GlueX-doc-2277-v1

Hall-D Computer Resources Usage

David Lawrence

July 23, 2013

Abstract

This document presents results from tests run on three JLab computers to measure the resource requirements for a typical offline processing job of GlueX production data. Tests were done using multi-threaded JANA processes to also verify the scaling of event rates as well as hyper-thread vs. physical core performance. Resident memory usage was measured for reconstruction jobs to be at about 1/3 the rate of what would be required for multiple processes.

1 Testing platforms and conditions

The testing was done using three computers that were temporarily removed from the JLab Scientific computing farm so that exclusive and interactive access could be granted. Details on the computers are given in table 1.

Table 1: Specifications for computers used in testing.

node	processor	RAM	OS
farm12032	2.0GHz Intel Xeon E5-2650 (Sandy Bridge)	32GB	CentOS 6.2 (x86_64)
farm11020	2.0GHz AMD Opteron 6128	64GB	CentOS 6.2 (x86_64)
farm11008	2.533GHz Intel Xeon E5630 (Westmere)	24GB	CentOS 6.2 (x86_64)

Tests were done using events similar to what is expected with the real data. Specifically, the Hall-D *sim-recon* code at svn revision r11231 was used. Events were generated using the PYTHA-based program *bggen*. The events were simulated and smeared using the *hdgeant* and *mcsmear* programs respectively. The resulting data file was passed through the *hd_eventfilter* program which discards events that are not expected to pass the L1 electronic trigger. This resulted in a file $filtered.hddm$ that contained the same information expected for real data, plus some Monte Carlo "truth" information that will not be in the actual data stream. Another difference with real data is that this

file was in HDDM format while the real data will be in EVIO format. This is relevant only in the limit where we become I/O bound rather than CPU bound.

Testing was done by having a script run the *hd_root* program repeatedly, with a different number of threads each time. Three plugins were used:

danarest invokes the actual reconstruction and write the reconstructed events out to a REST[1] file.

janarate records the event processing time and total integrated rate of every event into a ROOT tree

janadot records call graph and processing time spent in each JANA factory to help profile where the time is being spent.

The resource usage was recorded by using the standard */usr/bin/time -v* tool. This provides information not only on the time spent by the process, but the resident memory usage itself. It should be noted that there is apparently a long standing bug in the *time* tool that causes it to report resident memory usage that is about 4 times larger than it should. See figure 1 for details.

2 Memory Usage

Figure 2 shows the memory usage as a function of the number of processing threads for each of the 3 computers tested. The memory usage of a multi-threaded process can be compared to what one should expect for running multiple single-threaded processes using the numbers extracted from the linear fits shown on the plots. For example, the top plot indicates that a process with a single processing thread would use $565MB + 160MB = 725MB$. In the limit of a large number of threads, the multi-threaded memory usage is less than 1/4 of what multi-process would use. There more relevant numbers would be the total memory usage expected for running 32 processes vs. running a single 32-thread process. Table 2 gives these estimates for reconstruction jobs.

Table 2: Memory usage for 32 processes reconstruction processes or threads on a multi-core computer. Multi-threading requires significantly less RAM

node	RAM usage multi-process (estimated)	RAM usage multi-thread
farm12032	22.7 GB ± 0.9 GB	5.55 GB ± 0.04 GB
farm11020	20.7 GB ± 0.8 GB	5.55 GB ± 0.04 GB
farm11008	20.0 GB ± 0.9 GB	5.53 GB ± 0.04 GB

[1]REST is an HDDM format specifically designed to hold summary event information similar to a traditional DST.

```
        Command being timed: "hd_root -o /scratch/pbs/davidl/farm12032/2thread_j
anadot.root -PPLUGINS=danarest,janarate,janadot -PNTHREADS=2 -PEVENTS_TO_KEEP=20
000 -PJANA:BATCH_MODE=1 ./filtered.hddm ./filtered.hddm ./filtered.hddm ./filter
ed.hddm ./filtered.hddm ./filtered.hddm"
        User time (seconds): 2416.18
        System time (seconds): 21.04
        Percent of CPU this job got: 200%
        Elapsed (wall clock) time (h:mm:ss or m:ss): 20:13.95
        Average shared text size (kbytes): 0
        Average unshared data size (kbytes): 0
        Average stack size (kbytes): 0
        Average total size (kbytes): 0
        Maximum resident set size (kbytes): 3559312
        Average resident set size (kbytes): 0
        Major (requiring I/O) page faults: 0
        Minor (reclaiming a frame) page faults: 3486326
        Voluntary context switches: 186505
        Involuntary context switches: 32765
        Swaps: 0
        File system inputs: 1727048
        File system outputs: 59736
        Socket messages sent: 0
        Socket messages received: 0
        Signals delivered: 0
        Page size (bytes): 4096
        Exit status: 0
```

Figure 1: Example output from the */usr/bin/time -v* tool showing the command being analyzed and the resources it used. The value of "Maximum resident set size (kbytes)" is used to determine the memory used by this processes. A long standing bug causes this value to be scaled up by the "Page size" in kB. In this particular example, the actual memory usage 899 MB. The values reported in this document have been scaled by the page size to correct for this bug.

Table 3 shows total memory usage for 32 core computers running GlueX simulation. The values in the table were scaled up to what would be needed to fully utilize all 32 cores. At this point in time, the *hdgeant* program must be run single threaded so the memory usage reflects this. It is most likely that simulation farm jobs will be designed such that *hdgeant* will be run with *mcsmsear* being run immediately after as part of the same job. It would then only make sense to run the *mcsmear* program single threaded as well. This would lead to a memory requirement for simulation nodes to have at least 12.7 GB. This requirement may change significantly as we transition to Geant4 which has recently added some multi-threading capabilities.

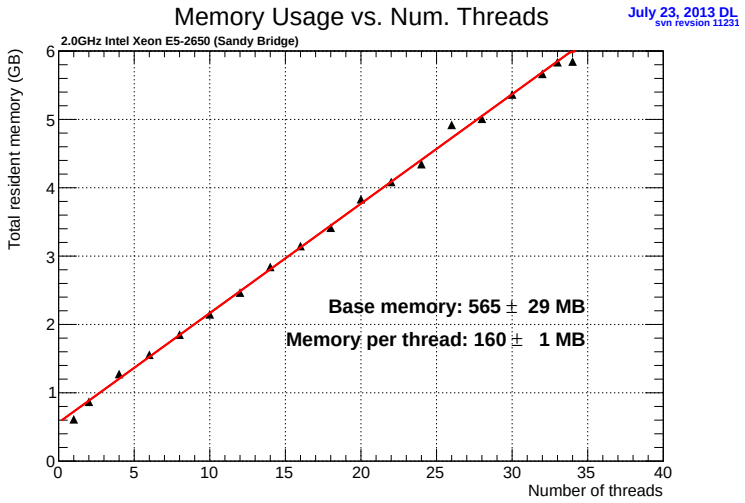
Table 3: Memory usage for 32 simulation processes or threads on a multi-core computer. Simulation jobs will likely include both *hdgeant* and *mcsmear* making the larger memory requirement the relevant one. See text for details.

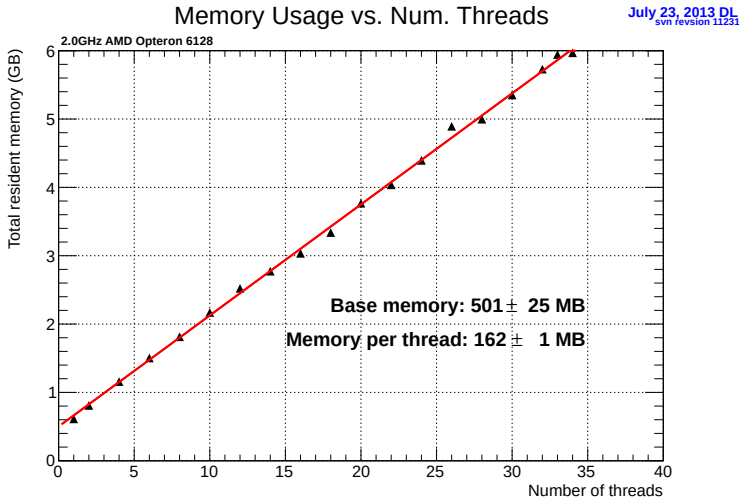
node	hdgeant	mcsmear (multi-thread)	mcsmear (multi-process)
farm12032	7.4 GB	1.1 GB	12.7 GB
farm11020	7.4 GB		
farm11008	7.1 GB		

Figure 3 shows the memory usage by the Monte Carlo smearing program *mcsmear*. This program performs second stage processing on Monte Carlo data to add additional effects and realistic detector resolutions. Its resource requirements must therefore be included in those needed for simulation. This test was limited to only the Sandy Bridge architecture and a few threads in order to identify the trend. As indicated in the caption of the figure, a fully utilized 32-core computer will require 1.1 GB of memory for multi-threaded processing and 12.7 GB of memory for multi-process. It should be noted that at the time of this writing, the first stage simulation done by *hdgeant* must be run multi-process as it has no multi-threading capability.

2.1 Raw Data Reconstruction Rates

Figures 4-5 show the reconstruction rates as a function of the number of processing threads for each of the three computers used in the current study. For the Intel processors the regions where hyper-threads were used are clearly visible for these plots. Separate linear fits were done to the regions dominated by physical cores(red) and hyper threading (blue). The green fits are to regions where the processor is over-subscribed with more threads than available cores+hyperthreads. The data from the two Intel computers indicate that hyper threads give the equivalent of about 25%-30% of a physical core. By contrast, the AMD processor shows less rate per physical core, but all physical cores appear to contribute equally even when all 32 cores are utilized. The end result is that AMD chip appears to have roughly equivalent processing power as the Sandy Bridge Intel chip when all cores are utilized. If not all cores are utilized however, the Intel processor will provide better compute capability to those processes that are running than the AMD processor.





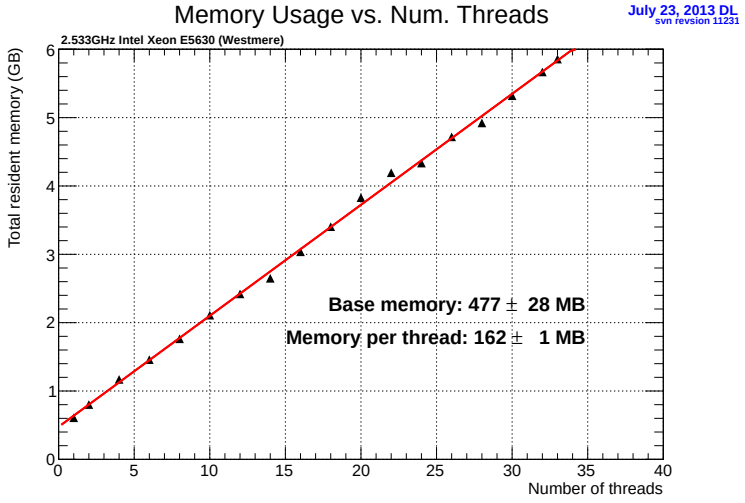


Figure 2: Memory usage vs. the number of threads for each of the 3 computers tested. The values of "Base Memory" and "Memory per thread" are the offset and slope of the linear fits shown. The total memory used by a single threaded process would be approximately the sum of these two numbers. This indicates memory usage by multi-threading rises at roughly 25% the rate that it does for multi-process.

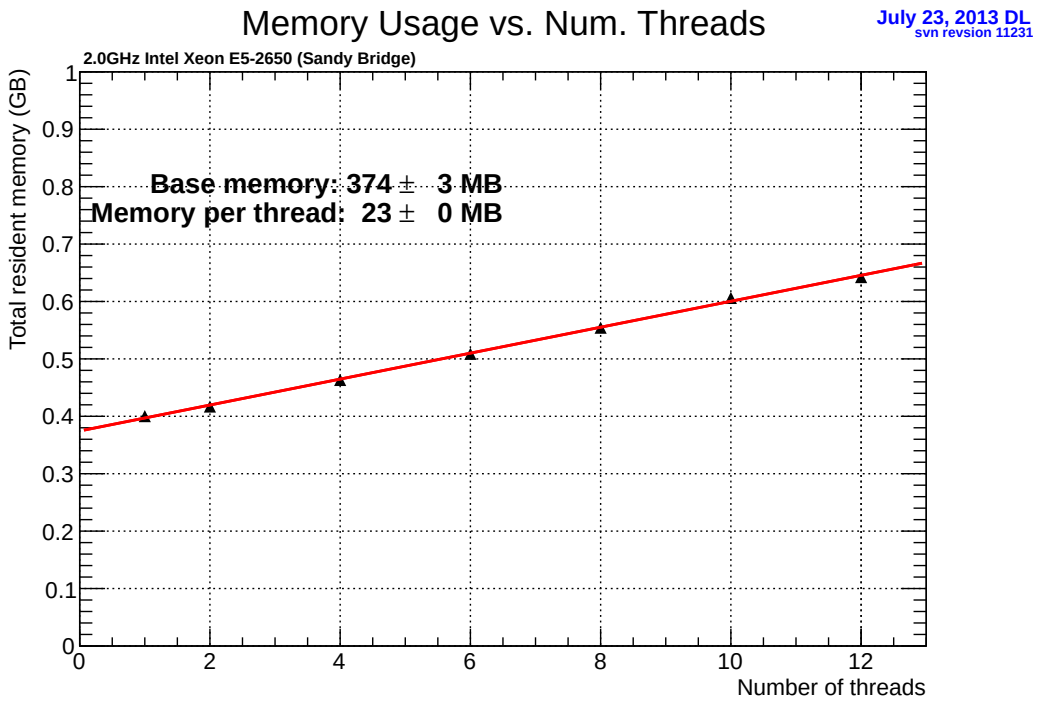


Figure 3: Memory usage vs. the number of threads when running the *mcsmear* program The program applies detector resolutions and is a necessary step in generating Monte Carlo data for GlueX. (It is not needed for processing real data). This exercise was not carried out to the full 32 threads, but extrapolation indicates only 374 MB + (32 threads$\times$23 MB) = 1.1 GB would be needed for multi-threaded operation while 32 threads$\times$(374 MB + 23 MB) = 12.7 GB would be needed for multi-process.

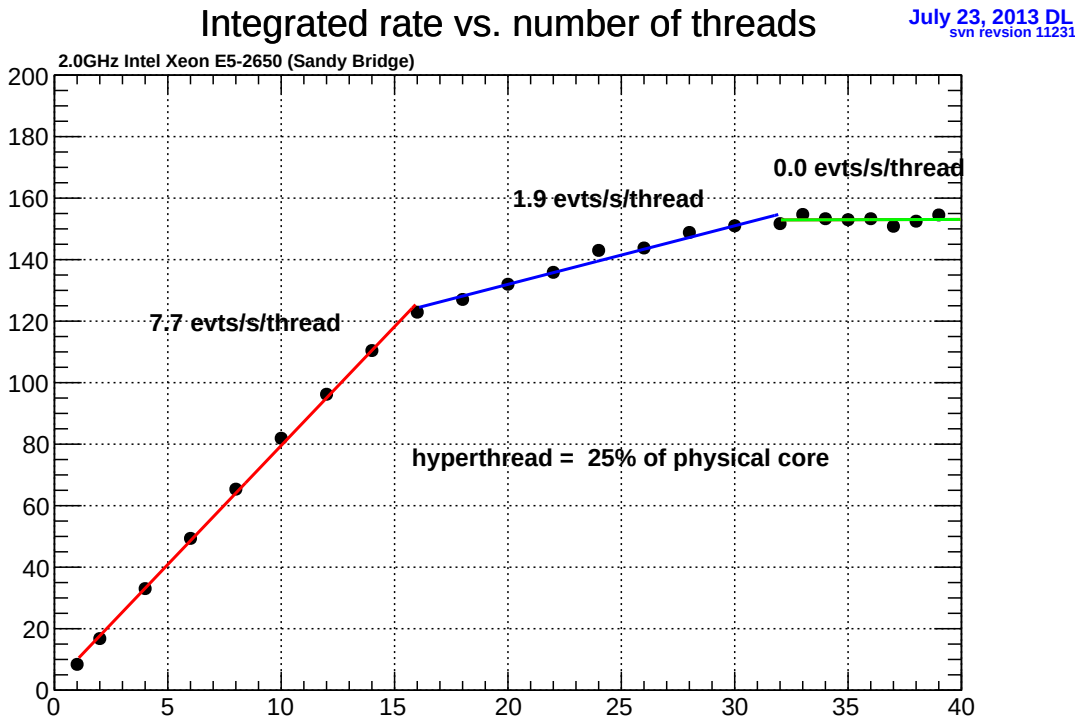


Figure 4: Event reconstruction rate vs. number of processing threads on a 16-core, 2.0GHz Intel Sandy Bridge computer. Hyper-threading was enabled and additional performance is achieved at a rate equivalent to about 25% of a physical core per hyperhread. This is determined by fitting the physical cores only region (red) and the hypthreads region (blue) to lines and comparing the slopes. The green line is in the region where the processor is over-subscribed and no additional performance benefit is seen from the extra threads.

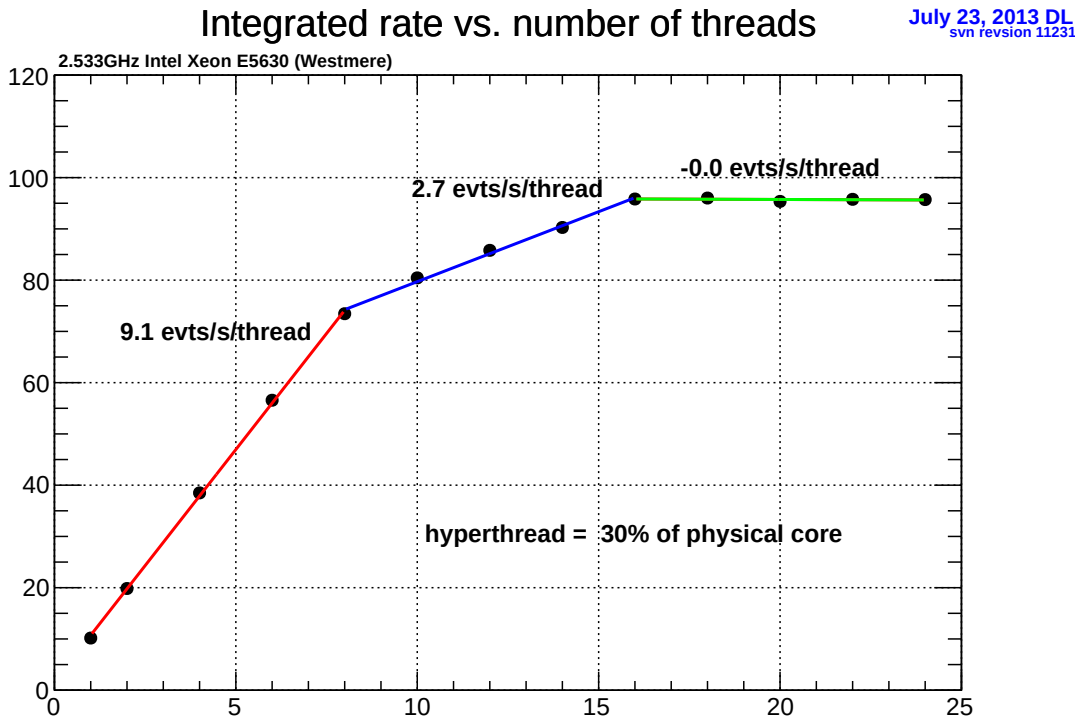


Figure 5: Event reconstruction rate vs. number of processing threads on a 8-core, 2.533GHz Intel Westmere computer. Hyper-threading was enabled and additional performance is achieved at a rate equivalent to about 30% of a physical core per hyperhread. This is determined by fitting the physical cores only region (red) and the hypthreads region (blue) to lines and comparing the slopes. The green line is in the region where the processor is over-subscribed and no additional performance benefit is seen from the extra threads.

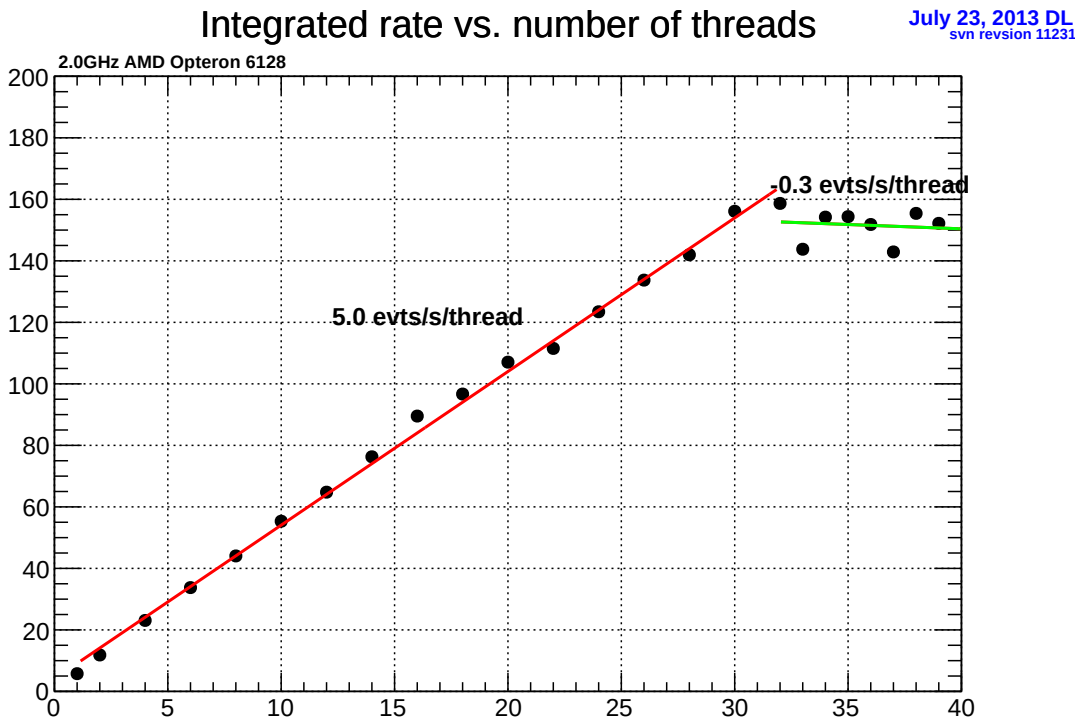


Figure 6: Event reconstruction rate vs. number of processing threads on a 32-core, 2.0GHz AMD Opteron computer. The green line is in the region where the processor is over-subscribed and no additional performance benefit is seen from the extra threads.

2.2 Summary

- Reconstruction on single physical core of 2.0 GHz Sandy Bridge: 130 ms/event
- Total reconstruction rate on 2.0GHz Sandy Bridge (16 cores + 16 HT): 150 Hz
- Total memory for 32 core computer doing reconstruction: 5.6 GB
- Total memory for 32 core computer doing simulation: 12.7 GB