Not reviewed, for internal circulation only

Application of rule-based data mining techniques to real time ATLAS Grid job monitoring data

R. Ahrens[1,2], T. Harenberg[1], S. Kalinin[1], P. Mättig[1], M. Sandhoff[1], T. dos Santos[1], F. Volkmer[1] on behalf of the ATLAS Collaboration

[1] Bergische Universität Wuppertal - Gaußstr. 20, 42097 Wuppertal, Germany
[2] Fachhochschule Köln - Betzdorfer Straße 2, 50679 Köln, Germany

E-mail: volkmer@physik.uni-wuppertal.de

Abstract. The Job Execution Monitor (JEM) is a job-centric grid job monitoring software developed at the University of Wuppertal and integrated into the pilot-based PanDA job brokerage system leveraging physics analysis and Monte Carlo event production for the ATLAS experiment on the Worldwide LHC Computing Grid (WLCG). With JEM, job progress and grid worker node health can be supervised in real time by users, site admins and shift personnel. Imminent error conditions can be detected early and countermeasures can be initiated by the Job's owner immedeatly. Grid site admins can access aggregated data of all monitored jobs to infer the site status and to detect job and Grid worker node misbehavior. Shifters can use the same aggregated data to quickly react to site error conditions and broken production tasks. In this work, the application of novel data-centric rule based methods and data-mining techniques to the real time monitoring data is discussed. The usage of such automatic inference techniques on monitoring data to provide job and site health summary information to users and admins is presented. Finally, the provision of a secure real-time control and steering channel to the job as extension of the presented monitoring software is considered and a possible model of such the control method is presented.

ATL-SOFT-PROC-2012-046
22 May 2012

1. Introduction

For the ATLAS experiment roughly 800,000 [1] computing jobs are running daily to perform physics analysis and Monte Carlo production. The details of the job execution are hidden from the user and to some extent from the site admin if no other worker node monitoring tool is installed. But even if available, most machine monitoring tools like Munin [2], Zabbix [3] or Ganglia [4] only monitor general machine health and have no understanding of grid job specific health, let alone allowing to compare the former with the latter. The more job oriented PanDA Monitor can provide information about job meta data and the current job status but can provide inside into job log files only after the job has finished and the output sandbox has been written to a storage element.

JEM closes the gap between real-time site wide summary monitoring and job specific, in-depth monitoring. It does so by monitoring a grid job from the "inside", providing a wide variety of general machine system metrics and specific job health metrics (and allowing to correlate them). To provide live monitoring, all locally gathered information is checked by a sophisticated trigger architecture for relevance and is, upon approval, transmitted to a central server where the data of all jobs is aggregated, stored and analyzed. The data then is provided to interested users,

Not reviewed, for internal circulation only

both in detailed per-job views and in broader analyses featuring correlation and aggregation of data. This allows for advanced monitoring using data mining techniques like outlier detection, metric preemption and metric clustering.

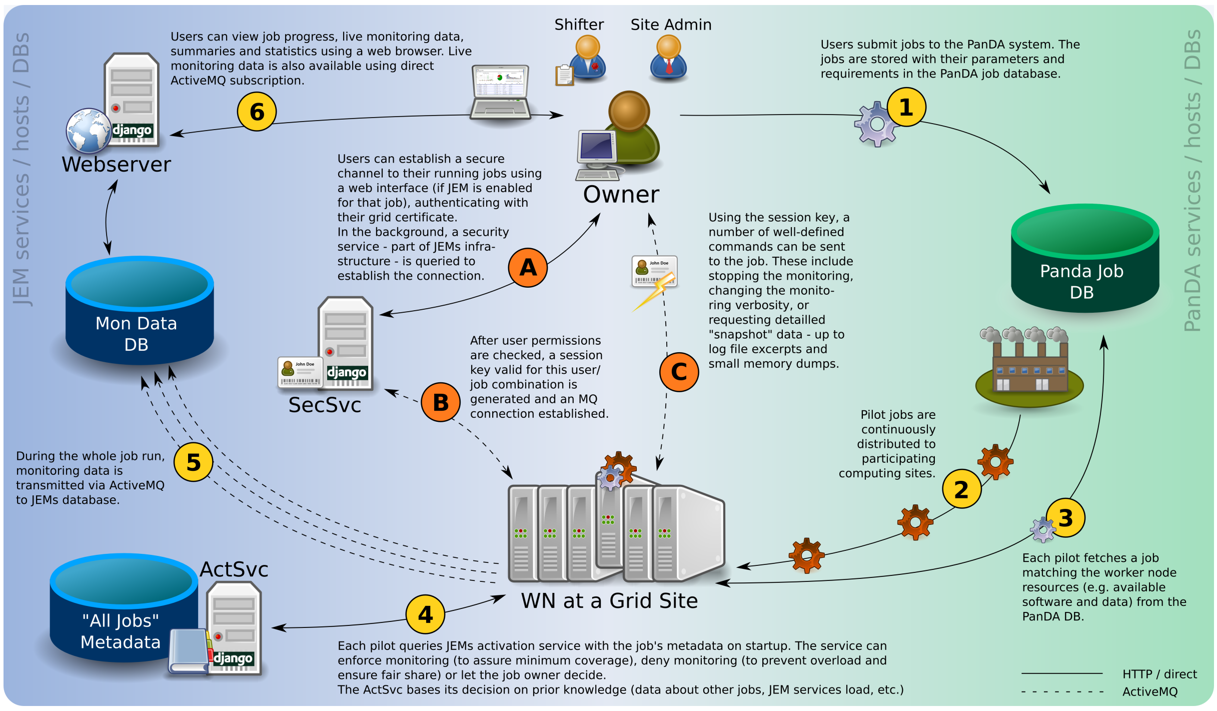
JEM services / hosts / DBs
PanDA services / hosts / DBs
Users can view job progress, live monitoring data, summaries and statistics using a web browser. Live monitoring data is also available using direct ActiveMQ subscription.
Shifter
Site Admin
Users submit jobs to the PanDA system. The jobs are stored with their parameters and requirements in the PanDA job database.
6
1
Webserver
Owner
Users can establish a secure channel to their running jobs using a web interface (if JEM is enabled for that job), authenticating with their grid certificate. In the background, a security service - part of JEMs infra-structure - is queried to establish the connection.
Using the session key, a number of well-defined commands can be sent to the job. These include stopping the monitoring, changing the monito-ring verbosity, or requesting detailled "snapshot" data - up to log file excerpts and small memory dumps.
Panda Job DB
A
Mon Data DB
After user permissions are checked, a session key valid for this user/ job combination is generated and an MQ connection established.
C
SecSvc
B
Pilot jobs are continuously distributed to participating computing sites.
During the whole job run, monitoring data is transmitted via ActiveMQ to JEMs database.
5
2
3
ActSvc
Each pilot fetches a job matching the worker node resources (e.g. available software and data) from the PanDA DB.
"All Jobs" Metadata
4
WN at a Grid Site
Each pilot queries JEMs activation service with the job's metadata on startup. The service can enforce monitoring (to assure minimum coverage), deny monitoring (to prevent overload and ensure fair share) or let the job owner decide.
The ActSvc bases its decision on prior knowledge (data about other jobs, JEM services load, etc.)
HTTP / direct
ActiveMQ
django

Figure 1. JEM hosts and services embedded into the PanDA job submission framework.

2. JEM Architecture

2.1. ATLAS grid job life cycle

In the standard LCG job life cycle a central broker manages the job distribution based on information about job type, available free computing resources at grid sites and job type priorities like user or group fair share. In this architecture jobs are "pushed" from the central instance to the computing sites. ATLAS uses a different system, PanDA [5, 6], where the jobs are "pulled" from central databases to a worker node.

A user defines a job or task and all the necessary metadata then gets transmitted via a small client script to the central PanDA server, which stores the job definition data in a database for further processing. Before transmission, the submission script validates the user's grid certificates and the plausibility of the job metadata. See Figure 1 point 1.

The core pilot factories, mainly installed at CERN, while additional factories are distributed at several ATLAS grid sites, are running to submit so-called pilot jobs, or "pilots", to their corresponding tiers (Figure 1 point 2).

These small pre-jobs get started via the classical gLite/Condor infrastructure. The pilot tests the worker nodes for the availability of various services that are necessary to successfully run a job and then fetches an appropriate job for the current site based on queue, hardware and software conditions (Figure 1 point 3). The pilot also cleans up after the job, manages the input and output sandboxes, stores output files on the local storage element and updates the PanDA server every few minutes about the jobs state.

Not reviewed, for internal circulation only

2.2. *Activation Service*
Once a pilot has fetched all job metadata from the central PanDA server, a relevant subset of the metadata is transferred via JSON[1] [7] encoded HTTP requests to the Activation Service (ActSvc). See Figure 1 point 4. The ActSvc is a web server running at Bergische Universität Wuppertal, which is the first part of the JEM monitoring framework the job sees.

The ActSvc has a set of customizable rules about the planned monitoring coverage for a user or site, how verbose a job should be monitored and the absolute maximum amount of jobs to be monitored by JEM. With the metadata sent by the pilot the ActSvc checks its internal rule base if this particular job should be instrumented with JEM or not.

The upper limit for the number of parallel monitored jobs is determined by the maximum capacity of all subsystems, which must not be overloaded with too much data flow to prevent a system failure and thus the loss of monitoring data.

Every pilot queries the ActSvc for JEM approval before job start. After the job has finished a corresponding notification is sent to the ActSvc as well. This way, with the metadata sent by the pilots, the ActSvc is able to internally keep track of all currently running jobs worldwide and is able to calculate a live monitoring coverage for all sites and users. This coverage is used in the above-mentioned rules to calculate nominal and actual fractions of instrumented jobs.

Once the rule base inside the ActSvc has concluded that the job should be instrumented with JEM the internal database is updated, the relevant numbers are recalculated, and the pilot is instructed to add JEM to the job.

Since the ActSvc has to be fast and reliable most of the system working data is not stored in the database but in a Memcached[2] [8] to allow quick access and manipulation of the data. The database can be optimized for storing the monitoring data as relatively large bulk inserts compared to small inserts for every activation request multiple times a second.

3. Job Monitoring
JEM contains a small wrapping layer application, which is executed as the pilot's payload in place of the actual user job application ("user script") - its analysis- or a Monte Carlo program. The user script is spawned alongside of the JEM data-gathering and transmission module. The wrapping application acts as a watchdog for the data gathering sub process, which can be respawned if errors occur. This architecture ensures that JEM is as non-intrusive to the job as possible.

Periodically, system metrics are checked using different methods (e.g. parsing the proc file system and several other machine information data or by executing scripts to gather data) by the data-gathering module. The exact amount of data, the polling frequency and the types of data being recorded is fully configurable on a per-job basis, with sensible defaults being chosen for ATLAS' needs.

All gathered system metrics and monitoring data are stored in a ring buffer allocated in a shared memory block using a binary chunk format. This provides a high-throughput and non-blocking concurrent writer architecture that allows for sophisticated real-time data selection techniques. A schematic can be seen in Figure 2.

This allows different processes like script monitors, file monitors, process monitors or data transmission modules to access the same data. This is necessary because JEM can monitor subprocesses by attaching itself to the process, and direct access to the memory of the main JEM process is not possible. Data transmission via UNIX sockets, pipes or local network has been identified as being too slow and error-prone.

[1] JavaScript Object Notation

[2] Memcached is a key-value based memory cache, accessible via UNIX sockets or network, without any persistence at all. If the cache is full, old data is overwritten. Data integrity is never guaranteed.

Not reviewed, for internal circulation only

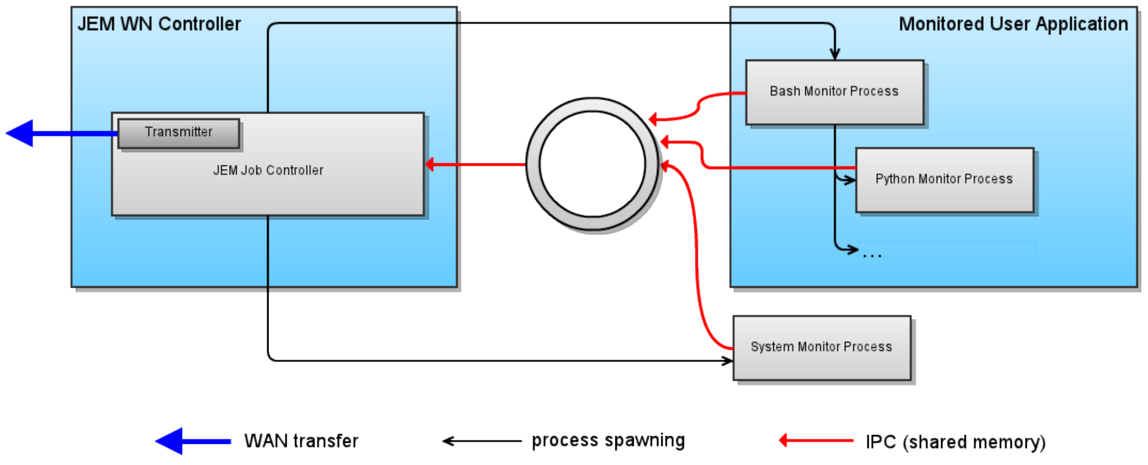

Figure 2. Interprocess communication (IPC) is implemented as a ring buffer in shared memory space. Different data gathering processes can then directly write into the ring buffer while a single control process is tasked with data selection for transmission.

3.1. In Place File Monitoring

All files in the working directory of the job are naturally visible to JEM and can be parsed and analyzed using regular expressions. If a direct monitoring of the running computing binaries is not desired or possible a parsing of log files might yield interesting insights into the job progress or possible problems.

This pattern-based log-file monitoring setup can be defined by the user and is then regularly evaluated by the data gathering process.

3.2. Script Monitoring

JEM can be used to directly monitor the program flow of the user's job if instructed to do so. This is implemented by using script monitors for bash and python scripts, which hook into the interpreter and monitor the program flow. Events can be generated for every evaluated expression, function call, exception and function return.

This can be as verbose as a remote line by line debug execution, which however slows down the job considerably and uses a lot of resources both on the worker node and the receiving end, so a sensible trade-off between performance cost and monitoring data depth must be considered.

3.3. Binary monitoring

If the job binary is properly prepared a so-called CTracer [9] can be attached to the running job's binary and can, depending on the verbosity level, be as verbose as a line by line execution debugger, generating events for every function call or function return, and even create snapshots of application memory.

4. Data transmission and live job monitoring

Before the data is transmitted, a set of configurable triggers check each chunk and either approve or discard it, depending on the configuration. With this mechanism several chunks can be grouped and analyzed and new chunks can then be generated containing the results of the analysis. This way, before sending the data, complex cross-monitor rules can be implemented

Not reviewed, for internal circulation only

(e.g. transmission of detailed memory usage metrics on specific error conditions that are identified by symptomatic log file entries).

The data is sent as raw data chunks, directly from the shared memory, via the standard LCG messaging infrastructure ActiveMQ[3] [10] using the STOMP protocol [11] once the trigger have approved the chunks for transmission.

Every job creates a virtual topic on the ActiveMQ server where all the job's data is published. At any time a user can subscribe to a running job's ActiveMQ channel, in addition to the JEM server data acquisition service, and get a copy of the data being transferred. A live data viewer [12] is available and will present any monitoring data for the job in real time. See Figure 1 point 6. It is also possible to use additional ActiveMQ servers as channels to transmit the data to other consumers, making the overall system scalable to arbitrary complexities.

Using the default configuration, all relevant monitoring data and the JEM logs itself are stored in the job's output sandbox (in addition to the live-transmission) and can be reviewed after a job has finished and the output sandbox files are stored on a storage element. This "post-mortem" analysis allows to identify problems in failed jobs even if JEM's live transmission was disabled, e.g. due to an activation request rejected by the ActSvc.

5. Data Analysis

Once job monitoring data reaches the ActiveMQ server a daemon reads all new data from the messaging server and inserts them into a local object relational database. The web framework Django [13] is used to provide the main web front end, accessing this database. Framework-relevant services like the ActSvc are also implemented using Django.

At the JEM site, several daemons regularly analyze the new data for faulty job behavior by checks for violations of different rules. These violations are then stored in separate data containers to allow easy access to problem data without having to regularly check the whole job for problems. These rules include checking various single metrics for violations of defined thresholds. For example the violation of the job resident set size of more than 2 GB, which is currently defined by the Virtual Organisation atlas [14] as the maximum allowed memory footprint of a job. Also network errors, network bandwidth usage and CPU time allocations are checked. Currently, for example, the following metrics per job are monitored [12]:

- CPU usage times: IO block time, system time, user time
- Memory: RSS[4]

per worker node (WN):

- CPU usage times normalized to one core: IO wait, nice, system, user, steal
- Memory usage: free RAM(physical memory), free swap space, allocated for VMs
- Network usage: transferred and received number of bytes, packets and dropped packets, errors; numbers of listening and connected sockets

per site:

- CPU usage times: minimal IO wait, average IO wait, maximal IO wait
- The list of active worker nodes (as seen by JEM)

In the future, the checking of violation of a combination of different metrics will also be possible. For example, if the iowait amount of time spent by the CPU is high compared to the amount of time spent in user mode, the number of "read"-system calls is high and the used

[3] Apache ActiveMQ is a message broker using an open sourced implementation of JMS 1.1 as part of the J2EE 1.4 specification.

[4] Resident Set Size, the physical RAM footprint of a process.

Not reviewed, for internal circulation only

network bandwidth is low, then the job is inefficiently handling too many small files. Another possible explanation of such a violation would be the usage of a not optimized for reading ROOT file. After the job owner would be notified about this problem, the ROOT tree file could then be optimized.

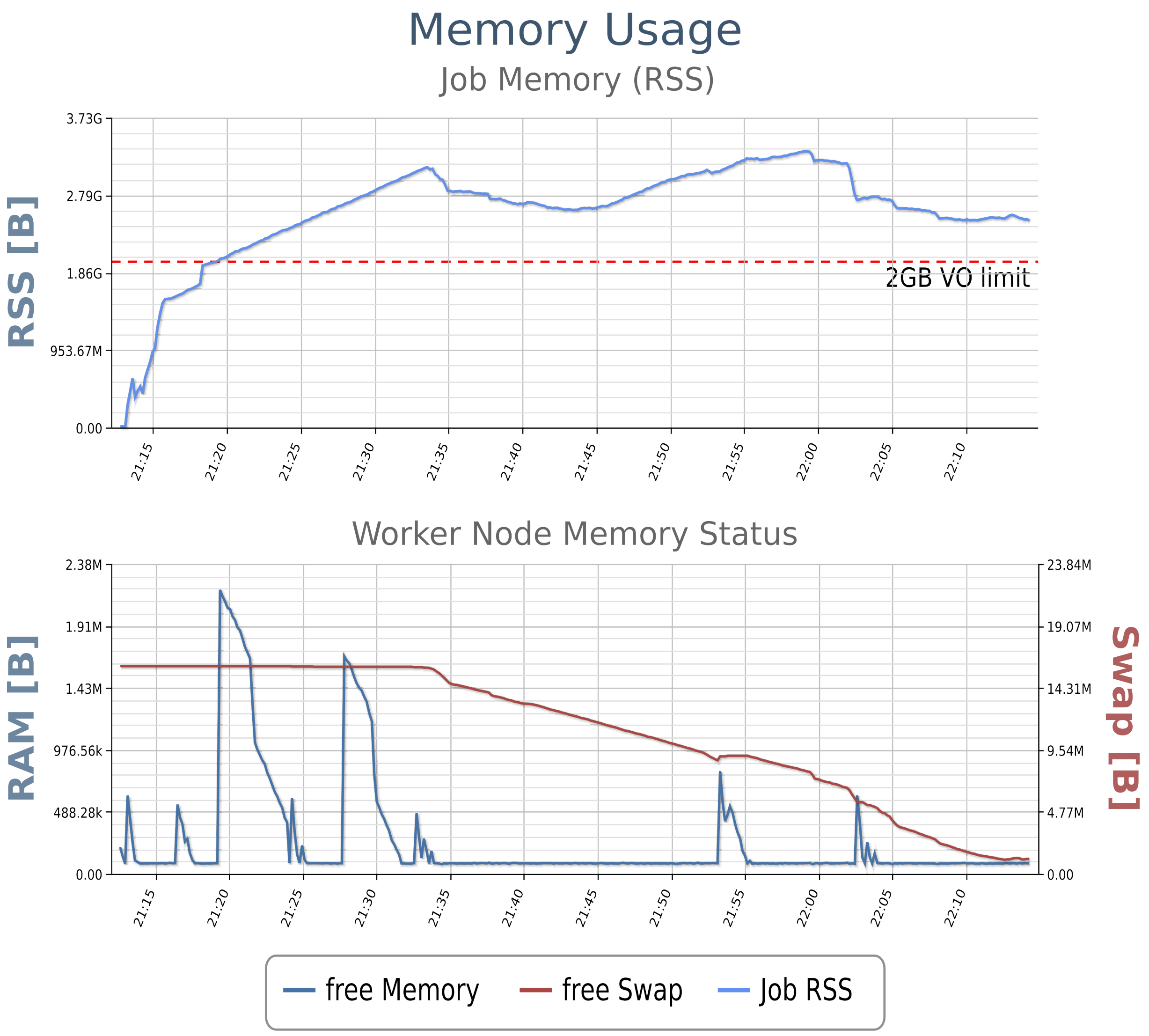

Figure 3. The resident set size of the job increased during its runtime, well over the assigned 2GB, while the free RAM and the free swap of the WN decreased significantly as other jobs of the same user.

Automatic analysis of JEM live data checks all monitored jobs periodically for faulty behavior and can generate simple rule based warnings to single jobs. For example, violations of thresholds and extrapolated trends of single or combined metrics as well as aggregations of many jobs running for the same user or on the same site / worker node can be detected. Outliers can be identified and corresponding actions taken (e.g. notifications to the owner of the job or the site admin). Figure 3 shows the metrics of a typical misbehaving job versus the corresponding WN metric. Such faulty resource usage has the potential of affecting other jobs on the same WN and can be detected in real time using JEM. The distribution of all monitored jobs' average memory consumption is shown in the bottom plot of Figure 3 for one site. Faulty job patterns can be

Not reviewed, for internal circulation only

detected, allowing site admins to take measures without the need to check regularly. Various other metrics also provided by JEM, like CPU consumption, disk IO or network traffic, can be checked in the same way.

6. Future Projects

6.1. The JEM feedback channel

At the moment JEM needs to be configured before the job starts to run on a WN. Because of that, it is not possible for e.g. the job submitter to change the configuration of JEM while it is running. To do this JEM would need to provide a feedback channel to the job submitter over which this feedback channel JEM could be instructed to change its configuration, intensify the monitoring or get a snapshot of the current state of the job. Currently a finished concept for the feedback channel has been drafted and a first working implementation is in testing.

Since it is not always possible to send messages directly to a WN a message broker is needed. The ActiveMQ is such a message broker and enables the establishment of a connection between the user and the JEM instance. Currently STOMP is used as the connection protocol to the ActiveMQ.

To open an authenticated feedback channel to a JEM instance the user needs to connect to the Security Service (SecSvc) this is done in a secure way, e.g. via HTTPS. The task of the SecSvc is to check the identities of the JEM instance and the user and to generate a session key for the communication between the JEM instance and the user. Also the SecSvc needs to be able to access the VOMS[5] [15] database to check the privileges of the user. The SecSvc is a trusted server, which can identify itself via a public key certificate. See Figure 1 point A.

If the SecSvc has received a request it sets a signal for the JEM instance, telling the system that the user is trying to establish a connection to the job. In our concept it is intended that the JEM instance, which is running on the WN periodically checks if a signal had been set which informs JEM to open a feedback channel.[6] To open a feedback channel the JEM instance needs to open a unique topic on the ActiveMQ message broker. As soon as the JEM instance has done this it will send a request for a key generation to the SecSvc. In this request the JEM instance also sends the name of the owner and the site on which it is running. This information is needed to check if the request of the user is valid. If the user request is invalid the process is aborted and the JEM instance disconnects from the message broker. See Figure 1 point B.

If the user's request is valid the SecSvc and the JEM instance generate a secret key via the Station-to-Station [16] protocol, which is then transmitted to the user. The user can then directly communicate with the JEM instance over a channel signed by the secret key. See Figure 1 point C.

The advantage of the SecSvc is that the checking of the identity of the user and the rights of that user is not part of JEM instance, so the WN has only to check the identity of the SecSvc. In addition, the signing (or even encryption) of the feedback commands can be done via Message Authentication Codes (MAC) [17].

Currently, the only purpose of this feedback channel is to change the monitoring activity of JEM itself. The channel is not intended to allow communication with the monitored job (and the corresponding user application), or to give full access to the remote WN, shell access, or similar control. For this reason, a well-defined and limited set of commands is provided to perform actions on the monitoring instance such as "start live monitoring" (in case JEM

[5] The Virtual Organization Membership Service is a system for managing authorization data within multi-institutional collaborations.

[6] This is a necessity since it is not feasible to open one feedback channel for every running JEM-instance, as this would consume large amounts of resources with negligible benefit, because only a very small fraction of jobs will be controlled by users.

was only started in dormant state for this job), "stop live monitoring", "increase monitoring verbosity" and "return an in-depth snapshot of current monitoring data".

Not reviewed, for internal circulation only

6.2. Adaptive Job Monitoring

A method to automatically decide when and where additional monitoring of jobs will be necessary and helpful based on currently monitored metrics is being worked on.

To achieve this JEM needs to be able to determine when to increase or decrease monitoring for a specific entity. This entity can be a user, a job-set, a production task, a site, a computing element or a worker node. A broad coverage of at least one monitored job per all known instances of these entities would be the first step to feed a set of rules with input which then decide when and where monitoring activity should be increased based on irregular or faulty behavior. A complete coverage is however not possible due to the vast amount of parallel running jobs and the limited resources of the JEM system. To feed the JEM ActSvc with additional data where to instrument jobs with JEM, a live monitoring of the current ATLAS wide job brokering system PanDA would also be helpful.

Additionally, a dedicated website at the JEM page will provide an interface for users and site admins to request additional monitoring for their jobs or their site (this can be implemented by updating the activation services rule base according to those requests).

To increase monitoring for a specific entity a new rule with a limited lifetime would need to be dynamically added to the ActSvc and from then on jobs matching the criteria of the entity requesting monitoring can be activated. This will however happen with a delay, especially if the entity is not starting new jobs at the moment for any reason.

We propose a dormant JEM running mode which gets launched by the pilot for all jobs, which are not actively monitored. This dormant JEM would simply establish periodical, lightweight communication with the proposed signal service and only upon a command start gathering and transmitting monitoring data to the JEM servers.

Also a notice should be sent via mail or web interface to a person in charge, either the site admin and/or the current shifter that JEM has possibly detected faulty behavior and the incident needs to be investigated further.

7. Summary

JEM can improve the understanding of different job errors by providing informative and exhaustive information from the "inside" of the running job and the worker node running the job to any interested person. It is easy to use by simply adding a single additional parameter to the job submission script.

The JEM framework can aggregate monitoring data up to higher entities like a worker node or the job set of a specific user to further investigate any problems (see Figure 4). A possible clustering of errors or an interconnection of single faulty jobs leading to a machine failure can thus be detected and visualized.

The JEM framework is able to be easily adapted to different grid infrastructures, not only using the PanDA job brokerage system: Only a few key interfaces like the call to the ActSvc or a different data transmission system would need to be implemented by the infrastructure to start using JEM.

The current development of JEM concentrates on improving the user interfaces on the web site and further analyses on gathered monitoring data to advise a future adaptive monitoring where to intensify the monitoring.

Not reviewed, for internal circulation only

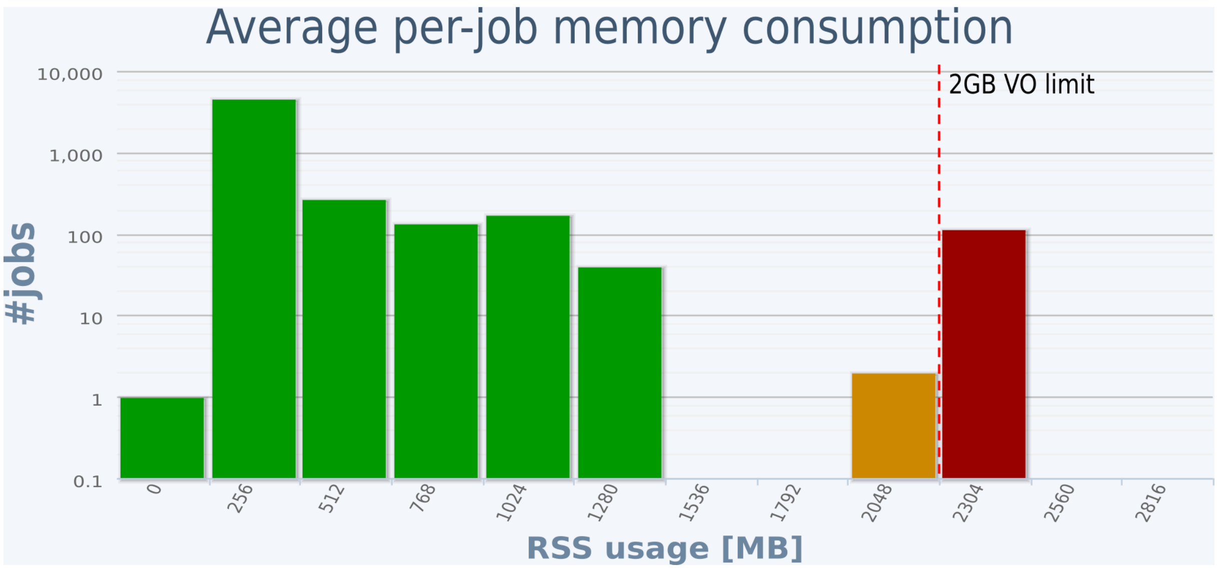

Figure 4. Average job resident set size of all monitored jobs measured running on the Pleiades grid cluster in Wuppertal.

References

[1] ATLAS Dashboard, Daily Job Summary, http://dashb-atlas-job.cern.ch/dashboard/request.py/dailysummary
[2] Munin project page, http://munin-monitoring.org/
[3] Zabbix, The Enterprise-class Monitoring Solution for Everyone, Zabbix project page, http://www.zabbix.com/
[4] Ganglia, Monitoring clusters and Grids since the year 2000, http://ganglia.info/
[5] T. Maeno *et al.*, "Evolution of ATLAS PanDA System", to appear in Proceedings of the Computing in High Energy and Nuclear Physics 2012 International Conference
[6] PanDA: distributed production and distributed analysis system for ATLAS, T Maeno 2008 J. Phys.: Conf. Ser. 119
[7] The application/json Media Type for JavaScript Object Notation (JSON), RFC 4627, http://tools.ietf.org/html/rfc4627
[8] memcached, a distributed memory object caching system, http://memcached.org/
[9] Application of Remote Debugging Techniques in User-Centric Job Monitoring, T. dos Santos et al., ACAT 2011
[10] Apache ActiveMQ, http://activemq.apache.org/
[11] STOMP, The Simple Text Oriented Messaging Protocol, http://stomp.github.com/
[12] JEM Viewer page, http://jem.physik.uni-wuppertal.de/JEM/
[13] Django, The Web framework for perfectionists with deadlines, https://www.djangoproject.com/
[14] EGI Operations Portal, http://operations-portal.egi.eu/vo
[15] Virtual Organization Membership Service, http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html
[16] Authentication and authenticated key exchanges, Whitfield Diffie, Paul C. Oorschot und Michael J. Wiener, Designs, Codes and Cryptography, 1992, Volume 2, Number 2, Seiten 107-125
[17] Message Authentication Code, http://de.wikipedia.org/wiki/Message_Authentication_Code