

### A Case for Redundant Arrays of Inexpensive Disks

Patterson, Gibson and Katz (Seminal paper)  
Chen, Lee, Gibson, Katz, Patterson (Survey)

Circa late 80s..

- MIPS = 2<sup>Year-1984</sup> Joy's Law
- There seems to be plenty of main-memory available (multi megabytes per machine).
- To achieve a balanced system, Secondary Storage Systems have to match the above developments.
- Caches (built with SRAM technology) feed CPUs with instructions fast and provide a bridge between the top two layers of the memory hierarchy
- SLED (Single Large Expensive Disk) had shown modest improvement...
  - Seek times improved from 20ms in 1980 to 10ms in 1994
  - Rotational speeds increased from 3600/minute in 1980 to 7200 in 1994

### Amdahl's Law

$$S = \text{effective speedup} = 1 / [ (1-f) + (f/k) ]$$

- f = fraction of work done in faster mode
- k = speedup while in faster mode

Example: suppose application does I/O during 10% of its time

When computers become 10X faster,  
Amdahl's law says that effective speedup is only 5X !

When computers become 1000000X faster,  
Amdahl's law says that speedup will be ONLY 10X !!!

### Data Sheet

Comparison of Data between two Disk Units of the time...

IBM 3380	Conner CP 3100
14" in diameter	3.5" in diameter
7,500 Megabytes	100 Megabytes
\$135,000	\$1,000
120-200 IO's/sec	20-30 IO's/sec
3 MB/sec	1MB/sec
24 cube feet	.03 cube feet

- Key Observation (Inexpensive vs. expensive disks):  
Number of I/Os differ only moderately

3

### Core of the Proposal

- Build I/O systems as ARRAYS of inexpensive disks.
  - Stripe data across multiple disks and access them in parallel to achieve both higher data transfer rates on large data accesses and...
  - higher I/O rates on small data accesses
- Idea not entirely new...
  - Prior very similar proposals [Kim 86, Livny et al, 87, Salem & Garcia-Molina 87]
- 75 inexpensive disks have potentially 12 times the I/O bandwidth of an IBM 3380 with lower power consumption and cost!

4

### RAID = Reliability May Suffer?

MTTF: mean time to failure

MTTF for a single disk unit is long..

- For IBM 3380 is estimated to be 30,000 hours (> 3 years)
- For CP 3100 is around 30,000 hours as well..

- For an array of 100 CP3100 disk the...

$$\text{MTTF} = \text{MTTF\_for\_single\_disk} / \text{Number\_of\_disk\_in\_the\_Array}$$

I.e., 30,000 / 100 = 300 hours!!! (or once a day!)

- That means that we are going to have failures very frequently

5

### A better solution: RAID

Idea: make use of extra disks for reliability!

Core contribution of paper (in comparison with prior work):

- Provide a full taxonomy (RAID-levels)
- Qualitatively outlines the workloads that are "good" for every classification
- RAID ideas are applicable to both hardware and software implementations

6

## Basis for RAID Taxonomy

- **Two RAID aspects taken into consideration:**
  - Data striping : leads to enhanced bandwidth
  - Data redundancy : leads to enhanced reliability
  - Mirroring, parity, or other encodings
  - but also throughput issues for accesses of different size, as we know ...

7

## Data Striping:

- Data striping distributes data transparently over multiple disks to make them appear as a single fast large disk. Allows multiple I/Os to happen in parallel.
- Granularity of data interleaving
  - **Fine grained (bit interleaved)**
    - Relatively small units; all I/O requests access all of disks in the disk array.
      - High transfer rates
      - Only one logical I/O request at a time
      - All disks must waste time positioning for each request: bad!
  - **Coarse grained (block-interleaved)**
    - Interleave in relatively large units so that small I/O requests only need a small number of disks while large requests can access all disks in the array

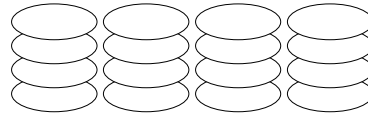
8

## Data Redundancy

- **Method for computing redundant information**
  - Parity (3,4,5), Hamming (2) or Reed-Solomon (6) codes
- **Method for distributing redundant information**
  - Concentrate on small number of disks vs. distribute uniformly across all disks
  - Uniform distribution avoids hot spots and other load balancing issues.

9

## RAID Level 0

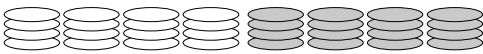


### Non-redundant RAID Level –0

- Strip sectors
- Lowest cost
- Use for low-reliability applications
- Best “write” bandwidth performance –
  - does not need to update redundant information.
- Does not necessarily have best read performance:
  - Redundancy schemes that duplicate data, such as mirroring, may perform better on reads by selectively scheduling requests on the disk with the shortest expected seek and rotational delays.

10

## Mirrored RAID Level 1



- Uses twice as many disks as a nonredundant disk array.
- Strip sectors of data.
  - Whenever data written to disk its written to a redundant one as well.
  - When retrieved, retrieve from disk with shorter queueing, seek and rotational delays. Can reduce by 45%.
- Reliability: excellent
- Trades capacity for performance (expensive!)
- Recovery? (simple)
- Appropriate applications:
  - Availability and transaction rate more important than storage efficiency.

11

## RAID Levels 2 and 3

- **Parallel Access**
  - All member disks participate in every I/O request.
  - Spindles synchronized.
- **Data Striping Used**
  - Small strips

12

### RAID 2

- Goal: improve cost of recovery from failed components
- Striping is done at the word level.
- Use Memory Error Correcting Codes (ECC) [Hamming 50]
  - Error-correcting code calculated across corresponding bits on each data disk; bits of code stored in corresponding bit positions on multiple parity disks.
  - Hamming code: corrects single-bit errors and detect double-bit errors.
- On a single read/write, all disks accessed.
- For large writes, as good as 1. Appropriate for supercomputers
- Bad for small data transfers
- Still costly.
  - # of redundant disks = log(total number of disks). More efficient as number of data disks increases.
- Recovery: If a disk fails, several of the parity component will have inconsistent values, and the failed component is the one held in common by each incorrect subset.
  - Multiple redundant disks needed to identify the failed disk; only one needed for recovery.
- Drives have to be rotationally synchronized in order to get the benefits!

13

### RAID Level 3

- Simplified, cheaper version of RAID 2
  - Requires only a single redundant disk
  - Employs parallel access, with data distributed in small strips.
- Most disk controllers can tell if the disk failed.
- Bit-interleaved parity
- Each read request reads from all data disks and each write request
- Cannot do much about the "random" error
- But it can easily protect against a bad drive.. (Recovery is easy).
- Very high-grained interleaving

- RAID 2 and 3 are of practical value mostly for supercomputer applications – high bandwidth
- Simpler to implement than levels 4,5,6(although they complete the discussion in terms of RAID levels)

14

### RAID - Level 4 Block-Interleaved Parity

- Striping entity is a block
  - Data interleaved across disks in blocks of arbitrary size rather than in bits.
  - Size of blocks = "striping unit."
- Read requests smaller than the striping unit access only a single data disk.
- Block-level parity used
- Write requests must update the requested data blocks and must compute and update the parity block.
  - For Large writes that touch blocks on all disks, calculate parity directly.
  - For small write requests that update only one data disk, parity computed by noting differences.
- No synchronized drives required to read a block..
- If a disk crashes, then recovery is straightforward.
- Small write requests require 4 disk I/Os.
- Parity disk gets most of the "traffic" (bottleneck)

15

### RAID Level-5 Block-Interleaved Distributed Parity.

- Distribute parity blocks
- No single disk becomes the bottleneck
- Best small read, large read and large write performance of any redundant disk arrays.
- Still high cost for small WRITES
- Reconstruction of a failed drive is not as easy as in other levels

16

### RAID Level 6 P+Q Redundancy

- Parity can correct any single self-identifying failure.
  - As disk arrays get larger, multiple failures are possible, stronger codes needed.
  - When a disk fails in a parity-protected disk array, recovering contents of failed disk requires successful reading of contents of all nonfailed disks.
    - Probability of a random bit error is high.
  - Need stronger error-correcting codes.
- P+Q Redundancy
- Reed-Solomon codes to protect against 2 failures using 2 disks. Otherwise similar RAID-5

### Small Writes in RAID 5: why it is a Problem

A single WRITE is translated into five operations:

1. READ old data
2. READ old parity block
3. Compute XOR (new data with old data block)
4. WRITE new data block
5. WRITE new parity block

18

## Solving Small Write Problem

[Stodolsky, Gibson, 1993]

- **Parity Logging:**
  - Increases throughput for workloads emphasizing small, random write accesses in a redundant disk array by logging changes to parity in a segmented log for efficient application later.
  - Log segmentation allows log operations that are large enough to be efficient yet small enough to allow in-memory application of a log segment.

19

## Issues

- What happens when new disks are added into the system?
  - Got to change layout
  - Got to rearrange data
- In addition, many spare disks may be wasted..
- Solution to the above is the HP Auto RAID [AutoRAID, SOSP 95]
  - Core idea:
    - mirror active data (hot)
    - RAID-5 for not very active data (cold)
  - Assumptions:
    - Only part of data is “active” any any moment in time.
    - Working set changes slowly (so that migration is enabled).

20