What's In Rocks 5.4

Channel Bonding

Channel Bonding

I am trying to channel bond 4 gigabit NICs on the compute nodes to
(theoretically) improve performance, but am having some difficulty doing it correctly.
I can easily add the configuration by hand, but then if a nodere-installs, it would be lost.
I have experimented withthe rocks add hostinterface command, but cannot seem to
figure out how to tell it to enslave the other ethernets. What is the correct way to do this?

Hello all:
Has anyone done channel bonding before on Rocks? Detailed instruction would be helpful.

It was mentioned the other day that ROCKS does not support the bonding network cards (NICs).
I was a little confused by this as CentOS certainly does as a subset of RedHat Linux (or so it appears).
Would someone be kinda enough to fill me in on what aspect of ROCKS does not support bonding (BOND0, etc)?

So I decided to start the next step which is to channel bond each nodes 2 ethernet cards. I got up to compute node 3. And decided to check things out with the previous nodes. For some reason, rsh is now refusing connections from the headnode to the channel bonded compute nodes. It was working on eth0 perfectly. What could cause this to happen?

- I mean, how hard can it be?
  - With the Rocks Command Line, I figured we could do it with 3 existing commands

Channel Bonding

- Created a new command to configure channel bonding

```
# rocks list host interface compute-0-1
SUBNET  IFACE MAC               IP           NETMASK     MODULE NAME        VLAN OPTIONS
private eth0  00:1e:4f:b0:74:ef 10.1.255.253 255.255.0.0 tg3    compute-0-1 ---- -------
------- eth1  00:10:18:31:74:43 ------------ ----------- tg3    ----------- ---- -------
```

- Run the command:

```
# rocks add host bonded compute-0-1 channel=bond0 interfaces=eth0,eth1 \
  ip=10.1.255.253 network=private
```

Channel Bonding

- The result

```
# rocks list host interface compute-0-1
SUBNET  IFACE MAC               IP           NETMASK     MODULE  NAME        VLAN OPTIONS CHANNEL
private bond0 ----------------- 10.1.255.253 255.255.0.0 bonding compute-0-1 ---- ------- -------
------- eth0  00:1e:4f:b0:74:ef ------------ ----------- tg3     ----------- ---- ------- bond0
------- eth1  00:10:18:31:74:43 ------------ ----------- tg3     ----------- ---- ------- bond0
```

- Can apply the change on the fly:

```
# rocks sync config
# rocks sync host network compute-0-1
```

Firewall Configuration via the Rocks Command Line

- But, for channel bonding to work, we needed to make the firewall adapt to the configured interfaces

```
-A INPUT -i eth0 -j ACCEPT
-A INPUT -p tcp --dport 0:1024 -j REJECT
-A INPUT -p udp --dport 0:1024 -j REJECT
```

```
-A INPUT -i bond0 -j ACCEPT
-A INPUT -p tcp --dport 0:1024 -j REJECT
-A INPUT -p udp --dport 0:1024 -j REJECT
```

Firewall Configuration via the Rocks Command Line

- Added a boat load of commands

```
add appliance firewall {appliance} [action=string] [chain=string]
add firewall [action=string] [chain=string] [network=string]
add host firewall {host} [action=string] [chain=string] [network=string]
add os firewall {os} [action=string] [chain=string] [network=string]
close appliance firewall {appliance} [network=string] [protocol=string]
close firewall [network=string] [protocol=string] [service=string]
close host firewall {host} [network=string] [protocol=string] [service=string]
close os firewall {os} [network=string] [protocol=string] [service=string]
dump appliance firewall
dump firewall
dump host firewall
dump os firewall
list appliance firewall [appliance]...
list firewall {None}
list host firewall [host]...
list os firewall [os]...
open appliance firewall {appliance} [network=string] [param=string]
open firewall [network=string] [protocol=string] [service=string]
open host firewall {host} [network=string] [protocol=string] [service=string]
open os firewall {os} [network=string] [protocol=string] [service=string]
remove appliance firewall {appliance} [action=string] [chain=string]
remove firewall [action=string] [chain=string] [network=string]
remove host firewall {host} [action=string] [chain=string] [network=string]
remove os firewall {os} [action=string] [chain=string] [network=string]
report host firewall {host}
```

Firewall Configuration via the Rocks Command Line

- The commonly used commands will be:

```
rocks open host firewall {host} [network=string] [protocol=string] [service=string]

rocks close host firewall {host} [network=string] [protocol=string] [service=string]

rocks add host firewall {host} [action=string] [chain=string] [network=string]

rocks list host firewall [host]...
```

- For example, to open up web access on a public interface:

```
# rocks open host firewall compute-0-0 network=public \
  protocol=tcp service=www
```

The Private Network Doesn't Have to be "eth0"

- But to get the new firewall configuration working, we had to break the "eth0 = private network" relationship

```
# rocks list host interface compute-0-0 compute-0-1 compute-0-2
HOST          SUBNET   IFACE MAC                IP            NETMASK     MODULE  NAME
compute-0-0: private bond0 ----------------- 10.1.255.254 255.255.0.0 bonding compute-0-0
compute-0-0: ------- eth0  00:0e:0c:a7:57:d7 ------------ ----------- ------- -----------
compute-0-0: ------- eth1  00:19:b9:21:b8:b6 ------------ ----------- ------- -----------
compute-0-2: ------- eth0  00:10:18:31:74:7e ------------ ----------- ------- -----------
compute-0-2: private eth1  00:1e:4f:b0:72:2f 10.1.255.252 255.255.0.0 ------- compute-0-2
compute-0-2: ------- eth2  00:0e:0c:5d:7e:59 ------------ ----------- ------- -----------
```

The Private Network Doesn't Have to be "eth0"

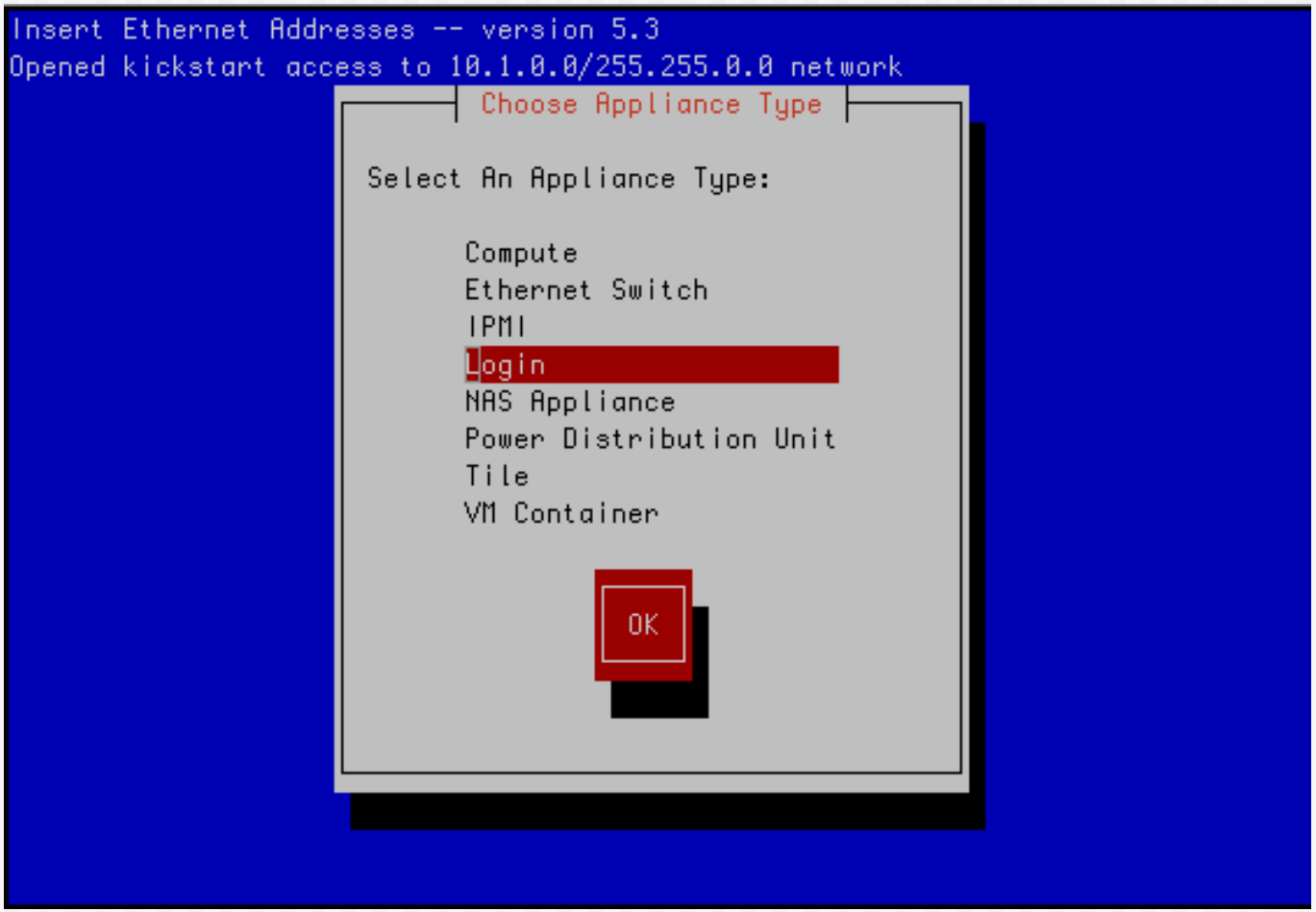
- Now when the "private" network cable moves from one interface to another, the "private" network configuration will follow the cable
  - The interfaces will never be renamed

Login Appliance

Login Appliance

- Administrators have asked for a node in the cluster that is not the frontend where users can login, develop and launch their code
- I mean, how hard can it be?

Login Appliance



Login Appliance

- Created two new attributes:
  - submit_host
    - One can submit jobs to the queuing system from this host
  - exec_host
    - Jobs can be executed on this host

Login Appliance

- Login appliance:
  - submit_host = true, exec_host = false
- Compute node:
  - submit_host = false, exec_host = true
- Can set/unset the attributes for any host
  - Can easily make all tile hosts execution hosts
  - Can easily exclude specific hosts as queuing system resources

Avalanche Installer Retooled

Avalanche Installer Retooled

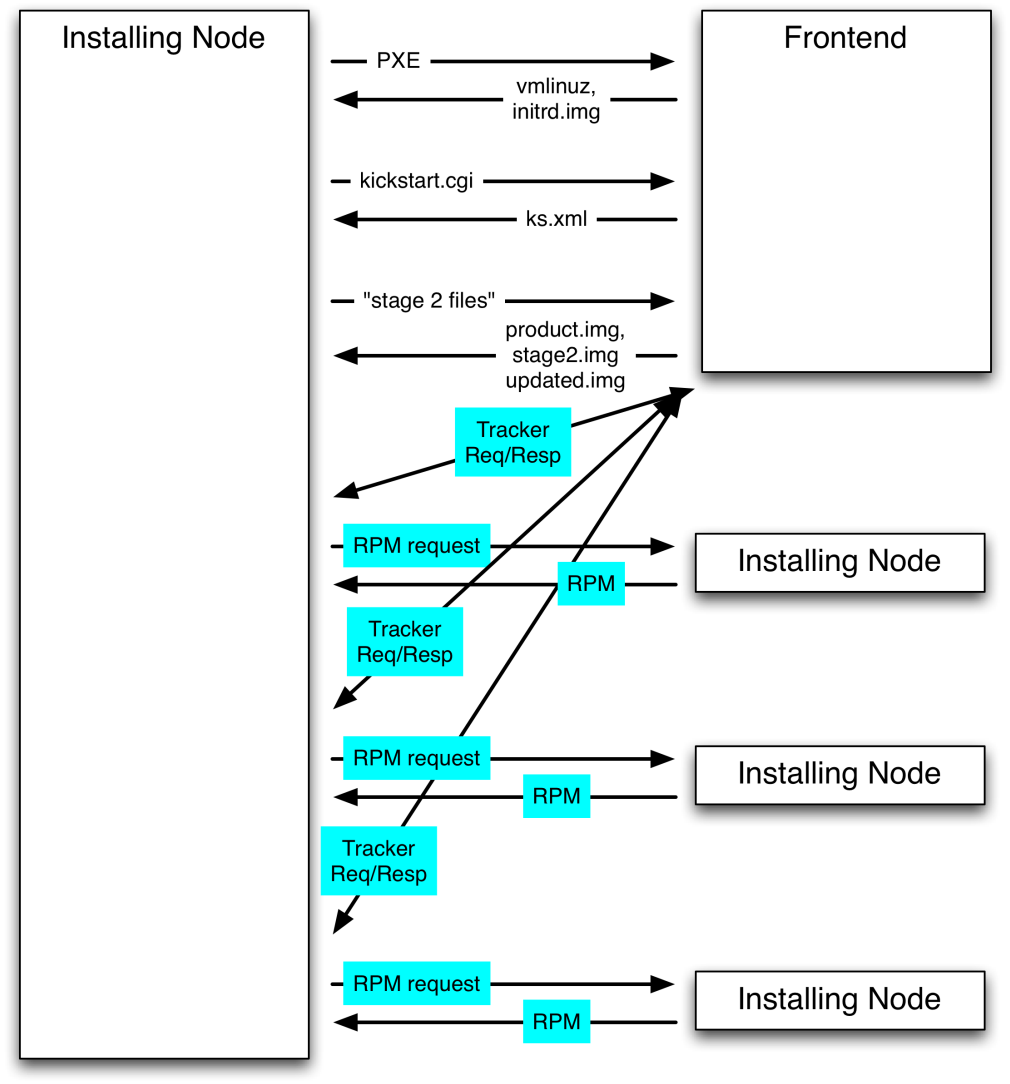
- We went to Nebraska
  - We came back humbled

Avalanche Installer Retooled

- Found several issues that limited scalability
  - Some of the easier fixes made it into Rocks 5.3
- Realized that we needed to get as much traffic off the frontend as possible

Avalanche Installer Retooled

- vmlinuz, initrd.img
  - ~18 MB
- ks.xml
  - ~0.3 MB
- “stage 2 files”
  - ~200 MB



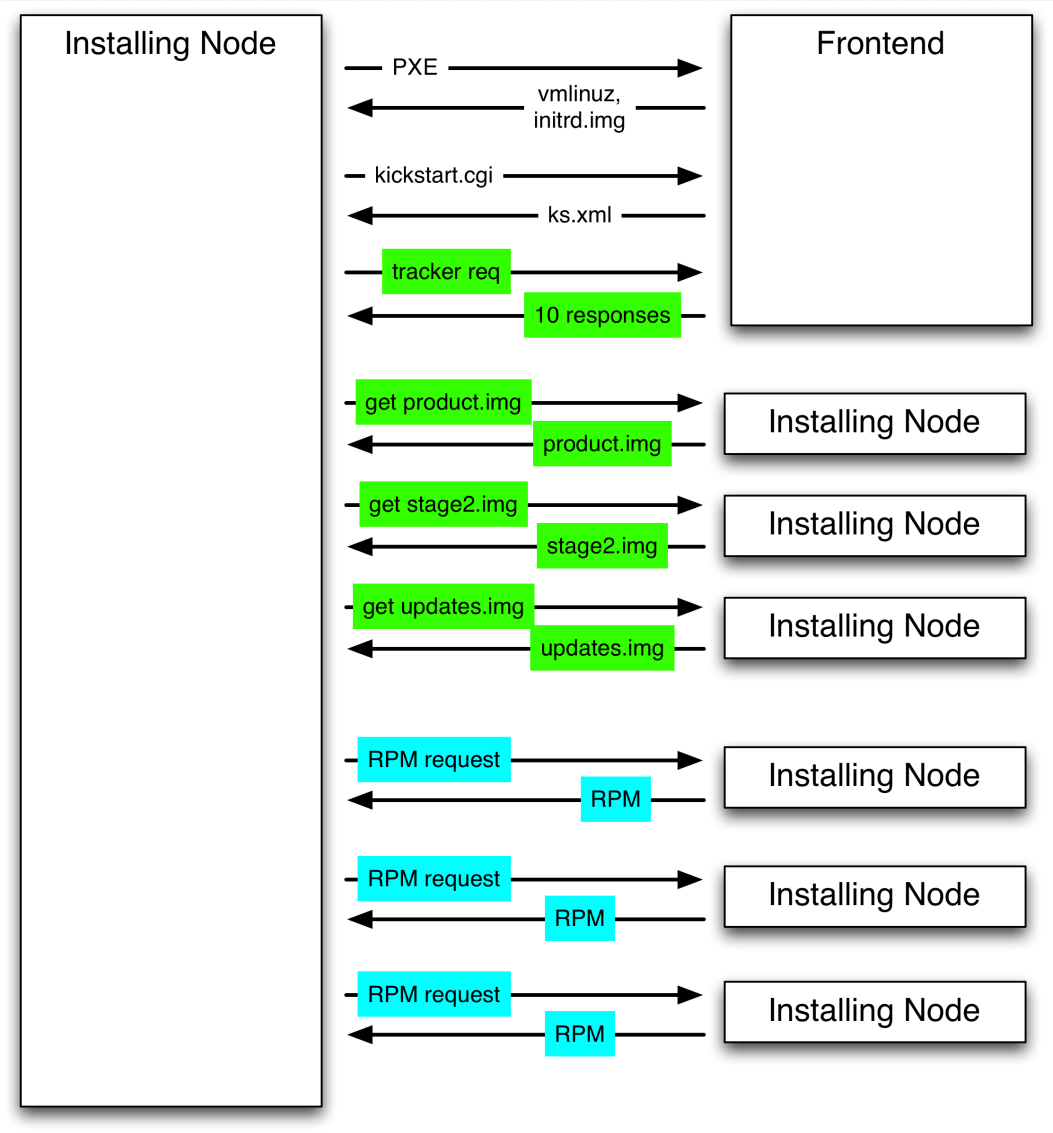
Avalanche Installer Retooled

- The Opportunity: distribute the “stage 2 files” with BitTorrent
- Had to rewrite the client-side of Avalanche in C
  - Which I loved!

Avalanche Installer Retooled

- Added a “package predictor”
  - When a client asks for a package, the tracker returns a list of the next 10 packages that the client will likely ask for
  - Reduces the tracker load on the frontend by 10x
- Only assign 3 clients for each package
  - Previous version sent back **all** the available clients for a package
  - Reduces the tracker response message size for large concurrent reinstallations

Avalanche Installer Retooled



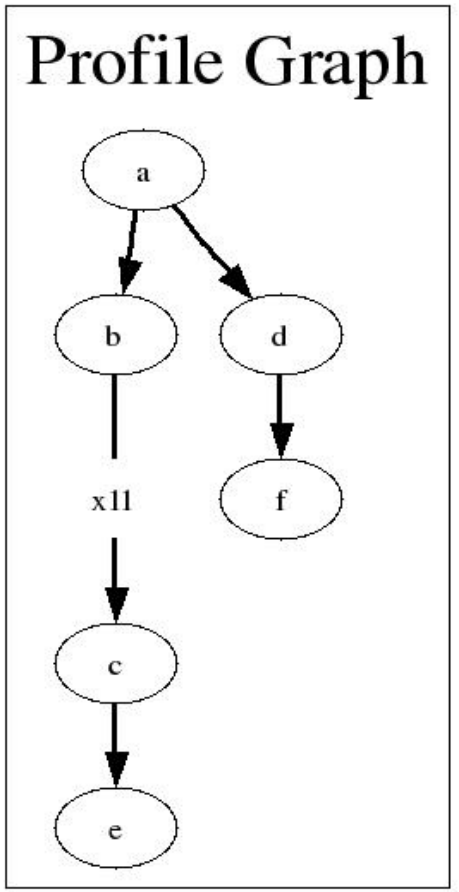
Avalanche Installer Retooled

- Can have multiple “trackers” and “package servers”
  - Previous version: only the frontend tracked and served packages
- Tracker assigns clients based on “least-recently used”
- Group clients by “co-op”
  - A client will try to get a package from members of its “co-op” first
  - Can set the “co-op” with an attribute
    - Default co-op is the rack id

Graph Traversal Fixed

Graph Traversal Fixed

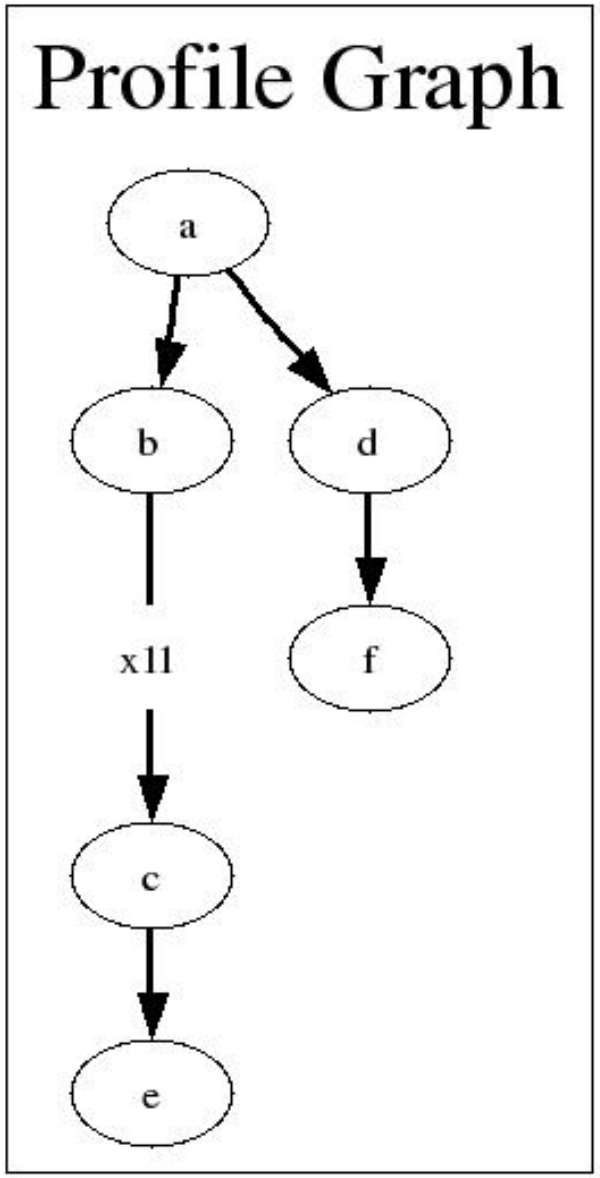
```
<edge from="b" cond="x11">
        <to>c</to>
</edge>

<edge from="c">
        <to>e</to>
</edge>
```



- Original implementation had a major bug
  - If x11 was false, "c" would be omitted, but "e" would be included!

Rocks Graph Fixes

- Fix: prune the tree
  - stop traversing at “b” if “x11” is false



Multiple DNS Zones

Multiple DNS Zone support

- Multiple subnets
- Each subnet maps to a DNS zone
- Serve DNS for multiple interfaces
- Customize zone names
  - Private network need not be "`.local`"
- Examples
  - Optiputer network - `<hostname>.optiputer.net`
  - Local network - `<hostname>.local`

Multiple DNS Zones

```
# rocks add network optiputer 192.168.0.0 255.255.0.0 \
  servedns=true dnszone=myri

# rocks list network
NETWORK    SUBNET         NETMASK         MTU   DNSZONE             SERVEDNS
private:   10.1.0.0       255.255.0.0     1500  local               True
public:    137.110.119.0  255.255.255.0   1500  rocksclusters.org   False
optiputer: 192.168.0.0    255.255.0.0     1500  myri                True

# rocks sync dns
```

Updates

Software Update

- Support for Rocks published updates
  - Patches
  - Security fixed
  - Rocks and/or CentOS packages
- Not the same problem as general software update

rocks update

1. Downloads new packages from ftp.rocksclusters.org for your release of Rocks
2. Runs any update shell scripts we provide
3. Removes any update XML files for Rolls you don't have
4. Creates an Update Roll
5. Create an Update Yum Repository
6. Does a "yum update" on fronted using only this repository

What About Compute Nodes?

- You now have an Update Roll
  - Enable the Roll
  - Rebuild the distribution
- Pick One
  - Re-install nodes
  - Run “yum update” on nodes

“rocks run host” Retooled

“rocks run host” Retooled

- Now 100% tentakel free!

```
rocks run host [host]... {command} [collate=string]
    [command=string] [delay=string]
    [managed=boolean] [stats=string] [timeout=string]
    [x11=boolean]
```

- Collate: prepend the name of the host on each line of the output
- Delay: delay X seconds between command launches
- Managed: only execute on “managed” nodes (e.g., not NAS appliances)
  - A managed host has the attribute “managed” set to “true”
- Stats: print how long it took to run each command
- Timeout: terminate after X seconds
- X11: set to ‘no’ to disable X11 forwarding

Features We're Considering

Features We're Considering

- Console access to virtual compute nodes from within a virtual frontend
  - Related:
    - Allow users to power on/off VMs from within virtual frontend
- Multiple distribution support
- Global/OS/Appliance/Host hierarchy cleanup

Features We're Considering

- Roll "personalities"
  - A method to select several rolls by clicking one checkbox
- Making the SGE job queue data collection more efficient

Features We Should Consider?