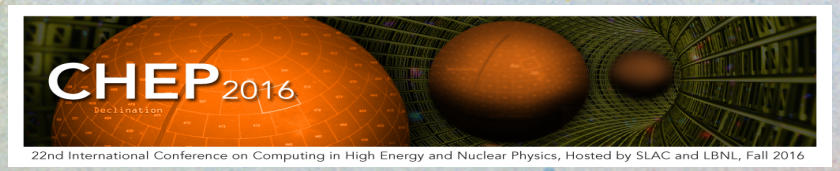


CMS readiness for multi-core workload scheduling

Antonio Pérez-Calero Yzquierdo,

on behalf of the CMS Collaboration, Computing and Offline, Submission Infrastructure Group

CHEP 2016 San Francisco, USA
October 10th, 2016

Ciemat
Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

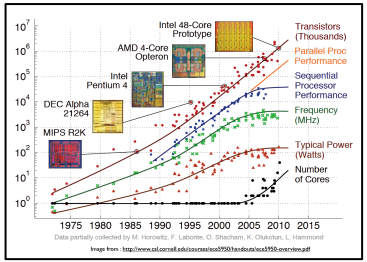
PIC
port d'informació científica

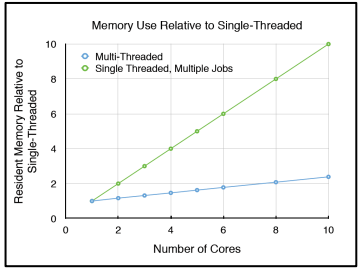
Outline

- Multicore jobs
- CMS global pool and multicore pilot model
- Multicore pilot deployment to CMS sites
- Results and performance
- Conclusions

Multicore jobs for LHC Run 2

- **Motivation** for multithreaded applications:
  - **Hardware evolution**: best exploitation of current multicore CPU architectures
  - **Evolution of LHC conditions**: increased data volumes and **event complexity**
    - memory per core
    - time per event
- **We need tools to handle multicore jobs to our CPU resources**
- **However, single core** and **multicore** jobs will coexist during Run 2
  - **Ability to scheduling both types of jobs is mandatory**
- CMS priority in 2015 was enabling multicore job allocation to Tier-1 resources, hence the **first phase of deployment focused on Tier-1s**
- Deployment to the majority of CMS pledged CPUs at **Tier-2s continued in Spring 2016**

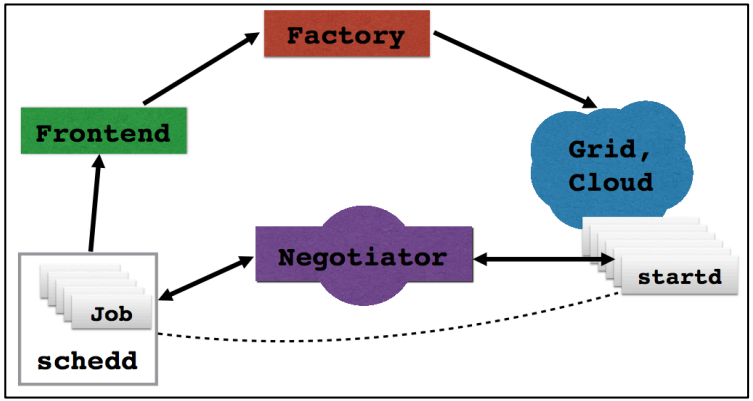


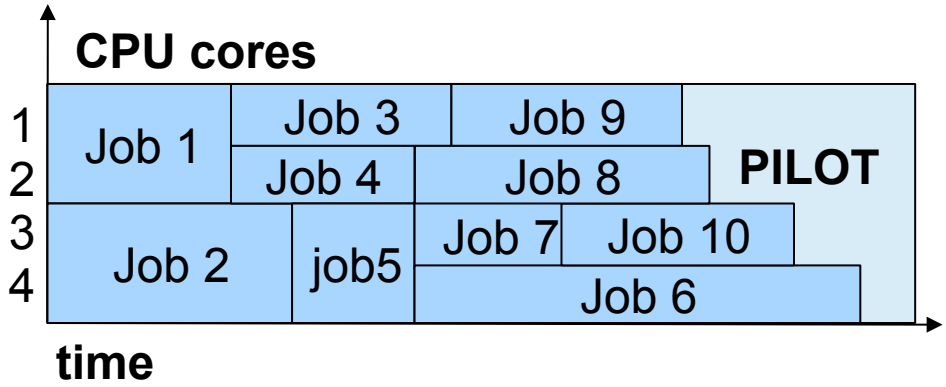


CMS Global pool

- **CMS Submission Infrastructure Global Pool** is a single **HTCondor** pool covering CPU resources accessible to CMS (pledged and opportunistic) and handling both centralized production workflows and analysis tasks
  - **GlideinWMS Frontend and factories** manage a transient pool of computing resources by means of **pilot job submission to execute nodes matching job resource requests**
  - **HTCondor Negotiator connects the submit nodes (schedds) to matching pilots running at execute nodes (startds)**

- CMS **main ingredient** for common allocation of single and multicore jobs: **adoption of multicore pilots** with internal **dynamic partitioning** of resources (HTCondor ***partitionable slot*** pilots)





See "Stability and scalability of the CMS Global Pool"

CMS Multicore pilot model

Advantages of the allocation of CMS computing resources with **all multicore partitionable pilots**:

- **Flexible system** which allows CMS control of **scheduling diverse jobs types** (single-core, multi-core, high-memory, etc) without intervention from the grid sites (e. g. no **specialized queues required**)
- Compared to using a combination of single and multi-core pilots, it **removes unwanted effects from single-core and multi-core pilot competition** (for resources at the sites and for matching to jobs once running)
- **Reduced number of pilots to manage the CMS global pool**, contributing to the scalability of the system

Disadvantages of multicore partitionable pilots:

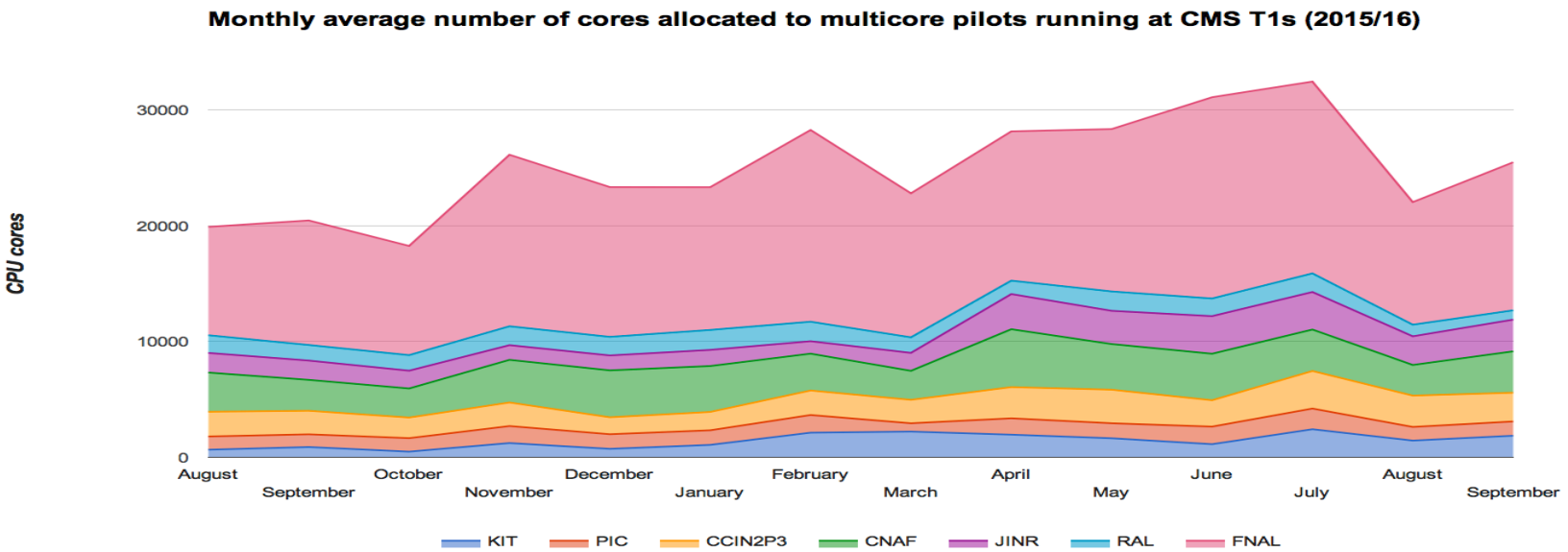
- **Inefficiencies** in the **internal scheduling** of jobs to pilots, which are **otherwise external to the VO but detected at the LRMS**
- More **complex monitoring and accounting** of allocated and used resources, compared to „one pilot, one payload“ models
- **Slower ramp-up** of remote resource allocation compared to single core pilots in multi-VO sites

Objectives:

- **Successful scheduling of multicore and single-core jobs**
- **High efficiency CPU usage**, minimizing any inefficiencies deriving from scheduling

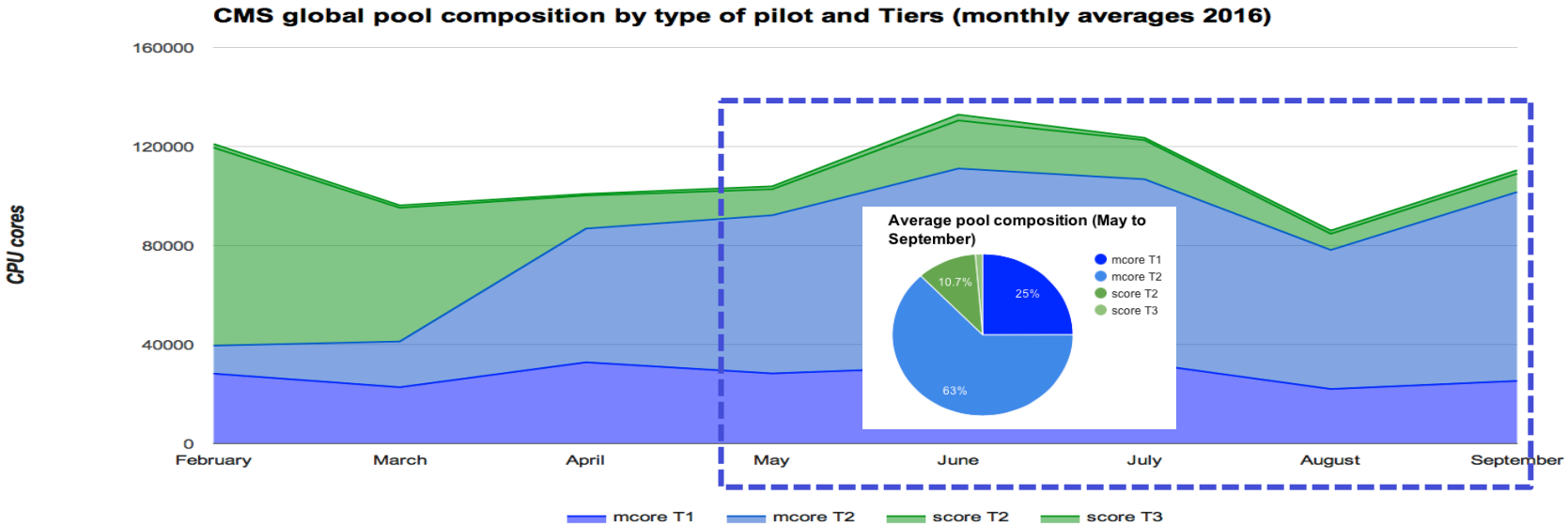
Multicore pilot deployment

- First deployment phase: submitting multicore pilots to all **Tier-1s since 2015**
  - **First tests 2014, fully deployed by late 2015, as submission of single-core pilots to Tier-1s was stopped**
  - **In 2016, regularly using 25,000 to 30,000 CPU cores (average), with peaks of up to 40,000 CPU cores**



Multicore pilot deployment

- Second deployment phase: **Tier-2s during Spring 2016**
  - Pool size average (May to September 2016) is of 112,000 CPU cores
  - Pool composition, once deployment to biggest Tier2s has finished, is close to **90% multicore** (***multicore global pool***)



Multicore pilot deployment

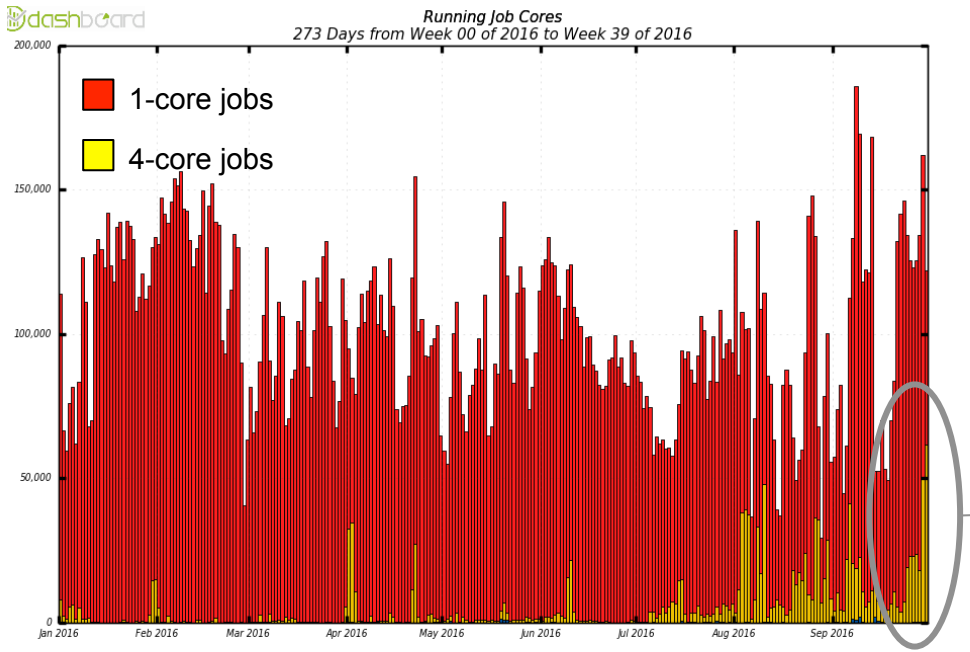
- Deployment to a **wide variety of CE and batch system technologies**: close collaboration with **CMS supporting sites admins** and **GlideinWMS Factory Operations** team (OSG), building from the experience gained in the Tier-1 deployment phase
  - In total **26 Tier-2s in a couple of months**
- Use a **standard pilot size for most resources**
  - **8 CPU cores, 16 GB memory (min), 48h walltime**

T1 sites	CE/batch_system
T1_DE_KIT	CREAM/GE
T1_ES_PIC	CREAM/PBS
T1_FR_CCIN2P3	CREAM/GE
T1_IT_CNAF	CREAM/LSF
T1_RU_JINR	CREAM/PBS
T1_UK_RAL	ARC/HTCondor
T1_US_FNAL	HTCondor/HTCondor

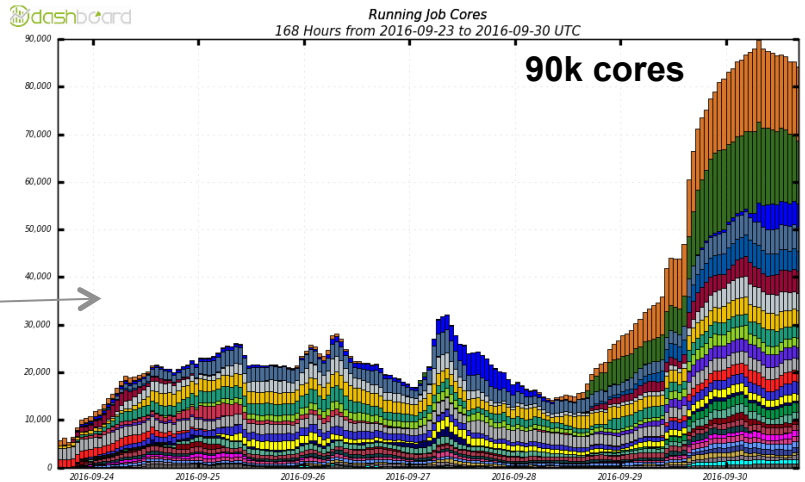
T2 sites	CE/batch_system
T2_BE_IIHE	CREAM/PBS
T2_CH_CSCS	ARC/SLURM
T2_DE_DESY	CREAM/PBS+ARC/PBS
T2_DE_RWTH	CREAM/PBS
T2_EE_Estonia	ARC/SLURM
T2_ES_CIEMAT	CREAM/PBS
T2_FR_CCIN2P3	CREAM/GE
T2_FR_GRIF_IRFU/LLR	ARC/HTCondor+CREAM/HTCondor
T2_FR_IPHC	CREAM/PBS
T2_IT_Bari	CREAM/HTCondor
T2_IT_Legnaro	CREAM/LSF
T2_IT_Pisa	CREAM/LSF
T2_PT_NCG_Lisbon	CREAM/GE
T2_UK_London_Brunel	ARC/HTCondor
T2_UK_London_IC	CREAM/GE
T2_UK_SGrid_RALPP	ARC/HTCondor
T2_US_Caltech	HTCondor/HTCondor
T2_US_Florida	HTCondor/PBS
T2_US_MIT	HTCondor/HTCondor
T2_US_Nebraska	HTCondor/HTCondor
T2_US_Purdue	HTCondor/PBS
T2_US_UCSD	HTCondor/HTCondor
T2_US_Vanderbilt	HTCondor/SLURM
T2_US_Wisconsin	HTCondor/HTCondor
T2_CH_CERN	CREAM/LSF+HTCondor/HTCondor

Running multithreaded jobs

- CMS is **successfully employing multithreaded jobs**, used regularly for certain tasks, such as **data and MC reconstruction** [1]
- The workload in the pool has been still **mostly single-core for the current Run II period** (MC generation and analysis jobs)
- **Multicore workload** has been steadily increasing along the current year



- Cores allocated to running jobs at T1s+T2s during 2016, as a function of the number of cores requested per job
- Regular mode of operation: **4-core jobs** being matched to the standard **8-core pilots**
- **Currently running 2016 data re-Reconstruction,** with contribution from **all multicore enables sites,** including **HLT**



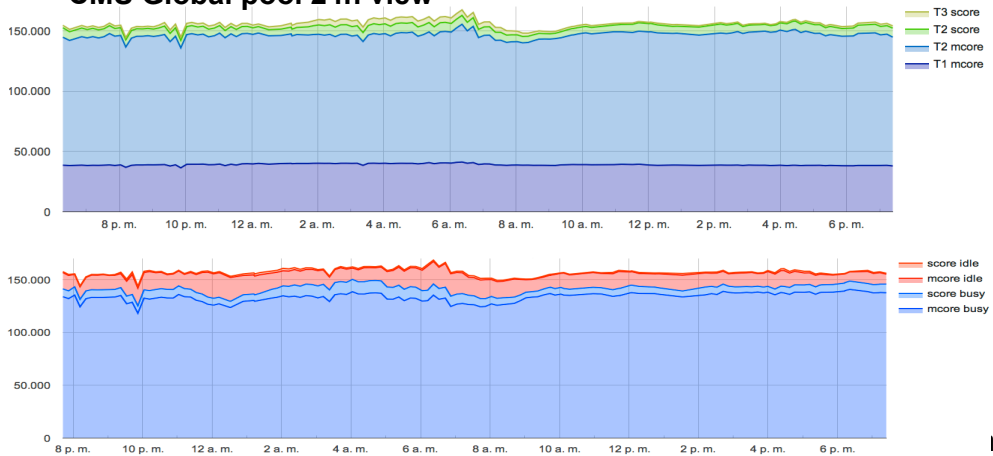
[1] See "CMS Event Processing Multi-core Efficiency Status"

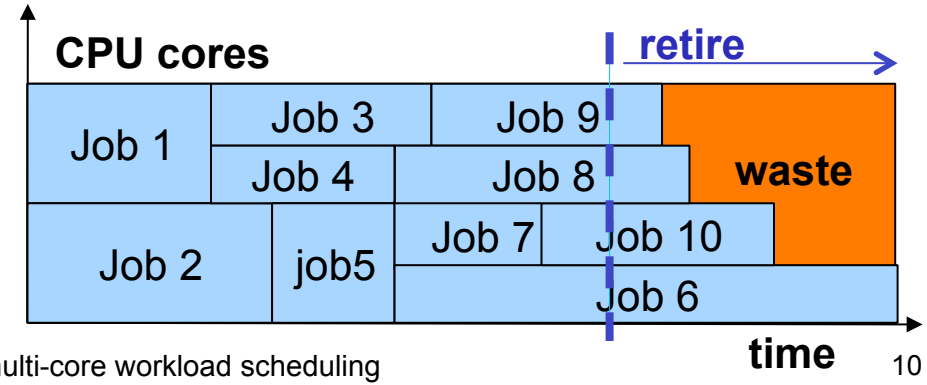
Scheduling performance

Main sources of scheduling inefficiency:

- **Retiring glideins**: pilots stop accepting new jobs close to the end of its lifetime, hence unused cores while the pilot is draining. **=> ~5% of the pool cores**
- **Memory constrained slots**: partitionable slot with available CPU cores but insufficient memory (typically<2GB). **Not in general an issue** (except for some particular periods at some sites, e.g. running HI data reconstruction jobs, wrong memory requests, etc).
- **Usable but unmatched slots**, with available CPU and memory, but not matching any jobs: caused by a variety of reasons such as **decreasing workload**, **restrictive site white-listing**, slow response from the job-submit components, slow negotiation cycle, etc. **=> ~5% of the pool cores**

CMS Global pool 24h view





Outlook into efficiency optimization

- CMS has **transitioned to using multi-core pilots** for resource allocation, however **still dominated by single-core jobs**: moving to employing multithreaded workflows primarily (centralized production but also analysis)

- **Wastage** in “usable but unclaimed cores” continuously being **reduced** by means of a **more efficient and faster job-to-pilot matchmaking**: incremental improvements on the CMS Global Pool (negotiator and schedds) performance

- **Improved workflow management and job-submit components**
  - Automation to reduce the need for human intervention (e.g. workflow resubmission) producing a **more continuous flow of requests**
  - Automation in **removing overloaded unresponsive submit-nodes** from the pool

Outlook into efficiency optimization

Improved management of pilot pressure

- **Resource-based response to workload pressure** from the glideinWMS frontend/factory (work in progress with glideinWMS team)

- **Cancelling of out-of-sync excess queued pilots** during irregular workload submission phases

More efficient matchmaking to running pilots

- **Depth-wise filling of the resources**, opposed to breadth-wise filling pattern employed by HTCondor negotiator by default: working in **close collaboration with the HTCondor developers team** (**prototype in global pool test-bed**)

- Resource requests expressions (**jdl-magic**) allowing for **resizable jobs**, which can be allocated all idle cores in the matching pilot (**prototype in global pool test-bed**)

Retiring pilots

- **Active pruning of the retiring pilot pool**: removal of pilots in long draining stage, profiting from automated job resubmission

Conclusions

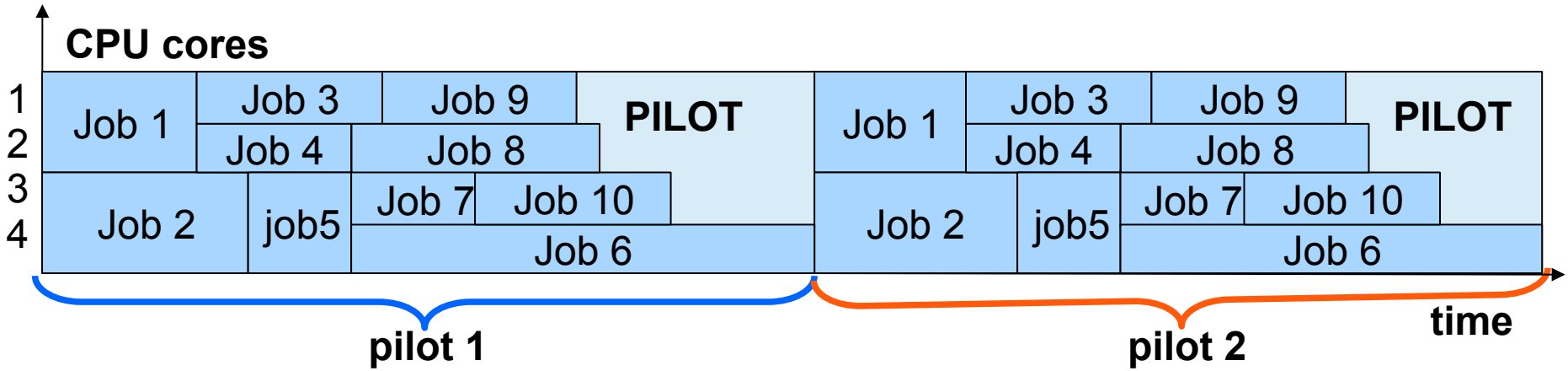
In summary:

- CMS experience with **multicore pilots** goes back to **2014**, then introduced to T1s, then **fully deployed in 2015, then on to biggest T2s in Spring 2016**.
- Using regularly ~100,000 cores in multicore pilots, **90% of the total CMS pool** (**multicore global pool**)
- **Multicore job submission working successfully. CMS switching to multicore jobs for centralized production (e.g. data and MC reconstruction).**
- **CPU allocation inefficiencies understood, being mitigated from multiple fronts.**

Extra slides

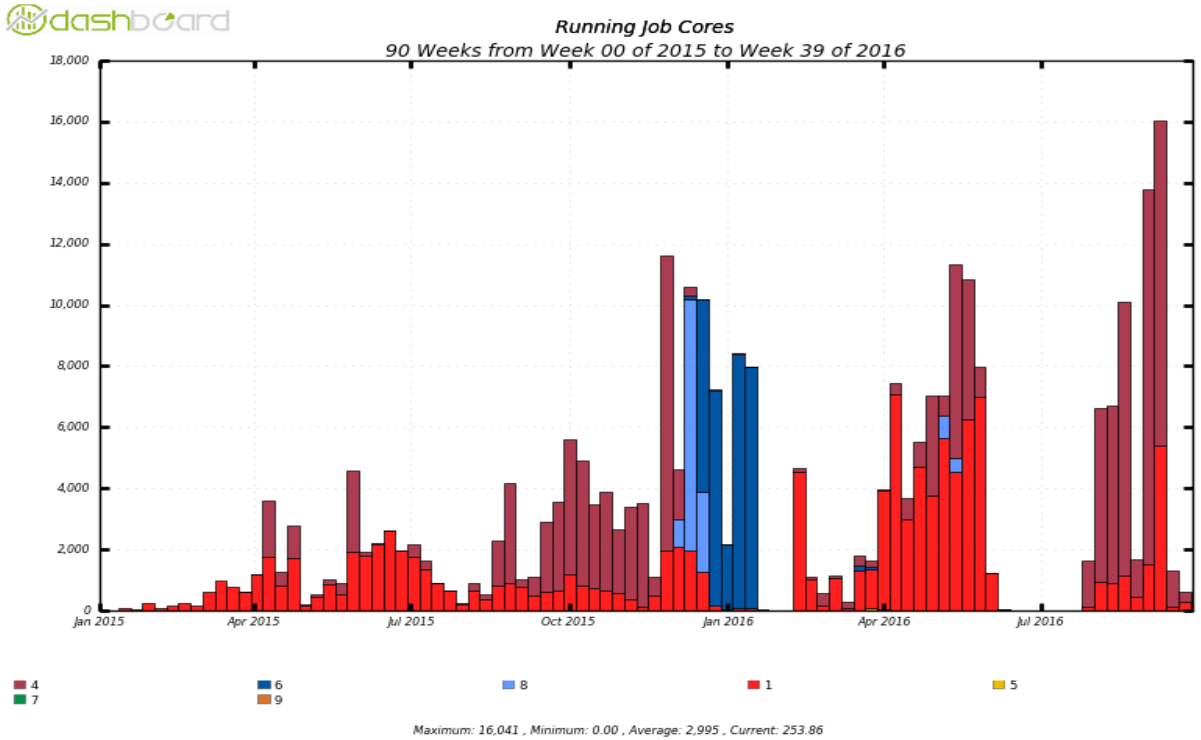
Pilot fragmentation

- **Fragmentation of the pilot internal resources** by single-core jobs limits the pilot ability to pull new multicore jobs.
  - However, renewal of finite-lifetime pilots provides fresh **non-fragmented pilots continuously**
  - **No forced defragmentation** of the pilots used
  - **Tuning of the system is crucial** as performance depends on multiple parameters (ratio of job and pilot running times, mixture of single and multicore jobs, etc)



Running multithreaded jobs (T0)

- Multithreaded jobs first introduced to **Tier-0** tasks, performing **prompt data reconstruction**, since the beginning of LHC Run II



- 4-core jobs for p-p data reconstruction during 2015 and 2016 data taking
- 6 and 8-core jobs employed on heavy-ion run data at the end of 2015

Related contributions at CHEP'16

- Stability and scalability of the CMS Global Pool: pushing HTCondor and glideinWMS to new limits
- CMS use of allocation based HPC resources
- Experience in using commercial clouds in CMS
- Software and Experience with Managing Workflows for the Computing Operation of the CMS Experiment
- CMS Event Processing Multi-core Efficiency Status
- Effective HTCondor-based monitoring system for CMS (poster)
- Connecting restricted, high-availability, or low-latency resources to a seamless Global Pool for CMS (poster)
- CMS Connect (poster)
- The CMS HLT as a dynamic cloud resource (poster)

References

- GlideinWMS: http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/index.html
- HTCondor: http://research.cs.wisc.edu/htcondor/
- The pilot way to Grid resources using glideinWMS, I. Sfiligoi et al. http://dx.doi.org/10.1109/CSIE.2009.950
- Evolution of CMS workload management towards multicore job support, A. Perez-Calero et al. http://dx.doi.org/10.1088/1742-6596/664/6/062046
- CMS multicore scheduling strategy, A. Perez-Calero et al. http://dx.doi.org/10.1088/1742-6596/513/3/032074

Abstract

In the present run of the LHC, CMS data reconstruction and simulation algorithms benefit greatly from being executed as multiple threads running on several processor cores. The complexity of the Run-2 events requires parallelization of the code in order to reduce the memory-per-core footprint constraining serial-execution programs, thus optimizing the exploitation of present multi-core processor architectures. The allocation of computing resources for multi-core tasks however becomes a complex problem in itself. The CMS workload submission infrastructure employs multi-slot partitionable pilots, built on HTCondor and GlideinWMS native features, to enable the scheduling of single and multi-core jobs simultaneously. This provides a solution for the scheduling problem in a uniform way across grid sites running a diversity of gateways to compute resources and batch system technologies. This contribution will present this strategy and the tools on which it has been implemented. The experience of managing multi-core resources at the Tier-0 and Tier-1 sites during 2015 will be described, along with the current phase of deployment to Tier-2 sites during 2016. The process of performance monitoring and optimization in order to achieve efficient and flexible use of the resources will also be described.