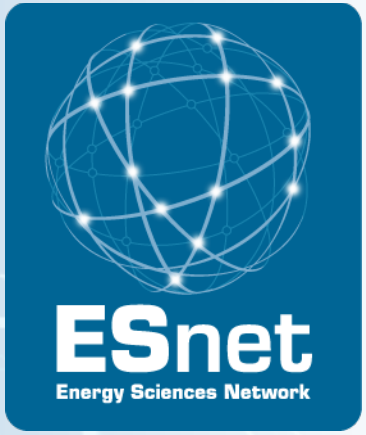


How to Build a Low Cost Data Transfer Node

Brian Tierney (bltierney@es.net)

Eric Pouyoul (lomax@es.net)

Eli Dart (dart@es.net)

Overview

- The Science DMZ
- DTN Introduction
- Gathering Requirements
- Break
- Design Process
- Performance Tuning
- Future: trends and hypes

Science Needs – the "Data Deluge"

"Data Deluge" is a nice cliché

- Dramatic and catchy
- It's been used many times before
- That's because it's a good way of describing the issue succinctly

Data volume is causing many disciplines to re-think their strategies, collaboration structures, etc. – Examples:

- Genomics
- Materials science (e.g. light source users)
- Medicine

Most disciplines do not have deep internal networking expertise – they need help to use networks well

Benefit of networks is often unrealized

The network serves many scientists poorly or not at all

- Without in-house (or in-collaboration) support, scientists are typically left to figure networking out for themselves
- The typical kit is composed of hosts with default configurations, and the SSH toolset
- If problems are encountered, the scientist is expected to find the right networking organization, explain the problem in networking terms, and help troubleshoot the problem
  - This model has demonstrably failed
  - Typical problem resolution time is on the scale of weeks
- Most scientists that have tried to use the network for data transfer or visualization have failed and have stopped trying – they “know” it can’t be done (this also means there are no trouble reports)

Current Problems – Technical Causes

Network device and host issues → poor performance

- Many networks still have packet loss (incorrect config, cheap hardware, poor design, etc)
- System defaults are wrong – hosts still need to be tuned

Wrong tool for the job → poor performance

- Use of SSH-based tools is common
- SSH has built-in, protocol-level performance limitations (10x to 50x slower than GridFTP on systems with high-performance storage)

Security blocks scientists at every turn

- Tools are blocked at the network layer or disallowed by policy
- Firewalls cause poor performance
- High-performance tools are often incompatible with system access technologies (e.g. SSH)

Consequences if these problems persist

Science will proceed with or without networks

Networks will decrease in relevance for most scientific disciplines and the institutions that support the science unless networks can be made more useful

- Lower return on investment in scientific infrastructure
- Longer time to discovery, loss of institutional leadership in key fields
- Reduction of technological and scientific leadership for USA

If the high-performance networks built for science are not useful – if the scientists can't use them effectively – then we have built a facility that is of no value to science

- Productivity/collaboration benefits from networking not realized
- Network-enabled modes of discovery unavailable
- For many disciplines, THIS IS THE CURRENT STATE

All Is Not Lost

Bad outcomes are not certain

There are successes – they just need to be generalized

Leadership by the networking community is required

- The old model of providing a toolkit and expecting scientists to learn networking has demonstrably failed
- The networking community must provide useful services and useful documentation for those services

Remember – *Most users are not networking experts, and it is unreasonable to expect them to become experts*

How to enable scientific use of networks?

It must be easy for scientists to use the network!

In many cases, the data-intensive part of a scientific experiment can run on one machine

Build a simple enclave for data-intensive services

- Near site perimeter
- Separate security policy
- No need to burden converged, multi-service network infrastructure with high-data-rate WAN flow requirements
  - Switches and routers with deep packet buffers are expensive
  - Debugging performance problems in converged networks is labor-intensive

Traditional Network DMZ
http://en.wikipedia.org/wiki/DMZ_(computing)

Network DMZ:

- A physical or logical subnetwork that contains and exposes an organization's external services to a larger untrusted network, usually the Internet.
- Commonly used architectural element for deploying WAN-facing services (e.g. email, DNS, web)

Traffic for WAN-facing services does not traverse the LAN

- WAN flows are isolated from LAN traffic
- Infrastructure for WAN services is specifically configured for WAN

Separation of security policy improves both LAN and WAN

- No conflation of security policy between LAN hosts and WAN services
- DMZ hosts provide specific services
- LAN hosts must traverse the same ACLs as WAN hosts to access DMZ

The Science DMZ

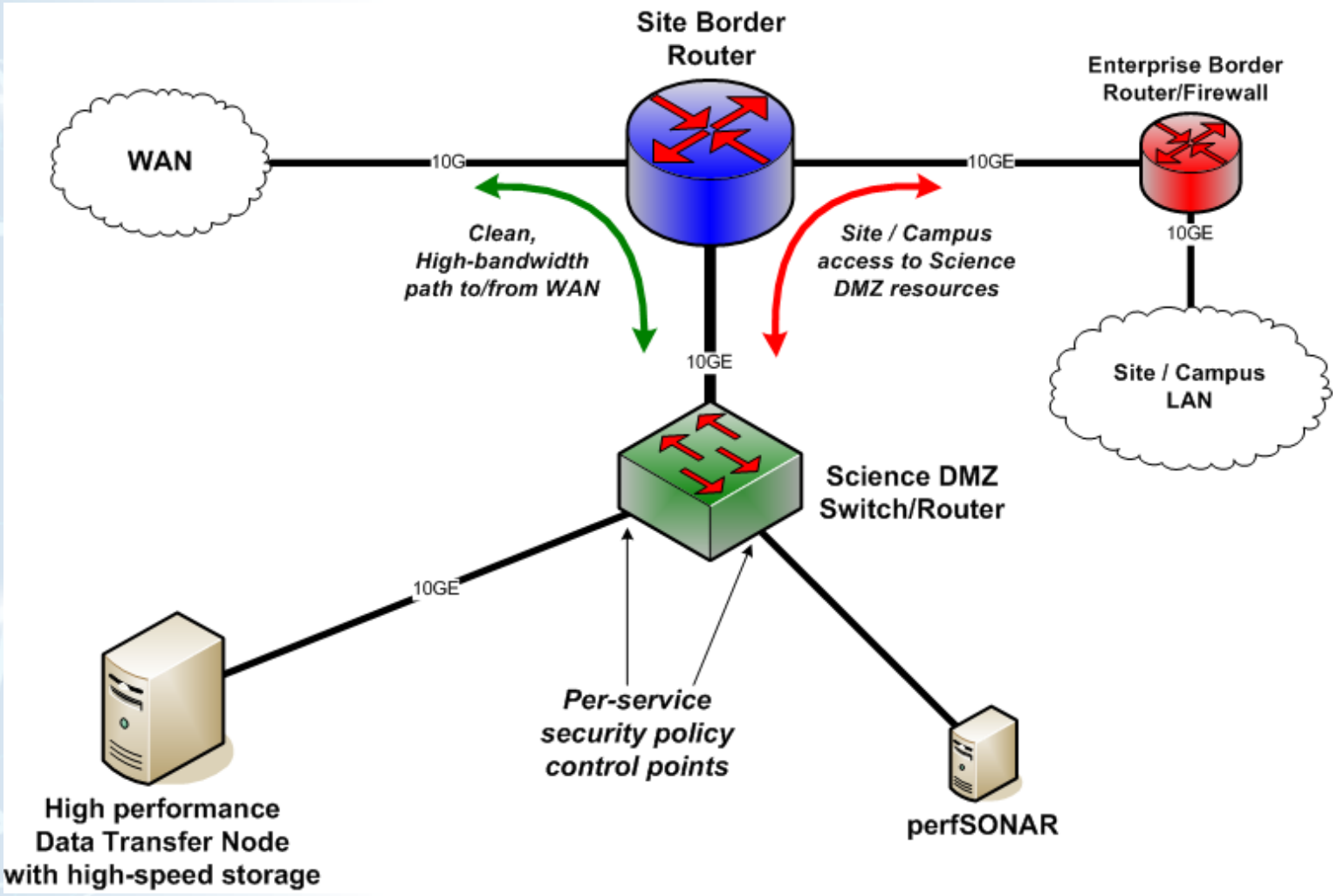
Science DMZ – a well-configured location for high-performance WAN-facing science services

- Located at or near site perimeter on dedicated infrastructure
- **Dedicated, high-performance data movers**
- Highly capable network devices (wire-speed, deep queues)
- Virtual circuit infrastructure
- perfSONAR

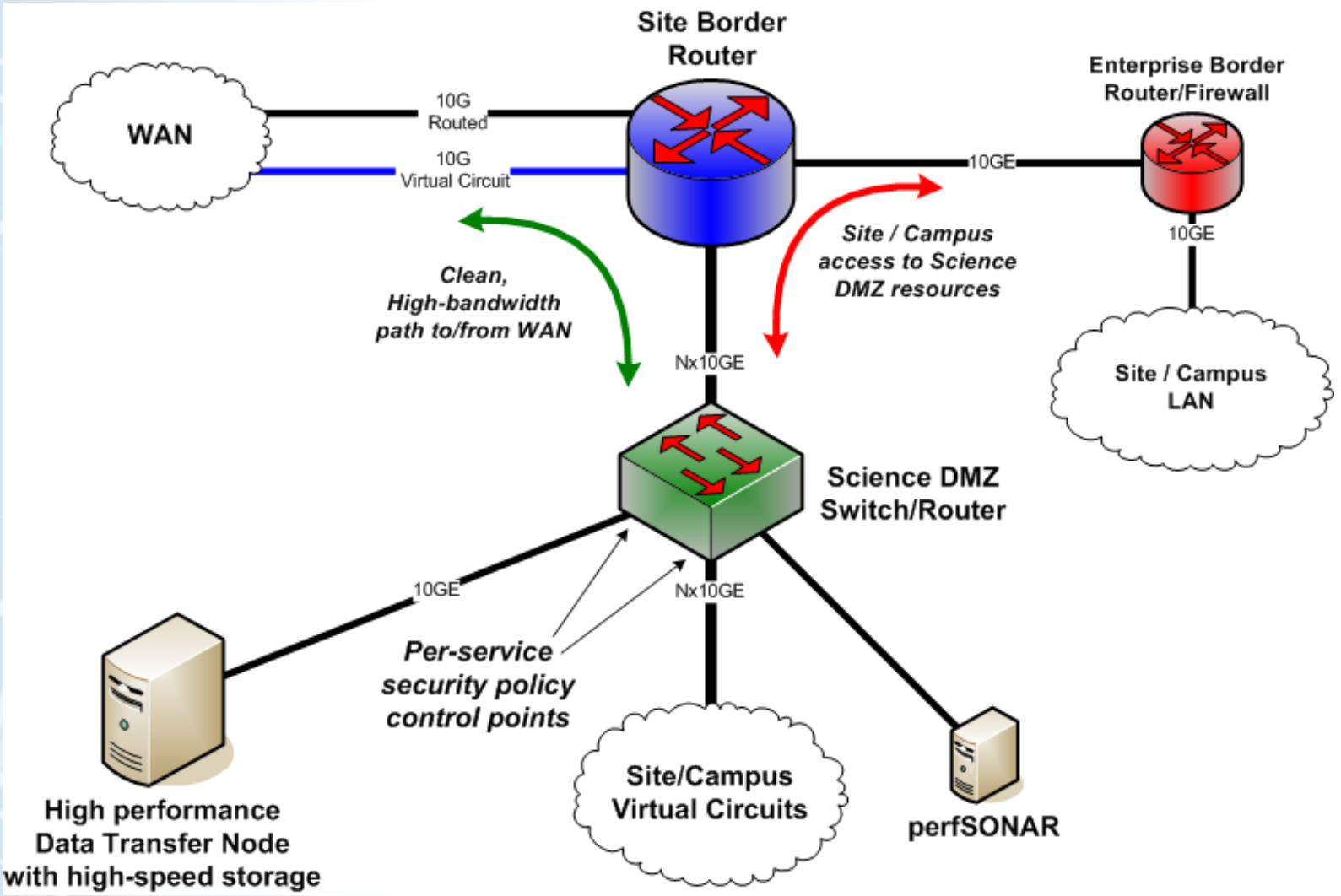
DYNES project is a specific example of this general architectural theme

- Dedicated infrastructure for data movers
- Virtual circuit termination
- Many high-bandwidth science sites have moved to this architecture already as a matter of necessity – DYNES is expanding on this because it works

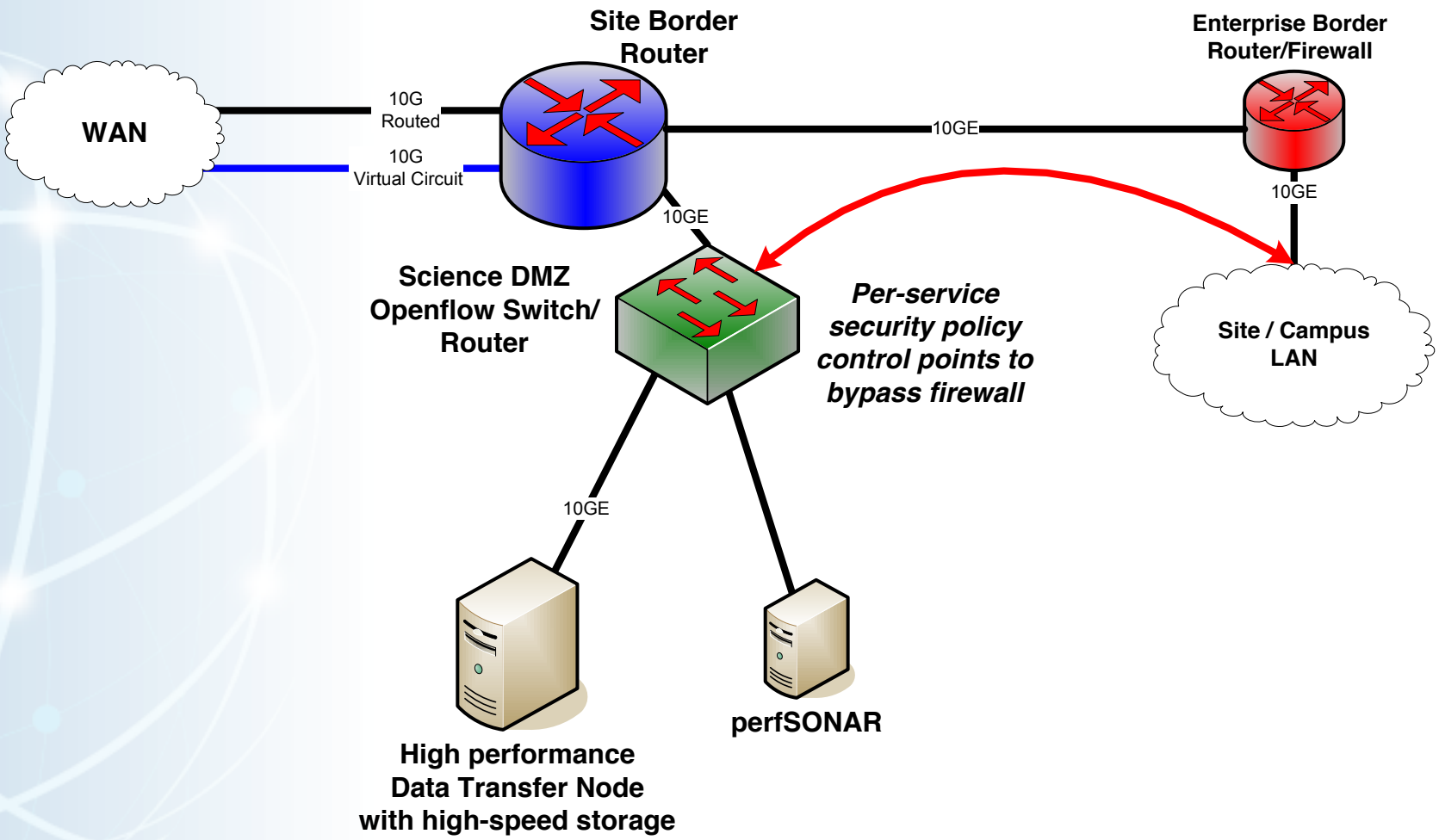
Science DMZ – Conceptual Diagram



Science DMZ - Advanced



Science DMZ and OpenFlow



The Data Transfer Node (DTN)

Dedicated, high-performance host for long-distance data transfer

High performance disk, for example:

- High-speed local RAID
- Fibrechannel attachment to SAN, if available
- Lustre or GPFS filesystem mount (e.g. when deployed at supercomputer center)

High-speed network connection (10G)

- Connected to Science DMZ
- Separate security policy from business traffic

Multiple sites and facilities are deploying DTNs (Supercomputer centers, labs, experiments, etc)

Significant performance gains from DTN deployment

Science DMZ Benefits

LAN infrastructure need not carry wide area science traffic

- Science traffic has different characteristics than business traffic
- Deep output queues, dedicated interfaces, etc. are expensive
- Accurate counters, per-filter counters, etc. are expensive

LAN transfers are much easier than WAN transfers

- Internal transfer of data from the site/campus LAN to/from the local Science DMZ will be much easier to debug, and is much more tolerant of the loss typically found in LANs
- LAN losses do not affect WAN transfers

Packet loss and WAN performance

TCP needs very very low packet loss to achieve good WAN performance.

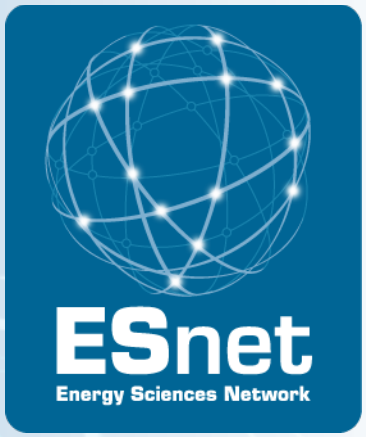
ESnet example: bad line card was causing 0.0046% loss in one direction

- 1 packets out of 22000 packets

Performance impact of this: (outbound/inbound)

- To/from test host 1 ms RTT : 7.3 Gbps out / 9.8 Gbps in
- To/from test host 11 ms RTT: 1 Gbps out / 9.5 Gbps in
- To/from test host 51ms RTT: 122 Mbps out / 7 Gbps in
- To/from test host 88 ms RTT: 60 Mbps out / 5 Gbps in

VERY few end users know enough about network capability to complain about 60 Mbps performance



Building a Data Transfer Node

DTN Introduction

Performance hosts can be of different flavors:

- Test hosts: maximum performance, runs well defined and limited tests. Used by engineering (network, backend). Those hosts are typically deployed in the infrastructure or are shippable.
- Prototype hosts: support a specific application or workload. Reliability and usability may not matter.
- Service hosts: those hosts are production level and provides a service to users (such a Science DMZ DTN).

What is your Data Transfer Node ?

One host does not fit all needs

- Capacity (GB vs TB vs PB)
- Performance
- Backend topology (Infiniband, Fiber)
- Application support (gridFTP, Lustre, streaming,…)
- Infrastructure integration

Design: Multi-purpose versus Custom

Multi-purpose

- No choice: Apps, Collaboration
- Large scale
- Few limitations
- Known and supported

- Expensive
- Complex
- Large footprint

Custom

- Cheaper
- The right tool for the problem
- Simple
- Small footprint

- Limited
- Small scale
- No third party support

Balanced System Design

A host is a system made of “fundamental” elements:

- Motherboard (architecture)
- Processing
- Memory
- I/O (storage)
- Networking

The performance of any system is limited by the element behaving as a bottleneck

Design is done by adjusting “performance knobs” on each element, by using faster harder, and then fine tuning.

Architecture: Intel versus AMD

Common

- Operating System support
- Reliability
- Virtualization

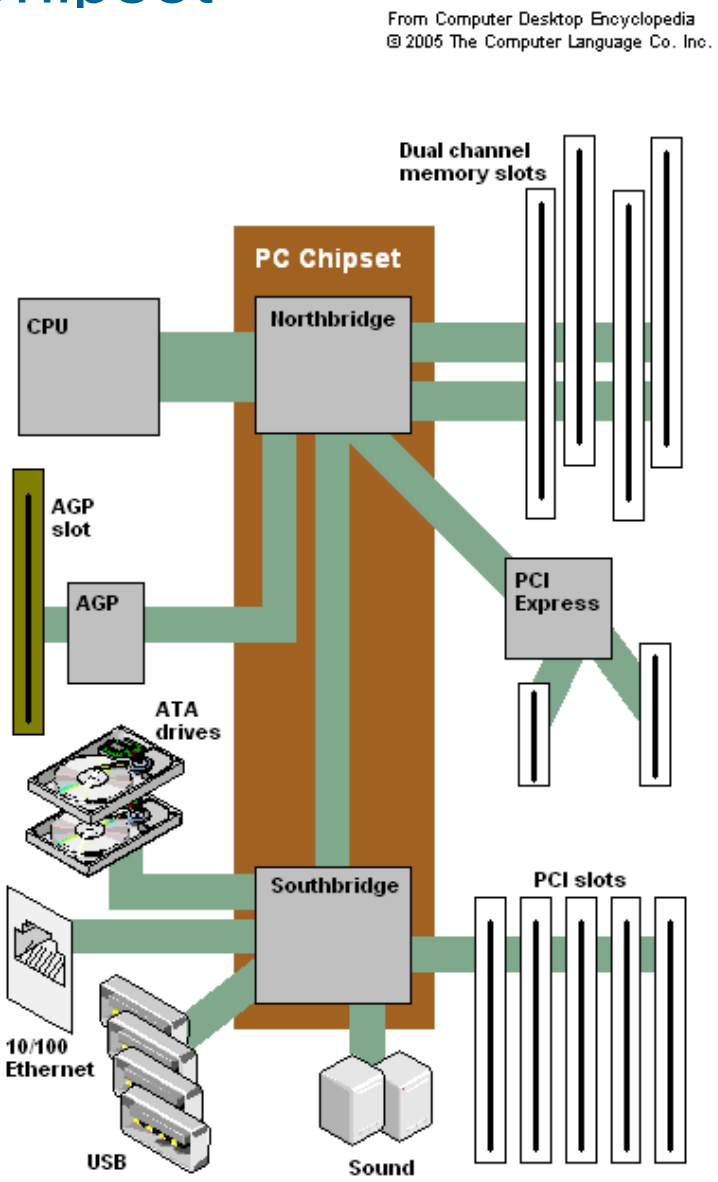
Intel

- Faster clock
- Lot of choices in motherboards

AMD

- More cores
- More energy efficient
- Cheaper

Plumbing: Chipset



Architecture and chipset matter

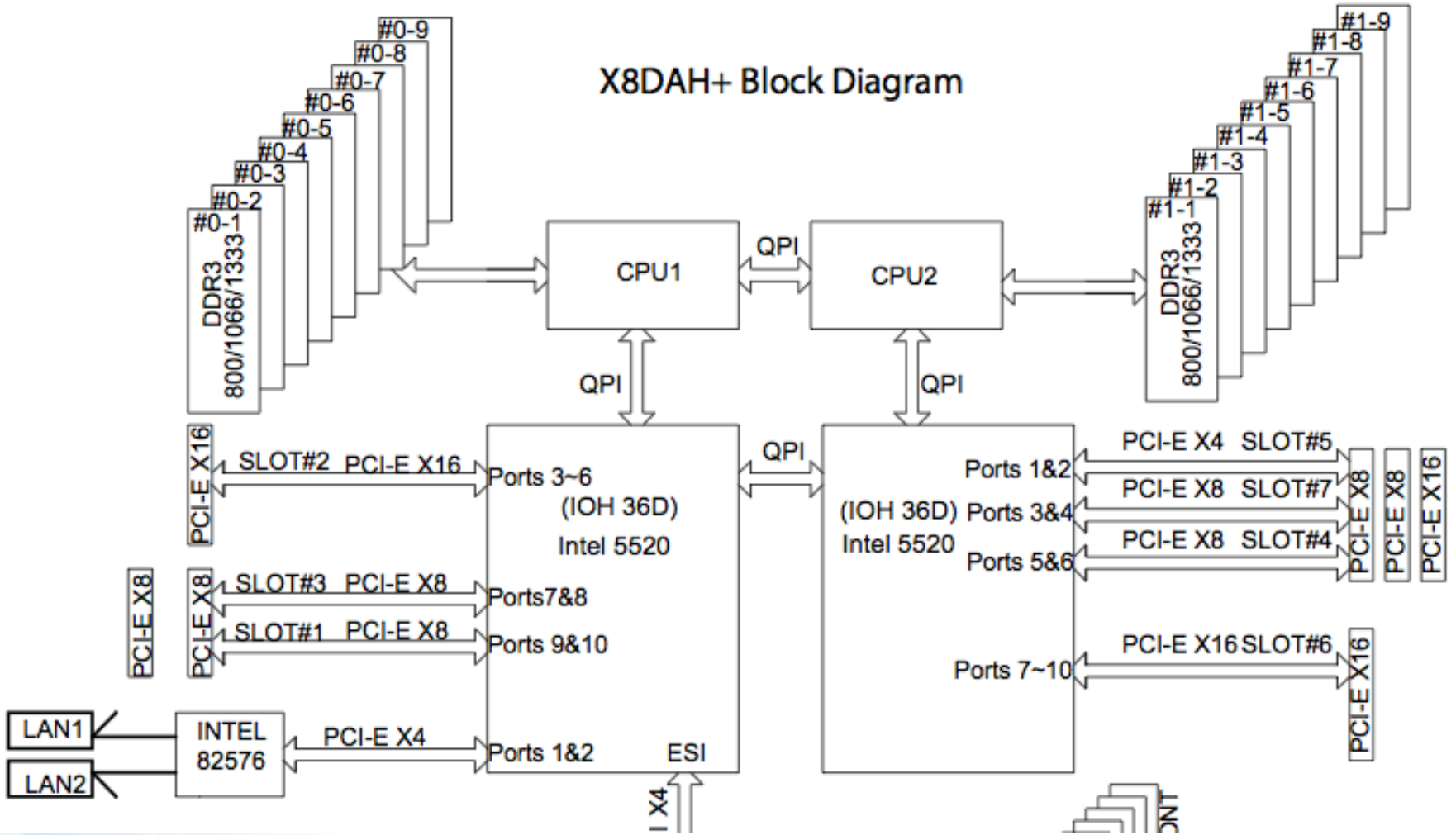
Memory bandwidth

- typical 8 GB/sec (stream)
- High end 31 GB/sec (stream)

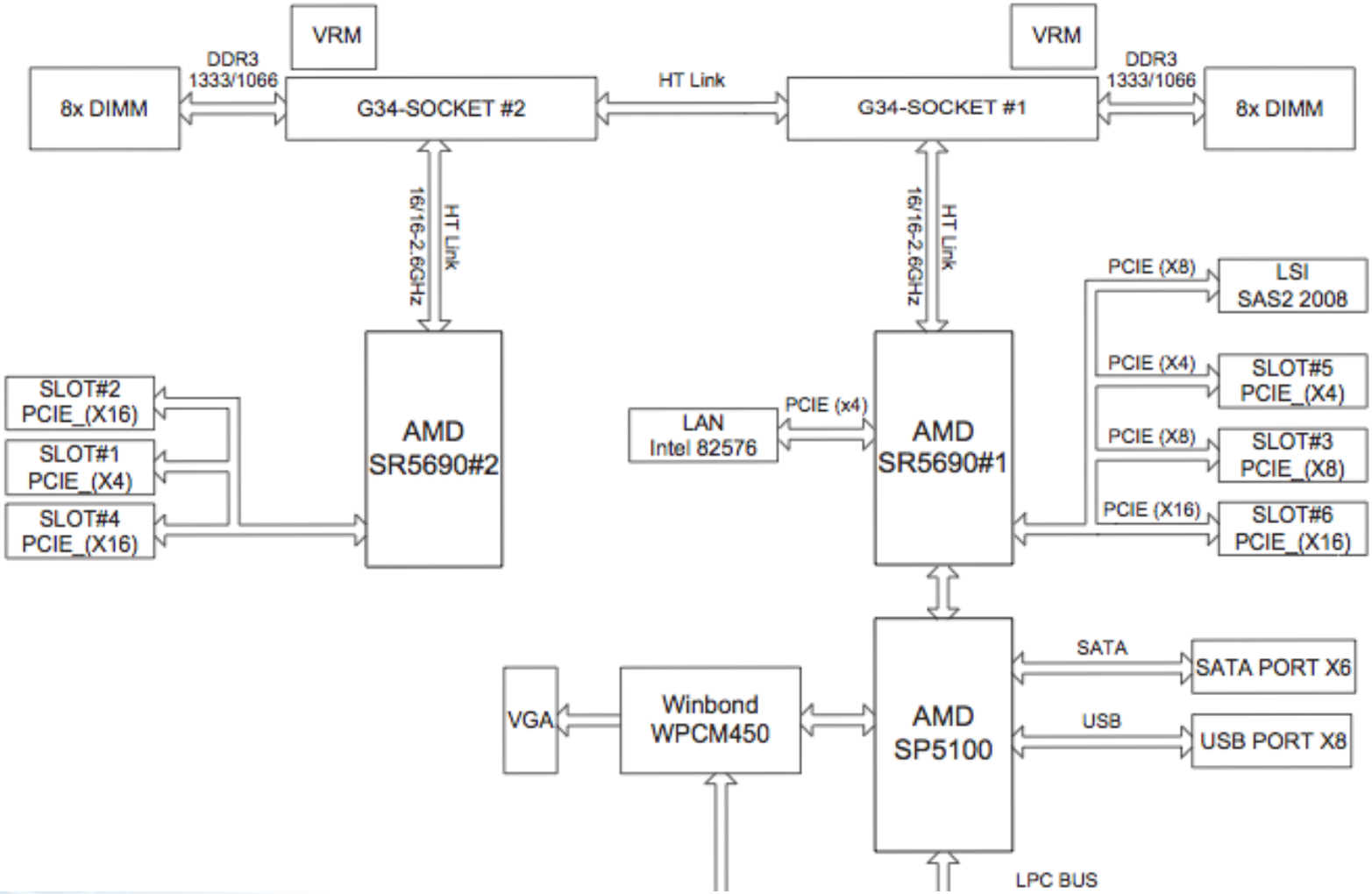
PCIe bandwidth

- PCIe 2.0: (500 MB/sec per lane)
  - Typical up to 4 GB/sec (8 lanes or x8)
  - High end up to 8 GB/sec (16 lanes or x16)
- PCIe 3.0: will double bandwidth

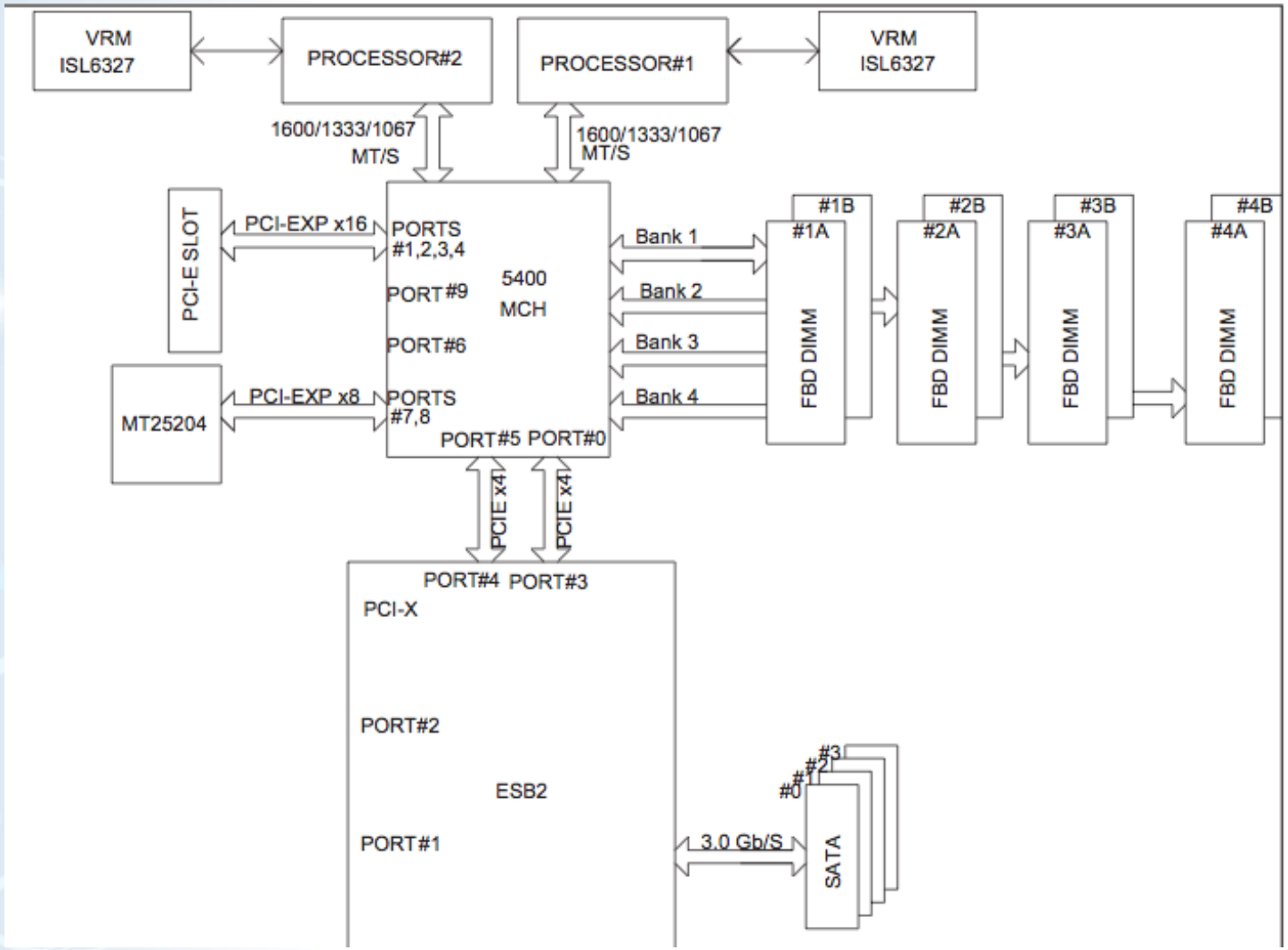
Intel: SuperMicro X8DAH+-F



AMD: SuperMicro H8DG6-F



Low performance: SuperMicro X7DWT



So, AMD or Intel ?

- Currently, Intel has a faster bus (QPI) than AMD's HT's
- Intel has advantage on pure computation
- Due to bugs in the chipset handling QPI, high performance PCIe based card (i.e. > 8 lanes), AMD is better for fast I/O (lower latency)
- AMD can support both DDR2 and DDR3 memory DIMMS, while Intel can only support DDR3 ➔ AMD has potentially a better value
- AMD typically supports architecture much longer than Intel (backward compatibility).

AMD and Intel alternates as the leader in performance computing (look at manufacturing problems, etc)

Processing

Sequential processing

- Pure clock matters
- L1/L2 Cache may increase performance
- Will likely be blocked by memory or I/O

Parallel processing

- Number of sockets, number of cores
- Synchronization
  - Explicit (locks, messages)
  - Implicit (MPP)
- Will likely be blocked by slow threads or nodes

Memory

Memory type:

- DDR2 if moderate memory usage, DDR3 if heavy memory usage.
- Be aware of best price / capacity.
- Always follow motherboard, chipset recommendations for best performance.

Memory size:

- Enough memory for application: never swap
- Plan for I/O cache (raw, files system) if needed

Storage

Storage I/O

Usage pattern:

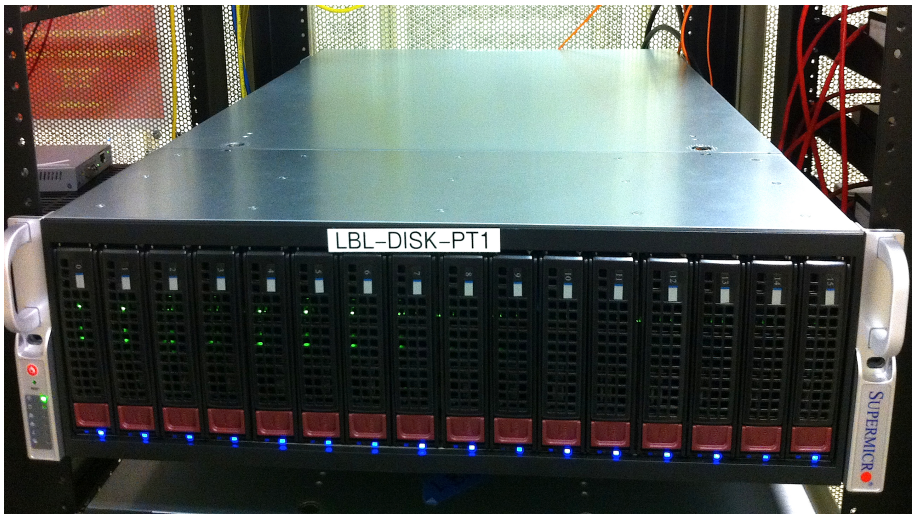
- Sequential versus Random
- Read versus Write
- Level of parallelism
- Capacity
- Reliability

Storage type

- Local disks: RAID set(s)
- Storage subsystem: FC attached
- Networked Storage (Ethernet, Infiniband)

Local Storage

- I/O via PCIe
- Integrated server
- Customization



- Small footprint
- Faster I/O
- Good price/performance
- Standard technologies
- Can scale up to 40TB

- No distributed storage
- Limited scaling
- Rough disk management

Local Storage: RAID Levels

- RAID0: stripes data. Best performance, no reliability
- RAID1, 10 : mirrors data: Best reliability, half capacity, half performance
- RAID5: Decent reliability, 2/3 of capacity. Performance varies.
- RAID6: Similar to RAID5, but supports two disk failures.
- Other RAID: vendor specific. Dedicated to a given workflow
- File System RAID: BSD's ZFS and Linux' BTRFS

RAID: Great (cheap), but Experiment First

RAID is a bottleneck !

Performance depends on RAID engine

Select the right RAID Level

- RAID0: need best I/O performance but can afford losing all dataset.
- RAID5/6: need reliability, can afford to only have 2/3 of capacity and performance of RAID0.

Select the right RAID Controller

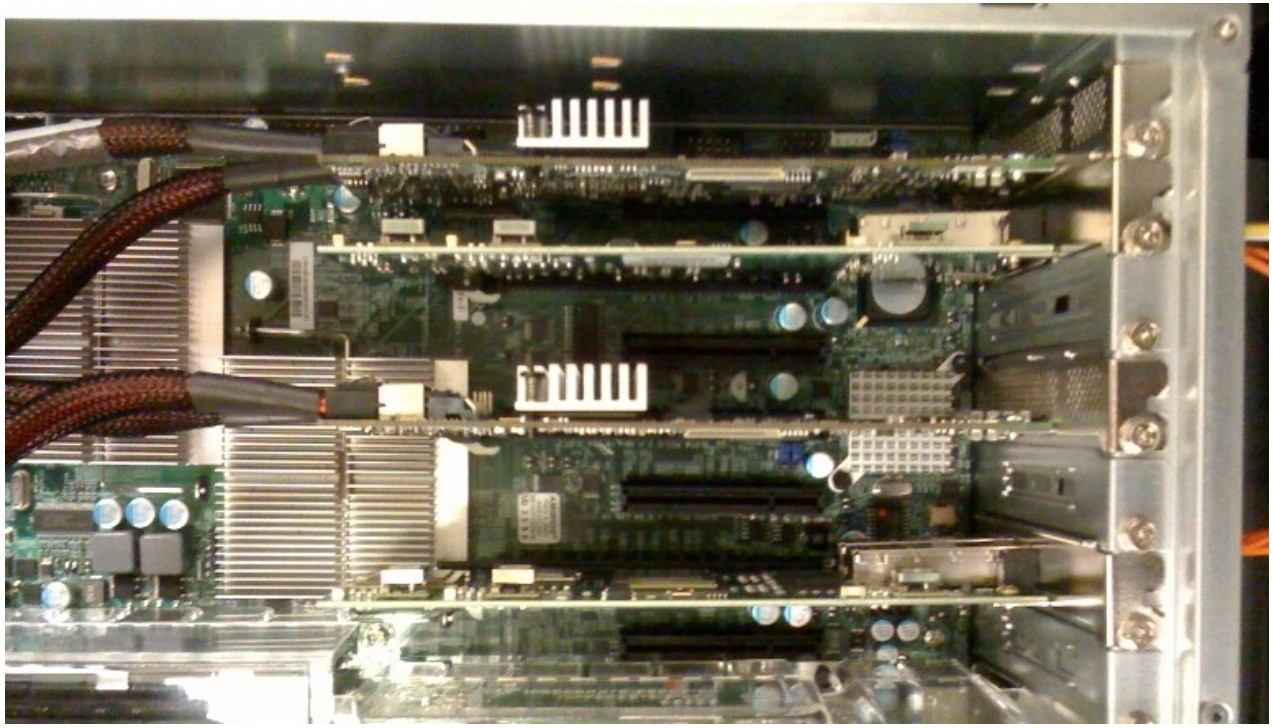
Plan for expansion

Experiment on a prototype system first

RAID Controllers

- Often optimized for a given workload, rarely for performance.
- RAID0 requires less CPU than other RAID levels.
- The CPU required to process queries is a factor of the number of drives.
- Remember this is a PCIe card: verify number of lanes.

RAID, Drive Topology & File System



- Number of drives and mid-plane topology (cabling)
- Number of controllers and best performance RAID set
- Online spare drives
- File System constraints (several file systems versus single)

Focus on your Requirements

- How many file systems ?
- How much performance per file system?
- How much parallelization per file system?
- How much future expansion?

Exercise: I/O Subsystem Design

- 16 SAS drives chassis, wired in 4 sets of 4 drives
- 130 MB/sec per drive
- 2 x 8 ports RAID controllers (Group 1 & 3)
  - RAID0: peak performance per RAID set at 4 drives
  - RAID5: peak performance per RAID set at 5 drives
- Or 1 x 24 port RAID controller (Group 2 & 4)
  - RAID0: peak performance per RAID set at 6 drives
  - RAID5: peak performance per RAID set at 8 drives
- Group 1&2: Test host (Pure performance): few big files (>4GB)
- Group 3&4: DTN (Production Drop Box) : lots of medim/small files (< 4GB)

1. Applications
2. RAID Level and disk topology (how many drives per raid set)
3. File System (type and topology)
4. Expected performance

Group 1 - Testing Host
2 x 8 ports RAID Controllers

1. Application(s)

2. RAID Level and topology

3. File System & topology

4. Expected Performance

Group 2 - Testing Host
1 x 24 ports RAID Controller

1. Application(s)

2. RAID Level and topology

3. File System & topology

4. Expected Performance

Group 3 - Drop Box
2 x 8 ports RAID Controllers

1. Application(s)

2. RAID Level and topology

3. File System & topology

4. Expected Performance

Group 4 - Drop Box
1 x 24 ports RAID Controller

1. Application(s)

2. RAID Level and topology

3. File System & topology

4. Expected Performance

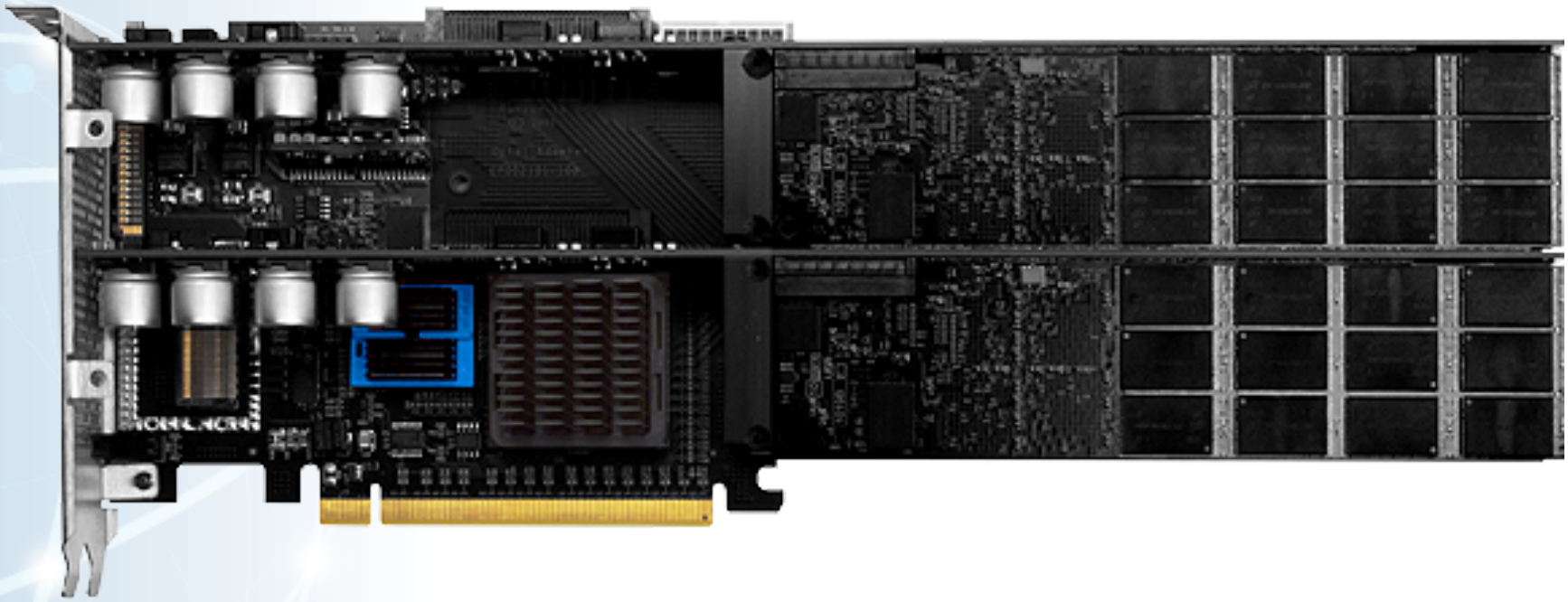
Break

Local Storage Discussion Alternatives Technologies

- SSD PCIe Cards
- JBOD and File Systems (ZFS, BTRFS)
- Software compression

- Open discussion items

SSD Storage: the wow factor



SSD: Current State

- 6 GB/sec read (PCIe 2.0 x16) ! Expect even more with PCIe 3.0.
- Acceptable/Excellent MTBF
- Still more expensive
- Potentially harder to deploy within standard IT
- Migration path / Hybrid – Needs high end RAID Controllers

Software RAID: JBOD and File System

- Linux MD: simple to deploy. Unstable, difficult to reach good performance.
- Solaris, BSD' ZFS: excellent balance hardware/software. Can quickly require lot of CPU power (both cores and clock rate)
- Linux BTRFS: not mature yet, but on its way to be.
- Compression: in some cases, increases I/O bandwidth (assuming you have enough cores to burn)

Network

Network Interface Controller

- Data Link: depends on infrastructure
  - Copper 1Gb
  - Fiber Ethernet (1G, 10G, 40G)
  - Converged Ethernet
- Driver support with Operating System: fine grain tuning
- Protocol offloading
- Dual port is mostly for fail over

- Double check connector type before ordering optics
- Remember PCIe bandwidth still matters

Interrupt Affinity

Interrupts are triggered by I/O cards (storage, network). High performance requires lots of interrupts per seconds.

Interrupt handlers are executed on a core

By default, core 0 gets all the interrupts, or interrupts are dispatched in a round-robin fashion among the cores: both are bad for performance:

- Core 0 get all interrupts: with very fast I/O, the core is overwhelmed and becomes a bottleneck
- Round-robin dispatch: very likely, the core that executes the interrupt handler will not have the code in its L1 cache.

A simple solution: interrupt binding

- Each interrupt is statically bound to a given core (network -> core 1, disk -> core 2)
- Works well, but can become an headache and does not fully solve the problem: one very fast card can still overwhelm the core.
- Need to bind application to the same cores for best optimization: what about multi-threaded applications, for which we want one thread = one core ?

PCI optimization: MSI-X

Extension to MSI (Message Signaled Interrupts)

Increases the number of interrupt “pins” per card

Associates rx/tx queues to a given core

Ensures that the thread that runs the program and the asynchronous events it may receive (incoming network packets, asynchronous I/O etc.) run on the same core

- maximizes L1 cache hits.

Requires chipset, card, and operating system support.

Specifically optimized in Linux’ kernel > 2.6.26

This is a major optimization for parallel applications

Interrupt Coalescence

Avoid flooding the host system with too many interrupts, packets are collected and one single interrupt is generated for multiple packets.

- Not all NICs support it
- 75-100 micro-seconds timeout recommended
- Can be critical for high performance NIC (10Gb, 40Gb…)

Protocol Offloading

- Each NIC offers different level of processing offloading, if any.
- Some controllers offer TCP offloading. Not always best throughput, but can save CPU cycles.
- Support for higher level protocols (OFED, iWARP) for specific applications.
- Support for layer 2 protocols (i.e. RoCE)

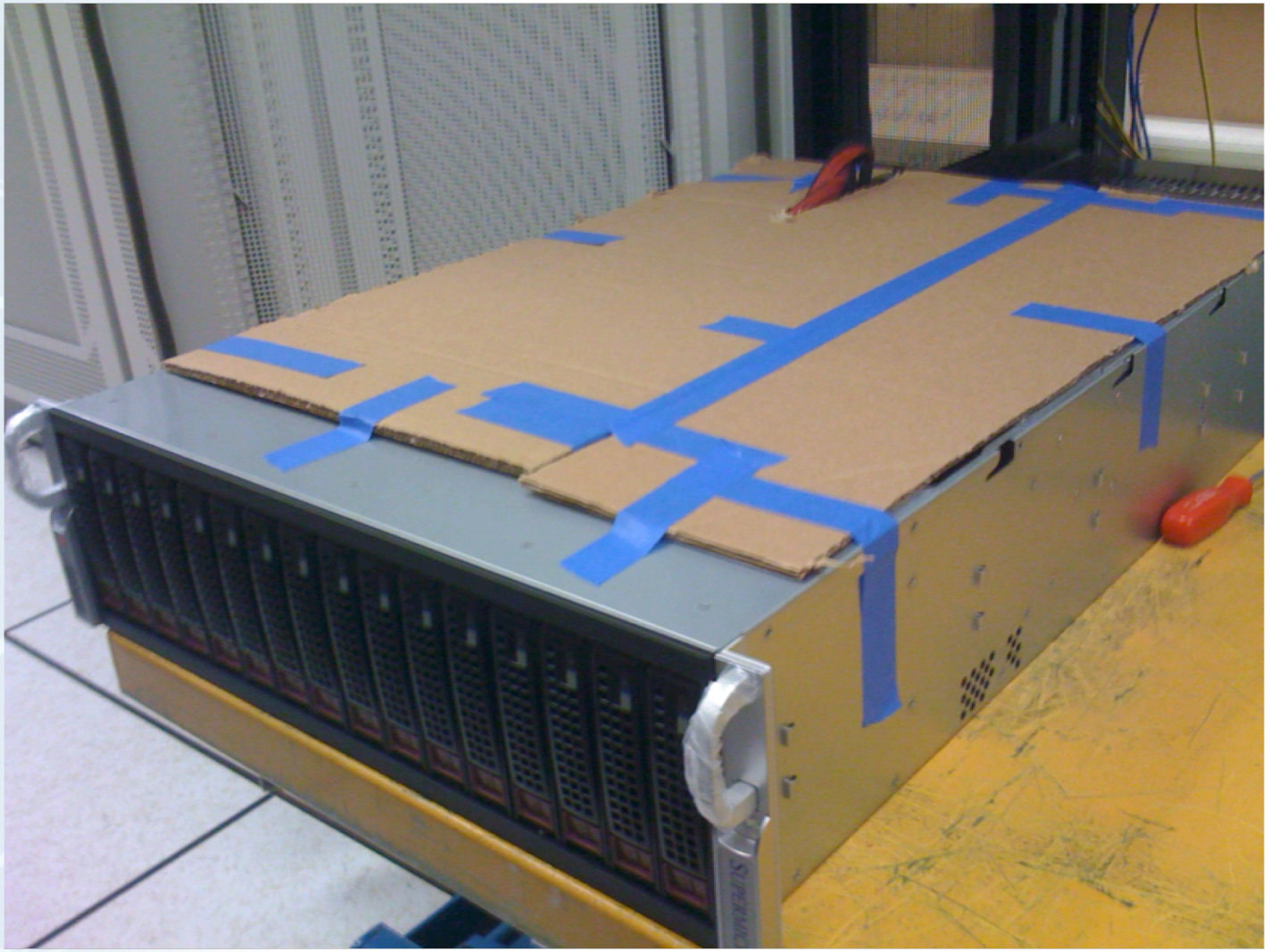
- Protocol offloading is key to achieve better performance
- Not yet mature, may cause headaches
- Requires more powerful NIC's
- PCIe 3.0 will encourage offloading (virtualization, OpenFlow ?)

Large Selection of NIC's

- Major brand names: Myricom, Intel, Chelsio, Mellanox…
- Many more obscure brands

- Depending on requirements, other features to look for:
  - Supported optics
  - Jumbo Frame support
  - MSI-X
  - Supported offloaded protocols / OFED support
  - Driver support
  - PCIe bus (8 lanes vs. 16 lanes)

Experiment



Design Process

1. Identify requirements.
2. Choose hardware.
3. Build a prototype
4. Tune until required requirements are achieved
5. Document

Identify requirements: What do you want to achieve ?

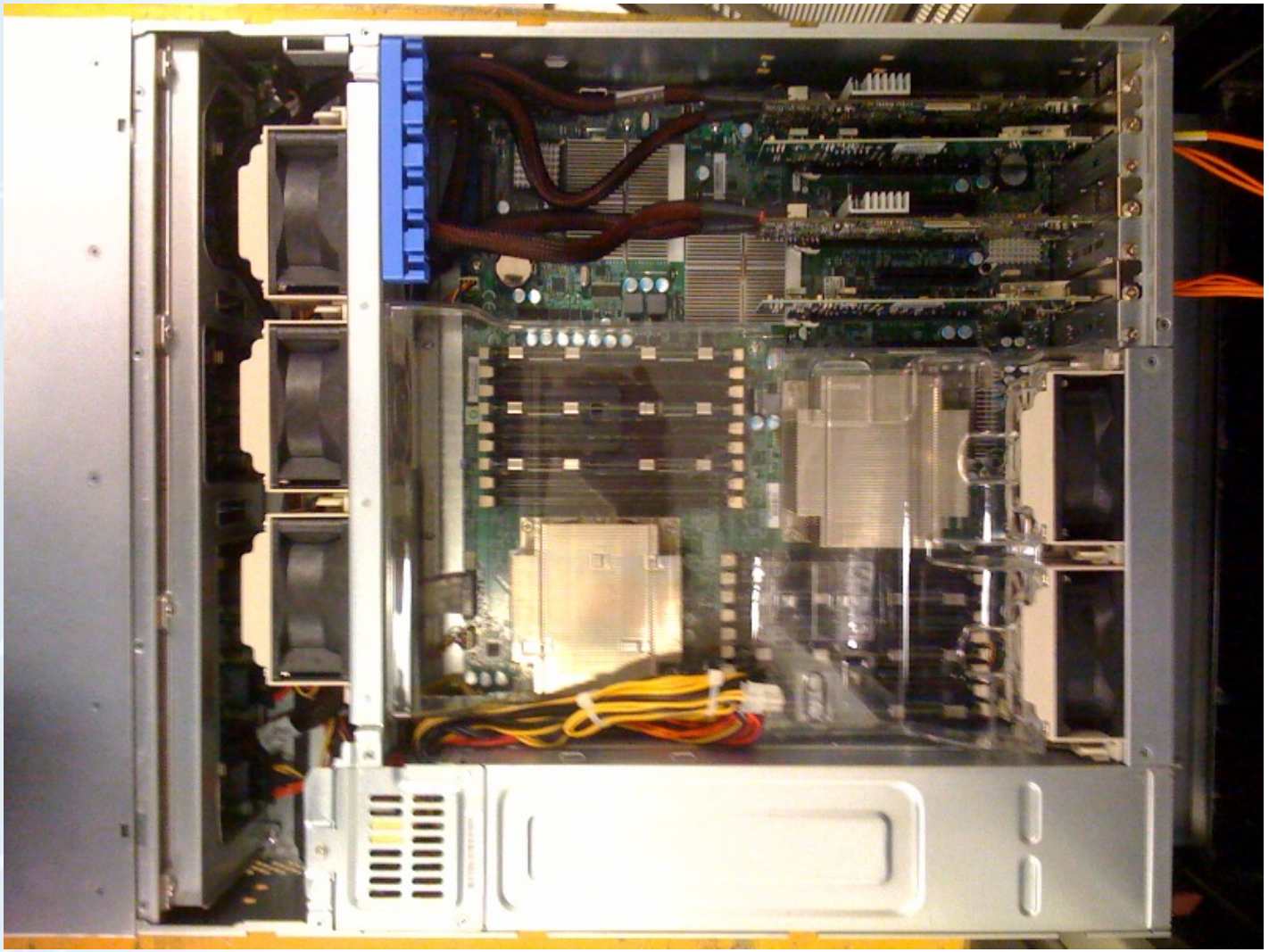
- Workload and workflow: Application driven
- Usage : heavy, low, burst
- Quantity: how many units will be deployed
- Deployment: needs to fit in the infrastructure
- Cost: prototype phase, per unit, operation

Select Hardware

- Select I/O cards and NICs based on the needs. Be conservative, plan for at least 20% more performance that what is needed.
- Select CPU architecture based on application work load.
- Select motherboard that allows all the elements to communicate at the same time at the required performance.
- Finds chassis that fit all racks and fulfills the deployment constraints (power, mount)

- Having good relationship with vendor is very useful.
- Some parts may just not behave as hoped.

Prototype



Experiment and Tune

- Measure “bare metal” performance first with test on raw devices (best work flow).
- Experiment with each element by itself initially, with work loads similar to the applications to run.
- Keep on tuning until convinced no progress can be made (keep an eye on the bare metal performance measurements)
- Write down all the experiments in a spreadsheet

Tuning I/O

- I/O performance is always limited by a single bottleneck at any given time.
- Hardware bottlenecks: disks, cpu, memory, bus
- Software bottleneck: Operating System, middleware, applications

Disk Performance Issues

Disks are mechanical data storages. Their performance depend on:

- Disk speed (Rotation Per Minute): 7,200, 10,000 or 15,000 rpm
- Geometry
- Sequential and random access (head seek)
- Sustained and Peak performance

How to build a high performance I/O subsystem:

- Partitioning (short-stroking)
  - Outer tracks of disks are the fastest
- Workflow optimization (readahead, filesystem)
- Use of caches
- More disks !

RAID and Performance

- Right RAID Level ?
- Need a better controller ?
- Need better drives ?
- Adjust strip size when possible
- Disable any “smart” controller built-in options

Tool: vmstat

From man page:

reports information about processes, memory, paging, block IO, traps, and cpu activity.

- Shown true I/O operation
- Shows CPU bottlenecks
- Shows memory usage
- Shows locks

```
$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 22751248 192800 1017000    0    0     0     0    4    7  0  0 100  0  0
```

I/O testing tool: dd

From man page: *"convert and copy a file"*

- Generate I/O traffic
- Control block size, seek length
- Input and output agnostic (raw or file)
- Can be used in parallel

$ dd if=/storage/data1/test-file1 of=/dev/null bs=4k

13631488+0 records in

13631488+0 records out

55834574848 bytes (56 GB) copied, 54.1224 seconds, 1.0 GB/s

Example of a “dd test”

```
# dd of=/dev/null if=/storage/data1/test-file1 bs=4k &
# dd of=/dev/null if=/storage/data1/test-file2 bs=4k &

# dd of=/dev/null if=/storage/data2/test-file1 bs=4k &
# dd of=/dev/null if=/storage/data2/test-file2 bs=4k &

# dd of=/dev/null if=/storage/data3/test-file1 bs=4k &
# dd of=/dev/null if=/storage/data3/test-file2 bs=4k &
```

Example vmstat / dd

vmstat 1

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache     si   so    bi    bo   in   cs   us sy id wa st
 6  0      0 150132 215204 23428260    0    0     0     0 16431 2245   0 13 86  0  0
 2  3      0 1692948 218924 21920000   0    0  4428 499712 24599 5341  1 29 65  6  0
 2  5      0 1610216 222512 22001264   0    0  3532 725012 25230 5363  0 15 75 10  0
 4  5      0 720020 224532 22865412    0    0  2048 847296 24566 4277  0 13 65 22  0
 3  7      0 1917556 225440 21686980   0    0  1672 1099036 27333 4314 0 17 60 23  0
 6  7      0 1419324 225496 22180252   0    0     0 1312704 29410 25386 0 24 45 31  0
 3  6      0 391860 225560 23182336    0    0     4 1261536 25797 27532 0 20 48 32  0
 8  4      0 80624 224672 23486864     0    0     0 1296932 26799 3373  0 22 52 26  0
 3  6      0 88860 224184 23475516     0    0     0 1322248 28338 3529  0 22 51 27  0
```

I/O Testing Tips

- Two windows, one with dd, one with vmstat
- Influence of the read and write caches
- Flush caches before running tests:

```
# echo 3 > /proc/sys/vm/drop_caches
```

- Test file size: three times the memory size to eliminate cache effects
- Influence of the block size: use block size that matches application's I/O pattern
- Remote Console (IPMI)

Linux I/O Scheduler

I/O scheduler: Choice of different policies.

- Default policy is “fair”, meaning bad for performance.
- Typically deadline scheduler is better for performance, but favors the most I/O hungry application.

In /boot/grub/grub.conf:

```
title CentOS (2.6.35.7)
root (hd0,0)
kernel /vmlinuz-2.6.35.7 ro root=/dev/VolGroup00/
LogVol00 rhgb quiet elevator=deadline
initrd /initrd-2.6.35.7.img
```

I/O Tuning: readahead

Useful optimization when workload is mostly sequential read

Need to play with to find good value for your host

Does not always play nice with hardware optimization (but is often better than hardware optimization)

Needs to be set at boot time (e.g.: in /etc/rc.local)

Interesting reading
http://www.kernel.org/doc/ols/2004/ols2004v2-pages-105-116.pdf

```
/sbin/blockdev --setra 262144 /dev/sdb
/sbin/blockdev --setra 262144 /dev/sdc
/sbin/blockdev --setra 262144 /dev/sdd
```

File Systems Performance

- Very few file systems are designed for high performance
- EXT4 is currently the fastest production file system for Linux.
- ZFS provides “smart” software RAID and compression on Solaris
- BTRFS: bleeding edge, integrates RAID and compression on Linux
- File systems must be tuned for performance
- Compromise performance versus data reliability: be careful for what you ask for !

File System Optimization (1)

File System independent optimization (in /etc/fstab)

```
/dev/sdb1 /storage/data1 ext4     noatime,nodiratime 0 0
```

File System specific optimization (EXT4)

```
/dev/sdb1 /storage/data1 ext4 inode_readahead_blks=64,
  data=writeback,nobh,barrier=0,commit=300,noatime,nodiratime 0 0
```

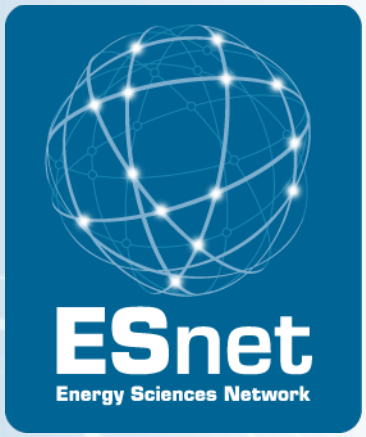
Inode_readahead: useful when directories have lots of files

Data=writeback: metadata is written onto the disk in a “lazy” mode

barrier=0: does no longer enforce journal write ordering.

File System Optimization (2)

- Necessary in order to get performance close to bare metal
- Be careful what you ask for: some of the optimization may render the file system less reliable in case of crashes



Network / TCP Tuning

TCP Autotuning Settings: http://fasterdata.es.net/TCP-Tuning/

Linux 2.6: add to /etc/sysctl.conf

net.core.rmem_max = 16777216

net.core.wmem_max = 16777216

autotuning min, default, and max number of bytes to use

net.ipv4.tcp_rmem = 4096 87380 16777216

net.ipv4.tcp_wmem = 4096 65536 16777216

FreeBSD 7.0+: add to /etc/sysctl.conf

net.inet.tcp.sendbuf_max=16777216

net.inet.tcp.recvbuf_max=16777216

Mac OSX: add to /etc/sysctl.conf

```
kern.ipc.maxsockbuf=16777216
net.inet.tcp.sendspace=8388608
net.inet.tcp.recvspace=8388608
```

Windows XP

- use “DrTCP” (http://www.dslreports.com/drtcp/) to modify registry settings to increase TCP buffers

Windows Vista/Windows 7: autotunes by default, 16M Buffers

Selecting TCP Congestion Control in Linux

To determine current configuration:

- sysctl -a | grep congestion
- net.ipv4.tcp_congestion_control = cubic
- net.ipv4.tcp_available_congestion_control = cubic reno

Use /etc/sysctl.conf to set to any available congested congestion control.

Supported options (may need to enabled by default in your kernel):

- CUBIC, BIC, HTCP, HSTCP, STCP, LTCP, more..
- E.g.: Centos 5.5 includes these:
  - CUBIC, HSTCP, HTCP, HYBLA, STCP, VEGAS, VENO, Westwood

Use modprobe to add:

- /sbin/modprobe tcp_htcp
- /sbin/modprobe tcp_cubic

Additional Host Tuning for Linux

Linux by default caches ssthresh, so one transfer with lots of congestion will throttle future transfers. To turn that off set:

```
net.ipv4.tcp_no_metrics_save = 1
```

Also should change this for 10GE

```
net.core.netdev_max_backlog = 250000
```

Warning on Large MTUs:

- If you have configured your Linux host to use 9K MTUs, but the MTU discovery reduces this to 1500 byte packets, then you actually need 9/1.5 = 6 times more buffer space in order to fill the pipe.
  - Some device drivers only allocate memory in power of two sizes, so you may even need 16/1.5 = 11 times more buffer space!

NIC Tuning (See: http://fasterdata.es.net)

Defaults are usually fine for 1GE, but 10GE often requires additional tuning

Intel e1000 (often integrated into motherboards) the driver defaults to 256 Rx and 256 Tx descriptors. Chipset now supports 4096.

```
ethtool -G rx 4096 tx 4096
```

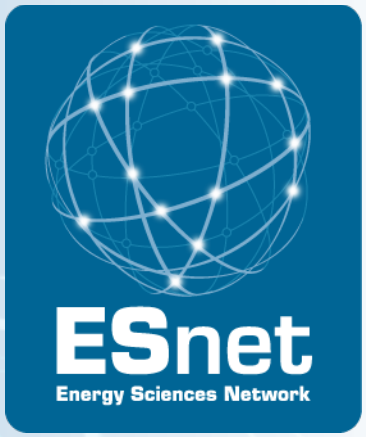
Myricom 10Gig NIC

- Up to 2x increase by increasing the interrupt coalescing timer:

```
ethtool -C interface rx-usecs 75
```

Chelsio 10Gig NIC

- TCP Segmentation Offloading (TSO) and TCP Offload Engine (TOE) on the Chelsio NIC radically hurt performance on a WAN (they do help reduce CPU load without affecting throughput on a LAN).
- To turn off TSO do this: `ethtool -K interface tso off`



Advanced Topics

Remote DMA (RDMA)
OSCARS: On-demand Layer 2 circuits

Remote Direct Memory Access (RDMA)

Bypass CPU

Low latency: one-sided protocol (decreases network overhead)

Out of band synchronization

Zero copy transmission (send & receive memory in place)

Initially designed for HPC (i.e. LAN).

Requires hardware support

Not integrated in mainstream software stacks (but supported in Linux, Solaris, Windows and OSX as 3rd party package)

Applications must be modified to support RDMA

new API, not socket-based

RDMA and Wide Area Networks

Motivation:

- Leverage existing LAN technologies
- Shared memory model (classic in HPC)
- Data can be too large to be transferred

Issues, limitations:

- May not behave well on WAN (latency, packet loss, packet ordering)
- May require specific network QoS – assumes no packet loss with no reordering
- Requires specific kernels

iWARP: RDMA over TCP

- IETF standard for RDMA over TCP
- Benefits: offloads TCP onto the network controller (TOE) – needed for zero-copy receive. (Chelsio, Intel controllers)
- Supports NFS, SDP, SRP and iSCSI.
- Available in Linux with Open Fabric Alliance (OFED)

- Works well in LAN environments
- Shows poor performance on WAN: on long distance, TCP can be the bottleneck.

RoCE: RDMA over Converged Ethernet

- Layer 2: avoids TCP all together
- Available in Linux with OFED
- Supports all Infiniband protocols: SDP, SRP, MPI, MPI-IO
- Requires hardware support (Mellanox)

- No TCP = no sophisticated congestion protocol.
- Very sensitive to packet loss and packet reordering

Virtual Circuits

- Virtual circuits provide a virtual link between two points in a network
- Typically traffic engineering is applied (reserved bandwidth, specific network path)
  - Needed to avoid packet loss
- Harder when there are multi-domains / multi-organizations
- Very powerful – two hosts on different continents can appear to each other as if they are directly connected, with a guarantee of service
- Drivers: scientific collaboration, cloud computing, security
- Limited deployments: no commercial services.

OSCARS: Layer 2 virtual circuit on demand

- Driven by scientific collaboration (LHC, JGI,…)
- End to End across multi-domains
- Provides reservation and scheduling services
- Middleware: can be integrated into applications

- Deployed and in operation
- Requires participating domain to deploy the service

More Information

ESnet's Data Transfer Node:

http://fasterdata.es.net/fasterdata/data-transfer-node/

Other ESnet Tutorials, including our "Bulk Data Transfer" tutorial

http://fasterdata.es.net/fasterdata/learn-more/

OSCARS

http://code.google.com/p/oscars-idc/

Email: lomax@es.net