



Systems & Technology Group

Parallel File Systems from the Application Programmer Perspective (Part 1)

Scott Denham

sdenham@us.ibm.com

IT Architect – Industrial Sector

IBM Deep Computing

Agenda

- **Some basic assumptions**
- **Components of the I/O subsystem**
- **The I/O Stack**
- **Parallelism In I/O**
- **GPFS – a parallel file system**
- **Performance considerations**
- **Performance analysis**

Basic Conceptual Disk I/O

```
DO I=1,NVEC  
WRITE(5)  
VEC(1,I)  
...
```

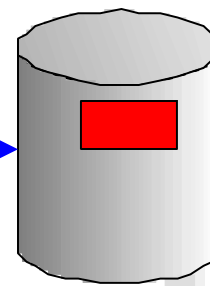


A computer

Some data



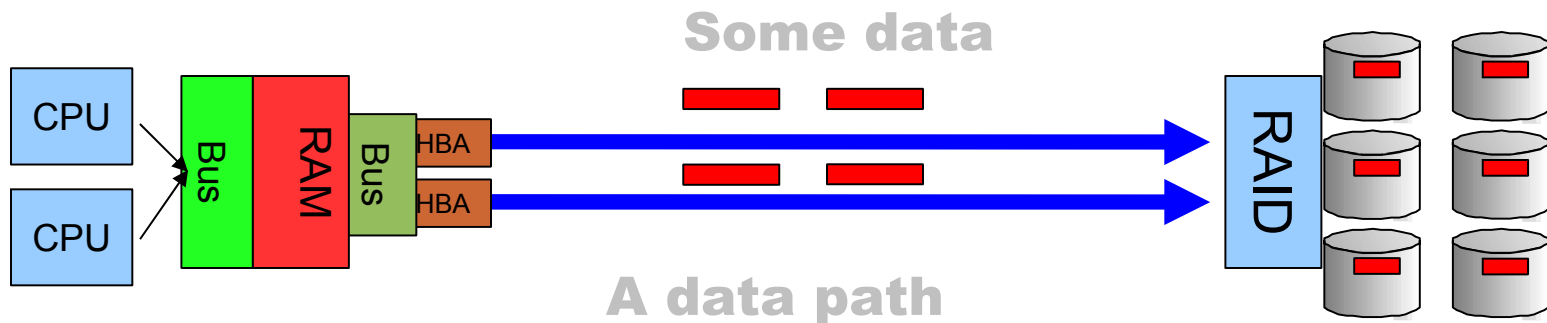
A data path



**A storage device
“(disk)”**

Or is it???

```
DO I=1,NVEC
WRITE(5)
VEC(1,I)
...
```



A computer

- Frontside bus
- Cache
- PCI Bus
- Interrupt routing
- Controller paths
- ...

- Redundant pathing
- Data speed
- Latency
- Congestion
- Zoning
- Bus Arbitration
- ...

A storage device “(disk)”

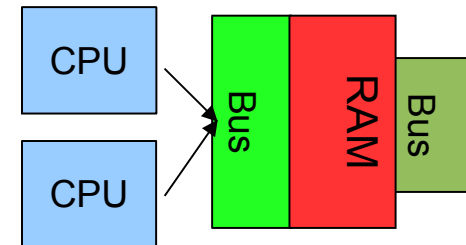
- Controller cache
- Device cache
- Striping
- Redundant Parity
- ...

Some challenges emerge

- **Systems are becoming increasingly complex**
 - Clusters / Grids
 - Large scale parallelism (Blue Gene)
 - Multicore processors (Power6, Clovertown, Niagara ...)
 - Heterogeneous systems (CellBE, GP-GPU, FPGA)
- **Technology elements shift at different rates.**
 - Step changes in processor technology as feature size shrinks
 - Interconnects are more constrained by the physics of distance
 - Disks quickly grow denser, but not proportionally faster
- **Some awareness of the underlying hardware and O/S infrastructure can lead to better performance.**

Components of the I/O subsystem

- **The Processor** (under control of the application)
 - Operates on or generates the data in main memory
 - Initiates transfer through some form of IO statement
 - Eventually must wait for the I/O operation to complete



- **The Operating System**

- Almost universally “owns” the I/O components and enforces order
 - Whose data is it anyway?
 - Where is it located on the disk?
 - How does it get there?
 - When is a physical operation required?
- May move the data from the application's memory space to make I/O seem to be complete, or to condition it for transfer to the device, or to insure that it does not change before the operation
- May attempt to help by guessing what will happen next, or remembering what happened last
- Deals with unexpected conditions or errors
- Maintains (we hope!) some record of activity



Components of the I/O subsystem

- **The I/O Adapter (AKA “Channels, I/O Bus, HBA, HCA ...)**
 - Copies data between a location in main memory and a specific bus, E.G.
 - SCSI (Parallel)
 - SAS (Serial-attached-SCSI)
 - PATA / SATA (PC Heritage)
 - Fibre Channel
 - Reports back to the OS when the operation is complete
 - May contain memory for an intermediate copy of the data
 - May be able to work with the disks to sustain multiple operations simultaneously
 - Adjustable queue depths
 - “Elevator Seek”, queue reordering



Components of the I/O subsystem



▪ The Disk Drive

- Single disks have an integrated control function and attach directly to the bus
- Most physical disks store data in units of 512 bytes. Best performance occurs when I/O operations are for an aligned number of full sectors.
- Commonly described in terms of “heads”, “cylinders”, although the physical hardware no longer matches the logical geometry.
- Modern disks include several MB of memory cache that allows them to pre-fetch, coalesce multiple smaller operations into a single one, and return recently accessed data without reading it from the spindle again.
- Write cache involves risk; a power failure in the middle of an operation can lead to corrupted data or data in an unknown state. Generally disabled in server-class drives.

Components of the I/O subsystem

▪ The Disk Subsystem

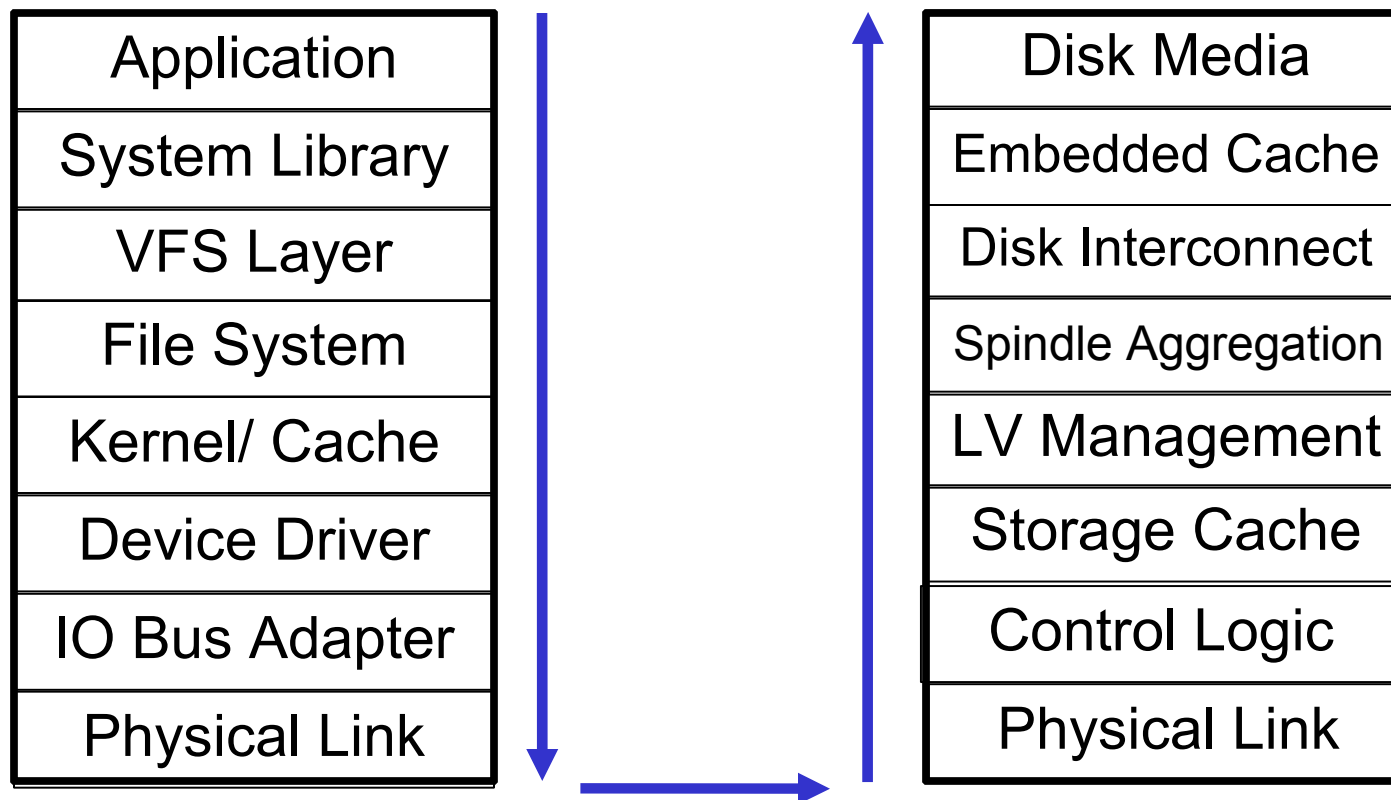
- More complex disk systems create virtual “disks” or logical units (LUN)s from larger ones, using various RAID technologies.
- Controller may include substantial (GB's) of cache to improve access to smaller, repeatedly used files.
- May include block-layer functions like snapshot and replication
- Often can present the same collection of disks to multiple hosts simultaneously, but... SHARED DISK != SHARED DATA!



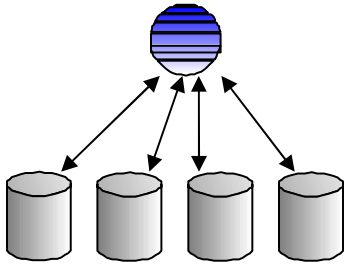
Components of the I/O subsystem

- **The File System** - Without some form of order, a disk device or subsystem is just a stream of bytes, which can be addressed on sector (512) byte boundaries.
 - Structural information defines containers for specific data collections; files, directories, etc.
 - Metadata information defines characteristics of files and directories; ownership, permissions, creation and access times, etc.
 - Allocation of raw blocks to files is best not done first-come-first-served, which would lead to excessive fragmentation.
 - Requests for resources must be coordinated by the OS to prevent two applications from claiming the same block.
 - Filesystems may include advanced functions like journalling, file level snapshots, Information Lifecycle Management, etc.
 - Most modern OS's provide a distinct filesystem layer API, which allows various non-native file systems to be added seamlessly.
 - Filesystems are often optimized for specific objectives or environments:
 - Streaming media (large files, predominantly sequential access, reuse)
 - E-mail / newsgroups (many small files)

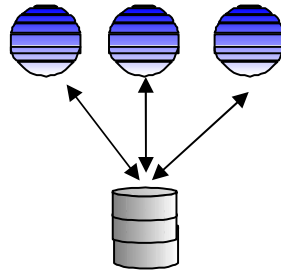
The I/O Stack



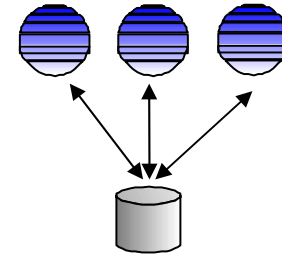
Parallelism can take on many forms



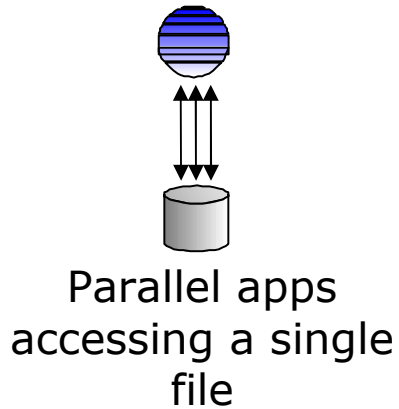
File system on striped device



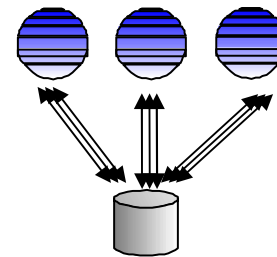
Multiple hosts sharing partitioned storage



Multiple hosts sharing common file system



Parallel apps accessing a single file

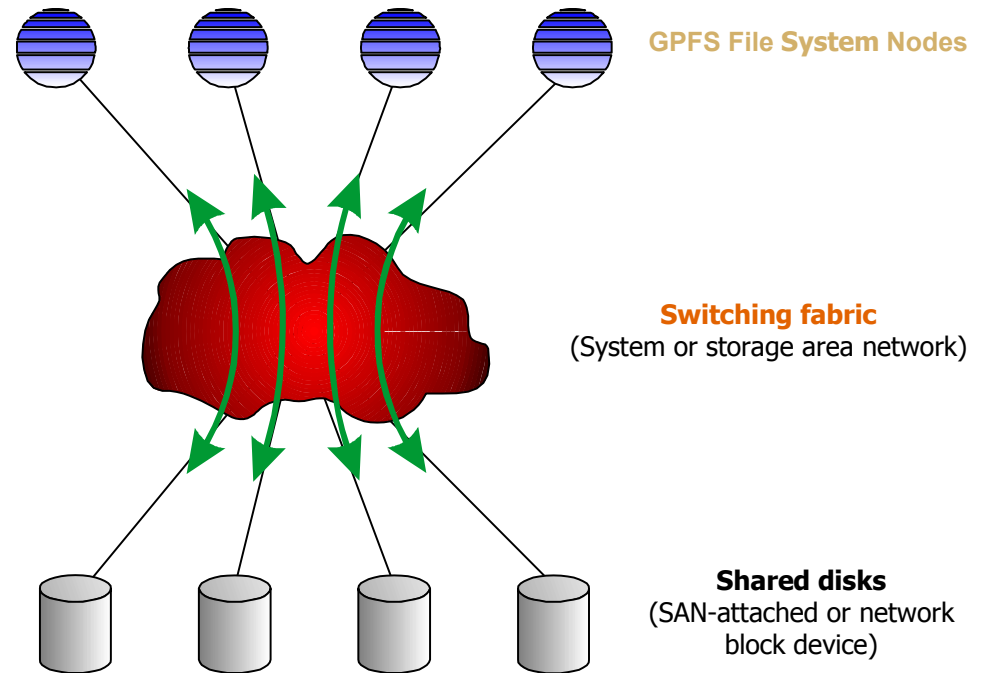


Parallel Apps on Multiple hosts accessing a single file

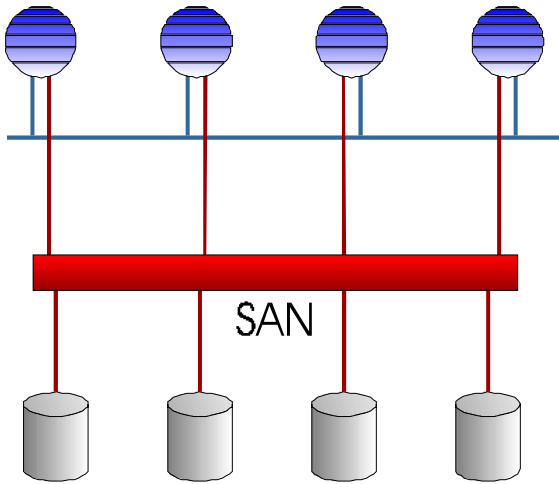
What is GPFS?

Parallel File System for Cluster Computers Based on Shared Disk (SAN) Model

- *Cluster* – a collection of fabric-interconnected nodes (IP, SAN, ...)
- *Shared disk* - all data and metadata on fabric-attached disk
- *Parallel* - data and metadata flows from all of the nodes to all of the disks in parallel under control of distributed lock manager.

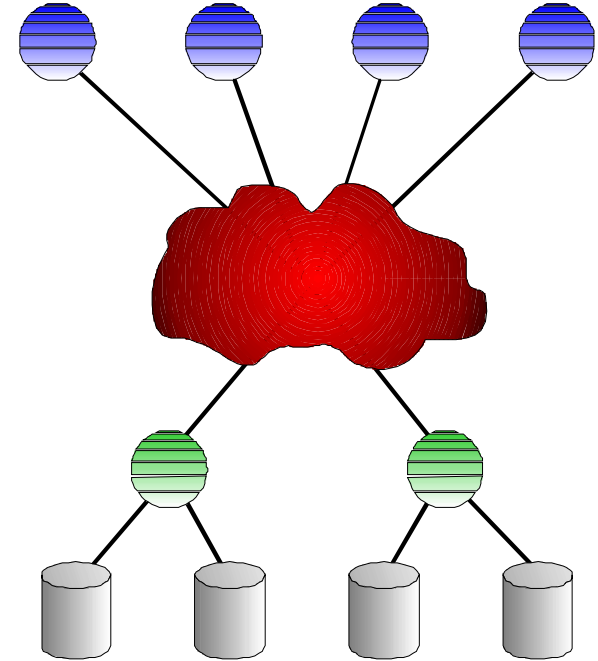


GPFS Configuration Examples



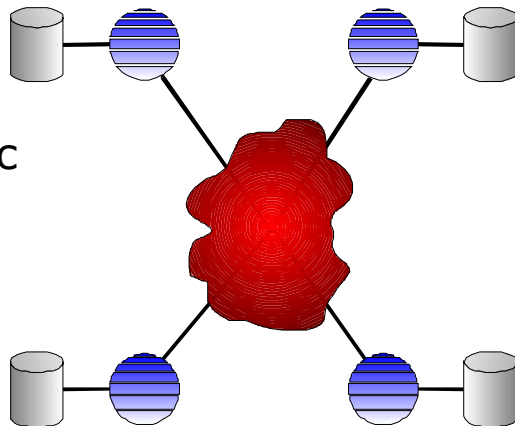
Storage Area Network

Fibre Channel,
iSCSI



Cluster with dedicated I/O
(block server) nodes

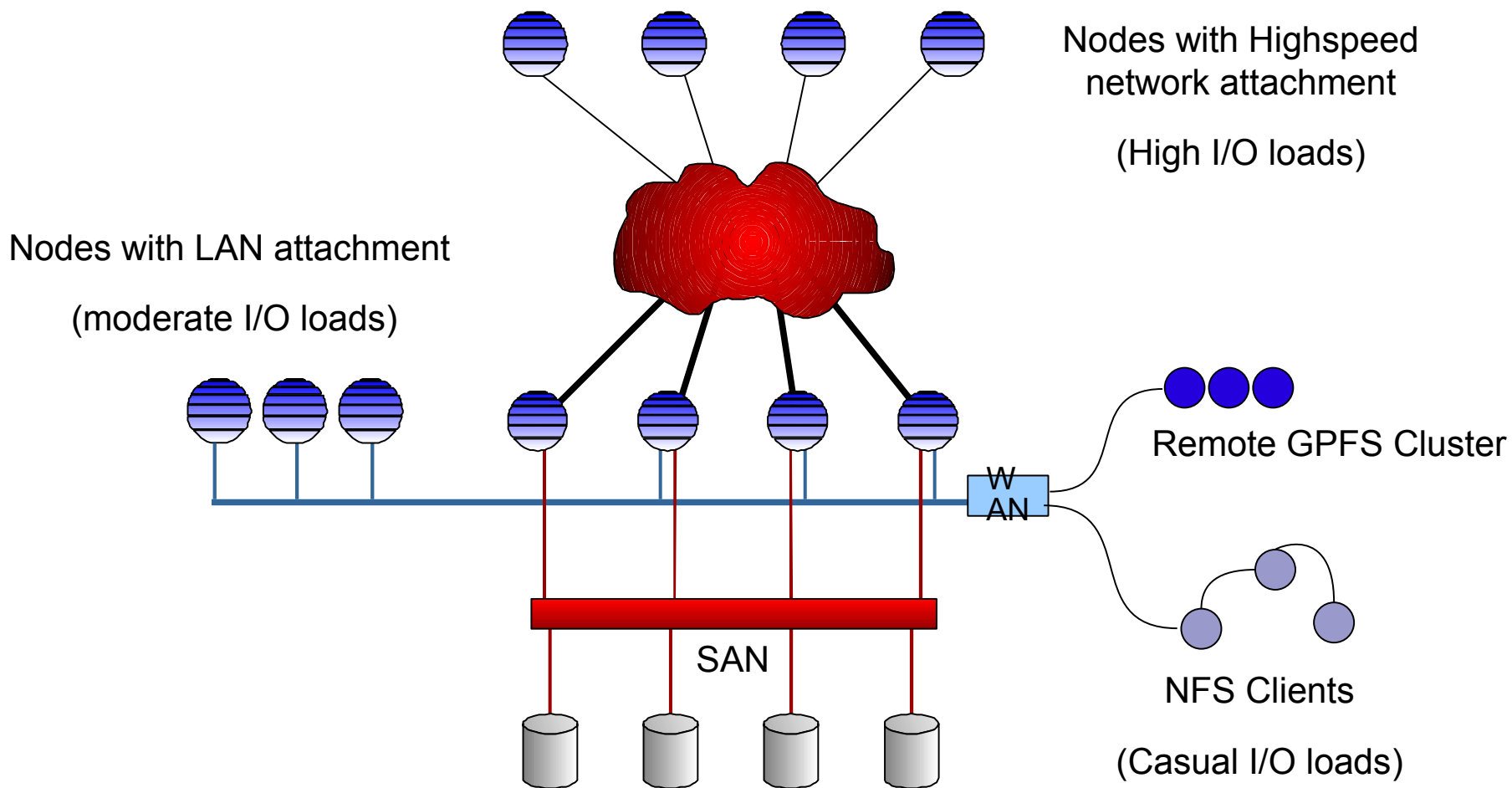
Symmetric cluster



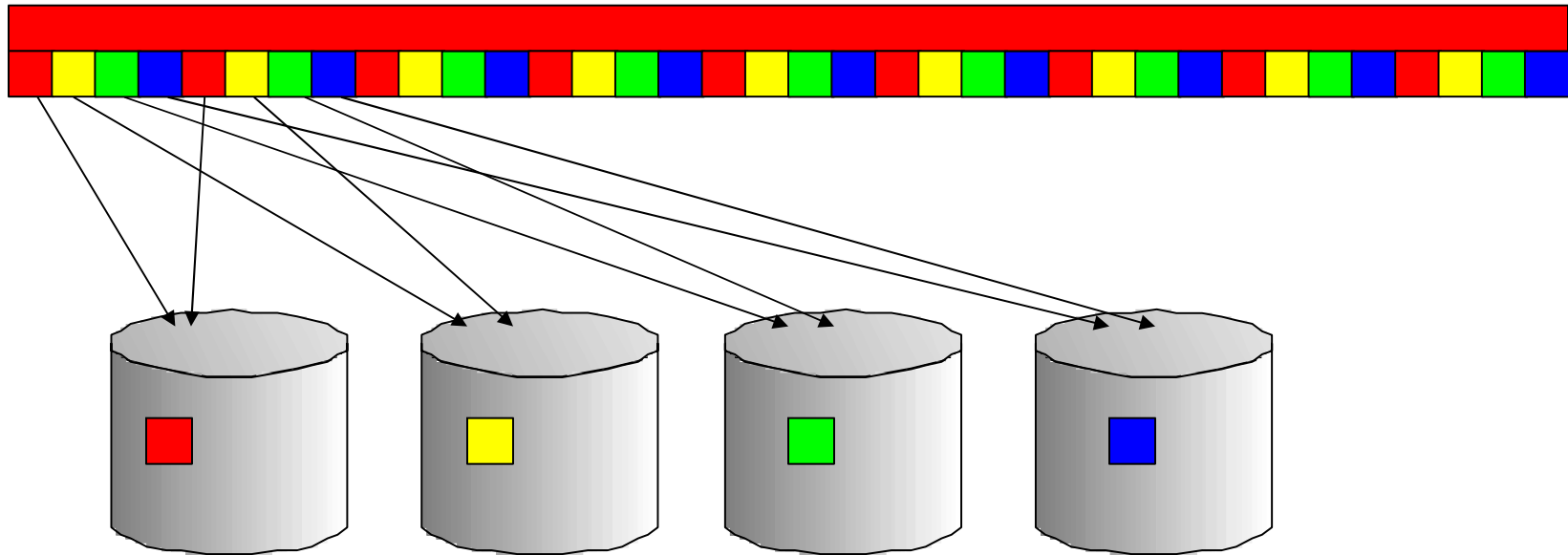
Software Shared Disk

◆ NSD (GPFS internal)

GPFS Configuration Examples



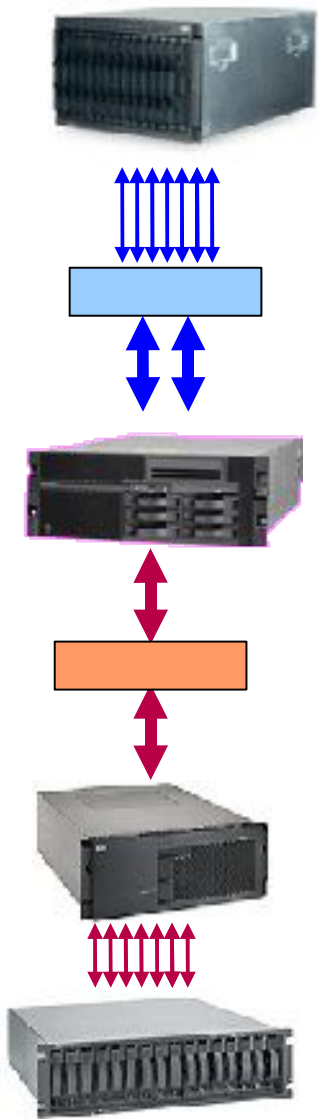
Parallel Block distribution



Some important factors:

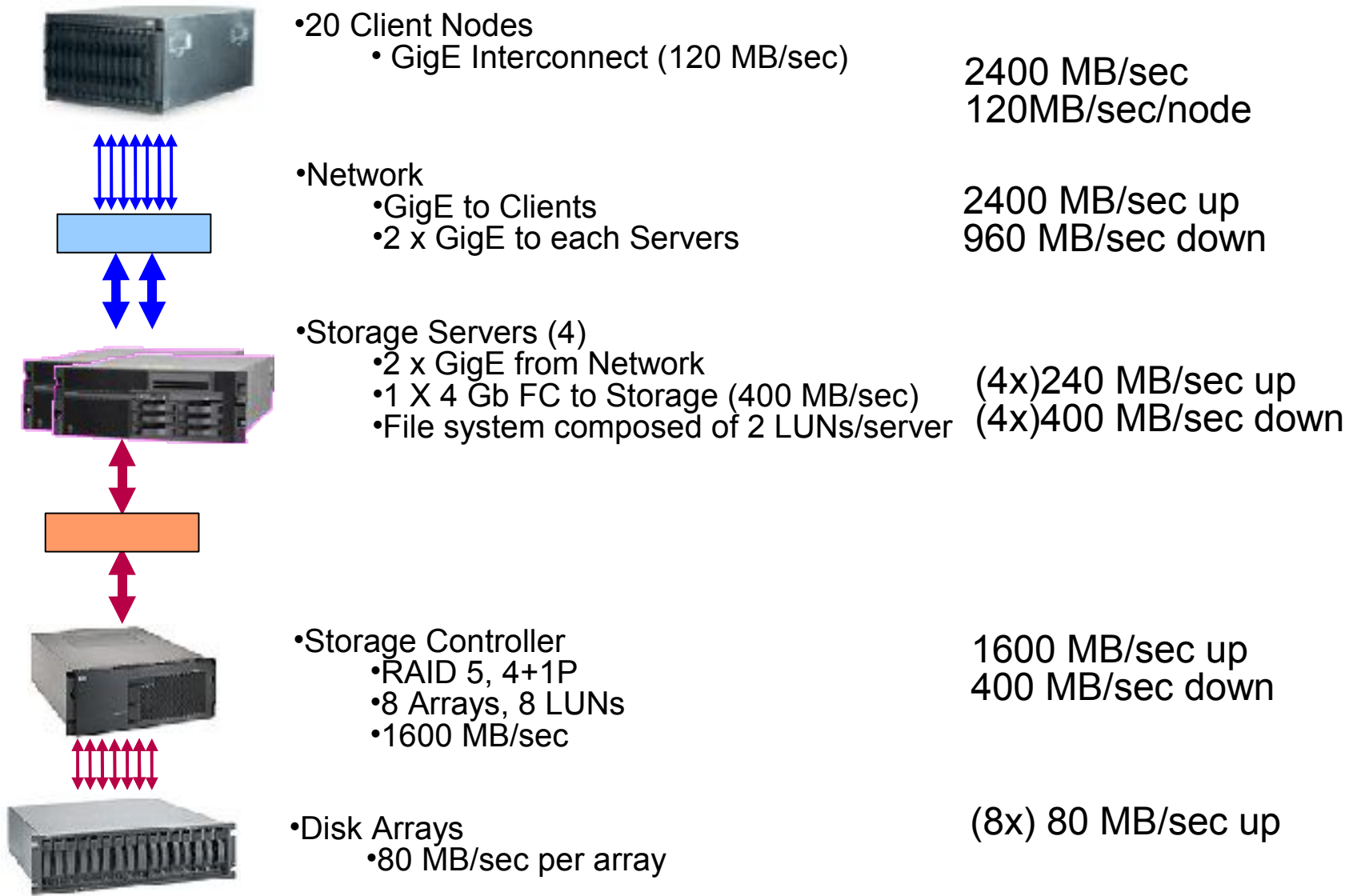
- Block Size – The units into which the file I/O is divided
 - Does it fit nicely on the disk hardware? (sector size, stripe size, RAID?)
 - Does it move easily through the S/W and H/W stack? Network?
 - Is it appropriate for the application?
 - This is generally the minimum unit of transfer. Too large = waste!
 - What are the natural sizes in the application?
- Access Pattern
 - In the example above, a stride of 4 results in all I/O going to 1 disk
 - If random, pre-fetch techniques may hurt more than help
 - Look for ways to be sequential

Potential Performance Bottlenecks

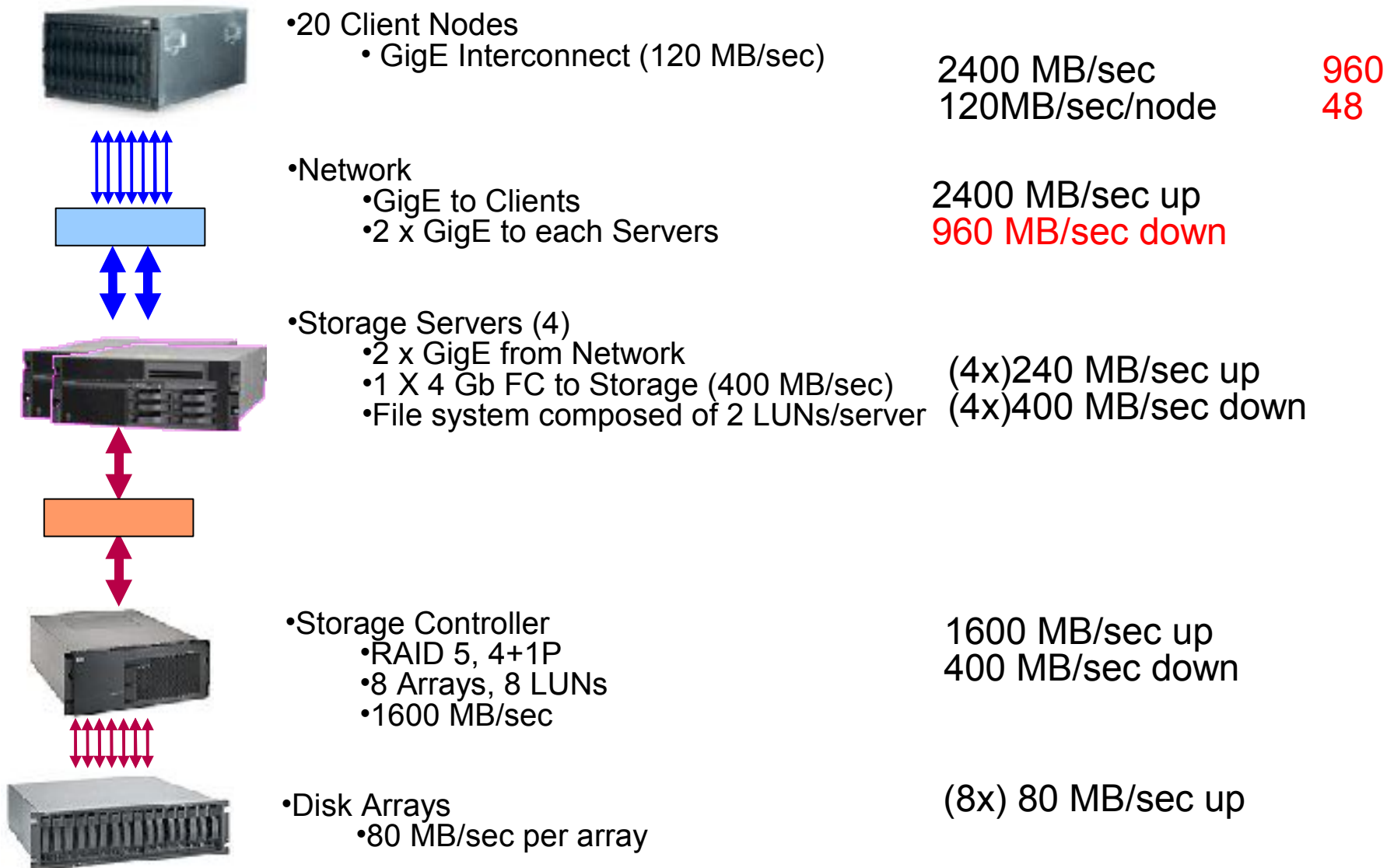


- Client nodes
 - CPU capacity
 - Application I/O request structure
 - PCI Bus Bandwidth
 - Network Tuning
- Network
 - Bandwidth
 - Topology
 - Latency
- Storage Server
 - CPU capacity
 - Memory
 - Disk Attachment
- Storage Fabric
 - Bandwidth
 - Topology
 - Disk Attachment
- Storage Controller
 - RAID Configuration – Class, stripe size
 - Cache
 - LUN Distribution
- Disk Arrays
 - Individual Disk speed and interface
 - Topology

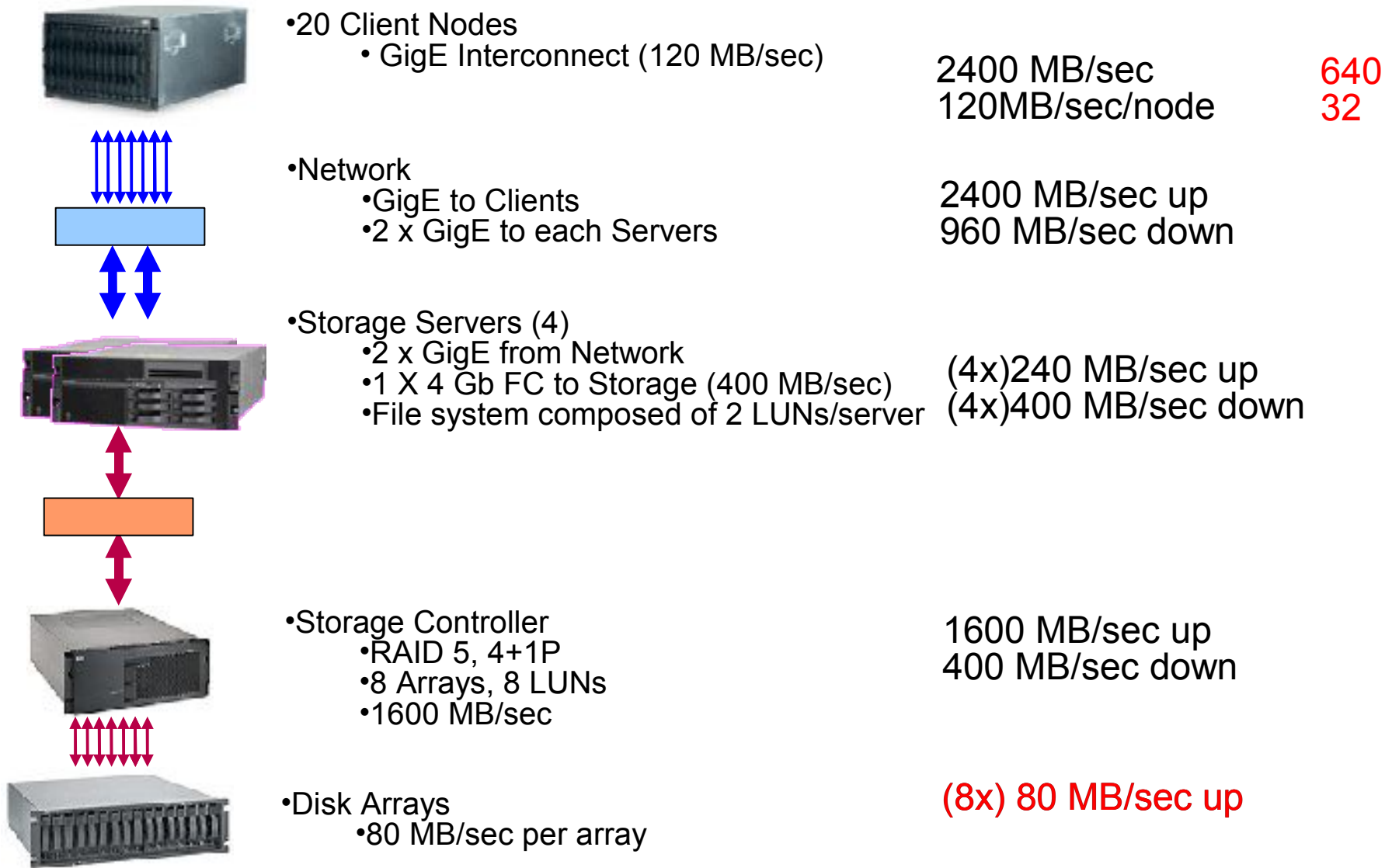
Performance bottleneck example (expectation)



Performance bottleneck example (network constraint)

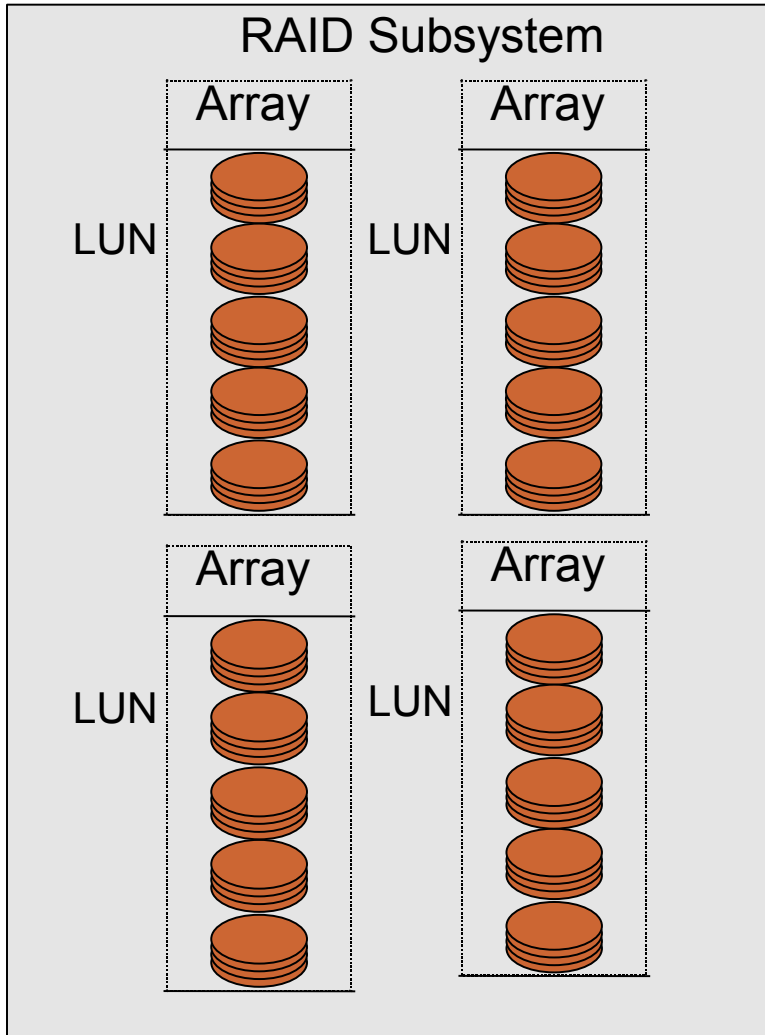


Performance bottleneck example (LUN constraint)

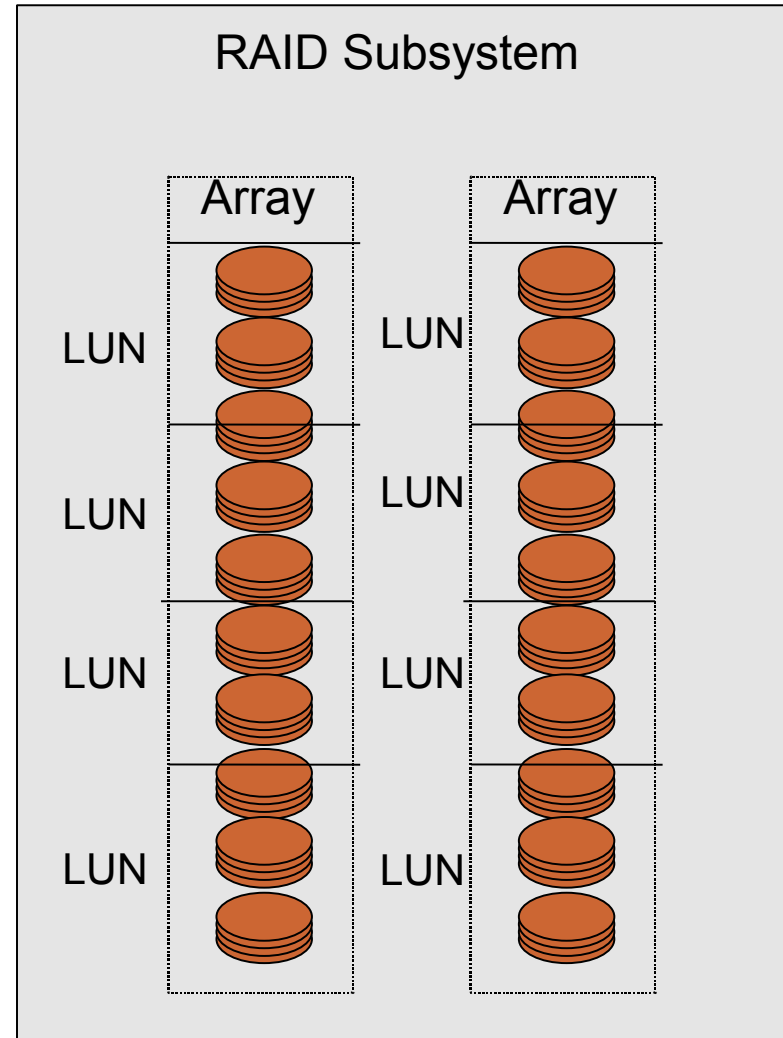


Performance bottleneck example

Striping is good.



Plaid is bad!



General Performance Methodology

- **Understand the Application requirement**
 - Request sizes, access patterns
 - Is it realistic for the available hardware? Assume 100% efficiency
- **Consider the objectives**
 - Do all clients read at once?
 - Average rate per client vs peak rate at one client
 - Read vs Write ratios
 - Write is almost always slower; cache and prefetch can't help much.
 - Write cache can help, but consider the risk to data AND METADATA.
- **Consider each layer of the stack**
 - Measure independently where possible
- **Non-linearity**
 - Congestion, especially in the network layers can lead to drastic decreases in throughput due to dropped packets, retransmission, etc.
- **“... when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind.” (Lord Kelvin, 1824-1907)**
 - “The system feels slower than it did last year”
 - “This crash analysis ran 20% longer than it did last year with the same data”

Some helpful tools for performance analysis

■ **lostat -k 60 2**

- First set of numbers is cumulative since boot, and often uninteresting
- Second set reflects events of the last 60 seconds
 - Are LUNS in a parallel file system balanced?
 - (Bytes transferred / sec) / (transfers per second) \approx transaction size



Linux

```
avg-cpu:  %user   %nice   %sys %iowait  %idle
           0.42    0.00    9.44   56.50   33.63
```

Device:	tps	kB_read/s	kB_wrtn/s	kB_read	kB_wrtn
sda	0.52	0.00	5.73	0	344
sdb	40.21	39233.53	6661.11	2354404	399733
sdc	0.00	0.00	0.00	0	0
sdd	0.00	0.00	0.00	0	0
sde	0.00	0.00	0.00	0	0
sdf	36.58	38422.72	6580.90	2305747	394920
sdg	0.00	0.00	0.00	0	0
sdh	0.55	0.15	0.12	9	7
sdi	0.03	0.00	0.02	0	1
sdj	3.73	326.57	342.60	19597	20559
sdk	0.00	0.00	0.00	0	0
sdl	0.00	0.00	0.00	0	0
sdm	0.00	0.00	0.00	0	0
sdn	9.40	327.15	363.31	19632	21802
sdo	0.00	0.00	0.00	0	0

OS Disk

GPFS 1
~900KB/xferGPFS 2
~35-80KB/xfer

Some helpful tools for performance analysis



lostat on AIX 5

```

tty:          tin          tout      avg-cpu: % user % sys % idle % iowait
              0.1          33.0
  
```

```

Disks:      % tm_act      Kbps      tps      Kb_read      Kb_wrtn
hdisk7      0.0          0.0      0.0          0          0
hdisk4      0.0          0.0      0.0          0          0
hdisk10     0.0          0.0      0.0          0          0
hdisk6      0.0          0.0      0.0          0          0
hdisk8      0.0          0.0      0.0          0          0
hdisk5      0.0          0.0      0.0          0          0
hdisk9      0.0          0.0      0.0          0          0
hdisk0      0.1          0.3      0.1          0          20
hdisk3      0.0          0.0      0.0          0          0
hdisk1      0.0          0.4      0.1          4          20
hdisk2      0.0          0.0      0.0          0          0
hdisk13     0.0          0.0      0.0          0          0
hdisk14     0.0          0.0      0.0          0          0
hdisk11     0.0          0.0      0.0          0          0
hdisk15     0.0          0.0      0.0          0          0
hdisk12     0.0          0.0      0.0          0          0
hdisk17     0.0          0.0      0.0          0          0
hdisk16     0.0          0.0      0.0          0          0
cd0         0.0          0.0      0.0          0          0
  
```

OS Disk(s)

Some helpful tools for performance analysis



■ Nmon

- IBM casual tool, freely available from developerworks:
 - <http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmon>
- Combines features of “top”, “iostat”, “netstat”, “ifconfig”, etc
- Available for AIX and both ppc64 and x86 versions of mainstream Linux
- Includes data capture and a post processing tool “nmon analyzer”
- Example: the 'n' (network) and 'a' (adapter) views

```

┌--nmon-----r=Resources-----Host=gandalf-----Refresh=4secs---14:51.39
└─┘
| Network-----|
|I/F Name Recv=KB/s Trans=KB/s packin packout insize outsize Peak->Recv Trans |
|   en0      0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0      0.0 |
|   en1      0.0      0.1      0.2      0.5     92.0    107.0     0.0      0.3      0.3 |
|   lo0      0.1      0.1      1.0      1.0     98.0     98.0     0.1      0.1      0.1 |
| Total      0.0      0.0 in Mbytes/second |
|I/F Name  MTU  ierror  oerror  collision Mbits/s Description |
|   en0    1500    0      0      0      10 Standard Ethernet Network Interface |
|   en1    1500    0      3      0     1024 Standard Ethernet Network Interface |
|   lo0   16896    0      0      0      0 Loopback Network Interface |
|-----|
| Adapter-I/O -----|
|Name          %busy      read      write      xfers Disks Adapter-Type |
|ssa0           0.0      0.0      0.0 KB/s      0.0 16 IBM SSA 160 SerialRAI |
|ide0           0.0      0.0      0.0 KB/s      0.0 1 ATA/IDE Controller De |
|sisscsia0     100.0  15996.5  0.0 KB/s    3999.1 4 PCI-X Dual Channel Ul |
|TOTALS  4 adapters  15996.5  0.0 KB/s    3999.1 28 TOTAL(MB/s)=15.6 |
|-----|

```


Some helpful tools for performance analysis

- **Unix “dd” command from raw devices (as root)**
 - “time dd if=/dev/sdf of=/dev/null bs=1M count=1000”
 - Bypasses most OS function to measure hardware.
 - Safe for read. Can only write to an unused disk or array!
- **Network performance tests**
 - Iperf
 - netperf
- **File system specific tools**
 - GPFS – mmfsadm dump waiters (as root; use mmfsadm with caution!)

```
# mmfsadm dump waiters
0x4070D050 waiting 0.051998887 seconds, NSD I/O Worker: for I/O completion on disk sdf
0x40709A20 waiting 0.132997154 seconds, NSD I/O Worker: for I/O completion on disk sdf
0x40702DC0 waiting 0.039999144 seconds, NSD I/O Worker: for I/O completion on disk sdf
0x40701BB0 waiting 0.117997475 seconds, NSD I/O Worker: for I/O completion on disk sdf
0x407009A0 waiting 0.008999808 seconds, NSD I/O Worker: for I/O completion on disk sdb
0x406F8B30 waiting 0.069998502 seconds, NSD I/O Worker: for I/O completion on disk sdf
```

Some cautions

- **Don't predict I/O performance using the results of small files**
 - Read caching takes place at multiple levels. Many Unix and Linux filesystem implementations can draw upon any unused host memory to cache previously read (or written) data. Thus
 - “dd if=/dev/zero of=/work/testfile count=1000 bs=32K”
 - “time if=/work/testfile of=/dev/null bs=32k”
 - This may be OK if the test reflects the application requirement, but consider also the effect of many parallel tasks or jobs.
 - Cache characteristics can be quite different for different file systems. Some are cache tunable, others are largely not.
- **Don't focus on a single element of the I/O stack.**
 - Squeezing 10% more out of your I/O controller will not help if you are bound at the network layer.
- **Consider parallel effects**
 - Scaling may fall off badly from the $N \times$ single stream ideal
- **Resist the temptation to turn all the knobs at once!**
 - You may fix it and not know why
 - You may improve one area and degrade another, leading you to think neither change had any effect.
- **Don't forget what you have done.**
 - Take notes