Experience with online systems
A (very) persona! view

Matthias Richter

Dep. of Physics, University of Oslo

CWG13 Meeting
April 04 2014

Running an online system for physics analysis

Some technical challenges

- data transport
- cluster infrastructure
- file systems and package distribution
- access of large data sets
- process orchestration

Some algorithmic challenges

- Fast algorithms
- Scalable processing instances
- Data model
- ...

Some collaborative challenges

- Combination of computer scientists (online) and physicists (offline)
- ...

Design considerations

Some questions to be answered at the beginning

- Target data rate
- Data/event dropping policy
- Physics objectives
- Developer community
- Hardware platforms: where is the project running in development, test, and production mode?
- What are the use cases? Collect all! use cases together with the detector groups
- What is the system supposed to do? Everything? NO! Focus on concrete use cases within an open architecture
- Does the design of the trigger system support online data filtering at all?

Technical aspects

... that's of minor interest for CWG13

Software

- Split into small "topological" packages
- Interfaces and data structures with a conservative evolution policy
- Control of dependencies
  - define module dependencies right from the beginning
  - revise dependencies regularly
  - use the linker functionality
- Sand box for developers
  - stand alone suite for development
  - preferably small dependencies
- Documentation and tutorials
- Unit tests are part of the development
- Automatic build and verification

Development strategy

- Break-up into manageable pieces from the beginning
- Strict release schedule
- Data structures take a key role in the processing
  - Evolution of basic data structures needs to be controlled by a central body
  - Restricted evolution advisable
  - Avoid streaming of data structures or design the streaming carefully
- Develop algorithms with the online-needs in mind
- Respect deadlines, find alternative solutions before reaching the deadline

"Slow" algorithms

- At each point there is a certain bandwidth required to avoid congestion, depending on the data size at this point
  $\Rightarrow$ Mapping of requirements in the processing topology
- Early identify bottlenecks, often they are neither related to hardware, network bandwidth, nor CPU power
- Data bound algorithms often don't scale in parallizations
- Avoid conversion of data structures
- Break data into manageable entities
  - distributed memory blocks with appropriate indexing allow to provide only the necessary parameters to the algorithm
  - trade-off between data duplication and access of big data "lumps"

Operational aspects

- Operation in the experiment environment needs simple recovery procedures
- Failure tracing required to quickly identify the cause of a problem
- There are only two possibilities: either expert presence 24/7 or the system is fully automatic
- Rely on standard tools whenever possible

Commissioning policies

Online system has direct impact to the data taking

$\Rightarrow$ Failures are irreparable

You better follow the simple rule:

Never change a running system

$\Rightarrow$ do your homework well in advance

Must-have-tools

- Error handling scheme
  - different categories to be defined
  - automatic handling depending on severity
- Central logging facility
  - central collection
  - masking
  - tools for browsing
- System simulation
- Visualization of data flow and performance hot spots
- Configuration manager

Making use of manpower

Naturally a **complex system**

$\Rightarrow$ only **a handful of experts** can understand the full system

Bottom-to-top approach:

- Design and develop from the fundamental building block upwards
- Steering and control via scripts with full feature control
- Early development of high-level access tools, to allow non-experts to join and contribute

A simple playground for developers:

- Sandbox to plug-in algorithms
- Data access identical to production system including limitations
- Flexible interface: Identical code runs in sandbox or production system
- Parallel or paralyzed: inbuilt feasibility check for parallel processing

Take-away messages for CWG13

⇒ The first things to do:

- Carefully design the data model involving all use cases and user groups
  ⇒ STICK TO IT
- Motivate developers to follow the framework by providing an appropriate sandbox
- Define and provide all fundamental framework concepts: logging, error definition and propagation, parallization tools