ATLAS software stack on ARM64

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 072001 doi:10.1088/1742-6596/898/7/072001

ATLAS software stack on ARM64

Joshua Wyatt Smith[a], Graeme A Stewart[b], Rolf Seuster[c] and Arnulf Quadt[a] on behalf of the ATLAS Collaboration

[a]II. Physikalisches Institut, Georg-August Universität, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany
[b]School of Physics and Astronomy, University of Glasgow, University Avenue, Glasgow G12 8QQ, Scotland
[c]University of Victoria, PO Box 1700 STN CSC, Victoria BC V8W 2Y2, Canada

E-mail: joshua.wyatt.smith@cern.ch

Abstract. This paper reports on the port of the ATLAS software stack onto new prototype ARM64 servers. This included building the "external" packages that the ATLAS software relies on. Patches were needed to introduce this new architecture into the build as well as patches that correct for platform specific code that caused failures on non-x86 architectures. These patches were applied such that porting to further platforms will need no or only very little adjustments. A few additional modifications were needed to account for the different operating system, Ubuntu instead of Scientific Linux 6 / CentOS7. Selected results from the validation of the physics outputs on these ARM 64-bit servers will be shown. CPU, memory and IO intensive benchmarks using ATLAS specific environment and infrastructure have been performed, with a particular emphasis on the performance vs. energy consumption.

1. Introduction

The ATLAS experiment [1] is exploring new hardware and software platforms that, in the future, may be more suited to its bulk production workloads. An example is simulation: a CPU intensive workload that would profit drastically if it was more "portable" and therefore usable on a wider variety of platforms.

One such alternative hardware platform is the ARM (Advanced RISC (Reduced Instruction Set Computing) Machine) architecture, which is designed to be extremely power efficient and is found in most smartphones and tablets. It is an architecture where less instructions are used on the CPU, thereby reducing the number of transistors required that then reduces overall power consumption.

2. Hardware

CERN openlab recently installed a small cluster of ARM 64-bit (Aarch64) evaluation prototype servers (Aarch64_Proto). Each server is based on a single-socket ARM 64-bit system on a chip, with 32 Cortex-A57 cores. In total, each server has 128 GB RAM connected with four fast memory channels. Another type of ARM server is also maintained (HP Moonshot) as well as two types of Intel servers (Intel Atom and Intel Xeon). It is important to note the Intel Atom has been discontinued but still provides interesting results. The features of each server are described in Table 1. ARM is significantly newer to this server market and this is reflected in

the semiconductor fabrication size. Only recently has an ARM server been able to compete with Intel in this respect.

Table 1. The different setups for Intel and Aarch64 servers.

Name	Processor	Cores	RAM	Cache	Fabrication (Release)	OS
HP Moonshot	X-Gene, 2.4 GHz	8 Armv8	64 GiB DDR3 (1600 MHz)	32 KiB L1/core, 256 KiB L2/core pair, 8 MiB L3	40 nm (2014)	Ubuntu 14.04
Aarch64_Proto	-, 2.1 GHz	32 Cortex-A57	128 GiB DDR3 (1866 MHz)	32 KiB L1, 1 MiB L2	16 nm (-)	Ubuntu 14.04
Intel Atom	Intel Atom Processor C2750, 2.4GHz	8	32 GiB DDR3 (1600 MHz)	24 KiB L1d, 32 KiB L1i, 1 MiB L2	22 nm (2013)	Fedora 21
Intel	Intel Xeon CPU E5-4650, 2.70 GHz	32	512 GiB DDR3 (1600 MHz)	32 KiB L1(d)(i)/core, 256 KiB L2/core, 20 MiB L3	32 nm (2012)	Scientific Linux CERN 6

3. AthSimulation

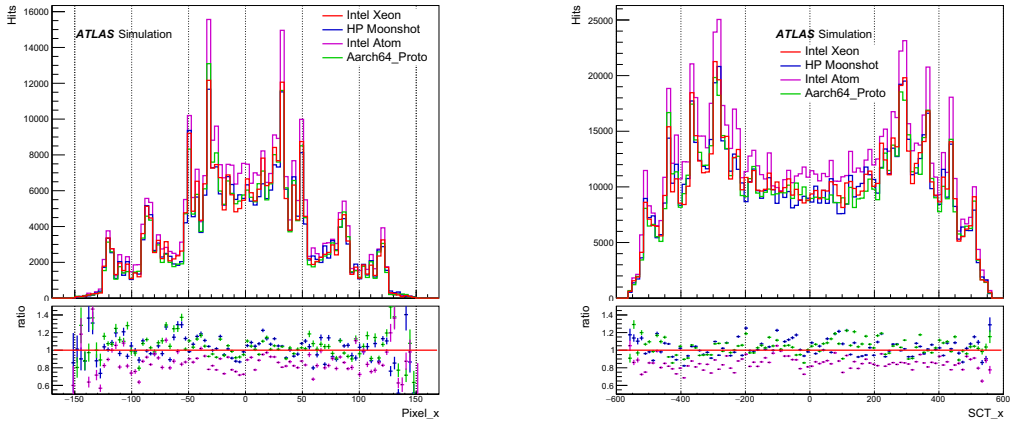
The ATLAS codebase (Athena) is powerful and complex, consisting of around 2400 packages. Due to its size, this makes porting to alternative architectures more difficult. Thus, a project called AthSimulation was chosen which consists of a subset of packages from a full Athena release. AthSimulation is capable of carrying out CPU intensive simulations needed for the experiment. At around 350 packages, the porting process becomes significantly easier and faster. AthSimulation is comprised of the following sub-projects:

- LCG: This projects consists of the external tools that are needed in any ATLAS software. This includes packages such as ROOT, python, various event generators and many others. These packages form the base of the pyramid. Minor changes such as compilation options have to be tweaked for Aarch64.
- AtlasExternals: This project contains ATLAS specific patches and changes. An example is a modified version of Geant4. In this version of the build, AtlasExternals consists of 18 packages. A patch of interest, which propagates through the build, is how the random number generator (AtlasDSFMT) is built. By default *-sse* compiler options are used for x86 (this is an Intel instruction) and cannot be used for Aarch64.
- Gaudi [2]: This is a framework that provides common interfaces for HEP experiments. It is externally maintained and required no changes when compiled on Aarch64.
- AthSimulation: This project is made up of a subset of packages from the ATLAS code. It is a lightweight version of ATLAS Simulation, needing minimal dependencies to be compiled. This project is built with CMake, thus requiring rewrites of the CMake steering files. However, once this was done, no changes or additions were needed for this project to compile on Aarch64.

To ensure reproducibility and bookkeeping, the above was built using the automation server tool, Jenkins. This enabled the ability to rebuild the software stack on multiple servers with little user input. AthSimulation was compiled on both Aarch64 servers, while the Intel servers made use of the equivalent pre-existing builds already available from the ATLAS nightlies.

3.1. Validation

The strategy was to first verify that the results from the different architectures were comparable. Thus, initial validation was carried out. The benchmark was run on each server, with the output HITS files put through another reconstruction phase on a traditional x86 server running Athena. This is an important distinction as it shows that data files created on one architecture can still be opened and manipulated on another - described as heterogenous computing. Due to the nature of simulation, which is a Monte Carlo process, numerical identity is not expected. Some reasons can include and are not limited to random numbers being generated in an architecturally specific way, as well as the way floating point numbers are handled by each compiler. Figure 1 shows the hits in the pixel and SCT detectors for different architectures compared to the Intel Xeon. When compared to the Intel Xeon, the ARM and Intel Atom servers give similar distribution shapes, but around 10-15% less hits on ARM and 15-20% more hits on Intel Atom. Further research needs to be done to understand why this occurs.



(a) (b)

Figure 1. Results showing the hits resulting from 100 $t\bar{t}$ events in the (a) pixel and (b) silicon microstrip (SCT) detectors on ATLAS. The ratio between the three servers and the Intel Xeon is shown in the ratio plots. The Intel Xeon is taken as the "accepted" distribution due to being closest as to what is used in server farms. [3]

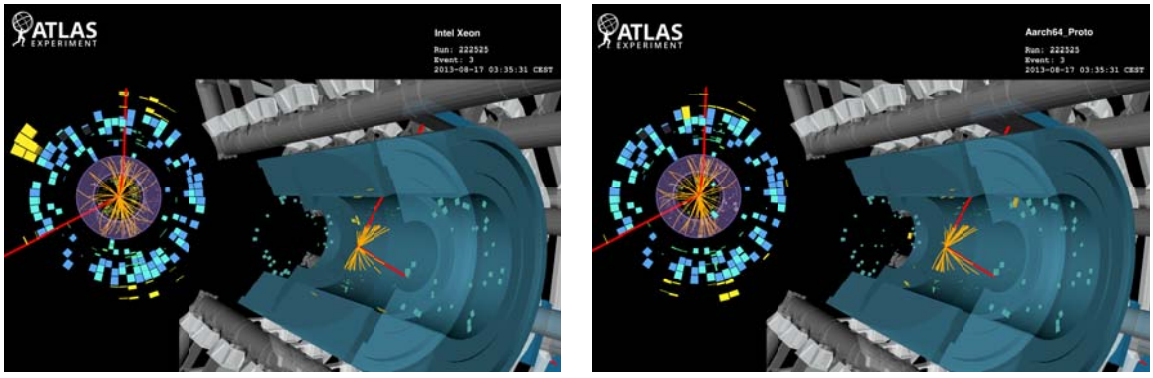
One can also gain insight into individual events by comparing event displays. Figure 2 shows the same $t\bar{t}$ event simulated on Intel Xeon and Aarch64_Proto. One can clearly see this difference in energy depositions due to a different number of hits and also a very minor difference in tracks. However the general topology of the event matches well.

3.2. Power Measurements

Power measurements were taken every 10 seconds. Figure 3 shows the results of the benchmark for a period of 6 hours on a single core. The total times for each server are (in units of hours): *Intel Atom=18.03, HP Moonshot=15.10, Aarch64_Proto=10.46* and *Intel Xeon=6.33*.

It is clear that the Intel Xeon performs the best in terms of time, but it also uses the most energy as shown in the top graph. The middle graph takes this into account and shows Events/kWh. Here, one can see that the Aarch64_Proto is more power conservative for the overall benchmark.

The bottom graphs shows the difference in the total power while running the benchmark and the idling power. This is essentially how much extra energy one CPU uses when under a full



(a) (b)

Figure 2. Same $t\bar{t}$ event simulated on the (a) Intel Xeon and (b) Aarch64_Proto. [3]

load.

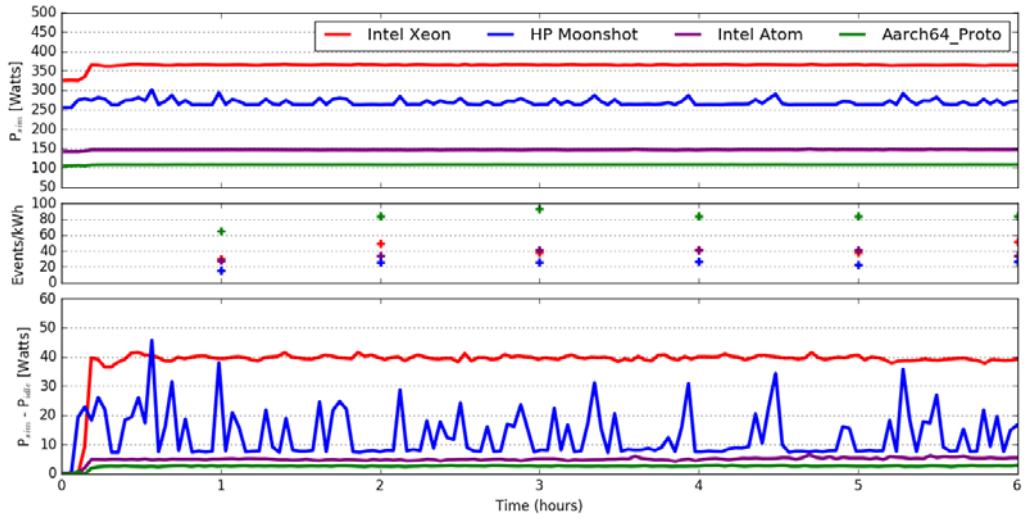


Figure 3. Power measurements for each of the servers. Top: Total power for 6 hours. Middle: Events/kWh calculated for each hour. Bottom: Idling power of each server subtracted from total power when running the simulation.

To take I/O into account, the benchmark was repeated on the Aarch64_Proto and the Intel Xeon server with multiple jobs running. However, in this benchmark, 8 $t\bar{t}$ events are simulated on an increasing number (2,4,8,16,32) of cores.

Figure 4 shows the results. Due to a minor bookkeeping error, test times had to be read from the top graph when calculating kWh. The uncertainty in reading these measurements "by eye" is incorporated into the conservative error bars of the lower graph.

4. Results

Validation tests show that ARM servers register around 10-15% less hits while the Intel Atom shows approximately 15-20% more hits. Further validation over a more statistically significant sample needs to be done, this will help decrease statistical fluctuations seen in Figure 1 as well as help to understand why a different number of hits are registered on different architectures.

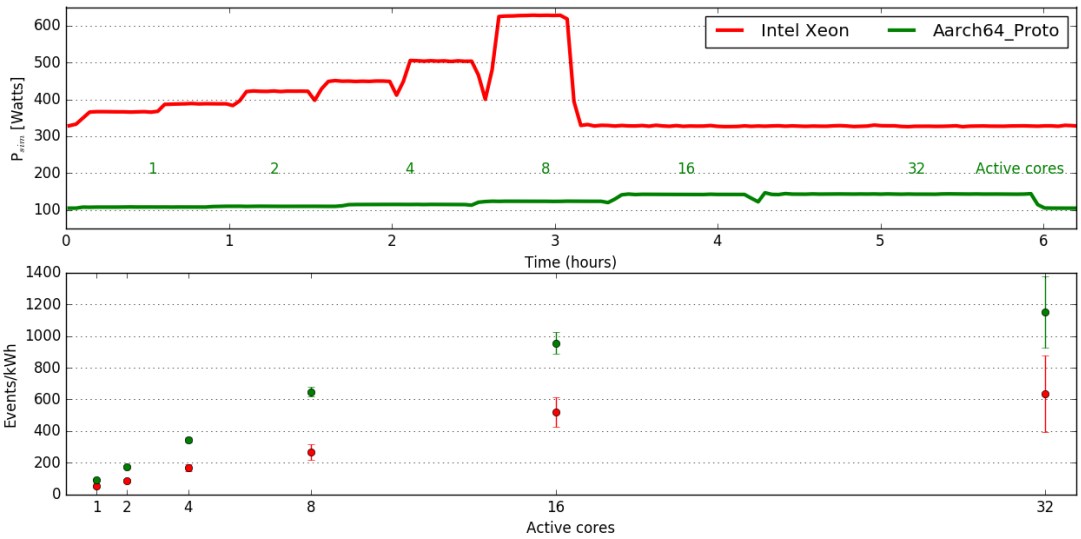


Figure 4. Power measurements for identical benchmark running on increasing number of cores. Top: total time and power for all tests. Bottom: Events/kWh calculated for each test.

Power measurements on a single core show that the Aarch64_Proto performs the best on an event/watt basis. Even though the time to process 100 events is slower, the overall power consumption is much better.

When loading the servers with an increasing amount of jobs, the ARM servers clearly uses significantly less power. At 32 active cores the benchmark test time increases due to developing bottlenecks. Taking total processing time and power consumption into the equation, the ARM server performs between 1.7 and 2.4 times more efficiently than the Intel server while under load.

5. Conclusions

The porting of AthSimulation to Aarch64 as well as initial validation and power measurements are presented. Four servers, two ARM and two Intel, are compared alongside each other in a real-world ATLAS simulation benchmark that was ported to the ARM architecture. The results show that ARM servers have improved dramatically over the past few years. Their 64-bit architecture is now competitive with the traditional Intel machines. In terms of performance per watt, these results show that the Aarch64_Proto performs more effectively than a standard Intel Xeon server. The next step is to integrate the Aarch64 architecture into ATLAS nightly builds. This will allow for quicker and more in-depth performance studies.

Acknowledgments

The authors would like to thank Attila Krasznahorkay for useful discussions and tips when porting the code and to Zachary Marshall for providing the knowledge required to run the AthSimulation benchmark. Also, many thanks to CERN openlab and Techlab, specifically Aritz Brosa Iartza for helping with power measurements and for maintaining the hardware.

References

[1] ATLAS Collaboration 2008 *Journal of Instrumentation* **3** s08003
[2] Barrand G *et al.* 2001 *Comput. Phys. Commun.* **140** 45–55
[3] Smith J W, Stewart G, Seuster R and Quadt A (ATLAS Collaboration) 2016 URL `https://cds.cern.ch/record/2220902`