The machine/job features mechanism

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 092032 doi:10.1088/1742-6596/898/9/092032

The machine/job features mechanism

M Alef[1], T Cass[2], J J Keijser[3], A McNab[4], S Roiser[2], U Schwickerath[2] and I Sfiligoi[5]

[1] Karlsruhe Institute of Technology
[2] CERN
[3] NIKHEF
[4] University of Manchester
[5] Fermi National Accelerator Laboratory

E-mail: andrew.mcnab@cern.ch

Abstract. Within the HEPiX virtualization group and the Worldwide LHC Computing Grid's Machine/Job Features Task Force, a mechanism has been developed which provides access to detailed information about the current host and the current job to the job itself. This allows user payloads to access meta information, independent of the current batch system or virtual machine model. The information can be accessed either locally via the filesystem on a worker node, or remotely via HTTP(S) from a webserver. This paper describes the final version of the specification from 2016 which was published as an HEP Software Foundation technical note, and the design of the implementations of this version for batch and virtual machine platforms. We discuss early experiences with these implementations and how they can be exploited by experiment frameworks.

1. Introduction

The discussions which eventually led to the Machine/Job Features (MJF) specification began within the HEPiX[1] Virtualisation Working Group, which was set up to look at the integration of VM-based resources within High Energy Physics. The ideas were generalised to apply to batch systems and the work was carried forward by the Machine/Job Features task force of the Worldwide LHC Computing Grid[2] (WLCG). The specifications, implementations, and monitoring provided by the task force are published on the task force's web site[3].

The completed version of the Machine/Job Features (MJF) specification was described in HEP Software Foundation technical note HSF-TN-2016-02[4] "Machine/Job Features" published in February 2016. As the schema is designed in an extensible way, further additions are expected and can be readily accommodated, but this paper describes the situation at the end of 2016.

One notable feature of the mechanism is that the information provided can and should be correct at the level of the individual worker node, rather than a site or cluster-wide average.

2. Definitions

The terms used in the specification are based on resources presented as batch queues and job slots. On VM-based systems, references to "job" in the specification are to be interpreted as "virtual machines" and "machines" as "hypervisors". When jobs are running within virtual machines, the entity that provides the system level configuration or contextualization of the VM acts as the "resource provider".

3. Environment variables

For each job, two environment variables may be set, with the names $MACHINEFEATURES and $JOBFEATURES.

These environment variables are the base interface for the user payload. Their values must be provided for the job by the resource provider. In the case of virtual machines on Infrastructure-as-a-Service cloud platforms, the virtual machine may discover the values to set for the environment variables from "machinefeatures" and "jobfeatures" metadata keys provided by the resource provider via the cloud infrastructure. These metadata keys should only be accessed once in the lifetime of each virtual machine. Alternatively, the values to set may be supplied as part of the contextualization of the virtual machines.

4. Directories, keys, and values

The environment variables point to directories created by the resource provider. Inside, the file name is the key, the contents are the values, so that files can be referred to with expressions like $MACHINEFEATURES/shutdowntime . The directory name should not include the trailing slash. These directories are either local directories in the filesystem or sections of the URL space on an HTTP(S) server. The user positively determines whether the files are to be opened locally or over HTTP(S) by checking for a leading slash or the prefix http:// or https:// respectively. This can often be achieved using library functions which transparently handle local files and remote URLs when opening files.

For example in Python,

```
import urllib
shutdowntime = int(urllib.urlopen(os.environ['MACHINEFEATURES']
                                  + '/shutdowntime').read())
```

Unlike metadata keys, the key/value files may be accessed multiple times to check for changes in value or in the absence of caching by the user. An HTTP(S) server may provide HTTP cache control and expiration information which the user may use to reduce the number of queries. All files in the directories must be readable by both the user and the resource provider services, and have file names which only consist of lowercase letters, numbers, and underscores.

The specification defines the key/value pairs set out in Table 1.

Table 1. Key/Value pairs in the Machine/Job Features specification

$MACHINEFEATURES	$JOBFEATURES
total_cpu	allocated_cpu
hs06	hs06_job
shutdowntime	shutdowntime_job
grace_secs	grace_secs_job
	jobstart_secs
	job_id
	wall_limit_secs
	cpu_limit_secs
	max_rss_bytes
	max_swap_bytes
	scratch_limit_bytes

5. Implementations

The role of the WLCG task force shifted during 2016 to supporting the development and deployment of software to create the Machine/Job key/value pairs as described in the technical note.

RPMs were provided for PBS/Torque, HTCondor, Grid Engine, developed with the help of sites running these batch systems. These RPMs install scripts which determine the relevant values from the operating system and batch system parameters if available, potentially overridden by values chosen by the site administrators. A base set of scripts, common to all implementations, populate a $MACHINEFEATURES directory on the local filesystem. An additional flavour of "only-mf" is provided which includes only these common scripts and only populates $MACHINEFEATURES.

For each batch flavour, additional scripts are provided which parse job-specific information to populate the $JOBFEATURES directory. For example with HTCondor the job's $_CONDOR_JOB_AD file is parsed. Where possible the same code is used for all flavours with flavour-specific code put into dedicated scripts. This modular approach also allows sites with unanticipated configurations to replace only one or two scripts rather than maintain a fork of the whole framework.

As well as RPMs, it is also possible to install the scripts directly using a configuration management system. In most cases, the RPMs or scripts can be installed without any site-specific customisation other than the measured value of HEPSPEC06[5] for the hardware.

Machine/Job Features key/value pairs are also provided by default at VM-based sites managed by Vac[6] or Vcycle[7].

6. Deployment

During 2016 the task force supported the deployment of the MJF scripts at sites willing to test the software.

In total, a dozen mainly Tier-1 and larger Tier-2 sites were providing Machine/Job Features using these scripts or systems by the start of CHEP 2016.

To monitor the progress of the deployment and to validate the sites as they are enabled, a SAM/ETF monitor has been written and deployed. This is run at all WLCG sites and provides pass, fail and warning outcomes plus a detailed report of the outcome for each key/value pair tested. The tests are included in the MJF scripts distribution and may be run directly by sites or within test jobs they themselves submit while deploying the MJF functionality.

7. Applications

The sites participating so far have also been invaluable in studies of the variation in HEPSPEC06 and the DIRAC 2012 fast benchmark[8] (DB12) between processor versions and sites, including one study[9] by LHCb presented at CHEP 2016.

The MJF values are also used natively by DIRAC[10] to discover the CPU and wallclock limits of a job slot, as an alternative to DIRAC's support for discovering this information using the commands provided by the various batch systems. MJF provides a consistent API at all sites, rather than the inconsistent interface even within a given batch system where the commands necessary to discover job limits are not available to jobs themselves at some sites. This information is used by LHCb, for example, to calculate how many Monte Carlo events may be simulated within the available time.

DIRAC's support for MJF has been of particular use in the DIRAC VMs used by LHCb and the GridPP DIRAC service, and in the ATLAS Vacuum VMs[11]. The MJF API is used to discover the maximum lifetime of the VM itself, which is used by the relevant pilot framework to request a payload job of the correct duration, as a substitute for the job slot time limit API provided by the various batch systems. These VMs are examples of the wider "Vacuum

Platform" described in HSF-TN-2016-04[12] also presented[13] at CHEP 2016, which relies on the MJF mechanism.

One novel feature introduced by the MJF API is the pair of $MACHINEFEATURES/shutdowntime and $JOBFEATURES/shutdowntime_job keys which give any planned shutdown time either for the whole machine or for the job in question. This may be used to signal that compatible jobs should finish early, and can be used by sites to avoid a period of draining before shutting down a machine. Instead of not accepting new jobs for a period equal to the maximum job lifetime before the shutdown and having an increasingly idle machine as that time approaches, a site can announce a shutdown time, limit new jobs to ones compatible with this feature (from specific experiments for instance), and know that the jobs will try to occupy all the available time up to the shutdown time.

8. Summary

We have presented a description of the Machine/Job Features mechanism developed by HEPiX and WLCG working groups and now specified by an HEP Software Foundation technical note. We have described implementations provided and deployed by sites, and explained how this common API may be used by applications to improve the efficiency of their use of the resources presented by sites.

References

[1] https://www.hepix.org
[2] Bird I 2011 *Annual Review of Nuclear and Particle Science* **61** 99-118
[3] https://twiki.cern.ch/twiki/bin/view/LCG/MachineJobFeatures
[4] MAlef M et al HSF-TN-2016-02 "Machine/Job Features Specification" (HEP Software Foundation)
[5] http://w3.hepix.org/benchmarks/
[6] McNab A et al 2014 *J. Phys.: Conf. Ser.* **513** 032065
[7] McNab A, Love P, MacMahon E 2015 *J. Phys.: Conf. Ser.* **664** 022031
[8] https://github.com/DIRACGrid/DB12
[9] Charpentier P "Benchmarking worker nodes using LHCb
[10] Tsaregorodtsev A et al 2008 J. Phys.: Conf. Ser. 119 062048 simulation productions and comparing with HEP-Spec06", presented at the CHEP 2016 conference.
[11] Taylor R et al "Consolidation of Cloud Computing in ATLAS", presented at the CHEP 2016 conference.
[12] McNab A HSF-TN-2016-04 "The Vacuum Platform" (HEP Software Foundation)
[13] McNab A "The Vacuum Platform", presented at the CHEP 2016 conference.