МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
(НИЯУ МИФИ)

ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ И ТЕХНОЛОГИЙ
КАФЕДРА №40 «ФИЗИКА ЭЛЕМЕНТАРНЫХ ЧАСТИЦ»

УДК 539.12, 004.42

На правах рукописи

МОНАКОВ НИКИТА ГЛЕБОВИЧ

СИСТЕМА УПРАВЛЕНИЯ ПРОЦЕССАМИ ОБРАБОТКИ В ГЕОГРАФИЧЕСКИ РАСПРЕДЕЛЕННОЙ СРЕДЕ ОБРАБОТКИ ДАННЫХ ЭКСПЕРИМЕНТА SPD

Направления подготовки:
09.04.04 «Программная инженерия»,
14.04.02 «Ядерные физика и технологии»
Диссертация на соискание степени магистра

Научный руководитель,
к.ф.-м.н., доц. _____ Е. Ю. Солдатов

Научный консультант,
к.т.н., с.н.с. _____ А. Ш. Петросян

Москва 2025

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

СИСТЕМА УПРАВЛЕНИЯ ПРОЦЕССАМИ ОБРАБОТКИ В ГЕОГРАФИЧЕСКИ РАСПРЕДЕЛЕННОЙ СРЕДЕ ОБРАБОТКИ ДАННЫХ ЭКСПЕРИМЕНТА SPD

Студент	_____ Н. Г. Монаков
Научный руководитель, к.ф.-м.н., доц.	_____ Е. Ю. Солдатов
Научный консультант, к.т.н., с.н.с.	_____ А. Ш. Петросян
Рецензент, к.ф.-м.н., с.н.с.	_____ Ф. В. Прокошин
Рецензент, к.т.н., с.н.с.	_____ Д. А. Олейник
Секретарь ГЭК, к.ф.-м.н., доц.	_____ Е. Ю. Солдатов
Зав. каф. №40, д.ф.-м.н., проф.	_____ М. Д. Скорохватов
Рук. учеб. прог., к.ф.-м.н., доц.	_____ Е. Ю. Солдатов

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Любой физический эксперимент ставит перед собой амбициозные теоретические и технические задачи, но работа над экспериментами в рамках мегасайнс проектов, таких как эксперимент SPD (Spin Physics Detector) на проекте NICA (Nuclotron based Ion Collider fAcility), поднимает и последующие этапы сбора, хранения и обработки полученных данных на самый высокий технический уровень. Системы и подсистемы, управляющие процессами на этих этапах, должны быть реализованы, кроме прочего, с учетом возможности изменения как ресурсов самих систем, так и объёмов поставляемых данных с физической части эксперимента.

Целью проводимой работы является создание и поддержка подобной системы управления процессами обработки данных для эксперимента SPD. Актуальность работы обусловлена нуждами коллаборации в разрабатываемом функционале ввиду больших трудностей при полностью ручном управлении обработкой данных в масштабах эксперимента физики высоких энергий. В ходе научно-исследовательской работы планируется изучить существующий мировой опыт построения систем, ответственных за управление процессами обработки данных, в том числе географически распределенных, изучить используемые в них технические средства, провести их анализ, выбрать наиболее подходящие. После чего встроиться в создание и последующее поддержание соответствующей системы для эксперимента SPD.

1 ФИЗИЧЕСКАЯ УСТАНОВКА

1.1 КОМПЛЕКС NICA

Цель проекта «NICA» [1] заключается в создании на территории Российской Федерации экспериментальной базы мирового уровня для проведения фундаментальных исследований по ряду ключевых направлений современной физики высоких энергий, а также прикладных исследований. Кроме того, проект предусматривает участие учёных из научных организаций стран-участниц и других государств, присоединившихся к проекту, что также делает эксперимент крупным и важным международным проектом в сфере теоретической физики.

Системы комплекса NICA

В состав комплекса входят следующие основные системы:

1) Ускорительный блок, включающий:
   - сверхпроводящий синхротрон «Нуклотрон» с системами вывода и перенаправления пучков;
   - инжекционный комплекс (источники ионизированных и поляризованных частиц, линейные ускорители);
   - бустер в виде сверхпроводящего синхротрона;
   - сверхпроводящий коллайдер тяжёлых ионов и поляризованных частиц.
2) Экспериментальные установки:
   - MPD (MultiPurpose Detector) — установка для исследования плотного барионного вещества в сталкивающихся пучках;
   - BM@N (Baryonic Matter at Nuclotron) — установка для исследования плотного барионного вещества с использованием выведенных из Нуклотрона пучков;
   - SPD (Spin Physics Detector) — установка для изучения спиновой структуры нуклона в опытах на поляризованных пучках.
3) Инновационный блок, включающий экспериментальные зоны, каналы и стенды для:
   - прикладных и инновационных исследований;

- разработки микроэлектроники;
- решения задач биомедицины и материаловедения.

4) Вычислительно-информационный блок, включающий:
   - распределённый информационно-вычислительный комплекс для моделирования, обработки, анализа и хранения накопленных данных;
   - сетевую инфраструктуру;
   - набор информационных сервисов.

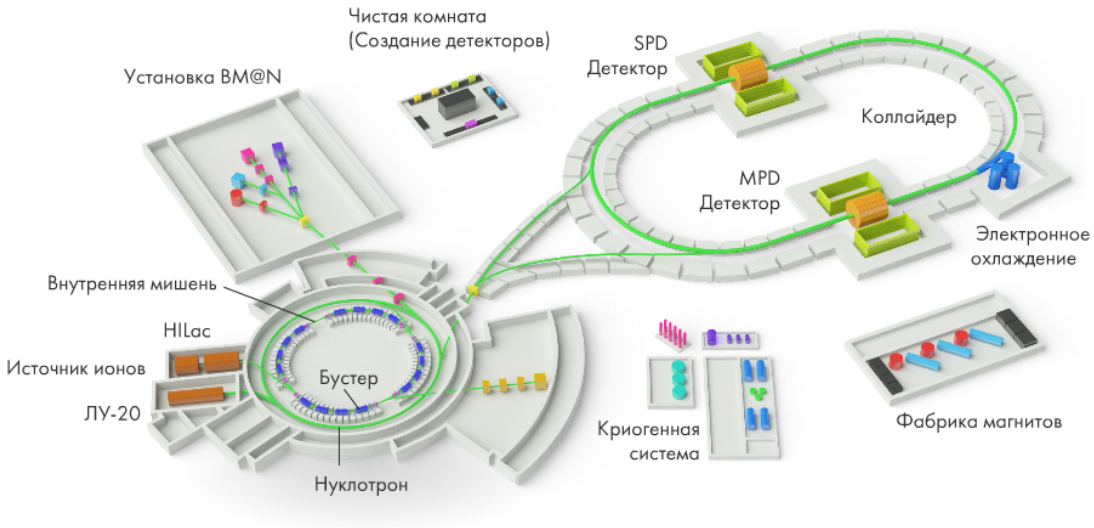


Рисунок 1 — Комплекс NICA

Для проведения фундаментальных исследований на комплексе NICA были налажены внутрироссийские и международные научные взаимодействия, объединяющие учёных, инженеров и специалистов из стран-участниц проекта. В инновационной и прикладной деятельности задействованы также коллективы ОИЯИ, включающие специалистов из разных государств, что позволяет решать широкий спектр задач.

Говоря о фундаментальных исследованиях, коллаборация ставит перед собой цели, охватывающие широкий спектр явлений, проявляющихся в реакциях с тяжёлыми ионами, поляризованными адронами и лёгкими ядрами и связанных со структурой вещества, участвующего в сильных взаимодействиях.

Основные направления научно-исследовательской программы

1) **Поиск и экспериментальное исследование фазовых переходов и критических явлений в сильно взаимодействующей ядерной**

материи при экстремальных барионных плотностях. [2]
Такая материя существовала лишь на ранних стадиях эволюции Вселенной, когда плотность и температура вещества были экстремально высоки, при невысоких же температурах она может возникать внутри ядер или нейтронных звёзд. Расчёты квантовой хромодинамики на решетках (КХДР) предсказывают фазовые переходы, связанные с деконфайнментом и восстановлением хиральной симметрии при достаточной плотности энергии, что приводит к формированию кварк–глюонной плазмы. Однако современные КХДР-методы не охватывают область высоких барионных плотностей, характерную для столкновений тяжёлых ионов на комплексе NICA, что, несомненно, повышает значимость получения ожидаемых экспериментальных данных. Энергетический диапазон комплекса позволяет изучать как адронную, так и кварк–глюонную фазы, и рассматривая температуру и барионную плотность, нормированные к «нормальным» условиям, в качестве ключевых наблюдаемых характеристик, косвенно указывающих на статистические закономерности в процессах столкновений.
Кроме того, важно не только зарегистрировать сам фазовый переход, но и установить его природу: будет ли это непрерывный переход или фазовый переход первого рода с критической конечной точкой. До настоящего времени ни первое, ни второе не наблюдалось экспериментально, что делает поиск этого перехода одной из приоритетных задач программы. Поскольку теоретические предсказания в области конечных барионных плотностей пока ограничены, эксперименты начнутся с изучения диагностических наблюдаемых, уже исследованных на ускорителях RHIC и SPS: выходов частиц и их спектров, флуктуаций поперечного импульса, а также комбинированных распределений при различных энергиях и типах пучков. Будут исследоваться фемтоскопические корреляции, направленные, эллиптические и более высокие гармоники для разных адронов, лептонов и фотонов, а также асимметрии по заряду и спину.

2) **Экспериментальное изучение спиновой структуры нуклона и лёгких ядер. [3]**
Современное описание спиновой структуры адронов строится через набор распределённых функций, зависящих от спина и поперечного импульса, при этом особую роль играют корреляции между различными компонен-

тами поперечного импульса. В программе комплекса будет исследоваться полный набор TMD-функций (transverse momentum–dependent) ведущего порядка с помощью жёстких зондов: дилептонов в Дрелл-Яновских процесах, кварквониев, прямых фотонов, адронов с большим поперечным импульсом. Кроме того, планируется изучение интегрированных по поперечному импульсу партонных распределений и кварк-глюонных корреляторов. Особое внимание будет уделено тензорным распределениям поляризованного дейтрона – уникальной возможности комплекса.

3) **Исследование поляризационных эффектов в столкновениях тяжёлых ионов и в системах с несколькими нуклонами.**
Поляризация гиперонов является чувствительным индикатором динамики адронной и кварк–глюонной среды, включая такие характеристики, как гидродинамическая завихренность и спиральность. Программа включает изучение поляризации гиперонов в различных кинематических областях в зависимости от типа ядер, энергии и центральности столкновения. Планируется также сравнение поляризационных эффектов в реакциях с адронами, лёгкими и тяжёлыми ядрами, а также систематическое исследование тензорной поляризации векторных мезонов и дилептонов.
4) **Изучение динамики реакций и модификаций свойств адронов в ядерной среде. [4] [5] [6]**
Теоретические исследования указывают на возможность частичного восстановления хиральной симметрии, что приводит к изменению спектральных функций адронов в плотной среде. Наиболее чувствительными зондами таких модификаций для векторных мезонов являются дилептоны: их угловые распределения и тензорная поляризация отражают состояние среды в момент взаимодействия и не искажаются сильными взаимодействиями на более поздних стадиях. В настоящее время отсутствуют данные по дилептонам с инвариантной массой в диапазоне нескольких ГэВ; программа NICA устранит этот пробел, что особенно важно в связи с достижением максимальной плотности адронов именно в этой области.
5) **Исследование структуры ядер на межнуклонных расстояниях, порогового образования странных гиперонов и поиски гиперядер в опытах с фиксированной мишенью и пучками, извлечёнными из Нуклотрона.**

Флуктуации на коротких расстояниях, связанные с концепциями флуктонов и кумулятивных процессов, могут быть изучены в обратной кинематике при рассеянии ядерных пучков на водородной мишени, что позволяет детектировать все частицы в конечном состоянии. Рассеяние поляризованных дейтронов даст возможность прямой проверки тензорной природы флуктуаций. Усиленное образование странности в столкновениях тяжёлых ионов по сравнению с элементарными процессами является чувствительным признаком фазовых переходов; это явление особенно наглядно при образовании гиперядер, которое планируется исследовать как на столкновениях пучков при максимальной барионной плотности, так и на фиксированных мишенях.

6) **Разработка теоретических моделей изучаемых процессов и теоретическое сопровождение экспериментов.**
Теоретическая программа включает детальное моделирование процессов в конкретных экспериментальных условиях комплекса, а также исследовательские разработки новых методов, в частности непертурбативной и решётчатой квантовой хромодинамики.

Ускорительный блок

Ускорительный блок проекта NICA предназначен для генерации пучков заряженных частиц, которые используются экспериментальными установками комплекса для реализации научных программ в множестве областей физики. Как упоминалось ранее, речь идет о:

- релятивистской ядерной физике;
- спиновой физике;
- радиобиологии;
- множестве прикладных направлений.

Важно отметить тот факт, что комплекс обладает достаточно уникальными характеристиками, а именно:

- разнообразием состава и структуры ускоряемых ионных пучков;
- гибкостью в реализации исследовательских программ;
- высокой светимостью при изучении взаимодействий частиц в доступном энергетическом диапазоне;

- множеством прикладных направлений;

что делает работы по обеспечению его успешного функционирования важным вкладом в развитие современной российской и мировой науки.

Структура блока приведена на рисунке 2 и включает в себя следующие компоненты:

1) **Инжекционная цепочка для лёгких ионов $\frac{q}{A} = \frac{1}{3}$.**
   предназначена для ввода в Нуклотрон пучков различных ионов от протонов до магния, включая: поляризованные протоны с энергией до 12 МэВ и дейтроны с энергией до 6 МэВ/нуклон. Основные элементы цепочки:
   - лазер;
   - дуоплазматрон;
   - SPI (источник поляризованных частиц);
   - резонансный линейный ускоритель ЛУ-20М;
   - каналы распределения пучков между компонентами системы.

   В зависимости от задач эксперимента может использоваться один из трёх перечисленных источников. Параметры источников и соответствующих пучков приведены в таблице 1.

Параметр	**Источник**			
	Лазер	**Дуоплазматрон**		**SPI**
Частицы	ионы до Mg^{10+}	H^+, D^+	He^{2+}	$\uparrow H^+$, $\uparrow D^+$
Частиц на импульс	$\sim 10^{11}$	$\sim 5 \cdot 10^{12}$	$\sim 10^{11}$	$5 \cdot 10^{11}$
Частота импульсов, Гц	0.5	1	1	0.2

Таблица 1 — Параметры источников лёгких ионов

Созданный пучок попадает в модернизированный ускоритель ЛУ-20М, который представляет собой совокупность форинжектора на основе трех секций ускорителя с пространственно-однородной квадрупольной фокусировкой , ускорителя типа Альварез ЛУ-20, а также промежуточных линий транспортировки пучка низкой и средней энергии.

Отношение заряда иона к его массе q/A	1.0	0.5	≥ 0.3
Энергия инжекции, [КэВ]	31	61.8	103
Максимальный ток, [мА]	10	20	10
Выходная энергия [МэВ/нуклон]	0.156		
Эмиттанс (на выходе) $[\pi \cdot$ см $\cdot$ мрад]	≤ 0.5		
Передача, %	> 85	> 89	> 93
В аксептансе ЛУ-20, %	70	71	80

Таблица 2 — Основные параметры форинжектора

Номер гармоники	$2\beta\lambda$
Энергия инжекции	156 КэВ/нуклон
Выходная энергия	5 МэВ/нуклон
Рабочая частота	145 МГц
Диаметр промежутка	1,4 м
Длина промежутка	14,4 м
Число дрейфовых трубок	$57 + 2$ полу-трубки
Фактор качества	40000
Синхротронная фаза	$31,5°$
Фокусирующая структура	FODO
Градиент магнитного поля в квадрупольных линзах	$(58.4 \div 7.4)$ Тл/м
Аксептанс	220 $\pi \cdot$ мм $\cdot$ мрад

Таблица 3 — Основные параметры ЛУ-20М

2) **Инжекционная цепочка для тяжелых ионов** $\frac{q}{A} = \frac{1}{6}$.
предназначена для ввода в Бустер пучков ионов вплоть до золота с энергией до 3.24 МэВ/нуклон. Основные элементы цепочки:
   - криогенный источник многозарядных тяжёлых ионов КРИОН-6Т;
   - высокочастотный резонансный линейный ускоритель HILAc;
   - каналы распределения пучков между компонентами системы.

   Источник разработан в ОИЯИ и основан на явлении, обнаруженном при изучении режимов работы электронно-лучевого источника EBIS[7], работающего в отражательном режиме. Суть явления заключается в проявлении облаком многократно отраженных электронов, заключенных в сильное магнитное поле соленоида, свойств, характерных для фазового перехода; что имеет результатом постепенное увеличение плотности электронной плазмы и переходу в стационарное состояние, называемое электронной струной. В таблице 4 приведены параметры источника и пучка,

создаваемого данным источником.

Частицы в основном режиме	Au^{31+}
Частиц на импульс	$\sim 2{,}5 \cdot 10^9$
Рабочая частота, Гц	0.6
Максимальная частота, Гц	10

Таблица 4 — Параметры источника тяжёлых ионов

Испущенный пучок через линию транспортировки пучка низкой энергии попадает в секцию с пространственно-однородной квадрупольной фокусировкой, где ускоряется до 300 КэВ/нуклон. Оттуда по линии транспортировки пучка средней энергии пучок попадает в две последовательно расположенные секции линейных укорителей с дрейфовыми трубками с квадрупольным дублетом, обеспечивающие выходную энергию 3.2 МэВ/нуклон. Ускоренный пучок по линии транспортировки отправляется в Бустер для последующего ускорения.

3) **Синхротроны.**

Бустер

Данный синхротрон выступает промежуточным ускорителем тяжелых ионов и их инжектором в Нуклотрон. Исходя из этого он выполняет следующие задачи:

- **аккумулирование** ионов, ускоренных HILAc, до уровня 2×10^9 ионов Au^{31+};
- **ускорение частично ионизированных ионов** благодаря обеспечению сверхвысокого вакуума в камере пучка;
- **формирование необходимого фазового объёма пучка** посредством системы электронного охлаждения;
- **ускорение тяжёлых ионов** до энергии, необходимой для их полной ионизации;
- **полная ионизация тяжелых ионов** на мишени, установленной в канале;
- **инжекция подготовленного пучка в Нуклотрон** с предварительным отсеиванием ионов с нецелевым зарядом.

Сам синхротрон представляет из себя установку периметром 210.96 метров, состоящую из 4 сверхпроводящих арок, каждая из которых имеет угол поворота 90°и состоит из 5 периодов, каждый из которых, свою

очередь, содержит фокусирующую и дефокусирующую квадрупольные линзы, 2 дипольных магнита и 4 малых свободных промежутка, предназначенных для размещения мультипольных корректоров, коллиматоров и диагностического оборудования. Между собой арки соединены прямыми участками, на которых размещаются системы ввода и вывода пучка, система электронного охлаждения, а также высокочастотная система ускорения и управления пучком.

Проектные возможности установки позволяют ей выводить в Нуклотрон пучок тяжелых ионов с энергией до 578 МэВ/нуклон.

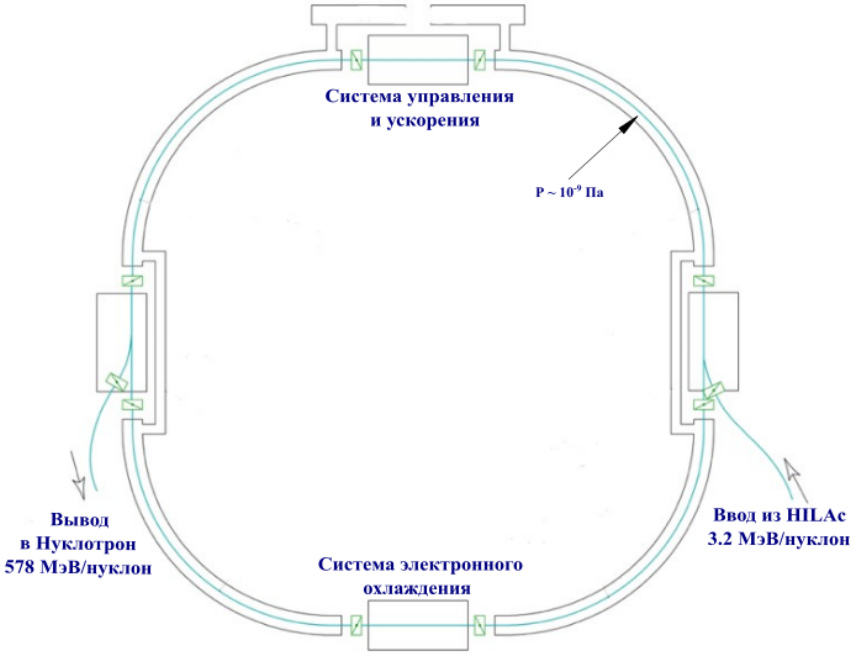


Рисунок 2 — Схема Бустера

Нуклотрон

Второй синхротрон является основным ускорителем всего комплекса и инжектором в накопительные кольца. Для самого же Нуклотрона, как было указано, инжекторами являются ЛУ-20М для легких и Бустер для тяжелых ионов соответственно. Из этого вытекают основные задачи, поставленные перед установкой, а именно:

- ускорение лёгких ионов для экспериментов на фиксированных мишенях;
- ускорение тяжелых ионов для экспериментов на фиксированных мишенях;
- ускорение различных ионов для экспериментов на внутренней мише-

ни;

- ускорение лёгких ионов для последующей инжекции в накопительные кольцах;
- ускорение тяжелых ионов для последующей инжекции в накопительные кольцах;
- ускорение различных ионов для последующей инжекции в накопительные кольцах при работе в несимметричной моде.

Совокупность 96 дипольных и 64 квадрупольных сверхпроводящих магнитов, разделенная на 8 суперпериодов, создаёт в кольце Нуклотрона периметром 251.5 м магнитное поле силой 1.8 Тл с магнитной жесткостью 38.5 Тл·м, что соответствует кинетической энергии:

- для протонов - 10.7 ГэВ
- для дейтронов и легких ионов с $Z/A = 1/2$ - 4.95 ГэВ/нуклон
- для ионов с $Z/A = 1/3$ - 2.6 ГэВ/нуклон
- для ионов золота - 4.95 ГэВ/нуклон

После ускорения пучки выводятся посредством системы медленного вывода, работающей по двухступенчатой схеме с эффективностью превышающей 95%, в павильоны для экспериментов на фиксированных мишенях и системы быстрого вывода, состоящей из магнита Ламбертсона и импульсного ударного магнита, в накопительные кольца Коллайдера.

4) **Сверхпроводящие накопительные кольца.**
5) **Транспортные каналы для передачи пучков между элементами комплекса.**

1.2 ЭКСПЕРИМЕНТ SPD

Экспериментальная установка SPD [8; 9], строящаяся в одной из двух точек пересечения пучков коллайдера NICA, предназначена для всестороннего изучения спиновой структуры протона и дейтрона.

Основное внимание будет уделено изучению их поляризованной глюонной компоненты в реакциях инклюзивного рождения чармониев, очарованных частиц и прямых фотонов, приведенных на рисунке 3, а также прочих спинзависимых явлений в столкновениях поляризованных пучков протонов и дейтронов с энергией в системе центра масс до 27 ГэВ и светимостью до 10^{32} см$^{-2}$с$^{-1}$, которая обеспечит поток данных $\sim$ 20 ГБ/с [10]. Посредством измерения со-

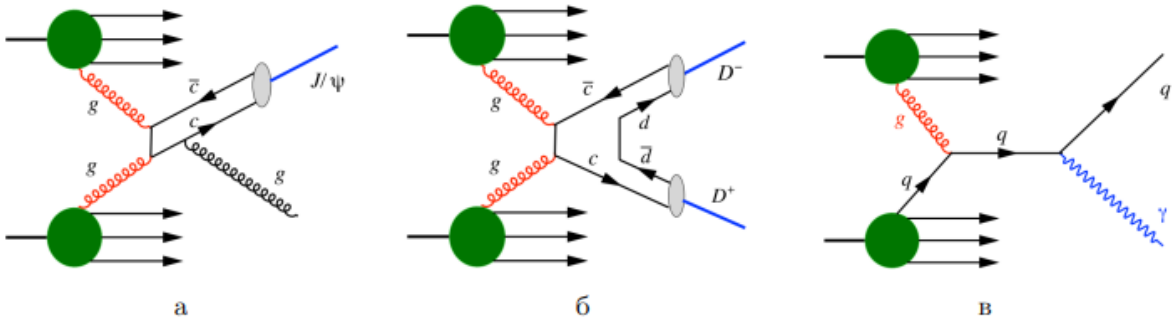


Рисунок 3 — Диаграммы Фейнмана для процессов рождение: (а) чармониев, (б) очарованных частиц, (в) прямых фотонов

ответствующих спиновых асимметрий будут получены данные по корреляциям между направлениями спина протона (дейтрона), его импульса, а также направлением спина, продольным и поперечным импульсами глюонов внутри протона (дейтрона).

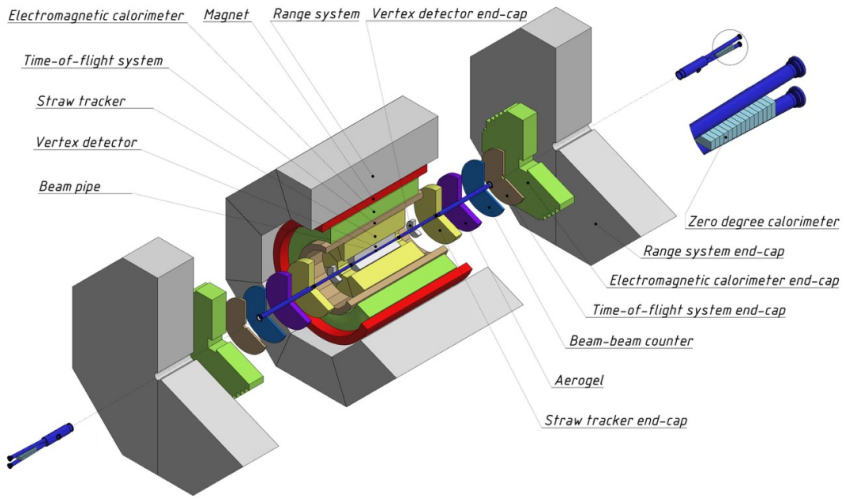


Рисунок 4 — Схема SPD

Для покрытия поставленных физических целей установка реализована как универсальный 4π детектор цилиндрической формы со следующим набором подсистем:

- **RS** - мюонная система;
- **TS** - трековая система для идентификации и восстановления импульсов заряженных частиц;
- **VD** - кремниевый вершинный детектор для реконструкции вторичных вершин распадов мезонов;
- **BBC** - внутренний и внешний счетчики столкновений для контроля по-

ляризации и светимости;

- **ZDC** - калориметры нулевых углов по обе стороны установки для контроля поляризации и светимости;
- **ECal** - электромагнитный калориметр для регистрации фотонов;
- **TOF** - время-пролетная система для разделения π/K и K/p
- **FARICH** - аэрогелевый детектор для разделения π/K и K/p

Данная программа исследований, на текущий момент, разделяется на два этапа, в соответствии с максимально достижимой яркостью на установке. На первом этапе работы установки основное внимание планируется уделить изучению спиновых эффектов в упругих p-p и d-d рассеяниях, поиску мультипартонных корреляций и новых связанных состояний. При этом количество обрабатываемых событий на втором этапе увеличится на порядок по сравнению с первым, а именно: с $2{\times}10^{11}$ до $2{\times}10^{12}$ событий ежегодно. Сопутствующие такому количеству событий объемы данных измеряются в петабайтах и сопоставимы с результатами экспериментов на БАК(Большой Адронный Коллайдер). В этом свете становится логичной возможность перенятия опыта соответствующих экспериментов; подробнее об этом будет рассказано в следующей главе. Здесь же рассмотрим, скорее, предметную область в виде этапов обработки больших объемов экспериментальных данных и высокоуровневой архитектуры системы, ответственной за эту обработку.

Система обработки данных SPD

Ввиду того, что работа в рамках диссертации сосредоточена на разработке системы управления процессами обработки данных, отдельно перечислим принципиальные шаги работы с данными (рисунок 5), где на каждом шаге будут получены выходные данные в соответствующих форматах, обеспечивающих их оптимальную обработку и хранение, а также воспроизводимость результатов и их соответствие текущей физической программе эксперимента.

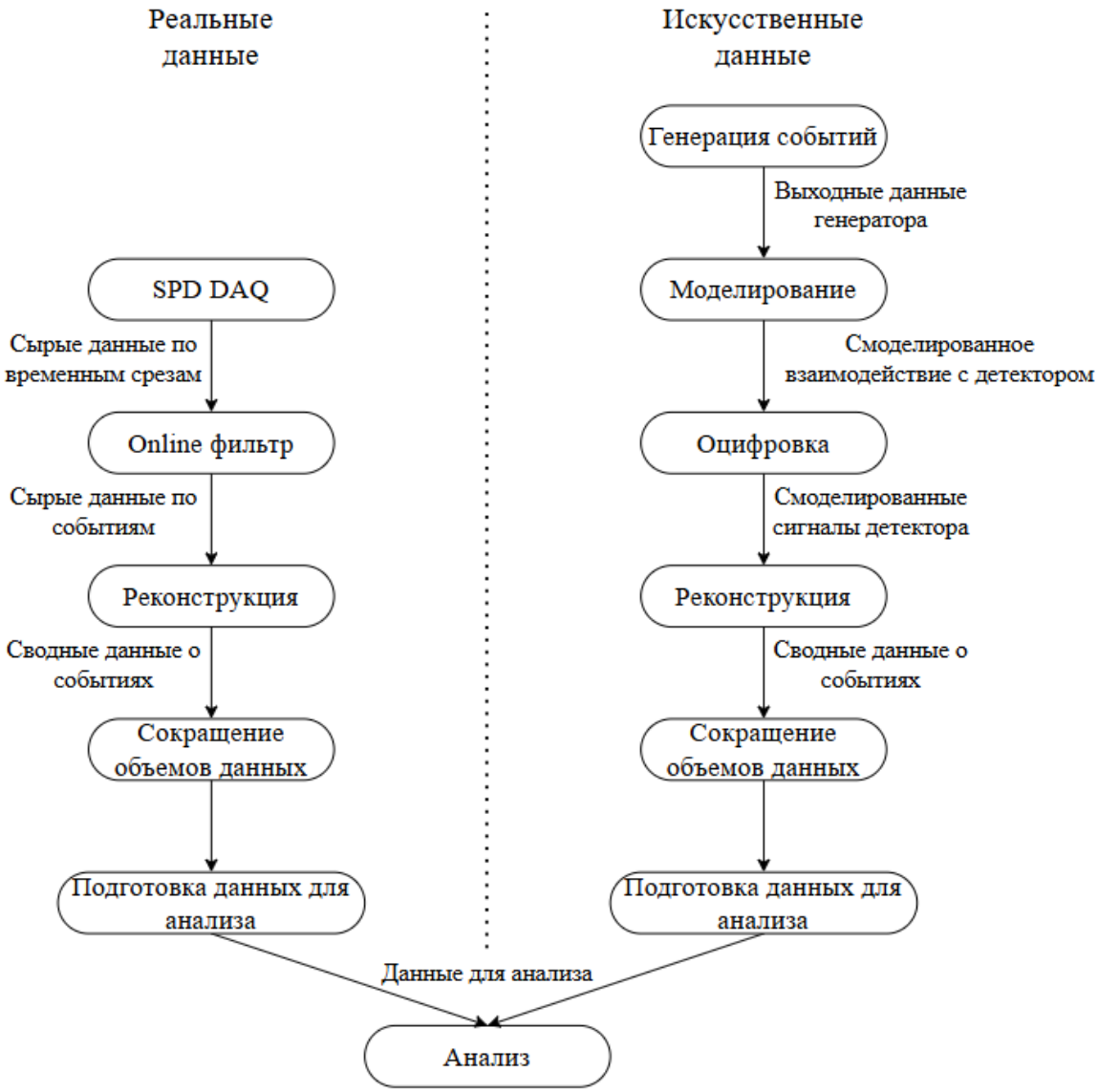


Рисунок 5 — Шаги работы с данными в рамках эксперимента SPD

Кроме того, приведем краткий обзор всей системы обработки на примере процесса MC моделирования, приведённого на рисунке 6.

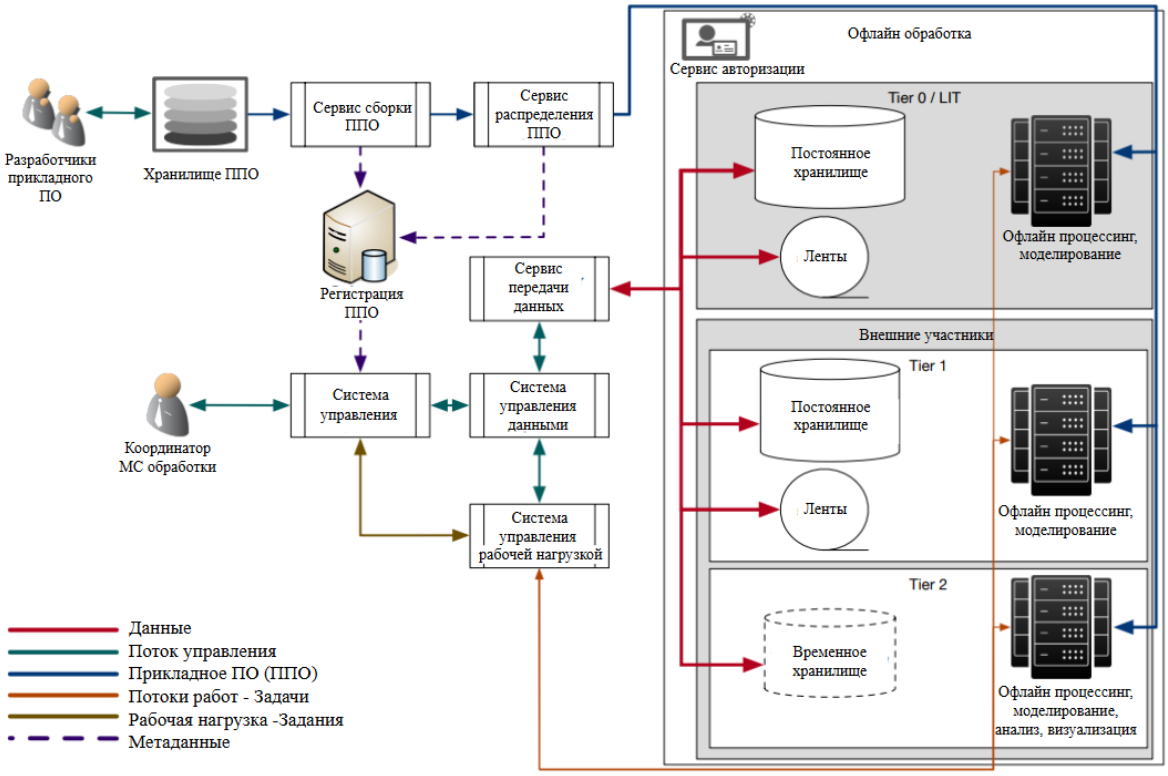


Рисунок 6 — Взаимосвязь компонентов системы обработки данных SPD

Здесь можно видеть фактически весь цикл работы с данными:

- координатор обработки собирает задания у заинтересованных физических групп;
- посредством системы управления эти задания поступают на обработку системой управления рабочей нагрузки;
- последняя из упомянутых систем занимается разбиением задания на задачи, и распределением получившихся небольших задач по кластерам и узлам непосредственной обработки;
- кроме того, эта система тесно связана с системой управления данными, осуществляющей консистентный контроль гетерогенной и георасределенной инфраструктуры хранения
- система управления, в добавлении к сказанному, также получает информацию о прикладном программном обеспечении, которое было разработано и поддерживается участниками коллаборации эксперимента SPD. На данный момент в качестве такого ПО выступает фреймворк SPDRoot [11], который в перспективе нескольких лет может быть замещен новым программным пакетом Sampo.

Все обращения к частям информационной экосистемы ОИЯИ произво-

дятся, разумеется, с использованием сервисов аутентификации и авторизации, позволяющих обеспечить безопасность и доступность разработанных программных средств путём выпуска, обмена и проверки токенов и сертификатов.

2 СИСТЕМА УПРАВЛЕНИЯ ЗАДАНИЯМИ И ВЗАИМОСВЯЗЬ IT-СОСТАВЛЯЮЩИХ СИСТЕМЫ

Как было упомянуто, ожидаемая скорость поступления данных будет порядка 10 петабайт в год, что даже с учетом онлайн-фильтра, отсеивающего около 90% данных, будет приводить к постоянному накоплению данных для обработки, кроме того, данные будут использоваться множеством разных физических групп, с соответственно разными программами исследований, разными нуждами в программном обеспечении и необходимых ресурсах. Ввиду этого возникает необходимость существования система высокого уровня, которая позволит наиболее эффективным образом организовывать создание, обработку и отслеживание заданий в условиях постоянного дефицита производительных мощностей, ожидаемого исходя из количества событий/год $\sim 10^{12}$ размером по 10-15 КБ каждое, с затратами на обработку от 1 до 10 секунд, и ведущего к необходимости постоянной работы 6-60 тысяч процессоров.

Проблема нехватки мощностей при работе с большими объемами данных с экспериментальных установок, конечно, не уникальна для SPD и возникала практически во всех экспериментах в физике элементарных частиц или физике высоких энергий, начиная с середины прошлого века. Но особенно ярко данная проблема проявилась в рамках экспериментов на БАК, где объемы данных настолько возросли, что стало недостаточно уже используемых решений в виде вычислительных кластеров и модульного подхода к созданию сервисов; а потребовалось разработать глобально распределенную вычислительную среду, способную справляться с поступающими экспериментальными данными. Для начала рассмотрим некоторые примеры соответствующих технических решений, разработанных и используемых в рамках экспериментов на SPS(Super Proton Synchrotron) и LHC(Large Hadron Collider).

2.1 РЕАЛИЗАЦИЯ ПОДОБНОЙ СИСТЕМЫ ЭКСПЕРИМЕНТА ATLAS

Коллаборацией ATLAS(A Toroidal LHC Apparatus) [12] разработан набор взаимодействующих систем и сервисов для хранения и обработки данных. Верхним уровнем системы является ProdSys(Production System) [13]. Это система, ответственная за управление процессами обработки данных, управляющая и/или взаимодействующая с прочими имплементированными сервисами, вроде системы управления данными Rucio [14], системы управления нагрузкой PanDA(Production and Distributed Analysis) [15], информационной системы CRIC(Computing Resource Information Catalogue) [16].

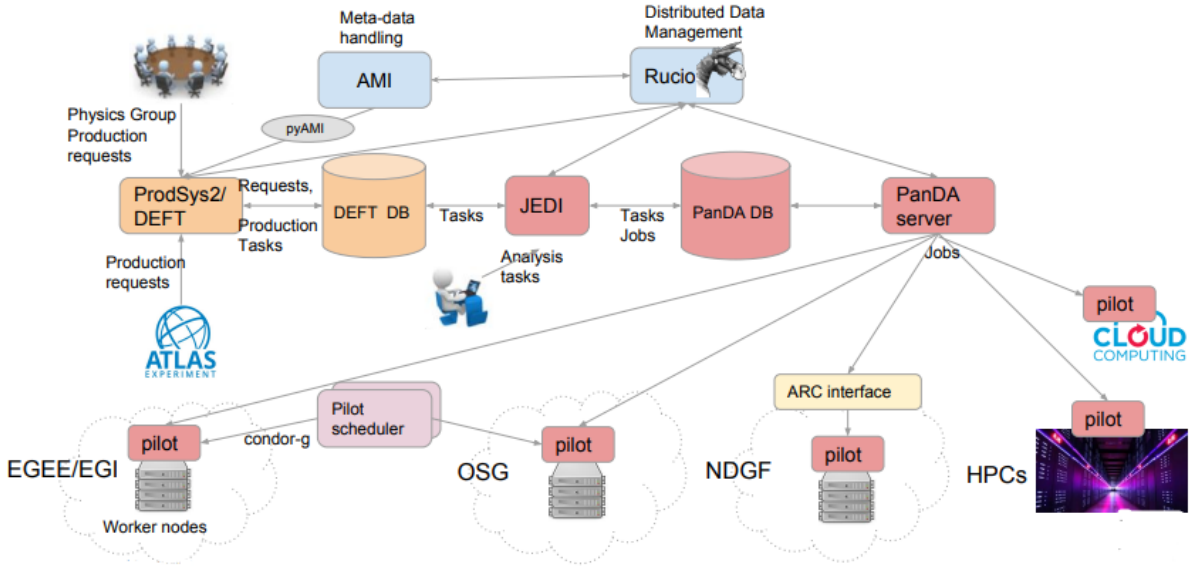


Рисунок 7 — Взаимосвязь сервисов и систем работы с данными ATLAS

ProdSyS

Система разработана для запуска заданий на разнообразном наборе вычислительных ресурсов для большого числа пользователей с множеством различных комплексных рабочих процессов. Чтобы успешно справляться с вытекающими из вышеперечисленного вызовами, быть способной развиваться и приспосабливаться к новым требованиям, система разделена на несколько независимых централизованных сервисных уровней.

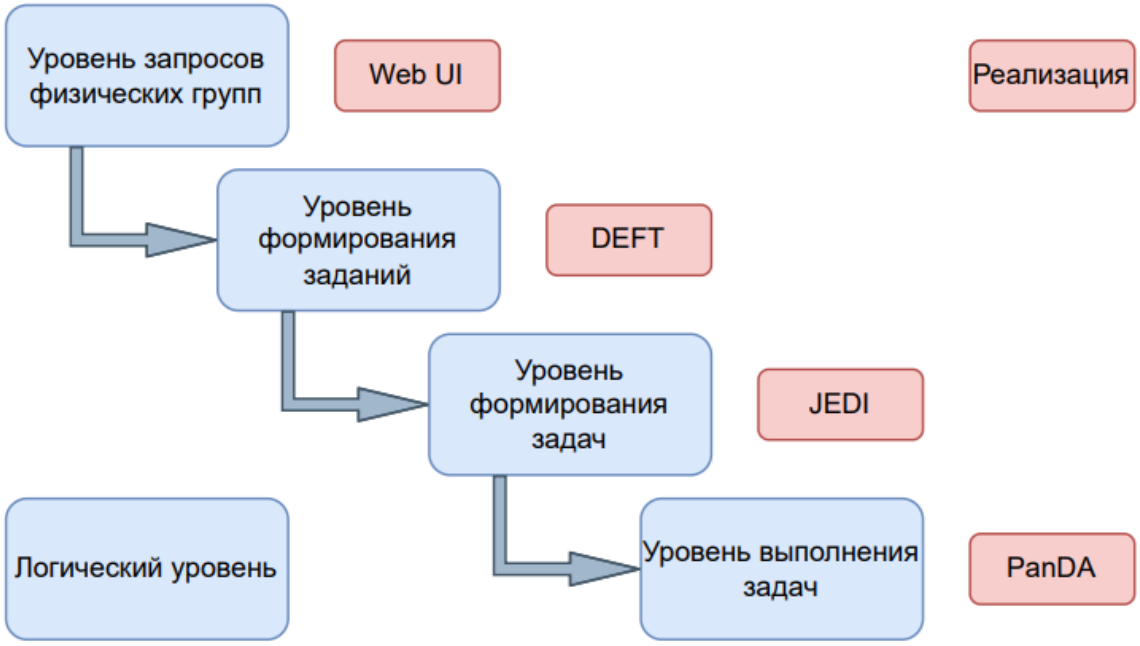


Рисунок 8 — Компоненты ProdSyS

При этом, инициируя процесс обработки данных, физические группы взаимодействуют со всей системой ProdSyS только на верхнем уровне и работают не с конкретными задачами и файлами, а с их совокупностями: заданиями, датасетами и контейнерами соответственно. В рамках работы группы система предоставляет определённым представителям группы роль Production Manager для обеспечения возможности определять поток обработки данных для заданий группы, управлять заданиями, определять требуемые ресурсы и прикладное программное обеспечение. Именование создаваемых объектов реализовано в специальном формате, представленном на рисунке 9, для упрощения, унификации работы и взаимодействий подсистем, а также для удобства физических групп.

Рисунок 9 — Структура имен заданий и датасетов

DEFT(Database Engine For Tasks) отвечает, как следует из рисунка 8, за формирование заданий, а также за хранение их состояний для систем обработки данных и общей системы мониторинга.

Rucio

Rucio [17]— это фреймворк с открытым исходным кодом, разработанный для организации, управления и обеспечения доступа к данным в рамках данного эксперимента и в дальнейшем распространившийся в своем использовании как на другие эксперименты на БАК, так и на сторонние эксперименты.

Основные используемые понятия:

- **Пространство имён** отвечает за адресацию данных различными способами.
- **Учётные записи** обеспечивают авторизацию, аутентификацию и управление правами доступа.
- **Хранилище** предоставляет единый интерфейс к распределённым дата-центра
- **Подписки** используются для реализации политик передачи данных на масштабах всей системы.
- **Правила репликации** гарантируют консистентное распределённое состояние пространства имён на хранилище.

Ключевые особенности Rucio:

- **масштабируемость:** поддержка распределённого хранения данных в гетерогенных центрах обработки данных, расположенных географически удалённо. В рамках эксперимента система продемонстрировала успешную работу с более чем двумя сотнями элементов хранения;
- **оптимизация потока данных:** хотя в общем случае Rucio не взаимодействует напрямую с физической сетевой инфраструктурой, оно может использовать накопленные за историю работы сетевые метрики — пропускную способность, потерю пакетов и задержку — для выбора оптимальных путей передачи данных;
- **гибкость:** поддержка разнородных систем хранения, таких как дисковые массивы, ленточные библиотеки, облачные хранилища. Это достигается благодаря встраиванию существующих или добавлению любых кастомных протоколов в систему посредством плагинов, использующих внутренние интерфейсы взаимодействия Rucio, имитирующие стандартные POSIX-операции;

- **приближенное удовлетворение ACID:** глобальный учёт данных с поддержкой транзакционности;
- **интегрируемость:** бесшовная интеграция с ProdSyS и PanDA, возможность интеграции сторонних сервисов и систем ;
- **разграничение доступа:** реализация системы идентификации, аутентификации и авторизации, в том числе с использованием сертификатов и сторонних систем, в частности SSO;
- **хранение метаданных:** для обеспечения иерархической организация данных с использованием правил репликации.

Данные в Rucio организуются с помощью идентификаторов данных (Data Identifiers), которые представляют собой многоуровневую структуру, приведенную на рисунке 10.

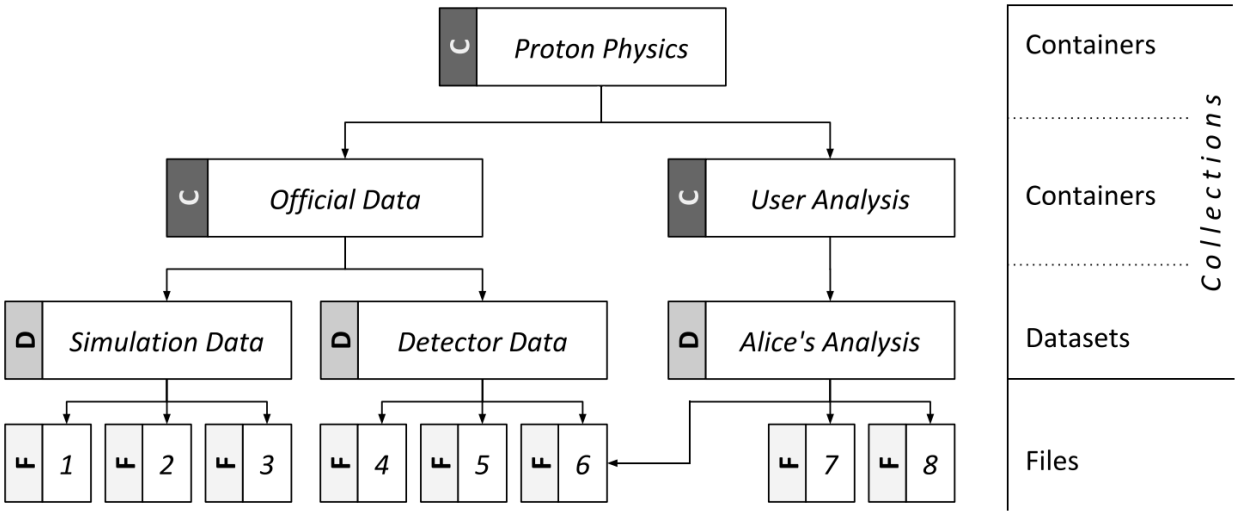


Рисунок 10 — Структура пространства имен

Наименьшая единица, которой оперирует система - реальный файл на хранилище. Файлы объединены в логические группы - датасеты, предназначенные для упрощения массовых операций и структурирования данных. Важным пунктом для распределенной системы тут является возможность объединения в датасет как сугубо локальных файлов, так и файлов с разных точек хранения. Датасеты, в свою очередь, также могут быть объединены в связный логический объект - контейнер, что позволяет единообразно оперировать еще большими объемами данных и организовывать эти самые данные в иерархичную структуру произвольной глубины.

В целях унификации подхода к хранению и удобства оперированием фай-

лами и коллекциями (датасетами и контейнерами) их именование подчиняется согласованной структуре, в формате <scope>:<name>, где <scope> разделяет глобальное пространство имён, что повышает гибкость организации и предоставления прав, а <name> - уникально идентифицирует сам объект внутри <scope>.

Как было сказано ранее, Rucio хранит метаданные для обеспечения корректной организации файлов, в частности: чек-сумму, вычисляемую с использованием алгоритмов MD5 или Adler32, каждого файла, зарегистрированного в системе, проверяемую при любом обращении к файлу, будь то использование для анализа, перемещение или добавление в коллекцию, что позволяет поддерживать консистентность хранимых и используемых данных; статус доступности, отображающий наличие правил репликации и хотя бы одной реплики на хранилище; флаг отображения файла в коллекциях и прочие метаданные.

Также имеются встроенные средства мониторинга, используемые для отслеживания состояния системы, объемов и типов управляемых ресурсов, доступности систем хранения и эффективности взаимодействия между этими системами.

PanDA

Система распределения и обработки рабочих нагрузок, спроектированная для работы на масштабах эксперимента LHC. PanDA была разработана для удовлетворения комплексных требований по обработке данных, моделированию детектора и физическому анализу в рамках эксперимента ATLAS, после чего, аналогично Rucio, применялась на ряде других установок по исследованию в области физики высоких энергий и не только.

В рамках своего дебютного эксперимента система успешно справлялась с:

- оркестрацией выполнения заданий на более чем двух сотнях вычислительных центров в 40 странах по всему миру;
- распределением нагрузки, поступающей от тысяч ученых;
- объемом нагрузки на уровне экзобайт;
- встраиванием сторонних прикладных программ и сценариев выполнения обработки.

Одним из ключевых принципов проектирования PanDA стала тесная интеграция управления нагрузкой и данными — в частности, с вышеописанной системой распределённого управления данными Rucio — что позволило эффективно обрабатывать около 100 ПБ данных ежегодно. Также проектирование велось с учётом гибкости к адаптации новых технологий в области обработки, хранения данных, сетей и распределённых вычислительных сред. Система справилась с бесшовной интеграцией широкого спектра ресурсов: от Всемирной вычислительной сети LHC (Worldwide LHC Computing Grid или WLCG) и волонтёрских вычислений до независимых от WLCG систем высокопроизводительных вычислений, лидирующих вычислительных центров и коммерческих облачных сервисов.

Краеугольным понятием в работе PanDA является термин **"Задание"** или **"Task"**. Это объект с чётко определёнными состояниями, соответствующий одной **рабочей нагрузке**. По сути, задание есть экземпляр рабочей нагрузки вместе с инструкциями на её выполнение, ссылками на необходимые данные и набором метаданных. Задание принимает на вход данные и выдаёт результат обработки. Соответственно, цель задания — полностью обработать входные данные. Как правило, входными и выходными данными являются коллекции файлов, однако могут использоваться и другие форматы, такие как: набор последовательных чисел, метаданные, коллекции чанков (sub-file data) или уведомления.

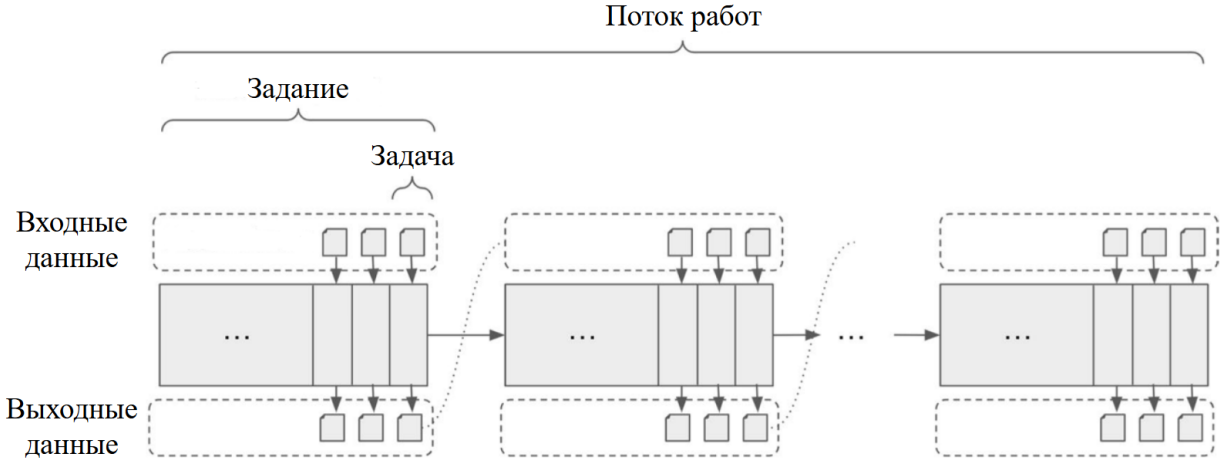


Рисунок 11 — Иерархия объектов в процессе обработки

Поток работ представляет собой верхнеуровневую единицу в иерархической организации обработки данных, объединяющую несколько рабочих нагру-

зок и описывающую их топологию. Каждый поток соответствует научной цели, разбитой на промежуточные шаги, необходимые для её достижения. **Рабочая нагрузка** — это программный модуль или приложение, использующее вычислительные ресурсы для выполнения одного из промежуточных шагов потока работ. Пользователь может запускать все шаги вместе или их часть в составе конкретного потока.

В рамках PanDA реализована возможность выполнения нагрузок в двух режимах:

1) Управляемая нагрузка, запускаемая централизованно на уровне виртуальной организации (VO). Это позволяет эффективно задействовать значительный объём вычислительных ресурсов, выделенных конкретной VO.
2) Пользовательская нагрузка, инициируемая отдельным исследователем, использующим свои квоты, но выполняющая задачи в интересах VO.

Один поток может включать как управляемые, так и пользовательские нагрузки. Общая структура потока приведена на рисунке 11, но важно отметить, что поток не обязательно должен быть линейным, он может реализовывать как параллельное выполнение, так и наложение времени выполнения конкретных заданий, что является дополнительным инструментом оптимизации процесса обработки. Минимальным же объектом в работе PanDA является, как можно видеть на том же рисунке 11, **задача** или **job** — объект с набором состояний, представляющий собой часть разбиения одного задания. Каждая задача получает на вход свою часть данных и по завершении формирует часть выходного результата. Сама задача выполняется на **работнике** - абстракции вычислительного ресурса, на физическом уровне реализованной в виде кластеров или рабочих узлов с разным набором процессоров, графических ускорителей, операционной памяти и средств хранения. Во многом при назначении работников на определенные задачи система старается реализовывать подход "Алгоритмы к данным" , однако нередко приходится прибегать к куда менее эффективным с точки зрения передачи трафика стратегиям "Данные к алгоритмам" или вообще пересылать и то, и другое на третий ресурс ввиду политики глобального распределения ресурсов и приоритетов. **Глобальное распределение ресурсов** — механизм, определяющий, как совокупные вычислительные ресурсы делятся между различными группами VO и типами пользовательских заданий. Управляемые системой ресурсы динамически разбиваются на несколько глобальных

доменов, которые назначаются каждому заданию в зависимости от запускающего его субъекта. Внутри системы многие компоненты оперируют именно этими доменами для планирования, приоритизации и учёта использования ресурсов в соответствии с заданными квотами и политиками. Говоря об упомянутой уже не раз приоритизации, скажем, что **приоритет** задания или задачи определяет, какой из конкурирующих процессов в рамках одного домена будет обслужен раньше. Как правило, задачи наследуют приоритет задания, которое их породило, однако первые несколько из них получают повышенный приоритет для оперативного сбора метрик, необходимых для дальнейшей обработки. Данный приоритет является универсальным, то есть распространяется на любое событие, связанное с процессом, будь то переходы состояния, обращение к внешним системам или непосредственное выполнение.

Описав объекты, которыми оперирует система, перейдем к описанию её управляющих компонентов.

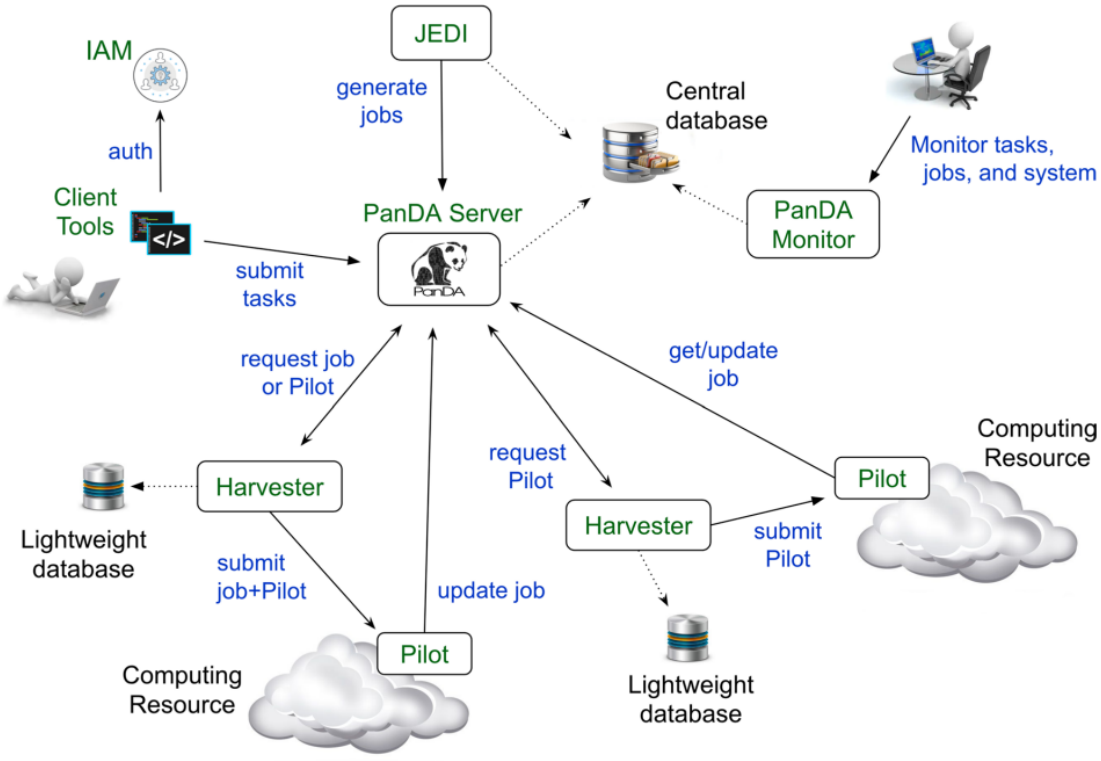


Рисунок 12 — Архитектура PanDA

Архитектура системы представлена на рисунке 12, где изображён ряд пока не затронутых составляющих:

- **JEDI** (Job Execution and Defnition Interface) — высокоуровневый движок для динамического формирования задач с целью оптимального использо-

вания гетерогенных географически распределённых ресурсов. Среди выполняемых им функций:

- приём и разбор спецификаций пользовательских заданий
- сбор информации о входных коллекциях и чанках
- определение точек хранения результатов выполнения задач и заданий в целом
- выбор вычислительных ресурсов с учётом характеристик и требований каждой процесса
- формирование задач и их назначение на вычислительные элементы с учётом политики глобального распределения ресурсов
- оптимизация параметров задачи на основе результатов предыдущих в том же задании
- динамическая балансировка нагрузки
- поддержка управляющих команд в реальном времени

- **PanDA-сервер** — центральный пул задач, реализованный как RESTful веб-сервис, развернутый на серверном ПО Apache, обеспечивающий асинхронную связь с пользователями, пилотами и сборщиками через HTTPS, а также набор планировщиков задач, выполняемых по расписанию. Сервер управляет задачами на всём их жизненном цикле. Основные функции сервера:
  - приём задач, генерируемых JEDI, и задач из прочих источников, генерируемых, в основном, для тестирования;
  - обеспечение доступности входных данных на хранилищах, связанных с вычислительными ресурсами на которых запланировано выполнение задачи;
  - отправка задачи на рабочий узел после проверки доступности на нем входных данных;
  - отслеживание состояний задач на рабочих узлах ;
  - передача метаданных чанков и отслеживание их обработки на соответствующем уровне разбиения, если задача сконфигурирована для подобной обработки;
  - выполнение пост-обработки после завершения задачи на узле
  - отслеживание команд на ручное управление задачами;
  - отправка обратной связи по задачам, сгенерированных JEDI.

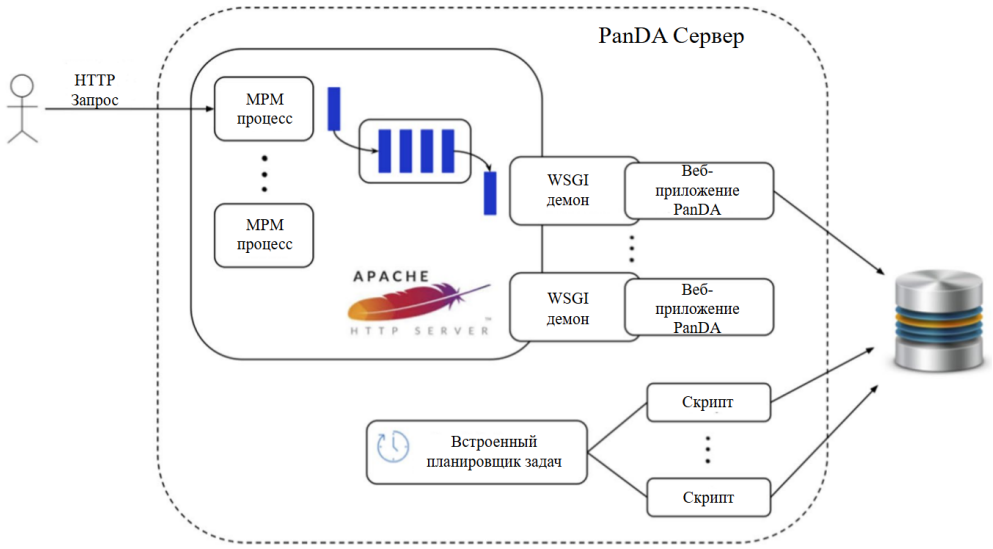


Рисунок 13 — Архитектура экземпляра PanDA-сервера

Благодаря архитектуре PanDA-сервера, приведённой на рисунке 13, его можно легко масштабировать за счёт добавления новых экземпляров. Архитектура включает Apache HTTP-сервер, который управляет WSGI-демонами и динамически масштабируемыми дочерними процессами для обработки запросов от пользователей и управляемых агентов, таких как **пилоты** и **сборщики**. Поддерживаются синхронные и асинхронные запросы: первые блокируют отправителя до завершения, вторые — сразу возвращают ответ, а обработка продолжается в фоновом режиме. Встроенный планировщик задач заменяет cron, исключая наложения запусков, обеспечивая контроль за параллельным выполнением скриптов и синхронизацию между экземплярами сервера.

- **Пилот** (Pilot) — временный агент, выполняющий задачу на работнике и периодически отчитывающийся перед PanDA-сервером о различных метриках в течение всего своего жизненного цикла.
- **Сборщик** (Harvester) — компонент, генерирующий и отправляющий пилоты с использованием подходящего протокола для каждого поставщика ресурсов.
- **Система мониторинга PanDA** (BigPanDAmon) — веб-интерфейс для мониторинга задач и заданий в PanDA, предоставляющий единый портал для конечных пользователей, администраторов и технических специалистов. На портале приведены совокупные отчёты, дашборды и графические представления основных объектов системы (задания, задачи, площадки,

пользователи). Интерфейс позволяет как детально исследовать причины отказов на узлах обработки, так и отслеживать общую картину — производительность площадок или ход масштабных кампаний по обработке. Сервис реализован на основе фреймворка Django с ядром из базовых представлений и набором подключаемых модулей. Внутренний ORM-слой Django позволяет сервису поддерживать Postgre и Oracle, используемые для получения и хранения промежуточных данных. Сами данные система аккумулирует из разных источников, среди них: база данных PanDA, CRIC, Rucio, Elasticsearch-кластер с логами PanDA/JEDI-приложений, MONIT Grafana с учётными и статистическими данными для ATLAS.

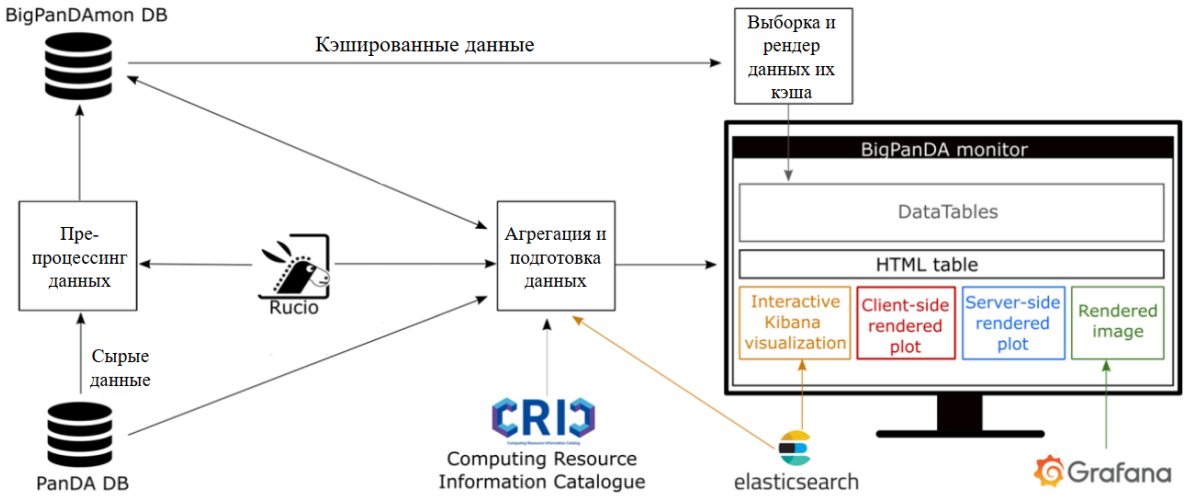


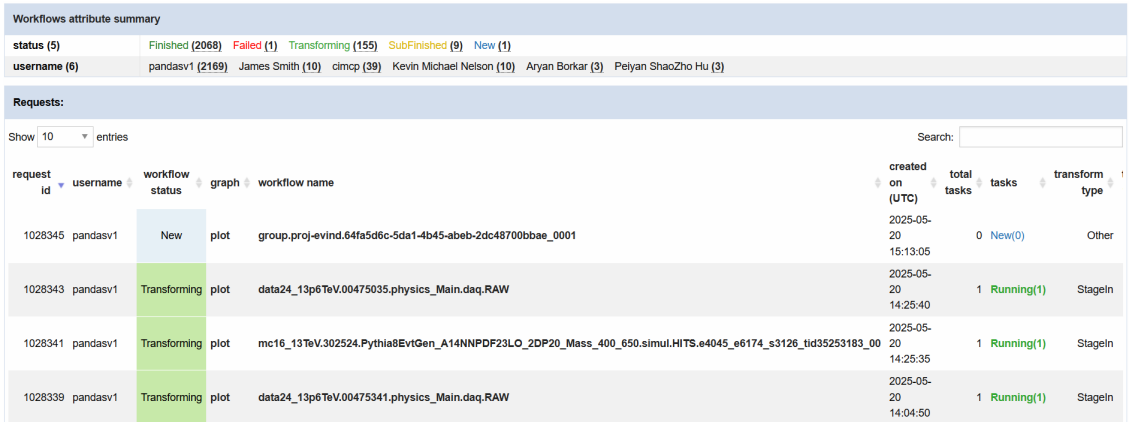
Рисунок 14 — Потоки данных для системы мониторинга

Для обеспечения контролируемого доступа к данным используется аутентификация посредством протокола OAuth2 с SSO-провайдерами в лице ЦЕРНа, GitHub и Google. Также имплементированы механизмы кэширования и предварительной обработки: кэш на уровне БД хранит готовые к рендерингу данные, регулярные фоновые задачи подготавливают отчёты заранее, общее CephFS хранилище содержит логи и отрисованные графики Grafana, что позволяет улучшить производительность конечных запросов в момент обращения пользователей. Серверная отрисовка статического контента производится встроенными средствами Django, а динамика и интерактивность обеспечиваются благодаря использованию AngularJS. В общей сложности система включает более 70 различных представлений, доступных в виде интерактивных веб-страниц или JSON, в дополнение к 54 эндпоинтам для запросов дополнительных данных по требованию.

Некоторые представления приведены ниже.

Cloud	GOC site name	Tier	Queue name	Status	Type	Workflow	System	Copytools	Associated Harvester	Min RSS [GB]	Max RSS [GB]	Max time [hours]	Max input size [GB]	Last comment
RU	JINR-LCG2	T2	JINR	online	production	pull_ups	---	rucio	CERN_central_A	0	16.4	72	---	no active blacklisting rules defined

Рисунок 15 — Статус центров обработки в РФ



Workflows attribute summary

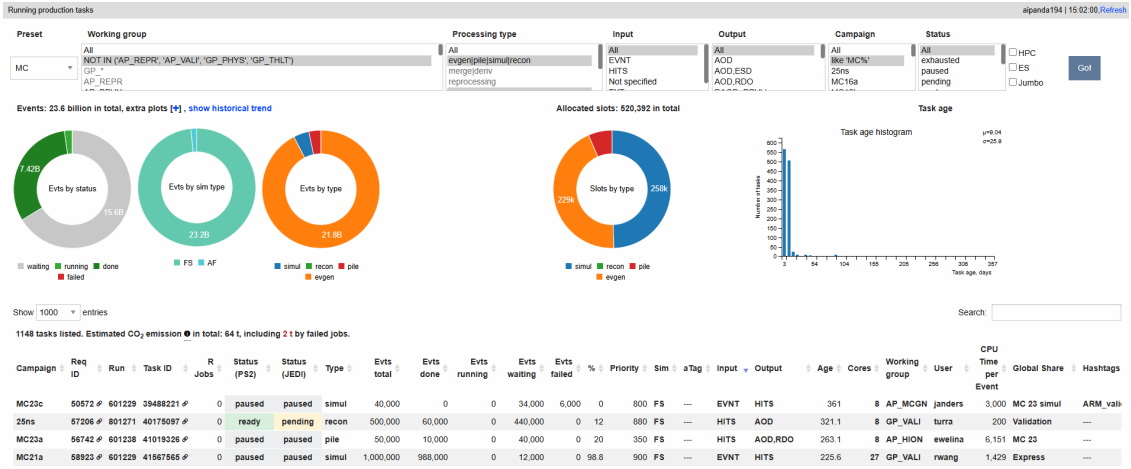
status (5)	Finished (2068) Failed (1) Transforming (155) SubFinished (9) New (1)
username (6)	pandasv1 (2169) James Smith (10) cimcp (39) Kevin Michael Nelson (10) Aryan Borkar (3) Peiyan ShaoZho Hu (3)

Requests:

Show 10 entries Search:

request id	username	workflow status	graph	workflow name	created on (UTC)	total tasks	tasks	transform type
1028345	pandasv1	New	plot	group.proj-evind.64fa5d6c-5da1-4b45-abeb-2dc48700bbae_0001	2025-05-20 15:13:05	0	New(0)	Other
1028343	pandasv1	Transforming	plot	data24_13p6TeV.00475035.physics_Main.daq.RAW	2025-05-20 14:25:40	1	Running(1)	StageIn
1028341	pandasv1	Transforming	plot	mc16_13TeV.302524.Pythia8EvtGen_A14NNPDF23LO_2DP20_Mass_400_650.simul.HITS.e4045_e6174_s3126_tid35253183_00	2025-05-20 14:25:35	1	Running(1)	StageIn
1028339	pandasv1	Transforming	plot	data24_13p6TeV.00475341.physics_Main.daq.RAW	2025-05-20 14:04:50	1	Running(1)	StageIn

Рисунок 16 — Общий мониторинг заданий



Running production tasks

Events: 23.6 billion in total, extra plots [+] , show historical trend

Allocated slots: 520,392 in total

Task age

Show 1000 entries Search:

1148 tasks listed. Estimated CO_2 emission in total: 64 t, including 2 t by failed jobs.

Campaign	Req ID	Run	Task ID	R Jobs	Status (PS2)	Status (JEDI)	Type	Evts total	Evts done	Evts running	Evts waiting	Evts failed	%	Priority	Sim	aTag	Input	Output	Age	Cores	Working group	User	CPU Time per Event	Global Share	Hashtags
MC23c	50572	601229	39488221	0	paused	paused	simul	40,000	0	0	34,000	6,000	0	800	FS	---	EVNT	HITS	361	8	AP_MCGN	janders	3,000	MC 23 simul	ARM_vali
25ns	57206	801271	40175097	0	ready	pending	recon	500,000	60,000	0	440,000	0	12	880	FS	---	HITS	AOD	321.1	8	GP_VALI	turra	200	Validation	---
MC23a	56742	601238	41019326	0	paused	paused	pile	50,000	10,000	0	40,000	0	20	350	FS	---	HITS	AOD,RDO	263.1	8	AP_HION	ewelina	6,151	MC 23	---
MC21a	58923	601229	41567565	0	paused	paused	simul	1,000,000	968,000	0	12,000	0	98.8	900	FS	---	EVNT	HITS	225.6	27	GP_VALI	rwang	1,429	Express	---

Рисунок 17 — Мониторинг выполняемых заданий

PanDA поддерживает выполнение множества типов обработки, перечень которых представлен на рисунке 18 и доступен посредством функционала, предоставляемого специализированными Python модулями и инструментами командной строки из библиотеки **PanDa Client** [18]. Данная библиотека отвечает за обеспечение связи между пользовательским веб-интерфейсом и PanDA-сервером, что позволяет ускорить процесс создания заданий путем автоматизации ряда процессов и внедрения валидации. В случае, если пользователь хочет запустить задание вручную или использовать нестандартное для текущих процессов обработки прикладное ПО, то он может как использовать методы из модуля Client, так и инструменты командной строки, которые приведены ниже.

Основные компоненты и функции PanDA client включают:

- Отправка заданий; автоматизация процесса подготовки и запуска заданий на распределённых ресурсах [prun ...]:
  - –inDS: Входной набор данных;
  - –outDS: Выходной набор данных;
  - –exec: Команда для выполнения;
  - –bexec: Создание бинарных файлов;
  - –nFiles: Количество файлов для обработки;
  - –writeInputToTxt: Запись входных файлов в текстовый файл;
  - –outputs: Спецификация выходных файлов;
  - –containerImage: Образ контейнера для выполнения;
  - –architecture: Запуск на GPU ;
  - –match: Фильтр входных данных;
  - –nJobs: количество задач в задании;
  - –nEventsPerJob: количество событий в задаче;
  - ...
- Интеграции со специализированным прикладным ПО эксперимента - Athena [pathena ...]:
  - –inDS: Входной набор данных;
  - –outDS: Выходной набор данных;
  - –trf: Трансформационный скрипт;
  - –containerImage: Образ контейнера для выполнения;
  - –nJobs: количество задач в задании;
  - –nEventsPerJob: количество событий в задаче;
  - ...
- Интерфейс для управления заданиями и мониторинга их статуса [pbook ...]:

Monte-Carlo Production
Data Reprocessing
High Level Trigger
Tier-0 spill-over
SW Validation
Physics groups production
Derivation production in trains
Open-ended production
Users Analysis

Рисунок 18 — Типы заданий

- kill: прервать выполнение задания;
- finish: завершить выполнение задания;
- retry: перезапустить выполнение задания;
- show: показать статус задания;
- pause: приостановить выполнение задания;
- resume: возобновить выполнение задания.

- Настройка параметров отправки заданий и управления ресурсами

В рамках Python модулей также доступен широкий перечень возможностей для работы с заданиями:

- insertTaskParams(taskParams, ...) - создание заданий, ожидает на вход исчерпывающий словарь параметров;
- killTask(jediTaskID, ...) - принудительное завершение задания без каких либо условий;
- finishTask(jediTaskID, ...) - прекращение генерации новых задач в рамках задания;
- retryTask(jediTaskID, ...) - повторный запуск задач в рамках частично завершенных заданий;
- pauseTask(jediTaskID, ...) - приостановление выполнения задач в рамках задания;
- resumeTask(jediTaskID, ...) - возобновление выполнения задач в рамках задания;
- getJediTaskDetails(jediTaskID, ...) - получения детализации параметров задания;
- getPandaIDsWithTaskID(jediTaskID, ...) - получения внутреннего идентификатора задания в рамках PanDA;
- getTaskStatus(jediTaskID, ...) - получение текущего статуса задания.

Аналогичный функционал также реализован для конкретных задач в рамках задания.

2.2 РЕАЛИЗАЦИЯ ПОДОБНОЙ СИСТЕМЫ ЭКСПЕРИМЕНТА COMPASS

Базируясь в CERN, эксперимент COMPASS [19] реализует работу с данными, опираясь на IT-сервисы, предлагаемые департаментом информационных технологий европейской организации ядерных исследований. К 2020 году [20] [21] была произведена масштабная модернизация инфраструктуры IT-сервисов эксперимента: вывод системы AFS [22] из эксплуатации, подготовка к замене системы долговременного хранения CASTOR [23] на CTA [21], пакетная обработка перешла из-под управления LSF [24], под управление HTCondor [25]. Наиболее важным аспектом модернизации было внедрение процессов обработки задач в эксперименте внутрь PanDAWMS [26] [27].

Архитектура приобрела следующий вид (рисунок 19)

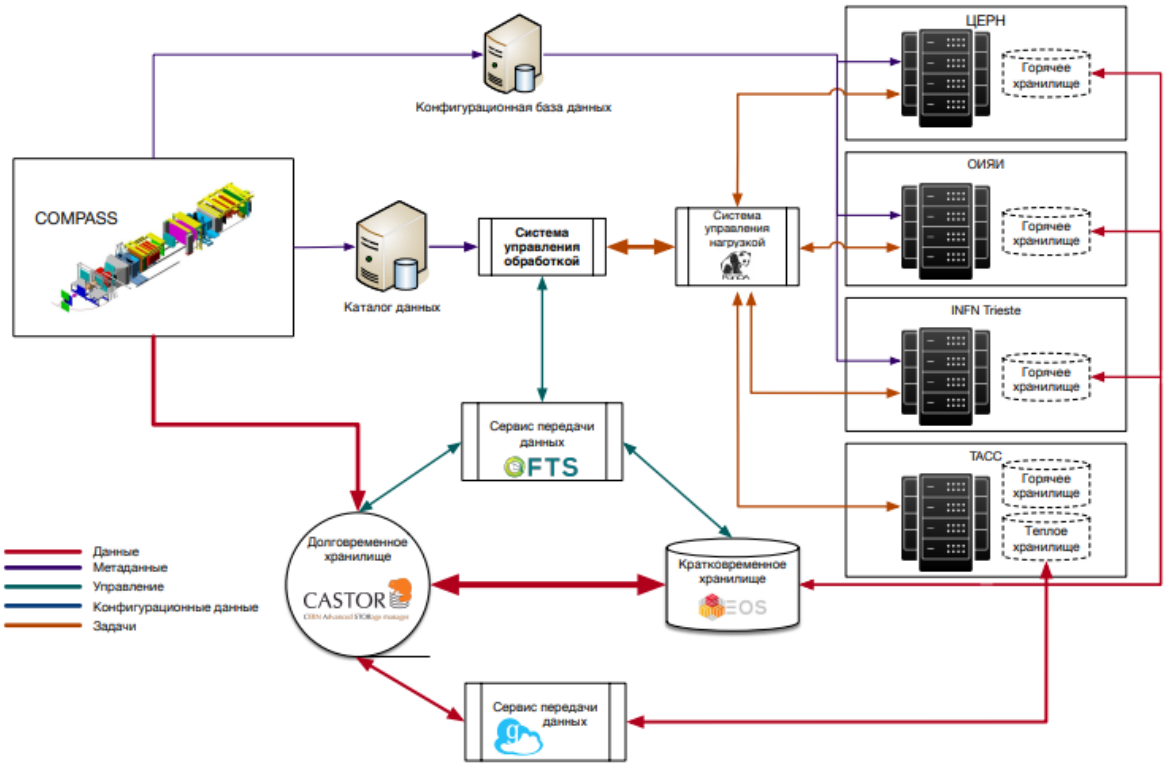


Рисунок 19 — Взаимосвязь сервисов и систем работы с данными COMPASS на 2020 год

После завершения модернизации процесс управления обработкой данных и мониторинга сильно видоизменился.

Ключевые пункты:

- описание заданий было перенесено в web-интерфейс;

- обработка перенесена в георасспределенную систему;
- взаимодействие с вычислительными центрами стала осуществлять посредством сервиса PanDA, с использованием x509-сертификатов для авторизации, протокола передачи данных XRootD и сервиса FTS;
- мониторинг заданий и задач в созданном интерфейсе.

Так же в 2022 году был осуществлён переход на новую систему хранения CTA [21], заменившую CASTOR, однако с точки зрения объектов хранения изменений не произошло, база данных системы все ещё хранит историю выполнения задач и является каталогом обработанных данных с соглашением об именовании: год/период/название обработки/номер сброса детектора-номер файла-параметры обработки, пример приведён на рисунке 20.

```
/data/2016/oracle_dst/P11/slot2/mDST/mDST-276384-2-7.root.001
```

Рисунок 20 — Структура имен заданий и датасетов

В итоге, с точки зрения процессов, система управления приобрела следующий вид:

- **управление заданиями** – веб-сервис, где менеджер, специализирующийся на управлении обработкой, может инициировать обработку задания, полученного от физических групп, а также контролировать их выполнение с помощью широкой системы мониторинга. На рисунке 21 можно видеть форму создания и таблицу отслеживания, а на рисунке 22 - интерфейс мониторинга состояний текущих заданий;
- **управление выполнением задач** – обеспечивается системой PanDA, ответственной за определение, доставку и непосредственное выполнение задач на вычислительные ресурсах;
- **управление процессами обработки** – обеспечивается рядом сервисов, обеспечивающих автоматизированную обработку задач;
- **управление данными** – операции с данными до, во время и после основного этапа обработки, в том числе: подготовка данных, работа с промежуточными файлами и архивирование результатов выполняется набором сервисов;
- **сбор данных для мониторинга** – совокупность отдельных сервисов, обеспечивающих подробный сбор разнородной информации о ходе процесса обработки на всех его этапах.

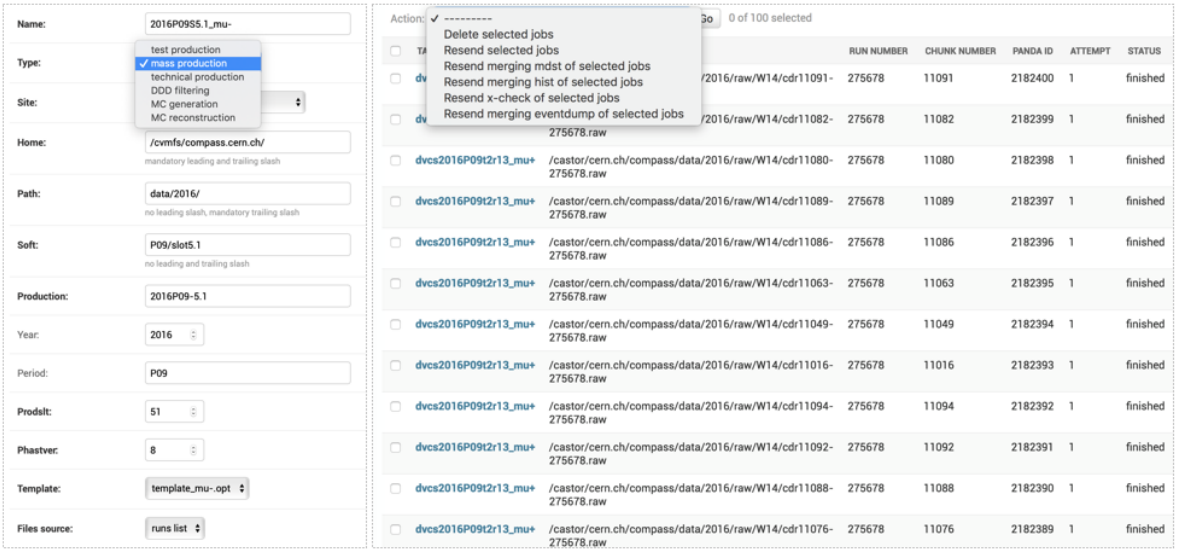


Рисунок 21 — Панель для формирования заданий COMPASS

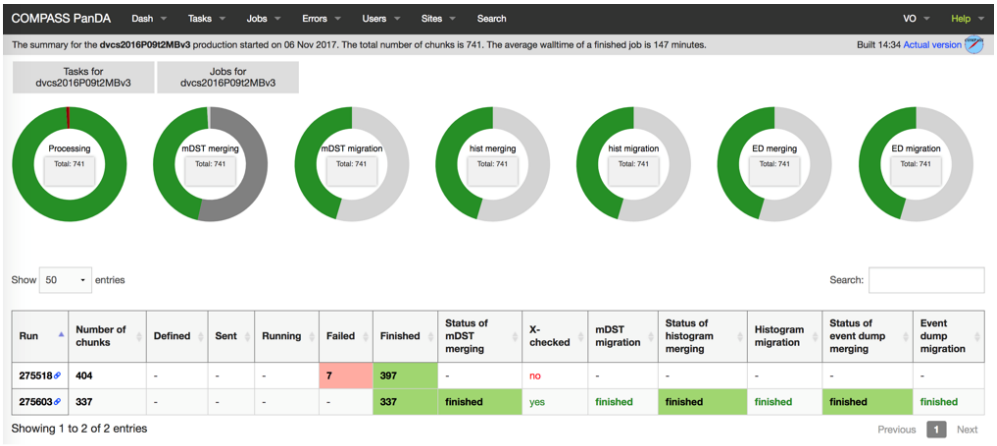


Рисунок 22 — Представление статистики

А с точки зрения выстроенной инфраструктуры и её компонентов:

- Сервер системы управления процессами обработки данных:
  - отдельный сервер СУБД MySQL для хранения информации о заданиях и составляющих их задачах;
  - расширенное приложение Django Admin для управления процессами обработки данных;
  - приложение для управления очередями в PanDA, оформленное как Django-приложение;
  - набор управляющих сервисов, обеспечивающих сопровождение процессов обработки данных.
- Система PanDAWMS:
  - непосредственно сервер PanDA;

    - отдельный сервер СУБД MySQL для хранения информации о заданиях в PanDA;
    - сервис Auto Pilot Factory;
    - веб-приложение PanDA Cache.
- Приложение PanDA Monitor, приспособленное к работе в контексте эксперимента COMPASS.

Общий анализ системы показал, что она во многом схожа с рассмотренной ранее системой эксперимента ATLAS, но использует собственное прикладное программное обеспечение для анализа в лице Coral и Phast, а также не имеет системы управления данными, что привело к существенному усложнению системы управления.

3 АНАЛИЗ ТЕХНОЛОГИЙ ДЛЯ ПОСТРОЕНИЯ СИСТЕМЫ

Проанализировав поставленные перед системой требования, обусловленные областью её применения, а также рассмотрев аналогичные системы, которые успели зарекомендовать себя в рамках других масштабных физических экспериментов, была выработана следующая схема составляющих.

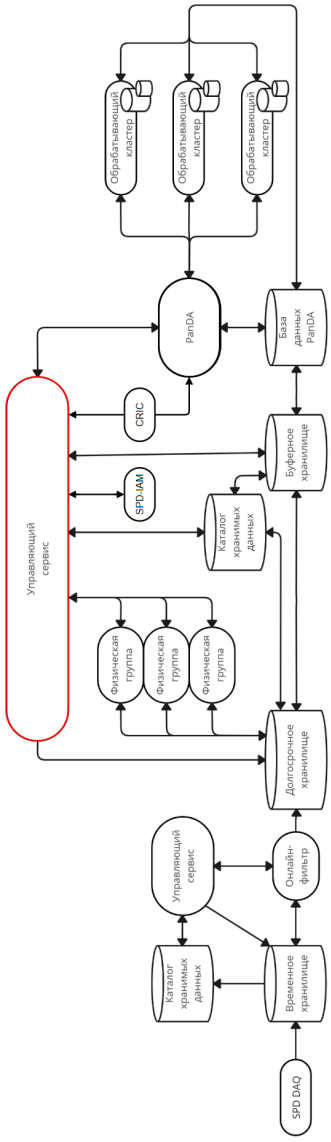


Рисунок 23 — Схема IT-составляющих

3.1 ВЫБОР ФРЕЙМВОРКА ДЛЯ ПРИЛОЖЕНИЯ

В ходе работы рассматривались два фреймворка, для реализации веб-приложение. Наиболее популярными в данной сфере решениями являются Django и Angular, ниже приведены их ключевые особенности.

Django

- MVT-архитектура (Model View Template):
  - разделение обязанностей;
  - повторное использование кода;
  - масштабируемость.
- ORM (Object Relational Mapper):
  - оптимизация часто используемых операций с базой данных;
  - экранирование параметров запросов;
  - автоматическая обработка подключений к базе данных и управление соединениями;
  - поддержка миграций.
- Реализованный admin-интерфейс.
- Встроенные функции обеспечения безопасности:
  - аутентификация и управление сессиями пользователей;
  - защита от CSRF (Cross-site Request Forgery);
  - автоматическая обработка подключений к базе данных и управление соединениями;
  - защита от XSS (Cross-site Scripting);
  - защита от SQL-инъекций;
  - управление паролями;
  - ограничение доступа на уровне представлений;
  - HTTPS (HyperText Transfer Protocol Secure).

Angular

- MVT-архитектура (Model View Template):
  - разделение обязанностей;
  - повторное использование кода;

    - масштабируемость.
- SPA (Single Page Application):
    - маршрутизация и навигация между компонентами;
    - поддержка Lazy Loading для оптимизации загрузки приложений.
- Dependency Injection.
- HTTP клиент и обработка запросов:
    - встроенный HTTP клиент для выполнения запросов к серверу;
    - интерцепторы для обработки HTTP запросов и ответов.
- Формы и валидация:
    - модуль ReactiveForms для создания и валидации форм;
    - встроенные механизмы валидации данных на стороне клиента.
- Тестирование:
    - модульное тестирования компонентов и сервисов;
    - инструменты для автоматизации тестирования.
- Встроенные функции обеспечения безопасности:
    - защита от CSRF (Cross-site Request Forgery);
    - защита от XSS (Cross-site Scripting);
    - HTTPS (HyperText Transfer Protocol Secure).

Из приведенных выше особенностей видится, что для нашей задачи разумнее будет использовать фреймворк Django, ввиду более широких возможностей обеспечения безопасности, встроенного функционала для работы с базами данных и инструментов аутентификации.

Однако в случае необходимости внедрения широких возможностей для взаимодействия с пользователями в будущем можно рассмотреть совместное использование фреймворков для обеспечения наилучшей реализации веб-приложения: Django для серверной части и Angular для клиентской. Это позволит разделить логику приложения на независимые части, что облегчит разработку, тестирование и поддержку, а также позволит использовать все преимущества обоих фреймворков.

3.2 OAUTH 2.0

OAuth 2.0 – это открытый протокол авторизации, разработанный для безопасного делегирования доступа к защищённым ресурсам без передачи учётных

данных пользователя сторонним приложениям. Основной документ, регламентирующий работу протокола, представлен в RFC 6749 [28]. В данной подглаве приведён обзор ключевых концепций и механизмов протокола, необходимых для общего понимания его работы и использования в рамках реализации разрабатываемой системы.

Протокол был разработан, чтобы удовлетворить следующие задачи:

- **безопасное делегирование доступа**: создает возможность предоставления доступа к закрытым ресурсам без хранения данных для аутентификации пользователя;
- **гибкость и расширяемость**: предоставляет механизм, легко адаптируемый к различным сценариям; в нашем случае - связывает все системы в общей программной экосистеме эксперимента;
- **управление полномочиями**: разделяет процесс аутентификации и авторизации;
- **минимизация рисков**: снижает вероятность проблем безопасности, связанных с передачей и хранением учётных данных;
- **бесшовность взаимодействий**: снимает с пользователя необходимость частого входа в аккаунты и скрывает соответствующие переходы.

Для обеспечения описанных выше возможностей система реализует потоки сообщений и сами сообщения определенным образом, для понимания которого сначала приведем основные термины для этой подглавы.

Роли:

- **Resource Owner** (Владелец ресурса)- субъект, который имеет возможность предоставить доступ к защищённому ресурсу. Чаще всего это конечный пользователь, который владеет данными. В нашем случае - это пользователи и физические группы, которые владеют правами на использование вычислительных мощностей и просмотр информации о заданиях.
- **Client** (Клиент) - приложение, которое запрашивает доступ к ресурсам ресурсного владельца от имени последнего. Для работы с доступом клиент оперирует специальными токенами доступа (access tokens). В нашем случае клиент - часть веб-приложения, запрашивающая, хранящая и использующая токены доступа.
- **Authorization Server** (Сервер авторизации) - доверенный сервер, который проводит процесс аутентификации и выдаёт клиенту токены. Сервер

авторизации также хранит сведения о предоставленных разрешениях, выпущенных токенах и их метаданные. Само название связано с тем, что сервер выпускает токены для авторизации, а в нашем случае таким сервером выступает SPD-IAM, являющейся единой точкой входа для всех сервисов, обслуживающих эксперимент SPD.

- **Resource Server** (Сервер с ресурсом) - сервер, на котором находятся защищённые ресурсы. Принимает токен доступа для проверки прав клиента и последующего предоставления доступа. Сервер авторизации и сервер с ресурсом могут быть реализованы в рамках одной системы, как, отчасти, в нашем случае, но с логической точки зрения это разные роли. В разрабатываемой системе такими серверами выступают: часть разрабатываемого сервиса, сервис управления данными, сервис хранения топологии системы и сервис обработки заданий.

  И основные понятия:

- **Access Token** (Токен доступа) - короткоживущий токен, выдаваемый клиенту, с помощью которого он может получить доступ к защищённым ресурсам. Содержит закодированную информацию о разрешениях, собственном времени жизни и прочие метаданные. Токен доступа обеспечивает уровень абстракции и заменяет различные конструкции авторизации единым токеном, понятным серверу с ресурсами. Такая абстракция позволяет выдавать токены доступа, обладающие более ограниченными правами по сравнению с разрешением, которое можно было бы получить, используя данные, предоставленные пользователем серверу авторизации, и избавляет сервер с ресурсами от необходимости поддерживать множество способов аутентификации.
- **Refresh Token** (Рефреш-токен) - долгоживущий токен, который может использоваться клиентом для получения нового токена доступа.
- **Scope** (Разрешения) - строка или множество строк, определяющих защищённые ресурсы, на которую предоставляется доступ. Разрешения помогают точно задать, к каким именно данным или операциям клиенту разрешено обращаться.
- **Authorization Grant** - сценарий передачи учетных данных, представляемых владельцем ресурса и разрешающих доступ к соответствующим защищённым ресурсам. Рассматриваемый протокол определяет несколько

стандартных методов для реализации этого процесса:

- **Authorization Code** (Код авторизации) - метод, использующий специальный код, используемый при коммуницировании с участием сервера авторизации, выступающего посредником между клиентом и владельцем ресурса. Суть метода в том, чтобы вместо запрашивания разрешения непосредственно у владельца ресурса, клиент перенаправляет владельца ресурса на сервер авторизации; далее сервер авторизации выполняет аутентификацию владельца ресурса и получает у него соответствующие разрешения; затем сервер авторизации перенаправляет владельца ресурса обратно к клиенту вместе с кодом авторизации, который владелец ресурса и предоставляет клиенту. Поскольку владелец ресурса аутентифицируется только перед сервером авторизации, его учётные данные никогда не передаются клиенту, что повышает безопасность и уменьшает нагрузку на сервер, который хранит ресурсы. После получения клиентом кода авторизации он обменивает данный код на токены у сервера авторизации. Последний обмен происходит напрямую, т.е. без участия владельца ресурсов, что предотвращает возможность перехвата получаемых токенов как самим владельцем, так и третьими лицами. Данный подход является рекомендованным к использованию и выбран как единственный для реализуемой системы.
- **Implicit** - метод, представляющий собой упрощённый вариант использования кода авторизации. Вместо выдачи клиенту кода авторизации клиент получает непосредственно токен доступа. В таком случае промежуточные учётные данные не выдаются и не используются для получения. При выдаче токена доступа сервер авторизации не выполняет аутентификацию клиента, хотя в некоторых случаях подлинность клиента может проверяться по URI перенаправления, используемому для доставки токена доступа клиенту. Токен доступа при этом может оказаться доступным владельцу ресурса или другим приложениям, имеющим доступ к агенту (например, браузеру) владельца ресурса. Отсутствие промежуточных переходов и обменов, разумеется, упрощает работу и улучшает отклик приложения, но, очевидно, несет проблемы, связанные с безопасностью.

- **Resource Owner Password Credentials** (Учетные данные владельца ресурса) - метод, где учетные данные владельца ресурса могут быть использованы напрямую для получения токена доступа. Хотя данный способ требует от клиента прямого доступа к учетным данным владельца ресурса, эти данные используются лишь один раз для обмена на токен доступа. Таким образом, клиенту не нужно хранить учетные данные владельца ресурса для дальнейшего использования.
- **Client Credentials** (Учетные данные клиента) - метод, где используются только учетные данные клиента. Способ применим в тех случаях, когда область доступа ограничена защищёнными ресурсами, находящимися под контролем самого клиента, либо защищёнными ресурсами, заранее согласованными с сервером авторизации; зачастую предполагается доступ клиентом, действующим от собственного имени.
- **Extension Grants** - также существует возможность реализации собственных методов.

- **Client Credentials** (Учетные данные клиента) - Пара строк «client_id/client_s которые используются клиентом для обеспечения безопасности работы с токенами.

Взаимодействия по соответствующему протоколу с использованием кода авторизации приведено на рисунке 24

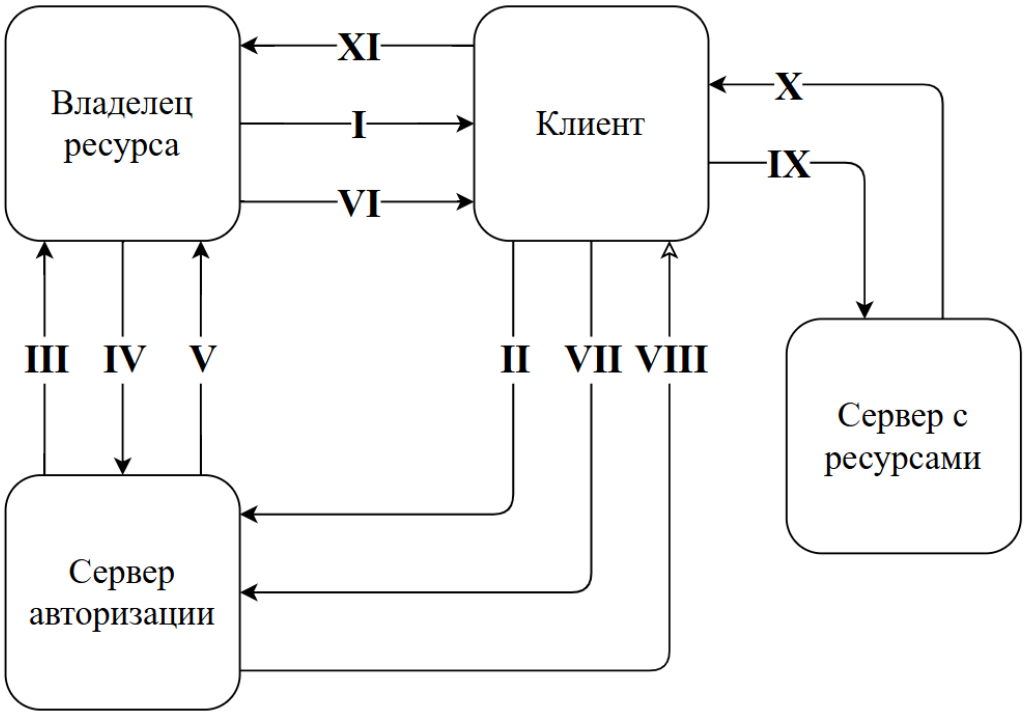


Рисунок 24 — Взаимодействие между ролями по протоколу OAuth 2.0 с использованием Authorization Code

Шаги взаимодействия:

I. Попытка доступа владельца ресурсов к защищенным ресурсам без соответствующих валидных токенов.
II. Перенаправление владельца ресурса на сервер авторизации.
III. Запрос аутентификации.
IV. Пользователь предоставляет свои учетные данные.
V. Сервер авторизации отдает владельцу ресурсов код авторизации.
VI. Владелец ресурсов передает код авторизации клиенту.
VII. Клиент отправляет код авторизации и свои учетные данные серверу авторизации.
VIII. Сервер авторизации выпускает токены и отдает их клиенту.
IX. Клиент отправляет токен доступа серверу с ресурсами.
X. Сервер с ресурсами предоставляет соответствующие ресурсы в соответствии с токеном доступа.
XI. Клиент предоставляет пользователю запрошенные ресурсы.

Примеры соответствующих сообщений будут приведены в главе, посвященной реализации механизма предоставления прав в рамках разрабатываемой системы.

3.3 JWT

В прошлой подглаве мы рассмотрели протокол, позволяющий внедрить сквозную авторизацию во многих системах. В процессе рассмотрения были упомянуты токены, ответственные за непосредственное хранение разрешений и обеспечение безопасного доступа. Одним из вариантов таких токенов являются JWT, которым и будет посвящена данная подглава.

JSON Web Token (JWT) [29] — это компактный URL-безопасный формат представления передаваемой между двумя сторонами информации в виде JSON-объекта, который может быть определённым образом подписан и/или зашифрован.

Использование JWT может поспособствовать решению следующих задач:

- **Передача доказательства аутентификации**: часто может использоваться в классической stateless-схеме, где клиент хранит токен самостоятельно и передаёт его с каждым запросом. Это позволяет повсеместно подтверждать права доступа, одновременно снимая часть нагрузки с сервера и способствуя его горизонтальной масштабируемости. Однако в реализуемом сервисе мы не применяем stateless-подход. Соответственно, JWT сохраняется в сессионном хранилище; при каждом запросе токен извлекается из сессии и проверяется его валидность. Такое хранение, с одной стороны, обеспечивает самодостаточность JWT как носителя набора доступных прав, а с другой - добавляет централизованный контроль: при необходимости можно немедленно удалить информацию о сессии, тем самым предотвратить дальнейшее использование скомпрометированного токена.
- **Передача заявленных прав**: JWT содержит набор утверждений (claims) о пользователе; такие утверждения могут быть публичными, приватными и заранее зарезервированными спецификацией. В главе, посвящённой авторизации и аутентификации, будут затронуты приватные утверждения, используемые в нашей системе; сейчас же рассмотрим только предусмотренные стандартом:
  - **iss** (издатель токена) - поле идентифицирует субъект, выпустивший JWT. Обработка этого поля обычно зависит от конкретного приложения. Значение хранимое в поле представляет собой строку формата

StringOrURI [29], чувствительную к регистру;

- **sub** (субъект токена) - поле идентифицирует субъект, которому выдан данный JWT. Как правило, утверждения в токене содержат информацию именно об этом субъекте. Значение в поле должно быть уникальным в контексте издателя. Проще говоря, это идентификатор пользователя в системе, выпускающей токены. Как и iss, обработка sub не стандартизирована и зависит от конкретного приложения, а их значения имеют одни и те же особенности;
- **aud** (получатели токена) - поле определяет получателей, для которых предназначен JWT. В общем случае значение в поле — это массив строк StringOrURI, чувствительных к регистру. Системы, намеренные обрабатывать данный токен, должны определить себя значением в этом массиве;
- **exp** (время истечения срока действия токена) - поле определяет момент, после наступления которого JWT должен считаться невалидным; хотя, иногда, допустима небольшая поправка, с целью учёта рассогласованности системных часов. Само значение должно быть числом в формате NumericDate [29];
- **nbf** (время активации токена) - поле, идентичное exp, с той только разницей, что оно задает время после которого токен считается валидным;
- **iat** (время выпуска токена) - поле, также похожее, и хранящее время создания токена.

- **Универсальность и переносимость**: JWT-формат является совершенно независимым от конкретного языка программирования или среды выполнения. Будучи основанным на стандартах JSON [30] и Base64 [31] он широко совместим с множеством клиентских библиотек.
- **Поддержка распределённых систем и микросервисов**: любой микросервис или сторонний элемент единой информационной системы, получив валидный токен, может извлечь из него информацию о пользователе, правах и привязанных данных, не обращаясь за этим к центральному сервису аутентификации.

Как уже было сказано, JWT может быть подписан и/или зашифрован. Строго говоря, согласно стандарту, JWT формируется на основе структуры

либо JWS (JSON Web Signature) [32], либо JWE (JSON Web Encryption) [33]. JWS обеспечивает целостность и аутентичность токена посредством цифровой подписи. Именно подписанные JWT (реализованные как JWS) наиболее распространены: сервер создаёт подписанный токен, а получатели верифицируют, что содержимое не было изменено третьими лицами. В нашем сервисе используются именно такие подписанные JWT; в последующих главах, посвящённых реализации сервиса, термины "токен" и "JWT" следует воспринимать именно как JWS, содержащий в качестве полезной нагрузки структурированные данные в формате JWT. Сами данные в таком случае никак не сокрыты, поэтому могут быть получены простым декодированием токена. JWE, напротив, обеспечивает конфиденциальность содержимого токена, скрывая полезную нагрузку; что может быть необходимо при передаче чувствительной информации, однако в рамках реализуемой системы это свойство не востребовано, поэтому не будем глубоко погружаться в этот аспект.

Подписанный токен представляет собой строку из трёх сегментов в формате Base64Url, разделенных точками:

header.payload.signature

- **header** (заголовок) - JSON-объект, описывающий криптографические операции, применяемые к JWT, а также информацию о содержимом, например, тип хранящегося токена для случаев, если принимающая сторона ожидает не только JWT. В обязательном порядке объект должен содержать поле "alg" указывающее на алгоритм, использованный для подписи. Сам JSON кодируется, и результат является первой частью токена;
- **payload** (полезная нагрузка) - также JSON-объект, хранящий набор утверждений, описанных выше, в формате ключ-значение. Важно упомянуть, что для этого сегмента нет обязательных полей. JSON также кодируется, в результате чего получается вторая часть токена;
- **signature** (подпись) - третья часть токена обеспечивает его целостность и аутентичность. Для её получения выполняется следующая процедура:
  - формируется строка из закодированных header и payload, разделенных точкой;
  - к полученной строке применяется криптографический алгоритм, указанный в поле alg;

- результат работы алгоритма кодируется и добавляется в виде последнего сегмента токена.

В спецификации также приведен набор рекомендованных к использованию алгоритмов, таких как HMAC (HSxxx) и RSASSA-PKCS1-v1_5 (RSxxx) [34] с хешированием SHA-256, SHA-384 и SHA-512. В целях повышения безопасности в информационной системе эксперимента SPD используются алгоритмы с асимметричным шифрованием, в частности RS256. При его использовании сервер-издатель создаёт цифровую подпись, применяя свой приватный RSA-ключ к хэшу заголовка и полезной нагрузки токена. Любой проверяющий может убедиться в подлинности и целостности JWT, используя соответствующий публичный ключ.

3.4 APACHE2

Apache2 – это свободный открытый кроссплатформенный веб-сервер, быстро набравший популярность и на данный момент являющийся одним из самых популярных инструментов за счёт своей надёжности, гибкости и развитой модульной архитектуры. Основное назначение Apache – приём HTTP-запросов от клиентов и выдача им веб-контента с учётом конфигурации сервера и виртуальных хостов. В качестве выдаваемых ресурсов могут выступать как статические, так и динамические страницы.

Как уже было упомянуто, Apache построен на модульной архитектуре: ядро сервера (core) реализует базовые функции, а подключаемые модули расширяют функциональность по мере необходимости, причем, как правило, модули узконаправленные и решают строго одну определенную задачу. Такая особенность позволяет включать или отключать отдельный функционал без изменения кода ядра.

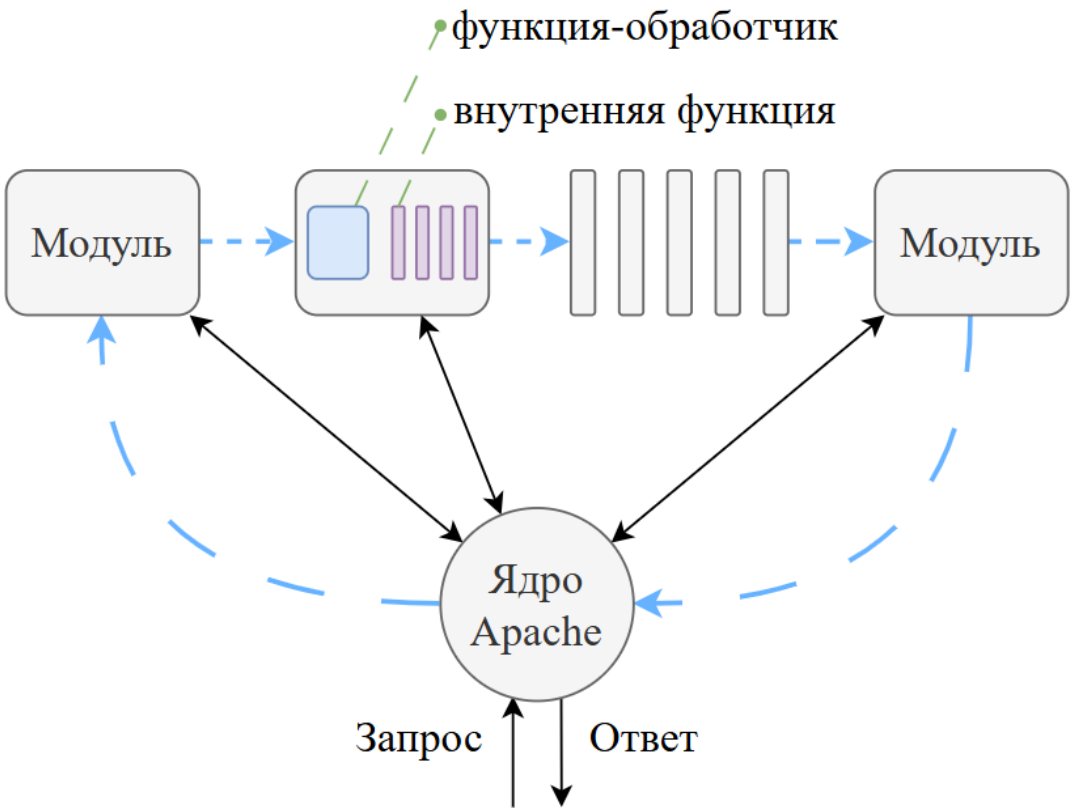


Рисунок 25 — Архитектура Apache (черные линии - фактический поток, голубые - логический)

Работа сервера основана на модели многопроцессной обработки запросов (Multi-Processing Modules, MPM). Доступно использование различных MPM-модулей, таких как:

- prefork, когда каждый запрос обрабатывается отдельным процессом;
- worker, в котором существует несколько процессов, и каждый из них имеет свой пул потоков;
- event, добавляющий слушающий поток в каждый процесс для освобождения рабочих потоков.

В современных системах зачастую используется последний из перечисленных MPM-модулей, и наша система в этом вопросе - не исключение. Поэтому приведем простую диаграмму и кратко опишем, как работает Apache при использовании модуля event.

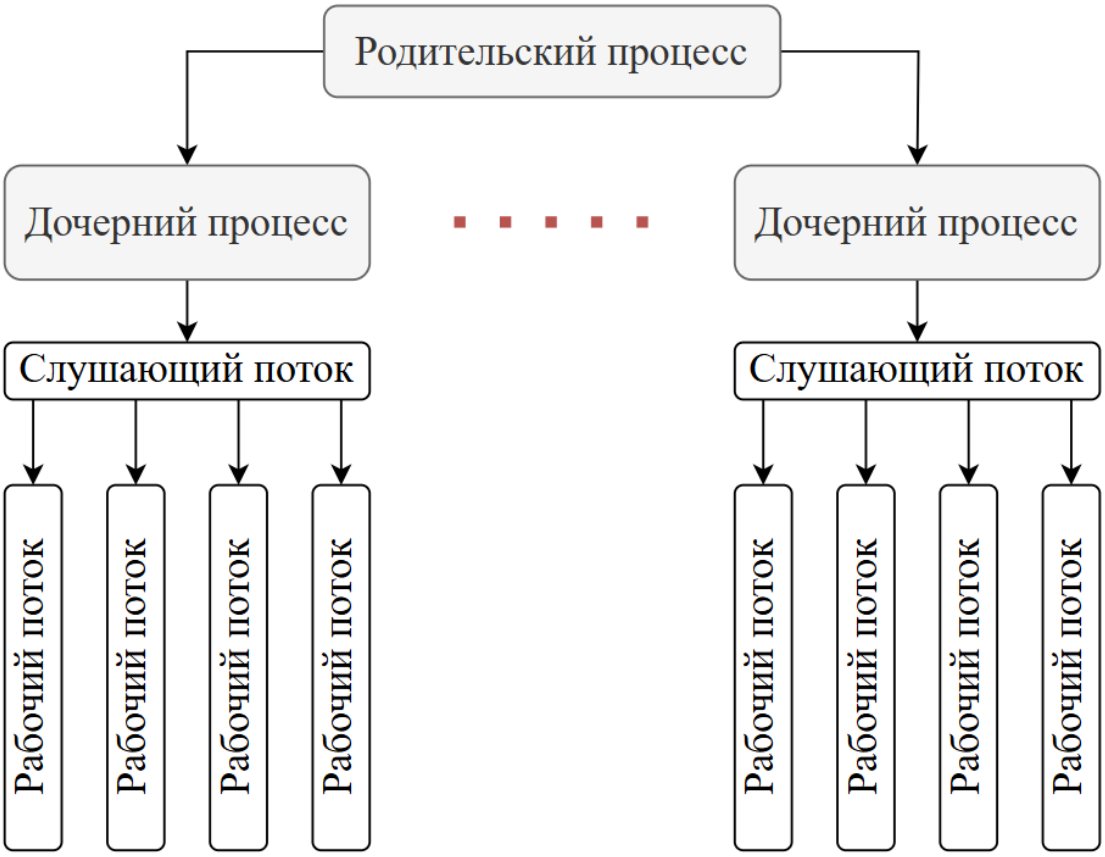


Рисунок 26 — Структура процессов и потоков в Apache

1) Инициализация родительского процесса:
   - считывается основной "/etc/httpd/conf/httpd.conf" и дополнительные "/etc/httpd/conf.d/*.conf" конфигурационные файлы;
   - загружаются необходимые модули;
   - по директиве Listen 443 создается сокет, привязанный к комбинации адрес:порт.
2) Создание рабочих процессов:
   - родительский процесс порождает рабочие процессы в количестве, заданном директивой StartServers;
   - каждый рабочий процесс наследует открытый ранее сокет.
3) Запуск пула потоков в рабочих процессах:
   - в каждом рабочем процессе запускается пул потоков;
   - общее число потоков задано директивой ThreadsPerChild;
   - один или несколько потоков становятся слушающими (принимают новые соединения и отслеживают поддерживаемые) , остальные - рабочими (непосредственно обрабатывают поступающий трафик);
   - слушающие потоки используют механизм неблокирующего ввода-

вывода.

Теперь слушающие потоки находятся в режиме ожидания подключений, и наш веб-сервис готов к работе.

После того, как слушающий поток обнаруживает новое TCP-соединение, он выполняет вызов `accept()`, тем самым получая это соединение и создавая новый сокет; после чего управление передаётся одному из рабочих потоков, который выполняет TLS рукопожатие, и далее занимается непосредственно разбором HTTP-заголовков, выполнением маршрутизации и генерацией ответа. Такой подход позволяет эффективно разграничивать ответственность потоков: слушающие потоки минимально заняты, что важно для своевременного приёма новых соединений, а рабочие потоки сосредоточены на выполнении более ресурсоёмких операций.

4 РАЗРАБОТКА ПРОТОТИПА ВЕБ-ПРИЛОЖЕНИЯ

В ходе работы был изучен фреймворк Django, с использованием которого реализовано веб-приложение для создания, управления и отслеживания заданий на обработку.

Средствами ORM-слоя реализованы модели заданий (общая, Simulation, Reconstruction), формы для создания соответствующих заданий и их цепочек, что позволяет собирать необходимые параметры, формировать из этих параметров словарь и отправлять его в очередь PanDA для дальнейшей обработки.

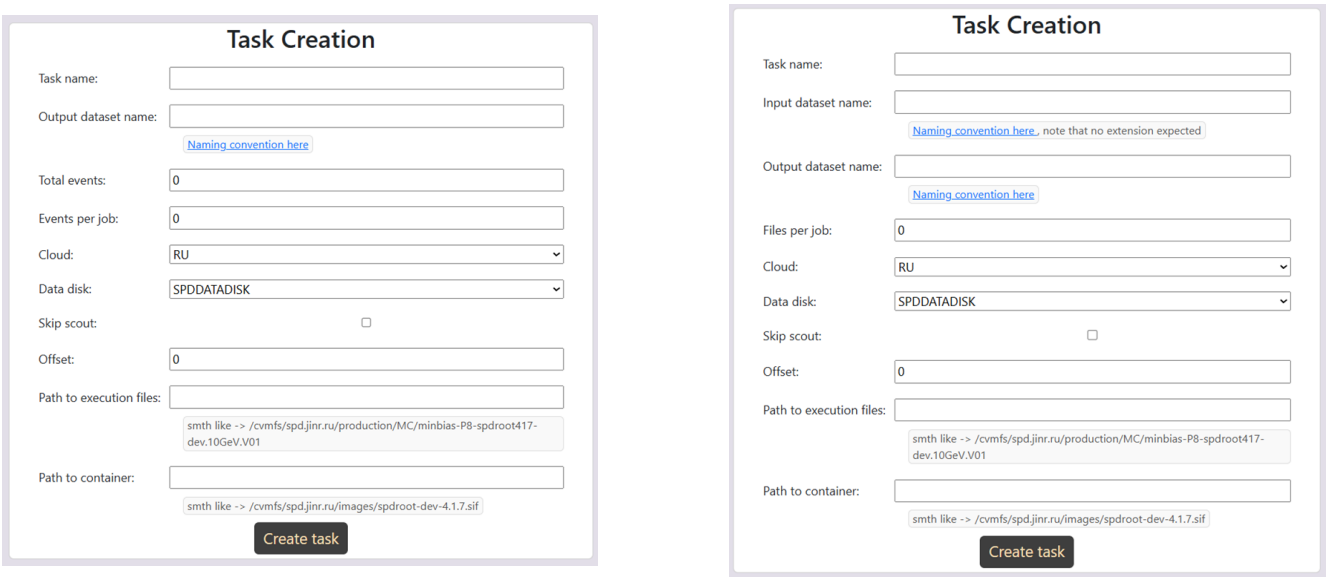


Рисунок 27 — Создание заданий Simulation(слева), Reconstruction(справа)

Внедрены средства просмотра всех заданий (листинг 1, рисунок 28), завершённых заданий (рисунок 29). Реализованы возможности сортировки и фильтрации, а для завершённых - дополнительно существует возможность выгрузки данных для отчётности в формате CSV. Для представленных таблиц реализованы модели на стороне сервера и представления на стороне базы данных.

```
@session_login_required
def show_all_tasks(request):
    fields = T_Task.get_fields_all()
    result = T_Task.objects.using('panda_db').select_related('jedi_task').
   values(*fields)
    filter_parameter = request.GET.get('taskField', None)
    sort_type = request.GET.get('sortType', 'sortTaskIdDown')
    page_num = request.GET.get('pageNum', 1)
    task_value = request.GET.get('taskValue')

    if filter_parameter:
        result = result.filter(**{filter_parameter + '__icontains': request.GET
   .get('taskValue','')})

    if sort_type and sort_type in sort_mapping:
        reverse_sort = sort_mapping.get(sort_type).startswith('-')
        sort_key = sort_mapping.get(sort_type).lstrip('-')
        result = sorted(result, key=lambda x: (x.get(sort_key) is None, x.get(
   sort_key), x.get('taskid')), reverse=reverse_sort)

    paginator = Paginator(result, 10)
    thisPage = paginator.get_page(page_num)
    len = paginator.get_elided_page_range(number=page_num, on_each_side = 2,
   on_ends=3)
    return render(request, 'pilot/pilot-main.html',
            {"len": len,
             "thisPage": thisPage,
             "thisPageNumber": page_num,
             "task_fields": fields,
             "sortType": sort_type,
             "taskField": filter_parameter,
             "taskValue": task_value})
```

Листинг 1 — Отображение всех заданий

Select field | Enter value | Filter

Task ID	Task name ↑ ↓	Parent ID	Creator	Status	Done jobs	Default/Current priority	Total events	Submit time ↑ ↓	Start time ↑ ↓	End time ↑ ↓
370	PROD2025-017.SIM	370	Elena Zemlyanichkina	running	2	900/900	0	17:03, 20 May 2025	17:11, 20 May 2025	None
369	PROD2025-016.RECO	369	Elena Zemlyanichkina	finished	9999	900/900	0	03:31, 10 May 2025	08:33, 10 May 2025	22:12, 13 May 2025
368	PROD2025-016.SIM	368	Elena Zemlyanichkina	done	10000	900/900	0	15:08, 08 May 2025	15:16, 08 May 2025	22:07, 09 May 2025
367	PROD2025-015.RECO	367	Elena Zemlyanichkina	finished	9992	900/900	0	13:16, 07 May 2025	13:24, 07 May 2025	18:58, 09 May 2025
366	PROD2025-015.SIM	366	Elena Zemlyanichkina	done	10000	900/900	0	04:26, 06 May 2025	07:01, 06 May 2025	12:52, 07 May 2025
365	PROD2025-014.RECO	365	Elena Zemlyanichkina	finished	9991	900/900	0	23:13, 04 May 2025	23:21, 04 May 2025	20:37, 06 May 2025
364	PROD2025-014.SIM	364	Elena Zemlyanichkina	done	10000	900/900	0	14:21, 03 May 2025	14:28, 03 May 2025	21:35, 04 May 2025
363	PROD2025-013.RECO	363	Elena Zemlyanichkina	finished	9992	900/900	0	10:03, 02 May 2025	10:11, 02 May 2025	00:51, 04 May 2025
362	PROD2025-013.SIM.2	362	Elena Zemlyanichkina	done	10000	900/900	0	23:35, 30 Apr 2025	23:43, 30 Apr 2025	08:57, 02 May 2025
361	PROD2025-013.SIM	361	Elena Zemlyanichkina	aborted	869	900/900	0	10:59, 30 Apr 2025	11:10, 30 Apr 2025	15:13, 30 Apr 2025

Page: 1 2 3 ... 27 28 29

Рисунок 28 — Задания, отсортированные по убыванию Task Id

Select field | Enter value | Filter

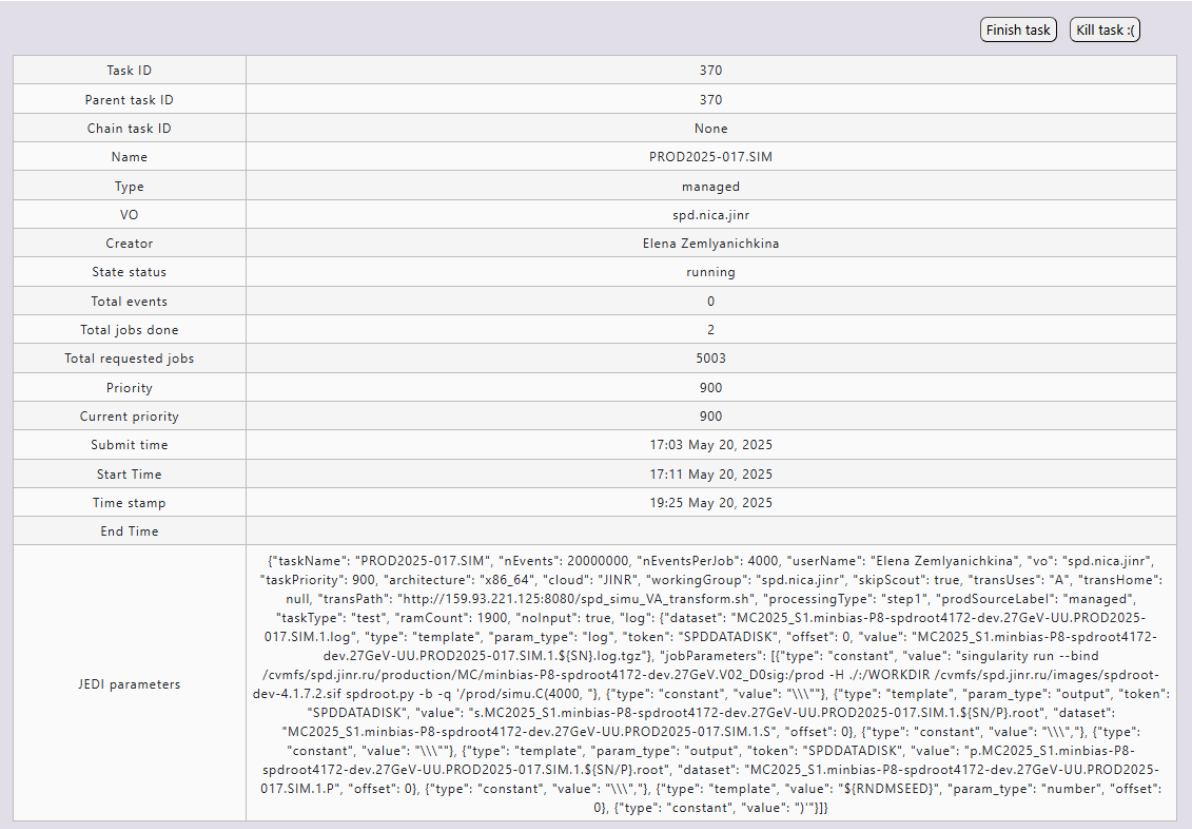
Task ID	Task name ↑ ↓	Status	Start date	End date	Walltime	Total events	Events per job	Total jobs	Out DS size, GB	Out Log size, GB
369	PROD2025-016.RECO	finished	10 May 2025	13 May 2025	30	None	None	9999	18557.43	5.07
368	PROD2025-016.SIM	done	08 May 2025	09 May 2025	22931	40000000	4000	10000	18365.39	1.92
367	PROD2025-015.RECO	finished	07 May 2025	09 May 2025	23	None	None	9992	18543.30	5.05
366	PROD2025-015.SIM	done	06 May 2025	07 May 2025	24486	40000000	4000	10000	18362.72	1.97
365	PROD2025-014.RECO	finished	04 May 2025	06 May 2025	24	None	None	9991	18540.20	5.05
364	PROD2025-014.SIM	done	03 May 2025	04 May 2025	24612	40000000	4000	10000	18358.86	1.96
363	PROD2025-013.RECO	finished	02 May 2025	04 May 2025	20	None	None	9992	18536.20	5.06
362	PROD2025-013.SIM.2	done	30 Apr 2025	02 May 2025	24899	40000000	4000	10000	18357.95	1.93
359	PROD2025-012.RECO	finished	28 Apr 2025	29 Apr 2025	24	None	None	9993	18546.74	5.08
358	PROD2025-012.SIM	done	25 Apr 2025	26 Apr 2025	23316	40000000	4000	10000	18360.80	1.89

Page: 1 2 3 4 5 6 7

Export csv

Рисунок 29 — Завершенные задания

Возможен просмотр данных о конкретном задании (рисунок 30), его предварительное завершение или отмена.



Finish task | Kill task :(

Task ID	370
Parent task ID	370
Chain task ID	None
Name	PROD2025-017.SIM
Type	managed
VO	spd.nica.jinr
Creator	Elena Zemlyanichkina
State status	running
Total events	0
Total jobs done	2
Total requested jobs	5003
Priority	900
Current priority	900
Submit time	17:03 May 20, 2025
Start Time	17:11 May 20, 2025
Time stamp	19:25 May 20, 2025
End Time	
JEDI parameters	{"taskName": "PROD2025-017.SIM", "nEvents": 20000000, "nEventsPerJob": 4000, "userName": "Elena Zemlyanichkina", "vo": "spd.nica.jinr", "taskPriority": 900, "architecture": "x86_64", "cloud": "JINR", "workingGroup": "spd.nica.jinr", "skipScout": true, "transUses": "A", "transHome": null, "transPath": "http://159.93.221.125:8080/spd_simu_VA_transform.sh", "processingType": "step1", "prodSourceLabel": "managed", "taskType": "test", "ramCount": 1900, "noInput": true, "log": {"dataset": "MC2025_S1.minbias-P8-spdroot4172-dev.27GeV-UU.PROD2025-017.SIM.1.log", "type": "template", "param_type": "log", "token": "SPDDATADISK", "offset": 0, "value": "MC2025_S1.minbias-P8-spdroot4172-dev.27GeV-UU.PROD2025-017.SIM.1.${SN}.log.tgz"}, "jobParameters": [{"type": "constant", "value": "singularity run --bind /cvmfs/spd.jinr.ru/production/MC/minbias-P8-spdroot4172-dev.27GeV.V02_D0sig:/prod -H ./:/WORKDIR /cvmfs/spd.jinr.ru/images/spdroot-dev-4.1.7.2.sif spdroot.py -b -q '/prod/simu.C(4000, "}, {"type": "constant", "value": "\\\""}, {"type": "template", "param_type": "output", "token": "SPDDATADISK", "value": "s.MC2025_S1.minbias-P8-spdroot4172-dev.27GeV-UU.PROD2025-017.SIM.1.${SN/P}.root", "dataset": "MC2025_S1.minbias-P8-spdroot4172-dev.27GeV-UU.PROD2025-017.SIM.1.S", "offset": 0}, {"type": "constant", "value": "\\\","}, {"type": "constant", "value": "\\\""}, {"type": "template", "param_type": "output", "token": "SPDDATADISK", "value": "p.MC2025_S1.minbias-P8-spdroot4172-dev.27GeV-UU.PROD2025-017.SIM.1.${SN/P}.root", "dataset": "MC2025_S1.minbias-P8-spdroot4172-dev.27GeV-UU.PROD2025-017.SIM.1.P", "offset": 0}, {"type": "constant", "value": "\\\","}, {"type": "template", "value": "${RNDMSEED}", "param_type": "number", "offset": 0}, {"type": "constant", "value": ")'"}]}

Рисунок 30 — Информация о задании

При работе с PandaWMS на данный момент задания могут находиться в следующих состояниях, выраженных полем "Status":

- **registered**: Задание было успешно доставлено и зарегистрировано в БД Panda;
- **defined**: Параметры задания были успешно распознаны ;
- **running**: Задание в процессе порождения и выполнения задач;
- **paused**: Процесс выполнения приостановлен;
- **aborted**: Задание было отменено;
- **failed**: Задание завершилось, но доля успешно выполненных заданий составила менее 90%;
- **finished**: Задание завершилось, доля успешно выполненных заданий составила от 90% до 100%;
- **done**: Задание завершилось полностью.

5 СОЗДАНИЕ ВЕБ-СЕРВЕРА

Создание веб-сервера осуществлено посредством серверного ПО Apache2, рассмотренного в части 3.4, и связанного с проектом через mod_wsgi.

Apache2 используется для обработки HTTP-запросов и отправки ответов на них пользователю. Основные параметры процесса Apache приведены ниже:

- Server MPM: event;
- StartServers: 2;
- MinSpareThreads: 25;
- MaxSpareThreads: 75;
- ThreadsPerChild: 25;
- ServerLimit: 16;
- MaxRequestWorkers: 400;
- MaxConnectionsPerChild: 0.

Также был создан конфигурационный файл, обеспечивающий корректную работу и включающий такие параметры, как: пути к статичным и медиа файлам, настройка необходимого виртуального окружения, использование HTTPS и так далее. Кроме того, добавлен отдельный конфигурационный файл для смены протокола HTTP $\rightarrow$ HTTPS, что гарантирует защищенное соединение даже при попытке клиента использовать незащищенный протокол.

Создана виртуальная машина на AlmaLinux OS 9, после настройки веб-сервер будет функционировать на ней. Контроль версий будет осуществляться посредством git-репозитория ОИЯИ git.jinr.ru [35].

```
<VirtualHost *:443>
    Основные настройки
    ServerName service-name.jinr.ru
    ServerAlias service-name.jinr.ru
    DocumentRoot /path/to/app

    Логирование
    ErrorLog /var/log/app/error.log
    CustomLog /var/log/app/access.log combined

    Настройки SSL
    SSLEngine on
    SSLCertificateFile /path/to/cert.pem
    SSLCertificateKeyFile /path/to/key.pem

    Интеграция WSGI
    WSGIScriptAlias / /path/to/app/wsgi.py
    WSGIDaemonProcess app \
        python-home=/path/to/venv \
        python-path=/path/to/app
    WSGIProcessGroup app

    Настройки доступа
    <Directory /path/to/app>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    Статические файлы
    Alias /static /path/to/staticfiles
    <Directory /path/to/staticfiles>
        Require all granted
    </Directory>

     Медиа файлы
    Alias /media /path/to/media
    <Directory /path/to/media>
        Require all granted
    </Directory>

</VirtualHost>
```

Листинг 2 — Конфигурация виртуального хоста Apache

6 МЕХАНИЗМ АУТЕНТИФИКАЦИИ И АВТОРИЗАЦИИ

В целях обеспечения сквозной аутентификации и, как следствие, бесшовного использования сервиса в рамках общей системы была произведена интеграция с SSO-провайдером SPD-IAM (spd-iam.jinr.ru) (рисунок 31). Взаимодействие с сервисом авторизации происходит по протоколу OAuth 2.0 со сценарием Authorization Code Flow; все сообщения передаются через HTTPS-соединение. При передаче конфиденциальная информация содержится только в теле запроса, а на хранении - в переменных окружения или локальном хранилище сессий, что обеспечивает защиту функционала сервиса от несанкционированного использования. Получаемый при этом соединении JSON-объект содержит самодостаточный набор метаданных и нагрузки в виде JWT, позволяющий как гарантировать неподдельность пакета, так и управлять доступом к защищенным частям сервиса. При получении такого токена реализуемый сервис производит проверку его валидности (листинг 3). Содержание получаемого токена приведено в листинге 4

```
def base64_url_decode(data):
    padding = 4 - (len(data) % 4)
    data += '=' * padding
    return base64.urlsafe_b64decode(data)
--Извлечение публичных ключей--
with open(SPD-IAM_TOKEN_KEY, 'r') as keystore_file:
    keystore = json.load(keystore_file)

keys = keystore['keys']
n = int.from_bytes(base64_url_decode(keys[0]['n']), byteorder='big')
e = int.from_bytes(base64_url_decode(keys[0]['e']), byteorder='big')
public_key = rsa.RSAPublicNumbers(e, n).public_key()
--Извлечение токена доступа--
signed_jwt = request.session.get('access_token')
--Проверка токена и логгирование ошибок--
try:
    decoded_jwt = jwt.decode(signed_jwt, public_key, algorithms=["RS256"])
except Exception as e:
    logger = logging.getLogger('JWT_verify_logger')
    logger.error(e)
```

Листинг 3 — Проверка валидности JWT

```
  "header": {
    "kid": "rsa1",
    "alg": "RS256"
  },
  "payload": {
    "sub": "значение скрыто",
    "iss": "https://spd-iam.jinr.ru",
    "groups": [
      "spd.nica.jinr/production",
      "spd",
      "spd.nica.jinr",
      "production",
      "spd/production"
    ],
    "preferred_username": "monakov",
    "organisation_name": "SPD",
    "client_id": "значение скрыто",
    "nbf": 1750159094,
    "scope": "scim iam phone openid profile offline_access groups rucio email
   wlcg wlcg.groups",
    "name": "Nikita Monakov",
    "exp": 1750162694,
    "iat": 1750159094,
    "jti": "значение скрыто",
    "email": "значение скрыто"
  }
```


Листинг 4 — Содержание JWT

Часть полезной нагрузки помещается в сессионное хранилище для обеспечения работоспособности системы, а именно:

- userName - для отображения имени пользователя;
- loged_in_timestamp | expiration_timestamp - для контроля времени предоставления доступа;
- production_member - для контроля возможности создания заданий;
- id_token - предоставляется стороне PanDA WMS для проверки доступа.

На текущий момент матрица прав доступа (таблица 5) достаточно простая, однако она полностью удовлетворяет текущим требованиям, а в случае необходимости её легко модифицировать посредством изменения реализованного промежуточного ПО и атрибутов сессии. Упомянутое ПО отвечает за контроль доступа к внутреннему функционалу сервиса, проверку срока действия токена доступа и его обновление при условии валидности рефреш-токена.

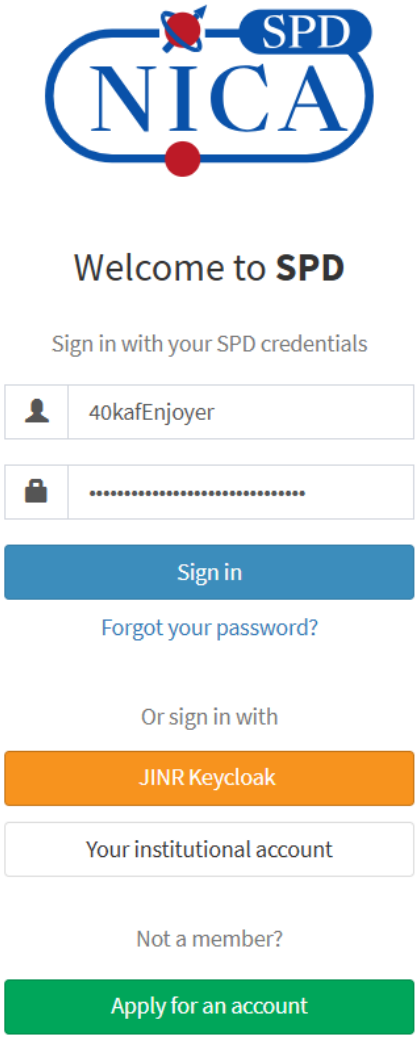


Рисунок 31 — Аутентификация через SPD-IAM

Роль / Действие	**Пользователь**	**Участник группы Production**	**Администратор**
Просмотр заданий	✓	✓	✓
Создание заданий	×	✓	✓
Изменение состояния заданий	×	Только если он создатель задания	Только если он создатель задания
Просмотр конфигурации сервера	×	×	✓

Таблица 5 — Матрица прав доступа

7 ТЕКУЩЕЕ ИСПОЛЬЗОВАНИЕ СИСТЕМЫ

Реализованный и описанный выше функционал, на данном этапе, полностью воплощает в себе MVP (Minimum Viable Product), поэтому, несмотря на продолжающуюся работу над сервисом, он уже был предоставлен для использования в процессе создания заданий. Это позволяет как понимать потребности физических групп, что позволит вести дальнейшую автоматизацию процессов с оглядкой на конечных пользователей, так и отслеживать корректность работы системы.

На данный момент, кроме тестовых заданий, были также запущены и успешно завершены цепочки на моделирование и реконструкцию для 17 различных версий и конфигураций прикладного программного обеспечения [36]. В роли последнего на текущий момент используется фреймворк SPDRoot [11], в рамках которого происходит генерация событий, рассчитываются траектории их распространения посредством Geant4 [37], создаются промежуточные хиты, восстанавливаются треки заряженных частиц, кластеры в электромагнитных калориметрах и т.д. По завершению реконструкции формируются данные, готовые к пользовательскому анализу.

В упомянутых выше цепочках заданий реализовывались различные детекторы и значения физических параметров. Краткая сводка по ним приведена в таблице 6 для неполяризованных p-p столкновений и в таблице 7 для неполяризованных d-d столкновений. На текущем этапе физический анализ полностью сфокусирован на рассмотрении событий с минимальным смещением.

За время использования сервиса на внутренних мощностях ОИЯИ и внешних серверах суммарно было выполнено 63 задания, из которых:

- 32 были завершены в статусе ’done’ и 31 в статусе ’finished’;
- 44 были на моделирование и 19 на реконструкцию;
- суммарно смоделировано более 1 миллиарда событий;
- выполнено около 400 тысяч задач;
- объем выходных датасетов превысил 600 ТБ.

Энергия [ГэВ]	**Генератор**	**Кол-во событий [миллионов]**	**Имя цепочки**	**Геометрия**
5	FTF	5	PROD2025-005	Micromegas, ST, ECal, RS, BBC, ZDC (sketch)
		40	PROD2025-007	
		40	PROD2025-008	
10	Pythia 8	5	PROD2025-001	Micromegas, ST, ECal, RS, BBC, ZDC (sketch)
		20	PROD2025-002	
		20	PROD2025-003	
		40	PROD2025-004	
	FTF	5	PROD2025-006	
		40	PROD2025-009	
27	Pythia 8	5	PROD2025-011	DSSD, ST, TOF, ECal, FARICH, RS, BBC, ZDC (sketch)
		40	PROD2025-012	
		40	PROD2025-013	
		40	PROD2025-014	
		40	PROD2025-015	
		40	PROD2025-016	
		20	PROD2025-017	

Таблица 6 — Сводка по симуляции и реконструкции p-p столкновений

Энергия	**Генератор**	**Кол-во событий**	**Имя цепочки**	**Геометрия**
4	FTF	5	PROD2025-010	Micromegas, ST, ECal, RS, BBC, ZDC (sketch)

Таблица 7 — Сводка по симуляции и реконструкции d-d столкновений

ЗАКЛЮЧЕНИЕ

На данном этапе научной работы:

- Изучены соответствующие системы, реализованные в рамках других мегасайнс экспериментов. Анализ показал, что в эксперименте ATLAS более сложные процессы обработки и другое прикладное ПО в виде Athena; в COMPASS - нет отдельной системы управления данными, подобной Rucio, ввиду чего большое количество дополнительного задач выполнялось системой, аналогичной разрабатываемой. Это привело нас к созданию собственной системы управления процессами обработки, но с учётом использования опыта обоих рассмотренных экспериментов.
- Изучены технологии, необходимые для реализации сервиса.
- Произведено полное развертывание и настройка приложения в рамках облачного сервиса ОИЯИ.
- Имплементированы механизмы коммуникации с прочими сервисами в рамках эксперимента.
- Внедрена возможность создания/мониторинга/завершения и отмены заданий.
- Реализована сквозная аутентификация на JWT.
- Сервис передан в эксплуатацию физическим группам.

СПИСОК ЛИТЕРАТУРЫ

1. Технический проект комплекса NICA. — 2025. — Дата последнего обращения: 20.06.2025. https://nica.jinr.ru/documents/TDR_spec_Fin0_for_site_eng.pdf/.
2. *Collaboration N.* Searching for a QCD mixed phase at the Nuclotron-based Ion Collider Facility : White Paper / Joint Institute for Nuclear Research (JINR). — 2010.
3. *Сисакян А.Н., Шевченко О.Ю., Нагайцев А.П., Иванов О.Н.* Эффекты поляризации в Дрелл-Яновских процессах. — 2010. — Дата последнего обращения: 20.06.2025. http://www1.jinr.ru/Pepan/2010-v41/v-41-1/04_sis.pdf.
4. *Iritani T., Cossu G., Hashimoto S.* Partial restoration of chiral symmetry inside hadrons // PoS. — 2014. — Т. LATTICE2014. — С. 338. — arXiv: 1412.2322 [hep-lat].
5. *Iritani T., Cossu G., Hashimoto S.* Lattice QCD study of partial restoration of chiral symmetry in the flux-tube // PoS / под ред. T. Iijima. — 2013. — Т. Hadron2013. — С. 159. — arXiv: 1401.4293 [hep-lat].
6. *Kou W., Chen X.* Exploring quantum entanglement in chiral symmetry partial restoration with 1+1 string model // Phys. Lett. B. — 2024. — Т. 853. — С. 138675. — arXiv: 2401.16673 [hep-ph].
7. *Овсянников В.* Техника и применение электронно-лучевых ионных источников : дис. ... д-ра техн. наук / Овсянников В.П. — Дубна : Объединенный институт ядерных исследований, 1996. — С. 33. — РГБ ОД, 9 96-2/1954-2.
8. *Ladygin V. P.* Spin Physics Detector at NICA // JPS Conf. Proc. — 2022. — Т. 37. — С. 011012. — arXiv: 2203.14704 [hep-ex].
9. *Guskov A.* Spin Physics Detector project at JINR // PoS. — 2022. — Т. PANIC2021. — С. 344. — arXiv: 2110.08930 [hep-ex].

10. Technical Design Report of the Spin Physics Detector at NICA / V. Abazov [и др.] // Natural Sci. Rev. — 2024. — Т. 1. — С. 1. — arXiv: 2404.08317 [hep-ex].

11. *SPD Collaboration.* Фреймворк SPDRoot. — Дата последнего обращения: 20.06.2025. https://git.jinr.ru/nica/spdroot.

12. The ATLAS Experiment at the CERN Large Hadron Collider / G. Aad [и др.] // JINST. — 2008. — Т. 3. — S08003.

13. Scaling up ATLAS production system for the LHC Run 2 and beyond: project ProdSys2 / M. Borodin [и др.] // J. Phys. Conf. Ser. — 2015. — Т. 664, № 6. — С. 062005.

14. Rucio - The next generation of large scale distributed system for ATLAS Data Management / V. Garonne [и др.] // J. Phys. Conf. Ser. / под ред. D. L. Groep, D. Bonacorsi. — 2014. — Т. 513. — С. 042021.

15. *Maeno T.* PanDA: Distributed production and distributed analysis system for ATLAS // J. Phys. Conf. Ser. / под ред. R. Sobie, R. Tafirout, J. Thomson. — 2008. — Т. 119. — С. 062036.

16. CRIC: Computing Resource Information Catalogue as a unified topology system for a large scale, heterogeneous and dynamic computing infrastructure / A. Anisenkov [и др.] // EPJ Web Conf. / под ред. C. Doglioni [и др.]. — 2020. — Т. 245. — С. 03032.

17. Rucio - Scientific data management / M. Barisits [и др.] // Comput. Softw. Big Sci. — 2019. — Т. 3, № 1. — С. 11. — arXiv: 1902.09857 [cs.DC].

18. *PanDA Team.* PanDA Client - Python API for PanDA Workload Management System. — 2025. — Дата последнего обращения: 20.06.2025. https://github.com/PanDAWMS/panda-client/blob/master/pandaclient/Client.py.

19. COMPASS (Common Muon and Proton Apparatus for Structure and Spectroscopy). 2025. — Дата последнего обращения: 20.06.2025. https://home.cern/science/experiments/compass.

20. *А.Ш. П.* Методика и программная инфраструктура глобально распределенной обработки данных эксперимента COMPASS : дис. ... канд. тех. наук / А.Ш. Петросян. — Дубна : Объединённый институт ядерных иссле-

дований, 2008. — Режим доступа: https://issc.jinr.ru/files/59/f_2_1466.pdf.

21. CTA (CERN Tape Archive). — 2025. — Дата последнего обращения: 20.06.2025. https://cta.web.cern.ch/cta/.
22. *Iven J., Lamanna M., Pace A.* CERN's AFS replacement project // J. Phys. Conf. Ser. / под ред. R. Mount, C. Tull. — 2017. — Т. 898, № 6. — С. 062040.
23. CASTOR (CERN Advanced STORage). — 2025. — Дата последнего обращения: 20.06.2025. https://castor.web.cern.ch/castor/.
24. IBM LSF. — 2025. — Дата последнего обращения: 20.06.2025. https://www.ibm.com/support/knowledgecenter/SSETD4/product_welcome_platform_lsf.html.
25. *University of Wisconsin–Madison.* HTCondor. — 2025. — Дата последнего обращения: 20.06.2025. https://htcondor.org/.
26. *Petrosyan A. S.* PanDA for COMPASS at JINR // Physics of Particles and Nuclei Letters. — 2016. — Т. 13, № 5. — С. 708—710.
27. *Petrosyan A.* COMPASS Production System Overview // EPJ Web of Conferences. — 2019. — Т. 214. — С. 03039—03039.
28. *Hardt D.* The OAuth 2.0 Authorization Framework. — 2012. — DOI: https://doi.org/10.17487/RFC6749. RFC 6749.
29. *M. Jones J. Bradley N. S.* JSON Web Token. — 2015. — DOI: https://doi.org/10.17487/RFC7519. RFC 7519.
30. *Bray T.* The JavaScript Object Notation (JSON) Data Interchange Format. — 2013. — DOI: https://doi.org/10.17487/RFC7158. RFC 7158.
31. *Josefsson S.* The Base16, Base32, and Base64 Data Encodings. — 2006. — DOI: https://doi.org/10.17487/RFC4648. RFC 4648.
32. *M. Jones J. Bradley N. S.* JSON Web Signature (JWS). — 2015. — DOI: https://doi.org/10.17487/RFC7515. RFC 7515.
33. *M. Jones J. H.* JSON Web Encryption (JWE). — 2015. — DOI: https://doi.org/10.17487/RFC7516. RFC 7516.
34. *Jones M.* JSON Web Algorithms (JWA). — 2015. — DOI: https://doi.org/10.17487/RFC7518. RFC 7518.

35. *SPD Collaboration*. Production manager control panel for the SPD experiment. — Дата последнего обращения: 20.06.2025. https://git.jinr.ru/monakov/highlevelservice.

36. *SPD Collaboration*. SPD MC productions 2025. — Дата последнего обращения: 20.06.2025. https://docs.google.com/spreadsheets/d/10uG1p6wPQ_GWe7wkZ_t4RPFLmsHuHxnth5MqM49uOYQ.

37. *Geant4 Collaboration*. Geant4 Toolkit for the simulation of the passage of particles through matter. — Дата последнего обращения: 20.06.2025. https://geant4.web.cern.ch/docs/.