Design and Implementation of a Monitoring System

Serguei Kolos
(University of California, Irvine)

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

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

The Monitoring Software Example

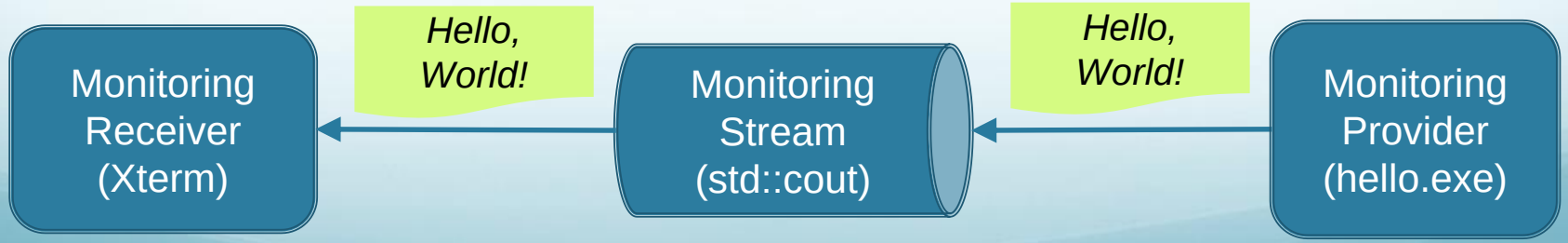
```
#include <iostream>

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

This is the monitoring message

This is the monitoring stream

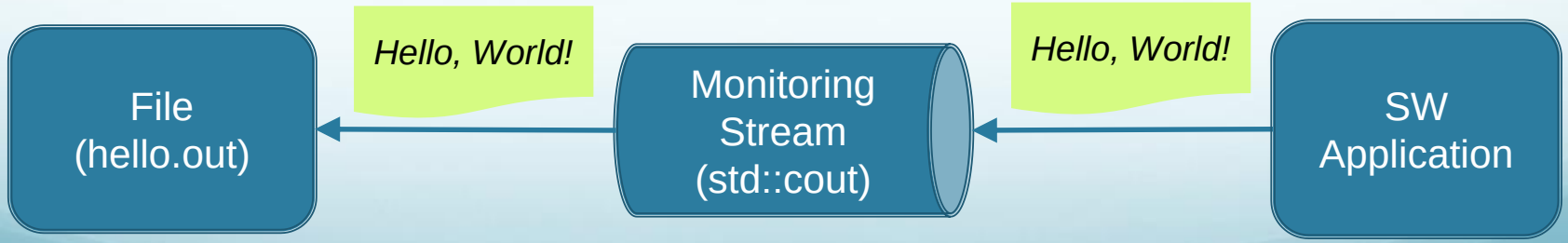
A simple Monitoring system architecture

- Monitoring Provider:
  - A SW applications which manifests its own state or the state of the controlled HW by sending some information to the Monitoring Stream
- Monitoring Receiver:
  - A SW application which receives the monitoring information for some purpose, e.g. displaying
- Monitoring Stream:
  - Provides transportation of the information from providers to receivers

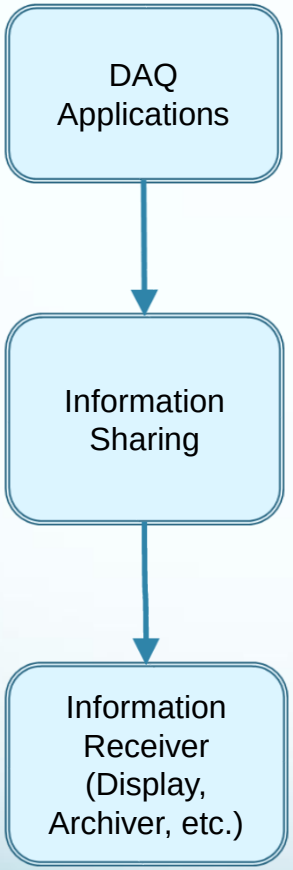


The benefits of the modular design

- In such an architecture all components are independent :
  - Providers have no assumptions about receivers
  - Receivers may do what ever they wants with the messages
- For example an Monitoring Information Archiving is just another Monitoring Receiver which puts the information to some permanent storage
  - > hello.exe > hello.out



The Monitoring System: The Basic Architecture

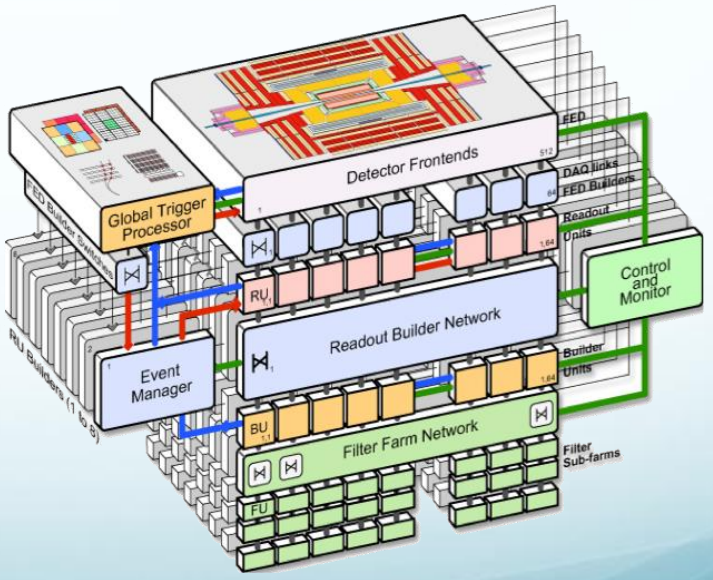


- The core of the Monitoring System is the Information Sharing service:
  - Provides implementation of the Monitoring Stream(s)
- Any DAQ application shall provide comprehensive information to the IS service about:
  - It's own state
  - The state of the controlled HW
- Receivers of the monitoring information shall be able to filter out a sub-set of the relevant information

Monitoring In the DAQ system

What is monitored in the 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 events recorded, trigger efficiencies, etc.
- System errors:
  - Any abnormal situation shall be reported using an appropriate severity
- Data Quality Monitoring:
  - Recorded data shall be constantly monitored for sensibility



Monitoring Information

- A simple 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
- Structured types are normally better:
  - A DAQ SW or HW element contains 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:
  - Each information object shall define its source:
    - Computer
    - Application ID
    - HW Module ID

- A 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

ERRORS

A special type of information

Errors classification

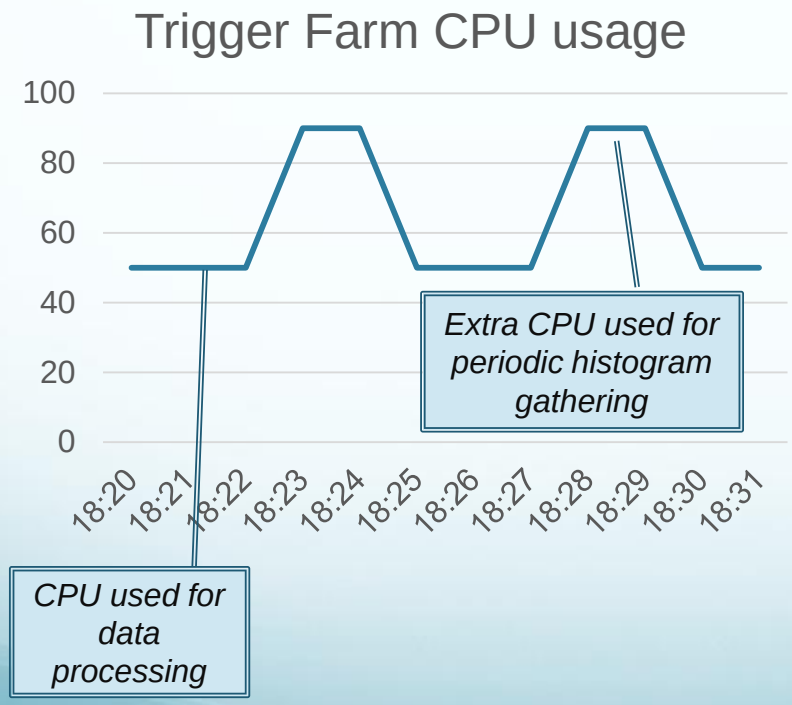
- For proper error handling it's important to establish common understanding of the error types:
  - WARNING – system operates correctly but dangerously close to a certain limit
  - ERROR – an abnormal situation is encountered but the work can be continued
  - FATAL – game is over, that's all folks, I'm dead
- 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.
- Errors should contain human readable text explaining the problem:
  - A shifter calling you in the middle of a night can just read it

Don’t neglect WARNINGs

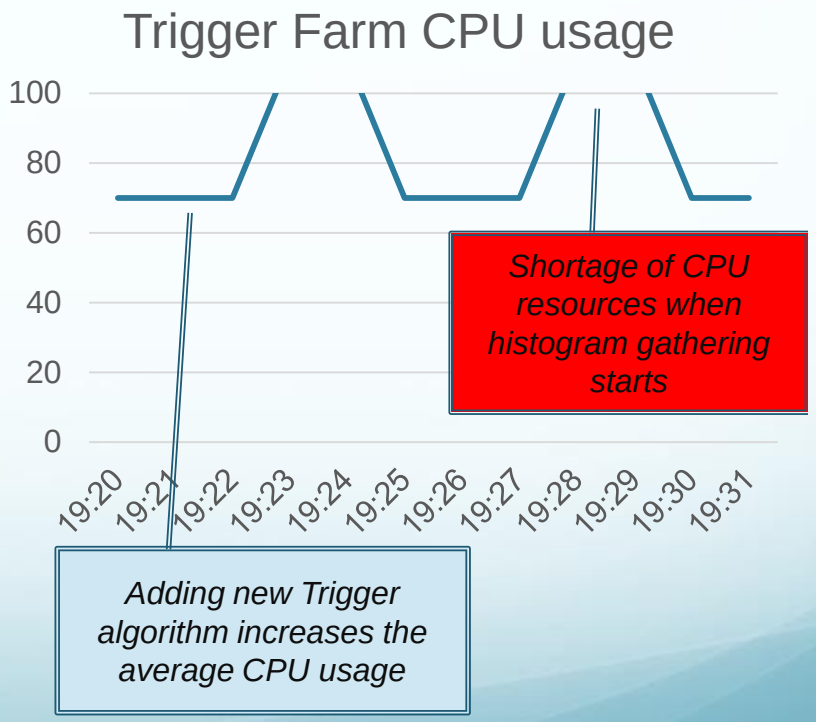
- Reporting an ERROR is straightforward:
  - When something goes wrong, one should shout out loud
- Reporting WARNINGs is much more cumbersome but is equally important:
  - When everything is good but close to the limit, shout loud as well
  - Requires some extra health-checking code to verify operational 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: real life example

Everything goes well so far



Backpressure causes the dead-time



The DAQ specialty: Data Quality Monitoring

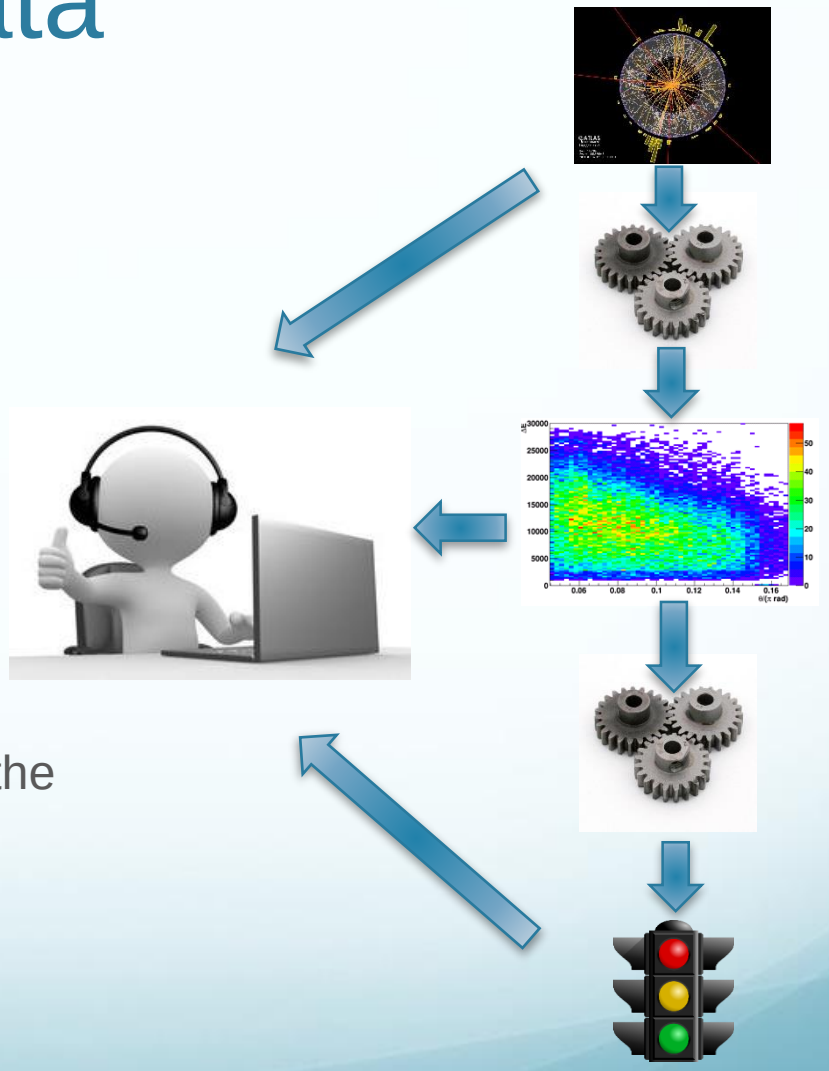
Data Quality Monitoring

- Watching out the behavior of the DAQ system itself is not sufficient:
  - The DAQ may be functioning perfectly but at the same time is taking meaningless data

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

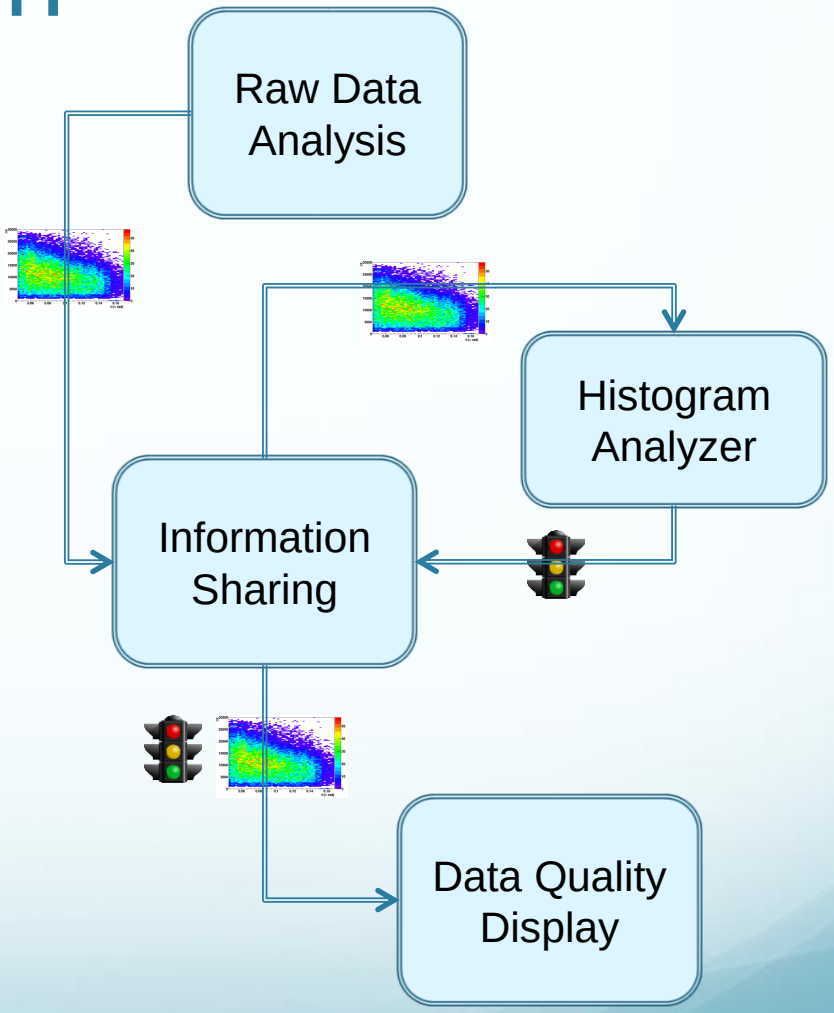
Data Quality analysis: The Flow of Data

- Simple analysis can be done by using graphical representation of a reconstructed event
- One can analyze histograms:
  - Produced by the trigger software
  - Produced by dedicated applications
- Histograms can be checked:
  - By eyes of an expert
  - By automatic algorithm producing the color code status



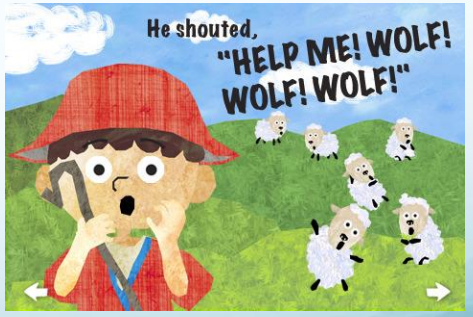
Data Quality Monitoring: The Design

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



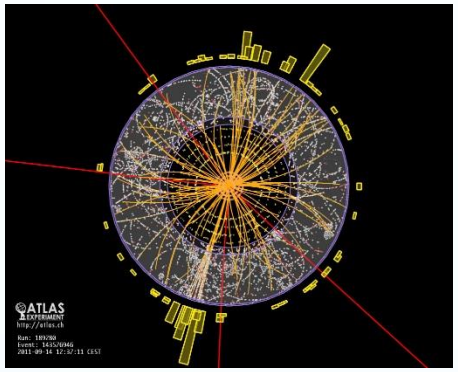
The Special Monitoring Outcome: Alarms

- A Monitoring system produces alarms when certain conditions are satisfied:
  - It's a way of getting immediate shifter attraction
- Visually an alarm is a color item which turns red
  - Green – good,
  - Orange – stay tuned,
  - Red – an ALARM, immediate intervention is required, can be accompanied by a siren sound
  - Don't forget about color blind people – an alarm icon shall have different shape and picture
- Do not overuse alarms!
  - Alarms shall be used for displaying critical errors only

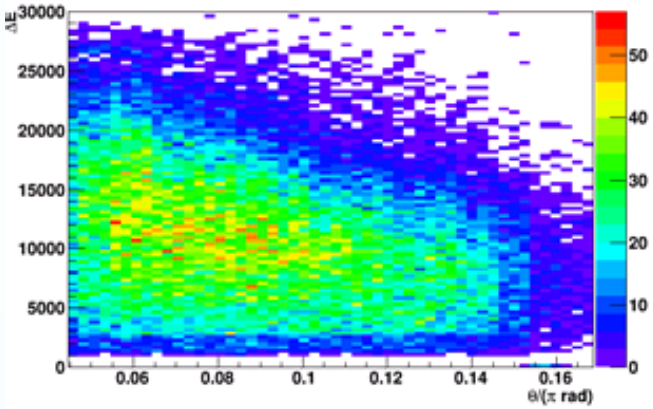


Visualizing Monitoring Information

Visualization: Information Types



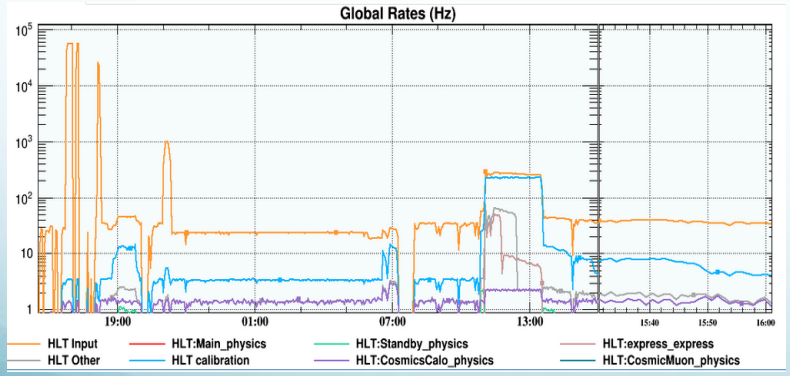
Raw Events



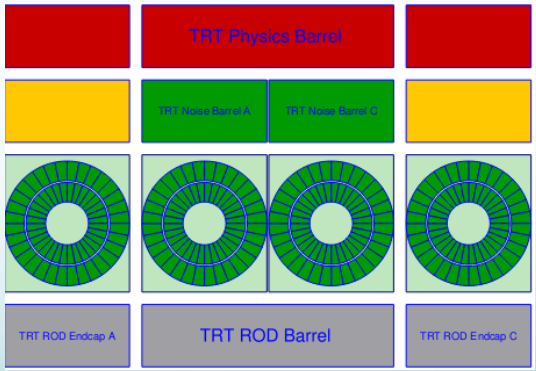
Histograms

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



Rates



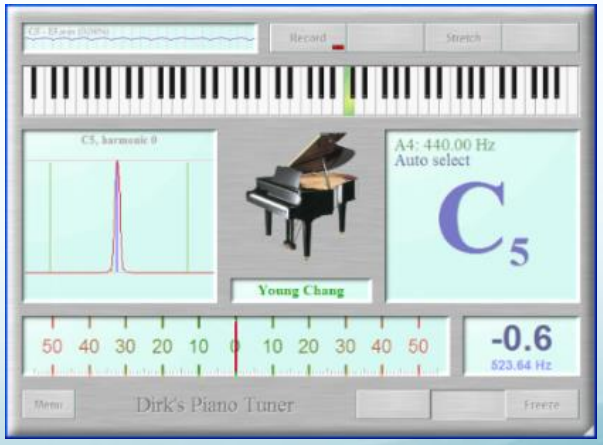
Data Quality status and alarms

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 system operation:
  - They can replace an expert by providing an unexperienced user with the missing capabilities
  - Spotting problems becomes trivial if they are clearly presented



Scaling up the Monitoring System

The HEP Reality

- Modern HEP detectors have millions channels
- DAQ system for such an experiment runs on several thousands computers:
  - ~10^3 computers
  - ~10^4 CPU Cores and SW applications
- How to scale up the Monitoring System?

Distributed Information Sharing service

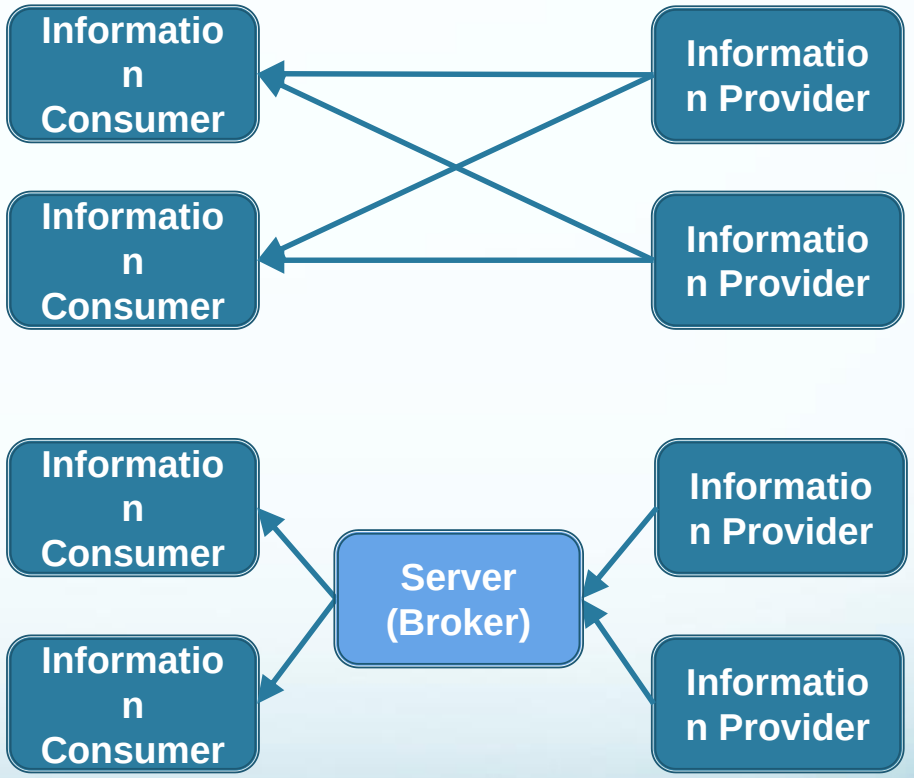
- A Distributed Information Sharing service is the key to the successful scaling of the Monitoring System
- The Service shall provide location independent API:
  - Information Providers and Consumers shall not know anything about distributed communications
  - They can be transparently move from one computer to another
- Depending on the DAQ system specific requirements one has to an choose appropriately two IS properties:
  - The information access type
  - The communication model

Push vs Pull Information Access

- Push Model, also know as Subscribe/Callback:
  - An Information Consumer subscribes for the relevant subset of the information to receive only what it needs
- Pull Model:
  - An Information Consumer 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

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

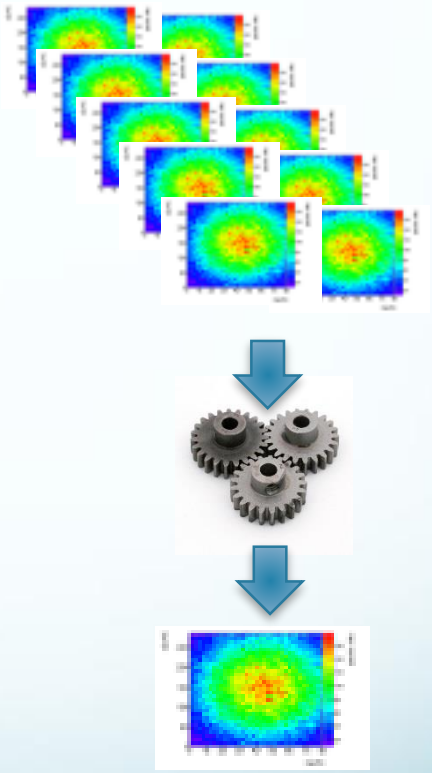
- Pro:
  - Has no single point of failure
  - Scales better
- Cons:
  - Requires more connections
  - Consumers may have a negative impact to the main activity of the Providers
  - More difficult to implement and maintain

Client-Server

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

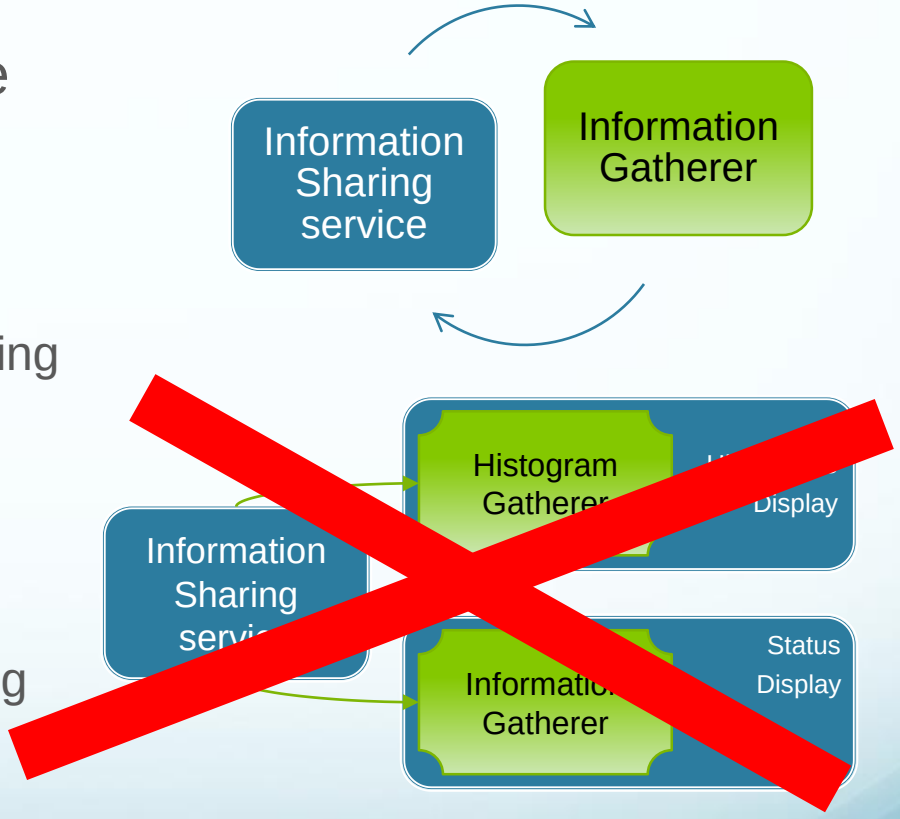
Information Gathering

- Monitoring information produced by individual DAQ Applications has to be collected to provide high-leve system status, e.g.:
  - Gathering resources used by individual computers of the Trigger Farm one can access the full load of the Trigger system
  - Gathering histograms from all Trigger applications will give accumulated statistics over all processed events
- A Monitoring system shall provide a flexible and powerful mechanism for Gathering homogenous information of arbitrary types



Information Gathering: The Design

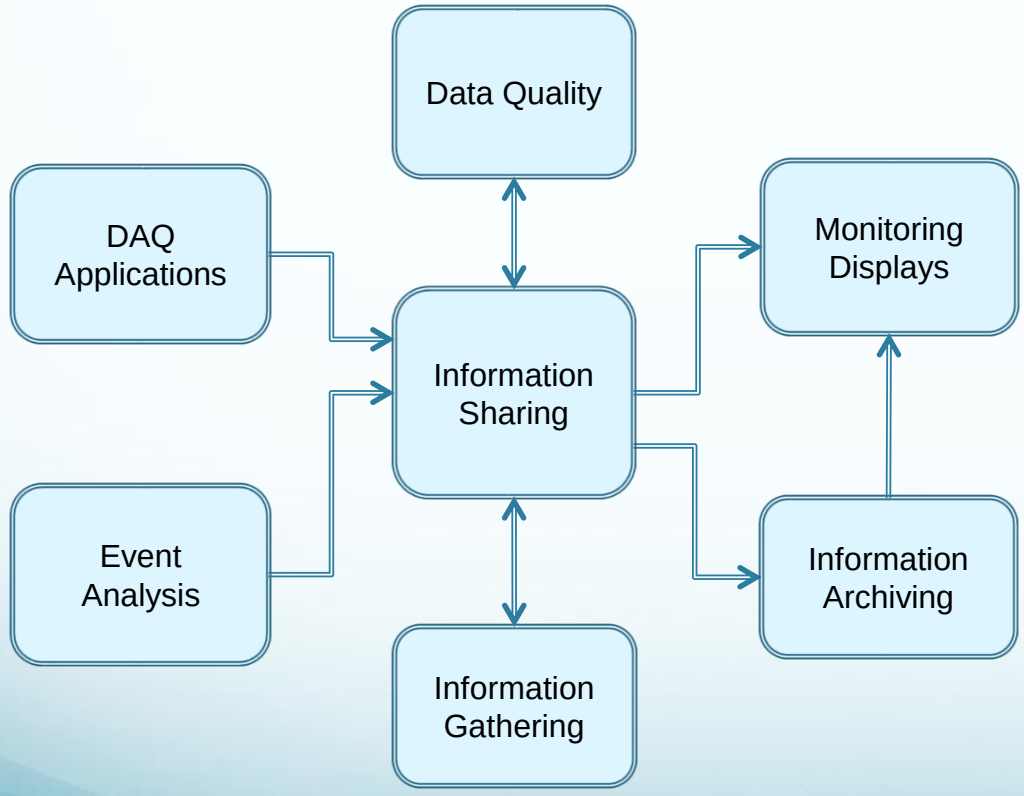
- Gathering shall be done once in a single place:
  - A dedicated service shall be provided for that
  - Gathered information should be put back to the Information Sharing service
- Gathering shall be flexible:
  - It shall support collection of arbitrary information
  - It shall support arbitrary gathering 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

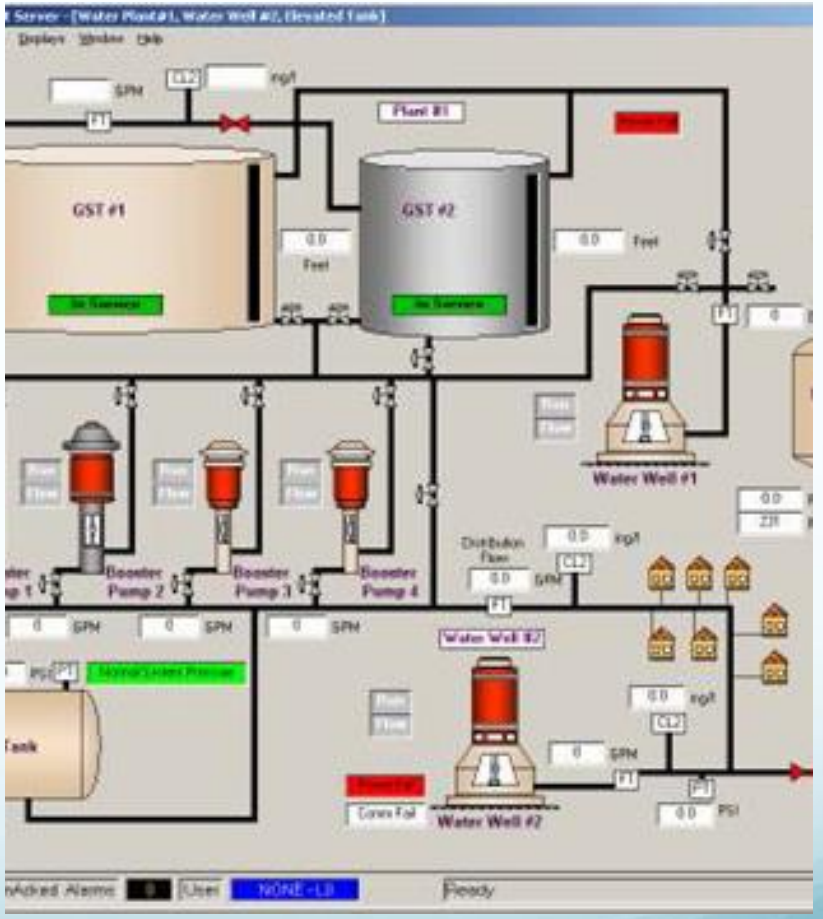


- The core of the Monitoring System is the Information Sharing service
- All other services are either Monitoring Providers or Receivers
- Some of them may do both at the same time

The Monitoring System Implementation

Commercial Solutions: SCADA systems

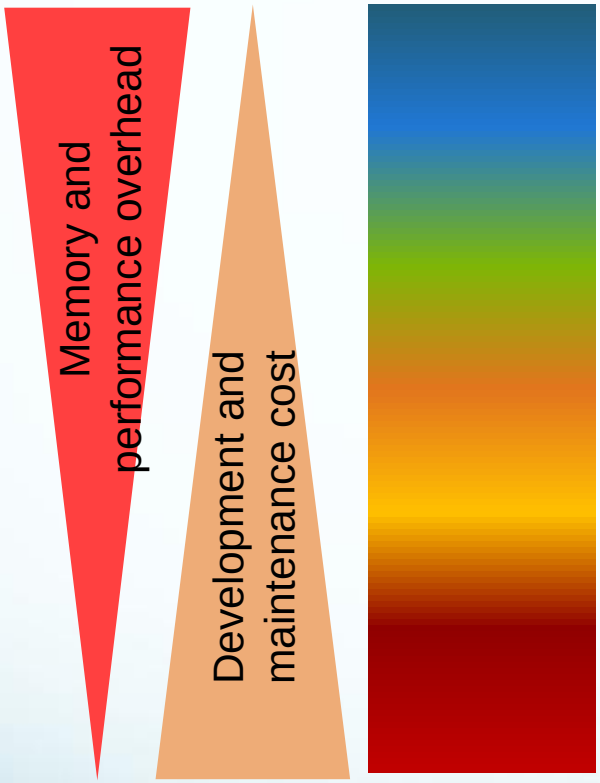
- Supervisory Control and Data Acquisition
  - It is primarily dedicated for control but does the monitoring as well
  - Modern implementations scale well with the number of controlled devices
- A SCADA system can be implemented using LabView:
  - Graphical programming language for the system design
  - 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 Inter Process Communication technology to be used:
    - Ideally the IS API shall be independent of the IPC technology
  - Decide what communication patterns to use:
    - Push, Pull or mixed, Client/Server vs Peer-to-Peer
- Data exchange format:
  - It may or may not depend on the chosen IPC technology
- Data archiving technology:
  - Consider volume and rate requirements carefully
- Visualization technology:
  - Use the same GUIs for online and archived information

The spectrum of communication technologies



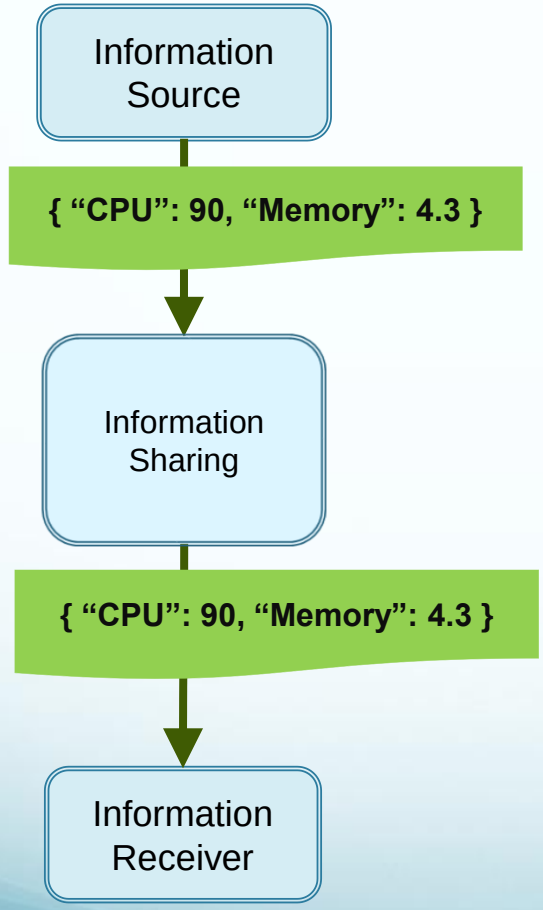
- Ice from ZeroC
- CORBA: TAO, omniORB, JacORB, ORBacus, …
- Messaging systems: Qpid, ActiveMQ, RabbitMQ, ...
- Libraries: Boost ASIO, ZeroMQ, ACE, …
- Socket API

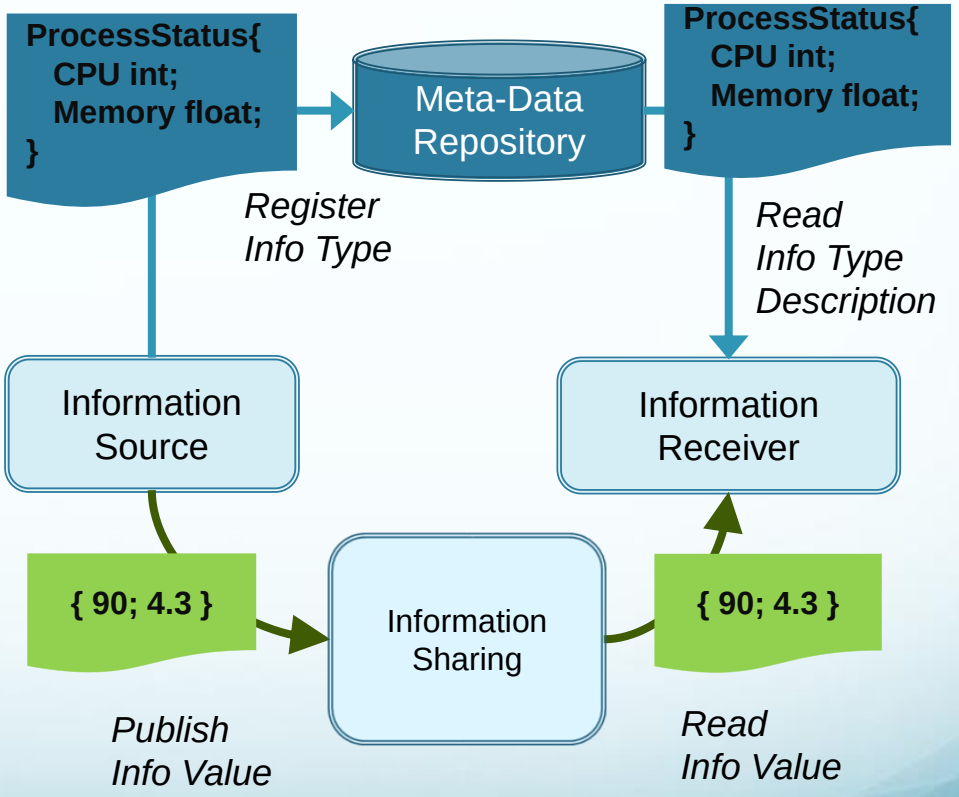
- The choice depends on your requirements:
  - System size, Programming languages, available resources, implementation time scale, etc.

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 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

Meta-Information: Complexity vs. Performance





Data Archiving Technologies

- A choice strongly depends on the requirements for the particular experiment
- Large HEP experiments store ~10TB 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

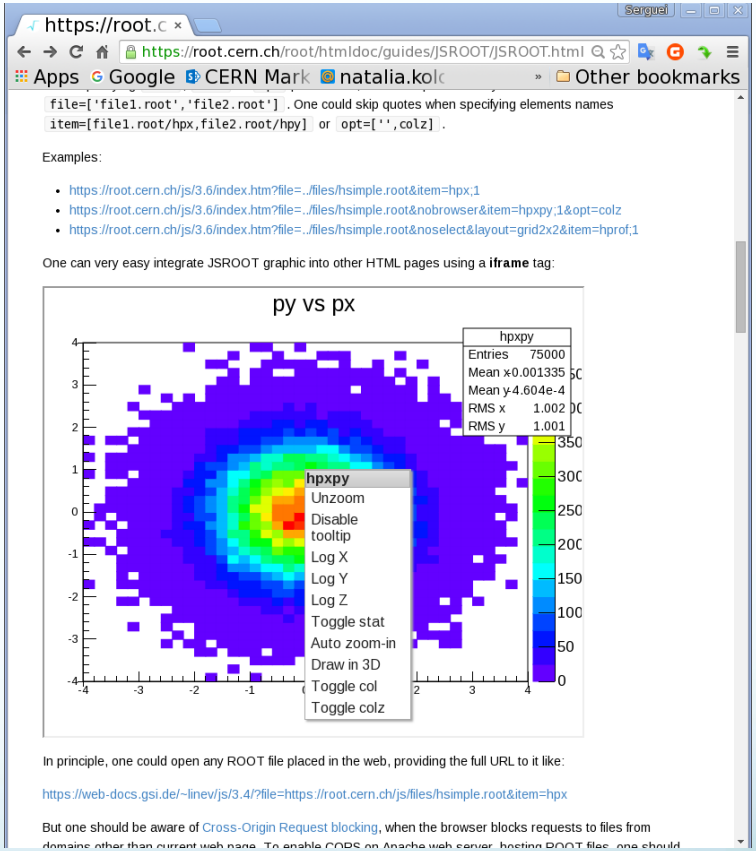
- Normally is bound to some specific programming languages:
  - Qt – C++, Python
  - Swing – Java
- Run-time libraries have to installed together with the custom GUI application
- Good performance

WEB Browsers

- Visualization is easily customizable (javascript, CSS, etc.)
- No additional software has to be installed
- Available all over the globe
- Does not scale for large amount of information

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



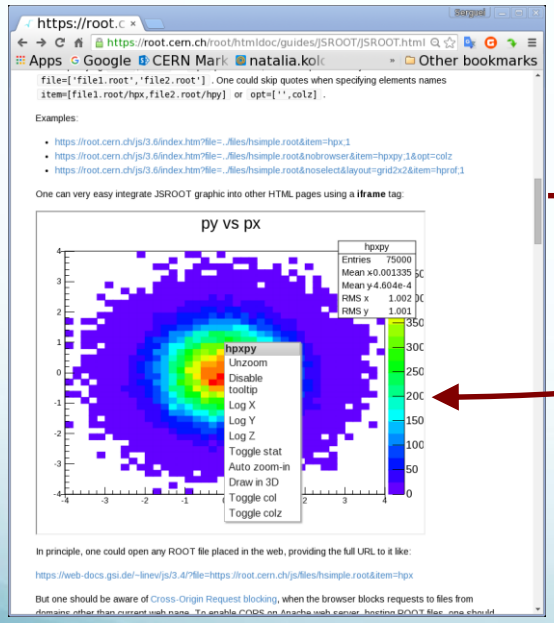
https://root.cern.ch/root/htmldoc/guides/JSROOT/JSROOT.html

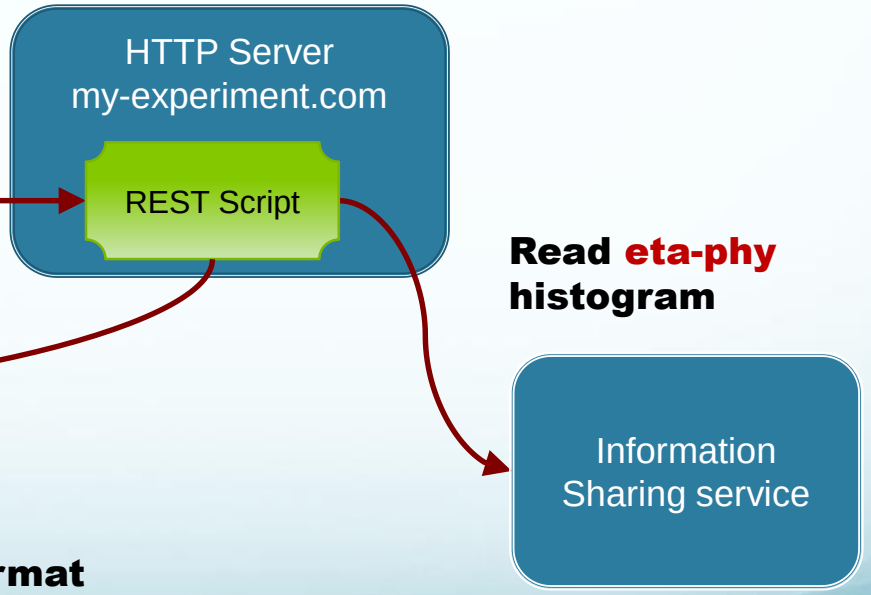
Monitoring Information access via WEB

- Nowadays the ultimate rule is - If information is not present in the Web is does not exist!
  - That's especially true for HEP where most of the experiments are built and operated by international collaborations
- Using REST is a simple way of adding WEB access to any 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 has a unique URL
- Use HTTP GET to read the corresponding object:
  - **GET http://my-experiment.com/histograms/eta-phy**

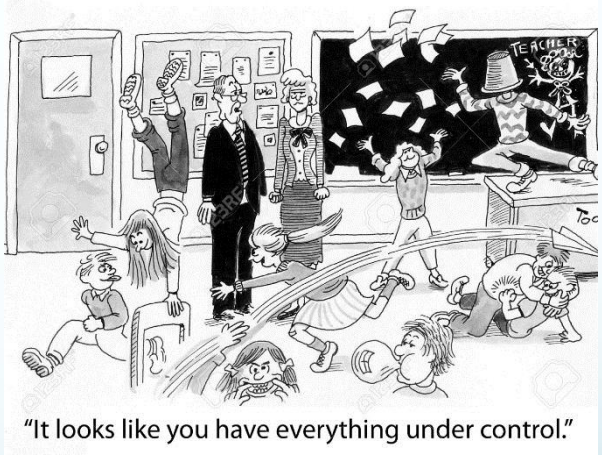




Final Remarks

Monitoring vs Control

- Monitoring is often considered as an ad hoc system which can be developed at the very end at ones leisure
  - Nothing could be more wrong than this assumption!
- Control won't work without monitoring:
  - One can't claim to be controlling something without knowing if commands were executed correctly or if not what was the last error
- There is one interesting consequence of that:
  - If one designs a DAQ system the Monitoring shall be the first component to be implemented



Set your Priorities!

- DAQ system implementation shall start from providing the Online Monitoring services!
  - The Online Monitoring interfaces have to be carefully designed in advance
  - The Online Monitoring system (or at least a working prototype) shall be in place and ready to be used
  - All other services shall use all that from the very beginning
- There are multiple benefits:
  - Already at the development & testing phase monitoring information will be reported to the right place
  - This simplifies debugging and testing of the DAQ system
  - This will help improving the design and implementation of the Online Monitoring itself by uncovering weak points, funding bugs, suggesting improvements, etc.