Russian Technological Journal.
2025;13(6):7–24

ISSN 2782-3210 (Print)
ISSN 2500-316X (Online)

УДК 004.9:681.3
https://doi.org/10.32362/2500-316X-2025-13-6-7-24
EDN WGZAHH

НАУЧНАЯ СТАТЬЯ

Организация и исследование кластерных вычислительных систем с функциональной архитектурой, определяемой исполнимыми моделями. Автоматные исполнимые модели обработки информации

Г.В. Петушков [@]

МИРЭА – Российский технологический университет, Москва, 119454 Россия
[@] *Автор для переписки, e-mail: petushkov@mirea.ru*

• **Поступила:** 04.08.2025 • **Доработана:** 12.09.2025 • **Принята к опубликованию:** 06.10.2025

Резюме

Цели. Актуальной является задача совершенствования функциональной архитектуры кластерных вычислительных систем за счет внедрения методологий создания программного обеспечения прикладного и промежуточного уровней на основе формализованных спецификаций. Одна из таких методологий основана на использовании автоматных спецификаций программного обеспечения вычислительных систем. Сложность решения задачи вызвана разветвленностью построенных алгоритмов, а также наличием циклических участков. Время выполнения разветвленных участков программы и число проходимых циклов зависят от вида вводимых условий и на практике могут быть определены при помощи детальной имитационной модели и анализа созданной на ее основе управляющей программы. Цель работы – нахождение подходов к определению функциональной архитектуры, которые возможно применять практически на основных уровнях предметной ориентации кластерных вычислительных систем.

Методы. Предлагаемые и использованные методы основаны на концепции организации и исследования вычислительных систем кластерного типа с функциональной архитектурой, определяемой исполнимыми автоматными моделями.

Результаты. Предложены методы построения автоматных и логико-вероятностных моделей кластерных вычислительных систем и создания на этой основе программных средств. Вводится понятие логико-вероятностной модели «темпоральная вероятностная система канонических уравнений», которая позволит получить наглядную формализацию и реализацию автоматных моделей и рабочих программ, характерных для кластерных и других приложений, и в существенной степени сократить число «инкрементных» сложений при перечислении моментов дискретного времени. Главной особенностью новой логико-вероятностной модели является сохранение в ее основе исходной системы канонических уравнений.

Выводы. 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. Показано, что исполнимые автоматные модели могут применяться практически на всех уровнях предметной ориентации кластерных вычислительных систем.

Ключевые слова: вычислительная система кластерного типа, приложение промежуточного уровня, функциональная архитектура, логико-вероятностная модель, логико-алгебраическая модель, режимы обработки запросов, результаты моделирования

Для цитирования: Петушков Г.В. Организация и исследование кластерных вычислительных систем с функциональной архитектурой, определяемой исполнимыми моделями. Автоматные исполнимые модели обработки информации. *Russian Technological Journal*. 2025;13(6):7–24. https://doi.org/10.32362/2500-316X-2025-13-6-7-24, https://www.elibrary.ru/WGZAHH

Прозрачность финансовой деятельности: Автор не имеет финансовой заинтересованности в представленных материалах или методах.

Автор заявляет об отсутствии конфликта интересов.

RESEARCH ARTICLE

Organization and study of cluster computing systems with functional architecture determined by executable models. Automata executable models of information processing

Grigory V. Petushkov [@]

MIREA – Russian Technological University, Moscow, 119454 Russia
[@] *Corresponding author, e-mail: petushkov@mirea.ru*

• **Submitted:** 04.08.2025 • **Revised:** 12.09.2025 • **Accepted:** 06.10.2025

Abstract

Objectives. An urgent task is to improve the functional architecture of cluster computing systems by introducing methodologies for creating software at the applied and intermediate levels based on formalized specifications. One such methodology is based on the use of automatic specifications for computer systems software. The complexity of resolving the problem is caused by the branching of the algorithms built, as well as the presence of cyclic sections. The execution time of the branched sections of the program and the number of cycles run depends on the type of conditions entered. In practice it can be determined using a detailed simulation model and analysis of the control program created on its basis. The aim of the work is to find approaches to the definition of functional architecture which can be applied practically at the main levels of the subject orientation of cluster computing systems.

Methods. The methods proposed and used are based on the concept of organization and research of cluster-type computing systems with a functional architecture as defined by executable automatic models.

Results. The paper proposes methods of constructing automatic and logical-probabilistic models of cluster computing systems and creating software tools based on them. The concept of the logical-probabilistic model “temporal probabilistic system of canonical equations (CES)” is introduced. This enables a visual formalization

to be obtained, as well as implementation of automatic models and work programs typical for cluster and other applications. It also significantly reduced the number of “incremental” additions when enumerating discrete time moments. The main feature of the new logical-probabilistic model is the preservation of the original CES in its basis. **Conclusions.** The work concludes that the choice of the system and functional architecture of a computing cluster should be determined not so much by the peak characteristics of the communication equipment specified by the manufacturer, as by the actual indicators achieved at the level of user applications and cluster usage modes. It is also shown that executable automatic models can be applied at almost all levels of cluster computing systems subject orientation.

Keywords: cluster computing system, intermediate level application, functional architecture, finite automaton models, logical-probabilistic model, logical-algebraic model, query processing modes, simulation results

For citation: Petushkov G.V. Organization and study of cluster computing systems with functional architecture determined by executable models. Automata executable models of information processing. *Russian Technological Journal.* 2025;13(6):7–24. https://doi.org/10.32362/2500-316X-2025-13-6-7-24, https://www.elibrary.ru/WGZAHH

Financial disclosure: The author has no financial or proprietary interest in any material or method mentioned.

The author declares no conflicts of interest.

ВВЕДЕНИЕ

Кластеризация – одно из самых современных направлений в области создания вычислительных систем. Появление кластерных вычислительных систем обусловлено прогрессом в области сетевых технологий, чаще всего локальных. При соединении машин в кластер объединение компьютеров производится при помощи сетевых технологий на базе шинной архитектуры или коммутатора, что обусловило увеличение числа приобретенных или арендованных в качестве облачных сервисов вычислительных кластеров [1]. По прогнозам ряда маркетинговых компаний ожидается, что рынок кластерных вычислений к 2032 г. вырастет до 102.4 млрд долларов[1].

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

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 (Nvidia Corporation, США). Этот кластер теоретически может достичь производительности около 497.9 экзафлопс (497900000 терафлопс), устанавливая новые стандарты в суперкомпьютерной мощности. Целью компании xAI (США) 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: система управления потоком работ; система мониторинга кластера; библиотеки для параллельной работы; средства для управления кластером; глобальное пространство процесса, связывающего воедино все узлы кластера; система управления ресурсами; сетевая (возможно, параллельная) файловая система; сетевые, в т.ч. облачные сервисы, обеспечивающие доступ к кластеру многих

[1] Cluster Computing Market Overview. https://www.marketresearchfuture.com/reports/cluster-computing-market-1746. Дата обращения 02.06.2025. / Accessed June 02, 2025.

[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 100000 Nvidia H100 GPUs on a single fabric. Published July 22, 2024. https://www.tomshardware.com/pc-components/gpus/elon-musk-fires-up-the-most-powerful-ai-training-cluster-in-the-world-uses-100000-nvidia-h100-gpus-on-a-single-fabric. Дата обращения 02.06.2025. / Accessed June 02, 2025.

[3] Полмиллиона GPU за 4 месяца: как Маск строит самый мощный кластер в мире. https://www.braintools.ru/article/18041. Дата обращения 02.06.2025. [Half a million GPUs in four months: how Musk is building the world’s most powerful cluster. https://www.braintools.ru/article/18041. Accessed June 02, 2025 (in Russ.).]

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 [2].

Важный шаг в развитии науки и промышленности связан с разработкой и внедрением суперкомпьютера с AI-способностями «ELBJUWEL» (AI – artificial intelligence, искусственный интеллект)[4]. 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 [3–5].

Следующей проблемой, с которой сталкиваются суперкомпьютерные центры, является неэффективное использование 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 Национальном исследовательском университете «Высшая школа экономики» для высокопроизводительного вычислительного кластера «cHARISMa» разработана система мониторинга задач, автоматически формирующая выводы об их производительности [6]. 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 [7], исследование моделей машинного обучения для прогнозирования рисков основных сердечно-сосудистых событий у пациентов с инфарктом миокарда и различными генотипами [8] и многие другие.

Дополнительные сведения о существующих программных пакетах в кластерных системах приведены в работах [9–11].

К российским кластерным проектам относятся суперкомпьютер МВС-100К, установленный в Межведомственном суперкомпьютерном центре Российской академии наук, и суперкомпьютер «Ломоносов», установленный в Научно-исследовательском вычислительном центре Московского государственного университета имени М.В. Ломоносова в рамках проекта СКИФ[5]. Суперкомпьютеры «Червоненкис», «Галушкин» и «Ляпунов», созданные компанией «Яндекс», также имеют кластерную архитектуру[6]. Они работают на графических ускорителях Nvidia A100 (GPU Nvidia A100 с тензорными ядрами) с системой связи InfiniBand на базе коммутаторов Mellanox (Израиль)[7].

Многие вопросы, связанные с вычислительными ресурсами, необходимыми рядовым пользователям и организациям, возникают в связи с организацией и использованием вычислительных кластеров. Поэтому обзор литературы необходимо дополнить анализом некоторых характерных зарубежных источников. В работах [12, 13] отмечаются недостатки кластерных вычислительных систем. Некоторые из этих недостатков находятся в противоречии с преимуществами, что объясняется спецификой предприятий и пользователей: кластерами трудно управлять без опыта; при большом размере кластера будет трудно обнаружить неисправность.

Проблема при поиске неисправности возникает, поскольку потребитель имеет дело с единой сущностью, и при обнаружении неисправности неясно, с каким из компонентов связана проблема.

К недостаткам кластерных вычислительных систем вычислений может быть также отнесено следующее обстоятельство [14]: не всем потребителям кластеры 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 требуется обладание специальными техническими навыками для работы и администрирования.

[4] *ParTec AG*: *A More Efficient Supercomputer for the AI Revolution*. Frankfurt, Bloomberg; 2024. 43 p.

[5] Центр коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова. https://parallel.ru/cluster. Дата обращения 02.06.2025. [Center for Collective Use of Ultra-High-Performance Computing Resources at Lomonosov Moscow State University. https://parallel.ru/cluster. Accessed June 02, 2025 (in Russ.).]

[6] Чернявцева В. *Яндекс создал три мощнейших в России суперкомпьютера*. https://nplus1.ru/news/2021/11/15/chervonenkis. Дата обращения 02.06.2025. [Chernyavtseva V. *Yandex has created three of Russia's most powerful supercomputers*. https://nplus1.ru/news/2021/11/15/chervonenkis. Accessed June 02, 2025 (in Russ.).]

[7] Россия внезапно ворвалась в мировой топ самых мощных суперкомпьютеров. https://www.cnews.ru/news/top/2021-11-16_rossijskie_superkompyutery. Дата обращения 02.06.2025. [Russia suddenly burst into the world's top most powerful supercomputers. https://www.cnews.ru/news/top/2021-11-16_rossijskie_superkompyutery. Accessed June 02, 2025 (in Russ.).]

Большое число рассмотренных вычислительных кластеров средней и низкой стоимости создается на основе разного рода коммутаторов, в т.ч. коммутаторов Infiniband и Ethernet. В представленном на рис. 1 примере организации вычислительного кластера и его инфраструктуры трафики различных локальных сетей могут пересекаться, если это не затрудняет выполнение основной функции узлов кластера. Узлы кластера N_1–N_{16} обрабатывают пользовательскую нагрузку; U_1, U_2 – управляющие узлы, отслеживающие состояние аппаратного и программного обеспечения кластера и принимающие действия по его перенастройке в связи с каким-либо событием, происходящим в кластере; M_1, M_2 – общие резервные хранилища, в которых хранится информация, доступная всем узлам кластера и используемая ими для доступа к общим данным, в т.ч. к данным о вышедшем из строя узле, которыми может воспользоваться резервный узел; S_1, S_2 – серверы, доступные по общедоступной и клиентской сетям. Коммутатор уровня L_2 частной сети осуществляет обмен данными между узлами кластера, используя аппаратные MAC[8]-адреса. По частной сети передаются командные сообщения, используемые узлами для проверки работоспособности, перенастройки и синхронизации кластера.

Коммутатор уровня L_3 общедоступной сети осуществляет обмен данными, используя межсетевые IP[9] или аппаратные MAC-адреса. На уровне общедоступной сети виртуализируется доступ к кластеру, как к единой системе. Локальная сеть, построенная на основе коммутатора уровня L_{2+} 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 [15].

Для решения поставленных в настоящей работе актуальных задач – организации эффективной функциональной архитектуры кластеров за счет создания нового обеспечения прикладного класса и класса *middleware* 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* [15].

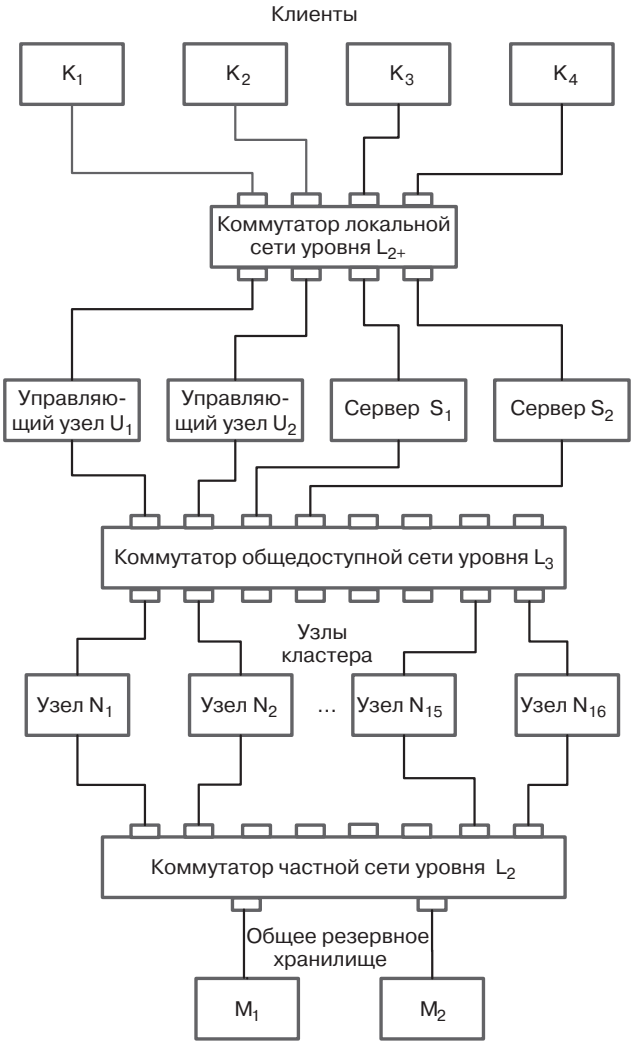


Рис. 1. Вариант организации вычислительного кластера и его инфраструктуры

1. АВТОМАТНЫЕ МОДЕЛИ КЛАСТЕРНЫХ ПРИЛОЖЕНИЙ ПРОМЕЖУТОЧНОГО УРОВНЯ

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» [15], «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

[8] Media access control – контроль доступа к среде.

[9] Internet protocol address – уникальный числовой идентификатор устройства.

определяться в основном приложениями и режимами использования кластера, в т.ч. 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, т.е. на основных уровнях абстракции – от концептуального представления до деталей реализации. Выбор приведенных далее примеров моделей, основанных на схемах программ, базируется на соблюдении высокого уровня общности: алгоритмы должны содержать все базовые алгоритмические конструкции, позволяющие реализовывать последовательности, ветвления и циклы; должна быть предусмотрена 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.

Однако исследовать реальную работу приложений можно только на работающем кластере. Разрешить проблему на предварительных этапах с меньшими 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, т.е. 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.

Удобными моделями для последующего использования в этих целях являются: язык граф-схем алгоритмов (ГСА), конечные автоматы и логико-алгебраические модели на основе логики предикатов первого порядка. В настоящем подразделе предлагается использовать модель конечного частичного автомата Мура[10] [16]. Эта модель также хорошо известна по работам в области микропрограммирования [17, 18]. На рис. 2 и 3 представлены выбранные для иллюстрации создания моделей приложения примеры ГСА: $ГСА_1$ и $ГСА_2$. Основные критерии для выбора – обычные требования корректности ГСА и наличие последовательностей операторов и ветвлений. $ГСА_1$ (рис. 2) 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); 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 рассматриваются как имена унарных предикатов. Значения логических условий – 0 (истина, true) или 1 (ложь, false) вычисляются по завершении выполнения операторов, в т.ч. операторов ввода входных условий (входных сигналов, или входных символов, частичного автомата).

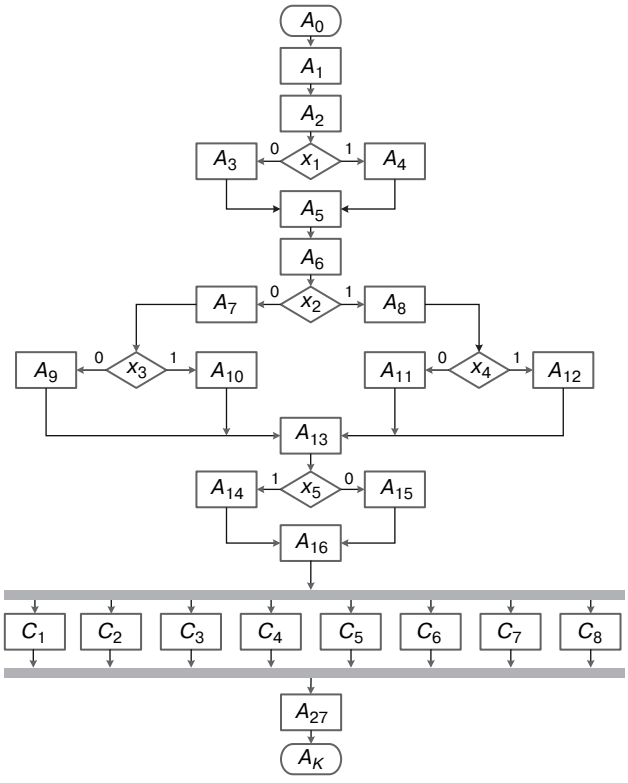


Рис. 2. Граф-схема алгоритма $ГСА_1$ работы приложения для кластера

[10] Гуренко В.В. *Введение в теорию автоматов*. М.: МГТУ им. Н.Э. Баумана, 2013. https://rusist.info/book/10028635?ysclid=mf5p2z07v616437010. Дата обращения 02.06.2025. [Gurenko V.V. *Introduction to Automata Theory*. Moscow: Bauman Moscow State Technical University; 2013. https://rusist.info/book/10028635?ysclid=mf5p2z07v616437010. Accessed June 02, 2025 (in Russ.).]

Построение и исследование автоматных моделей будут проведены для методов SPMD (Single Program, Multiple Data – одиночная («клонируемая») программа, множественный поток данных). В дальнейших подразделах будут построены другие исполнимые модели на основе логико-алгебраического подхода: MPMD (Multiple Programs, Multiple Data – множественный поток программ, множественный поток данных) и MPSD (Multiple Programs, Single Data – множественный поток программ, одиночный поток данных) [1]. Последний метод больше всего подходит к конвейерной обработке данных. Все эти методы используются для достижения параллелизма. Существует ряд вариантов реализации перечисленных методов, используемых в вычислительных кластерах.

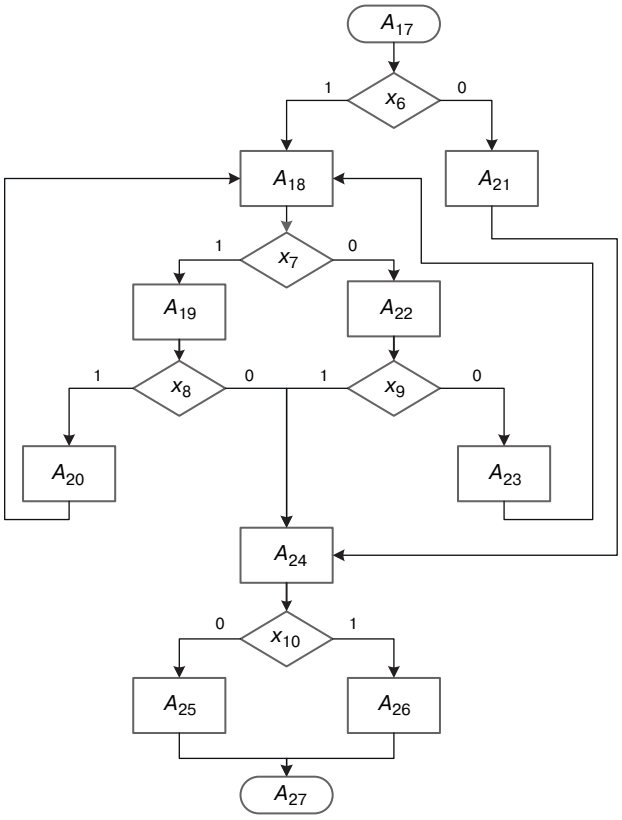


Рис. 3. Граф-схема алгоритма $ГСА_2$ для одной копии параллельного участка приложения для кластера

В качестве начального языка для задания частичных автоматов выбраны системы канонических уравнений (СКУ) [18, 19], которыми описываются переходы из одних состояний в другие. Допускаемые параллельные переходы соответствуют, например, представлению параллельных участков в модифицированных логических схемах алгоритмов, известному из работ по микропрограммированию [20]. В этих схемах параллельные участки считаются частными логическими схемами алгоритмов и допускают простую переинтерпретацию в графическую форму ГСА. Языки параллельных ГСА использовались также в работах [21, 22]. Структуризация состояний иерархических автоматов была предложена ранее в ряде работ [23–25].

В предлагаемых далее автоматных СКУ-моделях использованы следующие понятия. Операторам поставлены во взаимно однозначное соответствие так называемые частные события, представленные унарными предикатами вида $A_t(t)$, 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. 4 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 1–16, а второму – 17–27.

На рис. 2 и 3 использованы общепринятые для ГСА обозначения для логических условий: x_1, x_2, …, x_{10}, также рассматриваемые в языке начального уровня СКУ для задания частичных конечных автоматов как имена унарных предикатов. На рис. 4 и далее на рис. 5 использованы другие имена для трех входных переменных, также удобные в дальнейшем при тестировании приложений при помощи анализа частичных автоматов: x (значение x = true после окончания работы любого оператора без вычисления или без ввода значения условия), nx (значение nx = true, если после окончания работы оператора вычислено или введено ложное значение условия, в противном случае nx = false), yx (значение yx = true, если после окончания работы оператора вычислено или введено истинное значение условия, в противном случае yx = false). Места вычисления и проверки этих переменных однозначно

определяются местом оператора, что при необходимости может быть использовано для обычной нумерации логических условий x_1, x_2, …, x_{10} и использовано далее при составлении СКУ и логико-алгебраических выражений. На рис. 4 первое состояние обозначено как начальное, стартовое (Start), а 28-е – как конечное, завершающее (End).

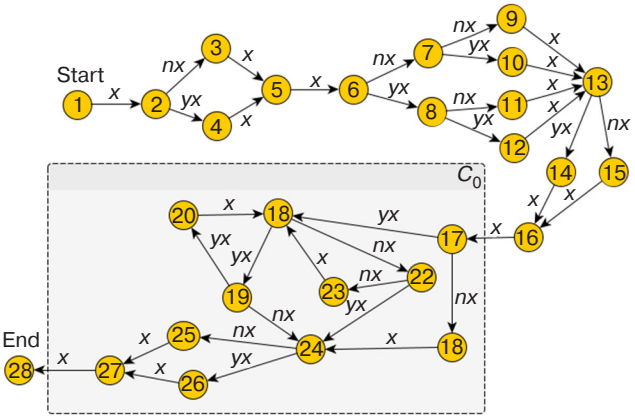


Рис. 4. Граф переходов состояний автоматной модели последовательного приложения; выделен участок C_0, предназначенный для последующего клонирования при переходе к режиму SPMD

Систему канонических уравнений возможно считать системой продукционных правил, предназначенной для представления знаний в автоматных моделях. Продукции возможно использовать для представления знаний, которые могут принимать формы правил вида «*посылка → заключение*, *условие → 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 [26–28]. Набор продукций образует *продукционную систему*, для которой задаются специальные процедуры выбора продукций и выполнение той или иной продукции из числа выбранных.

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 [17, 18, 29]. В дальнейшем понятие «продукция» будет использовано также и при построении логико-алгебраических моделей систем кластерного типа.

2. АВТОМАТНАЯ СКУ-МОДЕЛЬ ПОСЛЕДОВАТЕЛЬНОЙ ЧАСТИ ПРИЛОЖЕНИЯ

На рис. 4 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 $\Gamma\text{СА}_1$, представленной на рис. 2. Остальная часть состояний a_{17}, a_{18}, …, a_{27} принадлежит подграфу C_0, который построен путем разметки состояний автомата Мура на $\Gamma\text{СА}_2$.

На данном этапе рекуррентные предикатные уравнения СКУ $\boldsymbol{SP}_{\mathbf{Seq}}$ для $\Gamma\text{СА}_1$ составлены без учета структурированных операторов C_1, C_2, …, C_8 и, соответственно, без структурированных состояний a_{17}, a_{18}, …, a_{27} автомата, т.е. модель пока охватывает только операторы A_0, A_1, …, A_{16}:

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

где $x_{\text{begin}}(t)$ – входная переменная («сигнал»).

Копии (или клоны) модуля приложения, составленного по СКУ $\boldsymbol{SP}_{\mathbf{Seq}}$ для $\Gamma\text{СА}_1$, 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:

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

В соответствии с этим уравнением при появлении истинного значения входной переменной $x_{\text{begin}}(t)$ = true в автомате в следующем такте устанавливается начальное событие $A_0(t+1)$ = true, что соответствует его зарождению. Это событие сохраняется до тех пор, пока истинно условие его сохранения $A_0(t)\ \&\ \neg x_0(t)$ при $A_0(t+1)$ = true и $x_0(t)$ = false. Далее, при поступлении входного сигнала $x_0(t)$ = true в автомате формируется 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 $A_1(t)\ \&\ \neg z_1(t)$ его сохранения. Оно завершится ($A_1(t+1)$ = false, т.е. это высказывание станет ложным) при выработке оператором A_1 признака $z_1(t)$ = true окончания своей работы. Как видно из рекуррентных предикатных уравнений данной СКУ $\boldsymbol{SP}_{\mathbf{Seq}}$, событие установления истинности антецедента (правого высказывания) происходит в фиксированный момент времени t, а событие установления истинности консеквента (левого высказывания) происходит в следующий момент времени $(t+1)$.

3. АВТОМАТНАЯ СКУ-МОДЕЛЬ РАБОТЫ ОДНОГО ИЗ ПАРАЛЛЕЛЬНЫХ УЧАСТКОВ (КЛОНОВ) ПРИЛОЖЕНИЯ

На рис. 4 представлен граф переходов состояний автоматной модели последовательного приложения; выделен участок C_0, предназначенный для последующего клонирования при переходе к режиму SPMD.

Рекуррентные предикатные уравнения СКУ $\boldsymbol{MD}_{\mathbf{Clon}}$ для ГСА_2:

$$A_{17}(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee A_{17}(t)\ \&\ \neg z_{17}(t);$$
$$A_{18}(t+1) = A_{17}(t)\ \&\ z_{17}(t)\ \&\ x_6(t) \vee A_{20}(t)\ \&\ \&\ z_{20}(t) \vee A_{23}(t)\ \&\ z_{23}(t) \vee A_{18}(t)\ \&\ \neg z_{18}(t);$$
$$A_{19}(t+1) = A_{18}(t)\ \&\ z_{18}(t)\ \&\ x_7(t) \vee A_{19}(t)\ \&\ \neg z_{19}(t);$$
$$A_{20}(t+1) = A_{19}(t)\ \&\ z_{19}(t)\ \&\ x_8(t) \vee A_{20}(t)\ \&\ \neg z_{20}(t);$$
$$A_{21}(t+1) = A_{17}(t)\ \&\ z_{17}(t)\ \&\ \neg x_6(t) \vee A_{21}(t)\ \&\ \neg z_{21}(t);$$
$$A_{22}(t+1) = A_{18}(t)\ \&\ z_{18}(t)\ \&\ \neg x_7(t) \vee A_{22}(t)\ \&\ \neg z_{22}(t);$$
$$A_{23}(t+1) = A_{22}(t)\ \&\ z_{22}(t)\ \&\ \neg x_9(t) \vee A_{23}(t)\ \&\ \neg z_{23}(t);$$
$$A_{24}(t+1) = A_{19}(t)\ \&\ z_{19}(t)\ \&\ \neg x_8(t) \vee A_{22}(t)\ \&\ z_{22}(t)\ \&\ \&\ x_9(t) \vee A_{21}(t)\ \&\ \neg z_{21}(t) \vee A_{24}(t)\ \&\ \neg z_{24}(t);$$
$$A_{25}(t+1) = A_{24}(t)\ \&\ z_{24}(t)\ \&\ \neg x_{10}(t) \vee A_{25}(t)\ \&\ \neg z_{25}(t);$$
$$A_{26}(t+1) = A_{24}(t)\ \&\ z_{24}(t)\ \&\ x_{10}(t) \vee A_{26}(t)\ \&\ \neg z_{26}(t);$$
$$A_{27}(t+1) = A_{25}(t)\ \&\ z_{25}(t) \vee A_{26}(t)\ \&\ z_{26}(t) \vee A_{27}(t)\ \&\ \&\ \neg z_{27}(t).$$

Совместно используемые автоматная СКУ-модель $\boldsymbol{SP}_{\mathbf{Seq}}$ и одна копия СКУ-модели $\boldsymbol{MD}_{\mathbf{Clon}}$, взятые вместе, задают единую автоматную модель, обозначенную как $\boldsymbol{SP}_{\mathbf{Seq}} * \boldsymbol{MD}_{\mathbf{Clon}}$, где символ «*» обозначает операцию объединения двух СКУ в одну общую. Граф переходов состояний для этой модели представлен на рис. 4. Другая интерпретация, как было указано ранее, позволяет считать граф переходов состояний $\boldsymbol{SP}_{\mathbf{Seq}} * \boldsymbol{MD}_{\mathbf{Clon}}$ последовательной композицией частичных автоматов.

4. ПОСЛЕДОВАТЕЛЬНО-ПАРАЛЛЕЛЬНАЯ КОМПОЗИЦИЯ АВТОМАТОВ, ОПРЕДЕЛЯЮЩИХ РАБОТУ КОМПЬЮТЕРОВ КЛАСТЕРА

Граф-схема алгоритма ГСА_1 на рис. 2 содержит параллельные участки, обозначенные сокращенно как структурированные операторы C_1, C_2, …, C_8. При полном одноуровневом представлении каждый из этих операторов заменяется на ГСА_2 на рис. 3 и при программной реализации выполняется независимо от других на «своем» компьютере кластера. Полная сетевая СКУ-модель последовательно-параллельной сети автоматов представлена выражением следующего вида:

$$\boldsymbol{SP}_{\mathbf{Seq}} * (\boldsymbol{ParReplicate}(\mathbf{1..8})\boldsymbol{MD}_{\mathbf{Clon}}),$$

где $\boldsymbol{Replicate}(\mathbf{1..8})$ – включение СКУ в общую систему 8 раз подряд, $\boldsymbol{Par}$ – указание, что эти реплики надо выполнять параллельно.

Граф переходов состояний для данной сети частичных автоматов со структурированными состояниями C_1–C_8 представлен на рис. 5. При построении 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».

Достоинством метода формализации алгоритмов с использованием СКУ является компактное логическое описание функций переходов [16, 17]. Аналогично структурированным состояниям C_1–C_8 вводятся одноименные структурированные события. «Вложением» в каждое структурированное событие являются уже составленные автоматные СКУ $\boldsymbol{MD}_{\mathbf{Clon}}$.

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

$$C_1(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee C_1(t)\ \&\ \neg w_1(t);$$
$$C_2(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee C_2(t)\ \&\ \neg w_2(t);$$
$$C_3(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee C_3(t)\ \&\ \neg w_3(t);$$
$$C_4(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee C_4(t)\ \&\ \neg w_4(t);$$
$$C_5(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee C_5(t)\ \&\ \neg w_5(t);$$
$$C_6(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee C_6(t)\ \&\ \neg w_6(t);$$
$$C_7(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee C_7(t)\ \&\ \neg w_7(t);$$
$$C_8(t+1) = A_{16}(t)\ \&\ z_{16}(t) \vee C_8(t)\ \&\ \neg w_8(t).$$

Каждое из событий C_1–C_8 зарождается при истинности составного высказывания $A_{16}(t)$ & $z_{16}(t)$, т.е. 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, …, 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.

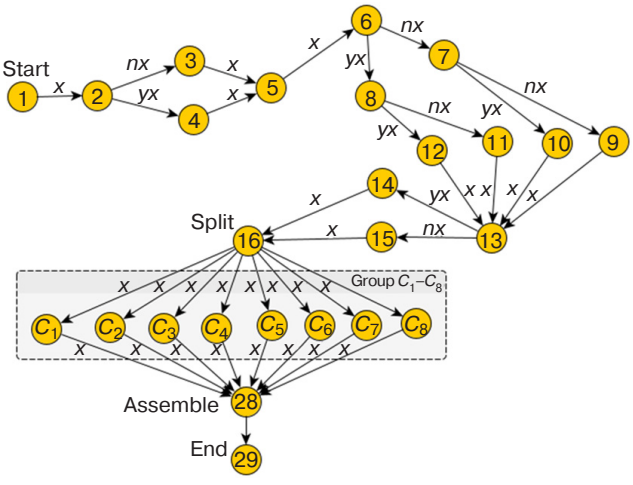


Рис. 5. Последовательно-параллельная сеть автоматов с вложенными состояниями выполнения приложения в кластере в режиме обработки SPMD

Система канонических уравнений $\boldsymbol{S}_{\mathbf{D}}$, описывающая зарождение и сохранение этих событий, имеет следующий вид:

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

Следующее единственное уравнение, обозначенное дополнительно как система СКУ $\boldsymbol{D}_{\mathbf{9}}$, 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, имеют следующий вид:

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

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

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

$$\text{SPMD: } \boldsymbol{SP}_{\mathbf{Seq}} * \boldsymbol{S}_{\mathbf{C}} * \boldsymbol{S}_{\mathbf{D}} * \boldsymbol{D}_{\mathbf{9}} * \boldsymbol{A}_{\mathbf{28}} * \boldsymbol{A}_{\mathbf{29}},$$

причем инициирование описанных ранее параллельных реплик $\boldsymbol{ParReplicate}(\mathbf{1..8})\boldsymbol{MD}_{\mathbf{Clon}}$ осуществляется при наступлении событий C_1–C_8, определяемых подсистемой СКУ $\boldsymbol{S}_{\mathbf{C}}$, а завершение выполнения этих реплик происходит при наступлении событий D_1–D_8, определяемых подсистемой СКУ $\boldsymbol{S}_{\mathbf{D}}$. 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. Предполагается, что другие виды межмодульного взаимодействия в кластере не рассматриваются. Поэтому здесь не рассматриваются вопросы композиции автоматов и другие способы построения сложных автоматов из простых [30].

Автоматное программирование в настоящее время предлагается считать одной из технологий, в существенной степени сокращающей сроки составления программ и упрощающей их тестирование [31]. Системы канонических уравнений также позволяют создавать на их основе программы прикладного, промежуточного и системного уровней.

Как известно, к нарушению корректности ГСА приводят следующие и другие конфигурации: взаимная блокировка, неоднозначность, зависание. Поэтому задачу проверки граф-схемы на корректность предлагается решать на абстрактной модели взаимодействий алгоритмов – на сетях Петри [32, 33]. Методики перехода от параллельной ГСА к сети Петри приведены, например, в работах [21, 22].

5. ФОРМАЛИЗАЦИЯ ЛОГИКО-ВЕРОЯТНОСТНЫХ МОДЕЛЕЙ СЕТЕЙ ЧАСТИЧНЫХ АВТОМАТОВ, СОЗДАВАЕМЫХ НА ОСНОВЕ ЯЗЫКА СКУ

Вводится понятие логико-вероятностной модели «темпоральная вероятностная СКУ» (ТВСКУ), 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:

$$\text{ТВСКУ} = (\text{СКУ}_0, S, X, T_X, W_{TX}, T_Z, W_{TZ}),$$

где сетевая СКУ_0 – исходная, принятая в качестве начального языка описания сети конечных автоматов СКУ, 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 $\{S_0(t), S_1(t), \ldots, S_n(t)\}$, заданных 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, т.е. 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, системой СКУ $\boldsymbol{SP}_{\mathbf{Seq}}$, выполняются действия, заданные следующими тремя 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 $z_1(t)$ – т.е. 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 происходят аналогично. Так, путем перехода от события к событию реализуется логико-вероятностная модель кластерного приложения.

На рис. 5 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 $\boldsymbol{MD}_{\mathbf{Clon}}$ для представления структурированных событий C_1–C_8.

6. РЕЗУЛЬТАТЫ СТАТИСТИЧЕСКИХ ЭКСПЕРИМЕНТОВ С МОДЕЛЯМИ КЛАСТЕРНЫХ СИСТЕМ В ПАРАЛЛЕЛЬНО-ПОСЛЕДОВАТЕЛЬНОМ РЕЖИМЕ SPMD

На основании автоматных вероятностных СКУ-моделей построены имитационные статистические модели выполнения приложений. В таблице 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 времени выполнения приложения на одном узле $(t_{\text{посл}} + Nt_N)$ к сумме $(t_{\text{посл}} + t_N)$ времени выполнения того же приложения в параллельном режиме на всех N узлах кластера t_N и времени $t_{\text{посл}}$ однократного выполнения последовательного участка приложения:

$$k = (t_{\text{посл}} + Nt_N) / (t_{\text{посл}} + t_N).$$

Предполагалось, что исполнение параллельных участков происходит независимо друг от друга. Поскольку времена выполнения последовательного и распараллеливаемого участка заранее неизвестны, их статистические характеристики определяются путем выполнения статистического эксперимента. Поэтому в приведенной формуле $t_{\text{посл}}$ 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 таблицу. 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).

Таблица. Оценки коэффициента ускорения вычислений k, определенные при помощи статистических моделей

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 k от числа узлов в кластере иллюстрирует рис. 6.

При $T = 0$ последовательного участка нет, поэтому значения коэффициента ускорения равны числу задействованных узлов. С ростом значения T эффект от распараллеливания менее выражен, т.к. на результат вычисления коэффициента ускорения время выполнения последовательного участка оказывает большее влияние.

Сложность решения задачи вызвана разветвленностью выбранного алгоритма, а также наличием

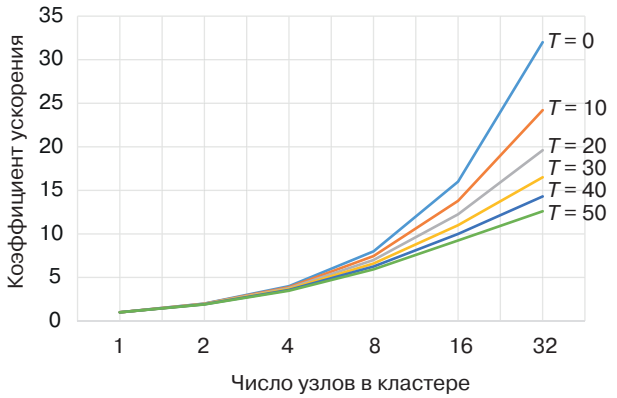


Рис. 6. Результаты статистических экспериментов с моделями кластера, выполняющего последовательно-параллельное приложение L_{Seq} * ***ParReplicate***$(i = 1..8)C_i$

циклов. Время выполнения разветвленных участков программы и число проходимых циклов зависят от вида вводимых условий и на практике могут быть определены 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», т.е. запустить ее на компьютере и исследовать процессы смены событий.

Вычисление коэффициента ускорения k проводилось на основе результатов статистического моделирования. Например, на рис. 7 приведена гистограмма распределения времени $t_{вып} = t_{посл} + Nt_N$ выполнения приложения без распараллеливания при $N = 1$.

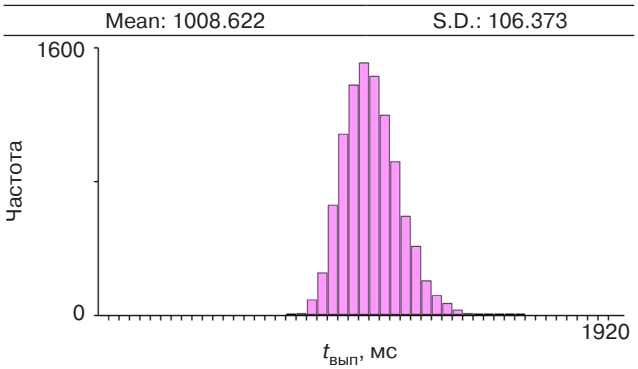


Рис. 7. Гистограмма распределения времени выполнения $t_{вып} = t_{посл} + 32t_{32}$ приложения при $N = 1$ без распараллеливания; по оси абсцисс указаны границы частотных классов для гистограммы, шаг гистограммы – 40 мс; по оси ординат указано число попаданий в каждый частотный класс при объеме выборки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

Здесь $Mean1 = t_{посл} + 32t_{32} = 1008.622$ мс – оценка математического ожидания времени выполнения приложения без распараллеливания.

На рис. 8 приведена гистограмма распределения времени выполнения приложения при $N = 32$ с последовательным участком и распараллеливанием. Здесь $Mean2 = t_{посл} + t_{32} = 80.266$ 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 $t_{посл} = 49.991$ мс.

Тогда $k = (t_{посл} + 32t_{32})/(t_{посл} + t_{32}) = 12.58$. Такой же результат дает вычисление коэффициента ускорения по известной формуле для второго закона Амдала [1]:

$$k = N / [\beta N + (1 - \beta)] = 12.61$$

при $N = 32$ с долей последовательных вычислений

$$\beta = t_{посл} / (t_{посл} + 32t_{32}) = 49.991 / 1008.622 = 0.04957.$$

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

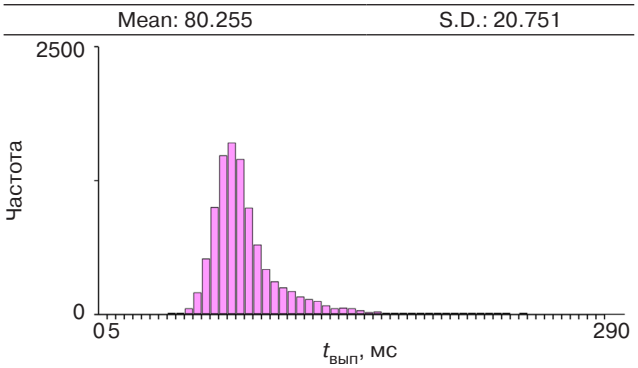


Рис. 8. Гистограмма распределения времени выполнения $t_{вып} = t_{посл} + t_{32}$ приложения при $N = 32$, с последовательным участком и распараллеливанием остальной части; по оси абсцисс указаны границы частотных классов для гистограммы, шаг гистограммы – 5 мс; по оси ординат указано число попаданий в каждый частотный класс при объеме выборки – 10000

7. ПЕРЕХОД ОТ АВТОМАТНЫХ МОДЕЛЕЙ К АСИНХРОННЫМ ЛОГИКО-АЛГЕБРАИЧЕСКИМ МОДЕЛЯМ ФУНКЦИОНИРОВАНИЯ КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ПРОМЕЖУТОЧНОМ УРОВНЕ *MIDDLEWARE*

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

моделей, описан с позиций различных применений и обоснован в работах [34, 35]. Однако на заключительных этапах тестирования моделей при исследовании их динамики рекомендуется непосредственно использовать сети Петри, заменяя или убирая при этом «нововведения», касающиеся дополнительного использования логики предикатов первого порядка. Для сохранения преемственности в названиях модулей кластерного приложения для названий позиций выбраны имена операторных вершин в $ГСА_1$ (рис. 2) и $ГСА_2$ (рис. 3), а для индексации переходов автоматной сети Петри – сдвоенные индексы.

На рис. 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. 9а и 9б составлены канонические уравнения, а для фрагмента на рис. 9в – логико-алгебраические выражения.

Система канонических уравнений для примеров рис. 9а и 9б имеет следующий вид:

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

Автоматная СКУ-модель имеет синхронный характер, и при моделировании приходится вести отсчет текущего времени выполнения событий, что замедляет работу моделирующей программы. Согласно введенным ранее обозначениям для входных переменных в частичных автоматах на рис. 4 и рис. 9б x = true, $nx = \neg x_1$ и $yx = x_1$.

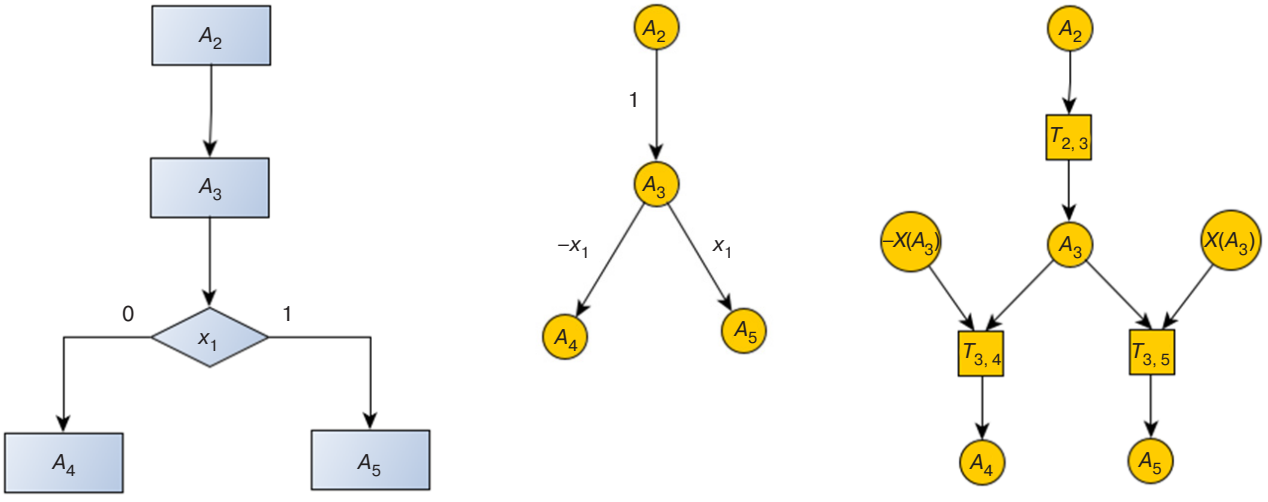
Система асинхронных ЛАОВ для примера на рис. 9в представлена в следующем виде:

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

где undef – неопределенное значение логического условия.

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. 9б), послужившего основой для построения сети Петри (рис. 9в), для чего необходимо соблюдать правило формирования условий операторами, в т.ч. операторами ввода условий.



(а) (б) (в)

Рис. 9. Фрагменты ГСА (а), графа переходов частичного автомата (б) и сети Петри (в)

Правила срабатывания переходов могут далее модифицироваться или дополняться 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 ЛАОВ, для чего привлекается другой аппарат, например, аппарат систем алгоритмических алгебр [36, 37], сетей абстрактных машин [38], реляционных исчислений и алгебр [39].

ЗАКЛЮЧЕНИЕ

1. Основанием, поддерживающим актуальность решаемых в статье 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 могут быть основаны на организации эффективного использования и управления работой кластера за счет лучшей проблемной ориентации путем создания приложений и программного обеспечения промежуточного уровня *middleware*.
2. Метод, предлагаемый и использованный в работе, основан на концепции проектирования архитектуры, определяемой исполнимыми моделями, являющейся разновидностью предметно-ориентированного проектирования.
3. Отличительной особенностью методов, предлагаемых в работе, является использование автоматного, сетевого автоматного и, в перспективе, логико-алгебраического подходов к определению системной и функциональной архитектуры, которые применяются 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, а также возможность их использования в качестве формализованных спецификаций.

СПИСОК ЛИТЕРАТУРЫ

1. Воеводин В.В., Воеводин Вл.В. *Параллельные вычисления*. СПб.: БХВ-Петербург; 2002. 608 с.
2. Pleiter D. Supercomputer Architectures: Current State and Future Trends. *The AQTIVATE Project, European Union's HORIZON MSCA Doctoral Networks Programmer, Grant Agreement No. 101072344.* September 2023. 38 p.
3. Boldyrev A., Ratnikov F., Shevelev A. Approach to Finding a Robust Deep Learning Model. *IEEE Access*. 2025;13: 102390–102406. https://doi.org/10.1109/ACCESS.2025.3578926, https://doi.org/10.48550/arXiv.2505.17254
4. Мишенин Р.М., Костенецкий П.С. Моделирование потока задач вычислительного кластера НИУ ВШЭ с использованием SLURM Simulator. В сб.: *Параллельные вычислительные технологии* (*ПаВТ'2025*): сборник трудов XIX Всероссийской научной конференции с международным участием. М.: 2025. С. 324.
5. Promyslov G., Efremov A., Ilyasov Y., Pisarev V., Timofeev A. Efficiency of Machine Learning Tasks on HPC Devices. В сб.: *Параллельные вычислительные технологии* (*ПаВТ'2025*): сборник трудов XIX Всероссийской научной конференции с международным участием. М.; 2025. С. 56–81.
6. 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. In: Sokolinsky L., Zymbler M., Voevodin V., Dongarra J. (Eds.). *Parallel Computational Technologies* (*PCT'2024*). *Communications in Computer and Information Science*. Springer; 2024. V. 2241. P. 49–64. https://doi.org/10.1007/978-3-031-73372-7_4
7. Сластников С.А., Жукова Л.Ф., Семичаснов И.В. Приложение поиска, анализа и прогнозирования данных в социальных сетях. *Информационные технологии и вычислительные системы.* 2024;1:97–108. https://doi.org/10.14357/20718632240110
8. Kirdeev A., Burkin K., Vorobev A., Zbirovskaya E., Lifshits G., Nikolaev K., Zelenskaya E., Donnikov M., Kovalenko L., Urvantseva I., Poptsova M. Machine learning models for predicting risks of MACEs for myocardial infarction patients with different VEGFR2 genotypes. *Front. Med.* 2024;11:1452239. https://doi.org/10.3389/fmed.2024.1452239

9. Аль-Хулайди А.А., Садовой Н.Н. Анализ существующих программных пакетов в кластерных системах. *Вестник Донского государственного технического университета* (*Вестник ДГТУ*). 2010;10(3-46):303–310. https://elibrary.ru/mvsqql
10. Ладыгин И.И., Логинов А.В., Филатов А.В., Яньков С.Г. *Кластеры на многоядерных процессорах.* М.: Издательский дом МЭИ; 2008. 112 с. ISBN 978-5-383-00142-4. https://elibrary.ru/qmsnap
11. Кокоц А.В. Разработка программной модели функционирования кластерной вычислительной системы. *Вычислительные сети. Теория и практика*. 2016;2(29):6.1. URL: https://network-journal.mpei.ac.ru/. Дата обращения 02.06.2025.
12. Kaur K., Rai A.K. A Comparative Analysis: Grid, Cluster and Cloud Computing. *Int. J. Adv. Res. Computer Commun. Eng*. 2014;3(3):5730–5734.
13. Omer S.M.I., Mustafa A.B.A., Alghali F.A.E. Comparative study between Cluster, Grid, Utility, Cloud and Autonomic computing. *IOSR Journal of Electrical and Electronics Engineering* (*IOSR-JEEE*). 2014;9(6):61–67. http://doi.org/10.9790/1676-09636167
14. Kumar R. Comparison between Cloud Computing, Grid Computing, Cluster Computing and Virtualization. *Int. J. Mod. Computer Sci. Appl*. (*IJMCSA*). 2015;3(1):42–47. http://doi.org/10.13140/2.1.1759.7765
15. Воеводин Вл.В., Жуматий С.А. *Вычислительное дело и кластерные системы*. М.: Изд-во МГУ; 2007. 150 с. ISBN 978-5-211-05440-0
16. Хопкрофт Д., Мотвани Р., Ульман Д. *Введение в теорию автоматов, языков и вычислений*: пер. с англ. М.: Вильямс; 2015. 528 с. ISBN 978-5-8459-1969-4
17. Баранов С.И. *Синтез микропрограммных автоматов* (*Граф-схемы и автоматы*). Л.: Энергия; 1979. 231 с.
18. Вашкевич Н.П. *Синтез микропрограммных управляющих автоматов*. Пенза: Изд-во Пенз. политехн. ин-та; 1990. 115 с.
19. Вашкевич Н.П., Сибиряков М.А. Формальные автоматные модели алгоритмов обработки кэшируемой информации. *Современные наукоемкие технологии*. 2016;(8-2):205–213. https://elibrary.ru/whksst
20. Лазарев В.Г., Пийль Е.И., Турута Е.Н. *Построение программируемых управляющих устройств*. М.: Энергоатомиздат; 1984. 264 с.
21. Анишев П.А., Ачасова С.М., Бандман О.Л. *Методы параллельного программирования*. Новосибирск: Наука; 1981. 180 с.
22. Юдицкий С.А., Магергут В.З. *Логическое управление дискретными процессами. Модели, анализ, синтез*. М.: Машиностроение; 1987. 176 с.
23. Girault A., Lee E.A. Hierarchical finite state machines with multiple concurrency models. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*. 1999;18(6):742–760. https://doi.org/10.1109/43.766725
24. Stefansson E., Johansson K.H. Hierarchical finite state machines for efficient optimal planning in large-scale systems. In: *2023 European Control Conference* (*ECC*). June, Bucharest, Romania. IEEE; 2023. https://doi.org/10.23919/ECC57647.2023.10178139
25. Alur R., Yannakakis M. Model checking of hierarchical state machines. *ACM SIGSOFT Software Engineering Notes*. 1998;23(6):175–188. http://doi.org/10.1145/503502.503503
26. Болотова Л.С. *Системы искусственного интеллекта. Модели и технологии, основанные на знаниях*. М.: Финансы и статистика; 2012. 664 с. ISBN 978-5-279-03530-4
27. Тейз А., Грибомон П., Луи Ж. *Логический подход к искусственному интеллекту: от классической логики к логическому программированию*: пер. с франц. М.: Мир; 1990. 429 с. ISBN 5-03-001636-8
28. *Представление и использование знаний*; под ред. Х. Уэно, М. Исидзука. М.: Мир; 1989. 220 с. ISBN 5-03-000685-0
29. Евреинов Э.В., Косарев Ю.Г. *Однородные универсальные системы высокой производительности*. Новосибирск: Наука, Сибирское отд.; 1966. 308 с.
30. Белов В.В., Воробьев Е.М., Шаталов В.Е. *Теория графов*. М.: Высшая школа; 1976. 392 с.
31. Поликарпова Н.И., Шалыто А.А. *Автоматное программирование*: 2-е изд. СПб.: Питер; 2011. 176 с. ISBN 987-5-4237-0075-1
32. Питерсон Дж. *Теория сетей Петри и моделирование систем*: пер. с англ. М.: Мир; 1984. 368 с.
33. Котов В.Е. *Сети Петри*. М.: Наука; 1984. 160 с.
34. 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;2:3–16. https://elibrary.ru/pfpgml
35. Зинкин С.А. Элементы новой объектно-ориентированной технологии для моделирования и реализации систем и сетей хранения и обработки данных. *Информационные технологии*. 2008;10:20–27.
36. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. *Алгеброалгоритмические модели и методы параллельного программирования*. Киев: Академпериодика; 2007. 634 с.
37. Ющенко Е.Л., Цейтлин Г.Е, Грицай В.П., Терзян Т.К. *Многоуровневое структурное проектирование программ. Теоретические основы, инструментарий*. М.: Финансы и статистика; 1989. 208 с. ISBN 5-279-00233-X
38. Gurevich Y. Abstract State Machines: An Overview of the Project. In: Seipel D., Turull-Torres J.M. (Eds.). *Foundations of Information and Knowledge Systems. Lecture Notes in Computer Science.* Springer; 2004. V. 2942. P. 6–13. https://doi.org/10.1007/978-3-540-24627-5_2
39. Мейер Д. *Теория реляционных баз данных*: пер. с англ. М.: Мир; 1987. 608 с.

REFERENCES

1. Voevodin V.V., Voevodin Vl.V. *Parallel'nye vychisleniya* (*Parallel Computing)*. St. Petersburg: BHV-Petersburg; 2002. 608 p. (in Russ.).
2. Pleiter D. Supercomputer Architectures: Current State and Future Trends. *The AQTIVATE Project, European Union's HORIZON MSCA Doctoral Networks Programmer, Grant Agreement No. 101072344.* September 2023. 38 p.
3. Boldyrev A., Ratnikov F., Shevelev A. Approach to Finding a Robust Deep Learning Model. *IEEE Access*. 2025;13: 102390–102406. https://doi.org/10.1109/ACCESS.2025.3578926, https://doi.org/10.48550/arXiv.2505.17254
4. Mishenin R.M., Kostenetskii P.S. Modeling the task flow of the HSE computing cluster using SLURM Simulator. In: *Parallel Computing Technologies* (*PCT'2025*): Proceedings of the 19th All-Russian Scientific Conference with International Participation. Moscow; 2025. P. 324 (in Russ.).
5. Promyslov G., Efremov A., Ilyasov Y., Pisarev V., Timofeev A. Efficiency of Machine Learning Tasks on HPC Devices. In: *Parallel Computing Technologies* (*PCT'2025*): Proceedings of the 19th All-Russian Scientific Conference with International Participation. Moscow; 2025. P. 56–81 (in Russ.).
6. 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. In: Sokolinsky L., Zymbler M., Voevodin V., Dongarra J. (Eds.). *Parallel Computational Technologies* (*PCT'2024*). *Communications in Computer and Information Science*. Springer; 2024. V. 2241. P. 49–64. https://doi.org/10.1007/978-3-031-73372-7_4
7. Slastnikov S.A., Zhukova L.F., Semichasnov I.V. Application for data retrieval, analysis, and forecasting in social networks. *Informatsionnye tekhnologii i vychislitel'nye sistemy = Journal of Information Technologies and Computing Systems.* 2024;1:97–108 (in Russ.). https://doi.org/10.14357/20718632240110
8. Kirdeev A., Burkin K., Vorobev A., Zbirovskaya E., Lifshits G., Nikolaev K., Zelenskaya E., Donnikov M., Kovalenko L., Urvantseva I., Poptsova M. Machine learning models for predicting risks of MACEs for myocardial infarction patients with different VEGFR2 genotypes. *Front. Med.* 2024;11:1452239. https://doi.org/10.3389/fmed.2024.1452239
9. Al-Khulaidi A., Sadovoy N. Analysis of existing software packages in the cluster systems. *Vestnik Donskogo gosudarstvennogo tekhnicheskogo universiteta = Vestnik of Don State Technical University*. 2010;10(3):303–310 (in Russ.). https://elibrary.ru/mvsqql
10. Ladygin I.I., Loginov A.V., Filatov A.V., Yankov S.G. *Klastery na mnogoyadernykh protsessorakh* (*Clusters on Multi-Core Processors*). Moscow: MPEI Publ.; 2008. 112 p. (in Russ.). ISBN 978-5-383-00142-4. https://elibrary.ru/qmsnap
11. Kokots A.V. Development of a software model of an effective cluster computing system *Vychislitel'nye seti. Teoriya i praktika = Network Journal. Theory and Practice.* 2016;2(29):6.1. Available from URL: https://network-journal.mpei.ac.ru/ (in Russ.). Accessed June 02, 2025.
12. Kaur K., Rai A.K. A Comparative Analysis: Grid, Cluster and Cloud Computing. *Int. J. Adv. Res. Computer Commun. Eng*. 2014;3(3):5730–5734.
13. Omer S.M.I., Mustafa A.B.A., Alghali F.A.E. Comparative study between Cluster, Grid, Utility, Cloud and Autonomic computing. *IOSR Journal of Electrical and Electronics Engineering* (*IOSR-JEEE*). 2014;9(6):61–67. http://doi.org/10.9790/1676-09636167
14. Kumar R. Comparison between Cloud Computing, Grid Computing, Cluster Computing and Virtualization. *Int. J. Mod. Computer Sci. Appl*. (*IJMCSA*). 2015;3(1):42–47. http://doi.org/10.13140/2.1.1759.7765
15. Voevodin Vl.V., Zhumatii S.A. *Vychislitel'noe delo i klasternye sistemy* (*Computing and Cluster Systems*). Moscow: MSU Press; 2007. 150 p. (in Russ.). ISBN 978-5-211-05440-0
16. Hopcroft J.D., Motwani R., Ulman J.D. *Vvedenie v teoriyu avtomatov, yazykov i vychislenii* (*Introduction to Automata Theory, Languages, and Computations*): transl. from Engl. Moscow: Vil'yams; 2015. 528 p. (in Russ.). ISBN 978-5-8459-1969-4 [Hopcroft J.E., Motwani R., Ulman J.D. *Introduction to Automata Theory, Languages and Computation*. Boston, etc.: Addison-Wesley Publ. Comp.; 2001. 521 p.]
17. Baranov S.I. *Sintez mikroprogrammnykh avtomatov* (*Graf-skhemy i avtomaty*) (*Synthesis of Microprogrammed Automata* (*Graph-Schemes and Automata*)). Leningrad: Energiya; 1979. 231 p. (in Russ.).
18. Vashkevich N.P. *Sintez mikroprogrammnykh upravlyayushchikh avtomatov* (*Synthesis of Microprogram Control Automata*). Penza; 1990. 115 c. (in Russ.).
19. Vashkevich N.P., Sibiryakov M.A. The formal automatic models of algorithms of processing of cached data. *Sovremennye naukoemkie tekhnologii = Modern High Technologies*. 2016;(8-2):205–213 (in Russ.). https://elibrary.ru/whksst
20. Lazarev V.G., Piil' E.I., Turuta E.N. *Postroenie programmiruemykh upravlyayushchikh ustroistv* (*Construction of Programmable Control Devices*). Moscow: Energoatomizdat; 1984. 264 p. (in Russ.).
21. Anishev P.A., Achasova S.M., Bandman O.L. *Metody parallel'nogo programmirovaniya* (*Methods of Parallel Programming*). Novosibirsk: Nauka; 1981. 180 p. (in Russ.).
22. Yuditskii S.A., Magergut V.Z. *Logicheskoe upravlenie diskretnymi protsessami. Modeli, analiz, sintez* (*Logical Control of Discrete Processes. Models, Analysis, Synthesis*). Moscow: Mashinostroenie; 1987. 176 p. (in Russ.).
23. Girault A., Lee E.A. Hierarchical finite state machines with multiple concurrency models. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*. 1999;18(6):742–760. https://doi.org/10.1109/43.766725
24. Stefansson E., Johansson K.H. Hierarchical finite state machines for efficient optimal planning in large-scale systems. In: *2023 European Control Conference* (*ECC*). June, Bucharest, Romania. IEEE; 2023. https://doi.org/10.23919/ECC57647.2023.10178139

25. Alur R., Yannakakis M. Model checking of hierarchical state machines. *ACM SIGSOFT Software Engineering Notes*. 1998;23(6):175–188. http://doi.org/10.1145/503502.503503
26. Bolotova L.S. *Sistemy iskusstvennogo intellekta. Modeli i tekhnologii, osnovannye na znaniyakh* (*Artificial Intelligence Systems. Models and Technologies Based on Knowledge*). Moscow: Finansy i statistika; 2012. 664 p. (in Russ.). ISBN 978-5-279-03530-4
27. Thayse A., Gribomont P., Louis J. *Logicheskii podkhod k iskusstvennomu intellektu: ot klassicheskoi logiki k logicheskomu programmirovaniyu* (*Logical Approach to Artificial Intelligence: From Classical Logic to Logical Programming*): transl. from French. Moscow: Mir; 1990. 429 p. (in Russ.). ISBN 5-03-001636-8
28. Ueno H., Ishizuka M. (Eds.). *Predstavlenie i ispol'zovanie znanii* (*Representation and Use of Knowledge*): transl. from. Japan. Moscow: Mir; 1989. 220 p. (in Russ.). ISBN 5-03-000685-0
29. Evreinov E.V., Kosarev Yu.G. *Odnorodnye universal'nye sistemy vysokoi proizvoditel'nosti* (*Homogeneous Universal Systems of High Productivity*). Novosibirsk: Nauka; 1966. 308 p. (in Russ.).
30. Belov V.V., Vorob'ev E.M., Shatalov V.E. *Teoriya grafov* (*Graph Theory*). Moscow: Vysshaya shkola; 1976. 392 p. (in Russ.).
31. Polikarpova N.I., Shalyto A.A. *Avtomatnoe programmirovanie* (*Automata Programming*): 2nd ed. St. Petersburg: Piter; 2011. 176 p. (in Russ.). ISBN 987-5-4237-0075-1
32. Peterson J. *Teoriya setei Petri i modelirovanie system* (*Petri Net Theory and the Modeling of Systems*): transl. from Engl. Moscow: Mir; 1984. 368 p. (in Russ.).
[Peterson J.L. *Petri Net Theory and the Modeling of Systems.* NY: Prentice-Hall; 1981. 310 p.]
33. Kotov V.E. *Seti Petri* (*Petri Nets*). Moscow: Nauka; 1984. 160 p. (in Russ.).
34. Volchikhin V.I., Zinkin S.A. Logic and algebraic models and methods in designing functional architecture of distributed data storage and processing systems. *Izvestiya vuzov. Povolzhskii region. Tekhnicheskie nauki = University Proceedings. Volga Region. Technical Sciences*. 2012;2:3–16 (in Russ.). https://elibrary.ru/pfpgml
35. Zinkin S.A. Elements of a new object-oriented technology for modeling and implementing systems and networks for storing and processing data. *Informatsionnye tekhnologii = Information Technologies*. 2008;10:20–27 (in Russ.).
36. Andon F.I., Doroshenko A.E., Tseitlin G.E., Yatsenko E.A. *Algebroalgoritmicheskie modeli i metody parallel'nogo programmirovaniya* (*Algebraic Algorithmic Models and Methods of Parallel Programming*). Kiev: Akademperiodika; 2007. 634 p. (in Russ.).
37. Yushchenko E.L., Tseitlin G.E, Gritsai V.P., Terzyan T.K. *Mnogourovnevoe strukturnoe proektirovanie programm. Teoreticheskie osnovy, instrumentarii* (*Multilevel structural design of programs. Theoretical foundations, tools*). Moscow: Finansy i statistika; 1989. 208 p. (in Russ.). ISBN 5-279-00233-X
38. Gurevich Y. Abstract State Machines: An Overview of the Project. In: Seipel D., Turull-Torres J.M. (Eds.). *Foundations of Information and Knowledge Systems. Lecture Notes in Computer Science.* Springer; 2004. V. 2942. P. 6–13. https://doi.org/10.1007/978-3-540-24627-5_2
39. Maier D. *Teoriya relyatsionnykh baz dannykh* (*The Theory of Relational databases*): transl. from Engl. Moscow: Mir; 1987. 608 p. (in Russ.).
[Maier D. *The Theory of Relational databases:* 1st ed. Computer Sci. Press; 1983. 656 p.]

Об авторе

Петушков Григорий Валерьевич, проректор, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: petushkov@mirea.ru. SPIN-код РИНЦ 4985-4344, https://orcid.org/0009-0006-0801-429X

About the Author

Grigory V. Petushkov, Vice-Rector, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: petushkov@mirea.ru. RSCI SPIN-code 4985-4344, https://orcid.org/0009-0006-0801-429X