What Worries me in ATLAS and LHC Software

Workshop on LHC Software Readiness
NSS-IEEE October 26, 2005

Paolo Calafiura
LBL/NERSC-ATLAS

My (Selected) Concerns

Disclaimer: I am ***not*** representing ATLAS here

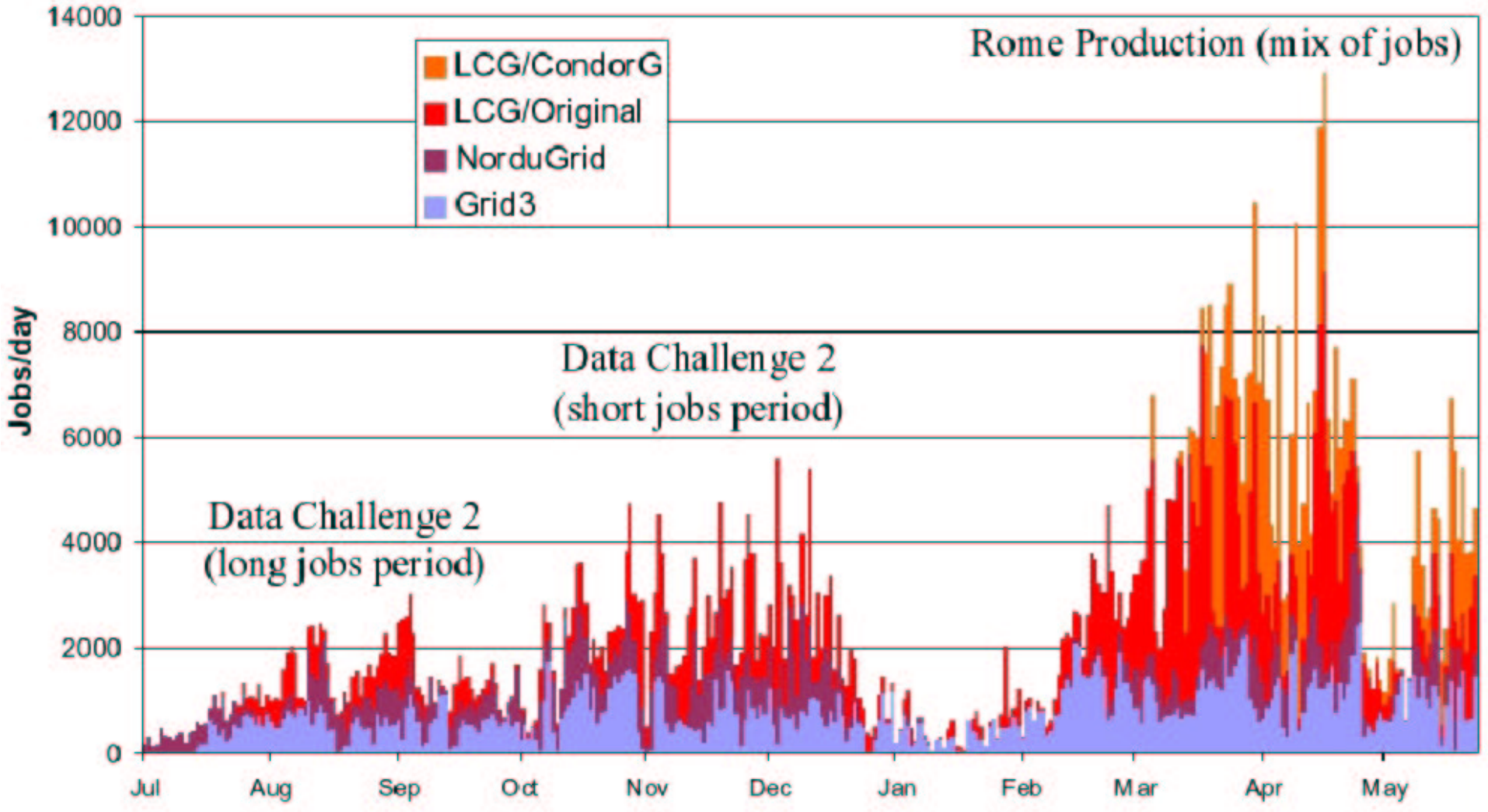
— but I'll try to report what I hear in the corridors…

- Production/Analysis System
  - — Limited integration with rest of ATLAS software
  - — Reliability/Scalability
- I/O: performance and schema evolution
- Software Usability:
  - — Complexity of job configuration
  - — Difficulties in setting up runtime & development environment

Production Systems

- Two major scalability problems observed during Data Challenges and physics productions
  - — Querying of the replica catalog
    - moving to a distributed data management system
      - — dataset rather than file-oriented
      - — designed to avoid repeated catalog queries
  - — Job submission times. Goal ~12K jobs/day
    - Moving towards an "agent-based" system a-la-LHCb in which "pilot" jobs are submitted and later activated from ATLAS-specific brokerage system.
    - Logging and bookkeeping could be made faster with tighter integration with Athena (ATLAS sw framework)
      - — python job control later
- Not enough coordination with rest of sw community

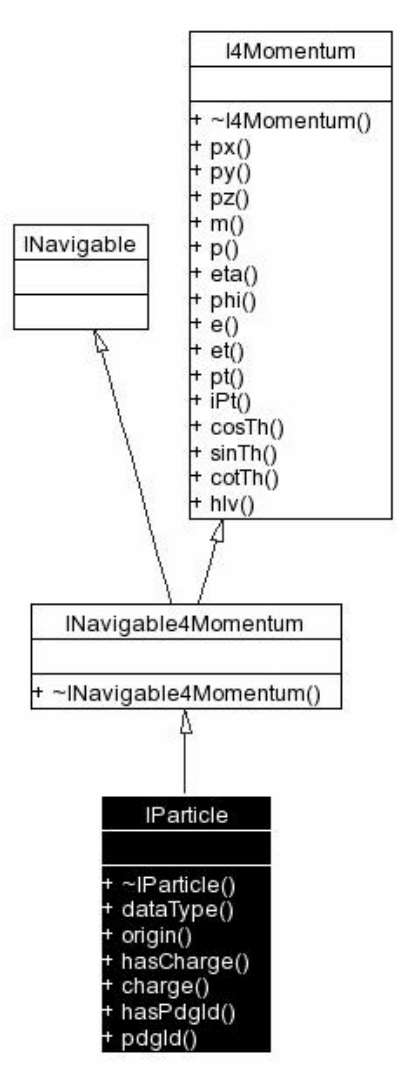
Average during Rome production ~3K jobs/day



I/O Issues

- AOD (analysis format) I/O performance.
  At least 10x slower than physicists requirements.
  - — Object creation is major factor
    - Complex inheritance structure
  - — AOD analysis written in terms of abstract interfaces
    - Pointers are a bane in POOL/ROOT
      - — No ROOT Tree splitting (aka column-wise access)
- POOL/ROOT automatic schema evolution does not work for complex objects.
  - — Works fine for struct-like objects
    - Introduce intermediate persistent objects when needed
    - Better Performance in “bulk” reading (20-30%)
      - — I suspect even better performance in histogramming mode
- Risk (esp with AOD): users may bypass transient EDM

IParticle


I4Momentum
+ ~I4Momentum()
+ px()
+ py()
+ pz()
+ m()
+ p()
+ eta()
+ phi()
+ e()
+ et()
+ pt()
+ iPt()
+ cosTh()
+ sinTh()
+ cotTh()
+ hlv()
INavigable
INavigable4Momentum
+ ~INavigable4Momentum()
IParticle
+ ~IParticle()
+ dataType()
+ origin()
+ hasCharge()
+ charge()
+ hasPdgId()
+ pdgId()

Intermediate State Details

- When an EDM class Foo changes in nontrivial ways, introduce an intermediate state representation object, say, Foo_p1.
- Use a custom converter to fill Foo_p1 from Foo on output and vice versa on input; let POOL/ROOT stream Foo_p1 automatically.
- The next time that Foo changes, create a corresponding new intermediate state representation object, say, Foo_p2, with a UUID different than that of Foo_p1 in its selection.xml file.
- Edit the custom converter to check the UUID of the pointed-to data, distinguish thereby whether the pointed_to object is a Foo_p1 or a Foo_p2, build the corresponding object, and fill Foo from it

Software Usability

Physicists complain that “Athena” is hard to use

— Good! It means they are using it!

- Hard to setup runtime environment
  - — cmt is powerful but developer-oriented
    - Too many details to provide
    - It takes an expert to dig oneself out in case of problems
  - — Lack of a user-friendly, robust configuration management tool is costing years of wasted manpower to ATLAS
    - The perfect LHC (HENP) shared project that never was!
- Hard to run a job “in batch”
  - — Lack user-friendly, generic mechanism to submit an Athena job on the GRID

Job Configuration

- Athena jobs are python scripts running c++ code.
  - — We could not have survived without scripting
    - Four detector configuration in next production cycle
    - Infinite variations in CTB simulation
    - Alternative reconstruction strategies (calos, tracking)
- Now with >10K lines of python we do need job configuration architecture (learning on the job…). Inputs:
  - — G4 Simulation configuration (Manuel talk N19-6)
  - — Tracking “flag-driven” configuration
  - — “Data-driven” configuration prototype
    - Users think about their jobs in terms of input data and desired results
  - — Mirroring of Gaudi Property objects in python
    - Strong syntax checking, delayed C++ object instantiation

Egoless Programming?

Shared development works: Gaudi, G4, [P,C]OOL

Competition is good, but only when followed by collaboration

- —ROOT/SEAL merge is a dream come true
- —Will same miracle happen with GRID software? And distributed analysis?

A bad sign: ILC software

- — three competing efforts (understandable)
  - Each one apparently starting from scratch!
    - — Not quite: one seems to be based on ROOT…

Credits

- Thanks to D. Malon, M. Nowak, P. Van Gemmeren