

HEP Code and Lessons

Tom LeCompte

*High Energy Physics Division
Argonne National Laboratory*

*Computation Institute
University of Chicago*

Before We Take Off

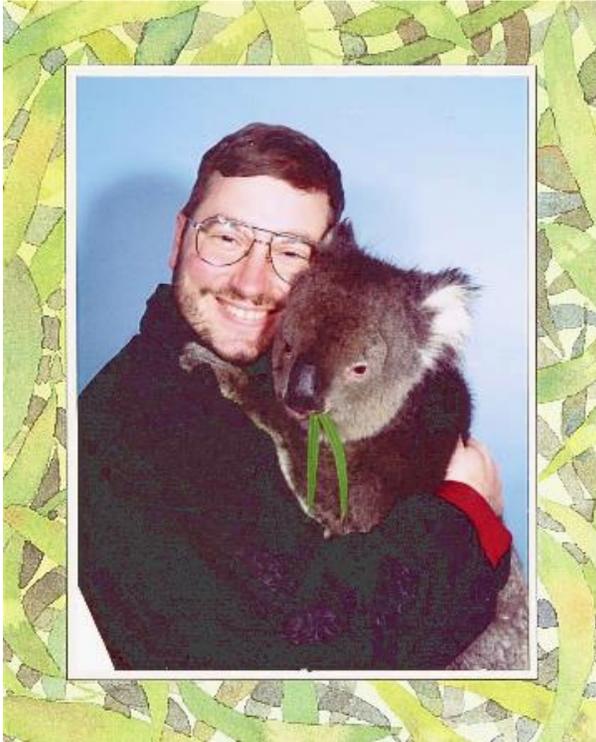
STOP ME if I go too fast or you have questions!!



I know I talk too fast, so please interrupt me – my goal is not to cover as much material as possible: it's to *uncover* as much material as possible



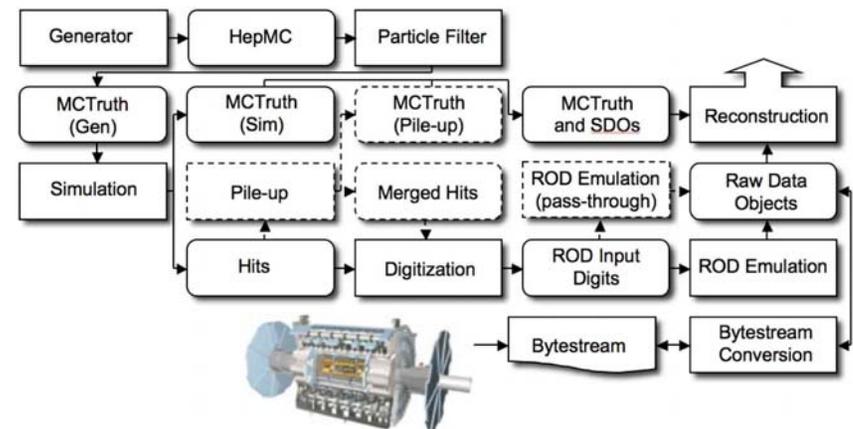
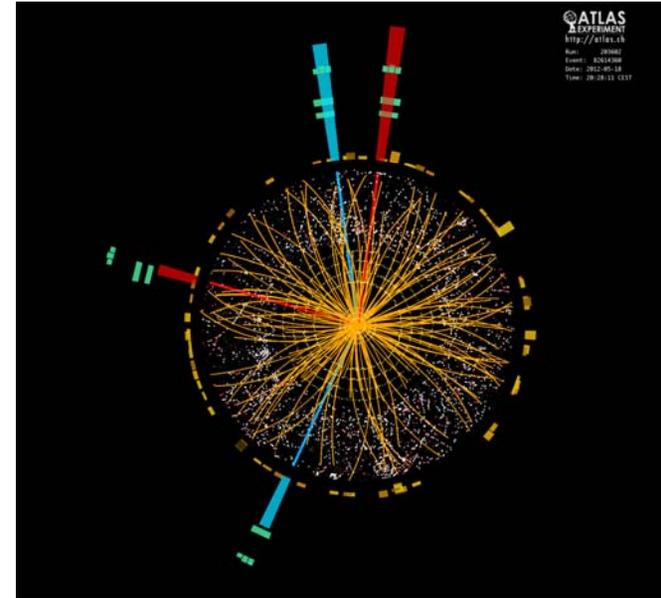
Who Is This Guy?



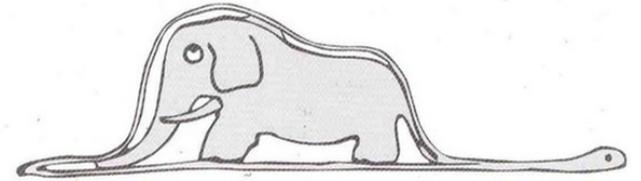
- This is a picture of a small, round-faced, adorable creature.
- Holding a koala.
- I am a high energy physics experimentalist, concentrating on hadron collider experiments: colliding counter-circulating beams of protons and other atomic nuclei and looking at the outcome of these collisions.
- I have worked at pretty much every such collider:
 - CDF at the Fermilab Tevatron
 - STAR at Brookhaven's RHIC
 - ATLAS at CERN's LHC
- Recently I was the physics coordinator of ATLAS
 - One of two experiments that discovered the Higgs boson in 2012
- I am very much a “domain scientist”.

Collider Physics for Non-Physicists

- We collide particles together and measure the energies and trajectories of the products in our detectors
- We then compare these results with simulation – at multiple levels
 - Does the detector respond to these particles in the way we expect?
 - Do we see the number of particles in various categories that we expect?
 - Etc.
- This is a complex process, and *we are as dependent on the simulation chain as we are on the data chain.*
 - ATLAS uses about a billion cpu-hours per year on this.



Quick Outline

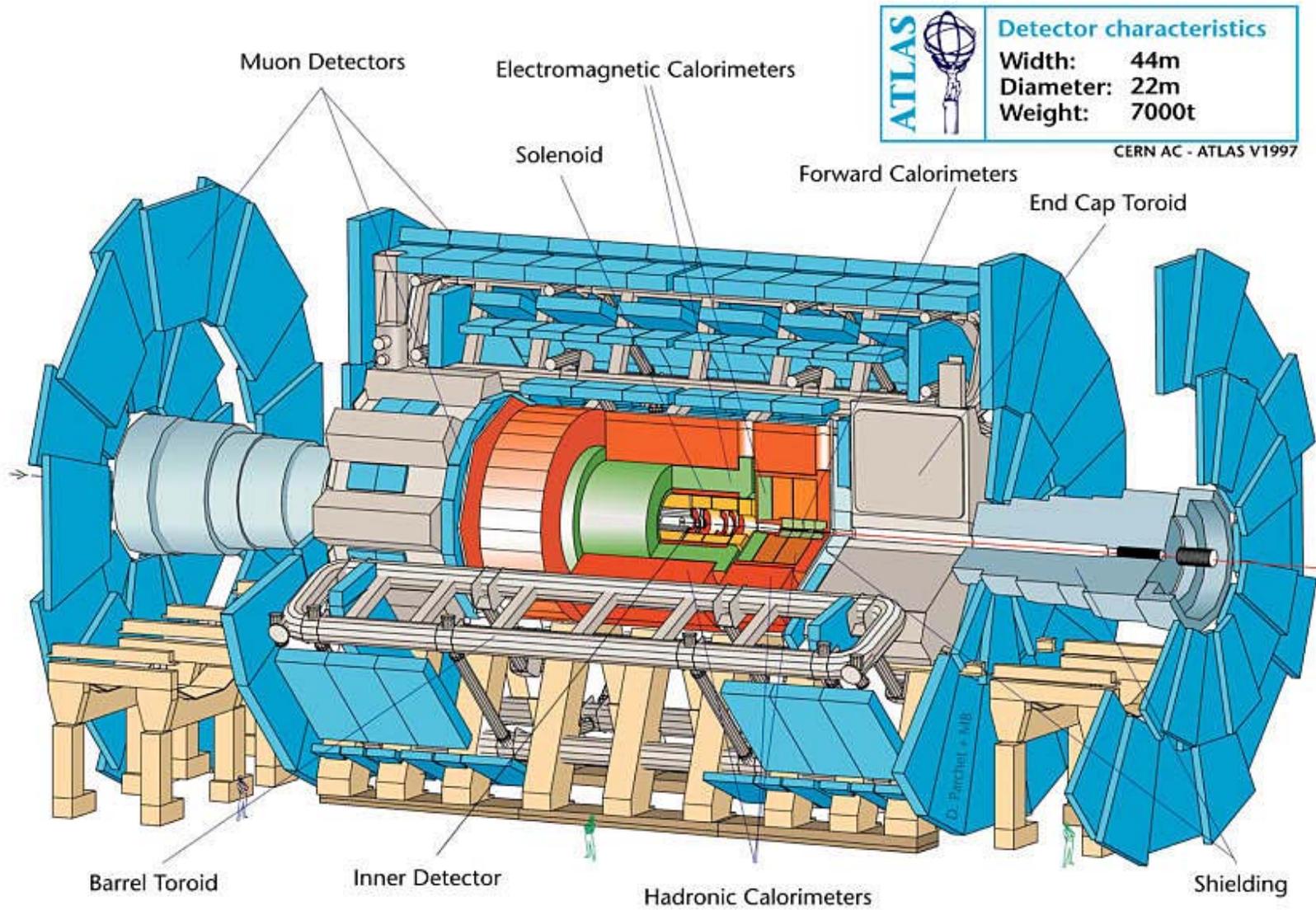


- What problem are we trying to solve?
 - A description of High Energy Physics
 - Where the computational limits are
- Where are we on the road to solution
- What the future might look like

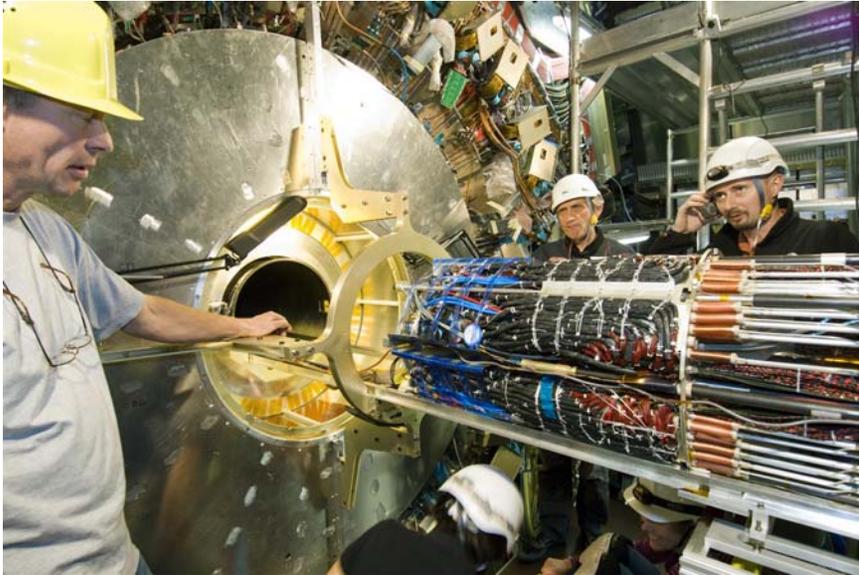
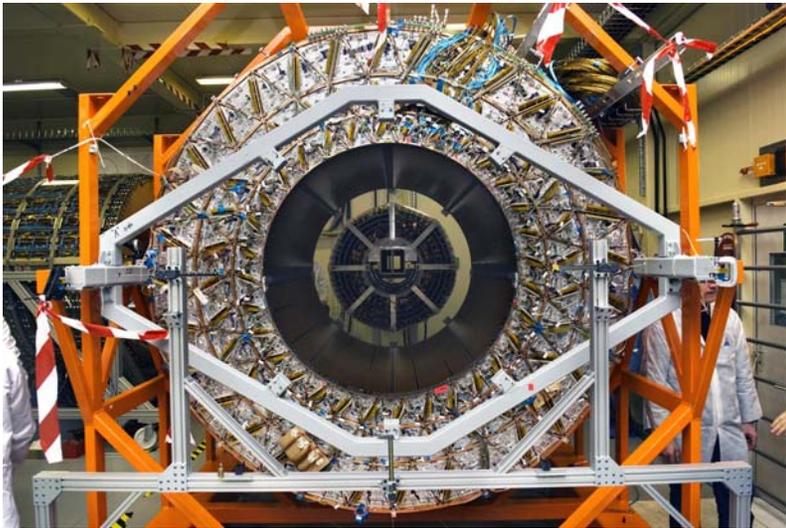
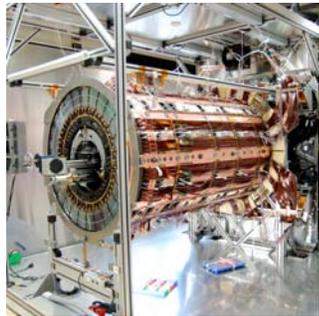
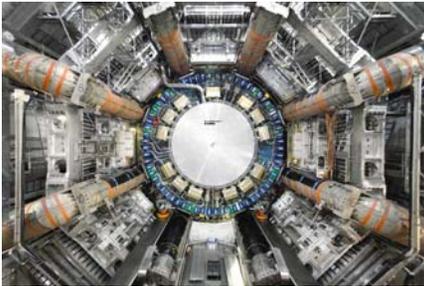
I was asked to focus on “performance and portability”



Tools of the Trade



What It Looks Like In Real Life



Science Output

- Discovery of the Higgs Boson in 2012
 - Led to the 2013 Nobel Prize for Peter Higgs and Francois Englert
- A total of 418 papers on both searches for new phenomena and precision measurements (growing at 1-2 papers/week)



One Higgs Boson Event - $2\mu 2e$

$$M_{12} = 76.8 \text{ GeV}$$

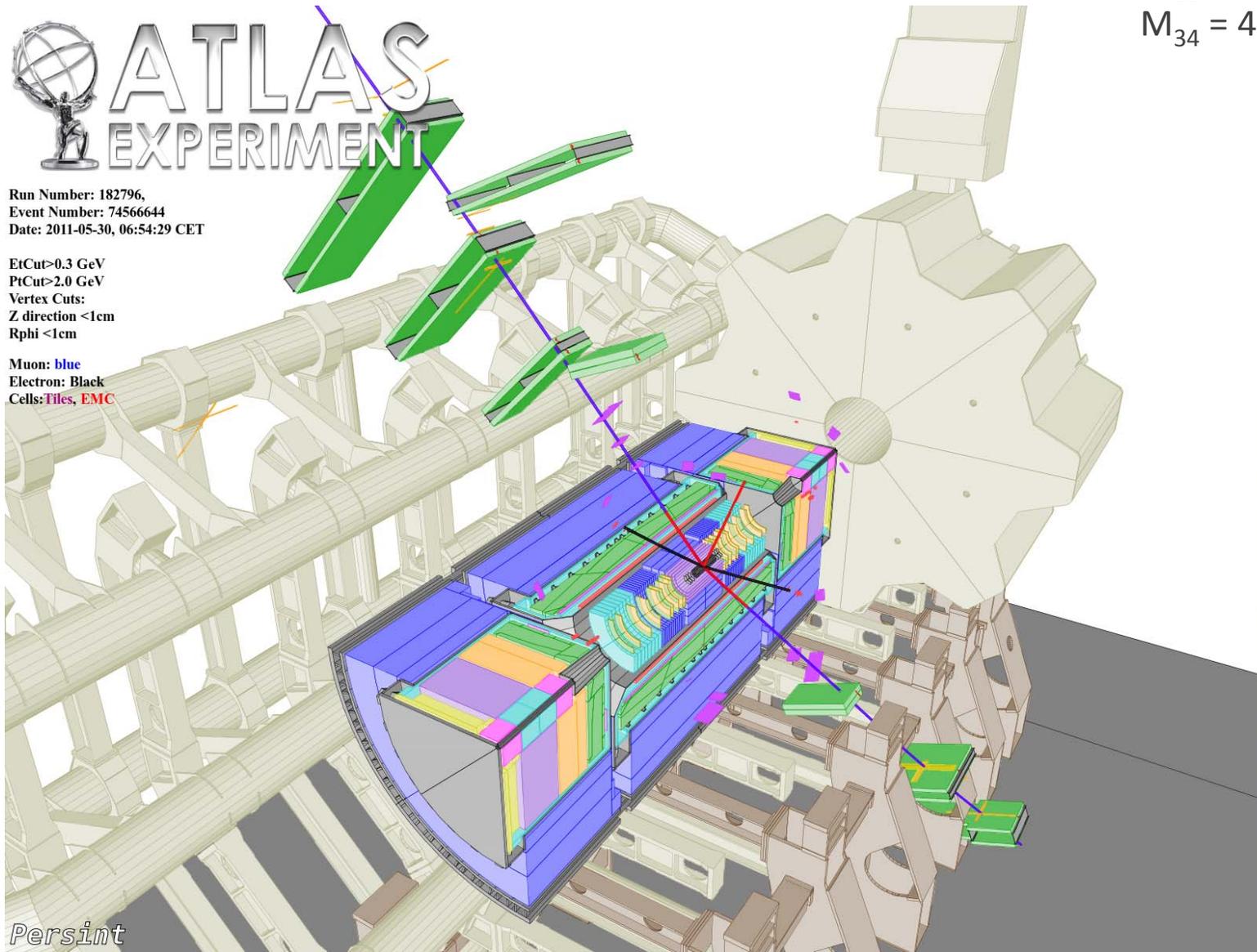
$$M_{34} = 45.7 \text{ GeV}$$



Run Number: 182796,
Event Number: 74566644
Date: 2011-05-30, 06:54:29 CET

EtCut > 0.3 GeV
PtCut > 2.0 GeV
Vertex Cuts:
Z direction < 1cm
Rphi < 1cm

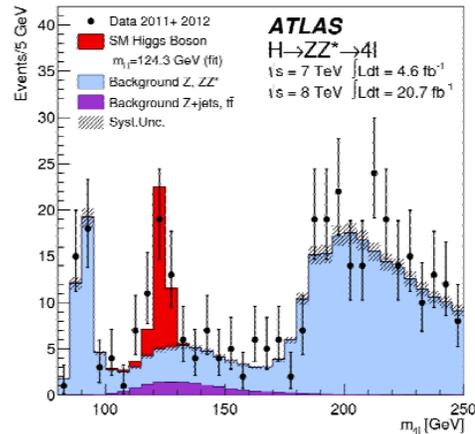
Muon: blue
Electron: Black
Cells: Tiles, EMC



Persint



Future LHC Science Goals



- Discovery of the Higgs Boson was nice. But...
 - We don't understand why it's mass is what it is: it's too light to be heavy and too heavy to be light. This is suggestive of other particles lurking just beyond present sensitivity.
 - We don't know if this is the only Higgs boson, or if there are more.
- We don't know what Dark Matter is, other than it makes up $\sim 25\%$ of the universe. But...
 - It sure would be nice to be able to make some in the laboratory.
- To answer these questions, we would like to collect $\sim 100x$ the data at close to design energy (13-14 TeV, rather than the 7 & 8 TeV we have)



How It Works

Simulated Data Chain

Event Generation



Simulation

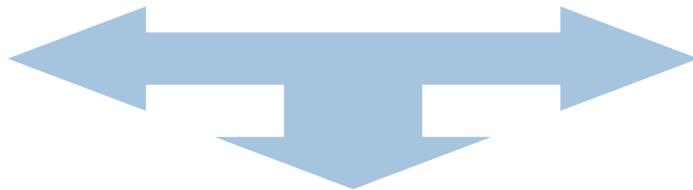


Reconstruction

Simulates the physics process of interest: produces lists of particles and their momenta

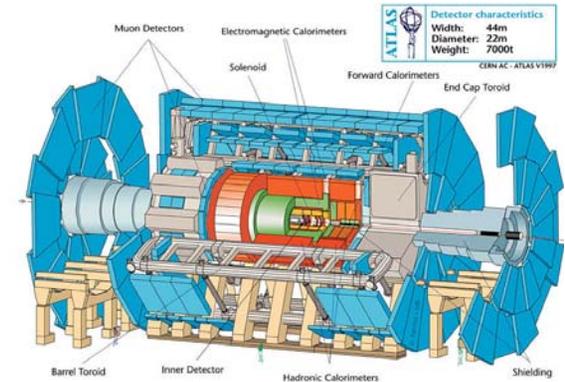
Simulates the interaction of these particles with the detector

Infers particles that must have been present based on the detector response



Analysis: comparing the two

Real Data Chain



Reconstruction

The Slide That Doesn't Fit Anywhere Else

- The key element is the “event” – in principle, a single proton-proton interaction. In real life, there are ~20 uninteresting events that occur at the same time (we call this “pile-up”)
- We write 600 events (each ~2 MB) per second.
 - 10^7 (running) seconds in a year → 6 billion events/year
 - Simulation needs a comparable number → 2 billion events/campaign
- It's natural to assign one event per rank in everything we do
 - The single event becomes the quantum of computing
 - “Embarrassingly parallel”, “Pleasantly parallel”... (I've had good luck with “pre parallelized”)



Computing to Reach The Science Goals

- ATLAS uses about a billion CPU-hours per year on the Grid
 - This does not include the cycles spent calibrating or reconstructing the data; the problem is defined as what happens after this point
- “The Grid” is the Worldwide LHC Computing Grid
 - 140,000 Xeon-class cores
 - Distributed in ~100 farms
 - 2 GB each
 - Jobs are single-core (~12 hours)
- Event Generation
- Simulation
- Reconstruction and Analysis



Summary of the Problem We Are Trying To Solve

- We have 16 million lines of code – **code we are not allowed to break**
 - About 45% is ATLAS code
 - The other 55% is common HEP code
- All 16 million was written under the assumption that computing is expensive and memory is cheap
- Most of the 16 million expects to be run single-threaded
- The needs of the experiment are growing exponentially
 - Every 3 years the data volume triples (4 triple-ings planned)
- The Grid is not
- The computational complexity of the events is also growing
 - There are jobs that don't run on the Grid – they run out of CPU

This is why we are interested in high performance computing, despite the non-optimal nature of our code.



Alpgen

- Alpgen: “a generator for hard multiparton processes in hadronic collisions”
 - An event generator (see <http://mlm.web.cern.ch/mlm/alpgen/>)
 - Simulates the physics process of interest
 - Produces lists of particles and their momenta
- Written in FORTRAN77 (with optional F90 part)
 - About 250K lines of code
- Runs in 3 phases
 - “Warmup” – pre-calculates phase space integrals
 - “Event generation”
 - “Unweighting”
 - A 4th non-Alpgen “afterburner” phase is needed to interface with the rest of ATLAS code
- Makes up ~5% of ATLAS Grid computing
 - Hoped to do 40% of this: 2% is big enough to be interesting, but small enough that the experiment will not collapse if we failed.

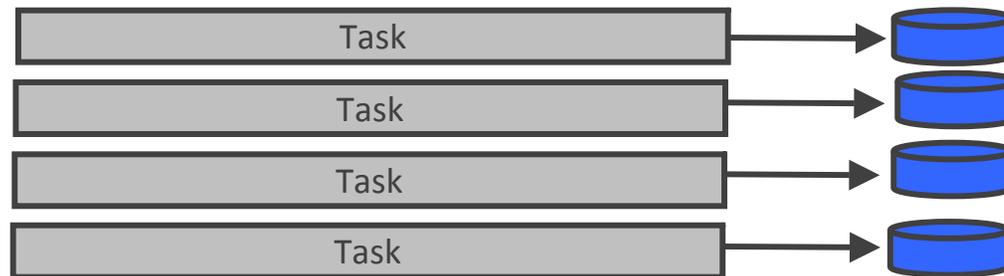


Getting Alpgen to Run on Mira

Start with a single Alpgen Process, and get it to compile



The first step is to run multiple copies of the same task

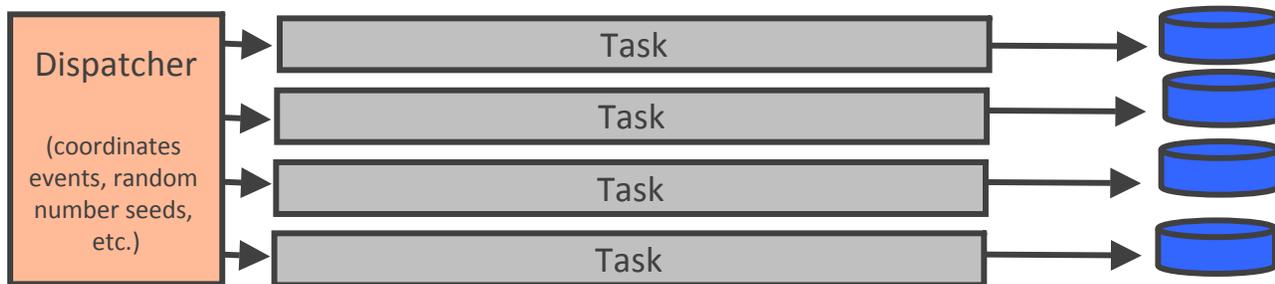


The problem here is that you get multiple identical output files – not very helpful.



Getting Alpgen to Run on Mira

We need a dispatcher: some MPI code to coordinate the work – in particular, giving every rank it's own random number seed



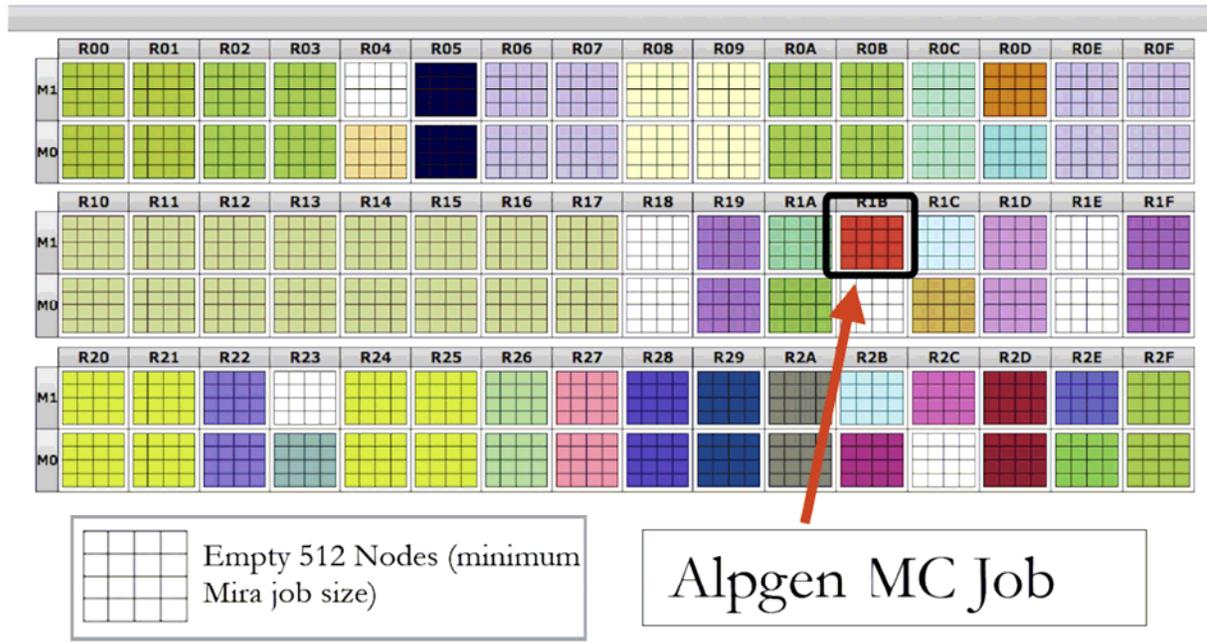
- This alone will run in a minimum Mira partition
 - 512 nodes x 32 ranks per node
 - It is, however, severely I/O limited (more on that later)
- We are using up random number seeds 16000x faster than grid jobs
 - Requires some coordination with the experiment



Where we were 1 year ago

- We could run in the minimum Mira partition (and only in the minimum efficiently)
- Event generation rate was 1/15 of a Grid node
 - ALCF suggests a nominal of 1/10

Mira Activity



At this point, we are limited by I/O.



Getting to 786432



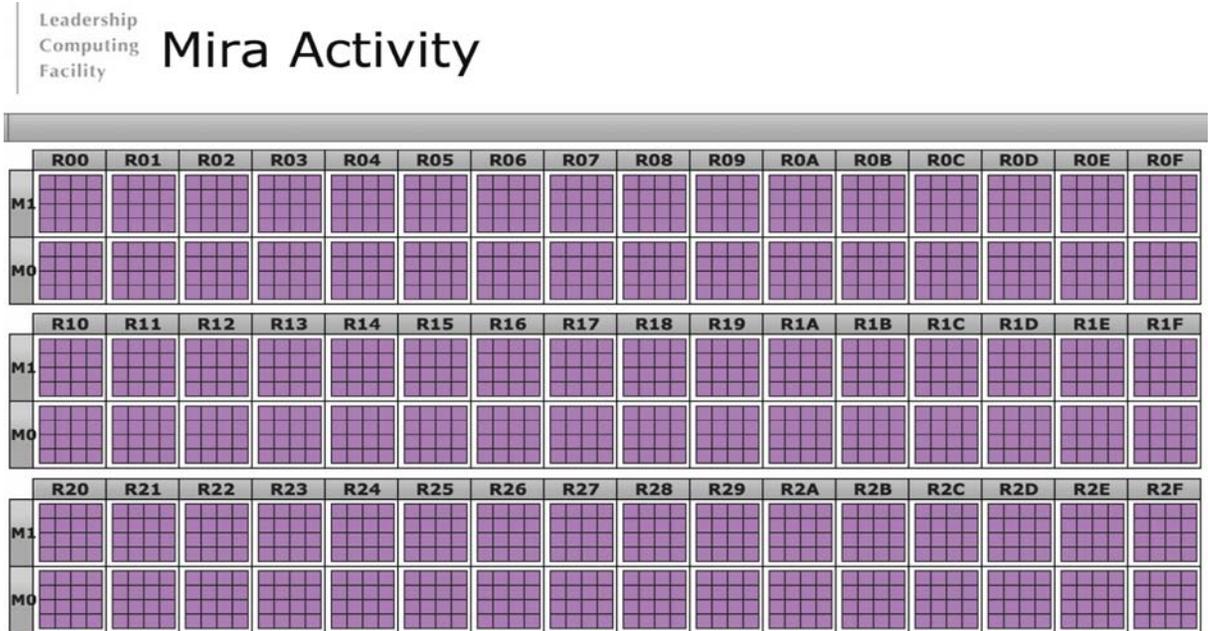
- Major changes to the code were impossible
 - It's not our code
 - It's been validated by the experiment – major changes would break that
- Memory footprint was reduced
 - Proton models (“PDFs”) except the one in use were no longer loaded into memory
 - Allowed us to fit into “-c64 mode”: ~180 MB per rank
- Output was moved from the file system to a RAMdisk, and collected later
 - This means there is a certain tension for the best use of memory
 - More ranks?
 - More RAMdisk (allows for longer jobs)
- Stdin/stdout/stderr were bottlenecks
- Our target was improved scalability
 - But in the process, it sped up the code by more than a factor 20
 - Today one Mira node is 1.5x faster than a Grid node

See T. Uram's talk



Where we are today

- We can run using the entire machine
 - For throughput reasons, we normally limit ourselves to 1/3 of the machine: a million parallel processes
- Based on this success, the experiment asked us to do all the Alpgen generation for the next two years



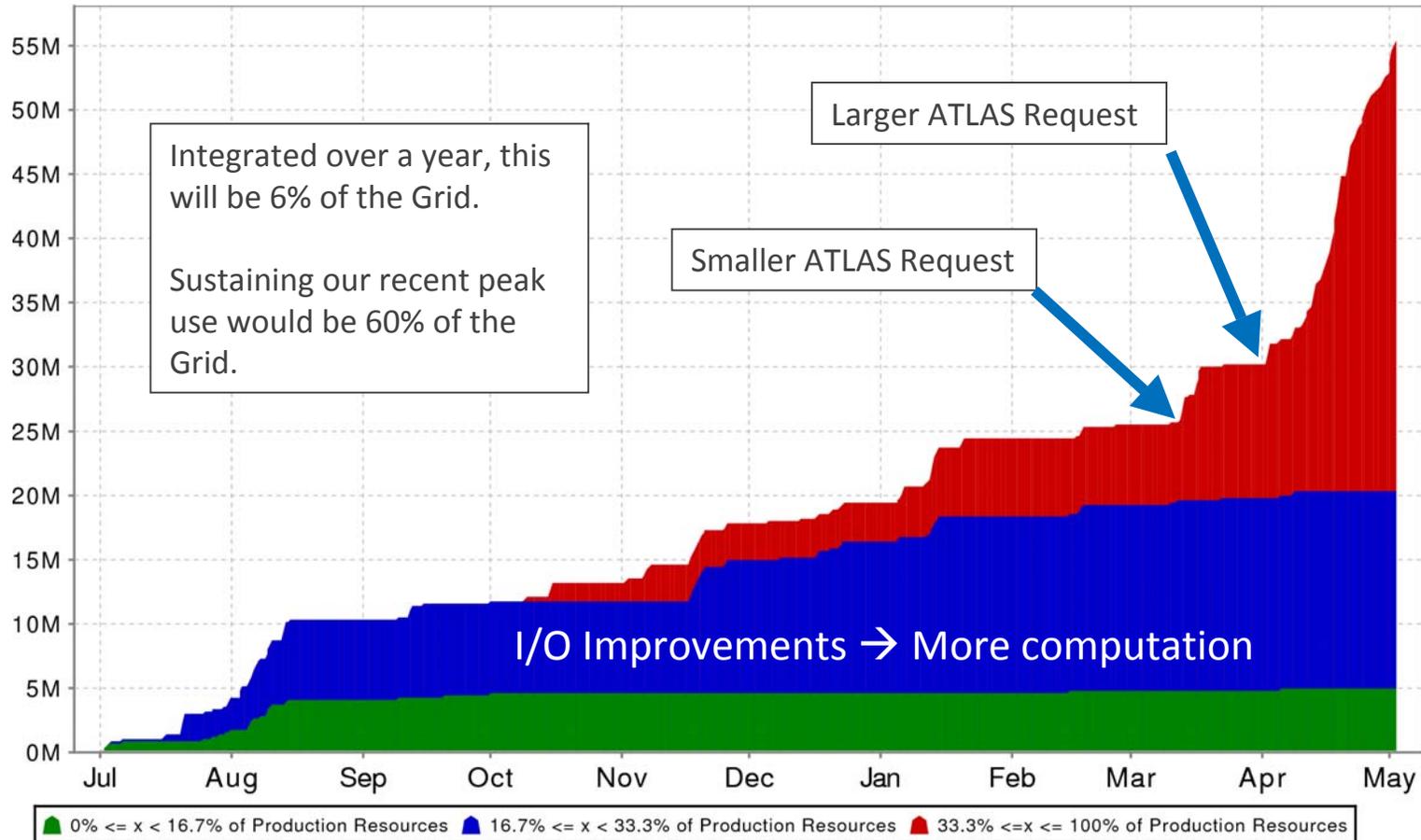
While this job was running, Mira was producing the equivalent computing as 5 or 6 ATLAS Grids.

On our best days, we provide the equivalent computing capacity of the whole ATLAS Grid.

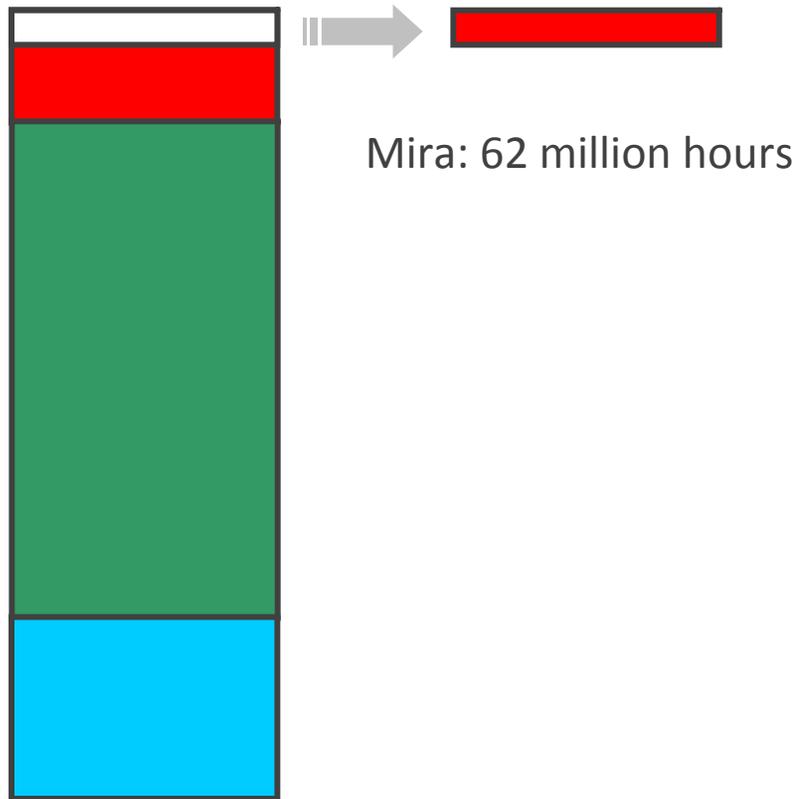


Where we are today

HadronSim
Machine: MIRA
Total core hours used per category
2014-07-01 to 2015-05-03



ALCC-2014 Use



Grid: One billion+ hours

- 70 million Grid-equivalent cpu-hours of event generation have been run on Mira and thereby offloaded from the Grid
 - 2 FTE worth of effort
 - If we were a country, we would be the 7th largest provider of cycles to ATLAS
- This is only event generation (red), but the recovered cycles can be used for whatever the experiment wants



What is this unweighting you promised to talk about?

- When an event generator makes an event, it assigns it a probability
- Unweighting involves
 - Generating a random number
 - If it's less than the weight, keep the event
 - Otherwise, disregard the event
- Alpgen does this as a separate step
 - Efficient: you can rescale to the highest weight produced
 - Usually about 1%, so you gain a factor of 100

This makes the generator output look like the data.



Our Next Target: Sherpa

- Sherpa (<https://sherpa.hepforge.org/doc/SHERPA-MC-2.1.1.html>) is an event generator similar to Alpgen
 - It attempts to capture more of the physics
 - It is 10+% of ATLAS gris usage
- It's written in C++ (about 700K lines of code)
 - It knows about MPI – but was intended to speed up desktops (8 ranks or so)
- It is code that writes code – it writes out object files that contain the code for the actual calculation, code that needs to be compiled later
- It runs in three stages
 - Write the code for the calculation (single-threaded)
 - Calculate phase-space integrals
 - Generate events
 - The 4th stage “afterburner” is not required (but some format fix-ups are)

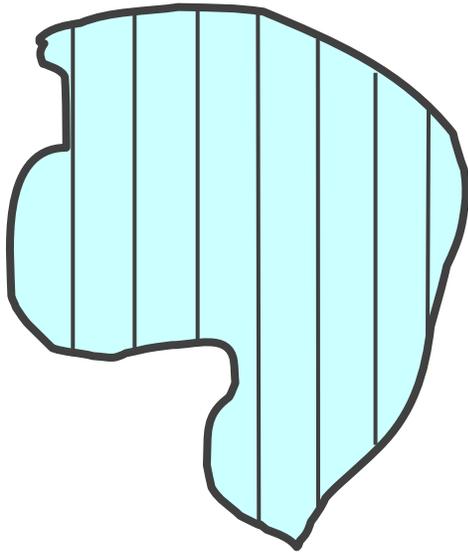


Phase-space integrals 1

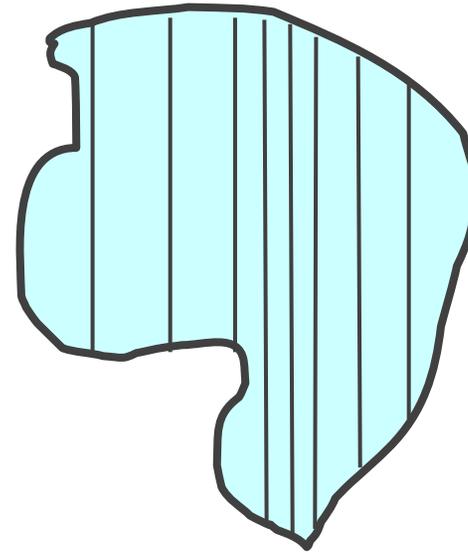
- These take between 1-2 weeks per process on a single core.
 - Too long for the Grid
 - The experiment needs a few thousand processes (remember, 418 papers)
- MPI helps, but not as much as we would like
 - Scalability initially topped out at ~60
 - With some tricks, we have gone up to 200-300 (our record is 768)
 - This fits nicely as a 3-node Edison job, and we've submitted hundreds
 - If we could fix this, we could calculate processes too complex to calculate today



Phase-space integrals 2



You can have each rank work on part of the problem, and sum them later. Equal spacing means some ranks finish sooner than others.



You can better balance things with unequal spacing, trying to match areas. There is a limit to this – if you knew the areas exactly, you wouldn't need to do the integral.



Sherpa Event Generation Today

- Our record is 16K x c8 or 131088 ranks
- A 16K job has no better throughput than a 12K (98304 ranks), so I have been saying we scale to about 100K ranks.
- Even this is a bit of a cheat, as a 12K job actually runs twelve 1K subjobs, each in its own directory.

The screenshot shows the Mira Activity web interface. The top part features a grid of job status indicators, organized into three main sections (R00-R0F, R10-R1F, R20-R2F) with sub-sections M1 and M0. Each cell in the grid is a small grid of colored squares representing individual jobs. Below the grid is a table of running jobs.

job Id	Project	Run Time	Walltime	Location	Queue	Nodes	Mode
390564	EddySim_CLF	11:11:00	12:00:00	MIR-40800-73B71-1-1024	prod-long	1024	c16
390957	QMCSIM	06:48:55	12:00:00	MIR-44000-78FF1-16384	prod-capability	16384	c1
391461	WindNoise	05:28:01	06:00:00	MIR-04C00-37FF1-2048	prod-short	2048	script
391483	WindNoise	05:26:51	06:00:00	MIR-04800-37BF1-2048	prod-short	2048	script
391387	LatticeQCD_2	00:31:37	06:00:00	MIR-04000-377F1-4096	prod-short	4096	script
387232	LiquidWater	00:31:06	06:00:00	MIR-40000-73371-1-1024	prod-short	1024	script
391896	NRSafety	00:30:32	06:00:00	MIR-40080-733F1-1-1024	prod-short	1024	script
392017	LatticeQCD_2	00:27:11	00:59:00	MIR-40400-73771-1-1024	prod-short	1024	script
391412	HadronSim	00:01:26	04:00:00	MIR-00000-38FF1-0100-16384	prod-capability	16384	script



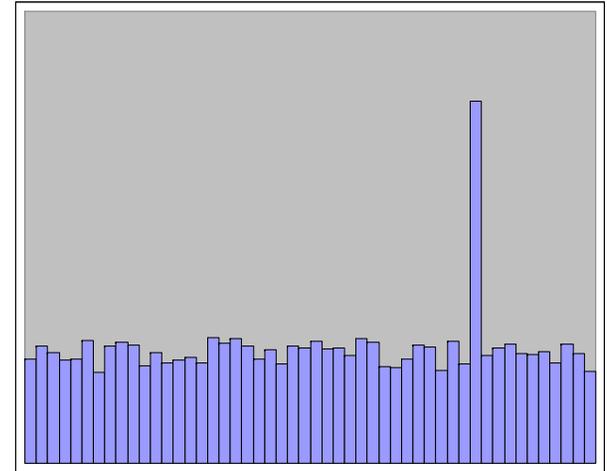
Sherpa Limitations

- Sherpa loves it's I/O
 - One example – the subprocesses are typically stored as 500-2000 .o object files
 - The authors provided an option to consolidate this into ~6
 - There are thousands of file opens in the course of a job
- Running the jobs in 12 subdirectories, each with it's own complete copy of Sherpa and its files helps somewhat
- Sherpa uses about 1.2 GB per rank
 - Prohibits c16 mode
 - Can we cobble together a pseudo c12? Perhaps use the leftover for a RAMdisk?
 - Or perhaps we could squeeze Sherpa into a smaller footprint
 - Again the question of how best to use the memory that we have



Sherpa Limitations 2

- The design of Sherpa has problems with a large number of ranks:
 - Sherpa generates a fixed number of events per rank
 - The time it takes to generate an event varies
 - Can be a factor of 100, although 10 is more typical
 - Sherpa unweights on the fly rather than in a separate step
- All of these things make Sherpa susceptible to “stragglers” – one rank that takes much longer than the others to complete
- The initialization takes about an hour
 - Makes debugging difficult
 - What can it possibly be doing – an hour is ~one trillion calculations!

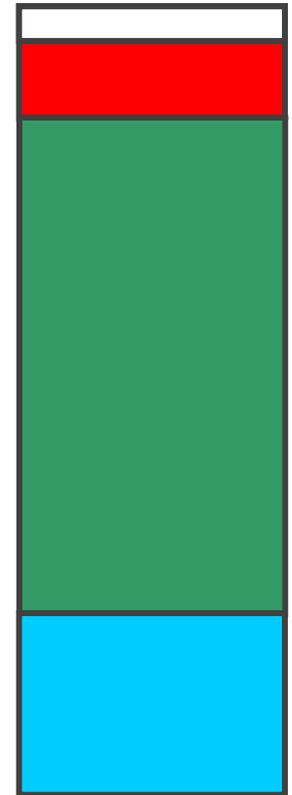


I have no doubt that we will sort this all out – but it is real work.

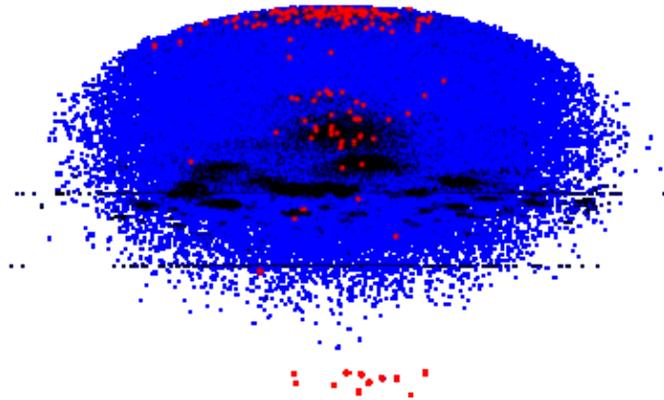


Simulation/Digitization

- Simulates the interaction of these particles with the detector
- This is the green part of the diagram
 - Once ~70% of the usage
 - Falling to 60% or even less – but only because the red (generation) has grown
- This uses a C++ toolkit called Geant4
 - <http://geant4.cern.ch/>
 - About 1 million lines of code
 - The only program HEP uses for this
 - Modular, so different models can be inserted in the same framework
 - Identifies the position and amount of energy deposits in the detector
 - Energy is all that matters



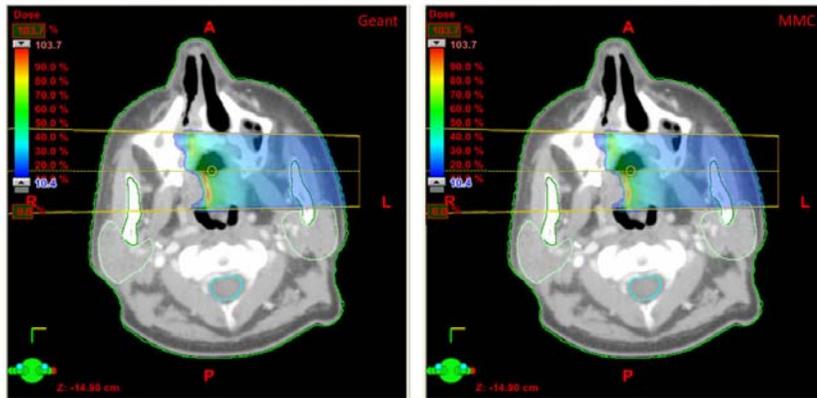
Geant4 goes beyond HEP



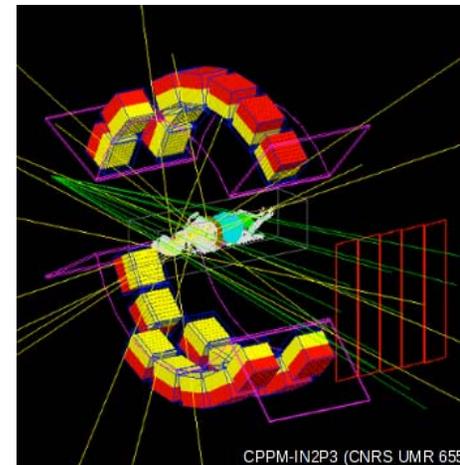
Electron avalanches in thunderstorms



Radiation effects on satellites



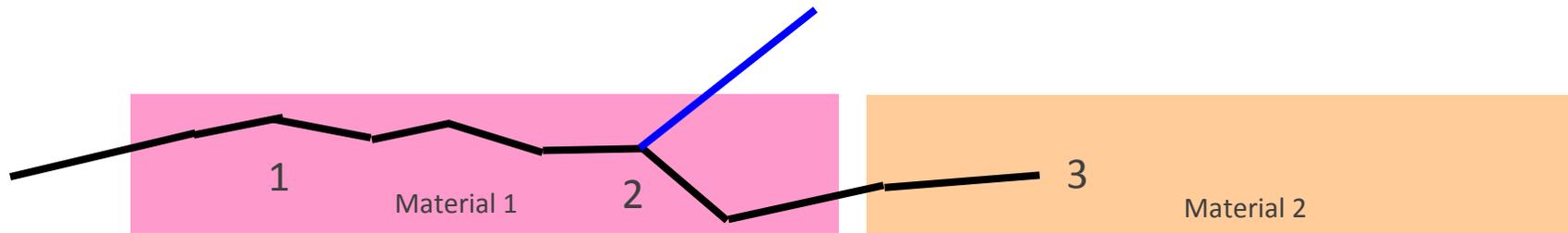
Dose calculations in proton therapy



Improved PET scanning development



How It Works



- The program takes small steps with each particle through the detector. At each step, one of three things can happen:
 - 1. The particle deposits some energy via ionization, and takes another step
 - 2. A new particle is created, and added to the list of particles
 - 3. The particle is absorbed, stopped, or exits the detector and removed from the list
- Geant starts with the particles produced in the collision and continues until the particle list is empty
- This takes a few minutes per event (for a typical ATLAS event)



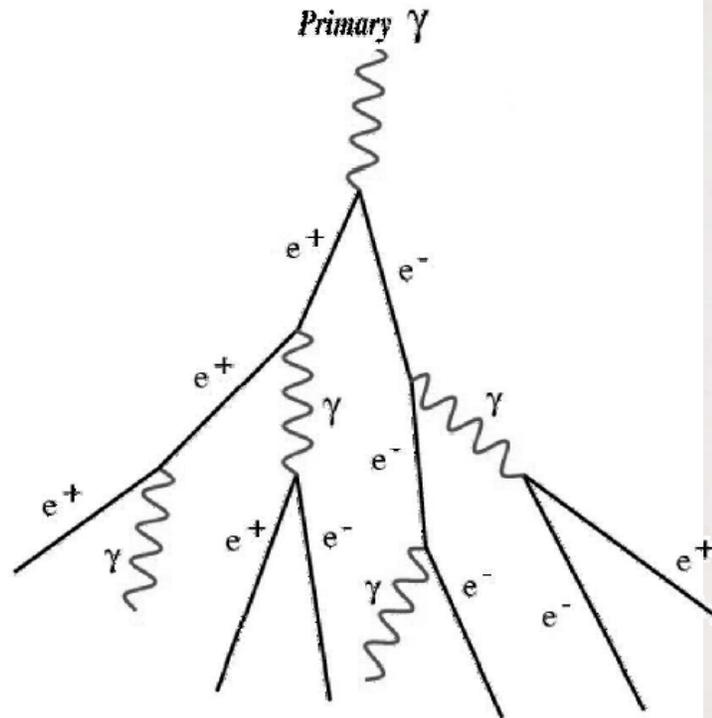
Electromagnetic Showers in Geant

An electron or positron can interact with material and produce a photon.

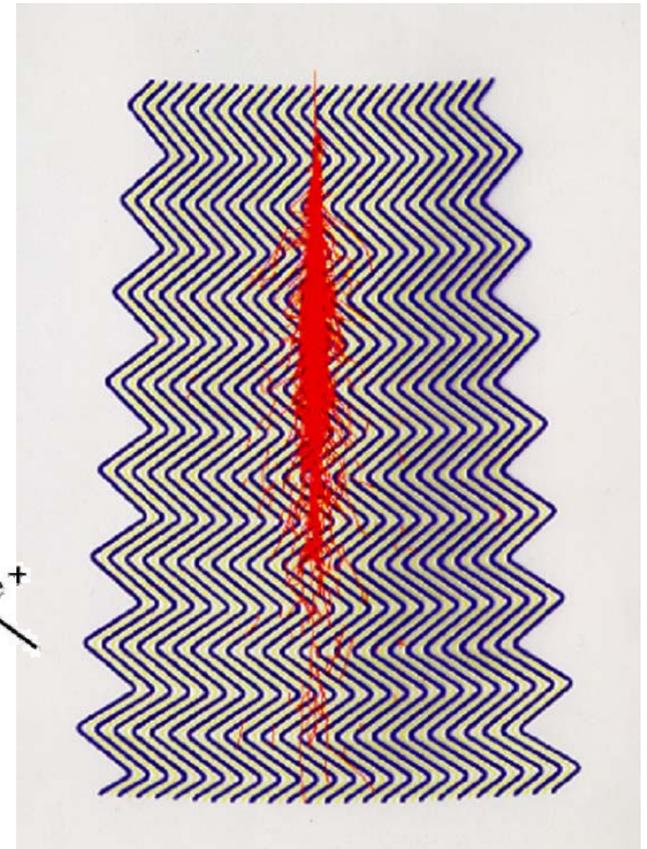
A photon can interact with material and produce an electron and a positron.

This process continues until all the electrons are stopped and all the positrons have annihilated.

Thousands of secondaries are produced per primary.



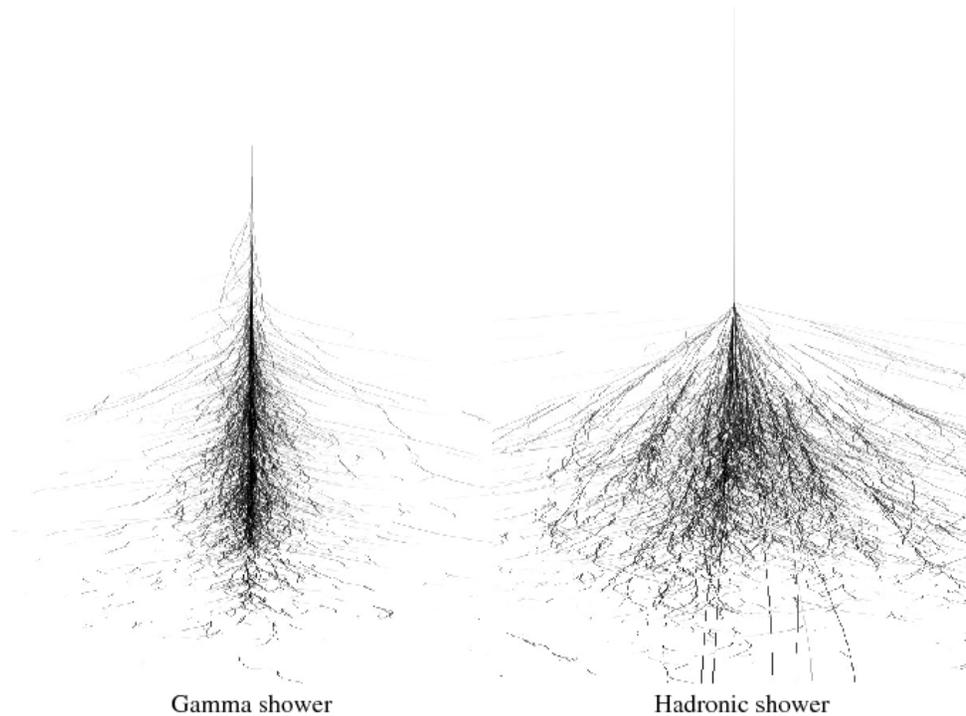
A schematic of an electromagnetic shower



A GEANT simulation of an electromagnetic shower



Hadronic Showers



- EM showers all look the same



- Hadronic showers are like snowflakes
 - Every one is unique

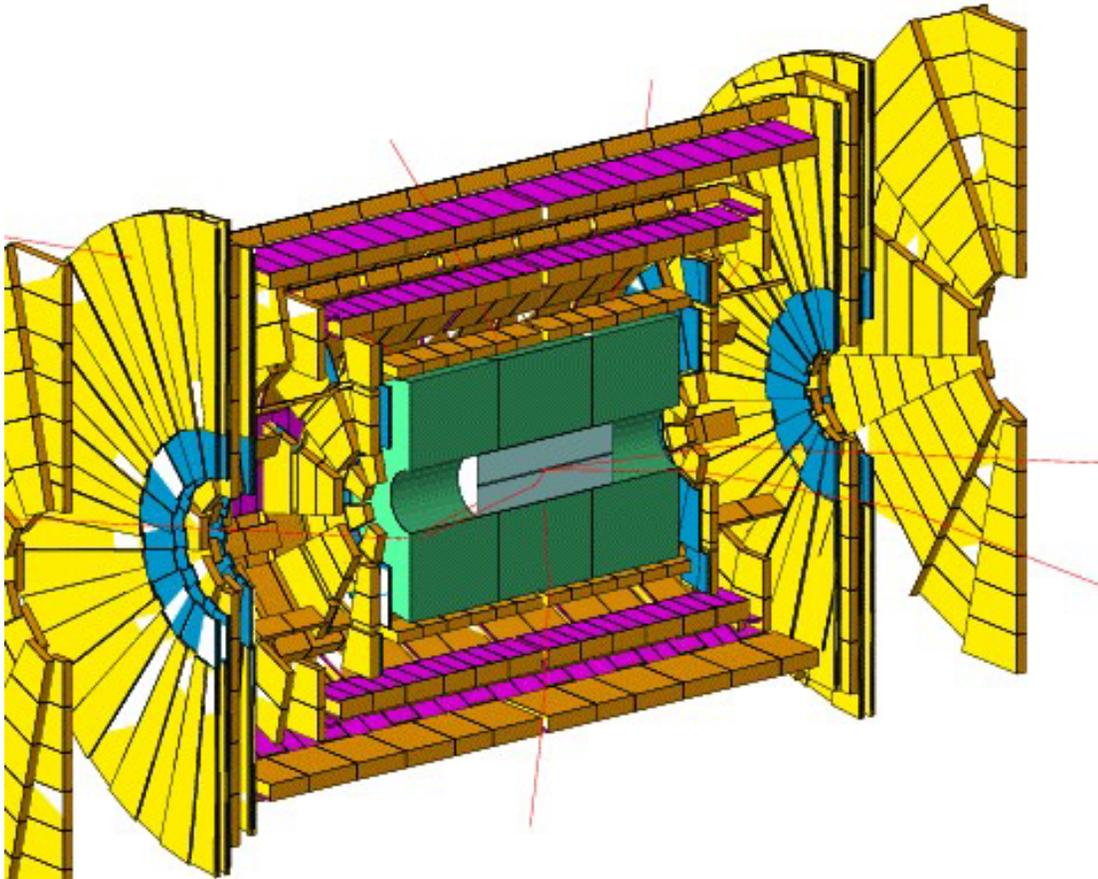


The same sort of thing happens with hadronic showers – showers produced by particles like protons, neutrons and their cousins.

These images are simulations of cosmic rays interacting with air.



Geant4 and ATLAS

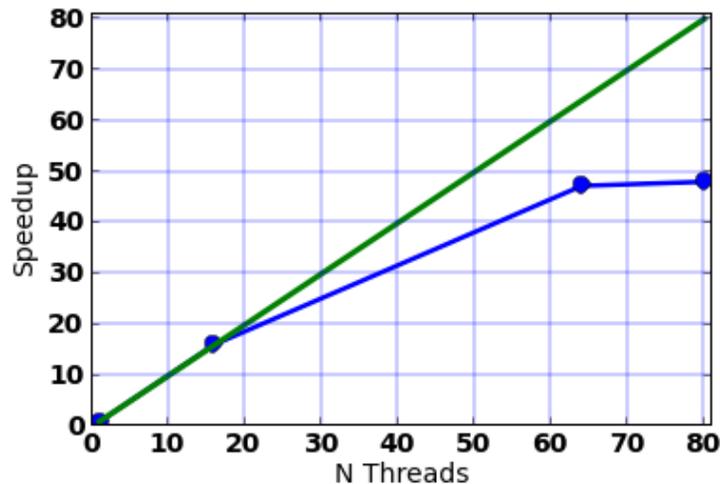


- A Geant4 model for ATLAS exists (and has been used for about 5 billion simulated events)
- It has about 400,000 volumes, representing 100 million readout channels
 - It incorporates detector symmetries to keep the volume count low
- The simulation runs inside the ATLAS framework and takes about 2 GB, single-threaded



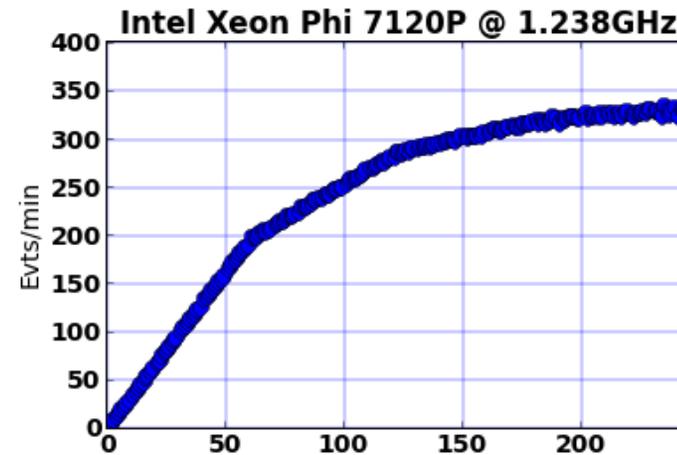
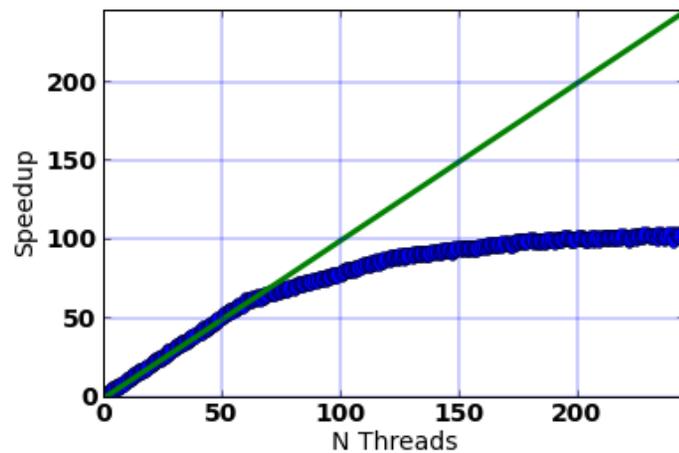
Recent Geant Developments

- There is now a multi-threaded version of Geant4. (G4.10)
- It makes the detector geometry and the physics list read-only and shared
 - The physics list is the model for how particles interact with matter
- This allows us to go from –c8 mode on Mira to 64 threads
 - Gain a factor of 6 (should be 8)
 - Overcommitting to 80 threads gets us a factor of almost 6.1

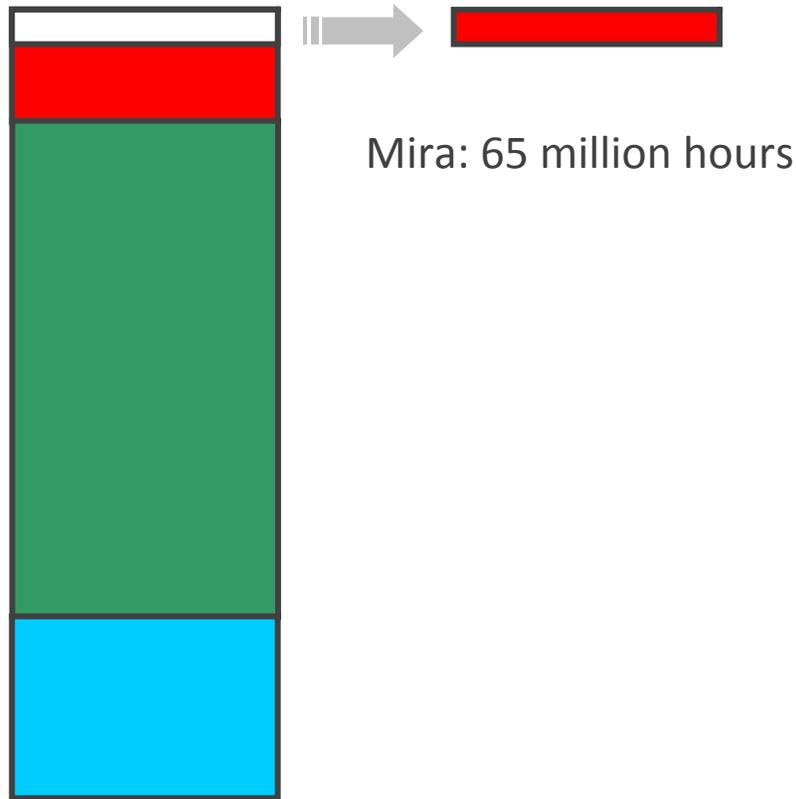


Geant and the Future

- Scaling performance on present-generation Phis is promising
- Several development directions ongoing
 - Improved use of vector operations
 - ~30% of the time is spent in 2% of the code – but Geant still has 2000 classes
 - Use of GPUs (a substantial job – 2000 classes)
 - A possible complete refactoring of the code



Growth Area: Event Generators

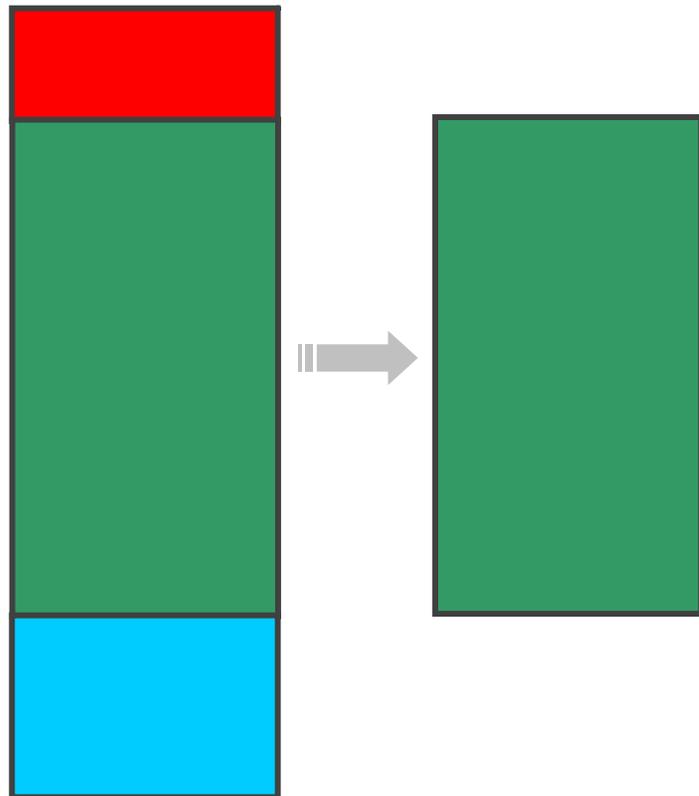


Grid: One billion+ hours

- We could – and intend to - double this by porting a 2nd generator, Sherpa, to Mira.
- If we succeed in scaling the Sherpa integration phase, this could increase by an order of magnitude
 - We would be able to simulate events we couldn't simulate before. These events will be highly in demand.
- We would like the number of simulated events to scale with the number of real events
 - Without HPCs, this looks unlikely



Growth Area: Simulation

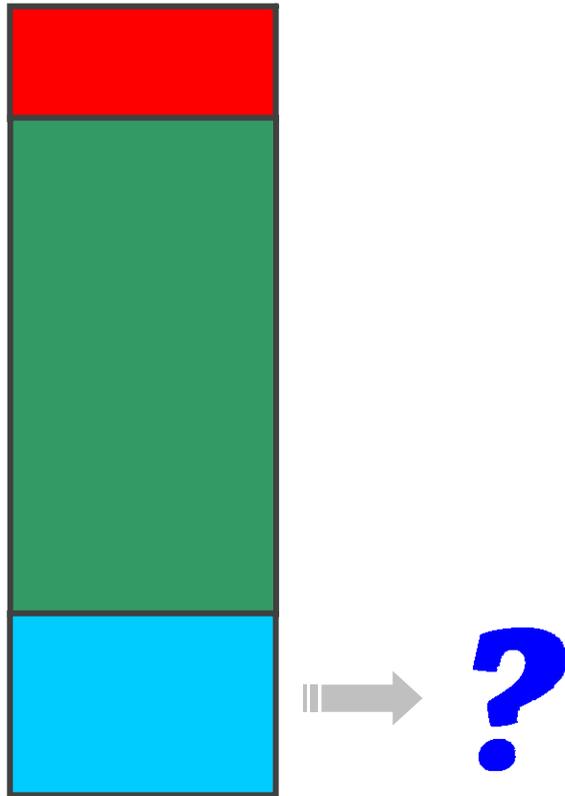


- Virtually all simulation could be done on HPCs
 - The Geant4 kernel runs well today
 - The ATLAS software framework would need to be ported*
 - This is more a long problem than a hard problem.

- Unlike event generation, this is data-intensive
 - Today occupies many 10's of PB
 - Scales with the data collected
 - Around 2025 will reach 1 EB.

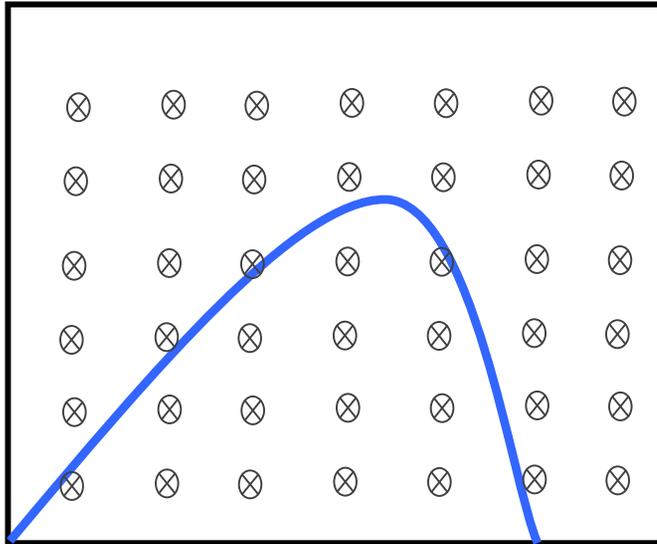
* Runs on Edison today; but all announced supercomputers have different architectures.

Growth Area: Reconstruction and Analysis



- Reconstruction is the use case the Grid was built around – this is I/O heavy
 - Under what, if any, circumstances does it make sense to move this piece to HPCs?
 - I don't know the answer today – but the it is probably related to data availability at the HPC sites
- Analysis might grow by a lot
 - The analysis paradigm is determined by the amount and nature of the computing available.

How We Set Limits Today



Suppose our theory has parameters x and y , e.g. masses of unknown particles. We then take a grid of (x,y) points, and see if the point is allowed or excluded by the data.

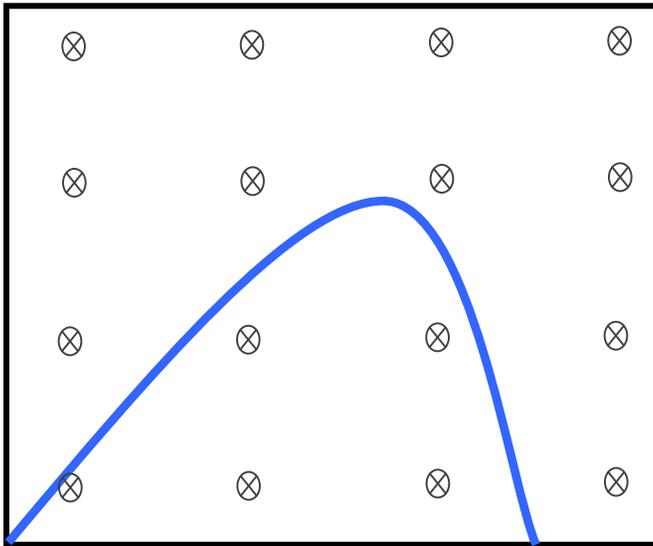
Based on the outcome of those tests, we interpolate an **exclusion curve**.

These points are run on the Grid, so run at different places and time. There is no opportunity for coordination.

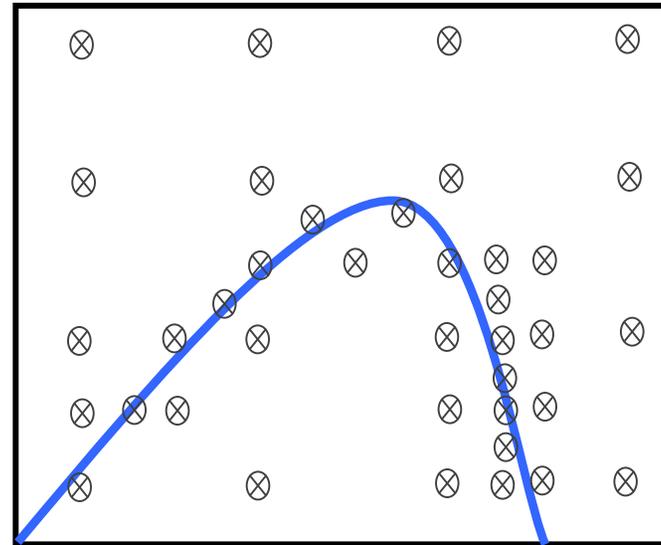


How We Might Set Limits Tomorrow

Start with a coarse grid



Iteratively refine the grid



These points are run on the at the same place and time.
The program can coordinate, and converge on a better answer, faster.

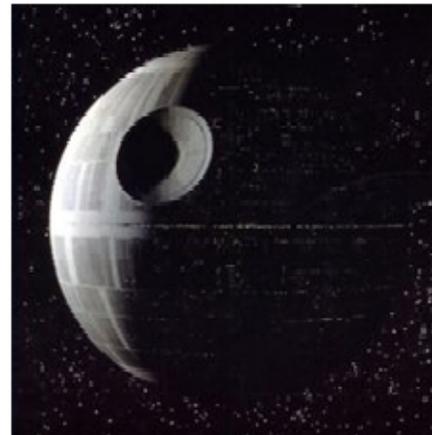
Additional Benefits

- The past slide shows how better limits can be obtained at reduced computational cost.
- What if we were to increase computational cost?
 - We often have models with >2 parameters
 - We set limits in 2-d slices, with the other parameters kept constant
 - “Doing anything else would require a supercomputer!”
 - Could we set 3-d limits?

2-d slices don't easily separate this



from this.



Final Thoughts

- HEP is a non-traditional user of HPCs: our codes have co-evolved with HTC farms and now the ultimate farm, the Worldwide LHC Computing Grid
- Even so, starting with event generators, we are now providing computing equivalent to a medium-sized country
- Moving into other event generators and simulation/digitization will allow us to grow, concurrent with new HPC deployment
 - Our fractional use of these systems will be about what it is today – but will be a larger part of the experiments computing ecosystem
- Adapting these codes is challenging – but possible
 - I would argue that this is necessary for us to reach our science goals
 - We're starting to see the benefits now







Leftovers



Event 1 - 4 μ

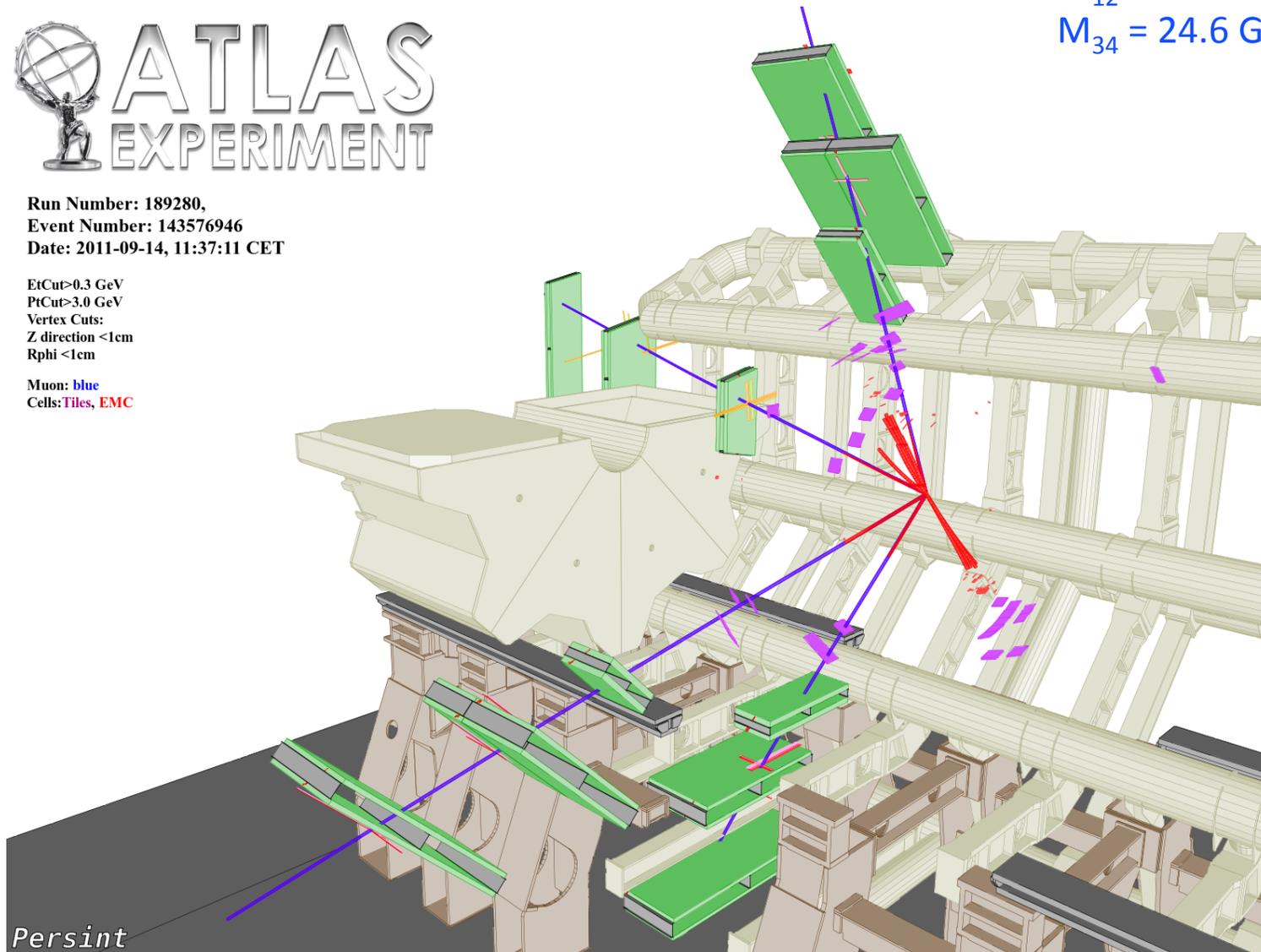


Run Number: 189280,
Event Number: 143576946
Date: 2011-09-14, 11:37:11 CET

EtCut>0.3 GeV
PtCut>3.0 GeV
Vertex Cuts:
Z direction <1cm
Rphi <1cm

Muon: blue
Cells: Tiles, EMC

$M_{12} = 89.7 \text{ GeV}$
 $M_{34} = 24.6 \text{ GeV}$



Persint



Event 2 - 2 μ 2e

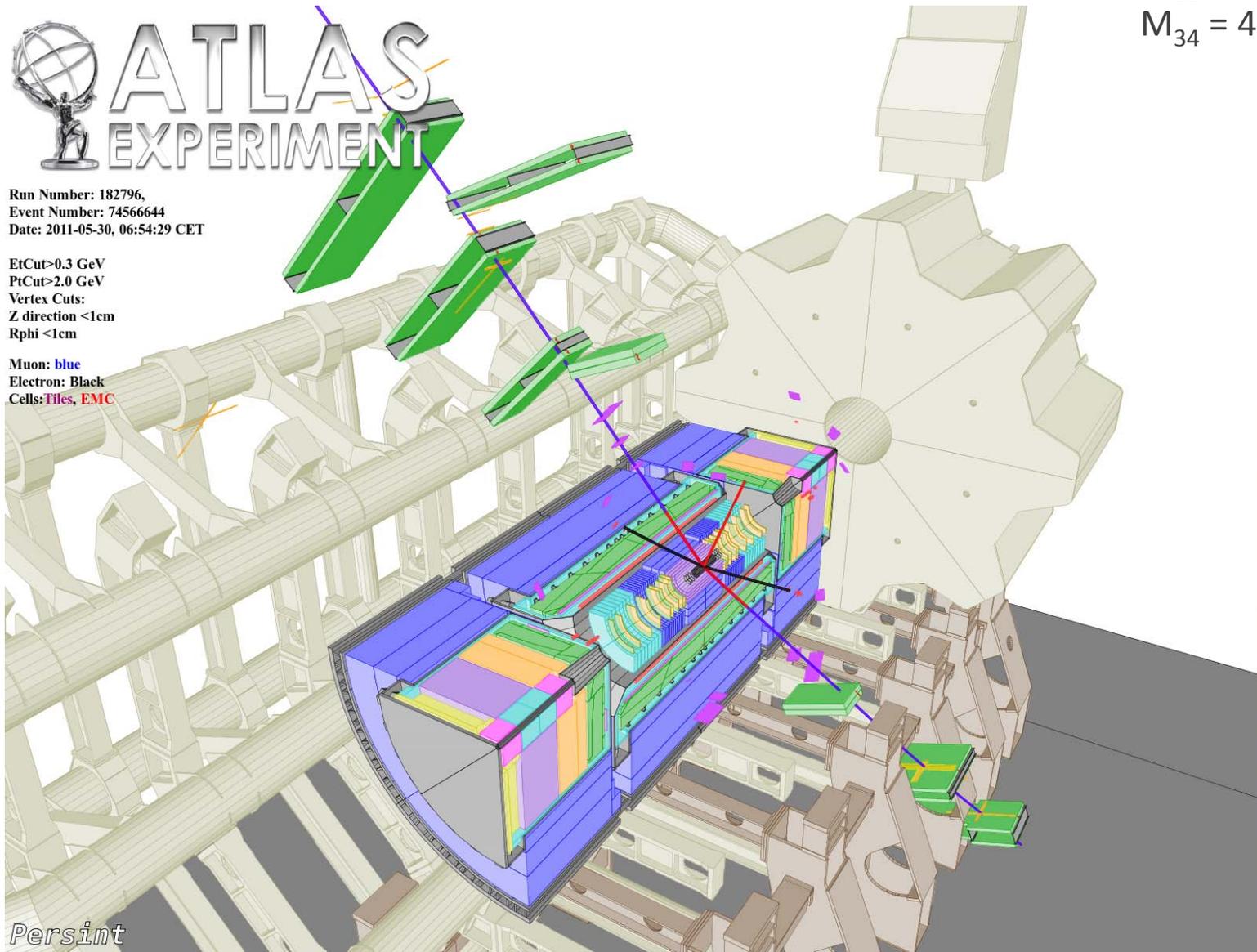


Run Number: 182796,
Event Number: 74566644
Date: 2011-05-30, 06:54:29 CET

EtCut > 0.3 GeV
PtCut > 2.0 GeV
Vertex Cuts:
Z direction < 1cm
Rphi < 1cm

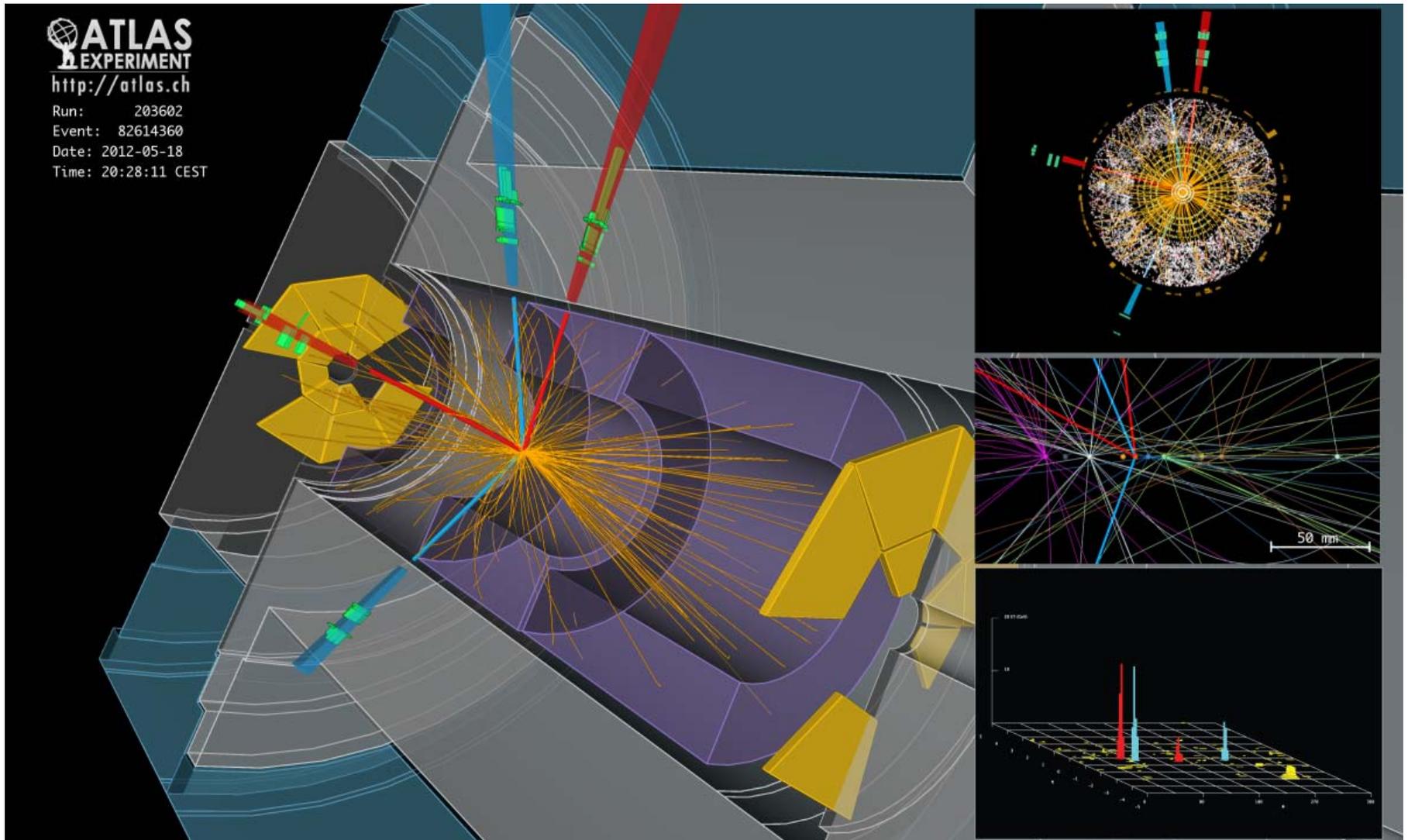
Muon: blue
Electron: Black
Cells: Tiles, EMC

$$M_{12} = 76.8 \text{ GeV}$$
$$M_{34} = 45.7 \text{ GeV}$$



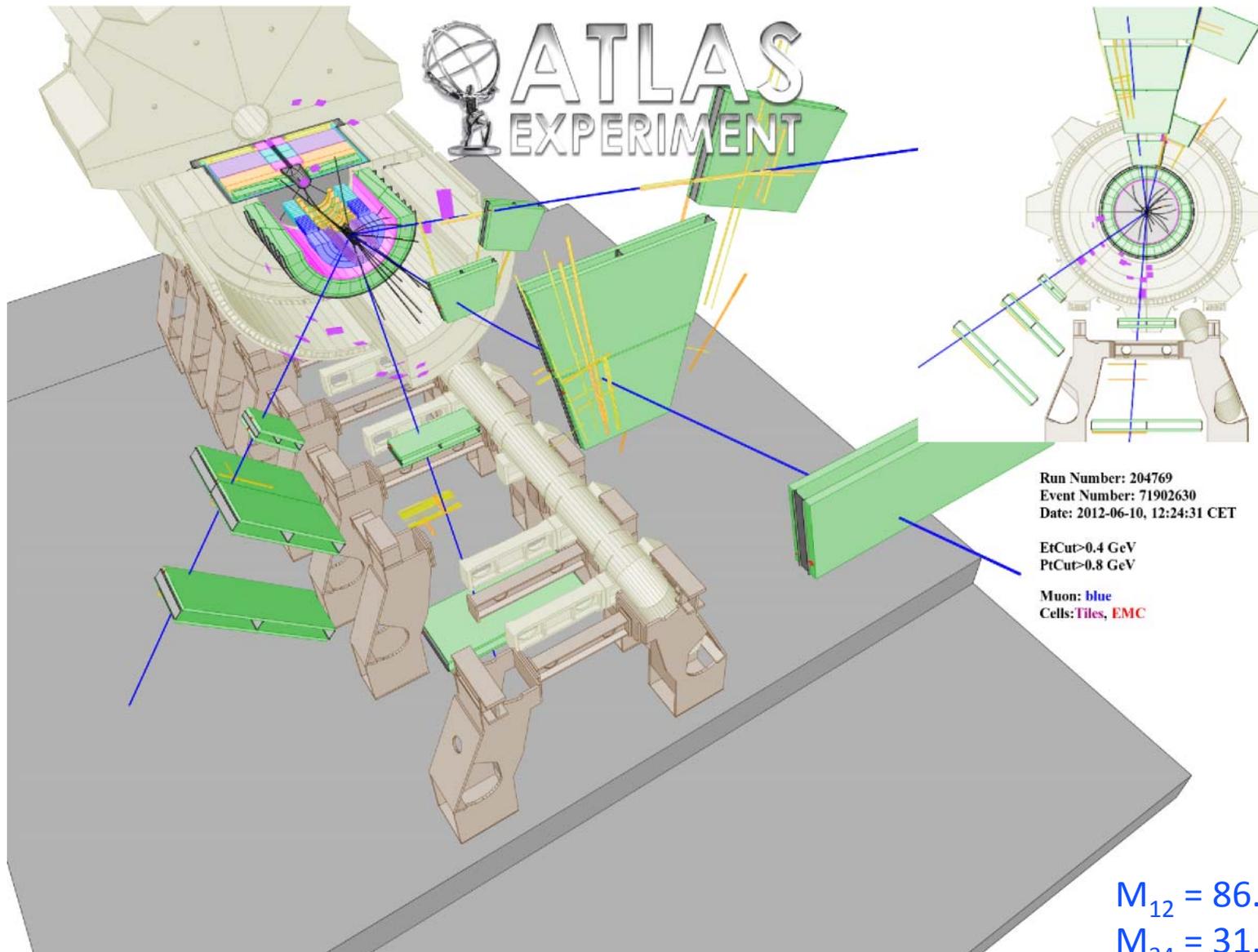
Event 3 - 4e

$$M_{12} = 70.6 \text{ GeV}$$
$$M_{34} = 44.7 \text{ GeV}$$

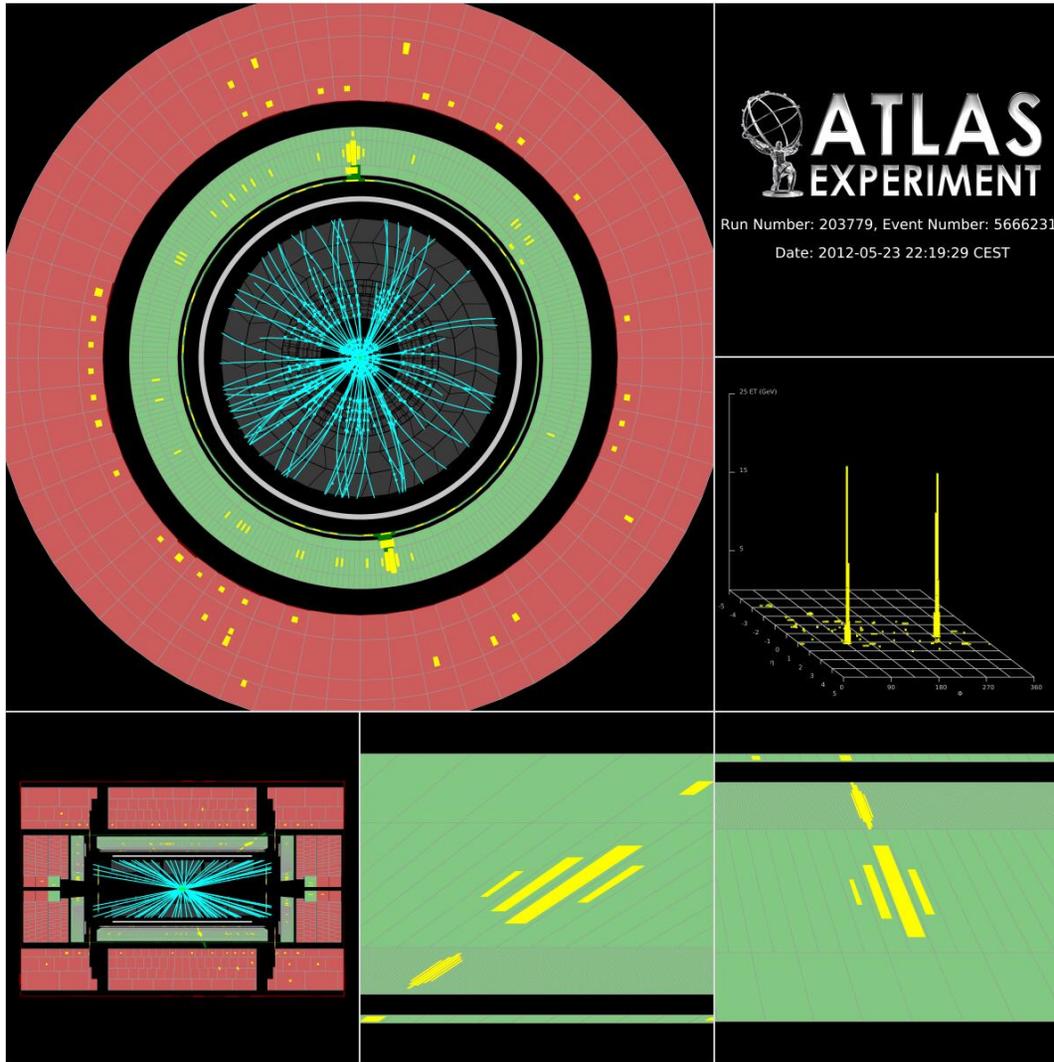


Event 4 - 4 μ

(Everything goes forward)



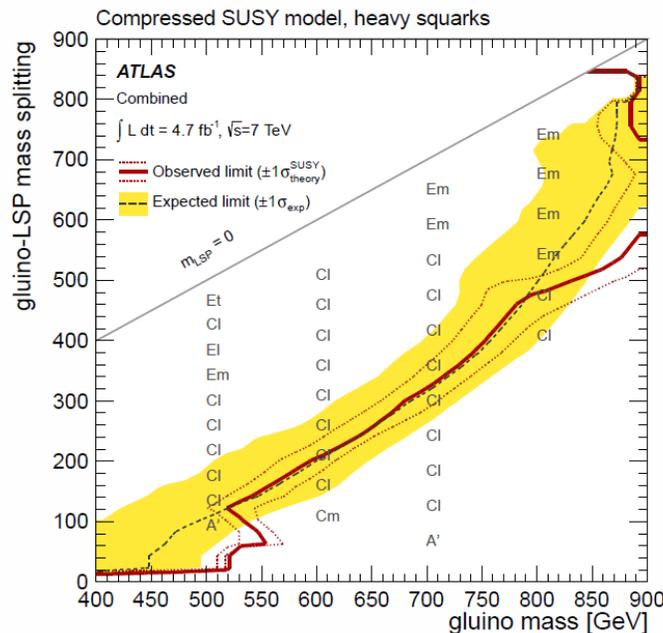
A Typical Pretty Two Photon Event



- Photons are obvious, even with pile-up
 - Note that low p_T tracks are suppressed in the display
- One can see how the EM showers can be used to point back to the primary vertex
 - Usually points to within ± 1 interaction of the correct vertex
 - This is as good as it needs to be; beyond this it's diminishing returns
- Three photon regions: central, endcap, transition
 - The transition region is difficult

How We Got Into This

- When I came back after being ATLAS physics coordinator, I was looking for some physics to do.
- Steve Martin (NIU) and I asked ourselves about the sensitivity of the LHC experiments to moderately compressed SUSY spectra
 - Answer: better than you might think (c.f. PRD84 015004 and PRD85 035023)
- ATLAS did the study, and here's the outcome (as shown in the 7 TeV paper):



The problem

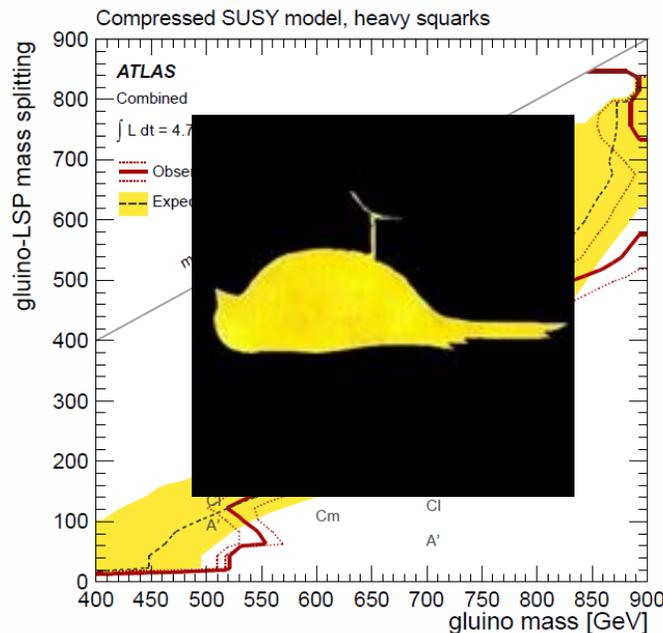
We didn't simulate these points, so we have no idea what the limits are here.

- The observed limit is better than our expected limit. We (perhaps foolishly) did not run any points out there.
- This set of points spent ~2 months in the queue waiting for open slots.
- We decided not to delay this paper by another ~2 months



Conclusions From That Last Plot

- This is our canary in a coal mine
- The upcoming shortage of computing isn't upcoming any more – it's here
- The science is starting to be no longer limited by the number of events we can record, but the number of events we can simulate. *This is unacceptable.*



The problem

We didn't simulate these points, so we have no idea what the limits are here.

- I sometimes get asked “You were physics coordinator. Couldn't you use your clout to jump the queue?”
- Probably, but that's just buying another canary.



Argonne Leadership Computing Facility Partners



Hal Finkel – *Catalyst*
(formerly HEP)



Tom Uram - *Software Development Specialist*



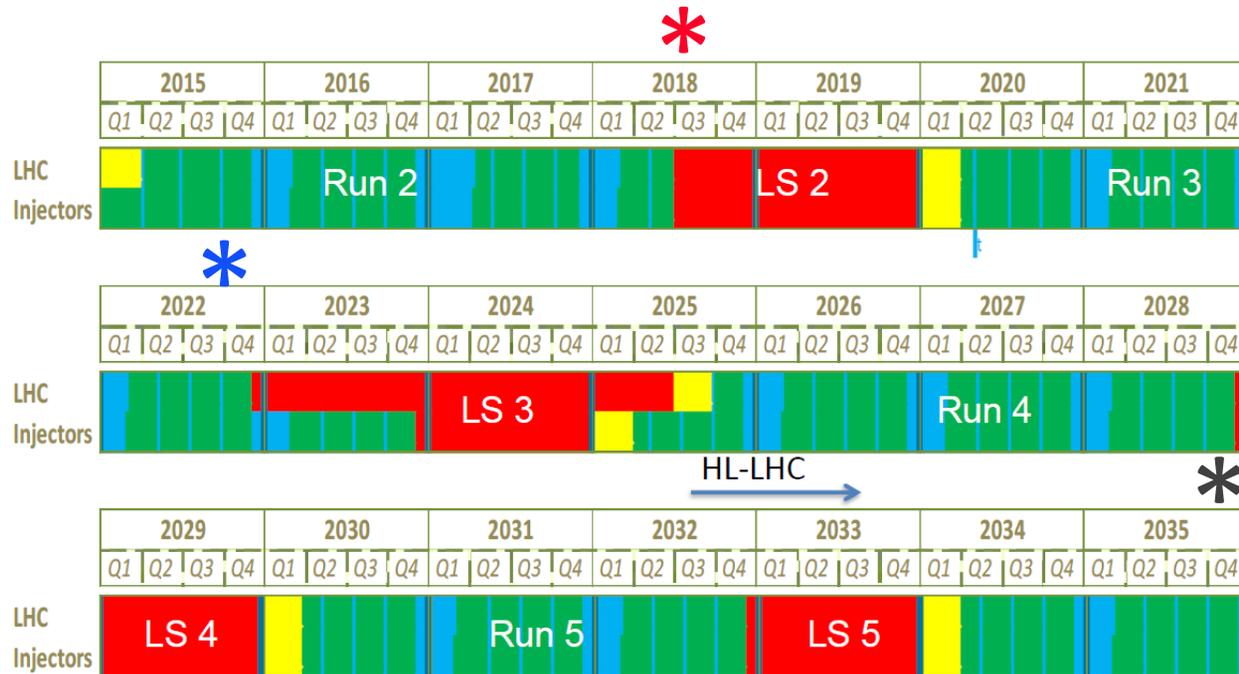
Venkat Vishwanath -
Computer Scientist

These are among ALCF's very best people – it shows they are serious in applying high performance computing to HEP.

Doug Benjamin (Duke) has also been very helpful – particularly where this work touches the rest of ATLAS.



Future Plans



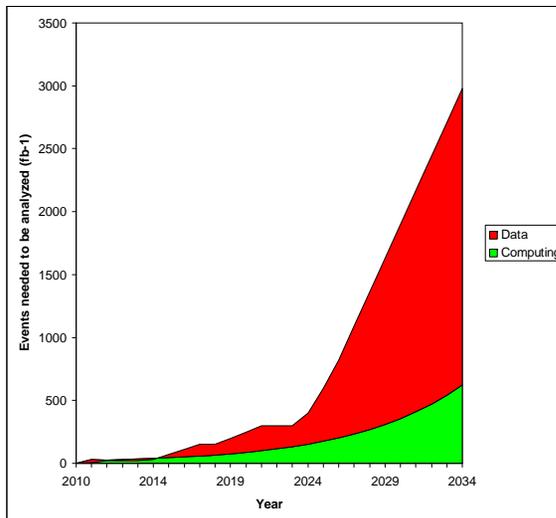
- * Have 3x today's data
- * Have 10x today's data
- * Have 100x today's data

High Performance Computing Motivation in HEP

There are two problems we are trying to solve

- The capacity problem

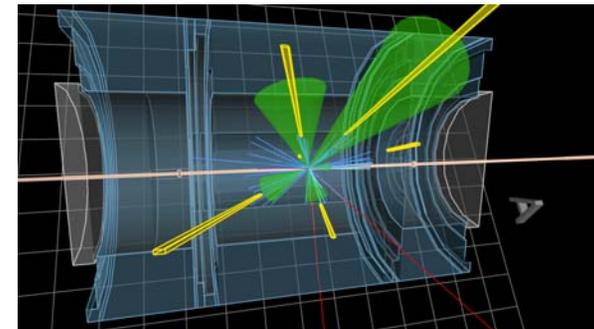
- Our needs are growing faster than the Grid is growing – and likely can grow



The green assumes 15% growth per year from Run 1, and that Run 1 had exactly enough capacity.

- The capability problem

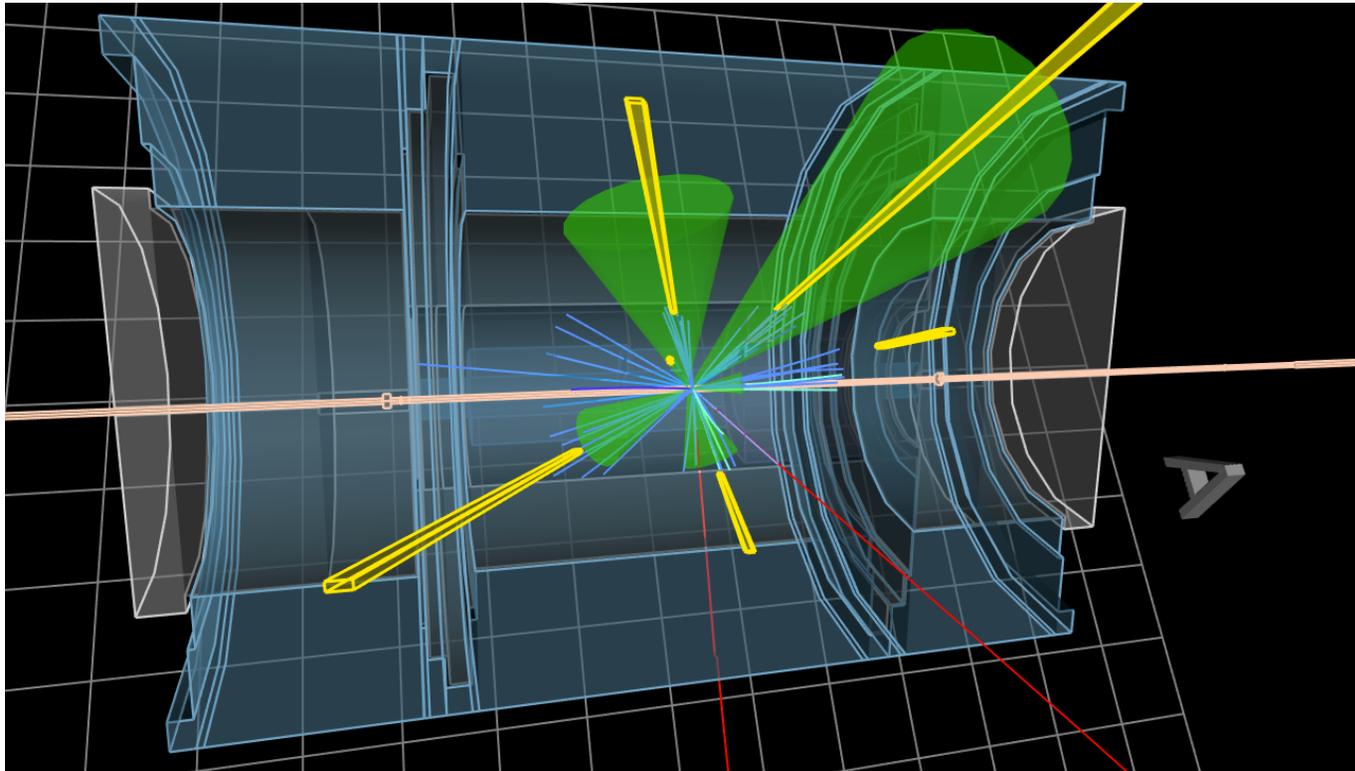
- Some problems can't run on the Grid
 - Highly filtered samples can produce zero events in a job – which looks like a failure and triggers a resubmit. Which triggers a resubmit. And so on.
 - Sherpa integration times stretch into the weeks, but can be done overnight with a few hundred threads (scaling is poor today, but this makes doing this *possible*. Efficient comes later)



We see HPCs playing a role in both of these



Events Like These:



- This is a $Z (\rightarrow \tau\tau) + 5 \text{ jet}$ event, with highly filtered τ decays.
- ATLAS requested that we make them, because they failed on the Grid
 - The degree of filtering is so high, runs often have zero events pass – misinterpreted as a failure. This is not a problem with thousands of cores.
- There are another 46,998 events just like this one.