LSST activities at CC-IN2P3

status and perspectives

Fabio Hernandez
fabio@in2p3.fr

Visit of NCSA to CC-IN2P3, Lyon, May 27th, 2015

Objectives

- To summarise ongoing LSST-related activities at CC-IN2P3
- To present ideas on possible further contributions
- To identify concerns and commonalities between NCSA and CC-IN2P3 that could lead to joint work

Contents

- Ongoing activities
  - *Qserv integration cluster*
  - *binary distribution of the stack*
  - *analysis of I/O activity induced by the stack*
- Potential further contributions on two themes:
  - *efficient data access*
  - *convenient tools for end-users*

Ongoing activities

Qserv integration cluster

- Goals

  to provide developers a realistic testbed platform for developing, integrating, operating and validating Qserv

  to exercise the system with realistic use cases (e.g. CFHT data processing using LSST software stack) and provide feedback to developers

- Aggregated capacity: 400 CPU cores, 800 GB memory, 500 TB disk

  eventually composed of 50 boxes + virtual machines for building and packaging the software

  25 boxes operational, remaining hardware already delivered and being deployed

- Qserv development team routinely working with it

Binary distribution of LSST stack

- Goal

  to provide a ready-to-use distribution of stable versions of LSST software stack

- Benefits

  no need to build the software from scratch in order to use it

  self-contained, unique binary distribution to be used for personal computers, worker nodes, virtual machines, containers, etc., for consistency

  usable also for developing or extending modules on top of the built-in stack components

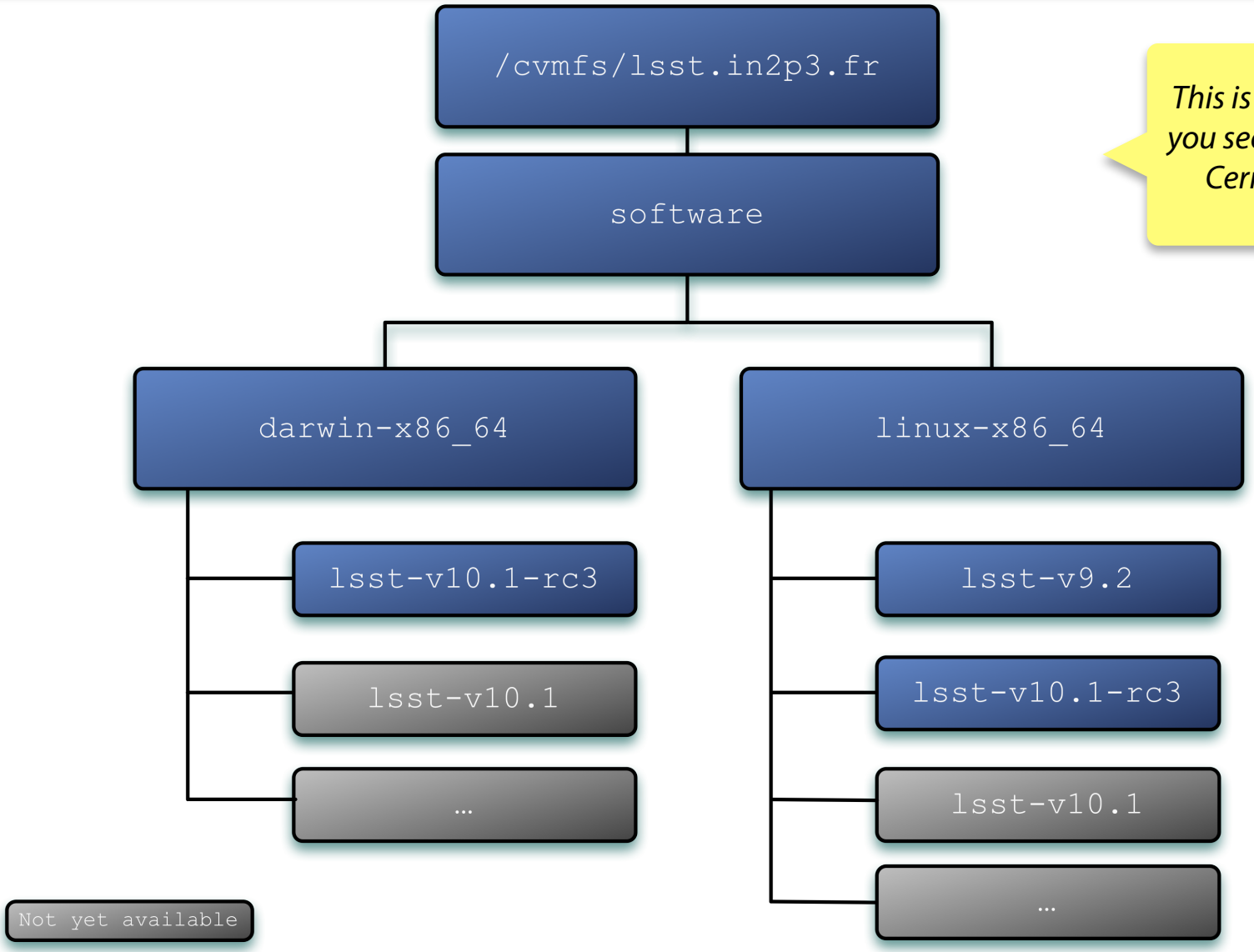
- CernVM FS-based LSST repository hosted and maintained at CC-IN2P3

  LSST v9.2 and 10.1-rc3 available; v10.1 in preparation

  works on Ubuntu 14.04, Scientific Linux 6 & 7, CentOS 7, MacOS X

  requires installation of CernVM FS client software on the execution machine (one time process)

Binary distribution of LSST stack (cont.)



This is the namespace you see after installing CernVM FS client

Binary distribution of LSST stack (cont.)

- How to use

```
$ cd /cvmfs/lsst.in2p3.fr/software/linux-x86_64/lsst-v10.1

$ source loadLSST.sh
```

- This command sets up the LSST environmental variables and gets you ready to use the stack
- Since LSST stack is not relocatable, we build each stable official version from source for distributing via CernVM FS
- More information: https://github.com/airnandez/lsst-cvmfs

Analysis of I/O activity

- Goal: to understand file I/O patterns induced by LSST software stack

  to identify the characteristics of the data storage platform which best suits the needs of LSST

  or alternatively, to provide feedback to LSST software developers to take into account the intrinsic limitations of the available storage platforms

- Status

  developed tool for collecting machine-parseable data on I/O activity at the file system-level

  details: https://github.com/airnandez/cluefs

- Ongoing work

  developing the tools (Python notebooks) for initial exploration and analysis of the data collected when using the LSST stack for processing CFHT data

  example of preliminary findings: a single index file of astrometry.net is open 210 times when executing the stack demo — not necessarily an anomaly, but an interesting fact we could exploit

  exploring how to use the collected data to build a simulation model of the I/O activity induced at the scale required for the DRP

  work just started by a PhD student from IHEP (Beijing, China) staying for 1 year at CC-IN2P3

  simulator to be built on top of SimGrid: http://simgrid.gforge.inria.fr

Foreseen activities

Object storage-based data repository

- Goal: explore object storage as an alternative mechanism to conventional networked file systems, as a repository for LSST data

  what it would take to use object storage (such as OpenStack Swift or Amazon S3) for LSST?

  model seems well suited for serving large quantities of immutable files

- Potential benefits

  *LSST data would be accessible via **standard protocols**, both from co-located worker nodes or from remote locations (e.g. your laptop, wherever you are): **no confinement of the data to a single site***

  *make the stack cloud-ready, which allows for **separation of processing clusters from storage clusters***

  elasticity of cloud-based object storage model, interesting for computing center operators

- Status

  previous encouraging results evaluating object storage for storing LHC data: https://speakerdeck.com/airnandez/files-without-borders

  OpenStack Swift evaluation cluster available at CC-IN2P3

- Ways to go forward

  1) modify the Butler to read and write files using Swift/S3 protocols, or

  2) implement a synthesised file system to expose Swift/S3 backends using the file system API — transparent to the Butler, no modifications needed

- Questions

  this is considered a topic of interest by the LSST DM leaders

Object storage for data exchange

- Goal

  explore object storage backends as a mechanism for inter-site data exchange

  decouple the storage system used for inter-site data exchange from the system used for on-site data processing

- Idea

  use OpenStack Swift as a reception/emission buffer for inter-site data exchange

  transfer data using standard protocols, in particular HTTP-based, leveraging the work of a huge community

- Comments

  this may not be an immediate concern for LSST-DM, but experience with LHC data processing shows that it is a topic that must be addressed early on and requires preparation and several iterations

Cloud-based Python notebook execution engine

- Goal

  provide a mechanism for end-users to use the stack for exploring LSST data via Python notebooks

  ***no software installation**: use the cloud-based software distribution*

  ***no need for the user to be near where the data are**: use cloud-based data access mechanisms*

- Principle of operation

  end-user opens a Python notebook and directly uses the stack and access the data, from his laptop, independent of his location

  execution engine runs near the location of the data, i.e. on a VM at CC-IN2P3, and access data located on site (either on a networked file system or in Swift)

  private VM instantiated on demand for the end-user, mounts the stack via the binary distribution and access data over the local network

  we could provide every end-user a cloud-based, Internet-accessible personal storage space for his own storage needs: here convenience is more important than performance

- Comments

  this is a topic to discuss with the IPAC team, in charge of end-user interface

Unified cluster management for LSST DRP

- Goals

  explore suitability of new tools for managing clusters, for bulk data processing

- Idea

  *would it be interesting to build a **single cluster**, composed of compute nodes physically located at NCSA and CC-IN2P3 for DRP?*

  tools such as Apache Mesos look promising for this

- Benefits

  be prepared for alternative modes of cluster-based data processing for the era 2020-2030, beyond the conventional single-site cluster batch system

  we could exploit time shifts between Urbana-Champaign and Lyon for round-the-clock operations

- Constraints

  this would need the processing workflows to be aware of the physical location of the data (at the site level), to avoid unnecessary access to the data over the network across the Atlantic

Memory-based file cache at cluster level

- Goal

  explore a system for caching data files (e.g. immutable images) in memory, for bulk processing

  the system aggregates the memory of a cluster of machines and exposes it as a synthesised file system with single namespace

- Principle of operation

  applications open files as usual, file chunks enter the cache on demand (i.e. at open) from the disk-based repository

  subsequent operations on the same file will exploit the cluster memory cache: read operations will retrieve the data from memory in another machine in the same cluster

  inspired from Spark's TachyonFS

  furthermore, system could exploit known file formats: for instance, opening a FITS file would automatically load all the header records into cache

- Use case

  bulk processing of set of images, requiring repeated access to the same files

- Comment

  not clear if a cache system like this would benefit the LSST workflows as currently designed

comments & questions