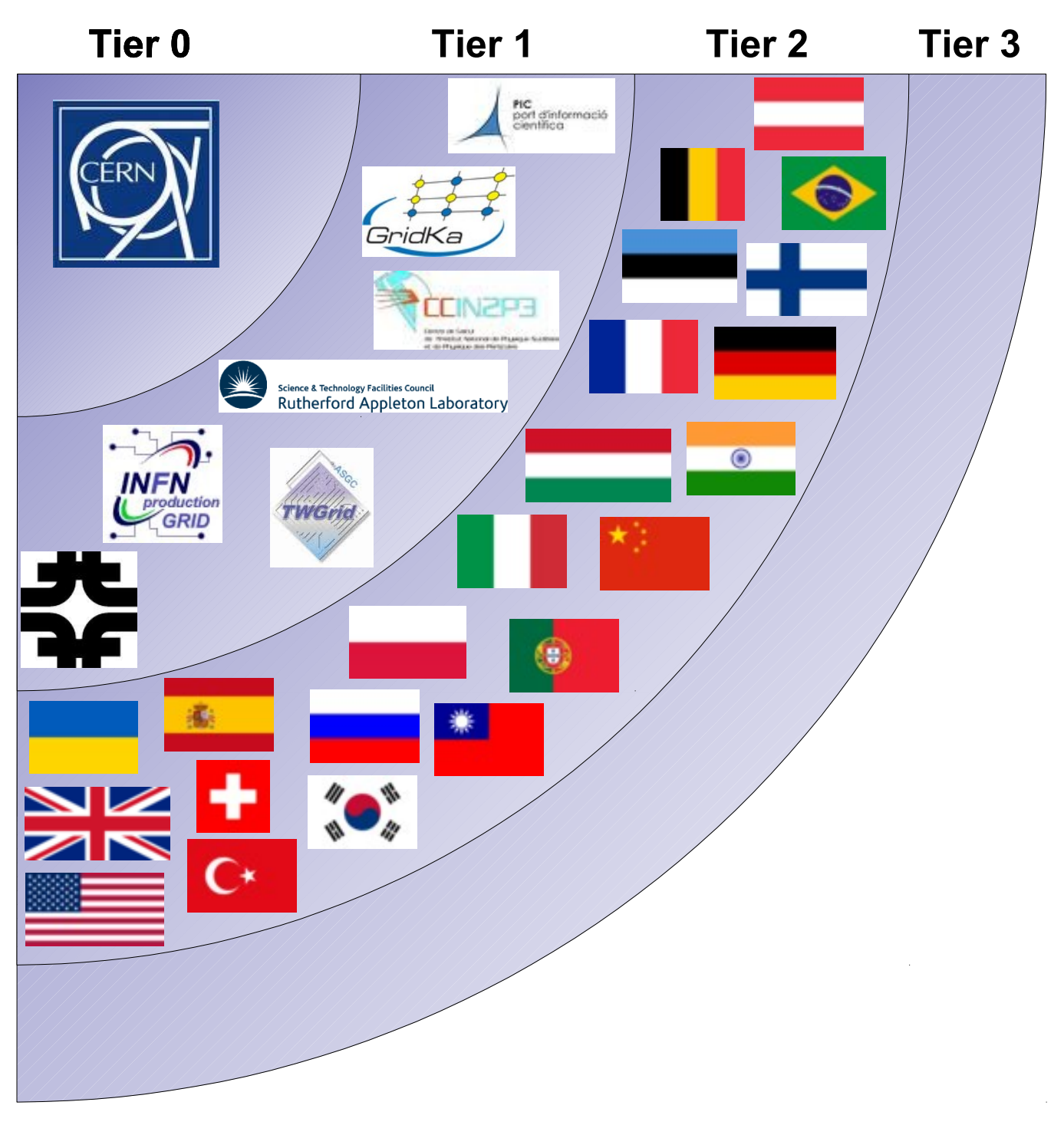
Deployment of the CMS Software on the WLCG Grid.

Wolf Behrenhoff, Christoph Wissing (DESY), Bockjoo Kim (University of Florida), Stijn Blyweert, Jorgen D'Hondt, Joris Maes, Michael Maes, Petra Van Mulders, Ilaria Villella (Vrije Universiteit Brussel), Lukas Vanelderen (Universiteit Gent)

CMS Computing Infrastructure

The tiered CMS computing infrastructure is distributed among more than 50 compute centres in 22 countries.

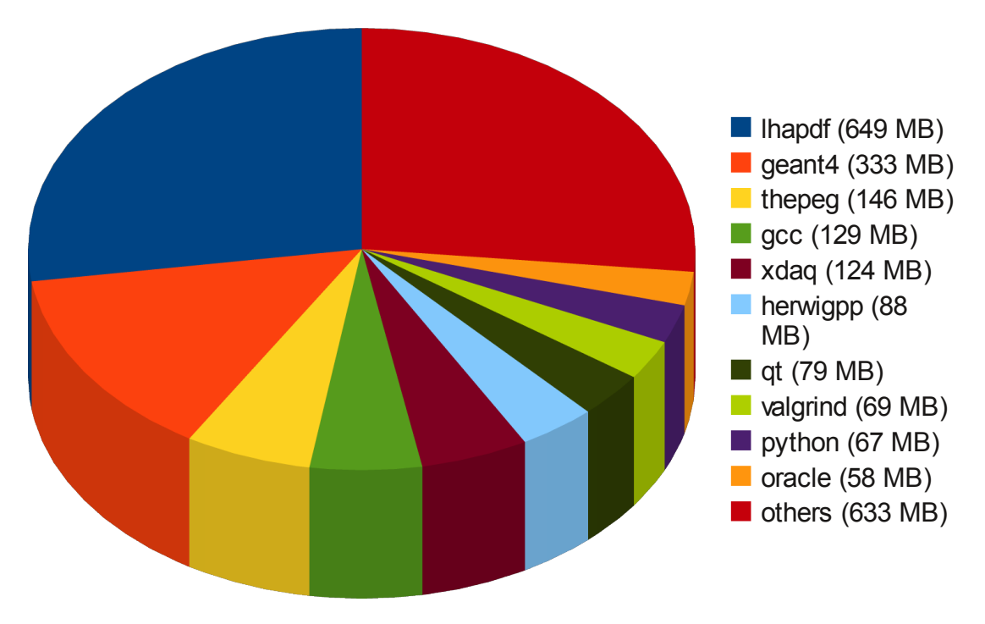

Tier 0
Tier 1
Tier 2
Tier 3
CERN
PIC port d'informació científica
GridKa
CCIN2P3
Science & Technology Facilities Council Rutherford Appleton Laboratory
INFN production GRID
ASGC TWGrid

The different tiers are used for the following purpose:
- Tier 0 – prompt reconstruction, store data and export to T1s
- Tier 1 – Re-reconstruction, long term storage of data
- Tier 2 – MC production, user analysis
- Tier 3 – MC production, user analysis

All centers need CMS software!

Packaging of CMSSW

Size of a single CMSSW version (3_6_0)
- 5.5 GB of files
- 114,598 files and 4,197 symlinks in 11,115 directories
- 4.8 million lines of code: *.cc: 2.5M, *.h: 1M, *.py: 0.8M
- 2.4 GB externals in 83 packages:



There are different types of CMS software upgrades: patch releases are small (a few 100 MB) updates to an already installed release, they usually share all of the external dependencies. Minor releases share most external software with the previous release while new major releases come with a lot of new external packages and thus need almost as much additional disk space as the installation of a single release. The CMS collaboration requires sites to provide 200 GB of disk space for CMSSW. Currently (Oct 2010) the size of all production releases amounts to a total of 60 GB.

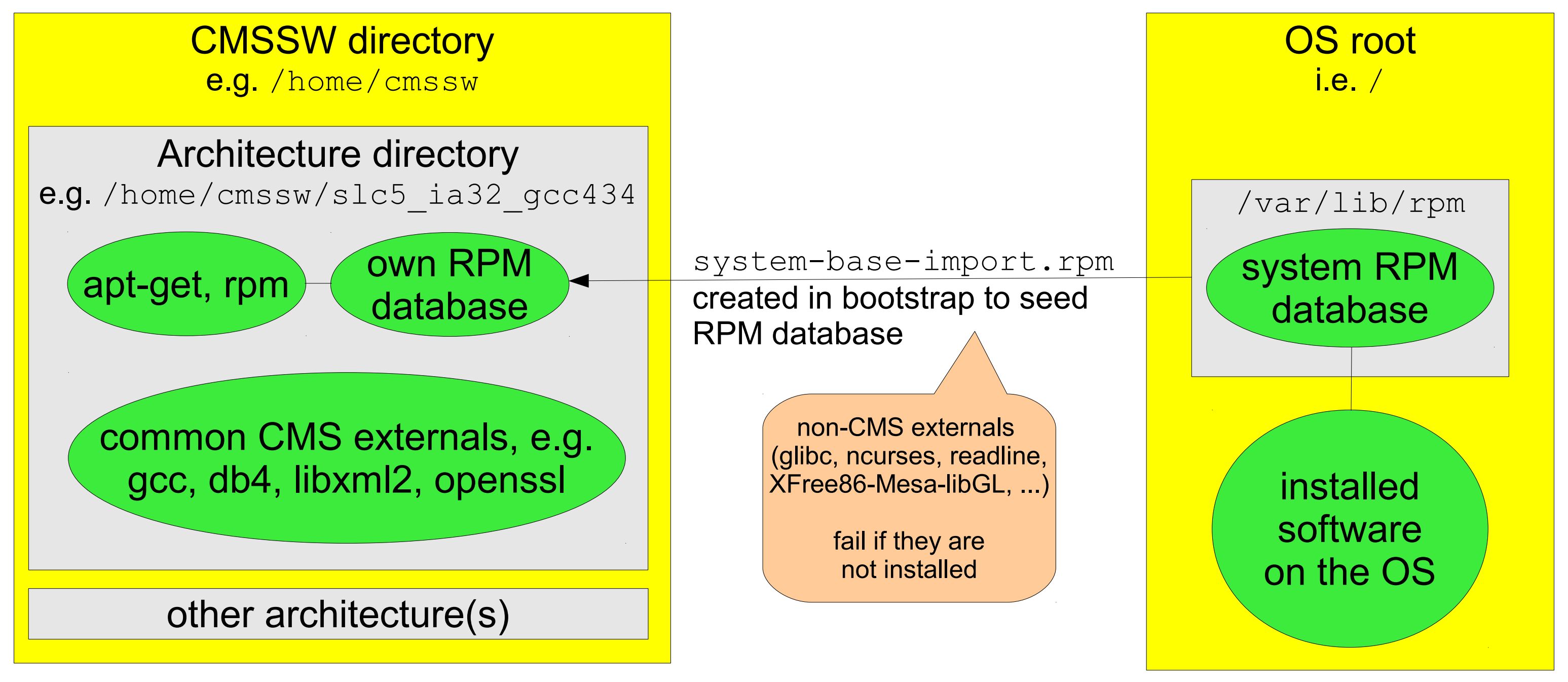
Packaging using rpm and apt-get
CMS software and required external libraries are provided in packages in the RPM format to allow for easy handling of dependencies using the Advanced Packaging Tool (apt).

No stand-alone grid jobs
Because the CMS software is large and monolithically packaged, each CMS computing centre needs its own software installation. It is not feasible to create stand-alone grid jobs which contain a complete CMSSW installation.

Installing CMS Software (CMSSW)

General
Installation of CMSSW is done in two steps: the bootstrap which needs to be done once per architecture and the actual installation of the CMS Software which needs to be repeated for every version of the software. The whole process needs to be run under a normal user account.

Bootstrap
The bootstrap script must be run first. It creates the directory structure for a given architecture (OS and compiler version, 32/64 bit) and creates a CMS internal RPM database. CMSSW has some system dependencies – the bootstrap collects information about installed packages of the operating system and creates *system-base-import.rpm* which provides these dependencies. This RPM file is then installed in the CMS RPM database. The bootstrap also installs a few basic packages such as apt-get.

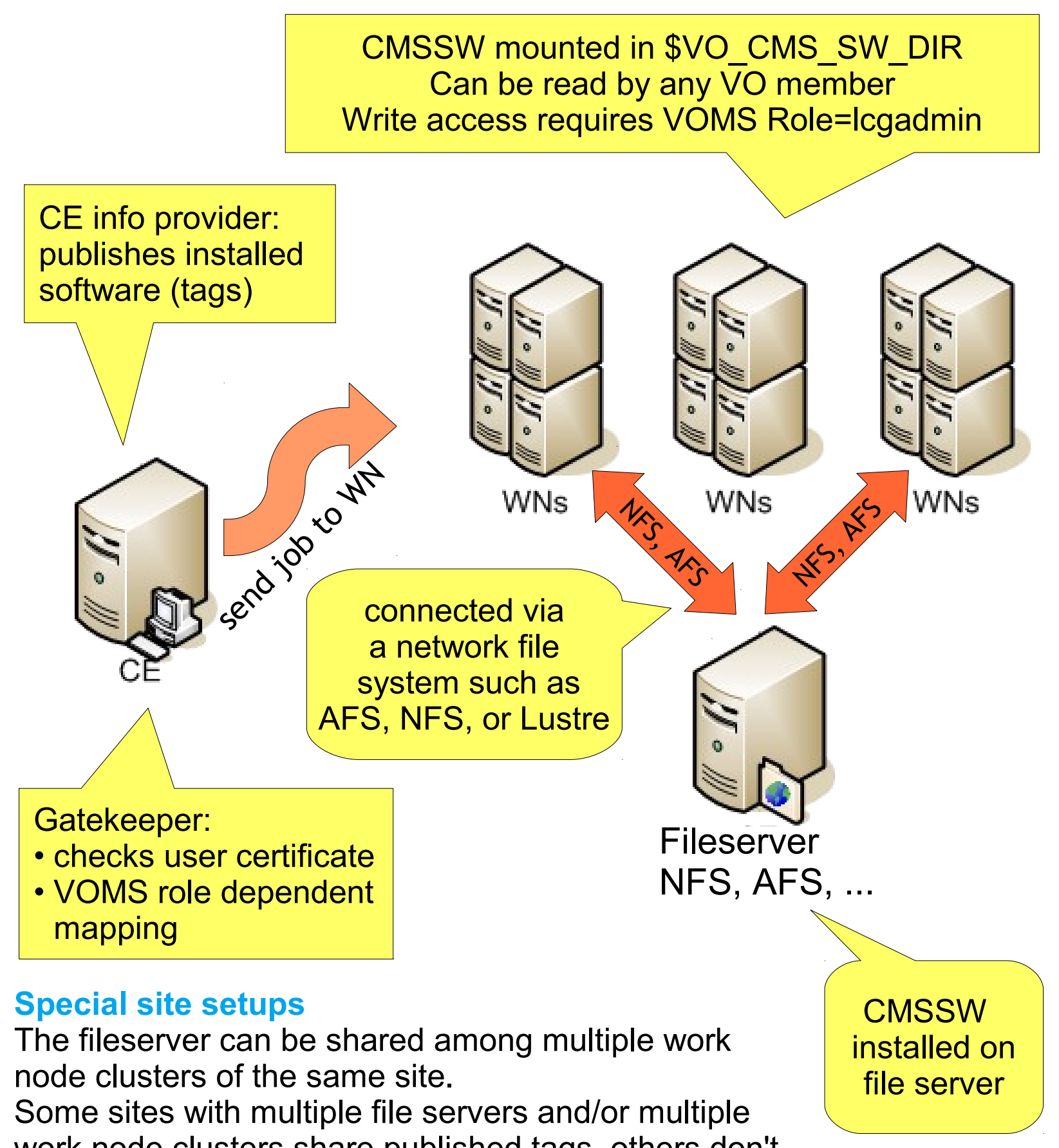

CMSSW directory
e.g. /home/cmssw
Architecture directory
e.g. /home/cmssw/slc5_ia32_gcc434
apt-get, rpm
own RPM database
common CMS externals, e.g. gcc, db4, libxml2, openssl
other architecture(s)
system-base-import.rpm
created in bootstrap to seed RPM database
non-CMS externals (glibc, ncurses, readline, XFree86-Mesa-libGL, ...)
fail if they are not installed
OS root
i.e. /
/var/lib/rpm
system RPM database
installed software on the OS

Installing and removing CMSSW releases
After the bootstrap CMSSW releases can be installed. This is done by setting up the environment so that the CMS apt-get and the CMS rpm database is used. The final step is to run apt-get and let it install a certain CMSSW release including all dependencies.

Removing a CMSSW release is done using a remove script which takes care of all dependencies. The CMSSW release to uninstall and dependent packages which are not required by any other main CMSSW release are removed.

CMSSW on Grid sites

CMSSW on the Grid
The CMS software is deployed to the grid sites centrally unless requested otherwise. The deployment team takes care of problems, contacts a site only if needed (e.g. permission errors). Due to differences in the grid middleware, there are two deployment teams, one for OSG sites (North and South America) where installation is done direct installation on the computing element, and one for gLite and ARC sites (Europe and Asia) where the installation is done via a Grid job from a worker node.

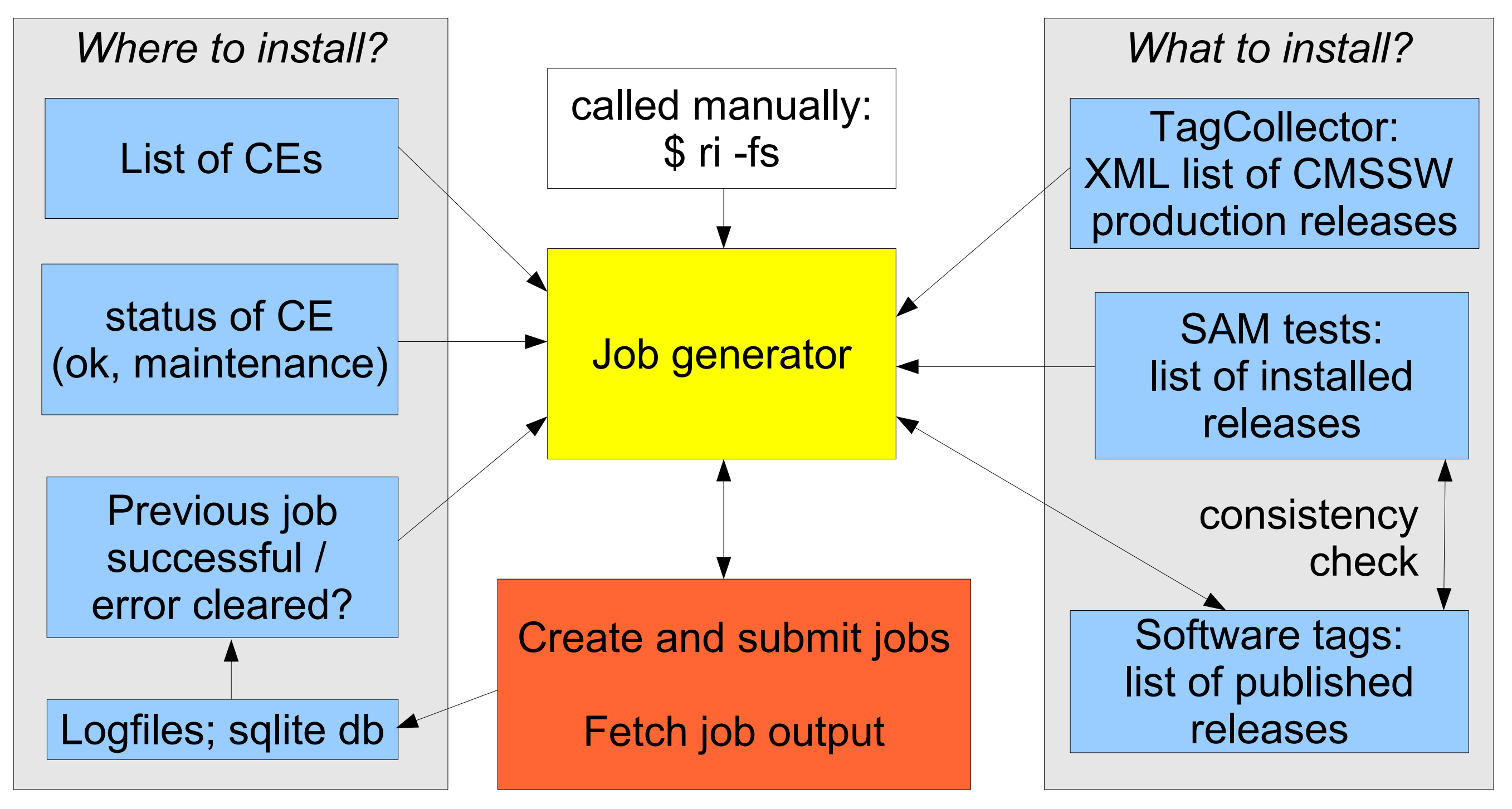

CMSSW mounted in $VO_CMS_SW_DIR
Can be read by any VO member
Write access requires VOMS Role=lcgadmin
CE info provider: publishes installed software (tags)
WNs
WNs
WNs
send job to WN
NFS, AFS
NFS, AFS
CE
connected via a network file system such as AFS, NFS, or Lustre
Gatekeeper:
• checks user certificate
• VOMS role dependent mapping
Fileserver
NFS, AFS, ...
CMSSW installed on file server

Special site setups
The fileserver can be shared among multiple work node clusters of the same site.
Some sites with multiple file servers and/or multiple work node clusters share published tags, others don't.

Automated Deployment – gLite and ARC

The deployment framework is built around a central grid job generator (yellow). If called, it can figure out automatically which installation jobs need to be sent to which computing element (CE). It receives a manually maintained list of CEs, and checks the status of the CE. An installation job is sent only if no other installation job is running and if the CE is not in maintenance and if a previous job was successful.

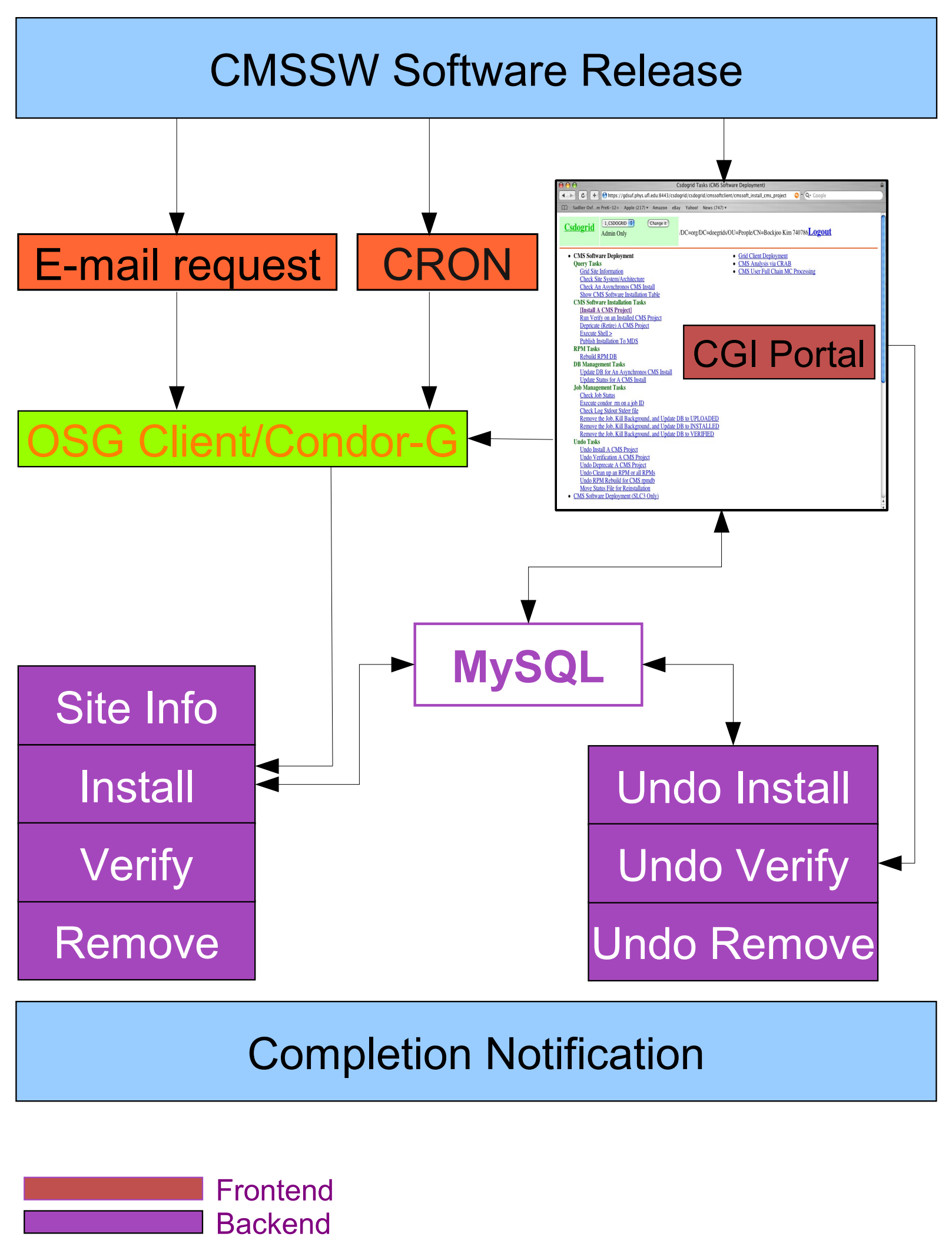
The job generator then compares the list of available production releases from the TagCollector with the list of installed releases on the CE. If the lists are different, it sends a grid job to install missing releases or to remove deprecated releases.


Where to install?
List of CEs
status of CE (ok, maintenance)
Previous job successful / error cleared?
Logfiles; sqlite db
called manually:
$ ri -fs
Job generator
Create and submit jobs
Fetch job output
What to install?
TagCollector: XML list of CMSSW production releases
SAM tests: list of installed releases
consistency check
Software tags: list of published releases

Manual additions to the automated system
- List of CEs: There is no reliable way to determine the list of centrally managed CEs. Some sites don't want central installation, sometimes “test CEs” exist, entries in the CMS SiteDB might be out of date, …
- List of releases: While the list of production releases is available, some sites additionally request certain releases to be installed. So a list of additions is maintained.

Special use cases for the Job Generator
In case of problems or certain special site requests, it can be required to send jobs manually. The job generator supports to send user-defined shell scripts as well as installation and removal requests.

Install Automation/Grid Portal for OSG Sites



Object Oriented Deployment Framework

The deployment framework is written in well-documented, object oriented Perl code. It consists of 8 classes and several thousand lines of code.

The framework is used for the deployment and for various monitoring web sites.

LcgCE	JDL	CMSSWInstall
Computing Element specific functions	Grid job submission	Manage CMSSW installations

Usage examples - CMSSWInstall:
Bootstrap a new CMSSW area (if not already done) and install a CMSSW release:

```
$sw = CMSSWInstall->new(“arch”, “path”);
$sw->install(“CMS_version”);
```

Correct environment variables, common pitfalls (e.g. fakesystem packages, NFS locking problems) are handled automatically and are hidden from the interface.

Usage example - LcgCE:

```
$ce = LcgCE->new(“grid-ce4.desy.de”);
say “Latest swinst test ”. localtime $ce->samTime();
```

Many CE specific functions such as reading the *swinst* SAM test, reading and setting software tags via lcg-tags and lcg-info, getting the tier and sitename are implemented.

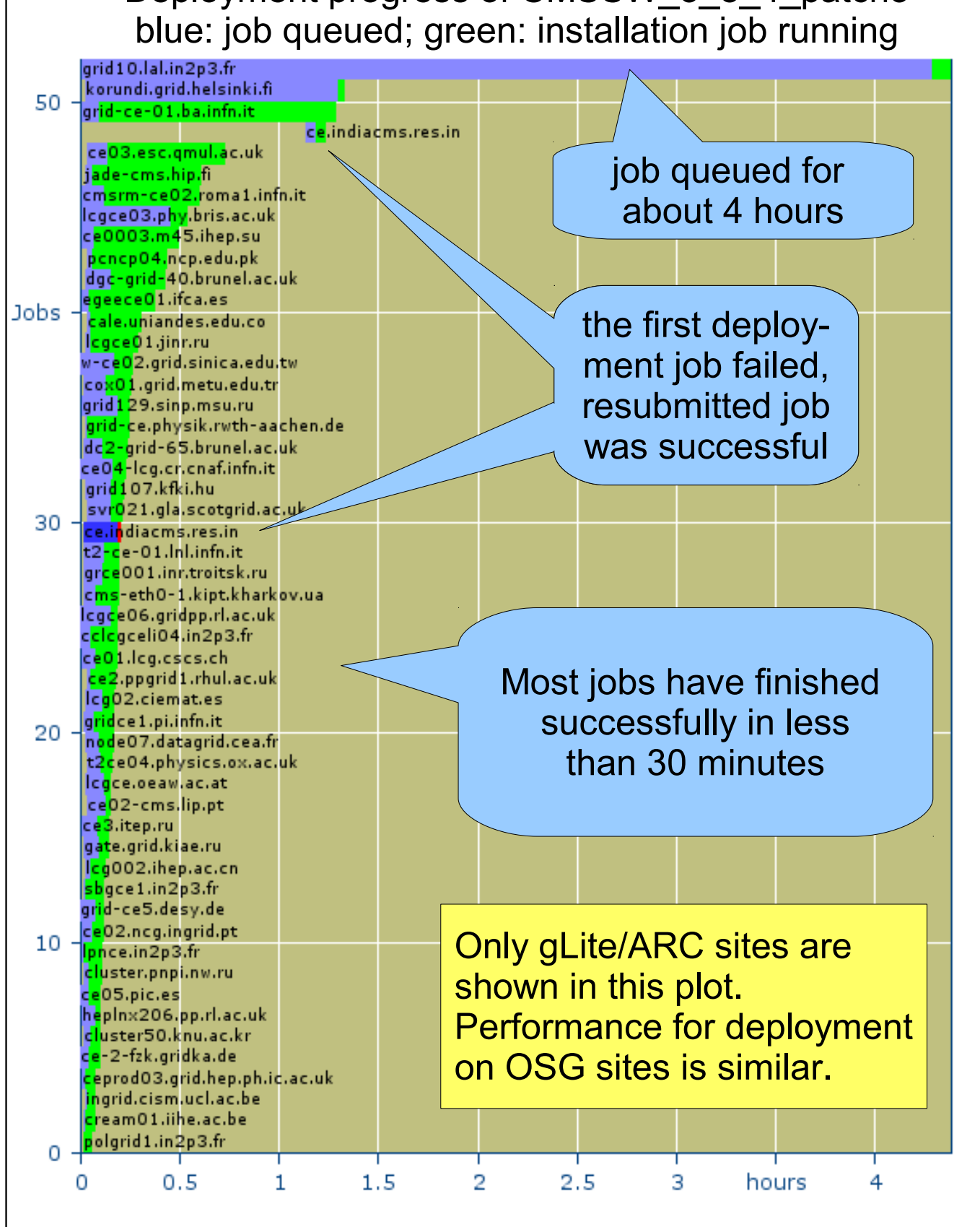
Usage example - JDL:
Submit a file test.sh to a CE, wait for job to finish and display stdout.

```
$j = JDL->new(); $j->addExecutable(“test.sh”);
$j->setRequirement(“...”); $jdl->submit();
sleep 30 while !$j->finished();
$j->retrieveOutput(); say $j->getStdOut();
```

MonALISA reporting can be switched on with a single method call.

Monitoring Deployment Jobs and Statistics

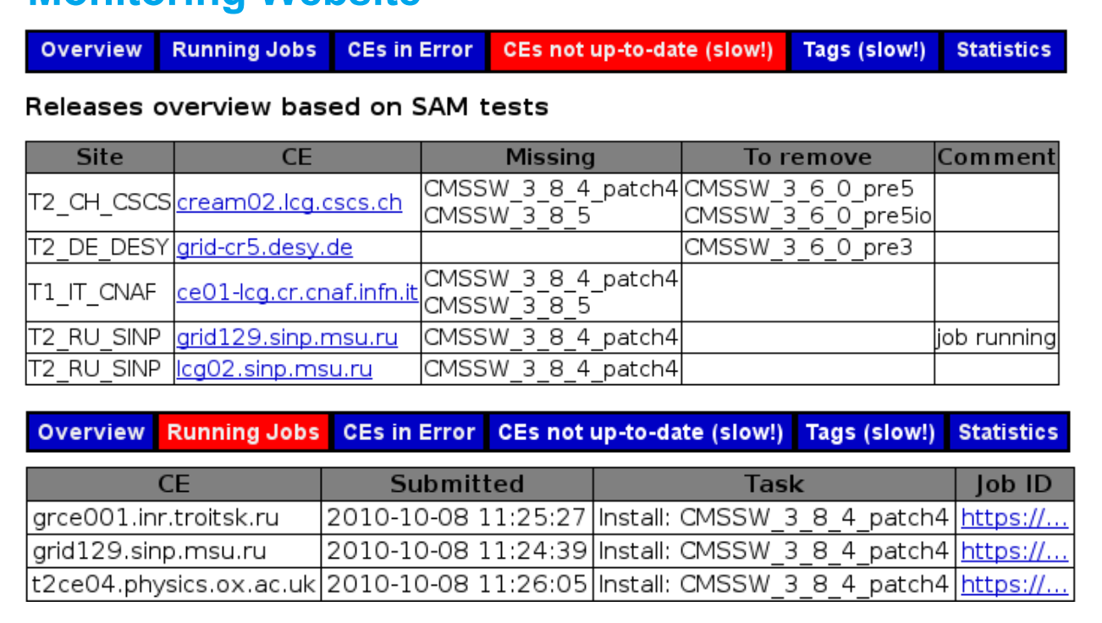
Deployment progress of CMSSW_3_8_4_patch3
blue: job queued; green: installation job running



CMSSW installations
Deployment of a newly released CMSSW version usually takes from about half an hour for patch releases up to about 5 hours for full releases for *most* of the grid sites. Delays up to a few days can be caused by long queues and more importantly by sites in downtime and by errors during the installation.

This year there were about 80 deployment rounds.

Monitoring Website



Overview | Running Jobs | CEs in Error | CEs not up-to-date (slow!) | Tags (slow!) | Statistics

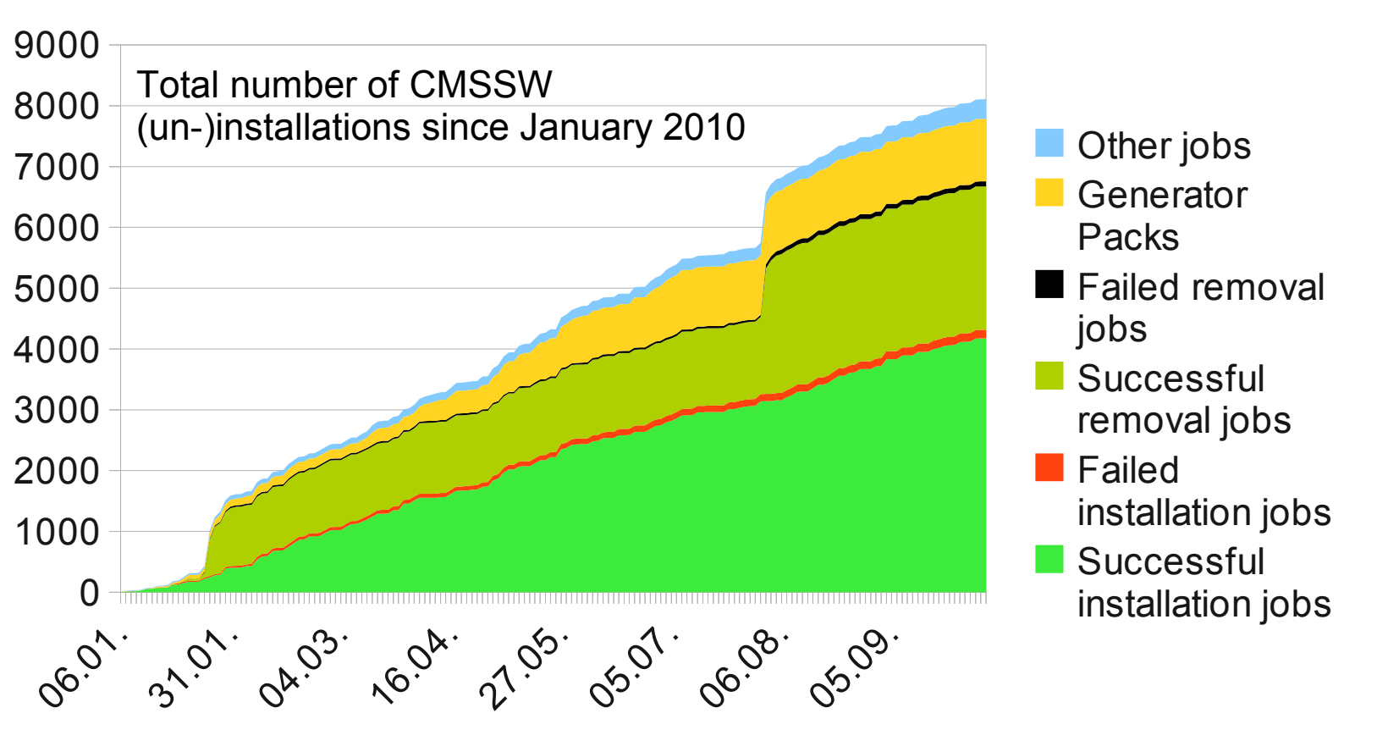
Releases overview based on SAM tests

Site	CE	Missing	To remove	Comment
T2_CH_CSCS	cream02.lcg.cscs.ch	CMSSW_3_8_4_patch4 CMSSW_3_8_5	CMSSW_3_6_0_pre5 CMSSW_3_6_0_pre5io	
T2_DE_DESY	grid-cr5.desy.de		CMSSW_3_6_0_pre3	
T1_IT_CNAF	ce01-lcg.cr.cnaf.infn.it	CMSSW_3_8_4_patch4 CMSSW_3_8_5		
T2_RU_SINP	grid129.sinp.msu.ru	CMSSW_3_8_4_patch4		job running
T2_RU_SINP	lcg02.sinp.msu.ru	CMSSW_3_8_4_patch4		

Overview | Running Jobs | CEs in Error | CEs not up-to-date (slow!) | Tags (slow!) | Statistics

CE	Submitted	Task	Job ID
grce001.inr.troitsk.ru	2010-10-08 11:25:27	Install: CMSSW_3_8_4_patch4	https://...
grid129.sinp.msu.ru	2010-10-08 11:24:39	Install: CMSSW_3_8_4_patch4	https://...
t2ce04.physics.ox.ac.uk	2010-10-08 11:26:05	Install: CMSSW_3_8_4_patch4	https://...

Monitoring deployment status
A monitoring website has been developed to view the status of the deployment by combining information from SAM tests, from the BDII and from the deployment job log files. It does not only show which releases need to be installed or removed on which computing elements but also provides a convenient access to job logfiles in case of failed jobs. When looking at logfiles and searching for problems, the downtimes for the relevant site are shown (available for T1 and T2 only). In addition, links to relevant pages the LCG Savannah system are provided.



CMSSW Removal Procedure

(1) Deprecation proposal in Hypernews
↓ usually +/- one week until final decision
(2) Deprecation announcement, TagCollector updated
↓ next time the job generator is executed
(3) Removal of software tags, no new jobs accepted
↓ 5 days
(4) Send jobs for uninstallation
time

The removal of CMSSW releases on grid sites is done in several steps. At first a proposal to deprecate releases is sent to Hypernews. If there are no objections, the TagCollector is updated accordingly.

However, removal jobs are not sent immediately. Rather the published software tags are removed from all CE info providers. That ensures all running jobs and all jobs which are currently queued can still finish successfully but no new jobs using the deprecated releases can be submitted. After 5 days, actual removal jobs are sent.

The whole removal procedure is implemented in the Job Generator, i.e. it takes care of the delay between removing the tags and sending the removal job.

Usually deprecation proposals are sent 2 or 3 times a year, deprecating a rather long list of releases. Of course, one removal grid job is sufficient to remove all deprecated releases.

HELMHOLTZ | ASSOCIATION
CHEP TAIPEI TAIWAN 2010