Experience with the Thumper

Wei Yang

Stanford Linear Accelerator Center

May 27-28, 2008

US ATLAS Tier 2/3 workshop University of Michigan, Ann Arbor

- **Thumper hardware configuration**
- **ZFS**
- **Sequential IO performance**
- **Simulated “Random” IO**
- **Performance issue with ZFS and Xrootd**

Thumper — SUN X4500 4U box

2x2 AMD Opteron 2.8Ghz, 16GB memory, 4x 1GigE

Service processor for remote control

Solaris 5.10 x86_64 and ZFS

48x 1TB (cheap) SATA drive, no RAID controller

- 2 mirror drives for OS (Solaris disk suite)
- 2 hot spare drive for zpool
- 44 drives for data
  - Total 32 TB usable
  - 4x RAID Z2 in one ZFS storage pool, each RAID Z2

    11 drives, including two drives for (dual) parities.

ZFS

End-to-End Data Integrity in ZFS

- ZFS Pool is a Merkle Tree with checksum in data block's parent
- Detect silent data corruption

Copy-on-write

- Allocate new block to write, then update points
- Never overwrite live data

Raid Z and Raid Z2

- Similar to Raid 50/60 but use variable stripe width

  Every block is its own RAID-Z stripe, regardless of blocksize
- Eliminate read-modify-write in RAID 5/6
- Eliminate "write hole"
- With COW, provide fast reconstruction when not near full capacity

No special hardware – ZFS loves cheap disks

End-to-End Data Integrity in ZFS

Disk Block Checksums

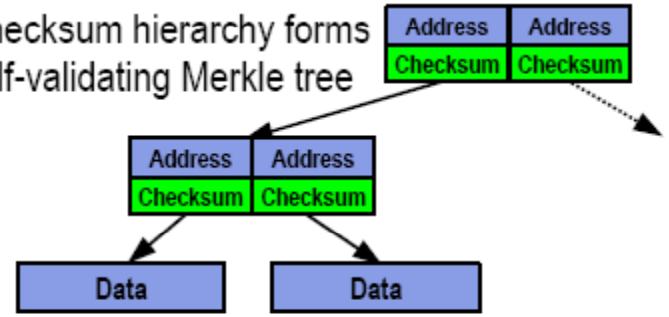
- Checksum stored with data block
- Any self-consistent block will pass
- Can't detect stray writes
- Inherent FS/volume interface limitation

Disk checksum only validates media

✔	Bit rot
✗	Phantom writes
✗	Misdirected reads and writes
✗	DMA parity errors
✗	Driver bugs
✗	Accidental overwrite

ZFS Data Authentication

- Checksum stored in parent block pointer
- Fault isolation between data and checksum
- Checksum hierarchy forms self-validating Merkle tree

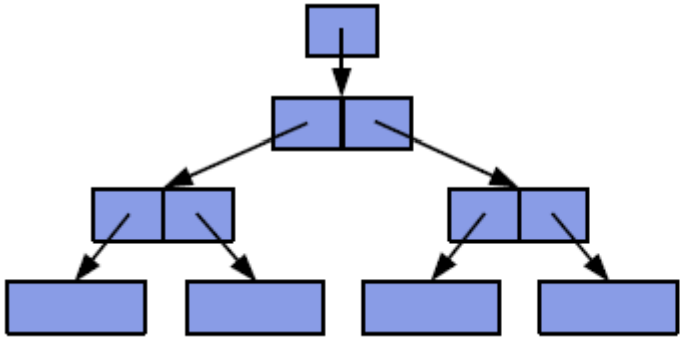


ZFS validates the entire I/O path

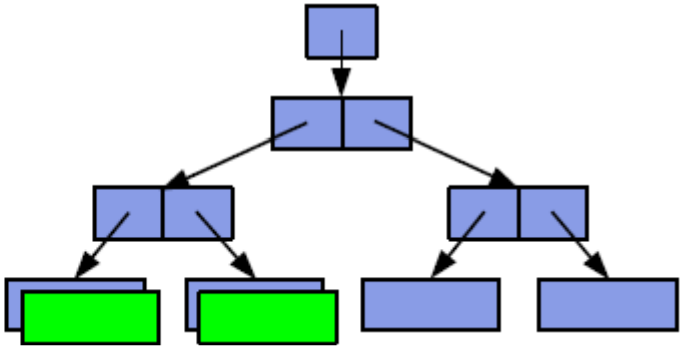
✔	Bit rot
✔	Phantom writes
✔	Misdirected reads and writes
✔	DMA parity errors
✔	Driver bugs
✔	Accidental overwrite

Copy-On-Write Transactions

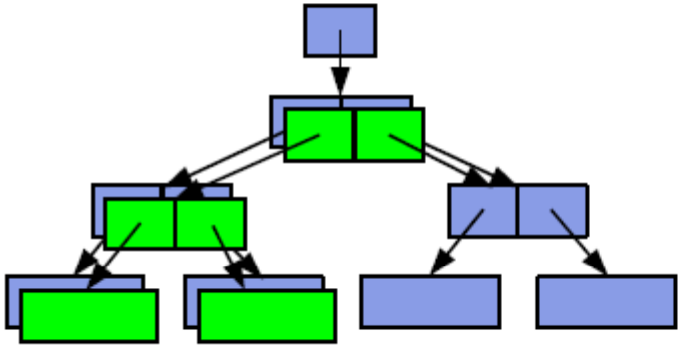
1. Initial block tree



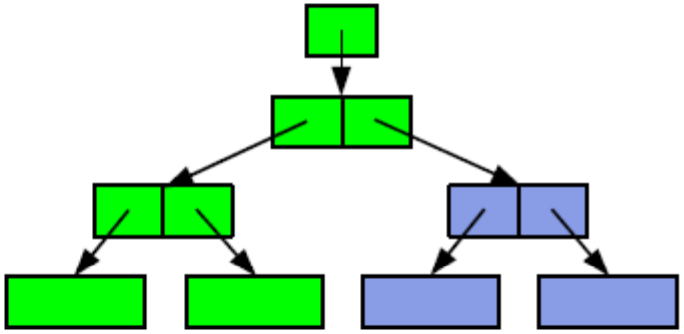
2. COW some blocks



3. COW indirect blocks



4. Rewrite uberblock (atomic)



Single Sequential IO

Write: dd bs=32k if=/dev/zero of=/…

Read: dd bs=32k if=/… of=/dev/null

Size (GB)	Write (MB/s)	Read(MB/s)
1	1024	533
5	697	592
10	750	538
64	562	576
256	451	504

Random IO

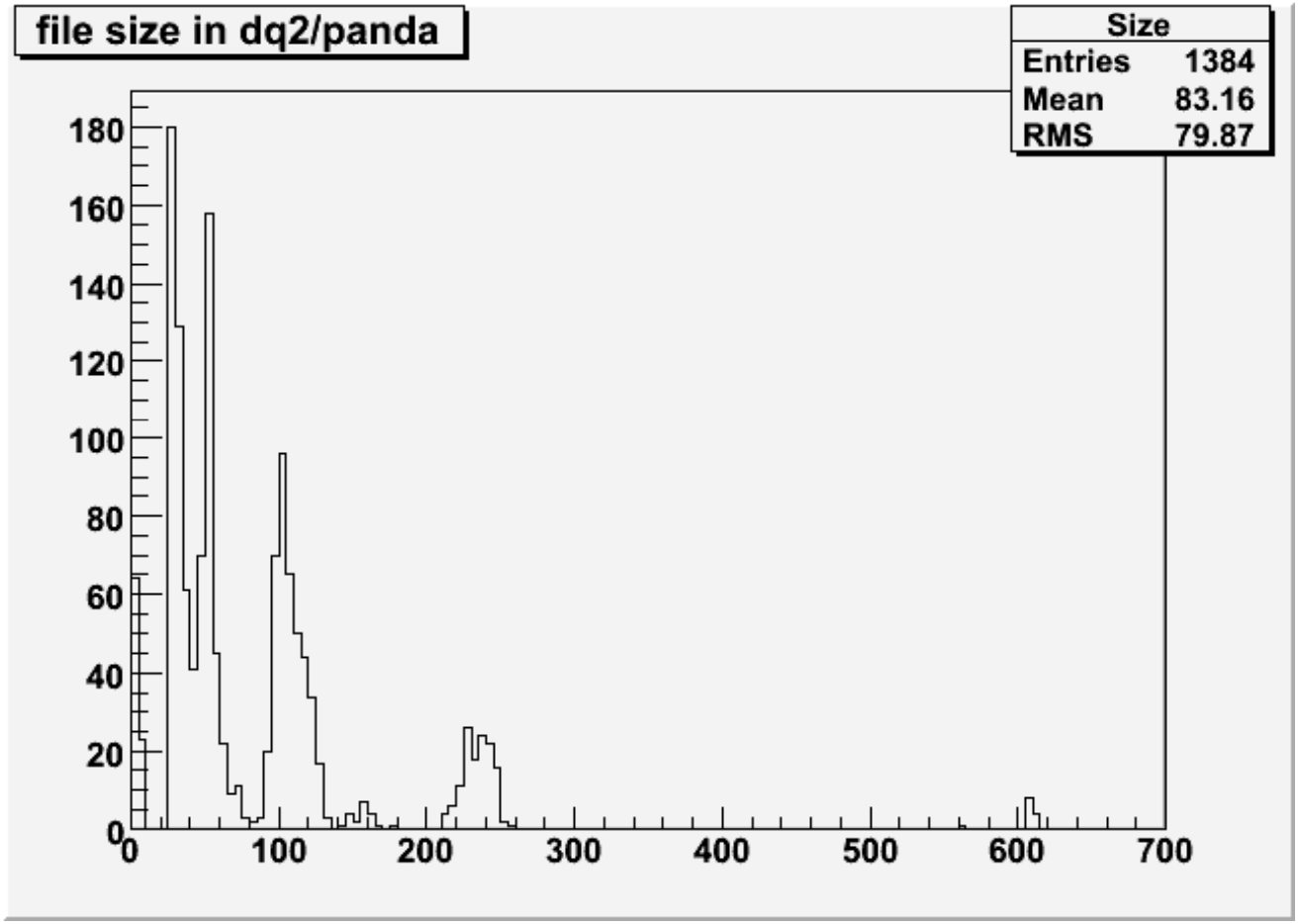
Real usage is the best place to observe thumpers "random" IO

- None of Babar, Glast and ATLAS thumpers is busy at this time.

Simulated "Random" IO

- Use 1384 files from …/dq2/panda,
  Rename them to 0,1…,1384
- Batch jobs to read them back in random order
- Try to similar Panda Production
  Copy the whole file using Xrootd native copying tool (xrdcp)

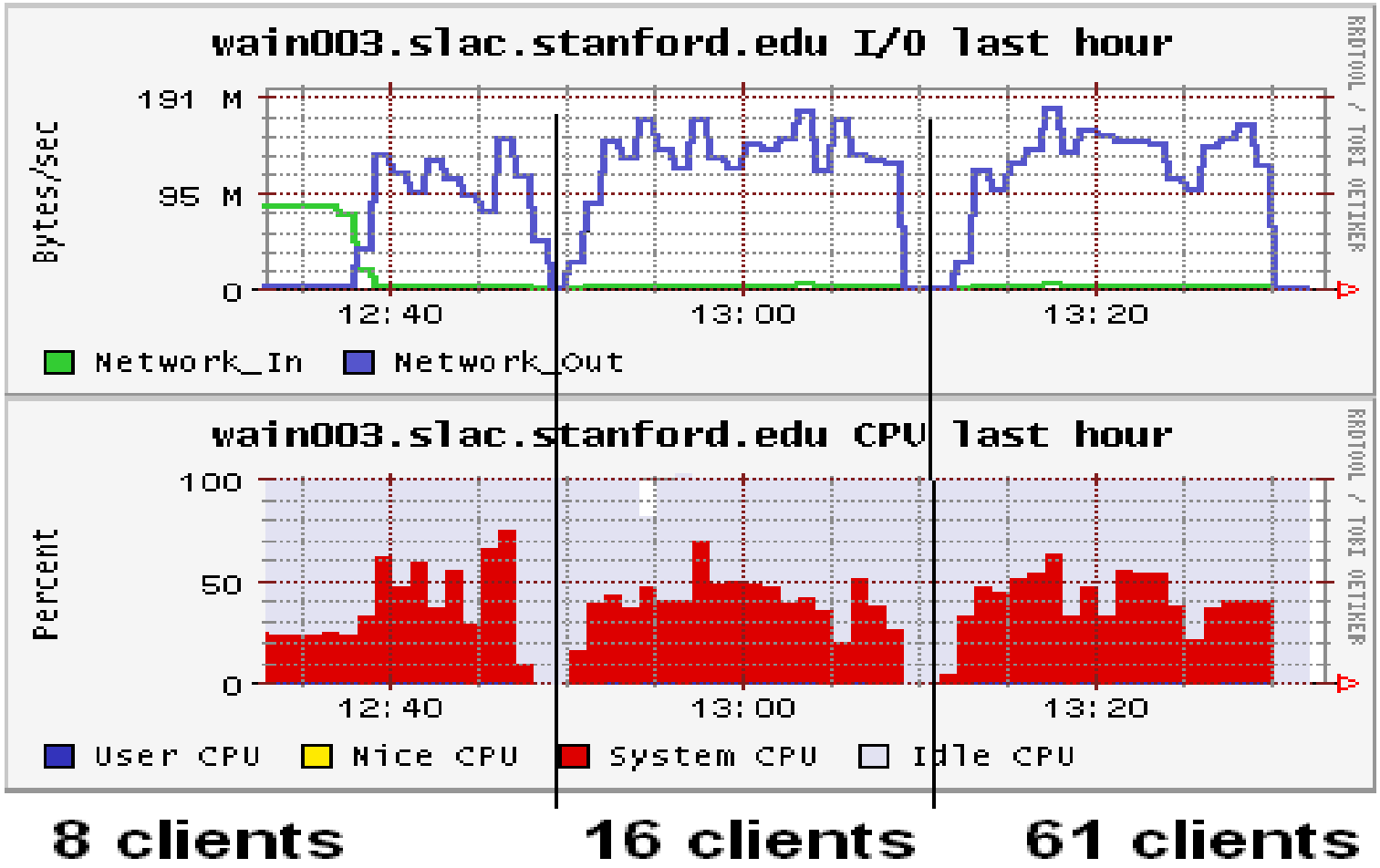
File size distribution (MB)




```
#!/bin/bash

typeset –i i

while [ 1 ]; do
    i=$RANDOM*1384/32768
    xrdcp -s root://wain003.slac.stanford.edu//path/$i /dev/null
done
```



Simulated Random IO: ~140 MByte/s for reading

Solaris and ZFS issues

900 xrootd reading and 450 xrootd writing of many 28 MB files

All start at the same time

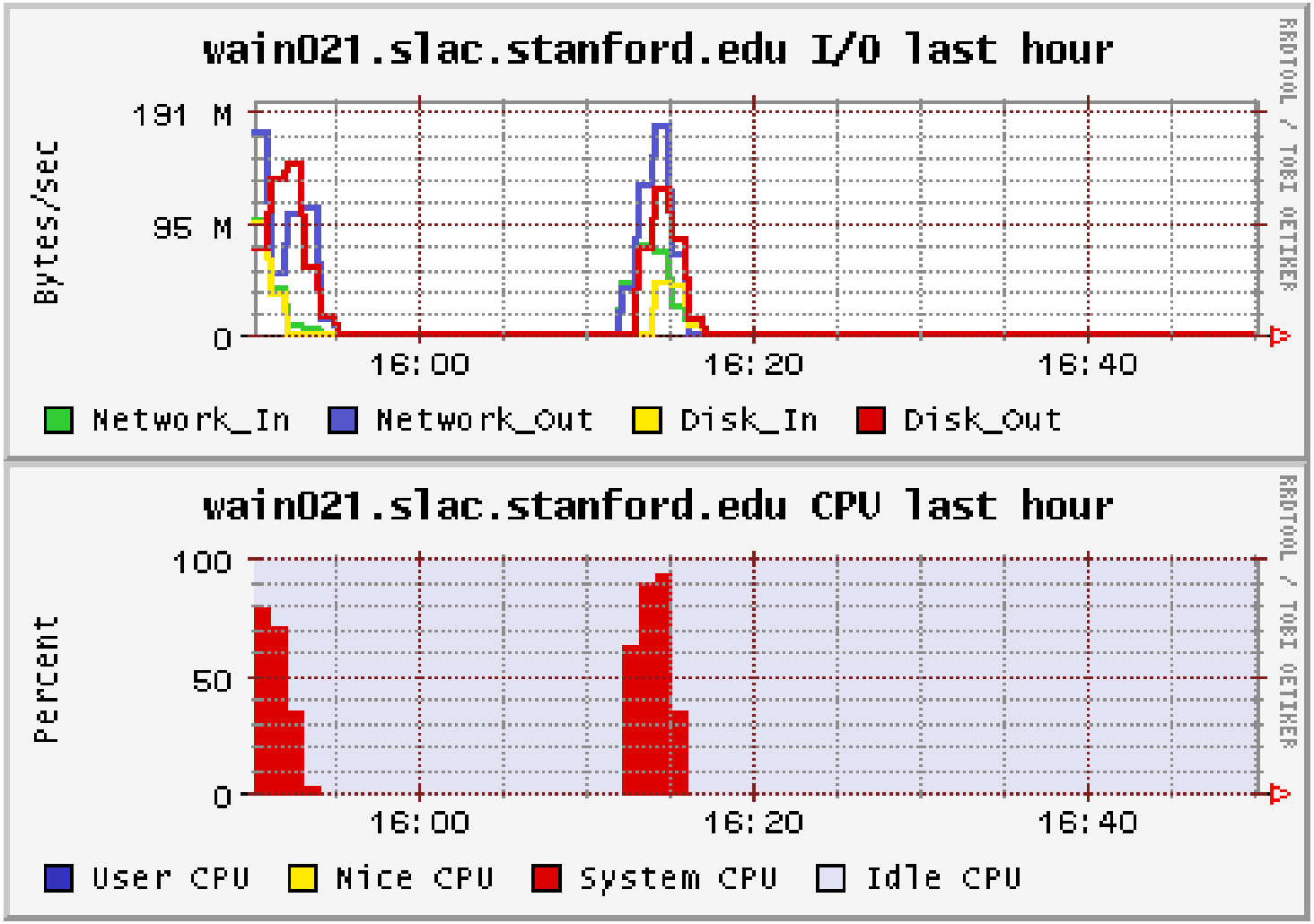
Solaris 10 x86_64 update 3

Do not take all connections, clients have to keep on trying

Solaris 10 x86_64 update 5

Take all connections, however Xrootd doesn’t respond to many of them

Limiting the Kernel memory usage of ZFS to 11GB solve the problem



Xrootd and ZFS complete all 900 reads and 450 writes after the memory usage of ZFS is limited to 11 GB

Summary

Thumper and ZFS well fit the need of scientific data storage

provides a relatively inexpensive, high density entry level storage.

ZFS is a filesystem and volume manager with many good features

- Run on Solaris 10
- A Linux port is in progress, however, it is based on FUSE
- ZFS sometimes use too much memory for caching

Simulated Random reading is around 140MB/s