ECE590-04
Engineering Robust Server Software

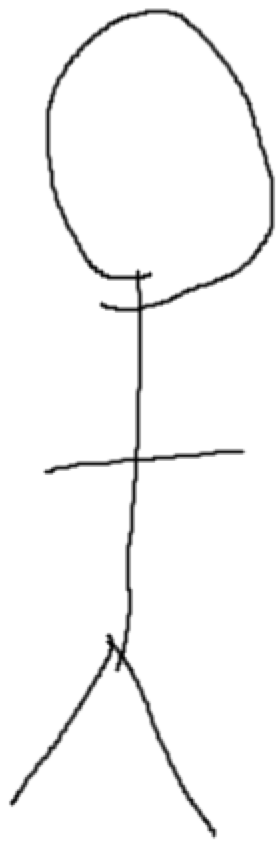
Spring 2018

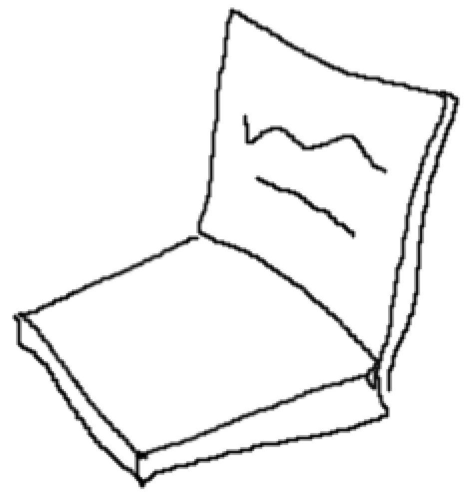
IO Performance and Scalability

Tyler Bletsch
Duke University

Basics of large-scale storage systems and their performance

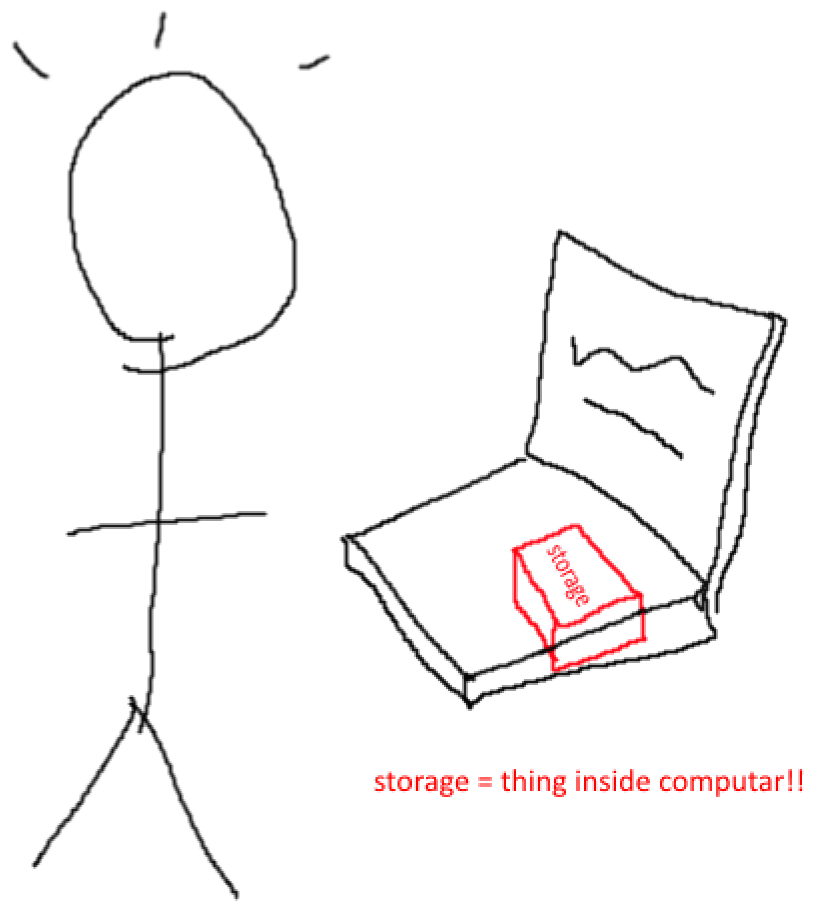
Average person's view of storage





storage = computar does it !!

Average engineer's view of storage



Performance and Scalability

- What is **performance**?
  - Speed of IO operations
  - Measured in **IO Operations Per Second (IOPS)** or **throughput (MB/s)**

- Key insight: We can increase performance (and capacity) by adding drives!
  - How? Recall from ECE550 that RAID allows the use of multiple drives in a way that maintains reliability with aggregating performance

- So we can just keep adding drives forever, right?
  - Well, eventually the case is full or you're out of ports...

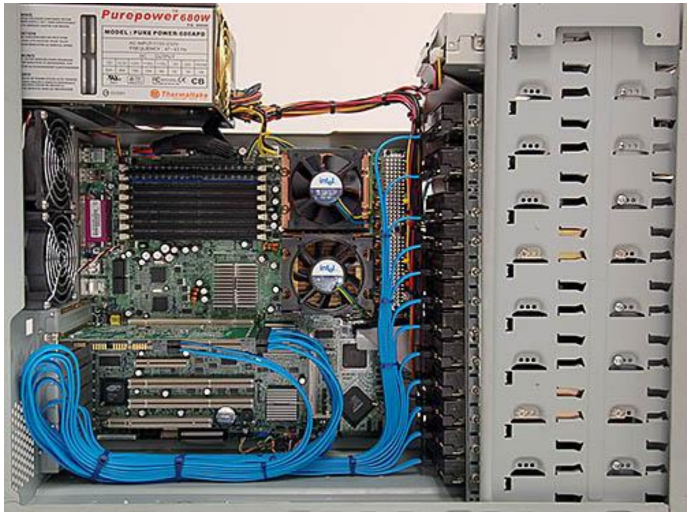


Image from http://s57.photobucket.com/user/junfeng/media/mediaserver2.jpg.html

Performance and Scalability

- What is **scalability**?
  - A measure of how big can you grow a system

- <u>Hardware example:</u>
  a traditional desktop PC has a fixed number of drive bays and drive connections
- <u>Software example:</u>
  a database may use a data structure whose performance becomes unacceptable at a certain number of records (e.g. if it relies on a linear search)

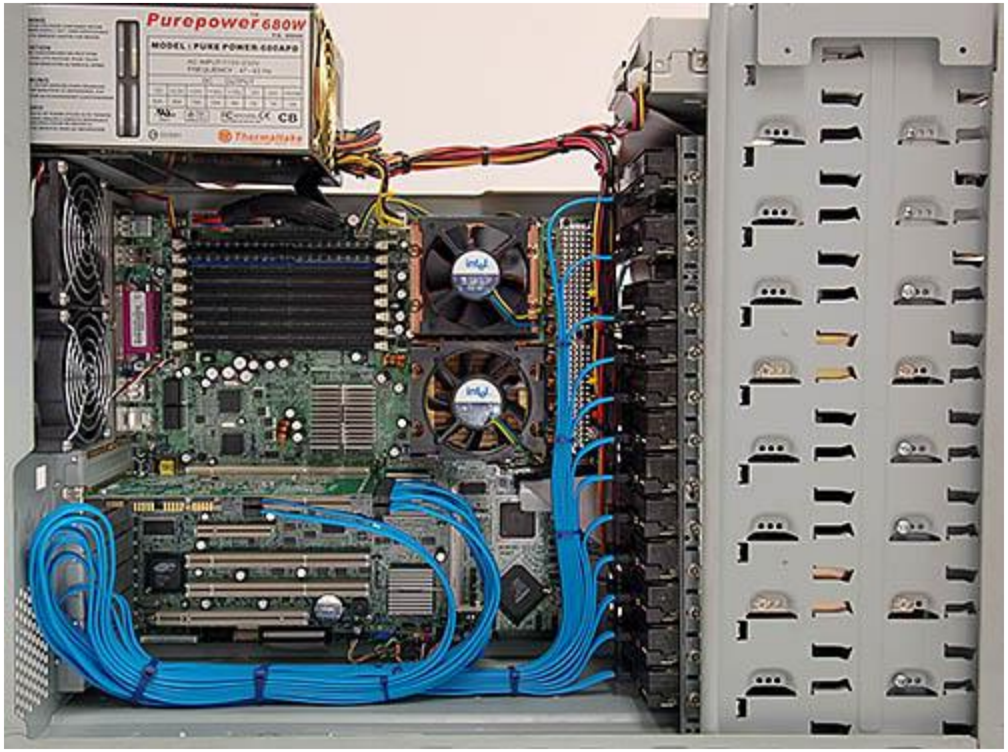
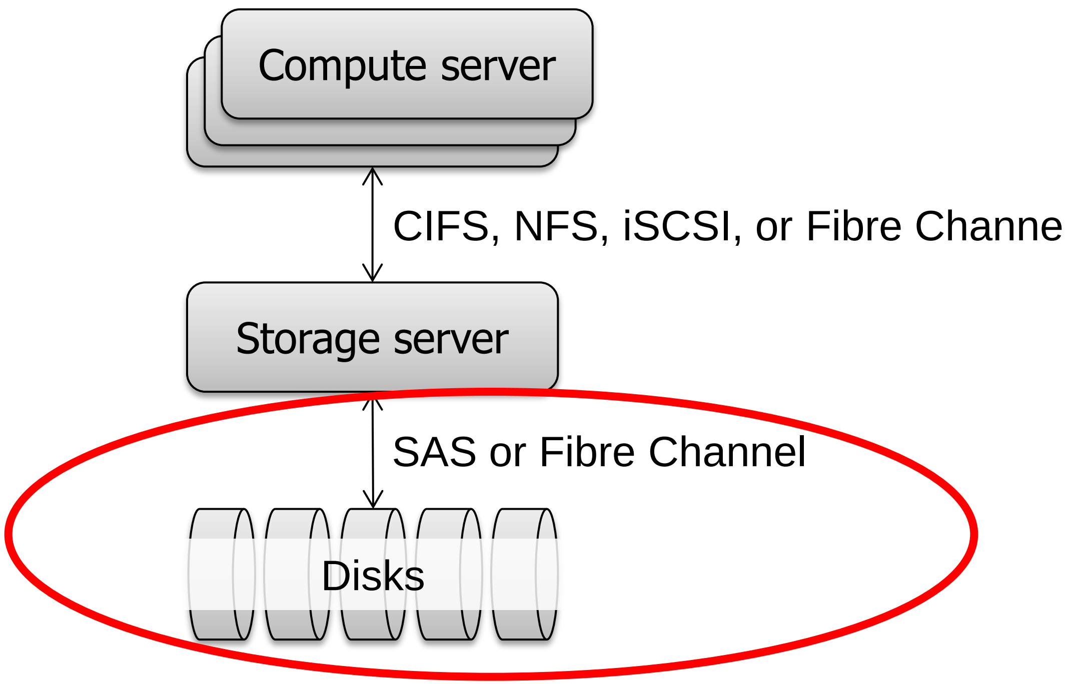


Image from http://s57.photobucket.com/user/junfeng/media/mediaserver2.jpg.html

Storage in enterprise-scale environments

- On a laptop/desktop, you have Directly Attached Storage (DAS)
  - Poor scalability!
- Storage at large scale is separated from the compute servers



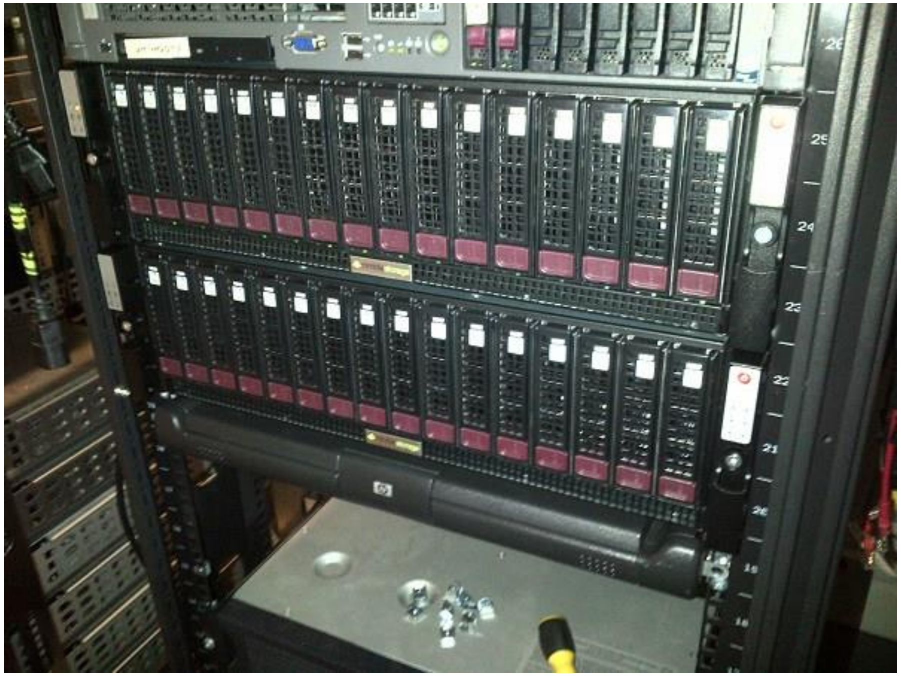
High disk density

- Disks installed in enclosures separate from storage servers, connected with high speed bus (SAS or Fibre Channel)

A disk shelf with 24 3.5-inch drives

From http://imn.de/product/show/1377/ds4246-ds4246-shelf

Two disk shelves and a storage server in a rack

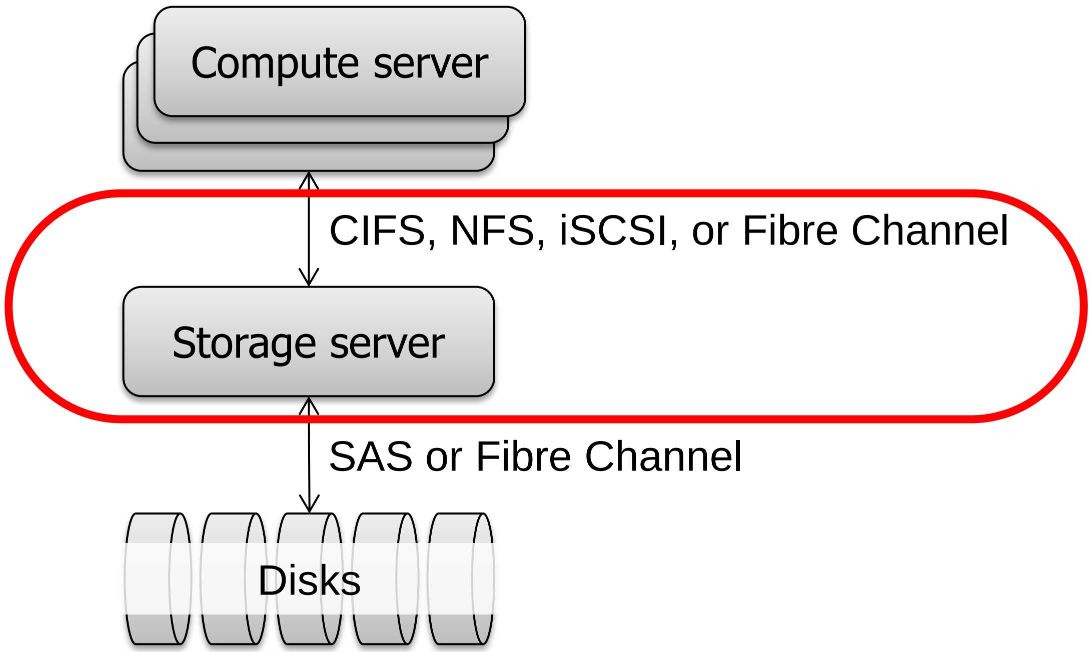


From http://www.talking-it.com/2014/storage/nimble/installing-a-nimble-es1-expansion-shelf-hard-bit/

- Result: Vastly improved **scalability** of physical storage

Storage in enterprise-scale environments

- Moving up the stack...



The storage server

- Storage servers aggregate the disks and apply RAID
  - *They also do a LOT of other things to enhance storage!*
    See my Enterprise Storage Architecture course to learn more...

- How to connect the aggregated storage to compute servers?
  Two methodologies:
  - **SAN: Storage Area Network**
    - Storage server divides storage into "virtual block devices"
    - Clients make "read block"/"write block" requests just like to a hard drive, but they go to the storage server
    - Examples: Fibre Channel and iSCSI
  - **NAS: Network-Attached Storage**
    - Storage server runs a file system to create abstraction of files/directories
    - Clients make open/close/read/write requests just like to the OS's local file system
    - Examples: NFS and CIFS (also known as 'Windows shares')

The story so far

- **Performance** is how many IOPS or MB/s
- Can be increased by adding drives
- Every system has limits to its ability to grow; the *ability* to *scale* up a system is referred to as **scalability**
- Scalability of physical storage is the amount we can add drives to a system
  - Direct-attach systems have poor physical storage scalability; NAS and SAN schemes are much better in this regard

- Two questions:
  - How <u>else</u> could we influence performance?
  - How much performance do we <u>need</u>?

Influencing IO performance

- Quantity of disks (as stated previously)

- Access pattern
  - If on HDDs, performance is driven by seek time:
    better to read few big chunks of data than many small ones

- Caching effects
  - Can we add caches (RAM or SSD cache) or to server or client?

- Type of media
  - Higher RPM HDDs
  - Solid State Drives (SSDs) – flash solid-state storage

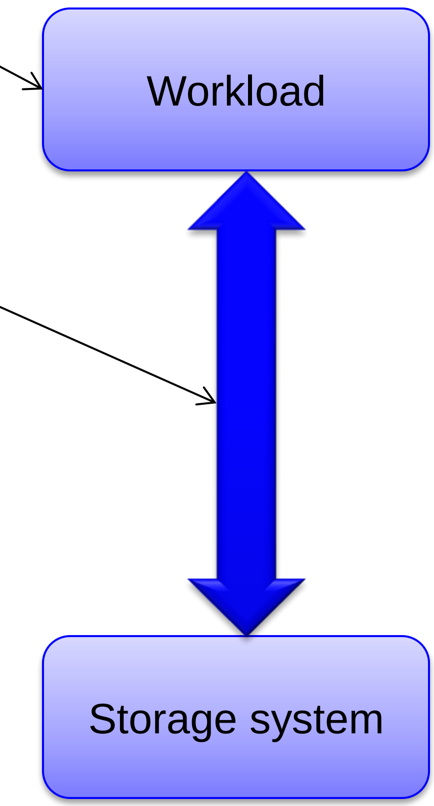
How to size storage systems

The problem

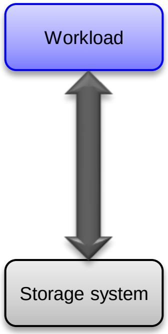
- **Workload characterization**: Determining the IO pattern of an application (or suite of applications)
  - We do so by measuring it, known as **workload profiling**
- **Storage sizing**: Determining how much hardware you need to serve a given application (or suite of applications)
- The challenge of characterization and sizing
  - Storage is a complex system!
  - Danger: high penalty for underestimating needs...

Two kinds of metrics

- Inherent **access pattern metrics**:
  - Based on the code
- Resulting **performance metrics**:
  - The performance observed when those access patterns hit the storage system
- Sometimes difficult to separate:
  - Common one that’s hard to tell: *IOPS*
  - Did we see 50 IOPS because the workload only made that many requests, or because the storage system could only respond that fast?
  - Was storage system mostly idle? Then IOPS was limited by workload.

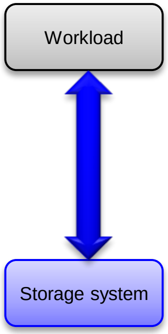


Access pattern metrics



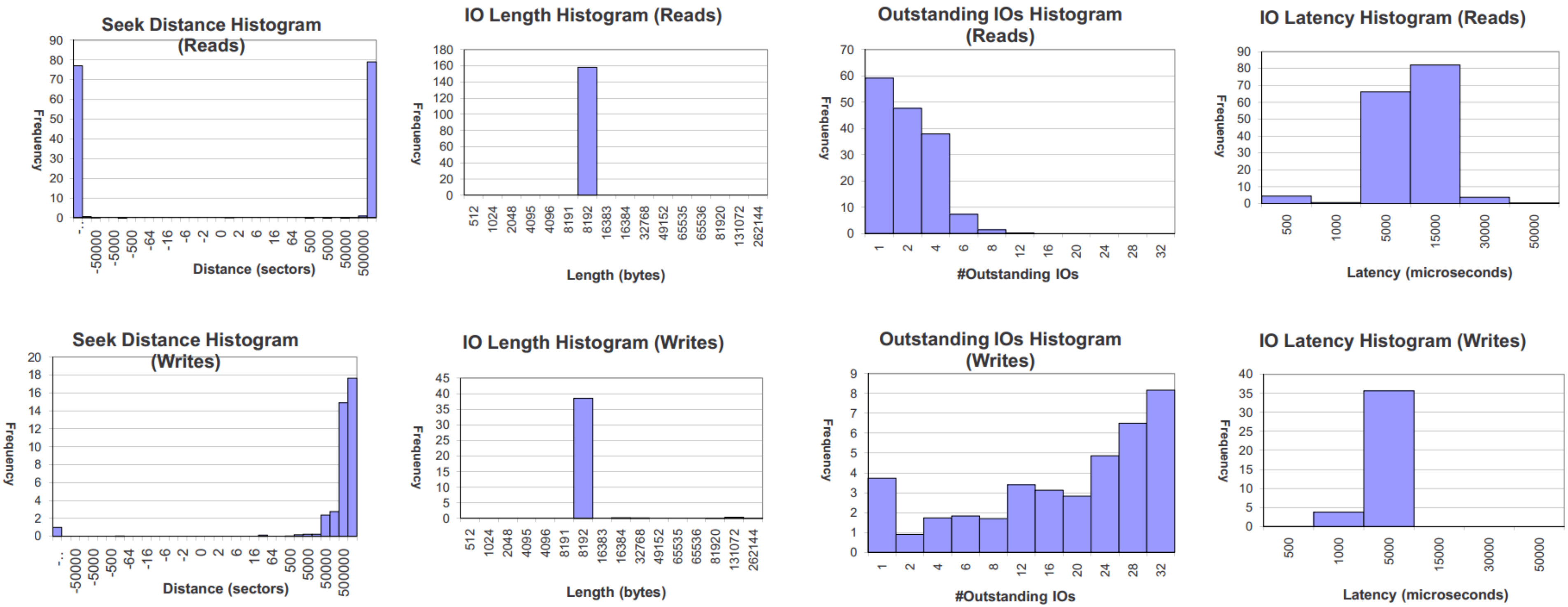
- Random vs. sequential IO
  - Often expressed as random%
  - Alternatives: average distance, seek distance histogram, etc.
- IO size
- IOPS
  - If controller/disk utilization was low, then IOPS represent storage *demand* (the rate the app asked for)
  - Alternative metric: inter-arrival time (average, histogram, etc.)
- Reads vs. writes
  - Often expressed as read%
  - May also split all of the above by read vs. write (read access pattern often different from write pattern)
- Breaking down application: can we identify separate threads?
  - Is it 50% random, or is there one 100% random thread and one 100% sequential thread?

Performance metrics



- IOPS (if storage system was bottleneck)
  - Alternative metric: IO latency (average, histogram, etc.)
  - Alternative metric: throughput (for sequential workloads)
- Queue length: number of IO operations outstanding at a time
  - A measure of IO parallelism

Example of metrics



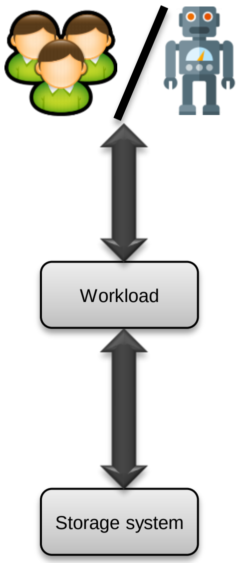
- Metrics for “DVDStore”, a web store benchmark.
  - Random workload (seek distance ≠ 0)
  - IO size = 8k
  - Short read queue, long write queue
  - Reasonable latency (within usual seek time)
  - Seek distance for writes is biased positive (likely due to asynchronous write flushing doing writes in positive order to minimize write seek distance)

From “Storage Workload Characterization and Consolidation in Virtualized Environments” by Ajay Gulati, Chethan Kumar, Irfan Ahmad. In VPACT 2009.

How to get these metrics?

- **Profiling:** *Run* the workload and *measure*
- Two problems:
  1. How to "run"?
     - Most workloads interact with users
       - Need user behavior to get realistic access pattern!
     - Where to get users?
       - App already in production? Use actual users
       - If not, fake it: **synthetic load generation** (extra program pretends to be users)
       - What about so-called **benchmarks**?

  2. How to "measure"? We'll see in a bit...

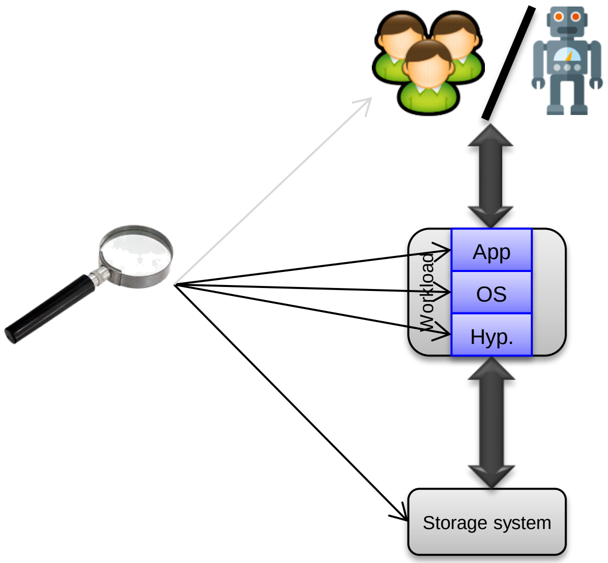


Benchmarks: How to "run"

- **Benchmark**: a program used to generate load in order to measure resulting performance. Various types:
  - **The application itself**: You literally run the real app with a synthetic load generator.
    - Example: Microsoft Exchange plus LoadGen
  - **Application-equivalent**: Implements a realistic task from scratch, often with synthetic load generation built in.
    - Example: DVDStore, an Oracle benchmark that literally implements a web-based DVD store.
  - **Task simulator**: Generate an access pattern commonly associated with a certain *type* of workload
    - Example: Swingbench DSS, which generates database requests consistent with computing long-running reports
  - **Synthetic benchmark**: Generate a mix of load with a specific pattern
    - Example: IOZone, which runs a block device at a given random%, read%, IO size, etc.

Methods of profiling: How to "measure"

- App instrumentation
  - Requires code changes
- Kernel instrumentation
  - Can hook at system call level (e.g. strace) or block IO level (e.g. blktrace).
  - Can also do arbitrary kernel instrumentation, hook anything (e.g., systemtap)
- Hypervisor instrumentation
  - Hypervisor sees all I/O by definition
  - Example: vscsiStats in VMware ESX
- Storage controller instrumentation
  - Use built-in performance counters
  - Basically this is kernel instrumentation on the storage controller kernel
- User-level metrics (e.g. latency to load an email)
  - These don't directly help understand storage performance, but they *are* the metrics that users actually care about



Sizing

- Now we know how workload acts;
  need to decide how much storage gear we need to buy

- Will present basic rules, but there are complicating factors:
  - Effects of storage efficiency features?
  - Effects of various caches?
  - CPU needs of the storage controller?
  - Result when multiple workloads are combined on one system?

- Real-world sizing of enterprise workloads:
  - For commercial apps, ask the vendor – companies with big, expensive, scalable apps have sizing teams that write sizing guides, tools, etc.
  - On the storage system side, ask the system vendor – companies with big, expensive, scalable storage systems have sizing teams too.

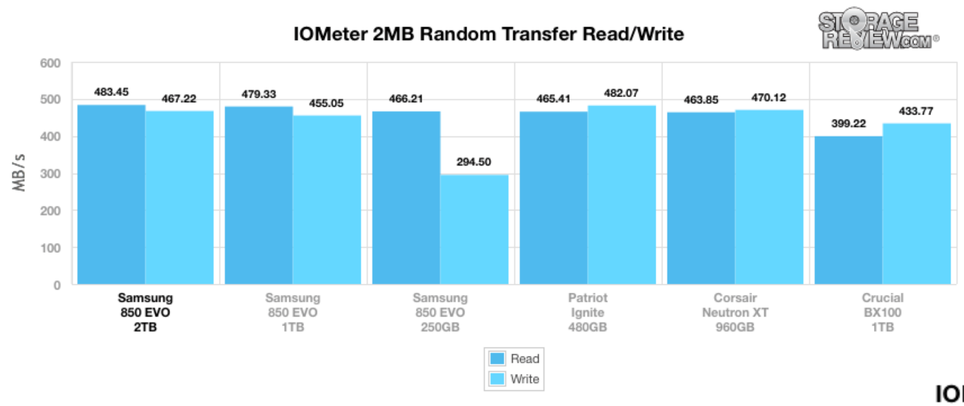
Disk array sizing

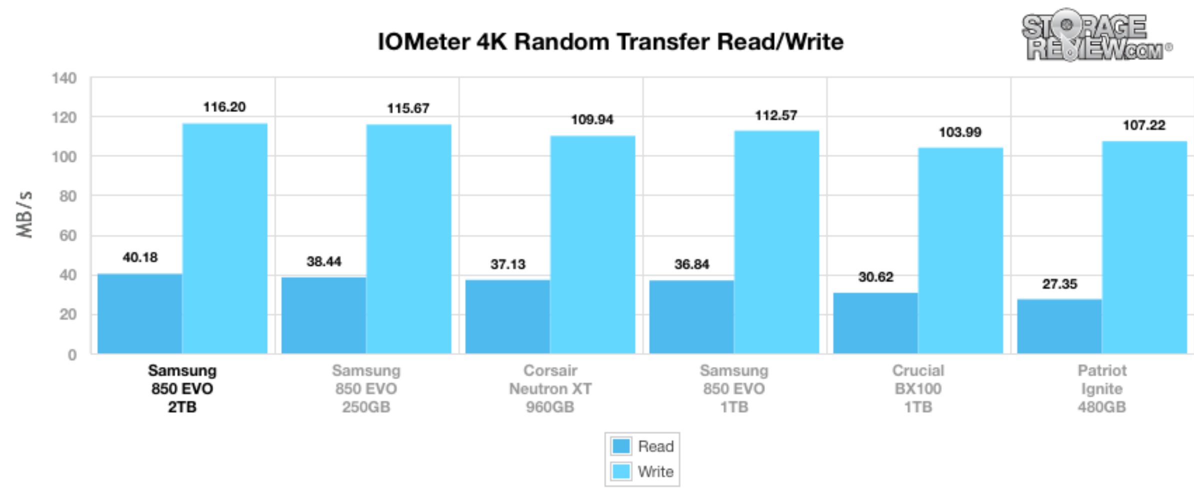
- Recall: In a RAID array, performance is proportional to number of disks; this includes IOPS
- Each disk “provides” some IOPS: $IOPS_{disk}$
- Our workload profile tells us: $IOPS_{workload}$
- Compute $\frac{IOPS_{workload}}{IOPS_{disk}}$: get number of disks needed
- Add overhead: RAID parity disks, hot spares, etc.
- Add safety margin: 20% minimum, >50% if active/active

- Note: this works for SSDs too, $IOPS_{disk}$ is just way bigger

Characterizing disks

- Use synthetic benchmark to find performance in the extremes (100% read, 100% write, 100% seq, 100% random, etc.)
- Results for Samsung 850 Evo 2TB SSD:





From http://www.storagereview.com/samsung_850_evo_ssd_2tb_review

Interpolation-based sizing

- For large/complex storage deployments with mixed workloads, simple IOPS math may break down
- Alternative: **measurement with interpolation**
  - Beforehand:

    foreach (synthetic benchmark configuration with access pattern **a**)

      foreach (storage system configuration **s**)

        set up storage **s**, generate IO pattern **a**, record metrics as **M**[**a**,**s**]
  - Later, given real workload with access pattern $\mathbf{a}_{\mathbf{given}}$ and performance requirements $\mathbf{M}_{\mathbf{required}}$
    - Find points **a,s** in table where **a** is near $\mathbf{a}_{\mathbf{given}}$ and performance **M[a,s]** > $\mathbf{M}_{\mathbf{required}}$
    - Deploy a storage system based on the constellation of corresponding **s** values.
      - Can interpolate storage configurations **s** (with risk)
      - Pessimistic model: Can just pick from systems where **a** was clearly "tougher" and performance **M**[**a,s**] was still sufficient
- Why do all this? Because **s** can include ALL storage config parameters (storage efficiency, cache, config choices, etc.)

Combining workloads

- Rare to have one storage system handle just ONE workload; shared storage on the rise

- Can we simply add workload demands together?
  - Sometimes...it's complicated.

  - Example that works: two random workloads run on separate 3-disk RAIDs will get similar performance running together one 6-disk RAID

  - Example that doesn't: a random workload plus a sequential workload wrecks performance of the sequential workload
    - Random IOs will "interrupt" big sequential reads that would otherwise be combined by OS/controller.

From "Storage Workload Characterization and Consolidation in Virtualized Environments" by Ajay Gulati, Chethan Kumar, Irfan Ahmad. In VPACT 2009.

Workload combining

RAID5 config

	Workload	LUN configuration	IOPS	IO latency	Application Metric
Random	DVDStore	2+1	130	100	6132 TPM
Random	OLTP	2+1	141	30	5723 TPM
	DVDStore (Shared)	5+1	144	30	7630 TPM
	OLTP (Shared)	5+1	135	30	5718 TPM

Table 1. Comparison of DVDStore and OLTP when run in isolation and shared mode

	Workload	LUN configuration	Throughput	95% tile latency	Application Metric
Random	DVDStore	2+1	130 IOPS	100	6132 TPM
Sequential	DSS	2+1	44 MB/s	30	6 completed transactions
	DVDStore (Shared)	5+1	164 IOPS	15	7630 TPM
	DSS (Shared)	5+1	31 MB/s	1	3 completed transactions

Table 2. Comparison of DVDStore and DSS when run in isolation and shared mode

- “OLTP” = “Online Transaction Processing” (normal user-activity-driven database)
- “DSS” = “Decision Support System” (long-running report on a database)

- Table 2: DVDStore benefits a little from twice as many disks to help with latency, but DSS’s sequential IO gets wrecked by the random interruptions to its stream

From “Storage Workload Characterization and Consolidation in Virtualized Environments” by Ajay Gulati, Chethan Kumar, Irfan Ahmad. In VPACT 2009.

Conclusion

- To **characterize** a workload, we must **profile** it
  - Run it (generating user input if needed)
  - Measure IO metrics in app/kernel/hypervisor/controller
- Can use workload profile for **sizing**:
  to identify storage gear needed
  - Basic rule: provision enough disks for the IOPS you need
  - Past that, look for published guidance from software/hardware vendor
  - Failing that, use successive experiments with differing gear to identify performance trends