011101000110010101110010011000010111001101100011011000010110110001100101

Mass Storage on the Terascale Computing System

Nathan Stone

Pittsburgh Supercomputing Center

April 18, 2001

011101000110010101110010011000010111001101100011011000010110110001100101

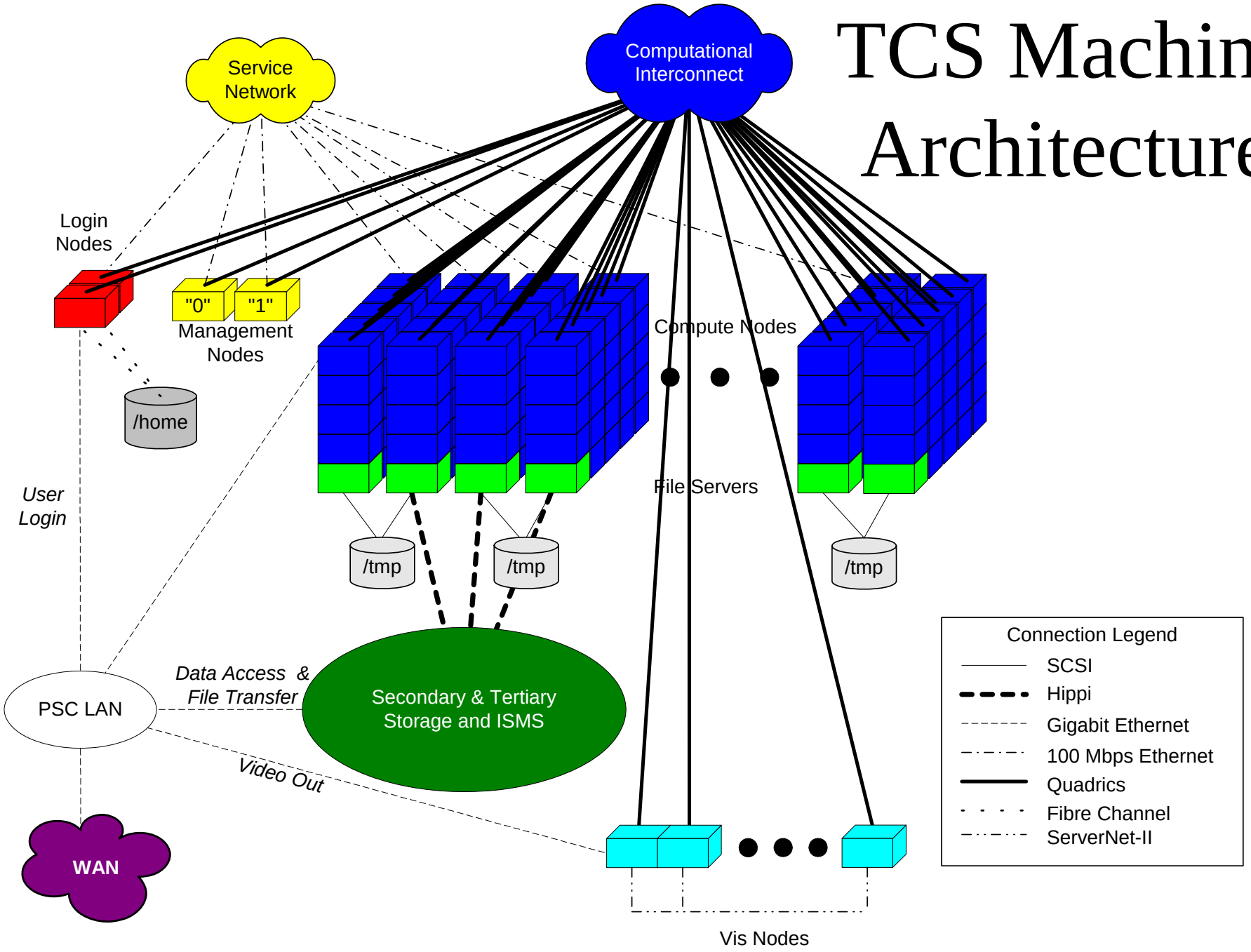
Overview

- General Architecture
- Resource Management
- Checkpoint System
- Storage Architecture

Terascale Machine Architecture

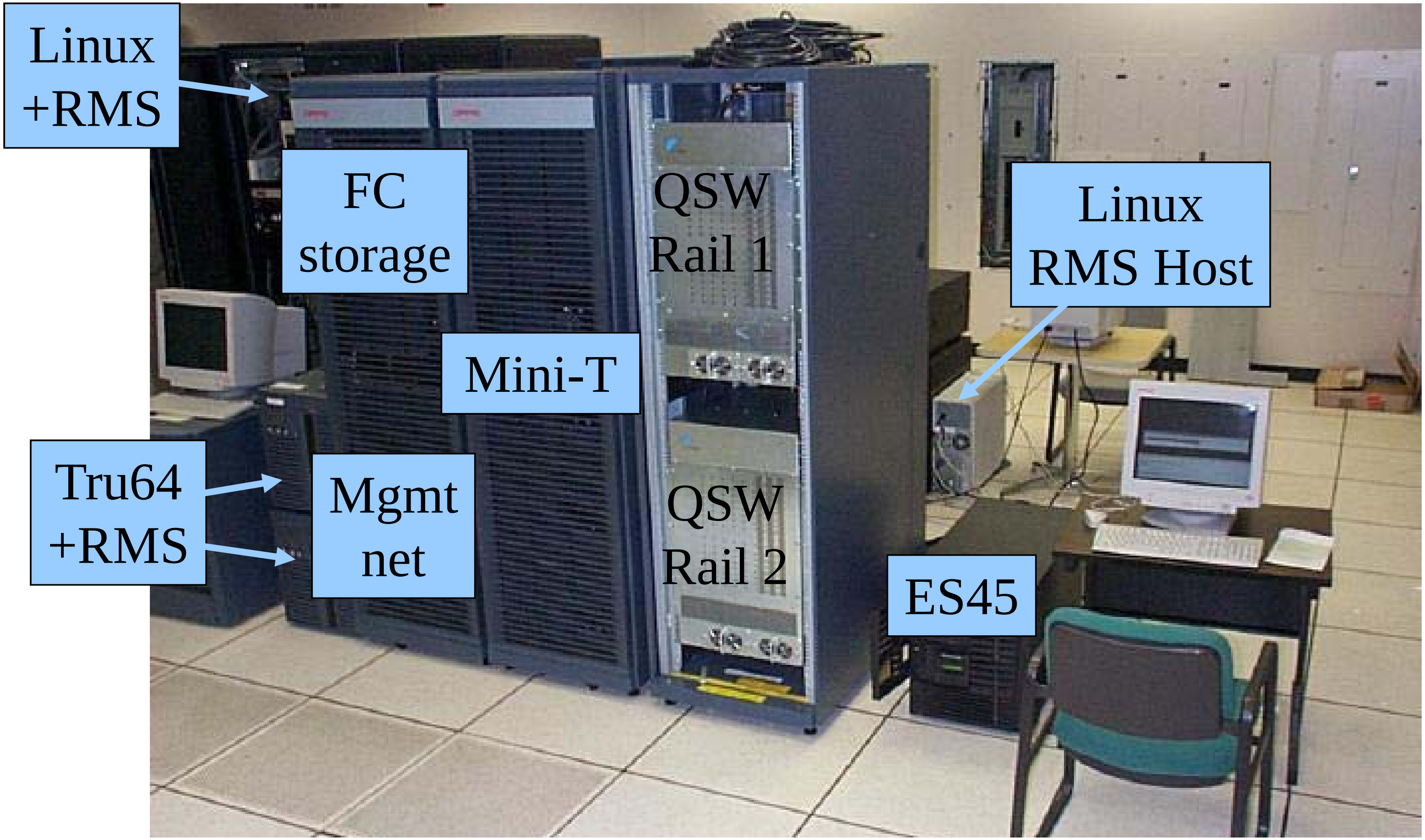
- Compaq AlphaServer SC system
- 6 Tflop peak computational power
- 750+ quad EV68 nodes
- Dual-rail Quadrics interconnect
- Partitioned into separate Login/Compute/FileServing/Vis domains
- Hierarchical Storage Manager

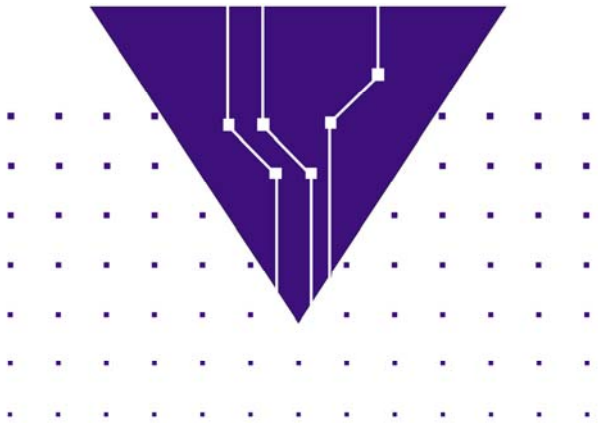
TCS Machine Architecture



Initial System Deployment

AlphaServer SC Testing



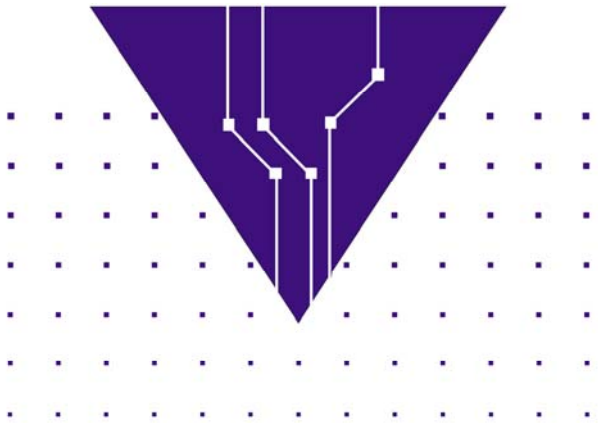


Resource Management

01110100011001010111001001100001011100110110001101100001011011000110101

Resource Management

- Utilizing Portable Batch Scheduler (PBS)
- Custom port of PBS to Tru64/AlphaServer SC
  - integrated with the Resource Management System (RMS) → job control, data collection
  - exploits the event-handling system
- Currently running 3 queues:
  - Day: FIFO, allows interactive allocations
  - Night: favors large jobs
    - Nightly “drain” at 8:00pm
  - Priority: special dispensation only



Checkpoint System

01110100011001010111001001100001011100110110001101100001011011000110101

Checkpoint Motivation

Primary purposes:

1. To enable job recoverability following machine failure
2. To optimize resource management associated with checkpointing
3. To maximize machine utilization

Checkpoint Features-1

The PSC TCS checkpoint library:

- Automates restart for failed jobs via PBS (IFF…)
- Supports restarting on different nodes
- Facilitates access to tuned storage space
- Manages the resources associated with active jobs
  - *e.g.* DB and FS
- Runs IO tasks concurrent with compute jobs
  - For certain IO “plans”

Checkpoint Features-2

- Presents a simple, fixed API to the user
  - configurable and expandable functionality
- Accounts for the loss of compute time
- Maintains the security of user data
- Scales to large node count
- Allows users to keep the checkpoint files
  - offline analysis

Checkpoint Implementation-1

The TCS Checkpoint library:

- Supports multiple recovery “plans”
  - Duplication
  - XOR-based parity (across nodes)
  - Redirection to centralized RAID
- Can regenerate lost checkpoint files
- Has a single API for C/C++/FORTRAN
- Stores status information centrally in the RMS DB
- Is configurable on-the-fly via /etc/tcsiod.conf

Checkpoint Implementation-2

- Requires a single daemon on each node
  - ➢Root privileges and local FS access
- IPC via a simple, extensible socket protocol
- Calculates "lost" compute time
  - = time from the last complete checkpoint to recovery
  - ➢Refunds?
- Anticipates the arrival of alternative data-migration protocols

Checkpoint User Interface

Initialize	int tcs_init(int *rank);
Is this job being restarted?	int tcs_isRestart();

Open File (reading/writing)	int tcs_open_write(char *prefix); // return LUN
	int tcs_open_read(char *prefix); // “ “

Close File	int tcs_close(int *lun, int *keep);
Read Data	int tcs_read(int *lun, void *A, int *len, int *datasize);
Write Data	int tcs_write(int *lun, void *A, int *len, uint *datasize);
Finish	int tcs_finalize();

Sample User Source Code

Initialize

Recover from failed Job

Calculate

Write Checkpoint File

Finish

```
tcs_init(rank)

if (tcs_isRestart()){
  lun = tcs_open_read("myfile")
  tcs_read(lun, array, arrayLen, elementSize)
  tcs_close(lun, keep)
}

while (do_more_calculations) {
  // calculate here…
  lun = tcs_open_write("myfile")
  tcs_write(lun, array, arrayLen, elementSize)
  tcs_close(lun, keep)
}

tcs_finalize()
```


0111010001100101011100100110000101110011011000110110000101101100011001010111001001100001011100110110001101100001011011000110010101110010011000010111001101100011011000010110110001100101

Sample User Run Script

Select Drain Plan

Set size of XOR set

Run Job

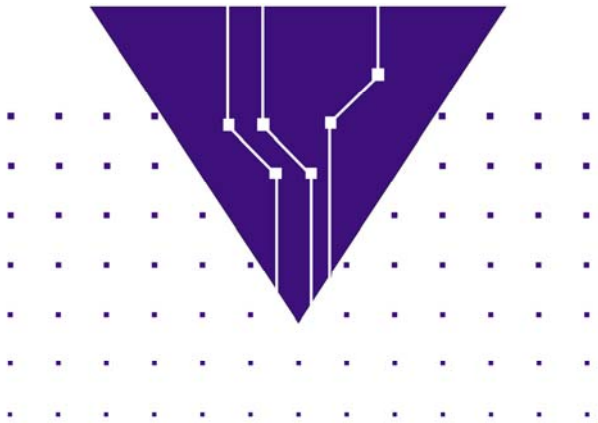
```
#!/bin/csh

# Select Drain Plan: (A1, A2, B1, B2, C1, C2 – default is C1)
setenv TCS_PLAN C1

# Set number of nodes per XOR set (default is 8)
setenv TCS_NODES_XOR 4

prun -n ${RMS_PROCS} -N ${RMS_NODES} ./a.out
```

Checkpoint Installation

- Install include/, bin/, lib/, init.d/, etc. ∀ FS
  - ➢ handled @ PSC by “depot”
- Create /etc/tcsiod.conf ∀ FS
- Execute “bin/tcsconfigchk” ∀ nodes
- Execute “init.d/tcsiod start” ∀ nodes

Checkpoint Future Work

- Internal timing/performance logging
- Native ELAN communication for data migration w/o prun
  - both homogenous and heterogeneous clusters
- IO “redirection”
  - output goes directly to File Servers for remote writing (bypass local disk)



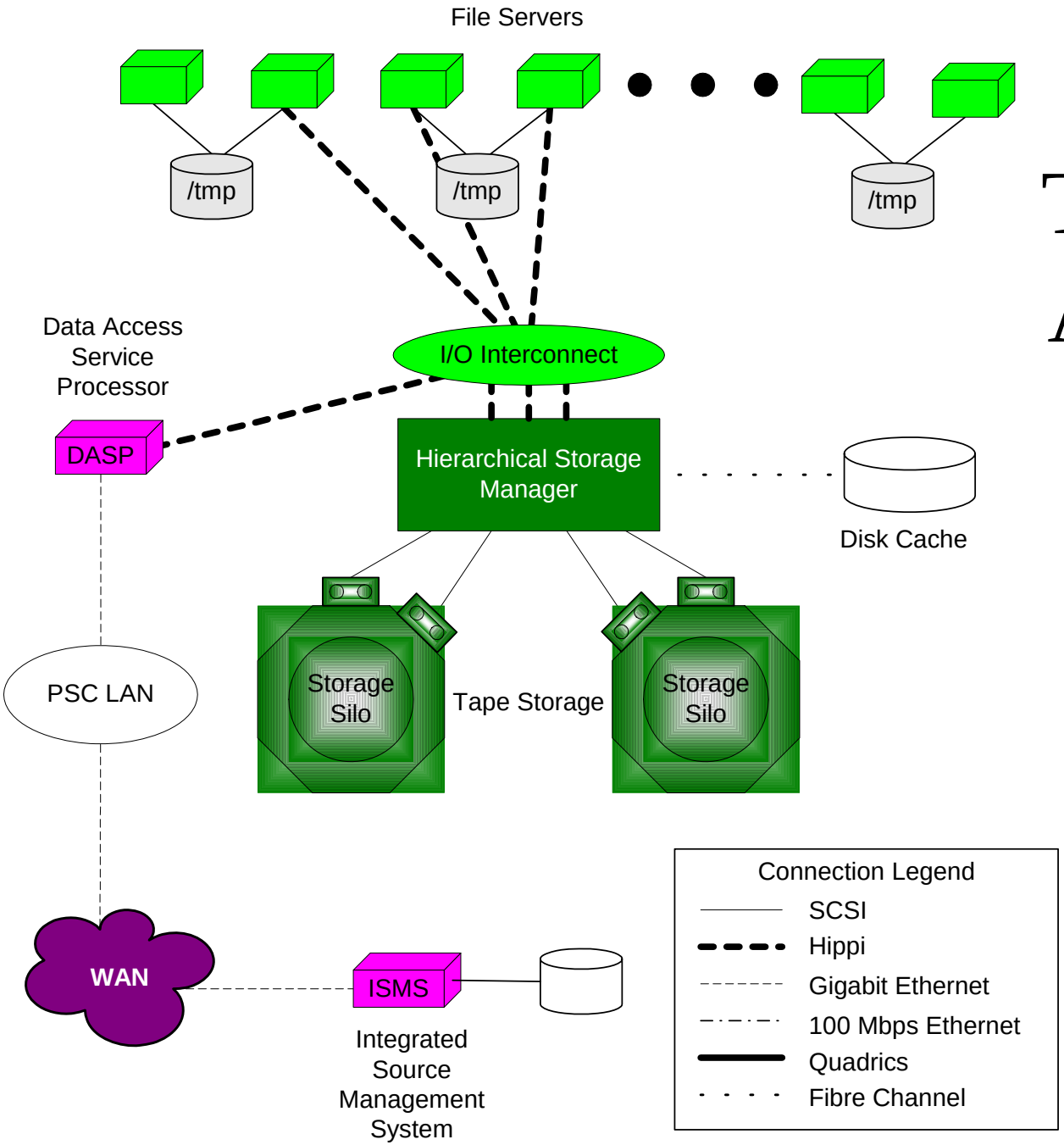
Storage Architecture

01110100011001010111001001100001011100110110001101100001011011000110101

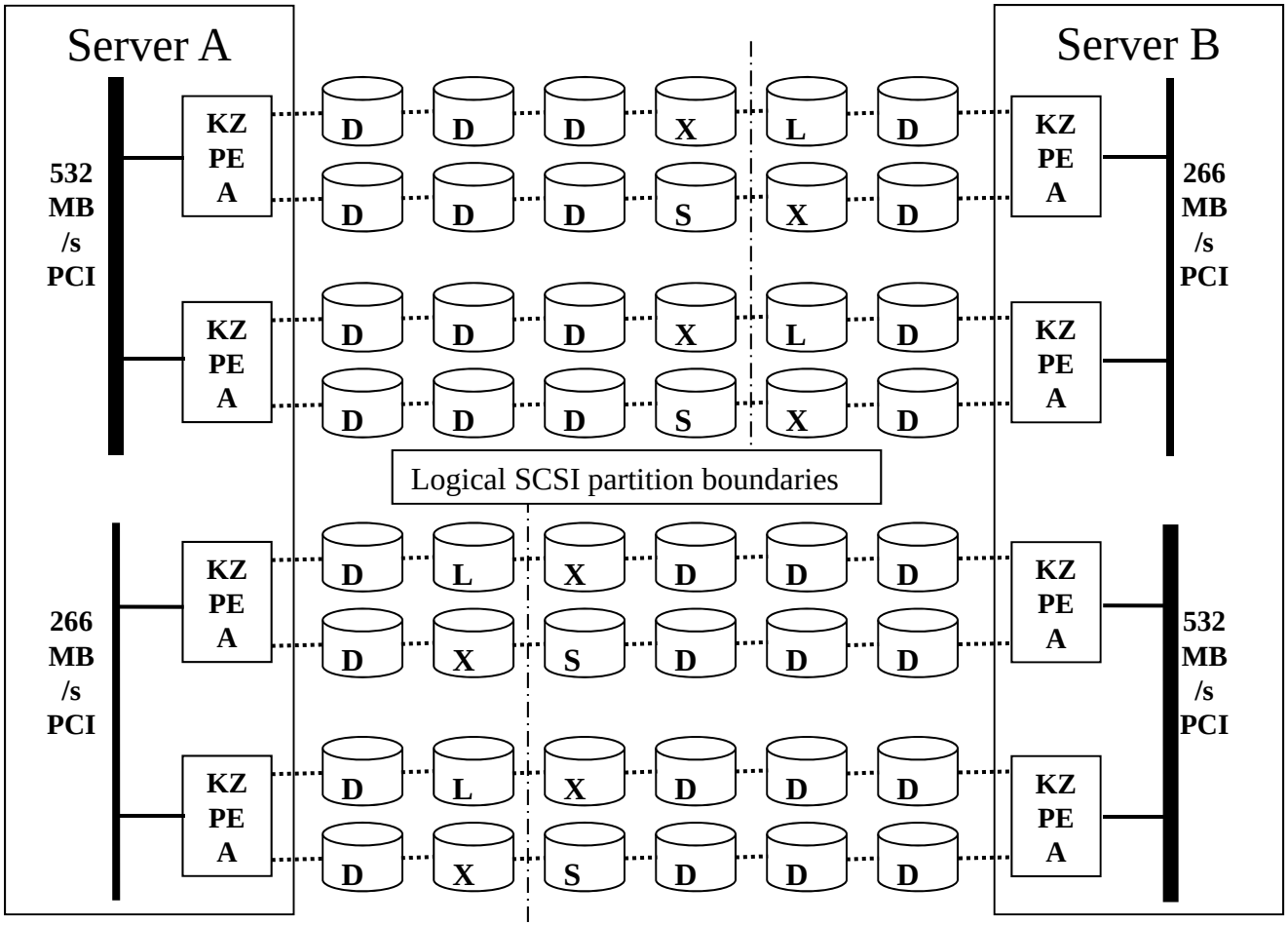
File Servers

- Reside in separate “TruCluster domains”
- Each FS will support:
  - 500 GB of storage space
  - 500 MB/s to disk (400 MB/s over QSW)
  - Multi-initiator SCSI
  - RAID-ed storage
  - Potential for “highly-tuned” volumes
    - e.g. writing to outer cylinders, direct/raw IO
- Can support fail-over of FS nodes

TCS Storage Architecture



File Server Bus Configuration*



D: Data storage
S: spare
L: Local storage
X: Expansion slot

Questions or Comments?

- Nathan Stone
  - nstone@psc.edu | www.psc.edu/~nstone/
- PSC Advanced Systems Group
  - advsys@psc.edu
- PSC Terascale Computing System Status
  - http://www.psc.edu/machines/tcs/status/