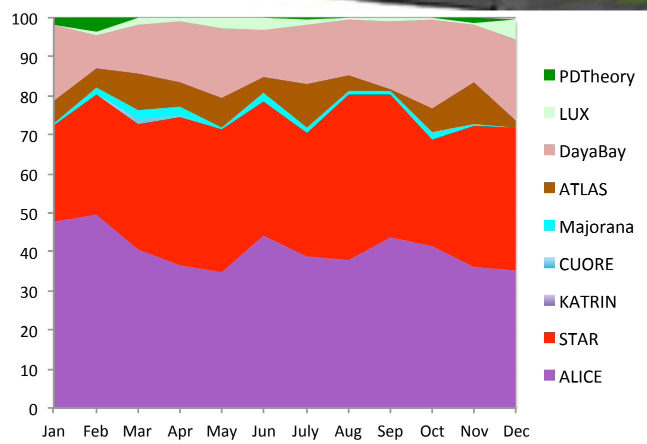
Software distribution via cvmfs @ NERSC



Markus Fasel[1]
Jefferson Porter[1,2]

[1]Lawrence Berkeley National Laboratory

[2]National Energy Research Scientific Computing Center NERSC

CernVM Users Workshop 2016, Rutherford Appleton Laboratory

Goal

Nuclear / High Energy Physics

- Centralized Software management
- Controlled environment

High Performance Computing

- Optimized systems
- Restrictions

Combine the two worlds

Outline

- Introduction of NERSC HPC systems
- Ways of implementing cvmfs on NERSC HPC Systems
- Shifter
- Our experience with ways of mimicking cvmfs on the Cori supercomputer

NERSC systems

Computing systems

Cori Phase1

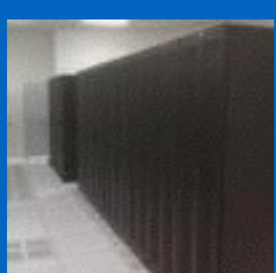
1630 Nodes
52k CPUs
Intel Xeon Haswell
32 cores / node
128 GB RAM / node
28 PB SCRATCH
750 TB Burst Buffer

Edison

2.58 PF

5576 Nodes
134k CPUs
Intel Xeon Ivy Bridge
24 cores / node
64 GB RAM / node
7.6 PB SCRATCH

PDSF



Batch farm
Mix of AMD and Intel CPUs
120 Nodes
~3k Cores
2-4 GB RAM / core

Common file systems

/project: 9.1 PB quota-based GPFS, for long term storage (not optimized for I/O)

/home: 275 TB, user home dirs

HPSS: 240 PB (max) tape file system for data archiving

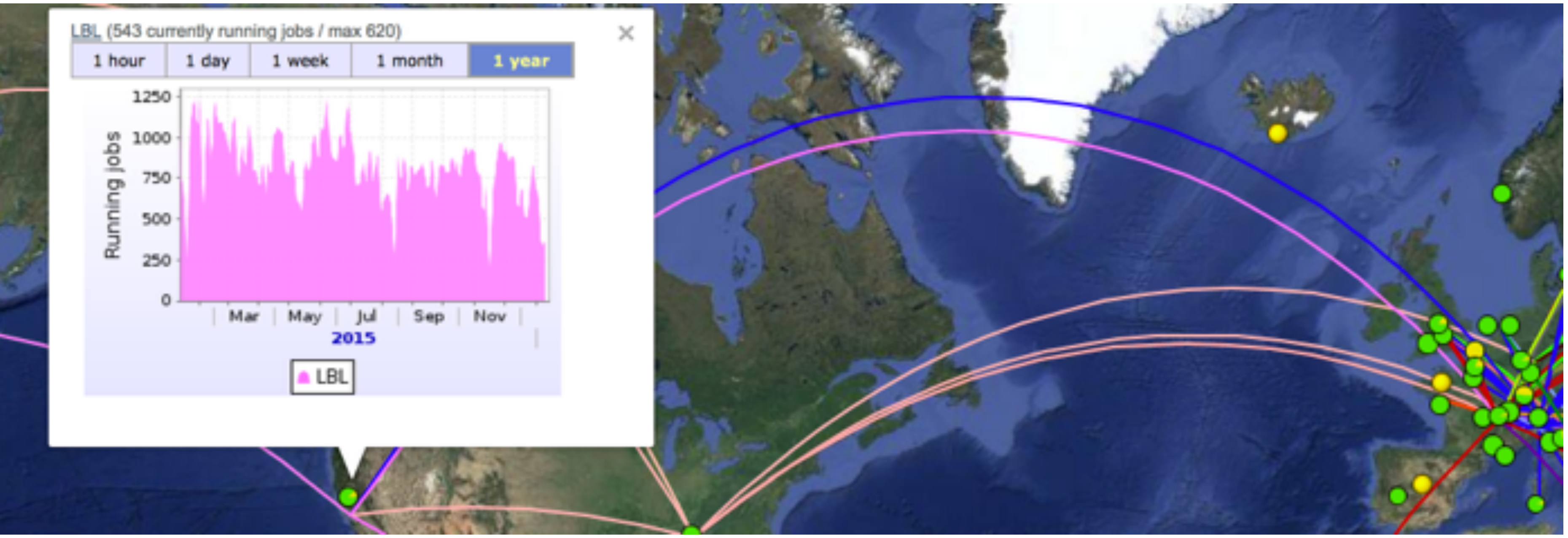
Central services

Data transfer nodes

- 4 nodes, 10 Gigabit wan connection per node

Science gateway for web apps

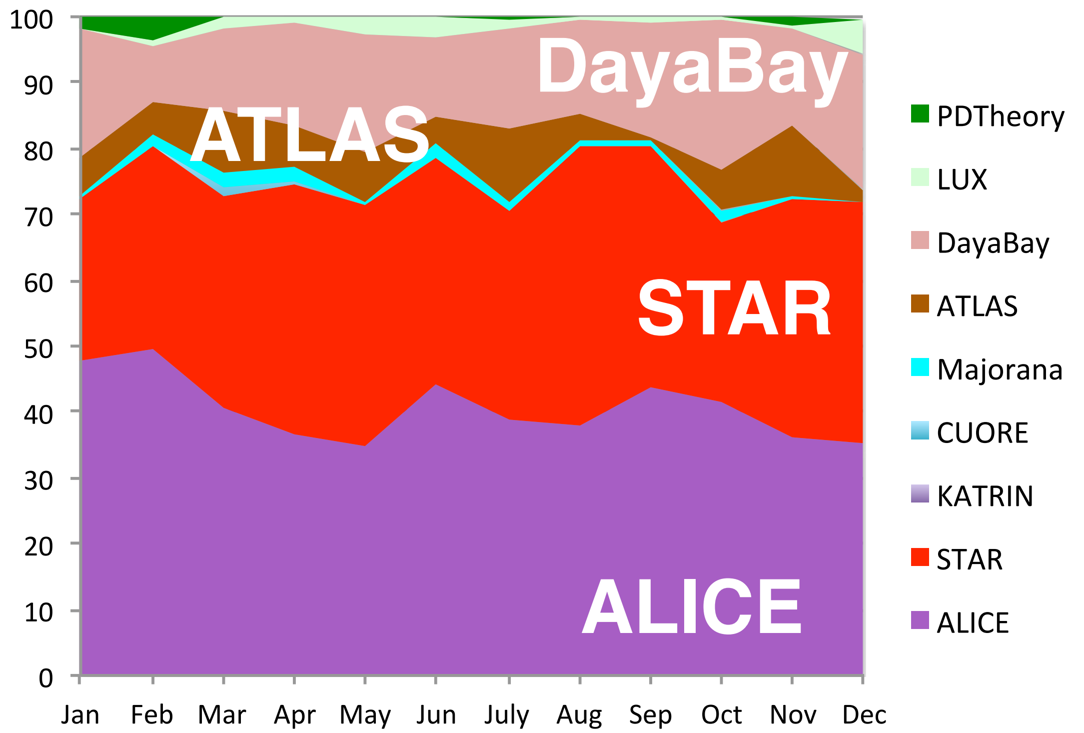
PDF: HEP/NP Cluster at NERSC



Conventional HEP/NP Cluster

- ~3200 cores
- Univa Grid Engine
- OSG Compute element
- Serves as
  - ALICE Tier2 Grid site
  - ATLAS Tier3 Grid site

Cluster usage 2015



Cori, now and future

Most modern supercomputer at NERSC

- Named after the bio-chemicist Getry Cori
- Connected to
  - 28 PB Lustre scratch file system
  - Burst Buffer

Burst Buffer

File system for I/O intensive jobs

- Cray Data Warp technology
- SSD based
- Size
  - At Phase 1: 750 TB
  - At Phase 2: ~1.5 PB

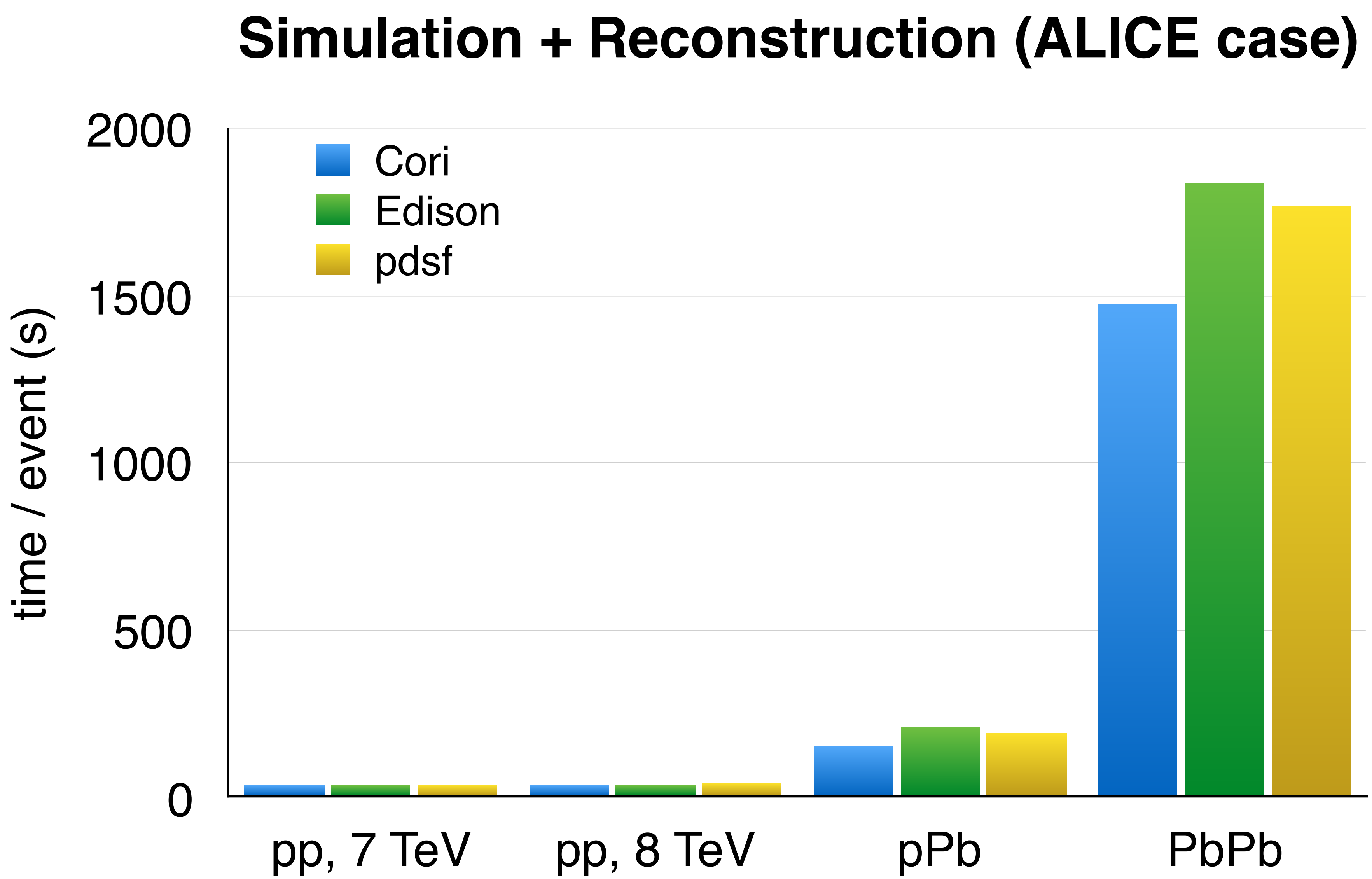
Now: Phase 1

- Started December 2015
- 2K Haswell nodes
- 32 cores / node
- 128 GB RAM / node

This year: Phase 2

- Planned for late 2016
- ~9K Knight Landing nodes
- 60+ cores / node
- 96 GB / node

Performance on HPC systems competitive



High performance cluster are competitive compared to standard batch farms

PDSF has a mixture of different CPU types

- Same performance to Cori for jobs on same CPU type

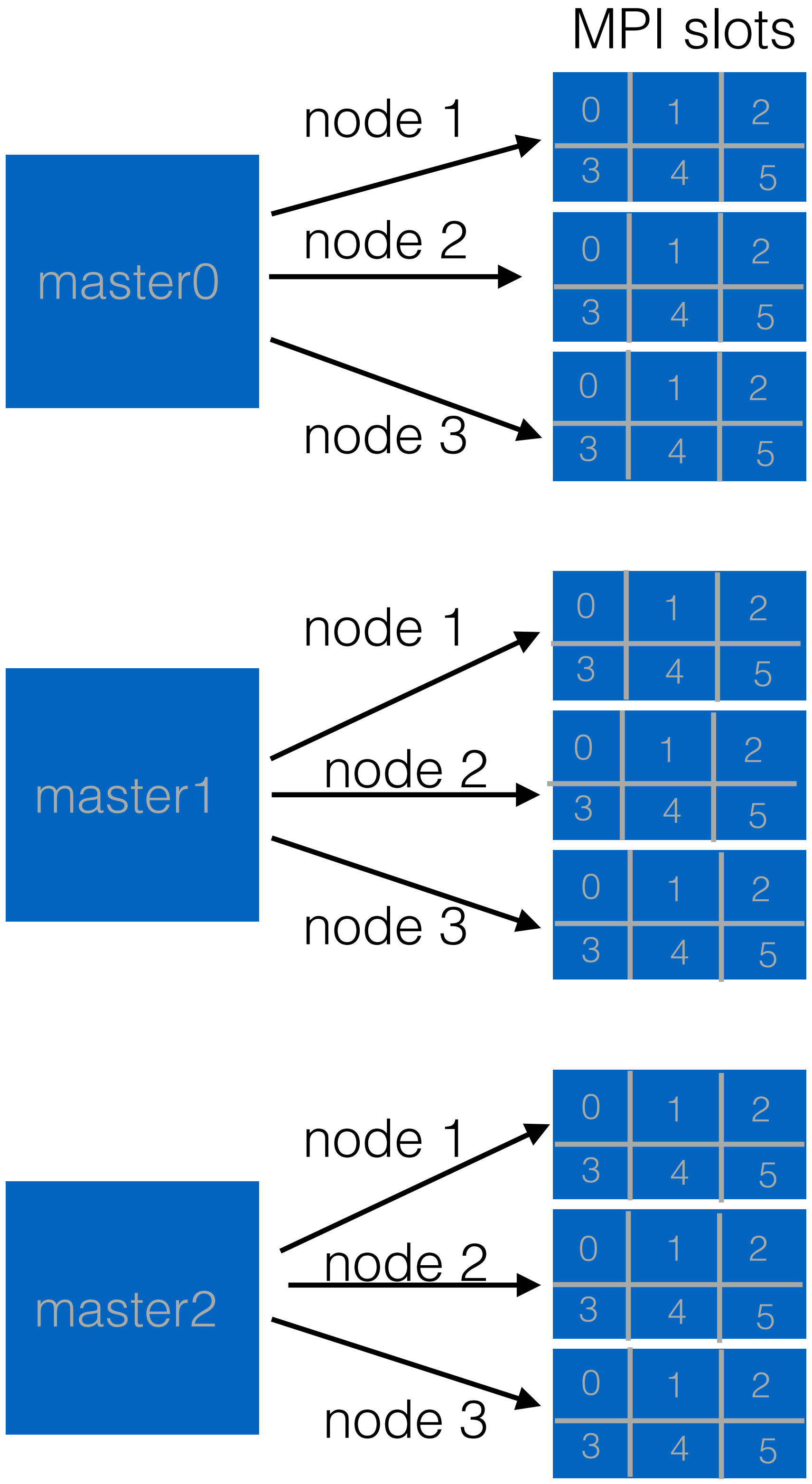
HPC Interface for Nuclear Physics Jobs: ANALISA

Tool which runs multiple serial jobs as a MPI job

- Submitter:
  - Splits a master into n sub jobs
- Worker (MPI):
  - Runs the subjobs (payload)
- Job description: config, json, xml

Flexibility

- Single-node - use backfill capabilities
- Multiple nodes for large productions
  - All jobs start in unison



Hides complexity of resource management for the user

Challenges in deploying software via cvmfs on HPC systems

- Special Linux kernel & OS
- No root access
  - No fuse
- No local disk
- No external network connection
  - Cori does have external network

User defined images with **SHIFTER**

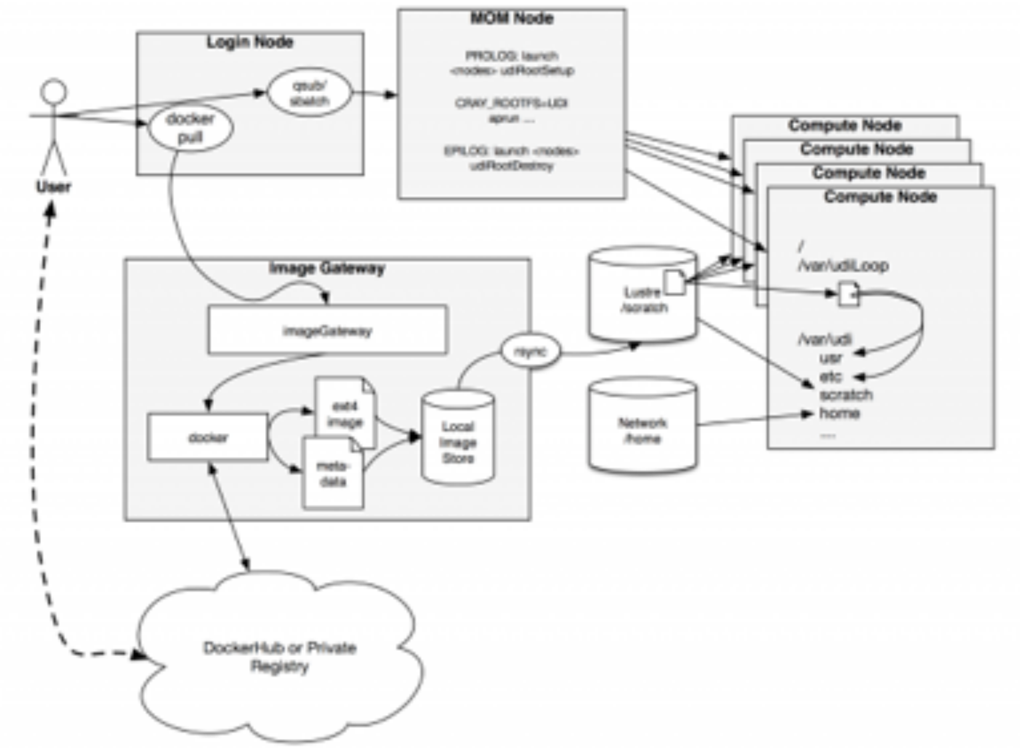
Tool to run linux containers on HPC systems

Advantages:

- Provide native software environment (i. e. Scientific Linux)
- Performance scaling with number of nodes
- Integrated into the batch system

Additional use case:

- Direct mimicking of cvmfs by dumping the file system content into a docker image



- approved for release through a BSD license
- Goal: usage at other centers
    - Strong interest from Cray

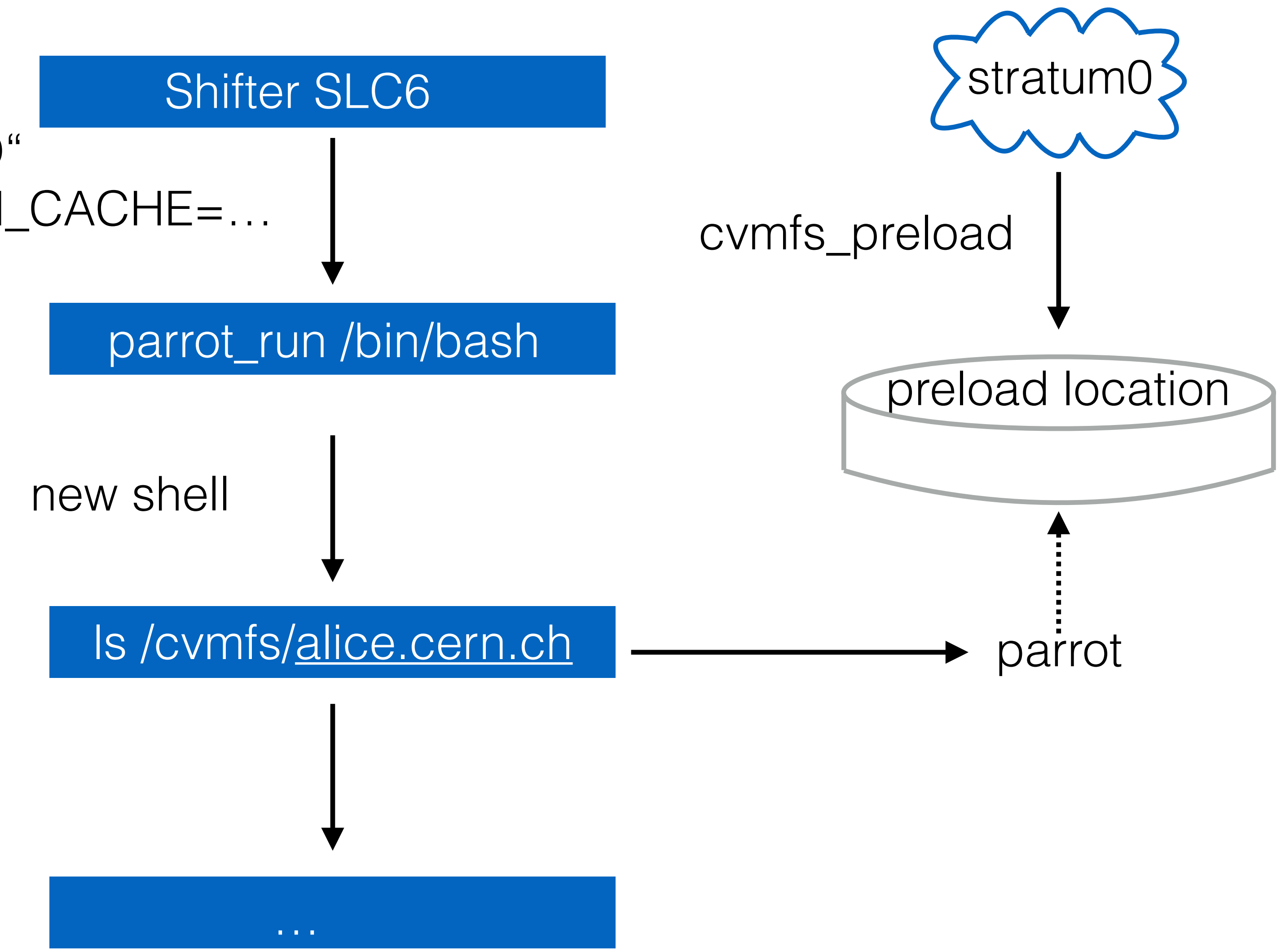
Building an image with a cvmfs repository

- **Access cvmfs:**
  - Use cvmfs_snapshot to pull down full repository
  - Rsync CVMFS onto image
- **Use uncvmfs to dedupe files:**
  - Python routines that crawls repository
  - Finds duplicate files and replaces them with hard links
- **Convert ext4 image with to squashfs image:**
  - Compresses data, inodes, and directories
  - Read only file system
- **Can make a fresh image ~daily:**
  - CVMFS update ~2 hours
  - Squashfs conversion ~8 hours
  - Copy into place ~1 hour

Tested with ATLAS, ALICE, and CMS simulations out to 1000 nodes

cvmfs via parrot: static cvmfs_preload repository

export HTTP_PROXY=„INVALID“
export PARROT_CVMFS_ALIEN_CACHE=…
…



Parrot

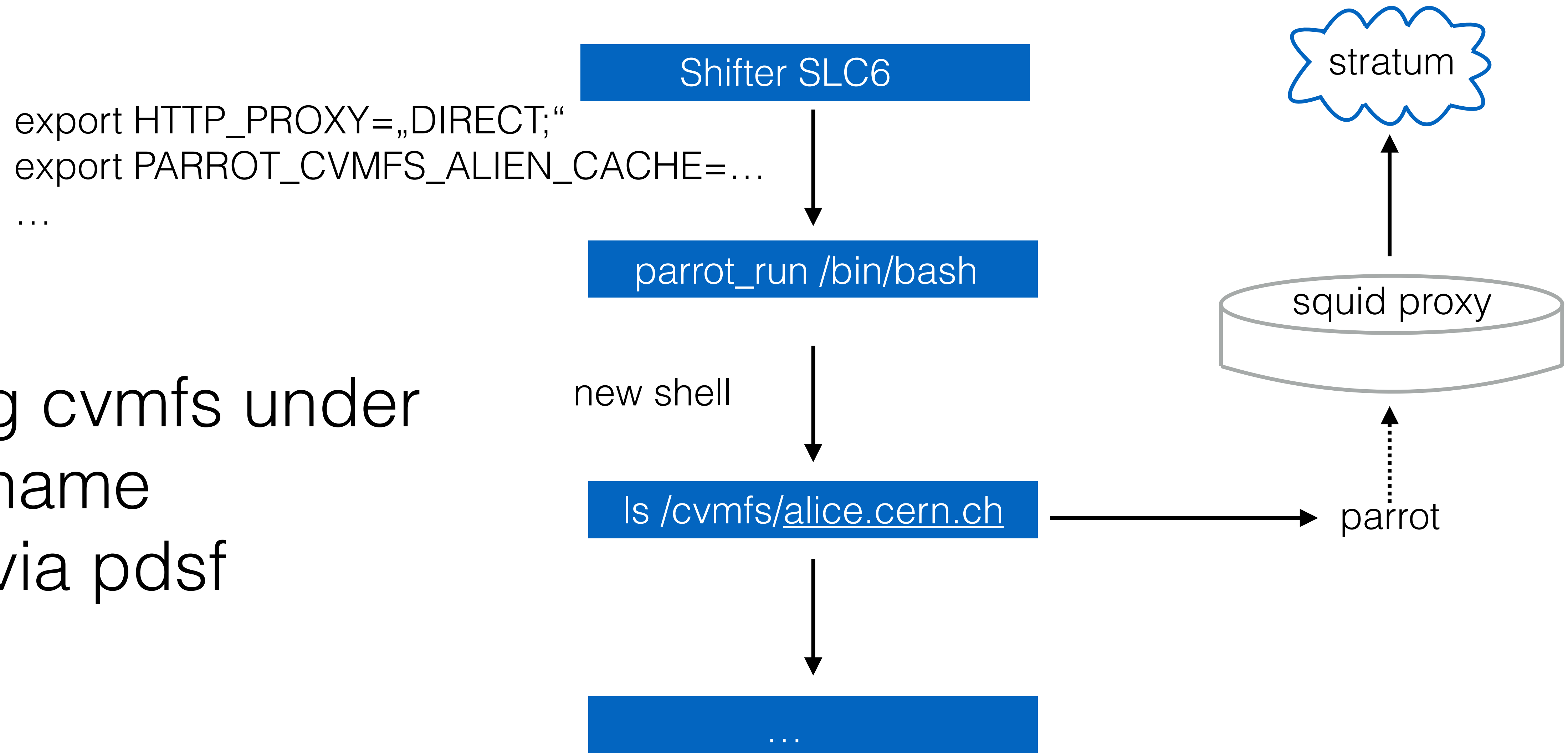
- mounting cvmfs under original name
- using preload from external location

Using shifter only to provide SLC6 environment

Preload options:

- ☑ GPFS file system ⇒ non-purgeable
- ☑ Lustre scratch ⇒ purgeable, needs special allocation
- ☐ Burst Buffer ⇒ purgeable, created per job

cvmfs via parrot: squid servers and dynamic cache



Parrot

- mounting cvmfs under original name
- Access via pdsf squids

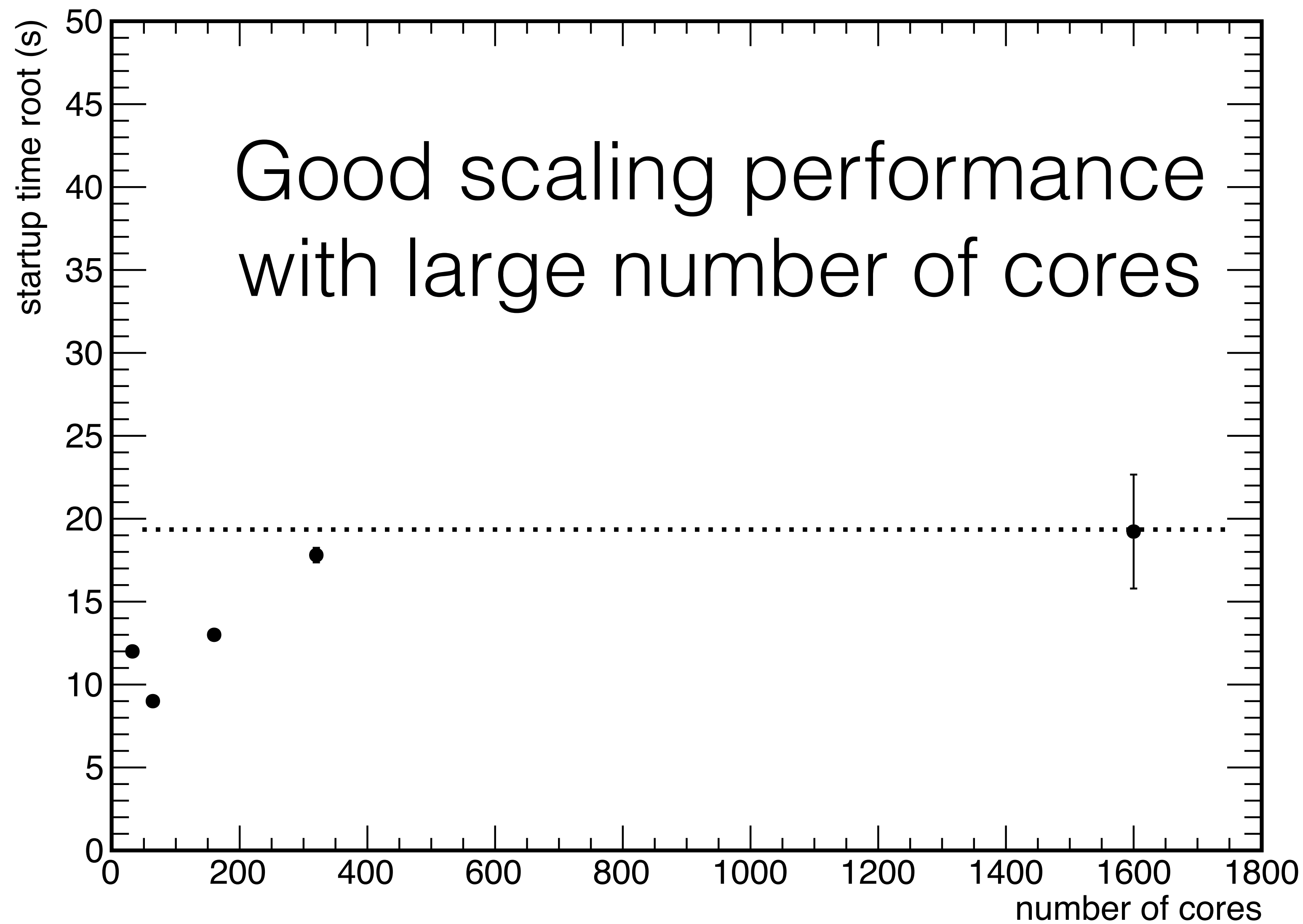
Using shifter only to provide SLC6 environment

Load options:

☑ Large global alien cache on lustre scratch
☑ Per-node dynamic alien cache on lustre
☐ Per-node dynamic alien cache in memory

Tests with cvmfs repo in shifter image

Performance:



Issues:

- Large image (e.g. ALICE: ~600 Gb, 15M inodes)
- Long time to build it
  - Daily
- Needs to be produced by a NERSC staff

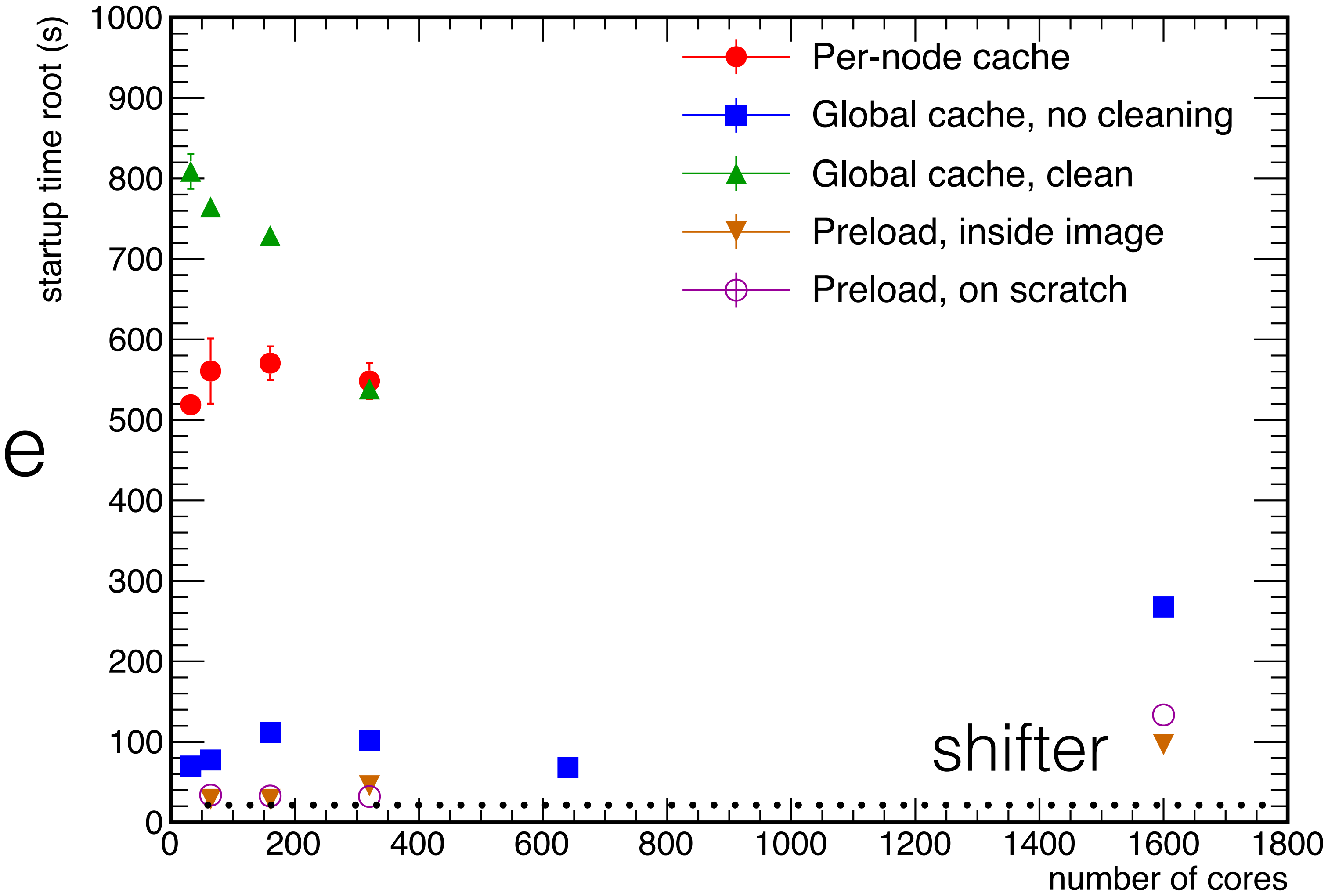
R&D prospect

Tests of the cvmfs access using parrot

Performance:

Content from SQUID

⚡Network has influence



Issues:

- Several R&D issues open
- Performance using squid:
  - Per-node cache / fresh cache poor

Conclusions

- Centralized software distribution via cvmfs crucial for high-energy nuclear physics experiments
- Restrictions in HPC systems require mimicking techniques for cvmfs
- Several techniques available on NERSC systems
    - Parrot + preload
    - Squashfs + shifter
- Thanks to shifter, a native operating system environment is made available to the compute nodes

Work in progress:

- Scale tests of the preload
- Squid server on Cori