Advanced Visualization System for Monitoring the ATLAS TDAQ Network in Real-Time

S. Batraneanu, D. Campora, B. Martin,
D. Savu, S. Stancu, L. Leahu

Outline

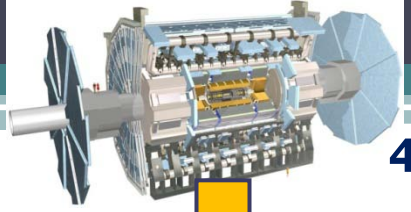
Introduction
- ATLAS trigger and data acquisition(TDAQ) system
- Monitoring the ATLAS TDAQ networks

Advanced visualization system for the TDAQ networks
- Design
  - Motivation, requirements and challenges
  - Hierarchical 3D model and its layout rules
- Implementation
  - Client architecture
  - Performance optimizations
  - Real-time update
  - Interaction mechanisms

Conclusions and future work

ATLAS Trigger and Data Acquisition System(TDAQ)

- Rejection factor:10^6
- Outstanding accuracy and efficiency
- 3 networks
  - DataCollection(Level-2)
  - BackEnd(Level-3)
  - Control
  - 6 chassis routers, 200 edge switches
  - 7000 interfaces, 2000 nodes
- Demanding performance requirements



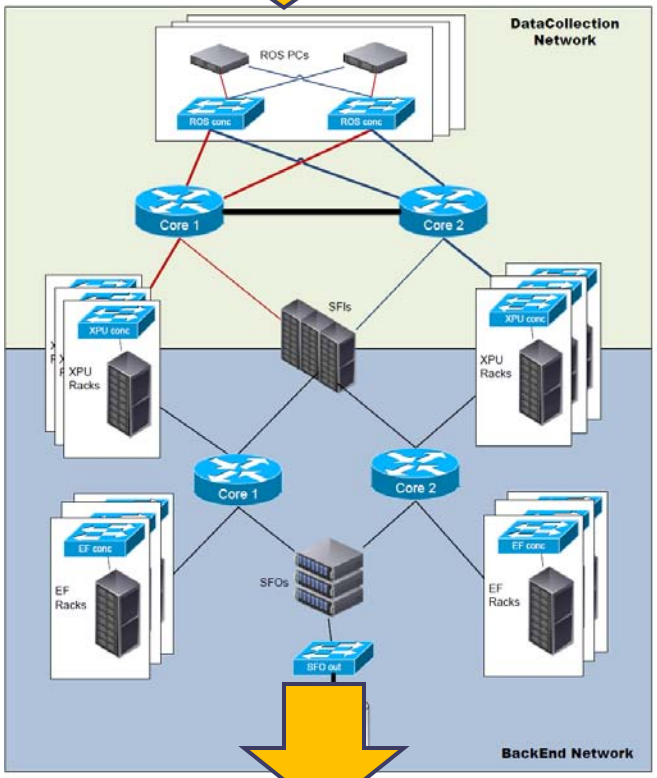
40 MHz

Level-1

100 kHz

Level-2

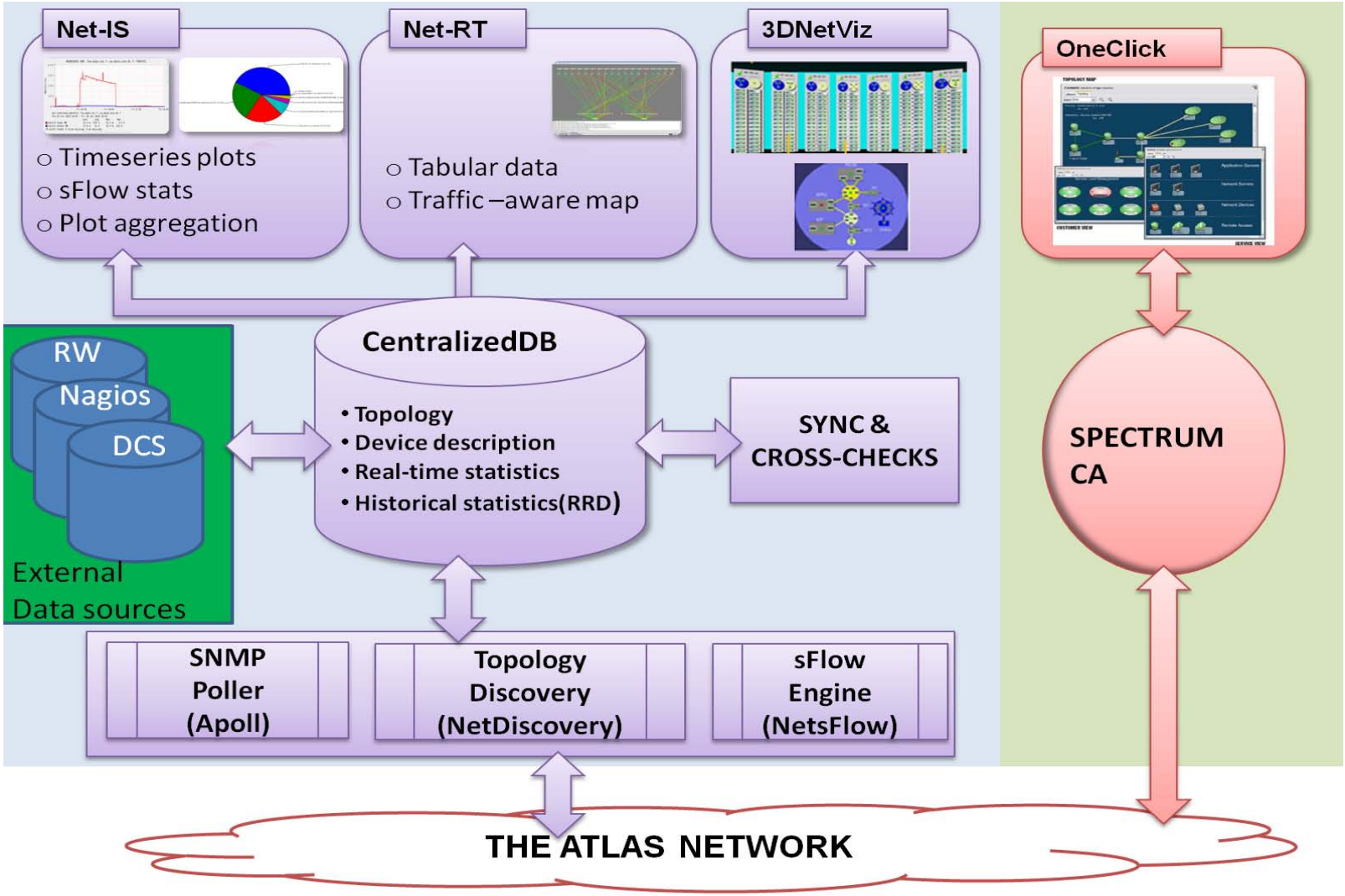
Level-3



Mass storage

200-300 Hz

Monitoring software framework



Advanced visualization system -requirements and challenges-

An **efficient** visualization system should :

- Be intuitive
- Follow the system's architecture and data flow
- Display the different types of monitoring data in real-time
- Offer the right level of detail
- Provide clear indications regarding the problem

Main implementation challenges

- Large scale system and overlapping networks
- Large variable space
- Real time update (30 seconds)
- Operation on multiple OSes : Windows, Linux

2D vs. 3D Visualization

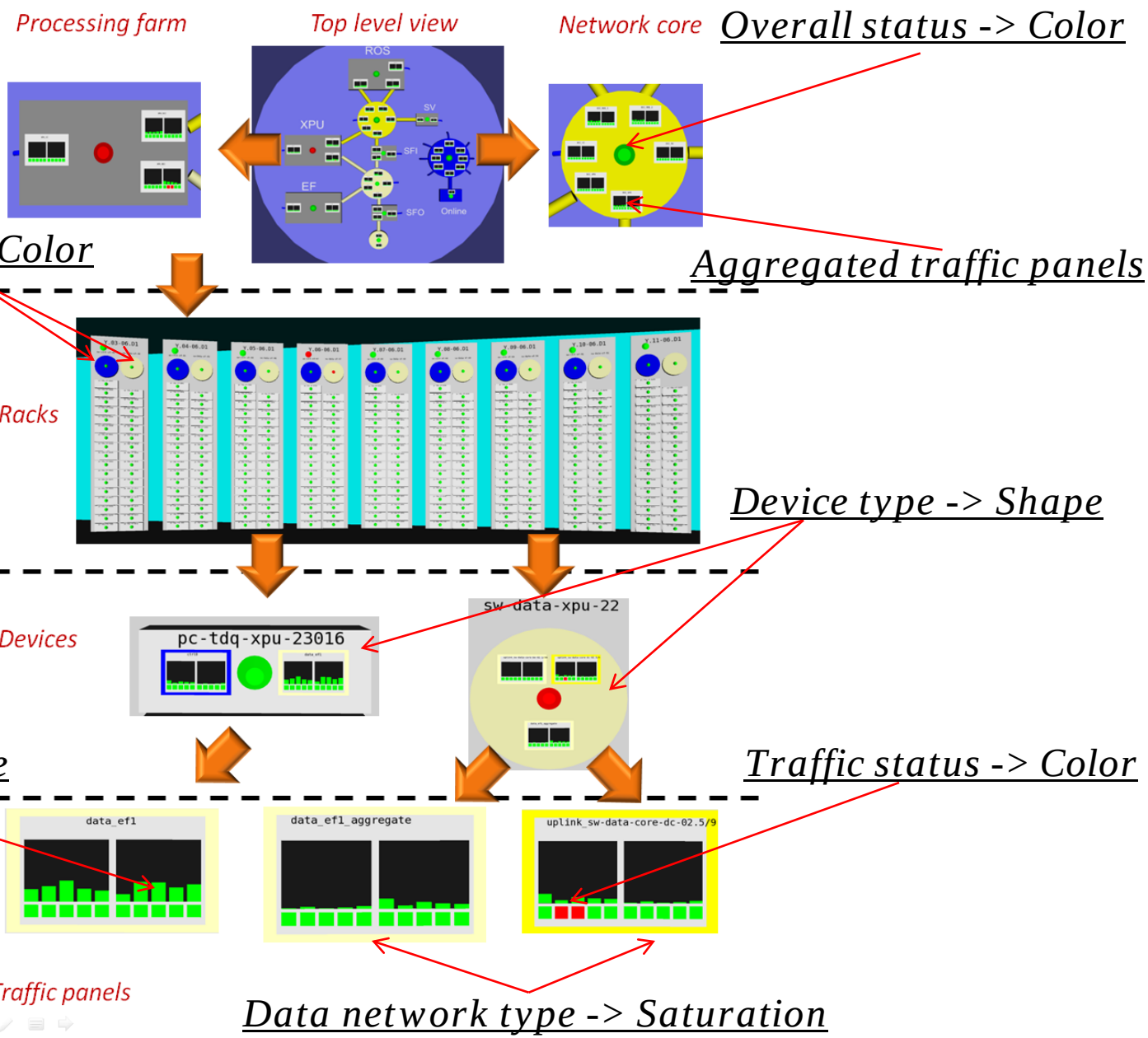
2D Visualization

- Relatively **inexpensive** in terms of resources and setup
- **Visual clutter** for overlapping networks
- **Two** degrees of freedom and restricted navigation paradigms

3D Visualization

- **Demanding** in terms of processing power, configuration
- Offers **additional dimension** -> better candidate for large scale complex models
- **Six** degrees of freedom and natural navigation paradigms (walk, fly)
- **Camera-object distance** can be evaluated

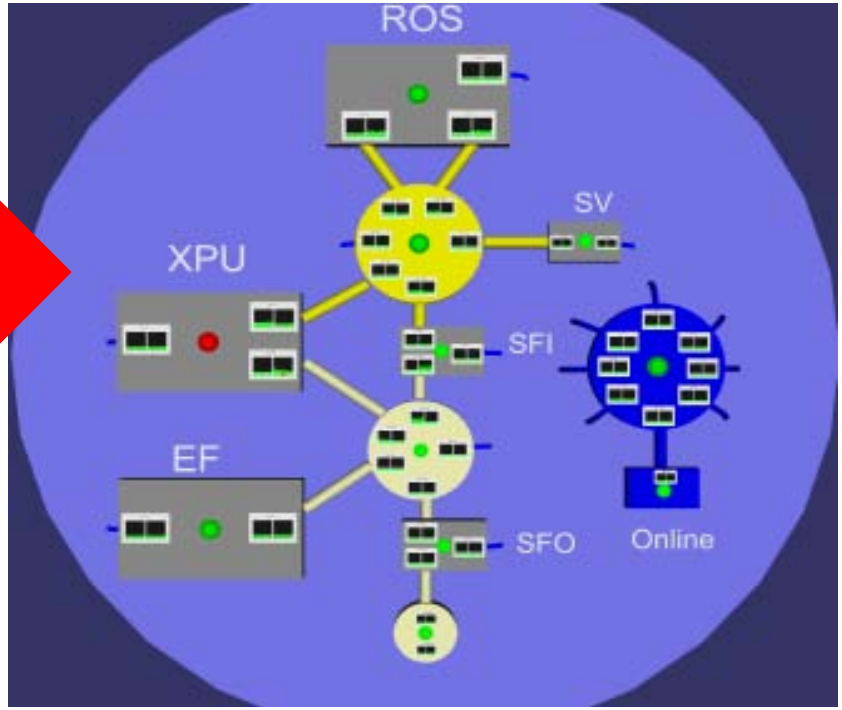
Hierarchical 3D Model



Optimized object layout

Top level view

- Follows data flow
- Control network as a backplane



Panoramic rack view

Aspect ratio improvement

OSG implementation - overview

OpenSceneGraph(OSG)

- Open source, portable, high-performance framework based on C++ and OpenGL
- Rigorous structure based on STL and design patterns (visitor, callback)

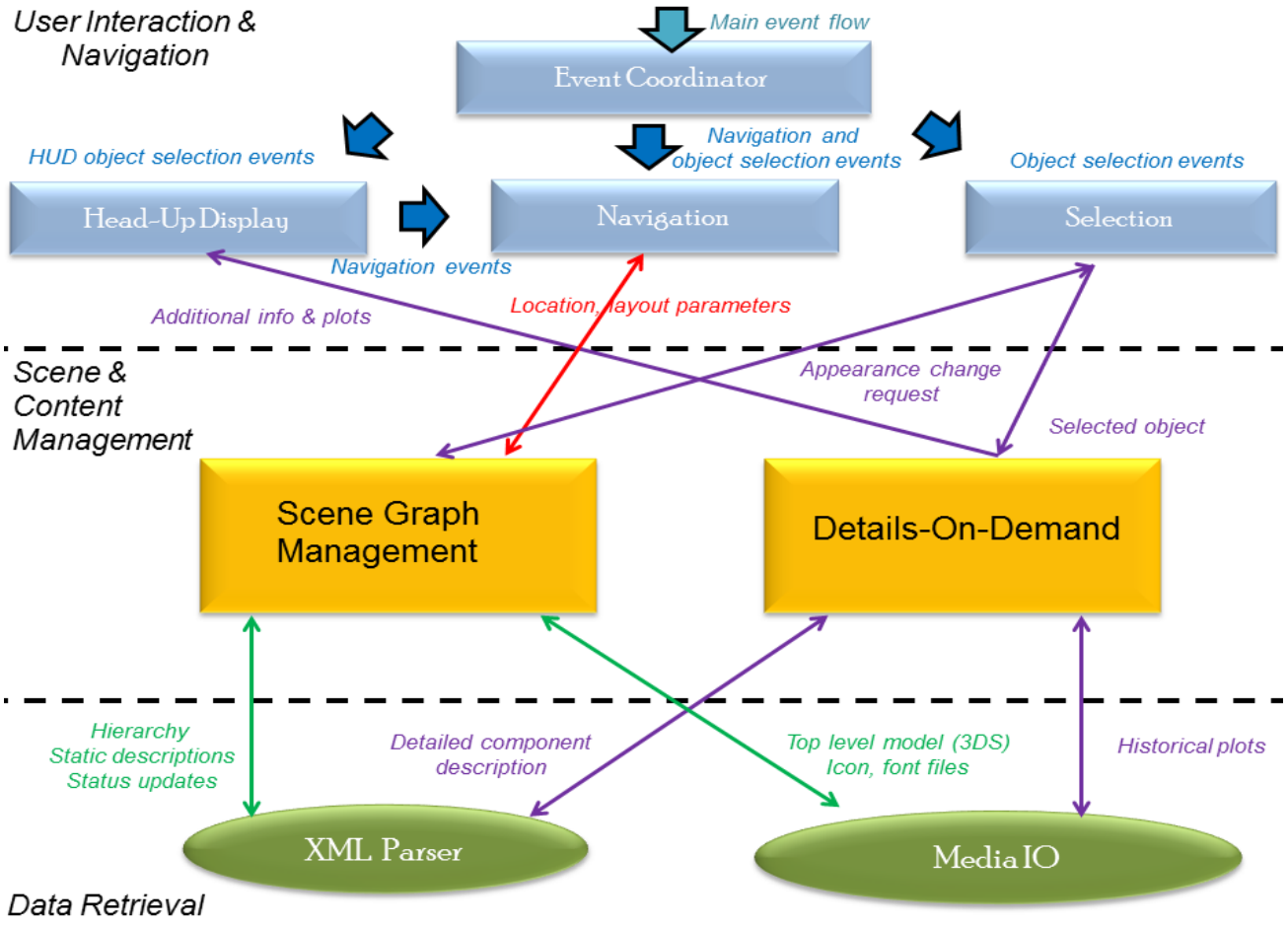
Why did we choose OSG?

- Thin wrapper on top of OpenGL -> access to low-level rendering options
- Rendering statistics display and API -> essential for performance tuning
- Bit masks for selection and specialized event handlers for interaction

Profited from the scene creation to perform several optimizations

- Frame rate >30fps
  - Minimize overall traversal time: UPDATE+CULL+DRAW
  - Adjust LOD ranges
- Real-time update impact minimization

Client architecture



Rendering and scene graph optimizations

Rendering optimizations

Impacts DRAW traversal time

Geometry rendering -> best solution was to use vertex arrays + triangle primitives + color binding per vertex –> 14% decrease

Custom geometry nodes optimized for fast rendering -> 15% decrease

Text rendering -> ~75% decrease

Low resolution object versions to use in Level Of Detail

Scene graph restructuring

Impacts CULL traversal time

Eliminated Transform nodes at the panel level-> ~66% decrease

LOD node rearrangement for flexibility

Real-time update

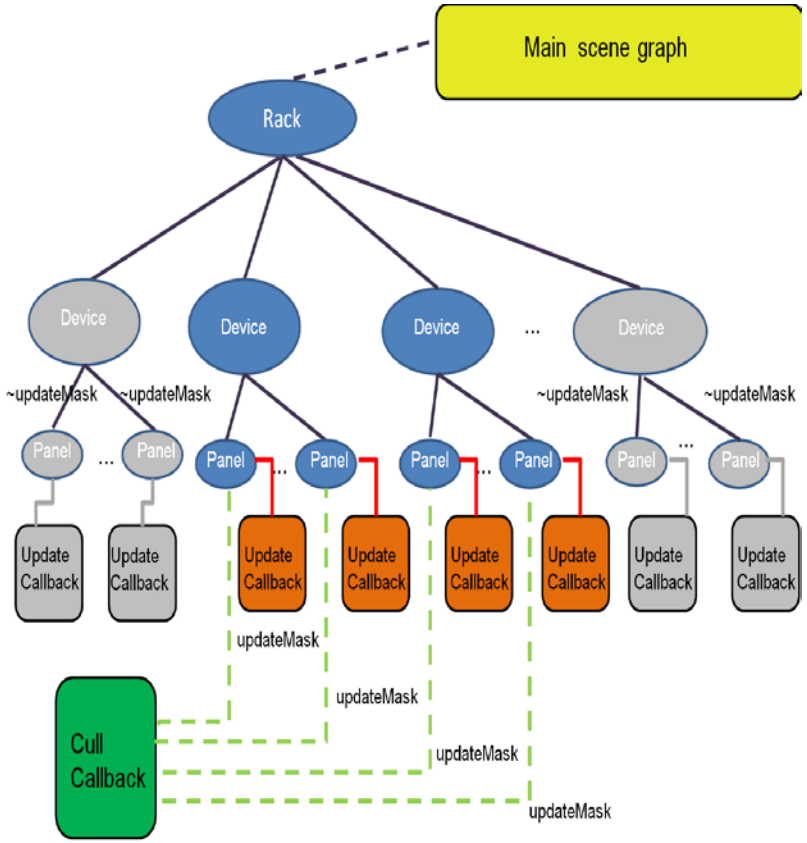
Based on visibility and proximity

New targeted update mechanism based on **temporal coherence**

Tested different granularities

- Individual node -> <30fps
- Device node – 26% increase
- Rack node – 38% increase

Decreased maximum completion time by spreading the updates over multiple frames



Interaction mechanisms

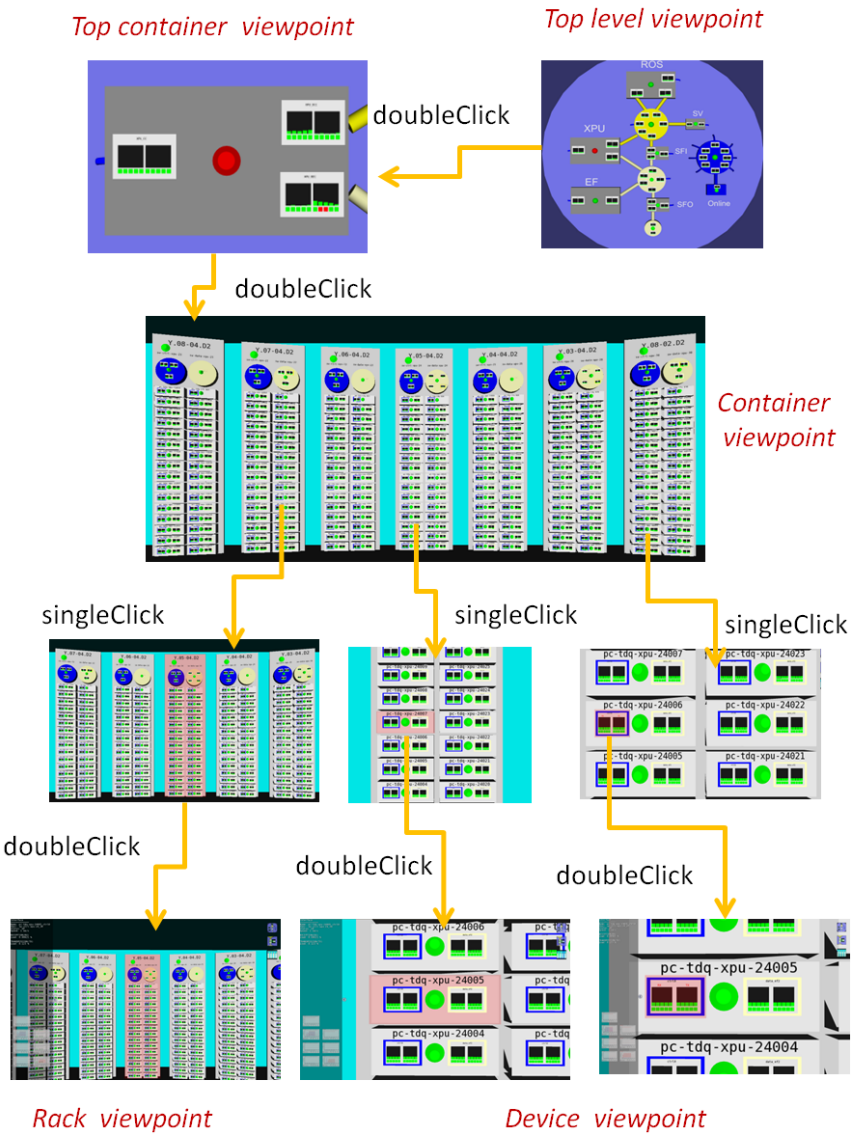
Mixing free and guided navigation

Context aware navigation

- Based on layout parameters
- Different navigation paradigm
- Radial navigation
- Field of view and speed control

Selection and highlight mechanism

Details-On-Demand in a Head-Up Display



Conclusions and future work

- Identified specific visualization requirements
- Chose 3D visualization and InfoViz guidelines
- Used open-source low-level framework OSG
- Intuitive interaction and navigation
- Frame rate > 30fps

Future work

- Integration of data taking parameters
- Rule-based expert system to improve error propagation rules
- Multiple views