НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЕ УЧРЕЖДЕНИЕ

ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ ИМ. Г.И. БУДКЕРА СО РАН

М.Н. Ачасов, А.Г. Богданчиков, В.П. Дружинин,
А.А. Король, С.В. Кошуба

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ СБОРА ДАННЫХ ДЕТЕКТОРА СНД

ИЯФ 2003-59

Новосибирск
2003

Программное обеспечение системы сбора данных детектора СНД

М.Н. Ачасов, А.Г. Богданчиков, В.П. Дружинин, А.А. Король, С.В. Кошуба

Институт ядерной физики им. Г.И.Будкера,
630090, Новосибирск, Россия

Аннотация

Представлено новое программное обеспечение системы сбора данных детектора СНД для экспериментов на коллайдере ВЭПП-2000. Сформулированы требования к системе, приведено описание архитектуры системы, процессов, работающих с потоком событий, процессов контроля и вспомогательных сервисов.

Software for data acquisition system of the Spherical Neutral Detector

M.N. Achasov, A.G.Bogdanchikov, V.P.Druzhinin, A.A.Korol, S.V.Koshuba

Budker Institute of Nuclear Phisics,
630090, Novosibirsk, Russia

New software for data acquisition system of the Spherical Neutral Detector for experiments at VEPP-2000 collider is presented. System requirements are formulated; architecture, data-flow processes, control processes and auxiliary services are described.

Содержание

Введение

В 1999 году было принято решение о модернизации комплекса ВЭПП-2М [1] для повышения светимости и увеличения энергии в системе центра масс с 1.4 до 2 ГэВ. Новый проект получил название ВЭПП-2000.

В связи с этим потребовалась модернизация [2] систем детектора СНД [3]:

- замена трековой системы,
- создание системы идентификации частиц,
- модернизация оцифровывающей электроники детектора,
- модернизация системы сбора данных [4],
- модернизация системы обработки данных.

Важнейшей причиной необходимости модернизации системы сбора данных (ССД) для эксперимента СНД на ВЭПП-2000 явилось увеличение потока сырых событий, считываемых с электроники до 4Мбайт/сек. Это вызвано:

- ослаблением требований в системе отбора первичного триггера для улучшения качества отбора за счет полной реконструкции событий,
- увеличением среднего размера событий в связи с модернизацией электроники для повышения быстродействия и точности измерений,
- повышением светимости ускорителя.

Старая ССД [5] не могла удовлетворить новым требованиям:

- Использование КАМАК ограничивало максимальную скорость чтения событий до 200Гц;
- Старая ССД работала на кластере компьютеров VAX station с операционной системой VMS и тактовой процессора менее 100МГц, которые во многом исчерпали свой ресурс.
- В старой ССД отсутствовала возможность наращивать вычислительные мощности. Система разрабатывалась в начале 90-ых годов прошлого века и основывалась на имеющихся в то время технологиях. После первого запуска системы было сделано множество изменений и дополнений в ССД, призванных сделать ее переносимой и распределенной. Но, несмотря на это, она по-прежнему зависела от устаревшей и не развиваемой в последнее десятилетие платформы VAX-VMS. ССД не была масштабируемой, так как это не было заложено при проектировании.

Со времени создания ССД для СНД на ВЭПП-2М с развитием информационных технологий появились новые возможности для разработки программного обеспечения (ПО) ССД:

Новые возможности	Применение в ССД
Высокопроизводительные компьютеры.	Появилась возможность улучшить отбор событий с помощью полной реконструкции событий в режиме on-line, и значительно ослабить условия отбора в первичном триггере.
Многочисленные системы управления базами данных (СУБД), поддерживающих стандартный интерфейс – язык запросов SQL.	Повышение надежности системы с помощью средств, разработанных для хранения данных с распределенным доступом.
Удешевление носителей информации.	Уменьшение времени доступа к файлам с событиями: появилась возможность хранить большую часть отобранных реконструированных событий на дисках.
Развитие технологий передачи данных.	Чтение электроники через Ethernet, использование разделяемых дисков для хранения и передачи данных.
Развитие объектно-ориентированного программирования.	Новые условия для разработки и поддержки программного обеспечения.

Десятикратное увеличение потока сырых событий, модернизация детектора, изменения в оцифровывающей и считывающей электронике, привели к необходимости разработки и реализации принципиально новой архитектуры ПО ССД детектора СНД на основе новых информационных технологий.

Глава 1

Система сбора данных детектора СНД

1.1 Электроника детектора СНД

Управление подсистемами детектора производится с помощью электроники, которая выполнена в стандарте КАМАК. Сырые события считываются через электронику в стандарте КЛЮКВА [6]. Оцифровывающая электроника СНД занимает 15 крейтов, каждый крейт вмещает 16 информационных плат (ИП), осуществляющих оцифровку сигналов с детектора и две служебные платы: интерфейс первичного триггера и процессор ввода-вывода (ПВВ). По сигналу первичного триггера (ПТ) [7] процессор ввода-вывода, выполненный на основе ПЛИС (программируемая логическая интегральная схема) ALTERA [8], считывает информацию со сработавших каналов и передает ее компьютеру через сеть Ethernet.

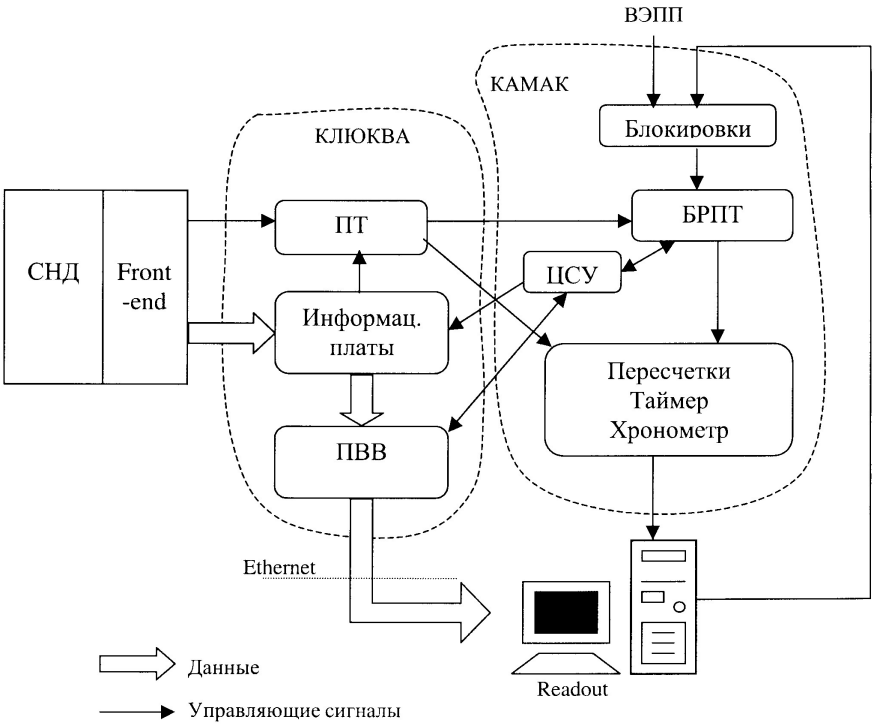

Рис. 1.1. Потоки данных и управляющие сигналы в электронике.

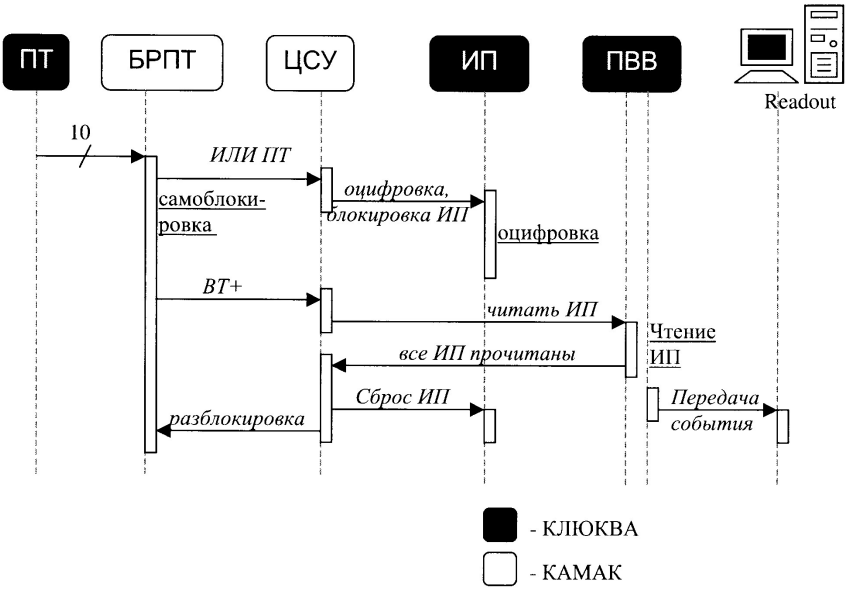

Рис. 1.2. Временная диаграмма чтения событий в электронике: ПТ – первичный триггер, БРПТ – блок решений первичного триггера, ЦСУ – центральная система управления, ИП – информационные платы, ПВВ – процессор ввода/вывода.

Потоки данных и управляющие сигналы в электронике представлены на рисунке 1.1, временная диаграмма чтения событий приведена на рисунке 1.2. Десять сигналов запуска формируются в системе ПТ и поступают на блок решения первичного триггера (БРПТ), где собираются по «ИЛИ». После срабатывания «ИЛИ ПТ» БРПТ блокирует свои входы и выставляет сигнал, который блокирует пересчетные схемы в стандарте КАМАК (пересчетки) и таймер. Сигнал «ИЛИ ПТ» поступает на входы модуля центральной системы управления (ЦСУ) [9], которая раздает управляющие сигналы в КЛЮКВУ: для ИП Т2АМ [10] и ПА24 - сигнал фиксации результатов оцифровки, для ИП А24М [11] - сигнал начала оцифровки. После задержки, достаточной для завершения аналого-цифрового преобразования, БРПТ через ЦСУ передает сигнал на считывание ПВВ. ПВВ читает события с ИП в своем крейте, записывает их во внутренний кольцевой буфер. ПВВ передает данные из буфера в компьютер через сеть Ethernet. Это может происходить одновременно с операцией записи, поэтому на временной диаграмме у ПВВ выделены два потока команд (thread). В случае заполнения кольцевого буфера, поток команд ПВВ, считывающий ИП, блокируется. ЦСУ ожидает прихода сигналов от всех

ПВВ о готовности к чтению ИП. Когда все сигналы выставлены, ЦСУ посылает сигнал сброса на ИП и сигнал разблокировки на БРПТ.

Время, затрачиваемое на оцифровку, составляет около 30 мкс, на чтение информационных плат - около 40 мкс, то есть весь цикл чтения одного события составляет 70 мкс.

Кроме этого, входы БРПТ периодически блокируются:

– компьютером на время чтения пересчеток и таймера (занимает около 1 мс один раз в 15 секунд),

– сигналами от комплекса ВЭПП-2000 (занимает около 40 мс один раз в секунду).

Таким образом, полное мертвое время при частоте событий 1 кГц составляет около 11%.

1.2 Использование ССД

Основные задачи ССД:

– чтение, отбор, и запись событий. Сырые события считываются с электроники, приводятся к формату реконструкции, реконструируются и отбираются. По реконструированным событиям определенного типа проводятся калибровки, которые будут доступны впоследствии для обработки,

– контроль аппаратуры во время захода,

– калибровка электроники и детектора,

– сохранение условий записи событий.

Сбор данных ведется под контролем оператора. Разработка, настройка и детальная диагностика и подсистем выполняется экспертами.

Возможные взаимодействия оператора с ССД:

- Подготовка параметров для запуска захода. Перед запуском оператор выставляет основные параметры захода, как, например, тип захода и условия окончания захода.
- Запуск захода. Все процессы ССД запускаются автоматически на компьютерах предназначенных для работы в on-line. Заход оператор запускает через графический интерфейс. Сразу после запуска захода выполнится подготовка к чтению событий: измерение и загрузка пьедесталов электроники, загрузка команд в ПВВ, открытие файлов, регистрация захода в базе данных. Если подготовка проходит успешно, то начинается чтение событий.
- Пауза в чтении событий. Оператор может приостановить процесс чтения событий, например, в случае кратковременного отсутствия пучков.

- Мониторинг подсистем. Оператор должен иметь возможность следить за состоянием подсистем ССД и детектора.
- Визуализация событий. Оператор должен иметь возможность посмотреть недавно считанные события.
- Контроль. О ходе захода оператор может узнать из журнала оператора. В случае внештатной ситуации система должна привлечь внимание оператора.
- Завершение захода. Чтение сырых событий может быть завершено в трех случаях:
  - выполнение условия остановки, например, когда набрано достаточное число событий,
  - по инициативе оператора,
  - выявление критической ошибки.

  Остановка захода должна сопровождаться звуковым оповещением.
- Генераторные калибровки. Между заходами оператор должен иметь возможность запускать генераторные калибровки электроники.

Возможные взаимодействия экспертов по подсистемам с ССД:

- Контроль над процессами ССД. Эксперт должен иметь средства:
  - просмотра сообщений от процессов системы,
  - запуска и остановки любых процессов ССД,
  - получения информации о том, какие процессы ССД, на каких компьютерах работают, и какими ресурсами владеют,
  - настройка обработки аварийного завершения процесса ССД: звуковое оповещение, автоматический запуск программы восстановления контекста.
- Конфигурирование системы. Эксперты должны иметь возможность оперативно создавать и редактировать конфигурации подсистем детектора и модулей ПО ССД.
- Детальная информация о подсистемах детектора. Детальная информация о подсистемах детектора регулярно сохраняется, а эксперт может ее получить впоследствии.
- Проверка работоспособности системы. Эксперт должен иметь возможность проверить состояние модулей системы.
- Использование сервисов. Для разработки программного обеспечения должны быть предоставлены библиотеки с описаниями, позволяющие использовать сервисы ССД, как, например, интерфейс к журналу оператора.

1.3 Типы данных

В ССД используются следующие типы данных:

- Сырые события. Эти события считываются из электроники и записываются во временный буфер для использования третичным триггером. В качестве временного буфера используется разделяемый диск (SCSI, в перспективе - «fibre channel»), суммарный объем событий – несколько десятков Гбайт.
- События в формате обработки. Отобранные третичным триггером события хранятся на всем протяжении эксперимента для дальнейшей обработки. Суммарный объем - несколько Тбайт. Предполагается, что в связи с удешевлением носителей информации, все отобранные события будут храниться на массиве жестких дисков с обязательным резервным копированием. Это не исключает использования более дешевых носителей информации.
- Состояние электроники. Данные о состоянии электроники СНД и ВЭПП-2000 необходимы для обработки событий и для выяснения причин возможных неполадок. Текущее состояние электроники доступно динамически в базе данных состояния системы, откуда периодически выборочно сохраняются. Носитель – жесткий диск, суммарный объем - несколько десятков Гбайт.
- Калибровочные события. Некоторые типы событий (e^+e^-, $\gamma\gamma$) после реконструкции используются для калибровки on-line. В общем потоке эти события составляют около 10%, третичный триггер сохраняет их на диске, процесс калибровки вносит по ним дополнения в базу данных калибровки, затем они удаляются. Общий объем – несколько Гбайт. Кроме on-line-калибровочных событий есть еще космические калибровочные события, которые пишутся раз в неделю при полном отсутствии пучков. Их полный объем составляет несколько процентов от общего числа событий. Запись и хранение космических событий ведется так же, как и экспериментальных событий.
- База данных калибровок. Результаты калибровок электроники и калибровок по событиям хранятся в базе данных в виде массивов, и используются процессами реконструкции. Общий объем несколько сотен Мбайт.
- База данных состояния системы. Содержит текущую информацию о состоянии системы сбора данных, как то: текущее состояние электроники, состояние процессов системы оператора, основные параметры захода. Доступ осуществляется по имени домена и имени переменной. Общий объем около Мбайта.
- Информация о заходе. На каждой стадии, через которые проходит файл с событиями (чтение, отбраковка, калибровка и реконструкция), процесс, отвечающий за проведение данной стадии, добавляет запись о параметрах и результатах обработки в базу данных заходов. Кроме этой информации, во время захода туда периодически записываются выборки из базы данных состояния системы, называемые «бухгалтерией». Доступ осуществляется по номеру захода и имени подсистемы. Для случаев, когда

используются повторяющиеся данные, такие, как загрузка первичного триггера, предусмотрено использование ссылок на готовые конфигурации подсистем. Общий объем – несколько сотен Мбайт.

- Информация о запуске процессов системы. Информация о том, на каком компьютере, какое число процессов, какие процессы и с какими аргументами должны быть запущены. Объем – несколько Кбайт.
- Журнал оператора. Весь вывод процессов системы производится в лог-журнал оператора. Также имеются функции поиска и возможность добавления сообщений оператором через web – интерфейс. Объем – несколько сотен Мбайт.
- База данных операторов и ответственных за подсистемы лиц. Используется для доступа к операторскому интерфейсу и персональным настройкам, включая подписку на сообщения выбранного типа для рассылки по электронной почте. Объем – несколько Мбайт.

1.4 Архитектура ССД

Поток сырых событий, объемом до 4 МБайт/с (рис. 1.3), вычитывается с ПВВ через Ethernet. Затем события переводятся в формат, используемый в обработке, благодаря чему поток уменьшается до 1 МБайт/с, и проходят полную реконструкцию и отбор, который отбрасывает около 90% событий. В итоге, выходной поток событий, сохраняемый для последующей обработки, составляет около 100 Кбайт/сек. Кроме того, процесс программного отбора отбирает коллинеарные события, по которым выполняется калибровка систем СНД.

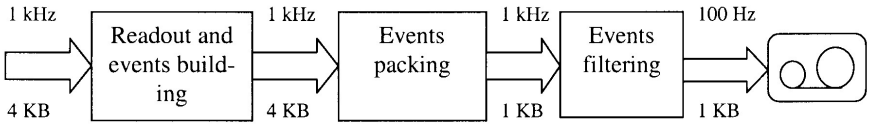

Рис. 1.3. Основной поток событий в ССД.

Особенности обработки потока сырых событий:

– Для уменьшения мертвого времени процесс вычитывания электроники (Readout) должен работать на выделенном компьютере, отдельно от вычислительно емких процессов, к которым относится процесс третичного триггера (L3).
– L3 проводит реконструкцию событий, благодаря чему отбор событий более предсказуем, чем у аппаратного первичного триггера. Для работы L3 требуются значительные вычислительные мощности. Так, обработка одного сырого события занимает около 3 мс на компьютере Athlon с тактовой частотой процессора 1 ГГц. Поэтому желательно распределить работу третичного триггера между несколькими компьютерами.
– Для обработки события должны быть записаны последовательно.

Были рассмотрены два возможных варианта обработки событий:

1) Мультиплексирование сырых событий от Readout (рис. 1.4).

   Считав очередное сырое событие, процесс Readout сразу передает его на обработку одному из L3-процессов. После фильтрации процесс, условно обозначенный на рисунке “Sum”, собирает события и статистику от одного захода воедино.

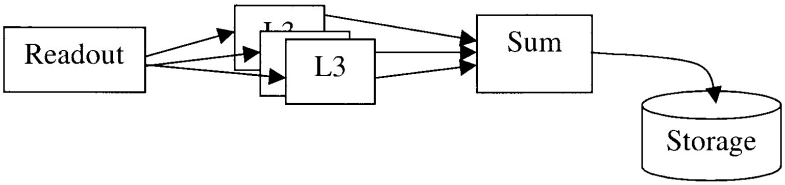

Рис. 1.4. Мультиплексирование событий.

2) Использование буфера для передачи событий (рис. 1.5).

   Во время захода все сырые события Readout пишет в буфер на разделяемый диск, по окончании записи захода освобождает его, после чего любой из свободных L3-процессов может «захватить» буфер и полностью его обработать.

   Этот вариант проще и надежней первого варианта с мультиплексированием: в нем отсутствуют дополнительные операции мультиплексирования и демультиплексирования, и он не требует синхронной передачи сырых событий.

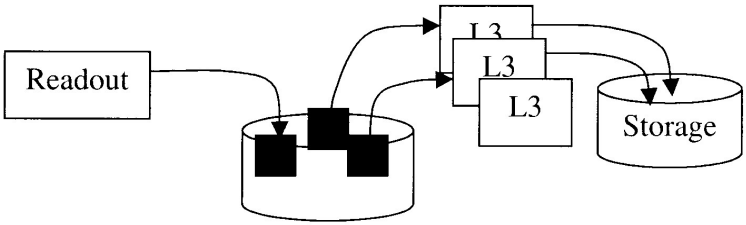

Рис. 1.5: Передача событий через разделяемый буфер.

В итоге был выбран вариант с использованием буферов на разделяемом диске (SCSI) для передачи сырых событий от Readout процессам L3. Такая технология позволяет:

– наращивать вычислительные мощности третичного триггера путем подключения дополнительных компьютеров,

– разделить во времени процесс чтения электроники и программную обработку событий. Отпадает необходимость синхронизовать Readout и L3 процессы для передачи сырых событий. Разделение во времени повышает устойчивость системы к сбоям, так как остановка любой из частей не приводит к остановке всей системы.

Недостатком данного подхода является задержка получения информации, которая может быть получена по реконструированным событиям. Частично этот недостаток компенсируется наличием простейших оперативных систем контроля, которые не используют реконструированные события.

Общая схема системы сбора данных представлена на рис. 1.6. События с электроники детектора после оцифровки считываются процессом Readout с крейтов КЛЮКВЫ. После построения сырых событий Readout сохраняет их в разделяемом буфере, откуда они уже доступны L3-процессам. Небольшую часть событий Readout предоставляет для визуализации в режиме on-line и оперативного контроля за работой электроники. Отобранные L3-процессом «интересные» события сохраняются для обработки в off-line, а калибровочные события - для процесса калибровки.

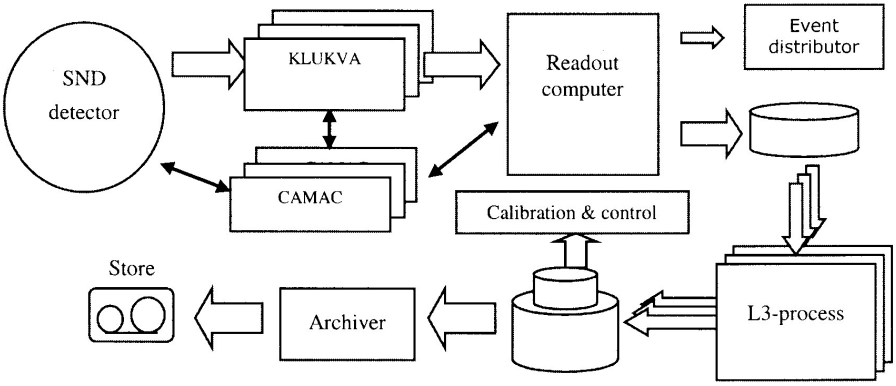

Рис. 1.6. Общая схема ССД с СНД на ВЭПП-2000.

Для согласованной работы процессов ССД требуется система синхронизации. Были рассмотрены два варианта синхронизации процессов:

1) Скроллинг базы данных и файловой системы. Для получения команды или имени ожидаемого ресурса, процесс системы периодически проверяет значение или наличие переменной в базе данных. Недостатками этого метода являются избыточные обращения к базе данных и задержка реак-

ции на изменения состояния. К достоинствам метода можно отнести его простоту – нет необходимости в дополнительных процессах;

2) Использование сервера синхронизации. Этот вариант подразумевает, что все процессы подключены к серверу синхронизации и взаимодействуют через него.
   Стоит отметить, что стандартных объектов синхронизации («семафор», «сигнал», «условие») достаточно для сообщения процессам о приходе команд, но недостаточно для распределения ресурсов (файлов, разделов), когда между процессами требуются более сложные сценарии синхронизации.
   Достоинство этого способа - отсутствие задержек. Существенный недостаток – дополнительная зависимость от процесса синхронизации, который должен отображать свое состояние в базе данных для восстановления контекста после сбоя.

Были опробованы оба метода, и первый выбран за основу как более простой и надежный. Для передачи процессам команд от оператора используется тот же механизм.

1.5 Требования к реализации ПО ССД

ПО ССД СНД разрабатывалось и реализовывалось с учетом следующих общих требований:

- Операционные системы: ПО ССД должно работать в операционной системе Linux и, по возможности, быть как можно менее зависимым от особенностей платформы и операционной системы.
- Хранение данных: для доступа к данным могут использоваться свободно распространяемые системы управления реляционными базами данных, поддерживающие язык запросов SQL. Ориентация на SQL обеспечит переносимость кода и независимость от конкретной СУБД.
- Организация ПО: разрабатываемые модули системы должны удовлетворять общим требованиям к ПО СНД. В первую очередь, это относится к исходному коду и организации пакетов. ПО СНД состоит из одноуровневых пакетов. Каждый пакет должен содержать: документацию, историю изменений, исходный код модулей и тестовых программ, специализированный файл сборки и файл с выходными потоками тестовых программ для выявления непредвиденных изменений в пакете. Все пакеты должны храниться в системе контроля версий CVS. Между пакетами системы допускаются зависимости. Изменения пакета могут привести к тому, что в зависимые пакеты потребуется внести исправления. Работоспособный набор согласованных пакетов называется релизом (release), который создается регулярно согласованно со всеми разработчиками проекта. Если с помощью релиза отбираются и записываются экспериментальные события, то он должен быть доступен и в дальнейшем.

- Модульность системы: ПО ССД должно быть разбито на слабо зависимые модули. Для каждого модуля должен быть определен интерфейс, четко определяющий ожидаемое поведение модуля. Сначала разрабатывается прототип системы, состоящий из модулей-прототипов, которые только имитируют работу, а затем, по мере готовности электроники и с развитием ПО ССД, модули-прототипы дорабатываются до состояния готовности к работе в экспериментах. Так обеспечивается планомерное, пошаговое развитие в соответствии с текущим состоянием и техническими возможностями ССД.
  Некоторые модули могут иметь одинаковый интерфейс и быть полностью взаимозаменяемыми. Например, модуль физической передачи данных через SCSI-диск может быть заменен модулем передачи через сетевой диск, при этом остальные модули останутся без изменений.

- Надежность: ССД детектора представляет собой сложную систему, с которой взаимодействует много людей. Необходимо предусмотреть реакцию системы на нештатные ситуации. Для этого нужны такие средства как: контроль над процессами и устройствами, звуковое оповещение, журнал оператора, диагностика и восстановление, тестирование модулей.

- Пользовательский интерфейс: пользовательский интерфейс должен быть простым в использовании и интуитивно понятным.

- Документирование: каждый разрабатываемый модуль должен иметь два типа документации: для разработчиков и для пользователей. Для разработчиков документация хранится вместе с исходным кодом в пакетах, для пользователей отдельно - в публикациях и web.

- Файловая система: ССД может использовать разделяемую сетевую файловую систему, используемую в группе СНД – NFS (в перспективе – AFS). Это удобно для администрирования, разработки ПО, хранения различных конфигураций, исполняемых программ и библиотек. Тем не менее, ПО ССД не должно жестко зависеть от разделяемой файловой системы.

- Сеть: все компьютеры ССД связаны в сеть, где доступен протокол TCP/IP, по которому можно пересылать данные, необходимые для контроля, конфигурирования и мониторинга.

- Языки программирования: в качестве основного языка программирования для разработки ПО СНД был выбран язык программирования C++. Для конкретных задач язык должен выбираться после оценки наиболее важных критериев:
  - требования к скорости исполнения готовой программы,
  - время разработки,
  - возможности поддержки программы,
  - знание языка разработчиком, требуемое время на обучение,
  - развитие языка, поддержка в будущем.

Глава 2

Компоненты программного обеспечения системы сбора данных

2.1 Основные процессы ССД

К основным процессам системы относятся: процесс чтения сырых событий (Readout), третичный триггер (L3), процесс считывания информации с пересчеток и процесс калибровки.

2.1.1. Чтение сырых событий с электроники - процесс Readout

Процесс Readout через ПВВ выполняет загрузку пьедесталов в ИП и чтение фрагментов событий из ИП. В ПВВ есть три раздела памяти, называемые ОЗУ данных и ОЗУ команд. ОЗУ данных служит для чтения и передачи данных от ИП, а ОЗУ команд - для записи команд ПВВ. Если в условиях запуска захода определена загрузка пьедесталов, то перед началом захода процесс Readout измеряет с помощью генератора необходимые пьедесталы для ИП, загружает их в ОЗУ данных, а затем загружает в ОЗУ команд код для записи пьедесталов. Затем, для инициализации чтения событий, Readout загружает в ОЗУ команд программу чтения фрагментов событий из ИП и передачи их через Ethernet. В случае срабатывания первичного триггера ПВВ читает информацию со сработавших ИП и передает ее Readout. Если в крейте нет сработавших плат, ПВВ посылает пустой фрагмент.

Фрагменты событий от ИП индексируются номером события и номером крейта. Каждый фрагмент состоит из событий с подсистем, где каждое событие индексируется адресом и субадресом. Процесс Readout собирает полное событие из фрагментов, имеющих общий номер события, а затем записывает его в разделяемый буфер. В случае потери значительной части события Readout выдает предупреждение в журнал оператора, а событие выбрасывается.

Процесс Readout регулярно опрашивает базу данных состояний, на предмет наличия новых команд оператора. Для того, чтобы перевести подсистему в новое состояние, оператор через интерфейс оператора изменяет значение переменной, отвечающей за состояние Readout.

Состояния подсистемы чтения электроники и переходы между ними показаны на рисунке 2.1.

Состояния:

- Idle: полная остановка чтения. Сырые события, состояние электроники не считываются.
- Monitoring: события не читаются, работает только отображение состояний электроники. Это основное состояние, когда не идет заход.

– Configuration: подготовка к запуску захода, калибровка электроники. На этом этапе может потребоваться интерактивное взаимодействие с оператором в случае появления предупреждений или ошибок.

– Readout: чтение сырых событий и визуализация состояний электроники. События и состояния электроники пишутся в разделяемый буфер для передачи L3-процессам.

– Pause: приостановка захода. Вся запись останавливается, а работает только визуализация загрузки электроники. Заход при этом не прекращается.

Переходы между состояниями:

– Initialization: запуск процесса, открытие соединений с базами данных, с серверами системы.

– Monitoring: переход к мониторингу электроники.

– Full stop: полная остановка чтения электроники и событий.

– Start: запуск захода.

– Error: произошла фатальная ошибка в электронике при старте захода.

– Success: переход к чтению событий и записи состояний электроники.

– Pause: приостановка захода оператором. Переход может быть использован оператором, например, в случае пропажи пучков.

– Readout: продолжение захода.

– Stop: остановка захода. Переход может произойти как по команде оператора, так и по выполнению любого из условий окончания захода: достижения максимального числа событий, достижение максимального размера файла с событиями, исчерпание лимита времени.

– Auto-start: автозапуск нового захода. Остановка захода по условию и автоматический запуск нового. Переход “auto-start” необходим тогда, когда в обычном режиме требуется частое вмешательство оператора для перезапуска заходов.

Исходный код подсистемы передачи событий (рис. 2.2) разбит по модулям таким образом, что все методы, относящиеся к физической передаче данных, выделены в отдельные абстрактные классы. Благодаря такому подходу, переход к другому способу передачи сырых событий (через разделяемую файловую систему GFS [12], через разделы разделяемых SCSI-дисков, сетевые диски и пр.), не приведет к изменениям в остальном коде и даже не потребует его перекомпиляции.

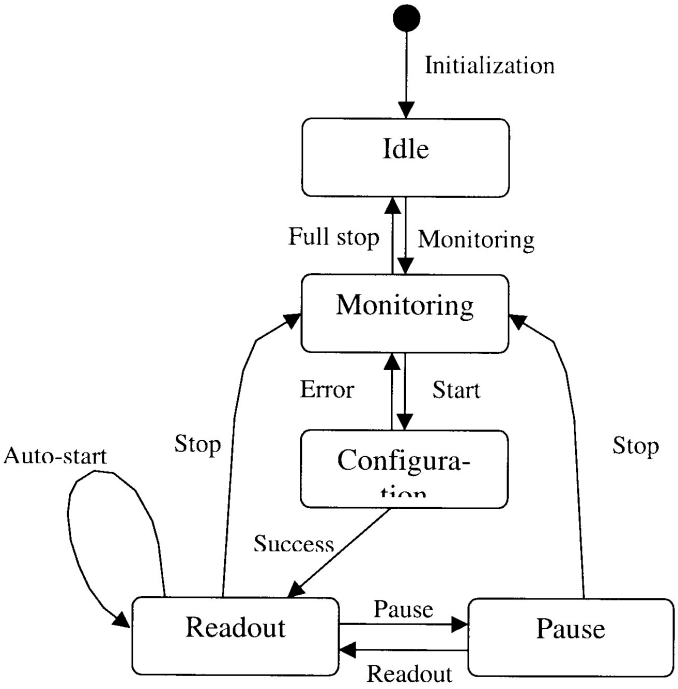

Рис. 2.1. Основные состояния Readout.

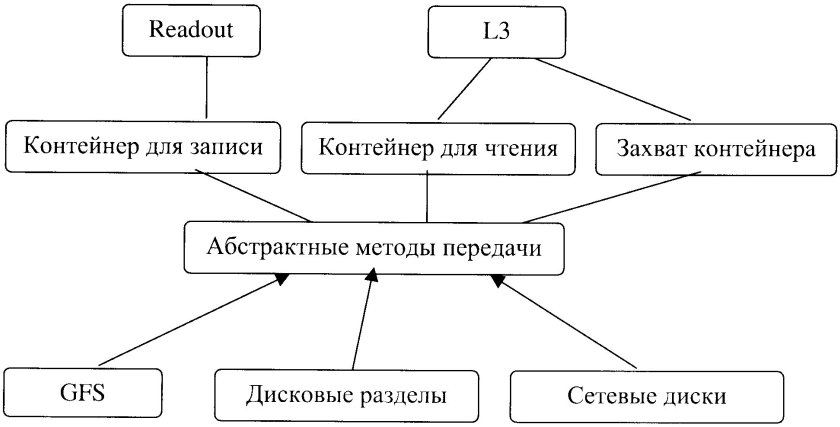

Рис. 2.2. Связь модулей для передачи событий.

2.1.2. Чтение пересчеток

Кроме процесса Readout, непосредственно с электроникой работает процесс Scalers, который считывает с плат пересчеток статистику срабатываний в различных подсистемах электроники. Информация с пересчеток читается периодически (раз в 15 секунд), и записываются в базу данных состояний системы, откуда она будет доступна для мониторинга. Кроме базы данных, примерно раз в минуту эта информация сохраняется в файл. Затем она будет использоваться при реконструкции событий.

Процессом чтения пересчеток управляет процесс Readout через базу данных состояний, аналогично тому, как оператор управляет процессом Readout. Различие заключается в том, что переменная в базе данных состояний используется не только для передачи команд, но и для обратной синхронизации, когда Readout ждет готовности от Scalers. Значения переменной состояний для процесса Scalers:

- Stop - полная остановка, выставляется процессом Readout.
- View - мониторинг, выставляется процессом Readout. В этом состоянии, процесс Scalers обновляет пересчетки в базе данных состояния системы, но в файл не пишет.
- Init - инициализация пересчеток, выставляется процессом Readout при инициализации электроники перед началом записи файла заходов.
- Ready - конец инициализации, выставляется процессом чтения пересчеток. Процесс Readout ждет появления этого состояния, после чего продолжает конфигурацию остальной электроники. После того, как процесс чтения пересчеток выставил состояние Ready, он ожидает появления состояния Run.
- Run - запись пересчеток в файл и обновление базы данных.

2.1.3. Упаковка и фильтрация событий

После стадии считывания сырых событий из электроники процессом Readout и записи в разделяемый буфер, следует стадия, в которой события приводятся к формату off-line обработки, реконструируются и фильтруются. На этой стадии с файлом событий работают два процесса: L3-shell и L3-framework.

Для того чтобы использовать в полном объеме разработки программного обеспечения off-line СНД, процесс L3-framework выполнен в полном соответствии с правилами написания модулей off-line, а все нестандартные для off-line функции: регистрация on-line процесса, получение доступа к файлам из буфера и запуска framework, вынесены в отдельный процесс L3-shell (рис. 2.3).

Процесс L3-Shell должен работать постоянно на всех компьютерах, которые отведены под L3-обработку событий. У этих компьютеров должен

иметься доступ к разделяемому буферу. Процесс L3-shell ожидает появления в разделяемом буфере нового файла с сырыми событиями, после чего захватывает его для обработки процессом L3-framework.

Программа L3-framework выполнена в формате СУМО [13], разработанным для процессов off-line. Такая программа состоит из модулей-классов, каждый из которых должен объявить в методе регистрации, с какими входными данными он будет работать, и какие выходные данные будет производить. По этой информации менеджер framework определяет, в какой последовательности вызывать модули. Так, в L3-framework модули в обработке потока выстраиваются в следующей последовательности:

- чтение сырых событий из буфера,
- перевод сырых событий во внутреннее представление,
- модули программного отбора.

После чтения из буфера процесс L3-framework, используя базу данных конфигурации электроники, упаковывает сырые события. В упакованном формате события используются для реконструкции. В процессе перевода формата:

- вычисляется номер канала по совокупности чисел: номер крейта, номер платы и субадрес в плате,
- осциллограммы сигналов с проволочек, приходящих от плат flash-ADC, пересчитываются в две амплитуды.

За счет этого перевода формата объем данных уменьшается примерно в три раза.

Модули программного отбора выполняют:

- предварительную реконструкцию событий,
- отбор и запись калибровочных событий,
- деление событий - небольшая часть событий записывается не проходя отбора,
- фильтрацию реконструированных событий,
- запись отобранных событий.

Реконструкция, проводимая L3-framework триггером, является предварительной. Окончательная реконструкция проводится с калибровками, которые вычисляются на основе событий, для которых проводится реконструкция. Для сравнения могут сохраняться и события после предварительной реконструкции. Размер реконструированного события составляет около 30% от упакованного, поэтому хранение реконструированных событий не повысит значительно требований к носителям информации.

Калибровочные события, такие, как $e^+e^- \to e^+e^-$ и $\gamma\gamma$, выделяются процессом программного отбора событий и записываются в отдельный файл, который позже будет обработан процессом калибровки.

Небольшая часть упакованных событий пропускается без применения фильтрации, и впоследствии может быть использована для оценки работы процесса отбора.

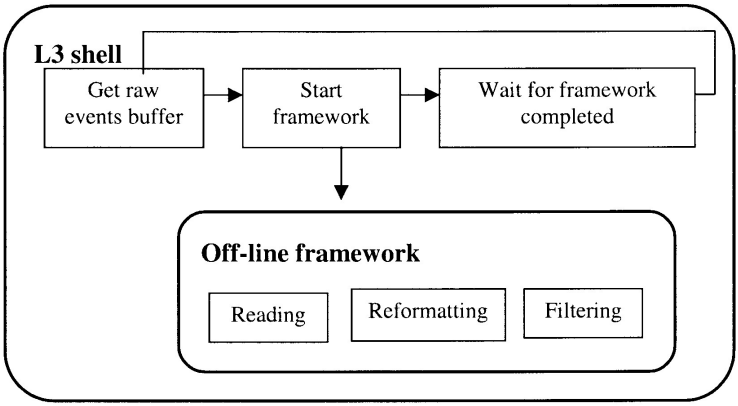

Рис 2.3: Структура L3.

2.1.4. Процесс калибровки

Как и L3-framework, программа калибровки выполнена в системе СУМО [13]. По калибровочным событиям, получаемых от процесса L3-framework, процесс калибровки набирает статистику, на основе которой вычисляются калибровочные коэффициенты и находятся отклонения от нормальной работы в электронике. Когда обработано заданное число событий одного типа, процесс калибровки добавляет новую запись в базу данных калибровок.

Для корректного набора статистики процесс калибровки должен обрабатывать файлы с калибровочными событиями в строгой последовательности. Набор статистики, необходимый для получения калибровки, завершается после обработки определенного числа калибровочных событий. Кроме того, окончание набора статистики должно совпадать с окончанием обработки файла с калибровочными событиями, которые были отобраны процессом L3-framework из одного захода. В отличие от модуля чтения в процессе L3, модуль чтения в процессе калибровки последовательно читает файлы с калибровочными событиями, самостоятельно определяя имя следующего файла. После того, как файл обработан, процесс калибровки сбрасывает в служебный файл имя обработанного файла, число обработанных событий каждого типа и набранную статистику. Только после создания служебного файла обработанный файл с калибровочными событиями удаляется. При запуске процесс калибровки читает служебный файл и продолжает набирать сохранен-

ную в нем статистику. Этим обеспечивается устойчивость к сбою системы и непредвиденному завершению процесса калибровки.

Процесс калибровки получает события через разделяемые диски, аналогично тому, как получает события L3 от Readout. Только в этом случае поток событий значительно меньше, и при максимальной загрузке составляет около (1КГц*1Кбайт)/10=100КБайт/сек. Поэтому для передачи файлов можно использовать не только разделяемые диски, но и более дешевые технологии, как, например, сетевую файловую систему (NFS, AFS).

2.2 Управление системой

Для управления и контроля в системе предусмотрен графический интерфейс оператора, модули контроля над подсистемами детектора, система запуска и остановки процессов, система распределения и восстановления ресурсов, процесс информирования оператора и экспертов СНД.

2.2.1. Операторский интерфейс

Реализован графический интерфейс оператора на основе динамических web-страниц.

Интерфейс оператора состоит из следующих окон:

- Выборка сообщений из журнала оператора.
- Добавление нового сообщения в журнал оператора.
- Управляющая панель – запуск, остановка, приостановка захода и полная остановка. Там же есть выключение сирены и открытие окон к прочим возможностям оператора.
- Изменение условий захода.
- Запуск необходимых процессов в системе.
- Получение информации о работающих и зарегистрированных процессах в системе.
- Изменение личных настроек – цветовых настроек и log-журнала.
- Получение данных по пользователям СНД с возможностью внесения частичных изменений в собственные данные.
- Изменение подписки на сообщения.

2.2.2. Контроль подсистем детектора

Медленный статистический контроль за подсистемами детектора выполнен в виде модулей процесса калибровки. По статистике срабатываний определяются неверно работающие подсистемы.

На стадиях чтения сырых событий и отбора проводится контроль на целостность полученных данных, как, например, диапазон значений и размер.

Быстрый контроль осуществляется с помощью:

– специальных программ оперативного контроля электроники, которые получают события от процесса Readout,
– визуализации событий от детектора,
– визуализации загрузок электроники, которая обновляется несколько раз в минуту.

2.2.3. Контроль над процессами системы

Процесс Starter

Все процессы в on-line системе запускаются процессом, работающим на каждом из компьютеров on-line системы – процессом Starter. Сам процесс Starter запускается как один из процессов-демонов при запуске операционной системы. Как и все процессы ССД Starter регистрируется на сервере-распределителе доменов.

С помощью процесса Starter оператор может запускать и останавливать процессы системы. Это осуществляется путем внесения изменений в таблицу, доступную через web – интерфейс. В интерфейсе отображены поля таблицы:
– Host – имя компьютера.
– Process – тип процесса .
– Command – команда, выполняющаяся для запуска процесса.
– Wanted – желаемое число процессов данного типа.
– Editor – оператор, сделавший последние изменения.

Через определенный интервал времени Starter читает из базы данных список процессов, которые должны работать на данном компьютере, а затем проверяет, какие из этих процессов реально работают. Если число запущенных процессов данного типа больше (или меньше) заданного в списке числа, то starter «убивает» лишние (или запускает) процессы. Отсутствующие в списке процессы Starter не контролирует, поэтому для остановки определенного процесса нужно установить число процессов данного типа в ноль.

Особенности реализации:

– Оптимизирован вывод в журнал оператора в случае неудачного запуска процесса. Журнал оператора не перегружается избыточными сообщениями о повторных неудачных попытках.

– Процесс starter, как и все остальные процессы on-line, после запуска должен получить имя рабочего домена у сервера - распределителя доменов, и только после этого продолжать работу. Возникает сложность, когда сам процесс - распределитель доменов должен быть запущен стартером. Специально для такого случая starter сначала проверяет, на каком компьютере должен работать его распределитель доменов, и если на том же самом, то starter использует отложенное получение имени домена – только после первого цикла проверки/запуска процессов.

Процесс Domain Distributor

При регистрации процесс системы должен получить имя домена у сервера распределителя доменов “Domain Distributor” (DD). Имя домена имеет формат: «PROCESS-ТИП-ИД», где «ТИП» - тип процесса, а «ИД» - отличительный номер для домена процессов одного типа. Так как каждому процессу ССД при запуске должен выдаваться домен, то число доменов не должно быть меньше процессов соответствующего типа. Процесс использует домен с именем, полученным от DD, для хранения информации о собственном состоянии в базе данных состоянии системы. Обрыв связи с DD расценивается как непредвиденное завершение процесса, и сервер выполняет программу, определенную процессом для восстановления контекста после сбоя. Путь к ее местонахождению должен быть записан в домене процесса под фиксированным именем “RECOVER_PATH”. В журнал оператора DD запишет сообщение об ошибке с атрибутом сирены, чтобы обратить внимание оператора.

Возможна ситуация, когда непредвиденно закончит работу сам процесс DD. Для такого случая специально разработана схема перезапуска, в которой поддерживается непрерывная работа процессов системы. После потери связи с DD процесс системы регулярно делает попытки соединения с сервером, и если процессу это удается, то он перерегистрирует имя своего домена. Сервер DD всю свою информацию о работающих процессах системы кэширует в соответствующих доменах, в специальной переменной под фиксированным именем “STATE_INFO”. По значению этой переменной можно определить, занят ли домен, находится ли в стадии восстановления контекст, соответствующий данному домену, действительна ли информация о занятости домена. После запуска DD кэширует эту информацию, помечая занятые домены, как недействительные. Если в течение определенного интервала времени (15 секунд) процесс не перерегистрирует имя домена, то домен будет помечен как восстанавливающийся и запущена программа восстановления.

Состояние доменов системы и переходы между ними представлены на рис. 2.4.

Есть четыре состояния домена, которые отображаются в зарезервированной специально для этого переменной домена:

- Empty – домен не занят, и может быть использован любым новым процессом соответствующего типа.
- Busy – домен используется процессом.
- Recover – по информации, оставленной в домене, производится восстановление контекста процесса.
- Confirmation - ожидание перерегистрации процесса.

Переходы между состояниями домена:

- Process start – инициализация процесса; процесс получает имя домена, с которым работает в дальнейшем;
- Process exit – корректное завершение процесса; домен освобождается для дальнейшего использования;
- Process crash – непредвиденное окончание процесса, может быть следствием, например, сбоя в программе. В таком случае выдается сообщение в журнал оператора, и запускается процесс восстановления, путь к которому должен содержаться в переменной домена под зарезервированным именем;
- Distributor restart after crash – распределитель доменов, как и любой другой процесс в системе, может быть непредвиденно прерван. В таком случае процессы не останавливают работу, одновременно ожидая появления распределителя;
- Confirmation timeout - сразу после старта распределитель доменов переводит домены, которые находились в состоянии “Busy” в состояние “Confirmation”. Работающие процессы в течение определенного времени (порядка 15 секунд) должны перерегистрировать свое право на работу с доменом. По истечении этого времени считается, что все процессы, не перерегистрировавшие свои домены, непредвиденно закончили работу, и все такие домены переводятся в состояние “Recover”;
- Process alive – после перезагрузки распределителя доменов процесс перерегистрируется, и его домен переводится в состоянии “Busy”.

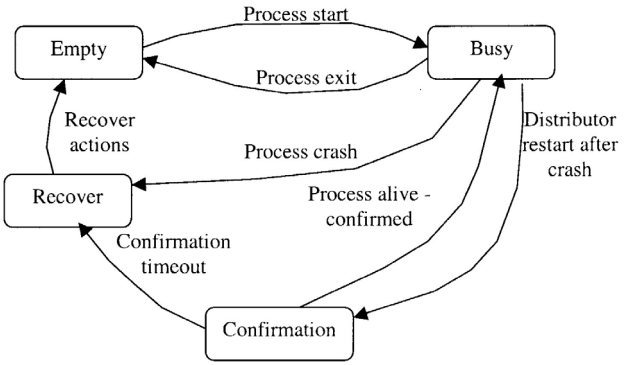

Рис. 2.4: Схема состояний

Процесс Informer

У процессов весь вывод представлен в виде сообщений, который хранится в журнале оператора (log-журнал). Просмотреть журнал в любой момент можно через web. Для того чтобы немедленно обратить внимание оператора на важнейшие сообщения и оповестить ответственных за подсистемы лиц, в системе постоянно работает процесс Informer - анализатор вновь приходящих

сообщений в журнал оператора. После прихода нового сообщения анализатор выполняет следующие действия:

– Проверяет, установлен ли у сообщения флаг сирены, и если установлен, то informer запускает процесс-сирену. Сирена работает до тех пор, пока ее не выключит оператор (выключение доступно через web-интерфейс);
– Выбирает из базы данных пользователей, подписавшихся на данный тип сообщений. Этим пользователям informer высылает копию сообщения по e-mail. Таким образом, эксперт подсистемы автоматически получает электронную почту о случившихся неполадках.

2.3 Сервисы системы

В системе разработаны вспомогательные компоненты, обеспечивающие синхронизацию процессов, а также передачу, хранение и доступ к данным.

2.3.1. Раздатчик событий

Процессам визуализации необходимы события, которые в данный момент читаются с электроники. Процедура раздачи событий не должна существенно замедлять основную работу процесса Readout - чтение с ПВВ и запись событий в разделяемый буфер. Это выполнимо, так как процессам визуализации требуется лишь незначительная часть основного потока. А так как нет строгих требований к выборке событий, то передачу можно организовать по запросу, где процесс Readout выступает в роли сервера, а процессы визуализации - в роли клиентов.

После записи очередного сырого события в разделяемый буфер Readout проверяет, есть ли запросы на подсоединение. Если есть, то Readout открывает новые каналы для передачи сырых событий. Затем Readout проверяет, есть ли запросы на событие, и если есть, то передает в соответствующие каналы событие, до этого записанное в разделяемый буфер.

Были разработаны два варианта раздачи сырых событий – по сети TCP/IP и локально через разделяемую память.

В дополнение была разработана функция ограничения потребления процессом процессорного времени. Она применима к процессам с циклами, где между двумя входами в цикл выполняется примерно одинаковое количество операций. В точке входа в цикл ставится вызов функции, которая подсчитывает время CPU, затраченное на выполнение кода процесса и время, прошедшее с момента прошлого входа в цикл. Их отношение сравнивается с требуемым потреблением времени CPU, выраженному в процентах. Если истрачено больше времени то процесс «засыпает» на время, необходимое для достижения требуемого уровня загрузки.

С помощью такого ограничения клиенты уменьшают число запросов к серверу и его загрузку. Это наиболее актуально для ситуации, когда процесс-раздатчик и процессы-потребители событий работают на одном компьютере.

2.3.2. Хранение данных

Для длительного хранения больших объемов данных, таких, как события для последующей реконструкции, используются диски с обязательным резервным копированием. Информация о местонахождении файлов с событиями и резервных копиях хранится в базе данных описаний заходов.

Данные небольшого объема, требующие быстрого доступа, хранятся в базах данных. Для каждого из типов данных, таких как параметры, калибровки, описания заходов, журнал оператора, информация о состоянии системы, разработаны структуры данных и соответствующие интерфейсы для доступа. За базовую основу взята реляционная СУБД MySQL. Это позволяет использовать разработанный для нее инструментарий по резервному копированию, администрированию, редактированию. Использование стандартизованного языка SQL позволяет минимизировать затраты в случае необходимости перейти на другую SQL-ориентированную СУБД.

Большинство интерфейсов, таких, как интерфейсы к калибровкам, конфигурациям, журналу оператора, предназначены для очень широкого круга пользователей: разработчиков on-line, специалистов по подсистемам, ответственных за калибровки и реконструкцию. Поэтому важнейшими требованиями к реализации этих интерфейсов является сокрытие от пользователей обращений к базе данных, независимость от конкретной реализации и простота использования.

Калибровки

Интерфейс к калибровкам написан на языке C++. Каждая из калибровок представляет собой массив чисел одного типа. Поиск калибровки в базе данных производится по ключу, состоящему из типа, времени действия и версии калибровки. Кроме этого, имеется возможность выборки калибровок по отдельным каналам (индексам массива).

Параметры системы и описания заходов

Доступы к параметрам подсистем и к описаниям заходов тесно связаны и потому имеют единый интерфейс.

Для запуска заходов ответственные по подсистемам должны подготовить необходимые конфигурации. Если меняются параметры подсистемы, то создается конфигурация с новым номером версии. Перед запуском захода в описания заходов добавляется новая запись. Процессы на каждом из этапов обработки пополняют эту запись информацией о том, какие были использованы конфигурации.

Описания заходов индексируются номером захода и состоят из набора версий - ссылок на конкретные конфигурации подсистем.

Для обеспечения обратной совместимости, доступно только расширение структуры конфигурации, а изменение или удаление - запрещено.

Журнал оператора

Инструментарий доступа к журналу оператора состоит из нескольких интерфейсов, написанных на языках PHP и C++, позволяющий просто и удобно выбирать, добавлять и ожидать сообщения. Каждое сообщение имеет атрибуты:

- Сирена – включена, если требуется включить сирену, чтобы немедленно привлечь внимание оператора.
- Дата и время сообщения.
- Раздел, – к какому разделу относится сообщения (напряжение питания, запуск процессов и пр.).
- Тип сообщения (ошибка, фатальная ошибка, предупреждение, информация, трассировка, отладка).
- Отправитель: пользователь, хост, процесс.
- Сообщение – краткое содержание сообщения. Это основное поле.
- Дополнения к сообщению произвольного MIME-типа. В дополнении может быть приведено полное описание проблемы, рисунки, диаграммы, графики и пр. Каждое дополнение имеет атрибуты: тип, размер и ”развернутость”. Атрибут «развернутость» является рекомендацией для программ просмотра log-журнала. Означает, требуется ли включить дополнение в тело сообщения или достаточно отобразить только ссылку на него.

По этим полям можно отсортировать желаемые сообщения. Добавление сообщения и выборка сообщений доступна через пользовательский интерфейс в web и через программный интерфейс, написанный на языке C++.

База данных пользователей

Интерфейс к базе данных пользователей реализован на PHP, позволяет каждому пользователю, зарегистрированному в системе, найти информацию о других и частично изменить информацию о себе. Информация содержит: имя, отчество, фамилию, комнату, телефон и адрес электронной почты. Почтовый адрес может быть использован для подписки на сообщения системы. Каждый пользователь имеет возможность подписаться на новые сообщения по интересующей теме с определенным уровнем важности.

Текущая информация о состоянии системы

Значения переменных с информацией о состоянии системы хранятся в базе данных и индексируются именем домена и именем переменной. Каждому процессу ССД отводится отдельный домен, в котором он хранит ключевую информацию о своем состоянии. Некоторые переменные, характеризующие состояние, необходимо менять и читать только синхронно, для чего реализованы нераздельное обновление и нераздельное чтение нескольких переменных одного домена.

Реализованы интерфейсы на C++ и PHP. Реализован полный доступ через командную строку, а также редактирование отдельных доменов через web-интерфейс.

Заключение

В рамках проекта модернизации ССД были определены следующие цели:

- оценка применимости существующего инструментария для систем сбора данных в специфических условиях эксперимента СНД;
- разработка архитектуры системы сбора данных, как в целом, так и ее частей в отдельности;
- реализация и тестирование системы сбора данных.

Для достижения вышеперечисленных целей были поставлены и решены следующие задачи:

- изучение особенностей детектора и электроники СНД;
- выявление основных требований, определяющих архитектуру и состав ССД;
- поиск и изучение существующих готовых решений для ССД, оценки их применимости, выяснение мнений и пожеланий пользователей ССД с СНД;
- изучение и оценка возможных реализаций ССД, оптимальный выбор физической и программной архитектуры ССД;
- проверка пропускной способности разделяемого буфера;
- разработка и создание прототипа ССД;

В ближайшее время планируется интеграция прототипа ССД с новой электроникой детектора СНД и ВЭПП-2000.

Авторы выражают благодарность за многочисленные обсуждения и предложения Д.А.Букину, В.Б.Голубеву и С.И.Середнякову.

Работа частично поддержана грантами Президента РФ по поддержке ведущих научных школ НШ-1335.2003.2, РФФИ 03-02-16581-а, РФФИ 02-02-16348-а, Лаврентьевского конкурса молодежных проектов СО РАН 2002г «Разработка и создание трековой системы для экспериментов с детектором СНД на коллайдере ВЭПП-2000».

Литература

[1] Skrinski A.N. VEPP-2M status and prospects and ϕ-factory project at Novosibirsk. / A.N. Skrinski // Proc. of Workshop on physics and detectors for DAΦNE 95, Frascati, April 4-7, 1995. – INFN – Laboratori Nazionali di Frascati, 1995. – Frascati physics series. – Vol. IV. – P. 3 – 18.

[2] Проект модернизации детектора СНД для эксперимента на ВЭПП-2000 / Г.Н. Абрамов, В.М. Аульченко, М.Н. Ачасов и др. // Препринт ИЯФ 2001-29, Новосибирск, 2001.

[3] Сферический нейтральный детектор (СНД) для электрон-позитронного накопителя ВЭПП-2М. / В.М. Аульченко, М.Н. Ачасов, С.Е. Бару и др. // Препринт ИЯФ 99-16, Новосибирск, 1999.
Spherical Neutral Detector for VEPP-2M collider. / M.N. Achasov, V.M. Aulchenko, S.E. Baru et al. // Nucl. Instr. Meth. A449 125 (2000)

[4] Data Aquisition System for the SND2000 Experiment / M.N. Achasov, A.G. Bogdanchikov, D.A. Bukin et al. // In proc. of CHEP2001, Sept. 3-7, 2001, Beijing, (2001) 556-559.

[5] Data Acquisition System of SND Experiment / D.A.Bukin, T.V.Dimova, V.P.Druzhinin et al // In proc. of CHEP97, Berlin, 1997.

[6] V.M. Aulchenko et al, Nucl. Instr. and Meth. A409 (1998) 639.

[7] Первичный триггер детектора СНД на ВЭПП-2М / Д.А. Букин, Ю.С. Великжанин, В.Б. Голубев и др. // Препринт ИЯФ 98-29, Новосибирск, 1998.

[8] FLEX 10K Embedded Programmable Logic Family Data Sheet / ALTERA Corporation, 2003 http://www.altera.com/literature/ds/dsf10k.pdf

[9] Служебные блоки системы сбора данных КЛЮКВА / С.Е. Бару, В.С. Кириченко, Г.А. Савинов и др. // Препринт ИЯФ 88-26, Новосибирск, 1988.

[10] Информационные платы ТП, del`taT и T2A системы сбора данных КЛЮКВА / Аульченко В.М., Байбусинов Б.О., Титов В.М // Препринт ИЯФ 88-22, Новосибирск, 1988.

[11] Информационная плата A32 системы сбора данных КЛЮКВА / Аульченко В.М., Леонтьев Л.А., Усов Ю.В. // Препринт ИЯФ 88-30, Новосибирск, 1988.

[12] O'Keeffe M. Accelerating technical computing with Sistina Global File System - based storage clusters / M. O'Keeffe // Systina Software Inc., 2001, *http://www.sistina.com/downloads/whitepapers/Tech_WP102A.pdf*

[13] SND Off-line Framework / D.A. Bukin, V.N. Ivanchenko, A.A. Korol et al. // Proc. of CHEP2001, Sept. 3-7, 2001, Beijing, (2001) 145-148.

М.Н. Ачасов, А.Г. Богданчиков, В.П. Дружинин,
А.А. Король, С.В. Кошуба

Программное обеспечение системы сбора данных детектора СНД

M.N. Achasov, A.G.Bogdanchikov, V.P.Druzhinin,
A.A.Korol, S.V.Koshuba

Software for data acquisition system of the Spherical Neutral Detector

ИЯФ 2003-59

Ответственный за выпуск А.М. Кудрявцев
Работа поступила 17.09. 2003 г.

Сдано в набор 18.09. 2003 г.
Подписано в печать 19.09.2003 г.
Формат 60x90 1/16 Объем 2.0 печ.л., 1.6 уч.-изд.л.
Тираж 125 экз. Бесплатно. Заказ № 59

Обработано на IBM PC и отпечатано
на ротапринте ИЯФ им. Г.И. Будкера СО РАН,
Новосибирск., 630090, пр. Академика Лаврентьева, 11