Министерство науки и высшего образования Российской Федерации Федеральное государственное автомное образовательное учреждение высшего образрвания
«Национальный исследовательский ядерный университет «МИФИ»

УДК 539.12.01

ОТЧЕТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

СИСТЕМА УПРАВЛЕНИЯ ПРОЦЕССАМИ ОБРАБОТКИ В ГЕОГРАФИЧЕСКИ РАСПРЕДЕЛЕННОЙ СИСТЕМЕ ОБРАБОТКИ ДАННЫХ SPD

Научный руководитель _____ А.Ш. Петросян

Студент _____ Н. Г. Монаков

Москва 2023

Содержание

Введение

Любой физический эксперимент ставит перед собой трудные теоретические и технические задачи в области проведения, но работа над экспериментами в рамках мега-сайнс проектов, таких как эксперимент SPD в рамках проекта NICA, поднимает и последующие этапы сбора, хранения и обработки полученных данных на сложный технический уровень. Система, управляющая этими процессами должна быть реализована, кроме прочего, с учетом возможности изменения как её ресурсов, так и мощностей поставки данных с самого эксперимента. В ходе научной работы планируется изучить существующий мировой опыт построения подобных систем, в том числе географически распределенных, и встроиться в создание и поддержание данной системы для эксперимента SPD.

1 Эксперимент SPD

Экспериментальная установка SPD [1, 2], размещаемая в одной из двух точек пересечения пучков коллайдера NICA, предназначена для всестороннего изучения спиновой структуры протона и дейтрона.

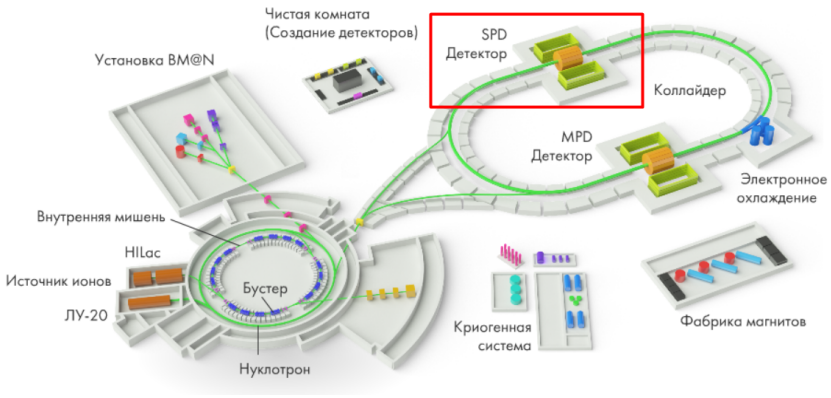

Рисунок 1 — Комплекс NICA

Основное внимание будет уделено изучению их поляризованной глюонной компоненты в реакциях инклюзивного рождения чармониев, открытого чарма и прямых фотонов, 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. Посредством измерения соответствующих 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). 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.

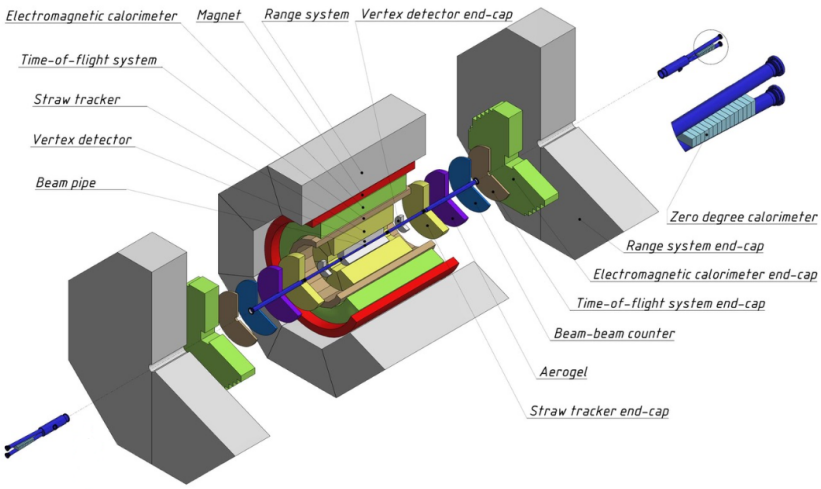

Рисунок 2 — Схема SPD

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 необходима система высокого уровня, которая позволит наиболее эффективно 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, 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, аналогично с ней сталкиваются и эксперименты на БАК, ПСС и т.д. Рассмотрим некоторые примеры технических решений.

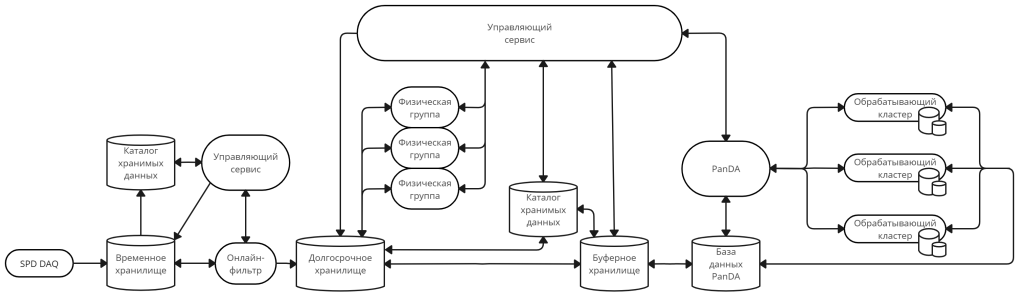

Рисунок 3 — Схема IT-составляющих

2.1 Реализация подобной системы ATLAS

Коллаборацией ATLAS [3] разработан набор взаимодействующих систем и сервисов для хранения и обработки данных. Верхним уровнем системы является ProdSys [4]. Это система, ответственная за управление процессами обработки данных и управляющая прочими имплементированными сервисами, вроде системы управления данными Rucio [5], системы управления нагрузкой PanDA [6]. При этом на уровне ProdSys пользователь работает не с конкретными задачами и файлами, а с их совокупностями: заданиями, датасетами/контейнерами соответственно, присваивая первым статусы, такие как 'waiting', 'Registered', 'Running', 'Done', 'Finished' и некоторые другие, обуславливающие дальнейшие действия системы. Именование реализовано в специальном формате, представленном на рисунке 4.

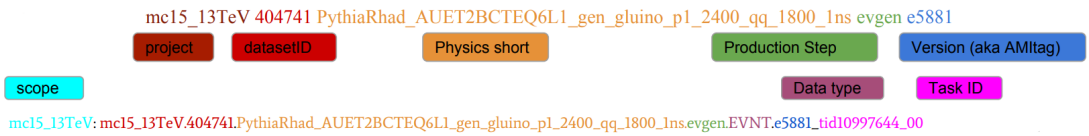

Рисунок 4 — Структура имен заданий и датасетов

Дочерние сервисы Rucio и PanDA содержат информацию о всех конкретных файлах и задачах. Информация о ресурсах хранится в информационной системе AGIS [7].

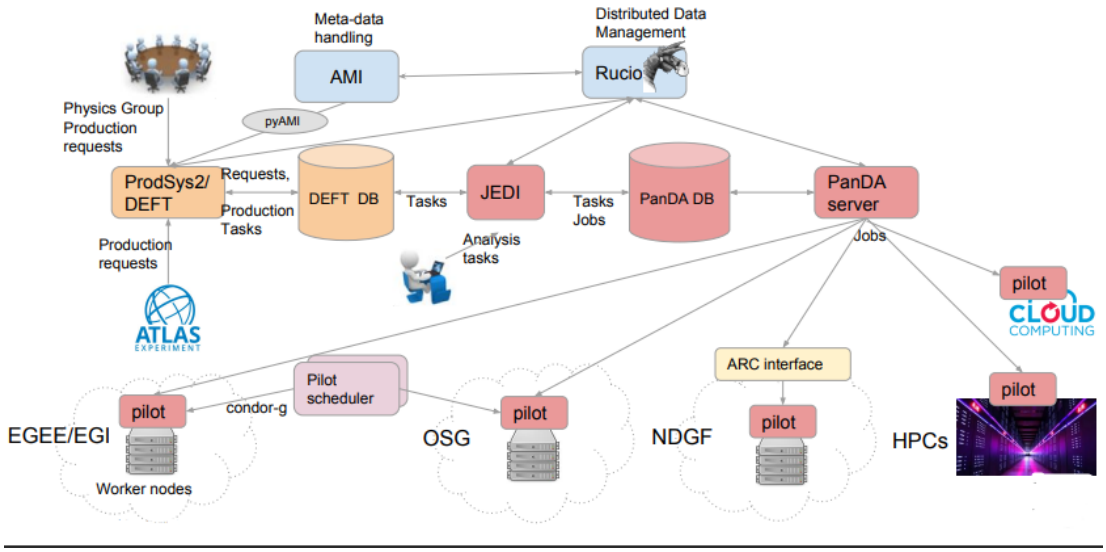

Рисунок 5 — Взаимосвязь сервисов и систем работы с данными ATLAS

Реализован набор средств мониторинга и управления обработкой, включая веб-интерфейс и интерфейс командной строки. Эксперимент использует исключительно распределенные ресурсы для хранения, обработки и анализа данных.

Система управления данными обеспечивает автоматическую передачу, учет и хранение данных в соответствии с нуждами коллаборации на более чем двух сотнях элементов хранения различных типов [8].

Система управления нагрузкой эксперимента производит подбор ресурса под задачу на основании различных метрик: доступности мощностей, наличия необходимых для выполнения задачи данных и тд.

PanDA предназначена для выполнения всех типов обработки, перечень типов представлен на рисунке 6, представляя им единое ПО, систему мониторинга, вида, аналогичного рисунку 12 и вычислительные ресурсы.

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 6 — Типы заданий

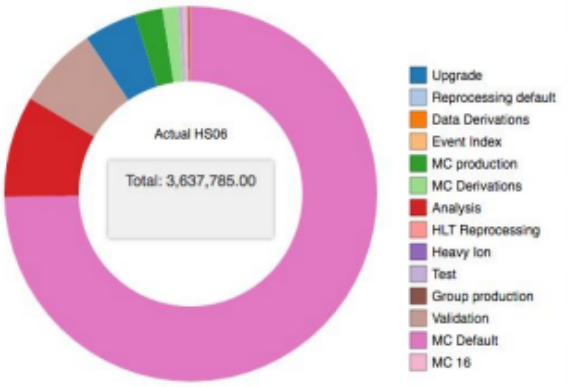

Рисунок 7 — Представление статистики

Система реализует конвейерную обработку данных, при которой передача данных и выполнение заданий производится параллельно: сервер PanDA отправляет запросы в систему DDM для отправки файлов на систему хранения или для объединения выходных файлов, а DDM [9] реализует фактическую передачу данных. DDM передает файлы, а затем отправляет сообщение о завершении операции на сервер PanDA. Задачи ждут в очереди задач после отправки в PanDA и переходят в статус “готова к выполнению” только тогда, когда все входные файлы переданы и готовы к обработке. Пилоты выбирают активированные задания и могут работать сразу после того как DDM подготовил входные файлы. Точно так же пилоты копируют выходные файлы в локальные хранилища и немедленно завершаются, чтобы освободить вычислительные мощности. Затем сервер PanDA отправляет в DDM запросы на перемещение файлов.

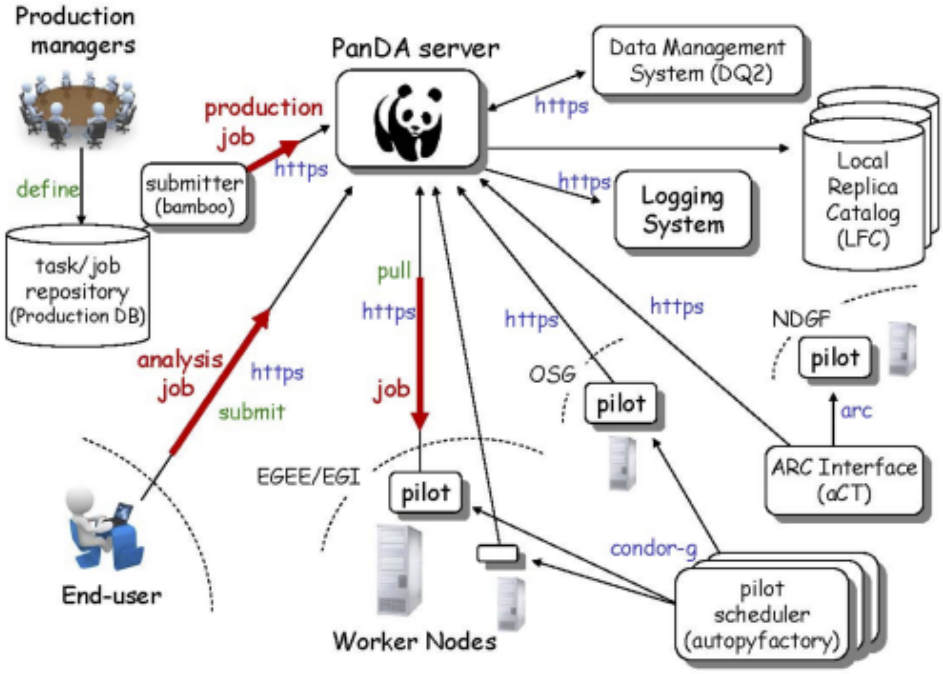

Рисунок 8 — Архитектура PanDA

2.2 Реализация подобной системы COMPASS

Базируясь в ЦЕРН, эксперимент COMPASS [10] реализует работу с данными опираясь на IT-сервисы, предлагаемые департаментом информационных технологий европейской организацией ядерных исследований, такие как распределенная файловая система AFS [11], позже упраздненная, системой долговременного хранения CASTOR [12], системой пакетной обработки под управлением LSF [13], позже заманённой на систему управления нагрузкой PanDA. Изначально и необработанные, и промежуточные данные размещались на CASTOR. С введением в эксплуатацию сервиса EOS [14], он стал использоваться в качестве временного хранилища данных во время обработки. На ленточное хранилище CASTOR доставлялись только финальные данные для долгосрочного хранения.

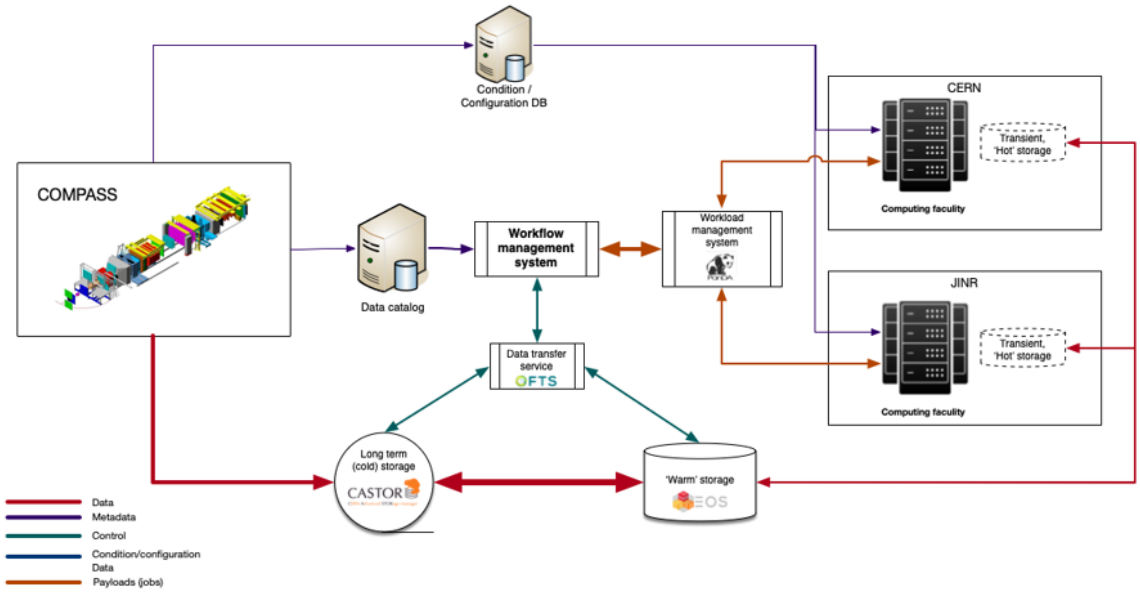

Рисунок 9 — Взаимосвязь сервисов и систем работы с данными COMPASS

Все собранные данные записываются системой сбора информации DAQ на CASTOR, данные и метаданные регистрируются в специализированной базе данных, когда задание на обработку сформировано и запущено, производится подготовка входных данных: запрашивается перенос данных с лент долгосрочного хранения на дисковый массив, после на вычислительную ферму, через сервера PanDA отправляются задачи обработки этих данных. После обработки результаты отдельных задач перемещаются для ожидания завершения остальных задач и подготовки процедуры слияния, а затем отправляются на CASTOR для долгосрочного хранения.

База данных системы 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 приведен на рисунке 10.

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

Рисунок 10 — Структура имен заданий и датасетов

Управлением обработкой занимается менеджер обработки данных. Для построения вычислительной среды эксперимента создан служебный пользователь, в рабочей директо-

рии на AFS которого размещается необходимое программное обеспечение. Реализован автоматический перезапуск задачи при ошибках. Менеджер обработки подготавливает список файлов с входными данными, устанавливает или проверяет установку необходимой версии программного обеспечения, задает параметры в специальном файле и отправляет файлы на обработку.

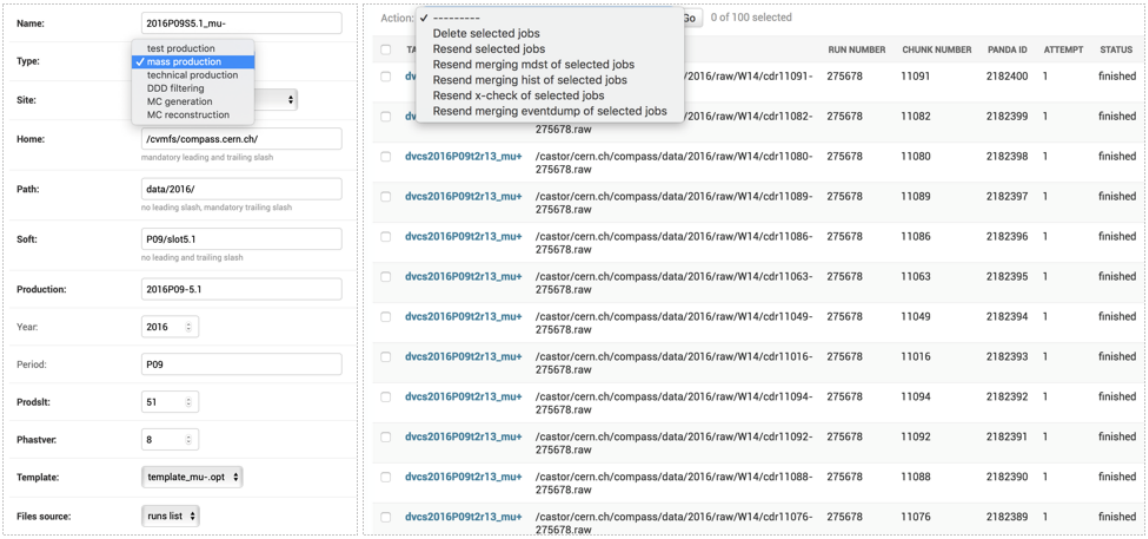

Рисунок 11 — Панель для формирования заданий COMPASS

Реализован интерфейс для мониторинга состояния заданий

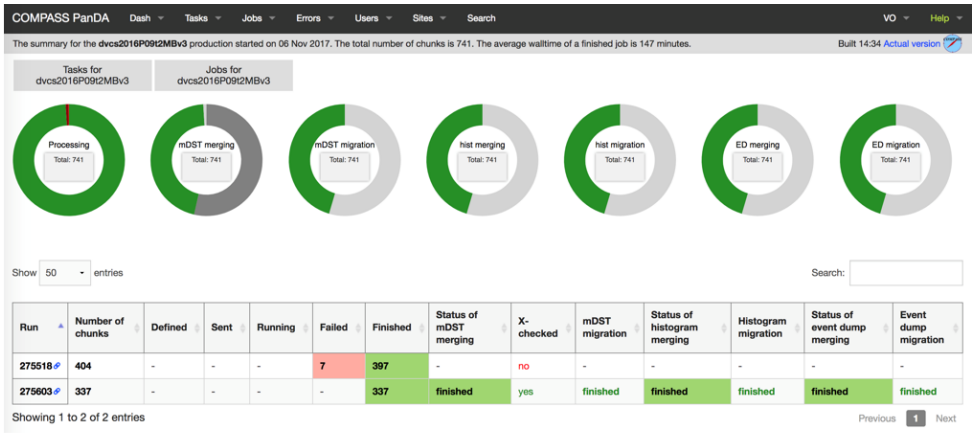

Рисунок 12 — Представление статистики

3 Анализ технологий для построения системы

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 мы могли бы рассмотреть совместное использование фреймворков для обеспечения наилучшей реализации веб-приложения: Django для серверной части и Angular для клиентской. Это может позволить разделить логику приложения на независимые части, что облегчит разработку, тестирование и поддержку, а также позволит использовать все преимущества обоих фреймворков.

4 PanDa Client

Созданные в рамках веб-приложения задания должны быть отправлены на выполнение. PanDA client реализует соответствующий функционал, используемый для взаимодействия с PanDA системой управления рабочими нагрузками. 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 с 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 ...]
  - kill
  - finish
  - retry
  - show
  - showl
  - ...
- Настройка параметров отправки заданий и управления ресурсами.

5 Создание прототипа веб-приложения

В ходе работы было начато изучение фреймворка Django, с использованием которого была начата реализация веб-приложения для создания, управления и отслеживания заданий на обработку. Реализованы средства регистрации и авторизации пользователей

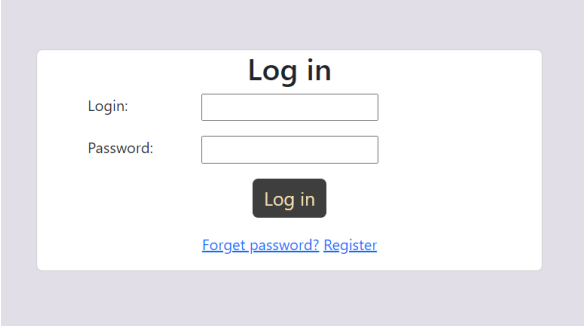

Рисунок 13 — Авторизация

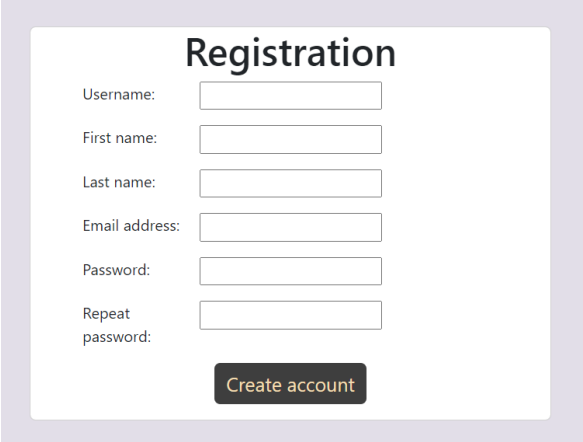

Рисунок 14 — Регистрация

Создана модель задания, форма для создания задания пользователем, возможность проверки текущего состояния, а также его редактирования, в зависимости от прав доступа редактирующего.

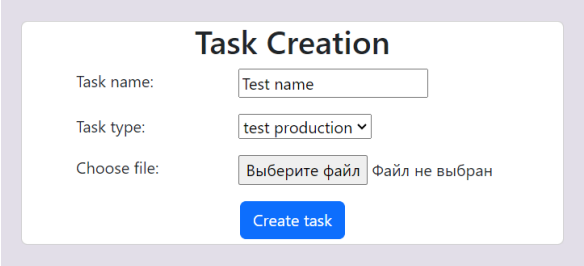

Рисунок 15 — Создание задания

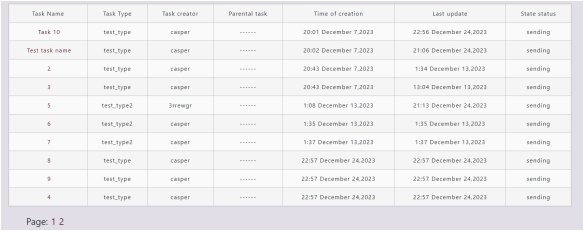

Рисунок 16 — Задания

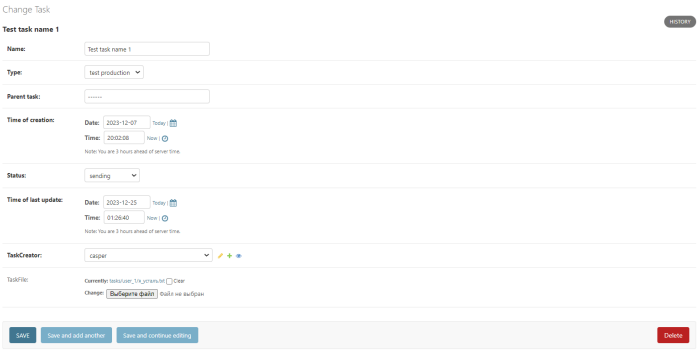

Рисунок 17 — Редактирование задания

Добавлены возможности фильтрации и сортировки заданий.

Name ↑ ↓	Year ↑ ↓	Energy ↑ ↓	Polarization	Statistics ↑ ↓	Dataset ID ↑ ↓	Task Type	Task creator	Parental task	Last update ↑ ↓	State status
123	1	1.0 GeV	temp type 1	1.0 MM	1	Data	casper	------	13:40 April 17,2024	toBroken
0	1	1.0 GeV	temp type 1	1.0 MM	1	Data	casper	------	10:17 April 17,2024	sending
01	1	1.0 GeV	temp type 1	1.0 MM	1	Data	casper	------	14:35 April 17,2024	sending
01111	1	1.0 GeV	temp type 1	7.0 MM	6	Data	casper	------	14:51 April 17,2024	sending
01312	2	2.0 GeV	temp type 1	2.0 MM	2	Data	casper	------	14:55 April 17,2024	sending
3_3_casper	3	3.0 GeV	temp type 1	3.0 MM	3	Data	casper	------	13:06 April 17,2024	sending
5_5_casper	5	5.0 GeV	temp type 1	5.0 MM	5	Data	casper	------	13:08 April 17,2024	sending
6_6_casper	6	6.0 GeV	temp type 1	6.0 MM	6	Data	casper	------	13:12 April 17,2024	sending
6_6_casper555	6	6.0 GeV	temp type 1	6.0 MM	6	Data	casper	------	13:13 April 17,2024	sending
7_7_casper	7	7.0 GeV	temp type 1	7.0 MM	7	Data	casper	------	12:58 April 17,2024	sending

Page: 1 2

Рисунок 18 — Сортировка по энергиям

taskName 123 Filter

Name ↑ ↓	Year ↑ ↓	Energy ↑ ↓	Polarization	Statistics ↑ ↓	Dataset ID ↑ ↓	Task Type	Task creator	Parental task	Last update ↑ ↓	State status
123	1	1.0 GeV	temp type 1	1.0 MM	1	Data	casper	------	13:40 April 17,2024	toBroken
123_31_casper	123	123.0 GeV	temp type 1	31.0 MM	132	Data	casper	------	13:20 April 17,2024	registered
9_9_123	9	9.0 GeV	temp type 1	9.0 MM	9	Data	123	------	14:04 April 18,2024	sending

Page: 1

Рисунок 19 — Фильтр по части имени задания

6 Создание веб-сервера

Создание веб-сервера осуществлено посредствам серверного ПО Apache2, связанного с проектом через mod_wsgi.

Apache2 используется для обработки HTTP-, FTP-, SMTP-запросов и отправки ответов на них пользователю. Основан на модульной архитектуре, что ведет к гибкости и масштабируемости. Архитектура Apache2 включает в себя ядро сервера, модули, конфигурационные файлы и виртуальные хосты.

Создан виртуальный хост для обработки запросов, отредактированы некоторые файлы конфигураций для их обработки, например конфигурационный файл "apache2/sites-available/***.conf для связывания Django-приложения с Apache через wsgi. Внесли соответствующие правки в wsgi.py и settings.py, добавив каталог apache2 в PATH приложения и прописав ip-адрес соответственно.

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

7 Механизм аутентификации и авторизации

Внедрен механизм аутентификации и авторизации посредством стороннего сервиса, а именно, spd-iam.jinr.ru, что позволило встроить разрабатываемое приложение в общую систему с единой точкой входа. Взаимодействие с сервисом авторизации осуществляется по протоколу OAuth2, с наиболее безопасным типом потока авторизации: "grant_type": authorization_code.

В дипломе здесь будет большой теоретический блок, посвященный OAuth 2.0

Получаемый при этом соединении JSON-объект содержит самодостаточный набор метаданных и нагрузки в виде JWT, позволяющий как валидировать неподдельность пакета, так и управлять доступом к защищенным частям сервиса.

В дипломе здесь будет большой теоретический блок, посвященный JWT

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

- userName
- 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

Заключение

На данном этапе научной работы начато и продолжается освоения ПО, необходимого для реализации системы контроля данных высокого уровня. Были изучены соответствующие системы, реализованные в рамках других мега-сайнс экспериментов, произведено полное развертывание и настройка приложения в рамках облачного сервиса ОИЯИ, имплементированы механизмы коммуникации с прочими сервисами в рамках эксперимента, начат процесс масштабной унификации кода приложения и расширены его внутренние возможности.

Список литературы

[1] V. P. Ladygin. Spin Physics Detector at NICA. *JPS Conf. Proc.*, 37:011012, 2022.

[2] Alexey Guskov. Spin Physics Detector project at JINR. *PoS*, PANIC2021:344, 2022.

[3] G. Aad et al. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, 3:S08003, 2008.

[4] M. Borodin, K. De, J. Garcia Navarro, D. Golubkov, A. Klimentov, T. Maeno, and A. Vaniachine. Scaling up ATLAS production system for the LHC Run 2 and beyond: project ProdSys2. *J. Phys. Conf. Ser.*, 664(6):062005, 2015.

[5] Vincent Garonne, R. Vigne, G. Stewart, M. Barisits, T. Beermann, M. Lassnig, C. Serfon, L. Goossens, and A. Nairz. Rucio - The next generation of large scale distributed system for ATLAS Data Management. *J. Phys. Conf. Ser.*, 513:042021, 2014.

[6] Tadashi Maeno. PanDA: Distributed production and distributed analysis system for ATLAS. *J. Phys. Conf. Ser.*, 119:062036, 2008.

[7] A. Anisenkov, A. Di Girolamo, A. Klimentov, D. Oleynik, and A. Petrosyan. AGIS: The ATLAS Grid Information System. *J. Phys. Conf. Ser.*, 513:032001, 2014.

[8] Danila Oleynik, Artem Petrosyan, Vincent Garonne, and Simone Campana. ATLAS DQ2 deletion service. *J. Phys. Conf. Ser.*, 396:032083, 2012.

[9] Vincent Garonne, Martin Barisits, Thomas Beermann, M. Lassnig, Cedric Serfon, and Wen Guan. Experiences with the new ATLAS Distributed Data Management System. *J. Phys. Conf. Ser.*, 898(6):062019, 2017.

[10] COMPASS (Common Muon and Proton Apparatus for Structure and Spectroscopy). `https://home.cern/science/experiments/compass`.

[11] J. Iven, M. Lamanna, and A. Pace. CERN's AFS replacement project. *J. Phys. Conf. Ser.*, 898(6):062040, 2017.

[12] CASTOR (CERN Advanced STORage). `https://castor.web.cern.ch/castor/`.

[13] IBM LSF. `https://www.ibm.com/support/knowledgecenter/SSETD4/product_welcome_platform_lsf.html`.

[14] EOS. `https://eos-web.web.cern.ch/`.