Multi-threaded ATLAS Simulation on Intel Knights Landing Processors

Steven Farrell[1], Paolo Calafiura[1], Charles Leggett[1], Vakhtang Tsulaia[1], Andrea Dotti[2], on behalf of the ATLAS Collaboration

[1]Lawrence Berkeley National Laboratory, [2]SLAC National Accelerator Laboratory

E-mail: SFarrell@lbl.gov

Abstract. The Knights Landing (KNL) release of the Intel Many Integrated Core (MIC) Xeon Phi line of processors is a potential game changer for HEP computing. With 72 cores and deep vector registers, the KNL cards promise significant performance benefits for highly-parallel, compute-heavy applications. Cori, the newest supercomputer at the National Energy Research Scientific Computing Center (NERSC), was delivered to its users in two phases with the first phase online at the end of 2015 and the second phase now online at the end of 2016. Cori Phase 2 is based on the KNL architecture and contains over 9000 compute nodes with 96GB DDR4 memory. ATLAS simulation with the multithreaded Athena Framework (AthenaMT) is a good potential use-case for the KNL architecture and supercomputers like Cori. ATLAS simulation jobs have a high ratio of CPU computation to disk I/O and have been shown to scale well in multi-threading and across many nodes. In this paper we will give an overview of the ATLAS simulation application with details on its multi-threaded design. Then, we will present a performance analysis of the application on KNL devices and compare it to a traditional x86 platform to demonstrate the capabilities of the architecture and evaluate the benefits of utilizing KNL platforms like Cori for ATLAS production.

ATL-SOFT-PROC-2017-017
24 January 2017

1. Introduction

In the multi-core computing era, processor chip trends such as increasing core multiplicity, decreasing memory per core, and increasing importance of vector processing are changing the way scientific software developers write efficient, scalable code. Modern computing devices such as Intel's Xeon Phi line of many-core processors are good examples of what will be used more frequently in high performance computing facilities. These devices are best utilized with highly-parallel applications, so scientific computing models must adapt for greater concurrency and intelligent usage of memory resources.

High energy physics (HEP) experiments such as ATLAS[1] are no exception to this paradigm shift. Particle collision data is typically trivially parallelizable, but production software such as the Athena framework[2] have historically been written for sequential processing. In order to ensure that ATLAS can efficiently utilize modern computing devices and devices of the future, a large campaign is underway to adopt a multi-threading concurrency model for parallelism and efficient use of memory resources[3][4]. ATLAS simulation is the most advanced use-case for multi-threading, with a nearly complete configuration working and performing well on traditional Intel Xeon devices.

In this paper we will share results and experience preparing the ATLAS simulation software for the Knights Landing generation of Intel Xeon Phi processors. Section 2 gives a brief overview of the hardware used. Section 3 details the multi-threaded ATLAS simulation application. Performance results on Xeon and Xeon Phi machines are then given in Section 4. Ideas for future work and conclusions are given in Section 5.

2. Intel Xeon Phi processors

Current state-of-the-art processors for high-performance computing offer a wide array of capabilities and challenges. Devices such as FPGAs and general-purpose GPUs offer a high degree of parallelism with low power consumption for effective throughput. However, both of these devices use highly specialized programming models and have challenging constraints on memory capacity and data bandwidth. In response, Intel has been pursuing an alternate model that promises high performance with ease of use: the Xeon Phi product line.

Intel Xeon Phi processors are built with Intel's Many-Integrated-Core (MIC) architecture. General features of the product include high core multiplicity, deep vector registers, and low power consumption (relative to Xeon devices). Xeon Phi chips run a Linux OS, making them substantially easier to use than FPGAs and GPUs.

The current (2nd) generation of Xeon Phi processors is codenamed Knights Landing (KNL). KNL chips are the first release of the product line to offer full x86 binary compatibility and the first which can be installed as a host device or as a coprocessor. They are available with up to 72 Airmont cores and 4-way hardware threads, giving a maximum of 288 threads of execution. For SIMD parallelism, KNL devices have two 512-bit vector units per core and support AVX-512 instructions. Finally, the KNL generation introduces a deeper memory hierarchy compared to previous releases, providing both traditional DDR4 RAM as well as 8-16 GB of on-package, high-bandwidth MCDRAM. The MCDRAM can be utilized as an additional addressable memory space ("flat" mode), as a transparent cache ("cache" mode), or as a mixture of both ("hybrid" mode).

Xeon Phi processors are well suited for high performance computing facilities. A number of planned supercomputers will be based on Xeon Phi processors. At NERSC, the Cori supercomputer will have 9,300 KNL nodes with 68 cores each (2.5 million possible threads of execution). The Theta system at Argonne National Lab will have over 2,500 KNL nodes as well and will be a stepping-stone machine for the massive future Aurora system. Aurora is planned for 2018 to have over 50 thousand nodes equipped with 3rd-generation Xeon Phi (codenamed Knights Hill) processors.

3. Multi-threaded ATLAS simulation

The ATLAS simulation application (G4Atlas) is used to produce simulated ATLAS data in the Athena production framework[6]. It has been used extensively in the ATLAS experiment for many years for data analysis. It uses the Geant4[7] particle simulation toolkit to model physics processes and detector response. Production is traditionally performed with sequential jobs or multi-process jobs in the AthenaMP framework[8]. In the latter case, worker processes are forked from the main process after initialization of the job and before the event loop. This procedure allows worker processes to implicitly share some memory pages via the Linux copy-on-write mechanism.

An effort is currently underway to migrate the ATLAS simulation application to a multi-threading processing model (G4AtlasMT) in the AthenaMT (Multi-threaded Athena) framework. AthenaMT, which is based on the Gaudi concurrent framework, uses Intel Threading Building Blocks (TBB) for task-based parallelism. It schedules algorithms to operate on event data as tasks to run concurrently on different threads. This model allows both inter-event and intra-event parallelism. The simulation application uses few algorithms, however, with most of

the computation work happening in one algorithm (G4AtlasAlg) which simply invokes Geant4. The result is that the G4Atlas runs effectively with only inter-event parallelism. Memory savings are achieved by sharing physics and geometry tables across threads within Geant4. An illustration of the AthenaMT algorithm scheduling model is shown in Figure 1.

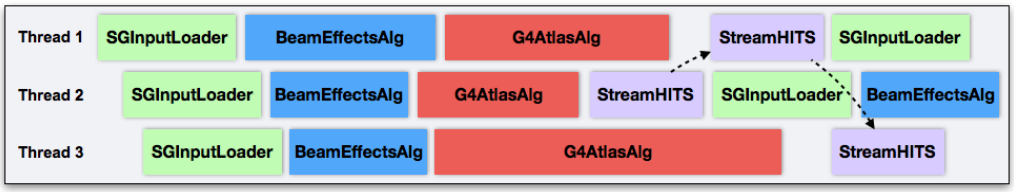

Figure 1. Illustration of worker thread processing in ATLAS multi-threaded simulation. SGInputLoader preloads some data from the input file to kickoff the event data flow. BeamEffectsAlg applies beam corrections and smearing to the input generated event. G4AtlasAlg is the main simulation algorithm which invokes Geant4. StreamHITS is the output stream algorithm which writes hit collections to the output file. StreamHITS is not cloned for concurrent processing. One instance serves all worker threads. Algorithm sizes are not shown to scale.

ATLAS simulation is potentially a good use-case for Xeon Phi processors. Relative to other ATLAS production workloads, simulation is CPU-heavy and uses little I/O. Not coincidentally, these are the same reasons that simulation is the primary ATLAS workload for supercomputers. The support for multi-threading is expected to be a powerful advantage in running effectively in the constrained memory environment of Xeon Phi cards. However, some challenges are expected as well. It is well known that vectorization is essential for effective utilization of KNL processors, but ATLAS simulation code does not vectorize well. Also, the highly object-oriented nature of ATLAS and Geant4 code tends to result in large code size and poor memory access patterns, which could hurt performance on KNL devices.

4. Performance measurements

The runtime performance of G4AtlasMT was measured on both Xeon and Xeon Phi machines. For the Xeon measurements, both a 16-core Ivy Bridge machine (E5-2650 v2 @ 2.60GHz) and a Cori Phase 1 Haswell node (E5-2698 v3 @ 2.30GHz) were used. The Xeon Phi measurements were taken on a KNL testbed (7210 @ 1.30GHz) for Cori Phase 2.

The important performance metrics are the event throughput and the memory consumption (RSS), and the scaling of these metrics with the number of worker threads. Figures 2 and 3 show the measurements for the simulation of a $Z \to \tau\tau$ sample. On the Xeon, the throughput scales perfectly up to the physical number of cores on the machine (16), and small gain is seen in the hyper-threading regime. The memory consumption shows a nice gradual scaling with each additional worker thread adding only about 70 MB. On the Xeon Phi, good scaling is again seen up to the number of physical cores on the device (64), with substantial throughput gains seen in hyper-threading all the way up to the maximum 256 threads. As with the Xeon, the memory consumption on the Xeon Phi is gradual and linear, reaching about 14 GB when the device is fully loaded. For the sake of comparison, the scaling results for purely multi-process jobs are shown in Figure 4. The per-worker contribution to the memory consumption is about five times larger in multi-process jobs compared to multi-threaded jobs, a substantial reduction in memory footprint.

To test the scaling of G4AtlasMT in more extreme configurations, a single-muon particle gun sample was used. Whereas the $Z \to \tau\tau$ sample is representative of typical ATLAS simulation

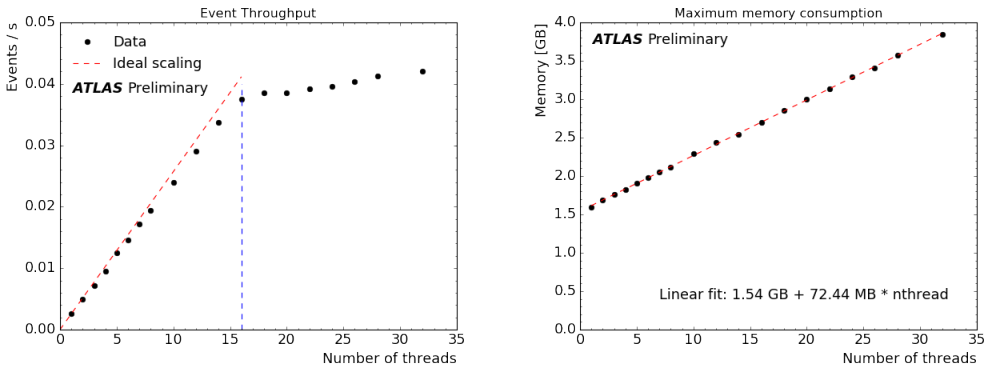

Figure 2. Event processing throughput (left) and memory consumption (right) on Intel Ivy Bridge Xeon for multi-threaded jobs with a $Z \to \tau\tau$ sample. The number of events processed is scaled as 50 times the number of threads [9].

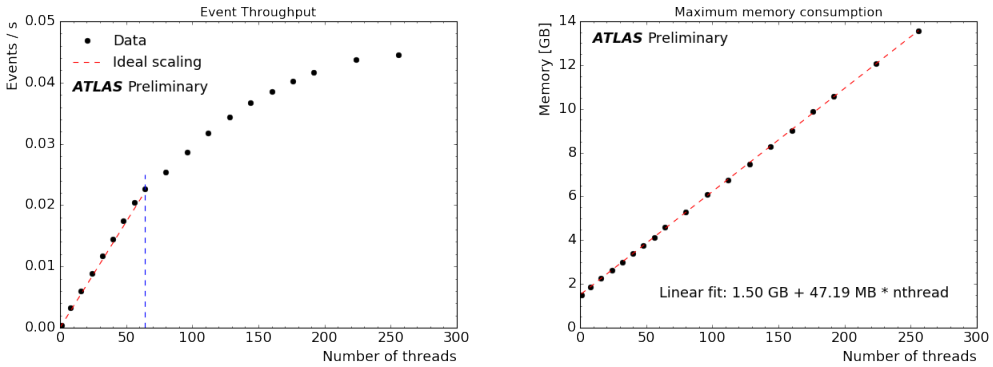

Figure 3. Event processing throughput (left) and memory consumption (right) on Intel KNL Xeon Phi for multi-threaded jobs with a $Z \to \tau\tau$ sample. The number of events processed is scaled as 10 times the number of threads [9].

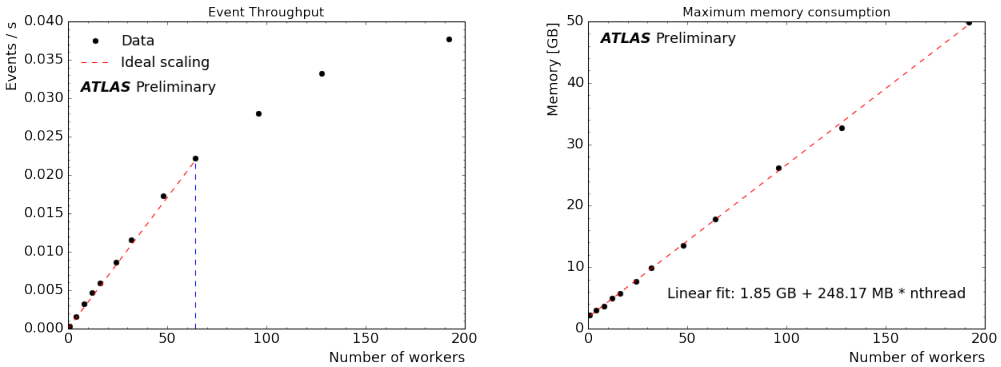

Figure 4. Event processing throughput (left) and memory consumption (right) on Intel KNL Xeon Phi for multi-process jobs with a $Z \to \tau\tau$ sample. The number of events processed is scaled as 10 times the number of threads [9].

production jobs and may take around 5 min per event, the single-muon sample typically takes less than one second to simulate one event. This applies more pressure to the scheduling system and other pieces of the framework infrastructure. Figures 5 and 6 show the results for the Xeon and the Xeon Phi, respectively. In this case, the throughput scales poorly above 180 threads. The source of the poor scaling was discovered to be the bottleneck in the sequential output stream which writes the simulated hit collections to the output file.

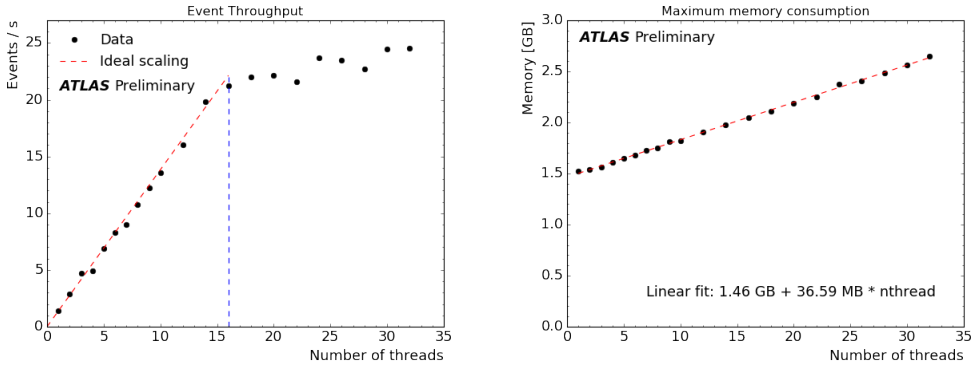

Figure 5. Event processing throughput (left) and memory consumption (right) on Intel Ivy Bridge Xeon with a single-muon sample. The number of events processed is scaled as 1000 times the number of threads [9].

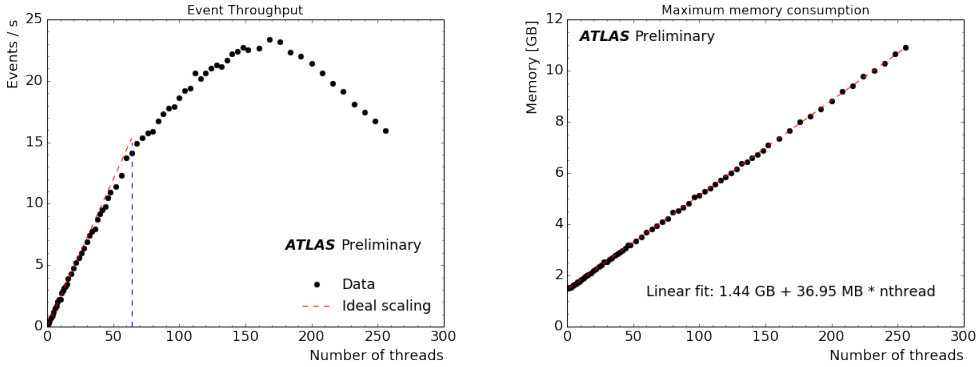

Figure 6. Event processing throughput (left) and memory consumption (right) on Intel KNL Xeon Phi with a single-muon sample. The number of events processed is scaled as 1000 times the number of threads. The sharp decrease in throughput starting around 180 threads is due to a bottleneck in the output serialization layer [9].

Despite good scaling results on the KNL, the absolute event throughput is not impressive. Table 1 summarizes and compares the measured event throughput for a single worker thread and for a fully-loaded device. The maximal throughput achieved on the KNL with the $Z \to \tau\tau$ sample is only slightly higher than the 16-core Ivy Bridge. A fairer comparison would be a 32-core Haswell processor, which should have substantially higher throughput. The single-thread performance on KNL is observed to be about 6-7 times slower than the Xeon. While some slowdown is expected due to the reduced clock-rate and sophistication of the Airmont cores, this large difference warrants further investigation.

To further understand the performance characteristics on KNL, Intel VTune Amplifier was used to collect and summarize various metrics based on hardware counters. Table 2 shows some of the interesting metrics reported by VTune when comparing G4AtlasMT on a Haswell to the KNL. The clocks-per-instruction rate on Haswell is fairly reasonable, but on KNL an average of three clock cycles are needed to execute every instruction. In addition, VTune reports that the application is highly front-end bound, meaning that the processors are frequently unable to load instructions fast enough to fill the execution pipeline. Finally, we see that the rate of instruction cache misses is nearly 1 on KNL. Such results can be due to poor code layout and large code size.

Table 1. Throughput summary table for an Ivy Bridge Xeon and a KNL Xeon Phi. Results are shown for $Z \to \tau\tau$ and single-muon samples and are split for the case of a single worker thread and a fully-loaded device (or best performing configuration). Ratios of the Xeon Phi to Xeon throughput are shown in the KNL speedup column.

Sample	Threads	Throughput [events/s]		KNL speedup
		Ivy Bridge	KNL	
$Z \to \tau\tau$	single	0.00257	0.000345	0.134
	full	0.0421	0.0445	1.06
Single μ	single	1.38	0.239	0.173
	full	24.6	23.2	0.943

Table 2. Profiling metrics obtained with VTune Amplifier. A single worker thread was used to process a $Z \to \mu\mu$ sample.

Architecture	CPI rate	Front-end bound	ICache misses	Bad speculation	Back-end bound
KNL	3.0	60.2%	0.96	2.4%	18.6%
Haswell	0.9	31.5%	0.09	11.7%	27.6%

5. Conclusion

It has been shown that multi-threaded ATLAS simulation can run on Knights Landing Xeon Phi machines. Good scaling is observed in typical production samples in terms of event throughput and in memory consumption. Multi-threading allows for substantial decreases in the memory footprint of jobs relative to multi-process jobs.

More work is needed to understand and improve the performance on KNL in order to use this architecture effectively. The current performance achieved is comparable to a 16-core Ivy Bridge Xeon, which falls short of the full potential of KNL processors. Since the profiling studies thus far have pointed to issues with large code size and poor code layout, steps should be taken to try and mitigate these problems. Some things to try include pruning unused or unnecessary pieces of code out of the shared libraries, improving code inlining, using statically linked libraries for problematic parts of the builds (e.g. Geant4), and using profiler guided optimization to improve the binaries.

References

[1] ATLAS Collaboration, 2008 "The ATLAS Experiment at the CERN Large Hadron Collider," JINST **3**, S08003. doi:10.1088/1748-0221/3/08/S08003

[2] Calafiura P, Lavrijsen W, Leggett C, Marino M, Quarrie D 2004 "The athena control framework in production, new developments and lessons learned" *Interlaken, Computing in high energy physics and nuclear physics* 456-458

[3] Calafiura P, Lampl W, Leggett C, Malon D, Stewart G A, Wynne B, 2015 "Development of a Next Generation Concurrent Framework for the ATLAS Experiment," J. Phys. Conf. Ser. **664**, no. 7, 072031. doi:10.1088/1742-6596/664/7/072031

[4] Stewart G A *et al.*, 2016 "Multi-threaded software framework development for the ATLAS experiment," J. Phys. Conf. Ser. **762**, no. 1, 012024. doi:10.1088/1742-6596/762/1/012024

[5] Clemencic M, Hegner B, Mato P, Piparo D, 2014 "Introducing concurrency in the Gaudi data processing framework," J. Phys. Conf. Ser. **513**, no. 2, 022013. doi:10.1088/1742-6596/513/2/022013

[6] Aad G *et al.*, 2010 "The ATLAS Simulation Infrastructure," Eur. Phys. J. C **70** 823. doi:10.1140/epjc/s10052-010-1429-9

[7] Agostinelli S *et al.*, 2003 "Geant4–a simulation toolkit," Nucl. Instrum. Meth. A **506** 250-303. doi:10.1016/S0168-9002(03)01368-8

[8] Calafiura P, Leggett C, Seuster R, Tsulaia V, Gemmeren P V, 2015 "Running ATLAS workloads within massively parallel distributed applications using Athena Multi-Process framework (AthenaMP)" J. Phys. Conf. Ser. **664** no. 7, 072050. doi:10.1088/1742-6596/664/7/072050

[9] Farrell S, Dotti A, Calafiura P, Leggett C, Tsulaia V, 2016 "Multi-threaded ATLAS Simulation on Intel Knights Landing Processors" ATL-SOFT-SLIDE-2016-739. https://cds.cern.ch/record/2220833