СПОРТИВНОЕ ПРОГРАММИРОВАНИЕ
Техническая заметка
https://doi.org/10.62105/2949-6349-2026-3-2-e202608
УДК 004.75

Высокотехнологический вычислительный кластер для проведения олимпиад по искусственному интеллекту

А. Ш. Халилов [1]✉, **Д. А. Подлесных** [1], **Д. А. Купцов** [1], **М. А. Лойко** [1]

[1] Федеральное государственное автономное образовательное учреждение высшего образования «Московский физико-технический институт (национальный исследовательский университет)», г. Долгопрудный, Российская Федерация

✉ khalilov.ash@phystech.edu

Аннотация

Актуальность. Рост популярности соревнований по искусственному интеллекту и машинному обучению, а также увеличение числа участников требуют создания специализированной вычислительной инфраструктуры, обеспечивающей равные условия и изоляцию при работе с графическими ускорителями.

Цель исследования. Описать архитектуру и практическую реализацию вычислительного кластера, подготовленного для проведения олимпиад по искусственному интеллекту и машинному обучению.

Методы исследования. Анализ требований очного соревнования к изоляции участников, одинаковой программной среде, равномерному распределению графических ускорителей, сохранению пользовательских данных и контролю сетевого доступа.

Результаты. Разработано техническое решение на базе RKE2/Kubernetes, NVIDIA A100 с разделением Multi-Instance GPU, персональных JupyterLab-рабочих мест, NFS-хранилища, защищенного веб-доступа и мониторинга Prometheus/Grafana. Проектная конфигурация рассчитана на 66 изолированных рабочих мест на 22 рабочих узлах и позволяет назначать каждому участнику отдельный GPU-экземпляр профиля 2g.20gb.

Выводы. Предложенная инфраструктура обеспечивает воспроизводимое развертывание и управляемую эксплуатацию среды для массового соревнования по машинному обучению. Новизна кейса состоит в адаптации облачно-нативных средств управления GPU-инфраструктурой к задачам очного финала школьной олимпиады по искусственному интеллекту.

Ключевые слова: искусственный интеллект, машинное обучение, вычислительный кластер, Kubernetes, RKE2, NVIDIA A100, MIG, JupyterLab, спортивное программирование, олимпиада, информационные технологии в спорте

Russian Journal of Information Technology in Sports, 2026, 3 (2), e202608
Российский журнал информационных технологий в спорте. 2026. Т. 3, № 2. e202608

SPORTS PROGRAMMING
Technical note
https://doi.org/10.62105/2949-6349-2026-3-2-e202608
UDC 004.75

High-technology computing cluster for artificial intelligence Olympiads

A. Sh. Khalilov [1]✉, **D. A. Podlesnykh** [1], **D. A. Kuptsov** [1], **M. A. Loiko** [1]

[1] Moscow Institute of Physics and Technology (National Research University), Dolgoprudny, Russian Federation

✉ khalilov.ash@phystech.edu

Abstract

Relevance. The growing popularity of artificial intelligence and machine learning competitions, as well as the increasing number of participants, require specialized computing infrastructure that ensures equal conditions and isolation when working with GPU accelerators.

Objective. To describe the architecture and practical implementation of a computing cluster prepared for artificial intelligence and machine learning Olympiads.

Research methods. Analysis of the requirements of an onsite competition: participant isolation, identical software environments, equal allocation of GPU accelerators, persistent user data, and controlled network access.

Results. The technical solution uses RKE2/Kubernetes, NVIDIA A100 GPUs partitioned with Multi-Instance GPU technology, personal JupyterLab workstations, NFS storage, secure web access, and Prometheus/Grafana monitoring. The target configuration supports 66 isolated workstations on 22 worker nodes and assigns a separate 2g.20gb GPU instance to each participant.

Conclusion. The proposed infrastructure provides reproducible deployment and manageable operation of an environment for mass machine learning competitions. The novelty of the case is the adaptation of cloud-native GPU infrastructure management to the tasks of a mass onsite school artificial intelligence Olympiad.

Keywords: artificial intelligence, machine learning, computing cluster, Kubernetes, RKE2, NVIDIA A100, MIG, JupyterLab, competitive programming, Olympiad, information technology in sports

Введение

В России несколько лет проводятся студенческие и школьные олимпиады, хакатоны и интенсивы по искусственному интеллекту. Всероссийская олимпиада по искусственному интеллекту входит в систему олимпиад школьников и ориентирована на участников, владеющих математикой, информатикой и программированием в задачах машинного обучения [1]. Ее не следует смешивать с профилем «Искусственный интеллект» Всероссийской олимпиады школьников (ВсОШ) по информатике: с 2025/2026 учебного года предмет «Информатика» во ВсОШ проводится по четырем профилям — программирование, информационная безопасность, робототехника и искусственный интеллект [2]. В настоящей статье рассматривается инфраструктура для олимпиады по искусственному интеллекту как самостоятельного соревнования, имеющего собственный регламент и очный финал.

Пояснения используемых технических терминов приведены в таблице 1 (глоссарий).

Для олимпиад по классическому программированию обычно достаточно системы проверки решений, компиляторов и ограничений по времени выполнения. В соревнованиях по искусственному интеллекту эта модель оказывается неполной. Участникам требуются графические ускорители, объемные наборы данных, интерактивные среды разработки, сохранение промежуточных файлов и возможность повторять вычислительные эксперименты. Поэтому инфраструктура становится частью методического дизайна: если программная среда, доступ к данным или производительность ускорителей различаются между участниками, то сравнение результатов становится менее строгим.

Существуют несколько близких инфраструктурных моделей. Классические HPC (High-Performance Computing, см. таблицу 1) кластеры хорошо подходят для длительных научных расчетов и пакетной постановки заданий, однако их модель очередей неудобна для очного тура, где десятки участников одновременно работают в интерактивных средах [3, 4]. Публичные облачные сервисы дают гибкость, но усложняют контроль стоимости, внешних сетевых соединений и воспроизводимости окружения [5]. Локальные рабочие станции проще для пользователя, но требуют ручной подготовки и хуже масштабируются при необходимости централизованно обновлять окружение, контролировать доступ и собирать метрики. В рассматриваемом проекте выбран гибридный подход: участники работают в привычных веб-средах JupyterLab, а организаторы управляют ресурсами через Kubernetes, GPU-квоты и единый контур мониторинга [6, 7].

Инфраструктурная задача относится и к области интеллектуального спорта. С момента создания Федерации спортивного программирования различные соревнования по программированию, робототехнике, информационной безопасности, программированию беспилотных систем и хакатоны стали рассматриваться как близкие формы интеллектуального спорта. Для таких мероприятий техническая платформа влияет не только на удобство, но и на честность соревнования: она должна обеспечить одинаковые стартовые условия, устойчивость во время тура и контролируемый доступ к внешним ресурсам.

Цель статьи — представить описание кейса проектирования и подготовки вычислительного кластера для олимпиад по искусственному интеллекту. В статье фиксируются архитектура, проектные параметры, фактически примененные контуры доступа и перечень метрик, которые следует использовать при приемке и сопровождении соревнования.

Глоссарий основных терминов

В статье используются англоязычные инженерные термины. Для удобства читателей ниже приведены краткие пояснения.

Таблица 1: Глоссарий основных терминов
Table 1. Glossary of basic terms

Термин	Пояснение
Kubernetes	Система управления контейнерами на группе серверов. Она запускает сервисы, следит за их состоянием и распределяет нагрузку по узлам.
RKE2	Дистрибутив Kubernetes, рассчитанный на эксплуатацию в производственной среде. В статье он выступает как основа кластера.
Ansible	Инструмент автоматизации ИТ-процессов с открытым исходным кодом, используемый для управления конфигурациями, развертывания программного обеспечения и оркестрации сложных задач на множестве серверов одновременно.
containerd	Программа, которая непосредственно запускает контейнеры на сервере. Kubernetes управляет ею через стандартный интерфейс.
Контейнер	Изолированная программная среда с приложением и зависимостями. Контейнер помогает запускать одинаковое окружение на разных серверах.
Pod	Минимальная единица запуска в Kubernetes. В данном проекте рабочее место участника оформлено как pod с JupyterLab и SSH-доступом.
Control plane	Управляющая часть кластера, которая хранит состояние системы и принимает решения о запуске сервисов.
Worker node	Рабочий сервер, на котором запускаются пользовательские среды и выполняются вычисления.
GPU	Графический ускоритель. В задачах искусственного интеллекта он ускоряет обучение и запуск моделей.
MIG	Multi-Instance GPU, режим разделения NVIDIA A100 на несколько независимых GPU-экземпляров. Для участника такой экземпляр выглядит как отдельная видеокарта с фиксированной квотой памяти и вычислений.
HPC	High-Performance Computing (высокопроизводительные вычисления) — кластеры, предназначенные для ресурсоёмких научных и инженерных расчётов, обычно использующие пакетную обработку заданий и параллельные вычисления.
JupyterLab	Веб-среда для программирования, работы с notebook-файлами, анализа данных и запуска Python-кода.
Notebook	Интерактивный документ, где код, результаты вычислений, графики и пояснения находятся в одном файле.

Продолжение на следующей странице

Таблица 1 – продолжение

Термин	**Пояснение**
NFS	Сетевая файловая система. Она позволяет нескольким серверам обращаться к файлам, расположенным на отдельном сервере хранения.
Persistent Volume	Постоянный том данных в Kubernetes. Он сохраняет файлы при перезапуске контейнера.
PVC	Persistent Volume Claim, заявка контейнера на постоянный том. В проекте через PVC создаются личные каталоги участников.
StorageClass	Правило Kubernetes, описывающее, как создавать тома хранения и какие свойства они имеют.
StatefulSet	Тип приложения Kubernetes для сервисов, которым нужны стабильные имена и постоянные тома данных.
Ingress	Правило входящего веб-доступа, которое связывает доменное имя участника с сервисом внутри кластера.
TLS	Механизм шифрования соединения, используемый в HTTPS.
SSH	Защищенный протокол удаленного доступа к командной строке; в проекте он нужен для работы через редакторы с удаленным подключением.
Sidecar	Дополнительный контейнер внутри того же pod-а. Он не является основной программой, но добавляет к ней отдельную функцию, например SSH-доступ.
Requests/limits	Минимальные запрошенные и максимальные допустимые ресурсы контейнера: процессор, память и GPU.
Device plugin	Компонент Kubernetes, который сообщает кластеру о доступных аппаратных устройствах, например GPU или MIG-экземплярах.
DCGM exporter	Сервис NVIDIA, который передает метрики GPU в систему мониторинга Prometheus.
node-exporter и kube-state-metrics	Сервисы, которые собирают системные метрики серверов и состояние объектов Kubernetes.
Pod affinity и topology constraints	Правила размещения pod-ов. Они помогают запускать рабочие места в нужном сегменте инфраструктуры и избегать нежелательной концентрации нагрузки.
DNS	Система сопоставления доменных имен и IP-адресов. В проекте она также используется как один из уровней ограничения доступа к внешним ресурсам.
CDN	Распределенная сеть доставки контента. Из-за CDN один сервис может использовать много IP-адресов, что затрудняет простую блокировку по адресам.
Deep packet inspection	Глубокая проверка сетевого трафика. Такой подход точнее простых списков блокировок, но требует больше ресурсов и сложнее в эксплуатации.

Продолжение на следующей странице

Таблица 1 – продолжение

Термин	Пояснение
VPN	Виртуальная частная сеть, позволяющая объединять разные сетевые сегменты поверх существующего интернета или внутренней сети.
PoE	Power over Ethernet, питание сетевого оборудования по тому же кабелю, по которому передаются данные.
STP	Spanning Tree Protocol, механизм защиты локальной сети от петель.
Clonezilla	Инструмент для создания и восстановления образов дисков; использовался для одинаковой подготовки компьютеров участников.
OBS	Программа записи экрана; использовалась для фиксации работы на компьютерах участников.
Prometheus и Grafana	Prometheus собирает метрики, а Grafana показывает их в виде панелей мониторинга.

Постановка задачи и требования к соревновательной инфраструктуре

Для организации очного финала олимпиады требуется предоставить участникам одновременный и сопоставимый доступ к GPU-ресурсам. Классические системы пакетной постановки задач на кластер плохо подходят для этой модели: участник должен не только отправить задание в очередь, но и интерактивно работать с кодом, данными и результатами экспериментов. Поэтому нужны видеокарты, поддерживающие надежное разделение ресурсов, и платформа, способная автоматически выдавать эти ресурсы отдельным пользовательским средам. В проекте таким аппаратным основанием стал графический ускоритель NVIDIA A100.

Кластер проектировался как специализированная среда для участников, выполняющих задания по искусственному интеллекту в интерактивных рабочих местах. Основные требования включали изоляцию пользовательских окружений, выделение сопоставимых GPU-ресурсов, сохранение персональных файлов между перезапусками, единый доступ к общим наборам данных, защищенный веб-доступ, возможность SSH-подключения для удаленных редакторов, централизованный мониторинг узлов, контейнеров, GPU и хранилища.

В качестве базовой платформы выбран Kubernetes, поскольку он предоставляет декларативную модель управления контейнеризованными приложениями, механизмы планирования pod-ов, ограничения ресурсов и абстракции хранилища [6]. Для развертывания используется RKE2 как дистрибутив Kubernetes, ориентированный на промышленную эксплуатацию, с containerd и типовой моделью управляющего и рабочего узла [8]. Управление жизненным циклом серверов вынесено в Ansible, что позволяет представить установку в виде последовательности повторяемых playbook-ов и уменьшить зависимость от ручных действий администратора [9].

GPU-часть инфраструктуры строится вокруг NVIDIA A100 и механизма Multi-Instance GPU. MIG позволяет разделить один физический ускоритель на несколько аппа-

ратно изолированных экземпляров [10]. В Kubernetes обнаружение и выдача GPU-ресурсов контейнерам выполняются через NVIDIA GPU Operator и связанные компоненты: device plugin, container toolkit и DCGM exporter [11]. Такой выбор позволяет перейти от ручного назначения видеокарт к ресурсной модели Kubernetes, где контейнер запрашивает конкретный тип MIG-ресурса.

Ограничением такого подхода остается масштаб: обеспечить каждого участника GPU-ресурсом целесообразно прежде всего в очном финале на несколько десятков человек, который проходит после дистанционного отбора. Масштабирование на сотни участников потребовало бы либо существенно большего числа ускорителей, либо иной соревновательной модели, например пакетной проверки без постоянного интерактивного доступа к GPU.

Техническое решение и его новизна

Ключевая особенность решения состоит в объединении трех контуров, которые в учебных мероприятиях часто существуют раздельно: GPU-кластера, пользовательских интерактивных сред и контролируемой сети рабочих мест. На уровне вычислений NVIDIA A100 делится на MIG-экземпляры. На уровне Kubernetes каждый такой экземпляр становится отдельным ресурсом, который можно запросить для конкретного pod-а. На уровне пользователя ресурс закрепляется за персональной JupyterLab-станцией. На уровне аудитории ноутбуки участников подключаются к подготовленной сети с резервированным питанием, ограничением доступа и записью экрана.

Новизна кейса состоит не в применении Kubernetes или MIG по отдельности, а в их адаптации к требованиям очной олимпиады. В типовом вычислительном центре GPU-кластер обслуживает длительные исследовательские проекты, а в обычном учебном классе участники работают на локально подготовленных компьютерах. Рассматриваемое решение соединяет эти модели: участник получает привычную интерактивную среду, а организатор сохраняет централизованное управление ресурсами, доступом, данными и мониторингом.

Архитектура сети рабочих мест

Рабочие места участников размещались в аудиториях МФТИ, подключенных через гигабитный Ethernet и имеющих постоянное подключение к интернету с фиксированным внешним IP-адресом на скорости 1 Гбит/с. В каждой аудитории использовался коммутатор с поддержкой PoE и STP, подключенный к источнику бесперебойного питания. Он питал коммутаторы уровня доступа на столах участников по Ethernet-кабелю. Сами рабочие места были организованы на ноутбуках, поэтому сеть и пользовательские устройства сохраняли работоспособность при кратковременных проблемах с электропитанием.

Фиксированный внешний IP-адрес упростил ограничение доступа к кластеру: инфраструктура могла принимать подключения только из подготовленных аудиторий. Для исключения помощи извне потребовалось ограничить доступ к социальным сетям, мессенджерам и другим нерелевантным ресурсам. Простая блокировка по IP-адресам недостаточна, поскольку многие сервисы используют CDN и часто меняют адреса. Поэтому была применена гибридная схема: часть ресурсов блокировалась по IP-адресам и подсетям, собираемым отдельным скриптом, а часть — на уровне DNS-разрешения имен.

Таблица 2: Состав технического решения
Table 2. Composition of the technical solution

Проблема соревнования	**Принятое решение**	**Практический эффект**
Одинаковая GPU-квота	Разделение A100 на MIG-профили 2g.20gb	Участник получает отдельный GPU-экземпляр, а не конкурирует за общий ускоритель
Массовое создание рабочих сред	StatefulSet с JupyterLab и декларативные Kubernetes-манифесты	Рабочие места создаются и перезапускаются одинаковым способом
Сохранность файлов	Персональные persistent volumes на NFS	Перезапуск контейнера не уничтожает файлы участника
Единый датасет	Общий NFS-том в режиме только для чтения	Все участники читают один и тот же набор данных
Контроль внешнего доступа	Фиксированный внешний IP, DNS- и IP-блокировки, сетевые политики	Снижается риск помощи извне и отвлечения на нерелевантные ресурсы
Повторяемая подготовка аудиторий	Clonezilla для образов компьютеров, OBS для записи экрана, PoE/STP-сеть с ИБП	Условия работы в аудиториях становятся более единообразными и проверяемыми
Наблюдаемость	Prometheus/Grafana и GPU-метрики DCGM	Организаторы видят состояние рабочих мест, узлов и ускорителей во время тура

В пробном туре такая схема проверялась на сценарии, где примерно каждый третий контрольный запрос вел к заблокированному ресурсу; запросы успешно блокировались. Во время основных туров в первые 20 минут наблюдался небольшой рост числа блокировок, после чего обращения к запрещенным ресурсам почти прекращались. Это косвенно показывает, что участники переходили к решению задач, а не пытались регулярно обращаться к заблокированным сервисам. Более точная оценка эффективности сетевых ограничений должна основываться на журналах DNS и пограничного оборудования.

На компьютеры участников устанавливался идентичный набор программного обеспечения при помощи Clonezilla. На всех компьютерах велась запись экрана средствами OBS. Чтобы не занимать пропускную способность сети во время тура, записи переносились на сетевое хранилище ночью после завершения соревновательного дня.

Архитектура кластера

Проектная конфигурация кластера включает один узел control plane, отдельный NFS-сервер и до 22 worker-узлов с GPU. Control plane размещает управляющие компоненты RKE2, ingress-контур, выпуск TLS-сертификатов и мониторинг. Worker-узлы маркируются

как узлы рабочих мест и принимают pod-ы участников. NFS-сервер предоставляет персональные тома и общий каталог с наборами данных. Сетевая модель использует стандартные CIDR-диапазоны Kubernetes для pod- и service-сетей, а доступ пользователей организован через домены вида competitorN.vserosai.ru.

Выделение большого числа GPU-ресурсов в одном сегменте облачного провайдера может быть затруднено. Для кластера, распределенного между сегментами, возможны два базовых варианта: использовать штатную статическую или динамическую маршрутизацию между сегментами либо назначать отдельным узлам роль межсегментных L3-маршрутизаторов и поднимать связность через VPN. Второй вариант менее устойчив в условиях, где VPN-трафик может блокироваться организационными политиками или внешними сетевыми ограничениями.

С точки зрения равенства условий желательно сокращать кросс-сегментный пользовательский трафик. Для этого рабочее место участника и точку входа его трафика можно размещать в одном сегменте облачного провайдера с помощью правил размещения pod affinity, topology constraints и привязки DNS-записи к балансировщику соответствующего сегмента. По той же причине конкурсные группы, например участников одного класса, целесообразно размещать в одном центре обработки данных: отказ отдельного сегмента тогда затрагивает ограниченную и заранее понятную группу участников.

В таблице 3 фиксируются параметры, заложенные в конфигурационных файлах развертывания. Они являются проектными параметрами системы и не подменяют эксплуатационные измерения.

Таблица 3: Проектные параметры вычислительного кластера
Table 3. Design parameters of the computing cluster

Компонент	**Параметр**
Kubernetes-дистрибутив	RKE2, целевая версия Kubernetes 1.34, containerd
Control plane	1 узел, RKE2 server, kubeconfig для администрирования
Worker-узлы	до 22 узлов, каждый помечается как workstation-node=true
GPU	NVIDIA A100; MIG включен; профиль 2g.20gb; 3 MIG-экземпляра на GPU
Рабочие места	до 66 JupyterLab-станций: 3 станции на worker-узел
Ресурсы станции	запрос: 2 CPU, 16 Gi RAM, 1 nvidia.com/mig-2g.20gb; лимит: 4 CPU, 32 Gi RAM
Персональное хранилище	PVC на NFS, 100 Gi на участника, режим ReadWriteOnce
Общие данные	NFS-том shared-dataset только для чтения, 500 Gi, режим ReadOnlyMany
Доступ	JupyterLab по HTTPS; sidecar-контейнер с SSH для удаленной разработки
Ingress и TLS	Nginx Ingress Controller, cert-manager, Let's Encrypt
Мониторинг	kube-prometheus-stack, хранение метрик Prometheus 30 дней, панели Grafana для Kubernetes, node-exporter и NVIDIA GPU

Использование StatefulSet для рабочих мест обусловлено необходимостью стабильной идентичности pod-ов и связанного с ними persistent volume claim [12]. Kubernetes Persistent Volumes и StorageClass отделяют жизненный цикл пользовательских данных от жизненного цикла контейнеров [13]. Для общего набора данных применяется том только для чтения, что уменьшает риск случайного изменения исходных данных участниками и упрощает воспроизводимость заданий.

Организация пользовательских рабочих мест

Каждое рабочее место представляет собой pod с основным контейнером JupyterLab и sidecar-контейнером SSH-сервера. JupyterLab выбран как интерактивная среда для задач анализа данных и машинного обучения [7]. Базовый образ jupyter/tensorflow-notebook предоставляет готовое Python-окружение с библиотеками для вычислительных экспериментов [14]. Для участника задаются пользователь competitor, персональный каталог /home/competitor/workspace и каталог /home/competitor/datasets, доступный только для чтения.

Выдача ресурсов выполняется через механизм requests/limits, то есть через минимальные запрошенные и максимальные допустимые ресурсы контейнера. Для GPU используется ресурс nvidia.com/mig-2g.20gb: 1; это означает, что планировщик размещает pod только на узле, где доступен соответствующий MIG-экземпляр. На уровне узлов предварительно включается MIG-режим, очищаются существующие GPU-экземпляры и создаются три экземпляра профиля 2g.20gb. В совокупности такая схема задает одинаковую аппаратную квоту для каждого участника и предотвращает конкуренцию нескольких рабочих мест за один и тот же GPU-экземпляр.

Доступ к JupyterLab защищается индивидуальными токенами, сгенерированными для каждого рабочего места. SSH-доступ использует отдельные пароли и уникальные внешние порты. Учетные данные хранятся отдельно от исходного кода проекта и ротируются между турами. Передача учетных данных участникам выполняется организационной командой через заранее определенный канал; такой порядок уменьшает риск случайной публикации токенов вместе с конфигурациями.

Хранение данных, безопасность и сетевые ограничения

Подсистема хранения разделяет персональные и общие данные. Персональные тома создаются динамически через NFS Subdir External Provisioner; политика Retain и параметр архивирования при удалении уменьшают риск потери работ при ошибочных операциях администратора. Общий набор данных монтируется в режиме ReadOnlyMany, то есть доступен многим рабочим местам только для чтения. Это соответствует соревновательному требованию неизменности исходного датасета для всех участников. NFS версии 4.1 используется как простая и хорошо поддерживаемая модель сетевой файловой системы для Linux-кластеров [15].

Сетевой периметр включает ingress-доступ по HTTPS, TLS-сертификаты cert-manager и базовые NetworkPolicy для namespace рабочих мест [16, 17]. Политики разрешают DNS, необходимые обращения к сервисам Kubernetes, доступ к NFS и ограниченный исходящий TCP-трафик. Список блокируемых доменов и подсетей применяется как дополнительный слой контроля. В отличие от deep packet inspection, такая схема проще в эксплуатации

и требует меньше ресурсов, хотя хуже распознает сложные случаи обхода через CDN и зеркала.

С точки зрения соревновательной честности ключевыми являются не только криптографические средства, но и операционные правила: запрет общего доступа к персональным томам, разделение учетных данных, журналирование административных действий, ограничение коммуникации между рабочими местами и контроль пакетов, устанавливаемых во время тура. Эти правила должны быть синхронизированы с регламентом олимпиады и правилами обработки персональных данных участников.

Мониторинг и эксплуатация

Для мониторинга используется kube-prometheus-stack, то есть готовый набор компонентов Prometheus и Grafana для Kubernetes: Prometheus собирает метрики кластера, node-exporter отвечает за системные показатели узлов, kube-state-metrics — за состояние объектов Kubernetes, Grafana предоставляет панель наблюдения [18, 19]. GPU-метрики собираются через DCGM exporter, входящий в экосистему NVIDIA для мониторинга ускорителей [20]. В конфигурации предусмотрены готовые панели Grafana для Kubernetes-кластера, node-exporter и NVIDIA GPU.

Таблица 4: План эксплуатационной оценки
Table 4. Operational evaluation plan

Показатель	**Метод фиксации**	**Как используется в статье**
Доступность рабочих мест	число Ready-pod-ов в namespace workstations	Подтверждает, что проектная емкость выведена в рабочее состояние
Время холодного старта станции	интервал от создания или перезапуска pod-а до readiness probe	Показывает, сколько времени требуется на восстановление рабочего места
Корректность GPU-выдачи	nvidia-smi внутри рабочих мест и Kubernetes resource allocation	Подтверждает, что каждый участник получает отдельный MIG-экземпляр
Утилизация GPU во время тура	Prometheus/DCGM exporter	Позволяет оценить достаточность выбранного профиля 2g.20gb
Доступность общего датасета	контрольное чтение из /home/competitor/datasets	Проверяет единый доступ участников к исходным данным
Стабильность хранилища	события Kubernetes, NFS-логи, ошибки PVC	Позволяет выявить узкие места персонального и общего хранилища
Сетевые ограничения	журналы DNS, пограничного оборудования и контрольные запросы	Подтверждает работу сетевого контура соревнования
Готовность аудиторий	проверка Clonezilla-образов, OBS-записи, PoE/STP-сети и ИБП	Подтверждает одинаковые условия работы за физическими компьютерами

Эксплуатационные сценарии вынесены в набор скриптов: добавление и удаление worker-узлов, создание и перезапуск рабочих мест, генерация токенов, загрузка данных в общий каталог, загрузка файлов в персональный том, резервное копирование и очистка томов. Такой слой управления важен для соревнования, где часть операций должна выполняться быстро и воспроизводимо, но не обязательно требовать от оператора знания всех деталей Kubernetes API.

Контроль готовности системы перед стартом соревнования должен включать доступность всех worker-узлов, корректную рекламу MIG-ресурсов, запуск всех JupyterLab-pod-ов, доступность доменов участников, чтение общего датасета, запись в персональные тома, видимость GPU из Python-окружения, отсутствие незапланированного доступа между рабочими местами, актуальность резервной копии, корректность DNS- и IP-блокировок, а также готовность записи экрана на компьютерах участников.

Обсуждение

Предложенная архитектура занимает промежуточное положение между учебной лабораторией и высокопроизводительным вычислительным центром. От классических HPC-систем она наследует задачу плотного распределения ускорителей, централизованного мониторинга и управления общими данными; от облачно-нативных платформ — декларативное развертывание, контейнерную изоляцию и автоматизированное восстановление сервисов. Подход согласуется с общей тенденцией сближения HPC и нагрузок искусственного интеллекта, где контейнеризация и оркестрация становятся инструментами воспроизводимости и масштабирования [3, 4]. Эволюция Kubernetes из промышленных систем управления контейнерами, а также практика легковесной контейнеризации показывают, почему такой стек удобен для кратковременных массовых вычислительных мероприятий [21, 22, 5].

Ключевое инженерное решение состоит в использовании MIG как единицы соревновательной квоты. Это снижает различия в доступных GPU-ресурсах между участниками по сравнению с режимом разделяемого полного GPU и делает планирование рабочих мест более предсказуемым. Вместе с тем MIG не решает все вопросы изоляции: CPU, RAM, сеть и файловая система по-прежнему требуют отдельных лимитов, политик и эксплуатационного контроля.

Второй важный выбор — JupyterLab как основной пользовательский интерфейс. Он снижает порог входа для школьников, знакомых с notebook-сценариями, и подходит для задач анализа данных; сама notebook-модель широко используется как формат воспроизводимых вычислительных рабочих процессов [23]. Ограничение состоит в том, что интерактивная среда требует тщательного контроля зависимостей, доступа к пакетным репозиториям и сохранения состояния. Для задач с повторным обращением к одним и тем же данным полезен опыт систем распределенной обработки с рабочими наборами, поскольку он подчеркивает значение локальности данных и повторного использования промежуточных результатов [24].

Отдельной особенностью кейса является связь вычислительного кластера с физической инфраструктурой аудитории. Даже хорошо настроенный Kubernetes-кластер не решает задачи очного соревнования полностью, если локальная сеть нестабильна, компьютеры участников различаются по программному составу или отсутствует проверяемый контроль сетевого доступа. Поэтому техническое решение следует рассматривать как единую систему: облачно-нативный кластер, сеть аудиторий, одинаковые образы рабочих

компьютеров и организационные процедуры.

Выводы

Описан кейс построения вычислительного кластера для олимпиад по искусственному интеллекту. Инфраструктура основана на RKE2/Kubernetes, NVIDIA A100 с MIG-разделением, JupyterLab-рабочих местах, NFS-хранилище, ingress/TLS-контуре и мониторинге Prometheus/Grafana. Проектная конфигурация рассчитана на 66 изолированных рабочих мест и задает одинаковую GPU-квоту для каждого участника.

Практическая ценность подхода состоит в воспроизводимом развертывании и управляемой эксплуатации среды для массового соревнования по машинному обучению. Особенность решения заключается в том, что участник получает привычную интерактивную среду, а организатор сохраняет контроль над ресурсами, данными, сетевым периметром, состоянием кластера и физическими рабочими местами. Такой подход может быть использован при подготовке олимпиад, учебных интенсивов и хакатонов, где требуется сочетание честного распределения ускорителей и централизованной эксплуатации.

Список литературы / References

1. Всероссийская олимпиада по искусственному интеллекту. Официальный сайт. URL: https://ai.edu.gov.ru/

All-Russian Olympiad in Artificial Intelligence. Official website. (in Russ.)

2. Минпросвещения направлены разъяснения по вопросу об участии в олимпиаде по предметам, в которых предусмотрены профили // КонсультантПлюс. URL: https://www.consultant.ru/law/hotdocs/90793.html

Ministry of Education explanations on participation in Olympiad subjects with profiles. ConsultantPlus. (in Russ.)

3. Zacharov I., Arslanov R., Gunin M., Stefonishin D., Pavlov S. et al. “Zhores” — Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology. *Open Engineering*, 2019, 9(1), pp. 512–520. https://doi.org/10.1515/eng-2019-0059

4. Reed D.A., Dongarra J. Exascale computing and big data. *Communications of the ACM*, 2015, 58(7), pp. 56–68. https://doi.org/10.1145/2699414

5. Armbrust M., Fox A., Griffith R., Joseph A.D., Katz R., Konwinski A. et al. A view of cloud computing. *Communications of the ACM*, 2010, 53(4), pp. 50–58. https://doi.org/10.1145/1721654.1721672

6. Kubernetes Documentation. Production-Grade Container Orchestration. URL: https://kubernetes.io/docs/

7. JupyterLab Documentation. URL: https://jupyterlab.readthedocs.io/

8. RKE2 Documentation. URL: https://docs.rke2.io/

9. Ansible Documentation. URL: https://docs.ansible.com/

10. NVIDIA. Multi-Instance GPU User Guide. URL: https://docs.nvidia.com/datacenter/tesla/mig-user-guide/

11. NVIDIA. GPU Operator Documentation. URL: https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/

12. Kubernetes Documentation. StatefulSets. URL: https://kubernetes.io/docs/concepts/workloads/controll

13. Kubernetes Documentation. Persistent Volumes. URL: https://kubernetes.io/docs/concepts/storage/persistent-volumes/

14. Jupyter Docker Stacks Documentation. URL: https://jupyter-docker-stacks.readthedocs.io/

15. Haynes T., Noveck D. Network File System (NFS) Version 4 Minor Version 1 Protocol, RFC 8881, 2020. https://doi.org/10.17487/RFC8881

16. Kubernetes Documentation. Network Policies. URL: https://kubernetes.io/docs/concepts/services-networking/network-policies/

17. cert-manager Documentation. URL: https://cert-manager.io/docs/

18. Prometheus Documentation. URL: https://prometheus.io/docs/

19. Grafana Documentation. URL: https://grafana.com/docs/grafana/latest/

20. NVIDIA. DCGM Exporter. URL: https://github.com/NVIDIA/dcgm-exporter

21. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Borg, Omega, and Kubernetes. *Communications of the ACM*, 2016, 59(5), pp. 50–57. https://doi.org/10.1145/2890784

22. Merkel D. Docker: lightweight Linux containers for consistent development and deployment. *Linux Journal*, 2014, (239). URL: https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment

23. Kluyver T., Ragan-Kelley B., Perez F., Granger B., Bussonnier M., Frederic J. et al. Jupyter Notebooks — a publishing format for reproducible computational workflows. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, IOS Press, 2016, pp. 87–90. https://doi.org/10.3233/978-1-61499-649-1-87

24. Zaharia M., Chowdhury M., Franklin M.J., Shenker S., Stoica I. Spark: Cluster Computing with Working Sets. *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing*, 2010. URL: https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets

Информация об авторах / Information about the authors

Халилов Амир Шамилевич – Факультет прикладной математики и информатики Московского физико-технического института; руководитель проектной команды по подготовке вычислительной инфраструктуры / *Amir Sh. Khalilov* – Faculty of Applied Mathematics and Computer Science, Moscow Institute of Physics and Technology; head of the project team for computing infrastructure preparation; *khalilov.ash@phystech.edu*

Подлесных Дмитрий Алексеевич – Кафедра информатики и вычислительной математики Московского физико-технического института, член Центральной предметно-методической комиссии по информатике / *Dmitry A. Podlesnykh* – Department of Informatics and Computational Mathematics of Moscow Institute of Physics and Technology; member of the Central Subject-Methodological Commission in Informatics; *podlesnykh.da@phystech.edu*

Купцов Дмитрий Алексеевич – участник проектной команды по подготовке и эксплуатации соревновательной инфраструктуры МФТИ / *Dmitry A. Kuptsov* – member of the project team for preparing and operating the competition infrastructure at Moscow Institute of Physics and Technology; *kuptsov.da@phystech.edu*

Лойко Михаил Алексеевич – участник проектной команды по подготовке и эксплуатации соревновательной инфраструктуры МФТИ / *Mikhail A. Loiko* – member of the project team for preparing and operating the competition infrastructure at Moscow Institute of Physics and Technology; *loiko.ma@phystech.edu*

Вклад авторов / Contribution of the authors (CRediT)

Халилов А.Ш. (Khalilov A.Sh.) – концептуализация (Conceptualization), разработка методологии (Methodology), программное обеспечение (Software), формальный анализ (Formal analysis), исследование (Investigation), подготовка исходного текста (Writing – original draft).

Подлесных Д.А. (Podlesnykh D.A.) – методологическая редактура (Writing – review & editing), контроль (Validation).

Купцов Д.А. (Kuptsov D.A.) – ресурсы (Resources), контроль (Validation), исследование (Investigation).

Лойко М.А. (Loiko M.A.) – ресурсы (Resources), контроль (Validation), исследование (Investigation).

Все авторы участвовали в обсуждении архитектуры, проверке технических формулировок и согласовании итоговой версии статьи.

Благодарности / Acknowledgements

Авторы благодарят организационную и инженерную команды олимпиады по искусственному интеллекту за постановку практических требований к инфраструктуре и проверку эксплуатационных сценариев. Исходные конфигурации и эксплуатационные материалы хранятся в рабочем репозитории проекта и могут быть предоставлены редакции по запросу.

The authors thank the organizational and engineering teams of the artificial intelligence Olympiad for formulating practical infrastructure requirements and validating operational scenarios. The source configurations and operational materials are stored in the project working repository and may be provided to the editorial office upon request.

Конфликт интересов / Conflict of interests

Авторы декларируют отсутствие конфликта интересов. Любые результаты исследований доступны по прямому запросу.

The authors declare no conflict of interests. Any research results are available upon direct request.

Цитирование / Citation

Халилов А.Ш., Подлесных Д.А., Купцов Д.А., Лойко М.А. Высокотехнологический вычислительный кластер для проведения олимпиад по искусственному интеллекту // Российский журнал информационных технологий в спорте. 2026. Т. 3, № 2. e202608. https://doi.org/10.62105/2949-6349-2026-3-2-e202608 EDN WWEKIM

Khalilov A.Sh., Podlesnykh D.A., Kuptsov D.A., Loiko M.A. High-technology computing cluster for artificial intelligence Olympiads. *Russian Journal of Information Technology in Sports*, 2026, 3 (2), e202608. (In Russ.) https://doi.org/10.62105/2949-6349-2026-3-2-e202608 EDN WWEKIM

Получено/Received: *18.05.2026*
Одобрено/Accepted: *18.06.2026*
Опубликовано/Published: *21.06.2026*