Exploiting Opportunistic Resources for ATLAS with ARC CE and the Event Service

D Cameron¹, A Filipčič², W Guan³, V Tsulaia⁴, R Walker⁵ and Torre Wenaus⁶ for the ATLAS Collaboration

¹ Department of Physics, University of Oslo, P.b. 1048 Blindern, N-0316 Oslo, Norway
² Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
³ University of Wisconsin-Madison, Department of Physics, 1150 University Ave, Madison, WI 53706, USA
⁴ Lawrence Berkeley National Lab, 1 Cyclotron Rd, Berkeley, CA 94720, USA
⁵ Ludwig-Maximilians-Universität München, Am Coulombwall 1, DE-85748 Garching, Germany
⁶ Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract. With ever-greater computing needs and fixed budgets, big scientific experiments are turning to opportunistic resources as a means to add much-needed extra computing power. These resources can be very different in design from those that comprise the Grid computing of most experiments, therefore exploiting them requires a change in strategy for the experiment. They may be highly restrictive in what can be run or in connections to the outside world, or tolerate opportunistic usage only on condition that tasks may be terminated without warning. The ARC CE with its non-intrusive architecture is designed to integrate resources such as High Performance Computing (HPC) systems into a computing Grid. The ATLAS experiment developed the Event Service primarily to address the issue of jobs that can be terminated at any point when opportunistic computing capacity is needed by someone else. This paper describes the integration of these two systems in order to exploit opportunistic resources for ATLAS in a restrictive environment. In addition to the technical details, results from deployment of this solution in the SuperMUC HPC in Munich are shown.

1. Introduction

The ATLAS experiment [¹] at CERN’s Large Hadron Collider processes and stores data on the Worldwide LHC Computing Grid (WLCG) [²], which consists mainly of dedicated computing and storage resources at various laboratories and academic institutes. Increasingly, ATLAS has been making use of other kinds of resources which are not dedicated to the experiment, such as High Performance Computing (HPC) centres and Cloud computing. Figure 1 shows the CPU consumption of successful ATLAS jobs, split by type of resource, for the first 10 months of 2016. It is clear that Grid computing still provides the vast majority of computing power, but a significant fraction is now provided by HPC (9.65%) and Cloud computing (6.15%).

There are two main reasons for this: one is that many non-standard resources are provided opportunistically, i.e. ATLAS does not have to pay for a fixed CPU capacity but the provider allows ATLAS to take advantage of them when they are not being used. Examples of this include HPC centres where ATLAS can run in scheduling holes (so-called “backfilling”) and volunteer computing where members of the public run ATLAS tasks on their home PCs (this is included
Figure 1. CPU consumption of successful ATLAS jobs by type of resource for the first 10 months of 2016.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Grid</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Intel</td>
<td>Intel, PowerPC, ARM, custom, etc.</td>
</tr>
<tr>
<td>Operating System</td>
<td>Linux RHEL6 or similar</td>
<td>Mostly customised Linux</td>
</tr>
<tr>
<td>Access control</td>
<td>X509 certificates</td>
<td>ssh to login node, one-time passwords, multi-factor authentication</td>
</tr>
<tr>
<td>Network connectivity</td>
<td>Full and unlimited</td>
<td>Restricted or completely forbidden</td>
</tr>
<tr>
<td>Software installation</td>
<td>CernVM File System [3]</td>
<td>Package installation, CVMFS mirroring to local filesystem</td>
</tr>
<tr>
<td>External access</td>
<td>Grid Computing Element gateways</td>
<td>No user-specific services, only ssh access to edge node</td>
</tr>
<tr>
<td>Resource allocation</td>
<td>Guaranteed pledge</td>
<td>Fixed allocation, sporadic (if opportunistic), jobs may be terminated</td>
</tr>
</tbody>
</table>

Table 1. Comparison of features of Grid sites and typical HPC centres.

in the “cloud” element in figure 1 because virtualisation techniques are used). It should be noted that many WLCG Grid sites which are shared between experiments also provide extra “opportunistic” CPU time to ATLAS when other experiments do not use them. The other reason is that many funding agencies do not want to provide dedicated resources for WLCG but prefer that LHC experiments share common resources with other sciences, which typically use HPC centres.

HPC resources present many challenges to ATLAS workflows. To start with, Grid resources are generally homogenous and their architecture has been driven by the requirements of the WLCG community. Each HPC facility is unique and their design is driven by many diverse applications. Many assumptions that ATLAS makes about its computing resources are not valid on HPC centres and so the ATLAS systems must adapt to the different conditions presented.

Table 1 compares several features of Grid and HPC resources. ATLAS systems and workflows were designed around the Grid environment and so do not fit easily into the HPC environment. This paper examines a combination of solutions to some of the challenges posed by HPC -
specifically the ATLAS Event Service and the ARC Computing Element. Section 2 gives an overview of the ATLAS Event Service and its benefits for opportunistic resources. In Section 3 the ARC Computing Element is presented along with its advantages for generic resources like HPC facilities. The integration of these two components and results from a test setup are presented in Section 4 and conclusions are made in Section 5.

2. ATLAS Event Service

The ATLAS Event Service (AES) is designed to open opportunistically used resources to efficient utilization, and to improve overall efficiency in the utilization of processing power and storage. The AES implements quasi-continuous event streaming through worker nodes, enabling full and efficient CPU exploitation through the lifetime of the job, whether that is 30 minutes, 30 hours, or 30 seconds from now. The AES achieves this by decoupling processing from file-level granularity, streaming events into a job and streaming the outputs away in a quasi-continuous manner, with a tunable event upload frequency typically set to 15 minutes or so. While the job persists, it will elastically continue to consume events and stream away outputs with no need to tailor workload execution time to resource lifetime. When the job terminates for whatever reason, losses are limited to the last few minutes, corresponding to a single event when the task is ATLAS Monte Carlo simulation. The approach offers the efficiency and scheduling flexibility of preemption without the application needing to support or utilize checkpointing.

The Event Service also has benefits for conventional processing on clusters and grids. Processing resources can be reassigned in a quick and agile way, e.g. to transition quickly from single-core to multi-core, avoiding inefficiencies from resource draining, because the AES can vacate a resource at any time with negligible losses. Predicting job duration in order to tailor jobs to a slot lifetime is a subject of much detailed study, often with imprecise results; the AES makes such predictions unnecessary. The AES also does away with couplings between input file size, output file size, and job duration, allowing them to be independently tailored.

Storage is the largest cost component in ATLAS computing, making continuous improvement in storage efficiency an essential complement to expanding processing resources. The available disk storage volume is not going to scale with the processing in coming years, it is too expensive. The AES improves storage efficiency by driving down the need for data replicas through the efficient use of wide area network (WAN) data access, thus decoupling processing from data locality requirements. Data retrieval across the WAN is fully asynchronous to the processing in the AES design, avoiding inefficiencies from WAN latencies. Data access in the AES is designed to be mediated by the Event Streaming Service (ESS) capable of providing additional efficiency measures such as utilizing local cache preferentially over WAN access, and marshaling data sent over the WAN to limit data transferred to what is actually needed by the application. The ESS is a work in progress, not yet part of the deployed Event Service.

Crucial to the AES’s agility in vacating resources with negligible losses is the fine grained streaming of outputs (and bookkeeping metadata) of the resource in near real time. This results in outputs that are many (one for every 15 or so core-minutes) and small (typically a few MB or less). They must be sent to a (generally remote) store that is fast, highly scalable and efficient in handling small objects. Object stores are an ideal technology, and are the basis of AES output management. PanDA (the ATLAS workflow management system) automatically merges outputs into conventional ATLAS data files when AES jobs complete. Promptly streaming away outputs has further benefits in minimizing local storage needs, making outputs quickly available to users, and avoiding outputs being trapped on storage local to the processing when it becomes temporarily inaccessible, a common problem on the Grid.

The Event Service workflow is shown schematically in figure 2. Event Service processing begins with the dispatch of an AES job from PanDA to the pilot placing the job request. Receiving the AES job, the pilot goes into event processing mode, entering an event loop in
Figure 2. ATLAS Event Service architecture.

which new event ranges to process are requested every few minutes. In the target architecture
the event list is passed to a data fetcher that interacts with an Event Streaming Service (ESS) to
acquire the data, from local cache if available, otherwise over the WAN. The fetch is asynchronous
with the execution of the payload, AthenaMP \cite{7} in the present instance.

Parallel workers managed by AthenaMP consume events and produce outputs in an area
monitored by an output stager service. As new outputs appear every few minutes per worker,
the stager uploads them to an object store and informs PanDA of the completion. PanDA uses
event level bookkeeping to keep track of the processing, marking events as finished or failed.
Failed or timed out events are reassigned to other workers. In this way a job advances towards
completion, flexibly consuming whatever resources are available as expressed through pilots
acquiring slots and requesting work. PanDA detects when all events for a job are successfully
processed, and triggers a merge job that merges the small object store resident outputs into
conventional ATLAS files, with whatever granularity is desired for the final output files.

A specialization of the Event Service for HPC facilities, Yoda \cite{8}, was developed to
accommodate the particular features of HPC architectures, most notably the lack of outbound
access from worker nodes. Yoda is a miniaturization of the PanDA event workflow management
to operate within the HPC resource itself, with MPI replacing HTTP communication. Yoda’s
master/client architecture allows tailoring of workloads automatically and dynamically to
whatever scheduling opportunities the resource presents.

3. The Advanced Resource Connector and ARC Control Tower
The Advanced Resource Connector Computing Element (ARC-CE) \cite{9} is software developed by
the NorduGrid collaboration to connect diverse computing resources into a computational Grid.
It is designed to be a lightweight, non-intrusive middleware providing an external interface to
a batch system which does not require any modification or extra software on the worker nodes
themselves. A key design feature of ARC-CE is that it can manage data transfers for the
jobs running on the computing resource. Rather than each job copying data in and out, the
ARC-CE manages data transfer in a coordinated scheduled fashion. This architecture makes
ARC-CE very suitable in a restricted connectivity environment such as an HPC centre where no communication with the outside is allowed on the worker nodes.

The ATLAS experiment uses ARC-CE in many computing centres, usually with an additional component to interface it to PanDA. The ARC Control Tower (aCT) [10] was developed initially for ATLAS to serve NDGF Tier-1 and associated Tier-2 sites. The distributed nature of NDGF Tier-1 for both computing clusters and more notably the distributed storage pools was incompatible with a standard pilot job execution workflow. The pilot jobs usually transfer the input files from close storage to a local disk and push the outputs to the same storage after the payload execution. Remote transfers in case of NDGF would be too expensive and unmanageable if the worker nodes would transfer from a remote storage pool. In addition, some of the NDGF clusters are part of larger shared infrastructure, such as HPC supercomputers, where installation of the grid middleware on the worker nodes is not possible.

To address these constraints, ARC-CE is used to transfer the input and output files remotely while the batch jobs only execute the payload and do not spend the precious time on the worker nodes on transfers. ARC-CE provides an input file cache to minimize the number of remote transfers. To make this work in the ATLAS context, the pilot model needed to be adapted so that a fully defined job is submitted to ARC-CE to prepare the input files in advance of the batch job submission, as illustrated in figure 3.

![Figure 3. ARC Control Tower and ARC-CE.](image)

This architecture is used to separate the job execution part from the file transfers and external communication to PanDA:

- aCT communicates with PanDA and submits predefined payload to ARC-CE
- ARC-CE transfers input and output files and submits to the batch system
- The pilot wrapper on worker nodes only executes the payload without accessing the external network, although outbound connectivity may still be used for CVMFS and conditions database access
- The ATLAS batch job does not use the grid middleware, it can execute on minimal operating system installations

This workflow is optimal for sites with capable shared filesystem which caches the input files. It also fits well HPC centres with restricted connectivity, where the ATLAS software is installed locally on the shared filesystem if CVMFS cannot be configured due to site restrictions.
4. Test Setup and Results

The combination of AES and ARC-CE allows ATLAS a model for exploiting HPC centres or other restricted opportunistic resources where jobs may be preempted without notice. The following workflow explains the steps required:

- Run Event Service jobs and do asynchronous data staging of inputs and outputs with ARC CE
- The job writes each event + metadata to working directory
- Once finished, the names of events are put in a special file
- ARC CE detects the job finished in the batch system, and uploads the events listed in the file
- ARC Control Tower tells Panda which events succeeded
- If preempted, a special ARC CE plugin runs, which:
  - tells ARC CE that the job succeeded
  - finds the events that were produced and puts them in the special file

One thing which quickly became apparent was that uploading one event at a time by a single ARC-CE would not scale. If a site was running AES jobs over 10,000 cores and assuming each event is processed in 5 minutes, the ARC-CE would have to sustain an event upload rate to the remote object store of 30Hz. This led to development of a zip functionality in the pilot so that a group of events could be uploaded to the object store instead of one at a time. The pilot writes each event locally as it is produced and at the end of the job it combines all the events into a single zipped file for ARC-CE to upload to the object store. If the job is preempted then the special plugin takes care of creating the zip file.

Some other changes were required in ARC-CE and aCT to facilitate the integration: ARC-CE added support for data transfers to and from Object Stores using the S3 protocol [11], and aCT added support for getting event ranges and reporting their status to PanDA.

The target of this solution is opportunistic running on HPC centres, and the system used for primary testing and deployment was the SuperMUC HPC facility [12] at the Leibniz Supercomputing Centre in Munich, Germany. The HPC facility consists of two parts deployed in phases:

- Phase 1 (2011-13): 155k cores, 3.2 PFlops
- Phase 2 (2015): 86k cores, 3.6 PFlops

As of the time of writing SuperMUC was number 27 on the Top500 list. ATLAS is allocated fixed periods of time on the system and in addition can run opportunistically when resources are not used by anyone else. When running opportunistically the jobs can be terminated at any time without warning. For testing AES and ARC-CE integration, preemptable jobs were run on phase 1 over a period of 3 days. These jobs were allowed to use up to 300 nodes with 24 cores each (total 7200 cores) concurrently but would be immediately killed if another user required the resource.

Figure 4 shows the completed jobs per hour (left) and average number of cores (slots) used per hour (right) by AES jobs over the period of the test. It can be seen that there are times when all 7200 cores were fully available for ATLAS and there are other times when fewer resources were available. The sudden drops in running cores correspond to the batch system preempting jobs to allow a large parallel job to start. In general each drop in running cores coincides with a spike in completed jobs and it is at this point that the AES becomes most effective, ensuring the events produced by the terminated jobs are not lost.
Table 2. Results from the 3-day test at SuperMUC.

<table>
<thead>
<tr>
<th></th>
<th>Completed jobs</th>
<th>Preempted jobs</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of jobs</td>
<td>3861</td>
<td>1014</td>
<td>4875</td>
</tr>
<tr>
<td>Events (M)</td>
<td>1.93</td>
<td>0.23</td>
<td>2.16</td>
</tr>
<tr>
<td>CPU Hours (k)</td>
<td>196</td>
<td>24</td>
<td>220</td>
</tr>
</tbody>
</table>

Table 2 shows the results of the test. Almost 5000 jobs ran and around 20% were preempted by the system compared to 80% which ran to completion. In terms of events produced, around 10% of the total 2.16 million events were produced by preemtied jobs compared to 90% of events from jobs which completed. In other words 230,000 events were produced from jobs which were preempted and these events were saved instead of being lost. The saved events corresponded to 24,000 CPU hours being used instead of sitting idle or being wasted.

5. Conclusion

The combination of ARC-CE and ATLAS Event Service has been shown to be an effective way of exploiting opportunistic resources in restricted environments. The ARC-CE’s data handling capabilities allow the AES to run its output streaming model effectively without requiring network connectivity. This solution also required very little development of ARC-CE and AES when compared to other solutions for similar environments. The non-intrusive nature of ARC-CE and the Event Service paradigm are particularly suited to HPC centres, but work is on-going to apply the same principles in other areas such as volunteer computing and cloud computing. Future plans include scaling up the deployment on SuperMUC and other HPC centres and adapting the model to the next evolution of the Event Service, the Event Streaming Service.

References


