XCPU

A New, 9P-based Framework for Cluster Management

latchesar Ionkov
Andrey Mirtchovski
Los Alamos National Laboratory

{lionkov,andrey}@lanl.gov

Contents

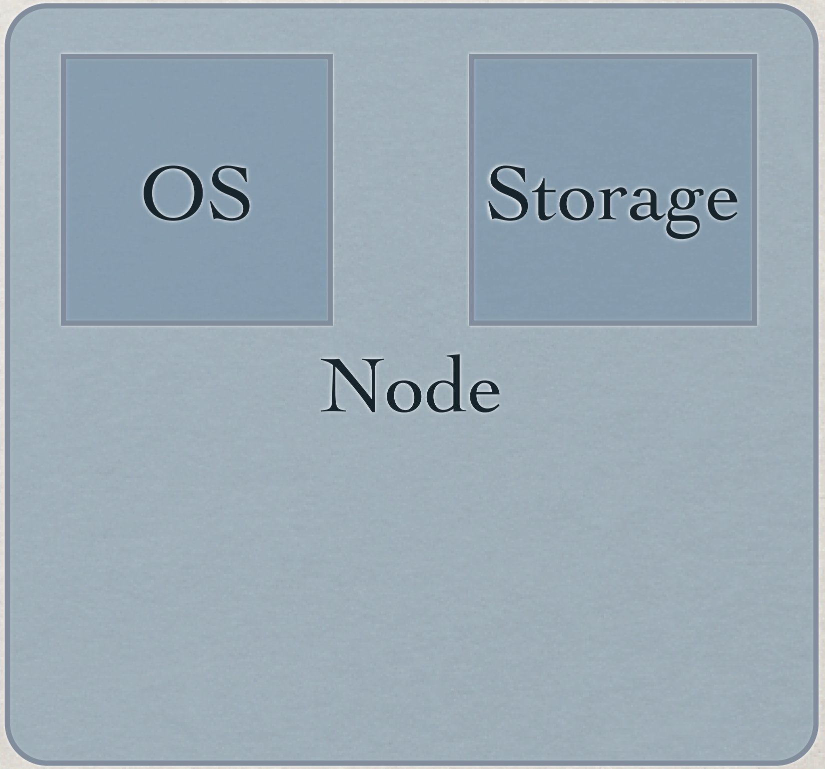
- Introduction
- A brief discussion of clustering
- XCPU:
  - Goals
  - Design
  - Implementation
  - Performance
- Examples

Introduction

- HPC: an inseparable part of scientific progress
- A recent design at LANL was deemed "computationally light" because it used only 1% of LANL's computing capability during the past two years
- Top 500: 72% clusters (vs 0% ten years ago)
- So, what are clusters?

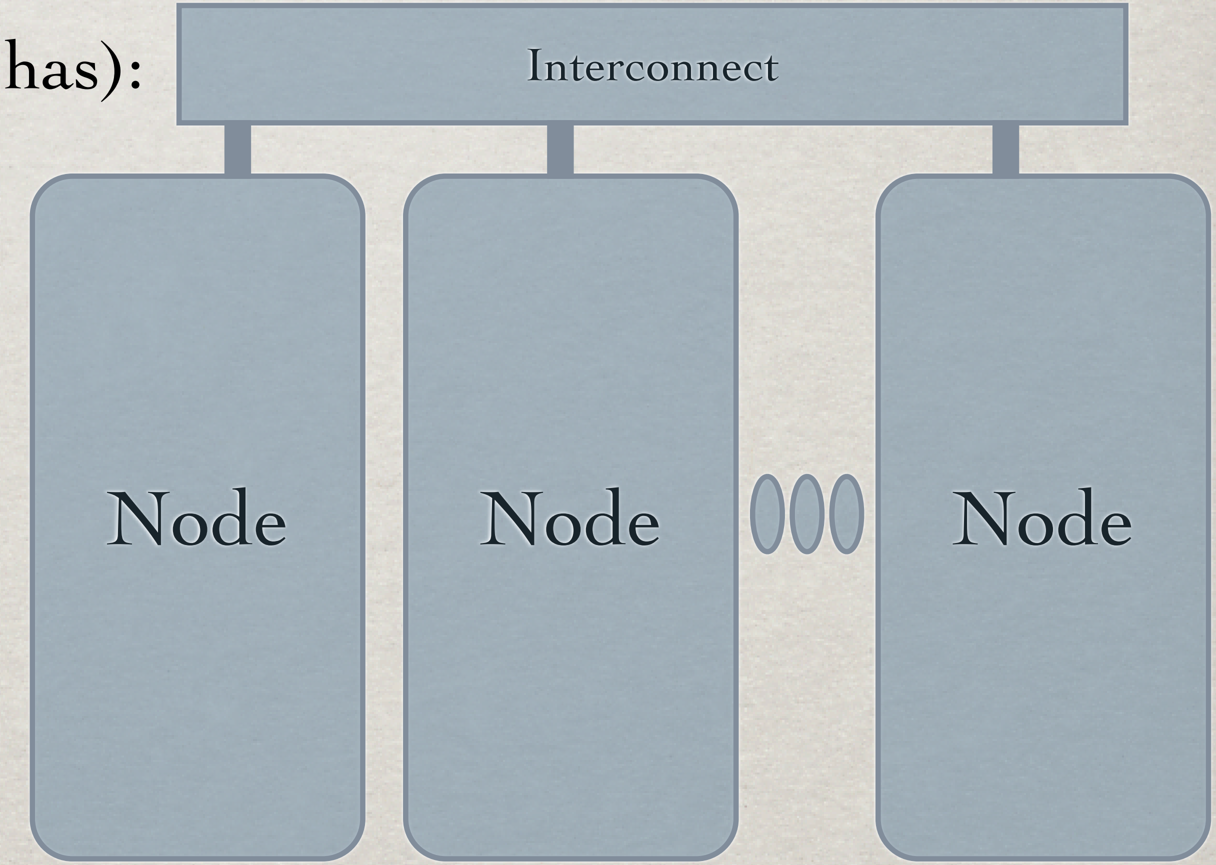
A Brief History of Clusters (sort of)

- A Single Node has:
  - OS
  - Storage
  - Daemons
- Noise



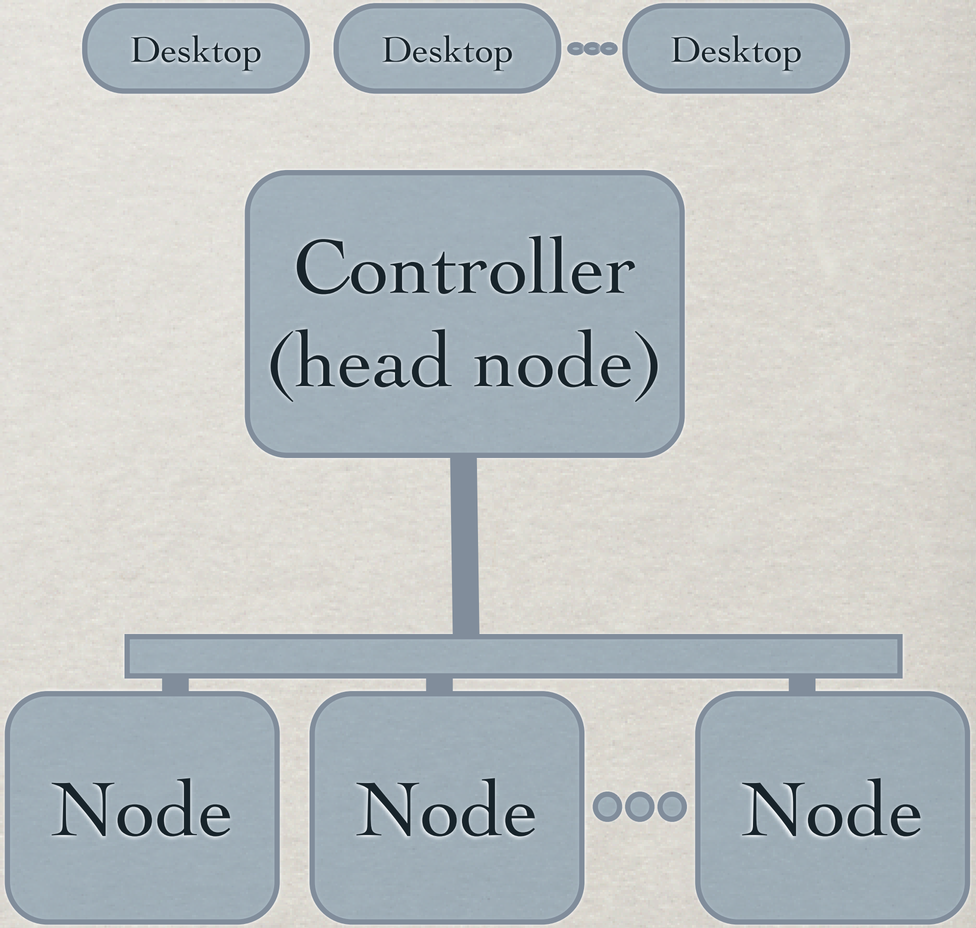
ABHOC (CONT'D)

- A Set of Nodes (usually has):
  - Identical OS
  - Network
  - FS
  - More Noise



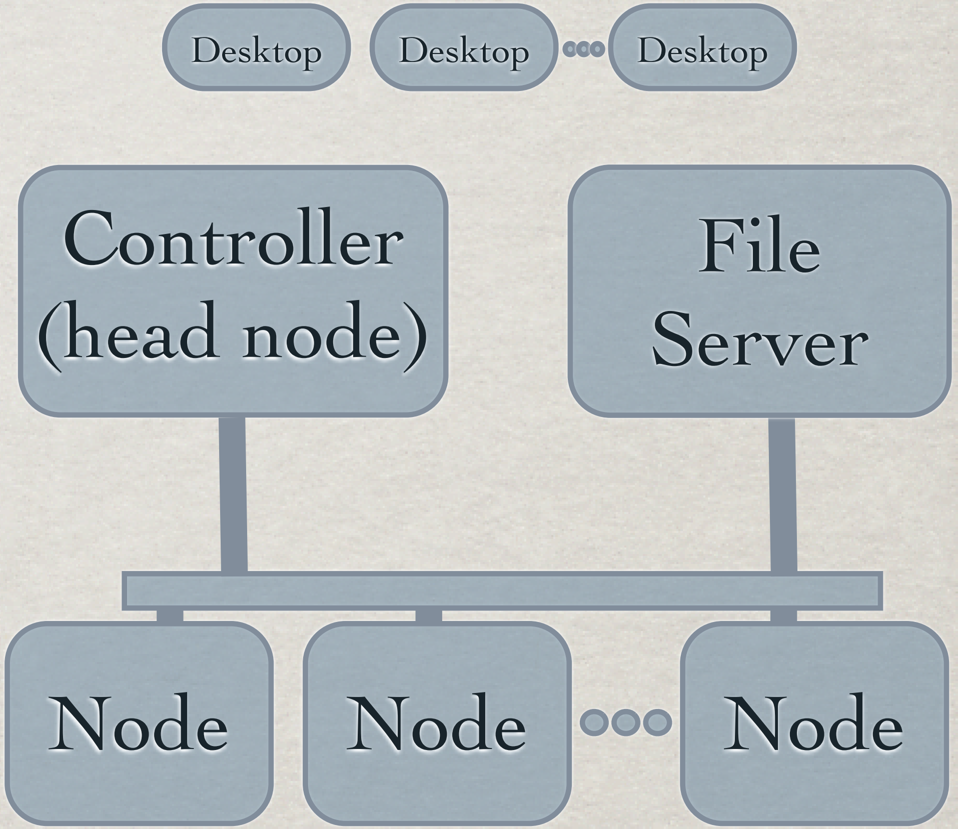
The Head Node

- The Head Node
- Usually same OS
- Usually same network
- Allows connections from remote machines (desktops)
- Has all necessary information about the cluster



The File Server

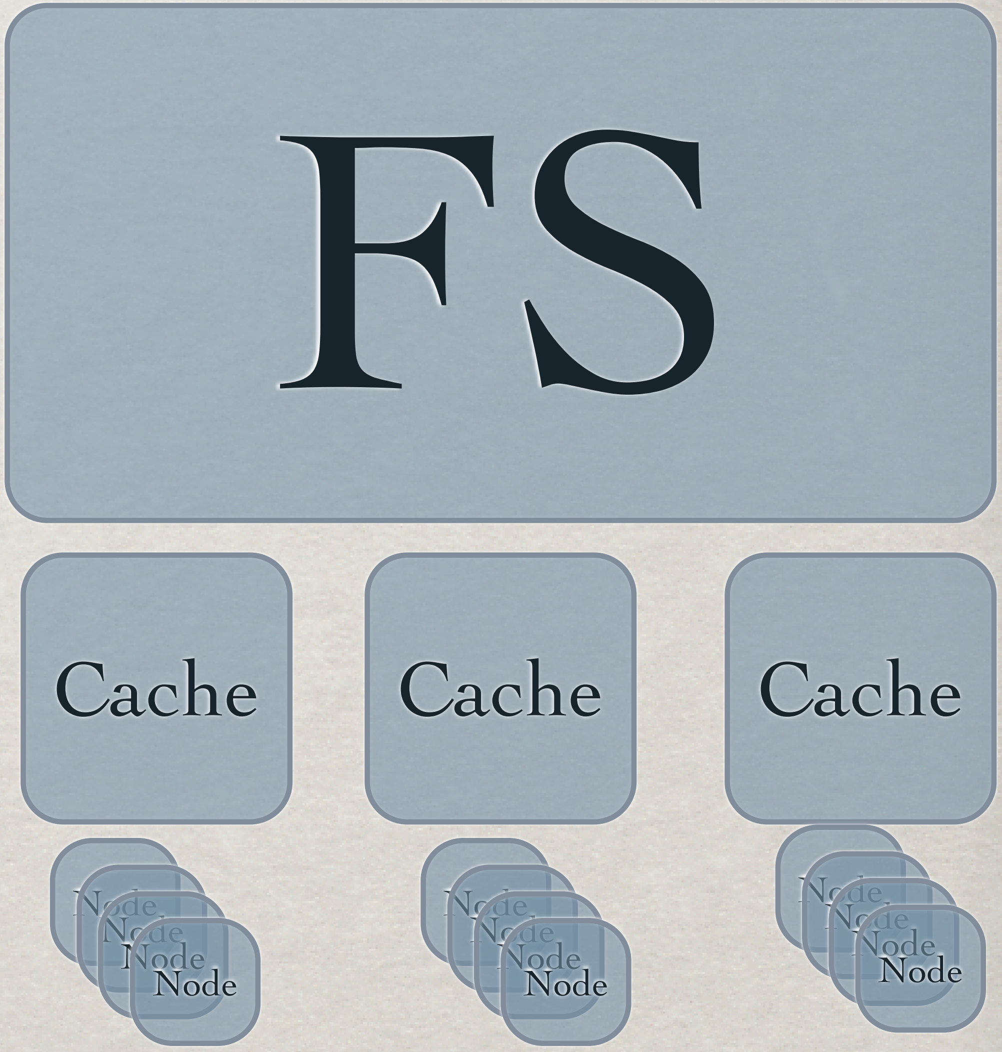
- Massive amounts of storage
- Somehow must be delivered on-demand to the end nodes
- Scalable?



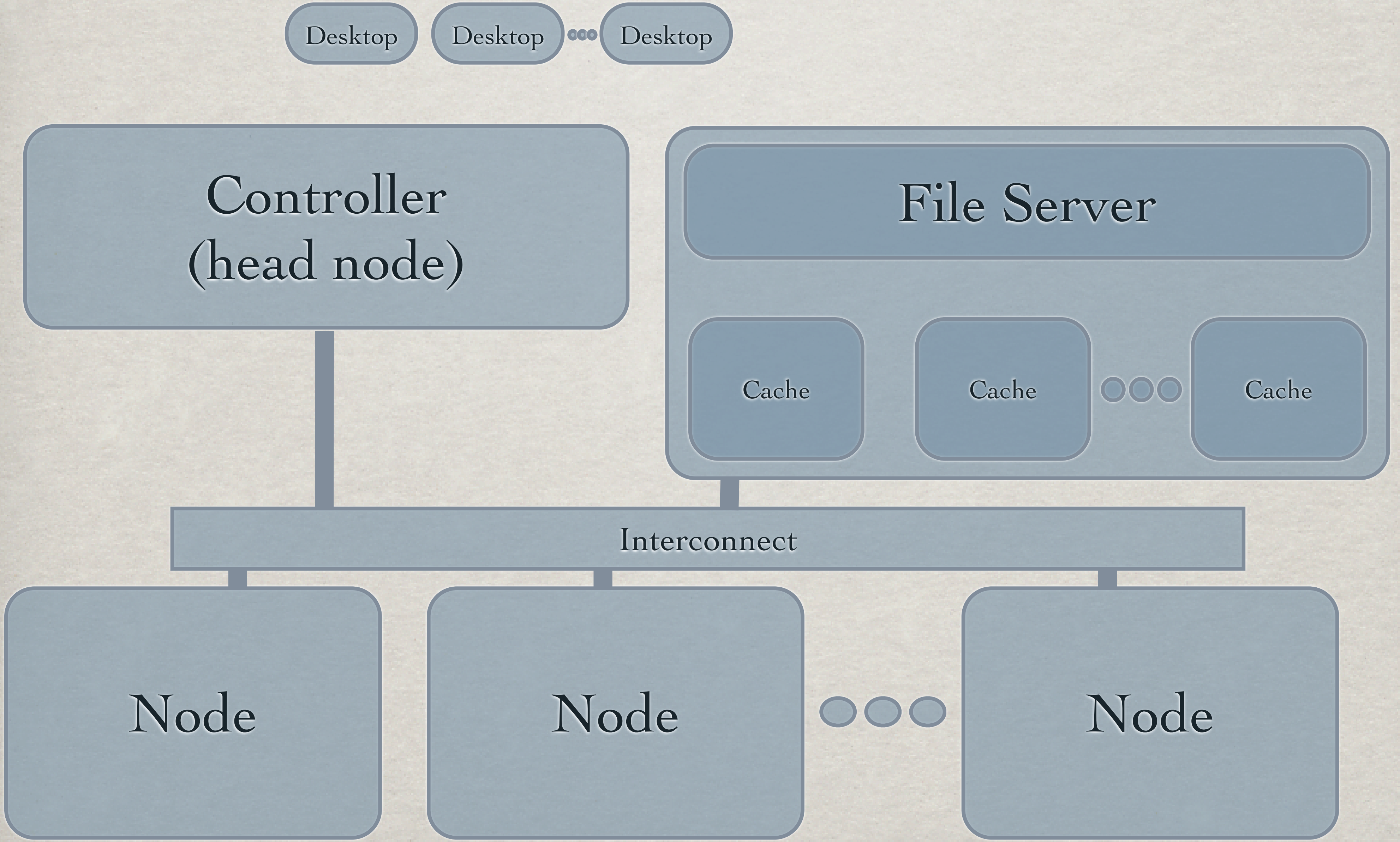
The File Server (in detail)

- Use caching to distribute the data

- But what about writes?

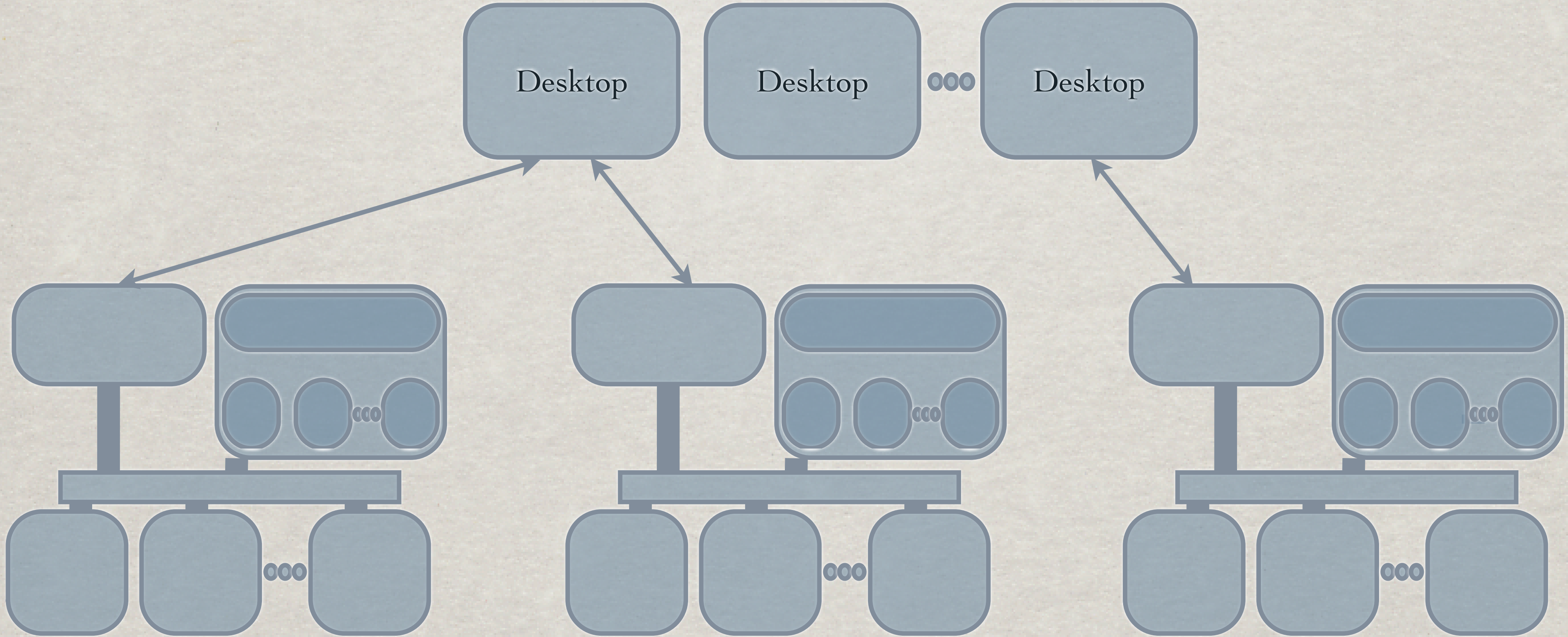


A Cluster:


Desktop
Desktop
Desktop
Controller
(head node)
File Server
Cache
Cache
Cache
Interconnect
Node
Node
Node

And Finally: Sets Of Clusters...

- “Billions and Billions”
- LANL has at least 5 operating at the same time



Now To Drive The Whole Thing

- Scheduler
- Job Starter
- Accounting
- Authentication
- Resource Discovery
- ...

Problems:

- Speed
- Speed
- Speed
- How high can we score on the Top 500?
- Factors which impact performance:
    - Hardware
    - Software

Problems (cont'd):

- 10 years ago there were no clusters in the Top 500 list
- 5 years ago 70% of the machines (including clusters, MPP and constellations) had fewer than 256 processors
- Now: 91% of the Top 500 list have 512 or more processors
- How fast has software moved in the past 5 years?

What We've Seen

- There is room for improvement on the software side of things
- Simple systems ultimately perform better than more complex ones (and are easier to administer)
- If it works well people will keep using it (provided it performs well)
- Simplicity: not necessarily the number of elements involved, but how they interact

Enter XCPU

- A novel cluster management system
- Designed with simplicity as the underlying paradigm
- Aims to replace a very successful cluster framework: B-Proc
- Aims to extend beyond the single system image to clusters of arbitrary configurations

Goals

- Scalability: thousands of nodes

- Heterogeneity: OS-independent, hardware-independent

- Flexibility: no restriction of the form and design of the cluster

- Performance: b-proc is the fastest system we know. XCPU should match it within a factor of five (16mb image over 1024 nodes in < 20 seconds)

Goals (cont'd)

- No head nodes
- Disconnected operation
- Ability to resume sessions

- Starting point:
- What type of resource are we most successful in sharing today?

Design

- Split in Two: Clients and Servers
- Servers serve (synthetic or real) files
- Clients use standard file operations to access them
- Mounted or directly connected to over a/any network

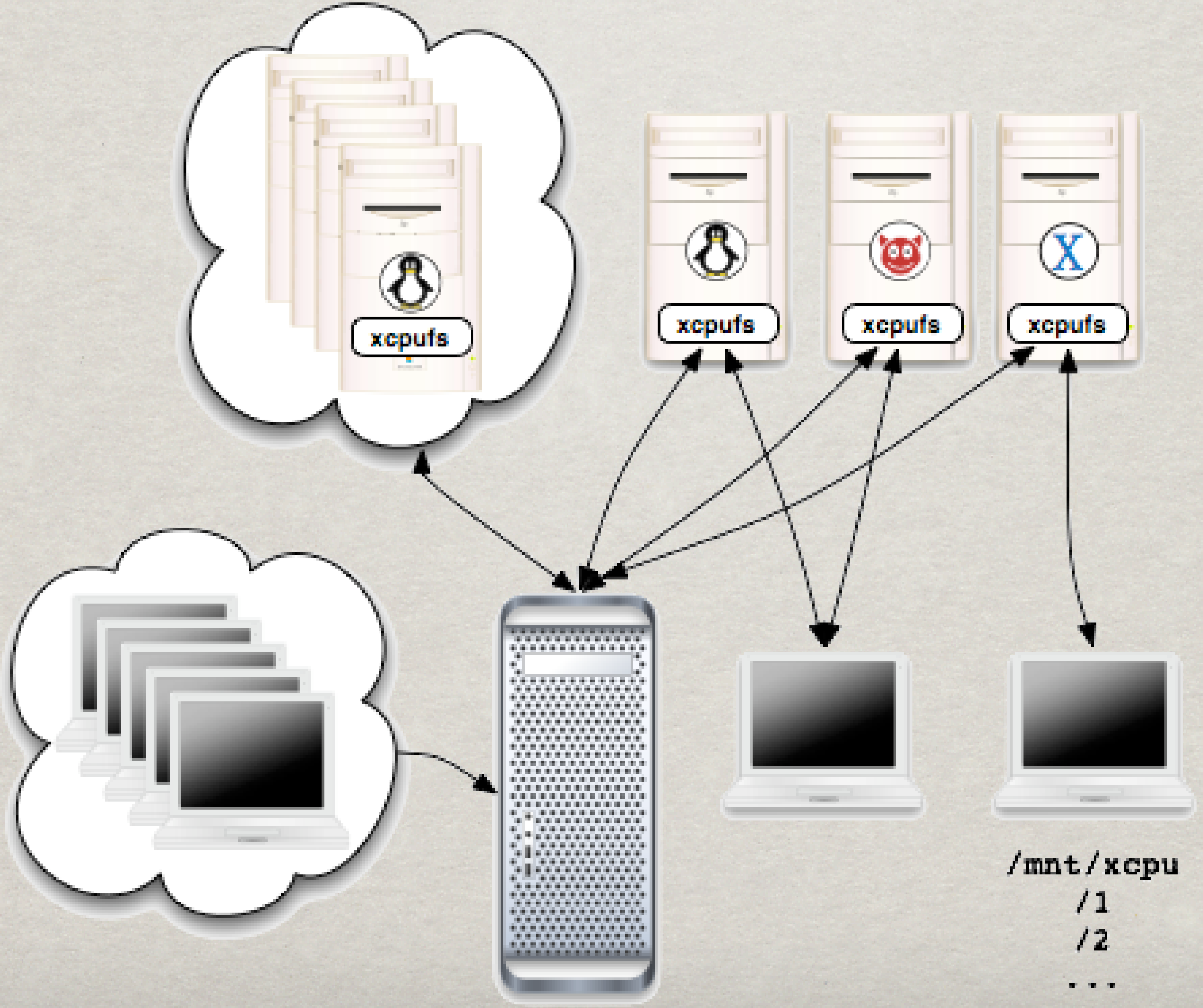
Servers

- Provide a location to store binaries and input files
- Control application execution (start/stop/checkpoint)
- Federate input/output from/to clients
- Able to act as clients when tree-spawn execution is required
- Mountable (via v9fs) by any machine with permissions to do so
- Speed: 16MB binary copied and executed to 1024 nodes in 3 seconds (our current best is 6)

Clients

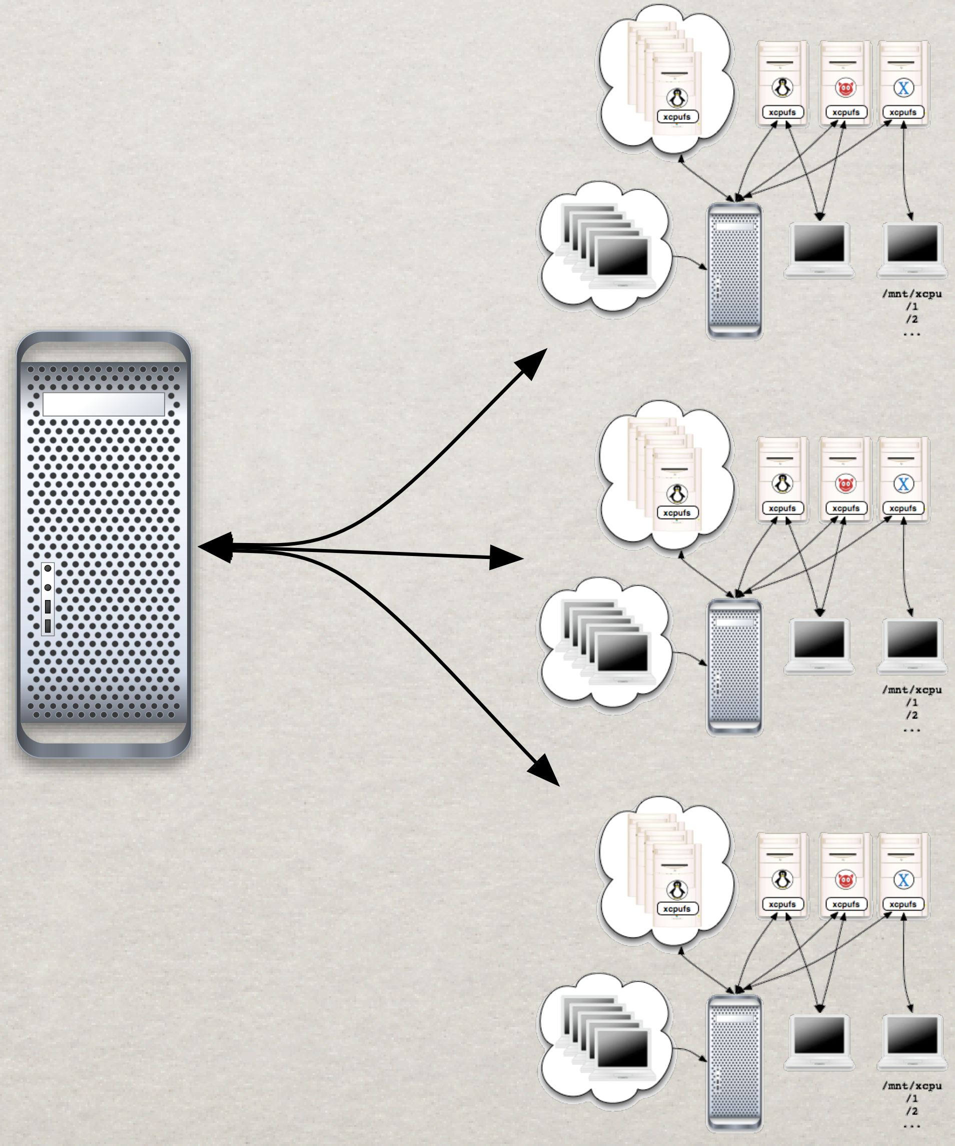
- Connect to one or more servers
- Create sessions
- Copy binary/input files/arguments
- Locate and copy additional libraries if necessary
- Federate input/output to/from servers
- Unexpected bonus: allow pipes to be executed across clusters!
  - `#!/bin/bash\nexec $*\nexit 1\n`
  - `xrx -a tar zxf - < somefile.tgz`

The XCPU Environment



The XCPU environment (cont'd)

```
/mnt/xcpu/
  cluster1/
    node1/
      session1/
  cluster2/
    node1/
      session1/
      ...
    node2/
    ...
  cluster3/
    node1/
      session1/
```



File Hierarchy

- Top Level:
  - `arch`
  - `clone`
  - `env`
  - `procs`
  - `state`
  - `auth`

File Hierarchy

- Session Directory
  - `argv`
  - `ctl`
  - `exec`
  - `env`
  - `fs`
  - `state`
  - `stdin`
  - `stdout`
  - `stderr`
  - `stdio`
  - `wait`
  - `id`

Example

```
$ mount -t 9p 192.168.100.101 /mnt/xcpu/1 -o port=6666
$ cd /mnt/xcpu/1
$ ls -l
-r--r--r-- 1 root root   0 Jul 25 10:19 arch
-r--r--r-- 1 root root   0 Jul 25 10:19 clone
-rw-r--r-- 1 root root   0 Jul 25 10:19 env
-r--r--r-- 1 root root   0 Jul 25 10:19 procs
-r--r--r-- 1 root root   0 Jul 25 10:19 state
$ tail -f clone &
1234
$ ls -ld 1234
-r--r--r-- 1 andrey root   0 Jul 25 10:19 1234
$ cd 1234
$ ls -l
-rw-rw---- 1 andrey root 0 Jul 25 12:58 argv
-rw-rw---- 1 andrey root 0 Jul 25 12:58 ctl
-rw-rw---- 1 andrey root 0 Jul 25 12:58 env
drwx------ 1 andrey root 0 Jul 25 12:58 fs
-r--r--r-- 1 andrey root 0 Jul 25 12:58 stderr
-rw-rw---- 1 andrey root 0 Jul 25 12:58 stdin
-rw-rw---- 1 andrey root 0 Jul 25 12:58 stdio
-r--r--r-- 1 andrey root 0 Jul 25 12:58 stdout
-rw-rw---- 1 andrey root 0 Jul 25 12:58 wait
$ cp /bin/date fs
$ echo exec date > ctl
$ cat stdout
Tue Jul 25 12:59:11 MDT 2006
$
```

Security

- Public/Private Key
- Identity vs TLS
- The Lamentable Introduction of an Administrative Account

Monitoring: Statfs

- Another file server
- Also a client
- Pings XCPU nodes periodically (with an adjustable frequency)
- Used by clients when they want to execute a job on all nodes without having to know where they are
- Basic FIFO scheduling

Scheduling

- We don't want to do scheduling, there are many other systems that can do it for us much better
- Maui/Torque integration
- LSF (?)
- PBS
- Scheduling across administrative domains?

Implementation

- OS Independent
- Language Independent
- Current implementation written in C using standard, POSIX-compliant code (no GNU-isms)

Plan 9 & 9P

- “Everything is a file”
  - network (/tcp)
- Source of our protocol: 9P
- Robust
- Portable
- Works over all kinds of connections (tcp/rudp/ib/cell’s dma)
- Scalable

9P

Version	Auth
Error	Flush
Attach	Walk
Open	Create
Read	Write
Clunk	Remove
Stat	Wstat

Code

- ~20k SLOC
- Includes all libraries + client, server and monitoring code
- Libraries allow new file servers and clients to be created very easy (100 lines of code gives you a fully functional mountable client)

Portability

- Anything with a socket :)
- Linux
- *BSD
- Darwin
- Most if not all portability issues arise from different representations of system resources /proc is the best example

Future

- Interface to debuggers?
- Fully integrated resource discovery?
- Monitoring and control
- Resilience?

Thank You!

http://xcpu.org