Design and Implementation of a Monitoring System

Serguei Kolos
(University of California, Irvine)

The Outline

- What is Online Monitoring?
- The basic Monitoring System Architecture
- What shall be monitored in a DAQ system?
- Scaling up to the size of the HEP experiments
- Technologies for the Monitoring System implementation
- Final Remarks

What Monitoring is used for?

- A good Monitoring System should be capable to answer all possible questions about the system being monitored:
  - What happened?
  - Where that happened?
  - When that happened?
- There are some questions it is not competent to answer by itself, e.g.:
  - What to do next?
- But it provides all possible information to those who can answer such questions

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

How a Monitoring System is organized?

- All of us are equipped with a perfect monitoring system:
  - We have 5 types of sensors for detecting events in the outside space
  - The information is transferred via nerve fibers to our brains which initiates an appropriate reaction
  - Important information is recorded for the future reference

- Monitoring for the DAQ system acts in a similar way:
  - The HW and SW elements play a role of sensors by publishing their status to the Monitoring System
  - The Monitoring System transports this information to the “brains”:
    - A Human operator
    - An Expert system
  - It also records the sub-set of information for the off-line analysis

A simplest Monitoring Code Sample

This is a monitoring message

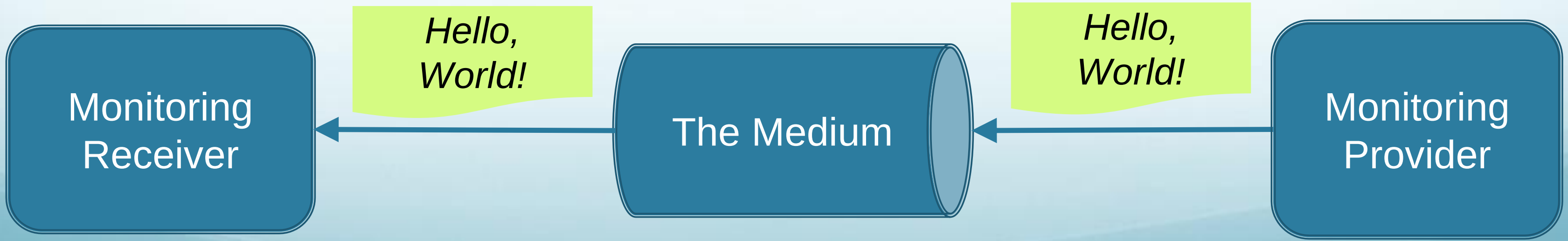
```
print("Hello, World")
```

This is the monitoring API function

The Basic Monitoring system architecture

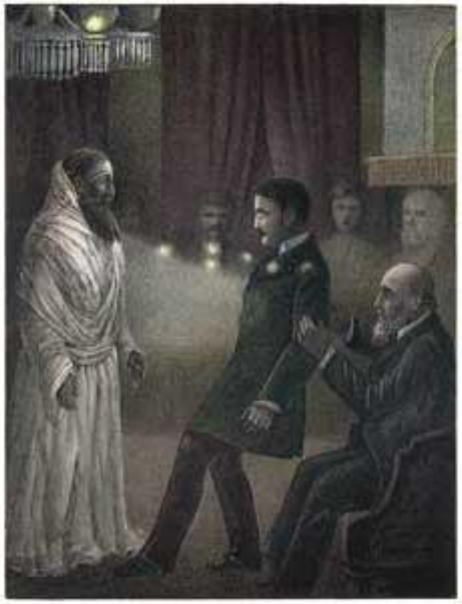
- The Monitoring API – is a well-designed set of functions, which can be used by:
  - Monitoring Provider (***SW application***) to manifest its own state or the state of the controlled HW
  - Monitoring Receiver (***SW application***) to receive the monitoring information for some purpose, e.g. displaying, archiving, analyzing, etc.

- The Medium:
  - Implements the Monitoring API and provides transportation of the information from providers to receivers



The Open Architecture

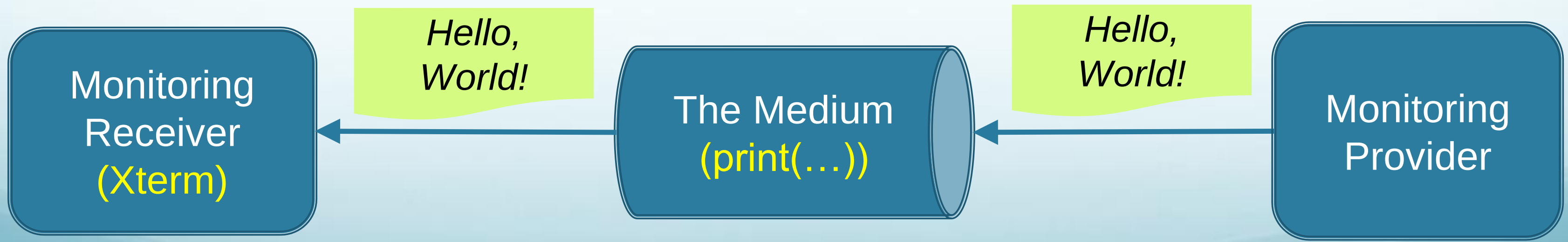
- Having a Medium in between Providers and Receivers make the architecture of the monitoring system open:
  - Providers and Receivers can be freely added/removed at any moment
  - Neither Providers nor Receivers know anything about one another
  - New Mediums can be added at any moment without affecting the clients:
    - Transporting messages through different networks
    - Implementing specific destination for some types of messages, e.g. a database



Is our hello.py application well designed?

print("Hello, World")

- The issue is that the "print(…)" function provides both the API and the Medium implementation tightly bound together
- The destination for the messages passed to "print(…)" is fixed
- Can we do better?



Logging Module to the rescue

```
import logging
logging.basicConfig(format='%(asctime)s %(levelname)s [%(filename)s:%(lineno)s - %(funcName)s()] %(message)s', level=logging.INFO)

logging.info("Hello, World")
```

Use standard well-designed API

The output format can be changed at any moment

```
2017-02-02 16:21:13,583 INFO [hello.py:4 - <module>()] Hello, World
```

A lot of extra information is available for free

A custom log stream can be easily implemented and plugged in without touching the application code

Some Basic Tips

- Don't neglect the Monitoring – use it properly from the very beginning of a new software project development :
  - If you start using the print-outs at the beginning then sooner or later you will spend weeks migrating to a proper monitoring API
- Stick to the Open Architecture if possible – separate Monitoring API from the data transport
- Use a standard API whenever it is available:
  - Java and Python have nicely designed logger APIs
- If the chose programming language has no such API, e.g. C++:
  - Think about using a third party implementation or implement your own one as early as possible

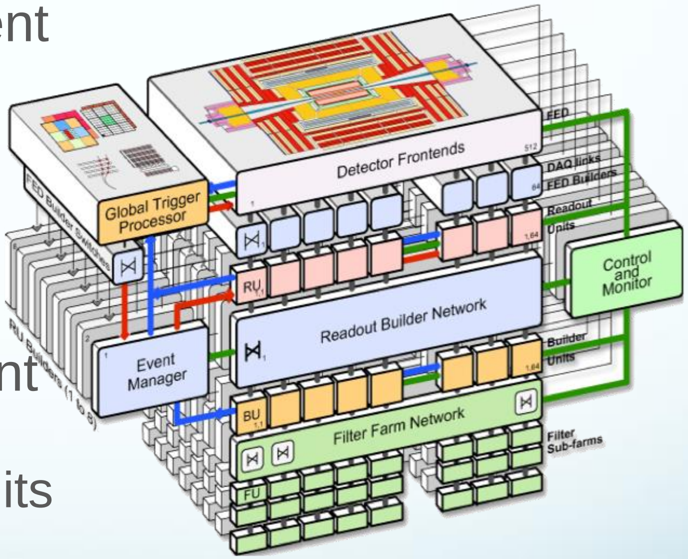
Monitoring In the DAQ system

What shall be monitored in a DAQ System

- DAQ System status:
  - Health of individual HW & SW components:
    - resources consumption, IO rates, etc.
  - Surrounding environment:
    - temperature, humidity, etc.
  - Operational statistics:
    - number of triggers, number of recorded events, etc.
- Data Quality Monitoring:
  - Recorded data shall be constantly monitored for sensibility
- System errors:
  - Any abnormal or even suspicious situation shall be immediately reported

The Monitoring System Scalability Requirements

- DAQ systems may be drastically different in their sizes depending on the experiment
- A DAQ system of a modern HEP experiment includes:
  - O(1K) computers and network devices
  - O(10K) SW applications
  - O(100K) HW sensors
- A monitoring system for such an experiment shall be implemented in a distributed way:
  - Puts forward some extra requirements to its design and implementation



Use Logging API for Error reporting

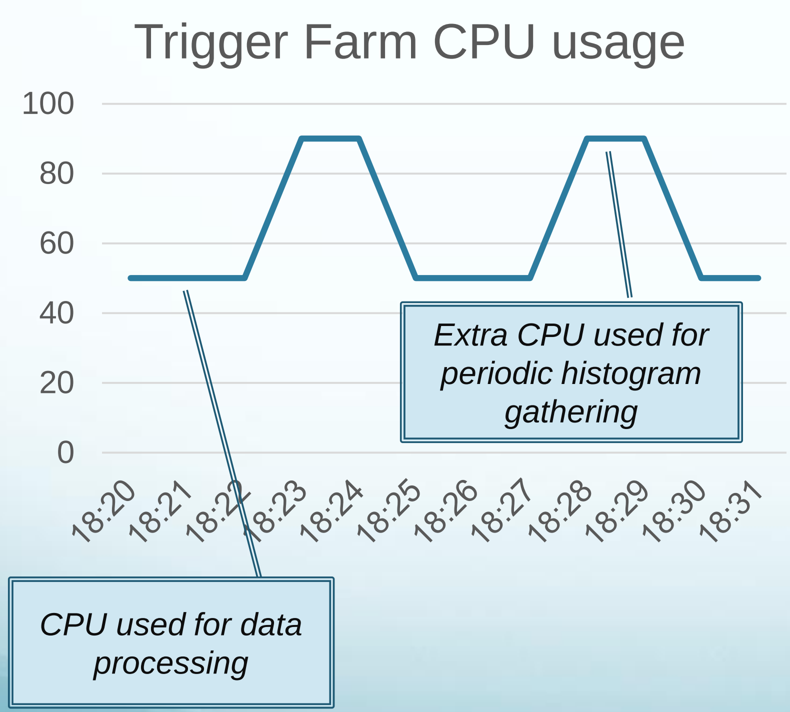
- Logging API is an ideal candidate for error reporting:
  - Some extra work has to me done to use it properly
- Errors should contain clear explanation of the actual problem:
  - A shifter calling you in the middle of a night can just read it
- Errors shall be ready for machine processing in the first place:
  - Every error shall contain a unique ID which corresponds to the given issue
  - Error shall contain all parameters, which are specific for the issue occurrence, e.g. file name, event ID, computer name, etc.

Error Types

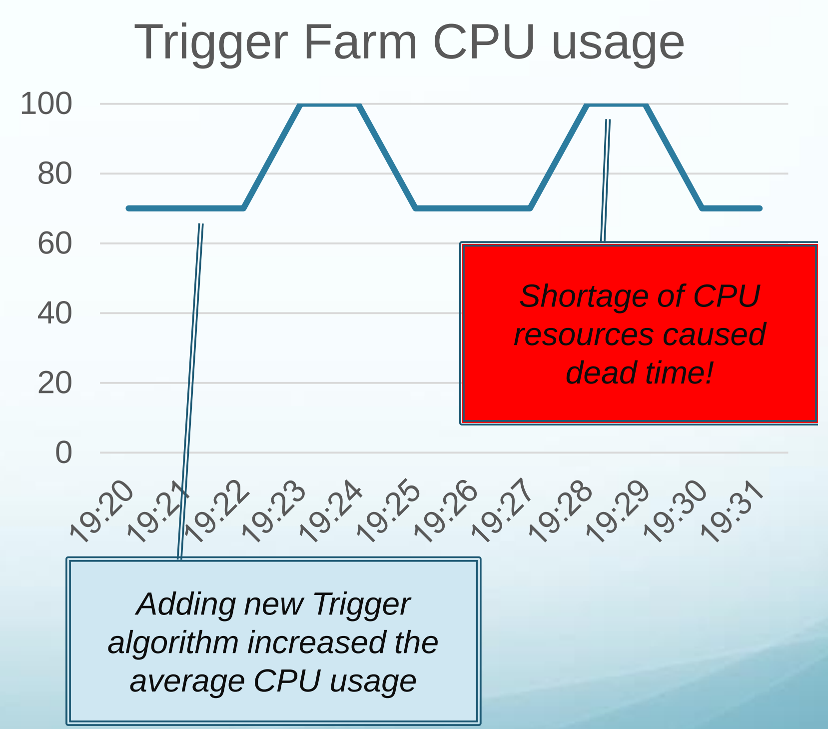
- Reporting an ERROR or CRITICAL message is straightforward:
  - When something goes wrong, the monitoring system shall scream
- Reporting WARNINGs is much more cumbersome but it might be equally important:
  - Nothing wrong happened so far, but the system is dangerously close to a certain limit
  - Requires some extra health-checking code to verify operational conditions
- Don't neglect warnings as sooner or later they become errors:
  - If that happens during data taking it may seriously affect the efficiency

When to issue a WARNING: A real life example

Everything goes well so far



CPU shortage causes dead-time



DAQ system status information

- A DAQ system has many numeric parameters to be monitored:
  - Logging API is not very convenient for that

- A simple piece of monitoring information has a form of a <name: number> pair:
  - The “name” is a unique identity of the information
  - The “number” represents the information value for the given moment

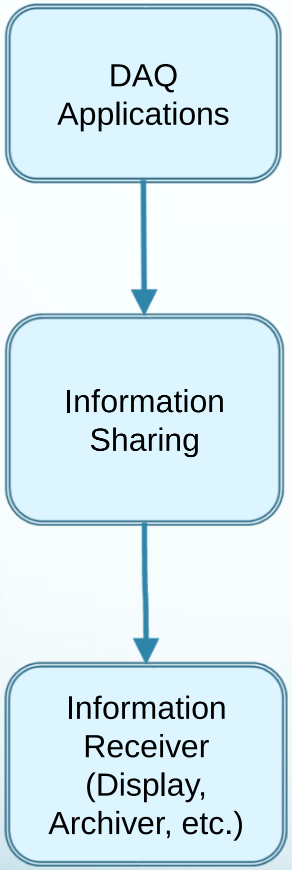
- More generally a piece of information is represented by a <name: object> pair:
  - The “object” represents a DAQ SW or HW element containing multiple properties to be monitored
  - Individual attributes contain values of the properties for the same time point
  - All attributes are sent to the Monitoring Stream in one go

Common Information Properties

- The origin of the information:
  - Computer Name (or IP address)
  - Application ID
  - HW Module ID

- The time stamp:
  - Use the best possible precision (nanoseconds)
  - Use UTC time
  - Conversion to the human readable local time shall be done by receiver applications with respect to the specific context and location

Generalized Monitoring Architecture



- The core of the Monitoring System is implemented by the Information Sharing Service:
  - Provides the Monitoring API
  - Provides one or several Medium implementations
- Any DAQ application shall report to the IS service the following information:
  - It's own state
  - The state of the controlled HW
  - Errors
- Receivers of the monitoring information shall be able to receive an arbitrary sub-set of the available information

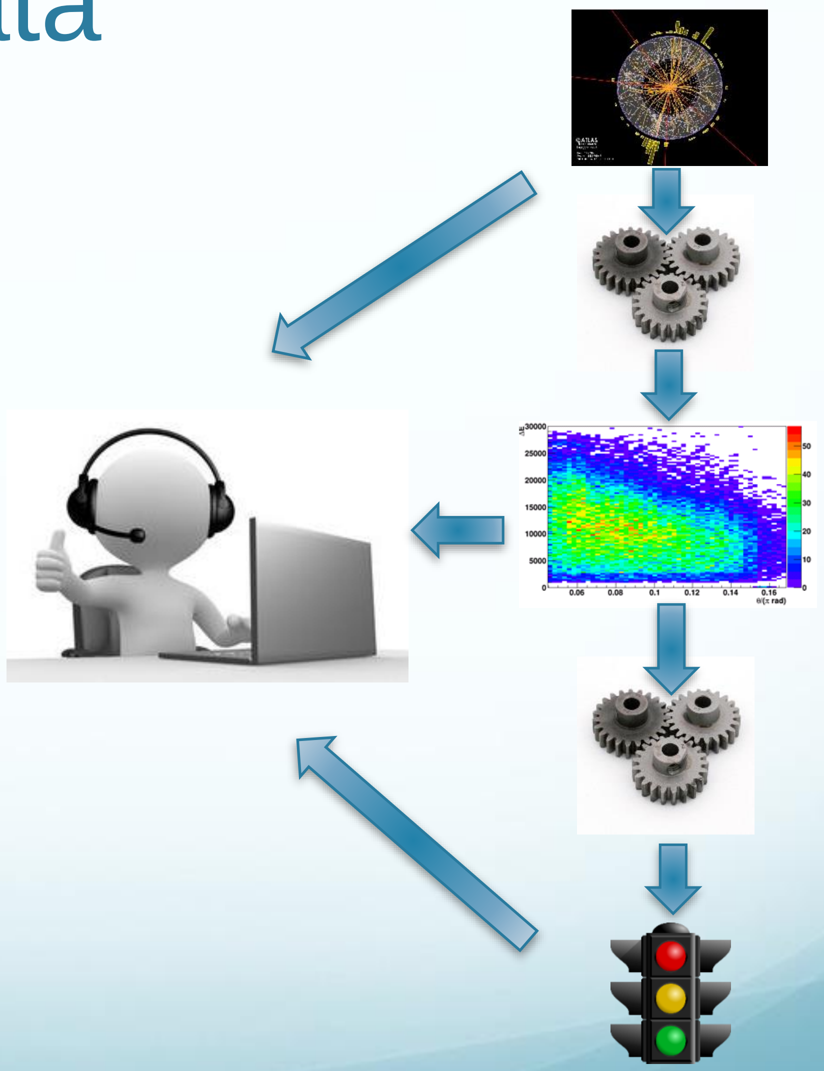
The DAQ specialty: Data Quality Monitoring

Data Quality Monitoring

- Watching out the behavior of the DAQ system itself is not sufficient:
  - The DAQ may be functioning perfectly well but at the same time is taking meaningless data, for example, due to wrong calibration constants

- A dedicated service is required for checking that:
  - Detector readout provides meaningful data
  - Trigger does reasonable selection

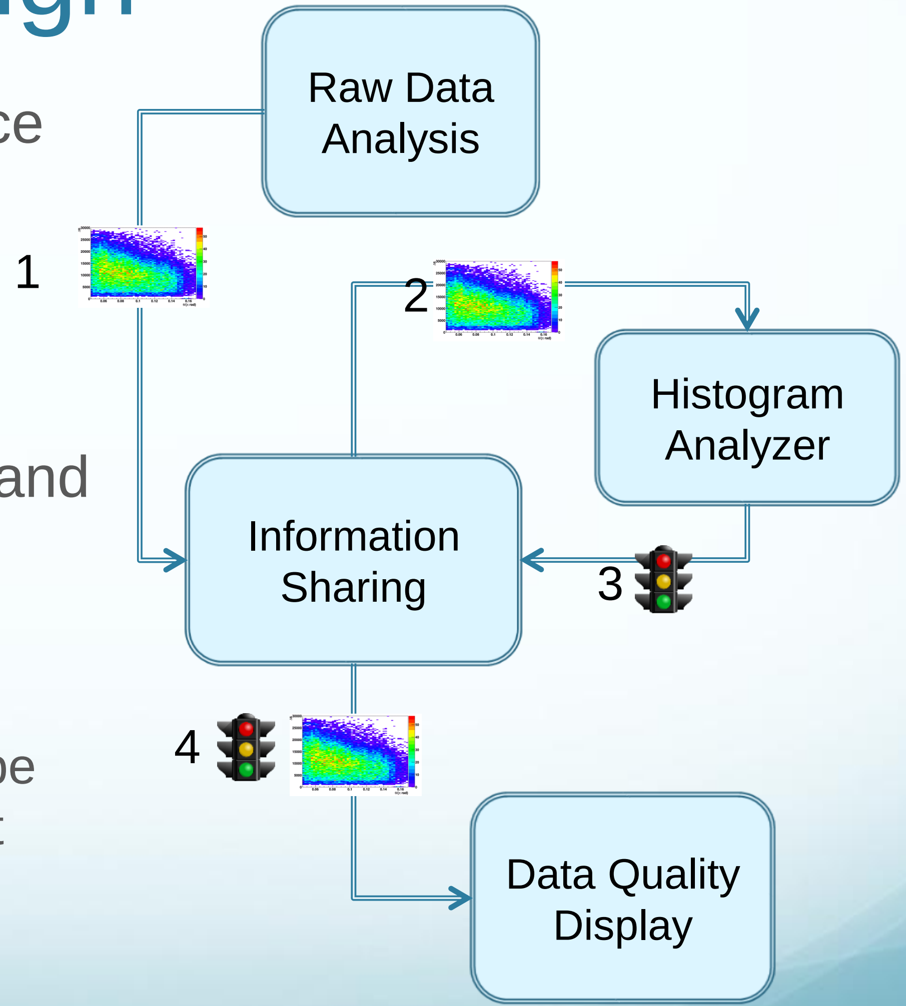
Data Quality analysis: The Flow of Data



- Simple analysis can be done by using graphical representation of a reconstructed event:
  - Only experts can do that!
- Advanced analysis can use histograms as input:
  - Histograms are produced either by the trigger selection software or by dedicated applications using event sampling
- Histograms can be checked:
  - By eyes of an expert
  - By automatic algorithm producing alarms

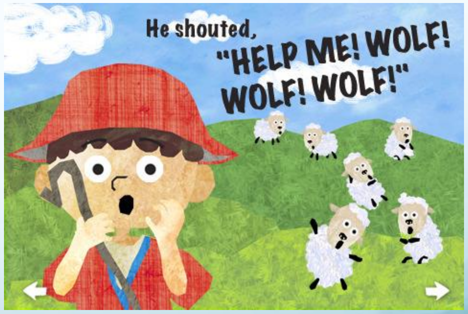
Data Quality Monitoring: The Design

- Use Information Sharing service for:
  - Publishing histograms
  - Publishing alarms
- Separate out data processing and visualization:
  - A dedicated software system for automatic histograms analysis
  - Multiple display instances can be used simultaneously by different users



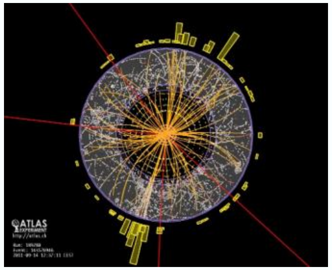
A Special Type of a Monitoring System Outcome: Alarms

- A Monitoring system produces an alarm when immediate shifter attraction is required
- This can be achieved by using visual items which can change their colors (and shapes - don't forget about color blind people):
  - Green – good,
  - Orange – stay tuned,
  - Red – an ALARM
- A system may also make a sound
- Do not overuse alarms!
  - Alarms shall be used for displaying critical errors only

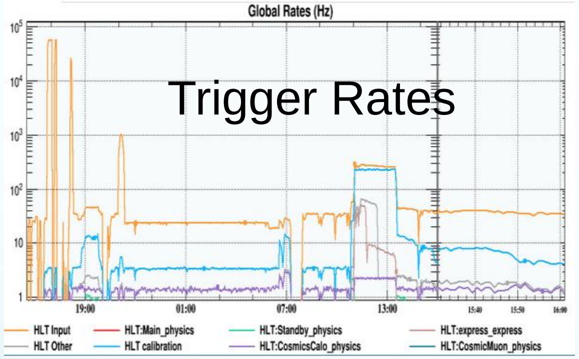


Visualizing Monitoring Information

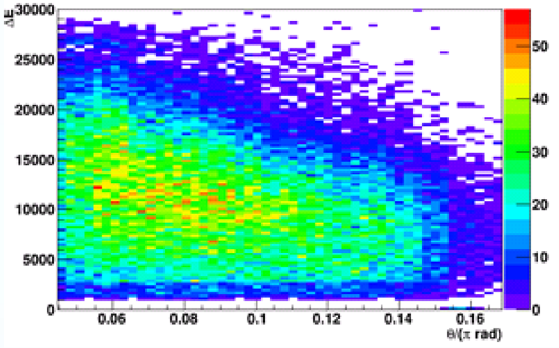
Visualization: Information Types



Raw Events



Trigger Rates

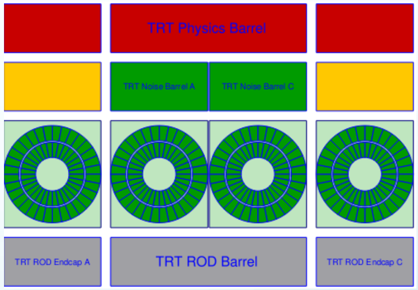


Histograms

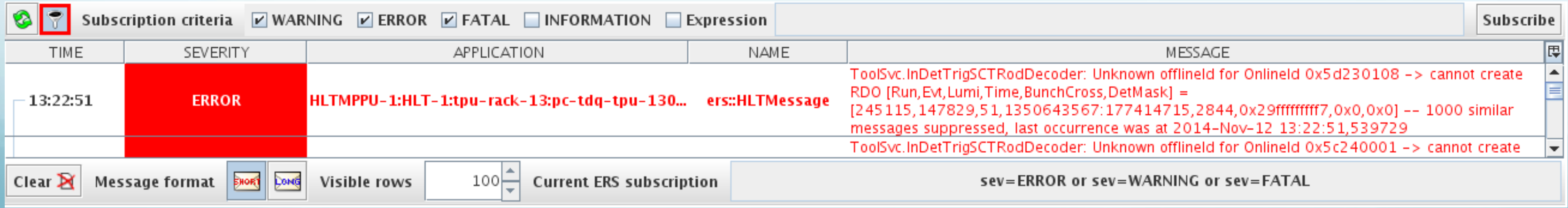
Deadtime Configuration	
Simple	6
Complex0	^{0}11/459, ^{1}42/381, ^{2}9/351, ^{3}7/350
Complex1	^{0}11/459, ^{1}42/381, ^{2}9/351, ^{3}7/350

Configuration

Data Quality status and alarms



Errors

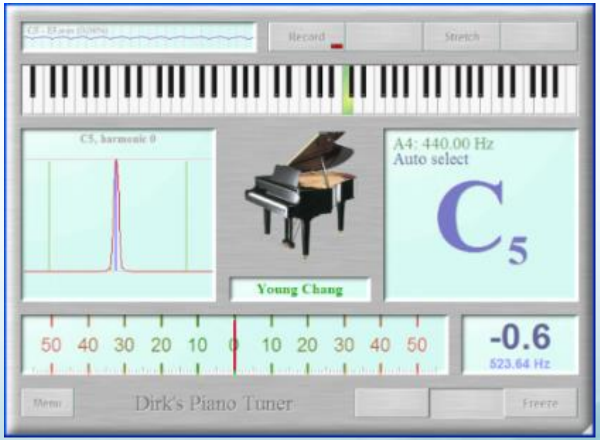


Monitoring Displays

- Monitoring data types are too diverged to be presented by a a single application
  - Event Display – shows reconstructed events
  - Histogram display - shall be configurable:
    - Histograms, references, draw options, etc.
  - Status display – shows individual values and evolution graphs
- Try to minimize the number of displays:
  - One display per information type
  - It shall be flexible and configurable
  - It shall be used for displaying both the online and archived information

Explore the power of Visualization!

- Properly designed Monitoring GUIs is one of the keys for successful experiment operation:
  - They can replace an expert by providing an unexperienced user with missing capabilities
  - Spotting problems becomes trivial if they are clearly presented
  - Well known problems can also be fixed by a non-expert with a clear guidance of the dedicated GUI



Scaling up the Monitoring System

The HEP Experimental Realm

- The modern HEP detectors have millions of data channels
- The DAQ system for such an experiment consists of:
  - O(1K) computers
  - O(10K) CPU Cores and SW applications
- How does this affect a Monitoring System?

Distributed Information Sharing service

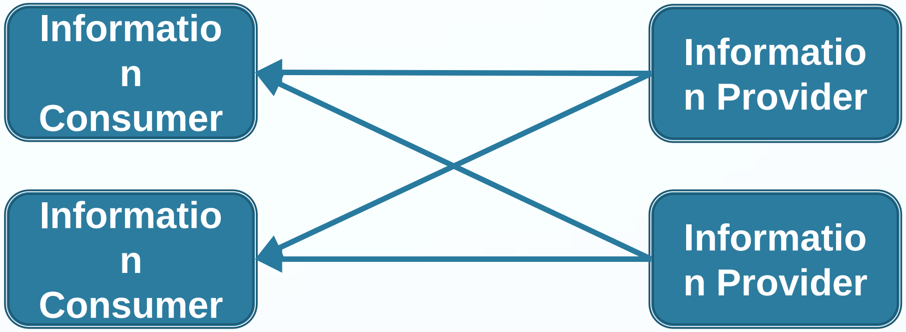
- The core of the Monitoring system is provided by the Information Sharing service:
  - The service must be scalable to a given number of Providers and Receivers
- There are two key properties of the Information Sharing architecture which have to be chosen with respect to the specific requirements of the experiment:
  - The information access model
  - The communication model

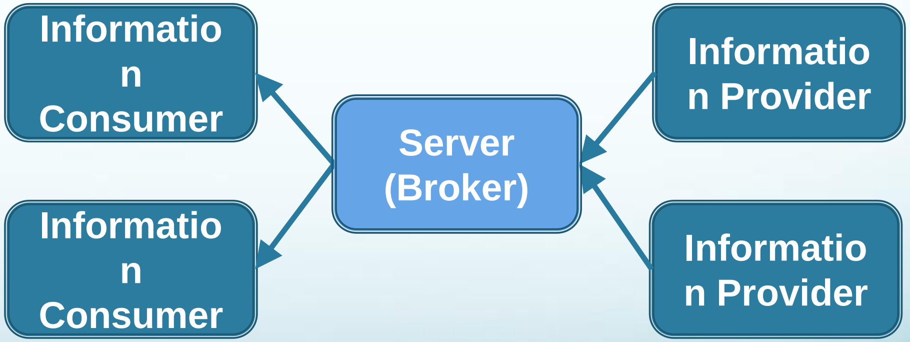
Push vs Pull Information Access Models

- Push Model, also know as Subscribe/Callback (asynchronous):
  - An Information Receiver subscribes for the relevant subset of the information to get only what it needs and only when the information is updated
- Pull Model (synchronous):
  - An Information Receiver sends a request to the IS service when it needs some information and get it back as a result of this request
- Supporting both models might be the best option:
  - Each Receiver may choose the most appropriate way for information access

Communication Model

- Peer-to-Peer:
  - Information Providers announce available information
  - Information Consumers directly connect to the Providers to read information or subscribe for the updates
- Client-Server:
  - Information Providers update information on some server periodically or on demand
  - Information Consumers connect to the server to read information or subscribe for the updates





Which one is better?

Peer-to-Peer

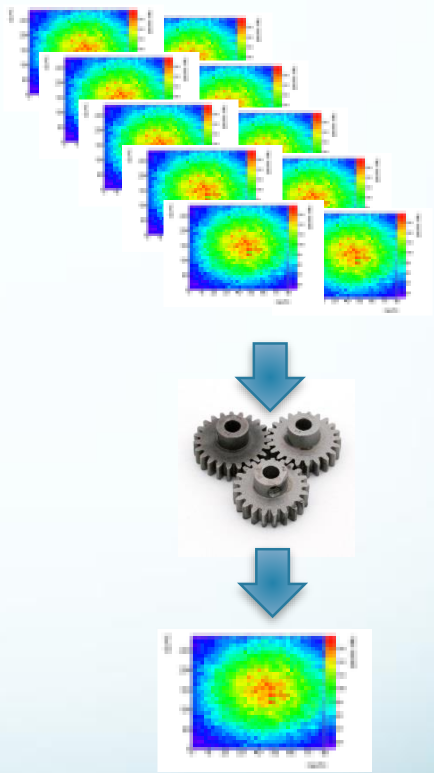
- Pro:
  - Has no single point of failure
  - Scales better
- Cons:
  - Requires more connections
  - Providers are exposed to Receivers due to the direct connections
  - More difficult to implement and maintain

Client-Server

- Pro:
  - Separates Providers from Receivers
  - Simplifies information access
  - Implementation and maintenance is simple
- Cons:
  - Has a single point of failure
  - Scalability requires multiple servers and additional HW resources

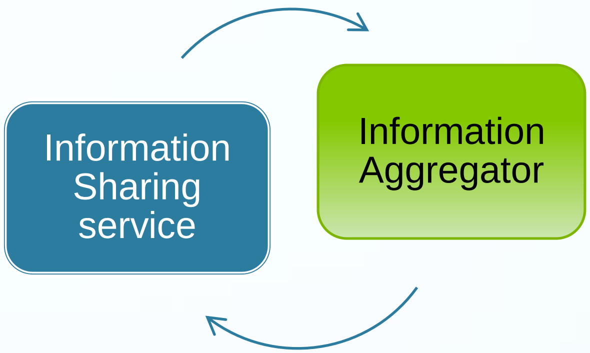
Information Aggregation

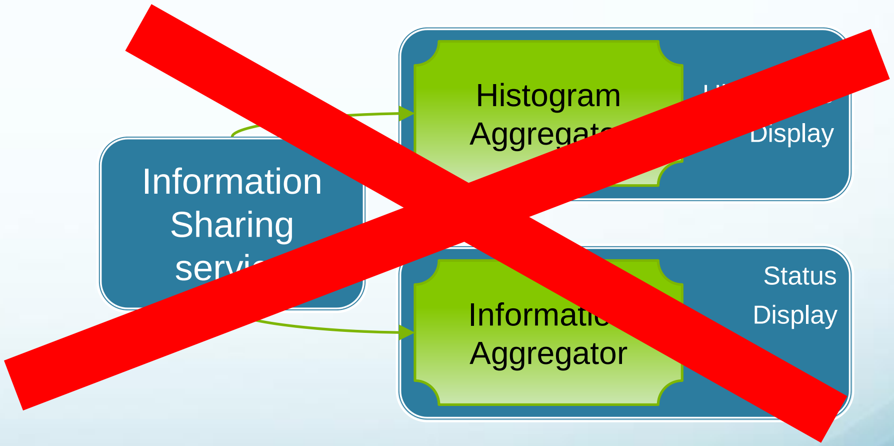
- Monitoring information produced by individual DAQ Applications has to be collected to provide high-level system status, e.g.:
  - Aggregating counters produced by individual computers of the Trigger Computer Farm one can access a complete state of the Trigger system
  - Aggregating histograms from all Trigger applications will give accumulated statistics for all processed events
- The Monitoring system shall provide a flexible way for aggregating homogenous information of arbitrary types



Information Aggregation: The Design

- Aggregation shall be done once in a single place:
  - A dedicated service shall be provided for that
  - Gathered information should be stored in the Information Sharing service
- Aggregation shall be flexible:
  - It shall support collection of arbitrary information
  - It shall support multiple gathering algorithms:
    - Sum or Average
    - Custom algorithms

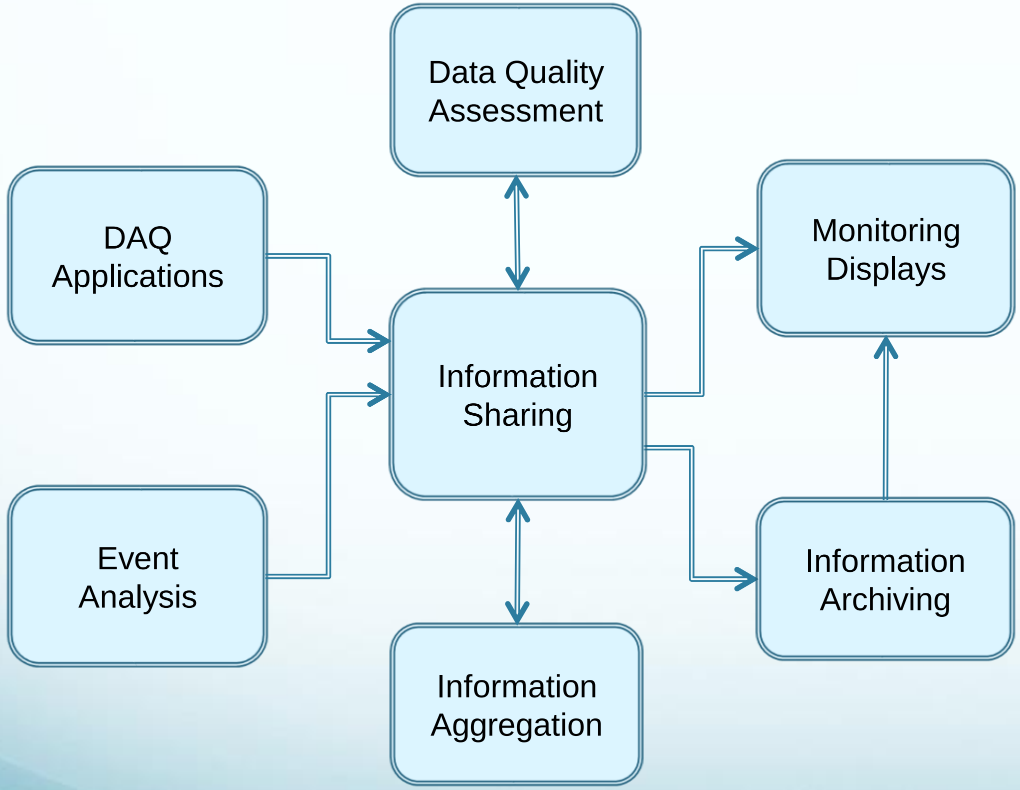




Archiving Monitoring Information

- Ideally all monitoring information shall be archived to a permanent storage:
  - Do post mortem analysis
  - Special attention shall be paid to WARNINGs
  - Investigating problems
- Conceptually Archiver is just a special type of the Information Receiver
- In practice the task is non-trivial due to the huge amount of information
- It might be feasible to have multiple Archivers for different information types:
  - Histograms, errors, operational status

The Distributed Scalable DAQ Monitoring System Architecture

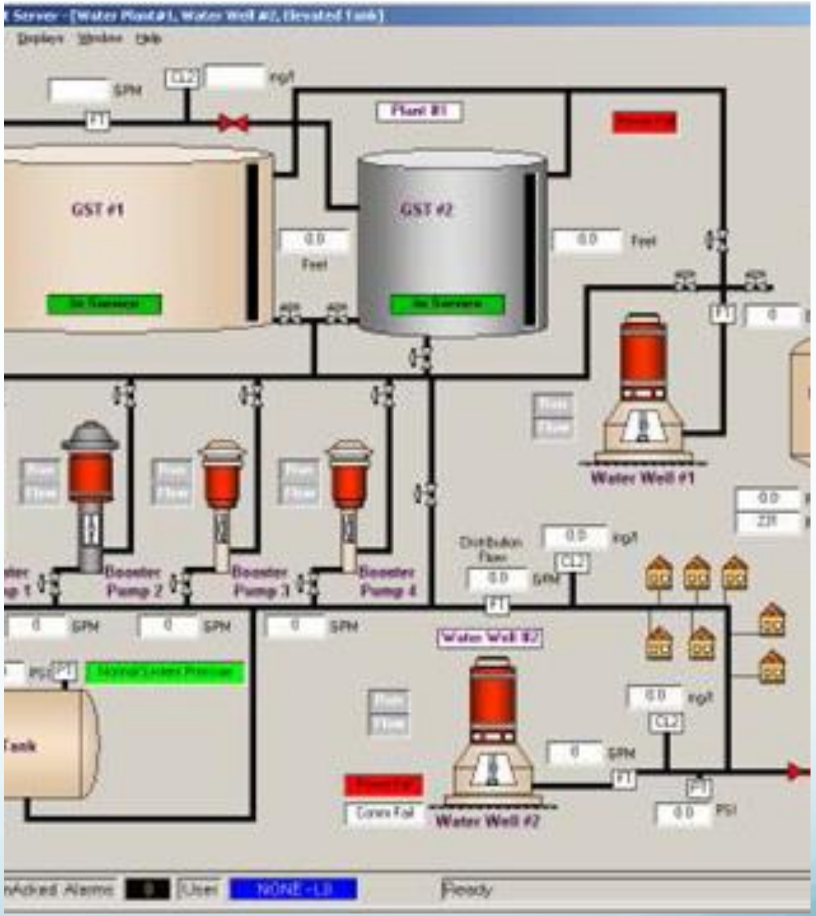


- The core of the Monitoring System is the Information Sharing service
- All other services are either Monitoring Providers or Receivers
- Some of them may do both at the same time

A Monitoring System Implementation

Commercial Solutions: SCADA systems

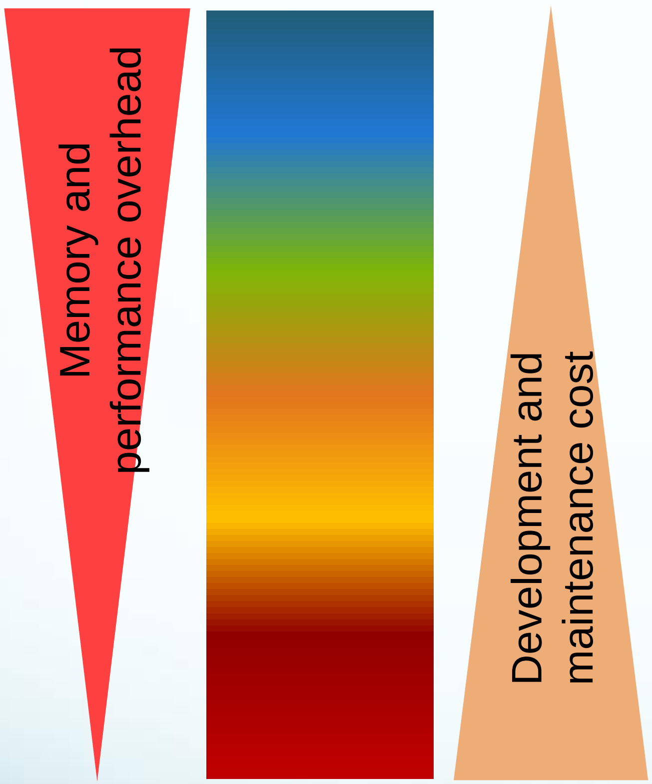
- Supervisory Control and Data Acquisition
  - It is primarily dedicated for control but does the monitoring as well
  - Modern implementations scale well with the number of controlled devices
- A SCADA system can be implemented using LabView:
  - Graphical programming language for the system design
  - Powerful and configurable graphical interface
- Is used mostly for HW control and monitoring
  - May not fit well to the DAQ specific monitoring, i.e. Data Quality



Custom solution: Technologies to choose

- Information Sharing:
  - Choose one of the existing Inter Process Communication solutions
- Information exchange format:
  - It may or may not depend on the chosen IPC technology
- Information archiving technology:
  - Consider volume and rate requirements carefully
- Information visualization technology:
  - Use the same GUIs for online and archived information

The spectrum of the IPC technologies



- Ice from ZeroC
- CORBA: TAO, omniORB, JacORB, ORBacus, …
- Messaging systems: Qpid, ActiveMQ, RabbitMQ, ...
- Libraries: Boost ASIO, ZeroMQ, ACE, …
- Socket API, TCP, HTTP, etc

- The choice depends on your requirements:
  - System size, Programming languages, available resources, implementation time scale, etc.

Data format for network transfer

- For HTTP communication Json is the natural format:
  - For example an information about SW process can look like
  - { “CPU”: 90, “Memory”: 4.3, … }
- Advantages of Json:
  - Simple, Human readable, self-contained
- Performance is the weak point:
  - Parsing Json takes significant amount of CPU
  - Transferring attributes names add noticeable overhead for the network bandwidth utilization
- Compact protocol buffer format can be considered as an alternative:
  - E.g. google/protobuf, binary Json, etc.

Data Archiving Technologies

- A choice strongly depends on the requirements of a particular experiment
- Large HEP experiments store O(1)TB of monitoring information per year
- Traditional (SQL) databases are not good for that
- Big Data approach is the new trend in this area
  - Hadoop, Teradata, Cassandra and many others

Visualization Technologies

GUI Frameworks/Libraries

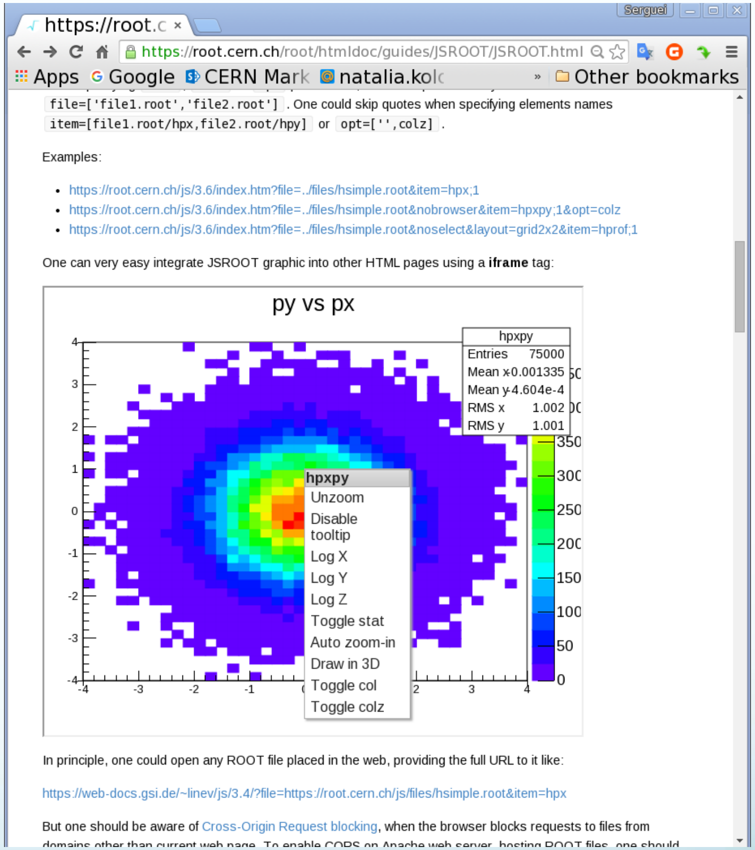
- Normally is bound to some specific programming languages:
  - Qt – C++, Python
  - Swing – Java
- Run-time libraries have to installed together with the custom GUI application
- Good performance

WEB Browsers

- Visualization is easily customizable (javascript, CSS, etc.)
- A lot of out-of-the-box graphical visualization libraries are available
- No additional software required on a client computer
- Available all over the globe
- May not give adequate performance for quasi real-time systems

Visualization Technologies: physics special

- Physicist needs histograms, which severely limits a spectrum of available visualization technologies:
- ROOT is C++ only … wait, It **was** C++ only
- ROOT 6 contains JavaScript library for histograms visualization in Web browsers:
  - JSROOT



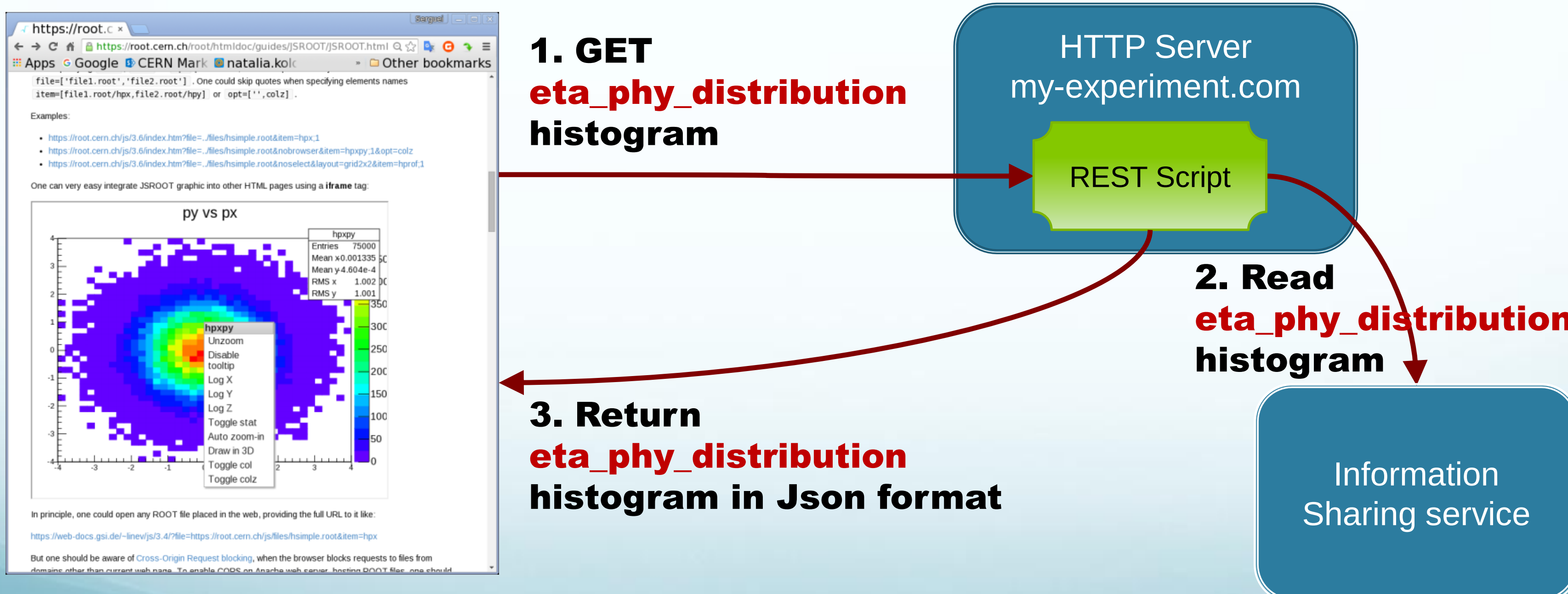
https://root.cern.ch/root/htmldoc/guides/JSROOT/JSROOT.html

Monitoring Information access via WEB

- Nowadays, if information is not present in the Web it virtually does not exist
  - That's especially true for HEP area where most of the experiments are built and operated by international collaborations
- Using REST is a simple way of adding WEB access to arbitrary monitoring data
- REST – **Re**presentational **S**tate **T**ransfer
  - It is *an architecture style*
  - Based on HTTP
  - Stateless
  - Client-server communication

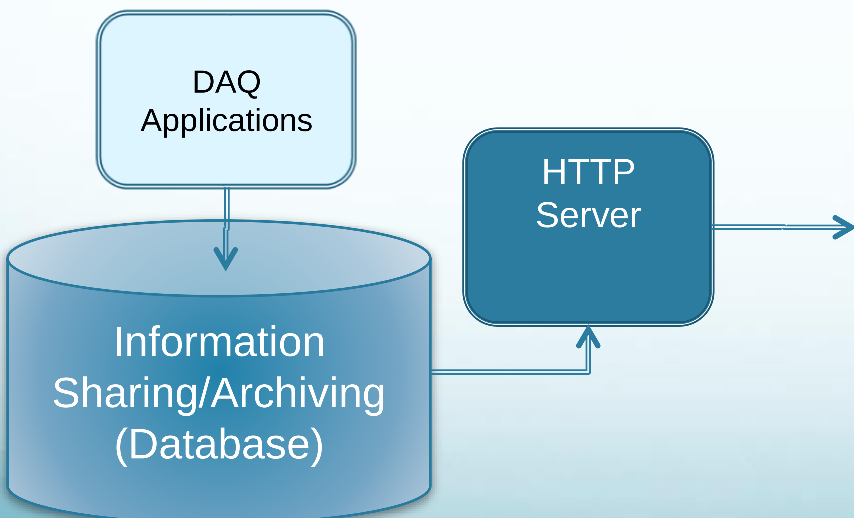
Every monitoring object is a WEB resource!

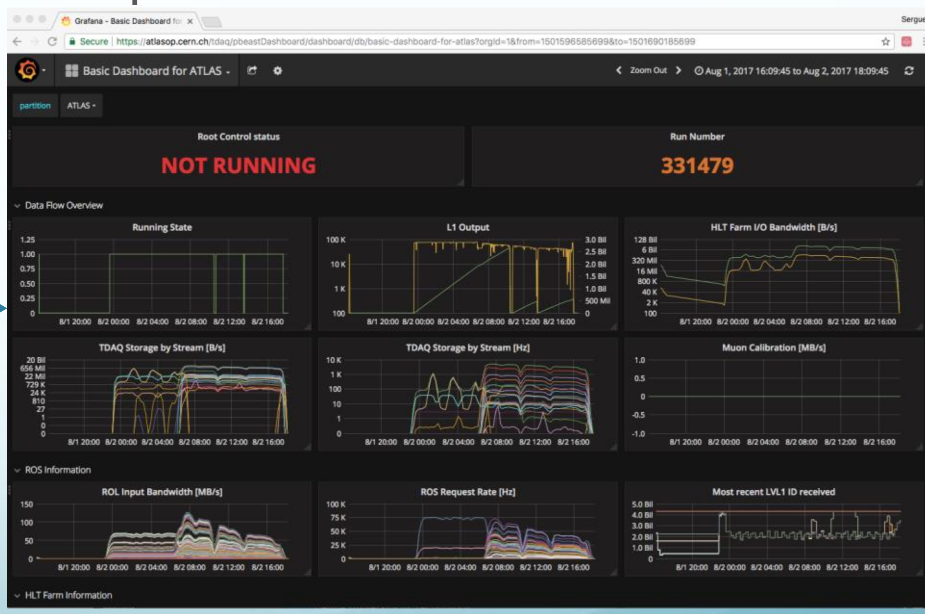
- Each information object is associated with a unique URL
- Use HTTP GET to read the value of the corresponding object:
  - **GET http://my-experiment.com/histograms/eta_phy_distribution**



Can we make it more simple?

- One interesting possibility is to join Information Sharing and Archiving into a single component:
  - ‘Old school’ but quite viable solution
- This solution follows a classical Web Design pattern:
  - Huge number of out-of-the-box implementations exist on the market





Final Remarks

Monitoring vs Control

- Monitoring is often considered as an ad hoc system which can be developed at leisure
  - Big mistake!
- Control won't fly without monitoring:
  - One can't claim to be controlling something without knowing the actual state of the controlled system

"It looks like you have everything under control."

Set up your Priorities Properly!

- A good DAQ system implementation shall start from having in place a good and complete Online Monitoring API :
  - Use standard Monitoring APIs as much as possible
  - Mediums implementations may come later
- These efforts will be rewarded:
  - The properly reported monitoring information will greatly simplify development and reduce the time for testing and debugging
  - This will help improving design and implementation of the Online Monitoring system itself by uncovering weak points, funding bugs, removing bottlenecks, etc.
  - In the end that reduces the time for DAQ system development and improves its quality!