Scientific Computing on Emerging Infrastructures

using “HTCondor”

HTCondor Week, 20th May 2015

Sanjay Padhi

University of California, San Diego
Scientific instruments:
→ More precise
→ Processing and analysis needs more resources

Software developments:
→ reduces processing timing

Complexity (PileUps):
→ increase in resource demand

We need large (sets of) resources:
→ use them efficiently

Emerging infrastructures (timing):
→ Decouple at software layer
   → HTC & HPC
→ Not one size fits all
Challenges with LHC Evolutions

LHC 8 TeV (04\textsuperscript{th} April 2012 - 16\textsuperscript{th} Dec. 2012 = 256 days), with \sim\hspace{1mm}10\text{hr/day}

Benchmark Points (approximate):

1. Run 1 (8 TeV, 2012); Lumi = 5 \times 10^{33}; HLT rate: 350 (base)+350 Hz; \# 6.4 billion evts

2. Run 2 (13-14 TeV, 2015); Lumi = 0.7 \times 10^{34}; HLT rate: ?? Hz; \# \sim\hspace{1mm}3.0 billion evts

3. Run 2 (13-14 TeV, 2016); Lumi = 1.5 \times 10^{34}; HLT rate: 1084 \pm 37 Hz \# 9.9 billion evts

4. Run 2 (13-14 TeV, 2017); Lumi = 1.5 \times 10^{34}; HLT rate: 1084 \pm 37 Hz \# 9.9 billion evts

5. Run 3 (14 TeV, 2019); Lumi = 2.0 \times 10^{34}; HLT rate: 1431 \pm 51 Hz \# 13.2 billion evts

6. Run 4 (14 TeV, 2025); Lumi = 5.0 \times 10^{34}; HLT rate \sim\hspace{1mm}10 \text{kHz} \# 92 billion evts
Evolution of Computing Model

**GRID**
- Virtual organizations (VOs) of group of users
  - Trusted by sites
  - Executive Policies
  - Provisioning by middleware
  - Provisioning by users
  - Resource Pledges
    - CPU, Disks, etc.
  - Resource limitations
    - (No elasticity)
  - Velocity: Days/Weeks @ Site Admins

**CLOUD** (Commercial)
- See: Talk by J. Kinney (AWS)
  - Pay-As-You-Go
  - Security (You decide)
  - Provisioning by users
    - Empower users
  - Elasticity
  - Volume:
    - Near infinite capacity
  - Advanced technologies
  - Variety: Spot Markets
  - Velocity: 3.5min startup

**HYBRIDS**
- We have to live using a combination of GRIDs and CLOUDs
  - Need homogeneity at the harnessing level.
  - HTCondor?

---

**GRID:** Distributed Computing with “user” level resource access across participating sites
Around 2004, CDF started using Distributed Computing between FNAL & INFN

→ Homogeneity was achieved at the “user access level” using HTCondor

Operated until the “end” using the hybrid model

→ for Simulations and User activities: Local and GRID
Evolution of Late-binding technology: (HT)Condor


Only one communication language among all - ClassAds

Now: OSG Connect?
Evolution of Late-binding technology: (HT)Condor

Early 2006: late binding glidein based WMS
- Location: Build 32 @ CERN

Jaime Frey          Todd Tannenbaum
Evolution of Late-binding technology: (HT)Condor

USCMS glidein based WMS development efforts (independently) in ~Dec. 2006

Igor sfiligoi (FNAL) and I worked together joining CMS in April 2008

→ Integration of various production aspects from previous generation gWMS

First deployment of glideinWMS at UCSD (2008) for user analysis activities

- Various glideinWMS developments
- User Analysis Framework for CMS using glideinWMS
- Integrated submissions infrastructure to ARC and CREAM-CEs in glideinWMS

First production version co-developed & demonstrated during CCRC-08

 Scalability and interoperability within glideinWMS

D Bradley¹, I Sfiligoi², S Padhi³, J Frey¹ and T Tannenbaum¹

¹University of Wisconsin, Madison, WI, USA
²Fermilab, Batavia, IL, USA
³University of California, San Diego, La Jolla, CA, USA

In the CMS CCRC-08 exercise, glideinWMS successfully integrated over 4000 CPUs from more than 40 sites across EGEE, NorduGrid, and OSG. This was the first time that the NorduGrid ARC interface was used in CMS. The other sites were accessed via the gt2 protocol.
Late-binding technologies in Scientific Computing

GlideinWMS: currently used in almost all CMS Production & Analysis infrastructure

The Pilot Way to Grid Resources Using glideinWMS.

Currently: CMS (glideinWMS), ATLAS (PanDA), LHCb (DIRAC), ALICE (AliRoot)
- All are pilot-based late-binding technologies → widely used

Late-binding changed the course of LHC Computing → Revolutionary new direction
- Homogeneity, virtual batch system, separates provisioning from Scheduling

Wonderful! What is missing?
- Fault Tolerant (Manual? Pilot waste is not a waste, rest is Condor's problem)
- Elasticity (No. Use opportunistic resources or ask WLCG for more future pledges)
- Multi-tenancy (Absolutely, but all users/VOs should use the same Linux OS)
- Virtualization (No, Why do you need it?)

GRID solutions have architectural problem (Everything is a “job”)
- No elastic service, No mechanism for your own distributed applications.
Computing Challenges with LHC Evolutions
Recent evaluation of processing time, including optimizations in CMSSW

Run 1 (B1) = 6.4 billion evts:
Using $5 \times 10^{33}$; Time $\sim$ 15 sec/evt
On a 15k CPU site $\sim$ 2.5 months

SDSC ($\sim$3k cores) used for parked data
Without including GEN-SIM:
Total = $6.4 + 6.4$ (MC) = 12.8 billion evts
On a 15k CPU site $\sim$ 5 months
Using current estimate:
- T1 (2015): 300kHS06 ~37k cores
- T1 (2016): 400kHS06 ~50k cores
- T1 (2017): 520kHS06 (?) ~65k cores

In any case “burst” modeling will require being part of a “huge” pool of resources
- Amazon EC2 is a good example

Note: Use of GPUGPU/Co-Processors will help

(Need CMSSW with CUDA → huge software change else very low efficiency)

**Essential:** Clustering of GPUs per CPU becomes essential in the Workload management system
Burst Modeling

Finite number of resources by the experiments (Compute as well as Storage)

“Burst” usage is modeled using delays (~months) due to (re)processing capabilities

Elasticity in the system is really essential
Scalability and Elasticity in Scientific Computing:

- Currently scalability is defined based on “owned” or opportunistic resources
- Purchasing/Owning can have delays with “hardware technology of that date”
- LHC Run-I experience strongly suggests the need for elasticity due to “burst” (irregular fast turn-around) processing needs by its proponents
- On-demand provisioning of VMs using Amazon EC2 can help

If the “batch” system busy but cloud available: Expand batch system into the cloud

Demonstrated similar expansion/”shrink” to cloud using custom made auto-elasticity:

Multi-tenant approach - Virtualization

OS level provisioning not only can satisfy other experiments needs

- It adds elasticity in the computing system in case of “burst needs”
- One can also add private VMs to the whole central pool for enhanced usage

Not possible to use via glideinWMS unless all experiments use exact same linux flavor

- Resource clusterization/partition is complicated via glideinWMS

Virtualization depends on need
1. Low level virtualization
   → Containers (shares kernel)
   → LXC (cgroups), Docker, etc.

2. Hypervisor-based (heavy):
   - emulates virtual hardware
   - Hyper-V, KVM, Xen, etc.
Transition from GRID to EC2 Cloud - Hypervisors

GRID Sites - GlideinWMS
Condor Worker  Condor Worker  Condor Worker  Condor Worker
OS & Other items Enforced

Cloud: AWS EC2
Condor Worker  Condor Worker  Condor Worker  Condor Worker

OS & Other items Enforced

```
-bash-4.1$ condor_status
 Name OpSys Arch State Activity LoadAv Mem ActvtyTime
 ip-172-31-18-37.us LINUX X86_64 Unclaimed Benchmark 0.660 590 0+00:00:04
 Machines Owner Claimed Unclaimed Matched Preempting

 X86_64/LINUX 1 0 0 1 0 0
 Total 1 0 0 1 0 0

-bash-4.1$
```
CMS Analysis workflow using Amazon EC2
Scheduler was installed on a non-T1/T2 site
- by default 0 WNs associated with the site
Auto-elasticity based on submitted jobs
- Computing-on-demand
- “Custom made” auto-elasticity implementation
- Site expands/shrinks based on submitted jobs

Input data for CRAB3 to the WN via xrootd
CMSSW via CVMFS
GRID user NOT part of the “docker” group

→ People (Italians?) are interested in pushing this forward to the WLCG

HTCondor with CCB is not very docker/container friendly (can be fixed)

→ Google users found ways to use within their own “containerized clusters”

→ Can work: docker run --rm --privileged --net=host -ti -e 'spadhi/condorv2'

Amazon ECS works extremely well with user containers

See talk by: J. Kinney
Applications need to be:

- Fault Tolerant (Withstand failure)
- Scalable (Auto load balance)
- Elastic (Can grow and shrink based on demand)
- Leveraging the modern kernel isolation (cgroup)
- Mixed workload, Multi-tenancy, etc.
- Virtualization

Scheduling batch is “simple”

→ Can we schedule services? Hadoop, Squids, PhEDex, etc.

Mesos: A Platform for Fine Grained Resource share

- Overcomes static partitioning issues
- Build and run distributed systems
- Multi-resource scheduling (Mem, CPU, disk, etc.)
- Supports Docker Containers
- Isolation between tasks and containers
Rapid developments in the industry toward GPUs (also MICs) with high performance
We are likely to be forced into new Architectures by industry developments
For CMS this will be a very large change (impossible before LS2, hard say HL-LHC)

- Development of CMSSW using dedicated environment CUDA or OpenCL
- Even then we will need both CPUs and parallel processing units.

Clusterization of compute resources will be important
Cloud providers use VMs with direct access to the GPU

→ Use IO memory management unit to handle direct memory access

Use hypervisor for passing through PCI devices to guest VMs upon creation.

1:1 or 1:N mapping of VMs to GPUs is possible
Scientific Computing is evolving into the Cloud era

→ Most likely we will stay with the Hybrid model of GRIDS and CLOUDS

Enormous developments from Public/Commercial CLOUDS pioneered by AWS

HTCondor is still evolving with the fast paced technological evolutions

HPC is all about timing and parallelism (Not part of this talk)

- Mira at Argonne: 260,000 parallel threads producing 30M $W+5jet$ evts/hr
- GeantV vector prototypes use new VecGeom classes (dynamic threads)

Next generation supercomputer probably be FPGA based

- performance, capacity, and optimized price/performance/watt (power)
- Embedded Processors → FPGA design to the SW design
- PetaLinux Embedded Linux solutions on Xilinx (FPGA) processing systems
- For Triggers or Highly Streaming Computing → “Hardware/soft Scheduler”
Case Study: Amazon Web Services for the CMS Experiment

Lothar A Bauerdick, Tommaso Boccali, Ken Bloom, Daniele Bonacorsi, Ian Fisk, Maria Girone, Claudio Grandi, David Lange, Sanjay Padhi, Frank Wuerthwein

* Fermilab, Batavia, IL, United States
b Universita di Pisa and INFN, Italy
c University of Nebraska, Lincoln, United States
d Universita e INFN, Bologna, Italy
e European Organization for Nuclear Research, Geneva, Switzerland
f Lawrence Livermore Nat. Laboratory, Livermore, United States
g University of California, San Diego, United States

E-mail: Lothar.Bauerdick@cern.ch, Tommaso.Boccali@cern.ch, kenbloom@unl.edu, daniele.bonacorsi@bo.infn.it, Ian.Fisk@cern.ch, Maria.Girone@cern.ch, Claudio.Grandi@cern.ch, David.Lange@cern.ch, Sanjay.Padhi@cern.ch, fkw@ucsd.edu

ABSTRACT: In this document we propose a case study of using Amazon Web Services (AWS) for the evolution of the CMS computing model. With the evolution of LHC energy and luminosity, the CMS model is expected to expand into the on-demand cloud era. We discuss the requirements, benchmark tests, services and infrastructure changes needed in order to integrate elasticity into the CMS computing model. The proposed study will also be used as a cost evaluation model for burst usage related to CMS production and analysis workflows.

Submitted as a CMS IN Note and has been approved by AWS
Thanks to the support from USCMS & AWS for this proposal
3.1 Phase-I of the proposal

There are a few high level goals for this phase using the proposed AWS pilot program. The evaluation period will be between May 2015 - March 2016. At peak, the workflow is expected to use up to 56K core instances (m3.2xlarge) for a month spread over the evaluation period. Based on the spot pricing of $0.0641/hour per node using the current generation of computing resources. The node consists of 8 vCPUs, 30 GiB of memory with 160 GB of SSD disks. We plan to incorporate demand driven burst usage at FNAL Tier1 via:

1. Event generation using Pythia, aMC@NLO, etc. with LHE datasets [Development and Integration Phase].

2. Hadronisation and full detector simulation [Development, Integration and Production Phase].

3. Digitization and Reconstruction with final AOD/mini-AOD format [Production Phase].

4. Data transfer of the final and intermediate products using PhEDEx to the FNAL Tier1 [Production Phase].

5. Data transfer of the final and intermediate products using PhEDEx or Amazon CLI to a S3 storage [Development and Integration Phase].

6. User analysis capabilities using mini-AOD with elasticity for a period of 1 month [Development and Integration Phase].
3.2 Phase-II of the proposal

In this Phase April 2016 - March 2017, we plan to enhance the AWS pilot program into a full production mode. The workflow in this period is expected to use up to at most a factor of 4 (See Table 2) more resources \((4 \times 100,000 = 400,000)\) than that is owned by the CMS Collaboration worldwide for a period of 1 month. The exact scale for Phase-II will be decided based on the experience gained during Phase-I studies. We plan to use this for full production and analysis usage by the whole collaboration consisting of more than 3000 users. The following workflows will be evaluated:

1. Event generation using Pythia, aMC@NLO, etc. with LHE datasets \([Production Phase]\).

2. Hadronisation and full detector simulation \([Production Phase]\).

3. Digitization and Reconstruction with final AOD/mini-AOD format \([Production Phase]\).

4. Data transfer of the final and intermediate products using PhEDEx to CERN and FNAL \([Production Phase]\).

5. Input RAW data transfer using PhEDEx from CERN and FNAL to AWS S3 \([Production Phase]\).

6. Prompt/Re-Reconstruction of LHC 2015/2016 data to its final data formats \([Production Phase]\).

7. Data transfer of the final and intermediate products using PhEDEx and Amazon CLI to a S3 storage \([Production Phase]\).

8. User analysis capabilities using mini-AOD with elasticity for a period of 1 month \([Production Phase]\).
We need to create Alarms for safety reasons