DOI 10.37882/2223-2966.2026.02.20

ПРИМЕНЕНИЕ ДИНАМИЧЕСКОЙ БАЛАНСИРОВКИ НАГРУЗКИ ДЛЯ ЭФФЕКТИВНОЙ ГЕНЕРАЦИИ ФРАКТАЛЬНЫХ БАССЕЙНОВ ПРИТЯЖЕНИЯ НА СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМАХ

APPLICATION OF DYNAMIC LOAD BALANCING FOR EFFICIENT GENERATION OF FRACTAL BASINS OF ATTRACTION ON SUPERCOMPUTER SYSTEMS

T. Leontyeva
S. Bryutova

Summary. Generating fractal basins of attraction for Lagrange points using Newton's method is an important but extremely computationally complex task in nonlinear dynamics, depending on the efficiency of parallelisation. At the initial stage, it was established that traditional static data decomposition causes a significant load imbalance due to the irregular complexity of fractal boundaries. As a result, the scalability of the algorithm on a supercomputer cluster decreased sharply, reaching only 5.8 % with 448 cores. To solve the problem, a dynamic Master/Worker architecture with a fine-grained task pool and point-to-point MPI exchange was adapted. Performance analysis and overhead profiling were performed. The application of the dynamic method provided stable efficiency of 32–34 % when scaling to hundreds of cores, which is 5.7 times higher than the original approach. The generation time for an 8K image was reduced from 25 to 4.4 minutes. The proposed approach demonstrates the effective use of supercomputer resources for a wide class of tasks with irregular computational structures.

Keywords: fractal structure, Newton's method, three-body problem, load balancing, dynamic scheme, master-worker, parallel code acceleration, High-performance computing (HPC).

Леонтьева Татьяна Владимировна
Кандидат технических наук, доцент,
Санкт-Петербургский Политехнический университет Петра Великого
leontieva_tv@spbstu.ru
Брютова София Даниловна
Санкт-Петербургский Политехнический университет Петра Великого
bryutova.sd@edu.spbstu.ru

Аннотация. Генерация фрактальных бассейнов притяжения точек Лагранжа методом Ньютона является важной, но крайне вычислительно сложной задачей нелинейной динамики, зависящей от эффективности распараллеливания. На начальном этапе было установлено, что традиционная статическая декомпозиция данных вызывает значительный дисбаланс нагрузки из-за нерегулярной сложности фрактальных границ. В результате масштабирование алгоритма на суперкомпьютерном кластере резко снижалось, достигая лишь 5.8 % при 448 ядрах. Для решения проблемы была адаптирована динамическая архитектура Master/Worker с мелкозернистым пулом задач и точечным обменом MPI. Проведён анализ производительности и профилирование накладных расходов. Применение динамического метода обеспечило стабильную эффективность 32–34 % при масштабировании до сотен ядер, что в 5.7 раза выше исходного подхода. Время генерации 8K-изображения сократилось с 25 до 4.4 минут. Предложенный подход демонстрирует эффективное использование ресурсов суперкомпьютеров для широкого класса задач с нерегулярной вычислительной структурой.

Ключевые слова: фрактальная структура, метод Ньютона, проблема трёх тел, балансировка нагрузки, динамическая схема, мастер-воркер, ускорение параллельного кода, высокопроизводительные вычисления.

Введение

Построение фрактальных бассейнов притяжения методом Ньютона представляет собой вычислительно емкую задачу с высокой степенью параллелизма. Кроме того, её отличительную черту составляет крайне неравномерная трудоемкость, поскольку вблизи фрактальных границ число итераций, необходимых для достижения сходимости, возрастает на порядки по сравнению с внутренними областями.

В таких условиях традиционное статическое распараллеливание (фиксированное разбиение данных) становится неэффективным. Оно приводит к существенному дисбалансу нагрузки, когда часть вычислительных узлов простаивает в ожидании завершения расчетов наиболее сложных участков.

Целью работы является повышение эффективности использования суперкомпьютера путем реализации динамической схемы распараллеливания (архитектура Master-Worker), обеспечивающей адаптивное распределение задач и минимизацию простоя ядер.

Обзор литературы

Проблема визуализации бассейнов притяжения в задаче трех тел подробно рассматривается в современной астрофизике. В работах Е. Зоотоса [1, 2] показано, что применение метода Ньютона к уравнениям движения приводит к формированию фрактальных структур, геометрия которых критически зависит от параметров системы. Вычислительная сложность таких задач классифицируется как нерегулярная, поскольку время сходимости итерационного процесса варьируется на по-

рядки в зависимости от близости начальной точки к границе фрактала.

В контексте высокопроизводительных вычислений (HPC) задачи генерации фракталов (например, множества Мандельброта или Жюлиа) традиционно используются для тестирования алгоритмов балансировки нагрузки. Как отмечают А. Грамма и А. Гупта [3], статическое распределение данных (Block distribution) в таких случаях неизбежно приводит к простою вычислительных узлов. Л.Б. Соколинский [4] исследует модель BSF (Bulk-Synchronous Farm) и указывает, что для итерационных алгоритмов с переменной трудоемкостью необходимы адаптивные схемы планирования.

Существует несколько подходов к решению проблемы дисбаланса. Первый заключается в использовании динамической балансировки (Dynamic Load Balancing, DLB), где задачи распределяются по запросу [5]. Второй подход, описанный в работе Д. Бадена [6], предполагает использование иерархических структур данных (декомпозиция областей), однако он сложен в реализации. Для кластерных систем с распределенной памятью наиболее перспективной считается схема Master-Worker («Управляющий–Рабочий»), эффективность которой доказана для широкого класса задач [7]. В данной работе мы адаптируем этот подход для специфики расчетов на языке Python с использованием библиотеки mpi4py [8, 9], учитывая накладные расходы на межпроцессное взаимодействие. Данный инструмент является фактически стандартом для реализации MPI в интерпретируемых средах, обеспечивая высокую эффективность при сохранении удобства разработки [10].

Методология исследования

Реализованный алгоритм основан на итерационном методе Ньютона для системы нелинейных уравнений, задающей положение равновесных точек в системе «Земля–Луна». В случае задачи трёх тел состояния равновесия определяются уравнениями, получаемыми из градиента эффективного потенциала двух тех (с учетом гравитационной и центробежной сил) $\nabla\Omega(x, y) = 0$.

Метод Ньютона итеративно уточняет приближение:

$$\upsilon_{n+1} = \upsilon_n - J^{-1}(\upsilon_n)F(\upsilon_n),$$

где $\upsilon_n = [x, y]^T$ — вектор состояния, $F(\upsilon_n)$ — вектор-функция системы уравнений, J — матрица Якоби.

Разные начальные условия (x_0, y_0) сходятся к разным решениям (точкам Лагранжа), тем самым формируя на плоскости бассейны притяжения различных корней.

Неоптимизированный параллельный алгоритм основывался на модели Bulk-Synchronous Farm со статической декомпозицией данных [4], при которой все изображение ($W \cdot H$) фиксированно разделялось на равные горизонтальные полосы, соответствующие количеству MPI-процессов. Каждый процесс вычислял свою часть независимо. Такой подход не учитывал крайне нерегулярный характер вычислительной сложности фрактальных структур, где число итераций метода Ньютона существенно возрастает в окрестности фрактальных границ. В результате заранее заданное равномерное разбиение данных приводило к выраженному дисбалансу нагрузки: процессы, получившие относительно «простые» области, завершали работу значительно раньше остальных и вынуждены были простаивать, ожидая окончания вычислений в наиболее трудоёмких участках изображения, что непосредственно приводило к деградации масштабируемости. При увеличении числа задействованных ядер до 448 эффективность параллельного варианта снижалась до 5.8 %, что фактически делало дальнейшее наращивание ресурсов бессмысленным.

Кроме того, указанные недостатки статической схемы усугублялись необходимостью синхронизированного слияния результатов на узле-мастере с использованием операции MPI_Gatherv. Хотя эта стадия не являлась главным источником задержек, обязательное коллективное взаимодействие усиливало эффект простоя: все процессы, включая те, что завершили свою часть значительно раньше, были вынуждены ждать наиболее «медленный» процесс, создавая дополнительный барьер и снижая общую производительность. В результате, наложенные ограничения показали неэффективность статического подхода при решении задачи, для которой характерны значительная неравномерность вычислений и существенные различия во времени сходимости метода Ньютона в зависимости от начальных условий.

Для устранения дисбаланса была разработана адаптированная под условия задачи версия архитектуры Master/Worker, опирающаяся на динамический пул задач. Оптимизация заключается в переходе к мелкозернистой параллельности (одна строка изображения — это одна задача) и использовании точечных сообщений MPI (MPI_Send, MPI_Recv) для асинхронного обмена заданиями.

Процесс с рангом 0 (Мастер) берет на себя роль планировщика. Он не участвует в вычислениях напрямую, а управляет очередью задач и агрегирует результаты. Остальные процессы (Воркеры) работают в бесконечном цикле, запрашивая задачи у Мастера.

Такая схема обеспечивает автоматическую балансировку: если Воркер получает «легкую» строку (которая сойдется за небольшое количество итераций), он бы-

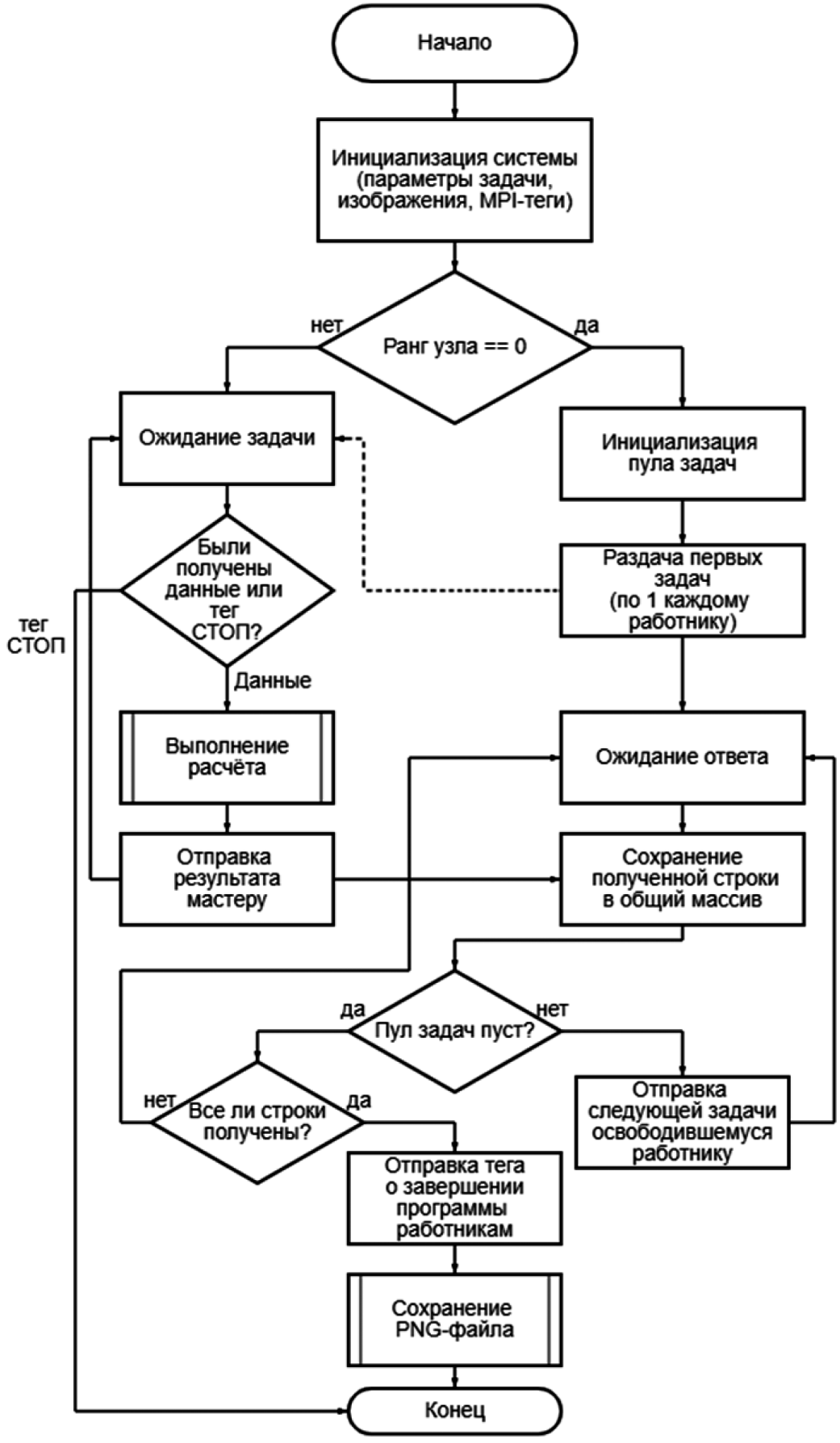

Рис. 1. Блок-схема алгоритма

стро возвращает результат и получает следующую задачу. Воркер, получивший «тяжелую» строку (фрактальная граница), тратит больше времени, но не задерживает других, так как они продолжают разбирать задачи из общего пула.

Алгоритм был реализован на языке Python с использованием библиотеки mpi4py. Выбор интерпретируемого языка высокого уровня обусловлен необходимостью быстрого прототипирования. Несмотря на наличие Global Interpreter Lock (GIL) и накладные расходы на сериализацию объектов при передаче сообщений MPI, высокая арифметическая сложность вычисления каждой точки (множественные операции с плавающей точкой, тригонометрия) позволяет нивелировать эти задержки. Как будет показано далее, даже в условиях интерпретатора динамическая балансировка позволяет достичь приемлемой эффективности масштабирования. Для минимизации накладных расходов на передачу данных использовались точечные неблокирующие операции или оптимизированные стандартные send/recv с буферизацией типов numpy.

Для оценки качества распараллеливания использовались стандартные метрики высокопроизводительных вычислений.

Ускорение (Speedup) $S(n)$ на n процессорных ядрах определялось как отношение времени последовательного выполнения T_1 к времени параллельного выполнения T_n:

$$S(n) = \frac{T_1}{T_n}$$

Эффективность (Efficiency) $E(n)$ показывает среднюю долю полезной загрузки каждого ядра и рассчитывается по формуле:

$$E(n) = \frac{S(n)}{n} \cdot 100\%$$

Данные метрики позволяют оценить масштабируемость алгоритма и накладные расходы на организацию вычислений.

Результаты

Для проведения исследования использовался суперкомпьютерный кластер «Политехник — РСК Торнадо», имеющий 612 вычислительных узлов с двумя 14-ядерными процессорами Intel Xeon E5-2697 v3, распределенную память, вследствие чего каждый узел имеет собственную оперативную память (64 ГБ), недоступную напрямую другим узлам. Для запуска задач на кластере используется система управления заданиями Slurm. Тестирование выполнялось для изображений разрешением 1600×1600, 4096×4096 и 8192×8192 пикселей.

Сравнение производительности проводилось относительно последовательной версии алгоритма и версии со статическим распределением данных. Сводные результаты представлены в Таблице 1.

Таблица 1.

№	*WxH*	*N*	Распределение нагрузки					
			динамическое				статическое	
			T_n	T_1	$S(n)$	$E(n)$	T_n	$E(n)$
1	1600×1600	28	162.24	1526.14	9.41	33.6 %	184.61	29.5 %
2	1600×1600	56	82.32	1526.14	18.5	33.1 %	147.61	18.5 %
3	1600×1600	112	42.43	1526.14	36.0	32.1 %	104.56	13.1 %
4	1600×1600	224	20.40	1526.14	74.81	33.4 %	79.59	8.6 %
5	4096×4096	224	132.80	9735.0	73.30	32.7 %	458.48	9.5 %
6	8192×8192	224	522.05	38940.0	74.60	33.3 %	1788.15	9.7 %
7	8192×8192	448	261.77	38940.0	148.76	33.2 %	1511.61	5.8 %

Обозначения в таблице:

N — количество ядер

T_1 — время выполнения последовательного алгоритма (сек.)

T_n — время выполнения параллельного алгоритма (сек.)

S(*n*)— ускорение

E(*n*) — эффективность

Полученное фрактальное изображение (см. рис. 2).

Визуальный анализ полученного фрактального изображения (Рис. 2) демонстрирует сложную структуру границ, разделяющих бассейны притяжения. Именно эти пограничные области соответствуют точкам с максимальной вычислительной трудоемкостью, требующим значительно большего числа итераций для сходимости метода Ньютона.

Тесты, проведенные для одного и того же разрешения (1600x1600) показывают, насколько эффективно алгоритм ускоряет решение задачи фиксированного размера при увеличении числа процессов.

Эффективность показывает, насколько оптимально были использованы выделенные вычислительные ресурсы.

Результаты замеров производительности, представленные в Таблице 1, подтверждают гипотезу о неэффективности статической декомпозиции для данного класса задач. При увеличении числа вычислительных ядер с 28 до 224 на задаче фиксированного размера (1600×1600) эффективность статического алгоритма демонстрирует

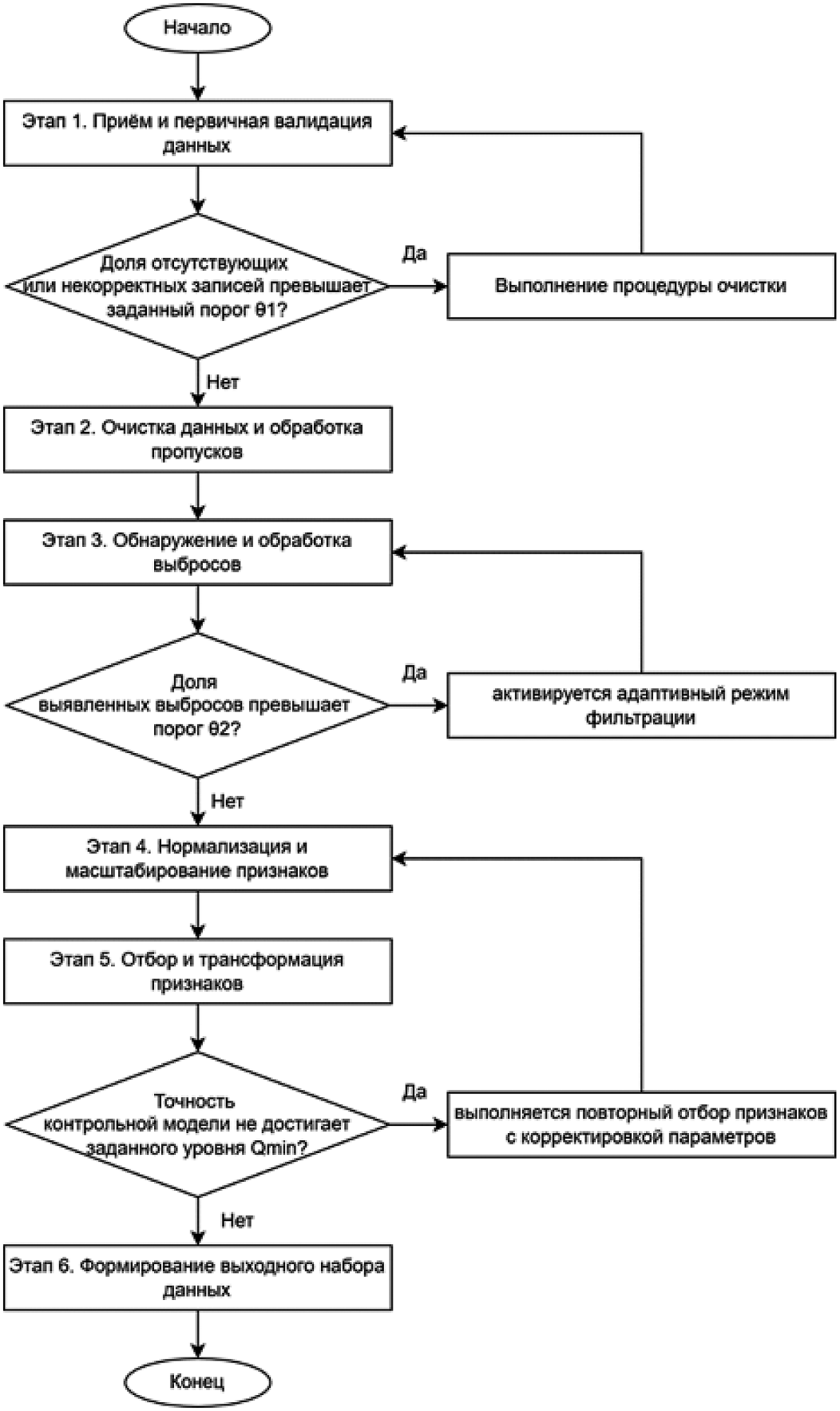

Рис. 1. Алгоритм предподготовки данных для подсистемы детектирования аномалий

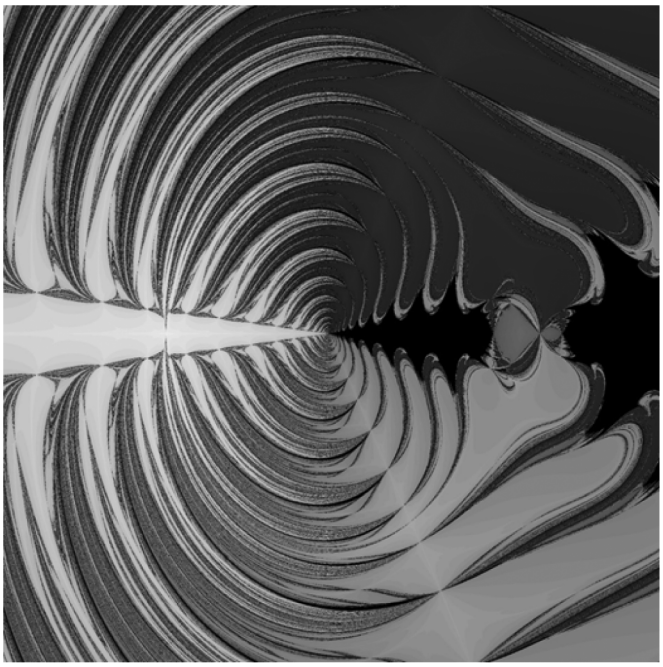

Рис. 2. Фрактал Ньютона

катастрофическое падение с 29.5 % до 8.6 %. Это доказывает, что с ростом параллелизма время простоя процессоров начинает доминировать над временем полезных вычислений из-за неустраненного дисбаланса нагрузки.

В отличие от статического подхода, предложенная динамическая схема Master-Worker показала высокую устойчивость к масштабированию. На Рис. 3 представлена зависимость ускорения $S(n)$ от числа ядер. График показывает практически линейный рост производительности: при удвоении вычислительных ресурсов (переход от 112 к 224 ядрам) ускорение также возрастает почти в два раза.

График эффективности на Рис. 4 подтверждает стабильность алгоритма. Кривая представляет собой почти горизонтальную линию, удерживаясь в узком диапазоне 32–34 %. Отсутствие нисходящего тренда при увеличении числа процессов доказывает, что механизм динамического распределения успешно компенсирует неравномерность вычислений.

Стабилизация эффективности на уровне ≈ 33% (вместо теоретических 100 %) объясняется архитектурными особенностями реализации. Во-первых, сказываются накладные расходы интерпретатора Python и библиотеки mpi4py на сериализацию объектов при обмене сообщениями. Во-вторых, схема с централизованным диспетчером (Master) неизбежно вносит задержки на коммуникацию. Тем не менее, достигнутый уровень является весомым показателем для высокоуровневого скриптового языка.

Рис. 3. График зависимости ускорения от количества ядер

Рис. 4. График зависимости эффективности от количества ядер

Практическая ценность подхода наиболее ярко проявилась при решении задачи сверхвысокого разрешения (8K, 8192×8192). Время генерации изображения было сокращено с 25 минут (статический подход) до 4.4 минуты (динамический подход), что соответствует ускорению в 5.7 раза.

Заключение

В работе представлен эффективный параллельный алгоритм построения фрактальных бассейнов притяжения методом Ньютона. Экспериментально подтверждено, что из-за специфики задачи традиционное статическое распараллеливание оказывается несостоятельным, демонстрируя падение эффективности до критической отметки 5.8 % при увеличении числа ядер.

Внедрение динамической архитектуры Master-Worker позволило преодолеть барьер дисбаланса нагрузки. Предложенное решение обеспечило стабильную эффективность вычислений на уровне 32–34 % даже с учетом накладных расходов интерпретируемого языка Python. Практическая значимость подхода подтверждается сокращением времени генерации изображений сверхвысокого разрешения (8K) в 5.7 раза — с 25 до 4.4 минут. Полученные результаты доказывают, что динамическая балансировка позволяет успешно масштабировать задачи с нерегулярной вычислительной сложностью на кластерных системах, автоматически адаптируясь к структуре исследуемых фракталов.

ЛИТЕРАТУРА

1. Zotos E.E. Fractal basins of attraction in the planar circular restricted three-body problem with oblateness and radiation pressure // Astrophysics and Space Science. — 2016. — Vol. 361, № 6. — P. 181.
2. Zotos E.E. Investigating the Newton–Raphson basins of attraction in the restricted three-body problem with modified Newtonian gravity / [Электронный ресурс]. — Режим доступа: https://arxiv.org/abs/1803.07400 (дата обращения: 08.10.2025).
3. Grama A. Introduction to parallel computing / A. Grama, A. Gupta, G. Karypis, V. Kumar. — Harlow: Pearson Education, 2003. — 636 p.
4. Соколинский Л.Б. Модель параллельных вычислений BSF и её применение для оценки масштабируемости итерационных алгоритмов // Вестник ЮУрГУ. Серия «Математическое моделирование. Программирование». — 2018. — Т. 11, № 4. — С. 5–22.
5. Casanova H. Dynamic Load Balancing of Fractal Image Compression on a Network of Workstations // Proceedings of the High-Performance Computing Symposium. — Montreal, 1997. — P. 12–19.
6. Baden S.B. Dynamic load balancing of parallel fractals // Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications. — New York: ACM, 1988. — P. 115–123.
7. Гергель В.П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем: учебник. — М.: Изд-во МГУ, 2010. — 544 с.
8. Dalcin L.D. MPI for Python / L.D. Dalcin, R.R. Paz, M. Storti // Journal of Parallel and Distributed Computing. — 2005. — Vol. 65, № 9. — P. 1108–1115.
9. Dalcin L.D. Parallel distributed computing using Python / L.D. Dalcin, R.R. Paz, P.A. Kler, A. Cosimo // Advances in Water Resources. — 2011. — Vol. 34, № 9. — P. 1124–1139.
10. Gorelick M. High Performance Python: Practical Performant Programming for Humans / M. Gorelick, I. Ozsvald. — Sebastopol: O'Reilly Media, 2020. — 468 p.
11. Волков К.Н. Балансировка нагрузки процессоров при решении краевых задач механики жидкости и газа сеточными методами // Вычислительные методы и программирование. — 2012. — Т. 13. — С. 120–130.
12. Расолько Г.А. Технологии программирования: учеб.-метод. пособие. В 2 ч. Ч. 2 / Г.А. Расолько, Е. В. Кремень, Ю.А. Кремень. — Минск: БГУ, 2022. — 1 электрон. опт. диск (CD-ROM).
13. OXRSE Training. Parallelising Mandelbrot Set Generation. University of Oxford, 2021 / [Электронный ресурс]. — Режим доступа: https://train.rse.ox.ac.uk/material/HPCu/high_performance_computing/parallel_computing/04_practical (дата обращения: 02.10.2025).

Журнал «Современная наука: актуальные проблемы теории и практики»