Dataflow Monitoring

Nicoletta Garelli

ALICE, ATLAS, CMS & LHCb joint workshop on DAQ@LHC

12-14 March 2013
Château de Bossey
What Dataflow Monitoring Means

• Monitoring in real time the flow of data to ensure optimal data taking
  – from detector readout to permanent storage
  – trigger & DAQ quantities (counters, data rate, buffer occupancy, etc.)

• Avoid dead-time
  – and eventually allow to fix problems

• Each experiment uses its own jargon to indicate the same thing
Requirements

**Basics**

- Access any relevant information in real time to follow data taking
- Online aggregation & data correlation
- Online problem detection: dead-time, data losses, etc.

**Added later**

- Archive: access historical data for diagnostics, statistics, post-mortem
- Use monitoring data to trigger alarms and/or automatic actions to recover problems
Evolution

• Users: shifters & experts

• LHC operations ...at the beginning
  – scattered information and rudimentary tools
  – shifters: intense monitoring activity
  – experts: high presence in control room + ringing on-call phone

• LHC operations ...routine
  – coherent information and optimized tools
  – automate as much as possible to reduce shifter’s tasks
    • see Luca’s talk of this morning
  – move from custom GUI to ubiquitous web based tools
    • let’s do all with a smartphone
THE 4 ARCHITECTURES
• Each experiment developed **4 different DAQ systems** using different technologies

• Variety reflected in dataflow monitoring middleware
  
  – LHCb & ALICE: Distributed Information Management (**DIM**)  
    • client/server paradigm, light weight
  
  – ATLAS: Information Service (**IS**)  
    • custom library on top of CORBA  
    • client-server communication model where information is stored in memory by so called IS servers
  
  – CMS: **Web Service**  
    • Cross-DAQ (**XDAQ**) framework
• ~300 processes on ~300 machines
• 100k dataflow information published every 5 s → ~3 GB/h
ALICE Dataflow Monitoring Architecture

- **DAQ processes**
  - Based on **DIM/SMI**
    - SMI: framework for designing and implementing distributed control systems developed by DELPHI
  - **MySQL**:
    - store system configuration
    - LDC&GDC write run info
    - Archive.
  - **"Logbook"**
    - much more than what you think
    - **PHP**
  - **2 monitoring applications**
    - Tcl/Tk
  - **Logbook** as visualization

- **DIM / SMI**
  - **Status Display**
  - **Backpressure Monitor**
  - **Logbook**
ALICE Visualization

**LDC status display**
- LDC name: alonglde
- Current Trigger rate: 5049.500
- Average Trigger rate: 4399.143
- Number of sub-events: 798887314
- Sub-event rate: 5049
- Sub-events recorded: 798887316
- Sub-event recorded rate: 5050
- Bytes injected: 5798311981688.54
- Byte injected rate: 4.003 GB/s
- Bytes recorded: 5798311981688.54
- Byte recorded rate: 4.003 GB/s
- Nb. events w/o HLT decision: 0
- Mem allocation failed: 0
- Average time bmAllocate: 0.000000

**GDC status display**
- GDC name: alonggdc
- Number of sub-events: 0
- Sub-event rate: 0
- Events recorded: 0
- Event recorded rate: 0
- Bytes recorded: 0
- Byte recorded rate: 0 B/s
- File count: 0
- Nb. incomplete events: 0
ATLAS DAQ

- O(20k) processes on ~2k machines
- 1M dataflow information published every 5-10 s \(\rightarrow\) ~4 GB/h
**ATLAS Dataflow Monitoring Architecture**

- **Mirror IS**: real-time copy from IS (ATCN) to mirror counterpart in CERN GPN

  - **Information Service (IS)**: ~100 servers
    - **DAQ processes**
      - **Web IS (Python CGI)**
        - **Web Server (Apache)**
          - **Web Browser**
      - **Monitoring GUI applications**
        - **Web IS (Python CGI)**
          - **Web Server (Apache)**
            - **Web Browser**

  - **WEB IS**: IS gives information access on demand via HTTP protocol
    - python wrapper accepts HTTP requests & sends back dynamically formed XML text (value of IS obj pointed by given URL)

  - **Archive**: None.
    - information stored & accessed for ~2 month in RDD files each ~30 s via network monitoring system
Shifter’s Tools in 2012

- **DAQ Panel**: tool portal for shifters
- **DFSummary**
  - dynamically constructed web page which computes & displays most important dataflow parameters (~200 variables)
  - ~30 s update rate
- **Busy Panel**: Qt application for monitoring dead-time
- **Shifter Assistant**
  - see Luca’s talk of this morning
• $O(20k)$ processes on $\sim2k$ machines
• $O(600k)$ dataflow information published every 1-5 s $\rightarrow \sim8\text{ GB/h}$
**XDAQ Monitoring & Alarming Service**

- Fully scalable distributed monitoring & alarming system
- Service-oriented architecture organized in 3-tier structured collection of communicating components:
  - **Sensor**: report monitoring data
  - **Eventing**: scalable publisher-subscriber service orchestrated by a load balancer application (**Broker**)
  - **Collector**: build relational tables
  - **Live Access Service**: presentation of raw data (**Web Service**)

**Archive**: automatic persistency of collected tables in ORACLE according to configuration
- Subset of information stored: ~30 GB/y
Monitoring as a Service

• XDAQ as a Service (XaaS): common infrastructure for both central DAQ & sub-detectors

• interoperable services providing standard functionalities for use in XDAQ environment

• All processes organized into searchable groups known as zones

• zone defines scope of a distributed XDAQ application

• Each zone has its own monitoring data types (flashlists)
CMS Visualization

- LabView DAQMon
LHCb DAQ

- O(40 k) processes on 2k machines
- 4M dataflow information published every 5s \( \rightarrow \sim 11.5 \text{ GB/h} \)
LHCb Dataflow Monitoring Architecture

Archive:
- PVSS ORACLE DB
- O(200k) values every 5s
LHCb Visualization

PVSS GUI

VT100 graphics
detailed UI
“YES, it does the job”

“... BUT ...”

- 4 different solutions for the same problem ...
- sharing experience and maybe even future common solutions?

→ Luciano’s talk on Thursday