WM Technical Evolution Group Report

Davide Salomoni, INFN
for the WM TEG

February 7, 2012

The WM TEG, organization

- Mailing list, wlcg-teg-workload-mgmt@cern.ch
  - 48 people currently subscribed; representation from experiments, EMI, EGI and sites.
  - Contact Torre/Davide if interested to join. (you can also self-subscribe to the list)
  - Archives: https://groups.cern.ch/group/wlcg-teg-workload-mgmt/default.aspx
- Indico page, https://indico.cern.ch/categoryDisplay.py?categId=3768
- Wiki page, https://twiki.cern.ch/twiki/bin/view/LCG/WorkloadManagementTechnicalEvolution
- About ~~300~~ 330 messages exchanged on the mailing list with sometimes lively discussions. Many thanks to all the contributors!
- Two face-to-face meetings (3-4/11/2011 @ CERN and 26-27/1/2012 @ NIKHEF).

Main areas tracked so far

- Commonalities with pilots and frameworks
- Support of jobs requiring whole nodes or multiple cores
- Support of CPU affinity and CPU sets
- Tagging I/O vs. CPU intensive jobs
- Requirements for a CE service
- Summary of the use of the IS by WLCG experiments
- Use of virtualization technologies
- Cloud computing
- MUPJ handling – mainly run by the Security TEG

Pilots and frameworks

- Reference: https://twiki.cern.ch/twiki/bin/view/LCG/WLCGWMTEGPilotCommonalities
- Status of the development/integration of the frameworks used by the 4 LHC experiments.
- Commonalities in the Grid submission layer.
  - ALICE, LHCb : direct submission to the CREAM CE (or to the gLite WMS).
  - CMS : glideinWMS fully relying on Condor (using GRAM or CREAM).
  - ATLAS : currently testing glideinWMS to handle Grid jobs submissions.
    - See talk by Rod Walker at our latest F2F.
    - Will scale up tests by making some glexec sites use only glideinWMS, define gFactory support model with CERN/OSG/UCSD.

Pilots and frameworks: streamed submission

- All VO's expressed interest for an extension of the CE interface to be able to *constantly keep N jobs queued at a given queue until a given condition is satisfied.* (e.g. no more work to be done, or until date X)
  - This should help keeping a constant pressure on sites without having to do explicit submissions (and AuthN steps) to the CE.
  - All jobs of a given "stream" will share the same jdl (i.e. the same requirements, I/O specifications, etc.)
  - The CREAM developers considered this feasible; the CREAM interface will therefore be extended to support requests for streamed submission.
  - We still need to check what can be done on OSG.

Pilots and frameworks: common job wrapper for pilots

- Currently, the pilot framework security relies on the fact that the pilot wrapper calls glexec to execute the user payload.
  - This requires review of all and each of the VO pilot frameworks. (to be repeated anytime something is changed)
  - This concerns also how credentials are managed.
- It was proposed to find commonalities in the pilot bootstrap script to factor out this part and enforce glexec calling, making the review unnecessary.
- Providing a common way of executing pilots would also reduce / eliminate the need to install VO-specific elements (e.g., VO-boxes) within a site boundary.
- During the discussion it emerged that this part is deeply buried in the Condor-based pilot, and is very difficult to factorize without impacting efficiency. In addition, that part would become as critical as glexec itself (but more complex).
  - It is not clear whether a common solution can be reached, nor the actual scope of the problem. Further investigations are needed on this topic.
  - In case a common solution can be found, EMI is interested in implementing it in CREAM and WMS.

Whole node / multi-core jobs

- Reference: https://twiki.cern.ch/twiki/bin/view/LCG/WMTEGMulticore
- A possible solution is represented by dedicated “whole-node” queues.
  - But this solution often leads to static partitioning of resources; resources also tend to have an increasingly number of cores per system, so a single system accounts for an increasingly large share of resources.
- Dynamic resource sharing is preferred by several sites, using standard Grid tools and commonly used LRMS’. This means specifying whole node / multi-core requirements in the JDL/RSL.
  - Whole node or not.
  - Number of requested cores. (fixed)
  - Variable number of cores. (e.g. minimum and/or maximum)
  - Requested total memory. (or per-core memory if the # of cores is not exactly specified)
- Advance reservation policies, if desired, should be implemented by sites, not by CE’s.

Whole node / multi-core: support

- CREAM
  - CREAM in EMI-1 already supports some JDL attributes allowing resource requirements.
    - Prompted by MPI communities; total # of CPUs, requested minimum # of cores per node, total # of nodes, whole node or not.
    - SGE support for these features in CREAM (provided in EMI by CESGA) is currently missing.
    - Need to add: min/max # of cores, total requested memory. (or per-core memory if # of cores is not specified)
- GRAM
  - The RSL attribute *xcount* was defined by Teragrid for MPI jobs and is usable on OSG.
  - Whole nodes are supported on PBS, LSF and Condor.
  - Current status of multi-core in GRAM: PBS support OK; LSF and SGE support to be investigated; Condor not supported.
  - Range of cores are currently not supported.
- Experiments
  - Both ATLAS and CMS stated they can work on their respective frameworks to accommodate arbitrary N-core job sizes, and that the resource requirements can be specified using either JDL (for CREAM) or RSL (for GRAM).

Whole node / multi-core: proposed changes and workplan

- We need to define a new JDL attribute for memory and implement requests for a variable # of cores.
  - CREAM developers agreed to implement this (manpower estimate one week) – to be formalized with EMI.
  - The GRAM job manager should be easily modified (as long as the underlying LRMS supports the feature). Need to investigate with the Globus team; if Globus not interested, the OSG software team can write a patch.
- Testing, two phases:
  1. Experiments will require either whole nodes (without dedicated queues), or an exact number of cores using JDL/RSL. We will start with # cores = 4 to make things simpler.
  2. Experiments may also require a variable # of cores; the job will be able to utilize as many cores as are made available.
     - Need to define an environment variable telling the job how many cores / how much memory the job has been allocated; will build on the proposal made last year by the HEPiX-virt WG.
- Sites / experiments:
  - CMS will modify WMagent / glideinWMS to support multi-core scheduling.
  - Potentially all sites currently supporting whole node testing for CMS plus GRIF will join the test.
- Information system: will need to define a way for sites to flag the max # of cores they support and whether they support whole nodes, and/or generic multi-core requests.

CPU affinity / CPU sets

- Reference: https://twiki.cern.ch/twiki/bin/view/LCG/WMTEGCPUAffinity
- *Soft* CPU affinity, enforced by the Linux kernel scheduler, may not be enough to prevent CPU intensive jobs to disrupt other jobs running on a shared worker node.
- Possible solutions:
  - VO wrapper script; the VO uses tasksets (CPU pinning) to assign processor affinities.
  - Native LRMS solutions would be preferable. LSF supports CPU pinning with a variety of options. Condor supports CPU affinity.
  - New features available in RHEL 6.
    - CPU sets, extend taskset to also include other parameters (e.g. memory); supported by PBS/Torque.
    - Linux containers (lxc), or *chroot* on steroids; early tests with PBS and Condor. Perhaps useful in the context of glexec as well. (→ security TEG?)
    - These solutions all require that WLCG jobs be run on RHEL 6 WN.

I/O- vs. CPU- intensive jobs

- Reference: https://twiki.cern.ch/twiki/bin/view/LCG/WMTEGIOvsCPUjobs
- Nodes can often support more CPU-bound than I/O-bound jobs. Some sites have found it beneficial to schedule user jobs onto different nodes, depending on the job type.
  - A hint to a site LRMS may help sites in scheduling jobs.
  - Local heuristics soon become complex.
- We propose to introduce a new parameter to specify whether a job is CPU- or I/O-bound. The parameter is a scalar in the range [0,1] (0=CPU-bound; 1=I/O bound).
  - The parameter does not translate into a specific resource to be allocated to the job; it may help sites distributing the jobs in an optimal way.
- Initially, start with only 0 and 1; as knowledge of job characteristics improve, a VO can specify different values.
  - E.g., CMS suggested they might assign 1 to merge and logcollect jobs, and 0 to other jobs.
- Pilot frameworks should honor the resource constraints (CPU vs. I/O) between the pilot and the jobs the pilot executes.
- Sites that do not want to make use of these tags may simply ignore them.
- Considerations similar to those made for multi-core jobs apply for what regards actual implementation / testing of these tags with CREAM and GRAM.
- Further tags may be added at a later date.

Requirements for a CE service

- Reference: https://twiki.cern.ch/twiki/bin/view/LCG/WMTEGCEService
- A "CE": a secure abstraction of resources at a site hiding the implementation details from the user communities.
  - For the foreseeable future we believe there will be multiple CE technologies in place
  - Current interfaces are CREAM and GRAM.
  - Possible input to technology providers: identification of a set of logical API's that each CE should provide to be usable by WLCG experiments.
- Some sites flagged that their current LRMS may have scaling / support issues and expressed the possibility to change LRMS. There is obviously the need that a CE supports the LRMS installed at a site.
  - EMI CREAM: PBS, LSF, SGE, partially Condor.
    - SLURM support in CREAM recently requested by EGI.
  - Globus GRAM: PBS, LSF, Condor, unofficially SGE, SLURM.
- A site requirement is the possibility to define a "CE alias" with a pool of individual CE's behind the alias, where the individual CE's share state data.
  - Partial support for this is in the CREAM version released with EMI-1.
  - A full solution for CREAM is planned for EMI-3.

Use of the Information System

- Reference: https://twiki.cern.ch/twiki/bin/view/LCG/WMTEGInformationServiceSummary
- Discussions in the TEG, plus other documents produced in 2011, showed that:
  - WLCG experiments tend to use the IS for static/quasi-static information, and in general for limited purposes. The general pattern is often “use for bootstrap, then refine with our own heuristics”.
  - The WMS may work today without querying the IS through explicit requirements, with its “-r” option, or with its recently introduced “replanning” feature.
  - Quality control of the IS content is important and needs to be automated. WLCG experiments have learnt by experience that no info is at least not worse than unreliable dynamic information.
  - Reliable storage information would be certainly desirable, but it is currently not available.
  - Having cached info in the IS is considered to be vital to overcome possible services outages.
  - Other, future services related e.g. to the integration of Cloud resources might possibly use the IS. However, it is still early to derive suggestions in this area.
- The conclusion is that, given also the late-binding nature of the experiment frameworks, in the future WLCG experiments will continue to need mostly a simple discovery service.

Virtualization technologies

- Reference: https://twiki.cern.ch/twiki/bin/view/LCG/WLCGWlMgmtTEGVirtualization
- Virtualization is being used at many sites (and it has been for several years in some cases) and also tested by some VO's.
  - Main advantages and disadvantages were discussed.
- We can differentiate between virtualization of service nodes vs. virtualization of non-reliable resources (e.g. worker nodes). For the latter:
  - Virtualization penalties must be carefully considered. It is normally not the case that a generic virtualization framework can be deployed without specific optimizations for efficiency.
    - E.g. take I/O penalties, which may be non-negligible. (ballpark figure: 20%)
  - Static virtualization of batch resources is an obvious possibility; however, a more dynamic allocation of VMs would bring in the possibility e.g. of automating deployment of intrusive updates, or of optimizing allocation and encapsulation of multi-core VMs used for multi-core jobs.
    - However, a scalable mechanism, possibly linked to an LRMS, is required to properly handle dynamic allocations. Early tests suggest that both e.g. Torque and LSF may have troubles scaling to several (tens of) thousands of VMs, at least if not properly configured.

Cloud computing

- Reference: https://twiki.cern.ch/twiki/bin/view/LCG/WLCGWlMgmtTECloudComputing
- Distinction between IaaS, PaaS and SaaS.
  - VO's already expressed interest in accessing IaaS entry points to sites.
  - In order to ensure uniformity of resource access across sites, we might define common "sizes" for provisioned resources.
- Batch cloud factory
  - A *dynamic* way of provisioning cloud resources through e.g. EC2 or OCCI interfaces.
  - Probably compelling for WLCG if performance penalties are under control, and if resource requirements can be easily fulfilled beyond the standard IaaS paradigm.
  - Accounting should be integrated with normal "batch" accounting. You pay for what you get.
  - Ongoing talks with CREAM developers to see if a batch factory can be integrated.
  - Adopt recommendations by the HEPiX-virt WG and possibly the EGI policy on the instantiation of VMs.
  - Authentication and authorization need to be examined more in detail (move beyond a simply username/password mechanism).
- Early tests on Cloud instantiations at WLCG sites include CERN's lxcloud and INFN's WNoDeS. (participating to the EGI distributed cloud TF)
  - Need to evaluate fairsharing, integration of multiple Clouds and possible relationships with an IS.

Outlook (1)

- ATLAS glideinWMS tests – while a VO activity, it is useful to share results within the TEG.
- Implementation / test of "streamed submissions" w/ CREAM and possibly GRAM.
  - CREAM: issue feature request to EMI; GRAM: investigate with Globus and OSG.
- Continue investigation on whether a common way to execute pilot jobs can be defined.
- Definition of new JDL attributes and provisioning of variable # of cores for multi-core jobs; testing.
  - CREAM: issue feature request to EMI; GRAM: investigate with Globus and OSG.
  - Define environment variable for variable # of cores.
  - Interactions with the IS.
- CPU affinity: continue evaluation of possible solutions, test report. (HEPiX?)

Outlook (2)

- Implementation / test of CPU- vs. I/O-bound tagging.
  - CREAM: issue feature request to EMI; GRAM: investigate with Globus and OSG.
- Requirements for a CE.
  - Testing of alias functionality in CREAM and WMS (EMI-1).
  - Identification of a set of logical API's to be provided by each CE technology?
- Virtualization technologies.
  - Share experience of both sites and VO's re use cases, configuration and performance. Possibly link to the support of multi-core jobs / CPU- vs. I/O-bound tagging.
- Cloud computing.
  - Adopt HEPiX recommendations (e.g. wrt VM capabilities) and consider the EGI policy for what regards running VMs.
  - Define common methods to access resources (e.g. via EC2 or OCCI) and common sizes to ensure consistency and avoid effort duplication; track possible integration into an IS.
  - Explore integration into an LRMS and with CREAM for the provisioning of cloud instances.
  - Track AuthN/AuthZ developments and integration of multiple cloud sites.

Thanks!

wlcg-teg-workload-mgmt@cern.ch