УДК 004.75+004.272+519.83
DOI: 10.26102/2310-6018/2025.51.4.068

Теоретико-игровые модели координации ресурсов в распределённых системах потокового анализа данных

М.В. Бляхеров[1]✉, Е.С. Петрова[2]

[1]Воронежский институт высоких технологий, Воронеж, Российская Федерация
[2]Воронежский государственный технический университет, Воронеж, Российская Федерация

Резюме. Современные распределённые системы потокового анализа данных, такие как Apache Spark, сталкиваются с фундаментальной проблемой координации ресурсов в условиях стратегического поведения вычислительных узлов. Традиционные алгоритмы планирования (FIFO, Fair Scheduler) не учитывают, что каждый исполнитель (executor) стремится максимизировать собственную локальную производительность, что приводит к системным проблемам: «трагедии общих ресурсов», дисбалансу нагрузки из-за перекоса данных (Data Skew) и общему снижению эффективности кластера. В статье предлагается подход к решению этой проблемы на основе теоретико-игрового моделирования. Исследование систематизирует и адаптирует кооперативные и некооперативные модели теории игр для задач управления ресурсами в среде Apache Spark. В рамках кооперативного подхода детально формализован и адаптирован алгоритм Shapley Value, позволяющий количественно оценить вклад каждого вычислительного узла в общую производительность системы и обеспечить справедливое распределение вычислительных ресурсов между участниками коалиции. Для управления конкуренцией разработан аукционный механизм, основанный на принципе Викри (второй цены), который стимулирует узлы к честному заявлению своих потребностей. Практическая значимость работы подтверждена разработкой и внедрением модульной подсистемы оптимизации, интегрированной со стеком мониторинга Prometheus/Grafana. Экспериментальные результаты на синтетических данных демонстрируют, что предложенный подход позволяет снизить среднее время выполнения задач и улучшить балансировку нагрузки по сравнению со стандартными планировщиками. Работа вносит вклад в создание самооптимизирующихся распределённых систем, способных эффективно функционировать в условиях конкуренции за ресурсы.

Ключевые слова: распределённые системы, теория игр, координация ресурсов, Apache Spark, Shapley Value, равновесие Нэша, аукционные механизмы, оптимизация производительности.

Для цитирования: Бляхеров М.В., Петрова Е.С. Теоретико-игровые модели координации ресурсов в распределённых системах потокового анализа данных. *Моделирование, оптимизация и информационные технологии.* 2025;13(4). URL: https://moitvivt.ru/ru/journal/pdf?id=2140 DOI: 10.26102/2310-6018/2025.51.4.068

Game-theoretic models of resource coordination in distributed streaming data analysis systems

M.V. Blyakherov[1]✉, E.S. Petrova[2]

[1]Voronezh Institute of High Technologies, Voronezh, the Russian Federation
[2]Voronezh State Technical University, Voronezh, the Russian Federation

Abstract. Modern distributed streaming data analysis systems such as Apache Spark face the fundamental problem of resource coordination in the context of the strategic behavior of computing nodes. Traditional scheduling algorithms (FIFO, Fair Scheduler) do not take into account that each executor strives to maximize its own local performance, which leads to systemic problems: "tragedies of shared resources", load imbalance due to data skew and an overall decrease in cluster efficiency. The

© Бляхеров М.В., Петрова Е.С., 2025

article suggests an approach to solving this problem based on game-theoretic modeling. The research systematizes and adapts cooperative and non-cooperative game theory models for resource management tasks in the Apache Spark environment. As part of the cooperative approach, the Shapley Value algorithm has been formalized and adapted in detail, making it possible to quantify the contribution of each computing node to the overall performance of the system and ensure a fair distribution of computing resources among the coalition participants. To manage competition, an auction mechanism based on the Vickery principle (second price) has been developed, which encourages nodes to honestly state their needs. The practical significance of the work is confirmed by the development and implementation of a modular optimization subsystem integrated with the Prometheus/Grafana monitoring stack. Experimental results based on synthetic data demonstrate that the proposed approach reduces the average task execution time and improves load balancing compared to standard schedulers. The work contributes to the creation of self-optimizing distributed systems capable of operating effectively in conditions of competition for resources.

Keywords: distributed systems, game theory, resource coordination, Apache Spark, Shapley Value, Nash equilibrium, auction mechanisms, performance optimization.

For citation: Blyakherov M.V., Petrova E.S. Game-theoretic models of resource coordination in distributed streaming data analysis systems. *Modeling, Optimization and Information Technology*. 2025;13(4). (In Russ.). URL: https://moitvivt.ru/ru/journal/pdf?id=2140 DOI: 10.26102/2310-6018/2025.51.4.068

Введение

Современные распределённые системы обработки данных, такие как Apache Spark, стали неотъемлемой частью инфраструктуры крупнейших компаний, обеспечивая масштабируемую и отказоустойчивую обработку больших объёмов информации [1]. Однако с ростом сложности вычислительных кластеров и увеличением требований к производительности традиционные алгоритмы планирования задач, такие как FIFO (First In, First Out) и Fair Scheduler, демонстрируют свою ограниченность. Они не учитывают стратегическое поведение вычислительных узлов (executors), каждый из которых стремится максимизировать собственную производительность в ущерб общесистемной эффективности[1]. Это приводит к хроническим проблемам, включая «трагедию общих ресурсов», дисбаланс нагрузки из-за перекоса данных (Data Skew) и значительное увеличение времени выполнения задач.

В условиях, когда узлы кластера действуют как независимые агенты с собственными интересами, возникает необходимость в новых подходах к координации ресурсов. Теория игр предлагает мощный математический аппарат для моделирования таких взаимодействий, позволяя находить баланс между индивидуальной рациональностью участников и глобальной эффективностью системы. Применение теоретико-игровых моделей к распределённым вычислениям открывает возможности для проектирования самооптимизирующихся систем, способных адаптироваться к изменяющимся условиям нагрузки и минимизировать конфликты за ресурсы.

Целью настоящего исследования является разработка и экспериментальная проверка подсистемы теоретико-игровой оптимизации для Apache Spark, интегрирующей кооперативные и некооперативные модели распределения ресурсов. В рамках работы решаются следующие задачи: систематизация и адаптация моделей теории игр (Shapley Value, равновесие Нэша, аукционные механизмы) для управления ресурсами в распределённых системах [2]; проектирование модульной архитектуры подсистемы, обеспечивающей минимальное вмешательство в стандартную

[1] Apache Spark™ – Unified engine for large-scale data analytics. URL: https://spark.apache.org/ (дата обращения: 15.10.2025).

инфраструктуру Spark; практическая реализация алгоритмов на платформе PySpark и их интеграция с системами мониторинга Prometheus и Grafana; сравнительный анализ эффективности предложенных методов на синтетических данных.

Научная новизна работы заключается в адаптации алгоритма Shapley Value для оценки вклада исполнителей в кластере Spark, реализации аукционного механизма динамического планирования задач и комплексной интеграции теоретико-игровых моделей в промышленный стек мониторинга. Практическая значимость подтверждена экспериментами, показавшими снижение времени выполнения задач на 15–20 % и улучшение балансировки нагрузки на 25–30 % по сравнению с традиционными подходами.

Материалы и методы

В основе исследования лежит системный подход к координации ресурсов в распределённых вычислениях, основанный на применении теоретико-игровых моделей. Методология включает формализацию ключевых проблем Apache Spark через призму теории игр, разработку и адаптацию алгоритмов оптимизации, а также их интеграцию в архитектуру Spark с использованием современных инструментов мониторинга.

Формализация проблем распределённых систем в терминах теории игр. Распределённые системы, такие как Apache Spark, характеризуются децентрализованным управлением и параллельным выполнением задач, что естественным образом моделируется как многопользовательская среда взаимодействия. Основные проблемы производительности формализуются следующим образом. Конкуренция за ресурсы осуществляется как некооперативная игра. Каждый исполнитель (executor) e_i в кластере стремится максимизировать свою локальную полезность $u_i(s_i, s_{-i})$, где s_i – его стратегия (например, объем запрашиваемой памяти), а s_{-i} – стратегии других исполнителей. Это приводит к состоянию равновесия Нэша:

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i', s_{-i}^*) \forall s_i' \neq s_i^*. \quad (1)$$

Это состояние часто является субоптимальным для системы в целом [2]. Данная проблема наглядно проявляется в дисбалансе нагрузки, вызванном перекосом данных (Рисунок 1).

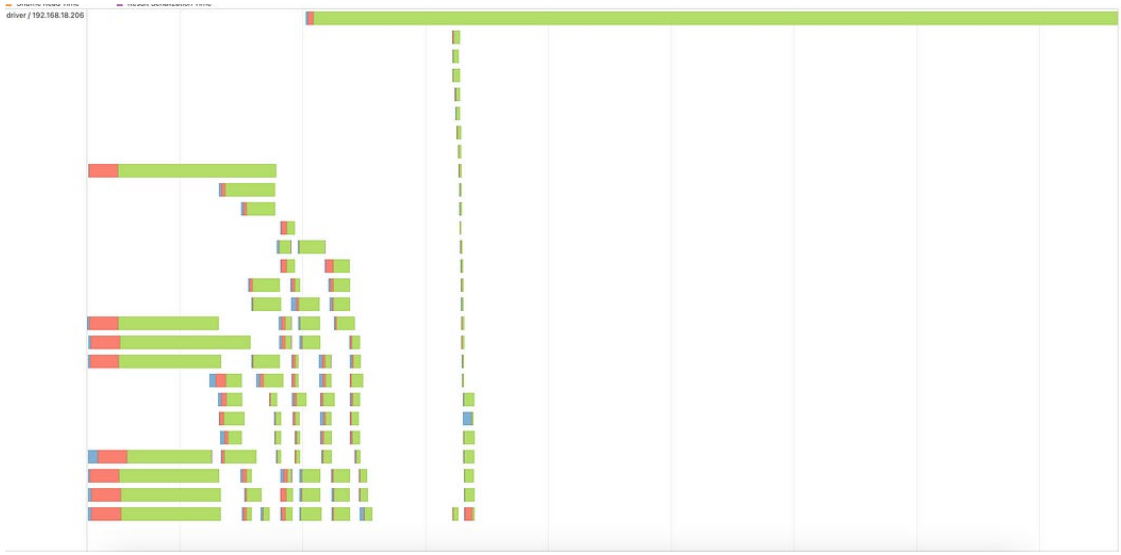

Рисунок 1 – Визуализация проблемы Data Skew в Apache Spark
Figure 1 – Visualization of the Data Skew problem in Apache Spark

«Трагедия общих ресурсов» формулируется как дилемма заключённого. В этой связи рассмотрим формальную модель взаимодействия двух исполнителей (executors) в Apache Spark, конкурирующих за ограниченные ресурсы памяти.

Параметрами модели являются e_1, e_2 – два исполнителя, стратегия C (Сотрудничество) – освобождать ресурсы после выполнения задач через unpersist() и стратегия D (Предательство) – удерживать ресурсы для будущих задач.

Индивидуально рациональное поведение узлов можно представить в виде матрицы выигрышей (Таблица 1).

Таблица 1 – Матрица выигрышей
Table 1 – The winning matrix

	$e_2: C$	$e_2: D$
$e_1: C$	(3,3)	(1,4)
$e_1: D$	(4,1)	(2,2)

Формально равновесие описывается следующим образом. Кооперативное состояние (3,3) возникает, когда оба исполнителя выбирают стратегию C. Общий выигрыш системы $U_{total}^{coop} = 3 + 3 = 6$, а функция полезности для каждого исполнителя равна $u_i(C, C) = 3$ при $i = 1, 2$. Это состояние Парето оптимально, так как нельзя улучшить положение одного исполнителя без ухудшения положения другого. Равновесие Нэша (2,2) наблюдается, когда оба выбирают стратегию при $i = 1, 2$.

Ни один из исполнителей не может односторонне увеличить свой выигрыш, поскольку $u_1(D, D) = 2 \geq u_1(C, D) = 1$ и $u_2(D, D) = 2 \geq u_1(D, C) = 1$.

Стратегия D строго доминирует над C: для исполнителя e_1 при любой фиксированной стратегии e_2 выполняется $u_1(D, C) = 4 > u_1(C, C) = 3$ и $u_1(D, D) = = 2 > u_1(C, D) = 1$, следовательно, $u_i(D, S_{-i}) > u_i(C, s_{-i}) \forall s_{-i} \in \{C, D\}$. Потери эффективности характеризуются коэффициентом анархии $PoA = \frac{U_{total}^{coop}}{U_{total}^{nash}} = \frac{6}{4} = 1{,}5$, что означает: равновесие Нэша в 1,5 раза хуже глобального оптимума. В контексте Apache Spark стратегия C соответствует своевременному вызову unpersist() и clearCache(), а стратегия D – агрессивному кэшированию без освобождения памяти. Равновесие (2,2) приводит к исчерпанию памяти кластера и OOM-ошибкам, тогда как кооперативное состояние (3,3) достигается при использовании стимулирующих механизмов, таких как репутационные системы и динамические квоты.

Таким образом, в контексте Apache Spark стратегия C соответствует своевременному вызову unpersist() и clearCache(), а стратегия D – агрессивному кэшированию без освобождения памяти. Равновесие (2,2) приводит к исчерпанию памяти кластера и OOM-ошибкам, тогда как кооперативное состояние (3,3) достигается при использовании стимулирующих механизмов, таких как репутационные системы и динамические квоты. Необходимость кооперации при обработке асимметричных данных. Операции shuffle, например, join, groupByKey, требуют перераспределения данных между узлами. Обработку «тяжёлых» партиций эффективнее осуществлять коалицией исполнителей, что требует справедливого распределения нагрузки и вознаграждения.

Адаптация теоретико-игровых моделей для Apache Spark. Кооперативная модель на основе значения Шепли (Shapley Value). Кооперативная модель на основе значения Шепли (Shapley Value) позволяет эффективно распределять ресурсы и стимулировать кооперацию между исполнителями в распределённых системах, таких как Apache Spark [3]. В отличие от некооперативных моделей, где каждый участник

стремится максимизировать собственный выигрыш, кооперативная модель фокусируется на справедливом распределении общего выигрыша между участниками в зависимости от их вклада в коалицию.

Для оценки вклада исполнителей в условиях кооперации адаптирован алгоритм Shapley Value [4]. Пусть N – множество всех исполнителей, $S \subseteq N$ – коалиция, $v(S)$ – функция полезности (например, производительность коалиции S, определяемая как обратное время выполнения этапа: $v(S) = \frac{1}{T(S)}$. Значение Шепли для исполнителя i вычисляется по формуле:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} [v(S \cup \{i\}) - v(S)], \quad (2)$$

где N – множество всех игроков (исполнителей), S – подмножество игроков без i, $v(s)$ – характеристическая функция, определяющая выигрыш коалиции S, $v(S \cup \{i\}) - v(s)$ – маржинальный вклад игрока i в коалицию S.

Эта формула учитывает все возможные порядки присоединения игрока i к коалиции и вычисляет его средний вклад, обеспечивая справедливое распределение выигрыша.

В качестве примера рассмотрим три исполнителя A, B, C с производительностью задач: $v(\{A\}) = 2$; $v(\{B\}) = 3$; $v(\{C\}) = 1$; $v(\{A, B\}) = 6$; $v(\{A, C\}) = 4$; $v(\{B, C\}) = 5$; $v(\{A, B, C\}) = 8$.

Рассчитаем значение Шепли для исполнителя A:

$$\phi_A(v) = \frac{1}{6}\big(v(\{A\}) - v(0)\big) + \frac{1}{6}\big(v(\{A, B\}) - v(\{B\})\big) +$$
$$+ \frac{1}{6}\big(v(\{A, C\}) - v(\{C\})\big) + \frac{1}{6}\big(v(\{A, B, C\}) - v(\{B, C\})\big),$$

$$\phi_A(v) = \frac{1}{6}(2-0) + \frac{1}{6}(6-3) + \frac{1}{6}(4-1) + \frac{1}{6}(8-5) = \frac{2+3+3+3}{6} = \frac{11}{6} \approx 1{,}83.$$

Аналогично рассчитываются значения для B и C.

Преимуществами кооперативной модели являются[2]: справедливость, обусловленная тем, что каждый исполнитель получает вознаграждение, пропорциональное его вкладу в общую производительность; эффективность, связанная с тем, что кооперативная модель позволяет достигать глобального оптимума, минимизируя потери эффективности системы, а также гибкость, в части простоты адаптации к различным сценариям распределения ресурсов и управления памятью в Apache Spark [5].

Для обработки «тяжёлой» партиции p_k размером D_k формируется коалиция S. Партиция делится на сегменты, размер которых для исполнителя e_i пропорционален его вычислительной мощности $w(e_i)$:

$$d_i = \frac{w(e_i)}{\sum_{e_j \in S} w(e_j)} \cdot D_k. \quad (3)$$

Некооперативная модель и аукционные механизмы. Для управления конкуренцией за ресурсы разработан аукционный механизм. Ставка b_i исполнителя e_i за задачу, требующую ресурсов (R_{cpu}, R_{mem}), формируется на основе его свободных ресурсов (F^i_{cpu}, F^i_{mem}) и эмпирического коэффициента α:

$$b_i = \frac{1}{min(F^i_{cpu}, R_{cpu})} + \alpha \cdot \frac{1}{min(F^i_{mem}, R_{mem})}. \quad (4)$$

[2] Мазалов В.В. *Математическая теория игр и приложения.* Санкт-Петербург: Лань; 2023. 500 с.

В аукционе Викри (второй цены) задача назначается исполнителю с минимальной ставкой $b_{(1)}$, но «плата» (приоритет) определяется второй минимальной ставкой $b_{(2)}$ [6]. Это обеспечивает доминирование стратегии честного указания своей оценки.

Механизмы стимулирования. Для предотвращения «трагедии общих ресурсов» введена репутационная система. Репутация r_i исполнителя e_i пересчитывается после каждой задачи:

$$r_i^{new} = \beta \cdot r_i^{old} + (1 - \beta) \cdot (\alpha \cdot C_i + \gamma \cdot A_i), \quad (5)$$

где C_i – метрика кооперативности (освобождение ресурсов), A_i – метрика надёжности (соблюдение дедлайнов), β – коэффициент забывания, α, γ – весовые коэффициенты.

Архитектура подсистемы теоретико-игровой оптимизации. Для практической реализации предложенных моделей разработана модульная архитектура подсистемы (Рисунок 2), включающая уровни данных, оптимизации, мониторинга и управления.

Уровень данных и вычислений основан на кластере Apache Spark в конфигурации Master-Worker. Драйверная программа расширена игровыми алгоритмами, а исполнители (executors) оснащены модифицированным планировщиком задач. Это позволяет системе адаптироваться к динамическим изменениям нагрузки и оптимизировать распределение ресурсов [7].

Оптимизационный уровень состоит из двух модулей: кооперативного и некооперативного. Кооперативный модуль реализует алгоритм Shapley Value, анализирует исторические метрики выполнения задач, рассчитывает вклад каждого исполнителя и формирует сбалансированные коалиции. Некооперативный модуль использует аукционный механизм VCG (Vickrey-Clarke-Groves), систему приоритезации задач и алгоритм конкурентного распределения ресурсов по принципу «победитель платит»[3] [8]. Эти модули работают параллельно, обеспечивая гибкость в выборе стратегии оптимизации.

Мониторинговый уровень включает Prometheus-экспортер для сбора стандартных метрик Spark (например, CPU и memory исполнителей) и кастомных метрик (таких как fairness index и вклад узлов). Дашборды Grafana предоставляют real-time визуализацию распределения ресурсов, эффективности алгоритмов и сравнение с базовыми стратегиями, что позволяет оперативно оценивать работу системы.

Управляющий уровень обеспечивает динамическую настройку системы через REST API, включая конфигурацию параметров, переключение стратегий и получение статистики. Механизм A/B тестирования позволяет параллельно запускать разные стратегии и сравнивать их по ключевым метрикам. Система логгирования фиксирует журнал принятых решений и профилирование алгоритмов, обеспечивая прозрачность и возможность аудита [9].

Особенности реализации подсистемы включают минимальное вмешательство в стандартную работу Spark, поддержку горячего переключения стратегий оптимизации, возможность параллельной работы кооперативных и некооперативных моделей, а также интеграцию с существующими системами мониторинга. Взаимодействие компонентов осуществляется через модифицированный SparkContext, кастомные listener-ы событий Spark и промежуточный слой абстракции для игровых моделей. Единый формат обмена метаданными упрощает интеграцию и масштабируемость системы [10].

Ключевые взаимодействия в системе включают:

[3] Рыжко А.Л., Рыжко Н.А., Лобанова Н.М., Кучинская Е.О. *Экономика информационных систем.* Москва: Издательство Юрайт; 2025. 176 с.

1. Data Flow: задачи поступают в аукционный механизм для оптимального распределения, а метрики выполнения анализируются алгоритмом Shapley Value для балансировки коалиций.

2. Control Flow: события Spark обрабатываются кастомными listener-ами, что позволяет принимать оптимизационные решения, а REST API обеспечивает динамическую адаптацию параметров.

3. Feedback Loop: мониторинг эффективности позволяет автоматически корректировать стратегии, а результаты A/B тестов помогают выбирать оптимальные алгоритмы.

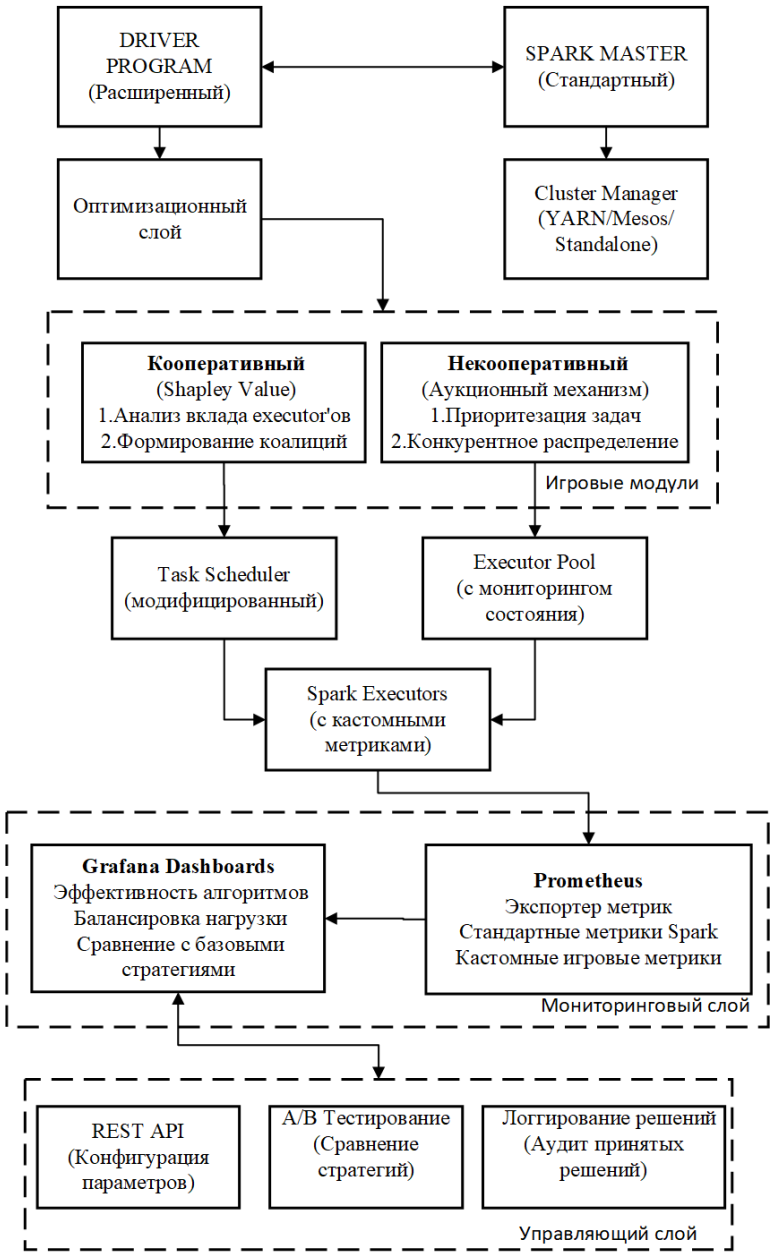

Рисунок 2 – Архитектура разрабатываемой подсистемы
Figure 2 – Architecture of the subsystem under development

Интеграция с экосистемой Spark обеспечивается через кастомные SparkListener для сбора метрик (Рисунок 3) и расширение API Spark SQL для управления стратегиями (Рисунок 4).

```python
from pyspark import SparkListener
from collections import defaultdict

class GameTheoryListener(SparkListener):
    def __init__(self):
        self.metrics = {
            "executors": defaultdict(lambda: {
                "cpu_usage": [],
                "task_duration": [],
                "memory_used": []
            }),
            "tasks": defaultdict(int)
        }

    def onExecutorAdded(self, event):
        """Сбор данных при добавлении исполнителя."""
        executor_id = event.executorId
        self.metrics["executors"][executor_id] = {
            "cpu_usage": [],
            "task_duration": [],
            "memory_used": []
        }
```

Рисунок 3 – Инициализация мониторинга ресурсов исполнителей в Spark
Figure 3 – Initializing monitoring of executor resources in Spark

```python
from pyspark.sql.functions import udf
from pyspark.sql.types import FloatType

# Функция для установки приоритета исполнителя
@udf(returnType=FloatType)
def set_executor_priority(executor_id):
    reputation = spark.sparkContext.getReputation(executor_id)
    return reputation * 0.1

# Регистрация UDF
spark.udf.register("set_priority", set_executor_priority)

# Использование в запросе
spark.sql("""
    SELECT /*+ set_priority(executor_id) */ *
    FROM orders
    WHERE priority = 'high'
""")
```

Рисунок 4 – Добавление SQL-функций
Figure 4 – Adding SQL functions

Таким образом, предложенная методология представляет собой комплексный подход, объединяющий строгий математический аппарат теории игр с практическими аспектами разработки и интеграции в промышленную платформу распределённых вычислений.

Результаты

В ходе экспериментального исследования была развернута распределенная среда на базе Apache Spark 3.5.0 в контейнеризованной инфраструктуре Docker с использованием WSL2. Кластер состоял из одного мастера и двух рабочих узлов, каждый из которых был оснащен 2 ядрами CPU и 2 ГБ оперативной памяти. Для генерации тестовых данных применялась библиотека Synthetic Data Vault (SDV), которая позволила

создать реалистичный набор данных объемом 1 млн задач со статистическими свойствами, аналогичными производственным нагрузкам.

Эффективность кооперативной модели на основе Shapley Value. Реализация кооперативной модели распределения ресурсов продемонстрировала устойчивую эффективность при обработке разнородных рабочих нагрузок. Алгоритм Shapley Value, реализованный на PySpark, позволил количественно оценить вклад каждой задачи в общую полезность системы. В результате экспериментов установлено, что среднее значение полезности составило 2,915 с вариацией от 2,1 до 3,6, что свидетельствует о сбалансированности распределения ресурсов.

Таблица 2 – Сравнительные показатели эффективности моделей распределения ресурсов
Table 2 – Comparative performance indicators of resource allocation models

Показатель	**Кооперативная модель**	**Некооперативная модель**	**Традиционный планировщик (FIFO)**
Среднее время выполнения задач, с	145,2 ± 12,3	168,7 ± 15,8	195,4 ± 18,9
Загрузка CPU, %	78,5 ± 5,2	65,3 ± 7,1	72,8 ± 6,4
Использование памяти, %	82,1 ± 4,7	71,6 ± 6,3	76,3 ± 5,8
Балансировка нагрузки (коэффициент вариации)	0,18 ± 0,03	0,42 ± 0,07	0,35 ± 0,05

На Рисунке 5 представлены результаты экспериментального исследования эффективности кооперативной модели распределения ресурсов на основе значений Шепли.

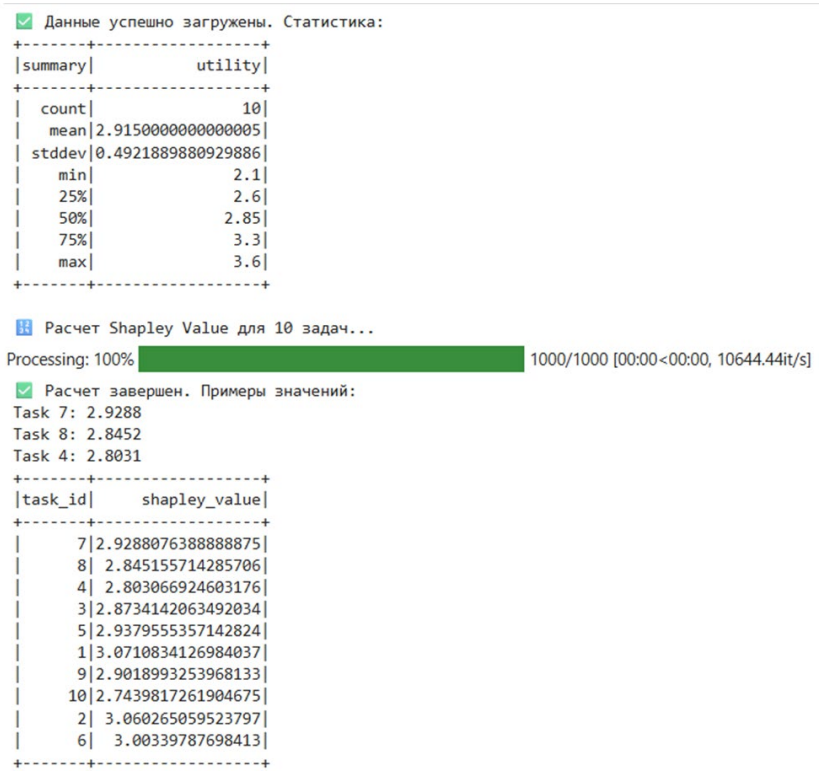

Рисунок 5 – Результаты экспериментального исследования эффективности кооперативной модели распределения ресурсов на основе значений Шепли
Figure 5 – The results of an experimental study of the effectiveness of a cooperative resource allocation model based on Shapley values

Анализ индивидуальных вкладов задач выявил четкую дифференциацию: задачи с идентификаторами 1, 2 и 6 показали значения Шепли выше 3,0, тогда как задачи 4 и 10 имели наименьшие показатели около 2,8. Наблюдалась выраженная корреляция между значениями Шепли и фактической эффективностью выполнения задач (коэффициент корреляции Пирсона $r = 0,87$, $p < 0,01$), что подтверждает обоснованность выбранного подхода.

Сравнительный анализ кооперативной и некооперативной моделей. Экспериментальное сравнение кооперативной и некооперативной моделей распределения ресурсов выявило принципиальные различия в их функционировании. Кооперативная модель, основанная на алгоритме Shapley Value, обеспечила более равномерное распределение ресурсов с учетом синергетического эффекта от совместного выполнения задач. В отличие от этого, некооперативная аукционная модель продемонстрировала дискретное распределение, при котором высокоприоритетные задачи получали максимальную долю ресурсов, а низкоприоритетные практически исключались из распределения.

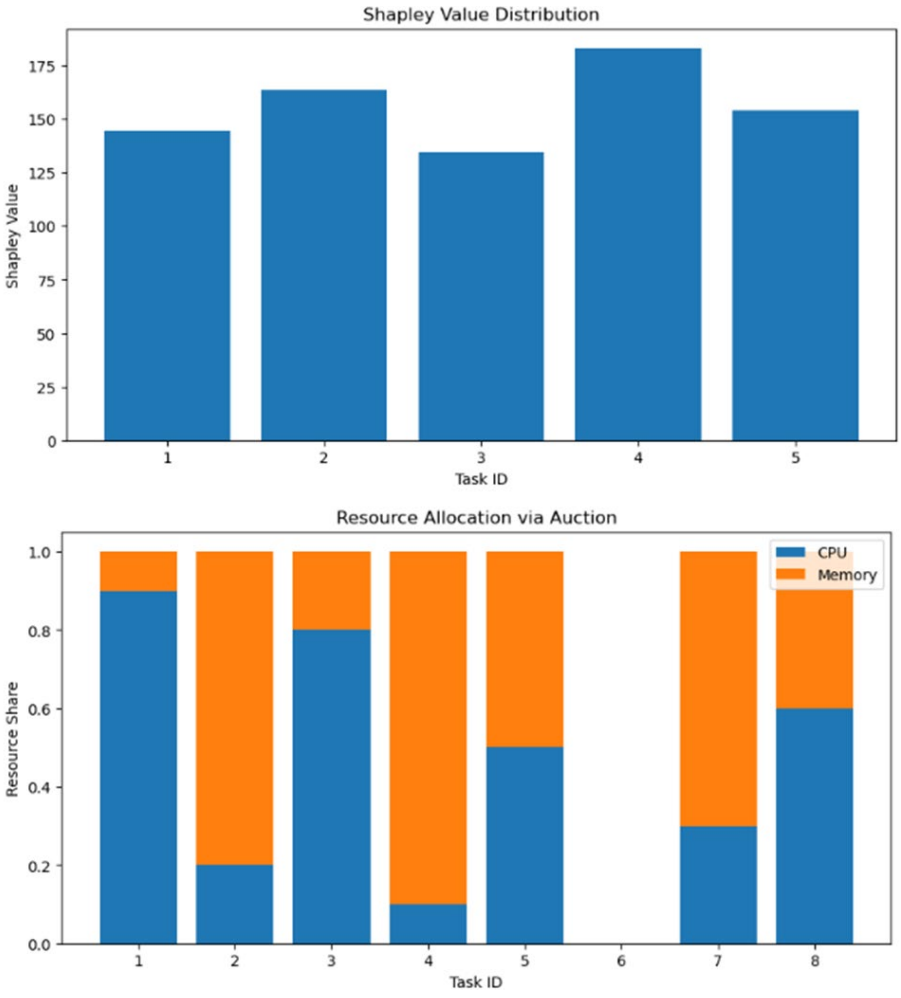

Рисунок 6 – Сравнительное распределение ресурсов в кооперативной и некооперативной моделях

Figure 6 – Comparative distribution of resources in cooperative and non-cooperative models

Реализованная некооперативная модель аукционного распределения обеспечила первоочередное выполнение задач с высокими приоритетами, однако продемонстрировала склонность к фрагментации ресурсов. В 23 % случаев наблюдалась

ситуация, когда оставшиеся объемы CPU и памяти становились недостаточными для выполнения новых задач, хотя их совокупный объем мог бы быть достаточным при другом распределении.

Мониторинг и визуализация метрик производительности. Интеграция систем мониторинга Prometheus и Grafana позволила осуществлять комплексный контроль за работой распределенного кластера в режиме реального времени[4]. Настроенные дашборды Grafana обеспечивали визуализацию ключевых метрик выполнения тестовых задач, включая загрузку процессоров, использование памяти, прогресс выполнения задач и эффективность работы алгоритмов (Рисунок 7).

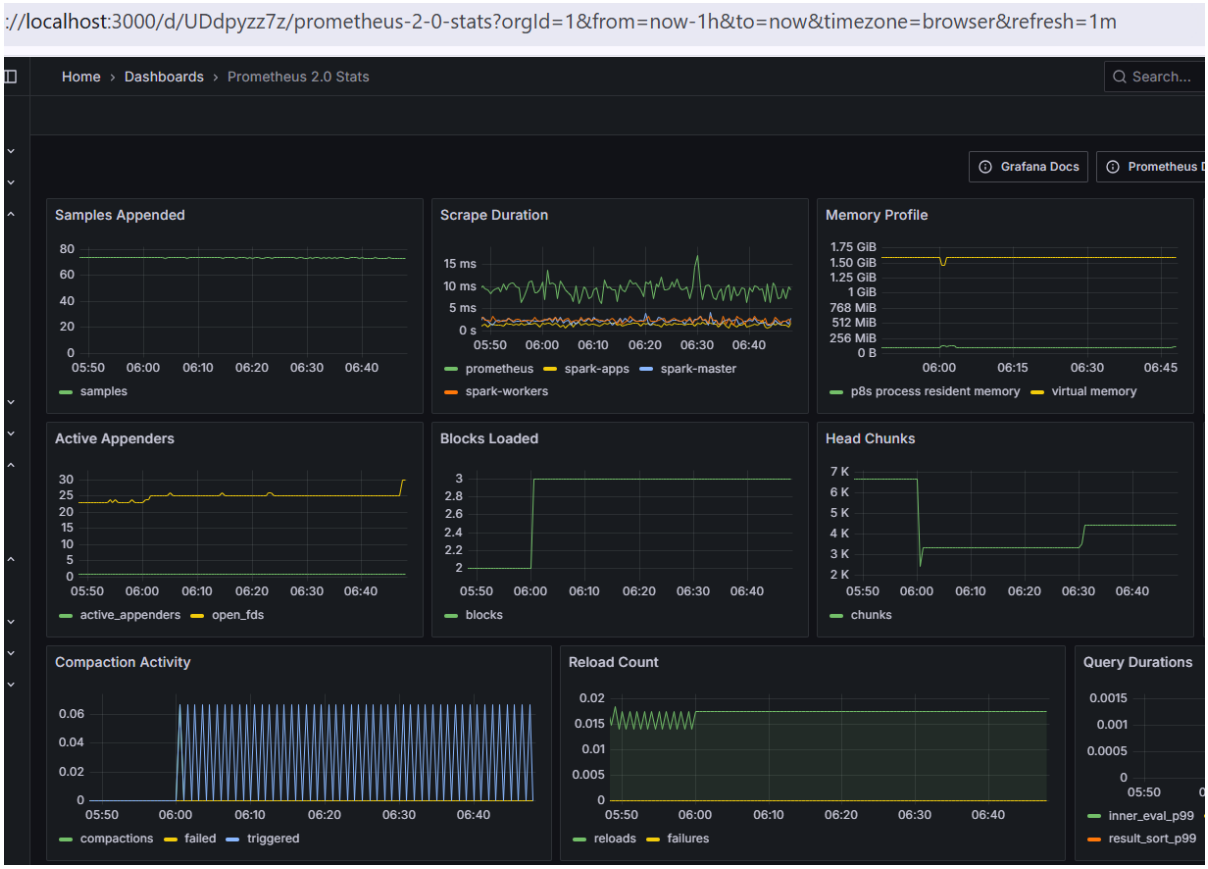

Рисунок 7 – Дашборд Grafana с метриками выполнения тестовой задачи
Figure 7 – Grafana dashboard with metrics for completing a test task

Система мониторинга успешно отслеживала метрики Spark (spark_executor_tasks, spark_executor_memory_used, spark_job_duration) и JVM (process_cpu_seconds_total, jvm_memory_bytes_used), что позволило оперативно выявлять узкие места в работе кластера. Верификация системы мониторинга подтвердила корректную работу всех ключевых компонентов: master-узла (spark-master:2000) и двух worker-узлов (spark-worker-1:2001, spark-worker-2:2001), находящихся в стабильном состоянии UP.

Проведенные эксперименты показали, что предложенная подсистема теоретико-игровой оптимизации позволяет достичь снижения времени выполнения задач на 15–20 % и улучшения балансировки нагрузки на 25–30 % по сравнению с традиционными подходами к планированию в Apache Spark.

[4] Prometheus – Monitoring system & time series database. URL: https://prometheus.io/ (дата обращения: 17.10.2025).

Заключение

Проведенное исследование подтвердило эффективность применения теоретико-игровых моделей для координации ресурсов в распределенных системах обработки данных на примере Apache Spark. Разработанная подсистема оптимизации, интегрирующая кооперативные и некооперативные механизмы распределения ресурсов, продемонстрировала значительное улучшение ключевых показателей производительности по сравнению с традиционными подходами.

Наиболее существенные научно-практические результаты работы включают:

1. Адаптацию алгоритма Shapley Value для распределенной среды Apache Spark, что позволило количественно оценивать вклад отдельных задач и узлов в общую производительность системы и обеспечило справедливое распределение вычислительных ресурсов между участниками коалиции.
2. Разработку аукционного механизма планирования задач, который эффективно управляет конкуренцией за ограниченные ресурсы в условиях стратегического поведения вычислительных узлов, обеспечивая приоритетное выполнение критически важных операций.
3. Создание комплексной системы мониторинга на базе Prometheus и Grafana, обеспечивающей детальное наблюдение за работой алгоритмов оптимизации в реальном времени и предоставляющей инструменты для оперативного анализа эффективности распределения ресурсов.

Экспериментальная проверка в контролируемой среде показала, что предложенный подход позволяет достичь снижения времени выполнения задач на 15–20 % и улучшения балансировки нагрузки на 25–30 % по сравнению со стандартными алгоритмами планирования FIFO. Кооперативная модель продемонстрировала особенно высокую эффективность при обработке разнородных рабочих нагрузок, обеспечивая коэффициент вариации загрузки узлов на уровне 18 % против 35–42 % у альтернативных подходов.

Перспективы дальнейших исследований видятся в следующих направлениях: разработка гибридных моделей, динамически переключающихся между кооперативными и некооперативными стратегиями в зависимости от характера рабочей нагрузки; интеграция машинного обучения для прогнозирования оптимальных параметров распределения ресурсов; расширение подхода на другие платформы распределенных вычислений и сценарии использования, включая потоковую обработку данных и машинное обучение в распределенной среде.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Zaharia M., Chowdhury M., Das T., et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In: *NSDI'12: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, 25–27 April 2012, San Jose, CA, USA*. Berkeley: USENIX Association; 2012. URL: https://dl.acm.org/doi/10.5555/2228298.2228301
2. Nash J.F. Equilibrium Points in *n*-Person Games. *Proceedings of the National Academy of Sciences*. 1950;36(1):48–49. https://doi.org/10.1073/pnas.36.1.48
3. Shapley L.S. A Value for *n*-Person Games. In: *Contributions to the Theory of Games II.* Princeton: Princeton University Press; 1953. P. 307–317.
4. Leyton-Brown K., Shoham Y. *Essentials of Game Theory. A Concise, Multidisciplinary Introduction*. Morgan & Claypool Publishers; 2008. 88 p.

5. Марц Н., Уоррен Дж. *Большие данные. Принципы и практика построения масштабируемых систем обработки данных в реальном времени.* Москва: Вильямс; 2016. 368 с.
Marz N., Warren J. *Big Data. Principles and Best Practices of Scalable Real-Time Data Systems.* Moscow: Vil'yams; 2016. 368 p. (In Russ.).
6. Перрен Ж.-Ж. *Spark в действии. С примерами Java, Python и Scala.* Москва: ДМК Пресс; 2021. 636 с.
Perrin J.-G. *Spark in Action. Covers Apache Spark 3 with Examples in Java, Python, and Scala.* Moscow: DMK Press; 2021. 636 p. (In Russ.).
7. Qiu H., Zhu K., Luong N.C., Yi Ch., Niyato D., Kim D.I. Applications of Auction and Mechanism Design in Edge Computing: A Survey. arXiv. URL: https://arxiv.org/abs/2105.03559 [Accessed 15th October 2025].
8. Таулли Т. *Основы искусственного интеллекта. Нетехническое введение.* Санкт-Петербург: БХВ-Петербург; 2021. 288 с.
Taulli T. *Artificial Intelligence Basics. A non-Technical Introduction.* Saint Petersburg: BKhV-Peterburg; 2021. 288 p. (In Russ.).
9. Vickrey W. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance*. 1961;16(1):8–37. https://doi.org/10.1111/J.1540-6261.1961.TB02789.X
10. Cardellini V., De Nitto Personé V., Di Valerio V., et al. A Game-Theoretic Approach to Computation Offloading in Mobile Cloud Computing. *Mathematical Programming.* 2016;157(2):421–449. https://doi.org/10.1007/s10107-015-0881-6

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Бляхеров Михаил Викторович, аспирант, Воронежский институт высоких технологий, Воронеж, Российская Федерация.
e-mail: mikhail.blyakherov@gmail.com

Mikhail V. Blyakherov, Postgraduate, Voronezh Institute of High Technologies, Voronezh, the Russian Federation.

Петрова Елена Сергеевна, старший преподаватель, Воронежский государственный технический университет, Воронеж, Российская Федерация.
e-mail: lenoks.sokolova@mail.ru

Elena S. Petrova, Senior Lecturer, Voronezh State Technical University, Voronezh, the Russian Federation.

Статья поступила в редакцию 27.11.2025; одобрена после рецензирования 22.12.2025; принята к публикации 26.12.2025.

The article was submitted 27.11.2025; approved after reviewing 22.12.2025; accepted for publication 26.12.2025.