OHP: An Online Histogram Presenter for the ATLAS experiment

A. Dotti¹, P. Adragna², D. Cimino³, M. Della Pietra⁴, R. Ferrari⁵, G. Gaudio⁵, C. Roda¹, D. Salvatore⁶, W. Vandelli⁵, P.F. Zema⁷

¹ Università and INFN Pisa
² Queen Mary, University of London
³ Università di Pisa
⁴ Università and INFN Napoli
⁵ Università and INFN Pavia
⁶ Università and INFN Cosenza
⁷ Università and INFN Cosenza and CERN
Outlook

- Introduction
- User Requirements
- Architecture
- Implementation
- Features
- Tests results and current applications
- Conclusions
Introduction

Sw components provided by ATLAS Monitoring Working Group (inside ATLAS TDAQ/HLT sub-system)

- OHP role is to present histograms produced by the ATLAS Monitoring Components in a user-friendly graphical application
- It is the evolution of various GUIs developed within the sub-detectors communities
- It unifies the approach to histogram visualization in ATLAS online environment
- For more details on ATLAS Monitoring System see W. Vandelli's presentation: “Strategies and Tools for the ATLAS On-line Monitoring”
Minimal Requirements

• The presenter must:
  – Be integrated in ATLAS TDAQ system
  – Be able to operate in two distinct modes:
    • Shifter (or tabbed) mode: display a pre-configured set of histograms
    • Expert (or browser) mode: allow to browse the collection of produced histograms
  – Automatically display histograms when they are available
  – Allow the shifters to interact with the displayed histograms: zooming, fitting, changing drawing styles, ...
  – Manage reference histograms
Desirable Requirements

- The presenter shall:
  - Minimize the network traffic: expected $O(100k)$ histograms
  - Be independent of the histogram producer: expected many different kinds of producers developed independently
  - Allow to interact with the histogram producers (to change scales, to reset histogram content, to turn on/off histogram production,...)
  - Allow to set graphic visualization attributes (log scales, draw format)
Introduction

Requirements

Architecture

Implementation

Features

Tests

Conclusions

Browser (or expert) mode:

histograms organized in folders

Actions:

- Save histograms in a ROOT file for offline analysis
- Print window content
- Pause/Resume application

Status bar
Tabbed (or shifter) mode: Pre-configured set of significant histograms
OHP and ATLAS Monitoring System

- Histograms are produced by dedicated applications and published on the Online Histogramming Service (OHS) accessible from any point in the system.
- OHP is the histogram visualization component of the ATLAS (highly distributed) monitoring system.
- OHP is a client of OHS and not of the histogram producers.

Data Flow Application → Event Monitoring Service → Monitoring Task → Monitoring Application → OHS → OHP

Schema from W. Vandelli
The interaction with OHS

• OHS provides a notification subscription mechanism:
  – The OHP subscribes to the desired histograms:
    • specifying the provider and histogram names
      (regular expressions are supported).
      Ex.: ProviderNumber1/TotalEnergy_histo
      Ex.: ProviderNumber2/.*
  – OHS notify OHP whenever an histogram is published or updated
  – The OHP will retrieve the histograms from the OHS and draw them in the GUI
Introduction

Requirements

Architecture

Implementation

Features

Tests

Conclusions

OHP and OHS may run on different machines connected via network
OHP

Core: provides access to histograms from GUI and controls the other components.
Histogram Receiver and Cache subsystems: Are responsible for receiving the notifications from OHS and retrieving histograms from OHS on request.
Filtering subsystem: its role is to filter the notifications that are related to the active (on focus) canvas.
How OHP works: an example

- Provider P1 publishes (updates) histogram H1 in OHS
- OHS sends a notification to OHP through the related Histogram Receiver
- OHS sends also a time-stamp Tn of the publication time (to be used in the cache)
How OHP works: an example

• The notification P1/H1 (with the time-stamp Tn) is stored in a cache (each Histogram Receiver owns a Cache)
• The notification P1/H1 is also sent to the Filtering Subsystem
How OHP works: an example

The Filtering subsystem checks if P1/H1 should be visualized in the active window.
- If not the notification is discarded, otherwise it is propagated to the GUI.
- When the active window changes the filtering subsystem changes its list of patterns.
How OHP works: an example

- The GUI asks, through the Core, the histogram object associated to P1/H1
- The histogram P1/H1 is received from OHS, stored in the Cache together with the time-stamp Tr and sent back to the GUI that draws it in the screen
How OHP works: an example

- Request from GUI also changing the active window (no notification from OHS involved)
- Compare the last notification time $T_n$ and retrieval time $T_r$
- Retrieve histogram from OHS only if needed
Implementation

- OHP has been developed in C++ trying to optimize the CPU load
- ROOT as the underlying technology for histogramming allows the use of all ROOT functionalities to interact with histograms (zooming, fitting)
- Qt is the used framework for the GUI implementation
Features: Configuration

- OHP can be fully configured through an ASCII file or a GUI panel
- In the configuration it is possible to specify:
  - Subscriptions
  - Tabs and content
  - Drawing options for histograms
  - Reference Histograms

Configuration GUI: the panel to set new tabs and their content
### Features: Sending Commands

- **Histograms Producers** can receive Commands from OHP to modify the histogram production mechanism.
- **Predefined commands:**
  - Reset: force a clean up of the histogram
  - Rebin: redefine axes
  - Update: force publishing on OHS of histograms
- **Custom command:** producer dependent command, the user can fill in the command content.
Features: Reference Histograms

• User can associate a reference histogram to an histogram in OHS that is displayed in the same window

• Reference histograms can be provided, locally, in a ROOT file or can be retrieved from the OHS

An Example of histogram with the reference histogram superimposed (in red). Graphic attributes (colours, lines, error bars) can be specified in the configuration
Test results and current applications

- A prototype version of OHP (with minimal functionalities) has been used during the test on beam of a slice of the ATLAS detector in 2004 and is currently being used for the commissioning of various ATLAS subsystems.

- Each OHP component has been successfully tested in stand-alone mode.

- We are now making a detailed test of the entire application in a realistic environment.

- We will start a new developing phase to optimize the code and to add new functionalities.
Conclusions

• OHP is an interactive application developed to present histograms produced by ATLAS Online Monitoring System

• The software life cycle of the application is at the moment in the release phase

• OHP is part of ATLAS TDAQ system

• OHP:
  - is fully configurable (different views of the produced histograms, drawing options)
  - can communicate with the histogram producers sending commands
  - optimizes network traffic
  - manages reference histograms

• OHP will be used in the ATLAS commissioning starting from March 2006 and we will also start the design phase of new functionalities
Backup material
Detailed schema of ATLAS Monitoring System

- Event Monitoring Service
- GNAM
  - Detector Specific Plug-in
- Athena
  - Detector Specific Athena Algorithm
  - Athena Monitoring
- Online Histogramming Service
- ROD
- ROS
- Event Builder
- OHP
- Gatherer
- Analysis Framework
- Monitoring Data Storage
- Event Displays

Schema from S.Kolos
The Role of the Core

- The core is the “control center” of OHP
- It is organized like a FSM
- It is responsible of the behaviour of the components providing a “mapping” between the States of the components and the States of OHP (CONFIGURED/RUNNING/PAUSED)
- The design pattern Singleton has been used to guarantee the possibility of concurrent access to resources (see next slide: plugins)
Future Upgrades

- **Help SubSystem**: provides http based help interface
- **Plugin SubSystem**: provides possibility to develop “user code” in dynamic libraries to extend OHP functionalities (modify graphic aspect or make automatic checks on histograms). Core code must be re-entrant
Architecture

OHP and OHS may run on different machines connected via network.