



# Lab Session

---

Rocks-A-Palooza I

Track 1

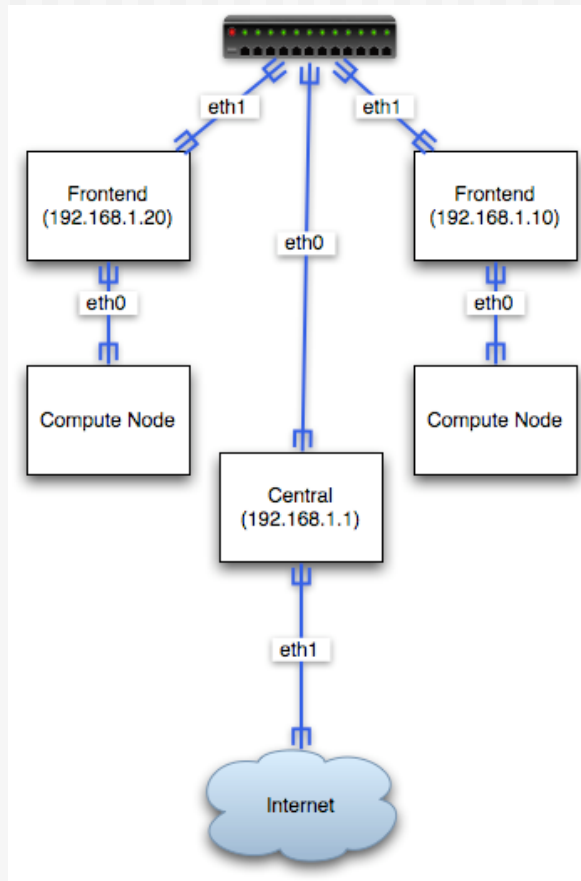
Session IV

# Cluster Building Time

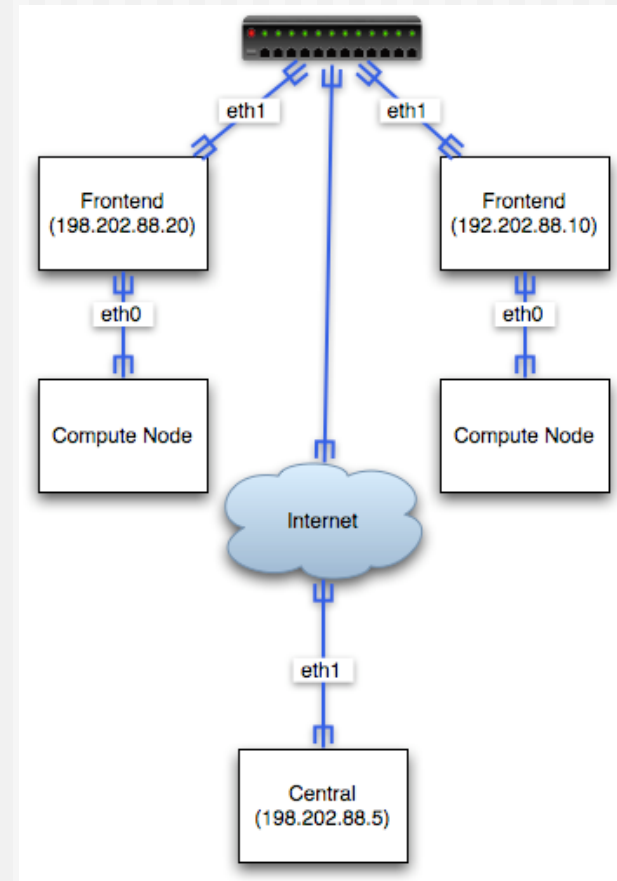
- ◆ Break into Groups
- ◆ Every Group Grab
  - 2 Servers
  - 2 Power Cords
  - 2 Ethernet Cables
    - 1 long
    - 1 short
  - 1 Keyboard / Mouse
  - 1 Monitor
- ◆ Small Clusters
  - 1 frontend
  - 1 compute
  - 1 cross-over Ethernet cable (no switch)



# Today's Lab Network



lab



reality

# Network Information

## ◆ Frontend Addresses

- ⇒ 192.168.1.10
- ⇒ 192.168.1.20
- ⇒ 192.168.1.30
- ⇒ 192.168.1.40
- ⇒ 192.168.1.50
- ⇒ 192.168.1.60
- ⇒ 192.168.1.70
- ⇒ 192.168.1.80

IP Address	192.168.1.xx
Netmask	255.255.255.0
Gateway	192.168.1.1
Nameserver	198.202.75.26

# Start Installing Your Frontend

## ◆ Installation Methods

- CDs
- Central

## ◆ CD

- Slow
- Does not require a network
- Type frontend
- Left side of room

## ◆ Central

- Fast
- Requires a network
- Type frontend `central=192.168.1.1`
- Right side of room

NPACI Rocks Cluster Distribution

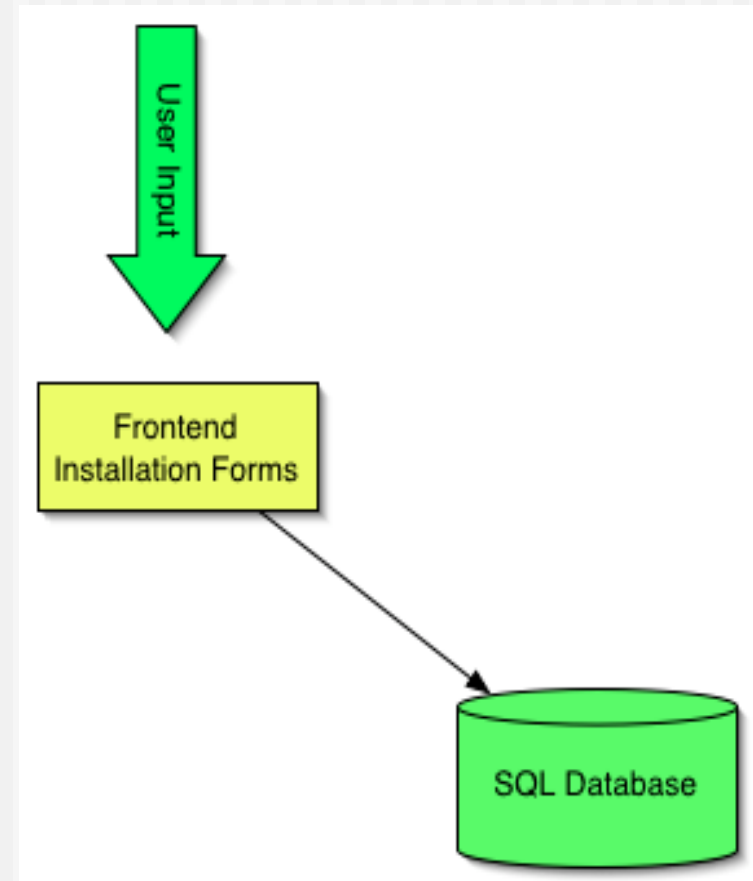
What do you want to kickstart?

- Frontend:  
type "frontend"
- Upgrade your frontend:  
type "frontend upgrade"
- Frontend Network Install  
type "frontend central=name"  
where name is "Rocks", or the  
FQDN of your central server.
- Rescue  
type "frontend rescue"
- Cluster node:  
do nothing or press return

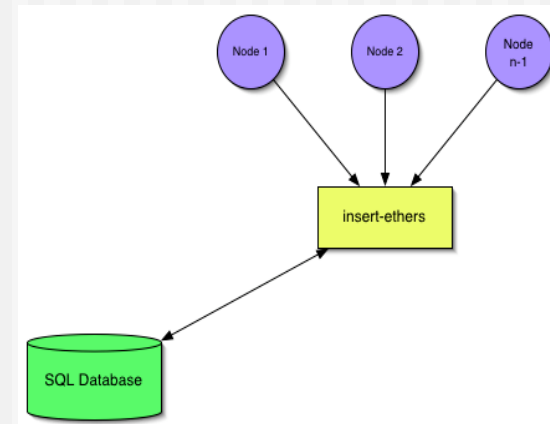
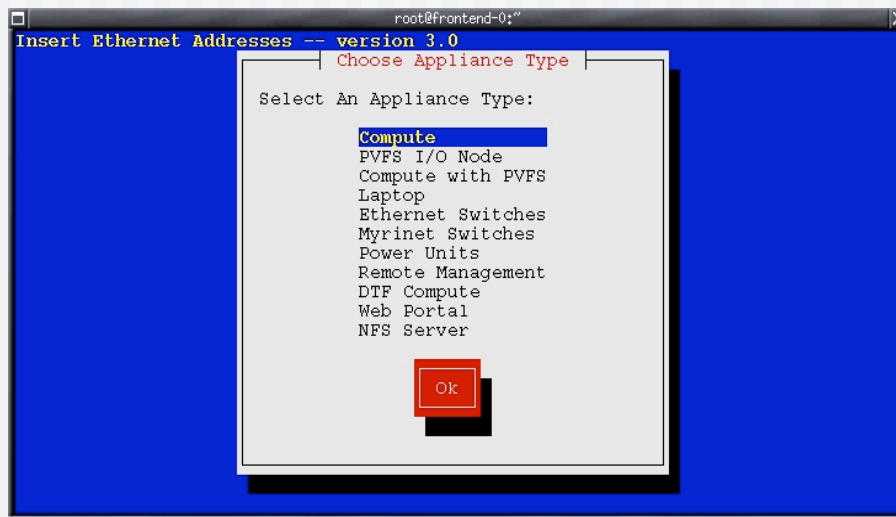


# Interactive Screen

- ◆ Fill out the screens we just talked about
- ◆ Use the provided network information
- ◆ Choose your own password
- ◆ All information goes into the cluster database



# Add Compute Node with Insert-ethers



- ◆ Collect the Ethernet MAC address of cluster nodes
- ◆ Only done once, during integration
- ◆ Populates cluster database

# Open Lab

---

- ◆ Rocks-A-Palooza
  - ➔ Is about you guys
  - ➔ Other topics
  - ➔ Questions
- ◆ Adult Swim
  - ➔ Go nuts on your clusters
  - ➔ Globus
  - ➔ SGE
  - ➔ PBS
  - ➔ Configuration Graph



**[adult swim]**



# Simple MPI Program

```
1: #include <stdio.h>
2: #include "mpi.h"
3:
4: int
5: main(int argc, char *argv[])
6: {
7:     int    numprocs;
8:     int    myid;
9:     int    namelen;
10:    char    processor_name[MPI_MAX_PROCESSOR_NAME];
11:
12:    MPI_Init(&argc, &argv);
13:
14:    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
15:    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
16:    MPI_Get_processor_name(processor_name, &namelen);
17:
18:    fprintf(stderr, "Process %d on %s\n", myid, processor_name);
19:
20:    MPI_Barrier(MPI_COMM_WORLD);
21:
22:    sleep(120);
23:
24:    MPI_Finalize();
25: }
```

# Simple MPI/SGE Submit Script

```
#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash

MPI_DIR=/opt/mpich/gnu

$MPI_DIR/bin/mpirun -np $NSLOTS -machinefile $TMPDIR/machines hello
```

# Compile / Run

---

## ◆ Compile

➔ `/opt/mpich/gnu/bin/mpicc -o hello hello.c`

## ◆ Run

➔ `qsub -pe mpich 2 hello.sh`

## ◆ Monitor

➔ `qstat`

# Example Run

```
mjk@rocks-52:~ — bash (ttyp1)

[mjk@rocks-52 mjk]$ /opt/mpich/gnu/bin/mpicc -o hello hello.c
[mjk@rocks-52 mjk]$ qsub -pe mpich 2 hello.sh
your job 4773 ("hello.sh") has been submitted
[mjk@rocks-52 mjk]$ qstat
job-ID prior name      user      state submit/start at   queue      master  ja-task-ID
-----
  4773    0 hello.sh   mjk       qw    05/17/2005 15:23:30
[mjk@rocks-52 mjk]$ qstat
job-ID prior name      user      state submit/start at   queue      master  ja-task-ID
-----
  4773    0 hello.sh   mjk       r    05/17/2005 15:23:41 compute-0- SLAVE
  4773    0 hello.sh   mjk       r    05/17/2005 15:23:41 compute-0- MASTER
          0 hello.sh   mjk       r    05/17/2005 15:23:41 compute-0- SLAVE
[mjk@rocks-52 mjk]$ ls -l hello.sh.*
-rw-r--r--  1 mjk      mjk           62 May 17 15:23 hello.sh.o4773
-rw-r--r--  1 mjk      mjk          106 May 17 15:23 hello.sh.po4773
[mjk@rocks-52 mjk]$ cat hello.sh.o4773
Process 0 on rocks-62.sdsc.edu
Process 1 on rocks-62.sdsc.edu
[mjk@rocks-52 mjk]$ qstat
[mjk@rocks-52 mjk]$ hostname
rocks-52.sdsc.edu
[mjk@rocks-52 mjk]$
```

# HPL.dat

```

HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
1000 Ns
1            # of NBs
64 NBs
1            # of process grids (P x Q)
1 Ps
2 Qs
16.0         threshold
3            # of panel fact
0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
8            NBMINS (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
2            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1            DEPTHS (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
80           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)

```

# Example HPL Run

```

mjk@rocks-52:~ -- bash (ttty1)
[mjk@rocks-52 mjk]$ cp /var/www/html/rocks-documentation/3.3.0/examples/HPL.dat .
[mjk@rocks-52 mjk]$ qsub -pe mpich 2 hpl.sh
your job 4776 ("hpl.sh") has been submitted
[mjk@rocks-52 mjk]$ qstat
job-ID prior name      user      state submit/start at    queue      master  ja-task-ID
-----
  4776    0 hpl.sh      mjk       qw    05/17/2005 18:11:43
[mjk@rocks-52 mjk]$ qstat
[mjk@rocks-52 mjk]$ cat hpl.sh.o4776
=====
HPLinpack 1.0 -- High-Performance Linpack benchmark -- September 27, 2000
Written by A. Petitet and R. Clint Whaley, Innovative Computing Labs., UTK
=====

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      : 1000
NB     : 64
P      : 1
Q      : 2
PFACT  : Left    Crout    Right
NBMIN  : 8
NDIV   : 2

```

# Linpack Scaling

- ◆ Then edit 'HPL.dat' and change:
  - 1 Ps
  - ⇒ To:
    - 2 Ps
  - ⇒ The number of processors Linpack uses is  $P * Q$
- ◆ To make Linpack use more memory (and increase performance), edit 'HPL.dat' and change:
  - 1000 Ns
  - ⇒ To:
    - 4000 Ns
  - ⇒ Linpack operates on an  $N * N$  matrix
- ◆ Submit the (larger) job:
  - ⇒ `qsub qsub-test.sh`

# Others Tasks

---

## ◆ Globus

- ➔ See grid roll usersguide
- ➔ Setup user keys
- ➔ `globus-job-run localhost /bin/hostname`
- ➔ `globus-job-run localhost/jobmanager-sge`

## ◆ Adding RPMs to nodes

- ➔ See usersguide for graph instructions

## ◆ Rebuild with Central/CDROM





# Questions?

---