NORDUGRID-TECH-24
12/3/2012

Job Usage Reporter of ARC – JURA

Technical description

Péter Dóbé *
Gábor Szigeti †

*dobe@iit.bme.hu
†szigeti@niif.hu

1 Introduction

The *Job Usage Reporter of ARC* (JURA) is a component which is capable of creating job usage records of multiple kinds, and sending them to multiple accounting services.

2 Overview of functionality

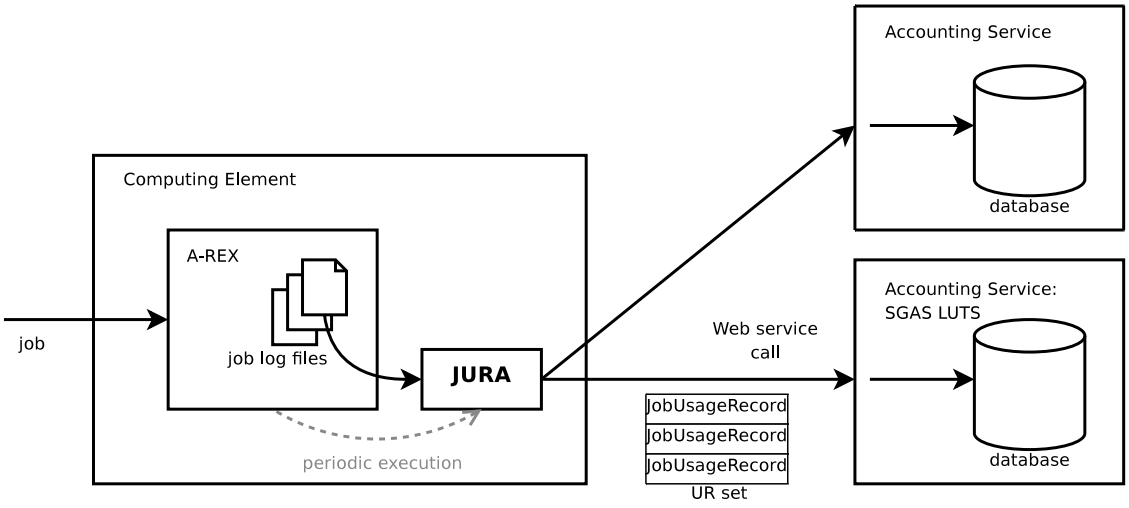

Figure 1: The usage reporting mechanism.

JURA is a stand-alone binary application which is periodically run by the A-REX (see Figure 1). There is no designated configuration file for JURA, nor is the configuration file of A-REX read directly by the application. Instead, options related to reporting are included within the job log files generated by A-REX or supplied via command line argument. The primary purpose of these job log files is to hold metadata about jobs starting, running and stopping. This is the main input of JURA. The format of the job log files is described in Section 3.2.

The application is run periodically. First, it processes the job log files, and based on the target accounting service specified in them, JURA creates usage records in a format appropriate for the target accounting service. Then these records are sent to one or more accounting services, referred to as *reporting destinations* in this document. Several reporting destinations are supported, these can be configured by the system administrator in the A-REX configuration file, and in addition, the user submitting the job can specify destinations in the job description.

Currently (in the December of 2011), the following job record formats are supported:

- Usage Record 1.0 XML format
- APEL (Accounting Processor for Event Logs) format
- Usage Record 2.0 (Computing Accounting Record) XML format

Currently JURA is capable to send Usage Record 1.0 formatted message to the SGAS Logging and Usage Tracking Service (LUTS). Communication with a LUTS server is done via a web service interface. JURA is securely authenticated by the server using X.509 certificates.

Sending APEL records to an APEL service is currently experimental, and it uses a python messaging library (stomppy).

The Usage Record 2.0 (CAR) formatted records are only stored locally, currently they are not sent to any service.

3 Operation

3.1 Invocation

JURA is a stand-alone executable application, executed by A-REX hourly (currently a hardcoded time interval, see Section 8). It has no separate configuration file, and does not process the A-REX configuration file. It receives all necessary options from A-REX in part through command-line arguments and mostly via variables inserted into the job log files (See Section 3.2). The following configuration variables can be present in the job log files:

- ***key_path*** – Path to the private key file used when submitting records.
- ***certificate_path*** – Path to the certificate file used when submitting records.
- ***ca_certificates_dir*** – Directory holding the certificates of trusted CAs.
- ***accounting_options*** – Additional configuration options for JURA.

The source of these variables is the "*grid-manager*" block of the A-REX configuration file (see Section 7).

The command line format of JURA is the following:

```
jura [-E <expiration_time>] [-u <url> [-u <url> [...]]] <control_dir> [<control_dir> [...]]
```

where *expiration_time* is the validity length of job log files in days, after which they are considered invalid; *control_dir* is the A-REX control directory for a mapped local UNIX user. The "*logs*" subdirectory of each control directory is traversed by JURA separately, in sequence.

The "*-u*" option can be used for interactive execution (e.g. from a terminal). In this case, usage data generated for each job is reported once to each of the specified destination URLs regardless of the content of the job log files, and no job log files are deleted.

3.2 Processing job log files

Job log files contain practically all input data (except those passed as command line arguments) for JURA. A-REX generates these files, at least two for each job and for each reporting destination: one at the time of job submission, another one after the job finishes, and possibly others at each start and stop event. Job log files are the main and only source of detailed resource usage information. Furthermore, they are used to communicate configuration parameters of JURA (see Section 3.1).

The job log files generated by A-REX reside under the directory *<control_dir>/logs*[**?**]. They have file name format *<ngjobid>.<random>*, where *ngjobid* is the identifier created for the job by A-REX, *random* is a randomly generated sequence of alphanumeric characters to avoid collision of different files pertaining to the same job.

A file consists of "*name=value*" lines, where "*value*" is either a job-related resource usage data or a configuration parameter. The URL of the reporting destination corresponding to the job log file is acquired from a "*jobreport=*" line in the A-REX configuration file. In addition to this server-side configuration, a limited number of destinations can be supplied by the submitter in the job description.

JURA generates records in the Usage Record (UR) format[**?**] proposed by the Open Grid Forum (OGF), using the information stored in the job log files. The generated UR is an XML representation holding consumption information for all commonly used resources and metrics. It can be extended by custom elements for non-standard resources and/or other types of job metadata. For a list of UR properties and their sources in the job log file, see Appendix A.

Some elements of UR are mandatory, these must all be present in the job log file to be able to generate a UR. For example, the job log file generated upon job submission contains no *status* entry, so this file is ignored, and no UR is generated from it.

An archiving functionality allows to store generated URs in a specified directory (see Section 7) on the disk. If enabled, valid UR XMLs are written to files named "*usagerecord.<ngjobid>.<random>*", where "*ngjobid*" and "*random*" match those of the source job log file. If a job log file is processed repeatedly – for example

because of temporary connection failures to a LUTS service – and a respective UR archive file already exists, then the UR is not generated again. Instead, the contents of the archive file are used without change (NB: the creation time stamp is also retained).

If interactive mode is not activated by the "-*u*" option (see Section 3.1), after successful submission to a reporting destination, the job log file is deleted, thus preventing multiple insertion of usage records. If submission fails, the log files are kept, so another attempt is made upon a subsequent run of JURA. This mechanism will be repeated until the expiration time passes (see "-*E*" command line switch in Section 3.1), at which point the next execution of JURA removes the file without processing.

3.3 Reporting to LUTS

In case of non-interactive invocation of JURA by A-REX, the generated URs are submitted to the accounting services specified by the reporting destination configuration parameters and if present, to the destinations specified in the job description as well. Under interactive mode of operation, they are submitted to the services given via the "-*u*" command line option. Reporting URs to several destinations is possible, but currently only SGAS LUTS destinations are supported.

LUTS has a simple custom web service interface loosely based on WS-ResourceProperties[**?**]. JURA uses the insertion method of this interface to report URs. The corresponding job log files are deleted after receiving a non-fault response from the service.

To increase communication efficiency JURA can send URs in batches provided that the server side supports this feature. LUTS accepts a batch of URs in a single request. The batch is an XML element called *UsageRecords*, containing elements representing URs.

The process of handling batches is the following: JURA does not send all usage records immediately after generation, but instead collects them in a batch until reaching the maximal number of records or until running out of job log files. The maximal number of URs in a batch can be set as a configuration parameter of JURA ("*jobreport_options*", see Section 7).

4 Security

The JURA executable runs with the same user privileges as the A-REX. The owner of a job log file is the local user mapped for the submitter entity of the corresponding job. Since these files contain confidential data, A-REX restricts access to them allowing only read access for the job owner, thus when JURA is executed by A-REX it is allowed to read and delete job log files.

All usage records are submitted using the X.509 credentials specified by the value of the `jobreport_credentials` value of the A-REX configuration file. No proxies are used.

The only access restriction made by a LUTS service is matching the Distinguished Name of the client (in this context JURA) with a set of trusted DNs. When access is granted, policies are then applied by LUTS, allowing either publishing and/or querying rights. Clients with publishing right can insert any UR, regardless of content. By default, querying right only allows retrieving URs pertaining to jobs submitted by the querying entity.

5 Implementation and API

JURA as part of the ARC software stack is written in C++, and utilizes the functionality provided by the ARC libraries, including secure HTTPS communication provided by the ARC plugable TLS and HTTP modules.

The modular design is also present in the usage reporting part of the JURA code, making it possible to extend JURAs support of accounting services. To create a JURA module one should simply write a C++ class which inherits from the abstract `Arc::Destination` class, and it must extend the two methods:

- `static Arc::Destination* Arc::Destination::createDestination(Arc::JobLogFile&)`

- `void Arc::Destination::report(Arc::JobLogFile&)`

The static `createDestination` method should initialize a object of the specialized class, using the configuration options specified in the passed `Arc::LogFile` object, and the memory allocated by the method should be freed by the caller. Then the `report` method should carry out the transfer of the UR, represented by the `JobLogfile` object, to the accounting service.

6 Installation and deployment

As part of ARC, JURA is distributed through usual ARC distribution channels, namely through Linux (RedHat, Fedora, Debian and Ubuntu), NorduGrid, EMI and EGI repositories. JURA is part of the `nordugrid-arc-arex` package and once installed the executable will be located in the `/usr/lib/arc` directory. Detailed install instructions can be found at the NorduGrid website.

Instructions on how to build JURA from source can also be found on the NorduGrid website.

The name of the reporting executable should be specified in the `arc.conf`, with the `jobreport_publisher` configuration command. It has to be a relative path starting from the *libexec/arc* directory under the install location. If it is not specified, the name `jura` is used.

The usage reporting can also be performed manually provided that access to the credentials are granted, by executing JURA with the proper command line arguments (see Section 3.1). The example command below will send generated usage records from the job log files in the standard location, "*/tmp/jobstatus/logs*" and send them to LUTS services. Files older than a week are deleted without processing.

```
jura -E 7 /tmp/jobstatus
```

7 Configuration

JURA is configured through the configuration file of A-REX[**?**]. As it was already mentioned in Section 3.1, JURA does not process the A-REX configuration file directly; the configuration values are propagated to JURA through the job log files. The following configuration commands in the "*grid-manager*" block of the A-REX configuration file are relevant for JURA:

- ***jobreport****=[URL ... number]* – specifies reporting destination URLs. Multiple entries and multiple URLs are allowed. *number* specifies for how many days the old records have to be kept if failed to be reported.
- ***jobreport_publisher****=filename* – specifies the name of the executable which will be run by the A-REX periodically to publish job reports if a jobreport URL is specified. The executable will be searched in the nordugrid *libexec* directory. The default name is jura.
- ***jobreport_credentials****=[key_file [cert_file [ca_dir]]]* – specifies the credentials for accessing the accounting service. Respectively path to private key (`key_file`), path to certificate (`cert_file`), and path to CA certificates directory (`ca_dir`)."
- ***jobreport_options****=[options]* – specifies additional options for JURA.

The *jobreport_options* configuration command allows passing a generic option string to JURA verbatim. This string is interpreted by JURA as a comma-separated list of "*name:value*" pairs (note the colon!), which represent service-related settings and extended reporting parameters. The job reporting options currently defined are:

- ***urbatch****:size* – sets the maximal number of URs in the batch sent within one request. Zero value means unlimited batch size. Default is 50.
- ***archiving****:dir* – enables archiving of generated URs in the given directory. If the directory does not exist, an attempt is made to create it. If this option is absent, no archiving is performed.

The example below is a part of the "*grid-manager*" block of the A-REX configuration. It enables logging of URs to two hosts, using the host credential files (placed in the standard locations), with a maximum of 50 URs per batch. Generated URs are archived in the directory "*/var/urs*". Job log files expire after a week.

```
...
jobreport="https://luts1.nordugrid.org:8443/wsrf/services/sgas/LUTS"
jobreport="https://luts2.nordugrid.org:8443/wsrf/services/sgas/LUTS 7"
jobreport_credentials="/etc/grid-security/hostkey.pem /etc/grid-security/hostcert.pem
     /etc/grid-security/certificates"
jobreport_publisher="jura"
jobreport_options="urbatch:50,archiving:/var/urs"
...
```

8 Limitations and future plans

In the following list some issues which limits the functionality of JURA is described:

- The time frequency of running JURA is not configurable. It is a hardcoded value in A-REX: 3600 seconds, i.e. one hour.
- The number of user-supplied reporting destinations is limited for the sake of robustness. This upper limit is hardcoded in A-REX: max. 3 destinations are parsed from JSDL, and max. 1 from RSL.
- The current implementation of JURA and A-REX supports only one expiration time for all the reporting destinations. Even though the configuration enables the specification of different expiration values per reporting destination, it is not taken into account by the system, the last value is used as the common expiration time value.
- It is not possible to use different credentials per destinations.
- Some optional UR properties are not supported (see App. A).
- Memory is not reported correctly. A bug in GNU "time" results in all memory usage set incorrectly as zero.
- Some necessary extensions to the generated UR are not yet filled though the information is already collected in the job log files.
- Detailed user identity information based on the X.509 proxy certificate content or other submitted credentials is missing from URs.

There is also plan to extend JURA with the following features:

- Support for other accounting systems besides LUTS, especially the EGEE Accounting Service. However, appropriate documentation on the service interfaces is missing as of now.
- Extend JURA to be able to read credentials from the command line in interactive mode.
- Enable generating coarser-grained "*Aggregate URs*" from multiple URs.
- Investigate the subject of project-related charging: who is responsible for determining the "*charge*" value; what rules should be applied?

A Generated Usage Record

The following table shows which properties in OGF UR[**?**] are filled, what data source was used for them, and which properties are missing.

Generated UR Property	**Source (job log entry)**	**Information content**
RecordId	nodename, ngjobid	Globally unique identifier for UR
GlobalJobId	globalid	Globally unique identifier of job: XML element as defined by BES
LocalJobId	localid	CE-specific identifier of job
GlobalUserName	usersn	DN of submitting user's certificate
LocalUserId	localuser	POSIX user on CE executing the job
JobName	jobname	Name of job, as given in job description
Status	status	Status of job
WallDuration [ISO 8601 duration]	usedwalltime [s]	Wall-clock time used by job
CpuDuration [ISO 8601 duration]	usedcputime [s]	CPU time used by job
StartTime [ISO 8601 time stamp]	submissiontime [ISO 8601 time stamp]	Time instant the job started
EndTime [ISO 8601 time stamp]	endtime [ISO 8601 time stamp]	Time instant the job ended
MachineName	nodename	Name of the machine where the job ran (first node from colon-separated list put into element)
Host	nodename	System hostname(s) where the job ran (nodes from colon-separated list put into separate elements)
SubmitHost	clienthost	System hostname the job was submitted from
Queue	lrms	Name of the queue from which the job was executed
ProjectName	projectname	Name of project, as given in job description
Memory [average virtual, kB]	usedmemory [kB]	Average total memory used by job
Memory [max physical, kB]	usedmaxresident [kB]	Maximal resident memory used by job
Memory [average physical, kB]	usedaverageresident [kB]	Average resident memory used by job
NodeCount	nodecount	Number of nodes (physical machines) involved in the job
ProcessID	**MISSING**	The process ID(s) of the job
Charge	**MISSING**	Total charge of the job (money or abstract credits)
Network	**MISSING**	Network usage of job
Disk	**MISSING**	Disk usage of job
Swap	**MISSING**	Swap usage of job
Processors	**MISSING**	Number of processors used or requested
TimeDuration	**MISSING**	Additionally measured time duration(s)
TimeInstant	**MISSING**	Additionally identified time instant(s)
ServiceLevel	**MISSING**	Quality of service associated with usage
Extended UR Property	**Source (job log entry)**	**Description**
RuntimeEnvironment	runtimeenvironment	Requested runtime environment, specified in job description (RTEs from space-separated list put into separate elements)