Atlas: Baidu's Key-value Storage System for Cloud Data

Song Jiang

Wayne State University

Chunbo Lai

Shiding Lin

Zhenyu Hou

Can Cui

Baidu Inc.

Liqiong Yang

Guangyu Sun

Peking University

Jason Cong

University of California Los Angeles

Cloud Storage Service

- Cloud storage services become increasingly popular.
  - Baidu Cloud has over 200 million users and 200PB user data.
- To be attractive and competitive, they often offer large free space and price the service modestly.
  - Baidu offers 2TB free space for each user.
- The challenge is how to economically provision resources and also achieve service quality.
  - A large number of servers, each with local large storage space.
  - The data must be reliably stored with a high availability.
  - Requests for any data in the system should be served reasonably fast.

Challenges on Baidu's System

- The workload
  - Request size is capped at 256KB for system efficiency.
  - Majority of the requests are for data between 128KB and 256KB.

Distribution of requests on a typical day in 2014.

Request Size (Bytes)	Read (%)	Write (%)	#Read / #Write
[0, 4K]	0.6%	1.2%	1.45
(4K, 16K]	0.5%	1.0%	1.41
(16K, 32K]	0.5%	0.8%	1.67
(32K, 64K]	0.8%	1.2%	1.94
(64K, 128K]	1.3%	1.7%	2.08
(128K, 256K]	96.3%	94.1%	2.84
Sum	100.0%	100.0%	2.78

- The Challenges
  - Can the X86 processors be efficiently used?
  - Can we use a file system to store data at each server?
  - Can we use an LSM-tree-based key-value store to store the data?

Challenge on Processor Efficiency

- The X86 processors (two 4-core 2.4GHz E5620) were consistently under-utilized
  - Less than 20% utilization rate with nine hard disks installed on a server.
  - Adding more disks is not an ultimate solution.

- The ARM processor (one 4-core 1.6GHz Cortex A9) can provide similar I/O performance.
  - The ARM processor is more than 10X cheaper and more energy-efficient.

- Baidu's customized ARM-based server.
  - Each 2U chassis has six 4-core Cortex A9 processors.
  - Each processor comes with four 3TB SATA disks.
- However, each processor can support only 4GB memory.
  - On each chassis only 24GB memory available for accessing data as large as 72TB data.

Challenge on Using a File System

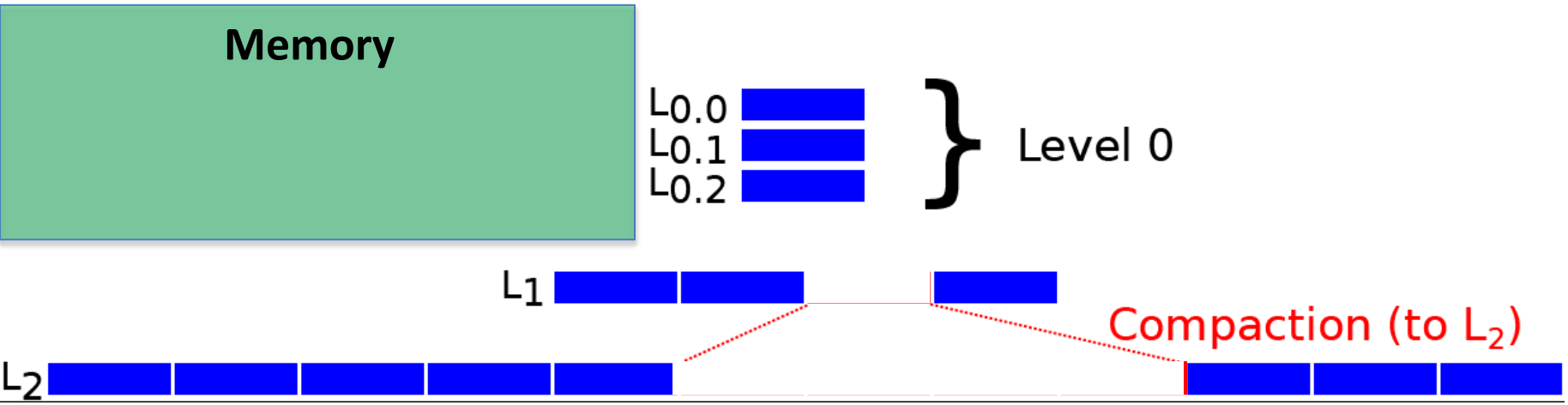
- Memory cannot hold all metadata.
  - Most files would be of 128-256KB.
  - Access on the storage has little locality.
  - More than one disk accesses are often required to access a file.
- The approach used in Facebook's Haystack is not sufficient.
  - There are 3.3GB metadata for 16TB 128KB-data.
  - System software and buffer cache also compete for 4GB memory.

Challenge on Using LSM-tree Based Key-value Store

- LSM-tree-based KV store is designed for storing many small key-value items, represented by Google's LevelDB.

- The store is memory efficient.
  - The metadata is only about 320MB for 16TB 128KB-data.

- However, the store needs constant compaction operations to sort its data distributed across levels of the store.
  - For a store of 7 levels, the write amplification can be over 70.
  - Very limited I/O bandwidth is left for servicing frond-end user requests.

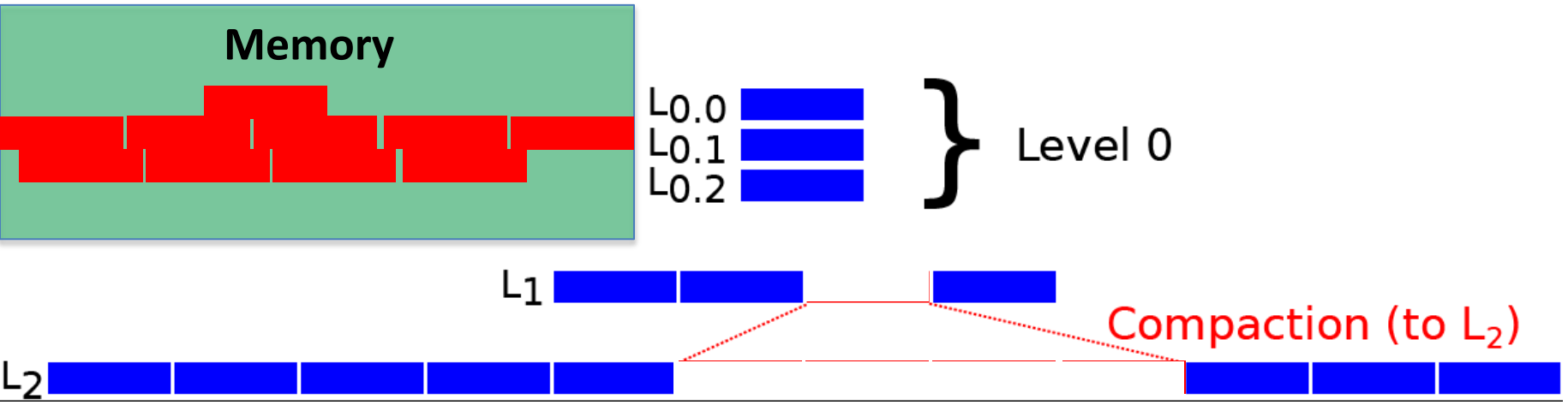


Challenge on Using LSM-tree Based Key-value Store

- LSM-tree-based KV store is designed for storing many small key-value items, represented by Google's LevelDB.

- The store is memory efficient.
  - The metadata is only about 320MB for 16TB 128KB-data.

- However, the store needs constant compaction operations to sort its data distributed across levels for such a small metadata.
  - For a store of 7 levels, the write amplification can be over 70.
  - Very limited I/O bandwidth is left for servicing frond-end user requests.

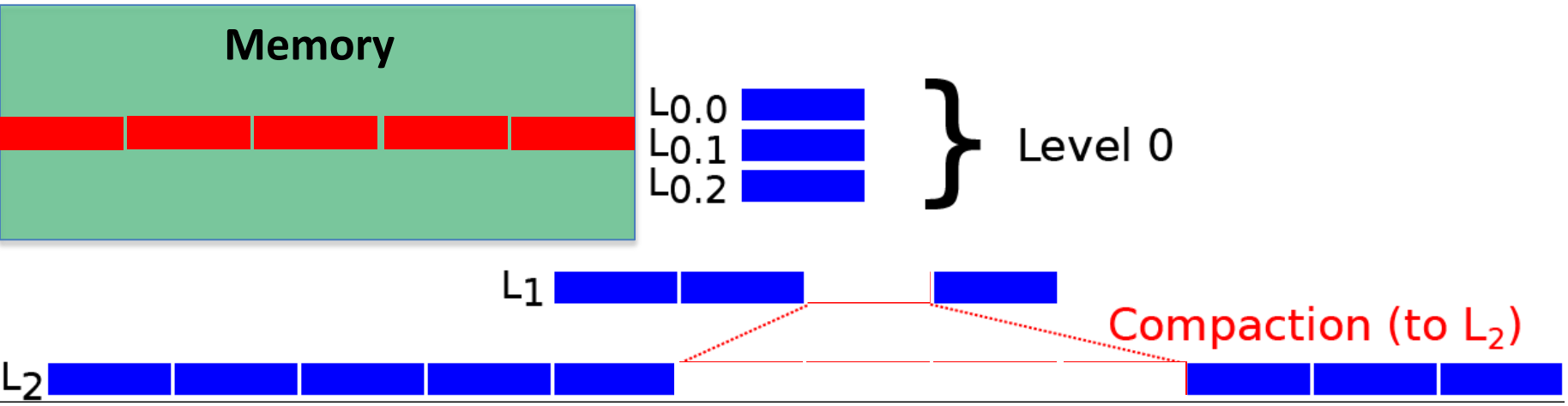


Challenge on Using LSM-tree Based Key-value Store

- LSM-tree-based KV store is designed for storing many small key-value items, represented by Google's LevelDB.

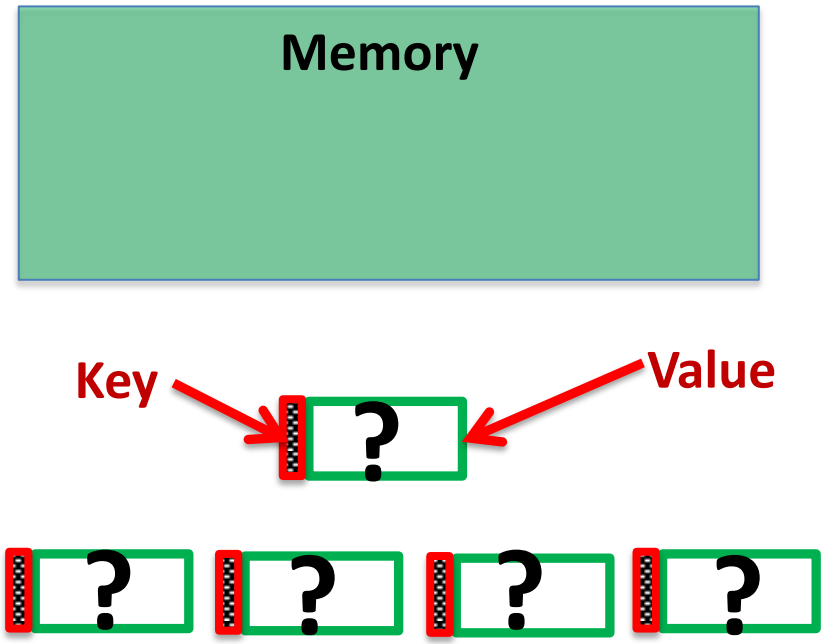
- The store is memory efficient.
  - The metadata is only about 320MB for 16TB 128KB-data.

- However, the store needs constant compaction operations to sort its data distributed across levels for such a small metadata.
  - For a store of 7 levels, the write amplification can be over 70.
  - Very limited I/O bandwidth is left for servicing frond-end user requests.



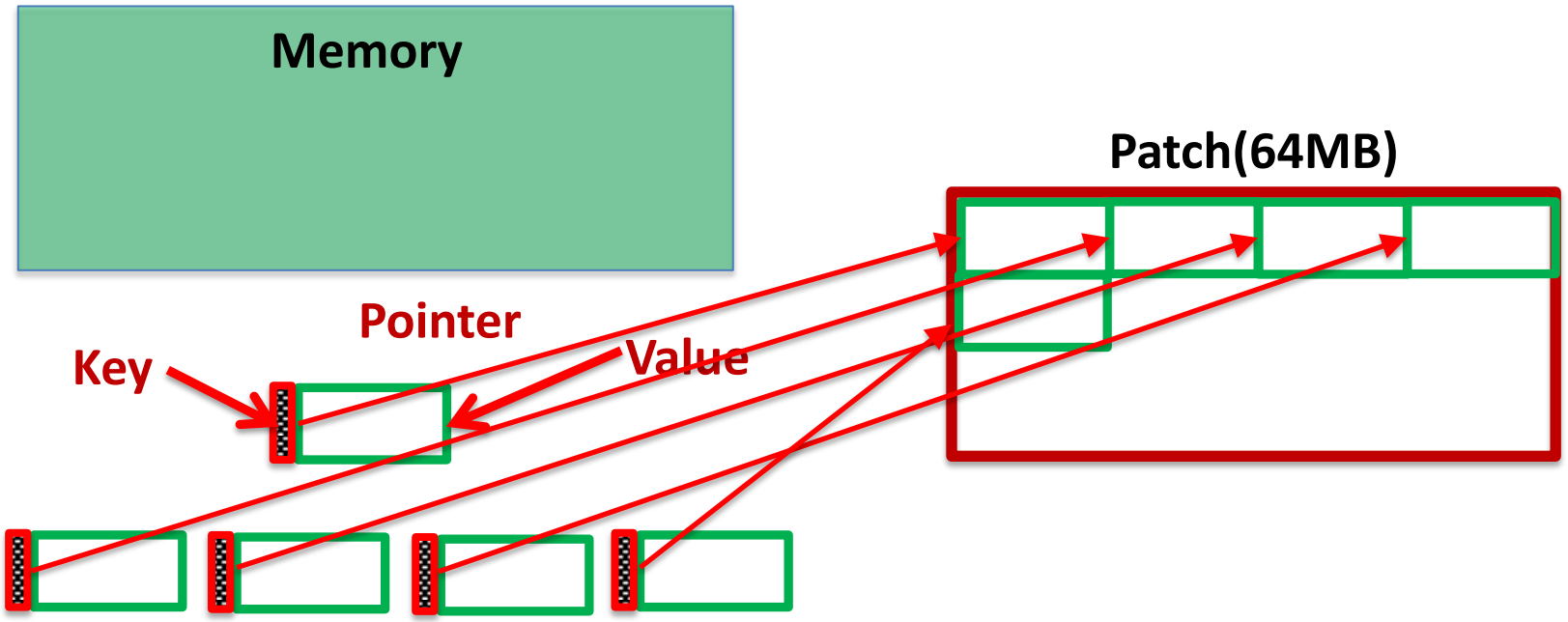
Reducing Compaction Cost

- In a KV item, value is usually much larger than the key.
- Values are not necessary to be involved in compactions.
- Move and place the values in a fixed-size container (block), and replace the values with pointers in KV items.



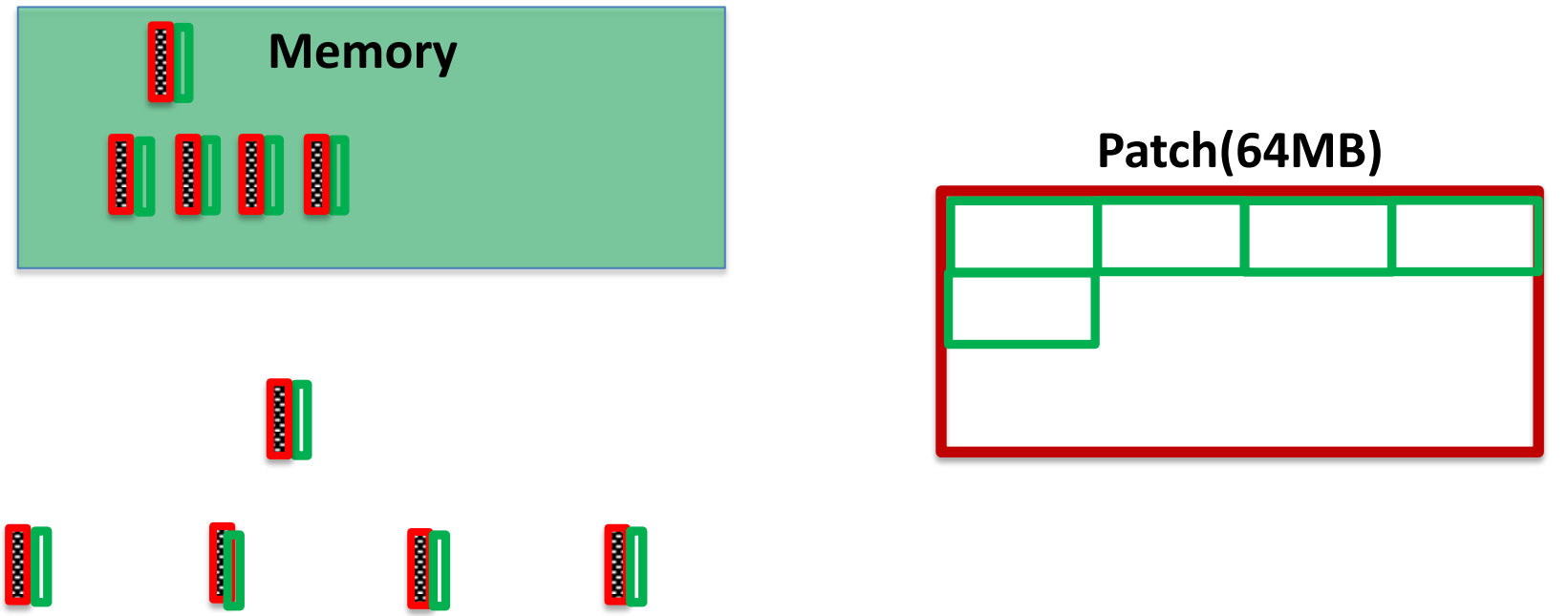
Reducing Compaction Cost

- In a KV item, value is usually much larger than the key.
- Values are not necessary to be involved in compactions.
- Move and place the values in a fixed-size container (block), and replace the values with pointers in KV items.



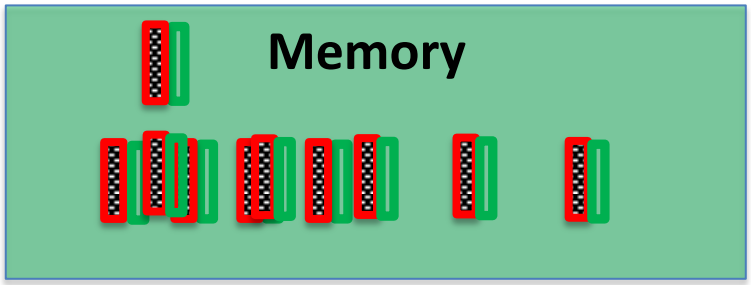
Reducing Compaction Cost

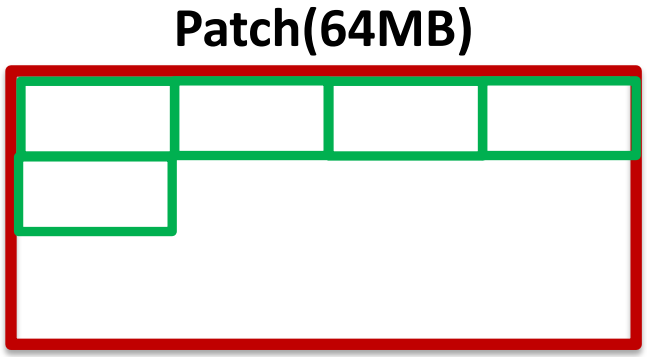
- In a KV item, value is usually much larger than the key.
- Values are not necessary to be involved in compactions.
- Move and place the values in a fixed-size container (block), and replace the values with pointers in KV items.



Reducing Compaction Cost

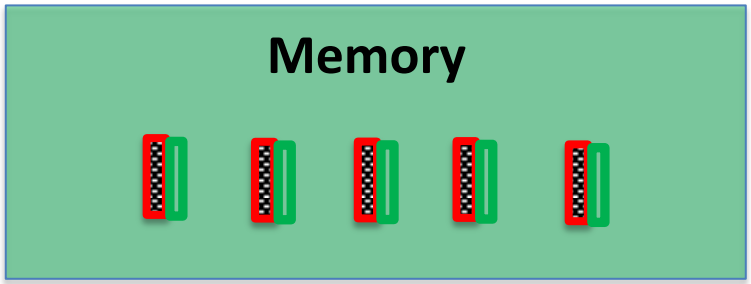
- In a KV item, value is usually much larger than the key.
- Values are not necessary to be involved in compactions.
- Move and place the values in a fixed-size container (block), and replace the values with pointers in KV items.

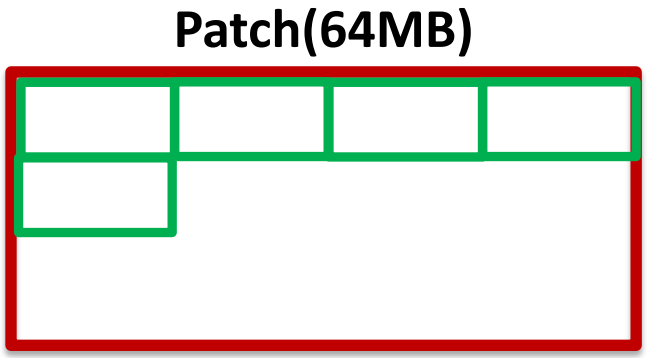




Reducing Compaction Cost

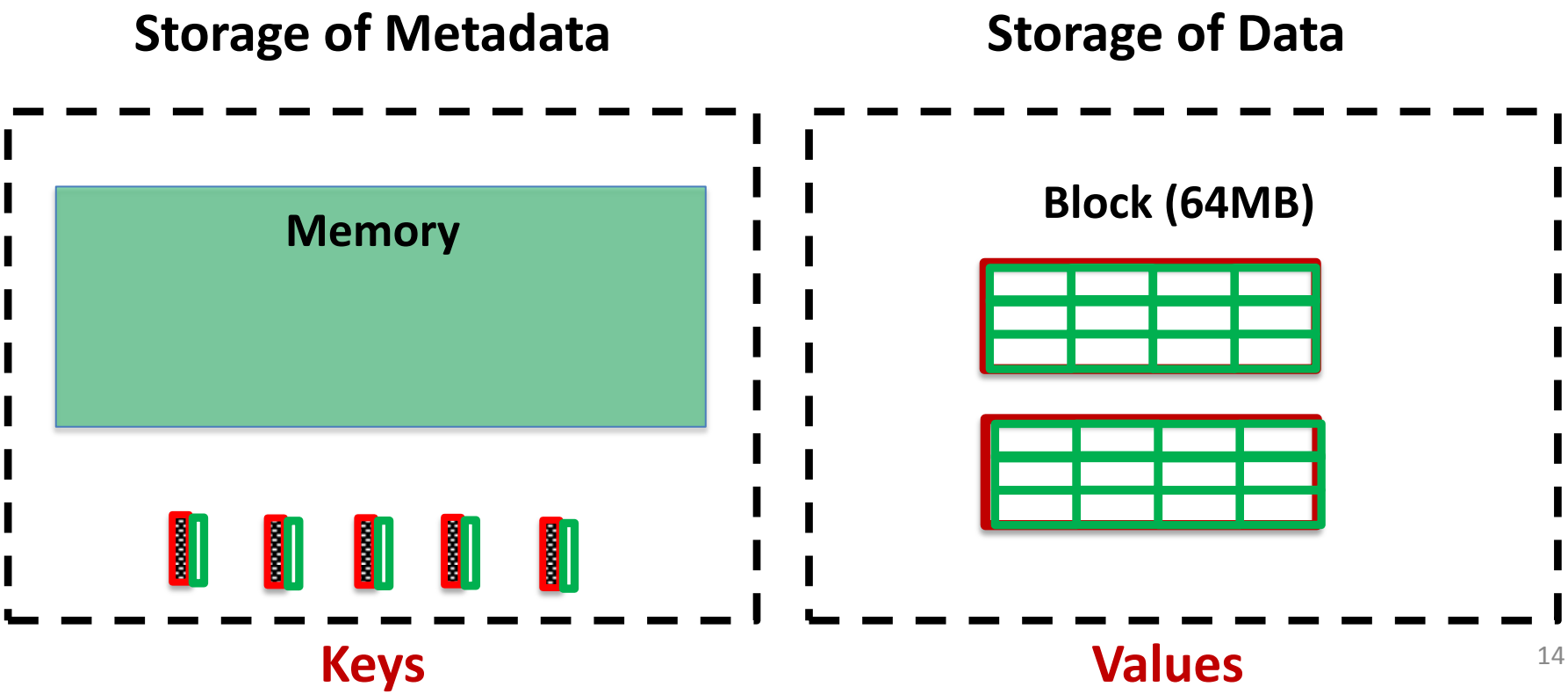
- In a KV item, value is usually much larger than the key.
- Values are not necessary to be involved in compactions.
- Move and place the values in a fixed-size container (block), and replace the values with pointers in KV items.



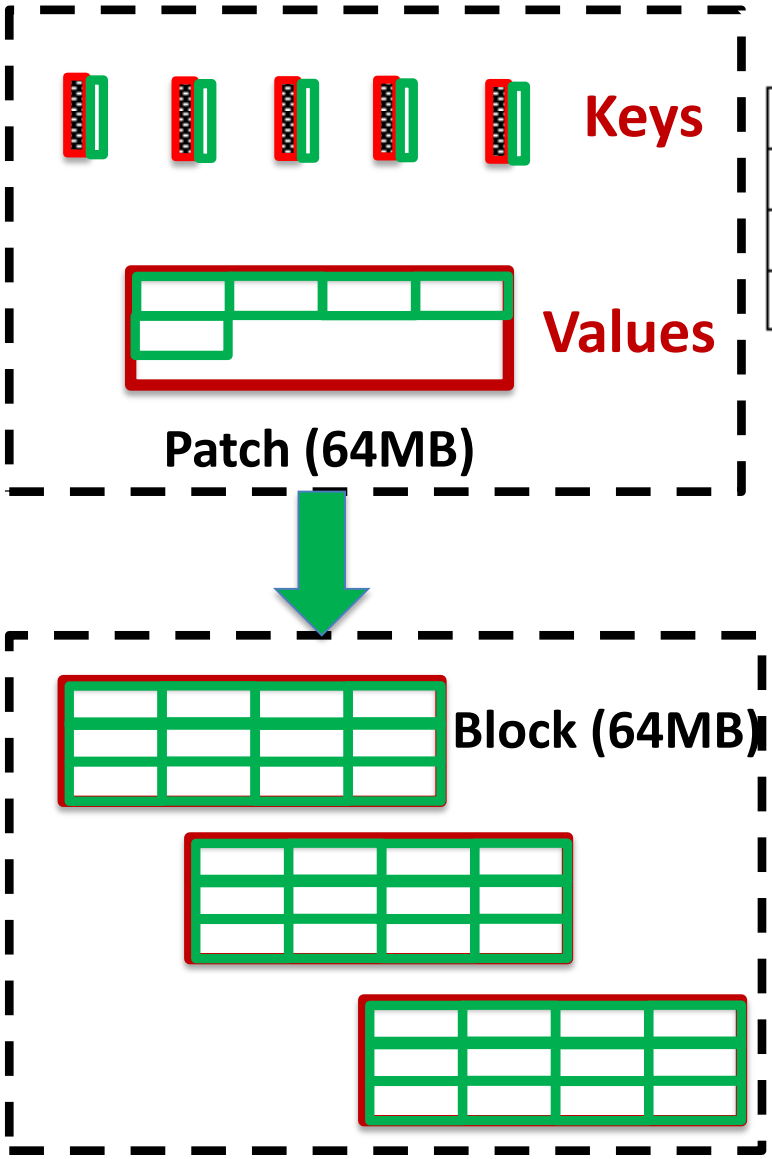


Features of Baidu's Cloud Storage System (Atlas)

- A hardware and software co-design with customized low-power servers for high resource utilization
- Separate metadata (keys and offsets) and data (value blocks) management systems.
- Data are efficiently protected by erasure coding.



Big Picture of the Atlas System



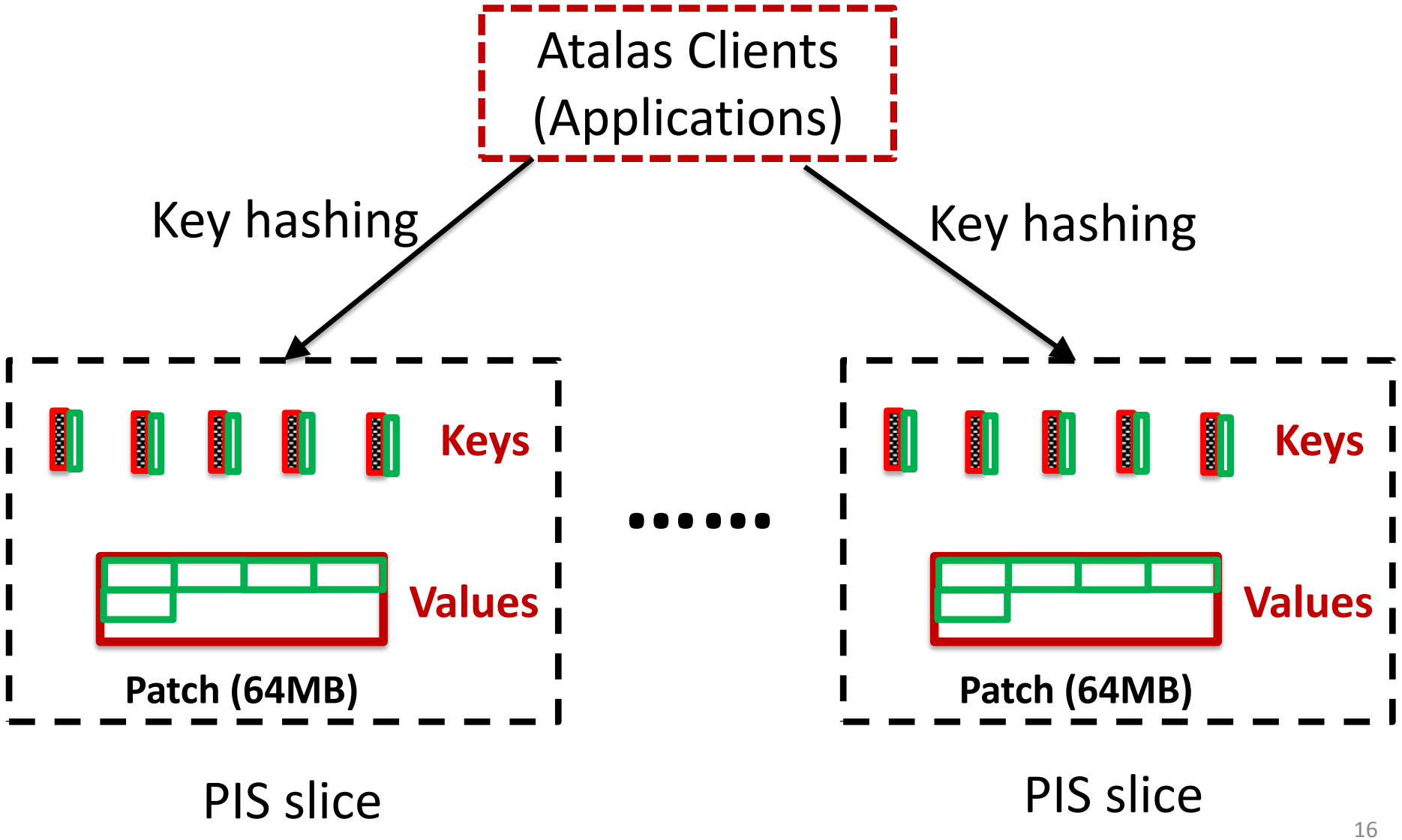
PIS (Patch and Index System)

Command	Format
Read	Get (UINT128 key, BYTE* value)
Write	Put (UINT128 key, BYTE *value)
Delete	Del (UINT128 key)

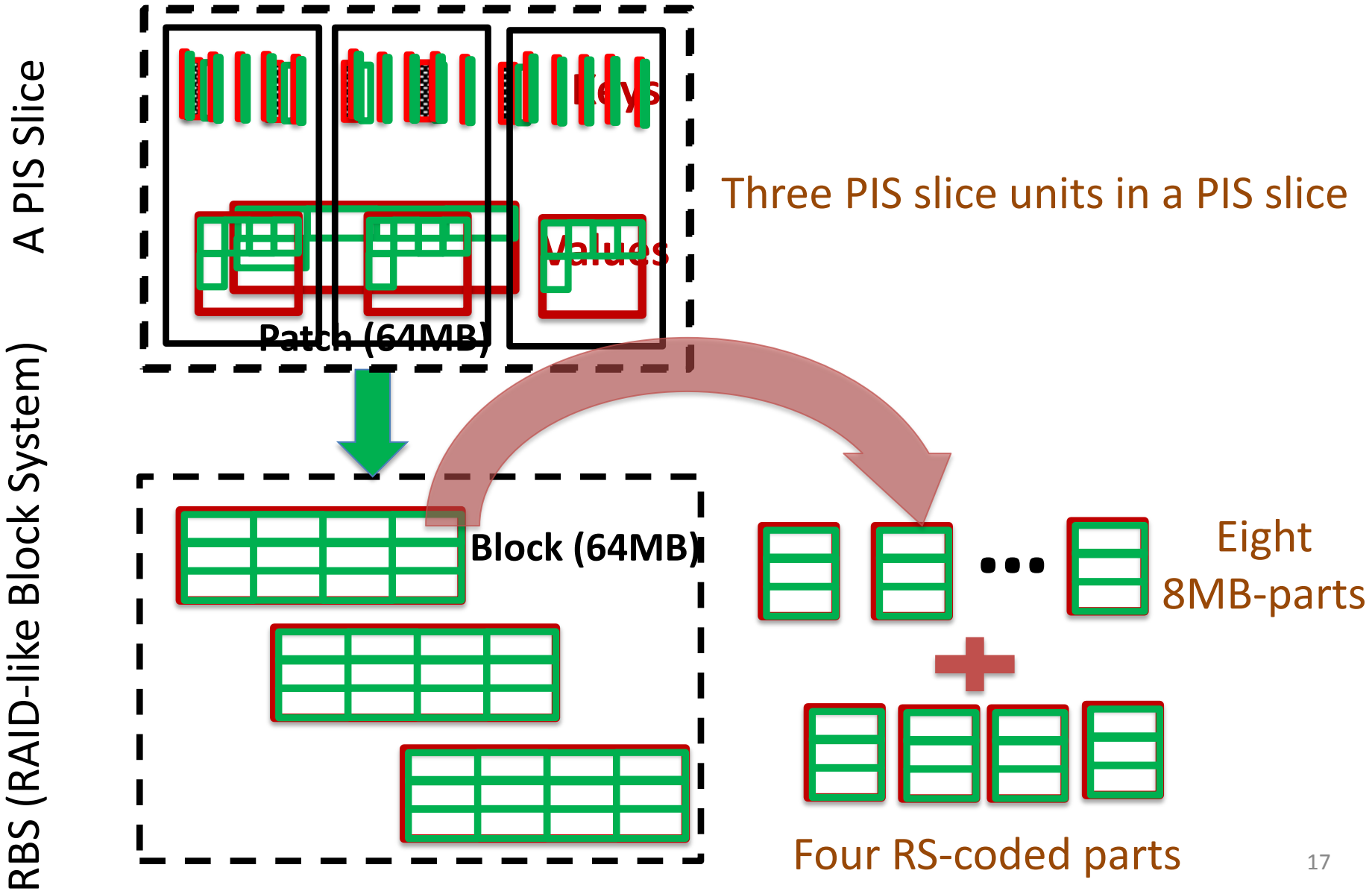
RBS (RAID-like Block System)

Command	Format
Write	Write (UINT64* block_id, BYTE *data)
Read	Read (UINT64 block_id, UINT32 offset, UINT32 length, BYTE* data)
Deletion	Delete (UINT64 block_id)

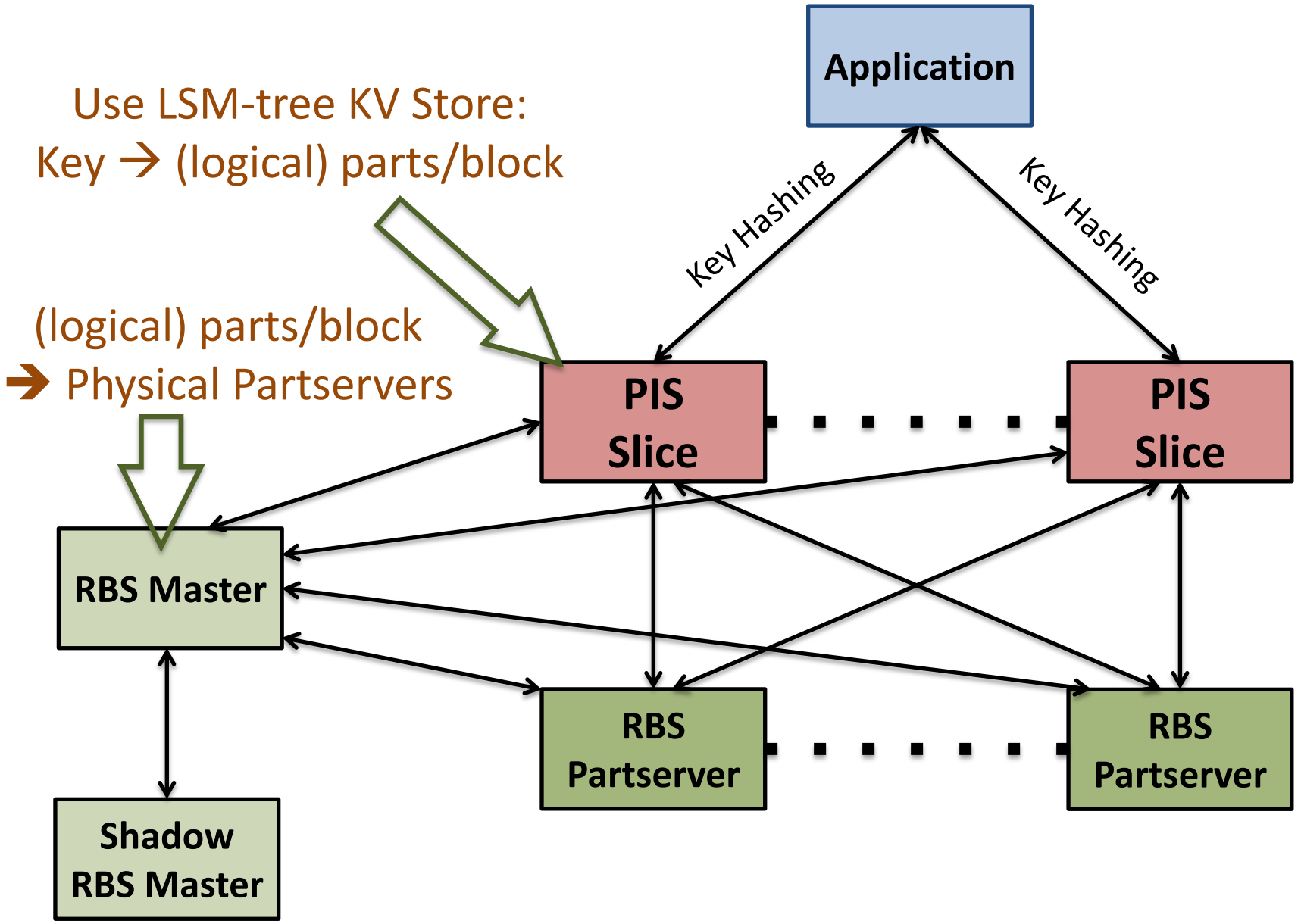
Distribution of User Requests



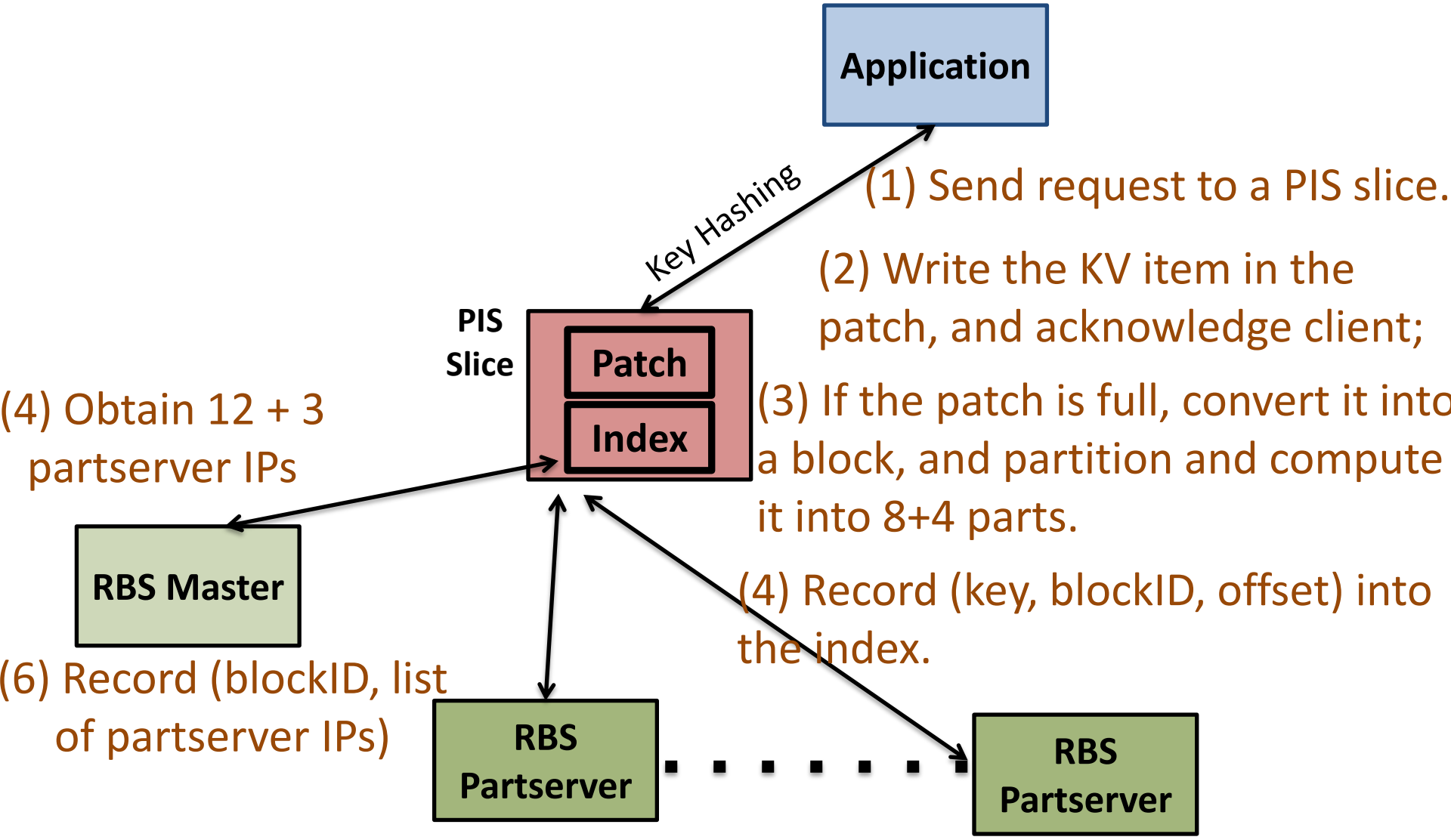
Redundancy for Protecting KV items



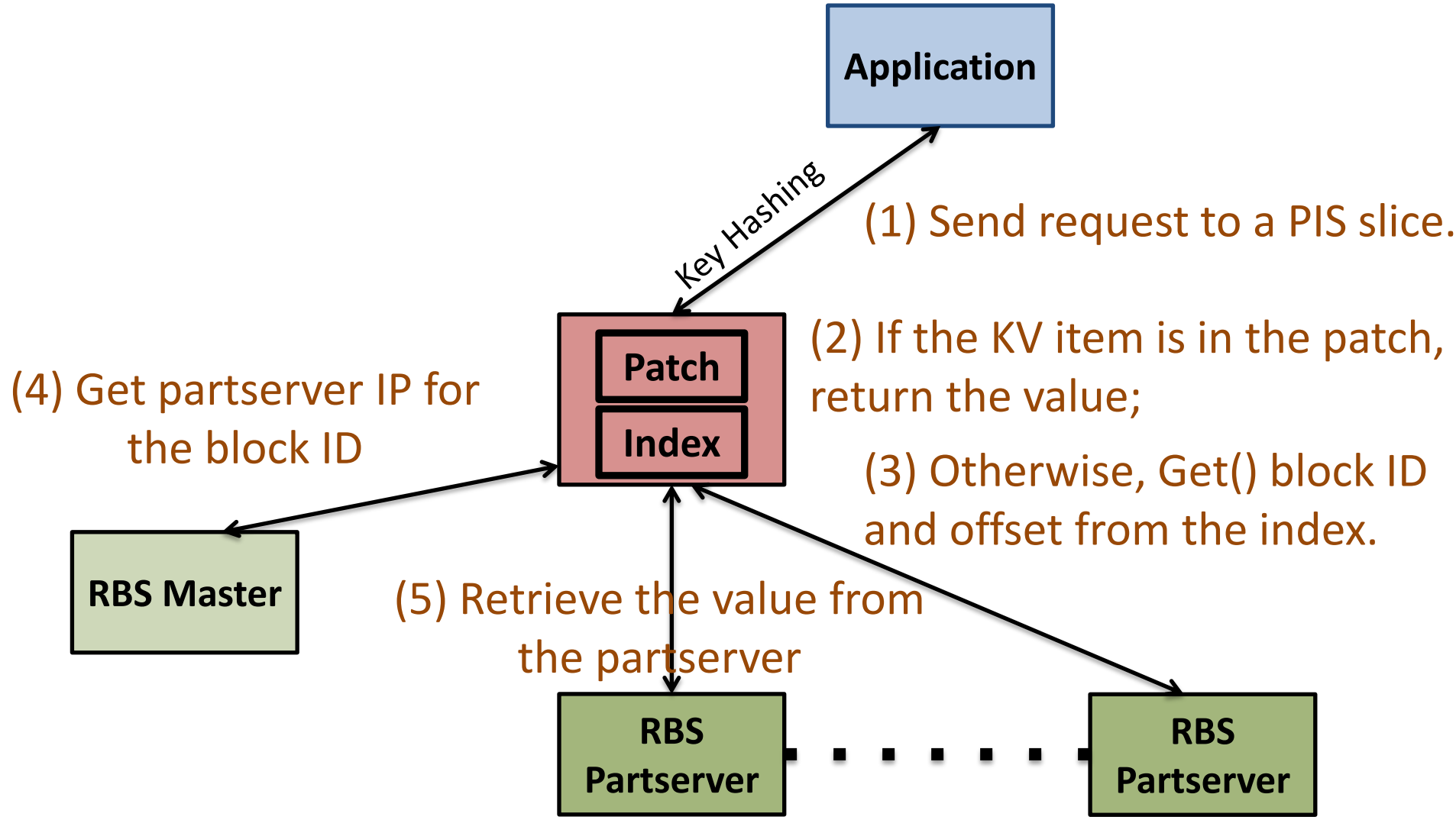
The Architecture of Atlas



Serving a Write Request



Serving a Read Request



Serving Delete/Overwrite Requests

- KV pairs stored in Atlas are immutable.
- Blocks in Atlas are also immutable.
- A new KV item is written into the system to service a delete/overwritten request.
- Space occupied by obsolete items are reclaimed in a garbage collection (GC) process.
- Periodically two questions are asked about a block in the RBS subsystem, and positive answers to both lead to a GC.

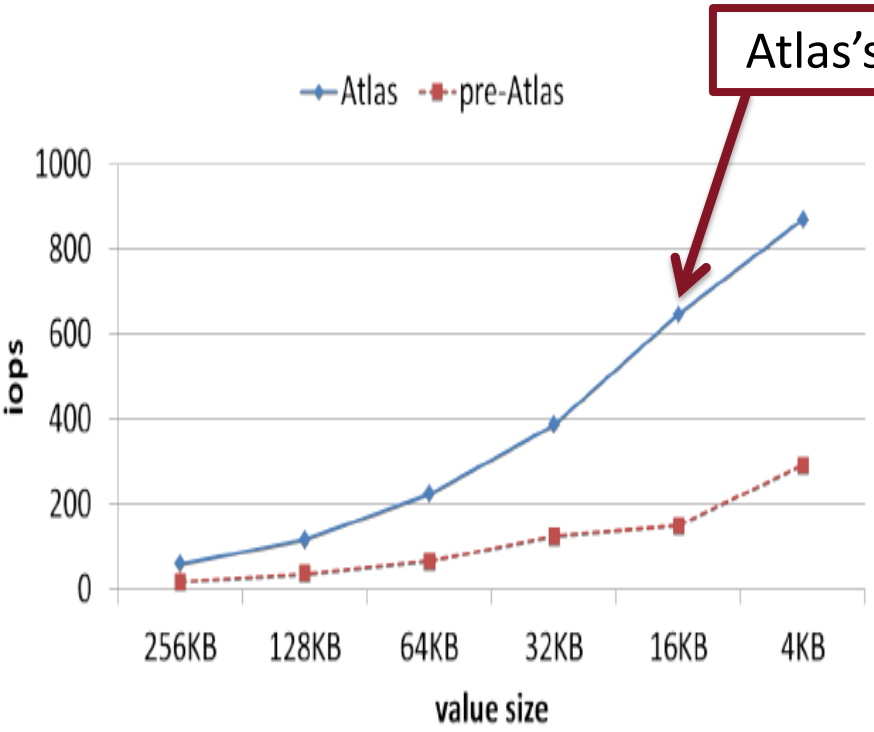
1) Is the block created earlier than a threshold (such as one week ago)?
2) Is the ratio of valid data in the block smaller than a threshold (such as 80%)?

Atlas's Advantages on Hardware Cost and Power

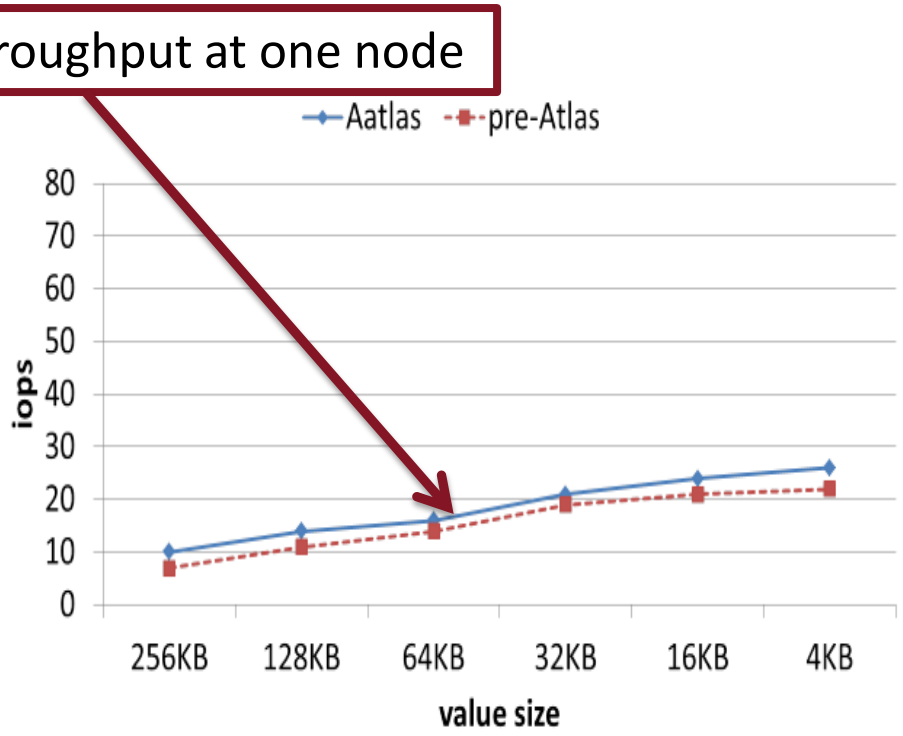
- Atlas saves about 70% of hardware cost per GB storage
  - Using ARM servers to replace x86 servers
  - Using erasure coding to replace 3-copy replication.
- Power consumption is reduced by about 53% per GB storage.
  - The ARM processors are more power efficient.
  - The ARM server racks are more space efficient, reducing energy cost for power supply and thermal dissipation.

Comparison with the Prior System

- Reference system (pre-Atlas)
  - Similar PIS subsystem.
  - All data are managed solely by the LSM-tree-based KV store.
- Run on a 12-server X86 cluster.



All writes

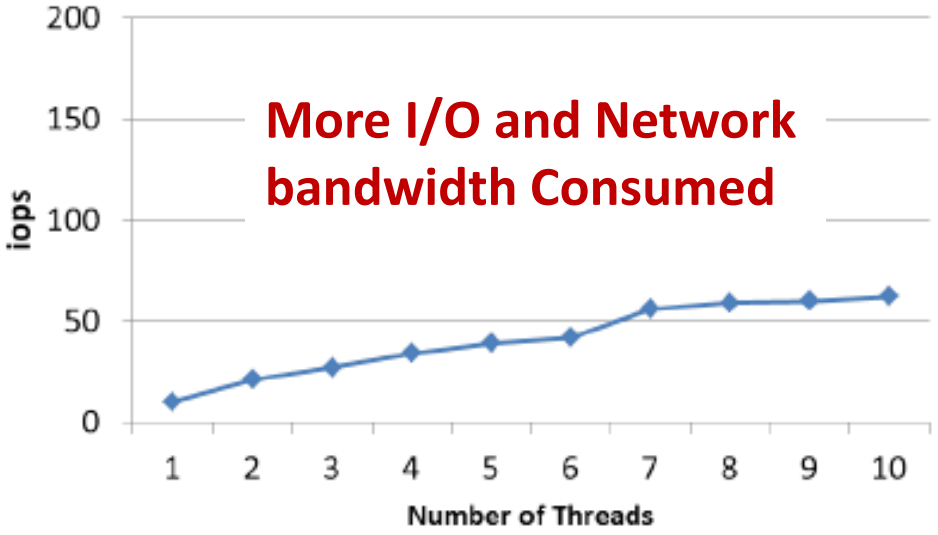


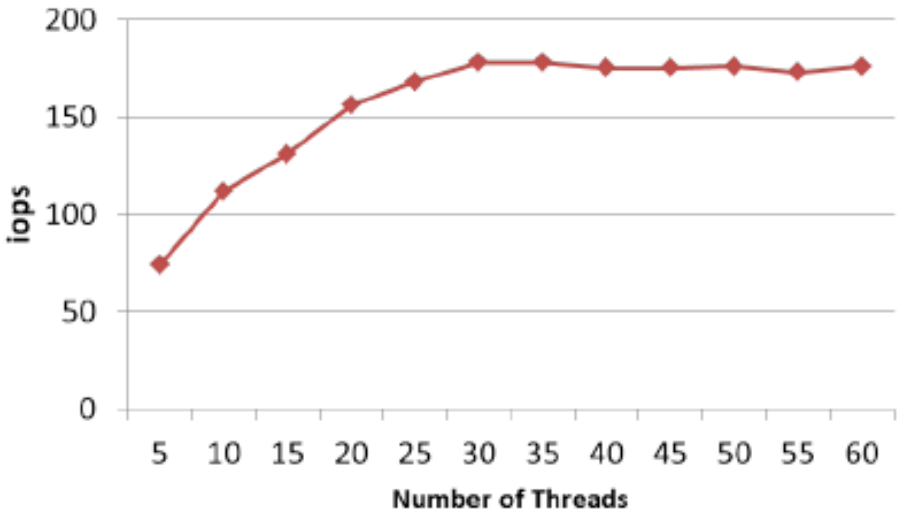
Read : Write = 3:1

Atlas on a Customized ARM cluster

- A cluster of 12 ARM servers.
- Each hosts multiple PIS slices and RBS partservers.
- Each server has a 4-core Marvell processor, 4GB memory, four 3TB disks.
- 1Gbps full-duplex Ethernet adapter.
- Request size is 256KB.

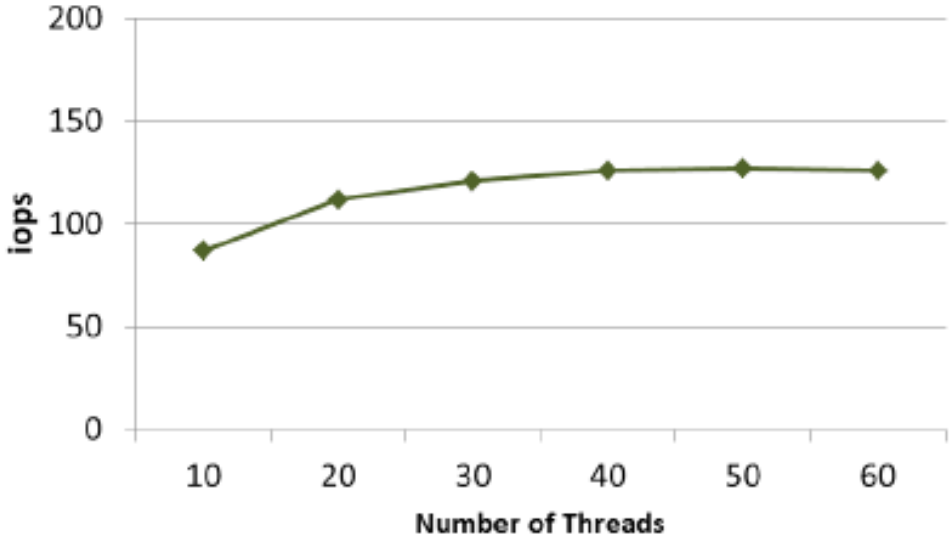
Throughput at One Node with Diff. Request Types





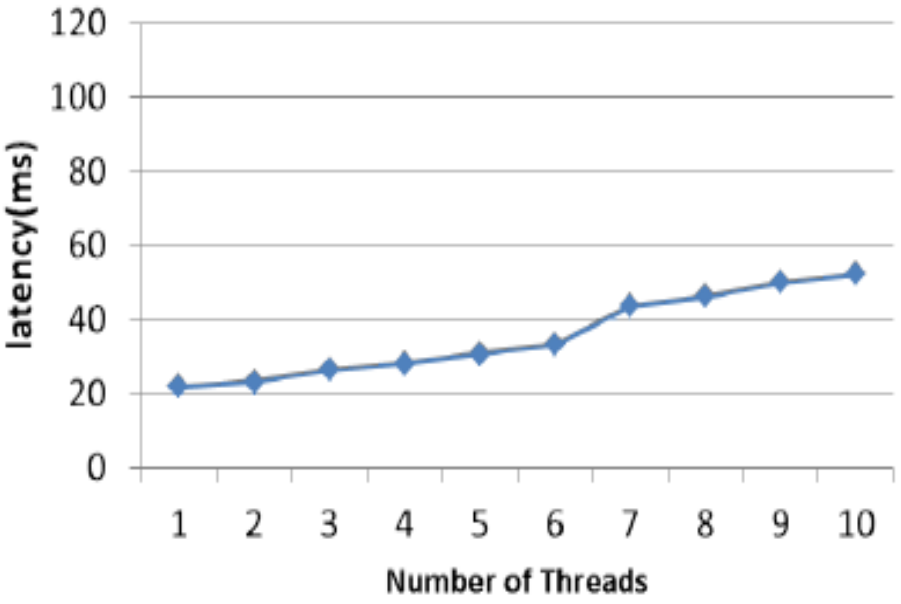
All writes

All Reads

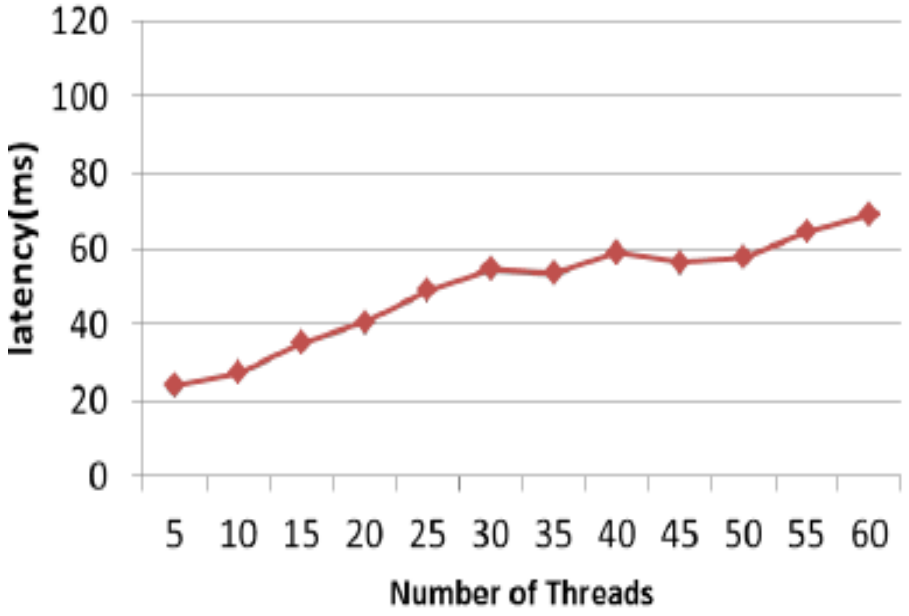


Read : Write = 3:1

Latencies with Diff. Request Types

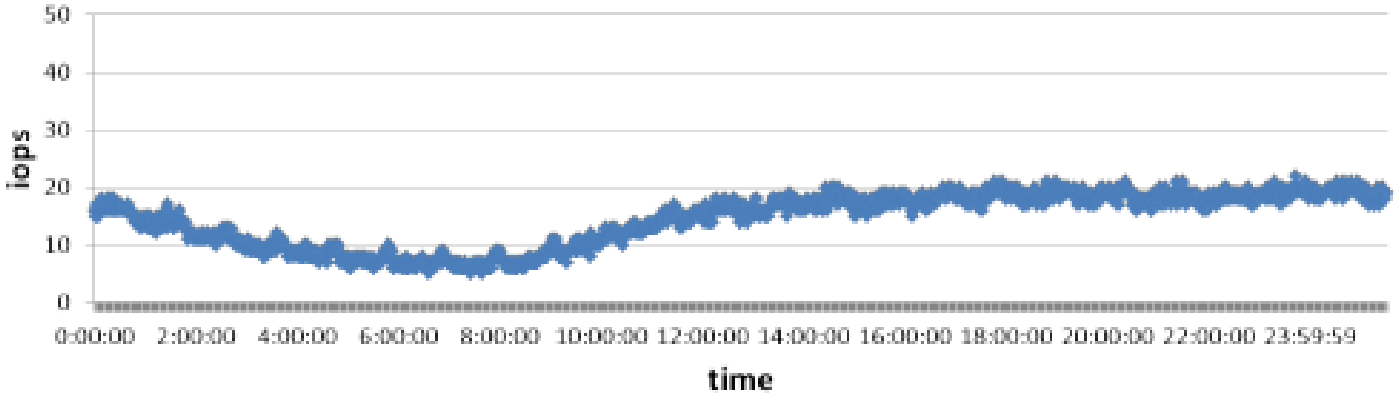


All writes

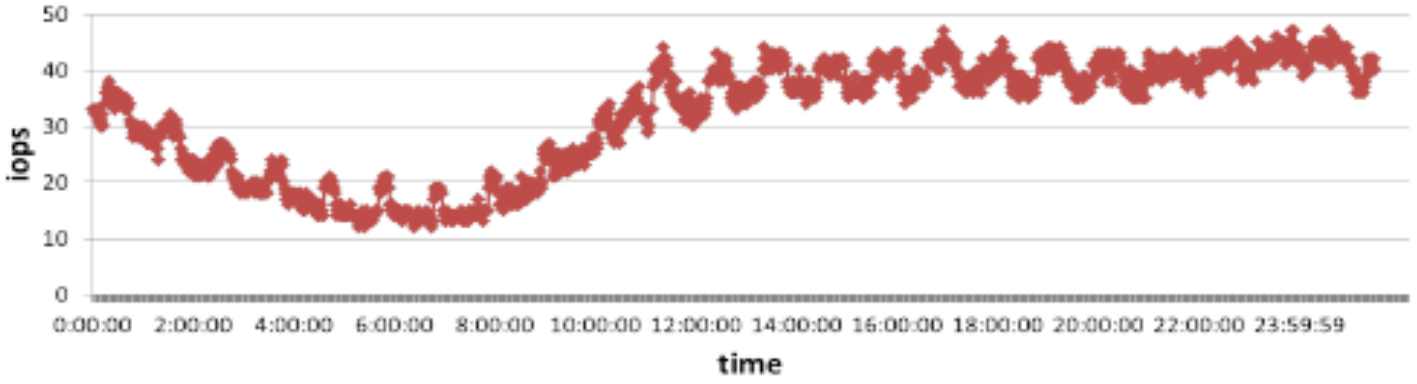


All Reads

Throughput at one Node of a Production System

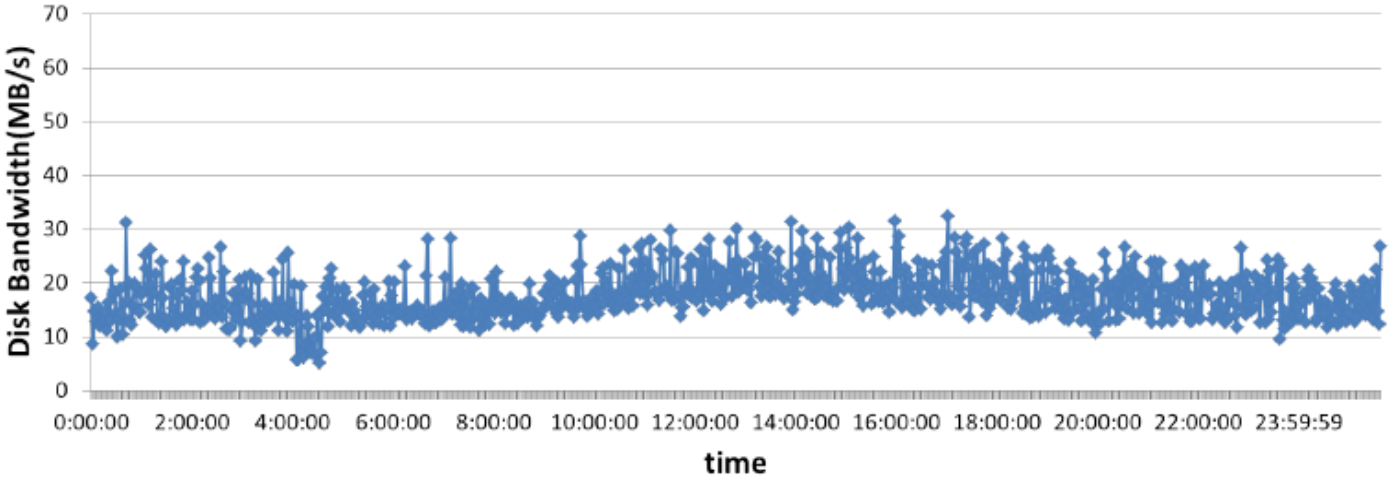


write

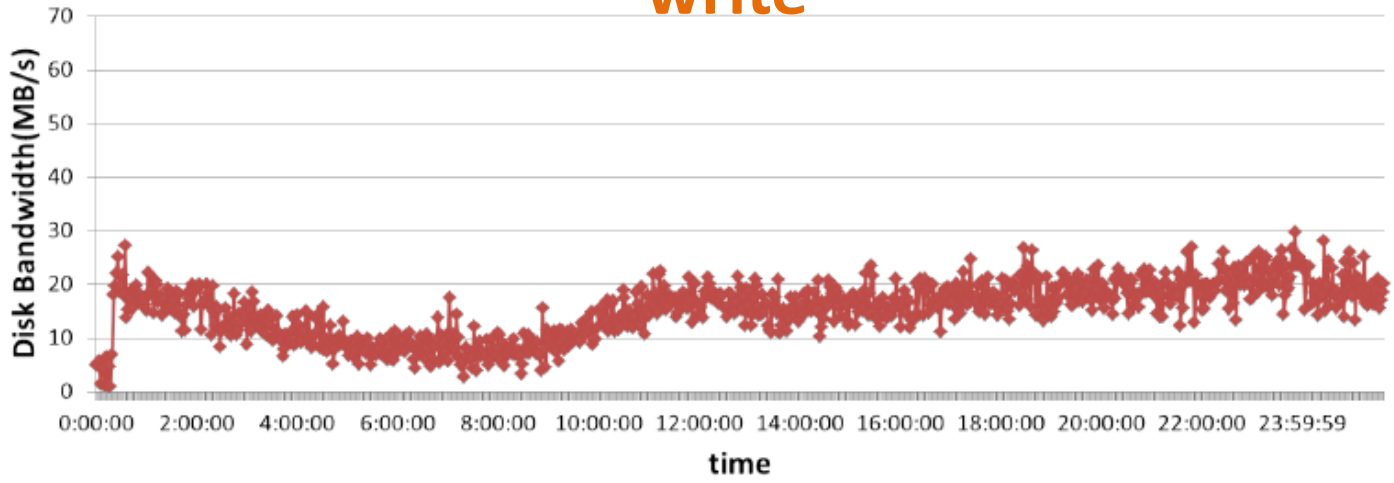


Reads

Disk Bandwidth at one Node of a Production System



write



Reads

Summary

- Atlas is an object store using a two-tier design separating the managements of keys and values.
- Atlas uses a hardware-software co-design for high cost-effectiveness and energy efficiency.
- Atlas adopts the erasure coding technique for space-efficient data protection.