Multi-threaded Event Reconstruction with a 48-core “Magny Cours” computer

David Lawrence JLab

Jun 15, 2010

```
processor   : 0
vendor_id   : AuthenticAMD
cpu family  : 16
model           : 9
model name  : AMD Opteron(tm) Processor 6174
stepping    : 1
cpu MHz     : 2200.024
cache size  : 512 KB
physical id     : 0
siblings    : 12
core id         : 0
cpu cores   : 12
apicid          : 0
fpu         : yes
fpu_exception   : yes
cpuid level     : 5
wp          : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx
                  fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm 3dnowext 3dnow
                  constant_tsc nonstop_tsc pni cx16 popcnt lahf_lm cmp_legacy svm extapic cr8_legacy
                  altmovcr8 abm sse4a misalignsse 3dnowprefetch osvw
bogomips    : 4403.52
TLB size    : 1024 4K pages
clflush size    : 64
cache_alignment  : 64
address sizes   : 48 bits physical, 48 bits virtual
power management : ts ttp tm stc 100mhzsteps hwpstate [8]
```

Default settings using ALT1 track fitter

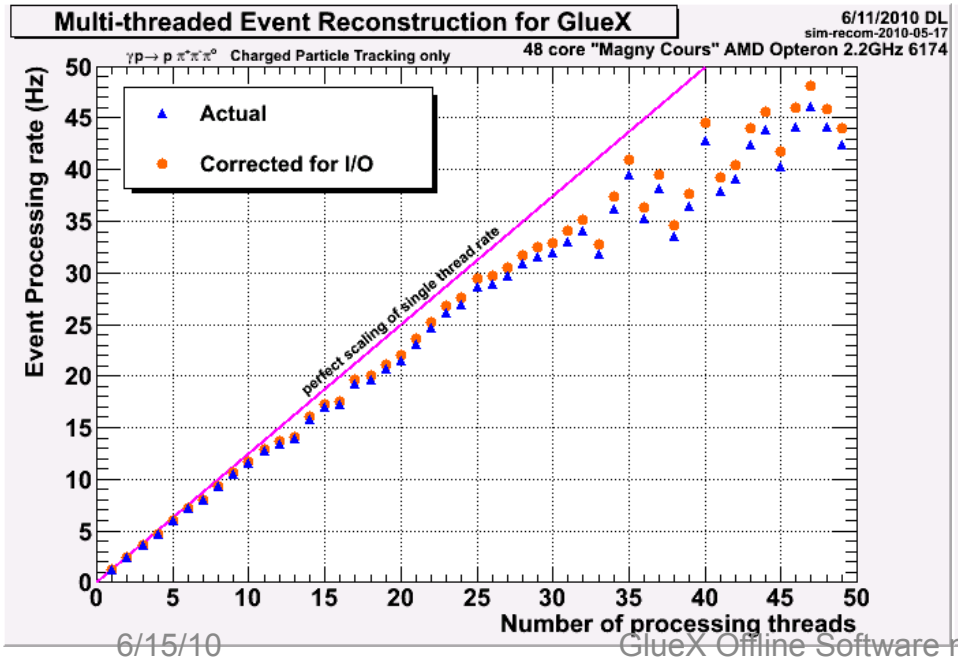
A sample of 1000 $\gamma p \rightarrow p\,\pi^+\pi^-\pi^o$ generated using *genr8* was used to test the multi-threading rate capability of a new 48 core AMD machine on temporary loan to JLab for testing.

Only charged particle tracking was done, but to the level of producing *DChargedTrack* objects (i.e. multiple mass hypotheses with PID decision).

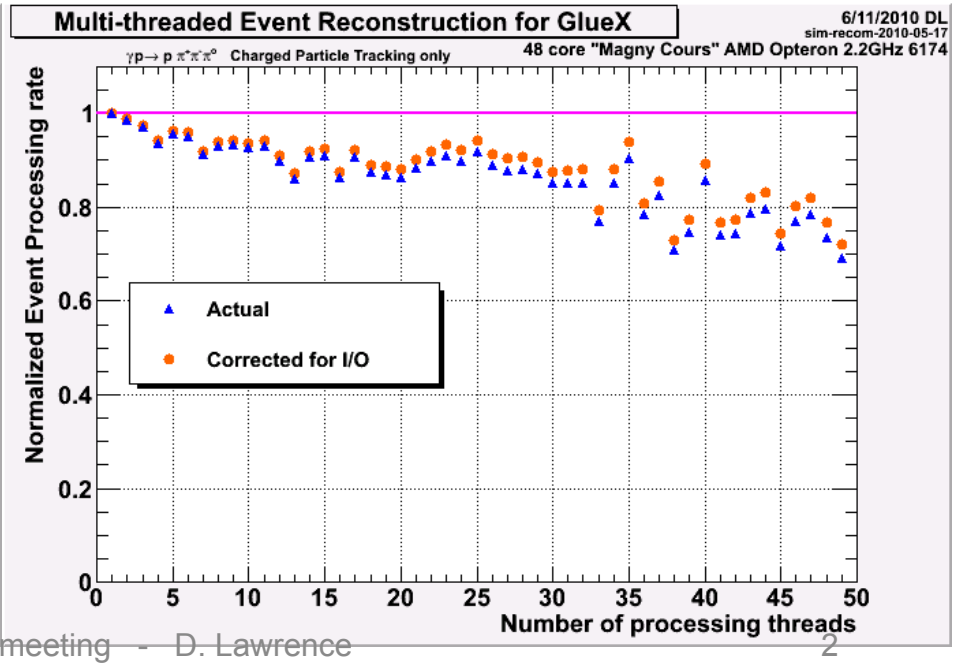
No output files were produced

janarate plugin was used to determine the average event processing time

Raw rate



Normalized to Single thread rate

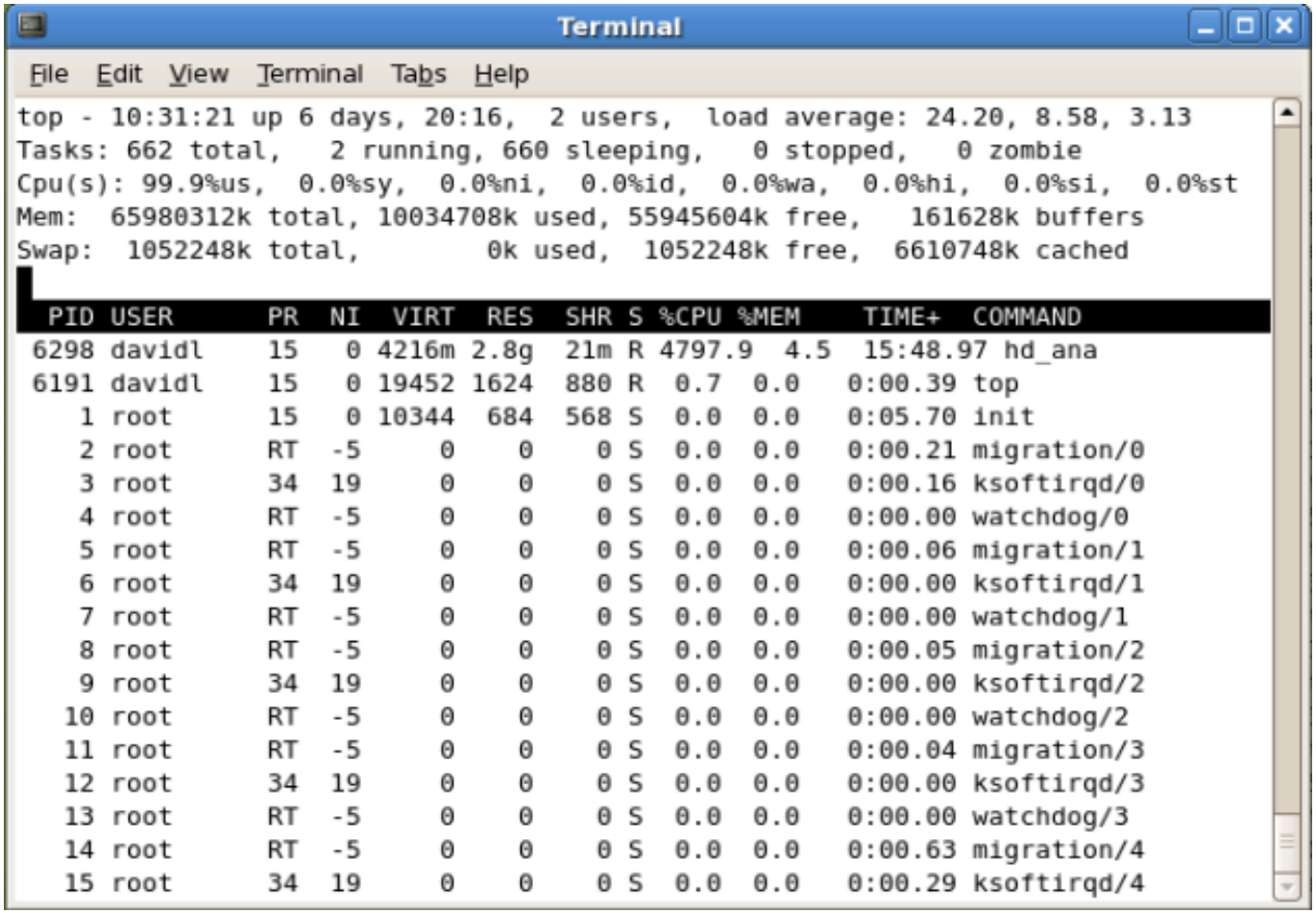


top

Memory usage is between 3 GB and 4 GB for single process running with 48 processing threads

CPU completely utilized in user space (like we want it!)

Negligible time spent sleeping in mutex locks or system calls



Alternate B-field Maps

(evidence of memory contention?)

Neither the Fine Mesh field nor the Constant field maps use any interpolation. The Fine Mesh map, however, accesses a large array in memory.

Raw rate

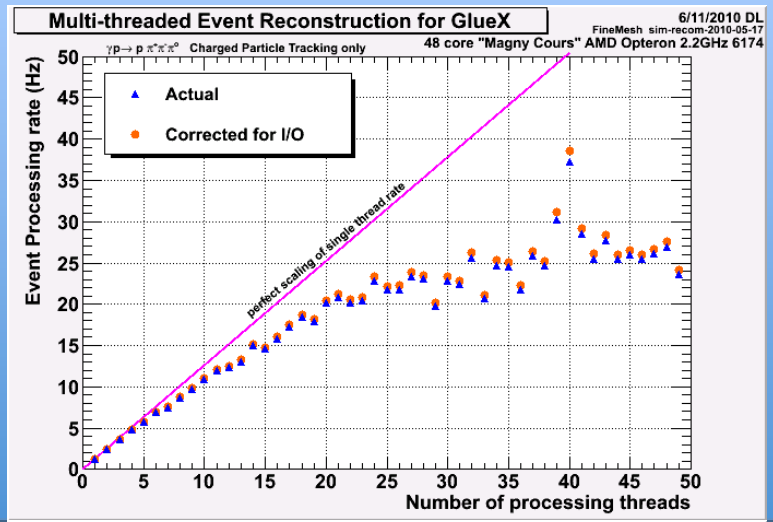
Normalized to Single thread rate

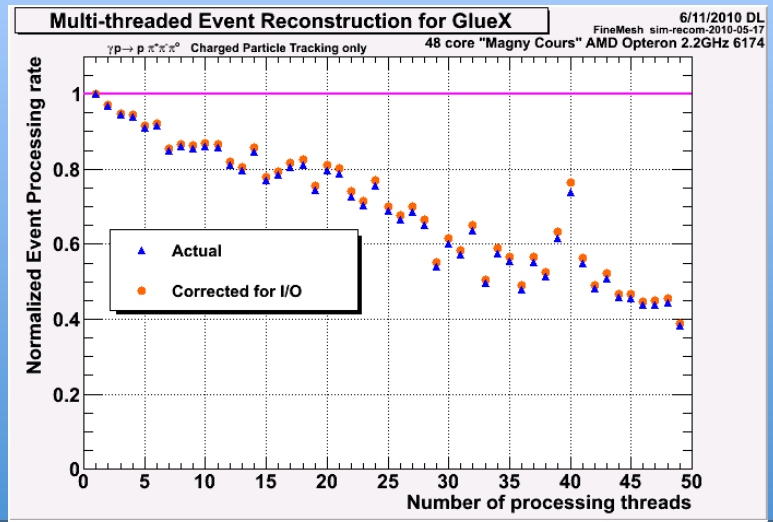
Fine Mesh B-field Map

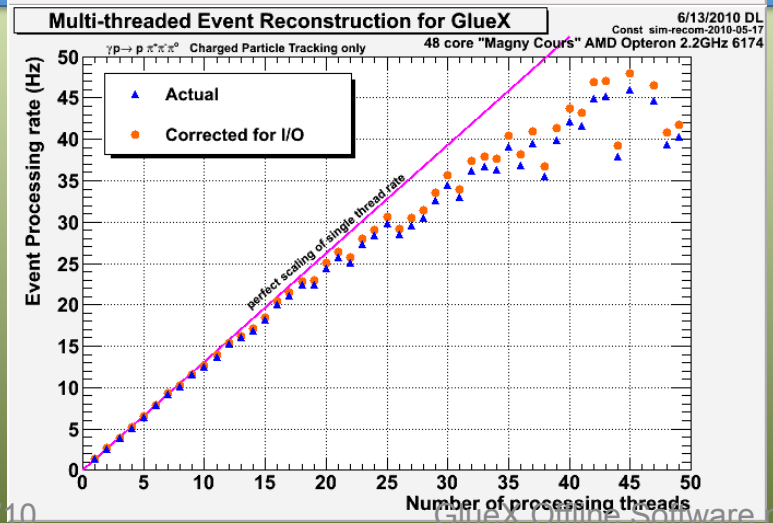
Constant B-field Map

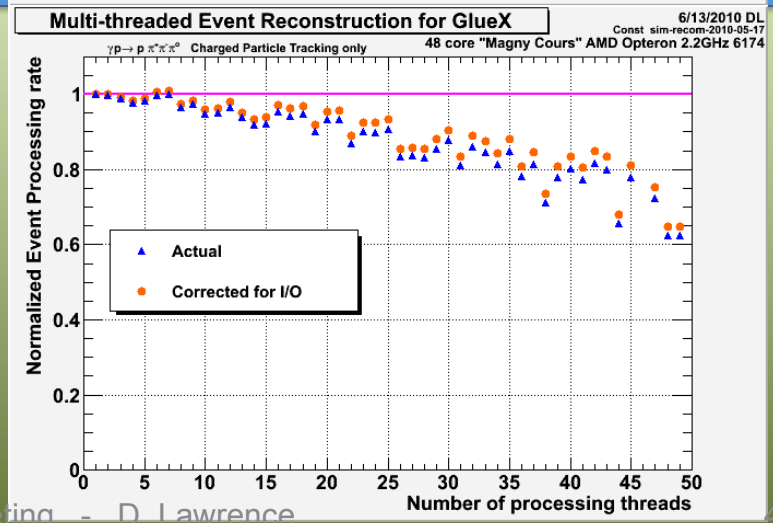
Big memory footprint

No memory footprint





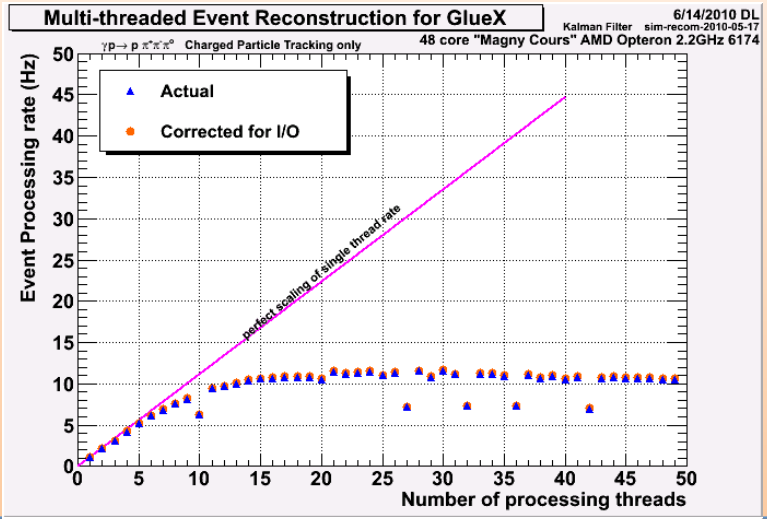


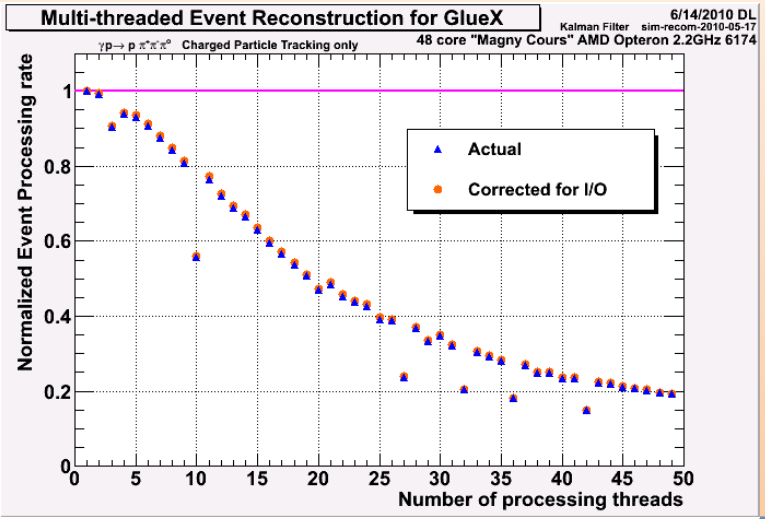


Kalman Filter

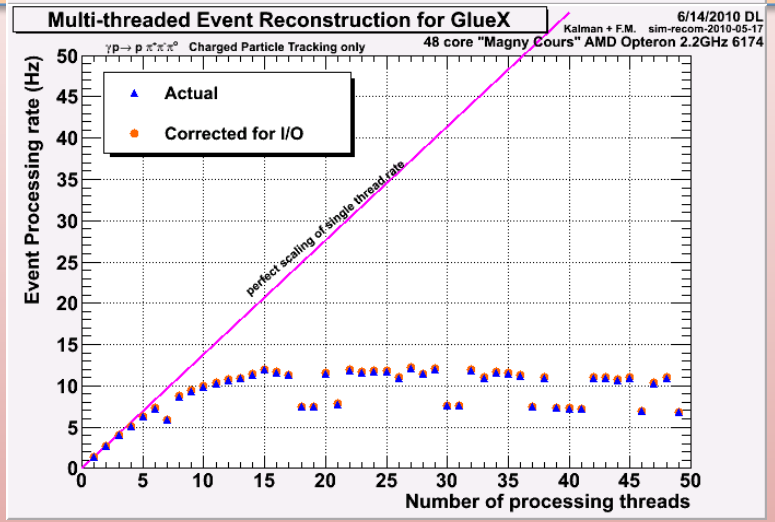
The Kalman Filter seems to be be particularly adept at hitting the bottleneck

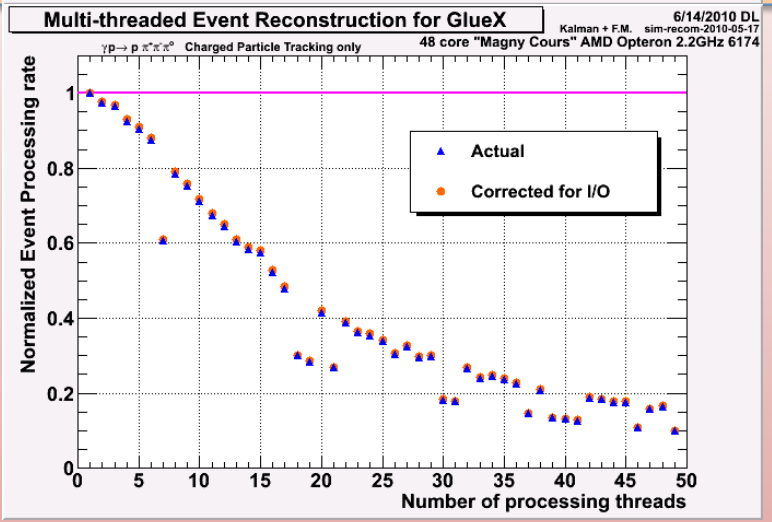
Fine Mesh B-field Map





Nominal B-field Map

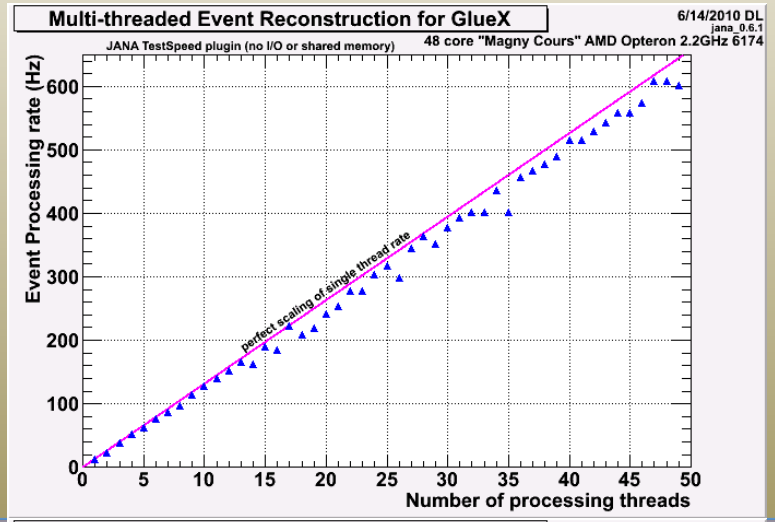


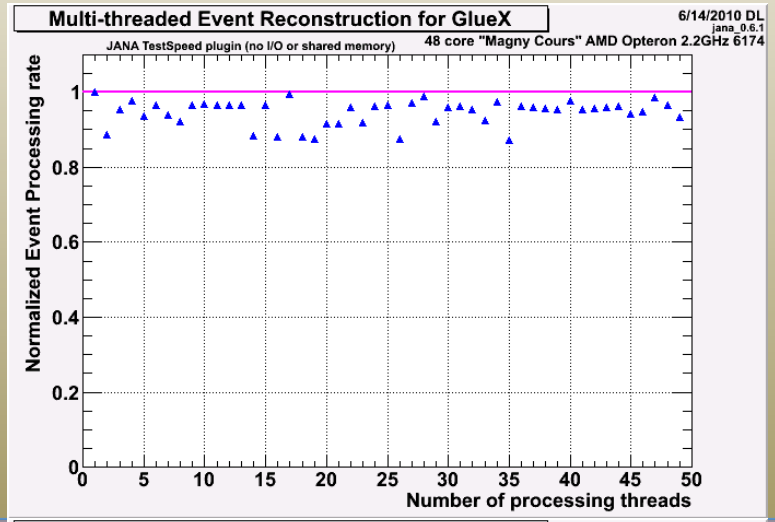


JANA's *TestSpeed* plugin

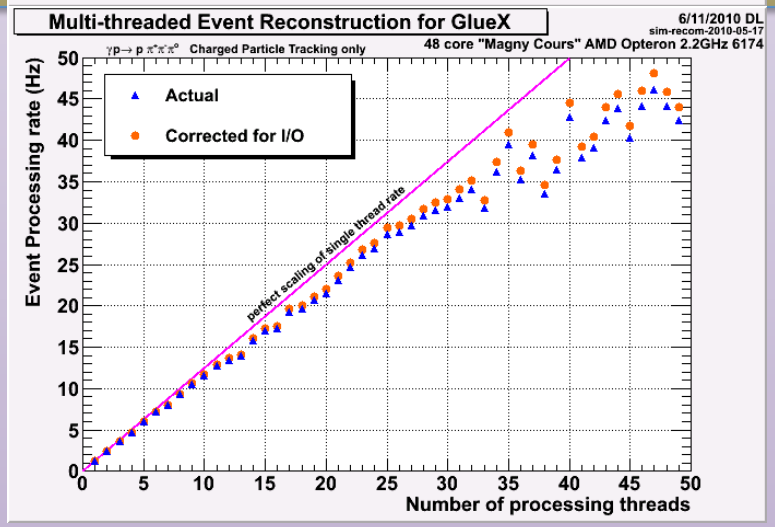
JANA's *TestSpeed* plugin fakes data input and uses no shared memory constructs (such as magnetic field or material maps).

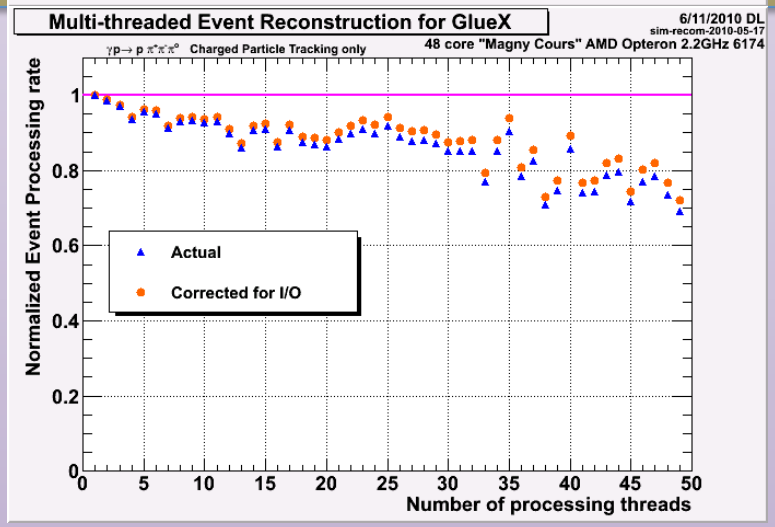
TestSpeed plugin





ALT1 Fitter





Why Multi-threading?

Multi-core processors are already here and commonly used. Industry has signaled that this will be the trend for the next several years. Consequence: Parallelism is required

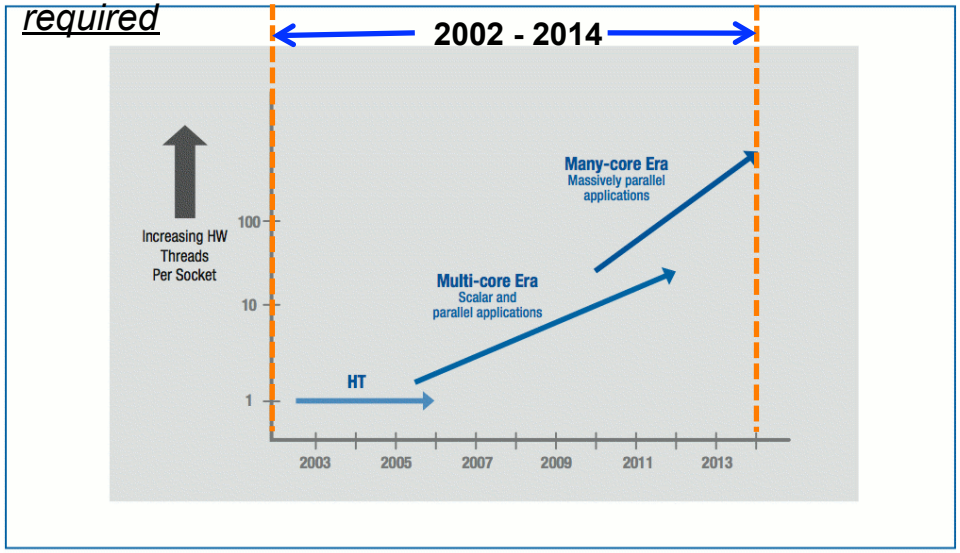
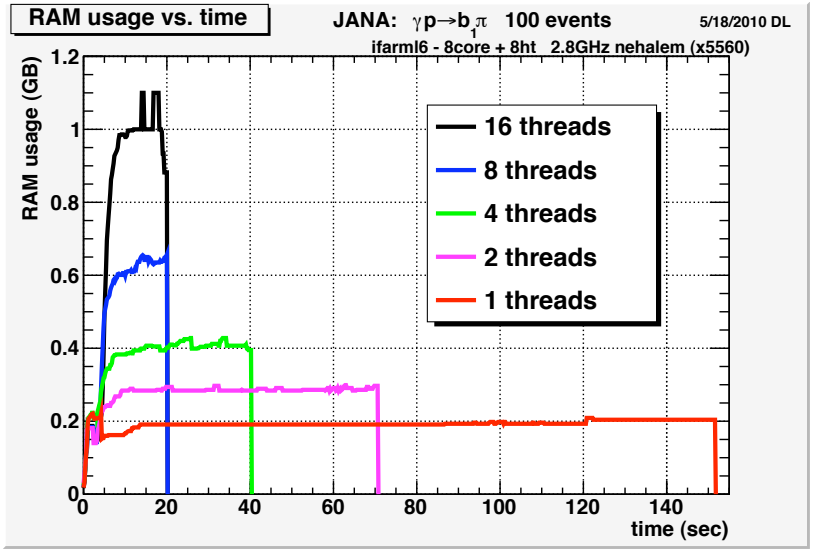
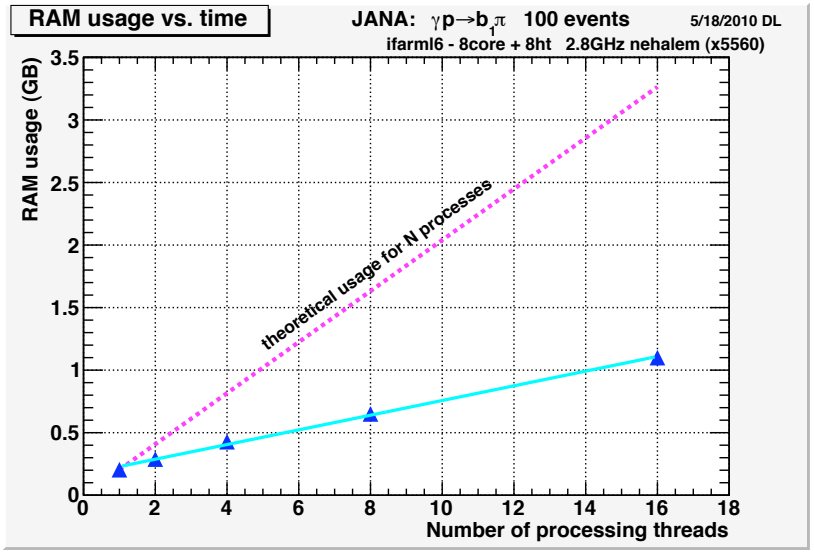


Figure 1: Current and expected eras of Intel® processor architectures





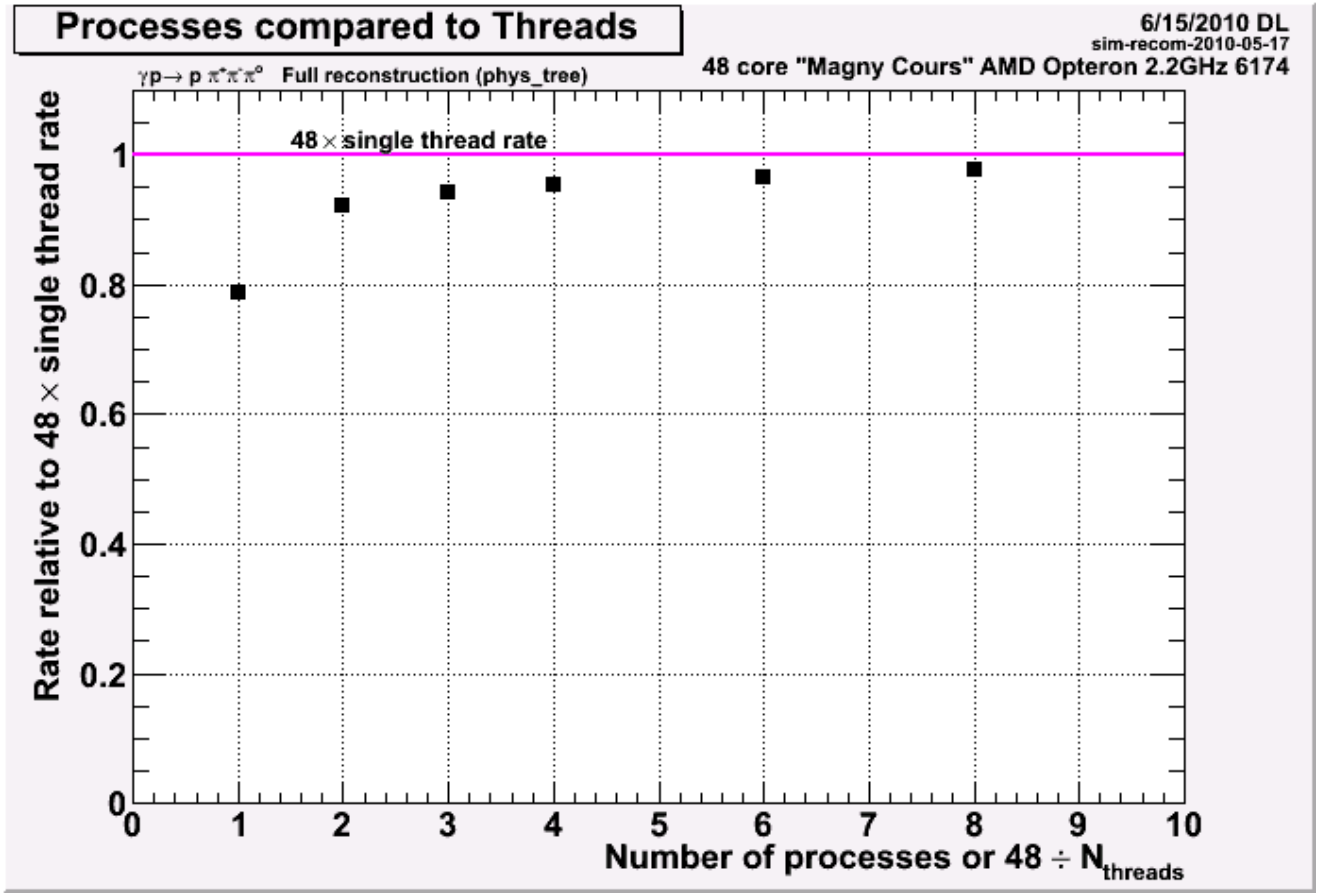
Maintaining a fixed memory capacity per core will become increasingly expensive due to limitations on the number of controllers that can be placed on a single die (#pins).

Example: Memory accounts for about 10%-25% of system cost today but that will increase by as much as 5%/year over the next several years leading to 50% of system cost going toward RAM

Using multiple processes

Simultaneously run multiple, independent processes, but each with multiple threads such that all cores have exactly 1 thread.

Performance improves with fewer threads and more processes for the heavily CPU bound job of full reconstruction (I/O bound jobs will see a penalty for multiple I/O streams)



Summary

- Tests done on a 48-core AMD “Magny Cours” machine showed rough scaling of event processing rate with number of threads
- Some evidence to suggest memory contention may be an issue.
- Kalman Filter may provide the right laboratory for understanding a key bottleneck (need to use oprofile?)
- Multiple processes appears to give an overall performance boost