MSST 2012 April 19th, Pacific Grove, CA

HRAID6ML: A Hybrid RAID6 Storage Architecture with Mirrored Logging

Lingfang Zeng, Dan Feng, Jianxi Chen and Wenguo Liu
Huazhong University of Science and Technology

Qingsong Wei
Data Storage Institute, A*STAR, Singapore

Bharadwaj Veeravalli
National University of Singapore

Outline

1. Background, Motivation, and Related work

2. Architecture

3. Design and Implementation

4. Performance Evaluations

1. Background and Motivation (1/3)

- □ HDD-based RAID suffers high latency of random accesses due to slow mechanical positioning nature of Hard Disk Drives (HDDs).
- □ NAND flash based Solid State Drives (SSDs) provide much higher random read performance and lower power consumption than HDD.
- □ The steady bit cost reduction of NAND flash memory now makes it economically viable to implement SSD using NAND flash memory [Yoo2011].
- □ RAID of SSDs is more cost-efficient than PCIe SSD in terms of capacity per dollar and bandwidth per dollar [Kim2011].

1. Background and Motivation (2/3)

□ Characteristic feature of SSD known as “erase-before-write” [Narayanan2009, Greenan2009, Kadav2009].

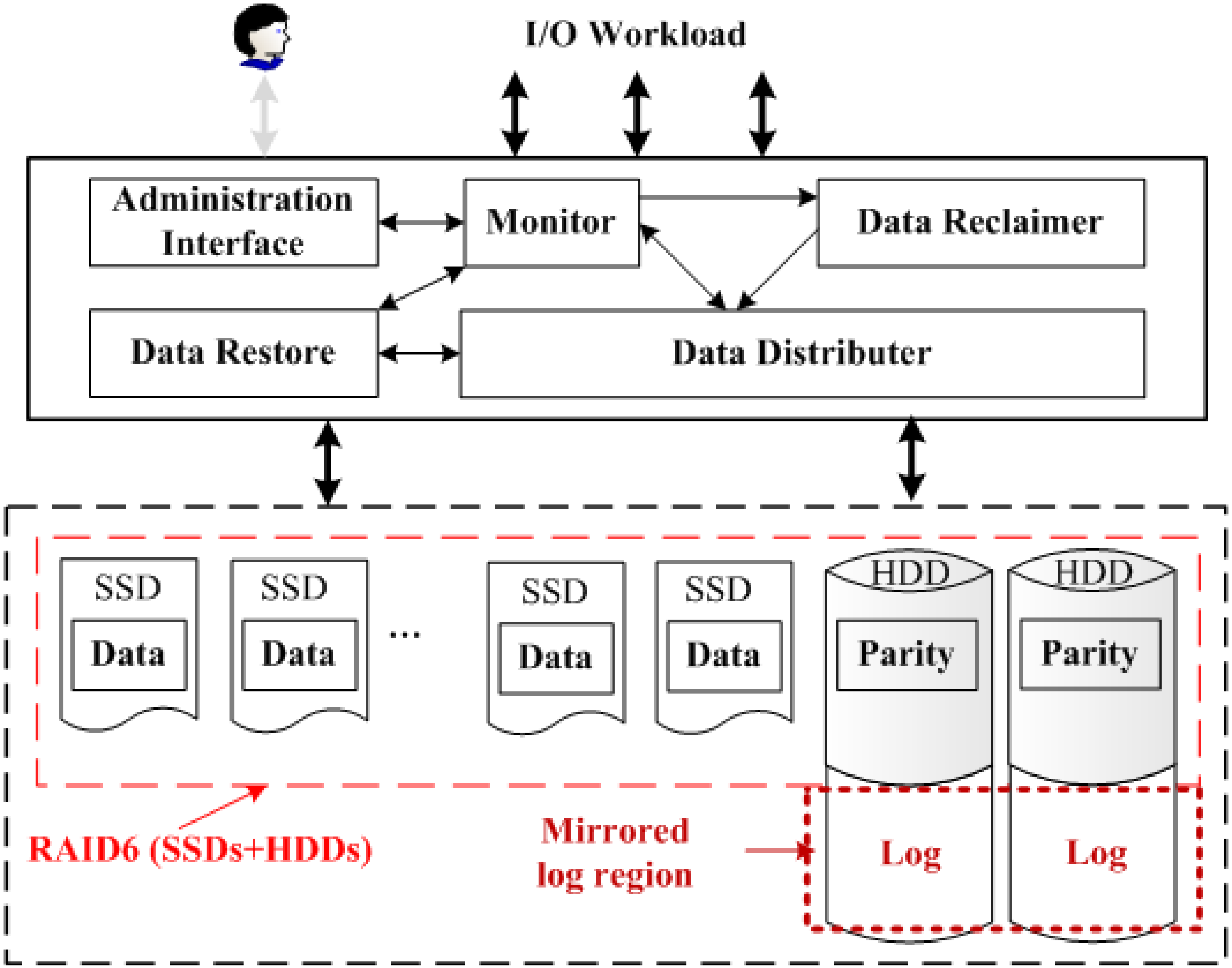
□ The RAID6 architecture is playing an increasingly important role in modern storage systems due to allowing the loss of any two drives. However, its high write penalty, because of the double-parity-update overheads upon each write operation.

□ RAID6L [Jin2011] integrates a log disk into the traditional RAID6 architecture, and alleviates its write penalty by simplifying the processing steps to service a write request. Different from RAID6L, HRAID6ML is not required a dedicated disk used as log region, moreover, HRAID6ML provides a mirrored log region to avoid log-data loss.

1. Background and Motivation (3/3)

□ **The difference between HPDA [Mao2010] and HRAID6ML is that HPDA is based on RAID4 architecture and required a dedicated log disk (part of the log disk space is wasted).**

2. Architecture

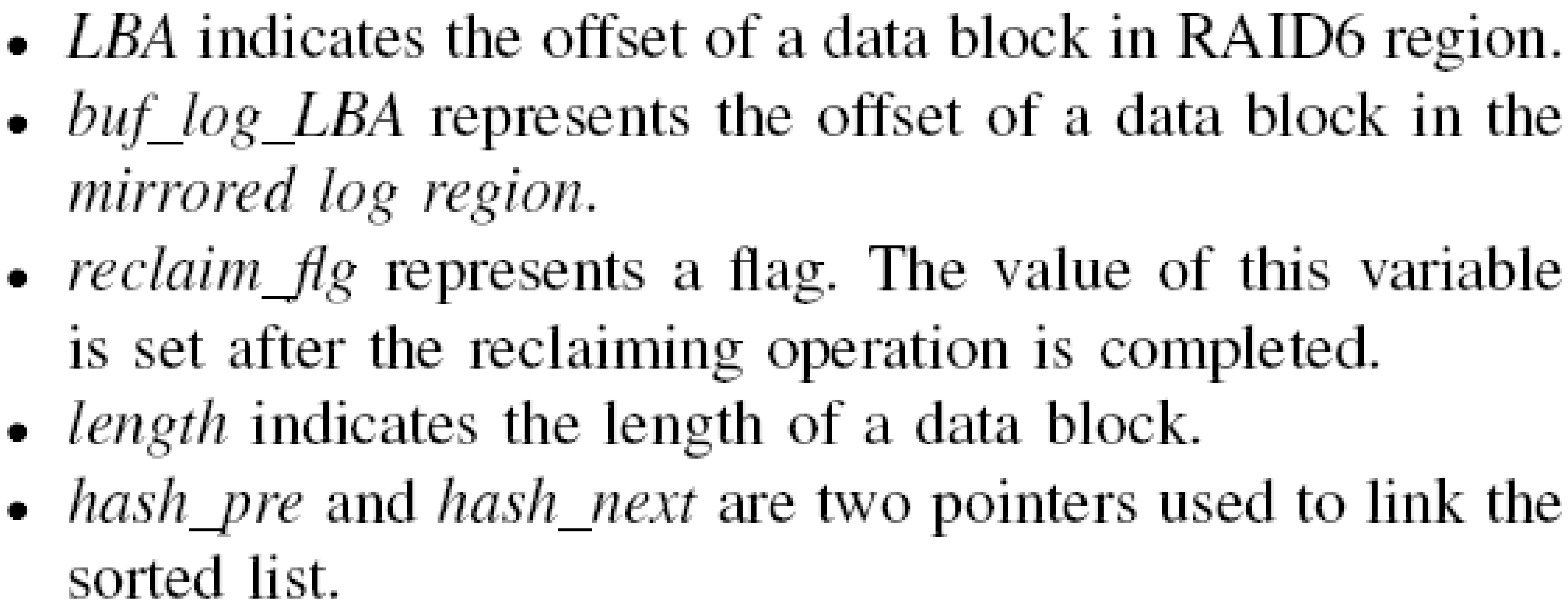


3. Design and Implementation (1/4)

- □ Implementation issues
  - We have implemented an HRAID6ML prototype in the Linux software RAID framework as an independent module.
  - We mainly modify the "handle_stripe6" function in original RAID6 module and add the hash list structure.
- □ Metadata refresh and consistency check
  - We update the HRAID6ML metadata (including the "blk_log_list") using asynchronous method: the strategy is to periodically refresh or to refresh when the system is idle.
  - We use a checksum algorithm to guarantee a very low failure rate for aforementioned HRAID6ML metadata.

3. Design and Implementation (2/4)

□ **The main variables in the entry are explained as follows:**



- *LBA* indicates the offset of a data block in RAID6 region.
- *buf_log_LBA* represents the offset of a data block in the *mirrored log region.*
- *reclaim_flg* represents a flag. The value of this variable is set after the reclaiming operation is completed.
- *length* indicates the length of a data block.
- *hash_pre* and *hash_next* are two pointers used to link the sorted list.

3. Design and Implementation (3/4)

□ Process flow of write/read request

- Write -- the Monitor first determines whether the request is sequential with its prior requests.
- Read -- first checks whether there is an entry corresponding to the request in the block-log list.
- An additional operation in HRAID6ML is the reclaiming operation.

3. Design and Implementation (4/4)

□ Recovery

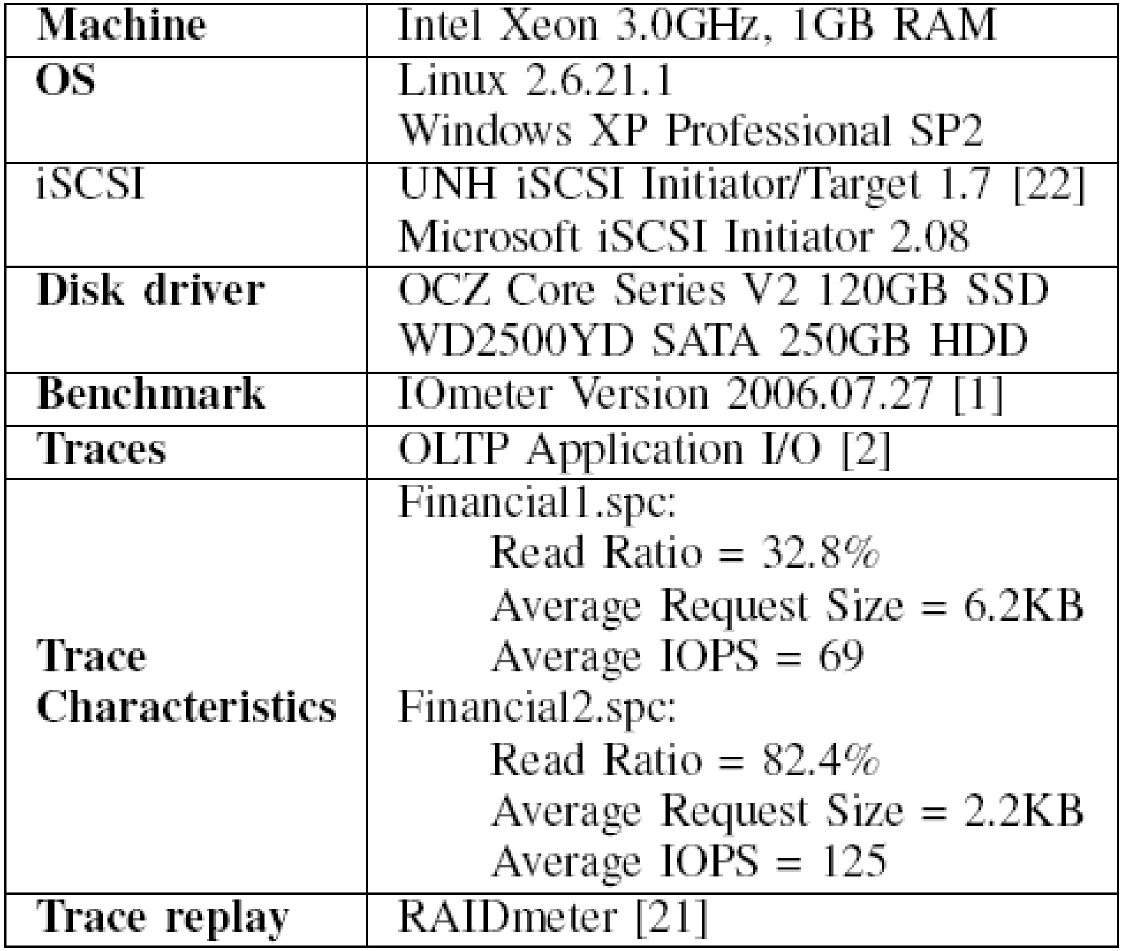
- If one parity disk fail, the Data Reclaimer is triggered to reclaim the write data from the mirrored logging (in the normal log region) to the RAID6 region according to the block-log list.
- If a SSD (data disk) and a parity disk (HDD) fail, each parity stripe loses one data block and one parity block.
- If two SSDs (data disks) fail, each parity stripe in the RAID6 region loses two data blocks.

□ Scalability

- Alleviated performance bottleneck
- Elimination of single point of failure

4. Performance Evaluations (1/3)

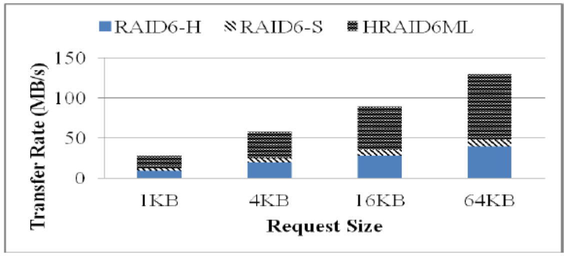
- Experimental setup and methodology



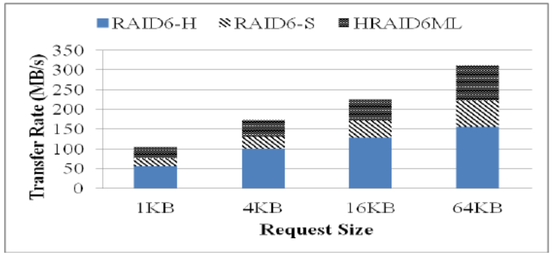
Machine	Intel Xeon 3.0GHz, 1GB RAM
OS	Linux 2.6.21.1 Windows XP Professional SP2
iSCSI	UNH iSCSI Initiator/Target 1.7 [22] Microsoft iSCSI Initiator 2.08
Disk driver	OCZ Core Series V2 120GB SSD WD2500YD SATA 250GB HDD
Benchmark	IOmeter Version 2006.07.27 [1]
Traces	OLTP Application I/O [2]
Trace Characteristics	Financial1.spc: Read Ratio = 32.8% Average Request Size = 6.2KB Average IOPS = 69 Financial2.spc: Read Ratio = 82.4% Average Request Size = 2.2KB Average IOPS = 125
Trace replay	RAIDmeter [21]

4. Performance Evaluations (2/3)

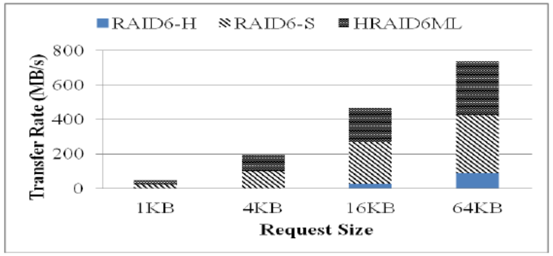
- **Throughput (Data transfer rate)**
  - **Random write requests: better than RAID6-H and RAID6-S 107.43% and 32.03% on average.**
  - **Requential write requests, HRAID6ML outperforms RAID6-S by 656.25% on average, but is inferior to RAID6-H by 89.85% on average.**



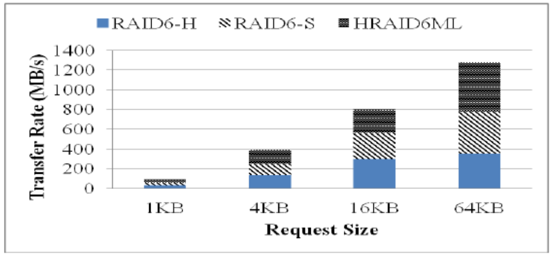
(a) Random write requests



(b) Sequential write requests



(c) Random read requests

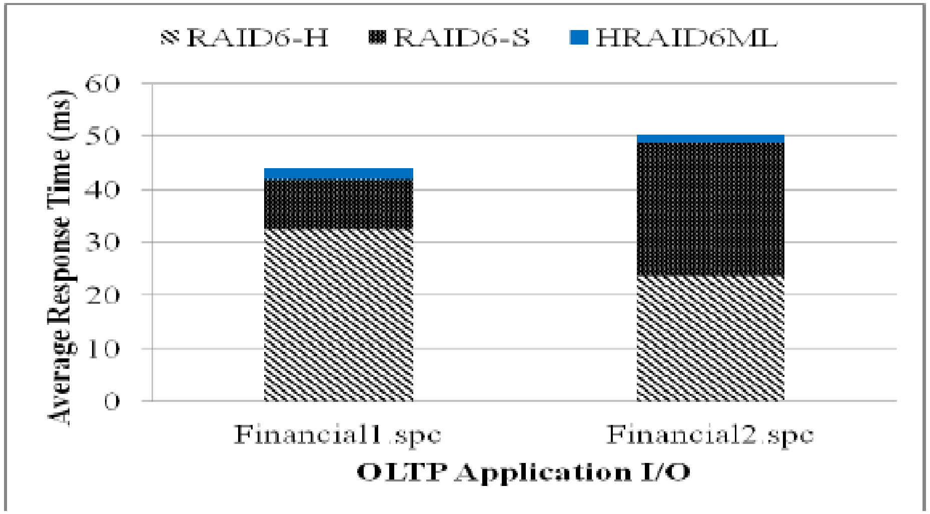


(d) Sequential read requests

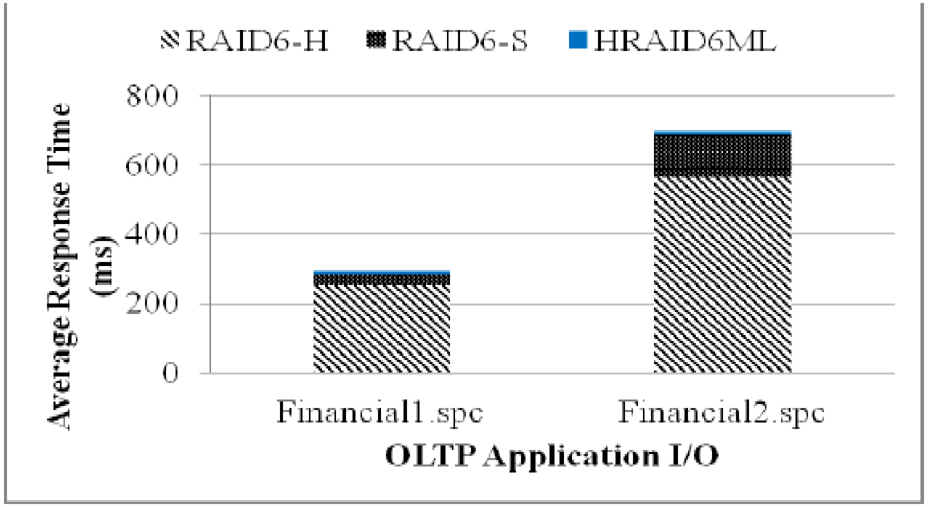
4. Performance Evaluations (3/3)

- □ **Average response time**

  •In terms of average response time, HRAID6ML outperforms RAID6-H by a factor of up to 15.29 and 14.84 respectively under the two traces, and outperforms RAID6-S by a factor of up to 4.38 and 15.73 respectively under the two traces.



(a) Normal mode



(b) Degraded mode

Thanks!