ОБЪЕДИНЁННЫЙ ИНСТИТУТ ЯДЕРНЫХ ИССЛЕДОВАНИЙ

ЛАБОРАТОРИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

На правах рукописи

Александров Евгений Игоревич

Разработка специализированных баз данных и информационных систем для экспериментальной физики высоких энергий

Специальность 2.3.5 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата технических наук

Дубна – 2023

Работа выполнена в Лаборатории информационных технологий Объединенного института ядерных исследований.

Научный руководитель:	*Иванов Виктор Владимирович, доктор физико-математических наук главный научный сотрудник ЛИТ ОИЯИ, профессор МИФИ*
Официальные оппоненты:	*Мисюрин Сергей Юрьевич, доктор физико-математических наук, профессор кафедры прикладной математики НИЯУ МИФИ.* *Крюков Александр Павлович, кандидат физико-математических наук, заведующий лабораторией НИИЯФ МГУ.*

С электронной версией диссертации можно ознакомиться на официальном сайте Объединенного института ядерных исследований в информационно-телекоммуникационной сети «Интернет» по адресу: https://dissertations.jinr.ru. С печатной версией диссертации можно ознакомиться в Научно-технической библиотеке ОИЯИ.

Ученый секретарь диссертационного
совета ОИЯИ. 05.01.2022.П
доктор физ.-мат. наук Е. В. Земляная

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы

Для современных научных исследований в области экспериментальной физики высоких энергий (ФВЭ) характерны длительность, сложность, высокая трудоемкость, большие временные и финансовые затраты, оперирование большими объемами данных, регистрируемых в ходе эксперимента. В частности, в ходе сеанса Run2 ATLAS системой сбора и обработки данных было получено около 20 биллионов событий. Ожидается, что в эксперименте MPD при беспрерывной работе в течение 4 – 6 месяцев в году и частоте ядро-ядерных соударений 7 кГц будет набираться до 20 миллиардов событий в год. В этой связи, особую актуальность приобретает задача автоматизации процесса сбора, обработки и анализа экспериментальных данных. Автоматизация современного физического эксперимента невозможна без применения информационно–вычислительного обеспечения, позволяющего собирать, хранить и обрабатывать большое количество информации, управлять экспериментом в процессе его проведения, обслуживать одновременно большое количество оборудования экспериментальной установки, а также выполнять другие действия, необходимые для своевременного получения качественных физических результатов.

Таким образом, крайне актуальными являются работы по разработке, внедрению и развитию автоматизированных информационных систем (АИС), обеспечивающих выполнение вышеуказанных задач, а также работы по созданию и модернизации баз данных (БД), являющихся основой функционирования АИС. Различные БД, такие, как геометрические, конфигурационные, БД состояний, БД метаданных и БД временных рядов, классифицируемые как по типу хранимой информации, так и по характеру организации данных, применяются практически во всех экспериментах ФВЭ. Однако, несмотря на схожесть задач, возникающих в разных экспериментах, с учетом специфики эксперимента, разные типы БД и соответствующие АИС необходимо разрабатывать для каждого эксперимента заново. Это еще более повышает актуальность разработки таких систем для каждого конкретного эксперимента. Кроме того, поиск и анализ признаков, по которым можно адаптировать рассматриваемые типы систем для разных экспериментов, является крайне важной задачей.

Целью диссертационной работы является разработка методик, методов и алгоритмов для создания специализированных БД и соответствующих информационных систем (ИС) для автоматизации процессов сбора, обработки и

анализа данных в экспериментах ФВЭ. В рамках диссертационной работы решались следующие задачи:

- Анализ существующих специализированных БД и ИС, используемых в экспериментальной ФВЭ.
- Развитие методов построения геометрических ИС, базирующихся на пакете FAIRRoot.
- Разработка гибких сервисов для автоматизированного поиска событий в экспериментах ФВЭ.
- Разработка методов построения конфигурационных ИС.
- Разработка метода интеграции динамического дерева в платформу Grafana.
- Создание алгоритма переноса данных из базы COOL DB в новый для эксперимента ATLAS формат, разработанный в рамках проекта CREST.
- Применение развитых методов, технологий и алгоритмов для разработки программного обеспечения (ПО), реализующего конкретные прикладные задачи.

Подходы и методы

Для решения поставленных задач в диссертации использованы принципы системного подхода для выявления проблемной ситуации и постановки задачи, методы объектно-ориентированного анализа для определения объектной модели, выявления функциональных задач, формулировки требований к их проектированию и реализации, графические методы для отображения объектов в виде образа системы для представления в обобщенном виде системных структур и связей, методы организации взаимодействия программ, программных систем и глобально распределенной обработки данных.

Научная новизна

1. Развита методика построения геометрических информационная систем для экспериментов физики высоких энергий, базирующихся на пакете FAIRRoot.
2. Разработана технология построения гибких сервисов для автоматизированного поиска событий в задачах физики высоких энергий.
3. Разработана методика построения конфигурационных информационная систем для экспериментов физики высоких энергий.
4. Разработан метод интеграции динамического дерева в платформу Grafana.
5. Разработан алгоритм конвертирования данных из COOL в CREST для баз данных состояний эксперимента ATLAS.

Практическая ценность

- Для проекта CBM реализован прототип геометрической ИС, включающей в себя геометрическую БД и API (Application Programming Interface) для использования ее в программах моделирования и реконструкции. Данное ПО включает хранение геометрической модели экспериментов на уровне root-файлов детекторов, поддержку разных версий геометрии, включая возможность загрузки классов на основе хранящейся в БД информации.
- Для проекта BM@N реализована геометрическая ИС.
- Геометрическая БД и соответствующее ПО могут быть адаптированы для задач эксперимента MPD путем конкретных настроек.
- В рамках проекта Event Index эксперимента ATLAS разработан гибкий сервис для автоматизированного поиска событий. Настройка рабочих процессов с помощью БД включает в себя как возможность задания порядка запуска задач, используемых для поиска или сбора событий по конкретным параметрам, так и задание различных алгоритмов обработки ошибок, возникающих в ходе работы, что обеспечивает гибкость сервиса.
- В рамках совершенствования систем мониторинга в проектах Event Index разработан и впервые реализован метод для хранения в БД временных рядов InfluxDB с учетом принятой в эксперименте ATLAS политики безопасности. Использование Grafana в связке с InfluxDB позволило увеличить скорость и гибкость мониторинга**.**
- Создана система мониторинга информационных сетей эксперимента ATLAS на основе динамического дерева, реализованного в рамках платформы Grafana.
- Разработано ПО для конвертирования данных из БД ORACLE Condition DB в новый формат, развитый в проекте CREST. Данное ПО предназначено для использования в LHC RUN4.
- Для эксперимента BM@N создана конфигурационная ИС, включающая конфигурационный менеджер. Рассматриваемая ИС путем конкретных настроек может быть применена в экспериментах MPD и SPD.

Положения, выносимые на защиту

- Методика построения геометрических информационных систем для экспериментов в области физики высоких энергий, на основе пакета FAIRRoot.
- Методика разработки конфигурационных информационный систем для задач физики высоких энергий.

- Технология разработки гибких сервисов для автоматизированного поиска событий в экспериментах физики высоких энергий.
- Метод построения системы мониторинга компьютерных сетей на основе динамического дерева, реализованного в рамках платформы Grafana.
- Алгоритм конвертирования данных из базы данных состояний, основанной на COOL, в новый формат системы CREST.

Апробация и степень достоверности

Основные положения и результаты работы докладывались и обсуждались на научных семинарах ЛИТ, рабочих совещаниях экспериментов BM@N, ATLAS, CBM и на различных международных конференциях, в том числе:

- Advanced Computing and Analysis Techniques in Physics Research (2021)
- Distributed Computing and Grid-technologies in Science and Education (2016, 2018, 2021)
- Mathematical Modeling and Computational Physics (2019)
- Symposium on Nuclear Electronics and Computing (2017,2019)
- Conference on Computing in High Energy and Nuclear Physics (2018, 2019).

Все реализованные системы были приняты к эксплуатации. Достоверность исследования подтверждается практическим использованием программного обеспечения в экспериментах ATLAS и BM@N и письмами от координатора разработки программной части эксперимента BM@N проекта NICA Константина Герценбергера, ATLAS EventIndex PI профессора Dario Barberis, ATLAS DAQ/HTL Network Coordinator Eukeni Pozo Astigarraga и ATLAS Database And Metadata Coordinator Andrea Formica.

Публикации и личный вклад

В основу диссертации положены 19 работ, 18 из которых опубликованы в рецензируемых изданиях, соответствующих требованиям к публикациям Положения о присуждении ученых степеней в ОИЯИ (приказ ОИЯИ от 11.02.2022 № 132).

Все предложенные методы, методики и алгоритмы, представленные в диссертации, были сформулированы при определяющем участии автора. Прототипы ИС, разработанные на основе указанных методик, за исключением реализаций WEB интерфейсов, реализованы лично автором диссертации. Алгоритм конвертирования данных из БД состояний, основанной на COOL, в новый формат системы CREST реализованы лично автором диссертации.

Соответствие диссертации паспорту специальности

В диссертационной работе присутствуют результаты в трех областях, соответствующих пунктам 3, 7 и 8 паспорта специальности:

3: Модели, методы, архитектуры, алгоритмы, языки и программные инструменты организации взаимодействия программ и программных систем.

7: Модели, методы, архитектуры, алгоритмы, форматы, протоколы и программные средства человеко-машинных интерфейсов, компьютерной графики, визуализации, обработки изображений и видеоданных, систем виртуальной реальности, многомодального взаимодействия в социокиберфизических системах.

8: Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования.

Структура и объем работы

Диссертация состоит из 5 глав, введения, заключения и приложения, содержит 101 страниц, включает 27 рисунков и библиографию из 95 наименований.

СОДЕРЖАНИЕ РАБОТЫ

Во введении обоснована актуальность выбранной темы, определены цель и задачи, решаемые в работе. Отражена практическая ценность полученных результатов и приведены сведения об апробации результатов диссертационной работы. Изложена структура диссертационной работы. Дано краткое описание экспериментов ФВЭ, обсуждаемых в диссертации и представлен краткий обзор решений по построению БД и соответствующих ИС для тех типов БД, которым и посвящены исследования настоящей диссертации, а именно: геометрических БД, конфигурационных БД, БД состояний, БД метаданных и БД временных рядов.

В последнее время в мире запущено или готовится к запуску достаточно большое число значимых экспериментов в области ФВЭ. Наиболее интересными из них, в том числе, с точки зрения новизны в области принимаемых решений по сбору и обработке информации, являются эксперименты на Большом адронном коллайдере (LHC) в CERN. Это такие эксперименты, как ATLAS, CMS, ALICE и эксперимент LHCb. Представляют интерес, в том числе, и являющиеся частью мега-проекта НИКА эксперименты BM@N, MPD и SPD, а также, готовящийся к запуску в ускорительном центре FAIR эксперимент CBM. Все эти эксперименты были либо недавно запущены, либо в той или иной степени близки к запуску.

Кроме того, все эксперименты на LHC интенсивно модернизируются в связи с увеличением светимости ускорителя.

Основные функции **геометрической БД** – это хранение, обработка и управление данными об идеализированной геометрической модели экспериментальной установки (ЭУ). Точные геометрические поправки, в частности, для юстировки детектора, не хранятся в геометрической БД, а содержатся в БД состояний как параметры калибровки.

Для упрощения работы с существующими БД для эксперимента CMS был разработан специализированный комплекс интерфейсов, который стал также использоваться в двух других экспериментах ATLAS и LHCb. Данный комплекс ПО предоставляет функциональные возможности для доступа к данным в реляционных БД с использованием общего API без языков C++ и SQL, позволяет сохранять состояния ЭУ в момент сбора событий, сохраняет объекты C++ и их коллекции в гибридном хранилище, объединяющем реляционные технологии хранения с технологиями хранения потоковых данных.

Эксперименты ATLAS и CMS имеют схожие подходы для геометрической БД, которые базируются на обращении к реляционной геометрической БД при помощи вышеуказанного ПО. Поддержка этого ПО является трудной задачей, а потому данные решения сложно использовать в других экспериментах ФВЭ.

В эксперименте ALICE на основе интеграции с пакетом ROOT создали собственный пакет AliRoot, включающий в себя классы, используемые для моделирования, реконструкции и анализа данных. В качестве системы хранения геометрии для AliRoot была разработана система ROOT/TGeo. Именно подход с тесной интеграцией с ROOT, по которому пошла ALICE, был взят за основу в ряде более поздних экспериментов, включая эксперименты мега-проектов FAIR и НИКА.

Данные **конфигурационной БД** - это данные, содержащие различную информацию о конфигурации ЭУ, а именно: структура ЭУ, настройки напряжения источников питания, программируемые параметры электроники и другие, необходимые для приведения различных систем ЭУ в рабочее состояние.

В эксперименте CMS разработана система OMDS (Online Master Database System), которая отвечает за хранение конфигурации ЭУ и данных, не относящихся к физическим событиям.

БД конфигурации эксперимента ALICE описывает электронику и некоторые настройки электроники для конкретной ЭУ, параметры системы управления экспериментом и другие параметры.

В эксперименте ATLAS конфигурационная БД описывает, хранит и обеспечивает доступ к информации не только о том, какие элементы установки

должны быть задействованы, но также о том где, когда и какие процессы должны быть запущены, каковы условия для их выполнения и завершения. Система работает в качестве отдельной компоненты рамках общей on-line системы ATLAS на основе использования объектной БД и показала свою высокую эффективность, но ее использование в других экспериментах достаточно проблематично в силу большой избыточности всей on-line системы для других не столь больших экспериментов.

БД состояний хранит данные, описывающие состояние любой ЭУ или подсистемы.

В эксперименте CMS есть 2 базы данных, отвечающих за хранение данных состояний. Это располагающаяся на стороне ЭУ БД с данными состояний, включая константы калибровки и юстировки, используется в on-line обработке. Другая БД расположена в вычислительном центре CERN и содержит копию предыдущей за все периоды времени.

В эксперименте ATLAS БД состояний имеет название COOL. Она хранит состояния и условия работы детекторных, триггерных, ускорительных систем и метаданные, включающие константы калибровки, юстировки и выравнивания, настройки триггеров, конфигурацию детекторов и настройку самой системы управления, содержит информацию мониторинга, гистограммирования, параметров запуска и другие элементы. В настоящее время разрабатывается новая система CREST с упрощенной структурой, одной из целей которой является облегчение сопровождения системы.

В эксперименте ALICE данные хранятся в ROOT объектах, размещенных как часть AliEn/Grid каталогов. Доступ к данным осуществляется с помощью программного пакета AMANDA. В настоящее время разрабатывается уже AMANDA 3, позволяющая осуществлять одновременный доступ с использованием нескольких клиентов.

Эксперимент BM@N использует для хранения данных состояний реляционную БД PostgreSQL. Использование данной БД обеспечивает корректный многопользовательский доступ (unidb) к фактической информации эксперимента для обработки данных.

Система метаданных физических событий обеспечивает управление информацией об уникальном номере события, сохраняет ссылки на происходящие в эксперименте события, сработавшие при on-line обработке триггеры, перечень восстановленных частиц и другую информацию, необходимую для удобного поиска определенных типов событий, требуемых для физического анализа. Данная ИС также отвечает за создание, поддержку и проверку качества каталога физических событий, а также проводит индексирование с целью обеспечения

быстрого поиска требуемого набора физических событий по необходимым признакам и параметрам, используемым в различных физических анализах, для их дальнейшей загрузки и обработки.

В эксперименте CMC за хранение метаданных отвечает система PhEDEx. основанная на высокоскоростном кластере БД Oracle.

В эксперименте ATLAS за хранение и использование метаданных отвечает система EventIndex. Это первое приложение, которое было полностью разработано с учетом использования NoSQL баз данных в качестве серверной части вместо традиционной реляционной БД.

В эксперименте ALICE система хранения метаданных реализована на основе каталога файлов. Это была первая система, реализованная в эксперименте. Распределенная вычислительная среда появилась позже как естественное продолжение парадигмы «файловой системы». В отличие от реальных файловых систем, она не имеет владельца файла. Она хранит только связь между логическим именем файла (LFN) и физическим именем файла в реальной файловой системе.

БД временных рядов используется для задач мониторинга в больших компьютерных системах, таких, как система сбора данных детекторов ФВЭ. ПО для решения таких задач создается с привлечением огромного количества различных технологий, а сам процесс разработки такого ПО достаточно трудоемкий. При этом реализация функций мониторинга сильно зависит от специфики системы. Поэтому при анализе созданных в разных экспериментах систем мониторинга наибольший интерес вызывают не конкретные реализации этих систем, которые без значительных изменений невозможно применить в других экспериментах, а общий инструментарий, использованный при их создании.

В первой главе дается описание проблем, возникающих при разработке ИС и геометрических БД для экспериментов ФВЭ. Приведены положения разработанной методики для создания геометрических ИС. В качестве реализации этой методики приведены объектная модель и архитектура системы для экспериментов CBM и BM@N.

Предназначенная для обработки модельных и экспериментальных данных программная среда экспериментов CBM, а также ряда экспериментов мега-проекта NICA, основана на ПО FairRoot. Одним из преимуществ этой среды является возможность самостоятельной работы с отдельными частями математической модели ЭУ, называемыми геометрическими модулями или модулями. FairRoot представляет собой объектно-ориентированный пакет для моделирования, реконструкции и анализа данных, разработанный для экспериментов на FAIR. В FairRoot геометрия всей ЭУ рассматривается как

совокупность детекторов, входящих в нее, а каждый детектор - как отдельный геометрический модуль, который содержится в файле формата ROOT.

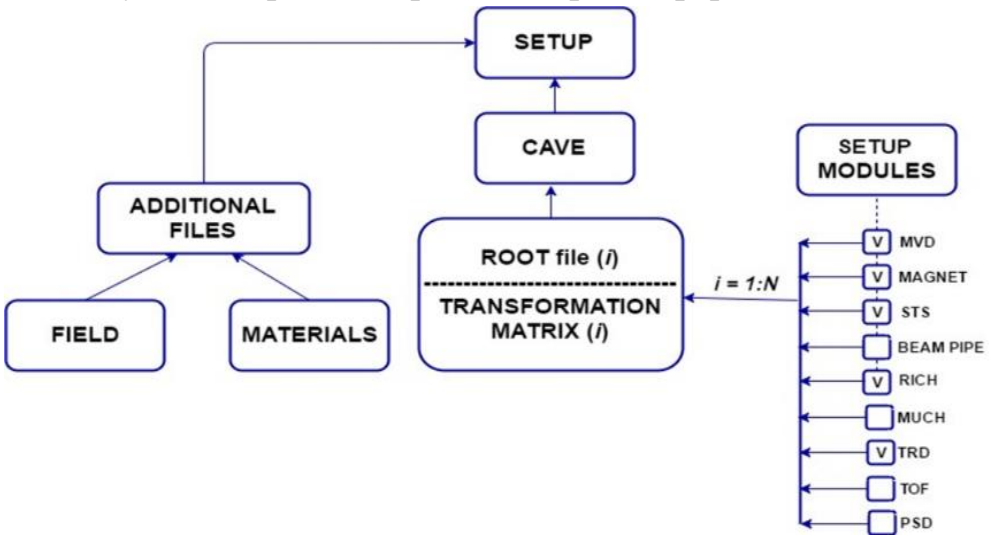

Рисунок 1: Структура Setup

Система поддержки геометрии ЭУ в эксперименте CBM в тот момент, когда диссертант приступил к исследованию данной проблемы, имела ряд недостатков. Например, файлы геометрии хранились в репозитории без метаданных и распространялись через их версии в репозитории, существовало большое разнообразие версий геометрии, многие устарели. Простое хранение файлов геометрии в репозитории неудобно в силу подверженности ошибкам, так как файловая система не следит за целостностью набора геометрии. Поэтому, распространение геометрии модулей через БД, где версия геометрии не зависит от внешних изменений, как это происходит с версиями файлов в репозитории, представлялось более оптимальным решением. Основным тезисом при разработке геометрической БД является тезис о том, что наименьшая единица гранулирования геометрии ЭУ, которая будет учитываться в самой БД, представляет собой модуль, представленный ROOT-файлом. Все содержимое геометрии модуля скрыто в этом ROOT-файле. Основываясь на этом постулате, все эксперименты, базирующиеся на FairRoot, имеют примерно одинаковую структуру геометрии, представленную на рис. 1. Как показано на рисунке, в состав ЭУ (SETUP) входят основные и дополнительные элементы. Все основные элементы геометрии вложены в элемент «пещера» (CAVE). Иногда некоторые модули могут располагаться внутри другого модуля, который в этом случае становится «родителем». Все модули геометрии имеют связь с родительским модулем, который хранит начало координат конкретного модуля.

Указанные особенности структуры геометрии и методов ее использования для экспериментов, использующих FairRoot, позволяют сформулировать общую методику к разработке геометрических БД.

- **Подготовка модулей геометрии.** Использование FairRoot предполагает, что все данные геометрии загружены в класс TGeoManager, работающий с геометрией в пакете ROOT. Для каждого элемента ЭУ требуется создать свой файл (модуль) с геометрией. В дальнейшем любые изменения внутри геометрии модуля будут порождать создание нового файла в геометрической БД. Данный подход позволит хранить историю. Для каждого из модулей должен быть создан c++ класс, наследующий базовый класс FairDetector, для ее обработки.
- **Подготовка данных магнитного поля и файла с материалами.** Так как файл с данными о магнитном поле обычно достаточно большой, то в БД хранится только ссылка на него, а сами файлы размещены на отдельном сервере. Другой важной частью геометрии является файл с материалами, используемыми в модулях. Обычно это файл в текстовом формате «geo», что позволяет легко просматривать существующие материалы при создании или модернизации модулей геометрии.
- **Создание распределенной системы хранения геометрии, обеспечивающей целостность данных.** Геометрия с течением времени постоянно меняется. При этом должна оставаться возможность загрузки как новых данных, так и более старых версий геометрии. Для решения данных проблем должна быть создана распределенная система хранения. На серверной части хранится полная версия всех геометрий. На клиентскую часть загружается только полная версия нужной геометрии.
- **Создание рабочей среды для геометрической БД.** Кроме создания самой БД требуется еще создать и удобную среду для работы с ней. Тут можно выделить 2 части. Первая часть для разработчиков: она обеспечивает удобную загрузку различных частей геометрии и сборку геометрии в единую ЭУ. Вторая часть для пользователей: она обеспечивает просмотр всех существующих официальных версий геометрии, установку нужной геометрии на локальный компьютер, загрузку выбранной геометрии в среду ROOT во время запуска программы и возможность внесения изменений в локальную копию геометрии путем отключения части модулей.
- **Сборка геометрии ЭУ.** На этом этапе происходит загрузка подготовленных данных в саму БД. Для геометрической БД заносить сдвиги позиции внутрь модуля необязательно, они прописываются в саму БД. Это дает возможность сдвигать модули без создания новых версий самого модуля. Также необходимо

заносить порядок загрузки модулей в системное окружение и отмечать модуль, относительно которого задаются координаты загружаемого модуля. Это актуально в случае, если один элемент ЭУ вставляется в другой.

- **Верификация**. Так как такая БД отвечает не только за хранение данных, но, в случае моделирования, и за сборку установки, то одной из задач является проверка алгоритмов сборки ЭУ. Проверка включает в себя 2 этапа. На первом этапе после моделирования сохраняется файл с полной геометрией. После чего данный файл визуализируется и визуально проверяется ситуация с ЭУ. На втором этапе происходит сравнение распределений, полученных после реконструкции. Сравниваются версии с использованием БД и с ручной загрузкой геометрии.
- **Конфигурирование и развертывание системы**. Последний этап при реализации предлагаемой методологии и первый этап в ее использовании. В описанных условиях система может быть адаптирована к любым экспериментам, использующим FairRoot. Задача решается созданием файла конфигурации с такими параметрами, как имя БД, пути хранения сопутствующих данных и т.д. Скрипт развертывания системы инициализирует систему с тем, чтобы ИС была сразу готова к использованию.

Согласно сформулированной методике была разработана и реализована геометрическая ИС для эксперимента CBM, которая позже была использована в качестве базовой и развита в рамках эксперимента BM@N.

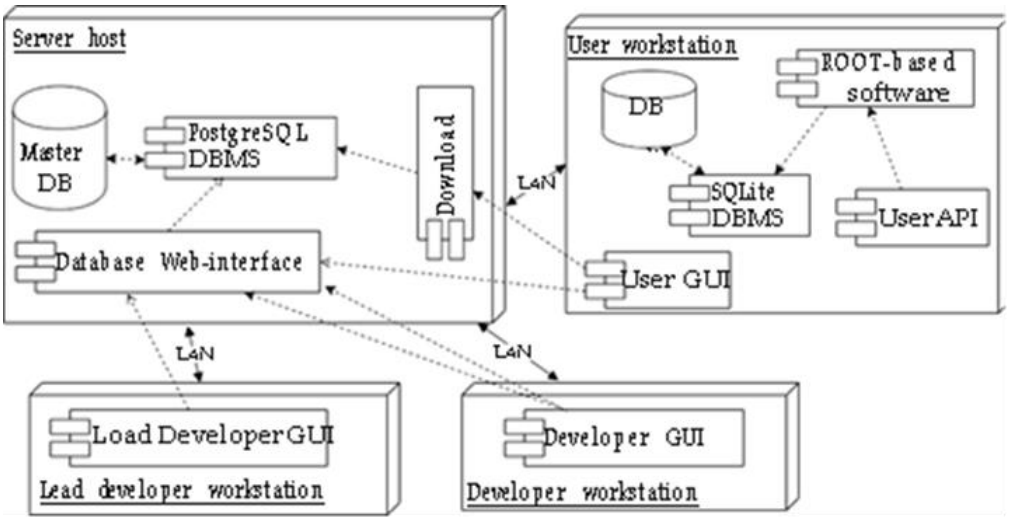

Рисунок 2: Общая архитектура геометрической ИС

Общая архитектура геометрической ИС, показанная на рис. 2, использует центральную и локальную БД геометрии. Центральная БД геометрии, реализованная на хост-сервере под сервером Apache с помощью PHP-скриптов, обеспечивает все функциональные возможности для манипулирования

геометрией. Локальная БД геометрии используется для загрузки геометрии ЭУ в задачах моделирования и реконструкции, реализована на базе системы управления базами данных (СУБД) SQLite как локальная реплика центральной БД PostgreSQL, но содержит в себе только проверенные и протестированные версии ЭУ. Благодаря небольшому размеру базы может быть создано любое их количество. Таким образом, масштабирование не приводит к увеличению времени загрузки локальной БД в рабочую среду Root.

Полная геометрия ЭУ строится и хранится в виде комбинации ссылок на составляющие ее элементы, включая матрицу преобразования и ссылку на родительский модуль, ссылки на файл с материалами и ссылки на описание магнитного поля. В диссертации приведена и описана соответствующая объектная модель данных.

Одной из проблем загрузки геометрии в программы симуляции является то, что для обработки геометрии каждого элемента ЭУ требуется написать собственный C++ класс и использовать при загрузке определенную версию этого класса. В ходе разработки унифицированной геометрической ИС был реализован подход, позволяющий не вносить изменения в код загрузки геометрии. Это стало возможным потому, что все соответствующие C++ классы и методы, в конечном счете связанные с загрузкой геометрии элемента ЭУ, обязаны реализовывать один и тот же интерфейс.

В рамках ИС для геометрической БД разработаны интерфейс прикладной программы (API) и веб-интерфейс; их можно использовать во всех экспериментах NICA посредством настраиваемой установки.

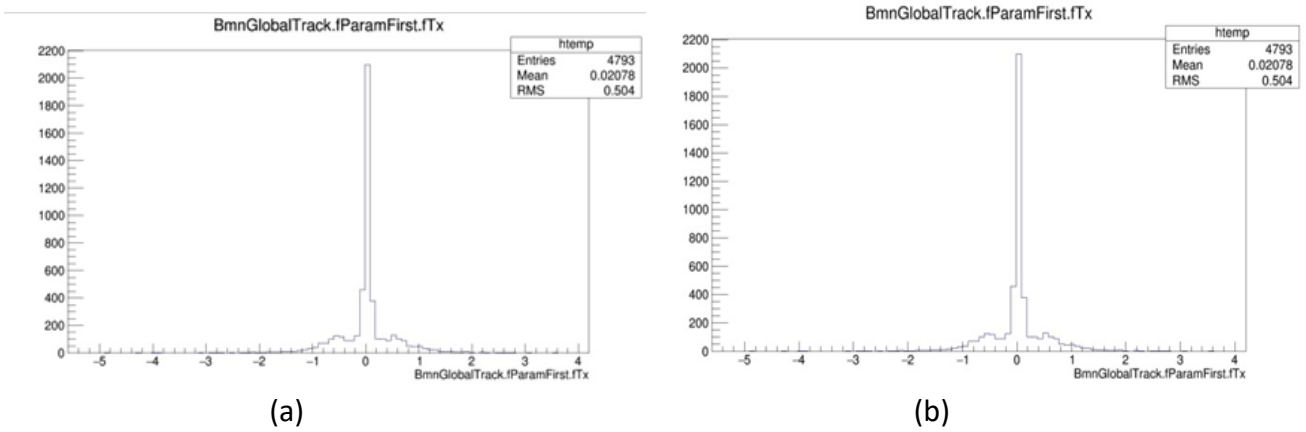

Рисунок 3: Пример гистограмм распределения восстановленных импульсов трека экспериментальных данных, полученных в ходе RUN6 (a) с использованием исходной геометрии; (b) с использованием геометрии из БД

API реализован как в виде набора макросов ROOT, так и в виде отдельного C++ класса. Пользователь может запускать следующие скрипты: просмотр списка версий детектора в локальной БД; создание локальной реплики БД; загрузка геометрии ЭУ для задач реконструкции или моделирования. При моделировании пользователь может использовать файл xml для сдвига или отключения любого модуля установки во время загрузки.

Класс реконструкции загружает только полную геометрию ЭУ. Класс для моделирования загружает геометрию ЭУ, как комбинацию модулей. После загрузки геометрия может быть сохранена в файл как полная геометрия ЭУ. Этот файл можно использовать для реконструкции.

Одной из основных идей тестирования БД Geometry является сравнение результатов работы задач моделирования и реконструкции. Результаты работы при одних и тех же входных параметрах должны быть одинаковыми для двух случаев: с БД геометрии и без нее. На рис. 3 представлено распределение проекций глобальных трековых импульсов вдоль оси X в реконструированных событиях эксперимента (a) с использованием исходной геометрии (b) с использованием геометрии из БД. Значения показателей Entries (количество событий), mean (среднее) и RMS (среднее квадратичное) одинаковы для обоих случаев, что говорит об идентичности полной геометрии, загруженной с помощью БД и без нее.

Во второй главе дается описание проблем, возникающих при разработке гибких сервисов автоматизированного поиска событий в экспериментах ФВЭ, когда количество анализируемых данных велико. Дано описание контекста, в котором проводились работы. Приведены положения предложенной методики. Представлен введенный в эксплуатацию гибкий сервис EventPickingService для эксперимента ATLAS, реализующий данную методику.

Системы поиска физических событий по метаданным являются важной компонентой ПО для многих современных экспериментов ФВЭ. В результате поиска пользователь обычно получает множество уникальных идентификаторов (GUID) искомых событий. Найденные события могут хранится в разных файлах, файлы могут находиться в разных вычислительных центрах, разбросанных по всему миру. Нахождение GUID является для физика только первым этапом поиска и сбора нужных ему событий. Для получения информации о месте хранения данных по GUID разработаны специальные системы, которые включают в себя и функции сбора требуемых данных в одном месте, извлечение события и размещение его в заданном файле. В ATLAS эти функции выполняют система управления научными данными Rucio и система анализа распределенных данных PanDa. Системы являются закрытыми, то есть требующими аутентификации

пользователя, и достаточно сложными в использовании. В ATLAS для некоторых процессов физического анализа потребовалось извлечение 50 000 событий в формате RAW из 18 миллиардов событий во втором сеансе (около 10 миллионов файлов). Это потребовало порядка 3 месяцев работы.

Так как следующий раунд этого анализа требовал обработки 136 тыс. событий, то решено было автоматизировать данный процесс. Работа по созданию такой автоматизированной системы стала для автора частью работы над диссертацией. Данная задача в том или ином виде стоит и для других экспериментов ФВЭ. Необходимо учитывать, что даже в рамках одного эксперимента ATLAS процесс извлечения событий для разных форматов существенно отличается. Для других экспериментов реализация самих шагов, требуемых для процесса извлечения события, будет совершенно другой. Поэтому сервис должен быть гибким, позволяющим динамически настраивать его функционал в зависимости от требуемых шагов. В силу того, что некоторые запросы требуют большого времени их выполнения, должна учитываться возможность распараллеливания работы сервиса. Необходимо также максимально обезопасить доступ к закрытым системам, используемым на разных этапах работы сервиса.

Все это легло в основу разработки архитектуры сервиса и методики создания гибких сервисов автоматизированного поиска событий.

Обычная архитектура сервиса могла бы включать традиционную архитектуру «клиент-сервер» с хранением и использованием необходимых данных в связанной с сервером БД. Однако система имеет ярко выраженную особенность. С одной стороны, она должна быть общедоступна для пользователей, размещающей в клиентской части свои запросы. С другой стороны, выполнение этих запросов связано с внешними системами, доступ к которым требуют более высокой степени защиты. Кроме того, внешние запросы часто требуют большого времени для их выполнения, достигающего иногда многих дней. Поэтому логичным предложением является добавление в архитектуру еще одной компоненты - демона, отвечающего за ту часть обработки запросов, которая связана с реализацией этих запросов через выполнение команд других систем. Таким образом, система должна состоять из трех основных компонент: веб-сервера, БД и демона. Методика создания сервиса должна описывать правила взаимодействия как этих трех компонент, так и принципы создания демона, как той компоненты, которая выполняет наиболее сложные операции. Основополагающим фактом является то, что все детали работы демона заносятся в БД.

Когда мы говорим о гибкости, одним из основных подходов является разработка приложения с возможностью подключения плагинов. Описание того, какие плагины и в каком порядке должны быть подключены, какие параметры при старте плагина нужно использовать, также можно хранить в БД, например, в формате XML или JSON, чем достигается максимальная гибкость процесса. Сами плагины, или точнее их типы, могут структурироваться в том числе так, чтобы было ясно какие из плагинов могут работать параллельно. Сам процесс выполнения плагинов должен соответствовать рабочему процессу (workflow), который также прописывается в БД в виде XML или JSON. Для выполняющихся последовательно в цепочке workflow плагинов должны быть согласованы их входные и выходные данные.

Указанные особенности структуры гибкого сервиса автоматизированного поиска и сбора событий в экспериментах ФВЭ, позволяют сформулировать общую методику разработки сервиса, включающую

- Создание клиент-серверной системы. Именно на этом этапе надо определиться, как будут взаимодействовать клиент с сервером.
- Разбиение процесса поиска и извлечения события на отдельные задачи.
- Разработка стандартизированного входного и выходного формата данных для автономных модулей.
- Реализация данных задач в виде автономных модулей. После того как заданы входные и выходные потоки, можно реализовать все автономные задачи.
- Создание рабочего процесса из автономных модулей для различных типов событий.
- Тестирование сервиса.
- Создание системы мониторинга сервиса.

В соответствии с разработанной методикой был создан и внедрен в эксплуатацию соответствующий сервис автоматизированного поиска и сбора событий для эксперимента ATLAS.

Сервис поиска и сбора событий может работать с событиями разного типа, такими как сырые данные (raw data), данные объекта анализа (AOD) и другими. Как показывает практика, даже во время выполнения одного запроса возникает необходимость внесения изменений в рабочий процесс. В результате важно, чтобы сервис был модульным, а рабочий процесс мог динамически изменяться в процессе эксплуатации сервиса.

После анализа типовых запросов были выделены два типа задач. К первому типу относятся задачи, использующие полные входные данные. Задания этого типа необходимо выполнять последовательно. Второй тип - это те, которые используют только часть ввода. Эти задачи могут выполняться параллельно.

Обычно параллельны целые цепочки последовательных задач, а не отдельные задачи. В этой цепочке задача использует все данные из предыдущей задачи, но первая задача в этой цепочке использует только часть данных из родительской задачи.

На рисунке 5 показан рабочий процесс сбора событий для анализа процесса рождения пар калибровочных бозонов W при взаимодействии двух фотонов («γγ → WW»). Веб-сервер берет данные от пользователя и на их основе создает начальные данные для первого задания и задает номер первого задания. Первое задание сортирует входные данные и разбивает их на подзадачи по номеру прогона (run). Второе задание запускает все данные разбиения в параллельном режиме. Для каждого прогона выполняются следующие задачи: получить GUID из прогонов и номера события, получить шаблон набора данных с помощью GUID, начать копирование события с помощью Panda, проверить выходные файлы, чтобы убедиться, что все события были скопированы, занести метаданные в систему Rucio. После того, как демон завершит рабочий процесс, он помечает его как завершенный. Веб-сервер наблюдает за всеми созданными запросами и если обнаруживает, что какой-то запрос закончен, то посылает результаты по нему пользователю.

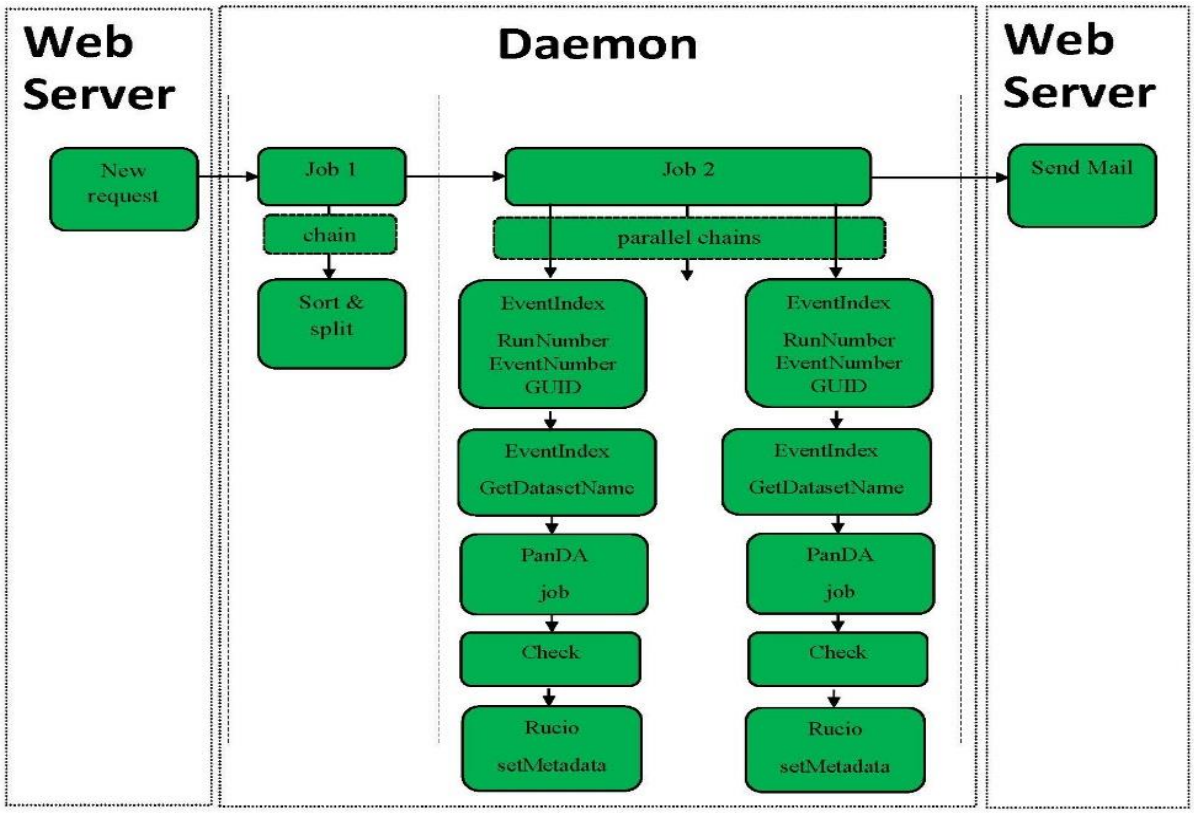

Рисунок 5: Workflow выбора событий для анализа «γγ→WW»

Как упоминалось ранее, служба выбора событий состоит из двух основных компонентов: веб-сервера и демона. Взаимодействие между этими компонентами

осуществляется с помощью БД PostgreSQL. Кроме того, в БД хранятся входные и выходные данные, состояния для всех запросов, заданий, цепочек или задач, журналы и весь рабочий процесс. В диссертации приведена структура БД службы выбора событий. Все основные элементы рабочего процесса имеют свои таблицы в БД для хранения текущего состояния, связь для ввода/вывода данных, статус и так далее. Входные и выходные данные хранятся в отдельной таблице в формате JSON, а элементы имеют только ссылку. Каждая реальная задача может сохранять в БД свои лог файлы. Задачи и цепочки порождают рабочие процессы. Рабочий процесс для задачи содержит в себе уникальный номер, соответствующей входному статусу для данной задачи, выходное состояние и полное имя JAVA-класса, реализующего его. Рабочий процесс для цепочки содержит в себе уникальный номер, соответствующей входному статусу для данной задачи, имя метода, реализующего конкретную задачу, выходное состояние, количество возможных перезапусков задачи в случае неудачи и время между перезапусками. Кроме этого, для балансирования нагрузки на сторонние сервисы процесс хранит максимальное значение одновременно выполняемых задач данного типа.

Сервис использовался для второго этапа анализа «γγ→WW» (136 тыс. событий). Использование сервиса позволило существенно сократить время сбора событий.

В третьей главе приведено описание особенностей задачи создания сервиса управления on-line приложениями, запускаемыми во время сбора экспериментальных данных. Приведены положения методики разработки таких сервисов в рамках ИС конфигурации детекторов. Представлены архитектура и модель БД, разработанных в соответствии с приведенными правилами для эксперимента BM@N. Описана структура и функции менеджера конфигурационной системы.

Архитектурное решение с веб-сервером, демоном и БД, как связующим звеном между ними, может быть использовано не только в сервисах поиска и сбора событий. В третьей главе добавлены положения методики для задач управления on-line процессами ФВЭ как развитие методики, описанной в предыдущей главе для другого класса задач. Кроме обычных параметров конфигурации детекторов предлагается заносить в БД информацию о приложениях, которые должны стартовать во время активации ЭУ при старте набора данных. Сервис должен запустить соответствующие приложения с нужными параметрами на заданных компьютерах в нужном порядке и отслеживать статус запущенных приложений, останавливать их по окончании набора данных, сообщать об ошибках и так далее. Задача является типичной для многих экспериментов, однако, как показал анализ существующих систем,

использование этих систем, реализованных в других экспериментах ФВЭ, приводит к избыточным требованиям по постановке и поддержке дополнительного ПО. Поэтому, для того чтобы избежать разработки трудоемкой части по созданию полной системы запуска и управления приложениями на разных компьютерах или кластерах компьютеров, можно создать свой менеджер управления задачами, использующийся для старта и управления приложениями API сервисов, созданных в других экспериментах.

Основным отличием данного типа задач от описанных в предыдущей главе особенностей системы поиска и сбора физических событий являются отсутствие дополнительных требований к безопасности при использовании внешних API, но при этом более строгие требования по времени отклика. Таким образом, добавляя к типовым веб-серверу и БД еще и менеджер конфигураций, мы не запрещаем возможность обмена командами и сообщениями между сервером и менеджером напрямую, без записи команд в БД, что уменьшает время отклика.

Структурно система может быть разбита на четыре главных целевых сегмента. Первый - это ввод и поддержка перманентных данных, то есть данных, которые описывают необходимые параметры приложения или ЭУ и заносятся в БД человеком. Эта функция обеспечивается взаимодействием между веб-сервером и специализированной БД. Остальные три целевых сегмента связаны с взаимодействием менеджера конфигураций с остальными компонентами. Во-первых, это получение менеджером конфигураций команд для выполнения. Данная функция реализуется путем обмена сообщениями между сервером и менеджером. Во-вторых, это чтение менеджером из БД информации, необходимой для выполнения команд, и запись в БД результатов их выполнения. Веб-сервер и, соответственно, пользователь могут получить эти динамические сведения из БД без участия менеджера. И, наконец, собственно выполнение команд производится менеджером с использованием соответствующих API. Данные, связанные с специализированной БД, представляются в виде, удобном для их сопровождения. Внешние API в общем случае используют структуру данных, которая может сильно отличаться от используемых в разрабатываемой специализированной БД, что требует также решения задачи on-line конвертации с использованием конвертированных данных в API.

Таким образом, последовательность действий такова:

- Выбор сервиса для запуска приложений (агента + соответствующий контроллер)
- Создание, менеджера, который осуществляет связь с веб-сервером, с БД и осуществляет запросы во внешний сервис, выполняя, таким образом, следующие функции:

- Слушает входящие запросы на управление on-line приложениями;
- Осуществляет чтение из БД, требуемых для выполнения запроса;
- Конвертирует данные к требуемому виду для обращения к API внешнего сервиса;
- С помощью вызова соответствующих API методов внешнего сервиса выполняет запрос;
- Передает на веб-сервер ответ на запрос;
- Записывает в БД результаты запроса.

➢ Создание веб-сервера и БД в удобном для пользователей виде;

➢ Настройка конкретных компьютеров для запуска приложений с использованием сервиса.

Данный подход был применен при проектировании и реализации конфигурационной БД и соответствующей ИС эксперимента BM@N.

Разработанная общая архитектура ИС представлена на рисунке 6. Ядром системы является БД конфигураций на базе PostgreSQL. В БД хранятся все данные конфигурации, как аппаратного, так и программного обеспечения, которые необходимы для on-line обработки событий. Клиент, реализованный в виде графического веб-интерфейса, который содержит веб-страницы, доступные для просмотра в любом браузере, взаимодействует с ConfigDb через веб-сервер Apache.

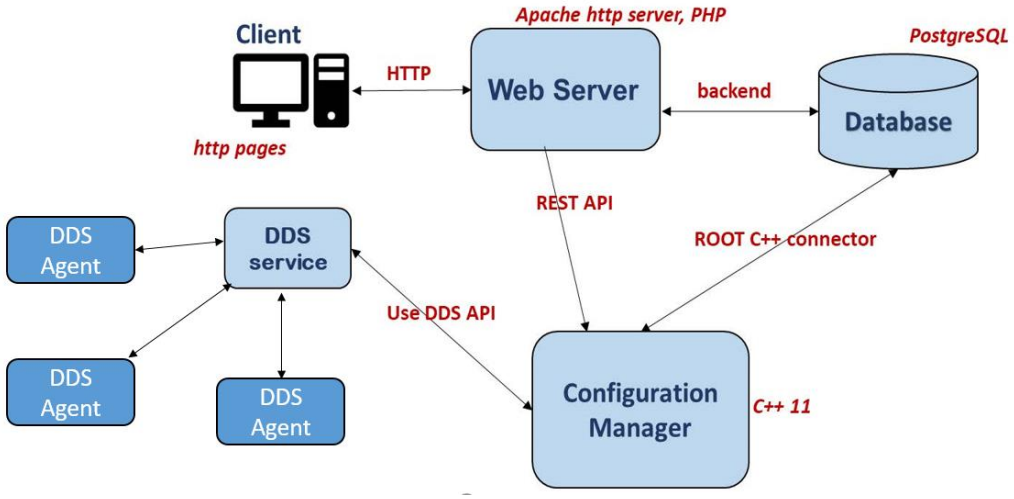

Рис. 6: Общая архитектура ИС конфигураций

Веб-сервер позволяет отправлять команды диспетчеру конфигураций (Configuration Manager), используя технологию REST API. Диспетчер конфигураций запускает нужные задачи с требуемыми параметрами на заданных компьютерах в установленном порядке, осуществляет управление приложениями, включая их остановку в назначенное время. Он работает как процесс-демон и выполняет полученные команды, взаимодействуя с сервером DDS через API DDS.

Кроме того, диспетчер конфигураций считывает информацию о конфигурации из БД и записывает необходимую информацию о запущенных, аварийно завершенных или остановленных задачах, используя среду ROOT C++.

Лог DDS сессии отображается в БД и обновляется по мере увеличения. Лог-и задач записываются в БД после завершения задачи. Текущее содержимое лог-а незавершенных задач можно получить по специальной команде, что используется в соответствующем веб-приложении. Кроме того, диспетчер конфигураций получает информацию о состоянии текущих задач через обратные вызовы API DDS для обновления динамической информации в БД конфигураций. В диссертации приведено полное описание структуры БД.

В четвертой главе дается краткое описание вычислительной и сетевой инфраструктуры эксперимента ATLAS. Представлена первая и модернизированная версия системы ее мониторинга, описаны проблемы, которые возникли в ходе модернизации, дана методика визуализации элемента «дерева» в Grafana и была представлена ее реализация на примере мониторинга для сетевой инфраструктуры эксперимента ATLAS.

Система мониторинга эксперимента ATLAS чрезвычайно важна. Вычислительная и сетевая инфраструктура эксперимента ATLAS состоит из 4020 вычислительных узлов, 285 сетевых коммутаторов и 14 778 коммутаторных интерфейсов (портов). Для мониторинга этой сети была разработана система NetIs. Первая ее версия была разработана в 2010 году на основе БД циклических временных рядов (RRD). Со временем возникла необходимость заменить NetIs более качественным сервисом, так как усилия по поддержке NetIs значительно увеличились вместе с размером и сложностью сетевой системы. Кроме того, RRD, используемая для хранения данных, со временем приводила к потере детализации, что делало инструмент непригодным для извлечения точных значений из прошлого. Еще одно неудобство использования первой версии заключалось в том, что графики, отображаемые NetIs, генерируются сервером, а потому достаточно статичны. В силу вышеуказанных причин, было решено модернизировать ту часть системы, которая относится к реализации панели мониторинга сети, и заменить соответствующую графическую панель на более современную и быструю панель с использованием платформы Grafana. Графический интерфейс NetIs содержит в себе динамическое дерево для выбора узлов, переключателей или интерфейсов и информационных панелей. Использование этого инструмента очень удобно, однако элемент «дерево» не поддерживается в Grafana. Таким образом, в рамках модернизации системы возникла задача разработки метода интеграции динамического дерева в платформе Grafana.

Создание новой панели — это основной способ добавления новых функций в Grafana. Этот путь требует наличия или приобретения навыков для улучшения самой платформы. Плагин должен быть скомпилирован как часть рабочего процесса разработчика Grafana и установлен на стороне сервера. Внутри этого пути, чтобы добавить какую-либо новую функцию, например, в нашем случае нам нужно установить позицию для выпадающего меню, требуется новый плагин. К сожалению, этот путь не гарантирует совместимость с новыми версиями платформы и, таким образом, требует постоянных модификаций плагина под новые версии Grafana. Таким образом, самый эффективный способ добавить дерево в Grafana — это добавить код JavaScript на текстовую панель. Дерево не является элементом самой Grafana и не может использовать обычные методы извлечения данных. Оно должно напрямую использовать объекты JavaScript библиотеки Grafana для взаимодействия с другими элементами. Этот метод можно применить для любого объекта HTML/JavaScript, включая выпадающее меню.

Новая версия NetIs использует код JavaScript в текстовой панели для реализации дерева (рис. 7) и некоторых других элементов. Текстовая панель является основным элементом Grafana. Она поддерживает код HTML и JavaScript внутри него. Это внешнее дерево JavaScript взаимодействует с кодом JavaScript Grafana, загруженным браузером, который включен в основную HTML-страницу. С помощью этой библиотеки пользователь может получать данные с других панелей и обновлять панель приборов.

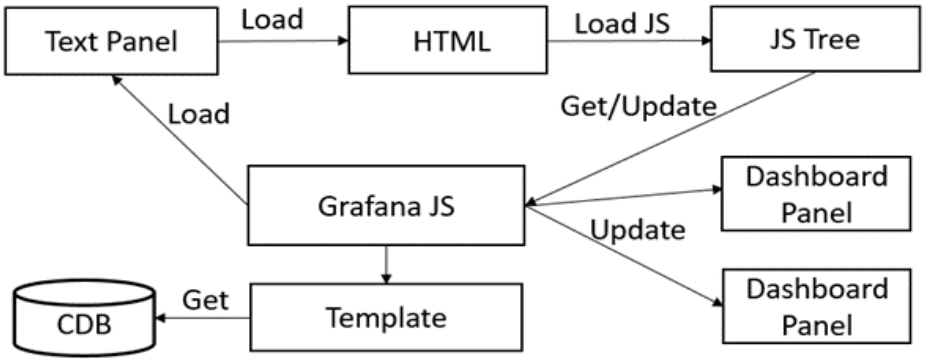

Рисунок 7: Схема взаимодействия внешнего дерева JavaScript с Grafana.

Исходные данные для построения дерева NetIs хранятся в БД CDB MySQL. У Grafana есть плагин для MySQL, но, как уже упоминалось, дерево не является элементом платформы, поэтому оно не может напрямую взаимодействовать с ее источником данных. Вместо этого дерево получает данные с помощью элемента шаблона Grafana. Шаблон хранит данные в виде списка специально структурированных строк, поэтому эти строки первым делом необходимо сформировать. Строки имеют следующую структуру: Function:Device::Linecard::Interface. Одна из проблем состоит в том, что данную

структуру не всегда можно получить из таблиц, так как у некоторых устройств отсутствует элемент linecard. В этих случаях при преобразовании, для использования вместо linecard, генерируется один из двух специальных элементов: PORTS или LAG. Элемент PORTS генерируется для реальных устройств, а элемент LAG для виртуальных.

В пятой главе дается описание БД условий (COOL) эксперимента ATLAS и тех проблем, что возникли в ходе ее эксплуатации. Приводится архитектура и модель данных проекта CREST, предназначенного заменить COOL в Run4. Даны существенные различия между моделями данных COOL и CREST. Приведены разработанные алгоритмы для конвертации данных и результаты тестирования на больших объемах данных.

Данные об условиях состоят из изменяющихся во времени величин, таких как данные калибровки и настройки установки, сила тока, напряжение, температура, информация о конфигурации запуска и сбора данных, информация о пучке ускорителя, конфигурация триггера, данные о состоянии установки и т.д.

Во время Run1 и Run2 данные ATLAS Conditions управлялись системой COOL/CORAL (CERN-IT). COOL — это C++ API на основе клиента CORAL для доступа к реляционной (Oracle) БД. Эта реализация БД на основе COOL имеет высокоуровневые функциональные возможности. Все подсистемы установки имеют свои собственные «базы данных COOL» (схемы) и хранят данные полезной нагрузки в специальных таблицах (папках). Данные полезной нагрузки хранятся в соответствии с интервалом достоверности (IOV). Каждый IOV определяется временем «с момента» и «временем до». COOL имеет два типа папок. Первый — это папки с одной версией: здесь можно добавлять IOV (без перезаписи, без определения метки папки). Второй — это папки, содержащие более одной версии: IOV могут храниться произвольным образом и группироваться по меткам, что позволяет перезаписывать полезную нагрузку. Глобальная метка — это набор нескольких меток папок. Полезные нагрузки извлекаются путем предоставления границ IOV и соответствующей метки.

Проект новой БД условий получил название CREST (Conditions Database REST API) и был начат по нескольким причинам:

- Кэширование БД COOL плохо оптимизировано, поскольку запросы, определенные с использованием разных границ IOV, могут возвращать одни и те же данные полезной нагрузки;
- Структура БД COOL сложная: данные условий разбросаны по 30 схемам и 10 тыс. папок (около 1 ТБ за период сбора данных);

- Долгосрочное обслуживание и развитие COOL проблематичны: COOL API (как и CORAL) не будет поддерживаться ИТ-подразделением CERN после запуска Run4;
- COOL не имеет встроенной поддержки управления глобальными метками.

Проект CREST для прототипа новой БД условий нацелен на улучшение доступа к данным, управления метаданными и, в частности, улучшение управления глобальными метками.

Проект CREST состоит из нескольких компонентов: сервера CREST, клиентской библиотеки C++, клиента командной строки CREST и конвертера COOL в CREST. Сервер CREST DB предоставляет функциональные возможности через REST. SQL не участвует в диалоге между клиентом и сервером, вместо этого внутренние ресурсы доступны через URL-адреса. Реализация основана на стандартных технологиях Java (JEE, Spring) и спецификациях (JAX-RS, JPA), как показано на рис. 8.

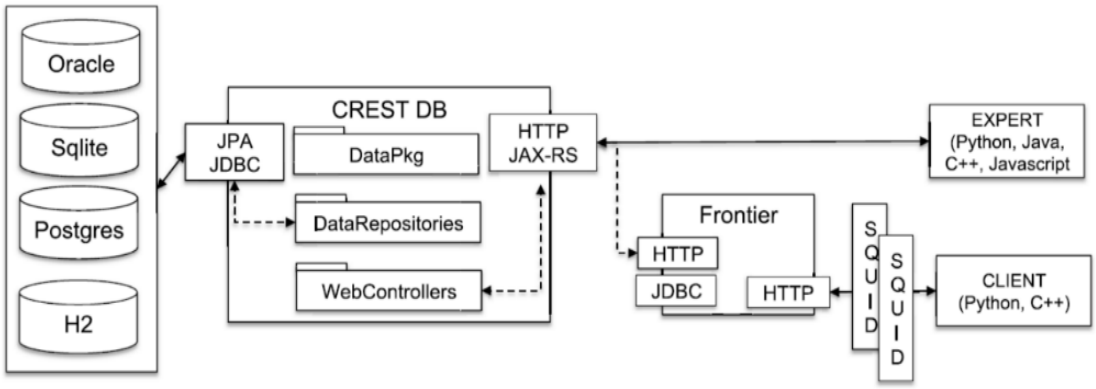

Рисунок 8: Архитектура CREST DB

Модель данных в CREST состоит из метаданных и данных полезной нагрузки, сгруппированных в пять таблиц, содержащих метаданные и данные полезной нагрузки. Первоначально подобный подход был осуществлен в эксперименте CMS для отдельного детектора. Данные условий хранятся в таблице PAYLOAD. Значения используются в виде агрегированного набора (обычно это заголовок и некоторые контейнеры параметров). Метаданные условий организованы в три таблицы, плюс одна, используемая в основном для сопоставления между метками папок и глобальными метками.

IOV в CREST содержит информацию о времени, которая хранится в одном столбце времени (время может быть представлено в виде метки времени, номера запуска и т. д.) и по умолчанию действует до следующего ввода времени. IOV указывает на одну полезную нагрузку с помощью хеш-ключа. TAG в CREST — это метка, используемая для идентификации определенного набора IOV. Для

облегчения переноса существующих данных из COOL была создана дополнительная таблица для метаданных меток папок. GLOBAL TAG — это метка, используемая для идентификации согласованного набора TAG, участвующих в данном потоке данных (например, кампания по переработке, производство MC и т. д.). Один и тот же TAG может быть связан с многими GLOBAL TAG.

IOV извлекаются отдельно от PAYLOADS. Это другое поведение по сравнению с COOL, который загружает IOV и PAYLOAD одновременно. Это дает несколько преимуществ: во-первых, доступ к PAYLOAD можно кэшировать, чтобы он стал быстрее, но также можно очень быстро проверить IOV, поскольку они де-факто являются метаданными PAYLOAD.

Существенные различия между моделями данных COOL и CREST требуют осторожного обращения в процессе преобразования данных. Правильность алгоритма конвертирования данных чрезвычайно важна. Ниже приведены основные отличия COOL от CREST, влияющие на алгоритм конвертирования:

- Каждый COOL IOV имеет два параметра, определяющих временной интервал: время начала и окончания, CREST IOV имеют только время начала, временем окончания является время начала следующего по времени IOV.
- В папке COOL есть список каналов, и данные хранятся в каждом канале для всех IOV. В модели данных CREST каналы хранятся в большом двоичном объекте (blob), БД ничего не знает о контенте blob и, соответственно, нет привязки каналов к IOV.
- COOL не имеет собственного API для поддержки глобальных меток.

Для упрощения задачи чтения данных из COOL было решено использовать уже существующий интерфейс IDatabase пакета COOL. Реализация данного интерфейса для CREST позволяла использовать утилиту AtlCoolCopy, задав в качестве выходной базы CREST. AtlCoolCopy - это широко используемый в эксперименте ATLAS инструмент для массового копирования и конвертации данных из COOL БД в другие системы, такие как root, MySQL и другие.

Данные из ATLAS COOL считываются по каналам, а не по IOV. Таким образом, на входе мы получаем для каждого канала список интервалов с данными, действительными на этих интервалах. CREST требует данные о всех каналах в конкретной точке, т.е. для формирования данных CREST требуется сначала считать значения всех каналов с нужной временной точкой из ATLAS COOL.

Список всех каналов для папки можно получить заранее. Это позволяет разработать две версии алгоритма конвертирования данных в зависимости от условия: только 1 канал или более одного канала содержится в папке.

Алгоритм конвертирования данных, когда в папке хранятся значения только одного канала, следующий.

Сначала производится чтение метаданных из COOL с дальнейшей их загрузкой в CREST. Затем происходит последовательное чтение самих данных из COOL. После каждого чтения первая точка интервала с данными сохраняется в JSON, который используется для передачи и хранения PAYLOAD в CREST. Для второй точки сохраняются пустые значения. Это означает, что больше данных нет. В случае пересечения на следующей операции пустой PAYLOAD перезаписывается данными со следующего интервала. В конце каждой итерации проверяется объем данных в памяти. Если он превышает заданный (5 Гб), то данные передаются на CREST-сервер. Это необходимо делать, так как объем данных некоторых папок в COOL слишком большой и превышает максимальный размер оперативной памяти, допустимый для работы приложений на кластере в CERN. После того как все данные будут считаны с COOL, происходит передача еще не переданных данных на CREST сервер и завершение работы алгоритма.

В случае, когда количество каналов больше 1, нужно использовать усовершенствованный алгоритм, схема которого приведена на рисунке 9.

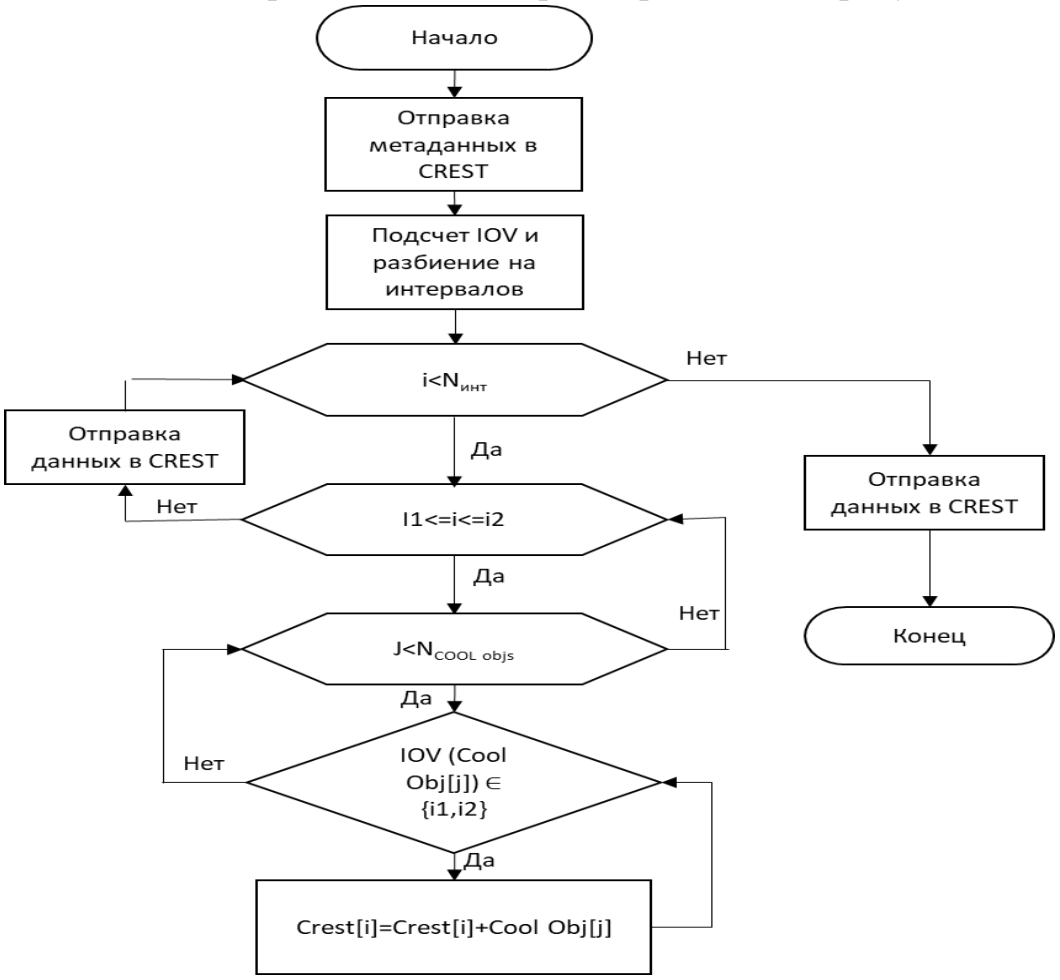

Рисунок 9: Случай с несколькими каналами

AtlCoolCopy построен на использовании COOL API, а, следовательно, он не умеет работать с глобальными метками. Передать данные о глобальной метке можно в момент передачи метаданных, но чтение информации о связи глобальных меток с помощью обычных средств COOL API совершить невозможно. В ATLAS Contitional за это отвечает отдельная БД, не связанная напрямую с COOL. Для конвертации данных, связанных с выбранным глобальными метками, используется Coolr Utility. Данная утилита является python-расширением COOL. Одной из ее возможностей является получение информации в формате JSON о всех папках, связанных с заданной глобальной меткой.

Проводилось 2 типа тестов: тестирование алгоритмов с использованием данных без глобальной метки; тестирование конвертации по глобальной метке.

В первом случае нужно было подготовить данные для QTests. QTests – это ежедневно проводимые тесты в ATLAS. В QTests используется лишь небольшой объем данных об условиях, но они охватывают большое количество систем. Список папок, которые использует QTests, был составлен с помощью сканирования лог-ов самого теста. В результате сканирования на входе конвертера получилось 115 различных папок из 13 различных подсистем ATLAS. Размер данных в отдельных папках превышал 10 Гб. Время непрерывной работы конвертера на кластере общего пользования CERN составляло примерно 1,5 дня.

На рисунке 10 приведен график зависимости средней скорости записи данных в CREST от количества пересылаемых пакетов для большого и малого объема данных. В папке «/PIXEL/PixMapOverlay» объем одного payload равен в среднем 42 килобайта. В папке «/Indent/Beampos» объем одного payload равен в среднем 220 байт. При маленьком объеме данных средняя скорость записи будет существенно ниже, чем при большом. Это логично, так как любой запрос имеет накладные расходы как в виде дополнительных данных, так и в виде дополнительного времени на каждое открытие и закрытие соединения с сервером. Для ускорения пересылки на сервер IOV группируются по 1000 штук в 1 запросе. Количество IOV выбрано так, чтобы гарантированно не иметь проблем с размером пересылаемых данных для любой из папок COOL, и при этом, чтобы время конвертирования было бы разумным. Как видно из графика (рис. 10) скорость существенно не меняется, что означает что сервер хорошо справляется с большим количеством IOV и объемом поступающих данных. Количество IOV для «/Indent/Beampos» гораздо больше, чем показано на графике (всего их 241186), но средняя скорость практически не изменяется.

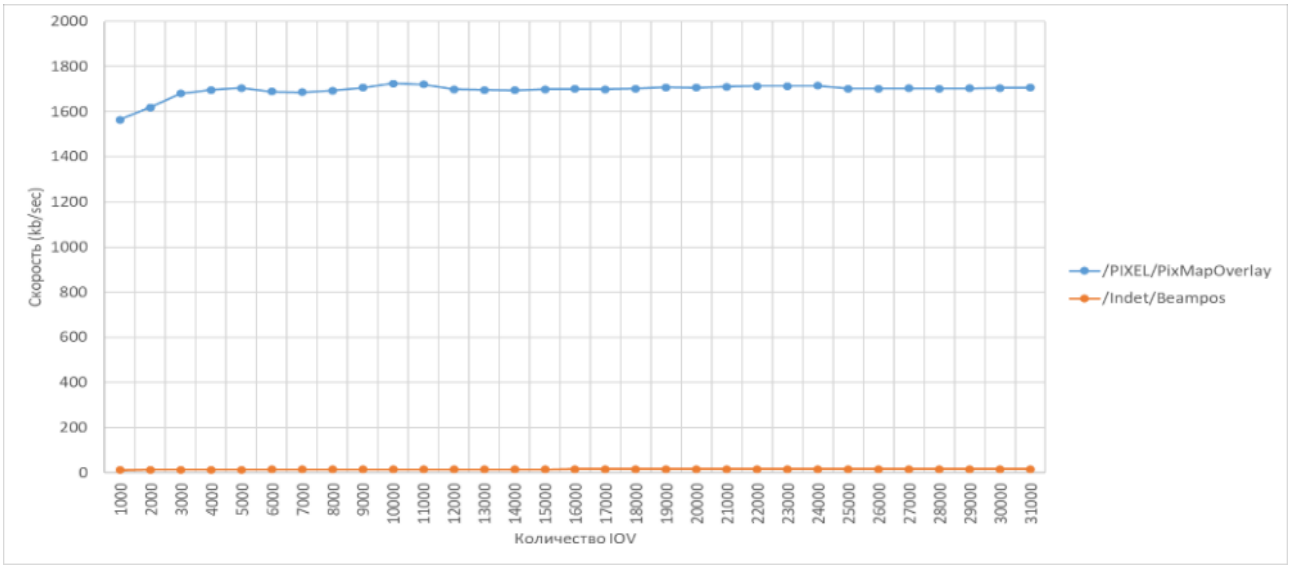

Рисунок 10: Зависимость средней скорости передачи от количества IOV

В результате тестирования был найден ряд проблем с производительностью CREST API и CREST server. Например, первоначально планировалось, что все IOV будут передаваться по одному в каждом запросе, но тесты показали, что большое количество IOV таким способом передать невозможно из-за слишком больших накладных расходов. Как результат, был реализован способ приема большого количества IOV в одном запросе.

В **заключении** сформулированы основные результаты диссертационной работы.

- Предложена методика построения геометрических ИС для экспериментов ФВЭ, на основе пакета FairRoot. С использованием данной методики создана геометрическая ИС для эксперимента BM@N.
- Разработана технология построения гибких сервисов для автоматизированного поиска событий в задачах ФВЭ. На ее основе реализован сервис Event Picking Service для эксперимента ATLAS.
- Разработана методика формирования конфигурационных БД и соответствующих систем на основе пакета FairRoot. На основе этой методики создан сервис для управления приложениями, запускаемыми во время сбора физических данных, эксперимента BM@N.
- Разработан метод интеграции новых динамических элементов в платформу Grafana. На основе этого метода проведена интеграция дерева в платформу Grafana. Данное дерево используется в обновленной системе мониторинга компьютерных сетей эксперимента ATLAS.

- Разработаны алгоритмы конвертирования различных данных Condition DB для эксперимента ATLAS из COOL формата в новый формат CREST, который планируется начать использовать в RUN4. Произведено тестирование алгоритма при больших объемах данных, подтвердившее стабильную работу сервера.

СПИСОК ОСНОВНЫХ ПУБЛИКАЦИЙ АВТОРА ПО ТЕМЕ ДИССЕРТАЦИИ

1. Alexandrov E., Formica A., Mineev M., Roe S. “The development of a new conditions database prototype for ATLAS RUN3 within the CREST project”, Изд: CEUR Workshop Proceedings, 3041, 86-90, 2021
2. Akishina E., Alexandrov E., Alexandrov I., Filozova I., Gertsenberger K., Ivanov V., Priakhina D., Shestakova G., “Development of the Geometry Database for the BM@N Experiment of the NICA Project”, EPJ Web of Conferences, Изд: EDP Sciences, 226, 03001, 2020
3. Alexandrov E., Kazymov A., Prokoshin F. “BigData tools for the monitoring of the ATLAS EventIndex”, CEUR Workshop Proceedings (CEUR-WS.org), ISSN:1613-0073, Изд: RWTH Aachen University, 2267, 91-94, 2018
4. Akishina E., Alexandrov E., Alexandrov I., Filozova I., Friese V., Ivanov V., “Development of the Geometry Database for the CBM Experiment”, Particles and Nuclei, Letters, Изд: JINR, Dubna, 15, 97-106, 2018
5. Akishina E., Alexandrov E., Alexandrov I., Filozova I., Gertsenberger K., Ivanov V. “Development of a Geometry Database and Related Services for the NICA Experiments”, Physics of Particles and Nuclei, ISSN:1063-7796, 52, 842-846, 2021
6. Akishina E., Alexandrov E., Alexandrov I., Filozova I., Friese V., Gertsenberger K., Ivanov V., Rogachevsky O., “Geometry Database for the CBM experiment and its first application to the experiments of the NICA project” CEUR Workshop Proceedings (CEUR-WS.org), ISSN:1613-0073, Изд: RWTH Aachen University, 2267, 504-508, 2018
7. Akishina E., Aleksandrov E., Aleksandrov I., Filozova I., Friese V., Ivanov V., “Experience of the Development of the Geometry Database for the CBM Experiment” EPJ Web of Conferences, ISSN:2100-014X, 214, 2019
8. Akishina E., Aleksandrov E., Aleksandrov I., Filozova I., Friese V., Ivanov V., “Geometry Database for the CBM Experiment”, CEUR Workshop Proceedings, ISSN:1613-0073, Изд: CEUR Workshop Proceedings, 2507, 306-310, 2019
9. Alexandrov E., Eukeni Pozo Astigarraga M., Avolio G. “Usage of Time Series Databases in the Grafana Platform for the Netis Service”, CEUR Workshop Proceedings, ISSN:1613-0073, 3041, 326-331, 2021

10. Alexandrov E., Alexandrov I., Barberis D., Prokoshin F., Yakovlev A. “Development of the ATLAS Event Picking Server”, CEUR Workshop Proceedings, ISSN:1613-0073, 3041, 223-228, 2021.
11. Alexandrov E., Alexandrov I., Gertsenberger K., Filozova I., Moshkin A., Chebotov A., Mineev M., Pryahina D., Shestakova G., Yakovlev A., Nozik A., Klimai P. «Development of Information Systems for Online and Offline Data Processing in the NICA Experiments», Physics of Particles and Nuclei volume 52, 801–807, 2021
12. Alexandrov E., Alexandrov I., Barberis D., Baranowski Z., Canali L., Dimitrov G, Fernandez Casani A., Gallas E., Montoro C., Gonzalez de la Hoz S., Hrivnac J., Iakovlev A., Kazymov A., Mineev M., Prokoshin F., Rybkin G., Sanchez J., Salt Cairols J., Vasileva P., Villaplana Perez M. “The ATLAS EventIndex and its evolution based on Apache Kudu storage” CEUR Workshop Proceedings (CEUR-WS.org), ISSN:1613-0073, Изд:RWTH Aachen University, 2267, 18-25, 2018
13. Alexandrov E., Aleksandrov I., Baranowski Z., Barberis D., Dimitrov G., Fernandez Casani A., Gallas E., Garcia Montoro C., Gonzalez de la Hoz S., Hrivnac J., Kazymov A., Mineev M., Prokoshin F., Rybkin G., Salt J., Sánchez J., Villaplana Perez M. “EI3 – the ATLAS EventIndex for LHC Run 3”, CEUR Workshop Proceedings, ISSN:1613-0073, Изд: CEUR Workshop Proceedings, 2507, 30-35, 2019
14. Alexandrov E., Alexandrov I., Baranowski Z., Barberis D., Dimitrov G., Fernandez Casani A., Elizabeth Gallas E., García Montoro C., Gonzalez de la Hoz S., Hřivnáč J., Kazymov A., Mineev M., Prokoshin F., Rybkin G., Sanchez J., Salt J., Villaplana Perez M. “Data-centric Graphical User Interface of the ATLAS Event Index Service”, EPJ Web of Conferences 245, 04036, 2020
15. Alexandrov E., Kotov V., Uzhinsky V., Zrelov P., “WEB service of Monte Carlo event generators in high energy physics” Изд: JINR, Publishing Department, 2011-130, 38-40, 978-5-9530-0312-4, 2011
16. Александров Е.И., Александров И.Н., Герценбергер К.В., Минеев М.А., Мошкин А.А., Пряхина Д.И., Филозова И.А., Чеботов А.И., Шестакова Г.В., Яковлев А.В., “Информационные системы для онлайн и офлайн обработки данных в современных экспериментах физики высоких энергий” Современные информационные технологии и ИТ-образование, том 15, №3, 645-650, 2019
17. Alexandrov E., Alexandrov I., Chebotov A., Gertsenberger K., Filozova I., Priakhina D., Shestakova G., «Status of the Configuration Information System

for the NICA Experiments», Physics of Particles and Nuclei Letters, 19, 543–546, 2022
18. E. Alexandrov, I. Alexandrov, A. Chebotov, K. Gertsenberger, I. Filozova, D. Priakhina and G. Shestakova, «Configuration Information System for online processing and data monitoring in the NICA experiments», Journal of Physics: conference series, 2438, 012019, 2023
19. Alexandrov E., Alexandrov I., Baranowski Z., Barberis D., Canali L., Cherepanova E., Dimitrov G. и др, «The ATLAS EventIndex: a BigData catalogue for all ATLAS experiment events», Computing and Software for Big Science, Изд:Springer-Verlag, 7, 2, 2023