Packaging and distribution issues

Flavia Donno, INFN-Pisa

EDG/WP8 EDT/WP4 joint meeting , 29 May 2002

PPARC

ESA

Outline

- Motivations and goals
- The EDG solution
- First evaluation of VDT/PACMAN
  (distributing EDG/UserInterface via pacman)
- Towards a common solution.

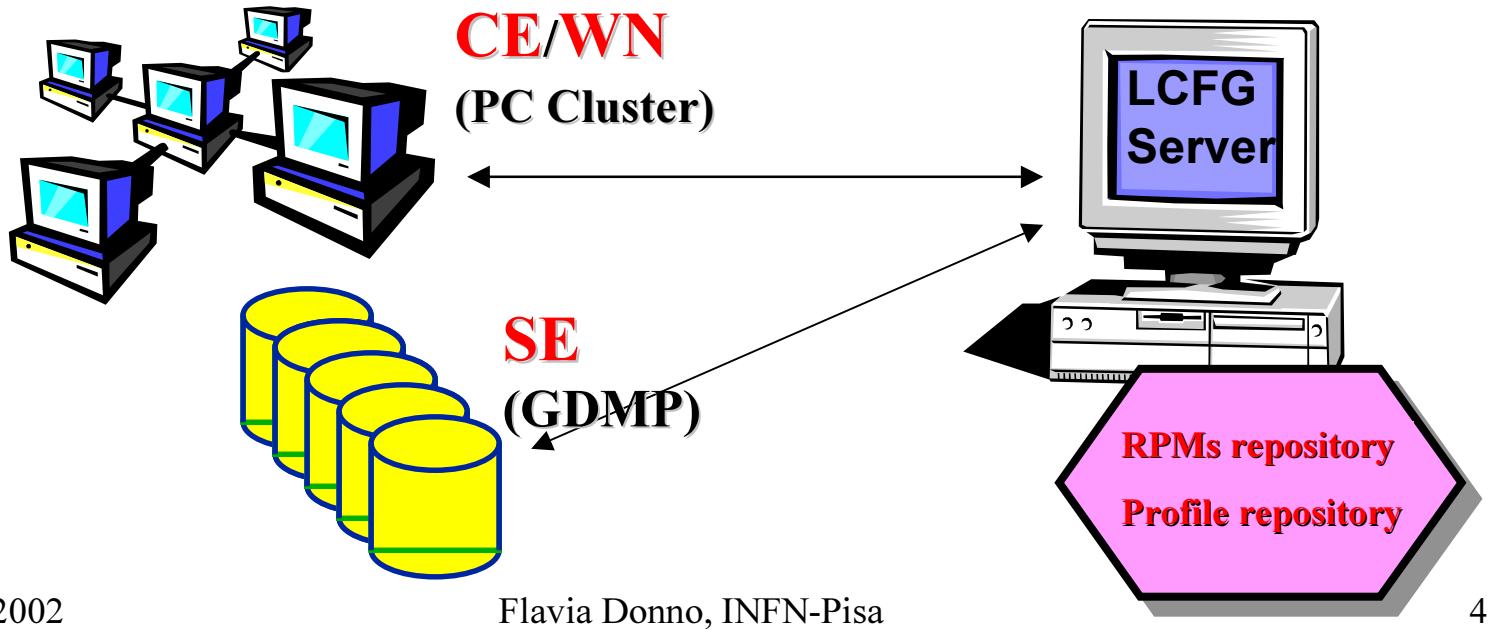
Motivations and goals

- **Deployment of common middleware tools**
  - Installation/configuration of middleware software

    (Which releases ? What components ? Relocatability ? …)
- **Installation/configuration of application software**
  - Variety of packaging (cmt, scram, softreltools, …) and distribution/deployment/configuration tools (rpm, pacman, dar, upd, …)
- **Support for deployment on desktops, departmental clusters, production farms**
  - EDG has a solution for managing “large” PC farms
- **Definition of requirements/recommendations**
  - Focus on distribution/deployment tools (pacman, upd, …)
  - Support for multiple packaging tools (?)
  - Modularity, Relocatability, Flexible Configuration
  - Versioning, publishing dependencies, etc.
  - Interoperability with “fabric management” tools (?)

The EDG solution

- **EDG/WP4 LCFG: "grid fabric management" tool**
  - Based on LCFG originally developed by Edinburgh University by Paul Anderson and Alastair Scobie to automatically manage "clusters" of machines (300 nodes).
  - A site consists of one or more "**software and profile servers**" and a number of "**clients**". Both clients and server can be **automatically configured** starting from one of the profile servers. The first profile server needs to be installed/configured manually.



The EDG solution: LCFG

- **Configuration files** (in XML format) are **distributed** to clients **via HTTP**. The **software repository** (**RPM** bundles) is **served via NFS** to the client from one of the servers.
- Configuration files describe **machines types** (CE, SE, WN, …). A machine type is defined by a list of RPMs and LCFG configuration objects (nfs, globus, gdmp, etc.). Inheritance is also supported.
- **Modularity** of LCFG objects for services/software configuration.
- Not only software but also **system management**: accounts, services, security, etc.
- The software **installation** step and the **configuration** step are kept separate for a better control.
- Tools are available for
  - starting/stopping/reconfiguring services,
  - uninstalling, downgrading, automatically updating rpms (**updaterpms**),
  - keeping the configuration homogeneous and coherent.

The EDG solution: LCFG

Drawbacks:

- LCFG is a powerful system management tool, not a distribution/deployment tool.
- It needs a local software/configuration repository
- Full control of the machine (LCFG light).
- Based on RPMs only but very modular (versioning, run-time dependencies, etc.)
- %Postinstall step disabled during rpm installation. (No relocatability in EDG for the moment, specific user environment and service configuration solutions).
- Configuration and Interdependency issues need to be addressed by the repository manager (through LCFG objects).
- It does not address the one-time installation need (EDG Userinterface)
- …

- ➔ A lot to learn from it for what concerns configuration management issues.

First evaluation of VDT/PACMAN

- **pacman is a package manager**
  - **you can transparently pull from a local or remote repository (http), install and manage software packages.**
- **Packages can be distributed in many forms: tarballs, rpms, …**
- **Pacman hides the details of:**
  - **Where do you get the software from?**
  - **Which version of the software is right for your system?**
  - **Whether there are dependent packages that you have to install first?**
  - **Whether you have to be root or not?**
  - **What are the exact instructions for installing the software?**
  - **How to setup environment variables and paths for the packages once they are installed?**
  - **How to conveniently setup the same environment on multiple machines?**
  - **When a new version of the package is available and when you should upgrade?**

First evaluation of VDT/PACMAN

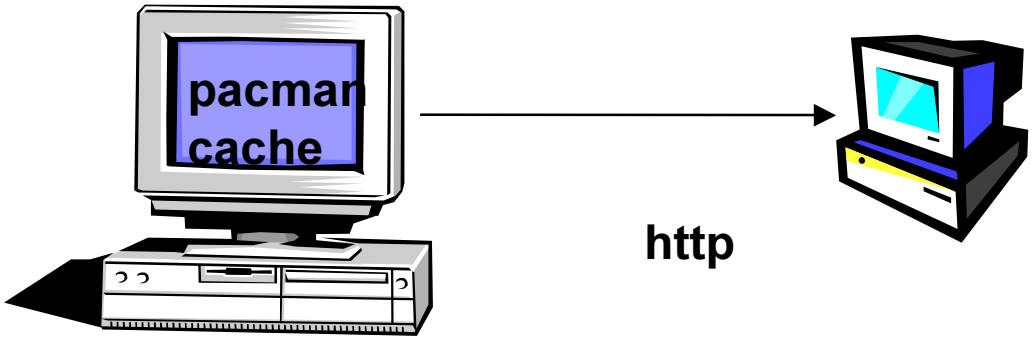
- Install pacman (untar a file + source a script)

- Setup the file (cache_starter) with the URLs of possible pacman caches (=software repository)

- Issue user commands:
  - `pacman -fetch <package>`
  - `pacman -install <package>`
  - `pacman -update <package>`
  - `pacman -uninstall <package>`
  - `pacman -remove <package>`

  - `pacman -local`

First evaluation of VDT/PACMAN



Package	Description	Installed?	Update?	Depends on	Daemons	Date fetched
ca_CERN -0.6 -1	EDG CERN Certification Authorities	yes	-	-	-	Mon Apr 22 17:39:51 2002
ca_CESNET -0.6 -1	EDG CESNET Certification Authorities	yes	-	-	-	Mon Apr 22 17:39:52 2002
ca_CNRS -0.6 -1	EDG CNRS Certification Authorities	yes	-	-	-	Mon Apr 22 17:39:52 2002
ca_CNRS -DataGrid -0.6 -1	EDG CNRS - DataGrid Certification Authorities	yes	-	-	-	Mon Apr 22 17:39:53 2002

First evaluation of VDT/PACMAN

Example of a *.pacman file:

```
name        = 'edg-userinterface-1.1.4'
description = 'EDG UserInterface Package'
url         = 'http://marianne.in2p3.fr/'
source      = ''
systems     = { }
depends     = ['edg-ui-external-1.1.4', 'edg-compiler-1.0-0',  \
              'ca_EDG-0.6-1', 'edg-rgma-2.2.3-1',              \
              'globus_edg_ui-2.0-21', 'grm-1.0.2-1',           \
              'edg-profile-0.3-1', 'edg-user-env-0.3-1',       \
              'edg-utils-1.0.14-1', 'userinterface-1.1.2-1',   \
              'userinterface-profile-1.1.2-1',                 \
              'workload-profile-1.1.2-1'
             ]

exists      = []
inpath      = []

bins        = []
paths       = []
enviros     = []
localdoc    = ''
daemons     = []
install     = { }
setup       = []
demo        = ''
```

source="http://datagrid.in2p3.fr/distribution/datagrid/wp1/RPMS/'
systems = { 'linux-i386': ['edg-userinterface-1.1.4.i386.rpm','edg-userinterface-1.1.4']}
Install={'root': ['./configure-userinterface']}

First evaluation of VDT/PACMAN

- Tested with EDG RPMs (EDG UserInterface)
- No check of dependencies for packages not installed with pacman
- No way to supercede default rpm installation options per package.
- Uninstalls only “dummy superpackage”
- No separate configuration step. Limited environment setup.
- Versioning not directly supported.
- Update = uninstall + install. It refreshes the internal packman database.
- No uninstall target.
- Manual generation of pacman files. How about an rpm->pacman conversion tool ? Attempt from Atlas ?

  (http://www.usatlas.bnl.gov/computing/software/pacman/ACF-cache/rpm2pacman)

- Fetching restarts from the beginning if an error occurs.
- Installation blocks if rpm already installed.
- Pacman database gets easily corrupted.

First evaluation of VDT/PACMAN

- Pacman offers a nice layer of abstraction vs existing packaging tools
- It allows for distributed management of a pacman cache (and so for software installation and configuration)
- It is easy to use at a user level and at a cache manager level (but quite limited at the moment).

But:

- Does it really offer a solution to the needs we have ?
- Can it be integrated into LCFG ? What are the requirements ? How about other management tools ?
- We still have to address the problems of versioning, interdependency, publishing of package metadeta info, configuration issues, etc.
- How about software distribution and VDT/EDG releases ?

Towards a common solution

We have:

- We have examined the EDG LCFG and VDT PACMAN tools
- They respond to different requests and offer “limited” solutions.

How do we proceed in DataTAG ?

We will:

- Continue the study of existing tools feeding back requirements to the developers - we are working with people from the Condor team.
- Report on existing technologies and tools
- Write a recommendations/requirements document (by the 15th of June)
- Feed our input back to GLUE and LCG for a common (?) solution.
- Create an experimental distribution with EDG and VDT software attempting to sort out configuration/installation issues.