
JINR
MLiT

Объединённый институт ядерных исследований

Распределённая гетерогенная вычислительная среда для обработки данных NICA

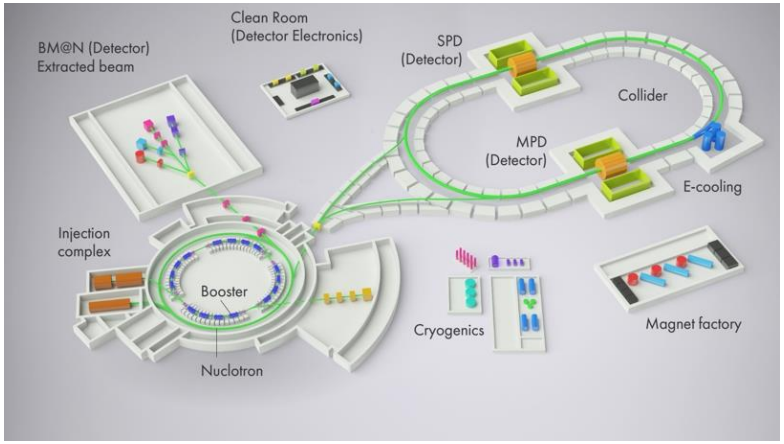
Игорь Пелеванюк

Научная специальность: 2.3.5 — Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей по техническим наукам.

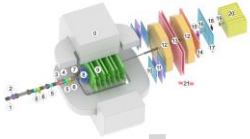
соискание ученой степени кандидата технических наук

Научный руководитель:
доктор технических наук
Кореньков Владимир Васильевич

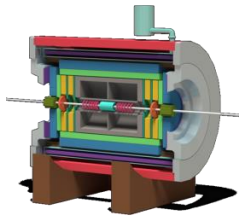
Актуальность темы исследования

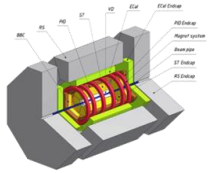


NICA (Nuclotron-based Ion Collider fAcility) — сверхпроводящий коллайдер тяжёлых ионов и протонов, строящийся с 2013 года на базе ОИЯИ. В его задачи входят исследования связанные с поиска смешанной фазы ядерной материи и исследования поляризационных эффектов в области энергий до 11 ГэВ/нуклон.



BM@N





SPD NICA

На ускорительном комплексе NICA планируется работа трёх больших детекторов:

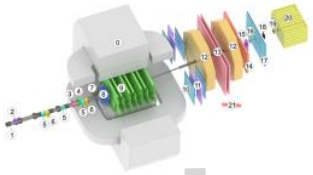
Детектор BM@N (Baryonic Matter at Nuclotron). Целью эксперимента является изучение взаимодействия релятивистских пучков тяжёлых ионов с фиксированными мишенями.

Детектор MPD (Multi-Purpose Detector) - для проведения экспериментов в области релятивистской ядерной физики при столкновениях пучков ядер тяжёлых элементов, ядер тяжёлых элементов с протонами и протон-протонных столкновениях.

Детектор SPD (Spin Physics Detector) предназначен для проведения экспериментов по физике спина при столкновениях пучков ядер лёгких элементов.

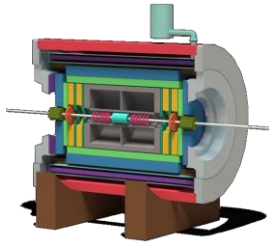
Актуальность темы исследования

По оценкам, приведенным в документе «TDR MPD: Data Acquisition System» от 2018 года, поток данных с детектора MPD составит, как минимум 7 ГБ/с. Эксперимент BM@N в ходе 8го сеанса за два месяца собрал более 400 ТБ данных. В эксперименте SPD в соответствии с «TDR of the Spin Physics Detector at NICA» поток получаемых данных приближаются к 20 ГБ/с.



BM@N

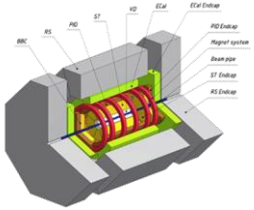
Работает
Монте-Карло,
Анализ,
Обработка данных с детектора
8й сеанс: 436 ТВ



NICA MPD

Декабрь 2025
Монте-Карло,
Анализ

Планируемый поток данных: 7 GB/s



SPD NICA

~ 2028
Монте-Карло,
Анализ

Планируемый поток данных: 20 GB/s

Обработка, хранение, передача и анализ таких объёмов данных требует большое количество вычислительных ресурсов и ресурсов хранения данных.

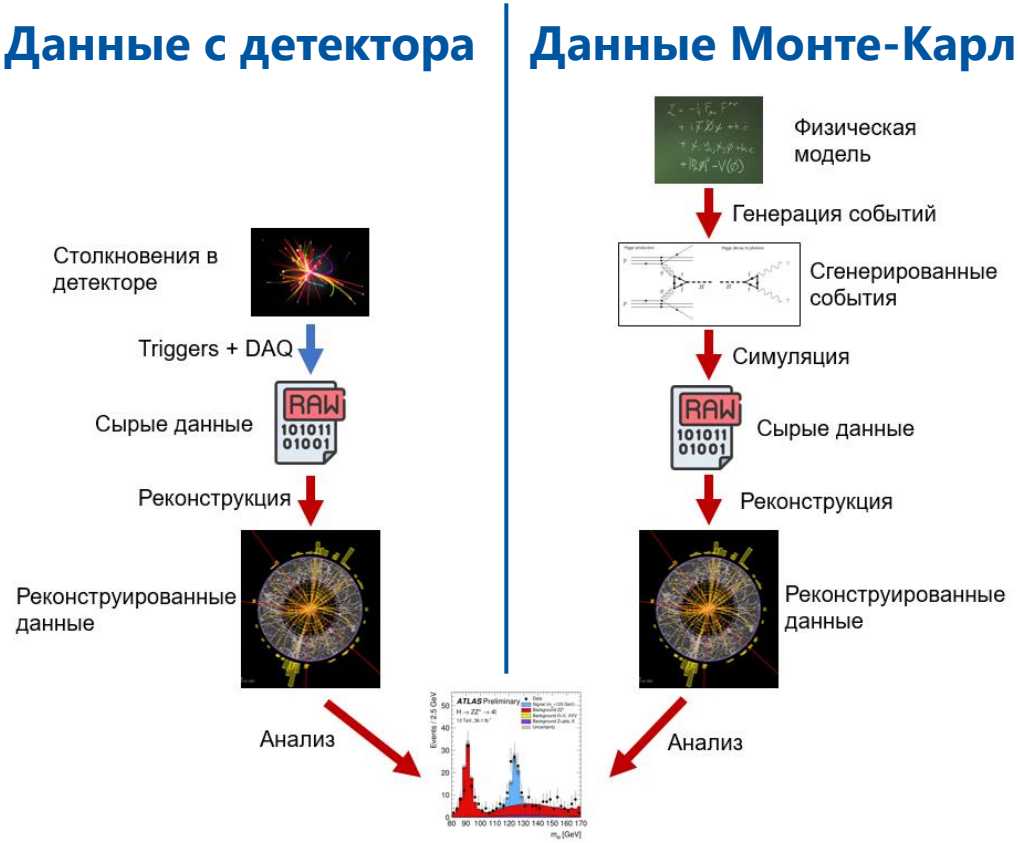
Цели и задачи исследования

Целью диссертации является:
построение распределённой гетерогенной вычислительной среды для обработки данных NICA.

Задачи:

1. Исследовать различные подходы к интеграции географически распределённых гетерогенных вычислительных ресурсов.
2. Построить распределённую гетерогенную вычислительную среду для решения задач оффлайн компьютинга экспериментов на ускорителе NICA.
3. Создать программный инструментарий для интеграции облачных инфраструктур на основе OpenNebula в распределённую гетерогенную среду.
4. Разработать и реализовать систему мониторинга пользовательских задач и передач данных.
5. Разработать и реализовать методику анализа производительности ресурсов входящих в созданную распределённую систему.

Схема работы оффлайн компьютинга



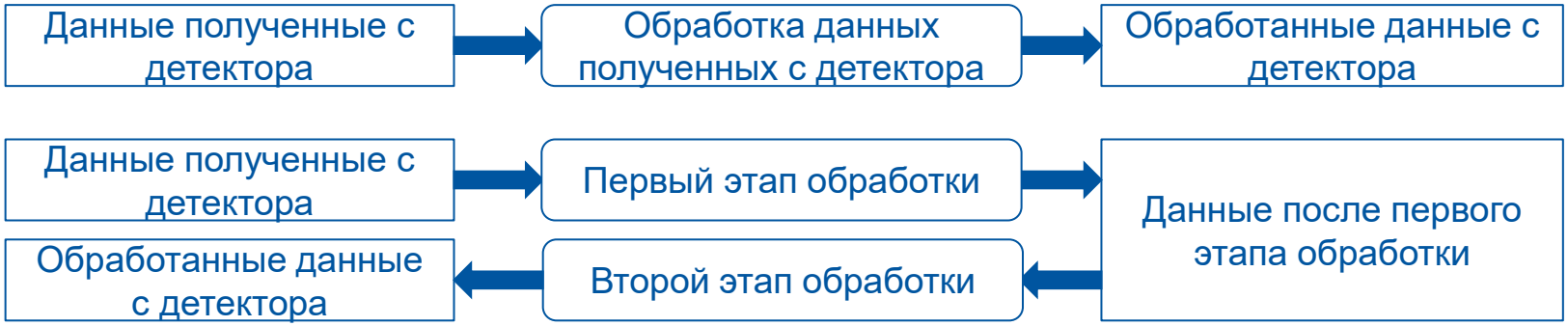
Данные с детектора: собираются системой считывания эксперимента и затем проходят фильтрацию на аппаратных или программных средствах. Затем они проходят процесс реконструкции, в результате которого получаются исходные данные для дальнейшего анализа.

Данные Монте-Карло – это полностью сгенерированные события. Сначала они проходят процесс симуляции во время которого симулируется их взаимодействия с детектором и электроникой. В результате получаются данные того же формата, что и данные получаемые с детектора. После этого они фильтруются, реконструируются и анализируются с помощью того же ПО, что и данные с детектора.

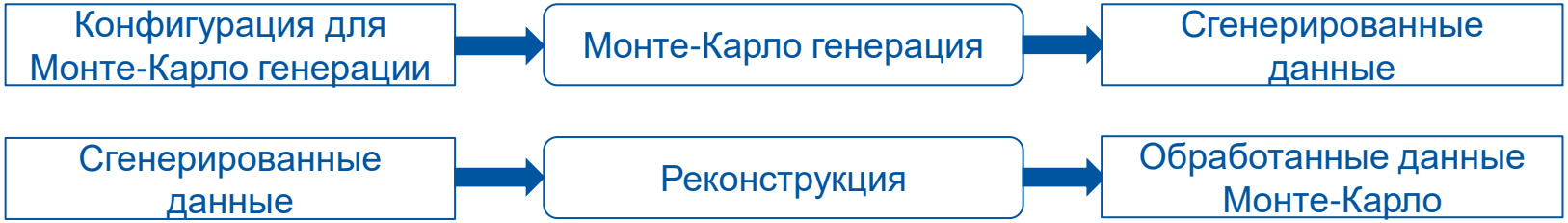
В данной работе акцент сделан на оффлайн вычисления (красные стрелки на рисунке выше). Однако, необходимо упомянуть, что в последнее время граница между триггерами и оффлайн вычислениями может размываться в моделях обработки некоторых экспериментов.

Основные классы задач связанных с вычислениями

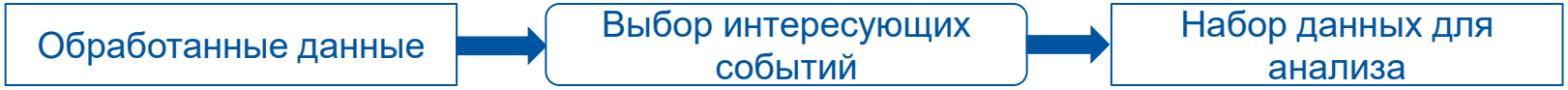
Обработка экспериментальных данных



Генерация и обработка данных Монте-Карло



Анализ данных

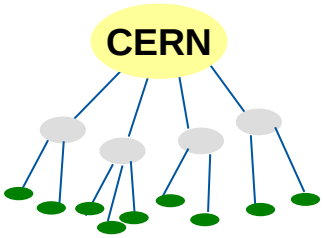


Грид вычисления

При отсутствии единого вычислительного ресурса, который способен полностью решить задачи компьютинга возможно использовать географически распределённую вычислительную систему.

ЦЕРН определят грид следующим образом: сервис для совместного использования вычислительных мощностей и ёмкости для хранения данных через Интернет.

Грид (определение Буйя/Венугопала)– это тип параллельной и распределённой системы, который позволяет динамически организовывать совместное использование, выбор и агрегирование географически распределённых автономных ресурсов в зависимости от их доступности, возможностей, производительности, стоимости и требований пользователей к качеству обслуживания.



Вычислительные ресурсы доступные экспериментам на ускорительном комплексе NICA

Характеристики вычислительных ресурсов

Ресурс	Аутентификация	Авторизация	Файловая система	Управление ресурсами	Количество доступных CPU ядер
Tier1	x509	VOMS	xfs	ARC(SLURM)	1500
Tier2	x509	VOMS	xfs	ARC(SLURM)	1000
Govorun	локальная	локальная	zfs/ Lustre	SLURM	200-2000
NICA кластер	локальная	локальная	nfs	SLURM	1000
DCC кластер	SSO	локальная	CEPH	SLURM	1000
Облако ОИЯИ	SSO	локальная	CEPH	OpenNebula	100
Облака стран-участниц ОИЯИ	локальная	локальная	CEPH	OpenNebula	650
Ресурсы коллабораций	-	-	-	-	-

Всего вычислительных ядер: более 5450

Системы хранения доступные экспериментам на ускорительном комплексе NICA

Характеристики систем хранения

	DDC CEPH	NICA EOS	Cloud CEPH	MLIT EOS	MLIT CTA	MLIT dCache
Система	CEPH	EOS	CEPH	EOS	CTA	Enstore
Аутентификация	локальная	локальная	локальная	x509 / локальная	x509 / локальная	x509
Авторизация	локальная	локальная	локальная	x509 / локальная	x509 / локальная	x509
Протокол удалённого доступа	-	xrootd	s3	xrootd / http	xrootd / http	xrootd
Выделенные ресурсы для NICA	есть	есть	-	есть	есть	есть
Поддержка FTS	-	требуется настройка	-	присутствует	присутствует	присутствует

Требования к системе организации распределённых вычислений и хранения данных

Существует набор систем которые можно использовать для организации распределённых вычислительных систем.

Наиболее важные требования к распределённой системе для организации компьютинга для экспериментов на ускорительном комплексе NICA:

- Открытый исходный код.
- Возможность модификации.
- Интеграция гетерогенных географически распределённых вычислительных ресурсов и систем хранения.
- Расширяемость и масштабируемость.
- Работа с наиболее популярными системами пакетной обработки данных, облаками и суперкомпьютерами.
- Управление потоками задач.
- Управление данными.
- Учёт потреблённых ресурсов.
- Поддержка автоматизации.
- Поддержка нескольких независимых групп пользователей/виртуальных организаций.

Существующие программные комплексы для организации распределённых вычислений и хранения данных

Из всех систем, наиболее подходящими кандидатами на роль система для организации распределённых вычислений можно выделить следующие.

Платформа DIRAC

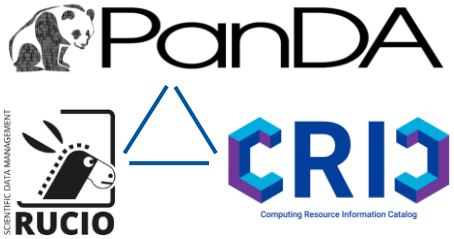
С 2002 года разрабатывалась для эксперимента LHCb. С 2009 года разрабатывается как универсальный инструмент для создания построения распределённых систем.

Язык: Python

Единая система для управления задачами и данными.

Единое решение для организации компьютинга

Экосистема PanDA



С 2005 года разрабатывалась для эксперимента ATLAS. С 2013 года разрабатывается для использования за пределами ATLAS.

Язык: Python

Система PanDA работает вместе с системами Rucio (управление данными) и CRIC (информационная система)

Экстремальная масштабируемость

Оба подхода способны решать задачи компьютинга экспериментов на ускорительном комплексе NICA.

Выбор платформы DIRAC

Для данной работы была выбрана платформа DIRAC Interware по следующим причинам:

1. Достаточная производительность при управлении задачами и данными.

На примере экспериментов Belle2, BES-III, France-Grilles. LHCb достигали 350 тысяч одновременно работающих задач при задействовании всех доступных ресурсов коллаборации.

2. Открытое сообщество пользователей, администраторов, разработчиков.

Ежегодные рабочие совещания DIRAC Users' Workshop, регулярные онлайн совещания.

3. Простота эксплуатации благодаря работе в рамках единой системы.

Для стабильной работы и развития системы на основе DIRAC достаточно 1 FTE.

4. Поддержка Multi-VO.

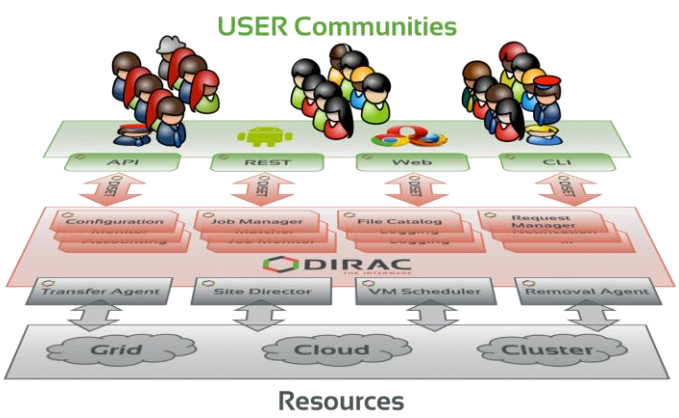
Значительно экономит усилия. Позволяет в рамках одной установки платформы поддерживать множество виртуальных организаций.

5. Наличие опыта в разработке компонентов DIRAC и его администрировании и налаженный канал связи с разработчиками.

Выполнение проекта создания системы мониторинга сервисов для эксперимента BES-III.

Платформа DIRAC Interware

DIRAC это платформа которая предоставляет все компоненты для построения распределённой инфраструктуры и объединяющей различные типы ресурсов, что позволяет использовать их единым образом.



Web

CLI

API

REST

Платформа DIRAC строится из компонентов четырёх видов:

1. **База данных** – хранит набор данных релевантных в рамках одной из подсистем. Предоставляет доступ только сервисам и агентам.
2. **Сервис** – служит для приёма и исполнения запросов от клиентов.
3. **Агент** – программа работающая периодически с определённым интервалом (аналог cron).
4. **Команда** – программа запускаемая для выполнения определённой операции

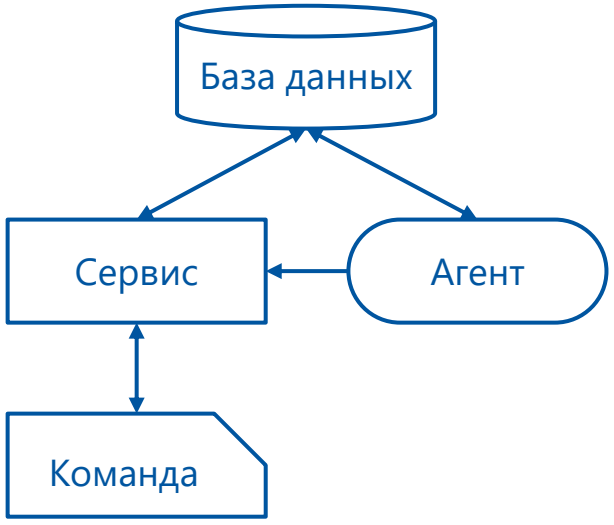
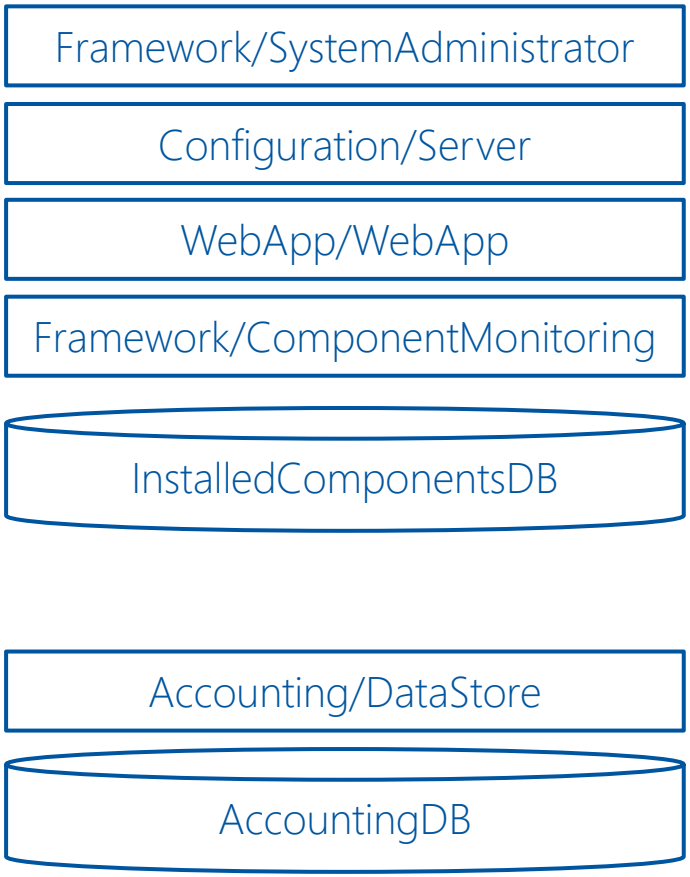


Схема взаимодействия компонентов DIRAC

Схема установки компонентов DIRAC

Шаг 1. Установка базовых компонентов платформы DIRAC



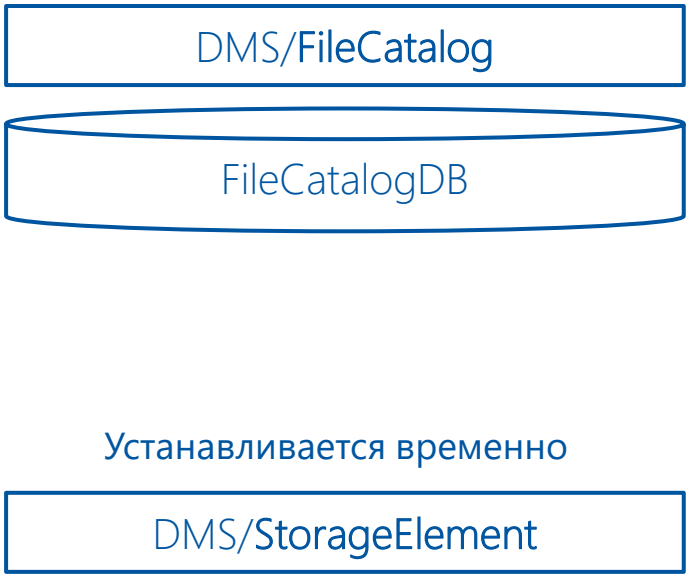
Первый шаг в установке платформы DIRAC – это установка и настройка базовых компонентов, без которых невозможно функционирование компонентов связанных с управлением задачами и данными.

Особенностью данных компонентов является то, что они взаимодействуют со всеми остальными.

В рамках первого шага так же устанавливаются компоненты связанные с системой учёта потреблённых ресурсов.

Схема установки компонентов DIRAC

Шаг 2. Установка компонентов отвечающих за управление данными.



Прежде чем устанавливать компоненты отвечающие за работу с данными необходимо подготовить систему управления данными.

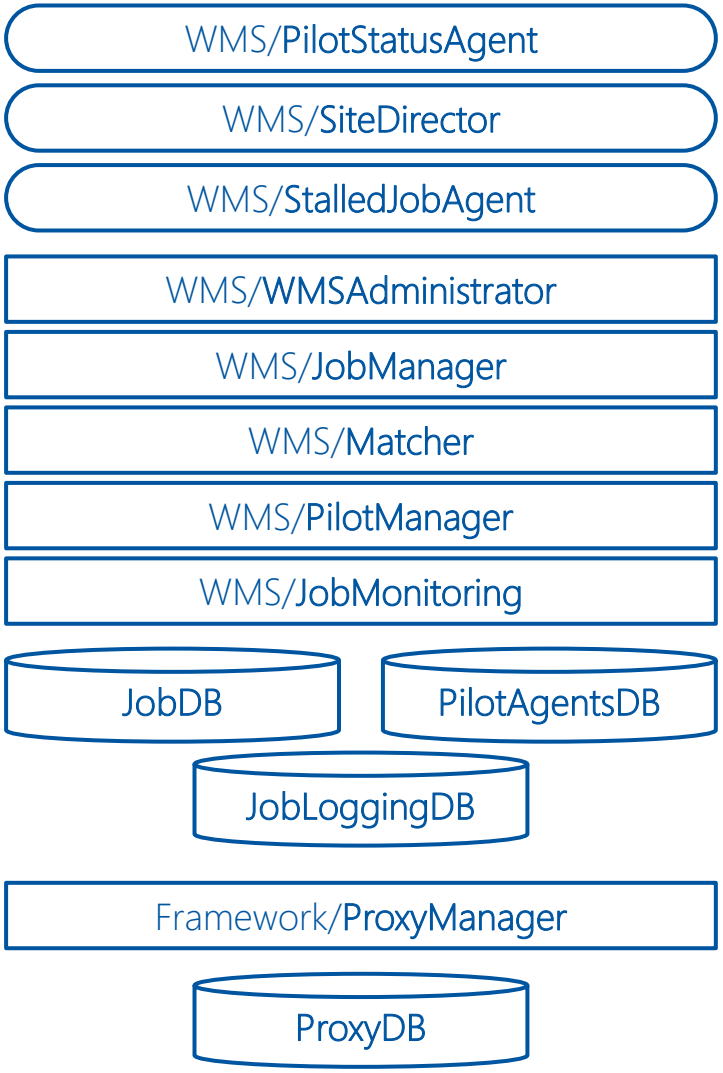
Система управления данными в DIRAC состоит из сервиса файлового каталога и соответствующей базы данных.

Тестирование данной системы предполагает установку временной системы хранения, которая выполняет те же функции, что EOS, dCache и др.

С использованием временной системы хранения становится возможным протестировать работу файлового каталога и его взаимодействие с системой учёта ресурсов.

Схема установки компонентов DIRAC

Шаг 3. Установка компонентов отвечающих за управление задачами.



Следующий шаг – это установка и настройка компонентов работа которых связана с управлением задачами.

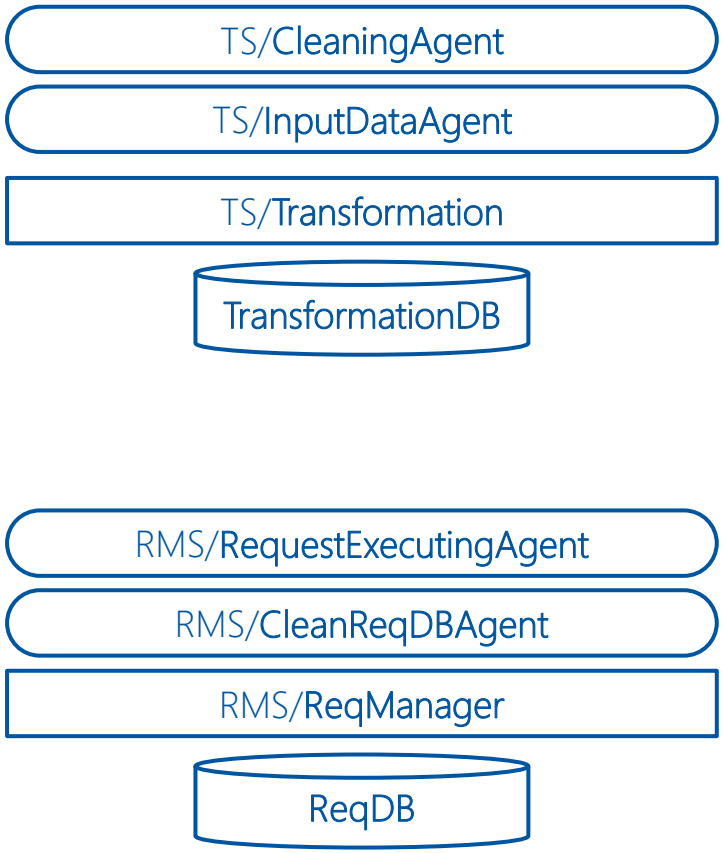
После установки этих компонентов становится возможным провести тесты. Для тестирования создаётся временный вычислительный ресурс, который запускает задачи прямо на сервере где установлены компоненты DIRAC. Тестирование призвано показать, что все компоненты связанные с управлением задачами корректно взаимодействуют.

Тестирование проверяет:

1. Создание задач, удаление задач.
2. Успешное выполнение задач и выполнение задач с ошибкой.
3. Корректное получение логов и статусов задач.
4. Корректное отображение данных в системе учёта ресурсов, что была установлена на предыдущем этапе.

Схема установки компонентов DIRAC

Шаг 4. Установка компонентов отвечающих за автоматизацию процессов управления данными и задачами.



Финальный шаг опционален и призван установить компоненты отвечающие за автоматизацию процессов запуска задач и передач данных.

За автоматизацию задач отвечает набор компонентов называемый Transformation System. Данная система позволяет установить набор правил в соответствии с которыми будут создаваться задачи. Использование данной системы полезно уже тогда когда все рабочие процессы и потоки данных надёжно налажены.

За автоматизацию передачи данных отвечает Request Management System. Использование данной системы позволяет создавать запросы связанные с удалением, копированием и репликацией данных. Использование данной системы оправдано при появлении сложных правил связанных с политиками хранения данных.

Ресурсы на которых работают компоненты DIRAC

Cloud

	dirac-conf	dirac-web	dirac-sl6	dirac-services
OS	CentOS	CentOS	ScientificLinux	CentOS
Version	7.5	7.5	6.10	7.5
CPU cores	4	4	2	8
RAM size	8 GB	8 GB	2 GB	16 GB

CICC

	dirac-services
OS	AlmaLinux
Version	9.5
CPU	14 cores, Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz
RAM size	128 GB
SSD size	1 TB

Изначально для разворачивания DIRAC использовалось облако ОИЯИ. Все сервисы и агенты были разделены между четырьмя виртуальными машинами с сохранения возможности их масштабирования.

После анализа наибольшая нагрузка оказалась на I/O локального хранилища. Загрузка CPU и RAM оказалась незначительной. С целью упрощения операций администрирования в момент обновления платформы DIRAC был осуществлён переход на физический сервер в рамках ЦИВК с SSD диском.

Интеграция ресурсов в DIRAC

Системы хранения

Интеграция системы хранения в DIRAC предполагает возможность проведения с помощью инструментов DIRAC базовых операций с файлами на интегрируемом хранилище.

В платформу DIRAC возможно интегрировать ресурсы позволяющие проводить аутентификацию по сертификату X509 и авторизацию с использованием VOMS.

Интеграция требует создания конфигурации новой системы хранения в конфигурационной системе DIRAC.

В DIRAC в ОИЯИ были интегрированы системы хранения:
Дисковое хранилище EOS
Ленточное хранилище dCache/Enstore
Ленточное хранилище CTA;

Вычислительные ресурсы

Интеграция вычислительного ресурса предполагает возможность использовать данный вычислительный ресурс для выполнения задач из очереди задач DIRAC.

В платформу DIRAC возможно интегрировать вычислительные ресурсы на которых возможно:

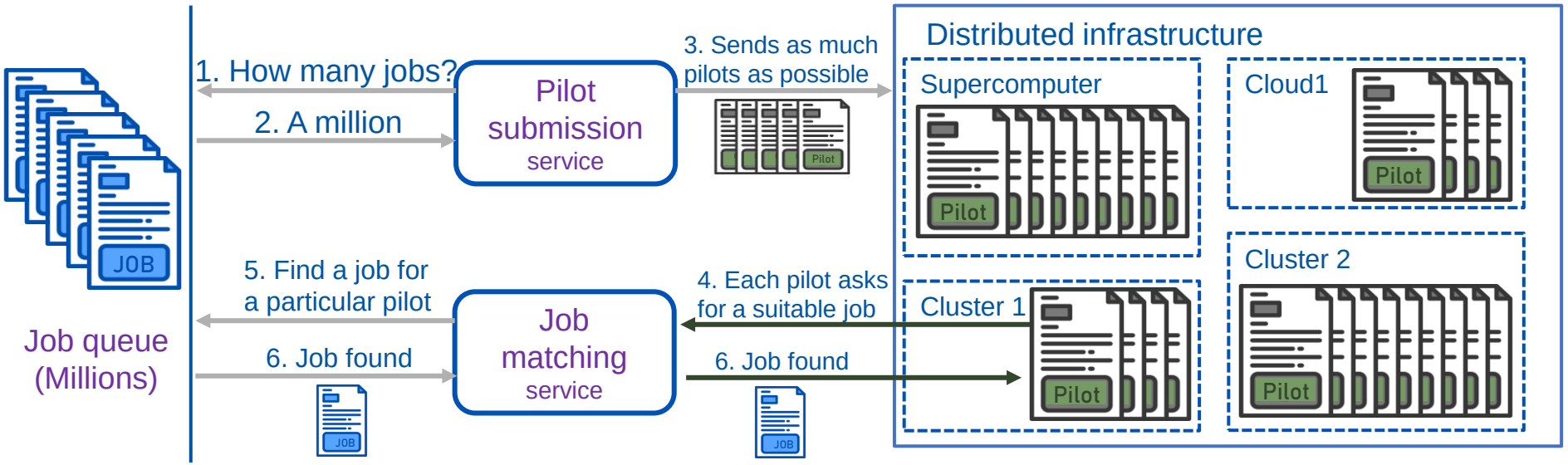
1. Запустить пилотную задачу.
2. Получить пользовательскую задачу.
3. Получать и передавать данные на интегрированные системы хранения.

Интеграция требует создания конфигурации нового вычислительного ресурса в конфигурационной системе DIRAC.

В DIRAC в ОИЯИ были интегрированы системы хранения:
Tier1, Tier2, суперкомпьютер Говорун, NICA кластер, DAQ Data Center кластер, ресурсы университетов UNAM и МИФИ.

Пилотный механизм запуска задач

Механизм использования пилотов для выполнения задач



Пилот - это специальная задача, которую можно отправить на кластер, суперкомпьютер, облако или практически любой другой ресурс, где она способна запуститься. После запуска пилот проверяет окружение, объём доступной оперативной памяти, производительность процессора и запрашивает у сервиса сопоставления заданий (JobMatching) подходящую пользовательскую задачу. Получив задачу, пилот инициирует её выполнение, выступая в роли обёртки для пользовательской задачи. После завершения пользовательской задачи пилот отправляет сводную информацию о ее выполнении и запрашивает новую пользовательскую задачу либо завершает выполнение.

Максимальная загрузка вычислительных ресурсов достигается при:

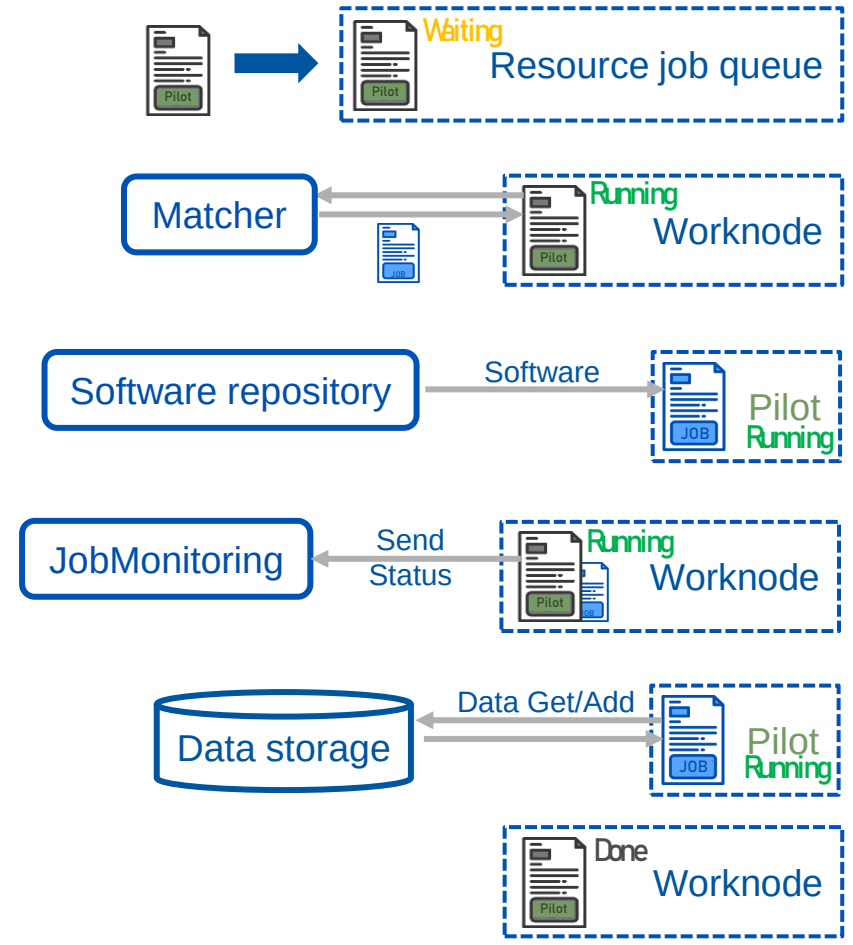
1. $N_{jobs\ in\ queue} >>> N_{available\ job\ slots}$
2. $T_{job\ execution} >>> T_{launch\ pilot}$

   Времени выполнения задачи значительно больше времени запуска пилота
3. $\frac{Input + Output\ data}{Execution\ time} \rightarrow 0\ byte/s$

   Операции с данными минимальны

Основные этапы интеграции вычислительных ресурсов

1. Вычислительный ресурс способен принимать в локальную очередь пилотные задачи DIRAC.
2. Пилотные задачи способны начать выполнение и запросить пользовательские задачи из очереди задач DIRAC.
3. Пользовательские задачи способны запускать программное обеспечение из репозиториев CVMFS экспериментов работающих на ускорительном комплексе NICA.
4. Пилотные задачи способны оповещать центральные сервисы DIRAC о статусе своей работы.
5. Пользовательские задачи способны получить доступ к данным находящимся под управлением DIRAC и способны сохранять туда новые данные.
6. После завершения пользовательской задачи пилотная задача может освободить занимаемое ей место и успешно завершиться.



По такой схеме были интегрированы все вычислительные ресурсы, кроме облачных.

Механизм интеграции облачных ресурсов построенных на основе OpenNebula

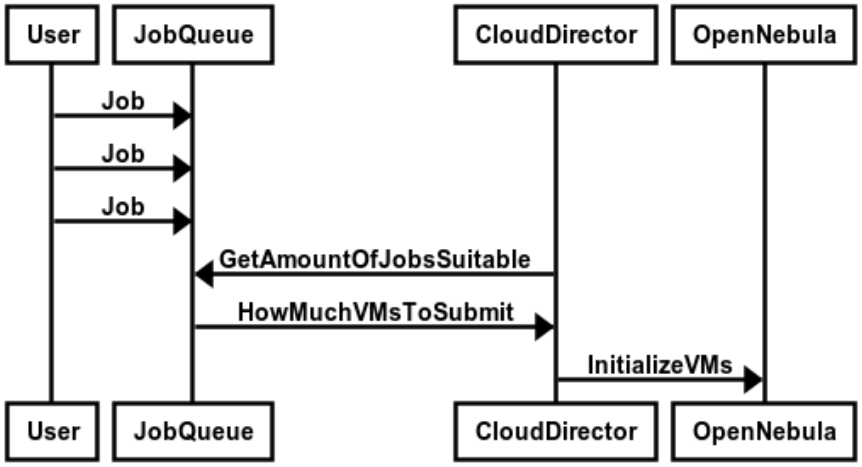


Схема запуска виртуальных машин

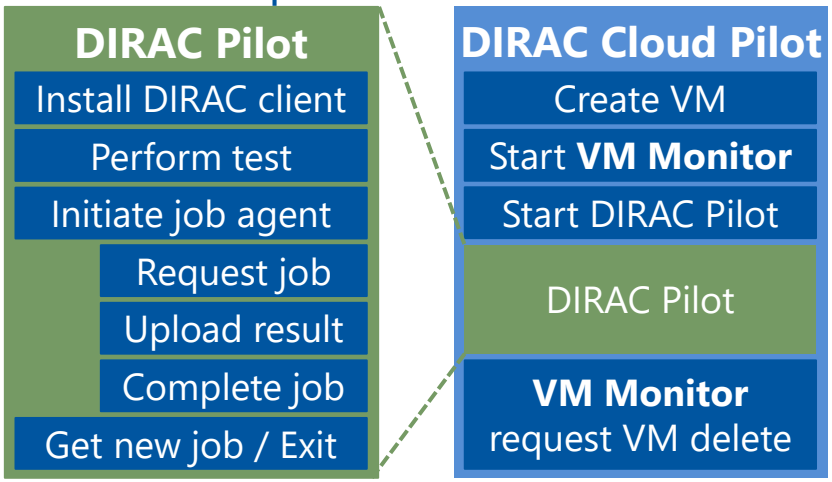


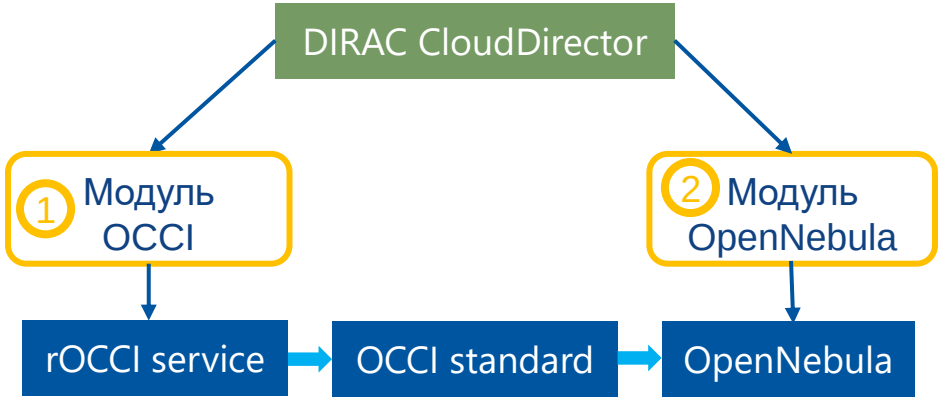
Схема запуска пилотных задач

1. Был разработан новый модуль для работы с облаками посредством общения с сервисом rOCCI в OpenNebula, который базируется на стандарте OCCI. Работа rOCCI сервиса оказалась нестабильной.
2. Был разработан новый модуль который работающий с OpenNebula напрямую посредством API.

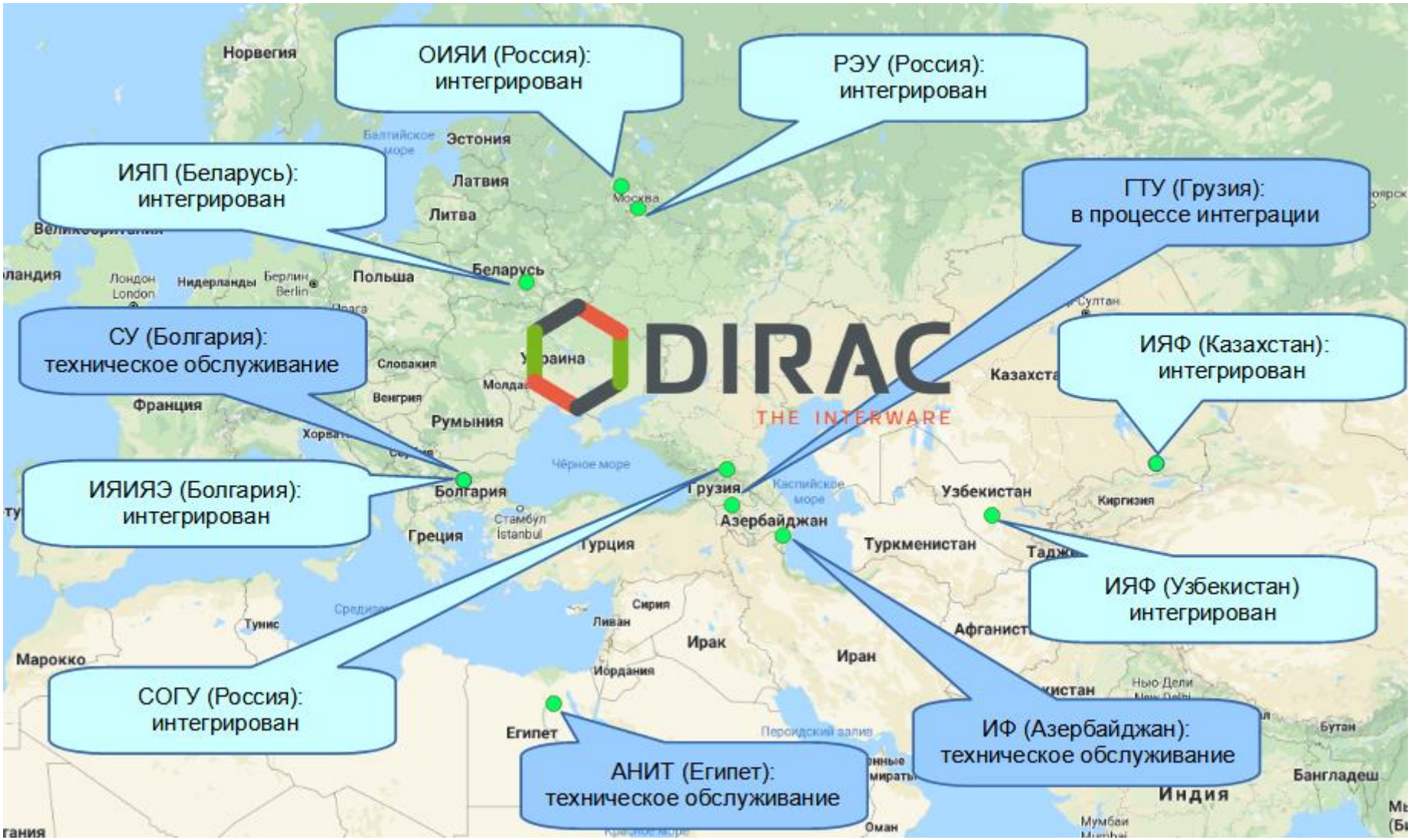
Разработанный модуль использовался в том числе и в IHEP в Пекине, для интеграции облака ОИЯИ.

Разработанные модули реализовывали функции:

1. Создания виртуальной машины (ВМ)
2. Запуска сервиса VM Monitor
3. Запуска пилотной задачи
4. Удаления ВМ



Общая схема облачных ресурсов интегрированных с использованием разработанного модуля

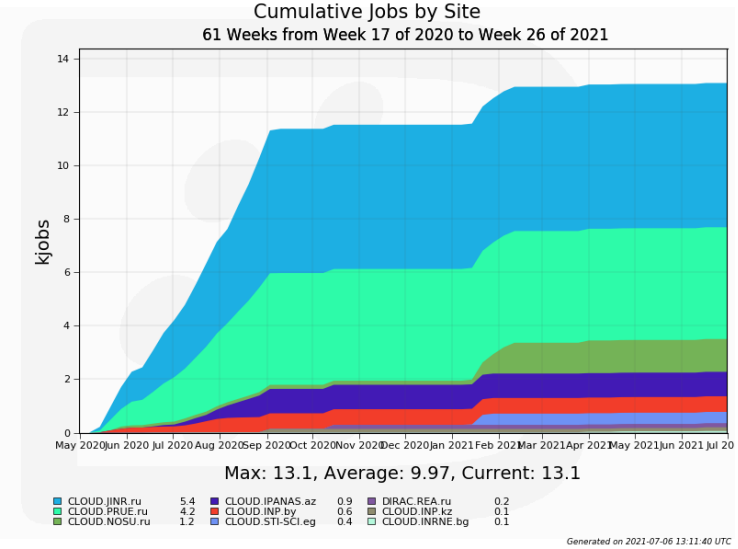


Тестирование интегрированных облачных ресурсах в распределённой информационно-вычислительной среды

Интегрированные облачные ресурсы были протестированы в проекте Folding@Home.
В качестве задач выбирались задачи моделирования молекулярной динамики вируса SARS-CoV-2.
Особенность задач проекта Folding@Home – малый размер входных и выходных данных.
Основные условия тестирования:

1. Проверка стабильности работы под нагрузкой
2. Постоянный контроль нагрузки
3. Учёт потреблённых ресурсов
4. Выполнения задач анализа вируса SARS-CoV-2

Всего было выполнено ~13000 задач.
Среднее время выполнения задачи ~ 15 часов.
Нормализованное время: 138000 HS06 дней.
Walltime: 23.5 лет



Team: Joint Institute for Nuclear Research

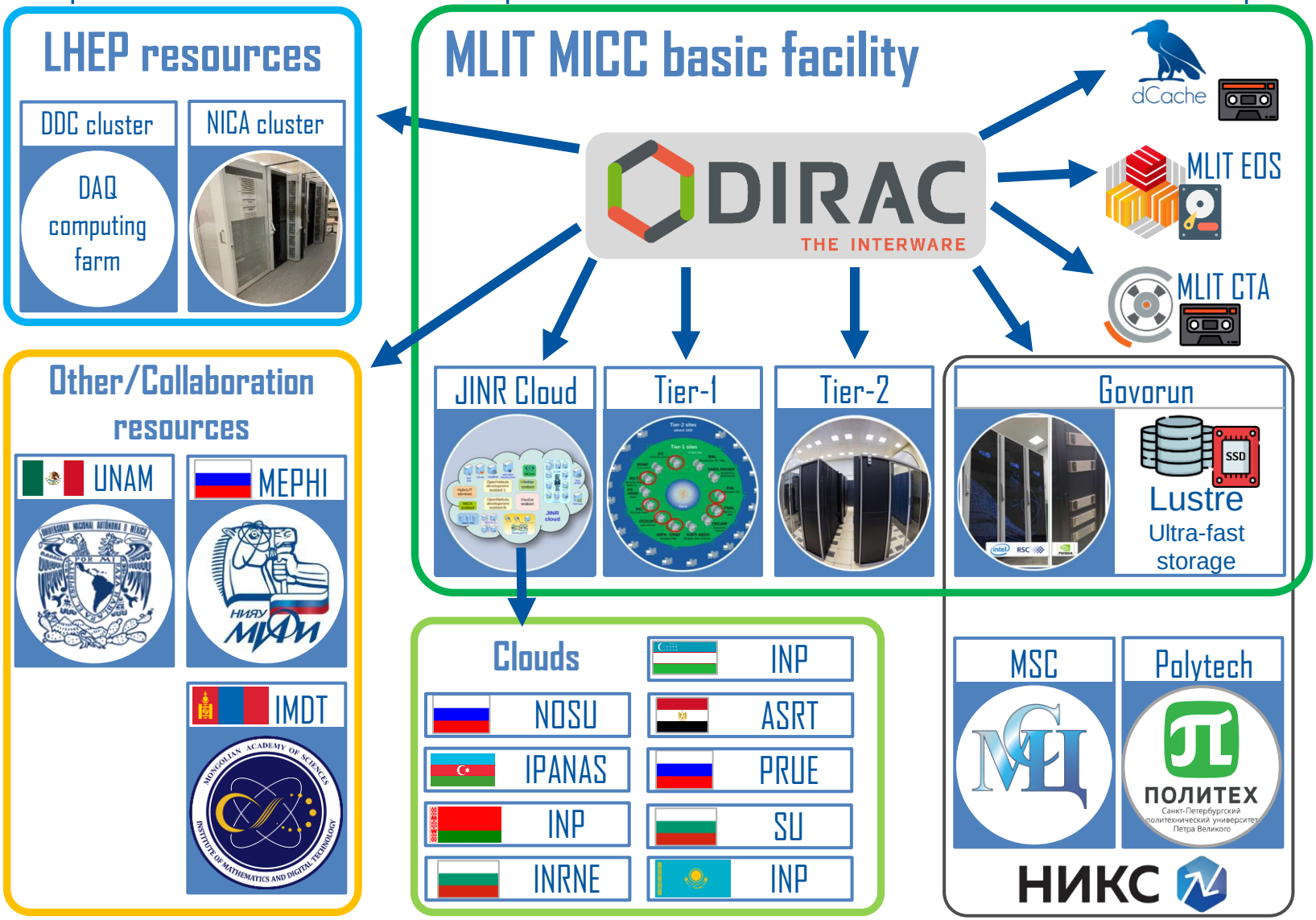
Date of last work unit	2021-06-19 14:13:01
Active CPUs within 50 days	36
Team Id	265602
Grand Score	31,898,062
Work Unit Count	13,418
Team Ranking	7349 of 226308
Homepage	http://www.jinr.ru/main-en/

Team members

Rank	Name	Credit	WUs
89 066	CLOUD.JINR.ru	12,645,224	5,355
105 254	CLOUD.PRUE.ru	9,453,851	4,175
158 292	CLOUD.NOSU.ru	4,154,811	1,352
238 209	CLOUD.IPANAS.az	1,542,618	910
242 757	CLOUD.INP.by	1,465,167	599
269 535	CLOUD.INP.kz	1,101,339	433
298 441	CLOUD.STI-SCI.eg	817,728	381
356 608	DIRAC.REA-Parallel.ru	471,543	155
434 015	CLOUD.INRNE.bg	245,781	58

Было продемонстрировано, что реализованный метод интеграции облачных ресурсов работает для задач не требующих большого количества данных.

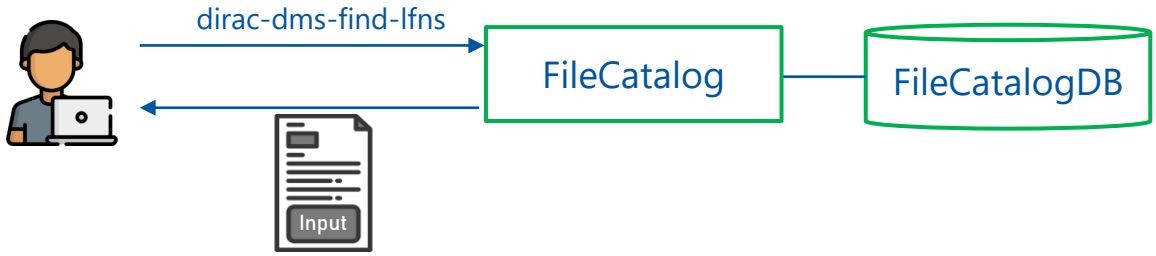
Общая схема построенной географически распределённой гетерогенной вычислительной среды



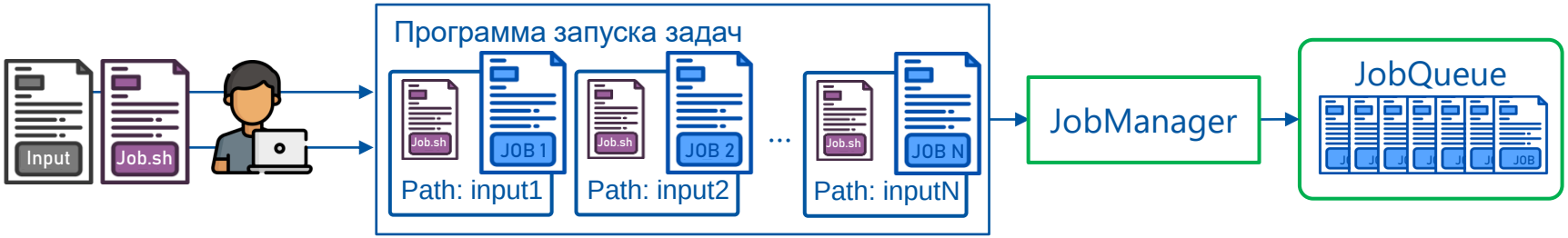
Автоматизированный запуск больших пакетов задач

Запуск больших пакетов задач предложено проводить в два этапа:

1. Получение списка входных файлов, которым требуется обработка.



2. Автоматизированный запуск задач.



Программа запуска задач для каждого из входных файлов полученных на первом этапе создаёт соответствующие задачи и отправляет их в очередь задач.

Пример задачи:

```
1. JobName: Имя_задачи_1
2. Executable: /bin/sh
3. Arguments: job.sh input1
4. InputSandbox: job.sh
5. OutputSandbox: std.out, std.err
```

Запуск прикладного ПО, который в итоге определяет тип задачи (реконструкция, моделирование, анализ), определяется пользователем в файле *Job.sh* .

Задача мониторинга и анализа работы распределённой гетерогенной вычислительной среды

Задачи возникающие при эксплуатации географически распределённой вычислительной среды при выполнении больших пакетов задач:

1. Проверка загрузки CPU конкретной задачи.
2. Проверка загрузки RAM конкретной задачи.
3. Использование дискового пространства конкретной задачей.
4. Подсчёт скорости передачи данных в рамках конкретной задачи.
5. Подсчёт суммарной скорости передачи данных между вычислительными ресурсами и системами хранения.
6. Детектирование аномалий при выполнении больших пакетов задач.
7. Сбор статистики о ходе выполнения задач в рамках больших пакетов задач.
8. Сравнение хода выполнения одного пакета задач с другим
9. Поиск причин возникновения проблем с выполнением задач.

1 Мониторинг пользовательских задач

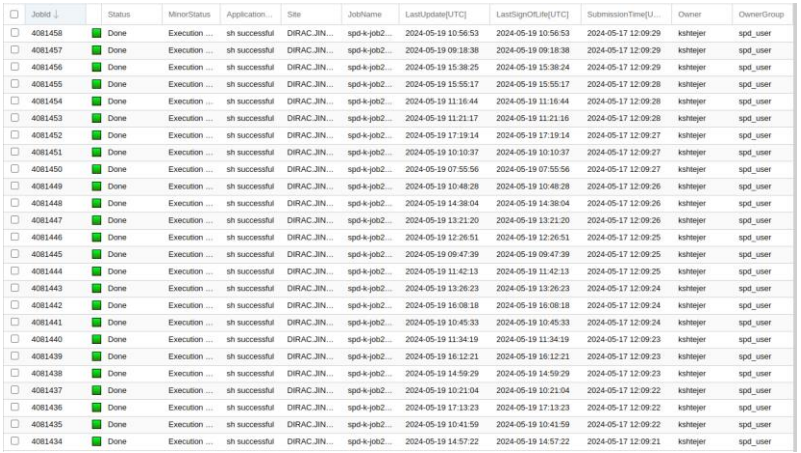
2 Мониторинг передач данных

3 Система анализа выполнения больших пакетов задач

Глава 3. Разработка систем мониторинга и анализа производительности распределённой гетерогенной вычислительной среды

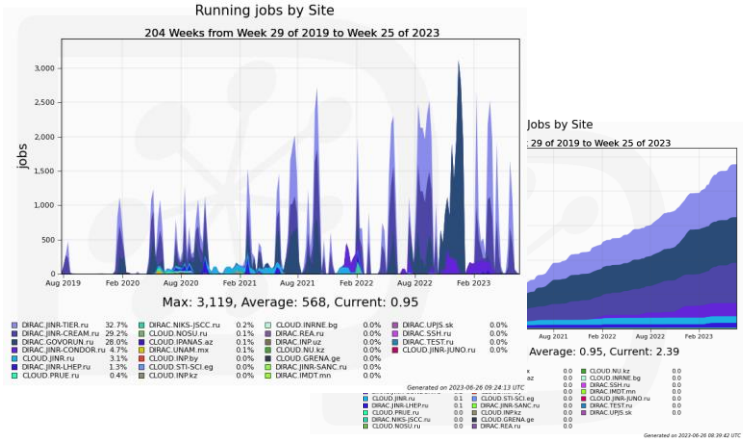
Платформа DIRAC предоставляет два средства которые частично позволяют решать данные вопросы: веб-интерфейс системы управления задачами и веб интерфейс системы аккаунтинга.

Веб-интерфейс системы управления задачами



JobId	Status	MinorStatus	ApplicationStatus	Site	JobName	LastUpdateUTC	LastSignOfLifeUTC	SubmissionTimeUTC	Owner	OwnerGroup
4081458	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 10:56:53	2024-05-19 10:56:53	2024-05-17 12:09:29	kshtejer	spd_user
4081457	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 09:18:38	2024-05-19 09:18:38	2024-05-17 12:09:29	kshtejer	spd_user
4081456	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 15:38:25	2024-05-19 15:38:24	2024-05-17 12:09:29	kshtejer	spd_user
4081455	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 15:55:17	2024-05-19 15:55:17	2024-05-17 12:09:28	kshtejer	spd_user
4081454	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 11:16:44	2024-05-19 11:16:44	2024-05-17 12:09:28	kshtejer	spd_user
4081453	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 11:21:17	2024-05-19 11:21:16	2024-05-17 12:09:28	kshtejer	spd_user
4081452	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 17:19:14	2024-05-19 17:19:14	2024-05-17 12:09:27	kshtejer	spd_user
4081451	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 10:10:37	2024-05-19 10:10:37	2024-05-17 12:09:27	kshtejer	spd_user
4081450	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 07:55:56	2024-05-19 07:55:56	2024-05-17 12:09:27	kshtejer	spd_user
4081449	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 10:48:28	2024-05-19 10:48:28	2024-05-17 12:09:26	kshtejer	spd_user
4081448	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 14:38:04	2024-05-19 14:38:04	2024-05-17 12:09:26	kshtejer	spd_user
4081447	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 13:21:20	2024-05-19 13:21:20	2024-05-17 12:09:26	kshtejer	spd_user
4081446	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 12:26:51	2024-05-19 12:26:51	2024-05-17 12:09:25	kshtejer	spd_user
4081445	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 09:47:39	2024-05-19 09:47:39	2024-05-17 12:09:25	kshtejer	spd_user
4081444	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 11:42:13	2024-05-19 11:42:13	2024-05-17 12:09:25	kshtejer	spd_user
4081443	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 13:26:23	2024-05-19 13:26:23	2024-05-17 12:09:24	kshtejer	spd_user
4081442	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 16:08:18	2024-05-19 16:08:18	2024-05-17 12:09:24	kshtejer	spd_user
4081441	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 10:45:33	2024-05-19 10:45:33	2024-05-17 12:09:24	kshtejer	spd_user
4081440	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 11:34:19	2024-05-19 11:34:19	2024-05-17 12:09:23	kshtejer	spd_user
4081439	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 16:12:21	2024-05-19 16:12:21	2024-05-17 12:09:23	kshtejer	spd_user
4081438	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 14:59:29	2024-05-19 14:59:29	2024-05-17 12:09:23	kshtejer	spd_user
4081437	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 10:21:04	2024-05-19 10:21:04	2024-05-17 12:09:22	kshtejer	spd_user
4081436	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 17:13:23	2024-05-19 17:13:23	2024-05-17 12:09:22	kshtejer	spd_user
4081435	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 10:41:59	2024-05-19 10:41:59	2024-05-17 12:09:22	kshtejer	spd_user
4081434	Done	Execution ...	sh successful	DIRAC.JIN...	spd-k-job2...	2024-05-19 14:57:22	2024-05-19 14:57:22	2024-05-17 12:09:21	kshtejer	spd_user

Веб-интерфейс системы аккаунтинга



С помощью **веб-интерфейса системы управления задачами** можно отслеживать ход выполнения задач, анализировать распределения задач между ресурсами и наблюдать, возможные проблемы на конкретных ресурсах.

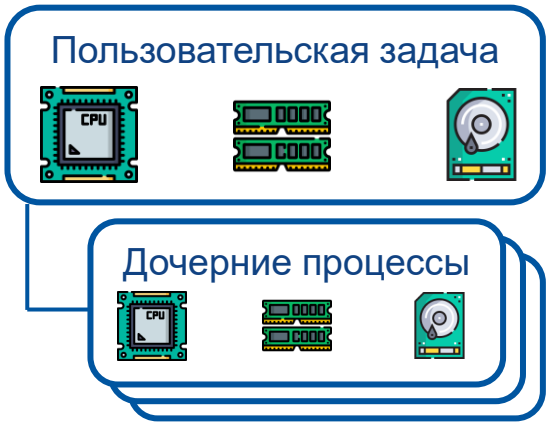
Веб-интерфейс системы аккаунтинга не позволяет получать точные данные из-за отсутствия стандартного интерфейса для экспорта данных и постоянной агрегации данных, которая не позволяет получить доступ к точным данным.

Готовых способов решить поставленные задачи в рамках платформы DIRAC не существует.

① Мониторинг пользовательских задач

Требуется периодически собирать данные о следующих показателях выполнения пользовательской задачи и её дочерних процессов:

1. Загрузка CPU
2. Объём используемой оперативной памяти
3. Объём данных которые читаются или пишутся на локальное хранилище



Существующие инструменты мониторинга можно разделить на две категории: локальные и централизованные.

К локальным инструмента можно отнести утилиты **`ps`, `top`, `htop`, `btop`** и т.п. Данные инструменты ставят своей целью предоставления оператору доступа к текущим показателям работающих процессов. Однако в них **не заложены функции сохранения показателей для последующего анализа**.

Существуют и централизованные системы мониторинга, такие как **`Nagios`, `Telegram + InfluxDB + Grafana`, `Prometheus`**. Они позволяют сохранять данные для последующего анализа, но в основном сосредоточены на сохранение информации об общей загрузке системы. Зачастую в них есть отдельные плагины позволяющие следить за стандартными процессами, такими как процессы веб-серверов, баз данных, отдельных сервисов. Для них можно было бы создать плагин, который бы собирал данные о работе пилотов. Но, для их работы потребуется установить **специальные клиенты** на все рабочие узлы распределённой сети.

① Мониторинг пользовательских задач

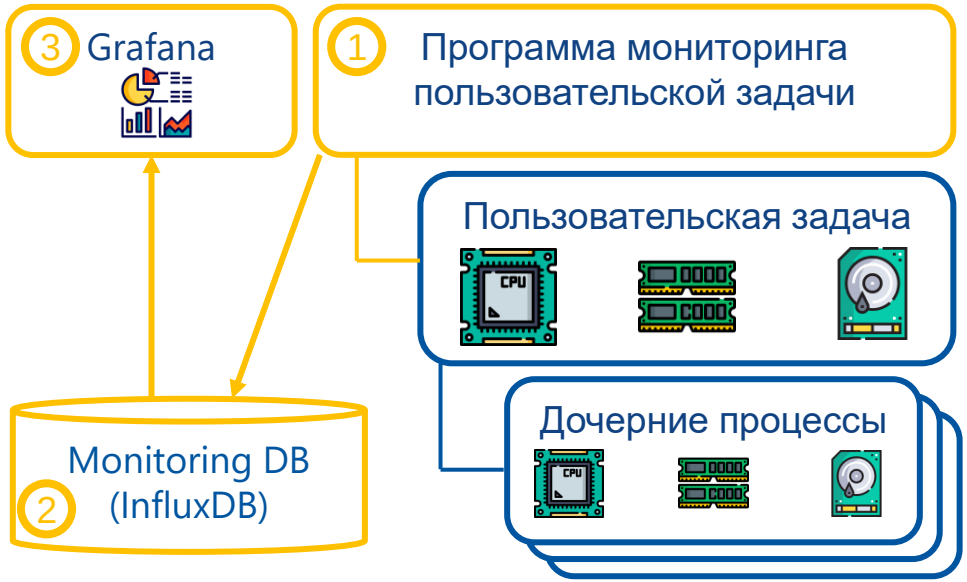
Для решения поставленной задачи было необходимо:

1. Разработать программу способную собирать данные о показателях работы пользовательских задач и их дочерних процессов с привязкой к ID задачи в системе DIRAC.
2. Периодически сохранять эту информацию в базе данных.
3. Разработать интерфейс для визуализации.

Для визуализации была выбрана система **Grafana** как наиболее универсальный инструмент работающий с большим количеством разных СУБД.

В качестве СУБД была выбрана **InfluxDB** так как она:

- Является СУБД временных рядов
- Позволяет гибко менять набор сохраняемых величин без необходимости менять схему базы данных
- Предоставляет защищённый http интерфейс для сохранения данных по сети



Отдельно, на языке **python** была разработана **программа мониторинга пользовательской задачи**, которая запускает пользовательскую задачу в виде дочернего процесса и с помощью библиотеки **psutil** способна получать доступ к текущим показателям работы пользовательской задачи .

① Мониторинг пользовательских задач

Созданная система позволяет запустить пользовательскую задачу таким образом, чтобы сохранить информацию о ходе выполнения задачи. Особенности системы:

- Для каждой задачи данные собираются и сохраняются в базу данных каждые 30 секунд.
- Сохраняется информация об ID задачи, загрузке CPU, RAM, количестве байтов записанных и прочитанных с диска.
- Данный подход позволяет как следить за задачами работающими в текущий момент времени, так и получать информацию о завершённых задачах.

Байт прочитано

Байт записано

Загрузка RAM

Скорость чтения

Скорость записи

Загрузка CPU (текущая)

Загрузка CPU (средняя за 20 мин)

Средняя загрузка CPU

Максимальная загрузка RAM

② Мониторинг передач данных

В соответствии с моделью запуска задач передача данных происходит в рамках выполнения пользовательских задач. Во время передачи данных, занятые задачей ресурсы CPU не используются.

В построенной инфраструктуре максимальная скорость потока передачи данных равна **100 МБ/с**. Однако, при массовой загрузке данных скорость передачи может быть ограничена многими факторами:

1. Скоростью чтения/записи локального хранилища.
2. Скоростью сети между рабочим узлом и сетевым оборудованием в кластере.
3. Скоростью сети между кластером и системой хранения.
4. Скоростью сети между узлом хранения данных и сетевым оборудованием системы хранения.
5. Скоростью чтения/записи системы хранения.
6. Скоростью работы баз данных в рамках системы хранения.

Проблема передачи данных между задачами и системами хранения усугубляется тем, что задачам необходимо делить каналы связи вместе другими процессами. Это могут быть другие задачи, параллельная передача данных между системами хранения, загрузка прикладного ПО, загрузка пакетов обновления.

② Мониторинг передач данных

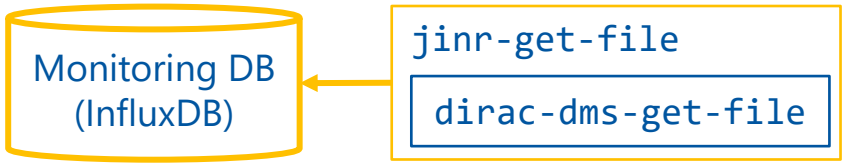
Для решения поставленной задачи необходимо:

1. Разработать метод позволяющий собирать данные о передачах данных.
2. Сохранять собранную информацию в базе данных.
3. Разработать интерфейс для визуализации.

Для каждой передачи данных в рамках пользовательских задач необходимо собирать следующую информацию:

1. ID задачи
2. IP адрес рабочего узла
3. Имя рабочего узла
4. Название вычислительно ресурса
5. Имя файла
6. Размер файла
7. Скорость передачи
8. Тип передачи (download, upload)
9. Статус передачи (success, fail)
10. Имя системы хранения

Для сохранения данных о передачах наиболее гибким, не требовательным к поддержке, и устойчивым к неполадкам решением является создание собственных программ, которые должны перенаправлять запросы на операции с данными к стандартным командам DIRAC параллельно извлекая и сохраняя информацию о передачах.



В качестве СУБД для сохранения данных о передачах была выбрана **InfluxDB**. Для визуализации – система **Grafana**. Причины выбора таких технологий такие же как и при разработке системы мониторинга задач.

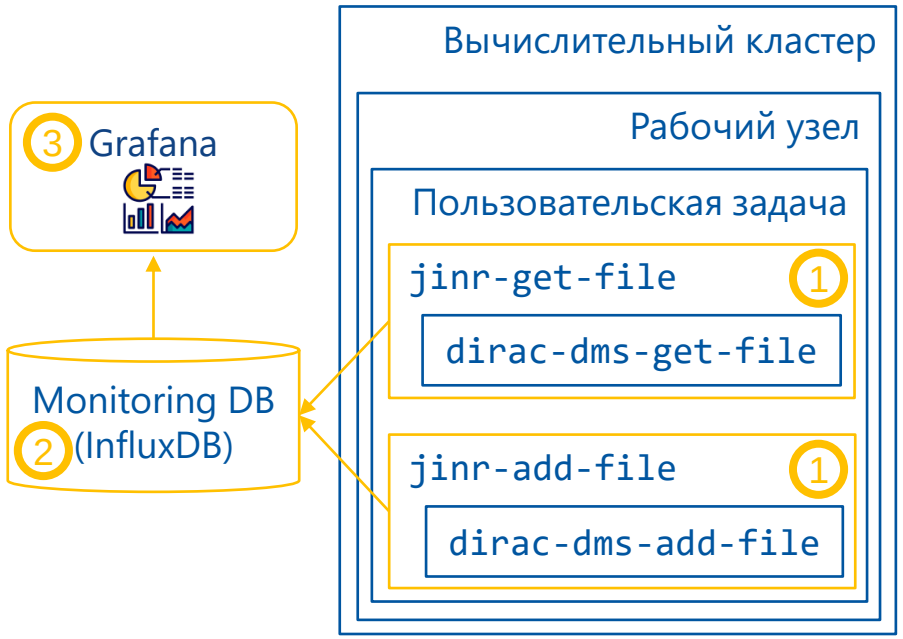
② Мониторинг передач данных

Для решения поставленной задачи необходимо:

1. Разработать метод позволяющий собирать данные о передачах данных.
2. Сохранять собранную информацию в базе данных.
3. Разработать интерфейс для визуализации.

Были разработаны специальные программы jinr-get-file и jinr-add-file. Данные программы принимают тот же набор параметров что и стандартные утилиты передачи данных DIRAC.

В момент вызова они собирают данные об **ID задачи**, **IP адресе рабочего узла**, **имени рабочего узла**, **названии вычислительного ресурса** и **типе передачи**. После чего фиксируют **время старта** передачи и перенаправляют параметры передачи стандартным командам и запускают их. После завершения передачи фиксируется **время окончания передачи**, сохраняется информация об итоговом **статусе передачи**, **названии системы хранения**, **имени передаваемого файла**, **размере файла**. Рассчитывается и сохраняется **скорость передачи**.



② Мониторинг передач данных

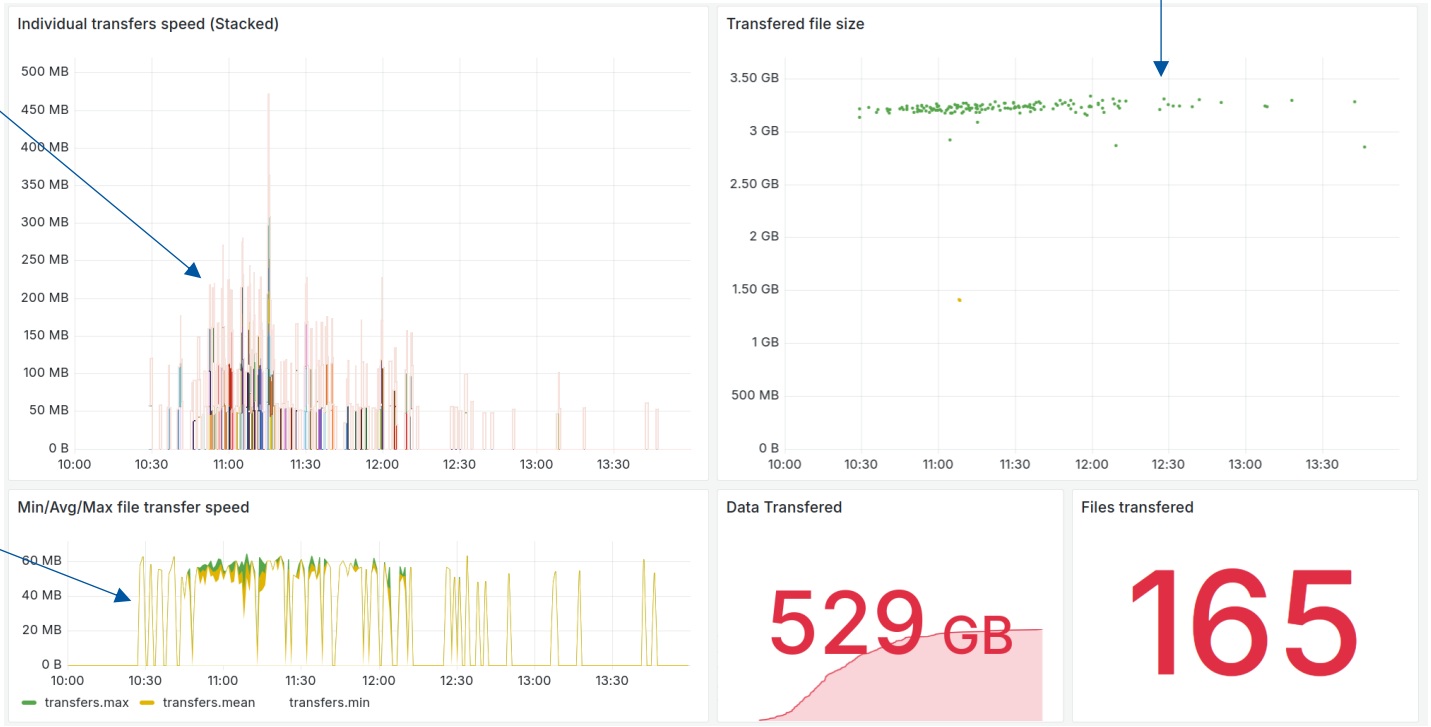
Созданная система позволяет анализировать передачи данных. Особенности системы:

- Запись данных активируется при использовании специально разработанных модулей
- Скорость передачи рассчитывается после завершения передачи
- Позволяет агрегировать данные по разным параметрам: вычислительным ресурсам, рабочим узлам, задачам, системам хранения, типам передач, статусам и т.п.

Размеры передаваемых файлов

Скорость передачи

Максимальная / Минимальная / Средняя скорость передачи файла



Система анализа выполнения больших пакетов задач

③

Факторы влияющие на скорость выполнения задач:

1. Скорость процессора.
2. Скорость передачи данных.
3. Скорость I/O при работе с оперативной памятью и локальной системой хранения.
4. Оптимизация прикладного ПО для конкретного вычислительного ресурса.

Можно сказать, что скорость процессора определяет скорость выполнения задачи, в то время как другие факторы её ограничивают. В идеальной ситуации время выполнения задачи должно прогнозироваться по следующей формуле:

$$Time = \frac{Amount\ of\ work}{CPU\ speed}$$

Основные вопросы которые возникают при выполнении больших пакетов задач следующие:

1. Эффективно ли выполняются задачи и можно ли улучшить эффективность их выполнения
2. В чём причина возникающих неполадок: сеть, особенности вычислительного ресурса, ошибки в прикладном ПО
3. Как один пакет задач отличается от другого с точки зрения выполнения в рамках распределённой инфраструктуры.

Система анализа выполнения больших пакетов задач

3

Платформа DIRAC собирает набор информации о тех задачах, которые выполняются с её помощь. Эти данные сохраняются во внутреннюю базу данных. Получить информацию о конкретной задаче можно в рамках веб-интерфейса DIRAC. Если необходима более сложная выборка – необходимо использовать SQL запросы.

Parameters for JobID:16638271

Name	Value
AgentLocalSE	JINR-EOS
CEQueue	htcondorce-juno
CPUNormalizationFactor	31.1
GridCE	cloud-ce01.jinr.ru
HostName	10-220-17-228.jinr.ru
JobType	User
JobWrapperPID	35
LastUpdateCPU(s)	2.0
LoadAverage	6.19
LocalAccount	juno
MatcherServiceTime	1.9406509399414062
Memory(kB)	104053668
MemoryUsed(kb)	65056.0
ModelName	Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
NormCPUTime(s)	46743.3
OutputSandboxMissingFiles	jsub.out, std.err
PayloadPID	40
PilotAgent	8.0.26
Pilot_Reference	htcondorce://cloud-ce01.jinr.ru/11189.24
ScaledCPUTime(s)	50462.09228904248
TotalCPUTime(s)	1503.0
WallClockTime(s)	1622.5753147602081

Данные распределены между двумя таблицами в базе данных JobDB: Jobs и JobParameters.

В таблице Jobs указаны все задачи: ID, статусы, время начала и конца выполнения, принадлежность к пользователю и группе.

В процессе выполнения задачи пилот заполняет таблицу JobParameters. Туда добавляется информация о пилоте, о том сколько времени работала задача, какова была оценка бенчмарка DB12, и многие другие.

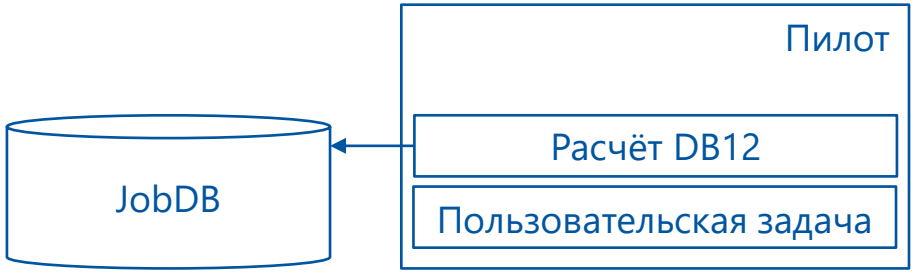
Для решения поставленной задачи необходимо использовать значение walltime и результаты бенчмарка DB12.

Система анализа выполнения больших пакетов задач

3

Бенчмарк – контрольная задача, необходимая для определения сравнительных характеристик производительности компьютерной системы.

В области вычислений для физики высоких энергий активно используется бенчмарк **HEP-SPEC2006**. Главный недостаток этого бенчмарка – это длительное время работы, что не позволяет проводить оценку производительности вычислительного ресурса динамически.



В рамках платформы DIRAC был разработан бенчмарк **DiracBenchmark2012** или **DB12**. Изначально он был разработан для прогнозирования продолжительности кампаний по генерации Монте-Карло данных эксперимента LHCb. Он длится меньше минуты и запускается каждый раз при запуске пилота.

При запуске DB12 выполняет определённую контрольную задачу и фиксирует время её выполнения. Исходя из этого времени высчитывается скорость ядра, на котором выполнялась контрольная задача. Считается, что контрольная задача требует объём вычислений в 250 DB12*сек.

Скорость ядра высчитывается по формуле:

$$CPU\ speed = \frac{250}{Time}$$

Бенчмарк DB12 откалиброван таким образом чтобы его результат соответствовал HEP-SPEC06.

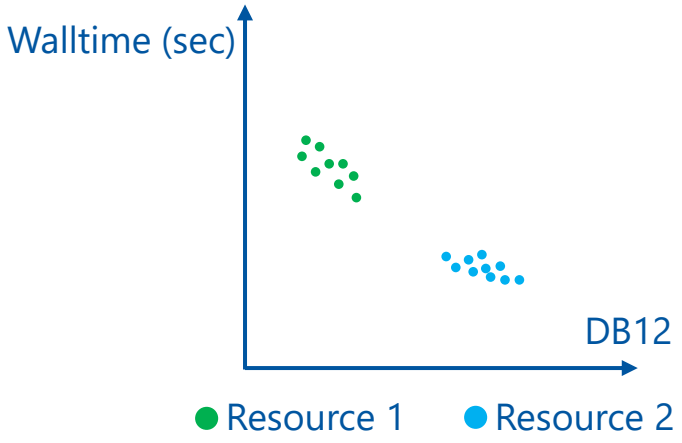
Таким образом каждой пользовательской задаче ставится в соответствие результат бенчмарка полученный пилотом перед запуском пользовательской задачи.

Система анализа выполнения больших пакетов задач

3

Для анализа процесса выполнения больших пакетов задач предлагается использовать точечный график. На этом графике:

1. Ось X соответствует значению бенчмарка DB12 с которым выполнялась пользовательская задача.
2. Ось Y соответствует полному времени выполнения задачи (walltime) в секундах.
3. Каждой завершённой задаче ставится в соответствие точка на графике в зависимости от того сколько времени выполнялась задача и какая была оценка производительности на бенчмарке DB12.
4. Цвет точек выбирается в зависимости задачи анализа. Цвет может зависеть от ресурса на котором выполнялась задача, рабочего узла, модели CPU, имени пользователя, группы пользователя, типа задачи.

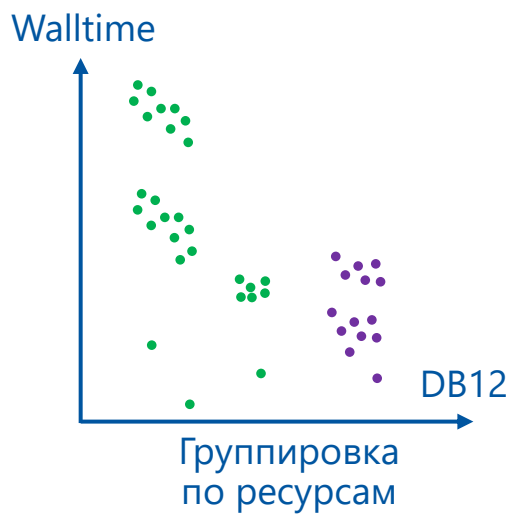


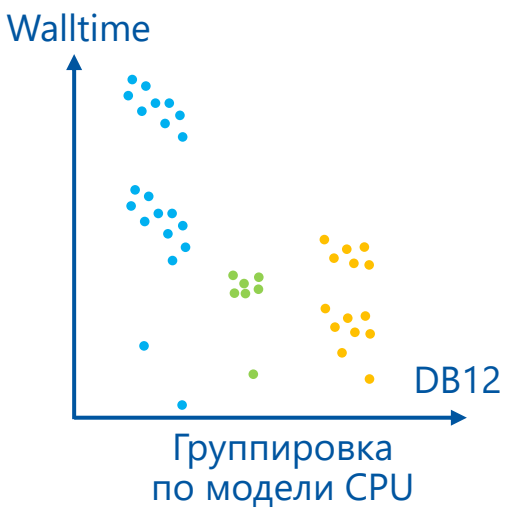
Данный точечный график даёт визуальное представление большого количества задач. Использование данной методики позволяет сопоставить время выполнения задач и производительность вычислительных ядер, на которых задачи выполнялись.

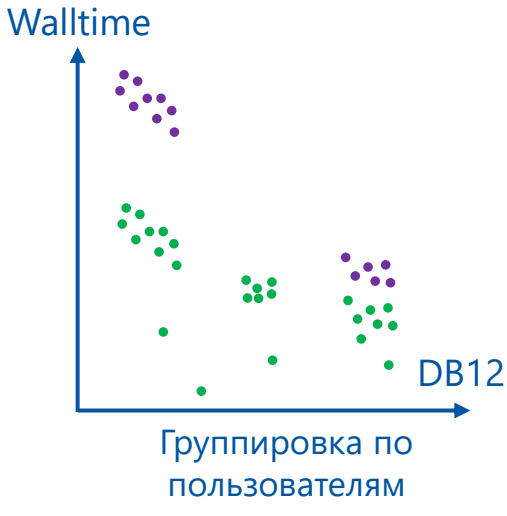
В полной мере, польза данного метода раскрывается при массовом запуске однотипных задач, которые имеют примерно одинаковое количество вычислительной работы.

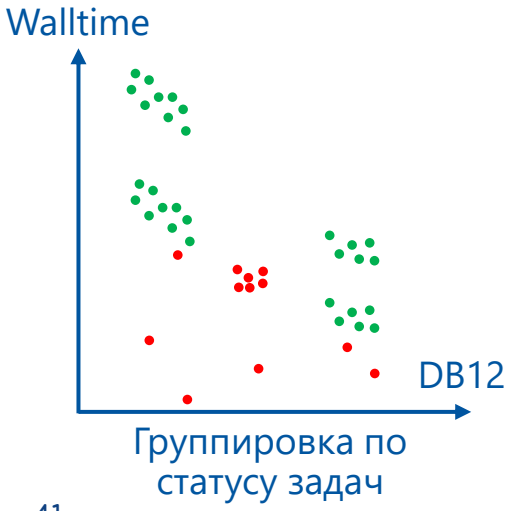
Система анализа выполнения больших пакетов задач

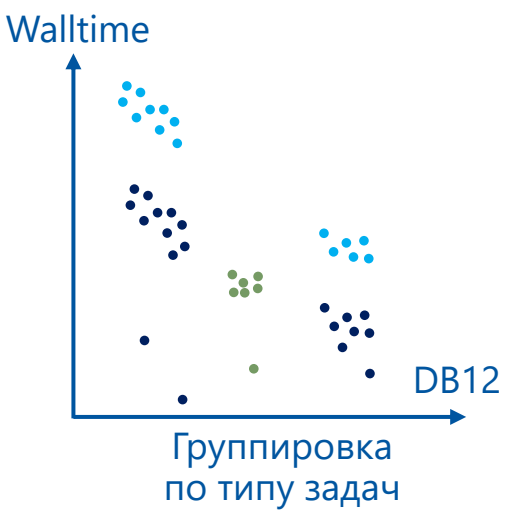
3

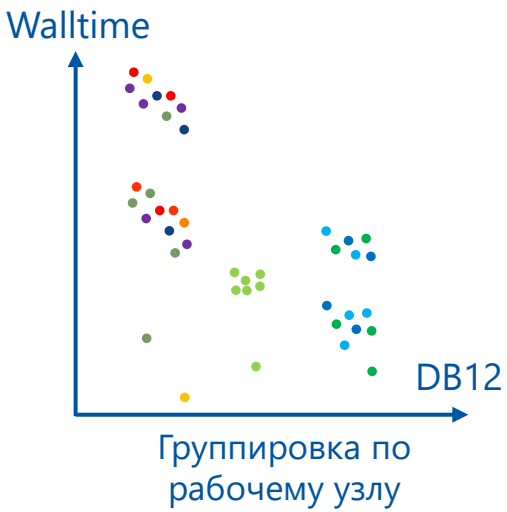










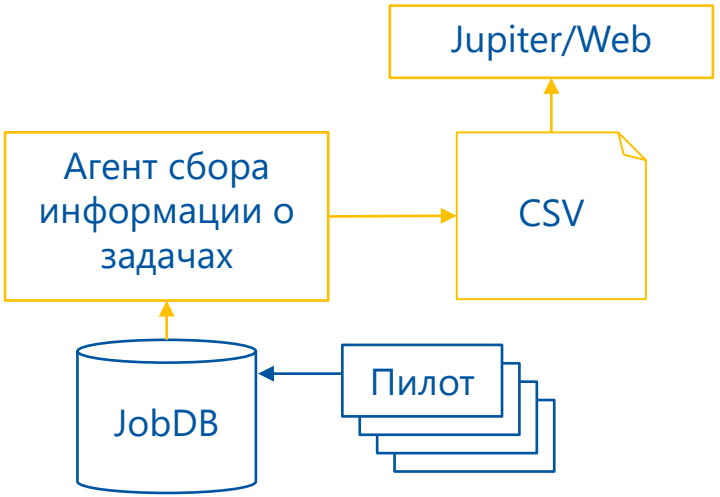


Система анализа выполнения больших пакетов задач

3

Для реализации предложенной методики необходимо разработать агент, который будет периодически собирать информацию о завершённых задачах из JobDB. Сохранять необходимо следующую информацию:

1. ID задачи
2. Время выполнения задачи (Walltime)
3. Оценку производительности DB12
4. Время начала выполнения задачи
5. Время окончания выполнения задачи
6. Имя ресурса
7. Имя рабочего узла
8. Модель CPU
9. Имя пользователя отправившего задачу
10. Имя задачи
11. Тип задачи
12. Статус с которым завершилась задача

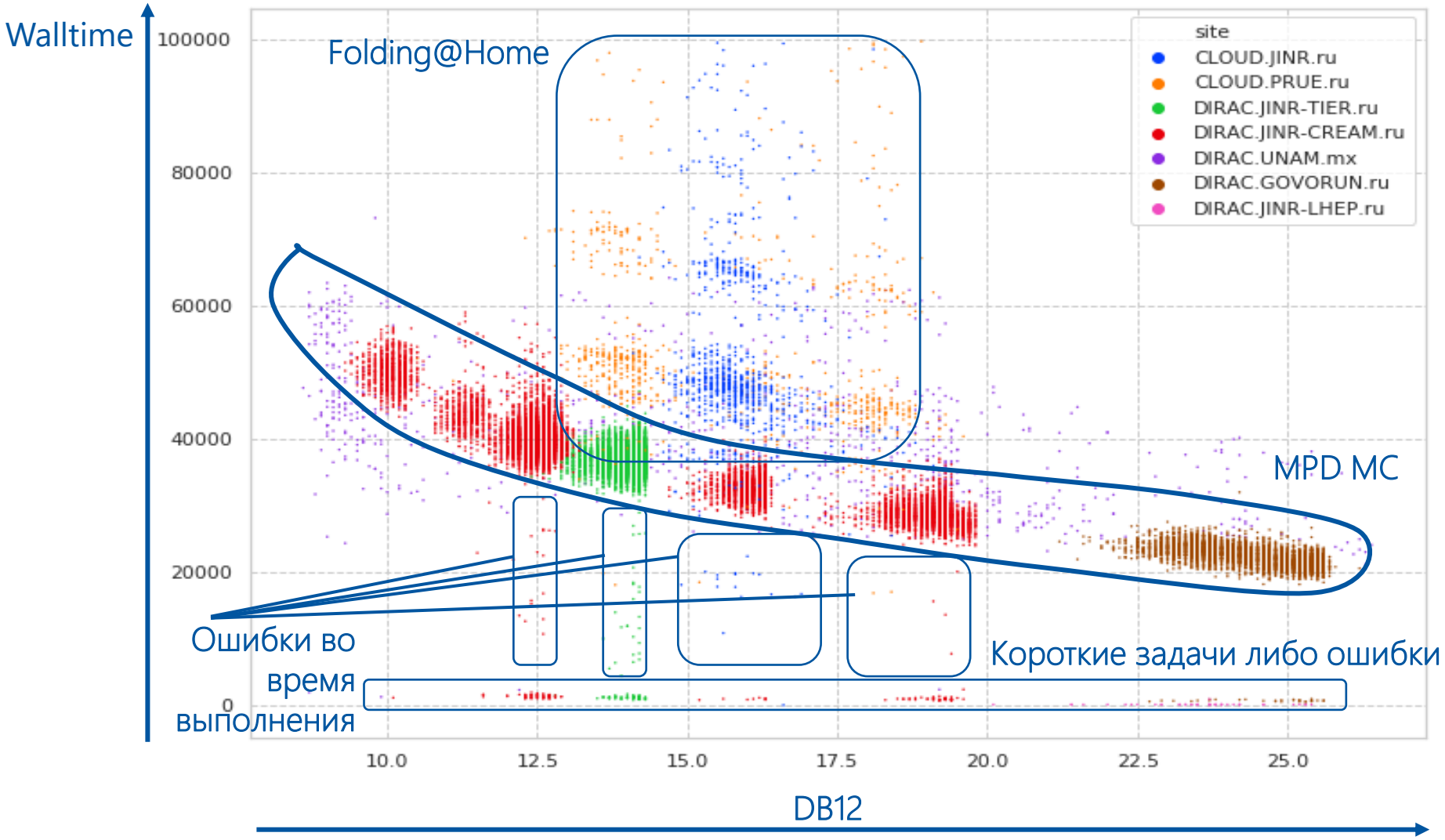


Данная архитектура была реализована. С целью упрощения установки разработанного модуля, в качестве средства хранения был выбран CSV файл.

Разработанный модуль позволил собирать данные о задачах для последующего анализа их выполнения в рамках распределённой сети.

Система анализа выполнения больших пакетов задач

③



Система анализа выполнения больших пакетов задач

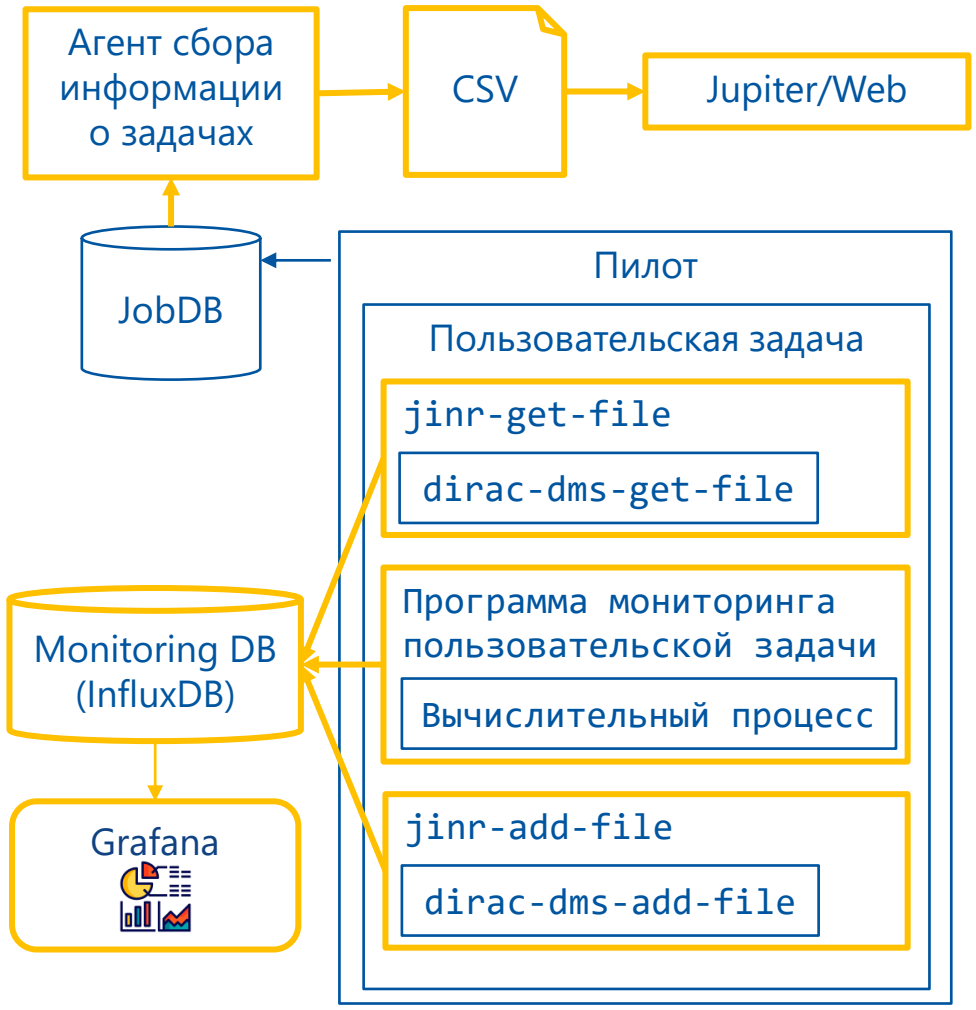
③

Использование предложенного метода позволяет:

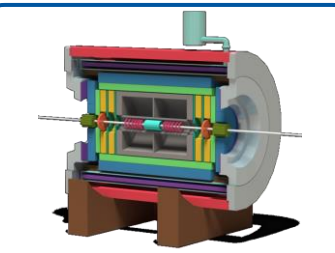
1. Получать информацию о структуре вычислительных ресурсов.
2. Получать информацию о времени выполнения задач на разных вычислительных ресурсах.
3. Получать информацию о ходе выполнения больших пакетов задач и их структуре.
4. Сравнивать между собой разные пакеты однотипных задач. К примеру использование разных версий прикладного ПО. Либо разные конфигурации вычислительных ресурсов.
5. Детектировать несоответствия между временем выполнения задач и оценками ресурсов бенчмарком DB12. Причиной может быть как особенности вычислительного ресурса, так и ошибки при расчёте бенчмарка DB12.

Мониторинг и анализ работы распределённой гетерогенной вычислительной среды

1. Мониторинг пользовательских задач позволяет получать детальную информацию о том как выполнялись конкретные задачи.
2. Мониторинг передач данных позволяет оценивать как скорость передачи в рамках отдельной задачи, так и оценивать потоки данных между вычислительными ресурсами и системами хранения
3. Разработанный и реализованный метод анализа выполнения больших пакетов задач позволяет визуально оценить ход выполнения десятков или сотен тысяч задач. Вместе с фильтрацией и группировкой данных по разным категориям использование этого метода позволяет детектировать задачи либо ресурсы, которые работают не эффективно.
4. Использование всех предложенных методов позволяет получать полную картину того, как большие пакеты задач выполняются на географически распределённых гетерогенных вычислительных ресурсах.

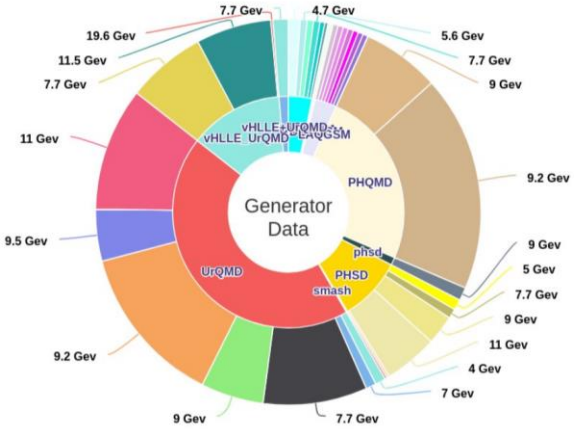


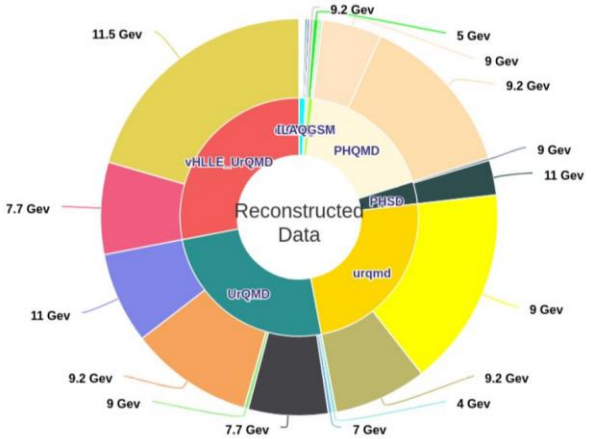
Использование построенной инфраструктуры экспериментом MPD



Декабрь 2025
Монте-Карло,
Анализ

Коллаборация MPD использует построенную инфраструктуру с 2019 года для генерации данных Монте-Карло. С 2023 года выполняются и задачи анализа.





36 Сеансов генерации

1.412 B Событий сгенерировано

568M Событий реконструировано

1.95 M Задач выполнено

2013 Общее CPU время

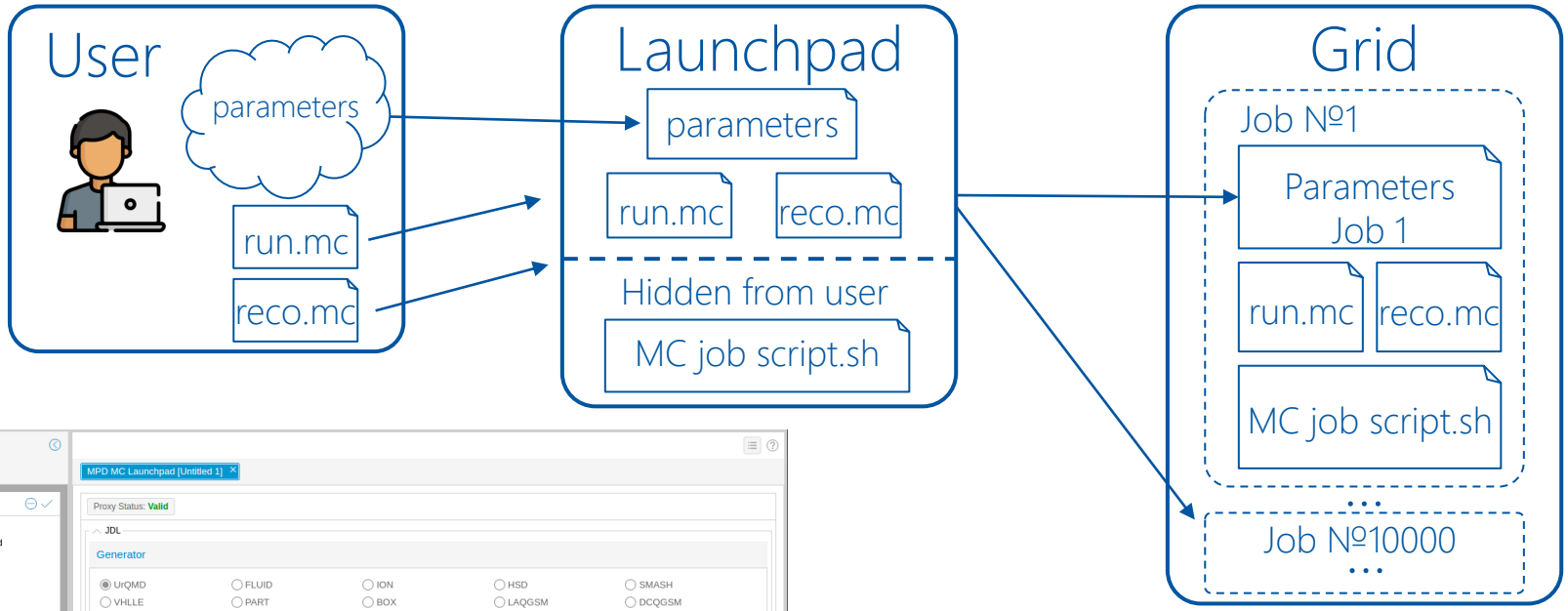
1.6 PB Данных сгенерировано

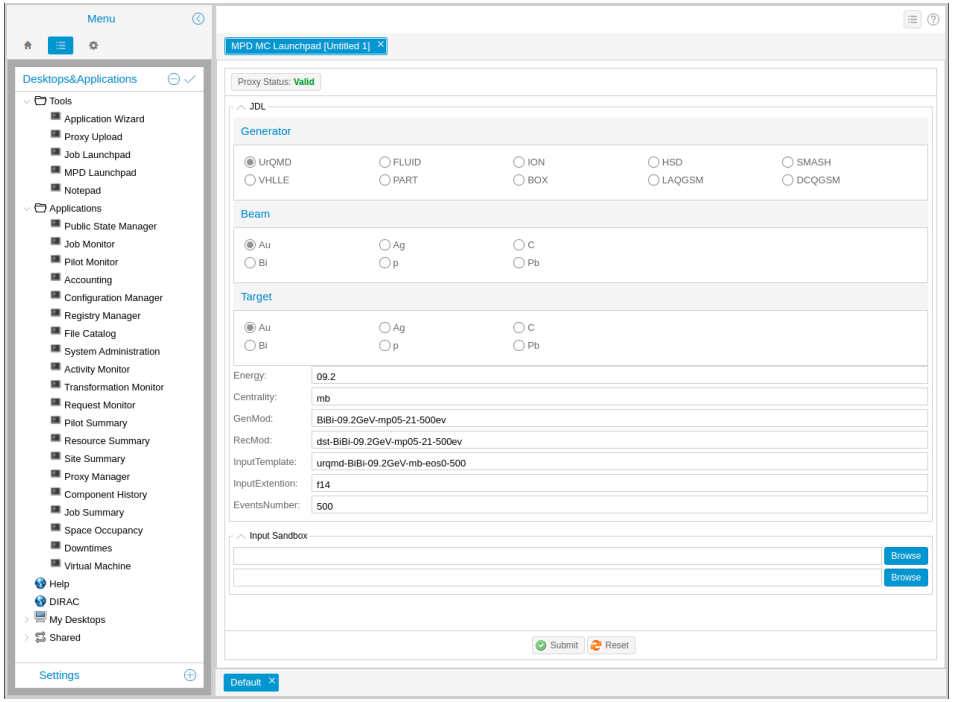
10 Сеансов анализа данных

500 M Событий проанализировано

Эксперимент MPD использует свою схему запуска задач без использования стандартной задачи и набора параметров. Вместо этого для каждой задачи формируется собственный shell скрипт в котором фиксируются имена входных и выходных файлов.

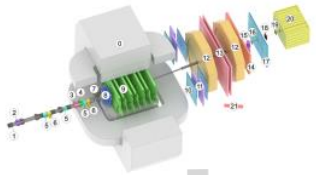
Веб-приложение автоматизированного запуска задач Монте-Карло





Для эксперимента MPD был разработан модуль DIRAC позволяющий отправлять задачи генерации Монте-Карло используя веб-приложение платформы DIRAC. В рамках разработанного модуля задаются необходимые параметры: выбор генератора, количество событий, энергия столкновений, центральность и др.

Использование построенной инфраструктуры экспериментом BM@N

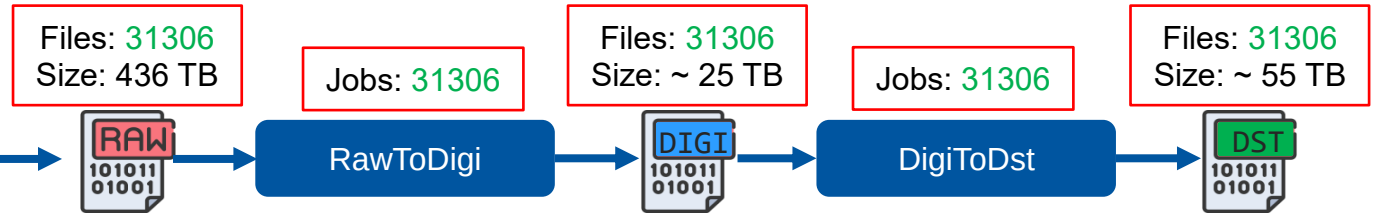


BM@N

Работает
Монте-Карло

Отбор событий для анализа

Обработка данных с детектора

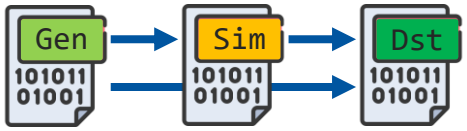


Подготовлен набор инструментов запуска следующих процессов:

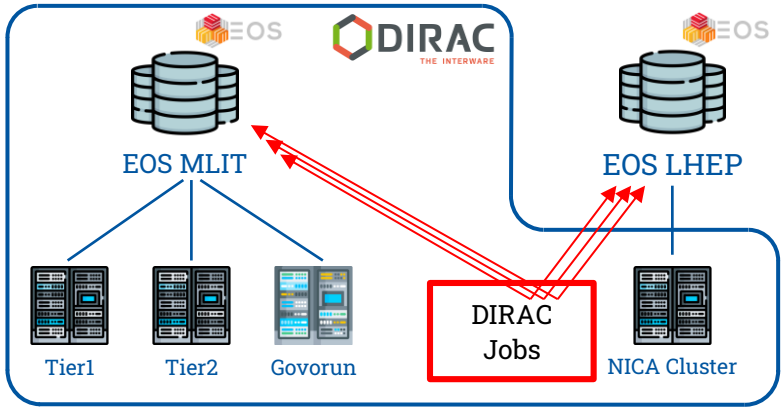
- Raw to Digi
- Digi to Dst

Аналогично разработаны инструменты для задач Монте-Карло:

- Gen to Sim
- Sim to Dst
- Gen to Dst

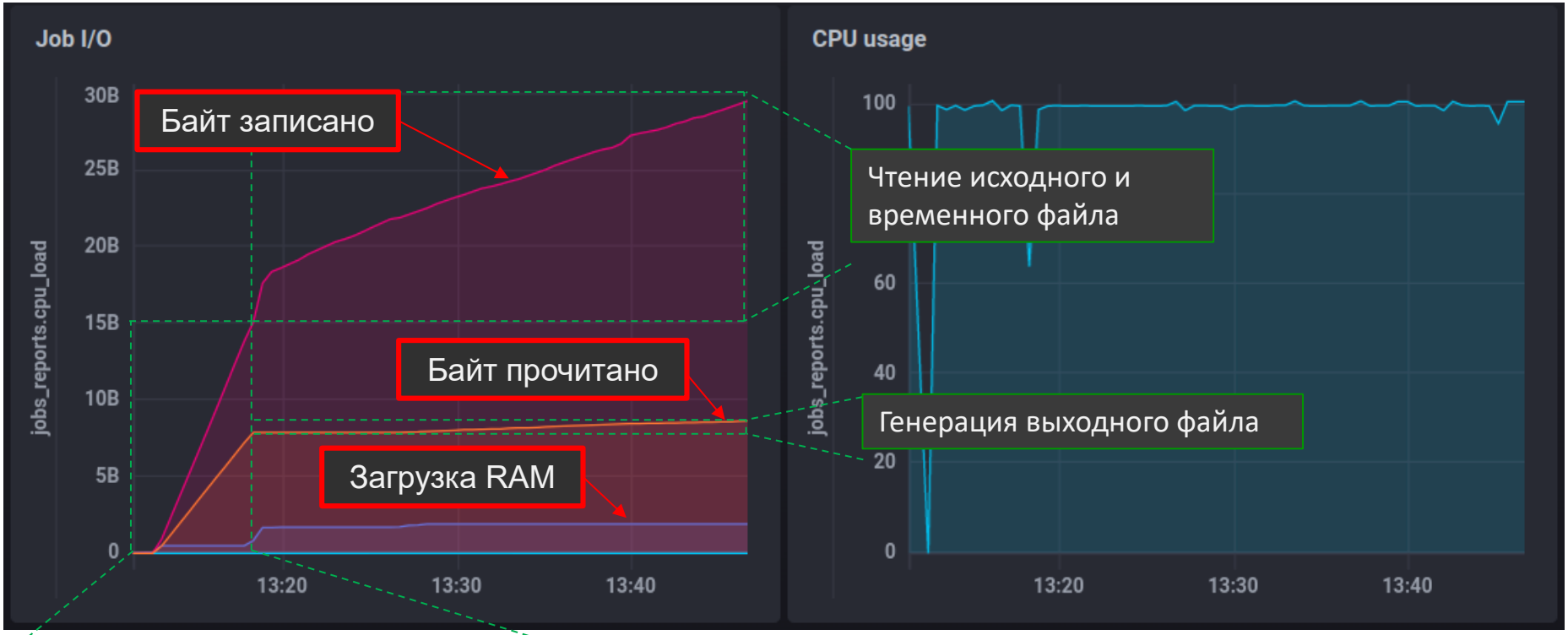


Разработан метод передачи данных между хранилищами, которые не интегрированы в систему DIRAC. Для этого используются специально разработанные задачи DIRAC.



Тест / Профилирование -> Сеанс обработки -> Анализ выполнения

Профиль задачи RawToDigi эксперимента BM@N

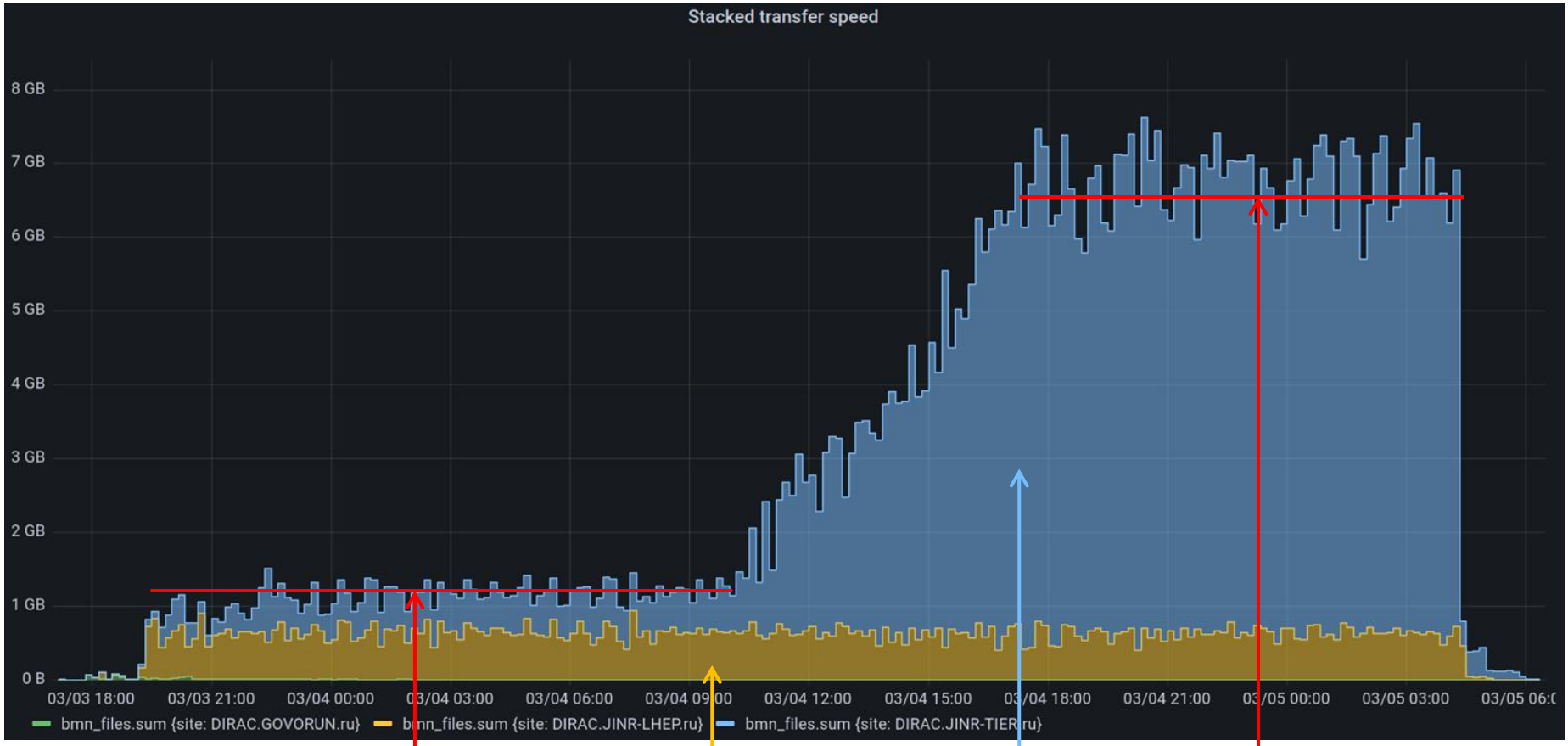


Чтение исходного файла размером 15 ГБ с параллельным созданием временного файла размером 8 ГБ

Исходный файл: 15 ГБ
Временный файл: **8 ГБ**
Выходной файл: **800 МБ**

Тест / Профилирование -> Сеанс обработки -> Анализ выполнения

Анализ загрузки сети во время сеанса обработки данных задачами RawToDigi



300 задач
4 МБ/с на задачу

NICA кластер

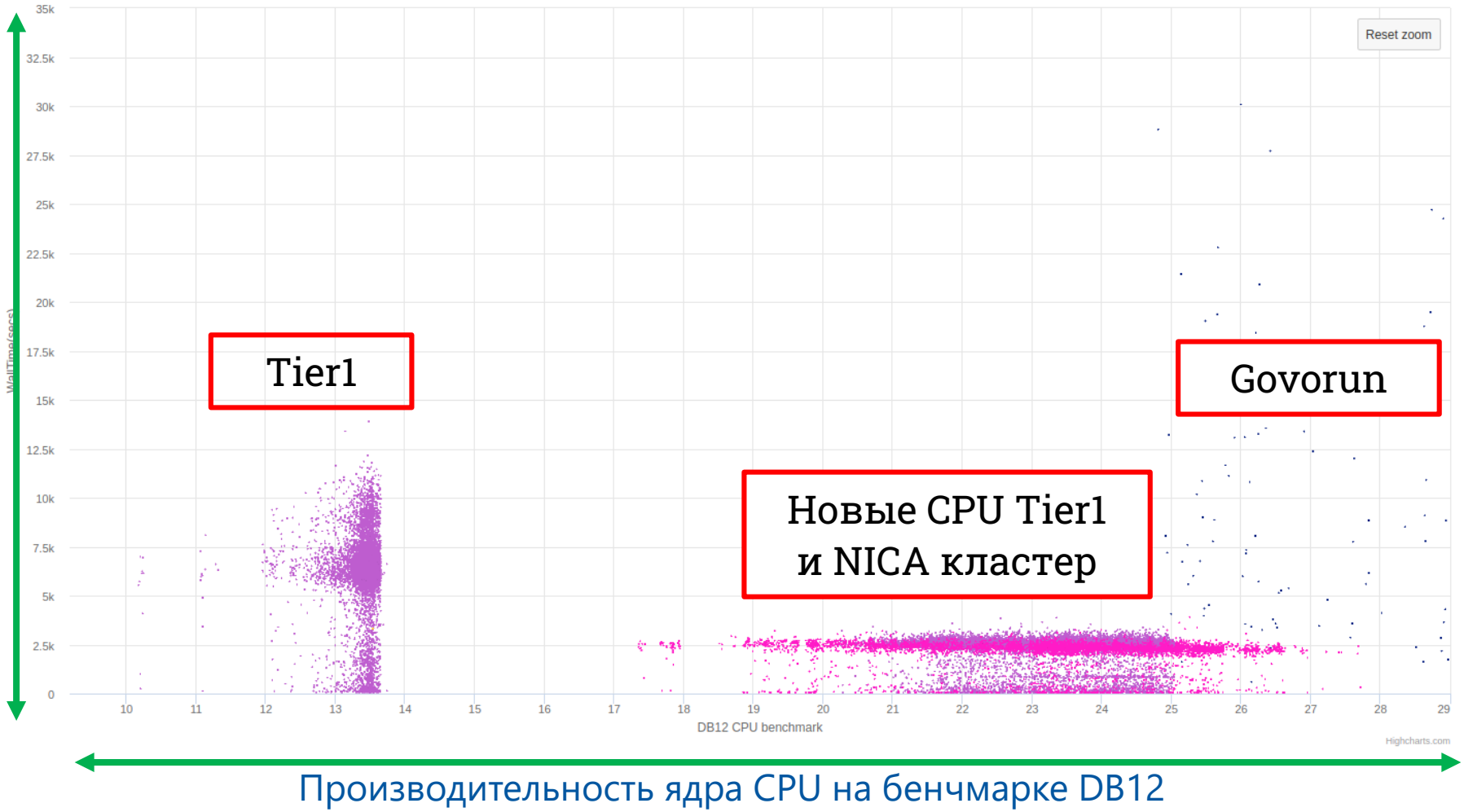
Tier1

1580 задач
4.1 МБ/с на задачу

Максимальная суммарная скорость передачи данных между EOS в ЛИТ и вычислительными ресурсами – 7.5 GB/s

Анализ выполнения задач RawToDigi

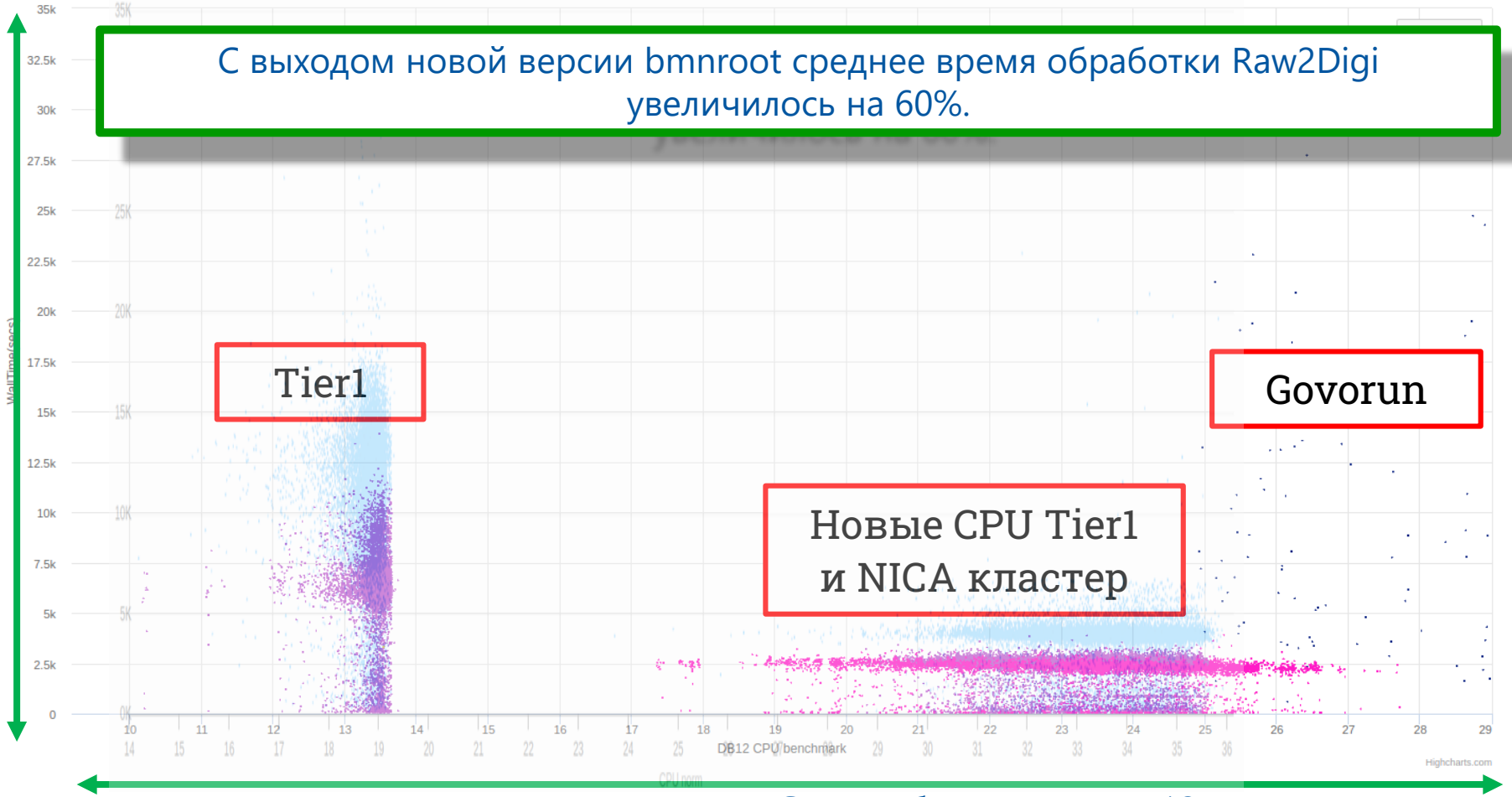
Продолжительность задачи



Производительность ядра CPU на бенчмарке DB12

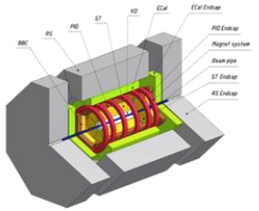
Сравнение двух сеансов обработки задач RawToDigi с использованием разных версий прикладного ПО

Продолжительность задачи



Производительность ядра CPU на бенчмарке DB12

Использование построенной инфраструктуры экспериментом SPD



SPD NICA

~ 2028
Монте-Карло,
Анализ

Коллаборация SPD использует построенную инфраструктуру с 2021 года для генерации данных Монте-Карло и анализа.

Simulation settings

$$p + p \rightarrow \pi^0 + X$$
$$\pi^0 \rightarrow \gamma\gamma$$

- SpdRoot version 4.1.5.1
- Two energies: pp @ $\sqrt{s} = 13$ GeV and pp @ $\sqrt{s} = 27$ GeV
- Particle generator: Pythia 8 (number of events: ~100M) ← using DIRAC platform

Часть слайда Штехер Катерин Диаз из доклада на SPD collaboration meeting 27.04.2023

Эксперимент SPD формирует задачи таким образом, чтобы последовательно выполнялись:

1. Генерация данных
2. Симуляция
3. Реконструкция
4. Формирования небольшого набора данных для последующего анализа на локальных вычислительных ресурсах

Защищаемые положения

1. Методика организации масштабируемой распределённой гетерогенной вычислительной инфраструктуры на основе инструментария DIRAC, позволяющая интегрировать широкий спектр вычислительных ресурсов и ресурсов хранения данных.

2. Распределённая гетерогенная вычислительная инфраструктура, введенная в промышленную эксплуатацию и включающая управление задачами, данными, автоматизацию рабочих процессов и централизованный учет потребляемых ресурсов.

3. Разработанные методы мониторинга и анализа производительности распределенных гетерогенных вычислительных ресурсов.

Научная новизна

1. Впервые в ОИЯИ построена распределённая гетерогенная вычислительная инфраструктура, включающая вычислительные ресурсы:

- Суперкомпьютер «Говорун»
- Кластеры Tier1 и Tier2
- NICA кластер
- DAQ Data Ceter
- Облачные инфраструктуры ОИЯИ и стран-участниц ОИЯИ
- Вычислительные ресурсы членов коллабораций экспериментов: UNAM, МИФИ

и ресурсы хранения данных:

- Дисковое хранилище EOS
- Ленточное хранилище dCache/Enstore
- Ленточное хранилище CTA;

Произведена адаптация рабочих процессов массового моделирования данных эксперимента MPD и BM@N для запуска задач и сохранения данных с использованием платформы DIRAC.

2. Разработан модуль который позволяет интегрировать в созданную инфраструктуру облачные ресурсы, работающие на базе ПО OpenNebula. Разработанный модуль был включен в официальный пакет ПО DIRAC.

Научная новизна

3. Предложены и реализованы методы мониторинга задач и передач данных. Разработан новый метод анализа выполнения больших пакетов задач в географически распределенных гетерогенных вычислительных ресурсов.

4. Предложен и реализован метод обработки данных, полученных в эксперименте BM@N, с использованием всего созданного инструментария. Этот подход позволяет произвести оценку требований к ресурсам, выполнить запуск задач, сохранить информацию о ходе выполнения задач и проанализировать процесс выполнения задач в рамках построенной вычислительной инфраструктуры.

Научно-практическая значимость

1. Построенная географически распределённая гетерогенная вычислительная инфраструктура с 2019 года используется для реализации программы сеансов массового моделирования данных эксперимента **MPD**. **SPD** проводит моделирование и анализ данных для эксперимента. Для обработки данных 8-го сеанса эксперимента **BM@N** была впервые разработана и реализована методика использования географически распределённой гетерогенной вычислительной инфраструктуры.

2. Создание модуля интеграции облачных ресурсов, работающих на базе ПО OpenNebula, позволило объединение облачных ресурсов ОИЯИ и стран-участниц для проведения вычислений. Эксперимент JUNO использовал этот модуль для отправки задач в облако ОИЯИ до момента перехода на систему HTCondor.

3. Созданные системы мониторинга позволяют своевременно получать информацию о ходе обработки данных, анализировать использование ресурсов вычислительных ресурсов.

4. Разработанный метод анализа выполнения больших пакетов задач позволяет анализировать производительность вычислительных ресурсов. Данный метод был адаптирован для инфраструктуры экспериментов BES-III, JUNO, CEPC в ИФВЭ в Пекине.

Список публикаций

1. Gertsenberger, K.V., Pelevanyuk, I.S. **BM@N Run 8 Data Processing on a Distributed Infrastructure with DIRAC**. Phys. Part. Nuclei Lett. 21, 778–781 (2024). https://doi.org/10.1134/S1547477124701334

2. Gertsenberger, K., Pelevanyuk, I., Klimai, P. et al. **Computing Software Architecture for the BM@N Experiment**. Phys. Part. Nuclei 55, 338–342 (2024). https://doi.org/10.1134/S1063779624030407

3. Campis, D., Ilina, A. & Pelevanyuk, I. **System for Analysis of the Performance of Scientific Jobs in Distributed Systems**. Phys. Part. Nuclei 55, 401–403 (2024). https://doi.org/10.1134/S1063779624030262

4. V. Korenkov, I. Pelevanyuk, A. Tsaregorodtsev - **DIRAC at JINR as a general-purpose system for massive computations**, J. Phys.: Conf. Ser. 2438, 012-029 (2023). https://doi.org/10.1088/1742-6596/2438/1/012029

5. Pelevanyuk, I.S., Campis, D. **Simulation of Job Execution in Distributed Heterogeneous Computing Infrastructures**. Phys. Part. Nuclei Lett. 20, 1276–1278 (2023). https://doi.org/10.1134/S1547477123050618

6. Pelevanyuk, I., Tsaregorodtsev, A. **DIRAC Interware as a Service for High-Thoughput Computing in JINR**. Phys. Part. Nuclei Lett. 19, 580–582 (2022). https://doi.org/10.1134/S1547477122050338

7. Nikolay Kutovskiy, Valery Mitsyn, Andrey Moshkin, Igor Pelevanyuk, Dmitriy Podgayny, Oleg Rogachevsky, Boris Shchinov, Vladimir Trofimov, Andrei Tsaregorodtsev **Integration of geographically distributed heterogeneous resources for the MPD experiment with the DIRAC Interware**, Phys. Part. Nuclei 52, 835–841 (2021) . https://doi.org/10.1134/S1063779621040419

8. Igor Pelevanyuk, "**Performance evaluation of computing resources with DIRAC interware**", AIP Conference Proceedings 2377, 040006 (2021) https://doi.org/10.1063/5.0064778

Список публикаций

9. Andrey Moshkin, Igor Pelevanyuk, Oleg Rogachevskiy - **Design and development of application software for MPD distributed computing infrastructure**; CEUR Workshop Proceedings, Vol-3041, pp. 321-325 (2021) http://doi.org/10.54546/MLIT.2021.76.62.001
10. Nikolay Kutovskiy, Igor Pelevanyuk, Dmitriy Zaborov – **Using distributed clouds for scientific computing**; CEUR Workshop Proceedings, Vol-3041, pp. 196-201 (2021) http://doi.org/10.54546/MLIT.2021.78.51.001
11. Korenkov, V., Pelevanyuk, I., Tsaregorodtsev, A. (2020). **Integration of the JINR Hybrid Computing Resources with the DIRAC Interware for Data Intensive Applications**. DAMDID/RCDL 2019. Communications in Computer and Information Science, vol 1223. Springer, Cham. https://doi.org/10.1007/978-3-030-51913-1_3
12. J Chen, I Pelevanyuk, Y Sun, A Zhemchugov, T Yan, X H Zhao and X M Zhang **Resources monitoring and automatic management system for multi-VO distributed computing system**, J. Phys.: Conf. Ser. 898, 092019 (2017) https://doi.org/10.1088/1742-6596/898/9/092019
13. **Multi-VO support in IHEP's distributed computing environment** - T Yan, B Suo, X H Zhao, X M Zhang, Z T Ma, X F Yan, T Lin, Z Y Deng, W D Li, S Belov, I Pelevanyuk, A Zhemchugov and H Cai, Journal of Physics: Conference Series 664 (2015) 062068, https://doi.org/10.1088/1742-6596/664/6/062068
14. Belov, S.D., Kadochnikov, I.S., Korenkov, V.V., Kutovskiy, N.A., Pelevanyuk, I.S., Semenov, R.N. & Zrelov, P.V. "**Integration of the parallel resources to the distributed cloud infrastructures for large scale projects**", CEUR Workshop Proceedings, Vol-2772, pp. 58-64 (2020)
15. Balashov, N.A. and Kuchumov, R.I. and Kutovskiy, N.A. and Pelevanyuk, I.S. and Petrunin, V.N. and Tsaregorodtsev, A.Yu. **Cloud integration within the DIRAC Interware**, CEUR Workshop Proceedings, Vol-2507, pp. 256-260 (2019)
16. Zrelov P., Korenkov V., Pelevanyuk I., Tsaregorodtsev A., **Accessing distributed computing resources by scientific communities using DIRAC service**, CEUR Workshop Proceedings, Vol-1752, pp. 110-115 (2016)

Спасибо за внимание

Пелеванюк Игорь Станиславович

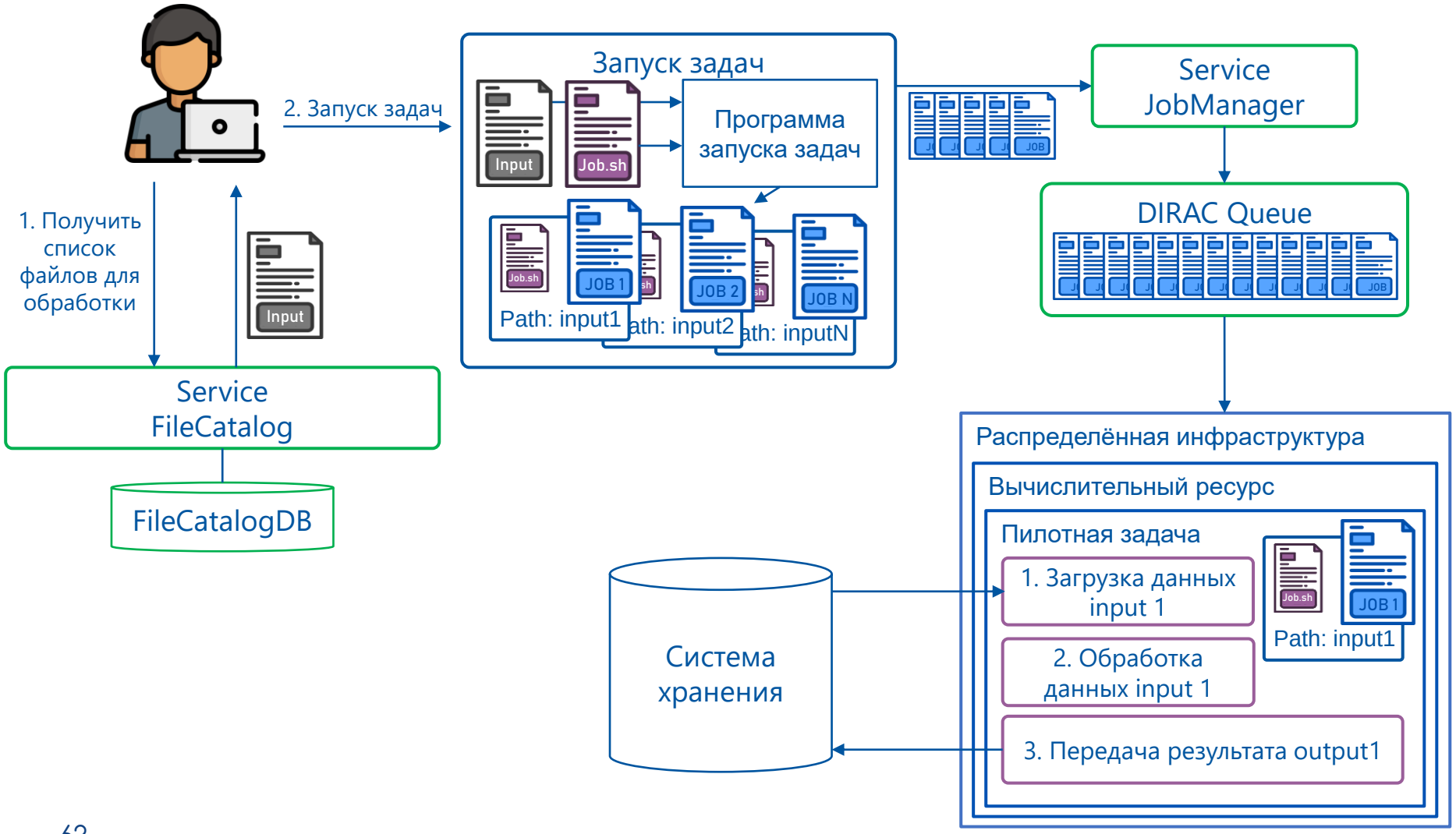
Научная специальность: 2.3.5 — Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей по техническим наукам.

соискание ученой степени кандидата технических наук

Научный руководитель:
доктор технических наук
Кореньков Владимир Васильевич

Резерв слайдов

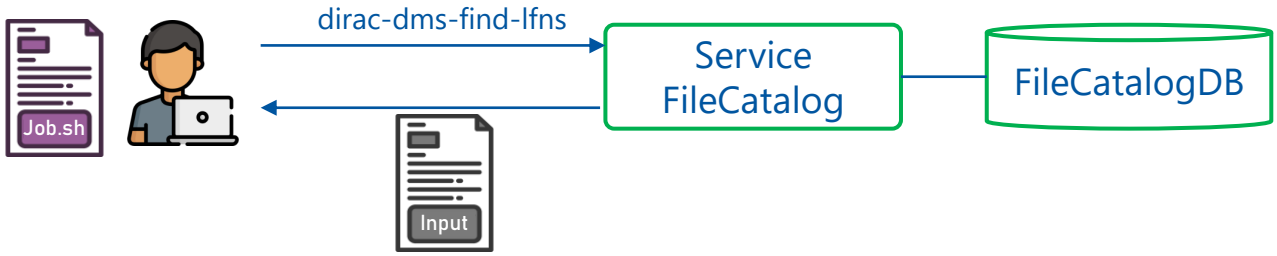
Глава 4. Апробация построенной распределенной гетерогенной вычислительной среды и инструментов для решения задач экспериментов на комплексе NICA



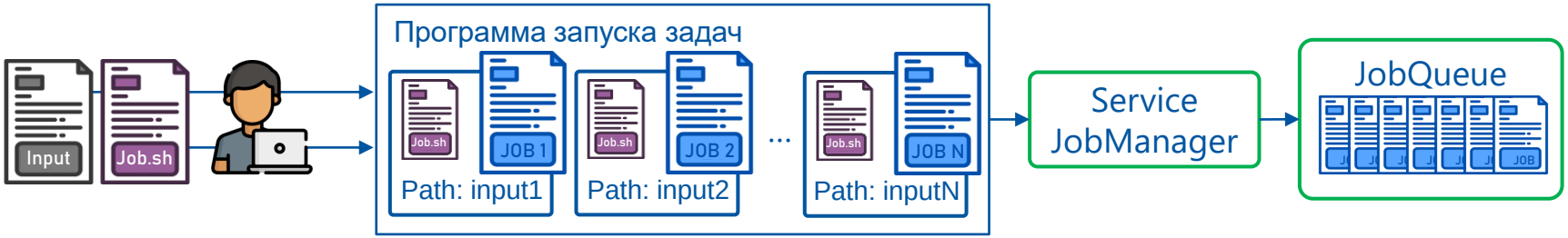
Запуск больших пакетов задач

Запуск больших пакетов задач предложено проводить в два этапа:

1. Получение списка входных файлов, которым требуется обработка.



2. Автоматизированный запуск задач.

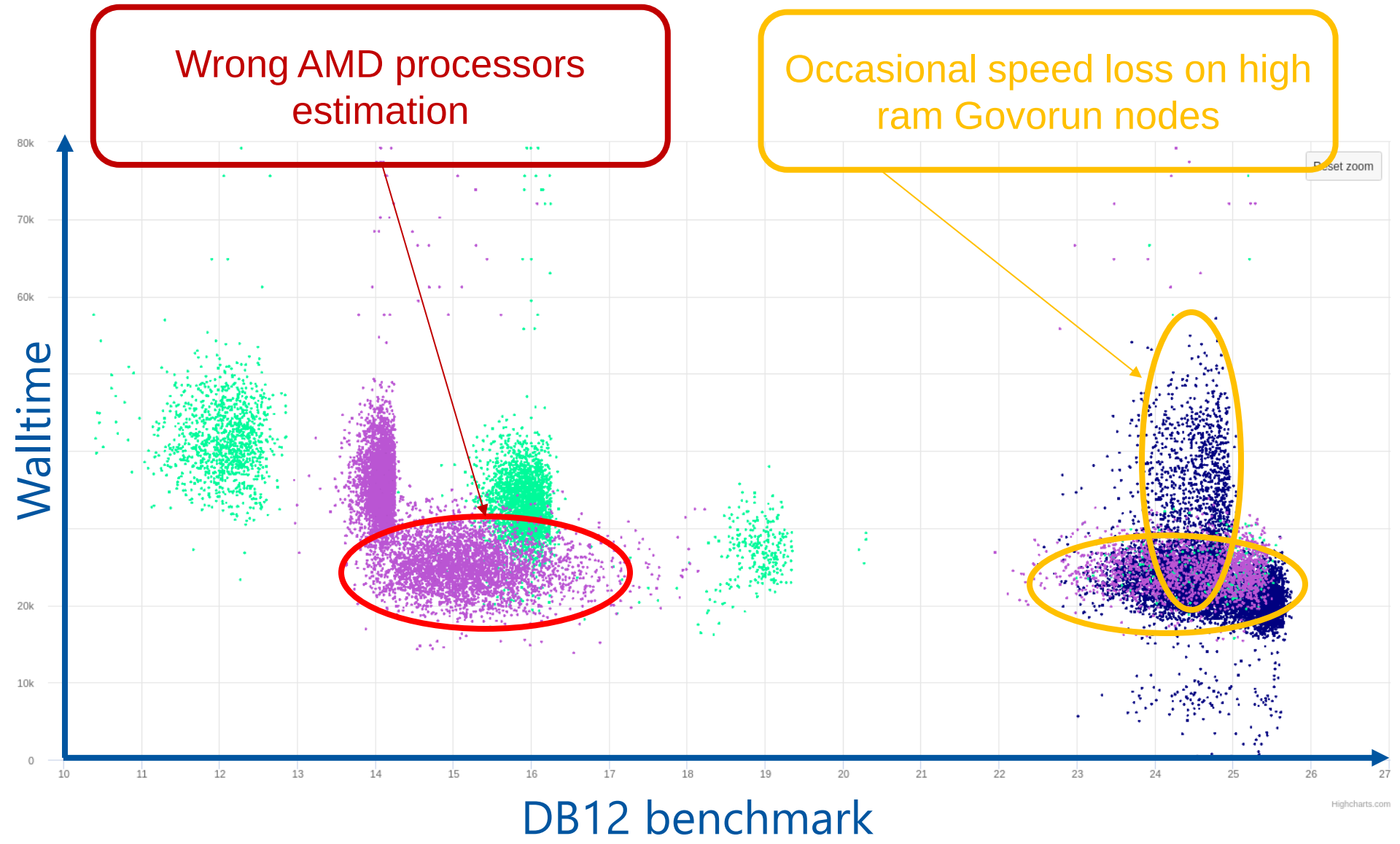


Программа запуска задач для каждого из входных файлов полученных на первом этапе создаёт соответствующие задачи и отправляет их в очередь задач.

Пример задачи:

```
1. JobName: Имя_задачи_1
2. Executable: /bin/sh
3. Arguments: job.sh input1
4. InputSandbox: job.sh
5. OutputSandbox: std.out, std.err
```

Discoveries



History of DIRAC at JINR

2013 – Development of monitoring system for BES-III installation. First tries to setup and configure DIRAC infrastructure.

2017 – DIRAC Interware installed; basic configuration done. Used for educational purposes. dCache storage integrated, Tier2 integrated.
2018 – HybriLIT integrated. JINR cloud integrated using OCCI protocol. Tests of full cycle of Monte-Carlo for BM@N were performed.
2019 – Clouds of JINR Member-States integrated by module developed in JINR. MPD starts using DIRAC for massive Monte-Carlo production. Tier1, Govorun and EOS integrated in DIRAC.
2020 – Folding@Home jobs submitted to clouds via DIRAC. Baikal-GVD jobs submitted to JINR and PRUE clouds.
2021 – First tests for SPD Monte-Carlo successfully done. First million jobs done!
2022– Total walltime exceeds 1000 years. DIRAC in JINR updated to use Python 3.
2023 – Full BM@N data processing for Run 8

List of participants

DIRAC: Igor Pelevanyk, Andrey Tsaregorodtzev, Anna Ilina
Baikal-GVD: Dmitry Zaborov
BM@N: Konstantin Gertsenberger
MPD: Oleg Rogachevskiy, Andrey Moshkin
SPD: Alexey Zhemchugov, Katherin Shtejer, Elena Zemlyanichkina, Artem Ivanov
Responsible for resources:
Govorun: Oksana Streltsova, Dmitry Podgainy, Dmitry Belyakov, Aleksandr Kokorev, Maxim Zuev
Tier-1,Tier-2, EOS: Valery Mitsyn
NICA cluster: Ivan Slepov
DDC cluster: Ilia Slepnev
Cloud: Nikolay Kutovskiy, Nikita Balashov
dCache: Vladimir Trofimov


JINR
MLiT