МОДЕРНИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ МЕДЛЕННОГО КОНТРОЛЯ ДЕТЕКТОРА ATLAS

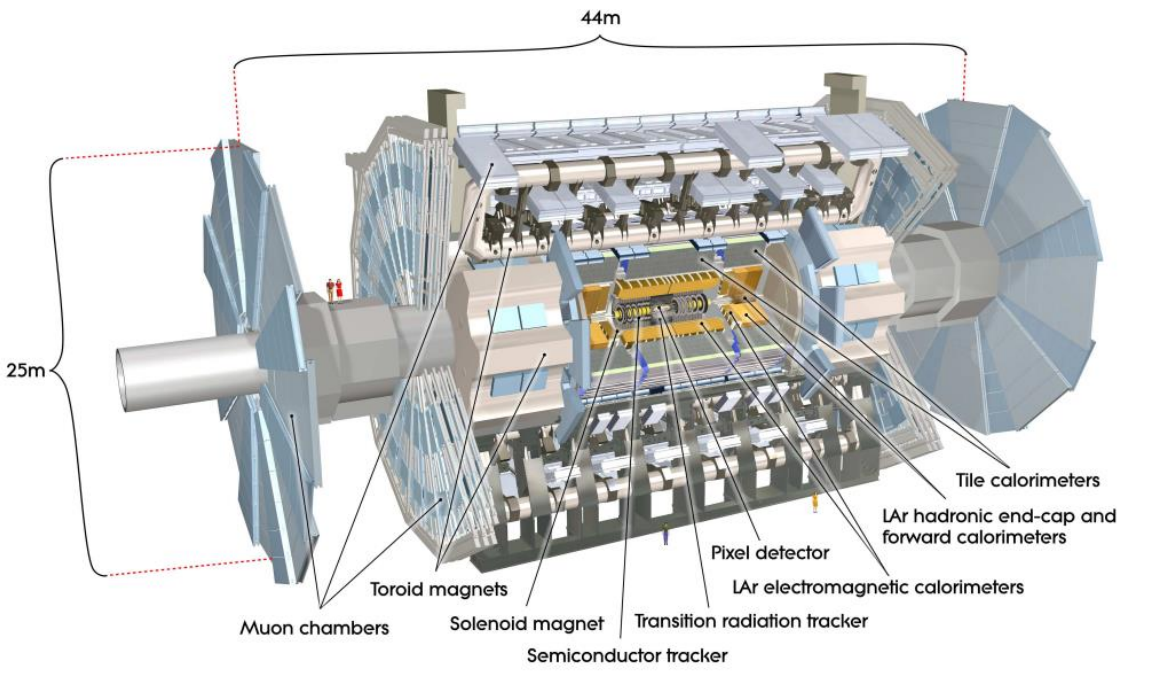
Доклад Кочергина И.А.
аспиранта МГУ им.М.В.Ломоносова
Научный руководитель: Смирнова Л.Н.

Черенковские чтения. Апрель 2023. Москва

План доклада

1. Детектор ALTAS
2. DCS – системы медленного контроля детектора
3. Программное обеспечение систем контроля
4. Модернизация программного обеспечения

1. Детектор ALTAS.



25 м в высоту и 44 м в длину. Общий вес около 7000 тонн.

12 отдельных поддетекторов

Радиация внутри экспериментальной каверны за пределами внутреннего детектора в диапазоне от 0,4 Гр в год в адроном калориметре и до 900 Гр в год в прямом направлении

Магнитные поля, превышающие 1 Тл в тороидальных полях ATLAS

Всего в детекторе порядка 100 000 000 каналов считывания

более 10 000 000 элементов данных

Эксперимент ATLAS управляется двумя сотрудничающими системами: DCS и системой запуска и сбора данных (Trigger and Data-Acquisition – TDAQ)

1. Детектор ALTAS. Подсистемы

Inner detector

- precision tracking detectors:
  - *Pixels*
  - *SCT – silicon microstrip*
- *Transition Radiation Tracker (TRT)*

Calorimeter system

- *LAr electromagnetic calorimeter*
- Hadronic calorimeters
  - *Tile calorimeter*
  - *LAr hadronic end-cap calorimeter*
  - *LAr forward calorimeter*

Muon system

- *Monitored drift tubes MDT*
- *Cathode strip chambers CSC*
- *Resistive plate chambers RPC*
- *Thin gap chambers TGC*

Common Infrastructure Control (CIC)

2. DCS. Система контроля детектора

Система управления детектором ATLAS была разработана и внедрена в рамках проекта Joint Controls Project (JCOP), сотрудничества группы управления CERN и групп DCS экспериментов LHC. Стандарты для аппаратного и программного обеспечения DCS были установлены вместе с политиками реализации как обычно для JCOP, так и специально для ATLAS. Сюда входят полевые шины, протоколы и общая SCADA, PVSS II. Целью общего подхода является сокращение рабочей силы, необходимой для разработки и обслуживания.

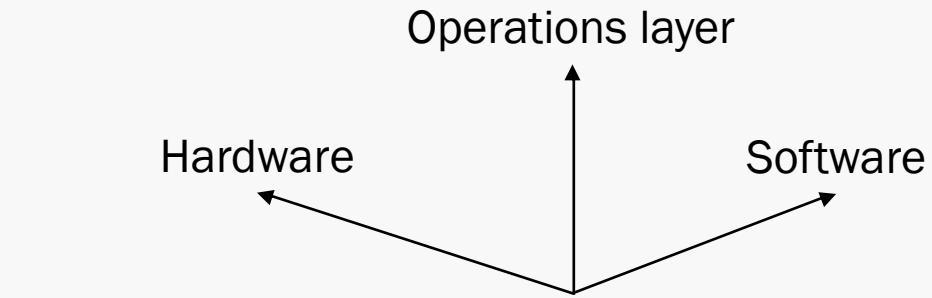
supervisory control and data acquisition software package (SCADA) – пакет программного обеспечения для диспетчерского управления и сбора данных

Prozess Visualisierungs - and Steuerungs System (PVSS) – Система визуализации и контроля процесса

Задачей системы контроля детектора является согласованное и безопасное управление и мониторинг ATLAS, его поддетекторов и технической инфраструктуры. DCS должна переводить детектор в любое требуемое рабочее состояние, постоянно контролировать и архивировать рабочие параметры, сигнализировать о любом ненормальном поведении. Кроме того, ATLAS DCS должна служить единым интерфейсом для всех поддетекторов и технической инфраструктуры эксперимента. Наконец, DCS должна поддерживать связь с другими системами, которые управляются независимо, такими как ускоритель LHC, технические службы CERN, магниты ATLAS и система безопасности детекторов (DSS).

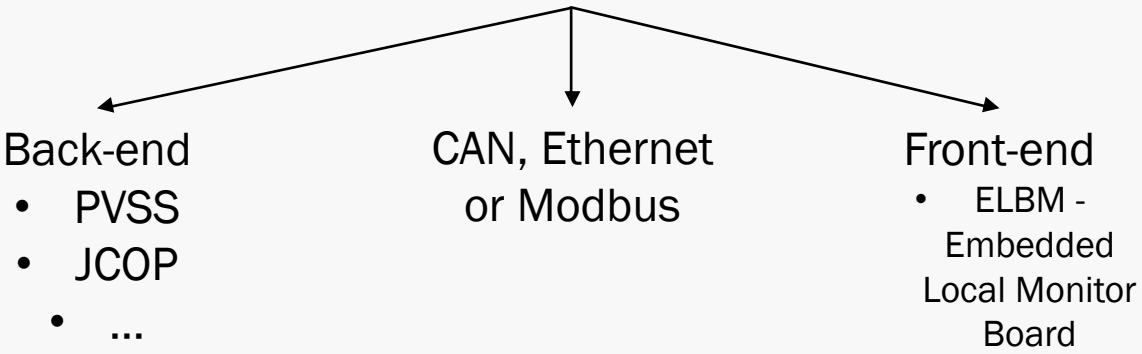
2. DCS. Архитектура

- Front-end (FE): включает аппаратное обеспечение DCS, такое как источники питания, датчики окружающей среды или контуры охлаждения.
- Back-end (BE): серверная часть обозначает программное обеспечение, используемое для интеграции внешних элементов управления, продукт промышленного диспетчерского управления и сбора данных (SCADA). PVSS служит в качестве базового программного обеспечения, а программные компоненты структуры JCOP облегчают интеграцию стандартных аппаратных устройств и реализацию однородных приложений управления.
- *Embedded Local Monitor Board (ELMB)*

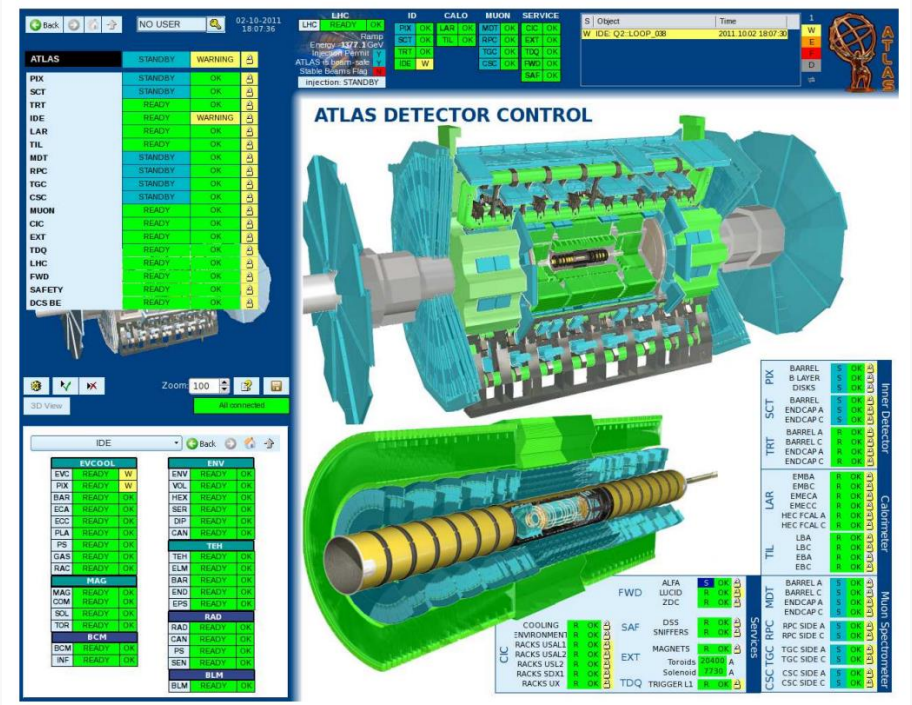


■ DCS – Detector Control System

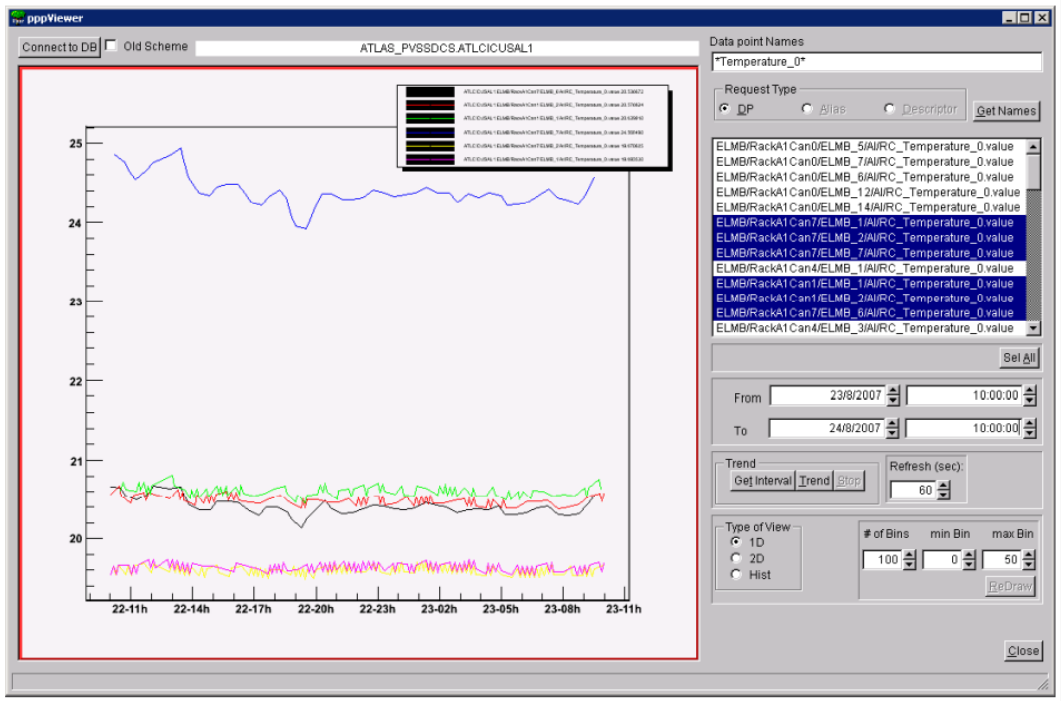
(https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AtlasDcs)



2. DCS. Operations layer.

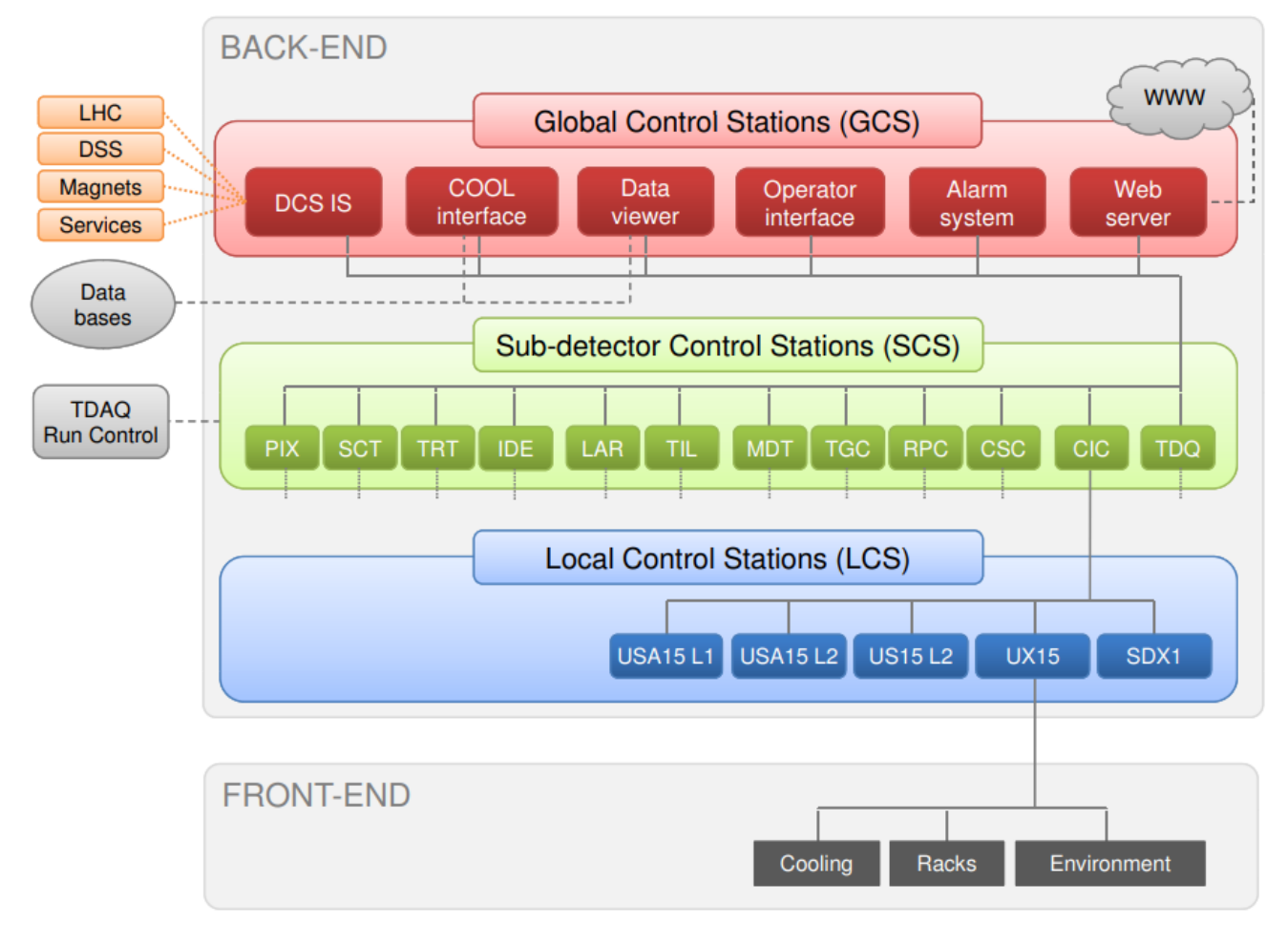


Интерфейс оператора (экран FSM), показывающий детектор в конфигурации STANDBY во время разгона LHC



Средство просмотра Root, показывающее несколько значений параметров с течением времени

2. DCS. Hardware and software components



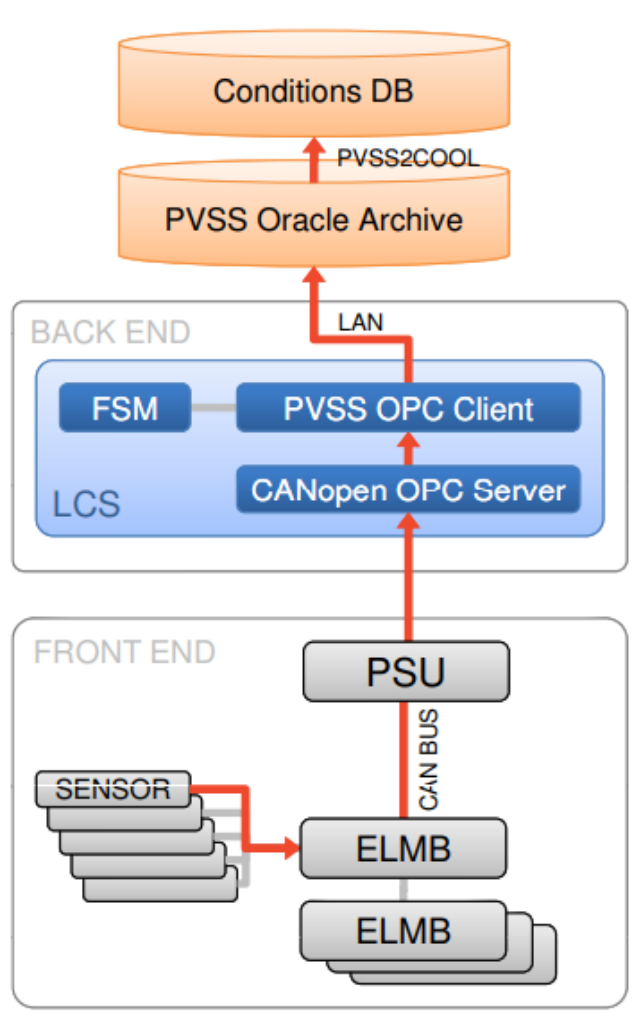
Back-end состоит из 3 уровней: локальные станции управления (LCS) для управления технологическим процессом подсистем, станции управления субдетекторами (SCS) для высокоуровневого управления субдетектором, обеспечивающие автономную работу, и Глобальные станции управления (GCS) с интерфейсом пользователя в диспетчерской ATLAS для общей работы. Важные требования к DCS FE I/O:

- низкая стоимость (т.е. использование коммерческих компонентов)
- низкое энергопотребление,
- Высокая входная-выходная (I/O) плотность каналов

Если электроника FE I/O расположена в камере детектора, необходимо выполнить дополнительные требования:

- возможность удаленного обновления прошивки
- нечувствительность к магнитным полям
- устойчивость к уровням радиации, присутствующим в этом месте, интегрированным в течение всего срока эксперимента.

2. DCS. ELMB

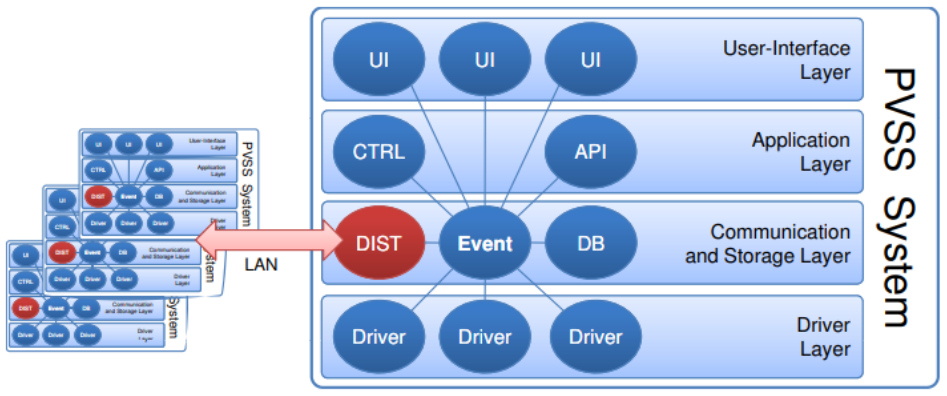


Embedded Local Monitor Board (ELMB)

ELMB был разработан как обычное недорогое устройство ввода/вывода. Он обеспечивает 64 аналоговых и 24 цифровых канала размером 50x67 мм^2, 8-битный микроконтроллер 4 МГц и интерфейс шины CAN. Он устойчив к сильным магнитным полям и жесткому радиационному излучению при суммарных дозах до 50 Гр. Кроме того, ELMB может быть встроен в пользовательские конструкции и имеет модульную, дистанционно расширяемую прошивку с приложением ввода-вывода общего назначения CANopen. Более 10000 ELMB используются во всех экспериментах LHC, более 5000 только в ATLAS.

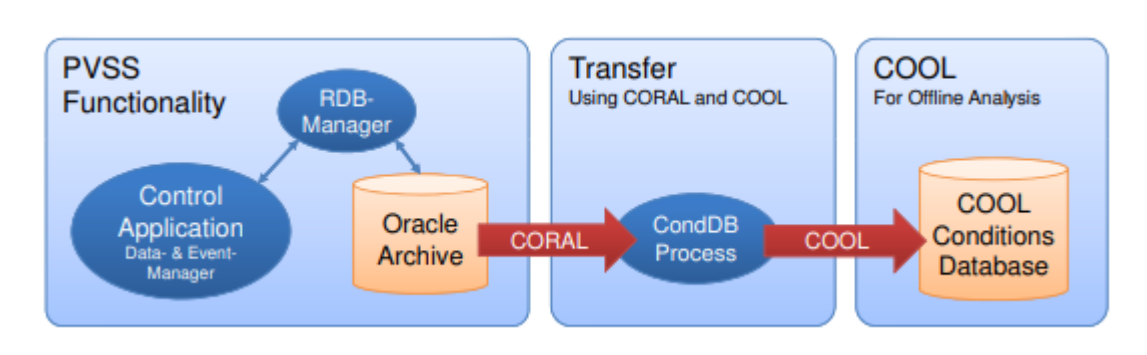
На рисунке показан типичный пример цепочки считывания на основе ELMB. ELMB размещаются в пещере ATLAS и собирают данные с датчиков, распределенных по всему объему детектора. Любой узел ELMB можно настроить для передачи калиброванных данных датчика либо через регулярные промежутки времени, либо при изменении. В последнем режиме, который сравнивает показания с предварительно заданными пороговыми значениями, достигается первый уровень сокращения данных. Все ELMB связаны с соответствующей LCS в подземных счетных комнатах через CAN. Эти шины CAN работают со скоростью 125 кбод, что обеспечивает максимальную длину шины около 500 м. В общей сложности 63 ELMB могут быть последовательно подключены к одной шине CAN и получать питание по шинному кабелю от специальных источников питания. Эти источники питания расположены в комнате подсчета, что позволяет в случае возникновения ошибок выключить и снова включить все узлы в шине. Они также контролируют потребление тока шинами, чтобы обнаружить эффекты старения ELMB из-за излучения.

3. ПО – программное обеспечение. PVSS. Медленный контроль



- Prozess Visualisierungs - and Steuerungs System (PVSS) - Process visualization and control system
- Станция управления (ПК) запускает «Project», который содержит ряд процессов «Managers». Их использование и тип зависит от типа приложения.
- Каждый проект PVSS использует центральную базу данных для всех текущих значений данных, хранящихся в объектах, называемых «Datapoints» (точки данных) которые содержат данные в реальном времени от оборудования FE, сгруппированные в структуру для данного устройства. Все менеджеры имеют полный доступ к базе данных, для которой PVSS обеспечивает прозрачную синхронизацию. Обработка данных выполняется в событийно-ориентированном подходе с использованием многопоточных процедур обратного вызова при изменении значения.
- Различные проекты могут быть связаны через локальную сеть для формирования «распределенной системы Distributed System», позволяющей удаленно получать доступ к базам данных и событиям всех подключенных проектов. Это обеспечивает масштабируемость до полного размера DCS ATLAS с более чем 150 различными станциями управления.
- Общий интерфейс прикладного программирования (API) позволяет расширить функциональность управляющих приложений с помощью дополнительных программных компонентов.

3. ПО. Базы данных



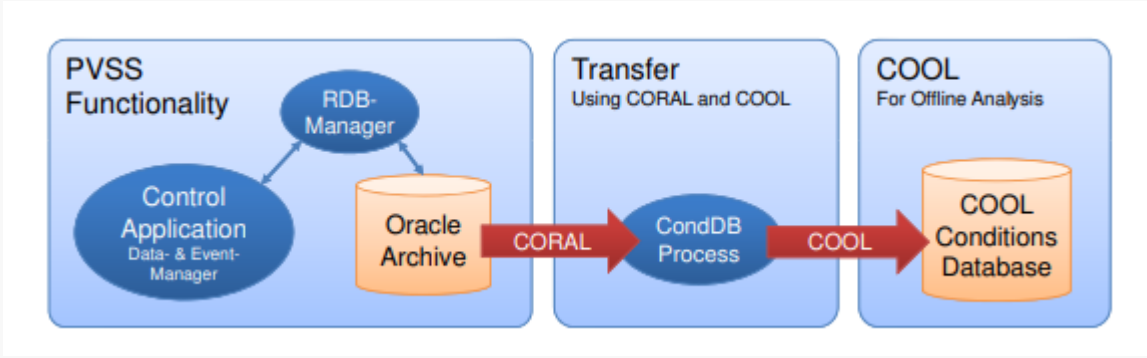
Данные об условиях детектора ATLAS хранятся в реляционной базе данных (CondDB) для эффективного доступа любыми автономными приложениями калибровки, реконструкции и анализа. Доступ осуществляется через специальный API под названием COOL, он оптимизирован только для автономных приложений, вместо того, чтобы использоваться для диагностики проблем детектора.

База данных конфигурации ATLAS DCS (ConfDB) предназначена для управления настройкой параметров детектора и DCS (таких как константы калибровки, настройки напряжения и пороги срабатывания сигнализации) в зависимости от режима работы эксперимента. Кроме того, в ней хранится конфигурация оборудования DCS. ConfDB представляет собой набор баз данных Oracle (по одной на поддетектор), доступных для всех приложений DCS. Доступ основан на компоненте Configuration DB FW, которая определяет модель данных, состоящую из двух основных объектов.

- «Configuration» содержит наборы устройств с их статическими свойствами, например, аппаратными адресами, настройками архивирования и т. д.
- «Recipe» представляет собой набор значений, зависящих от времени выполнения, таких как установки для выходных каналов и пороги оповещения, которые могут меняться в зависимости от режима работы детектора.

Чтобы обеспечить прямой и легкий доступ к архиву PVSS Oracle для участников совместной работы, было разработано специальное веб-средство просмотра данных (DCS Data Viewer: DDV).

3. ПО. COOL



Объем данных, хранящихся в CondDB, должен быть сведен к минимуму по двум следующим основным причинам:

- данные об условиях должны быть реплицированы на внешние сайты для анализа и реконструкции по всему миру
- для анализа требуется обширная обработка данных, поэтому поиск нужной информации в больших объемах данных об условиях является ненужным накладным расходом

COOL хранит логически сгруппированные данные в «folders» (папках). Программный COOL API состоит из библиотеки C++, которая опирается на низкоуровневую библиотеку доступа к БД под названием CORAL (разработанной CERN IT). Как и PVSS имеет концепцию точек данных.

Специальный процесс — PVSS2COOL — считывает выбранные данные PVSS из базы данных архива PVSS с помощью универсального API доступа (CORAL), отображает данные из точек данных PVSS в папки COOL соответствующей структуры и записывает их в базу данных CondDB.

Типы точек данных, представляющие устройства, связаны с соответствующими папками, что позволяет хранить в папке данные с одного или нескольких устройств каждого типа. Конфигурация папок COOL и данных, которые они должны содержать, определяются в PVSS, что позволяет специалистам DCS поддетекторов структурировать данные для автономного анализа.

4. Модернизация программного обеспечения систем контроля

Основные мотивы замены существующего API следующие:

- *Управление тегами и глобальными тегами*: некоторые функции, интенсивно используемые при управлении данными условий, не были разработаны в COOL API и добавляются поверх него с помощью специальных инструментов Python. Это увеличивает количество программных библиотек и затрудняет понимание использования COOL
- *Кэширование*: загрузка данных условий в стандартной обработке данных требует особого внимания к возможности кэширования запросов. В COOL API данные полезной нагрузки загружаются одновременно с IOV, что усложняет процесс кэширования. Часть кэширования была интегрирована в уровень IOVDbSvc, добавляя этому уровню дополнительную сложность, поскольку он содержит настраиваемые способы сделать запросы максимально воспроизводимыми для событий, близких по времени.
- *Структура базы данных*: возможность создавать таблицы непосредственно из COOL API оказывает большое влияние на количество таблиц, которые хранятся в схемах Oracle. Сегодня данные условий ATLAS распределены по примерно 30 схемам и суммируют около 10 000 таблиц (всего около 1 ТБ за период сбора данных). Каждый раз, когда системе приходится «менять» свой формат, создается новый набор таблиц.
- *Долгосрочное обслуживание и развитие*: COOL API (как и CORAL) не поддерживается IT-отделом после запуска Run3. Эволюция COOL API — сложная задача, требующая высокого уровня знаний.
- *Сохранение данных*: использование файлов sql-lite для сохранения данных может стать сложным, если мы примем во внимание увеличение числа таблиц и схем, предусмотренных также с учетом установки новых детекторов.

4. Модернизация ПО. CREST

Conditions data with REST interface, REST – **Representational State Transfer** — это архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Архитектурный стиль – это набор согласованных ограничений и принципов проектирования, позволяющий добиться определённых свойств системы.

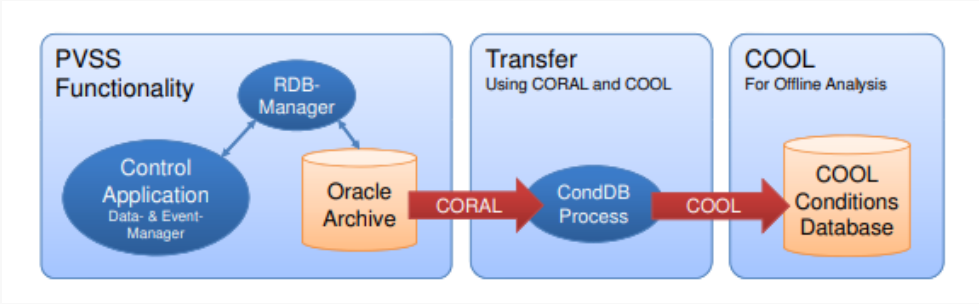
6 принципов REST:

- Клиент-серверная архитектура
- Stateless: сервер не должен хранить у себя информацию о сессии с клиентом. Он должен в каждом запросе получать всю информацию для обработки
- Кэширование
- Единообразие интерфейса
- Layered system (слоистая архитектура)
- Code on demand (код по требованию): Идея передачи некоторого исполняемого кода (по сути какой-то программы) от сервера клиенту.

Используется HTTP RESTfull API

Новый автономный механизм геометрии в настоящее время может создавать постоянную копию описания детектора в двух форматах: SQLite и JSON. Для этого прототипа выбрана версия JSON, и файл представлен в базе данных условий как тип BLOB-объекта, связанный с одним, так называемым, интервалом достоверности (IOV, который определяет время, в течение которого данные объявляются «действительными» и который простирается от 0 до бесконечности) и к одному тегу, который может соответствовать предыдущему упомянутому тегу геометрии. Для тестов использовался прототип базы данных CREST, развернутый на Openshift в ЦЕРНе, а также клиентские библиотеки Python для взаимодействия с сервером CREST через REST.

4. Модернизация ПО. Из PVSS2COOL в PVSS2CREST



Процесс PVSS2COOL берет выбранные данные из архива WinCC Oracle и передает их в базу данных COOL, в соответствии с предварительно созданным определением папки. Типы точек данных (которые представляют типы устройств) связаны с папкой, что позволяет хранить в папке данные с одного или нескольких устройств каждого типа. Приложение PVSS2COOL периодически запускается как центральная служба по сеансам для каждого детектора. Интервал между сеансами детектора в настоящее время выбран 15 мин. Это происходит без явного участия поддетекторов, которые все еще могут приостановить/восстановить процесс для любой определенной папки.

Модель данных в CREST состоит из пяти таблиц, содержащих метаданные и данные полезной нагрузки, первоначально вдохновлена базой данных условий CMS. Данные условий хранятся в таблице PAYLOAD. Значения используются в виде агрегированного набора, метаданные условий организованы в три таблицы. IOV в CREST содержит информацию о времени, которая хранится в одном столбце и по умолчанию действует до следующего ввода времени. TAG в CREST — это метка, используемая для идентификации определенного набора IOV. GLOBAL TAG — это метка, используемая для идентификации согласованного набора TAG, участвующих в данном потоке данных, один и тот же TAG может быть связан со многими GLOBAL TAG

IOV извлекаются отдельно от PAYLOADS. Это другое поведение по отношению к COOL, который загружает все (IOV+PAYLOAD) одновременно. Это дает несколько преимуществ: во-первых, доступ к ПОЛЕЗНОЙ НАГРУЗКЕ можно кэшировать, чтобы он стал быстрее, но также можно очень быстро проверить IOV, поскольку они де-факто являются метаданными ПОЛЕЗНОЙ НАГРУЗКИ.

Заключение

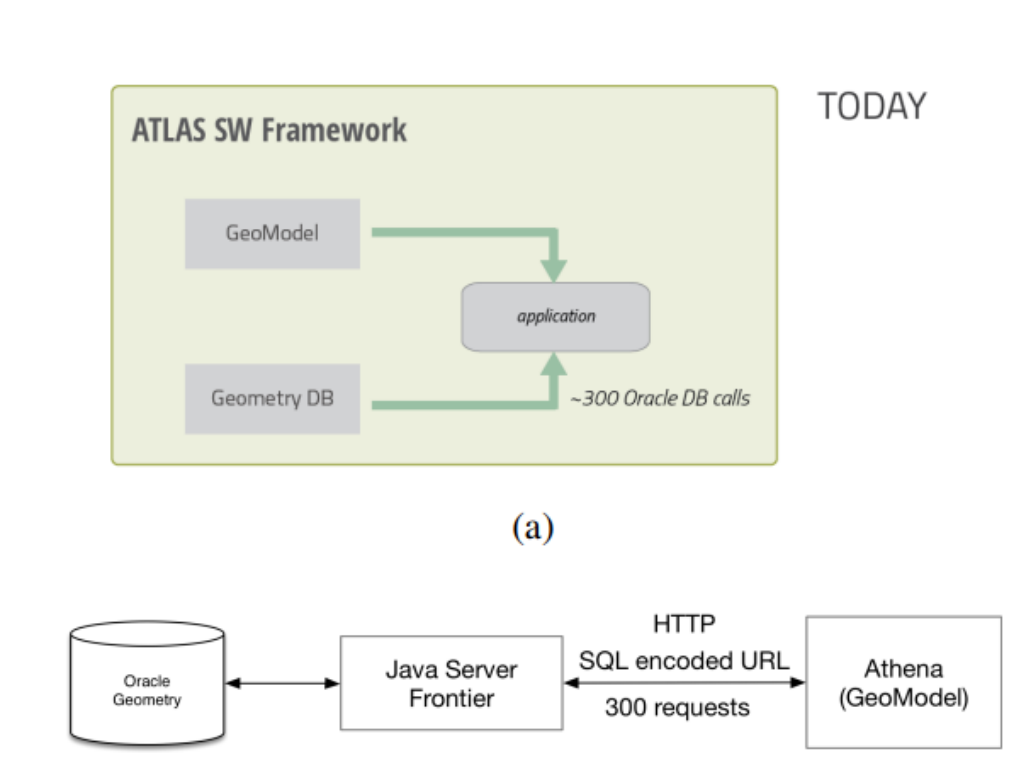
- Существующая организация системы контроля детектора обеспечила обработку, передачу и хранение информации о подсистемах детектора на этапах Run1 и Run2.
- В связи с развитием технологий, увеличением количества информации и ростом функционала путем интегрирования дополнительных программных и электронных модулей снижается эффективность работы системы контроля детектора и, в частности, работы существующих API
- В настоящее время идет глобальная модернизация программного обеспечения, перенос процессов для использования CREST в виде HTTP RESTfuII API
- В рамках апгрейда программного обеспечения производится модернизации инструмента PVSS2COOL в PVSS2CREST, в процессе согласование особенностей параметров с ответственными в ATLAS

Спасибо за внимание!

Литература:

- The ATLAS Experiment at the CERN Large Hadron Collider, 2008 IOP Publishing Ltd and SISSA
- The ATLAS Detector Control System, Conference Series 396 (2012) 012028
- The detector control system of the ATLAS experiment, A Barriuso Poy et al 2008 JINST 3 P05006
- A new mechanism to use the Conditions Database REST API to serve the ATLAS detector description, ATL-SOFT-PROC-2018-056
- THE DEVELOPMENT OF A NEW CONDITIONS DATABASE PROTOTYPE FOR ATLAS RUN3 WITHIN THE CREST PROJECT, E.Alexandrov1 , A.Formica2 , M.Mineev1,a , S.Roe3 on behalf of the Software and Computing Activity
- CREST Documentation, https://twiki.cern.ch/twiki/pub/AtlasComputing/ConditionsRest/Crest_Project_Description_20171129.pdf [accessed 2018-12-03]

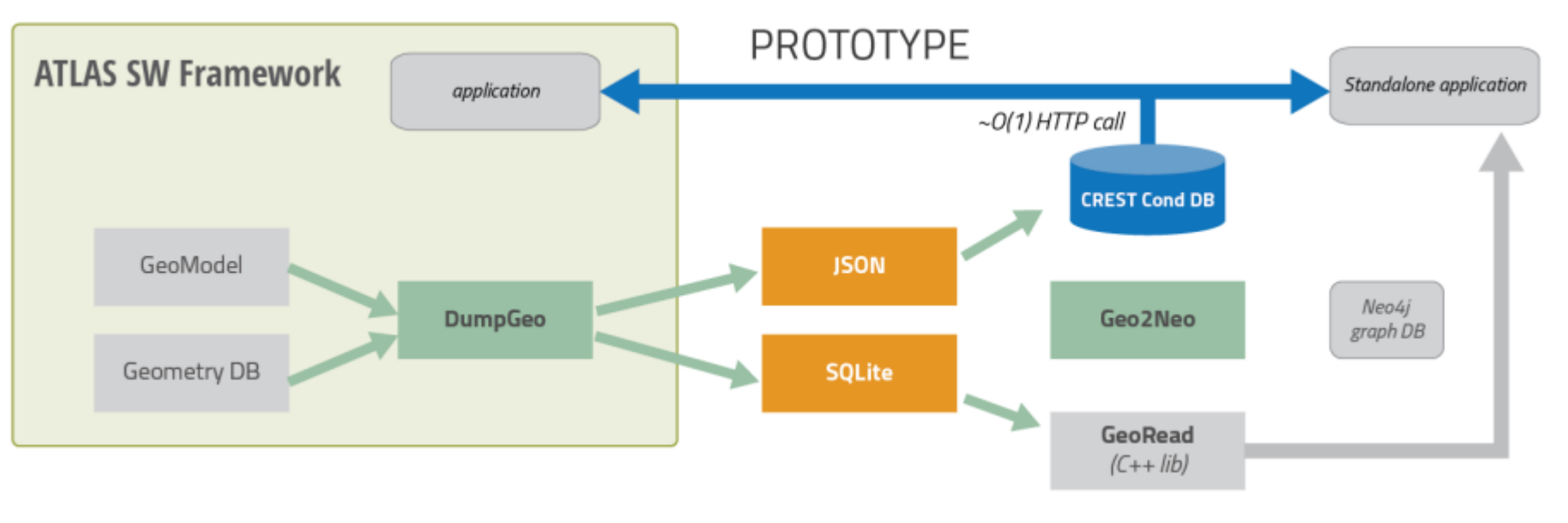
Старая архитектура.



a) the architecture currently implemented in ATLAS: the detector description is built on-the-fly, taking the definition of the GeoModel tree from C++ code and the geometry parameters from a dedicated Oracle database though a dedicated service; all operations are performed within the experiment's framework

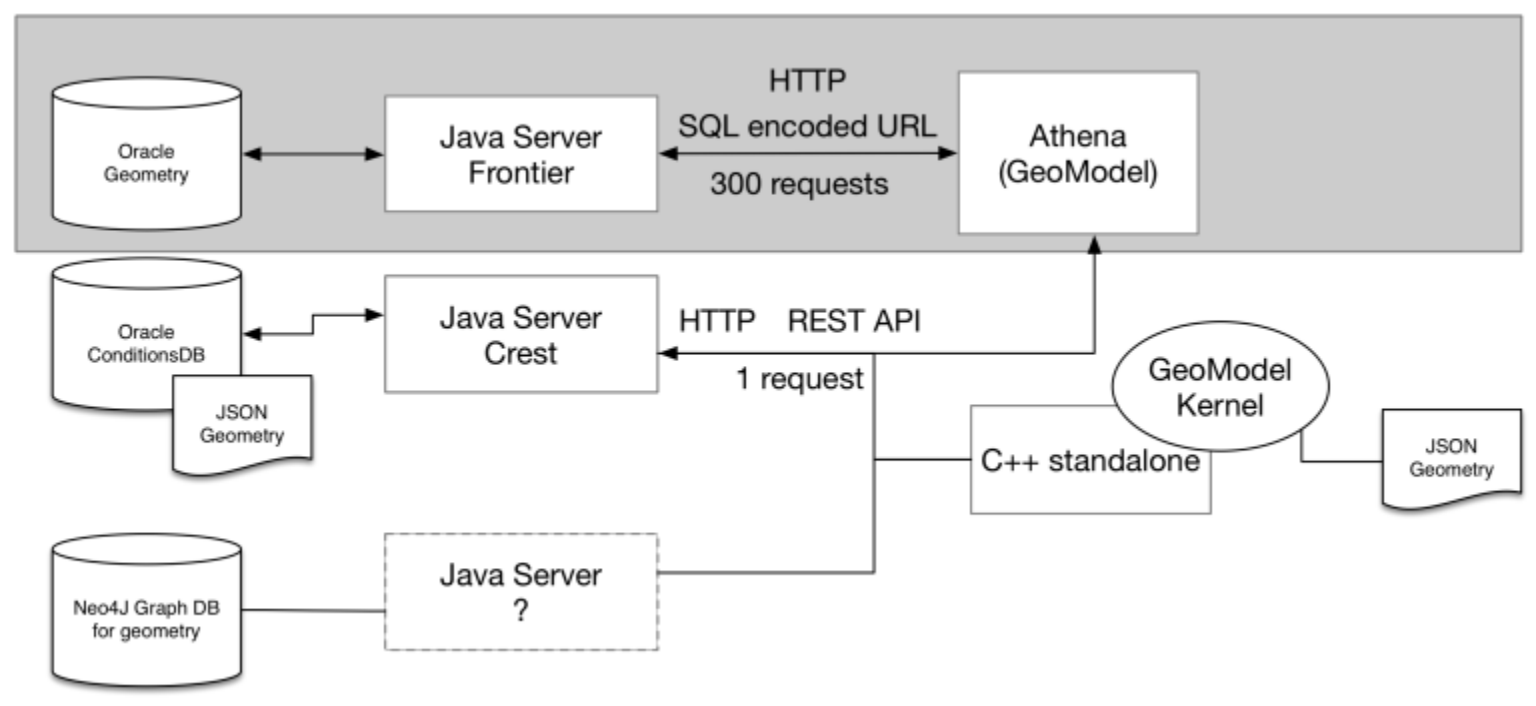
b) The current actors and actions involved in the retrieval of geometry parameters from the Oracle DB. Today, about 300 SQL queries are needed to retrieve the full set of geometry parameters, performed by each of the jobs processing the data.

Новая архитектура



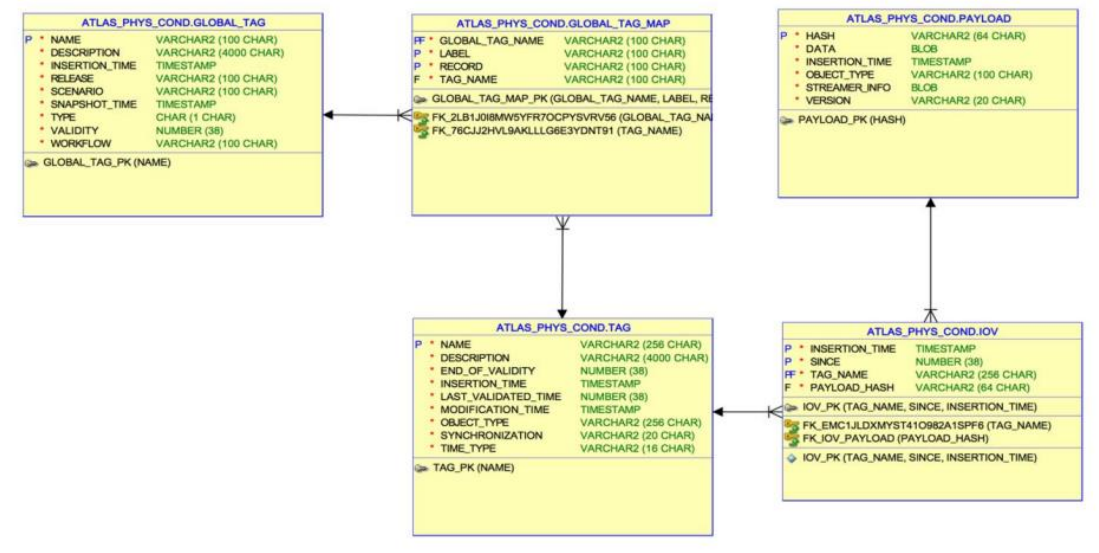
Новая архитектура. После создания описания детектора его постоянная копия сохраняется в файлах, из которых ее можно читать, обслуживать и запрашивать вне рамок эксперимента.

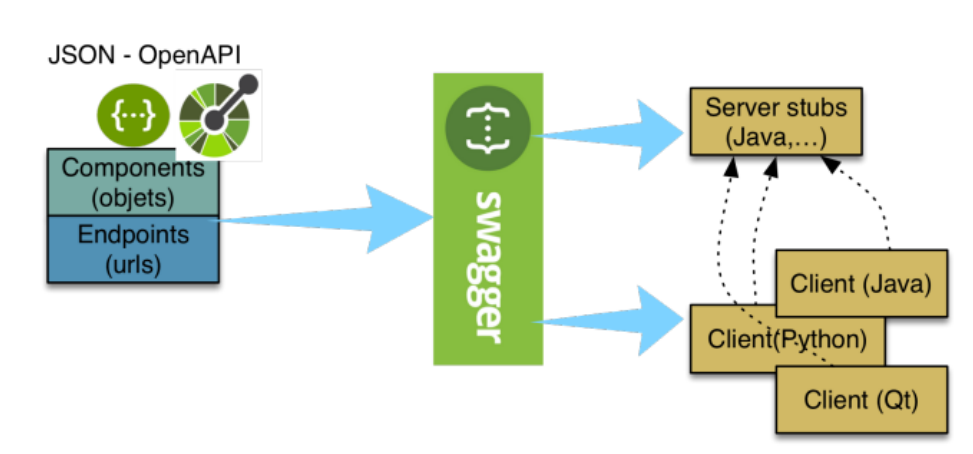
Интеграция



Обзор новой архитектуры, интегрированной с текущей. Серым цветом обозначена текущая система для извлечения данных геометрии из базы данных GeometryDB на базе Oracle через структуру эксперимента. В середине новая архитектура, которая использует HTTP REST API для извлечения данных геометрии из базы данных условий CREST, где они хранятся в виде больших двоичных объектов JSON. Тот же REST API может использоваться и автономными приложениями (справа на рисунке). Внизу предусмотрено будущее расширение архитектуры, запрашивающее и фильтрующее данные геометрии из экземпляра графовой базы данных Neo4j.

CREST. SWAGGER





- Данные условий: они хранятся в таблице PAYLOAD. Значения используются как агрегированный набор (обычно заголовок и некоторые контейнеры параметров).
- Метаданные условий: они организованы в 3 таблицы (плюс одна используется в основном для сопоставления между тегами и глобальными тегами).
  - IOV: содержит информацию о времени, которая хранится в 1 столбце времени (время может быть представлено в виде метки времени, номера запуска и т. д.), и по умолчанию она действительна до следующего ввода времени. IOV указывает на 1 полезную нагрузку через хеш-ключ sha256.
  - TAG: это метка, используемая для идентификации определенного набора IOV.
  - GLOBAL_TAG: это метка, используемая для идентификации согласованного набора тегов, задействованных в данном потоке данных. TAG может быть связан со многими GLOBAL_TAG.

В прототипе для создания серверного и клиентского кода использовался Swagger OpenAPI.