



Scalable C3 Power Tools

Presented by: John Mugler

Brian Luethke and Stephen L. Scott

Oak Ridge National Laboratory
Computer Science and Mathematics Division
Network and Cluster Computing Group

24 June 2003
ClusterWorld Conference
San Jose CA, USA

Motivation for building cluster tools

...Problem

- System Administrators DO NOT scale
 - Install / update operating system
 - Install / delete applications
 - Add / remove users
 - Add / remove nodes
 - Etc.
- Users DO NOT scale
 - Install applications
 - Move data files
 - Launch applications
 - Interact with active jobs
 - Etc.

...Solution

- Tools that...
 - Treat cluster as single machine
 - Single System Image (SSI)
 - Single System illusion (SSi)
 - Scale from 1-to-N nodes
 - DOE target is 10,000's of nodes
 - Scale to Federated clusters
 - Scale to Grid
 - Do anything only 1x
 - Install / configure / others...
 - Work smart, not hard!
 - Easy to → learn – use – adapt
 - Open source AND free???

Cluster Power Tools

C3: Cluster, Command, and Control

- Scalable systems administrator
- Single system illusion (SSI) for clusters and multi-clusters
- Application and administration tools for secure cluster through multi-cluster access crossing administrative domains

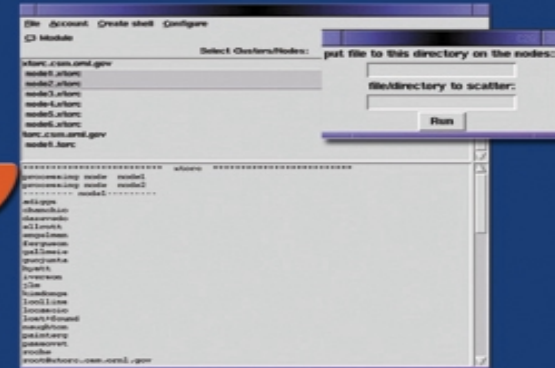
Applications

C3 Command Line Cluster Power Tools

- | | |
|--------------|-------------|
| ▪ cexec | ▪ clist |
| ▪ cpush | ▪ cname |
| ▪ cget | ▪ cnum |
| ▪ cpushimage | ▪ ckill |
| ▪ crm | ▪ cshutdown |



C2G: Cluster Control GUI



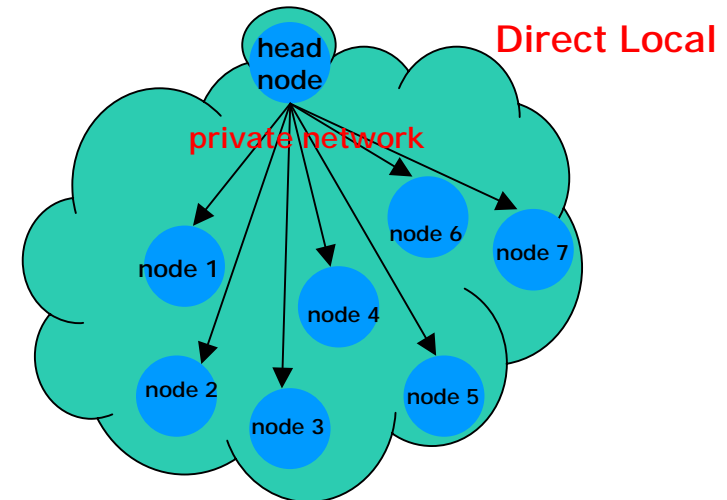
- Generic interface into common clustering applications
- Runtime pluggable modules may load varying front end tools
- Node browser provides ability to select entire clusters or specific nodes via a mouse click

Building Blocks

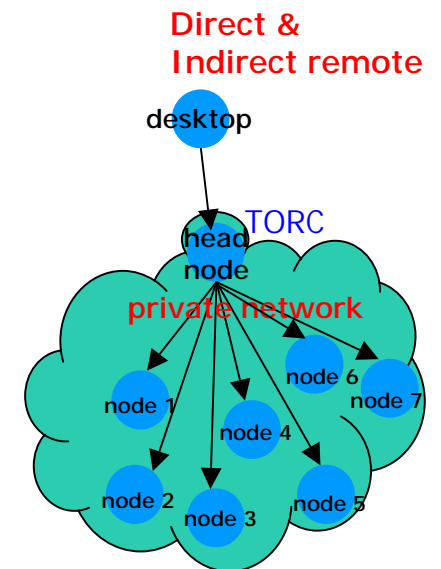
- System administration
 - `cpushimage` - “push” image across cluster
 - `cshutdown` - Remote shutdown to reboot or halt cluster
- User tools
 - `cpush` - push single file -to- directory
 - `crm` - delete single file -to- directory
 - `cget` - retrieve files from each node
 - `ckill` - kill a process 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 name from a given node position
 - `cnum` – returns a node position from a given node name

C3 Tools 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
 - Initiated from outside the cluster
- Indirect remote
 - The cluster nodes are not known at run time
 - The command is not run from the head node
 - Initiated from outside the cluster



- 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.
 - Can not have an *indirect local* cluster
 - information must be somewhere
 - When using a *indirect remote* cluster the default cluster on the remote head node is executed.
 - Resolving may cause an infinite loop
 - machine A → machine B → machine A → machine B → ...



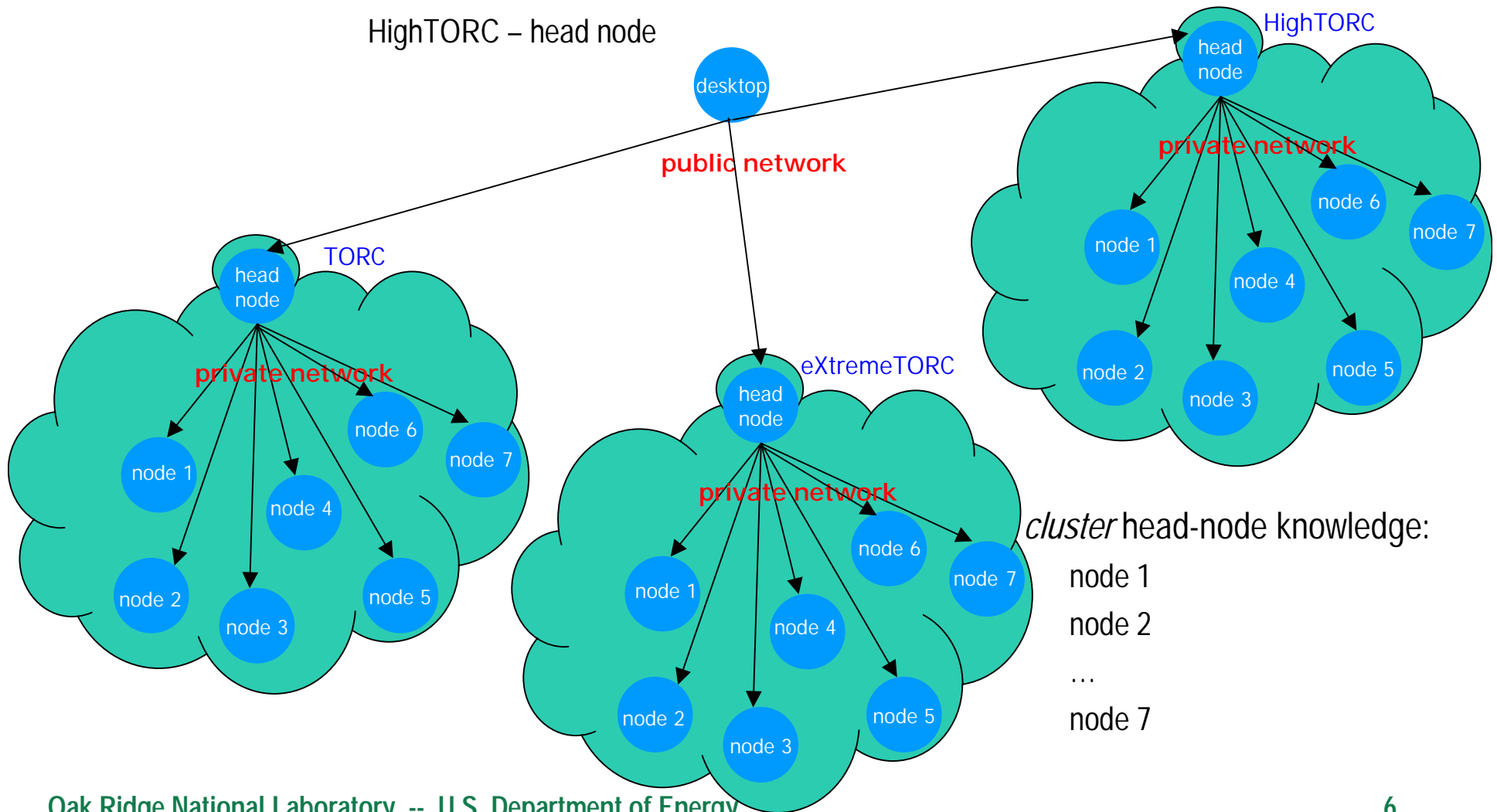
Execution Model: External to Multi-Cluster

desktop knowledge:

TORC – head node

eXtremeTORC – head node

HighTORC – head node



cluster head-node knowledge:

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

Standard C3 configuration file

```
cluster first { # The default cluster (first in the list) and a direct local cluster
  external:internal      # the head-node
  node1                  # compute nodes
  node2
  dead node3            # offline node w/o node ranges
}
```

```
cluster second { # direct remote - notice that the only difference between this
                 # cluster and the first cluster is that the head-node
                 # represents both the external and internal interface
  xtorc
  node[1-10] # example of node ranges
}
```

```
cluster feanor { # direct remote cluster
  feanor:node0
  node[1-64]
  exclude 4           # setting nodes offline in a range
  exclude [32-37]    # node can be marked offline in ranges also
  node[128-132]
}
```

```
cluster torc { # indirect remote cluster
  :torc          # external interface of remote cluster
}               # there is no "local" cluster here
```

range by ip address looks like:
192.168.1.[1-64]

may also specify individual nodes:
node3.csm.ornl.gov
node4.csm.ornl.gov
etc...

Proper use of **dead** and **exclude**

```
cluster not_reserved_words { # notice below that what separates dead and
                             # exclude from being hostname is
                             # that in the case of "dead" there are two
                             # words on a single line.
                             # in the case of "exclude" there is a space
                             # between the word exclude and the
                             # range. While this is possible it is not
                             # recommended.
external:internal           # head-node
dead                       # first node named "dead"
dead dead1                 # second node "dead1" offline
exclude[1-64]             # 64 nodes exclude1..exclude64
exclude [5-6]            # exclude5 and exclude6 are offline
exclude 8                 # exclude8 is offline
```

dead and **exclude** are not reserved words

they are treated as "offline" specifier when syntactically legal to do so

Node ranges similar to enumerated types

```
cluster example1 { # no placeholder, node1 is in position 0
  external:node0
  node[1-64]
}
```

```
cluster example2 { # placeholder added, node1 is in position 1
  outside:node0
  dead placeholder    # dummy entry to change to 1 based indexing
  node[1-64]
}
```

cpush example1:0 example2:1 /etc/passwd

**pushes passwd to first node in each of the
above clusters**

MACHINE DEFINITIONS & (Ranges) on Command Line

- Position number from configuration file
 - Begin at 0
 - Does not include head node
 - (--head) to execute on head node
 - **dead** and **exclude** maintain a nodes position format on command line
 - First cluster name from configuration file with a colon
 - **Cluster2:** represents 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 on node at position 4 on the default cluster

`cexec :1-4 cluster2: hostname`

cpush

Usage: **cpush** [OPTIONS] [MACHINE DEFINITIONS] source [target]

- h, --help display help message
- f, --file <filename> alternate cluster configuration file, default is /etc/c3.conf
- l, --list <filename> list of files to push (single file per line, column1=SRC column2=DEST)
- i interactive mode, ask once before executing
- head execute command on head node, does not execute on compute nodes
- nolocal the source file or directory lies on the head node of the remote cluster
- b, --blind pushes the entire file (normally cpush uses rsync)
- all executes on all clusters in the configuration file

cpush

- To move a single file

```
cpush /etc/passwd
```

this will sync /etc/passwd on all compute nodes with the one on the local machine.

- To move a single file, renaming it on the cluster nodes

```
cpush /root/passwd.restricted /etc/passwd
```

push an alternate passwd file to the clusters compute nodes

- To move a set of files listed in a file

```
cpush --list=/home/filelist
```

This pushes each file in the filelist where it is specified to send it. Filelist format is on the next slide.

Notes on using a filelist

- One file per line
- If no destination is specified then it will push the file to the location it is on the local machine
- Examples:
 - `/etc/group`
 - `/root/lilo.conf /etc`
 - `/root/passwd.new /etc/passwd`
 - The first line pushes the file `"/etc/group"` to `/etc` on each compute node
 - The second line pushes the file `"/root/lilo.conf"` to `/etc` on each compute node
 - The third line pushes the file `"/root/passwd.new"` to `/etc` on each compute node renaming the file to `"/etc/passwd"`
- Note: All options on the command line are applied to each file.

cpush

- To push a file hosted on the remote clusters head node to its compute nodes

```
cpush --nolocal : rats: gundam: /etc/passwd
```

The `--nolocal` option pushes the file from the remote machine to each of its compute nodes. In this case the default cluster, rats, and gundam will sync `/etc/passwd`.

- To push a file without using the rsync algorithm

```
cpush --blind ~/proj1/data.bin
```

This first checks if `~/proj1/data.bin` exists, if it does it deletes it and then pushes the file to the compute nodes

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 */etc/c3.conf* will be used
- i interactive mode, ask once before executing
- head execute command on head node, does not execute on

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. Example output on next slide.

cexec quotation example

- cexec "ps -A|grep xinetd"

```
***** local *****
processing node node1
***** local *****
processing node node2
***** local *****
----- node1-----
9738 ?    00:00:00 xinetd

----- node2-----
4856 ?    00:00:00 xinetd
```

- cexec ps -A |grep xinetd

```
9738 ?    00:00:00 xinetd
4856 ?    00:00:00 xinetd
```

Scalable format

- One cluster per configuration file
- Must be a local cluster
- Cluster broken up into smaller sub-clusters
 - Perfect squares are best
 - 8 8-way fan outs for a 64 node cluster
- Maximum fan out for each node (whichever is smaller)
 - Max the hardware can handle (usually around 64)
 - Size of switch for performance
 - With 64 64 way sub-clusters C3 scales up to 4096 nodes

To stage, or not to stage

- Two ways to configure cluster
 - Staging nodes include themselves in their list of responsibilities
 - Usually what you want – the command runs on every single node in the cluster
 - Staging nodes do not include themselves in their list of responsibilities
 - Dedicated staging nodes
 - System administrator uses
 - Staging nodes may have slightly different software configurations
- Advised that the administrator creates both configuration files

Scalable release configuration file options – 8-way fan

```
cluster part1 { # direct scalable
  node1
  node[2-8]
}
cluster part2 {
  node9
  node[10-16]
}
cluster part3 {
  node17
  node[18-24]
}
cluster part4 {
  node25
  node[26-32]
}
cluster part5 {
  node33
  node[34-40]
}
cluster part6 {
  node41
  node[42-48]
}
cluster part7 {
  node49
  node[50-56]
}
cluster part8 {
  node57
  node[58-64]
}
```

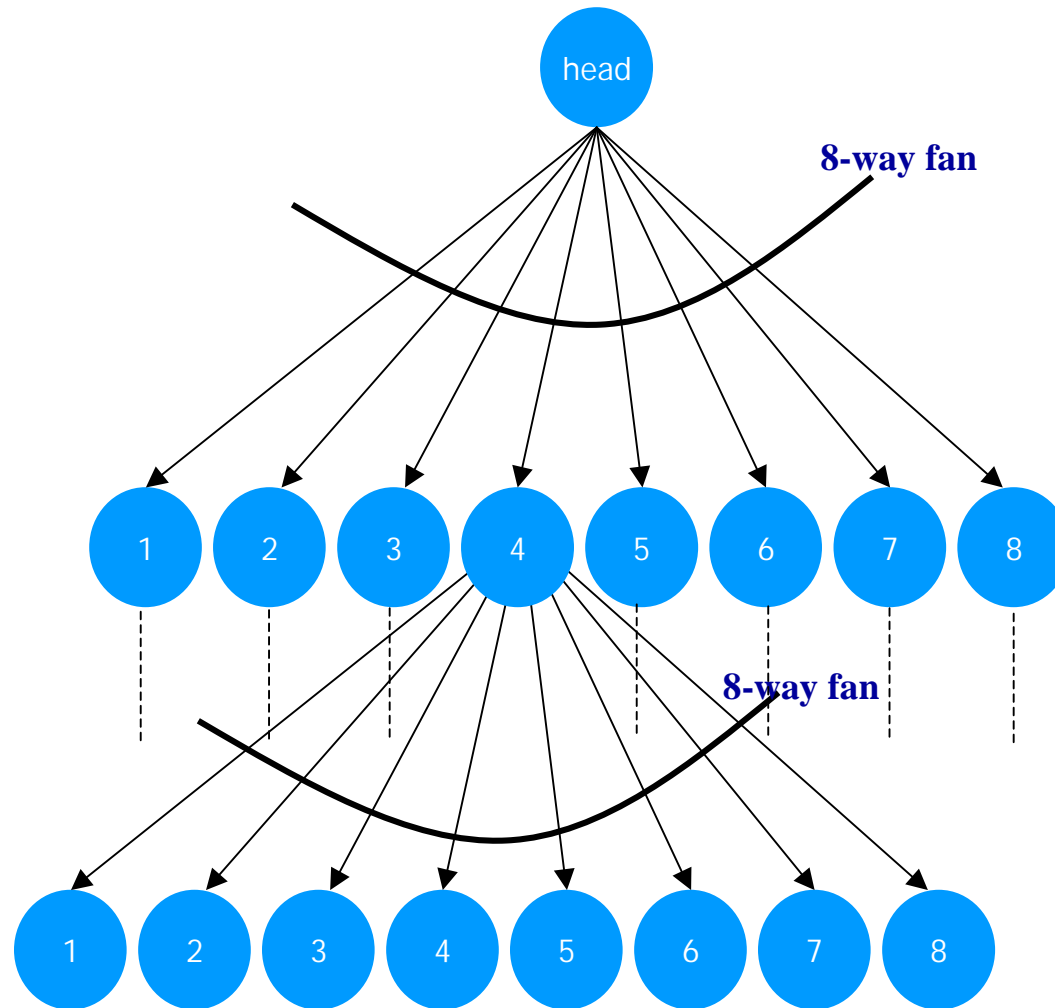
Head nodes configuration

```
cluster part1 { #indirect scalable
  :node1
}
cluster part2 {
  :node9
}
.
.
.
cluster part8 {
  :node57
}
```

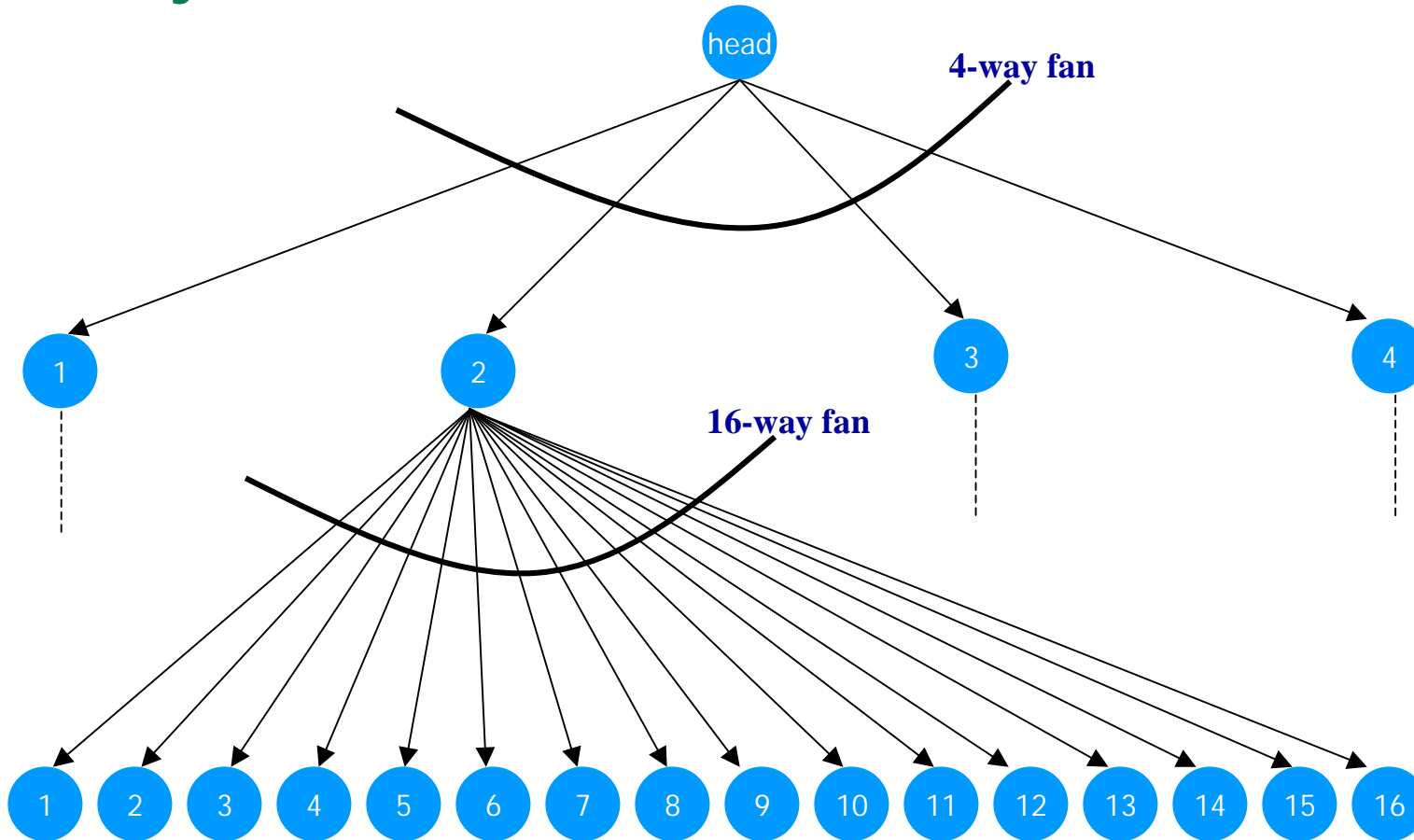
node1's configuration file

```
cluster part1 { # direct local
  node1
  node[2-8]
}
```

8-way



4x16-way



Scalable Command Line

- Essentially the same as non-scalable
- Needs **one** extra option if default configuration file is a scalable configuration file
 - **--all**
`cpush --all /etc/passwd`
- Ranges on command line
 - Need to use `cnum` to find sub-cluster/position for nodes
`cnum --all node46 node64`
 - Explicitly state each sub-cluster/position with no `--all`
`cexec part6:5 part7: part8:0-7 hostname`

Example: Managing Processes/Services

- SSHD configuration
 - Retrieve sshd configuration file from node 1 of the cluster
`cget :0 /etc/ssh/ssd_config .`
 - Edit configuration to your needs
 - Distribute configuration file
`cpush --all ./sshd_config /etc/ssh/sshd_config`
 - Restart service
`cexec --all service sshd restart`
- Process Management
 - Killing a users process
`ckill --all --user sgrundy c_model`
 - Killing all processes of a given name
`ckill --all -u ALL -s 9 a.out`

Example: File management

- Distributing data files (staging nodes included in their responsibilities list)
 - Two methods
 - NFS mounted file system (pull)
`cexec --all cp ~/proj/data.bin /tmp`
 - Non NFS mounted file system (push)
`cpush --all ~/proj/data.bin /tmp`
- Distributing data files (staging nodes not included in their responsibilities list)
 - Two methods
 - NFS mounted file system (pull)
`cexec --all cp ~/proj/data.bin /tmp`
 - Non NFS mounted file system (push)
`cpush --all --head ~/proj/data.bin /tmp`
`cpush --all --nolocal /tmp/data.bin`

Miscellaneous information

- You can only have ONE scalable cluster per configuration file
- Two commands do not receive benefits from the the scaleable model
 - cget
 - Gather operation is still many transferring to a single point
 - cpushimage
 - SystemImager does not support staging of images to nodes
 - Can manually stage images (next slide)
 - Scalable/non-scalable versions can coexist
 - Use the C3_CONF to set default configuration file
 - Use --file option at runtime

Scalable cpushimage

- SystemImager does not support staging images natively
 - Golden_client MUST not be one of the staging nodes
 - Staging nodes MUST not include themselves in their list of responsibilities
- You manually add images to staging nodes
 - `cexec --head --all getimage -image temp_image -golden_client node2`
- Retrieve a staging nodes image to head node
 - `getimage -image staging_node_full -golden_client node1`
- Retrieve a standard compute nodes image (same node as imaged before)
 - `getimage -image full_node -golden_client node2`
- Push staging image to staging nodes
 - `cpushimage --head --all -image staging_node_full`
- Push image to compute nodes
 - `cpushimage --all temp_image`
- Push compute image to staging nodes
 - `cpushimage --all --head full_node`

Timings: 4x16-way File transfer via cpush

1 Megabyte	Total Time	Load Average Head
Staging	2.36s	n/a*
Fan-out	14.35s	.80
Scalable Total	16.71s	not relevant
Non-Scalable	31.70s	10.57

10 Megabyte	Total Time	Load Average Head
Staging	12.43s	1.27
Fan-out	44.98s	.80
Scalable Total	57.51s	not relevant
Non-Scalable	2m 52.99s	48.03

100 Megabyte	Total Time	Load Average Head
Staging	1m 54.06s	3.30
Fan-out	5m 57.21s	.80
Scalable Total	7m 51.27s	not relevant
Non-Scalable	27m 00.67s	55.48

1 Gigabyte (1000 Megabyte)	Total Time	Load Average Head
Staging	20m 06.72s	16.48
Fan-out	59m 36.34s	.80
Scalable Total	1h 19m 43.06s	not relevant
Non-Scalable	4h 28m 27.90s	55.51

* Insufficient time was required to result in a change of load for the head node.
 Oak Ridge National Laboratory -- U.S. Department of Energy

Contact Information & Questions

scottsl@ornl.gov
luethkeb@ornl.gov
muglerj@ornl.gov

Stephen L. Scott
Brian Luethke
John Mugler

torc@msr.csm.ornl.gov

contact ORNL cluster team

www.csm.ornl.gov/torc/C3

version 3.1.2 (current release)

www.csm.ornl.gov/TORC

ORNL team site

www.openclustergroup.org

C3 v3.1.2 included in OSCAR 2.2.1