
# Введение в ССД
# Онлайн мониторинг

О.Соловьянов (на основе материалов S. Kolos PNPI / CERN)

# Содержание

- Что такое онлайн мониторинг?
- Основы архитектуры системы мониторирования
- Что будет мониторироваться?
- Масштабирование системы мониторирования для больших экспериментов
- Технологии реализации системы онлайн мониторирования

# Для чего нужен мониторинг?

- Хорошо спроектированная система мониторирования должна уметь отвечать на следующие вопросы:
  - Что произошло?
  - Где произошло?
  - Когда произошло?
- На некоторые вопросы она не в состоянии дать ответ:
  - Что делать?
- Но она должна дать как можно больше информации для принятия решения

"I swear to tell the truth, the whole truth, and nothing but the truth, from my perspective."

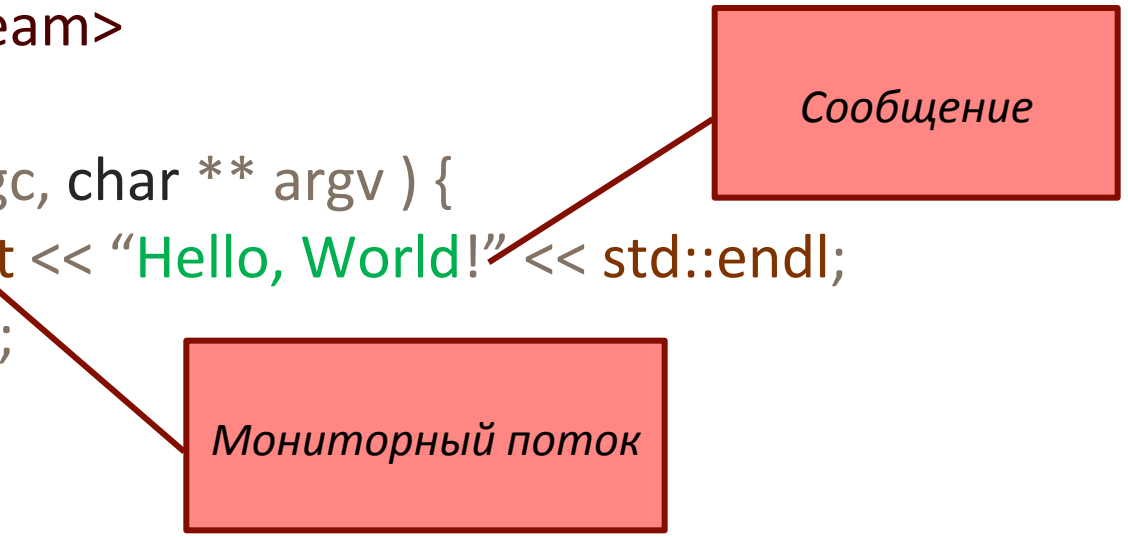
# Как устроена система мониторирования?

- Все мы уже обладаем неплохой системой мониторирования:
  - 5 типов сенсоров (чувств)
  - Информация передаётся в мозг по нервным волокнам
  - Важная информация записывается в память для дальнейшего использования
- Мониторинг ССД работает примерно таким же образом:
  - Электронные и программные компоненты играют роль сенсоров и сообщают информацию системе
  - Система мониторирования передаёт эту информацию в «мозг»
    - Оператор
    - Экспертная система
  - Записывает часть информации для последующей обработки и анализа

# Пример программы мониторирования

```
#include <iostream>

int main( int argc, char ** argv ) {
        std::cout << “Hello, World!” << std::endl;
        return 0;
}
```



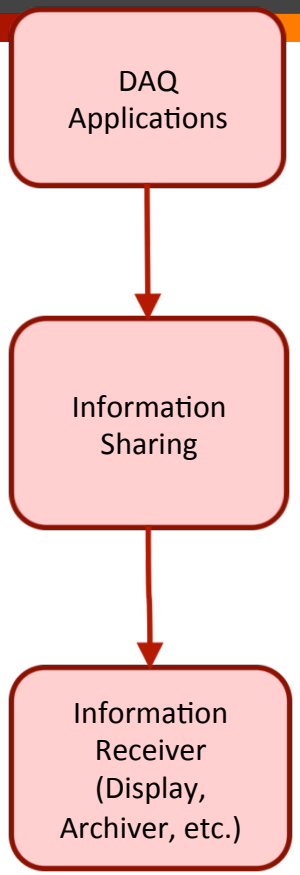
# Упрощённая архитектура системы мониторирования

- Мониторный поставщик (monitoring provider):
  - Программное приложение, отражающее своё состояние или состояние оборудования, путём посылки сообщений в мониторный поток
  - Мониторный получатель (monitoring receiver):
  - Программное приложение, получающее мониторную информацию для определённой цели, например для сообщения оператору
- Мониторный поток (monitoring stream):
  - Обеспечивает передачу информации от поставщика получателю

# Преимущества модульной архитектуры

- В подобной архитектуре все компоненты независимы:
  - Поставщики не зависят от получателей
  - Получатели не зависят от поставщиков и могут делать с информацией что им угодно
- Например, архиватор мониторной информации является всего лишь еще одним из получателей, и записывает её на постоянный носитель
  - > hello.exe > hello.out

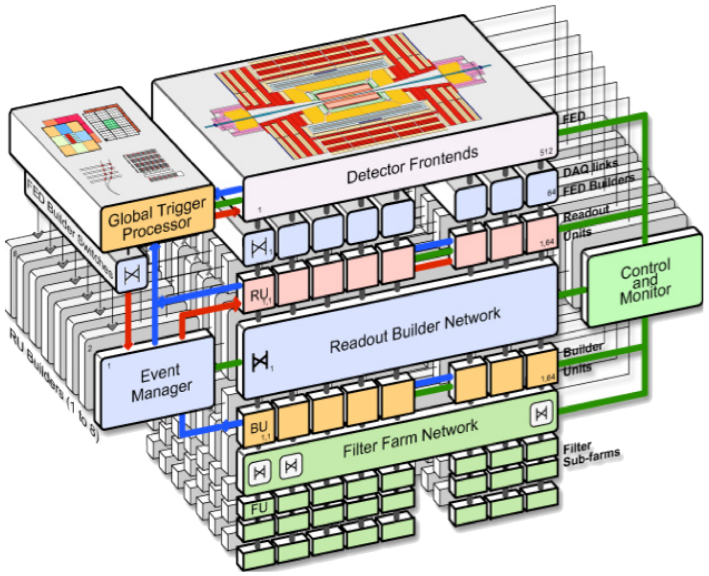
# Система мониторирования: основная структура




- Ядром мониторной системы является сервис обмена информацией:
  - Реализует потоки мониторирования
- Приложения ССД поставляют информацию:
  - О своём состоянии
  - О состоянии управляемого оборудования
- Получатели мониторной информации отфильтровывают нужные категории информации

# Что мониторируется в ССД

- Состояние системы сбора данных:
  - Состояние программных и аппаратных компонентов: потребление ресурсов (память, диск), скорость ввода/вывода, …
  - Параметры окружающей среды: температура, влажность, скорость вентиляции
  - Статистическая информация: число триггеров, эффективность триггера, число принятых и число записанных событий
- Ошибки системы:
  - Любая ненормальная ситуация должна быть сообщена с соответствующим параметром критичности
- Качество данных
  - Записываемые события должны обладать достоверностью и соответствовать ожидаемым параметрам

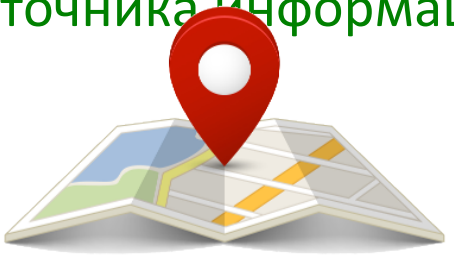


# Мониторная информация

- Простая информация имеет форму пары <имя: число>
  - «имя» является уникальным идентификатором информации
  - «число» представляет значение информации в данный момент
- Структурированные типы:
  - Программный компонент или оборудование обычно содержат множество единиц информации
  - Разные атрибуты содержат полную информацию в данный момент времени
  - Все атрибуты передаются в мониторную системы одновременно

# Общие атрибуты информации

- Источник информации:
  - Каждый информационный объект должен содержать описание источника информации:
    - Компьютер
    - Приложение
    - Модуль
- Временная метка:
  - С наилучшей доступной точностью (нс?)
  - Время в формате UTC
  - Перевод в локальное время производится получателем информации



# Классификация ошибок

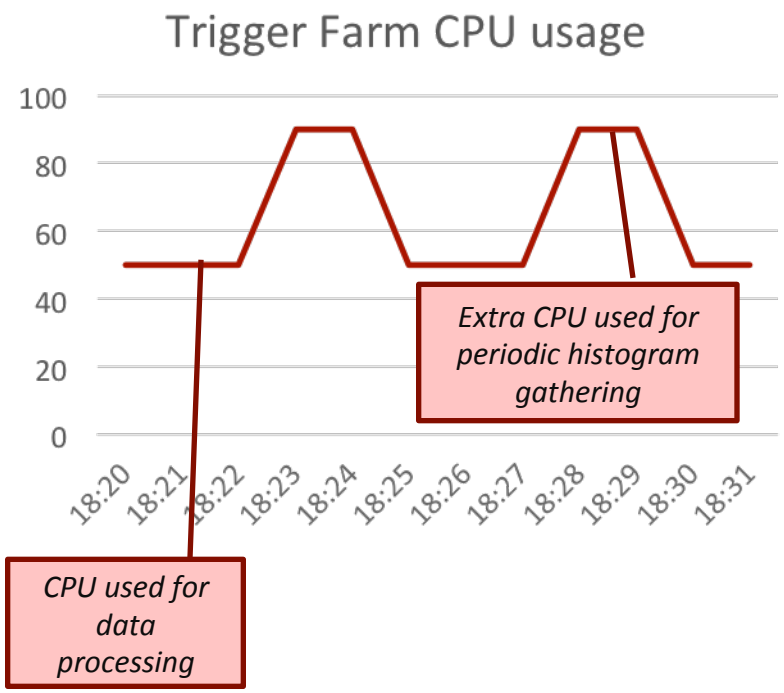
- Для правильной обработки ошибок необходимо договориться об их строгой классификации:
  - WARNING – система работает корректно, но уже на пределе некоторых параметров
  - ERROR – произошел сбой, но работа может быть продолжена дальше
  - FATAL – дальше продолжать нельзя без вмешательства
- Ошибки должны быть готовы к машинной обработке:
  - Каждая ошибка должна содержать уникальный идентификатор
  - Ошибка должна содержать все необходимые специфические параметры, например, имя файла, имя компьютера, номер события
- Ошибки должны содержать текст объяснения для оператора:
  - В случае возникновения ошибки оператор сможет просто продиктовать текст нужному эксперту

# Нельзя пренебрегать предупреждениями

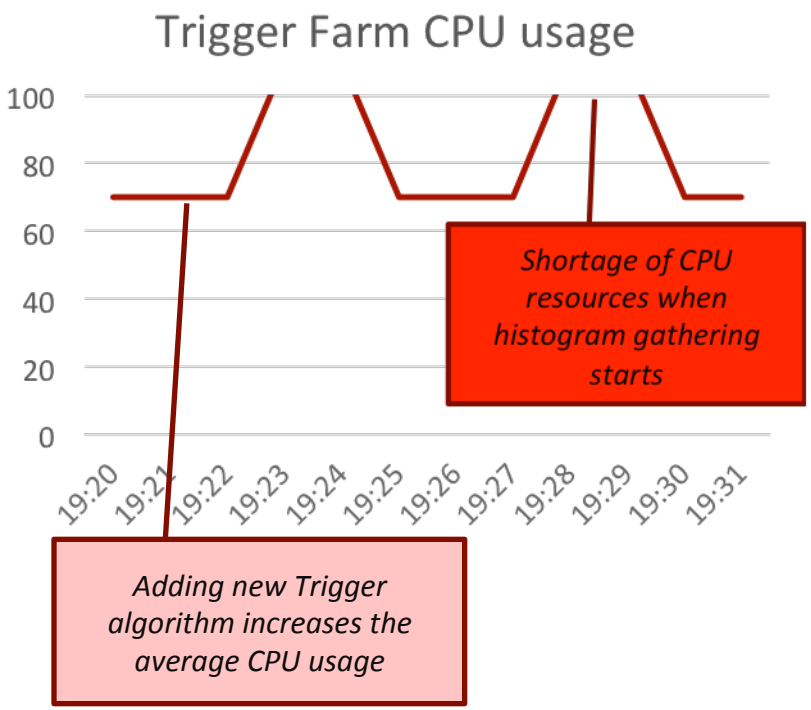
- Сообщить об ошибке достаточно просто:
  - Если что-то не так, то сообщить об ошибке
- Сообщение предупреждений очень важно:
  - Когда все ещё хорошо, но уже близко пределу, надо сообщить об этом
  - Требует дополнительных усилий для проверки условий
- Не следует пренебрегать предупреждениями, они рано или поздно перерастут в ошибки:
  - Если это случится во время набора реальных данных то это может привести к их потере

# Когда предупреждать: пример из жизни

## всё пока ещё хорошо




## «мёртвое время»

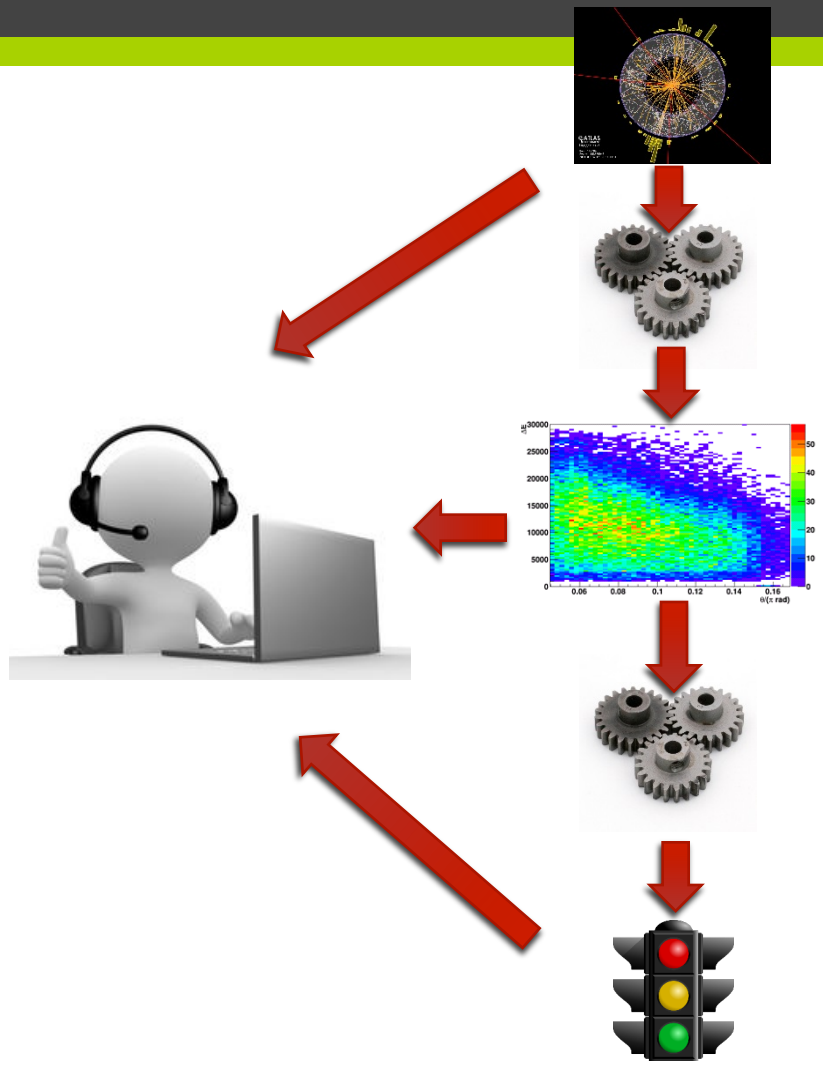


# Мониторинг качества данных

- Наблюдать только за поведением собственно ССД недостаточно:
  - ССД может работать безупречно, но принимать бесполезные или ошибочные данные
- Для проверки данных требуется специализированный сервис:
  - Детектор выдаёт корректные данные
  - Триггер отбирает нужные события

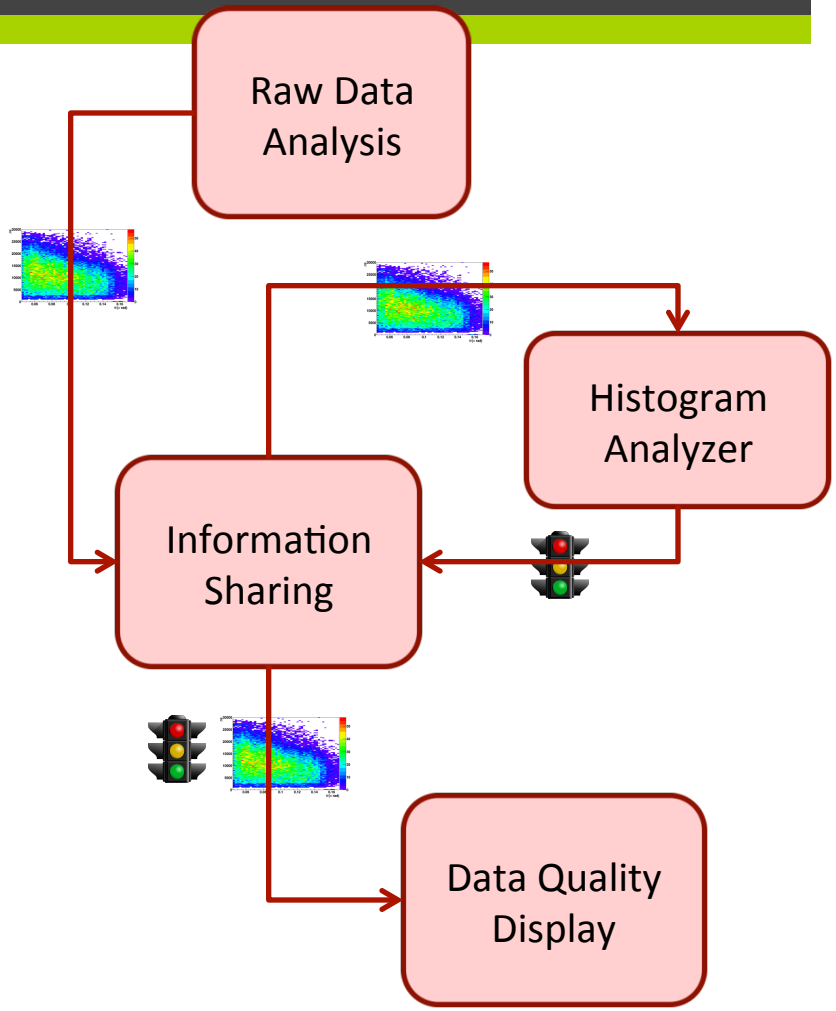
# Анализ качества данных

- Простой анализ может быть произведён путём просмотра данных или их графического представления
- Можно анализировать распределения:
  - Выдаваемые программой экспресс-анализа (реконструкции) событий
  - Специальными приложениями
- Распределения могут быть проверены:
  - Экспертом
  - Автоматическим алгоритмом



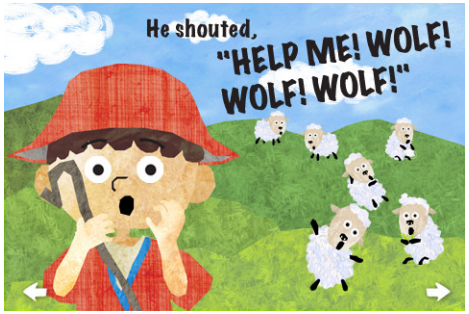
# Анализ качества данных: архитектура

- Обработка и визуализация отделены друг от друга:
  - Специализированная система автоматического анализа распределений
  - Несколько графических интерфейсов могут быть запущены несколькими пользователями одновременно

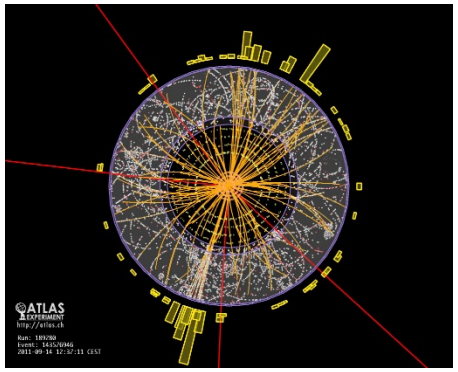


# Результаты мониторирования

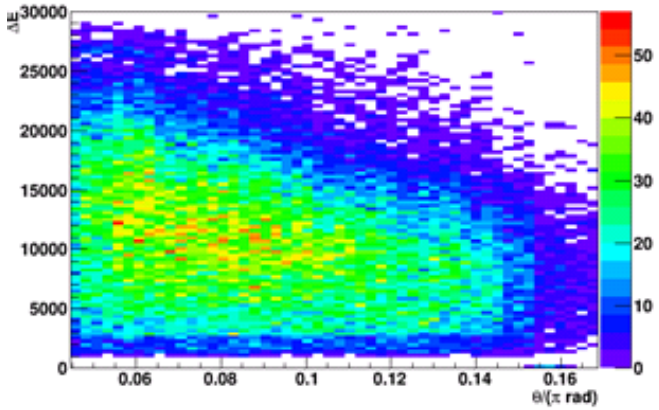
- Система мониторирования может сообщать о результатах проверки в виде «светофора»:
  - Способ привлечь внимание оператора
- Цветовая индикация
  - Зеленый - хорошо
  - Оранжевый – внимание!
  - Красный – плохо! Требуется немедленное вмешательство
  - Не надо забывать про людей не различающих цвета – следует использовать иконографику в дополнение к цвету
- Не стоит перебарщивать с количеством критических ошибок
  - Они должны соответствовать критическому положению дел



# Визуализация



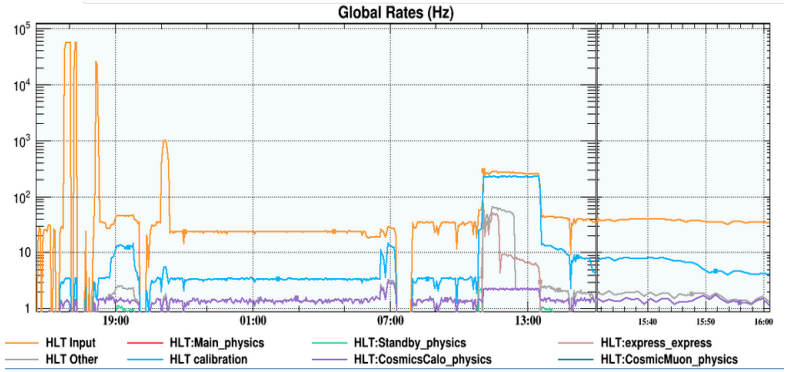

События



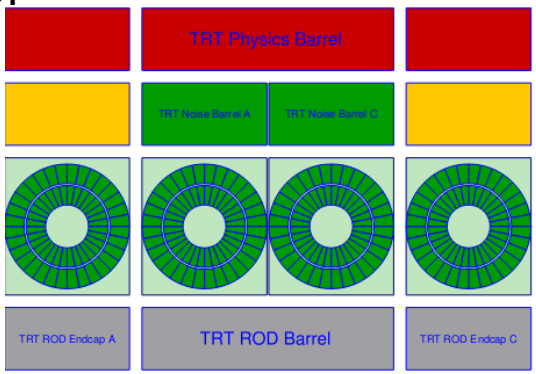

Распределения

| Deadtime Configuration | |
|---|---|
| Simple | 6 |
| Complex0 | $^{0}$11/459, $^{1}$42/381, $^{2}$9/351, $^{3}$7/350 |
| Complex1 | $^{0}$11/459, $^{1}$42/381, $^{2}$9/351, $^{3}$7/350 |

Конфигурация




Частоты

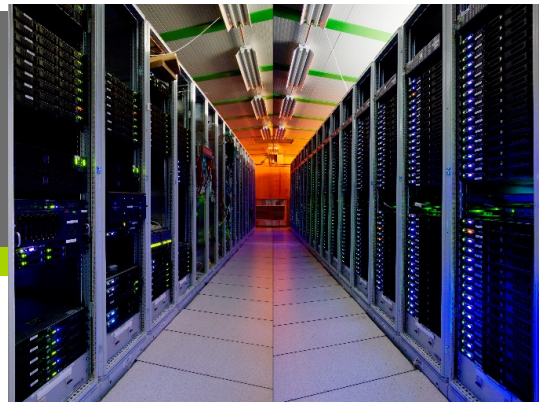



Качество данных

# Мониторные дисплеи

- Типы данных мониторинга слишком разнообразны для представления в единственном приложении
  - Event Display – показывает реконструированные события
  - Histogram display – показывает распределения:
    - Распределения, результаты автоматической проверки.
  - Status display – отдельные значения и графики (во времени)
- Тем не менее следует минимизировать их количество:
  - Один дисплей для типа информации (события, распределения, состояние)
  - Должен быть гибким и настраиваемым
  - Должен позволять показывать как текущую, так и архивную информацию

# Масштабирование



- Современные эксперименты в ФВЭ состоят из миллионов каналов детектирования
- Система сбора данных подобных экспериментов использует тысячи компьютеров:
  - ~$10^3$ компьютеров
  - ~$10^4$ ядер и программных приложений
- Как достичь масштабируемости системы мониторирования?

# Распределённый сервис обмена информацией

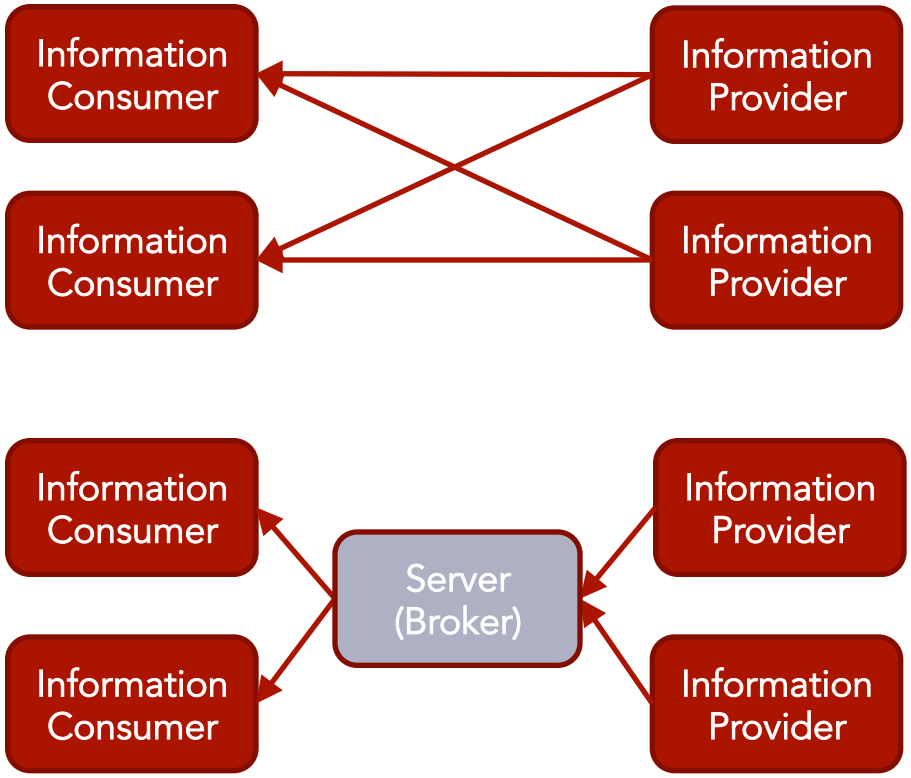
- Распределённый сервис обмена информацией является ключом к масштабированию системы
- Сервис должен представлять программный интерфейс не зависящий от места исполнения программы:
  - Поставщики и потребители информации не должны задумываться о «физическом» месте нахождения информации
  - Они должны иметь возможность легко быть перемещены с одного компьютера на другой
- В соответствии с требованиями к ССД необходимо определиться с:
  - Типом доступа к информации
  - Моделью обмена информацией

# Стратегии доступа к информации

- Push стратегия, также известная как Subscribe/Callback:
  - Потребитель информации подписывается на необходимый набор данных и получает его от поставщика по мере появления
- Pull стратегия:
  - Потребитель информации посылает запрос в сервис обмена и получает данные в результате этого запроса
- Наилучшим является поддержка обеих стратегий


# Модели обмена информацией

- Точка-точка
  - Поставщики информации объявляют о доступной информации
  - Получатели информации соединяются непосредственно с поставщиком
- Клиент-сервер
  - Поставщики информации публикуют её на некотором сервере, периодически или по запросу
  - Потребители информации соединяются с сервером для получения или подписки на информацию



# Какая модель обмена лучше?

## Точка-точка

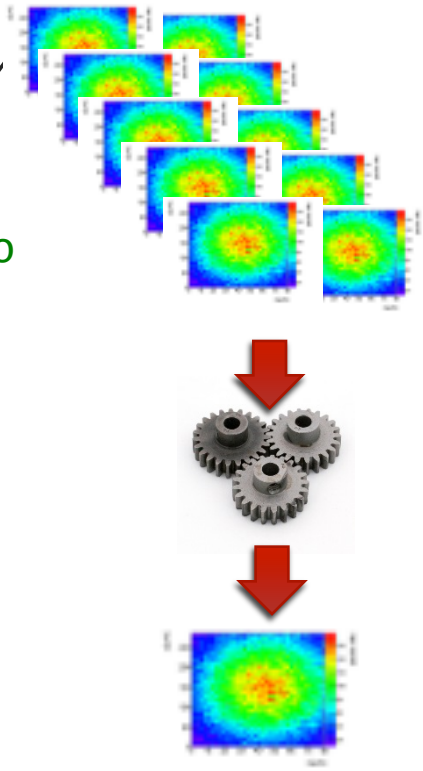
- За:
  - Отсутствует центральный сервер (критический компонент)
  - Хорошо масштабируется
- Против:
  - Требует множество соединений
  - Потребители информации могут замедлить работу поставщиков
  - Сложность реализации и сопровождения

## Клиент-сервер

- За:
  - Отделяет поставщиков от потребителей
  - Упрощает доступ к большим объёмам
  - Простота реализации и сопровождения
- Против
  - Наличие критического компонента (сервера)
  - Масштабирование требует нескольких серверов и дополнительного оборудования

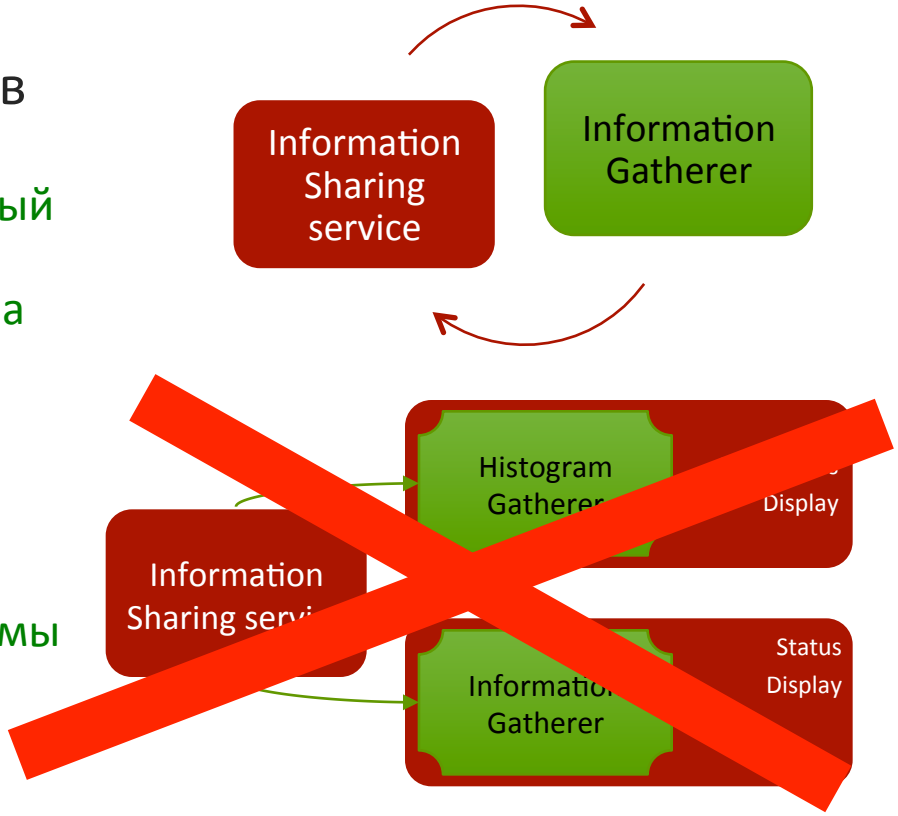
# Сбор информации

- Мониторная информация, произведённая разными приложениями (процессами) должна быть собрана и объединена, например для:
  - Собрав воедино использование ресурсов на нескольких компьютерах можно получить глобальную картину использования ресурсов
  - Собрав гистограммы всех триггерных приложений можно получить общую производительность/ эффективность триггера
- Система мониторирования должна предоставлять гибкий и производительный механизм для сбора однородной информации произвольного типа
  - Собрать все гистограммы с определённым именем и просуммировать
  - Собрать все параметры определённого типа и занести в гистограмму



# Сбор информации: архитектура

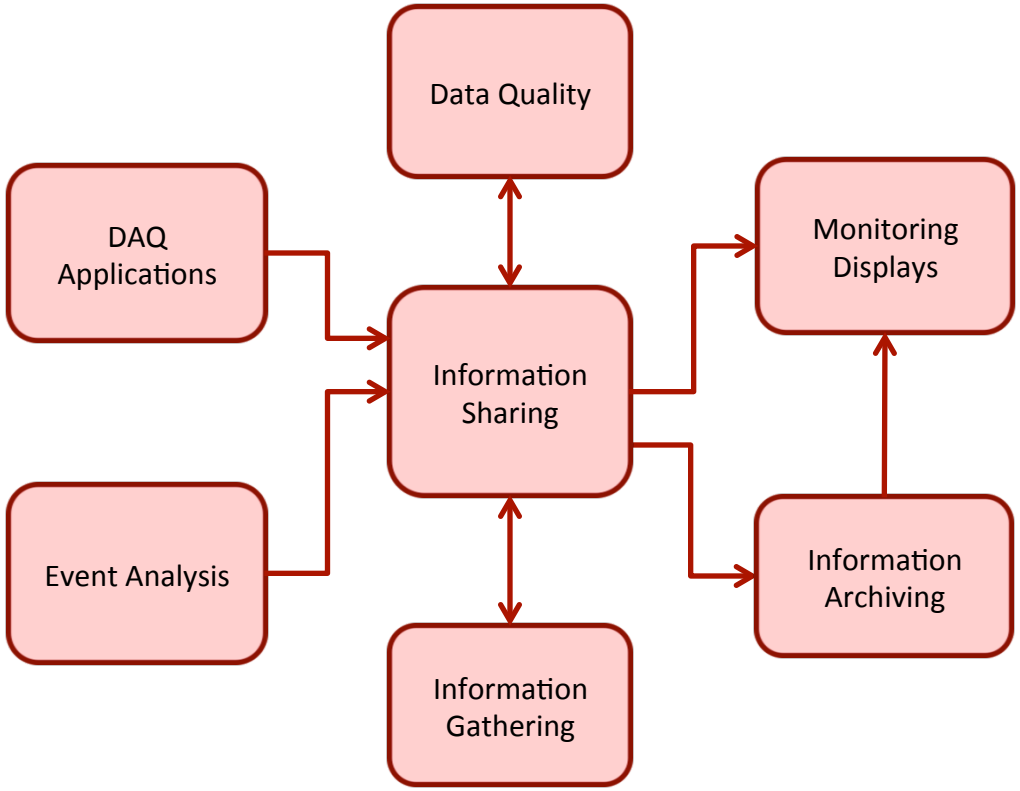
- Сбор должен производиться в одном единственном месте:
  - Необходим специализированный сервис
  - Собранная информация должна публиковаться для доступа других приложений
- Сбор должен быть гибким:
  - Поддерживать произвольные типы информации
  - Поддерживать разные алгоритмы сбора:
    - Суммирование, усреднение
    - Дополнительные алгоритмы



# Архивирование мониторной информации

- В идеале вся мониторная информация должна архивироваться:
  - Для последующего анализа
  - Для диагностики проблем
- Концептуально архиватор является одним из получателей информации
- На практике задача является сложной из-за огромного объёма информации
- Предпочтительно иметь разные архиваторы для разного типа информации:
  - Гистограммы, ошибки, статус

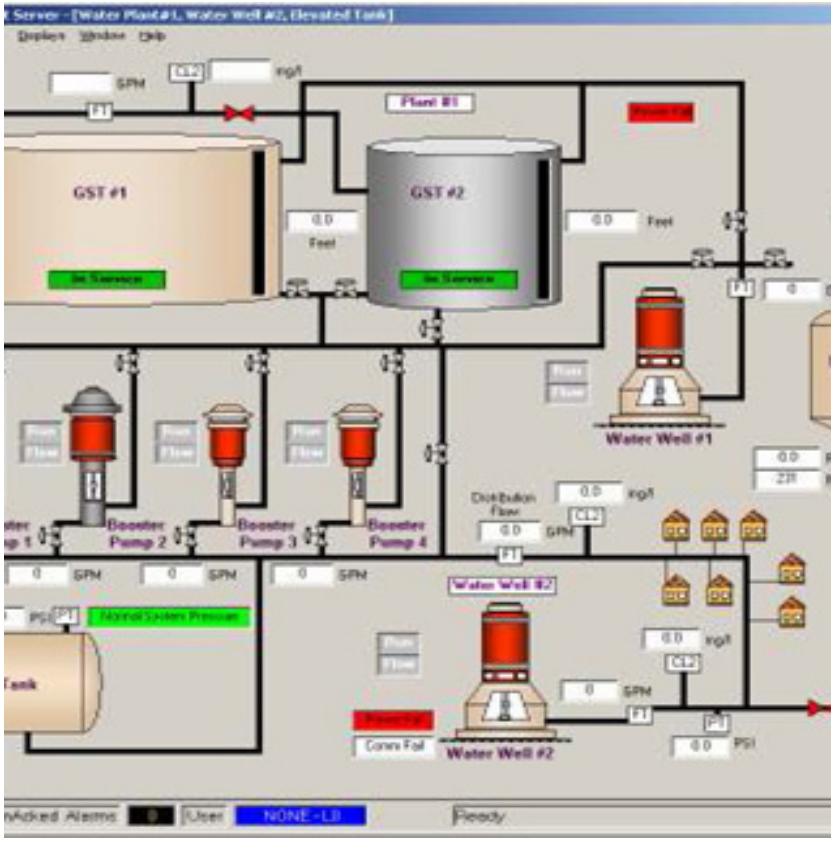
# Архитектура распределённого масштабируемого мониторинга ССД




- Ядром мониторирующей системы является информационный сервис
- Все другие сервисы являются либо поставщиками? Либо потребителями
- Некоторые из них могут одновременно быть и теми и другими

# Коммерческие решения: SCADA

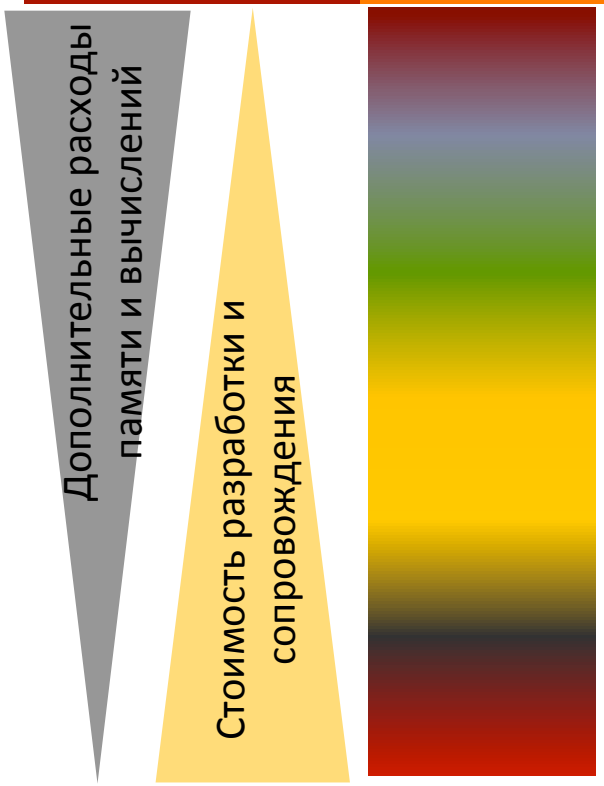
- Supervisory Control and Data Acquisition (SCADA)
  - В основном предназначена для управления но подходит также и для мониторинга
  - Современные реализации хорошо масштабируются
- Системы SCADA могут быть реализованы на LabView:
  - Графический язык разработки
  - Мощный настраиваемый графический интерфейс
- В основном используется для управления и наблюдения за оборудованием
  - Не всегда подходит для некоторых задач мониторирования, например для проверки качества данных



# Технологии разработки

- Обмен информацией:
  - Выбор технологии взаимодействия между процессами:
    - В идеале интерфейс информационного сервиса не должен зависеть от технологии взаимодействия (IPC)
  - Выбор стратегии и модели обмена:
    - Push, Pull или смешанная, клиент-сервер или точка-точка
- Формат обмена данных:
  - Может зависеть или нет от выбранной технологии взаимодействия
- Технология архивации:
  - Следует тщательно учитывать объём и частоту обновления информации
- Технология визуализации:
  - Следует использовать один и тот же графический интерфейс для доступа к текущей и архивированной информации

# Спектр коммуникационных технологий



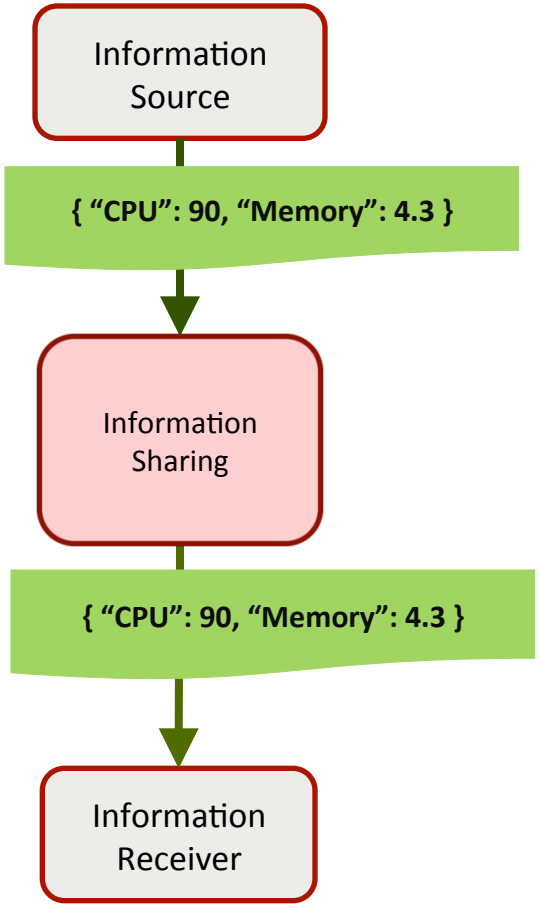

- Internet Communication Engine (ICE) от ZeroC
- CORBA: TAO, omniORB, JacORB, ORBacus, …
- Системы обмена сообщениями: Qpid, ActiveMQ, RabbitMQ, ...
- Библиотеки: Boost ASIO, ZeroMQ, ACE, …
- Socket API

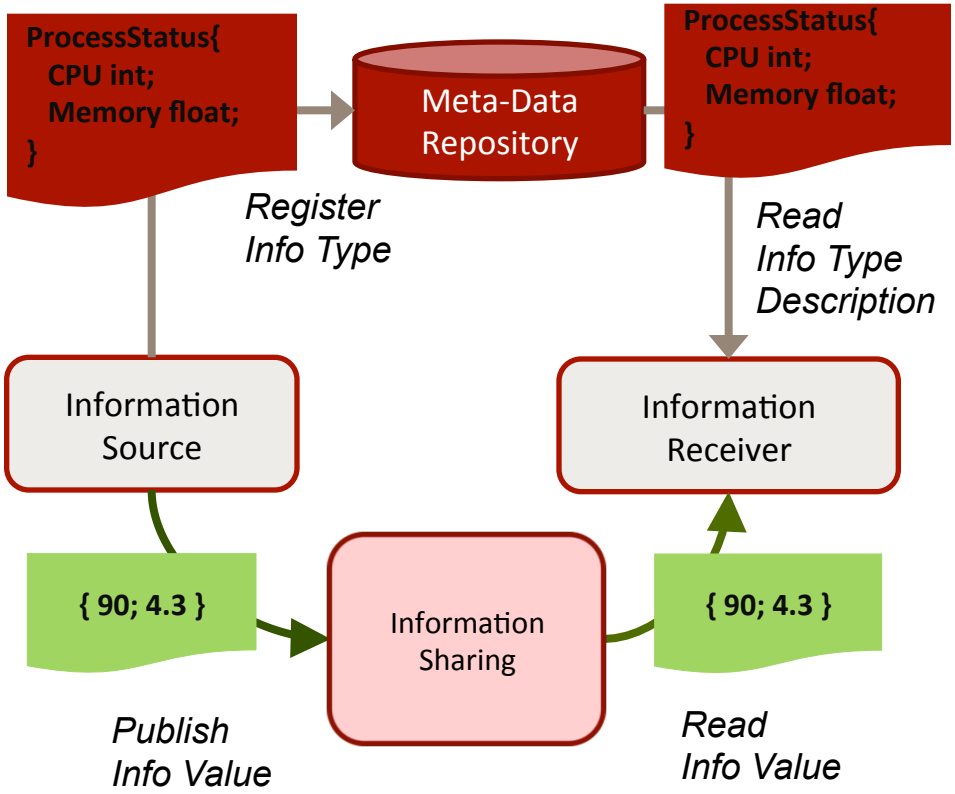
- Выбор технологии зависит от требований:
  - Размер системы, языки программирования, доступные ресурсы, сроки разработки…

# Формат данных для передачи по сети

- Для протокола HTTP JSON является естественным форматом:
  - Например информация о процессе может быть в виде: { “CPU”: 90, “Memory”: 4.3, … }
- Преимущества JSON:
  - Простой, текстовой, самодостаточный
- Слабым местом является производительность:
  - Разбор JSON требует много ресурсов CPU
  - Передача имён атрибутов увеличивает объём данных
- В качестве альтернативы можно использовать более компактные форматы:
  - Например google/protobuf

# Мета-информация



# Технологии архивирования данных

- Выбор очень сильно зависит от требований
- Большие эксперименты ФВЭ записывают ~10TB мониторной информации в год.
- Традиционные СУБД (SQL) не слишком подходят для этой задачи
- Подход «Больших Данных» (Big Data) является новым трендом
  - Hadoop, Teradata, Cassandra…

# Технологии визуализации

## GUI библиотеки

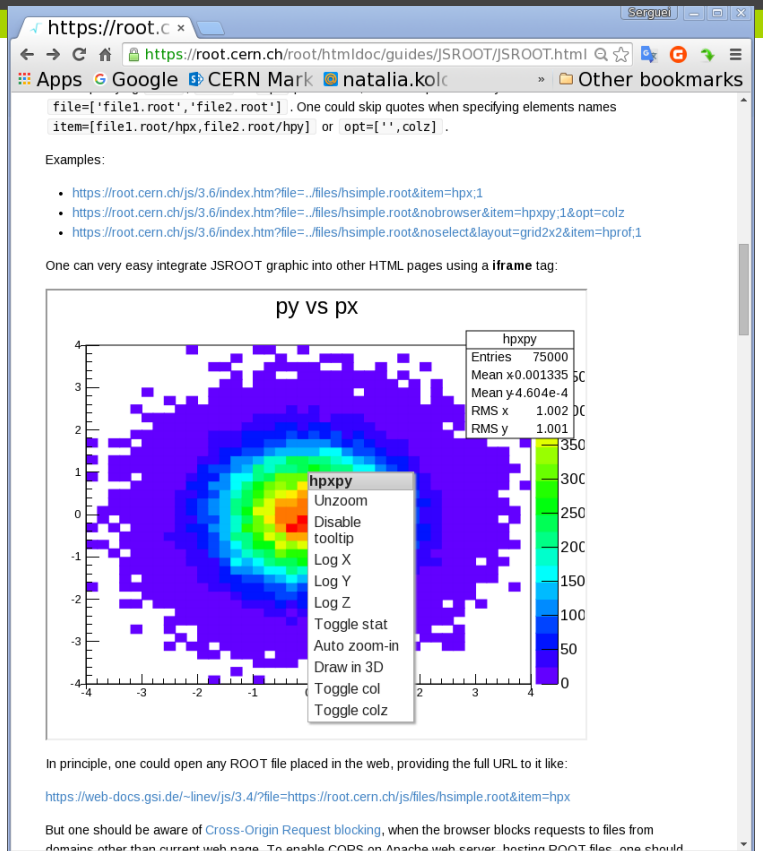
- Обычно привязаны к языку программирования:
  - Qt – C++, Python
  - Swing – Java
- Требуется установка дополнительных библиотек
- Хорошая производительность

## WEB браузеры

- Легко настраиваемое поведение (javascript, CSS, etc.)
- Не требуется установка дополнительного ПО
- Доступность на множестве систем/устройств
- Плохо масштабируется

# Технологии визуализации: ФВЭ

- Задачи мониторирования в ФВЭ требуют построение и анализ распределений (гистограмм):
- ROOT использует C++
- ROOT 6 включает в себя библиотеку JavaScript для использования в Web браузерах:
  - JSROOT



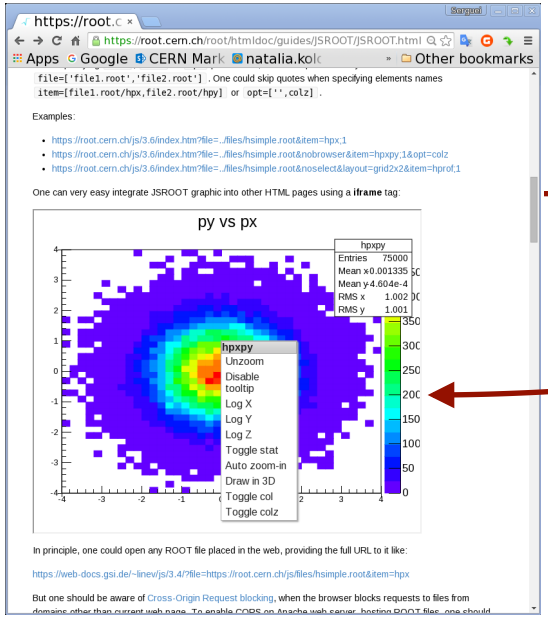

https://root.cern.ch/root/htmldoc/guides/JSROOT/JSROOT.html

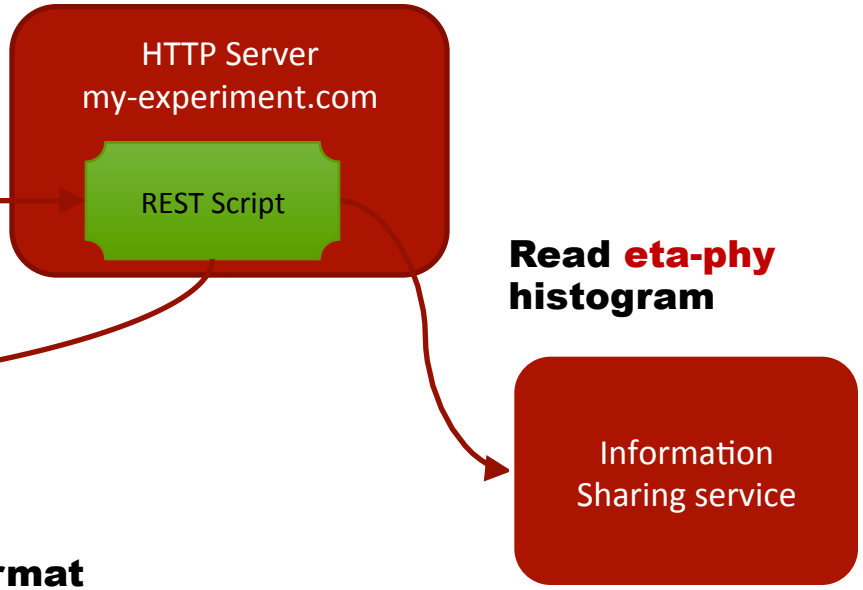
# Доступ к мониторной информации

- В настоящее время преобладает следующее мнение – если информации нет на Web – её не существует!
  - Это особенно верно в области ФВЭ, где все большие эксперименты проводятся международными научными сообществами
- Использование технологии REST является простым способом добавления мониторной информации на Web
- REST – **Re**presentational **S**tate **T**ransfer
  - На базе HTTP
  - Без состояния (контекста)
  - Архитектура клиент-сервер

# Каждый мониторируемый объект в сети является ресурсом!

- Каждый информационный объект имеет свой собственный URL
- Для чтения объекта используется запрос HTTP GET:
  - **GET http://my-experiment.com/histograms/eta-phy**



# Мониторирование в сравнении с управлением

- Система мониторирования обычно считается чем-то простым и несущественным, что можно сделать потом, в свободное от работы время
  - Это не так!

- Управление не будет работать без мониторинга:
  - Нельзя утверждать что вы чем то управляете если вы не знаете его состояния
- Отсюда легко вывести:
  - При разработке системы сбора данные, мониторный компонент должен быть запущен одним из первых!


"It looks like you have everything under control."

# Приоритеты


- Создание ССД должно начинаться с проработки системы мониторирования!
  - Интерфейсы мониторирования должны быть тщательно проработаны с самого начала
  - Система мониторирования (прототип) должна уже работать при первых тестах системы сбора данных
  - Все другие сервисы должны её использовать с самого начала
- Преимущества подобного подхода:
  - Уже во время разработки и настройки ССД вся необходимая информация о состоянии и производительности системы будет в наличии
  - Упрощает тестирование и диагностику ССД
  - Использование системы мониторирования на ранних этапах разработки ССД позволит своевременны выявить её слабые стороны и скорректировать их в сторону улучшения