УДК 04.0:04

# Постановка задачи переноса ресурсоемкой вычислительной нагрузки с локальных машин на удаленные вычислительные узлы

**Боженко Дмитрий Владимирович** bozhenkodv@student.bmstu.ru
**Соколов Александр Павлович** alsokolo@bmstu.ru

*МГТУ им. Н.Э. Баумана, Москва, Россия*

*В работе рассматривается проблема выполнения ресурсоемких вычислительных задач, возникающих при моделировании сложных физических и инженерных процессов. Традиционное использование локальных вычислительных ресурсов ограничивает масштаб и скорость проведения подобных расчетов, что особенно критично в задачах, требующих многократных итераций и перебора параметров. Одним из перспективных решений является перенос вычислительной нагрузки на удаленные вычислительные узлы, включая распределенные вычислительные среды. В статье формулируется постановка данной задачи, обсуждаются основные вызовы, связанные с организацией вычислительных процессов в распределенной среде, а также обосновывается актуальность разработки специализированных инструментов, опирающихся на современные подходы, включая графоориентированное представление вычислительных сценариев.*

***Ключевые слова**: удаленные вычисления, распределенные вычисления, высокопроизводительные вычисления, ресурсоемкие задачи, графоориентированный подход*

**Введение.** Одной из фундаментальных проблем современных научно-инженерных расчетов является высокая вычислительная нагрузка при моделировании сложных физических и инженерных систем. Даже при наличии мощных локальных станций, многие задачи оказываются ограниченными с точки зрения времени, объема данных и масштабируемости. В частности, высокопроизводительные вычисления (High Performance Computing, HPC) уже давно признаны необходимыми для решения крупных симуляционных задач в науке и технике [1].

В задачах, характеризующихся большими объемами данных, жесткой зависимостью между шагами расчета, сложной геометрией и материалами, традиционный подход «все на локальной машине» часто становится недостаточным. Так, в обзорной статье по применению HPC в инженерной симуляции указывается, что использование высокопроизводительных вычислений позволяет обрабатывать более сложные модели, получать более точные результаты и существенно уменьшать время расчетов [2]. Кроме того, в смежных областях, таких как вычислительная гидродинамика, рост вычислительной сложности моделей стимулирует активное использование распределенных вычислений, где задачи разбиваются на подзадачи и распределяются по удаленным узлам [3]. Это подтверждает, что сама природа симуляционных моделей побуждает к переходу от монолитных к распределенным подходам.

Одним из классов ресурсоемких и требующих привлечения значительных вычислительных мощностей являются задачи автоматизированного проектирования композиционных материалов (КМ) [4, 5]. С математической точки зрения указанные задачи являются обратными и предполагают необходимость многократного решения прямых задач микромеханики гетерогенных сред, часто, с применением ресурсоемкого метода конечных элементов. Задачи указанных типов являются актуальными и станут примером апробации разрабатываемых в рассматриваемом направлении научно-технических решений.

Перенос ресурсоемкой вычислительной нагрузки на удаленные вычислительные узлы становится архитектурно необходимым при масштабировании, когда рассматриваемые задачи обладают следующими характерными особенностями [6]: существенная сложность реализуемых алгоритмов; необходимость реализации сложных вычислительных методов [7]; взаимозависимость данных между этапами расчета; требование к наличию специфичных аппаратных средств (SSD, GPU) или лицензируемого ПО.

Таким образом, задача переноса ресурсоемкой вычислительной нагрузки с локальных машин на удаленные вычислительные узлы — это основополагающая часть архитектуры систем, предназначенных для масштабируемых симуляций и автоматизированного проектирования.

**Концептуальная постановка задачи переноса вычислительной нагрузки.** Во многих работах по планированию и распределению вычислительных задач используется понятие графовая модель, где часто каждая задача сложного вычислительного процесса сопоставлена вершине ориентированного ациклического графа (ОАГ) $G = (S, E)$, где $S$ — это множество задач заданного сложного вычислительного процесса; $E$ — множество ребер, отражающих существование зависимостей между выполняемыми задачами [8, 9] (рис. 1).

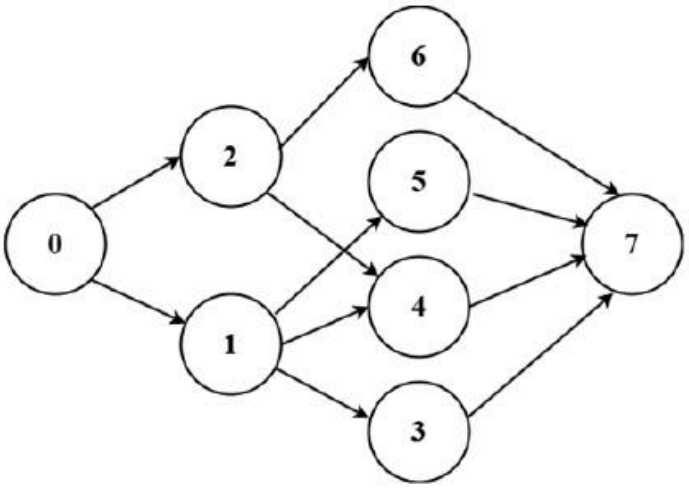


**Рис. 1.** Пример ОАГ с 8 вершинами (задачи) и 11 ребрами (зависимости)

Представленный ОАГ (рис. 1) описывает логическую структуру сложного вычислительного процесса. Узлам графа сопоставляются элементарные вычислительные задачи — завершенные фрагменты расчета с заданными входными данными и выходными результатами. Элементарной вычислитель-

ной задачей $s \in S$ может быть подготовка исходных данных (вершина 0), отдельный этап препроцессирования данных (вершины 1, 2), основной расчет (вершины 3–6), запись или агрегация полученных результатов расчета (вершина 7). Направленное ребро $e_{ij} \in E$ представляет собой четкую последовательность выполнения (зависимость) между задачами $s_i, s_j \in S$, где $i, j \in \in [1, |S|]$. Данную зависимость можно трактовать как канал передачи результатов (файлов, сообщений) или управляющую зависимость порядка выполнения. Это означает, что запуск задачи-потомка $s_j$ допускается лишь после логического завершения ее задачи-предшественника $s_i$ с возможными выходными результатами [10]. Веса $d_{ij}$ребер $e \in E$ естественно интерпретировать как объем передаваемых данных между инцидентными вершинами $s_i, s_j$. На основе данных весов можно вычислить временные накладные расходы на коммуникацию и передачу данных в гетерогенной сети.

В рамках работ [6, 7] описанный графоориентированный подход используется как средство описания вычислительного процесса на более низком уровне абстракции. Это означает, что графовая модель используется для описания сложных вычислительных методов (СВМ), где вершины графа представляют собой множество состояний данных СВМ, а ребра — представляют собой функции перехода из одного состояния данных в другое. В настоящей работе ОАГ, напротив, используется как надсистемная модель процесса для принятия решений о размещении и последовательности выполнения задач на гетерогенных ресурсах; центральной становится проблема планирования с учетом вычислительных и сетевых затрат.

Отдельно следует описать вычислительные узлы и сеть, поскольку рабочий процесс выполняется в рамках конкретного программного окружения. Для каждой задачи $s$ целесообразно фиксировать множество требований к ресурсам (память, число и тип ядер, наличие определенных компонентов программной среды). Таким образом, размещение задачи $s$ на вычислительный узел $u \in U$, где $U$ — множество всех доступных вычислительных узлов, сопровождается рядом следующих ограничений:

1) *ограничение ресурсов*. Каждая задача может быть назначена только на тот вычислительный узел, характеристики которого удовлетворяют ее требованиям по памяти, числу процессорных ядер, программному окружению и др.;

2) *ограничение целостности данных*. Для каждой зависимости между задачами должна быть обеспечена корректная передача выходных данных задачи-предшественника в качестве входных данных задачи-приемника. Таким образом, если в результате планирования и размещений две связанных задачи оказались на разных вычислительных узлах, между данными узлами должен существовать канал связи с удовлетворяющей пропускной способностью передачи данных;

3) *ограничение связности*. Каждая задача должна быть размещена хотя бы на одном вычислительном ресурсе, а вся совокупность назначений должна обеспечивать завершение вычислительного графа в полном объеме;

4) *ограничение на коммуникационные расходы*. При размещении задач необходимо учитывать характеристики каналов связи между узлами, включая пропускную способность и задержки, чтобы суммарные накладные расходы не превышали допустимого порога.

Как правило, характеристики каналов связи в сети задают с помощью матрицы оценки пропускной способности сети $B$

$$B = \begin{pmatrix} b_{11} & \cdots & b_{1|U|} \\ \vdots & \ddots & \vdots \\ b_{|U|1} & \cdots & b_{|U||U|} \end{pmatrix},$$

где $|U|$ — общее количество доступных вычислительных узлов; $b_{ij}$ — оценка пропускной способности (МБ/с, Гбит/с) между узлами $u_i, u_j \in U; i, j \in [1, |U|]$.

Также в описании сети может быть использована матрица оценки сетевых задержек

$$\Lambda = \begin{pmatrix} \lambda_{11} & \cdots & \lambda_{1|U|} \\ \vdots & \ddots & \vdots \\ \lambda_{|U|1} & \cdots & \lambda_{|U||U|} \end{pmatrix},$$

где $\lambda_{ij}$ — оценка сетевой задержки (мс) между узлами $u_i, u_j \in U; i, j \in [1, |U|]$.

Как правило, в гомогенных сетях, представленные матрицы оценки будут близки к симметричным. В гетерогенных сетях будет ярко выражена асимметричность данных матриц.

Таким образом, имея описание порядка выполнения задач $s$сложного вычислительного процесса, описанного графом $G = (S, E)$; описание требований каждой задачи $s$ к вычислительным ресурсам и программным компонентам; оценку характеристики каналов связи сети, где располагается множество доступных вычислительных узлов $U$, становится возможным решить задачу переноса ресурсоемкой вычислительной нагрузки с локальных машин на удаленные вычислительные узлы.

Итак, задача переноса ресурсоемкой вычислительной нагрузки с локальных машин на удаленные вычислительные узлы звучит следующим образом: необходимо построить размещение $\phi: S \to U$, т. е. выбрать сопоставление задач $s$ доступным вычислительным узлам$u$, которое позволит обеспечить корректную последовательность исполнения множества задач $S$ исходного ОАГ $G$ (с учетом выполнения всех ограничений, требований задач к вычислительной среде и архитектуры сети) и минимизировать выбранный целевой функционал (достичь выбранную практическую цель)

$$J = J(\phi) = \min_{\phi: S \to U} J(\phi).$$

В качестве практической цели может быть рассмотрена минимизация общего времени выполнения вычислительного процесса, включающего собственно время вычислений и совокупные затраты на коммуникацию между задачами, размещенными на разных вычислительных узлах.

Понимание задачи переноса ресурсоемкой вычислительной нагрузки с локальных машин на удаленные вычислительные узлы на концептуальном уровне важно, так как оно необходимо для формального описания (построения математической модели) процесса планирования и исполнения задач сложного вычислительного процесса, описанного в виде ОАГ, на вычислительных узлах в гетерогенной сети.

**Обзор методов оптимального распределения задач в гетерогенных средах.** В общем случае оптимальное планирование ОАГ на гетерогенных вычислителях является NP-трудной задачей [11], что подтверждается как классическими, так и современными исследованиями по расписанию ОАГ в распределенных и гетерогенных системах; поэтому на практике применяются приближенные алгоритмы и эвристики.

При постановке и решении задачи обычно оптимизируют многокритериальный функционал**,** в который, в зависимости от сценария эксплуатации, включают следующие цели [11–13]: минимизация времени завершения всего рабочего процесса — базовый критерий для HPC исполнений; минимизация стоимости при использовании облачных ресурсов; минимизация потребления энергии как самостоятельную или отдельную цель; сбалансированность нагрузки между узлами во избежание простоев.

Для решения задач такого типа предлагается ряд методов решения — от эвристических алгоритмов до алгоритмов многокритериальной оптимизации.

1) *HEFT (Heterogeneous Earliest Finish Time)*. Жадный эвристический алгоритм, который основан на задании приоритетов задачам $s$ в графе $G$ и размещении задачи на таком ресурсе, где она будет завершена раньше всего с учетом всех вышеописанных ограничений, зависимостей и межузловых передач. Алгоритм *HEFT* состоит из двух главных шагов [14]:

*Приоритезация*: для каждой вершины $s$ вычисляют верхний ранг — приближенную оценку критического пути, который равен сумме веса вершины и среднего веса всех ребер потомков. Такая приоритезация позволяет задать порядок задач от более критичных к менее критичным.

*Выбор ресурса*: для текущей вершины $s$ перебирают вычислительные узлы $u$ и подсчитывают наиболее ранний возможный момент завершения задачи. В подсчет входит время выполнения задачи, время готовности задач предшественников и задержка передачи данных от задач предшественников, если задачи назначены на разные узлы.

*HEFT* работает за полиномиальное время (вычисление приоритетов и поочередное назначение задач), устойчив к гетерогенности среды выполнения и имеет известные модификации, например алгоритм *IHEFT* [13].

2) *Алгоритмы критического пути.* Основная идея подхода заключается в том, чтобы максимально ускорить критический путь графа — самый длинный путь по суммарному времени вычисления задач и передачи данных по сети, поскольку именно критический путь определяет длительность выполнения процесса решения задачи при параллельном исполнении [14].

На первом этапе алгоритмы такого типа предполагают нахождения критического пути ОАГ. На втором этапе происходит распределение задач $s$ критического пути. Иногда, подсчет задач критического пути может выполняться на одном узле, если суммарные накладные расходы при передаче данных от узла к узлу стоят дороже вычислений. На третьем этапе оставшиеся задачи $s$, не принадлежащие критическому пути, распределяются по вычислительным узлам $u$ таким образом, чтобы не нарушить или не замедлить выполнение критического пути.

3) *Алгоритмы графового разбиения.* Основная идея подхода заключается в том, чтобы разделить весь исходный ОАГ на $k$ частей с примерно равной вычислительной нагрузкой. При этом задачи в каждой части должны быть тесно связаны между собой (между ними должна происходить интенсивная передача данных). Это позволяет минимизировать суммарный вес ребер между частями, тем самым снизить межузловые передачи.

На первом этапе выполнения алгоритма каждой задаче $s$ назначается вес $w_s$, который, как правило, пропорционален ожидаемому времени выполнения задачи $s$. Каждому ребру $e_{ij}$ назначается вес $w_{ij}$, который пропорционален объему передаваемых данных $d_{ij}$. Граф разбивается на $k$ частей, как правило, $k$ — количество доступных вычислительных узлов $U$. Опционально допускается задать коэффициент допустимого дисбаланса $\epsilon(3-10\,\%)$, показывающий, насколько части графа могут отличаться по суммарной «тяжести». После выполнения первого этапа выполняется процесс многоуровневого разбиения графа, состоящий из трех этапов [15].

*Сжатие:* алгоритм последовательно «склеивает» тесно связанные вершины, получая мультиграф, где структура обменов сохранена, но уменьшена размерность графа, по сравнении с исходным ОАГ.

*Грубое разбиение:* на мультиграфе быстро ищется начальное разбиение с балансировкой общего веса задач $w_s$ и суммой весов мультиребер, идущих между частями.

*Развертка и уточнение:* разбиение разворачивается обратно к исходному графу, на каждом уровне выполняются локальные улучшения, например, перемещения нескольких вершин между частями для лучшей балансировки веса ребер развернутого графа между частями.

Результатом работы алгоритма является отображение $\Pi: S \to \{1, \dots, k\}$: номер части для каждой вершины. Далее необходимо назначить части на вычислительные узлы $u$. Такое разбиение позволяет снизить коммуникационные расходы между вычислительными узлами. Также полученное разбиение может выступать в качестве начального распределения задач по вычислительным узлам для другого генетического или популяционного алгоритма.

4) *Генетические алгоритмы*. Поскольку классические алгоритмы непрерывной оптимизации предполагают вещественные параметры, напрямую применять их к задаче распределения задач $s$ на узлы $u$ недопустимо. Решение задачи переноса вычислительной нагрузки представляет собой дискрет-

ное решение: каждая вершина $s \in S$ из конечного множества узлов $S$ должна быть назначена на один из узлов $u \in U$ из конечного множества узлов $U$ при соблюдении вышеописанных ограничений. В таких условиях естественным выбором становятся генетические алгоритмы (ГА) [16], работающие с дискретными значениями и не требующие гладкости целевой функции.

В рамках ГА каждое кандидатное решение представляется вектором значений $x = (x_1, \dots, x_{|S|})$, где $x_i \in U$ — узел, на который назначена задача $s_i$. В качестве целевой функции выбирается однокритериальный функционал $J(\phi)$, представляющий собой суммарные вычислительно-коммуникационные затраты с использованием подсчета времени вычислений задач $s$ и времени межузловых передач при размещении $\phi$.

Алгоритм стартует с инициализации популяции, где часть особей формируется случайно, а часть берется из начального приближения, которое было получено с помощью эвристического алгоритма или алгоритма разбиения исходного графа. Дальнейшая эволюция идет итерациями: селекция отдает приоритет особям с меньшим значением целевой функции $J$, скрещивание комбинирует назначение задач у двух родителей (одиночное/двухточечное или равномерное), а мутация точечно меняет узлы у части задач, стремясь устранить перегрузку и сократить крупные межузловые передачи данных между задачами. После каждого скрещивания и мутации применяется перенос задач на совместимые узлы, сближение сильно связанных вершин, отказ от пар узлов без канала связи, чтобы вновь удовлетворить ограничениям и получить допустимое решение.

Процесс завершается по числу поколений или при стагнации лучшего значения $J$; на выходе выбирается элитная особь с минимальным значением целевой функции. На практике ГА применимы, когда приоритет отдан одному критерию, а структура ограничений и гетерогенность платформы делают пространство решений «зубчатым» и плохо пригодным для детерминированных локальных методов.

**Заключение.** В работе рассмотрено единое представление сложного вычислительного процесса в виде ориентированного ациклического графа, на основе которого на концептуальном уровне сформулирована задача переноса ресурсоемких вычислений с локальных машин на удаленные узлы. Определены ключевые условия допустимости и качества решений: соответствие требований задач ресурсам узлов, сохранение целостности межзадачных передач, а также учет ограничений на коммуникационные расходы с учетом характеристик сети. На этой основе задача сведена к оптимизационному выбору размещения, нацеленного на достижение практических критериев — прежде всего сокращения общей длительности выполнения процесса при одновременном учете вычислительных и сетевых затрат.

## Литература


[1] Qian D. High performance computing: a brief review and prospects. *National Science Review*, 2016, vol. 3, art. no. 16.

[2] Ren S., Li Y. A review of high-performance computing applications in high-speed rail systems. *High-speed Railway*, 2023, vol. 1, pp. 92–96.

[3] Elmisaoui S., Kissami I., Ghidaglia J.-M. High-Performance Computing to Accelerate Large-Scale Computational Fluid Dynamics Simulations: A Comprehensive Study. *International Conference on Advanced Intelligent Systems for Sustainable Development*, Springer Cham, 2024, pp. 352–360.

[4] Steinhauser M.O., Hiermaier S. A Review of Computational Methods in Materials Science: Examples from Shock-Wave and Polymer Physics. *International Journal of Molecular Sciences*, 2009, vol. 10, pp. 5135–5216.

[5] Соколов А.П., Першин А.Ю. Система автоматизированного проектирования композиционных материалов. Часть 1. Концепции, архитектура и платформа разработки. *Известия СПбГЭТУ ЛЭТИ*, 2020, № 8–9, с. 72–83.

[6] Соколов А.П., Першин А.Ю. Система автоматизированного проектирования композиционных материалов. Часть 2. Вычислительная подсистема, распределенные вычисления с применением графоориентированного подхода. *Известия СПбГЭТУ ЛЭТИ*, 2020, № 10, с. 49–63.

[7] Соколов А.П., Першин А.Ю. Графоориентированный программный каркас для реализации сложных вычислительных методов. *Программирование*, 2019, т. 47, № 5, с. 43–55. https://doi.org/10.1134/S0132347419050066

[8] Zhou J., Wang T., Cong P., Lu P., Wei T., Chen M. Cost and makespan-aware workflow scheduling in hybrid clouds. *Journal of Systems Architecture*, 2019, vol. 100. https://doi.org/10.1016/j.sysarc.2019.08.004

[9] Miao Z., Shao C., Li H., Tang Z. Review of Task-Scheduling Methods for Heterogeneous Chips. *Electronics*, 2025, vol. 14, pp. 11–91.

[10] Arabnejad V., Bubendorfer K. Cost Effective and Deadline Constrained Scientific Workflow Scheduling for Commercial Clouds. *International Symposium on Network Computing and Applications*, 2015, vol. 14, pp. 106–113.

[11] Shi Z., Dongarra J.J. Scheduling workflow applications on processors with different capabilities. *Future Generation Computer Systems*, 2006, vol. 22, pp. 665–675.

[12] Hussain M., Wei L.-F., Rehman A., Hussain A., Ali M., Hafeez M.J. An electricity price and energy-efficient workflow scheduling in geographically distributed cloud data centers. *Journal of King Saud University — Computer and Information Sciences*, 2024, vol. 36.

[13] Sandokji S., Eassa F. Dynamic Variant Rank HEFT Task Scheduling Algorithm toward Exascle Computing. *Procedia Computer Science*, 2019, vol. 163, pp. 482–493.

[14] Topcuoglu H., Hariri S., Wu M.-Y. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *Transactions on parallel and distributed systems*, 2002, vol. 13, pp. 260–274.

[15] Karypis G., Kumar V. Parallel Multilevel series k-Way Partitioning Scheme for Irregular Graphs. *SIAM Review*, 2010, vol. 41, pp. 413–658.

[16] Peng Z., Pirozmand P., Motevalli M., Esmaeili A. Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework. *Mathematical Problems in Engineering*, 2022, vol. 2022, pp. 4–15.

# Statement of the problem of transferring resource-intensive computing load from local machines to remote computing nodes

**Bozhenko Dmitry Vladimirovich** bozhenkodv@student.bmstu.ru
**Sokolov Aleksandr Pavlovich** alsokolo@bmstu.ru

*BMSTU, Moscow, Russia*

*This paper examines the problem of solving resource-intensive computational problems arising during the modeling of complex physical and engineering processes. Traditional use of local computing resources limits the scale and speed of such calculations, which is particularly critical for problems requiring multiple iterations and parameter selection. One promising solution is to offload the computational load to remote computing nodes. This paper formulates this problem, discusses the main challenges associated with organizing computational processes in a distributed environment.*

***Keywords:*** *remote computing, distributed computing, high-performance computing, resource-intensive tasks, graph-orienteered approach*