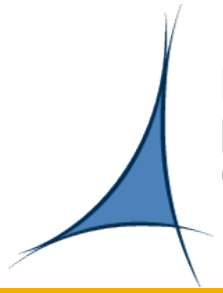
Alternative models to distribute VO specific software to WLCG sites: a prototype set up at PIC

Elisa Lanciotti, Arnau Bria, Gonzalo Merino
PIC Tier1, Barcelona

5th EGEE User Forum, Uppsala, April 12th 2010



PIC
port d'informació
científica

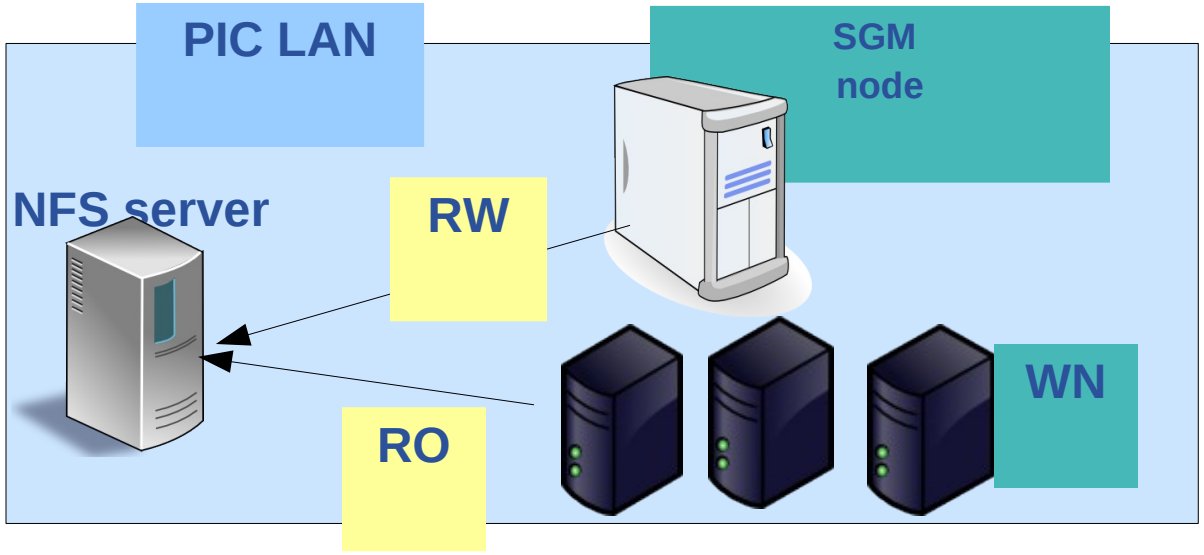
e-infrastructure

Contents

- **Current model to distribute VO specific software to WLCG sites**
- **Some issues with the current model**
- **An alternative model to distribute software packages to the WNs using web caching, focusing on LHCb use case**
  - Description of the setup at PIC
  - Functionality tests and preliminary results
- **A second alternative model using CVMFS**
  - Prototype at PIC and first tests
- **Summary and outlook**

Current model

- In a distributed computing model as WLCG the software of VO specific applications has to be efficiently distributed to any site of the Grid
- Applications software currently installed in a shared area of the site visible for all worker nodes (WN) of the site (NFS, AFS or other)
- The software is installed by jobs which run on the SGM node (a privileged node of the computing farm where the shared area is mounted in write mode)

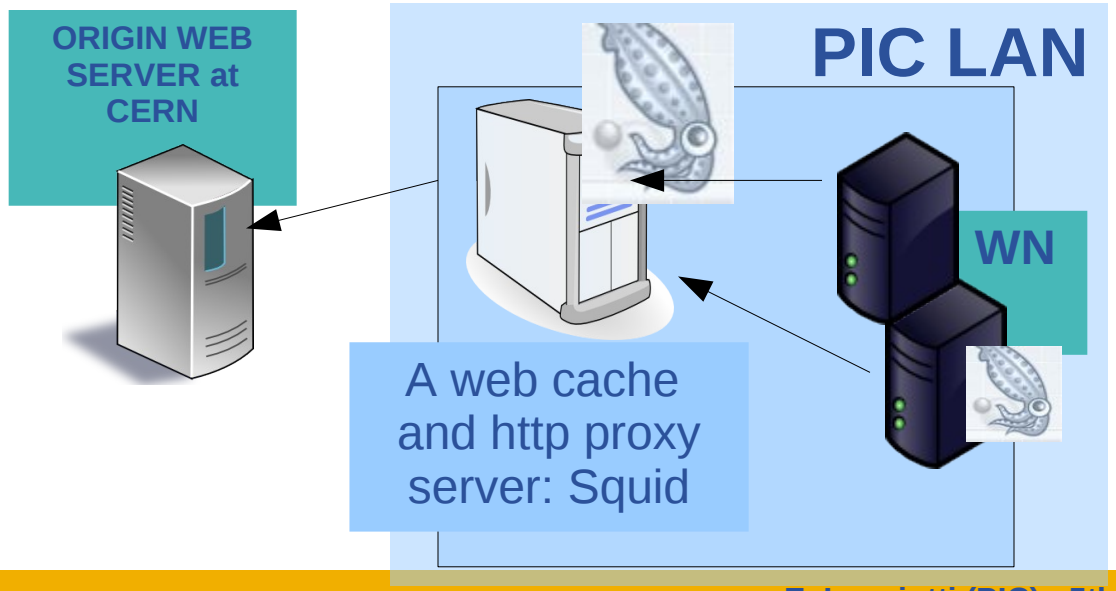


Some issues with the current model

- **Some issues observed with this model:**
  - NFS scalability issues
  - Shared area sometimes not reachable (not properly mounted on the WN, or too loaded NFS server..)
  - NFS locked by SQLite (known bug if NFS is mounted in r-w mode)
  - Software installation in many Grid sites is a tough task (job failures, resubmission, tags publications...)
  - Limited quota per VO in the shared area: if VOs want to install new releases and keep the old ones they have to ask for an increase of quota
- **For LHCb: In case the software area is not reachable jobs as fall back case can download the software from a web server and install the application locally**
  - Drawback: network latency to download the application tarballs through the WAN

An alternative model using http caching

- The software is accessed through http from a central web server at CERN
- A web cache at the site stores all the packages currently in use: Squid is a very good solution for this: optimizes network traffic and provides very efficient caching
- WNs access the software through the LAN
- Potentially more scalable: using local caches at the WN and using p2p transfer among WNs



Estimation of the disk cache on the WN about 3GB. Computed on the basis of LHCb use case: last 3-4 DaVinci (data analysis) versions in use + official production software

Test setup at PIC Tier1

- **Two level hierarchy:**
  - One main Squid server acting both as http proxy and as main cache at the site
  - One Squid at each WN with a small cache (2-3 GB) and limited memory requirement (recommended 32 MB memory for each GB of disk cache).
- **Configured a test queue with only these WNs visible through ce-test.pic.es**
- **Test job: a bash script which**
  - export VO_LHCB_SW_DIR=”.”
  - export http_proxy=localhost:3128
  - executes install_project DaVinci(v25r1,v25r2,v24r7)
  - runs DaVinci (analysis software of LHCb)
- **No development required: LHCb standard installation tools already foresee local installation**

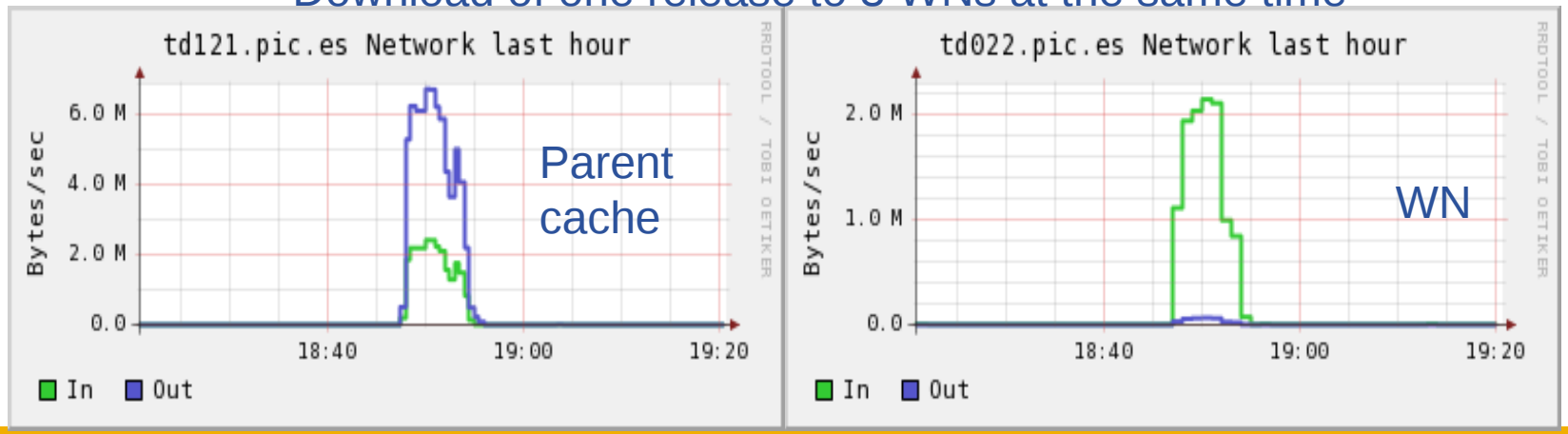
Functionality tests

First run: the job triggers the software download from the software repository at CERN

- software packages not cached on the WN
- Squid on the WN forwards the request to the parent cache.
- The parent cache doesn't have the objects cached either, so it sends the http request to the origin server
- About 5 minutes to download one release (600 MB). See picture.

- **Following runs: no network traffic. Installation time dominated by the software tarballs decompression: 1 min to uncompress one release for a 8 x Xeon X5355 @ 2.66 GHz core box, but increases dramatically with the number of concurrent jobs (disk contention)**

Download of one release to 3 WNs at the same time

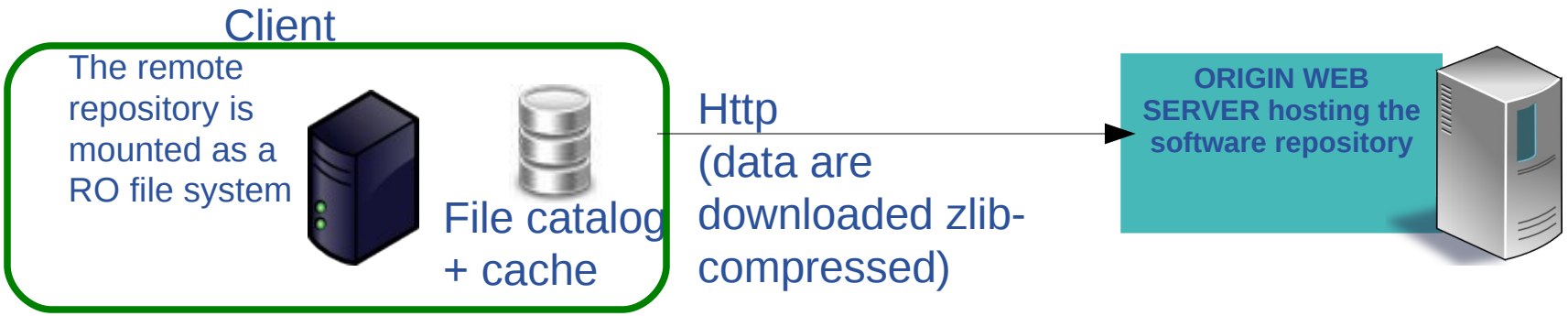


Advantages and drawbacks

- **Advantages wrt the current model:**
  - No need of jobs for software installation site by site
  - Ease of maintenance at sites: standard web caches, no need to worry about software area size or quota per VO.
  - No need to mount the shared area on all WNs (source of quite a lot GGUS tickets at many sites..)
- **Drawbacks:**
  - Unnecessary network traffic due to the fact that the software packages are delivered in tar archives of what the jobs need only 10% (in average)
  - Even in the best case (the package is already cached in the WN and no network traffic is generated) there is an overhead due to uncompress tar archives every time.
- **Possible solution: CVMFS, a network file system being developed at CERN in the framework of the CERNVM project**

Another alternative model yet: CVMFS to access software

- CVMFS is a client-server file system developed to deliver software distributions to virtual machines
- The software reside in form of binaries on a repository server
- The software repository is seen as a directory tree read-only file system by the remote system
- CVMFS is implemented as FUSE module
- Files are accessed through http(s) protocol, avoiding firewall issues
- Web caching reduces to the minimum network latency

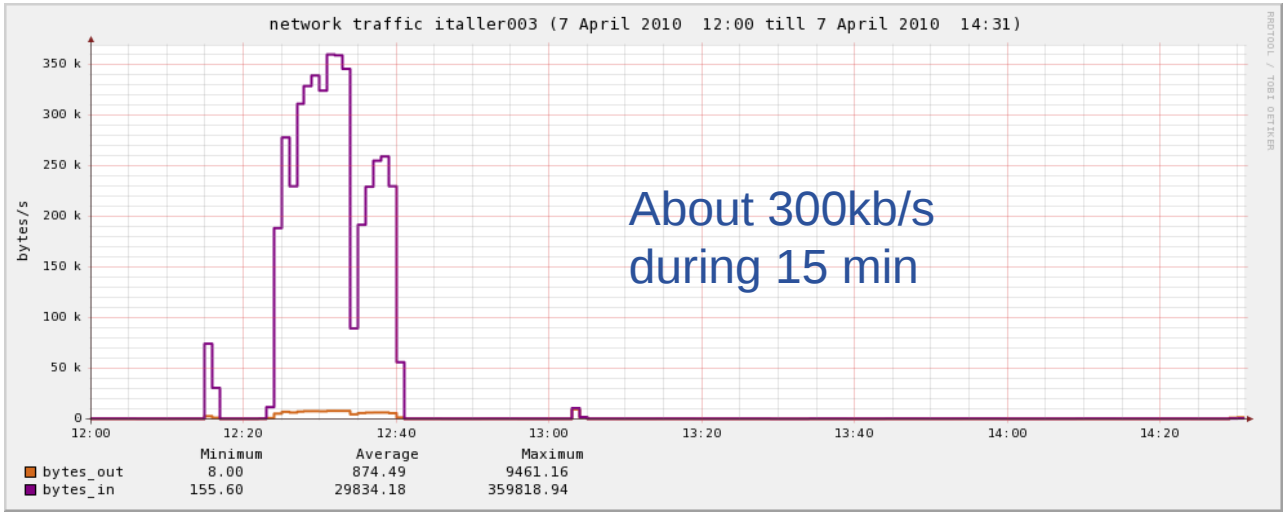


Using CVMFS to distribute software to 'real' worker nodes at PIC

- **Test setup at PIC: CVMFS installed in a WN at PIC. The LHCb software repository hosted by a CERN web server is mounted on the WN as a RO file system**
- **Preliminary results:**
  - First run of a test job which runs DaVinci-v25r1: it takes 15 minutes. About 300MB transferred through the WAN and cached locally
  - The actual content of the local cache on the WN is around 700MB (it also contains catalogue files needed by CVMFS client)
  - Running two more DaVinci versions (v24r7, v25r2) increase the local cache only by 100MB (many files are common)

    Second run of the same version: 35 s. No network traffic.

Incoming network traffic to the WN while running DaVinci-v25r1



Next to do

More tests foreseen, also in coordination with CVMFS developers. Even if CVMFS was designed to be used in virtual machines in the framework of CVM project, they consider interesting this application on real WNs

- **Compare the network traffic to the total amount of data transferred (this gives an idea of the quality of the compression)**
- **Run N concurrent jobs which download data from the same repository to see how the load increases on the origin web server hosting the repository**
- **Setup a Squid proxy local at the site that mirrors the main web server which hosts the software repository (load balance of the traffic). Compare the network latency in two cases**

Summary and outlook

- **Setup based on Squid has some advantages wrt the current model of software distribution, but also presents some drawbacks:**
  - Need to install Squid on every WN
  - Unnecessary network traffic to transfer big tar archives
  - Serious overhead at the beginning of the job to uncompress tar archives

    Not sure whether going on with testing activity
- **Setup based on CVMFS has just been started and only some preliminary results are available. It looks promising and more tests are foreseen, also in collaboration with CVMFS developers.**
  - So far no major drawback observed. Possible problems: the CVMFS repository has to be kept up to date. Need to verify if the VO takes care of that

Acknowledgements

- **Jakob Blomer (CERN) for his support with CVMFS**
- **Hubert DeGaudenzi (CERN) for his support with LHCb installation tools**

Thank you for your attention!

Questions?

Feedback to lanciotti@pic.es