УДК 004.421.6

Коваленко Татьяна Анатольевна, доцент кафедры информатика вычислительная техника, кандидат технических наук, Поволжский государственный университет телекоммуникации и информатики, г. Самара
Перепеченова Диана Ильдаровна, бакалавриат, Поволжский государственный университет телекоммуникации и информатики, г. Самара

DATAFLOW УНИВЕРСАЛЬНАЯ ПАРАДИГМА СОЗДАНИЯ ЯЗЫКОВ ДЛЯ БЕСШОВНОГО ПРОГРАММИРОВАНИЯ ГЕТЕРОГЕННЫХ РАСПРЕДЕЛЕННЫХ СИСТЕМ

Аннотация. В статье рассматривается ситуация на рынке современных вычислительных систем, характеризующаяся стремительным развитием распределённых и гетерогенных архитектур, объединяющих облачные платформы, edge-устройства и IoT-инфраструктуру. Отмечается, что традиционные императивные модели программирования не справляются с вызовами, порождаемыми такой средой: они недостаточно масштабируемы и плохо поддерживают переносимость между платформами. Авторы подчёркивают необходимость перехода к более абстрактным и декларативным подходам, способным скрыть сложность управления разнородными ресурсами. В качестве перспективного решения предлагается концепция предметно-ориентированного языка программирования (DSL), основанного на dataflow-парадигме. Такой язык позволяет описывать вычислительные процессы в виде графов потоков данных, что обеспечивает естественную поддержку параллелизма, асинхронности и распределённости. В работе детализируются ключевые требования к DSL: декларативное задание операторов и связей, динамическое планирование с учётом доступных ресурсов, аннотирование требований к исполнению (CPU, GPU, FPGA и др.), а также оптимизация передачи данных в географически распределённых средах. Предлагаемая архитектура включает модульные абстракции —

операторы, потоки и логические разделы графа, — что упрощает композицию и повторное использование компонентов. Особое внимание уделяется адаптивности: система автоматически реагирует на изменения нагрузки, отказы узлов и динамику доступных ресурсов. В заключение подчёркивается, что разработанный подход способен существенно упростить создание надёжных, масштабируемых и переносимых приложений для современных гибридных вычислительных сред.

Annotation. The article addresses the current state of the modern computing systems market, marked by the rapid advancement of distributed and heterogeneous architectures that integrate cloud platforms, edge devices, and IoT infrastructure. It is observed that traditional imperative programming models fail to meet the challenges posed by such environments—they lack sufficient scalability and offer poor support for cross-platform portability. The authors stress the necessity of shifting toward more abstract and declarative approaches capable of concealing the complexity of managing heterogeneous resources. As a promising solution, the paper proposes a domain-specific language (DSL) grounded in the dataflow paradigm. This language enables computational processes to be expressed as dataflow graphs, thereby providing natural support for parallelism, asynchrony, and distribution. The paper elaborates on key requirements for the DSL: declarative definition of operators and their interconnections, dynamic scheduling based on available resources, annotation of execution requirements (e.g., CPU, GPU, FPGA), and optimization of data transfer in geographically distributed environments. The proposed architecture incorporates modular abstractions—operators, data streams, and logical subgraphs—facilitating component composition and reuse. Special emphasis is placed on adaptability: the system automatically responds to variations in workload, node failures, and changes in resource availability. The paper concludes by underscoring that the proposed approach has the potential to significantly simplify the development of reliable, scalable, and portable applications for modern hybrid computing environments.

Ключевые слова: dataflow, распределенные системы, интернет, парадигмы, императивные модели.
Keywords: dataflow, distributed systems, Internet, paradigms, imperative models.

Современные вычислительные системы эволюционируют в направлении распределённости и гетерогенности, объединяя широкий спектр ресурсов — от масштабных централизованных облачных платформ до периферийных устройств интернета вещей (IoT). Такая тенденция предъявляет серьёзные вызовы разработчикам, которым необходимо учитывать разнообразие аппаратных и программных характеристик каждой среды и создавать сложные шаблонные решения для координации, передачи данных и обеспечения надежности. Исследования демонстрируют, что традиционные императивные модели программирования недостаточно эффективно работают в условиях распределённых и гетерогенных вычислительных инфраструктур, что замедляет процессы разработки, увеличивает вероятность ошибок и препятствует переносу приложений между платформами. Цель данного исследования — создание концептуальной основы для предметно-ориентированного языка программирования (DSL), основанного на dataflow-парадигме, предназначенного для единообразного описания вычислительных процессов, распределённых по разнотипным ресурсам. Такой язык призван снизить необходимость ручного управления инфраструктурой и синхронизацией через декларативное описание потоков данных и их автоматизированное выполнение в смешанных вычислительных средах. Актуальность работы подтверждается повсеместным распространением гибридных облаков, edge computing и IoT, где представленный подход способен значительно упростить процесс разработки.

Существующие парадигмы программирования демонстрируют фундаментальные ограничения при работе с гетерогенными распределёнными системами. Императивный стиль, базирующийся на фон-неймановской архитектуре, ограничен централизованной моделью управления и

взаимодействия «процессор-память». Как отмечается в работе [5, с.29]: «С ростом уровня распараллеливания становится всё более проблематичным эффективно загружать возросшее число функциональных узлов, используя традиционные фон-неймановские подходы». Этот дисбаланс приводит к затруднениям масштабирования и снижению эффективности использования вычислительных ресурсов. Альтернативные парадигмы, такие как событийно-ориентированная или акторная модели, обеспечивают лучшую поддержку асинхронности, но требуют от разработчиков значительных усилий по управлению состояниями и распределением задач. Ручная оркестрация компонентов с различным аппаратным обеспечением (CPU, GPU, FaaS, микросервисы) приводит к громоздкому и ошибкоёмкому коду, при этом затрудняя переносимость из-за жесткой зависимости от конкретных API и инфраструктурных особенностей. Главные проблемы: высокая сложность диспетчеризации, обеспечение устойчивости в динамически меняющихся условиях, низкий уровень абстракции от гетерогенной среды и ограниченные возможности для автоматической оптимизации передачи данных и планирования вычислений. Эти аспекты существенно осложняют разработку масштабируемых и эффективных приложений для современных распределённых систем.

Dataflow-парадигма, предлагающая представление вычислений в виде графа операторов, связанных асинхронными потоками данных, служит универсальной основой для преодоления этих ограничений. Данная модель естественно отражает природу распределённых вычислений, так как операторы исполняются независимо в момент готовности входных данных, устранением необходимости в глобальном состоянии и централизованном контроле. Как подчёркивается в исследовании [3, с.4]: «Алгоритмический граф даёт глубокое понимание механизмов распространения и трансформации данных», что критично для оптимизации параллельных систем. Dataflow-подход обеспечивает высокий уровень абстракции, позволяя декларативно описывать логику обработки данных без привязки к деталям реализации. В

результате он является оптимальной базой для создания эффективных DSL, которые скрывают сложность гетерогенных вычислительных сред. Эволюция распределённых вычислений и переход к облачным технологиям, отражённые в [2, с.7] («Бизнес-сообщество активно развивает новые спецификации, приведшие к появлению традиции облачных вычислений»), подтверждают стремление к универсальным моделям — где dataflow предстает как следующий эволюционный этап.

Исходя из анализа существующих ограничений и преимуществ dataflow, сформулированы основные требования к разрабатываемому DSL. Во-первых, язык должен позволять декларативно задавать графы преобразования данных, в которых разработчик определяет операторы и связи между ними без необходимости ручного контроля параллелизма и распределения. Во-вторых, необходимы встроенные механизмы выявления независимых вычислений и динамического планирования с учетом доступных вычислительных ресурсов. Третье требование связано с поддержкой гетерогенности путем аннотирования операторов требованиями к среде исполнения (CPU, GPU, FPGA, память). Четвертая важная особенность — эффективная оптимизация передачи данных в распределённых средах с географической разбросанностью и высокой латентностью. Как указывается в [6, с.1], оптимальное управление ресурсами требует учета характеристик гетерогенности, динамики и геораспределенности, что влияет на минимизацию затрат на коммуникацию и реализацию распределённого планирования.

Архитектурно DSL опирается на ключевые абстракции — операторы, потоки и разделы графа, которые представляют собой логические группы операторов, способные исполняться как единое целое на заданных ресурсах. Эта модульная конструкция, схожая с описанной в [8, с.56] для видеопроцессинга («алгоритм представлен как конфигурируемый вычислитель, состоящий из последовательно связанных библиотечных блоков»), обеспечивает удобство композиции и повторного использования компонентов. Механизм аннотирования ресурсов дает возможность разработчику указывать

требования к исполнению по типу ускорителя, объему памяти и геолокации. Планировщик, используя эти сведения, автоматически распределяет нагрузку между локальными кластерами, облачными сервисами и edge-устройствами, а также оптимизирует передачу данных, выбирая эффективные маршруты и протоколы для сокращения задержек и затрат. Исполнение поддерживает динамическую оптимизацию графа с адаптацией к состоянию системы, изменению нагрузки и отказам, что гарантирует устойчивость и эффективное использование ресурсов. Это соответствует цели, заявленной в [6, с.1]: «обеспечение эффективного и адаптивного управления вычислительными ресурсами на всех этапах их распределения и решения задач». Таким образом, DSL способствует бесшовному программированию современных гетерогенных распределённых систем.

Modern computing systems are evolving toward distributed and heterogeneous architectures, integrating a wide range of resources—from large-scale centralized cloud platforms to edge Internet of Things (IoT) devices. This trend poses significant challenges for developers, who must account for the diversity of hardware and software characteristics in each environment and create complex templated solutions for coordination, data transfer, and reliability assurance. Research shows that traditional imperative programming models are insufficiently effective in distributed and heterogeneous computing infrastructures, slowing down development processes, increasing the likelihood of errors, and hindering application portability across platforms. The aim of this study is to establish a conceptual foundation for a domain-specific language (DSL) based on the dataflow paradigm, designed to uniformly describe computational processes distributed across heterogeneous resources. Such a language is intended to reduce the need for manual infrastructure and synchronization management by enabling declarative description of data flows and their automated execution in hybrid computing environments. The relevance of this work is confirmed by the widespread adoption of hybrid clouds, edge computing, and IoT, where the proposed approach can significantly simplify the development process.

Existing programming paradigms exhibit fundamental limitations when applied to heterogeneous distributed systems. The imperative style, grounded in the von Neumann architecture, is constrained by a centralized model of "processor-memory" control and interaction. As noted in [5, p. 29]: “With increasing levels of parallelism, it becomes increasingly problematic to efficiently utilize the growing number of functional units using traditional von Neumann approaches.” This imbalance leads to difficulties in scaling and reduced efficiency in computational resource utilization. Alternative paradigms, such as event-driven or actor models, offer better support for asynchrony but require significant developer effort to manage state and distribute tasks. Manual orchestration of components with diverse hardware (CPU, GPU, FaaS, microservices) results in bulky and error-prone code, while also impeding portability due to tight coupling with specific APIs and infrastructure peculiarities. The main problems include high complexity of dispatching, ensuring resilience under dynamically changing conditions, low abstraction from the heterogeneous environment, and limited capabilities for automatic optimization of data transfer and computation scheduling. These aspects significantly complicate the development of scalable and efficient applications for modern distributed systems.

The dataflow paradigm—which represents computations as a graph of operators interconnected by asynchronous data streams—provides a universal foundation for overcoming these limitations. This model naturally reflects the nature of distributed computing, as operators execute independently as soon as input data become available, eliminating the need for global state and centralized control. As emphasized in [3, p. 4]: “An algorithmic graph provides deep insight into the mechanisms of data propagation and transformation,” which is critical for optimizing parallel systems. The dataflow approach offers a high level of abstraction, enabling developers to declaratively describe data processing logic without binding to implementation details. Consequently, it serves as an optimal basis for developing efficient DSLs that hide the complexity of heterogeneous computing environments. The evolution of distributed computing and the shift toward cloud technologies—reflected in [2, p. 7] (“The business community is

actively developing new specifications that have led to the emergence of the cloud computing tradition")—confirm the drive toward universal models, where dataflow emerges as the next evolutionary step.

Based on the analysis of existing limitations and the advantages of dataflow, the following key requirements for the proposed DSL have been formulated. First, the language must allow the declarative definition of data transformation graphs, where developers specify operators and their interconnections without manually managing parallelism or distribution. Second, it must include built-in mechanisms for identifying independent computations and performing dynamic scheduling based on available computing resources. The third requirement concerns supporting heterogeneity through annotations that specify execution environment requirements for operators (CPU, GPU, FPGA, memory). The fourth crucial feature is efficient optimization of data transfer in distributed environments characterized by geographical dispersion and high latency. As stated in [6, p. 1], optimal resource management must account for heterogeneity, dynamics, and geographical distribution, which directly affects minimizing communication costs and enabling distributed scheduling.

Architecturally, the DSL is built upon key abstractions—operators, streams, and graph partitions, where partitions represent logical groups of operators that can be executed as a single unit on designated resources. This modular structure, similar to that described in [8, p. 56] for video processing ("the algorithm is represented as a configurable processor composed of sequentially connected library blocks"), ensures ease of composition and component reuse. The resource annotation mechanism enables developers to specify execution requirements regarding accelerator type, memory capacity, and geolocation. Using this information, the scheduler automatically distributes workloads across local clusters, cloud services, and edge devices, while also optimizing data transfer by selecting efficient routes and protocols to reduce latency and overhead. Execution supports dynamic graph optimization with adaptation to system state, workload fluctuations, and node failures, thereby guaranteeing resilience and efficient resource utilization. This

aligns with the objective stated in [6, p. 1]: “ensuring efficient and adaptive management of computing resources throughout all stages of their distribution and task execution.” Thus, the DSL facilitates seamless programming of modern heterogeneous distributed systems.

СПИСОК ЛИТЕРАТУРЫ

1. А. Г. Феоктистов, Р. О. Костромин, М. Л. Воскобойников и др. Организация вычислительной среды разработки и применения научных рабочих процессов на основе контейнеризации // Вычислительные технологии. 2023. №6. С. 151–164.
2. А.Б. Клименко Эффективное распределение вычислительных ресурсов в геораспределенных гетерогенных динамичных вычислительных средах // Моделирование, оптимизация и информационные технологии. 2024. №4. С. 1–12.
3. Абдурахимова Мадина Анваровна, Видуто Анастасия Андреевна, Елисеева Диана Вячеславовна и др. ГЕТЕРОГЕННЫЕ СЕТИ: ПРЕИМУЩЕСТВА, НЕДОСТАТКИ, ОСНОВНЫЕ ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ РАЗВИТИЯ // Современная наука: актуальные проблемы теории и практики. 2024. №7. С. 26–30.
4. В. Д. Власенко Теория графов: Методические указания к самостоятельной работе для студентов математических и экономических специальностей. Хабаровск: Изд-во Хабар. гос. техн. ун-та, 2005. 23 с.
5. В.Н. Берцун МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ НА ГРАФАХ. Часть 1. Томск: Изд-во НТЛ, 2006. 88 с.
6. Г.И. Радченко РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ. Челябинск: Фотохудожник, 2012. 184 с.
7. Д. Н. Змеев, А. В. Климов, Н. Н. Левченко и др. ПОТОКОВАЯ МОДЕЛЬ ВЫЧИСЛЕНИЙ КАК ПАРАДИГМА ПРОГРАММИРОВАНИЯ БУДУЩЕГО // Информатика и её применения. 2015. №4. С. 29–36.

8. И. М. Никольский Распределенная обработка данных: учебно-методическое пособие. Москва: Издательство Московского университета, 2023. 28 с.
9. Сизов М.М., Зюбин В.Е. КОНФИГУРИРУЕМЫЙ ВЫЧИСЛИТЕЛЬ НА БАЗЕ FPGA ДЛЯ ПОТОКОВОЙ ОБРАБОТКИ ВИДЕОСИГНАЛОВ // Цифровая Обработка Сигналов. 2015. №4. С. 55–57.
10. Ю.В. ШОРНИКОВ, А.В. БЕССОНОВ, Д.Н. ДОСТОВАЛОВ Спецификация и инструментальный анализ гибридных систем // Научный вестник НГТУ. 2015. №4. С. 101–117.

References

1. Feoktistov A.G., Kostromin R.O., Voskoboinikov M.L. et al. Organization of a container-based computational environment for development and application of scientific workflows // Computational Technologies. 2023. No. 6. P. 151–164.
2. Klimenko A.B. Efficient allocation of computing resources in geographically distributed heterogeneous dynamic computing environments // Modeling, Optimization and Information Technologies. 2024. No. 4. P. 1–12.
3. Abdurakhimova M.A., Viduto A.A., Eliseeva D.V. et al. Heterogeneous networks: Advantages, disadvantages, key challenges and development prospects // Modern Science: Current Issues of Theory and Practice. 2024. No. 7. P. 26–30.
4. Vlasenko V.D. Graph Theory: Guidelines for independent study for students of mathematical and economic specialties. Khabarovsk: Publishing House of Khabarovsk State Technical University, 2005. 23 p.
5. Bertsun V.N. Mathematical Modeling on Graphs. Part 1. Tomsk: NTL Publishing House, 2006. 88 p.
6. Radchenko G.I. Distributed Computing Systems. Chelyabinsk: FotoKhudozhnik, 2012. 184 p.

7. Zmeev D.N., Klimov A.V., Levchenko N.N. et al. Dataflow model of computation as a programming paradigm of the future // Informatics and Its Applications. 2015. No. 4. P. 29–36.
8. Nikolaevskiy I.M. Distributed Data Processing: A teaching and methodological guide. Moscow: Moscow University Press, 2023. 28 p.
9. Sizov M.M., Zyubin V.E. Configurable FPGA-based processor for stream video signal processing // Digital Signal Processing. 2015. No. 4. P. 55–57.
10. Shornikov Yu.V., Bessonov A.V., Dostovalov D.N. Specification and tool-based analysis of hybrid systems // Scientific Bulletin of NSTU. 2015. No. 4. P. 101–117.