CernVM
SoFTware Appliance

CernVM on the Cloud

Predrag Buncic, CERN/PH-SFT

CernVM
SoFTware Appliance

CernVM Project

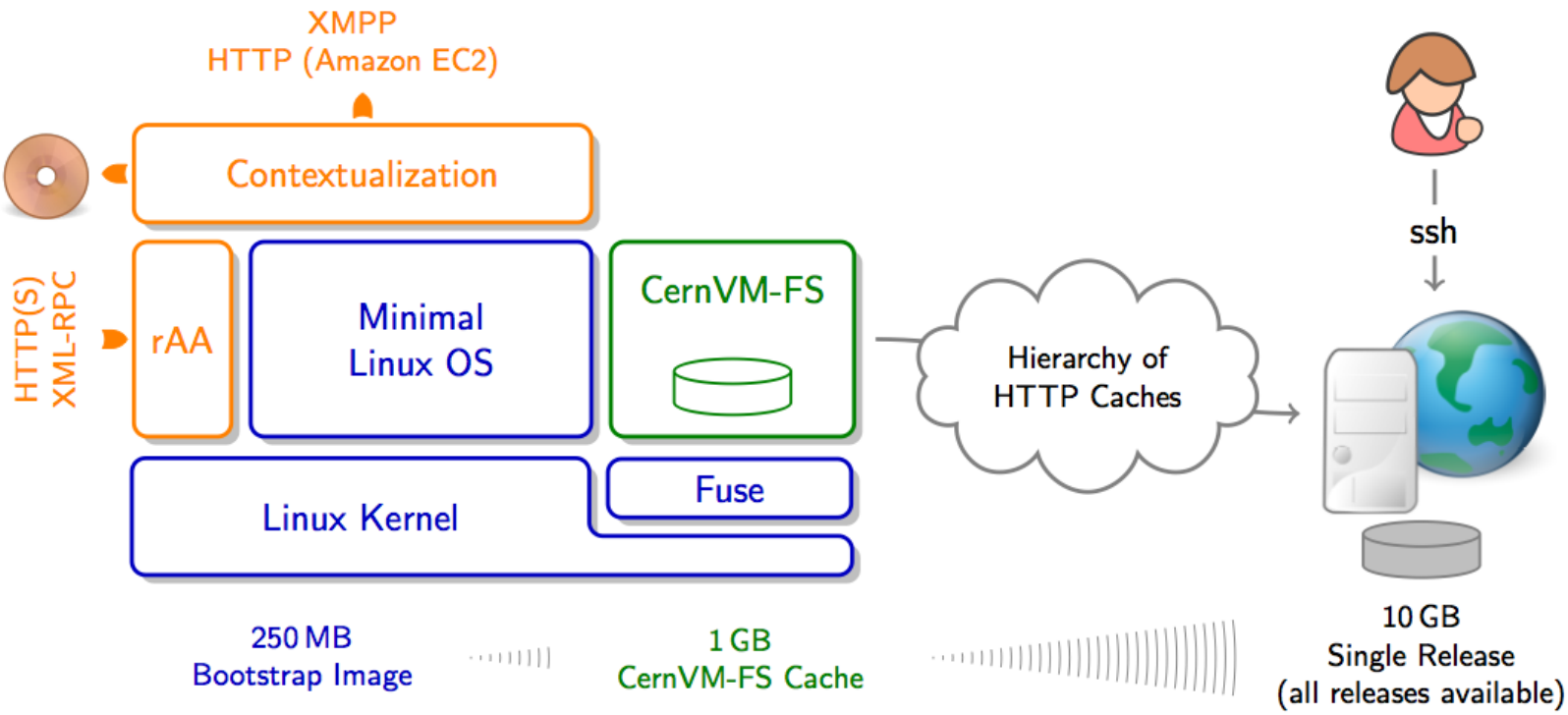
CernVM
SoFTware Appliance

Is CernVM suitable for deployment on Grid and Cloud infrastructure?

What are the benefits of going CernVM way comparing to more traditional[1)] approach to batch node virtualization?

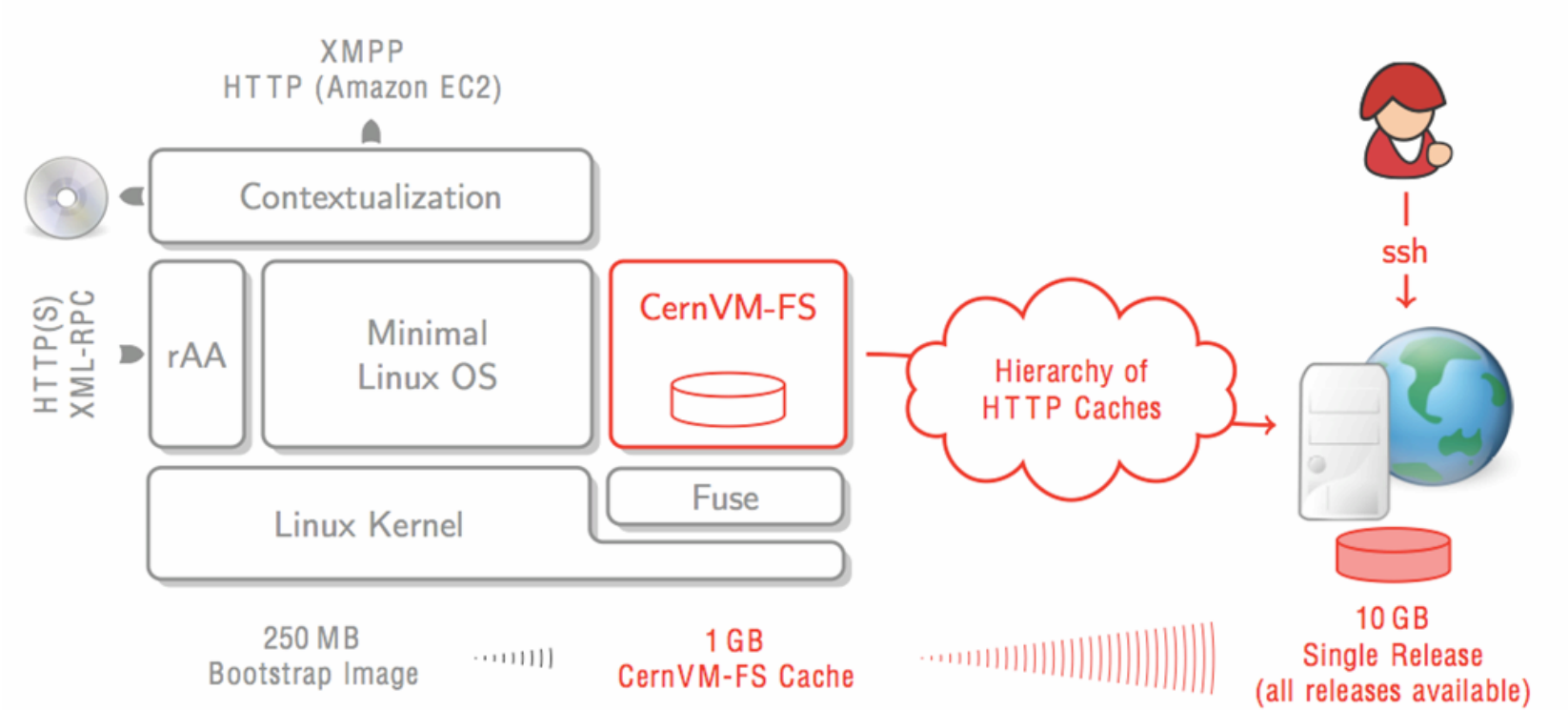
1) Traditional approach:
   - Take “standard” batch node [2GB] and add experiment software [10GB] and generate VM image. Have experiment and security team certify the image, deploy it to all sites and worker nodes. Repeat this procedure 1-2 times per week and per experiment.

CernVM Way



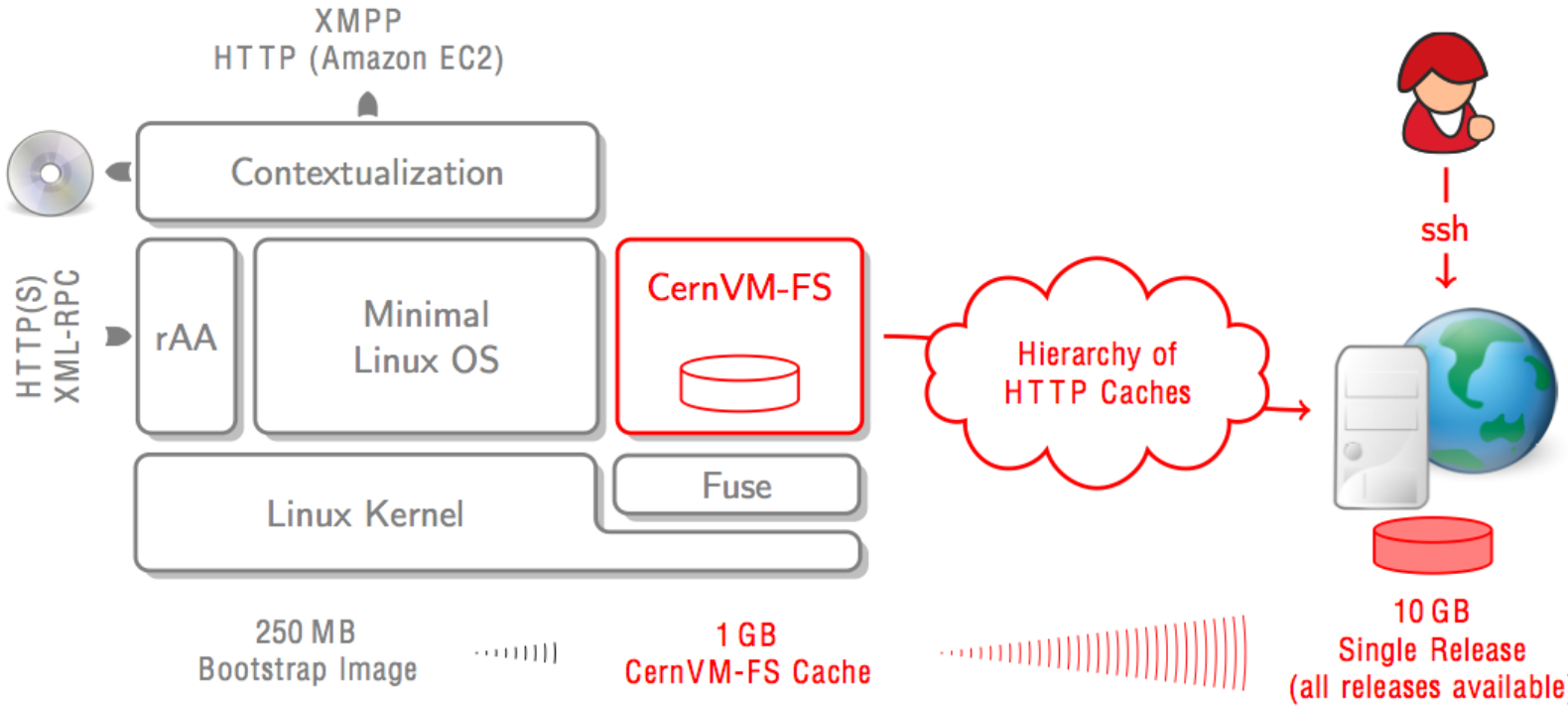
1. Minimal Linux OS (SL5)
2. CernVM-FS - HTTP network file system optimized for jus in time delivery of experiment software
3. Flexible configuration and contextualization mechanism based on public Cloud API

Part #1: Minimal OS image



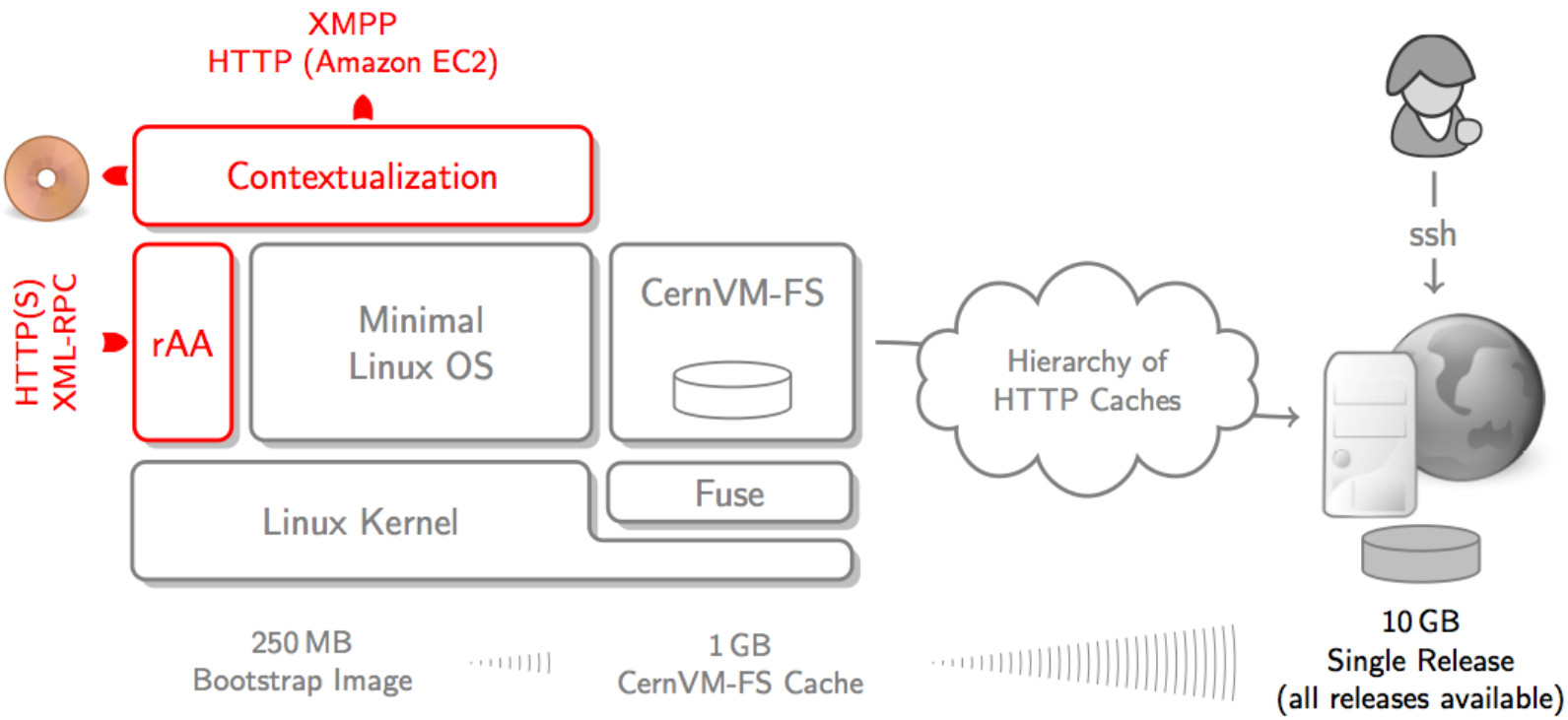
- Just enough OS to run LHC applications
- Built using commercial tool (rBuilder by rPath)
  - Top-down approach - starting from application and automatically discovering dependencies
- Small images (250MB), easy to move around

Part #2: CernVM-FS



- Experiment software is changing frequently and we want to avoid need to frequently update, certify and redistribute VM images with every release
- Only a small fraction of software release is really used
- Demonstrated scalability and reliability
- Now being deployed on across all Grid sites as the channel for software distributions

Part #3: Contextualization



- There are several ways to contextualize CernVM
  - ✓ Web UI (for individual user)
  - ✓ CernVM Contextualization Agent
  - ✓ Hepix CDROM method
  - ☞ Amazon EC2 API user_data method

CernVM
SoFTware Appliance

EC2 API – user data

Syntax

ec2-run-instances *ami_id* [-n *instance_count*] [-g *group* [-g *group* ...]] [-k *keypair*] [-d *user_data* |-f *user_data_file*] [--addressing *addressing_type*] [--instance-type *instance_type*] [--availability-zone *zone*] [--kernel *kernel_id*] [--ramdisk *ramdisk_id*] [--block-device-mapping *block_device_mapping*] [--monitor] [--disable-api-termination] [--instance-initiated-shutdown-behavior *behavior*] [--placement-group *placement-group*] [--tenancy *tenancy*] [--subnet *subnet*] [--private-ip-address *ip_address*] [--client-token *token*]

***user_data* specifies Base64-encoded MIME user data to be made available to the instance(s) in this reservation.**

Type: String
Default: None

EC2 Contextualization

- Basic principles:
  - Owner of CernVM instance can contextualize and configure it to run arbitrary service as unprivileged user
  - Site can use HEPIX method to inject monitoring and accounting hooks w/o functionally modifying the image

- The contextualization is based on rPath *amiconfig* package extended with CernVM plugin
  - This tool will execute on boot time (before network services are available), parse user data and look for python style configuration blocks.
  - If match is found the corresponding plugin will process the options and execute configuration steps if needed.

- For more info on CernVM contextualization using EC2 API, see:https://cernvm.cern.ch/project/trac/cernvm/wiki/EC2Contextualization

CernVM
SoFTware Appliance

user_data example

```
[cernvm]
# list of ',' separated organizations/experiments (lowercase)
organisations = cms
# list of ',' separated repositories (lowercase)
repositories = cms,grid
# list of ',' separated user accounts to create <user:group:[password]>
users = cms:cms:
# CVMFS HTTP proxy
proxy = http://<host>:<port>;DIRECT
# install extra conary group
group_profile = group-cms
# script to be executed as given user: <user>:/path/to/script.sh
contextualization_command = cms:/path/to/script.sh
# list of ',' seperated services to start
services = <list>
# extra environment variables to define
environment = CMS_SITECONFIG=EC2,CMS_ROOT=/opt/cms
```

cernvm-tools

- **A tool to simplify interaction with multiple clouds supporting minimal EC2 API**
  - Aims to simplify instantiation of contextualized CernVM images using EC2 user_data mechanism
- **Supports**
  - user_data templates, bulk submission
  - delegation of proxy certificate in user_data

- **Example:**

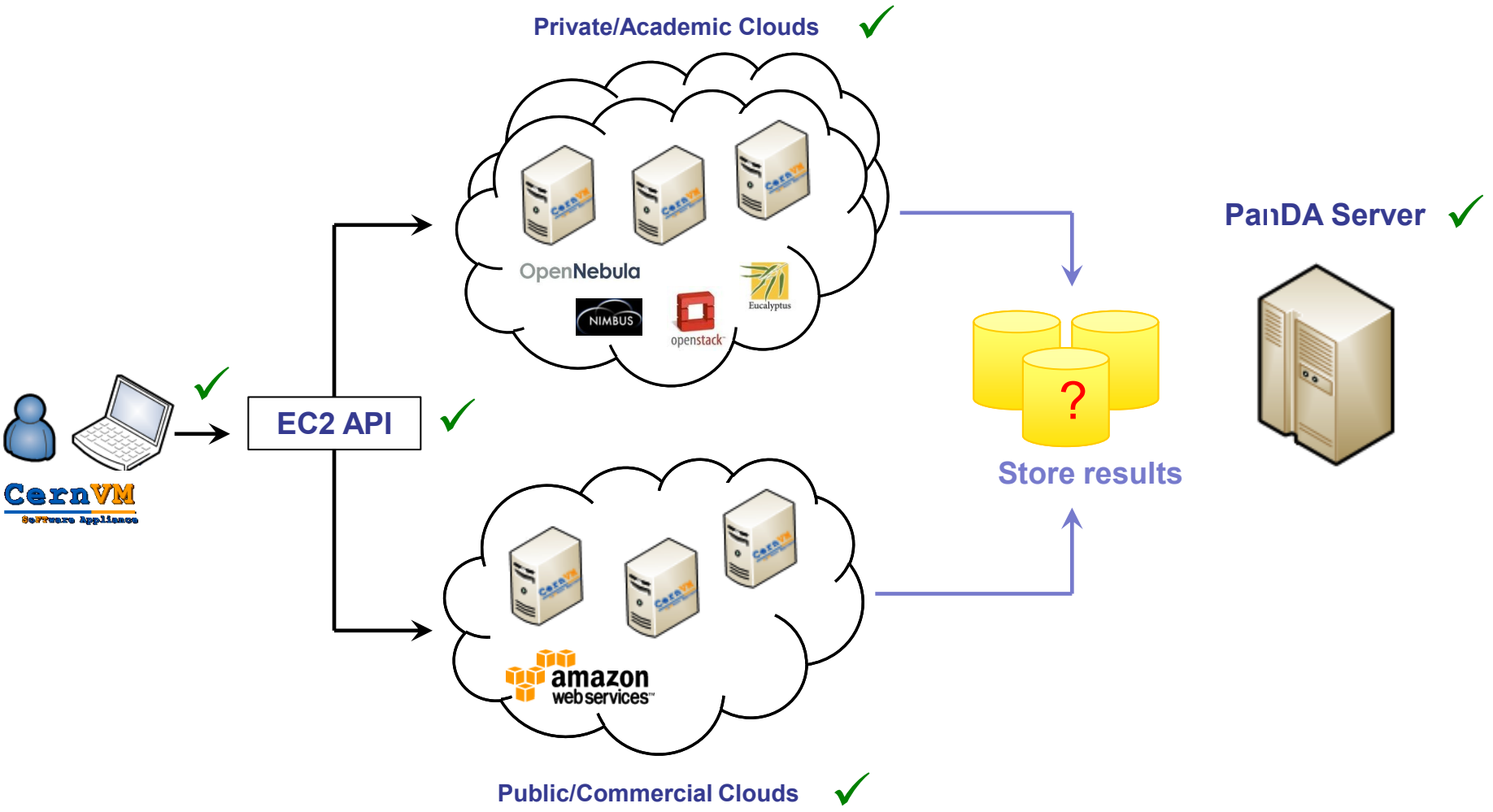
```
[pbuncic@localhost ~]$ cat panda-wn.tpl
[cernvm]
organisations = atlas
repositories  = atlas,grid,atlas-condb,sft
users = panda:atlas:
eos-server = eosatlas.cern.ch
contextualization_command =
panda:'/cvmfs/sft.cern.ch/lcg/external/experimental/panda-pilot/runPanda -m
/eos/atlas/vm/copilot -t 1'
[[IF X509_CERT]]x509-cert = [[X509_CERT]] [[ENDIF]]
```

CernVM
SoFTware Appliance

Now we have CernVM OS, FS, Ctx, API…

What's next?

Run PanDA in CernVM on the Clouds



EOS: Scalable Service Architecture

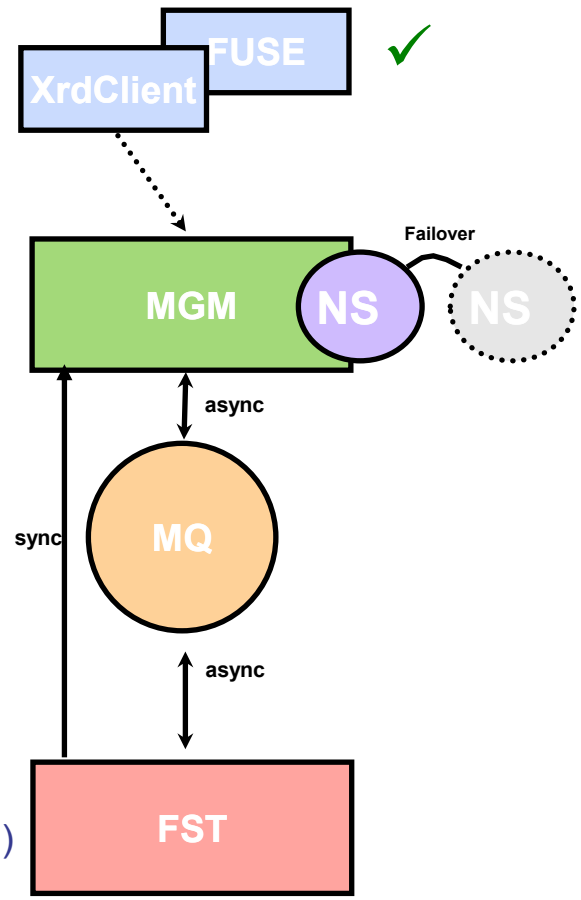
CernVM
SoFTware Appliance

Clients
XROOT client & FUSE
KRB5 + X509 authenticated

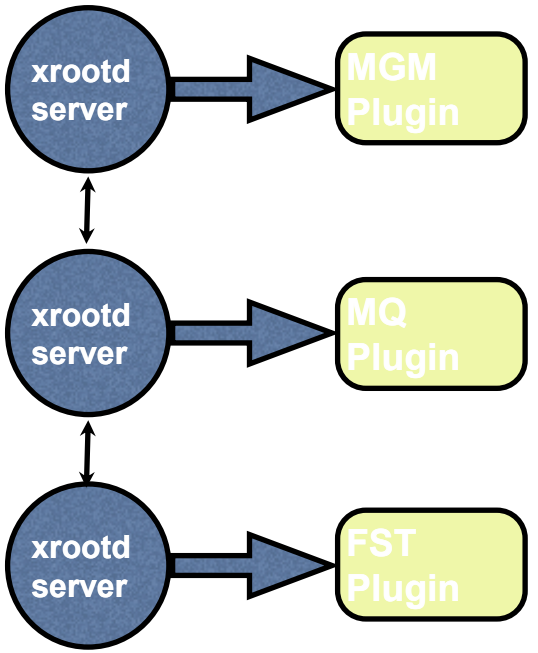
Management Server
Pluggable Namespace, Quota, ACLs,HA
Strong Authentication
Capability Engine
File Placement
File Location

Message Queue
Service State Messages
File Transaction Reports

File Storage
File & File Meta Data Store
Capability Authorization
File & Block Disk Error Detection (Scrubbing)
Layout Plugins [currently Raid-1(n)]



Implemented as plugins in xrootd



Andreas Peters IT/DM

```
[root@vmbsq090700 ~]# df /eos
Filesystem           1K-blocks      Used Available Use% Mounted on
eos                  3413376370848  23790724 3413352580124   1% /eos
```

CernVM
SoFTware Appliance

Using cernvm-tools

Listing available images:

```
[pbuncic@localhost ~]$ cvm --region CERN -H ls -i
AMI          LOCATION                              STATE     VISIBILITY ARCH TYPE
ami-00000008 hepix_sl55_x86_64_kvm                 available Public     i386 machine
ami-00000002 cernvm232_head_slc5_x86_64_kvm        available Public     i386 machine
ami-00000004 cernvm231_slc5_x86_64_kvm             available Public     i386 machine
ami-00000003 cernvm232_batch_slc5_x86_64_kvm       available Public     i386 machine
ami-00000010 lxdev_slc6_quattor_slc6_x86_64_kvm    available Public     i386 machine
```

Support for multiple regions:

```
[pbuncic@localhost ~]$ cvm --region EC2 -H ls -i ami-5c3ec235
AMI          LOCATION
STATE     VISIBILITY ARCH   TYPE
ami-5c3ec235 download.cernvm.cern.ch.s3.amazonaws.com/cernvm-2.3.1-
x86_64_1899.img.manifest.xml available Public     x86_64 machine
```

CernVM
SoFTware Appliance

Initializing credentials

Proxy certificate

```
[pbuncic@localhost ~]$ lcg grid-proxy-init
Your identity: /DC=ch/DC=cern/OU=computers/CN=pilot/copilot.cern.ch
Creating proxy ..................................................... Done
Your proxy is valid until: Thu May 19 06:43:50 2011

[pbuncic@localhost ~]$ lcg grid-proxy-info
subject  : /DC=ch/DC=cern/OU=computers/CN=pilot/copilot.cern.ch/CN=1766683191
issuer   : /DC=ch/DC=cern/OU=computers/CN=pilot/copilot.cern.ch
identity : /DC=ch/DC=cern/OU=computers/CN=pilot/copilot.cern.ch
type     : Proxy draft (pre-RFC) compliant impersonation proxy
strength : 512 bits
path     : /tmp/x509up_u500
timeleft : 11:59:56
```

- This proxy certificate is time limited and authorized only to
  - Request jobs from dedicated PanDA queue
  - Write (but not delete) files in a given EOS directory

CernVM
SoFTware Appliance

lxCloud

Starting contextualized CernVM images on lxCloud:

```
[pbuncic@localhost ~]$ cvm --region CERN run ami-00000003 --proxy --template panda-wn:1
r-47a5402e predrag  default i-195 ami-00000003 128.142.192.62 128.142.192.62 pending
default 0 m1.small 1970-01-01T01:00:00+01:00 default

[pbuncic@localhost ~]$ cvm --region CERN -H ls
ID    RID     OWNER   GROUP   DNS             STATE   KEY     TYPE
i-195 default predrag default 128.142.192.62 running default m1.small

[pbuncic@localhost ~]$ cvm --region CERN -H ls
ID    RID     OWNER   GROUP   DNS             STATE   KEY     TYPE
i-195 default predrag default 128.142.192.62 running default m1.small
i-196 default predrag default 128.142.192.63 running default m1.small
i-197 default predrag default 128.142.192.64 running default m1.small
i-198 default predrag default 128.142.192.65 running default m1.small
i-199 default predrag default 128.142.192.66 pending default m1.small
i-200 default predrag default 128.142.192.67 pending default m1.small
i-201 default predrag default 128.142.192.52 pending default m1.small
i-202 default predrag default 128.142.192.53 pending default m1.small
i-203 default predrag default 128.142.192.54 pending default m1.small
i-204 default predrag default 128.142.192.55 pending default m1.small
i-205 default predrag default 128.142.192.56 pending default m1.small
i-206 default predrag default 128.142.192.57 pending default m1.small
```

CernVM
SoFTware Appliance

Amazon EC2

Starting more contextualized CernVM images on EC2:

```
[pbuncic@localhost ~]$ cvm run ami-5c3ec235 -g default -t m1.large --kernel aki-9800e5f1 -
-key ami --proxy --template panda-wn:10
r-ad962dc1 392941794136  default i-f3b04a9d ami-5c3ec235   pending ami 0 m1.large
r-ad962dc1 392941794136  default i-f1b04a9f ami-5c3ec235   pending ami 1 m1.large
r-ad962dc1 392941794136  default i-cfb04aa1 ami-5c3ec235   pending ami 2 m1.large
r-ad962dc1 392941794136  default i-cdb04aa3 ami-5c3ec235   pending ami 3 m1.large
....

[pbuncic@localhost ~]$ cvm --region EC2 -H ls
ID         RID         OWNER        GROUP   DNS                  STATE   KEY TYPE
i-f3b04a9d r-ad962dc1 392941794136 default ec2-50-16-144-41   running ami m1.large
i-f1b04a9f r-ad962dc1 392941794136 default ec2-75-101-214-247 running ami m1.large
i-cfb04aa1 r-ad962dc1 392941794136 default ec2-184-72-183-26  running ami m1.large
i-cdb04aa3 r-ad962dc1 392941794136 default ec2-184-73-56-72   running ami m1.large
i-cbb04aa5 r-ad962dc1 392941794136 default ec2-50-16-32-51    running ami m1.large
i-c9b04aa7 r-ad962dc1 392941794136 default ec2-75-101-184-46  running ami m1.large
i-c7b04aa9 r-ad962dc1 392941794136 default ec2-50-19-38-225   running ami m1.large
i-c5b04aab r-ad962dc1 392941794136 default ec2-50-16-105-241  running ami m1.large
i-c3b04aad r-ad962dc1 392941794136 default ec2-174-129-86-61  running ami m1.large
i-c1b04aaf r-ad962dc1 392941794136 default ec2-50-19-9-47     running ami m1.large
```

CernVM
SoFTware Appliance

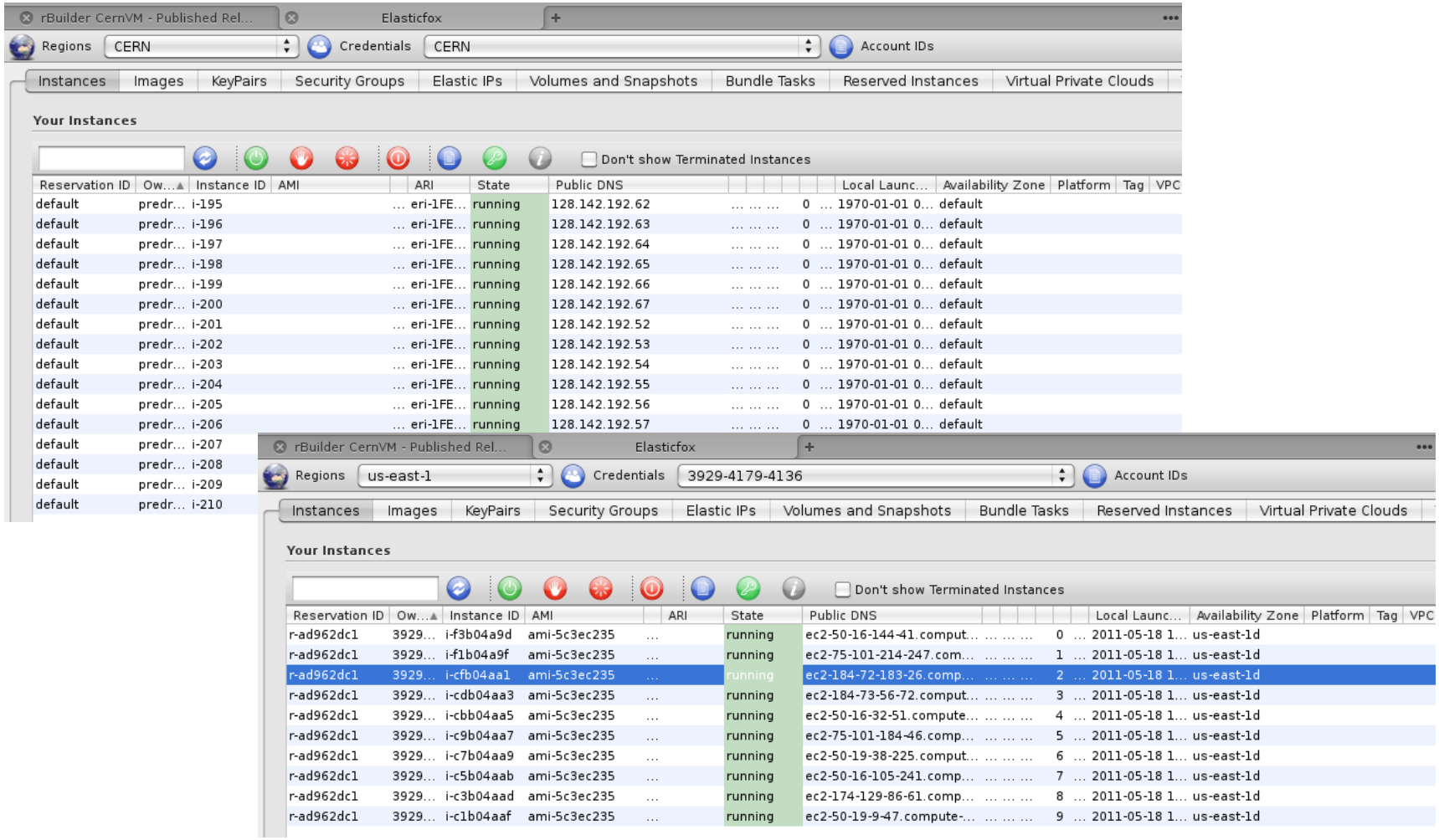
PanDA Monitor

Jobs:

PandaID, Owner, Working group	Job	Status	Created	Time to start	Duration	Ended/ Modified	Cloud/Site, Type	Priority
1237433059 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:47	1 day, 15:31:21	0:00:34	05-18 13:18	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.35311e2f-8899-483e-9830-7f095077949a							
1237433058 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:47	1 day, 15:28:03	0:03:52	05-18 13:15	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.8bf86120-e688-494f-b20d-9a5dbbf830a0							
1237433057 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:47	1 day, 15:26:30	0:05:26	05-18 13:13	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.20ef9979-d794-46d8-b34c-a953aad188fb							
1237433056 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:47	1 day, 15:26:01	0:05:55	05-18 13:13	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.da608a8f-aa70-4226-8cb1-db7a200f174d							
1237433055 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:47	1 day, 15:25:35	0:06:22	05-18 13:12	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.644be014-14f0-4b1d-a631-f3e41aa8c78d							
1237433054 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:47	1 day, 15:25:10	0:06:47	05-18 13:12	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.e3ffff3d-7769-4d43-9ff9-7018eaee6c25							
1237433045 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:46	1 day, 15:25:19	0:06:57	05-18 13:12	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.22b7d89d-5498-4434-a61c-a843d28fcd7b							
1237433043 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:46	1 day, 15:24:50	0:07:34	05-18 13:11	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.d217d914-c77c-4c98-b339-709511e5e5cd							
1237433042 pilot/copilot.cern.ch	trans=csc_evgen_trf.py, pkg=AtlasProduction/15.6.5.5	running	2011-05-16 21:46	1 day, 15:23:49	0:08:36	05-18 13:10	US/CERN.CERNVM, ptest	100
	Out: panda.destDB.7572d85d-2873-4315-a4bb-c78616bc4eac							

CernVM
SoFTware Appliance

Elasticfox



CernVM
SoFTware Appliance

Problems

No external connectivity to eosatlas.cern.ch (had to use local disk):

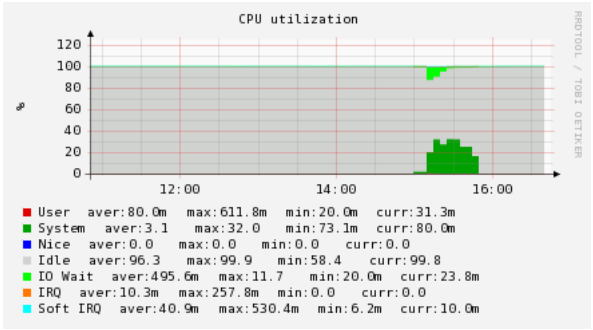
```
18 May 13:33:44| pilot.py    | Collecting WN info from: /tmp/panda-
pilot/3158/WORK/Panda_Pilot_3994_1305725624 (again)
18 May 13:33:44| pilot.py    | Job memory set by queuedata to:
18 May 13:33:44| pilot.py    | Local space limit: 5368709120 B
18 May 13:33:44| pilot.py    | !!FAILED!!1999!! Too little space left on local disk to run
job: 1967128576 B (need > 5368709120 B)
18 May 13:33:44| pilot.py    | Pilot was executed on host: ip-10-212-183-96
....
```

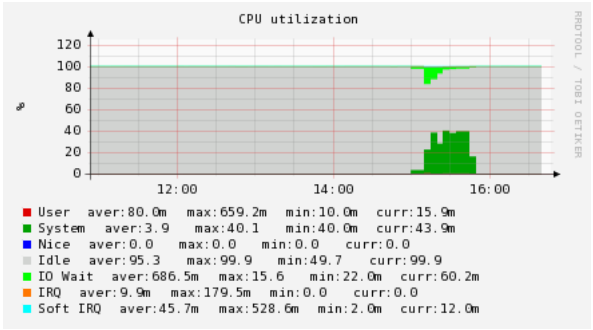
EOD storage worked well on lxCloud:

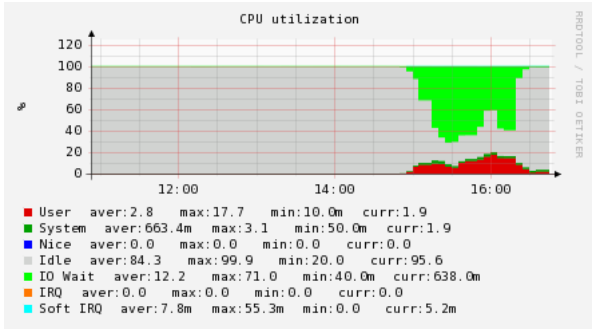
```
[root@vmbsq090700 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1             13577676   4131152   8762240  33% /
none                    513176         0    513176   0% /dev/shm
eos                 3413376370848  23790724 3413352580124   1% /eos

[panda@vmbsq090700 copilot]$ ls -1 *.root
024653d9-99f6-49e5-8dad-0b826f84fa3b_0.evgen.pool.root
02746a4b-ab5e-48a2-b5ff-4359f22b5da7_0.evgen.pool.root
0665543c-359a-4265-84ec-82ffdab2d8dd_0.evgen.pool.root
09719f1b-ae99-441f-9ee5-7fc265f099b4_0.evgen.pool.root
107acb02-7f9b-4b4d-bdaf-dc9b86c58dca_0.evgen.pool.root
```

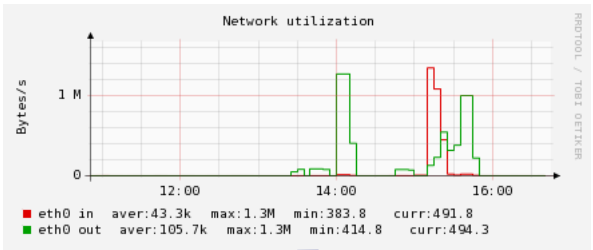
CernVM
SoFTware Appliance

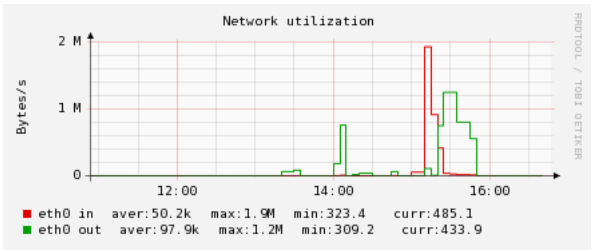
SLC5 vs SLC6

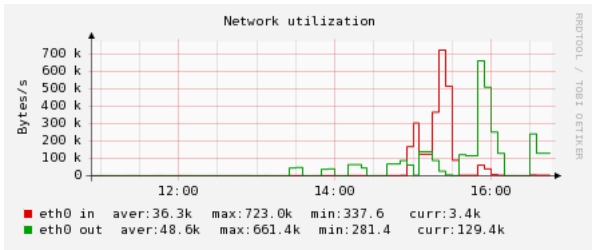












SLC5 hypervisor

SLC6 hypervisor

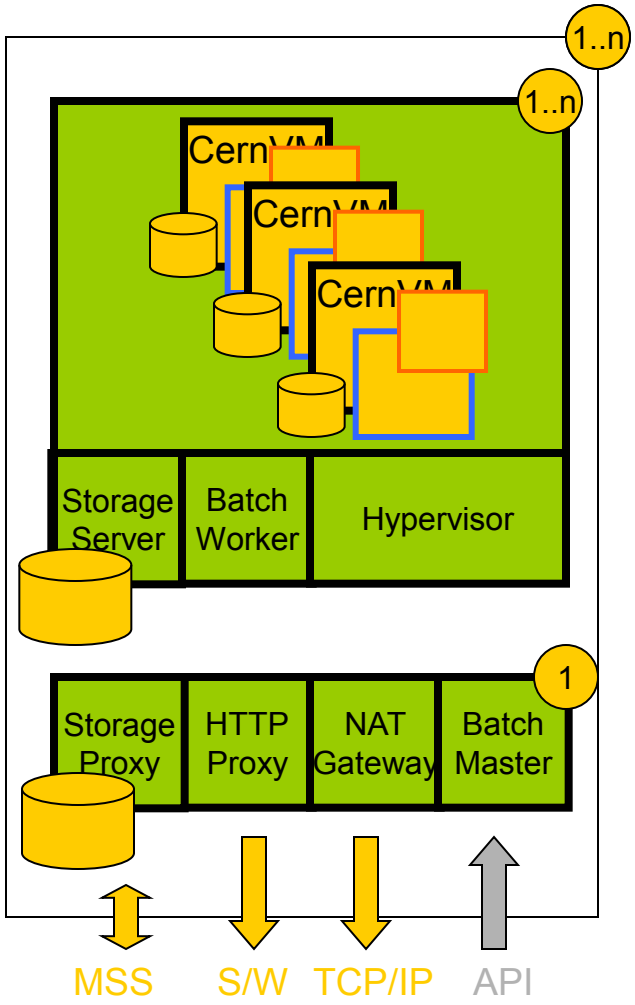
CernVM

SoFTware Appliance

Work in progress, blue sky ideas…

CernVM
SoFTware Appliance

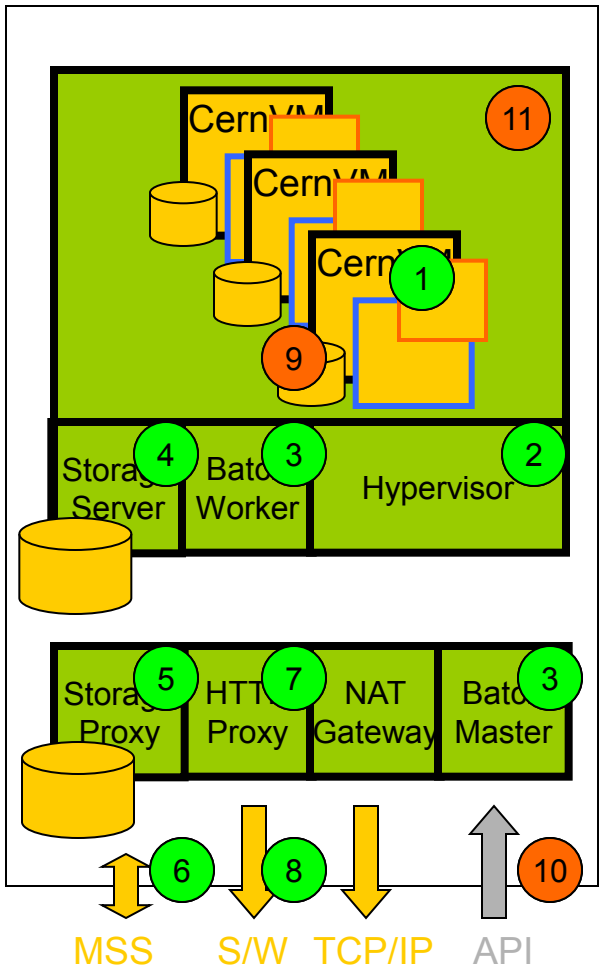
Computer Center in a Box



- Common services hosted by front-end node
  - Batch master, NAT gateway, storage and HTTP proxy
- Each physical node
  - Contributes to common storage pool
  - Runs hypervisor, batch worker
  - Exports storage local storage to common pool
- Virtual Machines
  - Started by the suitable Cloud middleware
  - Only limited outgoing network connectivity via gateway node/HTTP proxy
  - Access to data files via POSIX (file system) layer
  - Software delivered to VMs from Web server repository
  - Built form recipes and components stored in strongly versioned repository
- Access to external mass storage via storage proxy
- End user API to submit jobs

CernVM
SoFTware Appliance

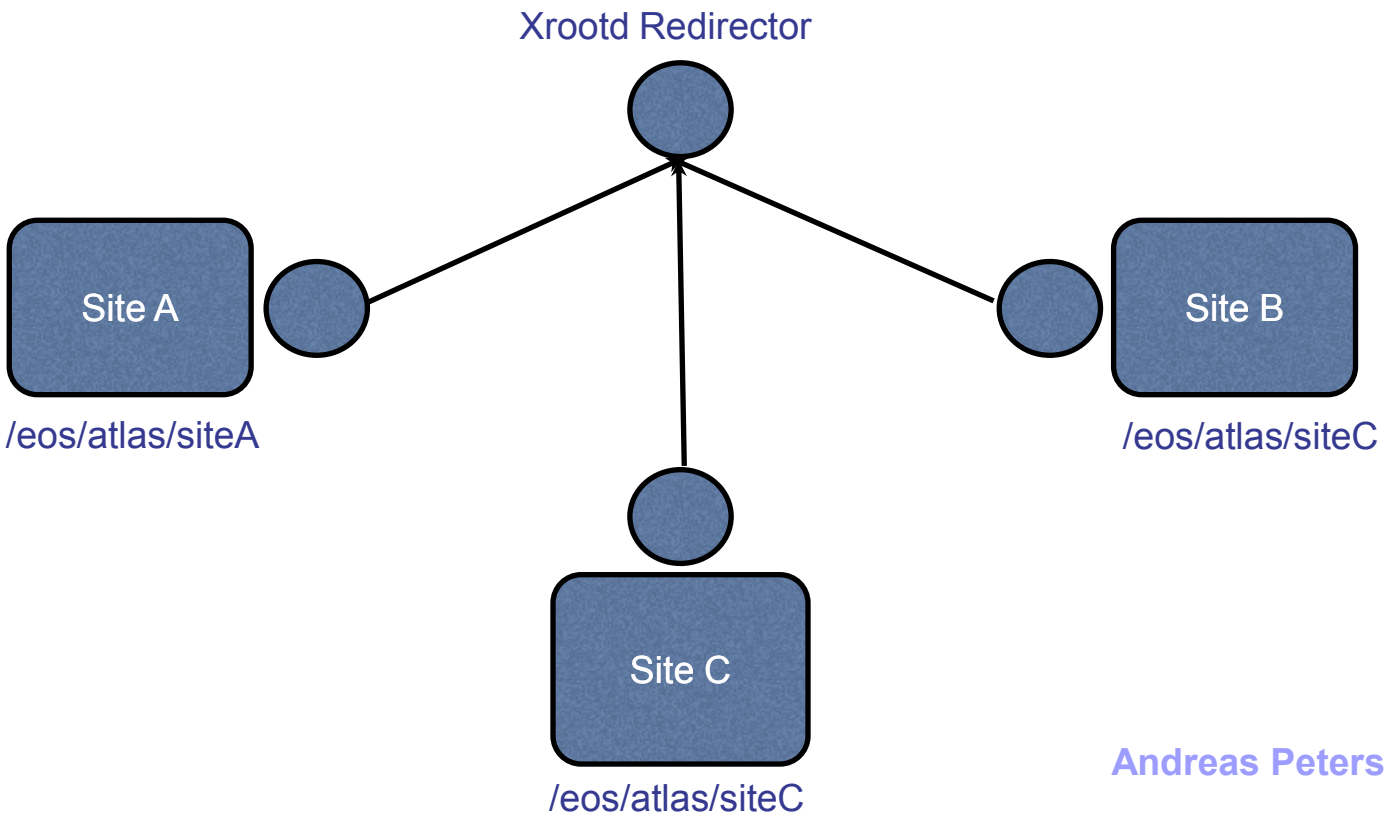
NA61 Testbed (long term data preservation use case)



1) CernVM for NA61
2) KVM - Linux native hypervisor
3) Condor - batch system that works well in dynamic environment and supports running jobs in VMs
4) Xrootd server - high performance file server, used by LHC experiments and distributed with ROOT
5) Xrootd redirector - each aggregates up to 64 servers, can cluster up to 64k servers
6) Xrootd supports access to MSS systems using SRM extension
7) Standard HTTP proxy (Squid)
8) CernVM-FS repository for software distribution
9) eosfs - POSIX File System for Xrootd
10) GANGA - user interface and front-end to various batch systems and Grid
11) Ganglia - monitoring

CernVM
SoFTware Appliance

Scaling EOS across the sites: Storage Cloud?



Andreas Peters IT/DM

☞ Storage Cloud plays fundamental role in Cloud concept, cannot be factored out as separate problem and solved later…

CernVM
SoFTware Appliance

In the pipeline…

- DIRAC Pilot for LHCb…
  - and other experiments that use pilot job frameworks
- On demand Condor cluster
  - contextualization templates currently under testing
- PROOF on demand
- GridFactory scheduler
  - Ad-hoc cloud for small workgroups
- Volunteer computing using BOINC adapted to run jobs in CernVM
  - Ongoing activity with Theory Group at CERN
  - MonteCarlo validation project

CernVM
SoFTware Appliance

Conclusions

- In combination with various **contextualization** options and **CernVM-FS**, just **one small image** can run frameworks of all LHC experiments and be **easily moved around** requiring far less frequent updates than traditional worker node
- **CernVM-FS** provides efficient, scalable, secure and maintenance free way do distribute software in CernVM and physical nodes alike
- **Flexible contextualization options** allow the same image to play different roles reducing the need for creation and certification of specialized images
- **Storage Cloud** plays fundamental role in Cloud concept, cannot be factored out as separate problem and solved later
- **lxCloud** is operational and usable, can run PanDA in CernVM with EOS as storage backend (thanks to Sebastian Goasguen and Ulrich Schwickerath)
  - Time to do some stress testing…