**Технические требования**

по подготовке, представлению и рассмотрению предложений и осуществлению отбора поставщика на выполнение работ по разработке информационной системы оркестрации и управления GPU кластером Run AI

## 1. Общие сведения

**Заказчик:** ООО «Единый интегратор UZINFOCOM».

**Цель отбора:** Отбор квалифицированного поставщика для разработки информационной системы оркестрации и управления GPU кластером Run AI

## 2. Технические требования к системе:

Поставщик должен обеспечить разработку и поддержку следующих основных функциональных модулей и подсистем информационной системы:

- система должна обеспечивать возможность распределения задач по всем узлам кластера, включая как CPU, так и GPU, для максимального использования доступных ресурсов;
- решение должно быть адаптировано для поддержки кластеров любых размеров, от небольших до экзафлопсных систем, с возможностью динамического добавления новых узлов и оборудования;
- система должна поддерживать специализированные библиотеки и драйверы для работы с GPU, обеспечивая оптимизацию под графические процессоры;
- поддержка систем балансировки нагрузки, таких как Kubernetes, Docker Swarm или Slurm, для автоматизации распределения задач, планирования их выполнения и обеспечения приоритетного доступа к критически важным задачам;
- система должна предоставлять инструменты для мониторинга состояния кластера, позволяя отслеживать метрики производительности GPU, загрузку ресурсов, температурные показатели и узкие места;
- интеграция с технологиями контейнеризации, такими как Docker и Kubernetes, для изоляции приложений, упрощения управления их зависимостями и ускорения развертывания приложений;
- возможность мониторинга и оптимизации энергопотребления с целью снижения затрат на эксплуатацию и предотвращения перегрева оборудования;
- поддержка сетевых технологий с низкой латентностью, таких как InfiniBand, для минимизации задержек при передаче данных между узлами;
- совместимость с распределёнными файловыми системами (Lustre, Ceph, Bee GFS) и NVMe-накопителями для обеспечения быстрого доступа к данным;
- наличие API для интеграции с внешними системами управления, автоматизации процессов и создания пользовательских решений;
- система должна поддерживать функции безопасности, включая разграничение прав доступа к ресурсам, аутентификацию пользователей и шифрование данных;
- возможность выполнения задач аварийного восстановления, резервирования данных и перезапуска задач в случае сбоев системы;
- пользователь должен иметь возможность настраивать параметры оркестрации, включая приоритет выполнения задач, распределение ресурсов и планирование вычислений;

Система оркестрации и управления GPU кластером также должна включать в себя несколько ключевых компонентов и инструментов. Назначение технологии - эффективное распределение задач и управление ресурсами.

**Основные сферы применения внутри кластера сводятся к исполнению следующего функционала:**

Использование инструментов оркестрации контейнеров, таких как Kubernetes, позволяет автоматизировать развертывание, масштабирование и управление контейнерными приложениями. Kubernetes поддерживает как декларативную конфигурацию, так и автоматизацию, что помогает управлять рабочими нагрузками и услугами в кластере.

Управление ресурсами: Инструменты, такие как Run.AI, помогают упростить управление кластером GPU, обеспечивая эффективное распределение ресурсов и балансировку нагрузки. Это позволяет оптимизировать использование GPU и повысить производительность системы.

Мониторинг и аналитика: Инструменты мониторинга позволяют отслеживать состояние узлов, производительность и использование ресурсов, что помогает своевременно выявлять и устранять проблемы.

Масштабирование, добавление новых узлы в кластер и масштабирование ресурсов в зависимости от потребностей. Это особенно важно для задач, требующих высокой вычислительной мощности, таких как обучение моделей глубокого обучения и обработка больших объемов данных.

Обеспечение безопасности данных и ресурсов в кластере является важным. Инструменты оркестрации должны включать в себя функции безопасности, такие как управление доступом, шифрование данных и защита от атак.

**Система должна развертываться на базе Kubernetes с использованием:**

– выделенного Namespace для управляющей плоскости;

– кастомного Kubernetes Operator для управления жизненным циклом ресурсов;

– обеспечивать мультикластерное управление с централизованным мониторингом географически распределенных узлов.

**Безопасность**

– система должна использовать TLS 1.3 и VPN для шифрования данных между управляющей плоскостью и вычислительными узлами;

– система должна поддерживать развертывание локального центра сертификации (CA) для air-gapped сред.

Система должна предоставлять предварительно настроенные Docker-образы для:

– Triton Inference Server с примерами инференса ONNX-моделей;

– Kubeflow Pipelines с шаблонами для распределенного обучения PyTorch;

– система должна позволять автоматически генерировать манифесты Kubernetes для задач Apache Airflow и Argo Workflows.

**Система должна расширять Kubernetes Scheduler, реализуя:**

– Gang Scheduling для одновременного запуска взаимозависимых Pod-групп;

– Bin-Pack/Spread стратегии для GPU задач с настройкой через аннотации scheduler.alpha.kubernetes.io/node-affinity;

– динамическое перераспределение ресурсов на основе метрик Prometheus (GPUUtil > 90%).

**Управление очередями:**

– система должна создавать иерархические очереди в формате;

– система должна блокировать задачи, превышающие квоту hard limits, кроме случаев over-quota с приоритетом PriorityClass: critical.

**Ресурсы и балансировка**

– квоты и Fairshare:

– система должна рассчитывать fairshare;

– при превышении fairshare-доли система обязана:

– инициировать мягкое вытеснение (graceful preemption) задач с сохранением состояния для последующего перезапуска;

– система должна предоставлять механизм репликации смонтированного PVC/PV с возможностью монтирования данного PVC/PV к любой workload задаче в кластере.

– перераспределять ресурсы между пулами через механизм DynamicQuotaReallocator;

– система должна поддерживать автоматический reclaim ресурсов через механизм kube-scheduler-simulator с интервалом в 5 минут.

**Отказоустойчивость**

– система должна обеспечивать горизонтальное масштабирование control-plane с использованием etcd-кластера и автоматическим восстановлением при потере 50% нод;

– система должна настраивать Pod Disruption Budget для критичных сервисов (например, min Available: 90%).

**Рабочие пространства (Workspaces)**

Создание и управление:

– система должна развертывать workspace как StatefulSet с:

– предустановленными образами JupyterLab + VS Code Server;

– Persistent Volume размером от 100 ГБ (класс хранилища: ssd);

– система должна автоматически приостанавливать неактивные workspace через заданный период времени, либо предоставлять другой механизм оптимизации.

– Доступ:

Система должна предоставлять доступ к workspace через:

– Ingress-контроллер с OAuth2-прокси;

– SSH-туннель с ключами Ed25519.

Обучение моделей. Конфигурация задач:

– система должна генерировать CRD (Custom Resource Definition) для задач обучения и мониторинга:

– система должна интегрироваться с Grafana, предоставляя дашборды:

– GPU Memory Usage (по узлам/проектам);

– Fairshare Utilization Heatmap.

Безопасность и RBAC. Аутентификация:

– система должна поддерживать:

– OIDC-провайдеры;

– многофакторную аутентификацию через TOTP (RFC 6238).

– Авторизация:

– система должна реализовывать роли:

– система должна синхронизировать роли с Kubernetes RBAC через kube-rbac-proxy.

Мониторинг и логирование по метрикам:

– система должна экспортировать метрики в формате OpenMetrics:

– gpu_cluster_queue_pending_jobs (тип: Gauge);

– gpu_utilization_per_project (тип: Histogram);

– допускается использовать альтернативные форматы.

Триггеры:

– система должна выполнять скрипты Python/Bash при срабатывании условий.

– Установка и требования. Поддержка air-gapped:

– система должна предоставлять скрипт для генерации offline-пакетов с зависимостями (например, airgap-bundle.tar.gz).

Управление GPU-ресурсами. Поддержка разнородных GPU:

– система должна автоматически определять типы GPU ( B100, B200 и т.д.) и их характеристики (память, версия CUDA);

– система должна учитывать лицензионные ограничения (например, NVIDIA vGPU) при распределении ресурсов;

– система должна приоритизировать задачи в зависимости от совместимости с GPU (например, задачи с FP16 → B200).

Автоскейлинг на основе метрик:

Система должна масштабировать пулы узлов при достижении порогов:

– gpu_utilization > 85% → увеличение узлов;

– gpu_utilization < 30% → уменьшение узлов.

Система должна интегрироваться с Karpenter для оптимизации затрат в облаке.

**Интеграция с MLOps-инструментами**

Продакшн-деплоймент:

Система должна поддерживать деплоймент моделей через различные инструменты, например:

– KServe для сервинга с канареечным развертыванием;

– Seldon Core с A/B-тестированием;

– система должна генерировать Helm-чарты для воспроизводимости окружений.

**Управление версиями и артефактами**

Версионирование:

Система должна хранить:

– версии моделей (формат ONNX, TensorRT);

– артефакты обучения (чекпоинты, логи, датасеты).

– система должна автоматически связывать артефакты с Git-коммитами (интеграция с GitHub/GitLab).

Политики энергопотребления:

– система должна переводить неиспользуемые GPU в режим low-power через Data Center GPU Manager (DCGM);

– система должна генерировать отчеты по энергозатратам на проект (кВт·ч/эксперимент).

Аналитика и отчетность. Финансовая аналитика:

– система должна рассчитывать ROI на GPU;

– система должна формировать отчеты по загрузке ресурсов в разрезе департаментов (Excel/PDF).

**6. Требования к участникам отбора:**

- Участник отбора должен быть юридическим лицом, зарегистрированным в Республике Узбекистан.
- Наличие опыта реализации аналогичных проектов не менее 3 лет.
- Наличие постоянного офиса и квалифицированного технического персонала в городе Ташкенте.
- Поставщик должен подтвердить финансовую устойчивость и отсутствие судебных разбирательств или ограничений.
- Гарантии конфиденциальности и защиты данных.
- Предоставить гарантии качества выполненных работ и соблюдения сроков реализации проекта.

**7. Документы, представляемые участниками:**

- Свидетельство о регистрации юридического лица.
- Копии лицензий на деятельность в области информационных технологий.
- Гарантийное письмо о своевременном и качественном выполнении работ.
- Рекомендательные письма от предыдущих заказчиков (если есть).
- Детализированное коммерческое (ценовое) предложение.

**ООО «Единый интегратор UZINFOCOM» оставляет за собой право отказаться от проведения отбора и отклонить все поступившие заявки в любое время до даты окончания приема заявок.**