МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
ФИЗТЕХ-ШКОЛА ФИЗИКИ И ИССЛЕДОВАНИЙ ИМ. Л.Д. ЛАНДАУ
КАФЕДРА «ФУНДАМЕНТАЛЬНЫЕ ВЗАИМОДЕЙСТВИЯ И КОСМОЛОГИЯ»

Дегтярев Артем Георгиевич

Разработка системы метаданных физических событий эксперимента BM@N мегапроекта NICA

Направление подготовки: 03.03.01 «Прикладные математика и физика»

Выпускная квалификационная работа
на соискание степени бакалавра

Работу выполнил:
студент 783а группы _____ Дегтярев Артем Георгиевич

Научный руководитель:
к.ф.-м.н. _____ Климай Петр Александрович

Москва — 2021

Аннотация.

Эксперимент BM@N (Baryonic Matter at Nuclotron) предназначен для изучения свойств ядерной материи при взаимодействии пучков легких и тяжелых ионов с энергиями 1–6 ГэВ на нуклон с фиксированной мишенью. Для проведения необходимого физического анализа реконструированных событий эксперимента важна возможность достаточно быстро выбирать только требуемые события по разным параметрам, для чего в свою очередь необходима разработка системы метаданных событий на базе каталога полученных событий столкновения частиц. В данной работе проведен обзор аналогичных систем, используемых в других экспериментах, сформулированы общие принципы построения СУБД метаданных событий для ускорительных экспериментов, предложена архитектура системы метаданных событий для BM@N, и представлены результаты тестов нескольких промышленных СУБД, которые могут использоваться для хранения метаданных: PostgreSQL, Apache HBase и Cassandra. Результаты работы используются коллаборацией BM@N для построения информационных систем эксперимента.

Оглавление

Стр.

Введение

Реализуемый на территории России проект класса мегасайенс NICA (Nuclotron based Ion Collider fAcility) нацелен на воссоздание и исследование в лаборатории ядерной материи в экстремальных условиях, возникавших в природе на ранних стадиях эволюции Вселенной и в недрах нейтронных звезд.

Современная фундаментальная наука достаточно глубоко продвинулась в понимании законов природы, лежащих в основе нашего мира. Одним из блестящих результатов нового тысячелетия стало экспериментальное подтверждение существования бозона Хиггса, который вместе с кварками и глюонами является фундаментальным кирпичиком материи. Однако механизм формирования и эволюции Вселенной, как и ряд обнаруженных до настоящего времени особенностей окружающего нас мира, на сегодняшний день остаются «terra incognita». Все еще нерешенными проблемами являются загадки наблюдаемой иерархии масс и процесса формирования элементарных частиц из кварков и глюонов, особенности удерживающих их сил внутри адронов, а также поведение материи при экстремально высоких температурах и плотностях и ее новых состояниях, предсказываемых теорией. Эти проблемы являются базисом для понимания природы Вселенной, как в масштабах Космоса, так и в Микромире.

Воссоздание в лабораторных условиях процессов, происходивших во Вселенной на разных стадиях ее эволюции, возможно при помощи современных ускорителей. Так, Большой адронный коллайдер (LHC) в ЦЕРН и коллайдер Брукхейвенской национальной лаборатории RHIC (США) сталкивают пучки частиц от протонов до тяжелых ионов с энергиями от сотен до тысяч миллиардов электронвольт, что позволяет исследовать процессы, происходившие во Вселенной в первые мгновения ее эволюции после Большого взрыва, когда возникли кварки и глюоны — «кирпичики материи».

Для проведения этих исследований в наиболее интересном диапазоне энергий сейчас создается ускорительно-экспериментальный комплекс NICA (ОИЯИ, Дубна, Россия).

Столкновения тяжелых ионов дают уникальную возможность для изучения ядерной материи вплоть до экстремальных плотностей и температур. Во время столкновения ядерная материя нагрета и сжата на очень короткий про-

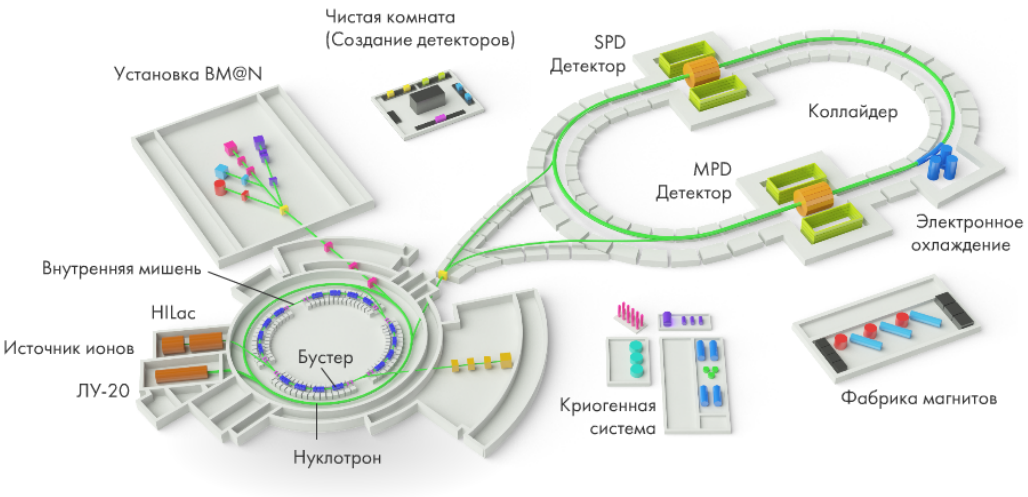

Рисунок 1 — NICA.

межуток времени. На умеренных температурах нуклоны возбуждаются до барионных резонансов, которые распадаются с испусканием мезонов. На высоких температурах кроме этого создаются барион-антибарионные пары. Эта смесь барионов, антибарионов и мезонов, все из которых являются сильно взаимодействующими частицами, обозначается как адронная материя или барионная материя, если барионы доминируют. Если в ней достаточно велика плотность энергии, становится видна кварк-глюонная подструктура нуклонов. При еще более высоких температурах или плотностях адроны плавятся, и составляющие, кварки и глюоны, образуют новую фазу, кварк-глюонную плазму (QGP).

В этих экстремальных условиях могут быть изучены следующие особенности сильно взаимодействующей материи: уравнение состояния (EoS) сильно взаимодействующей материи при высоких температурах и высоких плотностях чистых барионов; микроскопическая структура сильно взаимодействующего вещества как функция температуры и плотности барионов; модификации адронов в среде, которые могут дать информацию о начале восстановления киральной симметрии. Однако теоретические модели предлагают различные возможные сценарии для описания этих свойств сильно взаимодействующей материи. Таким образом, необходимы новые экспериментальные данные с высоким разрешением и статистикой.

Отношение произведенных мезонов к барионам в фаерболе увеличивается с ростом энергии столкновения. Столкновение ядра с ядром при кинетической энергии пучка нуклотрона в диапазоне от 1 до 4,5 ГэВ на нуклон создает фаербол с преобладанием барионов, в отличие от более высоких энергий в RHIC

или SPS. Согласно расчетам транспортной модели QGSM при энергиях нуклотрона плотности нуклонов в зоне столкновения двух ядер золота превышают плотность насыщения в 3 - 4 раза. При этих плотностях нуклоны начинают перекрываться, и ожидается, что при таких экстремальных условиях может произойти начало восстановления киральной симметрии, хотя кварки все еще подвержены конфайнменту. Это проявляется в модификации адронов в среде, в частности, в уширении при столкновении, уменьшении массы векторных мезонов, распадающихся на ди-лептоны, на которые не сильно влияют взаимодействия в конечном состоянии.

Соответствующие степени свободы при энергиях нуклотрона - это, прежде всего, нуклоны и их возбужденные состояния, за которыми следуют легкие и странные мезоны. Также партонные степени свободы должны проявляться в малых объемах пространства-времени и оставлять свои следы в конечных адронных наблюдаемых. В центре внимания экспериментальных исследований будут адроны со странностями, которые рано образуются при столкновении и не присутствуют в начальном состоянии двух сталкивающихся ядер, в отличие от нуклонов, составленных из легких (u, d)-кварков. Измеренные выходы образования легких и странных мезонов, а также гиперонов и антигиперонов показаны на рисунке 2 как функции от энергии нуклон-нуклонных столкновений.

Диапазон энергий пучка тяжелых ионов нуклотрона соответствует $\sqrt{s_{NN}} = 2.3 - 3.5$ ГэВ. Он хорошо подходит для исследования странных мезонов и мульти-странных гиперонов, которые образуются в столкновениях “ядро-ядро” вблизи кинематического порога. Столкновения тяжелых ионов - богатый источник странных частиц, а слияние лямбда-гиперонов с нуклонами может дать множество легких гиперядер. Ожидается, что изучение образования гиперядер позволит по-новому взглянуть на свойства взаимодействий гиперон-нуклон и гиперон-гиперон. На рисунке 3 представлены выходы гиперядер в зависимости от энергии нуклон-нуклонных столкновений в столкновениях Au + Au, предсказанных тепловой моделью. Максимум скорости образования гиперядер прогнозируется на уровне 4–5 ГэВ, что близко к диапазону энергий нуклотрона.

Таким образом, программа исследований столкновений тяжелых ионов на нуклотроне включает следующие темы: исследование динамики реакции и ядер-

ного уравнения состояния, исследование свойств адронов в среде, образование (мульти)-странных гиперонов на пороговых уровнях и поиск гиперядер. Для интерпретации экспериментальных данных по столкновениям тяжелых ионов и обеспечения нормировки измеренных спектров A + A планируется изучение столкновений протонов с протонами, нейтронами и дейтронами.

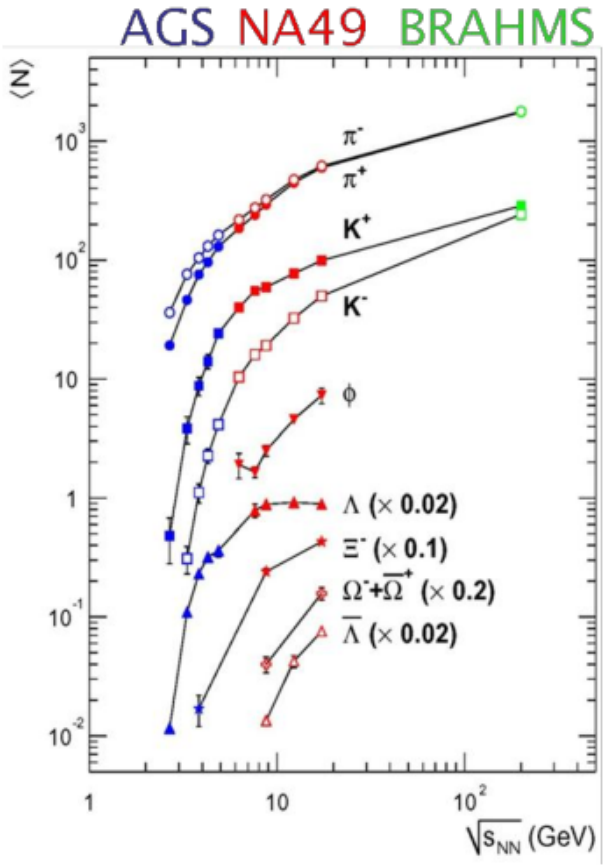

Рисунок 2 — Выходы образования мезонов и (анти-) гиперонов в зависимости от энергии нуклон-нуклонных столкновений. Диапазон энергий пучка тяжелых ионов нуклотрона BM@N соответствует $\sqrt{s_{NN}} = 2.3 - 3.5$ ГэВ.

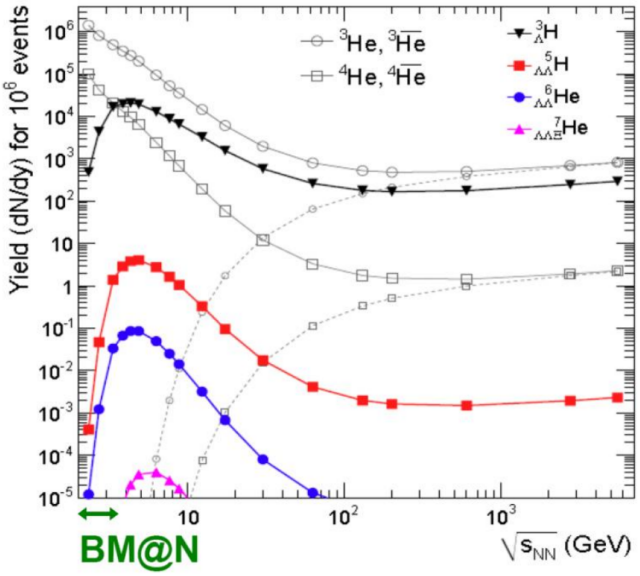

Рисунок 3 — Выходы образования гиперядер в зависимости от энергии нуклон-нуклонных столкновений, рассчитанные с помощью тепловой модели. Диапазон энергий пучка тяжелых ионов нуклотрона BM@N соответствует $\sqrt{s_{NN}} = 2.3 - 3.5$ ГэВ.

Глава 1. Информационные системы событий в экспериментах физики высоких энергий.

1.1 Предназначение и принцип работы информационных систем.

Обзор [1] показал, что различные IT-системы для экспериментов физики высоких энергий используются в большинстве крупных экспериментов по столкновению частиц, являясь их важной частью. Они позволяют автоматизировать работу с данными, в частности хранение и анализ данных. При этом существующие решения существенно зависят от условий конкретного эксперимента.

Частным случаем информационной системы является система метаданных событий. Далее рассматриваем именно такие системы. Главная их идея относительно проста и состоит в том, чтобы создавать метаданные событий вместе со ссылками на события, которые могут быть использованы для анализа. Данные хранятся в различных форматах, от сырых данных до реконструированных событий. Информационные системы используют метаданные, чтобы найти ссылки на конкретные события, отобранные по различным критериям, и уже затем получить события как таковые. Такие системы могут быть реализованы различными способами. Важной их частью являются системы индексации событий. Ниже представлены примеры информационных систем в нескольких крупных экспериментах физики высоких энергий с целью анализа существующих подходов, используемых для проектирования и реализации.

1.2 Информационные системы метаданных в эксперименте ATLAS LHC.

Одна из разрабатываемых информационных систем событий используется в эксперименте ATLAS LHC [2], который имеет большую экспертизу разработки и обновления систем метаданных событий. Ее разработка началась с создания глобального каталога событий “TAG database” на основе СУБД Oracle

и внешнего интерфейса для пользователей. Помимо этого, для создания каталога событий была разработана база данных AMI (Atlas Metadata Interface). TAG DB нужна, в первую очередь, для использования повторяющихся запросов при отборе событий, а также для проверки полноты и согласованности циклов обработки данных. Система использовалась для этих целей в качестве основного инструмента во время запуска LHC Run 1. Сохраненные теги были созданы из данных объекта анализа (Analysis Object Data - AOD) и содержали более 290 атрибутов событий, но, тем не менее, они имели размер в 100 раз меньший, чем соответствующий AOD [3]. Сначала теги хранились только в ROOT-файлах, но затем СУБД Oracle использовалась и для хранения тегов.

Результатом использования тегов стало значительное ускорение времени обработки из-за предварительного отбора событий перед поиском в содержимом AOD. Одним из ограничений TAG DB было отсутствие гибкости схемы базы данных, её было трудно изменять для удовлетворения новых потребностей работы коллаборации. Другой проблемой было то, что после трех лет использования с увеличивающимся объемом данных время отклика в некоторых случаях становилось слишком большим.

Новый проект под названием Event Index [4] был запущен в 2012 году с целью избежать этих ошибок. Главными функциями разрабатываемой системы должны были стать отбор событий (получение ссылок на события в требуемом формате), просмотр событий (получение списка событий, удовлетворяющих заданным условиям), проверка согласованности записи, обнаружение дублирующихся событий и измерение перекрытий цепочек триггеров. Отбор событий - это основное предназначение системы Event Index, поэтому она должна предоставлять указатель на файл, содержащий необходимое событие, после чего оно должно быть извлечено из файла и возвращено отправителю запроса.

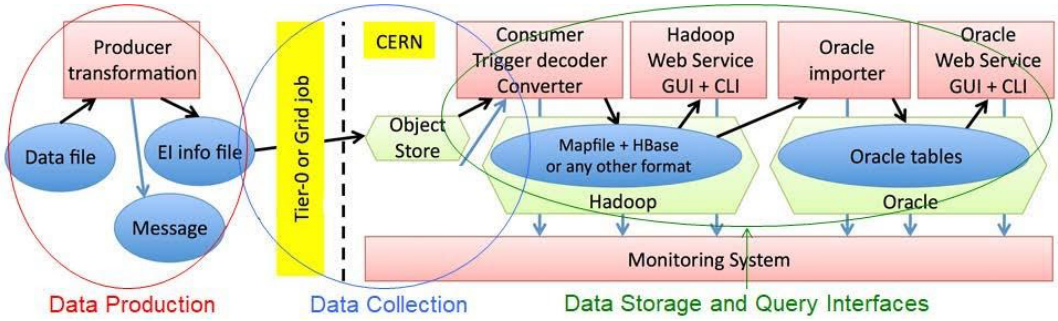

Рисунок 1.1 — Архитектура системы ATLAS EventIndex.

Архитектура системы Event Index показана на рисунке 1.1. Подсистема генерации данных автоматически создает и извлекает метаданные для хранения в EventIndex. Компонент сбора данных передает метаданные в Event Index с производящих сайтов в ЦЕРН, где данные сохраняются во временных файлах в кластере Hadoop. Система хранения обеспечивает доступ к данным. В этом проекте использовались новые технологии, которые в то время стали доступны сообществу разработчиков ПО с открытым исходным кодом. Особого внимания заслуживает экосистема Hadoop [5] с данными, хранящимися в файлах HDFS MapFiles, и внутренний каталог, основанный на распределенном хранилище данных HBase. База данных HBase позволила распараллеливать больше операций, и ATLAS решил использовать ее более широко, в том числе используя Apache Phoenix, обеспечивающий доступ SQL к таблицам данных HBase.

Этот выбор также был основан на решении технической группы, которая отметила [6], что в некоторых случаях базы данных NoSQL лучше подходят для нужд экспериментов, которые могут масштабироваться по горизонтали, поддерживая линейное увеличение рабочей нагрузки. В результате CERN IT начал поддерживать огромный кластер HADOOP, и были тщательно изучены экосистема Hadoop с распределенной файловой системой HDFS, нереляционная СУБД HBase, документоориентированная СУБД MongoDB, СУБД с хранилищами данных типа “ключ-значение” Cassandra и Oracle NoSQL Database. Помимо прочего, Hadoop и Cassandra были использованы в проекте ATLAS Distributed Data Management под названием DQ2.

1.3 Информационные системы метаданных в эксперименте CMS.

Эксперимент CMS [7] использует CouchDB и MongoDB на виртуальных контейнерах, совместно управляемых CMS и CERN IT. MongoDB была успешно использована в CMS для агрегирования данных, но оказалась неудобной для использования в эксперименте ATLAS, поскольку требовала явного разделения данных. CouchDB оказалась очень удобной для целей CMS, особенно для связанных с мониторингом.

Система Data Bookkeeping Service (DBS) была разработана в CMS для предоставления каталога метаданных событий, как смоделированных, так и экспериментальных. Каталог включает в себя всю информацию для отслеживания наборов данных, историю их обработки и связи между запусками, файлами и наборами данных. DBS предоставляет базы данных и службы для хранения и доступа к метаданным, связанным с данными CMS. Веб-службы DBS используют интерфейс REST и формат данных JSON для доступа и коммуникации [8].

Система DBS - это многоуровневое веб-приложение с модульным дизайном, который позволяет легко приспособить ее сразу под несколько СУБД. На начальном этапе она поддерживала три типа баз данных [9]: Oracle, MySQL и SQLite. В качестве системы постоянного хранения метаданных событий используется серверная часть базы данных Oracle. DBS - это объединенная система, в которую могут быть включены как реляционные, так и нереляционные базы данных. Служба агрегации данных (Data Aggregation Service - DAS) была разработана для обеспечения единообразного доступа к распределенным службам данных CMS и для возможности пользователей запрашивать подлежащие сервисы. Она объединяет информацию метаданных из различных служб данных в общие записи для просмотра конечными пользователями. DAS была построена на основе документо-ориентированной MongoDB, которая обеспечивает такие преимущества, как хранение без схемы, встроенный язык запросов и быстрые операции чтения и записи.

1.4 Информационные системы метаданных в эксперименте BES III.

Еще один пример системы метаданных событий, который показывает разнообразие подходов к построению таких систем - событийно-ориентированная система управления данными, которая была разработана для эксперимента BES III [10]. В эксперименте было несколько различных подходов для хранения данных и отбора событий. Сначала данные хранились в файлах ROOT, но затем было предложено [11] хранить их в таблицах HBase для ускорения обработки и анализа. В качестве первого шага разработчики эксперимента добавили мета-

данные уровня событий с такими параметрами, как номера запусков, решения триггеров высокого уровня, свойства событий (например, энергия) и данные частиц (типы частиц, треки и другие) в файлы ROOT. Решение было выбрано на основании того факта, что часто только одно из тысячи событий в файлах данных ROOT было полезно для физического анализа BESIII [12]. Эксперимент также хранил эти данные в реляционной базе данных.

Последние изменения, которые были разработаны и реализованы в BESIII, основаны на концепции, согласно которой метаданные уровня событий извлекаются и хранятся в базе данных NoSQL, в то время как реальные данные событий хранятся в файлах ROOT. EventIndexer, который является частью нового дизайна, использует кластер HBase для создания высокоуровневых индексов событий. HBase использует лексикографически упорядоченную структуру индекса для создания первичного ключа и кэширования его в памяти, что обеспечивает высокую эффективность запросов. Введена технология инвертированного индекса, имя и значение атрибута события разработаны как Rowkey, так что двоичный поиск может выполняться по первичному ключу. События, удовлетворяющие одному и тому же условию запроса, объединяются в одну запись HBase и возвращаются как результат запроса. Еще одно усовершенствование системы информации о событиях в эксперименте - это служба кэша событий, которая отвечает за кэширование данных о событиях на SSD или в оперативной памяти. Данные кэшируются на уровне события, а не всего файла.

Обзор показал, что подходы к использованию метаданных событий в экспериментах физики высоких энергий могут сильно различаться. Как реляционные, так и нереляционные базы данных используются в соответствующих системах для хранения метаданных.

Система метаданных событий находится в стадии разработки для экспериментов NICA. Чтобы понять ее место в процессе обработки данных, в следующем разделе представлена текущая модель данных событий, используемая в экспериментах проекта NICA.

Глава 2. Модель данных событий для экспериментов NICA.

Схема процесса обработки данных показана на рисунке 2.1. Передача данных от электроники детектора до хранилища данных организована как последовательный конвейер [13]. Система триггеров выполняет отбор событий в эксперименте в режиме онлайн. После приема последовательности триггерных сигналов, запрашивающих сбор данных, выбранные элементы детекторов генерируют фрагменты данных, которые передаются по оптическим каналам и собираются в события с помощью Event Builder. Затем события отправляются во временное хранилище данных - распределенную кластерную систему хранения с низкой задержкой и высокой пропускной способностью, основанную на устройствах твердотельной памяти. События также используются в операторных онлайн-системах, выполняющих проверку качества сырых данных, построение гистограмм и мониторинг событий. Из временного хранилища данные переносятся в постоянное хранилище данных, построенное с использованием экономичных компонентов, без особых требований ко времени задержки. Здесь данные преобразуются в формат ROOT [14] (иерархические деревья среды CERN ROOT, содержащие ветви с экспериментальными данными), затем происходит реконструкция событий и в распределенных вычислительных системах выполняется физический анализ с использованием программных фреймворков экспериментов NICA.

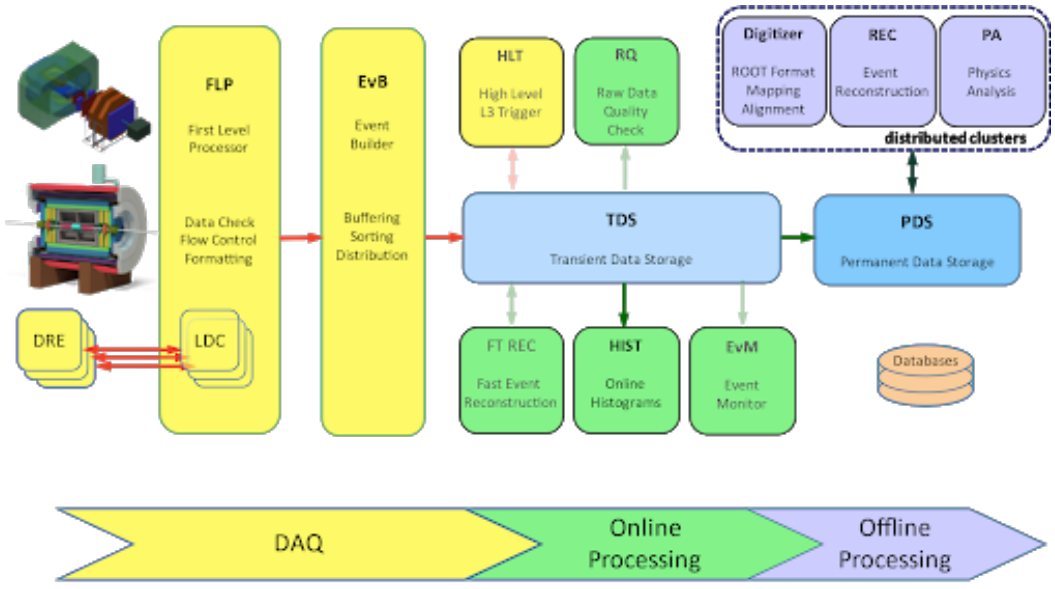

Рисунок 2.1 — Схема процесса обработки данных с детектора.

Программные фреймворки экспериментов NICA, предназначенные для обработки как экспериментальных, так и смоделированных данных, основаны на среде FairRoot [15]. Программное обеспечение FairRoot - это объектно-ориентированный пакет для моделирования, реконструкции и анализа данных, разработанный для экспериментов FAIR (Facility for Antiproton and Ion Research) в Институте GSI в Дармштадте (Германия). Основное назначение FairRoot состоит в том, что он предоставляет унифицированный пакет с общими механизмами для решения наиболее часто возникающих задач в физике высоких энергий. Этот пакет включает в себя основное программное обеспечение по моделированию детектора и автономному анализу данных. Он позволяет пользователям разрабатывать и конструировать модели детекторов и решать задачи анализа простым способом, предоставляя, кроме того, некоторые общие функции.

Пакет FairRoot используется в экспериментах NICA в качестве основы для главных программных сред [16] - BmnRoot, MPDRoot и SPDRoot, которые разрабатываются для экспериментов BM@N, MPD и SPD соответственно. Фреймворки NICA разработаны для исследования производительности детекторов, моделирования событий и разработки алгоритмов, которые будут использоваться для реконструкции и физического анализа событий столкновения частиц. Они реализованы на языке программирования C++ и основаны на среде CERN ROOT и пакете FairRoot. Основные задачи фреймворков выполняются с помощью макросов ROOT (интерпретируемых файлов с командами ROOT), которые соответствуют задачам моделирования, оцифровки, реконструкции и различных задач физического анализа.

Обработка данных событий в рамках NICA осуществляется в соответствии со следующими этапами (рисунок 2.2). Необработанные экспериментальные данные, полученные из системы сбора данных (DAQ - data acquisition system) в специальном двоичном формате, оцифровываются и преобразуются в формат ROOT с помощью цифрового преобразователя (макроса оцифровки). Полученный цифровой файл содержит заголовок и иерархическое ROOT-дерево с заголовками событий и числами, сгруппированными по событиям для всех детекторов эксперимента. Макрос реконструкции восстанавливает информацию о произведенных частицах, зарегистрированную детекторами, об их импульсах, типах, треках и других кинематических параметрах из информации, содержащейся в необработанных данных детектора. Восстановленная информация со-

хранится в файле DST (data summary tape - лента сводных данных). Файлы DST содержат ROOT-деревья с реконструированными данными сотен тысяч событий столкновения частиц, таких как соударения, треки частиц, первичные вершины и другие. Таким образом, сами физические события хранятся в двоичных или ROOT-файлах в зависимости от стадии обработки и часто объединяются в наборы данных (datasets).

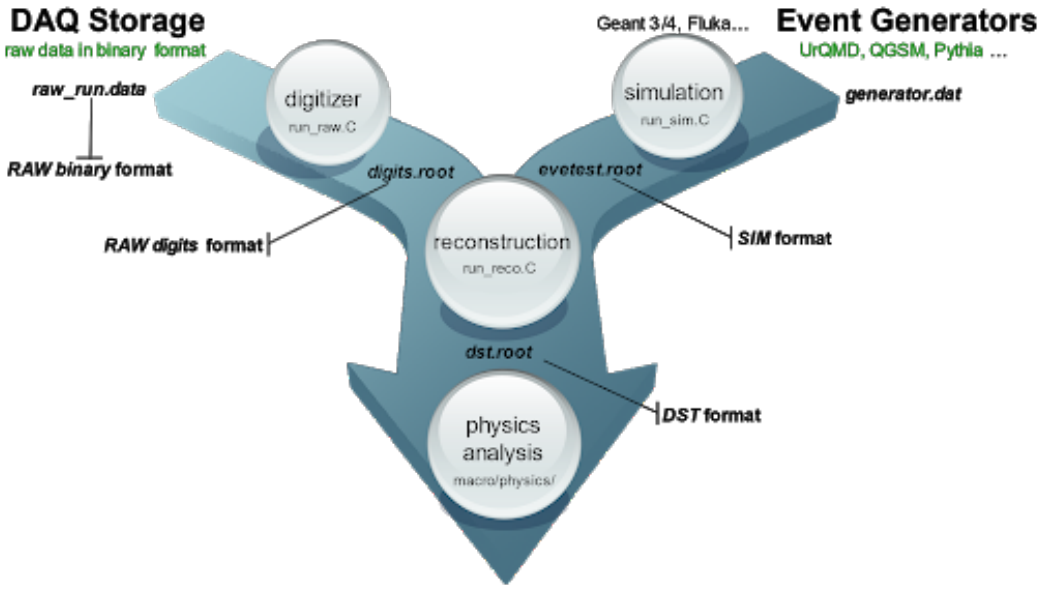

Рисунок 2.2 — Обработка данных событий во фреймворках эксперимента.

На последнем этапе различные виды физического анализа данных реконструированных событий используются для исследования свойств ядерной материи, образующейся при столкновениях. Участники коллаборации проводят физический анализ полученных файлов DST, которые являются входными для анализа в макросах ROOT. Задачи физического анализа часто выполняются в автономных распределенных вычислительных системах экспериментов NICA с файлами, содержащими реконструированные данные событий в иерархических ROOT-деревьях. Эти DST-файлы, хранящиеся в распределенных файловых системах, выбираются вручную для анализа, или формируется набор входных файлов DST с использованием базы данных условий [17] в соответствии с заданными критериями.

База данных условий была разработана как централизованное хранилище различной информации, необходимой для обработки данных о событиях, и хранит, в частности, метаданные о проведенных экспериментах, такие как сеанс (period number) и запуск (run number), соответствующий одному файлу необра-

ботанных данных, полученному из системы DAQ, время начала и окончания, частицы в пучке и тип цели, энергия пучка, информация о запуске, магнитное поле (значение напряжения), количество событий, путь к исходному файлу и размер. Эти метаданные можно использовать для выбора необходимых экспериментальных файлов с данными событий для анализа, но иногда требуется выбрать не набор файлов данных эксперимента (полные запуски) с сотнями тысяч событий, а список только тех событий, которые удовлетворяют определенным условиям, например, события, содержащие редкие распады, полученные с заданной логикой триггеров или с номерами треков, превышающими пороговое значение.

В результате, требуется новая информационная система на основе базы данных метаданных событий, называемая каталогом событий, предназначенная для хранения необходимой сводной информации о событиях столкновения эксперимента, чтобы только события, необходимые для физического анализа, быстро находились и выбирались в соответствии с заданными параметрами. Система метаданных событий, основанная на каталоге событий, обеспечит управление информацией о событиях, храня ссылки на них, а также триггеры, используемые в онлайн-системе сбора данных, список реконструированных частиц и другие метаданные событий, необходимые для удобного их поиска. Информация актуальна для оперативного доступа к необходимым событиям при автономном анализе. Кроме того, метаданные событий зависят от уровня данных (необработанные, цифровые, DST, ESD, AOD и прочие). Эта система также отвечает за создание, поддержание и проверку качества каталога физических событий и выполняет индексацию событий, для быстрого их поиска по заданным критериям. Кроме того, для использования выбранных событий в задачах анализа, система метаданных событий должна быть интегрирована с фреймворками эксперимента. В следующем разделе представлены дизайн и архитектура системы метаданных событий, завершение которой намечено на 2021 год.

Глава 3. Архитектура системы метаданных событий для экспериментов NICA.

Система метаданных событий, основанная на базе данных событий, содержит метаданные физических событий, которые включают в себя конкретную информацию о событиях столкновения частиц, что позволяет пользователям быстро искать требуемый набор событий на основе различных критериев и параметров и выбирать восстановленные события для дальнейшей обработки и физического анализа. Для определения необходимого набора метаданных событий, которые будут храниться в каталоге событий, необходимо проанализировать текущую структуру реконструированных данных экспериментов. Формат данных восстановленных событий варьируется в экспериментах NICA, ниже рассматривается структура файлов DST для эксперимента BM@N, первого эксперимента проекта NICA.

Файлы DST эксперимента BM@N содержат следующие данные:

1. Заголовок файла с некоторой служебной информацией.
2. DST-заголовок запуска с соответствующими номерами сеансов (Run) и номерами запусков, временами начала и окончания запусков, хранящимися в файле, информацией о частицах в пучке, его энергии, фиксированной цели и магнитном поле.
3. иерархическое ROOT-дерево (с именем «bmndata»), которое включает восстановленные данные событий для сотен тысяч событий столкновения частиц, произошедших в ходе запуска. Каждое событие, в свою очередь, содержит ветви дерева с заголовком события DST, который включает в себя номер события, время начала события, типы триггеров, прицельный параметр, общие заряды до и после столкновения и некоторую конкретную информацию с триггеров, содержащую массивы реконструированных событий и сегменты локальных треков для всех детекторов, массив глобальных треков, данные, полученные калориметрами, и информацию о первичной вершине (точке взаимодействия).

Сводные метаданные событий, которые будут храниться в базе данных событий, должны быть выбраны так, чтобы обеспечить возможность удобного поиска и выбора требуемых событий эксперимента по возможным запросам

участников коллаборации для физического анализа. Реконструированная информация о событиях отличается в разных экспериментах NICA, поэтому конкретный набор атрибутов метаданных будет определен на этапе развертывания системы метаданных событий с использованием файла конфигурации. Метаданные запуска уже хранятся в разработанной базе данных условий, также настраиваемой на этапе развертывания. Метаданные запуска можно использовать для выбора реконструированных файлов с событиями экспериментов в формате DST или для получения соответствующих номеров запусков, удовлетворяющих условиям. Каталог событий содержит сводные данные для выбора необходимых событий экспериментов NICA по критериям, таким как:

- period number (Run) и run number;
- software version;
- event time (начального момента времени достаточно, чтобы не хранить соответствующий интервал времени);
- число первичных и всех реконструированных треков - primary tracks и all tracks;
- числа треков положительно и отрицательно заряженных частиц из первичной вершины;
- обнаруженные первичные и вторичные частицы;
- количество срабатываний детекторов;
- суммарный начальный и конечный заряд в событии;
- и прочие используемые критерии для отбора необходимых реконструированных событий.

Кроме того, система метаданных событий может использовать метаданные запуска, которые перечислены выше и хранятся в базе данных условий, для предварительного отбора событий при указании соответствующих условий. Метаданные базы данных условий будут использоваться на первом этапе отбора событий по глобальным критериям запуска, затем каталог событий будет обрабатывать только отфильтрованные события за полученный период и номера запусков. В результате метаданные с номерами запусков, хранящиеся в базе данных условий, должны соответствовать метаданным физических событий, соответствующих запускам и хранящимся в каталоге событий. Общая коммуникационная схема развивающейся информационной системы представлена на рисунке 3.1.

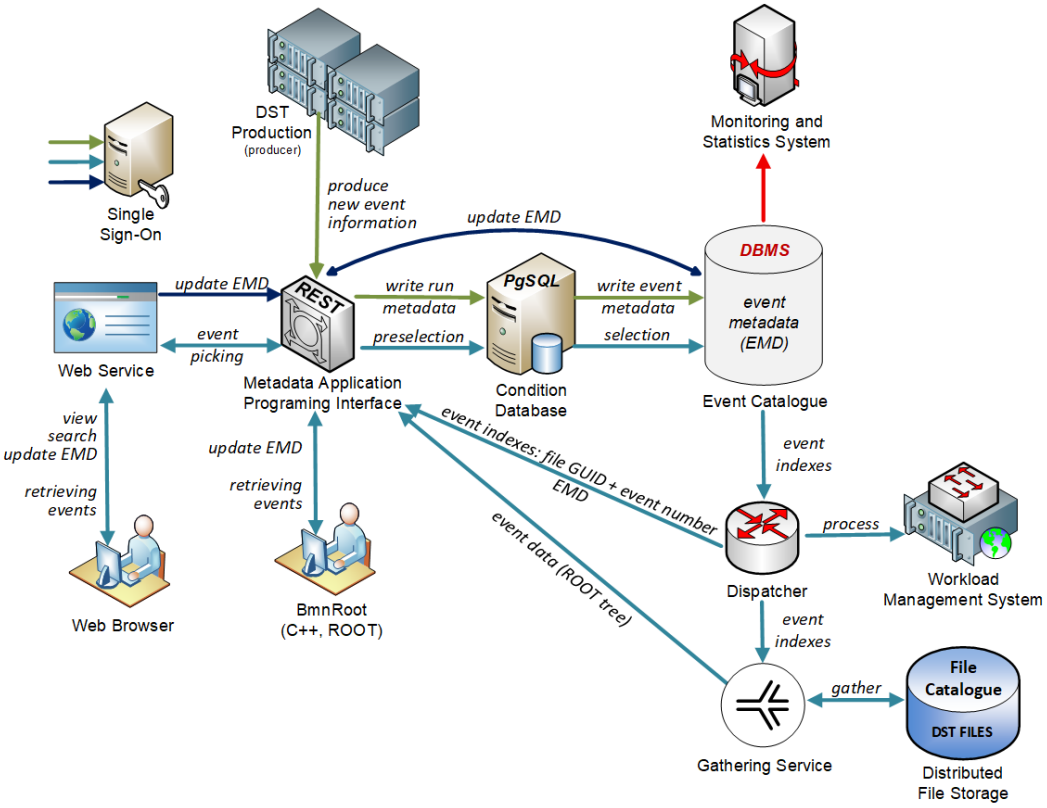

Рисунок 3.1 — Архитектура системы метаданных событий.

Метаданные событий, в дополнение к тегам, характеризующим события столкновений, хранят в файлах ссылки на соответствующие события и информацию о процессе обработки событий, например, относятся ли они к полным данным объекта анализа (Analysis Object Data - AOD) или коротким производным физическим данным (Derived Physics Data - DPD) для конкретного физического анализа. Каталог событий будет идентифицировать хранящиеся в нем события с помощью указателя файла, определяющего ссылку на файл с данными и номер события (уникальное целое число в прогоне) в файле, который однозначно определяет событие и позволяет найти его в распределенных файловых системах эксперимента. Вместо указателя файла при использовании каталога файлов, например Rucio [18], предпочтительно использовать идентификатор файла GUID, выполняющего требование, чтобы каждое событие имело отдельный идентификатор.

Система метаданных событий будет обеспечивать следующие основные функции:

– общее описание событий столкновения частиц и их идентификаторов, которое может быть использовано для отбора событий;

– запись и хранение необходимых метаданных событий в Каталоге событий;
– управление и обеспечение удобного доступа к метаданным;
– организация соответствующих онлайн и офлайн-интерфейсов.

Система метаданных событий предоставляет методы для задания условий отбора событий по заданным критериям, а также для извлечения определенных событий для дальнейшей обработки и физического анализа. Необходимые события могут быть запрошены в трех различных режимах: получение списка событий (с указателями на события в файлах или GUID-индексы), удовлетворяющих определенным условиям, загрузка набора выбранных событий, запуск обработки выбранных событий через систему управления рабочей нагрузкой.

Информация, записанная в базу данных метаданных событий, должна обеспечивать эффективный доступ к требуемым событиям без лишних операций ввода и вывода. Следует учитывать, что файлы данных экспериментов с событиями распределены между разными вычислительными центрами. Информационная система позволит членам коллаборации работать с коллекциями метаданных для физических событий, хранящихся в восстановленных файлах DST в распределенных хранилищах. База данных событий должна соответствовать файлам с реконструированными событиями, поэтому изменения в полученных файлах или в метаданных событий должны быть синхронизированы между собой. Кроме того, система управления рабочей нагрузкой эксперимента запускает различные пользовательские задачи в вычислительных системах, которые взаимодействуют с системой метаданных событий для выбора только требуемых для анализа событий. Внедрение системы включает разработку и поддержку инструментов, которые обеспечивают доступ к этим метаданным и получение событий, удовлетворяющих заданным условиям. В результате каталог событий должен обеспечивать механизмы интеграции с другими системами, размещение метаданных и управление службами записи, а также иметь низкое время отклика и высокую доступность. Информационная система отвечает за создание, поддержание и контроль качества каталога событий.

Система метаданных событий предоставляет веб-сервис с графическим интерфейсом, к которому будут иметь доступ участники коллаборации, для просмотра и поиска метаданных событий, хранящихся в каталоге событий, а также для получения отобранных событий. Ее можно использовать для возвра-

та идентификаторов событий и отображения сведений о найденных событиях. Должен быть реализован специализированный API для записи новых метаданных в каталог событий и для запроса событий, отобранных по критериям для их обработки в программном обеспечении эксперимента. Важно, чтобы система информации о событиях работала со всеми платформами, поддерживаемыми коллаборацией NICA, которые в настоящее время являются распространенными разновидностями Linux, такими как CentOS, Scientific Linux, Debian, Ubuntu.

Аутентификация в системе метаданных событий будет выполняться через единую систему аутентификации и авторизации, такую как LDAP, FreeIPA или SSO (Single Sign-On), как это делается для других служб экспериментов в соответствии с назначенной ролью. Существует три категории пользователей по уровню доступа к системе метаданных событий. Соответствующие функции: запись метаданных, администрирование метаданных, использование событий.

Все категории пользователей могут просматривать каталог событий для просмотра идентификаторов событий и метаданных, поиска и извлечения физических событий, удовлетворяющих заданным условиям и параметрам, но только записывающий и администратор могут индексировать восстановленные события, записывать метаданные событий в Каталог и обновлять текущие записи. Кроме того, администратор метаданных, имеющий полный доступ, может удалить существующие метаданные событий из Каталога, например, в случае внесения неправильных записей.

Предполагается, что количество полученных событий увеличится с текущего значения в сотни миллионов событий BM@N до миллиардов событий в год для каждого эксперимента NICA, включая будущие эксперименты MPD и SPD. Например, коллаборация MPD планирует собирать около 19 миллиардов необработанных событий в год [19]. Чтобы оценить приблизительный размер метаданных для 19 миллиардов событий, которые будут храниться в каталоге событий, предположим, что метаданные содержат 5 целочисленных значений по 4 байта и 5 целочисленных значений по 8 байтов, тогда общий размер метаданных событий составляет около одного терабайта. Не все полученные события будут проиндексированы в каталог, но тем не менее такое большое количество хранимых метаданных приводит к необходимости быстрых масштабируемых решений.

Система метаданных событий должна иметь высокую производительность, так как она должна подходить для записи большого количества метаданных для всех реконструированных событий, сохраненных в файлах ROOT, и, что особенно важно, обеспечивать одновременный доступ для большого количества заданий, контролируемых системой управления рабочей нагрузкой, которая распределяет их по вычислительным ресурсам. Более того, время извлечения событий для желаемого анализа должно быть в допустимых пределах. Поэтому выбор системы управления базами данных (СУБД) для реализации каталога событий, обеспечивающей эффективную обработку миллиардов записей, является одним из ключевых моментов при разработке. В следующем разделе представлено проведенное сравнение различных SQL и NoSQL СУБД и оценки производительности обработки записей тестовых баз данных для следующих систем: PostgreSQL, Hadoop HBase и Apache Cassandra.

Глава 4. Тестирование СУБД.

4.1 Реляционные и нереляционные СУБД

Выбор СУБД было решено начать со сравнительного обзора существующих систем. Они подразделяются на два вида: реляционные и нереляционные (или, как их еще называют, SQL и NoSQL). В экспериментах физики высоких энергий системы индексации и метаданных событий используют оба вида СУБД, поскольку в зависимости от особенностей эксперимента и те, и другие имеют свои преимущества, для оценки которых нужно провести тестирование. При этом чаще используются реляционные СУБД, однако такой выбор не всегда может быть оправдан. В одной из рассмотренных статей [20] приведено сравнение MySQL и HBase, одних из наиболее популярных СУБД, широко используемых в физике высоких энергий. Авторы приходят к выводу, что в различных аспектах HBase либо превосходит MySQL, либо дает похожий результат. Для сравнения были использованы реальные данные эксперимента BESIII Beijing Spectrometer, три датасета на 425 миллионов, 4 миллиарда и 39 миллиардов событий, записанных в ROOT-файлы. Данная статья представляла интерес еще и потому, что формат и объем данных были похожи на планируемые в экспериментах NICA.

Преимуществами нереляционных СУБД являются лучшая горизонтальная масштабируемость, децентрализованность и отказоустойчивость, обычно более высокая производительность, объектно-ориентированность и как следствие возможность проще изменять структуру данных в процессе работы с БД. Преимуществом реляционных баз данных является большая структурированность, гибкость запросов и простота администрирования. Обзор показал, что наиболее популярной и удобной для подобных задач среди реляционных является PostgreSQL, а среди нереляционных - HBase и Cassandra. Соответственно, эти СУБД были выбраны для дальнейшего тестирования.

4.2 База данных и платформы для тестирования

Было решено провести тестирование на тестовой базе данных, которая по своей структуре будет похожа на проектируемый каталог событий и будет содержать 500 миллионов событий со следующими атрибутами:

- software_id (int, 2 byte) = 0...3.
- period_number (int, 4 byte) = 1...7.
- run_number (int, 4 byte) = 1...1000.
- event_number (int, 4 byte) = 0...200000.
- detector_hit (int, 4 byte) = 0...max_int.
- primary_vertex (boolean, 1 byte) = false...true.
- if primary_vertex is false: primary_tracks (int, 4 byte) = 0, all_tracks (int, 4 byte) = 0...20; if primary_vertex is true: primary_tracks (int, 4 byte) = 1...30, all_tracks (int, 4 byte) = primary_tracks...300.
- positive_tracks (int, 4 byte) = 0...primary_tracks.
- particles (int, 4 byte) = 0...max_int.

Для оценки и сравнения различных СУБД были предложены 4 тестовых запроса, для каждого из которых оценивалось время ответа:

1. Select 1: period_number=6 AND software_id=0 AND primary_tracks > 5;
2. Select 2: period_number=6 AND software_id=2 AND primary_vertex = false AND detector_hit[bits:0-5];
3. Select 3: period_number=5 AND run_number > 100 AND primary_vertex = true AND particles[bits:5-9] > 4;
4. Select 4: period_number=4 AND software_id=1 AND primary_tracks>6 and all_tracks>40 AND particles[bits:10-14].

Для оценки производительности созданных баз данных использовались две аппаратные платформы. Первая аппаратная платформа (аппаратная конфигурация 1) представлена гипервизором XenServer на сервере Dell PowerEdge FX430, который имеет следующие характеристики ЦП, памяти, хранилища и сети: 2 x Intel Xeon E5-2680, 256 ГБ (240 ГБ используется) DDR4 2133 МГц, 2 x Intel SSD SC1BG400G4R, 10 Гбит / с Ethernet. Для хранения баз данных используется только один SSD-диск размером 400 ГБ, второй хранит только си-

стемную информацию. В Конфигурации 1 использовались следующие версии программного обеспечения: Scientific Linux 7.9, PostgreSQL 12.5, HBase 2.2.3, Hadoop 3.2.1.

Вторая аппаратная платформа (аппаратная конфигурация 2) основана на современном пользовательском персональном компьютере без виртуализации и имеет следующую конфигурацию: Intel Core i9-10900F 2,80 ГГц (20 ядер ЦП), 64 ГБ ОЗУ, 1 ТБ NVMe SSD-диск. Она использует следующее программное обеспечение: CentOS Linux 8.2, PostgreSQL 12.5, Apache Cassandra 3.11.8. Результаты тестов производительности, полученные для данных конфигураций и баз данных, представлены ниже.

4.3 PostgreSQL, аппаратная конфигурация 1.

PostgreSQL - это мощная объектно-реляционная СУБД с открытым исходным кодом, активная разработка которой насчитывает более 30 лет, что принесло ей репутацию надежной, функциональной и производительной. Она поддерживает индексы, которые являются общепринятым способом повышения производительности базы данных. Индекс позволяет серверу базы данных находить и извлекать определенные строки намного быстрее, чем он мог бы сделать без индекса, но индексы также добавляют накладные расходы системе в целом, поэтому их следует использовать разумно. Тестовые данные были сгенерированы случайным образом, и индексы не были добавлены в текущие тесты, но потенциально они могут повысить производительность каталога событий.

Левый график на рисунке 4.1 показывает зависимость времени выполнения запросов тестовых данных от количества записей метаданных событий, хранящихся в базе данных PostgreSQL. Видно, что время выполнения запросов увеличивается почти линейно. Время выполнения Select 3 намного больше, чем время выполнения других. Разницу во времени можно объяснить достаточно большим количеством записей результатов в Select 3, который представлен на правом графике на рисунке 4.1.

Дисковое пространство, которое занимает база данных PostgreSQL, оценивалось для разного количества записей метаданных событий, хранящихся в

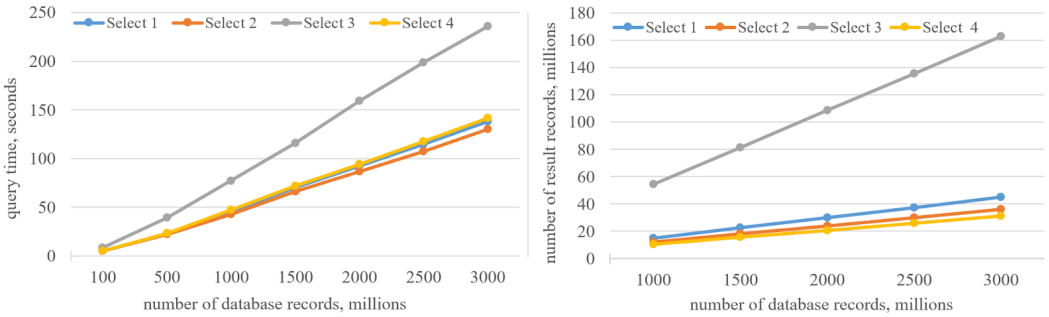

Рисунок 4.1 — Времена выполнения (левый график) и число возвращаемых результатов (правый график) для тестовых запросов в зависимости от числа хранимых записей о метаданных событий.

базе данных (левый график на рисунке 4.2). Размер, занимаемый тремя миллиардами записей, составлял около 300 гигабайт, но у сервера было только 240 ГБ оперативной памяти для кеширования. Несмотря на это, графики времени выполнения на рисунке 4.1 были линейными. Поэтому было проведено дополнительное исследование времени выполнения для случая, когда не хватает оперативной памяти для кэширования 3 миллиардов записей, хранящихся в базе данных. Для этой цели размер ОЗУ XenServer был ограничен набором меньших значений. Правый график на рисунке 4.2 показывает резкий перепад между 220 ГБ и 225 ГБ ОЗУ, что указывает на нехватку памяти для кэширования.

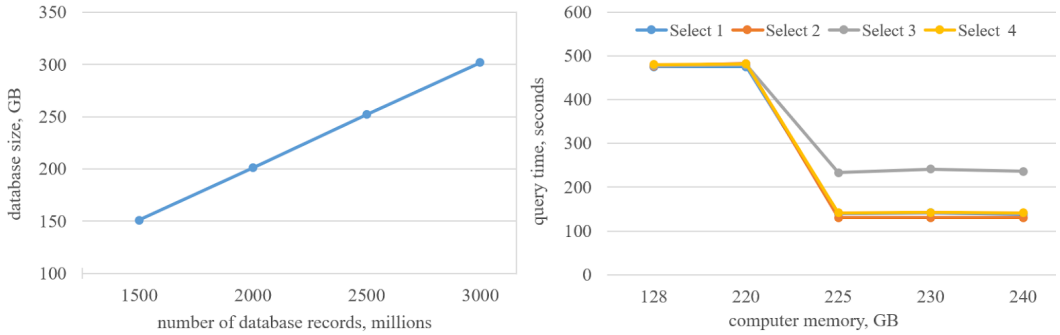

Рисунок 4.2 — Левый график: размер тестовой базы данных в зависимости от числа хранимых записей. Правый график: времена выполнения тестовых запросов в зависимости от размера ОЗУ сервера.

4.4 PostgreSQL, аппаратная конфигурация 2.

Кроме того, для оценки производительности созданной базы данных PostgreSQL с помощью созданных записей метаданных событий была использована аппаратная конфигурация 2.

Было получено линейное увеличение времени запроса для Select 1-4, аналогичное левому графику на рисунке 4.1. Время запроса для того же размера базы данных и записей с конфигурацией оборудования 2 меньше. Например, для 500 миллионов записей в базе данных событий (примерно 71 ГБ дискового пространства) времена запросов Select 1-4 составляют 11,2 с, 10,5 с, 15,0 с и 11,6 с, соответственно (рисунок 4.3). Следует отметить, что для 500 миллионов записей событий база данных событий может быть полностью кэширована в ОЗУ, как это показано на рисунке 4.1.

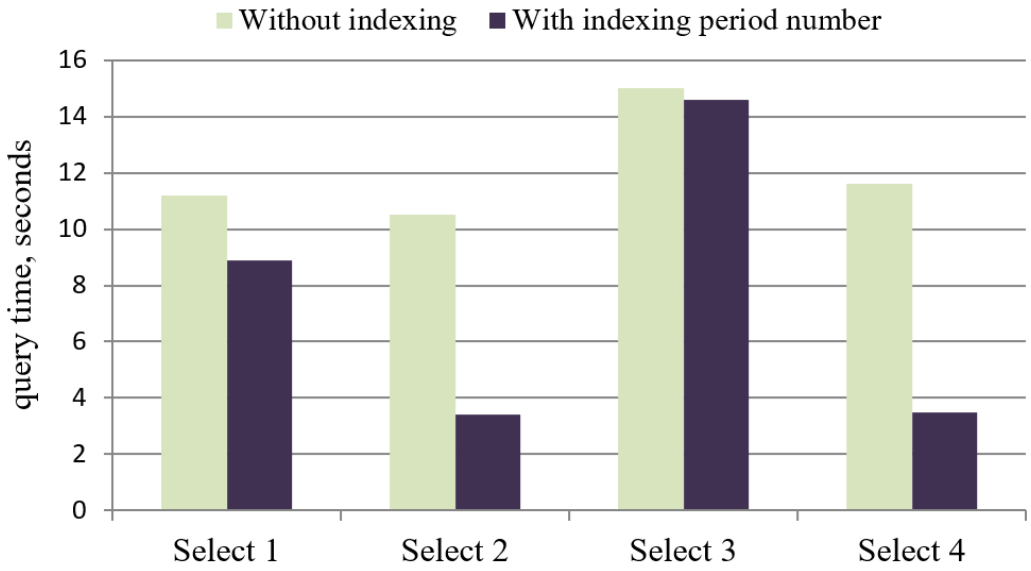

Рисунок 4.3 — Времена выполнения тестовых запросов для конфигурации 2 для БД с 500 миллионами событий.

Без пользовательской индексации PostgreSQL выполняет полное сканирование таблицы для отбора данных на основе критериев фильтрации. Чтобы проверить, как индексирование может улучшить производительность, был создан индекс B-tree по атрибуту period_number. После создания индекса время запроса для Select 1-4 было значительно сокращено и составило 8,9 с, 3,4 с, 14,6 с, 3,5 с соответственно. В результате ожидается, что настройка индексов для ба-

зы данных событий в PostgreSQL позволит повысить производительность после загрузки реальных данных о событиях в каталог событий.

4.5 Hadoop HBase

Hadoop HBase [21] - одно из хранилищ данных NoSQL, доступных в экосистеме Apache Hadoop. Это многомерная, распределенная и масштабируемая NoSQL СУБД с открытым исходным кодом, написанная на Java. HBase работает поверх HDFS (распределенной файловой системы Hadoop) и предназначен для обеспечения отказоустойчивого механизма хранения больших коллекций разреженных наборов данных. HBase может обеспечить высокую пропускную способность и низкую задержку, обеспечивая более быстрый доступ для чтения / записи огромных наборов данных.

HBase - это база данных, ориентированная на столбцы, то есть данные хранятся в ячейках, сгруппированных в столбцы, а не в строки. Столбцы логически сгруппированы в семейства столбцов, которые могут быть созданы во время определения схемы или в процессе работы с базой данных. Базы данных, ориентированные на столбцы, хранят все ячейки, соответствующие столбцу как непрерывные записи на диске, что значительно ускоряет доступ и поиск.

В выполняемых тестах используется псевдораспределенный режим конфигурации, что означает, что Hadoop HBase полностью запускается на одном хосте, но каждый сетевой сервис HBase выполняется как отдельный процесс. Кроме того, Apache Phoenix [22], механизм SQL с открытым исходным кодом для HBase, использовал стандартный JDBC API вместо обычного клиентского API HBase для создания таблиц базы данных событий, вставки метаданных и запроса необходимой информации о событиях. Он принимает SQL-запрос, компилирует его в серию сканирований HBase и организует их выполнение для получения обычных наборов результатов JDBC. Apache Phoenix обеспечивает обработку транзакций в реальном времени и операционную аналитику в Hadoop для приложений с малой задержкой.

Были исследованы три конфигурации базы данных с HBase.

- HBase C1. Одна большая таблица со всеми метаданными событий, без Apache Phoenix.
- HBase C2. Одна большая таблица со всеми метаданными событий, с Apache Phoenix
- HBase C3. Метаданные событий распределены по таблицам в зависимости от значений полей первичного ключа (software_id, period_number, run_number). Комбинация полей уникальна для каждой таблицы. Apache Phoenix не используется.

Из-за специфики определения данных и запросов в системе Hadoop HBase были сформулированы и использованы следующие тесты, которые проще, чем приведенные выше:

- Test 0: get all records (select count(*));
- Test 1: period_number = 6 AND software_id = 0 AND primary_tracks > 5;
- Test 2: period_number = 6 AND software_id = 2 AND primary_vertex = false;
- Test 3: period_number = 4 AND software_id = 1 AND primary_tracks > 6 and all_tracks > 40;
- Test 4: period_number = 5 AND run_number = 27 AND primary_vertex = true;
- Test 5: period_number = 5 AND run_number > 308 AND primary_vertex = true.

В таблице 1 показано время выполнения тестовых запросов для трех конфигураций базы данных событий на HBase, в которой хранится 500 миллионов записей. Последняя конфигурация, HBase C3 с разделением метаданных событий на несколько таблиц, демонстрирует существенно более высокую эффективность. Таким образом, этот подход наиболее предпочтителен при использовании баз данных на платформе HBase, но он требует тщательного выбора полей первичного ключа для разделения таблицы.

	HBase C1	HBase C2	HBase C3
Test 0	56 min	28 min	63 min
Test 1	29 min 55 sec	28 min 02 sec	5 min
Test 2	32 min 4 sec	28 min 47 sec	11 min
Test 3	30 min 20 sec	29 min 52 sec	8 min
Test 4	28 min	not supported by Apache Phoenix	1 min 52 sec
Test 5	29 min	not supported by Apache Phoenix	2 min 12 sec

Таблица 1 — Времена выполнения тестовых запросов для различных конфигураций HBase.

4.6 Apache Cassandra

Apache Cassandra [23] – распределенная нереляционная система управления базами данных с открытым исходным кодом. Единицей данных в Cassandra является столбец – пара «имя-значение». Столбцы, описывающие все сущности, объединенные общим значением или набором значений, группируются в широкие строки, к которым можно обращаться по ключу раздела (partition key). Также реализована возможность обращаться к части строки по кластерному ключу (clustering key), который также позволяет осуществить группировку данных. Кластерному ключу соответствует подмножество набора сущностей, соответствующего уникальному значению partition key, объединенное набором значений других параметров (названий столбцов) сущностей. Partition key и clustering key составляют primary key. Кроме того, есть вторичные ключи (индексы), которые также ускоряют доступ к данным. Таким образом, столбец соответствует одному параметру одной сущности, а широкая строка – нескольким сущностям, сгруппированным по набору параметров.

Таблица – это контейнер для упорядоченной коллекции строк, каждая из которых является контейнером для упорядоченной коллекции столбцов. Важное отличие от реляционной модели здесь состоит в том, что таблица может быть разреженной, то есть каждый столбец у одних сущностей может быть задан, а у других – нет, то есть, нет необходимости хранить null в качестве признака.

В Cassandra реализован язык CQL (Cassandra Query Language), который частично повторяет возможности SQL, но имеет ряд ограничений, вызванных

способом хранения данных. В частности, нет операций JOIN, оператора OR в WHERE, ограничены возможности фильтрации данных (в частности, побитовые операции) и использование операций сравнения. Для того чтобы запросы вида SELECT ... FROM ... WHERE ... были эффективны, необходимо указывать в WHERE все названия столбцов, из которых состоит partition key. Из этого и некоторых других требований следует, что правильный подход к хранению данных в Cassandra включает в себя проектирование таблиц по возможности под планируемые запросы и, соответственно, дублирование данных для использования в разных запросах.

Рассмотрим подробнее процесс построения СУБД для конкретного запроса (Select 1): period_number=6 AND software_id=0 AND primary_tracks $>$ 5.

Первый вывод состоит в том, что нужно группировать данные по (file_id, event_number), т.к. количество уникальных значений первичного ключа должно быть равно числу событий. Если делать эту группировку на уровне partition key, в этом случае мы имеем много "узких" строк вместо меньшего числа "широких", что является неправильным подходом и нивелирует преимущества нереляционной СУБД Cassandra. Следовательно, (file_id, event_number) должно являться clustering key или его частью.

В качестве partition key удобно взять (period_number, software_id, primary_tracks), либо часть этих параметров, а оставшуюся часть добавить слева в clustering key. По условию задачи, выборка по period_number и software_id производится почти всегда. Кроме того, строки не должны быть слишком широкими, так как существует ограничение в 2 миллиарда значений в одной строке. Из этого следует, что есть только 2 возможных построения PRIMARY KEY:

1. ((period_number, software_id, primary_tracks), file_id, event_number)
2. ((period_number, software_id), primary_tracks, file_id, event_number).

Для записи данных и выполнения запросов был установлен cassandra-driver для Python. Временем выполнения запроса считалось суммарное время подключения к кластеру, получения результатов и итерации по ним. Результаты выполнения для обоих ключей представлены в таблице 2 (для тестирования была создана БД с 500 миллионами событий):

Для event_number и * большая часть времени относится к итерации по запросам, поэтому в этом случае нет существенной разницы, какой из ключей использовать. Но полученное соотношение времен запросов для COUNT(*)

SELECT	PRIMARY KEY 1	PRIMARY KEY 2
event_number	19,1	19,6
*	29,5	28,5
COUNT(*)	1,5	1,8

Таблица 2 — Времена выполнения для первого запроса

сохраняется с ростом объема данных, поэтому primary_tracks лучше включить в partition key. Кроме того, было обнаружено, что эффективнее использовать вместо одного запроса, возвращающего большой объем данных и использующего операцию сравнения (period_number > 5) несколько подзапросов (period_number=6, period_number=7...). К примеру, это ускоряет выполнение запроса SELECT COUNT(*) примерно в 8-10 раз. Также для ускорения выполнения были использованы асинхронные запросы. Времена в таблице выше получены при их использовании.

Таким образом, можно предположить, что интересующие параметры при небольшом количестве их уникальных значений лучше делать частью partition key. При большом числе уникальных значений (например, run_number (INT) = 1...1000) в случае использования операции сравнения уже нет смысла включать параметр в partition key, т.к. это увеличит количество запросов до 1000 раз, что негативно скажется на времени выполнения запроса. Кроме того, clustering key, в отличие от partition key, дает возможность сортировки данных, что также может быть удобно для запросов по параметру с большим числом уникальных значений.

Если под каждый из запросов для тестирования создавать свою таблицу (т.е. дублировать данные с разными первичными ключами), то оптимальный вид ключей соответствует указанному в таблице 3.

В случае данной задачи на этапе тестирования и на этапе проектирования могут быть известны не все запросы, которые потребуются. Следовательно, требовалось понять, как создать базу данных, эффективно работающую на некотором множестве известных запросов, а именно на четырех тестовых, и оценить времена работы различных запросов к ней. При этом учитывалось:

1. Крайние слева параметры в первичном ключе должны быть именами столбцов, которые будут использоваться наиболее часто в планируе-

SELECT [...] FROM q.events WHERE	PRIMARY KEY
period_number = 7 AND software_id = 3 AND primary_tracks > 5	((period_number, software_id, primary_tracks), file_id, event_number)
period_number = 6 AND software_id = 2 AND primary_vertex = false	((period_number, software_id, primary_vertex), file_id, event_number)
period_number = 5 AND run_number>100 AND primary_vertex = true	((period_number, primary_vertex), run_number, file_id, event_number)
period_number = 4 AND software_id = 1 AND primary_tracks>6 AND all_tracks>40	((period_number, software_id, primary_tracks), all_tracks, file_id, event_number)

Таблица 3 — Оптимальные первичные ключи

мых запросах. В случае каталога событий BM@N, это period_number и software_id.

2. Крайние справа параметры в первичном ключе должны быть выбраны так, чтобы выполнялось условие уникальности ключей и при этом обеспечивалась возможность обращения к отдельным сущностям (событиям). В случае каталога событий BM@N, это file_id и event_number.
3. В partition key должны быть имена столбцов с наименьшим числом уникальных значений параметра для уменьшения числа используемых асинхронных запросов.
4. Имена столбцов с большим числом уникальных значений должны быть в clustering key.
5. В partition key не должно быть слишком много или слишком мало имен столбцов, а именно: строки должны быть широкими, но должно удовлетворяться ограничение в 2 миллиарда значений в строке (с учетом дальнейшего роста объема данных на BM@N).
6. Число столбцов в первичном ключе должно быть таким, чтобы не было слишком большого числа подразделов (для уменьшения числа асинхронных запросов).

7. В первичном ключе не должно быть столбцов со слишком большим числом уникальных значений. В случае каталога событий BM@N, это detector_hit и particles.

На основе этих требований был выбран PRIMARY KEY ((period_number, software_id, primary_vertex, primary_tracks), run_number, file_id, event_number). Run_number является частью clustering key в силу большого числа уникальных значений. Primary_vertex является частью распределительного ключа, т.к. он либо ускоряет запрос (например, второй), либо увеличивает число запросов всего в 2 раза. All_tracks не был добавлен в clustering key в силу п.7, потому что в этом случае число запросов для третьего или четвертого примера будет больше 12 миллионов, что очень неэффективно, т.к. время обращения к базе данных для каждого из асинхронных запросов составляет десятки миллисекунд, и суммарно оно существенно превысит время итерации по результату, которое обычно является определяющим для времени выполнения запроса.

Аналогичным образом можно создавать другие таблицы (с дублированием данных) с различными ключами, удовлетворяющими описанным выше требованиям.

Созданная база данных была протестирована на различных объемах данных (от 500 миллионов до 2 миллиардов событий), результаты представлены в таблице ниже.

Наиболее быстрыми оказались запросы COUNT(*), возвращающие число событий, потому что они не предполагают итерации по длинному “списку” результатов. Наиболее медленные запросы третьи, т.к. только в этом случае мы используем разбиение на подзапросы по run_number, из-за этого получается порядка сотни тысяч подзапросов, что явно больше желаемого числа подзапросов, которое составляет порядка нескольких тысяч и не больше, если мы хотим, чтобы время выполнения запроса укладывалось примерно в одну минуту.

Анализ также показал, что, как и ожидалось, времена выполнения первого, второго и четвертого запросов как на event_number, так и на все параметры событий, обусловлены главным образом временем итерации по результату, потому что зависимость времени выполнения запроса от числа возвращаемых событий является близкой к линейной. Такой результат связан с разбиением на небольшое число подзапросов.

Первый запрос		
Число записей	SELECT event_number	SELECT COUNT(*)
0.5B	5,5	1,7
1B	10,8	3,3
1.5B	17,4	4,9
2B	24,7	6,6
Второй запрос		
0.5B	6,5	1,8
1B	16,6	3,8
1.5B	21,1	5,6
2B	33,1	7,3
Третий запрос		
0.5B	77,1	10,5
1B	170,0	10,8
1.5B	242,0	12,2
2B	741,2	13,8
Четвертый запрос		
0.5B	5,1	1,7
1B	10,6	3,3
1.5B	16,0	4,9
2B	21,4	7,6

Таблица 4 — Времена выполнения запросов

4.7 Результаты тестирования

Проведенные тесты СУБД для реализации каталога событий показали, что реляционная база данных, построенная на СУБД PostgreSQL, обеспечивает приемлемое время выполнения запросов для выбора событий на современной аппаратной конфигурации и проста в настройке. Кроме того, более точная настройка индексов базы данных для хранимых метаданных событий может значительно улучшить производительность отбора событий. Более того, современные версии PostgreSQL предоставляют хранилища, ориентированные на столбцы, и решения для развертывания серверов распределенных баз данных для масштабирования обработки.

Протестированные нереляционные СУБД также могут продемонстрировать хорошее время получения метаданных. Однако настроить такие базы данных намного сложнее, и выбранная структура может подходить хорошо для одних типов запросов, но показывать низкую производительность для других, для которых она не предназначена. Кроме того, для эффективной работы с базами данных NoSQL необходимо развернуть и поддерживать кластер серверов баз данных.

Из-за потребности в дополнительной экспертизе в области баз данных NoSQL, которой в настоящее время недостаточно, их сложной конфигурации, неуниверсальности структуры базы данных для всех возможных запросов пользователей, необходимости поиска ресурсов для развертывания и обслуживания современного кластера серверов, а также потому что база данных событий на PostgreSQL показала приемлемую производительность для отбора необходимых событий для использования в физическом анализе, для разработки каталога событий была выбрана СУБД PostgreSQL. В то же время, дальнейшие исследования нереляционных баз данных и их производительности для реализации Event Database планируется продолжить на распределенных аппаратных ресурсах ОИЯИ.

Заключение

Был проведен обзор различных подходов к построению информационных систем в экспериментах физики высоких энергий. В частности, рассмотрены существующие системы метаданных событий в ускорительных экспериментах. Обоснована необходимость создания подобных систем в экспериментах NICA и рассмотрена архитектура создаваемой системы метаданных событий.

Был проведен обзор различных СУБД (систем управления базами данных) и тестирование трех наиболее часто использующихся в ускорительных экспериментах: PostgreSQL, HBase и Cassandra, подробно описаны подходы к построению соответствующих баз данных для максимизации их производительности. В результате тестирования была выбрана PostgreSQL на роль СУБД для каталога событий - ключевого элемента системы метаданных событий.

Планируются дальнейшее исследования реляционных и нереляционных баз данных и их производительности для использования в экспериментах мегапроекта NICA. В данный момент, разрабатываются пользовательские API-интерфейсы и дорабатывается каталог событий.

Настоящая работа выполнена в Лаборатории методов ядерно-физического эксперимента МФТИ. Автор благодарит своего научного руководителя П.А. Климая за постановку задачи, помощь и поддержку в ходе выполнения работы. Автор благодарен коллективу лаборатории методов ядерно-физического эксперимента МФТИ, преподавателям Института Ядерных Исследований РАН за теоретическую подготовку по физике, коллективу Объединенного Института Ядерных Исследований за сотрудничество при выполнении работы. Данное исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-02-40125

Список литературы

1. E. Alexandrov, I. Alexandrov, K. Gertsenberger, et al., Information Systems for Online and Offline Data Processing in Modern High-Energy Physics Experiments, Modern Information Technologies and IT-Education, No. 15(3), 654-671 (2019).

2. ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3, S08003 (2008).

3. W. Ehrenfeld, R. Buckingham, J. Cranshaw, et al., Using TAGs to speed up the ATLAS analysis process, J. Phys. Conf. Ser. 331, 032007 (2011).

4. D. Barberis, J. Cranshaw, A. Favareto, et al., The ATLAS EventIndex: Full chain deployment and first operation, Nuclear and Particle Physics Proceedings, Vol. 273-275, 913–918 (2016).

5. S. Bhosale, Prof. Devendra P. Gadekar A Review Paper on Big Data and Hadoop, International Journal of Scientific and Research Publications, Volume 4, Issue 10, (2014)

6. D. Barberis, D. Dykstra, R. Bartoldus,, et al., Report of the WLCG Database Technical Evolution Group, (CERN, 2012), p. 24.

7. CMS Collaboration, The CMS experiment at the CERN LHC, JINST 3, S08004 (2008).

8. M. Giffels, Y. Guo, D. Riley, Data Bookkeeping Service 3 - Providing event metadata in CMS, J. Phys.: Conf. Ser. 513, 042022 (2014).

9. A. Afaq, A. Dolgert, Y. Guo, et al., The CMS Dataset Bookkeeping Service, J. Phys. Conf. Ser. 119, 072001 (2008).

10. Y. Cheng, H. Li, Q. Xu, et al., EventDB: an event-based indexer and caching system for BESIII experiment, EPJ Web of Conferences 214, 04011 (2019).

11. X. Lei, Q. Li, B. Kan, et al., BESIII Physics Data Storing and Processing on HBase and MapReduce, J. Phys. Conf. Ser. 664, 072032 (2015).

12. B. Liu, High performance computing activities in hadron spectroscopy at BESIII, J. Phys. Conf. Ser. 523, 012008 (2014).

13. A. Baskakov, S. Bazylev, A. Fediunin and I. Filippov, MPD Data Acquisition System: Technical Design Report, JINR, Dubna, 2018, 74 p.

14. R. Brun, F. Rademakers, ROOT – An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Nucl. Inst. Meth. in Phys. Res. A. 389, 81-86 (1997).

15. Al-Turany M., Bertini D., Karabowicz R., et al, The FairRoot framework, Journal of Physics: Conference Series 396, 022001 (2012).

16. K. Gertsenberger, S. Merts, O. Rogachevsky, A. Zinchenko, Simulation and analysis software for the NICA experiments, European Physical Journal A 52, 214 (2016).

17. K. Gertsenberger, A. Chebotov, I. Alexandrov, et al., Development of the Condition Database for online and offline processing data of experimental setups of the NICA complex, Supercomputer technologies: proceedings of young scientists, 154-158 (2020).

18. M. Barisits, T. Beermann, F. Berghaus, et al., Rucio: Scientific Data Management, Computing and Software for Big Science 3, 11 (2019).

19. MPD Collab., The MultiPurpose Detector – MPD. Conceptual Design Report (JINR, Dubna, 2012), p. 259.

20. S. Dai, et al., Evaluating Index Systems of High Energy Physics, p.20 (2018)

21. H. Patel, HBase: A NoSQL Database, Technical Report, p. 14 (2017).

22. S. Akhtar, R. Magham, Pro Apache Phoenix. An SQL Driver for HBase, (Apress Media, 2017), p. 140.

23. S. Alapati, Expert Apache Cassandra Administration, (Apress, Berkeley, CA, 2018), p. 467.