Operating System Concepts

Lecture 34: Redundant Array of Inexpensive Disks

Omid Ardakanian
oardakan@ualberta.ca
University of Alberta

MWF 12:00-12:50 VVC 2 215

Storage management

- low-level (physical) formatting is marking out tracks and creating sectors on the disk
  - is usually part of the manufacturing process
- logical formatting is writing initial file-system data structures (free-space list, directory structure, etc.) onto a volume

Storage management

- partitioning is dividing a disk into block groups
  - each partition is treated as though it were a separate drive, so it can hold a file system or be used as raw disk (e.g., for the swap space)
    - partition information is written at a fixed disk location
    - a device entry is created for each disk partition (e.g., in /dev in Linux)
  - a device with a boot partition (i.e., a bootstrap program written in the boot block) is called boot disk

Storage management

- partitioning is dividing a disk into block groups
  - each partition is treated as though it were a separate drive, so it can hold a file system or be used as raw disk (e.g., for the swap space)
    - partition information is written at a fixed disk location
    - a device entry is created for each disk partition (e.g., in /dev in Linux)
  - a device with a boot partition (i.e., a bootstrap program written in the boot block) is called boot disk
- mounting is process of making a file system that resides on a partition (i.e., a **volume**) available to the system and its users

Swap space management

- swap space can be carved out of the file system (a large file) or created in a separate raw partition

Swap space management

- swap space can be carved out of the file system (a large file) or created in a separate raw partition
- if it is created in a raw partition, a separate swap-space storage manager will be responsible for allocating and deallocating disk blocks, optimizing for I/O speed
  - advantage: yields better performance than swapping in file system because all file-system services are bypassed
  - disadvantage: adding more swap space requires repartitioning the disk

Error detection and correction

- error correction code (ECC) is a form of redundancy introduced to detect and correct a limited number of errors
- it is calculated for every disk sector at write time and is stored on disk; it is then recalculated at read time to detect and correct problems
  - if only a few bits have been corrupted, a recoverable soft error is signalled and the error will be corrected
  - it more bits been corrupted, a non-correctable hard error is signalled

Error detection and correction

- error correction code (ECC) is a form of redundancy introduced to detect and correct a limited number of errors
- it is calculated for every disk sector at write time and is stored on disk; it is then recalculated at read time to detect and correct problems
  - if only a few bits have been corrupted, a recoverable soft error is signalled and the error will be corrected
  - it more bits been corrupted, a non-correctable hard error is signalled
- an error may indicate a bad sector
  - disk controller stores the list of bad blocks
  - sector sparing/forwarding is replacing bad sectors with spare sectors (which are not otherwise visible to the OS)

Error detection and correction

- error correction code (ECC) is a form of redundancy introduced to detect and correct a limited number of errors
- it is calculated for every disk sector at write time and is stored on disk; it is then recalculated at read time to detect and correct problems
  - if only a few bits have been corrupted, a recoverable soft error is signalled and the error will be corrected
  - it more bits been corrupted, a non-correctable hard error is signalled
- an error may indicate a bad sector
  - disk controller stores the list of bad blocks
  - sector sparing/forwarding is replacing bad sectors with spare sectors (which are not otherwise visible to the OS)
- various checksums and error detection and correction techniques (e.g., Reed-Solomon code) are used in enterprise products
  - they are not implemented for consumer products

Today's class

- How to use multiple disks in concert to build a faster, bigger, and more reliable disk system?
- What are the trade-offs in redundancy, performance, and reliability?

Today's class

- How to use multiple disks in concert to build a faster, bigger, and more reliable disk system?
- What are the trade-offs in redundancy, performance, and reliability?

first-generation Berkeley RAID prototype, with RAID 5 functionality implemented across 32 disk drives

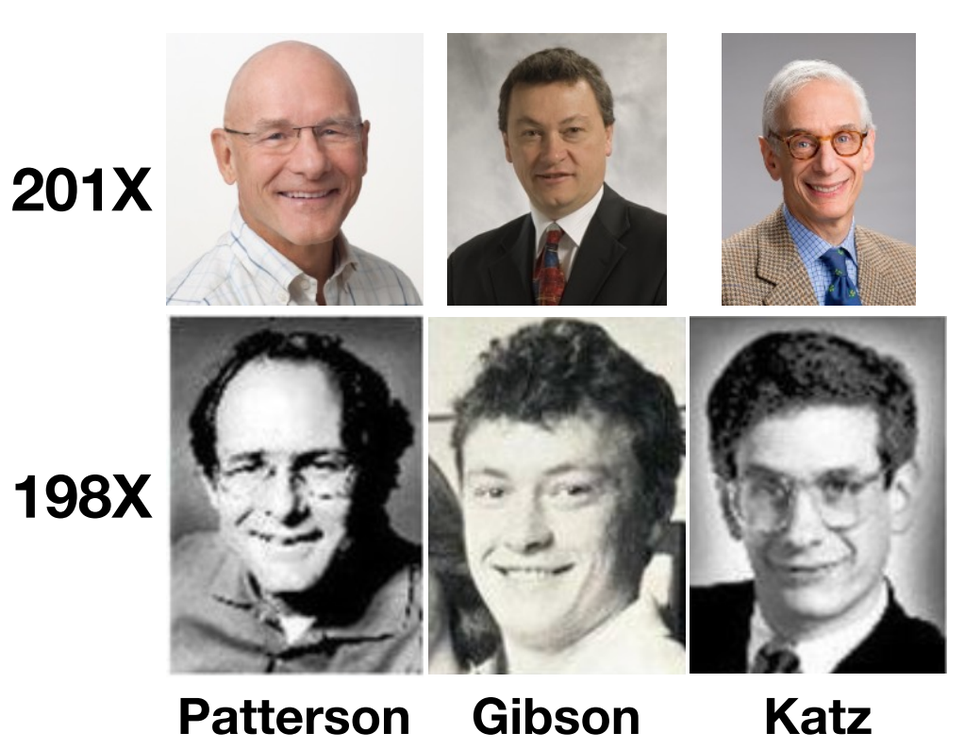
198X

Patterson Gibson Katz

R. H. Katz, "RAID: A Personal Recollection of How Storage Became a System," in *IEEE Annals of the History of Computing*, vol. 32, no. 4, pp. 82-87, Oct.-Dec. 2010.

Today's class

- How to use multiple disks in concert to build a faster, bigger, and more reliable disk system?
- What are the trade-offs in redundancy, performance, and reliability?



first-generation Berkeley RAID prototype, with RAID 5 functionality implemented across 32 disk drives

R. H. Katz, "RAID: A Personal Recollection of How Storage Became a System," in *IEEE Annals of the History of Computing*, vol. 32, no. 4, pp. 82-87, Oct.-Dec. 2010.

Storage devices were getting smaller and cheaper!

- 1980s — what could be done with very large numbers of lower-cost and lower-reliability smaller disk drives?
  - use them in parallel to increase I/O transfer rate (bandwidth)? **RAID 0**

Storage devices were getting smaller and cheaper!

- 1980s — what could be done with very large numbers of lower-cost and lower-reliability smaller disk drives?
  - use them in parallel to increase I/O transfer rate (bandwidth)? **RAID 0**
    - but the probability of a disk failing scales with the number of disks

More disks means more failures…

if p is the probability of a disk failure, the probability of having at least one failure given an array of N disks is $1-(1-p)^N$ assuming the independence of disk failure events **(not a good assumption though…)**

for example, if p=0.01 then with 100 disks the probability of having one or more disk failures is $1-(1-p)^N=0.634$

Storage devices were getting smaller and cheaper!

- 1980s — what could be done with very large numbers of lower-cost and lower-reliability smaller disk drives?
  - use them in parallel to increase I/O transfer rate (bandwidth)? **RAID 0**
    - but the probability of a disk failing scales with the number of disks
  - partition and redundantly spread data across these drives to improve reliability? **RAID 1 to RAID 5+**

Storage devices were getting smaller and cheaper!

- 1980s — what could be done with very large numbers of lower-cost and lower-reliability smaller disk drives?
  - use them in parallel to increase I/O transfer rate (bandwidth)? **RAID 0**
    - but the probability of a disk failing scales with the number of disks
  - partition and redundantly spread data across these drives to improve reliability? **RAID 1 to RAID 5+**

today virtually all server- and networked-based storage is based on RAID

"the RAID market was $25 billion per year in 2002, with more than $150 billion in RAID storage device sold since 1990"

Redundant Array of Inexpensive Disks (RAID)

- RAID offers a number of advantages over a single disk
  - performance: multiple disks in parallel can greatly speed up I/O time
  - capacity: multiple disks provide more space for persistent data storage
  - reliability: RAID makes data less vulnerable to the loss of a single disk using **some form of redundancy**

Redundant Array of ~~Inexpensive~~ Independent Disks (RAID)

- RAID offers a number of advantages over a single disk
  - performance: multiple disks in parallel can greatly speed up I/O time
  - capacity: multiple disks provide more space for persistent data storage
  - reliability: RAID makes data less vulnerable to the loss of a single disk using **some form of redundancy**

- RAID looks like a big, fast, and reliable disk to the file system
  - when a file system issues a logical I/O request to the RAID, the RAID internally must calculate which disk(s) to access to service the request, and issue one or more physical I/Os subsequently

Redundant Array of ~~Inexpensive~~ Independent Disks (RAID)

- RAID offers a number of advantages over a single disk
  - performance: multiple disks in parallel can greatly speed up I/O time
  - capacity: multiple disks provide more space for persistent data storage
  - reliability: RAID makes data less vulnerable to the loss of a single disk using **some form of redundancy**
- RAID looks like a big, fast, and reliable disk to the file system
  - when a file system issues a logical I/O request to the RAID, the RAID internally must calculate which disk(s) to access to service the request, and issue one or more physical I/Os subsequently
- at a high level, a RAID system is very much a specialized computer system with a microcontroller, memory, and disks (**storage as a system**)

Fail-stop fault model

- a disk could be either working or failed at a given time;
  a failed disk can be immediately detected
  - a working disk: all blocks can be read or written
  - a failed disk: data is permanently lost

Fail-stop fault model

- a disk could be either working or failed at a given time;
  a failed disk can be immediately detected
  - a working disk: all blocks can be read or written
  - a failed disk: data is permanently lost
- disk corruption and bad sectors are not defined under this model

How to evaluate a RAID design?

- capacity: how much useful capacity is available to clients of a RAID system comprised of N disks each having B blocks?
  - $N \times B$ without redundancy and $\frac{N \times B}{2}$ with mirroring
  - parity-based schemes tend to fall in between

How to evaluate a RAID design?

- capacity: how much useful capacity is available to clients of a RAID system comprised of N disks each having B blocks?
  - $N \times B$ without redundancy and $\frac{N \times B}{2}$ with mirroring
  - parity-based schemes tend to fall in between
- reliability: how many disk faults a RAID system can tolerate

How to evaluate a RAID design?

- capacity: how much useful capacity is available to clients of a RAID system comprised of N disks each having B blocks?
  - $N \times B$ without redundancy and $\frac{N \times B}{2}$ with mirroring
  - parity-based schemes tend to fall in between
- reliability: how many disk faults a RAID system can tolerate
- performance under some workload
  - performance metrics:
    - steady-state throughput: number of concurrent requests serviced per second
    - single-request (read/write) latency

RAID Level 0: striping

- spread logical data across multiple disks by taking sequential blocks of storage and allocating them one by one across the underlying physical drives
  - introduce no redundancy; just use N disks in parallel

	Disk 0	Disk 1	Disk 2	Disk 3
stripe	0	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	15

RAID mapping:

Disk = A % number_of_disks

Offset = A / number_of_disks

RAID Level 0: striping

- spread logical data across multiple disks by taking sequential blocks of storage and allocating them one by one across the underlying physical drives
  - introduce no redundancy; just use N disks in parallel
- advantage: high data transfer rate
- disadvantage: low reliability

	Disk 0	Disk 1	Disk 2	Disk 3
stripe	0	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	15

RAID mapping:
Disk = A % number_of_disks
Offset = A / number_of_disks

RAID Level 0: analysis

- total storage capacity is $N \times B$
- reliability is low since any disk failure will lead to data loss
- performance is great because all disks can be utilized in parallel to service I/O requests

RAID Level 0: analysis

- total storage capacity is $N \times B$
- reliability is low since any disk failure will lead to data loss
- performance is great because all disks can be utilized in parallel to service I/O requests
  - the bandwidth (for both sequential and random access) is N times the bandwidth of a single disk as all disks are utilized in parallel

RAID Level 0: analysis

- total storage capacity is $N \times B$
- reliability is low since any disk failure will lead to data loss
- performance is great because all disks can be utilized in parallel to service I/O requests
  - the bandwidth (for both sequential and random access) is N times the bandwidth of a single disk as all disks are utilized in parallel
  - the latency of a **single-block request** should be just about identical to that of a single disk because the request is redirected to one of the disks that contains this block

Chunk sizes

- a small chunk size: many files get striped across many disks
  - increasing the parallelism of reads and writes to a single file

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

with a chunk size of 1

Disk 0	Disk 1	Disk 2	Disk 3
0	2	4	6
1	3	5	7
8	10	12	14
9	11	13	15

with a chunk size of 2

RAID Level 1: mirrored storage

- store more than one copy of each block on separate disks to tolerate disk failures & improve reliability
  - typically RAID Level 1 keeps two physical copies of each block (duplicating each block)
  - to read a block, RAID can read either copy
    - can choose closest copy
  - to write a block, RAID must update both copies of the data
    - these writes can take place in parallel (have to deal with consistency problems)

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

RAID 1+0 as it uses mirrored pairs and stripes on top of them

Disk 0	Disk 1	Disk 2	Disk 3
0	1	0	1
2	3	2	3
4	5	4	5
6	7	6	7

RAID 0+1 as it contains two striping arrays and mirrors on top of them

RAID Level 1: mirrored storage

- store more than one copy of each block on separate disks to tolerate disk failures & improve reliability
  - typically RAID Level 1 keeps two physical copies of each block (duplicating each block)
  - to read a block, RAID can read either copy
    - can choose closest copy
  - to write a block, RAID must update both copies of the data
    - these writes can take place in parallel (have to deal with consistency problems)
- advantage: high reliability
- disadvantage: high storage overhead, relatively low write bandwidth

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

RAID 1+0 as it uses mirrored pairs and stripes on top of them

Disk 0	Disk 1	Disk 2	Disk 3
0	1	0	1
2	3	2	3
4	5	4	5
6	7	6	7

RAID 0+1 as it contains two striping arrays and mirrors on top of them

RAID Level 1: analysis

- capacity is $\frac{N \times B}{2}$ only half of the total disk capacity is used (high overhead)
- better reliability as it can certainly tolerate the failure of any one disk
  - RAID 1 can actually tolerate the failure of N/2 disks if they don't contain the same data

RAID Level 1: analysis

- capacity is $\frac{N \times B}{2}$ only half of the total disk capacity is used (high overhead)
- better reliability as it can certainly tolerate the failure of any one disk
  - RAID 1 can actually tolerate the failure of N/2 disks if they don't contain the same data
- performance
  - read latency is the same as the latency on a single disk as RAID only forwards the request to a disk that contains one of the two copies
  - write latency is the maximum of the write latency of the two disks because the two writes can happen in parallel
  - steady-state write throughput: the maximum bandwidth obtained during sequential (and also random) writing to a mirrored array is half of the peak write bandwidth of RAID Level 0
  - steady-state read throughput: the maximum bandwidth obtained during random reading from a mirrored array is the full possible bandwidth, but the maximum bandwidth obtained during sequential reading is lower than the peak read bandwidth (because successive sectors are not read by the same disk)

Using parity bits

- how to improve reliability without significantly reducing I/O bandwidth?
  - add parity bits to data striping

Using parity bits

- how to improve reliability without significantly reducing I/O bandwidth?
  - add parity bits to data striping
- Bit-interleaved parity (RAID level 3): interleave at the level of bits, i.e., store the parity bit in the parity disk

Using parity bits

- how to improve reliability without significantly reducing I/O bandwidth?
  - add parity bits to data striping
- Bit-interleaved parity (RAID level 3): interleave at the level of bits, i.e., store the parity bit in the parity disk
- Block-interleaved parity (RAID level 4): interleave at the level of blocks, i.e., store the parity block in the parity disk

RAID Level 4: saving space with parity

- data stripped across multiple disks
  - successive blocks are stored on successive (non-parity) disks
  - increased bandwidth over a single disk

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	**P0**
4	5	6	7	**P1**
8	9	10	11	**P2**
12	13	14	15	**P3**

RAID Level 4: saving space with parity

- data stripped across multiple disks
  - successive blocks are stored on successive (non-parity) disks
  - increased bandwidth over a single disk
- parity block (in orange) is constructed by XORing data bocks in stripe
  - $P0 = D0 \oplus D1 \oplus D2 \oplus D3$

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	**P0**
4	5	6	7	**P1**
8	9	10	11	**P2**
12	13	14	15	**P3**

RAID Level 4: saving space with parity

- data stripped across multiple disks
  - successive blocks are stored on successive (non-parity) disks
  - increased bandwidth over a single disk
- parity block (in orange) is constructed by XORing data bocks in stripe
  - $P0 = D0 \oplus D1 \oplus D2 \oplus D3$
- computing the parity bit will result in slower writes
  - it is done by RAID controller in its cache

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	**P0**
4	5	6	7	**P1**
8	9	10	11	**P2**
12	13	14	15	**P3**

RAID Level 4: saving space with parity

- data stripped across multiple disks
  - successive blocks are stored on successive (non-parity) disks
  - increased bandwidth over a single disk
- parity block (in orange) is constructed by XORing data bocks in stripe
  - $P0 = D0 \oplus D1 \oplus D2 \oplus D3$
- computing the parity bit will result in slower writes
  - it is done by RAID controller in its cache
- recovering data stored a disk that failed
  - if one sector of D2 is damaged, it can be replaced with $D2 = D0 \oplus D1 \oplus P0 \oplus D3$

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	**P0**
4	5	6	7	**P1**
8	9	10	11	**P2**
12	13	14	15	**P3**

RAID Level 4: saving space with parity

- for reads smaller than the block size, need to access only one disk (better throughput than RAID 3)

RAID Level 4: saving space with parity

- for reads smaller than the block size, need to access only one disk (better throughput than RAID 3)
- for large writes (e.g., a stripe of blocks), computing the parity block and storing it on the parity disk

RAID Level 4: saving space with parity

- for reads smaller than the block size, need to access only one disk (better throughput than RAID 3)
- for large writes (e.g., a stripe of blocks), computing the parity block and storing it on the parity disk
- for small writes
  - read current stripe of blocks, compute parity with the new block, write the parity block
  - better solution: read current version of block being written; read current version of parity block; compute how parity would change, i.e., if a bit on block changed, the corresponding parity bit needs to be flipped; write new version of block; write new version of parity block

RAID Level 4: saving space with parity

- for reads smaller than the block size, need to access only one disk (better throughput than RAID 3)
- for large writes (e.g., a stripe of blocks), computing the parity block and storing it on the parity disk
- for small writes
  - read current stripe of blocks, compute parity with the new block, write the parity block
  - better solution: read current version of block being written; read current version of parity block; compute how parity would change, i.e., if a bit on block changed, the corresponding parity bit needs to be flipped; write new version of block; write new version of parity block
- the disk containing parity block is updated on all writes

RAID Level 5: high I/O rate with rotating parity

- data and parity blocks are distributed across disks
  - spreads load evenly (load balancing)
  - multiple writes could potentially be serviced at the same time
  - all disks can be used for servicing reads

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	**P0**
5	6	7	**P1**	4
10	11	**P2**	8	9
15	**P3**	12	13	14
P4	16	17	18	19

RAID Level 5: high I/O rate with rotating parity

- data and parity blocks are distributed across disks
  - spreads load evenly (load balancing)
  - multiple writes could potentially be serviced at the same time
  - all disks can be used for servicing reads
- any disk can fail and data reconstruction is still possible

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	**P0**
5	6	7	**P1**	4
10	11	**P2**	8	9
15	**P3**	12	13	14
P4	16	17	18	19

Comparison

- RAID 5 vs. normal disk
  - RAID-5: better throughput, better reliability, good bandwidth for large reads, small waste of space (low storage overhead)
  - normal disks: perform better for small writes

Comparison

- RAID 5 vs. normal disk
  - RAID-5: better throughput, better reliability, good bandwidth for large reads, small waste of space (low storage overhead)
  - normal disks: perform better for small writes
- RAID 1 vs. RAID 5
  - RAID-1 wastes more space (high storage overhead)
  - for small writes: RAID 1 is better

Going beyond RAID level 5

- How to allow more disk failures?
  - RAID level 6 allows 2 disks to fail by utilizing 2 blocks of redundant data

Going beyond RAID level 5

- How to allow more disk failures?
  - RAID level 6 allows 2 disks to fail by utilizing 2 blocks of redundant data
  - must do something more complex that just XORing blocks! instead of parity use error correcting codes (check out EVENODD code for example)

M. Blaum, J. Brady, J. Bruck and J. Menon, "EVENODD: an optimal scheme for tolerating double disk failures in RAID architectures," *Proceedings of 21 International Symposium on Computer Architecture*, Chicago, IL, USA, 1994, pp. 245-254.