ATLAS Trigger and DAQ Projects

Rainer Bartoldus

Overview

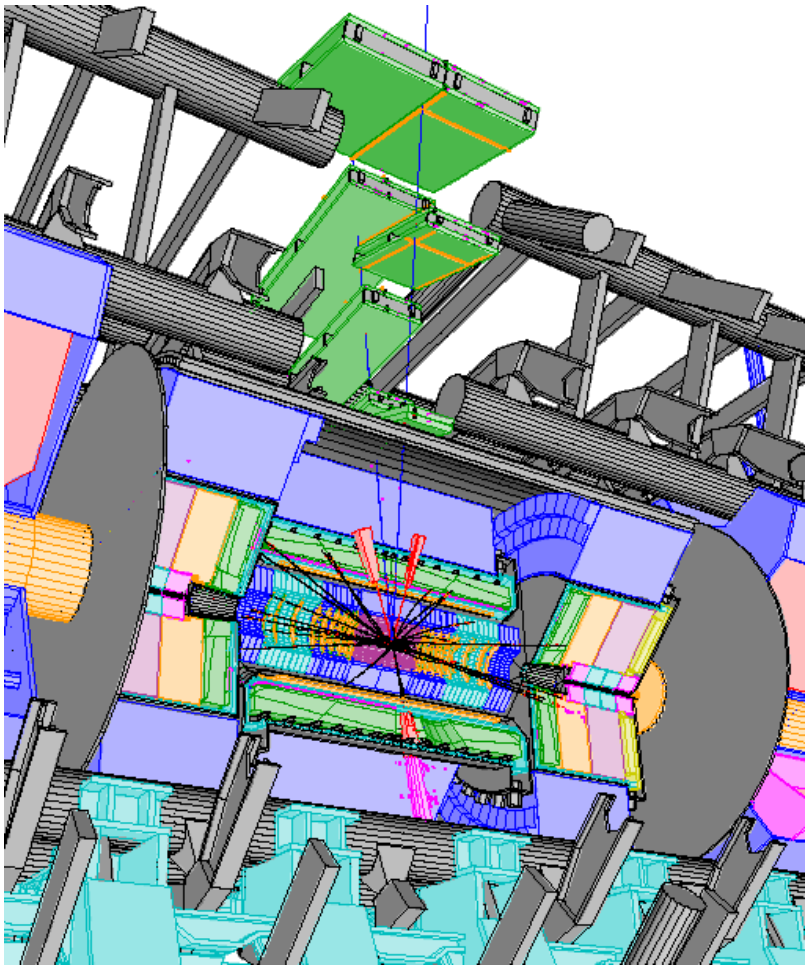
- This talk gives a brief introduction to ATLAS/TDAQ and highlights the following areas with SLAC leadership:

* High Level Trigger Configuration
* Partial Event Building
* Online Beam Spot Measurement
* Operational Reponsibilities

- SLAC's involvement in the HLT algorithm development will be covered in the next talk given by Ignacio Aracena

- The SLAC TDAQ team are: Sarah Demers and Ignacio Aracena (RAs), Andy Salnikov, Su Dong, R.B. (Lead)

ATLAS Trigger



Three-level hierarchy

40 MHz

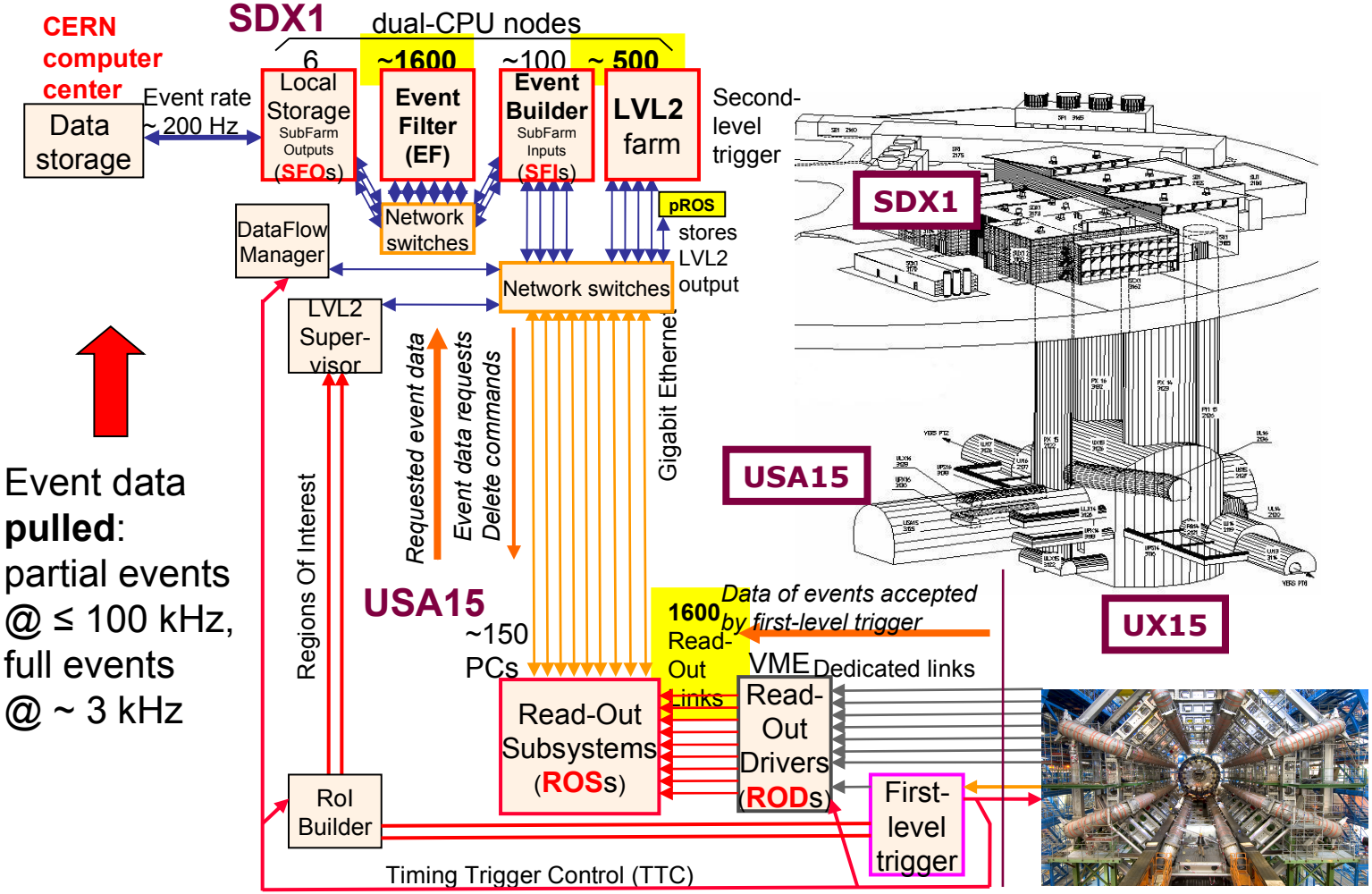
100 kHz

3.5 kHz

200 Hz

- **Level 1** (Region-of-Interest): Identify high-p_T lepton or jet candidates based on coarse information from calorimeter and muon chambers; 2.5 μs latency
- **Level 2**: Use L1 RoI (η,φ) as seed to guide reconstruction; typically 2% of the event is read at full granularity; 40 ms latency
- **Event Filter**: Processes fully built events to apply offline-like algorithms; ~1 s latency

Trigger / DAQ Architecture



Event data **pulled**: partial events @ ≤ 100 kHz, full events @ ~ 3 kHz

Event data **pushed** @ ≤ 100 kHz, 1600 fragments of ~ 1 kByte each

Hardware Commissioning

After having worked with 4+1 racks for 8 months, have started commissioning 3 new racks every week. Up to ~~17~~ 20 today, to reach a total of 23 by the end of the month.

Not all boxes arrived in mint condition...

High Level Trigger (HLT) Configuration

High Level Trigger (HLT) Configuration

* Configuration Challenge
  - When we joined ATLAS TDAQ two years ago, one of the major challenges (and concerns) was the distribution of configuration data to the thousands of HLT clients
    - With O(2000) nodes (400 L2 + 1600 EF) × 8 cores/node × 1 client/core × O(10-100) MB/client, the system has to generate and **deliver O(0.1-1) TB of data within O(1-10) s**
  - No single server can handle that kind of a load
    - Not feasible to handle 16000 connections to begin with...
    - Even if one managed to stagger them, and assuming 70 MB/s (unrealistic because of small packet/round trip overhead)
    - It would take **6.3 hours to configure the entire system**
      - Even if one compressed it to 10 MB it would still be 38 minutes between making a change and being ready to run
    - Clearly, the trigger menu, prescales etc., will have to be adjusted frequently at the beginning

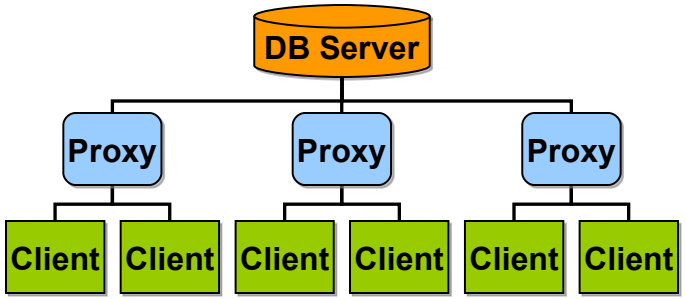
→ Had to find a way to turn the configuration around fast

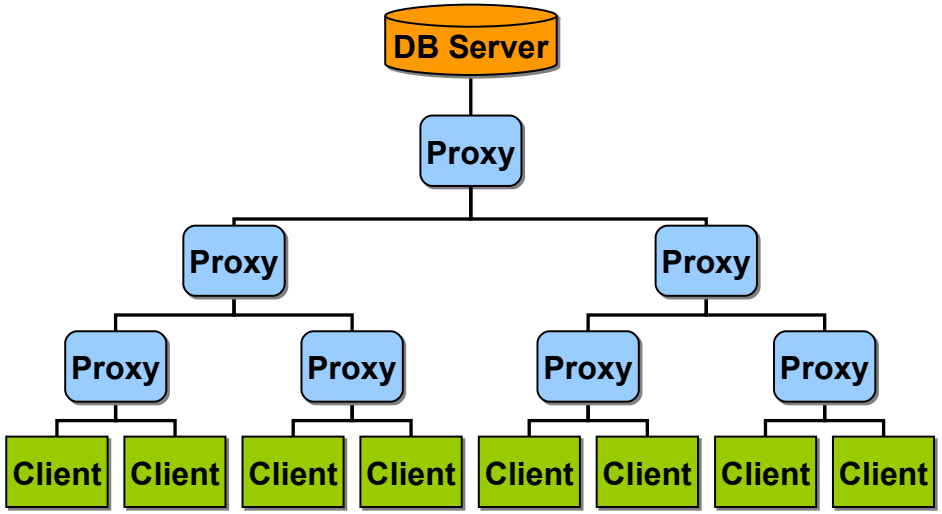
DbProxy

* Strategy
  - Must reduce the number of connections (→ multiplexing) and network traffic (→ caching)
  - In principle: add more servers, and bring them closer to the clients
* "DbProxy" Solution
  - Originally developed by Amedeo Perazzo, taking advantage of the open MySQL protocol
  - Successfully deployed in ATLAS Online in 2007
  - Since then an integral part of Technical Runs (TDAQ only) and Commissioning Periods (with all subdetectors)
    - Running was done against MySQL, or MySQL replicas of the ORACLE master
  - This solution has enabled HLT commissioning and scaling from 1 to 4 and now to 20+ racks

DbProxy Design

* Characteristics
  - Process that sits between HLT client and DB server; acts as a server downstream and as a client upstream
  - Transparent to the HLT client that connects to it
  - Can connect to other proxies so allows one to build hierarchies:

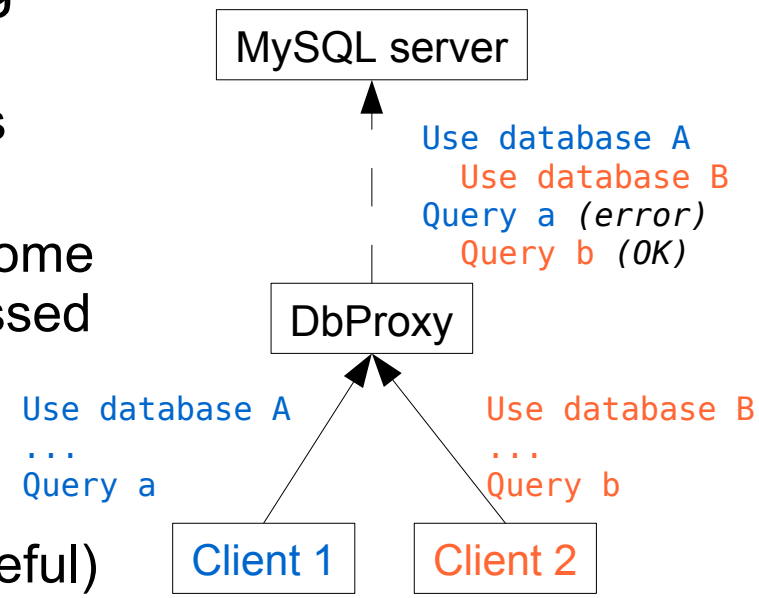




  - We found it practical - and sufficient - to follow the TDAQ farm segmentation:
    - One node-level proxy per node to serve 8 cores each
    - One rack-level proxy per rack to serve 30 nodes each
    - One top-level proxy for each of L2 and EF to serve 20+ racks

DbProxy Protocol

* Contraints imposed by multiplexing
  - Queries and responses have to be *context-free*, implying that all requests must be *atomic*
  - We found this not to be the case for some transactions and could get this addressed in the LCG (CORAL) code
  - This is an example of how a proxying technology puts constraints on the protocol that are not present in a (stateful) single server connection

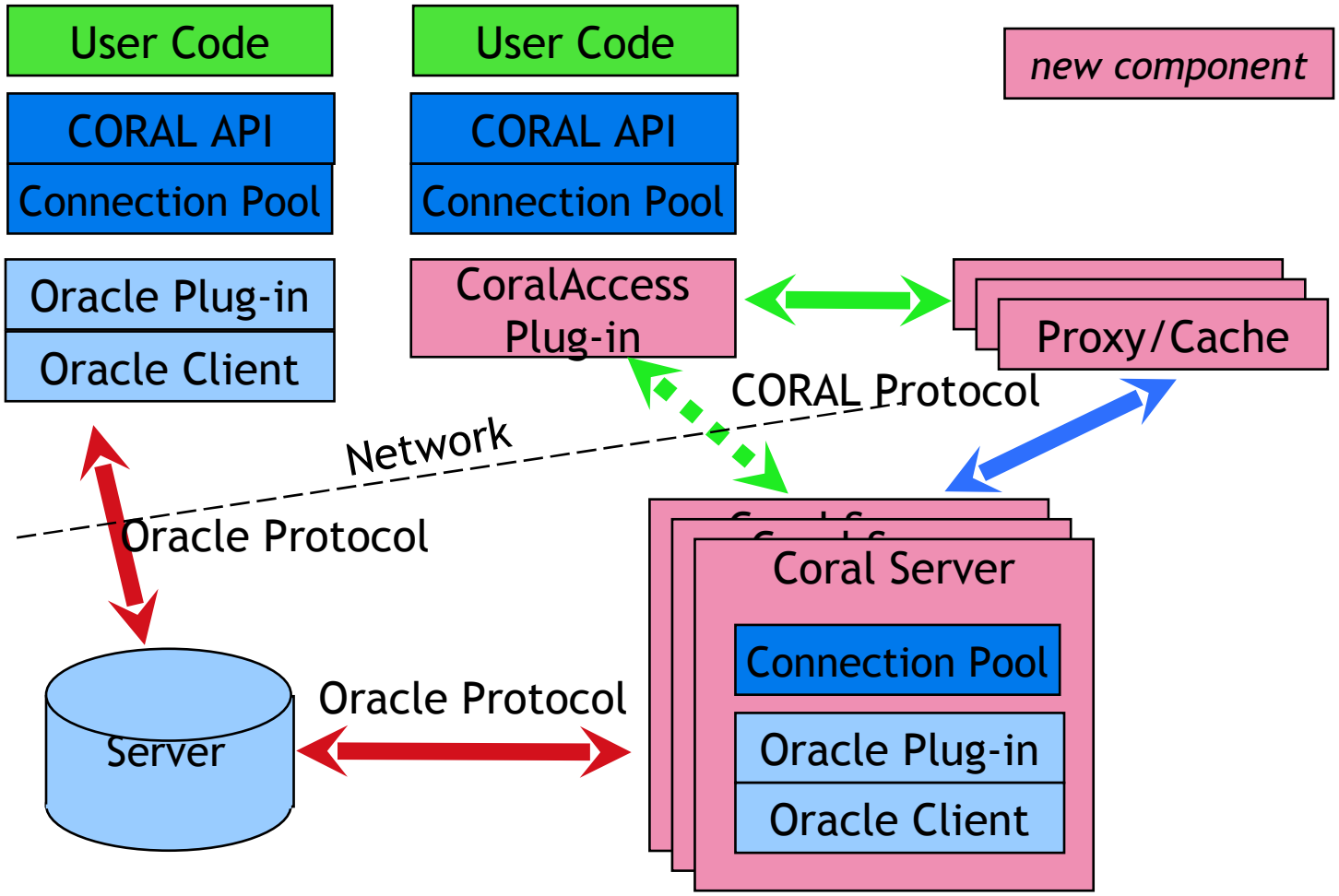


* Constraints imposed by caching
  - Caching also requires that query-reponses have to be *invariant*
  - The client must not ask, *e.g.*, “give me the last run number”, “the time of day”, or call random

CORAL Server

* Background
  - LHC experiments use a relational abstraction layer called CORAL (C++ client API to relational DBs)
  - This lets the client select between various technology plugins: MySQL, Oracle, SQLite, Frontier
  - However, this leaves the proxy tree technology-dependent, as in our current MySQL implementation
    - ORACLE's closed-source (proprietary) protocol prohibits server implementations
  - We decided that the best way out was to push the technology choice behind the proxy tree, *i.e.*, define an independent protocol
* CERN/IT
  - When we talked to our CERN/IT friends they had arrived at the same idea for independent reasons, mostly involving security
  - This is what launched our joint CORAL Server Project last fall

CORAL Server Components

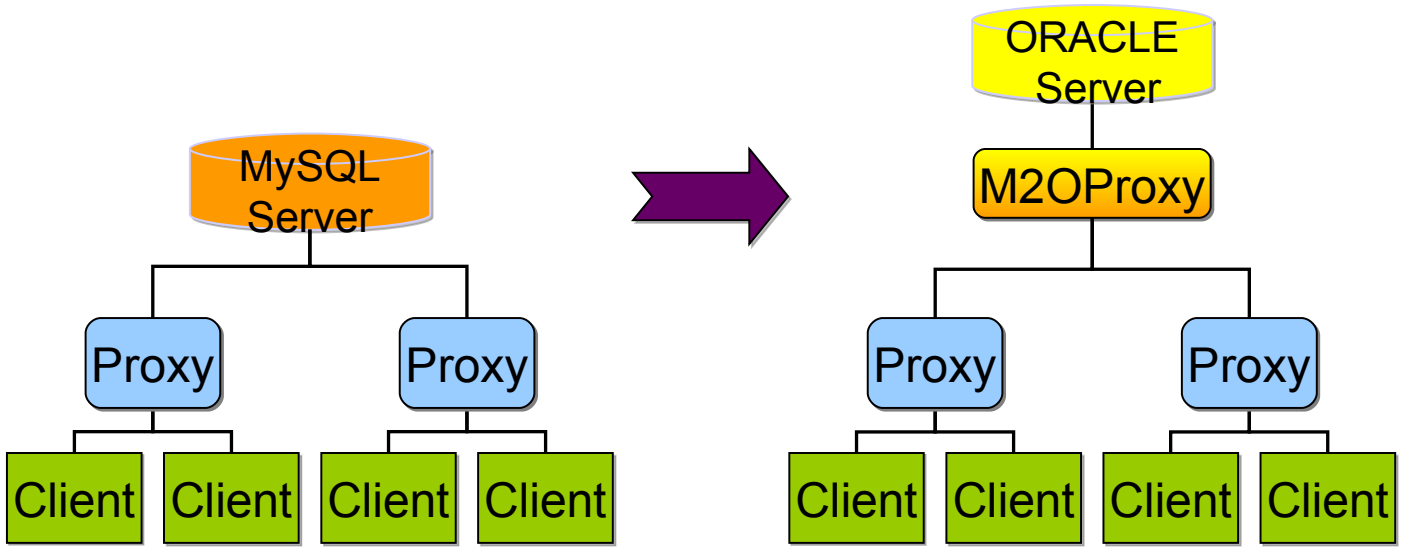


Picture courtesy Dirk Duellmann (CERN/IT)

CORAL Server Project

* Timeline and Milestones
  - We have been holding weekly meetings between the SLAC team and the CERN CORAL team since last fall
    - CERN is implementing the server and the client plugin, SLAC is contributing the new CORAL proxy
    - Our main stake has been to arrive at a protocol that meets the demands of a proxy for ATLAS Online
  - This spring, the project has reached official status within LCG, following endorsements from LHCb and Alice
  - Our goal has been to have a testable prototype by April 2008 to be commissioned before first data, unfortunately, CERN/IT has been hampered by man-power issues and the new agreed upon milestones extend to the end of this year
  - While this is a solution with presumably widespread applications reaching beyond the ATLAS community, ATLAS online needed something sooner...

"M2O" Translator Proxy



- To buy time for the CORAL project, we developed a "MySQL-to-Oracle" bridging proxy (Andy Salnikov)
- This was tested and successfully deployed last month
- It has already triggered much interest in the offline community; experts consider this an "amazing feat"
- It is not a permanent solution, though, due to strong dependence on non-standard type conversions

Proxy Performance

* Translation
  - Our early tests already established that the configuration through the M2O proxy is in fact *faster* than a direct ORACLE connection
    - This has to do with CORAL protocol overhead, involving schema discovery, which is effectively being cached by the M2O proxy
* Scaling
  - The infrastructure of the distributed proxy tree with the translator on top makes the entire farm behave like a single client (our design goal)
    - It makes 10000 clients all see a local MySQL server, while the ORACLE server sees a single client (*i.e.*, one L2 and one EF client)
  - It has been demonstrated that the configuration of the farm takes as long as the configuration of a single node (→ Scaling)

Partial Event Building

Event Building Capacity

* Partial Events
  - Fully built events are necessary for physics but not necessary for (all) calibrations of individual sub-detectors
    - *E.g.*: no need to read out the LAr calorimeter (more than half of all ROBs) for Inner Detector (ID) alignment
  - Can afford substantially higher rates by building partial events
* Available Bandwidth
    - (Up to) 100 kHz L1 accept rate, L2 accepts 3.5% → 3.5 kHz input to event building
    - Including data from all 1600 detector Read Out Buffers (ROBs) of O(1k) each, the full raw event size is ~1.6 MB
      - 1.6 MB × 3.5 kHz = 5600 MB/s total I/O to event building farm
      - Each event building node (SFI) handles 70 MB/s → 80 SFIs
    - Generally the available bandwith is:
    - Number of ROBs × fragment size per ROB × L2 accept rate
  - By reducing the number of ROBs one can gain in L2 accept rate

Partial Event Building

* Software Status and Testing
  - Code implemented and released this spring
  - First successful tests on the real partition using a “dummy” L2 algorithm
    - Running off simulated QCD dijet events, seeded by L1 electromagnetic object (L1 EM RoI)
    - When a good L1 EM RoI is found, L2 algo fills pre-defined list of ROBs
    - Event Builder sees non-empty list and builds event partially
    - Partial events are not processed by EF but written directly to stream
  - Performed Online validation
    - Verify that all (and only) requested ROBs appear in output stream(s)
    - If an event is triggered as both physics and calibration it is fully built, written to the physics stream, and stripped on the calibration stream
    - Event stripping is also in place for calibration events selected at EF

Partial Event Building (cont)

* Current Development
  - Started working on monitoring of the partial event build
    - Event sizes on input and output of EB, ROB distributions, etc.
    - At this stage can also determine “savings” compared to full build
      - Help to optimize bandwidth for each subsystem
  - Now integrating realistic algorithms
    - New calibration algorithm from the LAr calorimeter group
    - New calibration algorithm for the Inner Detector using isolated tracks
    - Started collaboration with TileCal group
  - This is becoming a little industry; expect more demand from subsystems following these successful examples
  - Anson Hook started working on exploiting this for our own project of ID alignment using off-center beam collisions

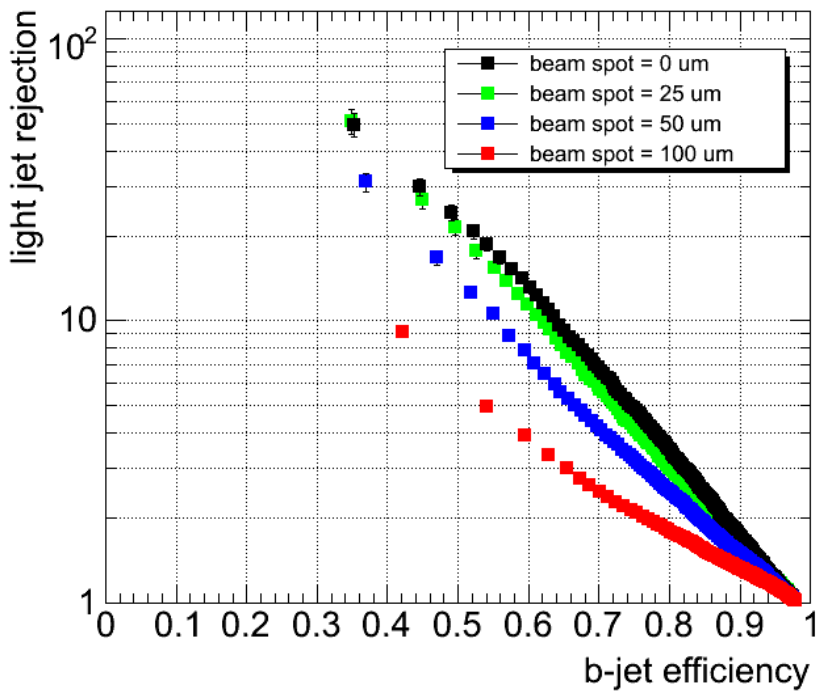
Online Beam Spot Measurement

Online Beam Spot Measurement

* Machine parameters (sizes and angle)
  - At design luminosity (10^{34} cm^{-2} s^{-1}): σ_{xy} ~ 17 μm
  - Startup pilot run (10^{31} cm^{-2} s^{-1}, β^* ~ 4 m): σ_{xy} ~ 45 μm
  - Bunch length: σ_z ~ 7.6 mm
  - Crossing angle: 0.3 mrad
* Variations during running (in position)
  - Expected to be ± 300-600 μm **during running** for the first 1-2 months
  - Eventually down to ± 30-50 μm under stable beams
  - Still ± O(1mm) jumps possible between fills
* Measurement with the ATLAS HLT
  - Useful in many respects, especially as feedback to LHC operators
  - *Critical* for algorithms that depend on IP position such as *b*-tagging

Beam Spot Information in the HLT

* HLT Algorithms Depending on the Interaction Point
  - The HLT can not only be source for the beam position, it also needs to know it in algorithms such as *b*-tagging
  - So far, the *b*-tagging slices and performance studies have assumed (0,0,z)
  - *b*-jet efficiency/light-jet rejection are much degraded already at shifts of O(50 μm)
  - It is apparent that the HLT beam position must be bootstrapped at the beginning of each fill, and likely also be updated during the run
  - This is a critical need for the experiment which has simply been overlooked



Beam Spot Algorithms

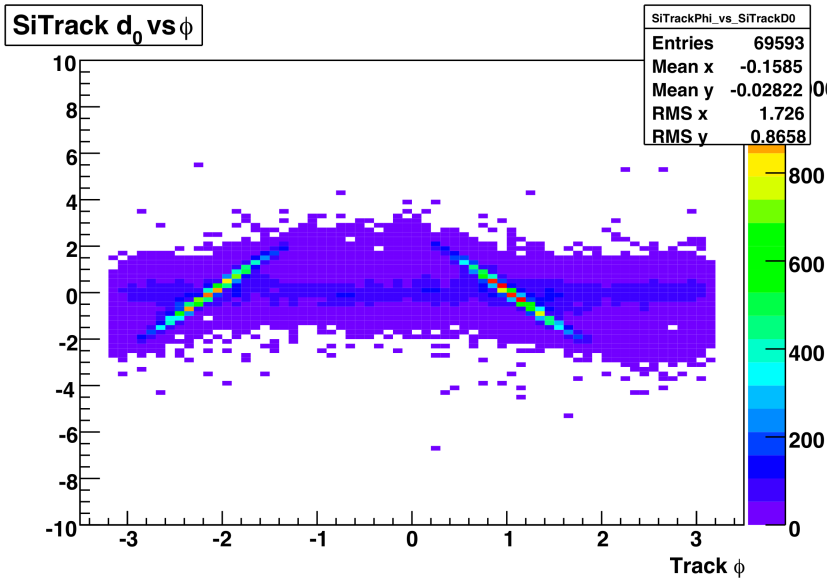
Two complementary algorithms are now available

* Track-based:
  - A. Cerri (CERN) implemented CDF L2 "d_0 versus φ" algorithm: extract IP position from amplitude and phase of sinusoidal shape
* Vertex-based:
  - D. Miller (SLAC) succesfully implemented an event-by-event vertex method
* Performance
  - The vertex algorithm has been demonstrated to yield 20-30 μm **per vertex** (*i.e.* times $1/\sqrt{N}$) with timing of < 300 μs running parasitically (feeding off already fitted tracks), well suited for L2
    - Assuming 25 Hz of usable events, this would result in a few μm position measurement every few seconds, and spot sizes and angles to similar precision within a few 10s to 100s of seconds

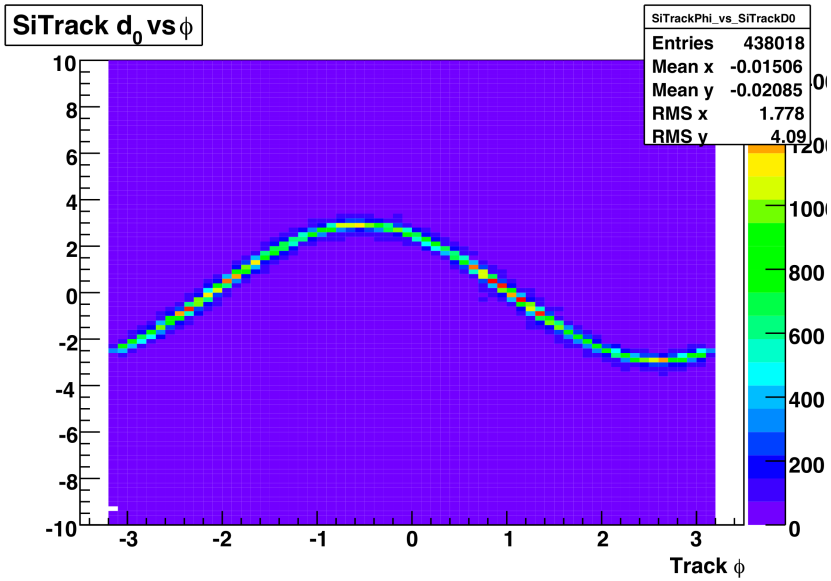
Beam Spot Tracking

Simulated *ttbar* events deliberately shifted to x=1.5mm y=2.5mm

Default tuning



Robust tuning



Our beamspot study revealed a bias that has its origin in this L2 tracking inefficiency (too narrow window around d_0=0). → Fixed

Beam Spot Technical Infrastructure

* Collecting the information from the farm
  - Building on existing infrastructure for online monitoring but need faster feedback than for bulk of histograms, O(1000) per client
  - Established a dedicated "Express Gatherer" to serve just beamspot and perhaps related information (*e.g.* online luminosity)
* Feeding it back to the machine
  - Information to be extracted from histograms, translated to ATLAS slow control and pushed into standard LHC interface (called DIP)
* Feeding it back into the HLT (ambitious)
  - DB access is much too slow and disruptive during the run
  - Our favored solution is to introduce a pseudo ROS (pROS), as used for the L2 decision, which pulls information from Gatherer
  - This naturally builds the parameters into every event, and they can be converted from the bytestream into conditions offline
  - Currently exploring this solution

Operational Responsibilities

* The SLAC team has taken on a number of online responsibilities
  - Sole responsibility for the DbProxy technology and the configuration infrastructure, involving conditions metadata, geometry and trigger configuration databases (A. Salnikov, S. Demers, Su D., R.B.)
  - DAQ Partial Event Build (I. Aracena)
  - TDAQ expert status and support for Technical Runs and Commissioning Runs; Participation in HLT farm commissioning (S. Demers, A. Salnikov, R.B.)
  - Currently performing HLT release validation (I. Aracena)
  - Coordination of Online Beam Spot Project (Su D., R.B.)

Summary (I)

* HLT Configuration
  - With the DB proxy technology, we solved the HLT configuration problem in a reliable, efficient and scalable way, leveraging on our expertise in online databases and network programming
  - We developed and deployed the MySQL<->ORACLE translator proxy as a stopgap for final commissioning and early data, which has spurred interest in the offline community
  - The CORAL Server project is ongoing as a joint project between SLAC and CERN/IT. It found support/demand by LHCb and Alice
* Partial Event Building
  - We have taken on a key role in the development and implementation of partial event building, which offers individual subsystems potentially large factors in calibration rates (statistics)
  - We are already working together with LAr, TileCal and ID to maximize their calibration yields and are preparing to include others

Summary (II)

* Online Beam Spot
  - We took the lead in providing an online beam spot measurement for ATLAS from the HLT, which will serve ATLAS monitoring, provide machine feedback to the LHC operators, and (difficult!) will also be fed back into the HLT for configuring beamspot-sensitive algorithms, such as *b*-tagging
  - This is a critical need that has simply been overlooked; is it also technically particularly difficult
  - Our algorithm work already exposed and addressed several issues with the HLT tracking
* Operations
  - We have been given TDAQ expert status and privileges, and are part of daily operations and expert response
  - SLAC has sole responsibility for the DB proxy configuration system; we actively participate in commissioning; and we expect to be there for first data!