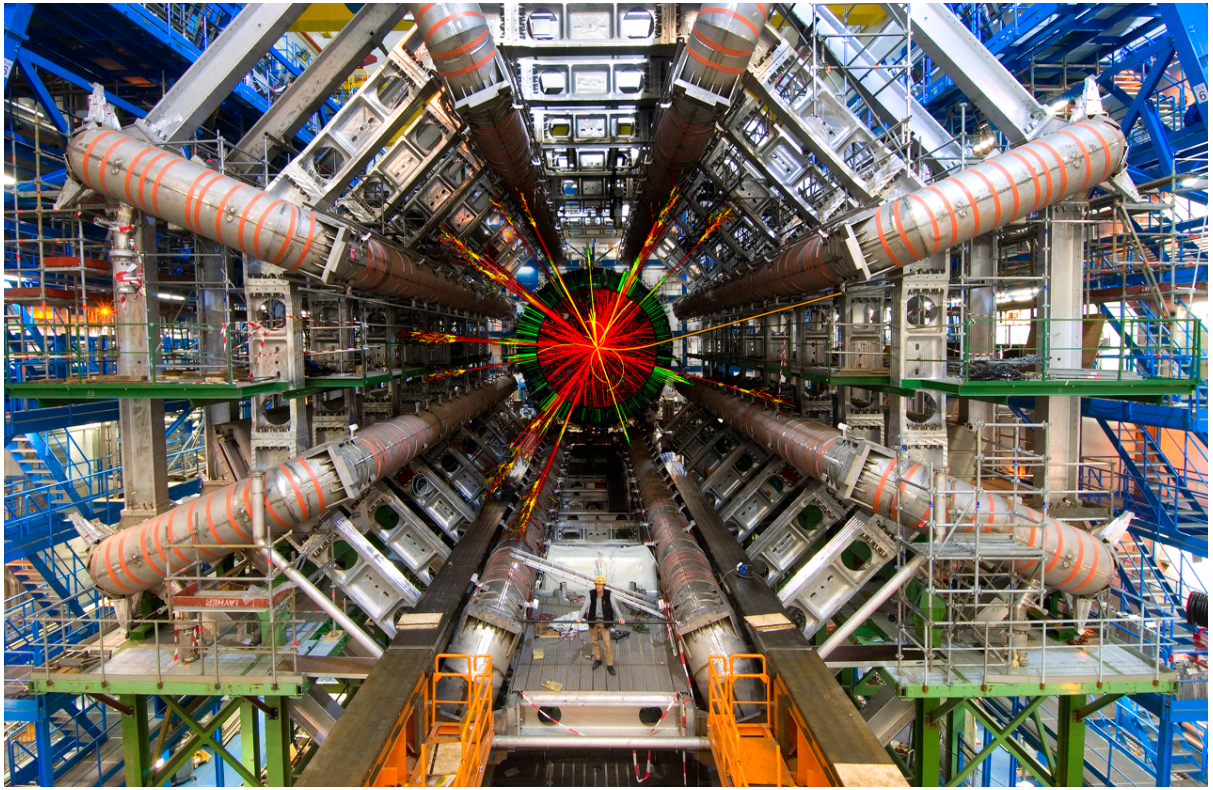
A Particle Physics Data Perspective



David Malon
malon@anl.gov

ICiS Workshop
August 2011

LHC experiments and their data

- Experiments at the Large Hadron Collider (LHC) are taking data now
- The ATLAS collaboration, for example, (approximately 3000 physicists from 37 countries) already has tens of petabytes of data in a globally distributed event store
  - "event" is what we call what happens when proton beams collide
  - LHC provides beams that collide to provide millions of proton bunch collisions per second, of which ATLAS records about 400 per second (a couple of megabytes of raw data per event)
    - Which 400 is based upon a complicated and evolving menu of "triggers"—must balance the physics agendas of a broad spectrum of research interests
  - Raw data and several stages of successively derived data, plus matching amounts of simulation
    - Not so different conceptually from Tim's LSST L1, L2, L3 data product model, except that L3 data clients (standard model physics, top physics, b physics, supersymmetry, exotics, …) are *inside* the collaboration, with the data product support implications that this entails
- It works. Good physics, impressive physics, is being done at the LHC.

But while it's big, isn't it an "easy" problem (computationally, not scientifically)?

- Processing is readily parallelizable, amenable to shared-nothing approaches?
  - Gigantic "for each event, …" analyses? (Well, not quite…)
  - "Just" throw lots of hardware at it?
- Even *almost*-shared-nothing event processing requires terabytes of auxiliary, time-varying data, much of which is database-resident
  - Different events may require different auxiliary data
- Big heterogeneous instruments (the ATLAS detector is really almost 100 detectors, for tracking and various kinds of calorimetry and muon spectrometry and more, further varying by geometry and proximity to the beam line and more) have interesting and complicated temporal database issues
  - For monitoring and configuration and detector status and collider status and calibration and alignment and conditions and …
  - Time-varying at very different granularities
- Almost fractal complexity of data provenance, and data versioning issues
  - Have you ever heard anyone say with a straight face that provenance is straightforward, or even easy?
- Data models are complex (adding array capabilities doesn't nearly suffice), and schema evolution requirements are substantial

LHC computing resources

- LHC computing relies on distributed cluster computing, integrated with computational grid software
  - LHC needs and contributions have driven grid development in the U.S. and Europe, and a great deal of experiment-specific infrastructure has been deployed as well
- National funding agencies have pledged resources (CPU and storage) at Tier 1 and Tier 2 centers around the globe
- Experiments also rely on potentially abundant but unpledged resources, especially for analysis
  - At your university, for example
  - Integrating such resources into the computing fabric is a challenge

LHC and leadership computing resources

- Community has not traditionally tapped leadership computing resources
  - Too data intensive, too many packages required, …
    - And imagine each node of a BlueGene/Q at Argonne attempting to connect to Oracle at CERN
  - For simulation, memory requirements per core are too high
  - But at some point these are excuses, and we need to define a program of work to leverage these resources and opportunities (and to influence decision-making about the next generation of leadership computing)
    - A few of us are beginning to try to do so
- If we were to produce 250 TB of simulation data at our local leadership computing facility and the center had the capacity and the willingness to host the data, would it allow access to 3000 ATLAS users?
  - To take a copy home, or
  - To run their 3000xN analyses where the data are?
  - Maybe not a show-stopper…

Files and databases

- Our raw data are in files, and so are our derived data
  - But they are decidedly not *flat* files
- Terabyte-scale metadata database for event selection
  - Key quantities describing each event, with references to event data at each processing stage sufficient for direct navigation to the correct data within the correct file
  - Scalability is a challenge even here
- We think your community must be better at databases than we are (SDSS, Millenium DB,…)
- Databases are used extensively—but not for event data
  - Temporal database infrastructure
  - Time-varying conditions and calibrations
  - Configuration
  - Trigger menus
  - Data quality
  - Luminosity
  - Lots of metadata at the run and sub-run and collection-of-runs level
  - …
- Plus the databases that support distributed data management
- Plus the databases that support distributed production

Storage management

- Storage limitations and storage management are a big part of operational decision-making and headaches
- Never enough: pledged storage constraints are a serious factor in limits on data rate
  - And if someone made 10 PB of space available to your experiment, but it was spread over 37 countries at centers of widely varying reliability and connectivity, would you be happy, or sad? How would you manage it?
- Sample issues:
  - What do we keep on disk? How many versions (earlier reprocessings) do we need to keep available and for how long?
  - Do we really need to keep processing stage N on disk: shouldn't people be using the derived data (stage N+1) for their analyses?
    - except for those two special use cases…
  - Can we increase group X's allocation for this special purpose?
  - How do we do replication, and control the lifetime of transient replicas? Demand-driven?
- And so on.
- Oh: and small files seem to be big headaches for us, too

Database access issues

- We work to minimize database access by physicists doing analysis
- A typical user task may be broken by the production system into 1000 jobs, each processing different files, each requiring access to database-resident conditions data
- Try not to let this trigger 1000 simultaneous Oracle connections
  - And recall how large ATLAS is, and how many users might be doing this
- A variety of strategies
  - Do database caching akin to web file caching
  - Build tagged database releases extracted into files for particular purposes,
  - Use hybrid strategies (temporal database used to determine which calibrations are needed, but calibrations themselves live in files, …)

Some issues we're grappling with

- Optimization of persistent data organization and I/O performance
  - Performance is everyone's highest priority for core software developers
  - Access pattern and access method dependencies
- Metadata, metadata, metadata
- Robustness: how does one deal with errors?
  - Inevitable at multi-petabyte scales?
- Increasingly-multi-core architectures
  - Even event-level parallel implementations need to share memory
  - I/O bottlenecks and serialization and merging results and getting the metadata right are keeping us busy
- How to take advantage of cloud computing
  - Harder for data hosting than for opportunistic simulation
- Should we be paying attention to NoSQL databases, key-value stores, …?
  - Exploratory projects underway
  - And is your community looking seriously at hybrid strategies, database support for *in situ* data, …
  - And does SciDB look promising for your use cases?

Related to Ryan's comments …

- We have coding rules for C++
  - If you're looking for a starting point, Google's rules are not onerous, by the way
  - We also have ATLAS framework-specific guidelines and "best practices"
  - And we run code checkers and leak checkers and so on
  - And automatic performance monitoring
  - Periodic domain-level package reviews
  - My group has a "two sets of eyes" rule for code inspection
    - Not ATLAS-wide, though
    - As much for cross training as for quality assurance
- Code check-in privileges are easy, but tagged code must go through an approval process prior to inclusion in a release
  - Package managers are responsible for monitoring sub-package tags
  - Tag approval meetings twice a week, including justification (not just "it passed developer tests")
- We do nightly builds and tests, and a huge array of larger-scale tests
  - Not just unit tests, but tests on data, including end-to-end tests
    - Full Chain Tests, Tier 0 Chain Tests, …
  - In addition to time-consuming physics validation for new releases
  - And still Bad Things happen sometimes