Nordugrid ARC Datastaging and Cache

Efficiency gains on HPC and cloud resources

Maiken Pedersen[1,*], *Balazs* Konya[2,**], *David* Cameron[1], *Mattias* Ellert[4], *Aleksandr* Konstantinov[1], *Oxana* Smirnova[2], and *Anders* Wäänänen[3]

[1]University of Oslo, P.O. Box 1072 Blindern, 0316 Oslo, Norway
[2]Lund University, Box 117, 221 00 Lund, Sweden
[3]Uppsala University, Box 516, 751 20 UPPSALA
[4]Niels Bohr Institute, University of Copenhagen, Blegdamsvej 17, 2100 Copenhagen

Abstract.
The Worldwide LHC Computing Grid (WLCG) is today comprised of a range of different types of resources such as cloud centers, large and small HPC centers, volunteer computing as well as the traditional grid resources. The Nordic Tier 1 (NT1) is a WLCG computing infrastructure distributed over the Nordic countries. The NT1 deploys the Nordugrid ARC-CE, which is non-intrusive and lightweight, originally developed to cater for HPC centers where no middleware could be installed on the worker nodes. The NT1 runs ARC in the native Nordugrid mode which contrary to the Pilot mode leaves jobs data transfers up to ARC. ARCs data transfer capabilities together with the ARC Cache are the most important features of ARC.

In this article we will describe the datastaging and cache functionality of the ARC-CE set up as an edge service to an HPC or cloud resource, and show the gain in efficiency this model provides compared to a traditional pilot model, especially for sites with remote storage.

1 Introduction

The Nordugrid Advanced Resource Connector (ARC) [1, 2] middleware was originally developed to allow Nordic compute and storage facilities contribute to the large compute needs of the LHC experiments, which for the Nordic countries primarily meant the ATLAS [3] experiment.

The Nordic sites, with their heterogeneous systems, wide-area network inaccessibility from the worker nodes, and separated compute and storage required special middleware which at the time was not available. Therefore ARC was developed to cater for these needs. Today ARC is widespread with roughly 200 WLCG sites deploying ARC.

An ARC-CE can be set up as easily for an HPC or cloud resource as it can for a traditional grid resource since it runs as an edge service to the cluster, and no grid middleware runs on the worker nodes. It can also be set up to run in a variety of different modes. In this

*e-mail: maikenp@uio.no
**e-mail: balazs.konya@hep.lu.se

article we will focus on the true-pilot mode and the native mode. In the Nordic Tier 1, all the contributing sites (Norway, Sweden, Finland, Denmark, and Slovenia) have from the start been HPC resources, but recently Norway moved its grid infrastructure to a local Openstack [4] cloud facility. As part of the migration plan, ARC was first run in the true-pilot mode, and then later in the native mode. This provided a good opportunity to measure the job's CPU efficiency in the two modes, compare them to each other, in addition to comparing the CPU efficiencies to the other NT1 (HPC) sites.

The CPU efficiency is simply defined as

$$\epsilon_{CPU} = \frac{T_{CPU}}{T_{WALL}}$$

The longer time it takes before a job's computation actually starts, due to e.g. downloading of input-data, the longer will the walltime be and the lower will ϵ_{CPU} be, for the same CPU-time.

2 Nordugrid ARC

All ARC sites are served via the ARC Control Tower (aCT) [5]. The aCT fetches payloads (jobs) from PanDA, the ATLAS Production and Distributed Analysis system, including the job's requirements, and from this creates a full job description with all necessary information for the job to run. This includes information about input and output files, and what requirements the job has in terms of e.g. memory and CPU time. On site, ARC picks up the job pushed to it by aCT, translates the job description into the language used by the site's batch system, and submits the job into the batch system. Since the job requirements are known, these can be used in the scheduling of the job in the batch system, an advantage over the traditional WLCG pilot-factory job submission [6], where pilots start running on a worker node before any job requirements are known.

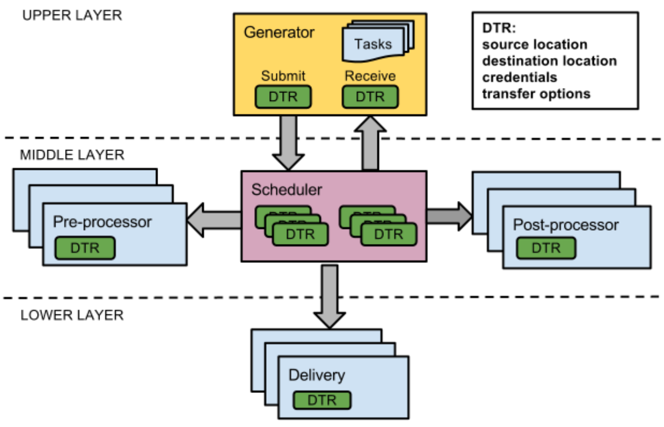


Figure 1: ARC Data Transfer Reloaded (DTR) architecture.

One of the cornerstones of ARC is the integrated job data-management functionality enabling data staging and disk space management in the ARC Data Transfer Reloaded

(DTR) framework and the ARC Cache. DTR is a three-layer architecture consisting of the Generator in the upper layer, the Scheduler and the Pre- and Post-processors in the middle layer and the Delivery in the bottom layer, see Figure 1. The upper and middle layers are components that run on the ARC-CE server, while the Delivery in the bottom layer can run on separate delivery host servers. When a job requiring input files is picked up by ARC's job management component AREX [1], the Generator creates one Data Transfer Request (also called DTR) per file. These DTRs are then sent to the Scheduler for processing. The Pre-processor checks if the file already exists in the ARC Cache, and if it exists, the DTR returns to the Generator with task accomplished and a pointer to the file, no new download is needed. If on the other hand the file is not in the cache, the DTR is sent to the Delivery service for remote download into the cache. The output data is handled in much the same way, creating a DTR for uploading to remote storage, but naturally skips the ARC Cache altogether.

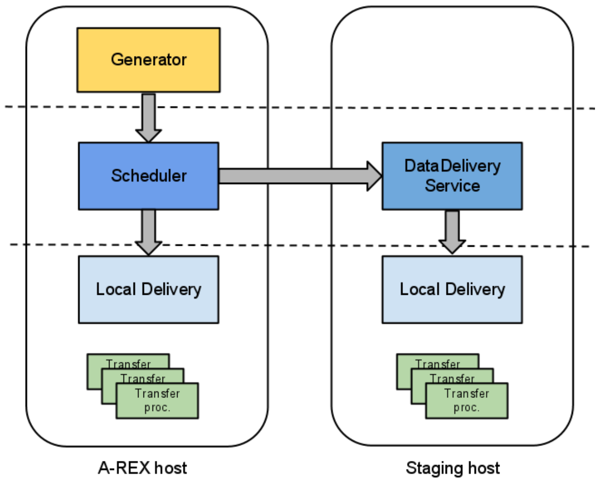


Figure 2: ARC multihost data delivery hosts.

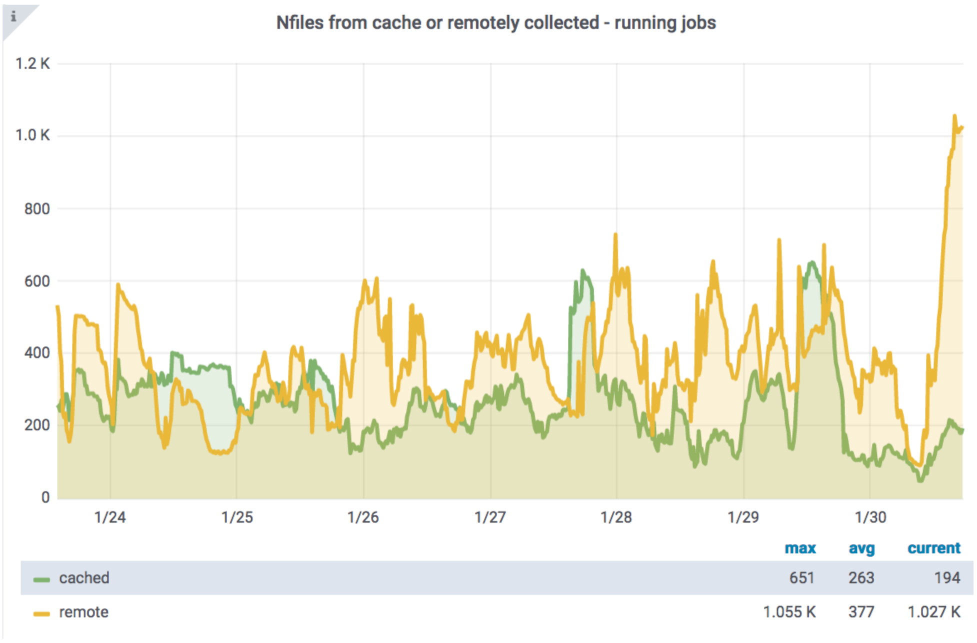
A very convenient feature of the ARC DTR is the possibility to have one or several separate ARC Data Delivery Services (DDS). In this way one avoids overloading the ARC-CE with data transfer requests. Figure 2 shows such a setup with the ARC-CE on the left, and a DDS host on the right. All the logic is handled by the ARC-CE, and only delivery is carried out by the DDS. If the site is configured to, the ARC-CE can also perform data delivery, in addition to the separate DDS host.

Only once all the job's input files are present in the cache, is the job sent to the batch system for computation. The input files are linked to the job's work directory, and as soon as the job gets a free slot in the batch system, the computation can start. DTR together with the ARC Cache results in very efficient job processing, and even more so if the input files in the cache are reused frequently. In Figure 3 the number of files for running jobs that are collected from the ARC Cache or collected remotely over a period of about 1 week for the Oslo site is shown. As can be seen, files from the cache (in green) are accessed very often, and on average about 40% of the time in the period shown. This gives a very important

[1]the Resource-coupled EXecution service

contribution to the overall efficiency of the system.

Another beneficial feature of the ARC Cache is the possibility to allow sharing of the cache to external trusted sites. This can reduce download time if another site nearby can fetch files from your cache rather than from a storage site far away. Also, the DTR can prioritize locations, which means that for example if you know that the network is better at one site compared to another, you can set a higher priority on this site, further increasing data staging efficiency.



	max	avg	current
cached	651	263	194
remote	1.055 K	377	1.027 K

Figure 3: Number of files collected from ARC Cache or from remote storage over the course of 6 days.

3 ARC in true-pilot mode

Many ARC sites in the WLCG use ARC as a simple pilot gateway, running ARC in the so-called ARC true-pilot mode, see Figure 4. On the worker node, the pilot takes care of downloading input files and uploading output files, in addition to sending the job heartbeat information to PanDA.

For sites with local storage as in Figure 4, the ARC true pilot mode works well. This is because PanDA, which orchestrates the data transfer and storage, activates jobs corresponding to the input files present on each site. All the pilot then needs to do is to stage in files from the local storage to the job's work directory. And since everything is local the time spent on this stage is negligible.

However, if the storage element is remote, the ARC true-pilot mode is not optimal. Oslo is part of the distributed Nordic Tier 1 (NT1) site, and therefore data needed for the jobs are stored on any of the sites' storage facilities managed by a central dCache server which is

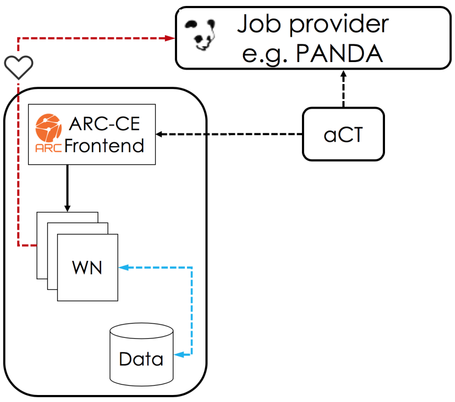


Figure 4: ARC true-pilot mode.

on the LHCOPN network. The storage is "local" to NT1, but not local to the site. Figure 5 shows the Oslo setup with ARC in the true-pilot mode and with remote storage. It is identical to the true-pilot mode in Figure 4 except for the placement of the storage which is remote.

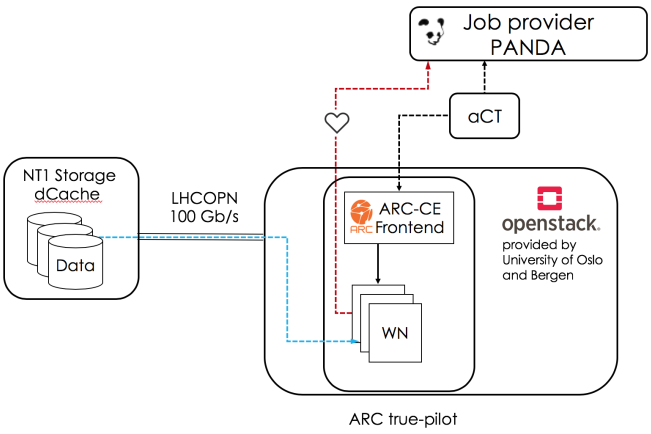


Figure 5: ARC and its interaction with the worker nodes, ARC Control Tower (aCT), PanDA and the Nordic Tier 1 (NT1) storage elements managed by dCache.

Figure 6 shows the CPU utilisation on a typical worker node in the Oslo cluster during the ARC true-pilot mode run. The period when the pilot is fetching input data coincides with the period when the CPU is idle (the green area in the figure), which can be up to several hours.

A job's walltime starts running as soon as the pilot starts executing on the worker node. However, the actual execution of the computation can only start once the pilot has fetched all the input files. In the meantime the worker node's CPU cycles are idle. With the worker nodes idle for such long periods of time as Figure 6 shows, the site's CPU efficiency will

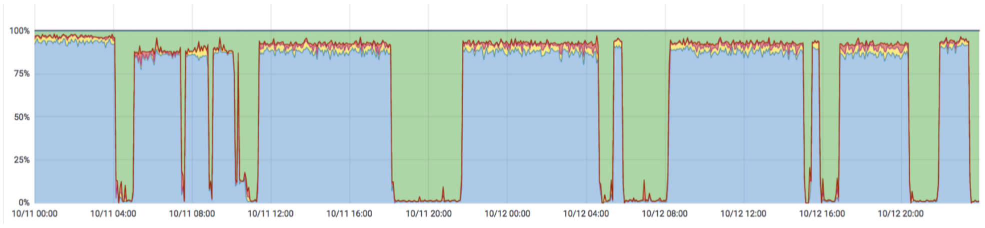


Figure 6: CPU utilisation on a typical worker node with the ARC pilot mode. The green areas show the CPU idle time, and the blue the execution of jobs.

be low. Figure 7 shows the CPU efficiency per job type integrated over one month. As can be seen, the jobtypes with inputfiles (all but MC Event Generation) have very low CPU efficiency, below 50%. This is not an acceptable utilisation of a site's resources.

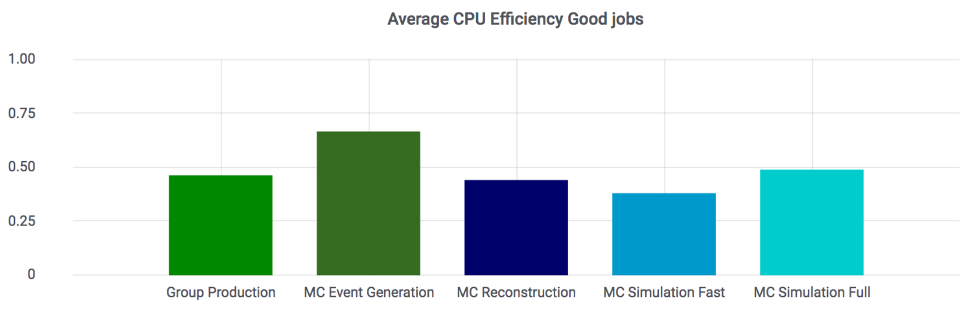


Figure 7: CPU efficiency (ϵ_{CPU}) for jobs in Oslo during the ARC true-pilot mode run.

One way of reducing idle CPU cycles is to backfill the nodes with jobs that are pre-emted in some way or other once higher priority jobs start. One such alternative is to run ATLAS@home [7] jobs in the background on the compute nodes. This has been demonstrated to have very good effect [8], and Oslo has also has had good experience with this [9]. Howevever, the largest component of idle CPU contribution, namely the period when jobs are staging in data, can be taken care of by ARC itself. The next section describes ARC's datastaging capabilities and will show how ARC Datastaging significantly increases the job CPU efficiency.

4 ARC in the native mode

When a job arrives on an ARC site running in the native mode, ARC will first fetch it's data from one of the storage elements in NT1, and only once all the data is fetched will the job be submitted to the underlying batch system. In this way the jobs can start the computation immediately as it arrives on a worker node, as explained in Section 2. The efficiency of this mode is apparent in Figure 8. There is no longer a gap between the (12 different) jobs running in this period and the worker node's time is now used optimally.

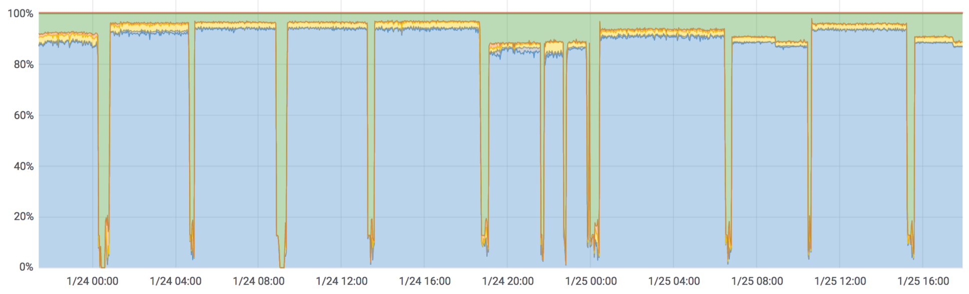


Figure 8: CPU utilisation of a typical worker node running with ARC in native mode.

The good CPU utilisation gives a very high job CPU efficiency as can be seen in Figure 9. Again the job efficiency is shown for jobs requiring a few or many input files (Group Production, MC Reconstruction, MC Simulation), and MC Event Generation which does not have any input files. The CPU efficiency is significantly higher than when running in the ARC true-pilot mode, with efficiencies around 80-90% and above. These numbers are in line with the efficiencies seen at other sites, in particular with the sites used in Figure 11 which will be addressed below.

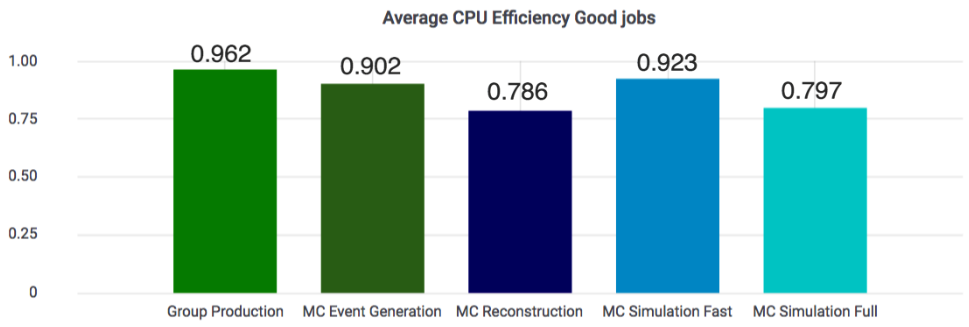


Figure 9: Average CPU efficiency sorted by job type for the UiO Openstack site UIO_CLOUD running in the ARC native mode. The plot shows the effiency averaged over a period of 2 weeks.

It is furthermore interesting to see how the the Oslo Openstack cluster compares to the other (HPC) sites in the Nordics, and as shown in Figure 10 it performs very well.

Finally, the last comparison in Figure 11 is made between NT1 (with remote storage) and 4 true-pilot ARC sites with local storage. These are queues from Canada: CA-SFU-T2_UCORE, Germany: FZK-LCG2 and DESY-HH_UCORE, and Poland: CYFRONET-LCG2_MCORE. The latter is similar in size to Oslo, while the three others are larger sites. When running ARC in native mode, i.e. using the ARC data staging capability and ARC Cache, the performance for the Nordic site is as good or comparable with the pilot sites even though the storage is remote. The native Nordugrid ARC mode is therefore a very good alternative for sites that either have no storage themselves, or for federated sites such

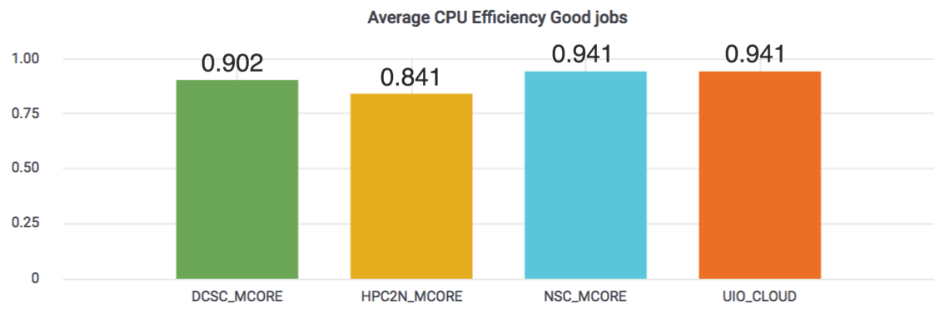


Figure 10: Average CPU efficiency for the Nordic Tier 1 sites, averaged over a period of 2 weeks.

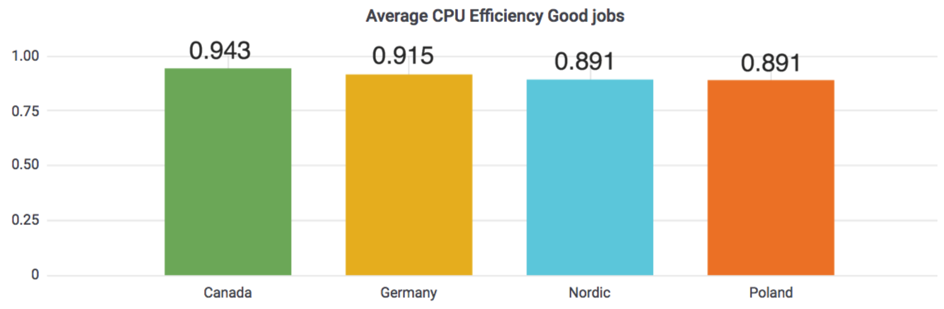


Figure 11: Average CPU efficiency for jobs in 4 different ARC true-pilot mode sites with local storage (Canada: CA-SFU-T2_UCORE, Germany: FZK-LCG2, DESY-HH_UCORE, Poland: CYFRONET-LCG2_MCORE) and with the Nordic Tier 1 running ARC in the native mode.

as the NT1 where the storage is shared among sites.

Although ARC until now mainly has been used by high-energy physics experiments, other research communities with federated storage and compute could also clearly benefit from the ARC data staging and caching capabilities.

5 Conclusion

For sites with remote storage, setting up an ARC-CE with the ARC Data Delivery service and the ARC Cache, instead of running in true-pilot mode, greatly increases the job CPU efficiency and brings it to the same level as for traditional WLCG pilot sites with large local storage. The ARC Cache has a high hit rate for ATLAS workloads, saving repeated download of input files. Finally, it has been shown that ARC can be set up successfully as an edge service on a cloud resource, and that the CPU efficiency is as good as for an HPC site.

References

[1] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. Nielsen, M. Niinimäki, O. Smirnova, A. Wäänänen, Future Generation Computer Systems **23**, 219 (2007)

[2] *ARC online documentation*, accessed: 2020-03-11, `http://www.nordugrid.org/documents/arc6/`

[3] G. Aad et al., Journal of Instrumentation **3**, S08003 (2008)

[4] *Openstack*, accessed: 2020-03-11, `https://www.openstack.org/`

[5] J. Nilsen, D. Cameron, A. Filipčič, Journal of Physics: Conference Series **664**, 062042 (2015)

[6] T. Maeno, K. De, T. Wenaus, P. Nilsson, R. Walker, A. Stradling, V. Fine, M. Potekhin, S. Panitkin, G. Compostella, Journal of Physics: Conference Series **396**, 032071 (2012)

[7] *ATLAS@home web site*, accessed: 2020-03-12, `https://lhcathome.web.cern.ch/projects/atlas`

[8] W. Wu, D. Cameron, Q. Di, *Using ATLAS@Home to exploit extra CPU from busy grid sites* (2018), `1811.12578`

[9] M. Pedersen, B. Konya (2019), 24th International Conference on Computing in High Energy & Nuclear Physics, `https://indi.to/kKhnH`