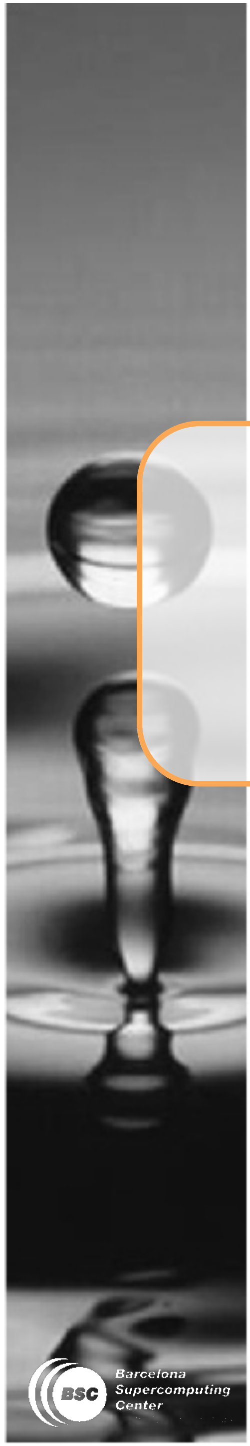


Adapting RAID technology to large heterogeneous clusters

Toni Cortes

Barcelona Supercomputing Center

Storage
System
Research
Group

Special thanks to

- **José Luis Gonzalez**
  - Universitat Politècnica de Catalunya
- **Prof. Dr. Jesús Labarta**
  - Universitat Politècnica de Catalunya
  - Barcelona Supercomputing Center

Raid 0

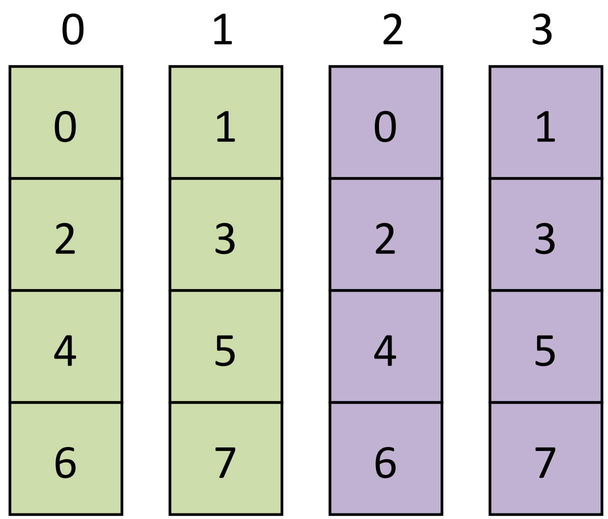
- **Data distribution**
  - Round-robin on all discs
- **Advantages**
  - Highest bandwidth
  - Highest capacity
- **Disadvantage**
  - Not tolerant to any failure

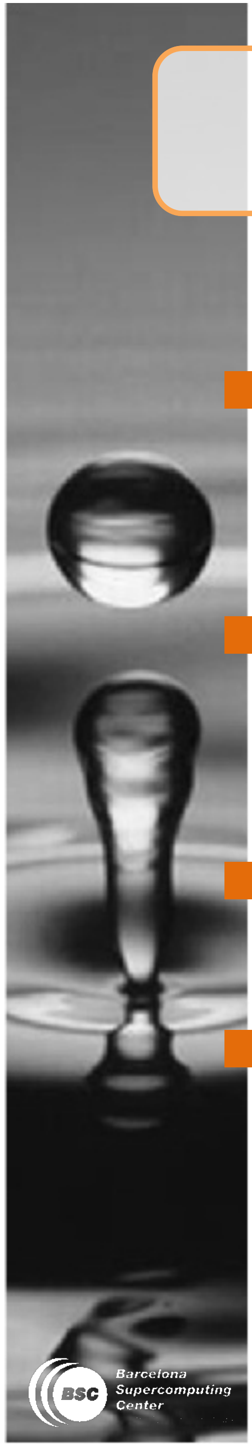
0	1	2	3
0	1	2	3
4	5	6	6
8	9	10	11
12	13	14	15

Storage System Research Group

Raid 1

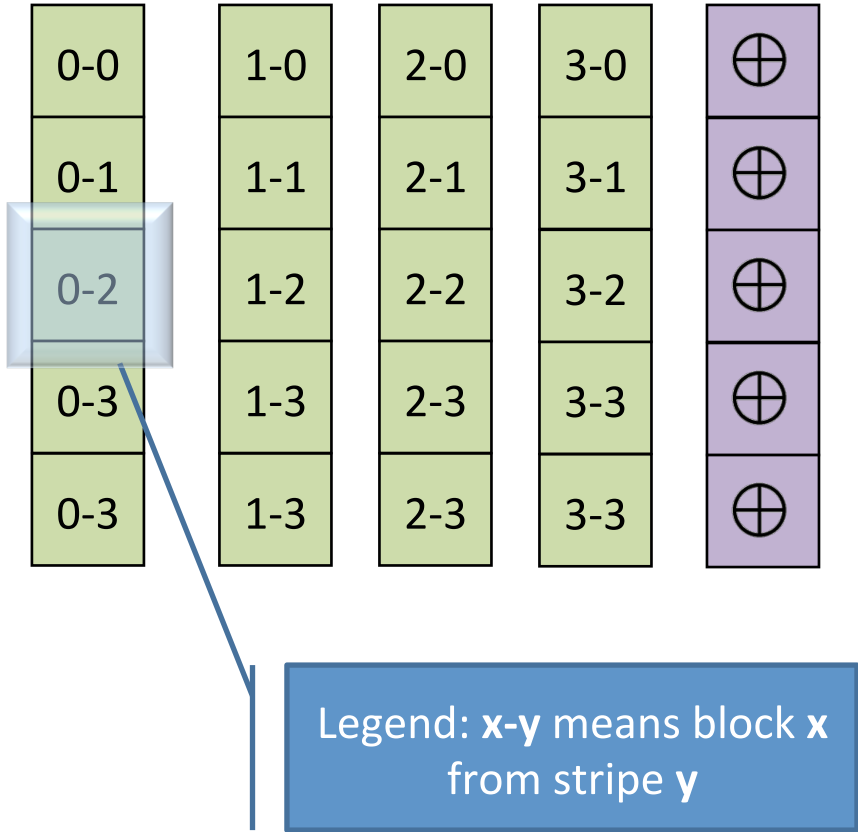
- **Data distribution**
  - Two replicated RAID0
- **Writes**
  - Done on both copies
- **Reads**
  - From any of the copies
  - Possible optimizations
- **Advantages**
  - Fault tolerant
- **Disadvantages**
  - Less parallelism
  - Wasted space





Raid 4

- **Data distribution**
  - RAID0 plus a parity discs
  - Parity computed using XOR
- **Writes**
  - Modified data
  - Newly parity block
- **Advantages**
  - Allows one disc failure
- **Disadvantage**
  - No write parallelism
  - Always use parity disc

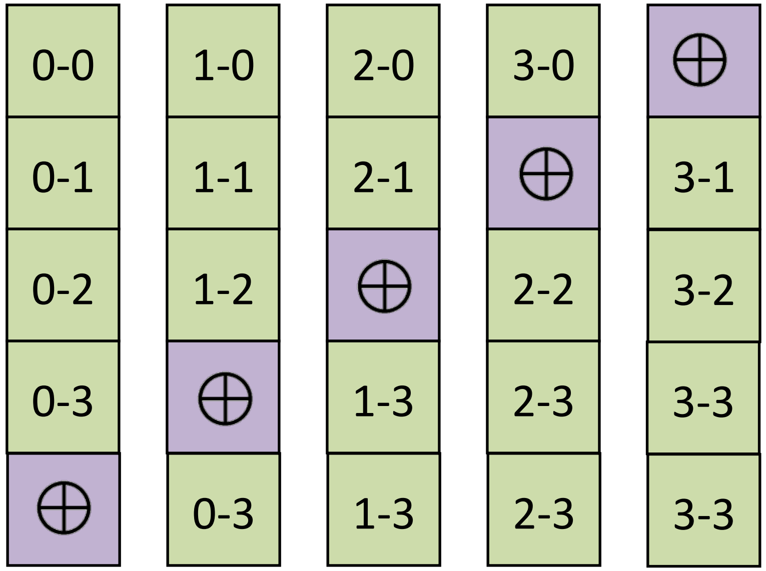


Terminology

- **Striping unit**
  - Block used to distribute data
- **Stripe**
  - Set of striping units that share parity computation
- **Parity block**
  - Block that keeps the "parity" of a stripe
    - Same size as a striping unit

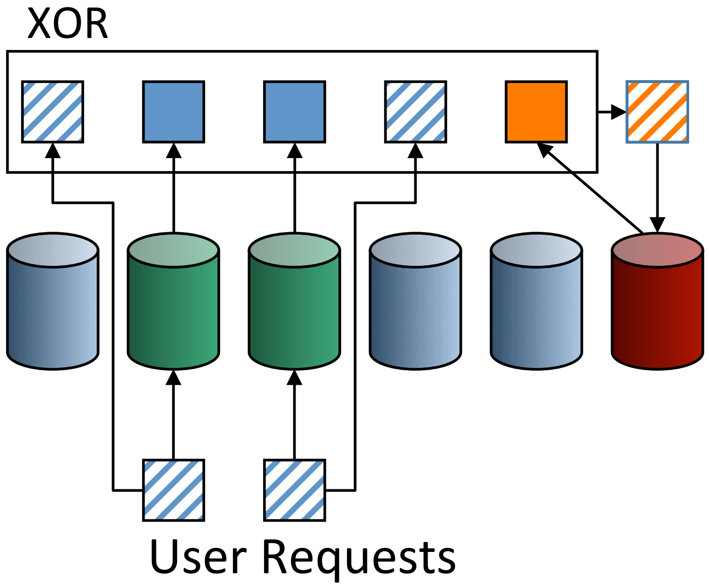
Raid5

- **Data distribution**
  - RAID4 plus interleaved parity
- **Advantages**
  - Good performance
  - Similar to RAID0
- **Disadvantage**
  - Only allows one disc failure
- **Slow small writes**
  - More later ☺



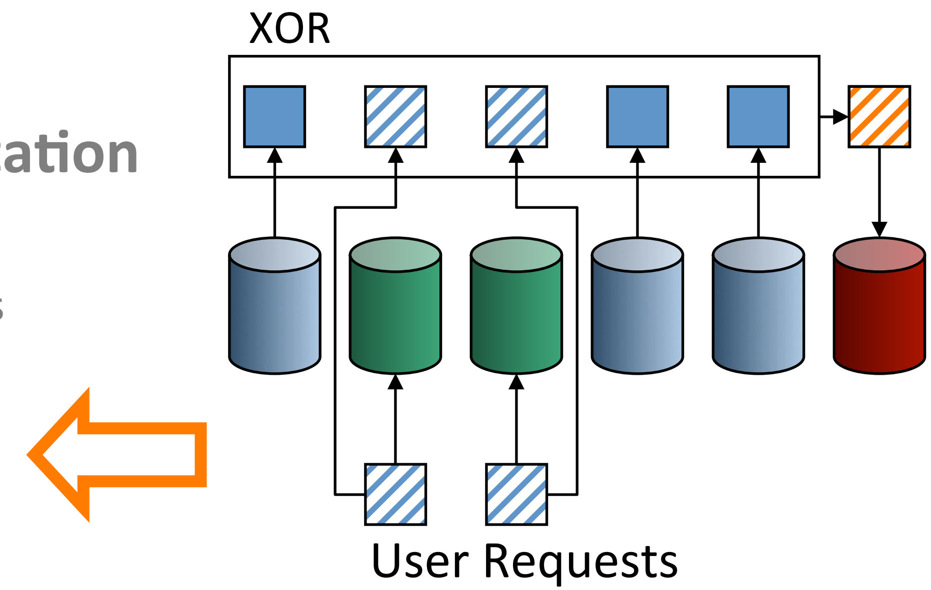
Parity Computation

- **Writing performance**
  - Full stripe ➔ Efficient
  - Small write ➔ Problem
- **"Small writes" implementation**
  - Read-write-modify
    - Better for "small" requests
  - Regenerate write
    - Better for "large" requests



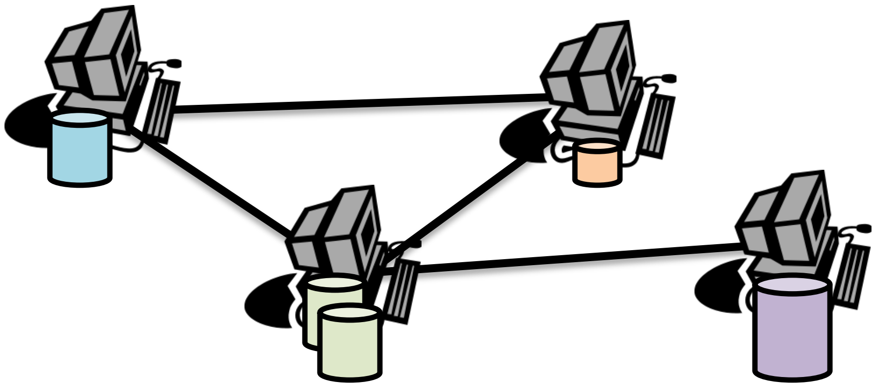
Parity Computation

- **Writing performance**
  - Full stripe ➔ Efficient
  - Small write ➔ Problem
- **"Small writes" implementation**
  - Read-write-modify
    - Better for "small" requests
  - Regenerate write
    - Better for "large" requests



Why is heterogeneity an issue?

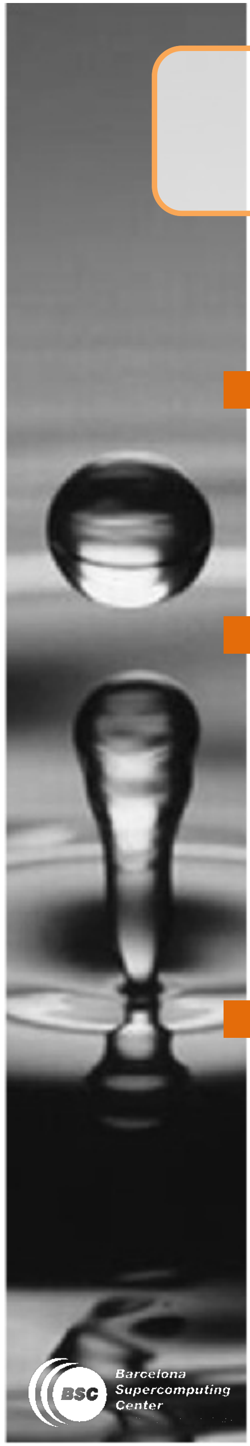
- **Definition**
  - A heterogeneous set of disks is a set of disks with **different performance and capacity characteristics**



- **They are becoming a common configuration**
  - Replacing an old disk
  - Adding new disks
  - Cluster build from already existing (heterogeneous) components

Barcelona Supercomputing Center

Storage System Research Group

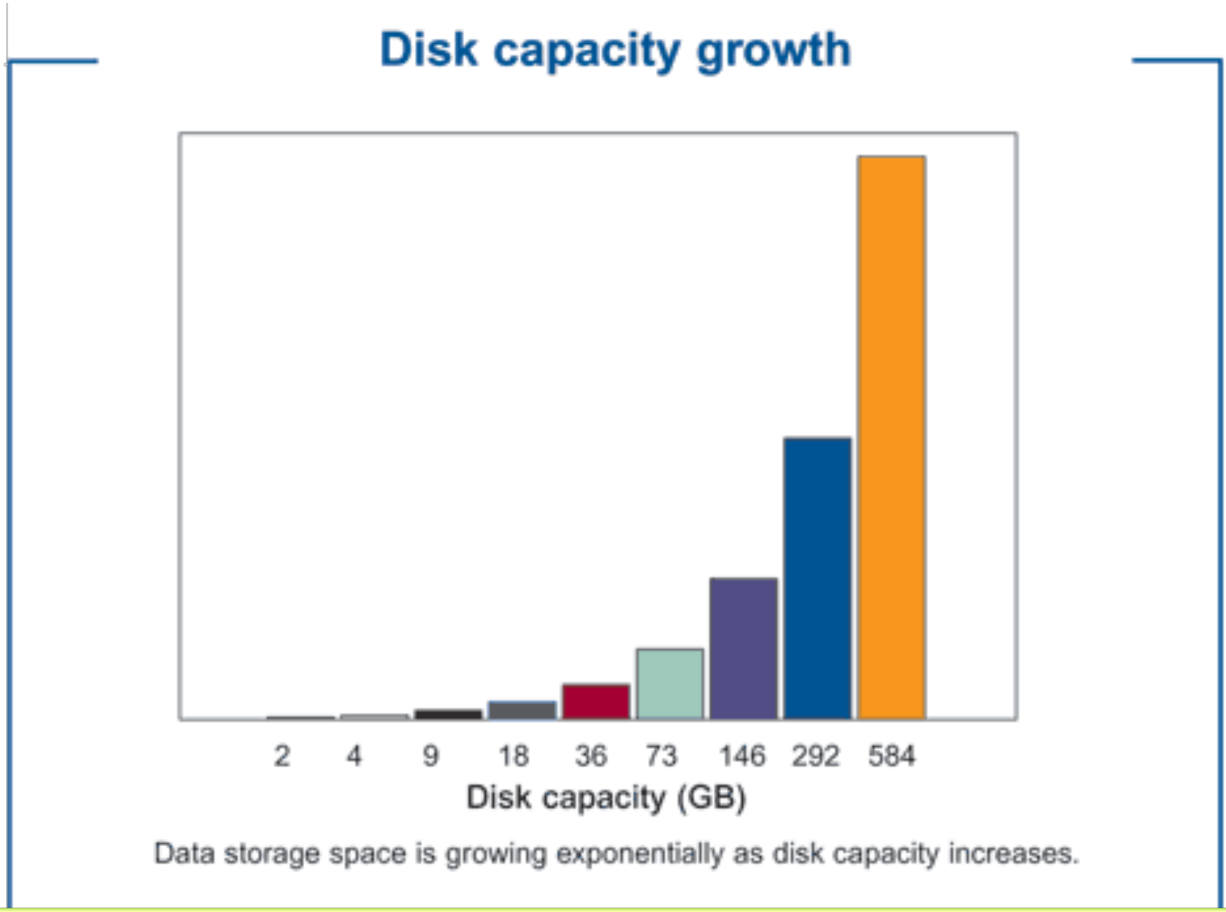


Traditional solution

- **Many systems just ignore it: all disks are treated as equal**
  - The usable size of all disks is like the smallest one
  - The performance of all disks is assumed as the slowest one
- **Implications**
  - No performance gain is obtained
    - Except for some implicit side effect
  - Not all potential capacity gain is obtained
    - Some systems use the unused disk space to build a virtual disk
- **Objective of this talk**
  - Show how to handle heterogeneous sets of disks
  - Show how to handle scalability
    - Specially how to grow storage systems with minimum overhead

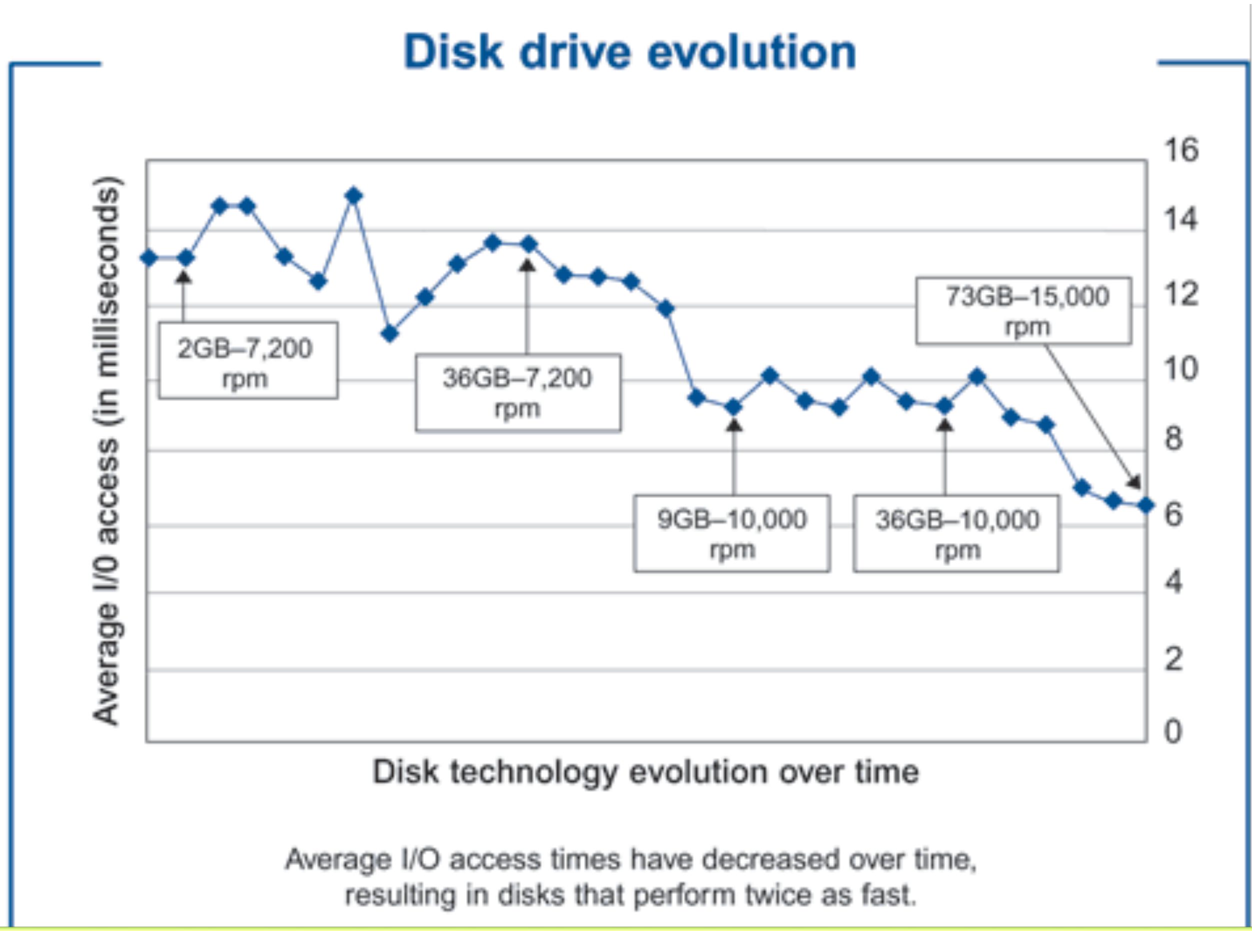
Storage System Research Group

Disk capacity



THE DATA STORAGE EVOLUTION. Has disk capacity outgrown its usefulness? by Ron Yellin (Terada magazine 2006)

Disk performance

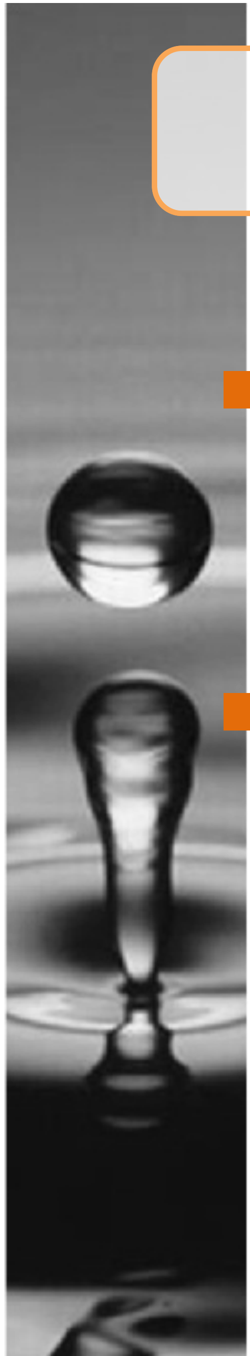


THE DATA STORAGE EVOLUTION. Has disk capacity outgrown its usefulness? by Ron Yellin (Terada magazine 2006)

Growth storage needs

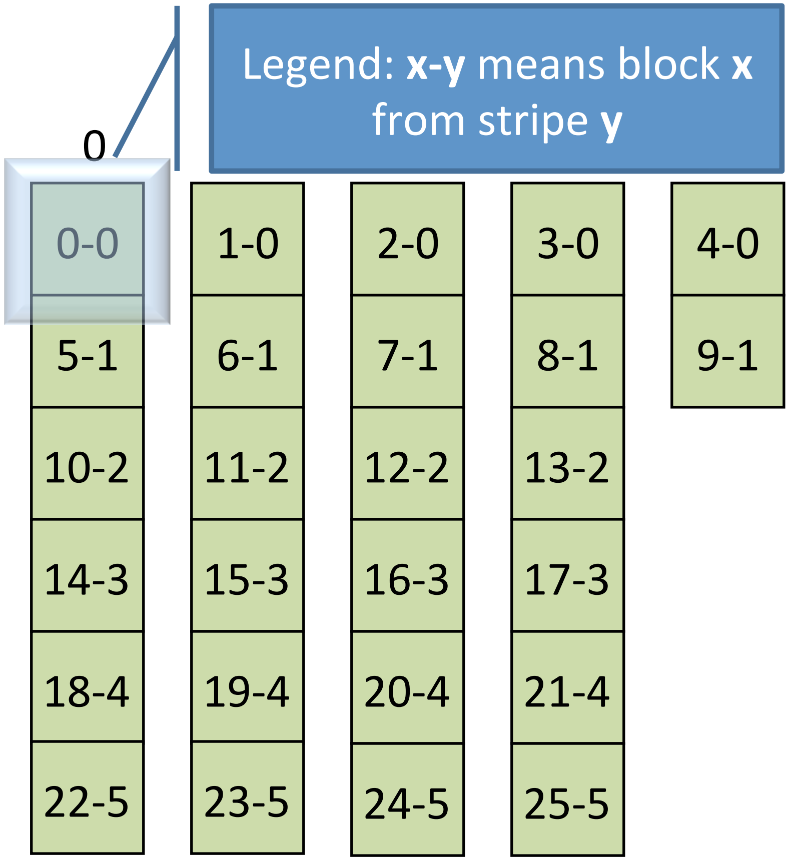
- **Information point of view**
  - *Increase of 30% each year*
    - How much information 2003?
      Peter Lyman and Hal R. Varian
      School of Information Management and Systems
      University of California at Berkeley
- **Manufacturers point of view**
  - *Increase capacity 50% each year*
    - Drive manufacturers
    - THE DATA STORAGE EVOLUTION. Has disk capacity outgrown its usefulness?
      by Ron Yellin
      Terada magazine 2006

Storage System Research Group



AdaptRaid0: Intuitive idea

- **Place more blocks on larger disks**
  - Assumed to be faster
  - Stripes with different sizes
- **Problem**
  - Variable parallelism
    - Long stripes in low @
    - Short stripes in high @



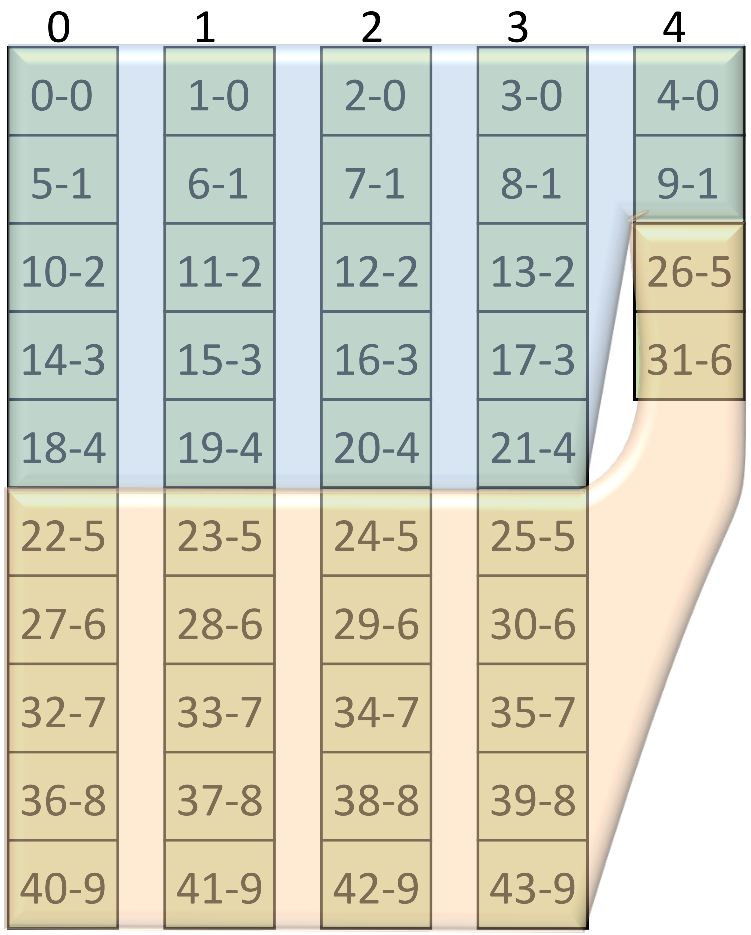
See T. Cortes and J. Labarta. Taking Advantage of Heterogeneity in Disk Arrays

Barcelona Supercomputing Center

Storage System Research Group

Reducing variance in parallelism

- **Use distribution like a repetition pattern**
  - All areas have short and long stripes
  - Hopefully, most large files will too



Storage System Research Group

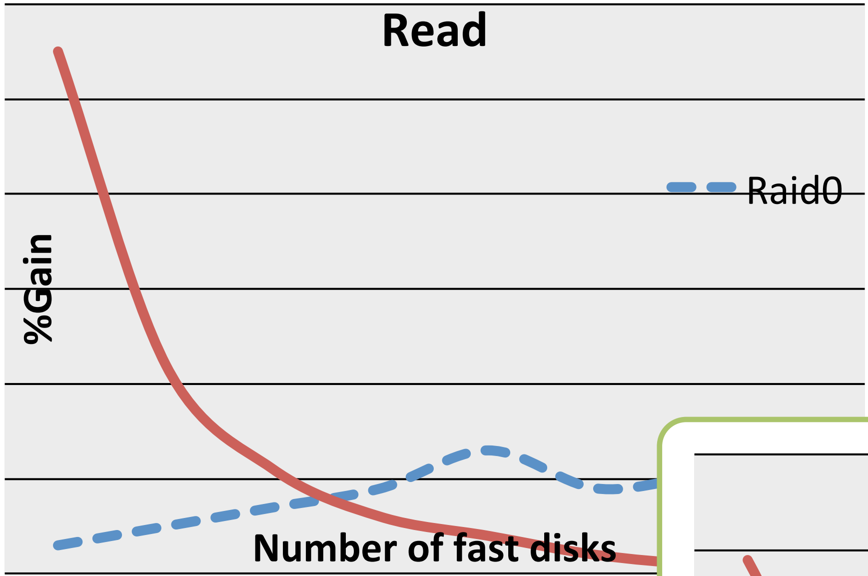
AdaptRaid0 parameters

- **Utilization factor (UF)**
  - One factor per disk
    - Larger disks have more blocks?
    - Faster disks have more blocks?
- **Stripes in pattern (SIP)**
  - We define a pattern using the UF
    - Large patterns allow more requests with good disks
    - Small patterns allow a better distribution

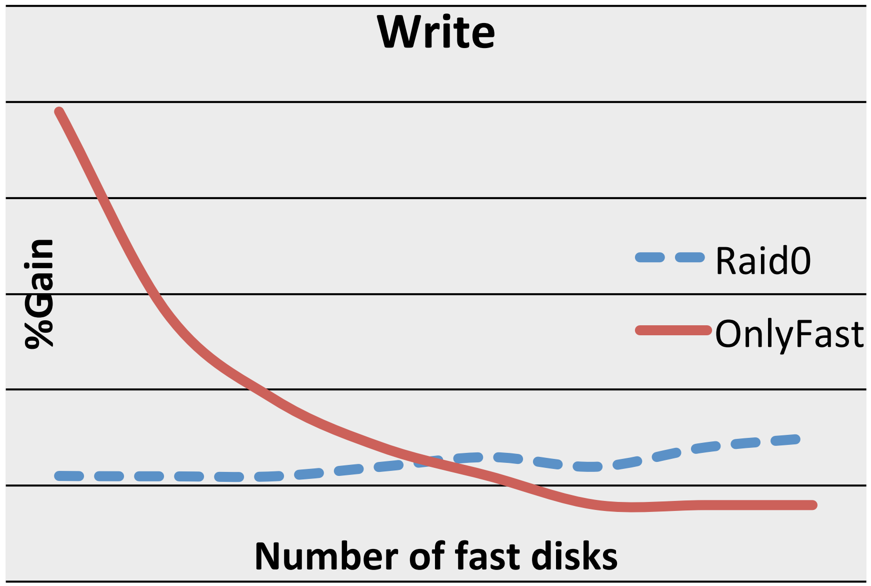
Computing the location of a block

- **Formulas**
  - *Disk(B)* =
    location[B % Blks_in_a_pattern].disk
  - *Pos(B)* =
    location[B % Blks_in_a_pattern].pos +
    (B/Blks_in_a_pattern) *
    Blks_per_disk_in_patern[*Disk(B)*]
- **Metadata**
  - Location[Blks_in_a_pattern] // [SIP * DSKS]
    - int disk, pos
  - Blks_per_disk_in_patterns[DSKS]

Performance



Using HP traces from 1999

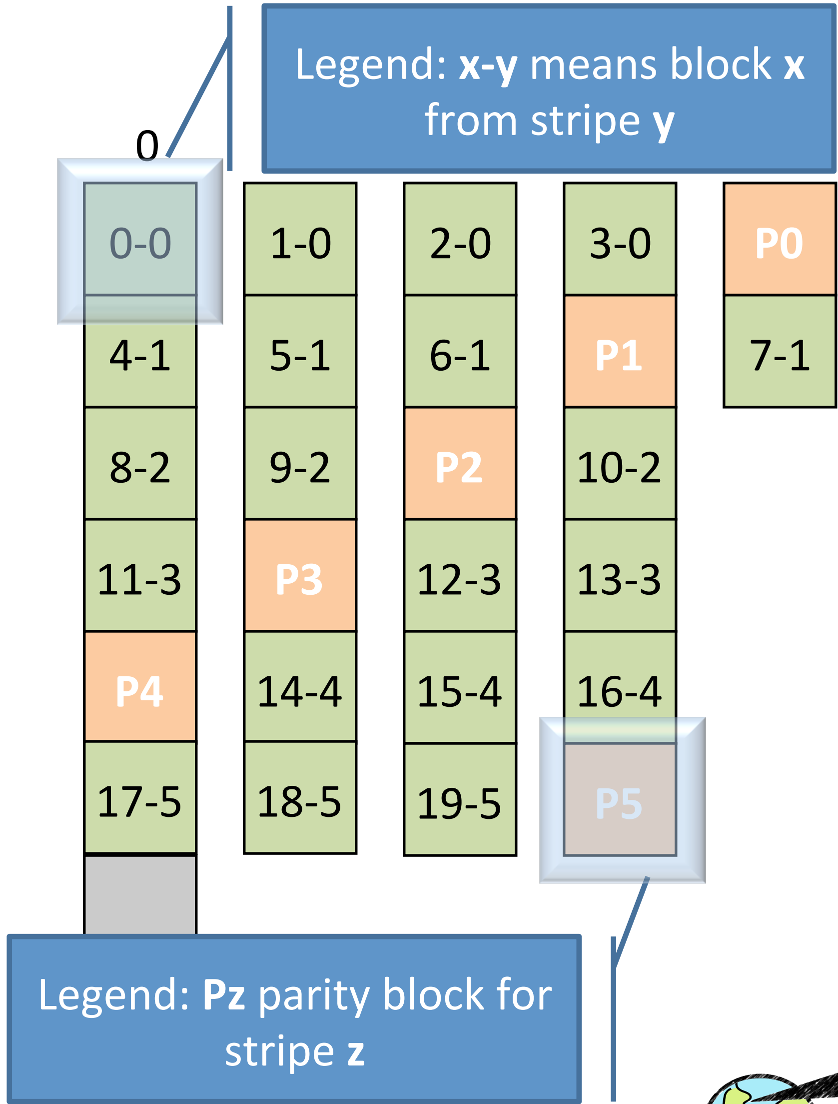


Parameter sensitivity

- **Utilization factor (UF)**
  - Depends on what the administrator wants
- **Strips in pattern (SIP)**
  - No big difference between the different values
  - The best option is SIP larger than DISKS*2
    - Measured experimentally

AdaptRaid5: Intuitive idea

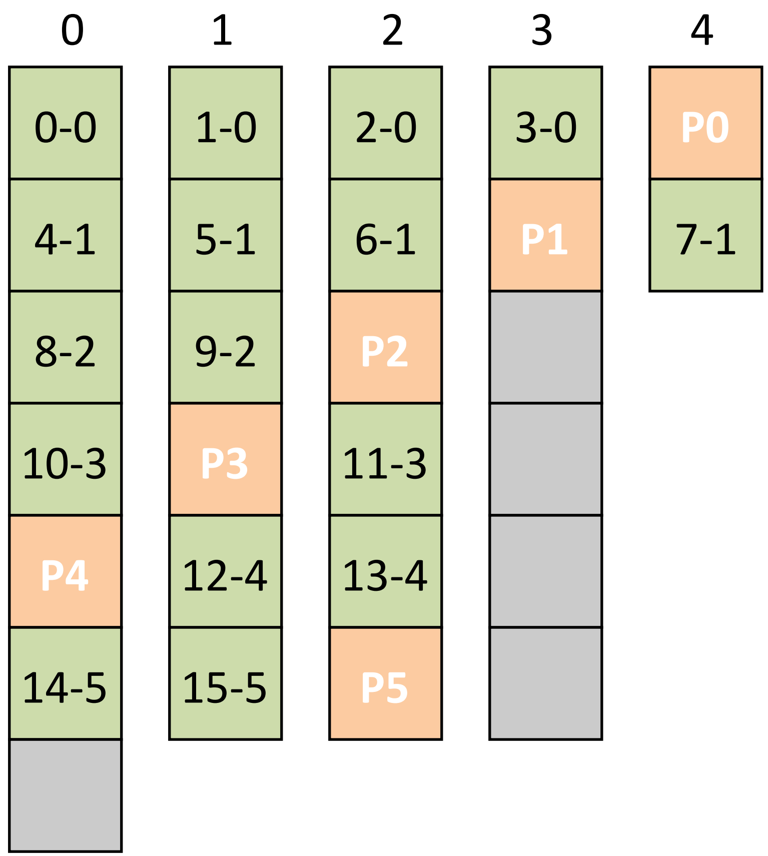
- **Place more blocks on larger disks**
  - Assumed to be faster
  - Stripes with different sizes
- **Singularities**
  - Last block disk 0 unused
    - Stripes need 2 blocks
- **Problem**
  - Small-writes



See T. Cortes and J. Labarta. Taking Advantage of Heterogeneity in Disk Arrays

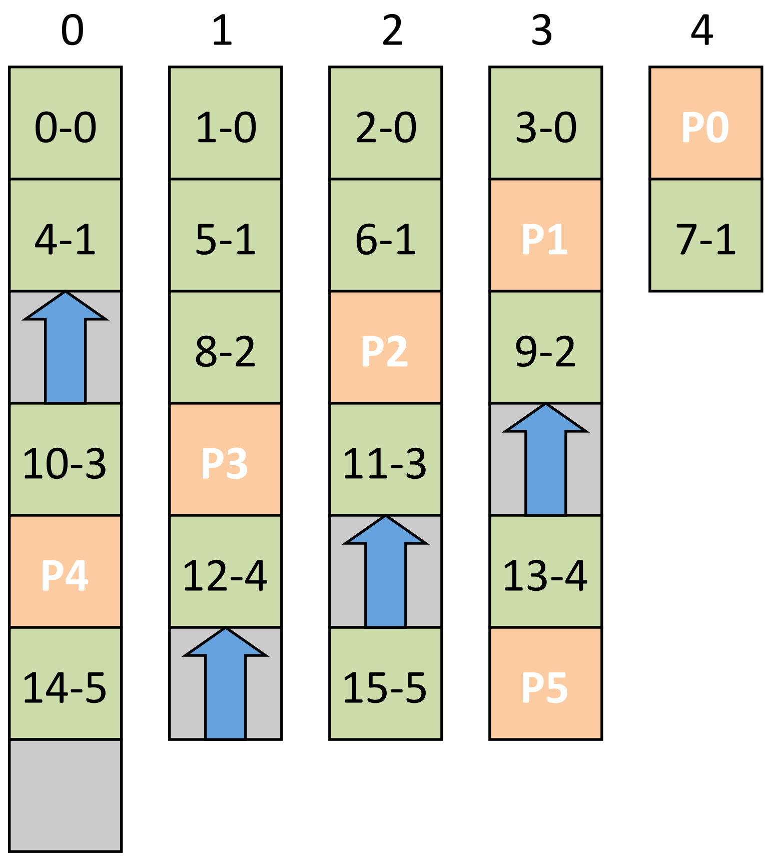
Reducing the small-write problem

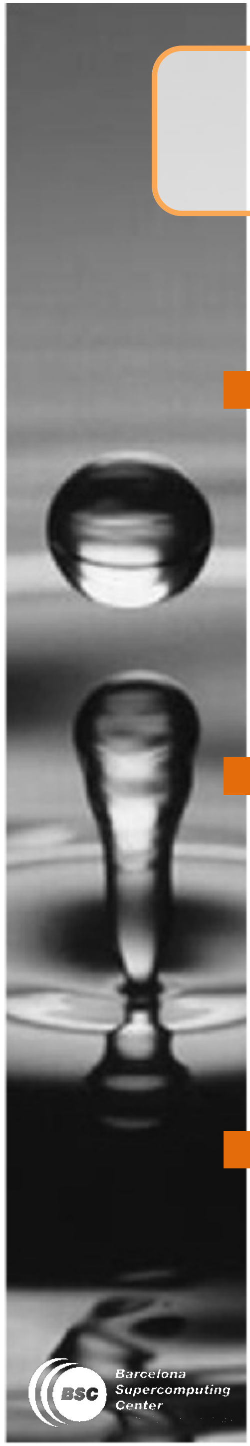
- Define the number of data blocks per stripe
  - Divisor of the number of data blocks in largest stripe
- **Problem**
  - Capacity wasted



Increasing effective capacity

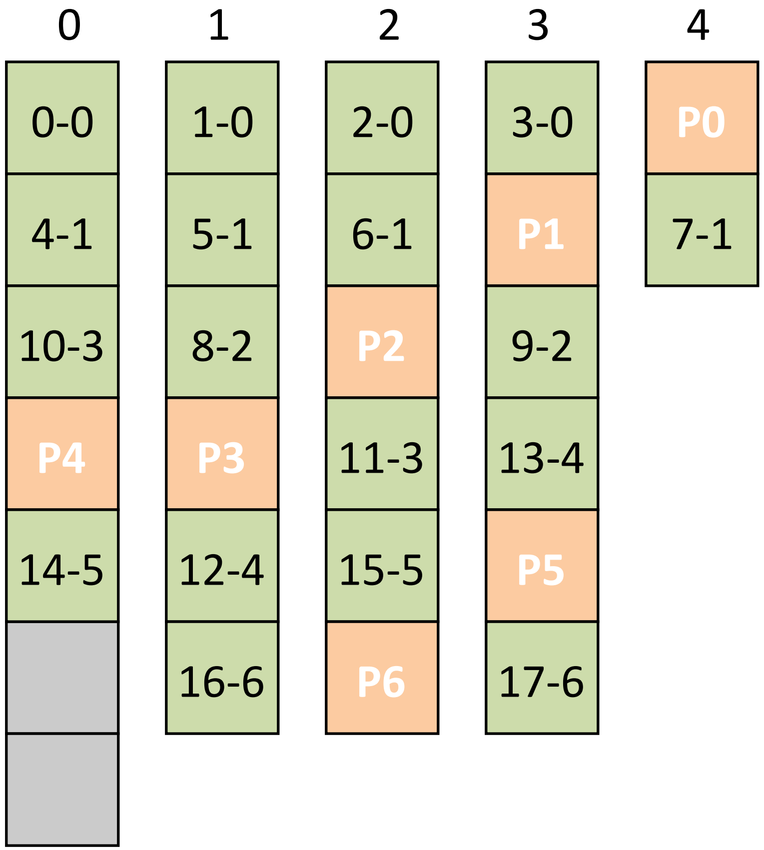
- **Step 1**
  - Use all disks leaving the unused block in RR way
    - Holland and Gibson 92
  - Better load distribution
  - No capacity is gained
- **Step 2**
  - Push blocks down "tetris-like"
- **Problem**
  - Variable parallelism





Increasing effective capacity

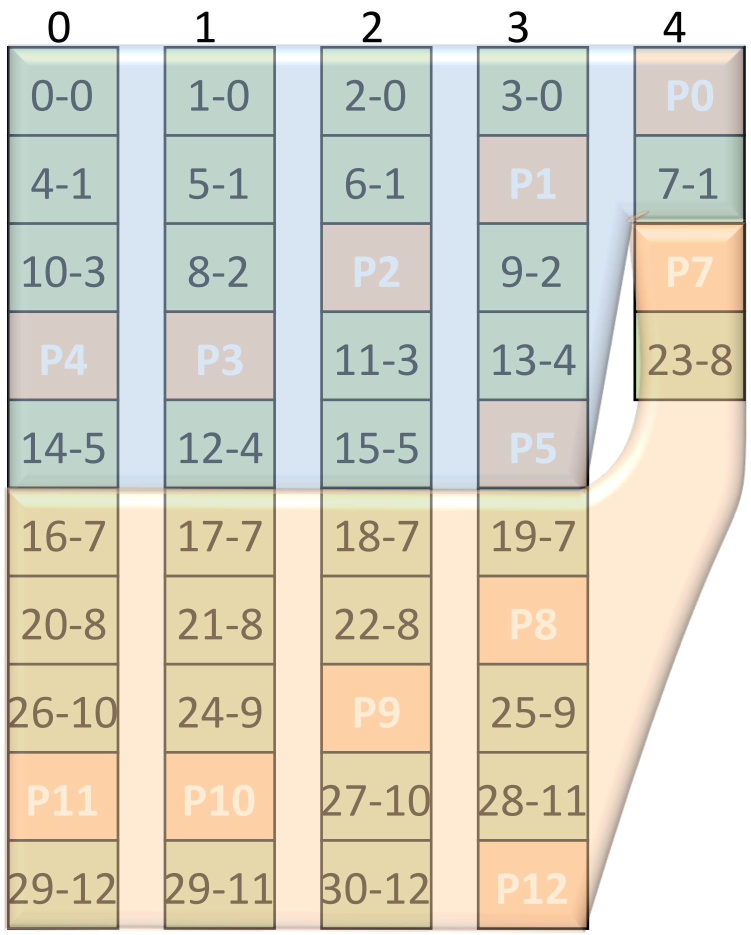
- **Step 1**
  - Use all disks leaving the unused block in RR way
  - Better load distribution
  - No capacity is gained
- **Step 2**
  - Push blocks down "tetris-like"
  - Similar to parity declustering
- **Problem**
  - Variable parallelism
    - Long stripes in low @
    - Short stripes in high @



Storage System Research Group

Reducing variance in parallelism

- Use distribution like a repetition pattern
- **Problem**
  - Minor problems with long/ short stripes
  - Too detailed for this tutorial
    - Ask me if interested



Storage System Research Group

AdaptRaid5 parameters

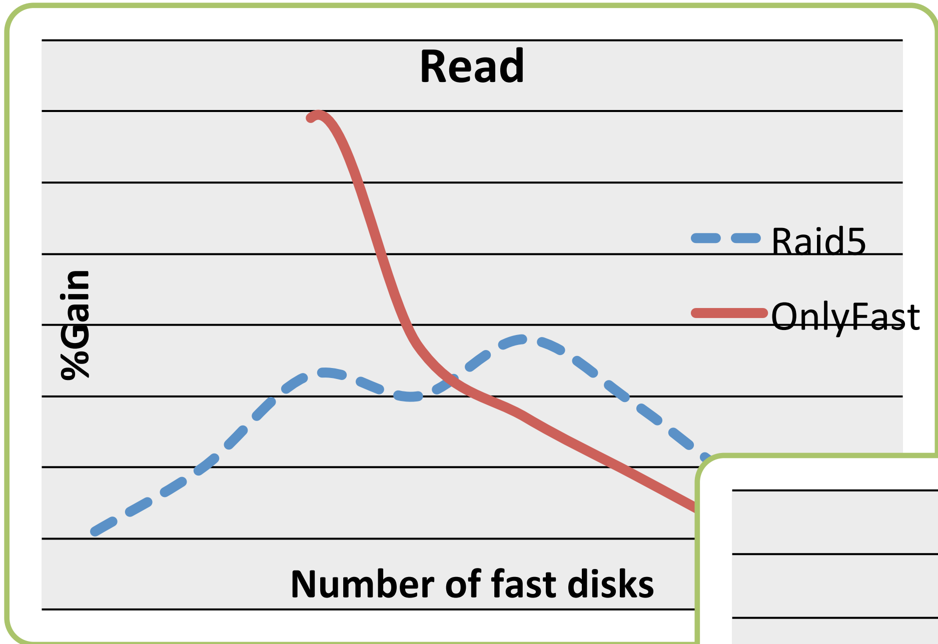
- **Utilization factor (UF)**
  - One factor per disk
    - Larger disks have more blocks?
    - Faster disks have more blocks?
- **Stripes in pattern (SIP)**
  - We define a pattern using the UF
    - Large patterns allow more requests with good disks
    - Small patterns allow a better distribution

Storage
System
Research
Group

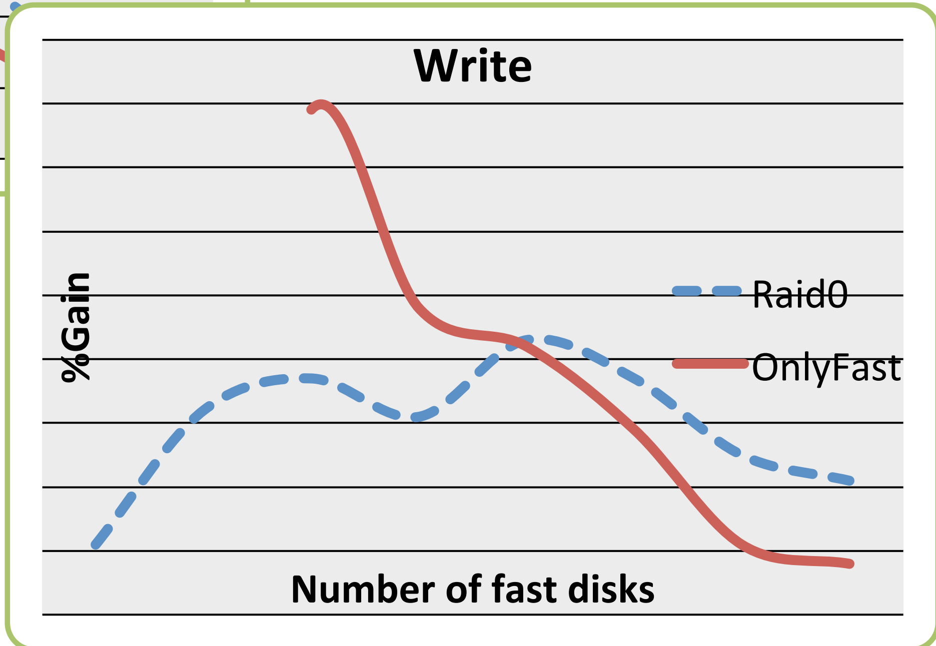
Computing the location of a block

- **Formulas**
  - *S* =
    stripe[B % Blks_in_a_pattern**]** +
    (B / Blks_in_a_pattern) * SIP
  - *Disk(Parity of S)* =
    parity[S % SIP]**.disk**
  - *Pos(Parity of S)* =
    parity[S % SIP**].pos** +
    (S/SIP) *
    Blks_per_disk_in_patern[*Disk(Parity of S)***]**
- **Metadata**
  - Stripe[Blks_in_a_pattern] // [SIP * DSKS]
  - Parity[SIP]
    - int disk, pos

Performance



Using HP traces from 1999



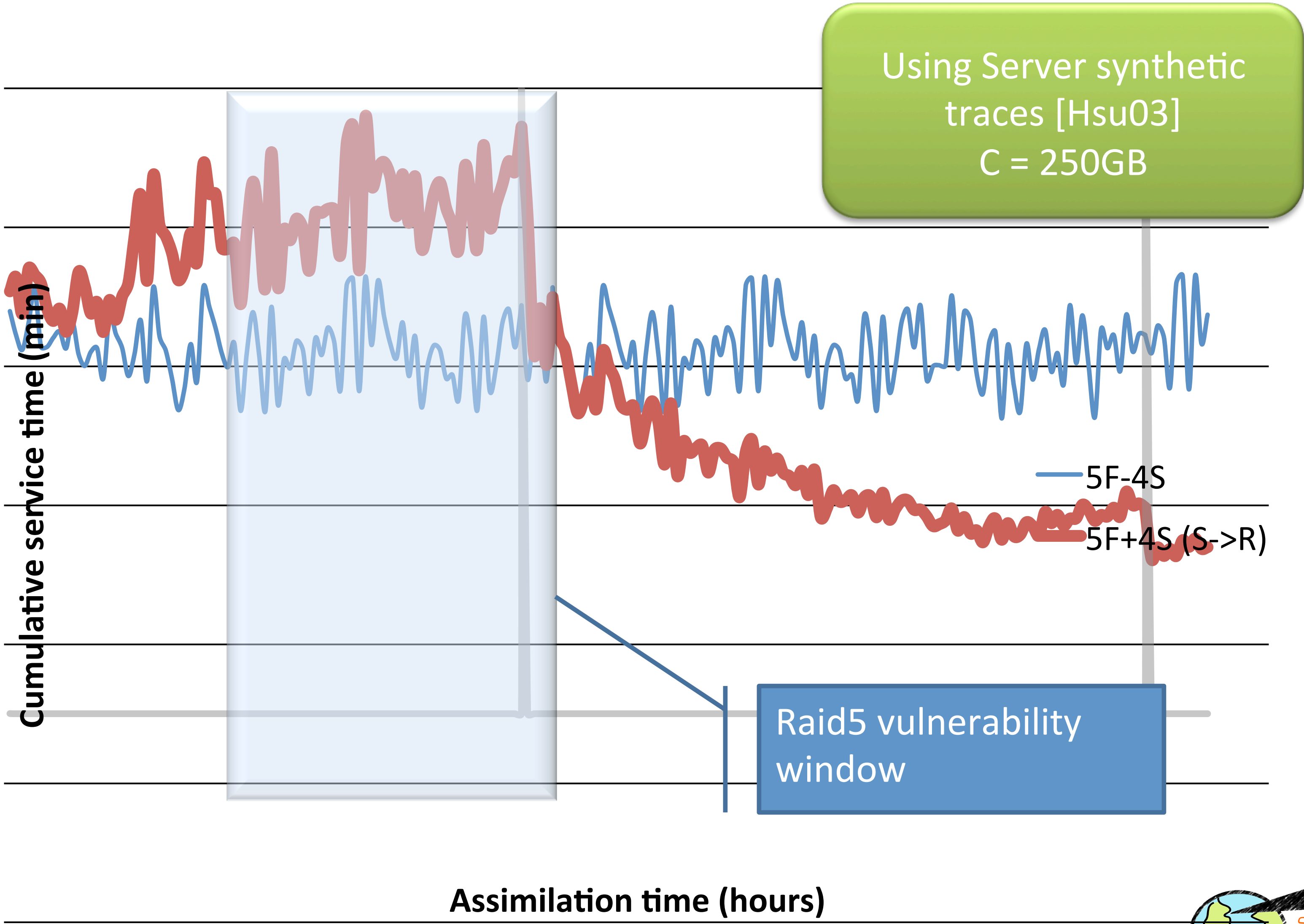
Parameter sensitivity

- **Utilization factor (UF)**
  - Depends on what the administrator wants
- **Strips in pattern (SIP)**
  - There is big difference between the different values
    - Especially for writes
      - Up to 5 times slower with small SIPs
      - More sensible for parity distribution
  - The best option is a SIP larger than DISKS*2
    - Measured experimentally

Recovering a disk

- **Performed in two steps**
  - Recovery of lost data (like in RAID5)
    - During this step no other disk can fail (**vulnerability window**)
  - Reorganization to improve disk usage
    - During this step a disk failure would not be fatal
- **Vulnerability window comparison**
  - In heterogeneous arrays reduced up to 30% (depending on load)
    - Disk are used better → reconstruction is faster
    - Some kind of parity declustering → reconstruction faster

Overhead for disk recovery



See J. L. Gonzalez, T. Cortes Evaluating the Effects of Upgrading Heterogeneous Disk Arrays

Scaling RAID architectures

- **Using traditional RAID architecture does not scale**
  - Including AdaptRAID
- **Adding news disk implies reorganizing the whole data**
  - Re-striping requires the movement of all data-blocks
  - Time $t_{striping}$ for re-layout grows linear in capacity:

$$t_{striping} = k * C_{old}$$

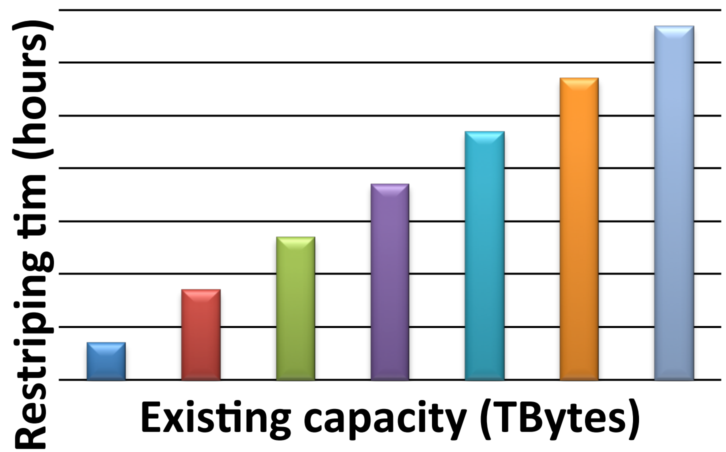
where k is a constant and C_{old} is the already stored capacity

- **Trend**
  - Newly integrated capacity C_{new} is always smaller than C_{old}

Storage System Research Group

How expensive is re-striping?

- **Assumptions**
  - 36 GByte of data can be re-distributed in each hour
  - 100 GByte of new capacity C_{new} have to added
  - Already existing capacity C_{old} between 100 GByte and 1 PByte

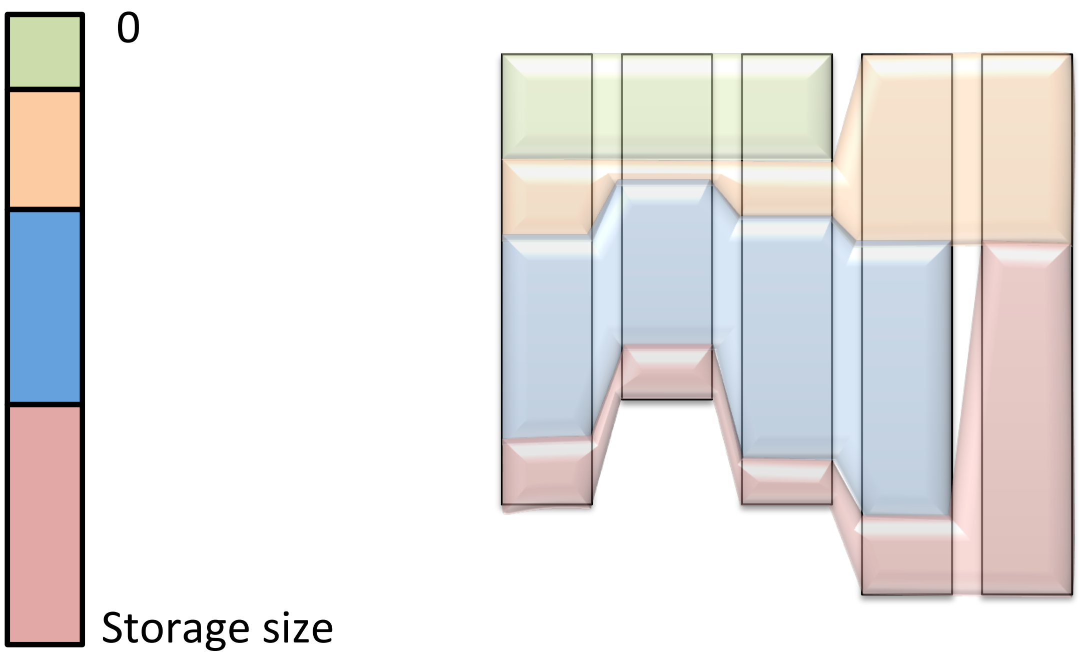


If re-striping is not the way …

- **Objective: find a way to**
  - Only migrate the needed amount of data
  - Continue having balanced load
  - Do not lose the deterministic behavior

AdaptiveZ: intuitive idea

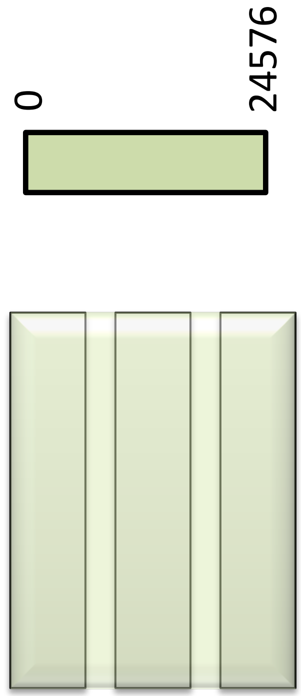
- **Divide the address space in zones**
  - Created dynamically each time new disks are added
  - Each zone has its own heterogeneous stripping “policy”
- **When new disks are added, only one zone is restriped**



See J. L. Gonzalez, T. Cortes. An Adaptive Data Block Placement based on Deterministic Zones (AdaptiveZ)

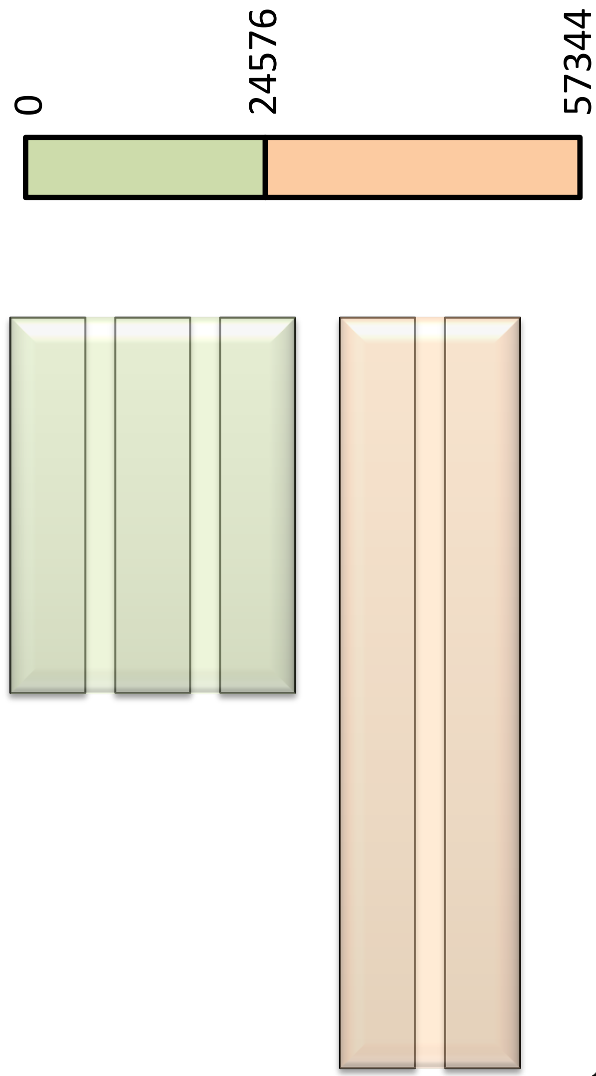
Initial setup

- **Let's assume a first array**
  - 3 disks 1Gbytes each
  - Striping units → 128KB
    - 8192 per disk
    - 24576 in total
  - **Initially we have one zone**



Adding 2 disks: naïve way

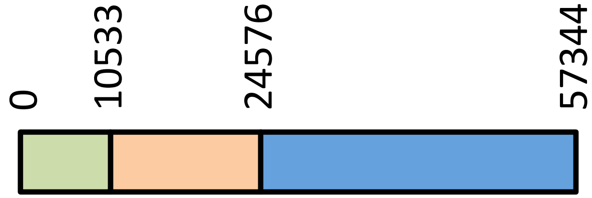
- **Adding 2 disks**
  - 2 disks 2Gbytes each
  - Striping units → 128KB
    - 16384 per disk
    - 32768 of new storage
    - 57344 final capacity
- **Create a new zone with only two disks**
  - No data movement
- **Problem**
  - NOT balanced
    - Only new data in new disks
  - No increase in parallelism

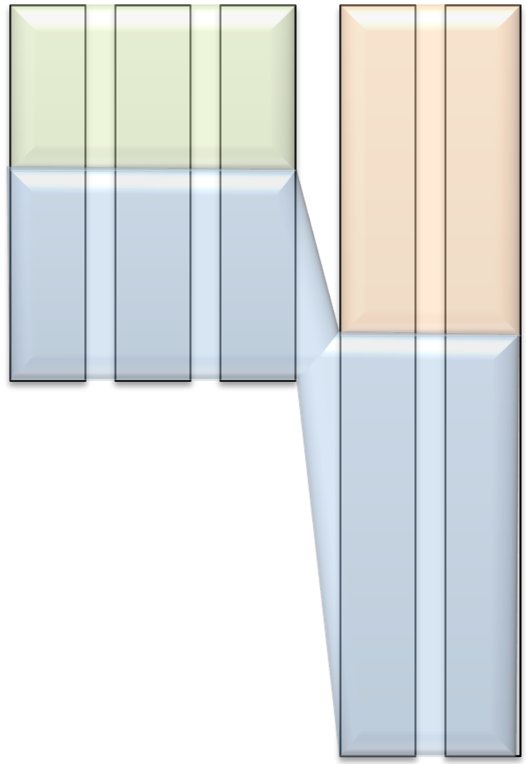


Storage System Research Group

Adding 2 disks: load balancing

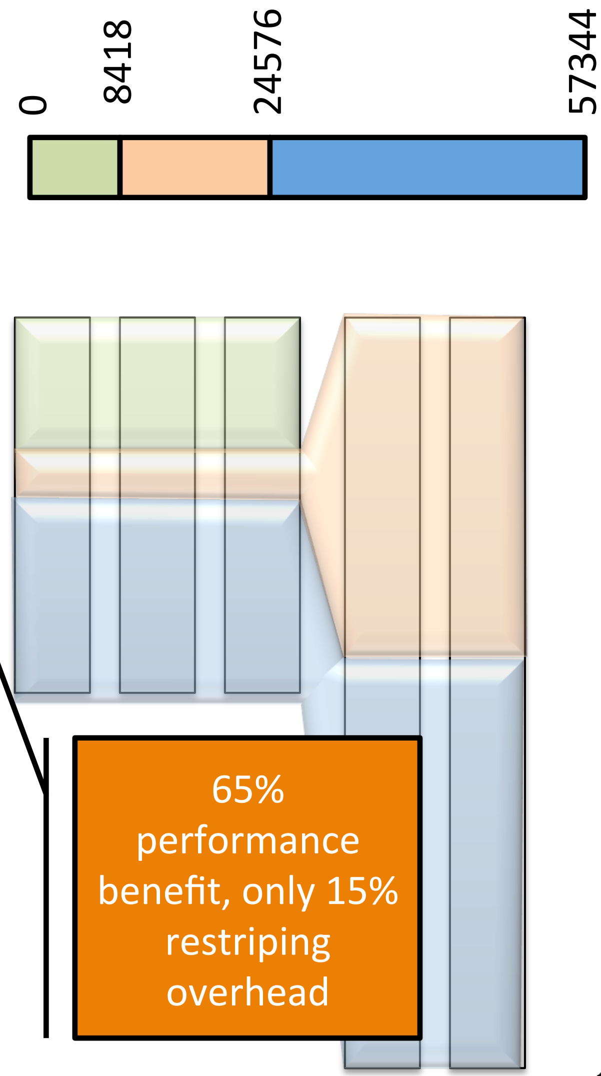
- Create two new zones
- Zone1: old data
  - Theoretical minimum movement to balance load
    - Assume new disks should have 2 times more SU
    - $C_{old}*(1-C_{old}/(C_{new} + C_{old}))$
      - 14043 SU
- Zone2: new data
  - Uses all disks
  - Only for new data
- **Problem**
  - Old data looses parallelism





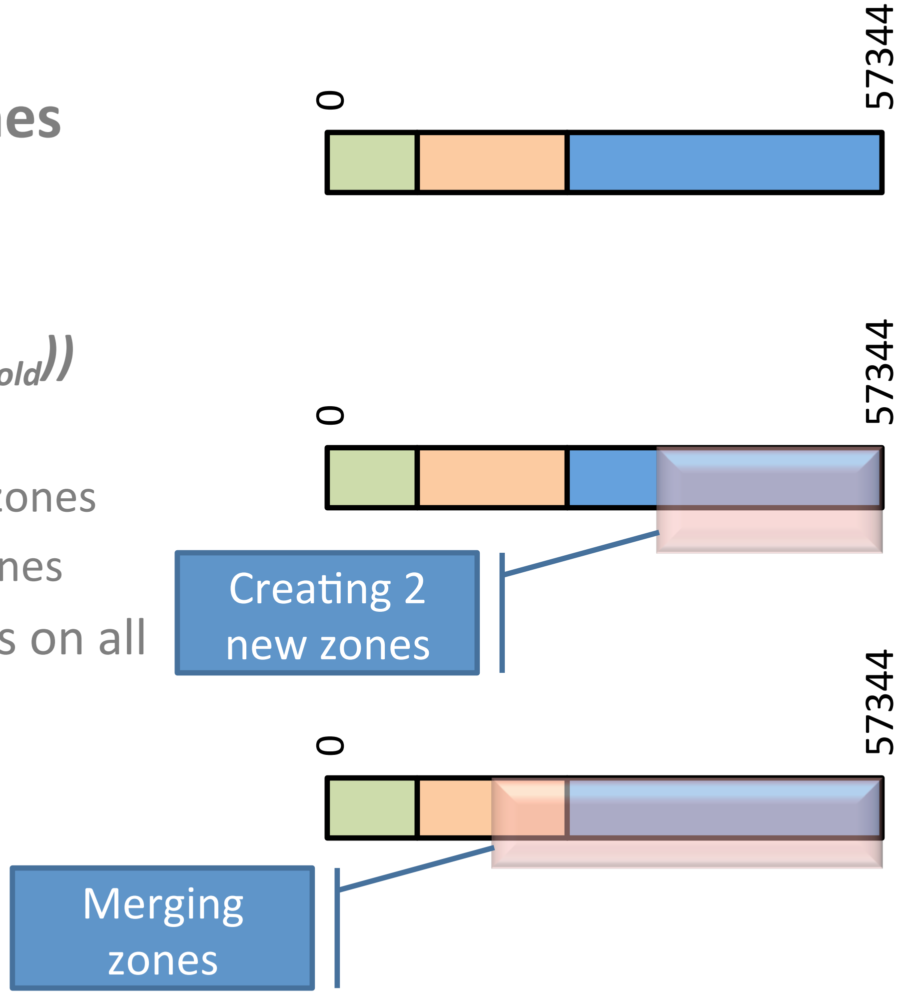
Adding 2 disks: more parallelism

- Create two new zones
- Zone1: old data
  - Increase 15% the theoretical minimum
    - ***1.15 * $C_{old}*(1-C_{old}/(C_{new}+C_{old}))$***
      - 16149 SU
  - Restripe these blocks on all disks
    - Find the adequate UF for this zone to guaranty global disk UF
    - **Small compared to full restripe**
- Zone2: new data



Adding 2 more disks

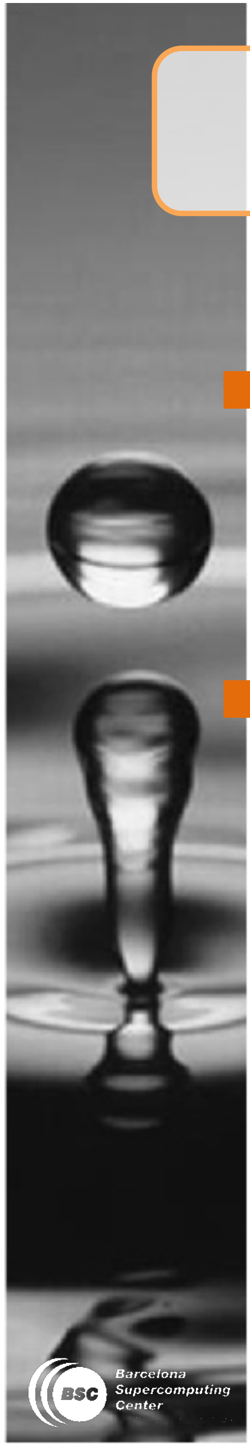
- Create two new zones
- Zone1: old data
  - Size = $\mathbf{1.15 * C_{old} * (1 - C_{old} / (C_{new} + C_{old}))}$
    - Depending on size
      - We create 2 zones
      - We merge zones
  - Restripe these blocks on all disks
- Zone2: new data



AdaptiveZ parameters

- **AdaptRaid parameters are used**
  - One set per zone
- **% increase on the size to restripe could be changed**
  - Our evaluation shows that 15% is a good tradeoff
  - and suggest no modifications

Storage System Research Group



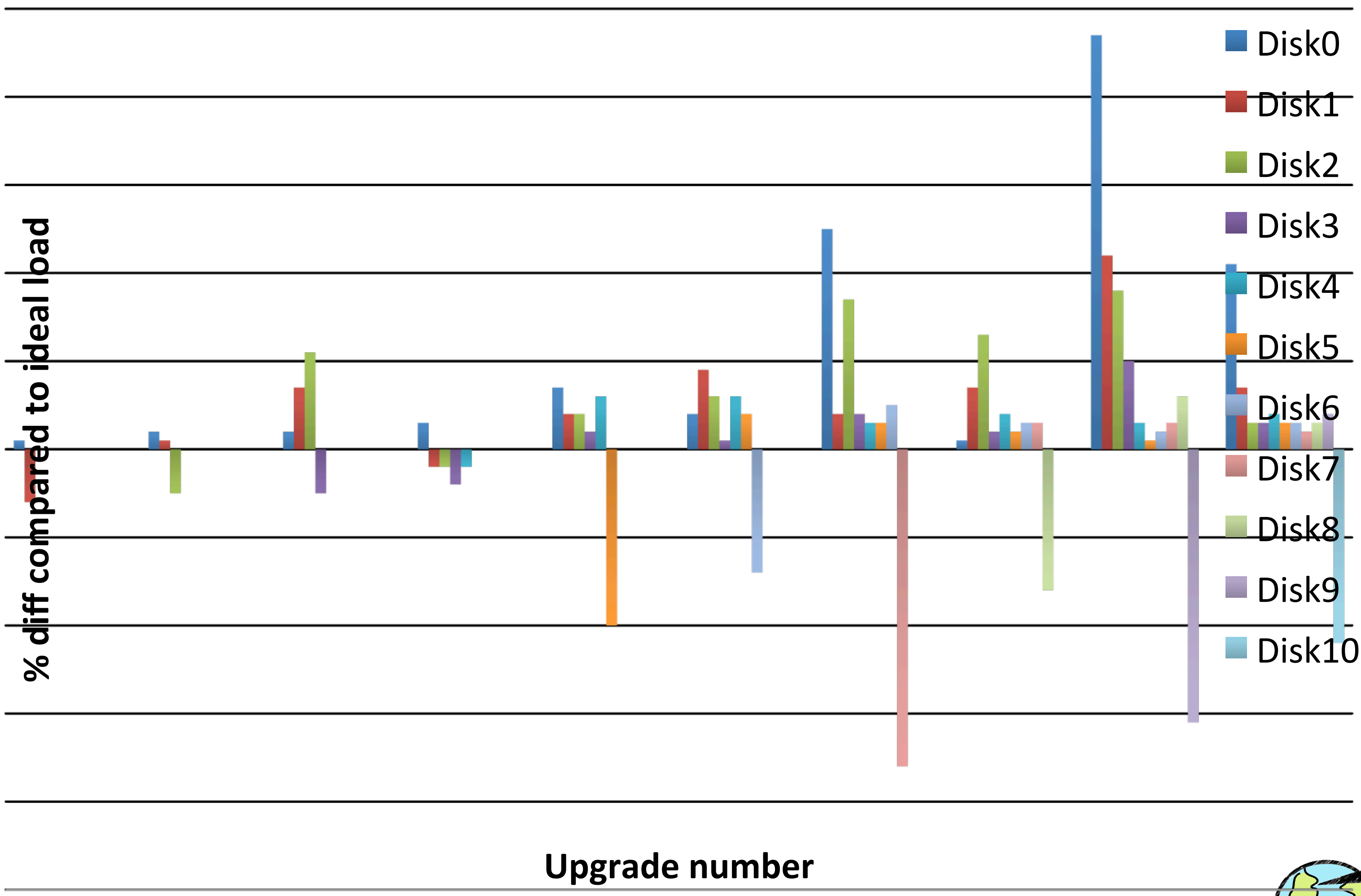
Computing the location of a block

- **Formulas**
  - *Zone = Search in zone tree*
  - Then use AdaptRaid mechanisms and metadata
- **Metadata**

```
Zone // Tree of AdaptRaid patterns
     // At most 2 new zones per upgrade
     // If upgrades every 6 months:
     //      40 zones in 10 years
     //      6 levels in the tree
```

Storage System Research Group

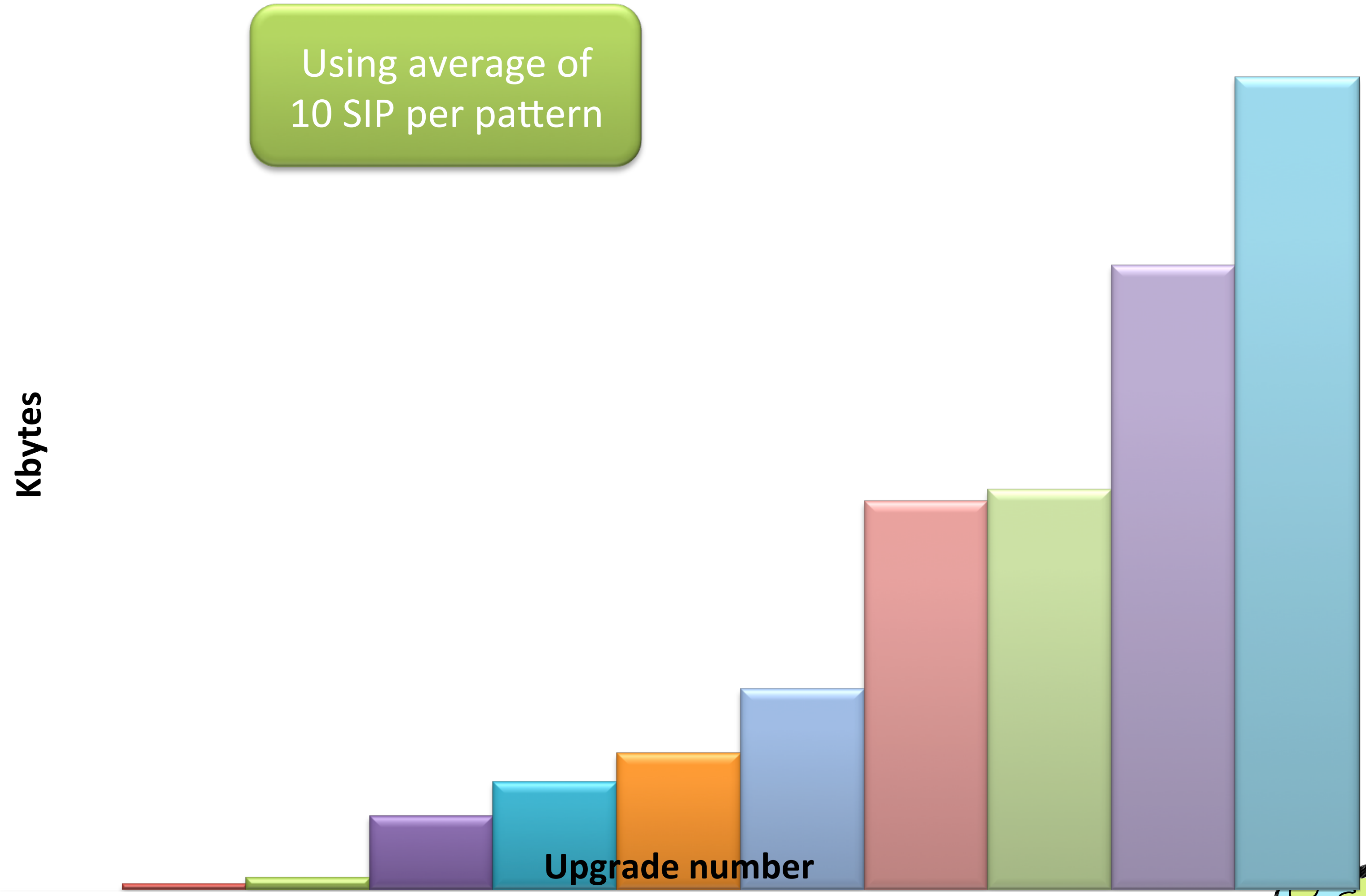
Load balance after 10 upgrades



Barcelona Supercomputing Center

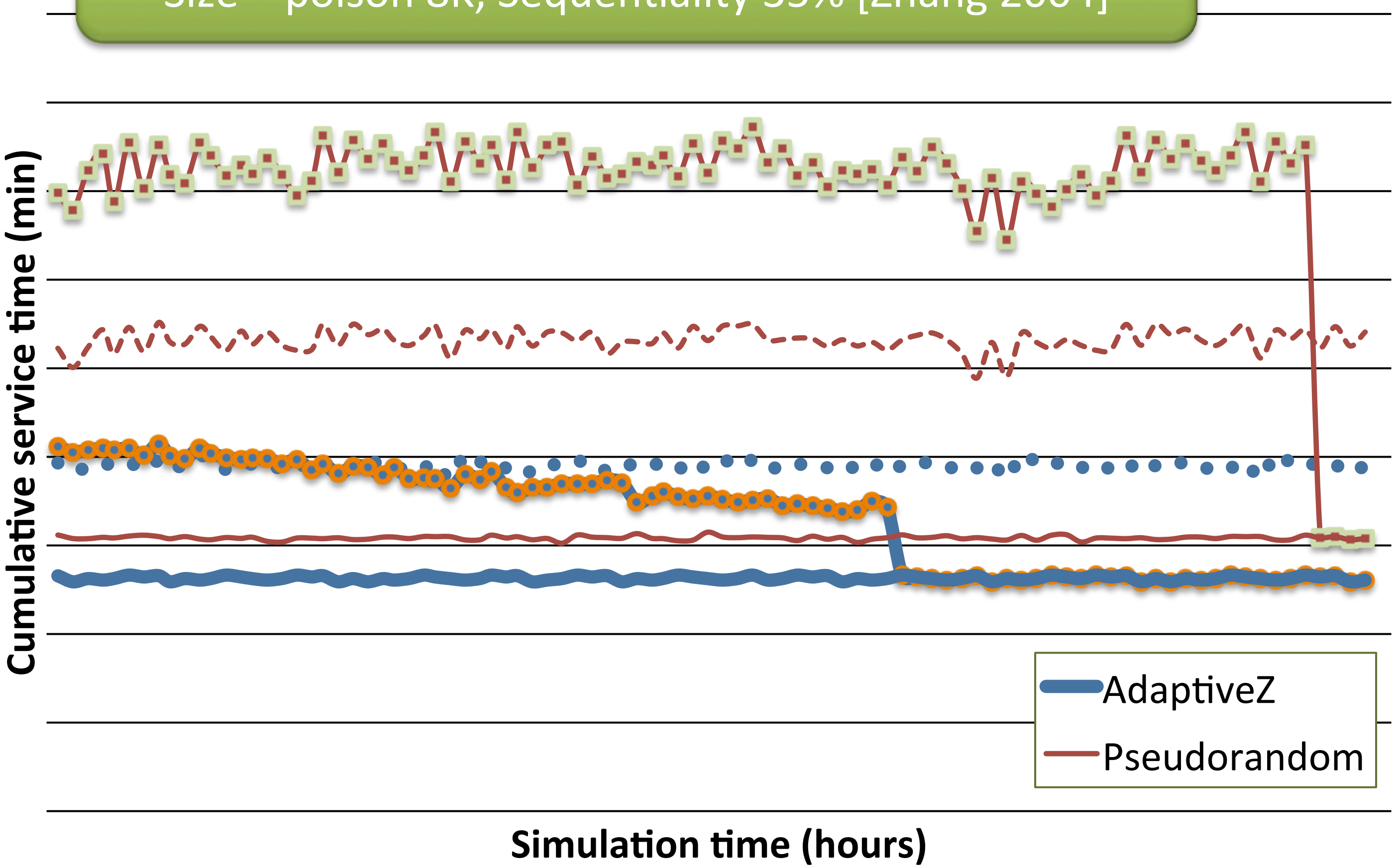
Storage System Research Group

Metadata size after 10 upgrades



Performance results

5 ms arrival time, 300 ms OFF periods, 70% reads
Size = poison 8K, Sequentiality 35% [Zhang 2004]



Conclusions

- **Heterogeneous storage systems can be handled**
  - We can take advantage of the heterogeneity
- **They can be scalable**

- **There is another way to solve the same problem**
  - Randomization