МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет **ФИЗИЧЕСКИЙ**

Кафедра **ФИЗИКО-ТЕХНИЧЕСКОЙ ИНФОРМАТИКИ**

Магистерская программа **ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ И СИСТЕМЫ**

Направление подготовки **03.03.02 ФИЗИКА**

Образовательная программа **МАГИСТРАТУРА**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Каня Кирилл Олегович

(Фамилия, Имя, Отчество автора)

Тема работы Разработка программного обеспечения для онлайн монитора светимости детектора Belle II

«К защите допущена»

Декан ФФ НГУ

д. физ.-мат. наук

Блинов В.Е. / _____

(фамилия, И., О.) (подпись, МП)

«____» _____ 2022 г.

Научный руководитель

д. физ.-мат. наук

г.н.с. ИЯФ СО РАН

Кузьмин А. С. / _____

(фамилия, И., О.) (подпись, МП)

«____» _____ 2020 г.

Дата защиты: «____» _____ 2022 г.

Новосибирск, 2022

Содержание

1 Введение

С 2018 года на электрон-позитронном коллайдере SuperKEKB эксперимент Belle II перешел в фазу 3. Эксперимент направлен на изучение CP нарушения в распадах $B-$ и $D-$мезонов, а также на изучение τ лептонов. Проектная светимость коллайдера $6 \cdot 10^{35}$ $\text{с}^{-1}\text{см}^{-2}$. SuperKEKB является ассимметричным коллайдером, с энергией электронных и позитронных пучков $E_{e^-} = 7$ ГэВ и $E_{e^+} = 4$ ГэВ соответственно.

Одной из основных подсистем детектора является электромагнитный калориметр. Он предназначен для измерения фотонов и электронов в широком диапазоне энергий 20 МэВ – 8 ГэВ. Данные с электромагнитного калориметра используются для измерения онлайн и офлайн светимости.

В ходе эксперимента важно тщательно контролировать процесс набора данных, особенно при изучении редких распадов, таковыми являются изучаемые распады $B-$ и $D-$ мезонов в эксперименте Belle II. Измерение светимости является одним из способов контроля. Светимость показывает количество соударений частиц в пучке в единицу времени на единицу площади. Измеряя светимость в режиме реального времени, можно настраивать параметры ускорителя для оптимальной работы.

Для измерения светимости в режиме реального времени в Институте Ядерной Физики был разработан модуль онлайн монитор светимости. Монитор светимости измеряет скорость счета e^+e^- рассеяния с торцевых частей электромагнитного калориметра. Для работы монитора светимости и взаимодействия с ним было написано специальное программное обеспечение (ПО). Основными задачами ПО является: считывание данных с монитора светимости, первичная проверка качества данных, расчет интегральных светимостей за различные промежутки времени, архивирование, отображение данных и передача данных. Основная версия ПО реализована, но, поскольку Belle II является крупным экспериментом с большим числом

взаимодействующих компонент, требования к программному обеспечению быстро меняются и необходимо не только наладить стабильную работу текущей версии ПО, но и выработать процесс по поддержке и обновлению всех программ, связанных с монитором светимости. Также важно обеспечить проверку корректности данных, предоставить экспертные инструменты для детальной диагностики, автоматизировать некоторые задачи, выполняемые экспертами.

Исходя из этих требований, работа включает в себя следующие задачи:

- Модернизация процесса чтения данных. Повышение стабильности работы, быстродействия и расширение функционала.
- Расширение системы мониторинга качества данных.
- Автоматизация работы экспертов.
- Анализ данных монитора светимости .

2 Коллайдер SuperKEKB и эксперимент Belle II

2.1 SuperKEKB и детектор Belle II

Эксперимент Belle, являющийся предшественником эксперимента Belle II, проводился с 1999 по 2010 в лаборатории высоких энергий KEK, Цукуба. Он был направлен на изучение распадов B-мезонов и подтверждение CP-нарушения, предсказанного Макото Кобаяши и Тосихидэ Маскава, которые были награждены Нобелевской премией за данное открытие. Считается, что CP-нарушение является одной из причин наблюдаемого доминирования вещества над антиматерией в нашей нынешней вселенной. Однако измеренный уровень CP-нарушения далек от предсказанного в теории значения. Для более детального изучения этого явления, поиска Новой физики, а также постановки более строгих ограничений на стандартную модель был запущен новый эксперимент Belle II [1].

Коллайдер SuperKEKB, на котором проводится эксперимент Belle II, представляет собой электрон-позитронный ускоритель с ассиметричной энергией пучков ($e^- = 7$ ГэВ и $e^+ = 4$ ГэВ) (Рис. 1).

Проектная светимость коллайдера составляет $6 \cdot 10^{35}$ с$^{-1}$см$^{-2}$, что в 30 раз превышает значение, достигнутое в предыдущем эксперименте Belle. Для достижения проектных параметров, коллайдер KEKB, использовавшийся в предыдущем эксперименте Belle, был модернизирован и получил название SuperKEKB. Проектная светимость достигается путем уменьшения поперечного размера пучка и увеличения угла их столкновения. В новом эксперименте Belle II планируется набрать в 50 раз больше данных.

Поскольку электрон-позитронные столкновения будут происходить с гораздо большей скоростью, также необходимо модернизировать и детектор Belle II. Основные требования, предъявляемые к детектору[2]:

- Отличное разрешение вершин (≈ 50 мкм).

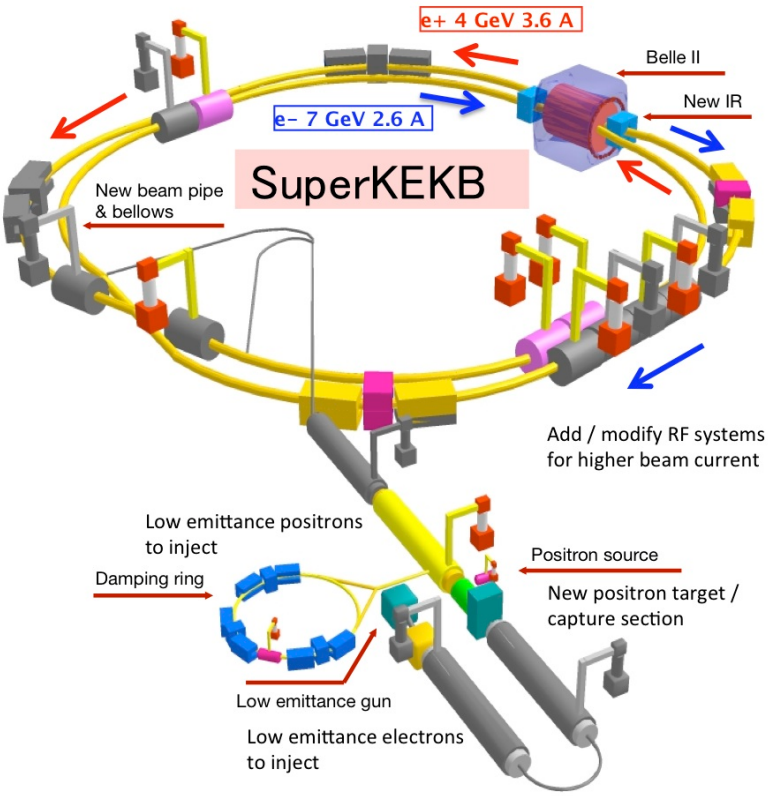

Рис. 1: Коллайдер SuperKEKB

- Очень высокая эффективность восстановления заряженных частиц с импульсами до нескольких сотен МэВ/c и улучшенная эффективность для заряженных частиц с импульсами до 50 МэВ/c.

- Очень хорошее разрешение импульса во всем кинематическом диапазоне эксперимента, т.е. до ≈ 8 ГэВ/c.

- Точные измерения энергии и направления фотонов от нескольких десятков МэВ до ≈ 8 ГэВ и эффективное обнаружение с 30 МэВ и выше.

- Высокоэффективная система идентификации частиц для разделения пионов, каонов, протонов, электронов и мюонов во всем кинематическом диапазоне эксперимента.

- Полный охват телесного угла.

- Быстрая и эффективная триггерная система, а также система сбора данных, способная хранить большое количество данных.

Детектор Belle II состоит из следующих подсистем (Рис. 2) [3]:

- Вершинный детектор (VXD) и пиксельный детектор (PXD), предназначенные для регистрации положения вершин распада частиц.
- Центральная дрейфовая камера (CDC), которая предназначена для регистрации траектории частиц, импульсов и удельного энерговыделения заряженных частиц.
- Времяпроекционные счетчики (TOP) и аэрогелевые черенсковские счетчики (ARICH), служащие для определения заряженных частиц.
- Электромагнитный калориметр (ECL), служит для регистрации фотонов и электронов, а также определения их энергий.
- Детектор K_L - мезонов и мюонов (KLM).

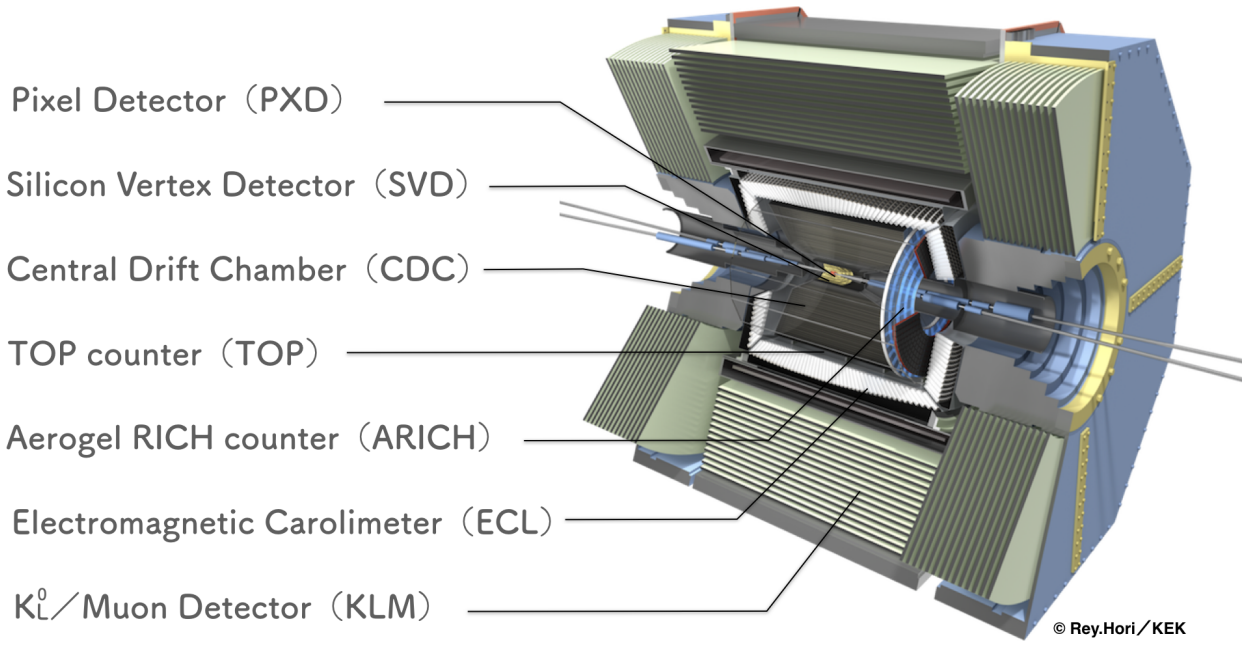

Рис. 2: Детектор Belle II

2.2 Электромагнитный калориметр

Продуктами распадов $B-$мезонов, изучаемых в эксперименте Belle II, в одной трети случаев являются частицы, распадающиеся на фотоны, например, нейтральные $\pi-$мезоны. Энергия фотонов находится в диапазоне от 20 МэВ до 4 Гэв. Поэтому электромагнитный калориметр, регистрирующий фотоны и измеряющий их энергию с высокой разрешающей способностью, является одной из основных подсистем детектора Belle II. В качестве сцинтилляционного кристаллического материала для калориметра Belle II был выбран CsI(Tl) из-за его высокого световыхода, относительно короткой длины излучения, хороших механических свойств и умеренной цены. Электромагнитный калориметр состоит из 8736 CsI(Tl) кристаллов и разделен на три части: торцевые – передняя (Forward) и задняя (Backward), а также цилиндрическая (Barrel) (Рис. 3). Основными задачами калориметра являются [4]:

- Регистрация фотонов и электронов.
- Идентификация электронов.
- Работа в широком диапазоне энергий 20 МэВ – 4 ГэВ в лабораторной системе отсчета.
- Измерение светимости: как онлайн, так и офлайн.

На каждом кристалле установлено 2 фотодиода, которые регистрируют сцинтилляционное излучение с кристаллов и преобразуют его в электрический сигнал. Далее сигнал с каждого фотодиода поступает на свой предусилитель, где преобразуется в импульс напряжения. Затем сигнал обрабатывается считывающей электроникой калориметра, представленной модулями ShaperDSP, в которых сигнал формируется, оцифровывается и по оцифрованным данным в ПЛИС (XILINX Spartan-3 [5]) вычисляется

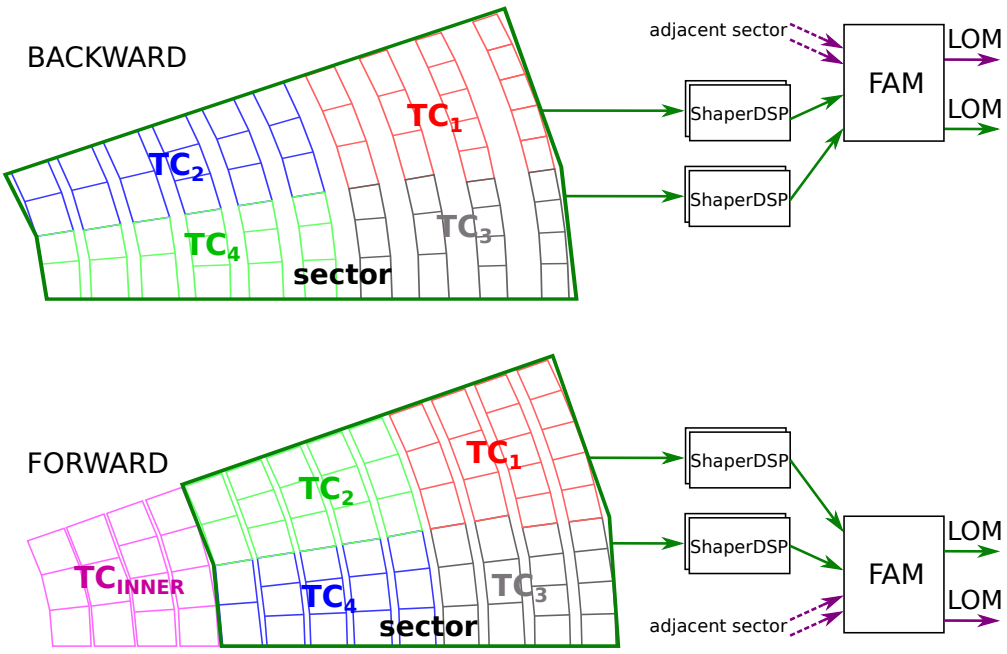

Рис. 3: Схема устройства триггерных ячеек в секторах и передача данных монитору светимости через FAM.

амплитуда и время срабатывания счетчика относительно сигнала триггера. Оцифровка осуществляется 18-битным АЦП (Analog device AD7641 [6]) с частотой 1,76 МГц. Каждый модуль обрабатывает сигналы с 8–16 кристаллов, которые образуют триггерную ячейку. Торцевые секции разделены на 16 секторов. Сектор содержит две триггерные ячейки, сигналы с которых, проходя через быстрый тракт формирования модуля ShaperDSP ($\tau_S \approx 0,2$ мкс), суммируются и передаются в модуль FAM (Flash-ADC trigger module). При превышении порога в ≈ 100 МэВ, модуль FAM определяет амплитуду и пиковое время триггерного сигнала, которые передаются в триггерную систему для формирования сигнала триггера, а также в модуль онлайн монитор светимости (LOM, Luminosity Online Monitor) (Рис. 3).

2.3 Мониторирование светимости

В эксперименте Belle II изучаются очень редкие распады $B-$ и $D-$ мезонов. Для данной цели был модернизирован коллайдер и его проектная светимость составляет $6 \cdot 10^{35}$ с$^{-1}$см$^{-2}$, что позволит набирать гораздо боль-

ший объем данных, чем в предыдущем эксперименте Belle. Даже небольшие изменения положения и размера пучка могут повлечь за собой сильное уменьшение светимости, а следовательно и уменьшение объёма набираемых данных.

Для мониторинга и настройки ускорителя для наиболее эффективной работы в эксперименте Belle II используются три монитора светимости работающих в режиме реального времени:

- Онлайн монитор светимости (LOM)
- Zero degree luminosity monitor (ZDLM)
- LumiBelle2

LumiBelle2 — монитор светимости, главной особенностью которого является скорость измерения светимости и возможность измерять вертикальный размер пучка. LumiBelle2 обнаруживает события радиационного электрон-позитронного рассеяния ($e^+e^- \to e^+e^-(\gamma)$)[7], в частности регистрируя отклоненный позитрон из позитронного кольца и фотон из электронного кольца. LumiBelle2 использует алмазные датчики, которые позволяют измерять светимость на частоте 1 кГц с точностью 0,1% [8]. Благодаря скорости работы, данный монитор способен измерять светимость отдельных пучков, а также используется для системы дизеринга.

ZDLM — монитор светимости, который измеряет светимость в относительных единицах и был разработан для первого эксперимента Belle. В эксперименте Belle II используется для дополнительной проверки качества данных совместо с LumiBelle2 и LOM. Принцип измерения светимости аналогичный LumiBelle2 (обнаружение рассеяния $e^+e^- \to e^+e^-(\gamma)$). В качестве датчиков используются черенковские счетчики и сцинтилляционные кристалы. Сигнал с обоих датчиков попадает в интегрирующий усилитель[9].

LOM — монитор светимости, который способен измерять светимость

в абсолютных единицах. В качестве данных LOM использует торцевые части электромагнитного калориметра, измеряя скорость счета событий электрон-позитронного рассеяния ($e^+e^- \to e^+e^-$).

2.4 Онлайн монитор светимости

Модуль онлайн монитор светимости (LOM) (Рис. 4) был разработан в ИЯФ СО РАН и произведен в 2016 году. Ключевой особенностью LOM

Рис. 4: Лицевая панель монитора светимости.

по сравнению с двумя другими мониторами светимости является более высокая точность измерения светимости (систематическая погрешность $\approx 2\%$), а также способность учитывать вето инжекции и мертвое время системы сбора данных с детектора. LOM способен измерять светимость в абсолютных единицах, в отличие от ZDLM и LumiBelle2, которые работают только в относительных единицах.

Монитор светимости направлен на измерение скорости счета e^+e^- рассеяния с торцевых частей электромагнитного калориметра, произошедшего в коллинеарных секторах торцевых частей (Рис. 6).

Для измерения скорости счета e^+e^- рассеяния в мониторе светимости используется ПЛИС Xlilinx Spartan-6 [10]. Для простоты взаимодействия со считывающим ПК используется микропроцессор с soft ядром. API монитора светимости работает на собственном протоколе, который использует протокол TCP в качестве транспортного.

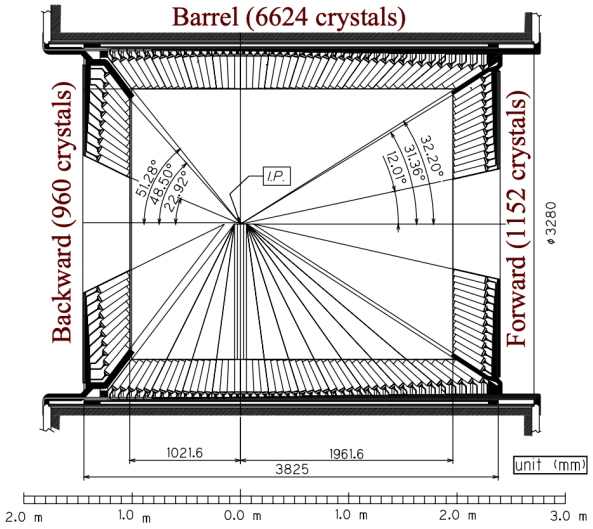

Рис. 5: Срез электромагнитного калориметра (ECL).

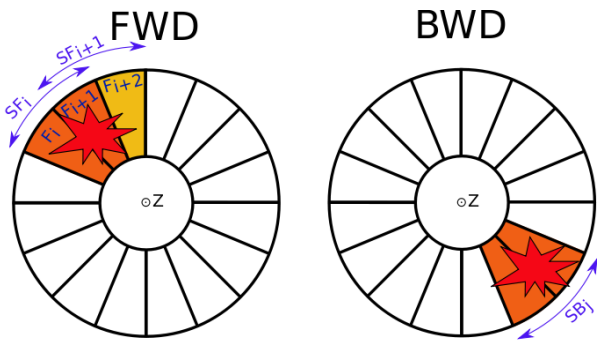

Рис. 6: Схема энерговыделения в секторах при возникновении события e^+e^- рассеяния.

Протокол устроен следующим образом. Первые 4 байта содержат в себе размер пакета, далее содержится строка обозначающая команду, которую необходимо выполнить. Строка записывается в стиле языка программирования C, следовательно она нуль терминированная. Далее содержатся данные к команде, набор регистров или данные которые нужно в них записать. Все слова имеют размер 4 байта и данные должны быть в Big-endian порядке. Формат ответа следующий. Первые 4 байта содержат длину пакета, далее нуль терминированная строка с сообщением ответа. За строкой содержатся данные ответа. Пример запроса и ответа к модулю представлен на Рис. 7.

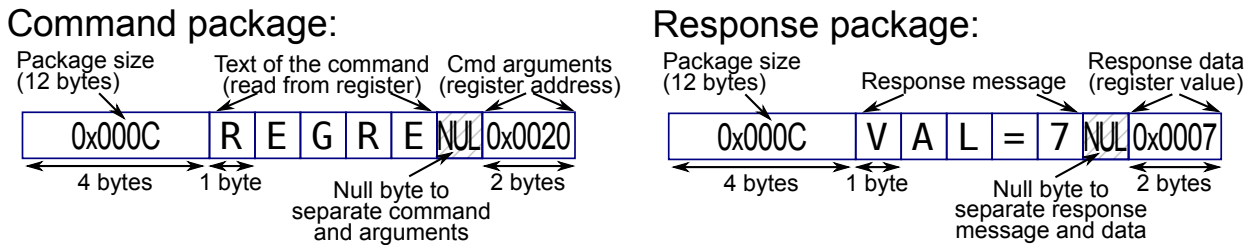

Рис. 7: Формат запроса на чтение регистра из монитора светимости.

Для улучшения быстродействия, API также предоставляет возможность в один запрос прочитать или записать значения группы регистров.

2.5 Система медленного контроля

Для обеспечения доступа к параметрам систем детектора Belle II и ускорителя в эксперименте используются две системы медленного контроля: Network Shared Memory 2 (NSM2) используется для сбора данных и контроля электроники детекторных систем, Experimental Physics and Industrial Control System (EPICS) используется в эксперименте Belle II для контроля систем и параметров ускорителя (светимость, токи пучков, уровень фона и т.д.).

Система NSM2 была разработана коллаборацией Belle II и является развитием предыдущей системы NSM. На данную систему возлагаются следующие основные задачи [11]:

- Контроль последовательности запуска и остановки всей системы сбора данных.
- Получение сообщений и статусов от подсистем системы сбора данных.
- Контроль системы высокого напряжения.
- Сбор мониторинговой информации.

NSM2 предоставляет два основных механизма обмена информацией по локальной сети, функционирующей на основе стека протоколов TCP/IP. Первый механизм предназначен для синхронизации заданного пространства памяти через передачу широковещательных пакетов UDP по различным хостам в сегменте сети. Второй механизм, под названием NSM-сообщение, используется для отправки данных при помощи пакетов TCP в пользовательской программе без каких-либо знаний о механизме TCP/IP или системных вызовах, адресах хоста или сетевых конфигурациях.

Система медленного контроля EPICS использует архитектурные подходы клиент/сервер и публикация/подписка для связи между различными компьютерами. Большинство серверов, называемых контроллерами ввода-вывода (IOC), выполняют реальные задачи ввода-вывода и локального

управления и публикуют эту информацию клиентам, используя надежные сетевые протоколы, разработанные для EPICS: Channel Access (CA) и pvAccess. Эти протоколы разработаны для высокоскоростных сетевых приложений, работающих в режиме реального времени, в которых используется EPICS [12]. Хранилище данных представляет собой распределенную базу данных. Данные идентифицируются с использованием уникальных идентификаторов, называемых Process Variables (PVs). Эти PV доступны по протоколу CA[13].

3 Программное обеспечение монитора светимости

3.1 Прошивка и программа ЦПУ

Данные со входа монитора светимости оцифровываются 4-мя восьмиканальными 12-битными АЦП, далее оцифрованные данные попадают на ПЛИС Xilinx Spartan-6. Основные задачи ПЛИС модуля:

- Измерение скорости счета e^+e^- рассеяния по оцифрованным входным сигналам.
- Запись данных во внутреннюю память.
- Взаимодействие со считывающим ПК.

Как было упомянуто в прошлой главе, в ПЛИС монитора светимости используется микропроцессор с soft ядром MicroBlaze [14]. Использование данного процессора позволяет достаточно легко реализовать взаимодействие со считывающим ПК через стек протоколов TCP/IP. Кроме того, процессор имеет доступ к адресному пространству и регистрам ПЛИС, таким образом предоставляя удобный способ для управления модулем. Процессор по принятой команде может как считать данные из памяти и отдать их, так и записать необходимые параметры.

Программа ЦПУ написана на языке C, в качестве библиотеки, реализующей стек TCP/IP используется lwIP [15]. При старте программа инициализирует lwIP, задает себе IP и MAC адрес, затем начинает ожидать подключения по указанному адресу. Если программа ЦПУ не является операционной системой, то lwIP запускается в режиме mainloop. Это означает, что входящие подключения не могут обрабатываться в отдельных потоках, может использоваться только callback API. Такой режим сложнее в разработке, зато позволяет использовать минимальный объем памяти и получить максимальную производительность. В данном режиме пакеты

складываются в очередь и по прерыванию таймера происходит обработка. Такой способ накладывает ограничения на количество сессий обрабатываемых программой ЦПУ. Поскольку обработка происходит в прерывании, то при обработки двух или более сессии возможны коллизии.

3.2 Считывающее ПО

Считывающее ПО является центральной компонентой, взаимодействующей с монитором светимости. Оно совмещает в себе несколько различных задач. Так, из-за невозможности параллельной обработки нескольких подключений ввиду ограничений режима mainloop lwIP, потребовалось реализовать прокси-сервер, который будет выставлять очередность запросов к монитору светимости. Также, для снижения нагрузки на монитор светимости, считывающее ПО кэширует значения, полученные с монитора светимости. При запросе данных у считывающего ПО, оно отдает значения, находящиеся в кэше, и не выполняет повторного запроса к монитору светимости. Таким образом получается кэширующий прокси-сервер, который имеет Application Programming Interface (API) для взаимодействия с монитором светимости.

Следующей важной задачей является расчет различных параметров ускорителя в режиме реального врмени:

- Светимость в абсолютных единицах.
- Интегральные светимости за различные промежутки времени.
- Максимальные светимости за различные промежутки времени.
- Светимости с учетом энергии пучков.
- Пьедесталы для каждого сектора.

Все перечисленные данные считывающее ПО сохраняет в две системы медленного контроля (EPICS и NSM2). В качестве библиотеки, реализующей взаимодействие с системой медленного контроля EPICS, используется PythonIOC [16]. Для обеспечения стабильной работы и восстановления данных при перезагрузках системы все интегральные значения и текущий контекст (номер захода, номер эксперимента, время) требуется записывать в базу данных. В качестве базы данных используется SQLite [17].

Также считывающее ПО записывает все данные, которые были считаны с монитора светимости, в файлы. Схема считывающего ПО представлена на рисунке 8.

Рис. 8: Схема устройства и взаимодействия со считывающим ПО.

Поскольку поток данных с монитора светимости невелик, то не накладывается серьезных ограничений на производительность ПО. Поэтому, а также для ускорения процесса разработки, был выбран язык Python. Диаграмма классов считывающего ПО представлена на рисунке 9. Для удобства взаимодействия с монитором светимости была реализована библиотека LOM Interface, которая поддерживает команды, реализованные в прошивке. Кроме того, она реализует сериализацию и десериализацию данных. Данная библиотека состоит из двух классов: LOMController и LOMSession. LOMSession предназначен для установления соединения с монитором све-

тимости по протоколу TCP; сериализацию и отправку данных; получение данных и последующую их десериализацию. В качестве библиотеки для сериализации используется struct. Кроме того, LOMSession используется для взаимодействия между прокси и произвольным числом подключенных клиентских программ.

LOMController используется для установки и чтения конфигурации монитора светимости. Так в данном классе считываются с базы данных актуальные величины калибровочных коэффициентов, а также реализована их подстройка с учетом аттенюаторных коэффициентов. Также в данном классе устанавливаются пороговые значения монитора светимости. Данный класс выступает в качестве интерфейса библиотеки и проксирует команды в LOMSession. Считывающее ПО использует данную библиотеку для взаимодействия с монитором светимости.

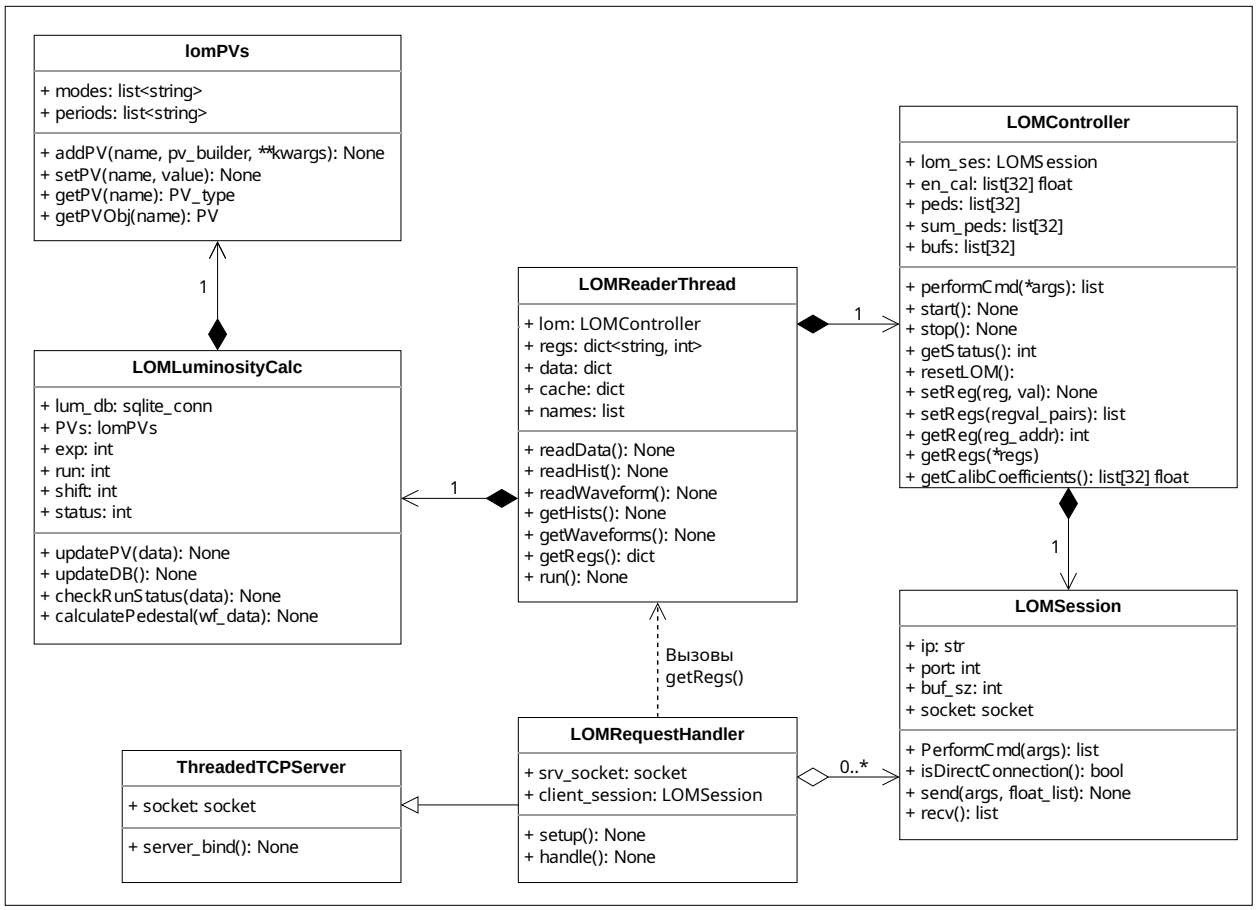

Рис. 9: Диаграмма классов считывающего ПО.

За считывание данных с монитора светимости с частотой 1 Гц отвечает класс LOMReaderThread. Данный класс считывает и кэширует последние считанные значения. Также данный класс записывает все считанные данные в файлы по заходам для каждого эксперимента. Далее LOMReaderThread отправляет полученные данные в LOMLuminosityCalc.

LOMLuminosityCalc отвечает за расчет всех значений интегральных светимостей, запись их в локальную базу данных и запись в систему медленного контроля EPICS. Также этот класс расчитывает значения пьедесталов по формам сигналов. Для взаимодействия с системой медленного контроля EPICS данный класс использует lomPVs класс, который отвечает за создание PV и предоставление доступа к ним.

В качестве реализации API используется класс LOMRequestHandler, который обрабатывает входные подключения, синхронизует доступ к кэшу из LOMReaderThread, а также предоставляет доступ для отправки запросов напрямую в монитор светимости.

3.3 Интегральные светимости и система медленного контроля EPICS

Модуль онлайн монитор светимости измеряет скорость счета e^+e^- рассеяния с торцевых частей электромагнитного калориметра. Считывающее ПО занимается расчетом светимости в абсолютных единицах. Расчет абсолютного значения светимости через скорость счета e^+e^- рассеяния выражается следующим образом:

$$L = \frac{1}{\epsilon\sigma}\frac{dN}{dt}, \tag{1}$$

где L светимость коллайдера, $\frac{dN}{dt}$ скорость счета e^+e^- рассеяния, а произведение $\epsilon \cdot \sigma$ – видимое сечение регистрации (σ_{vis}). Данная формула была реализована в считывающем ПО для расчета светимости, а видимое сечение регистрации было определено в результате моделирования методом Монте-Карло $\sigma_{vis} = 29.38$ нб.

В эксперименте Belle II эта величина называется *ускорительной светимостью*. Также выделяется *детекторная светимость* L_{det}, в которой светимость вычисляется только в период работы системы сбора данных, то есть пропорциональна доле живого времени детектора:

$$L_{det} \approx L \cdot \frac{t_{live}}{t_{total}}. \tag{2}$$

Используя светимость в абсолютных единицах, расчитанную по формуле 1, производится расчет интегральных и максимальных светимостей. Различным группам экспертов требуются интегральные и максимальные светимости за различные промежутки времени. Можно выделить основные требования групп экспертов к интегральным и максимальным светимостям:

- Группа операторов ускорителя. Для получения обратной связи с ускорителя с целью подбора оптимальных значений данной группе требуются следующие значения:
  - Мгновенная ускорительная светимость.
  - Интегральные ускорительные светимости за различные промежутки времени.
- Группа операторов детектора. Для анализа эффективности работы детектора данной группе необходимы следующие значения:
  - Получать мгновенную детекторную и ускорительную светимости.
  - Получать аналогичные интегральные светимости за различные промежутки времени.
- Группа экспертов по ECL. Для отслеживания корректности работы электромагнитного калориметра данной группе необходимо получать следующие данные:
  - Значения пьедесталов для каждого сектора.

  - Формы сигналов с каждого сектора.
  - Амплитудные гистограммы для каждого сектора.

- Группа экспертов по обработке данных. Для отбора заходов по светимостям данной группе необходимы следующие значения:
  - Интегральные светимости по заходам.

- Группа руководителей эксперимента. Данной группе для отслеживания достижения проектных значений светимостей необходимо получать следующие данные:
  - Значения интегральных светимостей.
  - Значения максимальных светимостей.

Все значения светимостей записываются в систему медленного контроля EPICS используя библиотеку pythonIOC. Каждую итерацию считывания данных с монитора светимости, происходит расчет светимостей и их запись в EPICS, а также в локальную базу данных SQLite.

3.4 Графический интерфейс монитора светимости

Для удобного отображения основных данных и параметров с монитора светимости используется графический пользовательский интерфейс (GUI). Интерфейс имеет два окна: основное и окно для отображения гисторамм. Интерфейс основного окна GUI представлен на рисунке 10.

Основное окно разделено на 6 областей:

- Раздел конфигурации.
- Область отображения пьедесталов.
- Область отображения счетчиков.

- Область отображения светимости и фоновых событий.
- Раздел просмотра амплитуд.
- Окно журналирования.

Рис. 10: Основное окно графического интерфейса онлайн монитора светимости.

В конфигурационном разделе отображается значение пороговых значений для амплитуд передней и задней торцевой части. Также отображается пороговое значение региона совпадения. Имеется возможность установить частоту считывания данных с монитора светимости и запустить процесс считывания данных с заданной частотой.

Область отображения пьедесталов показывает пьедесталы для каждого из 32 секторов. Расчет пьедесталов происходит в считывающем ПО в режиме реального времени по формам сигнала.

В разделе счетчиков отображаются основные времена и количество посчитанных событий. В разделе светимости отображается график мгно-

венной светимости за последние N минут и текущее значение мгновенной светимости. Под графиком аналогичным образом отображается график фоновых событий и текущее значение фоновых событий.

В области форм сигнала, отображаются формы сигналов 2-х секторов. По умолчанию показываются те два сектора, в которых было зарегистрировано событие e^+e^- рассеяния. Также можно выбрать отдельно сектора для отображения из передней и задней торцевой части. На графиках форм сигнала помимо самой формы сигнала отображается пороговое значение для данного сектора в виде красной пунктирной линии. Справа от графиков отображается тепловая карта секторов, в которой показывается энерговыделение в секторах. Также показывается график региона совпадения.

В окне журналирования показывается журнал работы графического интерфейса. Имеется возможность задать уровень журналирования. Поддерживаюстся 3 уровная:

- INFO – журналируются основные события работы интерфейса
- ERROR – журналируются только ошибки возникшие в процессе работы
- DEBUG – журналируются наиболее детально все этапы работы интерфейса

Журнал также по умолчанию записывается в файл.

В окне отображения гистограмм показываются гистограммы энерговыделения для каждого сектора. В окне отображаются 4 гистограммы и имеется возможность переключаться на следующие 4 сектора вперед и назад. Также можно выбрать избранные сектора и показывать их. Окно отображения гистограмм представлено на рисунке 11.

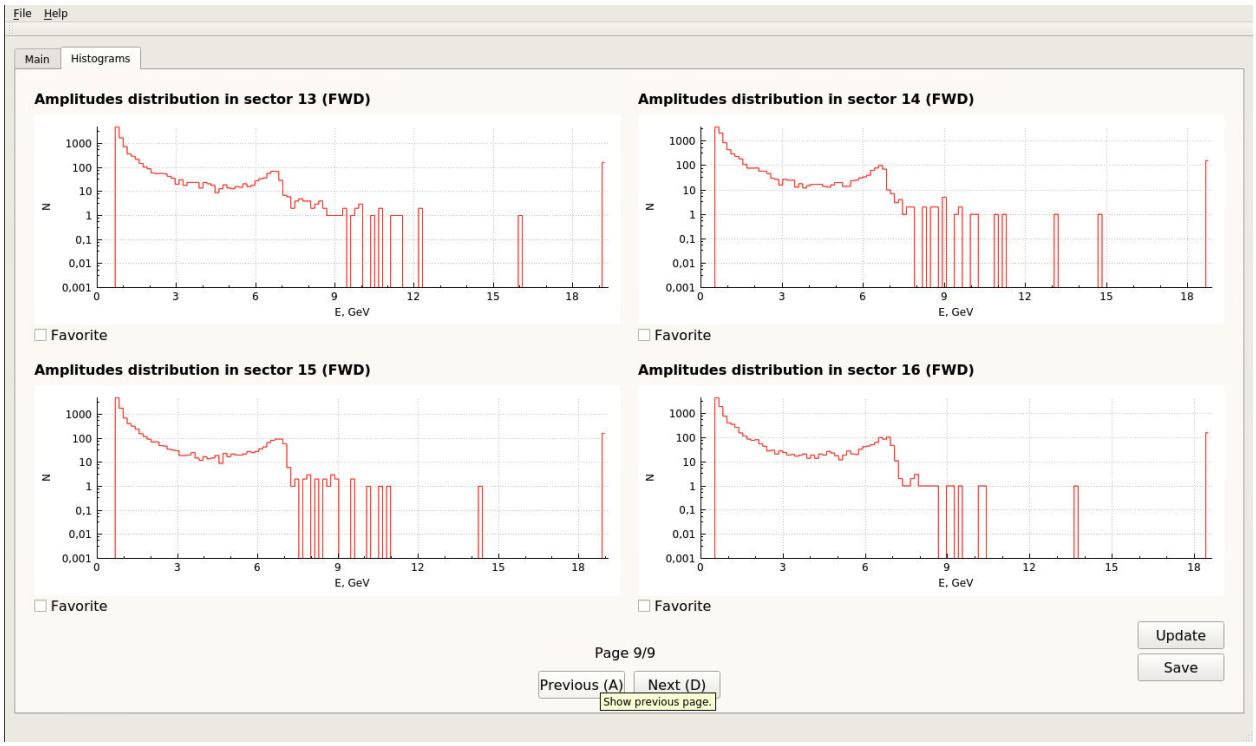

Рис. 11: Окно отображения гистограмм энерговыделения в торцевых частях ECL.

4 Модернизация системы сбора данных

4.1 Программа ЦПУ и оптимизация существующей системы

Программное обеспечение в эксперименте Belle II активно развивается, возникают новые задачи к мониторированию светимости. Таким образом, чтобы удовлетворять новым требованиям, важной задачей является оптимизация существующего ПО.

Одна из основных задач считывающего ПО — считывать значения с монитора светимости с частотой 1 Гц. Исходя из этого требования, необходимо минимизировать время на обработку считываемых значений. Основная часть обработки заключается в расчете интегральных и максимальных светимостей за различные промежутки времени. После расчета всех светимостей требуется их записывать в базу данных. Запись в базу данных после исполнения запроса требует обращения к диску, что значительно сказывается на времени обработки значений. Для уменьшения запросов к диску, было понижено требование к транзакции и минимизировано число записей при расчете различных значений светимости. Для считывания значений наиболее точно с частотой 1 Гц, был добавлен динамический таймаут в зависимости от времени, затраченного на обработку значений. Также при расчете светимостей требуется считывать различные параметры ускорителя с системы медленного контроля EPICS. В случае проблем с сетью, таких как большие задержки, или в случае недоступности EPICS или других различных факторов обработка зависает на таких участках кода. Для таких случаев в функцию, считывающую значения, добавлен таймаут по истечении которого сообщается об ошибке. В таком случае используется заранее определенное стандартное значение (в большинстве случаев используется значение за прошлую секунду). На рисунке 12 представлен график, на котором показано время затрачиваемое на обработку события. Видно,

что время на обработку событий значительно сократилось после 11 ноября 2020 г. около 12:00. В это время было обновлено считывающее ПО, в новой версии были включены все описанные оптимизации.

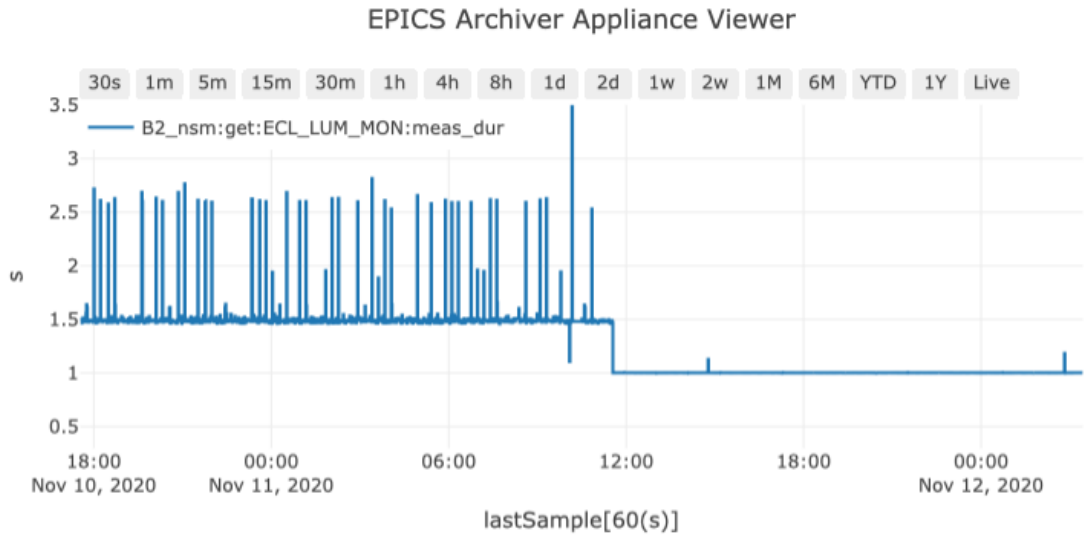

Рис. 12: Время, затрачиваемое на считывание и обработку событий.

Со стороны программы ЦПУ монитора светимости существует серьезное ограничение, что программа ЦПУ способна обрабатывать только одно подключение без возникновения взаимных исключений. Для сетевого взаимодействия со считывающим ПО используется библиотека lwIP. Поскольку программа ЦПУ не реализует операционную систему и не реализует многопоточность, то lwIP неспособен работать в режиме многопоточности [18]. В режиме mainloop пакеты записываются во внутренний буфер lwIP и обрабатываются в прерывании таймера. Проблема заключается в том, что при обработке пакетов внутри обработчика прерывания, может прийти еще одно прерывание. В таком случае возникает проблема взаимного исключения.

Чтобы избежать проблемы взаимных исключений, нужно вынести участки кода, которые долго исполняются из функции обработчика прерывания. Для избежания описанной проблемы, в обработчике прерывания только

выставляется флаг, о готовности считывать с буфера. В основном цикле программы данный флаг проверяется и если он установлен, то запускается обработка пакетов из буфера в основном цикле программы. В таком подходе при возникновении прерывания в момент обработки пакетов, флаг снова выставляется и не возникает проблемы взаимного исключения.

4.2 Модернизация программной архитектуры

После оптимизации работы существующей версии ПО, была проведена модернизация: добавлены новые возможности в API, новые переменные светимостей, проведен рефакторинг кода.

В рамках рефакторинга кода был упрощен процесс создания и обновления светимостей. Также улучшен процесс записи светимостей в EPICS. Для просмотра истории и сопоставления различных значений светимостей и других параметров используется EPICS Archiver [19]. EPICS Archiver позволяет построить график нужных величин в выбранном диапазоне времени. Для удобства просмотра истории в данной системе, для каждой переменной светимости было добавлено ее краткое описание и единицы измерения. Пример отображения светимости за фазу 3 эксперимента Belle II представлен на рисунке 13

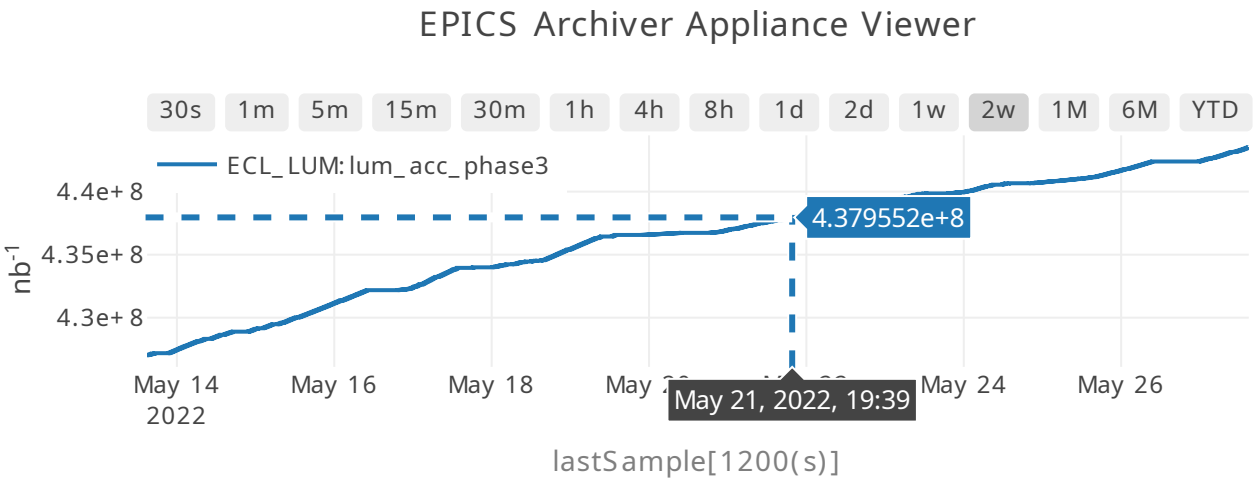

Рис. 13: Светимость набранная за фазу 3 (с начала 2019 года) в EPICS Archiver.

По запросу группы операторов ускорителя, добавлен набор PV с информацией о светимости за предыдущий день. Также добавлены новые переменные и реализован расчет светимостей с учетом энергии пучков. Каждую итерацию после считывания, при расчете светимостей из EPICS считываются энергии пучков, далее рассчитывается поправка на изменение сечения рассеяния от энергии и поправка на изменение эффективности регистрации от энергии. Поправки были получены в результате моделирования эксперимента методом Монте-Карло.

Для синхронизации данных и параметров монитора светимости расширено API монитора светимости. Добавлена возможность отправлять числа с плавающей точкой. Добавлены новые методы для получения коэффициентов энергетической калибровки, метод для остановки и продолжения чтения данных

4.3 Автоматизированные тесты

В программном обеспечении, которое активно развивается, дорабатываются существующие компоненты и добавляется новый функционал, важно сохранить стабильность работы. Любые вносимые изменения не должны нарушить работу существующих компонент. На практике часто используют автоматические тесты для проверки корректной работоспособности компонент ПО. Автоматические тесты представляют собой набор данных, которые подаются на вход тестируемой системы, затем после обработки данных, полученные выходные значения сравниваются с ожидаемыми. Если значение не совпадают, то тест считается неуспешным. Подача данных на вход и сравнение с ожидаемым значением происходит автоматически при запуске тестов.

Таким образом, сгенерировав набор стандартных входных данных и соответствующих им выходных данных проверяется написанное ПО. Также, часто проверяется, корректно ли система обрабатывает случаи плохих

входных данных. Каждый набор тестовых данных, проверющий какой-либо конкретный сценарий использования ПО, называется тестовым случаем. Существует множество различных фреймворков, которые созданы для упрощения создания тестовых случаев, предварительной инициализации и сброса состояния системы, удобного отображения процента пройденных тестовых случаев, а также для вывода набора тестовых данных, которые вызывают ошибку тестируемого ПО.

Для считывающего ПО особенно важно быть уверенным в корректности расчета интегральных и максимальных светимостей. Данная часть ПО наиболее часто подвергается изменениям. Чтобы избежать ошибок при внесении изменений потребовалось написать набор автоматизированных тестов, которые покрывают основные сценарии работы программы расчета значений светимости. Для написания автоматических тестов выбран фреймворк pytest [20]. Он позволяет удобно объединять тестовые сценарии в классы, проводить инициализацию и сброс системы после каждого тестового случая или после определенного набора тестовых случаев, объединенных в класс. Также этот фреймворк позволяет избежать дублирования кода для тестирования схожих компонент. В нём используется встроенный в язык Python механизм assert для проверки тестовых случаев. При тестировании расчета интегральных светимостей важно проверить следующие случаи:

- Задержки сети при считывании параметров с EPICS.
- Сброс светимостей при завершении соответствующего промежутка времени.
- Расчет светимости в не-физических заходах.

Написана базовая версия тестов для светимостей, реализована возможность смоделировать задержку сети и задать различные параметры ускорителя.

4.4 Микросервисная архитектура

Поскольку эсперимент Belle II постоянно развивается, увеличивается набор данных, то возрастают требования ко всем системам. Важно обеспечивать стабильность работы, иметь возможность добавлять новый функционал в уже работающие компоненты и в случае возникновения сбоев быстро детектировать и исправлять их.

В считывающем ПО онлайн монитора светимости часто добавляется новый функционал в расчет интегральных светимостей и во взаимодействие с системой медленного контроля EPICS. Часть кода, отвечающаяя за считывание данных, остается неизменной, но к ней предъявляются особые требования к скорости и стабильности работы. Считывание должно происходить с частотой 1 Гц. Чтобы удовлетворить этим требованиям и упростить внесение нового функционала, было принято решение перенести считывающее ПО на микросервисную архитектуру.

Микросервисная архитектура – это подход в разработке ПО, в котором компоненты ПО разделяются на небольшие независимые сервисы, взаимодействующие по сети. Под независимостью подразумевается, что каждый сервис может быть обновлен или заменен, не затрагивая другие сервисы. Поскольку каждый сервис работает в отдельном процессе и каждый сервис независим, то возникновение ошибок в одном из них никак не влияет на работу остальных сервисов.

Перенос считывающего ПО на микросервисную архитектуру, позволит легко вносить изменения в различные компоненты. Сбои одного сервиса не коснутся другого. Исходя из задач, выполняемых считывающим ПО, были выделены следующие сервисы:

- Кэширующий прокси.
- Сервис, рассчитывающий светимости и пишущий в EPICS.

- Сервис, пишущий в файл.
- API шлюз.

Кэширующий прокси будет выполнять 3 основные задачи:

- Считывать и кэшировать значения с монитора светимости.
- Выставлять очередность запросов к монитору светимости (Прокси).
- Отправлять считанные данные другим сервисам.

Все данные задачи требуют активного взаимодействия с монитором светимости, поэтому были объединены в один сервис.

Сервис по расчету светимостей будет расчитывать светимости, сохранять в локальную базу данных и взаимодействовать с системой медленного контроля EPICS.

Сервис для записи в файл, будет только записывать значения с монитора светимости в файл.

API шлюз как отдельный сервис будет доступен извне и будет отвечать за то, чтобы определить, к какому сервису отправить текущий запрос на обработку. Тем самым достигается изоляция внутренней архитектуры от пользователей.

Важный этап при проектировании миксросервисов — определить каким образом они будут взаимодействовать. Концептуально существуют 2 подхода: синхронный и асинхронный. Синхронный в микросервисной архитектуре используется крайне редко, поскольку в нем используется вызов удаленных процедур (RPC). В этом подходе сервис, выполняющий RPC запрос, должен дождаться его исполнения и получить ответ. Таким образом возрастает зависимость сервисов.

В считывающем ПО выбрано асинхронное взаимодействие с использованием очереди сообщений. В качестве библиотеки, реализующей очередь

сообщений, используется ZeroMQ [21]. Итоговая архитектура представлена на рисунке 14.

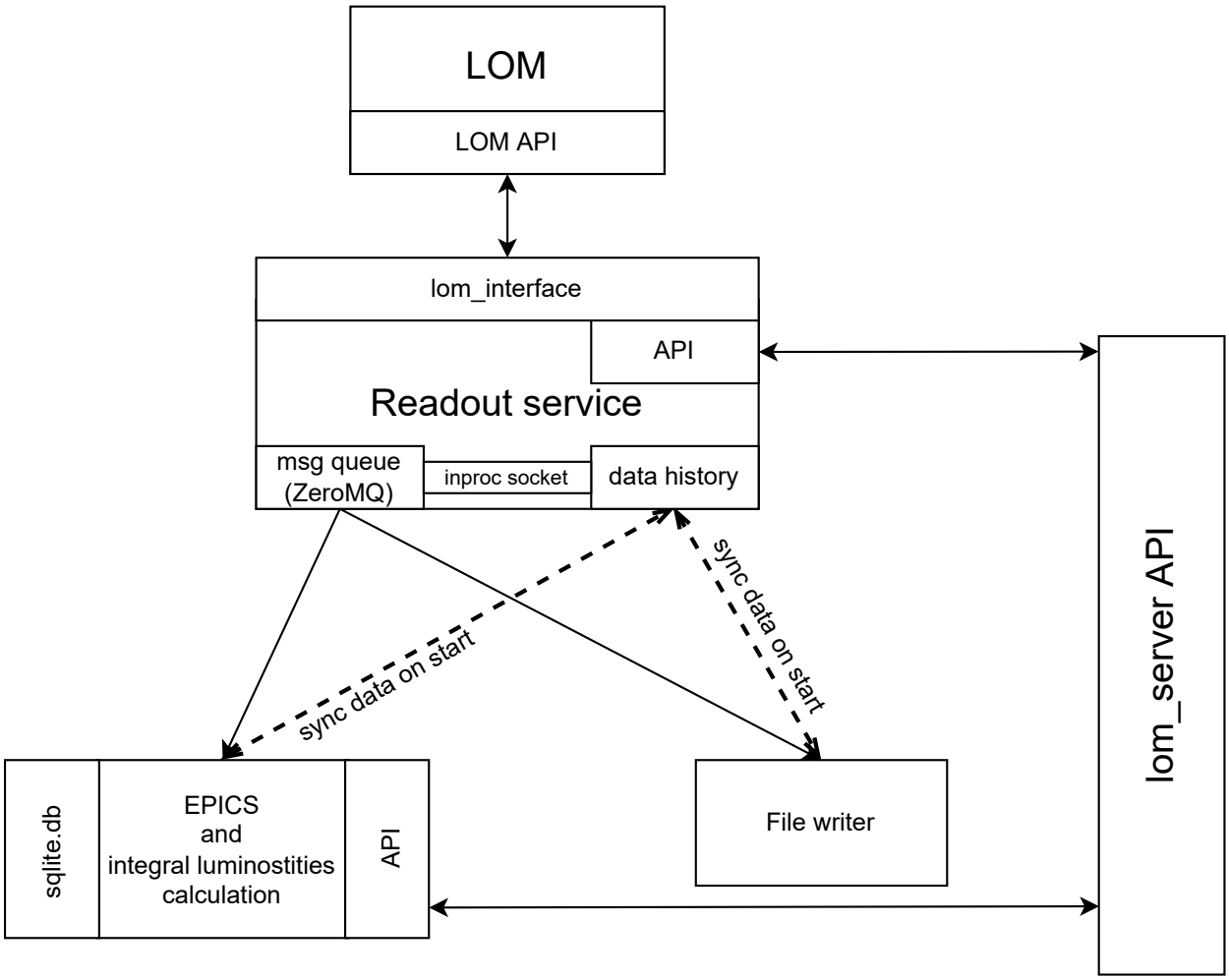

Рис. 14: Микросервисная архитектура считывающего ПО.

Кэширующий прокси после считывания значений будет отправлять сообщения с новыми данными. Сервисы, которые обрабатывают данные с монитора светимости подписываются на сообщения и они записываются в очередь и далее обрабатываются. В таком подходе отправителю не важно работает сервис подписчик или нет. Важно предусмотреть презапуск сервиса подписчика. Так, если кэширующий прокси сервис отправит сообщение с новыми данными, а сервис подписчик в данный момент не работал, то при запуске подписчика, он не получит утраченные сообщения. В случае расчета светимостей и записи данных в файл, требуется реализовать меха-

низм получения истории сообщений. В кэширующем прокси был реализован механизм хранения истории данных с монитора светимости. Сервис отправляет сообщение и после этого записывает сообщение в историю. История представляет собой двустороннюю очередь фиксированного размера. Если новое сообщение добавляется, то из очереди удаляется самое старое ссобщение. Каждое сообщение имеет уникальный идентификатор и он постоянно увеличивается. Таким образом история представляет собой очередь, в которой все события отсортированы по идентификаторам. Сервис подписчик получает сообщение и записывает идентификатор в файл, таким образом при перезапуске сервиса он знает идентификатор последнего обработанного сообщения. При запуске сервиса подписчика, прежде чем начать обрабатывать новые сообщения, делается запрос с указанием идентификатора последнего обработанного события. Сервис отправитель сообщений получает запрос и отправляет все события у которых идентификатор больше того, который отправил сервис подписчик. Поскольку история это отсортированная очередь, то поиск в ней имеет асимптотику $O(log(N))$, где N размер истории. Для нахождения идентификатора в истории используется алгоритм бинарного поиска.

ZeroMQ — программный пакет, который предоставляет сокеты для передачи атомарных сообщений. Данный программный пакет используется для организации различного взаимодействия: межпроцессного, межпотокового, сетевого по протоколу TCP и широковещательного. Можно соединять сокеты по схемам: многие ко многим используя механизм публикация-подписка, один к одному (запрос-ответ), а также один ко многим для реализации балансировки нагрузки. Данный фреймворк использует асинхронный подход для операций ввода-вывода, что позволяет реализовывать масштабируемые многопоточные приложения. ZeroMQ поддерживает множество языков программирования и работает на большинстве современных операционных системах.

Поскольку история сообщений работает в отдельном потоке, то требуется синхронизировать общие данные при доступе к ним. ZeroMQ рекомендует отказаться от синхронизации потоков стандартными методами языков программирования и перенести синхронизацию на уровень ZeroMQ. Для данной цели в ZeroMQ существуют сокеты специального типа (inproc), которые имеют интерфейс аналогичный обычным сокетам. Такое общение потоков становится безопасным, поскольку синхронизация осуществляется внутри библиотеки ZeroMQ.

5 Автоматизация работы

5.1 Энергетическая калибровка

Монитор светимости оцифровывает приходящий аналоговый сигнал, выдавая величину в каналах АЦП. Чтобы корректно задавать энергетические пороги на события e^+e^- рассеяния, а также чтобы отслеживать стабильность электроники монитора светимости, необходимо регулярно определять, как каналы АЦП пересчитываются в единицы энергии. Для этого используется процедура энергетической калибровки, определяющей калибровочный коэффициент для каждого сектора. Схема процедуры энергетической калибровки представлена на рисунке 15.

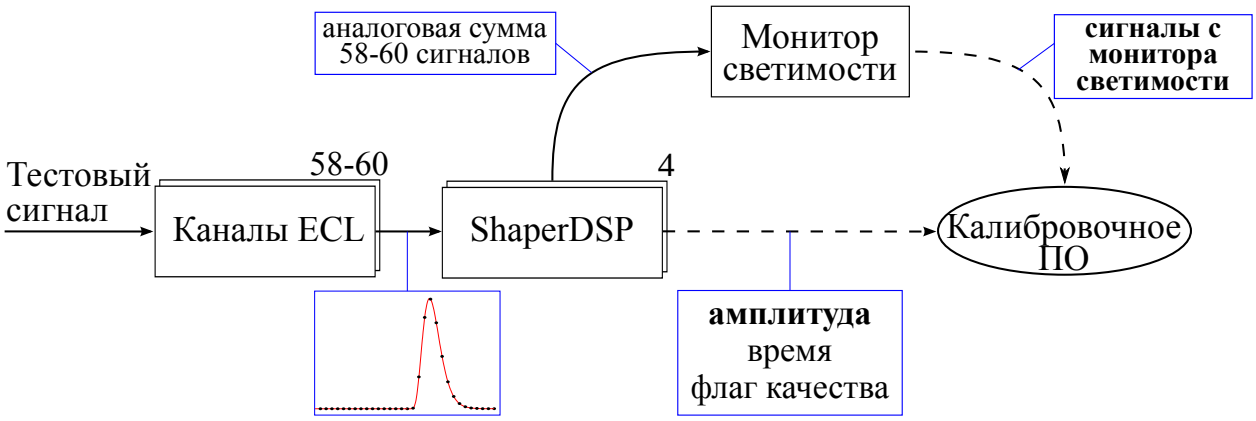

Рис. 15: Процесс энергетической калибровки монитора светимости.

В данной процедуре генерируется тестовый импульс для выбранного сектора. Тестовый импульс поступает на вход плат ShaperDSP, где происходит формирование и оцифровка сигнала. Сформированный и оцифрованный сигнал отправляется в модуль ECLCollector, где упаковываются данные и передаются в систему сбора данных (DAQ) на платы COPPER. Также с платы ShaperDSP сигнал, пройдя этап формирования, но не оцифрованный, отправляется в монитор светимости. Монитор светимости производит аналоговое суммирование сигнала с нескольких плат ShaperDSP и затем оцифровывает сигнал. Таким образом, сохранённые данные содержат сиг-

нал с монитора светимости в каналах АЦП и сигнал в системе сбора данных DAQ в энергетических единицах. В результате, калибровочный коэффициент для заданного сектора определяется по формуле:

$$\eta = \frac{\sum_i E_{DAQ}}{A_{LOM}} \frac{[\text{МэВ}]}{[\text{каналы АЦП}]} \tag{3}$$

Данная процедура проводится для каждого сектора.

При проведении энергетической калибровки в ручном режиме существует два неудобства: отсутствие прямого доступа к платам ShaperDSP и необходимость помнить соотношение номера сектора и номера плат ShaperDSP. Для упрощения данного процесса был разработан скрипт, который автоматически выставляет необходимую конфигурацию тестового импульса по заданным наборам секторов.

Всего калориметр содержит 8736 CsI кристаллов, сигнал с которых поступает на фотодиод и они объединены в триггерные ячейки. Подробнее объединение в триггерные ячейки описано в п. 2.2. Сигнал с двух триггерных ячеек попадает в плату ShaperDSP, которых всего 576. Далее данные с ShaperDSP отправляются в ECLCollector, которых 52, таким образом ECLCollector обрабатывает данные с 8-12 плат ShaperDSP. Далее ECLCollector упаковывает данные и отправляет их в модуль COPPER, к одному COPPER подключено сразу 2 ECLCollector (slot a и slot b).

На вход скрипту поступает набор секторов, для которых будет проводится энергетическая калибровка. Результатом работы скрипта является корректно выставленная конфигурация тестового импульса в соответствующих платах ShaperDSP.

Скрипт выполняется с модуля COPPER. Чтобы сконфигурировать тестовый импульс, требуется записать значение в определенные регистры ShaperDSP, поскольку прямого доступа к регистрам ShaperDSP нет, то требуется осуществлять запись через ECLCollector. В модуле ECLCollector реализован специальный протокол для доступа к регистрам ShaperDSP.

Сначала скрипт определяет к какой плате COPPER и какому из двух, подключенных к COPPER модулей ECLCollector, соответствует заданный сектор. Далее определяется на какие платы ShaperDSP и в какой регистр требуется установить тестовый импульс. Затем требуется осуществить запись рассчитанных значения регистров в ShaperDSP через модули ECLCollector.

Запись осуществляется при помощи системы медленного контроля NSM2 и специального протокола отправки команд на ShaperDSP, реализованного в программе ЦПУ ECLCollector. Чтобы обеспечить наилучшую скорость установки параметров, запросы к модулям ECLCollector выполняются асинхронно. После отправки NSM-сообщений, скрипт ожидает ответа от всех модулей, проверяет корректность ответных сообщений и выводит результат операции пользователю.

5.2 Мониторинг качества данных

В процессе работы коллайдера важно быстро обнаруживать и исправлять возникающие ошибки. Для данной цели непрерывно работают дежурные, которые отслеживают корректность работы всех систем эксперимента Belle II. При возникновении неисправностей или ситуаций, которые потенциально могу приводить к проблемам, требуется оперативно проинформировать ответственных дежурных.

Написан набор фоновых процессов, которые отслеживают стабильность работы: потребление памяти различных процессов, считывающих данные с LOM. Также реализованы скрипты, отслеживающие качество данных: для этого регулярно проверяется, что значения EPICS PV с монитора PV лежат в заранее определённом интервале допустимых значений.

В случае обнаружения ошибки с качеством данных, процесс посылает уведомление на почту ответственным лицам о том, что необходимо исправить ошибки. Если обнаруживается процесс, который начинает потреблять большое количество памяти, он будет принудительно перезапущен.

6 Анализ данных монитора светимости

В качестве дополнительной проверки корректности работы монитора светимости было проведено сравнение экспериментальных данных с данными, полученными в результате моделирования.

Сравнение проводилось по гистограммам энерговыделения в каждом секторе и в торцевых частях электромагнитного калориметра. Гистограммы энерговыделения для каждого сектора записываются в файл при работе считывающего ПО. Для расчета энерговыделения в торцевых частях ECL потребовалось написать программу, которая по записанным формам сигнала определяет события e^+e^- рассеяния. Программа затем определяет выделившуюся энергию, переводит значение в энергетические единицы и записывает данные в гистограмму. Далее значения гистограммы записываются в базу данных SQLite для последующего сравнения с моделированием. Моделирование осуществлялось методом Монте-Карло. Результаты сравнения энерговыделения в торцевых частях представлены на рисунке 16, а сравнение по секторам на рисунке 17.

Экспериментальные данные хорошо согласуются с моделированием, небольшое расхождение может быть объяснено трудностью моделирования фоновых событий.

Также в качестве дополнительной проверки, значения мгновенной светимости с монитора светимости сравнивались с другими работающими мониторами светимости участвующими в эксперименте. Для последующей проверки данных светимости были реализованы скрипты, которые позволяют сравнивать данные с разных мониторов светимости за заданные промежутки времени. Скрипты позволяют построить график светимости с различных мониторов, также построить отношение светимостей в зависимости от времени. Также, используя полученные данные, легко построить зависимость значений одного монитора светимости от другого и аппрокси-

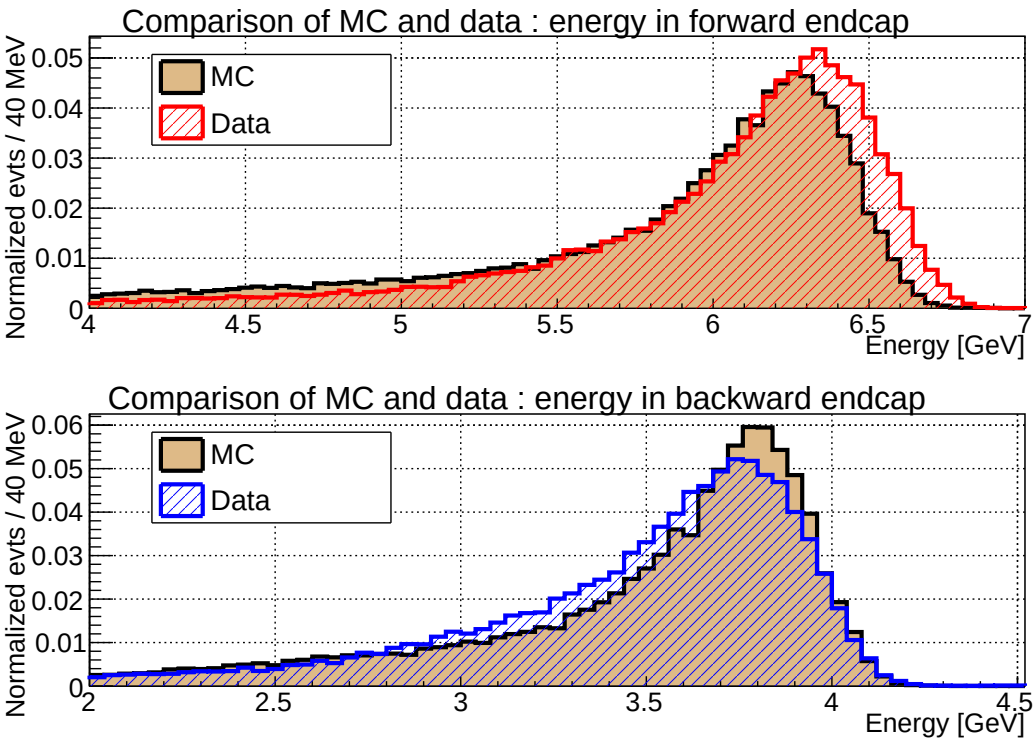

Рис. 16: Гистограммы энерговыделения в торцевых частях электромагнитного калориметра, полученные в результате моделирования и из экспериментальных данных.

мировать её линейной функцией, чтобы определить различие в значениях. Имеется возможность усреднять значения за выбранный интервал времени. Данные для скрипта берутся из root файлов.

Такая перекрестная проверка значений помогла обнаружить сбой в работе одного из мониторов светимости, за счёт чего эксперты успели исправить проблему в конфигурации до возобновления набора данных.

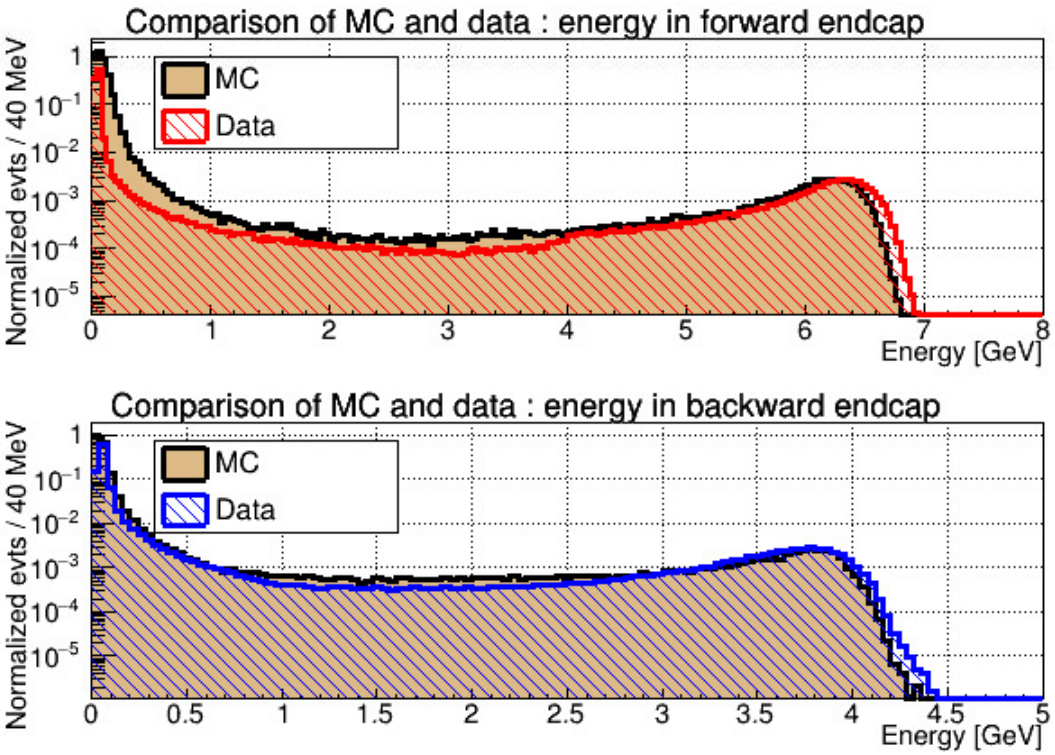

Рис. 17: Гистограммы энерговыделения в одном секторе из передней и задней торцевой части электромагнитного калориметра.

7 Графический интерфейс пользователя

Для отображения и удобного анализа данных, приходящих с монитора светимости, используется графический интерфейс онлайн монитора светимости. В рамках расширения API считывающего ПО был добавлен метод для получения коэффициентов энергетической калибровки. Поэтому были реализованы получение и распаковка коэффициентов энергетической калибровки. Затем, используя полученные коэффициенты, был добавлен код для точного пересчета гистограмм и форм сигналов из каналов АЦП в энергетические единицы.

Помимо информации на текущий момент времени, для детального анализа прошедших событий, графический интерфейс периодически сохраняет считанные формы сигналов с каждого сектора. Сохраненные формы сигналов затем становятся доступны на веб-странице дежуного (рис. 18).

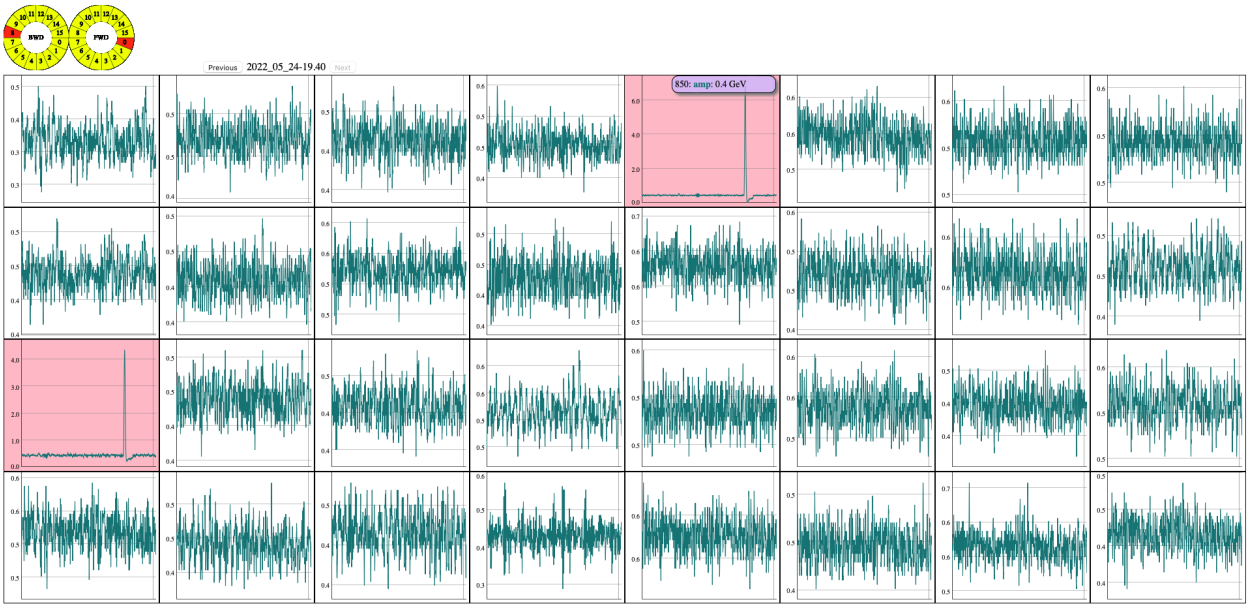

Рис. 18: Отображение ранее записанных форм сигналов на странице дежурного.

В классе, ответственном за взаимодействие с API монитора светимости, добавлена возможность десериализации чисел с плавающей точкой в

полученном ответе. Далее, при получении гистограмм и форм сигналов, отправляется запрос на получение калибровочных коэффициентов. Используя полученные коэффициенты, производится перевод гистограмм и форм сигналов из каналов АЦП в энергетические единицы.

Улучшена отрисовка гистограм, максимальное значение по оси X динамически определяется по массиву, содержащему гистограммы в энергетических единицах. Также добавлен логарифмический масштаб по оси Y.

8 Заключение

В ходе данной работы было улучшено ПО для онлайн монитора светимости. Модернизирована система сбора данных с монитора светимости:

- Реализовано считывание данных с частотой 1 Гц и оптимизирована работа с базой данных.
- Расширен набор интегральных светимостей. Произведен рефакторинг для упрощения добавления новых переменных по требованию экспертов.
- Расширено API монитора светимости.
- Написан набор автоматических тестов, позволяющих тестировать расчет, чтение и запись значений светимостей в систему медленного контроля EPICS.
- Реализована версия программы ЦПУ, которая способна обрабатывать множество одновременных подключений.
- Автоматизирован процесс энергетической калибровки онлайн монитора светимости.
- Реализована первая версия микросервисной архитектуры считывающего ПО.
- Реализован набор скриптов, позволяющих сравнивать значения светимостей с другими мониторами светимости, работающими в эксперименте.

Также проведено офлайн сравнение данных моделирования с экспериментальными данными, полученными с монитора светимости. Данные хорошо согласуются. Улучшено отображение данных в графическом интерфейсе монитора светимости. Добавлены дополнительные методы для синхронизации

параметров со считывающим ПО. Реализовано сохранение и отображение форм сигналов с монитора светимости на веб-странице дежурного. Написан набор скриптов для мониторинга потребления памяти, оповещения о значениях светимости, выходящих за заданные пределы.

Описанное в данной работе программное обеспечение активно используется в эксперименте Belle II с начала 2018 года.

Список литературы

1. *Belle II collaboration.* Belle II overview. — https://www.belle2.org.
2. The Belle II Physics Book / E. Kou [и др.] // Progress of Theoretical and Experimental Physics. — 2019. — Дек. — Т. 2019, № 12. — URL: http://dx.doi.org/10.1093/ptep/ptz106.
3. *al. A. A. et.* The Belle Detector // Nucl.Instrum.Meth. — 2002. — Т. A, № 479. — С. 117—232. — URL: http://www-f9.ijs.si/~krizan/belle/detect%5C_paper.pdf.
4. *Abe T., et al.* Belle II Technical Design Report. — 2010.
5. *Xilinx.* Spartan-3 FPGA Family Data Sheet. — https://docs.xilinx.com/v/u/en-US/ds099.
6. *Analog Devices I.* AD7641. — https://www.analog.com/media/en/technical-documentation/data-sheets/AD7641.pdf.
7. Early Phase 2 Results of LumiBelle2 for the SuperKEKB Electron Ring / S. Di Carlo [и др.] // J. Phys. Conf. Ser. — 2018. — Т. 1067, № 7. — С. 072025.
8. A fast luminosity monitor based on diamond detectors for the SuperKEKB collider / C. Pang [и др.] // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2019. — Т. 931. — С. 225—235. — URL: https://www.sciencedirect.com/science/article/pii/S0168900219304012.
9. *ZDLM Overview.* ZDLM monitor. — https://indico.in2p3.fr/event/10971/contributions/3821/attachments/2893/3516/LAL_15b1b.pdf.
10. *Xilinx.* Xilinx Spartan-6 DataSheet. — https://docs.xilinx.com/v/u/en-US/ds160.
11. *Nakao M., Suzuki S. Y.* Network shared memory framework for the Belle data acquisition control system. — 1999.
12. EPICS overview. — https://epics-controls.org/about-epics/.
13. *Dalesio L. R., Kraimer M. R., Kozubal A. J.* EPICS Architecture. — 1991.
14. *Xilinx.* MicroBlaze Soft Processor Core Overview. — https://www.xilinx.com/products/design-tools/microblaze.html#overview.
15. *Dunkels A.* Design and Implementation of the lwIP. — 2001. — Февр.
16. *M. A.* PythonIoc. — https://github.com/Araneidae/pythonIoc.
17. About SQlite. — https://www.sqlite.org/about.html.

18. Multiple Execution Contexts in lwIP code. — https://www.nongnu.org/lwip/2_1_x/pitfalls.html.

19. The EPICS Archiver Appliance. — http://slacmshankar.github.io/epicsarchiver_docs/details.html.

20. Python pytest framework. — https://docs.pytest.org/en/7.1.x/.

21. ZeroMQ framework. — https://zeromq.org.