FastScale: Accelerate RAID Scaling by Minimizing Data Migration

Weimin Zheng, Guangyan Zhang

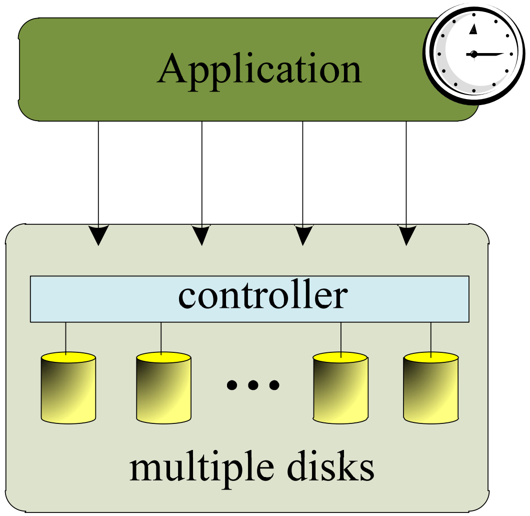
gyzh@tsinghua.edu.cn

Tsinghua University

Outline

- **Motivation**
- Minimizing data migration
- Optimizing data migration
- Evaluation
- Conclusions

Why Scale a RAID



- A disk is a simple computer
- **A RAID vol. can deliver high perf.**
  - **Multi disks serve an App concurrently.**
- **applications often require larger capacity and higher performance.**
  - **As user data increase and computing powers enhance**
- **One solution is to add new disks to a RAID volume**
  - **This disk addition is termed "*RAID scaling*".**
- **To regain a balanced load, some blocks needs to be moved to new disks.**
- **Data migration need to be performed online**
  - **To supply non-stop services.**

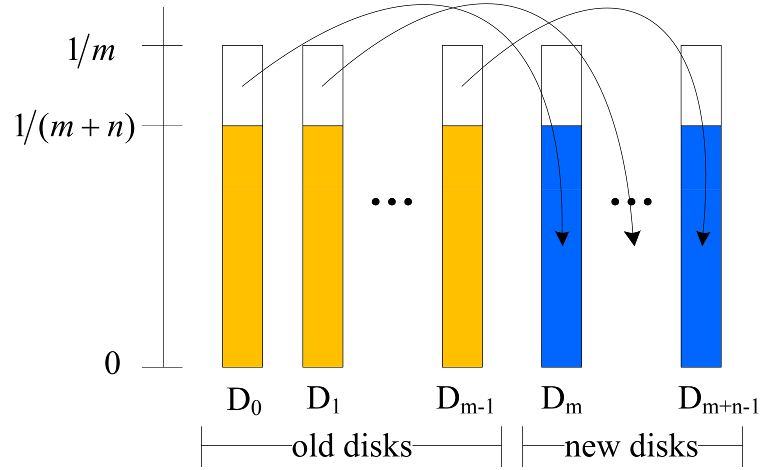
Limitation of Existing Approach

- **Existing approach to RAID scaling preserves the round-robin order after adding disks.**
  - **Pro: the addressing function is simple.**
  - **Con: all the data need to be moved**
- **Recent work has optimized data migration, among which one typical example is SLAS (*ACM TOS 2007*):**
  - **Uses I/O aggregation and lazy checkpointing to improve the efficiency**
  - **Due to migration of all the data, RAID scaling remains costly**

Can we reduce the total number of migrated data blocks?

Minimizing Data Migration

- FastScale moves only data blocks from old disks to new disks, while not migrating data among old disks.
  - It is enough for preserving the uniformity of data distribution
- In this manner, FastScale minimizes data migration for RAID scaling.



- We design an elastic addressing function, through which
  - the location of one block can be easily computed
  - without any lookup operation.

Optimizing Data Migration

- **FastScale also exploits physical properties to optimize online data migration.**
  - **First, it uses aggregate accesses to improve the efficiency of data migration.**
  - **Second, it records data migration lazily to minimize the number of metadata updates while ensuring data consistency.**

Results

- **Implemented FastScale and SLAS in DiskSim 4.0**
  - **Compared with SLAS, Round-robin RAID-0 scaling.**

- **Evaluation during RAID scaling:**
  - **reduce redistribution time by up to 86.06%**
  - **with smaller maximum response time of user I/Os**

- **Evaluation after 1 or 2 RAID scaling operations:**
  - **is almost identical with the round-robin RAID-0.**

Coverage of FastScale

- **In this paper, we only describe our solution for RAID-0, i.e., striping without parity.**
  - **FastScale can also work for RAID-10 and RAID-01.**
  - **Some large storage systems slice disks into many segments, several segments are organized into a RAID.**

- **Although we do not handle RAID-4 and RAID-5, we believe that our method provides a good starting point for efficient scaling of RAID-4 and RAID-5 arrays.**

Outline

- Motivation
- **Minimizing data migration**
- Optimizing data migration
- Evaluation
- Conclusions

Requirements for RAID Scaling

- **Requirement 1 (Uniform data distribution):**
  - **If there are B blocks stored on m disks, the expected number of blocks on each disk is approximately B/m so as to maintain an even load.**
- **Requirement 2 (Minimal Data Migration):**
  - **During the addition of n disks to a RAID with m disks storing B blocks, the expected number of blocks to be moved is B*n/(m+n).**
- **Requirement 3 (Fast data Addressing):**
  - **In a m-disk RAID, the location of a block is computed by an algorithm with low space and time complexity.**

Semi-RR: the Most Intuitive Method

- semi-RR is based on Round-robin scaling
  - Only if the resulting disk is one of new disks, it moves a data block.
  - Otherwise, it does not move a data block.
- Good news: Semi-RR can reduce data migration significantly.
- Bad news: it does not guarantee uniform distribution of data blocks after multiple scaling operations

FastScale: Min Migr. & Uniform Dist.

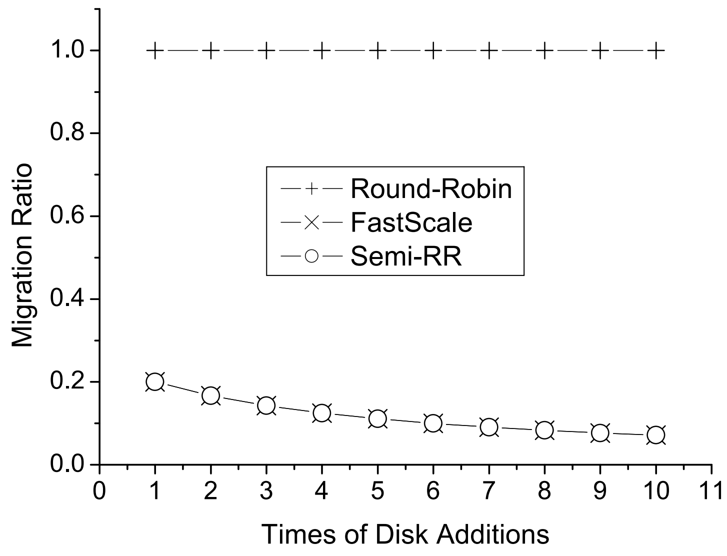
- take RAID scaling from 3 disks to 5 as an example.
- one RAID scaling process can be divided into two stages logically:
  - data migration and,
  - data filling.
- all the data blocks within a parallelogram will be moved.
  - 2 data blocks are migrated from each old disk.
  - while its physical block number is unchanged.
- An elastic function to describe the data layout

FastScale: Property Examination

- **Does FastScale satisfies the three requirements?**
  - **compared with the round-robin and semi-RR algorithms.**
- **From a 4-disk array, we add one disk repeatedly for 10 times, using the three algorithms respectively.**
- **Each disk has a capacity of 128 GB, and the block size is 64 KB.**
  - **In other words, each disk holds 2M blocks.**

Comparison in Migration Fraction

- Using the round-robin algorithm,
  - the migration fraction is constantly 100%
- using semi-RR and FastScale
  - The migration fractions are identical.
  - They are significantly smaller
  - Restricted by uniformity, they are also minimal.

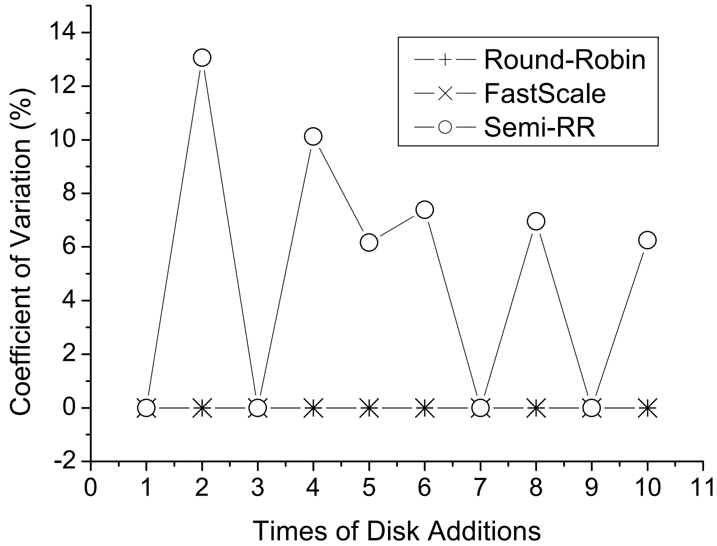


Compared in migration fraction, Semi-RR and FastScale win!

Comp. in Uniformity of Distribution

- **We use the coefficient of variation as a metric to evaluate the uniformity of data distribution across all the disks.**
  - **The C.V. expresses the std dev. as a percentage of the average.**

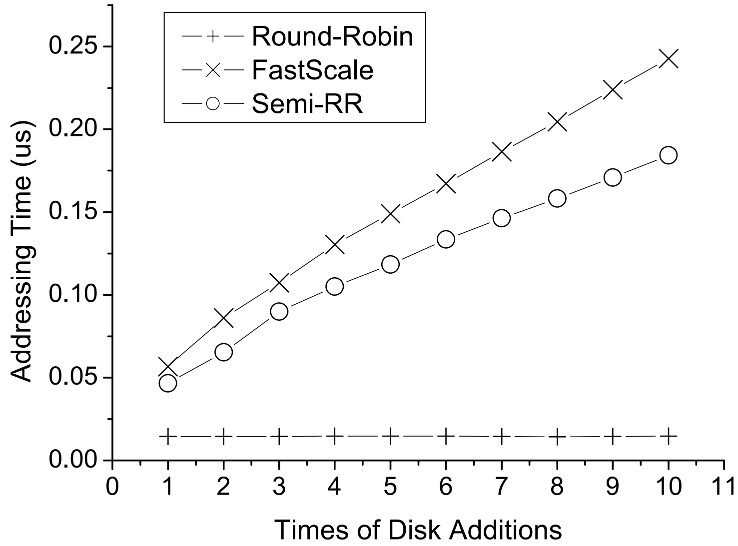
- **For the round-robin and FastScale algorithms,**
  - **C.V. remain 0 percent as the addition times increases.**
- **the semi-RR algorithm**
  - **causes excessive oscillation in the C.V.**
  - **Maximum is even 13.06%.**



Compared in uniformity of distribution, Semi-RR fails and FastScale wins again!

Comparison in Calculation Overhead

- we run different algorithms to calculate the physical addresses for all data blocks on a scaled RAID.
  - the average addressing time for each block is calculated.
  - Setup: Intel Dual Core T9400 2.53 GHz, 4 GB Memory, Windows 7
- The Round-robin algorithm has the lowest overhead,
  - 0.014 μs or so.
- FastScale has the largest overhead.
  - the largest time is 0.24 μs



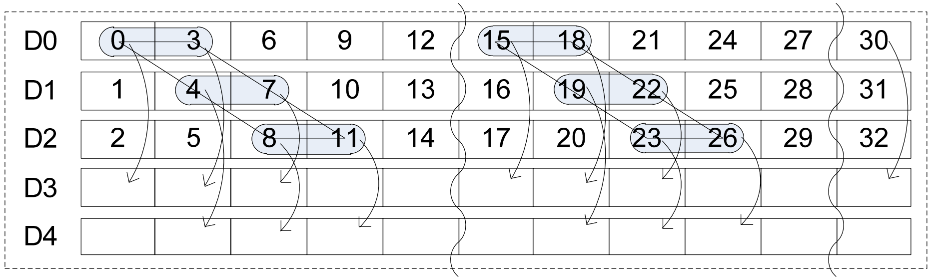
compared to milliseconds of disk I/O time, the calculation overhead is negligible.

Outline

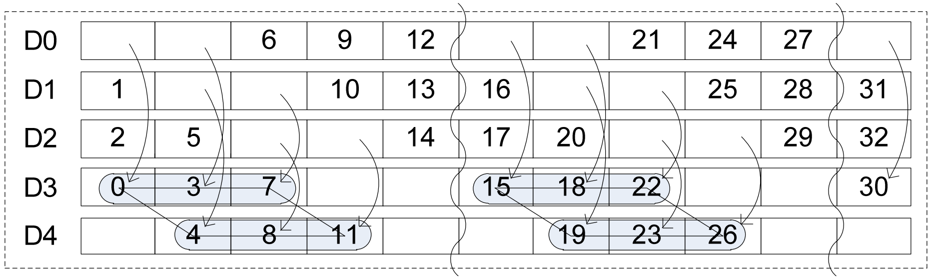
- Motivation
- Minimizing data migration
- **Optimizing data migration**
- Evaluation
- Conclusions

I/O Aggregation

- **Aggregate read:**
  - **Multiple successive blocks on a disk are read via a single I/O.**



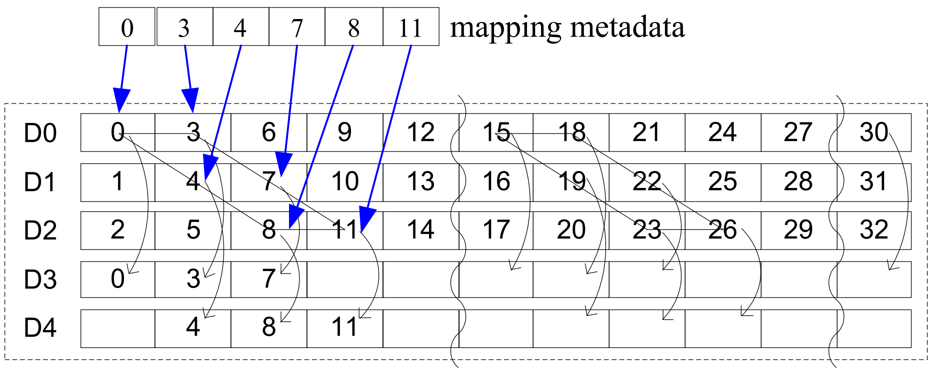
- **Aggregate write:**
  - **Multiple successive blocks on a disk are written via a single I/O.**



converts small requests into fewer, larger requests.
seek cost is mitigated over multiple blocks.

Why can Lazy Checkpointing work?

- **Each metadata update causes one long seek :**
  - **MetaData is usually stored at the beginning of member disks**
- **after data copying, new replica and original are valid.**
  - **block copying does not overwrite any valid data**
- **when the system fails and reboots, the original replica will be used.**

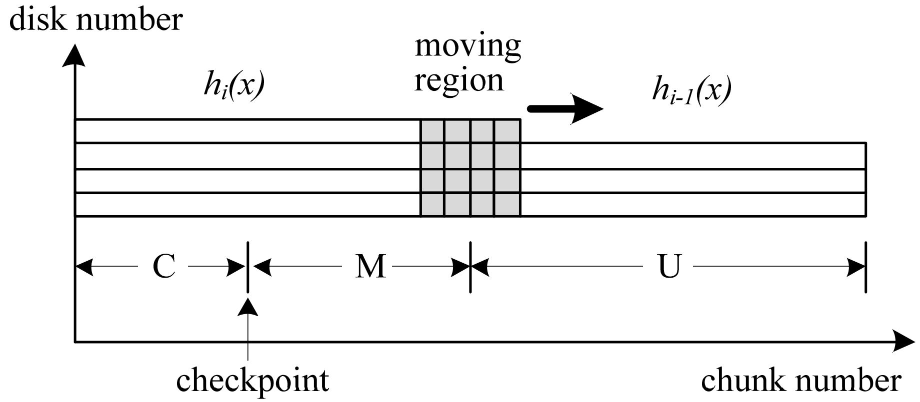


- **As long as data has not been written since being copied, the data remain consistent.**
  - **Only some I/Os are wasted**

not updating MD immediately does not sacrifice data reliability. The only threat is write to migrated data.

Lazy Checkpointing

- **data blocks are copied to new locations continuously**
  - **while the mapping metadata is not updated onto the disks until a threat to data consistency appears.**
- **In the figure,**
  - **"C": migrated and checkpointed**
  - **"M": migrated but not checkpointed;**
  - **"U":not migrated**



- **only when a user write request arrives in the area "M", data migration is checkpointed.**

lazy checkpointing minimizes the number of metadata writes without loss of data consistency.

Outline

- Motivation
- Minimizing data migration
- Optimizing data migration
- **Evaluation**
- Conclusions

Evaluation

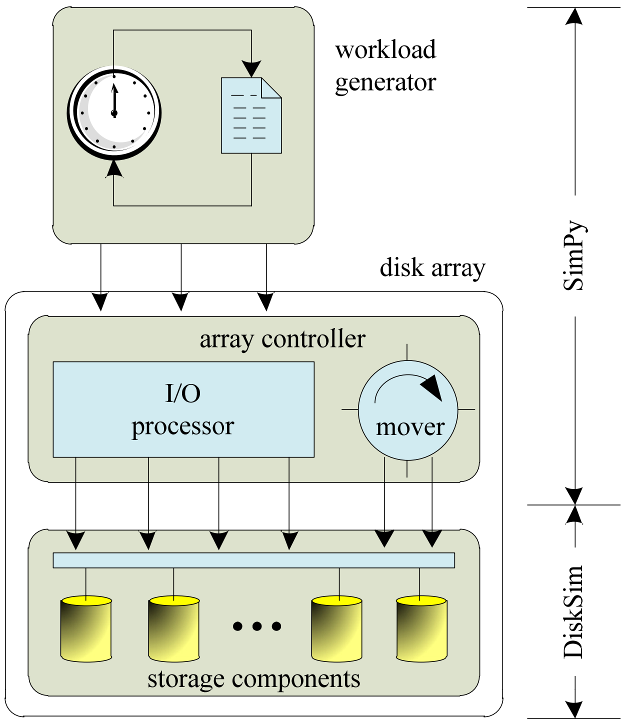
- **Questions that we want to answer:**
  - **Can FastScale accelerate RAID scaling?**
  - **What is the effect on user workloads?**
  - **How about the performance of a scaled RAID?**

- **We used detailed simulations to compare with SLAS**
  - **The simulator is implemented with DiskSim as a worker module**
- **with several disk traces collected in real systems**
  - **The traces are TPC-C, Financial trace from SPC, Web search engine trace from SPC**

Evaluation

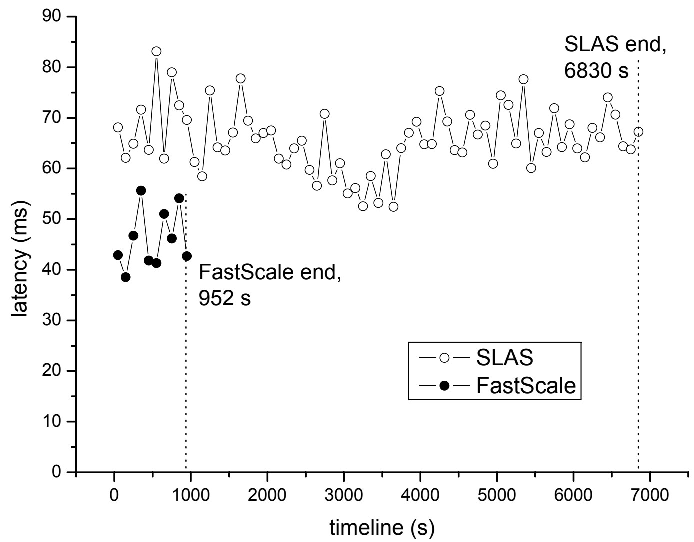
- The simulator is made up of a workload generator and a disk array.
  - workload generator initiates an I/O request at the appropriate time.

- The disk array consists of
  - an array controller and,
  - Storage components.
- The array controller is logically divided into:
  - an I/O processor and,
  - a data mover.
- The simulator is implemented in SimPy and DiskSim.



Scaling under the Financial Workload

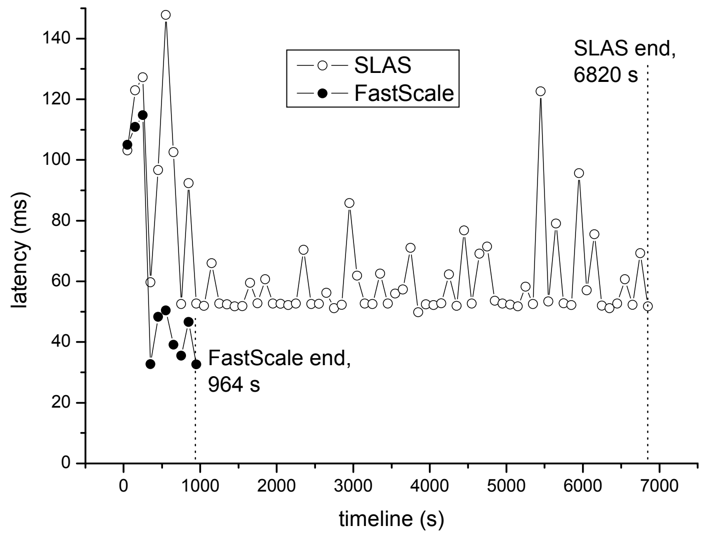
- Under the Fin workload, we conduct a scaling op:
  - adding 2 disks to a 4-disk RAID,
  - each disk has a capacity of 4 GB,
  - with the 32KB stripe unit size
- The figure plots local max latencies as the time increases



- FastScale accelerates RAID scaling significantly
  - 952s vs 6,830s, 86.06% improved
- local max latencies are also smaller

Scaling under the TPC-C Workload

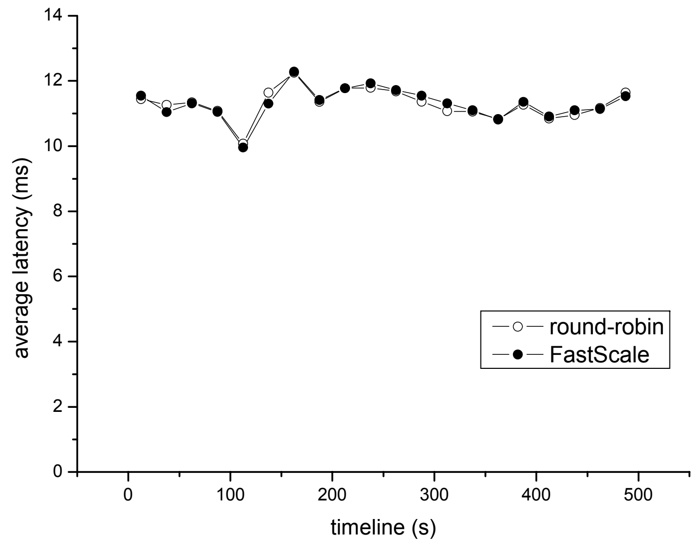
- **Under the TPC-C workload, we redo the scaling:**
  - **adding 2 disks to a 4-disk RAID,**
- **The figure plots local max latencies as the time increases**
- **Once again, shows the efficiency in improving redistribution time**
  - **964s vs 6,820s, 85.87% improved**
- **local max latencies are also smaller**



Fastscale improves the scaling efficiency of RAID significantly.

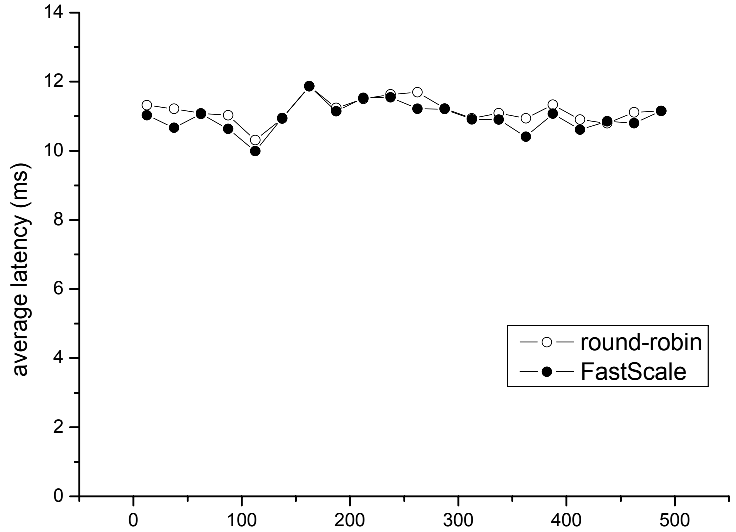
After One Scaling Operation

- **We compared the performance of two RAIDs scaled using FastScale and SLAS:**
  - **"4+1": adding 1 disk to a 4-disk RAID**
- **We replayed the Web workload on two RAIDs.**
- **The figure plots local avg latencies as the time increases**
- **the performances of the two RAIDs are very close.**
  - **For the round-robin RAID, the average latency is 11.36 ms.**
  - **For the FastScale RAID, the average latency is 11.37 ms.**



After Two Scaling Operations

- We compared the performance of two RAIDs scaled twice using FastScale and SLAS:
  - "4+1+1": adding 1 disk to a 4-disk RAID twice
- The figure plots local avg latencies as the time increases
- It again reveals the approximate equality in the performances.
  - For the round-robin RAID, the average latency is 11.21 ms.
  - For the FastScale RAID, the average latency is 11.03 ms.



the performance of the FastScale RAID-0 is almost identical with that of the RR RAID-0

Outline

- Motivation
- Minimizing data migration
- Optimizing data migration
- Evaluation
- **Conclusions**

Conclusions

- **FastScale accelerates RAID-0 scaling significantly**
  - **minimizes data migration without loss of the uniformity of data distribution**
  - **optimizes data migration with I/O aggregation and lazy checkpointing**
- **Compared with a round-robin scaling approach, FastScale can:**
  - **reduce redistribution time by up to 86.06%**
  - **with smaller maximum response time of user I/Os.**
- **the performance of the RAID scaled using FastScale is almost identical with that of the round-robin RAID.**

Thank you!

Questions?

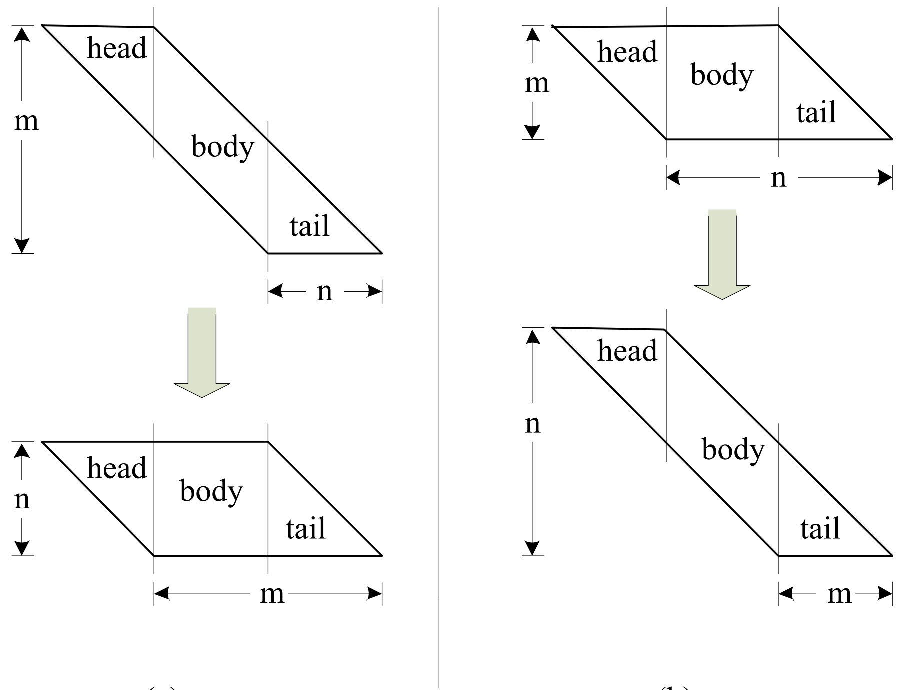
Guangyan Zhang

http://storage.cs.tsinghua.edu.cn/~zgy

How is a Block Moved?

- a parallelogram is divided into three parts:
  - a head triangle, unchanged shape
  - a body parallelogram,
  - a tail triangle, unchanged shape

- The body parallelogram:
  - If m>=n, not a rectangle, change it into a rectangle
  - Otherwise, change the rectangle into a parallelogram.



(a) m>=n (b) m<n

Comparison in Local avg Latencies

- Under the Fin workload, we conduct a scaling op:
  - adding 2 disks to a 4-disk RAID,
  - each disk has a capacity of 4 GB,
  - with the 32KB stripe unit size
- The figure plots local avg latencies as the time increases
- local avg latencies are close
  - **FastScale 8.01 ms,**
  - **SLAS 7.53 ms**
- shorter data redistribution time