УДК 681.518
DOI 10.25205/1818-7900-2020-18-3-69-80

Программное обеспечение для системы архивирования данных ЛИУ-20

В. Е. Ситнов, А. И. Сенченко, Г. А. Фатькин

Новосибирский государственный университет
Новосибирск, Россия
Институт ядерной физики им. Г. И. Будкера СО РАН
Новосибирск, Россия

Аннотация

Система управления рентгенографическим комплексом ЛИУ-20 основана на фреймворке Tango Controls. В ходе данной работы была доработана подсистема архивации данных HDB++ для эффективного хранения осциллограмм. Была добавлена поддержка СУБД PostgreSQL в качестве хранилища. В процессе работы была добавлена поддержка бинарного режима передачи данных в расширение PostgreSQL pguint для беззнаковых чисел. Были проведены тесты производительности модифицированного расширения pguint и нового интерфейса СУБД. Такое решение позволило значительно повысить производительность работы HDB++, снизить объём хранимых данных в базе, а также снять ограничения по размеру сохраняемых данных.

Ключевые слова

базы данных, ЛИУ-20, Tango Controls

Для цитирования

Ситнов В. Е., *Сенченко А. И.*, *Фатькин Г. А.* Программное обеспечение для системы архивирования данных ЛИУ-20 // Вестник НГУ. Серия: Информационные технологии. 2020. Т. 18, № 3. С. 69–80. DOI 10.25205/1818-7900-2020-18-3-69-80

Archiving System Software for LIA-20

V. E. Sitnov, A. I. Senchenko, G. A. Fatkin

Novosibirsk State University
Novosibirsk, Russian Federation
Budker Institute of Nuclear Physics SB RAS
Novosibirsk, Russian Federation

Abstract

The control system of the X-ray complex LIA-20 is based on the Tango Controls framework. Data archiving subsystem HDB++ was improved during the work for efficient storage of waveforms. PostgreSQL DBMS support as a storage was added. PostgreSQL extension “pguint” for unsigned numbers support was modified to support the binary transfer mode during this work. Performance testing of the modified “pguint” extension and the new DBMS interface was performed. This solution allowed to significantly increase the performance of HDB++, reduce the amount of data stored in the database, as well as remove restrictions on the size of the stored data.

Keywords

databases, LIA-20, Tango Controls

For citation

Sitnov V. E., Senchenko A. I., Fatkin G. A. Archiving System Software for LIA-20. *Vestnik NSU. Series: Information Technologies*, 2020, vol. 18, no. 3, p. 69–80. (in Russ.) DOI 10.25205/1818-7900-2020-18-3-69-80

© В. Е. Ситнов, А. И. Сенченко, Г. А. Фатькин, 2020

Введение

Рентгенографический комплекс ЛИУ-20 строится на базе линейного индукционного ускорителя, спроектированного на три последовательных пучка электронов с энергией до 20 МэВ. Максимальная частота выстрелов составляет 0,1 Гц [1]. ЛИУ-20 состоит из инжектора и ряда ускоряющих секций, источниками питания которых служат модуляторы – импульсные высоковольтные устройствами. Всего на установке 480 модуляторов, которые являются основными контролируемыми элементами, помимо них также необходимо управлять: источниками питания линз, вакуумом, импульсными трансформаторами, а также фиксировать данные с датчиков тока и положения пучка, и управлять рядом других устройств. Полная длина ускорителя составляет примерно 140 м, а управляющие модули расположены вдоль установки и выполнены на базе VME-крейтов соединённых между собой при помощи сети Ethernet. Все данные, получаемые с установки можно разбить на несколько классов по частоте поступления и объему:

Быстрые измерения. Осциллограммы длительностью до 10 мкс: напряжения на индукторах, сигналы с полосковых датчиков положения пучка и с трансформаторов тока.

Медленные измерения. Осциллограммы длительностью до нескольких миллисекунд: напряжение на зарядных устройствах, токи размагничивания, напряжение накала высоковольтных ключей и другие.

Каналы синхронизации. Обеспечивают запуск элементов установки в заданном порядке.

Каналы блокировок. Блокируют запуск ускорителя или подсистем в случае нештатной ситуации.

Технологический контроль. Данные о состоянии вакуума, источников питания, управление мишенного узла.

В табл. 1 приведены объемы данных, генерируемых системой управления [2].

Таблица 1

Объёмы передаваемых данных

Table 1

Transmitting data volumes

Тип каналов	Число каналов		Объем данных	
	на крейт	по всей системе	на крейт	по всей системе
Быстрые	22	594	214 КБ/выстрел	5,7 МБ/выстрел
Медленные	55	1485	0,5 МБ/выстрел	13,5 МБ/выстрел
Синхронизация	55	1485	0,5 КБ/выстрел	13,5 КБ/выстрел
Блокировки	55	1485	0,5 КБ/выстрел	13,5 КБ/выстрел
Технологический контроль	~40	1000	19 КБ/мин	513 КБ/мин
Всего	~280	6000	715 КБ/выстрел и 19,5 КБ/мин	19,3 МБ/выстрел и 540 КБ/мин

Все эти данные необходимо сохранять и анализировать для настройки электронно-оптического тракта с целью достижения оптимальных параметров пучка, а также для анализа и увеличения надёжности работы установки и отдельных её элементов, ведь проводимые эксперименты являются чрезвычайно дорогостоящими [3]. Как можно видеть из табл. 1, большую часть данных, которые необходимо анализировать, составляют осциллограммы (массивы точек).

Для ускорителей и прочих экспериментальных установок есть несколько программных каркасов систем управления, наиболее распространенными из которых являются: EPICS,

TANGO, FESA. Все они создавались в основном для синхротронов и других кольцевых ускорителей. В таких системах основными параметрами, которые необходимо контролировать, как правило, представляют собой отдельные значения, изменяющиеся с частотой не более чем 0,1 Гц. Большинство осциллографических измерений на таких установках поддаются простой обработке и представлению в виде ряда одиночных переменных. К примеру, осциллограммы с датчиков положения пучка представляются в виде координат центра масс: X, Y, заряда пучка C и его временной протяженности. А количество осциллограмм, которые сохраняются полностью, не очень велико. В связи с этим и разработанные системы хранения в основном ориентированы на хранение таких отдельных измерений.

На ЛИУ-20 встала проблема сохранения большого количества осциллограмм, которая в общем виде не решена ни в одном из распространенных программных каркасов. Программной основой системы управления ускорителем ЛИУ-20 является Tango Controls, система архивации данных которой называется HDB++. Для него в ходе этой работы было разработано решение, позволяющее эффективно хранить большое количество осциллограмм в СУБД PostgreSQL. Начнем наше рассмотрение с устройства системы архивирования Tango.

Система Tango Controls

Tango Controls – распределенная система управления устройствами и сбора данных с открытым исходным кодом, основанная на CORBA. Система имеет клиент-серверную объектно-ориентированную архитектуру [4].

Рассмотрим основные составные части Tango Controls, необходимые для понимания предлагаемого решения.

База данных конфигурации Tango. Используется для индексации и поиска остальных объектов в сети, а также для хранения параметров реализаций серверов устройств.

Tango-класс устройства. Описывает протокол взаимодействия с устройством, его атрибуты, параметры, статусы и т. д.

Tango-устройство. Это конкретный экземпляр Tango-класса.

Сервер устройств. Это процесс, который является контейнером для различных Tango-устройство.

Устройства обладают набором параметров, атрибутов и команд, определяемых классом устройства.

Параметры. Эти значения задаются при создании экземпляра устройства и используются для инициализации устройства. Типичными примерами могут быть: порт устройства, адрес устройства на шине, режим работы.

Атрибуты. Это величины, которые в зависимости от типа доступа (на чтение, на запись, на чтение и запись) могут изменяться сервером или клиентом. Атрибуты позволяют вести взаимодействие с устройством во время работы и отражают состояние устройства. Например, показания АЦП.

Команды. Позволяют выполнять различные действия с устройством. Могут иметь один входной и один выходной аргумент.

Параметры, атрибуты и аргументы команд могут быть одного из следующих типов, назначение которых в основном понятно из названия: *DevBoolean*, *DevShort*, *DevLong*, *DevFloat*, *DevDouble*, *DevUChar*, *DevUShort*, DevULong, *DevULong64*, *DevLong64*, *DevString*, *DevEnum*, *DevState*. Есть также и тип *DevEncoded*, использующийся для представления данных, не укладывающихся ни в один из перечисленных типов; представляет собой строку и бинарные данные.

Все величины могут быть одного из следующих форматов: скаляр (единичное значение), спектр (одномерный массив значений), изображение (двумерный массив). Стоит заметить, что осциллограмма, считываемая АЦП, в системе Tango Controls может быть удобно представлена одним атрибутом с форматом спектра или же двумерным массивом, где будут ото-

бражены пары значение-время измерения. В случае если значение атрибута не может быть установлено, например при обрыве связи с физическим устройством, атрибут также может хранить исключение.

Для удобства контроля за показаниями устройств, Tango Controls позволяет для каждого атрибута настроить сопутствующие параметры, включающие в себя описание, единицы измерения и два диапазона, выход за границы которых изменяет специальное поле атрибута – качество.

Начиная с версии 8.1 в Tango Controls появилась подсистема событий, основанная на протоколе ZeroMQ. Она позволяет клиентам реагировать на изменение данных, не проводя постоянный опрос устройств. Существует несколько типов событий:

Periodic – генерируется непрерывно с определенным временным интервалом.

Change – генерируется при изменении значений на заданную относительную или абсолютную величину.

Archive – может генерироваться периодически или по изменению. Используется системой архивирования.

Важной особенностью событий является то, что они могут быть сгенерированы как автоматически, так и непосредственно в коде разработчиком. Рассмотрим теперь устройство системы архивирования данных HDB++.

Система архивирования HDB++

Система HDB++ позволяет сохранять установленные и прочитанные значения атрибутов при наступлении специального события – archiving event, которое может возникать с установленной для каждого атрибута частотой либо генерироваться в коде класса устройства. Поддерживаются все типы данных Tango, однако HDB++ не производит архивацию данных атрибутов с режимом доступа только на запись.

Значения атрибутов, вместе с временем их получения, значением качества и исключением, если оно возникло, сохраняются в базу данных. Так как объём сохраняемых данных может быть значительным, HDB++ позволяет установить для атрибутов TTL (Time To Live, время жизни) – время в часах, в течение которого значения атрибутов имеет смысл хранить. По истечении TTL неактуальные данные могут быть удалены из базы. Это может быть выполнено, например, с помощью хранимой процедуры, которая периодически вызывается средствами планирования операционной системы (cron, systemd, Windows Scheduler).

Система состоит из двух элементов: Event Subscriber и Configuration Manager. Первый отвечает за подписку на события и непосредственное сохранение данных. Для балансировки нагрузки атрибуты могут архивироваться несколькими различными Event Subscriber. Configuration Manager обеспечивает настройку и управление пулом Event Subscriber. На рис. 1 приведена схема взаимодействий в системе HDB++.

Помимо графических приложений, HDB++ предлагает библиотеки для выгрузки данных из архива для языков Java, C++ и Python. С их помощью осуществляется представление данных пользователю без необходимости самому выполнять запросы к базе данных. Для взаимодействия с базой данных архиватор использует динамически подгружаемую библиотеку, которая позволяет расширять список поддерживаемых СУБД, без необходимости изменения кода Configuration Manager и Event Subscriber.

В HDB++ до начала этой работы была реализована поддержка двух СУБД для хранения архива – Apache Cassandra и MySQL. Apache Cassandra является кластерной СУБД и имеет встроенные средства для хранения массивов. Однако, есть ряд серьезных недостатков, препятствующих использованию этой СУБД для архивирования данных ЛИУ-20. Во-первых, Apache Cassandra не поддерживает беззнаковые числа. HDB++ обходит этот недостаток, используя для беззнаковых чисел больший по размеру тип без использования его отрицательной части, но проблема с беззнаковыми 64-битными числами на данный момент не решена.

Во-вторых, система требовательна к ресурсам сервера, разработчики рекомендуют использовать компьютер с двумя процессорными ядрами и 8 ГБ оперативной памяти. Ну и наконец, в рамках использования HDB++ на ЛИУ-20, как и на многих других ускорительных комплексах преимущества кластерных баз данных не используются, в то время как настройка кластера сопряжена с большими сложностями, поэтому использование этого типа СУБД является нерациональным.

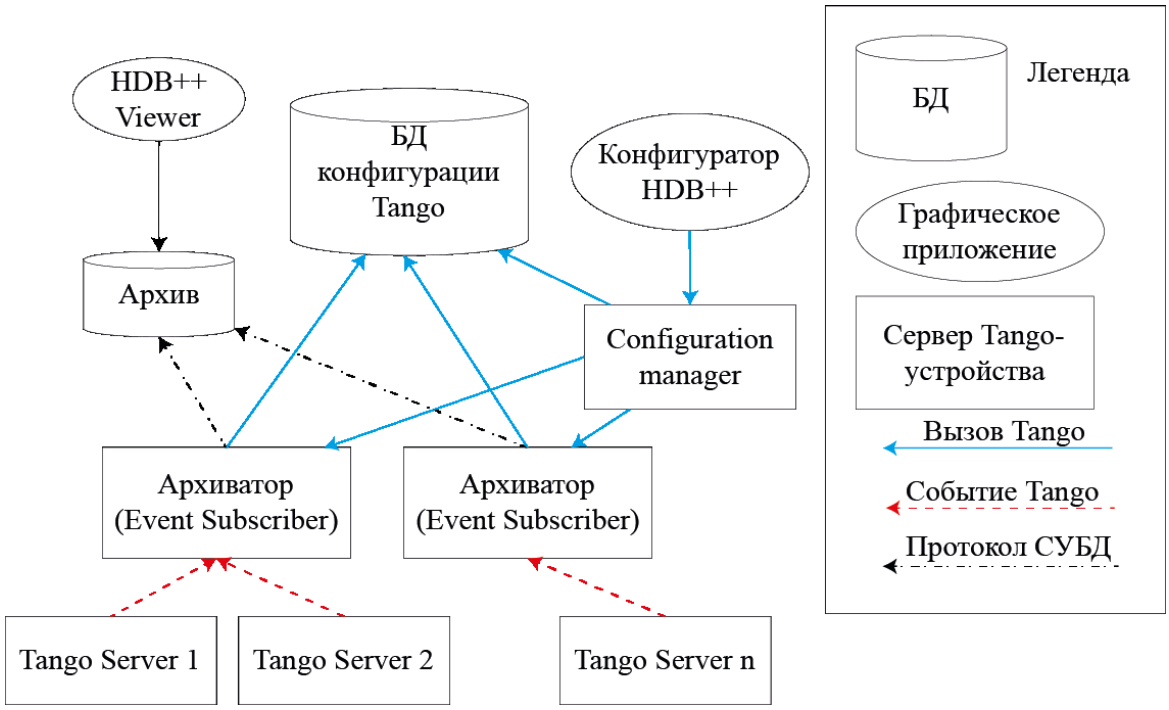

Рис. 1. Диаграмма взаимодействия HDB++
Fig. 1. HDB++ interaction diagram

MySQL поддерживает все необходимые типы и не столь требовательна к ресурсам сервера, однако имеет другие недостатки. В MySQL нет стандартных средств для хранения массивов в качестве значений. Для решения этой проблемы в HDB++ используется дополнительный столбец в таблицах, в которых записываются данные нескалярных форматов. Значение в этом столбце показывает индекс сохраняемого элемента в массиве, при этом значения остальных столбцов дублируются. Помимо идентификатора атрибута, временной метки генерации события archive event и самого значения атрибута HDB++ сохраняет временные метки генерации данных атрибута и вставки в базу данных, идентификатор исключения, если оно возникло, а также другие метаданные атрибутов. Такая избыточность при хранении массивов является серьёзным недостатком при хранении большого количества осциллограмм.

В связи с этим было решено добавить в HDB++ поддержку PostgreSQL. Эта СУБД широко применяется и на других установках в ИЯФ СО РАН. Для этого необходимо решить следующие задачи: во-первых, обеспечить конверсию типов Tango Controls и PostgreSQL, во-вторых, написать библиотеку, представляющую собой интерфейс для взаимодействия HDB++ с СУБД и её архитектуру. И наконец, необходимо заново реализовать те части HDB++, которые зависят от СУБД, то есть написать библиотеку для выгрузки данных, а также графическое приложение для их визуализации.

Библиотека MySQL-интерфейса для HDB++

Так как PostgreSQL, базируется на языке SQL, то для начала разработки собственного интерфейса логично ознакомиться со структурой и логикой уже существующего интерфейса MySQL.

Исходный код библиотеки написан, как и сам архиватор, на языке C++. Большая часть самого кода касается взаимодействия с СУБД и не представляет интереса, так как инструменты взаимодействия предназначены только для MySQL. Для оптимизации сохранения данных класс MySQL-интерфейса хранит в себе кэш, сопоставляющий с именем атрибута его идентификатор в базе данных. Также есть кэш и для описаний ошибок, но, так как последний может занимать значительный размер, при превышении размера кэша заданного значения новые значения не добавляются, а заменяют старые.

В процессе анализа исходного кода было обнаружено, что большие объекты, такие как вектора, содержащие данные нескалярных атрибутов, передаются между методами не по ссылке или указателю, а по значению. Это приводит к нежелательному копированию данных. Еще одна проблема, связанная с архивированием массивов, заключается в том, что массив сохраняется в базу одним запросом INSERT. При этом элементы массива передаются как параметры запроса, общий размер которых ограничен 64 КБ. На вставку одного элемента может приходиться от 28 до 66 байт (массивы размером 2 340 и 992 элементов) в зависимости от типа данных, режима доступа и сохраняемых временных меток. Эту проблему можно было бы решить, используя транзакции и вставляя за раз до 992 элементов.

Библиотека содержит код класса *HdbPPMySQL*, реализующий методы интерфейса *AbstractDB*. Интерфейс определяет методы для сохранения данных атрибута в архив (*insert_Attr*) и сохранения сопутствующих атрибуту параметров при старте архивации атрибута (*insert_param_Attr*), методы, которые вызываются при различных событиях: при добавлении атрибута в систему архивирования (*configure_Attr*), при изменении TTL атрибута (*updateTTL_Attr*) и при возникновении событий в HDB++, не имеющих отношения к системе событий Tango Controls, при добавлении и удалении атрибутов в системе HDB++, запуске, завершении и приостановки архивации атрибута (*event_Attr*).

Библиотека интерфейса поддержки PostgreSQL в HDB++

PostgreSQL поддерживает необходимые типы данных, кроме беззнаковых чисел. Есть несколько путей решения этой проблемы. Первый – воспользоваться большими по размеру типами чисел, как и поступили разработчики HDB++ для Apache Cassandra,. Проблемы с 64-битными числами можно решить, либо отобразив часть, не входящую в диапазон знакового 64-битного числа, на его отрицательную часть, либо воспользоваться типом DECIMAL. Оба способа имеют серьёзные недостатки. Первый лишает пользователя, решившего извлекать данные напрямую из базы данных, возможности совершать какие-либо арифметические операции над данными в рамках PostgreSQL. Тип DECIMAL реализуется с использованием длинной арифметики, поэтому обладает невысоким быстродействием. Разработчики PostgreSQL не рекомендуют использовать его для арифметических операций. Так как оба способа лишают пользователя части функционала, от них было решено отказаться.

PostgreSQL имеет систему расширений, которая в том числе позволяет определять новые типы данных. Расширение pguint для работы с беззнаковыми числами было реализовано одним из разработчиков этой СУБД Питером Эйзентраутом (Peter Eisentraut). Исходный код расширения доступен на GitHub [1].

PostgreSQL поддерживает два режима передачи данных – текстовый и бинарный. Все встроенные типы поддерживают оба режима, а для пользовательских типов обязательным

[1] https://github.com/petere/pguint

является только текстовый. Это связано с тем, что передача в бинарном режиме требует дополнительного определения сетевого формата. Однако передача в бинарном режиме является более оптимальной как в плане использования канала передачи, так и в плане быстродействия, поскольку в этом случае не производится медленных операций со строками.

В ходе этой работы была добавлена поддержка бинарного режима в pguint. Для этого был изменен вызов CREATE TYPE и указано расположение функций RECEIVE и SEND, осуществляющих конвертацию из внутреннего представления PostgreSQL в сетевое, а также реализованы эти функции для каждого из определяемых расширением типов. В качестве сетевого формата передачи было решено использовать big endian, в результате чего для знаковых и беззнаковых чисел, можно использовать тот же код, что и для встроенных типов. PostgreSQL является СУБД с открытым исходным кодом, поэтому найти соответствующий код для встроенных типов не составило проблемы.

При наличии всех необходимых типов данных можно составить окончательное сопоставление типов Tango Controls и PostgreSQL, после этого необходимо определиться со структурой базы данных. Так как проблемы MySQL-интерфейса связаны исключительно с хранением массивов, то общую архитектуру можно сохранить, убрав в таблицах с нескалярными типами столбец индекса элемента и заменив тип значения атрибута на массив соответствующего типа.

Для взаимодействия с СУБД PostgreSQL использовалась библиотека libpqtypes, которая использует библиотеку libpq и инкапсулирует логику работы с бинарными типами данных. Для того чтобы пользоваться типами pguint в рамках libpqtypes необходимо предоставить функции конвертирующие локальное представление в сетевое и обратно.

Тестирование производительности решения

После модификации расширения pguint было проведено тестирование производительности. В табл. 2 показано количество строк, вставляемых за одну секунду. В качестве тестовых данных использовались массивы по 4 096 элементов. Для измерений использовалось по 10 000 строк для каждого из типов. Тестирование проведено в режиме с транзакциями и без них. Сервер и клиент располагались на одном и том же хосте. Производительность, как и ожидалось, отличается незначительно. После выполнения проверки автору расширения был отправлен pull request на добавление нового функционала в основную ветку.

Таблица 2

Сравнение скорости работы встроенных типов и типов pguint в бинарном режиме передачи

Table 2

Performance comparison of internal and pguint types in binary mode

Режим передачи	int2	uint2	int4	uint4	int8	uint8
Без транзакций	367	371	302	293	211	211
С транзакциями	460	453	395	383	282	287

По сравнению с MySQL PostgreSQL-интерфейс обладает несколькими преимуществами. Во-первых, нет ограничений на количество элементов в нескалярных данных. Во-вторых, новый интерфейс позволяет сохранять данные массивов без ненужного дублирования данных.

Было произведено сравнение производительности и потребления памяти MySQL и PostgreSQL архивов. Для тестирования использовался атрибут *wave* стандартного тестового сервера Tango, представляющий собой одномерный массив из 256 элементов типа *DevDouble*

с доступом только на чтение, а также собственный сервер, имеющий единственный атрибут-спектр *arr* размером 4 096 элементов типа *DevULong* с доступом на чтение и на запись. Архивация первого массива производилась с периодом 100 мс, второго – 5 с. Размер выборки для первого атрибута 250 000 значений, для второго – 5 000 значений. В качестве оцениваемого параметра для производительности была выбрана разность между временными метками вставки значения в базу данных и генерации события achieve event, для потребления памяти – размер таблицы с данными, включая размер индексов. Результаты приведены в табл. 3. Все тесты проводились в одних и тех же условиях, сервера Tango-устройств были расположены на одном хосте.

Таблица 3

Результаты тестирования производительности

Table 3

Performance test results

СУБД	Атрибут	Время, мкс			Размер таблицы, МБ
		минимальное	среднее	максимальное	
MySQL	*wave*	2551	4669	253042	4439.66
PostgreSQL	*wave*	243	358	23087	767.84
MySQL	*arr*	50718	61145	198584	5511.35
PostgreSQL	*arr*	1641	2082	14173	250.94

Как видно из результатов, архивация нескалярных данных при использовании PostgreSQL значительно эффективнее, чем при использовании MySQL.

В настоящий момент HDB++ с базой данных PostgreSQL работает на ускорителе ЛИУ-20 уже в течение полутора лет. За это время проблем в работе обнаружено не было.

Библиотека для выгрузки данных для C++ и Python

Библиотека выгрузки данных необходима для инкапсуляции логики взаимодействия с базой данных, к тому же такая библиотека могла бы позволить запрашивать данные из архива по одному и тому же интерфейсу вне зависимости от используемой СУБД. К сожалению, такая возможность есть только для Java, C++ реализация предусматривает взаимодействие только с базой данных MySQL. В рамках этой работы была создана библиотека выгрузки для языков C++ и Python, так как все остальные программы ЛИУ-20 написаны на этих языках.

В минимальный набор функционала должны входить: получение количества записей для заданного атрибута в заданном промежутке времени, получение меток времени для заданного атрибута в заданном промежутке времени, получение значения либо описания ошибки по имени атрибута и метке времени. Для оптимизации выполнения запросов было решено добавить класс, представляющий атрибут и инкапсулирующий в себе идентификатор атрибута, что позволяет избежать лишнего обращения к данным архива и сравнения строк при каждой операции доступа к сохраненным значениям.

В процессе работы системы архивации на ускорителе выяснилось, что помимо доступа к значениям атрибутов по идентификатору или имени и метке времени нужно иметь возможность получить значения (либо описание ошибки, если есть) и временные метки сохранённых атрибутов в заданном промежутке времени. Для нескалярных атрибутов эти данные могут занимать большой объём, поэтому необходимо иметь возможность получить текстовое значение исключения, если оно есть, и временные метки в заданном промежутке без получе-

ния самих значений. Еще одна необходимая функция обнаружилась в процессе написания графической утилиты для визуализации данных. Для удобства выбора атрибутов нужна возможность получения списка имен архивируемых атрибутов.

Для того чтобы не писать один и тот же алгоритм на двух языках, существует инструмент для связывания библиотек написанных на C/C++ с языками Tcl, Perl, Python, Ruby, PHP, Java, C#, Scheme, OCaml – SWIG (simplified wrapper and interface generator). В случае Python SWIG использует C API, но инкапсулирует логику создания объектов внутри себя. SWIG преобразует данные типов C/C++ в объекты целевого языка. SWIG поддерживает большинство типов, определяемых стандартными библиотеками языков C и C++. Пользователь может сам написать правила для преобразования неподдерживаемых типов, используя средства конечного языка, в случае Python – Python C API. SWIG преобразует классы, поддерживает наследование (кроме множественного) и полиморфизм.

Одной из проблем было представление временных меток. В Python для этого есть класс *datetime.datetime*, в C++ было решено использовать *struct timeval*. Так как SWIG не поддерживает этот тип, правила преобразования пришлось написать самостоятельно, используя Python C API.

Наиболее удобное для дальнейшего анализа и эффективное по быстродействию и используемой памяти представление числовых массивов – это представление с использованием numpy. Эта библиотека предоставляет правила преобразования для SWIG массивов языка C в numpy-массивы. Массивы строк было решено представлять в виде вектора строк в C++ и списка строк в Python, SWIG изначально поддерживает преобразование между этими типами.

Для взаимодействия с базой данных, как и в серверной части, используются библиотеки libpq и libpqtypes. Все средства для взаимодействия с PostgreSQL архивом заключаются в классе, реализующем все вышеописанные методы. Параметры подключения к базе данных передаются как аргументы конструктора.

Утилита для выгрузки и визуализации данных

Нередко первичный анализ данных можно провести с помощью визуальной оценки данных. Для просмотра архива было реализовано графическое приложение. Его функционал включает просмотр архивируемых атрибутов, просмотр значений скалярных атрибутов в заданном интервале времени в виде графика, построение графиков для атрибутов типа спектр. Кроме того, реализовано сохранение выбранных данных в файл в форматах JSON и HDF5. Приложение реализовано на PyQt5. Для взаимодействия с базой данных архива используется описанная ранее библиотека.

Сохранение данных в формате JSON реализовано с использованием стандартного пакета json. Для формата HDF5 был использован пакет h5py. Пакет поддерживает работу с numpy-массивами, что позволяет избежать избыточных преобразований типов для больших объёмов данных. Процесс сохранения данных занимает значительное время, поэтому для отображения текущего состояния был добавлен индикатор выполнения. Внешний вид полученной программы показан на рис. 2. Исходный код разработанного программного обсечения доступен в git-репозитории Tango Controls [2].

[2] https://github.com/tango-controls-hdbpp/libhdbpp-postgresql

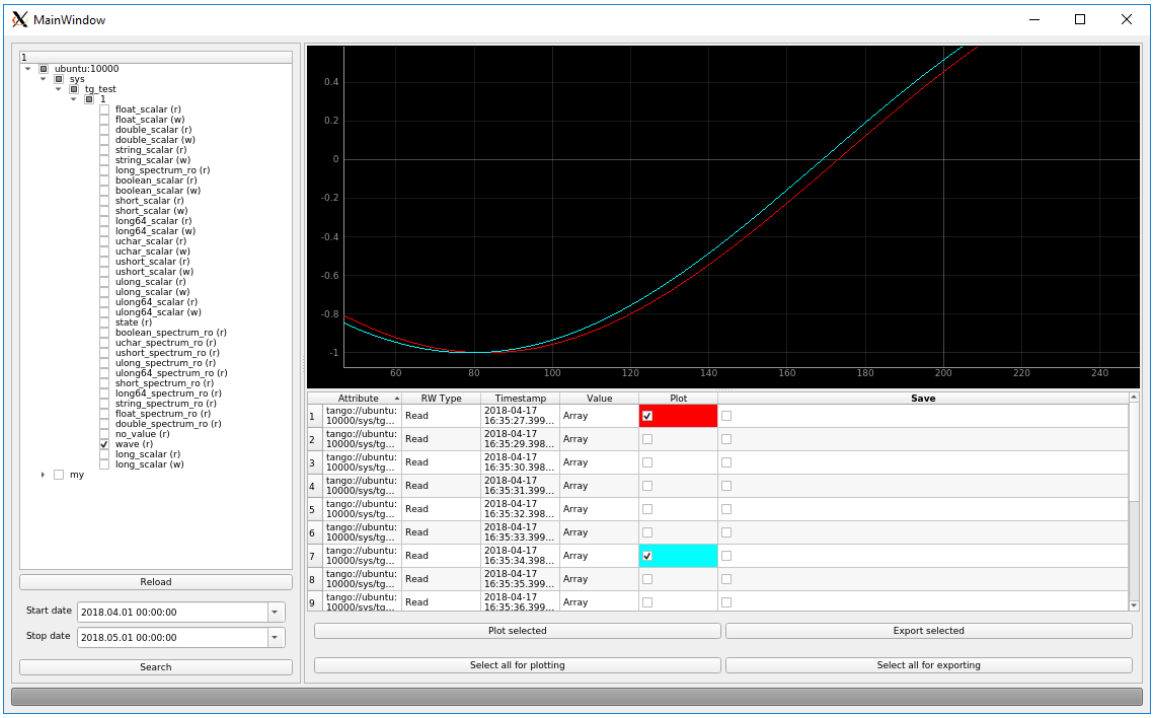

Рис. 2. Скриншот интерфейса утилиты для выгрузки и визуализации данных
Fig. 2. Screenshot of utility for exportation and visualizing data

Заключение

В связи с необходимостью архивирования осциллограмм на ЛИУ-20 было модифицировано существующие в системе управления Tango Controls решение, а именно реализовано следующее.

- Добавлена поддержка бинарного режима в расширение PostgreSQL pguint.
- Реализована библиотека для поддержки PostgreSQL в качестве СУБД для архива HDB++.
- Создана библиотека для выгрузки данных из PostgreSQL архива HDB++.
- Разработано графическое приложение для первичного анализа и выгрузки данных из PostgreSQL архива HDB++.

Итоговая система позволяет сохранять данные, необходимые в рамках работы с ЛИУ-20, в удобном неизбыточном формате. При этом производительность более чем в 10 раз выше, чем у MySQL-реализации.

Работоспособность программного обеспечения, описанного в данной работе, была протестирована на ускорителе ЛИУ-5, в данный момент оно используется на ЛИУ-20. Несмотря на то что система разрабатывалась для нужд ЛИУ-20, её можно использовать и в других проектах. Исходный код доступен в репозитории Tango Controls.

Работа продемонстрировала возможность эффективного сохранения осциллограмм в HDB++ с использованием PostgreSQL. Позже в ESRF был разработан интерфейс для TimescaleDB (является расширением PostgreSQL) с целью перехода от Apache Cassandra к более удобной архитектуре архива [3]. В качестве основы для схемы базы данных была использована схема из данной работы.

[3] *Bourtembourg R. et al.* Pushing the limits of tango archiving system using PostgreSQL and time series databases // ICALEPCS 2019. New York, October, 2019.

Список литературы

1. **Фаткин Г. А. и др.** Проект системы управления LIA-20 // Материалы 16-й Междунар. конф. ICALEPCS'17. Барселона, 2017. THPHA052. С. 1485–1488.
2. **Сенченко А. и др.** Программно-вычислительная инфраструктура системы управления Лиа-20 // 25-я Российская конференция ускорителей частиц (RuPAC'16). СПб., 2016; Женева, 2017. С. 739–741.
3. **Фатькин Г. А.** Система анализа данных рентгенографического комплекса на базе ускорителя ЛИУ-2 // Автометрия. 2015. Т. 51, № 1. С. 22–30.
4. **Фаткин Г. А.** система анализа данных радиографического комплекса на основе ускорителя Лиа-2 // Оптоэлектроника, приборостроение и обработка данных. 2015. Т. 51, № 1. С. 16–23.
5. **Chaize J.-M., Goetz A., Klotz W.-D., Meyer J., Perez M., Taurel E.** TANGO – an object oriented control system based on CORBA. In: International Conference on Accelerator and Large Experimental Physics Control Systems, 1999, p. 475–479.

References

1. **Fatkin G. A. et al.** LIA-20 Control System Project. In: Proc. 16th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'17). Barcelona, 2017, paper THPHA052, p. 1485–1488.
2. **Senchenko A. et al.** Software and Computational Infrastructure of LIA-20 Control System. In: 25th Russian Particle Accelerator Conf. (RuPAC'16). St. Petersburg, 2016; JACOW, Geneva, 2017, p. 739–741.
3. **Fatkin G. A.** Data Analysis system of the x-ray complex based on the LIU-2 accelerator. *Autometrics*, 2015, vol. 51, no. 1, p. 22–304.
4. **Fatkin G. A.** Data analysis system of a radiographic complex based on a LIA-2 accelerator. *Optoelectronics, Instrumentation and Data Processing*, 2015, vol. 51, no. 1, p. 16–23.
5. **Chaize J.-M., Goetz A., Klotz W.-D., Meyer J., Perez M., Taurel E.** TANGO – an object oriented control system based on CORBA. In: International Conference on Accelerator and Large Experimental Physics Control Systems, 1999, p. 475–479.

Материал поступил в редколлегию
Received
08.05.2020

Сведения об авторах

Ситнов Владимир Евгеньевич, магистрант факультета информационных технологий, Новосибирский государственный университет (Новосибирск, Россия); старший лаборант, Институт ядерной физики им. Г. И. Будкера СО РАН (Новосибирск, Россия)

vladimir@sitnov.org

Сенченко Александр Игоревич, научный сотрудник, Институт ядерной физики им. Г. И. Будкера СО РАН (Новосибирск, Россия); ассистент, Новосибирский государственный университет (Новосибирск, Россия)

senchenko.alexander@gmail.com

Фаткин Георгий Александрович, старший научный сотрудник, Институт ядерной физики им. Г. И. Будкера СО РАН (Новосибирск, Россия); доцент кафедры радиофизики и физико-технической информатики физического факультета НГУ, Новосибирский государственный университет (Новосибирск, Россия)

george.fatkin@gmail.com
ORCID 0000-0002-8426-0015

Information about the Authors

Vladimir E. Sitnov, Master's Student, Novosibirsk State University (Novosibirsk, Russian Federation); senior assistant, Budker Institute of Nuclear Physics SB RAS (Novosibirsk, Russian Federation)

vladimir@sitnov.org

Alexander I. Senchenko, researcher, Budker Institute of Nuclear Physics SB RAS (Novosibirsk, Russian Federation); assistant, Novosibirsk State University (Novosibirsk, Russian Federation)

senchenko.alexander@gmail.com

George A. Fatkin, senior researcher, Budker Institute of Nuclear Physics SB RAS (Novosibirsk, Russian Federation); associate professor, department of radiophysics and physical-technical informatics, faculty of physics, Novosibirsk State University (Novosibirsk, Russian Federation)

george.fatkin@gmail.com
ORCID 0000-0002-8426-0015