Control and monitoring of on-line trigger algorithms using a SCADA system

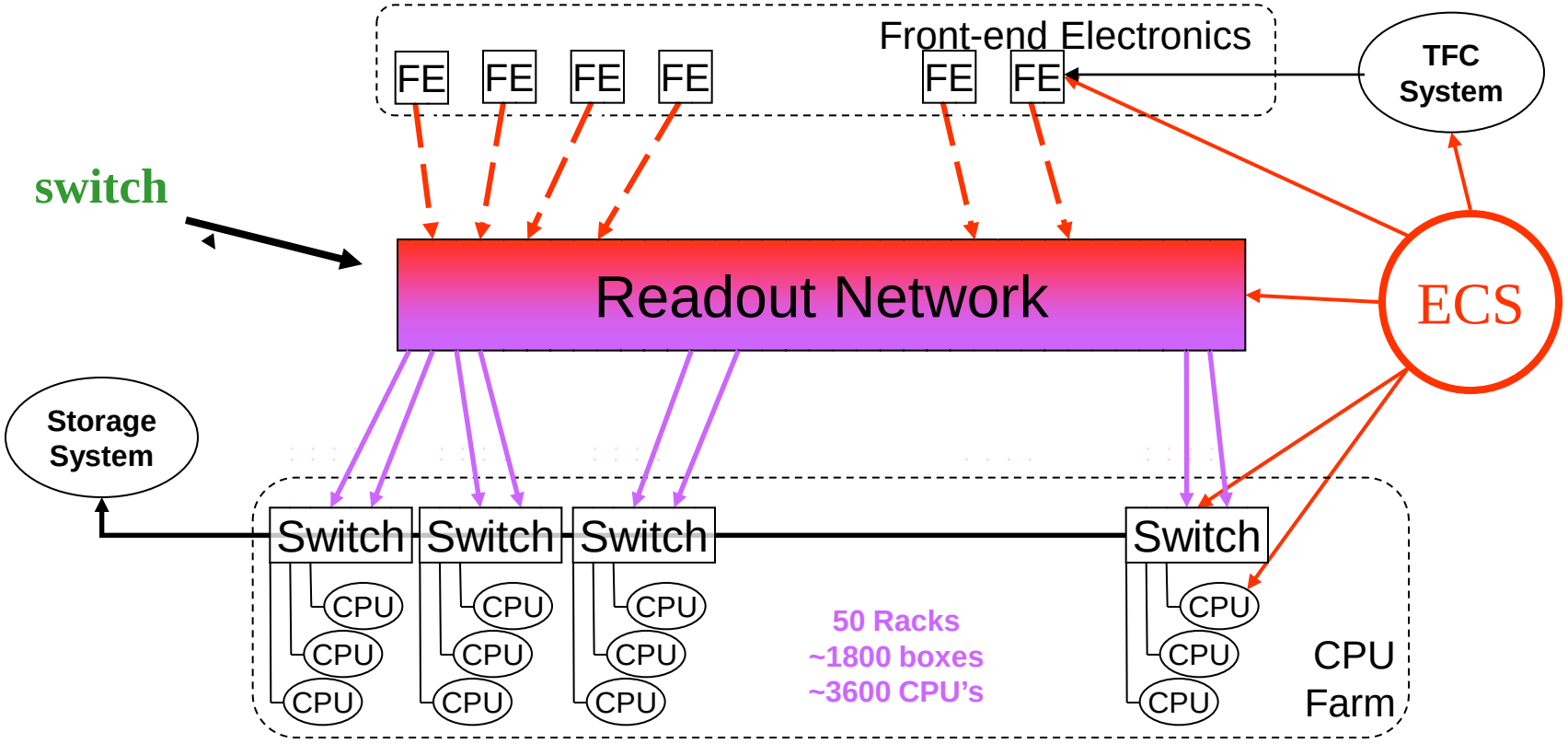
Eric van Herwijnen

Wednesday 15th February 2006

Contents

- Configuration of the LHCb DAQ
- The Problem
- A solution
- Gaucho architecture
- PVSS backend
- PVSS performance
- Root
- Summary

Configuration of the LHCb DAQ



The problem

- Control and monitor trigger (Gaudi) processes on Event Filter Farm
- Send monitoring data (counters, rates, histograms, status, error messages) to ECS
- Sum monitoring information
- Tag summed information and make available for further analysis

A solution

- SCADA=Supervisory Control And Data Acquisition
- LHCb's Experiment Control System uses PVSS
- PVSS for displaying counters, rates
- Root for displaying/manipulating histograms
- Histograms and counters accessed (e.g. via DIM)
- Automatic addition, display summed results

Gaucho (c++)

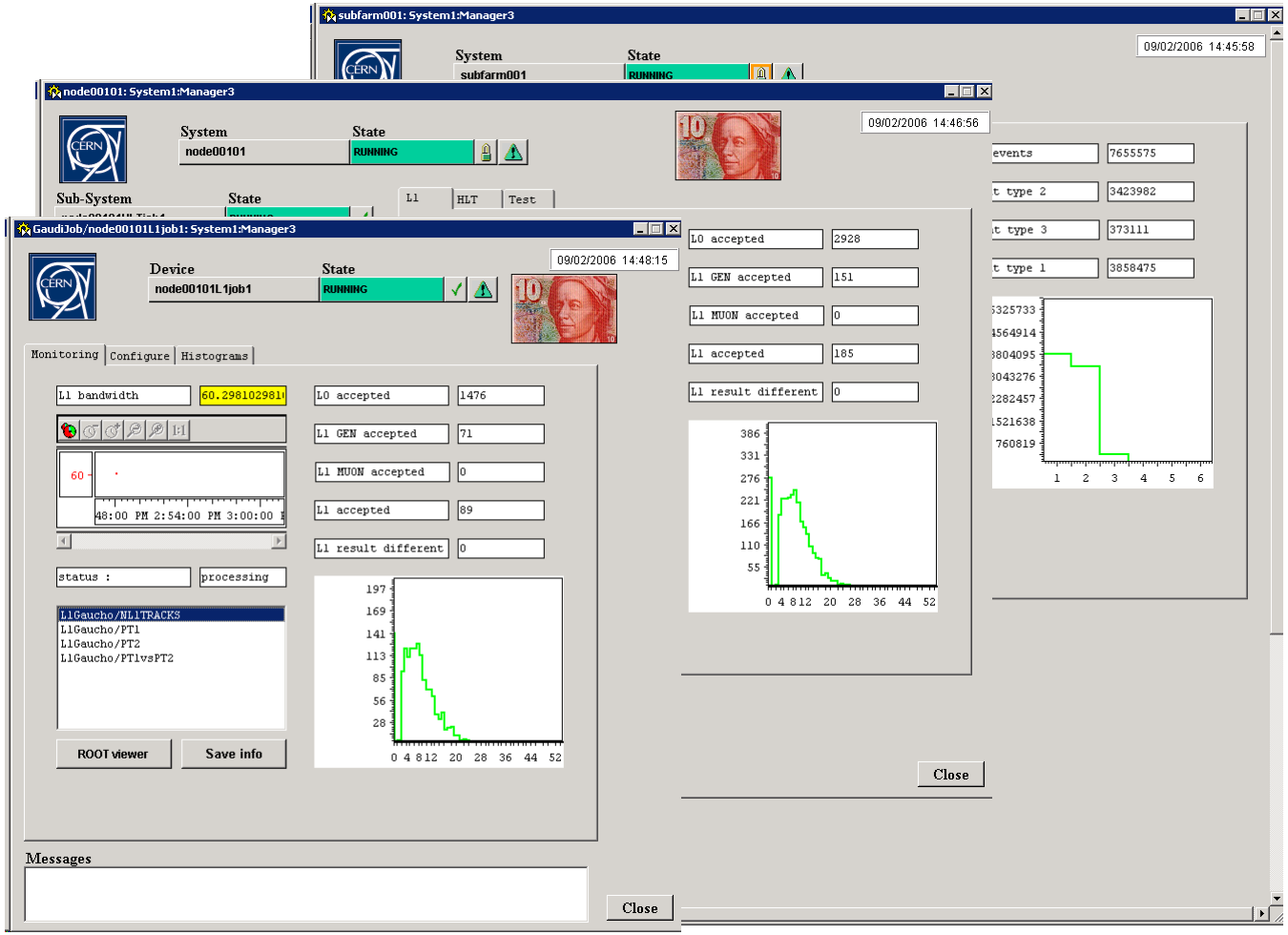
- GAUdi Component for Helping Online
- MonitorSvc to publish objects
- OnlineMessageSvc publishes messages
- A status variable publishes the state of the job (ready, running, stopped)
- Used by current LHCb trigger algorithm authors
- Information only sent when subscribed to, no performance penalty
- Current MonitorSvc/OnlineMessageSvc implemented using DIM, but this could be something else

PVSS backend

- DIM clients subscribe to counters and histograms
- Data viewed per job, summed per node, summed per subfarm
- Summed histograms are published by PVSS via DIM (PVSS acts as DIM server)

PVSS backend

- **Tree tier panel hierarchy: per subfarm, per node, per job**



Size of monitoring data

Item	Test algorithm	Typical Trigger algorithm
# strings	1 status, 5 comments	1 status, 3 comments
bytes for strings	228	188
# counters	4 ints, 1 long	3 ints
bytes for counters	20	12
# histograms (nbins)	4 1D (5,80,60,60), 1 2D (100)	24 1D (4x10,1X11,1X20,2x40,4x150,12x80), 2 2D (200)
bytes for histograms	1220	10044
Total #kb	1.5	10.5

Dataflow into PVSS

- Total for 40 nodes, 4 Trigger jobs, 2 Test job/node: 1.6 Mb every 20 secs (4 kbytes for counters, rest histograms)
- Well within capacity of network (12 Mb/s=100 Mbit/s)

PVSS performance

- Tests over 40 nodes (240 jobs)
- PVSS on Windows Xeon 3GHz CPU, 2 GB RAM
- Counters: implemented via “datapoint functions”
- Updated and summed every 20 secs
  - 3-12 % CPU usage, 700 Mb memory
- Histograms: updated and summed sequentially, once every 4 minutes
  - 5-55 % CPU usage, 700 Mb memory
- Memory use increases with time if update frequency increases
- PVSS can not display 2D histograms

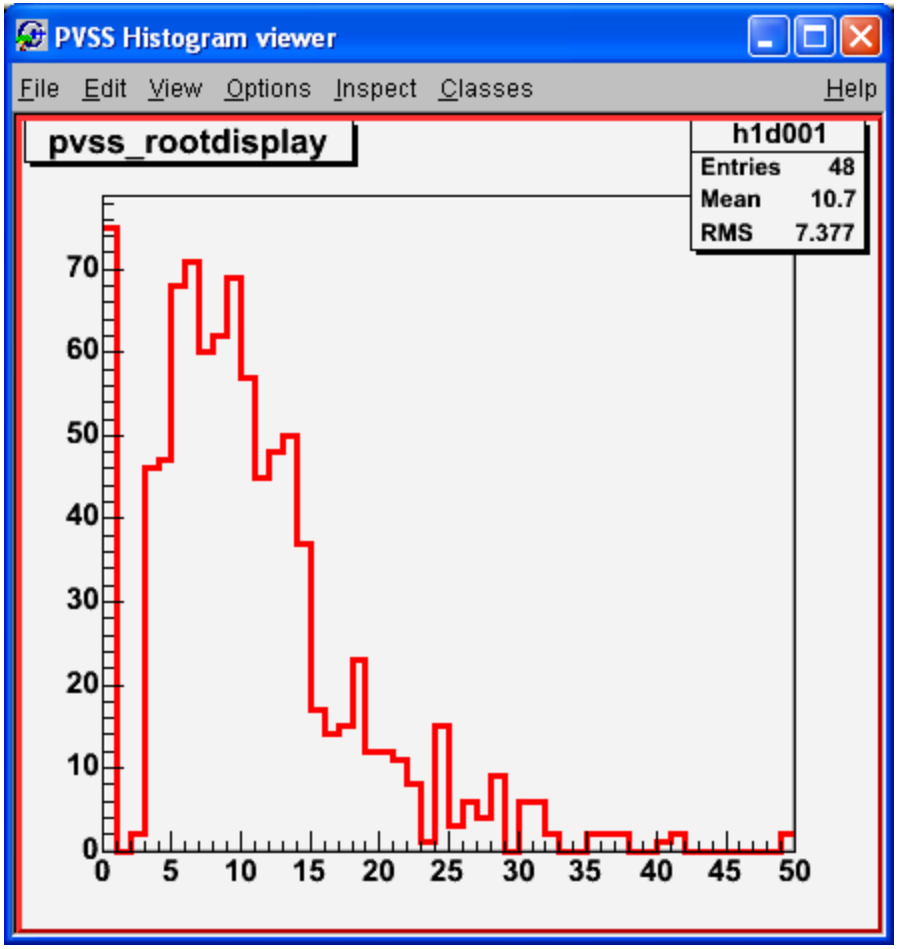
Root

- **PVSS is good for displaying counters; need a more scalable solution for manipulating histograms**
- **We plan to create stand alone "histogram adders"**
  - **finds out from PVSS which histograms to add (configuration)**
  - **subscribes directly to Gaudi jobs**
  - **publishes added histograms to clients (viewer, analysis programs)**
- **A stand alone histogram viewer will allow selection of histograms and display**
  - **Possibly a mixture of PVSS and Root**
  - **(Next release of PVSS uses QT)**

Current root viewer

- A root program implements a Dim client to display 1D and 2D histograms
- Viewer needs to be closed when viewing different histograms
- Data can be saved on demand or at regular intervals in text files (counters) or root files (histograms)

Root Viewer


PVSS Histogram viewer
File Edit View Options Inspect Classes Help
pvss_rootdisplay
h1d001
Entries 48
Mean 10.7
RMS 7.377
70
60
50
40
30
20
10
0
0 5 10 15 20 25 30 35 40 45 50

Summary

- Can use PVSS to monitor counters
- For a more scalable system, move manipulation of histograms outside of PVSS
- Adders connect directly to Gaudi jobs, controlled by PVSS, send results for further analysis and feed the viewer
- A separate histogram viewer is planned in Root/PVSS which will allow easy viewing/saving of histograms
- Build a histogram database to store details of histograms for quality analysis