Exploiting multicore compute resources in the CMS experiment

Exploiting multicore compute resources in the CMS experiment

J E Ramírez[1], A Pérez-Calero Yzquierdo[2,3], J M Hernández[3] on behalf of the CMS Collaboration.

[1] University of Puerto Rico Mayagüez, Mayagüez, Puerto Rico, USA
[2] Port d'Informatió Científica (PIC), Universitat Autónoma de Barcelona, Bellaterra (Barcelona), Spain
[3] Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas, CIEMAT, Madrid, Spain

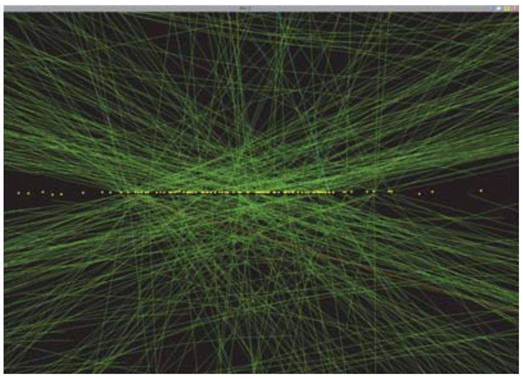
E-mail: `juaneduardo.ramirez@upr.edu`

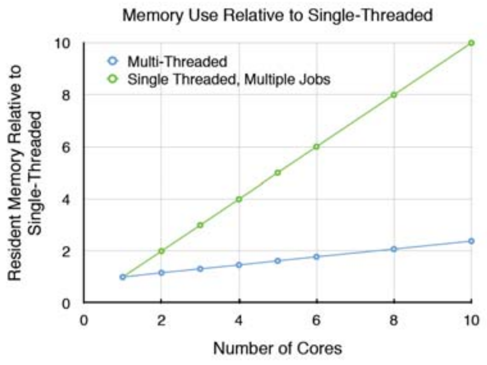
Abstract. CMS has developed a strategy to efficiently exploit the multicore architecture of the compute resources accessible to the experiment. A coherent use of the multiple cores available in a compute node yields substantial gains in terms of resource utilization. The implemented approach makes use of the multithreading support of the event processing framework and the multicore scheduling capabilities of the resource provisioning system. Multicore slots are acquired and provisioned by means of multicore pilot agents which internally schedule and execute single and multicore payloads. Multicore scheduling and multithreaded processing are currently used in production for online event selection and prompt data reconstruction. More workflows are being adapted to run in multicore mode. This paper presents a review of the experience gained in the deployment and operation of the multicore scheduling and processing system, the current status and future plans.

1. Introduction

The experimental collaborations taking data at the Large Hadron Collider (LHC) have begun transitioning to multi-core aware application frameworks and scheduling systems capable of executing multi-threaded applications across multiple cores[1, 2, 3]. This transition has been driven by an ever increasing volume of complex data resulting from the higher energy and luminosity delivered by the LHC compared to Run 1. The increase of intensity results in a higher number of concurrent collisions per event (pile-up). Figure1(a) shows a high pile-up event from CMS data with 78 reconstructed vertices. These busy events lead to a higher per-event processing time and memory usage. On the hardware side, the evolution of multicore technology has been driven by the power limitations on increasing the density of transistors on single core processors to enhance its performance, and as a result, the need to look for alternative solutions to increase CPU performance.

Utilizing multicore CPU capability requires software changes to transition from sequential programming to parallel processing. The multicore applications developed to adapt the code to this new architecture design offer several advantages. In a threaded process, memory is shared between threads and this reduces the memory consumption per core[4], thereby avoiding running into memory limitations as depicted in Fig.1(b). Another advantage concerns the experiment's


(a) High pileup CMS event with 78 reconstructed vertices



(b) Memory use of a multi-threaded application compared to n simultaneous single-threaded tasks

Figure 1. Drivers motivating use of multicore

Workload Management System (WMS). The number of jobs to be handled by the system is reduced and the output files produced are of larger sizes, which then requires less managing and merging operations. The software evolution to utilize multicore CPUs scattered across the Grid has required the modification of the applications themselves as well as the development of new resource allocation, scheduling tools and procedures.

For the LHC Run 2 data taking period that began in early 2015, the initial CMS priority in the deployment of multicore processing and scheduling was the provision of sufficient computing power to perform multithreaded prompt data reconstruction tasks. In 2015 it was expected to use 100% of the Tier-0 and 50% of the Tier-1's available CPUs. Thus, for the first phase of multicore deployment, the focus has been on T0 and T1s sites (see [5, 6] for a description of CMS tiered computing model and mesh configuration). Multicore resources are also starting to be deployed at CMS Tier-2 sites, with the objective of switching simulation and digitization tasks to multithreaded algorithms as well during 2016. Single core and multicore jobs will coexist during Run 2 and therefore, a strategy for scheduling both types of job is mandatory. Eventually, the use of multicore pilots by CMS will expand to manage majority of the resources at the supporting computing sites.

2. CMS model for multicore resource provisioning

The CMS workload management system (WMS) employs software agents (WMAgents[7]) to manage centralized workflows populating job queues, assigning job priorities, handling errors, job retries, merging output files and log collection. The allocation of execution nodes and the scheduling of jobs is done through GlideinWMS[8, 9, 10]. Resources are allocated at the grid sites by means of pilot submission as a result of job pressure. Jobs are matched to resources by managing a transient pool of computing resources controlled by *pilot jobs*, whose function is to schedule user jobs.

The model for the integrated scheduling of CMS jobs with different core count requests is the use of multicore pilots capable of internal partitioning of the resources into dynamic slots. A single pilot will run multiple payloads, as shown in Fig. 2 for the example of a 4-core pilot. Two multicore jobs are initially run by creating two slots of two cores each, followed by two additional single core jobs by further fragmenting one of the slots after the first 2-core job is finished, This continues until the pilots approaches the end of its defined lifetime. As a result, running multicore partitionable pilots on CPU resources allows managing all CMS workflows with a simplified model.

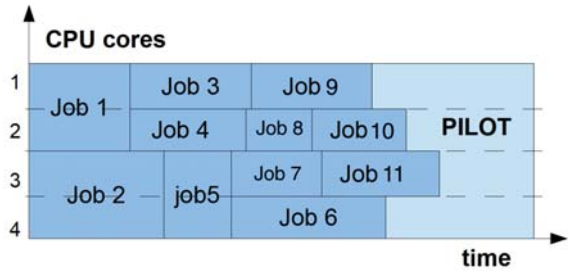


Figure 2. Schematic of a multicore partitionable pilot

Additional advantages of this model are: a total control of job scheduling priorities by CMS, as opposed to the scheduling at site level; avoiding competition of single and multicore pilots for resources at sites and for matching jobs once running, which simplifies the scheduling optimization; a reduction in the number of pilots needed to run the global pool of resources, which contributes to the scalability of the system.

The main disadvantages are the inherent inefficiency coming from the draining of retiring pilots and the potentially slow ramp up of allocated resources (in the case of irregular pilot submission patterns) in multi-VO sites with policies to protect their compute farms from excessive resource draining. A previous study[11] observed however negligible inefficiencies when pilot lifetimes are about ten times or longer than the average job length.

A potential limitation in the model is the inability to pull new multicore jobs once the pilot has internally fragmented the resource. However, the pool of pilots is continuously renewed (since pilots have a finite lifetime). Therefore providing fresh non-fragmented pilots (see Fig. 3). This avoids the costly need of induced defragmentation inside running pilots, which however could still be applied in case of urgency in executing a particular multi-threaded workflow.

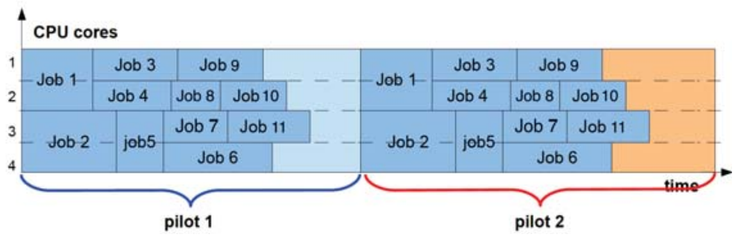


Figure 3. Schema of CMS model for multicore resource provisioning

The dynamic behaviour of the system with a dependence on a number of parameters (e.g. pilot and average job lifetimes, pilot and payload core counts, etc.) implies that its tuning is essential in order to achieve optimal performance.

3. Results for 2015

CMS has successfully deployed and tested its multicore architecture at the T0 and T1 computing sites during 2015. We present here four main results:

- **Deployment at T1 sites.** The multicore resource provisioning model has been incrementally deployed at T1s as they joined the pool of multicore-capable sites and tuned their batch systems to optimal performance when handling multicore requests. Figure 4(a) shows the monthly average number of cores allocated to running multicore pilots at all CMS T1s in the second half of 2015.

- **Mixed scheduling of multicore and single core jobs within multicore pilots**. In order to assess the multicore pilot model readiness to schedule a mixed workload consisting of multicore and single core jobs, prompt reconstruction 4-core jobs were tested at large scale at all CMS Tier-1 sites prior to the start of 2015 data taking, as shown in Fig.4(b). The target, set at 50% of the pledged T1 CPU cores, was essentially achieved during the moments of highest multicore job pressure. However, the small amount of data collected during 2015 made the urgency to use of T1 CPUs for prompt reconstruction lower than expected, so finally prompt data reconstruction was executed mainly at the T0.
- **Multicore Pilot job scheduling efficiency**. To measure the efficiency of multicore pilot internal payload scheduling, single core jobs were submitted with enough pressure to saturate the allocated resources of about 1000 CPU cores. Multicore pilots were typically 40 hours long, while single core jobs run with a variety of job lengths between 1 to 2 hours. The test, run over the lifetime of the pilots, yielded negligible scheduling inefficiencies, as shown in Fig.4(c).
- **Execution of multicore prompt data reconstruction jobs at the T0 during 2015**. As shown in Fig.4(d), a variety of core counts were employed for jobs running the T0 CPUs. Prompt data reconstruction for proton-proton collisions was typically executed as 4-core jobs, while other multicore job sizes were also employed, in particular for the end-of-year heavy ion run, with 8-core jobs.

4. Future outlook

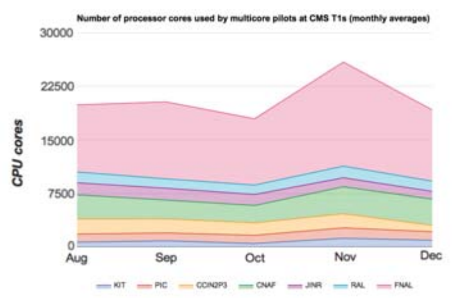
In 2016 it is expected that CPU resources at the T1 sites will be regularly used to run multithreaded tasks, such as data reconstruction jobs and also simulated data processing, as the Monte-Carlo simulation and digitization steps have been ported to run efficiently in multithreaded mode. The commissioning work done during 2015 with T1 sites ensures that both the sites and the multicore scheduling system are ready to reach this goal. In 2016 the pool of available multicore resources will be increased to include the major T2 sites. Finally, while CMS tools for multicore job management are ready, a complete deployment of monitoring tools is needed in order to continue optimizing the scheduling algorithms.

Acknowledgments

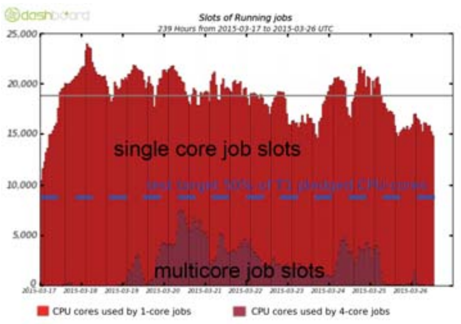
This work was supported in part by the U. S. National Science Foundation Award No. ACI-1440552, the Spain Ministery of Science and Innovation grant FPA2010-21816-C02-00, Spain Ministery of Economy and Competitiveness grant FPA2013-48082-C2-1/2-R and EU 7th Framework Programme INFRA-2010-1.2.1: Distributed computing infrastructure Contract Number RI-261323 (EGI-InSPIRE). The Port d'Informació Científica (PIC) is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universitat Autonoma de Barcelona.

References

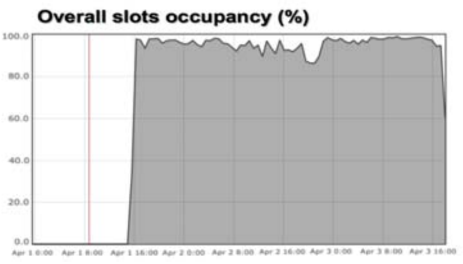
[1] Pérez-Calero Yzquierdo A, Hernández J, Holzman B, Majewski K, McCrea A and the CMS Collaboration 2014 *Journal of Physics: Conference Series* **513** 032074 URL `http://stacks.iop.org/1742-6596/513/i=3/a=032074`
[2] Hernández J M, Evans D and Foulkes S 2012 *Journal of Physics: Conference Series* **396** 032055 URL `http://stacks.iop.org/1742-6596/396/i=3/a=032055`



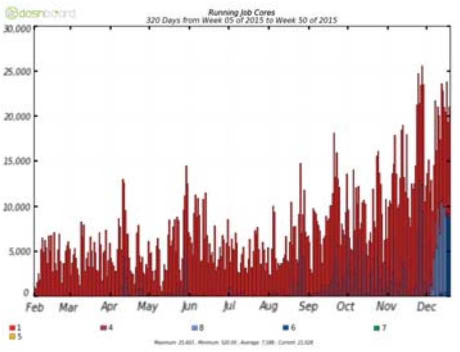
(a) Monthly average number of processor cores used by multicore pilots at T1s



(b) CPU cores in use by single and 4-core jobs running at T1s



(c) Percentage of overall slots occupancy as a measure of pilot internal scheduling performance



(d) Running jobs at the T0 as a function of time during 2015, colored by number of cores per running job

Figure 4. Results for 2015

[3] Forti A, Yzquierdo A P C, Hartmann T, Alef M, Lahiff A, Templon J, Pra S D, Gila M, Skipsey S, Acosta-Silva C, Filipcic A, Walker R, Walker C J, Traynor D and Gadrat S 2015 *Journal of Physics: Conference Series* **664** 062016 URL `http://stacks.iop.org/1742-6596/664/i=6/a=062016`

[4] Jones C D, Contreras L, Gartung P, Hufnagel D and Sexton-Kennedy L 2015 *Journal of Physics: Conference Series* **664** 072026 URL `http://stacks.iop.org/1742-6596/664/i=7/a=072026`

[5] Bayatyan G L, Della Negra M, Fo, Herv A and Petrilli A (CMS Collaboration) 2005 *CMS computing: Technical Design Report* Technical Design Report CMS (Geneva: CERN) submitted on 31 May 2005 URL `http://cds.cern.ch/record/838359`

[6] Ayllon A A, Salichos M, Simon M K and Keeble O 2014 *Journal of Physics: Conference Series* **513** 032081 URL `http://stacks.iop.org/1742-6596/513/i=3/a=032081`

[7] Fajardo E, Gutsche O, Foulkes S, Linacre J, Spinoso V, Lahiff A, Gomez-Ceballos G, Klute M and Mohapatra A 2012 *Journal of Physics: Conference Series* **396** 042018 URL `http://stacks.iop.org/1742-6596/396/i=4/a=042018`

[8] Sfiligoi I, Bradley D, Holzman B, Mhashilkar P, Padhi S and Wurthwein F 2009 *Computer Science and Information Engineering, 2009 WRI World Congress on* vol 2 pp 428–432

[9] GlideinWMS Homepage URL `http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/`

[10] HTCondor Homepage URL `http://research.cs.wisc.edu/htcondor/`

[11] Pérez-Calero Yzquierdo A, Hernández J M, Khan F A, Letts J, Majewski K, Rodrigues A M, McCrea A and Vaandering E 2015 *Journal of Physics: Conference Series* **664** 062046 URL `http://stacks.iop.org/1742-6596/664/i=6/a=062046`