DAQ software: Activity and Plans

Kostas KORDAS

Kostas Kordas, Mary Censa Ferrer
Frascati

31 Marzo 2005
ATLAS-Frascati mtg

Outline

A) Data Acquisition: General and the Data Collection system

B) Event Building: The Sub-Farm Input (SFI)

C) Event Monitoring: The SFI does not only do event building

D) Implementation and Tests: Lab32 at CERN and the lxcalc cluster here

E) New challenges in the Data Collection system

A1) Trigger and Data Acquisition System

1) 40MHz in Front-End pipeline memory

2) LVL1: decides in 2.5μs, accepts at 100kHz: push events to Read-Out Drivers (**RODs**)

3a) ROD $\rightarrow$ **ROB** (Read-Out Buffers), inside Read-Out System (**ROS**)
.....tranfer via 1628 Read-Out Links (ROLs), $\sim$ 1kBytes each event fragment

3b) Region of Interest Builder (RoIB) makes list & sends to L2 SuperVisor

4) L2SV assigns event to a L2 proccessor

5) L2: L2PU asks needed RoI data ($\sim$ 16 kBytes), decides in $\sim$ 10ms, accepts 3kHz

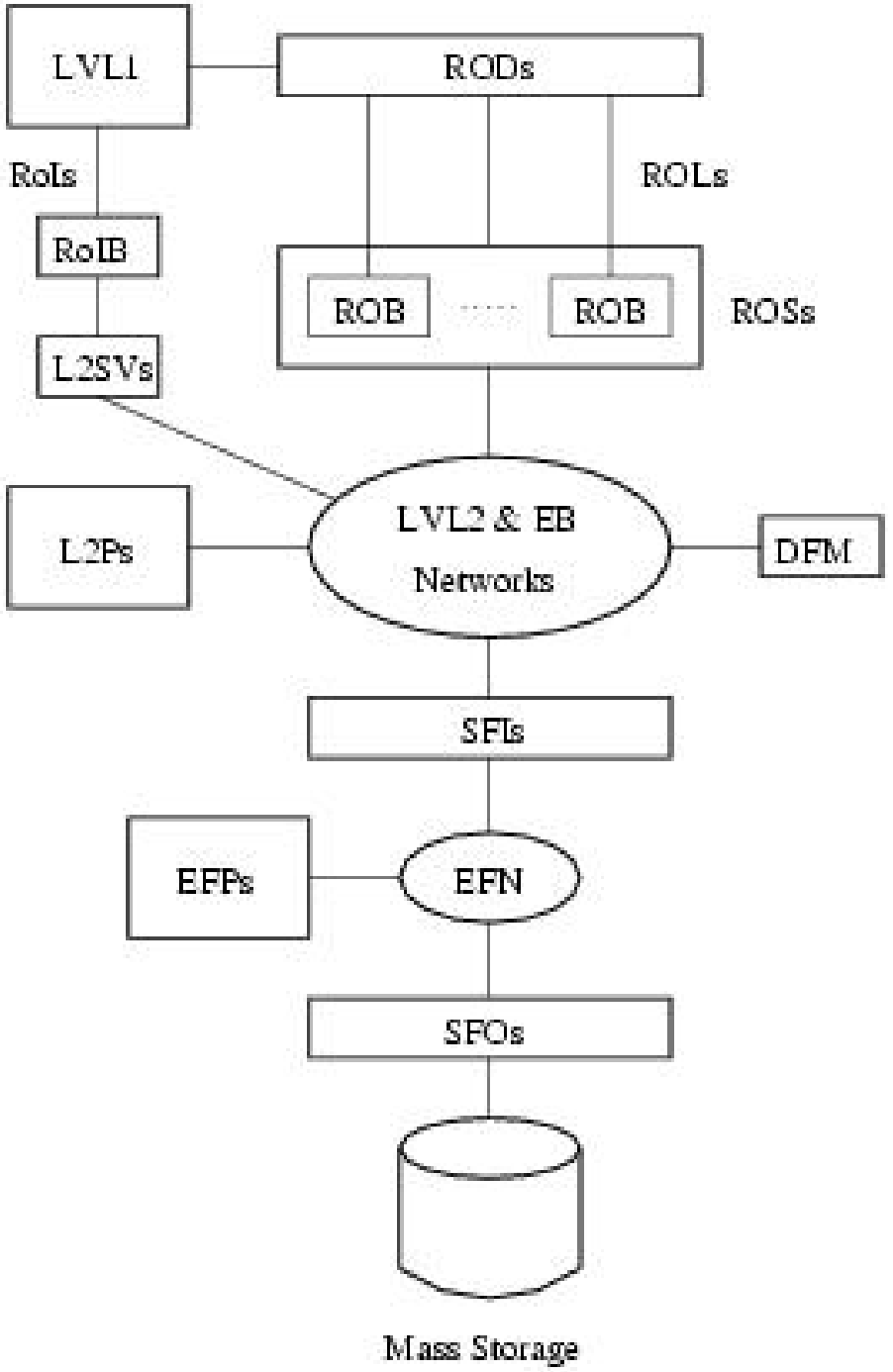
6) L2SV sends L2 decision to DFM (Data Flow Manager)

7) DFM clears ROSs or assigns Sub Farm Input (SFI) node for event building

8) **SFI**: a) aks all ROSs for event fragments, b) builds event and keeps it in memeory, c) notifies DFM to release ROSs

9) Event Filter asks full events from SFIs

10) EF: decision in $\sim$ sec, at 200Hz. If kept, event sent to mass storage, via the Sub Farm Outout (SFO)

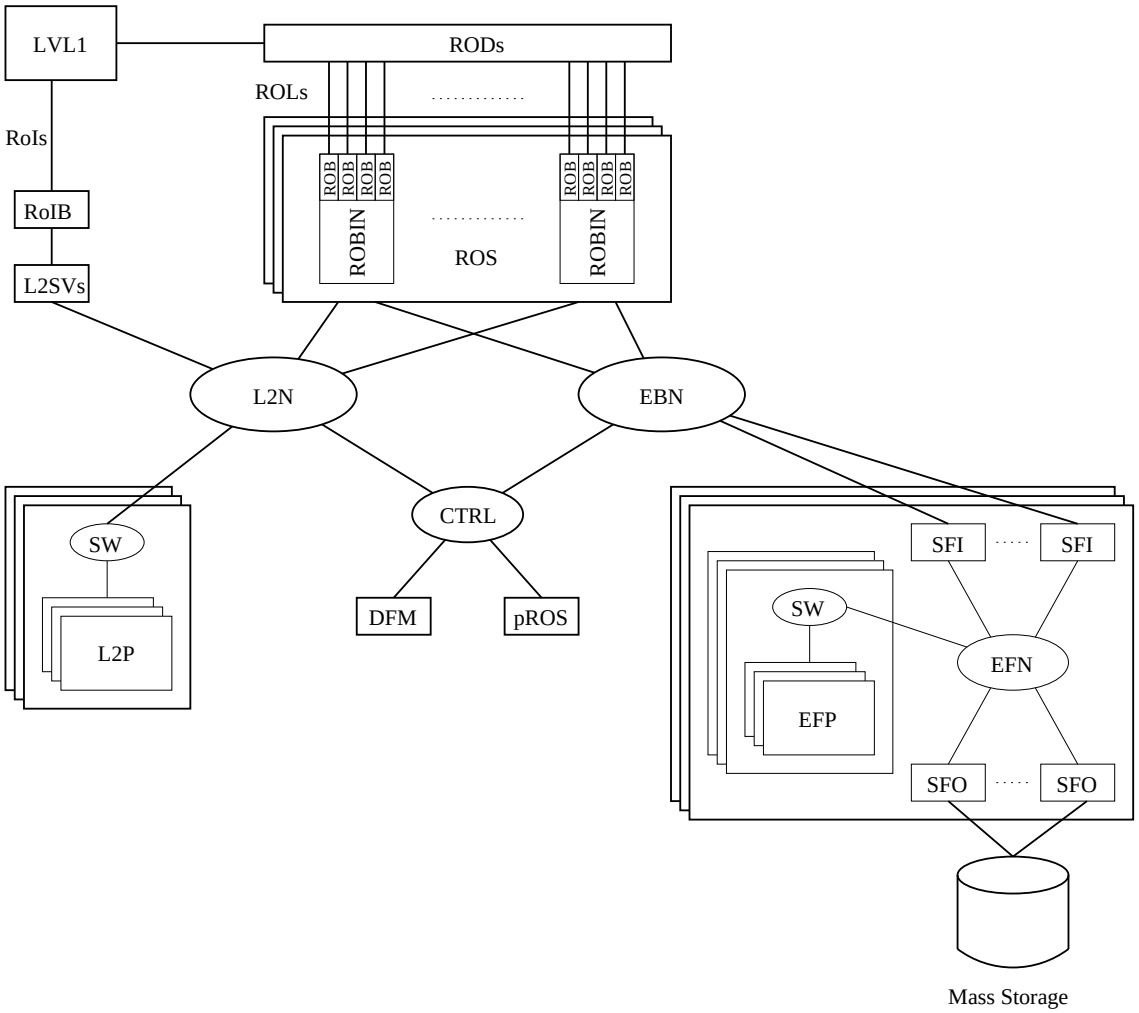


A2) Data Flow and High Level Trigger implementation

Data Flow: Guide data from RODs to Event Filter & disk.

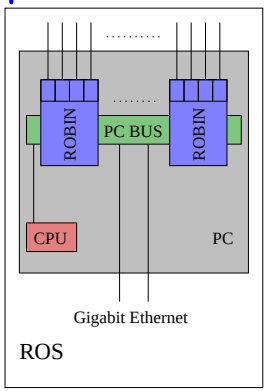
High Level Trigger: Level 2 and Event Filter (a.k.a Level 3)

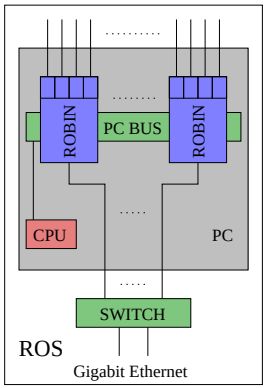
Implementation:



ROS: initially get data from the ROS PC. Later, SFI can get them from the Read-Out Buffers (on the ROBIN cards) directely)

3 ROBIN cards per ROS PC, 4 ROBs per ROBIN $\Rightarrow$ 12 ROBs per ROS.

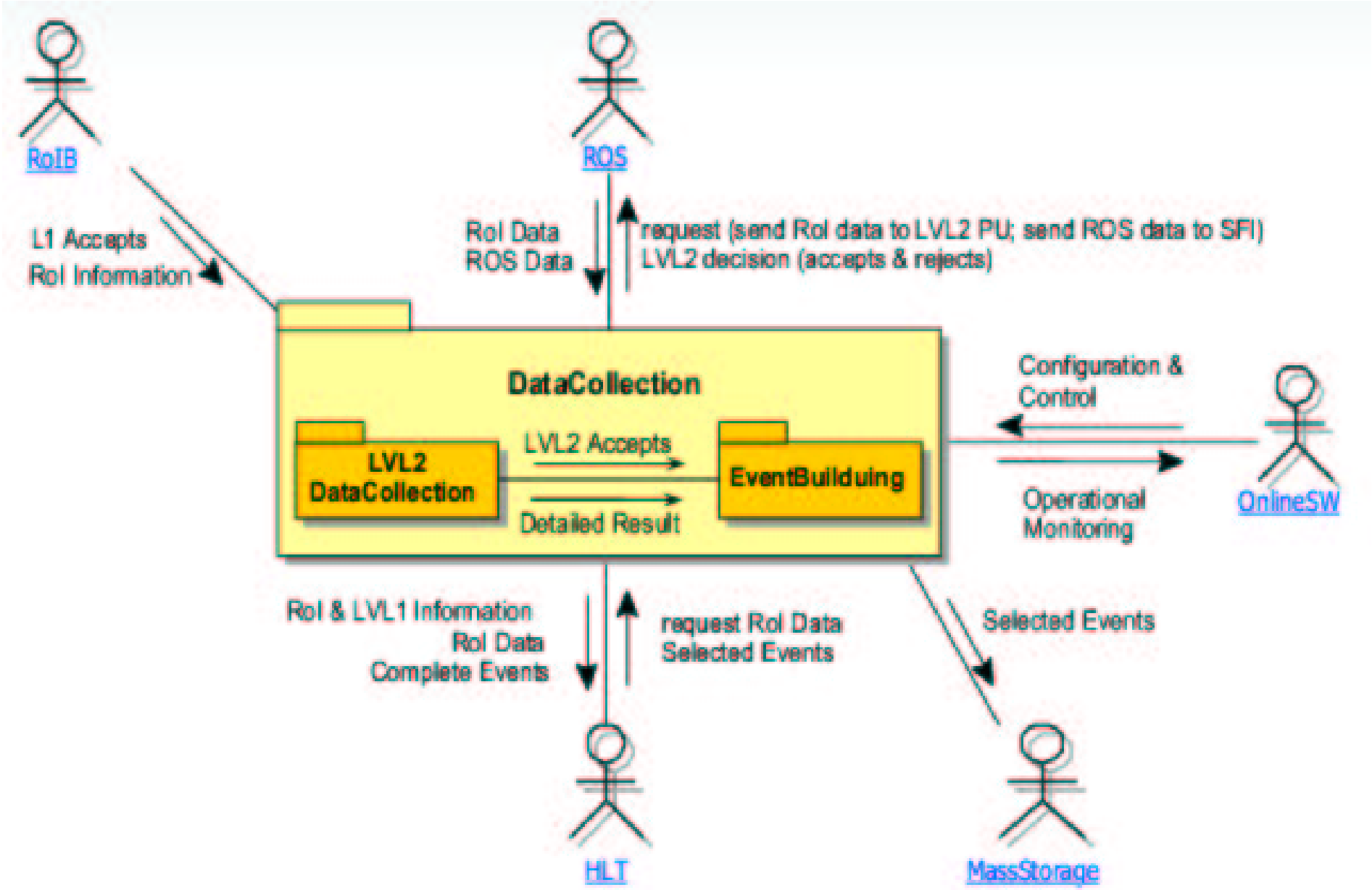




KK4

A3) Data Collection

Data Collection: Transport needed data where needed, as needed (fragments or full events)



B1) Event Building: the SFI

- Event Building is done in Linux PCs in the Sub Farm Input sustem.
- SFI is just a server $\Rightarrow$ Reacts on requests
- Runs independent tasks in parallel activities:

  -Dispatches incomming messages from the network

  -Requests event fragments from the ROSs

  -Indentifies events fragments and builds events

  -Re-asks ROSs for missing event fragments

  -Sends events to Event Filter upon request

  -Provides events for monitoring

B2) Event Building (EB): notes

- 30Hz EB rate per SFI (for events of $\sim$ 1.5 MBytes each)
- 3kHz EB rate achieved with 100 SFIs working in parallel
- SFIs request data fragments from ROS
- Today only complete events can be built
- Events can be requested from SFIs for monitoring
  - -A copy of the event is shipped to the monitoring PC
  - -Moderate rate, otherwise Event Building performance drops
- SFI keeps Events in memory until a EFD node requests an event
  $\rightarrow$ Backpressure in case EF does not empty SFI fast enough

Event Monotoring: general

• On-line Monitoring: provide on-line sample of event-data to users' "monitoring tasks" within the DAQ project.

• Allow user programs to request (full or fragment) event-data flowing through the DAQ system, according to criteria.
Source of data: ROD, ROS and SFI

• Monitoring should have a minimal effect on the DataFlow performance.

•Previous design had a unique "Distributor" between Sampler (e.g., SFI) and Monitoring Tasks: bottleneck.

•Re-desgin avoids this: tasks get data from Sampler directely:

1) Monitoring Task requests specific events from specific SFI

1.1) "Conductor" adds Monitoring Task to the "Monitoring Tree" and connects it to SFI

2) SFI fetches events with match selection criteria. Sends them to Monitoring Task directely.

3) Repeat till number of requested events is given.

Event Monitoring: our job

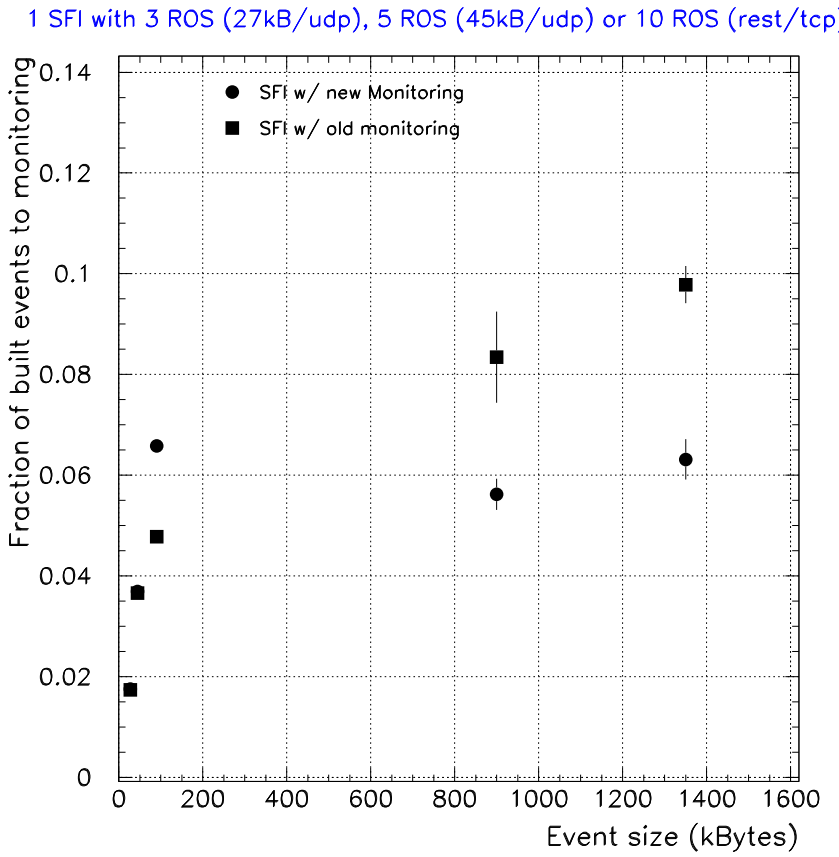
- We are responsible for the sampling functionality of the SFI.

- Implemented selection criteria in December (using the traditional monitoring service).

- Implemented new monitoring scheme, We are in the proccess of checking if harms the SFI's main job: event building!

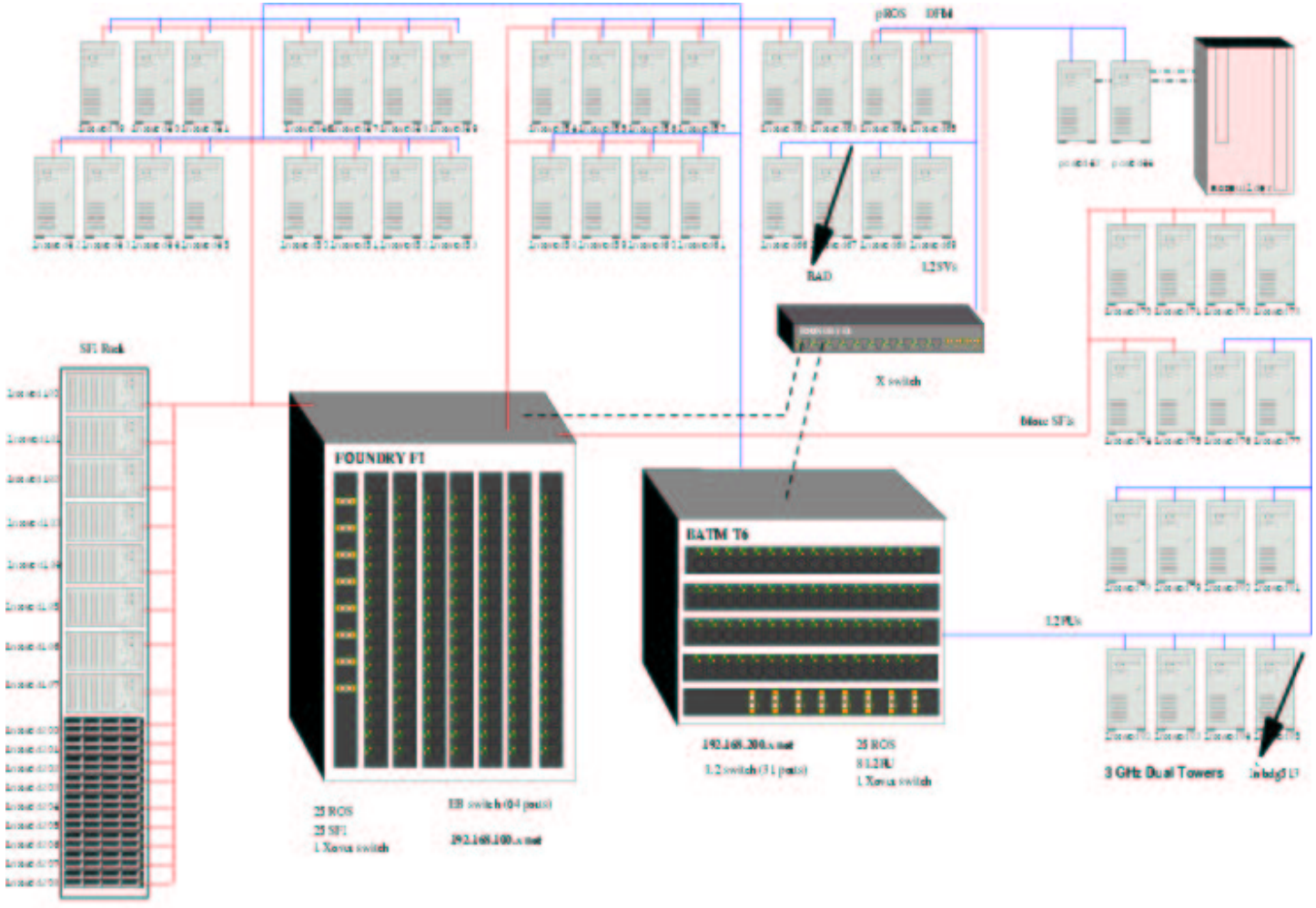
- Final implementaiton these days (till April 8) and tests before the new TDAQ release (end of April).

- New feature: adopt speed of requests to what the SFI can provide. Result: limit # of events sent to monitoring.



Lab32 set up for tests

- Check effect of monitoring on SFI's event-building capabilities.
- Setup at Lab32 (CERN): many ROS, 1 DFM, 1 SFI. monitoring



Comparison to other setups

- Cosmic ray stand:

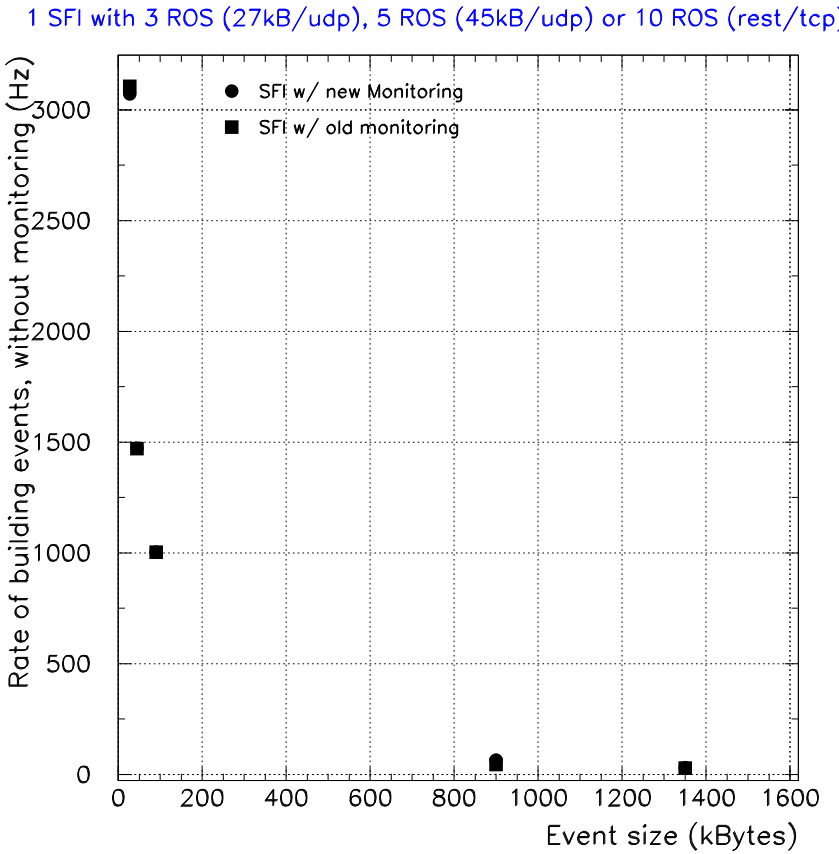
We have real data, going up to the ROS $\rightarrow$ disk.
No event building is done.

- lxcalc: we have 5 Linux PCs $\sim$ free for us.

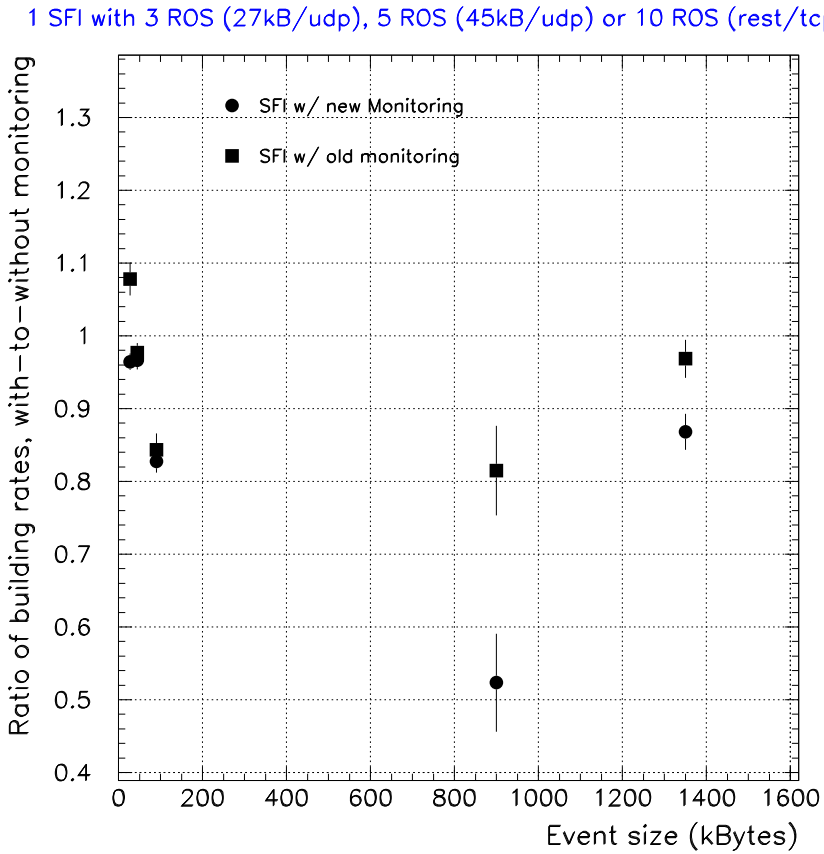
They have GigaBit ethernet connection, like in ATLAS setup.
- Have to make UDP protocol work for their communucation
(now we can only use TCP)
- Need to move to Scientific Linux

SFI tests with new monitoring

- Bulding rate without monitoring:



- Effect of monitoring on building rate:



Data Collection: Challenges ahead: 1

- Now we have 4 types of data in DAQ:

Physics, Fast Physics, Calibration, Forced Accepts.

- Possible Evolution of functionality

  -Fact: DFM (DataFlowManager) load balances SFIs.

  -DFM could assign Calibration Triggers to distinct SFIs, according LVL1 Trigger Type

  -These SFIs would request data fragments exclusively from SubDet X
  $\rightarrow$ Partial EventBuilding

  -EFD would request calibration data from those specific SFIs.

Data Collection: Challenges ahead: 2

• Partial EventBuilding according to LVL1 Trigger Type word is feasible:

-Based on a (per run) static assignment policy

-Can be added if there is clear demand

• RoI based partial EventBuilding?

-Today, SFI does not look inside EventData
→ Just header information

-RoI information not at hand at SFI
→ SFI would need to request data fragments from ROSs associated to LVL1 RODs first and then needs to decide which ROBs to request

-If demanded, would need some discussion in TDAQ and development in DataCollection

Plan

- Participate in this evolution of the Data Collection system
- Possibly participate in the installation of the system in ATLAS (starts in April-May).