RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ПРИКЛАДНЫЕ ПРОГРАММНЫЕ СИСТЕМЫ

УДК 004.415
10.25209/2079-3316-2025-16-4-3-21

Проектирование архитектуры программного обеспечения для анализа цифрового следа в образовательных системах

Михаил Андреевич **Степанов**[1]✉, Сергей Александрович **Сластников**[2], Никита Андреевич **Климин**[3]
[1–3] НИУ ВШЭ, Москва, Россия
[1]✉*mastepanov@hse.ru*

Аннотация. Современные образовательные платформы и инструменты электронного обучения генерируют большие объемы данных о действиях учащихся – так называемый *цифровой след*. Анализ этих данных позволяет отслеживать прогресс студентов, выявлять риски отчисления и прогнозировать академическую успеваемость. Однако многие учебные организации не располагают ИТ-инфраструктурой, необходимой для комплексного сбора, хранения и анализа цифрового следа. Существующие решения часто фрагментарны, ориентированы лишь на ретроспективный анализ или требуют ресурсоемких технологий больших данных, что затрудняет их применение в образовательных организациях.

В данной работе предложена архитектура программной системы для интеграции и анализа цифрового следа в образовательных системах, учитывающая эти недостатки. Описаны подходы к сбору данных через адаптеры, унификации и хранению разнородных данных, а также инфраструктурные решения, обеспечивающие надежность и масштабируемость системы (микросервисная архитектура, контейнеризация, CI/CD, мониторинг). Приведен экспериментальный анализ эффективности предложенного подхода на примере расчета показателя успеваемости, демонстрирующий снижение времени обработки и использования ресурсов по сравнению с традиционными подходами. Предложенные решения позволяют создавать гибкие и производительные системы учебной аналитики, пригодные для внедрения даже в условиях ограниченных ресурсов.

Ключевые слова и фразы: цифровой след; цифровой профиль; архитектура программных систем; микросервисы; инфраструктура; учебная аналитика

Для цитирования: Степанов М. А., Сластников С. А., Климин Н. А. *Проектирование архитектуры программного обеспечения для анализа цифрового следа в образовательных системах* // Программные системы: теория и приложения. 2025. **Т. 16**. № 4(67). С. 3–21. https://psta.psiras.ru/read/psta2025_4_3-21.pdf

Введение

Современные образовательные платформы и цифровые инструменты генерируют большие объемы данных о действиях обучающихся, формируя их цифровой след. Эти данные могут использоваться для прогнозирования успеваемости, выявления рисков отчисления и персонализации обучения, однако в большинстве учебных заведений подобная информация остается недостаточно востребованной из-за сложности интеграции в ИТ-инфраструктуру и высоких требований к вычислительным ресурсам [1]. В этой связи особую актуальность приобретает создание эффективной программной инфраструктуры для анализа цифрового следа, способной работать в условиях ограниченных ресурсов [2].

Целью данной работы является исследование подходов, разработка и экспериментальная верификация архитектуры системы сбора, хранения и анализа цифрового следа в образовательных системах, обеспечивающей эффективную работу в условиях ограниченных вычислительных ресурсов. В предлагаемой работе разработана оригинальная архитектура системы учебной аналитики, основанная на микросервисном подходе с адаптерами для интеграции разнородных образовательных сервисов с использованием колоночной базы данных, Apache Kafka в роли брокера сообщений, обеспечивающего асинхронный и устойчивый обмен данными, а также Apache Airflow для оркестрации процессов аналитической обработки.

В отличие от существующих подходов, данная работа представляет принципиально новое решение, сочетающее гибкость микросервисной архитектуры с эффективностью колоночных баз данных и асинхронной потоковой обработки. Предложенные архитектурно-технические решения, в отличии от предшествующих аналогов, позволяют существенно снизить затраты на развертывание и обслуживание аналитических систем, обеспечивая высокую скорость обработки данных и возможность оперативного реагирования на изменения цифрового профиля студентов. Результатом работы является экспериментально подтвержденная закономерность между архитектурным паттерном обработки данных и вычислительной эффективностью аналитических операций в контексте образовательных данных. Проведенные эксперименты позволили выявить пороговые значения объемов данных, при которых наблюдается инверсия эффективности различных подходов к обработке.

Для достижения поставленной цели решались следующие задачи:

(*1*) Анализ существующих подходов к интеграции и обработке данных цифрового следа в образовательных системах.

(2) Разработка архитектуры программной системы для сбора и интеграции данных из разнородных образовательных платформ на основе микросервисного подхода.

(3) Создание и внедрение инфраструктурных решений для обеспечения надежности и масштабируемости системы.

(4) Экспериментальное сравнение эффективности предложенной архитектуры с существующими подходами по критериям потребления ресурсов и скорости обработки данных.

1. Обзор существующих решений

Задача агрегирования и анализа образовательных данных привлекла внимание многих исследователей. Ранние работы фокусировались на анализе журналов LMS (Learning Management Systems) и успеваемости студентов по завершении курсов. Например, в работе [3] предложен подход для выявления студентов группы риска на основе истории их действий в ходе обучения программированию. Авторами отмечается фрагментированность данных о деятельности студентов в разных курсах и преобладание пост-аналитики (по завершении курса) над отслеживанием текущего прогресса обучающихся.

В более поздних исследованиях акцент смещается на использование методов больших данных и потоковой обработки. Так, в публикации [1] разработан конвейер *big data* для прогнозирования факторов успеха студентов, в котором данные из LMS обрабатываются с помощью Apache Kafka и Spark. Подобный подход позволяет интегрировать разнообразные события и проводить сложный анализ (например, предсказание отчислений) на больших массивах данных. Однако такие решения требуют существенных вычислительных ресурсов и сложной инфраструктуры, включая кластерные системы хранения и обработки данных. В условиях же отдельного университета, колледжа или школы, где объемы данных умеренны, а ИТ-ресурсы ограничены, развертывание тяжелых платформ может быть неоправданным. Это подтверждается экспериментальными замерами: например, использование Spark для вычисления даже относительно простых метрик успеваемости приводит к заметно большему времени выполнения и загрузке памяти по сравнению с более легковесными инструментами (см. раздел 2).

Одновременно появляются исследования, предлагающие *итеративный* анализ учебных данных по мере их накопления. В работе [4] реализован подход прогрессивного учета временных данных об успеваемости студентов

для повышения точности прогнозов. Тем не менее, вопрос унификации разнородных источников данных и оперативности их сбора в этой работе остаётся вне фокуса. Таким образом, на сегодняшний день отсутствует универсальное решение, сочетающее *гибкость интеграции данных* и *эффективность аналитики* в контексте образовательных систем.

2. Понятие цифрового следа в образовании

Цифровой след в образовательном контексте представляет собой совокупность всех электронных записей о деятельности студента в процессе обучения. Формирование цифрового следа происходит при взаимодействии обучающегося с различными информационными системами: LMS, системами видеоконференцсвязи, платформами для совместной работы, системами контроля версий кода и другими образовательными инструментами.

Структура цифрового следа включает несколько ключевых компонентов:

Поведенческие данные – информация о действиях пользователя (время входа в систему, продолжительность сессий, последовательность просмотра материалов, паттерны навигации);

Результаты деятельности – оценки, выполненные задания, созданные артефакты, результаты тестирований;

Коммуникационные данные – сообщения в форумах, комментарии, участие в дискуссиях, взаимодействие с преподавателями и сокурсниками;

Метаданные – контекстная информация о времени, месте, устройстве доступа, используемых ресурсах.

Анализ цифрового следа позволяет решать различные задачи в образовании: от прогнозирования академической успеваемости до выявления проблемных областей в учебных программах. В рамках предлагаемой системы цифровой след агрегируется из множественных источников, нормализуется и сохраняется в едином хранилище для последующего анализа.

3. Предлагаемая архитектура

Предлагаемая архитектура системы представляет собой легковесную платформу для анализа цифрового следа, специально адаптированную

для образовательных учреждений с ограниченными вычислительными ресурсами. В отличие от существующих решений, ориентированных на крупномасштабную обработку данных, предложенный подход спроектирован по принципам микросервисной архитектуры, где отдельные компоненты (сервисы-адаптеры) отвечают за интеграцию с конкретными образовательными платформами (например, LMS, системы тестирования, форумы, системы видеоконференций и др.).

Каждый адаптер автономно собирает события и данные своего сервиса (через его API или базы данных), преобразует их во *внутренний унифицированный формат*, а затем передает в центральное хранилище аналитики (Сервер 1 на рисунке 1). Для передачи данных используется брокер сообщений (например, Apache Kafka) в комбинации с механизмами кэширования, что обеспечивает асинхронность и высокую пропускную способность обработки потоков событий.

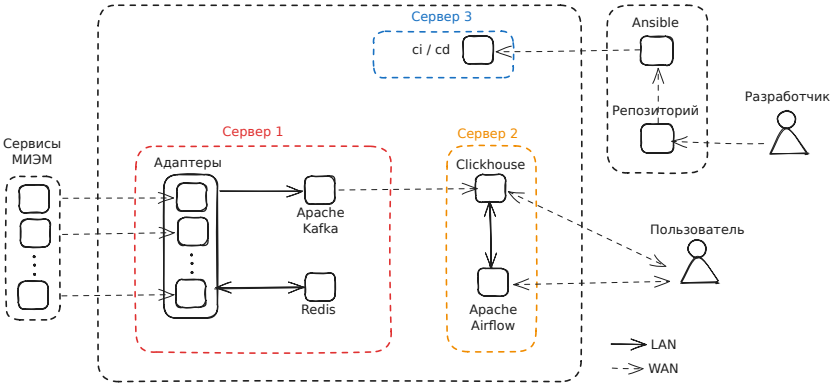

Рисунок 1. Архитектура инфраструктуры системы цифрового следа в МИЭМ НИУ ВШЭ

Такой подход позволяет существенно уменьшить задержки при передаче данных и нивелировать пиковые нагрузки за счет буферизации сообщений. Кроме того, благодаря выделению адаптеров в отдельные микросервисы, система обладает *устойчивостью к сбоям* и гибкостью: сбой или обновление одного адаптера не нарушает работу остальных, каждый компонент можно развёртывать и масштабировать независимо [5]. Центральный брокер и система очередей позволяют также оптимизировать количество прямых запросов к внешним учебным сервисам – вместо синхронного опроса данных используется подписка на события, что

снижает нагрузку на исходные системы и делает интеграцию более эффективной.

Собираемые различными адаптерами данные поступают в централизованное хранилище, реализованное на основе колоночной базы данных ClickHouse. Данный тип базы данных оптимизирован специально под аналитические запросы и агрегацию больших объемов информации [6], что делает его наиболее подходящим для работы с цифровым следом студентов, представляющим собой большие объемы данных разной структуры, которые необходимо оперативно агрегировать и анализировать [7]. Для эффективного управления потоком данных и обеспечения высокой производительности используется брокер сообщений Apache Kafka, который позволяет разгрузить основные компоненты системы и обрабатывать информацию асинхронно [8]. Kafka выступает промежуточным слоем, обеспечивая надежную передачу и временное хранение данных между адаптерами и центральным хранилищем.

Дополнительно в систему интегрирован Apache Airflow (Сервер 2 на рисунке 1), используемый для оркестрации аналитических процессов и автоматизации выполнения задач. Airflow позволяет создавать, планировать и контролировать выполнение последовательностей аналитических задач, гарантируя корректность обработки данных и своевременность получения результатов. Такое хранилище оптимизировано под массовые выборки и вычисления по столбцам (агрегации, сканирование большого числа строк) и обеспечивает высокую скорость выполнения сложных аналитических запросов по накопленным данным.

В нашем решении реализована центральная фактовая таблица (*data lake* таблица big_table в ClickHouse), в которую адаптеры последовательно добавляют события в нормализованном виде. Эта таблица содержит унифицированные поля (идентификатор сервиса, типа события, времени, участника и пр.) и специальные поля для каждого конкретного сервиса (они остаются пустыми для событий других типов) – подобная схема позволяет хранить разнородные записи в одной структуре, сохраняя при этом возможность эффективной фильтрации по типу сервиса. Для повышения производительности запросов применены оптимизации хранения: для категориальных полей (название сервиса, тип события и др.) используется тип сжатия LowCardinality, уменьшающий объем данных и ускоряющий фильтрацию, а на часто используемые поля (например, время события) добавлены индексы, что ускоряет диапазонные запросы.

4. Технологическая инфраструктура

Для обеспечения надежной работы системы цифрового следа с предложенной архитектурой применены современные инструменты контейнеризации и оркестрации сервисов. Каждый компонент системы (адаптеры, брокер Kafka, базы данных, аналитические сервисы) упакован в виде Docker-контейнера. Это гарантирует единообразие среды выполнения, упрощает развертывание на серверных мощностях образовательного учреждения и повышает переносимость решения [9] [10]. Поверх Docker-компонентов настроена автоматизация развертывания и обновления с помощью конвейера CI/CD.

В прототипе системы использован GitLab CI для непрерывной интеграции и доставки: при обновлении кода адаптера или другого сервиса автоматически запускаются тесты, собирается новый контейнерный образ и разворачивается на целевом сервере без остановки всей системы (Сервер 3 на рисунке 1). Инфраструктурные настройки (конфигурации сервисов, учетные данные, сетевые параметры) управляются декларативно с помощью инструментов *infrastructure as code* – например, Ansible. Это позволило свести к минимуму человеческий фактор при настройке системы и обеспечить воспроизводимость развёртывания на разных площадках (тестовый стенд, учебная лаборатория или продуктивный сервер). Кроме того, для поддержания стабильности [11] в длительной перспективе внедрены средства мониторинга и логирования: метрики работы сервисов собираются (например, посредством Prometheus), а агрегированные логи позволяют оперативно обнаруживать сбои.

Таким образом, система с предложенной архитектурой дополняется инфраструктурным контуром, обеспечивающим ее *масштабируемость и отказоустойчивость*. Даже при ограниченных вычислительных ресурсах предложенные меры (легковесные контейнеры Alpine, автоматическое восстановление сервисов, динамическое масштабирование критичных компонентов) повышают устойчивость работы. В таблице 1 приведено сравнение нового подхода с традиционными системами LMS и с системой из работы [12].

5. Оценка эффективности подхода

Для количественной оценки преимуществ предложенной архитектуры проведено экспериментальное сравнение производительности при разных методах обработки данных цифрового следа. В качестве тестового сценария выбрана вычислительно затратная аналитическая операция – определение

Таблица 1. Сравнительный анализ предлагаемого подхода с существующими решениями

Характеристика	*Предлагаемое решение*	*Подход из работы* [12]	*Традиционные системы LMS*
Архитектура	Микросервисная с адаптерами	Монолитная архитектура Big Data	Встроенные модули в LMS
Технологический стек	ClickHouse, Kafka, Airflow, контейнеризация	Hadoop, Spark, Hive	SQL, встроенные инструменты отчетности
Интеграция данных	Множество источников через унифицированные адаптеры	Преимущественно LMS и системы программирования	Только внутренние данные LMS
Масштабируемость	Горизонтальная, компонентная	Кластерная (требует существенных ресурсов)	Ограниченная
Требования к ресурсам	Умеренные (500 МБ RAM на сервис)	Высокие (кластеры, выделенные серверы)	Низкие (встроены в существующую инфраструктуру)
Оперативность анализа	Близкая к реальному времени	Периодическая (пакетная обработка)	Ретроспективная
Режим развертывания	Контейнеры, CI/CD	Кластерное развертывание	Интегрировано в LMS
Тип аналитики	Прогностическая и описательная	Преимущественно прогностическая	Преимущественно описательная

медианного балла студента на основе накопленных оценок (например, по результатам множества заданий или тестов). Данный расчёт был выполнен тремя способами:

- с использованием распределённой платформы Apache Spark (имитируя подход крупномасштабной обработки данных, применённый в работе [12]);
- средствами локальной обработки в Pandas (табличные вычисления в памяти на одной машине);
- с использованием библиотеки Polars (колончатая обработка в памяти).

Для имитации размеров данных крупных курсов были сгенерированы выборки: *Scenario A* – 2500 записей результатов на студента (что соответствует, например, курсу средней активности), и *Scenario B* – 25000 записей

на студента (что моделирует данные сразу по нескольким курсам или очень длительному курсу). Метриками эффективности выступали время выполнения вычисления и потребление оперативной памяти.

Выбор Pandas и Polars обусловлен их архитектурными особенностями. Pandas является де-факто стандартом для анализа данных в Python, обеспечивая богатую функциональность и простоту использования. Однако его однопоточная архитектура и использование row-based представления данных могут становиться узким местом при обработке больших объемов [13]. Polars, напротив, построен на основе колоночного представления данных и использует современные техники оптимизации: ленивые вычисления, автоматическую параллелизацию операций, оптимизацию запросов и эффективное использование CPU cache [14].

Apache Spark, хотя и предоставляет мощные возможности для распределенной обработки, имеет существенные накладные расходы на инициализацию SparkContext, сериализацию/десериализацию данных между узлами, координацию задач между исполнителями [15]. В условиях одиночного сервера эти накладные расходы не компенсируются преимуществами распределенной обработки [16].

Результаты эксперимента подтвердили существенное преимущество легковесных подходов над Spark в условиях одиночного сервера (2 CPU Intel Cascade Lake, 4 ГБ ОЗУ, 64 ГБ дискового пространства). Так, в случае *Scenario A* Spark демонстрировал самое большое время выполнения задачи при прочих равных условиях – задержка оказалась в несколько раз выше, чем при использовании pandas или Polars. По показателю использования памяти Spark также был менее эффективен, потребляя значительно больше ресурсов для обработки 2,5 тысяч записей на каждого студента.

Детальный анализ показал, что основные затраты времени в Spark приходятся на:

- Инициализацию SparkSession и создание RDD/DataFrame (до 15 секунд)
- Планирование и распределение задач (3-5 секунд)

Для *Scenario B* (увеличенный объём данных) платформа Spark ожидаемо улучшила относительные показатели (благодаря распределенной обработке), однако даже при 25 тысячах записей затраты времени у Spark оставались выше, чем в случае Polars, а использование памяти – самым высоким среди всех трёх решений. На рисунках 2 и 3 схематически показано сравнение времени выполнения в сценариях A и B для разных инструментов.

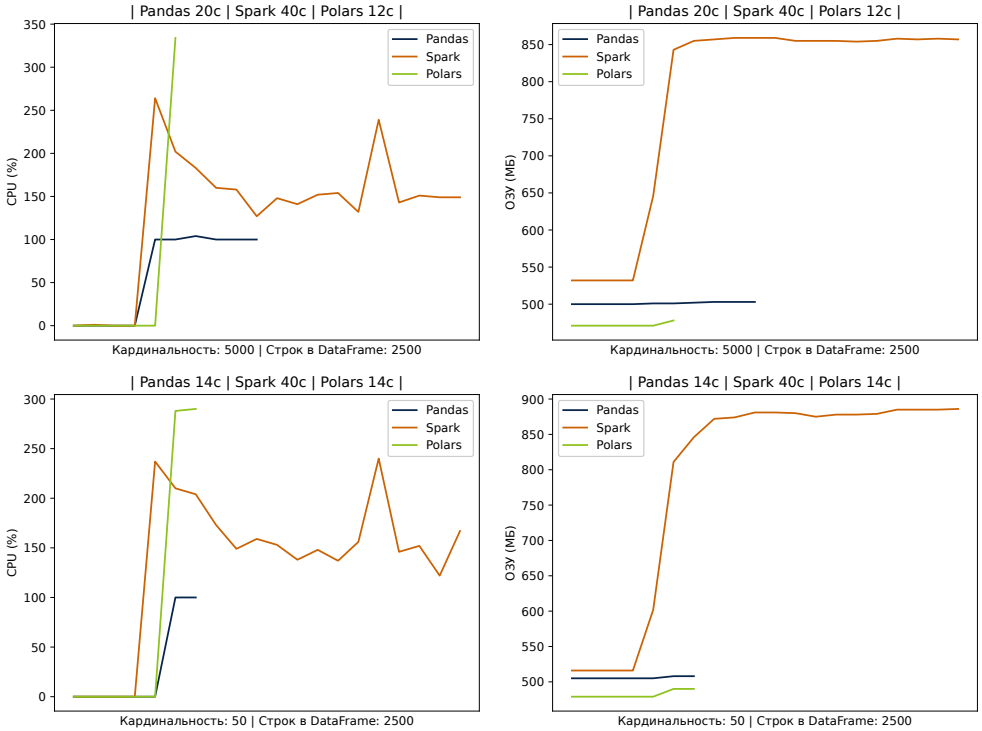

Рисунок 2. Время выполнение задачи и потребление ресурсов на Pandas, Spark и Polars для 2500 записей на студента

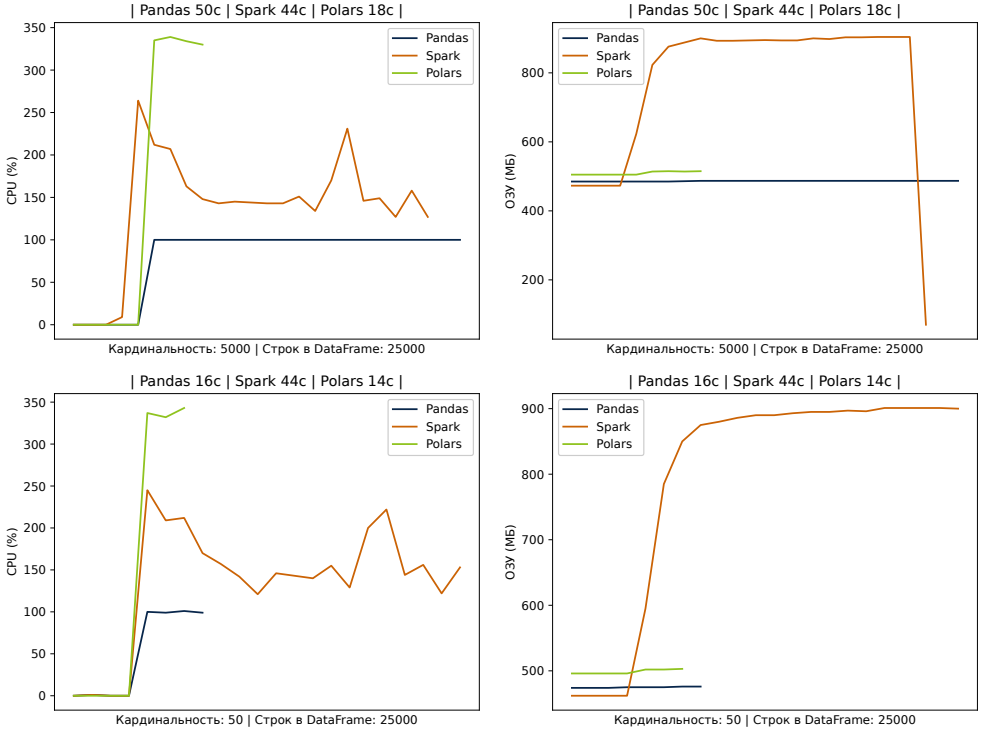

Рисунок 3. Время выполнение задачи и потребление ресурсов на Pandas, Spark и Polars для 25000 записей на студента

В частности, видно, что кривая времени для Spark проходит существенно выше, тогда как pandas и Polars обеспечивают более быструю обработку данных. Эти наблюдения согласуются с тем, что использование тяжеловесных фреймворков оправдано на *очень больших* данных и при наличии кластерной инфраструктуры, но в среде с ограниченными ресурсами оно приводит лишь к дополнительным накладным расходам. Наш подход, напротив, делает упор на оптимизацию хранения и обработки внутри одной масштабируемой узловой системы: применение колончатой базы и эффективных алгоритмов локальной обработки (например, Polars) позволяет достичь близкой к кластерным решениям скорости на типичных для отдельного вуза объёмах данных, не привлекая дорогостоящие ресурсы.

Легковесные методы (Pandas, Polars) существенно превосходят Spark по скорости в обоих сценариях, что особенно заметно на относительно небольшом объёме данных. Это демонстрирует эффективность выбранного в работе подхода в условиях ограниченных ресурсов. Данные по обработке тестовых данных приведены в таблице 2.

Таблица 2. Количественные результаты сравнения производительности различных инструментов обработки данных

Сценарий	*Инструмент*	*Макс. потребление памяти (МБ)*	*Время выполнения (сек)*	*Относит. время (%)*
Сценарий A (2500 записей, 50 студентов)	Spark	886	40,17	285,2
	Pandas	508	14,13	100,2
	Polars	490	14,10	100
Сценарий A (2500 записей, 5000 студентов)	Spark	859	40,22	332,3
	Pandas	503	20,14	166,4
	Polars	478	12,10	100
Сценарий B (25000 записей, 50 студентов)	Spark	901	44,32	314,4
	Pandas	476	16,11	114,3
	Polars	503	14,10	100
Сценарий B (25000 записей, 5000 студентов)	Spark	904	44,49	245,6
	Pandas	487	50,27	277,5
	Polars	515	18,11	100

Анализ результатов позволяет сделать важное наблюдение о пороговых значениях данных. При объемах до 100 МБ (примерно 1 миллион записей) накладные расходы Spark не окупаются. При объемах 100 МБ ÷ 1 ГБ

Pandas начинает испытывать проблемы с производительностью из-за неэффективного использования памяти, в то время как Polars сохраняет высокую производительность благодаря колоночному представлению и оптимизациям. Только при объемах свыше 10 ГБ распределенная природа Spark начинает давать преимущества, но даже в этом случае требуется кластерная инфраструктура для полноценного использования его возможностей.

Кроме производительности, важным результатом внедрения предложенной архитектуры стало повышение *гибкости и расширяемости* системы анализа цифрового следа. Благодаря модульности адаптеров стало возможным легко подключать новые источники данных: в рамках прототипа были реализованы адаптеры для нескольких популярных инструментов (система управления проектами, платформа асинхронного командного общения для студентов, система видеоконференцсвязи), и их интеграция в общую систему не потребовала изменений в основных компонентах – достаточно задать формат выходных событий и подключить их к брокеру сообщений. Автоматизированное развертывание через CI/CD позволило сократить время ввода системы в эксплуатацию: первоначальная настройка всех компонентов на сервере заняла считанные часы, а обновления (например, выпуск новой версии аналитического модуля) происходят практически без простоя системы. В совокупности, предложенное решение демонстрирует, что даже без использования громоздких технологий больших данных можно добиться эффективного и масштабируемого анализа образовательных данных.

Заключение

В работе предложена архитектура информационной среды для интеграции и анализа цифрового следа студентов в образовательных учреждениях, учитывающая ограничения вычислительных ресурсов и сложности интеграции данных из различных цифровых сервисов. Реализованный подход сочетает микросервисную модель с адаптерами, позволяющими оперативно интегрировать разнородные источники данных, и многоуровневую систему хранения (реляционную, документоориентированную и колончатую базы данных), что обеспечивает эффективную обработку и аналитику учебной активности. Экспериментальная проверка подтвердила преимущества предлагаемого решения по сравнению с подходами, предложенными в работах других исследовательских групп, за

счет существенного снижения времени обработки данных и уменьшения потребления ресурсов, а также повышения надежности и гибкости системы благодаря автоматизированному развертыванию и управлению инфраструктурой.

Ключевым результатом работы является демонстрация того, что анализ цифрового следа в образовательных системах не требует обязательного использования тяжеловесных платформ Big Data. Правильный выбор архитектуры и технологий позволяет достичь высокой производительности на типичных для образовательных учреждений объемах данных при умеренных требованиях к ресурсам.

Полученные результаты могут быть использованы для повышения качества образовательного процесса и поддержки принятия решений на основе комплексных данных цифрового следа студентов. Дальнейшее развитие подхода может быть связано с расширением интегрируемых сервисов и внедрением методов интеллектуальной аналитики.

Список использованных источников

[1] Siemens G., Gasevic D., Dawson S. *Preparing for the digital university: a review of the history and current state of distance, blended, and online learning*, Technical Report.– MOOC Research Initiative.– 2015.– 235 с. doi URL ↑4, 5

[2] Rodrigues M. W., Isotani S., Zárate L. E. *Educational Data Mining: A review of evaluation process in the e-learning* // Telematics and Informatics.– 2018.– Т. **35**.– № 6.– С. 1701–1717. doi ↑4

[3] Azcona D., Hsiao I-H., Smeaton A. F. *Detecting students-at-risk in computer programming classes with learning analytics from students' digital footprints* // User Modeling and User-Adapted Interaction.– 2019.– Т. **29**.– № 4.– С. 759–788. doi ↑5

[4] Trakunphutthirak R., Lee V. C. S. *Application of educational data mining approach for student academic performance prediction using progressive temporal data* // Journal of Educational Computing Research.– 2021.– Т. **60**.– № 3.– С. 742–776. doi ↑5

[5] Newman S. *Building Microservices: Designing Fine-Grained Systems*, 2nd ed..– O'Reilly Media.– 2021.– ISBN 978-1492034025.– 612 с. ↑7

[6] Anand V. *Up and Running with ClickHouse: Learn and Explore ClickHouse, It's Robust Table Engines for Analytical Tasks, ClickHouse SQL, Integration with External Application and Managing the ClickHouse Server*.– BPB Publications.– 2021.– ISBN 978-9391392246.– 280 с. ↑8

[7] Kleppmann M. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*, 1st ed..– O'Reilly Media.– 2017.– ISBN 978-1449373320.– 611 с. ↑8

[8] Narkhede N., Shapira G., Palino T. *Kafka: The Definitive Guide: Real-time Data and Stream Processing at Scale*, 1st ed..– O'Reilly Media.– 2017.– ISBN 978-1491936160.– 319 с. ↑8

[9] Kim G., Humble J., Debois P., Willis J., Forsgren N. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, 2nd ed..– IT Revolution.– 2021.– ISBN 978-1950508433.– 480 с. ↑9

[10] Morris K. *Infrastructure as Code: Managing Servers in the Cloud*, 1st ed..– O'Reilly Media.– 2016.– ISBN 978-1491924358.– 360 с. ↑9

[11] Brazil B. *Prometheus: Up & Running: Infrastructure and Application Performance Monitoring*, 1st ed..– O'Reilly Media.– 2018.– ISBN 978-1492034148.– 384 с. ↑9

[12] Fahd K., Miah S. J. *Designing and evaluating a big data analytics approach for predicting students' success factors* // Journal of Big Data.– 2023.– Т. **10**.– ид. 159.– 19 с. doi ↑9, 10

[13] McKinney W. *Data structures for statistical computing in Python* // *Proceedings of the 9th Python in Science Conference (SciPy 2010)* (Austin, Texas, USA June 28–July 3, 2010).– 2010.– С. 56–61. doi ↑11

[14] Mozzillo A., Zecchini L., Gagliardelli L., Aslam A., Bergamaschi S., Simonini G. *Evaluation of dataframe libraries for data preparation on a single machine* // *Proceedings of the 28th International Conference on Extending Database Technology (EDBT 2025)*.– Т. 2 (Barcelona, Spain, March 25–March 28, 2025), Advances in Database Technology.– т. **28**.– OpenProceedings.org.– 2025.– ISBN 978-3-89318-098-1.– С. 337–349. doi ↑11

[15] Karau H., Warren R. *High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark*, 1st ed..– O'Reilly Media.– 2017.– ISBN 978-1491943205.– 356 с. ↑11

[16] Zaharia M., Xin R. S., Wendell P., Das T., Armbrust M., Dave A., Meng X., Rosen J., Venkataraman S., Franklin M. J., Ghodsi A., Gonzalez J. E., Shenker S., Stoica I. *Apache Spark: a unified engine for big data processing* // Communications of the ACM.– 2016.– Т. **59**.– № 11.– С. 56–65. doi ↑11

Поступила в редакцию 16.05.2025;
одобрена после рецензирования 24.06.2025;
принята к публикации 12.08.2025;
опубликована онлайн 27.08.2025.

Рекомендовал к публикации *к.т.н. В. П. Фраленко*

Информация об авторах:

Михаил Андреевич Степанов

аспирант школы по техническим наукам НИУ ВШЭ, среди научных интересов - проектирование систем, системный анализ, высоконагруженные приложения

0000-0003-1331-8897

e-mail: *mastepanov@hse.ru*

Сергей Александрович Сластников

в 2010 году получил степень магистра по специальности «Прикладная математика» в Московском государственном институте электроники и математики (Россия), а в 2015 году — степень кандидата наук. В настоящее время — старший преподаватель, доцент Национального исследовательского института НИУ ВШЭ. Область его научных интересов включает методы оптимизации, системный анализ, прикладные модели искусственного интеллекта и машинное обучение. Автор более 15 статей.

0000-0003-4272-9951

e-mail: *pics/sslastnikov@hse.ru*
sslastnikov@hse.ru

Никита Андреевич Климин

сотрудник НИУ ВШЭ отдела прикладного искусственного интеллекта. Заинтересован в разработке интерпретируемых моделей и их прикладном использовании

0009-0009-6756-0866

e-mail: *nklimin@hse.ru*

Вклад авторов: *М. А. Степанов* – 70% (идея, методология, программное обеспечение, формальный анализ, расследование, сбор материала, написание черновой версии, доработка и редактирование); *С. А. Сластников* – 20% (идея, методология, валидация, курирование данных); *Н. А. Климин* – 10% (программное обеспечение, расследование).

Декларация об отсутствии личной заинтересованности: *благополучие авторов не зависит от результатов исследования.*

PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article APPLIED SOFTWARE SYSTEMS

UDC 004.415
10.25209/2079-3316-2025-16-4-3-21

Designing Software Architecture for Digital Footprint Analysis in Educational Systems

Mikhail Andreevich **Stepanov**[1], Sergei Aleksandrovich **Slastnikov**[2], Nikita Andreevich **Klimin**[3]

[1–3] National Research University Higher School of Economics, Moscow, Russia

[1]✉ *mastepanov@hse.ru*

Abstract. Modern educational platforms and e-learning tools generate large volumes of data about student activities, known as the "digital footprint". Analysis of this data allows tracking student progress, identifying dropout risks, and predicting academic success. However, many educational organizations lack the IT infrastructure necessary for comprehensive collection, storage, and analysis of digital footprints. Existing solutions are often fragmented, focused only on retrospective analysis, or require resource-intensive big data technologies, making them difficult to implement in educational organizations.

This paper proposes a software system architecture for integrating and analyzing digital footprints in educational systems that addresses these limitations. Approaches to data collection through adapters, unification and storage of heterogeneous data, as well as infrastructure solutions that ensure system reliability and scalability (microservice architecture, containerization, CI/CD, monitoring) are described. An experimental analysis of the effectiveness of the proposed approach is presented using the example of calculating performance indicators, demonstrating a reduction in processing time and resource utilization compared to traditional approaches. The proposed solutions allow for the creation of flexible and high-performance educational analytics systems suitable for implementation even in resource-constrained environments. (*In Russian*).

Key words and phrases: digital footprint; digital profile; software architecture; microservices; infrastructure; learning analytics

2020 *Mathematics Subject Classification*: 93B15; 68T09, 93C57

For citation: Mikhail A. Stepanov, Sergei A. Slastnikov, Nikita A. Klimin. *Designing Software Architecture for Digital Footprint Analysis in Educational Systems*. Program Systems: Theory and Applications, 2025, **16**:4(67), pp. 3–21. (*In Russ.*). `https://psta.psiras.ru/read/psta2025_4_3-21.pdf`

References

[1] G. Siemens, D. Gasevic, S. Dawson. *Preparing for the digital university: a review of the history and current state of distance, blended, and online learning*, Technical Report, MOOC Research Initiative, 2015, 235 pp.

[2] M. W. Rodrigues, S. Isotani, L. E. Zárate. "Educational Data Mining: A review of evaluation process in the e-learning", *Telematics and Informatics*, **35**:6 (2018), pp. 1701–1717.

[3] D. Azcona, I-H. Hsiao, A. F. Smeaton. "Detecting students-at-risk in computer programming classes with learning analytics from students' digital footprints", *User Modeling and User-Adapted Interaction*, **29**:4 (2019), pp. 759–788.

[4] R. Trakunphutthirak, V. C. S. Lee. "Application of educational data mining approach for student academic performance prediction using progressive temporal data", *Journal of Educational Computing Research*, **60**:3 (2021), pp. 742–776.

[5] S. Newman. *Building Microservices: Designing Fine-Grained Systems*, 2nd ed., O'Reilly Media, 2021, ISBN 978-1492034025, 612 pp.

[6] V. Anand. *Up and Running with ClickHouse: Learn and Explore ClickHouse, It's Robust Table Engines for Analytical Tasks, ClickHouse SQL, Integration with External Application and Managing the ClickHouse Server*, BPB Publications, 2021, ISBN 978-9391392246, 280 pp.

[7] M. Kleppmann. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*, 1st ed., O'Reilly Media, 2017, ISBN 978-1449373320, 611 pp.

[8] N. Narkhede, G. Shapira, T. Palino. *Kafka: The Definitive Guide: Real-time Data and Stream Processing at Scale*, 1st ed., O'Reilly Media, 2017, ISBN 978-1491936160, 319 pp.

[9] G. Kim, J. Humble, P. Debois, J. Willis, N. Forsgren. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, 2nd ed., IT Revolution, 2021, ISBN 978-1950508433, 480 pp.

[10] K. Morris. *Infrastructure as Code: Managing Servers in the Cloud*, 1st ed., O'Reilly Media, 2016, ISBN 978-1491924358, 360 pp.

[11] B. Brazil. *Prometheus: Up & Running: Infrastructure and Application Performance Monitoring*, 1st ed., O'Reilly Media, 2018, ISBN 978-1492034148, 384 pp.

[12] K. Fahd, S. J. Miah. "Designing and evaluating a big data analytics approach for predicting students' success factors", *Journal of Big Data*, **10** (2023), id. 159, 19 pp.

[13] W. McKinney. "Data structures for statistical computing in Python", *Proceedings of the 9th Python in Science Conference (SciPy 2010)* (Austin, Texas, USA June 28–July 3, 2010), 2010, pp. 56–61.

[14] A. Mozzillo, L. Zecchini, L. Gagliardelli, A. Aslam, S. Bergamaschi, G. Simonini. "Evaluation of dataframe libraries for data preparation on a single machine", *Proceedings of the 28th International Conference on Extending Database Technology (EDBT 2025)*. V. 2 (Barcelona, Spain, March 25–March 28, 2025), Advances in Database Technology, vol. **28**, OpenProceedings.org, 2025, ISBN 978-3-89318-098-1, pp. 337–349.

[15] H. Karau, R. Warren. *High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark*, 1st ed., O'Reilly Media, 2017, ISBN 978-1491943205, 356 pp.

[16] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. E. Gonzalez, S. Shenker, I. Stoica. "Apache Spark: a unified engine for big data processing", *Communications of the ACM*, **59**:11 (2016), pp. 56–65.