Workshop on the use of quattor for grid configuration

CERN, 26/3/04

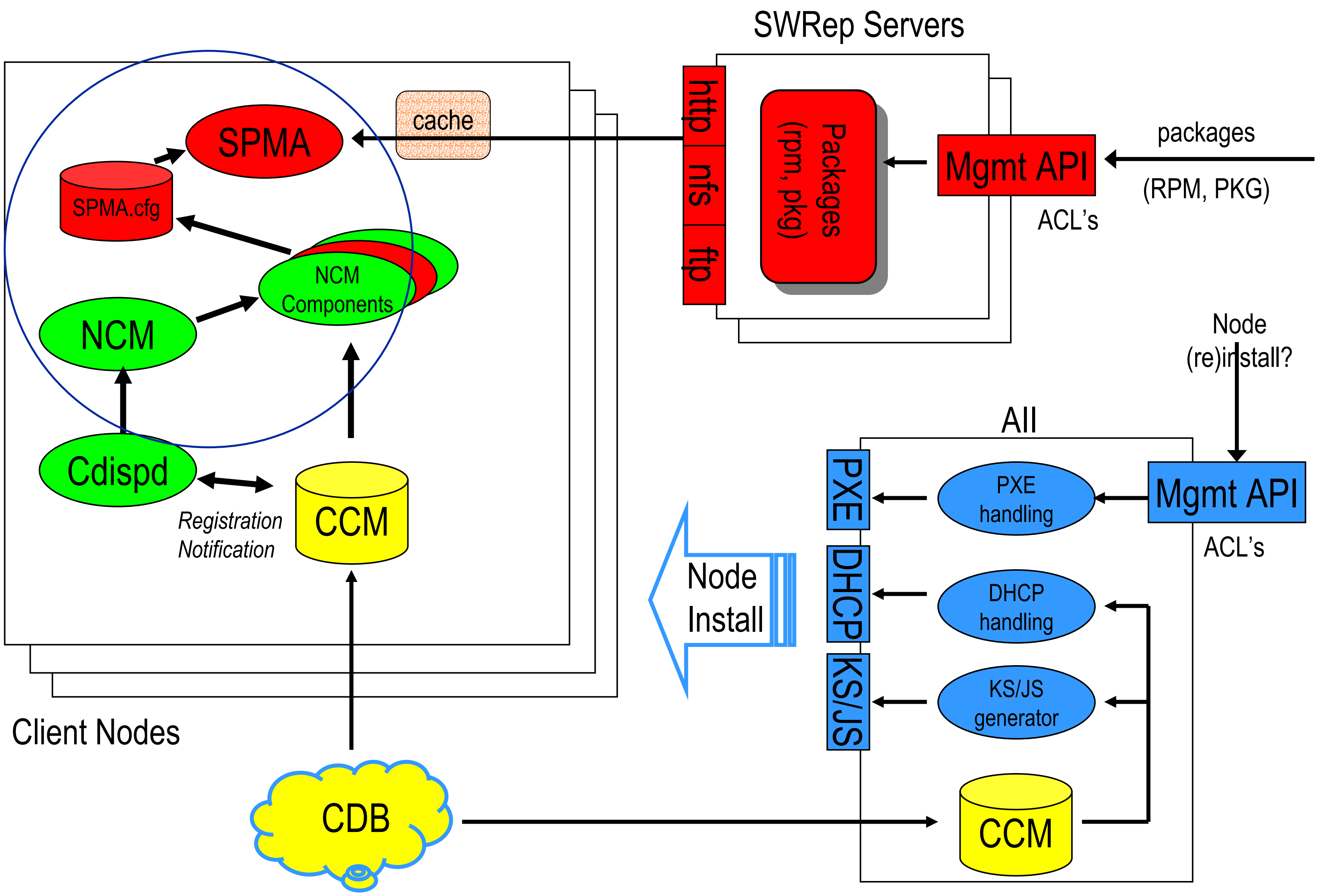
German Cancio

http://quattor.org

outlook

- NCM components: refresh
- Global Schema walktrough: Profile, templates, components
- Cal’s comments on templates

NCM in context



Node Configuration Management (NCM)

- Client software running on the node which takes care of "implementing" what is in the configuration profile
- Modules:
  - "Components"
  - Invocation and notification framework
  - Component support libraries

NCM: Components (I)

- "Components" (like SUE "features" or LCFG 'objects') are responsible for updating local config files, and notifying services if needed
- Components *register* their interest in configuration entries or subtrees, and get invoked in case of changes
  - Components can also be run manually, or via cron
- Components do only *configure* the system
  - Usually, this implies regenerating and/or updating local config files (eg. `/etc/sshd_config`)
- Use standard system facilities (SysV scripts) for *managing* services
  - Components can notify services using SysV scripts when their configuration changes.
- Possible to define configuration dependencies between components
  - Eg. configure *network* before *sendmail*
  - Components won't run if a pre-dependency is unsatisfied

NCM: Components (II)

- Components are written in **Perl**
- Each component can implement two methods:

- **`Configure()`:**
  - typically invoked on startup, or when there was a CDB configuration change
  - *Mandatory method*
- **`Unconfigure()`:**
  - invoked when a component is to be removed
  - *Optional method* – most of the components won't need to implement it.

Component (simplified) example

```
sub Configure {
  my ($self,$config) = @_;
  # access configuration information
  my $arch=$config->getValue('/system/architecture’); # NVA API
  $self->Fail (“not supported") unless ($arch eq ‘i386’);
  # (re)generate and/or update local config file(s)
  open (myconfig,’/etc/myconfig’); …
  # notify affected (SysV) services if required
  if ($changed) {
    system(‘/sbin/service myservice reload’); …
  }
}
```

NCM: Components (III)

NVA API: configuration access library

- This library allows hierarchical configuration structure access on the client side
- Most popular methods:
  - `$value=$config->`**`getValue`**`('/system/kernel/version');`
  - `If ($config->`**`elementExists`**`($path)) {…} else {…}`
  - `$element=$config->`**`getElement`**`($path);`
  - while loop:

```
while ($element->hasNextElement()) {
    my $newel=$element->getNextElement();
    ...
}
```

NCM support libs

Core functions:

- `$self->`**`log`**`(@array)`: write @array to component's log file
- `$self->`**`report`**`(@array)`: write @array to log and stdout.
- `$self->`**`verbose`**`(@array) $self->debug(@array)`: verbose/debug output
- `$self->`**`warn`**`(@array)`: writes a **`[WARN]`** message, increases # of warnings
- `$self->`**`error`**`(@array)`: writes an **`[ERROR]`** message, increases # of errors

Advanced support libraries available (from CERN's SUE tool):

- Configuration file manipulation
- Advanced file operations
- Process management
- **Exception management libraries**

NCM: packaging components

- Each component is packaged independently
- Improved build tools allow for easier packaging of components
  - No specfile necessary
  - No makefile necessary
- Portability
  - ‘make rpm’ – generates RPM
  - ‘make pkg’ – generates PKG (Solaris)
  - ‘make EDG_LSB=edg xxx’ – use LSB **or** EDG prefixes
- ‘make release’ – generate new version
  - checks in modified files to CVS
  - Prompts for ChangeLog entry
  - Generates a new CVS tag for the component

NCM: tools

- `ncm-ncd` (Node Configuration Deployer):
  - framework and front-end for executing components (via cron, cdispd, or manually)
  - Dependency ordering of components
- `cdispd` (Configuration Dispatch Daemon)
  - Monitors the config profile, and invokes registered components via the `ncm-ncd` if there were changes
- `ncm-query`
  - Tool for examining configuration information as cached on the node
- More details in
  - NCM design document http://edms.cern.ch/document/372643
  - NVA API tutorial
  - NCM component writer's guidelines

Global schema

1. How does the global schema look like

- Best is to go trough a complete example. See the dump of a LCG-2 Worker Node at CERN. Rafael: cdispd extensions.

2. How is this information extracted by components

- Go trough the following components: grub, RM, spma

3. How is the global schema generated out of PAN templates

- Propose to go trough the SYSTEM and SOFTWARE branches of that specific example, in reverse order (starting at the object profile). HARDWARE branch: if time permits.
- Already highlight and write down issues and improvement suggestions! (See Cal's mail)