



C3 Power Tools: The Next Generation...

Presented by: Stephen L. Scott
scottsl@ornl.gov

C3 Project Team:
Brian Luethke, Thomas Naughton, and Stephen L. Scott

9th EuroPVM/MPI & 4th DAPSYS Conference
Johannes Kepler University – Linz, Austria
September 30, 2002

C3 Overview

- Cluster Command & Control (C3)
 - Started in 1999
 - 2002 is 3rd generation
- Command line / scriptable cluster tools
 - Rapidly deploy software and system images
 - Parallel file scatter and gather
 - Parallel command propagation and execution
- Operates on cluster → multi-cluster → grid
 - Single entrance point across domains
- Single System illusion (SSi)
- Security for remote commands via ssh
- Cluster configuration file (dynamic)
 - Logical sub-cluster partition management
 - Logical meta-cluster partition management
 - Enhanced scalability via logical partitions
- Core infrastructure component of OSCAR cluster distribution

Requirements

- Hard Requirements
 - Rsync
 - SSH or RSH
 - Python 2.0 or greater
 - Perl 5.6 or greater
 - Linux
- Firm Requirements
 - SystemImager 1.50 or greater
- Soft Requirements
 - DHCP
 - DNS or properly configured hosts file
- Operating systems supported
 - Linux
 - RedHat – reference implementation
 - HP-UX
 - Solaris

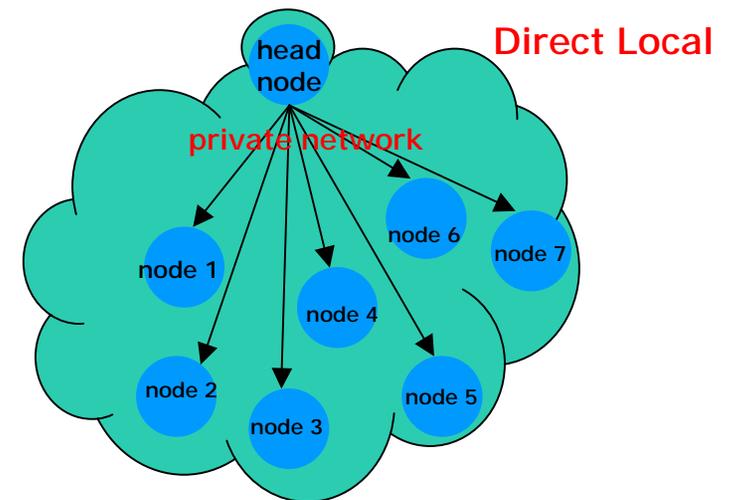
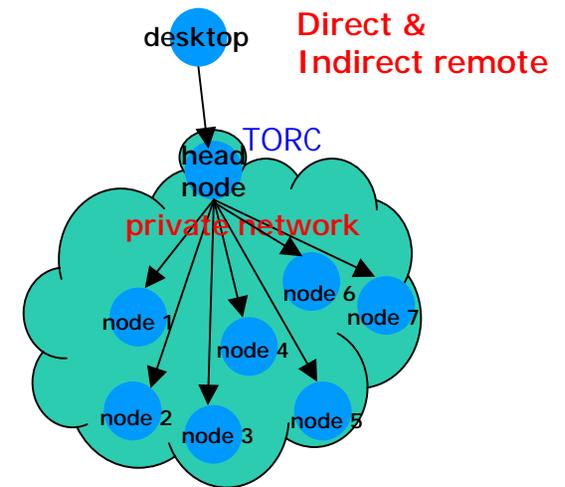
C3 Building Blocks

- System administration
 - `cpushimage` - "push" image across cluster
 - `cshutdown` - Remote shutdown to reboot or halt cluster
- User & system tools
 - `cpush` - push single file -to- directory
 - `crm` - delete single file -to- directory
 - `cget` - retrieve files from each node
 - `ckill` - kill a process on each node
 - `cexec` - execute arbitrary command on each node
 - `cexecs` – serial mode, useful for debugging
 - `clist` – list each cluster available and it's type
 - `cname` – returns a node position from a given node name
 - `cnum` – returns a node name from a given node position

Cluster Classification Scheme

- Direct local
 - The cluster nodes are known at run time
 - The command is run from the head node
- Direct remote
 - The cluster nodes are known at run time
 - The command is not run from the head node
- Indirect remote
 - The cluster nodes are not known at run time
 - The command is not run from the head node

- Notes:
 - Local or remote is checked by comparing the head node names to the local hostname
 - Indirect clusters will execute on the default cluster of the head node specified.



Cluster Configuration File

- default cluster configuration file

- `/etc/c3.conf`

```
Cluster torc { #direct local cluster
  orc-00b:node0
  node[1-4]
  exclude 3
}
Cluster htorc { #indirect remote cluster
  :htorc-00
}
```

- user specified configuration file

- `/...somewhere/list_of_nodes`

```
Cluster auto-gen { #direct remote cluster
  node0.csm.ornl.gov
  node1.csm.ornl.gov
  node2.csm.ornl.gov
  dead node3.csm.ornl.gov
  node4.csm.ornl.gov
}
```

- `node[1-4]` creates list of enumerated types starting with index 0
- zero based index
- command line 0 = node1, 1 = node2, etc...

Cluster Configuration File

• default cluster configuration file

• /etc/c3.conf

```
Cluster torc { #direct local cluster
    orc-00b:node0
    dead place_holder
    node[1-4]
    exclude 3
}
Cluster htorc { #indirect remote cluster
    :htorc-00
}
```

• user specified configuration file

• /...somewhere/list_of_nodes

```
Cluster auto-gen { #direct remote cluster
    node0.csm.ornl.gov
    dead place_holder
    node1.csm.ornl.gov
    node2.csm.ornl.gov
    dead node3.csm.ornl.gov
    node4.csm.ornl.gov
}
```

- node[1-4] creates list of enumerated types starting with index 0
- key **dead** creates null entry for index 0
- command line 0 = no entry, 1 = node1, 2 = node2, etc...

Configuration File Specifics

- Cluster tag
 - **Cluster** followed by the cluster name
 - Cluster name is used on command line
- Offline nodes
 - **Exclude** tag
 - Only applies to node ranges
 - Only applies to the first node range preceding it
 - Possible to exclude using both ranges and specifying single machines

```
node[1-64]
exclude [1-13]
exclude 16
place_holder[65-127]
exclude [65-127]
node[128-256]
exclude [130-152]
```

exclude is very useful when partitioning cluster via dynamic configuration file

- **Dead** tag
 - Only applies to single machine context

```
node1
dead node2
node3
```

dead is very useful when building configuration file dynamically via clist command

Miscellaneous configuration File Information

- **Exclude** and **dead** are NOT RESERVED WORDS
 - **dead4** and **exclude12** are valid node names.
 - C3 searches for white space after the **dead** and **exclude** tags to represent offline declarations
 - Setting nodes as offline when they are offline is required when using node ranges – this preserves node ranges and prevents hanging on dead nodes.
- Do NOT create an *indirect local cluster*
 - Resolving causes an infinite loop
- When using a *indirect remote cluster* the default cluster on the remote head node is executed.
 - Resolving may cause an infinite loop

Miscellaneous configuration File Information

- Cluster Definition Blocks as Meta-clusters (functional clusters)
 - Group based on hardware
 - interface type
 - others...
 - Groups based on software
 - server groups
 - like configurations
 - others...
 - Groups based on role
 - DNS servers
 - NIS servers
 - NFS servers
 - others...
 - Others...
- User specified cluster configuration files
 - Specified at runtime
 - User can create both sub-clusters and super-clusters
 - Useful for scripting

Meta-Cluster Configuration File

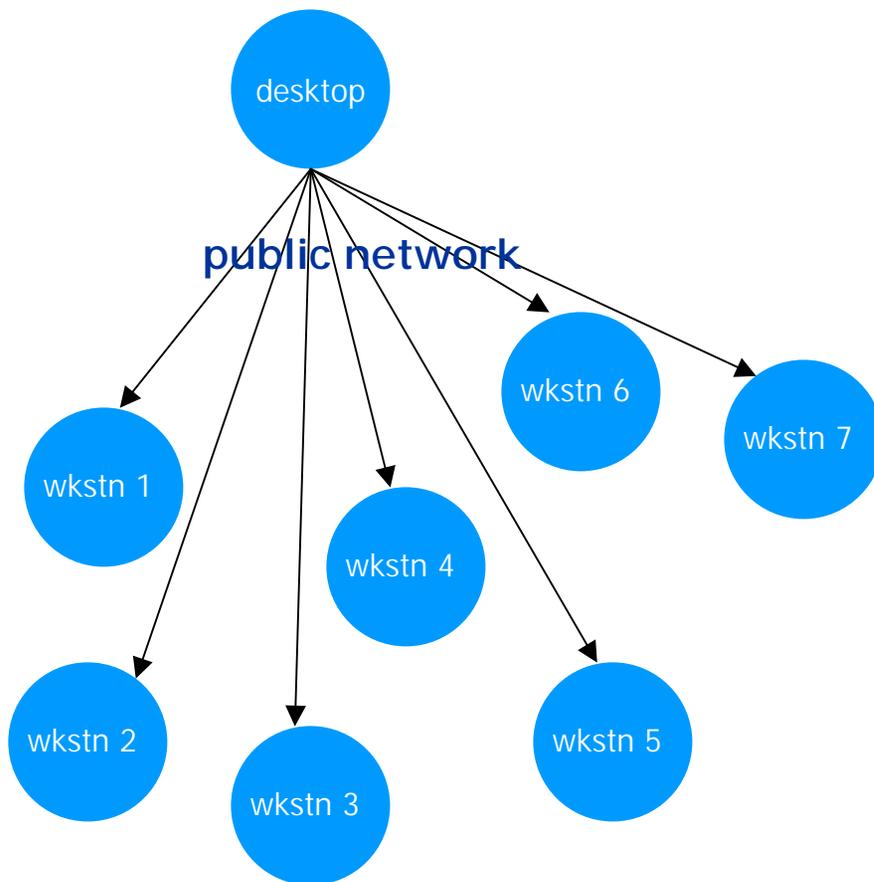
Example c3.conf file using meta-clusters

```
Cluster xtorc {           # Entire cluster defined
  xtorc:node0           # headnode
  dead place_holder    # consumes node0, making node1 = 1
  node[1-64]
}

Cluster gxtorc {         # gigabit ethernet cards
  xtorc:gnode0         # head node
  dead place_holder
  gnode[1-64]
  exclude 12           # gig nic is disabled on this node
}

Cluster pbs_servers { # pbs server group
  xtorc:node0
  node0
  node16
  node32
}
```

Execution Model – Workstation Pool



- desktop knowledge

- wkstn 1
- wkstn 2
- ...
- wkstn 7

- Example c3.conf on desktop

```
Cluster mine {  
  desktop  
  wkstn[1-7]  
}
```

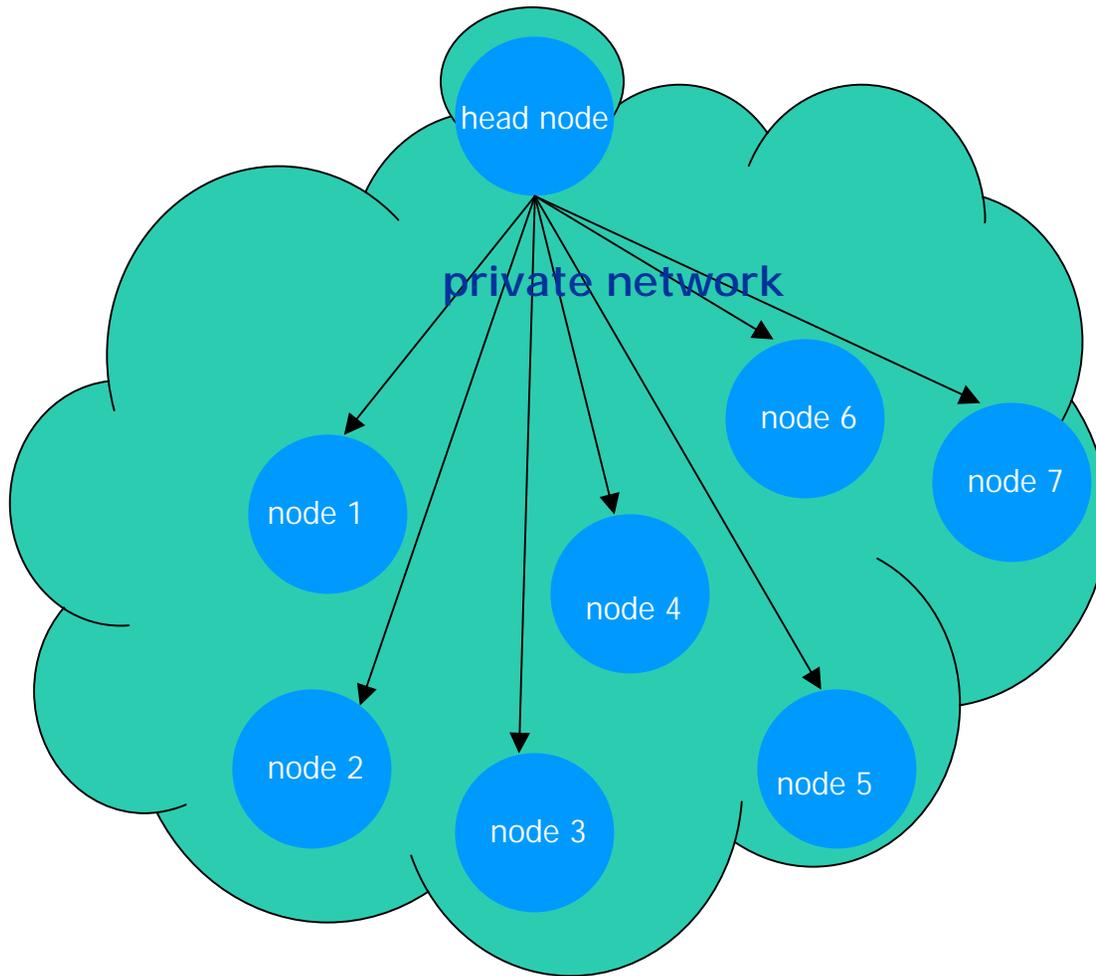
Execution Model: Workstation Pool

Direct local – two examples:

```
cluster range_example { #cluster is a reserved tag, range_example is the cluster name
    desktop           #only one machine name listed, assumed to be both
                    #internal/external interface
    wkstn[1-7]       #used range to define, the [ ] signifies range declaration
    exclude 3
    exclude 5-7
}
```

```
cluster no_range_example {
    desktop
    wkstn1
    wkstn2
    dead wkstn3
    wkstn4
    dead wkstn5
    dead wkstn6
    dead wkstn7
}
```

Execution Model – Cluster Internal



- head node knowledge:

- node 1
- node 2
- ...
- node 7

- Example c3.conf on head node

```
cluster local {  
  headnode  
  node[1-7]  
}
```

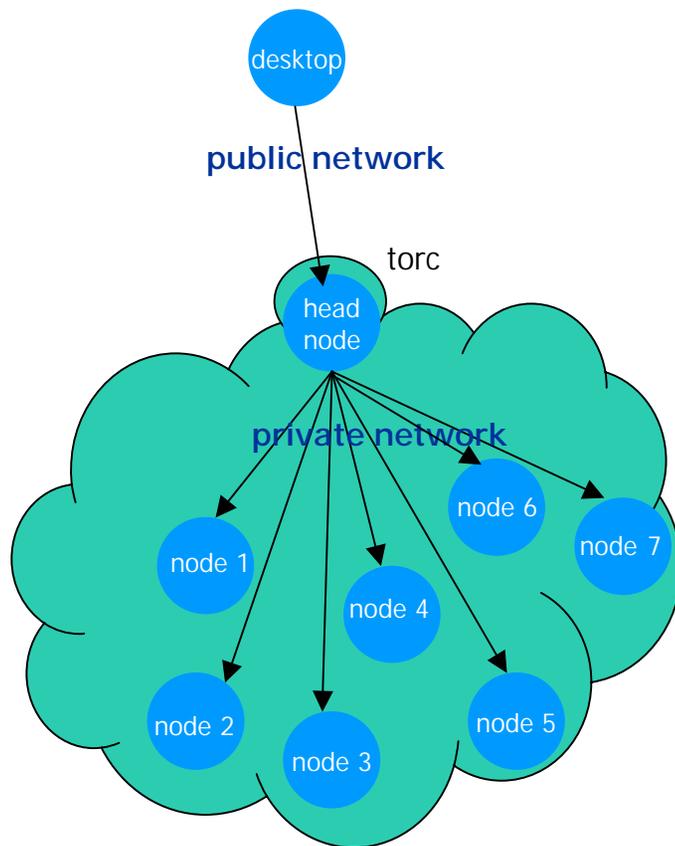
Execution Model: Cluster Internal

Direct local – two examples:

```
cluster example {  
    TORC:node0 # TORC external, node0 internal interface  
    node[1-7]  
}
```

```
cluster example {  
    node0 # node0 internal interface (may be no external interface)  
    node[1-7]  
}
```

Execution Model – External to Cluster



- desktop knowledge:
 - torc – cluster head node

- Example c3.conf on desktop

```
cluster torc {  
    :torc  
}
```

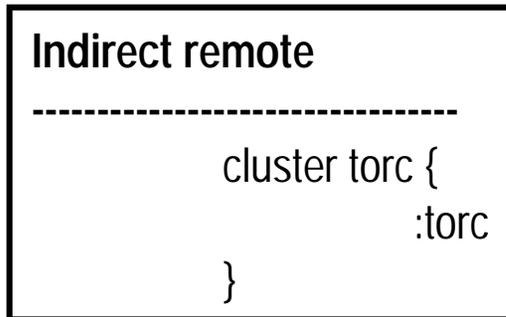
- head node knowledge:
 - node 1
 - node 2
 - ...
 - node 7

- Example c3.conf on head node

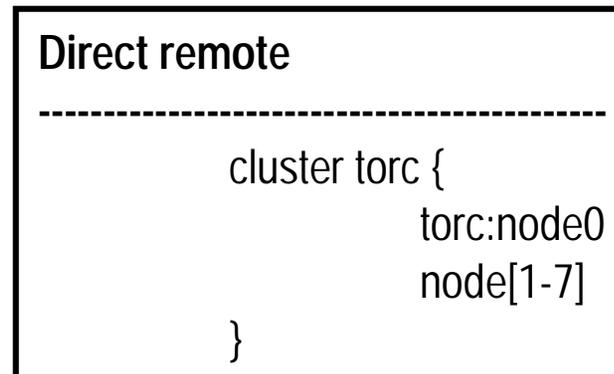
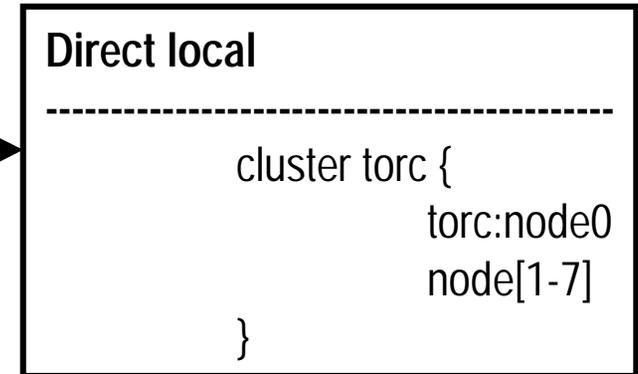
```
cluster local {  
    torc  
    node[1-7]  
}
```

Execution Model: External to Cluster

On desktop



On TORC



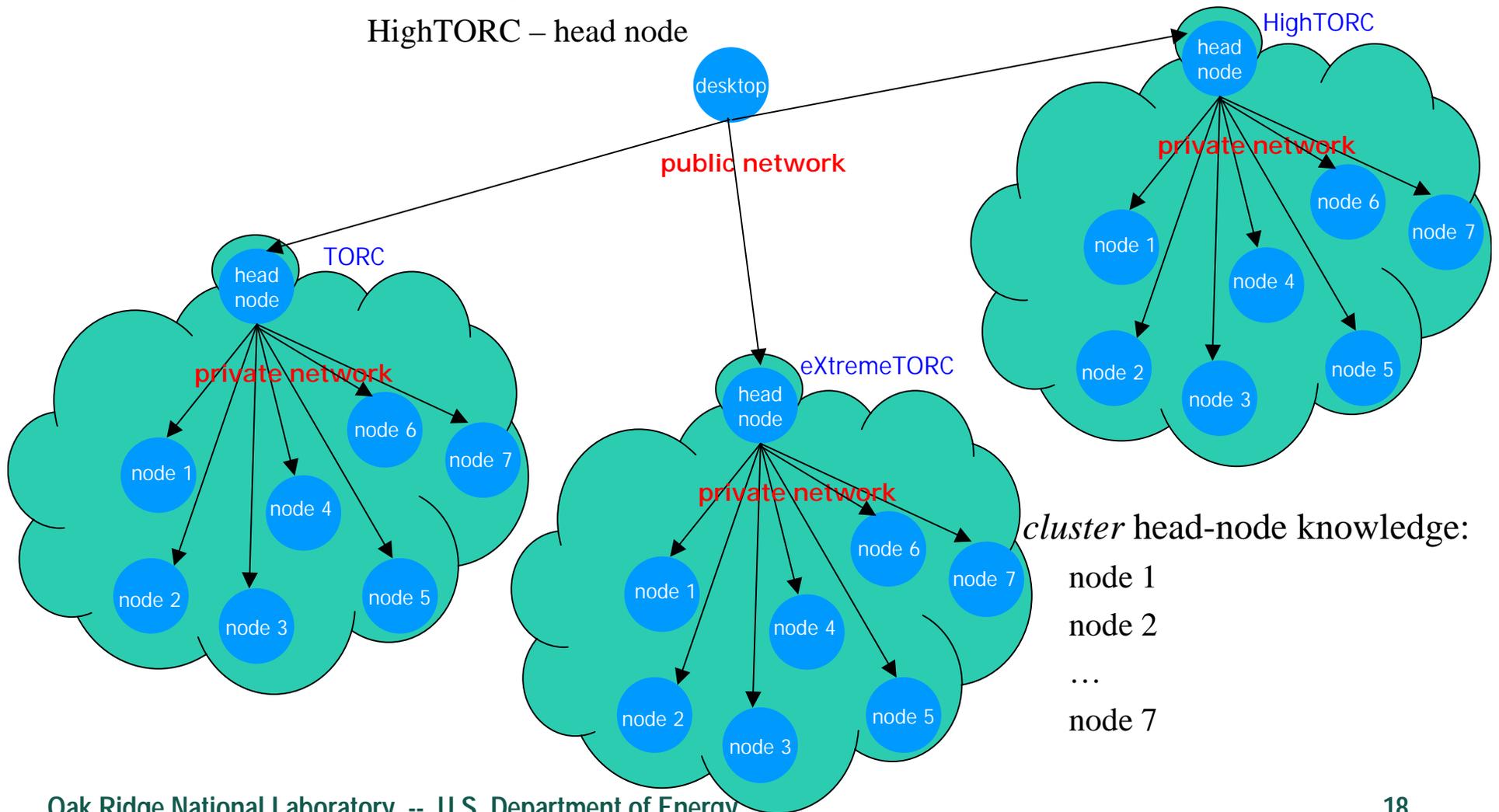
Execution Model: External to Multi-Cluster

desktop knowledge:

TORC – head node

eXtremeTORC – head node

HighTORC – head node



cluster head-node knowledge:

- node 1
- node 2
- ...
- node 7

Execution Model: External to Multi-Cluster

On desktop

```
Indirect remotes (several in one file)
-----
cluster torc {
    :torc
}
cluster exterme_torc {
    :xtorc
}
cluster high_torc {
    :htorc
}
```

On eXtremeTORC

```
Direct local
-----
cluster xtorc {
    xtorc:node0
    node[1-7]
}
```

MACHINE DEFINITIONS (Ranges) on Command Line

- [MACHINE DEFINITIONS] as used in command line
- Position number from configuration file
 - Begin at 0
 - Does not include head node
 - `dead` and `exclude` maintain a nodes position
- Format on command line
 - First cluster name from configuration file with a colon
 - `Cluster2:` would represent all nodes on cluster2
 - `:` signifies default cluster
 - ranges and single nodes are separated by a comma
 - `Cluster2:1-5,7` executes on nodes 1, 2, 3, 4, 5, 7
 - `:4` executes node at position 4 on the default cluster

Example: `cexec : torc:1-5,7 hostname`

cexec

Usage: cexec(s) [OPTIONS] [MACHINE_DEFINITIONS] command

--help -h	display help message
--file -f <filename>	alternate cluster configuration file if one is not supplied then <i>/etc/c3.conf</i> will be used
-i	interactive mode, ask once before executing
--head	execute on head node – not on compute nodes

Using cexecs executes the serial version of cexec

cexec

- to execute a command with wildcards on several clusters

```
cexec cluster1: cluster2:2-5 "ls /tmp/pvmd*"
```

This will execute "ls /tmp/pvmd*" on each compute node on cluster one and nodes 2, 3, 4, and 5 on cluster2. Notice the use of the quotes. This keeps the shell from interpreting the command until it reaches the compute nodes.

- Using pipes

```
cexec "ps -A |grep a.out"  
cexec ps -A |grep a.out
```

In the first example the | symbol is enclosed in the quotes. In this case "ps -A|grep a.out" is executed on each node. In this way you get the standard cexec output format with a.out in each nodes block if it exists.

In the second example "ps -A" is executed on each node and the all the a.out lines are grep'ed out. This demonstrates that placement of ""s is very important.

General Usage Notes

- By default C3 does not execute commands on the head node
 - Use `--head` option to execute only on the head node
- Interactive option only asks once before execution
- Commands only need to be homogeneous within itself
 - Example: binary and data on an Intel and HPUX
 - This example assumes NFS within individual clusters
 - Data can be pushed to both systems

```
cpush --head intel: hp: data.txt
```
 - Binary for each cluster

```
cpush --head intel: app.intel app
cpush --head hp: app.HPUX app
```
 - Then execute app

```
cexec --head intel: hp: app      fires on head node only
or
cexec intel: hp: app            fires on all nodes
```

Single cluster example

Rolling cluster upgrade:

- Build new image on single node
- Retrieve image with Systemimager
`getimage -golden-client=node0 -image=test_image`
- Push test image to subset of cluster (nodes 1 through 8 inclusive)
`cpushimage :1-8 test_image`
- Run needed tests on new image
- If the image works push image to entire cluster
`cpushimage test_image`
- If the test image did not work simply revert to known good image
`cpushimage good_image`

Single cluster example

Install an rpm on default cluster

- Push rpm out to cluster nodes
`cpush example-1.0-1.rpm`
- Use RPM to install application
`cexec rpm -i example-1.0-1.rpm`
- Check for errors in installation
`cexec rpm -q example`

Multiple cluster example

Install an rpm on two clusters

- First push rpm out to cluster nodes
`cpush : xtorc: example-1.0-1.rpm`
- Use RPM to install application
`cexec : xtorc: rpm -i example-1.0-1.rpm`
- Check for errors in installation
`cexec : xtorc: rpm -q example`

<code>: xtorc:</code>	specifies multiple clusters
<code>:</code>	represents the default cluster
<code>xtorc:</code>	represents cluster xtorc
all nodes on both clusters participate	

Multiple cluster example

Add a user

- Create a local add_user script
 - Very site specific
 - Example script given in the C3 tarball under the contrib directory
 - Calls the Linux useradd utility
 - Uses cpush after the useradd to push out new passwd/shadow files
 - Generates users ssh keys
- Example usage of above script from C3 contrib
 - `./add_user sgrundy users`
adds user sgrundy with group users to the local cluster
- Call add_user script with cexec
 - `cexec : torc: /root/bin/add_user sgrundy users`

Multiple cluster use notes

- High level administrators can easily set policies on several clusters from single access point.
 - Federated clusters – those within single domain
 - Meta-clusters – clusters spanning administrative domains
- **Very powerful, but with power comes danger...**
 - user error can be VERY bad
 - homogeneous within “self” is very important
 - simple `crm --all *` may bring down MANY nodes
 - Extend nearly all Unix/Linux gotcha’s to multiple clusters/many nodes – and very fast

Contact Information

<u>torc@msr.csm.ornl.gov</u>	contact ORNL cluster team
<u>www.csm.ornl.gov/torc/C3</u>	version 3.1.1 (current release)
<u>www.csm.ornl.gov/TORC</u>	ORNL cluster team site
<u>www.openclustergroup.org</u>	C3 v3.1.1 included in OSCAR 1.4