ECE590-03
Enterprise Storage Architecture

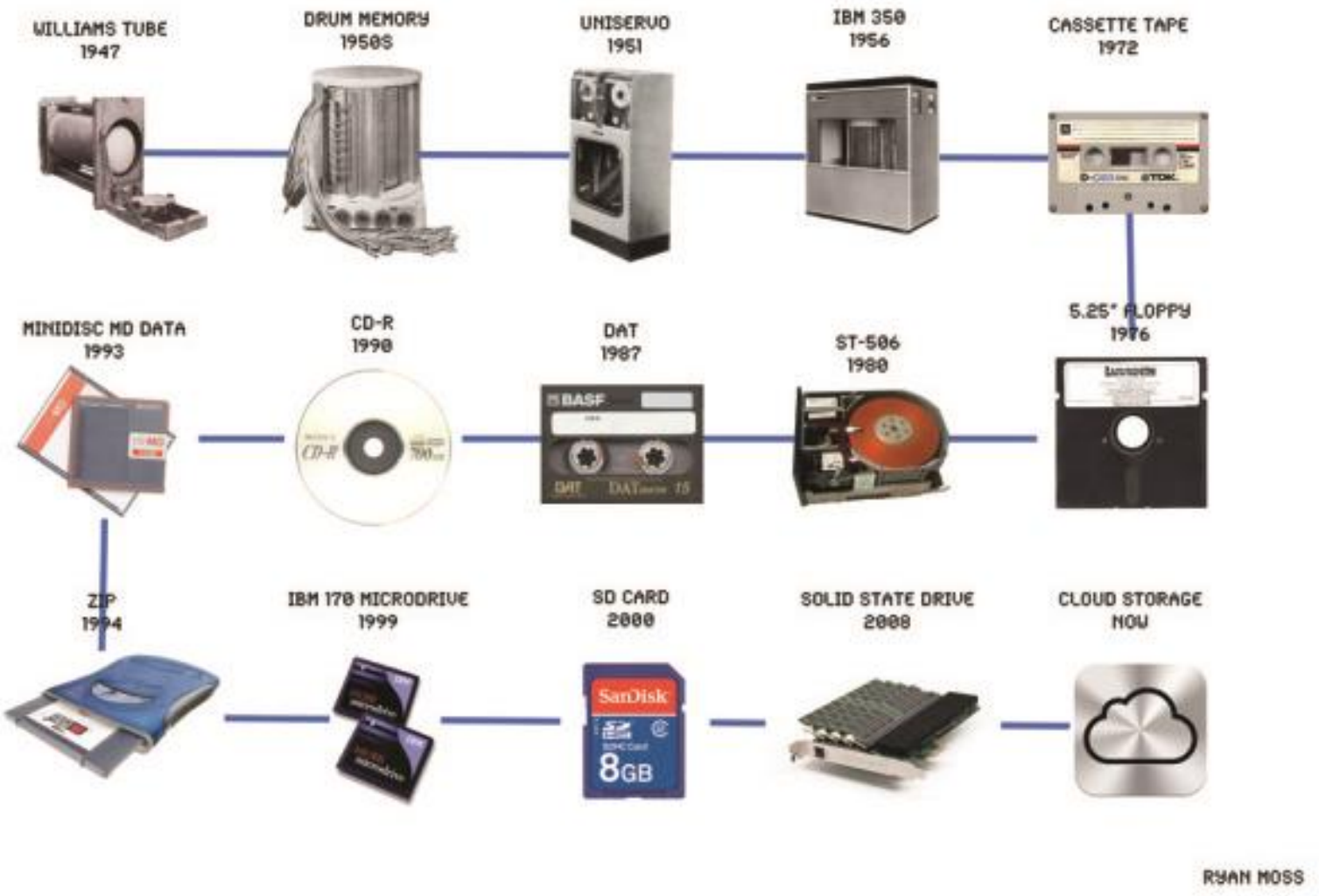
Fall 2017

Storage devices

Tyler Bletsch
Duke University

Slides include material from Vince Freeh (NCSU)

Basic storage device history



- From https://aaronlimmv.wordpress.com/2013/05/02/types-of-storage-and-basic-advantages-and-disadvantages/

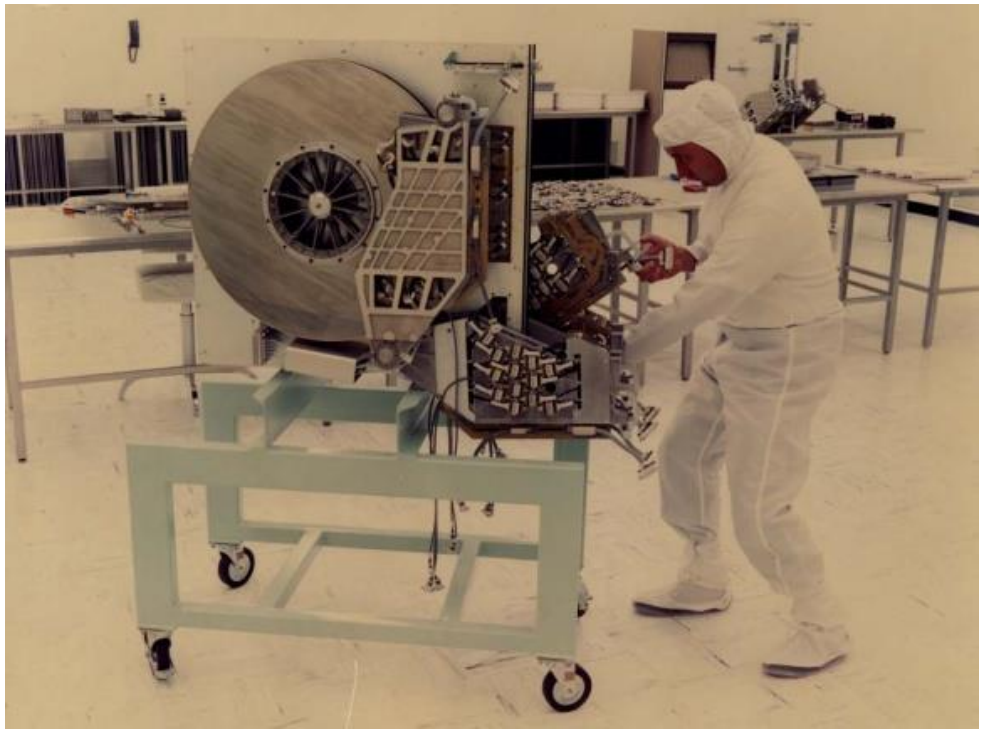
The ancient model of large enterprise storage

- DASD: Direct Access Storage Device
  - Starting with the IBM 350 in 1956
  - Your One Big Computer accesses your One Big Drive
  - Evolution: make the One Big Drive bigger and more reliable
  - Result: The One Big Drive became more and more expensive and critical
  - **Problem?**

An IBM 350 drive (5 MB) being loaded into a PanAm jet, circa 1956.

DASD problem: single point of failure

- The DASD was a single point of failure with *all* your data
  - Better treat it gently…



Man with amazing fashion sense moves a 250MB disk, circa 1979.

Key trend: consumerizaton

- A common evolution in IT:
  - Businesses use a fancy expensive "Enterprise Thing".
  - Normal people get a cheaper version, "Consumer Thing". It's cheap and good enough.
  - Consumer Thing gets better and better every year because:
    - There are more consumers than businesses (bigger market)
    - There are more vendors for consumers than for businesses (more competition)
    - The margins are thinner for consumer goods (more cut-throat competition)
  - A Smart Person finds a way to use the Consumer Thing for business.
  - Industry experts call the Smart Person dumb and say that no real business could ever use the Consumer Thing.
  - The Smart Person is immensely successful, and all businesses use the Consumer Thing.
  - Industry experts pretend they knew all along.

Consumerization in servers

- Big business use mainframe computers

- Everyone else uses microcomputers

- Microcomputers beat mainframes

- We start calling them "servers"

Piled up in a museum

- Mainframes almost entirely gone

Consumerization in storage

- Big business use DASDs

- Everyone else eventually gets small hard disks (SCSI)

- Disk arrays invented using "**JBOD**" and eventually "**RAID**"

- Storage companies based on disk arrays gain traction

- DASDs are entirely gone

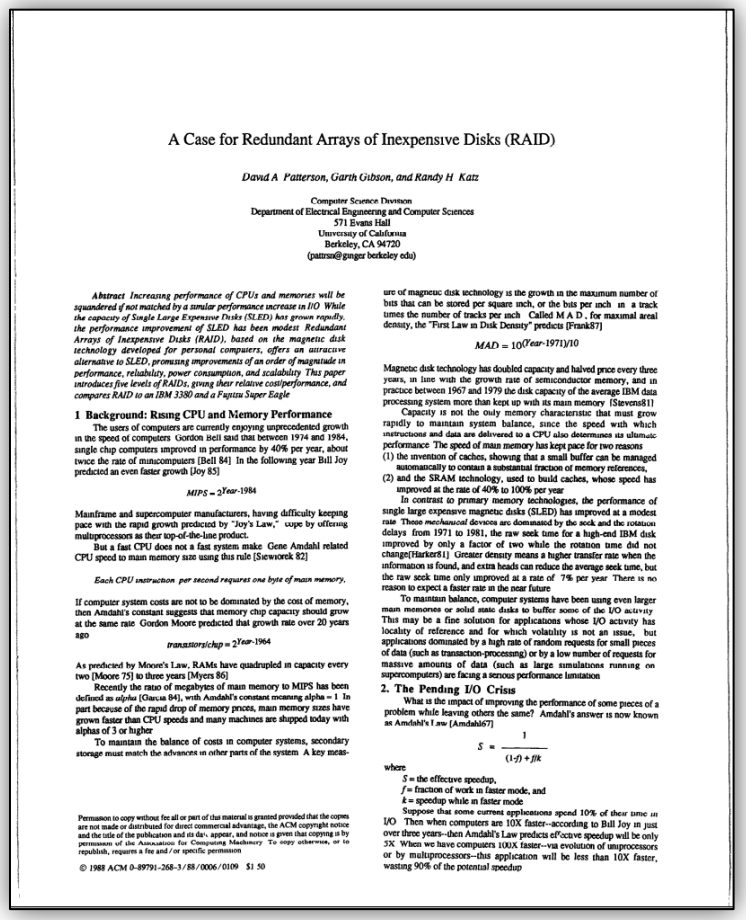
Disk arrays

- **JBOD**: Just a Bunch Of Disks
  - Multiple physical disks in an external cabinet
  - Array is connected to one server only.
  - Provides higher storage capacity with increased number of drives.
  - Effect on performance?
  - Effect on reliability?

- Can we do better?

Disk arrays

- **RAID**: Redundant Array of Inexpensive Disks
  - Academic paper from 1988
  - Revolutionized storage
  - Will discuss in depth later
  - Combine disks in such a way that:
    - Performance is additive
    - Capacity is additive
    - Drive failures can occur without data loss
  - Still directly attached to one server



A Case for Redundant Arrays of Inexpensive Disks (RAID)

David A Patterson, Garth Gibson, and Randy H Katz

Computer Science Division
Department of Electrical Engineering and Computer Sciences
571 Evans Hall
University of California
Berkeley, CA 94720
(pattrsn@ginger berkeley edu)

Abstract Increasing performance of CPUs and memories will be squandered if not matched by a similar performance increase in I/O While the capacity of Single Large Expensive Disks (SLED) has grown rapidly, the performance improvement of SLED has been modest Redundant Arrays of Inexpensive Disks (RAID), based on the magnetic disk technology developed for personal computers, offers an attractive alternative to SLED, promising improvements of an order of magnitude in performance, reliability, power consumption, and scalability This paper introduces five levels of RAIDs, giving their relative cost/performance, and compares RAID to an IBM 3380 and a Fujitsu Super Eagle

1 Background: Rising CPU and Memory Performance

The users of computers are currently enjoying unprecedented growth in the speed of computers Gordon Bell said that between 1974 and 1984, single chip computers improved in performance by 40% per year, about twice the rate of minicomputers [Bell 84] In the following year Bill Joy predicted an even faster growth [Joy 85]

$$MIPS = 2^{Year-1984}$$

Mainframe and supercomputer manufacturers, having difficulty keeping pace with the rapid growth predicted by "Joy's Law," cope by offering multiprocessors as their top-of-the-line product.

But a fast CPU does not a fast system make Gene Amdahl related CPU speed to main memory size using this rule [Siewiorek 82]

Each CPU instruction per second requires one byte of main memory.

If computer system costs are not to be dominated by the cost of memory, then Amdahl's constant suggests that memory chip capacity should grow at the same rate Gordon Moore predicted that growth rate over 20 years ago

$$transistors/chip = 2^{Year-1964}$$

As predicted by Moore's Law, RAMs have quadrupled in capacity every two [Moore 75] to three years [Myers 86]

Recently the ratio of megabytes of main memory to MIPS has been defined as *alpha* [Garcia 84], with Amdahl's constant meaning alpha = 1 In part because of the rapid drop of memory prices, main memory sizes have grown faster than CPU speeds and many machines are shipped today with alphas of 3 or higher

To maintain the balance of costs in computer systems, secondary storage must match the advances in other parts of the system A key measure of magnetic disk technology is the growth in the maximum number of bits that can be stored per square inch, or the bits per inch in a track times the number of tracks per inch Called M A D , for maximal areal density, the "First Law in Disk Density" predicts [Frank87]

$$MAD = 10^{(Year-1971)/10}$$

Magnetic disk technology has doubled capacity and halved price every three years, in line with the growth rate of semiconductor memory, and in practice between 1967 and 1979 the disk capacity of the average IBM data processing system more than kept up with its main memory [Stevens81]

Capacity is not the only memory characteristic that must grow rapidly to maintain system balance, since the speed with which instructions and data are delivered to a CPU also determines its ultimate performance The speed of main memory has kept pace for two reasons

(1) the invention of caches, showing that a small buffer can be managed automatically to contain a substantial fraction of memory references,
(2) and the SRAM technology, used to build caches, whose speed has improved at the rate of 40% to 100% per year

In contrast to primary memory technologies, the performance of single large expensive magnetic disks (SLED) has improved at a modest rate These *mechanical* devices are dominated by the seek and the rotation delays from 1971 to 1981, the raw seek time for a high-end IBM disk improved by only a factor of two while the rotation time did not change[Harker81] Greater density means a higher transfer rate when the information is found, and extra heads can reduce the average seek time, but the raw seek time only improved at a rate of 7% per year There is no reason to expect a faster rate in the near future

To maintain balance, computer systems have been using even larger main memories or solid state disks to buffer some of the I/O activity This may be a fine solution for applications whose I/O activity has locality of reference and for which volatility is not an issue, but applications dominated by a high rate of random requests for small pieces of data (such as transaction-processing) or by a low number of requests for massive amounts of data (such as large simulations running on supercomputers) are facing a serious performance limitation

2. The Pending I/O Crisis

What is the impact of improving the performance of some pieces of a problem while leaving others the same? Amdahl's answer is now known as Amdahl's Law [Amdahl67]

$$S = \frac{1}{(1-f) + f/k}$$

where
S = the effective speedup,
f = fraction of work in faster mode, and
k = speedup while in faster mode

Suppose that some current applications spend 10% of their time in I/O Then when computers are 10X faster--according to Bill Joy in just over three years--then Amdahl's Law predicts effective speedup will be only 5X When we have computers 100X faster--via evolution of uniprocessors or by multiprocessors--this application will be less than 10X faster, wasting 90% of the potential speedup

Next step: intelligent arrays

- Server acts as host for storage, provides access to other servers
  - Dedicated hardware for RAID
  - Optimized for IO performance
  - High speed cache
  - Can add various special features at this layer: access controls, multiple protocols, data compression and deduplication, etc.

Method of Attachment

- How to connect storage array to other systems?
  - DAS: Direct Attached Storage
    - One client, one storage server
  - SAN: Storage Area Network
    - Storage system 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
  - NAS: Network-Attached Storage
    - Storage system 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

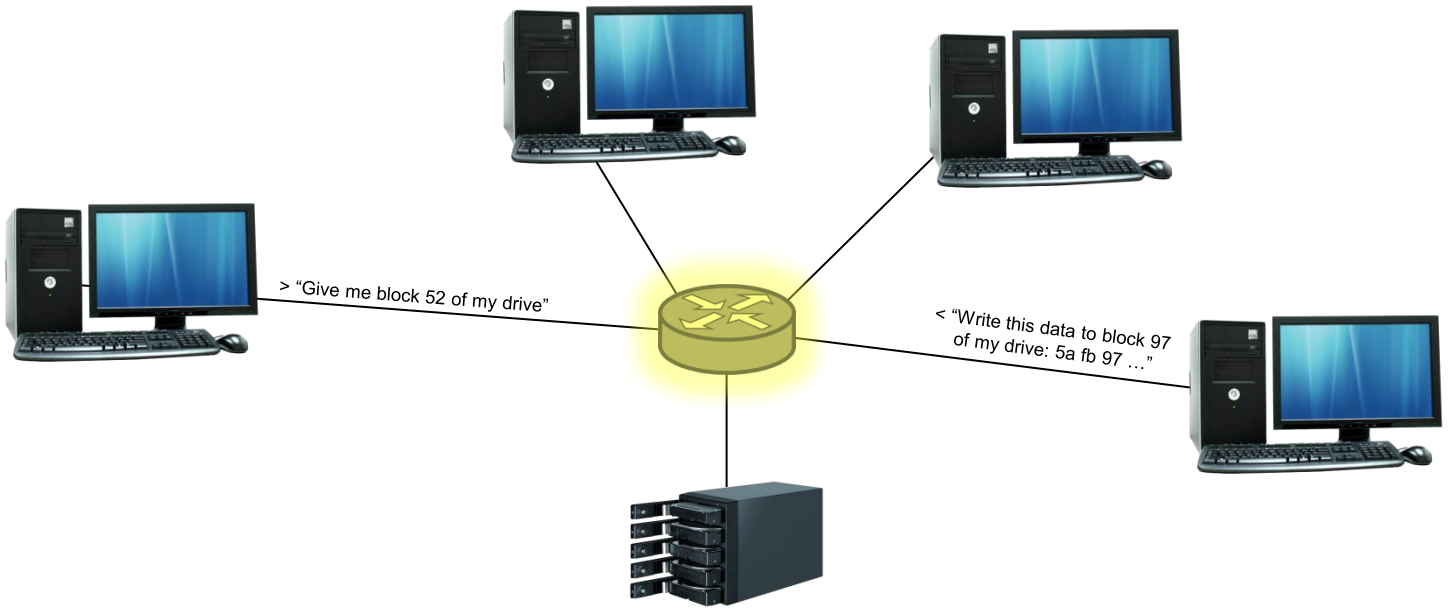
DAS: Direct Attached Storage

- One-to-one connection
- Historically: connect via SCSI ("Small Computer Systems Interface")
  - Even though actual SCSI cables/drives/systems are gone, the software protocol is still *everywhere* in storage. We'll see it again very soon*.
- Modern:
  - USB:
  - SATA (or since it's external, e-SATA): The protocol modern consumer drives use
  - SAS (Serial Attached SCSI): The protocol modern enterprise drives use

* see, I told you.

SAN: Storage Area Network (1)

- Split the aggregated storage into virtual drives called Logical Units (LUNs)
- Clients make read/write requests for blocks of "their" drive(s)
- Storage server translates request for block 50 of client 2 to actual block 4000
  (which in turn is block 1000 of disk 3 of the RAID array)

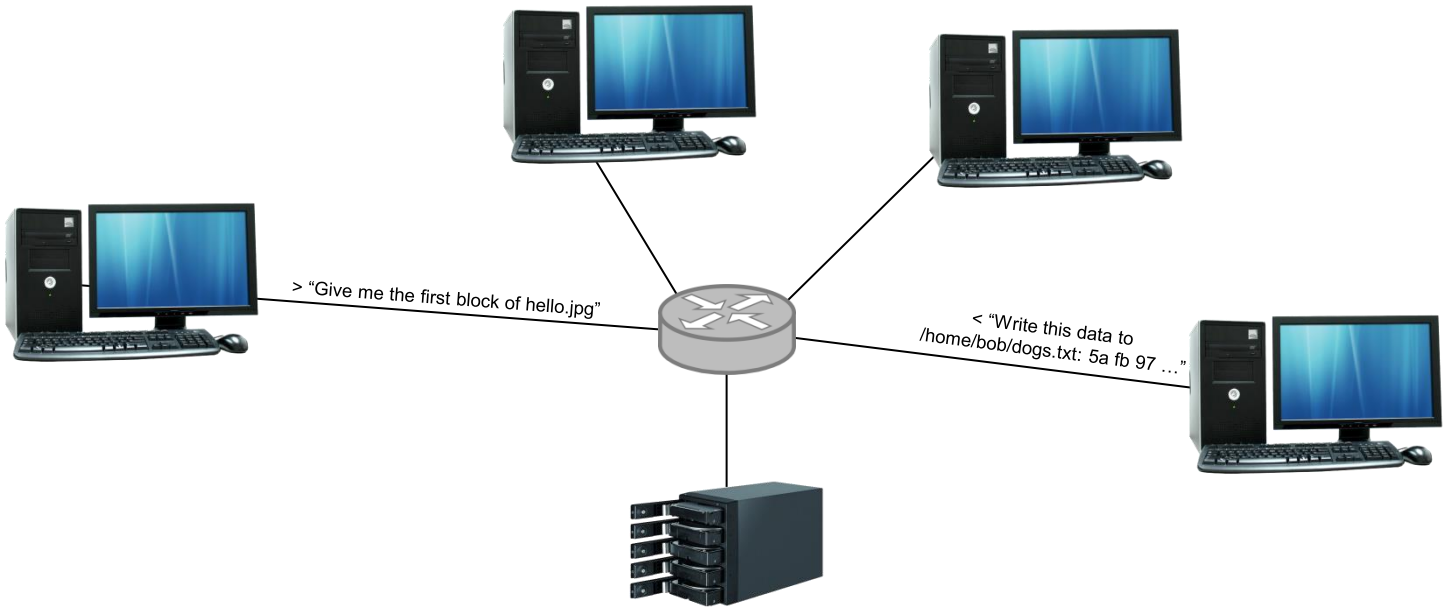


SAN: Storage Area Network (2)

- Historical protocol: Fibre Channel (FC)
  - A special physical network just for storage
  - Totally unlike Ethernet in almost every way
  - Still popular with very conservative enterprises
  - Actual traffic is SCSI frames
  - Clients and servers have special cards: a Host Bus Adapter (HBA) for FC
- Modern protocols:
  - Fibre Channel over Ethernet (FCoE):
    - Requires FCoE-capable switch
    - SCSI inside of an FC frame inside of an Ethernet frame
    - Clients and servers have special cards: a Converged Network Adapter for FCoE/Ethernet
  - iSCSI:
    - SCSI inside of an IP frame, usually inside of an Ethernet frame (but it's IP, so it could be inside a bongo drum frame)
    - No special switch or cards needed (though iSCSI HBAs do technically exist)

NAS: Network-Attached Storage (1)

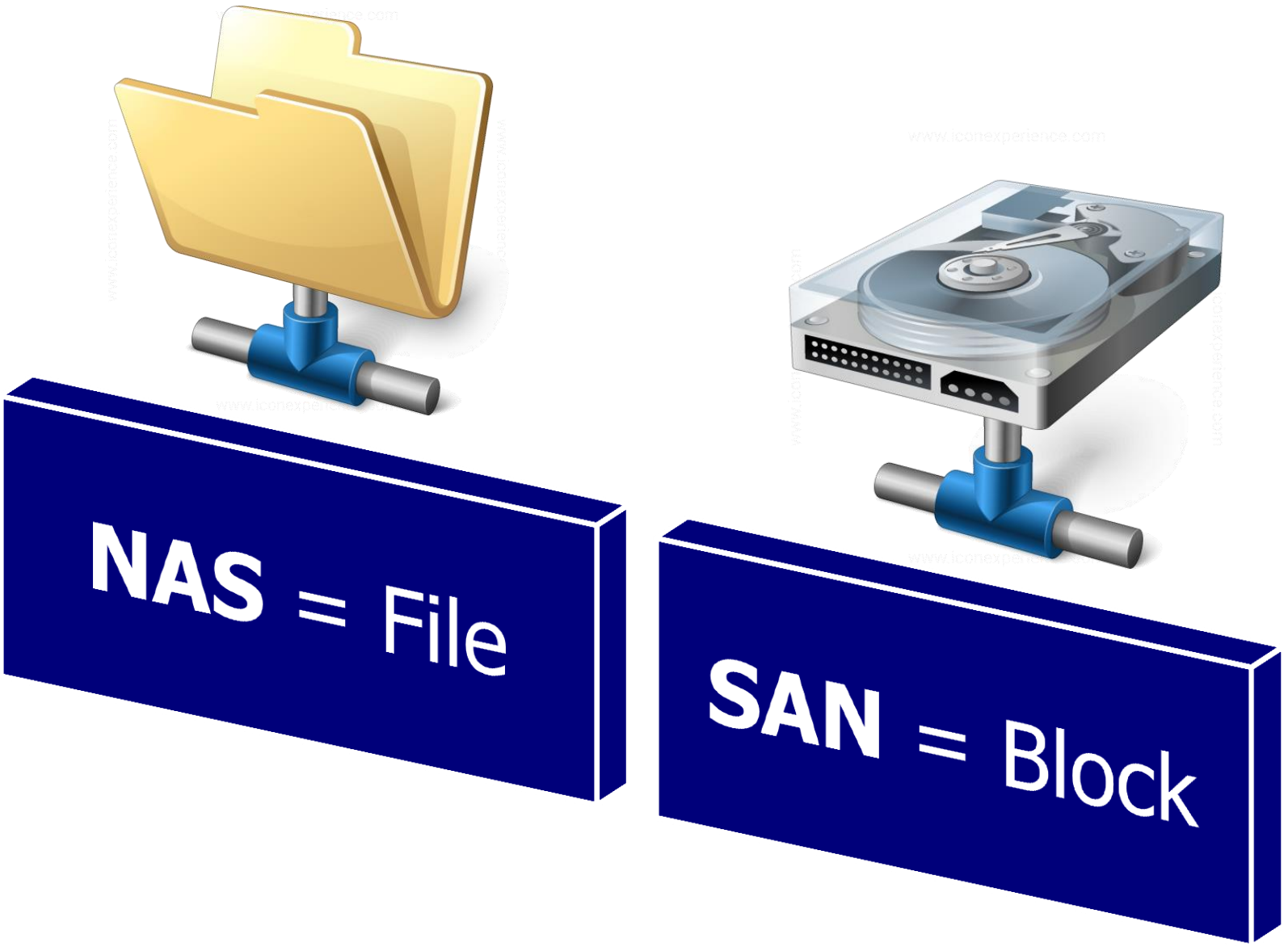
- Put a file system on the storage server so it has the concept of files and directories
- Clients make open/close/read/write requests for files on the remote file system



NAS: Network-Attached Storage (2)

- No special network or cards – works on normal IP/Ethernet
- Network File System (NFS):
  - Common for UNIX-style systems, invented by Sun in 1984
  - Literally just turns the system calls open/close/read/write/etc into "remote procedure calls" (RPCs)
  - Many revisions, we're up to NFS v4 now
- Server Message Block (SMB) also known as Common Internet File System (CIFS)
  - Microsoft Windows standard for network file sharing, developed around 1990
  - Really badly named
  - Many revisions, we're up to SMB 3.1.1 now
  - Native on Windows, supported on Linux with Samba (client and server)

How to tell NAS and SAN apart



System constraints

- What is a **tradeoff**?
- Constraints:
  - Cost
  - Physical environment
  - Maintenance & support
  - Compliance (regulatory/legal)
  - HW & SW infrastructure
  - Interoperability/compatibility

Management activities

- Provisioning: allocate storage for use
- Monitoring: ensure proper functioning over time
- Archival/destruction: retire data properly

Provisioning

- Based on workload requirements:
  - **Capacity** – capacity planning
  - **Performance** – workload profiling
  - **Security** – access rule creation, encryption policy
  - **Reliability** – type of redundancy, backup policy
  - **Other** – archival duration, regulatory compliance, etc.

Monitoring

- **Capacity**: watch usage over time, identify workloads at risk of running out, include in report
- **Performance**: collect metrics at storage layer and/or application layer, compare to requirement, alert on violation/deviation, add resources as needed, include in report
- **Security**: verify access control rules, deploy intrusion/anomaly detection, ensure at-rest and in-flight encryption is used where appropriate, include in report
- **Reliability**: receive alerts when failures occur at any layer, continually ensure that availability and backup policies remain satisfied, include in report
- **Other requirements**: keep 'em satisfied, include in report
- **Report**: Analyze collected statistics over time to assess cost and determine where array growth or configuration changes are needed.

The data lifecycle

From: http://www.spirion.com/us/solutions/data-lifecycle-management

Course project discussion

Project ideas

- Write-once file system*
- Network file system with caching*
- Deduplication*
- Special-case file system*
- File system performance survey
- Hybrid HDD/SSD system*
- Storage workload characterization
- Cloud storage tiering*

** Likely implemented via FUSE*

24

FUSE overview

FUSE

- File System in Userspace: Write a file system like you would a normal program.
- **You** implement the system calls: open, close, read, write, etc.

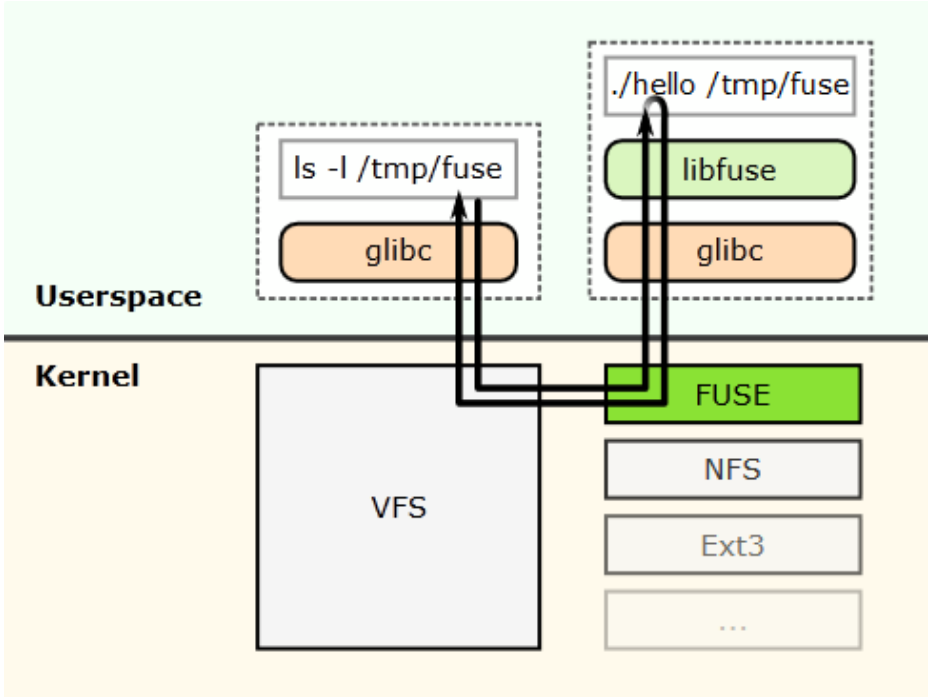


Figure from Wikipedia: http://en.wikipedia.org/wiki/Filesystem_in_Userspace

FUSE Hello World

```
~/fuse/example$ mkdir /tmp/fuse
~/fuse/example$ ./hello /tmp/fuse
~/fuse/example$ ls -l /tmp/fuse
total 0
-r--r--r-- 1 root root 13 Jan 1 1970 hello
~/fuse/example$ cat /tmp/fuse/hello
Hello World!
~/fuse/example$ fusermount -u /tmp/fuse
~/fuse/example$
```

- Let's walk through it:
  https://github.com/libfuse/libfuse/blob/master/example/hello.c

Project idea
Write-once file system

Write-once file system (WOFS)

- Normal file system
  - Read/write
  - Starts empty, evolves over time
  - Simplest implementation isn't simple
    - Fragmentation and indirection

- Write-once file system
  - Read-only
  - Starts "full", created with a body of data
  - Simple implementation
    - No fragmentation, little indirection

What is a WOFS for?

- CD/DVD images
  - *"Master" the image with the content in /mydir*
    `$ mkisofs -o my.iso /home/user/mydir`
  - *Write the disc image directly onto the burner*
    `$ cdrecord my.iso`
- Ramdisk images (e.g. cramfs, squashfs, etc.)

Major parts of a WOFS

- **Mastering program:**
  ```
  $ mkwofs myfilesystem.img data/
  ```

- **Mounting program (FUSE):**
  ```
  $ wofsmount myfilesystem.img dir/
  $ ls dir/
      …
  ```

- *Mounting program must not "extract" data at load time – data is retrieved from the image as read requests are handled!*

Project idea
Network file system with caching

Network File System without Special Sauce

- Simple idea:
  Put IO system calls over the network

- Complex consequences:
  - Stateful or stateless?
  - Caching? Cache coherency?
  - What server? How many servers?
  - Data compression?
  - Data reduction, e.g. "Low-bandwidth File System" (http://pdos.csail.mit.edu/papers/lbfs:sosp01/lbfs.pdf)

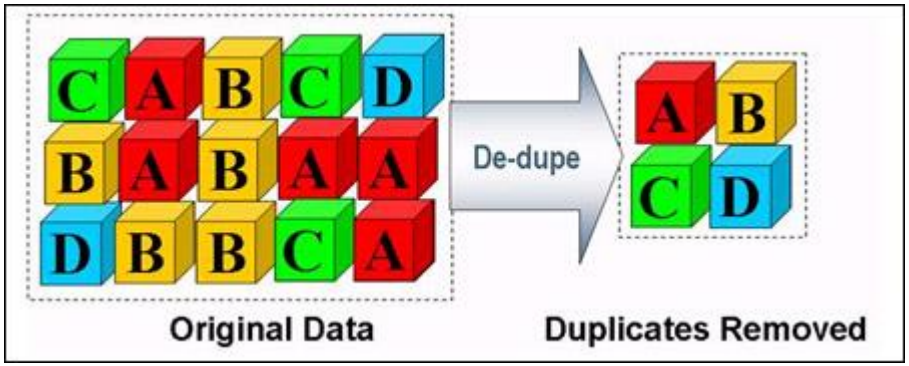
An *interesting* network file system

- A basic network filesystem is basic OS stuff
- Yours must could also optionally have:
  - Read caching and write-behind caching
  - Read caching and read-ahead optimization
  - Distributed storage over multiple servers
  - Compression
  - "Low-bandwidth file system" features
    - (Persistent disk cache, basically dedupe-on-the-wire)
  - Something else?

Project idea
Deduplication

Deduplication

- Will be covered later, here's the short version



- Split the file in to chunks
- Hash each chunk with a big hash
- If hashes match, data matches:
  - Replace this with a reference to the matching data
- Else:
  - It's new data, store it.

Figure from http://www.eweek.com/c/a/Data-Storage/How-to-Leverage-Data-Deduplication-to-Green-Your-Data-Center/

Common deduplication data structures

- Metadata:
  - Directory structure, permissions, size, date, etc.
  - Each file's contents are stored as a **list of hashes**

- Data pool:
  - A flat table of hashes and the data they belong to
  - Must keep a reference count to know when to free an entry

Design decisions

- **Eager** or **lazy**?

- **Fixed-** or **variable-sized** blocks?
  - Variable size via Rabin-Karp Fingerprinting

Project idea
Special-case file system

Special-case file system

- Sometimes "general purpose" is *too* general

- Example motivations:
  - Can we exploit a workload's peculiar access pattern?
  - Can we examine the data to present new organizational structures?
  - Can we map non-filesystem information into the file system?

Tips to keep in mind

- Performance: Disk seeks are the enemy!
  - Often, "Minimize seeks" = "Optimize performance"

- Metadata: Many files have metadata not usually exposed to the file system, such as JPEG EXIF tags, MP3 ID3 tags, DOC/DOCX author tags, etc.

- Anything can be a filesystem. You can have a file system represent:
  - A git server
  - An email account
  - A web server
  - A physical system (e.g. "Internet of Things"*)
  - A database (e.g. via the Duke registration system public API**)
  - More!

* This term is really dumb, and I'm sorry for using it.
** http://dev.colab.duke.edu/resource/duke-public-apis

Project idea
File system performance survey

File system performance survey

- Storage systems are enormously complex with many pieces affecting overall performance
  - Filesystem (ext3, ntfs, etc.)
  - Filesystem configuration (journaling, alignment, etc.)
  - Workload (benchmarks)
  - Underlying devices (SSD, HDD, and also RAID)
- It is useful to characterize how different configurations perform under different workloads

How to approach the problem

- Get hardware
  - Such as the **course server**!!
- Define your test variables
- Build a test harness
  - Automate all testing, it will run for days!
  - Automate data collation – don't scrape numbers by hand!
  - Get it all into a giant spreadsheet
- Data mining – find knowledge in the data
- Detailed write up of interesting conclusions

Project idea
Hybrid HDD/SSD system

Hybrid storage

- SSD is expensive per GB, cheap for random IO performance
- HDD is the opposite

- Can develop a software that gets best of both worlds

- Examples:
  - SSD as cache for HDD
  - SSD as write buffer for HDD
  - Auto-migrate "hot" data to SSD, "cold" data to HDD
  - Identify random workloads, migrate to SSD
- Mechanism:
  - File system (e.g. with FUSE)
  - Virtual block device (also possible with FUSE)

Evaluation

- Must include:
  - Benchmark of your system against pure HDD and pure SSD systems.
  - Measurement of FUSE overhead
  - Cost/benefit analysis based on HDD and SSD costs
  - All of the above must be conducted against a good cross-section of workloads

Project idea
Storage workload characterization

Storage workload capture

- In storage sizing, need to characterize workload
- Workload may be confidential or too complex to migrate

- Project: Use a technique to *record* a storage workload
  - Example 1: take a trace of read/write ops; need to *anonymize*, then be able to replay operations with equivalent performance
  - Example 2: monitor I/O ops, characterize nature of workload, then be able to simulate a request stream with similar characteristics

- Will need to prove the accuracy of your technique with statistical analysis across variety of workloads

Project idea
Cloud storage tiering

Cloud storage tier

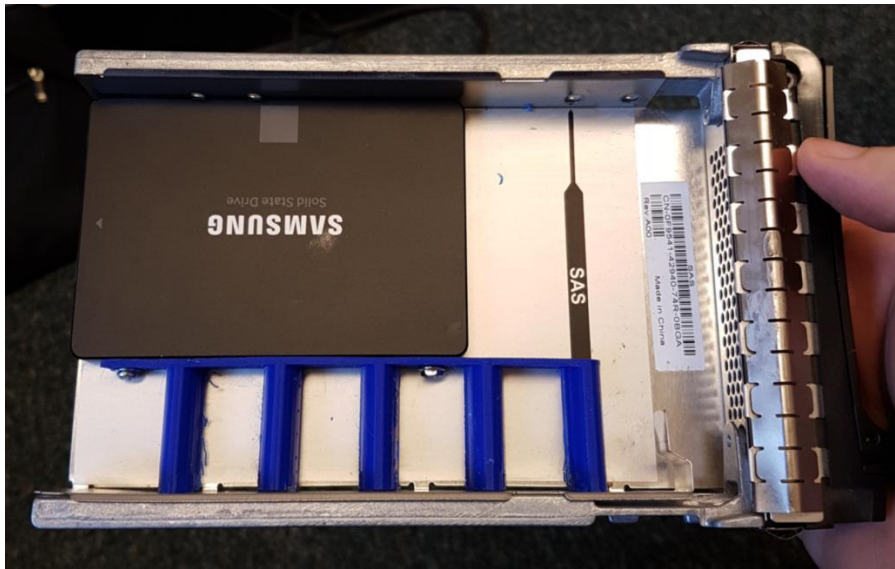
- Cloud storage (e.g. Amazon S3) is useful, generally pretty cheap
- Downside: internet latency and bandwidth

- Can develop a storage system which migrates "cold" or otherwise lower-priority data out to a cloud service, brings it back live on demand without user interaction
  - Optional enhancements:
    - Intelligent prediction algorithm for migration
    - Encryption for cloud-exported data
    - Compression for cloud-exported data
  - Can be implemented at block level or file system level

An important resource: the course reference server

Server overview

- A **storage server** has been built for this course for use by all students.
  - Dell PowerEdge 2950, a 2U rackmount storage system.
  - Has drives to experiment with RAID topologies, hybrid HDD+SSD storage, filesystem performance, and more.
  - Budget exists for upgrades on request.



Server stats

- **Processor**: Quad Core Xeon Processor E5310 2x4MB Cache, 1.60GHz, 1066MHz FSB
- **Memory**: 2GB 667MHz (4X512MB), Single Ranked DIMMs
- **Operating system**: Ubuntu Linux 16.04 LTS x64
- **Storage controller**: PERC 5/i, x6 SAS RAID Controller Card
- **Storage bays**: 1x6 Backplane for 3.5-inch SAS/SATA Hard Drives
- **Networking**: 2x 1GbE ethernet. One uplink connected at present.
- **Drives**:
  - [3x] Western Digital 250GB 7200rpm SATA 3Gbps 3.5-in HDD (circa 2007)
  - [1x] Samsung 850 EVO SSD, SATA, 250GB (new)
  - [1x] Zheino SSD, SATA, 30GB (the cheapest SSD on Amazon today)
  - [1x] Sandisk USB thumb drive, 30GB (contains the OS, not for testing!)
- **Features**: Redundant Power Supply, out-of-band BMC management via IPMI

Server access

Access it from campus or via VPN via SSH:

storemaster.egr.duke.edu

User accounts created upon request
(includes root access).

Students will need to share the server; the exact mechanism for doing so will be determined during the project outline phase.

Questions?