

Lessons learned from large scale LSF scalability tests

Sebastien Goasguen, Belmiro Rodrigues Moreira,
Ewan Roche, Ulrich Schwickerath, Romain Wartel

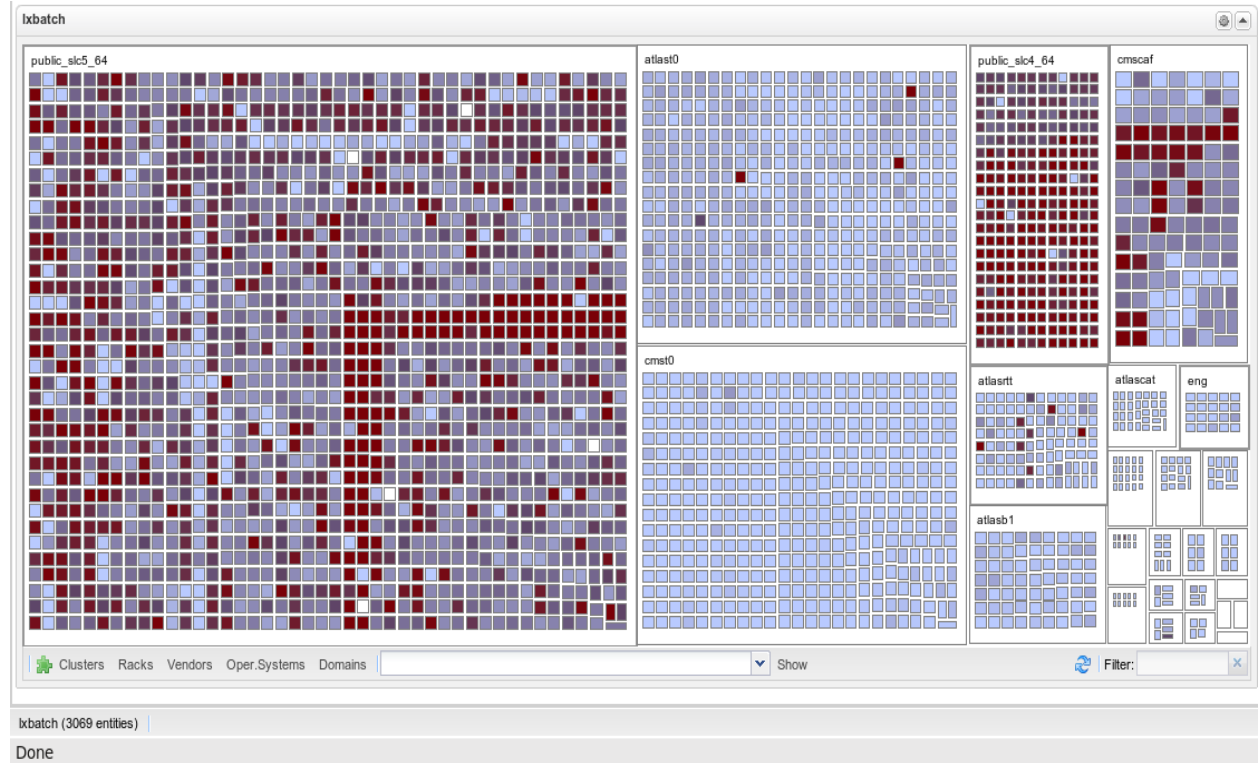
HEPIX2010, Cornell University

- ▶ Why we did it
- ▶ How we did it
- ▶ Lessons learned:
 - ▶ While doing it
 - ▶ From the actual measurements

Facts about the CERN batch farm Ixbatch

Size:

- ~ 3,500 physical hosts
- ~ 40 hardware types
- ~ 25,000 CPU cores
- ~ 80 processing queues
- ~ 180,000 jobs/day
- ~ 22,000 jobs total per time
- ~ 20,000 registered users



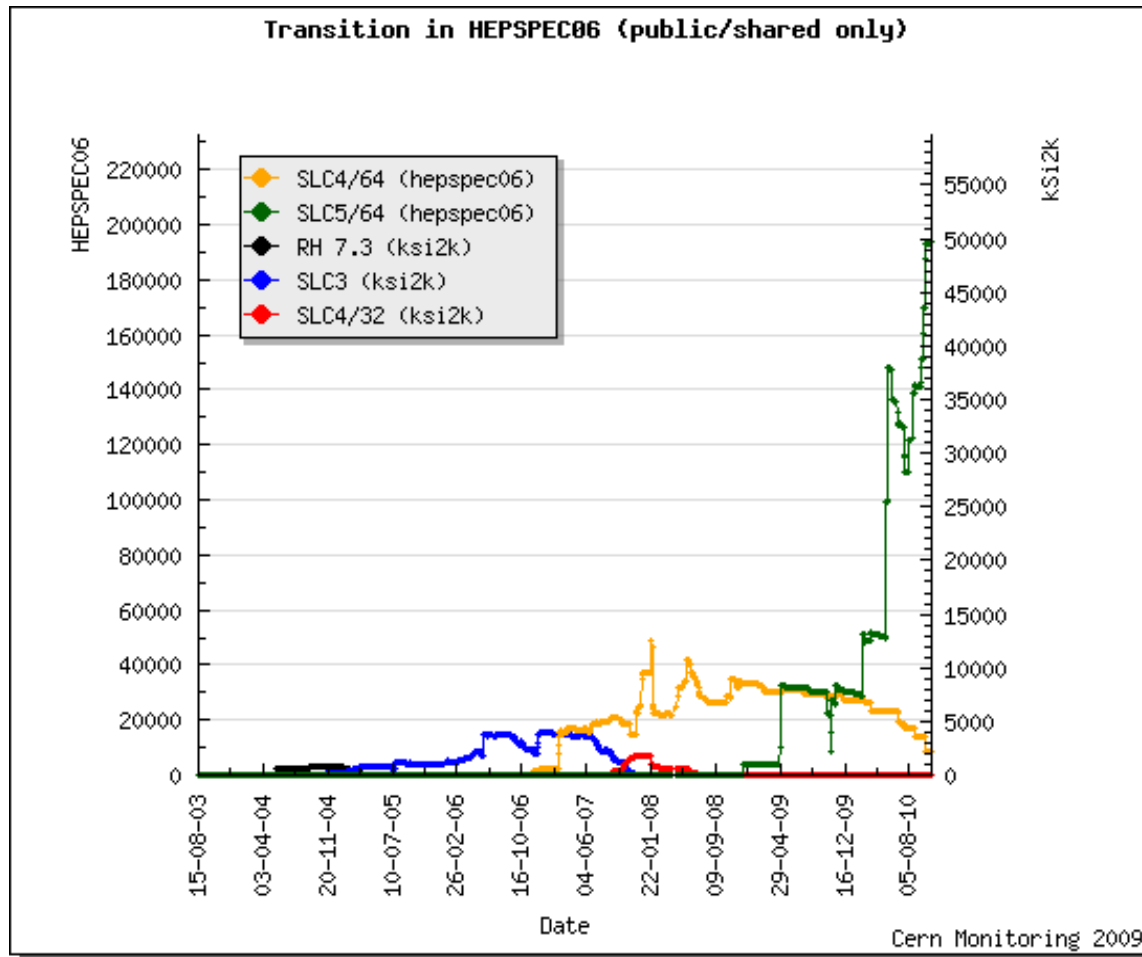
User jobs:

- ◆ Chaotic and unpredictable mixture of CPU or I/O intensive jobs
- ◆ mostly independent sequential jobs, few multi-threaded

Currently supported OS systems: SLC5 (64bit) and SLC4 (32/64bit)

Why scalability tests ?

Growth rate over the past years



▶ >50% dedicated

▶ One LSF instance

LSF limits

- ▶ Vendor information (Platform priv. Comm.) (*):
 - ▶ Communication layer ~20,000 hosts
 - ▶ Batch system layer ~5,000-6,000 hosts
 - ▶ Up to ~500,000 jobs
- ▶ Previous LSF scalability tests at CERN
 - ▶ Focus on batch queries and jobs
 - ▶ Focus on fail-over system layout
 - ▶ Limitations on WN not testable

(*) mostly based on simulations

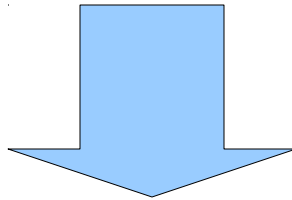
- ▶ We already start to see **slower response** times at our current scale when there are many concurrent queries
- ▶ **Virtualizing the batch worker nodes** has a big impact on the number of resources the batch system needs to manage
 - 8:1 ratio now, even more in the future?
- ▶ The use of virtualization enabled us to test LSF up to a larger scale than ever before
 - up to 16,000 worker nodes

How we did it

Temporary access to 480 new physical worker nodes

- ▶ 2 x XEON L5520 @ 2.27GHz CPUs (2 x 4 cores)
- ▶ 24GB RAM
- ▶ ~1TB local disk space
- ▶ HS06 rating ~97
- ▶ SLC5 **XEN** (1 rack with **KVM** for testing)
- ▶ Up to **44 VMs** per hypervisor (on private IPs)

By making use of our internal Cloud infrastructure !



Opportunity to test and probe in addition:

- ▶ The general infrastructure
- ▶ The dynamic extending and shrinking of the batch farm
- ▶ The worker node behavior at large scale (software setup)
- ▶ The image distribution system
- ▶ The image provisioning systems

Initial limitations to protect infrastructure

- ▶ No lemon monitoring of the virtual machines
- ▶ No AFS support in the virtual machines
 - ▶ Rapid coming and going of clients, issue with call backs
 - ▶ Use of ssh keys to access VMs instead of krb5
- ▶ No attempt to quattor-manage individual VMs

General infrastructure:

- ▶ DNS update rate exhausted when registering VMs
 - ▶ Throttle the creation/destruction rate
- ▶ High LDAP query rate
 - ▶ Fixed batch worker node tools (pool cleaner)
 - ▶ Switched off quota on the VMs
 - ▶ May need more ldap servers in the future

LSF itself ...

- ▶ Main issue: dynamic resizing of the farm
- ▶ Close collaboration with Platform Computing
- ▶ Tests beyond of what is officially supported
- ▶ Several iterations with bug fixes on
 - ▶ Mbatchd
 - ▶ Sbatchd
 - ▶ Lim

LSF software: main issues found/fixed

- ▶ Mbatchd performance issue with dynamic hosts
 - ▶ Internal reconfigurations slow down the system
 - ▶ Some mbatchd crashes, specifically while removing hosts
 - ▶ Internal 9999 limit on number of hosts in use
- ▶ Large memory footprint of mbatchd
 - ▶ Will hopefully be addressed in future LSF versions

LSF master hardware upgrade

- ▶ Started with 24GB RAM dual Nehalem CPU server
- ▶ Replaced by 48GB RAM dual Nehalem CPU server
 - ▶ Mbatchd forking problems with >12,000 VMs
 - ▶ Enough to reach ~16,900 VMs and ~400,000 jobs

LSF scalability test layout: the measurement grid

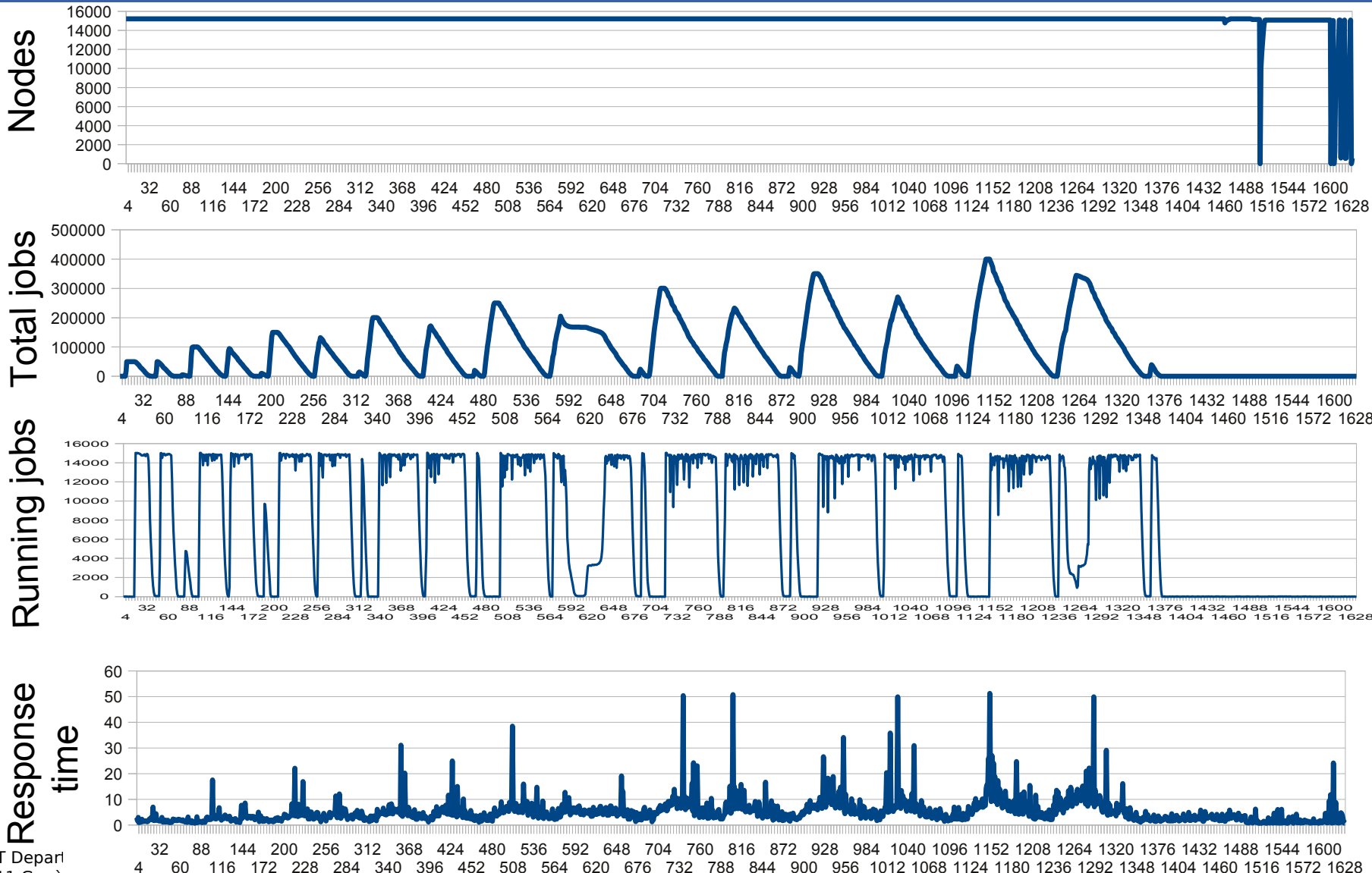
- ▶ 0 – 400,000 jobs
 - ▶ 50,000 jobs steps
 - ▶ Use of **job arrays** à 1,000 jobs each
 - ▶ Short micro benchmark jobs
 - ▶ Most tests done with dispatching enabled
- ▶ 0 – 15,000 worker nodes
 - ▶ Bin size ~2,500 worker nodes
 - ▶ Initially XEN, later XEN + KVM
 - ▶ ISF and ONE used as provisioning systems

LSF scalability test layout: probing the system

Data requisition from a monitoring script which

- ▶ Ran on the master and some clients in a loop
- ▶ Recorded the
 - ▶ Time stamp
 - ▶ Number of nodes and number of jobs
 - ▶ Average LSF communication response time
 - ▶ Average LSF batch command response time

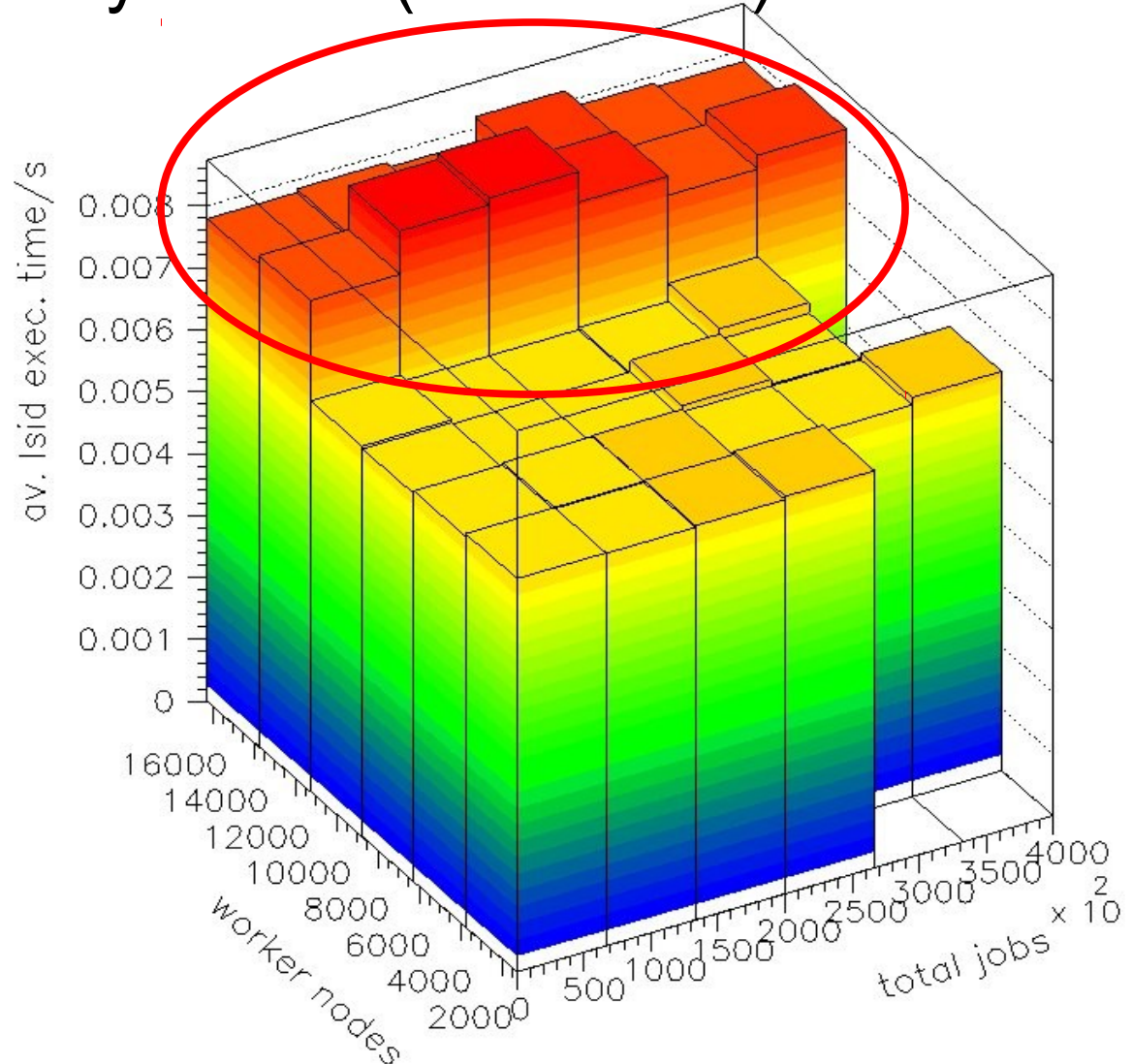
Lessons learned



Lessons learned

LSF communication layer: **Isid** (client view)

- ▶ Behaves as expected
- ▶ Fast response time
- ▶ Some increase > 12,500VM

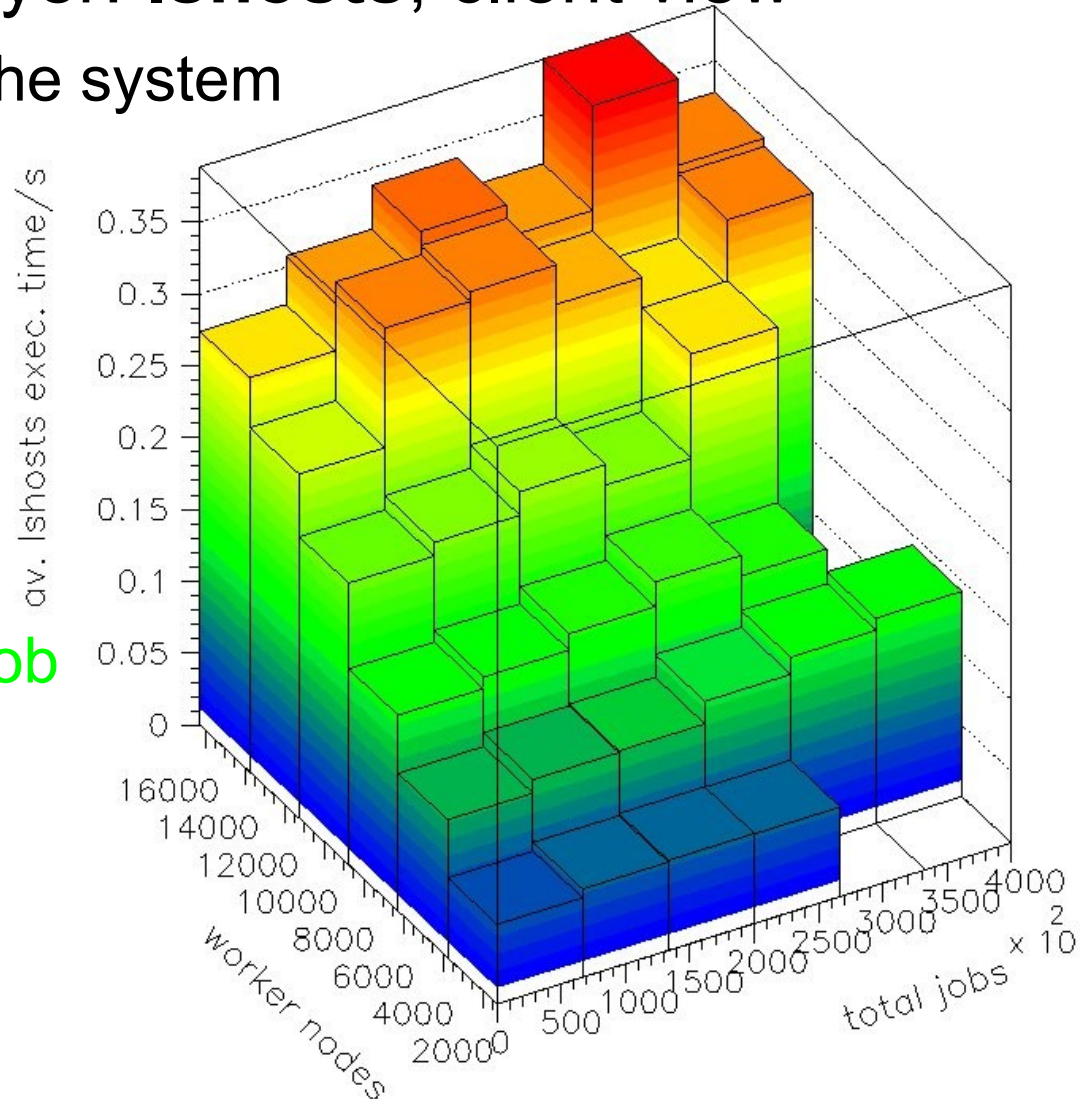


Lessons learned

LSF communication layer: **lshosts**, client view

Returns a list of hosts in the system

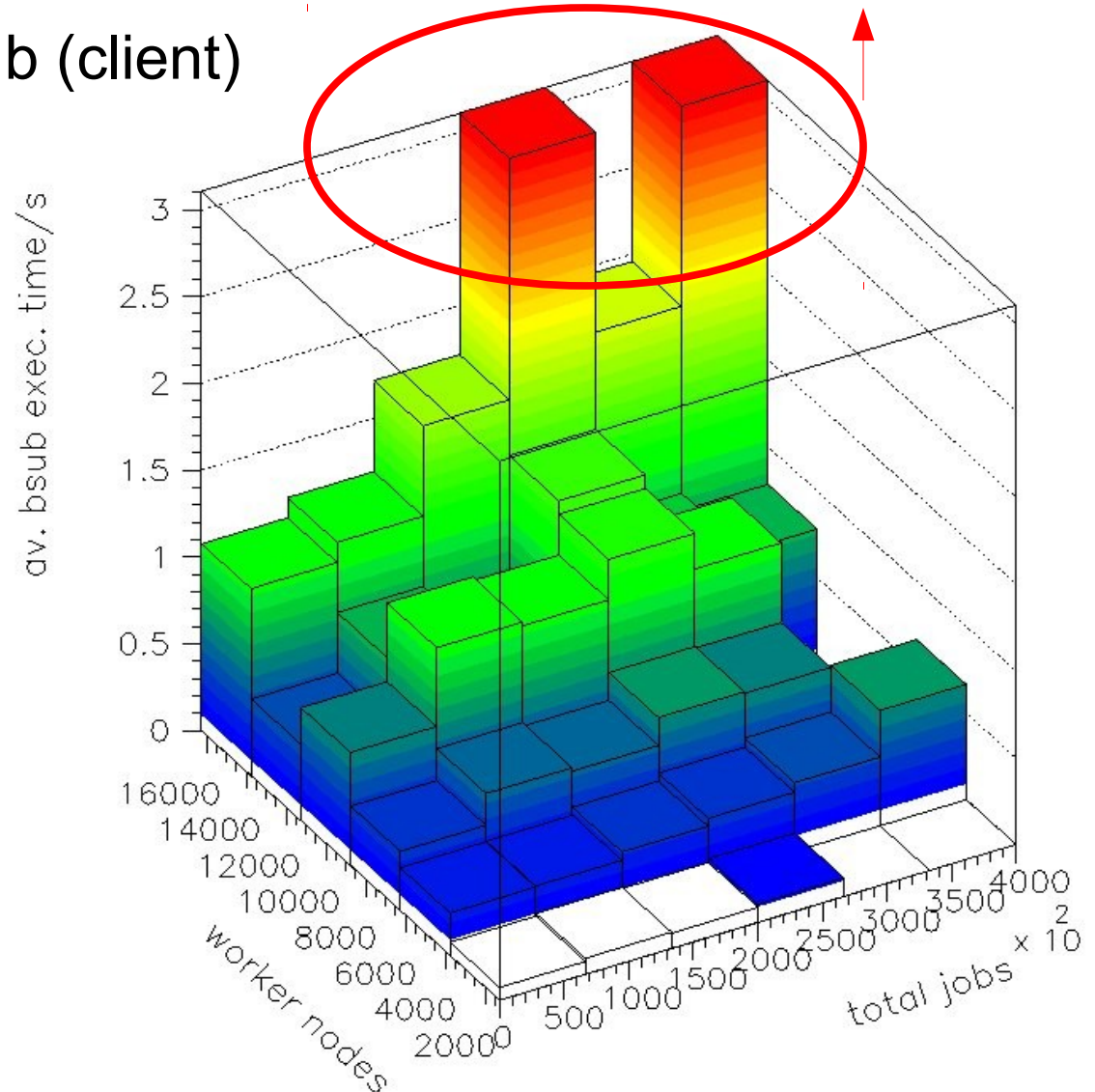
- ▶ Behaves as expected
- ▶ Small dependency on job number



Lessons learned

Batch system layer:
job submission with bsub (client)

Statistics + binning ?

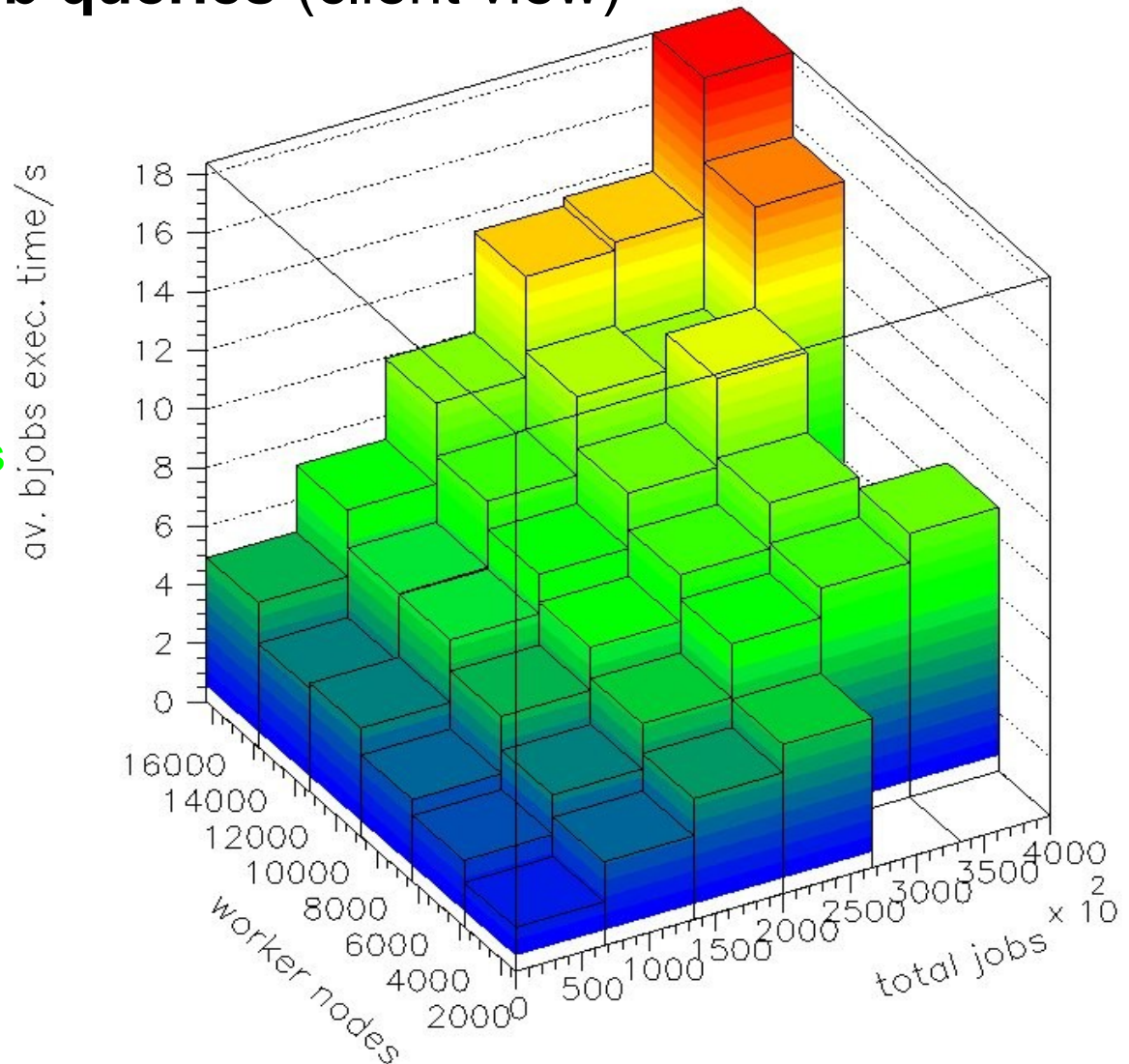


- ▶ T0 requires $t < 1s$!!!
- ▶ Issues > 10,000 nodes

Lessons learned

Batch system layer: **job queries** (client view)

- ▶ Increase with jobs
- ▶ Increase with worker nodes

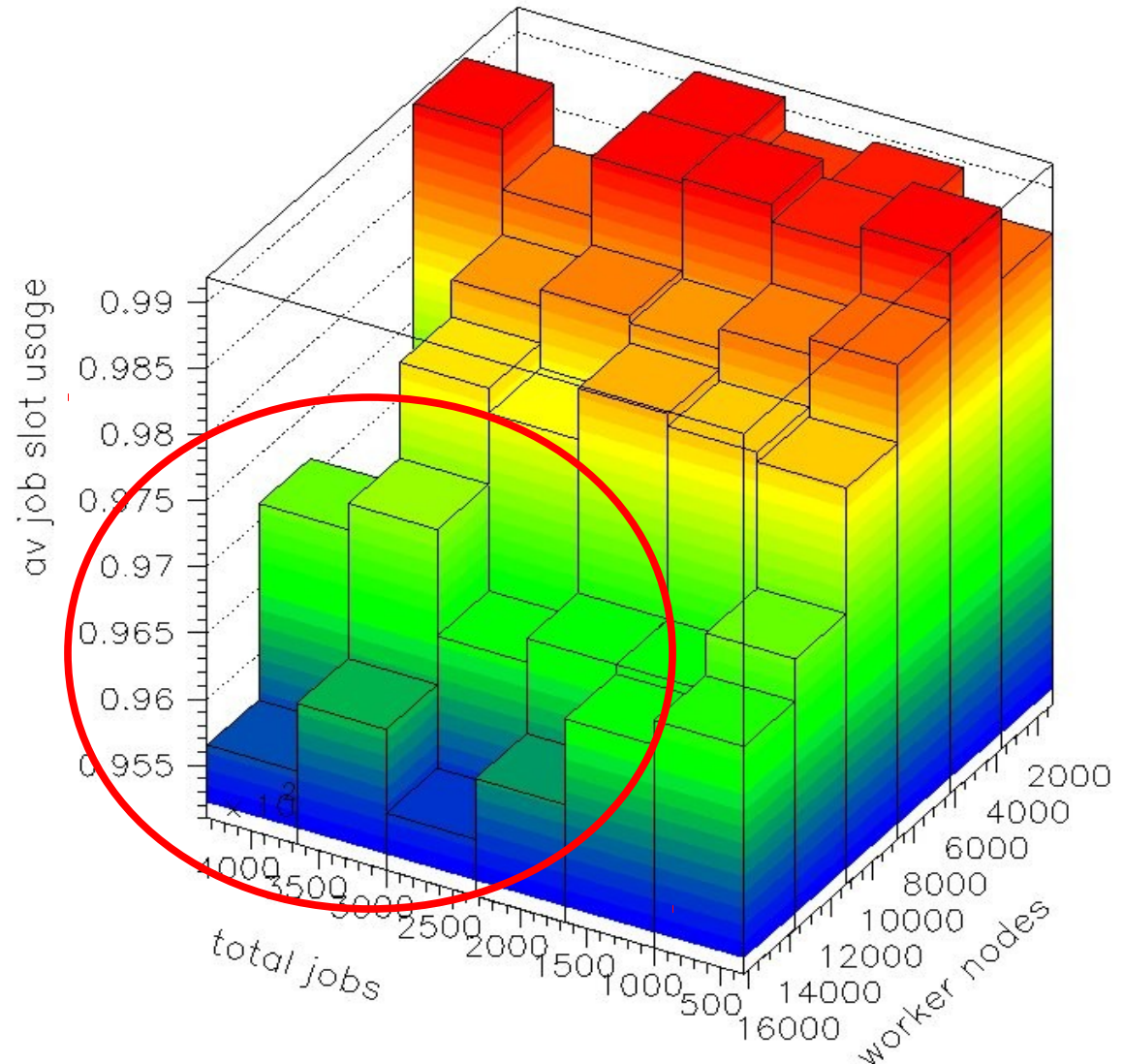


Lessons learned

Scheduling performance: average job slot usage

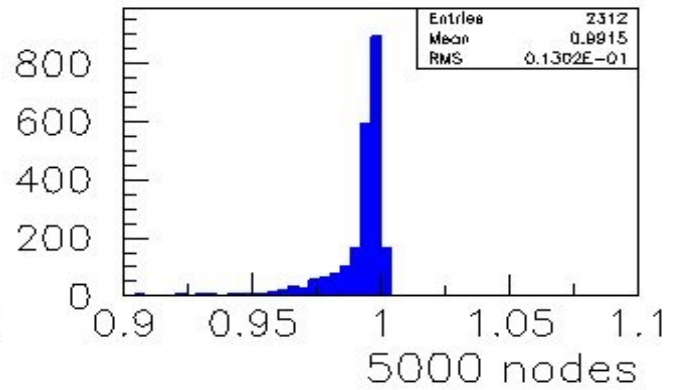
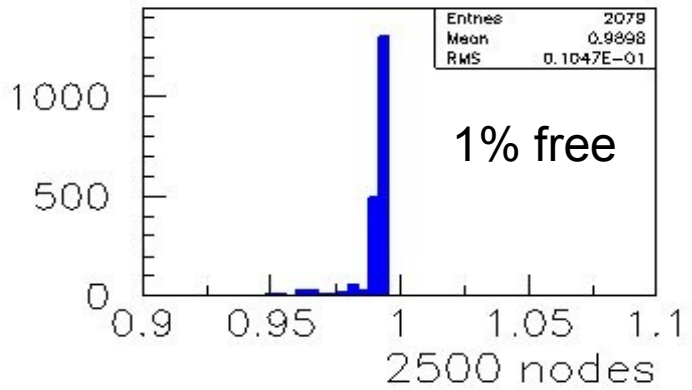
▶ Above 10,000 nodes up to 4%-5% empty job slots

Ratio of Running jobs and Job slots

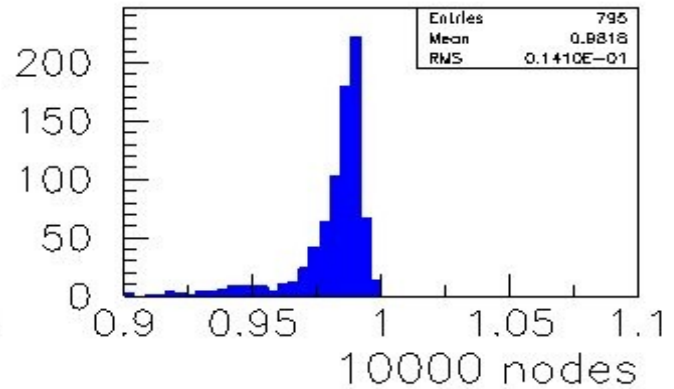
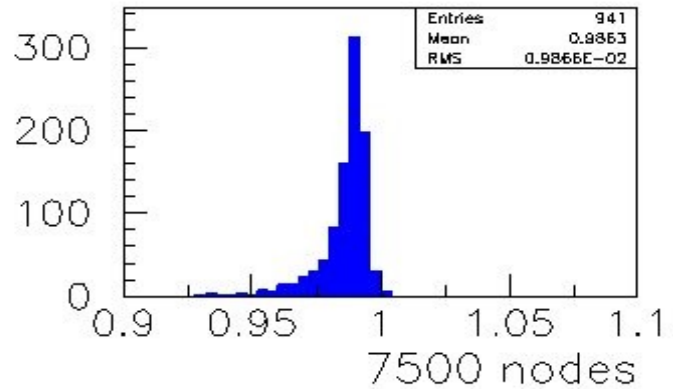


Scheduling performance

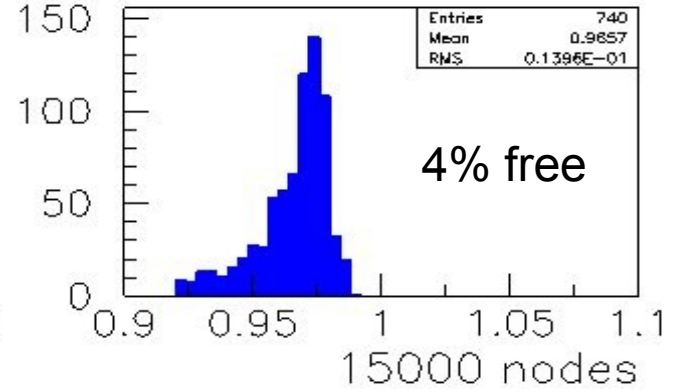
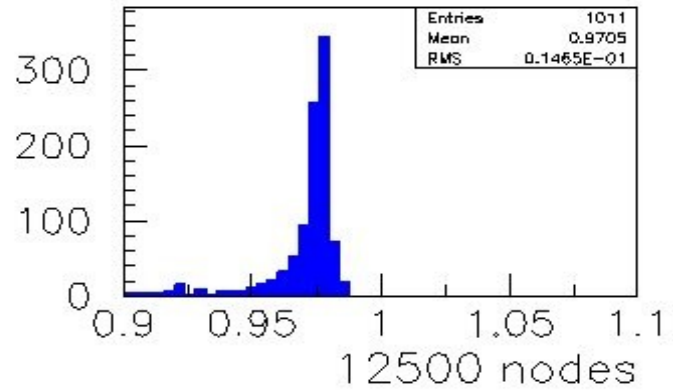
entries



entries



entries



**Ratio of
Running jobs and
Job slots**

Broadening
distributions



Thanks to our IaaS infrastructure we have brought LSF to its limits

- ▶ Up to 20x the number of jobs
- ▶ Up to 4-5x the number of worker nodes

The results are consistent with the expectations from the vendors' statements

The results have been made available to Platform and discussed with them in detail