Monitoring ARC services with GangliARC

D Cameron and D Karpenko
Department of Physics, University of Oslo, P.b. 1048 Blindern, N-0316 Oslo, Norway
E-mail: d.g.cameron@fys.uio.no

Abstract. Monitoring of Grid services is essential to provide a smooth experience for users and provide fast and easy to understand diagnostics for administrators running the services. GangliARC makes use of the widely-used Ganglia monitoring tool to present web-based graphical metrics of the ARC computing element. These include statistics of running and finished jobs, data transfer metrics, as well as showing the availability of the computing element and hardware information such as free disk space left in the ARC cache. Ganglia presents metrics as graphs of the value of the metric over time and shows an easily-digestible summary of how the system is performing, and enables quick and easy diagnosis of common problems. This paper describes how GangliARC works and shows numerous examples of how the generated data can quickly be used by an administrator to investigate problems. It also presents possibilities of combining GangliARC with other commonly-used monitoring tools such as Nagios to easily integrate ARC monitoring into the regular monitoring infrastructure of any site or computing centre.

1. Introduction
The Advanced Resource Connector (ARC) [1] is a Grid middleware originally conceived to connect distributed resources in Nordic countries into NorduGrid. It forms the basis now of national Grid infrastructures in those countries and in many other countries. The middleware consists of a Computing Element to connect computing resources to the Grid, a Storage Element to connect data storage resources, User Tools to enable access to those resources and an Information System to provide information on resources and jobs. This information is used by the user tools to find suitable sites to run jobs and by a web-based Grid Monitor to enable simple browsing of Grid resources and jobs. Information is always a snapshot of the present state and does not present a historical record or trends over time.

For a system administrator hosting an ARC Computing Element (A-REX) on their resources, a simple graphical overview of information through which problems can be instantly recognisable is preferable to sets of command line tools or multiple complex web pages over which the necessary data is spread. It is also helpful to view historical information when diagnosing an issue or to have a record for accounting or reporting purposes. Ganglia [2] is a monitoring framework commonly used for computing clusters which provides these two functions. Daemons on each cluster node collect information frequently on certain metrics such as CPU, network, memory and disk usage, which is aggregated and archived on a central host. A web-based tool can then be used to view how these metrics vary over time for different periods of time, typically from the last hour up to the last year. It is possible to see at a glance for example which nodes are down or have high load or have very little disk space left.
A key feature of Ganglia is the ability to add custom metrics, which can be any property that is able to be represented by a number. The GangliARC framework uses this feature to add ARC-based metrics to Ganglia, so that a system administrator can view ARC properties in the same way as any other property of a node. This paper explains how GangliARC works and presents examples of how useful the data can be and how easily it can be used to diagnose problems.

2. Architecture and Implementation

Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids [2]. It provides a central monitoring portal for a set of computing nodes and consists of three main components: a monitoring daemon (gmond) which runs on each node collecting information on certain metrics, a meta daemon (gmetad) which aggregates the metrics from a set of nodes, and a web front-end which presents the data graphically. Using a tree-like structure this architecture can scale up to thousands of nodes. Many large computing clusters use Ganglia to monitor their resources including nodes running ARC services such as A-REX.

Custom metrics can be added to the feed from gmond on any node using the command line tool `gmetric` or (in later versions of Ganglia) though modules written in C or Python. Adding custom metrics on a node makes graphs for those metrics appear automatically in the web front-end report for that node. GangliARC was created to add custom metrics showing ARC-related information to the feed from the node running A-REX. GangliARC is a daemon running on the A-REX node, which starts a Python script to regularly collect information and call the gmetric command for each metric. A configurable list of metrics are available:

- Last modification time of the A-REX heartbeat,
- free space in each ARC cache and total free ARC cache space,
- free space in each session directory (which contains the jobs’ working directories) and total session directory free space,
- information on the data transfer being performed by A-REX,
  – with ARC’s new data staging framework [3], the number of files in different transfer states, eg transferring, waiting on cache etc.,
  – if old data staging is used, the number of downloader and uploader processes,
- number of active jobs currently being processed by A-REX,
- number of failed jobs among last 100 finished,
- number of jobs in each A-REX job state (ACCEPTING, PREPARING etc.).

It is also possible to configure other options such as the frequency at which information is gathered, log file location and the paths to executables in the case of non-standard installation. Presenting these metrics in Ganglia’s web front-end gives an instant overview of what ARC is doing, and what it has been doing in the past. The next section shows how this information can be used to quickly diagnose common problems.

3. Examples

This section presents some example graphs from GangliARC under certain situations along with interpretations of the data. The examples come from three hosts:

- **pcoslo4.cern.ch** - a dual-core desktop computer used for development and testing which runs dummy test jobs
• ce01.titan.uio.no - an 8-core server at the University of Oslo, Norway, running the ARC CE for a 6000 core cluster of which 1000 cores are reserved for Grid usage for running simulation and analysis jobs for the ATLAS experiment [4] at CERN.

• pikolit.ijs.si - an 8-core server at the Jozef Stefan Institute, Slovenia, running the ARC CE for a 2000 core cluster also mainly running jobs for ATLAS.

Figure 1. Free space in ARC cache over one week.

Figure 1 shows the free space in an ARC cache located at /net/f9sn005/d01/cache over the space of one week (the forward-slash character is not permitted in metric names so it is replaced by an underscore). It can be seen that this cache is configured to have minimum free space of 1TB and when this limit is reached data is deleted until the free space reaches 2TB. The periodic sharp rises show the times when cleaning was triggered and the gradual downslopes show the rate at which data was being written to the cache.

Figure 2. Time since the last A-REX heartbeat, showing that the heartbeat stopped just before 17:00.

The metric for the time since the last A-REX hearbeat is shown in Figure 2. A-REX periodically touches a heartbeat file, which is used by the ARC information system to determine if A-REX is running normally. This metric measures the time since the modification time of that file. Usually it should have a value no greater than one or two minutes, but in this case it starts to steadily increase. This means that the heartbeat has stopped due to A-REX crashing or being stuck and so the administrator must intervene manually to restart the process.

The number of failed jobs out of the last 100 is shown in Figure 3 for two different situations on two different machines. The left graph for pcoslo4 shows a sudden jump to 100 at 15:00, indicating that all jobs suddenly failed. This could mean there is a serious problem with the site such as a file system failure. In the right graph for ce01 the failed jobs also increase, but more gradually and eventually they decrease. In this case it was more likely that a user submitted a large batch of bad jobs which all failed due to an error in their application or due to a trivial reason such as a typo in a job description.

In Figure 4 two sets of two graphs show the number of jobs in state INLRMS (jobs sent by ARC to the batch system and either running or queued there) and the total number of
jobs being processed by A-REX, over the last hour on pcosl04.cern.ch and the last month on ce01.titan.uio.no. For pcosl04 the number in INLRMS remains steady but the number of processing jobs is gradually increasing, showing that the site is receiving jobs faster than it can process them and perhaps the broker the user is using to decide where to submit jobs is not working optimally. The graph for ce01 shows a better situation where the number of jobs in the batch system is roughly the same as the total processed by ARC, implying that A-REX and the job broker are working together well.

**Figure 3.** Number of jobs that failed out of the last 100.

**Figure 4.** Number of jobs in the batch system and total number of active jobs in A-REX.

Information on the files currently being staged for jobs by A-REX is presented in Figure 5 and 6. In Figure 5 the legacy data staging system is used and the graph gives the number of downloader and uploader processes running over one day, for respectively staging in and staging out data for jobs. It can be clearly seen that the amount of input data far outweighs output data — the maximum number of downloaders were running continuously for more than 12 hours, while very few uploaders ran in the same period.

**Figure 5.** Active downloader and uploader processes over one day.
In ARC’s new data staging framework it is possible to split file transfers over several nodes in order to increase the available bandwidth. An example of this case is shown in Figure 6 where new data staging is used with three extra transfer nodes, f9sn00[5-7].ijs.si, and a maximum of total 100 transfers are allowed. The first three graphs give the number of active transfers on each of these nodes, showing the random distribution of the 100 available transfer slots. The bottom-right graph shows the total number of files in the A-REX queue for data staging — that is, files for all jobs that are waiting for input data to be downloaded or output data to be uploaded before proceeding. It shows that jobs arrive in bursts rather than in a regular pattern and towards the end of the graph 1000 files were transferred in about 20 minutes.

4. Integration with Nagios

Nagios [5] is a software framework for alerting a system administrator of problems in his system by conducting regular checks of nodes or services. Unlike the passive monitoring of Ganglia, Nagios is designed to instantly notify problems to the administrator, for example by sending an email or SMS. It also contains a web front-end to give an overview of the whole system, with a colour coding scheme designed to immediately draw attention to problems. Checks that it performs can range from simple ping of a node to complex calls to services. Normally two thresholds for each check are defined by the Nagios administrator - one past which the check is at a warning level, and a second past which the check is critical. Different levels of notification can be defined for the two levels, as well as more complex configurations which depend on knowledge of scheduled downtime or previous notifications.

A combination of Ganglia and Nagios is an ideal way to monitor properties of a node without the need for invasive remote checks. A Nagios check can simply use data from the Ganglia feed and compare those numbers to the configured accepted limits. In this way Nagios checks can be created for ARC properties monitored with GangliARC.

Figure 7 shows a screenshot of Nagios monitoring of two GangliARC metrics: number of failed jobs and time since the A-REX heartbeat as well as regular checks for current load and number of users. The limits for warning and critical are respectively 50 and 80 for failed jobs and 120 and 300 seconds for the A-REX heartbeat. The number of failed jobs is currently 9 so this check is ok, but the A-REX heartbeat has not been seen for 705 seconds, so this check has a critical status. In a real system this could lead to a notification to the administrator, who would hopefully be able to fix the problem before the users of that site noticed that their jobs...
Figure 7. Screenshot of Nagios monitoring, showing A-REX heartbeat status is critical.

were stuck.

5. Conclusion
This paper has presented the GangliARC monitoring tool for administrators of the ARC Computing Element, A-REX. It is a lightweight addition to an industry-standard tool and adds vital information at no extra cost, either in terms of computing resources or learning effort for the administrator. At a glance the administrator can see if all is well with A-REX and what its current job load looks like. It also allows remote detection and diagnosis of problems, and in combination with Nagios can provide alerts to enable the problems to be fixed before the users notice.

References
[1] Ellert M et al. 2007 Future Gener. Comput. Syst. 23 219–240 ISSN 0167-730X
[2] Massie M L, Chun B N and Culler D E 2004 Parallel Computing 30 817 – 840 ISSN 0167-8191
[3] Cameron D, Gholami A, Karpenko D and Konstantinov A 2011 J. Phys.: Conf. Ser. 331 062006
[4] ATLAS Collaboration 2008 Journal of Instrumentation 3
[5] Imamagic E and Dobrenic D 2007 Proceedings of the 2007 workshop on Grid monitoring GMW ’07 (New York, NY, USA: ACM) pp 23–28