Benchmarking high performance computing architectures with CMS' skeleton framework

Benchmarking high performance computing architectures with CMS' skeleton framework

E Sexton-Kennedy[1], P Gartung[1] and C D Jones[1]
[1]Fermilab, P.O.Box 500, Batavia, IL 60510-5011, USA

E-mail: sexton@fnal.gov

Abstract. In 2012 CMS evaluated which underlying concurrency technology would be the best to use for its multi-threaded framework. The available technologies were evaluated on the high throughput computing systems dominating the resources in use at that time. A skeleton framework benchmarking suite that emulates the tasks performed within a CMSSW application was used to select Intel's Thread Building Block library, based on the measured overheads in both memory and CPU on the different technologies benchmarked. In 2016 CMS will get access to high performance computing resources that use new many core architectures; machines such as Cori Phase 1&2, Theta, Mira. Because of this we have revived the 2012 benchmark to test it's performance and conclusions on these new architectures. This talk will discuss the results of this exercise.

1. Motivation

CMS has been continuously improving the thread efficiency of its multi-threaded applications. Since the time of our first report at ACAT in 2014[1] there has been significant improvements achieved by fixing concurrency issues in our legacy code and migrating it to the thread-friendly requirements of our multi-threaded framework. To see how this has improved our efficiency on conventional processors, see the contribution to the proceedings of this conference[2]. In 2016 CMS acquired access to a number of high-performance computing (HPC) systems. HPCs are a window into future computing architectures. CMS wanted to be sure that the our framework strategy based on Intel's Thread Building Blocks (TBB)[3] still worked on these systems. This paper reports on work done to evaluate our application performance on advanced many core architectures made available to CMS in 2016. In the future, these machines may dominate our production resources.

2. Methodology chosen: emulation

In order to emulate how the fully ported, or fully thread friendly, CMSSW applications would perform on these HPC architectures, we've used a simplified multithreaded framework[4] which implements the full framework's capabilities and features but does not depend on potentially thread unsafe user code. Instead the user code is emulated by integrating sin(float x) for enough loops to match the time spent in the original algorithm. The benchmark application chosen had 489 Producers[1] and both the times and dependencies of the algorithms were taking from the full reconstruction of 2011 on data that contained about 30 interactions per crossing. That data complexity matches the average complexity of data taken in LHC run 2 so far. This is important since it is known that the more complex a problem, the more it benefits from increasing the parallel resources used to solve that problem[5].
Once the above data from the full application is fed into the emulation of the multi-threaded framework, execution of the emulation on the different architectures can be compared for performance

[1] Producers as explained in reference 1, are algorithms that produce event data model objects

measurements when giving the emulation different amounts of concurrent resources. In the results plots of section 4, the emulation is run varying the amount of computing resources allocated to the problem. More details will be given in that section.

3. Measurement machines

In 2016 CMS was granted access to the following machines in order to conduct this study:

1. KNL from Fermilab's scientific computing division: Xeon Phi 7210: 1.3GHz with 64 cores and 256 hardware threads. per node, 96GB Memory/node or 1.5GB/core. The batch system is PBS.
2. ALCF BG/Q: It uses a PowerPC A2 1.6GHz with 16 cores and 64 hardware threads per node, 16GB Memory/node or 0.25GB/thread. The batch system is COBALT.
3. NERSC Edison: It uses an Ivy Bridge, 2.4GHz chip with 24 cores per node, 64GB Memory/node or 2.67GB/core. The batch system is SLURM.

CMS was also granted access to NERSC Cori Phase 1. It uses Haswell 2.3 GHz with 32 cores per node, 128GB Memory/node. It's batch system is also SLURM. Unfortunately Cori was taken down for it's upgrade to Cori Phase 2 before the measurements could be completed. Ivy Bridge and Haswell are in the same tick-tock Intel cycle and have comparable single threaded performance. The major difference between the two are performance per watt. Because of this we didn't aggressively pursue finding another source for the Haswell chip set.

4. Results

4.1. KNL results

On the KNL machine, 3 different resource configurations are measured as a function of the number of concurrent TBB threads or processes. The red curve of Figure 1 gives the resulting throughput as a function of the number of concurrent processes. The blue curve gives the resulting throughput as a function of the number of concurrent threads when the number of threads equals the number of concurrent event streams fed to the emulation. On the KNL you can see that there is no threading penalty relative to heavy weight processes, out to 64 cores, very little penalty (~2%) out to 128, and still acceptable losses (5%) out to 256. These numbers match the architecture descriptions of the KNL in section 3. The green curve measures throughput when the emulation is given additional threading resources beyond the number of events it is fed, in the ratio of 5 threads for every 4 events. Notice that the CMS multithreaded framework is able to use the additional resources to increase event throughput relative to the full process measurement. The limitation on the number of threads useable by one event comes from the data dependencies between algorithms and is not inherent in the framework itself.

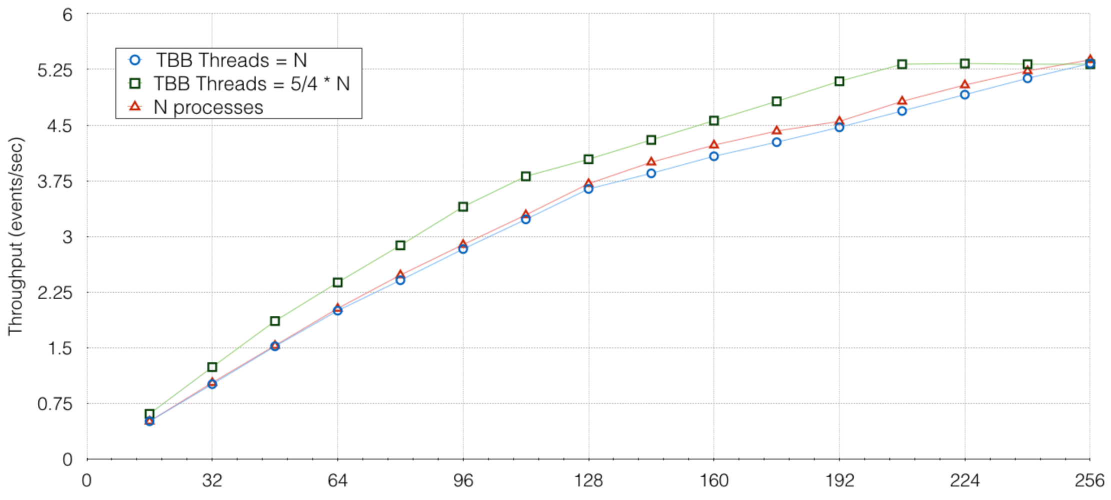

Figure 1. Total Throughput vs. N Processors or N Streams on KNL

4.2. ALCF BG/Q results

On the ALCF BG/Q machine, one can see from Figure 2 that it is clear that only the c16 mode is relevant for our application. The mode parameter is specified on the batch submission command line, and specifies the number of MPI ranks per node. In that mode we see perfect throughput agreement whether one process is using 16 threads or there is 16 concurrently running processes. At higher thread counts it is still possible to gain throughput by adding additional event streams. The green curve again measures throughput when the emulation is given additional threading resources beyond the number of events it is fed; and again the framework is able to use those additional resources.

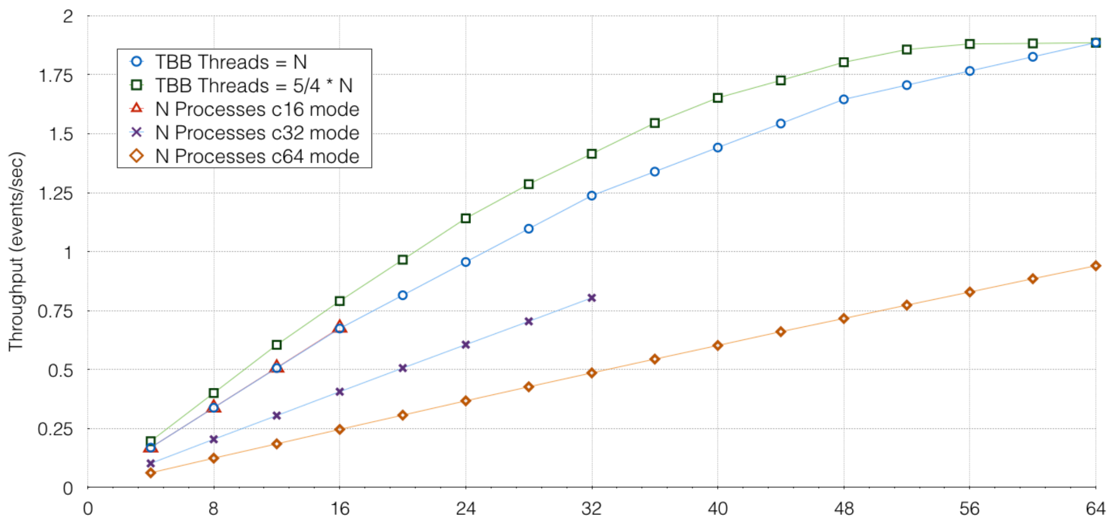

Figure 2. Total Throughput vs. N Processors or N Streams on ALCF BG/Q

4.3. NERSC edison results

On the Edison machine at NERSC multiple concurrent process throughput matches single process multiple event threads out to 20 simultaneous events. Beyond that there is some infrastructure limitation that would have to be tracked down and fixed or we would restrict our application to lower stream counts on this machine. Also note that out to 20 simultaneous streams when 25 threads are used, the green curve shows a ~15% improvement over the simultaneous multi-process case.

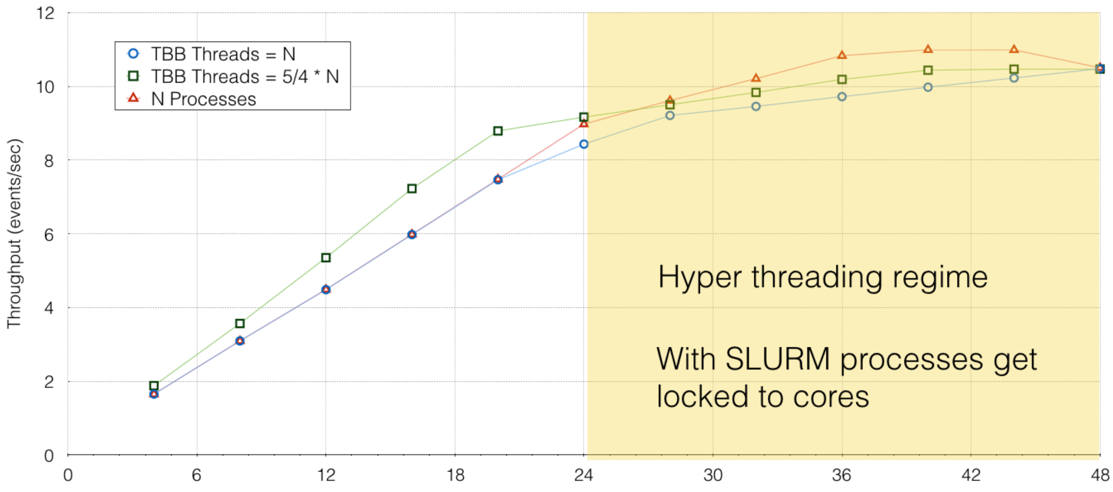

Figure 3. Total Throughput vs. N Processors or N Streams on Edison at NERSC

5. Conclusions and plans

The TBB technology underlying the CMS multi-threaded framework has been shown to scale to supercomputing architectures. Allocating more then one thread per event results in a 15% throughput benefit. Work has begun to test the functioning and scalability of the full CMSSW stack working on the KNL system. Once this is done we can investigate the optimal balance of memory consumption vs. throughput on this architecture.

6. References

[1] Elizabeth Sexton-Kennedy, *et al* J.Phys.Conf.Ser. 608 (2015) no.1, 012034 “Implementation of a Multi-Threaded Framework for Large-Scale Scientific Applications" DOI: 10.1088/1742-6596/608/1/012034
[2] Chris Jones, *et al* “CMS Event Processing Multi-core Efficiency Status” *To be published in the proceedings of CHEP 2016, San Francisco, 2016*
[3] TBB website https://www.threadingbuildingblocks.org
[4] https://github.com/Dr15Jones/toy-mt-framework
[5] Gustafson’s law: https://en.wikipedia.org/wiki/Gustafson's_law

Acknowledgments

This work was supported by Fermilab; Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

We would also like to thank the institutions who either loaned us or granted us time on their machines. This includes the HPC department of the FNAL SCD for their loan of a KNL machine.

The National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility, Contract No. DE-AC02-05CH11231

The Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357