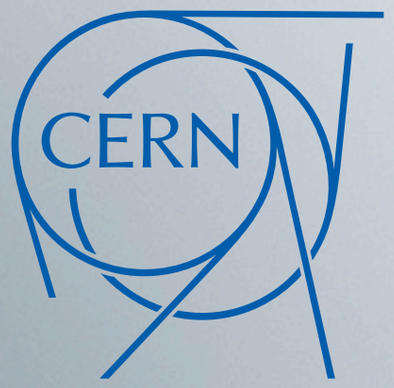
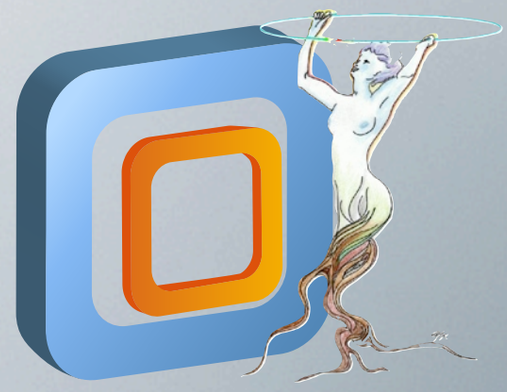




PROOF as a Service on the Cloud

a Virtual Analysis Facility based on the CernVM ecosystem

Dario Berzano

R.Meusel, G.Lestaris, I.Charalampidis, G.Ganis, P.Buncic, J.Blomer

CERN PH-SFT

CHEP2013 - Amsterdam, 15.10.2013

A cloud-aware analysis facility

Distributed computing on the Cloud

- Resources are **heterogeneous**
  → *Like the Grid but on a more generic level*
- Virtual machines are **disposable**
  → *Cattle, not puppies: http://bit.ly/cattlecloud*

Building a cloud aware application

- **Scale** quickly when resources vary
  → *No prior pinning of data to process to the workers*
- Deal **smoothly** with major failures
  → *Automatic **failover** and clear recovery procedures*

Usual Grid workflow → static job pre-splitting ≠ cloud-aware

A cloud-aware analysis facility

admins provide virtual clusters

user's workflow does not change

geographically distributed independent cloud providers

Virtual Analysis Facility → ***analysis cluster*** *on the cloud in one click*

PROOF is cloud-aware

PROOF: the Parallel ROOT Facility

- Based on unique advanced features of ROOT
- Event-based parallelism
- Automatic merging and display of results
- Runs on batch systems and Grid with PROOF on Demand

PROOF is interactive

- Constant control and feedback of attached resources
- Data is not preassigned to the workers → *pull scheduler*
- Workers dynamically attached to a running process

Interactivity is what makes PROOF cloud-aware

PROOF is cloud-aware

***PROOF on Demand**: runs PROOF on top of batch systems*

- Zero configuration
  Per-user, no system-wide installation
- Stability
  Potential failures affect only one user
- Self-servicing
  User can restart her PROOF server
- Advanced scheduling
  Uses scheduling

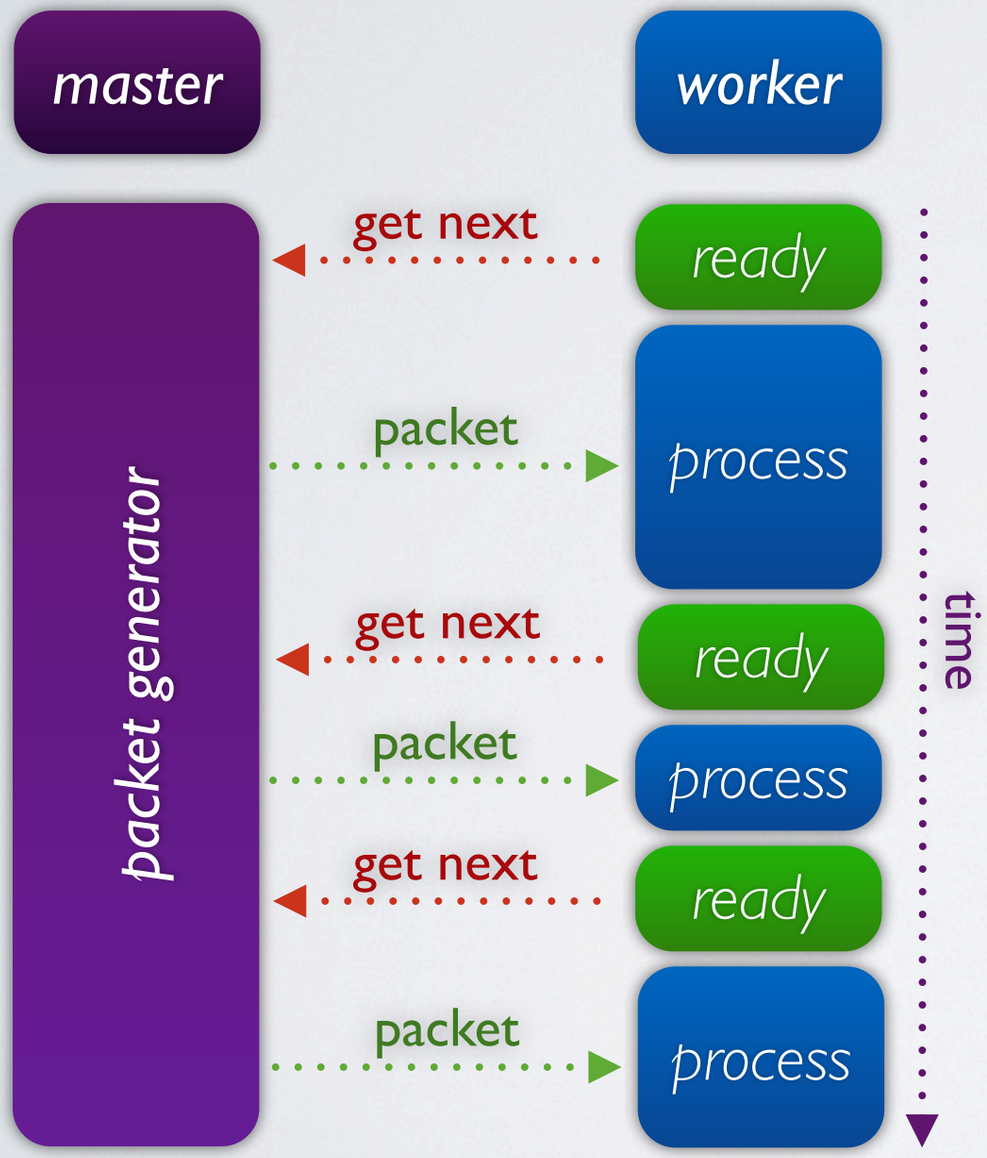
```
$> pod-server start
Starting PoD server...
updating xproofd configuration file...
starting xproofd...
starting PoD agent...
preparing PoD worker package...
select user defined environment script
selecting pre-compiled bins to be added
PoD worker package: /Users/volpe/.PoD/
-----------------------
XPROOFD [1675] port: 21001
PoD agent [1680] port: 22001
PROOF connection string: proof@pcphsft
-----------------------

$> pod-submit -r condor -q proof -n 10
Job ID: 12345
```

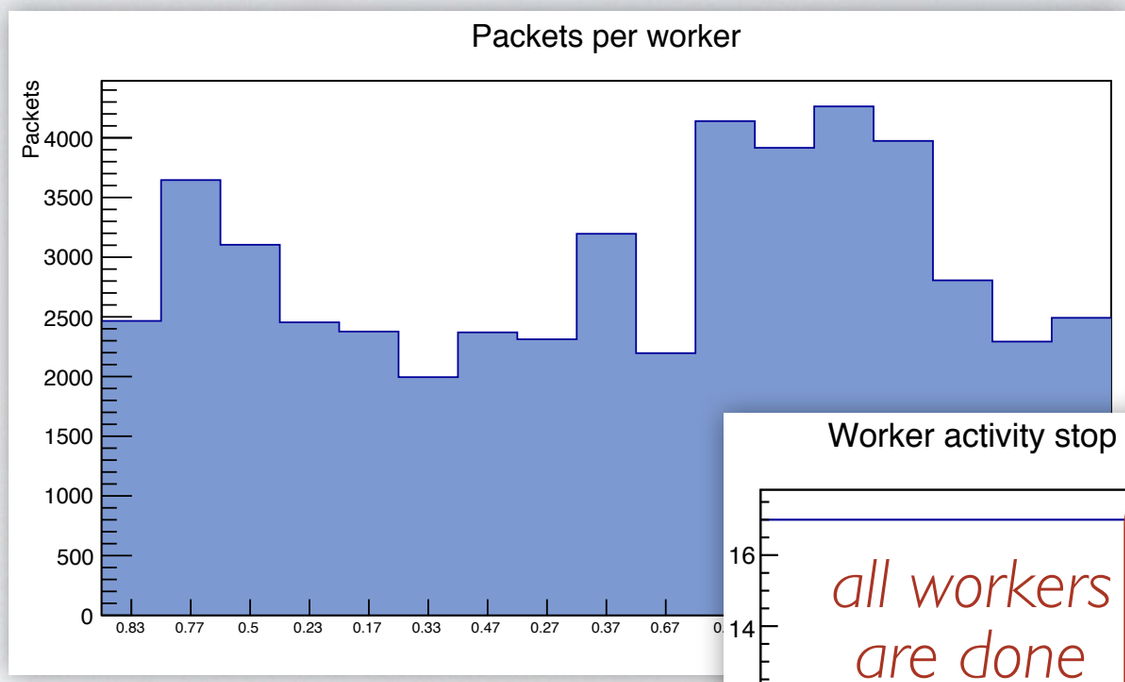
PoD PROOF on Demand pod.gsi.de

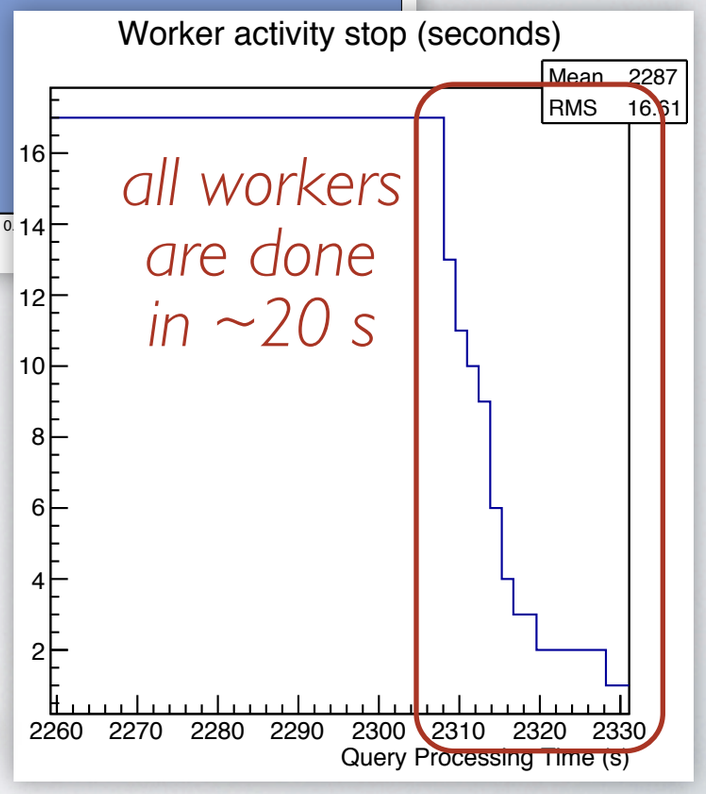
PROOF is cloud-aware

Adaptive workload: *workers ask for data to process when ready*



nonuniform workload distribution



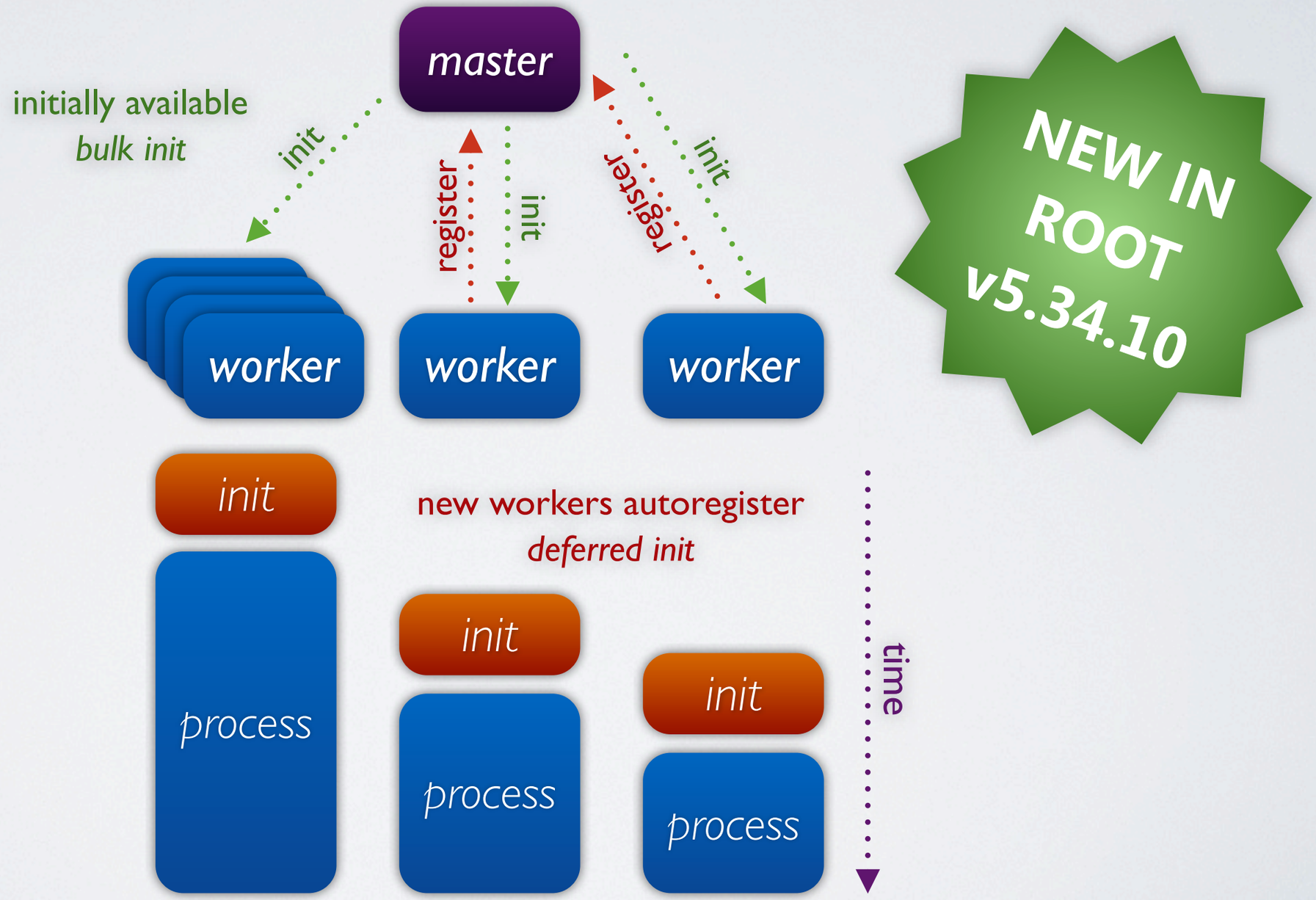


uniform completion time

PROOF is cloud-aware

Dynamic addition of workers

new workers can join and offload a running process



PROOF dynamic workers

User requests N workers

Old Workflow

Wait until "some" workers are ready

New Workflow

Wait until at least one worker becomes available

A bunch of workers is started

Run the full analysis on such workers only

Run the analysis

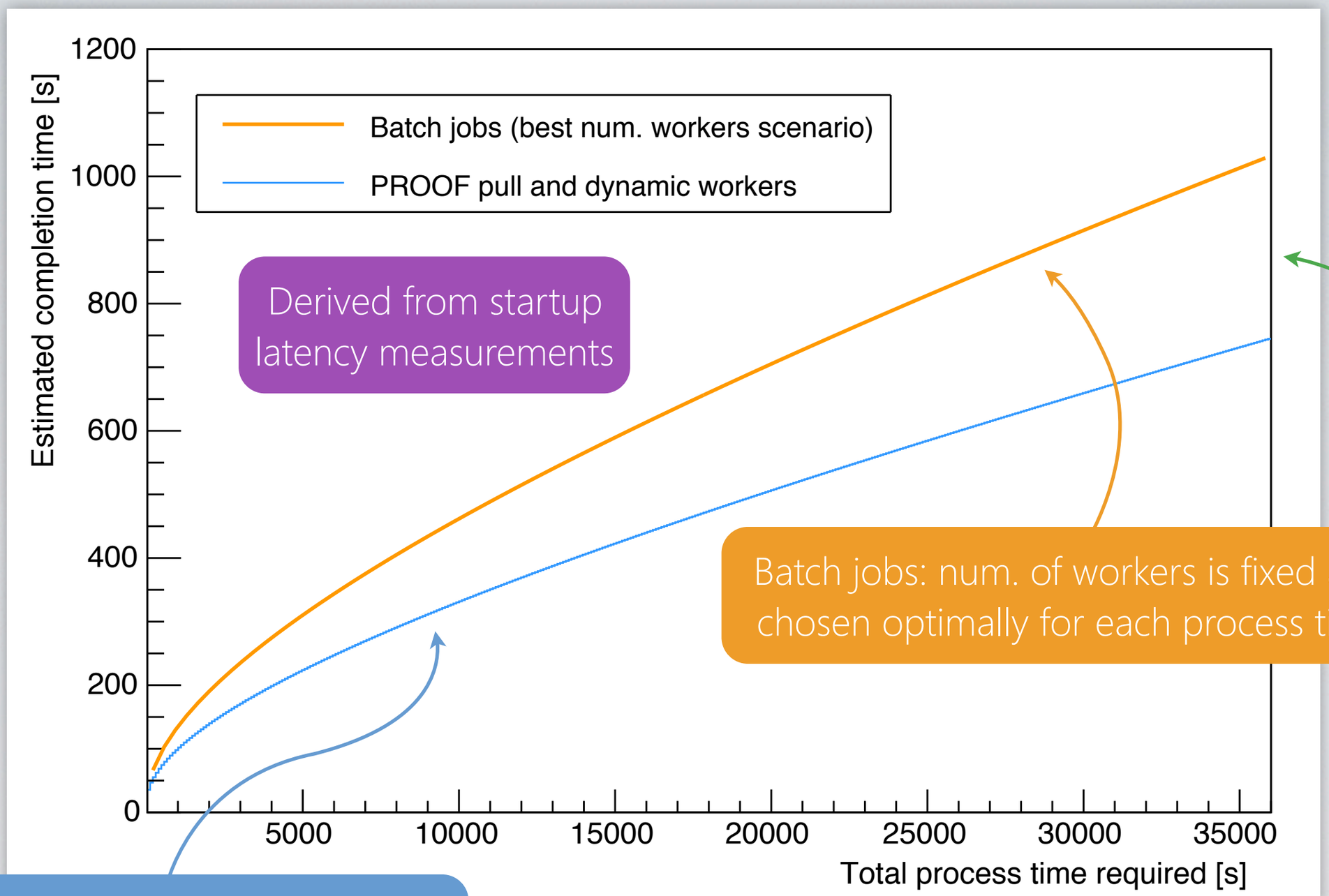
Other workers will gradually become available

They will be available at next run only

Additional workers join the processing

Minimal latency and optimal resources usage
*See **ATLAS use case** here: http://chep2013.org/contrib/256*

PROOF dynamic workers



Derived from startup latency measurements

Batch jobs: num. of workers is fixed and chosen optimally for each process time

Dynamic workers are never idle: uptime always spent in computing

Dynamic workers reduce completion time of a constant ~30%

The virtual analysis facility

PROOF	PoD	µCernVM	Elastiq
HTCondor	CVM online	CernVM-FS	authn/authz

- **What:** a cluster of µCernVMs with HTCondor
  → *One head node plus a scalable number of workers*

- **How:** contextualization configured on the Web
  → *Simple web interface: http://cernvm-online.cern.ch*

- **Who:** so easy that can be spawned by end users
  → *You can have your **personal analysis facility***

- **When:** scales up/down automatically whenever it's needed
  → *Frees unused resources*

The virtual analysis facility

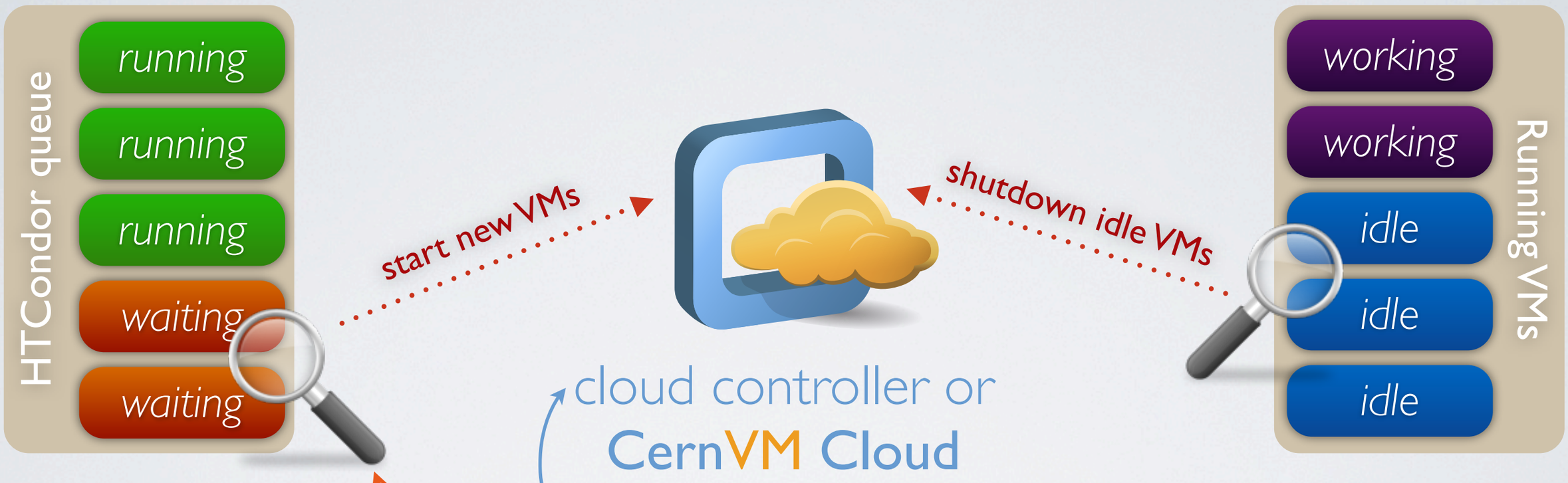
Integration to the CernVM ecosystem and HTCondor

- **µCernVM:** one-flavor SLC6 operating system on demand
  → *See previous talk: http://chep2013.org/contrib/213*
- **CernVM-FS:** HTTP-based cached FUSE filesystem
  → *Both OS and experiments software downloaded on demand*
- **CernVM Online:** safe context GUI and repository
  → *See previous talk: http://chep2013.org/contrib/185*
- **HTCondor:** light and stable workload management system
  → *Workers auto-register to head node: no static resources configuration*

*The full stack of components is **cloud-aware***

Elastiq queue monitor

Python app to monitor HTCondor and scale up or down



cloud controller or
CernVM Cloud

Jobs waiting too long will trigger a scale up

EC2 interface: credentials given in the context

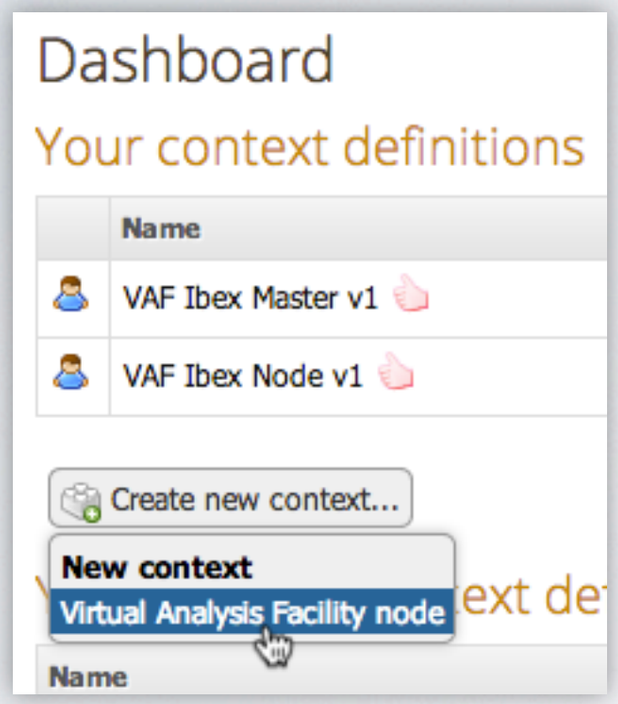
Experimental meta cloud controller
- Accepts scale requests
- Translates them to multiple clouds

Can be used on any HTCondor cluster

Code available at
http://bit.ly/elastiq

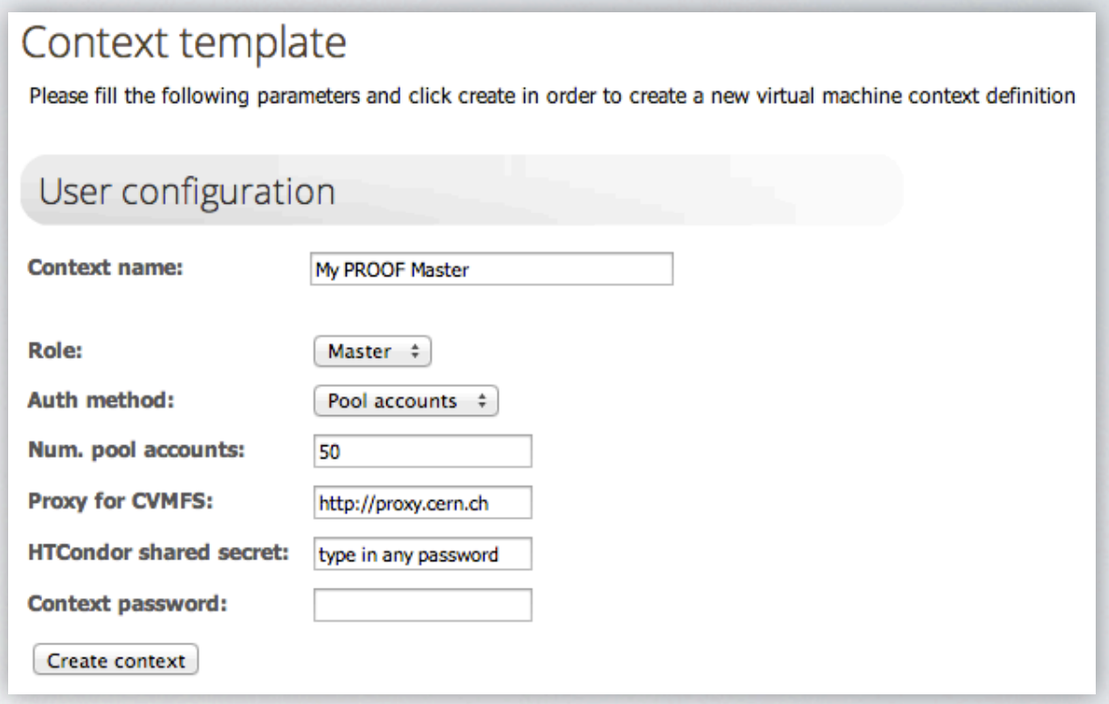
Elastic cloud computing in action

Context creation with CernVM Online: http://cernvm-online.cern.ch

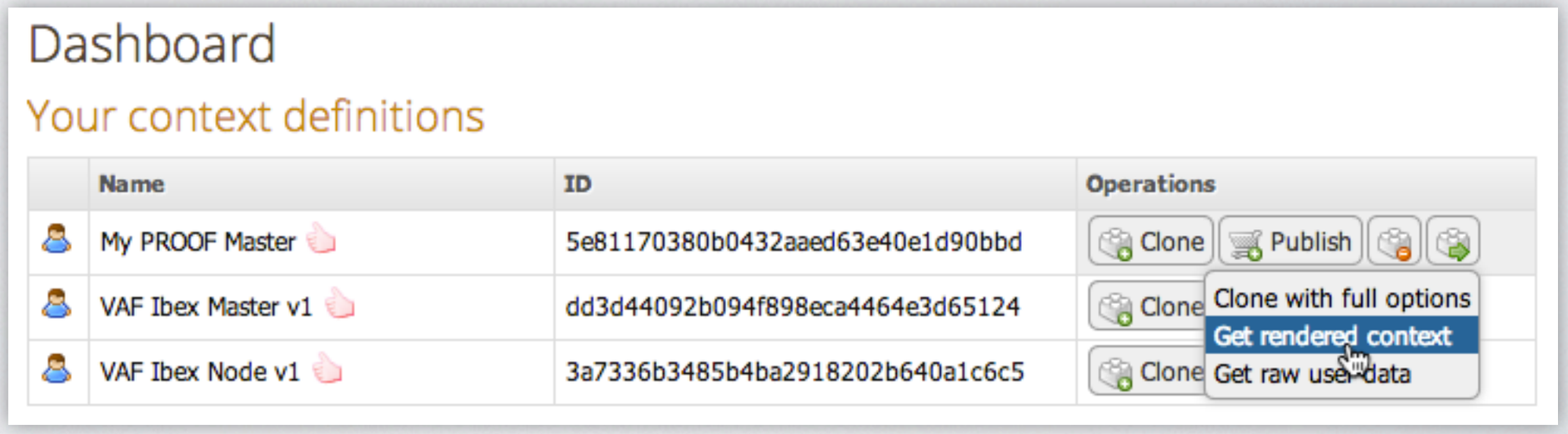


Create new special context

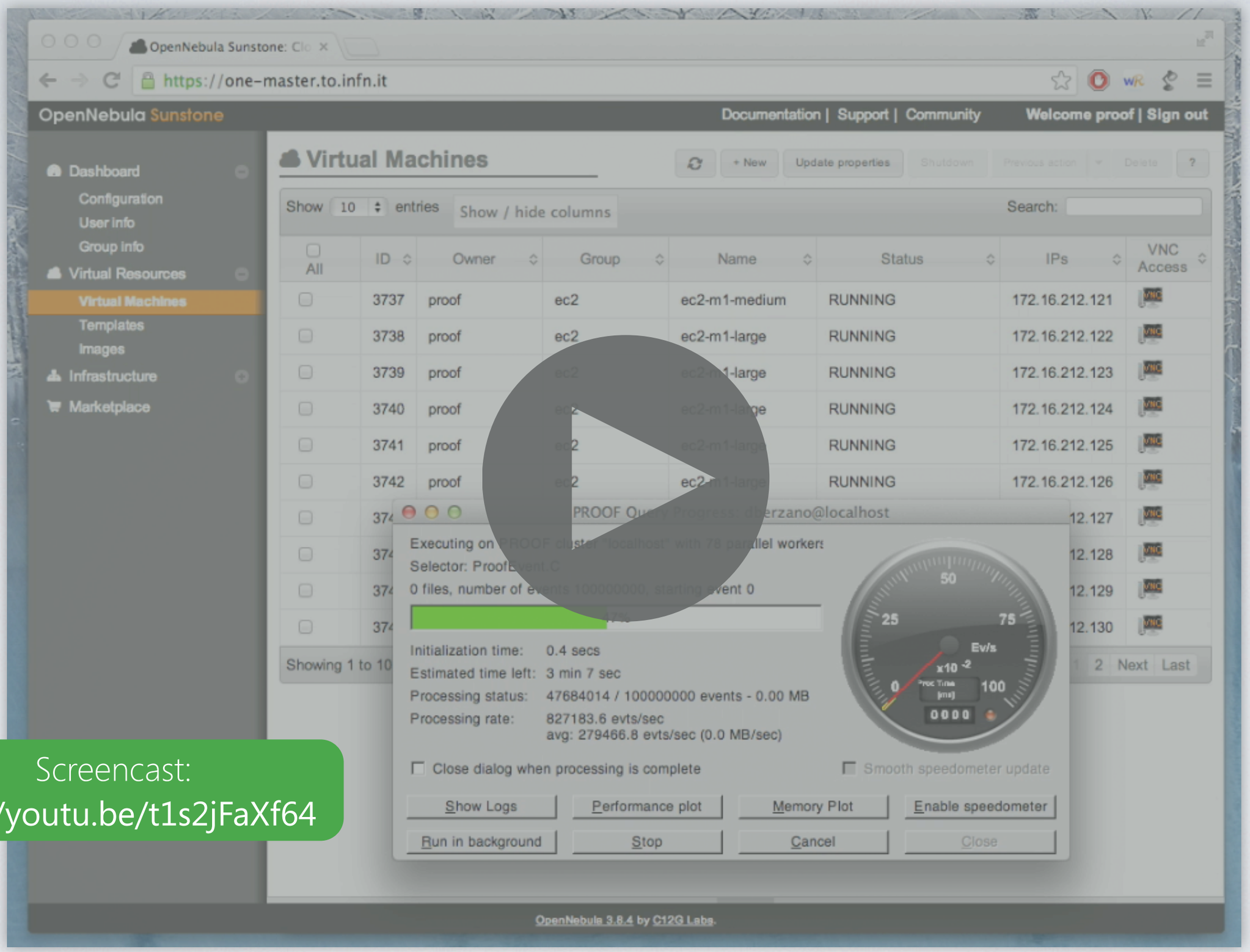
Customize a few options



Get generated user-data



Elastic cloud computing in action



Screencast:
http://youtu.be/t1s2jFaXf64

µCernVM+PROOF startup latency

Measured the delay before requested resources become available

Target clouds:

- Small: OpenNebula @ INFN Torino
- Large: OpenStack @ CERN (Agile)

Note: not comparing cloud infrastructures. Only measuring **µCernVM+PROOF latencies**.

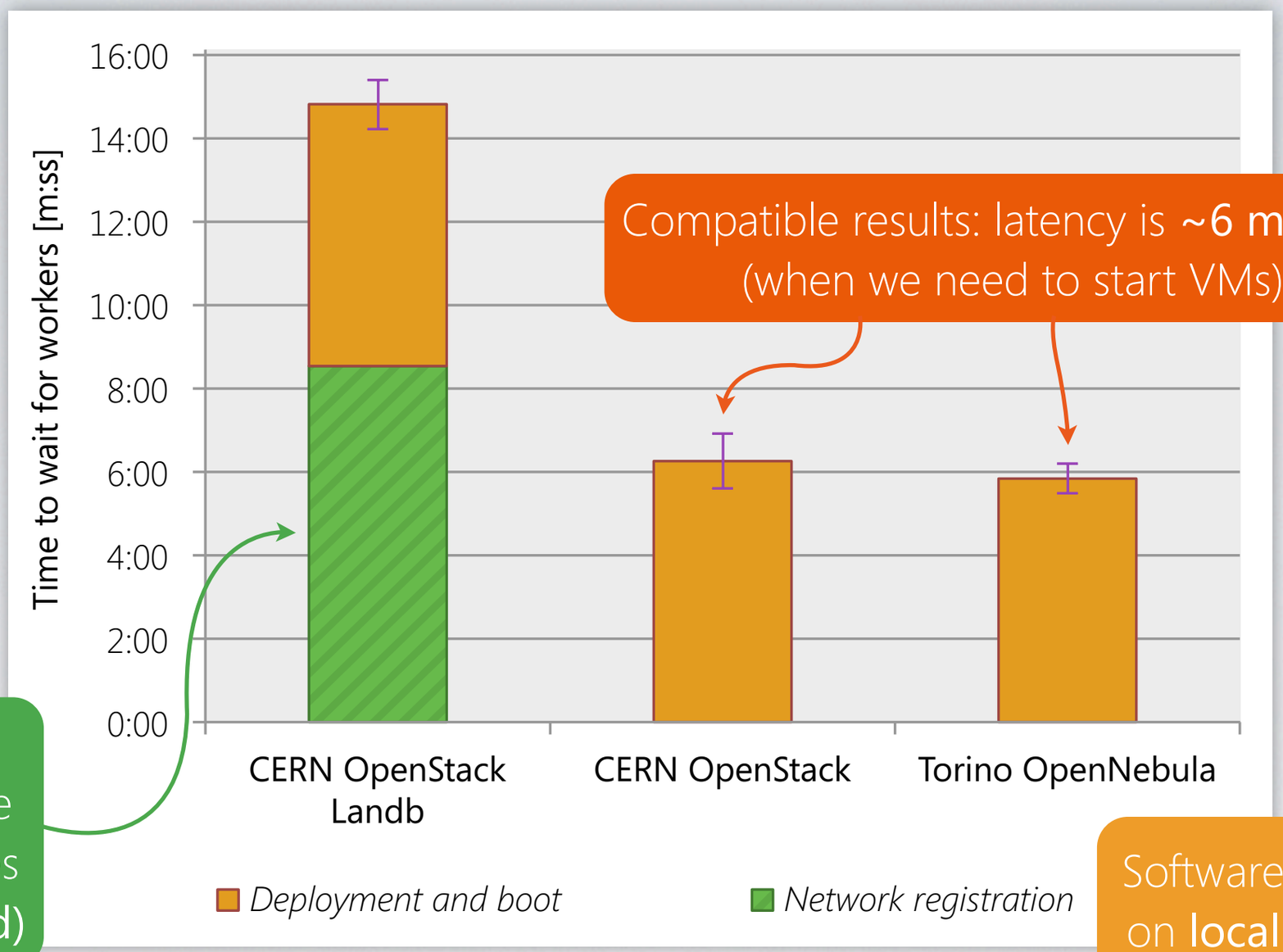
Test conditions:

- µCernVM use a HTTP caching proxy
  → *Precaching via a dummy boot*
- µCernVM image is 12 MB big
  → *image transfer time negligible*
- VMs deployed when resources are available
  → *rule out failures and delay due to lack of resources*

Measuring latency due to:

- µCernVM **boot** time
- HTCondor **automatic registration** of new nodes
- PoD and PROOF **reaction time**

µCernVM+PROOF startup latency



Compatible results: latency is **~6 minutes** (when we need to start VMs)

Latency due to legacy hostname registration issues **(this test ignored)**

Software and OS **precached** on **local Squid proxies** by a previous test run

Measured time elapsed between PoD worker's request and availability: **pod-info -l**

10 VMs started in the test

Conclusions

Every VAF layer is cloud-aware and elastic

- PROOF+HTCondor deal with dynamic addition/removal of workers
- µCernVM is very small and fast to deploy
- CernVM-FS downloads only what is needed

Consistent configuration of solid and independent components

- No login to configure: all done via CernVM Online context
- PROOF+PoD also work dynamically on the Grid
- Elastiq can scale any HTCondor cluster, not PROOF-specific
- Reused existing components wherever possible

Thank you for your attention!

References

- PROOF (the Parallel ROOT Framework)
  http://root.cern.ch/drupal/content/proof

- Virtual Analysis Facility client and Elastiq
  https://github.com/dberzano/virtual-analysis-facility

- The CernVM Ecosystem
  http://cernvm.cern.ch/portal/publications

- Cloud @ INFN Torino
  http://chep2013.org/contrib/474

- CERN Agile Infrastructure
  http://chep2013.org/contrib/86