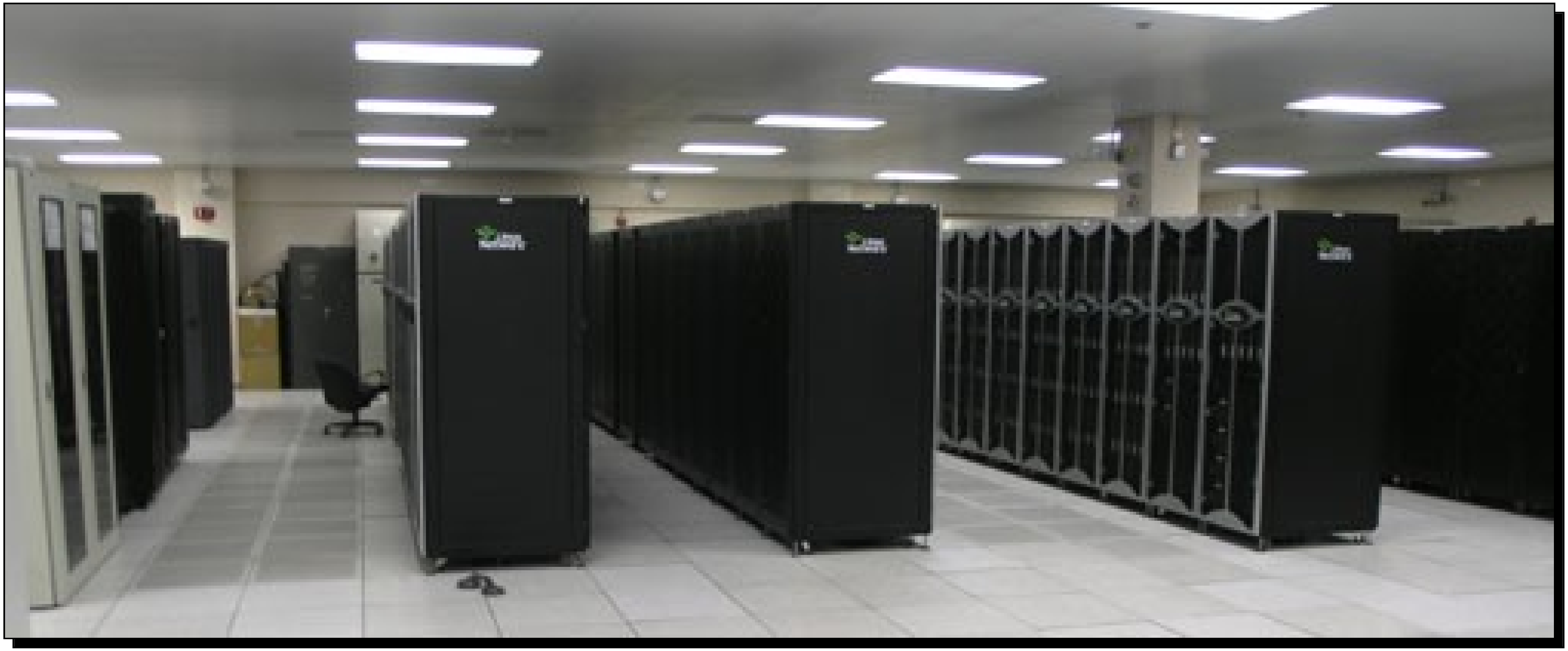
Coyote: all IB, all the time draft

Ron Minnich

Sandia National Labs



Acknowledgments

Andrew White, Bob Tomlinson, Daryl Grunau, Kevin Tegtmeier, Ollie Lo, Latchesar Ionkov, Josh Aune, and many others at LANL and Linux NetworX

Motivation

- I discovered in 2007 that some of our IB software is, ah, not quite as mature as I thought
- “IB-only boot? Solved problem”
- Well, maybe

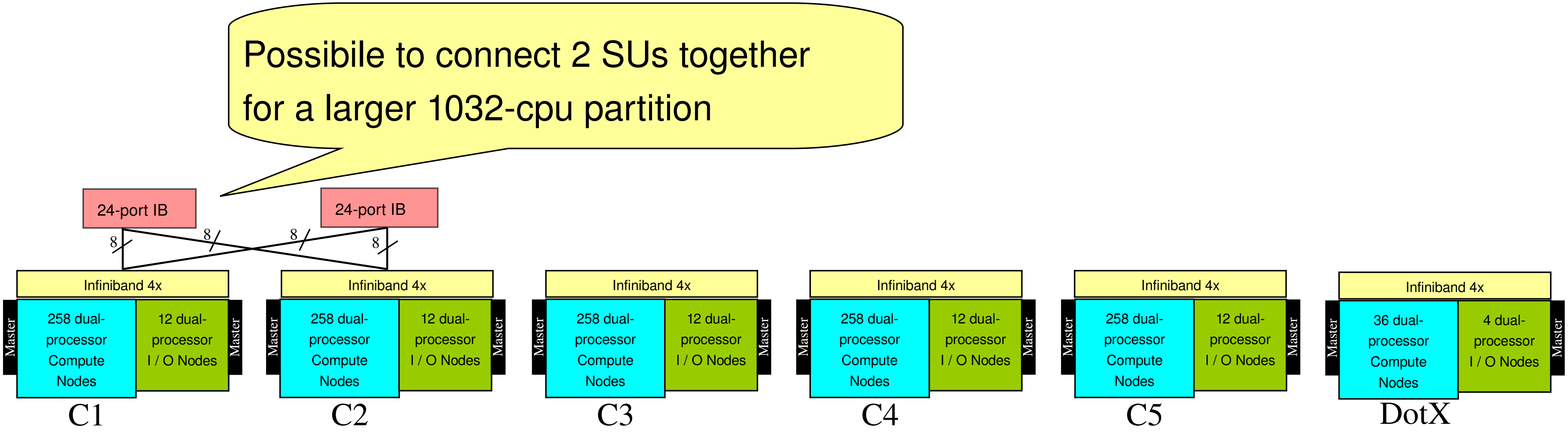
PXE on IB experiences: 2007

- I just tried for SC 07 to set up BluePod cluster to use the PXE-in-firmware on Mellanox cards
- Not surprised, not shocked: required wget this, patch that, things did not quite work
- So, IB has come far, but we're still lacking on the basics
- And I still talk to people who want an “IB only” solution -- and we did this in 2005 at LANL

Overview

- What Coyote is
- The challenge: IB only boot, compute, operate
- How it all fit together
- Challenges and fixes

Coyote



Possibile to connect 2 SUs together
for a larger 1032-cpu partition
24-port IB
24-port IB
8
8
8
8
Infiniband 4x
Master
258 dual-processor Compute Nodes
12 dual-processor I / O Nodes
Master
C1
Infiniband 4x
Master
258 dual-processor Compute Nodes
12 dual-processor I / O Nodes
Master
C2
Infiniband 4x
Master
258 dual-processor Compute Nodes
12 dual-processor I / O Nodes
Master
C3
Infiniband 4x
Master
258 dual-processor Compute Nodes
12 dual-processor I / O Nodes
Master
C4
Infiniband 4x
Master
258 dual-processor Compute Nodes
12 dual-processor I / O Nodes
Master
C5
Infiniband 4x
Master
36 dual-processor Compute Nodes
4 dual-processor I / O Nodes
Master
DotX

- Linux Networx system:
  - 5 Scalable Unit (SU) clusters of 272 nodes + 1 cluster (DotX) of 42 nodes:
  - Dual-2.6GHz AMD Opteron CPUs (single core)
  - 4GB memory / CPU
- 272 node SUs:
  - 258 compute nodes + 1 compute-master
  - 12 I/O nodes + 1 I/O-master
- 42 node DotX:
  - 36 compute nodes + 1 compute-master
  - 4 I/O nodes + 1 I/O-master
- Not pictured: 4 compile & 10 serial job nodes

- System Software
  - 2.6.14 based Linux – Fedora Core 3
  - Clustermatic V4 (BProcV4)
  - OpenMPI
  - LSF – Scheduler
  - PathScale Compilers (also gcc, pgi)
  - Mellanox AuCD 2.0 – OpenSM/Gen2
- System Monitoring
  - Hardware monitoring network (not shown) accessed via third network interface (eth2) on master nodes provides for console and power management via conman and powerman.
  - Environment monitoring via Supermon

Coyote boot software (beoboot)

- This software can support any cluster system
- i.e., on top of this:
- can build Rocks, Oscar, OneSIS, Tripod, etc.
  - This software is *not* bproc or Clustermatic specific
- It *is* (in my experience) the fastest, most scalable boot system
- Because it uses Linux to perform the boot, not PXE or similar systems

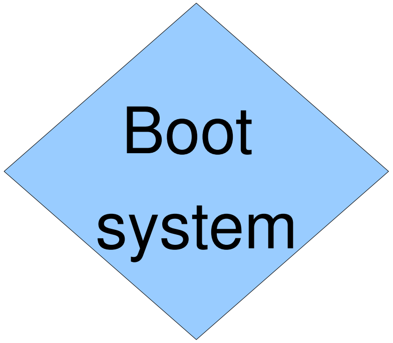
The Challenge: IB only compute, boot, operate

- Early goal was to build Coyote with one, not two, networks
- Experience on Pink and Blue Steel with Ether
  - Pink: Ethernet not needed, greatly reduced cost
  - Pink: Motherboard issues with Ethernet on IO nodes delayed delivery
  - Blue Steel: Ethernet *was* needed, greatly increased headaches

Digression: A note on failure models

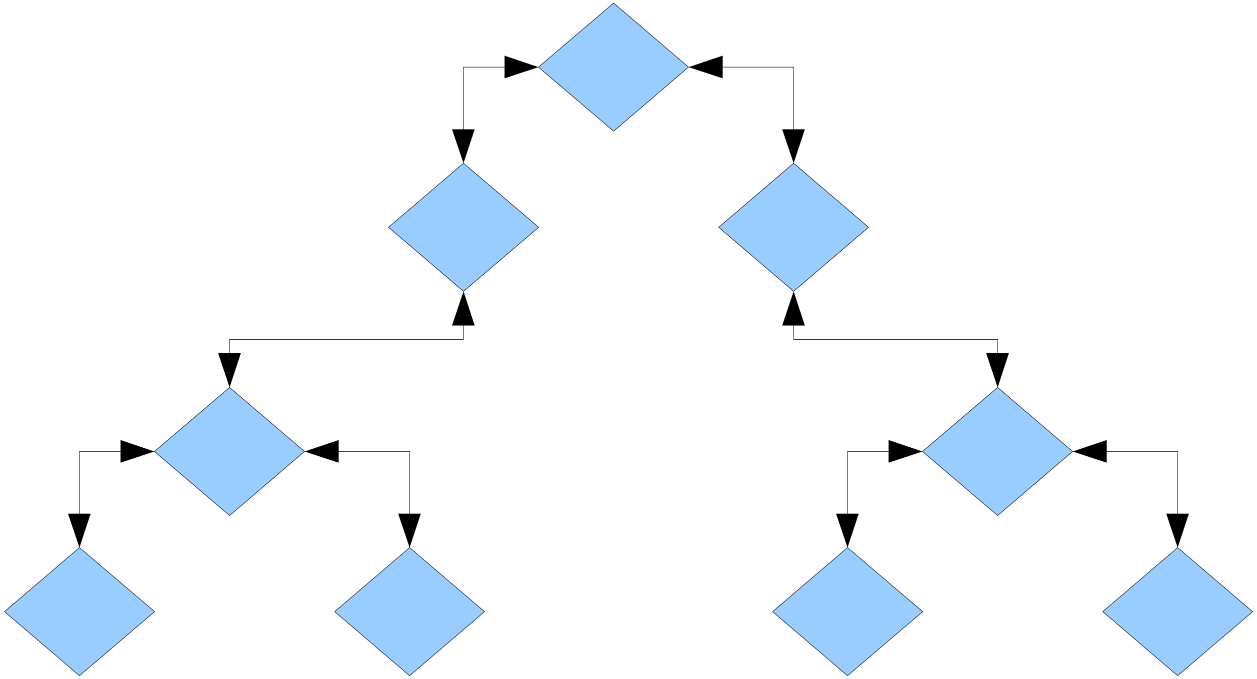
- It is odd to this day to see that the concept of points-of-failure is misunderstood
- People *do* understand a single point of failure
- People *don't always* understand that *multiple points of failure* is not the same as *no single point of failure*
- This confusion leads to strange design decisions

Example: boot management



- Here is a boot system for a 1024-node cluster
- “But it's a Single Point Of Failure”
- So people frequently do this:

Example: boot management: hierarchy of tftp servers



- What happens if one node goes out?
- Answer determines if this is MPOF
- In most cases, it is: you lose some nodes

Coyote software components Firmware (i.e. in BIOS/CF)

- LinuxBIOS
- Linux kernel with:
  - IB Gold stack, IPoIB
  - beoboot
  - kexec
- These components were sufficient to provide a high performance, scalable, ad-hoc boot infrastructure for Coyote

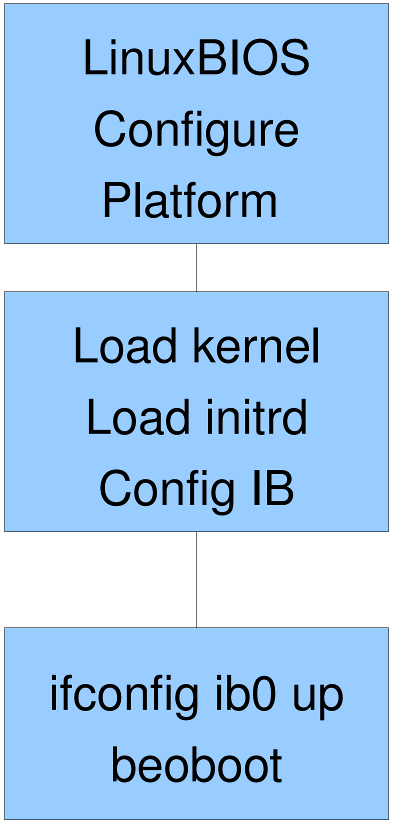
Note: Kernel was in Compact Flash

- In many cases we can put LinuxBIOS + Linux in BIOS flash
  - (see: http://tinyurl.com/2umm66) Linux + X11 BIOS!
- Once we add myrinet or IB drivers, standard FLASH parts are too small (only 1 MB)
- Long term goal:Linux back in BIOS FLASH
  - Else have to fall back to Ether + PXE!
- Newer boards will have 4 MB and up parts

Coyote master node

- This node controls the cluster
- It is contacted by the individual compute/IO nodes for boot management
- Provides a Single Point Of Failure model with *ad-hoc tree* boot system (more on that later)
- Fastest way to boot; far faster than PXE

Coyote boot process

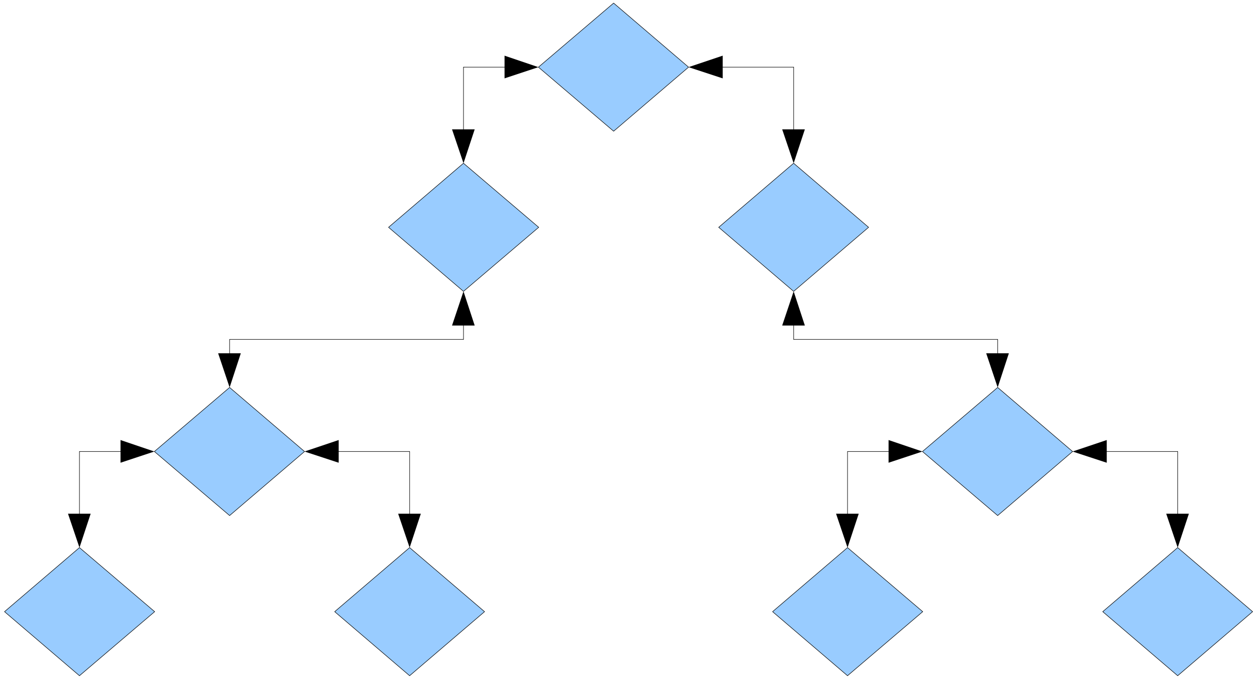


- LinuxBIOS is not required, just a good idea
- Initrd contains drivers
- At this point, modprobe+ifconfig worked fine (thanks!)
- Thanks to Hal for DHCP that worked

Why not just use PXE at this point?

- The problem: PXE is slow and unsophisticated
- Requires network card firmware to make the card act like an ethernet
- Simple-minded, slow, programmed-IO device model
- In practice, we have booted 1024 node clusters with Linux in the time it takes PXE to *not* configure one network interface

PXE inefficiencies lead to construction of unreliable boot setup

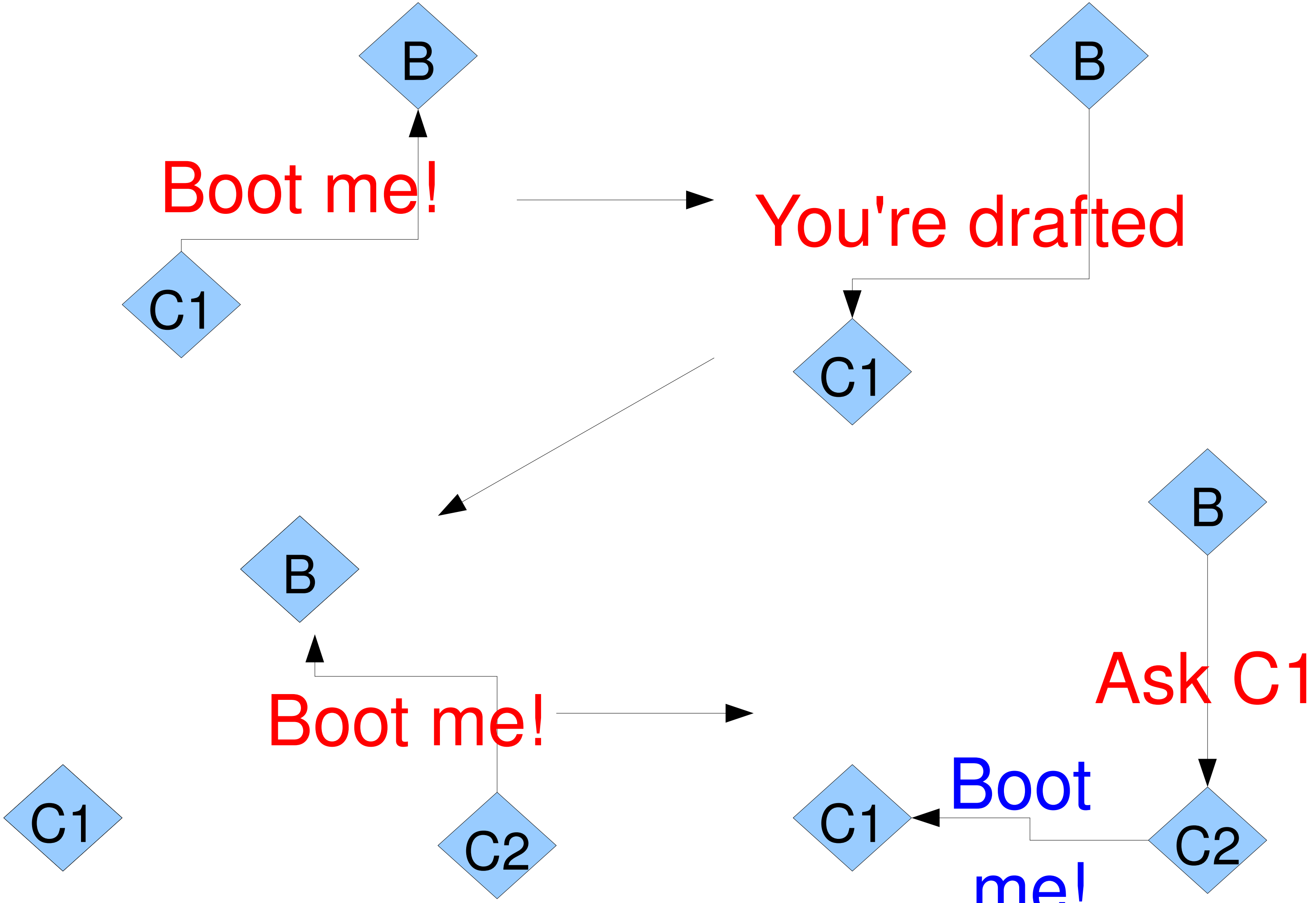


- Our old friend, MPOF, we meet again

The right way to boot

- Use the strengths of the HPC network and Linux
- We'd been doing this at LANL since 2000, and understood it pretty well
- The idea is simple: conscript the booting nodes to help boot other nodes
- That's the beoboot component

Booting fast and reliably



Ad-hoc tree boot

- In practice, this is incredibly fast
- Image distribution: 1024 nodes, < 10 seconds
- Most boot time: Linux serial output
- Extraordinarily reliable
  - Tested, fast Linux drivers
  - Not slow, buggy PXE drivers
  - Who wants 10Gb IB to emulate 10 Mb ethernet?

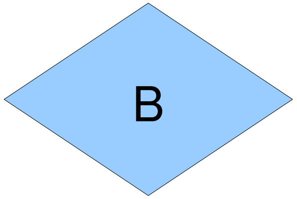
Next steps

- Beoboot used special protocol
- We have moved to 9p-based system called xbootfs

9P?

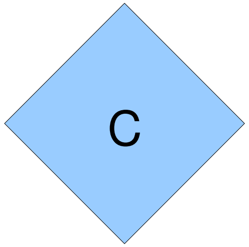
- New (to Linux) file system protocol
- Extremely light weight, easy to program
- Can be mounted via Linux 9p file system
- Created a new program called xbootfs
- Testing on clusters at LANL, SNL, partners

xbootfs



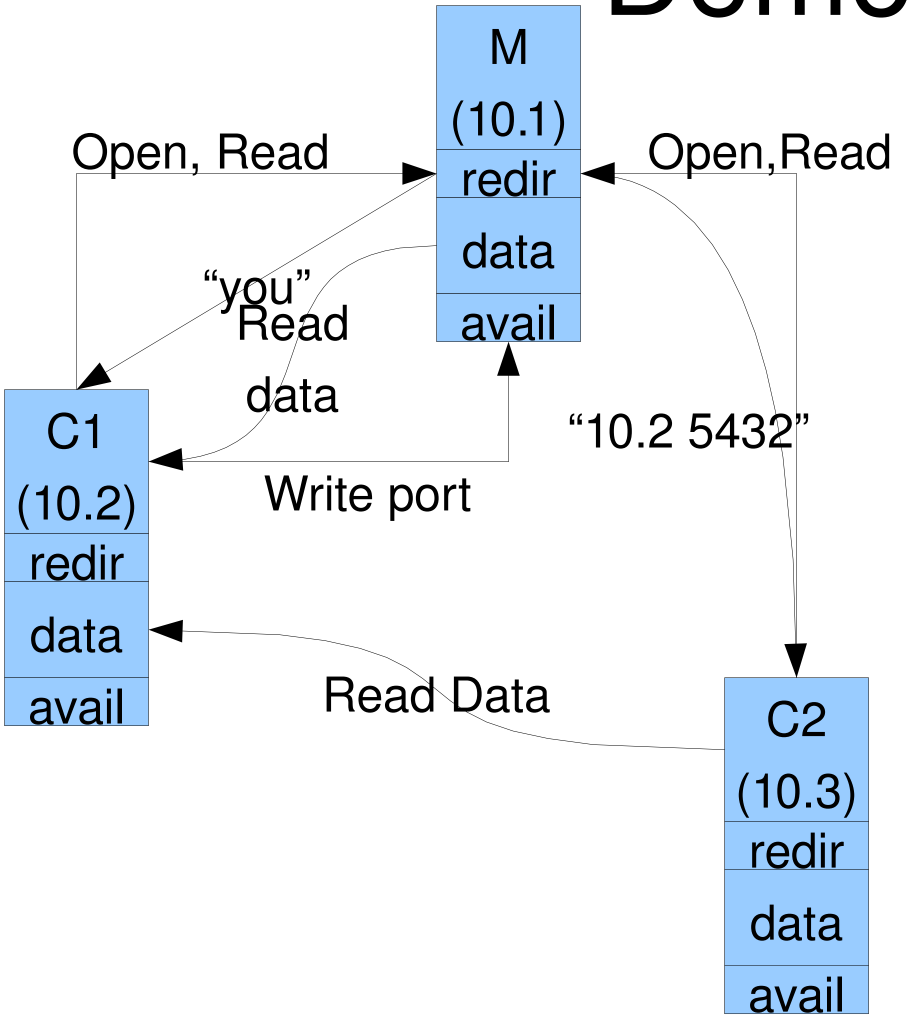
serves “files”:

data, avail, redir



- Client: Open redir
- Read “IP”, or “me”, or “you”
- If “me”, open “data”, read, done
- If “you”, also become server
- if “IP”, go to other client for data

Demo ...


M
(10.1)
redir
data
avail
Open, Read
Open,Read
“you”
Read
data
C1
(10.2)
redir
data
avail
Write port
“10.2 5432”
Read Data
C2
(10.3)
redir
data
avail

Once a client becomes a server

- Other clients are redirected to it
- Clean, easy recursion
- In tests, it's fast and easy to modify

Conclusions

- IB-only systems are best built with Linux “firmware”
- Ad-hoc trees use HPC network for booting, eliminate slow, failure-prone static trees
- Have been working since 2005
- Our next-gen software builds on 9p protocol (working on *BSD, Linux, MACOS, ...)