A centralized administration of the Grid infrastructure using Cfengine

Tomáš Kouba

Varna, NEC2009

Introduction

- **Typical grid computing center consists of:**
  - Many computers
  - Many services
    - Local
    - Grid enabled
  - Many users
  - A few working administrators
- **Administrator wants to minimize the effort to run services on long-term basis**

Introduction (cont.)

- **Computers and services need to be installed and configured**
- **Configuration steps should be documented**
- **Configuration files should be under version control**
- **Services/computers of the same kind should be configured automatically**
  - Administrator wants to do the work once (for every service)

Real numbers in Prague

- **2 sites, one big, one small with many services**
  - praguelcg2
    - 2 batch system servers (PBSPro, Torque)
    - 4 CEs, 1 SE (DPM) with 5 disk servers
    - ~330 WNs
    - 2 Alice voboxes
    - Essential services for the network
      - *monitoring, DNS, DHCP*
      - *LDAP, syslog*
    - backup server with tape backend
    - XEN hosts

Real numbers in Prague (cont.)

- prague_cesnet_lcg2
  - 10 WNs, but many essential services for VOs VOCE and Auger
  - 1x CE, 1x SE
  - 1x myproxy, 1x LFC
  - 1x LB, 1x top level BDII
  - 2x WMS, 2x VOMS server
  - 1x belle VO postgresql server

Possible solutions

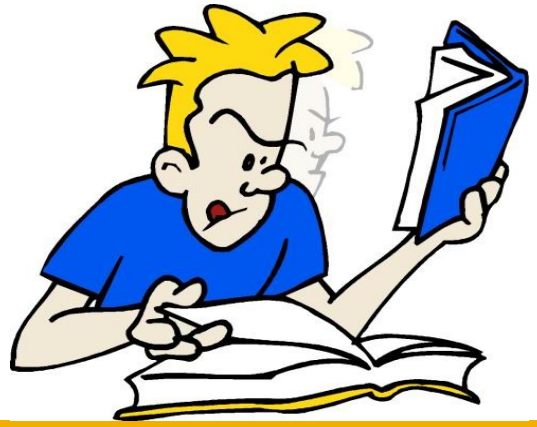
- **Snapshot images**
  - Configure one node, make an image, install the image
  - http://wiki.systemimager.org/
  - SGI Tempo
- **Bunch of shell scripts**
  - Many disadvantages, difficult to keep systematic
- **Systems for configuration description**
  - Cfengine, Puppet, Quattor

Why Cfengine

- Support for multiple UNIX flavours
- Generic (not only grid-oriented)
- Long development tradition (1st production version in 1993)
- Existing implementation on similar sites (scotgrid)
- Big community
- Optional commercial support

Basic philosophy

- **Shell scripts vs. Cfengine is like procedural vs. non-procedural programming**
  - C vs. Prolog
- **You describe the state not the steps**
- **One line in Cfengine language => many actions taken**
- **As my colleague says: “It's simple but It requires a bit different thinking”**
  - Can result in a steep learning curve



Basic philosophy (cont.)

- **The main buzz word is “class”**
  - Class is anything that describes a host
    - eg. hostname, required service (dhcp client), OS
- **Cfengine client**
  - Defines implicit classes
  - Reads the configuration (adds classes defined by the administrator)
  - Tries to fulfill the described state
    - eg. installs package, restart daemon

What Cfengine can do

- **Install software (in OS-dependent style)**
- **Edit configuration files (in declarative way)**
  - You define lines that should be present not the whole file
- **Copy files, check for running processes**
- **Execute generic shell commands**
- **Define classes depending of a result of other action**
  - This launches a cascade of actions
- **Define classes on arbitrary condition**
  - E.g. file change, modification time of a file
- **Cfengine client can be launched from cron or manually**

What Cfengine cannot do

- **It does not resolve deep dependencies**
  - Only one level of dependencies is recommended
  - Solved by multiple passes of the client
- **It does not force the administrator to stick to a consistent configuration**
  - It means chaotic configuration is possible
  - Should you configure ssh of UI in file describing ssh service or file describing UI node?
  - It is a good practice to define formal rules (coding style, service level separation) and force administrators to stick to it

Typical work before Cfengine

- **Security team decides to forbid ssh connection using password authentication.**
  - allow only ssh keys
- **On one machine, it means editing one file, issuing one service-reload command.**
- **On a bunch of machines it means for cycle with ssh and doing the same.**
  - editing then means sed-ing or copying new config file
- **Many problems**
  - offline machines, special machines, machines reinstalled afterwards

Typical work with Cfengine

- **The same thing takes a bit more work in Cfengine:**

```
editfiles:
   prague_cesnet_lcg2::
      { /etc/ssh/sshd_config
         HashCommentLinesMatching "^PasswordAuthentication.*yes$"
         AppendIfNoSuchLine "PasswordAuthentication no"
         HashCommentLinesMatching "^PubkeyAuthentication.*no$"
         AppendIfNoSuchLine "PubkeyAuthentication yes"
         DefineClasses "proc_service_sshd"
      }
shellcommands:
   scientific.proc_service_sshd::
      "/sbin/service sshd restart" umask=022
      "/sbin/chkconfig --level 2345 sshd on" umask=022
```

Typical work with CFEngine (cont.)

- **Submit changes to version control system (SVN in our case)**
- **Wait for cron to execute the cfengine or make it run via cfrun from the master server**
- **If the change is too invasive, the cfengine can be run on one machine only**
  - so the changes are properly tested
- **All the "additional" work pays off**
  - we have readable description of the change
  - our change is automatically checked for presence
  - other advantages coming from version control

Examples

- **https://savannah.cern.ch/bugs/?42592**
  - default RFIO port is wrong, you need to explicitly define it in /etc/shift.conf

```
editfiles:
   scientific_sl_5|scientific_slc_5::
      { /etc/shift.conf
         AutoCreate
         BeginGroupIfNoSuchLine "RFIO PORT"
            DeleteLinesStarting "RFIO PORT"
            Append "RFIO PORT 5001"
         EndGroup
      }
```

Questions and Links

- Q?
  - maybe
- A!

- Links
  - http://www.cfengine.org/
  - http://euvethker.blogspot.com/2005/12/cfengine-best-practices.html