Evolution of CMS workload management towards multicore job support

A Pérez-Calero Yzquierdo[1,2], J M Hernández[2], F A Khan[3], J Letts[4], K Majewski[5], A M Rodrigues[5], A McCrea[4] and E Vaandering[5] on behalf of the CMS Collaboration.

[1] Port d'Informació Científica (PIC), Universitat Autónoma de Barcelona, Bellaterra (Barcelona), Spain.
[2] Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas, CIEMAT, Madrid, Spain.
[3] National Centre for Physics, Quaid-i-Azam University, Pakistan.
[4] University of California San Diego, USA.
[5] Fermi National Accelerator Laboratory, USA.

E-mail: aperez@pic.es

Abstract. The successful exploitation of multicore processor architectures is a key element of the LHC distributed computing system in the coming era of the LHC Run 2. High-pileup complex-collision events represent a challenge for the traditional sequential programming in terms of memory and processing time budget. The CMS data production and processing framework is introducing the parallel execution of the reconstruction and simulation algorithms to overcome these limitations. CMS plans to execute multicore jobs while still supporting single-core processing for other tasks difficult to parallelize, such as user analysis. The CMS strategy for job management thus aims at integrating single and multicore job scheduling across the Grid. This is accomplished by employing multicore pilots with internal dynamic partitioning of the allocated resources, capable of running payloads of various core counts simultaneously. An extensive test programme has been conducted to enable multicore scheduling with the various local batch systems available at CMS sites, with the focus on the Tier-0 and Tier-1s, responsible during 2015 of the prompt data reconstruction. Scale tests have been run to analyse the performance of this scheduling strategy and ensure an efficient use of the distributed resources. This paper presents the evolution of the CMS job management and resource provisioning systems in order to support this hybrid scheduling model, as well as its deployment and performance tests, which will enable CMS to transition to a multicore production model for the second LHC run.

1. Multicore jobs for CMS during LHC Run 2

LHC experimental collaborations have dedicated a significant part of their programming efforts in recent years into developing multithreaded applications running on more than one processor core. The motivation for this comes, firstly, from the evolution of computer hardware, as over the last decade single-core CPUs have reached the limit in processor speed. Continuing to enhance the overall CPU performance therefore has progressed by adding more processor units to the CPU (*cores*). Secondly, the anticipated experimental conditions of the second LHC run (scheduled to start in Summer 2015) of higher energy and luminosity, imply the need to process increasing data volumes with increasing event complexities. This results in higher per-event processing time and memory usage as compared to the conditions during LHC Run 1. In this context, multicore applications aim at fully exploiting the capabilities of current multicore CPU architectures, in order to, for example, reduce memory consumption per core [1].

Evolving LHC experiments computing to use multicore CPUs scattered across the *Grid* has required not only to adapt the applications themselves but also the development of new resource allocation and scheduling tools and procedures, the latter being the main focus of this paper. This evolution has happened during the long shutdown period of the LHC (2013-2014) in parallel to the upgrades of the collider and of the detectors, all needed to continue pushing the boundaries of High Energy Physics in the coming years.

The CMS priority in the deployment of multicore resources for the start of data taking for the LHC Run 2 is to provide sufficient computing power to perform multithreaded prompt data reconstruction tasks. These workflows will be processed at the Tier-0 (T0) and Tier-1 sites (T1s), as they are estimated to require T0 plus 50% of the T1 available CPUs [2]. Thus, for the first phase of multicore deployment in 2015, the focus is on T0 and T1s sites. Simulation and digitisation tasks will then be switched to multithreaded algorithms as well, which will then require the deployment of multicore resources to CMS Tier-2 sites. In any case, single core and multicore jobs will coexist during Run 2, therefore a strategy for scheduling both types of jobs is mandatory. CMS model foresees the use of multicore pilots (see next section) to manage 100% of the resources at the sites, starting with the T0 and T1s in 2015.

This document is organized as follows: section 2 describes the infrastructure and strategy developed by CMS to support a simultaneous scheduling of single core and multicore jobs. Section 3 summarizes the scale tests conducted in order to examine the validity and performance of such a model. Finally, section 4 presents the conclusions from this study and the outlook for future developments towards the imminent restart of the LHC in 2015.

2. CMS workload management and submission infrastructure

The CMS workload management (WM) system, dedicated to centralized workflows such as data reconstruction and the generation of simulated events, is based on the concept of WMAgents [3]. A series of nodes under supervision of CMS WM team of operators, handle tasks such as workload splitting into jobs, job assignment to the appropriate list of computing sites, job prioritization, retrial of unsuccessful jobs and merge of output files and log collection. Resource allocation is performed by the CMS submission infrastructure (SI), which employs GlideinWMS [4, 5], a tool built on top of the HTCondor [6] batch system that matches jobs to resources managing a transient pool of computing resources controlled by so-called *pilot jobs*, which schedule user jobs in a pull mode.

Both elements are integrated in the schema summarized in Fig. 1. WMAgents populate central job queues. The GlideinWMS front-end machines perform a first stage of matching, contacting the factory machines, which submit pilots to all grid sites matching job description request. Pilots enter the queues at the local batch systems. Running pilots define a virtual pool of computing job slots. A second stage of match-making, of jobs to pilots, is then handled by the negotiator machines. Workload is finally pulled to the worker nodes (WNs) local to a specific site. See [7] for a detailed explanation of the integration of the different types of workloads and resources into the so-called Global Pool framework.

CMS main tool for the integrated scheduling of jobs with different core requests are multicore pilots, designed to control several batch slots and featuring a dynamic partitioning of the allocated resources [8]. Pilots take N slots from the local batch system and then arrange M internal slots according to job requirements. In the example shown in Fig.2, a single pilot managing 4 cores in a WN has split its resources initially to pull two jobs requesting 2 cores each. When the first 2-core job ends, the slot is divided into 2 sub-slots, used subsequently by 2

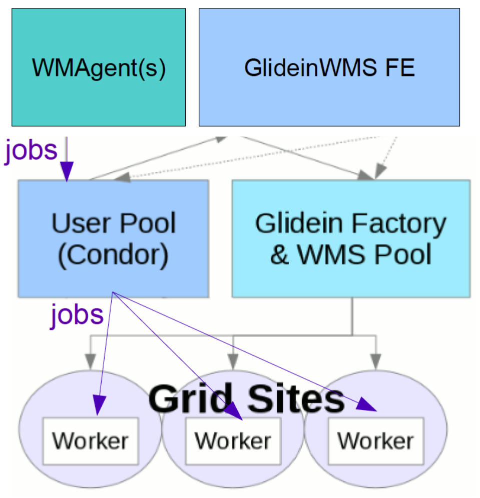

Figure 1. Schema of the current CMS workload management and submission infrastructure. See text for detailed description.

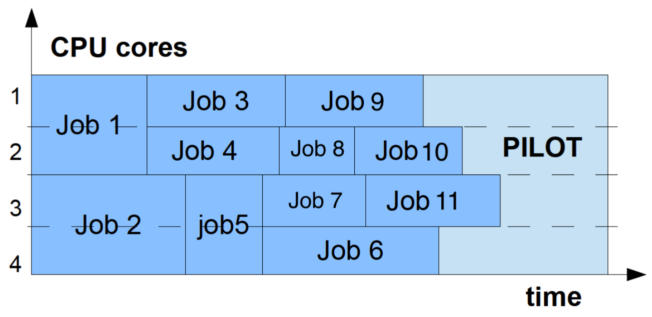

Figure 2. CMS multicore pilot for mixed single and multicore job scheduling strategy. See text for detailed description.

single-core jobs. When the second 2-core job ends, a third 2-core payload, with higher priority relative to single core jobs, is pulled and executed. After that point, in this example, the pilot resources are taken by single core payloads until the pilot lifetime ends.

The main advantage of a fully multicore pilot model is that CMS would gain total control on the scheduling priorities of single and multicore jobs, from production to analysis jobs, in line with the move of the submission infrastructure to the unified Global Pool described in [7], optimizing the use of the resources according to CMS needs. Also, not using single core pilots avoids them competing with multicore pilots for resources at the sites. As single core pilots are naturally easier to schedule, they may tend to exhaust the global share of CMS at the site. In this case, depending on the configuration of the local batch system, little or no multicore resource allocation may be achieved. Single core pilots compete as well with multicore pilots for jobs in the pilot-job negotiator, which can potentially reduce the overall occupancy of slots in multicore pilots. Finally, going to 100% multicore pilots also reduces slot negotiation times, as it helps in reducing the number of pilots the system needs to control, thus preventing potential scalability issues of the infrastructure.

Multicore pilots also present disadvantages. Inefficient use of the cores is observed at the end

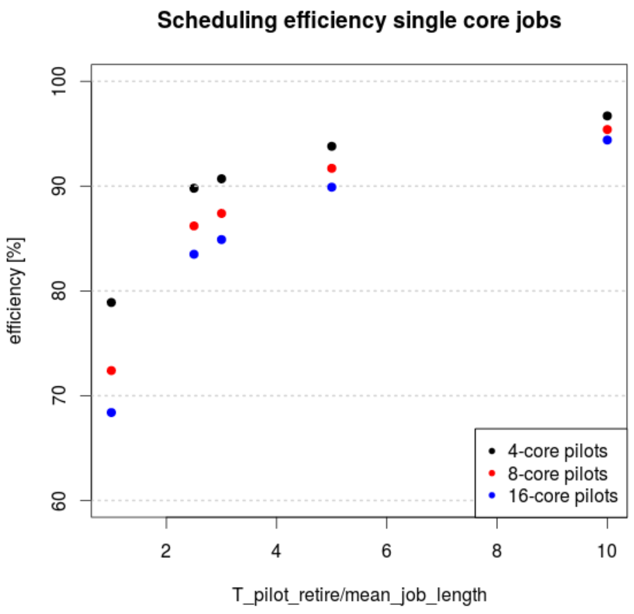

Figure 3. Simulation of efficiency in the use of cores by multicore pilots running single core payloads due to resource draining versus relative pilot to payload average running times, for different multicore pilot core counts. See text for detailed description.

of pilot lifetime, as pilots drain their resources before exiting the WNs. This draining period lasts as long as the longest running job takes to finish, the rest of cores waiting unused. This is a fundamental difference to single core pilots, which release their resources as soon as their only payload finishes. A simulation of this effect is shown in Fig. 3, with jobs of lengths according to a gaussian distribution, representing jobs of similar duration with certain spread such as production jobs, filling up slots of a multicore pilot until the pilot starts draining. Pilot running time is considered as multiples of the mean job running time. The scheduling inefficiency is then calculated as the fraction of resources unused over the whole pilot length. Multiple toy experiments are run and the average value is taken for different multicore pilot sizes (4, 8, 16) and as a function of the relative pilot to job length parameter (T_retire_time/mean _job_length). As observed, the higher the number of cores managed by the multicore pilot, the higher the draining inefficiency, and as expected, the longer the job duration compared to the pilot retire time, the larger the draining inefficiency. As a reference working point, 8-core pilots running single core jobs would be 95% efficient or higher when running on average 10 times or longer than the average payload.

Another disadvantage of allocating remote resources by multicore pilots is that the ramp up of multicore slots at a site after an idle period would be much slower than in the case of single core slots. The reason is that sites create multicore slots by draining their WNs of single core running jobs. Typically, a site would protect its farm from excessive draining (no more than a few percent of the total resources at any given time) to avoid low average farm occupancy, as they are in most cases responsible for the efficient use of the resources with respect to their funding agencies. An example is shown in Fig. 4, which presents the total available cores for multicore jobs for a period of a week at PIC, the Spanish Tier-1, for each of their users, CMS and ATLAS (shown as T1 plus T2). After a period with no requests from either experiment, multicore slots are returned to the single core pool, which is shown as a drop in the overall

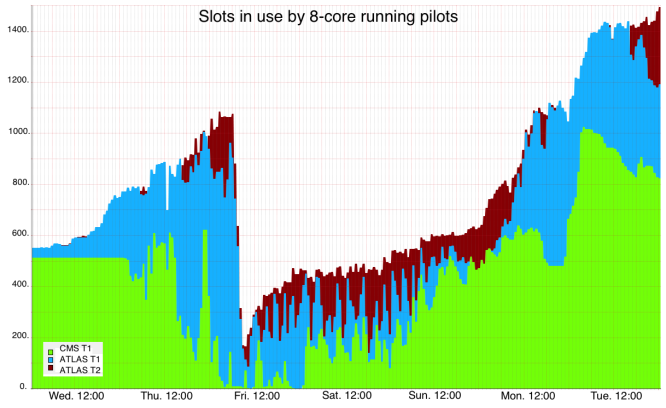

Figure 4. Total available cores for 8-core jobs for a period of a week at PIC site for CMS T1(green), ATLAS T1 (blue) and ATLAS T2 (brown) jobs. Note the slow ramp up of resources caused by the controlled draining of the WNs.

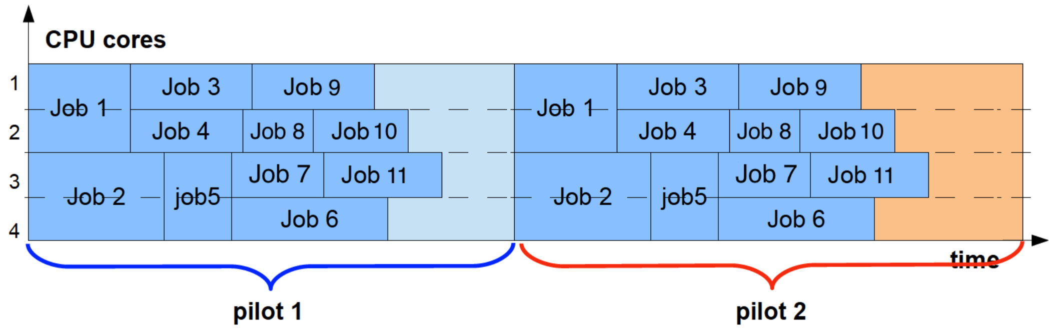

Figure 5. Pilot renewal as a source for defragmented resources. See text for detailed description.

available cores. Then, new requests reach the site again, however the controlled draining of WNs to create 8-core slots takes of the order of 4 days to reach a total of 1000 slots, comparable to CMS pledged cores at PIC.

A successful model based on multicore pilots must be able to provide enough resources for multicore payloads, preventing them from being used exclusively by single core jobs. As the example shown in Fig. 2 illustrates, single core payloads, if present in sufficient numbers and with higher priority relative to multicore jobs, would tend to exhaust the available resources of an individual pilot (*fragmentation*). In the CMS model the problem of fragmentation is related to the turn-over rate of finite-life pilots, as new pilots start with non-fragmented resources, see Fig. 5. A model with a sufficient pilot renewal rate thus satisfies the condition of being able to manage single core and multicore jobs simultaneously. The system however needs to be tuned on a set of parameters (job prioritization, jobs and pilots running times, single and multicore payload mixture, etc), so that forced pilot internal defragmentation, even if available as a standard HTCondor tool (called defragmentation daemon)[6], should not be required.

Many of the computing sites supporting CMS, and in particular most of the T1s, share their resources with other experiments, which may also be interested in running multicore jobs on the same clusters. CMS multicore job scheduling model should not impose requirements on

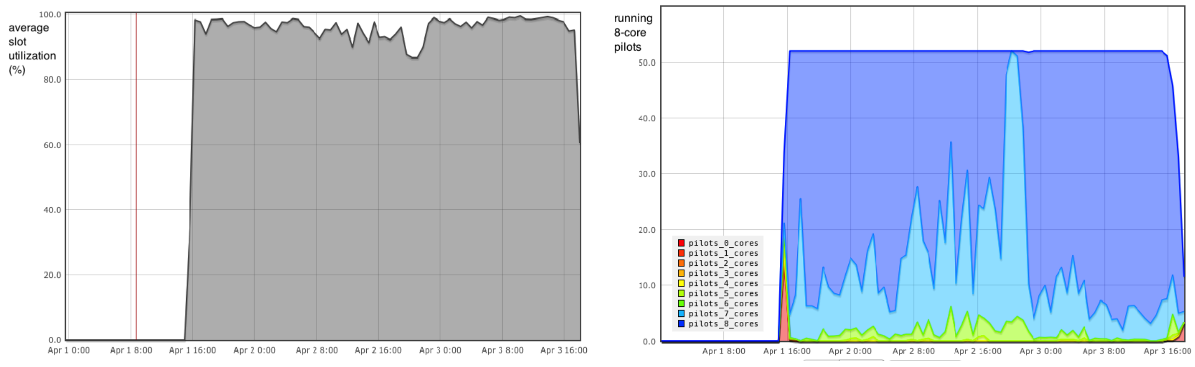

Figure 6. Multicore (8) pilots running single core payloads at PIC over a 3 day period: average core occupancy as a percentage of the cores being used to the total of available cores (left) and pilot distribution according to the number of occupied internal single-core job slots (right), from empty (red) to fully loaded pilots (dark blue).

the sites which would result in additional inefficiency and complexity in resources configuration. A WLCG task force[9], including representatives from sites and experiments, has been setup with the aim of exploring best practices for the deployment and most efficient use of the shared resources. One of the recommendations from the work of this group [10] is the use of a common core-count for multicore requests submitted to the same cluster. This eliminates the need for successive WN draining stages and allows to recycle empty multicore slots, preventing them from being fragmented and also permitting the use of resources on a fair-share policy. The example of PIC shown in Fig. 4 is the result of the agreement between ATLAS and CMS to use 8-core pilots as a common general default. In practice, many sites have adapted their batch systems on the basis of slot reutilization thanks to the use of a common core count, see for example reference [11].

3. Model and infrastructure tests

The CMS job scheduling and resource allocation infrastructure has been incrementally adapted in order to support the strategy described in the previous section. It has been repeatedly tested, as T1s joined the pool of multicore-capable sites, in order to verify the integrated scheduling of both multicore and single-core jobs, to study the fragmentation of resources and to identify any inefficient use of resources deriving from the scheduling.

Multicore pilots internal allocation efficiency for single core jobs has been under careful observation in order to measure the draining effect simulated and presented in Fig. 3. Figure 6 shows multicore pilots running single-core payloads at PIC for a particular period of activity of several days with sufficient job pressure to saturate the allocated cores. Pilot running times were about 40h long. Jobs running as internal payloads to the pilot, while presenting a wide and irregular running time distribution, averaged at a few hours value. As a result, pilot internal inefficiencies are measured to be small, with most pilots running fully loaded with 8 single core jobs and an average overall utilization of the available cores of 94% for this period.

Global performance results have been obtained by pushing the system to reach to the scale of the CMS target for 2015, i.e. 50% of the T1 resources, which should be available to be used by multicore jobs. Prompt data reconstruction multicore jobs were injected from the T0 infrastructure targetting 18.000 CPU cores at all the CMS T1s (at KIT, PIC, CCIN2P3, CNAF, JINR, RAL and FNAL sites). The test consisted of running 4-core jobs inside 8-core pilots, under heavy workload pressure from single core jobs as well, for a period of about a week in March

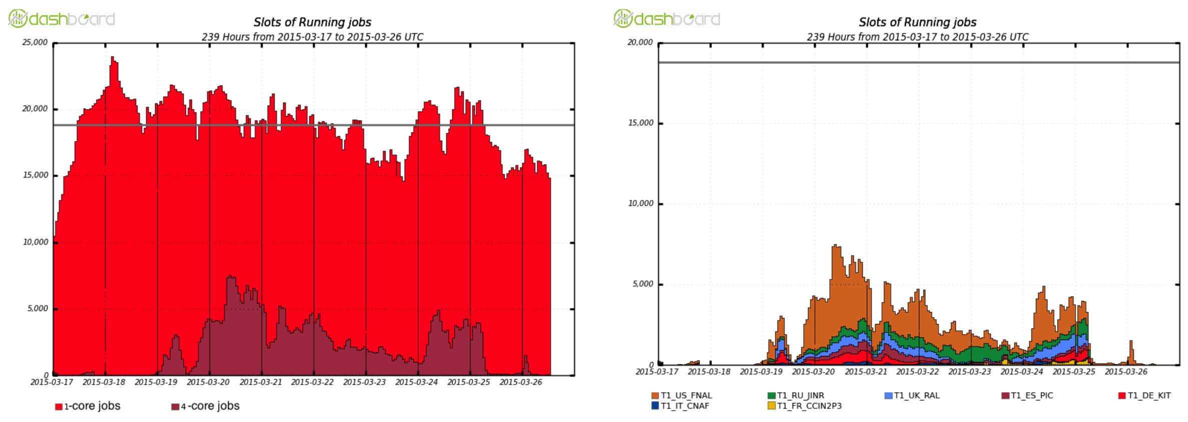

Figure 7. Results for the scale test of multicore prompt reconstruction jobs submission: job slots occupied at CMS T1 sites by single and 4-core jobs (left) and distribution of running multicore jobs across CMS T1s (right). A horizontal gray line appears in both plots as a reference of the size of the global pool of resources pledged to CMS at the T1s, with about 18.000 cores in total.

2015.

Results are presented in Fig. 7. The first and most obvious, yet important, observation is that multicore job submission works, as the CMS infrastructure successfully submitted and ran multicore jobs at all CMS T1s. The results are promising, close to the target overall, with peaks of 8.000 cores being taken for multicore jobs and multicore payloads being continuously processed, which implies a reasonable defragmentation rate, even with a very preliminary tuning. In fact, it was later noticed that the WMAgents handling these workflows were not yet managing mixed single core and multicore loads with correct job prioritization, assigning equal to priority to both tasks. As multicore payload priority should in general be higher than that of single core payloads, these results should not be regarded as the definitive performance of CMS model, but as a step in the right direction, which can only improve with proper task prioritization.

It should also be mentioned that, even if the test was, overall, close to the target scale, the results revealed noticeable variations from site to site. Work is ongoing at the time of writing this report in order to study and improve pilot allocation to each of the sites to reach targets at each site independently. This involves ensuring that GlideinWMS pilot factories produce enough multicore pilot pressure for each site, and also that the local resource managers (T1s batch systems), can handle this load appropriately.

4. Conclusions and outlook for future developments

As described in the previous sections, CMS will employ multithreaded applications for LHC Run 2, starting with the use of multicore jobs for prompt data reconstruction in 2015. This tasks will require, in addition to T0 CPUs, about 50% of the pledged CPU job slots at the T1s. CMS job management and submission system will integrate scheduling of single and multicore payloads into multicore partitionable pilots. This tool is ready and the principle has been successfully implemented and tested. Performance results show that scheduling inefficiency can be minimized to a negligible level with a reasonable tuning of the system parameters. Finally, the target for multicore resources at the T1s for 2015 has been already basically achieved, so CMS multicore job submission infrastructure is essentially ready and looking forward to the restart of the data taking.

Activities and milestones for the coming months include the optimization of site by site scale test results, the complete deployment of improved job and pilot performance monitoring tools and the continuation of tests and close performance monitoring for further optimization of mixed workloads scheduling.

Acknowledgments
The Port d'Informació Científica (PIC) is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universitat Autonoma de Barcelona. This work was supported partly by grants FPA2010-21816-C02-00 from the Ministry of Science and Innovation and FPA2013-48082-C2-1/2-R from the Ministry of Economy and Competitiveness, Spain. Additional support was provided by the EU 7th Framework Programme INFRA-2010-1.2.1: Distributed computing infrastructure Contract Number RI-261323 (EGI-InSPIRE).

References

[1] Jones C *et al* 2015, Using the CMS Threaded Application in a Production Environment, these proceedings and `https://indico.cern.ch/event/304944/session/2/contribution/120`
[2] Hufnagel D *et al* 2015, The CMS Tier-0 goes Cloud and Grid for LHC Run 2, these proceedings and `https://indico.cern.ch/event/304944/session/5/contribution/119`
[3] Fajardo E *et al* 2012, A new era for central processing and production in CMS, J. Phys.: Conf. Ser. 396 042018 doi:10.1088/1742-6596/396/4/042018
[4] Sfiligoi I *et al* 2009, The pilot way to grid resources using glideinWMS, `http://dx.doi.org/10.1109/CSIE.2009.950`
[5] GlideinWMS Homepage: `http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/`
[6] HTCondor Homepage: `http://research.cs.wisc.edu/htcondor/`
[7] Gutsche O *et al* 2015, Using the glideinWMS System as a Common Resource Provisioning Layer in CMS, these proceedings and `https://indico.cern.ch/event/304944/session/4/contribution/289`
[8] Perez-Calero Yzquierdo A *et al* 2014, CMS multicore scheduling strategy, J. Phys.: Conf. Ser. 513 032074 doi:10.1088/1742-6596/513/3/032074
[9] WLCG Multicore Deployment Task Force 2015, `https://twiki.cern.ch/twiki/bin/view/LCG/DeployMultiCore`
[10] Forti A, Perez-Calero Yzquierdo A *et al* 2015, Multicore job scheduling in the Worldwide LHC Computing Grid, these proceedings and `https://indico.cern.ch/event/304944/session/4/contribution/333`
[11] Templon J *et al* 2015, Scheduling multicore workload on shared multipurpose clusters, these proceedings and `https://indico.cern.ch/event/304944/session/6/contribution/281`