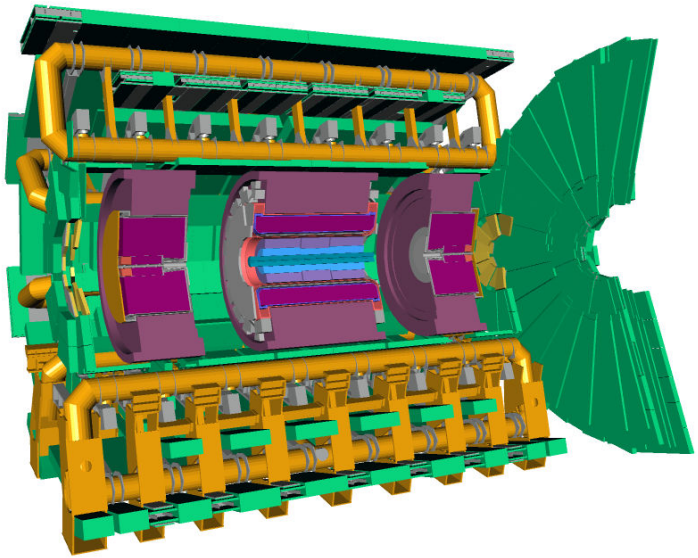
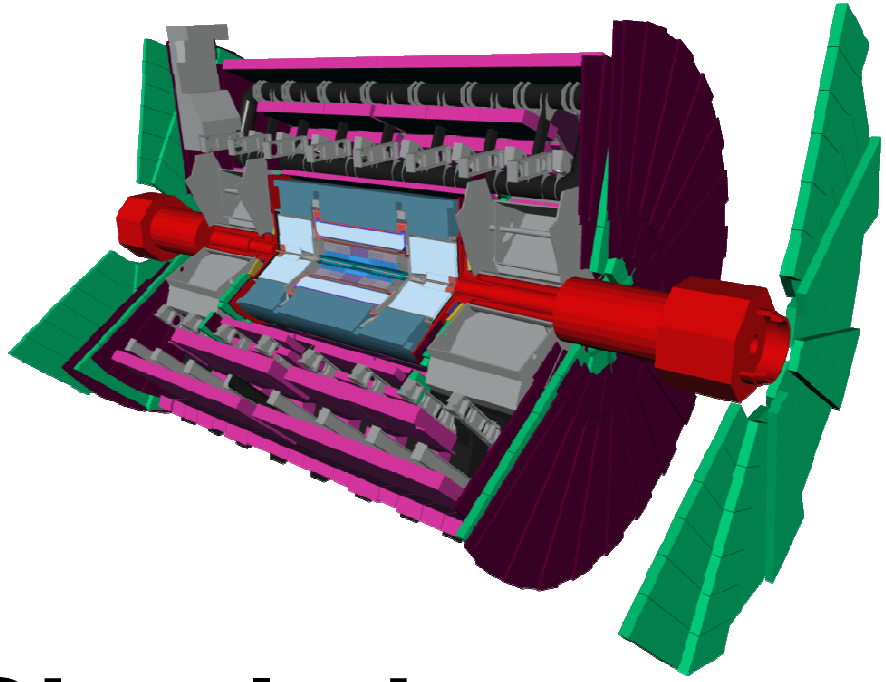
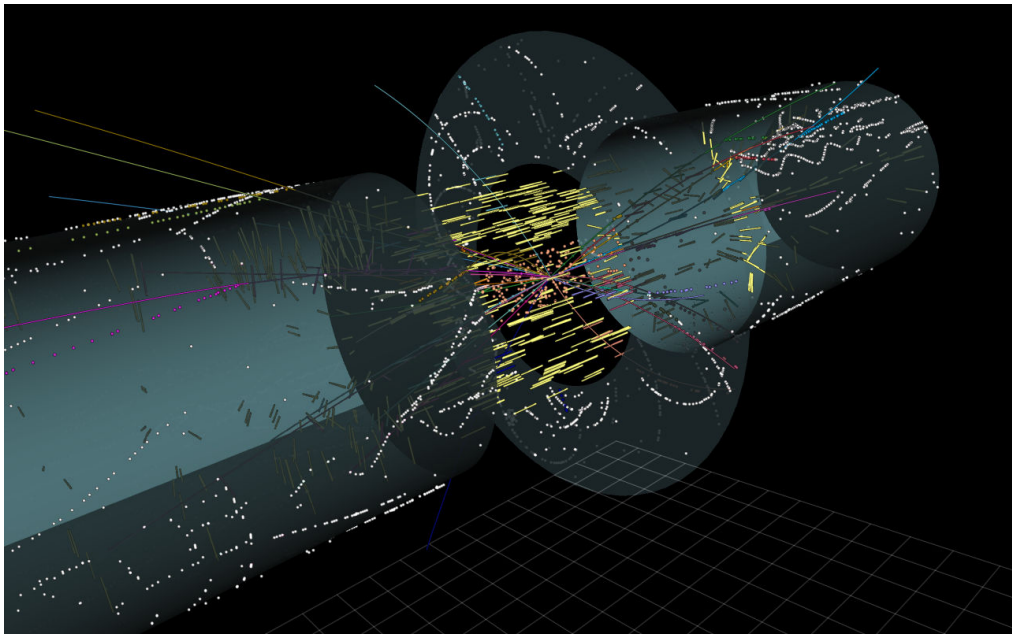




The ATLAS Simulation

Zachary Marshall
For the ATLAS Core Sim Group
Geant4 Workshop, Catania
16 Oct 2009

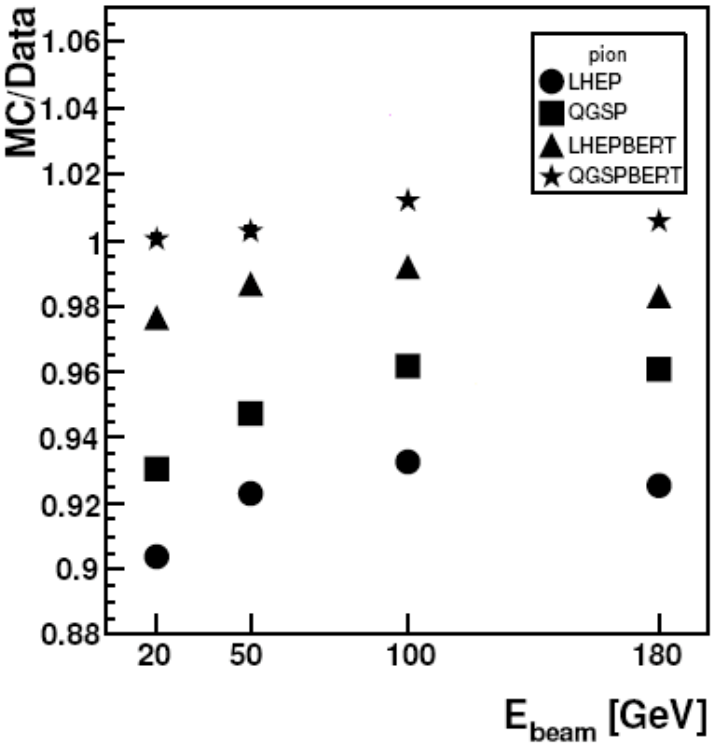


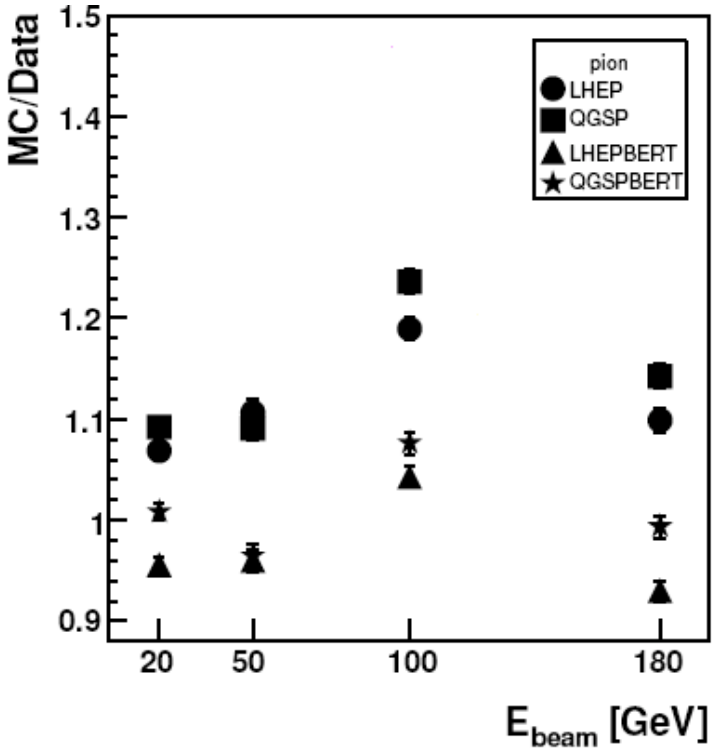
ATLAS Simulation Basics

- ATLAS has been using a Geant4-based simulation for many (~7) years now
  - Help from the Geant4 collaboration has been invaluable!
- In the last 18 months, ATLAS has
  - Moved from Geant4 8.3 patch 3 to Geant4 9.2 patch 2
  - Produced ~1 billion events on the Large Computing Grid
    - Another ~billion with fast simulations
  - 2009 production just launched (last week) with G4 9.2
- Out of the box, the ATLAS simulation uses
  - QGSP_BERT physics list
    - A neutron time cut at 150 ns makes this less costly
  - Most of the Geant4 defaults for other parameters
    - G4ClassicalRK4 stepper, 1mm range cuts (except in a few places), standard voxelization parameters, no step limitation
    - Several of these are now being optimized

Why QGSP_BERT?

- Comparisons with testbeam data indicate hadronic showers are best described by Bertini
  - Below, pion shower energy deposition mean (left) and RMS (right)



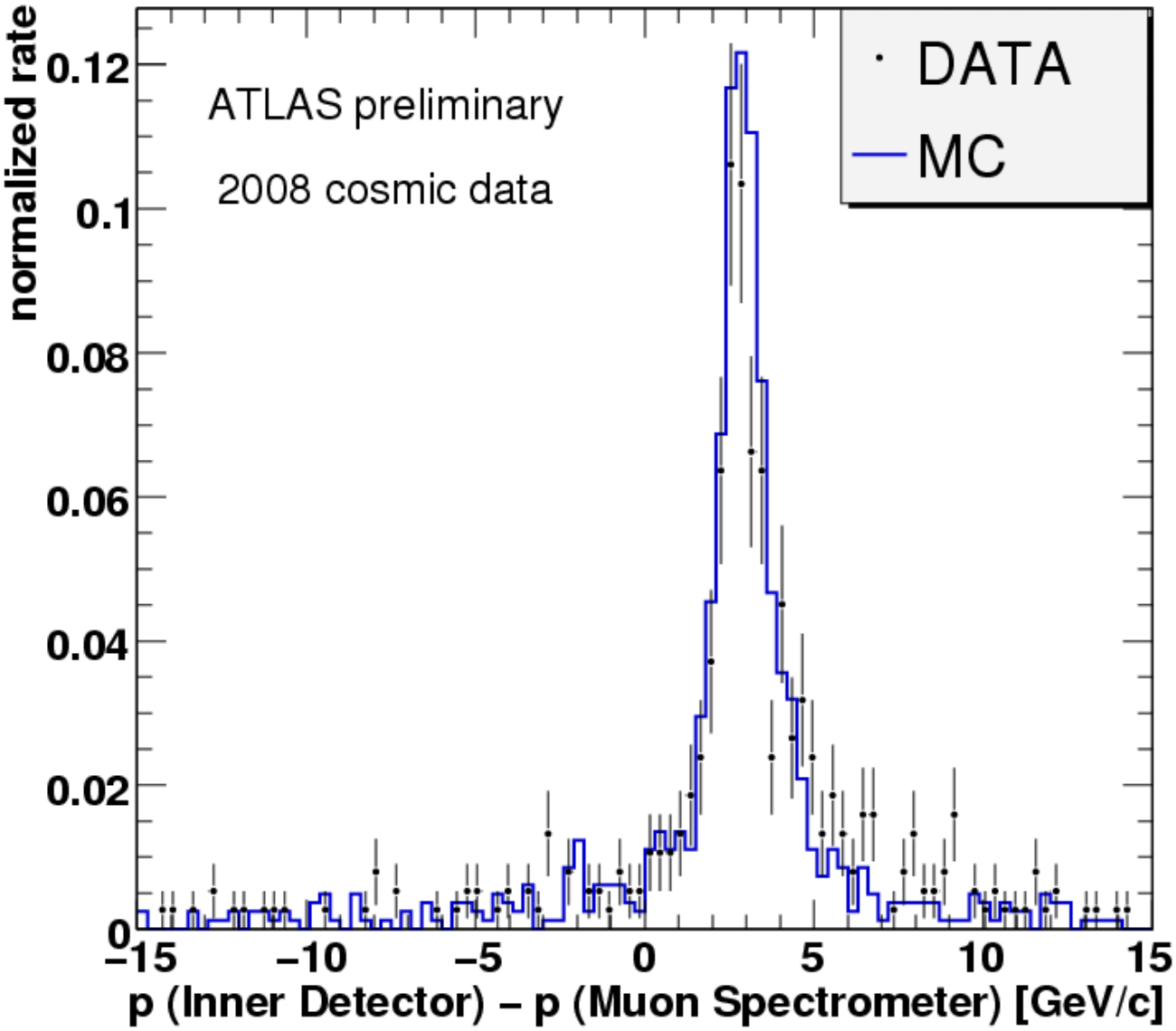


- Inner detector favors 'new' multiple scattering to get energy cluster sizes correct
  - EMV is not an option, even though it is faster

What Physics Can We Do?

- While we wait for the LHC, ATLAS has been collecting cosmic ray data
  - 10's of millions of projective events
  - Very useful for alignment studies - and we can even do a little physics with them (charge ratio, for example)
    - Also provides tests of our trigger and reconstruction software, object ID (we had better find muons…), etc
- So ATLAS has added cosmics to the simulation
  - *Not* part of the design, but it does *surprisingly* well
  - About 10M simulated events so far, another 50M on the way
- Took some time to learn what needed to be changed from 'standard' simulation to 'cosmic' simulation
  - Some details are omitted: atmospheric showers, elevator shafts (main shafts are there)
  - Some detectors behave differently
    - Cosmics do not come on a 25 ns clock like LHC collisions will!

Cosmic Ray Comparisons (I)



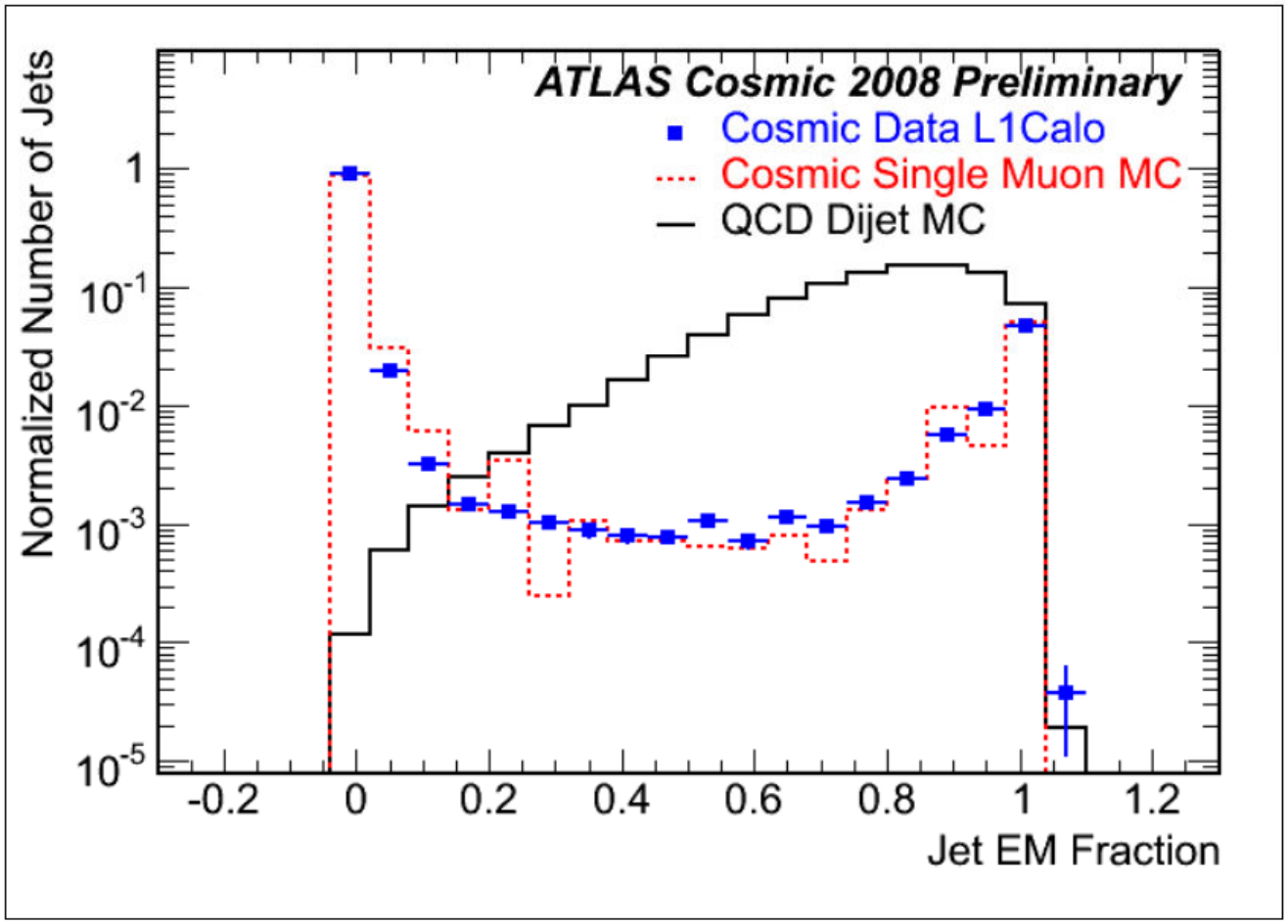
Cosmic ray energy loss in the calorimeters

Difference in momentum as measured in the inner detector and the muon spectrometer

Both have means of about 3 GeV

Cosmic Ray Comparisons (II)

Jets are in cosmic ray data too!



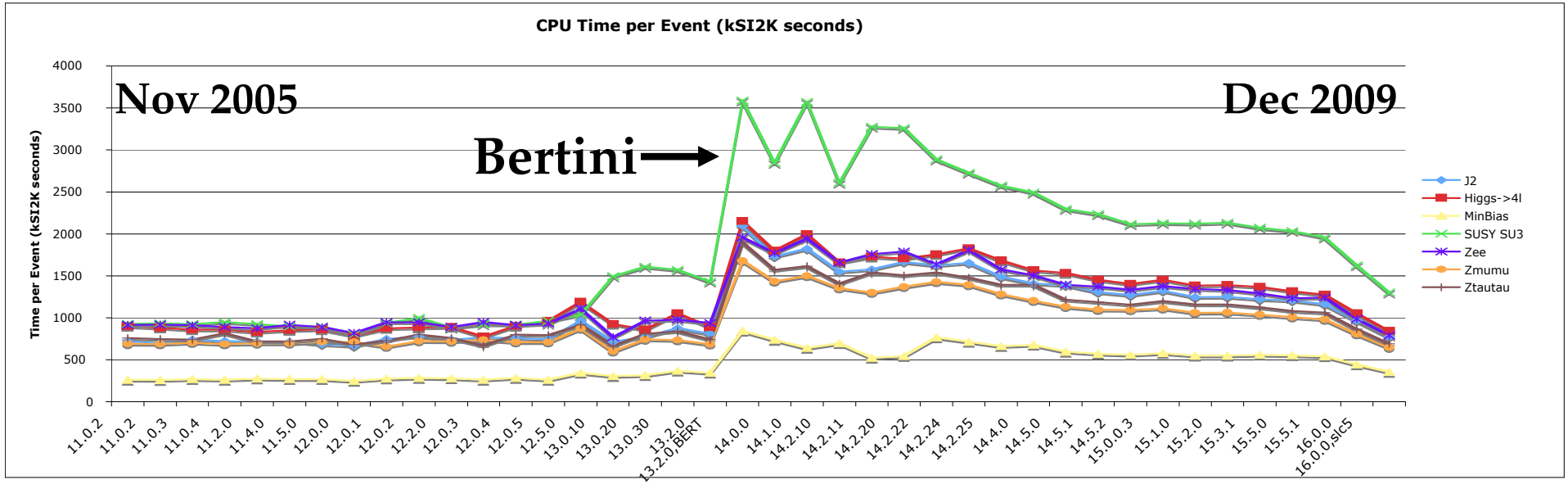
Fraction of jet energy in the EM calorimeter

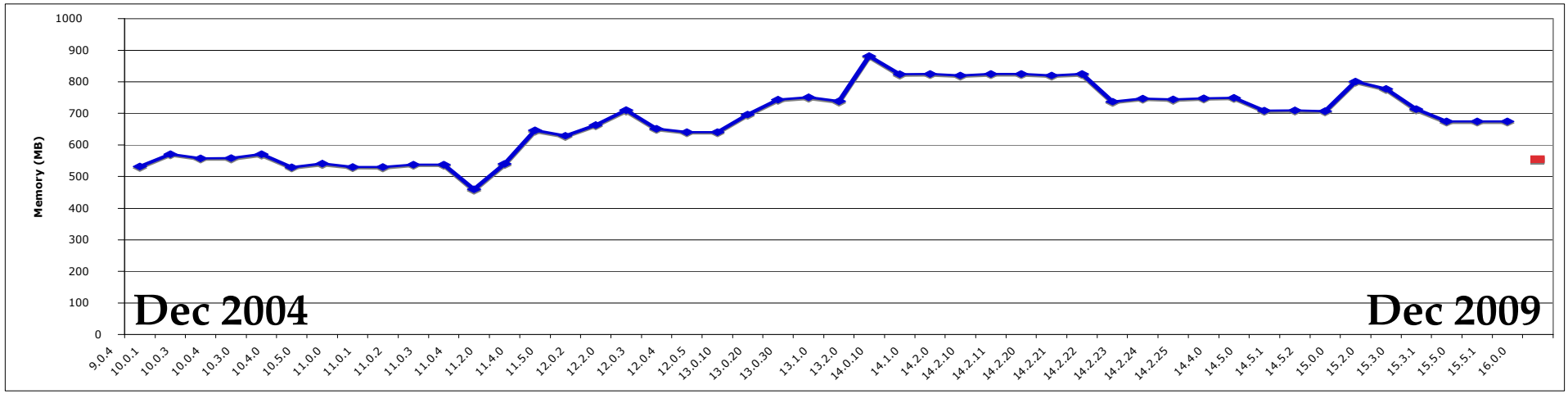
Cosmic ray data and MC have good agreement

Collision data will be something rather different…

Software Performance

We've been monitoring our performance for quite some time now…





Where Does The Time Go?

- Code profiles don't show any other hot spots, once the magnetic field is 'solved'
  - Also seems that parallelizing (or GPU processing) are a lot of trouble for not a lot of gain
    - Remember that it's easy to run N jobs on N cores already…
    - And on the Grid we have 2GB of memory per core, so that's not a problem in production (no need to try to 'share')
- Complicated geometry description in ATLAS
  - ~40% of all steps are 'just' transportation
  - Could try to simplify the description, but could cause a *lot* of trouble that way
- Complicated calorimetry
  - >90% of all steps in the calorimetry, most on EM particles
    - 40% transportation, 40% multiple scattering
    - Should be improved somewhat by the stepper dispatcher
- So as long as we're working on reducing that…

Coming Improvements

- Lots of work recently to improve performance
- Several new boosts for CPU performance
  - slc5/gcc43 is 20% faster than slc4/gcc34 !
  - New 'stepper dispatcher' can choose stepper and parameters based on particle type, detector region, energy, step length…
    - Reduces dramatically the 20% of our CPU time spent accessing the magnetic field
    - Also reduced by a new implementation of Runge-Kutta (15% performance gain to be confirmed with physics validation)
    - May also add field caching (via G4CachedMagneticField)
- ~150MB reduction in VMEM (total memory use)
  - Changing voxelization (geometry optimization) parameter in several very large volumes: 50-70 MB reduction, no CPU cost
  - Little tricks: light tcmalloc version (7-10MB), light Oracle version (20-40MB), floats instead of doubles (15MB), reducing cross section table size to 7 TeV (5MB), ROOT file writing options (5-10MB)

New Performance Profiling

- Recently started looking at alloc/dealloc per event
  - Frequently points to code that needs optimization
- Immediately found problem in G4String::operator==
  - Creating temporary objects - 600MB/event in ATLAS!
  - Quickly solved in both G4 and ATLAS code
- Two problems are apparent
  - Bertini is translated Fortran (1GB/event)
    - To be rewritten, look forward to testing this soon
  - G4Transportation::PostStepDoIt (2GB/event)
    - Have to be more clever about how to 'fix' this…
- Also attacking ATLAS code
  - Trimmed 400MB/event with simple code optimization…
  - 1.3GB/event with 'calibration hits', usually off in production
    - Resulted in a 10% CPU boost - if this scales, and we can get 2GB out of Geant4…

Stability and Bugs

- Production system performance is the key
  - Jobs retry automatically on the Grid, so crashes can be costly
- Crash rate for the year below 0.1% of jobs
  - <1 crash per 5000 events
  - Several non-critical issues found thanks to valgrind
    - Weren't causing crashes, but better to have them fixed…
- Only two crashes in main production these days
  - One undefined nucleus in hadronics, fairly rare
  - "Stuck tracks," tracks bouncing between two volumes and taking tiny (picometer) steps - no crash, but job won't stop!
    - Several solutions proposed to solve this issue
    - Bigger problem on the grid, where these waste 2 days of CPU
- Generally, we're very happy with the stability of G4
  - Which is why we want to stick with G4 9.2 through 2010!
  - Will also test new G4 versions to ensure they are as good

Looking Forward

- We are getting analyses ready to attach LHC data once it is available
  - Of course, good performance is crucial to have a fast turn-around time once we have new things to try!
- First: we *must* get the detector description right!
  - Already much progress purely from weighing detector components
    - Pixels are a bit light (in the services), but most of the ID is in good shape. Muon system still needs more dead material, but hundreds of tons have been added recently.
  - Will work from photon conversions in the inner detector
- After that, shower shapes in the calorimeters
  - This is where we'll need the G4 collaboration's help most
  - QGSP_BERT looks pretty good in test beam, but we need to exercise several models and get handles on systematics!

Summary and Conclusions

- ATLAS's Geant4 simulation is in good shape
  - Years of testing for stability and robustness
  - Many new performance improvements coming soon
- Physics descriptions are already behaving well
  - Testbeam was the first major test
  - Even cosmic ray data looks (unexpectedly) good
- Production on the Grid is ongoing
  - 1 billion full, >1 billion fast simulation events in 18 months
  - Just last week, launched the official 2009 production that will take us through first data taking
- We are all very much looking forward to testing our G4 simulation with LHC data this year!!
  - The talk at the next workshop should be an interesting one!!!

Backup slides

Charged Particle Transport

- Several familiar issues with transportation
  - G4Transportation alloc/dealloc lots of memory
    - In ATLAS, >2 GB per physics event
  - Default stepper (RK4) accesses the B-field 10x / step
- New solutions to old problems
  - New stepper from ATLAS re-implements RK4 with fewer field accesses. Should be the same otherwise.
  - New G4CachedMagneticField adds field caching in G4
    - Suggest those be included in geant4's next releases(!)
  - Alloc / dealloc in G4Transportation an open issue
    - When we have solved the bugs, perhaps this could be attacked (again) with an eye towards optimization?
    - In particular G4Transportation::PostStepDoIt()
- Also: how is parallel navigation these days??

Stability

- Two open bugs from our point of view
  - All things considered, that's not too bad…
- Crash in hadronics from a corrupt(?) nucleus
  - Seems new to me

```
*** Geant4 Hadronic Reaction Information ***
Process:  , Model:
Nucleus A, Z = 0 0
Projectile was a
```

- 'Stuck' tracks taking tiny steps
  - Probably an old story that we didn't pay close enough attention to
  - Tracks taking millions of very small steps, bouncing between two volumes / materials
    - But taking big enough steps so as to not trigger any of the 'stuck track' mechanisms built into G4Transportation
    - As a result, no warning from G4 - job just continues as long as it is allowed to (may recover after a very long time)
  - Still under investigation by both G4 and our groups
    - But important enough that I'll tell you more about it…

Stability (II)

- First attempted solution (increasing tolerance for zero size steps) resulted in G4 abandoning those tracks - so we found out that G4 can abandon tracks!

```
WARNING - G4PropagatorInField::ComputeStep():
 Zero progress for 51 attempted steps.
Proposed Step is 1.802421668501153e-10 but Step Taken is 1.802421668501153e-10
For Particle with Charge =-1 Momentum=0.08665330623096662 Mass=0.51099891
 in the volume Muon::ToroidShieldingInnerPlugs
```

  - Some guidance on setting tolerances would be helpful
  - Request a higher default 'warning' energy - 250 MeV is a lot!
  - ATLAS will test with a lower cutoff as well
- This is a rather serious problem in our prod. system!!
  - A track stuck like this can run a job to the CPU limit on the Grid, causing the job to fail
  - The jobs are automatically retried many times in order to allow for transient (Grid teething) failures
  - Our typical jobs take ~1/4 of the grid limit in time, so a stuck track in 1% of jobs (=1/5000 events, or 1/~2x10^{11} steps) means a **100% increase in CPU for the task!**
  - Current rate between 0.1% and 1% (depending on job type)

Physics

- A few problems/new ideas uncovered recently
- Constructing EM processes in a vector
  - Saved us 3.5 minutes/job in initialization time
  - No apparent memory or CPU cost
  - Request it be included in future G4 releases
- Bertini physics model allocates/deallocates quite a bit of memory (1.5 GB/event for ATLAS) - translated FORTRAN
  - To be rewritten, but some changes already patched
  - Look forward to having new code to try!
  - A recent reduction on the ATLAS side of 1.3 GB/event alloc/dealloc saved us 10% of our CPU time…
- New-ish problem with accessing indices out of range in hadronics (pointers, so no crash)
  - Not clear if this is causing any serious problems yet
  - How much valgrind-like testing is done by the G4 team?