Multi-threaded software framework development for the ATLAS experiment

2016 J. Phys.: Conf. Ser. 762 012024

(http://iopscience.iop.org/1742-6596/762/1/012024)

Multi-threaded software framework development for the ATLAS experiment

G A Stewart[1], J Baines[2], T Bold[3], P Calafiura[4], A Dotti[5], S A Farrell[4], C Leggett[4], D Malon[6], E Ritsch[7], S Snyder[8], V Tsulaia[4], P Van Gemmeren[6], B M Wynne[9] for the ATLAS Experiment

[1]University of Glasgow, University Avenue, Glasgow G12 8QQ, Scotland
[2]Rutherford Appleton Laboratory, Harwell, Didcot OX11 0QX, United Kingdom
[3]AGH University of Science and Technology, 30-059 Kraków, Poland
[4]Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA 94720, United States
[5]SLAC National Accelerator Laboratory, 2575 Sand Hill Rd, CA 94025, United States
[6]Argonne National Laboratory, 9700 Cass Ave, Lemont, IL 60439, United States
[7]European Organization for Nuclear Research (CERN), CH-1211 Geneva 23, Switzerland
[8]Brookhaven National Laboratory, Upton, New York, United States
[9]University of Edinburgh, Edinburgh EH9 3FD, Scotland

E-mail: graeme.andrew.stewart@cern.ch

Abstract.

ATLAS's current software framework, Gaudi/Athena, has been very successful for the experiment in LHC Runs 1 and 2. However, its single-threaded design has been recognised for some time to be increasingly problematic as CPUs have increased core counts and decreased available memory per core. Even the multi-process version of Athena, AthenaMP, will not scale to the range of architectures we expect to use beyond Run2.

ATLAS examined the requirements on an updated multi-threaded framework and laid out plans for a new framework, including better support for High Level Trigger use cases, in 2014. In this paper we report on our progress in developing the new multi-threaded task parallel extension of Athena, AthenaMT.

Implementing AthenaMT has required many significant code changes. Progress has been made in updating key concepts of the framework, allowing different levels of thread safety in algorithmic code. Substantial advances have also been made in implementing a data flow centric design, as well as on the development of the new 'event views' infrastructure. These event views support partial event processing and are an essential component to support the High Level Trigger's processing of certain regions of interest. A major effort has also been invested to have an early version of AthenaMT that can run simulation on many core architectures, which has augmented the understanding gained from work on earlier ATLAS demonstrators.

1. Introduction

Recent developments in microprocessor technology have underlined the trends that have been identified since the mid-2000s: that while the density of trasnsistors on CPUs has continued to rise more or less exponentially (Moore's Law[1]), clock speeds have stalled (Figure 1).

The fundamental limitations of clock speed, which are thermal in origin, underline that the next challenges for computing are almost entirely based around power efficiency. This drives new generations of CPU designs, which are both low power and have low memory per CPU core.

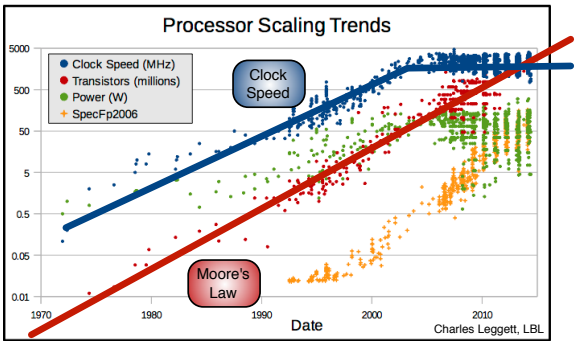

Figure 1. Scaling of key characteristcs of microprocessors since 1970 [2].

In addition to the challenges posed by these constraints, increasing application performance today means taking advantage of advanced CPU features, such as multi-cores and wide vector registers.

In this paper we describe how ATLAS experiment [3] at CERN is adapting to the software challenges from this technological evolution.

2. The Athena Framework

The ATLAS software framework, Athena[4], which is based on the Gaudi framework[5], was designed in the early 2000s. Thus it predates the stall of single core clock speed and is firmly based on a serial processing model. In spite of that, it has successfully processed billions simulated and collider events and played a key role in the discovery of the Higgs Boson[6] and hundreds of other ATLAS observations made during LHC Run1, from 2009 to 2012.

During LHC Run2, 2015 to 2018, increasingly difficult processing conditions have been overcome with the deployment of a multi-process version of Athena, AthenaMP[7]. This uses a multi-process model, where after initialisation multiple AthenaMP worker processes are forked to run the event loop. This allows the sharing of the memory pages allocated for large static structures, such as detector geometry and magnetic field, between worker processes, taking advantage of the Linux kernel's *copy on write* feature. However, there are limitations to how much memory can be saved in this fashion – the change of a single bit will result in an entire memory page being unshared, and C++'s memory model does not allow for fine control over which data objects are assigned to which physical pages. In addition, the further evolution of the LHC machine will increase the complexity of the events that ATLAS will observe and the physics goals of ATLAS necessitate in increase in the data taking rate of the experiment. This increase in event complexity and event rate will be a huge challenge for the experiment, in regards to the memory and CPU requirements of reconstructing events.

3. Future Framework Requirements

Aware of these challenges ATLAS launched a study group, the *Future Framework Requirements Task Force* in 2014. This group's report summarised the main requirements and recommendations of the software processing framework that ATLAS would deploy for Run3, 2021 to 2023:

- The increasingly difficult memory per core environments in the future would require ATLAS to develop a *multi-threaded* framework, in order to maximise memory savings.
- That such a framework should endeavour to provide advanced capabilities, such as event processing in partial regions of the detector, so that the ATLAS online and offline

frameworks could share code and development effort even more than was possible in Runs 1 and 2.

- That a continued development of multi-threading in the Athena framework offered the best chance of success, as, for a running experiment, continuity of operations had to be ensured and disruptive breaks in code continuity would be prohibitively expensive in development and validation effort.
- That in order to be able to exploit the largest range of practical memory and processor options the future framework should foresee exploiting parallelism both at the event level (multiple events processed at once) and the sub-event level. The sub-event parallelism is further divided into inter-algorithm parallelism, directly managed by the framework when algorithms can run independently; and intra-algorithm parallelism, where an algorithm itself exploits opportunities for concurrency.

4. AthenaMT

In the wake of the future frameworks report, the collaboration launched a development effort towards its future framework, which has been christened AthenaMT (*Athena Multi-Threaded*). Here we report on some of the main design discussions and conclusions that have been reached on the framework, as well as on early performance tests.

4.1. Underlying Components

As AthenaMT envisages exploiting parallelism at many levels, including triggered by an algorithm itself, it is important that threading is therefore coordinated across the execution of a program. If each component started its own thread pool the machine's resources would easily become over-contended, affecting memory footprint and throughput. Such a problem is far better suited to a task-based library model than one where thread control is delegated to components. It is also more suited to task queues than loop based parallelism, as much of the available parallelism is from unrelated tasks executing on different events or parts of an event. For the moment the choice has been made to base our threading on the Intel Threaded Building Block library[8].

4.2. Framework Components

Gaudi's original component model and state engine included a number of features that were used heavily in ATLAS software, but prove to be problematic in a multi-threaded environment. One of these was the concept of a *public tool*, which was a single Tool instance shared by multiple algorithms. Such tools in ATLAS were frequently used to store data locally, which would be passed between algorithms, which worked fine when the algorithm execution order was fixed and events were processed sequentially. However, in the multi-threaded case, when algorithms may be processing different events, such a design pattern breaks down. In addition a data dependency between these algorithms has been hidden from the scheduler, which may execute algorithms in the wrong order.

In order to overcome this problem, public tools will be removed from the framework. All tools will be *private* to algorithms and communication between algorithms must go via the event store *service*. It is worth pointing out that services in multi-threaded Gaudi will be event context aware, where as algorithms and tools will be only aware of the event that they are currently processing. Then, developers will be required to migrate public tools to private tools or to services, which ever is more appropriate.

A model of how AthenaMT components process within a single event is shown in Figure 2.

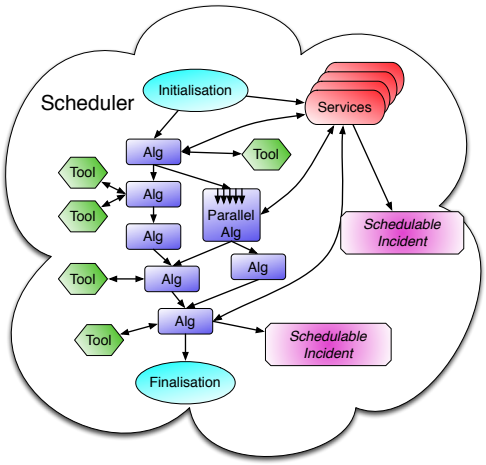

Figure 2. Framework elements involved in the processing of a single event in AthenaMT.

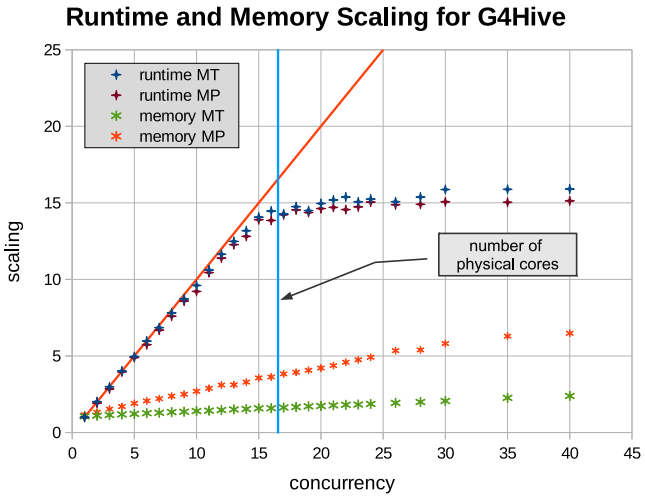

Figure 3. Event throughput and memory scaling comparing AthenaMP (multi-process) and AthenaMT (multi-threaded) approaches for ATLAS simulation.

4.3. Cloneable and Re-entrant Algorithms

For algorithms themselves we now envisage three classes, expressing different levels of thread safety:

Legacy A legacy algorithm can only have one instance and can only process one event at a time; if more than one event wishes to use a legacy algorithm the scheduler is forced to pause them.

Cloneable A cloneable algorithm can have multiple instances, each one of which can be used to process a different event; however, as each clone is a separate instance, memory usage is increased for cloneable algorithms.

Re-entrant A re-entrant algorithm has a single instance, but can be used to process multiple events simultaneously; such algorithms are ideal for both lowering memory consumption and for increasing throughput, but are the most difficult to program as their execution has to be thread safe.

Re-entrant algorithms also put design requirements onto the data handle implementation (data handles declare data dependencies and provide access to event store data), as the algorithm's execute method has to be `const`.

4.4. Event Views

In order to support the requirement of the ATLAS High Level Trigger, which selects events in the online system, it is necessary to process only certain *regions of interest*, which have been flagged by the ATLAS level 1 trigger. To do this, AthenaMT will support *Event Views*, which encapsulate only part of the event data. Development of these views is ongoing, but early prototypes supported the idea of the views themselves being objects within the main event store. Further, the prototype views interface has been made the same as any other ATLAS data proxy, thus algorithms can be made to use a view without any special coding required. Thus the goal, to share algorithmic code between online and offline systems is well on the way to being accomplished. Proper integration with the scheduler is underway.

4.5. Time Varying Data

Time varying data, such as detector conditions, currently relies heavily on Gaudi *incidents*, which trigger the retrieval of data when the algorithm or tool that requires some conditions data finds it is processing an event which is out of the current *interval of validity* (IOV). This blocks a thread while such data is retrieved and does not work well when events being processed may themselves cross IOV boundaries. In AthenaMT, conditions data will be considered as a normal data input to an algorithm, albeit using a different store, supporting multiple IOVs at once. Then conditions data will always be present before an algorithm that needs it is allowed to execute.

5. AtlasG4 Multi-Threaded

Although many pieces of AthenaMT are still under design and development, many simpler test cases and prototypes have been evaluated in order to validate the approach that has been adopted. This started with GaudiHive[9] prototypes, which demonstrated good scaling with limited LHCb reconstruction. An ongoing ATLAS prototype aims at developing a multi-threaded version of ATLAS simulation, suitable for running on Intel's Xeon Phi architecture, where memory is severely constrained. Results from these tests, shown in Figure 3, demonstrate both excellent throughput and memory scaling.

6. Summary and Conclusions

To face the challenges of future computing, with severe power and memory limitations, ATLAS has started to develop a new multi-threaded processing framework, AthenaMT. As well as supporting multi-threading, partial event processing in regions of interest will support HLT use cases better. Prototypes of this framework have already demonstrated excellent throughput and memory scaling. Many other aspects of the framework are being prototyped and implemented now. However, a huge migration effort will be required to port ATLAS's 3 million lines of C++ code to the new framework in time for Run3, and for this a substantial training effort and design review will also be required.

References

[1] Moore G E 1965 *Electronics* **38**
[2] Calafiura P, Lampl W, Leggett C, Malon D, Stewart G and Wynne B 2015 *Journal of Physics: Conference Series* **664** 072031 URL `http://stacks.iop.org/1742-6596/664/i=7/a=072031`
[3] ATLAS Collaboration 2008 *Journal of Instrumentation* **3** S08003 URL `http://stacks.iop.org/1748-0221/3/i=08/a=S08003`
[4] Calafiura P, Lavrijsen W, Leggett C, Marino M and Quarrie D 2005 The athena control framework in production, new developments and lessons learned *Computing in high energy physics and nuclear physics. Proceedings, Conference, CHEP'04, Interlaken, Switzerland, September 27-October 1, 2004* pp 456–458 URL `http://doc.cern.ch/yellowrep/2005/2005-002/p456.pdf`
[5] Barrand G *et al.* 2000 GAUDI - The software architecture and framework for building LHCb data processing applications *Proceedings, 11th International Conference on Computing in High-Energy and Nuclear Physics (CHEP 2000)* pp 92–95 URL `http://lhcb-comp.web.cern.ch/lhcb-comp/General/Publications/longpap-a152.pdf`
[6] Aad G *et al.* (ATLAS) 2012 *Phys. Lett.* **B716** 1–29 (*Preprint* `1207.7214`)
[7] Binet S, Calafiura P, Jha M K, Lavrijsen W, Leggett C, Lesny D, Severini H, Smith D, Snyder S, Tatarkhanov M, Tsulaia V, VanGemmeren P and Washbrook A 2012 *Journal of Physics: Conference Series* **368** 012018 URL `http://stacks.iop.org/1742-6596/368/i=1/a=012018`
[8] Intel Threaded Building Blocks URL `https://www.threadingbuildingblocks.org/`
[9] Clemencic M, Hegner B, Mato P and Piparo D 2014 *Journal of Physics: Conference Series* **513** 052028 URL `http://stacks.iop.org/1742-6596/513/i=5/a=052028`