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.
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: consumerization

- 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”
- 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
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: External drives, very fast as of USB 3.0
  - SATA (or if 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.*
• 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

NAS = File

SAN = Block
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

Course project discussion
FUSE in this course

- Project will involve writing filesystem code using FUSE

- Assignments “Program 0”, “Program 1”, “Program 2” are individual
  - Introduce you to FUSE
  - Work you through writing a basic filesystem
  - Prepare you for the project
• **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/Filesistem_in_Userspace
FUSE Hello World

```bash
~/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](https://github.com/libfuse/libfuse/blob/master/example/hello.c)
• Semester long effort in some area of storage
• Several choices (plus choose-your-own)
• Instructor feedback at each stage

• Any stage can result in a need for **resubmission** (grade withheld pending a second attempt).

• See **course site project page** for details
But what *is* the project?

- Start with a basic filesystem both group members wrote individually (Program 2)
- Add feature(s) that improve one or more of:
  - Availability/recoverability
  - Network-accessibility
  - Storage efficiency
  - Performance
  - Security
- Alternately, you may propose a **wildcard project** (custom goal, may or may not use FUSE at all)
Example projects

- **Availability/recoverability**
  - RAID at the filesystem level
  - Mirroring to second system (or cloud?)

- **Network-accessibility**
  - Make a network filesystem
  - Store to cloud service

- **Storage efficiency**
  - Filesystem deduplication
  - Filesystem compression

- **Performance**
  - Minimal-seek on disk data structures
  - Caching with read-ahead
  - Hybrid SSD+HDD filesystem

- **Security**
  - Access control list support
  - Per-user at-rest file encryption

**Wildcard projects**

- Special purpose file system (e.g. MP3 transcoding)
- Custom block device instead of file system
  - Custom RAID
  - Custom SAN
  - Block-level encryption
  - Block-level compression
  - Block-level deduplication
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

\[\text{Figure from http://www.eweek.com/c/a/Data-Storage/How-to-Leverage-Data-Deduplication-to-Green-Your-Data-Center/}\]
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.
Be thinking about possible projects as we go!

We’ll revisit project selection closer to the proposal...
Questions?