ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
УЧРЕЖДЕНИЕ НАУКИ
ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ ИМ. Г.И. БУДКЕРА
СИБИРСКОГО ОТДЕЛЕНИЯ РОССИЙСКОЙ АКАДЕМИИ НАУК

На правах рукописи

РЕМНЕВ МИХАИЛ АНАТОЛЬЕВИЧ

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМЫ СБОРА ДАННЫХ ЭЛЕКТРОМАГНИТНОГО КАЛОРИМЕТРА ДЕТЕКТОРА BELLE II

1.3.2. Приборы и методы экспериментальной физики

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор физико-математических наук
Кузьмин Александр Степанович

Новосибирск — 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования

В изучении свойств и поведения B и D-мезонов важную роль играют B-фабрики — электрон-позитронные коллайдеры, спроектированные для рождения большого количества B-мезонов, позволяющие изучать их редкие распады. В частности, такими B-фабриками являются e^+e^- коллайдер SuperKEKB и его предшественник KEKB [1, 2]. Эксперимент Belle, проводившийся на коллайдере KEKB, завершился в 2010 году, достигнув рекордной светимости, равной $2{,}11 \cdot 10^{34}$ см$^{-2}$с$^{-1}$. Однако, чтобы улучшить точность уже измеренных по данным Belle физических величин и исследовать редкие физические процессы, возникла необходимость в усовершенствовании коллайдера KEKB с целью повышения его светимости. Усовершенствование KEKB позволит повысить прежде достигнутую светимость в 40 раз за счёт использования схемы нанопучков. Это увеличение светимости приводит к увеличению уровня фона приблизительно в 20 раз [3] и в результате налагает новые требования на конструкцию систем, на их электронику, а также на программное обеспечение, использующееся для сбора данных в эксперименте Belle II [4], продолжающем эксперимент Belle [5].

Эксперимент Belle II начался в 2018 году. Цель эксперимента — набрать 50 абн$^{-1}$ данных для исследования физики B- и D-мезонов, τ-лептонов, нарушения CP-симметрии, определения параметров CKM-матрицы и поиска Новой физики. В 2019 году эксперимент перешёл к третьей фазе (Phase III) — регистрации событий в столкновении e^+e^- пучка с использованием всех подсистем детектора. В этой фазе важно иметь готовую систему сбора данных (DAQ, Data AcQuisition), которая позволит с минимальными задержками проводить чтение данных со всех систем детектора с частотой до 30 кГц, осуществлять запуск заходов, проводить мониторирование и калибровку систем детектора.

Электромагнитный калориметр (ECL, Electromagnetic CaLorimeter), на котором фокусируется данная работа, является важной системой детектора Belle II, предназначенной для измерения энергии зарегистрированных частиц. Считывающая электроника ECL была усовершенствована для работы в условиях увеличенного уровня фона, поэтому возникла необходимость разработки нового программного обеспечения, которое будет осуществлять инициализацию и управление электроникой ECL DAQ. Кроме того, поскольку эксперимент Belle II перешёл к другому фреймворку медленного контроля и отказался от

большинства программных средств, использовавшихся в эксперименте Belle, в пользу новых решений, возникла необходимость разработать заново или существенно переписать многое высокоуровневое программное обеспечение (ПО), использующееся в ECL DAQ. Разработанное ПО для ECL DAQ позволяет обеспечивать стабильную работу калориметра в ходе эксперимента, уменьшая время, необходимое дежурному на обнаружение и устранение потенциальных проблем, влияющих на качество записываемых данных.

Степень разработанности темы

Можно выделить три класса задач, которые решает программное обеспечение для системы сбора данных. Технические задачи относятся к проблемам быстродействия и масштабируемости системы на всех уровнях её работы. Задачи управления включают в себя управление конфигурациями системы, осуществление мониторинга качества данных и оперативное обнаружение сбоев DAQ. Интеграционные задачи подразумевают определение протоколов, использующихся для объединения отдельных узлов системы.

Одной из важных технических задач является разработка ПО для триггера высокого уровня. По сравнению с системой Belle DAQ, частота триггера первого уровня в системе Belle II DAQ выросла от 0,5 кГц до 30 кГц. По этой причине было решено использовать программный триггер высокого уровня для реконструкции и дополнительного отбора событий. Один из первых прототипов триггера высокого уровня был разработан в 2003 году для детектора ATLAS. На данный момент существует несколько реализаций триггера высокого уровня, однако только в Belle II используется единый фреймворк для обработки данных, который применяется как в триггере высокого уровня, так и при последующей детальной обработке сохранённых данных. Это позволяет лучше организовать процесс разработки, но приводит к дополнительным техническим требованиям на быстродействие программного обеспечения и более широкую оптимизацию под набор различных сценариев использования.

Для решения задач управления в Belle II в большинстве случаев используется фреймворк Control System Studio (CSS). С 2013 года CSS активно применяется в физике высоких энергий, представляя единый набор инструментов для мониторинга и управления детектором. Однако, как и для любых других фреймворков, ориентированных на построение графических интерфейсов пользователя, для CSS затруднена реализация процедур автоматического тестирования и интеграции с веб-интерфейсами, что является важной задачей в

эксперименте Belle II.

В рамках задач интеграции, распространённым решением для объединения приложений в сети сбора данных является система медленного контроля Experimental Physics and Industrial Control System (EPICS), широко используемая в физических экспериментах и постоянно развивающаяся с 1994 года. Модули, разработанные при помощи EPICS, способны обмениваться информацией, существуют готовые решения для архивирования данных мониторинга и для реализации систем оповещения о сбоях в системе сбора данных. Тем не менее в EPICS невозможно получить полный список запущенных модулей, что может затруднять администрирование системы. Поэтому для эксперимента Belle II используется новый фреймворк Network Shared Memory 2, который, в рамках данной работы, было необходимо расширить для решения существующих интеграционных задач.

Цели и задачи диссертационной работы

Целью данной работы является разработка программного обеспечения для системы сбора данных электромагнитного калориметра, в частности систем медленного контроля и мониторинга.

Задачи данной работы:

1. Разработка ПО для взаимодействия с электроникой калориметра, автоматизирующего следующие процессы:
   - Управление конфигурациями электроники.
   - Инициализация электроники и её диагностика.
2. Разработка модулей системы медленного контроля.
3. Создание системы мониторинга качества данных ECL в режиме реального времени.
4. Разработка системы DAQ для монитора светимости, являющегося отдельным модулем ECL.
5. Разработка инструментов для более детальной диагностики качества данных ECL по сохранённой информации.

Научная новизна

Детектор Belle II является уникальной установкой, нацеленной на проведение передовых исследований в области физики высоких энергий. Автором создано уникальное программное обеспечение, позволяющее инициализировать

считывающую электронику, эффективно организовывать процесс сбора данных с электромагнитного калориметра, автоматически обнаруживать и исправлять возможные сбои на всех уровнях работы системы, анализировать долговременную стабильность электроники сбора данных.

Ряд новых программных решений, которые было решено использовать в эксперименте, требует пересмотра всего существующего стека технологий, использующихся в DAQ. Система медленного контроля эксперимента Belle II использует новый фреймворк Network Shared Memory 2 (NSM2), разработанный в KEK. Кроме того, в эксперименте используются две базы данных (БД), БД калибровочных констант и БД конфигураций электроники. Они имеют существенно различающуюся внутреннюю структуру, поэтому их синхронизация, необходимая при разработке программного обеспечения для калориметра, является нетривиальной задачей. Для учёта этих требований были разработаны программные решения и модули, которые лучше сочетаются с новыми парадигмами построения ПО сбора данных. Данная работа рассматривает современные методики программирования в применении к задачам автоматизации физического эксперимента.

Впервые был рассмотрен процесс разработки систем медленного контроля и мониторинга в контексте методов инкрементного прототипирования. В частности, разработан модуль для экспорта библиотеки C++ в специально разработанный интерпретатор, что позволяет легко интегрироваться с другими запущенными процессами системы сбора данных.

В рамках данной работы автором применены методики быстрого прототипирования приложений в контексте разработки для фреймворка NSM2, реализованного на языке C. Чтобы ускорить процесс разработки, были рассмотрены различные методы реализации интерфейса Python к NSM2, представлены преимущества и недостатки возможных решений. Разработана библиотека pyNSM2, использующая программный модуль ctypes для быстрого доступа к функциям и структурам языка C. Разработаны инструменты, поддерживающие реализацию pyNSM2 в полном соответствии с основной реализацией NSM2.

Впервые разработана система сборки, позволяющая автоматически интегрировать независимо разрабатываемые на разных языках программирования библиотеки NSM2 и pyNSM2.

Теоретическая и практическая значимость

Разработанное программное обеспечение внедрено в систему сбора дан-

ных эксперимента Belle II и обеспечивает стабильный сбор и контроль данных с электроники электромагнитного калориметра и монитора светимости. Таким образом, созданное ПО позволяет считывать критически необходимые в физических исследованиях эксперимента Belle II данные, а также осуществлять мониторинг работы коллайдера SuperKEKB. Созданное ПО является одним из необходимых компонентов для проведения эксперимента Belle II. Суммарный объём конечного кода составляет около 45000 строк.

Разработанные библиотеки могут использоваться в других системах сбора данных, позволяя с минимальными временными затратами реализовывать системы мониторинга и медленного контроля.

Методология и методы исследования

В работе использованы методологии Rapid Application Development и Test-Driven Development. Программное обеспечение разрабатывалось на языках программирования C, C++, Python, Jython, PHP, JavaScript. Для обработки данных использовался фреймворк CERN ROOT и Belle II Analysis Software Framework. Для интеграции процессов медленного контроля, относящихся к ECL, с глобальной системой сбора данных использовалась разработанная для эксперимента Belle II программная библиотека daq_slc. В ПО системы сбора данных широко использовались программные пакеты EPICS, NSM2, Control System Studio, Qt 5, система управления базами данных SQLite.

Основные положения, выносимые на защиту

1. Разработана система инициализации и управления конфигурациями электроники калориметра.
2. Новый программный модуль языка Python предоставляет доступ к функциям системы медленного контроля NSM2 и широко используется в эксперименте Belle II, существенно ускоряя разработку приложений для системы сбора данных.
3. Создан веб-сервер ECL DAQ, который обеспечивает стабильную работу считывающей электроники калориметра, предоставляя централизованный доступ ко всем реализованным инструментам мониторинга качества данных.
4. Разработана независимая система сбора данных с онлайн-монитора светимости, сопутствующие модули мониторинга и калибровки.

Степень достоверности и апробация результатов

Разработанное программное обеспечение используется в эксперименте Belle II и позволяет обеспечивать стабильный сбор данных с электроники электромагнитного калориметра. Результаты данной работы были представлены на внутренних семинарах Belle II и конкурсах молодых учёных ИЯФ СО РАН, а также на следующих международных конференциях:

55-ая международная студенческая конференция, (МНСК 2017, Новосибирск, Россия, 16–20 апреля 2017 г.);

Instrumentation for Colliding Beam Physics (INSTR-20, BINP, Novosibirsk, Russia, 24–28 February 2020).

О результатах работ по тематике заявленной диссертации автор неоднократно докладывал на конкурсе молодых учёных ИЯФ СО РАН.

Публикации

Основные результаты диссертации представлены в пяти публикациях, из них пять в научных изданиях, рекомендуемых ВАК при Минобрнауки России [6–10].

Личный вклад соискателя

Изложенные в работе результаты получены автором лично либо при его определяющем вкладе. Автор принимал активное участие в проведении экспериментальных исследований и организации работы дежурных экспертов по ECL.

Автором было разработано программное обеспечение для работы с электроникой калориметра, синхронизирующее параметры электроники с базами данных, использующимися в эксперименте Belle II. Это позволило, с одной стороны, использовать актуальные параметры электроники в моделировании и, с другой стороны, быстро обновлять амплитудные пороги, используя новые калибровочные коэффициенты [6]. В рамках данной работы автор также расширил систему мониторинга качества данных, разработал программное обеспечение для инициализации электроники и управления системой сбора данных [7, 10]. Автором был реализован ряд модулей для быстрой интеграции с системой медленного контроля, которые используются в глобальной системе сбора данных эксперимента Belle II [9]. Также была разработана независимая система сбора данных с онлайн-монитора светимости.

Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Подготовка

полученных результатов к публикации была проведена совместно с соавторами. Все представленные в диссертации результаты получены лично автором, либо при его участии.

Структура и объём диссертации

Данная работа разделена на введение, 8 глав и заключение. Полный объём диссертации составляет **105** страниц, включая **42** рисунка и **1** таблицу. Список литературы содержит **117** наименований.

ГЛАВА 1. ЭКСПЕРИМЕНТ BELLE II

1.1 Коллайдер SuperKEKB

SuperKEKB представляет собой модернизацию асимметричного электрон-позитронного коллайдера KEKB в городе Цукуба, Япония. KEKB и работавший на нём детектор Belle относятся к одной из двух построенных B-фабрик, установок для изучения распадов B-мезонов, рождающихся в области энергии $\Upsilon(4S)$-резонанса. Данные, собранные на B-фабриках, используются для исследования нарушения CP-симметрии, определения параметров CKM-матрицы и поиска новой физики. Во всех существовавших и планирующихся B-фабриках используются асимметричные электрон-позитронные коллайдеры, для того, чтобы система B- анти-B-мезонов имела заметный лоренцевский буст (для эксперимента Belle II $\beta\gamma = 0{,}28$ [4, с. 139]). Это позволяет наблюдать эволюцию во времени нейтральных B-мезонов [11], а также измерять их время распада, основываясь на расстоянии, пройденном от точки столкновения пучков.

Устройство коллайдера SuperKEKB показано на рисунке 1. Электроны и позитроны накапливаются в кольцах high-energy ring (HER) и low-energy ring (LER), соответственно. Каждое кольцо состоит из 4 дуг и 4 прямых секций. Суммарная длина каждого из колец составляет 3016 м.

Для повышения светимости SuperKEKB осуществляется переход на схему нанопучков, впервые предложенную в проекте SuperB factory [12, 13].

Переход к схеме нанопучков приводит к росту эмиттанса из-за внутрипучкового рассеяния и к сокращению времени жизни пучка, обусловленному эффектом Тушека. Чтобы уменьшить эмиттанс, при переходе от KEKB к SuperKEKB энергия электронного пучка была снижена с 8 ГэВ до 7 ГэВ. В то же время, энергия позитронного пучка была повышена с 3,5 ГэВ до 4 ГэВ, чтобы ослабить влияние эффекта Тушека.

1.2 Детектор Belle II

Все компоненты детектора Belle II были модифицированы для работы в условиях увеличенного пучкового фона. Кроме того, для повышения точности собираемых данных, по сравнению с детектором Belle в детектор Belle II были добавлены новые подсистемы. Как показано на рисунке 1, детектор Belle II

получает данные с 7 подсистем.

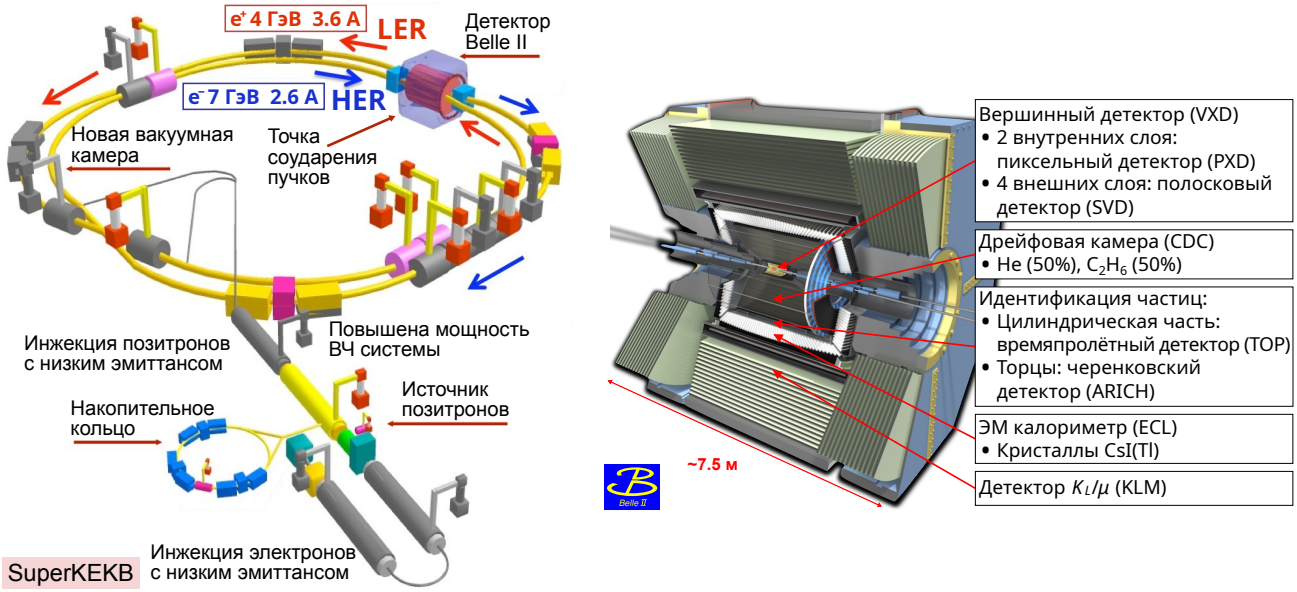

Рисунок 1 — Ускорительный комплекс SuperKEKB (слева) и детектор Belle II (справа)

Точки распада B-мезонов определяются двумя внутренними слоями кремниевого пиксельного детектора (PXD, PiXel Detector) и четырьмя слоями кремниевого вершинного детектора (SVD, Silicon Vertex Detector), которые расположены внутри цилиндрической беррилиевой трубки. Определение траекторий и скорости заряженных частиц осуществляется проволочной центральной дрейфовой камерой (CDC, Central Drift Chamber). Тип частицы устанавливается при помощи измерений dE/dx в CDC, а также данных, получаемых с времяпролётного (TOP, Time-Of-Propagation) счётчика в цилиндрической части детектора и регистратора колец от черенковского излучения (ARICH, Aerogel Ring Imaging Cherenkov) в торцевой части. Электромагнитный ливень поглощается в кристаллах CsI(Tl) калориметра (ECL, Electromagnetic CaLorimeter), расположенных внутри кольца соленоида. Мюоны и K_L мезоны детектируются массивами резистивных плоскопараллельных счётчиков (RPC, Resistive Plate Counters). Детектор покрывает полярный угол $17° < \theta < 150°$.

Обновления подсистем детектора Belle II позволят достичь более высокого координатного разрешения, повысить эффективность реконструкции K_S-распадов на два заряженных пиона, улучшить разделение пионов и каонов, а также уменьшить фон, возникающий при временном наложении двух сигналов.

Разумеется, также значительные изменения претерпели системы сбора дан-

ных с каждой из подсистем детектора. В частности, была модернизована практически вся электроника, осуществляющая чтение данных с ECL.

1.3 Электромагнитный калориметр

1.3.1 Описание калориметра

Среди распадов B-мезонов, треть всех частиц составляют π^0, распадающиеся на два гамма-кванта. Кроме π^0, рождаются и другие частицы, имеющие фотоны в конечном состоянии в широком диапазоне энергий от 20 МэВ до 4 ГэВ [4]. Поэтому, для целей эксперимента Belle II, крайне важен электромагнитный калориметр (ECL, Electromagnetic CaLorimeter), обладающий высоким энергетическим и координатным разрешением.

ECL детектора Belle II состоит из цилиндрической и двух торцевых частей. Цилиндрическая часть имеет длину 3 м и внутренний радиус 1,25 м. Расстояние от точки взаимодействия для переднего и заднего торцов составляет $z_1 = 1{,}96$ м и $z_2 = -1{,}02$ м, соответственно. Калориметр работает в магнитном поле с величиной индукции 1,5 Тл.

Как показано на рисунке 2, ECL состоит из 8736 кристаллов CsI(Tl) с суммарной массой примерно 43 тонны. Каждый кристалл имеет форму усечённой пирамиды, вершина которой направлена к окрестности точки взаимодействия. Отклонения от точки взаимодействия по ϕ и θ не превышают 1,3° в цилиндрической части, 4,3° в торцах и предназначены для смягчения влияния щелей между кристаллами. В цилиндрической части расположены 6624 кристалла, остальные 2112 находятся в торцах.

В цилиндрической части ECL размеры большинства кристаллов составляют 55 мм × 55 мм для передней поверхности кристалла и 70 мм × 70 мм для задней. В торцевых частях, ширина стороны передней поверхности кристалла меняется от 44,5 мм до 70,8 мм, задней от 54 мм до 82 мм. Длина кристаллов составляет 30 см ($16{,}2\,X_0$), что позволяет избежать ухудшения энергетического разрешения, связанного с флуктуациями поглощения энергии ливня [5].

ECL покрывает полярный угол $12{,}4° < \theta < 155{,}1°$, за исключением промежутков шириной в ~1° на линиях стыка между цилиндрической и торцевыми частями калориметра. В результате осуществляется покрытие телесного угла в $0{,}91 \cdot 4\pi$.

Основными задачами ECL являются:

- Высокоэффективное детектирование фотонов.
- Точное определение энергии и угловых координат фотонов.
- Идентификация электронов.
- Обеспечение информация для системы триггера.
- Измерение светимости в режиме реального времени.
- Детектирование и идентификация K_L^0, выполняющееся совместно с детектором долгоживущих K-мезонов и мюонов (KLM).

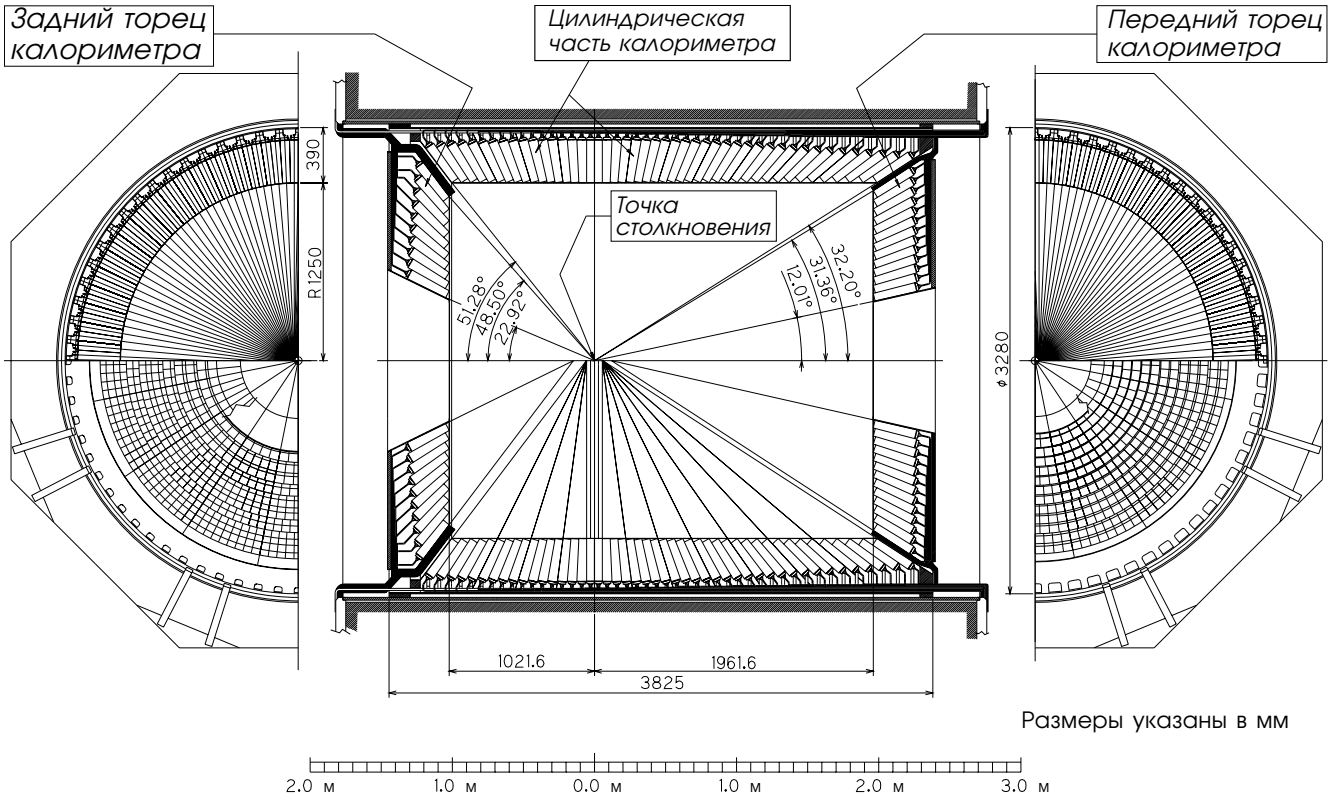

Рисунок 2 — Структура электромагнитного калориметра

1.3.2 Считывающая электроника ECL

Блок-схема электроники для считывания данных с калориметра в систему сбора данных (DAQ, Data AcQuisition) Belle II представлена на рисунке 3. В калориметре используются те же фотодиоды и зарядочувствительные предусилители (CSP, Charge Sensitive Preamplifier) [14], что и в эксперименте Belle

[15]. Каждый счётчик калориметра представляет собой сцинтиляционный кристалл CsI(Tl), свет с которого считывается двумя PIN фотодиодами Hamamatsu S2744-08 площадью 1×2 cm^2. Сигналы от фотодиодов регистрируются двумя CSP, расположенными на задней стороне счётчика.

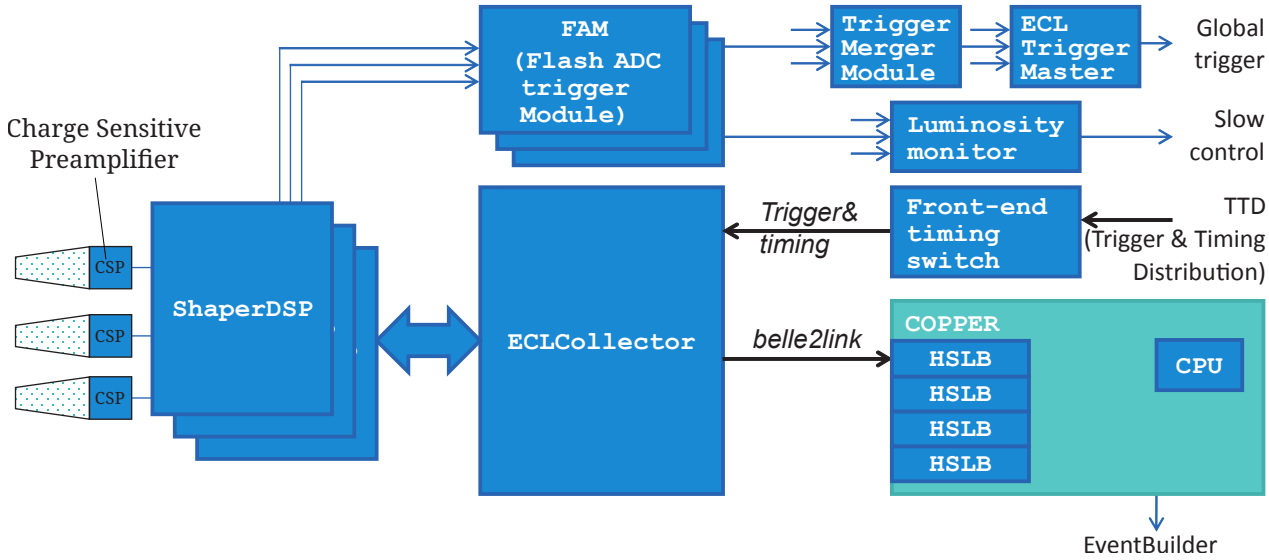

Рисунок 3 — Схема системы сбора данных с электромагнитного калориметра [10]

Далее сигналы от счётчиков поступают в модули усилителя-формирователя (ShaperDSP). Каждый ShaperDSP имеет 16 независимых каналов, в каждом канале происходит суммирование сигнала от двух предусилителей, формировка сигнала с временем достижения пика ~2,5 мкс и непрерывная оцифровка с частотой дискретизации 1,7 МГц. После получения сигнала триггера, данные АЦП передаются в программируемую логическую интегральную схему (ПЛИС) Spartan 3, в которой подгоняется оцифрованная форма импульса, вычисляется амплитуда, время и флаг качества данных с использованием алгоритма минимизации χ^2. В особых случаях, описанных в разделе 1.3.5, модуль ShaperDSP может также добавлять сырые данные АЦП (31 точку формы сигнала) к пакету отправляемых данных. Эта информация используется для идентификации частиц на стадии офлайн анализа [6], а также для мониторирования работы логики ShaperDSP и для данных со случайным запуском, которые используются для моделирования фона.

Кроме того, модули ShaperDSP передают аналоговую сумму 8–16 входных сигналов с быстрым временем формирования (время достижения пика 0,4 мкс)

в триггерный модуль оцифровки (FAM, Flash ADC trigger module). Эти быстрые сигналы используются, чтобы формировать решение нейтрального триггера. Амплитуда каждого сигнала умножается на конфигурируемый коэффициент в интервале [0,5, 1], называемый коэффициентом аттенюации. Эти коэффициенты калибруются специальной процедурой, чтобы выровнять цену каналов в энергетических единицах. Быстрые сигналы с торцов калориметра также используются для измерения светимости в режиме реального времени [16], как описано в разделе 1.4.

ShaperDSP — это модули VME формата 9U, которые установлены в 52 крейтах VME. Каждый крейт содержит от 8 до 12 таких модулей и один модуль коллектора (ECLCollector) формата 6U, который считывает данные с ShaperDSP.

В итоге, данные с модулей ECLCollector посылаются в 52 платы High Speed Link Board (HSLB) с использованием протокола Belle2Link [17]. HSLB являются частью интерфейсных модулей Common Pipelined Platforms for Electronics Readout (COPPER), где на каждый COPPER приходится два HSLB, то есть для считывания данных с ECL используется 26 модулей COPPER.

Также модули ECLCollector предоставляют возможность посылать калибровочные сигналы, генерируемые быстрым ЦАП. Эти сигналы близки к форме сигнала со счётчика CsI(Tl) и могут использоваться для калибровки электроники и проверки работы новых версий прошивки. Более детальное описание использующихся калибровочных процедур приведено в разделе 1.5.

1.3.3 Подгонка сигнала в ShaperDSP

Как описано в разделе 1.3.2, ShaperDSP выполняет оцифровку входного сигнала с периодом оцифровки $T_S = \frac{1}{1{,}7\ \text{МГц}}$. Далее, как показано на рисунке 4, этот сигнал подгоняется функцией

$$y(t) = A \cdot F(t - t_0) + P, \tag{1}$$

с параметрами подгонки A — амплитуда сигнала, t_0 — время начала сигнала, P — значение пьедестала АЦП. $F(t)$ — это функция отклика считывающей электроники ECL, которая описывается 9 заранее определёнными для каждого канала параметрами (используя калибровку формы сигнала, описанную в разделе 1.5). Для подгонки сигнала используется алгоритм минимизации χ^2,

который сходится за три итерации, при условии, что время начала сигнала лежит в интервале $\pm T_S$.

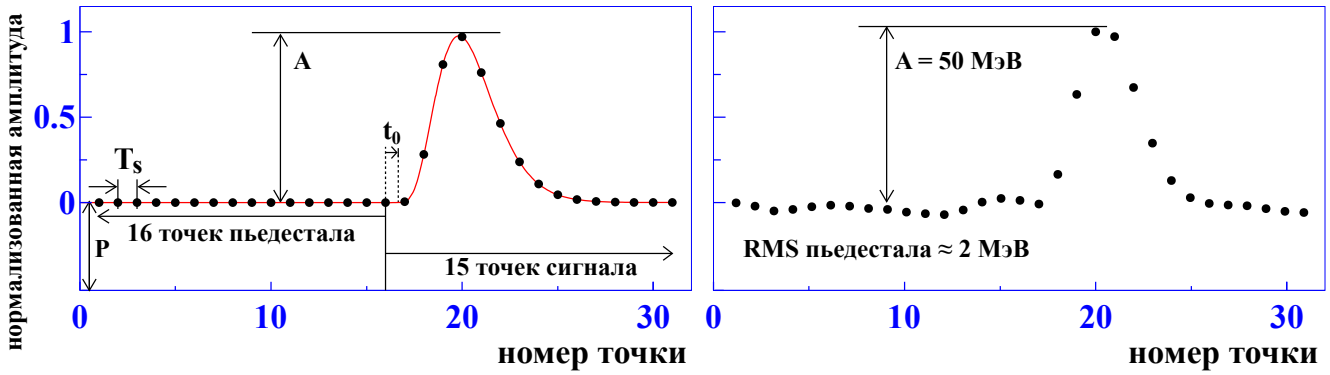

Рисунок 4 — Идеальная форма сигнала ShaperDSP (слева) и типичная форма сигнала из данных эксперимента (справа)

Чтобы обеспечить скорость обработки сигнала, достаточную для работы на проектной частоте триггера 30 кГц, алгоритм использует заранее вычисленные DSP коэффициенты, определённые из 16×192 значений табулированной функции F_i^k, 16×192 значений табулированной производной $F_i^{k\prime}$ и 16×16 коэффициентов ковариационной матрицы шумов S_{ij}:

$$F_i^k = F\left(i \cdot T_S + k \cdot \frac{T_S}{96}\right), i \in [0, 15], k \in [0, 192), \tag{2}$$

$$S_{ij} = \overline{(y_i - \overline{y_i})(y_j - \overline{y_j})}, i \in [0, 15], j \in [0, 15]. \tag{3}$$

Коэффициенты ковариационной матрицы шумов и, как следствие, DSP коэффициенты, обычно обновляются каждые два месяца, чтобы корректно учитывать изменения пучкового фона. Подробное описание процедуры вычисления DSP коэффициентов приведено в [18].

1.3.4 Амплитудные пороги

Помимо параметров функции $F(t)$ и коэффициентов DSP, конфигурация алгоритма подгонки также содержит 4 амплитудных порога, задаваемых отдельно для каждого канала. Эти пороги используются для того, чтобы пропустить обработку сигналов, имеющих незначительную амплитуду и сохранить дополнительные данные для сигналов с высокой амплитудой.

Порог **hit threshold** позволяет исключить данные фона ещё до начала процедуры подгонки сигнала. Оценка пьедестала АЦП $\widetilde{P} = (y[12] + y[13] +$

$y[14]+y[15])/4$ сравнивается с оценкой амплитуды зарегистрированного сигнала $\widetilde{A} = (y[20] + y[21])/2$. Если

$$\widetilde{A} - \widetilde{P} < \text{hit\_threshold}, \tag{4}$$

то ShaperDSP полностью пропускает алгоритм подгонки и отбрасывает данные АЦП.

Порог **low amplitude threshold** используется для определения случаев, когда должна использоваться упрощённая процедура подгонки. Для малых амплитуд, точность определения времени хуже, чем точность времени триггерного сигнала, поэтому время t_0 принимается равным триггерному времени t_{tr}, и подгонка сигнала продолжается с двумя плавающими параметрами (A и P).

На каждой итерации процедуры подгонки сигнала, амплитуда A сравнивается с порогом **skip threshold**. Если $A <$ skip_threshold, данные отбрасываются и дальнейшие итерации подгонки пропускаются.

По завершению процедуры подгонки, амплитуда сравнивается с порогом **ADC threshold**. Если A выше порога, то данные формы сигнала посылаются вместе с результатами подгонки.

В стандартной конфигурации, hit threshold равен "-10 МэВ", low amplitude threshold равен 2 МэВ, skip threshold равен 1 МэВ, ADC threshold равен 50 МэВ.

1.3.5 Сохранение данных формы сигнала

Существует три отдельных сценария, когда алгоритм подгонки, реализованный в модуле ShaperDSP, посылает данные формы сигнала.

Если амплитуда сигнала выше ADC threshold, то такие данные сохраняются и в дальнейшем используются в офлайн-анализе формы импульса для дискриминации между адронными и электромагнитными ливнями [6]. В случае событий случайного триггера, данные формы сигнала сохраняются для моделирования фона (эти данные подмешиваются к данным моделирования, чтобы более точно воспроизвести фоновые условия в эксперименте).

Также, для некоторой доли событий (на данный момент 1/1000) всегда сохраняются данные о форме сигнала. Они используются для проверки стабильности работы прошивки ShaperDSP с использованием процедуры, описанной в разделе 6.1.

1.4 Монитор светимости

Для измерения светимости по данным ECL в режиме реального времени, в Институте ядерной физики им. Г. И. Будкера СО РАН был разработан модуль Luminosity Online Monitor (LOM). Производство LOM было выполнено фирмой NOTICE (Республика Корея).

LOM использует быстрые сигналы с модулей ShaperDSP, относящихся к переднему и заднему торцам ECL. Каждый ShaperDSP обрабатывает сигналы с одной триггерной ячейки, области из 4×4 кристаллов CsI(Tl). Торцевая секция ECL разбита на 16 секторов, по четыре триггерные ячейки в каждом секторе задней секции и пять – в передней, как показано на рисунке 5. Внутренние триггерные ячейки в секторах передней секции не подключены к монитору светимости из-за высоких фоновых загрузок в этих ячейках.

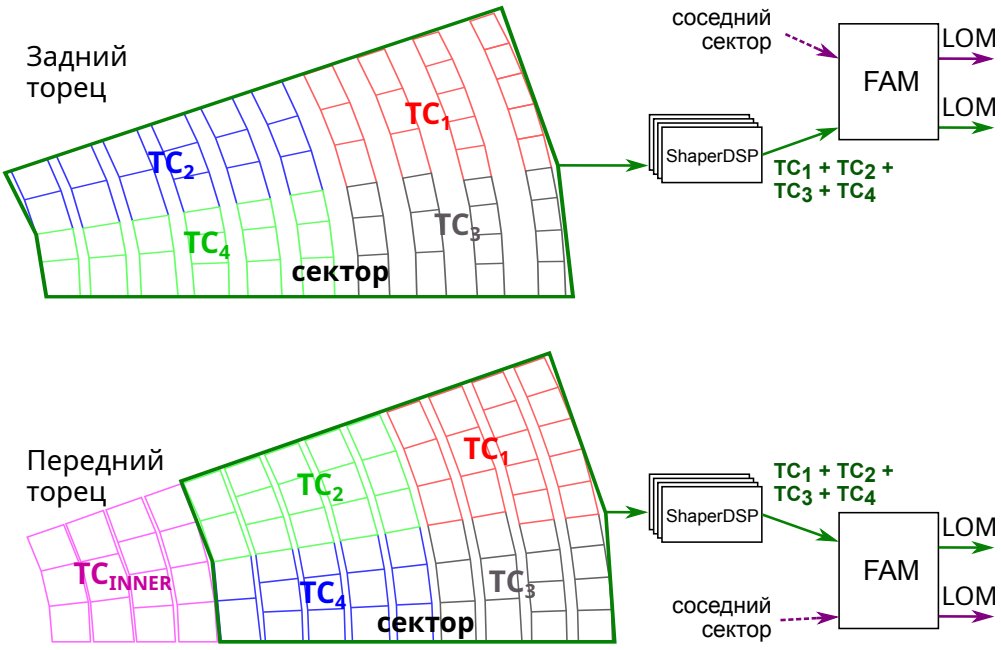

Рисунок 5 — Обработка сигналов с секторов торцевых секций. На рисунке TC (Trigger Cell) обозначает триггерную ячейку, LOM (Luminosity Online Monitor) — онлайн-монитор светимости

Как было упомянуто в разделе 1.3.2, ShaperDSP суммирует сигналы с 16 каналов триггерной ячейки и передаёт сумму в модуль FAM, который суммирует сигналы с четырёх ShaperDSP (одного сектора); один модуль FAM обрабатывает сигналы с двух секторов. Далее сигнал направляется по аналоговому выходу на вход онлайн-монитора светимости, который имеет 16 входных портов

для обработки 32-х сигналов с секторов торцевых секций; каждый порт имеет два сигнальных кабеля.

Для оцифровки входных сигналов с секторов торцевых секций, модуль оборудован четырьмя 12-битными восьмиканальными АЦП AD9637 [19] с частотой оцифровки 40 МГц.

Оцифрованные данные передаются на входы ПЛИС Xilinx Spartan-6 [20]. Чтобы обрабатывать внешние запросы к LOM, используется встроенный микропроцессор с программным ядром MicroBlaze. Исполняемая на микропроцессоре программа может получать и обрабатывать внешние запросы через интерфейс Ethernet по протоколу TCP.

Для управления монитором светимости был реализован гибкий программный интерфейс приложения (API, application programming interface), предоставляющий функции диагностики, переключения режима чтения данных, доступа к регистрам и флэш-памяти. Пример одного из запросов к модулю представлен на рисунке 6. Для улучшения быстродействия, API также предоставляет возможность в один запрос прочитать или записать значения группы регистров.

1.5 Калибровочные заходы

Поскольку ECL должен обеспечивать энергетическое разрешение лучше 1%, необходима процедура проверки стабильности амплитудных и временных характеристик каждого канала ECL с точностью лучше 0,2%. Для этой проверки используется ежедневная калибровка по тестовому сигналу, позволяющая своевременно обнаруживать потенциальные проблемы с электроникой. Обычно для калибровок предоставляется не более нескольких минут в день, поэтому важно иметь инструменты, позволяющие быстро набрать и проверить калибровочные данные.

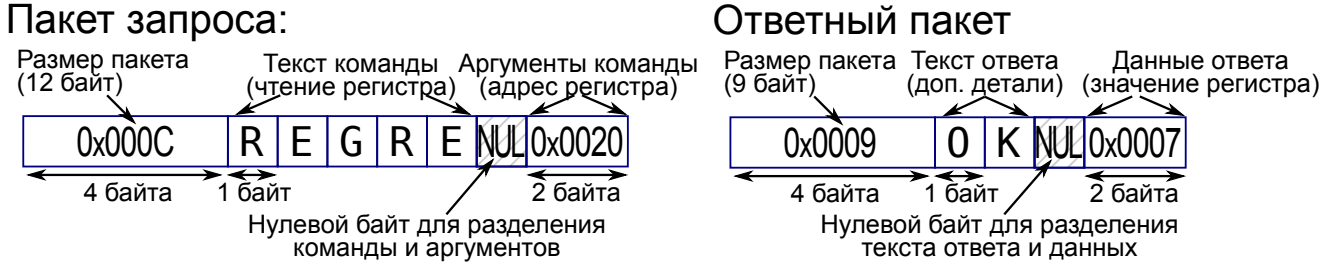

Рисунок 6 — Формат запроса на чтение регистра из монитора светимости

Периодически, как правило после замены одного из модулей ECL DAQ, выполняется калибровка описанных в разделе 1.3.2 коэффициентов аттенюации, при которой совместно набираются данные ECL (модули ECLCollector) и ECL триггера (модули FAM). Поскольку модуль ShaperDSP отправляет в систему ECL триггера сумму сигналов с 16 каналов ECL, требуется изолировать вклад каждого канала. Поэтому при выполнении калибровки для каждого из 16 каналов ShaperDSP набирается отдельный заход, где тестовый сигнал подаётся только на вход этого канала. Используя полученные данные, группа ECL триггера определяет 8736 аттенюационных коэффициентов, по суммарному числу каналов ECL.

Для повышения точности выдаваемой информации, сектора монитора светимости раз в две недели калибруются по генератору тестовых сигналов. Эта процедура во многом аналогична калибровке коэффициентов аттенюации, однако в ней не требуется изолировать вклад каждого канала. Для этой калибровки необходима синхронизованная процедура, которая одновременно конфигурирует и запускает сбор данных как с ECL, так и с монитора светимости.

Также, раз в месяц выполняется калибровка нелинейности, состоящая из нескольких заходов, где амплитуда тестового сигнала пошагово повышается от 1,5 МэВ до 12 ГэВ. Набранные данные используются в генерации DSP коэффициентов и позволяют обеспечивать нелинейность лучше $2 \cdot 10^{-3}$ на всём диапазоне входных значений АЦП.

1.6 Система сбора данных детектора Belle II

Система сбора данных Belle II DAQ предназначена для чтения сигналов детектора в момент получения триггера первого уровня и последующей передачи данных от внешней электроники через несколько шагов обработки в дисковый массив.

Сложность системы сбора данных зависит от множества факторов: сложности детектора (числа каналов и сложности считывающей электроники), объёма считываемых данных, частоты триггера. Если рассматривать детекторы предыдущего поколения, к примеру КМД-2, объём считываемых данных составлял около 500 кБ/с [21].

Увеличение светимости приводит к увеличению объёма данных и частоты триггера. Возрастают требования как к мощности серверов системы сбора

данных, так и к мощности считывающей электроники. Так, в большинстве детекторов LHC, исходя из входного объёма данных, доходящего до ТБ/с, используется «система сбора данных без триггера», в которой считывание информации происходит непрерывно. Решение о записи данных производится в триггере высокого уровня (High Level Trigger, HLT).

В Таблице 1 приведено сравнение характеристик систем сбора данных различных детекторов. В случае Belle II поток данных меньше, чем с детекторов LHC, поэтому было решено выполнять запись данных по сигналу триггера первого уровня. Тем не менее, ожидаемая высокая частота триггера требует использование HLT, который позволяет непрерывно проводить реконструкцию поступающих данных и фильтровать их по заданным критериям. Ввод HLT предоставляет новые возможности, но также и усложняет программное обеспечение сбора данных, в результате расширяя спектр поставленных задач, как будет описано далее.

Таблица 1 — Сравнение характеристик систем DAQ различных детекторов

Эксперимент	Число каналов	Частота триггера (L1 → HLT)	Поток данных (L1 → HLT)	Мёртвое время
СНД [22, 23]	2,4K	1 кГц	20 МБ/с	6%
КМД-3 [24, 25]	15K	1 кГц	90 МБ/с	5%
КЕДР [26]	20K	3,5 кГц	60 МБ/с	8,6%
KLOE [27]	23K	10 кГц	50 МБ/с	2,2%
BESIII [28–30]	30K	4 кГц	56 МБ/с	5%
Belle [31]	150K	0,5 кГц	15 МБ/с	10%
BaBar [32, 33]	230K	5,5 кГц → 800 Гц	0,2 ГБ/с → 26 МБ/с	2%
CLEO III [34]	400K	1 кГц	25 МБ/с	2%
LHCb [35, 36]	1,1M	40 кГц → 200 Гц	6 ГБ/с → 50 МБ/с	2,5%
Belle II [37][1]	8,4M	30 кГц → 6 кГц	18 ГБ/с → 3 ГБ/с	3,4%
ALICE [38, 39]	15M	500 кГц→ 2 кГц	15 ГБ/с → 2 ГБ/с	20%[2]
CMS [40–43]	78M	750 кГц→ 7,5 кГц	55 ГБ/с → 1 ГБ/с	3%
ATLAS [44–46]	100M	100 кГц→ 1 кГц	160 ГБ/с→ 1,6 ГБ/с	1%

Как указано в таблице, в случае Belle II мёртвое время DAQ при частоте

[1] Для Belle II приведены ожидаемые значения на проектной светимости.

[2] Мёртвое время при наборе данных для протон-протонных столкновений.

триггера 30 кГц составляет 3,4% от полного времени работы системы [47]. Это гарантирует хорошую работу DAQ для светимости $8 \cdot 10^{35}$ см$^{-2}$ с$^{-1}$, при которой ожидаемая частота первичного триггера составит 20 кГц.

1.6.1 Модули COPPER

Структура системы DAQ отображена на рисунке 7. Сигналы с подсистем детектора оцифровываются и далее передаются через оптоволокно в модули COPPER [48].

Для передачи данных используется высокоскоростной канал связи Belle2Link [17], который расширяет технологию RocketIO GTP [49] для взаимодействия между различными видами ПЛИС, использующимися в детекторе.

На модулях COPPER запущена операционная система Scientific Linux 5, что позволяет исполнять на них процессы медленного контроля и управления набором данных, которые взаимодействуют с электроникой детектора по протоколу Belle2Link.

Здесь и далее описывается сбор данных со всех подсистем Belle II, кроме пиксельного детектора (PXD). На проектной светимости, ожидаемый поток данных с PXD может достигать 30 Гб/с, что превышает пропускную способность COPPER, поэтому для этой подсистемы используются специальные модули ATCA [50].

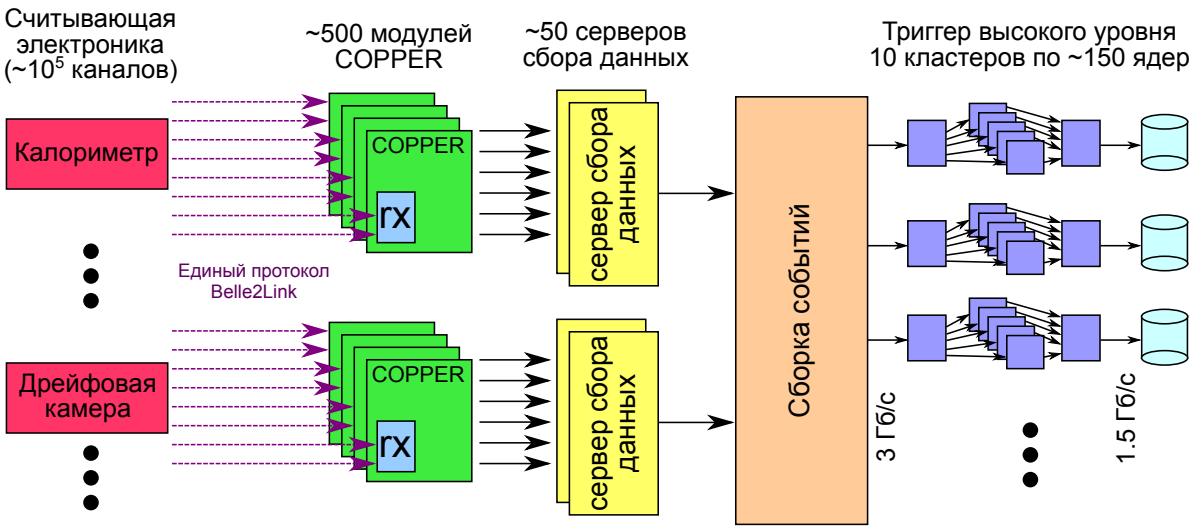

Рисунок 7 — Сбор данных в Belle II Data Acquisition System

1.6.2 Триггер высокого уровня

Данные с COPPER передаются по соединению FastEthernet в первую стадию построителя событий (Event Builder), где происходит дальнейшее разделение данных по событиям и снижение их объёма. В контексте DAQ, событием называется набор данных со всех подсистем детектора, относящихся к одному триггеру.

Из первой стадии EventBuilder данные переходят в распределённую систему триггера высокого уровня (HLT), которая осуществляет полную реконструкцию события. В Belle II используется два уровня триггера. Триггер первого уровня за ~4 мкс принимает решение о начале обработки события. HLT реализован в программном обеспечении и позволяет дополнительно снизить поток данных на 50%.

Далее данные со всех подсистем детектора, включая PXD, переходят во вторую стадию Event Builder, где происходит их окончательное объединение в события [51].

1.6.3 Фреймворк для обработки данных BASF2

Для онлайн-обработки данных на HLT, а также для офлайн-анализа данных, в Belle II применяется фреймворк BASF2 (Belle Analysis Software Framework 2). Для того чтобы обеспечить гибкость работы и исключить лишние зависимости, все задачи фреймворка выделены в модули, написанные на языке C++, от чтения из файла до полного моделирования детектора.

Поскольку предшествующий фреймворк BASF не предоставлял сохранность состояния объектов, а также из-за большого числа требуемых изменений при переходе от детектора Belle к Belle II, новую версию было решено переписать целиком, по возможности переиспользуя старые алгоритмы [52]. В BASF2 используются такие сторонние библиотеки как ROOT [53], boost [54], CLHEP [55] и libxml [56].

Новый фреймворк заимствует из BASF концепцию пути (Path), последовательности модулей, поочерёдно выполняющих над поступающими данными заданные действия. Конфигурация фреймворка и установка пути осуществляется через управляющие файлы, написанные на языке Python. Путь не обязательно линеен — BASF2 поддерживает условное ветвление на основе целочисленных значений, которые могут возвращать модули.

Для большого числа модулей в фреймворке поддерживается обработка данных, параллельная по числу обрабатываемых событий.

С путём также связано хранилище данных (DataStore), которое используется для передачи информации между модулями. В DataStore сохраняются все данные, обрабатываемые в модулях, которых имеют права как чтения, так и записи. Концепция пути и DataStore изображена на рисунке 8.

Как правило, модули в пути могут быть разделены на следующие группы:

- Построение геометрии подсистем детектора на основе данных в формате XML из центрального репозитория BASF2.
- Чтение локальных данных в DataStore, получение данных в реальном времени, моделирование физических процессов в детекторе или в его отдельных подсистемах.
- Пошаговый анализ данных, выполняемый десятками модулей разной степени сложности.
- Вывод полученной информации. Подразумевается сохранение в локальный файл, передача далее информации следующему звену обработки данных или отображение.

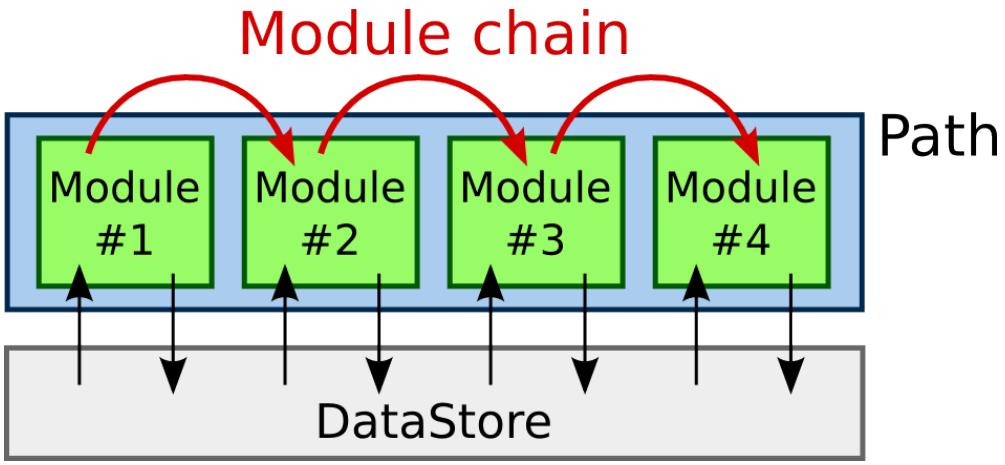

Рисунок 8 — Концепция пути в BASF2 [52]

1.7 Программное обеспечение медленного контроля и управления заходами

1.7.1 Использующиеся базы данных

Для хранения калибровочной информации и конфигурации системы сбора данных, в эксперименте Belle II используются две основные базы данных: ConditionsDB [57] и DAQ DB [58].

ConditionsDB предназначена преимущественно для хранения калибровочной информации. Калибровочные коэффициенты хранятся внутри файлов библиотеки CERN ROOT [53], а сопутствующие им метаданные (к примеру, версия калибровки) хранятся внутри таблиц PostgreSQL [59].

Поскольку почти каждый объект ROOT может быть преобразован в документ формата JavaScript Object Notation (JSON) [60], ConditionsDB может быть представлена как документоориентированная БД [61]. Однако следует учитывать, что, в отличие от стандартных документоориентированных БД, объекты ROOT поддерживают эволюцию схемы [62,63] и могут иметь довольно сложную внутреннюю структуру.

DAQ DB предназначена для хранения различных конфигураций, относящихся к системе сбора данных. Конфигурации организованы в документы (хранилища вида «ключ-значение»), которые сериализуются и сохраняются в таблицы PostgreSQL.

1.7.2 Система медленного контроля

В эксперименте Belle II под системой медленного контроля подразумевается сеть взаимодействующих демонов, которые мониторируют оборудование, предоставляют интерфейс чтения/записи к регистрам электроники детектора и управляют общей системой сбора данных. Она использует два фреймворка: Network Shared Memory 2 (NSM2) [37, 64] и Experimental Physics and Industrial Control System (EPICS) [65], причём существует несколько гибридных приложений, позволяющих этим фреймворкам взаимодействовать друг с другом.

Для подсистем детектора Belle II используется NSM2, в то время как EPICS преимущественно используется для медленного контроля SuperKEKB.

NSM2 является развитием фреймворка NSM [64], использовавшегося в эксперименте Belle и на установке Telescope Array. Основными добавлениями яв-

ляются возможность работы на 64-разрядных системах и автоматическое определение формата данных при выделении разделяемой памяти. Программа, использующая NSM2, создаёт одну или несколько структур языка C (struct) в разделяемой памяти и пересылает её всем другим программам NSM2. Пересылка выполняется каждые несколько секунд, используя широковещательный канал UDP. Также NSM2 поддерживает быстрые TCP-запросы, для обработки которых программа может задавать callback-функции.

1.7.3 Система управления заходами

Заходом в эксперименте Belle II называется относительно короткий (менее 8 часов) период набора данных, в течение которого параметры детектора и ускорителя остаются практически неизменными.

Под управлением заходами подразумевается управление системой сбора данных, изменение конфигурации триггера первого уровня и параметров HLT.

Для создания графических интерфейсов пользователя в системе управления заходами используется Control System Studio (CSS) [66] — мощная интегрированная среда разработки, основанная на Eclipse, которая может управлять DAQ как через NSM2, так и через EPICS.

Хотя сам интерфейс реализован на языке Java, внутренние скрипты для работы с системой медленного контроля реализуются либо на Javascript, либо на Jython [67] — гибриде языков Java и Python.

1.7.4 Монитор качества данных

Монитор качества данных (DQM, Data Quality Monitor) является важным инструментом в физическом эксперименте, позволяя в реальном времени проводить реконструкцию собранных данных, определять эффективность DAQ и быстро обнаруживать проблемы, которые нельзя заметить на стадии низкоуровневой проверки потока данных с отдельных модулей.

Программный дизайн DQM, реализованный в Belle II, описан в [68, 69]. В момент набора данные реконструируются и обрабатываются модулями BASF2 в режиме реального времени, которые генерируют гистограммы ROOT с информацией о текущем качестве данных.

Система DQM развёрнута на двух платформах. Первая работает на HLT и имеет доступ ко всем событиям, однако в ней недоступны данные пиксельного

детектора (на их обработку требуется слишком много времени). Вторая работает на отдельной вычислительной ферме ExpressReco. В ExpressReco DQM обрабатывается только 1% событий, но в ней включены данные от всех подсистем детектора. Для отображения гистограмм создан веб-сервер, использующий библиотеку JavaScript ROOT (JSROOT) [70].

Помимо отображения информации с пиксельного детектора в режиме реального времени, ExpressReco DQM независимо воспроизводит почти все гистограммы из HLT DQM (с меньшим объёмом статистики). Таким образом, если система HLT DQM по какой-то причине становится недоступна, дежурный эксперт может использовать ExpressReco DQM в качестве резервного варианта.

1.7.5 Инструменты для координации экспертов

Для организации работы дежурных экспертов в эксперименте Belle II используется множество веб-инструментов, наиболее важными являются:

- ShifTool, система для организации расписаний и рассылки оповещений дежурным экспертам, основанная на Easy!Appointments [71], менеджере расписаний с открытым исходным кодом. Исходно ShifTool была создана для использования в эксперименте KLOE II во Фраскати, позже адаптирована для Belle II.
- RocketChat [72], чат-сервер с открытым исходным кодом и поддержкой REST API для автоматизации любых действий.
- Confluence [73], вики-система для организации знаний. В эксперименте Belle II также используется для отображения текущего статуса дежурств.

1.8 Выводы к главе 1

В первой главе кратко описан эксперимент Belle II, приведена более детальная информация об электромагнитном калориметре и об онлайн-мониторе светимости. Также описана использующаяся в эксперименте система сбора данных. Описаны релевантные в дальнейших главах базы данных, фреймворки BASF2, EPICS, NSM2 и среда разработки CSS.

Основываясь на перечне используемых в Belle II программных средств, можно более детально сформулировать требования к программному обеспечению системы сбора данных с электромагнитного калориметра.

ГЛАВА 2. АНАЛИЗ ТРЕБОВАНИЙ СИСТЕМЫ СБОРА ДАННЫХ КАЛОРИМЕТРА

2.1 Требуемый функционал

На рисунке 9 приведён стандартный цикл задач, выполняемых ПО системы сбора данных в течение эксперимента.

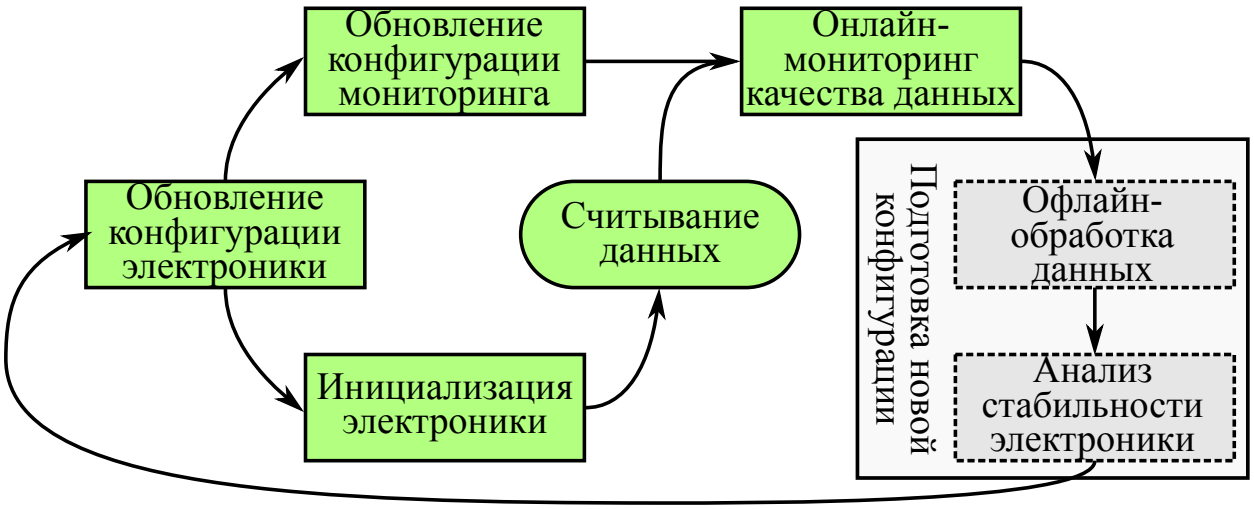

Рисунок 9 — Блок-схема задач системы ECL DAQ

На основе заранее подготовленных данных калибровки, генерируется новая версия конфигурации калориметра. Созданная конфигурация обновляется как в мониторе качества данных, так и в считывающей электронике. Далее выполняется считывание данных, параллельно с мониторингом качества данных и стабильности работы электроники. Сохранённые данные проходят через множество стадий офлайн-обработки, в частности выполняется временная и энергетическая калибровки, подготавливаются новые DSP коэффициенты, проводится детальный анализ данных. В итоге, результаты калибровки используются при создании новой версии конфигурации, возвращая рабочий процесс к началу цикла. Для чтения данных с калориметра используется универсальное программное обеспечение для получения данных с модулей COPPER [37]. Поэтому, как было описано во введении, данная работа фокусируется на всех остальных задачах.

Кроме того, поскольку монитор светимости должен предоставлять информацию непрерывно, вне зависимости от текущего статуса захода, для него необ-

ходимо разработать отдельную система сбора данных. Она должна решать все те же задачи, что и основная система DAQ — управление конфигурациями, инициализацию, мониторинг, запись и анализ данных, при этом интегрируясь с основной системой медленного контроля.

2.2 Требования к архитектуре системы

Помимо реализации требуемого функционала, важно, чтобы разрабатываемое ПО помогало обеспечить максимально высокую эффективность сбора данных. Чтобы свести время простоя детектора к минимуму, необходимо, чтобы ПО имело продуманный интерфейс пользователя, а также предоставлять инструменты, позволяющие быстро проводить диагностику и исправление потенциальных проблем с электроникой ECL. В рамках этого требования также важно реализовать систему автоматического оповещения о любых обнаруженных проблемах.

В ходе цикла жизни любого крупного программного продукта почти неизбежно возникают дополнительные задачи, не учтённые в начальной стадии планирования. Поэтому разработка ПО должна уделять внимание стандартным требованиям, предъявляемым к архитектуре программы [74]:

- Расширяемость системы.
- Лёгкость поддержки в дальнейшем.
- Создание переиспользуемых компонент.

Также ПО системы сбора данных должно удовлетворять двум основным сценариям использования. С одной стороны, поскольку ECL DAQ является распределённой системой, важно предоставлять программный интерфейс (API, Application Programming Interface) для доступа к наиболее востребованным функциям отдельных программных модулей.

С другой стороны, в периоды, когда на детекторе проводятся технические работы, ПО должно уметь работать в изолированном режиме, независимо от доступности DAQ DB или системы медленного контроля.

Эти сценарии использования хорошо соответствуют принципам разработки микросервисной архитектуры [75]. Микросервисы являются упрощённым вариантом сервис-ориентированной архитектуры, которая направлена на разра-

ботку распределённых систем с минимальными зависимостями между отдельными компонентами. При этом ключевыми требованиями к разработке являются сфокусированность каждого отдельного приложения на своей задаче, слабая связность между приложениями и стандартизация (сетевого) протокола взаимодействия между ними.

Следование этим требованиям обеспечивает большую свободу в выборе программных средств и упрощает переход на новые версии ПО: для обновления достаточно остановить одну малую компоненту системы, не затрагивая остальные.

Некоторые программные средства, упрощающие разработку микросервисов, описаны в разделе 2.5.2.

2.3 Требования к интерфейсу пользователя

В данной работе мы будем рассматривать 4 основных вида интерфейсов пользователя:

1. Интерфейс командной строки (CLI).
2. Текстовый интерфейс пользователя (TUI).
3. Графический интерфейс пользователя (GUI).
4. Веб-интерфейс пользователя (WUI).

Как правило, расширяемость системы проще всего достигается в CLI и сложнее всего — в WUI. Преимущественно это объясняется повышающейся сложностью разработки, в частности тем, что в графических приложения существенно труднее автоматизировать тестовые запросы. По этой причине важно либо предоставлять автоматизированные тесты, использующие специальные программные библиотеки для эмуляции ввода пользователя (такие как xdotool, sikuli, selenium), либо предоставлять API для доступа к основным функциям приложения.

Поскольку Belle II — международный эксперимент, эксперты очень часто управляют системой сбора данных, находясь в своих научных центрах вне KEK. Для это важно, чтобы все пользовательские интерфейсы были доступны удалённо, требовать минимальных настроек с клиентской стороны и работать при нестабильном соединении.

Для удалённого доступа к приложениям GUI на Unix-подобных операционных системах имеется два основных варианта: Virtual Network Computing (VNC) [76], в котором по сети передаются данные видеобуфера, и X2Go, в котором по сети передаются примитивы X11, составляющие графическое приложение. На практике, оба решения предоставляют примерно одинаковое быстродействие, однако VNC проще настроить на стороне клиента, поэтому было решено использовать этот вариант. Кроме того, в отличие от X2Go, VNC ориентирован на случаи, когда несколько клиентов управляют одной и той же удалённой сессией, что лучше подходит для задач данной работы.

2.4 Требования к программному интерфейсу

При разработке API рассматривалось два семейства протоколов: Remote Procedure Call (RPC) и REpresentational State Transfer (REST) API [77]. Поскольку RPC ориентирован на процедуры, реализации этого протокола использовались в системе медленного контроля и для автоматизированных тестов. REST API ориентирован на данные и поэтому использовался при реализации API для доступа к базе данных.

2.5 Используемые программные средства

Используемые программные средства были преимущественно определены существующими фреймворками, использующимися в эксперименте Belle II. Поэтому в разработке ПО использовались языки программирования C, C++ и Python. Для внутренней логики CSS использовался Jython. Программно-аппаратная (back-end) часть веб-интерфейсов разрабатывалась на языках PHP и Python, клиентская часть (front-end) — на языке JavaScript.

Комментарии к коду были написаны в формате Doxygen, [78] поэтому они могут быть легко экспортированы и скомпонованы в единую документацию.

На серверах системы сбора данных в эксперименте Belle II используется операционная система Scientific Linux 6, где проще использовать Python 2.7. Однако для офлайн-обработки данных преимущественно используется Python 3.8. Поскольку имеется такое различие версий, в рамках данной работы модули на языке Python разрабатывались с учётом поддержки совместимости между Python 2 и Python 3. Аналогично, поскольку стандартная версия компилятора на Scientific Linux 6 поддерживает только стандарт C++98, код C++ писался

с использованием данного стандарта. Исключение составляет код для офлайн-обработки данных, который использует стандарт C++17.

Процесс разработки был в значительной степени основан на модели Rapid Application Development (RAD) [79], где, в противовес каскадной модели (waterfall model), основным приоритетом ставится возможность быстро адаптировать проект к требованиям, которые могут измениться в ходе эксперимента. Одним из важных методов, использующихся в этой модели, является инкрементное прототипирование, когда требуемые программные модули разрабатываются на высокоуровневом языке программирования, в данном случае Python, и при необходимости переписываются на C++.

На практике RAD можно разбить на стандартные шаги: планирование, разработка и интеграция, однако этот подход имеет ряд особенностей, которые хорошо сочетаются с особенностями реализации приложений для ECL DAQ. В частности, проект разбивается на ряд слабо зависимых фрагментов, аналогично проектам, использующим микросервисную архитектуру. При этом RAD предполагает в первую очередь писать наиболее критичные фрагменты, позволяя на ранней стадии разработки обнаруживать узкие места в архитектуре ПО и потенциальные проблемы с быстродействием. Поэтому для большинства приложений, описанных далее, в первую очередь были написаны прототипы на Python или Bash, которые тестировались экспертами ECL.

Написание прототипов также помогает осуществлять другой важный принцип RAD — частая обратная связь с пользователями ПО, позволяющая лучше сформулировать функциональные требования, предъявляемые к проекту. Таким образом, шаги планирования и разработки в RAD неразрывно связаны. Прототипы используются для постоянной корректировки исходного плана, позволяя улучшить надёжность конечного продукта и удостовериться, что он точно соответствует требованиям пользователей.

2.5.1 Особенности использования Python

Главным преимуществом использования Python является сокращение сроков разработки приблизительно в два раза [80, 81], с пропорциональным сокращением размера исходного кода программы [82].

Главным недостатком Python в разработке систем медленного контроля является Global Interpreter Lock (GIL). Это мьютекс, использующийся в стандартной реализации интерпретатора Python, который гарантирует, что в каж-

дый конкретный момент времени только один из потоков использует высокоуровневые функции языка. В результате, из-за GIL, только операции ввода/вывода и системные запросы могут выполняться параллельно. Это существенно ухудшает производительность многопоточных приложений, написанных на Python, в некоторых случаях многопоточная реализация может работать даже медленнее, чем однопоточная [83, 84], как показано на рисунке 10.

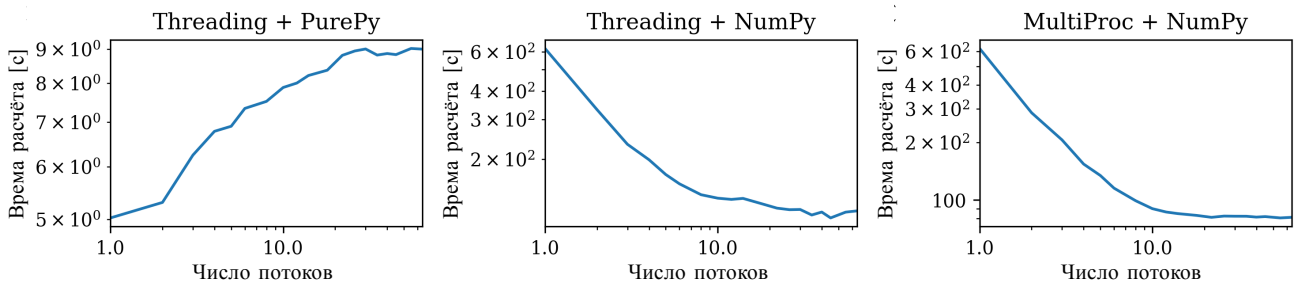

Рисунок 10 — Быстродействие итерационного метода решения уравнения $u_{xx} + u_{yy} = 0$ в нескольких реализациях на языке Python в зависимости от числа потоков [84]

Версия Python 3.2 [85] использует новую, оптимизированную реализацию GIL, однако в некоторых случаях она всё равно работает слишком медленно [86]. По этой причине, многопоточные приложения, для которых быстродействие являлось важным фактором, разрабатывались на C++. При разработке таких приложений на Python параллельные операции по возможности разделялись на отдельные процессы вместо отдельных потоков.

2.5.2 Генерация программного интерфейса

Чтобы уже на ранней стадии разработки предоставлять API для доступа к основным функциям приложения, на языке C++ была разработана простая библиотека генерации именованных функторов на стадии компиляции. Любая функция C++ или метод объекта могут быть в одну строку экспортированы в API, существенно упрощая интеграцию с системой DAQ.

Пример кода показан на рисунке 11. Определение функции обёрнуто в директиву препроцессора, которая регистрирует функцию с таким именем в хэш-таблице. Используя эту таблицу, становится очень просто создать и расширять консольный интерфейс пользователя. Реализация библиотеки для объявления

функторов доступна в репозитории на GitHub[3].

```
// Вывести первую строку указанного файла
FUNCTOR(head, string filename)
{
  ifstream f(filename);
  string line;
  getline(f, line);
  return line;
}
```

```
$ ./example_cui
> head "/etc/hosts"
127.0.0.1 localhost
>
```

Функция 'head' может быть вызвана из командной строки.

Рисунок 11 — Пример кода, определяющего функтор для функции head

В результате, в минимальном объёме кода удаётся создать набор функций, которые тут же интегрируются в консольный интерфейс с интерпретатором синтаксиса пользовательских команд. Также, для достижения максимально гибкой архитектуры, следующей философии UNIX, сервис можно разделить на три процесса, взаимодействующих между собой.

1. Центральная программа, интегрированная с консольным интерфейсом и выполняющая команды, посылаемые пользователем.

2. «Серверизатор», запускающий внутри себя центральную программу и переадресующий в неё клиентские TCP-запросы.

3. «Демонизатор», создающий отдельную сессию, в которой в фоновом режиме запускается серверизатор.

Данная архитектура опирается на средства межпроцессного взаимодействия UNIX. В частности, «серверизатор» запускает центральную программу и взаимодействует с ней через неименованные каналы (UNIX pipes). Данные, приходящие по сети, отправляются на стандартный вход центральной программы, вывод центральной программы отправляется обратно клиенту. При этом центральная программа может быть реализована на любом языке программирования, главное чтобы она предоставляла консольный интерфейс пользователя.

«Демонизатор» использует системный вызов `fork()`, чтобы запустить дочерний процесс, не зависящий от текущей терминальной сессии. Опционально, в демонизатор несложно добавить запись вывода программы в файл и мониторинг статуса запущенного процесса.

[3] https://github.com/mikhailremnev/functors/

Код программы на C++, в которой вместе интегрированы «серверизатор» и «демонизатор», доступна в репозитории на GitHub[4].

С использованием этой библиотеки было реализовано несколько тестовых приложений, которые легко интегрируются с C++, Python, и простыми shell-скриптами.

2.6 Выводы к главе 2

Сформулированы требования к разработке программного обеспечения в сети сбора данных, а также отдельный набор требований к разработке ПО для монитора светимости. Описаны основные задачи, которые должны осуществляться системами мониторинга и медленного контроля.

Сформированы требования к пользовательскому интерфейсу и программному интерфейсу. Перечислены и проанализированы используемые программные средства, рассмотрены различные подходы для ускорения разработки программного обеспечения.

[4] `https://github.com/mikhailremnev/serverize`

ГЛАВА 3. УПРАВЛЕНИЕ КОНФИГУРАЦИЯМИ КАЛОРИМЕТРА

Как будет сказано далее, конфигурация ECL определяется большим числом параметров. Для этого, система ECL DAQ должна предоставлять детальный мониторинг и гибкие настройки конфигурации. Также важна интеграция двух баз данных: ConditionsDB и DAQ DB для трёх типов информации, хранящейся в конфигурации ECL:

1. Статические параметры, то есть параметры, которые не зависят от калибровочных коэффициентов. Эти параметры обычно имеют одинаковое значение для всех модулей и модифицируются только для особых тестов. К примеру, битовая маска используемых модулей ShaperDSP является статическим параметром.

2. Энергетические пороги и коэффициенты аттенюатора — параметры, которые, как правило, могут быть определены как арифметическое выражение. К примеру, $50 \cdot$ `energy_conversion_coef` для порога величиной 50 МэВ. Коэффициенты для преобразования из единиц АЦП в энергию определяются из процедуры энергетической калибровки.

3. Коэффициенты DSP, которые определяются индивидуально для каждого канала, записываются во флэш-память ECLCollector и затем загружаются в SDRAM модулей ShaperDSP. Из-за большого числа этих коэффициентов, их требуется предварительно сжимать перед загрузкой в базу данных.

На каждый канал приходится 8 параметров (4 энергетических порога, 2 параметра для конфигурации порога χ^2, аттенюаторный коэффициент, коэффициент компаратора АЦП). Вдобавок на каждый ShaperDSP приходится 18 параметров (битовые маски, параметры сохранения формы сигнала). Учитывая, что ECL включает 8736 каналов и 576 ShaperDSP, конфигурация ECL суммарно включает $8 \cdot 8736 + 18 \cdot 576 = 80256$ значений.

Каждый тип захода имеет соответствующую конфигурацию. Для ECL используется одна основная конфигурация для набора физических данных и 14

конфигураций для различных типов калибровочных и тестовых заходов. Некоторые параметры идентичны в разных конфигурациях, некоторые зависят от типа захода.

3.1 Библиотеки для работы с базами данных

Хотя обе базы данных и используют одну и ту же систему управления БД PostgreSQL, передавать данные между ними довольно затруднительно из-за сильного различия внутренней структуры. Кроме того, ConditionsDB опирается на довольно крупный фреймворк BASF2 с большим числом внешних зависимостей, который затруднительно развёртывать и поддерживать на большом числе машин в сети DAQ.

Поэтому были разработаны две небольшие утилиты на языке программирования Python, осуществляющие доступ к ConditionsDB и DAQ DB. С их использованием была реализована схема передачи конфигураций, приведённая на рисунке 12. Данная схема позволяет, в частности, обновлять амплитудные пороги, записываемые в электронику, на основе актуальных данных энергетической калибровки. Эти две утилиты были добавлены в репозиторий эксперимента Belle II.

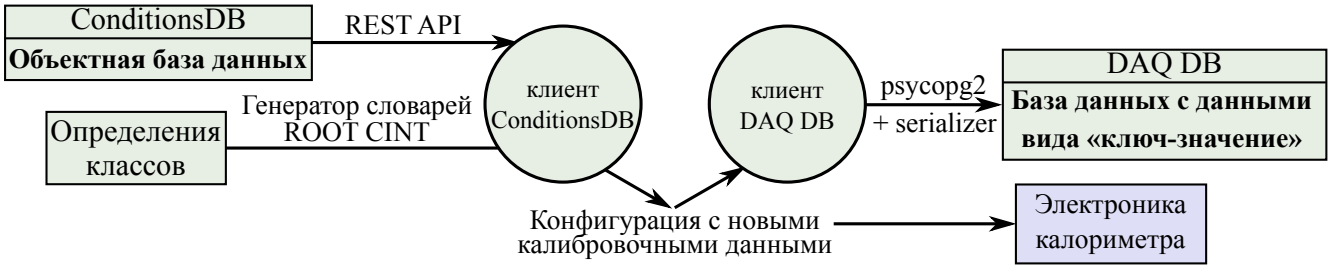

Рисунок 12 — Передача данных между ConditionsDB и DAQ DB

Кроме того, чтобы упростить развёртку этих библиотек в сети сбора данных, была тщательно протестирована обратная совместимость и подкорректирован алгоритм упаковки/распаковки объекта — это позволило обеспечить обратную совместимость вплоть до ROOT 5.34/26.

3.2 Схема конфигурационных данных

В исходной версии программной библиотеки медленного контроля для каждого модуля и для каждого типа захода требовалась отдельная конфигурационная таблица. Таким образом, исходная версия DAQ DB содержала ~900 таблиц. БД была реорганизована в 16 таблиц, разделённых на четыре категории:

1. Энергетическая калибровка.
2. Базовые параметры.
3. Значения порогов.
4. Параметры, зависящие от типа захода.

Кроме того, в конфигурации поддерживается синтаксис задания параметров как для всех каналов, так и для определённых групп:

```
        all.thr: 30 # порог=30 для всех каналов.
       barr.thr: 31 # порог=31 для каналов в цилиндрической части.
        fwd.thr: 32 # порог=32 передний торец.
        bwd.thr: 33 # порог=33 задний торец.
       col2.thr: 34 # порог=34 ECLCollector 2.
   col3.sh4.thr: 35 # порог=35 ShaperDSP 4, ECLCollector 3.
col5.sh6.ch7.thr: 36 # порог=36 Канал 7, ShaperDSP 4, ECLCollector 5.
```

Используемая схема также позволяет, если это потребуется, быстро реализовать наследование между базовыми и более специфическими конфигурациями.

3.2.1 Конфигурационные скрипты

В особых случаях, при настройке электроники для тестовых заходов или при использовании новых версий прошивки, возникает необходимость задать дополнительные параметры, не учтённые в текущей схеме. Поэтому был разработан скриптовый язык, позволяющий детально определять дополнительные действия, выполняемые на стадии установки конфигурации.

Чтобы минимизировать число внешних зависимостей, формальная грамматика языка была определена с использованием уже имеющихся на целевых

системах инструментов Flex [87] и Bison [88]. В спецификации Flex был определён сканер (преобразование текста в синтаксические элементы), в Bison — парсер (выстраивание полученных элементов в синтаксическое дерево), как показано на рисунке 13.

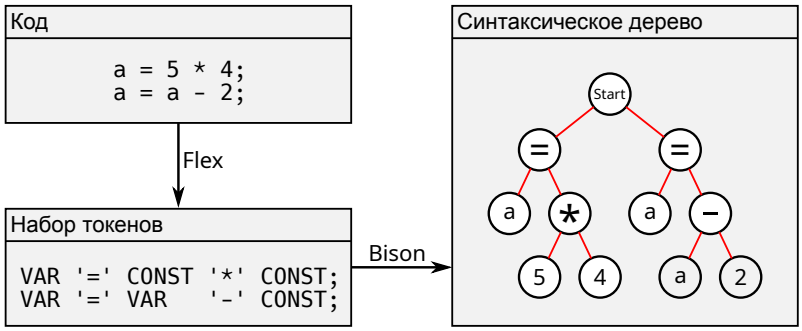

Рисунок 13 — Обработка скриптового языка

На первой стадии сборки, эти инструменты используются, чтобы преобразовать файл с формальным определением грамматики скриптового языка в интерпретатор на языке C++. На второй стадии сборки, интерпретатор компилируется вместе с остальными модулями программы.

При анализе скрипта генерируется синтаксическое дерево, по которому проходит интерпретатор, исполняя указанные команды. Поддерживаются арифметические операции, переменные с динамической типизацией, оператор if и циклы for. Также доступно несколько функций для работы с конфигурацией электроники. Язык использует подмножество синтаксиса C++, поэтому скрипты, написанные на нём, могут быть позже интегрированы в ПО медленного контроля с минимальными изменениями.

Чтобы интерпретатор мог конфигурировать электронику, используя внешние функции, достаточно указать эти функции в библиотеке для создания функторов, описанной в разделе 2.5.2. Таким же образом можно указывать методы и конструкторы классов. Разработанный парсер может быть запущен из нескольких потоков и поддерживает все стандарты языка C++, начиная с C++98. Исходный код реализации доступен в репозитории эксперимента Belle II[5].

В общем случае, если не ставится целью избежать дополнительных внешних зависимостей, вместо Flex и Bison для определения формальной грамматики рекомендуется использовать ANTLR, который проще в использовании [89]

[5] `https://stash.desy.de/users/remnev/repos/programmable_config/`

и обладает схожим быстродействием [90]. К сожалению, парсеры, реализованные в ANTLR, зависят от возможностей стандарта C++11. По этой причине их невозможно использовать на многих серверах системы сбора данных Belle II.

3.3 Синхронизация конфигураций с калибровочной базой данных

Программный фреймворк BASF2 предоставляет режим «позаходного моделирования» (run-dependent simulation), в котором параметры моделирования основываются на конфигурации детектора во время соответствующего захода.

С одной стороны, конфигурационные параметры из DAQ DB зависят от калибровочной информации из ConditionsDB и, с другой стороны, параметры позаходного моделирования из ConditionsDB зависят от конфигурационных данных из DAQ DB. Из-за этой циклической зависимости важно иметь надёжную процедуру синхронизации конфигурационных данных между двумя БД.

Одним из существенных затруднений в синхронизации двух БД является расхождение во времени между моментом, когда новая конфигурация электроники загружается в DAQ DB и моментом, когда эта конфигурация действительно устанавливается в электронике. Чтобы аккуратно учесть это расхождение, данная работа использует процедуру, проиллюстрированную на рисунке 14.

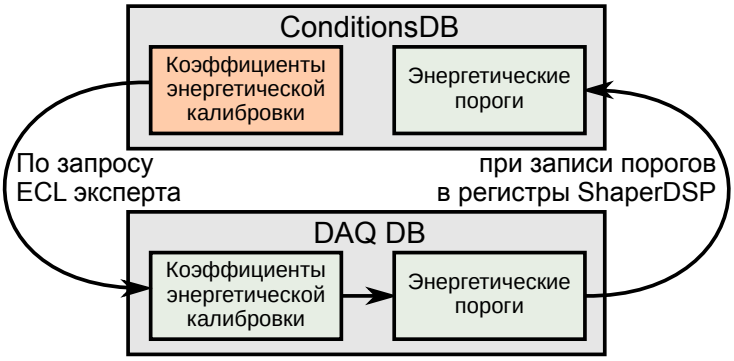

Рисунок 14 — Синхронизация коэффициентов энергетической калибровки и энергетических порогов между ConditionsDB и DAQ DB

Сначала, коэффициенты энергетической калибровки в DAQ DB обновляются по запросу ECL эксперта. Затем, энергетические пороги автоматически генерируются на основе новой калибровки и, при запуске нового захода, загружаются в считывающую электронику. В этот момент, новая версия энерге-

тических порогов сохраняется в ConditionsDB для использования в мониторе качества данных и в позаходном моделировании.

Аналогичные алгоритмы были реализованы, чтобы синхронизировать другие параметры электроники ECL, в частности коэффициенты аттенюатора и DSP коэффициенты.

3.3.1 Загрузка карты каналов в базу данных

Каждый счётчик калориметра ассоциирован с двумя индексами. В анализе данных, как правило, используется Cell ID, геометрический индекс кристалла от 1 до 8736. При конфигурации электроники важен Electronics ID, определяющийся на основе номера разъёма в ShaperDSP, к которому подключен кабель канала, номеру ShaperDSP и номеру ECLCollector. По этой причине, две БД хранят информацию в разном формате:

- ConditionsDB — данные в формате «Cell ID $\leftrightarrow$ значение».
- DAQ DB — данные в формате «Electronics ID $\leftrightarrow$ значение».

В рамках данной работы карта каналов, содержащая соответствие между Electronics ID и Cell ID, была добавлена в ConditionsDB, что позволяет утилитам, работающим с электроникой калориметра, всегда использовать актуальное соотношение этих двух индексов.

3.4 Синхронизация DSP-коэффициентов

Подгонка формы сигналов, поступающих с каналов калориметра, осуществляется в ПЛИС 576 модулей ShaperDSP. Поскольку во время сбора данных ионизирующее излучение может привести к сбоям в логике ПЛИС, необходимо отслеживать стабильность электроники. Для этого аппроксимация формы сигнала повторно осуществляется в программе-эмуляторе, встроенной в монитор качества данных (data quality monitor, DQM), в процедуре, показанной в разделе 6.1. Если результаты, поступившие с эмулятора и с ПЛИС расходятся, необходимо перезагрузить прошивку модулей ShaperDSP, иначе с калориметра будут приходить неправильные данные.

Проверка логики ShaperDSP может выдавать надёжные результаты только если DQM и электроника используют одну и ту же конфигурацию. В частности, конфигурация включает в себя 16 000 DSP-коэффициентов, определяющих

функцию подгонки сигнала для каждого модуля. Чтобы обеспечить непрерывную синхронизацию этих коэффициентов между электроникой и ConditionsDB, важно реализовать процедуру преобразования между ROOT файлами, сохранёнными в ConditionsDB и бинарными файлами, загружаемыми в динамическую память модулей ShaperDSP.

Такая процедура была реализована и встроена в BASF2, позволяя легко выполнять синхронные обновления DSP коэффициентов. Был разработан объект библиотеки CERN ROOT, который сохраняется в ConditionsDB и используется для генерации файлов с коэффициентами, впоследствии записывающимися во флэш-память модулей ECLCollector. Структура объекта была спланирована так, чтобы максимально упростить возможные изменения в формате DSP-файлов, при этом сохраняя обратную совместимость.

Процедура преобразования бинарных данных в объект ROOT также упаковывает коэффициенты, чтобы эффективнее использовать ресурсы калибровочной БД.

3.4.1 Алгоритм упаковки DSP-коэффициентов

Хотя сжатие DSP-коэффициентов при помощи алгоритма LZMA [91] может снизить размер выходного файла на ~50%, дополнительные процедуры упаковки могут ещё сильнее увеличить эффективность сжатия. В частности, в распакованном формате DSP коэффициенты сохраняются как 16-битные целые числа, однако для их хранения вполне возможно использовать меньшее число бит. Для этого используется алгоритм упаковки, специально подстроенный под периодическую структуру данных. Большинство массивов DSP коэффициентов F_i могут быть представлены в виде

$$F_{16n+m} = F_m + f_{16n+m}, \qquad n = 0 \dots 191, m = 0 \dots 15 \tag{5}$$

где для почти каждого i выполняется условие $|f_i| < 16$. Это означает, что массивы 16-битных чисел из $16 \cdot 192$ элементов могут быть реорганизованы как массив 16-битных чисел F_m на 16 элементов и массив 5-битных чисел f_i на $16 \cdot 191$ элементов. В результате, за счёт исключительно упаковки коэффициентов достигается следующий уровень сжатия:

$$\frac{16 \cdot 16 \cdot 192}{16 \cdot 16 + 5 \cdot 16 \cdot 191} \approx 3{,}2$$

На практике, алгоритм упаковки достигает уровня сжатия 2,9, потому что некоторые массивы DSP коэффициентов не могут быть представление в виде (5). Тем не менее этот алгоритм хорошо сочетается со стандартными алгоритмами сжатия данных. Использование его совместно с LZMA позволяет достичь уровня сжатия 7,9, как показано на рисунке 15.

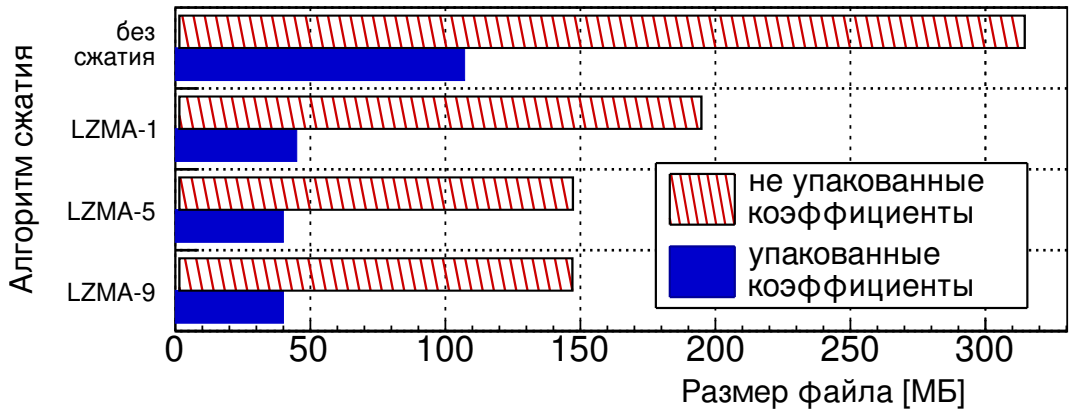

Рисунок 15 — Сравнение размеров выходного файла на разных уровнях сжатия для не упакованных и упакованных данных

Тестирование на одном из серверов HLT (процессор Intel Xeon E5-2650 v2) показало, что распаковка DSP-коэффициентов для одного модуля ShaperDSP занимает ~0,25 мс. Суммарное время распаковки занимает $576 \cdot 0{,}25 = 144$ мс, и укладывается в рамки ограничений на время инициализации.

3.5 Основные результаты главы 3

Конфигурация электроники калориметра разделяется на три группы параметров, каждая из которых имеет свои особенности. Кроме того, параметры могут варьироваться от типа захода и должны загружаться из двух баз данных.

По этой причине автором были разработаны клиентские приложения для работы с базами данных ConditionsDB и DAQ DB, переработана схема хранения конфигурационной информации в БД, реализован язык конфигурационных скриптов для более детальных настроек. Используя эти программные модули, были созданы инструменты для обновления амплитудных порогов, коэффициентов аттенюации и DSP коэффициентов в электронике калориметра.

Чтобы эффективно использовать ресурсы базы данных, был реализован алгоритм упаковки DSP коэффициентов, позволяющий снизить их итоговый

размер в 7,9 раз.

Основываясь на реализованных программах управления конфигурациями калориметра, можно разработать инструменты инициализации электроники и мониторинга качества данных с ECL.

ГЛАВА 4. ИНИЦИАЛИЗАЦИЯ ЭЛЕКТРОНИКИ КАЛОРИМЕТРА

4.1 Процедура инициализации

Инициализация системы сбора данных ECL — это сложная процедура, состоящая из большого числа шагов, которые могут быть разделены на две категории: обновление конфигурации во флэш-памяти ECLCollector и обновление конфигурации в энергозависимой памяти модулей ECLCollector и ShaperDSP. Как показано на рисунке 16, инициализация выполняется по соединениям JTAG [92], Ethernet и Belle2link [17], с трёх различных групп серверов.

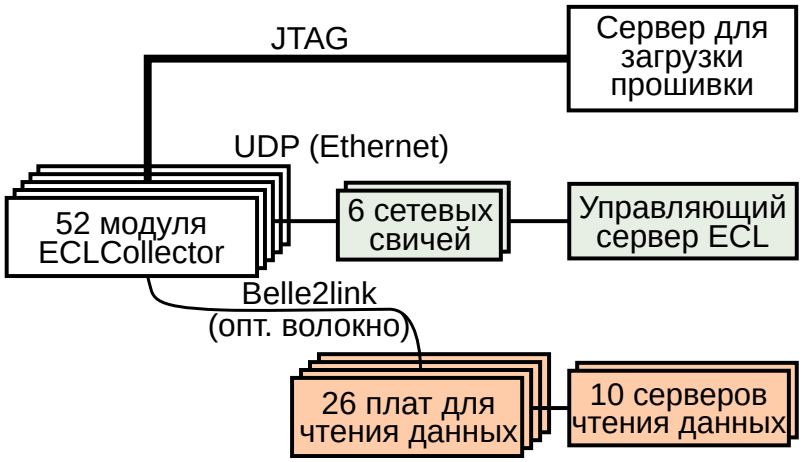

Рисунок 16 — Сервера, использующиеся в процедуре инициализации ECL

Во время холодного запуска, прошивка ECLCollector либо автоматически загружается из флэш-памяти, либо программируется извне через соединение JTAG. Затем ECLCollector устанавливает соединение по Ethernet к серверу, на котором установлено ПО для инициализации ECL. После этого, во флэш-память ECLCollector можно записать новую версию прошивки ECLCollector, прошивки ShaperDSP и коэффициентов DSP.

Далее, по запросу через Ethernet от ПО инициализации, ECLCollector конфигурирует подключенные модули ShaperDSP, загружая актуальную версию прошивки и DSP коэффициентов из своей флэш-памяти. Наконец, ПО устанавливает соединение по Belle2link для передачи данных и записывает все статические и зависящие от калибровки параметры с учётом текущей выбранной конфигурации.

Суммарно по соединению Belle2link устанавливается около 60 000 параметров, зависящих от калибровки и 20 000 статических параметров. Также, отдельные шаги инициализации могут зависеть от вида захода, типа триггера и списка активных модулей (некоторые модули ECLCollector могут быть отключены во время технических работ).

Чтобы эта процедура надёжно работала в различных сценариях использования, как ПО, так и прошивка должны поддерживать несколько резервных опций.

Прошивка ECLCollector принимает большинство своих команд как по Ethernet, так и по Belle2link. Некоторые параметры, зависящие от калибровки, могут быть указаны в заголовке файла с DSP коэффициентами.

Дополнительные требования к ПО инициализации ECL указаны в следующей секции.

4.2 Дополнительные требования к программному обеспечению

Как упоминалось в разделе 2.1, ПО для инициализации ECL должно поддерживать два существенно различных сценария использования: использование в период активного набора данных и использование на тестовых стендах/во время технических работ.

Для первого сценария использования, ПО должно получать все данные конфигурации из DAQ DB, а также корректно интегрироваться с системой медленного контроля. Поскольку библиотека медленного контроля реализована преимущественно на C++, основная часть библиотеки для инициализации ECL также написана на C++.

Для второго сценария использования, во время технических работ, как система медленного контроля, так и DAQ DB могут быть недоступны. Таким образом, важно иметь опцию использования резервной базы данных или загрузки конфигурационных параметров из кэшированных данных. Библиотека инициализации также должна поддерживать различные опции логирования, которые могут работать независимо от реализации логирования, использующейся в системе медленного контроля Belle II.

4.3 Фреймворк для работы с модулями ECLCollector

Инициализация электроники калориметра может проводиться двумя методами: через кабель CAT5 по протоколу Ethernet и через оптическое волокно по протоколу Belle2link. Исторически разработка ПО для взаимодействия с электроникой по Belle2link и по Ethernet велась независимо.

Поскольку основным методом инициализации является Belle2link, но отдельные операции быстрее проводятся через Ethernet, две отдельные библиотеки для взаимодействия с модулями ЭМ калориметра были расширены, переписаны на C++ и включены в единый фреймворк, позволяющий абстрагироваться от:

- Способа инициализации (Ethernet/Belle2link).
- Хранилища конфигурации (DAQ DB provider, DAQ DB, файл)
- Цели логирования (DAQ DB, файл, stdout)

4.3.1 Архитектура фреймворка

UML-диаграмма фреймворка приведена на рисунке 17.

Все операции ввода/вывода инкапсулированы внутри класса ECLCol, который имеет отдельную реализацию для каждого из трёх протоколов инициализации. Создание экземпляров класса ECLCol выполняется через паттерн «фабрика» [93], реализуемый классом ECLFactory. Помимо явного задания типа протокола, ECLFactory может автоматически определять требуемую реализацию ECLCol на основании имени хоста, где выполняется программа.

Иерархия хранилищ конфигурации может быть специфицирована как серия экземпляров класса ECLConfigLoader, которые реализуют паттерн «цепочка обязанностей» [93]. Этот класс поддерживает автоматическое кэширование запрошенных конфигураций с конфигурируемым временем жизни кэша. Использование «цепочки обязанностей» позволило оптимально адаптироваться к обоим сценариям использования — если DAQ DB недоступна, фреймворк автоматически перейдёт к следующему хранилищу конфигураций (резервной БД или файлам).

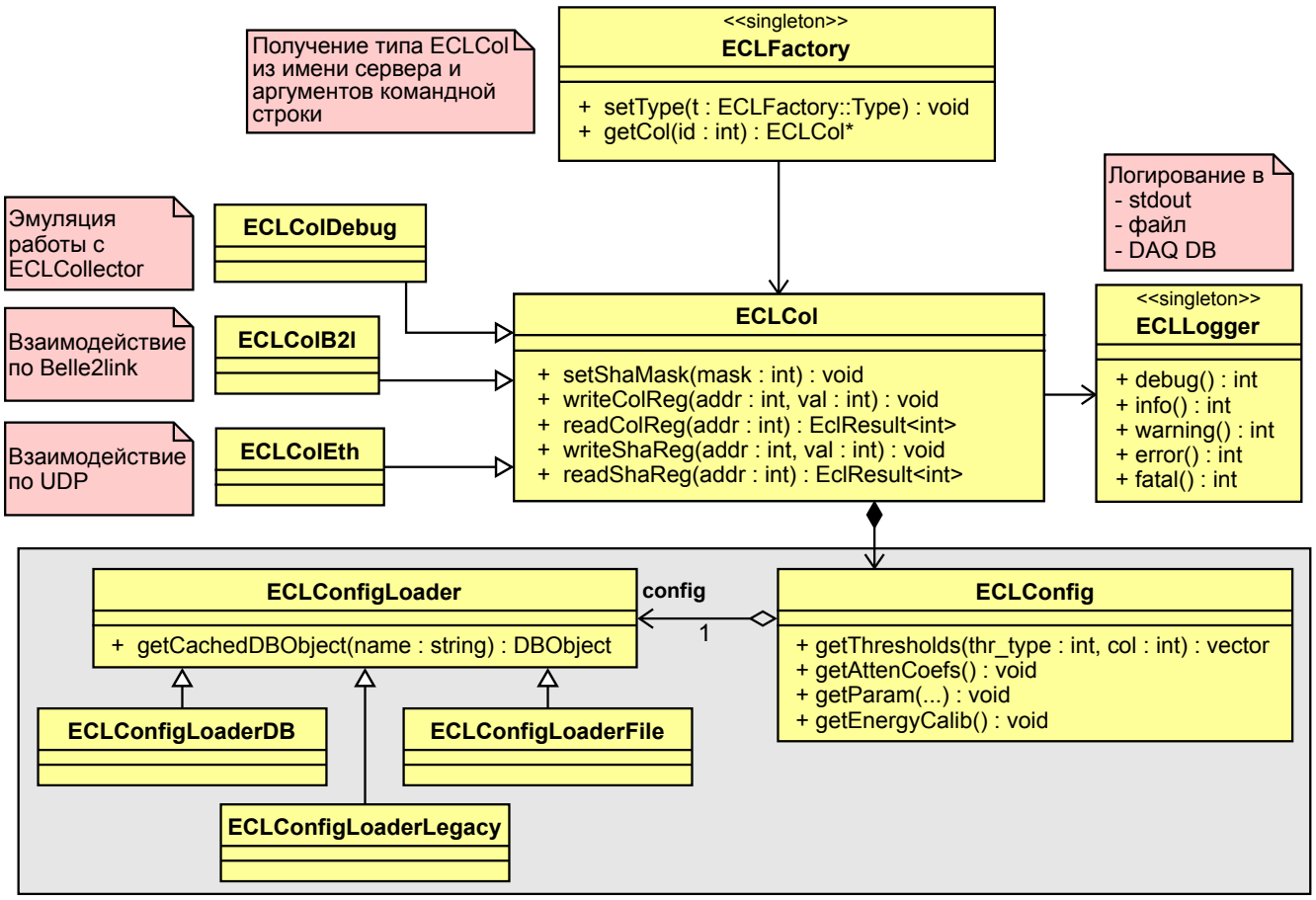

Рисунок 17 — UML-диаграмма основных классов фреймворка

4.3.2 Система сборки

В репозитории ПО для медленного контроля в качестве основной системы сборки используется GNU Make [94]. В случае исправления критических ошибок, сборка и развёртка ПО на целевых системах должна быть выполнена в кратчайшие сроки. Поэтому фреймворк инициализации ECL использует нерекурсивный Makefile [95], с помощью которого система быстрее строит дерево зависимостей между подпакетами и не требует повторных запусков Make.

Стоит отметить, что нерекурсивный Makefile пригоден только для небольших проектов. С ростом числа подпакетов, правила сборки становятся слишком сложными для поддержки [96]. В таком случае наиболее разумным шагом будет перейти к другой системе сборки, лучше подходящей для крупных проектов [97], к примеру SCons или Tup.

4.3.3 Ускоренная обработка исключений

Поскольку язык C++ оптимизирован для ситуаций, когда выбрасывается минимальное число исключений, использование исключений для обработки рутинных проблем приводит к замедлению работу программы.

Поэтому, для некоторых стандартных случаев (потеря и повторная пересылка пакетов, потеря соединения к БД) используется собственная система обработки исключений, подобная системе, используемой в языке Rust [98, 99].

Пример кода показан на рисунке 18. Идея заключается в использовании класса Result, реализованного наподобие std::variant, который может содержать либо возвращаемые данные, либо информацию об ошибке (как правило, std::string). Макрос TRY использует возможности компилятора GCC (statement expression), чтобы обрывать дальнейшую обработку выражения, если в функции чтения регистра возникла ошибка.

Таким образом, программа работает так же, как если бы использовались стандартные исключения, но при этом из объявления функции явно видно, может ли её исполнение привести к ошибке. Кроме того, в такой реализации очень легко задавать несколько попыток на выполнение операции, которая может привести к ошибке: для этого достаточно добавить макрос TRY_MULTIPLE, заданное число раз вызывающий внутри себя макрос TRY. Реализация класса Result и макроса TRY доступна в репозитории на GitHub[6].

Кроме того, как уже упоминалось выше, такая реализация позволяет немного улучшить быстродействие при обработке ошибок. Так, при частоте ошибок 0,1%, скорость работы программы, использующей класс Result, на 12% выше, чем у программы, использующей стандартные исключения C++.

```
Result<int> readRegister(int addr)
{
  // 0.1% chance for register reading to fail
  if (rand() % 1000 == 0)
    return ERROR("Could not read reg " + to_string(addr));
  else
    return rand() % 256;
}

Result<double> readTemperature()
{
  double temp = TRY(readRegister(10)) * 0.7 - 55;
  return temp;
}
```

```
int main()
{
  Result<double> result = readTemperature();
  if (result.isOk())
    printf("Temperature: %lf\n", result.getData());
  else
    printf("Error: %s\n", result.getError().c_str());

  return 0;
}
```

```
Temperature: 117.90

Error: Could not read reg 10
  in example.cpp:9
  in example.cpp:16
```

Рисунок 18 — Пример кода для обработки ошибок

[6] https://github.com/mikhailremnev/cpp_rustlike_error_handling

Помимо улучшения в производительности без усложнения программы, эта система интегрирована с расширениями компилятора GNU Compiler Collection, позволяя ещё на стадии сборки проекта обнаруживать места, где не выполняется обработка ошибок, что в результате улучшает стабильность кода.

4.3.4 Средства автоматизированного тестирования

Разработка архитектуры по возможности велась с учётом принципов TDD (test-driven development) [100]. Добавление нового функционала в фреймворк предварялось добавлением набора тестов, подтверждающих корректность работы нового кода в проекте. Использование методик TDD позволило лучше выстроить объектно-ориентированную архитектуру, улучшая инкапсуляцию отдельных программных компонент и в результате позволяя легче разбивать проект на независимые модули, что несомненно упростило разработку.

Был подготовлен класс ECLColDebug, который эмулирует логику прошивки модуля ECLCollector, позволяя автоматически тестировать новые версии фреймворка задолго до развёртки на целевой системе. Используя этот класс, другим экспертам по Belle II ECL удалось успешно реализовать инструменты для интеграционного тестирования, которые проверяют корректность совместной работы всех подпакетов фреймворка (обработку аргументов командной строки, загрузку конфигураций, обработку ошибок).

Чтобы отслеживать надёжность тестирования, в сборку проекта был добавлен дополнительный шаг для расчёта процента покрытия тестами с использованием программы gcov [101].

4.3.5 Внутренняя оптимизация запросов

Абстрагирование от низкоуровневых деталей процесса инициализации позволяет оптимизировать многие шаги, не внося серьёзных изменений в архитектуру программы. Помимо упомянутого выше кэширования запросов к БД, это позволяет оптимизировать работу с модулями ECL. В частности, при записи амплитудных порогов, модули ShaperDSP поддерживают возможность установить одно и то же значение для нескольких каналов. Такая же возможность предоставляется при установке аттенюаторных коэффициентов.

Низкоуровневые функции фреймворка анализируют записываемые данные и реорганизуют порядок запросов, чтобы максимально эффективно исполь-

зовать эти возможности. В результате этих улучшений, скорость инициализации увеличилась на ~30%.

4.4 Высокоуровневые утилиты инициализации

Все часто использующиеся утилиты работы с электроникой были переписаны и приведены к стандартному виду. Тривиальные проблемы, возникающие при инициализации, автоматически исправляются без вмешательства пользователя. К примеру, если соединение Belle2link было потеряно, утилита предпримет несколько попыток по восстановлению соединения и завершится с ошибкой только в случае, если все попытки оказались неудачными.

Чтобы максимально быстро возвращать калориметр в рабочее состояние, были проанализированы основные проблемы, возникающие со сбором данных, а также составлен список наиболее непонятных моментов для дежурных, пользующихся этими утилитами. Процесс исправления ошибок существенно упрощён и по возможности автоматизирован. Так, процедура перезаписи прошивки модулей ECLCollector была сокращена с 11 до 2 шагов.

Все высокоуровневые утилиты инициализации ECL также поддерживают указание отдельных групп ECLCollector по нескольким критериям:

- Для диагностики проблем с триггером ECL, удобно указывать модули ECLCollector по номеру триггерной группы.
- В случае исправления проблем с качеством данных, удобно указывать модули ECLCollector по номеру крейта.
- В случае проблем с DAQ, удобно указывать модули ECLCollector по номеру COPPER.

Таким образом, все утилиты инициализации ECL поддерживают несколько режимов маскирования, позволяя быстро исправлять возникающие проблемы. Эта функция также полезна для загрузки тестовой конфигурации в конкретный модуль ECLCollector.

Чтобы предоставить эксперту возможности для более тонкого контроля за процессом инициализации, была разработана расширенная оболочка командной строки. Это надстройка над командной оболочкой Bash, автоматически

подставляющая нужную последовательность команд в зависимости от текущего шага инициализации. В ней также поддерживается набор горячих клавиш, позволяющий переходить вручную к нужному шагу инициализации. Эксперт также может отредактировать каждую запускаемую команду или выполнить какие-либо другие команды между шагами инициализации. Командная оболочка также автоматически обнаруживает сообщения об ошибках и выдаёт советы по их исправлению.

4.4.1 Асинхронная инициализация модулей

Поскольку модули ECLCollector подключены к 26 COPPER, одним из важных требований при реализации высокоуровневых утилит инициализации ECL является координация параллельно запущенных процессов на COPPER. При этом также очень полезно минимизировать время отклика и быстро распространять сообщения от центрального сервера ко всем COPPER. Более того, как было сказано выше, система медленного контроля может быть не полностью доступна, когда требуется инициализировать ECL.

В стандартных случаях, когда система медленного контроля запущена и доступна, утилиты инициализации использует API, разработанный на основе фреймворка NSM2 и описанный в разделе 5.2.

Для второго сценария использования был разработан “малый медленный контроль”. Это простая сеть TCP серверов, которые могут быть быстро запущены по запросу. Такой запрос автоматически посылается инициализационными утилитами, если обнаружено, что система медленного контроля недоступна. Они также реализуют простую систему управления запущенными задачами с сигналом keepalive, так что при потере соединения или по отдельному запросу процесс инициализации может быть немедленно прерван.

Реализация данного сценария использования была вынесена в два программных модуля — в клиентскую и серверную части. Для реализации использовался язык Python с сетевой коммуникацией посредством протокола REST API на основе минималистичной библиотеки Falcon. Сервер необходимо запустить только на центральном узле сети и далее он автоматически подключается к указанным в конфигурации узлам по протоколу Secure Shell (SSH), запускает на них сервера второго уровня и осуществляет мониторирование стабильности соединения. Сервер также реализует систему контроля задач, позволяя фор-

мировать цепочки скриптов, выполняющихся на разных серверах в заданной очерёдности.

Для простоты развёртки клиент также реализован на Python и использует библиотеку urwid [102] для создания терминального интерфейса пользователя. Пример запущенного приложения показан на рисунке 19. Клиент также подсвечивает задачи, при исполнении которых возникли ошибки.

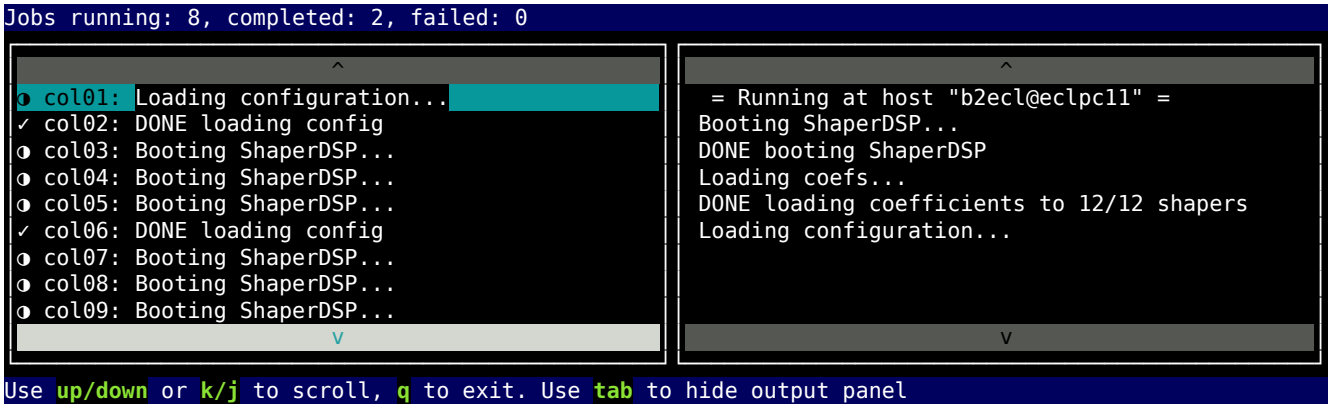

Рисунок 19 — Клиентское приложение для просмотра статуса запущенных задач

Библиотека urwid также позволяет отлавливать в терминале события мыши. Поэтому, с использованием веб-терминала наподобие Shell In A Box [103], можно предоставить дежурному довольно удобный веб-интерфейс, который можно будет при необходимости легко заменить стандартным клиентом с использованием современных средств веб-разработки.

Для быстрого тестирования также доступно консольное приложение, которое последовательно организует вывод с параллельно работающих процессов инициализации, позволяя легко интегрироваться с конвеером UNIX (UNIX pipeline).

4.5 Основные результаты главы 4

Автором было разработано ПО инициализации модулей ECLCollector и ShaperDSP. Инициализация выполняется через последовательно проводящиеся шаги, выполняющиеся по интерфейсу тестирования электроники JTAG, а также по протоколам UDP и Belle2link. Эти задачи выполняются новым фреймворком для работы с модулями ECLCollector, который позволяет абстрагиро-

ваться от используемого в данный момент интерфейса передачи данных, что также позволяет легко реализовывать автоматизированные тесты корректности процедуры инициализации.

В ходе инициализации загружается около 80000 параметров, причём инициализация надёжно работает в двух основных сценариях использования: в ситуации, когда необходима интеграция с системой медленного контроля и в изолированном режиме, который не зависит от глобальной системы DAQ.

Также были разработаны высокоуровневые утилиты инициализации, которые позволяют запускать 52 параллельных процесса на 26 удалённых серверах и отслеживать корректность их выполнения.

ГЛАВА 5. МЕДЛЕННЫЙ КОНТРОЛЬ И УПРАВЛЕНИЕ ЗАХОДАМИ

5.1 Управление доступом к Ethernet

Поскольку некоторые версии прошивки ECLCollector не могут одновременно справляться с запросами по Belle2link и по UDP, дополнительный демон управления заходами осуществляет мониторинг трафика Belle2link и, при необходимости, блокирует соединение Ethernet к модулям ECLCollector.

Модули ECLCollector разделены на 6 отдельных триггерных групп, доступ к каждой из которых предоставляется через отдельный сетевой свич HP1810. Хотя эти свичи и предоставляют функцию блокировки трафика на указанные узлы, она может быть активирована только через веб-интерфейс, требуя отдельный запрос на каждый из 52 коллекторов.

Чтобы автоматизировать этот процесс, был реализован программный модуль на языке Python, использующий библиотеку Selenium [104]. Когда запускается демон управления заходами, он использует этот модуль для инициализации соединения к веб-интерфейсу 6 свичей HP1810 и автоматически устанавливает настройки VLAN, таким образом блокируя или пропуская сетевой трафик к указанным ECLCollector.

5.2 Процессы медленного контроля

На каждом COPPER запущен один процесс NSM2, интегрированный в систему управления заходами. Он устанавливает конфигурацию для модулей ECLCollector и ShaperDSP, а также подстраивает некоторые элементы конфигурации на основе текущего типа захода и типа триггера, чтобы избежать высоких значений мёртвого времени. В частности, при необходимости повышается/понижается частота сохранения данных формы сигнала.

Этот процесс также предоставляет API для прямого доступа к регистрам модулей ECLCollector через протокол, используемый NSM2. В отличие от стандартных операций с переменными NSM2, этот API может использоваться таким образом, чтобы в рамках одного запроса послать очерёдность команд и пошагово получать статус их исполнения в качестве ответа.

5.3 Программная библиотека pyNSM2

Фреймворк NSM2, использующийся в системе медленного контроля эксперимента Belle II, целиком реализован на C, что обеспечивает высокое быстродействие, но порой увеличивает время, требуемое на разработку новых программ. Поэтому, для быстрой разработки прототипов, возникла необходимость разработать библиотеку pyNSM2, предоставляющую доступ к возможностям NSM2 на языке Python.

Чтобы упростить поддержку кода и улучшить быстродействие, в pyNSM2 было решено обращаться к низкоуровневым функциям NSM2, написанным на языке C, а не переписывать их на чистом Python. Для реализации взаимодействия между C и Python было рассмотрено несколько вариантов: ctypes [105], Python C API [106] и SWIG [107]. Все три варианта требуют спецификации интерфейса между C и Python, поэтому при крупных обновлениях библиотеки NSM2 также требуется модифицировать pyNSM2. Наиболее простой формат интерфейса оказался у библиотеки ctypes, поэтому его генерацию удалось автоматизировать с помощью отдельного шага сборки, выполняемого модулем pycparser. Этот модуль анализирует исходный код NSM2 и, следуя заданным правилам анализа синтаксиса языка C, генерирует выходной файл на языке Python. Таким образом, при изменении NSM2 требуется модифицировать только высокоуровневые функции pyNSM2.

Более того, ключевая возможность NSM2 — использование разделяемой памяти — полностью интегрирована в Python, позволяя напрямую доступаться к структурам, хранящимся в разделяемой памяти, с минимальными потерями быстродействия. Для разработки полноценных приложений медленного контроля также разработан модуль, конвертирующий функции Python в указатели на функции языка C, позволяя легко встраиваться в событийно-ориентированную архитектуру NSM2.

На данный момент модуль pyNSM2 добавлен в репозиторий с ПО для медленного контроля и активно используется экспертами нескольких подсистем детектора Belle II. В частности, в рамках данной работы в качестве примера возможностей pyNSM2 был реализован модуль rcview — TUI на базе модуля curses для быстрой диагностики текущего статуса захода, показанный на рисунке 20. В стандартной конфигурации, rcview отображает текущий статус захода, поток данных до каждого из серверов HLT и статус высоковольтных источников

питания каждой подсистемы детектора.

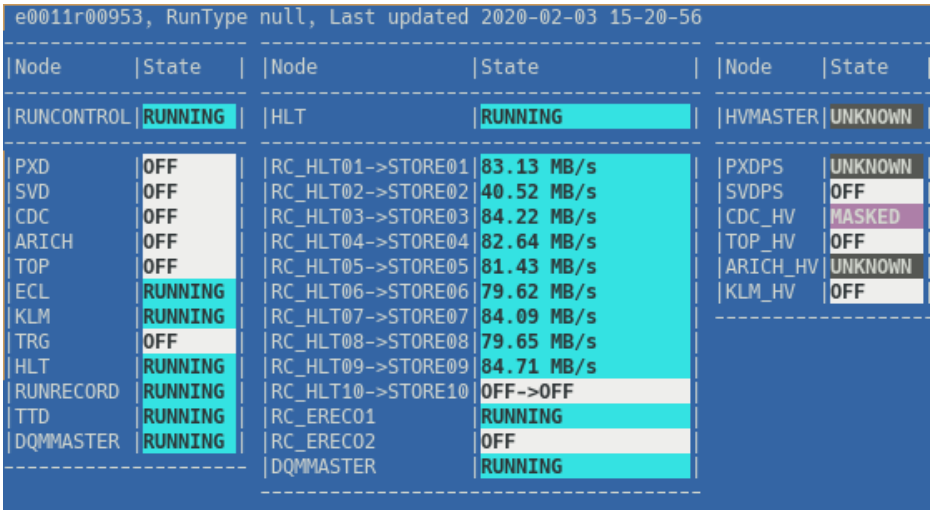

Рисунок 20 — Скриншот модуля rcview

В дополнение к pyNSM2, в репозиторий с ПО для медленного контроля был добавлен небольшой модуль Python, который реализует такие базовые функции как логирование (в таком же формате, какой используют приложения на C++) и чтение внутренних конфигурационных файлов. Также был добавлен модуль для работы с DAQ DB, описанный в разделе 3.1.

5.3.1 Система оповещений о статусе DAQ

За стабильностью набора данных в эксперименте Belle II круглосуточно наблюдает большое число экспертов по всем подсистемам детектора. Чтобы эффективно координировать их деятельность, важна общая система оповещений, предоставляющая краткую информацию о статусе набора данных. При этом наиболее удобно скомбинировать её с RocketChat, основным каналом коммуникации дежурных экспертов, описанным в разделе 1.7.5.

С учётом этих требования, на языке Python был разработан программный модуль SALS Agent (Stop, Abort, Load, Start; четыре основных действия при перезапуске захода). Он использует событийно-ориентированный функционал библиотеки pyNSM2 для отправления сообщений в RocketChat при изменении статуса захода. Пример работы SALS Agent показан на рисунке 21.

Кроме того, SALS Agent использует RocketChat API для автоматического удаления сообщений, которые не имеет смысла хранить в истории чата, оповещает о некоторых стандартных ошибках в системе DAQ и об изменении статуса

высоковольтных источников питания. По завершению захода, он предоставляет ссылку на веб-страницу одного из центральных серверов Belle II DAQ с детальной информацией о набранном заходе.

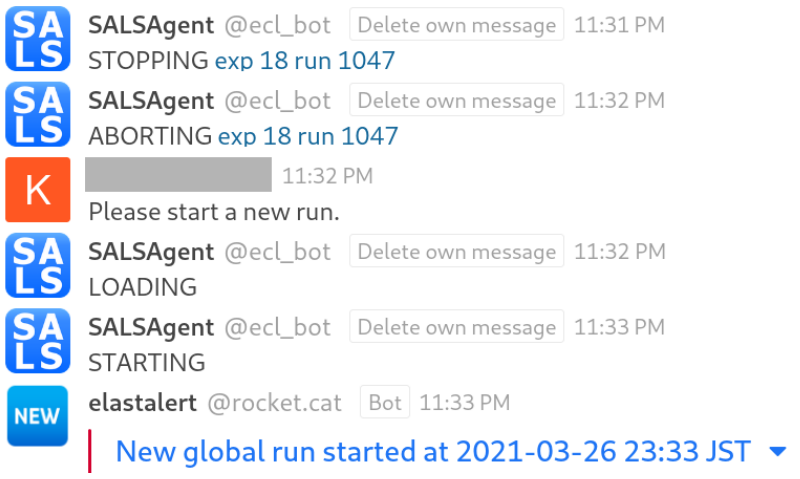

Рисунок 21 — Пример работы SALS Agent в RocketChat

Исходный код SALS Agent передан центральной группе DAQ, которая использует его для разработки ПО, автоматически исправляющего стандартные проблемы с системой сбора данных и перезапускающего заходы без вмешательства дежурного.

5.3.2 Средства автоматизированного тестирования

Хотя срок поддержки Python 2.7 закончился в 2020 году, на данный момент эта версия Python всё ещё является единственно доступной на большинстве серверов, использующихся в сети Belle II DAQ. Поэтому, чтобы одновременно упростить развёртку на существующих серверах и облегчить дальнейший переход на Python 3, при разработке pyNSM2 было решено поддерживать обе версии языка Python.

Большинство тестов были объединены в единый процесс интеграционного тестирования, когда запускается два приложения, клиент и сервер, взаимодействующих через pyNSM2. В ходе тестирования проверяются 4 возможные комбинации версий Python (2.7/3 на клиенте, 2.7/3 на сервере).

Также было реализовано несколько скриптов для модульного тестирования отдельных частей pyNSM2. Помимо своего основного назначения, эти скрипты используются как вспомогательные примеры, дополняющие докумен-

тацию программной библиотеки.

5.4 Координация деятельности дежурных

Как было описано в разделе 1.7.5, имя текущего дежурного должно отображаться сразу в трёх местах — ShifTool, RocketChat и Confluence. В рамках данной работы был реализован программный модуль на языке Python, который автоматически синхронизует расписание дежурных между этими тремя сервисами. На основе libxml [56] была разработана специальная библиотека для работы с Confluence, позволяющая использовать запросы XPath для модификации отдельных частей страницы с расписанием дежурных.

5.5 Графический интерфейс управления заходами

Для запуска калибровочных заходов и быстрого определения источника проблем в системе медленного контроля требуется иметь графический интерфейс пользователя (GUI), собирающий информацию с большого числа серверов DAQ. В рамках данной работы был расширен стандартный GUI управления заходами, использующийся в эксперименте Belle II. Конечный результат показан на рисунке 22.

Список добавлений, специфичных для ECL:

- Процедура установки конфигурации была оптимизирована, ПО автоматически проверяет, что все модули ECLCollector используют одну и ту же конфигурацию из DAQ DB.
- Для выбранных узлов DAQ можно отобразить логи в отдельном окне.
- Недавние локальные заходы ECL и их статус обработки отображаются на отдельной панели.
- Отображается текущая информация с монитора светимости.

При запуске ПО активирует TCP-сервер, предоставляющий API для доступа к большинству элементов интерфейса, позволяя управлять элементами GUI через консоль, что довольно часто актуально при удалённом SSH-подключении к компьютеру в пультовой.

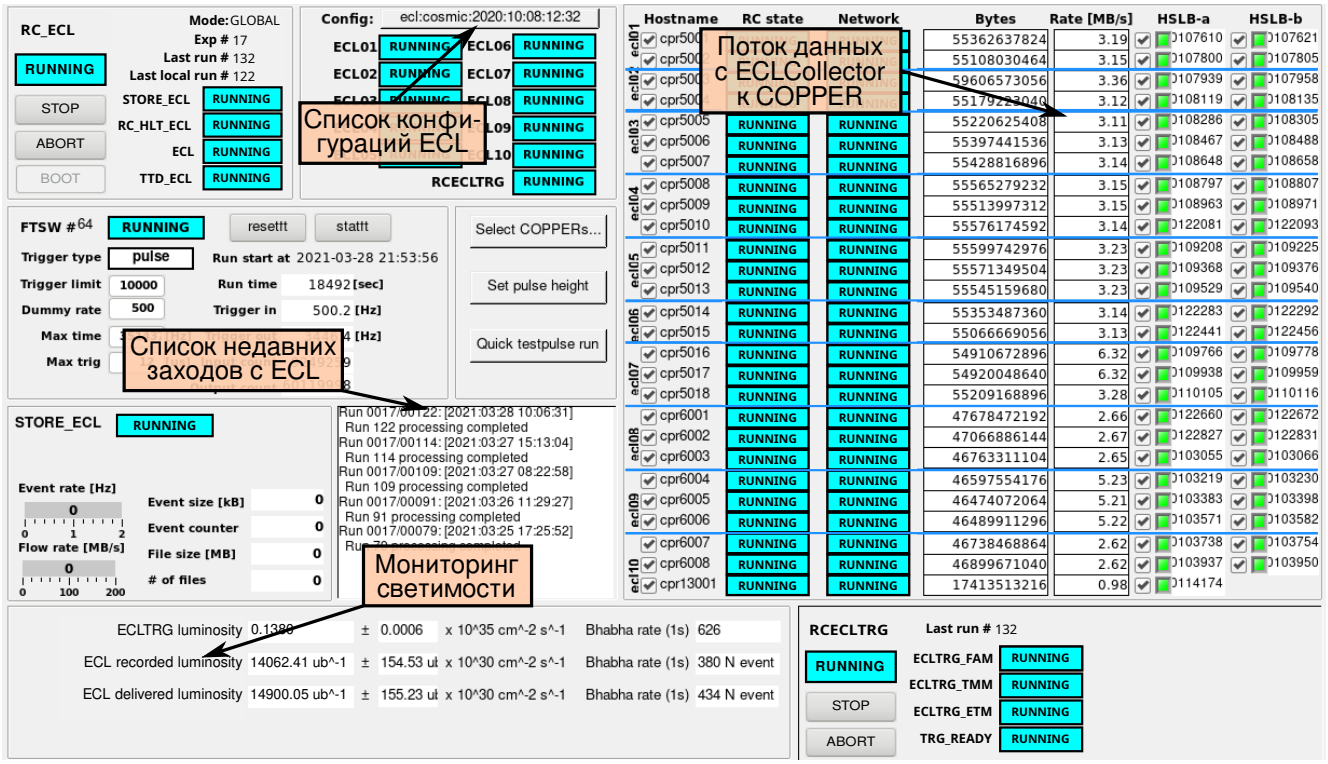

Рисунок 22 — Интерфейс запуска локальных заходов с ЭМ калориметром [10]

Использование этого TCP-сервера позволило автоматизировать несколько калибровочных процедур и разработать прототипы некоторых виджетов, позже добавленных в графический интерфейс.

Кроме того, при запуске калибровок, ПО для управления заходами ECL обновляет внутреннюю базу данных SQLite [108], добавляя в неё информацию о статусе DAQ, комбинированную со статусом ускорителя и информацией о текущей конфигурации электроники. Эта информация затем переносится на дисковое хранилище вычислительного центра KEK и сохраняется вместе с результатами обработки калибровочных данных. Было разработано несколько приложений, которые используют эту БД для отображения детальной истории калибровок по тестовому сигналу и поиска тестовых заходов с определённой конфигурацией.

Все расширения GUI управления заходами были разработаны на языке Jython [67].

5.5.1 Отображение логов

Чтобы предоставлять доступ к более детальной информации о статусе системы медленного контроля, в GUI необходимо отображать логи Belle II DAQ из базы данных PostgreSQL.

Для этого, фреймворк Control System Studio использует Jython (гибрид Java и Python). Из-за ряда особенностей интерпретатора Jython, довольно сложно обеспечить переносимость кода, работающего с базами данных PostgreSQL. Наиболее популярный модуль, psycopg2, использует скомпилированный код, зависящий от архитектуры процессора. Другой подходящий модуль, pg8000, использует библиотеку urllib2, имеющую проблемы совместимости с Jython.

Был разработан небольшой патч для библиотеки pg8000, обеспечивающий совместимость с Jython, разработан виджет, отображающий логи выбранных компонент DAQ.

5.6 Автоматизация калибровки по тестовому сигналу

Как было упомянуто в разделе 1.5, калибровка по тестовому сигналу должна проводиться максимально быстро. Для этого почти все действия, выполняющиеся при калибровке, были автоматизированы.

Когда набор калибровочных данных завершается, они автоматически переносятся на сервер вычислительного центра и незамедлительно начинают обрабатываться. Статус обработки данных отображается в GUI управления заходами.

Результаты калибровки далее переносятся на веб-сервер, где дежурный может построить графики, отображающие амплитудную и временную стабильность каждого канала ECL относительно заданных опорных значений. Визуализация осуществляется с помощью высокопроизводительной библиотеки Dygraphs и показана на рисунке 23. Приложение использует карту каналов ECL для отображения детальной информации о каждом канале калориметра (номер модуля ECLCollector, номер платы ShaperDSP).

Серверная часть реализована на языке Python, взаимодействие между клиентской и серверной частями выполняется через REST API. Через API можно получать как сырые данные JSON, так и заранее нарисованные графики в формате PNG. Сервер запускает несколько отдельных рабочих процессов, которые параллельно выполняют загрузку данных для указанных заходов. Кроме того, реализовано кэширование запросов и интеграция с сервером RocketChat, позволяющая в один клик отправлять информацию о результате калибровки.

Помимо веб-приложения, реализовано несколько консольных утилит, позволяющих автоматически отправить запрос на обновление калибровочных кон-

стант в ConditionsDB, в случае, если для отдельных каналов слишком сильно сдвинулись время и/или амплитуда.

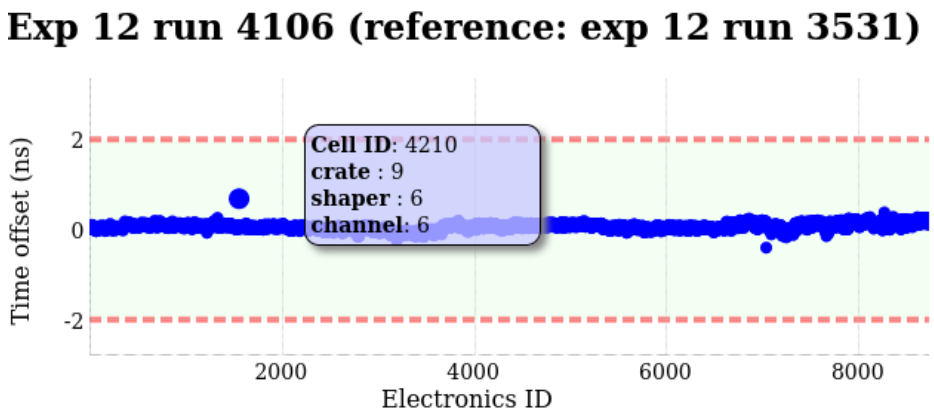

Рисунок 23 — Временной сдвиг каждого канала относительно опорного значения

5.6.1 Анализ долговременной стабильности электроники

Используя информацию из БД калибровочных заходов, эксперты по ECL могут следить, как меняется статус считывающей электроники с течением времени. Для этого в веб-приложения реализована функция отображения графиков временной и амплитудной стабильности указанного канала. Пример одного из таких графиков показан на рисунке 24.

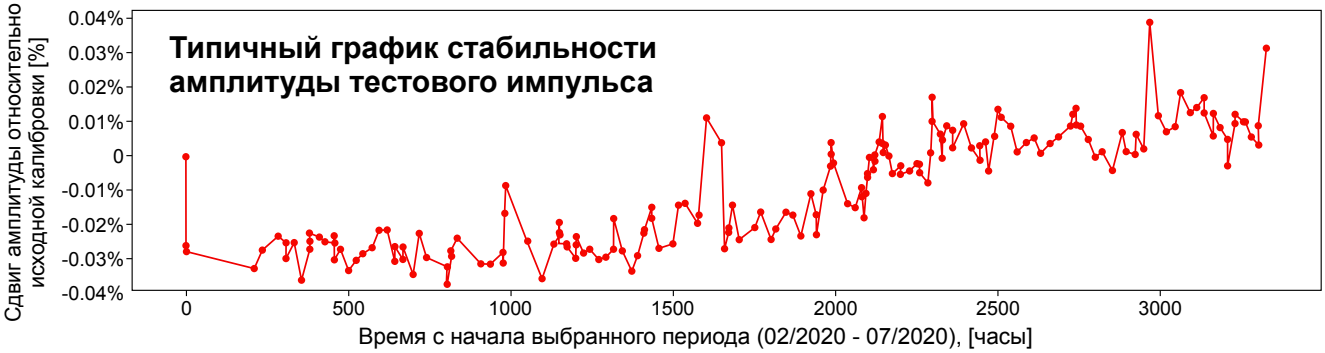

Рисунок 24 — Сдвиг амплитуды относительно исходной калибровки как функция времени. Рост амплитуды объясняется понижением температуры ECL на 1°C

5.7 Веб-сервер экспертов по ECL

Как было упомянуто в разделе 2.3, необходимо предоставить интерфейс для удалённого доступа к инструментам эксперта по ECL. Поэтому внутри сети сбора данных был создан Apache сервер с веб-приложениями для доступа к различным утилитам для управления ECL, как показано на рисунке 25.

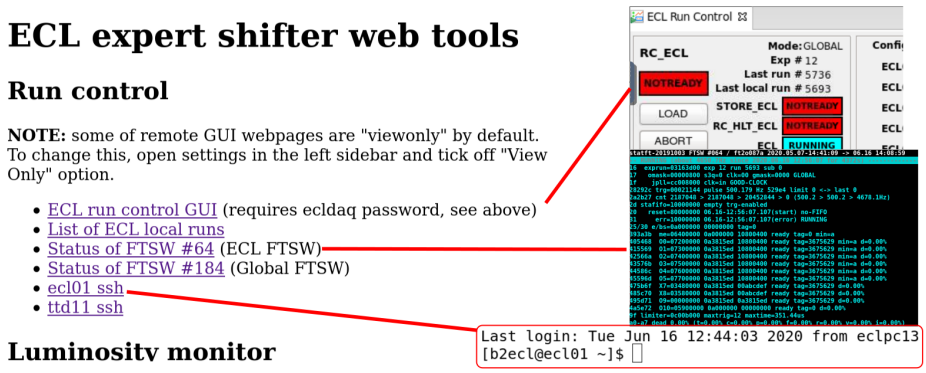

Рисунок 25 — Веб-страницы, использующиеся дежурными

Для быстрого доступа ко всем основным консольным утилитам в веб-сервер был добавлен встроенный эмулятор терминала на основе проекта Shell In A Box [103].

Для доступа к графическим приложениям используется протокол VNC и веб-клиент noVNC [109], встроенный в веб-сервер.

Описанное выше веб-приложение для визуализации данных калибровки по тестовому сигналу также интегрировано с этим сервером.

Кроме того, этот веб-сервер исполняет роль обратного прокси, позволяя дежурному создать всего один SSH-туннель до сервера экспертов по ECL и в результате получить безопасный доступ ко всем остальным серверам, находящимся в сети сбора данных. В результате, на машине пользователя требуются только клиенты Virtual Private Network (VPN), SSH и браузер, поддерживающий 5-ую версию HyperText Markup Language (HTML5).

Для пользователей, находящихся в ИЯФ, подготовлен отдельный промежуточный сервер. На нём через ocproxy [110] проведено VPN-соединение к веб-серверу экспертов по ECL. В результате, на клиентской машине должны быть только SSH и браузер, что дополнительно упрощает работу экспертов.

5.7.1 Встроенный оконный менеджер

В ходе дежурства используется большое количество веб-приложений, которые предоставляют разностороннюю информацию о состоянии калориметра, — гистограммы качества данных, монитор статуса ECL DAQ, сервис RocketChat. Чтобы более компактно организовать эту информацию, для ECL экспертов был подготовлен простой оконный менеджер, встроенный в веб-сервер.

Для этого используется стандартный HTML-элемент iframe, позволяющий внутри одной веб-страницы показывать несколько других веб-страниц. Его использование позволяет совместно отображать произвольное число инструментов эксперта, без необходимости открывать для каждого из них отдельную вкладку или окно браузера.

Эти iframe были интегрированы с оконным менеджером Simone [111], разработанным на Javascript. Его использование позволяет свободно перемещать отдельные iframe и более эффективно использовать экранное пространство, как показано на рисунке 26.

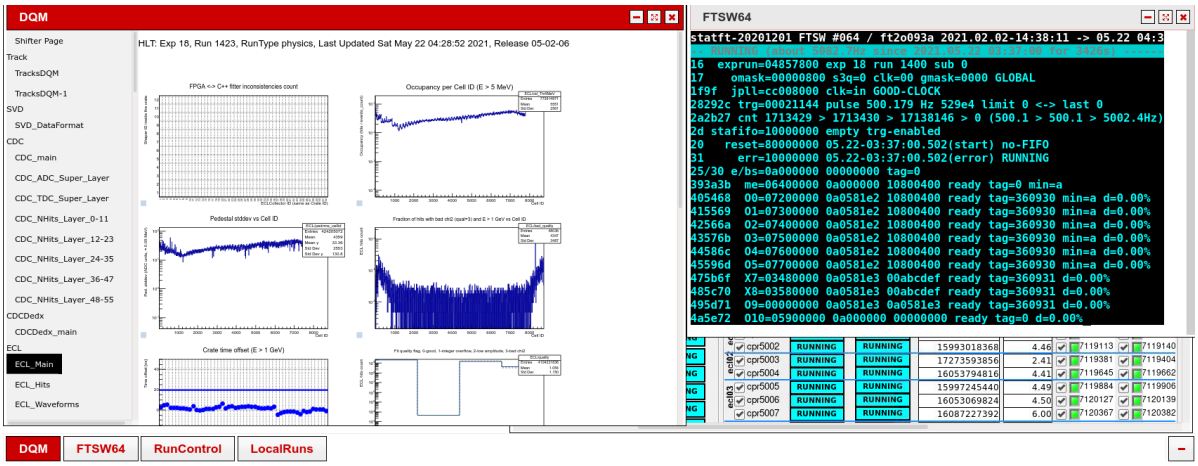

Рисунок 26 — Веб-интерфейс, комбинирующий несколько приложений во встроенном оконном менеджере

5.8 Документация для дежурного

Удалённые дежурства по калориметру выполняются десятками экспертов из разных точек мира, ежедневно и круглосуточно. Регулярно проводятся онлайн-собрания для обучения новых экспертов.

Для организации их работы крайне важно иметь детальную документацию по всем аспектам дежурств по калориметру. Документация многократно перерабатывалась и расширялась с учётом отзывов дежурных экспертов.

Руководство разделено на четыре основных секции, посвящённых средствам коммуникации, мониторингу качества данных, диагностике проблем и исправлению проблем с калориметром. Информация в этих секциях разбита на слайды, которые используются в обучении дежурных. В руководстве также содержится множество ссылок на Confluence с более детальным описанием упоминаемых процедур.

5.9 Основные результаты главы 5

Для того, чтобы централизованно управлять конфигурациями электроники ECL, запускать инициализацию и предотвращать сбои, автором был разработан ряд программных модулей, интегрирующихся с системой медленного контроля эксперимента Belle II.

Реализованы программные интерфейсы для управления сетевыми коммутаторами и модулями ECLCollector, для ускорения разработки была создана программная библиотека pyNSM2, которая позволяет писать узлы сети медленного контроля на языке Python.

Разработан большой набор инструментов, использующихся дежурными для отслеживания статуса ECL и позволяющих, совместно с системой мониторинга, оперативно реагировать на сбои в процессе сбора данных. Для дежурных подготовлена детальная документация.

ГЛАВА 6. МОНИТОРИНГ КАЧЕСТВА ДАННЫХ

6.1 Модули монитора качества данных (DQM)

Переменные, которые отслеживаются в мониторе качества данных ECL (ECL DQM), могут быть разделены на три основные группы гистограмм по их назначению. **Первая группа** включает гистограммы, которые отображают низкоуровневую информацию для обнаружения нарушений в целостности данных.

К примеру, к гистограммам первой группы относятся гистограммы загрузки каналов, которые используются для определения шумящих и/или не работающих каналов. В эту категорию также включены гистограммы, отслеживающие средние значения и ширины пьедесталов для каждого канала ECL. В качестве примера, одна из таких гистограмм приведена на рисунке 27.

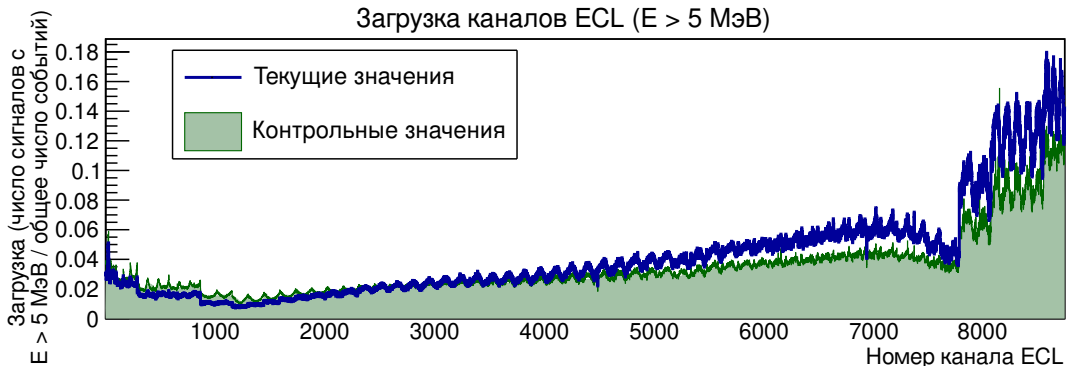

Рисунок 27 — Загрузка каналов ECL с порогом E $>$ 5 МэВ. Гистограмма с информацией для текущего захода наложена поверх контрольных данных

Также ECL DQM предоставляет гистограммы для проверки надёжности внешней информации от DAQ. В частности, данные от ECL, вместе с информацией о временном распределении триггера, могут быть использованы для подстройки параметров вето инжекции. Также добавлены несколько гистограмм для мониторирования пучкового фона.

Ко **второй группе** относятся гистограммы, отображающие калибруемые величины. В частности, к этой группе относятся гистограммы распределения времени сигнала и энергетического распределения в калориметре. По ним можно быстро обнаруживать проблемы с системой триггера и отслеживать пики в энергетическом распределении, относящимся к событиям e^+e^--рассеяния. Вре-

менное распределение для сигналов в ECL показано на рисунке 28. Пик в этом распределении соответствует срабатываниям от e^+e^--взаимодействия, подложка — пучковому фону.

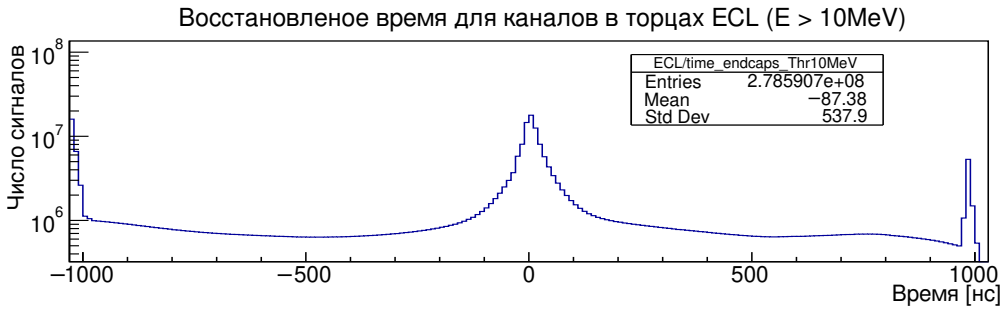

Рисунок 28 — Временное распределение в торцах ECL для сигналов с E > 10 МэВ

Алгоритм обработки сигналов, реализованный в модулях ShaperDSP, реконструирует время начала сигнала и амплитуду по данным АЦП, как описано в секции 1.3.3. Система ECL DQM может быть использована, чтобы в реальном времени мониторировать корректность работы этого алгоритма в модулях ShaperDSP. Для этого предназначена **третья группа** гистограмм.

Сохранённые данные формы сигнала обрабатываются программным модулем, реализованным на языке C++ для эмуляции алгоритма реконструкции параметров сигнала, использующегося в ПЛИС модуля ShaperDSP. Концептуальная схема описанной процедуры показана на рисунке 29. Вычисленные в

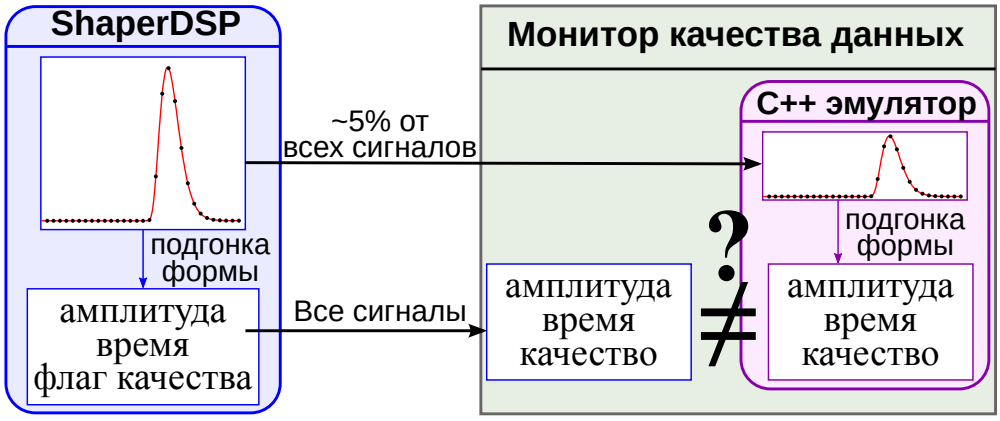

Рисунок 29 — Схема процедуры для проверки алгоритма реконструкции параметров сигнала

эмуляторе время и амплитуда сравниваются с аналогичными величинами из

электроники, затем, если были обнаружены расхождения, их число отображается в ячейке DQM гистограммы, соответствующей проблемному модулю.

Эмулятор использует DSP коэффициенты и значения амплитудных порогов, загружая их из ConditionsDB. Как было упомянуто в разделе 3.4, эти величины синхронизуются с DAQ DB, чтобы гарантировать использование одной и той же версии параметров как в эмуляторе, так и в электронике. По этой причине, для стандартной конфигурации число расхождений равно нулю.

6.2 Веб-интерфейс DQM

Как было упомянуто в разделе 1.7.4, DQM-гистограммы заполняются в реальном времени и отображаются на веб-сервере JSROOT. В рамках данной работы было внесено несколько добавлений в часть веб-сервера, относящуюся к ECL.

При наведении в гистограмме на бин, относящийся к одному из каналов калориметра, выводится всплывающая подсказка с информацией из карты каналов, описанной в разделе 3.3.1. Таким образом можно быстро определить, в каком модуле ECL возникла проблема. Пример всплывающей подсказки показан на рисунке 30.

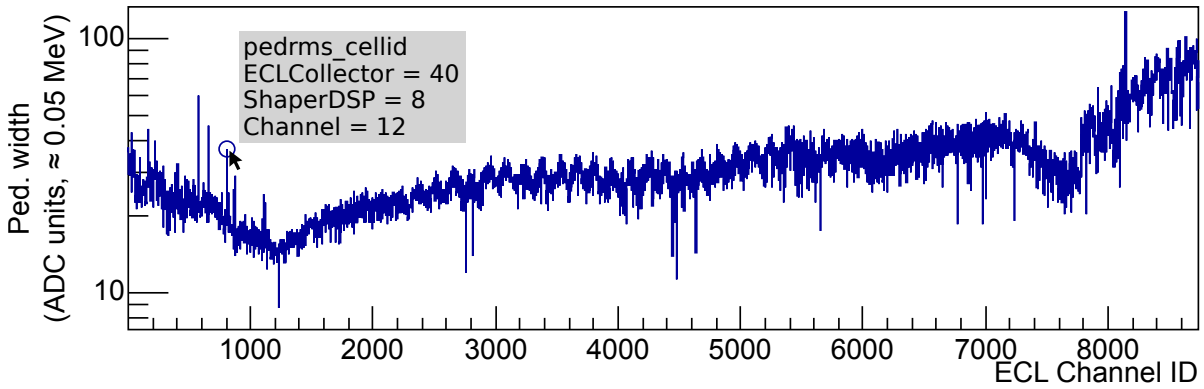

Рисунок 30 — Пример всплывающей подсказки с информацией из карты каналов

Также вместо жёсткой расстановки гистограмм по сетке, был использован адаптивный стиль, размещающий гистограммы в несколько колонок, число которых зависит от ширины окна браузера. Это изменение позволило более эффективно использовать экранное пространство.

6.3 Мониторинг фона инжекции

Детальные исследования фона инжекции показали, что высокий уровень фона при инжекции позитронов в low-energy ring (LER) может привести к ухудшению энергетического и временного разрешения ECL.

В частности, индикатором большого фона инжекции является наличие второго пика сигнала, расположенного в области пьедестала, как показано на рисунке 31. Детальные исследования продемонстрировали, что положение этого пика соответствует времени прохождения инжектированного банча. Подгонка такой формы приведёт к существенному завышению пьедестала и, следовательно, к систематическому занижению реконструированной амплитуды сигнала.

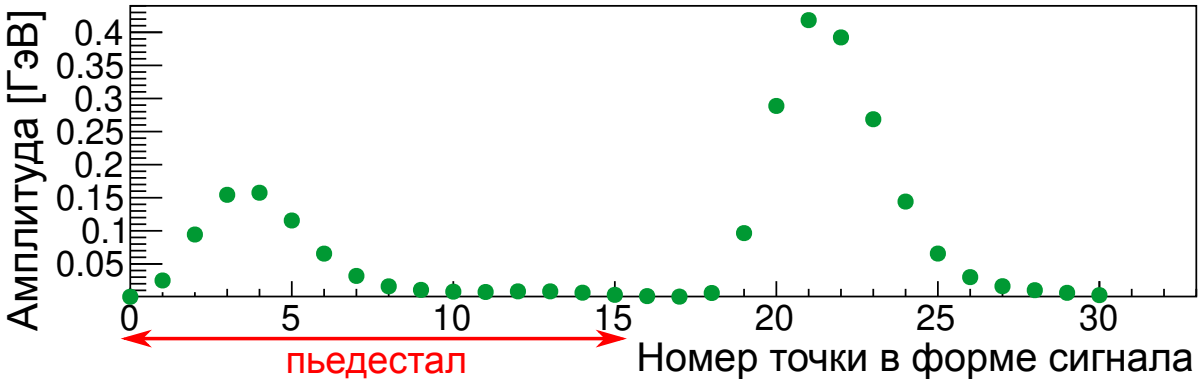

Рисунок 31 — Форма сигнала с пиком, относящимся к фону инжекции

При обработке в ECL DQM, форма сигнала для событий после инжекции обрабатывается модифицированным алгоритмом подгонки сигнала с двумя пиками и на основе полученных данных строится распределение по энергии пика в области пьедестала сигнала для разных интервалов времени после инжекции, что показано на рисунке 32.

6.4 DQM для интервала заходов

В задаче отслеживания качества данных зачастую полезно пронаблюдать зависимость мониторируемых величин от номера захода. С этой целью, эксперты из команды Belle II разработали программу, использующую библиотеку ROOT для постобработки DQM-гистограмм.

Эта программа была интегрирована в существующую систему медленного контроля. При остановке захода, она автоматически обрабатывает сохранённые

DQM-гистограммы, вычисляя интегральные величины, такие как, к примеру, доля сохранённых форм сигнала. Полученные результаты заносятся в локальную базу данных для дальнейшего анализа.

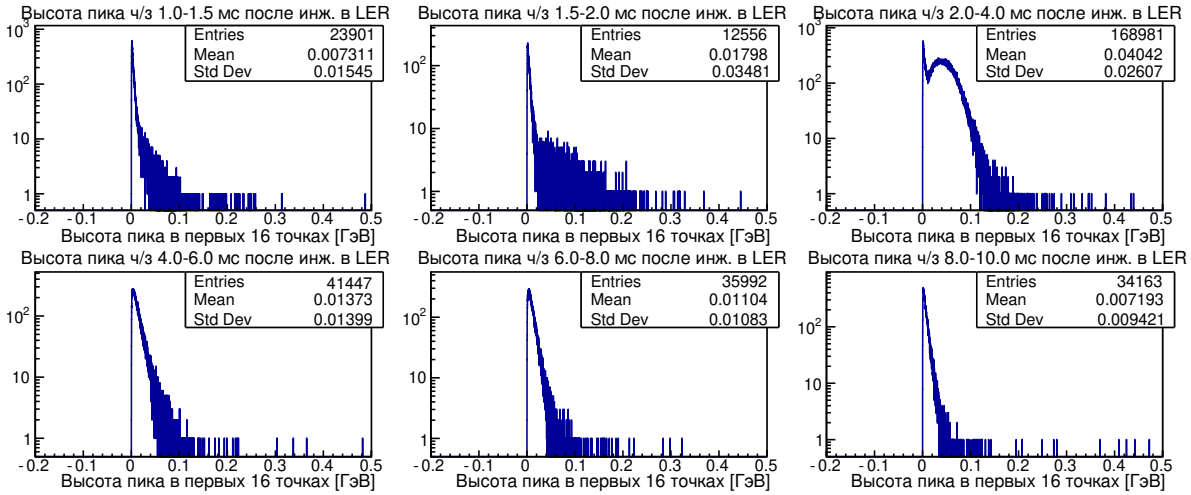

Рисунок 32 — Энергетическое распределение фона от инжекции в LER

Для упрощения доступа дежурных к этим данным, они отправляются на внешний веб-сервер, где эксперты могут строить графики по вычисленным величинам от номера захода. Чтобы автоматизировать передачу данных, был разработан простой API с авторизацией по токену. Каждый токен имеет свой набор разрешённых операций, который может быть в любой момент модифицирован на веб-сервере. Такая реализация позволяет минимизировать риски, связанные с нарушением информационной безопасности — злоумышленник может использовать токен только для ограниченного набора запросов, кроме того скомпрометированный токен легко можно отозвать.

Для отображения интегральных величин был адаптирован веб-интерфейс MiraBelle, который используется для анализа поведения заданных параметров на произвольном интервале заходов. Пример графика, построенного этим веб-интерфейсом по данным ECL показан на рисунке 33.

6.5 Взаимодействие с JSROOT, интеграция с EPICS

Чтобы более оперативно обнаруживать проблемы с качеством данных, была реализована программа, которая автоматически скачивает необходимые DQM-гистограммы и анализирует их на наличие проблем. Поскольку веб-сервер, где эксперты могут смотреть DQM-гистограммы, использует JSROOT, эта про-

грамма может работать как отдельный процесс, периодически запрашивающий данные с веб-сервера. Для десериализации данных используется класс TBufferJSON библиотеки ROOT.

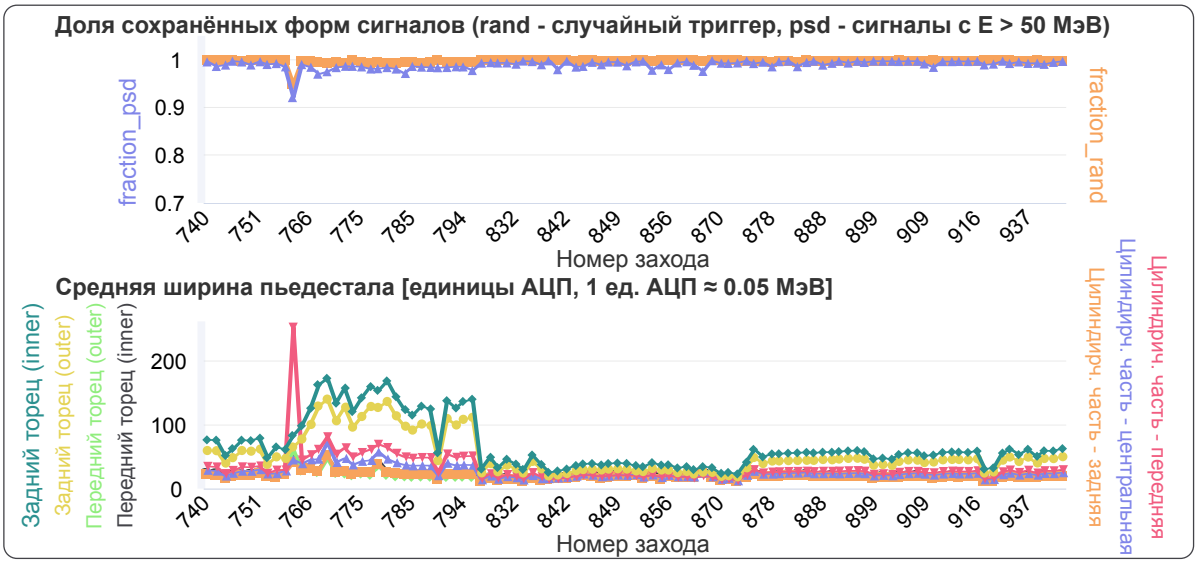

Рисунок 33 — Зависимость доли сохранённых форм сигнала и ширины пьедестала от номера захода

Было реализовано несколько узлов медленного контроля, которые автоматически оповещают дежурных в случае обнаружения проблем в DQM, прилагая к сообщению список необходимых действиях для её исправления. На практике использование таких оповещений позволило существенно ускорить время реагирования эксперта.

Разработанные программы также периодически обрабатывают полученные гистограммы и отправляют интегральные данные в EPICS, которые могут наблюдаться в других местах, например на пультовой ускорителя. В частности, отображается информация о текущем уровне фона.

Для этого совместного с группой ECL триггера был подготовлен узел системы медленного контроля, следящий, чтобы частота срабатываний триггерных ячеек калориметра не превышала определённый порог. Для определения порогов были проанализированы заходы для исследования пучкового фона из второй фазы эксперимента. Полученные значения загружены в DAQ DB и могут при необходимости подстраиваться экспертами.

Помимо использования дежурными, этот узел планируется интегрировать с системой автоматического подавления инжекции. Это позволит избежать ухуд-

шения энергетического и временного разрешения калориметра при высоком уровне пучкового фона.

6.6 Архивация данных мониторинга

Для того чтобы восстанавливать детальные условия, при которых запускался каждый из набранных заходов, необходимо архивировать текущие значения параметров электроники и конфигурации DAQ. В эксперименте Belle II для этого активно используется EPICS Archiver [112], позволяющий сохранять и сопоставлять историю множества величин, доступных в системе медленного контроля.

В частности, в подсистему управления заходами для ECL было добавлено несколько опций для экспорта параметров электроники, используя те же категории, что описаны в 3.2.

1. Статические параметры параллельно вычитываются со всех модулей ECL, проверяются на согласованность и передаются в EPICS Archiver.

2. Параметры, основанные на калибровке, преобразуются в объект ROOT и загружаются в ConditionsDB.

3. Для DSP-коэффициентов архивируется их номер версии.

6.7 Автоматическое оповещение дежурных

Чтобы максимально быстро обнаруживать и устранять проблемы с ECL, необходимо иметь систему узлов медленного контроля, которая способна быстро оповещать дежурного о необходимых действиях при обнаружении неполадок с качеством данных или с ECL DAQ.

Добавлены утилиты для автоматического отслеживания ряда проблем, с автооповещением по email и RocketChat. Пример оповещения по email показан на рисунке 34. На данный момент дежурному приходят оповещения по пяти категориям событий:

1. Проблемы с монитором светимости.

2. Некорректные данные на уровне распаковщика событий.

3. Проблемы с реконструкцией параметров сигнала в ПЛИС ShaperDSP.
4. Нарушение сетевой доступности серверов, относящихся к ECL.
5. Переполнение дискового пространства на этих серверах.

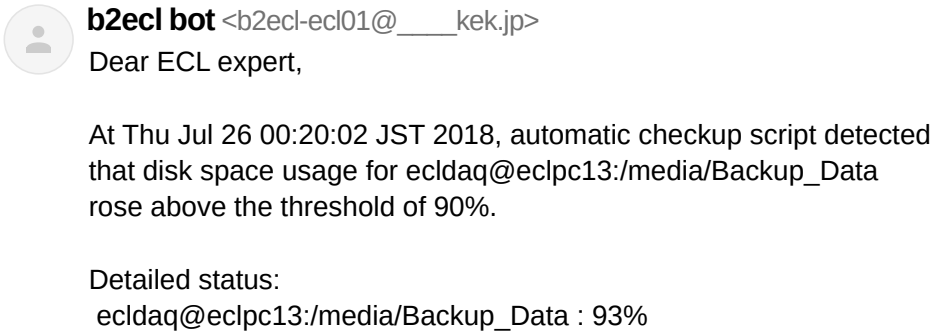

Рисунок 34 — Пример оповещения

В случае обнаружения некорректных данных из распаковщика событий дежурному также приходит скриншот DQM-гистограмм, которые позволяют идентифицировать, в каком секторе калориметра возникла проблема.

Каждое письмо содержит информацию об исправлении проблемы, со ссылками на документацию для дежурного по калориметру.

Настройки сервиса оповещения дежурных задаются через несколько конфигурационных файлов, которые позволяют отключать определённые виды оповещений и настраивать группы пользователей, получающих письма о той или иной проблеме. Сервис позволяет сохранять историю отосланных оповещений и проводить повторную отсылку сообщений о той же проблеме с заданным временным периодом.

6.8 Основные результаты главы 6

Автором были расширены средства мониторинга данных с ECL, разработаны средства автоматического оповещения об ошибках. Усовершенствован существующий веб-интерфейс, отображающий гистограммы мониторинга качества данных.

В частности, были реализованы и добавлены модули для мониторинга фона инжекции, для отслеживания динамики наблюдаемых параметров от захода

к заходу. Данные мониторинга непрерывно экспортируются в EPICS и архивируются в EPICS Archiver.

В случае выхода отслеживаемых значений из заданного диапазона, дежурные получают оповещение по email и в веб-приложении RocketChat.

ГЛАВА 7. СЧИТЫВАНИЕ ДАННЫХ С МОНИТОРА СВЕТИМОСТИ

7.1 Программная архитектура системы сбора данных

Измерение светимости в режиме реального времени является исключительно важной задачей как для детектора, так и для ускорительного комплекса. Поскольку актуальные данные светимости должны определяться вне зависимости от статуса захода и состояния системы триггера детектора, была разработана отдельная система сбора данных для онлайн-монитора светимости (LOM, Luminosity Online Monitor). Разработанное программное обеспечение LOM предоставляет данные в обе системы медленного контроля (EPICS и NSM2), сохраняет детальную информацию для последующего анализа и предоставляет утилиты для управления LOM. Блок-схема разработанного ПО представлена на рисунке 35.

Рисунок 35 — ПО для считывания данных с монитора светимости

Для считывания данных была разработана программная библиотека LOM Interface, поддерживающая все основные команды, реализованные в прошивке LOM. Чтобы лучше интегрироваться с другими проектами, эта библиотека была реализована как на языке программирования C++, так и на Python. В частности, эту библиотеку использует программа для считывания данных (readout software). Она является центральным компонентом ПО монитора светимости и выполняет следующие функции:

1. Считывание данных с монитора светимости с частотой 1 Гц.

2. Загрузка актуальной конфигурации из DAQ DB (калибровочные коэффициенты LOM).

3. Коррекция величины светимости в зависимости от энергии пучков.

4. Расчёт интегральной светимости и максимальной светимости за заданные периоды времени.

5. Запись данных на диск.

6. Выполнение роли прокси: обработка косвенных запросов к монитору светимости, а также кэширование этих запросов.

7. Экспорт информации о светимости в EPICS с использованием библиотеки PythonIOC [113].

Реализация прокси к LOM была необходима по нескольким причинам. Во-первых, текущая версия прошивки монитора светимости не является потокобезопасной, поэтому считывающая программа должна устанавливать очерёдность обработки запросов к LOM. Во-вторых, такая реализация позволяет кэшировать запросы к монитору светимости (при необходимости также доступен набор инструкций, которые читают данные напрямую, избегая кэш). В-третьих, это позволяет доступаться к данным монитора светимости из любого места сети DAQ.

Для работы с прокси не требуется дополнительных библиотек — библиотека LOM Interface одинаково работает как при взаимодействии непосредственно с LOM, так и при отправке запросов через прокси. С программой считывания данных взаимодействует несколько приложений (LOM data quality monitor, процесс NSM2, консоль LOM), которые будут описаны далее.

Считывающее ПО является многопоточным приложением, написанным на языке Python. Первый поток отвечает за чтение данных, остальные потоки создаются по необходимости, чтобы отвечать на запросы подключающихся клиентов. Поскольку существенная доля времени работы программы отводится на операции ввода/вывода, удаётся избежать проблем с Python GIL, описанных в разделе 2.5.1.

Описанная система успешно работает с начала второй фазы эксперимента Belle II (первые тесты совместной работы ускорителя и детектора, начавшиеся в марте 2018 года).

7.2 Мониторинг качества данных с LOM

Для детальной проверки качества данных, приходящих с монитора светимости, был разработан графический интерфейс пользователя (GUI), отображающий детальную информацию, такую как форма сигнала, относящегося к каждому сектору и некоторые низкоуровневые данные о пучковом фоне. GUI был разработан на языке программирования C++, с использованием фреймворка Qt 5 [114]. Скриншот GUI показан на рисунке 36

Рисунок 36 — Приложение для отображения данных с монитора светимости

В рамках данной работы, библиотека LOM Interface была интегрирована с этим приложением, также был добавлен системный демон, следящий за состоянием GUI, автоматически запускающий его при запуске системы и перезапускающий по запросу или по заданным условиям.

Чтобы интегрировать GUI в веб-интерфейс дежурного, описанный в разделе 5.7, это приложение запускается в отдельной сессии X11, доступ к которой предоставляется через веб-клиент noVNC [109].

7.3 Экспорт данных в систему медленного контроля

Функция экспорта в сеть медленного контроля EPICS встроена непосредственно в считывающее ПО монитора светимости. Для взаимодействия с EPICS

используется библиотека PythonIOC. Все основные переменные, экспортируемые в сеть медленного контроля, зарегистрированы на сервере EPICS Archiver [112], который сохраняет полную историю изменения значений, передаваемых в EPICS (к примеру, текущее значение светимости).

Поскольку на ранних стадиях эксперименте Belle II система медленного контроля NSM2 периодически работала нестабильно, данные в неё было решено передавать с помощью отдельного процесса, работающего независимо от считывающего ПО. Программа NSM2 node написана на языке C++ и использует библиотеку LOM Interface, чтобы получать кэшированные данные с монитора светимости и передавать их в сеть медленного контроля.

7.4 Консоль LOM

Оболочка командной строки для доступа к LOM (консоль LOM) представляет собой текстовый интерфейс пользователя (TUI) с набором команд для работы с монитором светимости.

Консоль LOM разработана на языке Python, для обработки команд и сохранения истории используется библиотека prompt_toolkit [115]. Данная программа использует LOM Interface для работы с LOM, а также поддерживает расширенный набор инструкций для управления считывающим ПО. Кроме того, консоль LOM поддерживает ряд высокоуровневых команд для установки амплитудных порогов в энергетических единицах и для выполнения различных отладочных процедур, в частности для энергетической калибровки.

7.5 Энергетическая калибровка LOM

Чтобы автоматизировать процедуру калибровки монитора светимости, описанную в разделе 1.5, использовался API для доступа к GUI управления заходами, описанный в разделе 5.5. Блок-схема энергетической калибровки монитора светимости приведена на рисунке 37. При запуске калибровочного захода, отдельная утилита параллельно конфигурирует ECL и LOM, затем запуская набор калибровочных данных в обеих системах DAQ.

Сохранённые файлы автоматически переносятся в файловое хранилище вычислительного центра, использующегося в эксперименте Belle II, где выполняется их дальнейшая обработка, необходимая для определения калибровочных коэффициентов.

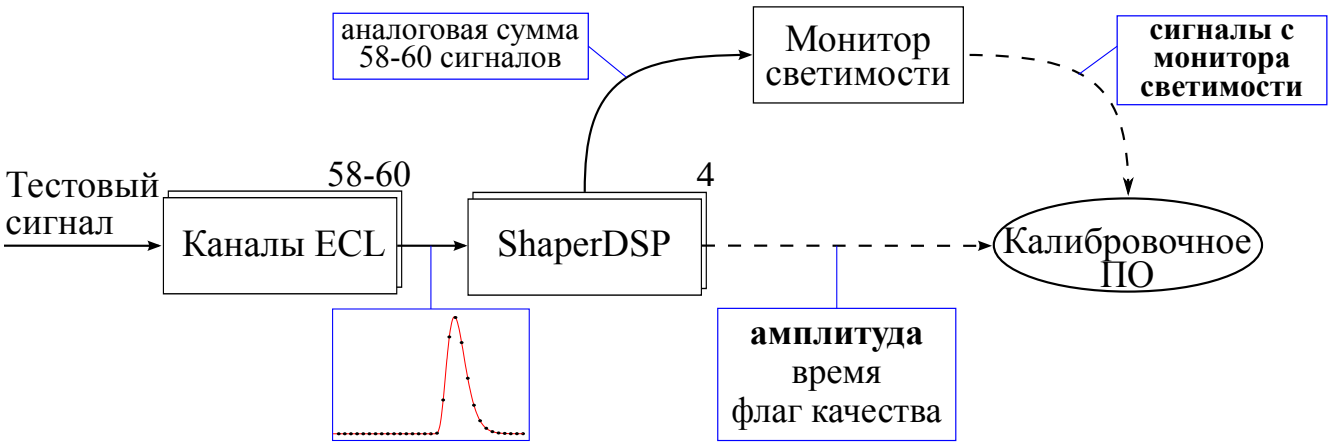

Рисунок 37 — Схема энергетической калибровки отдельного сектора монитора светимости

7.6 Средства автоматизированного тестирования

В ходе эксперимента, LOM должен непрерывно предоставлять данные. Новые версии ПО должны быть тщательно протестированы заранее, чтобы исключить ошибки со сбором данных в ходе эксперимента и сократить время, требуемое на обновление.

Чтобы тестировать считывающую программу LOM, был реализован режим эмулятора монитора светимости, в котором считывание происходит не из регистров LOM, а генерируется на основе ранее набранных данных. Это позволяет не только тестировать высокоуровневые функции (такие как подсчёт интегральной светимости), но и проверять работу других программ, использующих LOM (к примеру, монитор качества данных с LOM).

Также были разработаны утилиты для тестирования новых версий прошивки монитора светимости. Существует несколько автоматизированных тестов, использующих LOM Interface для проверки логики LOM. Существующие скрипты для обновления версии прошивки позволяют легко собрать новую версию прошивки, перепрограммировать модуль LOM и запускать набор автоматизированных тестов в одно действие.

7.7 Развёртка и сборка

Считывающее ПО LOM имеет свой набор внешних зависимостей, отдельный от общего программного обеспечения DAQ и медленного контроля. При каждом обновлении ПО, меняющем внешние зависимости, требуется выполнять

одинаковые шаги на главном, запасном и тестовом серверах.

Для обеспечения этих функций был создан отдельный репозиторий, в котором содержится список текущих зависимостей и набор скриптов, позволяющих быстро загрузить и скомпилировать необходимые версии внешних библиотек. ПО LOM настраивает переменные среды на стадии запуска, гарантируя использование одинаковых версий библиотек на разных целевых системах. Данные скрипты были подготовлены для работы на операционных системах (ОС), использующихся в сети DAQ: Scientific Linux 5-7 и ОС Ubuntu 18.06.

7.8 Основные результаты главы 7

Автором было разработано программное обеспечение для считывания данных с монитора светимости, работающее независимо от основной системы сбора данных детектора Belle II.

Разработанная система сбора данных управляет запросами к монитору светимости, задаёт их очерёдность и кэширует ответы на стандартные запросы.

Реализованы инструменты для мониторинга качества данных с монитора светимости, инструменты для энергетической калибровки и набор средств для развёртки и автоматизированного тестирования считывающего ПО.

ГЛАВА 8. ОБРАБОТКА ДАННЫХ

8.1 Временная калибровка

Программное обеспечение для считывания данных с калориметра, проверялось и отлаживалось на космических событиях. В рамках данной работы были обработаны заходы, полученные с использованием ECL DAQ и определены исходные значения коэффициентов для временной калибровки. Дальнейшее развитие процедур калибровки выходит за рамки этой работы и проводилось другой группой команды Belle II ECL.

Время сигнала, выдаваемое каждым счётчиком ECL, должно иметь одинаковое время относительно момента пересечения пучков. С этой целью для каждого канала вводится несколько поправочных коэффициентов, определяемых в ходе процедуры временной калибровки.

8.1.1 Временная калибровка по космическим событиям

Поскольку на начальной стадии эксперимента Belle II нельзя было провести калибровку по событиям столкновения e^+e^- пучков, она проводилась по событиям космических частиц. Космические частицы приходят случайным, не зависящим от времени образом, поэтому проводится временная калибровка счётчиков относительно друг друга. Это позволило получить начальные значения временных поправок, которые в дальнейшем были уточнены в калибровке по событиям e^+e^--рассеяния.

Для получения временных поправок был разработан модуль фреймворка BASF2, использующий метод минимизации χ^2:

$$f(t'_1, \ldots, t'_{8736}) = \sum_{n,i,j} \frac{\left(t_i^n - t_j^n - t'_i + t'_j - \Delta t_{i,j}^n\right)^2}{(\sigma_{i,j}^n)^2} \to \min, \tag{6}$$

Модуль BASF2 осуществляет отбор калибровочных событий и определяет калибровочные коэффициенты, что сводится к решению матричного уравнения вида $A\vec{t} = \vec{b}$.

Поскольку матрица A имеет размер 8736×8736 (число кристаллов в калориметре), стандартные функции работы с матрицами, предоставляемые библиотекой ROOT, требовали около часа для вычисления обратной матрицы на

процессоре Intel Xeon E5-2697. Чтобы оптимизировать эту часть калибровки, был реализован параллельный алгоритм LU-разложения, выполняющий обращение матрицы за 2 минуты.

В результате, после применения временных поправок, временное разрешение улучшилось на 18%, как показано на рисунке 38. Полученные коэффициенты были загружены в ConditionsDB и использовались в обработке данных с калориметра.

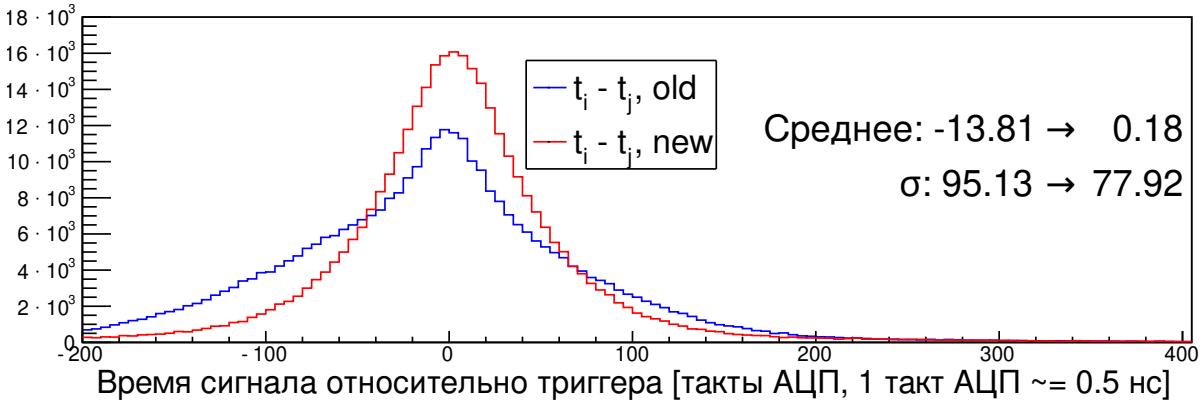

Рисунок 38 — Временное разрешение (σ) до (old) и после (new) применения временных поправок, полученных в процессе калибровки

8.1.2 Временная калибровка по событиям e^+e^--рассеяния

В рамках данной работы также было разработано два модуля BASF2, выполняющих временную калибровку по событиям e^+e^--рассеяния. Первый модуль обрабатывает полностью реконструированные события, проводя отбор по, в частности, числу треков и общему энерговыделению в калориметре. Далее, для каждого канала ECL модуль заполняет гистограмму разности между временем сигнала для этого канала и нулевым временем, вычисленным на основе информации с дрейфовой камеры.

Второй модуль выполняет несколько итераций подгонки полученных гистограмм функцией Гаусса, таким образом определяя значение временной поправки для каждого канала. Результаты работы модуля показаны на рисунке 39.

Полученные в результате калибровочные коэффициенты были загружены в ConditionsDB и использовались в обработке данных. Код модулей BASF2 был передан группе, проводящей калибровки Belle II ECL и продолжает использо-

ваться для вычисления новых версий временных поправок. Написанный код также использовался в работе по исследованию временного разрешения калориметра.

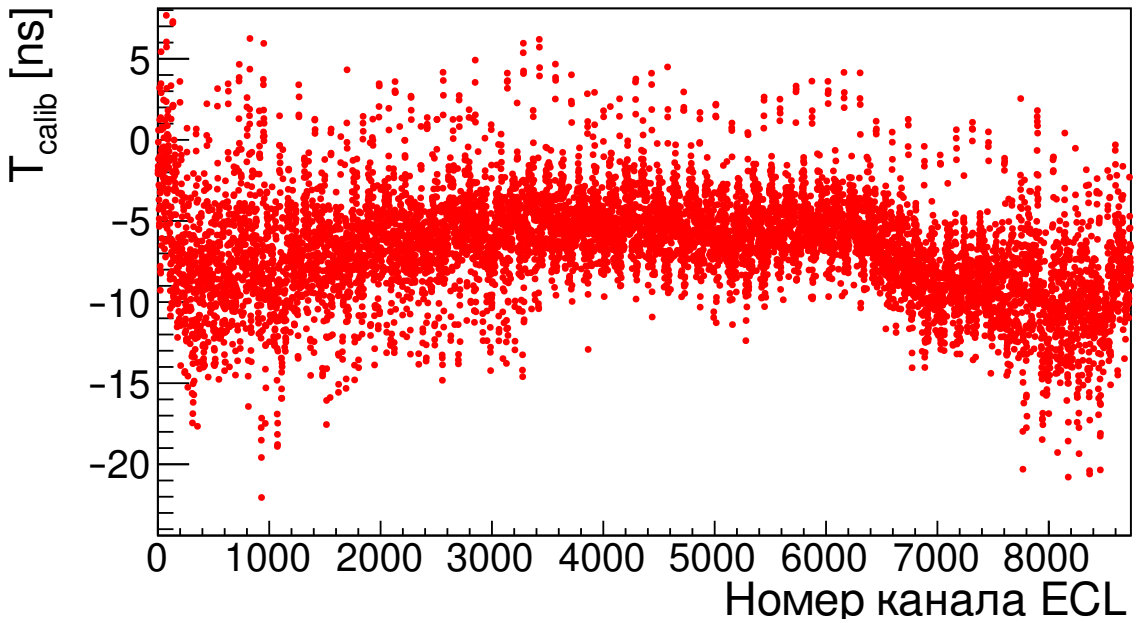

Рисунок 39 — Полученные результаты временной калибровки

8.2 Оптимизация распаковщика событий

Чтобы система HLT, описанная в разделе 1.6.2, не стала узким местом при обработке данных на проектной частоте триггера, необходимо своевременно определять и оптимизировать модули BASF2, замедляющие быстродействие системы. В частности, приоритетной задачей является оптимизация распаковки сырых данных с подсистем детектора, на которую уходит существенная доля процессорного времени.

Поэтому, в рамках данной работы была проанализирована производительность модуля ECLUnpacker, выполняющего распаковку данных с калориметра. Для этого было проведено профилирование кода с использованием утилиты callgrind [116].

Как было обнаружено по данным профилировщика на рисунке 40, значительная доля времени исполнения тратилась на логирование, что объяснялось некоторыми внутренними изъянами фреймворка анализа данных Belle II. Результаты исследования были представлены на совещании по программному

обеспечению Belle II, модуль распаковки событий с калориметра был переписан, что привело к повышению его скорости работы на ∼20%.

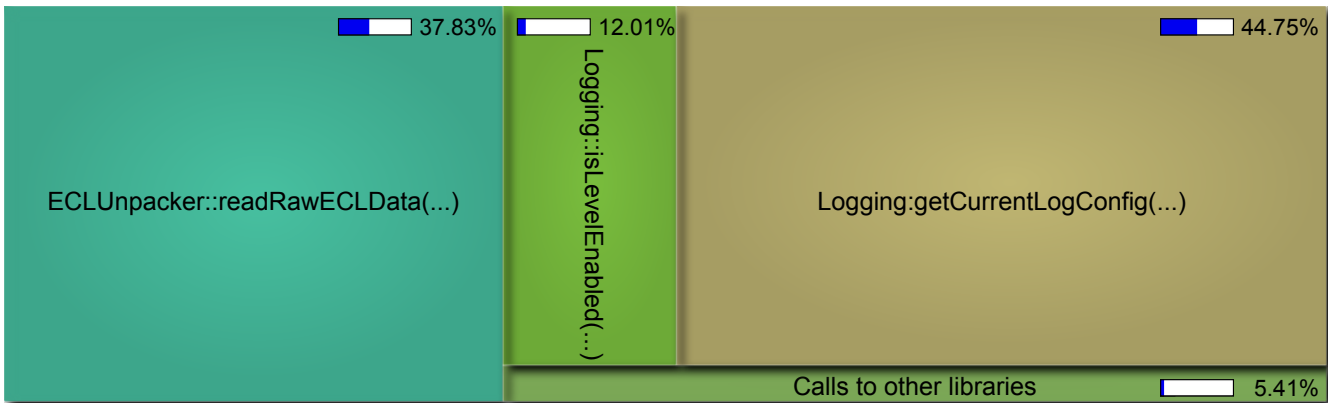

Рисунок 40 — Часть статистики вызовов из утилиты callgrind (GUI qcachegrind)

8.3 Позаходная конфигурация распаковщика событий

В ходе эксперимента возникают случаи, когда модуль ShaperDSP становится неисправен и должен быть заменён. При этом неизбежно набирается хотя бы один заход, содержащий данные с неисправной платы ShaperDSP. Данные с неё должны быть либо восстановлены, если это возможно, либо полностью исключены. Делать это разумнее всего в самом начале обработки, на стадии распаковки данных.

С этой целью был расширен модуль ECLUnpacker. При обработке данных, распаковщик сверяется со списком проблемных каналов, хранящемся в ConditionsDB для каждого захода. В зависимости от категории канала, возможны следующие варианты:

- Данные с проблемного канала/платы исключаются целиком.
- Данные отбираются по флагу качества подгонки.
- Берутся только данные АЦП и подгоняются заново.

Подобная реализация также позволяет уменьшить различия между данными эксперимента и позаходного моделирования — если какой-либо канал ECL исключён в эксперименте, он будет исключён и в моделировании.

8.4 Оптимизация калибровочных утилит

Belle Analysis Software Framework 2 (BASF2) поддерживает параллельную обработку событий отдельного захода, однако при большом объёме данных (больше десяти заходов) становится эффективным разделять обработку на несколько многопоточных процессов, каждый из которых обрабатывает отдельный фрагмент данных. Далее данные с отдельных файлов объединяются, проходят через дополнительную фильтрацию и переходят к конечной обработке. Схема этого процесса представлена на рисунке 41.

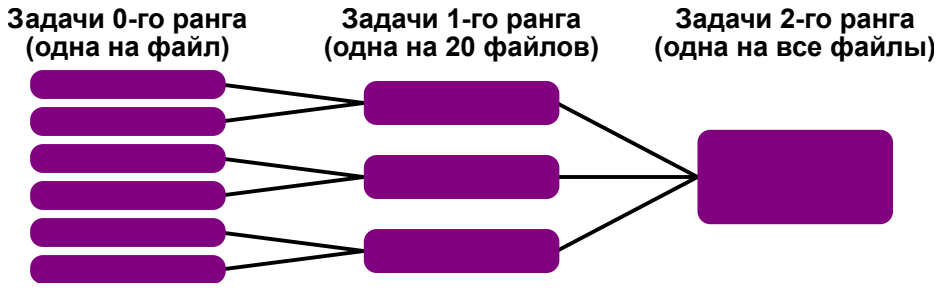

Рисунок 41 — Схема процесса калибровки, выполняющейся в несколько этапов

Разработанная процедура хорошо согласуется с библиотеками калибровки BASF2. Также, поскольку в BASF2 данные с отдельных потоков объединяются с помощью hadd [117] (утилиты объединения фалов библиотеки CERN ROOT), были внесены некоторые усовершенствования, увеличивающие быстродействие и понижающие потребление памяти в процессе объединения.

8.5 Калибровка нелинейности

В рамках данной работы была разработана первичная обработка данных калибровки нелинейности, описанной в разделе 1.5. Калибровочная процедура разделена на 9 отдельных заходов, каждый со своей конфигурацией. Информация о них заносится в БД калибровочных заходов, описанную в разделе 5.5. Каждый сохранённый файл обрабатывается в соответствии с конфигурацией, записанной в БД. Для повышения быстродействия используются утилиты из раздела 8.4.

Полученные результаты используются для быстрой диагностики возможных проблем в каналах калориметра. Выходные файлы передаются другим разработчикам в группе Belle II ECL для дальнейшего анализа.

8.6 Управляющие файлы Python

Для быстрой обработки данных ECL с использованием фреймворка BASF2, было разработано несколько управляющих файлов Python, позволяющих генерировать объекты ROOT TTree, содержащие детальную информацию с электроники калориметра. Для записи выходных файлов был разработан небольшой модуль TTreeWriter, который позволяет существенно упростить заполнение деревьев ROOT, как показано на рисунке 42.

```
tree  = ROOT.TTree('tree', 'TTree description')
evtn = array('i', [0])
trgtype = array('b', [0]*255)
energy  = array('d', [0]*100)
tree.Branch('evtn', evtn, 'evtn/I').\
     SetTitle('Event number')
tree.Branch('trgtype', trgtype, 'trgtype[255]/C').\
     SetTitle('Trigger type')
tree.Branch('energy', energy, 'energy[100]/D').\
     SetTitle('Hit energy for up to 100 hits')
evtn[0] = 1
new_trgtype = bytearray('random', encoding='ascii') + b'\0'
for i in range(len(new_trgtype)):
  trgtype[i] = new_trgtype[i]
energy[0:3] = [0.2, 0.3, 0.21]
```

```
tree  = ROOT.TTree('tree', 'TTree description')
branch_info = [
  ['evtn/I', 'Event number'],
  ['trgtype[255]/C', 'Trigger type'],
  ['energy[100]/D', 'Hit energy for up to 100 hits'],
]
writer = TTreeWriter(tree, branch_info)
writer.evtn = 1
writer.trgtype = 'random'
writer.energy = [0.2, 0.3, 0.21]
writer.fill()
```

Рисунок 42 — Стандартный код для заполнения дерева (слева) и код с использованием TTreeWriter (справа)

Данные файлы использовались для определения оптимальных параметров в конфигурации ECL, обнаружения шумящих каналов, анализа проблем с ПЛИС, исследования влияния пучкового фона и тестирования согласования моделирования калориметра с экспериментальными данными.

Вдобавок, для анализа условий пучкового фона было разработано несколько вспомогательных модулей Python, позволяющих объединить информацию из DAQ DB, ConditionsDB, EPICS Archiver и детальных данных, записанных с монитора светимости.

8.7 Основные результаты главы 8

Чтобы улучшить работы системы сбора данных, автором был подготовлен ряд инструментов для детальной обработки информации с калориметра. В частности, были разработаны первичные версии временной калибровки по космическим событиям и событиям Бабы. Также был разработан программный модуль, использующийся в первичной обработке данных для калибровки

нелинейности.

Был оптимизирован код распаковщика событий, что позволило ускорить критичную часть обработки данных с ECL на ~20%. Также в код распаковки событий была добавлена возможность дополнительной фильтрации данных по информации из ConditionsDB.

Подготовлена вспомогательная библиотека, позволяющая улучшить быстродействие обработки данных за счёт эффективного разделения задач по кластерам и модуль на языке Python, позволяющий существенно сократить код для работы с данными ROOT TTree.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано программное обеспечение для системы сбора данных электромагнитного калориметра. Рабочий процесс преимущественно ориентировался на принципы Rapid Application Development и Test-Driven Development. Использовалась микросервисная архитектура для лучшей интеграции с остальными компонентами системы сбора данных.

Подготовлены программные модули для синхронизации и управления конфигурацией электроники сбора данных в двух базах данных, использующихся в эксперименте Belle II. Конфигурация легко может корректироваться для разных типов заходов, история конфигурации архивируется и может быть в дальнейшем использована для моделирования калориметра с заданными настройками. Разработаны решения для задания более детальной тестовой конфигурации. Чтобы эффективнее использовать ресурсы БД, создан алгоритм упаковки, специально подстроенный под использующуюся структуру конфигурационных данных. Его использование позволило увеличить эффективность сжатия в три раза, не теряя при этом в быстродействии.

Разработан фреймворк, объединяющий все функции инициализации электроники калориметра. Использующаяся архитектура фреймворка позволяет абстрагироваться от используемого протокола, предоставляет возможности для автоматизированного тестирования программного обеспечения и может адаптироваться ко всем требуемым сценариям использования. Также были подготовлены высокоуровневые утилиты инициализации, которые позволяют довольно просто выполнять стандартные операции и поддерживают большое число дополнительных параметров работы. Эти утилиты регулярно используются дежурными экспертами для диагностики и исправления проблем с калориметром, а также при тестировании новых версий прошивки. Процедура инициализации может работать независимо от основной системы медленного контроля, развёртывая свою сеть взаимодействующих процессов.

Большинство разработанных приложений интегрированы с фреймворком Network Shared Memory 2. Для ускорения процесса разработки была реализована библиотека pyNSM2, позволяющая взаимодействовать с системой медленного контроля с использованием языка Python. Подготовлено несколько системных демонов, управляющих электроникой калориметра, расширен графический интерфейс для запуска калибровок, создана отдельная база данных с инфор-

мацией по набранным калибровочным заходам.

Все основные процедуры диагностики и устранения проблем с электроникой калориметра были внесены в документацию дежурных. Для экспертов разработан веб-сервер, объединяющий все необходимые утилиты для быстрого доступа. Разработано несколько приложений для мониторинга качества данных с калориметра, реализована система автоматических оповещений.

Было подготовлено программное обеспечение для считывания данных с монитора светимости и передачи информации о светимости в систему медленного контроля. Детальная информация о мониторе светимости также доступна на веб-сервере дежурного.

В ходе работы были разработаны и дополнены некоторые модули для обработки данных с калориметра. В частности, были разработаны исходные версии модулей временной калибровки, оптимизирована распаковка данных и реализованы процедуры для первичной обработки данных в калибровке нелинейности. Для детального исследования качества данных реализован набор управляющих скриптов Python, с отдельной библиотекой для структурированной записи выходных данных.

В заключение я бы хотел выразить благодарность и глубокую признательность моему научному руководителю Александру Степановичу Кузьмину за постоянное внимание и активное участие в данной работе.

Я очень благодарен своим коллегам из ИЯФ СО РАН, чьи идеи, советы и рекомендации очень помогли мне в моей работе: Шебалину Василию Евгеньевичу, Шварцу Борису Альбертовичу, Матвиенко Дмитрию Владимировичу, Боброву Александру Викторовичу, Грибанову Сергею Сергеевичу, Винокуровой Анне Николаевне, Козыреву Алексею Николаевичу, Гармашу Алексею Юрьевичу, Епифанову Денису Александровичу, Кроковному Павлу Петровичу, Кане Кириллу Олеговичу, Жуланову Владимиру Викторовичу, Коробову Александру Алексеевичу, Антоновой Яне Николаевне.

Также я бы хотел поблагодарить своих коллег по эксперименту Belle II: Сатору Ямаду, Микихико Накао, Оскара Хартбри, Дмитрия Ливенцева, Мартина Риттера, Бъёрна Спрака и Кристофера Харти. Наконец, я хочу выразить признательность всем участникам коллаборации Belle II и коллективу SuperKEKB, чьи усилия открыли возможность для осуществления этой работы.

СПИСОК ЛИТЕРАТУРЫ

[1] **Kurokawa, S.-I.** Overview of the KEKB accelerators / S.-I. Kurokawa, E. Kikutani. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A. — 2003. — Vol. 499. — P. 1–7. — DOI 10.1016/S0168-9002(02)01771-0. — Дата публикации: 21.02.2003.

[2] **Akai, K.** SuperKEKB collider / K. Akai, K. Furukawa, H. Koiso. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2018. — Vol. 907. — P. 188–199. — DOI 10.1016/j.nima.2018.08.017. — Дата публикации: 17.08.2018.

[3] First measurements of beam backgrounds at SuperKEKB / P.M. Lewis, I. Jaegle, H. Nakayama, A. Aloisio [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2019. — Vol. 914. — P. 69–144. — DOI 10.1016/j.nima.2018.05.071. — Дата публикации: 11.01.2019.

[4] Belle II Technical Design Report / T. Abe, I. Adachi, K. Adamczyk, S. Ahn [et al.]. — Текст : электронный. — 2010. — DOI 10.48550/arXiv.1011.0352. — Дата публикации: 01.11.2010.

[5] The Belle detector / A. Abashian, K. Gotow, N. Morgan, L. Piilonen [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research A. — 2002. — Vol. 479. — P. 117–232. — DOI 10.1016/S0168-9002(01)02013-7. — Дата публикации: 12.03.2002.

[6] CsI(Tl) pulse shape discrimination with the Belle II electromagnetic calorimeter as a novel method to improve particle identification at electron–positron colliders / S. Longo, J.M. Roney, C. Cecchi, S. Cunliffe [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2020. — Vol. 982, nr. 0168-9002. — P. 164562. — DOI 10.1016/j.nima.2020.164562. — Дата публикации: 01.12.2020.

[7] Data acquisition system for Belle II electromagnetic calorimeter / A. Kuzmin, M. Remnev, D. Matvienko, Y. Usov [et al.]. — Текст : электронный // Journal of Instrumentation. — 2020. — Vol. 15, nr. 7. — P. C07020. — DOI 10.1088/1748-0221/15/07/C07020. — Дата публикации: 13.07.2020.

[8] Data acquisition system for the calorimeter of the Belle II detector / A. Kuzmin, M. Remnev, D. Matvienko, Y. Usov [et al.]. — Текст : электронный // Physics of Atomic Nuclei. — 2021. — Vol. 84, nr. 1. — P. 42–44. — DOI 10.1134/S1063778821010257. — Дата публикации: 13.04.2021.

[9] Trigger slow control system of the Belle II experiment / C.-H. Kim, Y. Unno, H.E. Cho, B.G. Cheon [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A. — 2021. — Vol. 1014, nr. 0168-9002. — P. 165748. — DOI 10.1016/j.nima.2021.165748. — Дата публикации: 17.08.2021.

[10] Development of data acquisition system for Belle II electromagnetic calorimeter / V. Aulchenko, A. Bobrov, B.G. Cheon, A. Kuzmin [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A. — 2022. — Vol. 1030. — P. 166468. — DOI 10.1016/j.nima.2022.166468. — Дата публикации: 15.02.2022.

[11] **Abe, K.** B physics with an asymmetric collider at KEK / K. Abe. — Текст : электронный // Proceedings of the third meeting on physics at TeV energy scale. — 1990. — P. 318–331. — URL: `http://inis.iaea.org/search/search.aspx?orig_q=RN:22066683` (дата обращения: 16.12.2022).

[12] **Lafferty, G.** SuperB: A High-Luminosity Asymmetric e^+e^- Super Flavor Factory. Conceptual Design Report / G. Lafferty, M. Bona [et al.]. — Текст : электронный. — 2007. — DOI 10.48550/arXiv.0709.0451. — Дата публикации: 06.09.2007.

[13] Accelerator design at SuperKEKB / Y. Ohnishi, T. Abe, T. Adachi, K. Akai [et al.]. — Текст : электронный // Progress of Theoretical and Experimental Physics. — 2013. — Vol. 2013, nr. 3. — 03A011. DOI 10.1093/ptep/pts083. — Дата публикации: 26.03.2013.

[14] Development and evaluation of charge-sensitive preamplifier for CsI calorimeter in the KEK B-factory / M. Tanaka, H. Ikeda, K. Tamai, M. Takemoto [et al.]. — Текст : электронный // IEEE Transactions on Nuclear Science. — 1994. — Vol. 41, nr. 4. — P. 1208–1211. — DOI 10.1109/23.322885. — Дата публикации: 01.08.1994.

[15] Electromagnetic calorimeter for Belle II / V. Aulchenko, A. Bobrov, A. Bondar, B. G. Cheon [et al.]. — Текст : электронный // Journal of Physics: Conference Series. — 2015. — Vol. 587. — P. 012045. — DOI 10.1088/1742-6596/587/1/012045. — Дата публикации: 01.02.2015.

[16] Measurement of the integrated luminosity of the Phase 2 data of the Belle II experiment / F. Abudinén, I. Adachi, P. Ahlburg, H. Aihara [et al.]. — Текст : электронный // Chinese Physics C. — 2020. — Vol. 44, nr. 2. — P. 021001. — DOI 10.1088/1674-1137/44/2/021001. — Дата публикации: 21.01.2020.

[17] Belle2Link: a Global data Readout and Transmission for Belle II Experiment at KEK / D. Sun, Z. Liua, J. Zhao, H. Xu. — Текст : электронный // Physics Procedia. — 2012. — Vol. 37. — P. 1933–1939. — DOI 10.1016/j.phpro.2012.01.036. — Дата публикации: 02.10.2012.

[18] Upgrade of trigger and DAQ for CsI at Belle II / V. Aulchenko, B. G. Cheon, A. Kuzmin, D. Matvienko [et al.]. — Текст : электронный // Journal of Instrumentation. — 2014. — Vol. 9, nr. 09. — P. C09014–C09014. — DOI 10.1088/1748-0221/9/09/c09014. — Дата публикации: 12.09.2014.

[19] **Analog Devices, Inc**. AD9637 : [сайт]. — URL: `http://www.analog.com/media/en/technical-documentation/data-sheets/AD9637.pdf` (дата обращения: 16.12.2022). — Текст : электронный.

[20] **Xilinx**. Spartan-6 FPGA Data Sheet: DC and Switching Characteristics : [сайт]. — URL: `https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf` (дата обращения: 16.12.2022). — Текст : электронный.

[21] Transputer based data acquisition system for the CMD-2 detector / G.A Aksenov, V.S Banzarov, T.B Bolshakov, A.G Chertovskikh [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A. — 1996. — Vol. 379, nr. 3. — P. 550–551. — Proceedings

of the Sixth International Conference on Instrumentation for Experiments at e+ e- Colliders. DOI 10.1016/0168-9002(96)00560-8. — Дата публикации: 21.09.1996.

[22] Детектор СНД: Модернизация систем для экспериментов на ВЭПП-2000 и некоторые предварительные результаты экспериментов на ВЭПП-2М / Г. Н. Абрамов, П. М. Астигеевич, В. М. Аульченко, А. Г. Богданчиков [et al.]. — Текст : электронный // Препринт ИЯФ. — 2007. — Vol. 20. — URL: http://wwwold.inp.nsk.su/activity/preprints/files/2007_020.pdf (дата обращения: 16.12.2022).

[23] SND DAQ system evolution / G.A. Bogdanchikov, V.P. Druzhinin, A.A. Korol, S.V. Koshuba [et al.]. — Текст : электронный // Journal of Physics: Conference Series / IOP Publishing. — 2017. — Vol. 898. — P. 032027. — DOI 10.1088/1742-6596/898/3/032027. — Дата публикации: 02.10.2017.

[24] The CMD-3 Data Acquisition System Upgrade / A. Ruban, A. Kozyrev, I. Logashenko, A. Selivanov. — 2014. — Instrumentation for colliding beam physics. URL: https://indico.inp.nsk.su/event/0/contributions/81/ (дата обращения: 16.12.2022). — Текст : электронный.

[25] Архитектура системы регистрации и запуска детектора КМД-3 / В. М. Аульченко, Д. А. Епифанов, А. Н. Козырев, И. Б. Логашенко [et al.]. — Текст : электронный // Автометрия. — 2015. — Vol. 51, nr. 1. — P. 31–38. — Дата публикации: 01.01.2015.

[26] Детектор КЕДР / В. В. Анашин, В. М. Аульченко, Е. М. Балдин, А. К. Барладян [et al.]. — Текст : электронный // Физика элементарных частиц и атомного ядра. — 2013. — Vol. 44, nr. 4. — P. 1264–1345. — Дата публикации: 01.04.2013.

[27] Data acquisition and monitoring for the KLOE detector / A. Aloisio, F. Ambrosino, M. Antonelli, C. Bini [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2004. — Vol. 516, nr. 2-3. — P. 288–314. — DOI 10.1016/j.nima.2003.06.013. — Дата публикации: 03.10.2003.

[28] **Harris, F. A.** BEPCII and BESIII / F. A. Harris. — 2006. — 2nd International Workshop on e^+e^- Collisions from ϕ to ψ. URL: `https://www.phys.hawaii.edu/~fah/bestalks/besiii_novosibirsk.pdf` (дата обращения: 16.12.2022). — Текст : электронный.

[29] Trigger System of BESIII / W. Gong, Y. Guo, D. Jin, L. Li [et al.]. — Текст : электронный // 2007 15th IEEE-NPSS Real-Time Conference / IEEE. — 2007. — P. 1–4. — URL: `https://doi.org/10.1109/RTC.2007.4382859`. — Дата публикации: 12.11.2007.

[30] Design and construction of the BESIII detector / M. Ablikim, Z. An, J. Bai, N. Berger [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2010. — Vol. 614, nr. 3. — P. 345–399. — DOI 10.1016/j.nima.2009.12.050. — Дата публикации: 01.03.2010.

[31] Data acquisition system for the Belle experiment / M. Nakao, M. Yamauchi, S. Y. Suzuki, R. Itoh [et al.]. — Текст : электронный // IEEE Transactions on Nuclear Science. — 2000. — Vol. 47, nr. 2. — P. 56–60. — DOI 10.1109/23.846117. — Дата публикации: 01.04.2000.

[32] The BABAR detector / B. Aubert, A. Bazan, A. Boucham, D. Boutigny [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2002. — Vol. 479, nr. 1. — P. 1–116. — DOI 10.1016/S0168-9002(01)02012-5. — Дата публикации: 21.02.2002.

[33] The BABAR detector: Upgrades, operation and performance / B. Aubert, R. Barate, D. Boutigny, F. Couderc [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2013. — Vol. 729. — P. 615–701. — DOI 10.1016/j.nima.2013.05.107. — Дата публикации: 22.06.2013.

[34] **Wolf, A.** The CLEO III data acquisition system / A. Wolf. — Текст : электронный // 1995 IEEE Nuclear Science Symposium and Medical Imaging Conference Record / IEEE. — 1995. — Vol. 2. — P. 686–689. — DOI 10.1109/NSSMIC.1995.510363. — Дата публикации: 06.08.2002.

[35] **Jost, B.** The LHCb DAQ system / B. Jost. — Текст : электронный // 2000 IEEE Nuclear Science Symposium. Conference Record (Cat. No. 00CH37149) / IEEE. — 2000. — Vol. 3. — P. 26–1. — DOI 10.1109/NSSMIC.2000.949418. — Дата публикации: 06.08.2002.

[36] LHCb detector performance / R. Aaij, B. Adeva, M. Adinolfi, A. Affolder [et al.]. — Текст : электронный // International Journal of Modern Physics A. — 2015. — Vol. 30, nr. 07. — P. 1530022. — DOI 10.1142/S0217751X15300227. — Дата публикации: 05.03.2015.

[37] Data Acquisition System for the Belle II Experiment / S. Yamada, R. Itoh, K. Nakamura, M. Nakao [et al.]. — Текст : электронный // IEEE Transactions on Nuclear Science. — 2015. — Vol. 62, nr. 3. — P. 1175–1180. — DOI 10.1109/TNS.2015.2424717. — Дата публикации: 03.06.2015.

[38] **Fabjan, C.** The story of ALICE: Building the dedicated heavy ion detector at LHC / C. Fabjan, J. Schukraft. — Текст : электронный. — 2011. — DOI 10.48550/arXiv.1101.1257. — Дата публикации: 06.01.2011.

[39] The ALICE data acquisition system / F. Carena, W. Carena, S. Chapeland, V. Chibante Barroso [et al.]. — Текст : электронный // Nuclear Instruments and Methods in Physics Research Section A. — 2014. — Vol. 741. — P. 130–162. — DOI 10.1016/j.nima.2013.12.015. — Дата публикации: 21.03.2014.

[40] The architecture of the CMS Level-1 Trigger Control and Monitoring System using UML / M. Magrans de Abril, J. Melo, C. Larrea, J. Hammer [et al.]. — Текст : электронный // Journal of Physics: Conference Series. — 2011. — Vol. 331. — P. 022020. — DOI 10.1088/1742-6596/331/2/022020. — Дата публикации: 01.12.2011.

[41] **Barney, D.** Introduction to CMS for CERN guides. — 2013. — URL: `https://indico.cern.ch/event/23313/attachments/388446/540226/CMSfor_CERNguides.pdf` (дата обращения: 16.12.2022). — Текст : электронный.

[42] **Hegeman, J. G.** The CMS Data Acquisition System for the Phase-2 Upgrade / J. G. Hegeman. — Текст : электронный // 21st IEEE Real Time Conference. — Geneva. — 2018. — URL: `http://cds.cern.ch/record/2629375` (дата обращения: 16.12.2022).

[43] The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger / G. Badaro, U. Behrens, A. Bocci, J. Branson [et al.]. — Текст : электронный // EPJ Web Conf. — 2021. — Vol. 251. — P. 04023. — DOI 10.1051/epjconf/202125104023. — Дата публикации: 23.08.2021.

[44] **Buttinger, W.** The ATLAS level-1 trigger system / W. Buttinger. — Текст : электронный // Journal of Physics: Conference Series / IOP Publishing. — 2012. — Vol. 396. — P. 012010. — DOI 10.1088/1742-6596/396/1/012010. — Дата публикации: 19.06.2022.

[45] **Vazquez, W. P.** The ATLAS data acquisition system in LHC run 2 / W. P. Vazquez [et al.]. — Текст : электронный. — 2017. — Vol. 898, nr. 3. — P. 032017. — DOI 10.1088/1742-6596/898/3/032017. — Дата публикации: 10.11.2017.

[46] ATLAS data quality operations and performance for 2015–2018 data-taking / G. Aad, B. Abbott, D.C. Abbott, A. Abed Abud [et al.]. — Текст : электронный // Journal of Instrumentation. — 2020. — Vol. 15, nr. 04. — P. P04003. — DOI 10.1088/1748-0221/15/04/P04003. — Дата публикации: 02.04.2020.

[47] Data acquisition system for Belle II / M. Nakao, T. Higuchi, R. Itoh, S. Suzuki. — Текст : электронный // Journal of Instrumentation. — 2010. — Vol. 5, nr. 12. — P. C12004. — DOI 10.1088/1748-0221/5/12/C12004. — Дата публикации: 02.12.2010.

[48] Development of a PCI Based Data Acquisition Platform for High Intensity Accelerator Experiments / T. Higuchi, H. Fujii, M. Ikeno, Y. Igarashi [et al.]. — Текст : электронный. — 2003. — DOI 10.48550/arXiv.hep-ex/0305088. — Дата публикации: 28.05.2003.

[49] **Xilinx**. UG196 Virtex-5 FPGA RocketIO GTP Transceiver User Guide v2.1 : [сайт]. — URL: `http://xilinx.com/support/documentation/user_guides/ug196.pdf` (дата обращения: 16.12.2022). — Текст : электронный.

[50] The Belle II Pixel Detector Data Acquisition and Reduction System / B. Spruck, T. Geßler, W. Kühn, J. S. Lange [et al.]. — Текст : электронный // IEEE Transactions on Nuclear Science. — 2013. — Vol. 60, nr. 5. — P. 3709–3713. — DOI 10.1109/TNS.2013.2281571. — Дата публикации: 30.09.2013.

[51] Data flow and high level trigger of Belle II DAQ system / R. Itoh, T. Higuchi, M. Nakao, S. Y. Suzuki [et al.]. — Текст : электронный // IEEE Trans. Nucl. Sci. — 2013. — Vol. 60. — P. 3720–3724. — DOI 10.1109/RTC.2012.6418174. — Дата публикации: 24.01.2013.

[52] **Moll, A.** The Software Framework of the Belle II Experiment / A. Moll. — Текст : электронный // Journal of Physics: Conference Series. — 2011. — Vol. 331, nr. 3. — P. 032024. — DOI 10.1088/1742-6596/331/3/032024. — Дата публикации: 01.12.2011.

[53] ROOT, a data analysis framework : [сайт]. — URL: `https://root.cern.ch` (дата обращения: 16.12.2022). — Текст : электронный.

[54] Boost C++ libraries : [сайт]. — URL: `http://www.boost.org/` (дата обращения: 16.12.2022). — Текст : электронный.

[55] CLHEP : [сайт]. — URL: `http://cern.ch/clhep/` (дата обращения: 16.12.2022). — Текст : электронный.

[56] Libxml2 : [сайт]. — URL: `http://xmlsoft.org/` (дата обращения: 16.12.2022). — Текст : электронный.

[57] Belle II Conditions Database / M. Ritter, L. Wood, T. Kuhr, M. Bracko [et al.]. — Текст : электронный // Journal of Physics: Conference Series. — 2018. — Vol. 1085. — P. 032032. — DOI 10.1088/1742-6596/1085/3/032032. — Дата публикации: 01.09.2018.

[58] **Konno, T.** DAQ Database / T. Konno. — Текст : электронный // Trigger/DAQ workshop for Belle II experiment. — 2017. — URL: `http://hep5.phys.ntu.edu.tw/event/157/session/14/contribution/22/material/slides/` (дата обращения: 16.12.2022).

[59] PostgreSQL : [сайт]. — URL: `https://www.postgresql.org/` (дата обращения: 16.12.2022). — Текст : электронный.

[60] JSON : [сайт]. — URL: `https://www.json.org/` (дата обращения: 16.12.2022). — Текст : электронный.

[61] **Curé, O.** Chapter Two - Database Management Systems / O. Curé, G. Blin // RDF Database Systems. — Morgan Kaufmann, 2015. — P. 9–40. — ISBN: 0127999574. — Текст : электронный.

[62] **Brun, R.** Design, Development and Evolution of the ROOT System / R. Brun, P. Canal, F. Rademakers. — Текст : электронный // Proceedings of Science. — 2010. — Vol. ACAT2010. — P. 002. — DOI 10.22323/1.093.0002. — Дата публикации: 24.02.2011.

[63] **Janyst, L.** ROOT Data Model Evolution : [сайт]. — URL: `https://root.cern.ch/root/SchemaEvolution.pdf` (дата обращения: 16.12.2022). — Текст : электронный.

[64] **Nakao, M.** Network shared memory framework for the Belle data acquisition control system / M. Nakao, S. Y. Suzuki. — Текст : электронный // 1999 IEEE Conference on Real-Time Computer Applications in Nuclear Particle and Plasma Physics. 11th IEEE NPSS Real Time Conference. Conference Record (Cat. No.99EX295). — 1999. — P. 346–350. — DOI 10.1109/RTCON.1999.842639. — Дата публикации: 06.08.2002.

[65] **Dalesio, L.R.** EPICS Architecture / L.R. Dalesio, A.J. Kozubal, M.R. Kraimer. — Текст : электронный // ICALEPCS 91, KEK Proceedings 92-15. — 1992. — Vol. 278. — URL: `https://ci.nii.ac.jp/naid/10003218710/en/` (дата обращения: 16.12.2022).

[66] **Kasemir, K.** Control system studio guide / K. Kasemir, G. Carcassi. — Oak Ridge National Laboratory, 2011. — URL: `https://controlssoftware.sns.ornl.gov/training/2013_SNS/css_book.pdf` (дата обращения: 16.12.2022). — Текст : электронный.

[67] Jython Project : [сайт]. — URL: `https://www.jython.org/` (дата обращения: 16.12.2022). — Текст : электронный.

[68] The Slow Control and Data Quality Monitoring System for the Belle II Experiment / T. Konno, R. Itoh, M. Nakao, S. Suzuki [et al.]. — Текст : электронный // IEEE Transactions on Nuclear Science. — 2015. — Vol. 62. — P. 897–902. — DOI 10.1109/RTC.2014.7097454. — Дата публикации: 30.04.2015.

[69] **Braun, N.** Software for Online Reconstruction and Filtering at the Belle II Experiment / N. Braun, T. Kuhr. — Текст : электронный. — 2020. — DOI 10.48550/arXiv.2003.02552. — Дата публикации: 05.03.2020.

[70] JSROOT : [сайт]. — URL: `https://github.com/root-project/jsroot` (дата обращения: 16.12.2022). — Текст : электронный.

[71] Easy!Appointments - Open Source Appointment Scheduler : [сайт]. — URL: `https://easyappointments.org/` (дата обращения: 16.12.2022). — Текст : электронный.

[72] Rocket.Chat : [сайт]. — URL: `https://rocket.chat/` (дата обращения: 16.12.2022). — Текст : электронный.

[73] **Kohler, S.** Atlassian confluence 5 essentials / S. Kohler. — Packt Publishing Ltd, 2013. — ISBN: 1849689520. — Текст (визуальный) : непосредственный.

[74] **Moses, J.** Re-usability And Extendability In Object-oriented And Object-based Design / J. Moses. — WIT Press, 1994. — Vol. 9. — P. 427–440. — ISBN: 1853123536. — Текст : электронный.

[75] Microservices: yesterday, today, and tomorrow / N. Dragoni, S. Giallorenzo, A. Lluch Lafuente, M. Mazzara [et al.]. — Текст : электронный // Present and ulterior software engineering. — 2017. — P. 195–216. — DOI 10.1007/978-3-319-67425-4_12. — Дата публикации: 06.09.2017.

[76] Virtual network computing / T. Richardson, Q. Stafford-Fraser, K.R. Wood, A. Hopper. — Текст : электронный // IEEE Internet Computing. — 1998. — Vol. 2, nr. 1. — P. 33–38. — DOI 10.1109/4236.656066. — Дата публикации: 01.02.1998.

[77] **Fielding, R. T.** Architectural styles and the design of network-based software architectures / R. T. Fielding. — Текст : электронный. — 2000. — URL: `https://ics.uci.edu/~fielding/pubs/dissertation/top.htm` (дата обращения: 16.12.2022).

[78] Doxygen : [сайт]. — URL: `https://www.doxygen.nl/index.html` (дата обращения: 16.12.2022). — Текст : электронный.

[79] **Ince, D. C.** Software prototyping — progress and prospects / D. C. Ince, S. Hekmatpour. — Текст : электронный // Information and Software Technology. — 1987. — Vol. 29, nr. 1. — P. 8–14. — DOI 10.1016/0950-5849(87)90014-0. — Дата публикации: 10.06.2003.

[80] **Prechelt, L.** An empirical comparison of C, C++, Java, Perl, Python, Rexx and tcl / L. Prechelt. — Текст : электронный // IEEE Computer. — 2000. — Vol. 33, nr. 10. — P. 23–29. — Дата публикации: 01.04.2000.

[81] Prototyping with Python / A. Zeller, R. Gopinath, M. Böhme, G. Fraser [et al.] // The Fuzzing Book. — CISPA, 2020. — URL: `https://www.fuzzingbook.org/beta/html/PrototypingWithPython.html` (дата обращения: 16.12.2022). — Текст : электронный.

[82] **Agarwal, K.** Rapid Applications Development in Python / K. Agarwal. — Текст : электронный // Twenty Fourth Annual CCSC South Central Conference. — 2013. — URL: `https://www.researchgate.net/publication/297733448_Rapid_Applications_Development_in_Python` (дата обращения: 16.12.2022).

[83] **Beazley, D.** Understanding the Python GIL / D. Beazley. — Текст : электронный // PyCON Python Conference. — 2010. — URL: `https://dabeaz.com/python/UnderstandingGIL.pdf` (дата обращения: 16.12.2022).

[84] Python Global Interpreter Lock (GIL) + numerical data processing : [сайт]. — URL: `http://www.bnikolic.co.uk/blog/python-numerical-gil.html` (дата обращения: 16.12.2022). — Текст : электронный.

[85] Python 3.2 release notes : [сайт]. — URL: `https://docs.python.org/whatsnew/3.2.html` (дата обращения: 16.12.2022). — Текст : электронный.

[86] Convoy effect with I/O bound threads and New GIL : [сайт]. — URL: `https://bugs.python.org/issue7946` (дата обращения: 16.12.2022). — Текст : электронный.

[87] **Paxson, V.** Lexical analysis with flex / V. Paxson, W. Estes, J. Millaway. — Текст : электронный // University of California. — 2012. — URL: `https://westes.github.io/flex/manual/index.html` (дата обращения: 16.12.2022).

[88] **Donnelly, C.** Bison. The YACC-compatible parser generator / C. Donnelly, R. Stallman. — Iuniverse Inc, 2000. — ISBN: 9888381377. — Текст : электронный.

[89] An empirical evaluation of Lex/Yacc and ANTLR parser generation tools / F. Ortin, J. Quiroga, O. Rodriguez-Prieto, M. Garcia. — Текст : электронный // PLOS ONE. — 2022. — Vol. 17, nr. 3. — P. e0264326. — DOI 10.1371/journal.pone.0264326. — Дата публикации: 03.03.2022.

[90] **Parr, T.** Adaptive LL (*) parsing: the power of dynamic analysis / T. Parr, S. Harwell, K. Fisher. — Текст : электронный // ACM SIGPLAN Notices. — 2014. — Vol. 49, nr. 10. — P. 579–598. — DOI 10.1145/2714064.2660202. — Дата публикации: 01.10.2014.

[91] LZMA SDK : [сайт]. — URL: `https://www.7-zip.org/sdk.html` (дата обращения: 16.12.2022). — Текст : электронный.

[92] IEEE Standard Test Access Port and Boundary-Scan Architecture. — Текст : электронный // IEEE Std. 1149.1-1990. — 1990. — P. 1–139. — DOI 10.1109/IEEESTD.1990.114395. — Дата публикации: 21.05.1990.

[93] Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидс. — Питер, 2006. — ISBN: 5469011364. — Текст : электронный.

[94] **Stallman, R. M.** GNU Make Reference Manual / R. M. Stallman, R. McGrath, P. D. Smith. — 12th Media Services, 2016. — ISBN: 168092155X. — Текст (визуальный) : непосредственный.

[95] **Miller, P.** Recursive make considered harmful / P. Miller. — Текст : электронный // AUUGN Journal of AUUG Inc. — 1998. — Vol. 19, nr. 1. — P. 14–25. — URL: `https://grosskurth.ca/bib/1997/miller.pdf` (дата обращения: 16.12.2022).

[96] Non-Recursive Make Considered Harmful: Build Systems at Scale / A. Mokhov, N. Mitchell, S. Peyton Jones, S. Marlow. — Текст : электронный // SIGPLAN Not. — 2016. — Vol. 51, nr. 12. — P. 170–181. — DOI 10.1145/3241625.2976011. — Дата публикации: 08.09.2016.

[97] **White, J.** Contemporary Build Systems and Techniques for Improvement / J. White, K. Zemoudeh. — Текст : электронный // Proceedings of the International Conference on Software Engineering Research and Practice (SERP). — 2014. — P. 1. — URL: https://worldcomp-proceedings.com/proc/p2014/SER2661.pdf (дата обращения: 16.12.2022).

[98] **Matsakis, N. D.** The Rust language / N. D. Matsakis, F. S. Klock. — Текст : электронный // ACM SIGAda Ada Letters. — 2014. — Vol. 34, nr. 3. — P. 103–104. — DOI 10.1145/2692956.2663188. — Дата публикации: 26.11.2014.

[99] **Klabnik, S.** The Rust Programming Language (Covers Rust 2018) / S. Klabnik, C. Nichols. — No Starch Press, 2019. — ISBN: 1593278284. — Текст (визуальный) : непосредственный.

[100] **Beck, K.** Test-driven development: by example / K. Beck. — Addison-Wesley Professional, 2003. — ISBN: 9780321146533. — Текст (визуальный) : непосредственный.

[101] Gcov: a test coverage program : [сайт]. — URL: https://gcc.gnu.org/onlinedocs/gcc/Gcov.html (дата обращения: 16.12.2022). — Текст : электронный.

[102] urwid : [сайт]. — URL: http://urwid.org/ (дата обращения: 16.12.2022). — Текст : электронный.

[103] Shell In A Box : [сайт]. — URL: https://github.com/shellinabox/shellinabox (дата обращения: 16.12.2022). — Текст : электронный.

[104] Selenium : [сайт]. — URL: https://www.selenium.dev/ (дата обращения: 16.12.2022). — Текст : электронный.

[105] ctypes : [сайт]. — URL: https://pypi.org/project/ctypes/ (дата обращения: 16.12.2022). — Текст : электронный.

[106] **Van Rossum, G.** Python/C API reference manual / G. Van Rossum, F. L. Drake Jr. — Текст : электронный // Python Software Foundation. — 2002. — URL: https://docs.python.org/3/c-api/index.html (дата обращения: 16.12.2022).

[107] SWIG : [сайт]. — URL: http://www.swig.org/ (дата обращения: 16.12.2022). — Текст : электронный.

[108] SQLite : [сайт]. — URL: https://www.sqlite.org/ (дата обращения: 16.12.2022). — Текст : электронный.

[109] noVNC : [сайт]. — URL: https://github.com/novnc/noVNC (дата обращения: 16.12.2022). — Текст : электронный.

[110] ocproxy : [сайт]. — URL: https://github.com/cernekee/ocproxy (дата обращения: 16.12.2022). — Текст : электронный.

[111] Simone : [сайт]. — URL: https://github.com/cezarykluczynski/simone (дата обращения: 16.12.2022). — Текст : электронный.

[112] EPICS Archiver : [сайт]. — URL: https://slacmshankar.github.io/epicsarchiver_docs/index.html (дата обращения: 16.12.2022). — Текст : электронный.

[113] PythonIOC : [сайт]. — URL: https://github.com/Araneidae/pythonIoc (дата обращения: 16.12.2022). — Текст : электронный.

[114] Qt framework : [сайт]. — URL: https://www.qt.io/ (дата обращения: 16.12.2022). — Текст : электронный.

[115] Prompt toolkit : [сайт]. — URL: https://python-prompt-toolkit.readthedocs.io/en/master/ (дата обращения: 16.12.2022). — Текст : электронный.

[116] **Weidendorfer, J.** Sequential Performance Analysis with Callgrind and KCachegrind / J. Weidendorfer. — Текст : электронный // Tools for High Performance Computing / под ред. Resch Michael, Keller Rainer, Himmler Valentin [et al.]. — Berlin, Heidelberg : Springer Berlin Heidelberg. — 2008. — P. 93–113. — DOI 10.1007/978-3-540-68564-7_7. — Дата публикации: 24.06.2008.

[117] Developments in ROOT I/O and trees / R. Brun, P. Canal, M. Frank, A. Kreshuk [et al.]. — Текст : электронный // Journal of Physics: Conference Series. — 2008. — Vol. 119. — P. 042006. — DOI 10.1088/1742-6596/119/4/042006. — Дата публикации: 01.07.2008.