Scientific Computing on Emerging Infrastructures using “HTCondor”

HTCondor Week, 20th May 2015

Sanjay Padhi

University of California, San Diego

Scientific Computing

LHC probes nature at 10^{-17}cm → Weak Scale

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:

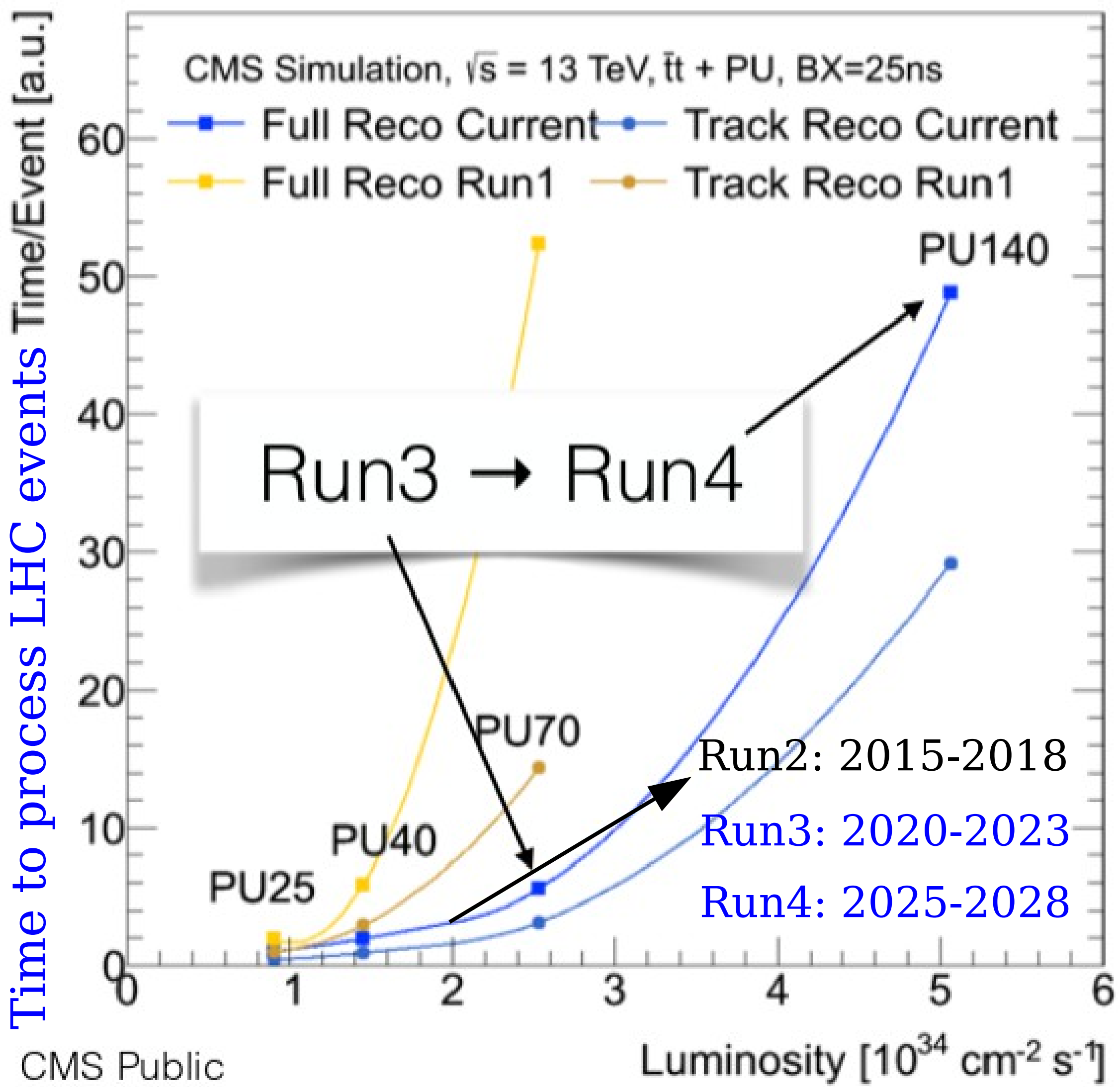
→ use them efficiently

Emerging infrastructures (timing):

→ Decouple at software layer

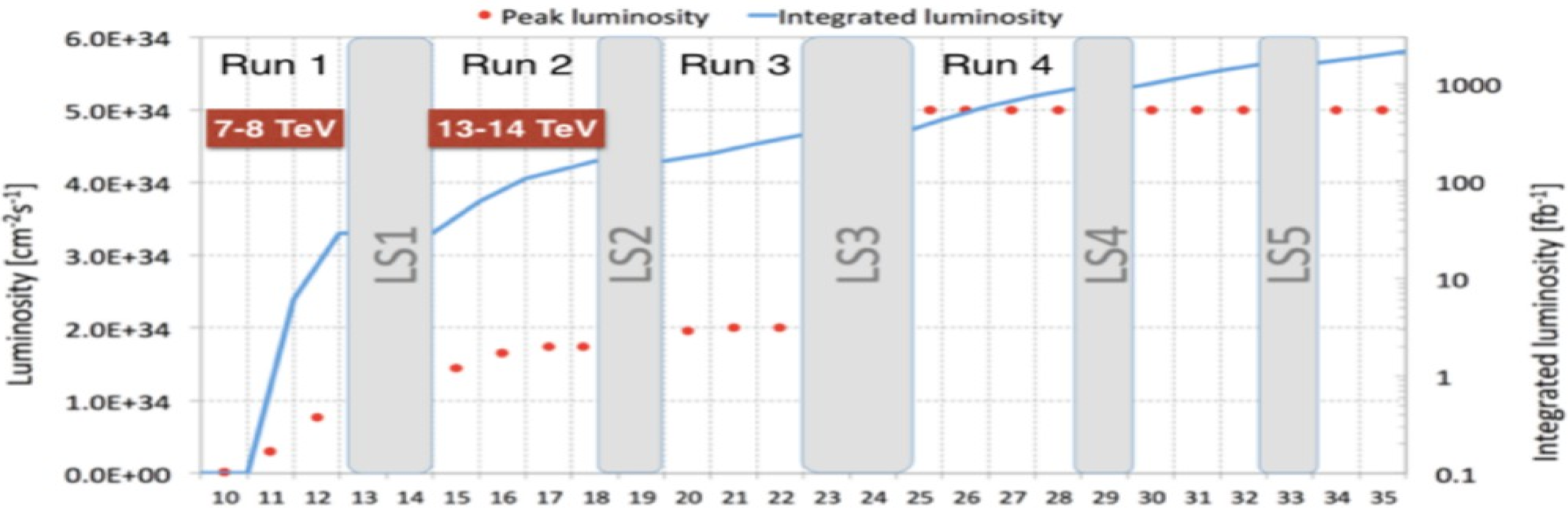
→ HTC & HPC

→ Not one size fits all



Number of events and complexity

Challenges with LHC Evolutions



LHC 8 TeV (04th April 2012 - 16th Dec. 2012 = 256 days), with ~10hr/day

Benchmark Points (approximate):

1. Run 1 (8 TeV, 2012); Lumi = 5 x 10^{33} ; HLT rate: 350 (base)+350 Hz; # 6.4 billion evts
2. Run 2 (13-14 TeV, 2015); Lumi = 0.7 x 10^{34}; HLT rate: ?? Hz; # ~3.0 billion evts
3. Run 2 (13-14 TeV, 2016); Lumi = 1.5 x 10^{34}; HLT rate: 1084 $\pm$ 37 Hz # 9.9 billion evts
4. Run 2 (13-14 TeV, 2017); Lumi = 1.5 x 10^{34}; HLT rate: 1084 $\pm$ 37 Hz # 9.9 billion evts
5. Run 3 (14 TeV, 2019); Lumi = 2.0 x 10^{34}; HLT rate: 1431 $\pm$ 51 Hz # 13.2 billion evts
6. Run 4 (14 TeV, 2025); Lumi = 5.0 x 10^{34}; HLT rate ~ 10 kHz # 92 billion evts

Evolution of Computing Model

GRID

Virtual organizations (VOs) of group of users

→ Trusted by sites

→ Executive Polices

→ ~~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

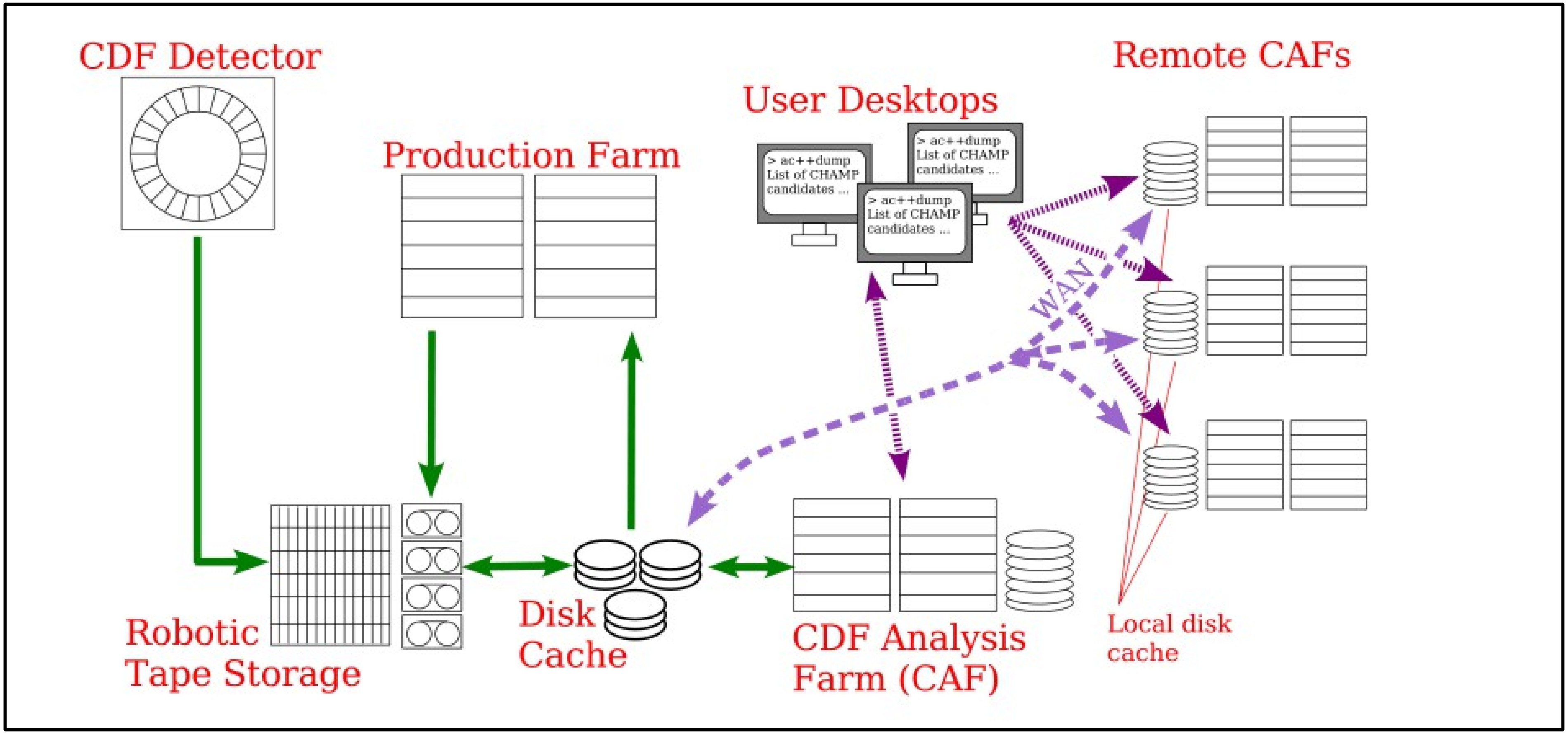
→ Need homogeneity at the harnessing level.

HTCondor?

GRID: Distributed Computing with "user" level resource access across participating sites

Evolution of Computing Model – CDF, FNAL

CDF used FNAL local resources for its processing and user activities



Around 2004, CDF started using Distributed Computing between FNAL & INFN

→ Homogeneity was achieved at the "user access level" using HTCondor

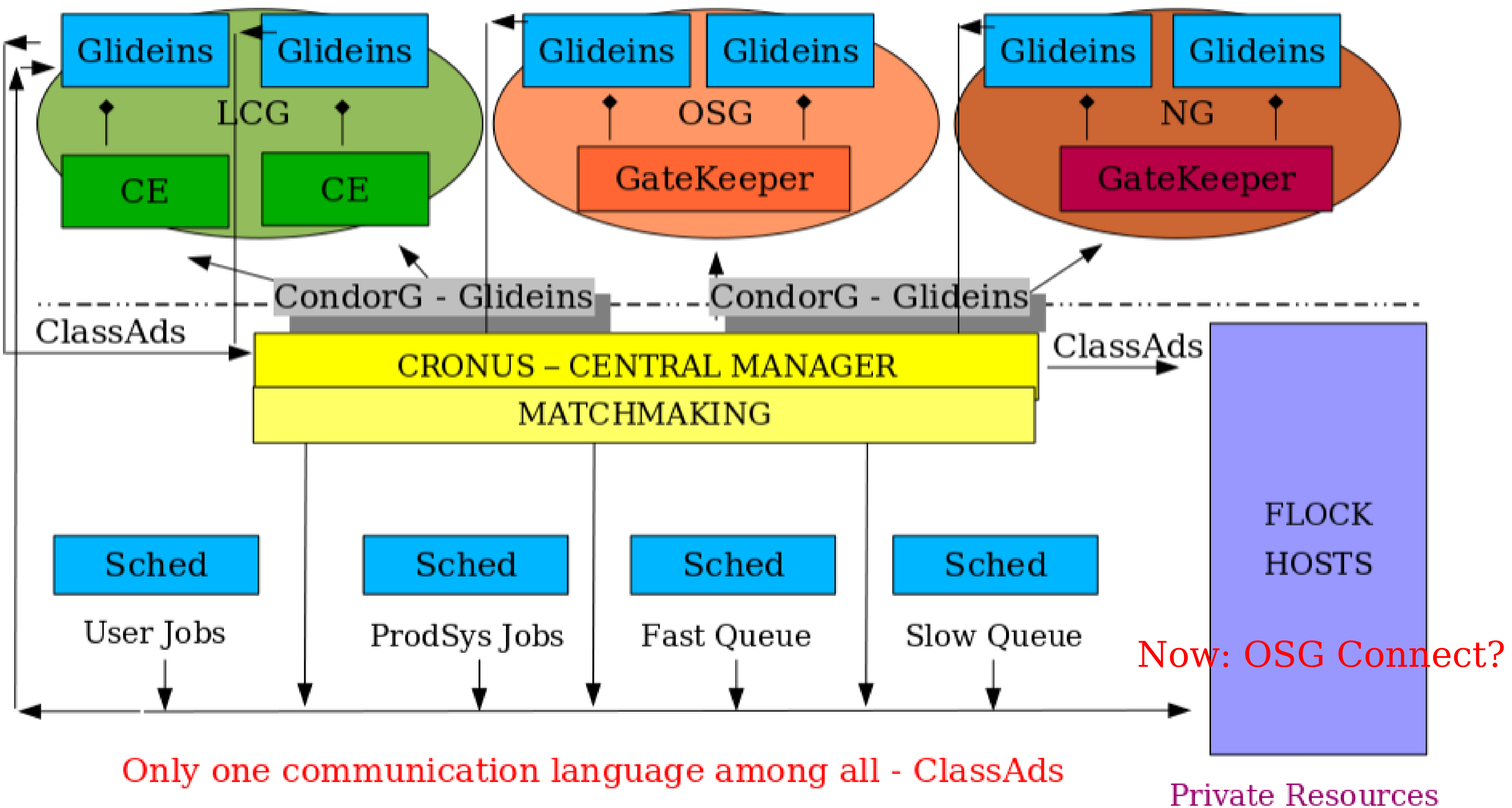
Operated until the "end" using the <u>hybrid model</u>

→ for Simulations and User activities: Local and GRID

Evolution of Late-binding technology: (HT)Condor

Sanjay Padhi, "CRONUS: A Condor Glide-in Based ATLAS Production Executor",

CHEP 07, 2-9 Sept. 2007, Canada



Only one communication language among all - ClassAds

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

FERMILAB-CONF-10-258-CD

D Bradley[1], I Sfiligoi[2], S Padhi[3], J Frey[1] and T Tannenbaum[1]

[1]University of Wisconsin, Madison, WI, USA
[2]Fermilab, Batavia, IL, USA
[3]University 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.

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

The Pilot Way to Grid Resources Using glideinWMS.

Igor Sfiligoi, Daniel C. Bradley, Burt Holzman, Parag Mhashilkar, Sanjay Padhi, and Frank Würthwein. *CSIE (2), page 428-432. IEEE Computer Society, (2009)*

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

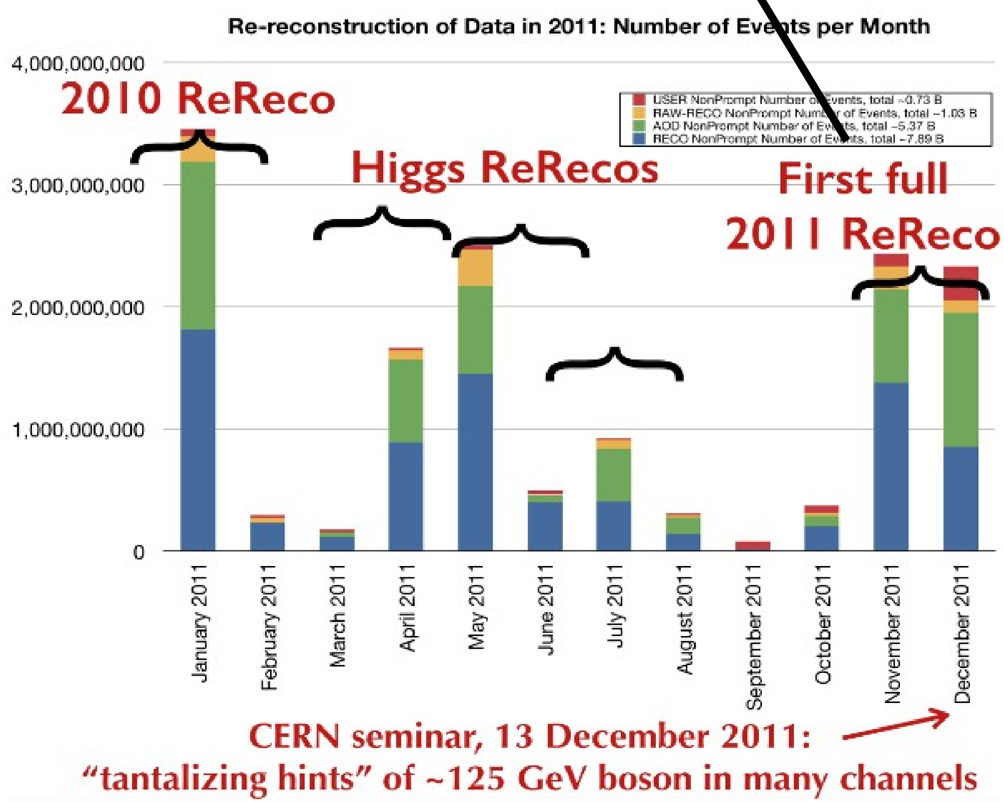
Event (Re)processing - “Burst usage” - 7/8 TeV LHC

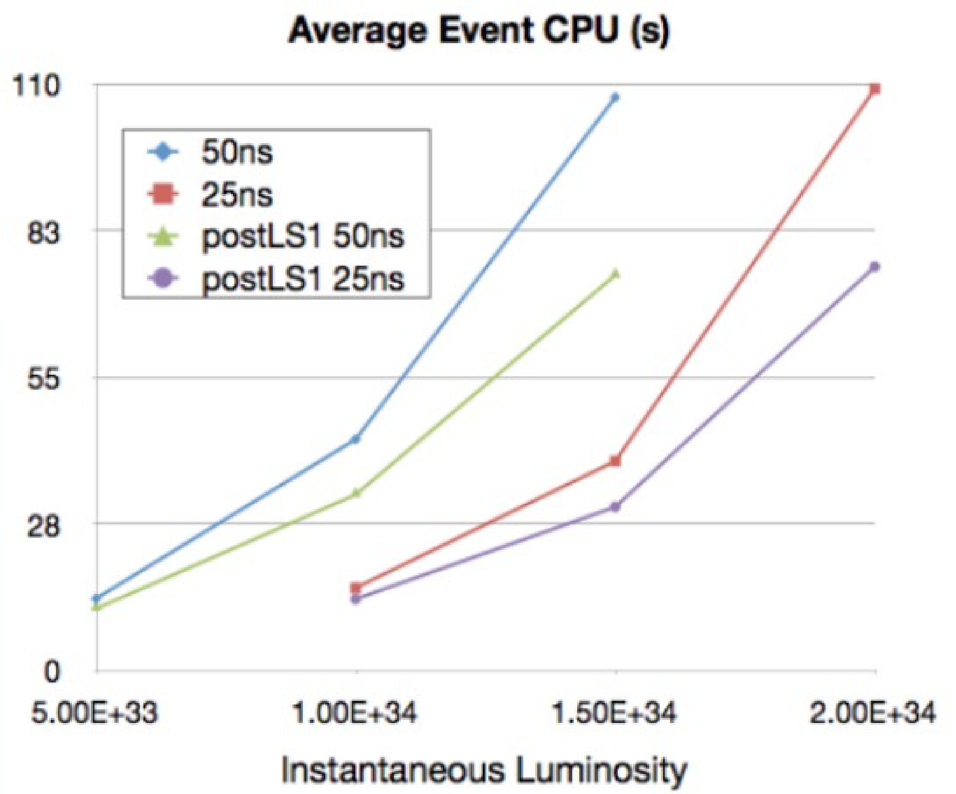
Recent evaluation of processing time, including optimizations in CMSSW

Run 1 (B1) = 6.4 billion evts:

Using 5 x 10^{33} ; Time ~ 15 sec/evt

On a 15k CPU site ~ 2.5 months





SDSC (~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 ~ 5 months

CMS Resources

Using current estimate:

- T1 (2015): 300kHS06 ~37k cores
- T1 (2016): 400kHS06 ~50k cores
- T1 (2017): 520kHS06 (?) ~65k cores

Very optimistic

B2: Run 2 (13-14 TeV, 2015); Lumi = 0.7 x 10^{34}; HLT rate: ?? Hz; # ~3.0 billion evts

Total: 3.0 + 4.0 (MC) ~ 7.0 billion events

Reprocessing Time on a 37k site @ 15 sec/event ~1.4 months

B3. Run 2 (13-14 TeV, 2016); Lumi = 1.5 x 10^{34}; HLT: 1084 $\pm$ 37 Hz # 9.9 billion evts

Total: 9.9 + 9.9 (MC) ~ 20 billion events

Reprocessing Time on a 50k site @ 30 sec/event ~ 4.6 months

Total Reprocessing time B2 + B3 ~5.4 months

B4. Run 2 (13-14 TeV, 2017); Lumi = 1.5 x 10^{34}; HLT: 1084 $\pm$ 37 Hz # 9.9 billion evts

Total Reprocessing time on a 65k site with B2 + B3 + B4 ~7.7 months

Assuming no of events MC : Data = 1:1 & no other T1 usage

In any case "burst" modeling will require being part of a "huge" pool of resources

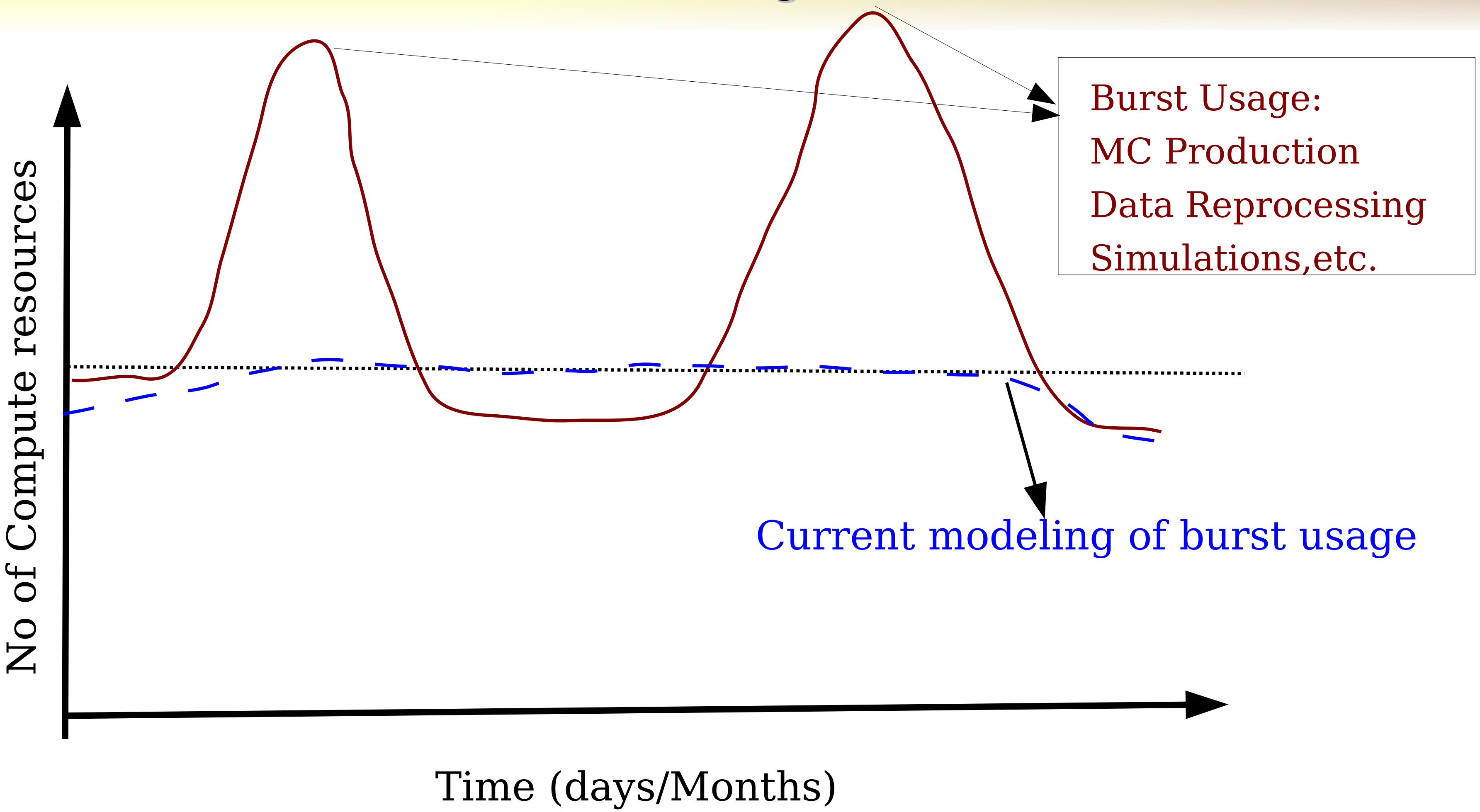
- Amazon EC2 is a good example

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)

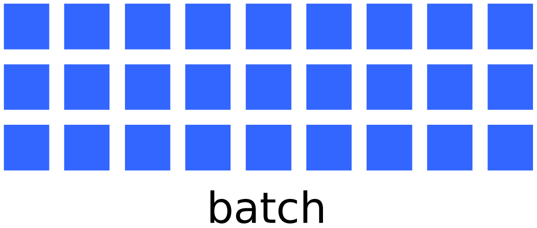
“Burst” usage is modeled using delays (~months) due to (re)processing capabilities

Elasticity in the system is really essential

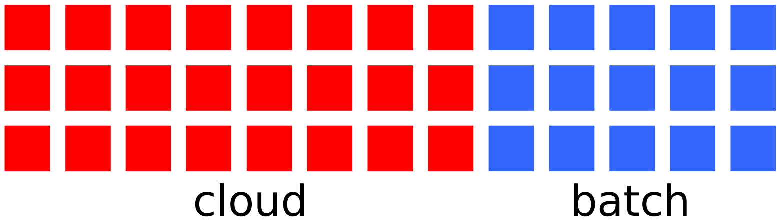
Elasticity in Scientific Computing

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:

S. Padhi, J. Kinney, “Scientific Computing using Amazon”, CERN, Feb. 02, 2015

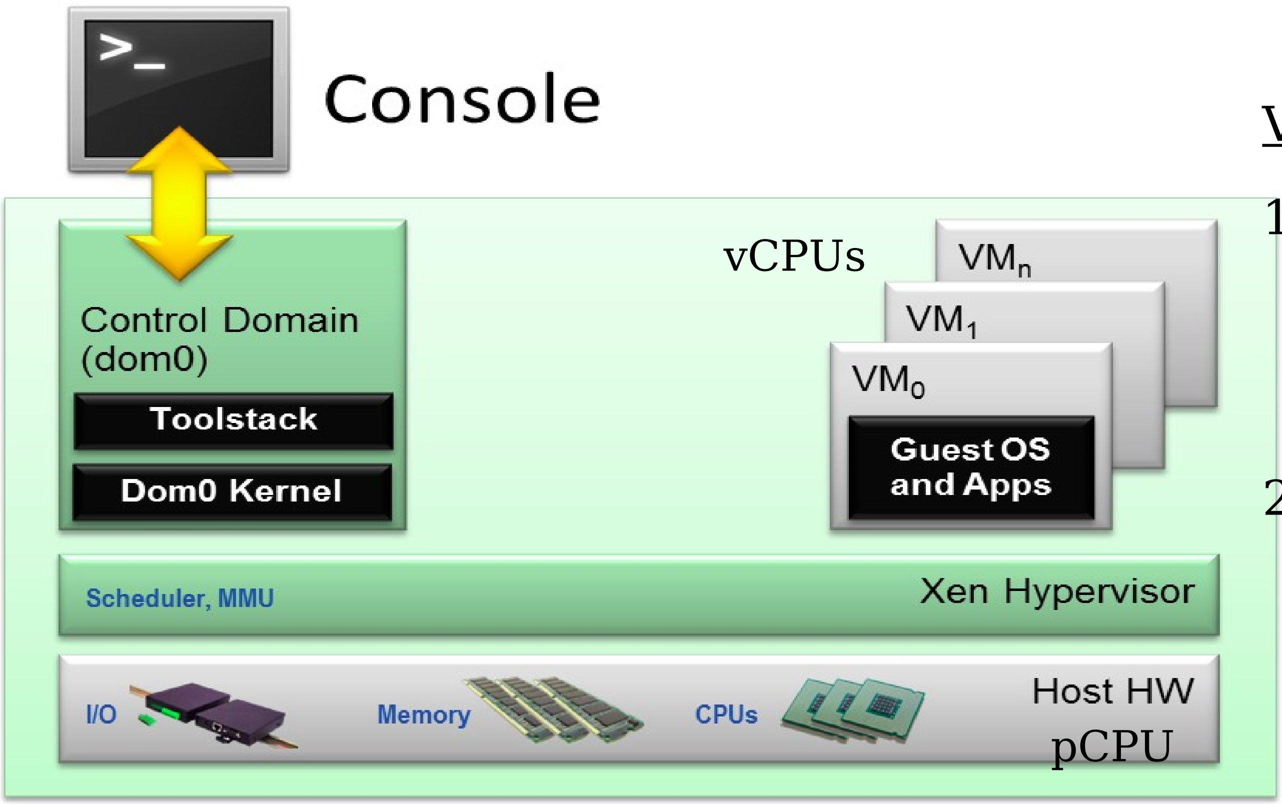
Multi-tenant approach - Virtualization

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

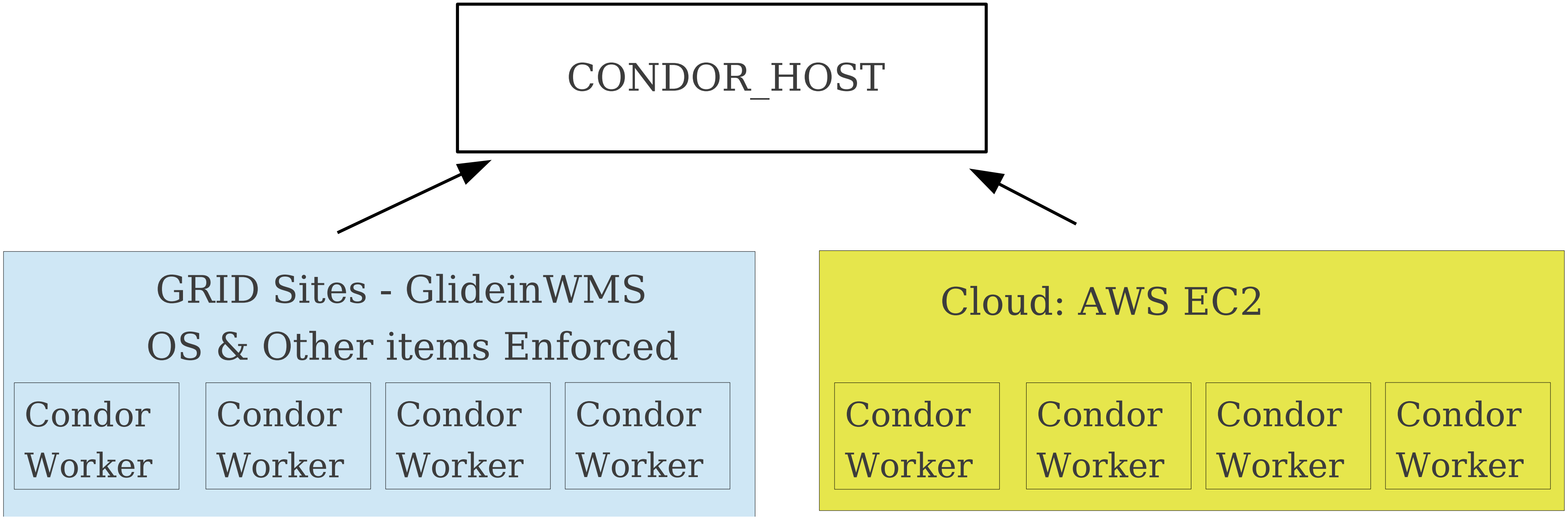
- Resource clusterization/partition is complicated via glideinWMS



<u>Virtualization depends on need</u>

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



Filter by tags and attributes or search by keyword — 1 to 1 of 1

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-b69dd240	t1.micro	us-west-2b	running	Initializing	None	ec2-52-24-194-170.us-...

```
-bash-4.1$ condor_status
-bash-4.1$ condor_status
Name                OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

ip-172-31-18-37.us LINUX      X86_64 Unclaimed Benchmar  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$
```

Transition from GRID to EC2 Cloud - Hypervisors

CMS Analysis workflow using Amazon EC2

Scheduler was installed on a non-T1/T2 site

- by default 0 WNs associated with the site

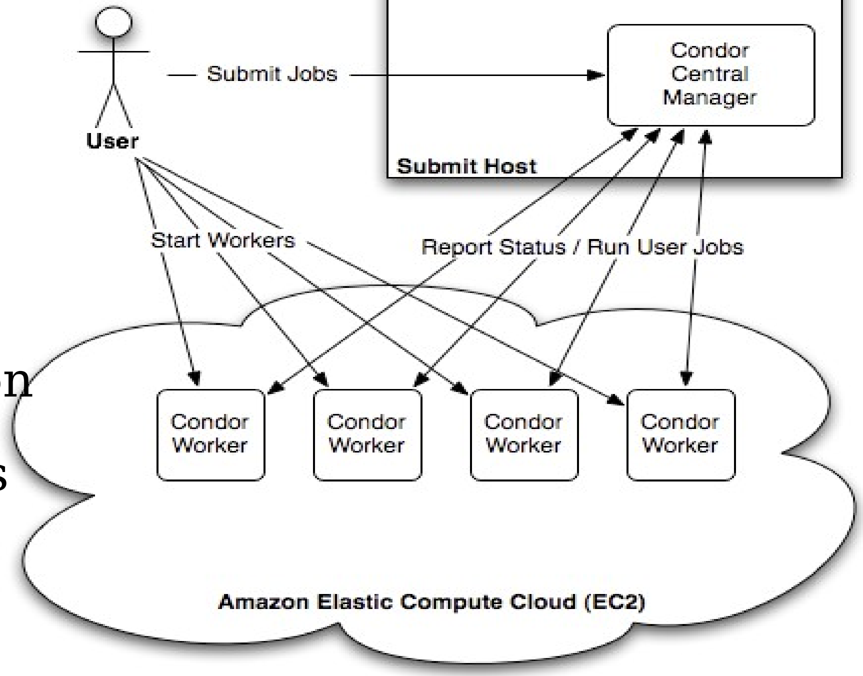
Auto-elasticity based on submitted jobs

- 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

Scheduler @ Brown Uni.
Thanks to M. Narain!



Start » [Mmascher] » Tasks » Jobs

Data | Charts | Show 25 entries

Task: 150131_110508_crab3test99:mmascher_crab_tutorial_Data_analysis_test5 NJobTotal: 17 Pending: 6 Running: 1 Unknown: 0 Cancelled: 0 Success: 10 Failed: 0 WNPostProc: 0 ToRetry: 0

Id	Status	AppExitCode	Site	Retries	Submitted	Started	Finished	Wall Time	Job Log	File Access	FTS File Status
1	finished	0	unknown	1	2015-01-31T11:05:43	2015-01-31T15:05:39	2015-01-31T15:09:37	00:03:58	Job Log,Job Log JSON,Post Job Log	File Info	N/A

Attempt No.	Restarts No.	Error Code/ Details	Job Status	Site	Submitted	Started	Finished	Wall Time	Job Log
1	1	0 / Application finished properly	finished	unknown	2015-01-31T11:05:43	2015-01-31T15:05:39	2015-01-31T15:09:37	00:03:58	Job Log,Job Log JSON,Post Job Log
		Postprocessing step finished properly							

Id	Status	AppExitCode	Site	Retries	Submitted	Started	Finished	Wall Time	Job Log	File Access	FTS File Status
2	finished	0	unknown	1	2015-01-31T11:05:43	2015-01-31T14:58:25	2015-01-31T15:02:28	00:04:03	Job Log,Job Log JSON,Post Job Log	File Info	N/A
3	finished	0	unknown	1	2015-01-31T11:05:43	2015-01-31T15:12:40	2015-01-31T15:18:52	00:06:12	Job Log,Job Log JSON,Post Job Log	File Info	N/A
4	finished	0	unknown	1	2015-01-31T11:05:43	2015-01-31T14:01:11	2015-01-31T14:52:00	00:50:49	Job Log,Job Log JSON,Post Job Log	File Info	N/A
5	finished	0	unknown	1	2015-01-31T11:05:43	2015-01-31T15:21:40	2015-01-31T16:08:07	00:46:27	Job Log,Job Log JSON,Post Job Log	File Info	N/A
6	finished	0	unknown	1	2015-01-31T11:05:43	2015-01-31T16:20:32	2015-01-31T16:31:45	00:11:13	Job Log,Job Log JSON,Post Job Log	File Info	N/A
7	finished	0	unknown	1	2015-01-31T11:05:43	2015-01-31T16:37:24	2015-01-31T16:47:19	00:09:55	Job Log,Job Log JSON,Post Job Log	File Info	N/A
8	running	N/A	unknown	1	2015-01-31T11:05:43	2015-01-31T17:41:23	1970-01-01T00:00:00	00:00:00	Not available	File Info	N/A
9	pending	N/A	unknown	1	2015-01-31T11:05:43	1970-01-01T00:00:00	1970-01-01T00:00:00	00:00:00	Not available	File Info	N/A
10	finished	0	unknown	1	2015-01-31T11:05:43	2015-01-31T17:20:29	2015-01-31T17:31:35	00:11:06	Job Log,Job Log JSON,Post Job Log	File Info	N/A

Transition from GRID to EC2 Cloud - Containers

CONDOR_HOST

See talk by Greg Thain

GRID Sites (Not for today)

~~OS and other items Enforced~~

Condor Cont.	Condor Cont.	Condor Cont.	Condor Cont.

CLOUD: Amazon EC2 or ECS

Condor Worker	Condor Worker	Condor Worker	Condor Worker

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

The New Reality

Applications need to be:

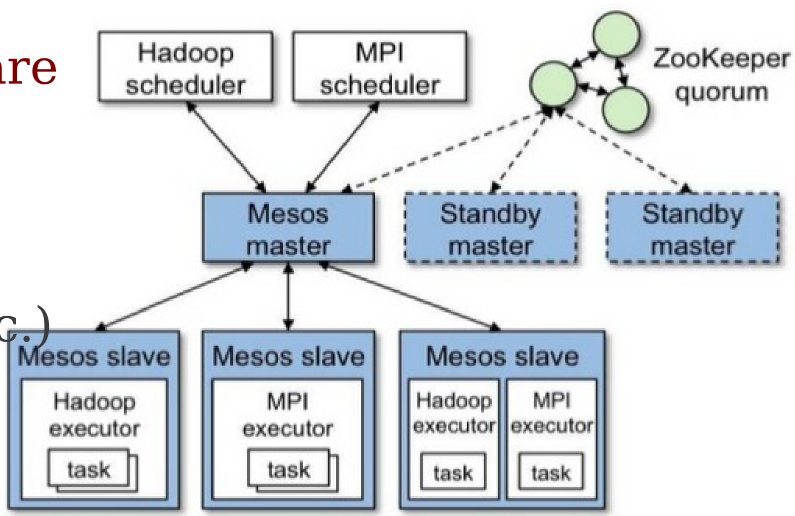
- Fault Tolerant (Withstand failure)
- Scalable (Auto load balance)
- Elastic (Can grow and shrink based on demand)
- Leveraging the modern kernel isolation (cgroup)
- Mixed workload, Multi-tenancy, etc.
- Virtualization

Scheduling batch is “simple”

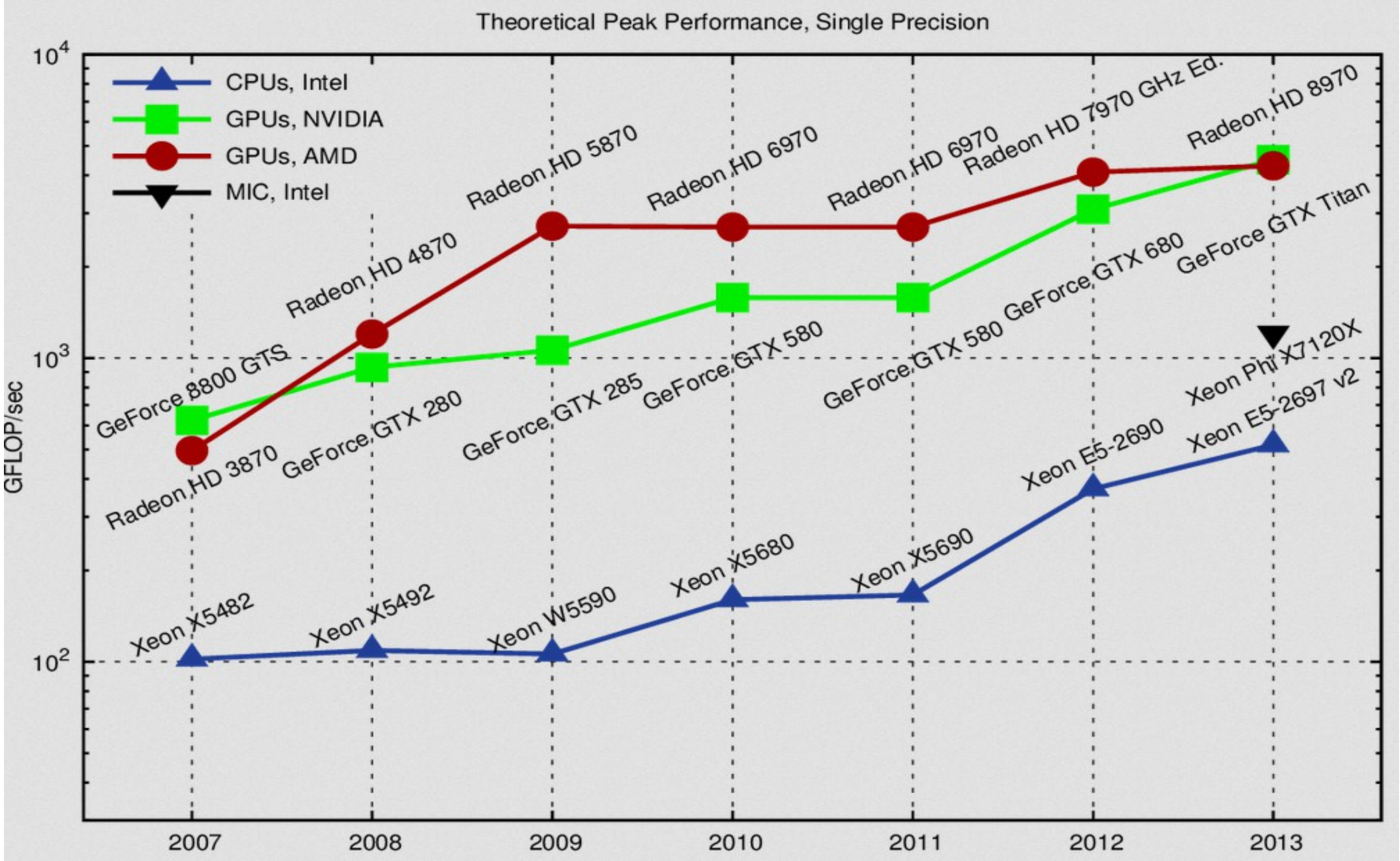
→ Can we schedule services? Hadoop, Squids, PhEDex, etc.

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



Novel Architectures - Parallel and Co-processor framework



Rapid developments in the industry toward GPUs (also MICs) with high performance

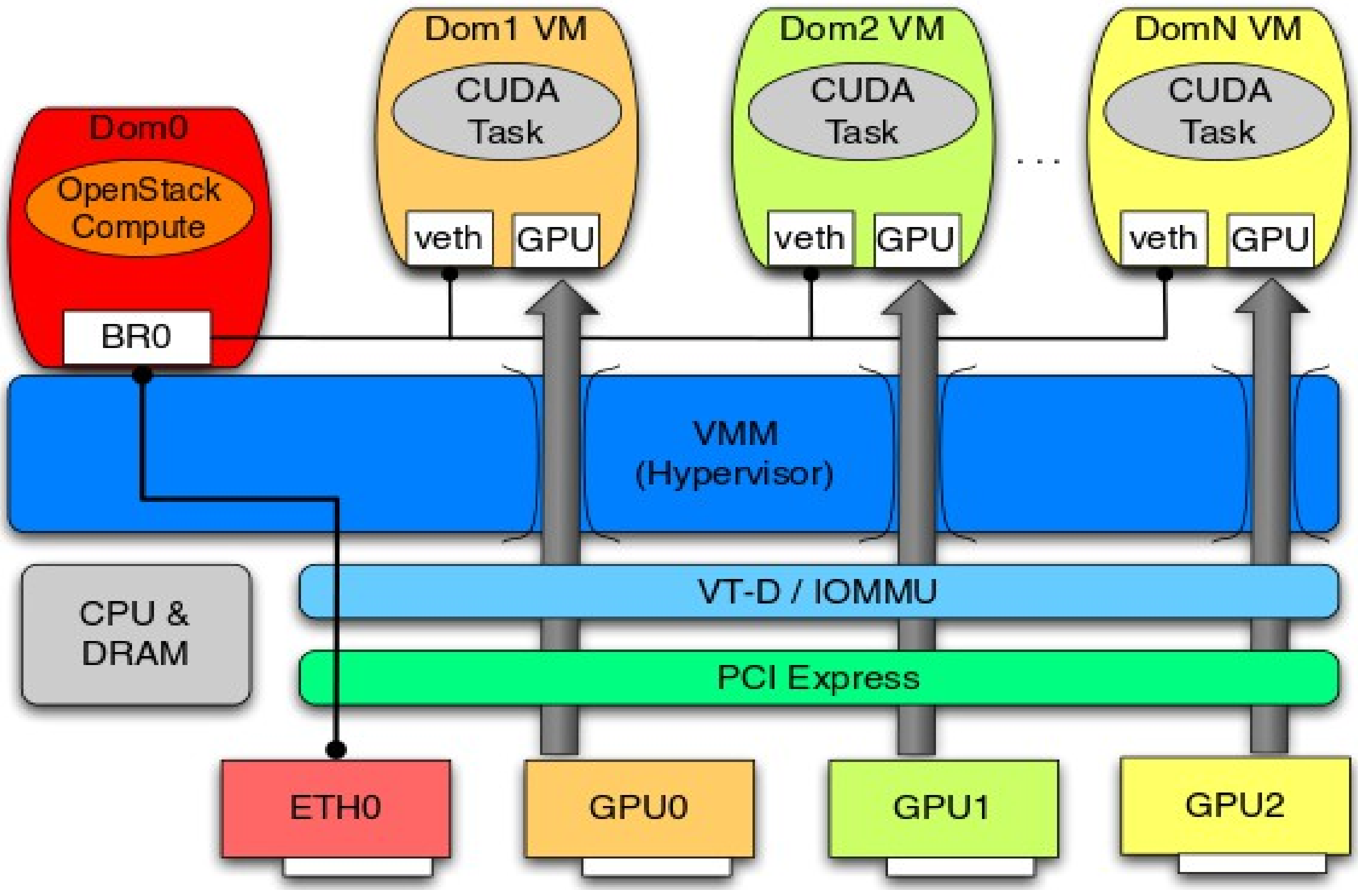
We are likely to be forced into new Architectures by industry developments

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

Virutalization and Clustering



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

Summary and Outlook

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,[a] Tommaso Boccali,[b] Ken Bloom,[c] Daniele Bonacorsi,[d] Ian Fisk,[a] Maria Girone,[e] Claudio Grandi,[d] David Lange,[f] Sanjay Padhi,[g] Frank Wuerthwein[g]

[a] *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

CMS Proposal Phase-I : May 2015 – March 2016

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 T1 via:

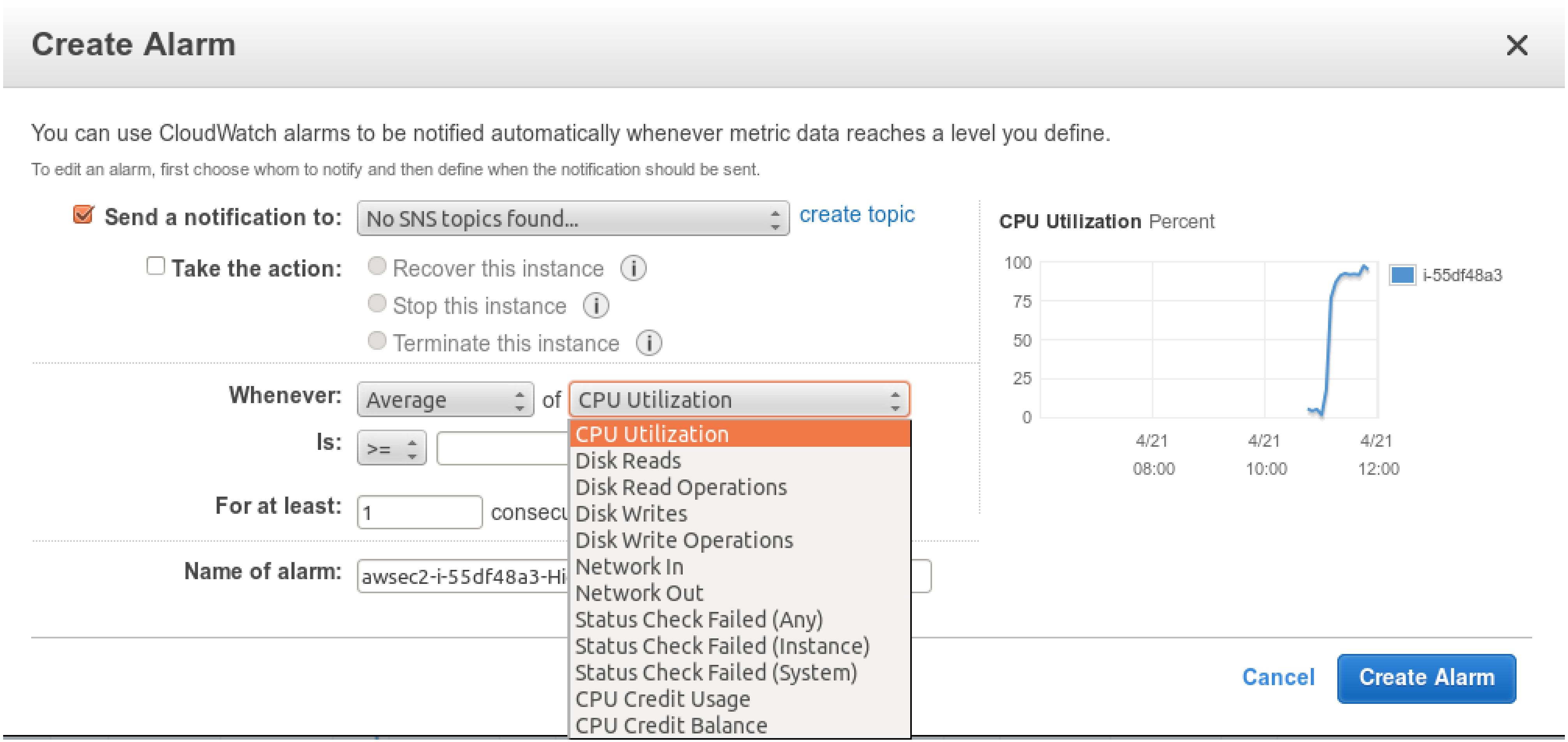
1. Event generation using Pythia, aMC@NLO, etc. with LHE datasets [*Development and Integration Phase*].
2. Hadronisation and full detector simulation [*Development, Integration and Production Phase*].
3. Digitization and Reconstruction with final AOD/mini-AOD format [*Production Phase*].
4. Data transfer 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 $\boxed{\text{factor of 4 (See Table 2) more resources } (4 \times 100,000 = 400,000)}$ 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 evaluted:

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*].

Alarms



We need to create Alarms for safety reasons