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»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

СИСТЕМА УПРАВЛЕНИЯ ПРОЦЕССАМИ ОБРАБОТКИ ДЛЯ СПЕЦИАЛИЗИРОВАННОГО ВЫЧИСЛИТЕЛЬНОГО КОМПЛЕКСА SPD ONLINE FILTER

Студент _____ А. В. Плотников

Научный консультант _____ Д. А. Олейник

Москва 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время практически любой эксперимент в физике высоких энергий предполагает проведения большого количества измерений, порождающего, в результате, гигантский объём данных. При этом данные нужно принимать, систематизировать, обрабатывать и сохранять.

Остро данный аспект ощущается в том числе и в сфере физики высоких энергий. При экспериментах на коллайдере во время столкновения пучков частиц могут происходить сотни тысяч событий в секунду. И хоть на данный момент детекторные системы позволяют регистрировать подобные события с достаточно высокой точностью, всё также остаётся необходимость эффективно управлять большим потоком данных с детекторов.

Учитывая, что системы хранения имеют ограниченный размер, существует необходимость отсеивать ненужные, в том числе и фоновые, события. Чаще всего в таких случаях применяют аппаратные триггерные системы, но в некоторых ситуациях подобные конструкции невозможны из-за особенностей эксперимента. В частности, необходимой может оказаться частичная реконструкция событий, для которой предполагается создание фильтров, обрабатывающих данные в режиме реального времени.

В данной работе будет рассмотрена подобная система для разрабатываемого эксперимента SPD на коллайдере NICA (Дубна, Россия) [1]. А также будет спроектирована одна из подсистем – система управления процессами обработки.

Цель и задачи работы

Цель работы: разработка системы (Workflow management system), позволяющей параллельно и наиболее результативно управлять большим количеством процессов обработки данных и контролировать статусы выполнения цепочек обработки для физического эксперимента SPD.

Задачи:

1. ознакомиться с проектом системы Workflow management system (WfMS): определить основные сервисы WfMS, выявить их основной функционал и принцип взаимодействия с другими системами SPD Online Filter;
2. изучить выбранное программное обеспечение, по необходимости дополнить/переработать стек;
3. создать сервис для работы с WMS:
    1. отправление заданий на обработку;
    2. периодический опрос для отслеживания статуса;
    3. изменение приоритета заданий;
    4. отмена заданий.
4. разработать сервис для опроса DMS:
    1. получение информации о готовности входных датасетов;
    2. формирование выходного датасета и датасета логов;
    3. удаление датасета.

1. SPD Online Filter

1.1 Принцип работы SPD Online Filter

SPD Online Filter — это высокопроизводительная вычислительная система для высокопропускной первичной обработки данных эксперимента SPD [2].

Эта вычислительная система должна осуществлять следующее преобразование данных: идентифицировать физические события во временных интервалах; реорганизовать данные в событийно-ориентированном формате; отфильтровать события, оставляя только интересные с точки зрения эксперимента; согласовывать выходные данные, объединять события в файлы и файлы в наборы данных для будущей обработки.

Концепция базовой обработки:

- Реконструкция треков и связывание их с вершинами;
- Связывание срабатываний электромагнитного калориметра и пробежной системы с каждой вершиной по времени;
- Определение несвязанных срабатываний детектора;
- Объединение с необработанными сигналами от других субдетекторов;
- Формирование блоков данных и сохранение частично реконструированных событий.

Так как SPD Online Filter предназначен для обработки большого потока данных. Данные объединяются в наборы файлов в зависимости от стадии их обработки. Каждый файл в наборе может быть обработан независимо, однако наборы обрабатываются в некоторой заданной последовательности. Каждый файл можно обработать за некоторое разумное время на ограниченном вычислительном ресурсе (вычислительном узле).

Разбиение данных на более мелкие части позволяет управлять объемом обработки и минимизировать последствия при возникновении ошибок при обработке больших объемов данных. Работа с отдельными частями позволяет

изолировать ошибки, которые могут возникнуть в процессе обработки, и снизить их воздействие на всю систему.

Такой подход также обеспечивает масштабируемость системы: добавление дополнительных вычислительных ресурсов позволяет обрабатывать еще большие объемы данных без значительного изменения общей архитектуры системы. Это особенно полезно для систем, работающих с высокочастотными данными или при необходимости быстрой обработки информации в режиме реального времени.

По результатам анализа первичных требований к системе были выделены основные компоненты:

1. Workflow management system
2. Data management system
3. Workload management system & pilot

Архитектура системы SPD Online Filter приведена на рис. 1.

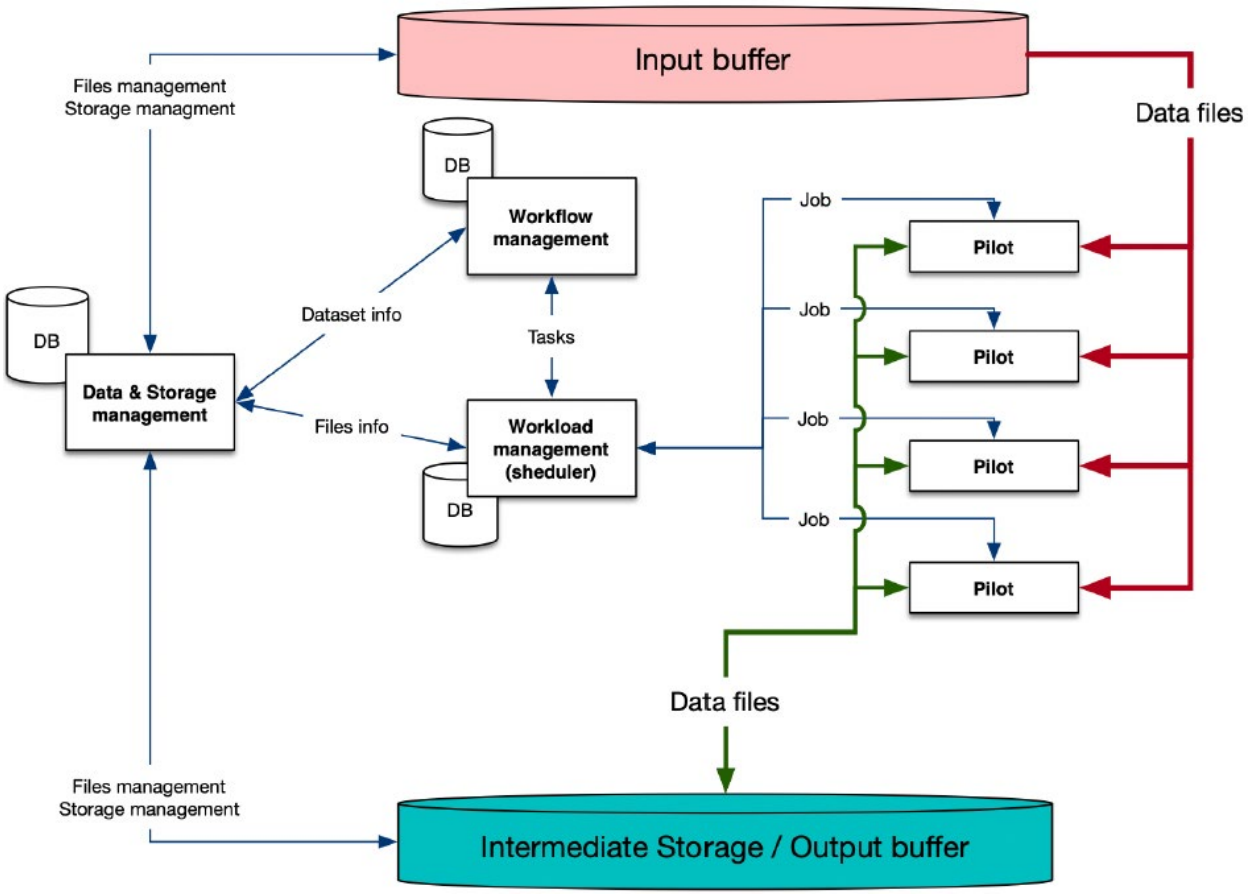


Рис. 1 Архитектура SPD Online Filter [3]

1.2 Входные данные

Входные данные, приходят из системы сбора данных (DAQ) во входной буфер.

- Период набора – ассоциирован с физической задачей. Состоит из набора ранов. Имеет дату начала, дату окончания.
- Ран – интервал, когда условия эксперимента неизменны (калибровки, например). Измеряется часами. Состоит из фреймов.
- Фрейм – нумерованный блок данных с детектора. Может содержать информацию о множестве событий. Продолжительность секунды.
- Датасет – логическое объединение файлов в наборы для обработки. Датасет является единицей обработки для одного задания.
- Файл – количество данных для обработки одной задачей.

2. Система управления процессами обработки (WfMS)

После попадания в SPD Online Filter, данные регистрируются в системе управления данными (Data Management System - DMS). Затем данные готовы к последующей обработке при помощи системы управления процессами обработки (Workflow Management System - WfMS).

WfMS принимает эти зарегистрированные данные и сопоставляет их с определенным шаблоном цепочки обработки, который предоставлен оператором обработки данных. Каждый этап этой цепочки обработки генерирует соответствующие задания для выполнения. Эти задания затем направляются в систему управления нагрузкой (Workload Management System - WMS), где они распределяются на вычислительные ресурсы для выполнения.

Этот процесс позволяет эффективно организовать и контролировать обработку данных по заданной цепочке этапов. Задания каждого этапа обработки создаются автоматически и передаются в систему управления нагрузкой для выполнения на доступных вычислительных ресурсах. Такой подход обеспечивает систему планирования и контроля рабочих процессов, что улучшает эффективность обработки данных и ускоряет процесс анализа.

Система управления процессами обработки (WfMS) имеет ряд функциональных требований, чтобы эффективно организовать и контролировать последовательности обработки данных:

1. Организация последовательностей обработки данных:

• Последовательность: состоит из логически связанных шагов обработки данных, где результат одного шага является входными данными для следующего.

• Шаг обработки: представляет собой действие, которое выполняется над набором данных. Эти шаги могут быть типа "map" (однотипная обработка для каждого элемента данных) или "merge" (объединение гомогенных данных).

• Описание шаблона: выражается с помощью CWL (Common Workflow Language).

2. Формирование запроса на обработку данных:

Для каждого шага обработки определяются необходимые параметры для формирования задания. Шаблон задания будет рассмотрен далее.

3. Выполнение запроса на обработку данных:

• Создание заданий для каждого шага обработки данных на основе сформированных параметров.

• Отправка созданных заданий в систему управления нагрузкой для выполнения.

• Осуществление контроля выполнения обработки данных путем опроса системы управления нагрузкой.

Эти функциональные требования позволят системе WfMS эффективно управлять процессами обработки, обеспечивая оптимизацию обработки данных, контроль ошибок и достижение заданных целей обработки.

В основном процессе обработки данных в SPD Online Filter можно выделить следующие шаги:

1. Декодирование данных;

2. Выявление событий и реструктуризация данных;

3. Фильтрация выявленных событий;

4. Верификация данных;

5. Объединение файлов (уменьшение количества файлов, увеличение размера файлов);

Такой набор последовательных шагов обработки будет называться цепочкой обработки. Каждый из шагов в такой цепочке может быть определен шаблоном.

Во время работы системы предполагается создание промежуточных данных, которые будут удаляться после получения выходного набора данных.

В результате проектирования были определены основные сервисы системы WfMS:

• Сервис для взаимодействия с оператором обработки данных:

1) вывод информации о заданиях;

2) изменение приоритетов заданий или их отмена.

- Сервис для опроса DMS:

1) получение информации о готовности входных датасетов;

2) формирование выходного датасета;

3) удаление датасета;

- Сервис генерации заданий:

1) определение последовательностей обработки данных;

2) запрос шаблонов;

3) создание заданий по шаблонам для последовательностей.

- Сервис для работы с WMS:

1) отправление заданий на обработку;

2) периодический опрос для отслеживания статуса;

3) изменение приоритета заданий;

4) отмена заданий.

Архитектура WfMS приведена на рис. 2.

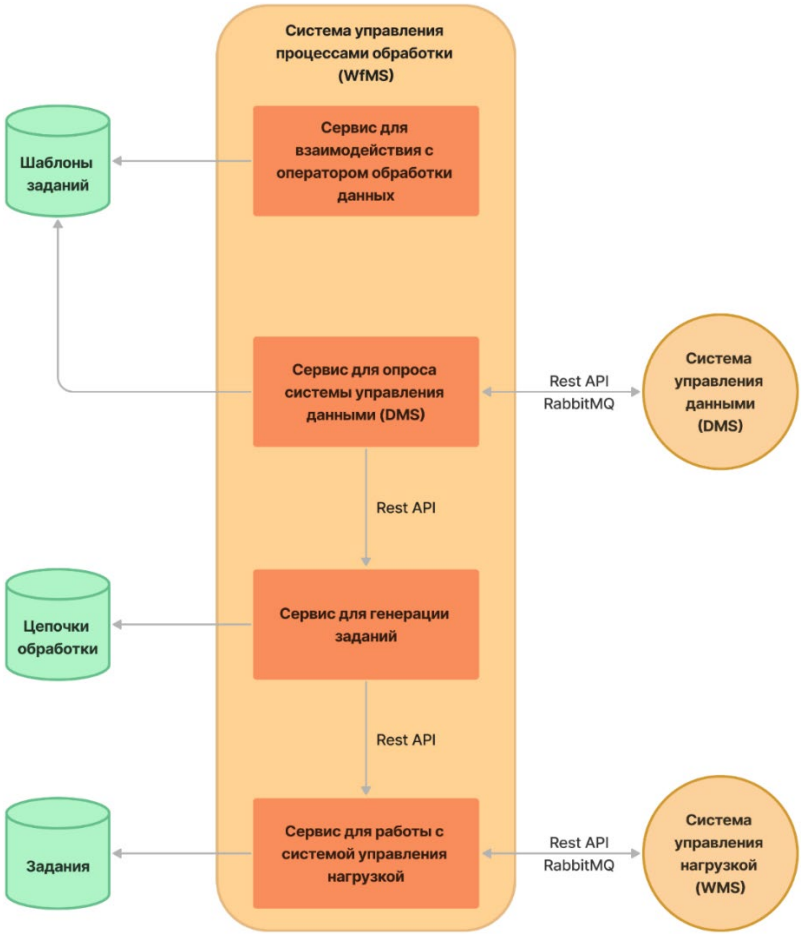


Рис. 2 Архитектура Workflow Management System

В данной системе применяется микросервисная архитектура. В рамках этой архитектуры приложение разбивается на отдельные сервисы, которые могут быть развернуты и масштабированы независимо друг от друга. Сервисы взаимодействуют между собой через API-интерфейсы, что позволяет каждому из них функционировать автономно. В отличие от монолитной архитектуры, микросервисы позволяют быстрее внедрять новые функции и вносить изменения, не требуя переработки большого объема существующего кода.

Основные преимущества микросервисной архитектуры включают:

- Повышенная доступность и отказоустойчивость. Сбой одного из сервисов не приводит к отказу всей системы.
- Масштабируемость. При достижении предельной нагрузки на микросервис можно развернуть дополнительные экземпляры этой службы и распределить нагрузку, поддерживая таким образом работоспособность большого количества экземпляров.

3. Взаимодействие с Data Management System

Далее представлено подробное описание процесса взаимодействия с Data Management System (DMS). Основной задачей данного взаимодействия является получение новых данных, регистрация промежуточных данных и выгрузка выходных данных. Кроме основных функций, взаимодействие должно быть реализовано с учетом обеспечения высокой эффективности и максимальной скорости, исключения узких мест, а также минимизации возможных ошибок.

Взаимодействие с DMS включает следующие этапы:

1. Получение входного набора данных. После поступления данных в SPD Online Filter, DMS регистрирует их в своей системе. После этого WfMS может получить информацию об этом датасете через Rest API.

2. Регистрация выходного набора данных. Перед любым этапом обработки данных WfMS должен зарегистрировать выходной набор данных, чтобы знать, где и как искать полученные данные. Для этого также используется Rest API. Промежуточные и выходные данные сопровождаются логами, которые регистрируются аналогичным образом.

3. Обработка промежуточных данных. В ходе обработки данных появляются промежуточные результаты. Если в цепочке обработки имеется n этапов, то при сохранении всех промежуточных данных после каждого этапа объем используемой памяти значительно увеличивается. Поэтому принято решение сохранять только один набор входных данных и один набор промежуточных данных. Это значит, что после завершения этапа обработки и появления нового датасета, входные данные для этого этапа удаляются. Входные данные для первого этапа остаются до завершения всей цепочки обработки, чтобы можно было восстановить данные в случае сбоя системы. В данном взаимодействии используется асинхронный метод, позволяющий не останавливать работу WfMS и не ждать удаления данных. Логи для промежуточных и выходных данных также должны удаляться аналогичным способом.

Далее представлена таблица (см. Табл. 1) взаимодействия WfMS и DMS со всеми необходимыми параметрами:

Табл. 1 Взаимодействие WfMS и DMS

Запрос	Тип запроса	Input 1	Input 2	Output	Примечания
Получение входного датасета	Rest API (GET)	Timestamp / None	-	[{ • uid • timestamp • dat_in_uid • file_num }, ...]	Возвращает список параметров датасетов, добавленных в систему позднее timestamp. В случае None возвращает все датасеты.
Создание выходного датасета	Rest API (POST)	Name	meta	UID	
Удаление датасета	RabbitMQ (DELETE)	UID	-	-	Удаление датасета — область ответственности DMS.
Получение информации о хранилище	Rest API (GET)	Type	-	ID, URL	
Получение статуса датасета	Rest API	UID	-	Status	Определение готовности датасета к дальнейшей обработке

Таким образом, в этом разделе детально рассмотрено взаимодействие с DMS. Описаны зоны ответственности каждой системы и перечислены все необходимые входные и выходные параметры. Для обеспечения высокой эффективности взаимодействия с DMS мы используем различные стратегии и технологии. В случаях, где важна асинхронность, применяются асинхронные запросы. Это позволяет выполнять несколько операций параллельно, максимально задействуя ресурсы системы и ускоряя процесс взаимодействия.

3.1 Сервис для опроса DMS

Для получения информации о новых данных, WfMS периодически запрашивает DMS о наличии новых данных. Частота запросов задается параметром при запуске программы. При этом WfMS должен хранить время последнего полученного датасета, чтобы система управления данными знала, какие датасеты передавать, избегая повторной передачи одних и тех же данных.

При первом опросе timestamp не передается, и DMS возвращает информацию обо всех зарегистрированных датасетах. В полученном JSON у каждого датасета есть параметр timestamp регистрации. WfMS выбирает самый поздний timestamp и сохраняет его, отправляя его при каждом следующем запросе и обновляя по необходимости.

Чтобы сохранить этот параметр в случае перезагрузки системы или сбоя, он сериализуется. Сериализация — это процесс преобразования структуры данных в битовую последовательность, обратной которой является десериализация. Эта операция позволяет быстро сохранять и считывать объекты в долгосрочной памяти. В данном случае это удобно, так как работа с базой данных была бы медленнее и затратнее. В Python для этого используется модуль pickle, реализующий двоичные протоколы для сериализации и десериализации объектов.

После получения информации о новых датасетах данный сервис сопоставляет датасет с готовым шаблоном цепочки обработки и получает необходимые параметры для поиска конкретной цепочки. Затем сервис отправляет информацию о датасете и цепочке с конкретными параметрами в сервис для генерации заданий через Rest API.

4. Взаимодействие с Workload Management System

Далее представлено подробное описание процесса взаимодействия с Workload Management System (WMS). WMS отвечает за распределение нагрузки по узлам. В SPD Online Filter система управления задачами (WfMS) формирует задания для каждого этапа обработки на основе шаблона и передает их WMS, которая затем отвечает за их выполнение. Как и в случае с взаимодействием с DMS, этот процесс должен обеспечивать высокую эффективность и скорость взаимодействия, исключение узких мест и минимизацию возможных ошибок. SPD Online Filter обрабатывает большой объем данных, которые необходимо быстро обрабатывать.

Для достижения максимальной эффективности используются следующие методы взаимодействия:

1. Сопоставление данных с шаблоном обработки. После получения новых данных WfMS сопоставляет их с шаблоном цепочки обработки, формирует задание для первого этапа и передает его WMS. WMS обрабатывает данные и выдает новые результаты, для которых WfMS формирует следующее задание, и так до завершения всей цепочки обработки.

2. Мониторинг выполнения заданий. После передачи задания в WMS необходимо отслеживать его выполнение. WfMS регулярно запрашивает у WMS информацию о количестве обработанных файлов и количестве ошибок.

3. Управление приоритетами. В случае экстренных ситуаций, например, если входной буфер переполняется быстрее, чем SPD Online Filter может обработать данные, WfMS может изменять приоритеты заданий для ускорения обработки.

4. Отмена заданий. В некоторых случаях может потребоваться отмена заданий, например, если количество ошибок превышает количество обработанных файлов.

Эти методы позволяют эффективно выполнять требуемые функции и обеспечивать стабильную работу системы.

Далее представлена таблица (см. Табл. 2) взаимодействия WfMS и WMS со всеми необходимыми параметрами:

Табл. 2 Взаимодействие WfMS и WMS

Запрос	Тип запроса	Input 1	Input 2	Output	Примечания
Отправка задания	RabbitMQ (POST)	Task	-	-	Описание задания будет представлено далее.

Получение статуса обработки задач	Rest API (GET)	Id	-	{
				• total
				• running
				• canceled
				• killed
				• failed
				}	

Изменение приоритета задания	Rest API (PUT)	Id	rank	-	
Отмена задания	Rest API (PUT)	Id	-	-	

Задание представляет собой единицу рабочей нагрузки, связанную с обработкой определенного датасета. Если обработка данных выполняется в несколько этапов, каждый этап соотносится с отдельным заданием. Задание формально определяется, как набор входных данных и обработчик, который выполняет необходимые операции над данными и генерирует выходной набор данных и логов.

Далее представлено описание задания, которое описано в формате JSON (см. Табл. 3)

Табл. 3 Описание задания (Task)

Ключ	Тип	Описание
task_id	int	Уникальный идентификатор задания
executable	str	Исполняемый файл запуска
args	str | None	Флаги для исполняемого файла задания
rank	int	Приоритет задания
device_type	str	Определение выбранного вычислительного ресурса,

		который будет использоваться для выполнения задания (CPU/GPU)
mode	str	Тип задания (Map/Merge)
retry	int	Максимальное число попыток выполнения задач в случае неудачной попытки
dat_in_uid	list[int]	Уникальные идентификаторы входного датасета
dat_out_uid	list[int]	Уникальные идентификаторы выходного датасета
dat_stor_url	str	Унифицированный указатель ресурса выходного хранилища
dat_out_orl	str	Унифицированный указатель ресурса выходного датасета
log_out_uid	int	Уникальный идентификатор выходного датасета логов
log_stor_url	str	Унифицированный указатель ресурса выходного хранилища для хранения логов
log_out_url	str	Унифицированный указатель ресурса выходного датасета логов

На рис. 3 и 4 показана отправка задания в очередь.

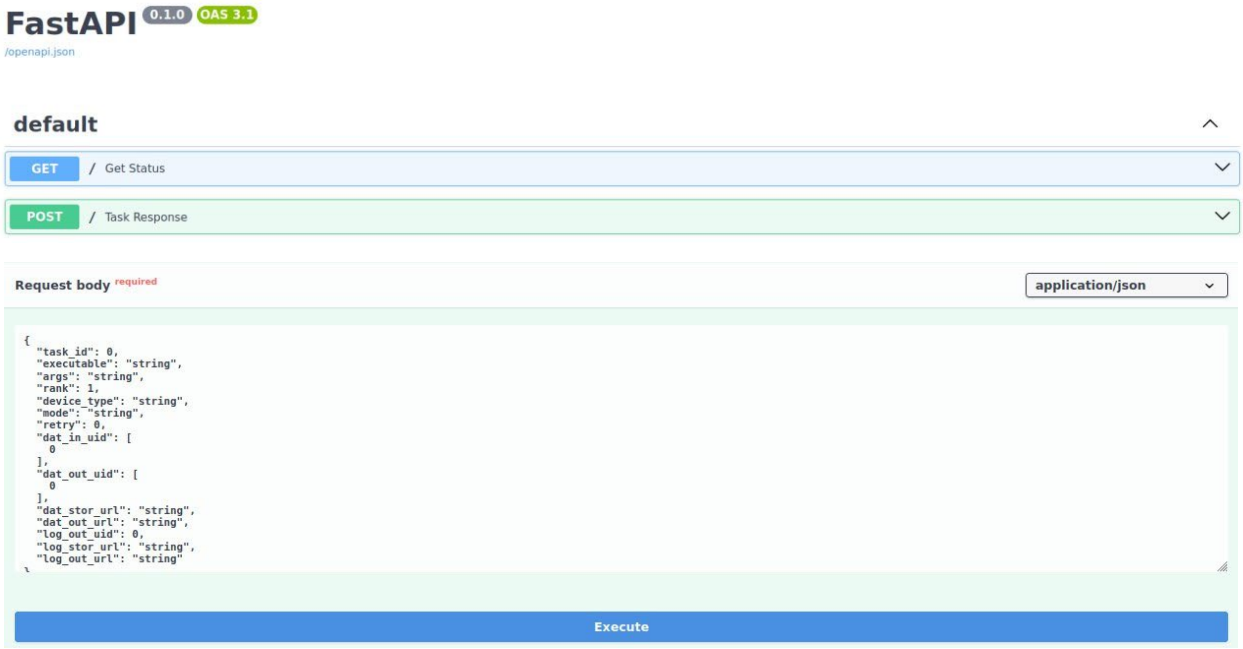
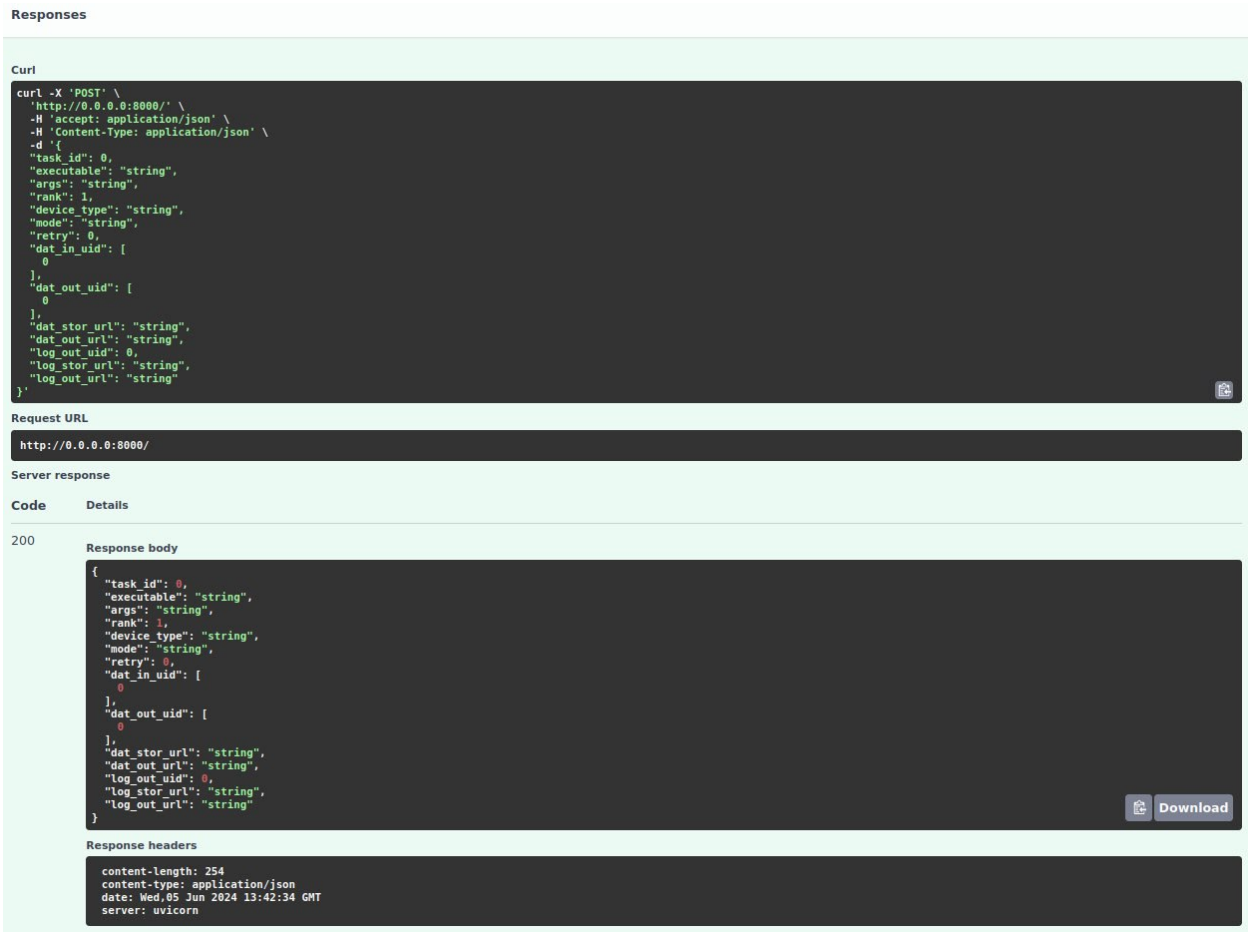


Рис. 3 Отправка задания в SwaggerUI



а)

б)

Рис. 4 Отправка задания в очередь:

а) отправленный запрос и код успешного выполнения;

б) просмотр задания в очереди RabbitMQ

Таким образом, в этом разделе детально описано взаимодействие с Workload Management System (WMS). Описаны зоны ответственности каждой системы и перечислены все необходимые входные и выходные параметры. Для обеспечения высокой эффективности взаимодействия с WMS мы используем различные стратегии и технологии. В случаях, где важна асинхронность, применяются асинхронные запросы для отправки заданий. Это позволяет

выполнять несколько операций параллельно, максимально задействуя ресурсы системы и ускоряя процесс взаимодействия.

4.1 Сервис для опроса WMS

Данный сервис нужен для того, чтобы следить за выполнением задания. Он получает из очереди задание и записывает в свою БД.

Описание базы данных заданий представлена в Табл. 4:

Табл. 4 Описание базы данных заданий

Ключ	Тип	Описание
task_id	int	Уникальный идентификатор задания
executable	str	Исполняемый файл запуска
args	str | None	Флаги для исполняемого файла задания
rank	int	Приоритет задания
device_type	str	Определение выбранного вычислительного ресурса, который будет использоваться для выполнения задания (CPU / GPU)
mode	str	Тип задания (Map / Merge)
retry	int	Максимальное число попыток выполнения задач в случае неудачной попытки
dat_in_uid	list[int]	Уникальные идентификаторы входного датасета
dat_out_uid	list[int]	Уникальные идентификаторы выходного датасета
dat_stor_url	str	Унифицированный указатель ресурса выходного хранилища
dat_out_orl	str	Унифицированный указатель ресурса выходного датасета
log_out_uid	int	Уникальный идентификатор выходного датасета логов
log_stor_url	str	Унифицированный указатель ресурса выходного хранилища для хранения логов
log_out_url	str	Унифицированный указатель ресурса выходного датасета логов
status	str	Статус выполнения задания (in_progress / finished / canceled)

После этого данный сервис периодически запрашивает у WMS информацию о завершении задания. Как только задание завершается, сервис обновляет статус задания в базе данных на "finished" и передает информацию в сервис для генерации новых заданий. Затем он отправляет данные в DMS для очистки промежуточных данных, передавая поле uid (как для обычных датасетов, так и для логов) с помощью RabbitMQ. Также сервис передает информацию о uid входного и выходного датасетов в сервис для генерации заданий. На основе этих данных сервис обновляет статусы датасетов на "finished" и "ready" соответственно и приступает к созданию нового задания для последнего датасета.

5. Стек технологий

Выбор языка программирования:

Весь программный стек SPD основан на Python и C++. Учитывая, что вычисления производятся не в самой системе, для построения микросервисной архитектуры целесообразно выбрать Python.

Выбор асинхронного фреймворка:

Для всех сервисов, не взаимодействующих с оператором обработки данных, используется асинхронный фреймворк для Python — FastAPI [4]. FastAPI — это современный асинхронный фреймворк для создания веб-приложений и REST API на основе Python. Вот основные преимущества, по которым был выбран FastAPI:

- Высокая производительность: FastAPI использует возможности асинхронного программирования и механизмы компиляции для достижения высокой производительности, обрабатывая большое количество запросов с высокой скоростью.

- Поддержка асинхронности: FastAPI полностью поддерживает асинхронное программирование. Он позволяет создавать асинхронные функции-обработчики запросов, использовать асинхронные операции с базами данных и вызывать асинхронные зависимости, что эффективно использует системные ресурсы и обеспечивает хорошую отзывчивость приложения.

- Интеграция с другими инструментами: FastAPI легко интегрируется с другими инструментами и библиотеками Python. Он поддерживает широкий спектр баз данных (SQL и NoSQL), аутентификацию и авторизацию, асинхронные HTTP-клиенты, логирование и многое другое. FastAPI также может быть использован с множеством инструментов для развертывания и контейнеризации, таких как Docker и Kubernetes.

Выбор брокера сообщений:

Брокеры сообщений представляют собой программные модули, обеспечивающие взаимодействие и передачу информации между приложениями, сервисами и системами. Они преобразуют сообщения из

различных протоколов обмена, позволяя связанным службам обмениваться данными независимо от языка программирования или платформы.

Брокеры сообщений выполняют отправку, маршрутизацию, хранение и доставку сообщений конечным получателям. Они действуют как посредники между приложениями, позволяя отправителям отправлять сообщения без необходимости знать точное количество и местоположение получателей.

Использование брокеров сообщений создает гибкую и масштабируемую архитектуру, где различные компоненты системы могут взаимодействовать асинхронно, обмениваясь сообщениями через брокер. Брокеры сообщений широко применяются для реализации распределенных систем, микросервисных архитектур, обработки событий и других сценариев, требующих надежной и гибкой передачи сообщений между компонентами системы.

RabbitMQ [5] — это распределенный брокер сообщений с открытым исходным кодом, предназначенный для эффективной доставки сообщений в сложных сценариях маршрутизации. RabbitMQ функционирует как кластер узлов, где очереди распределяются и реплицируются по узлам для обеспечения высокой доступности и устойчивости к сбоям.

Принцип работы RabbitMQ:

- Отправители (publishers) отправляют сообщения на обменники (exchange).
- Обменники направляют сообщения в очереди и другие обменники.
- При получении сообщения RabbitMQ отправляет подтверждения отправителям.
- Получатели (consumers) поддерживают постоянные TCP-соединения с RabbitMQ и объявляют, какую очередь они получают.
- RabbitMQ доставляет сообщения получателям.
- Получатели отправляют подтверждения о успешном или неудачном получении сообщения.

• После успешного получения сообщение удаляется из очереди.

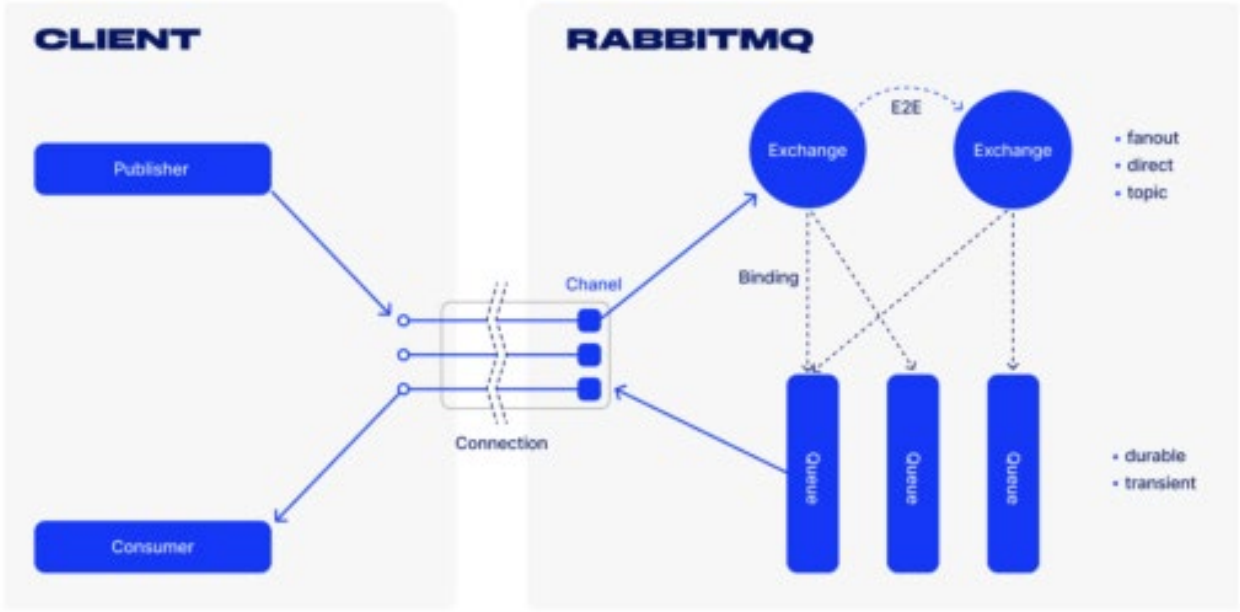


Рис. 5 Общий принцип работы RabbitMQ

Вся инфраструктура разворачивается с использованием docker-compose [6], что позволяет независимо работать с различными сервисами без ограничений по выбору инфраструктуры. Каждый сервис имеет свою собственную базу данных, к которой другие сервисы не имеют прямого доступа, предотвращая узкие места, вызванные одновременной работой нескольких сервисов с одной и той же таблицей.

Более того, такая архитектура обеспечивает возможность горизонтального масштабирования.

ЗАКЛЮЧЕНИЕ

Современные эксперименты в области физики высоких энергий, сталкиваются с необходимостью эффективной обработки огромных объемов данных, генерируемых детекторными системами. Эксперимент SPD на коллайдере NICA предъявляет высокие требования к системе фильтрации данных в реальном времени, учитывая интенсивность событий и необходимость оперативной обработки информации. В этом контексте проектирование системы управления процессами обработки данных становится не просто задачей, а ключевым фактором для достижения успешных результатов в исследованиях физики высоких энергий. Эффективное управление процессами обработки данных в вычислительных комплексах имеет ключевое значение для обеспечения производительности и надежности системы.

В ходе работы были рассмотрены основные аспекты функционирования системы SPD Online Filter эксперимента SPD и её подсистемы - системы управления процессами обработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. International spin physics collaboration at the collider NICA [Электронный ресурс] // http://spd.jinr.ru/

2. Oleynik D. Data processing in HEP experiments. [Электронный ресурс] // https://lit.jinr.ru/sites/lit.jinr.ru/files/pdf/HEPNICA_computing_2022_OleynikD.pdf

3. Abazov V. M. [и др.] Technical Design Report of the Spin Physics Detector at NICA. 2024.arXiv:2404.08317v1 [hep-ex]

4. FastAPI [Электронный ресурс] // https://fastapi.tiangolo.com/

5. RabbitMQ [Электронный ресурс] // https://www.rabbitmq.com/

6. Docker-compose [Электронный ресурс] // https://docs.docker.com/compose/