Journal of Physics: Conference Series **396** (2012) 032115 doi:10.1088/1742-6596/396/3/032115

Multi-core job submission and grid resource scheduling for ATLAS AthenaMP

D. Crooks[1], P. Calafiura[2], R. Harrington[3], M. Jha[4], T. Maeno[5], S. Purdie[3], H. Severini[6], S. Skipsey[1], V. Tsulaia[2], R. Walker[7], A. Washbrook[3] on behalf of the ATLAS collaboration

[1] Kelvin Building, Physics and Astronomy, University of Glasgow, Glasgow, G12 8QQ
[2] Lawrence Berkeley National Lab, 1 Cyclotron Road, Berkeley, CA 94720, USA
[3] SUPA, School of Physics and Astronomy, The University of Edinburgh, James Clerk Maxwell Building, Mayfield Road, Edinburgh, EH9 3JZ, United Kingdom
[4] INFN CNAF and INFN Bologna, V.Le Berti Pichat 6/2, IT-40127 Bologna, Italy
[5] Brookhaven National Laboratory, NY, USA
[6] University of Oklahoma, 660 Parrington Oval, Norman, OK 73019, USA
[7] Ludwig-Maximilians-Universitat Muchen, Munchen, Germany

E-mail: awashbro@ph.ed.ac.uk

Abstract. AthenaMP is the multi-core implementation of the ATLAS software framework and allows the efficient sharing of memory pages between multiple threads of execution. This has now been validated for production and delivers a significant reduction on the overall application memory footprint with negligible CPU overhead. Before AthenaMP can be routinely run on the LHC Computing Grid it must be determined how the computing resources available to ATLAS can best exploit the notable improvements delivered by switching to this multi-process model. A study into the effectiveness and scalability of AthenaMP in a production environment will be presented. Best practices for configuring the main LRMS implementations currently used by grid sites will be identified in the context of multi-core scheduling optimisation.

1. Introduction

ATLAS Monte Carlo simulation, data reprocessing and user analysis jobs run successfully on computing resources at over 100 computing sites worldwide. As the number of CPU cores resident on computing servers (or "worker nodes") at these sites has increased it has been preferable to allocate one job slot per core in order to maximise resources. Although the number of cores per worker node has increased, the ratio of device memory to the number of cores has remained approximately constant. A value of 2 to 3 GB per CPU core is typical at present and is lower for sites taking advantage of hyperthreading technology to double their effective core count.

The memory footprint of ATLAS event reconstruction jobs is expected to exceed 2 GB per core due to the higher amount of pileup events expected in future data taking conditions. Running one ATLAS job per core will be difficult to sustain without memory limits on the worker node being reached and so less jobs will have to be allocated per worker node unless memory pressure can be mitigated. To address this issue, AthenaMP provides a method of

enabling maximum memory sharing between multiple Athena worker processes. Almost 80% memory sharing can be achieved with negligible CPU overhead [1].

Although the advantages of using AthenaMP are clear, effort is now required to define the best approach to configure local resource management systems (LRMS) and batch scheduler software to run ATLAS multi-core jobs in a timely manner. In particular, some care is needed to avoid scheduling contention for jobs requiring different CPU and memory resources to run.

This note will look at how AthenaMP can be incorporated into the ATLAS production system. The implementation and the main features of AthenaMP will be outlined in Section 2. In Section 3 some of the issues faced for sites accepting multi-core jobs from ATLAS will be discussed. Section 4 will cover some of the possible multi-core job scheduling scenarios in further detail by modelling scheduling behaviour in a controlled environment. The current status of multi-core readiness in the production system will be described in Section 5. Recommendations for wider deployment across grid infrastructures and future prospects in this area will be outlined in Section 6.

2. AthenaMP

AthenaMP [2] is an extension of the Athena software framework which provides a method of maximum memory sharing between multiple Athena worker processes whilst retaining event based parallelism. By incorporating all multi-process semantics into the existing Athena/Gaudi framework [3] changes to client code using AthenaMP can be avoided.

2.1. Implementation and workflow

A schematic of the job workflow for AthenaMP is shown in Figure 1. After an initialisation phase an allocated number of worker processes is created prior to the main event loop. A call to the OS `fork()` routine clones the address space of the parent process for each worker process. The Linux kernel Copy On Write mechanism (CoW) is then used for efficient memory allocation with only the differences in memory use between a worker and master process taking up additional overhead. The underlying process creation and communication management to worker processes is provided by a custom multiprocessing C++ library. A bootstrap function handles I/O reinitialisation to allow each process to run in a separate working directory with no communication required between processes. Input events are then allocated via a shared queue and the master process remains idle until all events are processed. Output files generated by worker processes are then merged before the finalisation step is run. More details on the design of AthenaMP are provided in [2].

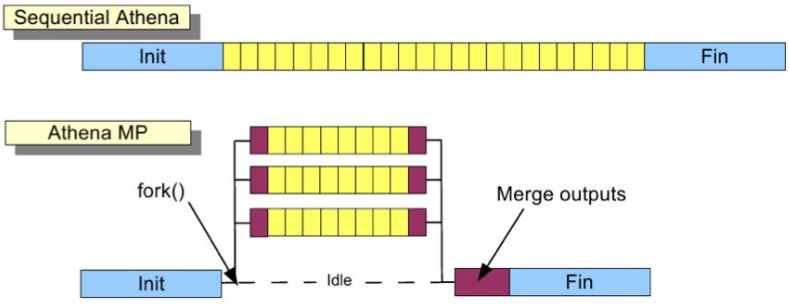

Figure 1. Job flow for serial and multi-process implementations of the ATLAS Athena framework

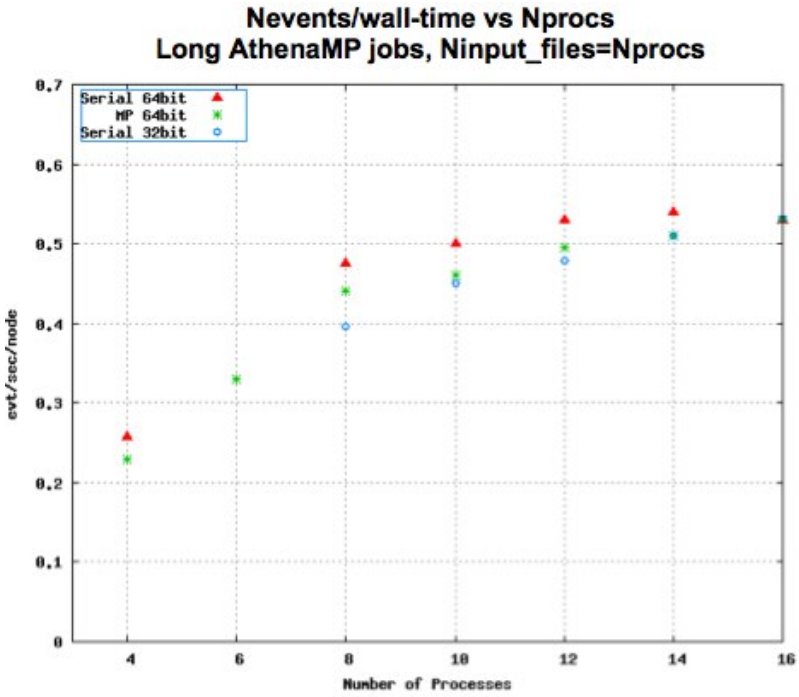

Figure 2. Job event throughput for an increasing number of worker processes. The number of input files is equal to the number of processes.

2.2. Running considerations

The event throughput for AthenaMP does not scale linearly with N processes and it is evident from Amdahl's law [4] that any serialisation steps during execution will significantly affect scaling ability. The main areas of serialisation are in the job initialisation and file merging steps. Any undue latency in these areas needs to be minimised whilst retaining significant memory sharing ability.

In order to increase the proportion of the job in parallel mode it is desirable to increase the length of the event loop. A common approach is to match the number of input files with the number of worker processes. Figure 2 shows the event throughput (i.e. the number of events processed per second per node) of a typical ATLAS event reconstruction job for a increasing number of worker processes. As the number of files (and processes) increases the overall event throughput is comparable to running N jobs in serial mode.

Although a longer event loop (and a higher number of input files) is preferable this will result in an increase in processing time during the finalisation stage due to file merging and output validation. The time taken for merging has been reduced by switching to a faster merge algorithm to concatenate event data and metadata files.

Note that the timing of the worker process `fork()` is crucial in order to enable the maximum amount of memory to be shared. If the `fork()` call is made before data common to worker processes (such as detector conditions data) is allocated into memory then an unnecessary duplication of memory pages will result. Conversely, a late `fork()` call will result in a larger proportion of the overall execution time to be run in serial mode. The optimal approach is found by processing the first input event before the `fork()` is called. A large amount of memory sharing is then enabled with only a small serialisation penalty incurred.

In addition to the running modes described above there are a number of AthenaMP job

options that can affect execution time. Instead of an event queue it is possible to define a fixed allocation of events for each worker process. However, due to large deviations in event processing times this is now generally discarded in favour of the event queue model.

3. ATLAS Production and Analysis Framework

A series of simulation exercises at the ATLAS Tier-0 computing centre validated AthenaMP for production use. The next step in deployment is to ensure that AthenaMP can run successfully and in a timely manner on grid sites providing computing resources to ATLAS.

3.1. ATLAS PanDA System

A key component of the ATLAS distributed computing operations is the ATLAS Production and Distributed Analysis system (PanDA) which provides robust workload management for Monte Carlo simulation, data reprocessing and user analysis. The PanDA server is the main component which provides a task queue managing all job information centrally. Jobs are submitted to the PanDA server, on which a brokerage module prioritises and assigns work to site queues based on a number of factors such as CPU availability and input data locality. One or more pilot factories installed in each regional cloud preschedule pilots directly to grid computing sites using Condor-G [5]. The amount of pilots sent to a site queue is determined by the number of current queued and running pilots on the site queue. Pilots retrieve jobs from the PanDA server to run jobs on worker nodes once CPU slots become available [6]. Each pilot executes a job payload on a worker node, detects zombie processes, reports job status to the PanDA server, and recovers failed jobs. More information on the PanDA system can be found in [7].

3.2. Multicore site configuration

For grid sites wishing to pledge multi-core resources in the ATLAS PanDA system there are two main issues to address:

(i) Should multi-core jobs reserve all the cores available on a worker node (i.e. "wholenode" execution)?

(ii) Should a dedicated set of resources be provided to new multi-core queues?

For (i), wholenode execution allows runtime inspection of the worker node hardware to define the number of cores rather than relying on information from PanDA configuration or external grid information systems. The reservation of a entire worker node also guarantees that memory and CPU resources will be dedicated to AthenaMP execution. Furthermore, a wholenode job can still retain the option to use less than the maximum cores available if further memory limitations were observed.

Although this appears to be the simplest approach there are still some issues to address. As seen in Section 2.2, the number of cores is an important factor in determining optimum job length and so this may need to be included as part of any AthenaMP-based task definition. In addition, an increasing amount of cores per worker node (with 64 core worker nodes already in use) will reduce job scheduling flexibility.

For (ii), grid sites could partition a small but dedicated amount of resources for exclusive multi-core use. This modification also requires minimal configuration changes to the existing PanDA model and jobs sent to multi-core queues can be scheduled and run in exactly the same manner as single core jobs.

An important issue arises when determining how many resources will need to be reserved. It is clear that too low an estimate will result in a large waiting time but an arguably larger problem occurs when allocated resources are too generous. In low usage periods, worker nodes attributed to a dedicated queue will be left unoccupied and cannot be reassigned for single core

jobs, or for any other use without continual intervention by site administrators. This will impact overall site throughput and has led candidate sites to be initially conservative with multi-core resource allocation.

3.3. Resource Allocation

An alternative option to dedicated multi-core resource allocation is to allow the in-built batch system scheduler algorithms to dynamically assign resources. This can provide an efficient automated mechanism to enable the regulation of multi-core use without loss of job throughput. Although this is a more flexible approach there is the potential for scheduling contention which may cause undue latency not evident in the current ATLAS production system.

The scheduling of jobs with different CPU and memory resource requirements is a well defined problem which most scheduler implementations already address successfully. In particular, this scenario is well covered for the simultaneous execution of MPI jobs across multiple woker nodes. However, in the ATLAS production system:

- The job submission rate is dependent on batch system load.
- The job lifetime depends upon external brokerage (which in turn is decided in part by batch system load).
- Grid job queues are not (in general) exclusive for ATLAS use.

These additional factors would need to be taken into consideration when configuring a batch system scheduler to accept both single core and multi-core ATLAS jobs competing for the same computing resources.

4. Multicore Scheduling Simulation

The submission of single core and multi-core jobs of a site accepting workload from the ATLAS PanDA system was modelled in a controlled environment. The components of the testbed and simulation workflow are shown in Figure 3.

4.1. Testbed Configuration

The Torque resource manager and Maui scheduler [8] provided a working batch system solution for the testbed and is a common choice for grid sites available to ATLAS. A feature useful in Torque was the ability for a single worker node to host multiple batch client instances [9]. This enabled the scaling of computing resources to any appropriate test setting. Furthermore, the number of slots per worker node could be set to any value. For the tests described below, a testbed of 100 8-core "virtual" nodes was chosen to capture scheduling conditions expected at a small Tier-2 grid site.

Although tools exist to evaluate scheduler response without the need for job submission [10] it was necessary to include an approximation of job brokerage and pilot factory submission rates to model realistic ATLAS PanDA submission patterns. Job submission and timing, pilot activation, job brokerage and queue performance monitoring were controlled by a suite of steering scripts created for this study.

Jobs submitted to the testbed batch queue were stored on a list populated from input configuration files at the start of the test and from simulated pilot factories throughout the test run. The pilot factories used the same submission algorithm currently used in production. To emulate job brokerage it was not necessary to model the entire PanDA brokerage system. Instead, a simple tally of the number of jobs available for processing was stored. If the tally was non-zero then any pilots queued in the list were switched from an "idle" state to an "active" state by adjusting the job length to a running time representative of an Athena job running in

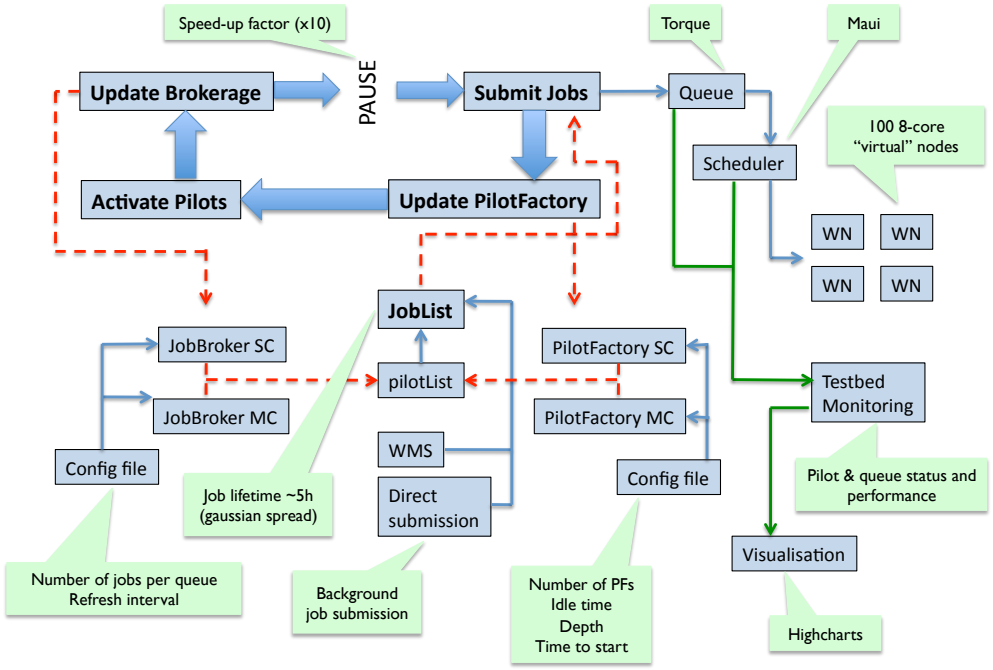

Figure 3. Scheduler simulation steering script components and workflow.

an production environment. The job tally could then be refreshed at regular time intervals or boosted at discrete points during the test run.

Scheduling patterns only become apparent after a number of hours so a configuration option was added to speed up the simulation by a global scaling factor. A $10x$ speedup was used for all the simulations described below.

4.2. Observations

A number of simulations were run to identify common scheduling scenarios that could be observed at grid sites that accept multi-core jobs running on the same resources as single core jobs. In each test, a number of job and queue-based metrics were collected to evaluate relative performance. Job and queue utilisation, average job wait, average pilot wait and a derived brokerage value were able to show how the testbed scheduler responded to different job submission patterns.

The average job wait measured the time a pilot job spent on the batch queue before being executed on a worker node. In isolation this is not enough to fully describe the queue performance primarily due to the variation in the number of pilot jobs submitted by a pilot factory at any given time. To complement this metric the average pilot wait was used to measure the time interval for *any* new pilot job to start running. Both these metrics were averaged over a rolling 12 hour period. Wait times for smaller time periods were also collected to show short term scheduler response patterns.

The brokerage value was a reflection of the CPU availability algorithm used to determine brokerage decisions by the PanDA Server. A higher value denotes an increase in brokerage weight which will consequently attract more jobs to be sent to a queue. This was evaluated by considering the number of running and idle pilots and is averaged over a 3 hour period.

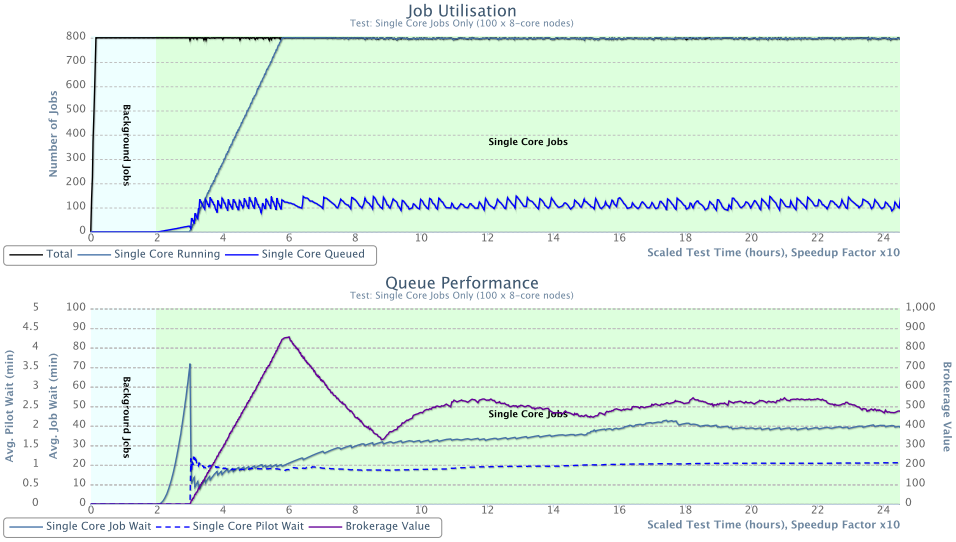

Figure 4. Job utilisation and queue performance for single core pilots.

4.2.1. Scenario 1: Single core pilot jobs The first simulation used one single core pilot factory and is shown in Figure 4. In the initial phase of the test the batch queue was populated with non-pilot jobs (or "background jobs") in order to avoid edge effects with the scheduler due to synchronised job start and completion. Pilot jobs were then introduced to the test which gradually occupied slots on the worker node testbed as the background jobs complete execution. Once the pilot jobs have fully occupied the available slots it is observed that the number of queued jobs is regulated by the pilot factory. This submission pattern continues through the remainder of the test lifetime.

4.2.2. Scenario 2: Single core and multi-core pilot jobs Figure 5 shows a simulation of a single core and multi-core pilot factory submitting jobs to the same batch queue. In this test, the multi-core pilot factory is introduced once single core jobs have fully utilised the testbed. It is observed that the submitted multi-core pilots reside on the queue for a substantial time before a worker node can be allocated. During this period the overall utilisation of the queue drops considerably due to multi-core jobs blocking resources for lower priority single core jobs. In addition, it is seen that the multi-core job utilisation became greater than for single core jobs despite equal pilot job submission rates and scheduler weighting. If this scheduling scenario were to occur the multi-core pilot submission rate would have to be throttled to balance resource allocation.

4.2.3. Scenario 3: Multi-core idle pilot jobs In this simulation (Figure 6) idle pilot jobs are submitted to the batch queue to reflect the scenario where no multi-core jobs have been brokered to the site. An overall queue utilisation drop is observed in the same manner as scenario 2 despite the lack of multi-core workload available.

4.2.4. Scenario 4: Scheduler backfilling Figure 7 shows the effect of scheduling configuration tuning beyond the simple FIFO scheduling model used in previous simulations. In this case, a backfilling algorithm was enabled to allow single core jobs to run before higher priority multi-core jobs when slots were available. This is in contrast to scenarios 2 and 3 where single core jobs are forced to wait for higher priority jobs to be cleared from the queue. Indeed, this approach allows for a higher utilisation of single core jobs as a result. This approach provides overall

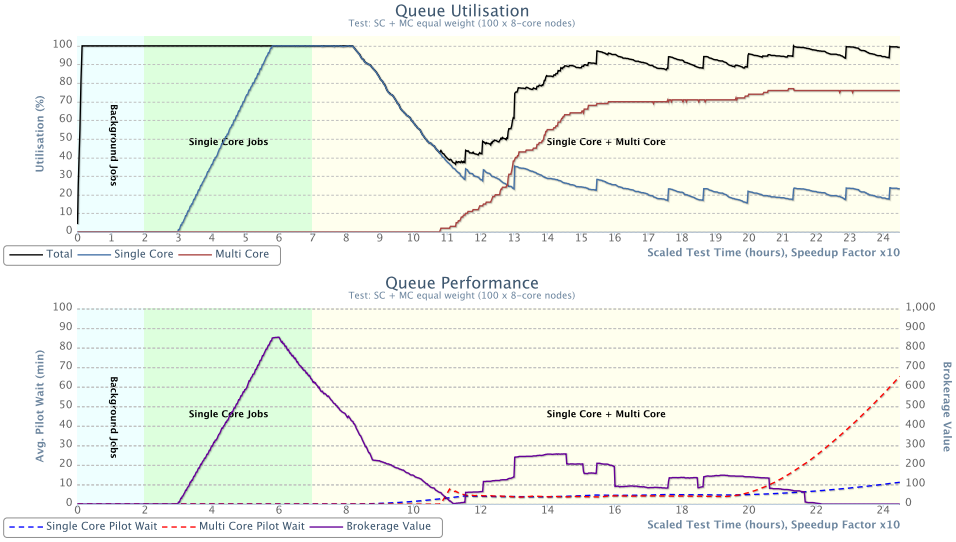

Figure 5. Queue utilisation and queue performance for single core and multi core pilots.

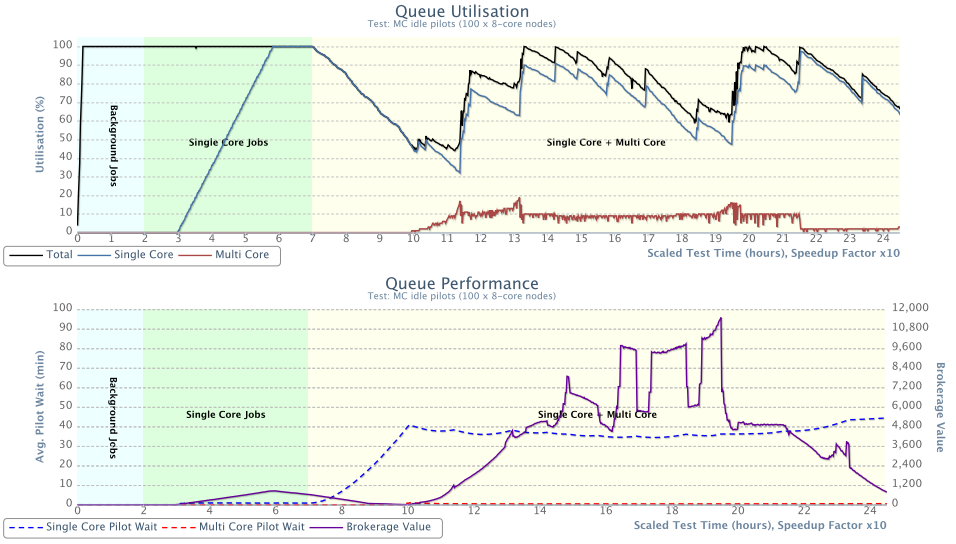

Figure 6. Queue utilisation and queue performance for single core and idle multi core pilots.

higher queue utilisation and further tuning using more detailed backfilling calculation could yield improved results. In particular, the *a priori* knowledge of job lifetime can be used directly by the scheduler to determine whether a single core job can run within the timeframe of a node reserved for multi-core use. At present, job lifetime is not accurately provided through the pilot mechanism or from grid middleware.

5. Current Production Status

Several grid sites have already pledged multi-core resources to allow the testing of AthenaMP jobs in the ATLAS PanDA system. A core count parameter in each new queue definition is used to determine the number of worker processes that can be created by an AthenaMP job running

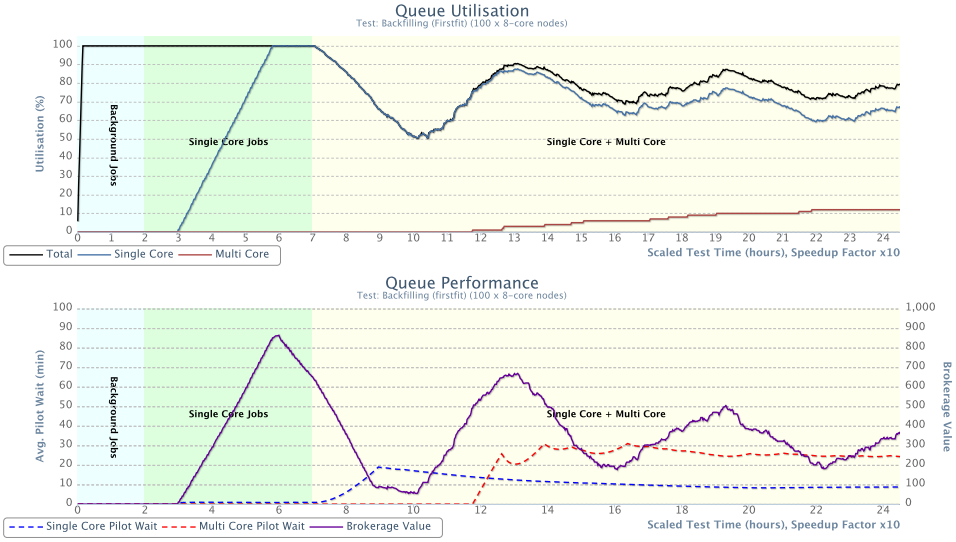

Figure 7. Queue utilisation and queue performance for single core and multi core pilots with scheduler backfilling enabled.

at the site. At this time it is assumed that the core count value refers to the maximum number of cores available on a worker node accessible by the queue. The multi-core queues reside on grid sites with a variety of hardware and different LRMS and scheduler implementations and sites have been free to define queue partitions to match their current workload. The attributes of the multi-core queues available to ATLAS are shown in Table 1.

Site Name	**Cores/Node**	**LRMS**	**Grid Excl.**	**Mcore Queue**	**Pledge**
BNL (US)	8/24	Condor	No	Dedicated	50
ECDF (UK)	8/12	SGE	No	Shared	N/A
Glasgow (UK)	8/12/64	Torque/Maui	Yes	Shared	N/A
INFN-T1 (IT)	4	LSF	Yes	Dedicated	8
Lancaster (UK)	8	Torque/Maui	Yes	Dedicated	8
OSCER (US)	8	LSF	No	Shared	N/A
RAL (UK)	8	Torque/Maui	No	Dedicated	15

Table 1. List of Grid sites used for AthenaMP testing. Sites that allow access to resources through Grid submission only is shown in the *Grid Excl.* column. The *Mcore Queue* column denotes which sites have partitioned resources for exclusive multi-core use. *Pledge* is the amount of worker nodes pledged by the site for exclusive multi-core use.

In most cases only a small amount of worker nodes have been partitioned for exclusive use with the anticipation of a future increase given demand. Conversely, queues at Glasgow and ECDF use the same resources as advertised for single core queues. At ECDF, no additional batch queue has been created. Instead, multi-core jobs submitted to the queue are prepended with appropriate batch system flags to inform the scheduler that a whole worker node is required for execution.

5.1. Functional Testing of Multi-core Queues

A series of functional tests were sent to participating sites to test the availability of multi-core resources on the grid. In each instance, 50 short-lived AthenaMP jobs known to complete successfully were submitted to each multi-core production queue. The difference in time between job submission and the start of execution (the queue response time) is shown in Figure 8.

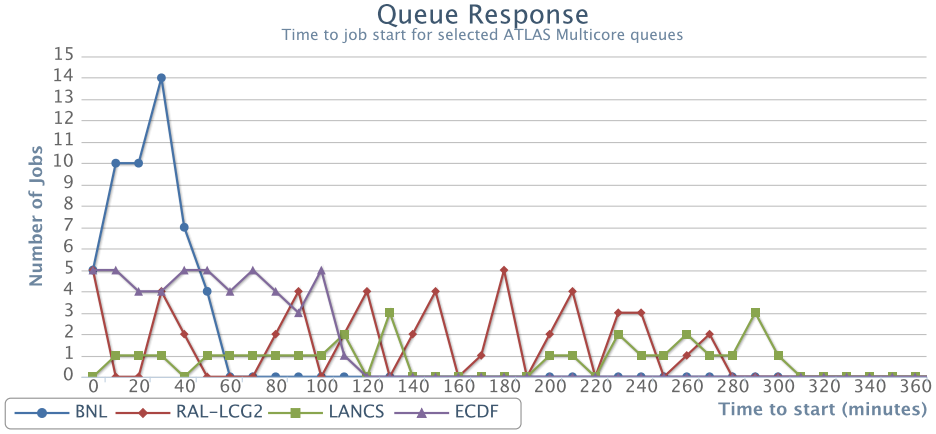

Figure 8. Queue response times for a selection of ATLAS multi-core queues.

For sites that allocated worker nodes for exclusive multi-core use (e.g. BNL) the response profile is directly related to the pledged resources available at the time of testing. For sites using the shared resource approach (e.g. ECDF) the response profile exhibited is dependent on the relative prioritisation of ATLAS grid jobs at the site during the time of the functional test. Further sets of functional tests at shared multi-core sites consequently showed large variations in queue response time.

6. Discussion

The motivation for successfully scheduling and running multi-core jobs alongside single-core jobs is not unique to ATLAS and is relevant for other LHC experiments and for other Virtual Organisations (VOs) using grid resources. Common approaches to multi-core job specification and resource advertising could be captured by modifying the middleware frameworks currently used by grid sites.

The LCG Technical Evolution Group has recommended that additional parameters are available in Job Description Languages (JDLs) in each middleware stack [11]. The number of requested cores, the total memory for the job (or memory per core), wholenode availability, and the minimum and maximum number of cores should be included. Some of these specifications are already available as part of MPI support and could immediately be used for multi-core job specification.

An additional recommendation is the option for sites to advertise multi-core queue status through central grid information systems. This could include whether a site can accept wholenode jobs and the maximum number of cores per job supported. Resource information publishing could also be extended to include dynamic information indicating multi-core readiness based on current site workload. For example, an availability metric could indicate if load conditions were favourable for multi-core job submission and advanced queue status and performance metrics (as shown in Figures 4 to 7) could potentially be useful for central job brokerage.

As discussed in Section 4.2.4, an approximation of job lifetime is a useful parameter to provide more efficient job scheduling. At present a realistic evaluation of job lifetime is not delivered as part of grid job submission but may have to be considered as grid sites begin to run both single core and multi-core jobs for the ATLAS production system.

References

[1] Multicore in production: advantages and limits of the multi-process approach in the ATLAS experiment J. Phys.: Conf. Ser. 368 012018
[2] Harnessing multi-cores: strategies and implementations in ATLAS J. Phys.: Conf. Ser. 219 042002
[3] Mato P 1998 Gaudi - architecture design document Tech. Rep. LHCb-98-064 Geneva
[4] Amdahl G 1967 Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities AFIPS Conference Proceedings (30) pp. 483-485
[5] The Condor Project. **http://www.cs.wisc.edu/condor**
[6] The ATLAS PanDA Pilot in Operation 2011 J. Phys.: Conf. Ser. 331 062040
[7] Overview of ATLAS PanDA Workload Management 2011 J. Phys.: Conf. Ser. 331 072024
[8] The Torque Resource Manager **http://www.adaptivecomputing.com/products/open-source/torque**
[9] Torque Multi-MOM mode **http://www.clusterresources.com/torquedocs21/1.8multimom.shtml**
[10] The Maui Scheduler **http://www.adaptivecomputing.com/resources/docs/maui/mauiadmin.php**
[11] WLCG Workload Management TEG **https://twiki.cern.ch/twiki/bin/view/LCG/WMTEGMulticore**