

Argonne LCF Status

Tom LeCompte
High Energy Physics Division
Argonne National Laboratory

Tom Uram, Venkat Vishwanath
Argonne Leadership Computing Facility
Argonne National Laboratory

Doug Benjamin
Department of Physics
Duke University

Strategic Thoughts

“Why do you rob banks?” “Because that’s where the money is.”

Attributed to Willie Sutton

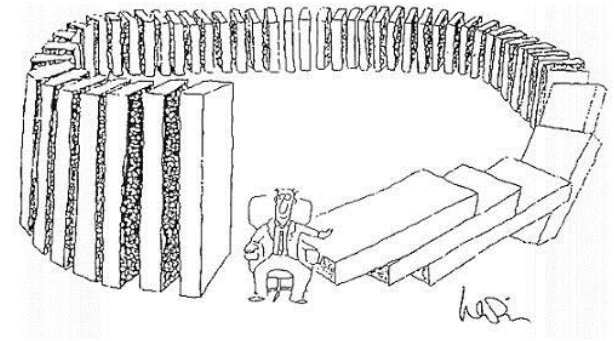


- On the ATLAS side, Simulation is “Where the money is”
 - This is something like ~60% of our grid use
 - Event generation is well-suited for HPC, but is < 10-15% or so
 - This will be helpful, of course, but it’s a smaller piece
- On the HPC side, large (>500K core computers) are “Where the money is”
 - 1/3 of the Top 500 computational capacity is in four computers (Titan, Mira, Sequoia, K)
 - 1/2 of the Top 500 computational capacity is in 21 computers
- We want a “run-anywhere” design
 - More total resources
 - More flexibility in utilizing them
 - Protects us against future changes

If we offload a cycle from the grid, we can use the recovered cycle any way we want.



Consequences




- We need to adapt to run on a rather spartan architecture (typical for these giant computers)
 - No/little/slow TCP/IP (=no/little/slow database access)
 - Relatively little memory per core
 - Some use non x86-instruction sets
- We need to connect this to the grid – otherwise it's worthless
- We are solving both problems with an x86-based front end
 - Lives on the grid – looks like a fast Tier-2
 - Receives jobs
 - Adapts them to the supercomputer
 - Sends the output to a grid SE


Looks a lot like a many-core architecture for future PCs.

How it Works (Alpgen example)

- User sends a prun job to the front-end with an Alpgen input file and a number of events
- The front-end then
 - Runs a “warmup job” to create the .grid1,.grid2 files
 - Creates a “purely computational” job for the HPC
 - Notifies the HPC that there is a job waiting
 - Polls the HPC to see when it is done
 - Collects the HPC output
 - Runs an “unweighting” job
 - Places the output on the grid
- Output can be found here:
 - user.lecompte.000072._00003.alpout.unw in dataset user.lecompte.ANALY_ANLASC.001/ (BG/P)
 - user.lecompte.000072._00006.alpout.unw in dataset user.lecompte.ANALY_ANLASC.001/ (BG/Q)



Logically similar to fetching database files for simulation



Logically similar to a final format fix-up for simulation



ALCF Resources

- Intrepid
 - BlueGene/P based (PowerPC 450 cores) – 2006-vintage technology
 - 163,840 cores (40960 nodes x 4 cores/node @850 MHz)
 - Each core is 1/3 to 1/5 the speed of a typical x86
 - 470 MB per core
 - Ideal job sizes: 512-4096 nodes, 6 hours or less
 - About 100-1000 x86-days
 - Two small development systems (Challenger and Surveyor)
- Mira
 - BlueGene/Q based (PowerPC A2 cores) – Intrepid’s successor
 - 786,432 cores (49152 nodes x 16 cores/node @1.6 MHz)
 - Each core is comparable to the speed of a typical x86
 - 1 GB per core
 - Ideal job sizes: 512-4096 nodes, 6 hours or less
 - About 2000-15000 x86-days
 - Two “small” development systems (Cetus and Vesta)
 - These are Top500 #139 and #140



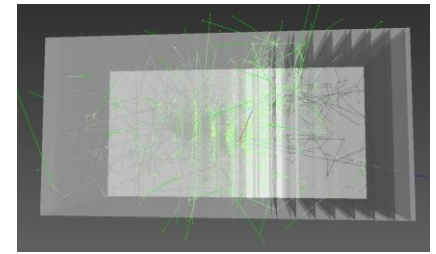
Other Possibilities



- We have 1M hours allocated on Mira, and 0.5M on Intrepid
- I have a handshake agreement that we can go beyond this, plus
 - Access to 1M hours on Hopper (at NERSC, at LBNL): x86 with 156,213 cores
- There are additional US supercomputers outside of the National Labs, at institutes with some affiliation to ATLAS
 - Examples: Stampede at Texas, Blue Waters at Illinois
- There are some worldwide computers with similar architectures
 - SAKURA is a BG/Q system, for example, like Mira



Code Issues



- Everything we have seriously tried runs: Geant4, Root, Alpgen, Sherpa
 - Haven't tried Athena, just the foundational code above
- Code changes are minor
 - There's a gcc bug that has to be programmed around in Alpgen
 - (bug exists, but is not triggered, on an x86)
 - We use MPI to ensure unique random number seeds
 - G4 & Root require no changes
 - Rolf warns us that there is an "is this an x86?" in one spot in the ATLAS repository
- Build environment issues are more substantial
 - Geant4's build environment makes some x86-based assumptions in paths
 - Freetype2 (a Root dependence) is complicated
- Optimization
 - Effects are large and not-simple
 - Enabling one FPU on the BG/P almost doubles the speed
 - Enabling the second one slows it down by ~8%
 - We decided not to spend a lot of time on this: the strategy is to get going first, and improve things second



Status



- We have accepted an Alpgen grid job (4-vectors), run it on BG/P (and also BG/Q) and placed the output on the grid.
 - Technically, we could go into production immediately
 - One person no longer has to sit there entering passwords from her cryptocard non-stop.
 - Politically, this needs some discussion
 - The communication between systems that lets us do this automatically works, but could use some more development
 - In particular, better error-checking and robustness should be in place before entering a production mode.
 - Developed at ALCF, usable anywhere

- Sherpa-MPI is an obvious next target

- After that – maybe showering of Alpgen 4-vectors?

- We will be evolving the HPCs (multiple) \leftrightarrow x86 negotiation
 - Include allocation quotas, expected time to complete, allow failed jobs to restart...



The Future

- Today, at submission time, a job specifies a machine and a number of cores. The hooks are in place to make this dynamic.
- We want to be able to select computers at run-time, based on
 - Estimated completion time
 - Quota available
 - Upcoming jobs (we don't want to start a job that will block a later one)
- We want to dynamically alter the “shape” of a job
 - Recast a 1000-core 2-hour job as a 2000-core 1-hour job
- The vision is to be able to run opportunistically, in the “nooks and crannies”
 - There is a lot (by our standards) of CPU available this way
 - It tends to be available at certain days and times (Sunday nights) – accepting longer latencies will enable us to do this



Thoughts In Lieu of Conclusions

- We're using prun now – should we continue this in the future?
 - Maybe there should be a “phpc” mode in addition to prun and pathena
- Accepting longer (10 days?) job latencies would help
- ATLAS tries very hard in several places to make the jobs x86-sized.
 - This is ~1000x smaller than we would like.
 - Random number seed starvation is a concern. Alpgen (for example) asks for two five-digit seeds. The second seed is made unique across nodes, so we burn through seeds 10's of thousands of times faster.
- Validation of Alpgen 4-vectors is taking longer than we would like (delivered 21 Feb)
 - Single-core runs can be compared bit-for-bit with x86 (OK with Alpgen)
 - Multi-core runs can't (different seeds)
 - We need to think about this systematically
- A well-designed single event server may make running ATLAS Simulation easier
 - Especially if it can send events via MPI as well as TCP/IP
- An integrated simulation framework may make running ATLAS Simulation harder

