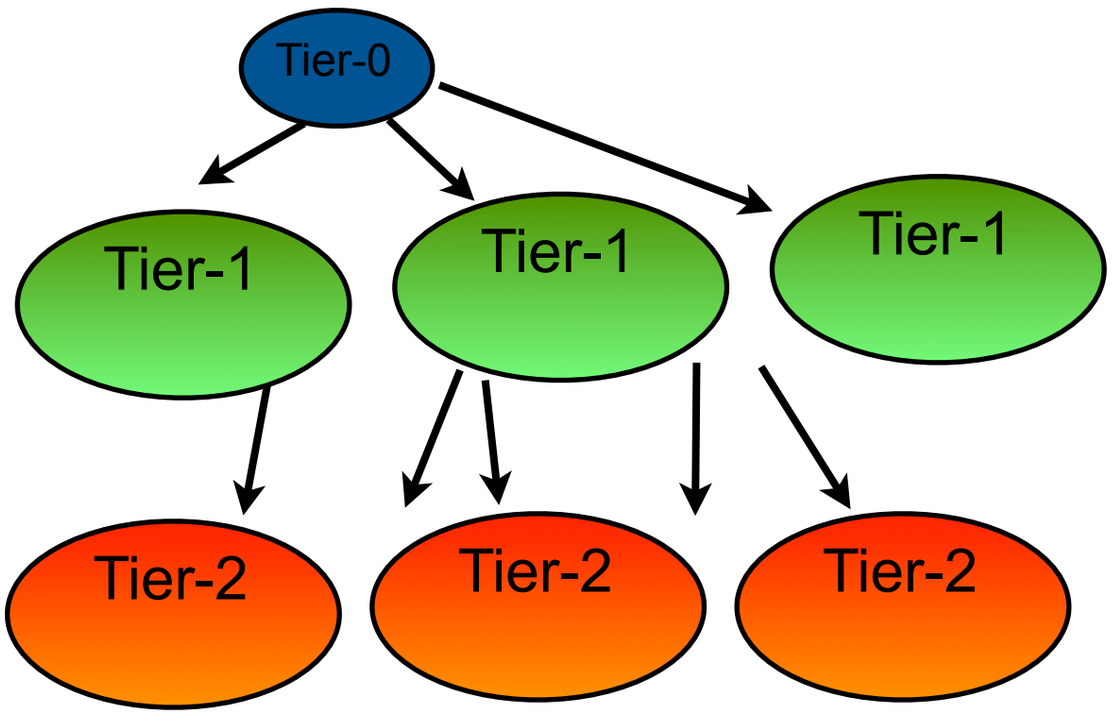
Opportunistic Resource Usage in CMS

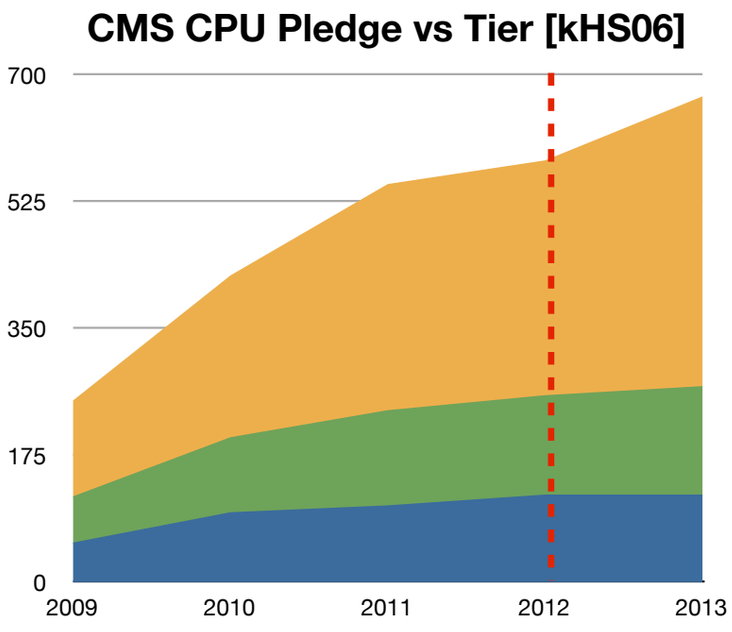
Peter Kreuzer
Dirk Hufnagel
(for the CMS Collaboration)

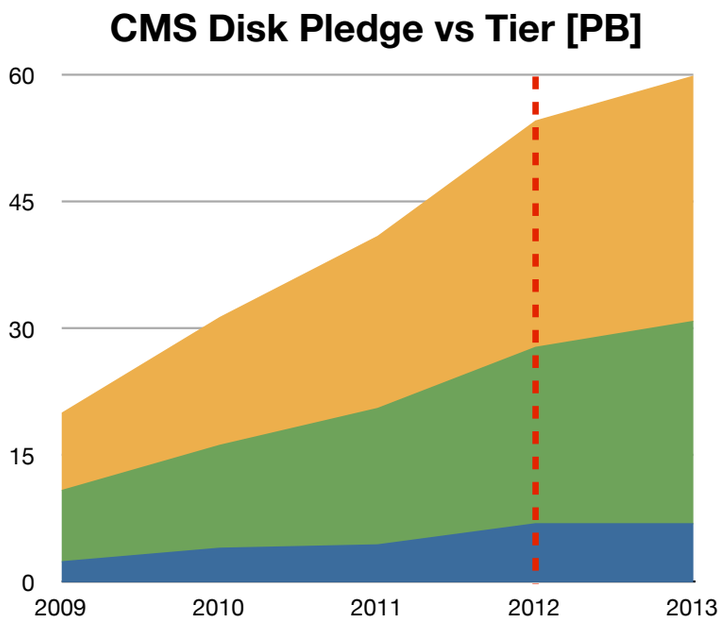
CHEP'2013 Amsterdam, October 14, 2013

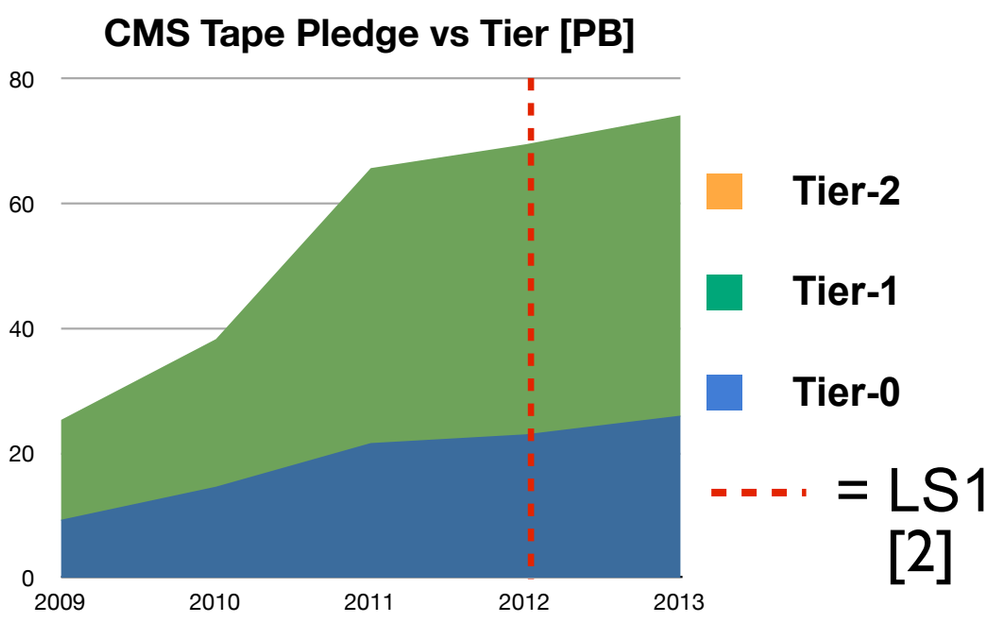
Resource Planning Model

- CMS [1] yearly resource planning model is in form relative resource increase at various Tier levels
- Relatively *flat resource budget* since 2009, with fluctuations mainly driven by the LHC program



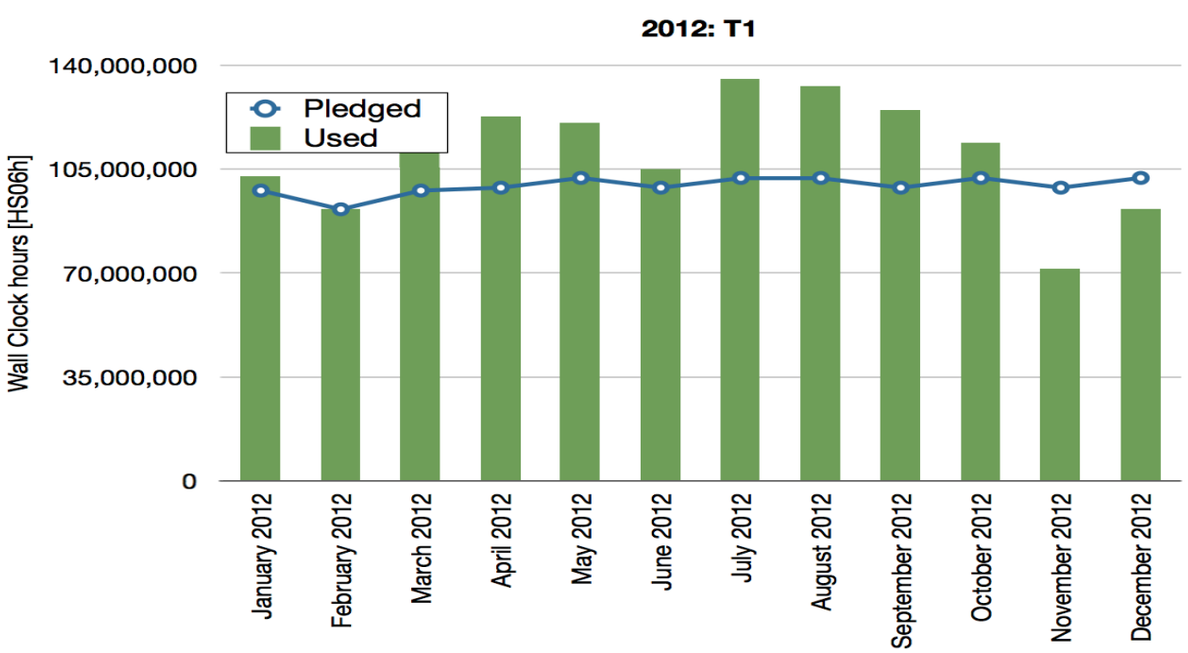


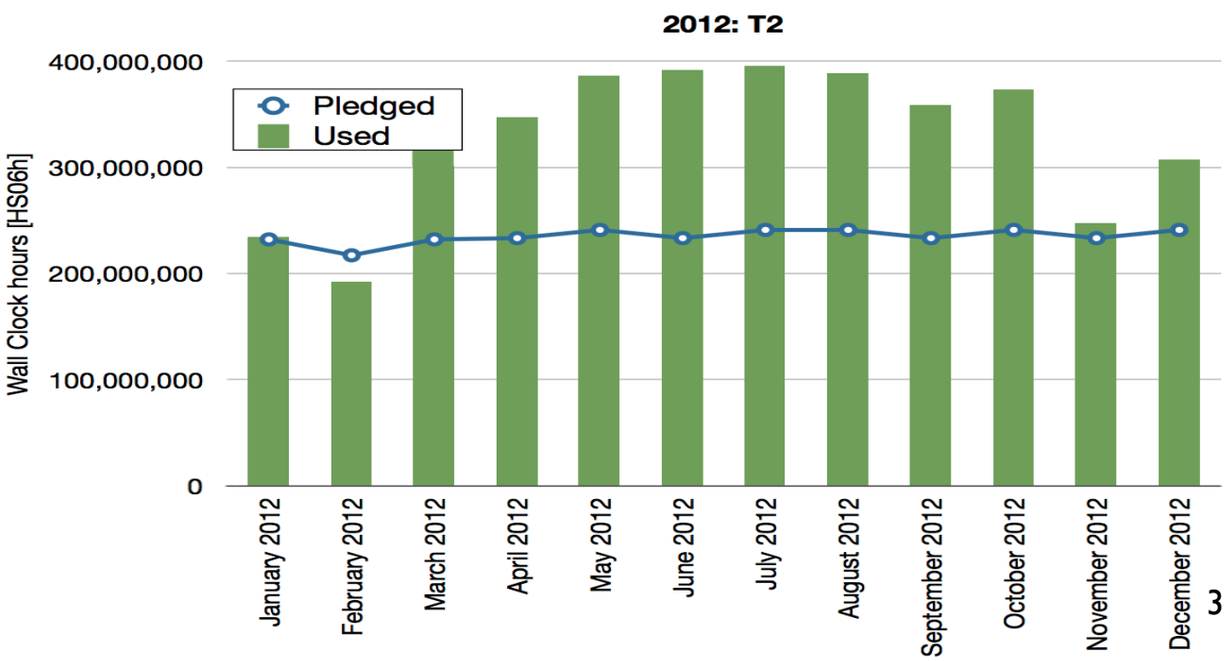




Resource Utilization

- The Yearly resource request and utilization are carefully scrutinized and endorsed (WLCG [3] Resource Review Board)
- The storage utilization typically matches with the planning, while the processing utilization (CPU) regularly goes beyond.
- So far extra CPUs were typically opportunistic resources that are already linked to CMS (non-pledged Tier-1/2 or Tier-3)
- Here we cover a more general concept of *Opportunistic Resources*, useful in case of deviations from flat resource budget





Opportunistic Resources: Motivation

- The LHC physics program invests more money into people than computing hardware
  - ➢ Need to optimize the productivity of human assets
- Our human assets are distributed worldwide with access to shared resources
  - ➢ Need to encourage an open architecture that allows dynamic integration of shared resources (i.e. that are working around missing services)
- One of our standard workflows – organized data processing – is an issue of short term peak computing demands
  - ➢ It makes little sense to solve this issue via "steady state hardware capacity" only

Goals and Challenges

of opportunistic resources

- Cover Resource needs at *10% level* averaged over 1 year (yet resources may double during short periods)
- Integrate opportunistic use cases *transparently*: setup embedded in the CMS Workflow Management System (*Glidein WMS [4]*)
- *Runtime challenge*: use resources for which we don't control the environment (CVMFS [5] , Parrot [6])
- *Submission challenge*: use resources for which we don't control the interface (BOSCO [7], OpenStack [8])
- *Networking challenge*: use resources for which we don't control the infrastructure (data input and stage out)
- *Operational challenge*: integrate resources for transparent and routine operations/monitoring

Software Challenges

depending on opportunistic resource

- How to get CMS jobs into opportunistic resources ?
- Type of opportunistic resource:
  - Use-case 1: *non-CMS WLCG resource*
  - Use-case 2: *non-WLCG resource*
  - Use-case 3: *cloud (EC2 [9]) resource*
- Software challenge:
  - *Submission challenge* (use-case II, III most challenging)
  - *Runtime challenge* (all 3 use-cases)

Use-case I

- Pick non-CMS WLCG site that allows opportunistic resource usage (started with FermiGrid, have expanded to a few more OSG sites)
- Solve the runtime problem
  - Parrot-wrap CMS payload to simulate the CMS software environment (see Slide 10 below)
    - CMSSW (framework), Workload Management python runtime code and grid UI (needed for lcg-cp stage-out) accessed via CVMFS
    - Site configuration in local file system emulated via Parrot: CVMFS/Frontier Proxy, remote data access via xrootd, stage-out via lcg-cp to CMS site
      [cf abstract #94 "*CMS Use of a Data Federation*"]

Use-case II

- Pick non-WLCG site that allows (opportunistic) resource usage (NERSC, not truly opportunistic but same challenges)
- Identical runtime problem to use case (1)
- Submission problem:
  - No GRID interface to site
  - BOSCO and tunnel via ssh. Integrated into GlideIn WMS [cf abstract #226 "*Accessing opportunistic resources with Bosco*"]
    - ➢ Performing Re-Reconstruction workflows at SDSC, see back-up slide (access via BOSCO, otherwise a somewhat "individual" setup)

Use-case III

- Use Cloud (OpenStack / EC2) resource that allows opportunistic usage (CMS exploring several options)
- Identical runtime problem to use case (I) and (II)
- Submission problem:
  - No GRID interface to Site
  - OpenStack / EC2 integration in new Glidein SW
    [cf abstract #94 "*Usage of the CMS Higher Level Trigger Farme as Clound Resource*"]
- In addition there are non-commercial contributions

Parrot Wrapper

- Wrapper script around the CMS payload in the Glidein WMS pilot [4].

  https://github.com/dcbradley/parrot_glidein_wrapper

- Uses Parrot [6] to make remote IO accessible from local jobs (without requiring root privileges), effectively allowing to "mount" CVMFS

- Also auto-detects site proxies to put them into the CMS site configuration and in general provides the CMS specific site configuration files

Proxy Setup

- Important for scalability to cache Frontier and CVMFS requests
  - Auto-detected proxy at site if available
  - Fall back to whatever is configured through the parrot wrapper provided site configuration
- CMS is in the process of setting up global fall-back proxies at CERN/Fermilab (CERN deployed and tested)
- For large resources we can always do special setups (use "local" or "close" CMS owned proxy server)

Stage-out

- To remove dependency on site deployed software, use EMI-2 CVMFS GRID UI at grid.cern.ch and then stage out to CMS site with lcg-cp
- CMS site and stage-out settings are configured in the parrot wrapper provided site configuration
- As we scale up, need to spread the stage out load over multiple sites, otherwise could run into networking/ scaling issues

Operational Aspects

- Goal is to have the opportunistic usage as transparent as possible for the operations
- Create “fake CMS site” that maps to opportunistic resources, configure the fake site in all the relevant CMS information systems so it can be used by the production tools

Operational Aspects

- Two options mapping physical resources to “fake site” :
  - Meta site: shared wrapper configuration for multiple physical resources (using glidein frontend groups). One single site configuration (fallback proxy, stage out location etc).
  - Dedicated opportunistic site: custom optimized site configuration (proxy and stage out) possible
- Balance between optimizing a specific resource and keeping the number of sites manageable in terms of operations overhead
  - Examples of Meta site: OSG sites (small scale opportunistic)
  - Example of dedicated site: SDSC (thousands of cores)
- Day-to-Day operations challenge: monitoring and trouble-shooting
  - Specially challenging at Meta sites, since physical site non-visible (the price to pay for transparent use of resources ?)

Conclusions

- Opportunistic Resources are useful in order to optimize the productivity of human assets and to cover short-term peak computing demands
- Depending on the type of opportunistic resource, the SW challenges are at submission time and/or at run time
- CMS already tested at large scale, with successful and encouraging results
- Operational challenges, for example in terms of monitoring, are non-negligible

Glossary and References

[1] *CMS* : Compact Muon Solenoid

[2] *LS1* : LHC Long Shutdown 1 (2013-14)

[3] *WLCG* : Worldwide LHC Grid

[4] *Glidein WMS* : Workflow Management System used by CMS (http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/index.html)

[5] *CVMFS* : CernVM File System

[6] *Parrot* : Virtual File System (http://www3.nd.edu/~ccl/software/parrot/)

[7] *BOSCO* : Job Submission Manager
(https://twiki.grid.iu.edu/bin/view/CampusGrids/BoSCO)

[8] *Open Stack* : Open Sources SW for building Clouds
(http://www.openstack.org/)

[9] *EC2* : Amazon Elastic Compute Cloud (http://aws.amazon.com/ec2/)

Thank You !

Back-Up Slides

The San Diego (SDSC) Example