Extending Functionality Through the Rocks Command Line
It’s Python Time!
Motivation

◆ Lack of consistency in Rocks commands
  ✅ add-extra-nic (15 flags)
  ✅ 411put
  ✅ rocks-dist
  ✅ dbreport (~ a dozen reports)

◆ Extensible to other groups
  ✅ How do I add a flag to an existing command?
  ✅ How do I add a new command?
  ✅ How do I document my command?
Motivation

   [-list-rcfiles] [-list-project-info] [--verbose] [--dump] [--del] [--list]
   [--password password] [--db database] [--user host]
   [--if interface (default: eth1)] [--mac mac address]
   [--module linux driver module name] [--ip ip address]
   [--netmask netmask (default /24)] [--gateway ip address of gateway]
   [--name hostname on new interface] [--site client ip] node

   [-d dirname] [-g path] [-l lang] [-r release] [--help] [--list-rcfiles]
   [--graph-draw-order] [--graph-draw-edges] [--graph-draw-key] [--graph-draw-all]
   [--notorrent] [--rcfile arg] [--host host] [--password password]
   [--db database] [--user host] [--arch architecture] [--comps path]
   [--dist dirname] [--graph-draw-size arg] [--graph-draw-format arg]
   [--mirror-dir dirname] [--mirror-host hostname] [--root dirname]
   [--cdrom /mnt/cdrom] [--with-roll rollname-rollversion]
   [--path single path item] command
Available commands:
dist dvd makecontrib makesitenodes copycd usb copyroll cdrom paths graph dist2mirror

© 2008 UC Regents
Goals

- Consistent
  - Interface
  - Argument parsing
  - Usage / Help

- Extensible
  - Easy to add commands (3rd party rolls)
  - Easy to modify commands

- Easy to guess the right command

- Purge all –flags from Rocks (see previous slide)

- Inspired by Trac
Verb Based

- “add”, “set”, “enable”, …
  - Modify the cluster database
- “list”, “dump”, “report”
  - Inspect the cluster database
- About 20 verbs in the command line so far
- You can even add your own
Grammar

- rocks <verb> <object…> <subject> <params…>
  - Object is general to specific
    - “host” “interface”
    - “network” “subnet”
    - “viz” “layout”
  - Subject is typed
    - host
    - appliance
    - Network

Example:

```bash
# rocks list host interface tile-0-0
```

```
<table>
<thead>
<tr>
<th>SUBNET</th>
<th>IFACE</th>
<th>MAC</th>
<th>IP</th>
<th>NETMASK</th>
<th>GATEWAY</th>
<th>MODULE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>private</td>
<td>eth0</td>
<td>00:13:72:ba:be:42</td>
<td>10.255.255.254</td>
<td>255.0.0.0</td>
<td>-------</td>
<td>tg3</td>
<td>tile-0-0</td>
</tr>
<tr>
<td></td>
<td>eth1</td>
<td>00:0a:5e:1a:6d:64</td>
<td>------------</td>
<td>--------</td>
<td>-------</td>
<td>skge</td>
<td>----------</td>
</tr>
</tbody>
</table>
```
Implementation

- **Python**
  - Similar to existing dbreport code
  - Very small modules

- **Command line is identical to the directory hierarchy**
  - Verbs are directories
  - Objects are directories
  - Subjects are `__init__.py` files

- **Commands are added by adding directories**
Rolls Can Add Commands

- Similar to the configuration graph
- Rolls can add command line
  - Files: commands
  - Directories: verbs and objects
- Think hard before adding another verb
add

- Creates new entries in the cluster database
- Examples:
  - Hosts
  - Appliances
  - Rolls
rocks add distribution

```python
import rocks.commands

class Command(rocks.commands.createCommand, rocks.commands.dbCommand):
    
    def run(self, params, args):
        
        if len(args) != 1:
            self.abort('must supply one distribution')

dist = args[0]

        if dist in self.getDistributionNames():
            self.abort('distribution "%s" exists' % dist)

        self.db.execute("""insert into distributions (name) values""
                        ('%s')""" % dist)
```

© 2008 UC Regents
dump

- Returns cluster database information in the form of rocks command lines
- Examples:
  - Hosts
  - Network
- Same as –dump flag on insert-ethers
rocks dump host

# rocks dump host
/opt/rocks/bin/rocks add host vizagra cpus=1 rack=0 rank=0 membership="Frontend"
/opt/rocks/bin/rocks add host tile-0-1 cpus=2 rack=0 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-0 cpus=2 rack=0 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-2 cpus=2 rack=0 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-3 cpus=2 rack=0 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-3 cpus=2 rack=1 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-2 cpus=2 rack=1 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-1 cpus=2 rack=1 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-0 cpus=2 rack=1 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-0 cpus=2 rack=2 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-1 cpus=2 rack=2 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-2 cpus=2 rack=2 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-3 cpus=2 rack=2 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-0 cpus=2 rack=3 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-1 cpus=2 rack=3 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-2 cpus=2 rack=3 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-3 cpus=2 rack=3 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-0 cpus=2 rack=4 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-1 cpus=2 rack=4 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-2 cpus=2 rack=4 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-3 cpus=2 rack=4 rank=3 membership="Tile"
rocks dump host

```python
import os
import sys
import string
import rocks.commands

class Command(rocks.commands.HostArgumentProcessor):
    def run(self, params, args):
        self.db.execute("select n.cpus, n.rank, m.name from nodes n, memberships m where n.membership=m.id and n.name='%s' % host")
        (cpus, rank, membership) = self.db.fetchone()
        self.dump("add host %s cpus=%s rank=%s rank=%s.%s %s")
        "host, cpus, rank, membership")
```
list

- Reports information in human readable format
- No side-effects on the database
- Examples:
  - Hosts
  - Appliances
  - Rolls
rocks list host

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>vizagra:</td>
<td>Frontend</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-1:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-2:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-3:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-3:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-2:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-1:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-0:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-0:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-1:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-2:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-3:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-3-0:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-3-1:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-3-2:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-3-3:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-4-0:</td>
<td>Tile</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-4-1:</td>
<td>Tile</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-4-2:</td>
<td>Tile</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-4-3:</td>
<td>Tile</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>--------</td>
</tr>
</tbody>
</table>
rocks list host

```python
import rocks.commands

class Command(rocks.commands.HostArgumentProcessor,
               rocks.commands.list.command):
    pass

class Command(Command):
    def run(self, params, args):
        self.beginOutput()

        for host in self.getHostnames(args):
            self.db.execute("select m.name, n.cpus, n.rank, n.comment from
                            nodes n, memberships m where
                            n.membership=m.id and n.name='%(host)'",
                            host=host)
            self.addOutput(host, self.db.fetchall())

        self.endOutput(header=['host', 'membership',
                               'cpus', 'rank', 'rank', 'comment'])
```

© 2008 UC Regents
set

- Modifies entries in the cluster database
- Examples:
  - Network Interfaces
  - Appliance Assignment
  - Rack / Rank
- add-extra-nic
  - Rocks add host interface
  - Rocks set host interface
add-extra-nic is now …

◆ Start
# rocks list host interface compute-1-1
SUBNET IFACE MAC IP NETMASK GATEWAY MODULE NAME
private eth0 00:0e:0c:5d:7e:5e 10.255.255.251 255.0.0.0 ------- e1000 compute-1-1
-------- eth1 00:30:1b:b2:ea:61 ------------------ --------- ------- tg3

◆ Configure
# rocks set host interface ip compute-1-1 eth1 192.168.1.1
# rocks set host interface gateway compute-1-1 eth1 192.168.1.254
# rocks set host interface name compute-1-1 eth1 fast-1-1
# rocks set host interface subnet compute-1-1 eth1 public

◆ Verify
# rocks list host interface compute-1-1
SUBNET IFACE MAC IP NETMASK GATEWAY MODULE NAME
private eth0 00:0e:0c:5d:7e:5e 10.255.255.251 255.0.0.0 ------- e1000 compute-1-1
public eth1 00:30:1b:b2:ea:61 192.168.1.1 255.255.255.0 192.168.1.254 tg3 fast-1-1
rocks set host interface ip

def run(self, params, args):
  (args, iface, ip) = self.fillPositionalArgs(('iface', 'ip'))
  hosts = self.getHostnames(args)
  if len(hosts) != 1:
    self.abort('must supply one host')
  if not iface:
    self.abort('must supply iface')
  if not ip:
    self.abort('must supply ip')
  ip = ip.upper() if not ip else ip
  for host in hosts:
    self.db.execute("""update networks, nodes set
                    networks.ip=NULLIF("%s", NULL) where
                    nodes.ip=%s and nodes.id=%s
                    (networks.device="%s" or networks.mac="%s")"" %
                    (ip, host, iface, iface))
Start and stop something
NULL commands
Reserve the verbs for use on other Rolls
Think “abstract base class”
sync

- Synchronizes the database state to software configuration files
- Similar to the old “insert-ethers – update”
- Not complete yet, see Rocks VI (5.1)
rocks sync config

```python
import os
import sys
import string
import rocks.file
import rocks.commands

class Command(rocks.commands.sync.command):
    
    def run(self, params, args):
        cmd = '/opt/rocks/sbin/insert-ethers --update'
        for line in os.popen(cmd).readlines():
            self.owner.addText(line)
```

For each system configuration file controlled by Rocks, first rebuild the configuration file by extracting data from the database, then restart the relevant services.

<example cmd='sync config'>
Rebuild all configuration files and restart relevant services.
</example>

© 2008 UC Regents
Extensibility

◆ New commands
  ✐ Add directories
  ✐ Add __init__.py code

◆ Existing commands
  ✐ Some commands can be extended
  ✐ Plugins
rocks sync users

- Run after useradd
  - Populate auto.home
  - Cleanup password file
  - Send 411 files
- Two plugins
  - Fixnewusers
  - 411
- Partial Ordering
- Other Rolls can add more plugins to this command
- Command must be design for plugins (not default)
import rocks.commands

class Command(rocks.commands.sync.command):
    ""
    Update all user-related files (e.g., /etc/passwd, /etc/shadow, etc.)
on all known hosts. Also, restart autofs on all known hosts.
    ""
    <example cmd='sync users'>
    Send all user info to all known hosts.
    </example>
    ""

    def run(self, params, args):
        self.runPlugins()
411 plugin

```python
import os
import rocks.commands

class Plugin(rocks.commands.Plugin):
    """Force a 411 update and re-load autofs on all nodes""

    def provides(self):
        return '411'

    def requires(self):
        return ['fixnewusers']

    def run(self, args):
        
        # force the rebuild of all files under 411's control
        
        for line in os.popen('make -C /var/411 force').readlines():
            self.owner.addText(line)

        
        # restart autofs on all known hosts
        
        cmd = '/opt/rocks/bin/tentakel "service autofs reload"'
        for line in os.popen(cmd).readlines():
            self.owner.addText(line)
```

© 2008 UC Regents
auto.home / passwd plugin

```python
import os
import string
import rocks.commands

class Plugin(rocks.commands.Plugin):
    
    def provides(self):
        return 'fixnewusers'

    def run(self, args):
        # scan the password file for any '/export/home' entries
        new_users = []
        default_dir = '/export/home'

        file = open('/etc/passwd', 'r')

        for line in file.readlines():
            l = string.split(line[1:-1], ':')

            if len(l) < 6:
                continue

            username = l[0]
            homedir = l[5]

            if homedir[len(default_dir)] == default_dir:
                new_users.append(username)

        file.close()

        hostname = '%%s,%%s' %
        (self.db.getGlobalVar("Kickstart", 'PrivateHostname'),
         self.db.getGlobalVar("Kickstart", 'PrivateDNSDomain'))

        for user in new_users:
            # for each new user, change their default directory to
            # /home/username
            cmd = '/usr/sbin/usermod -d %s %s %s
              (os.path.join('/home', user), user)
            for line in os.popen(cmd).readlines():
                self.owner.addText(line)

        # then update the auto.home file
```

© 2008 UC Regents
Argument Processing

- rocks <verb> <object…> <subject> <params…>
- Subject is typed by first object
  - host -> one or more hostname
  - roll -> one or more roll names
- Params are in key=value form
- Same as –flag=value but easier to read
Helper classes and functions

◆ ArgumentProcessors
  ➜ Class to parse the subject in a standard way
  ➜ Exists for hosts, rolls, appliances, …

◆ Parameters Parsing
  ➜ fillPositionalArgs
  ➜ fillParams
HostArgumentProcessor

◆ Command must derive from 
  `rocks.commands.HostArgumentProcessor`

◆ `self.getHostnames(args)`
  
  ✐ Return a list of hostname as they appear in the cluster database
  ✐ If `args = None` all the host in the cluster are returned
  ✐ `args` can also be a group
    • Rack0, rack1
  ✐ Or an appliance type
    • Compute, Tile, …
import rocks.commands

class command(rocks.commands.HostArgumentProcessor,
               rocks.commands.list.command):
    pass

class Command(command):
    ""
    List the membership, CPU count, physical position info and comment for
    a list of hosts.
    ""
    <arg optional='1' type='string' name='host' repeat='1'>
    Zero, one or more host names. If no host names are supplied, info about
    all the known hosts is listed.
    </arg>

    <example cmd='list host compute-0-0'>
    List info for compute-0-0.
    </example>

    <example cmd='list host'>
    List info for all known hosts.
    </example>
    """

    def run(self, params, args):
        self.beginOutput()
        for host in self.getHostnames(args):
            self.db.execute("select m.name, n.cpus,
                             n.rack, n.rank, n.comment from
                             nodes n, memberships m where
                             n.memberid=m.id and n.name='%s'" % host)
            self.addOutput(host, self.db.fetchone())
        self.endOutput(header=['host', 'membership',
                                'cpus', 'rack', 'rank', 'comment'])
args = None

# rocks list host

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>vizagra:</td>
<td>Frontend</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-1:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-0:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-2:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-3:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-3:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-2:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-1:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-0:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-0:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-1:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-2:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-3:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>--------</td>
</tr>
</tbody>
</table>
args = list of hosts

```
# rocks list host tile-0-0 10.255.255.253 tile-3-0.local

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-0-0:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-------</td>
</tr>
<tr>
<td>tile-0-1:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>-------</td>
</tr>
<tr>
<td>tile-3-0:</td>
<td>Tile</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>-------</td>
</tr>
</tbody>
</table>
```
args = rack

# rocks list host rack2

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-2-0: Tile</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td></td>
<td>-------</td>
</tr>
<tr>
<td>tile-2-1: Tile</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td>-------</td>
</tr>
<tr>
<td>tile-2-2: Tile</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>-------</td>
</tr>
<tr>
<td>tile-2-3: Tile</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td>-------</td>
</tr>
</tbody>
</table>
args = appliance type

# rocks list host tile

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-0-0:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-1:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-2:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-0-3:</td>
<td>Tile</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-0:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-1:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-2:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-3:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-0:</td>
<td>Tile</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-1:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-2:</td>
<td>Tile</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>--------</td>
</tr>
</tbody>
</table>
Any combination is fine

# rocks list host tile-2-0 rack1 frontend

<table>
<thead>
<tr>
<th>HOST</th>
<th>MEMBERSHIP</th>
<th>CPUS</th>
<th>RACK</th>
<th>RANK</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile-1-0:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-1:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-2:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>--------</td>
</tr>
<tr>
<td>tile-1-3:</td>
<td>Tile</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>--------</td>
</tr>
<tr>
<td>tile-2-0:</td>
<td>Tile</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>--------</td>
</tr>
<tr>
<td>vizagra:</td>
<td>Frontend</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>--------</td>
</tr>
</tbody>
</table>

© 2008 UC Regents
# ArgumentProcessors

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Helper Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplianceArgumentProcessor</td>
<td>getApplianceNames</td>
</tr>
<tr>
<td>DistributionArgumentProcessor</td>
<td>getDistributionNames</td>
</tr>
<tr>
<td>HostArgumentProcessors</td>
<td>getHostnames</td>
</tr>
<tr>
<td>MembershipArgumentProcessor</td>
<td>getMembershipNames</td>
</tr>
<tr>
<td>NetworkArgumentProcessor</td>
<td>getNetworkNames</td>
</tr>
<tr>
<td>RollArgumentProcessor</td>
<td>getRollNames</td>
</tr>
</tbody>
</table>
RollArgumentProcessor

```python
import os
import stat
import time
import sys
import string
import rocks.commands

class Command(rocks.commands.list.command):
    """
    List the status of available rolls.
    """
    def run(self, params, args):
        self.beginOutput()
        for (roll, version) in self.getRollNames(args, params):
            self.db.execute("""select version, arch, enabled from rolls where name='%s' and version='%s'""" % (roll, version))
            row = self.db.fetchone()
            self.addOutput(roll, row)
        self.endOutput([], ['name', 'version', 'arch', 'enabled'], trimOwner=0)
```

© 2008 UC Regents
No Parameter

```bash
# rocks list roll

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>ARCH</th>
<th>ENABLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>viz:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>sge:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>kernel:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>updates:</td>
<td>5.1</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>java:</td>
<td>4.3.2</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>xen:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>CentOS:</td>
<td>5.1</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>ganglia:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>web-server:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>base:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
<tr>
<td>hpc:</td>
<td>5.0</td>
<td>i386</td>
<td>yes</td>
</tr>
</tbody>
</table>
```

© 2008 UC Regents
Version Parameter

# rocks list roll version=4.3.2

NAME   VERSION ARCH ENABLED
java:  4.3.2   i386 yes
Summary

- ArgumentProcessors standardize the handling of command line subjects
- Calling the helper function with an empty list returns all subject in the database
- HostArgumentProcessor knows about more than just host names
- RollArgumentProcessor can filter on versions
fillParams

- Create local variables based on command parameters (key=value)
- Argument a list of (key, default) tuples
- If the parameter is not found on the command line the default value is used
def maketorrent(self, filename, data):
    info = {}
    info['length'] = os.stat(filename)[stat.ST_SIZE]
    info['name'] = os.path.basename(filename)
    data['info'] = info
    encoded = BitTorrent.torrent.bencode(data)
    file = open('%s.torrent' % (filename), 'w')
    file.write(encoded)
    file.close()

def run(self, params, args):
    if len(args) != 1:
        self.abort('must supply one file')
    filename = args[0]

    (timestamp, ) = self.fillParams([('timestamp', time.time())])
    try:
        creation_date = int(timestamp)
    except:
        creation_date = int(time.time())

    data = {
        # announc string
        #
        'announce': 'http://%s:7625/announce' % (localhost)
    }

    data['creation date'] = creation_date

    #
rocks: add host

```python
73:    basename, rack, rank = host.split('-')
74:    self.db.execute("""select m.name from
75:        appliances a, memberships m where
76:        a.name='%s' and m.appliance=a.id""" % basename)
77:        membership, = self.db.fetchone()
78:        rack = int(rack)
79:        rank = int(rank)
80:    except:
81:        membership = None
82:        rack = None
83:        rank = None
84:
85:        # fillParams with the above default values
86:        (membership, numCPUs, rack, rank) = self.fillParams(
87:            [('membership', membership),
88:             ('cpus', 1),
89:             ('rack', rack),
90:             ('rank', rank)])
91:
92:        if not membership:
93:            self.abort('membership not specified')
94:        if rack == None:
95:            self.abort('rack not specified')
96:        if rank == None:
97:            self.abort('rank not specified')
98:
99:    self.db.execute("""insert into nodes
100:        (site, name, membership, cpus, rack, rank)
101:        values
102:            (0,
103:             '%s',
104:             (select id from memberships where name='%s'),
105:             '%d',
106:             '%d',
107:             '%d')""" %
108:            (host, membership, int(numCPUs), int(rack), int(rank)))
109:
110:
111:
```
fillPositionalArgs

- Allows for parameters to have implied keys (just values on command line)
- This is an optimization for ease of use, not ease of software
- Argument is a list of keys
  - No default value processing, if a key is specified it is required
  - Use this only when a parameter is required
- Example:
  - `# rocks set network netmask optiputer netmask=255.255.255.0`
  - `# rocks set network netmask optiputer 255.255.0.0`
rocks set network netmask

def run(self, params, args):
    (args, netmask) = self.fillPositionalArgs(('netmask',))

    if not len(args):
        self.abort('must supply network')
    if not netmask:
        self.abort('must supply netmask')

    for network in self.getNetworkNames(args):
        self.db.execute("update subnets set netmask='\%s' where subnets.name='\%s'", (netmask, network))
rocks set host interface

def run(self, params, args):
    
    (args, iface, mac) = self.fillPositionalArgs(('iface', 'mac'))
    
    hosts = self.getHostnames(args)
    
    if len(hosts) != 1:
        self.abort('must supply one host')
    if not iface:
        self.abort('must supply iface')
    if not mac:
        self.abort('must supply mac')
    
    for host in hosts:
        self.db.execute("update networks, nodes set 
        networks.mac=NULLIF('%s','NULL') where 
        nodes.name='%s' and networks.node=nodes.id and 
        (networks.device='%s' or networks.mac='%s')
        (mac, host, iface, iface))" %
Help and Docstrings

◆ The command line is the documentation
  ➔ No more out of date man pages
  ➔ Still needs a cookbook document, but reference is part of the code
◆ We’ve been looking at this all session
◆ Class docstring """"text"""
◆ Command line has an XML format
# rocks list roll help
rocks list roll [roll]...

Description:

List the status of available rolls.

Arguments:

[roll]

List of rolls. This should be the roll base name (e.g., base, hpc, kernel). If no rolls are listed, then status for all the rolls are listed.

Examples:

$ rocks list roll kernel
List the status of the kernel roll

$ rocks list roll
List the status of all the available rolls
```python
import os
import stat
import time
import sys
import string
import rocks.commands

class Command(rocks.commands.RollArgumentProcessor):
    rocks.commands.list.command):
        
        List the status of available rolls.
        
        <arg optional='1' type='string' name='roll' repeat='1'>
        List of rolls. This should be the roll base name (e.g., base, hpc, kernel). If no rolls are listed, then status for all the rolls are listed.
        </arg>

        <example cmd='list roll kernel'>
        List the status of the kernel roll
        </example>

        <example cmd='list roll'>
        List the status of all the available rolls
        </example>

    
    def run(self, params, args):
        self.beginOutput()
        for (roll, version) in self.getRollNames(args, params):
            self.db.execute("\"\"\"select version, arch, enabled from rolls where name='\%s' and version='\%s'\"\"\" %
                (roll, version))
            for row in self.db.fetchall():
                self.addOutput(roll, row)
        self.endOutput(header=['name', 'version', 'arch', 'enabled'],
                           trimOwner=True)
```
<arg>

- Attributes
  - name (required)
  - optional (default = “0”)
  - type (default = “string”)
  - repeat (default = “0”)

- Example:
  ```xml
  <arg type='string' name='network' repeat='1'>
    One or more named networks that should have the defined netmask.
  </arg>
  ```
<param>

Attributes

- name (required)
- optional (default = “1”)
- type (default = “string”)
- repeat (default = “0”)

Example:

<param type='string' name='iface'>
  Can be used in place of the iface argument.
</param>
<example>

_attributes

(cmd(required))

_example:
<example cmd='set host interface mac compute-0-0
eth1 00:0e:0c:a7:5d:ff'>
Sets the MAC Address for the eth1 device on host compute-0-0.
</example>
<related>

Example

<related>set host interface iface</related>
<related>set host interface ip</related>
<related>set host interface gateway</related>
<related>set host interface module</related>
Help

- `rocks <verb> <object…> <subject> help`
  - Loads the command module
  - Parses the XML docstring
  - Format and output help as 80 column text

- Debug syntax with `format= parameter`
help format=raw

# rocks list roll help format=raw
1:
2: List the status of available rolls.
3:
4: <arg optional='1' type='string' name='roll' repeat='1'>
5: List of rolls. This should be the roll base name (e.g., base, hpc, kernel). If no rolls are listed, then status for all the rolls are listed.
6: </arg>
7: </example>
8: </example>
9:
10: <example cmd='list roll kernel'>
11: List the status of the kernel roll
12: </example>
13:
14: <example cmd='list roll'>
15: List the status of all the available rolls
16: </example>

© 2008 UC Regents
# rocks list roll help format=parsed

{'related': [], 'example': [(u'list roll kernel', u'
List the status of the kernel roll
'), (u'list roll', u'
List the status of all the available rolls')], 'description': u'
List the status of available rolls.

param: [], 'arg': [(u'roll', u'string', 1, 1), u'List of rolls. This should be the roll base name (e.g., base, hpc, kernel). If no rolls are listed, then status for all the rolls are listed.']}

© 2008 UC Regents
Docbook

◆ Roll Usersguide Command Reference is generated automatically

```
# rocks list roll help format=docbook
<section id="rocks-list-roll" xreflabel="list roll">
<title>list roll</title>
<cmdsynopsis>
  <command>rocks list roll</command>
  <arg rep="repeat" choice="opt">roll</arg>
</cmdsynopsis>
<para>
  List the status of available rolls.
</para>
```

© 2008 UC Regents
Viz Roll
Dozen+ Command

# rocks list roll command viz

COMMAND
create viz layout
disable chromium
disable hidebezels
enable chromium
enable hidebezels
list dmx layout
list host xconfig
list viz layout
start chromium
start dmx
start xlogo
stop xlogo
sync viz
LCD Bezels
rocks enable hidebezels

- Draws pixels behind the bezels (mullions) of the LCD monitors
- Calculated offset for TwinView and normal modes
- Reset the X11 configuration on all nodes
- Great mode for moving graphics
import rocks.commands.enable
import os

class Command(rocks.commands.enable.command):
    """
    Enable Bezel Hiding mode.
    """
    <example cmd="enable hidebezels">
    </example>
    """

    MustBeRoot = 0

def run(self, params, args):

    os.system('touch ~/.hidebezels')

    # If the database videowall layout has two (or more) entries
    # for the same host and card we know we are in twinview
    # mode. In this case we need to reconfigure and restart
    # X11 for this host.

    self.db.execute("""select n.name, v.cardid
    from nodes n, videowall v where v.node=n.id""")
    dict = {}
    for key in self.db.fetchall():
        if dict.has_key(key):
            dict[key] = 1    # TwinView host
        else:
            dict[key] = 0    # NonTwinView host (so far)

    for (host, card) in dict.keys():
        if dict[(host, card)]:
            os.system('ssh -f ')
            os.system('"%s /usr/X11R6/bin/xrandr -d :0 -s 1' % host)
rocks disable hidebezel

- All pixels are drawn and the bezels break apart the image
- Removes any offset from previous mode
- Resets the X11 configuration on all nodes
- Great for static images and text
import rocks.commands.enable
import os

class Command(rocks.commands.disable.command):
    
    Disable Bezel Hiding mode.

    <example cmd="disable hidebezels">
    </example>
    
    MustBeRoot = 0

def run(self, params, args):

    os.system('/bin/rm ~/.hidebezels')

    # If the database videowall layout has two (or more) entries
    # for the same host and card we know we are in twinview
    # mode. In this case we need to reconfigure and restart
    # X11 for this host.

    self.db.execute("""select n.name, v.cardid
    from nodes n, videowall v where v.node=n.id"""")
    dict = {}
    for key in self.db.fetchall():
        if dict.has_key(key):
            dict[key] = 1  # TwinView host
        else:
            dict[key] = 0  # NonTwinView host (so far)

    for (host, card) in dict.keys():
        if dict[(host, card)]:
            os.system('ssh -f 
            %s /usr/X11R6/bin/xrandr -d :0 -s 0
            % host)
More Commands …

- Starting and stopping
  - Chromium
  - DMX
  - Xlogo image

- Re-writing X11 config files

- What do you want to see?

- HINT: This is interactive right now
Roll Screen Development

Debugging assistance for building Rocks Rolls with screens

OSGC, May 2008
Nadya Williams nadya@oci.uzh.ch
University of Zurich
Very Brief Rolls Overview

- Rolls provide for a cluster customization
- Rolls reliably install and configure a software on a cluster frontend
  - Extend/modify stock OS
  - Add third party packages
- Interaction during install only via screens
- Fully tested before release
Building Roll’s Screen
Screen’s Functions

- Collect user input before roll installation
- Verify user input and forward it to the rocks installer
- Depends on: what a roll developer puts in!
Screen’s Pros and Cons

- Laconic form
- Provides help
- Verify input correctness
- Easy and fast to use

- Limited space
- Don’t assume user can type
- Can’t foresee all site details
- Screens are available only during cluster install
Prepare for the roll build

- Check out CVS Rocks distribution
  
  # cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT login
  
  # cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT checkout
  
  -r ROCKS_$VERSION

- Change to the top-level rolls’ directory:
  
  # cd rocks/src/roll

- Create new roll directory
  
  # bin/make-roll-dir.py -n YourRollName -v YourRollVersion
  
  # ls YourRollName
  
  graphs/ Makefile nodes/ src/ version.mk

- Create your roll
Add your Screen

- Prerequisites from rocks cvs repository
  - roll/bin/
  - roll/base/src/screens
  - Your basic roll structure is ready
    - Skeleton xml files, makefiles …
  1. Add screen xml file to nodes/
  2. Add javascript
  3. Add screen to the graph
screen-myroll.xml file

```xml
<kickstart>
  <screen>
    <title>Your Roll Title</title>
    <code>
      <include file="javascript/yourRoll.js">
    </code>
    <variable>
      <label>My field label</label>
      <name>Info_NameForAppsGlobals</name>
      <type>string</type>
      <size>20</size>
      <default>ReasonableDefaultValue</default>
      <value><var name="Info_NameForAppsGlobals"/></value>
      <help>Help text for the variable field.</help>
      <validate>check_function1</validate>
    </variable>
    ...
  </screen>
</kickstart>
```
Screen XML Language

Screen contains

➥ Title
  • Your title description

➥ code
  • Specify javascript file

➥ variable
  • Simple syntax. Supports multiple variables.
  • Defines info about variable “appearance” on the screen
  • Defines info that goes to app_globals table
Screen <variable> attributes

- **name**
  - Format: `Service_Component`
  - `Service` and `Component` are columns names in the `app_globals` database table.
  - Use “Info” for Service:
    `<name>Info_GfarmMetaServer</name>`

- **value**
  - Format same as “name”
    `<value>`
    `<var name="Info_GfarmMetaServer"/>`
    `</value>`

- **default**
  - Sets the value of this variable
    `<default>pine.forrest.edu</default>`

- **label**
  - Sets the form label

- **type**
  - String, menu, ipv4-address

- **size**
  - Sets the size of the screen field

- **help**
  - Sets the explanation for the variable

- **validate**
  - Sets the javascript validation function
    - Use the same name as in myroll.js javascript file
    `<validate>check_GFmetaserver</validate>`
Add screen to the graph

In graphs/default/myroll.xml:

- **add ordering**

```xml
<order head="screen-timezone">
  <tail>screen-myroll</tail>
</order>
<order head="screen-myroll">
  <tail>screen-partitioning</tail>
</order>
```

- **add edges**

```xml
<edge from="server">
  <to>screen-myroll</to>
  <to>myroll-server</to>
  <to>myroll-client</to>
</edge>
```
File myroll.js

- Contains screen variable verification
- One function per variable
- Can use existing code from other rolls
  - Copy desired file from otherroll/include/javascript/ to myroll/include/javascript/
- For examples see rolls:
  - base
  - gama
Debugging Roll’s Screen
**Screen html files**

- **Create screen html files:**
  
  ```
  # export PATH=$PATH:`pwd`/rocks/src/roll/bin
  # cd myroll/
  # make-screen-val.py -x screen-myroll.xml myroll
  ```

  **Make-screen-val.py actions:**
  - Uses your javascript
  - Creates myroll/screenval/ and all files under it

- **View html files:**
  
  ```
  # cd screenval/
  # firefox file:///<path>/myroll/screenval/rocks.html
  ```

  Note: will not work with Safari

May 14, 2008  
OSGC
View your Screen

Welcome to Rocks

Help

Gfarm Metaserver Host:
Gfarm Metaserver for your cluster. Specify FQDN of this host or FQDN for another host.

Gfarm FS Node:
Specify FQDN of your cluster if this host is a Gfarm FS node. Default is none.

Gfarm Configuration

Gfarm Metaserver Host: pine.hpcc.jp
Gfarm FS Node: none

Back  Validate
Try your Screen

Welcome to Rocks

Help
Gfarm Metaserver Host:
Gfarm Metaserver for your cluster. Specify FQDN of this host or FQDN for another host.

Gfarm FS Node:
Specify FQDN of your cluster if this host is a Gfarm FS node. Default is none.

Gfarm Configuration

Gfarm Metaserver Host: pine.hpcc.jp
Gfarm FS Node: no

Back Validate

"no" is not a valid IP address
What can you test

- Test input fields one by one
  - alter the default value and press “validate”
- Test your javascript
  - If don’t see expected behavior, check your javascript
- Test, test, test…
  ```
  while ( errors ) {
    recreate screenval/ with make-screen-val.py
    reopen rocks.html
    fix another error in your javascript
  }
  ```
  Lets see this in action
When Things Go Wrong

- Problem with javascript syntax
- One of javascript files is absent
- Names mismatch
- Function is not returning value
Thank you!

Questions ?