УДК 004.4'22, 004.75, 004.85 DOI: 10.14529/cmse250203

ВЫПОЛНЕНИЕ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ НА MLOPS ПЛАТФОРМЕ НИУ ВШЭ*

© 2025 А.С. Хританков, В.А. Полежаев, Г.А. Жуликов, М.С. Галынчик, Н.А. Климин, К.Е. Сахаров, В.О. Минченков, И.В. Спирин, И.И. Крупнов, С.Ф. Якушева, А.С. Маратканова, В.И. Козырев, П.С. Костенецкий, Х.М. Салех

Национальный исследовательский университет Высшая школа экономики (109028 Москва, Покровский бул., д. 11)

E-mail: akhritankov@hse.ru

Поступила в редакцию: 19.05.2025

Несмотря на распространение и успешные применения средств интеллектуального анализа и обработки данных для решения отдельных прикладных задач, все еще не решена проблема разработки технологии создания таких программных средств. В работе в контексте единого процесса MLOps создания технологий машинного обучения рассматриваются возникающие задачи автоматизации и выполнения распределенных вычислительных экспериментов на базе единой вычислительной платформы. Разрабатываемая в НИУ ВШЭ платформа MLOps предназначена для развертывания интеллектуальных веб-сервисов и программных средств анализа данных. Платформа должна управлять доступными локально и в облачной среде разнородными ресурсами и объединять их с ресурсами вычислительного кластера cHARISMa НИУ ВШЭ под управлением Slurm. Таким образом актуальна задача интеграции указанных ресурсов для проведения вычислительных экспериментов, реализации конвейеров настройки моделей машинного обучения, решения задач обработки и анализа данных. Особенностями решаемой задачи являются рассмотрение процесса вычислений, как составной части технологии создания интеллектуальных веб-сервисов, обусловленная этой технологией необходимость использования разнородных ресурсов и использование единой гибридной платформы для выполнения вычислений. В работе предложено решение указанной задачи интеграции вычислений и приведены результаты апробации решения для интеллектуальных веб-сервисов. Показана принципиальная возможность такой интеграции разнородных ресурсов в одном вычислительном эксперименте на базе расширяемой пользователем объектной модели эксперимента и предметно-ориентированного языка его спецификации, решены вопросы динамического управления развертыванием интеллектуальных приложений, интеграции конвейеров обработки данных, веб-сервисов и наборов данных для выполнения распределенных вычислительных экспериментов.

Ключевые слова: распределенные вычислительные эксперименты, машинное обучение, облачные технологии, MLOps.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Хританков А.С., Полежаев В.А., Жуликов Г.А. и др. Выполнение распределенных вычислительных экспериментов на MLOps платформе НИУ ВШЭ // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2025. Т. 14, № 2. С. 42–66. DOI: 10.14529/cmse250203.

Введение

Распределенные вычислительные системы используются во многих областях при проведении вычислительных экспериментов, обучении моделей машинного обучения, обработки больших данных, в научных вычислениях. Вычислительные кластеры, в том числе со спе-

*Статья рекомендована к публикации программным комитетом Всероссийской научной конференции с международным участием «Параллельные вычислительные технологии (ПаВТ) 2025».

циализированными ускорителями вычислений, Грид-системы [1], системы общественных вычислений (public/volunteer computing), облачные центры обработки данных совместного использования, туманные вычисления (fog computing) [2] с использованием ресурсов оконечных и промежуточных устройств и другие составляют современный спектр подходов к решению задач в области.

Разработка таких распределенных систем [3] и создание методов и средств, облегчающих разработчикам и исследователям доведение результатов своих прикладных научных исследований до промышленного применения на их основе, составляют одну из важнейших задач прикладной науки и экономики.

Интегрированный процесс создания приложений в области машинного обучения (MLOps) объединяет этапы моделирования, разработки программного обеспечения, тестирование, внедрение и использование таких приложений и интеллектуальных сервисов в их составе. Сходные с MLOps подходы применяются организациями для создания и последующего развития нескольких интеллектуальных приложений разными командами.

Центр искусственного интеллекта НИУ ВШЭ [4] разрабатывает **MLOps фреймворк**— библиотеку программных средств автоматизации процедур процесса MLOps, на ее основе создана распределенная вычислительная **платформу MLOps** развертывания интеллектуальных веб-сервисов. Фреймворк автоматизирует процедуры процесса MLOps и скрывает излишнюю для исследователей данных сложность создания и эксплуатации облачных веб-сервисов и выполнения расчетных конвейеров в распределенной среде. Платформа объединяет облачные вычислительные ресурсы, облачные средства хранения данных и суперкомпьютерный кластер cHARISMa. Платформа может быть использована разработчиками-исследователями как облачный сервис (SaaS) при создании интеллектуальных сервисов и приложений с подключением конечных потребителей сервисов через API по сети Интернет.

Для этого необходимо решить проблему интеграции доступных платформе разнородных вычислительных ресурсов [5], как локально на ПК разработчика приложения, так и в центре обработки данных и ресурсов вычислительного кластера НИУ ВШЭ. Задача возникает при проведении распределенных вычислительных экспериментов в области машинного обучения и в настоящий момент часто решается вручную, от случая к случаю с использованием подручных программных средств и с привлечением специалистов в области высокопроизводительных вычислений, разработчиков облачного программного обеспечения и других.

В настоящей работе проблема интеграции вычислительных ресурсов рассматривается, как совокупность задач модельно-ориентированной разработки ПО (MDSD), то есть разработки модели предметной области интеллектуальных приложений, объединяющей различные их компоненты на общих правилах, и задач создания программных средств трансляции моделей приложений и исполнения приложений с использованием доступных вычислительных ресурсов. Выделение концептуальной модели предметной области, описывающей структуру приложения и его сервисов позволяет решать дальнейшие задачи воспроизводимости и прослеживаемости распределенных вычислительных экспериментов сразу для широкого класса приложений в области машинного обучения. Не связанные с фреймворком разработчики приложений в области биоинформатики, анализа данных физических экспериментов, медицинской диагностики, интеллектуального поиска, заботы и аналитики по сотрудникам и других могут использовать реализованную в MLOps фреймворке модель для описания структуры своих приложений, фреймворк обеспечивает интеграцию компо-

нентов этих приложений, в том числе конвейеров обучения расчетных моделей в рамках вычислительных экспериментов.

Предлагаемый подход к интеграции разнородных вычислительных ресурсов на единой платформе на основе модели и его реализация составляют основной результат данной работы. Анализ проблемы интеграции и существующих решений в области приведены в разделе 1. В разделе 2 более полно изложены основные решения по автоматизации процесса MLOps и интеграции вычислительных ресурсов для его автоматизации. В разделе 3 изложены алгоритмы и структура предлагаемого решения. Экспериментальная проверка его реализуемости и расчеты эффективности использования вычислительных ресурсов приведены в разделе 4.

1. Анализ литературы и альтернативные решения

Задача интеграции вычислительных ресурсов состоит в их объединении в общую среду с единым доступом и автоматическим планированием и распределением задач. При этом вычислительные ресурсы могут быть разнородными. Выделяют несколько типовых решений. Вычислительные кластеры (compute cluster) состоят из обычных ЭВМ с централизованным управлением. Распространенная гибридная архитектура кластеров сочетает использование процессоров (CPU), графических карт (GPU) и иногда специализированных ускорителей [6]. Грид-системы (grid computing) объединяют кластеры, суперкомпьютеры и другие ресурсы, разделенные географически и часто связанные с экспериментальными установками. Грид-системы отличает отсутствие централизованного управления, разнородность и низкая связанность вычислительных узлов. Виртуальные организации объединяют пользователей, использующих Грид-системы для решения задач [1]. К разновидности Грид-систем относят системы общественных вычислений (public/volunteer computing), в которых расчеты проводятся пользователями, добровольно предоставляющими свои компьютеры. Технология облачных вычислений (cloud computing) позволяет проводить вычисления на сторонних серверах центра обработки данных, предоставляемых в аренду — «облаке». Преимуществами облачных вычислений являются отсутствие необходимости приобретения и самостоятельного обслуживания оборудования и гибкая тарификация.

Традиционными подходами к интеграции удаленных ресурсов являются использование стандартных сетевых протоколов, таких как SSH, и написание программ-агентов, работающих на стороне ресурса [7]. Известно решение задачи интеграции суперкомпьютера НИЦ «Курчатовский институт» с Грид-системой и облачной платформой с помощью унифицированных интерфейсов и программы PanDA [8]. Также произведена интеграция суперкомпьютера «Говорун», двух Грид-кластеров, кластера NICA и облака ОИЯИ с помощью библиотеки DIRAC Interware [9]. Для решения задачи интеграции гетерогенных вычислительных ресурсов применяются технологии виртуализации [10]. Для интеграции суперкомпьютеров и облачных вычислений предложена концепция MC2E [5].

В научной среде популярна организация процесса вычислений в виде конвейера (workflow) из множества соединенных инструментов [11], в том числе контейнеров [12], изолирующих исполнение отдельных этапов. Конвейеры расчетов (pipeline) используются для обучения моделей и проверки гипотез. В работе [6] рассматривается задача интеграции высокопроизводительных вычислений, технологий анализа данных и машинного обучения в рамках одного конвейера и предлагается парадигма HPC Workflow-as-a-Service.

Вычислительный эксперимент проводится над математической моделью объекта с помощью вычислительных и логических процедур, осуществляемых соответствующими программными средствами на вычислительных системах. Проведение вычислительных экспериментов в области машинного обучения предполагает определение цели эксперимента, проверяемых гипотез о свойствах данных, критериев успешности, используемых данных, структуры расчетной модели и связанной задачи оптимизации для подбора параметров модели, определения конвейеров с необходимыми расчетами для проверки выбранных гипотез.

При проведении вычислительных экспериментов для улучшения воспроизводимости и повышения доверия к их результатам, разработки и улучшения конвейеров расчетов [13, 14], необходимо устанавливать происхождение данных и определять произведенные с ними операции, что составляет задачу прослеживаемости (data provenance) [15]. Для ее решения используются технологии распределенных реестров; моделирование конвейера расчетов с помощью графов знаний, семантических сетей и онтологий [16].

MLOps платформа автоматизирует процессы, связанные с проведением вычислительных экспериментов, обучением, внедрением и наблюдением за использованием расчетных моделей и интеллектуальных сервисов на их основе. Платформа решает задачу интеграции разнородных вычислительных ресурсов, суперкомпьютера cHARISMa НИУ ВШЭ, локальных вычислительных ресурсов исследователя и облачных ресурсов для проведения распределенных вычислительных экспериментов.

На решение сходных задач направлены несколько программных систем. Система Asperitas [17] предназначена для управления виртуальными сетями и вычислительными кластерами и хранения данных и включает в себя облачную среду, оркестраторы и приложения для контроля на основе программного средства автоматизации сборки программ Ansible. Система Fanlight (Desktop-as-a-Service) [18] предназначена для создания виртуальных рабочих мест с доступом к интегрированным прикладным математическим пакетам и аппаратным ресурсам высокопроизводительных вычислительных комплексов, подключенных систем хранения данных, вычислительных серверов. При построении системы Everest была произведена интеграция облачных ресурсов и Грид-систем [3]. При этом вышеперечисленные платформы не имеют специализации на технологиях машинного обучения и собственных вычислительных ресурсов, доступных пользователям системы.

Платформа DataMall [19] использует методы машинного обучения, инфраструктуру БД и единую облачную экосистему для работы с объектами на основе больших данных. Также платформа играет роль биржи специалистов, данных и ресурсов. Об интеграции вычислительных кластеров, Грид-систем и других ресурсов не упоминается.

Коммерческие платформы, такие как VK ML Cloud, Google Cloud AI, Amazon SageMaker и Azure AI Studio, предоставляют доступ только к облачным ресурсам их разработчиков с возможностью подключения дополнительных вычислителей в режиме гибридного облака (hybrid computing). Платформа HuggingFace использует облачные ресурсы партнеров, таких как Amazon и Google. Платформа MLSpace интегрирует суперкомпьютеры Christofari и Christofari Neo. Система ClearML рассчитана на использование ресурсов пользователя. Таким образом, крупнейшие коммерческие ML-платформы [20] решают задачи интеграции меньшего числа ресурсов по сравнению с разрабатываемой MLOps платформой. Программные средства с открытым исходным кодом, такие как MLflow, Kuberflow,

Airflow и TensorFlow Extended [20], предназначены для общей организации выполнения конвейеров расчетов и не решают задачу интеграции вычислительных ресурсов.

2. Процесс MLOps и архитектура платформы

В области машинного обучения вычислительные эксперименты проводятся для проверки гипотез о соответствии или оптимизации параметров предлагаемой расчетной модели имеющимся данным. Результаты такого эксперимента в на практике используют для создания интеллектуальных сервисов, предоставляющих обученные расчетные модели для использования через интерфейсы прикладного программирования (API). Большой объем доступных данных и вычислительная сложность моделей требует применения методов высокопроизводительных научных вычислений, использования аппаратных ускорителей и параллельных алгоритмов на вычислительных кластерах [21].

Одним из перспективных подходов к гибкому созданию интеллектуальных приложений на практике считается объединение в одном процессе моделирования данных, программной реализации и развертывание их на облачной инфраструктуре. Такой процесс поставки и предоставления в использование прогнозных моделей кратко называют MLOps. При разработке фреймворка в этом процессе были выделены следующие процедуры, выполняемые командами исследователей и разработчиков, схематично представленные на рис. 1.

На первом этапе моделирования ставится задача прикладного исследования, выполняется сбор наборов данных, исследовательский анализ данных, разработка моделей и экспериментальное исследование их качества.

На этапе реализации выполняется разработка интеллектуального приложения, в том числе алгоритма воспроизводимого обучения моделей, так называемого конвейера расчетов (pipeline), и веб-сервисов, представляющих результаты обучения по API. После этого выполняется оценка качества обучения с помощью конвейера расчетов. При необходимости, конвейер используется для дообучения и адаптации ранее разработанных моделей под новые задачи или новые наборы данных.

На третьем этапе применения проводится тестирование и верификация разработанных приложений и контроль качества. После этого, сборка и поставка модулей для развертывания на целевой платформе, предоставление доступа и подключение разнородных вычислительных ресурсов. После развертывания выполняется наблюдение за входными, выходными данными приложения и качеством его работы.

Основные задачи, на решение которых направлен MLOps фреймворк при использовании командами исследователей и разработчиков процессу, следующие: стандартизация описания конвейеров обучения, автоматизация дообучения и адаптации интеллектуальных приложений, автоматизация построения отчетов по результатам экспериментов, упрощение процедур создания и развертывания интеллектуальных приложений, повышение уровня технологической готовности исследовательских расчетных моделей, интерпретация результатов расчетов и построенных модельных прогнозов, облегчение поиска и обнаружения развернутых приложений и наборов данных.

Решение представленных сложных задач в комплексе требует проработки архитектуры программной системы и определение принципов ее разработки и развития.

Во-первых, рассматриваемый фреймворк использует декларативное описание структуры и конфигурации приложений, входящих в их состав конвейеров, веб-сервисов и других компонентов. В отличие от программного способа определения состояния, при де-

Рис. 1. Решаемые системой задачи автоматизации процесса MLOps

кларативном описании именно MLOps фреймворк определяет алгоритм разбора различий (reconciliation) между указанным разработчиком целевой и текущей конфигурацией приложения на MLOps платформе.

Во-вторых, возникает задача предоставления разработчикам удобного языка для описания такой конфигурации. В отличие от имеющегося низкоуровневого декларативного описания в инфраструктуре Kubernetes, ориентированной на широкий класс облачных приложений, MLOps фреймворк предлагает высокоуровневый предметно-ориентированный язык, специализированный для интеллектуальных приложений и программных средств машинного обучения. В основе языка лежит объектная модель, определяющая конфигурации и структуры компонентов приложений и вычислительных экспериментов. Для совместимости, язык описания конфигурации разработан как внутренний DSL с использованием средств расширения самой инфраструктуры Kubernetes.

Выполненное на предлагаемом языке описание структуры конвейера расчетов уже не зависит от реализации алгоритмов фреймворка и внутренней структуры и технологий MLOps платформы, что позволяет решать задачу выполнения распределенных вычислительных экспериментов и интегрировать разнородные вычислительные ресурсы. При этом MLOps платформа обеспечивает необходимую настройку, обмен данными и запуск компонентов конвейера расчетов на каждом используемом вычислителе.

Решение двух указанных задач позволит использовать возможности высокопроизводительных вычислительных систем и методы научных вычислений при создании интеллектуальных приложений в отраслях на основе объединенного процесса MLOps.

3. Реализация платформы и распределенные вычислительные эксперименты

MLOps платформа объединяет облачные вычислительные ресурсы под управление инфраструктуры Kubernetes, подключаемое через API облачное объектное хранилище, системные сервисы, такие как система управления версиями, реестр образов и пакетов, API

шлюзы, системы мониторинга, библиотеки программ, предоставляемые разработчикам-исследователям и другие подсистемы.

В данном разделе изложены основные решения по реализации платформы MLOps и об организации распределенных вычислительных экспериментов на ее основе.

3.1. Интеграция на основе общей объектной модели

Методология модельно-управляемой разработки программного обеспечения (model-driven software development) предусматривает разработку предметно-ориентированных языков (DSL), использование концептуальных, имитационных, расчетных моделей для формализации решаемых задач в выбранной предметной области, и автоматической трансляции таких моделей в программное обеспечение или интерпретации этих моделей. Система с открытым кодом MLDev реализует расширяемую модель вычислительного эксперимента и интерпретатор такой модели, следуя указанному подходу. Модель эксперимента задается декларативно на внутреннем DSL, который может быть расширен пользователем. В ходе интерпретации система MLDev обеспечивает интеграцию программных средств, используемых в вычислительном эксперименте, и воспроизводимое выполнение заданных на этом DSL конвейеров расчетов для проверки гипотез [14].

Декларативное описание структуры вычислительного эксперимента также применяется в инфраструктуре Kubernetes для указания конфигурации приложений, развертываемых на вычислительных ресурсах в центрах обработки данных. Предлагаемая Kubernetes объектная модель позволяет описывать широкий класс веб и серверных приложений, веб-сервисы и исполняемые фоновые задачи. Конфигурация приложения задается набором документов, называемых манифестами.

Таким образом, использование подходов, аналогичных примененным в MLDev, и разработка совместимой, понятной специалистам в машинном обучении объектной модели распределенных вычислительных экспериментов для инфраструктуры Kubernetes позволит интегрировать разнородные вычислительные ресурсы.

Суть предлагаемого решения состоит в следующем. Пользователи фреймворка записывают структуру размещаемого интеллектуального приложения, в том числе интеллектуальных сервисов, вычислительных экспериментов и наборов данных в виде структурированных документов — манифестов. Фреймворк выполняет трансляцию из этих манифестов в объектной модели фреймворка в объектную модель приложений Kubernetes и отслеживает их соответствие. Kubernetes приводит в соответствие указанную в модели целевую конфигурацию приложения и фактически исполняемые в облачной вычислительной среде процессы, тома дисков, сетевые ресурсы.

Структура интеллектуального программного приложения и его компонентов в предлагаемой объектной модели приведена на рис. 2 ниже.

Компонент — это любой размещаемый разработчиками-исследователями на MLOps платформе исполняемый программный модуль. Экземпляры компонента, как такового, создавать не допускается, в модели определено несколько видов компонентов, которые пользователь может использовать.

Для размещения расчетной модели пользователь описывает в конфигурации приложения интеллектуальный веб-сервис (ML-компонент). После создания, экземпляр ML-компонента предоставляет API для вызова расчетной модели в интерактивном режиме. ML-компонент не имеет значимого состояния, параметры модели, сторонние данные и дру-

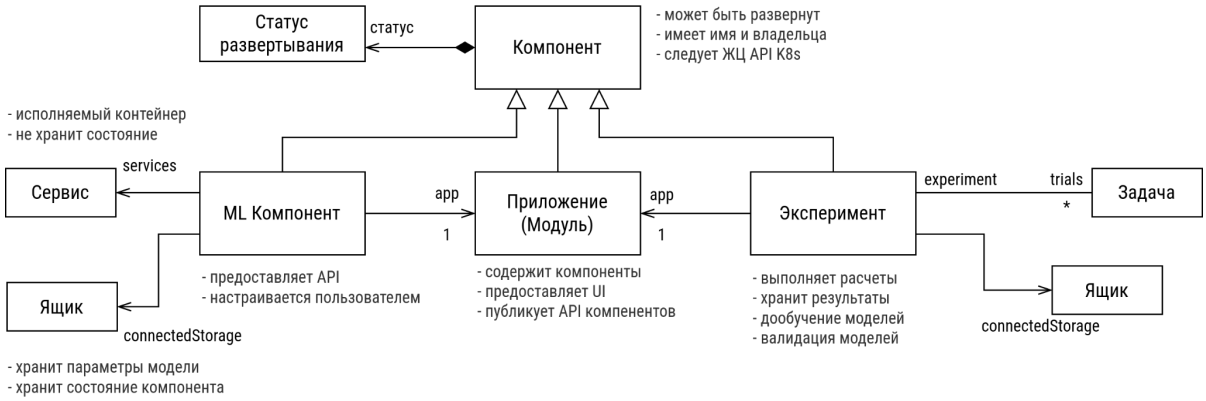

Рис. 2. Фрагмент объектной модели интеллектуального приложения с пояснениями

гие настройки хранятся в ящике данных (Ящик). Поэтому ML-компонент может быть перезапущен фреймворком в любое время.

Конвейер вычислительного эксперимента (Эксперимент) подразумевает исполнение длительной расчетной задачи, результатом которой будет набор файлов, размещаемых в ящиках данных. Этот набор файлов может содержать как отчеты об эксперименте, так и данные статического веб-сайта для отображения в пользовательском интерфейсе MLOps платформы. Расчетная задача может выполняться как на ресурсах платформы, на стороннем подключенном вычислительном кластере, так и выполнена заранее и ее результаты загружены до запуска эксперимента.

Приложение группирует другие компоненты, позволяя им взаимодействовать между собой и с пользовательским интерфейсом приложения. Например, дообученная в эксперименте модель может быть использована в ML-компоненте того же приложения. Изменяемые системой атрибуты приложения содержат сведения о статусе развертывания и готовности приложения к работе, ссылки и адреса URL, по которым можно получить доступ к компонентам приложения, использованные для развертывания версии источников из Git репозитория приложения.

Пользователи используют ML-компонент для предоставления результатов обучения, например, расчетных моделей, для использования в интерактивном режиме через формируемый фреймворком API, и принимающие на вход наборы данных эксперименты для выполнения конвейеров расчетов (pipelines) в фоновом режиме.

Структура вычислительного эксперимента в объектной модели фреймворка предусматривает как использование системы MLDev с несколькими конвейерами, подключаемыми модулями на этапах конвейера, модулями наборов данных, так и упрощенный режим с использованием одного конвейера в эксперименте. Трансляция структуры эксперимента с объектной модели MLDev в объектную модель фреймворка, записанную на внутреннем DSL, выполняется с использованием дополнительных программных модулей библиотеки MLDev.

Хранимые в приложении наборы данных могут быть использованы повторно другими приложениями на MLOps платформе и вне ее. Объектная модель фреймворка предусматривает возможность версионирования и добавления метаданных к данным в формате FAIR [22]. При этом версии набора данных становятся доступны по уникальным URL, пригодным для цитирования.

Для реализации указанного выше подхода фреймворк использует средства расширения объектной модели Kubernetes (Kubernetes operator framework) и определения вида ресурсов (CRD) при реализации объектной модели. Поэтому для создания экземпляров ресурсов из объектной модели фреймворка пользователям достаточно стандартного API Kubernetes.

Для создания ресурсов Kubernetes по добавленным пользователями манифестам фреймворк реализует программные компоненты управления ресурсами инфраструктуры Kubernetes (контроллеры CRD), для каждого класса объектной модели отдельный контроллер.

Определение ресурса CRD содержит контрольные соотношения (правила валидации) манифестов, проверяемые перед добавлением таких CRD самим Kubernetes. При обработке созданного CRD системой, обработчик системы дополнительно проверяет выполнение соотношений и ограничений, устанавливаемых моделью предметной области. В случае ошибки при добавлении, система сохраняет подробное сообщение об ошибке для просмотра пользователем стандартными средствами Kubernetes и устанавливает подходящий статус ресурса.

3.2. Выполнение распределенных вычислительных экспериментов

Для проведения вычислительного эксперимента в составе интеллектуального приложения необходимо подготовить манифесты с описанием конвейера расчетов и других параметров эксперимента в терминах ресурсов фреймворка.

Пример описания структуры эксперимента в объектной модели фреймворка приведен на рис. 3.

Объектная модель фреймворка содержит конвейер расчетов (ExperimentPipeline), который определяет этапы вычислительного эксперимента, ящик данных (DataBox) задает область хранения данных, используемых приложением, набор данных (DatasetComponent) определяет повторно используемый набор файлов для обучения расчетных моделей, интеллектуальный сервис (MLComponent) определяет программный сервис вычислений с использованием расчетных моделей машинного обучения, интерфейсный компонент (APIComponent) определяет API для доступа к сервисам эксперимента.

Каждому этапу конвейера сопоставлен образ контейнера, реализующий расчет. Этап принимает данные через входные переменные, выполняет расчет, возвращает данные через выходные переменные. Каждая переменная ассоциирована с ящиком данных, значения входных и выходных переменных передаются в виде файлов. При запуске эксперимента для каждой переменной область хранения данных подключенного ящика монтируется в контейнер этапа. Программа, выполняющая расчет, получает путь к файлам каждой переменной через переменные окружения. Таким образом, широкий класс программ, которые используют файлы в качестве входов и выходов, могут быть включены в конвейер как этап вычислительного эксперимента.

В текущей реализации фреймворк реализует хранение данных ящиков в объектном хранилище, совместимом с API S3.

После создания ресурсов эксперимента фреймворком, конвейер может быть исполнен. Для запуска необходимо отправить запрос к сервису конвейеров фреймворка, передав значения входных переменных или пути к ящикам данных в теле запроса и указав перечень запрашиваемых результатов.

Далее алгоритм выполнения конвейера эксперимента следующий. Получив запрос на запуск, сервис конвейеров проверяет входные данные, следуя структуре конвейера, форми-

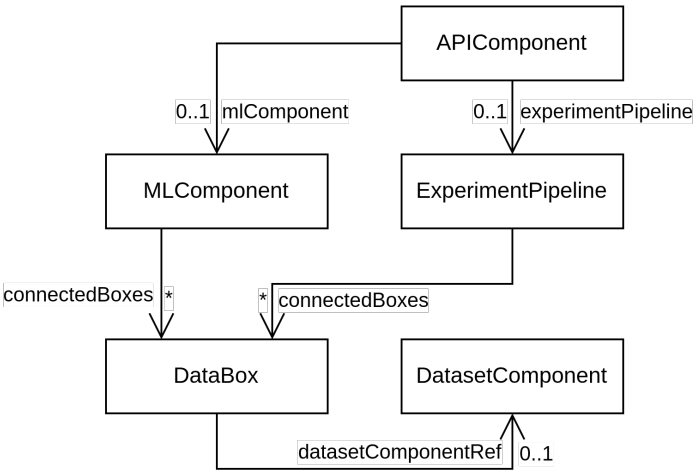

Рис. 3. Пример структуры эксперимента в объектной модели фреймворка

рует спецификацию вычислительной задачи Kubernetes, добавляя последним этап с проверкой результатов расчетов и создает ресурс задачи (Job), запуская таким образом конвейер. При этом этап конвейера сопоставляется отдельному контейнеру системы контейнеризации containerd при запуске этапа в инфраструктуре Kubernetes, или контейнера Singularity, при запуске на вычислительном кластере.

Каждому запуску присваивается уникальный идентификатор отслеживания. Это позволяет узнать статус выполнения конвейера и получить результаты расчетов. Фреймворк реагирует на события вычислительной задачи и обновляет статус запуска в системной базе данных.

3.3. Библиотека расширений конвейера эксперимента

Фреймворк предлагает готовые программные модули, которые включаются в вычислительный эксперимент при помощи механизма расширения объектной модели MLDev. Такие программные модули составляют библиотеку тегов фреймворка.

Теги библиотеки позволяют включить в вычислительный эксперимент готовую функциональность, объединить программные модули библиотеки фреймворка и разработанные программы этапов эксперимента в единый конвейер.

Для использования этих тегов в вычислительном эксперименте, необходимо описать эксперимент MLDev с тегом конвейера фреймворка UnipPipeline. Этапы конвейера указываются тегами контейнеров ContainerStage. В режиме подготовки запуска MLDev prepare реализация тега контейнера выполняет сборку образ контейнера, в режиме запуска run исполняет контейнер на основе собранного образа. Тег контейнера указывает исходный код, используемые библиотеки, команду или скрипт запуска и позволяет запускать в контейнере широкий класс программ и вычислительных экспериментов MLDev, используемых в расчетах в машинном обучении.

Библиотека фреймворка содержит следующие классы на языке Python, реализующие следующие теги разметки YAML для использования при написании схемы эксперимента:

- Тег SingularValues выполняет регуляризацию нейросетевых методов обучения путем ограничения собственных значений параметров модели.

- Тег StageSrc позволяет использовать в этапах конвейера набор данных DatasetComponent.
- Тег CharismaStage позволяет запускать этап конвейера в вычислительном кластере cHARISMa НИУ ВШЭ или любом другом кластере с планировщиком задач Slurm.
- Тег SynthesizeDataStage обучает статистическим свойствам набора данных генеративные модели и с их помощью генерирует синтетические данные.
- Тег Report создает по шаблону отчеты в виде веб-сайтов по проведенному вычислительному эксперименту.
- Тег UnipPipeline определяет конвейер расчетов для запуска на MLOps платформе и преобразует описание эксперимента MLDev в манифесты фреймворка.
- Тег AttentionInterpStage реализует алгоритм интерпретации результатов расчетов с использованием карты внимания в глубоких нейронных сетях.
- Тег StructuredInterpStage реализует алгоритмы интерпретации результатов классификации с помощью методов LIME и SHAP.

```
report: !Report                          params:
name: report                               template: *template
inputs:                                    # ...
  - &template !StageVar                    # другие параметры отчета
    name: template                       model:
    path: report/data/template             pic1: *pic1
    # ...                                  table: !Table
    # другие входные данные                  title: "Финальные метрики"
outputs:                                   data: *table
  - !StageVar                              # ...
    name: report                           # заполнение остальных полей
    path: report/data/report               # шаблона
```

Рис. 4. Пример фрагмента YAML с описанием отчета

3.4. Генератор отчетов Report

Тег Report библиотеки тегов фреймворка предоставляет функциональность для ускорения подготовки повторяющихся отчетов в формате веб-страниц о проведенных вычислительных экспериментах посредством параметризуемых шаблонов. Тег предоставляет возможность отобразить полученные в ходе работы конвейера работы файлы в формате текста, изображений, графиков или таблиц. Параметры тега в YAML манифесте определяют содержание отчета, в том числе задают однозначное соответствие файлов, создаваемых в ходе работы предыдущих этапов, и маркеров, с помощью которых данные будут размещены в отчете, а также их формат. Пример описания отчета приведен на рис. 4. Шаблон отчета и текст, который не будет изменяться при повторном запуске, можно указать в шаблоне отчета в формате Markdown. В этом же файле может быть указано расположение маркеров, определенных в манифесте.

В ходе подготовки отчета указанные в манифесте файлы будут преобразованы в соответствии с указанным ранее форматом в текст, изображения, графики или таблицы. Изоб-

ражения будут сжаты и закодированы в base64. При преобразовании графиков и таблиц они будут вначале сформированы в формате RST, а впоследствии будет сформирован соответствующий им HTML. После этого полученные данные будут вставлены в шаблон отчета. Конечным результатом будет HTML-файл со встроенными в него стилями, таблицами, графиками и изображениями для предоставления пользователю.

3.5. Представление результатов расчетов в виде сервисов

Результаты расчетов вычислительных экспериментов могут быть представлены MLOps платформой в виде RESTful веб-сервисов. Эти интеллектуальные сервисы позволяют использовать обученные ранее модели для обработки данных с помощью определяемого MLOps фреймворком API.

Для работы сервисам необходимо различное количество вычислительных ресурсов, которое может изменяться со временем в зависимости от количества запросов или объема обрабатываемых данных. Также могут возникнуть ситуации при которых сервисы резервируют вычислительные ресурсы, но не производят обработку данных ввиду отсутствия запросов через API сервиса. Это приводит к уменьшению утилизации выделенных вычислительных ресурсов MLOps платформы.

Задачу повышения эффективности использования вычислительных ресурсов MLOps платформы решает модуль автомасштабирования интеллектуальных сервисов. Модуль автомасштабирования интеллектуальных сервисов позволяет автоматически изменять количество реплик интеллектуального сервиса. В отличие от стандартных средств масштабирования для Kubernetes, модуль позволяет сократить количество реплик сервиса до нуля и увеличить при поступлении запроса через стандартный API сервиса.

Конфигурация алгоритма масштабирования определяет набор состояний целевого сервиса. Состояния сервиса в свою очередь определяют необходимое количество реплик и набор правил перехода в другие состояния. Перевод сервиса из одного состояния в другое модуль масштабирования осуществляет согласно правилам перехода на основе метрик, которые модуль масштабирования получает от Сервера метрик через определенные промежутки времени. Структура обменов данными между частями фреймворка и модуля приведена на рис. 5.

Особым образом обрабатывается выключение сервиса при его неиспользовании. Выключение интеллектуального сервиса может быть осуществлено как на основе правил перехода, задаваемых в конфигурации масштабирования, так и при получении сигнала от Менеджера оповещений при отсутствии запросов клиентов.

При выключении интеллектуального сервиса запросы к API сервиса перенаправляются на модуль автомасштабирования. Запуск сервиса после выключения производится при получении перенаправленного запроса модулем. Запуск сервиса может занять значительное время и может привести к ошибке истечения таймаута HTTP соединения. Чтобы избежать получения этой ошибки клиентом интеллектуального сервиса модуль автомасштабирования MLOps платформы периодически перенаправляет клиента на тот же адрес. Таким образом модуль будет поддерживать активное соединение, пока интеллектуальный сервис не будет полностью готов обрабатывать запросы клиента. После полного запуска целевого сервиса модуль масштабирования восстанавливает маршруты обработки запросов и только после этого перенаправляет клиента на целевой сервис.

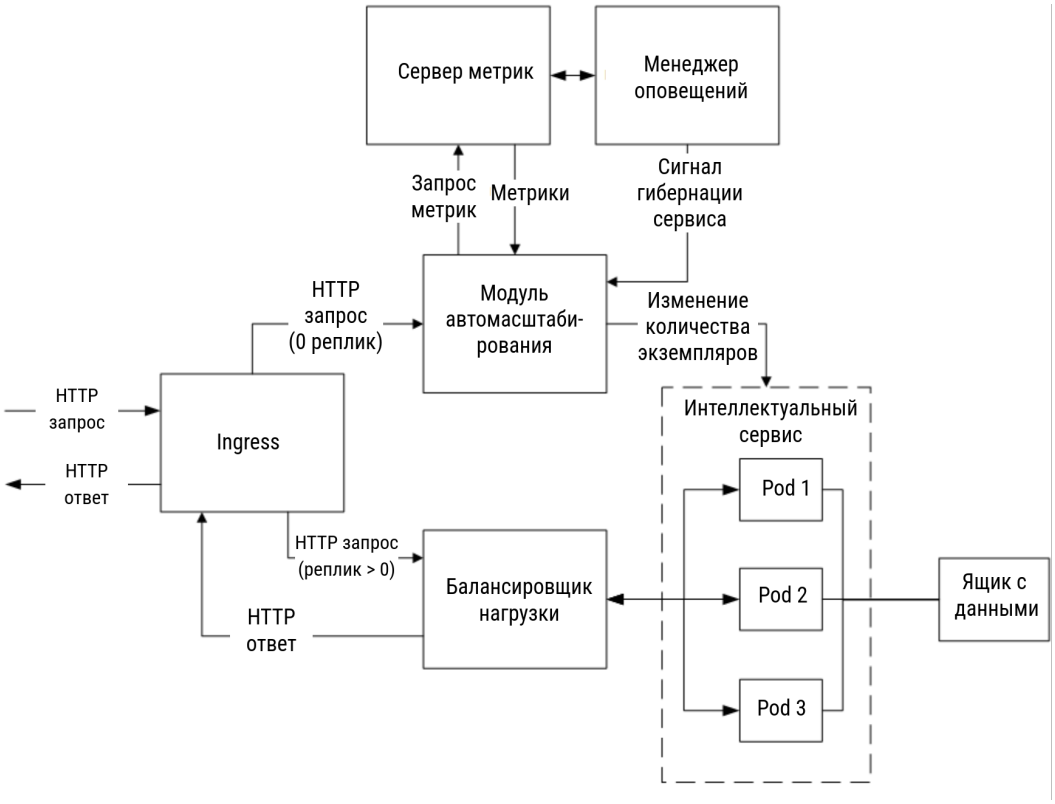

Рис. 5. Структура обменов данными при масштабировании интеллектуального сервиса

3.6. Интеграция суперкомпьютера cHARISMa

Основная способ запуска выполнения задач на вычислительном кластере cHARISMa использует систему управления заданиями SLURM. Данная система позволяет автоматически запустить задачу на подходящих узлах или поставить задачу в очередь, если в данный момент ресурсы недоступны.

Для взаимодействия конвейеров MLOps платформы с вычислительным кластером в рамках работы решены следующие задачи: передача самой вычислительной задачи на головной узел кластера, загрузка входных данных для вычислений, отслеживание текущего состояния задачи и выгрузку выходных данных после завершения.

Для передачи данных фреймворк использует соединение SSH. Во-первых, такой способ доступен для подключения к кластеру обычным пользователям, и он позволяет обеспечить защиту передаваемым данным. Во-вторых, тот же способ подключения используется для формирования расчетного задания для SLURM на головном узле в окружении пользователя, от имени которого фреймворк производит подключение.

Запуск задач через SLURM на кластере cHARISMa возможен в двух режимах: sbatch — запуск задач в «пакетном» режиме и srun — запуск в интерактивном режиме. Так как основное преимущество режима sbatch — это возможность подготовки окружения через скрипт, то этот способ и был выбран для дальнейшей реализации.

Для решения задачи передачи окружения пользовательской расчетной задачи фреймворк использует образы контейнеров Singularity в формате SIF, принимаемые вычислительным кластером. Данные решение позволяет сделать выполнение кода на кластере безопасным и дополнительно позволяет пользователю фреймворка подготовить вычислительную задачу без учета особенностей доступного на вычислительном кластере окружения, про-

тестировать запуск задачи на своей стороне. При подготовке конвейера с использованием системы MLDev и библиотеки тегов фреймворка тег CharismaStage создает SIF образ контейнера непосредственно из образа контейнера Docker, что упрощает работу пользователей уже имеющих опыт работы с контейнерами.

Для решения задачи мониторинга процесса расчетов необходимо в дополнение к системе HPC TaskMaster иметь постоянный процесс для слежения за вычислительной задачей. Теоретически данный процесс мог быть запущен и на головном узле вычислительного кластера и отправлять статусы на внешний адрес MLOps платформы. Однако такое решение требовало, чтобы этот процесс работал постоянно, что не соответствовало концепции работы с вычислительным кластером.

Вместо этого процесс опроса статуса вычислительной задачи запускается как часть конвейера расчетов и через соединение SSH просматривает содержимое файла со статусом, доступное на головном узле. В зависимости от текущего состояния задачи sbatch в данном файле происходит постепенное изменение статусов: running, error и finished. Далее этот статус транслируется в статусы этапа конвейера расчетов в объектной модели фреймворка. В случае отключения соединения SSH фреймворк выполняет повторное переподключение через некоторое время ожидания.

После завершения процесса на вычислительном кластере выходные данные копируются через SSH обратно на узел MLOps платформы и передаются далее по конвейеру через ящики данных.

Таким образом весь процесс взаимодействия управляемой фреймворком MLOps платформы с вычислительным кластером можно описать следующими шагами:

- подготовка описания конвейера с входными данными и исполняемыми процессами;
- сборка SIF контейнера с реализацией вычислительной задачи;
- подготовка реквизитов для подключения по SSH и к репозиторию образов контейнеров;
- загрузка контейнера в ящик данных в хранилище S3;
- загрузка входные данные в ящик данных в хранилище S3;
- запуск конвейера на выполнение через подключение к головному узлу кластера.

Для анализа эффективности использования вычислительных ресурсов на вычислительном кластере cHARISMa в НИУ ВШЭ использована специализированная система мониторинга эффективности HPC TaskMaster [23]. Система позволяет автоматически обнаруживать неэффективные и некорректно запущенные пользовательские задачи за счет настройки подходящих индикаторов, параметров и тегов задач [24]:

- непараллельные задачи, ошибочно запущенные в параллельном режиме;
- неравномерное распределение ресурсов между узлами; простои выделенных вычислительных ресурсов;
- низкая загрузка CPU или GPU;
- слишком объемная запись данных на локальный SSD;
- слишком частые обращения к СХД и другие.

Использование системы HPC TaskMaster на вычислительном кластере cHARISMa более чем в 5 раз снижает время ожидания задач в очереди и освобождает более 7% вычислительных ресурсов в пользу эффективных вычислительных задач.

Для обеспечения взаимодействия с MLOps фреймворком система HPC TaskMaster расширена путем реализации REST API на базе Django REST Framework. Данное расширение

позволяет MLOps фреймворку получать детальную информацию о выполняемых задачах, включая основные сведения о задаче и запрошенных ресурсах, а также агрегированные метрики использования ресурсов. Данные в API представлены в формате JSON, который далее передается и обрабатывается в MLOps фреймворке. Для обеспечения безопасности и изоляции данных в API HPC TaskMaster реализован механизм авторизации на основе WckeyToken. Доступ к информации о задачах ограничивается значением wckey — специального идентификатора, который указывается фреймворком при постановке задачи в очередь планировщика SLURM. API предоставляет данные только о тех задачах, которые имеют соответствующий wckey, что позволяет организовать контролируемый доступ к задачам, запущенным через MLOps фреймворк.

4. Пример применения для реализации MLOps платформы

Проект «MLOps-система исполнения и мониторинга ИИ-моделей» выполняется в составе стратегического проекта № 4 «Цифровая трансформация: технологии, эффекты, эффективность» программы академического лидерства «Приоритет 2030». Проект направлен на создание и развитие центров компетенций по развитию и применению технологий искусственного интеллекта в НИУ ВШЭ и их интеграцию в образовательный процесс, в сферах управления и медицины совместно с партнерами. Разрабатываемая в проекте MLOps платформа позволяет автоматизировать процессы развертывания, поддержки и эксплуатации цифровых сервисов на основе искусственного интеллекта.

Ключевая ценность для пользователей и разработчиков моделей машинного обучения состоит в снижении трудоемкости развертывания интеллектуальных приложений и сервисов на основе данных информационных систем НИУ ВШЭ; снижении модельного риска при применении расчетных моделей за счет своевременного определения снижения качества их работы и изменений в распределениях входных данных, повышении устойчивости работы сервисов за счет балансировки и предоставления релевантных вычислительных ресурсов.

На платформе развернуты интеллектуальные сервисы для обнаружения текста, написанного с помощью нейросетей, подготовки юридических документов на основе больших языковых моделей, сервис прогнозирования и классификации тикетов в системе управления задачами Jira, интеллектуальный ассистент сотрудника учебного офиса, отвечающий на вопросы по образовательному процессу, интеллектуальный сервис анализа изображений глазного дна и другие. В 2024 г. на платформе развернуто 8 интеллектуальных приложений и сервисов, с ожидаемым приростом по 3–5 сервисов в год до 2030 г.

4.1. Приложение обучения моделей классификации изображений

Для демонстрации работы и расчета показателей эффективности проведен вычислительный эксперимент в ходе которого решается задача идентификации объектов на изображениях. В рамках вычислительного эксперимента обучена модель RTMDet [25], предназначенная для решения задачи идентификации на наборе данных COCO. Модель реализована в модуле синтеза моделей машинного зрения для детектирования объектов и действий (AutoOD) [26, 27].

Набор данных состоит из порядка 5 тыс. примеров, разделенных на две выборки. В обучающей выборке 4 тыс. изображений и 1 тыс. в тестовой выборке. Также в набор данных входит разметка изображений в формате COCO с указанием координат объектов.

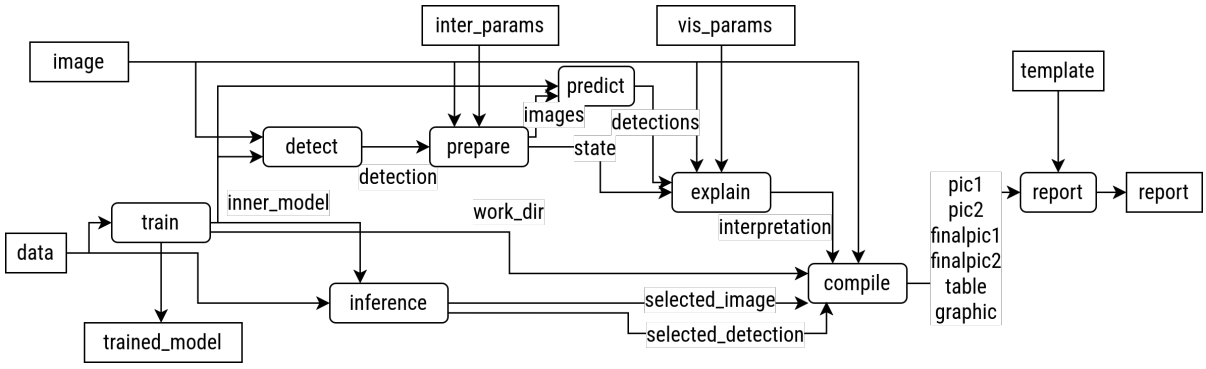

Рис. 6. Схема этапов конвейера расчета в эксперименте

Использованы следующие критерии качества модели — средняя точность (mean average precision) при пороге отношения пересечения к объединению 0.5 и 0.75 между рассчитанной моделью областью нахождения объекта на изображении и областью в разметке.

4.2. План и структура вычислительного эксперимента

Конвейер расчетов вычислительного эксперимента состоит из восьми этапов. Схема конвейера показана на рис. 6. Первый этап — обучение (train) на вычислительном кластере cHARISMa. На этом этапе используется базовый модуль AutoOD для обучения модели идентификации объектов на изображениях [27]. На вход передаются данные для обучения (data), в качестве результатов модуль возвращает обученную модель (trained_model) и передает ее в следующие этапы (inner_model). Также модуль возвращает содержимое рабочей директории обучения модели (work_dir), которая используется для анализа процесса обучения.

На втором этапе выполняется применение расчетной модели (inference). Обученная модель (inner_model) используется для идентификации объекта заданного класса на изображении из набора данных (data). Выбранное изображение (selected_image) и результат обнаружения (selected_detection) передаются далее в этап сбора данных (compile) для включения в отчет о запуске.

Этапы с третьего по шестой — это шаги процесса построения интерпретации результатов применения расчетной модели, выполняемые повторно используемым MLOps модулем библиотеки модулем фреймворка. Третий этап — обнаружение объекта (detect). В этом этапе обученная модель (inner_model) используется для идентификации объекта на исходном изображении (image). Результат идентификации (detection) передается в следующий, четвертый, этап — подготовку к интерпретации (prepare). На этапе подготовки на основе одного исходного изображения создается множество зашумленных версий (images) с вырезанной областью вокруг объекта. Количество созданных изображений, а также вид шума определяются параметрами интерпретации (inter_params). Также на этапе подготовки к интерпретации сохраняется информация о соответствии зашумленных участков изображениям (state). Пятый этап — идентификация объектов на наборе созданных изображений (predict). На пятом этапе обученная модель (inner_model) используется для обнаружения объектов на зашумленных изображениях (images) этапа подготовки к интерпретации. Результаты идентификации (detections) передаются в этап построения интерпретации. Шестой этап — этап расчета интерпретации. На основе результатов идентификации (detections), и с использованием информации о соответствии зашумленных участков изображению (state), рассчитыва-

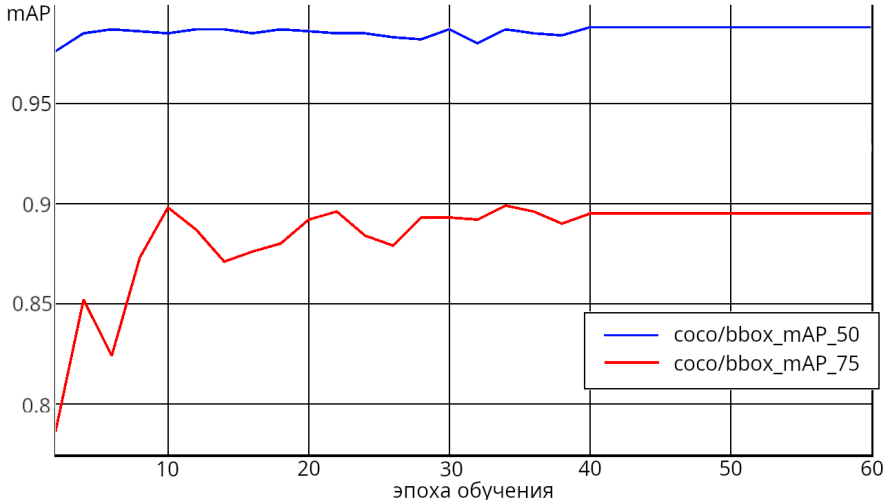

Рис. 7. Кривые обучения модели идентификации объектов в эксперименте в зависимости от пройденных эпох (по горизонтали); coco/bbox_mAP_50 — средняя точность при пороге 50%, coco/bbox_mAP_75 — средняя точность при пороге 75%

ются области, наиболее сильно влияющие на точность работы модели (interpretation). Для построения визуализации используется само исходное изображение (image) и параметры визуализации (vis_params).

Седьмой этап — сбор данных для включения в отчет (compile). В качестве входных данных этот этап принимает исходное изображение (image), результат идентификации объекта на исходном изображении (detection), изображение, выбранное на этапе применения модели (selected_image), результат идентификации на нем (selected_detection), рабочую директорию обучения модели (work_dir) и интерпретацию работы модели на исходном изображении (interpretation). На данном этапе создается визуализация идентификации объектов на двух входных изображениях и формируются таблицы с показателями точности обучения. Все эти данные передаются в этап построения отчета.

Восьмой этап — построение отчета с помощью поставляемой в составе фреймворка расширенной версии библиотеки mldev-reporting [14]. Из этапа сбора данных передаются два исходных изображения, два изображения с идентификацией объектов, изображение с интерпретацией, таблица финальных показателей качества и таблица изменения показателей в процессе обучения. Также в этот этап передается шаблон отчета, на основе которого в ходе этапа формируется финальный отчет о запуске эксперимента в виде веб-сайта (report).

4.3. Результаты и анализ эффективности

Для демонстрации работы и проведения измерений конвейер обучения исполнен со следующими параметрами: 60 эпох обучения, 8 изображений в одном минибатче, начальный шаг обучения 0.004. Полученные кривые обучения приведены на рис. 7. Значения показателей качества составили coco/mAP_0.5 = 0.976, coco/mAP_0.75 = 0.786.

Для анализа эффективности использования вычислительных ресурсов платформы воспользуемся методов эталонных систем [28]. В качестве эталона рассмотрим гипотетический способ организации расчетов на тех же вычислительных ресурсах, при котором пользовательская задача задействует все доступные ресурсы без накладных расходов и увеличивает

объем решаемой задачи за то же время. Для данного расчета считаем задачу произвольно делимой между вычислителями.

Обозначим через p_{ijt} доступную мощность вычислительного ресурса j для решения задачи i в интервал времени t продолжительности Δ_t. Пусть в тот же момент времени только доля c_{ijt} этого выделенного ресурса задействована для расчетов, остальная часть не используется или составляет накладные расходы. Тогда эффективность E решения группы вычислительных задач по отношению к рассмотренной эталонной модели составит

$$E = \frac{\sum_{ijt} p_{ijt} c_{ijt} \Delta_t}{\sum_{ijt} p_{ijt} \Delta_t}. \tag{1}$$

Данную эффективность можно интерпретировать, как отношение выполненной полезной работы в выделенные интервалы времени к теоретически возможному объему работы.

В эксперименте для выполнения этапов detect, explain, inference, predict, prepare, compile, report конвейера расчета использован узел с одним vCPU ядром Intel® Xeon® Gold 6338 (примем $p_{ijt} = 1$) и ускорителем NVIDIA® Tesla® T4 с 16 ГБ RAM (оценочно при экспериментальном сравнении, $p_{ijt} \approx 10$), и на вычислительном кластере cHARISMa для этапа train использована выделенная Slurm часть узла с четырьмя ядрами CPU (без hyper–threading) на Intel Xeon Gold 6152 2.1–3.7 GHz (оценочно, $p_{ijt} \approx 1$) и одной NVIDIA Tesla V100 32 GB (оценочно, $p_{ijt} \approx 40$). Для этапа сборки build использован ПК разработчика с восемью vCPU ядрами Intel i5-8300H (примерно $p_{ijt} \approx 1$). Доступная пропускная способность коммуникационной сети между узлами и с головным узлом кластера не ограничивалась и не измерялась. Результаты расчетов показателей эффективности приведены в табл. 1, названия этапов соответствуют схеме на рис. 3. Указанное в таблице время расчета измеряется от момента выделения вычислительных ресурсов задаче и до их освобождения.

Таблица 1. Эффективность E_i использования вычислительных ресурсов на этапе расчета i

Этап i	E_i	E_i^{cpu}	p_{it}^{cpu}	E_i^{gpu}	p_{it}^{gpu}	Время, с.
(build)	0.00	0.00	≈ 8	0.00	0	833
train	0.57	0.70	4	0.56	≈ 40	12240
(charisma-stage)	0.00	0.00	0.5	0.00	0	13053
detect	0.08	0.93	1	0.00	≈ 10	13
explain	0.04	0.41	1	0.00	≈ 10	5
inference	0.05	0.58	1	0.00	≈ 10	22
predict	0.37	0.96	1	0.31	≈ 10	69
prepare	0.04	0.48	1	0.00	≈ 10	241
compile	0.02	0.27	1	0.00	≈ 10	3
report	0.02	0.27	1	0.00	≈ 10	5

Учитывая весьма приблизительные оценки сравнительной производительности использованных вычислителей и сделанные предположения о делимости задачи, получаемое оценочное значение эффективности всего расчета лежит около $E \approx 0.55..0.56$ при вариации оценочных значений производительности вычислительных ресурсов от 67% до 150% и составляет примерно 97% от эффективности расчетов на этапе обучения.

Анализируя результаты расчетов, следует отметить следующее. Первое, использование постоянно работающего процесса для отслеживания поставленной в очередь задачи, обозначенного в таблице charisma-stage, не вносит существенного вклада в эффективность всего конвейера. Сборка и подготовка эксперимента build средствами библиотеки тегов фреймворка на ПК разработчика также не вносит существенного вклада в эффективность. Второе, подключение вычислительного кластера для запуска коротких задач менее 10 минут может быть неэффективным. В этом случае предпочтительно использовать облачные вычислительные ресурсы центра обработки данных.

Тем не менее, среди способов дальнейшего повышения эффективности можно выделить следующие:

- Расчет оценок эффективности в зависимости от используемых ресурсов для запускаемых задач в помощь разработчикам при планировании их использования на вычислительном кластере.
- Разделение исполнения этапов между разными узлами для исключения простоя высокопроизводительных ресурсов, когда они не запрашиваются задачей этапа.

Заключение

В работе рассмотрена актуальная проблема создания технологии создания программных средств интеллектуального анализа и обработки данных для решения отдельных прикладных задач. Исследованы задачи применения и автоматизации процесса MLOps на базе единой гибридной облачной вычислительной платформы, в том числе интеграции разнородных облачных, локальных и высокопроизводительных вычислительных ресурсов для проведения распределенных экспериментов, реализации конвейеров настройки моделей машинного обучения, решения задачи обработки и анализа данных

Основные результаты работы состоят в создании на основе модельно-ориентированного подхода (MDSD) расширяемой объектной модели интеллектуальных приложений, ее реализация с применением библиотеки MLDev предоставления пользователям возможности создания собственных конвейеров машинного обучения, выполняемых в распределенной среде. Для описания структуры и конфигурации интеллектуальных приложений разработан формальный язык на основе YAML и разработан программный фреймворк, управляющий исполнением приложений в инфраструктуре Kubernetes.

На примере задачи классификации и детектирования объектов на изображениях продемонстрирована возможность расширения пользователями объектной модели и создание сложных многоэтапных конвейеров расчетов прозрачным для пользователя образом. По результатам предварительных экспериментов и расчетов эффективности, вносимые MLOps фреймворком дополнительные накладные расходы невелики по сравнению с затратами на обучение расчетных моделей. В настоящее время разработанный фреймворк используется в MLOps платформе НИУ ВШЭ.

Благодарности

Исследование выполнено с использованием суперкомпьютерного комплекса НИУ ВШЭ [29].

Литература

1. Кореньков В. Грид-технологии: статус и перспективы // Вестник Международной академии наук. Русская секция. 2010. № 1. С. 41–44. DOI: 10.3997/2214-4609.20142827.
2. Pimenov A., Fedorov I., Bezzateev S. Fog computing architecture using blockchain technology // Information and Control Systems. 2022. Oct. No. 5. P. 40–48. DOI: 10.31799/1684-8853-2022-5-40-48.
3. Sukhoroslov O.V., Afanasiev A. Everest: A Cloud Platform for Computational Web Services // CLOSER. 2014. P. 411–416. DOI: 10.5220/0004941404110416.
4. Центр искусственного интеллекта – Национальный исследовательский университет «Высшая школа экономики». 2024. URL: https://cs.hse.ru/aicenter/.
5. Antonenko V., Chupakhin A., Kolosov A., *et al.* On HPC and Cloud Environments Integration // Performance Evaluation Models for Distributed Service Networks. Springer, 2021. P. 159–185. DOI: 10.1007/978-3-030-67063-4_8.
6. Ejarque J., Badia R.M., Albertin L., *et al.* Enabling dynamic and intelligent workflows for HPC, data analytics, and AI convergence // Future generation computer systems. 2022. Vol. 134. P. 414–429. DOI: 10.1016/j.future.2022.04.014.
7. Сухорослов О. Комбинированное использование высокопроизводительных ресурсов и Грид-инфраструктур в рамках облачной платформы Everest // Суперкомпьютерные дни в России. 2015. С. 706–711.
8. Велихов В.Е., Климентов А.А., Машинистов Р.Ю. и др. Интеграция гетерогенных вычислительных мощностей НИЦ «Курчатовский институт» для проведения масштабных научных вычислений // Известия Южного федерального университета. Технические науки. 2016. № 11 (184). С. 88–100.
9. Кутовский Н., Мицын В., Мошкин А. и др. Интеграция распределенных гетерогенных вычислительных ресурсов для эксперимента mpd с помощью DIRAC Interware // Физика элементарных частиц и атомного ядра. 2021. Т. 52, № 4. С. 999.
10. Feoktistov A.G., Sidorov I.A., Sergeev V.V., *et al.* Virtualization of heterogeneous HPC-clusters based on OpenStack platform // Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2017. Vol. 6, no. 2. P. 37–48. DOI: 10.14529/cmse170203.
11. Silva R.F.D., Badia R.M., Bard D., *et al.* Frontiers in scientific workflows: Pervasive integration with high-performance computing // Computer. 2024. Vol. 57, no. 8. P. 36–44. DOI: 10.1109/mc.2024.3401542.
12. Stubbs J., Cardone R., Packard M., *et al.* Tapis: An API platform for reproducible, distributed computational research // Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Vol. 1. Springer. 2021. P. 878–900. DOI: 10.1007/978-3-030-73100-7_61.
13. Воронцов К., Игловиков В., Стрижов В. и др. Проблемы проведения экспериментов и воспроизводимости исследований в науках о данных // Труды Московского физико-технического института. 2021. Т. 13, № 2 (50). С. 100–108. DOI: 10.53815/20726759_2021_13_2_100.

14. Khritankov A., Pershin N., Ukhov N., Ukhov A. MLDev: Data Science Experiment Automation and Reproducibility Software // International Conference on Data Analytics and Management in Data Intensive Domains. Springer. 2021. P. 3–18. DOI: 10.1007/978-3-031-12285-9_1.

15. Alam K., Roy B. Challenges of provenance in scientific workflow management systems // 2022 IEEE/ACM Workshop on Workflows in Support of Large-Scale Science (WORKS). IEEE. 2022. P. 10–18. DOI: 10.1109/works56498.2022.00007.

16. Dhruv A., Dubey A. Managing software provenance to enhance reproducibility in computational research // Computing in Science & Engineering. 2023. Vol. 25, no. 3. P. 60–65. DOI: 10.1109/mcse.2023.3314288.

17. Зыбин Р., Швецова В., Бадалян Д. и др. Облачная среда «Асперитас». 2022. Свидетельство о государственной регистрации программы для ЭВМ RU 2022682679.

18. Грушин Д., Самоваров О., Хашба Э. SaaS платформа организации единой web-среды исследований, разработок и образования «Fanlight». 2018. Свидетельство о государственной регистрации программы для ЭВМ RU 2018615444.

19. Насонов Д., Бутаков Н., Бухановский А. и др. Технология организации управления и обработки больших данных – DataMall. 2020. Свидетельство о государственной регистрации программы для ЭВМ RU 2020664222.

20. Kreuzberger D., Kühl N., Hirschl S. Machine learning operations (MLOPS): Overview, definition, and architecture // IEEE Access. 2023. Vol. 11. P. 31866–31879. DOI: 10.1109/access.2023.3262138.

21. Тютляева Е.О., Одинцов И.О., Мармузов Г.В. и др. Тенденции развития вычислительных узлов современных суперкомпьютеров // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 3. С. 92–114. DOI: 10.14529/cmse190305.

22. Wilkinson M.D., Dumontier M., Aalbersberg I.J., *et al.* The FAIR Guiding Principles for scientific data management and stewardship // Scientific Data. 2016. Vol. 3, no. 1. P. 1–9. DOI: 10.1038/sdata.2016.18.

23. Kostenetskiy P., Shamsutdinov A., Chulkevich R., *et al.* HPC TaskMaster - Task Efficiency Monitoring System for the Supercomputer Center // Parallel Computational Technologies / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer International Publishing, Jan. 2022. P. 17–29. DOI: 10.1007/978-3-031-11623-0_2.

24. Kostenetskiy P., Kozyrev V., Chulkevich R., Raimova A. Enhancement of the Data Analysis Subsystem in the Task-Efficiency Monitoring System HPC TaskMaster for the cHARISMa Supercomputer Complex at HSE University // Parallel Computational Technologies / ed. by L. Sokolinsky, M. Zymbler, V. Voevodin, J. Dongarra. Cham: Springer Nature Switzerland, 2024. P. 49–64. DOI: 10.1007/978-3-031-73372-7_4.

25. Lyu C., Zhang W., Huang H., *et al.* RTMDet: An Empirical Study of Designing Real-Time Object Detectors // CoRR. 2022. Vol. abs/2212.07784. DOI: 10.48550/ARXIV.2212.07784. arXiv: 2212.07784.

26. Сластников С., Чертова Э. Модуль синтеза моделей машинного зрения для детектирования объектов и действий. 2024. URL: https://cs.hse.ru/aicenter/rid_detection.

27. Сластников С., Чертова Э. Модуль синтеза моделей машинного зрения для детектирования объектов и действий. 2023. Свидетельство о государственной регистрации программы для ЭВМ RU 2023660157.

28. Хританков А.С. Метод анализа производительности распределенных приложений на основе эталонных моделей // Параллельные вычислительные технологии (ПаВТ'2011). 2011. С. 343–354.

29. Kostenetskiy P., Chulkevich R., Kozyrev V. HPC Resources of the Higher School of Economics // Journal of Physics: Conference Series. 2021. Jan. Vol. 1740, no. 1. P. 012050. DOI: 10.1088/1742-6596/1740/1/012050.

Хританков Антон Сергеевич, к.ф.-м.н., факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

Полежаев Валентин Александрович, факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

Жуликов Григорий Александрович, факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

Галынчик Максим Сергеевич, факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

Климин Никита Андреевич, отдел разработки программных систем, МИЭМ ВШЭ (Москва, Российская Федерация)

Сахаров Кирилл Евгеньевич, отдел разработки программных систем, МИЭМ ВШЭ (Москва, Российская Федерация)

Минченков Виктор Олегович, отдел разработки программных систем, МИЭМ ВШЭ (Москва, Российская Федерация)

Спирин Иван Вячеславович, факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

Крупнов Иван Игоревич, факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

Якушева Софья Федоровна, к.т.н., факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

Маратканова Александра Сергеевна, факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

Козырев Вячеслав Иванович, отдел суперкомпьютерного моделирования, НИУ ВШЭ (Москва, Российская Федерация)

Костенецкий Павел Сергеевич, к.ф.-м.н., доцент, отдел суперкомпьютерного моделирования, НИУ ВШЭ (Москва, Российская Федерация)

Салех Хади Мухаммед, к.т.н., факультет компьютерных наук, НИУ ВШЭ (Москва, Российская Федерация)

DOI: 10.14529/cmse250203

DISTRIBUTED COMPUTATIONAL EXPERIMENTS IN THE MLOPS PLATFORM OF HSE UNIVERSITY

© 2025 A.S. Khritankov, V.A. Polezhaev, G.A. Zhulikov, M.S. Halynchik, N.A. Klimin, K.E. Sakharov, V.O. Minchenkov, I.V. Spirin, I.I. Krupnov, S.F. Yakusheva, A.S. Maratkanova, V.I. Kozyrev, P.S. Kostenetskiy, H.M. Salekh

National Research University Higher School of Economics
(11 Pokrovskiy blvd., Moscow, 109028 Russia)
E-mail: akhritankov@hse.ru

Received: 19.05.2025

Despite the wide spread and successful application of data mining and processing tools for solving individual applied problems, the problem of developing a technology for creating such software tools has not yet been solved. In the context of a unified MLOps process for creating machine learning technologies, this paper considers the emerging problems of automating and executing distributed computing experiments on a hybrid cloud computing platform. The MLOps platform being developed at HSE University is designed to deploy intelligent services and data analysis software. The platform shall manage heterogeneous resources available locally and in the cloud environment and combine them with the resources of the HSE cHARISMa computing cluster managed with Slurm. Thus, relevant is the problem of integrating the specified resources for conducting computational experiments, implementing pipelines for setting up machine learning models, solving problems of data processing and analysis. The features of the problem being solved are the consideration of the computation process as an integral part of the technology for creating intelligent services, the need for using heterogeneous resources for this technology, and the use of the hybrid platform for the execution of computations. The paper proposes a solution to the problem of integrating computations and presents the results of testing the solution for intelligent services. We show the feasibility of such integration of heterogeneous resources in the same computational experiment based on an object model of the experiment extended by the user and a domain-specific language for its specification, and resolve the issues of dynamic management of the deployment of intelligent applications, integration of data processing pipelines, services and data sets for performing distributed computational experiments.

Keywords: distributed computing experiments, machine learning, cloud technologies, MLOps.

FOR CITATION

Khritankov A.S., Polezhaev V.A., Zhulikov G.A., *et al.* Distributed Computational Experiments in the MLOps Platform of HSE University. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2025. Vol. 14, no. 2. P. 42–66. (in Russian) DOI: 10.14529/cmse250203.

References

1. Korenkov V. GRID technologies: status and prospectives. Herald of the International Academy of Science. Russian Section. 2010. No. 1. P. 41–44. (in Russian) DOI: 10.3997/2214-4609.20142827.
2. Pimenov A., Fedorov I., Bezzateev S. Fog computing architecture using blockchain technology. Information and Control Systems. 2022. Oct. No. 5. P. 40–48. (in Russian) DOI: 10.31799/1684-8853-2022-5-40-48.

3. Sukhoroslov O.V., Afanasiev A. Everest: A Cloud Platform for Computational Web Services.. CLOSER. 2014. P. 411–416. DOI: 10.5220/0004941404110416.

4. Centre of Artificial Intelligence – HSE University. 2024. URL: https://cs.hse.ru/aicenter/ (in Russian).

5. Antonenko V., Chupakhin A., Kolosov A., *et al.* On HPC and Cloud Environments Integration. Performance Evaluation Models for Distributed Service Networks. Springer, 2021. P. 159–185. DOI: 10.1007/978-3-030-67063-4_8.

6. Ejarque J., Badia R.M., Albertin L., *et al.* Enabling dynamic and intelligent workflows for HPC, data analytics, and AI convergence. Future generation computer systems. 2022. Vol. 134. P. 414–429. DOI: 10.1016/j.future.2022.04.014.

7. Sukhoroslov O. Combined use of high-performance resources and Grid infrastructures within the Everest cloud platform. Supercomputer Days in Russia. 2015. P. 706–711. (in Russian).

8. Velikhov V., Klimentov A., Mashinistov R., *et al.* Integration of heterogeneous computing resources at NRI "Kurchatov Institute" for large-scale scientific computations. Izvestiya SFedU. Engineering Sciences. 2016. No. 11 (184). P. 88–100. (in Russian).

9. Kutovskiy N., Mitsyn V., Moshkin A., *et al.* Integration of distributed heterogeneous computing resources for the mpd experiment with DIRAC Interware. Physics of Particles and Nuclei. 2021. Vol. 52, no. 4. P. 999. (in Russian).

10. Feoktistov A.G., Sidorov I.A., Sergeev V.V., *et al.* Virtualization of heterogeneous HPC-clusters based on OpenStack platform. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2017. Vol. 6, no. 2. P. 37–48. (in Russian) DOI: 10.14529/cmse170203.

11. Silva R.F.D., Badia R.M., Bard D., *et al.* Frontiers in scientific workflows: Pervasive integration with high-performance computing. Computer. 2024. Vol. 57, no. 8. P. 36–44. DOI: 10.1109/mc.2024.3401542.

12. Stubbs J., Cardone R., Packard M., *et al.* Tapis: An API platform for reproducible, distributed computational research. Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Vol. 1. Springer. 2021. P. 878–900. DOI: 10.1007/978-3-030-73100-7_61.

13. Vorontsov K., Iglovikov V., Strijov V., *et al.* Roundtable: Challenges in repeatable experiments and reproducible research in data science. Proceedings of Moscow Institute of Physics and Technology. 2021. Vol. 13, no. 2 (50). P. 100–108. (in Russian) DOI: 10.53815/20726759_2021_13_2_100.

14. Khritankov A., Pershin N., Ukhov N., Ukhov A. MLDev: Data Science Experiment Automation and Reproducibility Software. International Conference on Data Analytics and Management in Data Intensive Domains. Springer. 2021. P. 3–18. DOI: 10.1007/978-3-031-12285-9_1.

15. Alam K., Roy B. Challenges of provenance in scientific workflow management systems. 2022 IEEE/ACM Workshop on Workflows in Support of Large-Scale Science (WORKS). IEEE. 2022. P. 10–18. DOI: 10.1109/works56498.2022.00007.

16. Dhruv A., Dubey A. Managing software provenance to enhance reproducibility in computational research. Computing in Science & Engineering. 2023. Vol. 25, no. 3. P. 60–65. DOI: 10.1109/mcse.2023.3314288.

17. Zybin R., Shvetsova V., Badalyan D., *et al.* Cloud environment “Asperitas”. 2022. (in Russian). Certificate of state registration of a computer program RU 2022682679.

18. Grushin D., Samovarov O., Hashba E. SaaS platform for organizing a unified web environment for research, development and education “Fanlight”. 2018. (in Russian). Certificate of state registration of a computer program RU 2018615444.

19. Nasonov D., Butakov N., Bukhanovsky A., *et al.* Technology for organizing management and processing big data – DataMall. 2020. (in Russian). Certificate of state registration of a computer program RU 2020664222.

20. Kreuzberger D., Kühl N., Hirschl S. Machine learning operations (MLOPS): Overview, definition, and architecture. IEEE Access. 2023. Vol. 11. P. 31866–31879. DOI: 10.1109/access.2023.3262138.

21. Tyutlyaeva E.O., Odintsov I.O., Marmuzov G.V., *et al.* Development trends of modern supercomputers. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2019. Vol. 8, no. 3. P. 92–114. (in Russian) DOI: 10.14529/cmse190305.

22. Wilkinson M.D., Dumontier M., Aalbersberg I.J., *et al.* The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016. Vol. 3, no. 1. P. 1–9. DOI: 10.1038/sdata.2016.18.

23. Kostenetskiy P., Shamsutdinov A., Chulkevich R., *et al.* HPC TaskMaster - Task Efficiency Monitoring System for the Supercomputer Center. Parallel Computational Technologies / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer International Publishing, Jan. 2022. P. 17–29. DOI: 10.1007/978-3-031-11623-0_2.

24. Kostenetskiy P., Kozyrev V., Chulkevich R., Raimova A. Enhancement of the Data Analysis Subsystem in the Task-Efficiency Monitoring System HPC TaskMaster for the cHARISMa Supercomputer Complex at HSE University. Parallel Computational Technologies / ed. by L. Sokolinsky, M. Zymbler, V. Voevodin, J. Dongarra. Cham: Springer Nature Switzerland, 2024. P. 49–64. DOI: 10.1007/978-3-031-73372-7_4.

25. Lyu C., Zhang W., Huang H., *et al.* RTMDet: An Empirical Study of Designing Real-Time Object Detectors. CoRR. 2022. Vol. abs/2212.07784. DOI: 10.48550/ARXIV.2212.07784. arXiv: 2212.07784.

26. Slastnikov S., Chertova E. Machine vision model synthesis module for object and action detection. 2024. URL: https://cs.hse.ru/aicenter/rid_detection (in Russian).

27. Slastnikov S., Chertova E. A program for synthesis of machine vision models to detect objects and activities. 2023. (in Russian). Certificate of state registration of a computer program RU 2023660157.

28. Khritankov A.S. A method for performance analysis of distributed applications based on reference models. Parallel Computational Technologies (PCT’2011). 2011. P. 343–354. (in Russian).

29. Kostenetskiy P., Chulkevich R., Kozyrev V. HPC Resources of the Higher School of Economics. Journal of Physics: Conference Series. 2021. Jan. Vol. 1740, no. 1. P. 012050. DOI: 10.1088/1742-6596/1740/1/012050.