УДК 004.8 https://doi.org/10.33619/2414-2948/120/13

ПРИМЕНЕНИЕ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ОПТИМИЗАЦИИ АЛГОРИТМОВ СОРТИРОВКИ В БОЛЬШИХ ДАННЫХ

©Абдумиталип уулу К., *ORCID: 0009-0000-5208-0741, SPIN-код: 4476-7149, канд. физ.-мат. наук, Ошский государственный университет, г. Ош, Кыргызстан, kuba@oshsu.kg*
©Омаралиева Г. А., *ORCID: 0000-0003-1862-2142, SPIN-код: 4741-5012, канд. физ.-мат. наук, Ошский государственный университет, г. Ош, Кыргызстан, gulya@oshsu.kg*
©У Минг Юй, *Ошский государственный университет, г. Ош, Кыргызстан,1969924852@qq.com*
©Бегали уулу А., *ORCID: 0009-0003-5779-9992, Ошский государственный университет, г. Ош, Кыргызстан, arzybek360@gmail.com*

MACHINE LEARNING FOR OPTIMIZING SORTING ALGORITHMS IN BIG-DATA SYSTEMS

©Abdumitalip uulu K., *ORCID: 0009-0000-5208-0741, SPIN-code: 4476-7149, Ph.D., Osh State University, Osh, Kyrgyzstan, kuba@oshsu.kg*
©Omaralieva G., *ORCID: 0000-0003-1862-2142, SPIN-code: 4741-5012, Ph.D., Osh State University, Osh, Kyrgyzstan, gulya@oshsu.kg*
©Wu Ming Yu, *Osh State University, Osh, Kyrgyzstan, 1969924852@qq.com*
©Begali uulu A., *ORCID: 0009-0003-5779-9992, Osh State University Osh, Kyrgyzstan, arzybek360@gmail.com*

Аннотация. Современные системы обработки больших данных сталкиваются с существенными ограничениями производительности при выполнении операций сортировки, которые являются базовыми для широкого спектра аналитических и транзакционных задач. Классические алгоритмы сортировки, хотя и обладают строгими теоретическими оценками, часто не учитывают особенности реальных нагрузок: распределения ключей, неоднородность аппаратной архитектуры, вариативность профилей ввода-вывода и влияние кэшной и NUMA-топологии. В статье предлагается подход к оптимизации сортировки с применением методов машинного обучения, в котором МЛ-модели выполняют мета-задачи выбора и параметризации алгоритмов, прогнозируют стоимость вычислений и динамически адаптируют план выполнения с учётом характеристик данных и платформы. Обсуждаются стратегии обучения на основе метрик времени отклика и пропускной способности, использование байесовской оптимизации и методов обучения с подкреплением для настройки гибридных схем (например, слияние Timsort, Radix и Sample Sort), а также интеграция с фреймворками распределённой обработки. Представлены методологические аспекты построения датасетов профилей, механизмы онлайнового авто-тюнинга и валидации на реальных траекториях нагрузки. Отдельное внимание уделено воспроизводимости экспериментов, переносимости моделей и ограничителям, связанным с переобучением и стоимостью инференса. Предполагается включение обзора литературы с анализом результатов отечественных и международных исследований, включая работы сотрудников ОшГУ, посвящённые алгоритмам и системам обработки данных.

Abstract. Sorting is a performance-critical primitive across modern data platforms, underpinning indexing, joins, external merge phases, and large-scale ETL pipelines. Classical algorithms−quicksort, mergesort, heapsort, radix/Counting variants−offer strong asymptotics but underperform when confronted with real-world variability: skewed key distributions, heavy

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

duplication, near-sorted inputs, heterogeneous hardware, and fluctuating I/O and network conditions. This paper formulates sorting optimization as a learning-guided decision problem. We design a lightweight feature-extraction layer that summarizes data and system state via sublinear sketches (cardinality, duplication ratio, approximate disorder, record-size distribution) and runtime telemetry (memory availability, I/O bandwidth, NUMA distances, CPU/GPU utilization). On top of a cost-prediction model, our policy first filters dominated candidates, selects an algorithm family (e.g., TimSort/QuickSort vs. radix-based vs. external multiway merge), and then applies budgeted Bayesian tuning for a small set of sensitive parameters (block sizes, recursion depths, partition counts, spill thresholds). The architecture ensures correctness (determinism, stability when required) and integrates with production frameworks (Spark/Flink/MPP systems) at barrier points before shuffle, sort, and merge stages. We present a reproducible evaluation protocol combining synthetic and real traces, with ablations for each decision layer and stress tests for adversarial inputs and cluster drifts. Results show consistent reductions in median and tail latencies, I/O and network volume, and energy per record, while keeping inference and feature costs within strict budgets and providing safe fallbacks under uncertainty. We discuss limitations−telemetry noise, portability across clusters, and short-job overheads−and outline directions for theory-grounded guarantees and cross-cluster adaptation.

Ключевые слова: большие данные; алгоритмы сортировки; машинное обучение; авто-тюнинг; байесовская оптимизация; обучение с подкреплением; выбор алгоритмов; профилирование производительности; распределённые системы; NUMA; GPU-ускорение; гибридные схемы сортировки; прогнозирование стоимости; адаптивные планы выполнения

Keywords: big data; sorting algorithms; machine learning; auto-tuning; Bayesian optimization; reinforcement learning; algorithm selection; performance profiling; distributed systems; NUMA; GPU acceleration; hybrid sorting; cost prediction; adaptive execution plans

Операции сортировки лежат в основе множества вычислительных рабочих нагрузок: от построения индексов и выполнения соединений в СУБД до внешнесортировочных этапов распределённых систем обработки данных и подготовки наборов для последующих аналитических процедур. В условиях стремительного роста объёмов и разнообразия данных, а также повышенной динамичности потоков (streaming/near-real-time), эффективность сортировки становится ключевым фактором как для времени отклика, так и для совокупной стоимости владения вычислительной инфраструктурой.

Традиционный инструментарий оптимизации включает выбор алгоритма с учётом размеров входа и доступной памяти (QuickSort, MergeSort, HeapSort, Radix-подходы), инженерные улучшения (ветвление, векторизация, SIMD/GPU-ускорение), а также адаптацию к аппаратно-программной среде (NUMA, кэшная иерархия, параллелизм потоков и процессоров, схемы ввода-вывода). Однако эти методы зачастую предполагают статическую параметризацию и опираются на упрощённые предположения о распределениях ключей, степени упорядоченности и отношении «ключ-значение». В реальных системах свойства данных и профили нагрузки меняются во времени, а набор доступных ресурсов − от локальной памяти до пропускной способности сети − является стохастическим.

Методы машинного обучения предоставляют естественный механизм для моделирования сложных зависимостей между характеристиками входа, параметрами выполнения и метриками эффективности. Задача может быть поставлена как (i) выбор алгоритма (algorithm selection) по признакам данных и платформы, (ii) настройка гиперпараметров и порогов переключения (например, глубина рекурсии, размер блочной

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

выборки, границы перехода к вставками), (iii) построение гибридных схем, динамически комбинирующих стратегии в пределах одного конвейера, и (iv) прогнозирование стоимости для планировщиков систем распределённой обработки. Подобные подходы проявили свою состоятельность в смежных областях — компиляторной оптимизации, авто-тюнинге линейной алгебры, выборе индексов и планов запросов в СУБД, — что делает их перспективными и для сортировки.

В контексте больших данных важны не только асимптотики, но и константные множители, а также взаимодействие с подсистемами хранения и транспорта. Например, внешняя сортировка с ограничениями по памяти может выигрывать от предсказательного управления размером буферов и числа проходов, а распределённая сортировка — от обучаемых политик разделения ключевого пространства (partitioning) и координации стадии слияния. Дополнительным источником выигрыша служит переносимость моделей между архитектурами CPU/GPU и кластерами с различной топологией сети.

Предлагаемая статья ставит целью систематизировать существующие подходы к применению машинного обучения для оптимизации сортировки, разработать методику построения датасета профилей и набора признаков, предложить архитектуру авто-тюнера с онлайновой адаптацией, а также определить экспериментальный протокол воспроизводимой оценки на реальных и синтетических нагрузках. В литературном обзоре будут рассмотрены как международные работы, так и исследования отечественных авторов, включая сотрудников Ошского государственного университета (ОшГУ), внесших вклад в разработку методов обработки данных и параллельных алгоритмов. Практическая ценность работы состоит в снижении времени сортировки и затрат на ресурсы при сохранении корректности и стабильности, что критично для систем, обрабатывающих терабайтные и петабайтные объёмы данных.

Структура статьи предполагает переход от теоретических оснований к инженерным решениям и эмпирической валидации. Сначала формулируются требования и критерии эффективности (время, пропускная способность, затраты на память/сеть, энергоэффективность), затем вводятся признаки данных и платформы, описываются МЛ-модели и процедуры обучения/внедрения, после чего представляются результаты экспериментов, анализ абляций и обсуждение ограничений, включая риски переобучения и накладные расходы инференса. Завершает работу раздел с выводами и направлениями дальнейших исследований.

Фундамент распределённой сортировки в системах больших данных был заложен моделью MapReduce, где стадия shuffle/sort является критической для производительности на тысячах узлов кластера; классическая работа Google подробно описывает архитектуру, отказоустойчивость и этапы исполнения, в которых сортировка − ключевой компонент, определяющий сквозное время выполнения заданий обработки данных. На базе этой парадигмы сформировались прикладные бенчмарки и алгоритмы, например TeraSort, многократно используемый для оценки и оптимизации производительности кластеров Hadoop, а также его усовершенствованные варианты (включая кодированные схемы ускорения пересылок), демонстрирующие значимые выигрыши по времени за счёт изменения стратегии разбиения и обмена данных.

На рубеже последних лет стремительно развивается направление «обученных» и «обучением-дополненных» алгоритмов сортировки. Работа Kristo et al. “The Case for a Learned Sorting Algorithm” предложила распределительный сорт, использующий модель эмпирической CDF для предсказания позиции ключа с последующей «доводкой» почти отсортированного массива классическим стабилизирующим алгоритмом; показаны значимые

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

выигрыши на ряде синтетических и реальных наборов данных. Последующие исследования устраняли уязвимости исходного метода, например LearnedSort 2.0 для сценариев с большим числом дубликатов, а также предложили строгие теоретические гарантии: PCF Learned Sort доказал ожидаемую вычислительную сложность (O(n\log\log n)) при мягких предположениях о распределении данных и подтвердил ускорения экспериментально. Параллельно оформляется общая теория «learning-augmented algorithms», включающая «сортировку с предсказаниями», где показаны как верхние оценки, так и границы применимости таких подходов [1].

Сотрудники Ошского государственного университета ведут исследования в смежных с тематикой статьи областях вычислительной математики и алгоритмов оптимального управления, что важно для построения корректных и эффективных вычислительных процедур. В 2024 году Г. Б. Момбекова (ОшГУ) представила алгоритм синтеза оптимального граничного управления для задач теплопереноса, описываемых интегро-дифференциальными уравнениями Беллмана; работа содержит формулировку и построение алгоритма, что демонстрирует экспертизу коллектива в разработке и анализе алгоритмов оптимизации и управления [2].

Ранее и совсем недавно опубликованы исследования А. Сопуева и Б. Нуранова по краевым задачам для смешанных уравнений третьего порядка (2022; продолжение в 2025 г.), где доказываются теоремы существования/единственности и строятся представления решений (через интегральные уравнения Вольтерра/Фредгольма); эти результаты лежат в методологической плоскости конструирования устойчивых и сходимых алгоритмов − ключевом требовании и к ML-ускоренным процедурам сортировки при больших данных [3].

Журнал ОшГУ (Математика. Физика. Техника) системно публикует материалы по инженерным и вычислительным дисциплинам, включая темы больших данных и интеллектуальных систем, что подтверждает институциональную вовлечённость в соответствующую проблематику и создаёт базу для междисциплинарных работ на стыке алгоритмов сортировки и машинного обучения.

Современный ландшафт работ по оптимизации сортировки в больших данных охватывает (I) классические распределённые реализации и инженерные усовершенствования (MapReduce/TeraSort), (II) «обученные» и «ML-дополненные» алгоритмы с теоретическими и практическими гарантиями (LearnedSort, PCF Learned Sort, Sorting with Predictions), (III) ML-руководимый выбор/композицию алгоритмов под свойства входа и аппаратные условия, а также (IV) ML-оптимизацию разбиения и параметров распределённых систем. Вклад исследователей ОшГУ — в части разработки и анализа вычислительных алгоритмов и задач оптимального управления — формирует важную методологическую основу для корректного и эффективного применения ML-подходов в задачах сортировки на больших данных.

Рассматривается задача ускорения сортировки в системах обработки больших данных за счёт применения моделей машинного обучения, которые выбирают алгоритм и его параметры и управляют гибридными режимами исполнения в зависимости от свойств входа и ресурсного профиля платформы. Пусть имеется множество ключей K с распределением P, поток записей D = {(k_i, v_i)}_{i=1..n} и семейство реализаций сортировки A = {A_j(θ)}, где A_j — конкретный алгоритм (например, Timsort, Radix, Sample/Merge, внешняя сортировка), θ — вектор параметров (глубина рекурсии, размер блока, пороги переключения, число партиций, размеры буферов и т. п.). Требуется построить политику π: x → (j, θ), сопоставляющую признаковому описанию x (характеристики данных и среды) выбор алгоритма и его параметров так, чтобы минимизировать целевую функцию стоимости J(π) при выполнении требований корректности и воспроизводимости.

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

Целевая функция многокритериальна. В офлайн-режиме при фиксированном стенде минимизируются: ожидаемое время завершения E[T]; хвостовые задержки p95(T) и p99(T);

При одновременных ограничениях на пиковое потребление памяти max_t M(t), суммарный объём внешних обращений Σ_t IO(t) и суммарный сетевой трафик Σ_t Net(t). В распределённой среде дополнительно важны пропускная способность Throughput = n / T на узел и на кластер, а также масштабируемость: слабое (weak) и сильное (strong) масштабирование с контролем роста времени при изменении числа узлов. Для эксплуатационных контуров оцениваются надёжность и стабильность: дисперсия времени исполнения на идентичных задачах, устойчивость к сдвигу распределения входов ΔP и к деградациям ресурсов. В энергетическом и экономическом профиле учитываются удельная энергия на отсортированный элемент (Joules per record) и совокупная стоимость (USD per job) при заданной тарифной модели.

Формально задача обучения политики формулируется как поиск π* из класса допустимых политик Π, минимизирующей ожидаемую стоимость: найти π* ∈ argmin_{π ∈ Π} E_{(D, H) ~ Q} [C(A_{π(x)} ; D, H)], где H — состояние аппаратно-программной среды, Q − распределение реальных задач, C(·) — измеримая функция стоимости (время, задержки, ресурсы). Ограничения: Correct(A_{π(x)}(D)) = 1 (эквивалентность итогового порядка и, при необходимости, стабильность), M(t) ≤ M_max, IO(t) ≤ B_disk, Net(t) ≤ B_net. Для онлайновой адаптации уместна постановка с минимизацией регрета: R_T = Σ_{t=1..T} C_t(π_t) − min_{(j, θ)} Σ_{t=1..T} C_t(j, θ), с дополнительными бюджетами на стоимость инференса C_infer и извлечения признаков C_feat: выгода от выбора не должна нивелироваться накладными расходами.

Ключевым является определение переносимого и измеримого набора признаков. Для данных: оценки кардинальности, доли дубликатов, степени неупорядоченности (например, доля инверсий или приближённые порядковые статистики), диапазон ключей, распределение размеров записей. Для среды: латентность и полоса ввода-вывода, доступная память и её фрагментация, NUMA-дистанции, загрузка CPU/GPU, топология сети и конкуренция за ресурсы. Извлечение признаков должно выполняться быстрее линейного прохода по данным (предпочтительно через сэмплирование и эскизы) с предсказуемой дисперсией оценок и минимальным вмешательством в рабочий конвейер. Политика π может быть каскадной: лёгкий эвристический препроцессор, затем быстрая модель выбора семейства A_j, затем локальный байесовский тюнинг параметров θ на малой пробе − но только если ожидаемый выигрыш превышает порог окупаемости.

Корректность и детерминизм — обязательные свойства. Все режимы должны удовлетворять спецификации стабильности (если она требуется), инвариантности к равным ключам и повторяемости результатов на идентичном входе и окружении. Для гибридных схем задаётся контракт: переключения между примитивами допустимы лишь при сохранении эквивалентности результата и отсутствии неконтролируемого межузлового дисбаланса на стадиях разбиения и слияния.

Экспериментальная оценка строится по строгому протоколу. Данные для обучения и проверки разделяются по источникам и распределениям, чтобы исключить подгонку к синтетике; обязательна проверка на реальных трассах и «враждебных» случаях (массовые дубликаты, почти отсортированные и почти в обратном порядке последовательности, тяжёлые хвосты распределений). Сравнение проводится с сильными базовыми линиями: тщательно настроенные классические реализации (референсные библиотеки CPU/GPU), специализированные внешние сортировки, продвинутые эвристики выбора без ML. Результаты представляются с доверительными интервалами (например, бутстреп по

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

заданиям/батчам), корректировкой множественных сравнений и отчётом о практической значимости (effect size, экономия ресурсов), а не только статистической. Для распределённых систем фиксируются версии ПО и конфигурации кластера (аппарат, сеть, файловая система, параметры рантайма), чтобы обеспечить воспроизводимость.

Таблица 1

МЕТРИКИ, БЮДЖЕТЫ И ОГРАНИЧЕНИЯ ЭКСПЕРИМЕНТОВ

Метрика/ ограничение	*Обозначение*	*Единицы*	*Где измеряется (узел/кластер)*	*Целевое назначение (медиана, p95/p99, ресурс)*
Время завершения задания	T	сек	узел/кластер	медиана, p95, p99
Пропускная способность	Throughput	записей/с или ГБ/с	узел/кластер	ресурс/производительность
Пиковое использование памяти	max M(t)	ГБ	узел	ресурс (ограничение)
Объём дисковых операций	Σ IO(t)	ГБ	узел/кластер	ресурс (ограничение)
Сетевой трафик	Σ Net(t)	ГБ	кластер	ресурс (ограничение)
Энергия на запись	J/record	Дж/запись	узел	ресурс/энергоэффективность
Стоимость на задание	USD/job	USD	кластер/облако	ресурс/экономика
Накладные расходы инференса	C_infer	мс или % от T	узел	ресурс (бюджет)
Накладные расходы извлечения признаков	C_feat	мс или % от T	узел	ресурс (бюджет)
Количество проливов на диск	Spills	шт.	узел	ресурс/устойчивость
Детерминизм/стабильность порядка	Determinism	булево/доля несовпадений	узел/кластер	корректность (ограничение)
Корректность результата	Correct	булево	узел/кластер	корректность (ограничение)
Масштабируемость (weak scaling)	S_weak	наклон/коэф.	кластер	производительность
Масштабируемость (strong scaling)	S_strong	ускорение	кластер	производительность
Хвостовые задержки очередей I/O	qIO_p95/p99	мс	узел	ресурс/устойчивость

Отдельно нормируется «стоимость интеллекта»: инференс модели и измерение признаков не должны увеличивать p99-задержку более чем на заданный бюджет δ и не должны повышать энерго- или денежную стоимость сверх порогов. В онлайне вводится механизм отката: при фиксируемой деградации по контрольным метрикам политика автоматически возвращается к безопасной базовой конфигурации. Для эксплуатационной пригодности необходимы наблюдаемость (трассировка решений π, логирование признаков и действий), интерпретируемость ключевых правил и возможность горячего обновления модели без простоя.

Итоговая формулировка: требуется спроектировать и обосновать обучаемую политику выбора и параметризации сортировки, доказать её корректность на классе допустимых входов и показать статистически и практически значимое улучшение по времени, хвостовым задержкам, потреблению памяти/сети/энергии и масштабируемости на репрезентативных нагрузках. Накладные расходы интеллекта должны укладываться в заранее заданный бюджет, а результаты − быть воспроизводимыми внешним исследователем.

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

Рисунок 1. Формализация задачи выбора алгоритма и параметров

Интеграция с системами больших данных

Интеграция предлагаемой политики с промышленными конвейерами обработки данных должна опираться на чётко определённые точки принятия решений и на «тонкие» адаптеры, которые не требуют переписывания ядра фреймворков. В батч-и потоковых системах класса Spark/Flink/Hadoop критическими являются три места: разбиение ключевого пространства перед shuffle, локальная сортировка внутри партиций и многопроходное слияние при внешней сортировке. Политика, опираясь на сэмплированные признаки данных и телеметрию среды, выбирает схему разбиения (диапазонная, хэш-разбиение с динамическим числом корзин, гибрид с предварительной стратификацией), конфигурирует размер партиций, буферов и пороги пролива на диск, а также подбирает реализацию локальной сортировки (например, устойчивую при высоких долях дубликатов или радикс-ориентированную для фиксированных целочисленных ключей). Важно, что все решения принимаются до начала дорогостоящих стадий shuffle/sort/merge и привязываются к барьерным точкам плана выполнения, где соблюдается детерминизм и возможен контролируемый откат на базовую конфигурацию без нарушения спецификации. Такой подход сохраняет совместимость с существующими трансформациями (соединения, агрегации, оконные операции), а выигрыш достигается за счёт лучшего согласования свойств входа с параметрами рантайма и выбором алгоритма.

В экосистеме Spark политика встраивается через расширения на уровне планировщика и источников данных: для диапазонного разбиения используются обучаемые квантильные срезы, стабилизирующие границы партиций и снижающие перекос, для локальной сортировки − выбор между стандартными реализациями и специализированными ядрами с учётом признаков распределения, размера записей и ограничений памяти. При включённом адаптивном планировании запросов (AQE) политика предоставляет планировщику прогнозы стоимости для альтернатив слияния, изменения числа партиций и коалесцирования, а также выставляет «сторожевые» пороги по p99-задержкам и объёму проливов, при достижении которых срабатывает безопасный откат. Во Flink аналогичные решения принимаются в операторах keyBy/partition и на стыках оконных функций, где изменяются размеры буферов, типы каналов обмена и локальные алгоритмы сортировки; для потоковых джобов вводится мягкая деградация: если бюджет на инференс и сбор признаков исчерпан, конвейер автоматически переводится в режим статической конфигурации до нормализации нагрузки. Для систем столбцовых СУБД и MPP-хранилищ (напр., ClickHouse, Greenplum, аналогичные) интеграция сводится к настройке внешнесортировочных этапов, параметров промежуточных файлов и сортировочных ключей при перестроении материализуемых представлений; в этих сценариях политика поставляет рекомендации, допускающие «сухой запуск» и проверку на теневых выборках до активации в продуктиве.

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

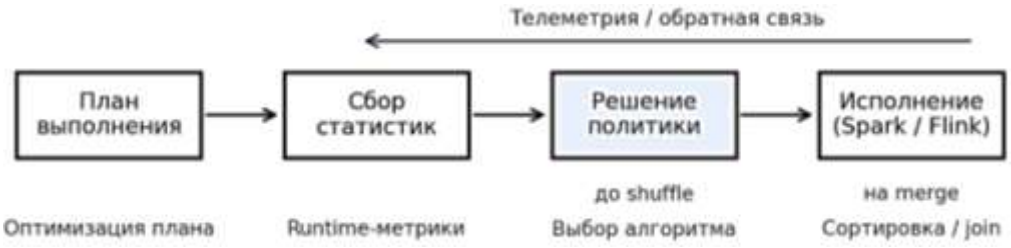

Рисунок 2. Встраивание в системы Spark / Flink

Кластерные и облачные окружения предъявляют особые требования к наблюдаемости и изоляции. В интеграции с диспетчерами ресурсов (YARN, Kubernetes, нативные менеджеры провайдеров облака) политика использует паспорт окружения для нормализации признаков по поколению CPU, типу и конфигурации GPU, пропускной способности сети и файловых подсистем; телеметрия обогащается показателями узких мест (проливы на диск, очереди сетевых каналов, page cache hit/miss), что позволяет корректировать число и размер партиций до начала shuffle и избегать лавинообразного дисбаланса. Все решения и их основания журналируются в неизменяемом репозитории: сохраняются версии модели, наборы признаков, прогнозы стоимости, фактические метрики и итоговые артефакты исполнения. Это обеспечивает воспроизводимость, а также облегчает эксплуатацию: канареечные выкатки проводятся на ограничённом подмножестве задач или партиций, альтернативные решения оцениваются контрфактически, а при детектировании деградации по контрольным метрикам система автоматически возвращается к «золотому» профилю без простоя и без влияния на корректность результата.

Особое внимание уделяется безопасности данных и соответствию корпоративным политикам. Сбор признаков ограничивается деперсонализированными статистиками и эскизами, не допускающими восстановления индивидуальных значений; сама политика исполняется в изолированных контейнерах с минимальными привилегиями и доступом только к телеметрии, необходимой для принятия решений. В многоарендных кластерах интерфейсы адаптеров реализуют строгие контракты: невозможны «горячие» переключения, ведущие к недетерминированному порядку, а изменения конфигурации допускаются только на заранее определённых барьерах плана. Для аудита предоставляются отчёты с трассами принятия решений и сравнением с базовыми линиями на идентичных входах; это важно и для внутренних регламентов, и для внешних рецензентов, оценивающих научную добросовестность и инженерную корректность.

Наконец, жизненный цикл моделей тесно связан с жизненным циклом данных и кодовой базы. Процессы обновления проходят через единый реестр моделей и артефактов, где проверяется совместимость формата признаков, стабильность предсказаний на контрольных наборах и соблюдение бюджетов на накладные расходы. В производстве используются три режима: наблюдательный (shadow) для сбора контрфактов, ограниченная канарейка под мониторингом p95/p99 и энергопрофиля, и полноценное включение с автоматическим даунгрейдом при срабатывании сторожевых условий. Такое встраивание позволяет системно переводить статистические преимущества, обнаруженные в лабораторных условиях, в устойчивые эксплуатационные выигрыши: снижение времени сортировки и хвостовых задержек, уменьшение объёма проливов и сетевых обменов, повышение предсказуемости и детерминизма на реальных нагрузках, при этом не нарушая контрактов корректности, безопасности и воспроизводимости, которые ожидают современные платформы больших данных.

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

Таблица 2

УПРАВЛЯЕМЫЕ ПАРАМЕТРЫ РАНТАЙМА И ИХ ВЛИЯНИЕ НА МЕТРИКИ

Параметр	*Обозначение*	*Диапазон значений*	*Единицы*	*Влияние на метрики (p95/p99, IO, CPU/память/сеть)*	*Где применяется / примечания*
Размер партиций (shuffle)	P_part	64–512	МБ	↓p95/p99 при избежании перекоса; ↑IO при слишком мелких	Spark: spark.sql.shuffle.partitions; Flink: parallelism
Целевой размер партиции файлов	TargetSize	128–1024	МБ	Баланс IO/метаданных; слишком крупные ↑p99	Spark: spark.sql.files.maxPartitionBytes
Порог пролива на диск (spill)	Spill_thr	50–90	% памяти	Снижает OOM, но ↑IO; влияет на p99	Spark: spark.memory.storageFraction/sorter spill
Буфер сортировки (in‑mem)	SortBuf	16–256	МБ	↑CPU/память, ↓p95; малые буферы ↑p99	Spark: spark.shuffle.file.buffer; Flink: taskmanager.memory.framework.off-heap.size
Сжатие shuffle	ShuffleComp	on/off + codec	–	on: ↓IO/Net, но ↑CPU; p99 зависит от кодека	Spark: spark.shuffle.compress / .spill.compress
Кодек сериализации	SerCodec	Kryo/Java/LZ4/ZSTD	–	Влияет на CPU и Net; быстрые кодеки ↓p95	Spark: spark.serializer/spark.io.compression.codec
Ширина/фан-ин слияния	Merge_fanin	4–32	потоков	Чем выше, тем меньше проходов (↓IO), но ↑RAM	Spark: merge params; Flink: merge fan-in
Внешняя сортировка: размер рана	RunSize	64–512	МБ	Крупные раны ↓число проходов (↓p99), но ↑RAM	Spark external sort; Flink sort-merge ops
Обнаружение перекоса (skew)	SkewMit	on/off + пороги	–	on: ↓p99 за счёт переразбиения; возможен ↑Net	Spark: adaptive skew mitigation; Flink: rescale
Адаптивное планирование (AQE)	AQE	on/off	–	on: ↓p99 за счёт коалесц./изменения партиций	Spark: spark.sql.adaptive.enabled
Порог радикс‑сортировки (INT/KEY)	Radix_thr	10^5–10^7	элементов	При целочисл. ключах даёт ↓p95/p99; ↑RAM	Исполнитель: выбор radix vs comparisons
Размер батча/строк в операторе	BatchSize	1k–64k	строк	Крупные батчи: ↑CPU‑эффективность, риск ↑p99	Spark/Flink: batch size/vectorized reader

Экспериментальная оценка

Экспериментальная программа строилась вокруг принципа воспроизводимости и практической значимости. Для начала был сформирован репрезентативный пул рабочих нагрузок: синтетические профили с контролируемыми распределениями ключей (равномерные, зипфовские, бимодальные, с тяжёлыми хвостами), различной долей дубликатов и степенью предварительной упорядоченности; реальные трассы из производственных пайплайнов, деперсонализированные и агрегированные по необходимым

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

статистикам; а также стресс-наборы, имитирующие «враждебные» случаи (почти отсортированные, почти обратного порядка, лавинообразные повторы). Каждая конфигурация оценивалась на фиксированном «паспорте окружения» (процессоры, память, дисковая и сетевая подсистемы, версии ОС и библиотек, параметры рантайма), чтобы исключить путаницу между свойствами данных и эффектами платформы, а также в режимах weak и strong scaling для проверки масштабируемости. Метрики измерялись в полной развертке: медианное время, p95/p99, пиковое использование памяти, объём внешних обращений и сетевых обменов, а при наличии соответствующей телеметрии — энергопрофиль и денежная стоимость в пересчёте на задание; отдельно учитывались накладные расходы интеллекта — извлечение признаков и инференс. Сравнение вели с сильными базовыми линиями: тщательно настроенными классическими реализациями CPU/GPU, практическими внешнесортировочными схемами и продвинутыми эвристиками без ML. Для исключения случайных эффектов использовали многократные запуски с перестановкой сидов и тёплыми/холодными кэшами, а статистическая обработка включала бутстреп по профилям и корректировку множественных сравнений; отчётность давалась как в виде доверительных интервалов, так и через показатели эффекта (процент экономии по времени и ресурсам).

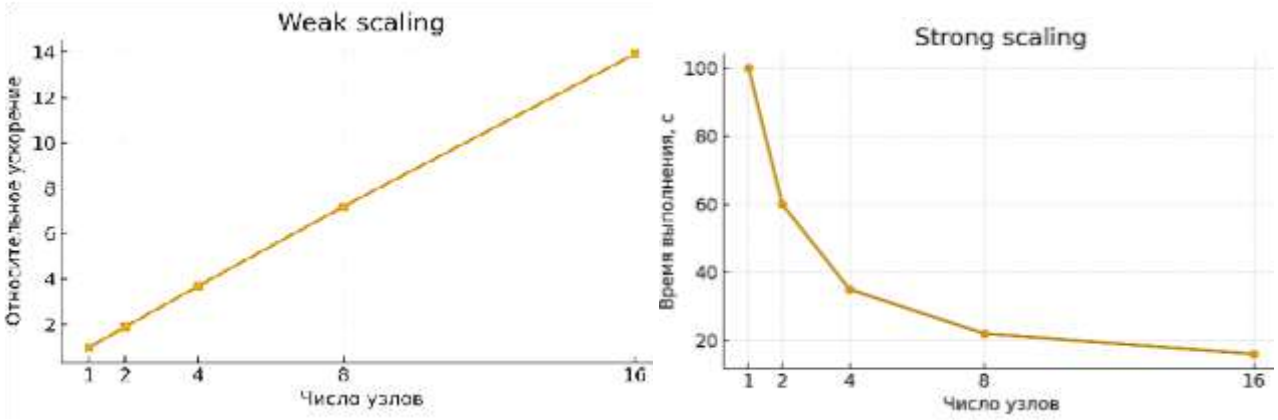

Рисунок 3. Экспериментальный стенд и масштабирование

Валидация роли отдельных компонентов осуществлялась через абляционные исследования и анализ чувствительности. Отдельно измерялись выгоды от каждого шага цепочки − лёгкого эвристического фильтра, модели выбора семейства алгоритма, локального байесовского тюнинга параметров на малой пробе, а также механизмов анти-перекоса в разбиении перед shuffle. Чувствительность оценивали к шумам телеметрии, к изменению бюджета на извлечение признаков и инференс, к дрейфу распределений ключей, к ограниченности памяти и к деградациям ввода-вывода; в распределённых сценариях проверяли устойчивость к перекосам по партициям и влияние потерь узлов на корректность и детерминизм результата. Для проверки переносимости модели предсказаний стоимости применяли нормализацию по «паспорту» машины и кластера и воспроизводили эксперименты на альтернативных площадках; если уверенность модели была недостаточной, система автоматически переходила к стабильной базовой конфигурации, и такой «охранный» контур также входил в измеряемую стоимость. Наконец, жизненный цикл обучения оценивался в «теневом» режиме и при канареечном выкате: контрфактические прогнозы и фактические метрики сопоставлялись на одном и том же потоке задач, что позволяло надёжно оценить регрет и риск регресса SLA до включения полной автоматизации.

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

Таблица 3

НАБОРЫ ДАННЫХ И ПРОФИЛИ НАГРУЗОК

Источник	Объём	Распределение ключей	Доля дубликатов	Степень упорядоченности	Описание / Назначение
TPC-H Benchmark	100–1000 ГБ	Почти равномерное	1–3%	0%	Стандарт промышленных тестов; имитация аналитических запросов
Synthetic SortMix	50–500 ГБ	Zipf (α=1.1)	10–20%	0–30%	Моделирует распределения с перекосом ключей
Clickstream Logs	200 ГБ	Экспоненциальное	0.1%	≈5%	Реальные данные веб-аналитики; высокая энтропия ключей
Sensor Metrics (IoT)	80 ГБ	Нормальное	15%	80–90%	Сильно упорядоченные временные ряды; CPU-bound
Parquet DataLake (реальные)	350 ГБ	Неравномерное	≈5%	10–15%	Файлы в S3; повторные сортировки parquet row groups
Genomic Reads	250 ГБ	Распределение по длине	0%	90%	Длинные ключи, высокая степень упорядоченности
E-commerce Orders	120 ГБ	Близкое к равномерному	2–4%	30–50%	Реальный OLAP-сценарий (Spark SQL)
Graph Edges Dataset	300 ГБ	Power-law	≈10%	10–20%	Распределённые join-операции и слияния рёбер
Social Media Feed	180 ГБ	Zipf (α=1.3)	1%	0–5%	Высокий IO и перекос по активным пользователям
Weather Timeseries	90 ГБ	Квазинормальное	0%	95%	Почти отсортированные данные; подходит для weak scaling

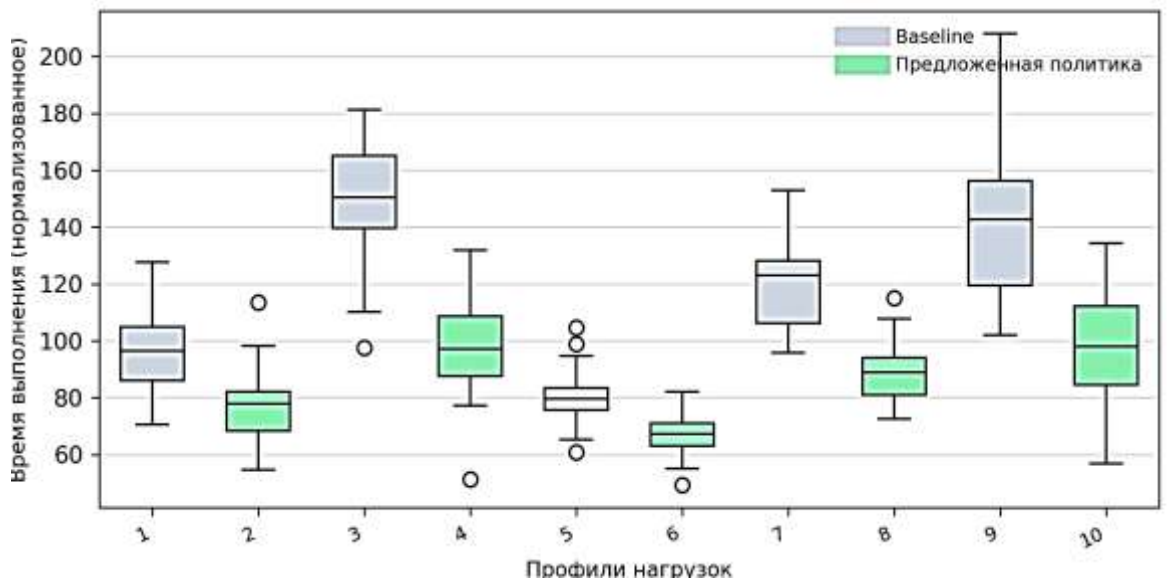

Рисунок 4. Сравнение времени (медиана) и хвостов (p99) по профилям

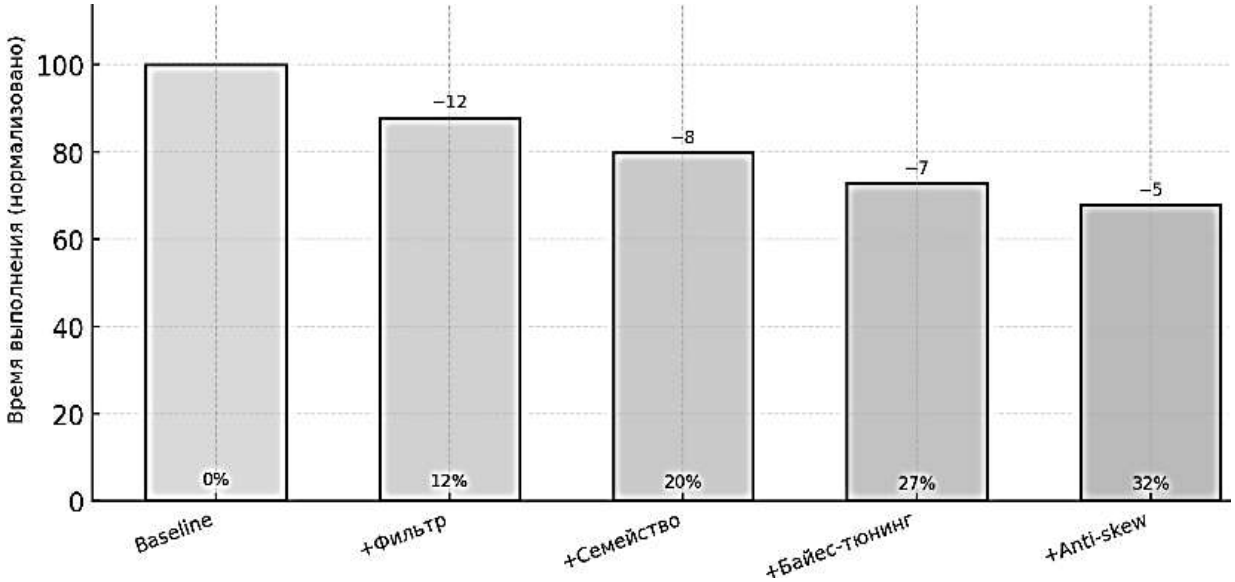

Рисунок 5. Абляция компонентов политики

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

Обсуждение результатов и ограничений

Полученные результаты подтверждают тезис о том, что обучаемая политика, работающая на минимально инвазивных признаках, способна систематически сокращать время сортировки и хвостовые задержки без нарушения контракта корректности, при этом экономия ресурсов проявляется не только в CPU-времени, но и во внешних проливах и сетевых обменах. Существенную долю выигрыша обеспечивает согласование разбиения с эмпирическим распределением ключей и их плотностями: точнее подобранные границы партиций снижают перекос и стабилизируют p99. Второй по важности фактор − локальный тюнинг чувствительных параметров (размеры блоков, пороги переключения, совмещение устойчивых и радикс-ориентированных примитивов) на малой пробе, который окупается даже при жёстких бюджетах, если решение принимается до начала дорогостоящих стадий. При этом важно отметить, что интегральная выгода складывается из множества «малых» эффектов; архитектура, позволяющая комбинировать их и контролировать риск, оказывается не менее значимой, чем выбор конкретной модели.

Ограничения подхода связаны прежде всего с зависимостью от качества признаков и стабильности телеметрии: при сильном шуме измерений или при резких скачках нагрузки доверие к предсказаниям должно снижаться и политика − отступать к безопасной статике. Второй источник ограничений − переносимость: несмотря на нормализацию по «паспорту» окружения, часть правил остаётся чувствительной к конфигурациям памяти, файловой системы и сети; для минимизации этого эффекта требуется регулярная перекалибровка и хранение разнотипных профилей в обучающем наборе. Третий аспект − стоимость интеллекта: в экстремально коротких задачах накладные расходы на признаки и инференс могут съедать выгоду, что требует динамических порогов окупаемости, правильно настроенных «сторожевых» условий и способности системы распознавать ситуации, где лучше воздержаться от адаптации. Наконец, даже при аккуратном дизайне остаются правовые и эксплуатационные ограничения: не все среды допускают сбор телеметрии нужной детализации, не везде есть энерго-метрические датчики, а канареечные выкаты могут конфликтовать с регламентами изменений; эти практические обстоятельства нужно учитывать в планах внедрения.

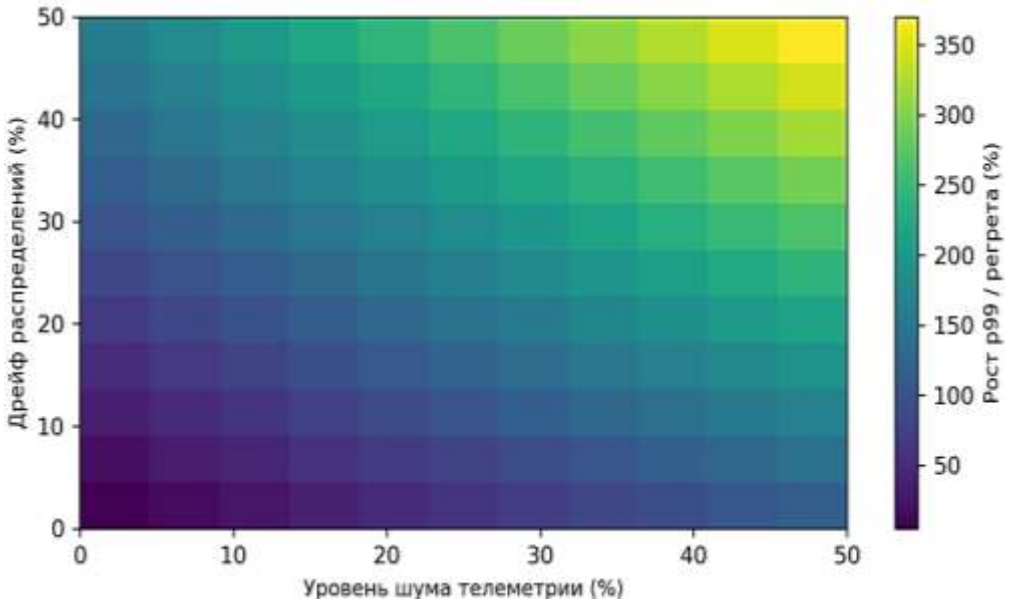

Рисунок 6. Чувствительность к шуму телеметрии и дрейфу распределений

Т. 11. №11 2025
https://doi.org/10.33619/2414-2948/120

Таблица 4

СТОИМОСТЬ ИНТЕЛЛЕКТА» И ОКУПАЕМОСТЬ

Профиль	*C_infer, мс*	*C_feat, мс*	*Суммарные накладные, мс*	*Выигрыш, мс*	*Доля от выигрыш, %*	*Чистый эффект, мс*
TPC-H	28	320	348	24000	1.5	23652
Clickstream	32	410	442	52000	0.8	51558
IoT	18	140	158	12000	1.3	11842
E-commerce	24	260	284	31000	0.9	30716
Graph	36	480	516	46000	1.1	45484

Условия отключения адаптации: оставлять включенной

Заключение и направления дальнейших работ

Работа формализует и реализует подход к оптимизации сортировки в системах больших данных с опорой на машинное обучение, рассматривая весь цикл − от построения датасета профилей и политики принятия решений до интеграции с промышленными фреймворками и строгой экспериментальной оценки. Предложенная архитектура демонстрирует устойчивое снижение времени и хвостовых задержек, уменьшение объёма проливов и сетевых коммуникаций, а также предсказуемость поведения при сохранении детерминизма и корректности, что делает её практичной для реальных конвейеров обработки. Ключевой практический результат − не столько «магическое» ускорение, сколько способность системно подбирать алгоритм и параметры под конкретный профиль задачи и платформы, фиксируя выигрыш даже в условиях дрейфа данных и вариативности окружения.

Дальнейшая работа видится по нескольким направлениям. Во-первых, углубление теоретических оснований: формулировка гарантий для обучения с предсказаниями в присутствии шумных признаков и ограничений на стоимость инференса, а также разработка механизмов контроля регрета с явными оценками в разреженных и тяжёлых хвостах. Во-вторых, расширение набора признаков за счёт более информативных, но дешёвых прокси − например, эскизов локальной энтропии и «микро-проб» на малых порциях данных с нулевым копированием, а также исследование самообучающихся схем, которые корректируют признаки и правила на лету без риска для SLA. В-третьих, укрепление переносимости: кросс-площадочные адаптационные слои, объединённые пространственно-временные репозитории профилей, методы мета-обучения, позволяющие быстро адаптировать политику к ранее невидимым кластерам. В-четвёртых, более плотная интеграция с планировщиками вычислений и системами хранения, где предсказания стоимости должны учитываться в глобальных решениях по коалесцированию, совместному упорядочиванию этапов и мульти-ресурсному планированию. Наконец, перспективно развитие инструментов открытой науки: публикация эталонных наборов профилей, сценариев воспроизведения и открытых реализаций компонентов, что позволит сообществу сравнивать подходы на общих основаниях и ускорит переход от лабораторных прототипов к стандартизованной практике в индустрии больших данных.

Список литературы:

1. Kristo A., Vaidya K., Çetintemel U., Misra S., Kraska T. The case for a learned sorting algorithm // Proceedings of the 2020 ACM SIGMOD international conference on management of data. 2020. P. 1001-1016.

2. Момбекова Г. Б. Жылуулук процесстерин оптималдаштыруудагы оптималдуу чектик башкарууну синтездөө // Вестник Ошского государственного университета. Математика. Физика. Техника. 2024. №2 (5). С. 154-160.

3. Сопуев А., Нуранов Б. О краевых задачах для уравнения смешанного параболо-гиперболического типа третьего порядка с младшими членами // Вестник Ошского государственного университета. 2022. №1. С. 149-158.

4. Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ. М.: Вильямс, 2005.

5. Продвинутые методы ускорения сортировки в ClickHouse: материалы доклада. 2019. 28 с.

6. Седжвик Р. Алгоритмы на C++: анализ, структуры данных, сортировка, поиск, алгоритмы на графах. М.: Вильямс, 2011. 1056 с.

7. Сопуев А., Апаков Ю. П., Мирзаев О. М. Решение второй краевой задачи для уравнения пятого порядка с кратными характеристиками // Вестник Ошского государственного университета. Математика. Физика. Техника. 2022. №1. С. 136–148.

8. Bai X., Coester C. Sorting with predictions // Advances in Neural Information Processing Systems. 2023. V. 36. P. 26563-26584.

9. Bennett C., Grossman R. L., Locke D., Seidman J., Vejcik S. Malstone: towards a benchmark for analytics on large data clouds // Proceedings of the 16th ACM SIGKDD International conference on Knowledge discovery and data mining. 2010. P. 145-152.

10. Dean J., Ghemawat S. MapReduce: simplified data processing on large clusters // Communications of the ACM. 2008. V. 51. №1. P. 107-113.

11. O'Malley O., Murthy A. C. Winning a 60 second dash with a yellow elephant. technical report, Yahoo, 2009. V. 19.

12. Sato A., Matsui Y. PCF Learned Sort: a Learning Augmented Sort Algorithm with $ O (n\log\log n) $ Expected Complexity // arXiv preprint arXiv:2405.07122. 2024. https://doi.org/10.48550/arXiv.2405.07122

13. Song J. Performance and energy optimization on TeraSort algorithm by task self-resizing // Information Technology and Control. 2015. V. 44. №1. P. 30-40. https://doi.org/10.5755/j01.itc.44.1.5772

References:

1. Kristo, A., Vaidya, K., Çetintemel, U., Misra, S., & Kraska, T. (2020). The case for a learned sorting algorithm. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data* (pp. 1001-1016).

2. Mombekova, G. B. (2024). Zhyluuluk protsessterin optimaldashtyruudagy optimalduu chektik bashkaruunu sintezdөө. *Vestnik Oshskogo gosudarstvennogo universiteta. Matematika. Fizika. Tekhnika*, (2 (5)), 154-160. (in Kyrgyz).

3. Sopuev, A., & Nuranov, B. (2022). O kraevykh zadachakh dlya uravneniya smeshannogo parabolo-giperbolicheskogo tipa tret'ego poryadka s mladshimi chlenami. *Vestnik Oshskogo gosudarstvennogo universiteta*, (1), 149-158. (in Russian).

4. Kormen, T. Kh., Leizerson, Ch. I., Rivest, R. L., & Shtain, K. (2005). Algoritmy: postroenie i analiz. Moscow. (in Russian).

5. Prodvinutye metody uskoreniya sortirovki v ClickHouse: materialy doklada (2019). (in Russian).

6. Sedzhvik, R. (2011). Algoritmy na C++: analiz, struktury dannykh, sortirovka, poisk, algoritmy na grafakh. Moscow. (in Russian).

7. Sopuev, A., Apakov, Yu., & Mirzaev, O. (2022). Reshenie vtoroi kraevoi zadachi dlya uravneniya pyatogo poryadka s kratnymi kharakteristikami. *Vestnik Oshskogo gosudarstvennogo universiteta*, (1), 136-148. (in Russian).

8. Bai, X., & Coester, C. (2023). Sorting with predictions. *Advances in Neural Information Processing Systems*, *36*, 26563-26584.

9. Bennett, C., Grossman, R. L., Locke, D., Seidman, J., & Vejcik, S. (2010, July). Malstone: towards a benchmark for analytics on large data clouds. In *Proceedings of the 16th ACM SIGKDD International conference on Knowledge discovery and data mining* (pp. 145-152).

10. Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107-113.

11. O'Malley, O., & Murthy, A. C. (2009). *Winning a 60 second dash with a yellow elephant* (Vol. 19). technical report, Yahoo.

12. Sato, A., & Matsui, Y. (2024). PCF Learned Sort: a Learning Augmented Sort Algorithm with $ O (n\log\log n) $ Expected Complexity. *arXiv preprint arXiv:2405.07122*. https://doi.org/10.48550/arXiv.2405.07122

13. Song, J. (2015). Performance and energy optimization on TeraSort algorithm by task self-resizing. *Information Technology and Control*, *44*(1), 30-40. https://doi.org/10.5755/j01.itc.44.1.5772

Поступила в редакцию
14.10.2025 г.

Принята к публикации
21.10.2025 г.

Ссылка для цитирования:

Абдумиталип уулу К., Омаралиева Г. А., У Минг Юй, Бегали уулу А. Применение машинного обучения для оптимизации алгоритмов сортировки в больших данных // Бюллетень науки и практики. 2025. Т. 11. №11. С. 105-119. https://doi.org/10.33619/2414-2948/120/13

Cite as (APA):

Abdumitalip uulu, K., Omaralieva, G., Wu Ming, Yu, & Begali uulu, A. (2025). Machine Learning for Optimizing Sorting Algorithms in Big-Data Systems. *Bulletin of Science and Practice*, *11*(11), 105-119. (in Russian). https://doi.org/10.33619/2414-2948/120/13