ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

На правах рукописи

Петушков Григорий Валерьевич

ФУНКЦИОНАЛЬНАЯ АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ КЛАСТЕРНОГО ТИПА НА ОСНОВЕ ИСПОЛНИМЫХ МОДЕЛЕЙ И ПРАВИЛ ПРЕДМЕТНОЙ ОБЛАСТИ

Специальность 2.3.2. Вычислительные системы и их элементы
(технические науки)

ДИССЕРТАЦИЯ

на соискание ученой степени
кандидата технических наук

Научный руководитель
доктор физико-математических
наук, профессор, академик РАН
Сигов Александр Сергеевич

Москва 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. Кластеризация – это одно из самых современных направлений в области создания вычислительных систем. Появление и развитие кластерных вычислительных систем обусловлено прогрессом в области сетевых технологий, повышенным спросом на доступные высокопроизводительные вычисления, растущим внедрением методов искусственного интеллекта и машинного обучения. Кластерные вычислительные системы, как правило, отличаются архитектурной простотой, обусловленной регулярностью и однородностью, разделением аппаратуры, связью между узлами в ограниченной окрестности. Как правило, вычислительный кластер определяется как разновидность параллельной или, реже, распределенной вычислительной системы, когда взаимосвязанные компьютеры образуют единый вычислительный ресурс, рассматриваемый пользователями с позиций единого системного образа.

Условно в состав вычислительных систем кластерного типа можно отнести грид-системы, ранние суперкомпьютеры и однородные вычислительные системы, функциональные архитектуры которых очень похожи и по функциональности относятся к классу систем для высокопроизводительных вычислений. Кластерную архитектуру имеют, например, российские суперкомпьютеры «Червоненкис», «Галушкин» и «Ляпунов», используемые в дата-центрах Яндекса. В решении проблемы обеспечения высокой производительности мультипроцессорных вычислительных систем большое значение имеет завершение отечественного проекта создания коммуникационной сети «Ангара», а также других коммуникационных структур, разработанных в России.

Вычислительным системам кластерного типа «родственны» ранние суперкомпьютеры – однородные и распределенные вычислительные системы, разработанные в разное время в Академгородке Сибирского отделения Академии наук РАН. В трудах ученых исследована организация параллельных вычислений в многомодульных системах, где в качестве модулей используются элементарные машины, обладающие возможностями хранения, переработки и транспортировки данных. Разработана первая в мире программно-реконфигурируемая ВС «Минск-222», созданы мини-ВС МИНИМАКС и СУММА, семейство микро ВС МИКРОС, МИКРОС-2 и МИКРОС-Т.

Существенный вклад в развитие высокопроизводительных вычислительных систем внесли Н. Ф. Бахарева, О. М. Брехов, М. В. Бобырь, В. В. Воеводин, Вл. В. Воеводин, Б. А. Головкин, В. К. Левин, А. В. Каляев, И. А. Каляев, В. С. Князьков, Б. В. Костров, В. П. Корячко, М. Г. Курносов, А. М. Ларионов, Д. А. Перепелкин, А. С. Сигов, А. С. Симонов, В. Н. Тарасов, В. С. Титов, В. В. Топорков, А. Г. Финогеев, В. Г. Хорошевский, J. J. Dongarra, M. J. Flinn, B. A. Forouzan, I. Foster, G. C. Fox, J. L. Hennessy, M. van Steen, A. Tanenbaum.

Вопросам формализации параллельных и распределенных компьютерных систем посвящены работы О. Л. Бандман, В. А. Вальковского, Н. П. Вашкевича, В. А. Горбатова, В. Н. Дубинина, Д. В. Жевнерчука, В. Е. Котова, В. П. Кулагина, Л. С. Ломакиной, П. П. Макарычева, Д. В. Пащенко, И. Г. Сидоркиной, А. А. Шалыто, P. J. Antsaklis, E. Boerger, Y. Gurevich, M. V. Iordache, R. Janicki, R. M. Keller, D. J. Kuck, A. Pritsker.

Кластерная архитектуру, например, имеет вычислительная система Blue Gene/L (фирма IBM) с производительностью до 1 петафлопс – это масштабируемый суперкомпьютер, который может состоять из более, чем 65536 вычислительных процессоров. Примером крупного кластера, используемого в астрофизических расчетах является, например, высокопроизводительный компьютерный кластер DEGIMA, используемый в Центре передовых вычислений Университета Нагасаки (Япония). Система состоит из 144-узлового кластера ПК, соединённых через интерфейс InfiniBand. В целом система содержит из 144 центральных процессоров и 576 графических процессоров.

Современные вычислительные кластеры строятся на основе архитектурных решений в области межузловых коммуникационных сетей, позволяющих создавать сетевые инфраструктуры, как коммерчески доступные, так и высшего диапазона производительности типа Infiniband (Mellanox, США), OmniPass (Intel, США), Ангара (РФ), 6D-тор (Fujitsu, Япония), Sugon (Китай), линейки коммутаторов CRAY высшего диапазона производительности (США) и др.

Появление кластеров, предназначенных для высокопроизводительных вычислений (HPC-кластеров), явилось одним из стимулов, наряду с грид-системами, для развития облачных вычислений. Облачные, кластерные и грид-

вычисления имеют ряд общих черт и используются для вычислений, предоставляемых как коммунальные услуги или сервисы. Кластерные, грид и облачные вычисления имеют глубокие эволюционные связи, однако процессы управления и особенности применения различаются. В настоящее время получает распространение облачная технология HPC-as-a-Service – высокопроизводительные вычисления как сервис, что в существенной степени расширяет границы применимости вычислительных кластеров и делает их ресурсы доступными для многих категорий пользователей.

Общепринято считать, что функциональность вычислительных систем кластерного типа во многом обусловлена не только спецификой операционных систем, но и программным обеспечением промежуточного типа, а также приложениями. На развитие вычислительных систем кластерного типа повлияла также широко известная модификация существующих операционных систем – так называемое распределенное пространство процессов Beowulf (BProc: Beowulf Distributed Process Space). Как известно, основное различие кластеров и грид-систем состоит в том, что в кластерах чаще всего организуется централизованное управление ресурсами, в том числе посредством произвольного компьютера кластера, а в грид-системах обычно используется распределенное управление ресурсами.

Задача использования преимуществ кластерных вычислений за счет выбора наиболее подходящей модели организации функциональной архитектуры является **актуальной**, поскольку ее решение позволит осуществлять предметную ориентацию вычислительных кластеров на конкретного пользователя, использующего кластер как конечный продукт, не требующий значительной модификации его функциональной архитектуры при внедрении. В диссертационной работе предлагается разрешать проблемную ситуацию за счет использования новых методов организации функциональной архитектуры вычислительных систем кластерного типа на основе исполнимых моделей и правил предметной области. В качестве интегрирующих компонент предлагается использовать известные в международной практике событийно-ориентированный подход (EDA – Event-Driven Architecture) [109], основанный на событиях, происходящих при работе приложений и управляющих программ в вычислительном кластере, а также метод разработки систем, основанный на использовании правил предметной области (DDD – Domain-Driven

Design) [110]. Рассматриваемые с позиций искусственного интеллекта, основанного на знаниях и правилах, подобные модели представляют собой сценарии действий, содержащие как процедурную, так и декларативную составляющие модели представления знаний. Декларативная составляющая задается при помощи исчисления предикатов первого порядка, а процедурная, или императивная, – алгебраическими моделями алгоритмов и частичными конечными автоматами.

Объектом исследования является функциональная архитектура вычислительных систем кластерного типа.

Предметом исследования являются исполнимые модели, методы и алгоритмы, определяющие функциональную архитектуру вычислительных систем кластерного типа на основе правил и знаний о предметной области.

Целью исследования является повышение эффективности и отказоустойчивости вычислительных систем кластерного типа за счет совершенствования процессов управления на основе новых моделей, методов и алгоритмов, учета требований и свойств предметной области.

Для достижения поставленной цели необходимо решить следующие основные **задачи**:

– провести анализ современного состояния, специфики использования, целей и задач совершенствования функциональной архитектуры вычислительных систем кластерного типа;

– определить подходы к разработке функциональной архитектуры, основанной на исполнимых моделях, знаниях и правилах предметной области, назначении вычислительных кластеров;

– разработать новые исполнимые модели и методы совершенствования вычислительных систем кластерного типа, определяющие их функциональную архитектуру, масштабируемость, основанные на анализе событий, происходящих при работе приложений и управляющих программ;

– разработать логико-вероятностные и исполнимые логико-алгебраические модели на основе методологического подхода к макро- и микроанализу функциональной архитектуры вычислительных систем кластерного типа;

– разработать сетевые автоматные, вероятностные автоматные и логико-алгебраические исполнимые модели функционирования вычислительных си-

стем кластерного типа как основы для создания управляющих программ, иллюстрирующих работу вычислительных кластеров при последовательно-параллельной и параллельно-конвейерной обработке заданий;

– разработать метод и модель функциональной организации масштабируемых вычислительных систем кластерного типа с повышенной отказоустойчивостью.

Методы исследования. При решении поставленных задач используются методы математической логики, теории графов, алгебры алгоритмов, теории автоматов и сетей Петри. При проведении экспериментов и проверке предложенных решений использованы методы статистического моделирования.

Научная новизна исследования заключается в следующем.

1. На основе проведенного анализа современного состояния и развития архитектуры вычислительных систем предложена концепция организации функциональной архитектуры вычислительных систем кластерного типа, *отличающаяся* базированием на исполнимых моделях, знаниях и правилах предметной области и *позволяющая* осуществлять предметную ориентацию вычислительных кластеров на конкретного пользователя, использующего кластер как конечный продукт.

2. Предложен метод разработки функциональной архитектуры вычислительных кластеров, основанный на исполнимых логико-алгебраических операционных моделях, отличающийся от известных тем, что он основан на знаниях и правилах предметной области, на назначении вычислительных кластеров и на анализе событий, происходящих при работе приложений и управляющих программ, и позволяющий создавать новые исполнимые модели для совершенствования вычислительных систем кластерного типа, определяющие их функциональную архитектуру, масштабируемость и соответствие предметной области.

3. Предложены новые исполнимые модели вычислительных систем кластерного типа на макроуровне, *позволяющие* расширить диапазон их функциональности и применимости: логико-алгебраические операционные и статистические модели «Круглый стол», «Резидент-агенты», «Собрание 1», «Собрание 2» и *отличающиеся* от известных ориентацией на современные диалого-

вые параллельные взаимодействия при обработке данных в больших социальных сообществах пользователей при помощи масштабируемых приложений, реализуемых кластером.

4. Разработаны новые детализированные автоматные, вероятностные автоматные и логико-алгебраические операционные модели вычислительных систем кластерного типа на микроуровне, *отличающиеся* исполнимым характером и *позволяющие* произвести проектирование приложений на уровне сетевых операций и дать оценку производительности кластера в целом.

5. Предложен метод повышения производительности вычислительных кластеров при выполнении сетевых приложений, *отличающийся* развертыванием операторов распределенного алгоритма на различных узлах кластера, и *позволяющий* при реализации выполнять вычисления, определяемые передачами управления и данных между узлами в параллельно-конвейерном режиме.

6. Предложена исполнимая логико-алгебраическая операционная модель функциональной организации вычислительных систем кластерного типа с повышенной отказоустойчивостью, *основанная* на динамическим резервировании и многократной замене узлов и *позволяющая* при ее реализации обеспечить отказоустойчивость кластера в условиях сильной деградации, обусловленной внутренними и внешними причинами.

Практическая значимость результатов исследования заключается в следующем.

1. Предложенная концепция организации функциональной архитектуры вычислительных систем кластерного типа *позволяет* расширить диапазон их применимости за счет ориентации на предметную область пользователя, использующего вычислительный кластер как конечный продукт с расширяемой функциональностью.

2. Предложенный метод разработки функциональной архитектуры вычислительных кластеров, основанный на учете знаний и правил предметной области, позволяет разрабатывать логико-алгебраические операционные модели, исполнительный характер которых предоставляет возможность учета и реализации событий, планируемых при работе создаваемых масштабируемых сетевых приложений и управляющих программ.

3. Предложенные и реализованные новые исполнимые модели вычислительных систем кластерного типа на макроуровне *позволяют* расширить диа-

пазон их функциональности и применимости: логико-алгебраические операционные и статистические модели «Круглый стол», «Резидент-агенты», «Собрание 1», «Собрание 2». Проведенные статистические эксперименты с данными моделями позволяют оценить характеристики производительности при различных уровнях параллельной обработки данных в кластерах, выбрать число узлов кластера и портов коммутатора. Время выполнения статистических моделей при 1000 – 10000 прогонах занимает не более 2 – 25 секунд, что позволяет оперативно получать характеристики производительности вычислительных кластеров и других вычислительных систем с похожей системной и функциональной архитектурой.

4. Разработанные новые автоматные, вероятностные автоматных и логико-алгебраические операционные модели вычислительных систем кластерного типа на микроуровне, *позволяют* учитывать особенности работы приложений на уровне сетевых операций и давать оценку производительности кластера в целом при реализации последовательно-параллельных приложений. Статистические эксперименты продемонстрировали полное соответствие полученных характеристик производительности для базовых примеров закону Амдала.

5. Предложенный метод повышения производительности вычислительных кластеров при выполнении сетевых приложений *позволяет* повысить их пропускную способность при развертывании операторов распределенного алгоритма на узлах кластера и выполнять вычисления, определяемые передачами управления и данных между узлами в параллельно-конвейерном режиме. Статистические эксперименты продемонстрировали повышение пропускной способности в 2-5 раз.

6. Предложенная исполнимая логико-алгебраическая операционная модель функциональной организации вычислительных систем кластерного типа с повышенной отказоустойчивостью, основанная на динамическим резервировании и замене узлов, *позволяет* обеспечить отказоустойчивость кластера в условиях сильной деградации, обусловленной внутренними и внешними причинами. Статистические эксперименты показали, что работоспособность кластера сохраняется при его деградации до одного узла с соответствующим понижением производительности. Результат приведенного анализа базового примера показал, что в условиях сильной, но не стопроцентной, деградации вычислительного кластера было не только обеспечено выполнение разбитого

на части задания, но и достигнуто ускорение вычислений в 5,33 раза за счет частичного сохранения параллельного режима. Модель расширена на случай анализа отказоустойчивости систем распределенных вычислений и волонтерских кластеров с большим и нестабильным числом узлов (до 100-150 и более вычислительных узлов).

Внедрение результатов исследования. Результаты диссертационной работы внедрены: в ООО «ПРАЙМГРУП», (г. Красногорск) – в ходе разработки центров обработки данных (ЦОД) и выработки системно-технических решений для предприятий топливно-энергетического комплекса; в АО «Концерн «Моринформсистем-Агат» (Корпорация морского приборостроения, г. Москва) – при использовании предложенной в диссертации методики оценки надежности кластерной вычислительной системы для ЦОД; в АО «Научно-технический центр «Атлас», (г. Москва) – результаты диссертации применены в ходе выполнения опытно-конструкторской работы «Платформа», в том числе внедрены предложенные в диссертации метод и исполнимые логико-алгебраические операционные модели функциональной организации вычислительных систем с повышенной отказоустойчивостью, отличающиеся динамическим резервированием и заменой узлов и позволяющие обеспечить отказоустойчивость кластера в условиях сильной деградации, обусловленной внутренними и внешними причинами. Результаты работы внедрены также в Федеральном государственном бюджетном образовательном учреждении высшего образования «МИРЭА – Российский технологический университет» (г. Москва) – в рамках преподавания дисциплин «Вычислительные системы реального времени», «Промышленный Интернет вещей», «Архитектура вычислительных средств систем управления». Все виды внедрений подтверждаются актами.

Достоверность и обоснованность результатов работы. Обоснованность и достоверность полученных результатов определяется корректным использованием теории алгоритмов и конечных автоматов, исчисления предикатов первого порядка, теории вероятностей и математической статистики, методов компьютерного моделирования, внедрением результатов в учебный процесс вуза и в промышленности.

Соответствие паспорту специальности. Результаты исследований соответствуют п.1 «Разработка научных основ создания и исследования общих

свойств и принципов функционирования вычислительных систем и их элементов»; п. 3 «Разработка научных подходов, методов, алгоритмов и программ, обеспечивающих надежность, сбое- и отказоустойчивость, контроль и диагностику функционирования вычислительных систем и их элементов»; п. 7 «Разработка научных методов и алгоритмов организации параллельной и распределенной обработки информации, многопроцессорных, многоядерных, многомашинных и специальных вычислительных систем» паспорта научной специальности 2.3.2 – «Вычислительные системы и их элементы».

Основные положения, выносимые на защиту:

1. Концепция предметной ориентации вычислительных кластеров на конкретного пользователя, использующего кластер как конечный продукт.

2. Метод разработки функциональной архитектуры вычислительных кластеров, основанный на знаниях и правилах предметной области, а также на учете событий, происходящих при работе приложений и управляющих программ.

3. Исполнимые логико-алгебраические операционные и статистические модели вычислительных систем кластерного типа на *макроуровне*, позволяющие расширить диапазон их функциональности и применимости: «Круглый стол», «Резидент-агенты», «Собрание 1», «Собрание 2» и отличающиеся от известных ориентацией на диалоговые параллельные взаимодействия при обработке данных и сообщений.

4. Автоматные, вероятностные автоматные и логико-алгебраические операционные модели вычислительных систем кластерного типа на микроуровне, позволяющие при реализации учитывать особенности работы приложений на уровне сетевых операций и давать оценку производительности кластера в целом при реализации последовательно-параллельных приложений.

5. Метод повышения производительности вычислительных кластеров при выполнении сетевых приложений на основе определяемых между узлами передач управления и данных в сетевом параллельно-конвейерном режиме.

6. Исполнимая логико-алгебраическая операционная модель функциональной организации вычислительных систем кластерного типа с повышенной отказоустойчивостью, основанная на динамическим резервировании и многократной замене узлов, позволяющая при реализации обеспечить отказоустойчивость кластера в условиях сильной деградации.

Апробация результатов исследования. Результаты исследования докладывались автором на XVII научно-практической конференции «Современные информационные технологии в управлении и образовании» ФГБУ НИИ «Восход» 19 апреля 2018 г. в секции «Построение облачной инфраструктуры для электронного государственного управления», доклад на тему «Перспективы развития элементной и микропроцессорной базы компьютеров и вычислительных систем»; на IX международной научной конференции «ИТ-Стандарт-2019», 11-12 марта 2019 года, проводившейся в рамках «Недели российского бизнеса», доклад на тему «Прогнозирование надежности по отношению к отказам программного обеспечения для сложных программно-аппаратных систем». Результаты исследования также докладывались в мае 2018 года на 3-й НТК РТУ МИРЭА, на секции «Вычислительные машины, комплексы и сети», тема доклада – «Прогнозирование надежности серверов и вычислительных систем».

Публикации. По результатам исследования автором опубликованы 13 печатных работ, в том числе 10 статей в журналах из перечня ВАК (3 статьи категории К1, 5 статей категории К2) и 3 статьи в других изданиях, а также учебное пособие.

Личный вклад автора. Все выносимые на защиту результаты получены автором диссертации лично и отражены в работах: [164, 166, 167, 168, 169] – статьи выполнены в соавторстве с научным руководителем, принявшим участие в определении направления исследования и концептуальных подходов, формулировании актуальных задач и методологии их решения; работы [165, 170, 171] выполнены автором лично; в других изданиях, выполненных в соавторстве, включая учебное пособие [172] и статьи [173, 174, 175, 176, 177], личный вклад диссертанта состоял в определении проблемной ситуации, в выборе и применении методов решения задач.

Структура диссертации. Диссертационная работа состоит из введения, четырех глав, заключения, списка литературы и приложений с общим объемом 193 страницы, включая основной текст на 162 страницах (в том числе 25 рисунков и 17 таблиц), список литературы на 15 страницах из 177 наименований и двух приложений на 16 страницах, в том числе приведены примеры текстов исполнимых моделей и 4 акта о внедрении.

ГЛАВА 1. АНАЛИЗ ОСОБЕННОСТЕЙ ОРГАНИЗАЦИИ ФУНКЦИОНИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ КЛАСТЕРНОГО ТИПА

1.1. Проблемы развития функциональной и системной архитектуры и области использования вычислительных систем кластерного типа

Появление кластерных вычислительных систем обусловлено прогрессом в области сетевых технологий, чаще всего локальных. В настоящей главе на основе проведенного исследования разработок в области функциональной архитектуры кластерных вычислительных систем показана актуальность выполнения работ в данном направлении, что обусловлено увеличением спроса на доступные высокопроизводительные вычисления, растущим внедрением методов искусственного интеллекта и машинного обучения.

Вычислительные кластеры появились вслед за появлением компьютерных сетей и представляли собой компьютеры, связанные высокоскоростными сетевыми коммуникациями. При внедрении вычислительных кластеров преследовались цели ускорения получения результатов и повышения отказоустойчивости, что привело к применению высокоскоростных средств связи и практической реализации понятия «единого системного образа». Управляющие узлы кластера координируют распределение нагрузки, обнаруживают сбои в работе узлов и планируют возможные замены; во многих случаях также резервируются сетевые коммуникации между узлами [1, 12].

Кластерные вычислительные системы, как правило, отличаются архитектурной простотой, обусловленной регулярностью и однородностью, разделением аппаратуры, связью между узлами в ограниченной окрестности, топологией, не зависящей от размерности кластера, то есть горизонтальной масштабируемостью. Реже для вычислительных кластеров характерна вертикальная или иерархическая масштабируемость [1, 12, 13]. Параллелизм типа «коллективного решения» особенно удобен при использовании вычислительных

кластеров для решения задач, распадающихся на большое количество однотипных подзадач, каждая из которых решается независимо от остальных [14, 15, 16].

Обычно вычислительный кластер определяется как разновидность параллельной или распределенной системы, которая состоит из нескольких связанных между собой компьютеров, используемых как единый, унифицированный компьютерный ресурс [17, 18, 19].

Кластеры, предназначенные для высокопроизводительных вычислений, так называемые HPC-кластеры (HPC – High Performance Computing), позволяют выполнять большое количество однотипных вычислительных операций за короткое время. При этом вычислительный процесс распараллеливается не только между ядрами одного вычислительного узла, а по всему набору компьютеров, которые связаны высокоскоростными каналами передачи данных. Современный уровень развития технологий позволяет организовывать кластеры производительностью в десятки, сотни и тысячи терафлопс [20 – 26]. Кластерную HPC-архитектуру имеют, например, российские суперкомпьютеры «Червоненкис», «Галушкин» и «Ляпунов», используемые в дата-центрах Яндекса [27].

В решении проблемы обеспечения высокой производительности мультипроцессорных вычислительных систем большое значение имеет завершение отечественного проекта создания коммуникационной сети «Ангара», а также других коммуникационных структур, разработанных в России [28, 29, 30].

Известен целый ряд подобных проектов, перечисленных на сайте *parallel.ru* лаборатории параллельных информационных технологий Научно-исследовательского вычислительного центра МГУ, на котором собрана информация о параллельных вычислениях, технологиях параллельного программирования, высокопроизводительных компьютерах, метакомпьютинге, суперкомпьютерных центрах и др.

Технологии построения систем распределенных вычислений, таких как грид-систем, также можно условно отнести к кластерным технологиям.

Вычислительным системам кластерного типа «родственны» однородные и распределенные вычислительные системы, разработанные в разное время в Академгородке Сибирского отделения Академии наук РАН. В трудах ученых исследована организация параллельных вычислений в многомодульных системах, где в качестве модулей используются элементарные машины, обладающие возможностями хранения, переработки и транспортировки данных [31, 32, 33]. Разработана первая в мире программно-реконфигурируемая ВС «Минск-222», созданы мини-ВС МИНИМАКС и СУММА, семейство микро ВС МИКРОС, МИКРОС-2 и МИКРОС-Т [34, 35].

Отличительной особенностью современных вычислительных систем кластерного типа на основе локальных сетей с коммутаторами 2-го (канального) и 3-го (межсетевого) уровней от разработанных ранее вычислительных систем является б**о**льшая гибкость и масштабируемость архитектуры, обеспечивающая лучшие возможности реконфигурации на уровне связей между модулями, повышенная отказоустойчивость за счет возможного резервирования каналов передачи данных [36 – 40].

Для многих кластерных вычислительных систем в силу особенностей их построения на основе коммутаторов второго и третьего уровней характерна работа в пиринговом режиме, когда вычислительный узел может работать как в режиме клиента, так и в режиме сервера. В подобных системах узлы кластера могут обмениваться ресурсами напрямую, без посредника-сервера, либо каждый из узлов может выполнять функции сервера или управлять работой кластера. Возможет также такой пиринговый режим работы кластера, при котором функции центрального сервера сведены лишь к контролю за потоками информации и временному размещению данных [41 – 44].

К числу впервые реализованных вычислительных кластеров принадлежит, например, «Беовульф» (NASA, США) – компьютерный кластер, состоящий из идентичных компьютеров потребительского класса, объединённых в небольшую локальную сеть с установленными библиотеками и программами, которые позволяют совместно использовать ресурсы [45]. Для параллельной

обработки информации обычно используется интерфейс передачи сообщений (MPI) или параллельная виртуальная машина (PVM) [38, 46].

Кластерная архитектуру одной из первых также получила вычислительная система Blue Gene/L (фирма IBM) с производительностью до 1 петафлопс; это масштабируемый суперкомпьютер, который может состоять из более, чем 65536 вычислительных процессоров [47].

Примером крупного кластера, используемого в астрофизических расчетах является, например, высокопроизводительный компьютерный кластер DEGIMA, используемый в Центре передовых вычислений Университета Нагасаки (Япония) [48]. Система состоит из 144-узлового кластера ПК, соединённых через интерфейс InfiniBand. Каждый узел состоит из процессора Intel Core i7 920 с тактовой частотой 2,66 ГГц, двух видеокарт GeForce GTX295, 12 ГБ памяти DDR3-1333 и Mellanox MHES14-XTC SDR InfiniBand-адаптера на материнской плате MSI X58 pro-E. Каждая видеокарта оснащена двумя графическими процессорами GT200. В целом система состоит из 144 центральных процессоров и 576 графических процессоров [48].

Технологии Infiniband и RoCE используются для высокоскоростного обмена данными в вычислительных кластерах [49, 50]. Специализированная сетевая архитектура Infiniband может достигать пропускной способности до 800 Гбит/с.

Архитектура современных кластеров строится на основе архитектурных решений в области межузловых коммуникационных сетей, позволяющих создавать сетевые инфраструктуры, как коммерчески доступные, так и высшего диапазона производительности типа Infiniband (Mellanox, США), OmniPass (Intel, США), Ангара (РФ), 6D-тор (Fujitsu, Япония), Sugon (Китай), линейки коммутаторов CRAY высшего диапазона производительности (США) и другие [51 – 54].

Технология создания кластеров на базе высокоскоростных коммутаторов доступна многим компаниям в разных странах; в настоящее время более 50 высокопроизводительных кластеров на коммутаторах входят в верхнюю

часть списка Top 500 [55]. Многие системы распределенных вычислений могут быть отнесены к кластерам. Отличительной особенностью таких кластеров, являющихся частями больших вычислительных систем и сетей, является относительно невысокая доступность узлов-компьютеров, которые в силу разных причин подключаются и отключаются в процессе работы. В это случае переменность конфигурации компенсируется подключением дополнительных узлов.

Естественная распределенность развертывания узлов кластера часто может быть связана с его реализацией на базе локальной или глобальной сети с предметной ориентацией на распределенные предприятия, филиалы, офисы, производственные участки и др. В первом случае кластер может быть развернут в среде, где каждый узел должен находиться рядом с управляемым объектом или с другим источником информации. Во втором случае кластер может выполнять функции так называемого метакомпьютера, узлы которого распределены в глобальной вычислительной сети, например, в Интернете. К крупномасштабным Интернет-кластерам может быть отнесена открытая грид-платформа BOINC (Berkeley Open Infrastructure for Network Computing) [56], предназначенная для «волонтерских» научных вычислений; распределенная сеть BOINC может содержать до миллиона и более активных пользовательских компьютеров, на базе которых организуется коллективный вычислительный кластер.

Несмотря на схожесть системных архитектур, области применения «сильно распределенных» и «сильно связанных» вычислительных систем различны. Однако это не исключает использования схожих алгоритмов взаимодействия узлов и серверов с пользователями. Например, параллелизм типа «коллективного решения», когда решаемая задача состоит из множества решаемых независимо друг от друга однотипных подзадач, удобен как для «сильно распределенных» (грид-системы, метакомпьютеры, волонтерские системы) и для «сильно связанных» вычислительных систем (кластеры и суперкомпьютеры). Однако при наличии подзадач, «сильно связанных» по управлению и

данным, более эффективным является использование «классических», или «эталонных» вычислительных кластеров и суперкомпьютеров. Отметим, что многие мощные суперкомпьютеры имеют архитектуру, сходную с кластерной.

В последнее время большое внимание уделяется так называемым системам высокой готовности (англ. High Availability Systems), от которых требуется минимизация времени простоя. При этом обычно выделяют четыре уровня готовности [57]:

Уровень готовности, %	Макс. время простоя	Тип системы
99,0	3,5 дня в год	Обычная (Conventional)
99,9	8,5 часов в год	Высокая надежность (High Availability)
99,99	1 час в год	Отказоустойчивая (Fault Resilient)
99,999	5 минут в год	Безотказная (Fault Tolerant)

Соблюдение этих уровней готовности особенно требуется от корпоративных кластерных систем. При подготовке вычислительного кластера к использованию на предприятии или учреждении возникает проблемная ситуация выбора приложений, для которых использование кластера будет приемлемым или наиболее эффективным по критерию производительности. Поэтому решение подобной проблемы является актуальной научной задачей.

1.2. Обзор известных функциональных и системных архитектур вычислительных систем кластерного типа

При соединении машин в кластер объединение компьютеров производится при помощи сетевых технологий на базе шинной архитектуры или коммутатора, что обусловило увеличения числа приобретенных или арендованных в качестве облачных сервисов вычислительных кластеров [1]. По прогнозам ряда маркетинговых компаний ожидается, что рынок кластерных вычислений вырастет с 67,59 млрд долларов в текущем году до 102,4 млрд долларов к 2032 году [58].

Постоянно расширяется сфера применения кластеров при организации информационных и предметно ориентированных систем, применяемых для сбора, обработки и последующего анализа данных. Вместе с тем ограниченность простых однородных систем кластеров усложняет создание систем, обеспечивающих высокий уровень структурно-функциональной динамики и эффективную проблемную ориентация на основе создания программного обеспечения промежуточного уровня *middleware*.

Функциональная архитектура вычислительных кластеров основана на согласованной работе следующих компонент: системы управления потоком работ; системы мониторинга кластера; библиотек для параллельной работы; средств для управления кластером; глобального пространства процесса, связывающего воедино все узлы кластера; системы управления ресурсами; сетевой (возможно, параллельной) файловой системы; сетевых, в том числе облачных сервисов, обеспечивающих доступ к кластеру многих пользователей [49]. Дополнительные сведения о существующих программных пакетах в кластерных системах приведены в работах [4, 60, 61].

К российским кластерным проектам относятся: МВС-100К установленный в Суперкомпьютерном Центре РАН; суперкомпьютер «Ломоносов», установленный в НИВЦ МГУ в рамках проекта СКИФ [64]. Суперкомпьютеры «Яндекса» – «Червоненкис», «Галушкин» и «Ляпунов», созданные компанией «Яндекс», также имеют кластерную архитектуру [65]. Суперкомпьютеры Яндекса работают на графических ускорителях NVIDIA A100 (GPU NVIDIA A100 с тензорными ядрами) с системой связи InfiniBand на базе коммутаторов Mellanox [66].

Интенсивное развитие приложений на основе машинного обучения и искусственного интеллекта определило необходимость обучения большого количества моделей. В настоящее время одним из самых мощных в мире является суперкомпьютерный кластер Colossus на базе графических процессоров Nvidia. Этот кластер теоретически может достичь производительности около

497.9 экзафлопс (497 900 000 терафлопс), устанавливая новые стандарты в суперкомпьютерной мощности; целью компании xAI является увеличение числа графических ускорителей (GPU – Graphics Processing Unit) в Colossus до 1 миллиона в ближайшие годы [2]. В настоящее время суперкластер xAI начал работать в режиме обучения системы искусственного интеллекта большой языковой модели (LLM – Large Language Model) с использованием более двухсот тысяч графических процессоров Nvidia H100, H200 и GB200, оптимизированных для решения задач глубокого обучения нейронных сетей. Сеть кластера основана на высокоскоростном коммутаторе Nvidia Spectrum-X Ethernet с пропускной способностью до 800 Гбит/с [3].

Функциональная архитектура вычислительных кластеров основана на согласованной работе следующих компонент: системы управления потоком работ; системы мониторинга кластера; библиотек для параллельной работы; средств для управления кластером; глобального пространства процесса, связывающего воедино все узлы кластера; системы управления ресурсами; сетевой (возможно, параллельной) файловой системы; сетевых, в том числе облачных сервисов, обеспечивающих доступ к кластеру многих пользователей [1].

Предполагается, что текущие проблемы в области высокопроизводительных вычислений сохранят свою актуальность и в будущем: потребность в дальнейшем значительном увеличении параллелизма и скорости передачи данных; развитие архитектуры и технологии высокопроизводительных вычислений; тенденция к выходу рабочих процессов и вариантов использования за пределы центров обработки данных; существование множества мощных научных и промышленных факторов; переход от высокопроизводительных вычислений как изолированных систем к высокопроизводительным инфраструктурам [4].

Важный шаг в развитии науки и промышленности связан с разработкой и внедрением суперкомпьютера с AI-способностями «ELBJUWEL» (AI – artificial intelligence) [5]. Усилия разработчиков направлены на создание уникальной инновационной платформы, которая объединит экспертизу в области

AI и высокопроизводительных вычислений. Описанию потребностей в высокопроизводительных вычислениях при решении задач машинного обучения посвящены работы [6, 7, 8].

Следующей проблемой, с которой сталкиваются суперкомпьютерные центры, является неэффективное использования ресурсов для высокопроизводительных вычислений при решении некоторых вычислительных задач. Такие задачи могут блокировать ценные вычислительные ресурсы и замедлять вычисления других пользователей. Для решения этой проблемы в НИУ ВШЭ для высокопроизводительного вычислительного кластера «cHARISMa» разработана система мониторинга задач, автоматически формирующая выводы об их производительности [9]. Этим университетом накоплен богатый опыт использования суперкомпьютерного комплекса на базе кластера «cHARISMa» при решении задач различными категориями пользователей. Среди этих задач, например, задачи поиска, анализа и прогнозирования данных в социальных сетях [10], исследование моделей машинного обучения для прогнозирования рисков основных сердечно-сосудистых событий у пациентов с инфарктом миокарда и различными генотипами [11] и многие другие.

Многие вопросы, связанные с необходимыми рядовым пользователям и организациям вычислительными ресурсами, возникают в связи с организацией и использованием вычислительных кластеров. В работах [43, 44] отмечаются недостатки кластерных вычислительных систем. Некоторые из отмеченных недостатков находятся в противоречии с преимуществами, что объясняется спецификой предприятий и пользователей: кластерами трудно управлять без опыта; при большом размере кластера будет трудно обнаружить, что что-то вышло из строя. Проблема при поиске неисправности возникает, поскольку потребитель имеет дело с единой сущностью, и не ясно при обнаружении неисправности, с каким из компонентов связана проблема.

К недостаткам кластерных вычислительных систем могут быть также отнесены следующие [42]: кластеры не для всех потребителей подходят для коммерческого и бизнес-использования, поскольку требуют специальных

навыков программирования, знания систем и языков программирования, которые не используются широко в коммерческих целях. От персонала требуются обладание специальными техническими навыками для работы и администрирования.

1.3. Инфраструктура и архитектура вычислительных систем кластерного типа

Большое число вычислительных кластеров средней и низкой стоимости создается на основе разного рода коммутаторов, в том числе коммутаторов Ethernet. Обычно с этой целью используется инфраструктура на базе локальной сети для последующего включения вычислительного кластера через коммутаторы L2 (подключение «классических» кластеров с аппаратными адресами на уровне доступа) или L3 (подключение с использованием IP-адресов на уровнях агрегации или ядра) [1, 62, 63].

На рис. 1.1 представлен вариант вычислительного кластера на базе инфраструктуры локальной сети. Данная сеть дооборудована частной сетью на основе коммутаторов Infiniband, Myrinet или Ethernet для взаимодействия рабочих станций, а также сетью хранения данных. Подобные кластеры иногда называются Cluster of Workstations (COW), то есть кластеры рабочих станций.

В представленном примере организации вычислительного кластера и его инфраструктуры трафики различных локальных сетей могут пересекаться, если это не затрудняет выполнение основной функции узлов кластера. Узлы кластера *Y1–Y16* (рабочие узлы) и *Y17–Y22* (резервные узлы) обрабатывают пользовательскую нагрузку; *Y0*, *R0* – управляющие (основной и резервный) узлы, отслеживающие состояние аппаратного и программного обеспечения кластера и принимающие действия по его перенастройке в связи с каким-либо событием, происходящим в кластере; *M1*, *M2* – общие резервные хранилища, в которых хранится информация, доступная всем узлам кластера через сервер или коммутатор и используемая ими для доступа к общим данным, в том числе

к данным о вышедшем из строя узле, которыми может воспользоваться резервный узел; *S1*, *S2* – серверы, доступные по общедоступной и клиентской сетям. Коммутатор уровня *L2* частной сети осуществляет обмен данными между узлами кластера, используя аппаратные *MAC*-адреса. По частной сети передаются командные сообщения, используемые узлами для проверки работоспособности, перенастройки и синхронизации кластера.

Коммутатор уровня *L3* общедоступной сети осуществляет обмен данными, используя межсетевые *IP* или аппаратные *MAC*-адреса. На уровне общедоступной сети виртуализируется доступ к кластеру, как к единой системе. Локальная сеть, построенная на основе коммутатора уровня *L2+* с добавленными функциями, обеспечивает доступ клиентов к кластеру. Наличие в инфраструктуре вычислительного кластера нескольких сетевых коммутаторов позволяет использовать три основных вида сетей: коммуникационный, транспортный и сервисный [68].

Для решения поставленных в настоящей работе актуальных задач – организации эффективной функциональной архитектуры кластеров за счет создания нового обеспечения прикладного класса и класса *middlware* важно ориентироваться на имеющееся развитое обеспечение: сервисы обработки сообщений – *message-oriented middleware*, сервисы, обеспечивающие аналитику больших данных и подключение к хранилищу данных – *data warehousing and big data analytics* и протоколы и продукты, обеспечивающие межпроцессные взаимодействия – *interprocess communications* [68].

Вычислительный кластер – это распределенная ВС, обычно построенная на базе серийно выпускаемого промышленностью компьютеров, связанных сетями Ethernet, Ангара, Myrinet, InfiniBand (обозначены условно на рис. 1.1) или другими относительно недорогими сетями. Такие кластеры принято относить к классу кластеров Beowulf [45]. Один из самых ранних примеров такой системы – Stone Soupercomputer. Еще одной характерной особенностью вычислительных кластеров является то, что главный функционал кластеров как

параллельных систем формируется на уровне приложений, связанном с уровнем кластерного промежуточного обеспечения *middleware* для взаимодействия с пользователем [1].

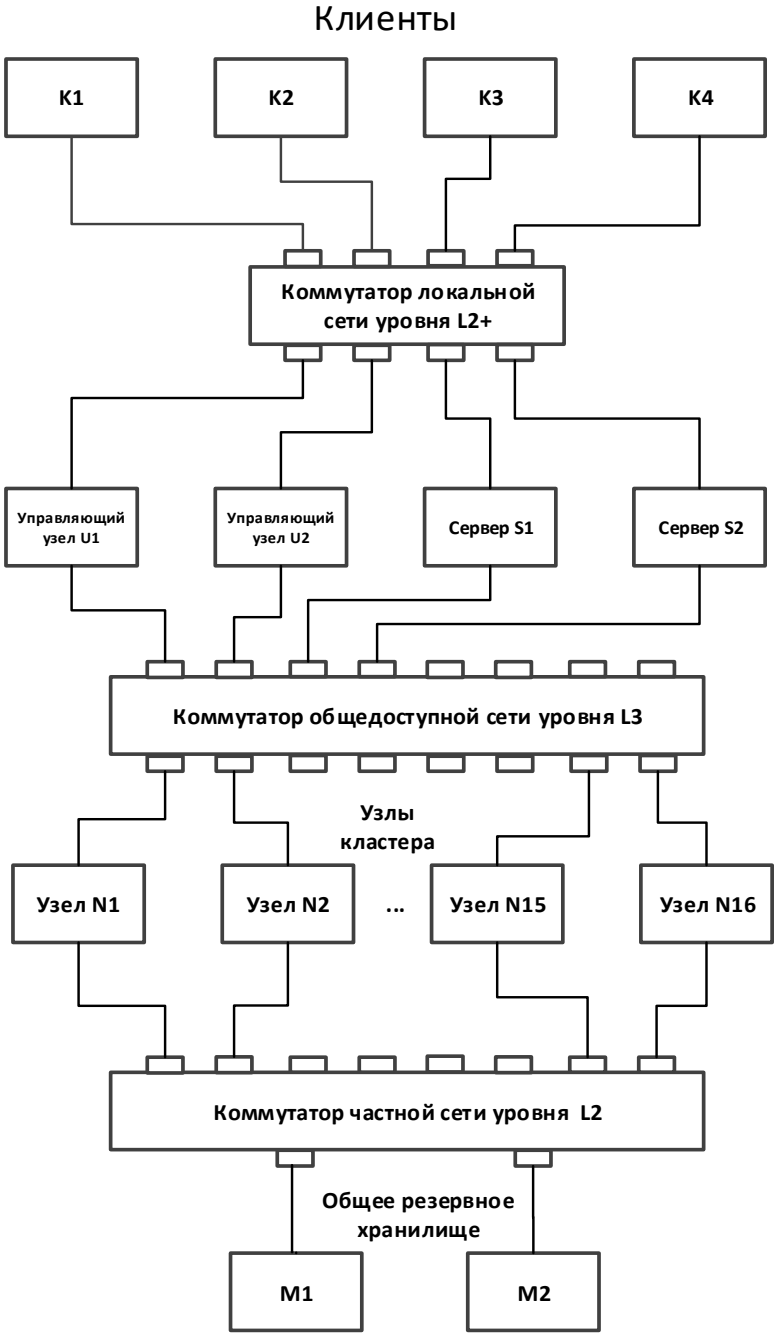


Рис. 1.1. Типовая архитектура вычислительного кластера, встроенного в инфраструктуру локальной вычислительной сети

Большинство вычислительных систем, называемых вычислительными кластерами, содержат компьютеры одного типа, что в существенной степени

упрощает управление системой. За счет объединения вычислительных ресурсов кластеры, как правило, обеспечивают более высокую производительность по сравнению с отдельно стоящими компьютерами. Наличие резервных компьютеров в кластере позволяет повысить его отказоустойчивость. Кластер удовлетворяет требованиям пользователя, если его программная система позволяет автоматически отключать неисправный компьютер и заменять его на резервный [58].

Примером отказоустойчивого кластера, предназначенного для высокоскоростной системы управления базой данных, является кластер IBM DB2. Системная архитектура кластера включает высокопроизводительные серверы и коммутаторы трех видов: коммутатор общедоступной локальной сети, быстродействующий системный коммутатора для рабочих узлов кластера и коммутатор сети SAN (Storage Area Network) накопителей информации.

Актуальной задачей является использование преимуществ кластерных вычислений по сравнению с грид-вычислениями за счет выбора наиболее подходящей модели организации функциональной архитектуры. Основное различие состоит в том, что в кластерах чаще всего организуется централизованное управление ресурсами, в том числе посредством произвольного компьютера кластера, а в грид-системах обычно используется распределенное управление ресурсами. Грид-системы строятся для разных групп пользователей и нередко предназначаются для использования в качестве коммунального ресурса, а специализированные кластеры чаще предметно ориентированы на конкретного пользователя [59].

Кластерные вычислительные системы могут в принципе иметь различные топологии. Например, отечественная сеть Ангара имеет топологию «многомерный тор» и создана на базе российской СБИС. Она сопоставима по своей функциональности, производительности и надежности с современными разработками в данной области (Cray, IBM, Mellanox) [51, 54, 63].

В работах [60, 61] проведено сравнение кластерных вычислительных систем с грид системами, пиринговыми и облачными системами; утверждается, что кластерные компьютерные системы при обработке большой рабочей нагрузке с большими наборами данных продемонстрировали приемлемые результаты. Они отличаются простотой реализации и управляемостью Отмеча-

ется, что промежуточное программное обеспечение *middleware* кластера в значительной степени способно обеспечить целостный и унифицированный образ системы. В работе [18] отмечается особая пригодность кластеров для реализации приложений баз данных, в силу использования однородного программного обеспечения. Следующие преимущества кластерных систем отмечаются в работе [19]: программное обеспечение автоматически устанавливается и настраивается, узлы кластера можно добавлять и ими легко управлять, поэтому его можно легко развернуть. Утверждается также, что это открытая система, и ее очень выгодно приобретать. Кластеры имеют много источников поддержки и поставок, отличаются скоростью и гибкостью; многие системы оптимизированы для производительности, и в них можно изменять конфигурации программного обеспечения.

Современные параллельные системы являются, по существу, гибридными и включают в свой состав параллельные компьютеры с многоядерными процессорами, общей памятью и сети с распределённой памятью для решения крупномасштабных задач [60]. Наборы инструментальных средств для параллельных вычислений позволяет программистам максимально эффективно использовать многоядерные процессоры, графические процессоры и вычислительные кластеры [61].

Известны другие успешные реализации проектов вычислительных систем кластерного типа, основанные на параллельных вычислениях. Например, недавно достигнутая производительность волонтерский сети Folding@Home стала выше, чем у всех суперкомпьютеров, достигнув производительности уровня в 2,4 эксафлопса [59] (1 эксафлопс означает 10^{18} операций с плавающей запятой в секунду). Более миллиона волонтеров-добровольцев, среди которых – российские участники, предоставили свободное время своих компьютеров для проведения удаленных расчетов, связанных с моделированием пространственной структуры S-белка коронавируса SARS-CoV-2 [59]. Проекты организации распределенных вычислений и обработки данных подобного уровня реализуются на основе технологий метакомпьютинга и грид-систем [60, 61, 64].

Вычислительные кластеры (High performance computing clusters, HPC) могут содержать до тысячи и более многоядерных процессоров. Многие кла-

стеры занимают высокие места в международном рейтинге TOP500. Программное обеспечение для объединения компьютеров в виртуальный суперкомпьютер открыло возможность быстрого создания кластеров [66].

Облачные, кластерные и грид-вычисления имеют ряд общих черт и используются для вычислений, предоставляемых как коммунальные услуги или сервисы. Кластерные, грид и облачные вычисления имеют глубокие эволюционные связи, однако процессы и особенности применение различаются. Грид-вычисления относятся в основном к классу распределенных вычислений. Облачные вычисления – в первую очередь это развивающаяся модель бизнес-вычислений, позволяющая клиентам получать доступ к облачным сервисам через Интернет. В этой связи роль кластерных и грид-вычислений еще более возрастает, поскольку их возможности становятся доступными практически с любого места. Однако вычислительные системы кластерного типа, ввиду их невысокой стоимости по сравнению с суперкомпьютерами, имеют и самостоятельное значение как для корпораций, так и для коллективов индивидуальных пользователей. Более того, достижения волонтерских, грид и кластерных вычислений становятся доступными для еще больших категорий пользователей [69, 70, 71].

1.4. Методы сравнительного анализа эффективности вычислительных систем кластерного типа

Вычислительные кластеры предназначены в первую очередь для реализации параллельных алгоритмов. Поэтому эффективность работы кластера определяется показателями эффективности реализуемых ими алгоритмов. К этим показателям обычно относят следующие [72]:

– ускорение относительно последовательного выполнения вычислений, то есть ускорение, достигаемое от использования параллельного алгоритма для нескольких процессоров, по сравнению с последовательным выполнением вычислений; среди параметров, определяющих вычислительную сложность решаемой задачи могут находится входные данные;

– эффективность использования процессоров, часто определяемая как средняя доля времени выполнения параллельного алгоритма, которую реально используют процессоры;

– стоимость вычислений, определяемая стоимостью машинного времени;

– масштабируемость: параллельный алгоритм считается масштабируемым (англ. *scalable*), если при росте числа процессоров он обеспечивает увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров; при анализе масштабируемости необходимо учитывать накладные расходы, например, на организацию взаимодействия процессоров, на синхронизацию параллельных вычислений. Эти хорошо известные важные показатели подробно описаны в учебной презентации [72].

Максимальное ускорение на практике ограничивается из-за наличия в решаемой задаче, наряду с распараллеленными участками, последовательных участков, которые не распараллелены по различным причинам. Например, одной из причин может быть необходимость выполнения системной программы, которая используется для организации параллельного режима (“*cobegin – coend*” или “*fork – join*”). Известным законом Амдала (англ. *Amdahl's law*) задается связь между ожидаемым ускорением параллельной реализации алгоритма и последовательной реализацией, в предположении, что количество процессоров и доля параллельной части независимы [73].

Объем данных, используемых при решении подзадачи, на которые разбивается исходная задача, может изменяться в зависимости от числа процессоров. Поэтому другим законом, называемым законом Густафсона – Барсиса (англ. *Gustafson – Barsis's law*), определяется оценка максимально достижимого ускорения выполнения параллельной программы, в зависимости от количества одновременно выполняемых потоков вычислений («процессоров») и доли последовательных расчётов. Закон Густафсона – Барсиса основан на предположении, что величину ускорения можно рассматривать и как увеличение объема выполненной задачи за постоянный промежуток времени; при

этом в качестве базового уровня используется гипотетический запуск задачи на одном компьютере [74].

Законами Амдала и Густафсона – Барсиса с различных точек зрения рассматривается достигаемое ускорение вычислений при переходе к параллельному решению задачи [75 – 88]. Усовершенствование данных законов, предпринимаемое рядом исследователей, основано на учете свойств реализуемых алгоритмов. Например, в предлагаемых формулах для вычисления ускорения вычислений могут присутствовать коэффициенты, характеризующие класс исполняемых алгоритмов в зависимости от связности по данным в предположении, что для алгоритмов с большей связностью по данным характерна большая интенсивность обменов между узлами кластера. При вычислениях коэффициента ускорения могут потребоваться параметры, характеризующих сложность алгоритмов, причем для определения этих параметров могут использоваться различные метрики. К недостаткам подобных подходов можно отнести «незаконченность» критериев оценки ускорения от распараллеливания оригинальных и разнообразных алгоритмов, требующих зачастую проведения уникального анализа для определения их сложности.

Решение задачи использования вычислительных систем кластерного типа может быть существенно облегчено, если она естественно распадается на ряд независимых подзадач, не связанных ни по управлению, ни по данным. Как было определено в названии диссертационной работы, к вычислительным системам кластерного типа можно отнести не только собственно вычислительные кластеры, но и при определенных допущениях некоторые грид-системы, системы с архитектурой «клиент-сервер» и системы волонтерских (добровольных) вычислений. Разнообразные системы подобного типа были подробно описаны в монографии [1], в материалах сайта parallel.ru [59] и во многих других работах [39 – 44, 57 – 61].

1.5. Рекомендации по выбору критериев для определения функциональной архитектуры вычислительных систем кластерного типа

Методика выбора структуры для кластерной вычислительной системы должна включать несколько взаимосвязанных этапов. Однако следует учесть факт, что оценки, базирующихся на законах Амдала и Густафсона – Барсиса, являются идеализированными и при практическом использовании требуют учета характеристик производительности вычислительного кластера и свойств реализуемых (различных) на нем приложений. Необходимо учитывать сложность реализуемых распределенных алгоритмов. В частности, замедляет выполнение параллельных приложений связность параллельных фрагментов по данным и управлению.

В плане обзора нельзя не отметить постановку проблемы определения оптимального количества узлов для кластера Kubernetes, решение которой может существенно повлиять на его производительность, доступность и стоимость. В статье [89, 107] рассмотрены факторы, влияющие на количество узлов, и даны рекомендации по принятию обоснованных решений при настройке кластеров Kubernetes. В данной статье перечислены факторы, влияющие на количество узлов, и сгруппированные по трем видам.

Рекомендации по производительности [89, 107]:

1. При выборе количества узлов в кластере Kubernetes важно оценить требования к производительности при рабочей нагрузке.

2. Каждый узел предоставляет кластеру такие ресурсы, как центральный процессор, память и хранилище.

3. Требования к наличию определенных ресурсов необходимо выполнять для эффективного решения задач пользователя, что крайне важно для обеспечения оптимальной производительности и масштабируемости.

Требования к доступности [89, 107]:

1. Высокая доступность является важным фактором при определении количества узлов. Кластер с достаточным количеством узлов обеспечивает избыточность и отказоустойчивость, сводя к минимуму риск сбоев в работе сервисов из-за отказа узлов.

2. Для поддержания надежности кластера важно соблюдать баланс между количеством узлов и требованиями к доступности. Для обеспечения высокой доступности в кластерах Kubernetes необходимо поддерживать избыточность узлов. Распределяя рабочие нагрузки между несколькими узлами, организации могут свести к минимуму влияние сбоев узлов на работу кластера.

Требования к экономической эффективности и использованию ресурсов [89, 107]:

1. Важно учитывать экономическую эффективность и рациональное использование ресурсов.

2. Добавление новых узлов в кластер увеличивает расходы на инфраструктуру, поэтому важно соблюдать баланс между производительностью, доступностью и стоимостью.

Данные рекомендации цитируются дословно, поскольку они относятся к большому числу разновидностей кластеров, а не только к кластеру Kubernetes. Возможность выбора необходимого числа узлов кластера и подключение при необходимости дополнительных узлов может упростить развертывание, масштабирование и управление приложениями. Однако следует учесть дополнительные факторы развития вычислительных систем кластерного типа: эволюция вычислительных кластеров и вычислительных систем другого типа привела к появлению грид и облачных систем, систем волонтерских, или метакомпьютерных, вычислений. Расширение возможностей вычислительных кластеров требует повышенной отказоустойчивости, масштабируемости и, соответственно, на их основе создание работоспособных высокопроизводительных вычислительных систем, распределенных грид-систем и волонтерских метакомпьютеров.

Отметим необходимость соблюдения баланса между такими противоположными требованиями как масштабируемость и рациональное использование ресурсов. Из анализа процитированной в настоящей главе литературы могут быть сделаны следующие заключения по этому поводу:

1. Вычислительные кластеры критичны к требованию масштабируемости, поскольку их узлы не комплектуются мониторами и обычно размещаются компактно в отдельном помещении с кондиционированием. Возможна также организация специального форм-фактора для размещения системных блоков, что ограничивает возможность масштабирования вычислительных ресурсов и,

следовательно, возможности распараллеливания вычислений.

2. На рисунке 1.1 настоящей главы была представлена типовая инфраструктура и распространенная архитектура вычислительного кластера с несколькими видами коммутаторов 2 и 3 уровней. Коммутатор частной сети Infiniband предназначен для обеспечения работы собственно кластера, а коммутатор Gigabit Ethernet может при необходимости обеспечивать использование ресурсов кластерных узлов многим пользователям организации, которым не всегда требуются параллельные вычисления. Поэтому представляет интерес построение гибридных структур, сочетающих возможности эксплуатации вычислительных и грид-кластеров в условиях работы в организации или учреждении типа вуза. Такие структуры как правило, должны иметь дополнительные вычислительные мощности, в том числе резерв для повышения отказоустойчивости.

Согласованную работу управляющих и рабочих узлов вычислительных кластеров обеспечивают следующие компоненты программного обеспечения, являющиеся компонентами функциональной архитектуры [14]:

1. Система управления потоком работ (Job Management System, JMS).

2. Система мониторинга кластера (Cluster Monitoring System, CMS).

3. Библиотеки для параллельной работы.

4. Средства для управления кластером, служащая также для обнаружения ошибок в его работе и для изменения его конфигурации (например, обнаруживает отказавшие и подключает резервные вычислительные узлы).

5. Глобальное пространство процесса, обеспечивающее координацию всех узлов кластера.

6. Система управления ресурсами, поддерживающая очередь задач и контролирующая доступ к ресурсам.

7. Сетевая (параллельная) файловая система, обеспечивающая доступ к данным со всех узлов.

8. Сетевые сервисы.

Компоненты 3–8 открыты для обновления или замены более эффективными для выполнения кластерных приложений. На основании проведенного обзора следует отметить, что главный функционал вычислительного кластера – параллельная работа узлов реализуется на уровне приложений и, частично, на уровне программного обеспечения *middleware* промежуточного класса

класса. Учитывая исследования, проведенные в настоящей работе, компоненты п. 3 расширяются таким образом: «библиотеки и системные программы для организации последовательной, последовательно-параллельной и параллельно-конвейерной работы. Состав компонент п. 8 расширяется так: «сетевые сервисы, в том числе средства реализации облачных HPC-сервисов (англ. HPC – High Performance Clusters)».

1.6. Современные тренды развития высокопроизводительных кластеров

Современные высокопроизводительные кластеры создаются в основном на базе трех технологий: с ориентацией на серверные процессоры (CPU – *Central Processing Unit*), на графические ускорители (GPU – *Graphics Processing Unit*) и на гибридные архитектуры с комбинированием CPU и GPU. Серверные процессоры – Intel Xeon, AMD EPYC и другие; в GPU-системах используются графические процессоры NVIDIA (A100, H100), AMD (Instinct) [90, 91, 92].

В настоящее время получает распространение облачная технология HPC-as-a-Service – высокопроизводительные вычисления как сервис [93], который может быть реализован на основе концепции виртуальных информационно-вычислительных лабораторий [94 – 100].

Отечественная компания «К2 Нейротех» создает способные к масштабированию суперкомпьютеры кластерного типа на основе мощных вычислительных ресурсов (с использованием как CPU, так и GPU), высокоскоростных сетевых соединений [103]. Ограниченные возможности к масштабированию имели многие вычислительные кластеры, приобретаемые из-за рубежа, поэтому задача производства масштабируемых по заказу пользователя в отечественной науке и промышленности стала актуальной.

Программно-аппаратные комплексы (ПАК-HPC) от компании «К2 Нейротех», выпускаемые в настоящее время (2024 г. – 2025 г.), обеспечивают производительность до 7,6 TFLOPS на сервер, имеют оперативную память до

8 ТБ DDR5, включают высокоскоростные сети Mellanox Infiniband и Ethernet до 100 Гбит/с. Постоянный мониторинг работы и состояния кластера обеспечивают системы мониторинга Zabbix и XDMoD [94, 95].

Один из суперкомпьютеров от компании «К2 Нейротех» поставлен в дата-центре на базе Новосибирского государственного университета. Семь высокопроизводительных серверов содержат 392 процессорных ядра. Обмен данными реализуется через коммутатор «Ангара», разработанный в «НИЦЭВТ» [105], обеспечивающий скорость передачи данных не менее 75 Гбит/с и . Предполагаемая пиковая производительность суперкомпьютера – не менее 47 TFLOPS; вместимость отказоустойчивого NFS-хранилища – не менее 40 Тбайт данных [106].

Программное обеспечение сети на базе отечественного коммутатора «Ангара» совместимо со следующими операционными системами: OpenSUSE/SLES 11 SP3/4, CentOS 6.0-7.3, ОС Astra Linux SE 1.3-1.5, ОС «Нейтрино» (QNX 6.5), ОС «Эльбрус», поддерживаются версии ядра Linux от 2.6.21 до 3.16.0. Поддерживаются следующие реализации библиотеки MPI: MPICH 3.0.4 (стандартная и оптимизированная), MPICH 3.2, OpenMPI 1.10. Работоспособность, надежность и производительность сетевого оборудования Ангара подтвердили организации РФЯЦ-ВНИИЭФ, ООО «Ниагара Компьютерс», АО «Т-Платформы», ЗАО «РСК Технологии», ИПМ им. М.В. Келдыша РАН, ИВМ РАН, ОИВТ РАН, ИДСТУ СО РАН, ФГУП НИИ «Квант», CADFEM-CIS, ТЕСИС, АО «МЦСТ» [106].

1.7. Выводы по 1-й главе

1. Проведенный обзор трудов в области построения и практического использования вычислительных систем кластерного типа показал, что сфера исследования, производства и применения этих систем увеличивается благодаря новым приложениям, связанным с обработкой больших данных, машинным обучением и развитием облачного доступа. Кластерные архитектуры исполь-

зуются при построении суперкомпьютерных систем; в учреждениях и организациях вычислительные кластеры используются как самостоятельные сущности, так и при подготовке задач для последующего их решения на суперкомпьютерных системах. Внедрение кластеров пониженной стоимости, так называемых кластеров-лоукостеров, позволяет организациям сократить расходы на IT-инфраструктуру. Масштабируемость, гибкость модульных аппаратных и программных решений для кластерных вычислений позволяет организациям уменьшать или увеличивать размер кластера в зависимости от изменяющихся потребностей. Поэтому задача исследования и реализации функциональной архитектуры вычислительных систем кластерного типа является актуальной.

2. Функциональная архитектура вычислительных кластеров основана на согласованной работе следующих компонент: системы управления потоком работ; системы мониторинга кластера; библиотек для параллельной работы; средств для управления кластером; глобального пространства процесса, связывающего воедино все узлы кластера; системы управления ресурсами; сетевой (возможно, параллельной) файловой системы; сетевых, в том числе облачных сервисов, обеспечивающих доступ к кластеру многих пользователей. Определено, что функциональная архитектура вычислительных кластеров, связанная с обеспечением параллельного выполнения заданий определяется в основном на уровне приложений и программного обеспечения промежуточного уровня. Анализ современных приложений показал, что для вычислительных систем кластерного типа характерно использование для решения задач вычислительного характера, требующего использование распараллеливаемых численных методов. Предложено расширить состав задач, определяющих функциональную архитектуры вычислительных систем кластерного типа, средствами для управления кластером при организации круглых столов, конференций, собраний, то есть задачами, требующими организации работы большого состава участников. Созданию и анализу результатов применения логико-алгебраических и логико-вероятностных исполнимых моделей макроуровня для вычислительных систем кластерного типа посвящена вторая глава.

3. Решения задач оценки производительности вычислительных кластеров имеют идеализированный и приближенный характер, требующей учета характеристик алгоритмов, сложность которых неизвестна с разной степенью чередования последовательных и параллельных участков. Поэтому поставлена задача на предварительном этапе решения подобных задач создавать и

исследовать автоматные, логико-алгебраические и логико-вероятностные исполнимые модели микроуровня для задания произвольной структуры приложений. Построению и исследованию моделей подобного типа посвящена третья глава.

4. Эволюция вычислительных кластеров и вычислительных систем другого типа привела к появлению грид и облачных систем, систем волонтерских, или метакомпьютерных, вычислений. Расширение возможностей вычислительных кластеров требует повышенной отказоустойчивости, масштабируемости и, соответственно, на их основе создание работоспособных высокопроизводительных вычислительных систем, распределенных грид-систем и волонтерских метакомпьютеров. В этой связи в последующих главах данного диссертационного исследования целесообразно решить задачи организации на основе исполнимых моделей коллективного обмена информацией между приложениями типа "*front-end*" при посредстве кластера, характерные для приложений, пока почти не свойственных современным вычислительным кластерам, но расширяющих сферу их применений.

5. При рассмотрении используются определения макроструктурного и микроструктурного уровней модели предметной области, следуя классической монографии известных советских ученых Э. В. Евреинова и Ю. Г. Косарева, намного опередившей свое время, [31]; на первом, макроструктурном уровне рассмотрения универсальных вычислительных систем различаются три основные части системы – элементарные машины, коммутаторы и каналы связи. На втором, микроструктурном уровне устанавливаются как неделимое целое элементы, из которых построены все макроэлементы, например, компоненты программного обеспечения.

6. Целесообразно разработать логико-алгебраические исполнимые модели функциональной организации вычислительных кластеров повышенной отказоустойчивости, которые являлись бы масштабируемыми и, соответственно, являлись бы масштабируемыми и реализуемые отказоустойчивые приложения.

7. На основании предыдущих пунктов выводов по первой главе сформулировано следующее резюме: на основе проведенного анализа современного состояния и развития архитектуры вычислительных систем предлагается концепция организации функциональной архитектуры вычислительных систем кластерного типа, отличающаяся базированием на исполнимых моделях,

знаниях и правилах предметной области и позволяющая осуществлять предметную ориентацию вычислительных кластеров на конкретного пользователя, использующего кластер как конечный продукт.

Примечание к п. 7. К исполнимым моделям относятся процедурные и, частично, декларативные модели искусственного интеллекта: системы продукций, сценарии, описываемые логико-алгебраическими операционными выражениями; многие дискретно-событийные модели: конечные автоматы, сети Петри, системы и сети массового обслуживания, системы моделирования дискретных событий, использующие генераторы псевдослучайных чисел, системы формализованных спецификаций типа алгебры алгоритмов и др. Многие из этих моделей, порознь или в комбинациях друг с другом, подходят для моделирования и спецификации распределенных вычислительных систем, в том числе вычислительных систем кластерного типа. Например, в настоящей работе предлагается использовать интегрированные модели на основе временных конечных автоматов с темпоральными событиями и переходами и систем моделирования дискретных событий, использующих генераторы псевдослучайных чисел; логико-алгебраические операционные выражения, интегрирующие алгебру алгоритмов с исчислением предикатов первого порядка.

Интеграция моделей искусственного интеллекта заключается в интерпретации переходов состояний в автоматах, в программах и в сетях Петри системами продукционного типа.

На все используемые варианты дискретно-событийных моделей даны ссылки по тексту диссертационной работы. В диссертации определены также некоторые методы, определяющие порядок построения дискретно-событийных моделей вычислительных систем кластерного типа. Применение этих методов иллюстрируется подробными примерами.

ГЛАВА 2. МЕТОДЫ ФОРМАЛИЗАЦИИ И ИССЛЕДОВАНИЕ ЛОГИКО-ВЕРОЯТНОСТНЫХ И ЛОГИКО-АЛГЕБРАИЧЕСКИХ ОПЕРАЦИОННЫХ МОДЕЛЕЙ ФУНКЦИОНАЛЬНОЙ АРХИТЕКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ КЛАСТЕРНОГО ТИПА

В настоящей главе построены имитационные логико-вероятностные и логико-алгебраические модели для ряда важных вариантов использования вычислительного кластера, проведены необходимые статистические эксперименты с данными моделями, давшие обоснования к реализациям соответствующему моделям программному обеспечению.

Предложен метод разработки функциональной архитектуры вычислительной системы кластерного типа, определяемой спецификациями в форме логико-вероятностных и родственных им логико-алгебраических моделей, что может ускорить подготовку кластера к эксплуатации в организации. Метод основан на интеграции предлагаемых моделей искусственного интеллекта и заключается в интерпретации переходов состояний в автоматах, в программах и в сетях Петри системами продукционного типа, а также на использовании логики предикатов первого порядка.

2.1. Логико-вероятностные и логико-алгебраические модели функциональной архитектуры вычислительных систем кластерного типа

При разработке моделей и приложений в настоящей главе предлагается использовать логико-вероятностный и логико-алгебраический подходы.

Первый подход предназначен для создания имитационных статистических моделей функционирования кластерных и других вычислительных систем. При создании статистических моделей используются программные генераторы псевдослучайных чисел с различными законами распределения, а также программы для сбора статистических данных о времени задержек, о загрузке компьютеров.

Второй подход основан на создании предварительных спецификаций для создания прототипного программного обеспечения для реальных приложений. В то же время ряд особенностей функционирования реального программного обеспечения в моделях не учтен, поэтому при программировании моделей следует учитывать возможность появления дополнительных задержек и передач управляющей информации.

В настоящей главе рассматриваются вопросы использования логико-вероятностного и логико-алгебраического подходов при разработке моделей и реальных приложений для вычислительных систем кластерного типа и других вычислительных систем. Логико-вероятностный и логико-алгебраический подходы соответствуют в совокупности методикам моделирования, получившим название «событийно-управляемое моделирование» (EDM – *Event-Driven Modeling*) [109] и разработок приложений на основе правил предметной области (DDD – *Domain-Driven Design*) [110].

Для лучшего запоминания этих задач также используются метафоры, но, естественно, имеются в виду задачи обработки данных и макроанализа производительности преимущественно кластерных вычислительных систем. Приведены примеры формализации модели «Круглый стол» о случайных парных взаимодействиях процессов в вычислительном кластере, модели «Резидент-агенты», посвященной формализации взаимодействия процесса подготовки заданий для кластера и последовательной загрузки агентов-исполнителей работой. Задачи «Собрание 1» и «Собрание 2» похожи на предыдущую, но в них задана параллельная и распределенная обработка информации. Анализ характеристик производительности моделируемых систем доведен до численных результатов и работающих приложений в кластере, построенном на основе коммутаторов 2 и 3 уровней.

На рис. 1.1 ранее была представлена распространенная структура вычислительной системы, включающей в свой состав вычислительный кластер. Как было сказано ранее, на основе коммутаторов локальных сетей различных уровней строятся вычислительные кластеры различной производительности –

от высокопроизводительных кластеров HPC до массовых недорогих кластеров, иногда называемых кластерами-лоукостерами, то есть кластерами низкой стоимости. Простота реализации обуславливает широкую распространенность кластеров вычислительных машин. Функциональная архитектура сетевых кластеров обычно строится на основе программного обеспечения различного уровня. Операционные системы узлов кластера не зависят друг от друга. Организация работы кластера начинается обычно с промежуточного уровня (уровня *middleware*) и уровня приложений. Работа важных приложений будет рассмотрена подробнее.

2.2. Выбор методологического подхода к макро- и микроанализу функционирования вычислительных систем кластерного типа

Имитационная, или поведенческая, модель вычислительного кластера должна быть основана на выборе концептуальной схемы. Концептуальная схема должна базироваться на определенном методологическом подходе. В рамках этого подхода описываются функциональные взаимосвязи системы, то есть ее функциональная архитектура. Общеизвестно, что методологический подход, выбранный исследователем, позволяет ему четко сформулировать описание системы [111]. Методологический подход в настоящей работе основан на выборе формальных моделей: логических исчислений, таких как исчисление высказываний и предикатов; исполнимых, или операционных моделей, таких как сети Петри и конечные автоматы; модели параллельных, конкурирующих и распределенных вычислений, которые относятся также к теоретическому и прикладному программированию [112, 113, 114]. К последнему относятся технологии автоматного программирования [115] и программирования на основе логико-алгебраических спецификаций [116, 117, 118].

В дальнейшем речь пойдет о *макро* и *микроанализе* функционирования вычислительных систем кластерного типа, поэтому проследим возникновение

этих терминов на примерах некоторых общенаучных и технических дисциплин. Средства вычислительной техники используются практически во всех сферах деятельности современного человека: привычными стали термины Интернет, Интернет вещей, большие данные, искусственный интеллект. Информатизация общества непрерывно растет, что в свою очередь требует постоянного развития компьютерной техники. Практически все отрасли науки и техники стали смежными по отношению к информатике и вычислительной технике.

Например, в экономике *микроанализ* исследует мотивы и принципы принятия субъектами решений, их взаимодействие [119]. *Макроанализ* позволяет исследовать экономику как систему связей между агрегатами – совокупностями экономических единиц, рассматриваемых как одна обобщенная единица [120].

В известной классической монографии [31] (авторы – Э.В. Евреинов и Ю.Г. Косарев) на *макроструктурном* уровне для универсальных вычислительных систем (УВС) различают элементарные машины, коммутаторы и каналы связи. На *микроструктурном* уровне рассматриваются структуры подсистем УВС, состоящие из микроэлементов, под которыми понимаются рассматриваемые как неделимое целое элементы, из которых построены все макроэлементы.

В монографии «Микроанализ производительности вычислительных систем» (автор – Б. Байцер) [121], определено, что «при *макроанализе* системные характеристики выводятся на основании поведения всей системы… при выполнении *микроанализа* исследования начинаются на самом нижнем уровне».

Указанные методы анализа положены в настоящей диссертационной работе в основу реализации функциональной архитектуры вычислительных систем кластерного типа на основе исполнимых моделей и правил предметной области.

2.3. Построение логико-вероятностной модели функционирования вычислительного кластера, реализующего равнодоступные парные взаимодействия (модель «Круглый стол»)

Рассматривается задача макроанализа производительности вычислительного кластера в следующей содержательной постановке. В кластере реализуется вычислительный процесс, сопровождаемый межузловым обменом запросами и затребованными данными. Как отправляющие запросы узлы (узлы-источники), так и узлы, принимающие запросы (узлы-приемники) выбираются в соответствии с заданным распределением вероятностей. Каждый запрос предваряется выполнением некоторой подготовительной программы на узле-отправителе. Ответ на запрос сопровождается выполнением некоторой программы на узле-приемнике запроса. Естественным ограничением является тот факт, что номера узла-источника и узла-приемника не могут совпадать. Запросы, подаваемые с одного и того же узла, могут повторяться, что может привести к задержкам при передаче и подготовке запроса. К задержкам также приводят обращения некоторых узлов-источников к одним и тем же узлам-приемникам. Таким образом в вычислительной системе кластерного типа будут взаимодействовать конкурирующие процессы, и образование очередей ожидания не нарушит работу системы.

Рассматриваемый режим работы кластера является важной разновидностью диалоговых взаимодействий в компьютерных сетях. На содержательном уровне постановки задачи можно привести метафорический аналог задачи о круглом столе. Согласно задаче «Круглый стол» в беседе на какую-то научную тему участвуют N ученых. Сразу после оглашения председателем вопроса, подлежащему обсуждению, каждый участник выбирает одного собеседника, готовит вопросы к нему, и затем вступает в беседу. В беседах могут участвовать только независимые пары ученых. Председатель переходит к оглашению следующего запроса только в том случае, когда все беседы завершены (число

«бесед» заранее ограничено). Результатом решения задачи должно быть экспериментально определенное распределение вероятностей (гистограмма) продолжительностей бесед, включающих также и время подготовки (обдумывания) предстоящей и состоявшейся впоследствии беседы.

Формализация решения задачи состоит в следующем построении логико-вероятностной модели функционирования вычислительного кластера, реализующего равнодоступные парные взаимодействия. Предлагается формализацию провести в два этапа. На первом этапе строится логико-вероятностная имитационная поведенческая модель, в которой понятие исполнимости в реальной системе ограничивается некоторыми модулями, работа которых задается распределением вероятностей некоторого интервала времени функционирования. На втором этапе формальное описание модели работа модулей уточняется, что позволяет называть модель логико-алгебраической, пригодной для последующей реализации на ее основе программного приложения, работающего в реальном вычислительном кластере.

Моделирование сложных крупномасштабных систем, к которым относится и моделирование кластерных вычислительных систем, представляет сложную задачу. Ее особенностью является большое число состояний, что затрудняет использование таких формализмов, как конечные автоматы и сети Петри. Поэтому для решения этой задачи выбрано построение моделей, основанных на исчислении предикатов первого порядка. Отличием от других логических моделей, использующих исчисление предикатов, является придание модели динамичности за счет использования операций модификации предикатов, определенных на множествах предметной области.

Рассмотрим полный орграф $G = (V, U)$, где V – множество вершин, $U = V^2$ – полное множество дуг, $V^2 = V \times V$. Введем S_x: $V_x \to \{\textbf{\textit{true}}, \textbf{\textit{false}}\}$ – унарный предикат, определенный на подмножестве вершин-источников запросов в кластерной вычислительной системе, S_y: $V_y \to \{\textbf{\textit{true}}, \textbf{\textit{false}}\}$ – унарный предикат, определенный на подмножестве вершин-приемников запросов в кластере;

подмножества $V_x \subset V$ и $V_y \subset V$ здесь имеют переменный состав и формируются в процессе моделирования. Истинные значения предикатов S_x и S_y, играющих соответственно роли характеристических функций для подмножеств V_x и V_y, будут соответствовать назначению соответствующих вершин из множества V на участие в передаче данных диалогов от источников к приемникам. Очевидно, что V_x и V_y, согласно постановке задачи, непересекающиеся подмножества ($V_x \cap V_y = \varnothing$) множества V. Предикаты S_x, S_y и множества V_x, V_y изменяются (эволюционируют) в процессе моделирования, причем на любом этапе моделирования множества V_x и V_y равномощны. Моделирование диалоговой системы «Круглый стол» состоит в формировании последовательности орграфов паросочетаний $H_0, H_1, H_2, \ldots, H_m$, имитирующих подключение и отключение участников диалога в кластерной вычислительной системе. Одновременно с этим производится сбор статистических данных, характеризующих загрузку и производительность системы. В теории графов паросочетание, или независимое множество дуг в орграфе, – это набор попарно несмежных дуг. Таким образом, орграфы паросочетаний будут содержать множества некоторых упорядоченных пар (x, y), где $x \in V_x$, $y \in V_y$, $x \neq y$. Задача моделирования упрощается тем, что x и y ($x \neq y$) выбираются из одного и того же множества V, то есть из множества участников диалога в кластерной вычислительной системе.

Орграф паросочетаний не может иметь петель, так как полагается, что участнику круглого стола нет необходимости в передаче сообщений самому себе. Если паросочетание орграфа покрывает все его вершины, то это соответствует связи максимального числа пар. Очевидно, что при четном числе вершин n число таких пар максимально и равно $n/2$. Число пар меньше данной величины в случаях, когда наблюдается конфликт запросов при обращении на передачу к одним и тем же приемникам.

Эволюцию орграфов паросочетаний при моделировании предлагается организовать при помощи следующего логико-вероятностного выражения, сочетающего в себе как декларативные, так и процедурные знания о предметной области:

$$L_0 = [(\exists_{\boldsymbol{Any}}\, x \in V) \neg S_x(x)]([(\exists_{\boldsymbol{One}}\, (x, y) \in V^2) \neg S_y(y)\ \&\ (x \neq y)]$$
$$(\{S_x(x) \leftarrow \boldsymbol{true}, S_y(y) \leftarrow \boldsymbol{true}, R(x, y) \leftarrow \boldsymbol{true}, Transfer(x, y) \leftarrow \boldsymbol{true},$$
$$Delay(x, y) \leftarrow \boldsymbol{true}, S_x(x) \leftarrow \boldsymbol{false}, S_y(y) \leftarrow \boldsymbol{false}, R(x, y) \leftarrow \boldsymbol{false},$$
$$Delay(x, y) \leftarrow \boldsymbol{false}, Transfer(x, y) \leftarrow \boldsymbol{false}\} \vee Ret) \vee Ret), \qquad (2.1)$$

где в квадратные скобки заключены условия успешного выполнения заданных операторов. В фигурные скобки заключены операции модификации предикатов. Квадратные и фигурные скобки являются элементами синтаксиса выражения (2.1) и доопределяют его операционную семантику. Оператор $(\exists_{\boldsymbol{Any}}\, x \in V) \neg S_x(x)$ реализует в модели псевдослучайный выбор будущей вершины x графа паросочетаний H при условии, что ранее эта вершина в граф не входила, то есть выбор реализуется успешно, если истинно высказывание $\neg S_x(x)$ при конкретном вхождении x. При выполнении данной операции используется генератор целочисленных псевдослучайных величин с заданным законом распределения; вид выбранного распределения вероятностей может задавать приоритетный выбор запросов от участников «круглого стола». При равномерном распределении моделируется равновероятный выбор запросов. Далее, оператор $\exists_{\boldsymbol{One}}\, (x, y) \in V^2$ осуществляет выбор кортежа (x, y) из декартова произведения $V^2 = V \times V$ с учетом того факта, что первый элемент кортежа x уже известен в результате предыдущего выбора и истинно высказывание $\neg S_y(y)$ & ($x \neq y$). Отметим, что операторы $\exists_{\boldsymbol{Any}}$ и $\exists_{\boldsymbol{One}}$ не удаляют элементы или кортежи из множеств, а лишь предварительно отмечают их в структурах – списках. Включение или удаление этих элементов осуществляется путем модификации соответствующих предикатов – характеристических функций множеств. В фигурные скобки заключены операции модификации унарных

$S_x(x) \leftarrow$ ***true***, $S_y(y) \leftarrow$ ***true*** и бинарного $R(x, y) \leftarrow$ ***true*** предикатов. где значения предметных переменных x, y известны. В результате подобных модификаций добавляются две вершины и инцидентная им дуга в очередной конфигурации графа H паросочетаний. Это построение является ядром моделирующей программы и в дальнейшем используется при реализации программного обеспечения кластерной системы.

Орграф паросочетаний имеет переменную (эволюционирующую) структуру, и его текущее состояние формально может быть определено следующим образом:

$$H = (V_x, V_y, S_x, S_y, R). \tag{2.2}$$

На рис. 2.1 представлен возможная реализация орграфа паросочетаний с шестью дугами, которые включаются в орграф при моделировании. Шести дугам соответствуют шесть кортежей (x_1, x_{13}), (x_4, x_7), (x_6, x_{14}), (x_{11}, x_5), (x_9, x_{16}), и $(x_{15,} x_3)$, входящих в бинарное отношение R. Это отношение является областью истинности одноименного бинарного предиката. Указанным кортежам в свою очередь соответствуют истинные высказывания $R(x_1, x_{13})$, $R(x_4, x_7)$, $R(x_6, x_{14})$, $R(x_{11}, x_5)$, $R(x_9, x_{16})$, и $R(x_{15,} x_3)$, где R – бинарный предикатный символ.

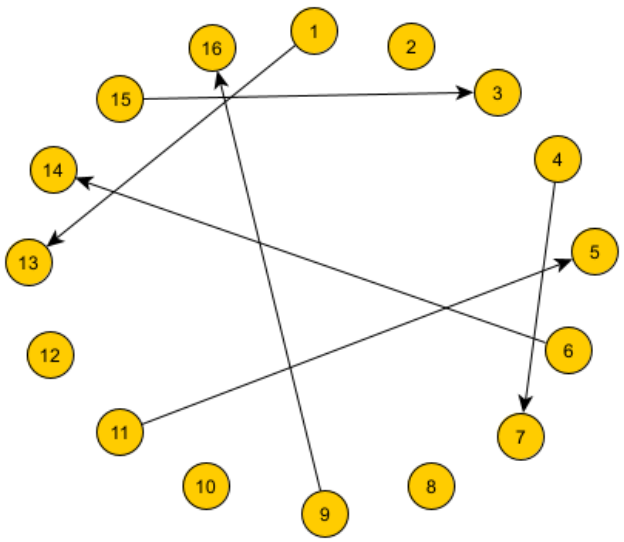


Рис. 2.1. Пример реализации орграфа паросочетаний с шестью дугами

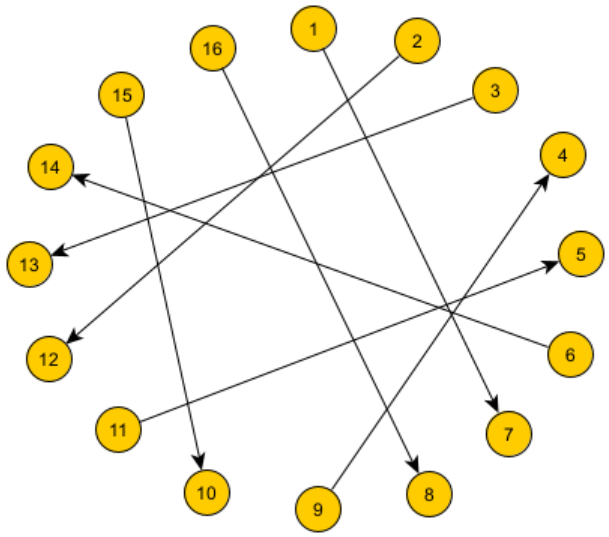


Рис. 2.2. Орграф паросочетаний с максимальным числом дуг

Остальные дуги в орграфе отсутствуют, поскольку, например, в кластерной системе и в модели произошел случай, когда для свободных узлов-источников сообщений не нашлось адресуемых свободных узлов-приемников. Поэтому реализация орграфа паросочетаний, определенного выражением (2.2), может иметь изолированные вершины. Орграф паросочетаний на рис.2.2 не имеет изолированных вершин и имеет максимально возможное число дуг – 8.

Продолжаем описание модели, описанной логико-вероятностным выражением (2.1). Все компоненты кортежа для графа H для этого примера были определены ранее. После добавления очередной дуги в данный граф путем модификаций предикатов $S_x(x) \leftarrow \textbf{\textit{true}}$, $S_y(y) \leftarrow \textbf{\textit{true}}$, $R(x, y) \leftarrow \textbf{\textit{true}}$, $Transfer(x, y) \leftarrow \textbf{\textit{true}}$ выполняется модификация предиката $Delay(x, y) \leftarrow \textbf{\textit{true}}$ – задержки на передачу информации и имитации проведения диалога между источником x и приемником y. В выражении (2.1) при $Transfer(x, y) \leftarrow \textbf{\textit{true}}$ имитируется передача информации от узла-источника к узлу-приемнику, а при $Transfer(x, y) \leftarrow \textbf{\textit{false}}$ передача заканчивается. По истечении времени задержки связь между источником x и приемником y разрывается, и соответствующая модификация графа паросочетаний выполняется путем удаления дуги (x, y) , где значения предметных переменных x, y известны:

$S_x(x) \leftarrow$ ***false***, $S_y(y) \leftarrow$ ***false***, $R(x, y) \leftarrow$ ***false***, $Delay(x, y) \leftarrow$ ***false***, $Transfer(x, y) \leftarrow$ ***false***,

и выполнение очередного этапа моделирования заканчивается. В случае, когда одновременный выбор вершин x и y ввиду их занятости невозможен, одна или обе условные части в квадратных скобках выражения (2.1) не могут быть выполнены успешно, выполняется один или оба оператора *Ret*, имитирующие отработку тайм-аута и передающие управление на повторные итерации выражения (2.1). В программной реализации происходит переход в состояние ожидания.

Выражение (2.1) позволяет строить на основе логико-вероятностного подхода имитационные модели большой размерности, в отличие от исполнимых автоматных моделей или моделей на базе сетей Петри. Например, на основе данного подхода проведено моделирование кластера «Круглый стол» с 8, 16, 24, 32, 64, 128, 256, 512 и 1024 пользовательскими компьютерами, и при выборках до 10000 прогонов модели время ожидания не превышало 10 секунд.

Примечание 1. Нотация α-дизъюнкции первоначально была предложена в системах алгоритмических алгебр В.М. Глушковым для формульной записи алгоритмов [116, 122]: $[\alpha](a \vee b)$ – тернарная операция, зависящая от условия α и операторов a и b, используется в настоящей работе для всюду определенных условий. При $\alpha =$ ***true*** выполняется оператор a, а при $\alpha =$ ***false*** – оператор b. Возможны различные суперпозиции α-дизъюнкций, получаемые подстановкой новых операций α-дизъюнкции на место операторов a и b: при $a = [\alpha_2](c \vee d)$ и $b = [\alpha_3](e \vee f)$ результирующее выражение получит следующий вид:

$$[\alpha]([\alpha_2](c \vee d) \vee [\alpha_3](e \vee f)).$$

В настоящей работе используется модификация следующей суперпозиции операций α-дизъюнкций:

$$[\alpha_1]([\alpha_2](\{a_1, a_2, \ldots, a_k\} \vee Ret) \vee Ret)).$$

Модификация заключается в том, что вместо пустого оператора E использован оператор возврата с ожиданием Ret (от англ. *return*) к проверке с ожиданием истинности соответствующего ему α-условия в случае его первоначальной ложности. В фигурные скобки заключен блок совместимых операций $\{a_1, a_2, \ldots, a_k\}$; в данной работе это элементы своего рода семафорной техники – модификации предикатов, отмечающие происходящие события в моделируемой системе. Символом запятой в блоке обозначена операция конкатенации, или последовательной композиции, операторов, взятая из алгебры операторов системы алгоритмических алгебр [116, 122]. Квадратные, круглые и фигурные скобки являются элементами синтаксиса формул и их нельзя заменять или переносить произвольно. Даная нотация была ранее введена в работах [117, 118, 123].

Отметим, что использование нотации систем алгоритмических алгебр в принципе не является обязательным – возможно использование и других нотаций, позволяющих записывать алгоритмы моделирования в виде формул.

Примечание 2. В формульной записи моделей использованы операторы $\exists_{Any}$ и $\exists_{One}$ выбора кортежей из отношений, нотация и семантика для которых были использованы в работах [117, 118, 123]. Основная форма записи первого из этих операторов имеет следующий вид: $\exists_{Any}<x_1, x_2, \ldots, x_k> R_1$, где $x_1, x_2, \ldots, x_k$ – элементы кортежа – предметные переменные, получающие значения выбранных предметных констант, возможно входящих в какой-либо кортеж отношения R_1 – области истинности одноименного k-арного предиката. В выражении (2.1) и других выражениях для формульной записи логико-вероятностных и логико-алгебраических моделей этот оператор заключен в квадратные скобки $[\exists_{Any}<x_1, x_2, \ldots, x_k> R_1]$, которые, изменяя интерпретацию, «превращают» данный оператор в условие, истинное при успешном завершении операции выбора произвольного кортежа и ложное в противном случае. При построении логико-вероятностных моделей такой оператор используется часто. Второй оператор вида $\exists_{One}<x_1, x_2, \ldots, x_k> R_2$ отличается от первого тем, что в отношении R_2 выбранный кортеж должен был быть единственным.

2.4. Переход от имитационной логико-вероятностной модели к логико-алгебраической спецификации функциональной архитектуры вычислительного кластера

Программа имитационного моделирования, созданная на основании логико-вероятностного выражения (2.1), предназначена для выполнения на одном компьютере в многопоточном режиме. Однако на основе этого выражения путем несложного изменения можно получить выражение, пригодное для спецификации программного обеспечения реальной кластерной вычислительной системы. При этом нужно учесть, что программа должна выполняться в каждом компьютере, подключенном к коммутатору вычислительного кластера. Первый шаг трансформации заключается в том, что выражений, подобных выражению (2.1), должно быть m – то есть по одному выражению для будущих программ на каждый компьютер кластера. Ниже представлено выражение (2.3), которое использовано в качестве прототипа спецификации сетевого приложения для i-го компьютера кластера:

$$
\begin{aligned}
&M_i = [Start(x_i)](\{Start(x_i) \leftarrow \textbf{\textit{false}}, Master(x_i) \leftarrow \textbf{\textit{true}},\\
&Work^*(x_i) \leftarrow \textbf{\textit{true}}\} \vee Ret), ([(\exists \textbf{\textit{One}}\,(x_i, y_j) \in V^2) \neg S_y(y_j)\ \&\ (x_i \neq y_j)]\\
&(\{S_x(x_i) \leftarrow \textbf{\textit{true}}, S_y(y_j) \leftarrow \textbf{\textit{true}}, R(x_i, y_j) \leftarrow \textbf{\textit{true}}, Transfer(x_i, y_j) \leftarrow \textbf{\textit{true}},\\
&Proc^*(x_i, y_j) \leftarrow \textbf{\textit{true}}, S_x(x_i) \leftarrow \textbf{\textit{false}}, S_y(y_j) \leftarrow \textbf{\textit{false}}, R(x_i, y_j) \leftarrow \textbf{\textit{false}},\\
&Transfer(x_i, y_j) \leftarrow \textbf{\textit{true}}, Proc^*(x_i, y_j) \leftarrow \textbf{\textit{false}},\\
&Master(x_i) \leftarrow \textbf{\textit{false}}, Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret), \qquad (2.3)\\
&i, j = 1, 2, \ldots, m;\ i \neq j.
\end{aligned}
$$

От выражения (2.1) выражение (2.3) отличается введением индексов i и j, которым в реальном сетевом приложении соответствуют адреса компьютеров, подключенных к коммутатору. Задано новое имя выражения M_i (первая буква от имени *Master*, что означает, что передающий узел кластера получает статус мастера). Введены новые унарные предикаты *Start*, *Master*, *Work**, по-

новому интерпретируется бинарный предикат *Proc**. Звездочки при предикатных именах *Work** и *Proc** означают, что при реализации прототипного программного обеспечения кластера этим предикатам соответствуют вычислительные процедуры.

На основании логико-алгебраического выражения M_i строится прототипное программное обеспечение, которое устанавливается на каждый узел вычислительного кластера. При дальнейшем описании там, где это не вызывает противоречий, для упрощения формулировок, будут использованы термины как от модели, так и от действующего кластера.

При $Start(x_i) = \textbf{\textit{true}}$ начинается работа приложения-мастера на узле x_i кластера, которая отмечается модификациями предикатов $Master(x_i) \leftarrow \textbf{\textit{true}}$ и $Work^*(x_i) \leftarrow \textbf{\textit{true}}$. Начиная с этих событий начинается работа узла-передатчика, получившего статус мастера, то есть инициатора последующих действий.

Следующее выражение:

$$[Start(x_i)](\{Master(x_i) \leftarrow \textbf{\textit{true}}, Work^*(x_i) \leftarrow \textbf{\textit{true}}\} \vee Ret) \qquad (2.4)$$

имеет такой смысл: оператор *Ret* возвращает управление проверке условия $[Start(x_i)]$ при ложности высказывания $Start(x_i)$. От оставшейся части выражения (2.3) выражение (2.4) отделено символом запятой, означающей в данном контексте последовательное выполнения последующих действий. Следующая часть выражения начинается с оператора $(\exists_{\textbf{\textit{One}}}(x_i, y_j) \in V^2)$ выбора единственного кортежа (x_i, y_j), где x_i соответствует узлу-передатчику, а y_j – узлу-приемнику кластера. Для реализации операции выбора необходимо, чтобы было истинно составное высказывание $\neg S_y(y_j)\ \&\ (x_i \neq y_j)$ – узел-приемник y_j кластера не занят, а x_i и y_j – разные узлы. Оператор $\exists_{\textbf{\textit{One}}}(x_i, y_j)$ путем обращения к коммутатору при известном адресе порта компьютера-источника x_i определяет ад-

рес порта компьютера-приемника y_j. Далее выполняются следующие модификации предикатов $S_x(x_i) \leftarrow \textbf{\textit{true}}$, $S_y(y_j) \leftarrow \textbf{\textit{true}}$, которым в реальной кластерной системе соответствуют события занятия узлов x_i и y_j. Модификации предиката $R(x_i, y_j) \leftarrow \textbf{\textit{true}}$ соответствует событие связывания узлов x_i и y_j кластера при подготовке с совместным действиям при реализации диалога и обработке данных; модификации предиката $Proc^*(x_i, y_j) \leftarrow \textbf{\textit{true}}$ соответствует выполнение операции обработки запроса; оставшаяся часть выражения (2.3) описывает события, связанные завершением сеанса связи и с освобождением пары узлов кластера x_i и y_j.

В каждом i-м элементе кластера локально формируется реализация простейшего графа паросочетаний H_i, который включает не более одной дуги (x_i, y_j):

$$H_i = (V_{xi}, V_{yj}, S_{xi}, S_{yj}, R_i);\ i, j = 1, 2, \ldots, m;\ i \neq j. \qquad (2.5)$$

Завершая описания событий в вычислительном кластере, отметим, что информация, указанная в выражении (2.3) и затрагивающая узел-приемник y_j, передается в данный узел от узла-мастера x_i при помощи управляющих сообщений. Детали формирования и передачи подобных сообщений в сетевом приложении для кластера просты и не затронуты в формализации.

2.5. Результаты макроанализа производительности вычислительного кластера на основе логико-вероятностной имитационной модели, задающей равнодоступные парные взаимодействия (модель «Круглый стол»)

Результаты макроанализа имитационной модели «Круглый стол» в основном режиме

Для режима «Круглый стол» достаточно сложно получить оценки вероятностных характеристик для распределений вероятностей времени подготовки запросов и времени обработки в реальной системе, поскольку нагрузка зависит от множества решаемых задач, времени суток и др. Экспоненциальное

распределение предполагает значительную вариабельность переменной и по сравнению с многими распространенными в теории массового обслуживания систем распределений имеет большую дисперсию [111]. Для сравнительного макроанализа кластерных вычислительных систем такие предпосылки приемлемы, так как они обычно позволяют имитировать работу системы в более жестких условиях. Следует сказать, что при статистическом моделировании можно сменять распределения вероятностей – от произвольных распределений, задаваемых при помощи кусочно-линейных аппроксимаций до многих других, применяемых при моделировании сложных систем.

Выражения (2.1), (2.2), (2.3) и (2.5) определяют основную концептуальную схему для разработки имитационной поведенческой модели функционирования вычислительного кластера, реализующего равнодоступные парные взаимодействия (модель «Круглый стол»).

При моделировании далее используются псевдослучайные величины, заданные смещенным экспоненциальным распределением, для которого функция плотности вероятности $f(t)$ имеет следующий вид [111, 124]:

$$f(t) = \lambda\, exp(-\lambda\,(t - t_0)) \text{ при } t \geq t_0 \text{ и } 0 \text{ при } t < t_0,$$

где t_0 – параметр смещения; в программе такие псевдослучайные величины генерируются при помощи функции ($expon(i, t_0, t_{ср})$), где среднее время $t_{ср}$ между событиями в потоке определяется равенством $t_{ср} = 1/\lambda$. Такая функция широко известна в теории вероятностей и массового обслуживания и нередко используется при статистическом моделировании систем, когда приходится оперировать со псевдослучайными величинами, минимальное значение которых ограничено некоторой постоянной величиной t_0 [124]. Функция распределения такой величины имеет следующий вид:

$$F(t) = 1 - exp(-\lambda\,(t - t_0)).$$

При проведении экспериментов с имитационной моделью число пар NxN изменялось от 2x2 до 1024x1024. Число портов N_P коммутатора выбрано равным N. Запись вида NxN означает, что каждым из N узлов кластера организуются

случайные парные обмены («беседы») с другими узлами из этого же множества узлов с учетом перечисленных ранее ограничений, связанными с конфликтами и образованием очередей. Результаты экспериментальных исследований имитационной модели кластера, работающего в режиме «Круглый стол» сведены в табл. 2.1.

Табл. 2.1

$N_{п/п}$	Число пар NxN	T_{Pair}, мс	T_1, мс	K_{Slow}
1	2x2	411	220	1,87
2	4x4	614	220	2,8
3	8x8	803	220	3,65
4	16x16	980	220	4,45
5	24x24	1077	220	4,9
6	32x32	1150	220	5,22
7	48x48	1240	220	5,63
8	64x64	1300	220	5,9
9	128x128	1449	220	6,59
10	256x256	1595	220	7,25
11	512x512	1732	220	7,87
12	1024x1024	1870	220	8,5

Предполагалось, что время подготовки запроса («обдумывания темы беседы») распределено в соответствии со смещенным экспоненциальным распределением с параметрами $t_{ср}$ = 100 мс, t_0 = 10 мс. Время обработки запроса («обдумывания ответа») определено теми же параметрами. В результате моделирования определялось значение главного параметра – времени T_{Pair} обслуживания запроса, включающего время его подготовки. Значения этого времени, оцененные по результатам статистического эксперимента, указаны в 3-й колонке табл. 2.1. Интерес также представляло соотношение между реальным временем T_{Pair} и «идеальным» T_1, которое было бы равно $2(t_{ср} + t_0)$ = 220 мс в случаях, когда граф паросочетаний имел бы N дуг при бесконфликтной ситуации. Коэффициент замедления обслуживания из-за взаимного влияния запросов равен $K_{Slow} = T_{Pair} / T_1$. При учете этого коэффициента возможно оценить работу кластера при наличии случайных парных взаимодействий. Рис. 2.3 и рис. 2.4 наглядно иллюстрируют влияние зависимости запросов друг от

друга при наличии конфликтов и очередей. Моделирование производилось при изменении $N = 2, 4, 8, 16, 24, 32, 48, 64, 128, 256, 512, 1024$; число портов N_P коммутатора равно N, объем выборки составлял 10000. Подтверждено, что благодаря организации очередей в коммутаторе кластерная система в режиме «Круглый стол» работает без тупиков.

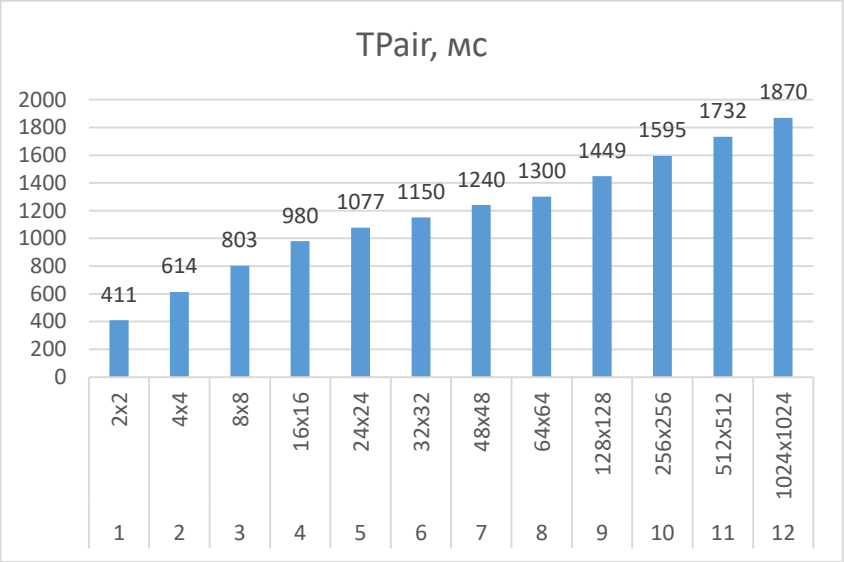


Рис. 2.3. Зависимость среднего времени обработки запроса T_{Pair} от числа узлов N в вычислительном кластере

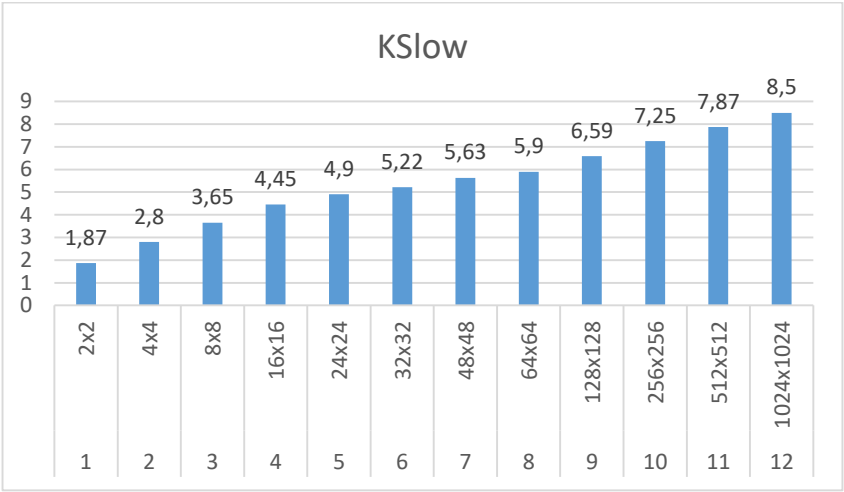
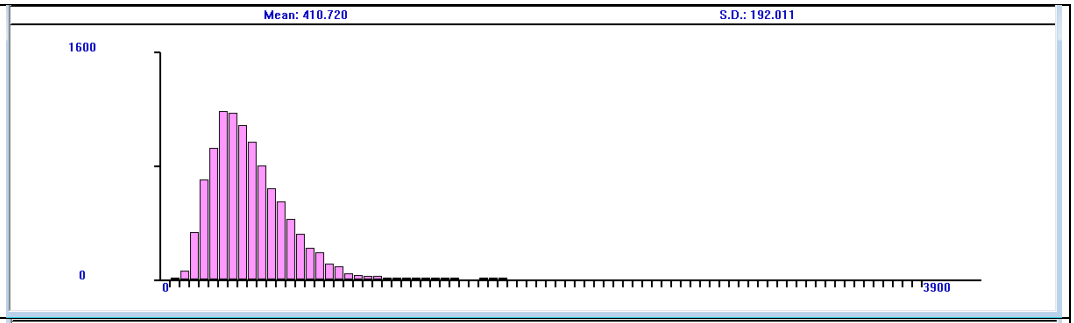
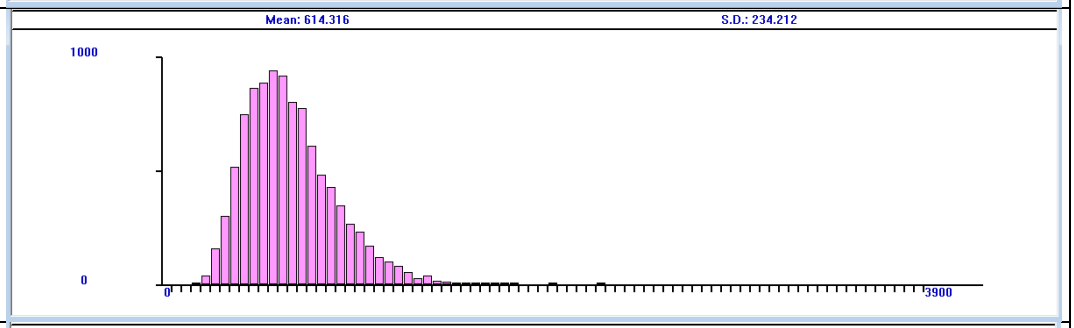
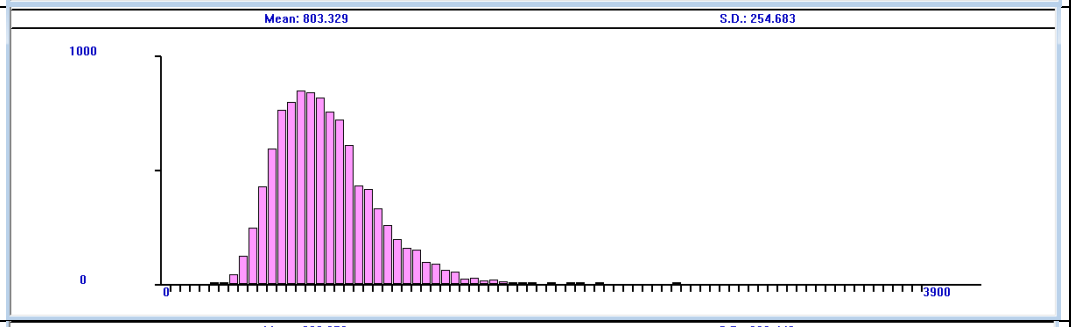
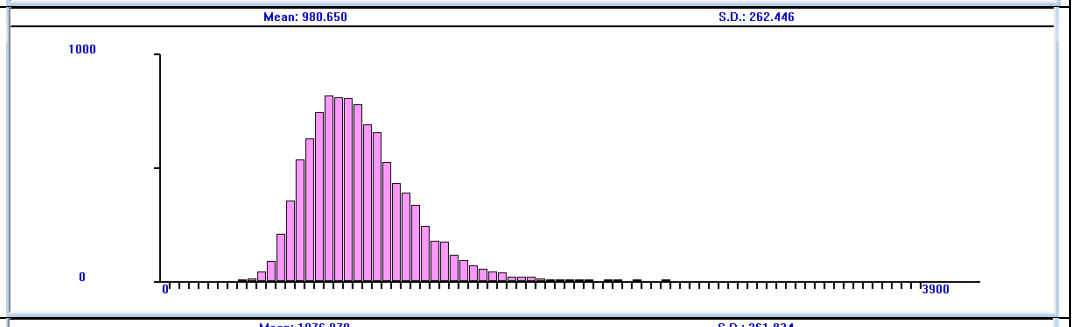
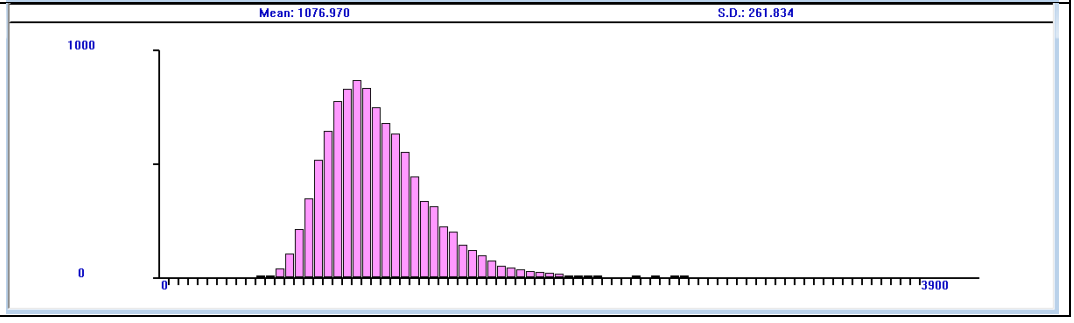
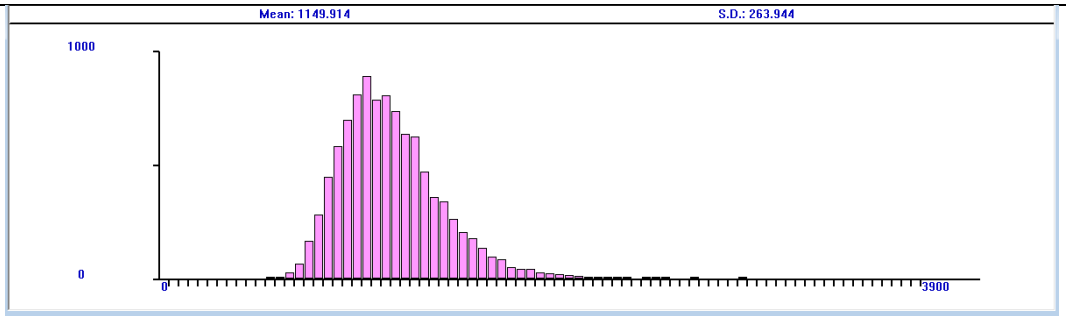
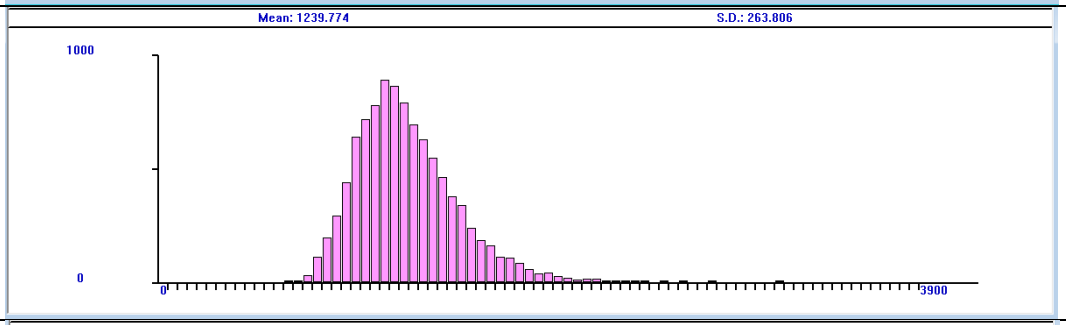
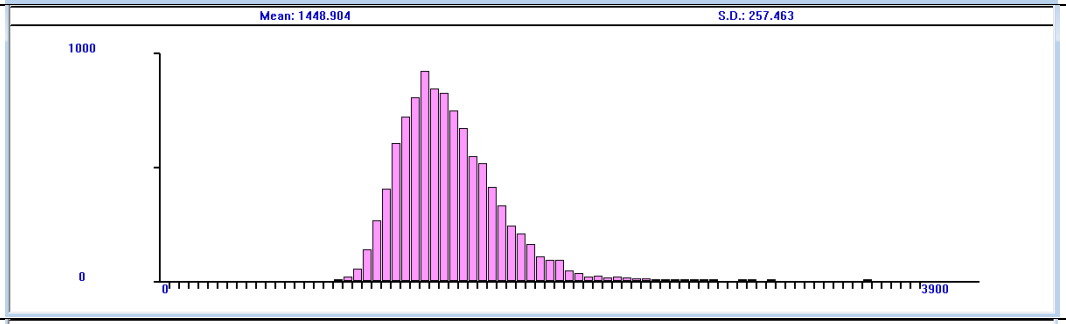
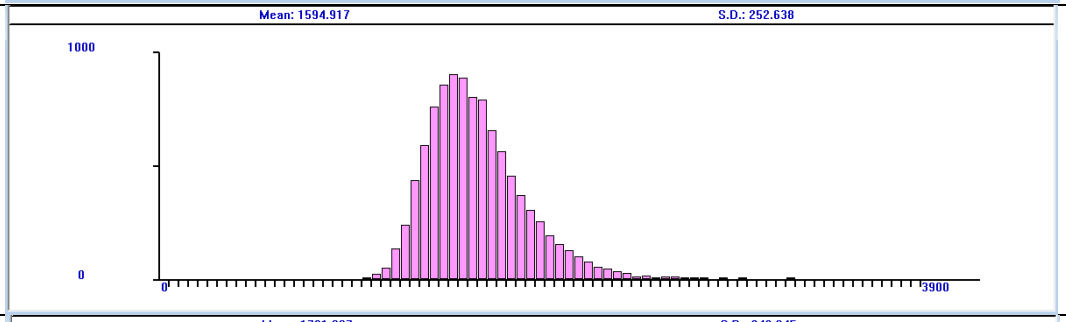
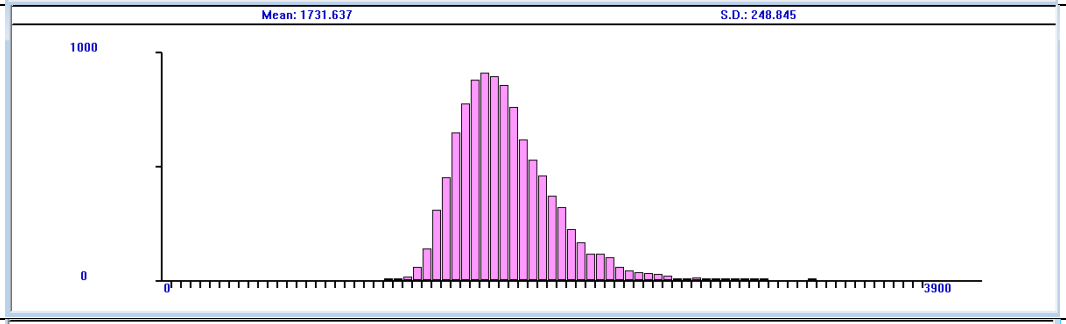
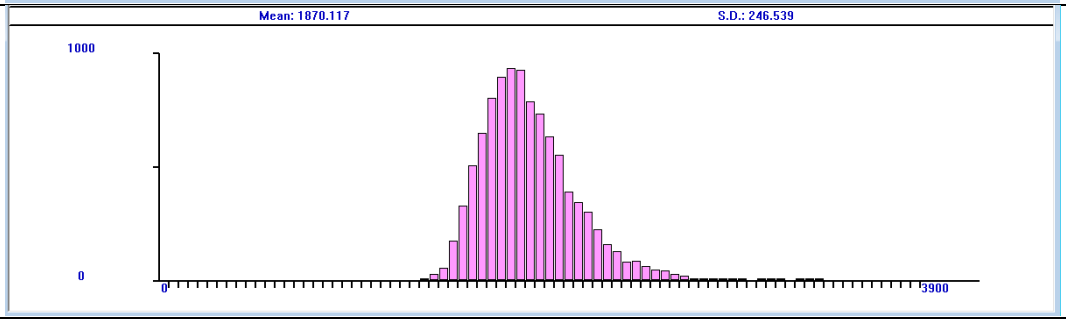


Рис. 2.4. Зависимость коэффициента замедления обработки запроса K_{Slow} от числа узлов N в вычислительном кластере

Ниже приведены 11 гистограмм распределения времени обработки запросов T_{Pair} для различного числа участников «круглого стола» – от 2 до 1024.

2x2	
4x4	
8x8	
16x16	
24x24	

32x 32	
64x 64	
128x 128	
256x 256	
512x 512	
1024x 1024	

Слева от каждой гистограммы указано число узлов, или участников «круглого стола». Гистограммы, как графические представления данных, позволяют наглядно визуализировать распределение частот величины времени обработки запроса. В верхней части каждого рисунка указаны оценки математического ожидания *Mean* и среднеквадратического отклонения *S.D.* (*Standard Deviation*) времени обработки запросов. По рисункам видно, что гистограммы имеют длинные «хвосты», что свидетельствует о наличии выбросов или экстремальных значений времени обработки запроса. Однако эти отклонения достаточно редки и не оказывают значительного влияния на работу вычислительного кластера в режиме «Круглый стол».

Результаты макроанализа имитационной модели «Круглый стол» при ограничении числа портов коммутатора локальной сети

Следующий этап макроанализа будет связан с выбором числа портов коммутатора. Для этого выберем $N = 1024$ и проведем эксперименты с моделью при изменяющемся числе портов N_P. Результаты испытаний сведены в табл. 2.2. Как и ранее, запись вида *N*x*N* означает, что каждым из *N* узлов кластера организуются случайные парные обмены с другими узлами из этого же множества.

Табл. 2.2

$N_{п/п}$	Число пар *N*x*N*	Кол-во портов N_P	T_{Pair}, мс	K_{Lim}
1	1024x1024	1024	1870	1
2	1024x1024	512	1873	1
3	1024x1024	256	1873	1
4	1024x1024	128	1905	1,02
5	1024x1024	64	2440	1,21
6	1024x1024	32	4017	2,15
7	1024x1024	24	5130	2,74
8	1024x1024	16	7404	3,96
9	1024x1024	8	14339	7,67

Как и следовало ожидать, с убыванием числа портов коммутатора кластера среднее время T_{Pair} подготовки и обработки запросов увеличивается. При числе портов 256, 512, 1024 значения T_{Pair} сохраняется почти равными, при этом максимальное число занятых портов приблизительно равно максимальному числу «беседующих» пар (≈200). При дальнейшем уменьшении числа

портов N_P время T_{Pair} увеличивается. Таким образом, пользователь кластера на основании табл. 2.1 и табл. 2.2 может оценить время T_{Pair} подготовки и обработки запросов и выбрать необходимое число портов коммутатора кластера. Рисунки 2.5 и 2.6 наглядно иллюстрируют влияние числа портов N_P на важнейшую характеристику производительности T_{Pair}.

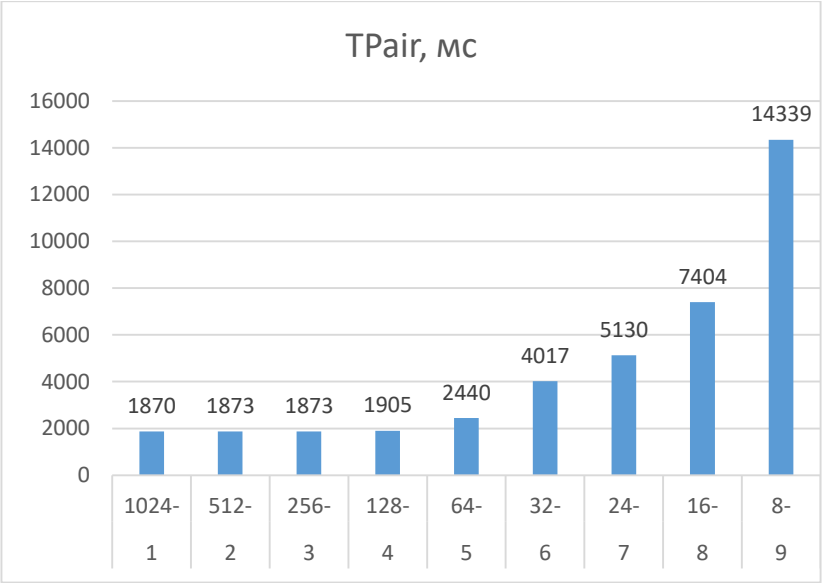


Рис. 2.5. Зависимость среднего времени обработки запроса T_{Pair} от числа N_P портов коммутатора при числе пар NxN = 1024x1024

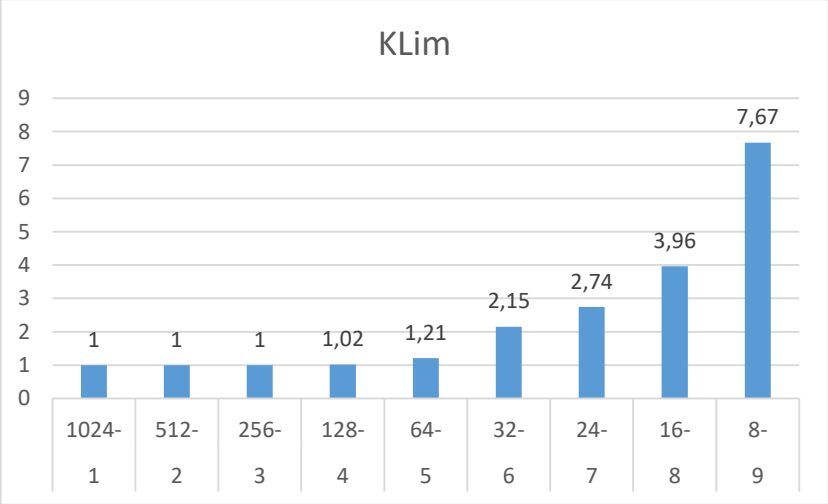
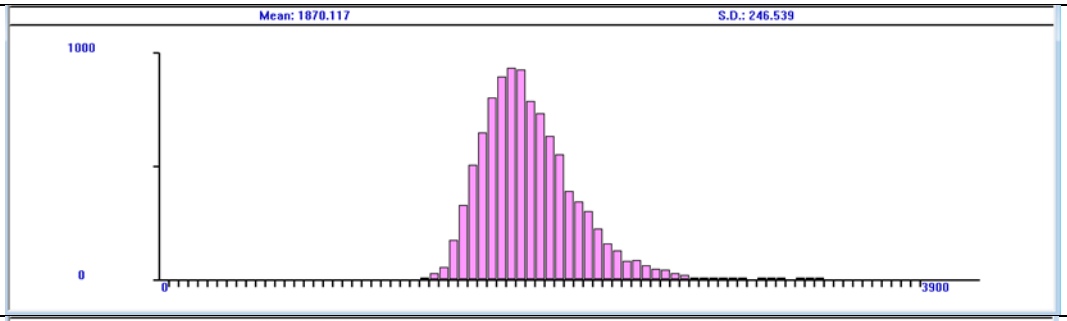
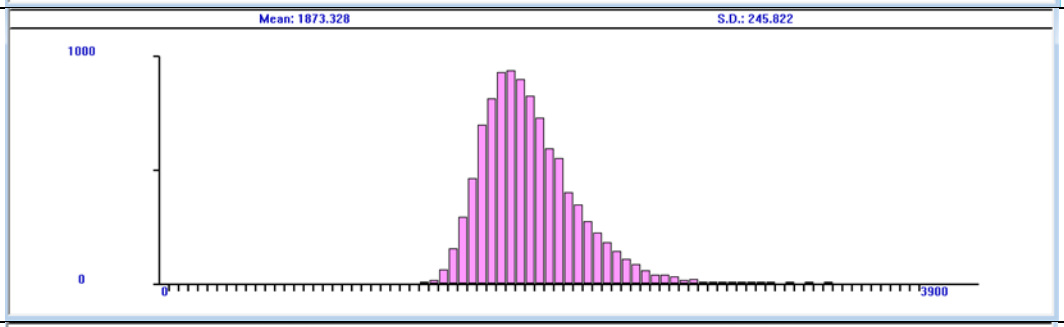
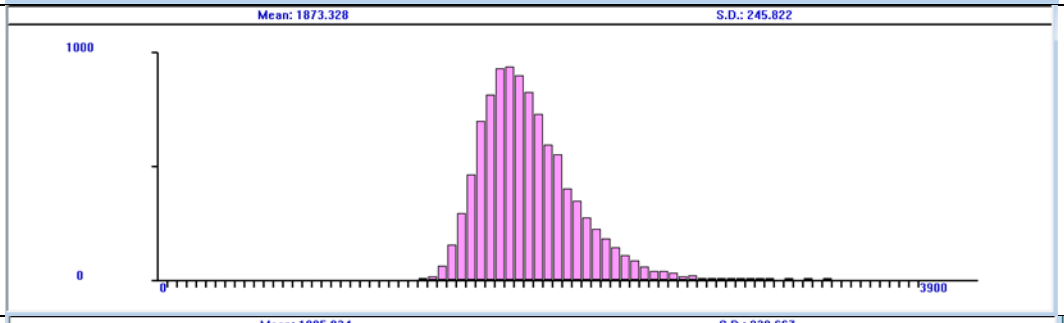
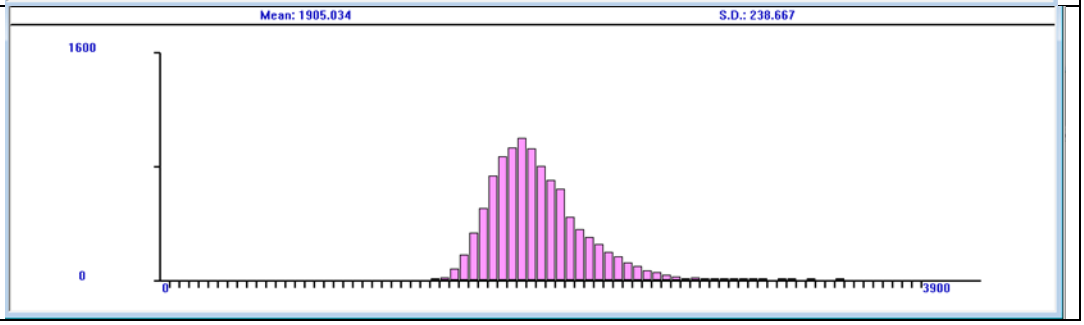
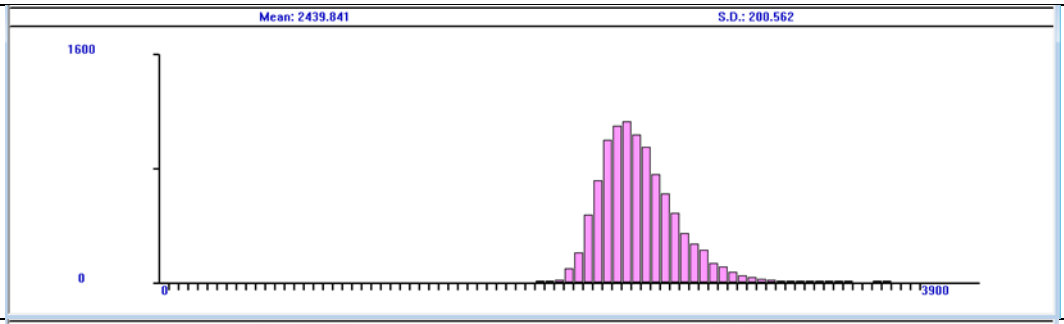
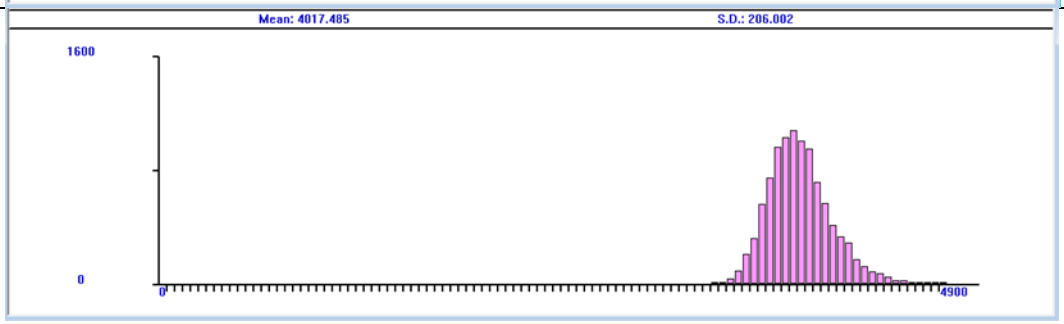


Рис. 2.6. Зависимость коэффициента замедления обработки запроса K_{Lim} от от числа N_P портов коммутатора при числе пар NxN = 1024x1024

Значения коэффициента K_{Lim} вычисляются как отношения времени обработки T_{Pair} при $N_p < 1024$ к наименьшему значению этого же параметра при $N_p = 1024$. Например, $K_{Lim,8} = T_{Pair,8}/T_{Pair,1024} = 14339/1870 = 7{,}67$.

Ниже представлены 6 гистограмм распределения времени T_{Pair} при $N = 1024$ и различных значениях числа портов N_p коммутатора COM. По рисункам, как и в предыдущем случае, видно, что гистограммы имеют длинные «хвосты», что свидетельствует о наличии выбросов или экстремальных значений времени обработки запроса. Однако эти отклонения достаточно редки и не оказывают значительного влияния на работу вычислительного кластера в режиме «Круглый стол».

1024 COM 1024	
1024 COM 512	
1024 COM 256	
1024 COM 128	

1024 COM 64	
1024 COM 32	

Как и для предыдущей серии статистических экспериментов, в верхней части каждого рисунка указаны оценки математического ожидания *Mean* и среднеквадратического отклонения *S.D.* (*Standard Deviation*) времени обработки запросов.

2.6. Логико-вероятностная модель «Резидент-агенты»: последовательная раздача заданий – последовательное выполнение

Пусть теперь предметная константа x_0 представляет в модели ведущий компьютер кластера, условно названный «резидентом», а множество $V = \{x_1, x_2,\ldots, x_m\}$ представляет агенты – «рядовые» машины кластера. Обозначим через $Q = \{x_0\} \times V$ – множество всех пар вида (x_0, x_i), $i = 1, 2,\ldots, m$, тогда модификации бинарного предиката $R(x_0, x_i) \leftarrow \boldsymbol{true}$ в модели соответствует включение дуги (x_0, x_i) в граф C устанавливаемых межузловых связей в кластере, а модификации $R(x_0, x_i) \leftarrow \boldsymbol{false}$ соответствует удаление дуги (x_0, x_i). Выборку кортежа (x_0, x_i) из отношения Q осуществляет оператор $\exists_{\boldsymbol{One}}(x_0, x_i) \in Q$. Остальные предикаты в выражении (2.6) имеют такой же смысл, как и в определениях моделей (2.1) и (2.3). Модификации унарного предиката *Resident* соответствует инициирование и завершение работы ведущего компьютера кластера:

$$M_0 = [Start(x_0)](\{Resident(x_0) \leftarrow \boldsymbol{true}, Work^*(x_0) \leftarrow \boldsymbol{true}\} \vee Ret),$$

$$SeqReplicate(i = 1..m)\colon ([(\exists_{\boldsymbol{One}}(x_0, x_i) \in Q) \neg S_x(x_i)]$$

$$(\{S_x(x_i) \leftarrow \textbf{\textit{true}}, R(x_0, x_i) \leftarrow \textbf{\textit{true}}, Proc^*(x_0, x_i) \leftarrow \textbf{\textit{true}},$$
$$S_x(x_i) \leftarrow \textbf{\textit{false}}, R(x_0, x_i) \leftarrow \textbf{\textit{false}}, Proc^*(x_0, x_i) \leftarrow \textbf{\textit{false}},$$
$$Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret)), Resident(x_0) \leftarrow \textbf{\textit{false}}). \quad (2.6)$$

Отличительной особенностью формулы (2.6) является использование оператора репликации *SeqReplicate*(*i = 1..m*), которая в формуле означает последовательную репликацию (повторение) записи формулы, заключенной в скобки справа, с последовательной подстановкой индексированных переменных:

$$([(\exists_{\textbf{\textit{One}}}(x_0, x_i) \in Q) \neg S_x(x_i)]$$
$$(\{S_x(x_i) \leftarrow \textbf{\textit{true}}, R(x_0, x_i) \leftarrow \textbf{\textit{true}}, Proc^*(x_0, x_i) \leftarrow \textbf{\textit{true}},$$
$$S_x(x_i) \leftarrow \textbf{\textit{false}}, R(x_0, x_i) \leftarrow \textbf{\textit{false}}, Proc^*(x_0, x_i) \leftarrow \textbf{\textit{false}},$$
$$Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret)). \quad (2.7)$$

В прототипной программе, управляющей согласно формуле (2.6) работой всего кластера, реплицируемой части данной формулы соответствует исполняемая циклически часть программы.

В выражении (2.6) считалось, что рабочие программы элементов кластера уже загружены. Однако во многих случаях требуется выполнение первоначальной загрузки, проводимой последовательно для всех элементов или параллельно, одновременно для всех элементов кластера.

При последовательной загрузке элементов кластера логико-алгебраическое и последовательном выполнении рабочих программ выражение M_{SeqSeq} будет иметь следующий вид (2.8):

$$M_{SeqSeq} = [Start(x_0)](\{ Start(x_0) \leftarrow \textbf{\textit{false}}, Resident(x_0) \leftarrow \textbf{\textit{true}},$$
$$Work^*(x_0) \leftarrow \textbf{\textit{true}}\} \vee Ret),$$
$$SeqReplicate(i = 1..m): ([(\exists_{\textbf{\textit{One}}}(x_0, x_i) \in Q) \neg S_x(x_i)]$$

$$
\begin{aligned}
&(\{S_x(x_i) \leftarrow \textbf{\textit{true}},\ R(x_0, x_i) \leftarrow \textbf{\textit{true}},\ Load^*(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&S_x(x_i) \leftarrow \textbf{\textit{false}},\ R(x_0, x_i) \leftarrow \textbf{\textit{false}},\ Load^*(x_0, x_i) \leftarrow \textbf{\textit{false}},\\
&Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret)),\ Resident(x_0) \leftarrow \textbf{\textit{false}}),\\
&[Start(x_0)](\{Start(x_0) \leftarrow \textbf{\textit{false}},\ Resident(x_0) \leftarrow \textbf{\textit{true}},\\
&Work^*(x_0) \leftarrow \textbf{\textit{true}},\ Start(x_0) \leftarrow \textbf{\textit{true}}\} \vee Ret),\\
&SeqReplicate(i = 1..m)\colon ([(\exists_{\textbf{\textit{One}}}(x_0, x_i) \in Q) \neg S_x(x_i)]\\
&(\{Start(x_0) \leftarrow \textbf{\textit{false}},\ S_x(x_i) \leftarrow \textbf{\textit{true}},\ R(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&Proc^*(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&S_x(x_i) \leftarrow \textbf{\textit{false}},\ R(x_0, x_i) \leftarrow \textbf{\textit{false}},\ Proc^*(x_0, x_i) \leftarrow \textbf{\textit{false}},\\
&Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret)),\ Resident(x_0) \leftarrow \textbf{\textit{false}}).
\end{aligned}
\tag{2.8}
$$

Здесь бинарный предикат *Load* моделирует процедуру начальной загрузки каждого элемента кластера.

2.7. Построение логико-вероятностных моделей функционирования вычислительного кластера, реализующего параллельный доступ «один ко многим» (модели «Собрание 1» и «Собрание 2»)

Следующие модели связаны с параллельной обработкой информации в кластерной вычислительной системе.

Модель «Собрание 1»

Ведущий компьютер кластера подготавливает данные а затем передает их для последующей параллельной обработки в ведомых компьютерах. Метафорическое название новой модели – «Собрание 1», где «председатель» (*Chairman*) подготавливает и раздает последовательно задания участникам (режим репликации – $SeqReplicate(i = 1..m)$). Новая логико-алгебраическая модель $C_{Seq,Par}$ (2.9) построена путем переинтерпретации предыдущей модели

(2.8) и изменения режима выполнения загруженных программ кластером с последовательного на параллельный (но при последовательной загрузке программ и данных на каждый элемент кластера):

$$
\begin{aligned}
&C_{Seq,Par} = [Start(x_0)](\{\ Start(x_0) \leftarrow \textbf{\textit{false}},\ Chairman(x_0) \leftarrow \textbf{\textit{true}},\\
&Work^*(x_0) \leftarrow \textbf{\textit{true}}\} \vee Ret),\\
&SeqReplicate(i = 1..m)\colon ([(\exists_{\textbf{\textit{One}}}(x_0, x_i) \in Q) \neg S_x(x_i)]\\
&(\{S_x(x_i) \leftarrow \textbf{\textit{true}},\ R(x_0, x_i) \leftarrow \textbf{\textit{true}},\ Load^*(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&S_x(x_i) \leftarrow \textbf{\textit{false}},\ R(x_0, x_i) \leftarrow \textbf{\textit{false}},\ Load^*(x_0, x_i) \leftarrow \textbf{\textit{false}},\\
&Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret)),\ Chairman(x_0) \leftarrow \textbf{\textit{false}}),\\
&[Start(x_0)](\{Start(x_0) \leftarrow \textbf{\textit{false}},\ Chairman(x_0) \leftarrow \textbf{\textit{true}},\\
&Work^*(x_0) \leftarrow \textbf{\textit{true}},\ Start(x_0) \leftarrow \textbf{\textit{true}}\} \vee Ret),\\
&ParReplicate(i = 1..m)\colon ([(\exists_{\textbf{\textit{One}}}(x_0, x_i) \in Q) \neg S_x(x_i)]\\
&(\{Start(x_0) \leftarrow \textbf{\textit{false}},\ S_x(x_i) \leftarrow \textbf{\textit{true}},\ R(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&Proc^*(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&S_x(x_i) \leftarrow \textbf{\textit{false}},\ R(x_0, x_i) \leftarrow \textbf{\textit{false}},\ Proc^*(x_0, x_i) \leftarrow \textbf{\textit{false}},\\
&Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret)),\ Chairman(x_0) \leftarrow \textbf{\textit{false}}).
\end{aligned}
\qquad (2.9)
$$

Здесь *ParReplicate*(*i = 1..m*) – оператор параллельной репликации выражения, записанного справа от него в круглых скобках, что соответствует в рабочей программе репликацией частей программы, соответствующей каждому выражению. Таких частей будет *m*, по одной на каждый ведомый компьютер. Поскольку полагается, что все реплики для выражения (2.9) независимы друг от друга, то все отношения, используемые в этом выражении, сегментируются, и каждый сегмент входит в реплику, то есть затем в реализованное программно выражение. Например, при $i = 2$ в программу для 2-го компьютера кластера входят кортежи $< x_2 >$ и $< x_0, x_2 >$. Передача и сбор результатов вычислений от каждого из *m* ведомых компьютеров сопровождается событием модификации предиката $Chairman(x_0) \leftarrow$ **false** в ведущем компьютере кластера.

Модель «Собрание 2»

При возможной реализации параллельной первоначальной загрузки рабочих программ в элементы кластера в параллельном широковещательном режиме, логико-алгебраическая модель $C_{Seq,Par}$ (2.9) преобразуется к следующему виду (2.10):

$$
\begin{aligned}
&C_{Par,Par} = [Start(x_0)](\{\ Start(x_0) \leftarrow \textbf{\textit{false}},\ Chairman(x_0) \leftarrow \textbf{\textit{true}},\\
&Work^*(x_0) \leftarrow \textbf{\textit{true}}\} \vee Ret),\\
&ParReplicate(i = 1..m)\colon ([(\exists_{\textbf{\textit{One}}}(x_0, x_i) \in Q) \neg S_x(x_i)]\\
&(\{S_x(x_i) \leftarrow \textbf{\textit{true}},\ R(x_0, x_i) \leftarrow \textbf{\textit{true}},\ Load^*(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&S_x(x_i) \leftarrow \textbf{\textit{false}},\ R(x_0, x_i) \leftarrow \textbf{\textit{false}},\ Load^*(x_0, x_i) \leftarrow \textbf{\textit{false}},\\
&Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret)),\ Chairman(x_0) \leftarrow \textbf{\textit{false}}),\\
&[Start(x_0)](\{Start(x_0) \leftarrow \textbf{\textit{false}},\ Chairman(x_0) \leftarrow \textbf{\textit{true}},\\
&Work^*(x_0) \leftarrow \textbf{\textit{true}},\ Start(x_0) \leftarrow \textbf{\textit{true}}\} \vee Ret),\\
&ParReplicate(i = 1..m)\colon ([(\exists_{\textbf{\textit{One}}}(x_0, x_i) \in Q) \neg S_x(x_i)]\\
&(\{Start(x_0) \leftarrow \textbf{\textit{false}},\ S_x(x_i) \leftarrow \textbf{\textit{true}},\ R(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&Proc^*(x_0, x_i) \leftarrow \textbf{\textit{true}},\\
&S_x(x_i) \leftarrow \textbf{\textit{false}},\ R(x_0, x_i) \leftarrow \textbf{\textit{false}},\ Proc^*(x_0, x_i) \leftarrow \textbf{\textit{false}},\\
&Work^*(x_i) \leftarrow \textbf{\textit{false}}\} \vee Ret)),\ Chairman(x_0) \leftarrow \textbf{\textit{false}}).
\end{aligned}
\tag{2.10}
$$

Выражения (2.6), (2.7), (2.8), (2.9) и (2.10) возможно отнести к классу логико-алгебраических спецификаций к программному обеспечению прикладного уровня и уровня *middleware* кластерной системы. Эту же роль, но на начальных этапах работ по организации функционирования кластера, выполняют логико-вероятностные модели. По завершении этапа формализации моделей вычислительных кластеров, необходимо перейти собственно к этапу макроанализа их производительности на основе проведения статистических экспериментов.

2.8. Результаты статистического моделирования к задачам организации функциональной архитектуры вычислительного кластера, работающего в режимах «Резидент-агенты» и двух вариантах режима «Собрание»

На основе логико-вероятностных и логико-алгебраических моделей построены имитационные поведенческие модели, позволяющие получить главные характеристики производительности – оценки среднего времени выполнения запроса, гистограммы распределения времени выполнения запроса. Определены следующие оценки математического ожидания времени выполнения запросов в вычислительном кластере:

T_{seq} – оценка математического ожидания времени выполнения запроса в режиме последовательной загрузки всех узлов кластера и дальнейшей последовательной же обработки запросов (режим «Резидент-агенты»);

T_{sp} – оценка математического ожидания времени выполнения запросов в режиме последовательной загрузки всех узлов кластера и дальнейшей параллельной обработки запросов всеми узлами кластера (режим «Собрание 1»);

T_{pp} – оценка математического ожидания времени выполнения запроса в режиме параллельной загрузки всех узлов кластера и дальнейшей параллельной обработки запросов всеми узлами кластера (режим «Собрание 2»).

Время загрузки программ в узлы кластера выбрано равным 10 мс; время обработки запроса задается смещенным экспоненциальным распределением, для которого функция плотности вероятности $f(t)$ имеет следующий вид [111]:

$$f(t) = \lambda\, exp(-\lambda\,(t - t_0) \text{ при } t \geq t_0 \text{ и } 0 \text{ при } t < t_0,$$

с параметрами $t_{\text{ср}} = 500$ мс, $t_0 = 100$ мс.

В табл. 2.3 и на рис. 2.7, 2.8 и 2.9 представлены результаты статистических экспериментов с имитационными моделями вычислительного кластера, работающего в режимах «Резидент-агенты», «Собрание 1» и «Собрание 2». Кроме указанных выше оценок математических ожиданий времен обработки запросов (при учете времени загрузки обрабатывающих программ и данных) T_{seq}, T_{sp} и T_{pp} приведены значения коэффициентов относительного выигрыша от организации параллельных режимов загрузки и обработки заданий в кластере:

$$K_{seq/sp} = T_{seq} / T_{sp},$$
$$K_{seq/pp} = T_{seq} / T_{pp},$$
$$K_{sp/pp} = T_{sp} / T_{pp}.$$

Табл. 2.3

Число узлов	Оценки среднего времени выполнения задания в кластере из N узлов (мс)			Относительный выигрыш по времени выполнения задания		
	T_{seq}	T_{sp}	T_{pp}	$K_{seq/sp}$	$K_{seq/pp}$	$K_{sp/pp}$
1	613	615	617	1,0	1,0	1,0
2	1227	881	871	1,39	1,41	1,01
4	2439	1186	1156	2,06	2,12	1,02
8	4873	1543	1473	3,16	3,31	1,05
16	9768	1950	1796	5,0	5,50	1,09
32	19516	2451	2141	7,96	9,11	1,14
64	39021	3114	2484	12,5	15,7	1,25
128	78056	4104	2835	19,0	27,5	1,45
256	156059	5722	3172	27,3	49,2	1,82
512	312174	8632	3522	36,2	88,6	2,45
1024	624333	14090	3860	44,3	161,7	3,65

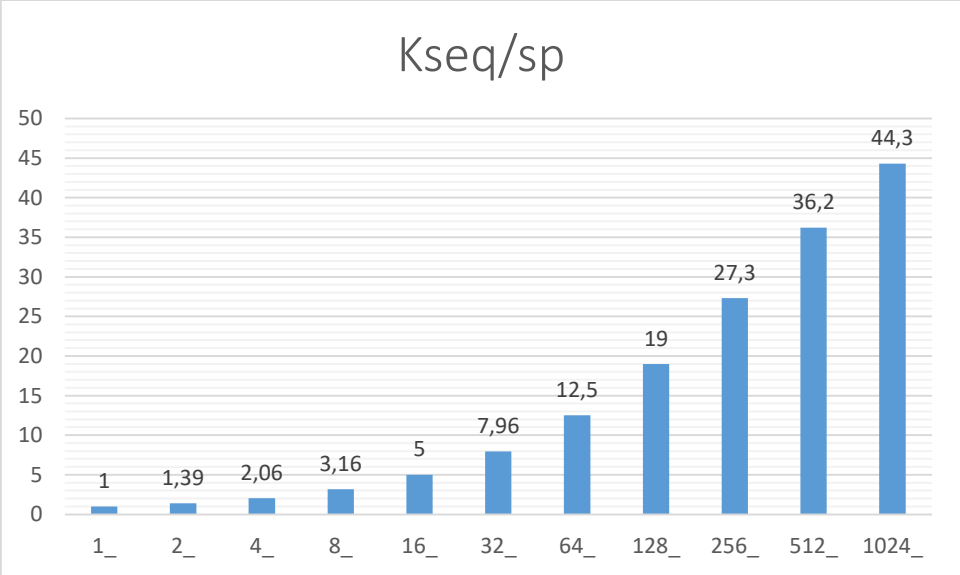


Рис. 2.7. Зависимость относительного выигрыша $K_{seq/sp}$ по времени подготовки и обработки запросов при переходе от последовательной загрузки и последовательной обработки запросов к последовательной загрузке и параллельной обработке запросов

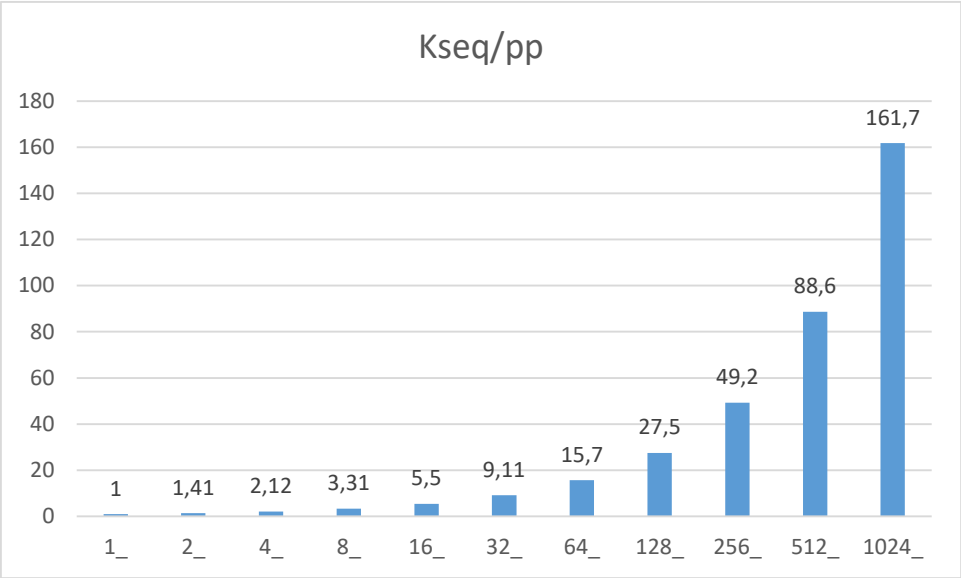


Рис. 2.8. Зависимость относительного выигрыша $K_{seq/pp}$ по времени подготовки и обработки запросов при переходе от последовательной загрузки и последовательной обработки запросов к параллельной загрузке и параллельной обработке запросов

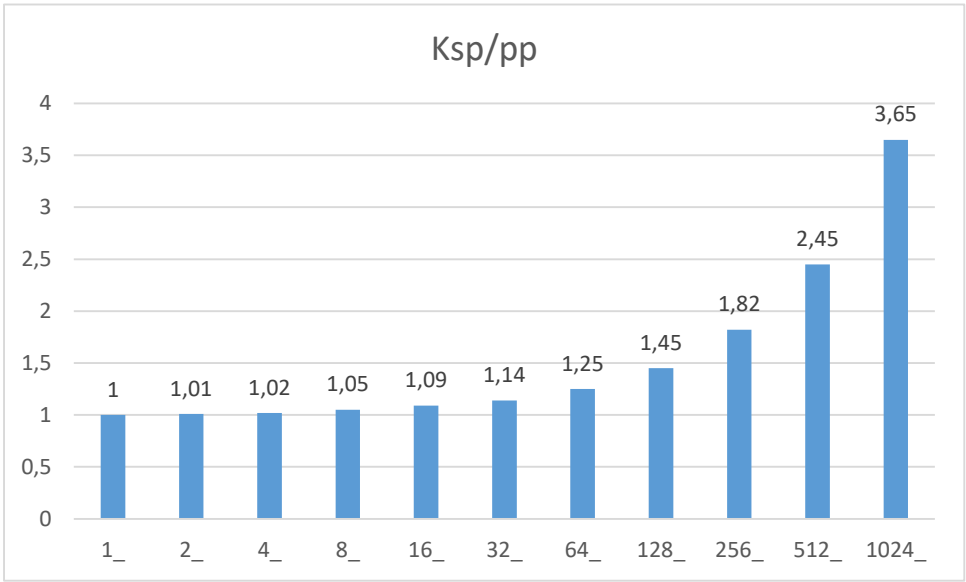
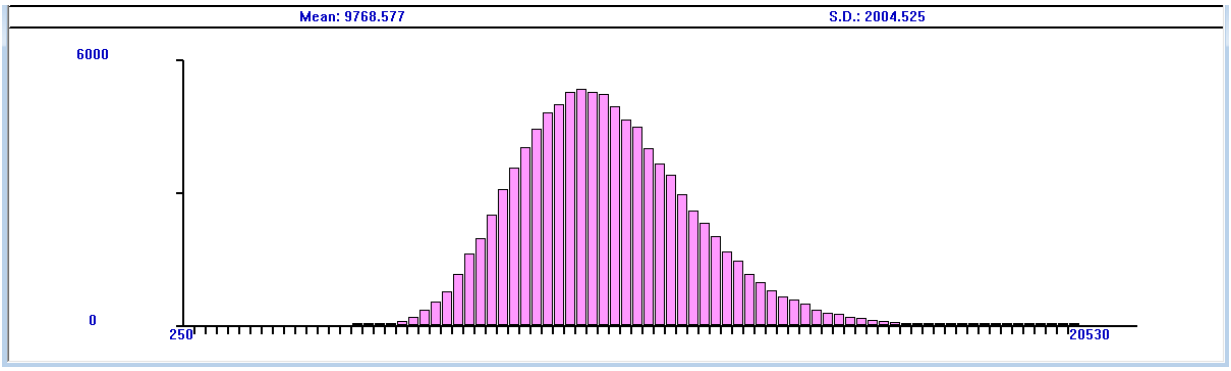


Рис. 2.9. Зависимость относительного выигрыша $K_{sp/pp}$ по времени подготовки и обработки запросов при переходе от последовательной загрузки и параллельной обработки запросов к параллельной загрузке и параллельной обработке запросов

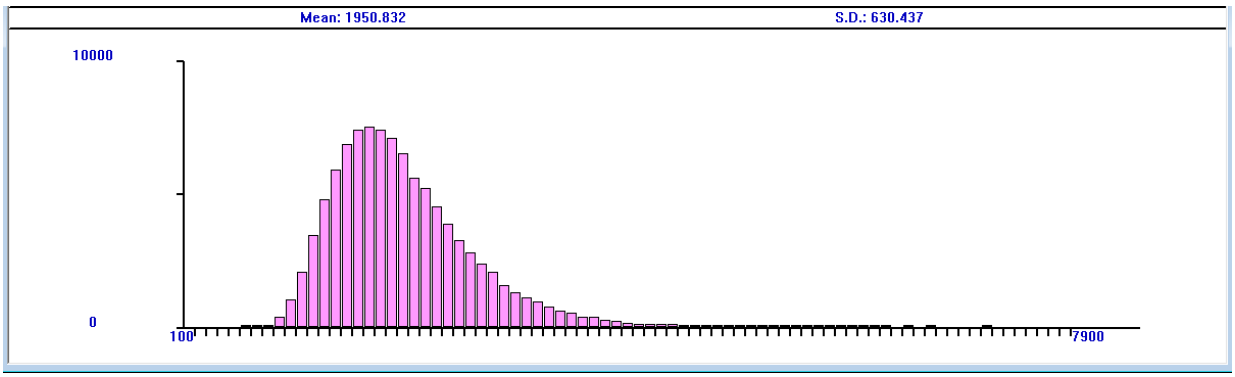
Диаграммы, показывающие зависимость коэффициентов относительного выигрыша при переходе к параллельным режимам работы приведены на рис. 2.7, 2.8 и 2.9.

Проведенный макроанализ производительности показывает, что при реализации указанных режимов рекомендуется организовать параллельную работу как при загрузке программ и данных на узлы кластера, так и, особенно, и при обработке запросов.

На рис. 2.10 представлены гистограммы для времени обработки заданий в предметно-ориентированной кластерной вычислительной системе при различных режимах использования.

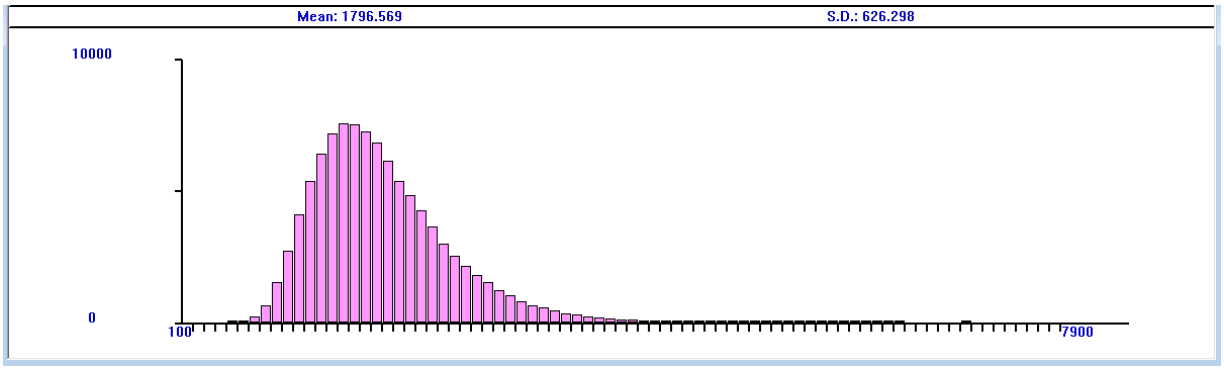


1) Кластер из 16 машин; режим работы «seq» последовательной загрузки и последовательного выполнения заданий; цена деления – 250 мс; объем выборки – 100000



2) Кластер из 16 машин; режим работы «sp» последовательной загрузки и параллельного выполнения заданий; цена деления – 100 мс; объем выборки – 100000

Рис. 2.10. (Начало). Гистограммы для времени обработки заданий в предметно-ориентированной кластерной вычислительной системе при различных ре-жимах использования



3) Кластер из 16 машин; режим работы «pp» параллельной загрузки и параллельного выполнения заданий; цена деления – 100 мс; объем выборки – 100000

Рис. 2.10. (Окончание). Гистограммы для времени обработки заданий в предметно-ориентированной кластерной вычислительной системе при различных режимах использования

Обращают на себя внимание длинные «хвосты» распределений времени обслуживания запросов, что, как и в предыдущих случаях, свидетельствует о наличии выбросов или экстремальных значений времени обработки запроса. Однако эти отклонения достаточно редки и не оказывают значительного влияния на работу вычислительного кластера в режимах «Резидент-агенты», «Собрание 1» и «Собрание 2».

2.9. Рекомендации по организации работы кластера при загрузке и выполнении рабочих программ и данных

При работе кластера в режиме «Круглый стол» реализуется диалоговое взаимодействие пар узлов, необходимость в широковещании отсутствует, поэтому нет необходимости в дополнительной информации к описанию. Работа вычислительного кластера в режимах «Резидент-агенты», «Собрание 1» и «Собрание 2» предполагает использование, в свою очередь, режимов запуска SPMD и MPMD параллельных программ [1, 38, 46]. При работе в режиме SPMD (*Single Program Multiple Data* – одна программа, множество данных)

создается единая для всех узлов кластера (или для всех потоков при многопоточных узлах) программа [1, 38]. Копии этой программы загружаются параллельно с управляющего узла или с сервера (рис. 1) на каждый рабочий узел кластера. У каждого узла (или потока) имеется собственный набор данных.

При работе в режиме MPMD (*Multiple Program Single Multiple Data* – множество программ, множество данных) для каждого узла (или потока) создается своя собственная программа. При параллельном выполнении этих программ каждая из них использует собственный набор данных [40, 41, 42].

Режим работы «Резидент-агенты» кластера возможно использовать в случае, когда на управляющем узле кластера («Резиденте») сформированы индивидуальные для каждого рабочего узла программы и данные, которые загружаются при отсутствии в кластере широковещательного режима. Такое явления может наблюдаться в TCP/IP сети, протоколы которой не допускают широковещание на сетевом уровне. Ситуация улучшается в случае, когда после загрузки программ и данных сразу начинается обработка на рабочем узле кластера, тогда в распределенном кластере устанавливается режим, занимающий промежуточное положение между режимом «Резидент-агенты» и «Собрание 1».

В режиме работы «Собрание 1» кластера индивидуальные данные загружаются последовательно с управляющего узла на рабочие узлы кластера по IP-адресам (которые имелись к моменту начала загрузки у пользователя и были заданы в программе клиента или управляющего узла). Сразу после завершения всех последовательных загрузок управляющий узел кластера, используя протокол ARP (*Address Resolution Protocol*) [67, 125] обеспечивает сопоставление IP- и MAC-адресов для последующей корректной широковещательной передачи данных в кластере, то есть при наличии MAC-адресов появляется возможность широковещательной параллельной загрузки копий одной и той же рабочей программы, которые (копии) должны затем параллельно обработать данные, загруженные ранее последовательно на каждый узел кластера.

В режиме «Собрание 2» предполагается, что MAC-адреса уже находятся в предварительно созданной временной таблице (кэш-памяти) соответствий IP и MAC-адресов для ускорения процесса передачи данных [67, 125]. Поэтому для широковещательной загрузки программ и данных в режимах SPMD и «Собрание 2» проблем не возникает.

Для справки: MAC-адрес (от англ. *Media Access Control* – управление доступом к среде передачи) – уникальный идентификатор, присваиваемый каждой единице сетевого оборудования или некоторым их интерфейсам в компьютерных сетях Ethernet и в некоторых других сетях стандартов IEEE 802 [67, 126].

2.10. Выводы по 2-й главе

1. Предложен логико-вероятностный подход к созданию моделей определяемой приложениями и программным обеспечением промежуточного уровня *middleware* функциональной архитектуры кластерных вычислительных систем, позволяющий ускорить создание имитационных моделей для ряда важных режимов использования кластера и осуществить переход к логико-алгебраическим формализованным спецификациям на программные приложения.

2. Предложена методология разработки функциональной архитектуры вычислительной системы кластерного типа, определяемой спецификациями в форме логико-вероятностных и родственных им логико-алгебраических моделей, что может ускорить подготовку кластера к эксплуатации в организации.

3. Построены новые имитационные логико-вероятностные и логико-алгебраические модели для ряда важных вариантов использования вычислительного кластера, проведены необходимые статистические эксперименты с данными моделями, давшие обоснования к реализациям соответствующему моделям программному обеспечению промежуточного уровня *middleware*.

В том числе:

3.1. Предложена и формализована новая исполнимая модель под условным названием «Круглый стол» о случайных парных взаимодействиях клиентов-собеседников в системе, построенной на базе вычислительного кластера.

Проведены статистические эксперименты с данной моделью, позволившие оценить среднее время обслуживания запросов (среднее время диалога, или парного взаимодействия, с учетом времени подготовки запроса и задержки из-за занятости портов коммутатора) и определить коэффициент замедления обработки запросов вследствие их влияния друг на друга. Например, при числе узлов кластера $N = 24$ значение коэффициента замедления диалога составило 4,9, а при $N = 1024$ значение этого коэффициента равно 8,5. Таким же образом получены зависимости среднего времени обработки запросов от числе портов коммутатора; например, при числе узлов $N = 1024$ и числе портов $N_P = 24$ среднее время парного взаимодействия равно 5,13 с.

3.2. Предложена методика перехода от имитационной логико-вероятностной модели к логико-алгебраической спецификации функционирования вычислительного кластера, что позволило обосновать его функциональную архитектуру и последующую реализацию режима «Круглый стол» в кластере.

3.3. Предложены, реализованы и исследованы режимы, логико-вероятностные и логико-алгебраические операционные модели работы вычислительных систем кластерного типа для сложных диалоговых взаимодействий пользователей: задачи «Резидент-агенты», «Собрание 1» и «Собрание 2» с вариантами организации параллельного доступа и обработки информации. Статистические эксперименты с моделями показали их работоспособность и адекватность функциональной архитектуре кластерных систем.

3.4. Использование предложенных моделей позволяет получить основные характеристики производительности и определяемую пользователем функциональную архитектуру вычислительных систем кластерного типа и определить пути повышения эффективности их функционирования.

ГЛАВА 3. АВТОМАТНЫЕ И ЛОГИКО-АЛГЕБРАИЧЕСКИЕ ИСПОЛНИМЫЕ МОДЕЛИ ФУНКЦИОНИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ КЛАСТЕРНОГО ТИПА

Рассмотрено построение автоматных и логико-вероятностных моделей кластерных вычислительных систем, нацеленные на создание на этой основе программных средств, являющихся основой функциональной архитектуры данных систем. Предложены новые детализированные автоматные, вероятностные автоматные и логико-алгебраические операционные модели вычислительных систем кластерного типа на микроуровне, отличающиеся исполнимым характером и позволяющие произвести проектирование приложений на уровне сетевых операций, а также дать оценку производительности кластера в целом.

Автоматное моделирование и программирование в настоящее время считается одной из технологий, в существенной степени сокращающей сроки составления программ и упрощающей их тестирование. Системы канонических уравнений (СКУ), используемые в качестве начального языка для задания частичных конечных автоматов, также позволяют создавать на их основе программы прикладного, промежуточного и системного уровней. В разработку методов автоматного моделирования и программирования большой вклад внесли профессора А.А. Шалыто (г. Санкт-Петербург), Н.П. Вашкевич (г. Пенза) и ряд других ученых.

Вводится понятие логико-вероятностной модели «темпоральная вероятностная СКУ», которая позволит получить наглядную формализацию и реализацию автоматных моделей и рабочих программ, характерных для кластерных и других приложений, например, с конвейерным параллелизмом, и в существенной степени сократить число «инкрементных» сложений при перечислении моментов дискретного времени.

Рассмотрены вопросы построения исполнимых логико-алгебраических моделей для различных приложений. Эти модели предполагается использовать в качестве формализованных спецификаций при разработке реальных

приложений. Рассмотрены логико-алгебраические операционные выражения (ЛАОВ) для моделей функционирования приложений кластерных систем в режиме SPMD (*Single Program, Multiple Data* – одиночный поток программ, множественный поток данных) и в режиме MPMD (*Multiple Programs, Multiple Data* – множественный поток программ, множественный поток данных) [1]. Оба этих режима допускают вариативность исполнения программ в широких пределах.

На основе исполнимых моделей рассмотрено выполнение на кластере последовательно-параллельных и параллельно-конвейерных приложений с реализаций произвольных алгоритмов. В результате ожидается сокращение сроков, отводимых на предметную ориентацию кластерных вычислительных систем на основных уровнях абстракции, от концептуального представления до деталей реализации. На основе статистических экспериментов с исполнимыми моделями получены основные характеристики, характеризующие эффективную работу кластеров при различных нагрузках. Проведенные эксперименты с моделями кластерных приложений и последующее программирование прототипных работающих приложений показали реализуемость и эффективность предлагаемого подхода.

Сделан вывод о том, что выбор системной и функциональной архитектуры вычислительного кластера должен определяться не столько указанными производителем пиковыми характеристиками коммуникационной аппаратуры, сколько реальными показателями, достигаемыми на уровне приложений пользователей и режимов использования кластера.

3.1. Автоматные модели кластерных приложений на основе языка систем канонических уравнений (СКУ)

Вычислительная система в работе [1] определяется на абстрактном уровне как совокупность работающих во времени функциональных устройств. При оценке качества работы предлагается абстрагироваться от содержательной части выполняемых операций и учитывать работу функциональных устройств в системе отсчета времени. Поэтому для анализа функционирования

вычислительных кластеров будет полезно строить формальные модели. Кроме того, как следует из курса «Вычислительное дело и кластерные системы» [68], «на практике не столько важны указанные производителем пиковые характеристики коммуникационной аппаратуры, сколько реальные показатели, достигаемые на уровне приложений пользователей». Из этого высказывания следует, что выбор системной и функциональной архитектуры вычислительного кластера должен определяться в основном приложениями и режимами использования кластера, в том числе реализуемыми на промежуточном уровне *middleware*. Поэтому условно можно считать часть прикладного программного обеспечения и программного обеспечения промежуточного уровня системным программным обеспечением, определяющим функциональность всего кластера компьютеров.

Главным эффектом от интерпретации предлагаемых моделей является возможность их использования в качестве формализованных спецификаций при описании распараллеленных процессов в кластерных вычислительных системах и сетях на уровне задач, данных, алгоритмов и машинных инструкций, то есть на основных уровнях абстракции, от концептуального представления до деталей реализации. Выбор приведенных далее примеров моделей, основанных на схемах программ, основан на соблюдении высокого уровня общности: алгоритмы должны содержать все базовые алгоритмические конструкции, позволяющие реализовывать последовательности, ветвления и циклы; должна иметься возможность переинтерпретации видов распараллеливания – на уровне задач, на уровне данных, на уровне алгоритмов и на уровне команд машинного уровня с возможностью чередования последовательных однопотоковых частей программы с многопотоковыми параллельными участками, инвариантные относительно платформ реализации – многоядерные процессоры, видеокарты, сети компьютеров.

Однако исследовать реальную работу приложений можно только на работающем кластере. Разрешить проблему на предварительных этапах с мень-

шими затратами усилий и средств возможно при помощи использования исполнимых формальных моделей, на основе которых следует построить имитационные модели работы кластера. Указанные модели могут включать характерные или упрощенные фрагменты реальных приложений.

На данном этапе построения моделей не рассматривается семантика данных и операций, то есть для сохранения общности моделей значения переменных и символов операций не интерпретированы. Предполагается, что методы создания и интерпретации моделей можно далее использовать при создании действующих интерпретированных приложений в случае, когда кластер будет введен в эксплуатацию. В этом случае формальные модели могли бы играть роль формализованных спецификаций.

Одной из удобных для последующего использования в этих целях моделей являются язык граф-схем алгоритмов, конечные автоматы и логико-алгебраические модели на основе логики предикатов первого порядка. В настоящем подразделе предлагается использовать модель конечного частичного автомата Мура [127, 128]. Эта модель также хорошо известна по работам в области микропрограммирования [129, 130]. На рис. 3.1 и рис. 3.2 представлены выбранные для иллюстрации создания моделей приложения примеры граф-схем алгоритмов $ГСА_1$ и $ГСА_2$. Основные критерии для выбора – обычные требования корректности ГСА и наличие последовательностей операторов и ветвлений; это означает, что рассматриваемые далее методы пригодны для любых ГСА. $ГСА_1$ (рис. 3.1) содержит операторные вершины (далее просто операторы) A_0, A_1, A_2, …, A_{16}, A_{27}, A_K.

Кроме того, $ГСА_1$ содержит параллельные фрагменты, представленные структурированными операторами C_1, C_2, …, C_8, каждому из которых соответствует «внутренняя» копия $ГСА_2$ (рис. 3.2); каждая копия, или клон, содержит локальные операторы A_{17}, A_{18}, A_{19}, …, A_{25}, A_{26}.

Обе ГСА содержат условные вершины (далее просто логические условия) x_1, x_2, …, x_5 ($ГСА_1$) и x_6, x_7, …, x_{10} ($ГСА_2$). Символы условий рассматрива-

ются как имена унарных предикатов. Значения логических условий – 0 (истина, ***true***) или 1 (ложь, ***false***) вычисляются по завершении выполнения операторов, в том числе операторов ввода входных условий (входных сигналов, или входных символов, частичного автомата).

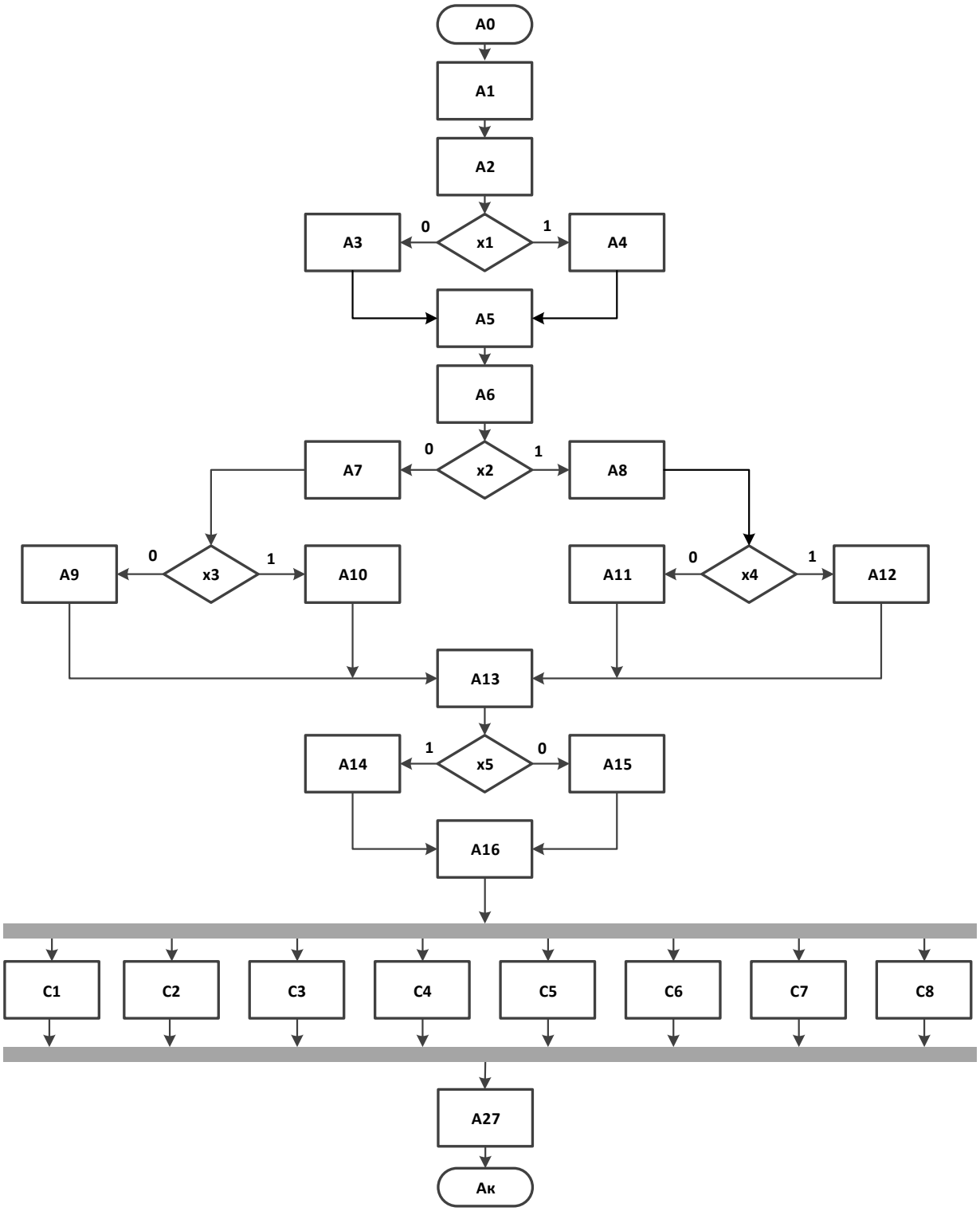


Рис. 3.1. Граф-схема алгоритма $ГСА_1$ работы приложения для кластера (*C1*, *C2*, …, *C8* – параллельные структурированные участки алгоритма)

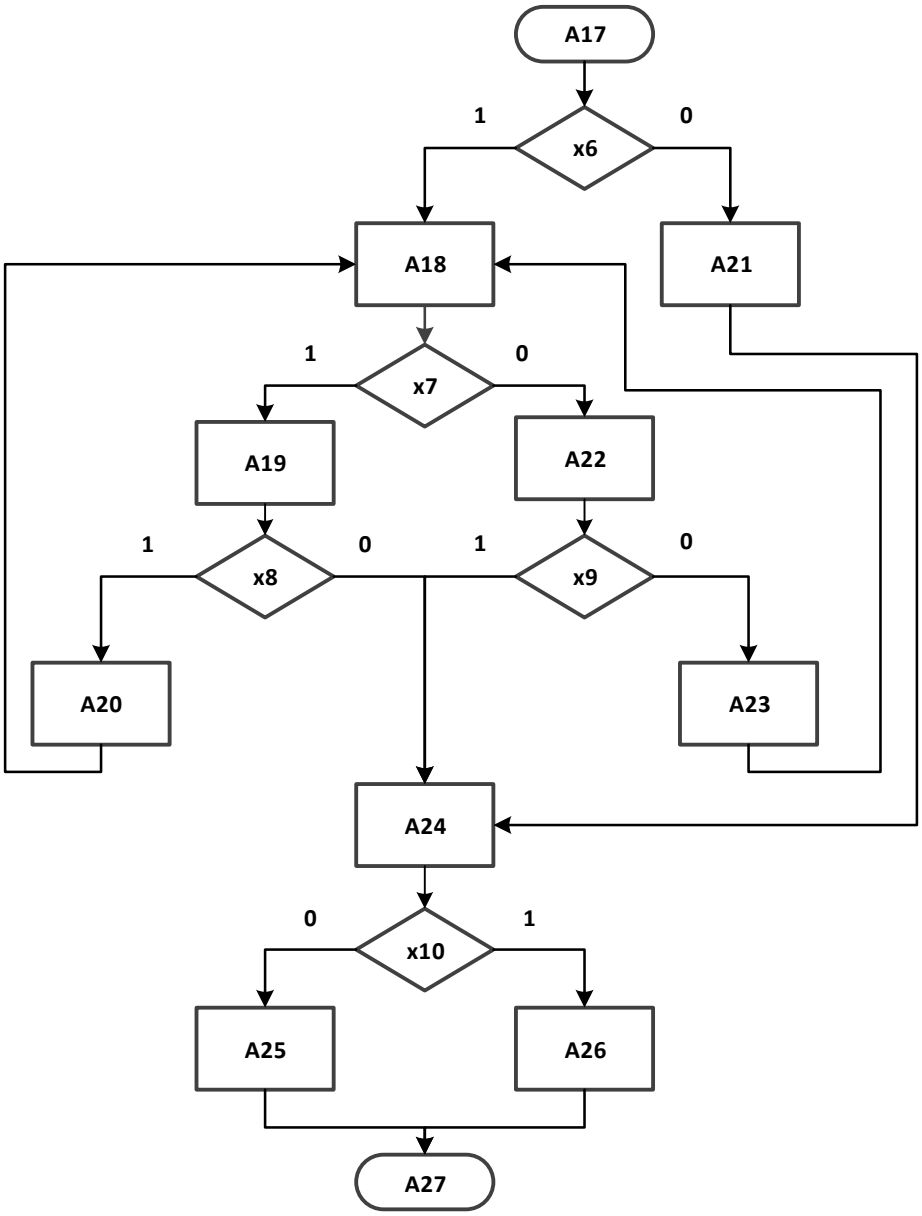


Рис. 3.2. Граф-схема алгоритма $ГСА_2$ для одной копии параллельного участка приложения для кластера

В качестве начального языка для задания частичных автоматов выбраны системы канонических уравнений [130, 131], которыми описываются переходы из одних состояний в другие. Допускаемые параллельные переходы соответствуют, например, представлению параллельных участков в модифицированных логических схемах алгоритмов (МЛСА), известному из работ по микропрограммированию [132]; в МЛСА параллельные участки считаются частными ЛСА и допускают простую переинтерпретацию в графическую форму ГСА. Языки параллельных ГСА использовались также в работах [133, 134]. Структуризация состояний иерархических автоматов была предложена ранее в ряде работ [135, 136, 137].

В предлагаемых далее автоматных СКУ-моделях (СКУ – система канонических уравнений [31, 130, 131]) использованы следующие понятия. Операторам поставлены во взаимно-однозначное соответствие так называемые частные события, представленные унарными предикатами вида $A_i(t)$, определенными на множестве значений дискретного времени t. Частичные входные переменные, или входные условия, представлены унарными предикатами вида $x_j(t)$, также определенными на множестве значений дискретного времени t. Введены также унарные предикаты вида $z_k(t\ )$, которые могут принимать истинные значения только после того, как соответствующие им события вида $A_k(t)$ уже произошли, что соответствует тому факту, что оператор A_k завершил свою работу. Таким образом, при $z_k(t)= 0$ (ложь) событие $A_k(t)$ сохраняется, а при $z_k(t\ )= 1$ (истина) – не сохраняется. Первое условие для события $A_k(t)$ позволяет продлить его выполнение, а при выполнении второго, противоположного условия, событие $A_k(t)$ завершается. Условию зарождения события соответствует переход из предшествующего ему события. Остальные особенности построения СКУ удобно пояснить на примерах перехода от ГСА к СКУ.

На рис. 3.3 представлен граф переходов состояний автоматной модели последовательного приложения, построенный путем перехода от $ГСА_1$ и $ГСА_2$; выделен участок C_0, предназначенный для последующего клонирования

при переходе к моделированию приложения, соответствующего режиму работы кластера SPMD. Данный граф, как потребуется в дальнейшем, может также рассматриваться как последовательная композиция двух частичных автоматов: первому автомату соответствуют состояния a_0, a_1, a_2, …, a_{16}, a_{28}, а второму – состояния a_{17}, a_{18}, …, a_{27}. В автоматной СКУ-модели эти состояния представлены унарными предикатами $A_0(t)$, $A_1(t)$, $A_2(t)$, …, $A_{16}(t)$, A_{28}, (t) и $A_{17}(t)$, $A_{18}(t)$, …, $A_{27}(t)$. На рис. 3.1 и рис. 3.2 использованы общепринятые для граф-схем алгоритмов обозначения для логических условий*:* x_1, x_2, …, x_{10}. В автоматной СКУ-модели для обозначения этих входных условий используются унарные предикаты $x_1(t)$, $x_2(t)$, …, $x_{10}(t)$, введен также унарный предикат $x_0(t)$ для обозначения начального (пускового) условия. Здесь предметная переменная t пробегает по множеству значений автоматного времени.

Систему канонических уравнений возможно интерпретировать как систему продукционных правил, предназначенную для представления знаний в автоматных моделях. Продукции возможно использовать для представления знаний, которые могут принимать формы правил вида *посылка* $\rightarrow$ *заключение*, *условие* $\rightarrow$ *действие*. Левая часть правила называется антецедентом, правая – консеквентом. Антецедент – это посылка правила (условная часть), состоит из элементарных высказываний, использующих логические символы И, ИЛИ, НЕ; консеквент (заключение) включает одно или несколько предложений, которые выражают либо некоторый факт, либо указание на определенное действие, подлежащее исполнению [138, 139, 140].

Набор продукций образует *продукционную систему*, для которой задаются специальные процедуры выбора продукций и выполнение той или иной продукции из числа выбранных.

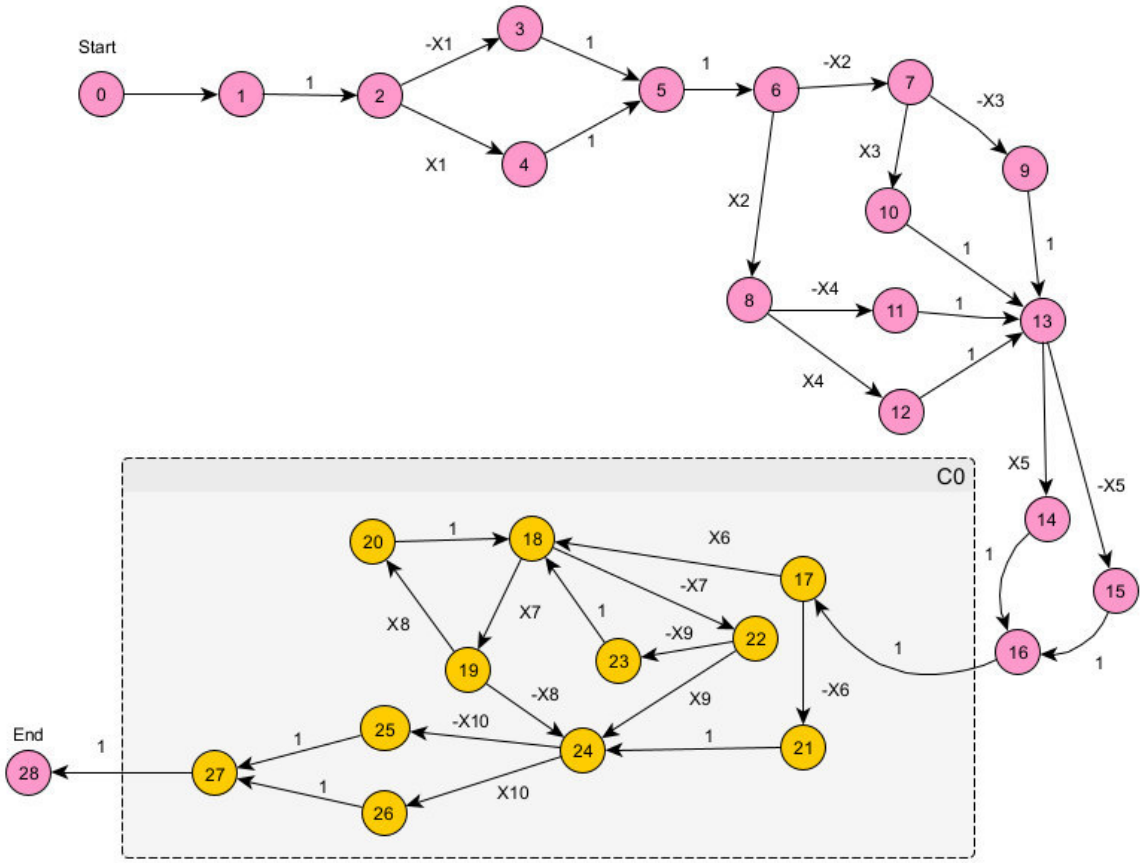


Рис. 3.3. Граф переходов состояний автоматной модели последовательного приложения; выделен участок C_0, предназначенный для последующего клонирования при переходе к режиму SPMD (*Single Program, Multiple Data*)

Отличительной особенностью СКУ, рассматриваемой в качестве разновидности продукционной системы, является размещение условной части (антецедента) справа, а действия, или заключения, (консеквента) – слева и следовательно, направление вывода – справа налево. Это по большей части связано с теорией и практикой синтеза микропрограммных автоматов и с представлением наряду с каноническими уравнениями функций возбуждения элементарных автоматов (*D*-триггеров, или элементов задержек) при унитарном кодировании состояний конечного частичного автомата [31, 129, 130]. В дальнейшем понятие «продукция» будет использовано также и при построении логико-алгебраических моделей систем кластерного типа.

3.2. Автоматная СКУ-модель последовательной части приложения

На рис. 3.3 был представлен граф переходов состояний автоматной модели последовательного приложения. Часть состояний a_1, a_2, …, a_{16}, a_{28} получена путем разметки состояний автомата Мура на $ГСА_1$, представленной на рис. 3.1. Остальная часть состояний a_{17}, a_{18}, …, a_{27} принадлежит подграфу C_0, который построен путем разметки состояний автомата Мура на $ГСА_2$

На данном этапе рекуррентные предикатные уравнения СКУ $\boldsymbol{SP_{Seq}}$ для $ГСА_1$ составлены без учета структурированных операторов C_1, C_2,…, C_8 и, соответственно, без структурированных состояний a_{17}, a_{18}, …, a_{27} автомата, то есть модель пока охватывает только операторы A_0, A_1, …, A_{16}:

$$A_0(t+1) = A_0(t\ ) \ \&\ \neg x_0(t) \vee x_{begin}(t);$$
$$A_1(t+1) = A_0(t) \ \&\ x_0(t) \vee A_1(t) \ \&\ \neg z_1(t);$$
$$A_2(t+1) = A_1(t) \ \&\ z_1(t) \vee A_2(t) \& \neg z_2(t);$$
$$A_3(t+1) = A_2(t) \ \&\ z_2(t) \ \&\ \neg x_1(t) \vee A_3(t) \ \&\ \neg z_3(t);$$
$$A_4(t+1) = A_2(t) \ \&\ z_2(t) \ \&\ x_1(t) \vee A_4(t) \ \&\ \neg z_4(t);$$
$$A_5(t+1) = A_3(t) \ \&\ z_3(t) \vee A_4(t) \ \&\ z_4(t) \vee A_5(t) \ \&\ \neg z_5(t);$$
$$A_6(t+1) = A_5(t) \ \&\ z_4(t) \vee A_6(t) \ \&\ \neg z_6(t);$$
$$A_7(t+1) = A_6(t) \ \&\ z_6(t) \ \&\ \neg x_2(t) \vee A_7(t) \ \&\ \neg z_7(t);$$
$$A_8(t+1) = A_6(t) \ \&\ z_6(t) \ \&\ x_2(t) \vee A_8(t) \ \&\ \neg z_8(t);$$
$$A_9(t+1) = A_7(t) \ \&\ z_7(t) \ \&\ \neg x_3(t) \vee A_9(t) \ \&\ \neg z_9(t);$$
$$A_{10}(t+1) = A_7(t) \ \&\ z_7(t) \ \&\ x_3(t) \vee A_{10}(t) \ \&\ \neg z_{10}(t);$$
$$A_{11}(t+1) = A_8(t) \ \&\ z_8(t) \ \&\ \neg x_4(t) \vee A_{11}(t) \ \&\ \neg z_{11}(t);$$
$$A_{12}(t+1) = A_8(t) \ \&\ z_8(t) \ \&\ x_4(t) \vee A_{12}(t) \ \&\ \neg z_{12}(t);$$
$$A_{13}(t+1) = A_9(t) \ \&\ z_9(t) \vee A_{10}(t) \ \&\ z_{10}(t) \vee A_{11}(t) \ \&\ z_{11}(t) \vee$$
$$\vee A_{12}(t) \ \&\ z_{12}(t) \vee A_{13}(t) \ \&\ \neg z_{13}(t);$$
$$A_{14}(t+1) = A_{13}(t) \ \&\ z_{13}(t) \ \&\ x_5(t) \vee A_{14}(t) \ \&\ \neg z_{14}(t);$$
$$A_{15}(t+1) = A_{13}(t) \ \&\ z_{13}(t) \ \&\ \neg x_5(t) \vee A_{15}(t) \ \&\ \neg z_{15}(t);$$
$$A_{16}(t+1) = A_{14}(t) \ \&\ z_{14}(t) \vee A_{15}(t) \ \&\ z_{15}(t) \vee A_{16}(t) \ \&\ \neg z_{16}(t).$$

Копии, или клоны, модуля приложения, составленного по СКУ $\boldsymbol{SP_{Seq}}$ для $ГСА_1$, загружаются на все вычислительные узлы кластера и выполняются в параллельном режиме, обрабатывая одни и те же данные или осуществляют ввод однотипных данных. Автоматная модель предполагает различные времена исполнения событий, соответствующих операторам приложения. При дальнейшем описании уравнений СКУ термины «событие» и «состояние» для сокращения описания будут рассматриваться как синонимы.

Ниже приведены описания некоторых ключевых уравнений из приведенной СКУ. Начальное уравнение имеет следующий вид:

$$A_0(t+1) = A_0(t\ ) \ \&\ \neg x_0(t) \vee x_{begin}(t).$$

В соответствии с этим уравнением при появлении истинного значения входной переменной («сигнала») $x_{begin}(t) = \boldsymbol{true}$ автомат в следующем такте устанавливается начальное событие $A_0(t+1) = \boldsymbol{true}$, что соответствует его зарождению. Это событие сохраняется до тех пор, пока истинно условие его сохранения $A_0(t) \ \&\ \neg\ x_0(t)$ при $A_0(t) = \boldsymbol{true}$ и $x_0(t) = \boldsymbol{false}$. Далее, при поступлении входного сигнала $x_0(t) = \boldsymbol{true}$ в автомате создается истинное условие $A_0(t) \ \&\ x_0(t)$ зарождения нового события $A_1(t+1)$:

$$A_1(t+1) = A_0(t) \ \&\ x_0(t) \vee A_1(t) \ \&\ \neg z_1(t).$$

Это событие сохраняется до тех пор, пока истинно условие его сохранения $A_1(t) \ \&\ \neg z_1(t)$. Оно завершится ($A_1(t+1) = \boldsymbol{false}$, то есть высказывание станет ложным) при выработке оператором A_1 признака $z_1(t) = \boldsymbol{true}$ по окончании своей работы. Как видно из рекуррентных предикатных уравнений данной СКУ $\boldsymbol{SP_{Seq}}$, событие истинности антецедента (правого высказывания) происходит в фиксированный момент времени t, а событие истинности консеквента (левого высказывания) происходит в следующий момент времени $t + 1$.

3.3. Автоматная СКУ-модель работы одного из параллельных участков (клонов) приложения

На рис. 3.3 представлен граф переходов состояний автоматной модели последовательного приложения; выделен участок C_0, предназначенный для последующего клонирования при переходе к режиму SPMD (*Single Program, Multiple Data*). Участок C_0 получен путем разметки состояний a_{17}, a_{18}, …, a_{27} автомата Мура на ГСА$_2$, представленной на рис. 3.2.

Рекуррентные предикатные уравнения СКУ $\boldsymbol{MD_{Clon}}$ для ГСА$_2$:

$A_{17}(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee A_{17}(t) \,\&\, \neg z_{17}(t);$

$A_{18}(t+1) = A_{17}(t) \,\&\, z_{17}(t) \,\&\, x_6(t) \vee A_{20}(t) \,\&\, z_{20}(t)$

$\vee A_{23}(t) \,\&\, z_{23}(t) \vee A_{18}(t) \,\&\, \neg z_{18}(t);$

$A_{19}(t+1) = A_{18}(t) \,\&\, z_{18}(t) \,\&\, x_7(t) \vee A_{19}(t) \,\&\, \neg z_{19}(t);$

$A_{20}(t+1) = A_{19}(t) \,\&\, z_{19}(t) \,\&\, x_8(t) \vee A_{20}(t) \,\&\, \neg z_{20}(t);$

$A_{21}(t+1) = A_{17}(t) \,\&\, z_{17}(t) \,\&\, \neg x_6(t) \vee A_{21}(t) \,\&\, \neg z_{21}(t);$

$A_{22}(t+1) = A_{18}(t) \,\&\, z_{18}(t) \,\&\, \neg x_7(t) \vee A_{22}(t) \,\&\, \neg z_{22}(t);$

$A_{23}(t+1) = A_{22}(t) \,\&\, z_{22}(t) \,\&\, \neg x_9(t) \vee A_{23}(t) \,\&\, \neg z_{23}(t);$

$A_{24}(t+1) = A_{19}(t) \,\&\, z_{19}(t) \,\&\, \neg x_8(t) \vee A_{22}(t) \,\&\, z_{22}(t) \,\&\, x_9(t) \vee$

$\vee A_{21}(t) \,\&\, \neg z_{21}(t) \vee A_{24}(t) \,\&\, \neg z_{24}(t);$

$A_{25}(t+1) = A_{24}(t) \,\&\, z_{24}(t) \,\&\, \neg x_{10}(t) \vee A_{25}(t) \,\&\, \neg z_{25}(t);$

$A_{26}(t+1) = A_{24}(t) \,\&\, z_{24}(t) \,\&\, x_{10}(t) \vee A_{26}(t) \,\&\, \neg z_{26}(t);$

$A_{27}(t+1) = A_{25}(t) \,\&\, z_{25}(t) \vee A_{26}(t) \,\&\, z_{26}(t) \vee A_{27}(t) \,\&\, \neg z_{27}(t).$

Совместно используемые автоматная СКУ-модель $\boldsymbol{SP_{Seq}}$ и одна копия СКУ-модели $\boldsymbol{MD_{Clon}}$**,** взятые вместе**,** задают единую автоматную модель, обозначенную как $\boldsymbol{SP_{Seq}}$*$\boldsymbol{MD_{Clon}}$, где символ звездочки обозначает операцию объединения двух СКУ в одну общую. Граф переходов состояний для этой модели

представлен на рис. 3.3. Другая интерпретация, как было указано ранее, позволяет считать граф переходов состояний $\boldsymbol{SP_{Seq}}$*$\boldsymbol{MD_{Clon}}$ последовательной композицией частичных автоматов.

3.4. Последовательно-параллельная композиция (сеть) автоматов, определяющая работу компьютеров кластера

Граф-схема алгоритма $ГСА_1$ на рис. 3.1 содержит параллельные участки, обозначенные сокращенно как структурированные операторы C_1, C_2, …, C_8. При полном одноуровневом представлении каждый из этих операторов заменяется граф-схемой алгоритма $ГСА_2$ на рис. 3.2 и при программной реализации выполняется независимо от других на «своем» компьютере кластера. Полная сетевая СКУ-модель последовательно-параллельной сети автоматов представлена выражением следующего вида:

$$\boldsymbol{SP_{Seq}}*(\boldsymbol{ParReplicate}(1..8)\boldsymbol{MD_{Clon}}).$$

Граф переходов состояний для данной сети частичных автоматов со структурированными состояниями $C_1 – C_8$ представлен на рис. 3.4. При построение сетевой СКУ-модели необходимо составить уравнения, включающие структурированные события. Каждое структурированное событие представляет вложенный частичный автомат. Использование иерархических конечных автоматов является принципиально важным способом проектирования программного обеспечения и соответствует понятию «подпрограмма» в составе главной программы.

Достоинством метода формализации алгоритмов с использованием систем канонических уравнений является компактное логическое описание функций переходов [130, 131]. Аналогично структурированным состояниям *C1–C8* вводятся одноименные структурированные события. «Вложением» в каждое структурированное событие являются уже составленные автоматные СКУ $\boldsymbol{MD_{Clon}}$.

Зарождение и продолжение событий $C_1 – C_8$ (инициирование и параллельная работа независимых друг от друга программных модулей) описываются следующей СКУ $\boldsymbol{S_C}$:

СКУ $\boldsymbol{S_C}$*:*

$$C_1(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee C_1(t) \,\&\, \neg\, w_1(t);$$

$$C_2(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee C_2(t) \,\&\, \neg\, w_2(t);$$

$$C_3(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee C_3(t) \,\&\, \neg\, w_3(t);$$

$$C_4(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee C_4(t) \,\&\, \neg\, w_4(t);$$

$$C_5(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee C_5(t) \,\&\, \neg\, w_5(t);$$

$$C_6(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee C_6(t) \,\&\, \neg\, w_6(t);$$

$$C_7(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee C_7(t) \,\&\, \neg\, w_7(t);$$

$$C_8(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee C_8(t) \,\&\, \neg\, w_8(t).$$

Каждое из событий $C_1 – C_8$ зарождается при истинности составного высказывания $A_{16}(t) \,\&\, z_{16}(t)$, то есть является следствием успешного завершения события A_{16}. Каждое из этих событий C_i сохраняется, пока не будет выполнено завершающее условие $w_i(t)$, $i = 1, 2, \ldots, 8$. События $C_1 – C_8$ начинаются одновременно, но не обязательно заканчиваются одновременно, поскольку завершающие условия могут не зависеть друг от друга. Однако переход к событию A_{28} должен произойти только после завершения всех событий $C_1 – C_8$. Поэтому далее в СКУ-модели должны следовать события, определяющие барьерную синхронизацию и состоящие в ожидании завершения событий $C_1 – C_8$ в каждой из ветвей, и последующего зарождения и сохранения событий-индикаторов $D_1 – D_8$ завершения работы всех ветвей.

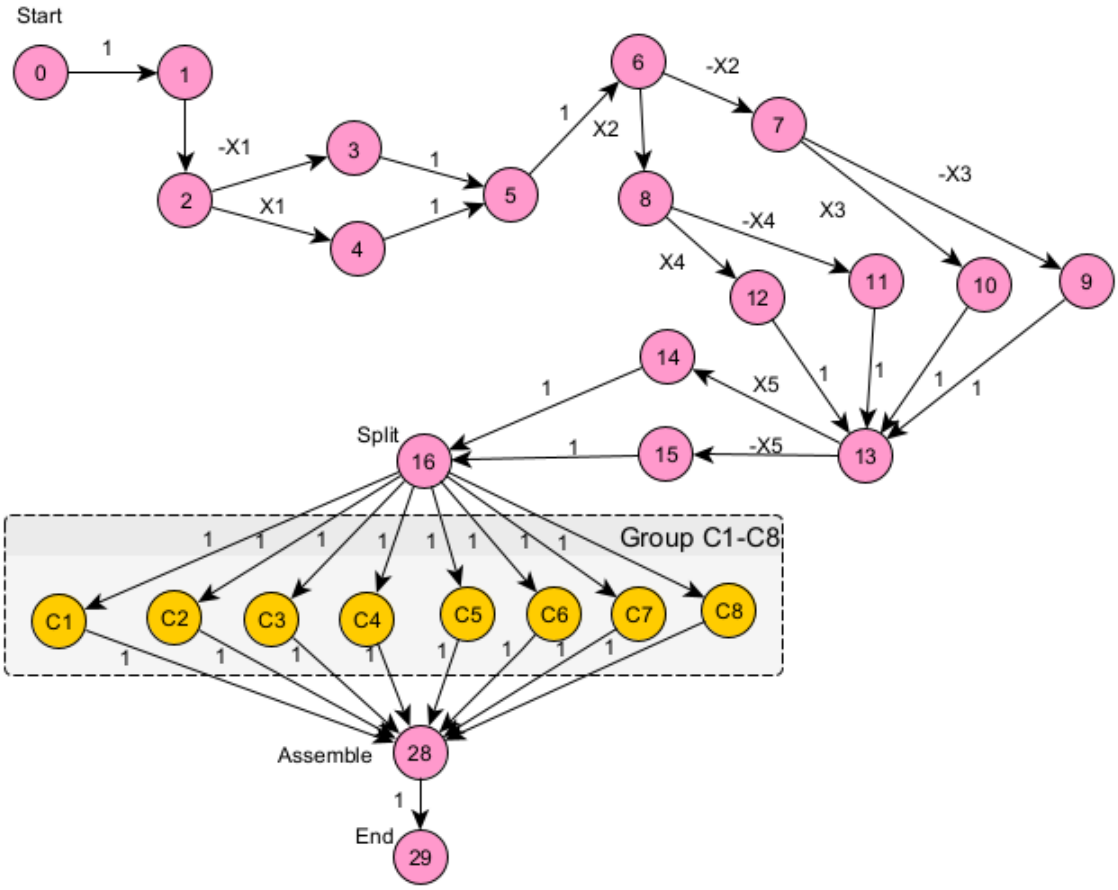


Рис. 3.4. Последовательно-параллельная сеть автоматов с вложенными состояниями выполнения приложения в кластере в режиме обработки SPMD (*Single Program, Multiple Data*)

Система канонических уравнений $\boldsymbol{S_D}$, описывающая зарождение и сохранение этих событий, имеет следующий вид:

СКУ $\boldsymbol{S_D}$*:*

$$D_1(t+1) = C_1(t) \,\&\, w_1(t) \vee D_1(t) \,\&\, \neg D_9(t);$$
$$D_2(t+1) = C_2(t) \,\&\, w_2(t) \vee D_2(t) \,\&\, \neg D_9(t);$$
$$D_3(t+1) = C_3(t) \,\&\, w_3(t) \vee D_3(t) \,\&\, \neg D_9(t);$$
$$D_4(t+1) = C_4(t) \,\&\, w_4(t) \vee D_4(t) \,\&\, \neg D_9(t);$$
$$D_5(t+1) = C_5(t) \,\&\, w_5(t) \vee D_5(t) \,\&\, \neg D_9(t);$$
$$D_6(t+1) = C_6(t) \,\&\, w_6(t) \vee D_6(t) \,\&\, \neg D_9(t);$$
$$D_7(t+1) = C_7(t) \,\&\, w_7(t) \vee D_7(t) \,\&\, \neg D_9(t);$$
$$D_8(t+1) = C_8(t) \,\&\, w_8(t) \vee D_8(t) \,\&\, \neg D_9(t).$$

Следующее единственное уравнение, обозначенное дополнительно как система СКУ $\boldsymbol{D_9}$, описывает ожидание наступления всех событий-индикаторов $D_1 – D_8$ завершения параллельной работы всех ветвей:

$$D_9(t+1) = D_1(t) \,\&\, D_2(t) \,\&\, D_3(t) \,\&\, D_4(t) \,\&$$
$$\&\, D_5(t) \,\&\, D_6(t) \,\&\, D_7(t) \,\&\, D_8(t) \vee D_9(t) \,\&\, \neg v_9(t).$$

Уравнения, описывающие переход к завершающим событиям A_{28} A_{29} и далее к событию A_1 управляющего модуля:

$$A_{28}(t+1) = D_9(t) \,\&\, v_9(t) \vee A_{28}(t) \,\&\, \neg z_{28}(t);$$
$$A_{29}(t+1) = A_{28}(t) \,\&\, z_{28}(t) \vee A_{29}(t) \,\&\, \neg z_{29}(t).$$
$$A_1(t+1) = A_{29}(t) \,\&\, z_{29}(t) \vee A_1(t) \,\&\, \neg z_1(t).$$

В целях использования уравнений для событий A_{28} и A_{29} в объединенном выражении для сети частичных автоматов, они обозначены как отдельные СКУ $\boldsymbol{A_{28}}$ и $\boldsymbol{A_{29}}$ соответственно.

Для реализации детализированной сетевой СКУ-модели выполнения приложения в кластере в режиме последовательно-параллельной обработки SPMD все уравнения необходимо объединить в следующей последовательности:

$$\text{SPMD: } \boldsymbol{SP_{Seq}{*}S_C{*}S_D{*}D_9{*}A_{28}{*}A_{29}},$$

причем инициирование описанных ранее параллельных реплик

$$\boldsymbol{ParReplicate}\textbf{(1..8)}\boldsymbol{MD_{Clon}}$$

осуществляется при наступлении событий $C_1 – C_8$, определяемых подсистемой СКУ $\boldsymbol{S_C}$, а завершение выполнения этих реплик происходит при наступлении событий $D_1 – D_8$, определяемых подсистемой СКУ $\boldsymbol{S_D}$. При работе реплик расширяется диапазон изменения переменной t, отсчитывающей системное время.

Результирующая общая СКУ-модель сети автоматов выполнения приложения в кластере в режиме последовательно-параллельной обработки *SPMD* относится к классу исполнимых моделей. Она легко программируется на алгоритмических языках, содержащих операторы передачи сообщений, а также на языке Ассемблера для микроконтроллеров и языке микропрограммирования. На ее основе построена имитационная модель, позволяющую исследовать функционирование кластерной системы на микроуровне.

Рассматриваемые в настоящей работе сети частичных конечных автоматов состоят из автоматов, моделирующих связанные интерфейсом передачи сообщений по входам и выходам программные модули приложения вычислительного кластера. Каждый модуль может принять сообщение на входе, передающее управление с данными, обработать его, и передать управляющее сообщение с данными следующему модулю.

Межмодульные взаимодействия в кластере на основе СКУ-модели сети частичных автоматов по принципу «каждый с каждым» рассмотрены в п. 3.5.

Автоматное программирование в настоящее время предлагается считать одной из технологий, в существенной степени сокращающей сроки составления программ и упрощающей их тестирование [115]. Системы канонических уравнений также позволяют создавать на их основе программы прикладного, промежуточного и системного уровней.

Как известно, к нарушению корректности граф-схем алгоритмов приводят следующие и другие конфигурации: дедлок, неоднозначность, зависание. Поэтому задачу проверки граф-схемы на корректность предлагается решать на абстрактной модели взаимодействий алгоритмов – на сетях Петри [113, 114, 141]; методики перехода от параллельной ГСА к сети Петри приведены, например, в работах [133, 134].

Далее необходимо построить сетевую автоматную СКУ-модель обмена данными между параллельными частями кластерного приложения в режиме «каждый с каждым». Это позволить имитировать в автоматных моделях связь параллельных частей одного и того же приложения по данным и обеспечить дальнейшую ее реализацию в вычислительных кластерах.

3.5. Сетевая автоматная СКУ-модель обмена данными между параллельными частями кластерного приложения в режиме «каждый с каждым»

Сетевая модель на рис. 3.5 связывает 6 подавтоматов $\boldsymbol{S_0}$, $\boldsymbol{S_1}$, $\boldsymbol{S_2}$, $\boldsymbol{S_3}$, $\boldsymbol{S_4}$ и $\boldsymbol{S_5}$, каждый из которых определен на множествах следующих состояний: $\boldsymbol{S_0}$ – A_1, A_2; $\boldsymbol{S_1}$ – A_3, A_4, A_5, A_6, A_{10}; $\boldsymbol{S_2}$ – A_3, A_4, A_5, A_6, A_{10}; $\boldsymbol{S_3}$ – A_3, A_4, A_5, A_6, A_{10}; $\boldsymbol{S_4}$ – A_3, A_4, A_5, A_6, A_{10}; $\boldsymbol{S_5}$ – A_3, A_4.

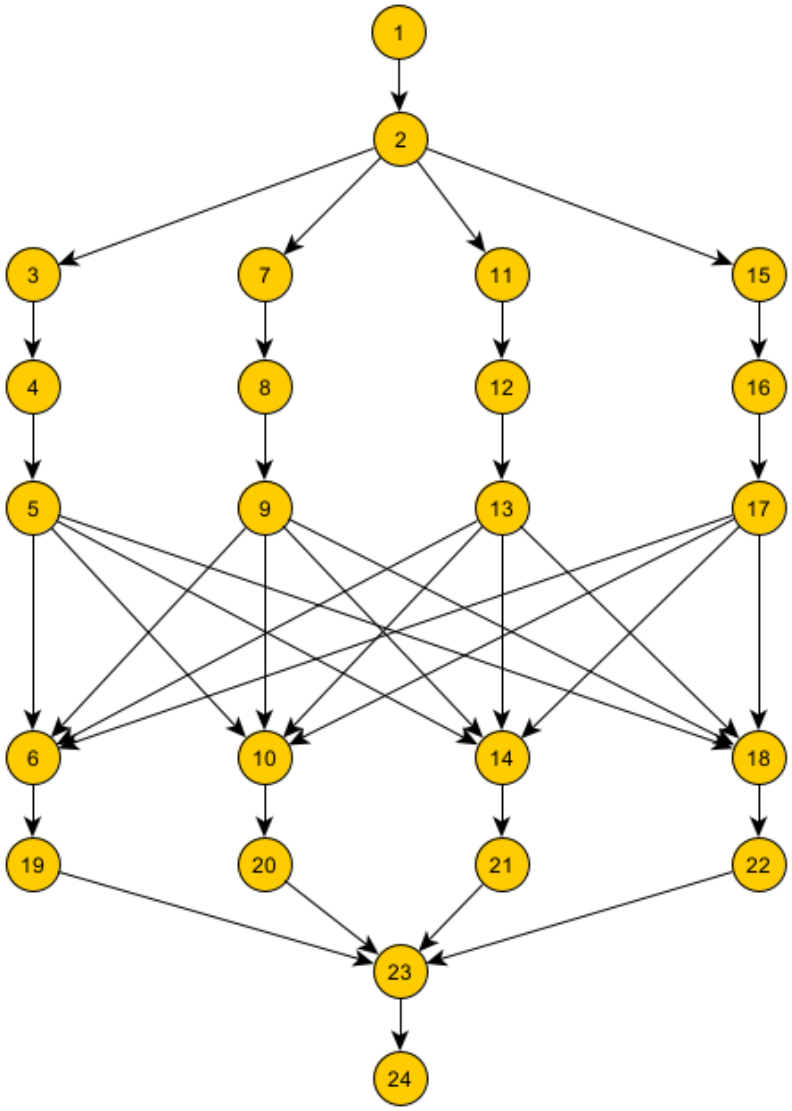


Рис. 3.5. Сетевая автоматная модель обмена данными между параллельными частями приложения в режиме «каждый с каждым»

Подавтоматам соответствуют части одного приложения, связанные по данным, причем каждая часть после загрузки размещена на своем узле вычислительного кластера. Для каждого подавтомата составлена система рекуррентных предикатных канонических уравнений, пригодная как для реализации исполнимой имитационной модели, так и для кластерного приложения.

Системы канонических уравнений для автоматной сетевой модели обмена данными между параллельными частями приложения

СКУ0:

$A_1(t+1) = A_1(t\ ) \ \&\ \neg x_0(t) \vee x_{begin}(t);$

$A_2(t+1) = A_1(t) \ \&\ x_0(t) \vee A_2(t) \ \&\ \neg z_2(t);$

СКУ1:

$A_3(t+1) = A_2(t) \ \&\ z_2(t) \vee A_3(t) \ \&\ \neg z_3(t);$

$A_4(t+1) = A_3(t) \ \&\ z_3(t) \vee A_4(t) \ \&\ \neg z_4(t);$

$A_5(t+1) = A_4(t) \ \&\ z_4(t) \vee A_5(t) \ \&\ \neg A_6(t);$

$A_6(t+1) = A_5(t) \ \&\ z_5(t) \ \&\ A_9(t) \ \&\ z_9(t) \ \&$

$A_{13}(t) \ \&\ z_{13}(t) \ \&\ A_{17}(t) \ \&\ z_{17}(t) \vee A_6(t) \ \&\ \neg z_6(t);$

$A_{19}(t+1) = A_6(t) \ \&\ z_6(t) \vee A_{19}(t) \ \&\ \neg A_{23}(t);$

СКУ2:

$A_7(t+1) = A_2(t) \ \&\ z_2(t) \vee A_7(t) \ \&\ \neg z_7(t);$

$A_8(t+1) = A_7(t) \ \&\ z_7(t) \vee A_8(t) \ \&\ \neg z_8(t);$

$A_9(t+1) = A_8(t) \ \&\ z_8(t) \vee A_9(t) \ \&\ \neg A_{10}(t);$

$A_{10}(t+1) = A_9(t) \ \&\ z_9(t) \ \&\ A_5(t) \ \&\ z_5(t) \ \&$

$A_{13}(t) \ \&\ z_{13}(t) \ \&\ A_{17}(t) \ \&\ z_{17}(t) \vee A_{10}(t) \ \&\ \neg z_{10}(t);$

$A_{20}(t+1) = A_{10}(t) \ \&\ z_{10}(t) \vee A_{20}(t) \ \&\ \neg A_{23}(t);$

СКУ3:

$A_{11}(t+1) = A_2(t) \ \&\ z_2(t) \vee A_{11}(t) \ \&\ \neg z_{11}(t);$

$A_{12}(t+1) = A_{11}(t) \ \&\ z_{11}(t) \vee A_{12}(t) \ \&\ \neg z_{12}(t);$

$A_{13}(t+1) = A_{12}(t) \ \&\ z_{12}(t) \vee A_{13}(t) \ \&\ \neg A_{14}(t);$

$A_{14}(t+1) = A_{13}(t) \ \&\ z_{13}(t) \ \&\ A_5(t) \ \&\ z_5(t) \ \&$

$A_9(t) \ \&\ z_9(t) \ \&\ A_{17}(t) \ \&\ z_{17}(t) \vee A_{14}(t) \ \&\ \neg z_{14}(t);$

$A_{21}(t+1) = A_{14}(t) \ \&\ z_{14}(t) \vee A_{21}(t) \ \&\ \neg A_{23}(t);$

СКУ4:

$A_{15}(t+1) = A_2(t) \,\&\, z_2(t) \vee A_{15}(t) \,\&\, \neg z_{15}(t);$

$A_{16}(t+1) = A_{15}(t) \,\&\, z_{15}(t) \vee A_{16}(t) \,\&\, \neg z_{16}(t);$

$A_{17}(t+1) = A_{16}(t) \,\&\, z_{16}(t) \vee A_{17}(t) \,\&\, \neg A_{18}(t);$

$A_{18}(t+1) = A_{17}(t) \,\&\, z_{17}(t) \,\&\, A_5(t) \,\&\, z_5(t) \,\&$

$A_9(t) \,\&\, z_9(t) \,\&\, A_{13}(t) \,\&\, z_{13}(t) \vee A_{18}(t) \,\&\, \neg z_{18}(t);$

$A_{22}(t+1) = A_{18}(t) \,\&\, z_{18}(t) \vee A_{22}(t) \,\&\, \neg A_{23}(t);$

СКУ5:

$A_{23}(t+1) = A_{19}(t) \,\&\, z_{19}(t) \,\&\, A_{20}(t) \,\&\, z_{20}(t) \,\&$

$A_{21}(t) \,\&\, z_{21}(t) \,\&\, A_{22}(t) \,\&\, z_{22}(t) \vee A_{23}(t) \,\&\, \neg z_{23}(t);$

$A_{24}(t+1) = A_{23}(t) \,\&\, z_{23}(t) \vee A_{24}(t) \,\&\, \neg z_{24}(t).$

Каждое каноническое уравнение задает вывод в исчислении предикатов первого порядка, приводящий либо к сохранению текущего состояния, либо к переходу к новому состоянию автомата. Данная модель дополняет автоматную модель последовательно-параллельной композиции (сети) автоматов, рассмотренную в п. 3.3, и позволяет связывать параллельные части приложений по данным. Для того, чтобы не усложнять автоматную модель, в рассмотренном примере использованы простые «линейные» структуры автоматов. Однако модель может быть усложнена и расширена за счет использования структурированных состояний, содержащих, в свою очередь, свои подавтоматы.

3.6. Формализация логико-вероятностных моделей сетей частичных автоматов, создаваемых на основе языка систем канонических уравнений (СКУ)

Вводится понятие логико-вероятностной модели «темпоральная вероятностная СКУ», которая позволит получить наглядную формализацию и реализацию автоматных моделей и рабочих программ, характерных для кластерных и других приложений, например, с конвейерным параллелизмом, и в существенной степени сократить число «инкрементных» сложений при перечислении моментов дискретного времени:

$$\text{ТВСКУ} = (СКУ_0, S, X, T_X, W_{TX}, T_Z, W_{TZ}),$$

где сетевая $СКУ_0$ – исходная, принятая в качестве начального языка описания сети конечных автоматов, система канонических уравнений, ограниченная описанием частичных автоматов и характерным для кластерных и конвейерных вычислительных систем в основном последовательным выполнением событий во времени; дополнительно допускается только простой параллелизм событий без взаимодействия копий ветвей $СКУ_0$, завершающихся барьерной синхронизацией ветвей; S – конечное множество событий $\{S_0(t), S_1(t), \ldots, S_n(t)\}$ представленных унарными предикатами; X – конечное множество входных событий, заданных унарными предикатами $\{X_0(t), X_1(t), \ldots, X_m(t)\}$; T_X – конечное множество случайных временных интервалов $\{t_{x0}, t_{x1}, \ldots, t_{xm}\}$ от текущих моментов до моментов наступления входных событий X; W_{TX} – конечное множество функций распределения вероятностей вида $P\{t_{xk} = i\} = p_{ki}$, $i = 0, 1, \ldots, i_k$, случайных временных интервалов из множества T_X; T_Z – конечное множество случайных временных интервалов $\{t_{z0}, t_{z1}, \ldots, t_{zn}\}$ сохранения, то есть выполнения событий из множества S; W_{TZ} – конечное множество функций распределения вероятностей вида $P\{t_{zr} = j\} = p_{rj}$, $j = 0, 1, \ldots, j_r$, случайных временных интервалов из множества T_Z.

Случайные величины (в программных реализациях – псевдослучайные величины) t_x и t_z принимают только целые неотрицательные значения. Рассматриваются только конечные распределения вероятностей целочисленных случайных величин. Возможно также использование целочисленных констант в качестве значений величин t_{xk}, $k = 0, 1, \ldots, m$, и t_{zr}, $r = 0, 1, \ldots, n$.

Структура приложения, содержащего последовательные и независимые параллельные секции, образующие сеть автоматов, может допускать более глубокий уровень вложенности, что характерно для большинства кластерных вычислительных систем.

Исходная неинтерпретируемость модели ТВСКУ позволяет применять ее на уровне программ, модулей программ, операторов, вплоть до уровня машинных команд и микропрограмм.

Общий подход к разработке статистической исполнимой модели кластерного приложения состоит в следующем. Используется метод организации последовательности событий, при реализации которого чередуются периоды зарождения событий с периодами сохранения событий. Пусть, например, для последовательной секции приложения, формализованной, например, СКУ $\boldsymbol{SP}_{Seq}$, выполняются действия, заданные следующими тремя выбранными уравнениями:

$$A_2(t+1) = A_1(t) \,\&\, z_1(t) \vee A_2(t) \& \neg z_2(t);$$
$$A_3(t+1) = A_2(t) \,\&\, z_2(t) \,\&\, \neg x_1(t) \vee A_3(t) \,\&\, \neg z_3(t);$$
$$A_4(t+1) = A_2(t) \,\&\, z_2(t) \,\&\, x_1(t) \vee A_4(t) \,\&\, \neg z_4(t);$$

Как было определено при построении какого-либо уравнения, зарождение очередного события, например, события $A_2(t+1)$, происходит в момент времени $t+1$. Для того, чтобы это событие состоялось в указанный момент времени, необходимо, чтобы в предыдущий момент времени t были истинны высказывания $A_1(t)$ и $z_1(t)$ – то есть событие $A_1(t)$ выполнилось бы последний раз в этот момент, на что указывало бы появление истинного значения высказывания $z_1(t)$ – окончания действия события $A_1(t)$. С момента времени $t+1$ начинается действие события $A_2(t+1)$, которое сохраняется до тех пор, пока будет ложно высказывание $z_2(t)$. Появление истинного значения высказывания $z_2(t)$ приведет к тому факту, что составное высказывание $A_2(t) \& \neg z_2(t)$ станет ложным и условие сохранения события $A_2(t)$ выполняться не будет. В следующий момент времени событие $A_2(t)$ состоится последний раз, и при истинности высказывания $A_2(t) \,\&\, z_2(t) \,\&\, \neg x_1(t)$ в следующий момент времени $t+1$ начнет выполняться (зародится) событие $A_3(t+1)$. «Продлить» сохранение события

$A_2(t)$ возможно, перейдя к отметке времени $t = t + t_{z2}$ его окончания, где значение временного интервала t_{z2} определяется при помощи датчика псевдослучайных чисел с заданным законом распределения. Действуя аналогично, можно отсрочить зарождение события $A_3(t+1)$, отсрочив действие входной переменной $x_1(t)$ на величину интервала времени t_{x1} путем перехода к временной отметке события $t = t + t_{x1}$, соответствующего активизации переменной $x_1(t)$. Значение переменной t_{x1} задается датчиком псевдослучайных чисел. Составное высказывание $A_2(t)\ \&\ z_2(t)\ \&\ \neg x_1(t)$ станет истинным, и далее произойдет зарождение события $A_3(t+1)$ в следующий момент времени $t+1$. Сохранение события A_3, зарождение и сохранение события A_4 происходит аналогично. Так, путем перехода от события с событию, реализуется логико-вероятностная модель кластерного приложения.

На рис. 3.4 была представлена сетевая автоматная модель выполнения приложения в кластере в режиме последовательно-параллельной работы *SPMD*. При принятом режиме работы независимых параллельных программ в параллельной секции режима *SPMD* возможно при разработке общей сетевой автоматной модели использовать независимые автоматные СКУ-модели вида $\boldsymbol{MD_{Clon}}$ для представления структурированных событий *C1 – C8*.

3.7. Результаты статистических экспериментов с моделями кластерных систем в режиме SPMD «Одиночный поток программ – множественный поток данных»

На основании автоматных СКУ-моделей и логико-алгебраических ЛАОВ-моделей построены имитационные статистические модели выполнения приложений. В табл. 3.1 представлены численные значения коэффициента ускорения выполнения выполняемого приложения в режиме *SPMD* на вычислительном кластере, полученные на построенной имитационной модели. За ускорение, следуя [1], взято отношение $k = (t_{\text{Посл.}} + N \cdot t_N)/(t_{\text{Посл.}} + t_N)$ времени выполнения приложения на одном узле $(t_{\text{Посл.}} + N \cdot t_N)$ к сумме $(t_{\text{Посл.}} + t_N)$ времени

выполнения того же приложения в параллельном режиме на всех узлах кластера t_N и времени $t_{\text{Посл.}}$ однократного выполнения последовательного участка приложения. Предполагалось, что исполнение параллельных участков происходит независимо друг от друга. Поскольку времена выполнения последовательного и распараллеливаемого участка заранее неизвестны, их характеристики определяются путем проведения статистического эксперимента. Поэтому в приведенной формулах $t_{\text{Посл.}}$ и t_N – это оценки математических ожиданий этих интервалов времени. При моделировании полагалось, что время работа каждого оператора распределено равномерно, от 1 до 9 мс. Переходы по условиям равновероятны (по 0,5).

Результаты сведены в табл. 3.1. Записи в заголовках столбцов означают, что коэффициент ускорения k вычислен при указанном значении времени $T = t_{\text{Посл.}}$, выраженном в единицах модельного времени (здесь – в миллисекундах, мс).

Табл. 3.1

N	T=0	T=10	T=20	T=30	T=40	T=50
1	1	1	1	1	1	1
2	2	1,98	1,96	1,94	1,92	1,9
4	4	3,88	3,77	3,67	3,57	3,48
8	8	7,46	7,0	6,6	6,25	5,93
16	16	13,8	12,25	11,0	10,0	9,25
32	32	24,2	19,6	16,5	14,3	12,6

Примеры зависимостей коэффициента ускорения от числа узлов в кластере иллюстрирует рис. 3.6. При $T = 0$ последовательного участка нет, поэтому значения коэффициента ускорения равны числу задействованных узлов. С ростом значения T эффект от распараллеливания менее выражен, так как на результат вычислении коэффициента ускорения время выполнения последовательного участка оказывает большее влияние.

Сложность решения задачи вызвана разветвленностью выбранного алгоритма, а также наличием циклов. Время выполнения разветвленных участков программы и число проходимых циклов зависит от вида вводимых усло-

вий и на практике может быть определено при помощи детальной имитационной модели. Как автоматные СКУ-модели, так и модели на основе логико-алгебраических выражений являются моделями исполнимого типа, потому что для исследования свойств и динамического поведения моделируемого объекта нужно модель «выполнить», то есть запустить ее на компьютере и исследовать процессы смены событий.

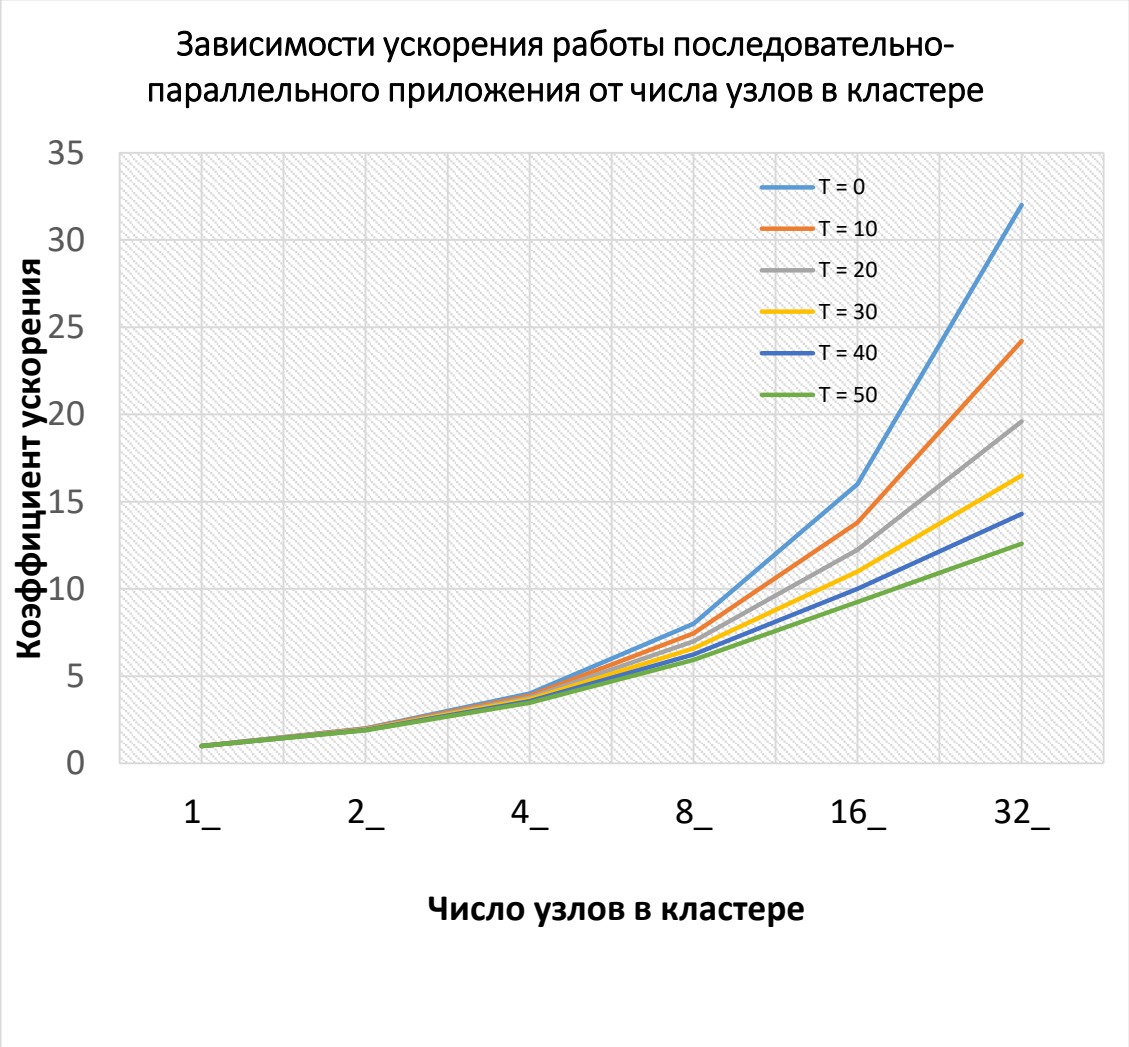


Рис. 3.6. Результаты статистических экспериментов с моделями кластера, выполняющего последовательно-параллельное приложение ***L_{Seq}*ParReplicate(i = 1..8)C_i***.

Вычисление коэффициента ускорения по известной [1] формуле Амдала $k = (t_{\text{Посл.}} + N \cdot t_N)/(t_{\text{Посл.}} + t_N)$ проводилось на основе результатов статистиче-

ского моделирования. Например, на рис. 3.7 приведена гистограмма распределения времени $t_{\text{Посл.}} + N \cdot t_N$ выполнения приложения без распараллеливания при $N = 1$.

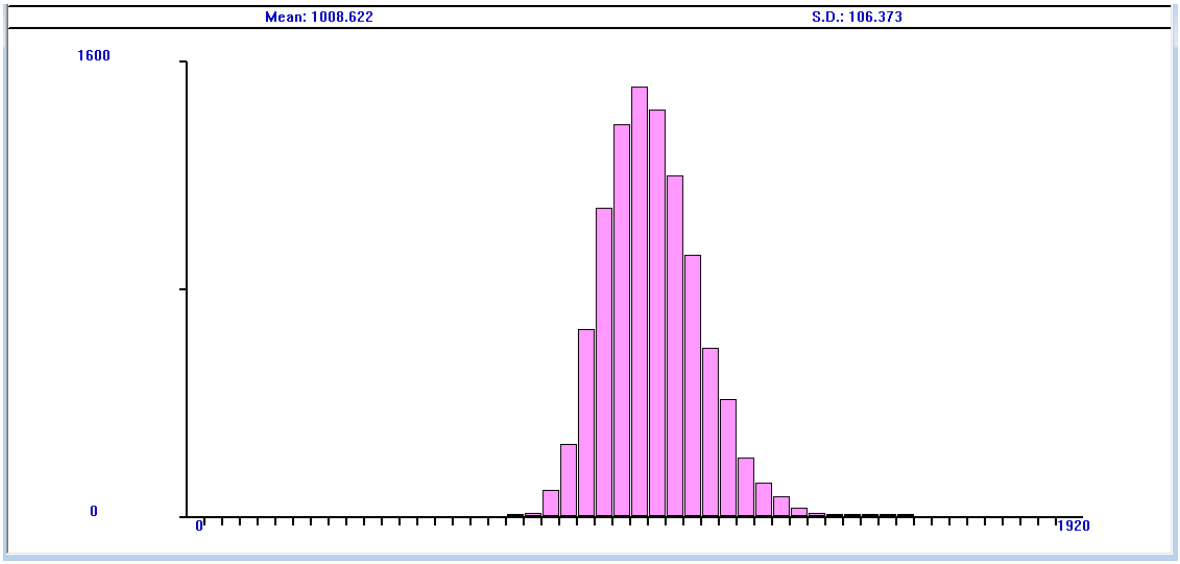


Рис. 7. Гистограмма распределения времени выполнения $t_{\text{вып}} = t_{\text{посл}} + 32t_{32}$ приложения при $N = 1$ без распараллеливания; по оси абсцисс указаны границы частотных классов для гистограммы, шаг гистограммы – 40 мс; по оси ординат указано число попаданий в каждый частотный класс при объеме выборки – 10000

Здесь $Mean1 = t_{\text{Посл.}} + 32 \cdot t_{32} = 1008{,}622$ мс – оценка математического ожидания времени выполнения приложения без распараллеливания.

На следующем рис. 3.8 приведена гистограмма распределения времени выполнения приложения при $N = 32$ с последовательным участком и распараллеливанием. Здесь $Mean2 = t_{\text{Посл.}} + t_{32} = 80{,}266$ мс – оценка математического ожидания времени выполнения приложения с последовательным участком и распараллеливанием остальной части. Оценка времени выполнения последовательного участка определялась в этом же эксперименте и равна $t_{\text{Посл.}} = 49{,}991$ мс. Тогда $k = (t_{\text{Посл.}} + 32 \cdot t_{32})/(t_{\text{Посл.}} + t_{32}) = 12{,}58$. Такой же результат дает вычисление ускорения по известной формуле для второго закона Амдала [1]:

$$k' = N / [\beta \cdot N + (1 - \beta)]$$

при $N = 32$, с долей последовательных вычислений:

$$\beta = t_{\text{Посл.}}/(t_{\text{Посл.}} + 32 \cdot t_{32}) = 49{,}991/1008{,}622 = 0{,}04957,\ k' = 12{,}61.$$

Небольшая погрешность объясняется использованием метода статистического моделирования при оценивании временных параметров в модели вычислительного кластера.

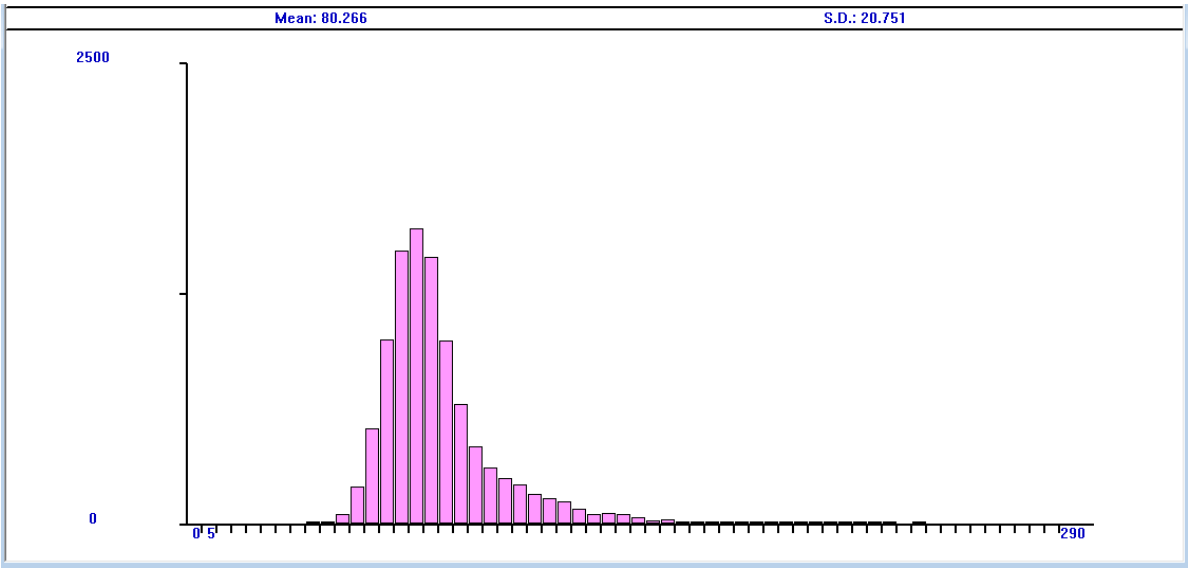


Рис. 3.8. Гистограмма распределения времени выполнения $t_{\text{вып}} = t_{\text{посл}} + t_{32}$ приложения при $N = 32$, с последовательным участком и распараллеливанием остальной части; по оси абсцисс указаны границы частотных классов для гистограммы, шаг гистограммы – 5 мс; по оси ординат указано число попаданий в каждый частотный класс при объеме выборки – 10000

3.8. Переход от автоматных моделей к асинхронным логико-алгебраическим моделям функционирования кластерных вычислительных систем

В основе логико-алгебраических моделей лежат логические исчисления и алгебраические системы. К логическим исчислениям относятся исчисление высказываний и исчисление предикатов. Аппарат логико-алгебраических операционных выражений (ЛАОВ), основанный на интеграции ряда формализаций дискретных процессов, описан с позиций различных применений и обоснован в работах [117, 118] при учете работ [116, 142, 143]. На заключительных этапах тестирования моделей при исследовании их динамики рекомендуется непосредственно использовать сети Петри [113, 114, 141, 145], заменяя или

исключая при этом «нововведения», касающиеся дополнительного использования логики предикатов первого порядка и операций с отношениями. Для сохранения преемственности в названиях модулей кластерного приложения для названий позиций выбраны имена операторных вершин в $ГСА_1$ (рис. 3.1) и $ГСА_2$ (рис. 3.2), а для индексации переходов автоматной сети Петри выбраны сдвоенные индексы.

На рис. 3.9 представлены примеры, иллюстрирующие переходы от ГСА к СКУ-модели частичного автомата и далее к начальным логико-алгебраическим выражениям для сети Петри. Для фрагментов на рис. 3.9, *а* и 3.9, *б* составлены канонические уравнения, а для фрагмента на рис. 3.9, *в* – логико-алгебраические выражения.

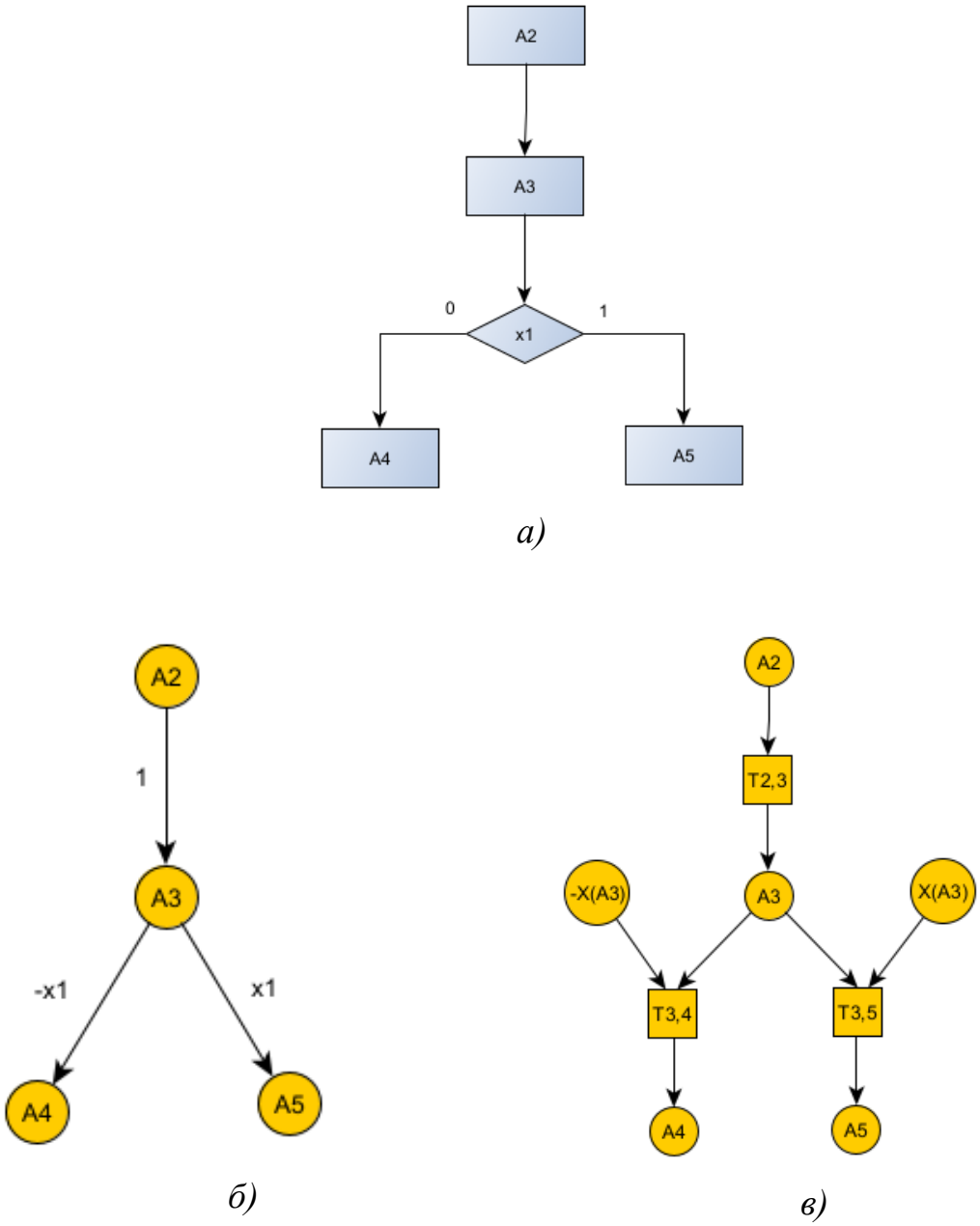
Система канонических уравнений для примеров на рис. 3.9, *а* и рис. 3.9, *б* имеет следующий вид:

$$A_3(t+1) = A_2(t) \,\&\, z_2(t) \vee A_3(t) \,\&\, \neg z_3(t);$$
$$A_4(t+1) = A_3(t) \,\&\, z_3(t) \,\&\, \neg x_1(t) \vee A_4(t) \,\&\, \neg z_4(t);$$
$$A_5(t+1) = A_3(t) \,\&\, z_3(t) \,\&\, x_1(t) \vee A_4(t) \,\&\, \neg z_4(t).$$

Система асинхронных логико-алгебраических операционных выражений для примера на рис. 3.9, *в* представлена в следующем виде:

$$T_{2,3}: [M(A_2)\&\neg M(A_3)](\{M(A_2) \leftarrow \mathbf{false}, M(A_3) \leftarrow \mathbf{true}\} \vee Ret);$$
$$T_{3,4}: [M(A_3) \,\&\, \neg M(A_4) \,\&\, \neg X(A_3)](\{X(A_3) \leftarrow \mathbf{undef}, M(A_3) \leftarrow \mathbf{false},$$
$$M(A_4) \leftarrow \mathbf{true}\} \vee Ret);$$
$$T_{3,5}: [M(A_3) \,\&\, \neg M(A_5) \,\&\, X(A_3)](\{X(A_3) \leftarrow \mathbf{undef}, M(A_3) \leftarrow \mathbf{false},$$
$$M(A_5) \leftarrow \mathbf{true}\} \vee Ret),$$

где ***undef*** **–** промежуточное неопределенное значение логического условия.



а) *б)* *в)*

Рис. 3.9. Фрагменты ГСА (*а*), графа переходов частичного автомата (*б*) и сети Петри (*в*)

Приведенные выражения ЛАОВ в данном случае интерпретируются как правила срабатывания переходов логической сети Петри. Здесь M – унарный предикат, или функция разметки позиций, одноименных операторам исходной ГСА; $M(A_i)$ – высказывание, истинность которого соответствует наличию одной метки в позиции A_i, а ложность – отсутствию метки; X – унарный предикат, определяющий условия в исходной ГСА; $X(A_i)$ – высказывание, принимающее истинное, ложное или неопределенное значения, определяемые по результату выполнения оператора A_i.

Оператор *Ret* усиливает процедурную составляющую ЛАОВ и осуществляет переход к его повторному выполнению при ложности условия, заключенного в квадратные скобки.

Логико-алгебраическая операционная модель, очевидно, может быть построена при использовании графа переходов состояний (рис. 3.9, *б*), послужившего основой для построения сети Петри (рис. 3.9, *в*), для чего необходимо соблюдать правило формирования условий операторами, в том числе операторами ввода условий.

Правила срабатывания переходов могут далее модифицироваться в соответствии с требованиями предметной области. Дополнительные события – передачи сообщений, приема сообщений, квитирования передач, определения длительности событий, представляемые операциями модификации бинарных или тернарных предикатов могут не соответствовать общепринятым понятиям сетей Петри. Поэтому правила переходов могут приобрести вид более общих логико-алгебраических операционных выражений, для чего привлекается другой аппарат – например, аппарат систем алгоритмических алгебр [116, 142], сетей абстрактных машин [143], реляционных исчислений и алгебр [144].

3.9. Графы переходов частичных конечных автоматов и системы логико-алгебраических операционных выражений

На рис. 3.10 представлена логическая (бинарная) сеть Петри, использованная при составлении систем логико-алгебраических уравнений $\boldsymbol{L_{Seq}}\boldsymbol{*}\boldsymbol{L_{Clon}}$, описывающих функционирование вычислительного кластера.

Используемая в настоящей работе расширенная нотация ЛАОВ была предложена в работах [117, 118]. Метод перехода от графов переходов состояний к автоматным сетям Петри известен и был ранее описан в ряде работ [133, 134], и его иллюстрирует рис. 3.9, *в*. Метод состоит в простановке переходов на дугах и расстановке меток на вершинах графа переходов, рассматриваемых как позиции.

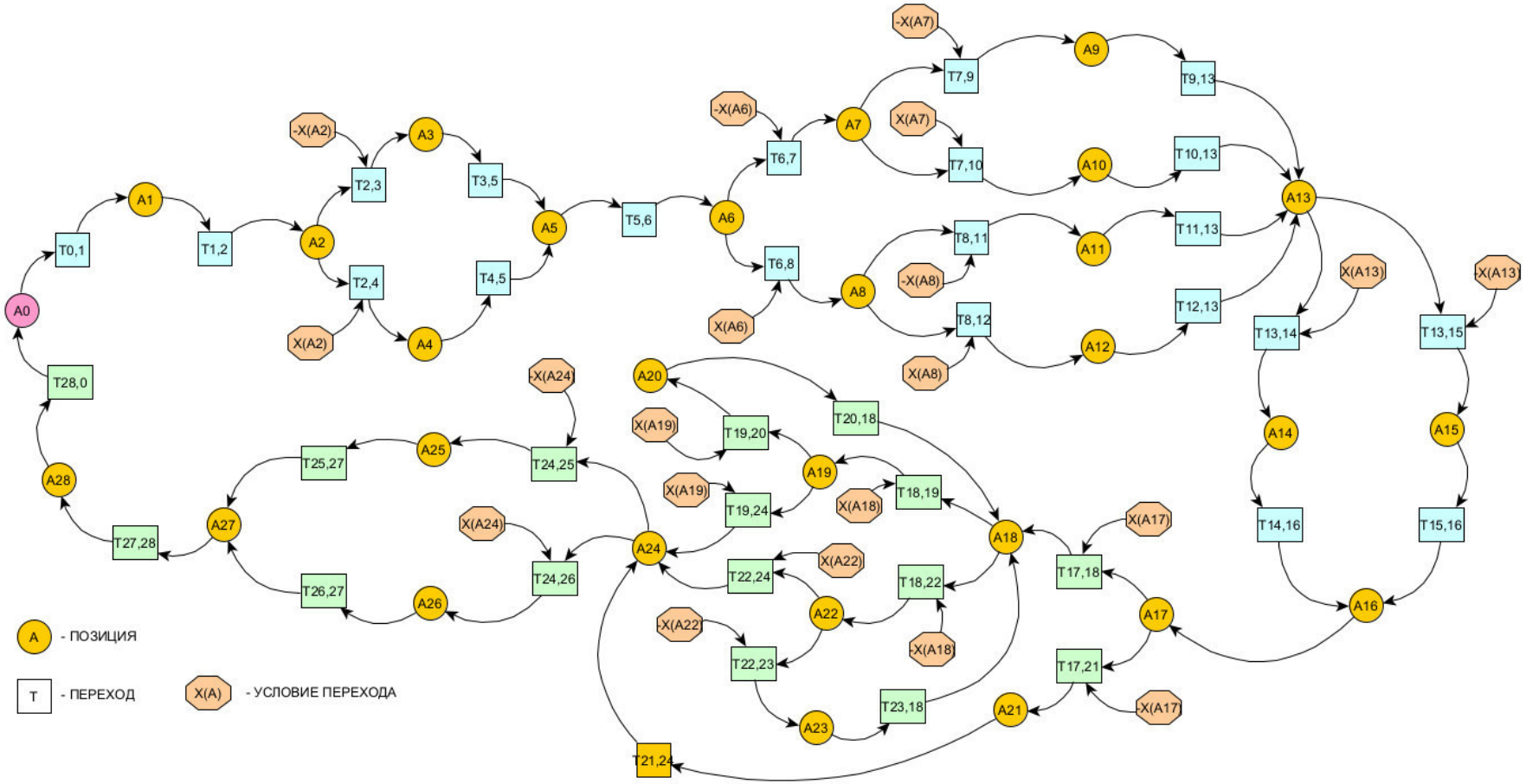


Рис. 3.10. Логическая (бинарная) сеть Петри, использованная при составлении систем логико-алгебраических уравнений $L_{Seq}*L_{Clon}$, и ***NETWORK-MPMD***, описывающих функционирование вычислительного кластера

Ввиду простоты этого метода, на начальных этапах достаточно при переходе от автоматных 1-ограниченных логических сетей Петри к построению ЛАОВ продукционного вида использовать только граф переходов частичного автомата, представленный на рис. 3.3 и граф переходов сети частичных автоматов на рис. 3.4. На заключительных этапах тестирования моделей при исследовании их динамики рекомендуется использовать непосредственно логические сети Петри.

Определим входные условия в графах переходов состояний частичных автоматов Мура: значение каждого высказывания $X(A_i)$ определяется результатом выполнения программного модуля A_i.

При построении ЛАОВ формально не требуется предварительное построение логической сети Петри, поскольку сами ЛАОВ являются мощным средством представления сложных асинхронных дискретных систем, однако графика сетей Петри, равно как и графов переходов состояний частичных автоматов, облегчают построение выражений ЛАОВ. В качестве примеров составления трех начальных выражений ЛАОВ системы $\boldsymbol{L_{Seq}}$ выбраны состояния 1, 2, 3 и 4 (рис. 3.3).

Первое выражение описывает переход от состояния 1 к состоянию 2:

$$T_{1,2}: [M(A_1)\&\neg M(A_2)](\{M(A_1) \leftarrow \boldsymbol{false}, M(A_2) \leftarrow \boldsymbol{true}\} \vee Ret),$$

где M: $A \rightarrow \{\boldsymbol{true, false}\}$ – унарный предикат, определяющий в терминах логических сетей Петри маркировку некоторой позиции из множества A; « $\leftarrow$ » – символ операции модификации предиката; в квадратные скобки заключено выражение для составного высказывания; в фигурных скобках заключена последовательность действий по модификации предикатов. Таким образом выражением $T_{1,2}$ описывается переход метки из позиции A_1 в позицию A_2 логической сети Петри. Программной переинтерпретации данного выражения соответствуют следующие действия: «по завершении выполнения программного

модуля A_1 управление и результаты передаются модулю A_2». Оператор *Ret* выполняет переход к началу системы ЛАОВ при ложности составного высказывания в квадратных скобках.

Выражениям $T_{2,3}$ и $T_{2,4}$ соответствуют переходы по условию, то есть по значению высказывания $X(A_2)$:

$T_{2,3}$: $[M(A_2) \& \neg M(A_3) \& \neg X(A_2)](\{X(A_2) \leftarrow \textbf{undef}, M(A_2) \leftarrow \textbf{false}, M(A_3) \leftarrow \textbf{true}\} \vee Ret);$

$T_{2,4}$: $[M(A_2) \& \neg M(A_4) \& X(A_2)](\{X(A_2) \leftarrow \textbf{undef}, M(A_2) \leftarrow \textbf{false}, M(A_4) \leftarrow \textbf{true}\} \vee Ret),$

где X: $A \rightarrow \{\textbf{true}, \textbf{false}, \textbf{undef}\}$ – частично-определенный унарный предикат, значение которого $X(A_i)$ при фиксированной предметной переменной A определяется результатом выполнения соответствующего программного модуля A_i; после проведенной проверки и переходе значение условия $X(A_i)$ становится неопределенным до очередного вычисления или ввода извне.

Полная система $\boldsymbol{L_{Seq}}$ логико-алгебраических операционных выражений для последовательного участка программы составлена в соответствии с верхней частью рис. 3.3 и имеет следующий вид:

Система $\boldsymbol{L_{Seq}}$:

$T_{1,2}$: $[M(A_1)\&\neg M(A_2)](\{M(A_1) \leftarrow \textbf{false}, M(A_2) \leftarrow \textbf{true}\} \vee Ret);$

$T_{2,3}$: $[M(A_2) \& \neg M(A_3) \& \neg X(A_2)](\{X(A_2) \leftarrow \textbf{undef}, M(A_2) \leftarrow \textbf{false}, M(A_3) \leftarrow \textbf{true}\} \vee Ret);$

$T_{2,4}$: $[M(A_2) \& \neg M(A_4) \& X(A_2)](\{X(A_2) \leftarrow \textbf{undef}, M(A_2) \leftarrow \textbf{false}, M(A_4) \leftarrow \textbf{true}\} \vee Ret);$

$T_{3,5}$: $[M(A_3)\&\neg M(A_5)](\{M(A_3) \leftarrow \textbf{false}, M(A_5) \leftarrow \textbf{true}\} \vee Ret);$

$T_{4,5}$: $[M(A_4)\&\neg M(A_5)](\{M(A_4) \leftarrow \textbf{false}, M(A_5) \leftarrow \textbf{true}\} \vee Ret);$

$T_{5,6}$: $[M(A_5)\&\neg M(A_6)](\{M(A_5) \leftarrow \textbf{false}, M(A_6) \leftarrow \textbf{true}\} \vee Ret);$

$T_{6,7}$: $[M(A_6) \& \neg M(A_7) \& \neg X(A_6)](\{X(A_6) \leftarrow \textbf{undef}, M(A_6) \leftarrow \textbf{false},$

$M(A_7) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{6,8}: [M(A_6) \& \neg M(A_8) \& X(A_6)](\{X(A_6) \leftarrow \mathbf{undef}, M(A_6) \leftarrow \mathbf{false},$

$M(A_8) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{7,9}: [M(A_7) \& \neg M(A_9) \& \neg X(A_7)](\{X(A_7) \leftarrow \mathbf{undef}, M(A_7) \leftarrow \mathbf{false},$

$M(A_9) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{7,10}: [M(A_7) \& \neg M(A_{10}) \& X(A_7)](\{X(A_7) \leftarrow \mathbf{undef}, M(A_7) \leftarrow \mathbf{false},$

$M(A_{10}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{8,11}: [M(A_8) \& \neg M(A_{11}) \& \neg X(A_8)](\{X(A_8) \leftarrow \mathbf{undef}, M(A_8) \leftarrow \mathbf{false},$

$M(A_{11}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{8,12}: [M(A_8) \& \neg M(A_{12}) \& X(A_8)](\{X(A_8) \leftarrow \mathbf{undef}, M(A_8) \leftarrow \mathbf{false},$

$M(A_{12}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{9,13}: [M(A_9) \& \neg M(A_{13})](\{M(A_9) \leftarrow \mathbf{false}, M(A_{13}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{10,13}: [M(A_{10}) \& \neg M(A_{13})](\{M(A_{10}) \leftarrow \mathbf{false}, M(A_{13}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{11,13}: [M(A_{11}) \& \neg M(A_{13})](\{M(A_{11}) \leftarrow \mathbf{false}, M(A_{13}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{12,13}: [M(A_{12}) \& \neg M(A_{13})](\{M(A_{12}) \leftarrow \mathbf{false}, M(A_{13}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{13,14}: [M(A_{13}) \& \neg M(A_{14}) \& X(A_{13})](\{X(A_{13}) \leftarrow \mathbf{undef}, M(A_{13}) \leftarrow \mathbf{false},$

$M(A_{14}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{13,15}: [M(A_{13}) \& \neg M(A_{15}) \& \neg X(A_{13})](\{X(A_{13}) \leftarrow \mathbf{undef}, M(A_{13}) \leftarrow \mathbf{false},$

$M(A_{15}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{14,16}: [M(A_{14}) \& \neg M(A_{16})](\{M(A_{14}) \leftarrow \mathbf{false}, M(A_{16}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{15,16}: [M(A_{15}) \& \neg M(A_{16})](\{M(A_{15}) \leftarrow \mathbf{false}, M(A_{16}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{16,17}: [M(A_{16}) \& \neg M(A_{17})](\{M(A_{16}) \leftarrow \mathbf{false}, M(A_{17}) \leftarrow \mathbf{true}\} \vee Ret).$

3.10. Система логико-алгебраических операционных выражений для распараллеленных участков кластерного приложения

Система ЛАОВ $\boldsymbol{L_{Clon}}$ для одной реплики C_0 для спецификации копий параллельных программ имеет следующий вид (выражения составлены в соответствии с графом, размещенном в затемненной области C_0 на рис. 3.3):

Система L_{Clon}:

$T_{17,18}$: $[M(A_{17}) \& \neg M(A_{18}) \& X(A_{17})](\{X(A_{17}) \leftarrow \mathbf{undef}, M(A_{17}) \leftarrow \mathbf{false}, M(A_{18}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{17,21}$: $[M(A_{17}) \& \neg M(A_{21}) \& \neg X(A_{17})](\{X(A_{17}) \leftarrow \mathbf{undef}, M(A_{21}) \leftarrow \mathbf{false}, M(A_{21}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{18,19}$: $[M(A_{18}) \& \neg M(A_{19}) \& X(A_{18})](\{X(A_{18}) \leftarrow \mathbf{undef}, M(A_{18}) \leftarrow \mathbf{false}, M(A_{19}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{18,22}$: $[M(A_{18}) \& \neg M(A_{22}) \& \neg X(A_{17})](\{X(A_{18}) \leftarrow \mathbf{undef}, M(A_{18}) \leftarrow \mathbf{false}, M(A_{22}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{19,20}$: $[M(A_{19}) \& \neg M(A_{20}) \& X(A_{19})](\{X(A_{19}) \leftarrow \mathbf{undef}, M(A_{19}) \leftarrow \mathbf{false}, M(A_{20}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{19,24}$: $[M(A_{19}) \& \neg M(A_{24}) \& \neg X(A_{19})](\{X(A_{19}) \leftarrow \mathbf{undef}, M(A_{19}) \leftarrow \mathbf{false}, M(A_{24}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{20,18}$: $[M(A_{20}) \& \neg M(A_{18})](\{M(A_{20}) \leftarrow \mathbf{false}, M(A_{18}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{21,24}$: $[M(A_{21}) \& \neg M(A_{24})](\{M(A_{21}) \leftarrow \mathbf{false}, M(A_{24}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{22,23}$: $[M(A_{22}) \& \neg M(A_{23}) \& \neg X(A_{22})](\{X(A_{22}) \leftarrow \mathbf{undef}, M(A_{22}) \leftarrow \mathbf{false}, M(A_{23}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{22,24}$: $[M(A_{22}) \& \neg M(A_{24}) \& X(A_{22})](\{X(A_{22}) \leftarrow \mathbf{undef}, M(A_{22}) \leftarrow \mathbf{false}, M(A_{24}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{23,18}$: $[M(A_{23}) \& \neg M(A_{18})](\{M(A_{23}) \leftarrow \mathbf{false}, M(A_{18}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{24,25}$: $[M(A_{24}) \& \neg M(A_{25}) \& \neg X(A_{24})](\{X(A_{24}) \leftarrow \mathbf{undef}, M(A_{24}) \leftarrow \mathbf{false}, M(A_{25}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{24,26}$: $[M(A_{22}) \& \neg M(A_{26}) \& X(A_{24})](\{X(A_{24}) \leftarrow \mathbf{undef}, M(A_{24}) \leftarrow \mathbf{false}, M(A_{26}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{25,27}$: $[M(A_{25}) \& \neg M(A_{27})](\{M(A_{25}) \leftarrow \mathbf{false}, M(A_{27}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{26,27}$: $[M(A_{26}) \& \neg M(A_{27})](\{M(A_{26}) \leftarrow \mathbf{false}, M(A_{27}) \leftarrow \mathbf{true}\} \vee Ret)$;

$T_{27,28}$: $[M(A_{27}) \& \neg M(A_{28})](\{M(A_{27}) \leftarrow \mathbf{false}, M(A_{28}) \leftarrow \mathbf{true}\} \vee Ret)$.

Для получения логико-алгебраических операционных выражений для программы вида *SPMD* с параллельными участками (рис. 3.4) выражение для перехода

$$T_{16,17}: [M(A_{16}) \,\&\, \neg M(A_{17})](\{M(A_{16}) \leftarrow \mathbf{false}, M(A_{17}) \leftarrow \mathbf{true}\} \vee Ret)$$

в системе $\boldsymbol{L_{Seq}}$ необходимо заменить на восемь следующих выражений:

$T_{16,C1}: [M(A_{16}) \,\&\, \neg M(A_{C1})](\{M(A_{C1}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{16,C2}: [M(A_{16}) \,\&\, \neg M(A_{C2})](\{M(A_{C2}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{16,C3}: [M(A_{16}) \,\&\, \neg M(A_{C3})](\{M(A_{C3}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{16,C4}: [M(A_{16}) \,\&\, \neg M(A_{C4})](\{M(A_{C4}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{16,C5}: [M(A_{16}) \,\&\, \neg M(A_{C5})](\{M(A_{C5}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{16,C6}: [M(A_{16}) \,\&\, \neg M(A_{C6})](\{M(A_{C6}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{16,C7}: [M(A_{16}) \,\&\, \neg M(A_{C7})](\{M(A_{C7}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{16,C8}: [M(A_{16}) \,\&\, \neg M(A_{C8})](\{M(A_{C8}) \leftarrow \mathbf{true}\} \vee Ret).$

Следующая система ЛАОВ описывает реплицируемый участок программы:

$\boldsymbol{ParReplicate(i = 1..8)L_{Clon}}:$

$\{\{T_{17,18}: [M(A_{17}) \,\&\, \neg M(A_{18}) \,\&\, X(A_{17})](\{X(A_{17}) \leftarrow \mathbf{undef}, M(A_{17}) \leftarrow \mathbf{false},$
$M(A_{18}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{17,21}: [M(A_{17}) \,\&\, \neg M(A_{21}) \,\&\, \neg X(A_{17})](\{X(A_{17}) \leftarrow \mathbf{undef}, M(A_{21}) \leftarrow \mathbf{false},$
$M(A_{21}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{18,19}: [M(A_{18}) \,\&\, \neg M(A_{19}) \,\&\, X(A_{18})](\{X(A_{18}) \leftarrow \mathbf{undef}, M(A_{18}) \leftarrow \mathbf{false},$
$M(A_{19}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{18,22}: [M(A_{18}) \,\&\, \neg M(A_{22}) \,\&\, \neg X(A_{17})](\{X(A_{18}) \leftarrow \mathbf{undef}, M(A_{18}) \leftarrow \mathbf{false},$
$M(A_{22}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{19,20}: [M(A_{19}) \,\&\, \neg M(A_{20}) \,\&\, X(A_{19})](\{X(A_{19}) \leftarrow \mathbf{undef}, M(A_{19}) \leftarrow \mathbf{false},$

$M(A_{20}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{19,24}: [M(A_{19}) \& \neg M(A_{24}) \& \neg X(A_{19})](\{X(A_{19}) \leftarrow \mathbf{undef}, M(A_{19}) \leftarrow \mathbf{false},$
$M(A_{24}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{20,18}: [M(A_{20}) \& \neg M(A_{18})](\{M(A_{20}) \leftarrow \mathbf{false}, M(A_{18}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{21,24}: [M(A_{21}) \& \neg M(A_{24})](\{M(A_{21}) \leftarrow \mathbf{false}, M(A_{24}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{22,23}: [M(A_{22}) \& \neg M(A_{23}) \& \neg X(A_{22})](\{X(A_{22}) \leftarrow \mathbf{undef}, M(A_{22}) \leftarrow \mathbf{false},$
$M(A_{23}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{22,24}: [M(A_{22}) \& \neg M(A_{24}) \& X(A_{22})](\{X(A_{22}) \leftarrow \mathbf{undef}, M(A_{22}) \leftarrow \mathbf{false},$
$M(A_{24}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{23,18}: [M(A_{23}) \& \neg M(A_{18})](\{M(A_{23}) \leftarrow \mathbf{false}, M(A_{18}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{24,25}: [M(A_{24}) \& \neg M(A_{25}) \& \neg X(A_{24})](\{X(A_{24}) \leftarrow \mathbf{undef}, M(A_{24}) \leftarrow \mathbf{false},$
$M(A_{25}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{24,26}: [M(A_{22}) \& \neg M(A_{26}) \& X(A_{24})](\{X(A_{24}) \leftarrow \mathbf{undef}, M(A_{24}) \leftarrow \mathbf{false},$
$M(A_{26}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{25,27}: [M(A_{25}) \& \neg M(A_{27})](\{M(A_{25}) \leftarrow \mathbf{false}, M(A_{27}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{26,27}: [M(A_{26}) \& \neg M(A_{27})](\{M(A_{26}) \leftarrow \mathbf{false}, M(A_{27}) \leftarrow \mathbf{true}\} \vee Ret);$

$T_{27,28}: [M(A_{27}) \& \neg M(A_{28})](\{M(A_{27}) \leftarrow \mathbf{false}, M(A_{28}) \leftarrow \mathbf{true}\} \vee Ret)\}\}.$

Затем необходимо завершить описание восьми реплик следующими выражениями:

$T_{28}: [M(A_{C1}) \& M(A_{C2}) \& M(A_{C3}) \& M(A_{C4}) \& M(A_{C5}) \& M(A_{C6}) \&$
$M(A_{C7}) \& M(A_{C8}) \& \neg M(A_{28})](\{M(A_{C1}) \leftarrow \mathbf{false}, M(A_{C2}) \leftarrow \mathbf{false},$
$M(A_{C3}) \leftarrow \mathbf{false}, M(A_{C4}) \leftarrow \mathbf{false}, M(A_{C5}) \leftarrow \mathbf{false}, M(A_{C6}) \leftarrow \mathbf{false},$
$M(A_{C7}) \leftarrow \mathbf{false}, M(A_{C8}) \leftarrow \mathbf{false}, M(A_{28}) \leftarrow \mathbf{true}\} \vee Ret);$
$T_{28,29}: [M(A_{28}) \& \neg M(A_{29})](\{M(A_{28}) \leftarrow \mathbf{false}, M(A_{29}) \leftarrow \mathbf{true}\} \vee Ret).$

Выражением T_{28} описывается процесс сборки результатов выполнения всех восьми реплик $\boldsymbol{ParReplicate(i = 1..8)L_{Clon}}$, выполняемых параллельно. Выражением $T_{28,29}$ текст модели завершается.

Полная модель с одним последовательным участком и восемью параллельно выполняемыми независимо друг от друга участками описывается выражением $\boldsymbol{L_{Seq}*ParReplicate(i = 1..8)L_{Clon}}$.

3.11. Организация работы и моделирование вычислительного кластера в режиме MPMD «Множество программ, множество потоков данных» и параллельно-конвейерном режиме NETWORK-MPMD

Режим MPMD [1, 46] в принятой в настоящей работе интерпретации соответствует сложному случаю, когда на разных узлах кластера в локальной сети выполняются разные программы, реализующие модули одного и того же распределенного алгоритма. В процессе реализации сетевого алгоритма модули передают друг другу управление и, при необходимости, промежуточные результаты. При описании исполнимой имитационной модели далее будет использовано (там, где это не вызовет противоречий) отождествление объектов модели и объектов моделируемой системы, что упростит описание и сделает его более компактным.

Логико-алгебраическая операционная модель, очевидно, может быть построена при использовании графа переходов состояний, изображенного на рис. 3.11, для чего необходимо соблюдать правило формирования условий операторами, в том числе операторами ввода условий.

Введен следующий бинарный предикат развертывания распределенного приложения:

$$D: Modules \times Nodes \rightarrow \{\boldsymbol{true}, \boldsymbol{false}\},$$

где $Modules = \{A_1, A_2, \ldots, A_{28}\}$ – множество модулей распределенного приложения; $Nodes = \{N_0, N_1, \ldots, N_{12}\}$ – множество узлов кластера, где N_0 – управляющий узел кластера, на котором размещены управляющие модули A_1, A_2, …,

A_{16}. Связь с распределенным приложением, размещенным на узлах N_1, N_2, …, N_{12}, осуществляется модулем A_{16} с узла N_0. Одноименное предикату D отношение (область его истинности) носит функциональный характер, так как каждый модуль распределенного приложения может быть размещен только на одном узле кластера.

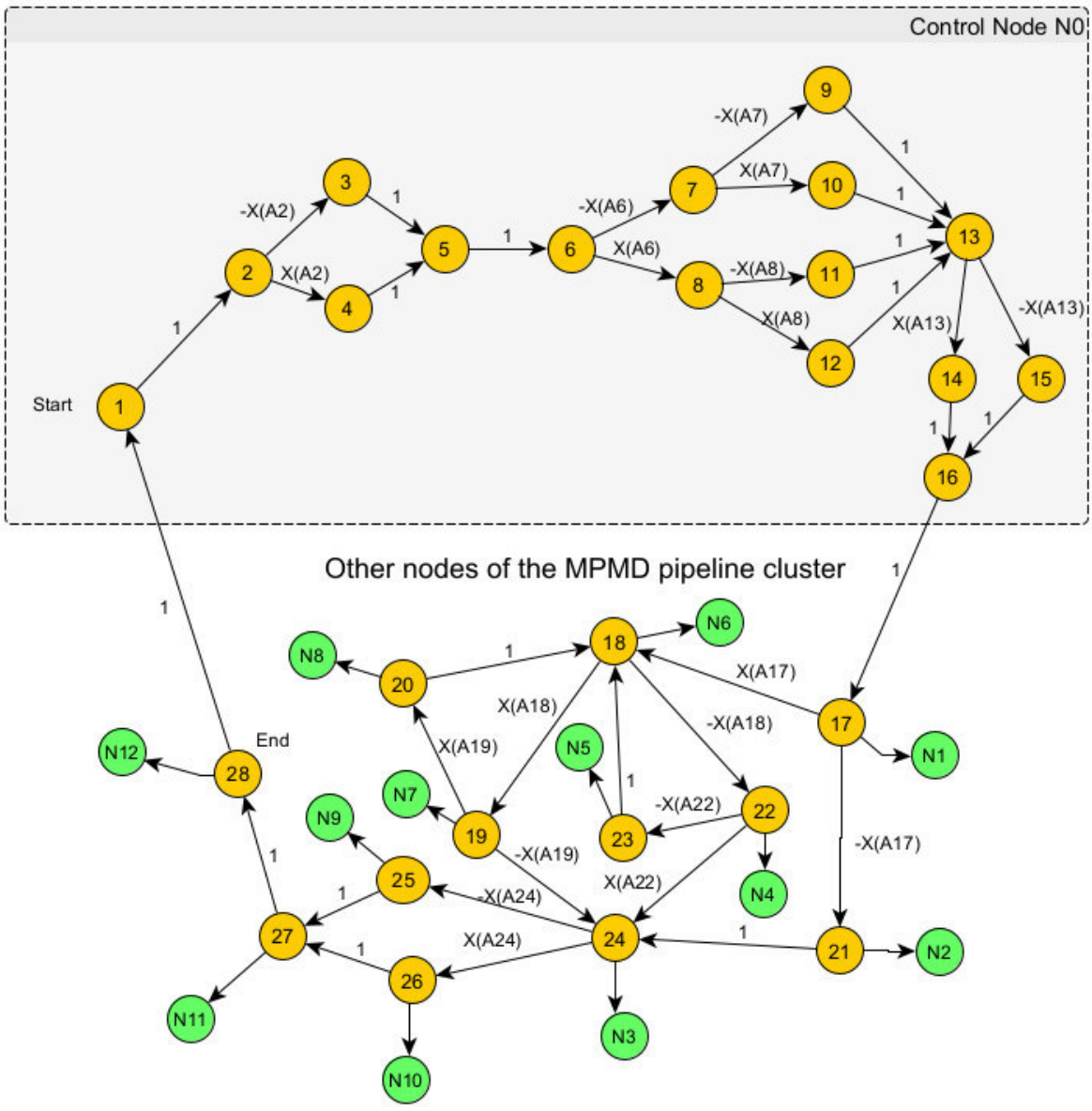


Рис. 3.11. Граф переходов состояний вычислительного кластера, соответствующий работе распределенного многомодульного приложения в параллельно-конвейерном режиме NETWORK-MPMD на нижней части рисунка; верхняя часть в рамке соответствует последовательному подготовительному этапу, выполняемому управляющим узлом (английские слова на рисунке переводятся как «другие узлы конвейерного MPMD-кластера»)

Поскольку управляющие модули A_1, A_2, …, A_{16} размещены на одном и том же узле, модели их функционирования соответствует составленная ранее система логико-алгебраических операционных выражений $\boldsymbol{L_{Seq}}$, в которой выражение $T_{16,17}$ необходимо заменить следующим выражением $Q_{16,17}$ для переходного модуля A_{16}, реализующего инициирование распределенного MPMD-приложения:

$$Q_{16,17}: [M(A_{16})]([S(N_0, N_1)^* \,\&\, R(N_1, N_0)^* \,\&\, \neg M(A_{17})](\{M(A_{16}) \leftarrow \boldsymbol{false},$$

$$S(N_0, N_1) \leftarrow \boldsymbol{true}^*, R(N_1, N_0) \leftarrow \boldsymbol{true}^*, M(A_{17}) \leftarrow \boldsymbol{true}\} \vee Ret) \vee Ret).$$

Данное выражение составлено в соответствии с алгеброй алгоритмов и содержит суперпозицию двух операций α-дизъюнкций. Проверяемые условия заключены в квадратные скобки.

Операционная часть ЛАОВ заключена в фигурные скобки, а расстановка круглых скобок определяет порядок выполнения операций проверки α-условий и модификации унарных предикатов. Операции, заключенные в квадратные и в фигурные скобки, выполняются последовательно. Модули одного и того же приложения могут выполняться параллельно только в том случае, когда проверки условий и модификации предикатов не зависят друг от друга.

При выполнении модуля $Q_{16,17}$ сначала проверяется высказывание $M(A_{16})$, истинность которого означает, что по завершении выполнения модуля A_{16} должен начать выполнение следующий модуль A_{17}. Здесь

$$M: Modules \rightarrow \{\boldsymbol{true}, \boldsymbol{false}\}$$

– унарный предикат, отмечающий выполнение одного из модулей множества *Modules*; истинность высказывания $M(A_i)$ означает, что модуль $A_i \in Modules$ выполняет свою часть распределенного приложения. Инициатором смены состояний данных модулей должен быть переходной модуль $Q_{16,17}$. В модели с высоким уровнем абстракции можно было бы ограничиться следующим выражением для переходного модуля:

$$Q'_{16,17}: [M(A_{16})](\{M(A_{16}) \leftarrow \boldsymbol{false}, M(A_{17}) \leftarrow \boldsymbol{true}\} \vee E),$$

где E – пустое событие, которое не влияет на смену состояния. Затем переход (в модели – переходный модуль $Q'_{16,17}$) задерживает метку на время ее передачи из входной позиции в выходную. В данном случае следует указать на некоторую аналогию модели ЛАОВ другому известному формализму временных автоматных сетей Петри, отличающемуся особенностью интерпретации. Сеть Петри – это модель, отображающая статику и динамику асинхронных взаимодействий процессов [133, 134, 147]. В связи с тем, что для первоначального описания распределенного приложения была предложена автоматная модель (рис. 3.3), дальнейший интерес представляют автоматные *SM*-сети Петри (англ. *State Machine*) [133]. Каждый переход таких сетей имеет не более одного входа и не более одного выхода, а сами сети представляют собой модели процессов с переходами по условиям. Если в сети такого вида перемещается только одна метка, то сеть является графом переходов состояний автомата. Временные сети Петри (англ. *Timed Petri Nets*) позволяют моделировать дискретно-событийные системы с учетом задержек и временных ограничений [141, 148].

Для построения логико-вероятностной модели функционирования вычислительного кластера по построенной логико-алгебраической операционной модели возможно использовать понятие временных сетей Петри в следующей интерпретации: метка, попавшая в очередную позицию, задерживается на некоторое время, заданное вероятностным распределением или фрагментом реализованной программы распределенного приложения. Затем переход (в модели – переходный модуль) задерживает метку на время ее передачи из входной позиции в выходную. При реализации модели в качестве элемента функциональной архитектуры кластера временные задержки формируются естественным образом, например, в процессе передачи управляющих сообщений. Этим и обусловлено различие выражений $Q_{16,17}$ и $Q'_{16,17}$ для переходного модуля.

Выражение для переходного модуля $Q_{16,17}$ (размещенного на узле N_0 кластера) соответствует варианту логико-алгебраической модели, дополненной учетом важных свойств и управляющих команд локальной вычислительной сети, на которой реализован вычислительный кластер. В этом выражении, после описанной ранее проверки высказывания $M(A_{16})$ на истинность, проверяется истинность сложного составного высказывания

$$[S(N_0, N_1)^* \,\&\, R(N_1, N_0)^* \,\&\, \neg M(A_{17})],$$

состоящего из трех высказываний, где отмеченные звездочками высказывания используются в следующей новой интерпретации. Здесь S и R – бинарные предикаты, моделирующие передачу (S) и прием (R) служебных сообщений:

S: $Nodes \times Nodes \rightarrow \{\textbf{\textit{true}}, \textbf{\textit{false}}\}$;

R: $Nodes \times Nodes \rightarrow \{\textbf{\textit{true}}, \textbf{\textit{false}}\}$.

Истинность высказывания $S(N_0, N_1)^*$ соответствует передаче «запрашивающего» сообщения из узла N_0 в узел N_1, а истинность высказывания $R(N_1, N_0)^*$ соответствует передаче «ответного» сообщения из N_1 в узел N_0. Передается информация о готовности адресуемого модуля A_{17} к приему сообщения, то есть (в модели) это информация о том, истинно ли высказывание $\neg M(A_{17})$. Если оно истинно при ложном высказывании $M(A_{17})$, то это означает, что модуль A_{17} не задействован ($M(A_{17}) = \textbf{\textit{false}}$) и готов к принятию управления от модуля A_{16}.

При соблюдении всех описанных условий устанавливается истинность составного высказывания. Звездочки, которыми отмечены высказывания $S(N_0, N_1)^*$ и $R(N_1, N_0)^*$, означает, что по завершении передач сообщений истинность этих высказываний по умолчанию меняется на ложность, которая сохраняется до повторного инициирования переходного модуля $Q_{16,17}$.

После установления истинности высказываний $M(A_{16})$ и $\neg M(A_{17})$ выполняются действия, предписанные выражением в фигурных скобках:

$$\{M(A_{16}) \leftarrow \textbf{\textit{false}},\ S(N_0, N_1) \leftarrow \textbf{\textit{true}}^*,\ R(N_1, N_0) \leftarrow \textbf{\textit{true}}^*,\ M(A_{17}) \leftarrow \textbf{\textit{true}}\}.$$

Данное выражение включает операции модификации предикатов *M*, *S* и *R*. Операция $M(A_{16}) \leftarrow$ ***false*** означает, что модуль A_{16} прекращает свою работу в связи с передачей управления и данных модулю A_{17}. Операция $S(N_0, N_1) \leftarrow$ ***true**** означает, что узел N_0 должен передать управление узлу N_1 кластера. Операция $R(N_1, N_0) \leftarrow$ ***true**** означает, что узел N_1 должен ответить узлу N_0, сообщая ему о готовности принять управление. Звездочки, которыми отмечены обе операции, означают, что высказываниям сразу после передачи сообщений будут присвоены ложные значения. Наконец, выполнение операции $M(A_{17}) \leftarrow$ ***true*** соответствует приему управления и данных модулем A_{17}.

При невыполнении хотя бы одного из описанных в квадратных скобках условий действия в фигурных скобках не выполняются, и в этом случае выполняется один из альтернативных для α-дизъюнкций операторов *Ret*. При помощи этого оператора после истечения заданного по умолчанию тайм-аута управление возвращается к повторному выполнению переходного модуля $Q_{16,17}$. Если указанная ситуация повторяется заранее предопределенное число раз, то в кластере регистрируется ошибочная или аварийная ситуация.

Система ***NETWORK-MPMD*** логико-алгебраических операционных выражений для MPMD-приложения содержит формализованные сетевые спецификации и имеет следующий вид:

Система ЛАОВ NETWORK-MPMD:

$T_{17,18}/N_1$: $[M(A_{17}) \,\&\, X(A_{17})]\ ([S(N_1, N_6)^* \,\&\, R(N_6, N_1)^* \,\&\, \neg M(A_{18}/N_6)]\{X(A_{17}) \leftarrow$ ***undef***, $M(A_{17}) \leftarrow$ ***false***, $S(N_1, N_6) \leftarrow$ ***true****, $R(N_6, N_1) \leftarrow$ ***true****, $M(A_{18}) \leftarrow$ ***true***$\} \vee Ret) \vee Ret)$;

$T_{17,21}/N_1$: $[M(A_{17}) \,\&\, \neg X(A_{17})]([S(N_1, N_2)^* \,\&\, R(N_2, N_1)^* \,\&\, \neg M(A_{21})]$
$(\{X(A_{17}) \leftarrow$ ***undef***, $M(A_{21}) \leftarrow$ ***false***, $S(N_1, N_2) \leftarrow$ ***true****, $R(N_2, N_1) \leftarrow$ ***true****, $M(A_{21}) \leftarrow$ ***true***$\} \vee Ret) \vee Ret)$;

$T_{18,19}/N_6$: $[M(A_{18}) \,\&\, X(A_{18})]([S(N_6, N_7)^* \,\&\, R(N_7, N_6)^* \,\&\, \neg M(A_{19})]$

$(\{X(A_{18}) \leftarrow \textbf{undef}, M(A_{18}) \leftarrow \textbf{false}, S(N_6, N_7) \leftarrow \textbf{true}^*, R(N_7, N_6) \leftarrow \textbf{true}^*, M(A_{19}) \leftarrow \textbf{true}\} \vee Ret) \vee Ret);$

$T_{18,22}/N_6$: $[M(A_{18}) \& \neg X(A_{18})]([S(N_6, N_4)^* \& R(N_4, N_6)^* \& \neg (A_{22})]$
$(\{X(A_{18}) \leftarrow \textbf{undef}, M(A_{18}) \leftarrow \textbf{false}, S(N_6, N_4) \leftarrow \textbf{true}^*, R(N_4, N_6) \leftarrow \textbf{true}^*, M(A_{22}) \leftarrow \textbf{true}\} \vee Ret) \vee Ret);$

$T_{19,20}/N_7$: $[M(A_{19}) \& X(A_{19})]([S(N_7, N_8)^* \& R(N_8, N_7)^* \& \neg M(A_{20})]$
$(\{X(A_{19}) \leftarrow \textbf{undef}, M(A_{19}) \leftarrow \textbf{false}, S(N_7, N_8) \leftarrow \textbf{true}^*, R(N_8, N_7) \leftarrow \textbf{true}^*, M(A_{20}) \leftarrow \textbf{true}\} \vee Ret) \vee Ret);$

$T_{19,24}/N_7$: $[M(A_{19}) \& \neg X(A_{19})]([S(N_7, N_3)^* \& R(N_3, N_7)^* \& \neg (A_{24})]$
$(\{X(A_{19}) \leftarrow \textbf{undef}, M(A_{19}) \leftarrow \textbf{false}, S(N_7, N_3) \leftarrow \textbf{true}^*, R(N_7, N_3) \leftarrow \textbf{true}^*, M(A_{24}) \leftarrow \textbf{true}\} \vee Ret) \vee Ret);$
$T_{20,18}/N_8$: $[M(A_{20})]([S(N_8, N_6)^* \& R(N_6, N_8)^* \& \neg M(A_{18})](\{M(A_{20}) \leftarrow \textbf{false}, S(N_8, N_6) \leftarrow \textbf{true}^*, R(N_6, N_8) \leftarrow \textbf{true}^*, M(A_{18}) \leftarrow \textbf{true}\} \vee Ret) \vee Ret);$
$T_{21,24}/N_2$: $[M(A_{21})]([S(N_2, N_3)^* \& R(N_3, N_2)^* \& \neg M(A_{24})](\{M(A_{21}) \leftarrow \textbf{false}, S(N_2, N_3) \leftarrow \textbf{true}^*, R(N_3, N_2) \leftarrow \textbf{true}^*, M(A_{24}) \leftarrow \textbf{true}\} \vee Ret) \vee Ret);$

$T_{22,23}/N_4$: $[M(A_{22}) \& \neg X(A_{22})]([S(N_4, N_5)^* \& R(N_5, N_4)^* \& \neg (A_{23})]$
$(\{X(A_{22}) \leftarrow \textbf{undef}, M(A_{22}) \leftarrow \textbf{false}, S(N_4, N_5) \leftarrow \textbf{true}^*, R(N_5, N_4) \leftarrow \textbf{true}^*, M(A_{23}) \leftarrow \textbf{true}\} \vee Ret) \vee Ret);$

$T_{22,24}/N_4$: $[M(A_{22}) \& X(A_{22})]([S(N_4, N_3)^* \& R(N_3, N_4)^* \& \neg M(A_{24})]$
$(\{X(A_{22}) \leftarrow \textbf{undef}, M(A_{22}) \leftarrow \textbf{false}, S(N_4, N_3) \leftarrow \textbf{true}^*, R(N_3, N_4) \leftarrow \textbf{true}^*, M(A_{24}) \leftarrow \textbf{true}\} \vee Ret) \vee Ret);$

$T_{23,18}/N_5$: $[M(A_{23})]([S(N_5, N_6)^* \& R(N_6, N_5)^* \& \neg M(A_{18})](\{M(A_{23}) \leftarrow \textbf{false},$

$S(N_5, N_6) \leftarrow$ ***true****, $R(N_6, N_5) \leftarrow$ ***true****, $M(A_{18}) \leftarrow$ ***true***$\} \vee Ret) \vee Ret);$

$T_{24,25}/N_3$: $[M(A_{24}) \& \neg X(A_{24})]([S(N_3, N_9)^* \& R(N_9, N_3)^* \& \neg M(A_{25})]$
$(\{X(A_{24}) \leftarrow$ ***undef***, $M(A_{24}) \leftarrow$ ***false***, $S(N_3, N_9) \leftarrow$ ***true****, $R(N_9, N_3) \leftarrow$ ***true****,
$M(A_{25}) \leftarrow$ ***true***$\} \vee Ret) \vee Ret);$

$T_{24,26}/N_3$: $[M(A_{22}) \& X(A_{24})]([S(N_3, N_{10})^* \& R(N_{10}, N_3)^* \& \neg (A_{26})]$
$(\{X(A_{24}) \leftarrow$ ***undef***, $M(A_{24}) \leftarrow$ ***false***, $S(N_3, N_{10}) \leftarrow$ ***true****,
$R(N_{10}, N_3) \leftarrow$ ***true****, $M(A_{26}) \leftarrow$ ***true***$\} \vee Ret) \vee Ret);$

$T_{25,27}/N_9$: $[M(A_{25})]([S(N_9, N_{11})^* \& R(N_{11}, N_9)^* \& \neg M(A_{27})]$
$(\{M(A_{25}) \leftarrow$ ***false***, $S(N_9, N_{11}) \leftarrow$ ***true****, $R(N_{11}, N_9) \leftarrow$ ***true****, $M(A_{27}) \leftarrow$ ***true***$\}$
$\vee Ret) \vee Ret);$

$T_{26,27}/N_{10}$: $[M(A_{26})]([S(N_{10}, N_{11})^* \& R(N_{11}, N_{10})^* \& \neg M(A_{27})]$
$(\{M(A_{26}) \leftarrow$ ***false***, $S(N_{10}, N_{11}) \leftarrow$ ***true****, $R(N_{11}, N_{10}) \leftarrow$ ***true****,
$M(A_{27}) \leftarrow$ ***true***$\} \vee Ret) \vee Ret);$

$T_{27,28}/N_{11}$: $[M(A_{27})]([S(N_{11}, N_{12})^* \& R(N_{12}, N_{11})^* \& \neg M(A_{28})]$
$(\{M(A_{27}) \leftarrow$ ***false***, $S(N_{11}, N_{12}) \leftarrow$ ***true****, $R(N_{12}, N_{11}) \leftarrow$ ***true****,
$M(A_{28}) \leftarrow$ ***true***$\} \vee Ret) \vee Ret);$

$T_{28,1}/N_{12}$: $[M(A_{27})]([S(N_{12}, N_0)^* \& R(N_0, N_{12})^* \& \neg M(A_{28})]$
$(\{M(A_{27}) \leftarrow$ ***false***, $S(N_{12}, N_0) \leftarrow$ ***true****, $R(N_0, N_{12}) \leftarrow$ ***true****,
$M(A_{28}) \leftarrow$ ***true***$\} \vee Ret) \vee Ret).$

Система логико-алгебраических операционных выражений ***NETWORK-MPMD*** положена в основу реально работающего приложения для вычислительного кластера, а также для исполнимой статистической модели.

3.12. Анализ сети Петри, описывающей параллельно-конвейерный NETWORK-MPMD-режим работы вычислительного кластера

Автоматная сеть Петри, описывающая параллельно-конвейерный ***NETWORK-MPMD***-режим работы вычислительного кластера при $n = 6$ процессах, представлена рис. 3.12.

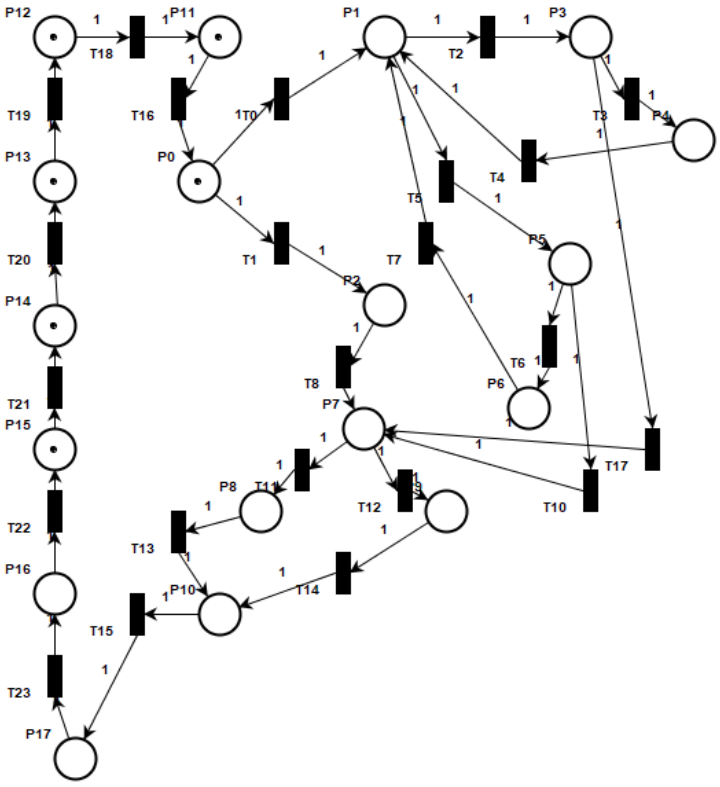


Рис. 3.12. Автоматная сеть Петри, описывающая параллельно-конвейерный *NETWORK-MPMD* режим работы вычислительного кластера при $n = 6$ процессах

Она построена на основании графа переходов состояний частичного автомата, представленного на нижней части рис. 3.11. При построении сети

Петри использовано открытое кроссплатформенное программное обеспечение PIPE2, широко известное в международной практике и предназначенное для работы с сетями Петри; оно доступно по гиперссылке URL: http://pipe2.sourceforge.net/.

Количественные характеристики для графов достижимых состояний (ГДС) сети Петри на рис. 3.12 сведены в табл. 3.2.

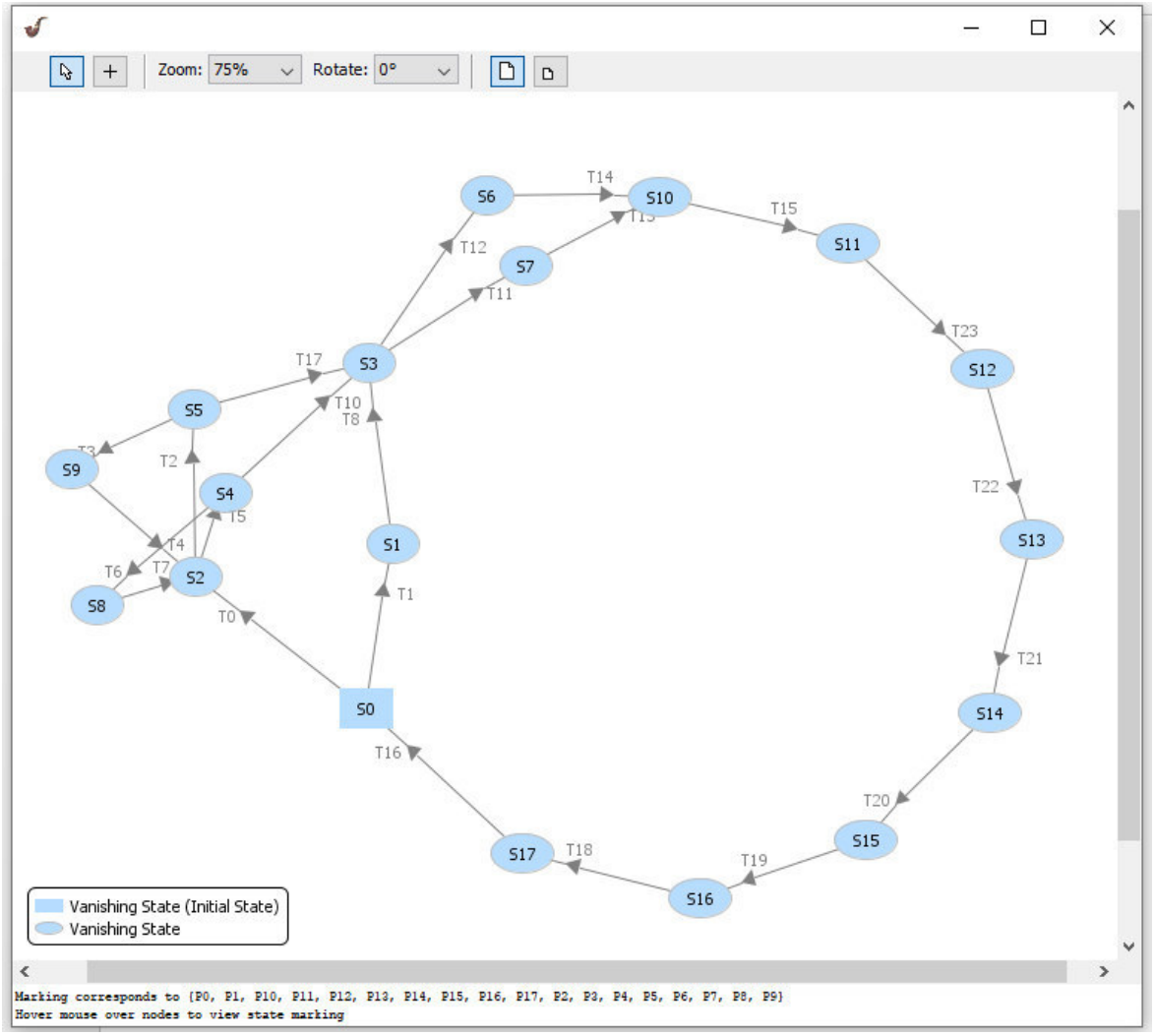


Рис. 3.13. Граф достижимых состояний для одного процесса

Табл. 3.2

Количество процессов	Количество состояний	Количество дуг
1	18	23
2	153	368
3	816	2760
4	3060	12880
5	8568	41860
6	18564	100464
7	31824	184184

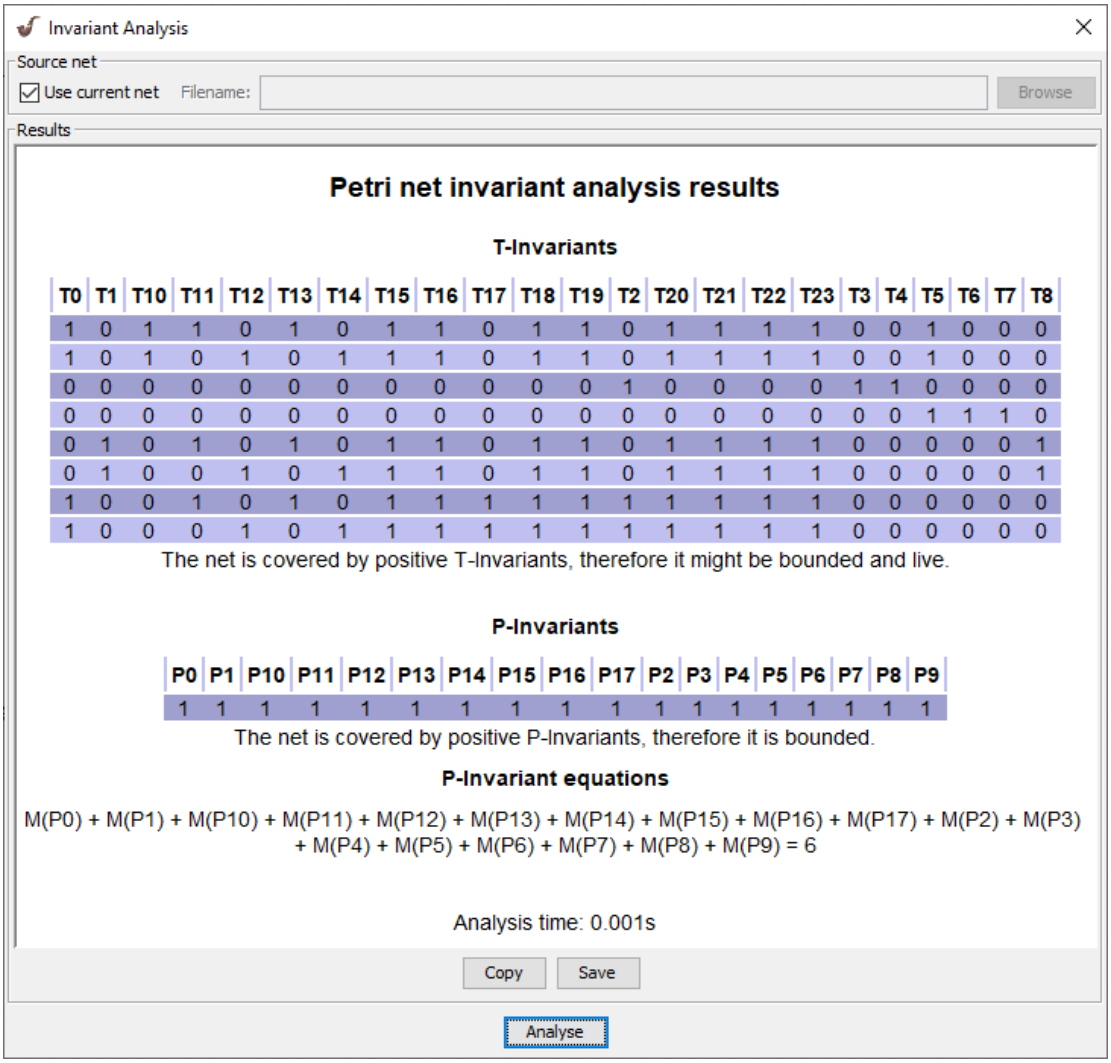


Рис. 3.14. Результат инвариантного анализа сети Петри для 6 процессов

По табл. 3.2 видно, что число состояний и связывающих их дуг в ГДС резко возрастает с ростом числа меток, представляющих выполняемые в кластере процессы. Для визуального анализа фактически доступен только ГДС на

рис. 3.13, полученный при помощи программы PIPE2 для одного процесса. Очевидно, что этот граф повторяет автоматную MPMD-модель (положительный факт контроля правильности построения сети Петри).

В основе исследования большинства свойств сетей Петри лежит анализ графов достижимости. Инвариантный анализ сети Петри на рис. 3.14 и автоматическое исследование ГДС показывают, что рассматриваемая сеть Петри, описывающая работу вычислительного кластера в параллельно-конвейерном режиме, является живой, 1-ограниченной (безопасной) и свободна от блокировок.

3.13. Результаты статистических экспериментов с моделью параллельно-конвейерного NETWORK-MPMD-режима загрузки кластера распределенными приложениями

MPMD-режим работы кластера относится к числу самых сложных для реализации в кластере вычислительных машин. Система ***NETWORK-MPMD*** составлена на основе рис. 3.13, на котором выполняется сложное распределенное многомодульное приложение. Производительность кластера можно повысить, если загружать новые сетевые программы и выполнять их в конвейерном MPMD-режиме ***NETWORK-MPMD***. Реализованная на основе ЛАОВ логико-вероятностная модель ***NETWORK-MPMD*** позволяет моделировать исполнение сразу нескольких приложений, запускаемых последовательно с интервалом *T*. Работа каждого из запускаемых последовательно приложений соответствует графу переходов состояний на нижней части рис. 3.13 и системе выражений ***NETWORK-MPMD***. Каждому оператору в модели требуется время выполнения, распределенное равномерно от 2 мс до 10 мс. Сообщения содержат около 1 Мбит информации. При работе для каждого оператора задан свой узел. Такая организация работы кластера необходима в том случае, когда данные находятся на различных узлах, и на узлы загружаются различные модули, связанные по управлению и данным.

Результаты статистических экспериментов с моделью параллельно-конвейерного MPMD-режима ***NETWORK-MPMD*** сведены в таблицы 3.3–3.9. Программная «прототипная» модель построена на языке общецелевой системы дискретно-событийного моделирования GPSS WORLD фирмы Minuteman Software (сайт *minutemansoftware.com*). После того, как прототипная модель будет удовлетворять исследователя, разрабатывается реальное кластерное приложение на основании автоматной или логико-алгебраической моделей, рассматриваемых как формализованные спецификации для проекта.

Главной целью экспериментов являлось определение границ пропускной способности кластера при типовой нагрузке. Отличительной особенностью модели является возможность учета произвольного алгоритма работы моделируемого приложения.

Каждая таблица содержит четыре колонки, для которых сохранены компьютерные рабочие названия: колонка *FACILITY* содержит имена узлов кластера – объектов моделирования, имитирующих обработку данных в конвейерном режиме; колонка *ENTRIES* содержит данные о числе входов в каждый узел при моделировании, эти данные позволяют определить правильность работы алгоритма путем проверки узлов на связность при работе; колонка *UTIL* содержит данные о загрузке узлов кластера, которая позволяет следить за загруженностью узлов в процессе всего времени прогона модели; последняя колонка *AVE. TIME* содержит данные о загрузке узлов в процессе выполнения конкретных модулей.

Результаты моделирования в таблице 3.3 при $T = 50$ мс показывают, что кластер работает при слабой нагрузке: например, узел *N_8* загружен на 3,3%, а узел *N_3* – на 14,1% (значения уровня нагрузки переведены в проценты). По наиболее загруженному узлу определяется уровень его критической нагрузки: при приближении этого уровня к 100% нагрузка приближается к критической для всего кластера. Колонка *ENTRIES* определяет число входов в каждый узел при моделировании: число входов 10000, равное числу прогонов моделирующей программы, говорит о том, что модули всех задач прошли через данный

узел; общее число входов определяется алгоритмом, который реализуется приложением. Нулевое число входов свидетельствует о том, что данный узел не задействован при выполнении всех задач в модели, что является признаком ошибки в алгоритме или о недостаточности объема выборки. Результаты всех экспериментов получены при объеме выборки 10000 (число прогонов модели), который определялся опытным путем. Дополнительно в каждой таблице указаны оценки математического ожидания *MEAN* и среднеквадратичного отклонения *ST.DEV* времени выполнения всего приложения в кластере. В каждой таблице приведена гистограмма времени выполнения приложения в модели: высоты прямоугольников пропорциональны накопленным частотам, «шаги» гистограммы – это интервалы частотных классов. По гистограмме делается вывод о работоспособности модели в стационарном режиме (или режиме работы без перегрузки). Длина «хвоста» гистограммы свидетельствует о времени выполнения задач, большего, чем в основных случаях.

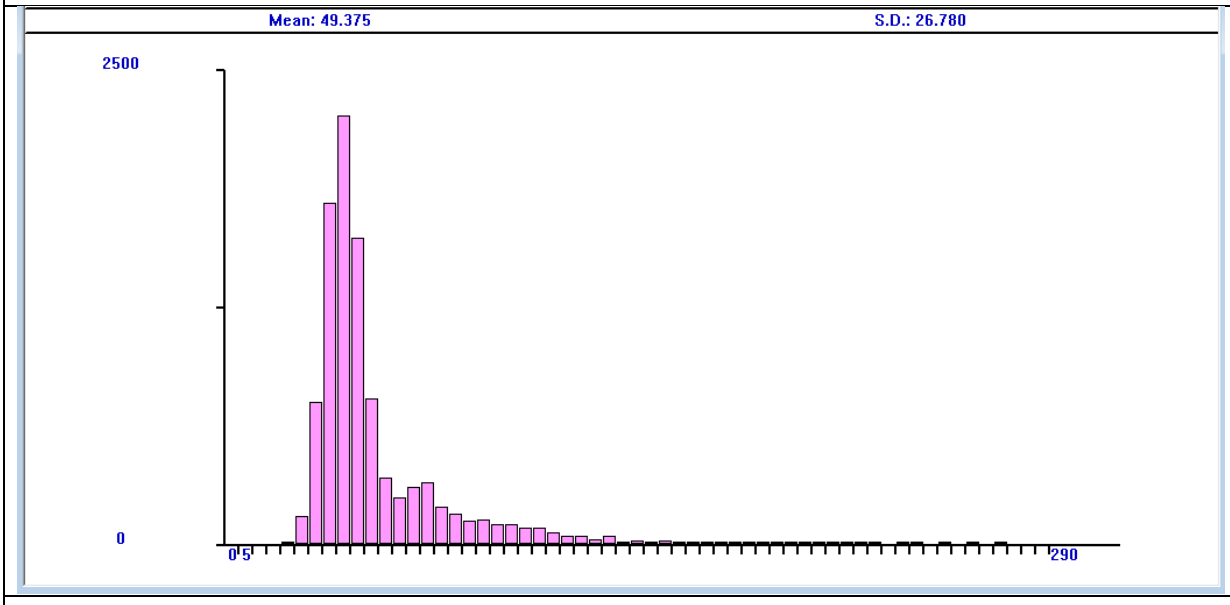
Таблица 3.3

Конвейерный MPMD-режим: слабая загрузка кластера

T=50 мс

FACILITY	ENTRIES	UTIL.	AVE. TIME
N_1	10000	0.140	6.990
N_6	9926	0.139	7.001
N_4	4982	0.070	7.004
N_5	2532	0.035	6.987
N_3	10000	0.141	7.045
N_9	4904	0.069	7.036
N_11	10000	0.140	6.985
N_7	4944	0.069	7.023
N_8	2398	0.033	6.976
N_2	5004	0.070	6.959
N_10	5096	0.071	7.002

MEAN=49.375 мс ST.DEV.=26.780 мс



Шаг 5; Выборка 10000

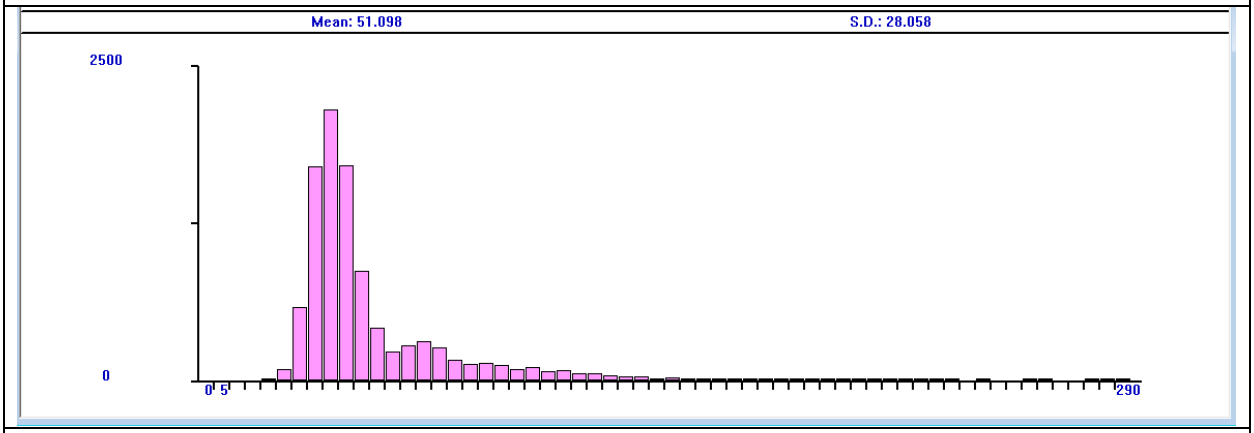
Таблица 3.4

Конвейерный MPMD-режим: слабая загрузка кластера

T=20 мс

FACILITY	ENTRIES	UTIL.	AVE. TIME
N_1	10002	0.351	7.014
N_6	9614	0.339	7.045
N_4	4802	0.168	7.007
N_5	2385	0.083	7.004
N_7	4812	0.168	6.985
N_8	2322	0.081	6.965
N_3	10000	0.352	7.032
N_2	5094	0.178	7.007
N_9	5025	0.177	7.038
N_11	10000	0.349	6.991
N_10	4975	0.175	7.052

MEAN=51.098 мс ST.DEV.=28.058 мс



Шаг 5; Выборка 10000

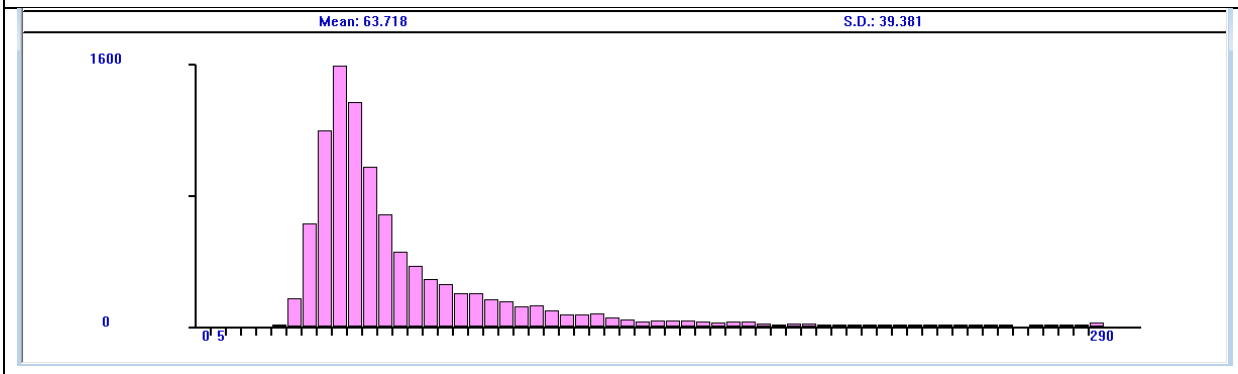
Таблица 3.5

Конвейерный MPMD-режим: нормальная загрузка кластера

T=10 мс

FACILITY	ENTRIES	UTIL.	AVE. TIME
N_1	10004	0.701	7.015
N_6	9883	0.694	7.030
N_4	4944	0.345	6.972
N_2	4990	0.350	7.009
N_3	10001	0.700	7.001
N_5	2474	0.173	7.012
N_10	4976	0.352	7.073
N_11	10000	0.703	7.033
N_9	5024	0.350	6.973
N_7	4938	0.348	7.042
N_8	2397	0.167	6.961

MEAN=63.718 мс ST.DEV.=39.381 мс



Шаг 5; Выборка 10000

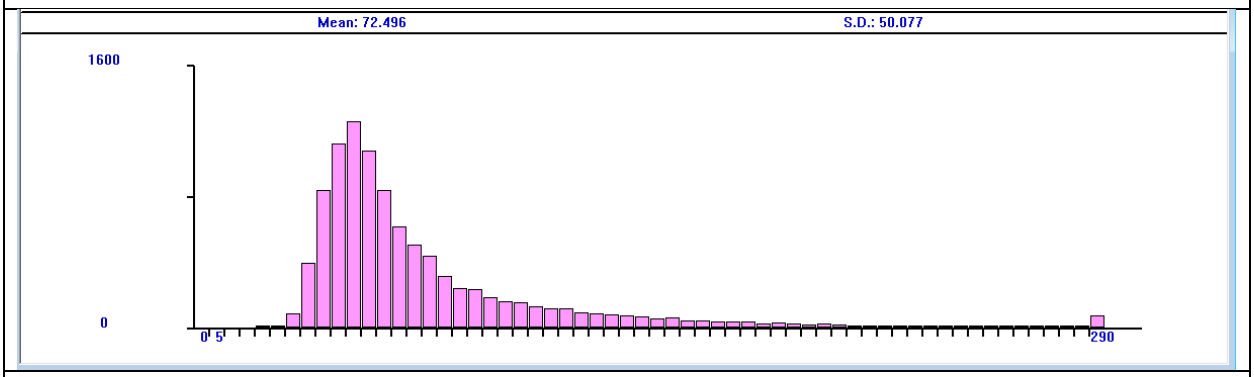
Таблица 3.6

Конвейерный MPMD-режим: нормальная загрузка кластера

T=9 мс

FACILITY	ENTRIES	UTIL.	AVE. TIME
N_1	10005	0.779	7.010
N_6	9925	0.771	6.996
N_4	5056	0.394	7.023
N_2	4984	0.386	6.966
N_3	10002	0.779	7.012
N_5	2476	0.194	7.041
N_10	4939	0.385	7.025
N_11	10001	0.781	7.036
N_9	5062	0.393	6.986
N_7	4868	0.378	6.996
N_8	2430	0.189	6.991

MEAN=72.496 мс ST.DEV.=50.077 мс



Шаг 5; Выборка 10000

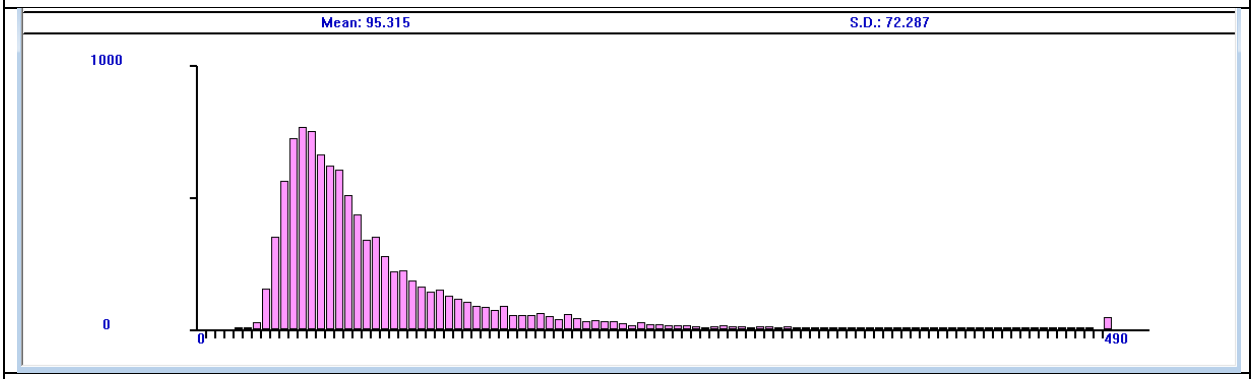
Таблица 3.7

Конвейерный MPMD-режим: нормальная загрузка кластера

T=8 мс

FACILITY	ENTRIES	UTIL.	AVE. TIME
N_1	10015	0.876	7.013
N_6	10085	0.881	6.997
N_4	4960	0.438	7.080
N_2	5013	0.440	7.035
N_3	10005	0.876	7.013
N_5	2492	0.217	6.987
N_10	5073	0.441	6.963
N_11	10001	0.871	6.975
N_9	4931	0.430	6.983
N_7	5125	0.448	6.999
N_8	2593	0.227	7.026

MEAN=95.315 мс ST.DEV.=72.287 мс



Шаг 5; Выборка 10000

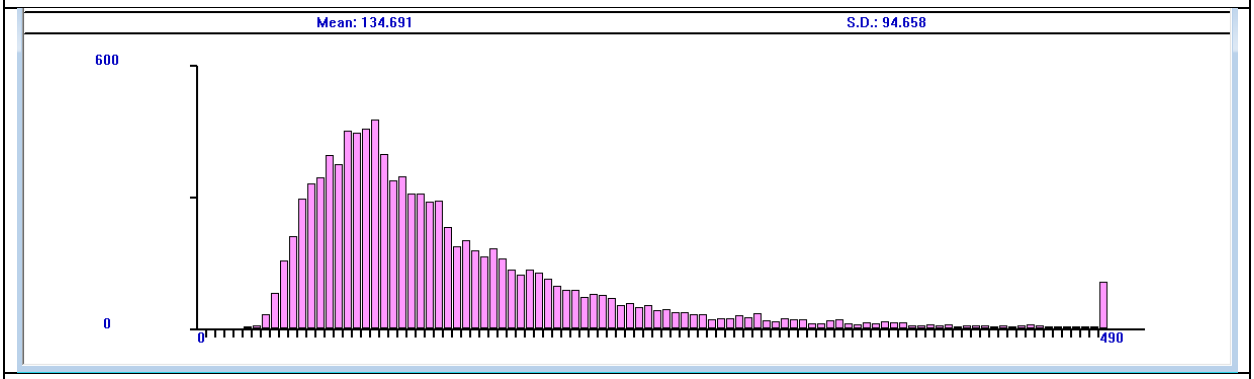
Таблица 3.8

Конвейерный MPMD-режим: критическая загрузка кластера

T=7.5 мс

FACILITY	ENTRIES	UTIL.	AVE. TIME
N_1	10018	0.936	7.025
N_6	10007	0.929	6.979
N_4	4981	0.461	6.958
N_2	4927	0.458	6.989
N_3	10002	0.929	6.979
N_5	2460	0.229	6.987
N_10	5017	0.467	7.002
N_11	10000	0.928	6.971
N_9	4984	0.466	7.025
N_7	5025	0.471	7.051
N_8	2470	0.229	6.956

MEAN=134.691 мс ST.DEV.=94.658 мс



Шаг 5; Выборка 10000

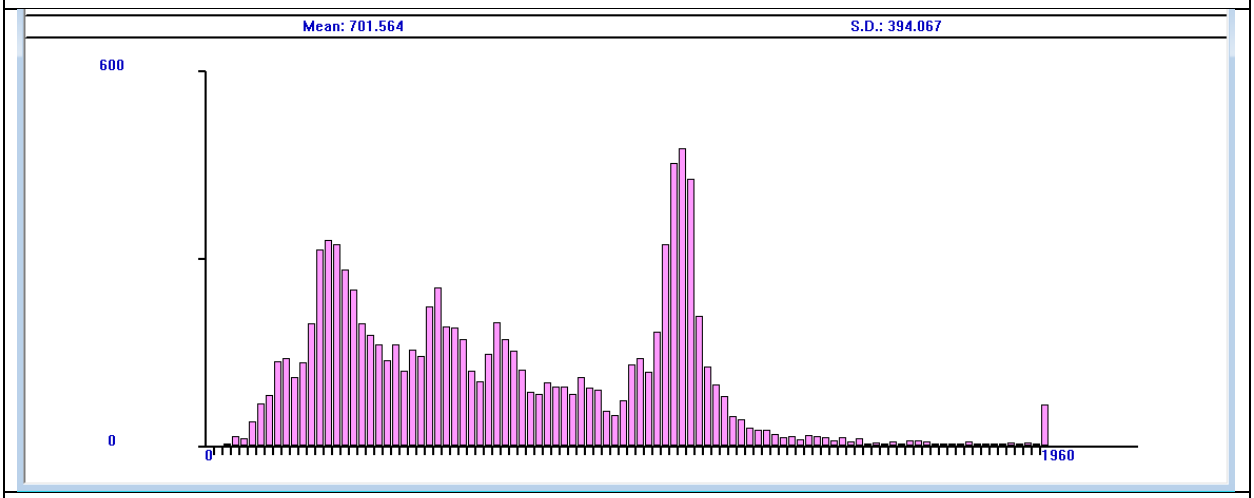
Таблица 3.9

Конвейерный MPMD-режим: перегрузка кластера

```
T=7 мс

FACILITY          ENTRIES   UTIL.     AVE. TIME
 N_1                10103    0.999       7.034
 N_6                 9846    0.971       7.009
 N_4                 4913    0.483       6.983
 N_2                 5131    0.507       7.021
 N_3                10016    0.992       7.041
 N_5                 2416    0.239       7.031
 N_10                4994    0.495       7.049
 N_11               10001    0.981       6.974
 N_9                 5020    0.496       7.031
 N_7                 4932    0.487       7.014
 N_8                 2469    0.242       6.980

MEAN=701.564 мс  ST.DEV.=394.067 мс
```




```
Шаг 20; Выборка 10000
```

Результаты, приведенные в таблице 3.4, также свидетельствуют о слабой загрузке кластера при T = 20 мс. Результаты в таблицах 3.5 (при T = 10 мс), таблице 3.6 (при T = 9 мс) и таблице 3.7 (T = 8 мс) соответствуют нормальной

загрузке, хотя среднее время *MEAN* выполнения приложения увеличилось почти вдвое с 51,1 мс до 95,3 мс. Результаты, приведенные в таблице 3.8, соответствуют критической загрузке кластера: среднее время *MEAN* выполнения приложения увеличилось до 134,7 мс, загрузка трех узлов составила более 92%. Более чем в 100 запусках приложений из 10000 время выполнения превысило 500 мс. Однако кластерная система, рассматриваемая как система массового обслуживания с регулярным потоком входных запросов находится на грани вхождения в нестационарный режим работы.

При дальнейшем уменьшении интервала времени *T* между последовательными запусками приложений до 7 мс кластерная система входит в нестационарный режим работы (таблица 3.9). С ростом объема выборки время выполнения увеличивается вплоть до прекращения работы моделирующей программы. Для входа в нестационарный режим достаточно, чтобы хотя бы загрузка одного из узлов достигнет почти 100% (узел *N_1* – 99,9%; узел *N_6* – 97,1; узел *N_3* – 99,2%; узел *N_11* – 98,1%). На наличие нестационарного режима указывает характер изменения фигуры в таблице 3.9 и резкое возрастание среднего времени *MEAN* = 701,6, которое при увеличении объема выборки тоже растет, что является одним из признаков нестационарности процесса.

Таким образом, предельное значение интервала времени *T* между последовательными запусками приложения в кластер, как видно по результатам моделирования, не должно быть меньшим, чем 7,5 мс, то есть интенсивность инициализаций λ не должна превышать 133,3 1/с. Результаты моделирования, приведенные в таблице 3.3, показывают, что при $T \geq 50$ мс приложения не оказывают влияния друг на друга ($\lambda \leq 20$ 1/с).

Обобщенные результаты статистических экспериментов с моделью ***NETWORK-MPMD*** параллельно-конвейерного выполнения приложения сведены в таблицу 3.10.

Табл. 3.10

T (мс)	λ (1/с)	$T_{Ср.}$ (мс)	K_F	$K_{Зам.}$
50	20	49,4	1	1
20	50	51,1	1,07	1,03
10	100	63,7	1,20	1,29
9	111,1	72,5	1,33	1,47
8	125	95,3	2,78	1,93
7,5	133,3	134,7	6,66	2,72

Здесь T – интервал между последовательными моментами запуска однотипных (но с разными данными) независимых по управлению и данным сетевых кластерных приложений; управляющая системная программа отправляет очередное приложение на обработку в сеть в соответствии с графом переходов состояний на рис. 3.13; λ – интенсивность инициализаций приложений; $T_{Ср.}$ – оценка математического ожидания времени выполнения одного приложения из нескольких, задействованных в параллельно-последовательном режиме; K_F – коэффициент, характеризующий относительный выигрыш по интенсивности запуска приложений при организации работы кластера в параллельно-конвейерном режиме – это в данном случае главный параметр, характеризующий эффективность использования вычислительного кластера; $K_{Зам.}$ – коэффициент, характеризующий замедление работы приложений при взаимном влиянии друг на друга при ожидании начала обработки в очередях к узлам.

Например, при значении $T = 8$ мс получена оценка среднего времени выполнения приложения $T_{Ср.} = 95{,}3$, значение $K_F = 125/20 = 2{,}78$ (то есть интенсивность λ потока запусков приложений увеличилась почти в 3 раза по сравнению с последовательной работой приложений, которая наблюдалась при λ = 20 1/с); $K_{Зам.} = 95{,}3/49{,}4 = 1{,}93$ – это значение определяется «накладными расходами» на организацию очередей ожидания к узлам, на которых расположена то или иная часть приложения, реализующая соответствующий оператор распределенного приложения.

Созданные исполнимые имитационные модели кластерных вычислительных систем позволяет оценить основные временные характеристики и за-

грузку узлов, что особенно важно при использовании кластера при конвейерном запуске приложений. Отличительной особенностью разработанных моделей является то, что основой их построения является автоматная и логико-вероятностная модели выполнения кластерного приложения, которые легко модифицируются и после некоторых дополнений могут использоваться в качестве исполнимых формализованных спецификаций при создании реального приложения.

Выводы по 3-й главе

1. Основанием, поддерживающим актуальность решаемых в главе задач, является ограниченность простых однородных систем кластеров, что усложняет создание систем, обеспечивающих высокий уровень структурно-функциональной динамики. Новые подходы к проектированию системной и функциональной архитектуры вычислительных кластеров основаны на организации эффективного использования и управления работой кластера за счет лучшей проблемной ориентации путем создания приложений и программного обеспечения промежуточного уровня *middleware*.

2. Подход, предлагаемый и использованный в главе, основан на концепции проектирования архитектуры, определяемой исполнимыми моделями, и которая является разновидностью предметно-ориентированного проектирования.

3. Отличительной особенностью моделей, предлагаемых в главе, является использование автоматного, сетевого автоматного и логико-алгебраического подходов к определению системной и функциональной архитектуры, которые могут применяться практически на всех уровнях предметной ориентации кластерных вычислительных систем и обеспечивают реализацию концепции архитектуры, формируемой при создании модели кластерной системы на различных уровнях абстракции – от концептуального представления до деталей реализации.

4. Показано, что главным эффектом от интерпретации предлагаемых автоматных моделей и методик является возможность их использования в качестве формализованных спецификаций при описании распараллеленных процессов в кластерных вычислительных системах на уровне задач, данных, алгоритмов и машинных инструкций.

5. Результаты проведенных статистических экспериментов показывают правильность построения вероятностно-автоматных СКУ-моделей и логико-вероятностных моделей и возможность их использования в качестве формализованных спецификаций. Анализ результатов статистического моделирования кластера, работающего в последовательно-параллельном режиме, показал полное соответствие второму законы Амдала с точностью до статистической погрешности.

6. Показано, что главным эффектом от интерпретации предлагаемых логико-алгебраических моделей является возможность их использования в качестве структурированных формализованных спецификаций при описании распараллеленных процессов в кластерных вычислительных системах на уровне задач, данных, алгоритмов и машинных инструкций.

7. Результаты проведенных статистических экспериментов показывают правильность построения логико-алгебраических моделей и логико-вероятностных моделей, так и формализованных спецификаций. Серия проведенных статистических экспериментов показала, что при работе в режиме параллельно-конвейерного запуска программ на исполнение на узлах кластерной вычислительной системы может быть получен выигрыш в производительности в 2-5 раз и более при контроле за соблюдением стационарного режима работы кластера.

ГЛАВА 4. МОДЕЛИ ФУНКЦИОНАЛЬНОЙ ОРГАНИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ КЛАСТЕРНОГО ТИПА С ПОВЫШЕННОЙ ОТКАЗОУСТОЙЧИВОСТЬЮ

Рассматриваются вычислительные системы кластерного типа с модульной избыточностью, когда задачи отказавших узлов автоматически перераспределяются на остальные работоспособные узлы. В реальных условиях эксплуатации вычислительного кластера рядовому собственнику или пользователю хорошо известные методы аппаратного резервирования кластера недоступны. Существуют и используются программные способы повышения отказоустойчивости кластера, которые могут быть ориентированы на различные операционные системы, на виртуализацию машин и баз данных. В числе этих способов – полнофункциональные отказоустойчивые методы, специфичные для приложений, основанных на задачах. Одни приложения допускают повторное выполнение нескольких реплик, а другие могут быть очень требовательными к времени решения задачи.

Целью данной главы является выбор и дальнейшая реализация подхода к формирования работоспособной (рабочей) структуры вычислительного кластера в условиях его частичной деградации, вызванной не только отказами программного обеспечения и оборудования, но и перегрузкой компьютеров.

Построение основной модели и ее расширения основано на формализованных логико-алгебраических спецификациях, что в целом соответствует современным концепциям разработок на основе моделей (MDD – *Model-Driven Development*) [108] и разработок приложений на основе правил предметной области (DDD – *Domain-Driven Design*) [110]. Предложенные модели возможно модифицировать для создания на их основе как отказоустойчивых вычислительных кластеров, так и распределенных грид-систем и волонтерских метакомпьютеров с повышенной отказоустойчивостью.

4.1. Вычислительные кластеры: особенности функционирования и отказоустойчивость

Вычислительный кластер – это разновидность параллельной или распределённой системы, состоящей из группы связанных между собой компьютеров, которые используются как единый, унифицированный компьютерный ресурс [149]. Из этого общепринятого определения следует, что выход из строя отдельных элементов вычислительного кластера нередко приводит к неработоспособности всей системы. В общем случае задача повышения отказоустойчивости является сложной и в большинстве случаев решается за счет введения избыточности – при возможности, резервируются компьютеры, коммутаторы, линии связи, электропитание и др.

Отказоустойчивыми кластерами (*Fault Tolerant Cluster*, FTC) [150] называются кластеры, отказ какого-либо узла в которых не приводит к полной неработоспособности всего кластера. В кластерах с холодным резервом, или «активный-пассивный», активный узел выполняет запросы, а пассивный ждет его отказа и включается в работу, когда отказ произойдет. Кластеры с горячим резервом, или «активный-активный», отличаются тем, что все узлы выполняют запросы, а в случае отказа одного нагрузка перераспределяется между оставшимися. Выделяются также кластеры с модульной избыточностью. Задачи отказавших узлов автоматически перераспределяются на остальные работоспособные узлы [151, 152].

Из состава отказоустойчивых кластеров выделяются кластеры непрерывной доступности (*Continuous Availability*, CA) [153], которыми пользователь может воспользоваться в произвольный момент времени и кластеры высокой доступности (*High Availability*, HA) [154], в течение отказа которых обслуживание пользователей будет прервано на время восстановления работоспособной конфигурации.

Существует большое число методов обеспечения отказоустойчивости вычислительных кластеров, различающихся предъявляемыми к ним требованиями. Источниками отказов могут быть оборудование, программное обеспечение, сетевые коммуникации, человеческие ошибки и окружающая среда.

Этапы реакции системы мониторинга кластера на отказы [155, 156]:

1. Ограничение сбоев в одной области системы для предотвращения распространения последствий на всю систему.

2. Обнаружение отказов. На этом этапе происходит распознавание неординарных событий в системе.

3. Диагностика. Производится при отсутствии полезного результата от реализации предыдущего этапа, то есть если не получена информация о местоположении и свойствах неисправности.

4. Реконфигурация. Действия этого этапа выполняются, когда обнаружены неисправность и постоянный отказ; в этом случае система может переконфигурировать свои компоненты либо для замены неисправного компонента, либо для его изоляции от остальной системы.

5. Восстановление. При восстановлении используются методы устранения последствий неисправностей. Основные подходы к восстановлению основаны на маскировке неисправностей, повторной попытке и откате.

6. Перезапуск. После восстановления неповрежденной информации реализуется какой-либо метод перезапуска приложения: возобновление всех операций с точки обнаружения сбоя; перезапуск только некоторых процессов; полная перезагрузка системы.

7. Ремонт неисправного компонента или переключение на резерв; при этом работа системы может быть прервана.

8. Реинтеграция восстановленного компонента в систему.

4.2. Содержательное описание модели, обеспечивающей повышение отказоустойчивости вычислительного кластера

Как отмечалось в первой главе, частная сеть кластера позволяет осуществить обмен данными между узлами. Должны передаваться не только данные, но и управляющие сообщения, при помощи которые узлы опрашивают друг друга. В литературе по вычислительным кластерам эти процессы опросов, получения и анализа ответов называются «сердцебиение», или «пульс» (*англ. heartbeats*) [157]. При помощи командных сообщений также осуществляется инициирование перенастройки и синхронизации работы узлов кластера. Для контроля ответов от узлов кластера при выполнении ими основных функций возможно использовать хорошо известный разработчикам компьютерных сетей механизм тайм-аутов. Другая дополнительная возможность заключается в обнаружении ошибочного результата как следствия сбоя, что не всегда возможно, если узел после опроса «молчит» [157].

Дальнейший выбор метода повышения отказоустойчивости вычислительного кластера будет основан на способности управляющего узла следить за аппаратным и программном обеспечением кластера. Возможна такая организация кластера, при которой управляющим узлом является один из его выделенных узлов. Выбранный метод не должен требовать модификации существующего программного обеспечения и также не должен требовать обязательного наличия операционной системы с открытым исходным кодом. Вместе с тем следует отметить, что задача осложняется тем, что исполнительные механизмы реализации взаимодействия узлов кластера обычно скрыты от приобретателя и потребителя. Хотя существуют коммерческие программные реализации различных фирм, но их код далеко не всегда является открытым.

В реальных условиях эксплуатации вычислительного кластера рядовому собственнику или пользователю хорошо известные методы аппаратного резервирования кластера недоступны. В то же время существуют и используются программные способы повышения отказоустойчивости кластера, тем более

что функциональность большинства вычислительных кластеров обеспечивается на уровнях прикладного (решение задачи) и промежуточного (реализация интерфейса с пользователем) программного обеспечения. Известны программные средства обеспечения отказоустойчивости кластера, которые могут быть ориентированы на различные операционные системы, на виртуализацию машин и баз данных [158].

Особый интерес представляют методы обеспечения отказоустойчивости вычислительных кластеров, основанные на реализации алгоритмов: так называемые ABFT-методы (*Algorithm Based Fault Tolerant techniques*) [158]. В числе этих отказоустойчивых методов – полнофункциональные отказоустойчивые методы, специфичные для приложений, основанных на задачах. Одни приложения допускают повторное выполнение нескольких реплик, а другие могут быть очень требовательными к времени решения задачи. Существуют приложения, реализующие различные алгоритмы восстановления потерянных данных, а также приложения, запускаемые повторно при меньшем числе процессоров [159].

Подход, используемый в настоящей работе для организации программного метода повышения отказоустойчивости кластера, соответствует работам [157, 158, 159]. Отличительной особенностью его реализации является то, что для упрощения программирования, доступного для прикладного или системного программиста, предложена масштабируемая логико-алгебраическая исполнимая модель процесса формирования работоспособной (рабочей) структуры вычислительного кластера в условиях его частичной деградации, вызванной не только отказами программного обеспечения и оборудования, но и перегрузкой компьютеров.

Некоторые общие известные вопросы анализа надежности и мониторинга функционирования вычислительных систем, рассматриваемых как сложные системы, описаны в работе [160]. Однако следует отметить трудоемкость нахождения временных параметров, характеризующих работу кластера

при отказах разного рода. Часть событий относится к числу очень редких, происходящих в течение часов, дней, месяцев, делающих сбор статистических данных очень трудоемким и требующим больших затрат времени. Другие события являются следствием неправильной эксплуатацией, неожиданным отключением электропитания, вредным влиянием окружающей среды.

Содержательная постановка задачи заключается в следующем. Пусть, например, время выполнения задачи условно равно единице. В процессе подготовки решения задачи на кластере, включающем, например, 16 узлов, задача разбивается на одинаковые части, тогда время выполнения каждой части равно 1/16.

В случае полностью надежного кластера общее время выполнения задачи без учета времени на сбор результатов выполнения всех подзадач будет равно 1/16. В случае отказа какого-либо одного узла в процессе выполнения своей подзадачи осуществляется ее перезапуск на резервном узле или на любом из узлов, только что выполнившим свою часть задания. Тогда для получения окончательного результата потребуется время 1/16 + 1/16 = 2/16. Такое время сохраняется и при первоначальных (если все отказы происходят за время выполнения основной части заданий) отказах 2, 3, ..., 8 узлов, если пренебречь временем загрузки тех частей заданий от общего задания, которые не были выполнены из-за отказов узлов.

Для работы кластера в режиме SPMD (*Single Program – Multiple Data*) обычно на каждый узел кластера параллельно в широковещательном режиме работы коммутатора загружаются копии одной и той же программы с настроечными параметрами и с копиями данных, сгруппированных в одном файле. В подобном несколько идеализированном случае время выполнения задания будет определяться так, как это показано в таблицах табл. 4.1 и табл. 4.2 при исходных $N = 32$ узлах и $N = 16$ в вычислительном кластере. Результаты легко распространяются на другое исходное число узлов.

Табл. 4.1

Число оставшихся узлов кластера	Время полного выполнения задания
$M = N = 32$	1/32
$16 \le M < 32$	2/32
$8 \le M < 16$	4/32
$4 \le M < 8$	8/32
$2 \le M < 4$	16/32
$M = 1$	32/32

Табл. 4.2

Число оставшихся узлов кластера	Время полного выполнения задания
$M = N = 16$	1/16
$8 \le M < 16$	2/16
$4 \le M < 8$	4/16
$2 \le M < 4$	8/16
$M = 1$	16/16

4.3. Формализованная исполнимая модель вычислительного кластера с повышенной отказоустойчивостью

Построение формализованной исполнимой модели вычислительного кластера с повышенной отказоустойчивостью основано на известном процессе «аварийного переключения» – перезапуска предварительно настроенных узлов кластера. Этот процесс является частью общего процесса отказоустойчивой кластеризации. Перезапуск происходит без вмешательства администратора после обнаружения аппаратных или программных сбоев [151].

В процессе формализации модели вводится в рассмотрение множество $Y = \{y_1, y_2, \ldots, y_K\}$ для обозначения множества узлов вычислительного кластера, и отдельно определено одноэлементное множество $\{y_0\}$, где y_0 обозначает выделенный управляющий узел кластера, соответствует адресу одного из узлов кластера и определяется сервером пользователя. В логико-алгебраической модели обозначения $y_0, y_1, y_2, \ldots, y_K$ представляют собой предметные константы в исчислении предикатов первого порядка. Здесь $K \ge N$, а разность $C = K - N$ равна числу резервных узлов. Вводится характеристическая функция (унарный предикат) $S_0(y)$ множества Y. В дальнейшем определении модели вводятся также вспомогательные характеристические функции $S(y)$, $S_1(y)$ и $S_2(y)$ для множеств Y, Y_1 и Y_2 соответственно, используемые при переборе узлов или изменении состава узлов в кластере.

В реальном приложении в качестве констант $y_0, y_1, y_2, \ldots, y_K$ могут рассматриваться *MAC*-адреса (*Medium Access Control addresses* – адреса управления доступом к среде передачи данных), или аппаратные адреса Ethernet, – физические адреса узлов вычислительного кластера, являющихся единицами сетевого оборудования в стандарте компьютерных сетей Ethernet [161]. В качестве адресов при передаче данных внутри подсетей для идентификации узлов кластера как компонентов сети InfiniBand используются *LID* (*Local ID*, локальные идентификаторы) [162], которые также можно использовать в качестве предметных констант, то есть значений предметной переменной *y*.

Имена подзадач, решаемых узлами кластера, задаются в модели множеством $X = \{x_1, x_2, \ldots, x_N\}$. Для множества X введена характеристическая функция $Q(x)$ и вспомогательная характеристическая функция $Q_1(x)$.

В логико-алгебраических операционных выражениях (ЛАОВ) исполнимой модели круглые, квадратные и фигурные скобки являются элементами синтаксиса. Выражение $[A]\{B\}$ соответствует репликации группы операций B, причем число повторений составного выражения B задается выражением A. Выражением $[\alpha](\{B_1\} \vee \{B_2\})$ задается выбор одной из альтернатив $\{B_1\}$ или $\{B_2\}$ в зависимости от истинности или ложности логического условия α. Особенности и источники выбранной нотации ЛАОВ были указаны во 2-й главе. Логика выполнения ЛАОВ определяется не только условиями, заключенными в квадратные скобки. Основной операцией является операция «←» модификации предиката, результатом выполнения которой является событие, заключающееся во включении какой-либо предметной константы или кортежа, содержащего константы, в область истинности какого-либо предиката, либо в исключении из нее.

В дальнейшем там, где это не будет противоречить семантике модели, будут отождествляться объекты в формальной модели с объектами предметной области – вычислительного кластера. Например, при выполнении опера-

ции модификации предиката $Send^*(y_0, y) \leftarrow \boldsymbol{true}$ в узле y_0 в его область истинности включается кортеж $< y_0, y >$, что интерпретируется как отправка сообщения от управляющего узла кластера y_0 рабочему узлу кластера, номер, или адрес, которого определяется как значение предметной переменной y. Звездочка при имени предиката означает, что в операции передачи и приема сообщения участвуют два узла – передающий y_0 и принимающий y. После получения подтверждения узлом y_0 об успешном приеме сообщения узлом y по умолчанию выполняется операция $Send^*(y_0, y) \leftarrow \boldsymbol{false}$. Подобная интерпретация логико-алгебраических выражений ЛАОВ практически не затрудняет составление сетевых приложений. От программиста требуется только знание основ сетевого программирования.

Другое выражение для операции модификации предиката в формальной модели $Ready^{**}(y, y_0) \leftarrow \boldsymbol{true}$ соответствует событию передачи ответного сообщения от рабочего узла y управляющему узлу y_0 кластера; по завершении приема сообщения в узле y_0 по умолчанию выполняется операция $Ready^{**}(y, y_0) \leftarrow \boldsymbol{false}$, соответствующая успешной передаче сообщения от узла y узлу y_0. Двойная звездочка означает, что инициатором передачи сообщения является узел y, а узел y_0 ожидает сообщение.

4.4. Логико-алгебраическая исполнимая модель приложения, предназначенного для повышения отказоустойчивости вычислительного кластера

Формализованная исполнимая модель предназначена для последующего использования в качестве формализованных спецификаций для программы, реализующей подход к повышению отказоустойчивости вычислительного кластера. Модель описана четырьмя логико-алгебраическими выражениями и носит декларативно-поведенческий характер. Все выражения ЛАОВ реализуются управляющей программой, размещенной на управляющем узле y_0 кластера, выдающего команды рабочим узлам кластера через коммутатор частной

сети. Кроме основного управляющего узла кластера имеется дополнительный узел, находящийся в горячем резерве.

Выполнение модели происходит в четыре этапа.

Этап 1. Управляющий узел вычислительного кластера определяет готовые к использованию узлы, которые имеются в наличии, путем опроса каждого узла; от каждого узла ожидается ответ о готовности до истечения тайм-аута t_1:

$$L_1 = [\forall_{\boldsymbol{Do}}(y \in Y)]\{S(y) \leftarrow S_0(y)\};\ Ready^{**}(y, y_0) \leftarrow \boldsymbol{false};$$
$$[\exists_{\boldsymbol{Any}}(y \in Y)\ S(y)](\{Send^{*}(y_0, y) \leftarrow \boldsymbol{true},\ \{TimeOut^{*}(t_1) \leftarrow \boldsymbol{true},$$
$$Ready^{**}(y, y_0) \leftarrow \boldsymbol{true}\},\ [Ready^{**}(y, y_0)]\ (\{S(y) \leftarrow \boldsymbol{false},$$
$$S_1(y) \leftarrow \boldsymbol{true}\} \vee S(y) \leftarrow \boldsymbol{false}) \vee E). \qquad (4.1)$$

Комментарии. $[\forall_{\boldsymbol{Do}}(y \in Y)]\{S(y) \leftarrow S_0(y)\}$ – для каждого узла y из множества узлов Y выполняется присваивание (модификация предиката) $S(y) \leftarrow S_0(y)$, где $S_0(y)$ и $S(y)$ – две одинаковые характеристические функции множества Y (из них $S(y)$ – вспомогательная), $\forall_{\boldsymbol{Do}}$ – квантифицированный оператор, указывающий на то, чтобы все действия в фигурных скобках были выполнены. При помощи оператора $\exists_{\boldsymbol{Any}}(y \in Y)\ S(y)$ последовательно из области истинности унарного предиката $S(y)$ последовательно перебираются константы $y_1, y_2, \ldots, y_K$, обозначающие в реальной системе рабочие узлы кластера.

Здесь и далее в описаниях этапов использована нотация ЛАОВ, основанная на интеграции алгебры алгоритмов В. М. Глушкова с логикой предикатов, которая отражена в работах [116, 118, 123].

При каждом успешном выборе последовательно выполняются действия, заключенные в фигурные скобки. Операция $Send^{*}(y_0, y) \leftarrow \boldsymbol{true}$ имитирует отправку в рабочий узел кластера опрашивающего сообщения о его готовности к работе. Затем в узле y_0 выдерживается тайм-аут t_1, инициируемый выполнением операции $TimeOut^{*}(t_1) \leftarrow \boldsymbol{true}$. В течение тайм-аута ожидается получение

ответа от опрашиваемого рабочего узла кластера $Ready^{**}(y, y_0) \leftarrow \boldsymbol{true}$ о его готовности к работе. Его готовность отмечается путем формирования области истинности предиката: $S_1(y) \leftarrow \boldsymbol{true}$. Так последовательно осуществляется выбор из множества Y доступных для опроса узлов кластера. Если за время таймаута t_1 рабочий узел не отвечает, то высказывание $Ready^{**}(y, y_0)$ остается ложным, и данный узел в область истинности итогового предиката $S_1(y)$ не включается, что следует из выражения:

$$[Ready^{**}(y, y_0)]\ (\{S(y) \leftarrow \boldsymbol{false}, S_1(y) \leftarrow \boldsymbol{true}\} \vee \{S(y) \leftarrow \boldsymbol{false}\}).$$

По завершении первого этапа модели должно быть сформировано множество Y_1 с характеристической функцией $S_1(y)$, содержащее все работоспособные узлы, способные работать в кластере. Далее рассматриваются следующие ситуации: если узлов достаточно для образования кластера, то следом выполнятся загрузка узлов программами и данными; если годных узлов оказалось больше, чем было необходимо для решения задачи, то они включаются в резерв, и, наконец, если узлов меньше, минимально необходимого числа для решения задачи, то управляющий узел, а через него и пользователь, оповещаются от этом.

Этап 2. Формирование состава рабочих узлов кластера, именование или нумерация этих узлов для использования имен или номеров в программах; происходит аварийный выход $Exit1(R_0) \leftarrow \boldsymbol{true}$, если не хватает узлов для формирования кластера для заранее определенного числа подзадач X; отношение R_0, связывающее подзадачи из множества X с узлами кластера из множества Y_1 передается всем узлам кластера:

$$\begin{aligned} L_2 = {} & [\exists_{\boldsymbol{Any}}(x \in X)\ Q(x)]([\exists_{\boldsymbol{Any}}(y \in Y_1)\ S_1(y)](\{R_0(x, y) \leftarrow \boldsymbol{true}, \\ & Q(x) \leftarrow \boldsymbol{false}, S_1(y) \leftarrow \boldsymbol{false}, S_2(y) \leftarrow \boldsymbol{true}, \\ & Q_1(x) \leftarrow \boldsymbol{true}\} \vee Exit1(R_0) \leftarrow \boldsymbol{true}) \vee E); \\ & [\forall_{\boldsymbol{Do}}(y \in Y_2)]\{SendD^*(y_0, R_0, y) \leftarrow \boldsymbol{true}\}. \end{aligned} \qquad (4.2)$$

Комментарии. В квадратные скобки выражения (4.2) заключены операторы, с которыми связаны по умолчанию логические условия, принимающие истинные значения, если выполнение операторов успешно завершено и ложные в противном случае. При помощи двух последовательно применяемых операторов выбора $[\exists_{Any}(x \in X)\ Q(x)]([\exists_{Any}(y \in Y_1)\ S_1(y)]$ отмечается именованная часть задачи, как значение предметной переменной x в области истинности унарного предиката $Q(x)$ и именованный узел кластера, как значение предметной переменной y в области истинности унарного предиката $S_1(y)$. Если выбор произведен успешно, то кортеж $<x, y>$ с фиксированными значениями его компонент включается в область истинности бинарного предиката $R_0(x, y)$. Затем значения переменных x и y исключаются из областей истинности унарных предикатов $Q(x)$ и $S_1(y)$ соответственно. Таким образом части общей задачи в модели «распределяются» по узлам вычислительного кластера.

При выполнении выражения (4.2) может случиться ситуация, когда значений переменной y будет недостаточно для заданного заранее числа значений переменной x. Для реального кластера это означает, что имеющегося числа рабочих узлов будет недостаточно для размещения частей исходного задания. В этом случае выдается сообщение пользователю; в модели этой выдаче соответствует операция $Exit_1(R_0) \leftarrow \boldsymbol{true}$. Если же рабочих узлов кластера достаточно для решения поставленной задачи, то результат, то есть таблица, соответствующая области истинности бинарного предиката R_0, передается в широковещательном режиме, то есть параллельно, всем рабочим узлам кластера. Этому действию в модели соответствует завершающая часть выражения (4.2):

$$[\forall_{Do}(y \in Y_2)]\{SendD^*(y_0, R_0, y) \leftarrow \boldsymbol{true}\},$$

где $SendD^*$ – тернарный (трехместный) предикат, моделирующий параллельную передачу данных R_0 от управляющего узла y_0 ко всем остальным рабочим узлам кластера. Широковещательная передача в модели задается оператором

$\forall_{\boldsymbol{Do}}(y \in Y_2)$ и реализуется оператором модификации предиката $SendD^*(y_0, R_0, y) \leftarrow \boldsymbol{true}$.

Этап 3. Загрузка программ и данных в узлы кластера в режиме широковещания; последующая обработка данных D в каждом узле кластера; передача результатов D_1 в управляющий узел y_0; формирование и передача в узел y_0 значений предиката $R_2(x, y)$, связывающего результаты выполнения подзадач с работоспособными узлами:

$$L_3 = [\forall_{\boldsymbol{Do}}(x, y) \in R_0]\{R_1(x, y) \leftarrow R_0(x, y), R_3(x, y) \leftarrow R_0(x, y)\};$$
$$[\forall_{\boldsymbol{Do}}(x, y) \in R_0]\{SendD^*(y_0, D, y) \leftarrow \boldsymbol{true}, TimeOut^*(t_2) \leftarrow \boldsymbol{true}\};$$
$$[\exists_{\boldsymbol{Any}}(x, y) \in R_1](\{Ready_1D^{**}(y, D_1, y_0) \leftarrow \boldsymbol{true},$$
$$Ready_2D^{**}(y, R_2(x, y), y_0) \leftarrow \boldsymbol{true}, R_1(x, y) \leftarrow \boldsymbol{false}\} \vee E). \qquad (4.3)$$

Комментарии. В первой строке выражения (4.3) формируются вспомогательные бинарные предикаты и одноименные им отношения $R_1(x, y)$ и $R_3(x, y)$ – копии предиката и одноименного отношения $R_0(x, y)$, сформированного в результате выполнения выражения (4.2), которое связывает части задания и рабочими узлами. Во второй строке описана загрузка программ и данных в узлы кластера в режиме широковещания. В третьей и четвертой строках в управляющем узле кластера происходит перебор пар кортежей $<x, y>$, составляющих область истинности бинарного предиката $R_1(x, y)$; каждый рабочий узел осуществляет передачу результирующих данных в управляющий узел; эти действия задаются операциями с тернарным предикатом $Ready_1D^{**}(y, D_1, y_0) \leftarrow \boldsymbol{true}$ для каждого y.

Аналогично происходит передача в узел y_0 значений предиката $R_2(x, y)$, указывающего на связь результатов выполнения подзадач с работоспособными узлами $Ready_2D^{**}(y, R_2(x, y), y_0) \leftarrow \boldsymbol{true}$. Напомним, что двойные звездочки соответствуют одноименной передаче сообщений из узла y в узел y_0 и приему этих сообщений управляющим узлом y_0.

Этап 4. Проверка результатов на наличие узлов, не выполнивших свои подзадачи:

$$L_4 = [\forall(x, y)\, R_2(x, y)](Exit_2(D_1) \leftarrow \textbf{\textit{true}} \ \vee$$
$$\vee \{[\forall_{\textbf{\textit{Do}}}(x, y) \in R_3]\{R_3(x, y) \leftarrow \neg\, R_2(x, y)\},$$
$$[\forall_{\textbf{\textit{Do}}}(x, y) \in R_3]\{S_0(y) \leftarrow \textbf{\textit{false}}\},$$
$$[\forall_{\textbf{\textit{Do}}}(x, y) \in R_2]\{Q_1(x) \leftarrow \textbf{\textit{false}}\}\}). \qquad (4.4)$$

Комментарии. Если все подзадачи выполнены, что соответствует в модели истинности всех значений предиката $R_2(x, y)$, то работа кластера признается успешной с выдачей результата $Exit_2(D_1) \leftarrow$ ***true*** пользователю через интерфейс *middleware*. В противном случае производится исключение отказавших узлов из множества Y_0, определение неисполненных подзадач, исключение уже исполненных запросов из множества X и повторное выполнение для неисполненных запросов действий, предписанных логико-алгебраическими выражениями L_1, L_2, L_3.

4.5. Пример работы вычислительного кластера с повышенной отказоустойчивостью

Для проверки модели рассматривается работа вычислительного кластера при $N = 16$, то есть в случае, когда для успешного выполнения задания, состоящего из 16 частей требуется нормальная работа 16 узлов кластера. В условном примере принято, что в процессе работы некоторые рабочие узлы могут выходить из строя и становиться недоступными. Считается, что причиной недоступности могут быть не только отказы оборудования и программного обеспечения, но и случаи перегрузки узлов при разрешенном доступе к ним со стороны пользователей общедоступной сети.

Перед началом работы кластера известно, что его оборудование включает 22 узла, из которых 16 рабочих и 6 резервных:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22

В процессе последовательного опроса всех узлов выясняется, что узлы 5, 8, 14 недоступны:

1, 2, 3, 4, ~~5~~, 6, 7, ~~8~~, 9, 10, 11, 12, 13, ~~14~~, 15, 16, 17, 18, 19, 20, 21, 22

К рабочим узлам кластера подсоединяются резервные узлы 17, 18, 19, и образуется полный состав узлов кластера для выполнения заранее подготовленных и загруженных 16 частей задания:

1, 2, 3, 4, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19

В процессе выполнения частей задания рабочие узлы 6, 10, 12, 17 не выдали ответ, а остальные узлы успешно справились с выполнением своих частей задания:

1, 2, 3, 4, ~~6~~, 7, 9, ~~10~~, 11, ~~12~~, 13, 15, 16, ~~17~~, 18, 19

Для выполнения оставшихся частей задания выбраны и перезагружены невыполненными ранее частями задания узлы 1, 2, 3, 4:

1, 2, 3, 4

Теперь узлы 2 и 3 не справились с выполнением своих частей задания:

1, ~~2~~, ~~3~~, 4

Для выполнения оставшихся частей задания выбраны и перезагружены узлы 1 и 4, которые успешно завершили выполнение задания:

1, 4

Время выполнения задания без распараллеливания принято за единицу. Тогда полное время выполнения задания складывается из трех интервалов: 1/16+1/16+1/16=3/16. Таким образом, коэффициент ускорения равен 1/(3/16) = 5,33, то есть в процессе выполнения задания кластер не только справился с решением задачи, но и позволил получить выигрыш от ее распараллеливания.

При вычислении времени выполнения задания не учитывались задержки в коммутаторе и время загрузки заданий. Полагалось, что эти временные параметры составляют лишь небольшую часть общего времени выполнения задания. При статистическом моделировании системы эти задержки вводятся дополнительно.

4.6. Об отказоустойчивости систем распределенных грид-вычислений, волонтерских кластеров и кластеров с высокой производительностью

Принципы реализации грид-систем и волонтерских систем имеют много общего с кластерной технологией, но проигрывают ей во времени доступа к вычислительным ресурсам. Распределенные узлы грид-систем могут подключаться и отключаться в процессе работы, поэтому их конфигурация, в отличие от конфигурации вычислительного кластера, как правило, нестабильна. Нестабильность конфигурации характерна также для кластерных вычислительных систем при наличии независимого доступа к компьютерам со стороны пользователей, компьютеры которых в инфраструктуре кластера подключены к локальной сети общего назначение, например, через коммутатор Ethernet, как показано на рис. 1.1 в первой главе.

Логико-алгебраическая исполнимая модель формирования на основе выражений ЛАОВ (4.1), (4.2), (4.3) и (4.4) конфигурации отказоустойчивого вычислительного кластера соответствует не только случаям сильной его деградации в процессе применений, но и случаям, когда ресурсы рабочих узлов отвлекаются по различным причинам на работу с другими приложениями. Это может приводить к ограничению доступа к узлам кластера. Поэтому программное обеспечение, построенное на основании выражений (4.1), (4.2), (4.3) и (4.4), позволяет обеспечить повышенную работоспособность кластера и при реализации на нем эпизодических (ограниченных) посторонних, «не кластерных», приложений. Однако при ограничении на возможность использования широковещательной передачи данных, что характерно для глобальных сетей, завершающее выражение в формуле (4.2):

$$[\forall_{\boldsymbol{Do}}(y \in Y_2)\, S_2(y)]\{SendD^*(y_0, R_0, y) \leftarrow \boldsymbol{true}\}$$

необходимо заменить при программной реализации следующим выражением (4.5), моделирующим последовательную передачу данных удаленным узлам кластера:

$$[\exists_{\boldsymbol{Any}}(y \in Y_2)\, S_2(y)](\{SendD^*(y_0, R_0, y) \leftarrow \boldsymbol{true}, S_2(y) \leftarrow \boldsymbol{false}\} \vee E). \qquad (4.5)$$

Аналогично, с такой же целью, в выражении (4.3) часть выражения:

$$[\forall_{\boldsymbol{Do}}(x, y) \in R_0]\{SendD^*(y_0, D, y) \leftarrow \boldsymbol{true}, TimeOut^*(t_2) \leftarrow \boldsymbol{true}\}$$

необходимо заменить на выражение (4.6):

$$[\exists_{\boldsymbol{Any}}(x, y) \in R_0]\{SendD^*(y_0, D, y) \leftarrow \boldsymbol{true}, TimeOut^*(t_2) \leftarrow \boldsymbol{true},$$
$$R_0(x, y) \leftarrow \boldsymbol{false}\} \vee E)\}. \qquad (4.6)$$

Таким образом, новые выражения (4.7) и (4.8) для модификаций частей модели L'_2 и L'_3, моделирующих работу отказоустойчивой вычислительной системы в глобальной сети без широковещания будут иметь следующий вид:

$$L'_2 = [\exists_{\boldsymbol{Any}}(x \in X)\, Q(x)]([\exists_{\boldsymbol{Any}}(y \in Y_1)\, S_1(y)](\{R_0(x, y) \leftarrow \boldsymbol{true},$$
$$Q(x) \leftarrow \boldsymbol{false}, S_1(y) \leftarrow \boldsymbol{false}, S_2(y) \leftarrow \boldsymbol{true},$$
$$Q_1(x) \leftarrow \boldsymbol{true}\} \vee Exit(R_0) \leftarrow \boldsymbol{true}) \vee E);$$
$$[\exists_{\boldsymbol{Any}}(y \in Y_2)\, S_2(y)](\{SendD^*(y_0, R_0, y) \leftarrow \boldsymbol{true}, S_2(y) \leftarrow \boldsymbol{false}\} \vee E), \quad (4.7)$$

$$L'_3 = [\forall_{\boldsymbol{Do}}(x, y) \in R_0]\{R_1(x, y) \leftarrow R_0(x, y), R_3(x, y) \leftarrow R_0(x, y)\};$$
$$[\exists_{\boldsymbol{Any}}(x, y) \in R_0]\{SendD^*(y_0, D, y) \leftarrow \boldsymbol{true}, TimeOut^*(t_2) \leftarrow \boldsymbol{true},$$
$$R_0(x, y) \leftarrow \boldsymbol{false}\} \vee E)\};$$
$$[\exists_{\boldsymbol{Any}}(x, y) \in R_1](\{ReadyD^{**}(y, D_1, y_0) \leftarrow \boldsymbol{true},$$
$$ReadyD^{**}(y, R_2(x, y), y_0) \leftarrow \boldsymbol{true}, R_1(x, y) \leftarrow \boldsymbol{false}\} \vee E). \qquad (4.8)$$

Остальные выражения L_1 (1) и L_4 (4.8) имеют прежний вид, так как они не используют описание широковещательной передачи данных.

4.7. Исследование и обоснование модели вычислительных кластеров с повышенной отказоустойчивостью

Использование статистических моделей, построенных на основе логико-алгебраических формализаций, позволяет получать основные характеристики для оценки производительности и чувствительности к отказам вычислительных кластеров. При определенных допущениях возможно оценивать некоторые характеристики вычислительных кластеров на основе обычных вероятностных расчетов, что позволит также оценить достоверность результатов статистического моделирования. Предлагается метод повышения отказоустойчивости вычислительных кластеров, основанный на комбинировании использования статистических и вероятностных моделей. Метод основан на первоначальном выборе работоспособных узлов кластера, определении необходимого числа узлов для выполнения распараллеленных вычислений в виде одинаковых и независимых друг от друга подзадач. Если после отбора узлов оказалось, что их числа недостаточно для параллельного выполнения всего вектора подзадач, то выполняется его часть, а остальные части выполняются на следующем шаге. Метод «работоспособен» и при сильной деградации кластера, вызванной различными причинами. Вычисления доводятся до конца, даже если после деградации кластера останется всего один рабочий узел. Естественно, время решения задачи при этом увеличивается.

Статистическая модель построена на основе использования логико-алгебраической исполнимой модели, то есть на основе выражений (4.1), (4.2), (4.3) и (4.4) и включает 4 этапа их применения.

При использовании вероятностной модели принято допущение, что все вычислительные узлы кластера, предоставляемые для решения задачи пользователя, одинаковы и находятся в одинаковых условиях эксплуатации. Для работы в составе кластера «заявлены» r рабочих узлов из n ($n > r$), предоставленных пользователю; ($n - r$) узлов – резервные. Каждый узел находится в работоспособном состоянии с вероятностью p; для оценки вероятности обнаруже-

ния k (k = 1, 2, …, n) работоспособных узлов из общего их числа n можно воспользоваться известной формулой Бернулли (4.9) для биномиального распределения вероятностей [163]:

$$f(k;n,p) = P(X = k) = C_n^k p^k (1-p)^{n-k}\ , \qquad (4.9)$$

где $C_n^k = \frac{n!}{(n-k)!k!}$ – биномиальный коэффициент.

Кумулятивная функция биномиального распределения вероятностей (4.10) позволяет определить вероятность того, что число отказавших узлов вычислительного кластера не будет превышать величины k [163]:

$$F(k;p,n) = P(X \le k) = \sum_{i=0}^{k} C_n^i p^i (1-p)^{(n-i)}. \qquad (4.10)$$

Математическое ожидание случайной величины, имеющей биномиальное распределение, равно $m = n \cdot p$, а дисперсия равна $\sigma^2 = n \cdot p \cdot (1 - p)$ [163].

Статистическая модель работы вычислительного кластера, построенная на основе выражений (4.1), (4.2), (4.3) и (4.4), проверялась путем вычисления некоторых ее характеристик на основании формулы (4.10) и далее произведена проверка некоторых основных характеристик на соответствие. В табл. 4.3 приведены значения следующих характеристик:

p – коэффициент готовности узла кластера к работе;

m^* – оценка математического ожидания числа работоспособных узлов в кластере, вычисленная по результатам статистического эксперимента;

m – математическое ожидание числа работоспособных узлов в кластере, вычисленное по формуле $m = n \cdot p$;

q_{15} – вероятность, вычисленная по формуле (4.10) при n = 22, k = 15 и при заданном в таблице значении p, того, что число рабочих узлов будет меньше, или равно 15.

Последний параметр q_{15} – это вероятность того, что для решения задачи обработки данных в параллельном режиме потребуется два или более временных интервалов $T_{\text{Обр.}}$, так как $r > k$. Однако при малых значениях q_{15} вероятность того, что время обработки займет один временной интервал $T_{\text{Обр.}}$, будет намного выше. В последней колонке табл. 4.3 указано число необходимых интервалов при заданных p, n = 22 и r = 16.

Табл. 4.3

Всего узлов $n = 22$; из них заявлены рабочими $r = 16$

p	m^*	m	q_{15}	$T_{Обр.}$(с)
0,4	8,77	8,8	0,9981	2
0,5	10,99	11,0	0,9738	2
0,6	13,17	13,2	0,8416	2
0,7	15,38	15,4	0,5058	2
0,8	17,60	17,6	0,1329	1
0,9	19,78	19,8	$0{,}4390 \cdot 10^{-2}$	1
0,95	20,88	20,9	$0{,}6835 \cdot 10^{-4}$	1
0,99	21,77	21,78	$0{,}1495 \cdot 10^{-8}$	1
0,999	21,97	21,98	$0{,}1680 \cdot 10^{-15}$	1
0,9999	22,0	22,0	0,0…	1
1	22,0	22	0	1

Например, при $p = 0{,}8$ вероятность q_{15} того, что время решения задачи займет 2 или более временных интервалов $T_{Обр.}$ равна 0,1329. Вероятность же того, что при заданных значениях p и q_{15} время обработки составит один интервал $T_{Обр.}$, равна $(1 - q_{15}) = 0{,}8671$. При дальнейшем увеличении коэффициента готовности p вероятность $(1 - q_{15})$ того, что время обработки составит величину одного интервала $T_{Обр.}$, приближается к единице. Значения m^* и m очень близки, что указывает на достоверность результатов статистического моделирования, при котором могут быть определены также и другие характеристики производительности и отказоустойчивости кластера.

В табл. 4.4 приведены аналогичные результаты для вычислительного кластера с большим числом рабочих $r = 128$ и $(n - r) = 22$ резервных узла. Эти результаты иллюстрируют масштабируемость разработанных моделей в широких пределах.

Табл. 4.4

Всего узлов $n = 150$; из них заявлены рабочими $r = 128$

p	m^*	m	q_{127}	$T_{Обр}$(с)
0,4	60,0	60	1,0	2
0,5	75,0	75	1,0	2
0,6	90,0	90	1,0	2
0,7	105,05	105	1,0	2
0,8	120,09	120	0,94	2
0,9	135,0	135	0,0256	1
0,99	148,5	148,5	0,0…	1
0,999	149,85	149,85	0,0…	1
1	150	150	0	1

В результате подтверждена достоверность предлагаемых моделей на основе логико-алгебраических выражений. К их положительным особенностям относятся следующие:

– предложенные логико-алгебраические модели, соответствуют их назначению для использования в качестве формализованных спецификаций при разработке функциональной архитектуры вычислительных систем кластерного типа;

– согласование статистических имитационных моделей с логико-алгебраическими исполнимыми моделями позволило обеспечивать масштабируемость создаваемых моделей, позволяющую получать характеристики производительности и отказоустойчивости как вычислительных кластеров так и распределенных грид-систем.

4.8. Выводы по 4-й главе

1. Неработоспособность или ограниченная работоспособность рабочего кластера может быть вызвана как относительно редкими отказами оборудования, так и перегрузками, вызванными фоновыми и волонтерскими задачами.

2. Успешное решение проблемы повышения отказоустойчивости и работоспособности вычислительных кластеров программными средствами связана

с возможностями пользователей и собственников кластеров по созданию и обслуживанию резерва аппаратных средств, а также по созданию отказоустойчивых приложений.

3. Предложенные логико-алгебраические исполнимые модели функциональной организации вычислительных кластеров повышенной отказоустойчивости являются масштабируемыми и, соответственно, являются масштабируемыми и реализуемыми отказоустойчивые приложения, поэтому предложенные модели возможно модифицировать для создания на их основе работоспособных распределенных грид-систем и волонтерских метакомпьютеров.

4. В основу создания отказоустойчивых и работоспособных функциональных архитектур кластерных и вычислительных грид-систем целесообразно положить подход, основанный на формализованных спецификациях, которыми являются исполнимые операционные модели.

5. В результате подтверждена достоверность предлагаемых моделей на основе логико-алгебраических выражений. К положительным особенностям предложенного метода обеспечения повышенной отказоустойчивости вычислительных кластеров относятся следующие:

– предложенные логико-алгебраические модели, соответствуют их назначению для использования в качестве формализованных спецификаций при разработке функциональной архитектуры вычислительных систем кластерного типа;

– согласование статистических имитационных моделей с логико-алгебраическими исполнимыми моделями позволило обеспечивать масштабируемость создаваемых моделей, позволяющую получать характеристики производительности и отказоустойчивости как вычислительных кластеров так и распределенных грид-систем;

– разработанные модели носят исполнимый, операционный и событийный характер, что позволяет учитывать функциональную архитектуру ограниченность моделей и принятые допущения не противоречат реальным вариантам использования большинства из распространенных вычислительных кластеров и других волонтерских вычислительных систем.

6. Статистические эксперименты показали, что работоспособность кластера сохраняется при его деградации до одного узла с соответствующим понижением производительности, но позволяет выполнить задание полностью в

большинстве случаев. Результат приведенного анализа базового примера показал, что в условиях сильной, но не стопроцентной, деградации вычислительного кластера было не только обеспечено выполнение разбитого на части задания, но и достигнуто ускорение вычислений в 5,33 раза за счет частичного сохранения параллельного режима. Модель расширена на случай анализа отказоустойчивости систем распределенных грид-вычислений и волонтерских кластеров с большим и нестабильным числом узлов.

ЗАКЛЮЧЕНИЕ

В диссертационной работе предлагается использовать новые методы организации функциональной архитектуры вычислительных систем кластерного типа на основе исполнимых моделей и правил предметной области. Рассматриваемые с позиций искусственного интеллекта, основанного на знаниях и правилах, подобные модели представляют собой сценарии действий, содержащие как процедурную, так и декларативную составляющие модели представления знаний. Декларативная составляющая задается при помощи исчисления предикатов первого порядка, а процедурная, или императивная, – алгебраическими моделями алгоритмов и частичными конечными автоматами. Знания и правила предметной области – это продукционные правила, инкорпорированные в событийные модели сценариев и частичных автоматов предметной области – приложений и вычислительных систем кластерного типа. В результате разработан ряд новых логико-алгебраических, автоматных и статистических моделей, отличающихся от известных удобной и эффективной интерпретацией и реализуемостью.

1. На основе проведенного анализа современного состояния и развития архитектуры вычислительных систем предложена концепция организации функциональной архитектуры вычислительных систем кластерного типа, *отличающаяся* базированием на исполнимых моделях, знаниях и правилах предметной области и *позволяющая* осуществлять предметную ориентацию вычислительных кластеров на конкретного пользователя, использующего кластер как конечный продукт.

2. Предложены методы разработки функциональной архитектуры вычислительных кластеров, основанные на исполнимых логико-алгебраических операционных моделях, *отличающиеся* от известных тем, что они основаны

на знаниях и правилах предметной области, на назначении вычислительных кластеров и на анализе событий, происходящих при работе приложений и управляющих программ, и *позволившие* создавать новые исполнимые модели для совершенствования вычислительных систем кластерного типа, определяющие их функциональную архитектуру, масштабируемость и соответствие предметной области.

3. Предложены новые исполнимые модели вычислительных систем кластерного типа на *макроуровне*, расширяющие диапазон их функциональности и применимости: логико-алгебраические операционные и статистические модели «Круглый стол», «Резидент-агенты», «Собрание 1», «Собрание 2», *позволяющие* реализовывать современные диалоговые параллельные взаимодействия при обработке данных в больших сообществах пользователей при помощи масштабируемых приложений, реализуемых кластером. Время выполнения статистических моделей при 1000 – 10000 прогонах занимает не более 2 – 25 секунд, что позволяет оперативно получать характеристики производительности Проведенные статистические эксперименты с данными моделями позволяют оценить характеристики производительности при различных уровнях параллельной обработки данных в кластерах, выбрать число узлов кластера и портов коммутатора вычислительных кластеров и других вычислительных систем с аналогичной системной и функциональной архитектурой.

4. Разработаны новые детализированные сетевые автоматные, вероятностные автоматные и логико-алгебраические операционные модели вычислительных систем кластерного типа на *микроуровне*, *отличающиеся* исполнимым характером и *позволяющие* произвести проектирование приложений на уровне сетевых операций и дать оценку производительности кластера в целом. Статистические эксперименты продемонстрировали полное соответствие полученных характеристик производительности для базовых примеров закону Амдала.

5. Предложенный метод повышения производительности вычислительных кластеров при выполнении сетевых приложений *позволил*, в *отличие от известных*, повысить их пропускную способность при развертывании операторов распределенного алгоритма на узлах кластера и выполнять вычисления, определяемые передачами управления и данных между узлами в параллельно-конвейерном режиме. Статистические эксперименты продемонстрировали повышение пропускной способности в 2-5 раз.

6. Предложена исполнимая логико-алгебраическая операционная модель функциональной организации вычислительных систем кластерного типа с повышенной отказоустойчивостью, *основанная* на динамическим резервировании и многократной замене узлов и *позволяющая* при ее реализации обеспечить отказоустойчивость кластера в условиях сильной деградации, обусловленной внутренними и внешними причинами. Статистические эксперименты показали, что работоспособность кластера сохраняется при его деградации до одного узла с соответствующим понижением производительности, но позволяет выполнить задание полностью в большинстве случаев. Результат проведенного анализа типового примера показал, что в условиях сильной, но не стопроцентной, деградации вычислительного кластера было не только обеспечено выполнение разбитого на части задания, но и достигнуто ускорение вычислений в 5,33 раза за счет частичного сохранения параллельного режима. Модель расширена на случай анализа отказоустойчивости систем распределенных грид-вычислений и волонтерских кластеров с большим и нестабильным числом узлов (до 100-150 и более вычислительных узлов).

СПИСОК ЛИТЕРАТУРЫ

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.

2. Tyson M. Elon Musk fires up ‘the most powerful AI cluster in the world’ to create the 'world's most powerful AI' by December – system uses 100,000 Nvidia H100 GPUs on a single fabric. Published July 22, 2024.
URL: https://www.tomshardware.com. Дата обращения 02.06.2025. Accessed 22 July 2024.

3. Полмиллиона GPU за 4 месяца: как строится самый мощный кластер в мире. https://www.braintools.ru/ Дата обращения 02.06.2025. Accessed 10 August 2025.

4. Pleiter D. Supercomputer Architectures: Current State and Future Trends. The AQTIVATE Project, European Union’s HORIZON MSCA Doctoral Networks programme, under Grant Agreement No. 101072344. September 2023. P. 1-38.

5. ParTec AG: A more efficient supercomputer for the AI revolution. Frankfurt, Bloomberg, 2024. P. 1-43.

6. Boldyrev A., Ratnikov F., Shevelev A. Approach to Finding a Robust Deep Learning Model. IEEE Access. 2025. 27 p.
https://doi.org/10.48550/arXiv.2505.17254.

7. Мишенин Р. М., Костенецкий П. С. Моделирование потока задач вычислительного кластера НИУ ВШЭ с использованием SLURM Simulator. XIX Всероссийская научная конференция с международным участием «Параллельные вычислительные технологии (ПаВТ'2025)», г. Москва, 8-10 апреля 2025 г. С. 324.

8. Efficiency of Machine Learning Tasks on HPC Devices / G. Promyslov, A. Efremov, Y. Ilyasov, V. Pisarev, A. Timofeev. Proceedings of Conference “Parallel Computational Technologies (PCT'2025)”, Moscow, 2025. P. 56-81.

9. Kostenetskiy P.S., Kozyrev V.I., Chulkevich R.A., Raimova A.A. Enhancement of the Data Analysis Subsystem in the Task-Efficiency Monitoring System HPC TaskMaster for the cHARISMa Supercomputer Complex at HSE University. Parallel Computational Technologies (PCT'2024). Communications in Computer and Information Science, Springer, 2024. Vol. 2241. P. 49-64. DOI: 10.1007/978-3-031-73372-7_4.

10. Сластников С. А., Жукова Л. Ф., Семичаснов И. В. Приложение поиска, анализа и прогнозирования данных в социальных сетях. Информационные технологии и вычислительные системы. 2024. № 1. С. 97-108. doi: 10.14357/20718632240110

11. Machine learning models for predicting risks of MACEs for myocardial infarction patients with different VEGFR2 genotypes / A. Kirdeev, K. Burkin., A. Vorobev, E. Zbirovskaya, G. Lifshits, K. Nikolaev, E. Zelenskaya, M. Donnikov, L. Kovalenko, I. Urvantseva, M. Poptsova. Frontiers in Medicine. 2024. Vol. 11. Article 1452239. P. 1-10. https://doi.org/10.3389/fmed.2024.1452239.

12. Chehab G., Martinez D. Computer Clusters, Types, Uses and Applications. Last updated: March 18, 2024. URL: https://www.baeldung.com/cs/computer-clusters-types. (Доступ свободный, дата обращения 02.06.2025).

13. Архитектурные аспекты параллелизма.

URL: http://www.intuit.ru/studies/courses/4447/983/lecture/14921?Page=3. (Доступ свободный, дата обращения 02.06.2025).

14. Аль-Хулайди А. А., Садовой Н. Н. Анализ существующих программных пакетов в кластерных системах // Вестник ДГТУ, 2010. Т. 10. № 3(46). С. 303-310.

15. Хенесси Д.Л., Паттерсон Д.А. Компьютерная архитектура. Количественный подход. 5-е изд. М.: Техносфера, 2016. 936 с.

16. Архитектуры вычислительных систем: кластерные системы. URL: https://uzor.belturs.ru/arkhitektury-vychislitel-nykh-sistem-klasternyye-sistemy. (Доступ свободный, дата обращения 02.06.2025).

17. Кластер (группа компьютеров). URL: https://ru.wikipedia.org/wiki/Кластер_(группа_компьютеров). (Доступ свободный, дата обращения 02.06.2025).

18. Эффективные кластерные решения.

URL: https://www.ixbt.com/cpu/clustering.shtml?ysclid=maavg9yxkq334655198. (Доступ свободный, дата обращения 02.06.2025).

19. Чем распределенная ВС отличается от кластерной? URL: https://ru.stackoverflow.com/. (Доступ свободный, дата обращения 02.06.2025).

20. Курносов М.Г. Многопроцессорные и многомашинные вычислительные системы. URL: https://www.mkurnosov.net/teaching/files/pct-lecture1.pdf.

21. Многопроцессорные и многомашинные вычислительные системы. URL: https://intuit.ru/studies/educational_groups/960/courses/460/lecture/10345?page=2&ysclid=maawtf2w8o719310938. (Доступ свободный, дата обращения 02.06.2025).

22. Виды кластерных систем.

URL: https://vuzlit.com/963152/vidy_klasternyh_sistem. (Доступ свободный, дата обращения 02.06.2025).

23. Computer cluster. URL: https://en.wikipedia.org/wiki/Computer_cluster. (Доступ свободный, дата обращения 02.06.2025).

24. What are High-Performance Computing Clusters and use cases of High-Performance Computing Clusters? URL: https://www.devopsschool.com/. (Доступ свободный, дата обращения 02.06.2025).

25. Introduction to Parallel Computing Tutorial. URL: https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial. (Доступ свободный, дата обращения 02.06.2025).

26. Olson C.F. Parallel algorithms for hierarchical clustering. URL: https://www.sciencedirect.com/science/article/pii/016781919500017I?via%3Dihub. URL: https://doi.org/10.1016/0167-8191(95)00017-I. (Доступ свободный, дата обращения 02.06.2025).

27. Суперкомпьютеры «Яндекса».
URL: https://ru.wikipedia.org/wiki/Суперкомпьютеры_«Яндекса». (Доступ свободный, дата обращения 02.06.2025).

28. Высокоскоростная сеть Ангара: архитектура и результаты применения / Симонов А.С., Жабин И.А., Куштанов Е.Р., Макагон Д.В., Семенов А.С., Щербак А.Н // Вопросы кибербезопасности. – 2019. – № 4(32) . – С. 46–53.

29. Высокоскоростная сеть Ангара. URL: http://www.nicevt.ru/vysokoskorostnaya-set-angara/. (Доступ свободный, дата обращения 02.06.2025).

30. Ангара (интерконнект).
URL: https://ru.wikipedia.org/wiki/Ангара_(интерконнект). (Доступ свободный, дата обращения 02.06.2025).

31. Евреинов Э.В., Косарев Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Изд-во «Наука», Сибирское отделение. Новосибирск, 1966. 308 с.

32. Димитриев Ю.К., Хорошевский В.Г. Вычислительные системы из мини-ЭВМ; Под ред. Э.В. Евреинова. – М.: Радио и связь, 1982. – 304 с.

33. Хорошевский В.Г. Инженерный анализ функционирования вычислительных машин и систем. – М.: Радио и связь, 1987. – 254 с.

34. Миренков М.М. Параллельное программирование для многомодульных вычислительных систем. – М.: Радио и связь, 1989. – 320 с.

35. Хорошевский В.Г. Архитектура вычислительных систем. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2005. – 510 с.

36. Корнеев В.В. Вычислительные системы. – М.: Гелиос АРВ, 2004. – 512 с.

37. Степаненко С.А. Мультипроцессорные среды суперЭВМ. Масштабирование эффективности. – М.: ФИЗМАТЛИТ, 2016. – 312 с.

38. Feng Liu, Haitao Wu, Xiaochun Lu, Xiyang Liu. Parallel Distributed Acceleration Based on MPI and OpenMP Technology // International Journal of Grid Distribution Computing. Vol. 8, No.6, 2015, pp.171-184.

39. Mittal G., Kesswani N, Goswami K. A Survey of Current Trends in Distributed, Grid and Cloud Computing // International Journal of Advanced Studies in Computer Science and Engineering (IJASCSE). Vol. 2, Issue 3, 2013, pp. 1-6.

40. Kahanwal B., Singh T. P. The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle // International Journal of Latest Research in Science and Technology. ISSN (Online): 2278-5299. Vol. 1, Issue 2, 2012, pp. 183-187.

41. Jungle Computing: Distributed Supercomputing beyond Clusters, Grids, and Clouds / Frank J. Seinstra, Jason Maassen, Rob V. van Nieuwpoort, Niels Drost, Timo van Kessel, Ben van Werkhoven, Jacopo Urbani, Ceriel Jacobs, Thilo Kielmann, Henri E. Bal. Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, 2010, pp. 1-31.

42. Kumar R. Comparison between Cloud Computing, Grid Computing, Cluster Computing and Virtualization // International Journal of Modern Computer Science and Applications (IJMCSA). Vol. 3, Issue No.1, January, 2015, pp. 42-47.

43. Kaur K., Rai A. K. A Comparative Analysis: Grid, Cluster and Cloud Computing // International Journal of Advanced Research in Computer and Communication Engineering. Vol. 3, No. 3, 2014, pp. 5730-5734.

44. Samah Mawia Ibrahim Omer, Amin Babiker A. Mustafa, Fatema Abdallah Elmahdi Alghali. Comparative study between Cluster, Grid, Utility, Cloud and Autonomic computing // IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE). e-ISSN: 2278-1676, p-ISSN: 2320-3331. Vol. 9, Issue 6, Ver. III, 2014, pp. 61-67.

45. An Assessment of Beowulf-class Computing for NASA Requirements: Initial Findings from the First NASA Workshop on Beowulf-class Clustered Computing / Sterling, T., Cwik, T., Becker, D., Salmon, J., Warren, M., Nitzberg, B. // IEEE Aerospace Conference Proceedings. July 1998, pp. 1–16. DOI:10.1109/AERO.1998.682207.

46. MPI: A Message-Passing Interface Standard. Version 3.1. Message Passing Interface Forum, June 4, 2015. – 836 p. URL: https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf. (Доступ свободный, дата обращения 02.06.2025).

47. Early Experience with Scientific Applications on the Blue Gene/L Supercomputer / G. S. Almasi, G. V. Bhanot, et al. // Lecture Notes in Computer Science 3648. – 2005. – P. 560–570. DOI:10.1007/11549468_63.

48. Hamada T., Nitadori K. 190 TFlops astrophysical N-body simulation on a cluster of GPUs. In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10). IEEE Computer Society, Washington, DC, USA, 2010. – P. 1–9. DOI:10.1109/SC.2010.1

49. Enabling the Modern Data Center – RDMA for the Enterprise. URL: https://www.infinibandta.org/wp-content/uploads/2019/05/IBTA_WhitePaper_May-20-2019.pdf. (Доступ свободный, дата обращения 02.06.2025).

50. RDMA_over_Converged_Ethernet. URL: https://en.wikipedia.org/wiki/RDMA_over_Converged_Ethernet. (Доступ свободный, дата обращения 02.06.2025).

51. Deploying HPC Cluster with Mellanox. InfiniBand Interconnect. Solutions. Reference Design. Rev 1.3. January 2017. – 40 p.

52. На пути к созданию отечественного суперкомпьютера субэкзафлопсной производительности. URL: https://www.ospcon.ru/files/media/Simonov.pdf. (Доступ свободный, дата обращения 02.06.2025).

53. Sugon. URL: https://en.wikipedia.org/wiki/Sugon. (Доступ свободный, дата обращения 02.06.2025).

54. InfiniBand Clustering. Delivering Better Price/Performance than Ethernet. URL: https://network.nvidia.com/pdf/whitepapers/IB_vs_Ethernet_Clustering_WP_100.pdf. (Доступ свободный, дата обращения 02.06.2025).

55. Список 500 самых мощных компьютеров мира. URL: https://parallel.ru/computers/top500.list61.html. (Доступ свободный, дата обращения 02.06.2025).

56. Anderson D. P. BOINC: A System for Public-Resource Computing and Storage. URL: https://boinc.berkeley.edu/grid_paper_04.pdf. (Доступ свободный, дата обращения 02.06.2025).

57. Зелов С. Кластерные технологии. URL:
https://compress.ru/article.aspx?id=9958&ysclid=maawhrd9vh678328188. (Доступ свободный, дата обращения 02.03.2025).

58. Cluster Computing Market Overview. URL: https://www.marketresearchfuture.com/reports/cluster-computing-market-1746. (Доступ свободный, дата обращения 02.06.2025).

59. Информационно-аналитический центр по параллельным вычислениям. URL: parallel.ru. (Доступ свободный, дата обращения 02.06.2025).

60. Кластеры на многоядерных процессорах / И. И. Ладыгин, А. В.Логинов, А. В. Филатов, С. Г. Яньков. – М: Издательский дом МЭИ, 2008. 112 с.

61. Кокоц А. В. Разработка программной модели функционирования кластерной вычислительной системы // Электронный журнал «Вычислительные сети. Теория и практика» / Network Journal. Theory and practice/ ВС/NW, 2016, № 2 (29). URL: https://network-journal.mpei.ac.ru/. (Доступ свободный, дата обращения 02.06.2025).

62. Hofert M. High Performance Computing Cluster Setup: A Tutorial. // Journal of Data Science: JDS. November 2024. P. 1-22. DOI:10.6339/24-JDS1159

63. Deploying HPC Cluster with Mellanox. InfiniBand Interconnect Solutions. Reference Design. Rev 1.3. Mellanox Technologies. January 2017. 40 p.

64. Центр коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М. В. Ломоносова. URL: https://parallel.ru/cluster. (Доступ свободный, дата обращения 02.06.2025).

65. Чернявцева В. Яндекс создал три мощнейших в России суперкомпьютера. URL: https://nplus1.ru/news/. (Доступ свободный, дата обращения 02.06.2025).

66. Россия внезапно ворвалась в мировой топ самых мощных суперкомпьютеров. URL: https://www.cnews.ru/news/top/2021-11-16_rossijskie_superkompyutery. (Доступ свободный, дата обращения 02.06.2025).

67. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы. Изд-во Питер, 2020. 1010 с.

68. Воеводин Вл. В., Жуматий С. А. Вычислительное дело и кластерные системы. М.: Изд-во МГУ, 2007. 150 с.

69. Куликов Г. Г., Антонов В. В., Конев К. А. Кластерное программное обеспечение автоматизированной информационной информационной системы // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». 2018. Т. 18, № 2. С. 19–28.

70. Разница между грид-вычислениями и кластерными вычислениями. URL: https://www.geeksforgeeks.org/difference-between-grid-computing-and-cluster-computing/. (Доступ свободный, дата обращения 02.06.2025).

71. Sadashiv N., Kumar S. M. D. Cluster, Grid and Cloud Computing: A Detailed Comparison // The 6th International Conference on Computer Science & Education (ICCSE 2011), August 3-5, 2011. SuperStar Virgo, Singapore. 2011, pp. 477-482.

72. Цымблер М.Л. Оценка эффективности параллельных вычислений. URL: https://mzym.susu.ru/courses/obsolete/sup/04%20Evaluation.pdf. (Доступ свободный, дата обращения 02.06.2025).

73. Закон Амдала. URL: https://ru.wikipedia.org/wiki/Закон_Амдала. (Доступ свободный, дата обращения 02.06.2025).

74. Закон Густавсона – Барсиса. URL: https://ru.wikipedia.org/wiki/Закон_Густавсона_—_Барсиса. (Доступ свободный, дата обращения 02.06.2025).

75. Amdahl, Gene M. Validity of the single processor approach to achieving large scale computing capabilities // Proceedings of the April 18-20, 1967, Spring Joint Computer Conference on – AFIPS '67 (Spring). P. 483–485. doi:10.1145/1465482.1465560. S2CID 195607370.

76. Хилл, Марк Д.; Марти, Майкл Р. (2008). Закон Амдала в эпоху многоядерных процессоров // Computer. 41 (7): 33–38. CiteSeerX 10.1.1.221.8635. doi:10.1109/MC.2008.209.

77. Рафиев, Ашур; Аль-Хайянни, Мохаммед А. Н.; Ся, Фэй; Шафик, Ришад; Романовский, Александр; Яковлев, Алекс (1 июля 2018 г.). Модели ускорения и масштабирования мощности для гетерогенных многоядерных систем. IEEE Transactions on Multi-Scale Computing Systems. 4 (3): P. 436–449. doi:10.1109/TMSCS.2018.2791531. ISSN 2332-7766. S2CID 52287374.

78. Аль-Хаянни, Мохаммед А. Ноаман; Ся, Фэй; Рафиев, Ашур; Романовский, Александр; Шафик, Ришад; Яковлев, Алекс. Закон Амдала в контексте гетерогенных многоядерных систем – обзор // IET Computers & Digital Techniques. 14 (4): 2020. 133–148. doi:10.1049/iet-cdt.2018.5220. ISSN 1751-8601. S2CID 214415079.

79. Hill, Mark D., Marty, Michael R. Amdahl's Law in the Multicore Era // Computer. 41 (7): 2008. P. 33–38. CiteSeerX 10.1.1.221.8635. doi:10.1109/MC.2008.209.

80. Rafiev, Ashur; Al-Hayanni, Mohammed A. N.; Xia, Fei; Shafik, Rishad; Romanovsky, Alexander; Yakovlev, Alex. Speedup and Power Scaling Models for Heterogeneous Many-Core Systems // IEEE Transactions on Multi-Scale Computing Systems. 4 (3): 2018. P. 436–449. doi:10.1109/TMSCS.2018.2791531. ISSN 2332-7766. S2CID 52287374.

81. Al-hayanni, Mohammed A. Noaman; Xia, Fei; Rafiev, Ashur; Romanovsky, Alexander; Shafik, Rishad; Yakovlev, Alex. Amdahl's law in the context of

heterogeneous many-core systems – a survey // IET Computers & Digital Techniques. 14 (4): 2020. P. 133–148. doi:10.1049/iet-cdt.2018.5220. ISSN 1751-8601. S2CID 214415079.

82. McCool, Michael D.; Robison, Arch D.; Reinders, James (2012). 2.5 Performance Theory. Structured Parallel Programming: Patterns for Efficient Computation. Elsevier. P. 61–62. ISBN 978-0-12-415993-8.

83. Gustafson, John L. (May). Reevaluating Amdahl's Law // Communications of the ACM. 31 (5): 1988. P. 532–533. CiteSeerX 10.1.1.509.6892. doi:10.1145/42411.42415. S2CID 33937392.

84. Snyder, Lawrence. Type Architectures, Shared Memory, and The Corollary of Modest Potential // Annu. Rev. Comput. Sci. 1: 1986. P. 289–317. doi:10.1146/annurev.cs.01.060186.001445.

85. Hill, Mark D.; Marty, Michael R. Amdahl's Law in the Multicore Era // IEEE Computer. 41 (7): 2008. P. 33–38. CiteSeerX 10.1.1.221.8635. doi:10.1109/MC.2008.209. UW CS-TR-2007-1593.

86. Dong Hyuk Woo; Hsien-Hsin S. Lee. Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era // IEEE Computer. 41 (12): 2008. P. 24–31. CiteSeerX 10.1.1.156.3907. doi:10.1109/mc.2008.494. S2CID 6136462.

87. Rafiev, Ashur; Al-Hayanni, Mohammed A. N.; Xia, Fei; Shafik, Rishad; Romanovsky, Alexander; Yakovlev, Alex. Speedup and Power Scaling Models for Heterogeneous Many-Core Systems // IEEE Transactions on Multi-Scale Computing Systems. 4 (3): 2018. P. 436–449. doi:10.1109/TMSCS.2018.2791531. ISSN 2332-7766. S2CID 52287374.

88. Al-hayanni, Mohammed A. Noaman; Xia, Fei; Rafiev, Ashur; Romanovsky, Alexander; Shafik, Rishad; Yakovlev, Alex. Amdahl's law in the context of heterogeneous many-core systems – a survey // IET Computers & Digital Techniques. 14 (4): 2020. P. 133–148. doi:10.1049/iet-cdt.2018.5220. ISSN 1751-8601. S2CID 214415079.

89. Оптимальная архитектура кластера Kubernetes: лучшие практики и ограничения. URL: https://kubernetesplatform.ru/article1. (Доступ свободный, дата обращения 02.06.2025).

90. Зотов А. В. HPC 2025: Революция вычислений от кластеров к платформам – как суперкомпьютеры становятся сервисом. URL: https://k2.tech/. (Доступ свободный, дата обращения 02.06.2025).

91. Серверы для научных вычислений: HPC решения.

URL: https://www.ittelo.ru/news/servery-dlya-nauchnykh-vychisleniy-hpc-resheniya/?ysclid=me9rcv85w2817574846. (Доступ свободный, дата обращения 02.06.2025).

92. Серверы для научных вычислений: HPC решения. URL: https://www.ittelo.ru/news/servery-dlya-nauchnykh-vychisleniy-hpc-resheniya/?ysclid=me7bsaglb7326706833. (Доступ свободный, дата обращения 02.06.2025).

93. Поздеев А., Халуторных Е. SaaS, PaaS, IaaS: в чем разница? 1 февраля 2024. URL: https://kontur.ru/talk/spravka/48573-saas_paas_iaas_v_chem_raznica. (Доступ свободный, дата обращения 02.06.2025).

94. Федосин, М. Е. Технология поддержки учета ресурсов в виртуальных информационно-вычислительных лабораториях / М. Е. Федосин // Фундаментальные исследования. – 2013. – № 1 (ч. 2). – С. 433–438.

95. McLennan M., Kennell R., HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering // Computing in Science and Engineering. – 2010. – № 12(2). – P. 48–52.

96. Klimeck G., McLennan M., Brophy S.P., Adams G.B., Lundstrom M.S. NanoHUB.org: Advancing Education and Research in Nanotechnology // Computing in Science and Engineering. – № 10(5), 2008. – P. 17–23.

97. Федосин М. Е. Разработка программно-аппаратного комплекса для предоставления доступа к высокопроизводительному программному обеспечению в концепции облачных вычислений // Международный журнал прикладных и фундаментальных исследований. – 2013. – № 6. – С. 110–111.

98. Федосин М.Е. Создание виртуальных информационно-вычислительных лабораторий на основе технологической платформы UniHUB // Системы управления и информационные технологии. – 2012.– № 3.1(49). – С. 175–178.

99. Зинкин С. А., Федосин М. Е., Савкина А. В. Описание запуска вычислительных заданий в веб центрах на основе логико-алгебраических спецификаций // Научно-технический вестник Поволжья. – 2013. – № 4. – С. 154–159.

100. Зинкин С. А., Федосин М. Е., Федосин А. С. Разработка программной и аппаратной архитектуры веб-центра с использованием логико-алгебраических моделей представления знаний // Научно технический вестник Поволжья. – 2013. – № 6. – С. 289–293.

101. Джафар М.С., Зинкин С. А., Карамышева Н. С. Организация функционирования распределенных вычислительных систем с переменной архи-

тектурой в виде облачного сервиса, формируемого по запросу клиента (реализация изменяемой системной архитектуры) // XXI век: итоги прошлого и проблемы настоящего *плюс*. – 2018. – Т. 7, № 4 (44). – С. 54–60.

102. Джафар М.С., Зинкин С. А., Карамышева Н. С. Организация функционирования распределенных вычислительных систем с переменной архитектурой в виде облачного сервиса, формируемого по запросу клиента (концептуальные графы распределенных алгоритмов) // XXI век: итоги прошлого и проблемы настоящего *плюс*. – 2018. – Т. 7, № 4 (44). – С. 136–146.

103. Суперкомпьютерный кластер: шаг вперед для науки и промышленности. URL: https://k2neurotech.cnews.ru/articles/2024-12-02_superkompyuternyj_klaster/. (Доступ свободный, дата обращения 02.06.2025).

104. Суперкомпьютерные решения от «К2 НейроТех» и «Гравитон» URL: https://bytemag.ru/superkompyuternye-resheniya-ot-k2-nejroteh-i-graviton-36411/. (Доступ свободный, дата обращения 02.06.2025).

105. Высокоскоростная коммуникационная сеть Ангара от НИЦЭВТ. URL:https://www.karma-group.ru/solutions/postroenie-it-infrostruktury/vysokoskorostnaya-set-angara/. (Доступ свободный, дата обращения 02.06.2025).

106. К2Тех развернул суперкомпьютер на базе Новосибирского госуниверситета. URL: https://bytemag.ru/k2teh-razvernul-superkompyuter-na-baze-novosibirskogo-gosuniversiteta-27709/. (Доступ свободный, дата обращения 02.06.2025).

107. How many Kubernetes nodes should be in a cluster? Feb 15, 2024. URL: https://www.thetechplatform.com/post/how-many-kubernetes-nodes-should-be-in-a-cluster/. (Доступ свободный, дата обращения 02.06.2025).

108. Разработка, управляемая моделями. URL: https://ru.wikipedia.org/w/index.php?tile=Разработка,_управляемая_моделями&stable=1. (Доступ свободный, дата обращения 02.06.2025).

109. Walker R. Event-Driven Architecture: The Definitive Guide. Aug 22, 2023. URL: https://appmaster.io/blog/event-driven-architecture. (Доступ свободный, дата обращения 02.06.2025).

110. Предметно-ориентированное проектирование. URL: https://ru.wikipedia.org/wiki/Предметно-ориентированное_проектирование. (Доступ свободный, дата обращения 02.06.2025).

111. Прицкер А. Введение в имитационное моделирование и язык СЛАМ II. – М.: Мир, 1987. – 646 с.

112. Теория и практика структурного тестирования программных систем // Л.С. Ломакина, А.С. Базин, А.Н. Вигура, А.В. Киселев. – Воронеж: Научная книга, 2013. – 220 с.

113. Кулагин В.П. Формирование информационных ресурсов на основе параллельных вычислений // Перспективы науки и образования. – 2013. – № 6. – С. 26-31.

114. Котов В.Е. Сети Петри. – М.: Наука, 1984. – 160 с.

115. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. – СПб.: Изд-во Санкт-Петербургского государственного университета информационных технологий, механики и оптики, 2008. – 167 с.

116. Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий / Ющенко Е. Л., Цейтлин Г. Е., Грицай В. П., Терзян Т. К. – М.: Финансы и статистика, 1989. – 208 с.

117. Волчихин В.И., Зинкин С.А. Логико-алгебраические модели и методы в проектировании функциональной архитектуры распределенных систем хранения и обработки данных // Известия ВУЗов. Поволжский регион. Технические науки. 2012. – № 2. – С. 3-16.

118. Зинкин С.А. Элементы новой объектно-ориентированной технологии для моделирования и реализации систем и сетей хранения и обработки данных // Информационные технологии. – 2008. – № 10. – С. 20-27.

119. Микроэкономика.
URL: https://ru.wikipedia.org/wiki/Микроэкономика. (Доступ свободный, дата обращения 02.06.2025).

120. Макроэкономика.
URL: https://ru.wikipedia.org/wiki/Макроэкономика. (Доступ свободный, дата обращения 02.06.2025).

121. Байцер Б. Микроанализ производительности вычислительных систем. – М.: Радио и связь, 1983. – 360 с.

122. Лекции по дискретной математике / Капитонова Ю.В., Кривой С.Л., Летичевский А.А., Луцкий Г.М. – СПб.: БХВ-Петербург, 2004. – 624 с.

123. Зинкин С.А. Интеллектуализация и интеграция систем и сетей хранения и обработки данных. Пенза: Изд-во ПГУ. – 2023. – 416 с.

124. Кудрявцев Е. М. GPSS World. Основы имитационного моделирования различных систем. М.: ДМК Пресс, 2004. – 320 с.

125. Forouzan B. A. TCP/IP Protocol Suite. McGraw-Hill, 2009. – 1024 p.

126. Standard Group MAC Addresses. A Tutorial Guide. URL: http://standards.ieee.org/regauth/groupmac/tutorial.html. (Доступ свободный, дата обращения 02.06.2025).

127. Гуренко В. В. Введение в теорию автоматов. М.: МГТУ им. Н. Э. Баумана, 2013.

128. Ульман Д., Хопкрофт Д., Мотвани Р. Введение в теорию автоматов, языков и вычислений. Изд-во «Вильямс», 2008. 528 с.

129. Баранов С. И. Синтез микропрограммных автоматов (граф-схемы и автоматы). Ленинград: Изд-во «Энергия». Ленинградское отделение, 1979. 231 с.

130. Вашкевич Н. П. Синтез микропрограммных управляющих автоматов. Пенза, Пенз. политехн. ин-т, 1990. 115 с.

131. Вашкевич Н. П., Сибиряков М. А. Формальные автоматные модели алгоритмов обработки кэшируемой информации // Современные наукоемкие технологии. – М.: Академия Естествознания, 2016. № 8. Ч. 2. С. 205-213.

132. Лазарев В. Г., Пийль Е. И., Турута Е. Н. Построение программируемых управляющих устройств. М.: Энергоатомиздат, 1984. 264 с.

133. Методы параллельного микропрограммирования / П. А. Анишев, С. М. Ачасова, О. Л. Бандман и др.; под ред. О. Л. Бандман. Новосибирск: Наука. Сибирское отделение, 1981. 180 с.

134. Юдицкий С. А., Магергут В. З. Логическое управление дискретными процессами. Модели, анализ, синтез. М.: Машиностроение, 1987. 176 с.

135. Girault A., Lee E. A. Hierarchical finite state machines with multiple concurrency models // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, July 1999, 18(6). P. 742–760.

136. Stefansson E., Johansson K. H. Hierarchical finite state machines for efficient optimal planning in large-scale systems // European Control Conference (ECC). June, Bucharest, Romania, 2023. P. 1-8.
DOI:10.23919/ECC57647.2023.10178139.

137. Alur R., Yannakakis M. Model checking of hierarchical state machines //ACM SIGSOFT Software Engineering Notes, 23(6), 1998. P. 175–188.

138. Болотова Л. С. Системы искусственного интеллекта. Модели и технологии, основанные на знаниях. М.: Изд-во «Финансы и статистика». 2012. 664 с.

139. Тейз, А. Логический подход к искусственному интеллекту: от классической логики к логическому программированию / А. Тейз, П. Грибомон, Ж. Луи и др. – М.: Мир, 1990. 429 с.

140. Представление и использование знаний. Под ред. Х. Уэно, М. Исидзука. М.: Мир, 1989. 220 с.

141. Питерсон Дж. Теория сетей Петри и моделирование систем. М.: Мир, 1984. 368 с.

142. Алгеброалгоритмические модели и методы параллельного программирования / Андон Ф. И., Дорошенко А. Е., Цейтлин Г. Е., Яценко Е. А. Академпериодика, 2007. 634 с.

143. Gurevich Y. Abstract State Machines: An Overview of the Project // Foundations of Information and Knowledge Systems. Lect. Notes Comput. Sci., vol. 2942, 2004. P. 6-13.

144. Мейер Д. Теория реляционных баз данных. М.: Мир, 1987. 608 с.

145. Iordache M. V., Antsaklis P. J. Supervisory Control of Concurrent Systems. A Petri Net Structural Approach. – Boston: Birkhauser, 2006. 281 p.

146. Levis T. G. Foundation of parallel programming: machine-independent approach. IEEE Computer Society Press, 1994. 282 p.

147. Ачасова С. М., Бандман О. Л. Корректность параллельных вычислительных процессов. Новосибирск: Наука, 1990. 253 с.

148. Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В. Моделирование параллельных процессов. Сети Петри. Санкт-Петербург: Профессиональная литература, АйТи-Подготовка, 2014. 400 с.

149. Кластер (группа компьютеров). URL: https://ru.wikipedia.org/wiki/Кластер_(группа_компьютеров)/.

150. Марлин Дж. Шесть типичных проблем отказоустойчивых кластеров // Открытые системы. № 5. 2014.

151. Отказоустойчивый кластер. URL: https://ru.wikipedia.org/wiki/Отказоустойчивый_кластер..

152. Koren I., Krishna C. M. Fault-Tolerant Systems. Publisher Morgan Kaufmann. 2020. 416 p.

153. How to deploy a fault tolerant cluster with continuous or high availability. URL: https://www.howtoforge.com/tutorial/how-to-deploy-a-fault-tolerant-cluster-with-continuous-or-high-availability/.

154. Peterson B. vROps Cluster Architecture: Stand-Alone, HA, and CA. May 7, 2022. URL: https://www.brockpeterson.com/post/vrops-cluster-architecture-ha-and-ca/.

155. Flinders M., Smalley I. What is high availability? July 29, 2024. URL: https://www.ibm.com/think/topics/high-availability/.

156. Failure Detection and Recovery in Distributed Systems. August 05, 2024. URL: https://www.geeksforgeeks.org/computer-networks/failure-detection-and-recovery-in-distributed-systems/.

157. Awati R. Heartbeat (computing). Published: Feb 14, 2023. URL: https://www.techtarget.com/searchdatacenter/definition/Heartbeat.

158. Algorithm-Based Fault Tolerance Techniques. URL: https://www.nla-fet.eu/wp-content/uploads/2016/01/NLAFET-D6.6-171031.pdf. NLAFET Consortium, 2015–2018. 28 p.

159. Chinnaiah M., Niranjan N. Fault tolerant software systems using software configurations for cloud computing // Journal Cloud Comp. Vol. 7, No. 1 2018. P. 1-17. https://doi.org/10.1186/s13677-018-0104-9.

160. Черкесов Г.Н. Надежность аппаратно-программных комплексов. – Питер, 2005. – 479 с.

161. MAC Ethernet Medium access control. URL: https://en.wikipedia.org/wiki/Medium_access_control/.

162. InfiniBand. URL: https://ru.wikipedia.org/wiki/InfiniBand/.

163. Феллер В. Введение в теорию вероятностей и ее приложения. В 2-х томах. Т. 1. М: Мир, 1984. 528 с.

164. **Петушков Г.В.**, Сигов А.С. Формализация и реализация логико-вероятностных и логико-алгебраических операционных моделей функциональной архитектуры кластерных вычислительных систем // Известия высших учебных заведений. Поволжский регион. Технические науки. – № 3. – 2025. – С. 26-62.

165. **Петушков Г.В.** Организация и исследование кластерных вычислительных систем с функциональной архитектурой, определяемой исполнимыми моделями. Автоматные исполнимые модели обработки информации // Russian Technological Journal. – Т. 13. – № 6. – 2025. – С. 7-24.

166. **Петушков Г.В.**, Сигов А.С. Организация кластерных вычислительных систем с функциональной архитектурой, определяемой исполнимыми моделями. Логико-алгебраические исполнимые модели обработки информации // XXI век: итоги прошлого и проблемы настоящего плюс. – Т. 14. – № 3(71). – 2025. – С. 10-22.

167. **Петушков Г.В.**, Сигов А.С. Функциональная организация вычислительных кластеров с повышенной отказоустойчивостью // XXI век: итоги прошлого и проблемы настоящего плюс. – 2025. – Т. 14. – № 3(71). – С. 55-64.

168. **Петушков Г.В.**, Сигов А.С. Анализ и выбор структуры многопроцессорной вычислительной системы по критерию быстродействия // Russian Technological Journal. – Т. 12. № 6. – 2024. – С. 20-25.

169. **Петушков Г.В.**, Сигов А.С. Технико-экономический анализ серверов как вычислительных модулей вычислительных систем класса WSC // Russian Technological Journal. – Т. 13. – № 1. – 2025. – С. 49-58.

170. **Петушков Г.В.** Методика анализа надежности и мониторинга функционирования вычислительных систем. Известия Юго-Западного государственного университета. – Т. 29. – № 1. – 2025. – С. 155-172.

171. **Петушков Г. В.** Оценка производительности вычислительной системы. Известия Юго-Западного государственного университета. – Т. 29. – № 2. – 2025. – С. 201-220.

172. Коваленко С. М., Асадова Ю. С., **Петушков Г. В.** Проектирование вычислительных комплексов: учебное пособие. Москва: РТУ МИРЭА, 2023. ISBN 978-5-7339-1749-8. Текст: электронный // Лань: электронно-библиотечная система. URL: https://e.lanbook.com/book/368978.

173. Коваленко С.М., Платонова О.В., **Петушков Г.В.** Развитие информационных технологий – ближайшие перспективы // Научно-технический вестник Поволжья. – 2018. – № 10. – С. 172-174.

174. Коваленко С.М. Романов А.М., **Петушков Г.В.** Перспективы развития аппаратной платформы компьютеров // Современные наукоемкие технологии. – 2018. – № 12. – С. 67-70.

175. Kovalenko S.M., **Petushkov G.V.**, Platonova O.V. Structure Selection Technique of Multi-Processor Computing Systems // International Journal of Engineering and Technology. – Vol. 7. – Issue 4. – 2018. – P. 29-31.

176. Kovalenko S. M., **Petushkov G. V.**, Platonova O. V. A Method of Estimating the Reliability of Highly Reliable Redundant Systems Based on Continuous Models // International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies. Vol. 10, Issue 7. 2019. P. 913-916.

177. Прогнозирование надежности по отношению к отказам программного обеспечения для сложных программно-аппаратных систем. / С.М. Коваленко, **Г.В. Петушков**, О.В. Платонова, М.М. Расулов // Сборник трудов IX международной научной конференции «ИТ-Стандарт-2019». Электронная версия. Изд-во РТУ-МИРЭА. – 2019. – С. 212-215.

ПРИЛОЖЕНИЕ А

А1. Реализации непосредственно исполнимых автоматных моделей последовательно-параллельного приложения на языке C++ (примеры реализаций для иллюстрации п. 3.1 и п. 3.4 3-й главы)

На рис. А1.1 и А1.2 представлены графы переходов состояний для модели последовательно-параллельного приложения из п. 3.1 и п. 3.4 3-й главы, реализуемого вычислительным кластером.

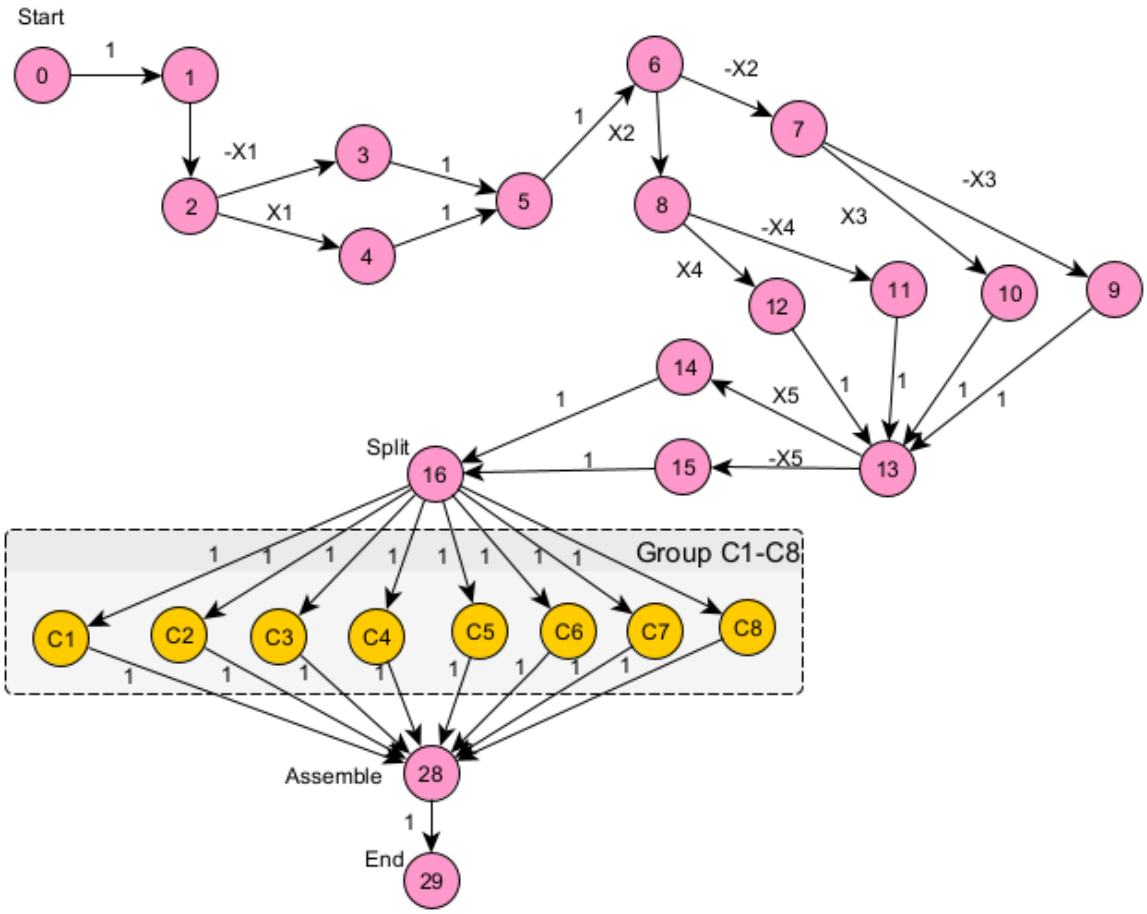


Рис. А1.1. Граф переходов состояний автоматной модели последовательно-параллельного приложения со структурированными состояниями

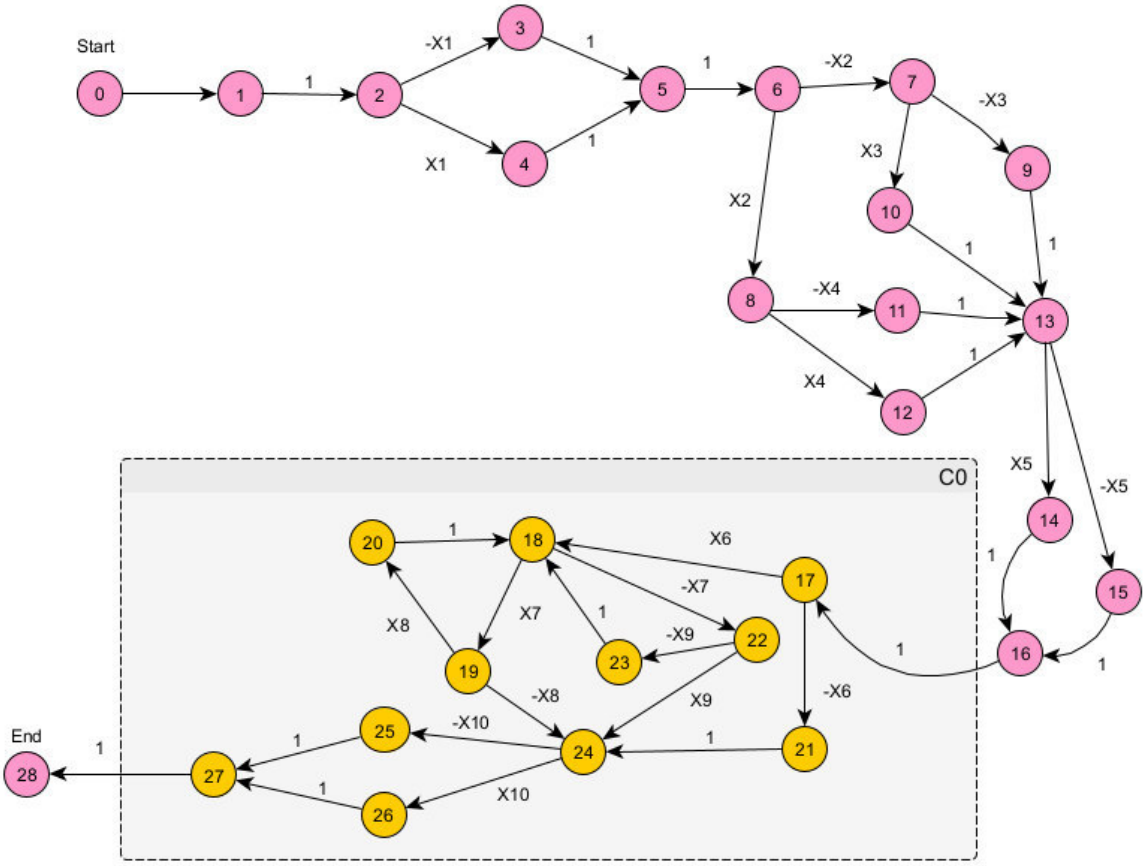


Рис. А1.2. Граф переходов состояний автоматной модели последовательно-параллельного приложения; выделен участок (раскрытое структурированное состояние) C_0, предназначенный для последующего клонирования при переходе к режиму SPMD (*Single Program, Multiple Data*)

Реализация на языке C++ исполнимой сетевой автоматной модели для вычислительного кластера

// Часть 1 последовательного участка (рис. А1.1)

```
#include <iostream>
#include <string>
using namespace std;
// Функция для ввода логических переменных
bool input(string s) {
    bool x;
    cout << "Введите " << s << " (0 или 1): ";
    cin >> x;
    return x;
}
int main() {
    // Ввод логических условий
    bool X1 = input("X1");
    bool X2 = input("X2");
    bool X3 = input("X3");
    bool X4 = input("X4");
    bool X5 = input("X5");
    bool X6 = input("X6");
    bool X7 = input("X7");
    bool X8 = input("X8");
    bool X9 = input("X9");
    // Флаги окончания
    bool End = false;
    // Флаги блоков
    bool A1=0, A2=0, A3=0, A4=0, A5=0,
    A6=0, A7=0, A8=0, A9=0;
    bool A10=0, A11=0, A12=0, A13=0,
    A14=0, A15=0, A16=0, A17=0;
    bool A18=0, A19=0, A20=0, A21=0,
    A22=0, A23=0, A24=0, A25=0, A26=0, A27=0;
    while (!End) {
        cout << "\n=== Новый проход ===\n";
        A1 = 1;
        cout << "Выполнен A1\n";
        A2 = A1;
        cout << "Выполнен A2\n";
        if (X1) {
            A3 = A2;
            cout << "→ X1=1: Выполнен A3\n";
        } else {
            A4 = A2;
            cout << "→ X1=0: Выполнен A4\n";
        }
        A5 = A3 || A4;
        cout << "Выполнен A5\n";
```

// Часть 2 последовательного участка (рис. А1.1)

```
    A6 = A5;
        cout << "Выполнен A6\n";
        if (X2) {
            A7 = A6;
            cout << "→ X2=1: Выполнен A7\n";
            if (X3) {
                A9 = A7;
                cout << "→ X3=1: Выполнен A9\n";
            } else {
                A10 = A7;
                cout << "→ X3=0: Выполнен A10\n";
            }
        } else {
            A8 = A6;
            cout << "→ X2=0: Выполнен A8\n";
            if (X4) {
                A11 = A8;
                cout << "→ X4=1: Выполнен A11\n";
            } else {
                A12 = A8;
                cout << "→ X4=0: Выполнен A12\n";
            }
        }
        A13 = A9 || A10 || A11 || A12;
        cout << "Выполнен A13\n";
        if (X5) {
            A14 = A13;
            cout << "→ X5=1: Выполнен A14\n";
        } else {
            A15 = A13;
            cout << "→ X5=0: Выполнен A15\n";
        }
        A16 = A14 || A15;
        cout << "Выполнен A16\n";
    A17 = A16;
    cout << "Выполнен A17\n";
    if (X6) {
        A18 = A17;
```

// Часть 3: клон распараллеленного участка C0 (рис. A1.2) автоматного приложения для вычислительного кластера

```
// Часть 3
  A21 = A17;
        cout << "→ X6=1: Выполнены A18 и A21\n";
        if (X7) {
           A19 = A18;
           cout << "→ X7=1: Выполнен A19\n";
           if (X8) {
              A20 = A19;
              cout << "→ X8=1: Выполнен A20\n";
           }
        }
        A22 = A21;
        cout << "Выполнен A22\n";
        if (X9) {
           A23 = A22;
           cout << "→ X9=1: Выполнен A23\n";
        }
        A24 = A20 || A23;
        cout << "Выполнен A24\n";
        A25 = A24;
        A26 = A24;
        cout << "Выполнены A25 и A26\n";
        A27 = A25 || A26;
        cout << "Выполнен A27\n";
        End = true;
     } else {
        cout << "→ X6=0: Возврат к началу цикла\n";
     }
  }
  // Финал
  cout << "\n=== Завершено ===\n";
  cout << "Выполнен A27 = " << A27 << endl;
  return 0;
}
```

Сетевая автоматная модель с распараллеленным участком для кластера на языке Ассемблера микроконтроллера Intel 8051 (иллюстрация примера на рис. A1.1, рис. A1.2)

```
 ORG 0000H
 ; Инициализация
 START:
MOV R0, #0    ; переменная ;для X1
MOV R1, #1    ; переменная ;для X2
MOV R2, #1    ; переменная ;для X3
MOV R3, #0    ; X4
MOV R4, #1    ; X5
MOV R5, #0    ; X6
MOV R6, #1    ; X7
MOV R7, #1    ; X8
MOV A,  #1 ; X9/X10 через
;аккумулятор
    ; A1
    SETB P1.0
    ; A2
    CLR P1.1
    ; X1
    MOV A, R0
    CJNE A, #0, X1_TRUE
 X1_FALSE:
    ; A4
    SETB P1.2
    SJMP AFTER_X1
 X1_TRUE:
    ; A3
    CLR P1.2
 AFTER_X1:
    ; A5
    SETB P1.3
    ; A6
    CLR P1.4
    ; X2
    MOV A, R1
    CJNE A, #0, X2_TRUE
 X2_FALSE:
    ; A8
    SETB P1.5
    SJMP AFTER_X2
 X2_TRUE:
    ; A7
    CLR P1.5
 AFTER_X2:
    ; X3
    MOV A, R2
    CJNE A, #0, X3_TRUE
 X3_FALSE:
    ; A9
    SETB P1.6
    SJMP AFTER_X3
 X3_TRUE:
    ; A10
    CLR P1.6
 AFTER_X3:
    ; X4
    MOV A, R3
    CJNE A, #0, X4_TRUE
 X4_FALSE:
    ; A11
    SETB P1.7
    SJMP AFTER_X4
 X4_TRUE:
    ; A12
    CLR P1.7
 AFTER_X4:
    ; A13
    MOV P2, #55H
    ; X5
    MOV A, R4
    CJNE A, #0, X5_TRUE
 X5_FALSE:
    ; A15
    MOV P2, #0FFH
    SJMP AFTER_X5
 X5_TRUE:
    ; A14
    MOV P2, #00H
 AFTER_X5:
    ; A16
    ACALL DELAY
 ; Parallel part
   ; A17
    ACALL DELAY
    ; X6
    MOV A, R5
    CJNE A, #0, X6_TRUE
 X6_FALSE:
    ; A21
    INC P2
    SJMP A18_ENTRY
 X6_TRUE:
    ; A18
    DEC P2
 A18_ENTRY:
    ; X7
    MOV A, R6
    CJNE A, #0, X7_TRUE
 X7_FALSE:
    ; A22
    CPL P1.0
    ; X9
    CJNE A, #0, X9_TRUE
 X9_FALSE:
    ; A24
  MOV P3, #0AAH
    SJMP X10_CHECK
 X9_TRUE:
    ; A23
    MOV P3, #55H
    SJMP X10_CHECK
 X7_TRUE:
    ; A19
    CPL P1.1
    ; X8
    MOV A, R7
    CJNE A, #0, X8_TRUE
 X8_FALSE:
    ; A24
    MOV P3, #0AAH
    SJMP X10_CHECK
 X8_TRUE:
    ; A20
    CPL P1.2
    ; A24
    MOV P3, #0AAH
 ; X10
 X10_CHECK:
    CJNE A, #0, X10_TRUE
 X10_FALSE:
    ; A26
    CPL P3.7
    SJMP END
 X10_TRUE:
    ; A25
    CPL P3.6
 ; A27
 END:
    SJMP END
 ; Задержка
 DELAY:
    MOV R0, #0FFH
 DL1: MOV R1, #0FFH
 DL2: DJNZ R1, DL2
     DJNZ R0, DL1
     RET
 END
```

А2. Реализация кластерного распределенного многомодульного приложения, работающего в параллельно-конвейерном режиме NETWORK-MPMD (пример реализации для п. 3.10 и п. 3.11 3-й главы)

На рис. А2.1 представлен граф переходов состояний для модели параллельно-конвейерного приложения **для** п. 3.10 и п. 3.11 3-й главы, реализуемого вычислительным кластером. Прилагается краткое описание реализации и протокол работы управляющей программы для вычислительного кластера.

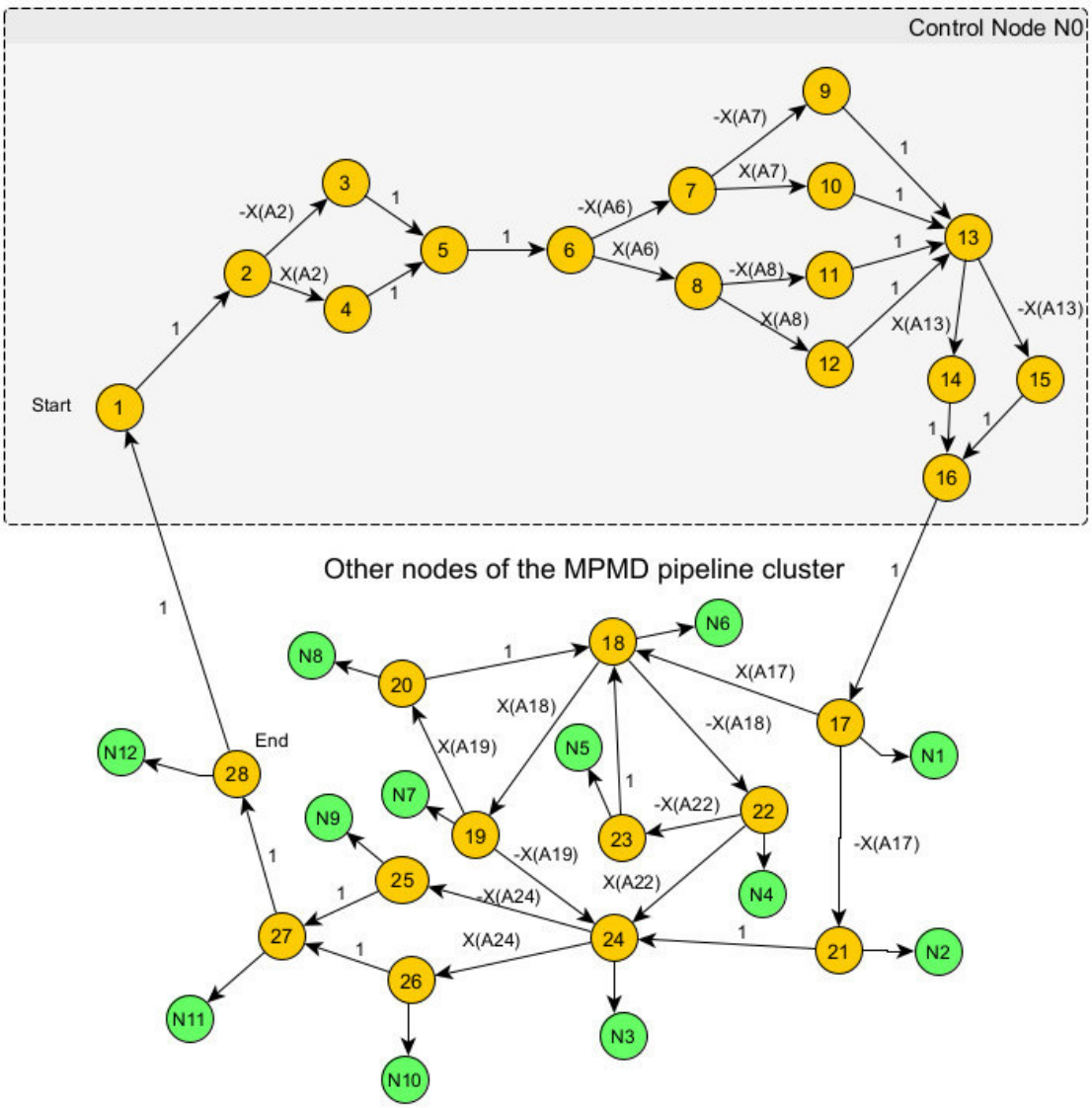


Рис. А2.1. Граф переходов состояний вычислительного кластера, соответствующий работе распределенного многомодульного приложения в параллельно-последовательном режиме NETWORK-MPMD на нижней части рисунка; верхняя часть в рамке соответствует последовательному подготовительному этапу, выполняемому управляющим узлом

Реализация и протокол работы управляющей программы для вычислительного кластера, реализованного на компьютерной сети и работающего в параллельно-конвейерном режиме NETWORK-MPMD

Полная версия распределенной программы позволяет формировать программы в соответствии со спецификациями на языке логико-алгебраических операционных выражений (ЛАОВ), рассмотренных в п. 3.10 и п. 3.11 3-й главы. Далее дано описание полнофункциональной управляющей программы, иллюстрирующей кольцевой фрагмент N_4, N_5, N_6 на рис. А2.1.

Программа реализует произвольно выбранную сеть узлов для пересылки сообщений между несколькими компьютерами. Каждый узел может принимать сообщения от предыдущего узла и передавать их следующему в цепочке, пока сообщение не вернется к отправителю.

Программа каждого рабочего узла состоит из двух компонентов:

• **Серверная часть** прослушивает входящие соединения и обрабатывает полученные сообщения.

• **Клиентская часть** отправляет сообщения следующему узлу в цепочке рабочих узлов кластера.

Формат сообщений.

Сообщения передаются в следующем формате:

IP:PORT|НОМЕР_ПЕРЕХОДА|ТЕКСТ_СООБЩЕНИЯ;

• IP:PORT - адрес и порт исходного отправителя;

• НОМЕР_ПЕРЕХОДА - счетчик количества пройденных узлов;

• ТЕКСТ_СООБЩЕНИЯ - текст передаваемого сообщения.

Порядок исполнения программы.

1. Инициализация

• Пользователь вводит адрес для прослушивания и адрес следующего узла.

2. Запуск потоков

• Серверный поток начинает прослушивать указанный адрес и порт.

• Поток ввода ожидает ввод сообщений от пользователя или от предыдущего рабочего узла кластера.

3. Обработка сообщений

При получении сообщения:

3.1. Составляется новое сообщение для отправки.

3.2. Увеличивается счетчик переходов.

3.3. Выводится информация о полученном сообщении.

3.4. Проверяется, является ли текущий узел конечным получателем:

• если «да» – выводится уведомление о доставке, пересылка прекращается;

• если «нет» – сообщение пересылается следующему узлу.

Пример работы программы

При запуске программы требуется ввести адрес для прослушивания и адрес следующего узла.

```
Введите адрес для прослушивания: 192.168.137.1:2015
Введите адрес следующего узла: 192.168.137.183:2015
```

Настройка узла

Далее можно отправлять сообщения.

Первый узел (192.168.137.112) отправляет сообщение второму узлу (192.168.137.1).

```
Запуск узла 192.168.137.112:2015 -> следующий 192.168.137.1:2015

>
=== Узел слушает 192.168.137.112:2015 ===
Следующий узел: 192.168.137.1:2015

hi
[#1] -> Отправлено 192.168.137.1:2015 <= "hi"
```

Отправка сообщения

Второй узел (192.168.137.1) отправляет полученное сообщение третьему узлу (192.168.137.183).

```
Запуск узла 192.168.137.1:2015 -> следующий 192.168.137.183:2015

>
=== Узел слушает 192.168.137.1:2015 ===
Следующий узел: 192.168.137.183:2015

[#2] <- Получено сообщение
     Отправитель: 192.168.137.112:2015
     Текст: "hi"
   Пересылаю дальше -> 192.168.137.183:2015
```

Следующий узел пересылает сообщение

Третий узел (192.168.137.183) отправляет полученное сообщение обратно первому узлу (192.168.137.112)

```
Запуск узла 192.168.137.183:2015 -> следующий 192.168.137.112:2015

>
=== Узел слушает 192.168.137.183:2015 ===
Следующий узел: 192.168.137.112:2015

[#3] <- Получено сообщение
     Отправитель: 192.168.137.112:2015
     Текст: "hi"
   Пересылаю дальше -> 192.168.137.112:2015
```

Следующий узел пересылает сообщение

Далее при повторном получении этого сообщения первый узел выводит уведомление, и пересылка останавливается.

```
Запуск узла 192.168.137.112:2015 -> следующий 192.168.137.1:2015

>
=== Узел слушает 192.168.137.112:2015 ===
Следующий узел: 192.168.137.1:2015

hi
[#1] -> Отправлено 192.168.137.1:2015 <= "hi"
> [#4] <- Получено сообщение
     Отправитель: 192.168.137.112:2015
     Текст: "hi"
   (+) Доставлено конечному получателю (192.168.137.112:2015)
```

Отправитель повторно получает сообщение

Данная реализация вычислительного кластера на компьютерной сети позволяет разрабатывать на ее основе широкий набор распределенных приложений по заданию пользователя, в том числе работающих в параллельно-конвейерном, последовательно-конвейерном и в последовательно-параллельном режимах.

ПРИЛОЖЕНИЕ Б

Акты о внедрении результатов диссертационной работы

ООО "ПРАЙМ ГРУП" ул. Успенская, д. 3, пом/эт/оф I/4/415, Московская область, г. Красногорск, 143409
тел./факс: +7 (499) 579-7701/02, www.primegroup.ru

Утверждаю
Генеральный директор
ООО «ПРАЙМ ГРУП»
_____ /М.И. Лукичев/
«___» _____ 2019 г.

Акт
о внедрении результатов диссертационной работы
Петушкова Г.В.

Настоящим актом подтверждаем, что результаты диссертационных исследований Петушкова Григория Валерьевича были применены в работе ООО «ПРАЙМ ГРУП» в ходе разработки центров обработки данных и системно-технических решений для предприятий топливно-энергетического комплекса.

Использование предложенной в диссертации методики выбора структуры кластерной вычислительной системы и оценок надежности вычислительной системы позволило существенно повысить надежность разработанных решений, снизить количество аварийных отказов серверов и обеспечить стабильность работы информационно-управляющих систем предприятий.

Начальник управления системных проектов и внедрений Департамента промышленности, к.т.н. _____ / А.Б. Купцов /

АКЦИОНЕРНОЕ ОБЩЕСТВО
КОНЦЕРН МОРИНФОРМСИСТЕМА-АГАТ
КОРПОРАЦИЯ МОРСКОГО ПРИБОРОСТРОЕНИ
Адрес: шоссе Энтузиастов, 29, Москва, Россия, 105275
Тел.: +7 (495) 673-4063, Факс: +7 (495) 673-4130
ИНН 7720544208, КПП 772001001
ОКПО 17597462, ОКВЭД 72.19
Интернет-сайт: www.concern-agat.ru
Электронная почта: info@concern-agat.ru

УТВЕРЖДАЮ
Директор производственно-
технического комплекса
_____ А.Х. Бекишев
«29» 09 2025 г.

АКТ
о внедрении результатов диссертационной работы Петушкова Г.В.

Настоящим актом подтверждаем, что результаты диссертационного исследования Петушкова Григория Валерьевича были применены в работе АО «Концерн «Моринсис-Агат» в ходе проектирования и изготовления систем жидкостного охлаждения центров обработки данных (ЦОД) с большим тепловыделением.

Использование, предложенной в диссертации методики, выбора структуры вычислительной системы и оценок надежности вычислительных систем (ВС) позволило существенно повысить надежность разработанных решений, снизить количество аварийных отказов ЦОД и обеспечить стабильность работы информационно-управляющих систем. Особенностями кластерных ВС является модульная структура, причем модули выполняются на типовом унифицированном, серийно выпускаемом оборудовании (микропроцессорах, планках памяти, адаптерах сетей) и объединяются в систему стандартным сетевым оборудованием, что делает ВС относительно дешевыми. Для таких систем важным качеством является достижение высокой производительности при сохранении достойных параметров надежности работы.

Это дало возможность создать ВС, которая позволяет использование большого массива вычислительных модулей или серверов стандартной унифицированной конфигурации числом сотни или тысячи штук. Для таких ВС важнейшей характеристикой является надежность работы в течение длительного

времени в режиме 24×7. Это позволяет отводить не менее 10 кВт мощности тепловыделения.

Использование рассматриваемой методики в ЦОД позволило повысить производительность ВС, понизить температуру тепловыделения, обеспечив надежность ВС в целом и выполнить требования технического задания.

Сухов Владимир Васильевич
доктор технических наук,
доцент
АО «Концерн «Моринсис-Агат»
«29» 09 2025 г.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт искусственного интеллекта

Кафедра промышленной информатики

«УТВЕРЖДАЮ»
Первый проректор РТУ МИРЭА
_____ Н.И. Прокопов
«1» сентября 2025 г.

АКТ

о внедрении результатов кандидатской диссертационной работы

Петушкова Григория Валерьевича

Комиссия в составе: председатель: зав. кафедрой промышленной информатики доцент, к.т.н., Холопов В.А.; члены комиссии: ученый секретарь, старший преподаватель Володина. А.М., доцент, к.т.н., доцент Каширская Е.Н., доцент, к.т.н., доцент Курнасов Е.В., доцент, к.т.н., доцент Рылов С.А., доцент, к.т.н., доцент Чижиков В.И.; составили настоящий акт о том, что следующие научные и практические результаты диссертационной работы Петушкова Г.В. внедрены в учебный процесс РТУ МИРЭА на кафедре промышленной информатики:

1. Разработанные новые исполнимые модели и методы совершенствования вычислительных систем кластерного типа, определяющие их функциональную архитектуру, масштабируемость, основанные на анализе событий, происходящих при работе приложений и управляющих программ используются в учебном процессе кафедры вычислительной техники.

2. Разработанные логико-вероятностные и исполнимые логико-алгебраические модели на основе методологического подхода к макро- и микроанализу

функциональной архитектуры вычислительных систем кластерного типа, а также сетевые автоматные, вероятностные автоматные и логико-алгебраические исполнимые модели функционирования вычислительных систем кластерного типа как основы для создания управляющих программ, иллюстрирующих работу вычислительных кластеров применяются в практических работах при обучении студентов по направлению подготовки 09.03.01 Информатика и вычислительная техника.

Предложенные модели и методы используются в рамках преподавания дисциплин «Вычислительные системы реального времени», «Промышленный интернет вещей», «Архитектура вычислительных средств систем управления».

Председатель комиссии:
зав. кафедрой ПИ, к.т.н., доцент Холопов В.А.

Ученый секретарь кафедры ПИ, старший преподаватель Володина А.М.

Члены комиссии: доц., к.т.н., доц. Каширская Е.Н.

доц., к.т.н., доц. Курнасов Е.В.

доц., к.т.н., доц. Рылов С.А.

доц., к.т.н., доц. Чижиков В.И.