A Scalable System Design for Data Reduction in Modern Storage Servers

Mohammadamin Ajdari

Presentation at Dpt. Of Computer Engineering, Sharif Univ. of Tech.
2020/1/22

My Education

Direct PhD in Computer Eng.

Degree from *POSTECH* (South Korea)

[2013 - 2019]

BSc in Electrical Eng. (Electronics)

Degree from *Sharif Univ. of Tech.* (Iran)

[2008-2013]

Long-Term Research/Engineering Projects

PhD

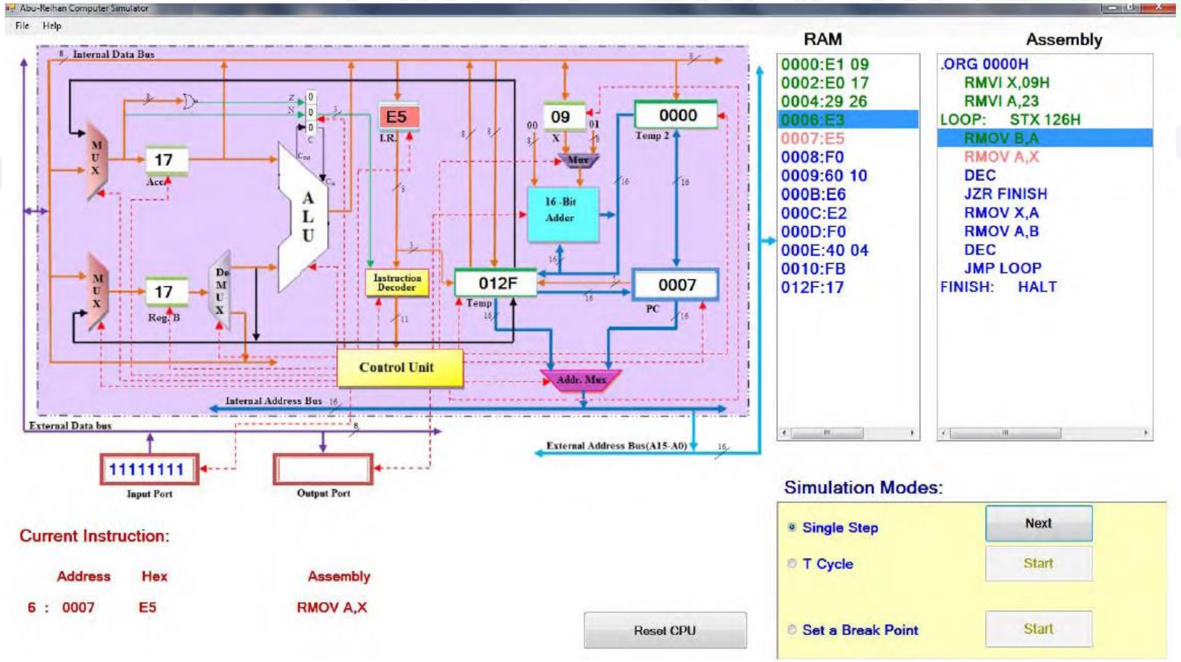
- **Scalable data reduction architecture (main author)**
  - **CAL'17, HPCA'19** (Best Paper Nominee), **MICRO'19**
  - **IEEE MICRO Top Pick'19 (**Honorable Mention**)**
- **Device centric server architecture (co-author)**
  - **MICRO'15, ISCA'18**
- **CPU performance modeling (co-author)**
  - **TACO'18**

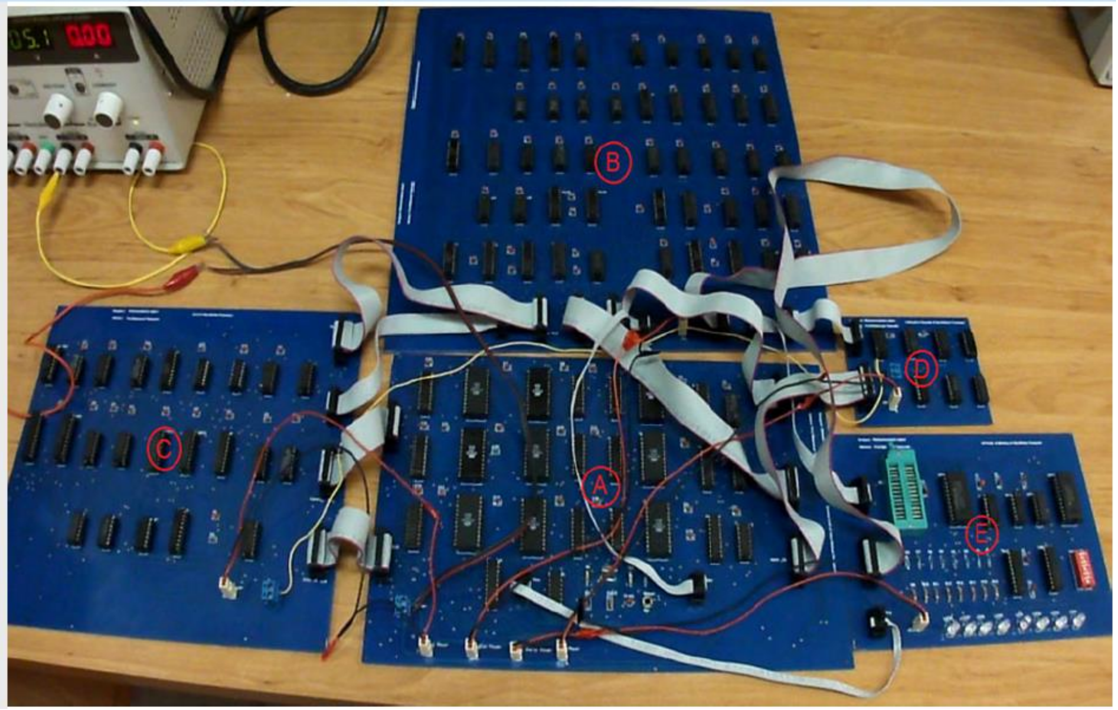
BSc

- **Design of a real computer system from scratch (main author)**
  - **ICL'12, IJSTE'16** (Best BSc Project Award)

Long-Term Research/Engineering Projects

- Scalable data reduction architecture (main author)

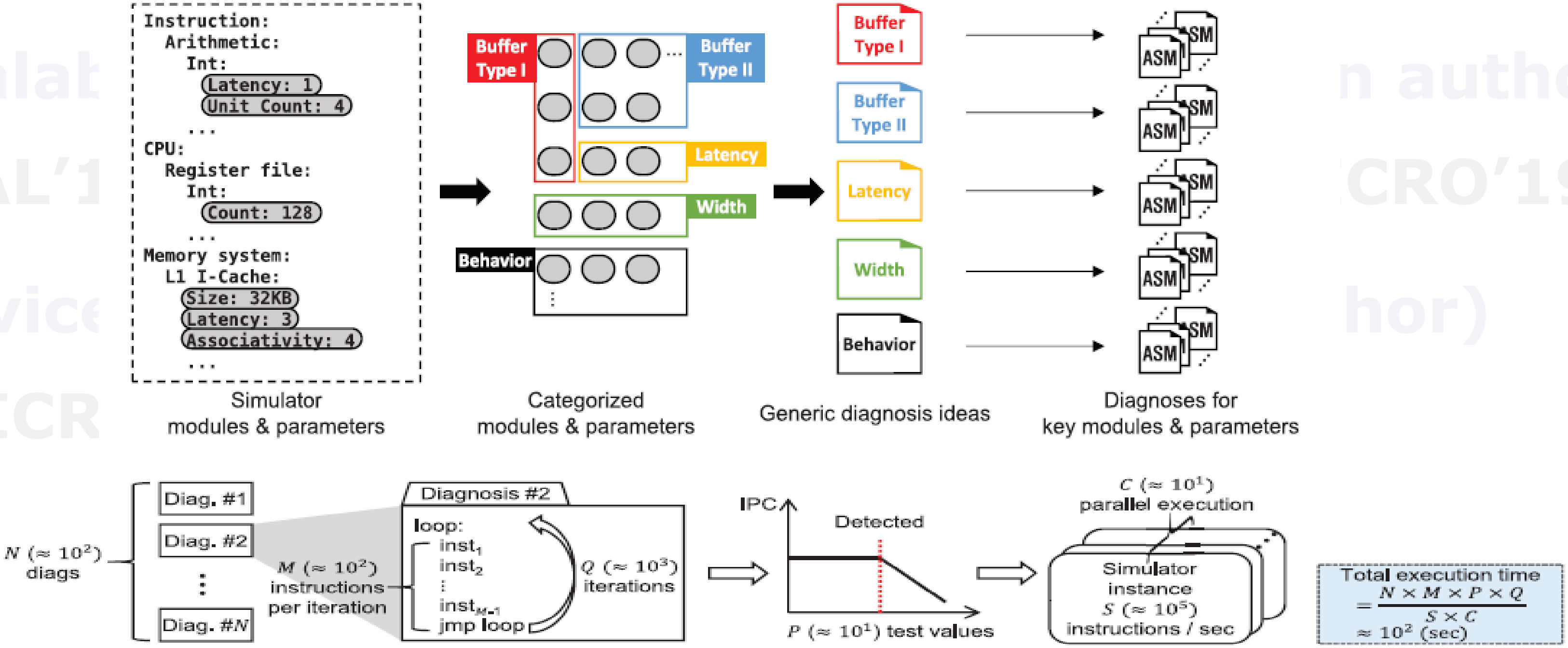




BSc

- **Design of a real computer system from scratch (main author)**
  - **ICL'12, IJSTE'16** (Best BSc Project Award)

Long-Term Research/Engineering Projects



PhD

- **CPU performance modeling (co-author)**
  - **TACO'18**

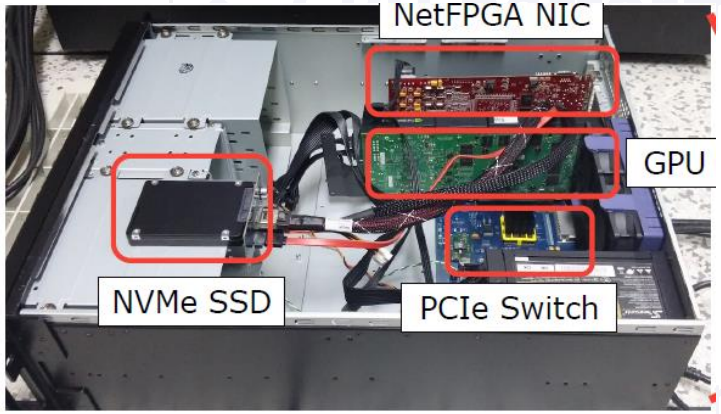
*JE Jo, GH Lee, H Jang, J Lee, M Ajdari, J Kim, "*DiagSim: Systematically Diagnosing Simulators for Healthy Simulations*", TACO 2018

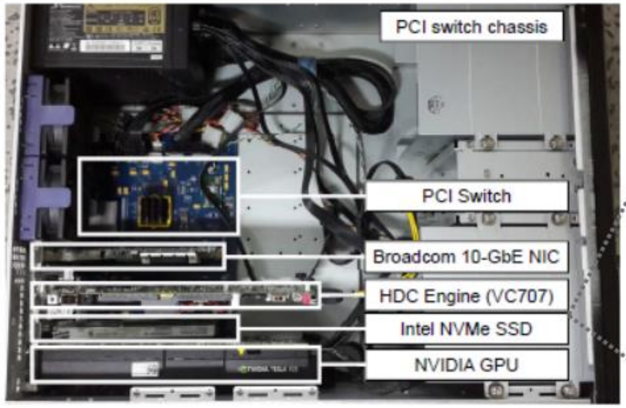
Long-Term Research/Engineering Projects

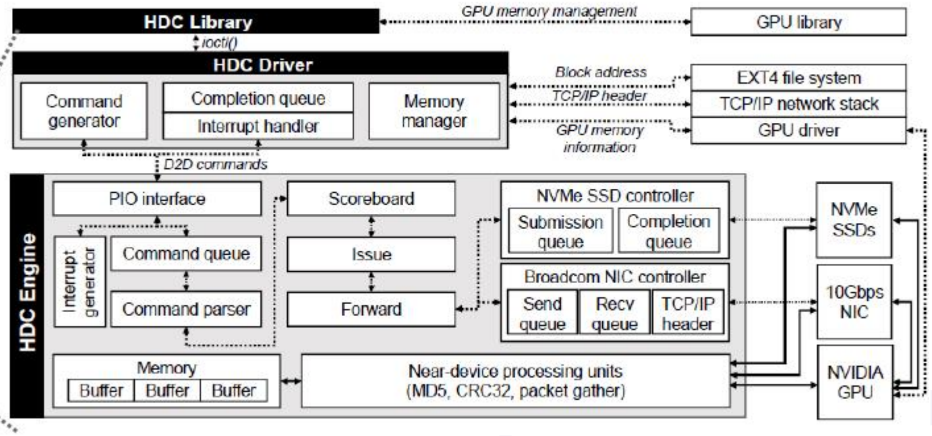
- Scalable data reduction architecture (main author)
  - CAL'17, HPCA'19 (Best Paper Nominee), MICRO'19

PhD

- **Device centric server architecture (co-author)**
  - **MICRO'15, ISCA'18**





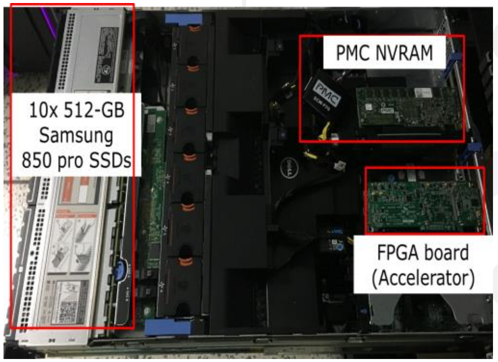


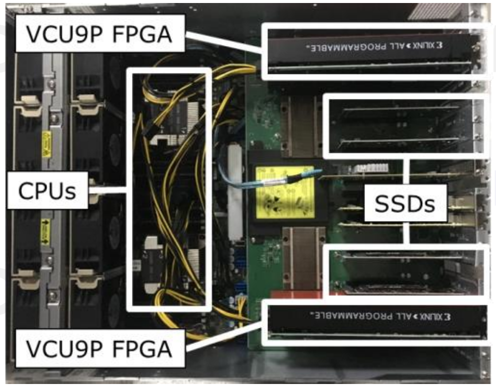
*J Ahn, D Kwon, Y Kim, M Ajdari, J Lee, J Kim, "*DCS: A fast and scalable device-centric server architecture*", MICRO 2015

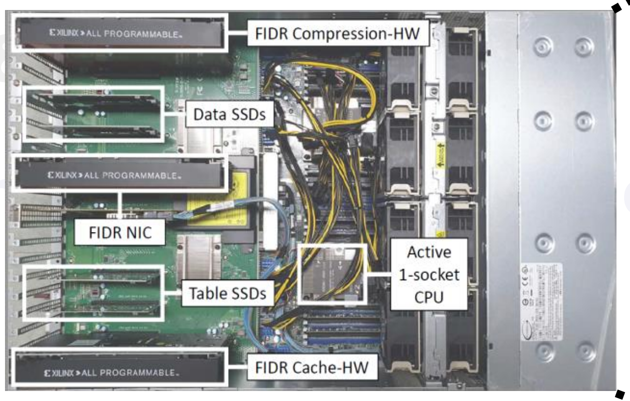
** D Kwon, J Ahn, D Chae, M Ajdari, J Lee, S Bae, Y Kim, J Kim, "*DCS-ctrl: A fast and flexible device-control mechanism for device-centric server architecture*", ISCA 2018

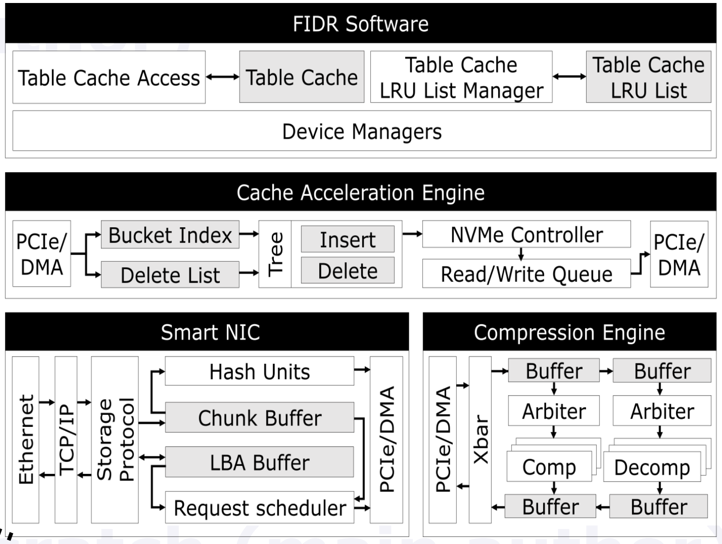
Long-Term Research/Engineering Projects

- **Scalable data reduction architecture (main author)**
  - **CAL'17, HPCA'19** (Best Paper Nominee), **MICRO'19**
  - **IEEE MICRO Top Pick'19** (Honorable Mention)









*M Ajdari, P Park, D Kwon, J Kim, J Kim, "*A scalable HW-based inline deduplication for SSD arrays* ", IEEE CAL 2017

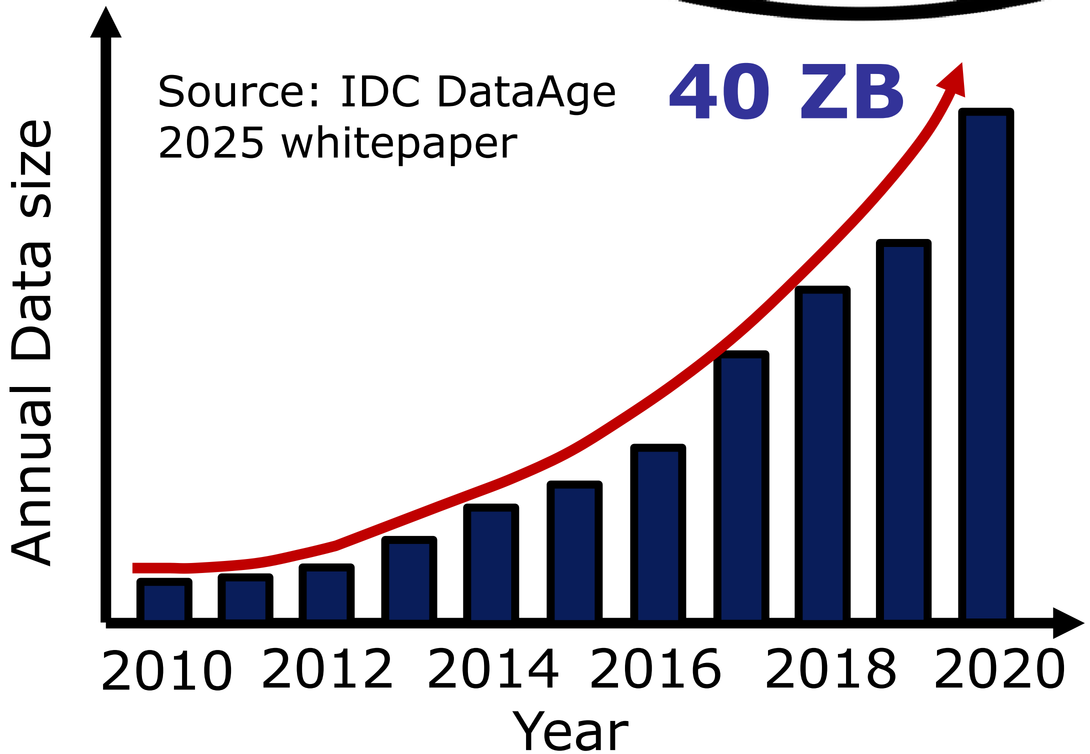
** M Ajdari, P Park, J Kim, D Kwon, J Kim, "*CIDR: A cost-effective in-line data reduction system for terabit-per-second scale SSD arrays*", HPCA 2019

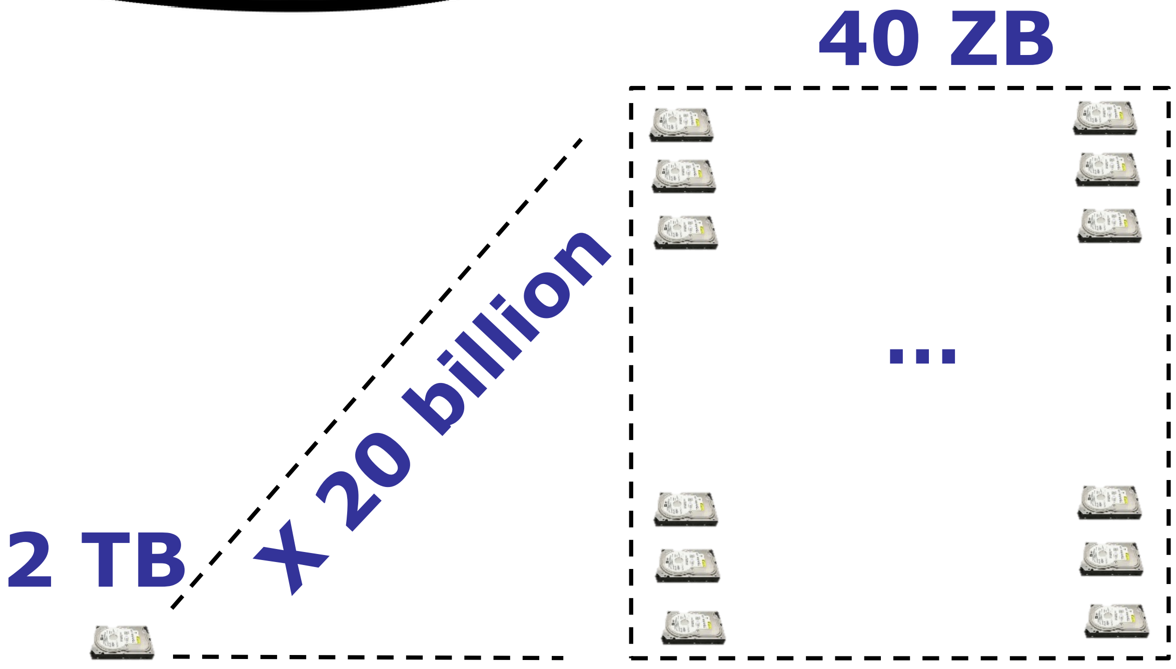
*** M Ajdari, W Lee, P Park, J Kim, J Kim, "*FIDR: A scalable storage system for fine-grain inline data reduction with efficient memory handling*", MICRO 2019

Index

- **Background**
  - Storage Systems and Trends
  - Basics of Data Reduction Techniques
- **Proposing New Data Reduction Architecture**
  - Deduplication for slow SSD Arrays
  - Deduplication and Compression for fast SSD Arrays
  - Optimizing for Ultra-scalability & more Workload Support
- **Conclusion**

Data Storage is Very Important





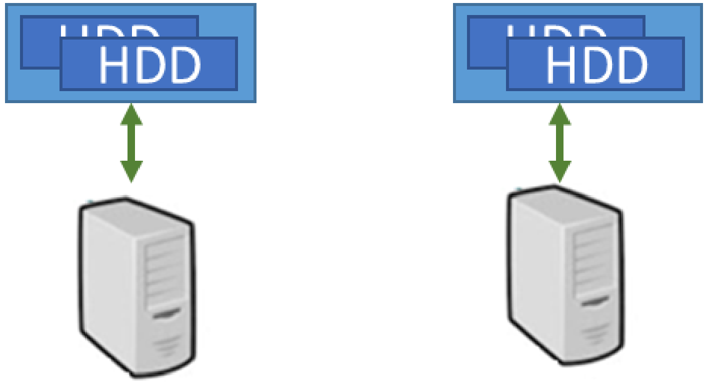
Storage System Types

➢ **Depends on type of HDD/SSD connection to a server**

1

Directly attached

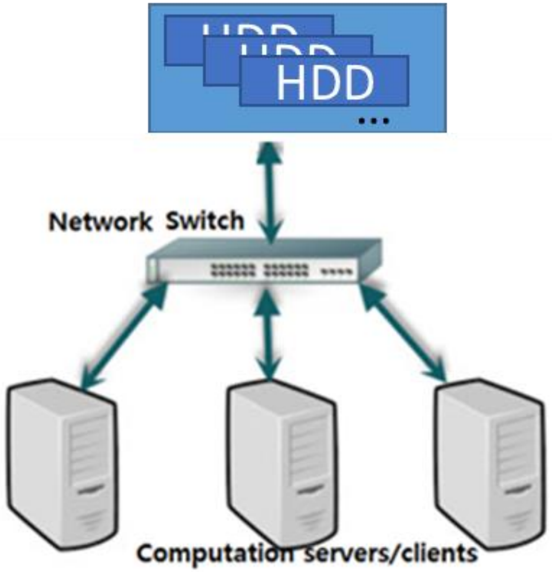
to the server motherboard



2

Indirectly attached

over a switched network



Storage System #1: Direct-Attached

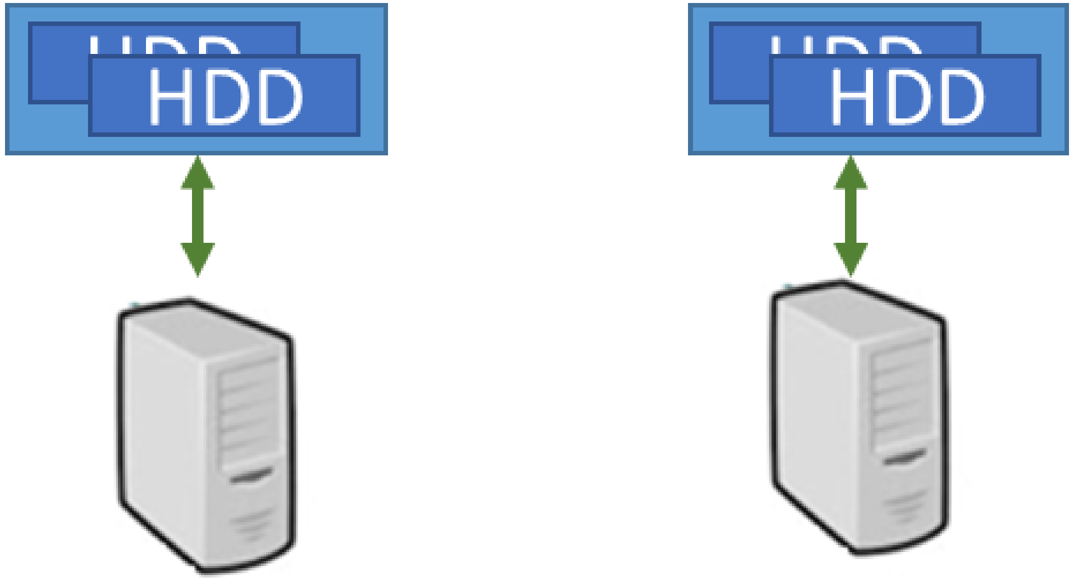
➢ Direct Attached Storage (DAS)

- Attach storage device (e.g., HDD) directly to the server

➢ Benefits

- Simple implementation
- Each server has fast access to

Its local storage

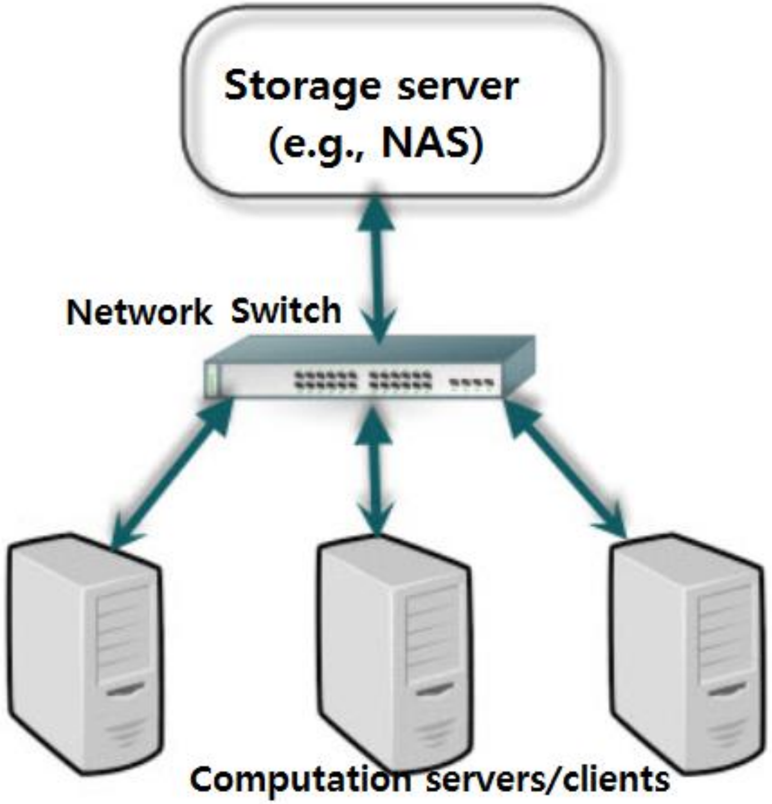


➢ Problems

- Storage & computation resources cannot scale independently
- Slow data sharing across nodes

Storage System #2: Network Attached

- **Storage over a switched network**
  - Storage system is almost a separate server on network (e.g., NAS)
- **Benefits**
  - Independent storage scalability
  - High reliability
  - Fast data sharing across nodes
- **Problems**
  - Complex implementation



In this talk, this is our choice of storage system

Storage Device Trend

HDD

Capacity : 2TB- **8 TB**
Throughput: **200 MB/s**
Latency : **over 1 ms**

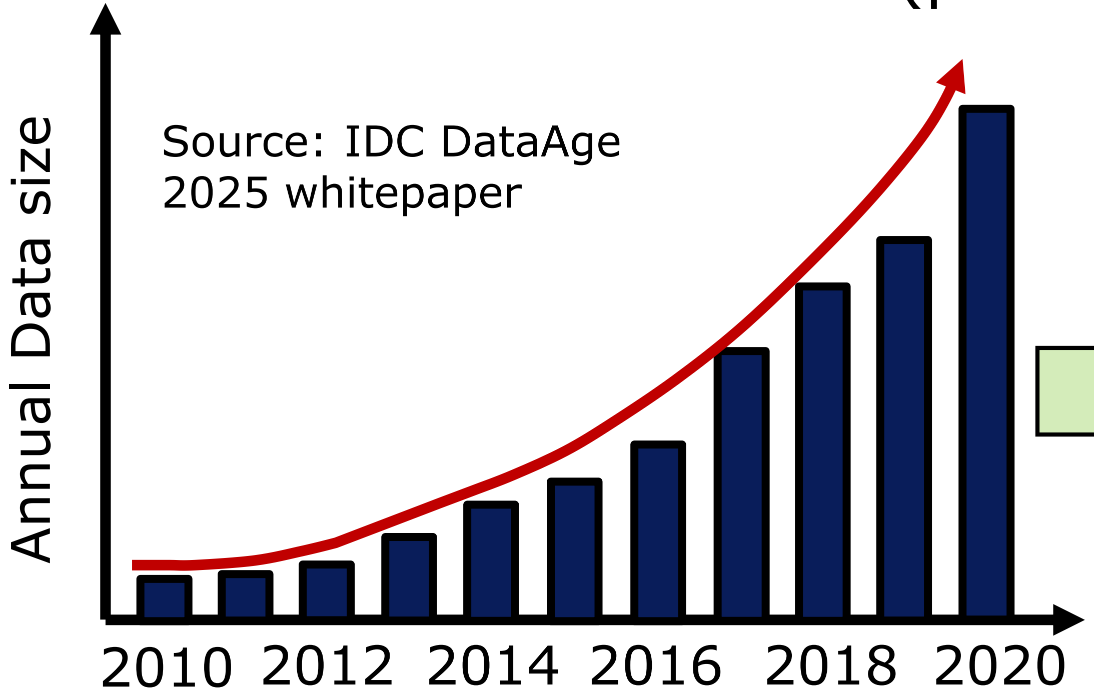
SSD

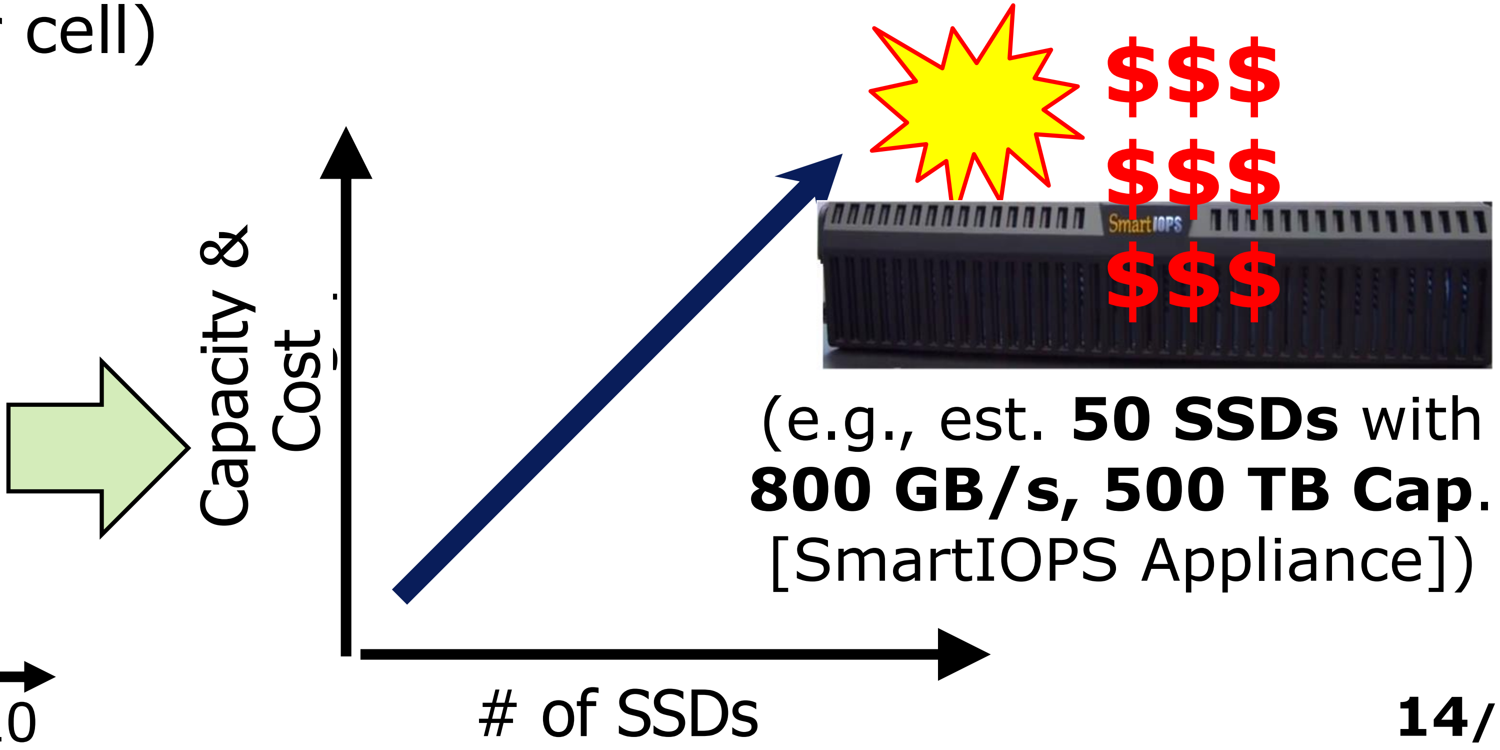
1 TB - **32 TB**
2 GB/s - **6.8 GB/s**
Over 20 µs

Fast, high capacity SSDs are replacing HDDs

But Modern Storage is Very Expensive

- **Average SSD Price Compared to HDD**
  - **3x-5x higher cost** (MLC SSD vs. HDD)
- **Limited lifetime of SSD flash cells**
  - **Max 5K-10K writes** (per cell)



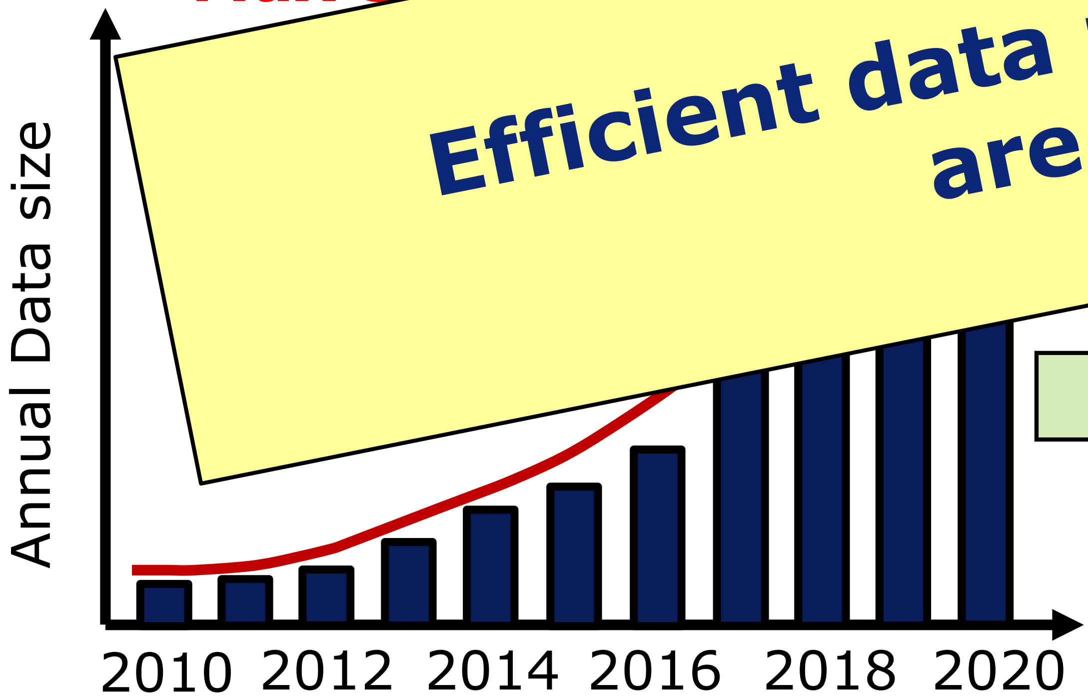


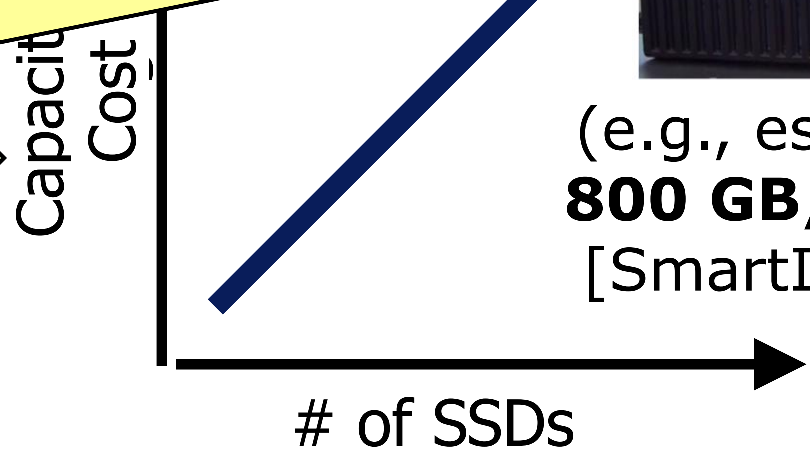
(e.g., est. **50 SSDs** with **800 GB/s, 500 TB Cap.** [SmartIOPS Appliance])

But Modern Storage is Very Expensive

- **Average SSD Price Compared to HDD**
  - **3x-5x higher cost** (MLC SSD vs. HDD)
- **Limited lifetime of SSD fl**
  - **Max 5K-10**

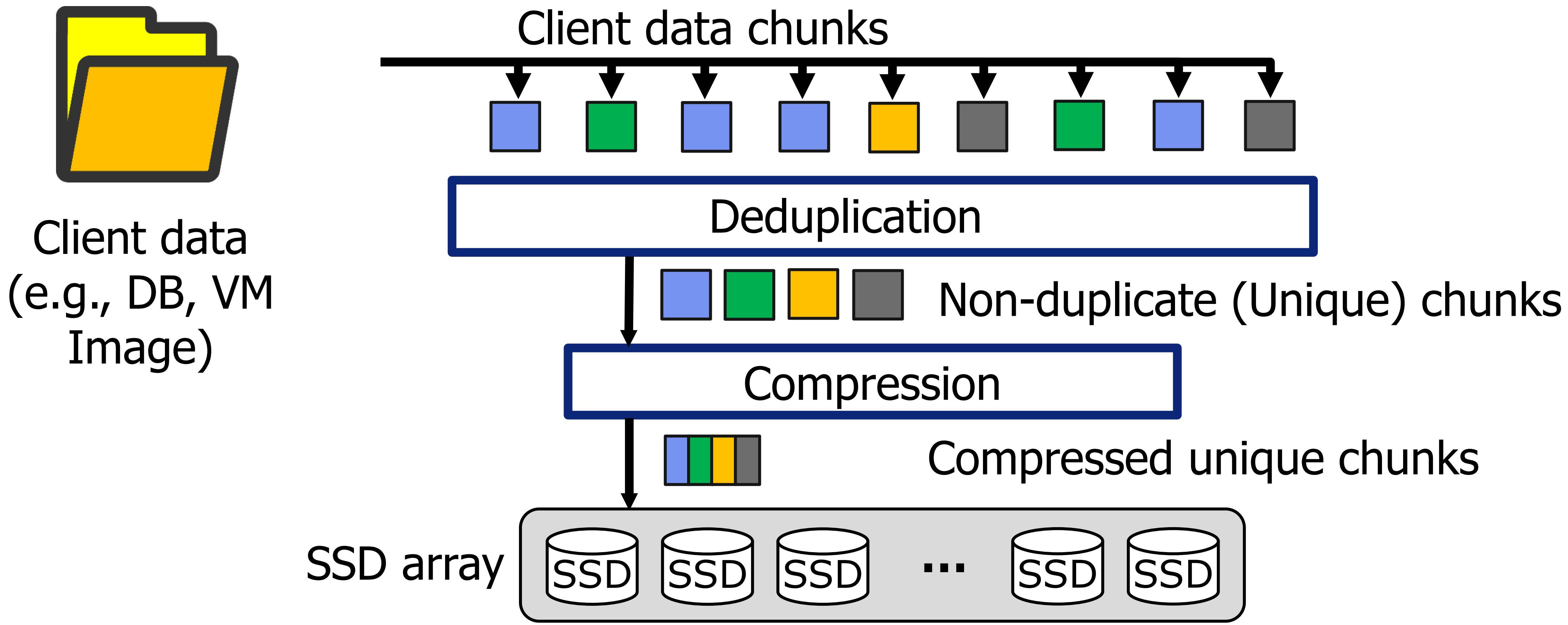
Efficient data reduction techniques are necessary!





(e.g., est. **50 SSDs** with **800 GB/s, 500 TB Cap.** [SmartIOPS Appliance])

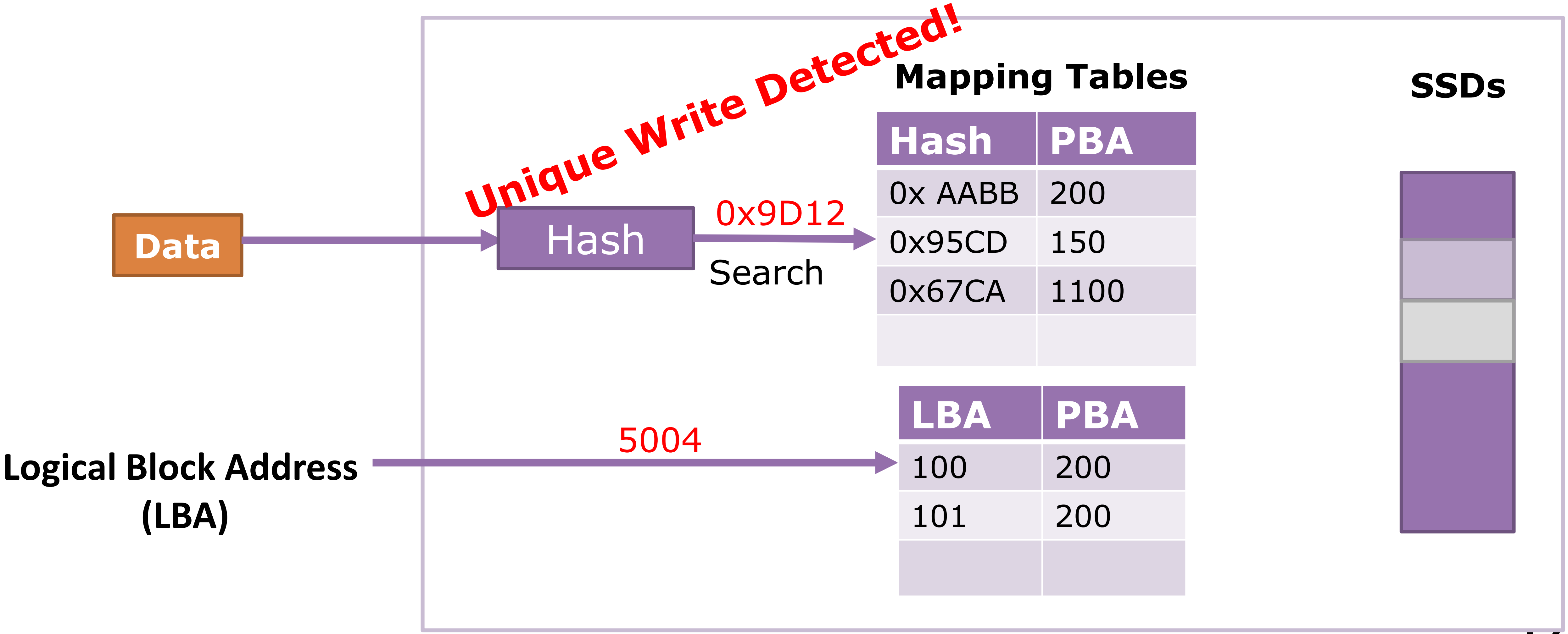
Data Reduction Overview



Deduplication + Compression
→ 60%-90% data reduction

Data Deduplication Basic Flow

- **Unique data write**

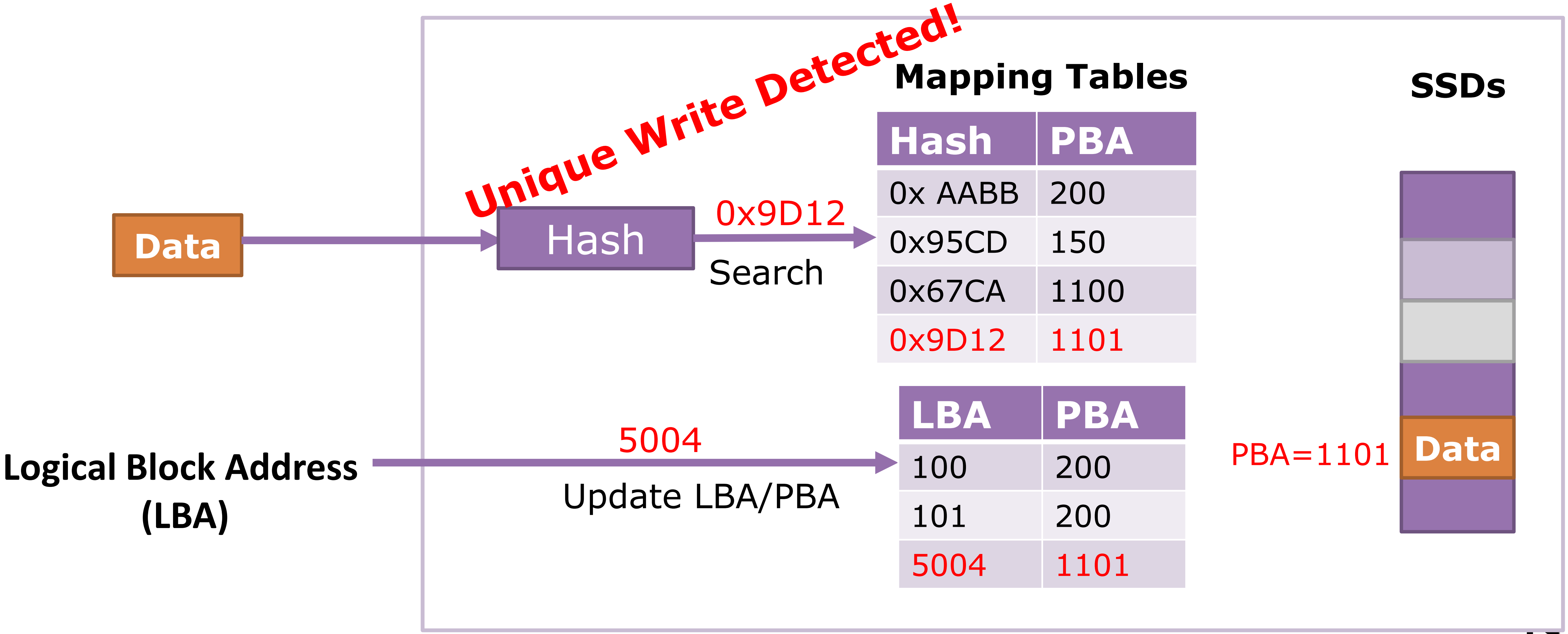


Hash	PBA
0x AABB	200
0x95CD	150
0x67CA	1100

LBA	PBA
100	200
101	200

Data Deduplication Basic Flow

➢ **Unique data write**

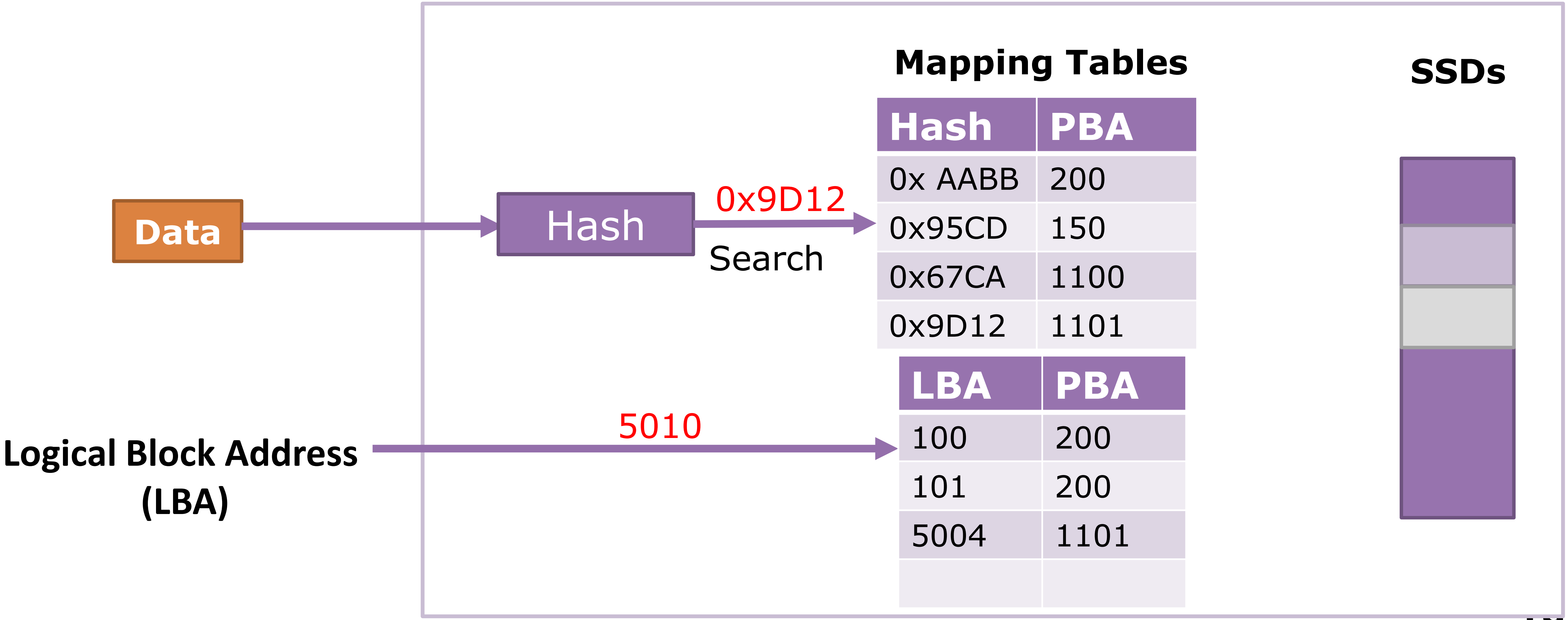


Hash	PBA
0x AABB	200
0x95CD	150
0x67CA	1100
0x9D12	1101

LBA	PBA
100	200
101	200
5004	1101

Data Deduplication Basic Flow

➢ **Duplicate data write**

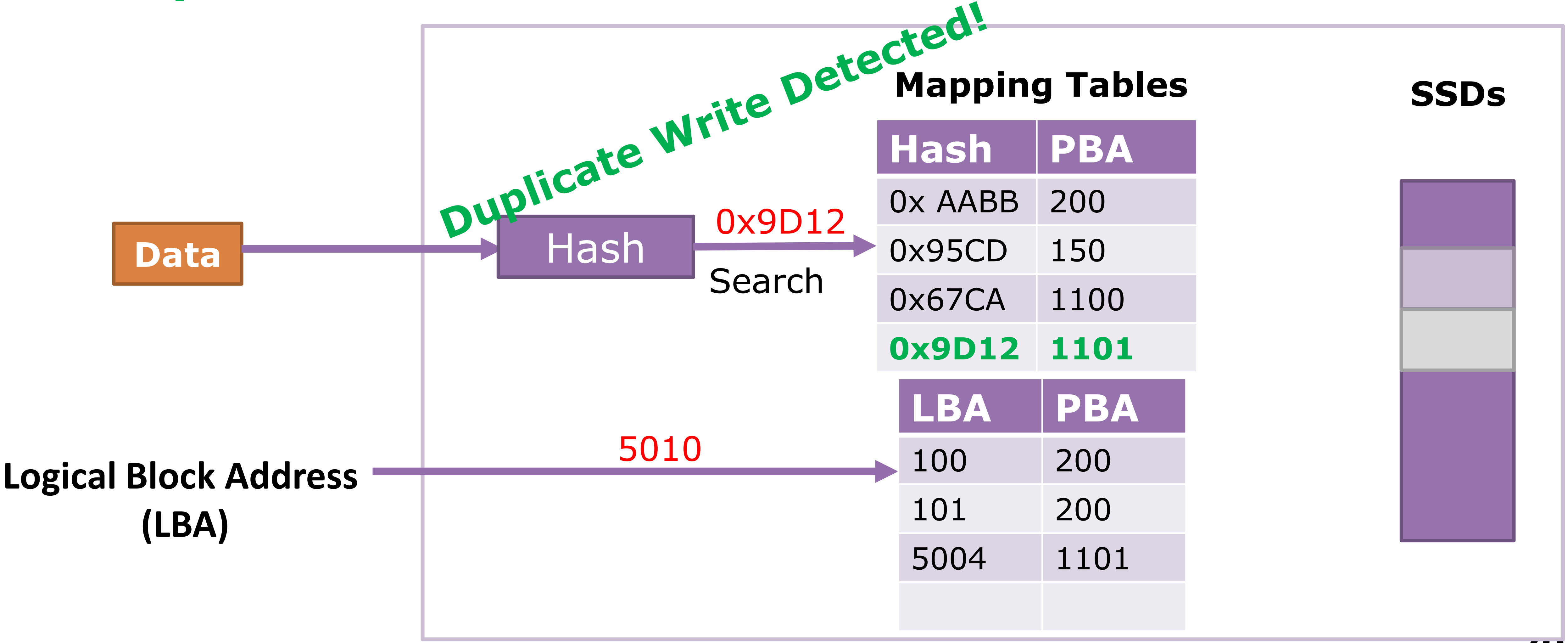


Hash	PBA
0x AABB	200
0x95CD	150
0x67CA	1100
0x9D12	1101

LBA	PBA
100	200
101	200
5004	1101

Data Deduplication Basic Flow

➢ **Duplicate data write**

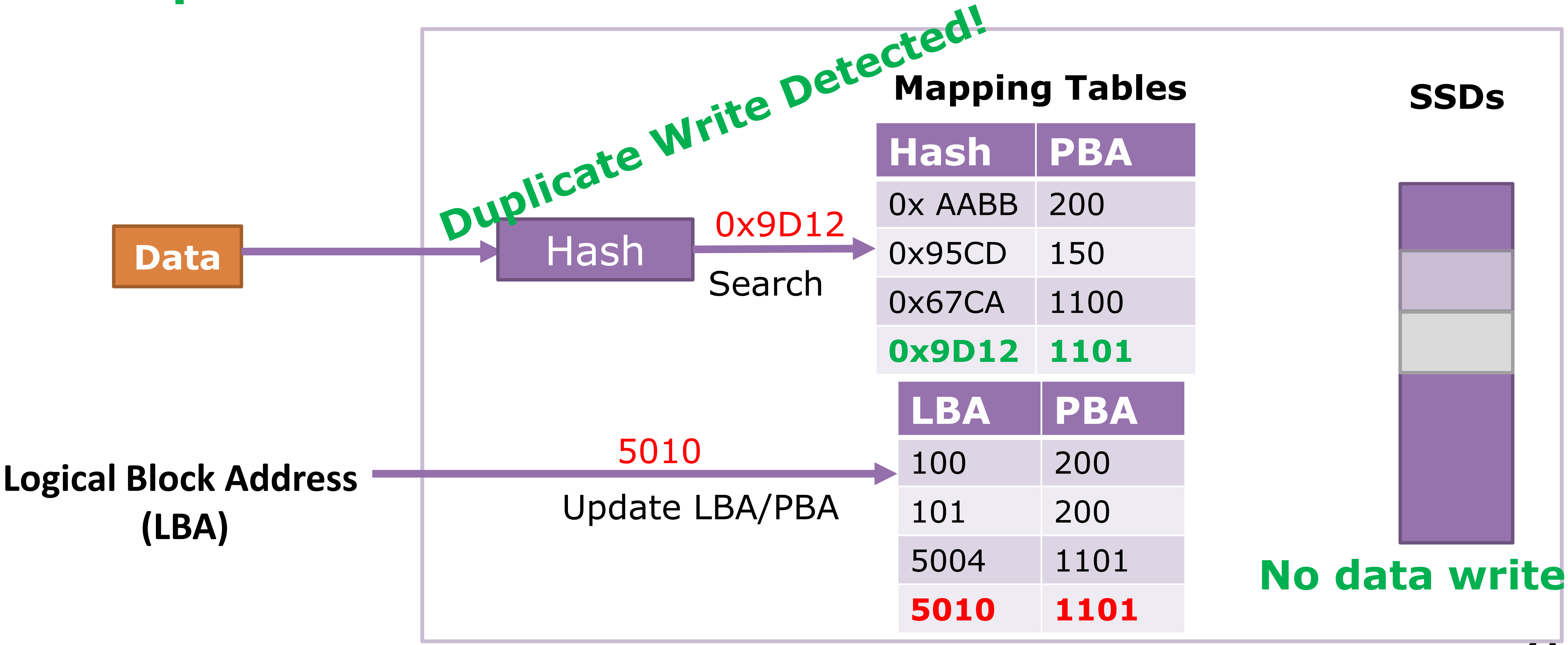


Hash	PBA
0x AABB	200
0x95CD	150
0x67CA	1100
0x9D12	**1101**

LBA	PBA
100	200
101	200
5004	1101

Data Deduplication Basic Flow

- Duplicate data write



Hash	PBA
0x AABB	200
0x95CD	150
0x67CA	1100
0x9D12	**1101**

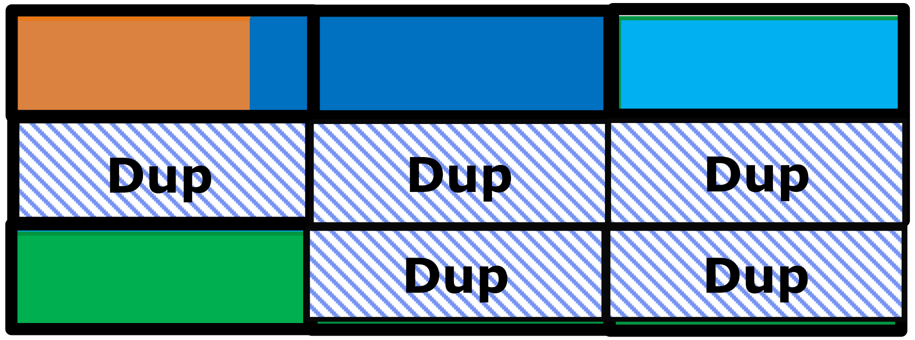
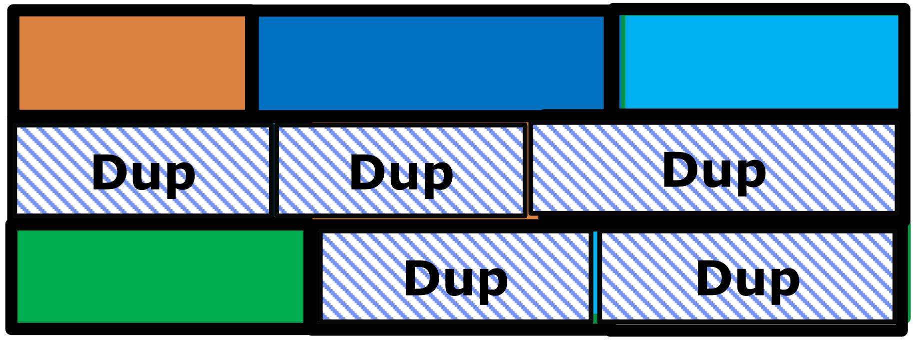
LBA	PBA
100	200
101	200
5004	1101
5010	**1101**

Data Reduction Main Parameters

- **Many parameters & design choices**
  - Granularity, hashing type, mapping table type, compression type, where/when to apply, dedup-compression or compression-dedup, how to reclaim unused spaces, …
- **Various trade-offs**
  - data reduction effectiveness, system resource utilization, latency, throughput, power consumption, …

Next few slides = 4 major parameters discussed

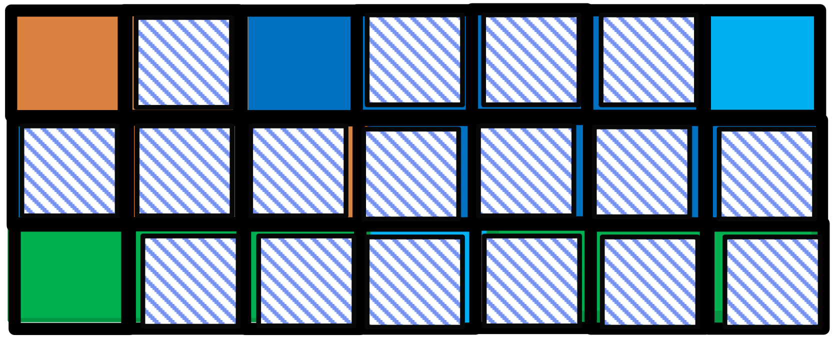
Parameter #1: Chunking Type

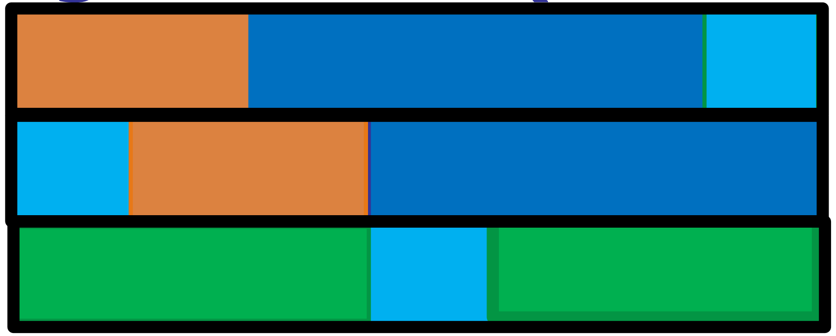
	Fixed sized	Variable sized
Data		
Pros/ Cons	+ Simple, easy to organize - sensitive to data alignment	+ sometimes detects more duplicates - Compute-intensive and complex
Commercial Usage	- Solidfire servers - HPE 3PAR servers	- PureStorage servers - Microsoft Clouds [ATC'12]

Parameter #2: Chunking Granularity

Small Chunks (1KB..8KB) | **Large Chunks (64KB..4MB)**

Data





Pros/ Cons

Small chunks:
+ High duplicate detection
- Heavy-weight mapping tables

Large chunks:
+ Lightweight mapping tables
- Less duplicates & RMW overheads

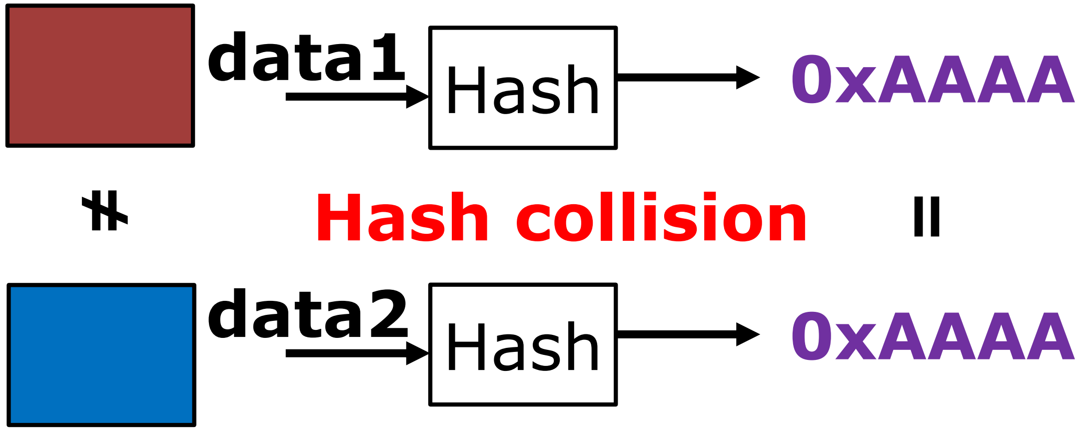
Commercial Usage

Small chunks:
- Solidfire servers (4 KB)
- HPE 3PAR servers (16 KB)

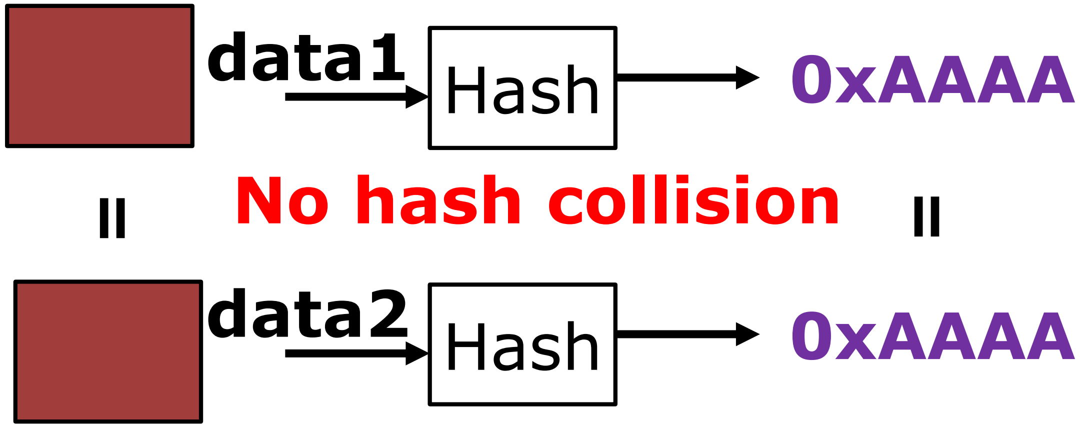
Large chunks:
- Some Microsoft Clouds (64 KB)

Parameter #3: Hashing Algorithm

Weak Hash (e.g., CRC)



Strong Hash (e.g., SHA2)



Pros/Cons

Weak Hash (e.g., CRC):

+ Fast calculation
- Hash collision =data loss! (needs bit-by-bit data comparison)

Strong Hash (e.g., SHA2):

+ No practical hash collision in PBs
- Compute-intensive

Commercial Usage

Weak Hash (e.g., CRC):

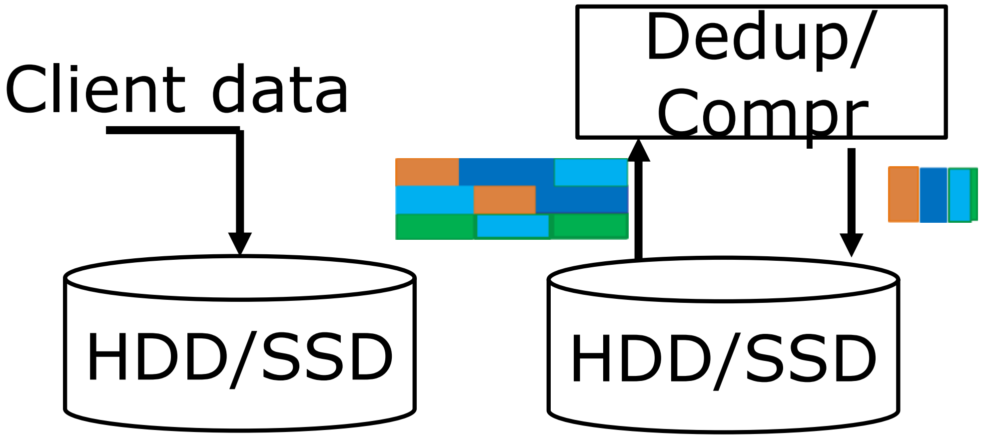
- PureStorage servers

Strong Hash (e.g., SHA2):

- Solidfire (SHA2 hash)
- Microsoft clouds (SHA1 hash)

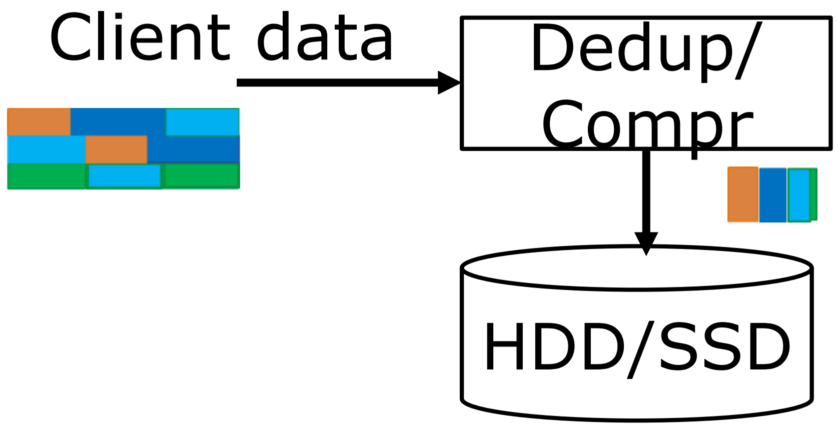
Parameter #4: When to Do Data Reduction

Offline Operation



Active time **Idle time**

Inline Operation



Active time

Pros/ Cons

Offline:
- \+ No impact on active IOs
- \- Requires idle time
- \- Reduces SSD lifetime

Inline:
- \+ Improves SSD lifetime
- \+ No idle time required
- \- Requires dedicated resources (CPU,…)

Commercial Usage

- Offline: HDD-based systems
- Inline: Most SSD-based systems

Data Reduction Main Parameters

➢ Our Choices

- **Inline data reduction** → Best for SSD array
- **Fixed sized chunking** → lightweight operation
- **64 KB to 4 KB chunking** → toward most effectiveness
- **SHA2 strong hashing** → no practical collision in PBs

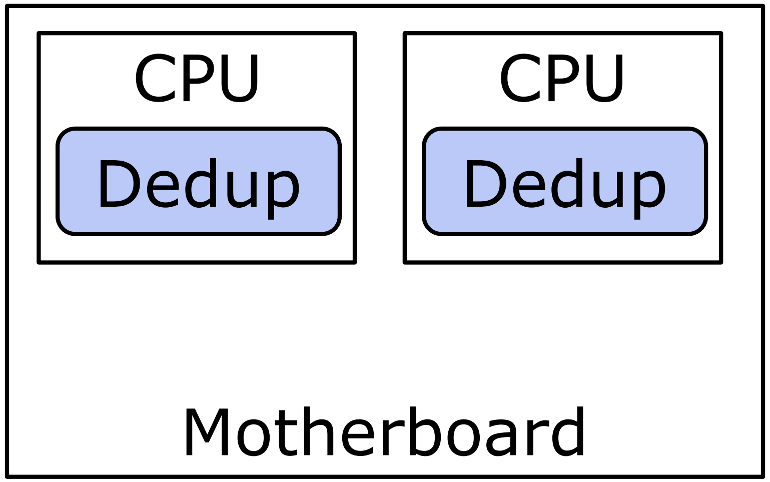
Overview of My Data Reduction Research

- **Maximize scalability of data reduction**
  - **Data reduction capability↑ Supported capacity ↑**
  - **Data reduction throughput↑ Overheads↓**
- **Deduplication for slow SSDs (CAL'17)**
  - SATA SSDs, <5 GB/s & <10 TB capacity, limited workloads
- **Deduplication and compression for fast SSDs (HPCA'19)**
  - PCIe SSDs, 10-100GB/s &100s TB capacity, limited workloads
- **Ultrascalability & workload support (MICRO'19)**
  - PCIe SSDs, 100> GB/s & 100s TB capacity, more workloads

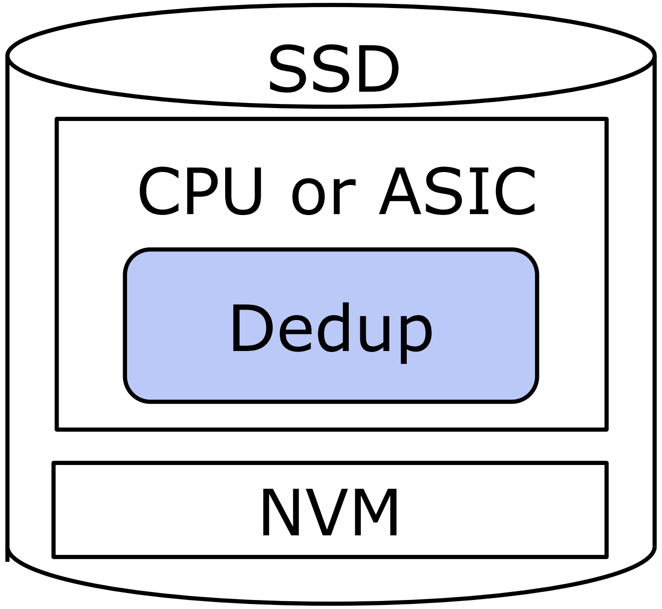
Index

- **Background**
  - Storage Systems and Trends
  - Basics of Data Reduction Techniques
- **Proposing New Data Reduction Architecture**
  - **Deduplication for slow SSD Arrays**
  - Deduplication and Compression for fast SSD Arrays
  - Optimizing for Ultra-scalability & more Workload Support
- **Conclusion**

Deduplication Approaches for SATA SSDs



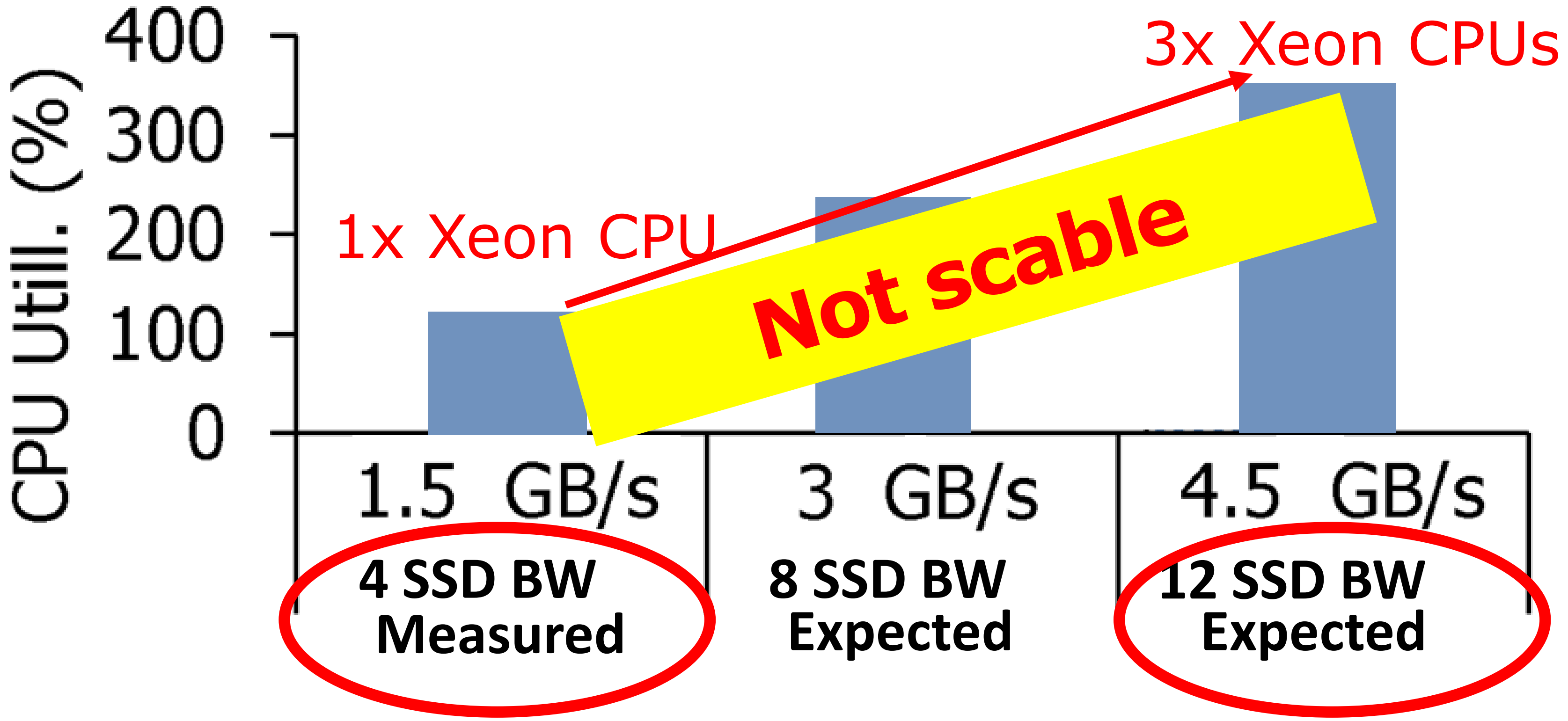
SW-based



Intra-SSD

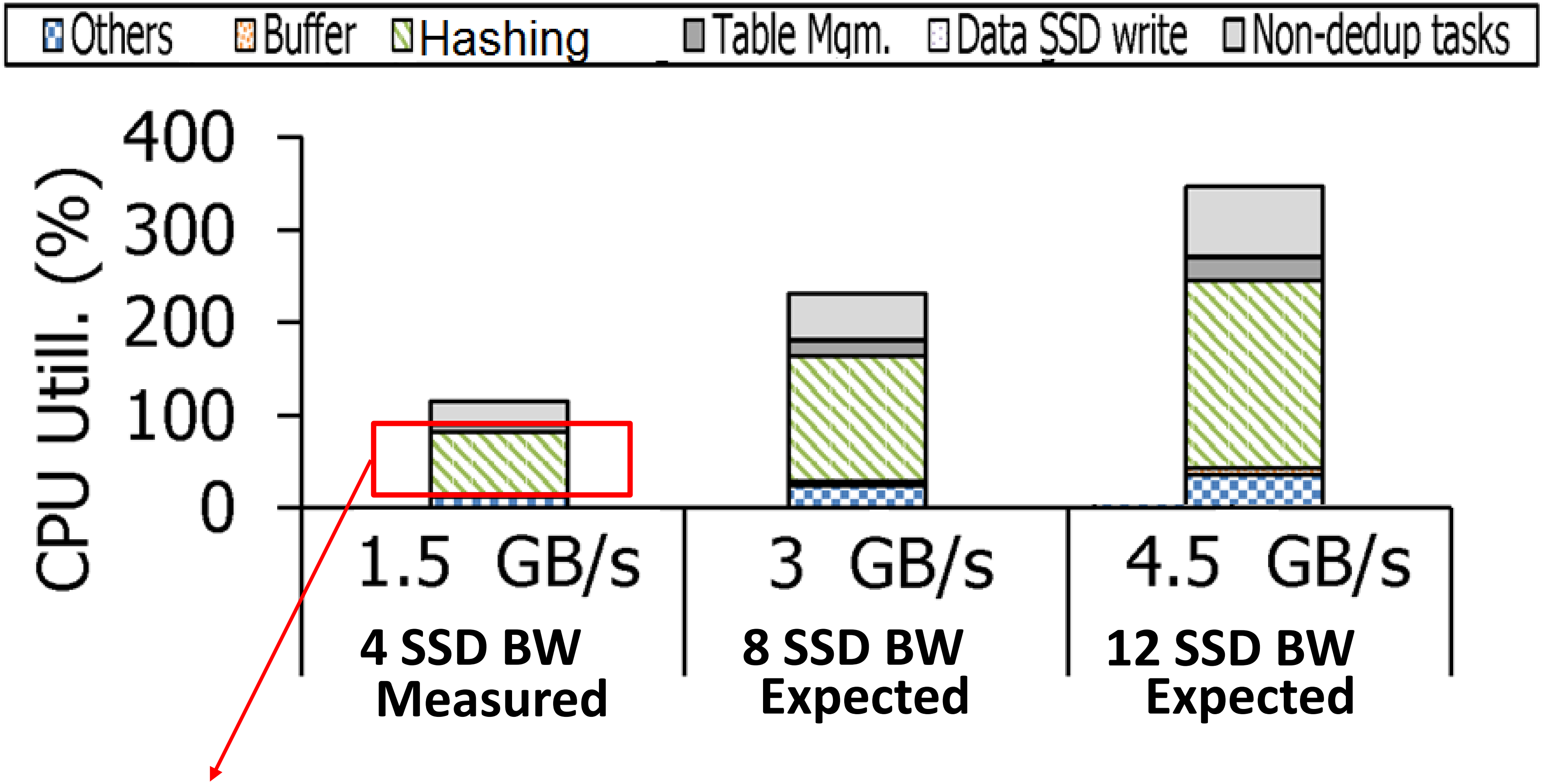
HW acceleration

1. SW-based Dedup: CPU Utilization



Excessive CPU utilization in deduplication

1. SW-based Dedup: CPU Utilization



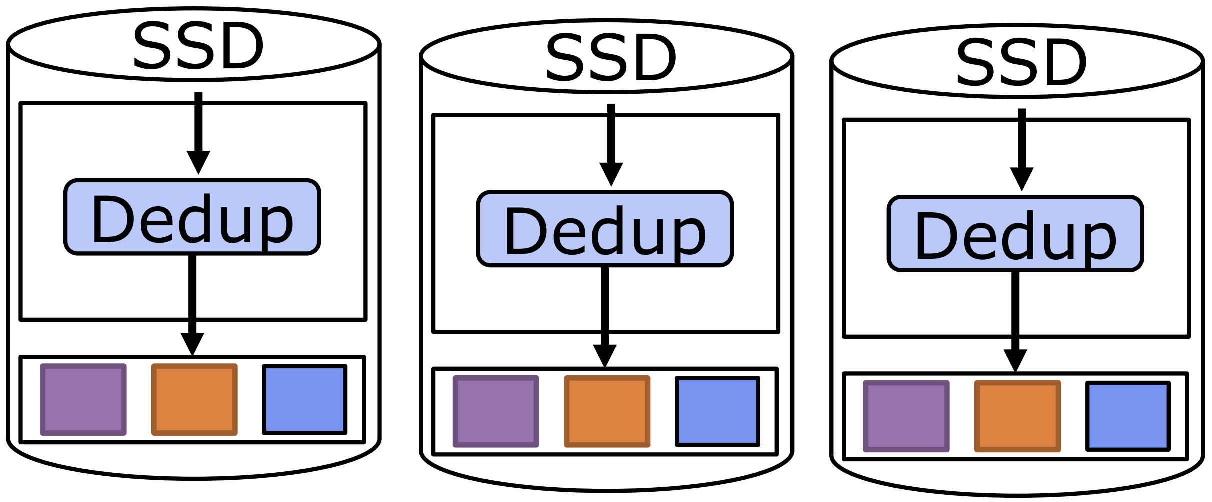
Hashing + Metadata management = 90% of CPU Util.

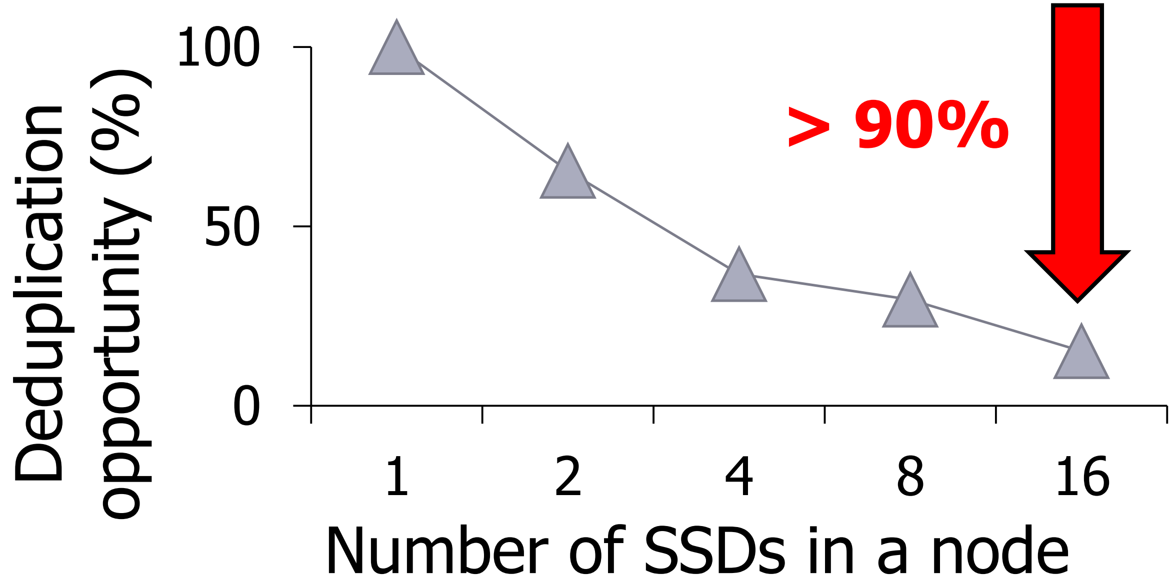
2. Intra-SSD Deduplication

- **Use embedded CPU or ASIC in SSD [FAST'11, MSST'12]**

(-) Low data reduction due to no inter-SSD deduplication

- Decentralized metadata management





Cannot detect duplicates in multiple SSDs!

Our Solution for Scalable Deduplication

1	Throughput scalability	→ Offload to HW (hash & metadata mgm)
2	Minimize Chip Power	→ Use FPGA or ASIC (not GPU)
3	High Dedup Capability	→ Centralize metadata mgm

Our Solution for Scalable Deduplication

2 M Ch

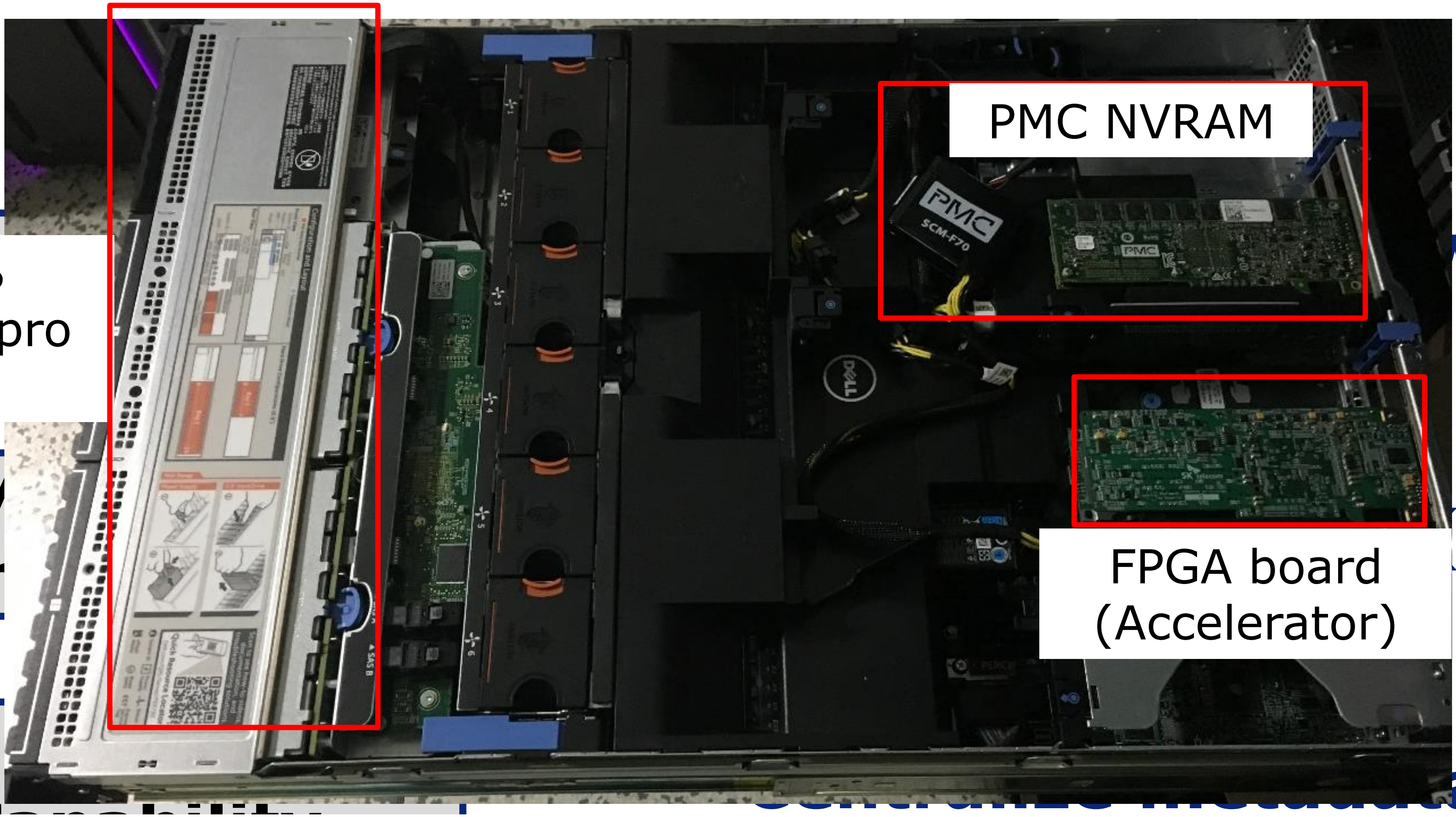
3 Hi

W

mgm)

(not GPU)

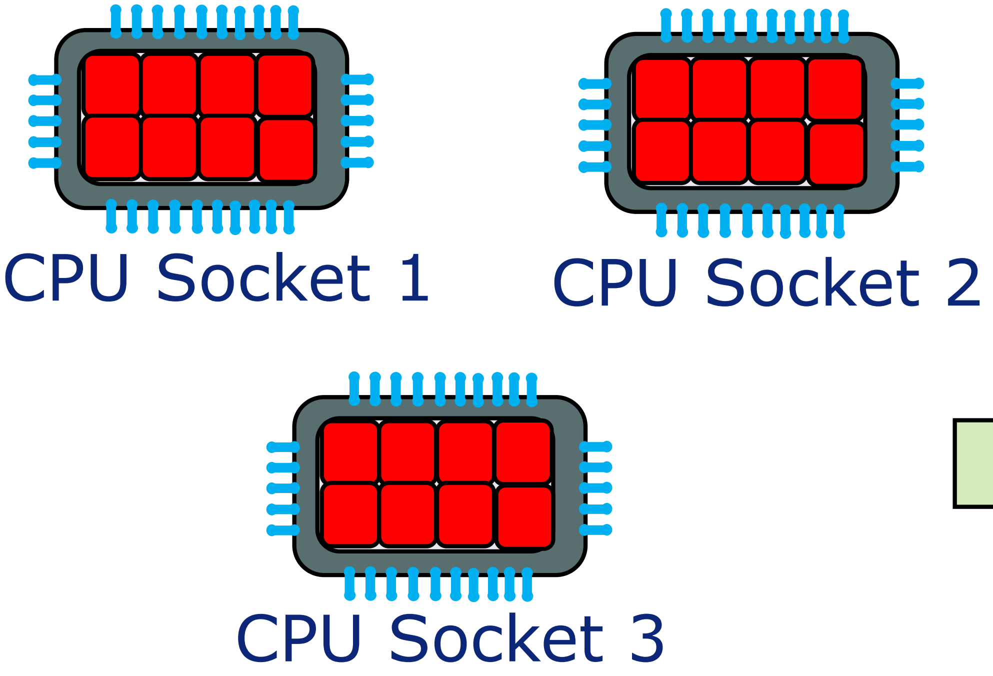
a mgm



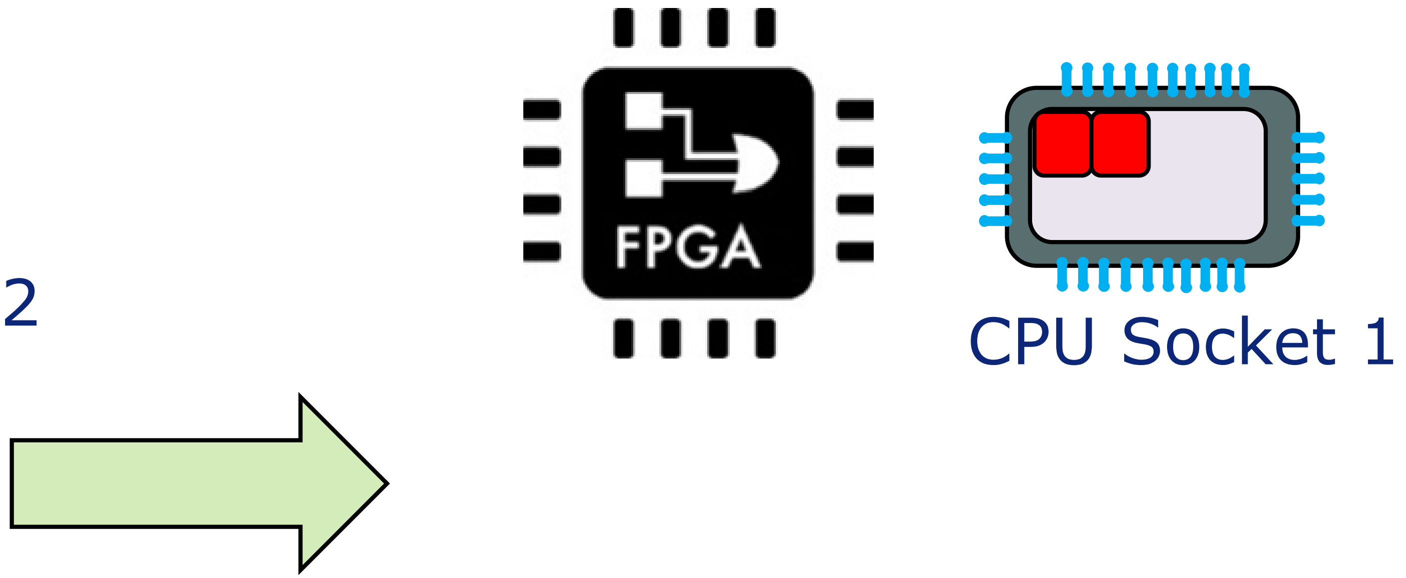
Prototype on real machine

Evaluation (at 4.5 GB/s)

Baseline



Our Proposed Design



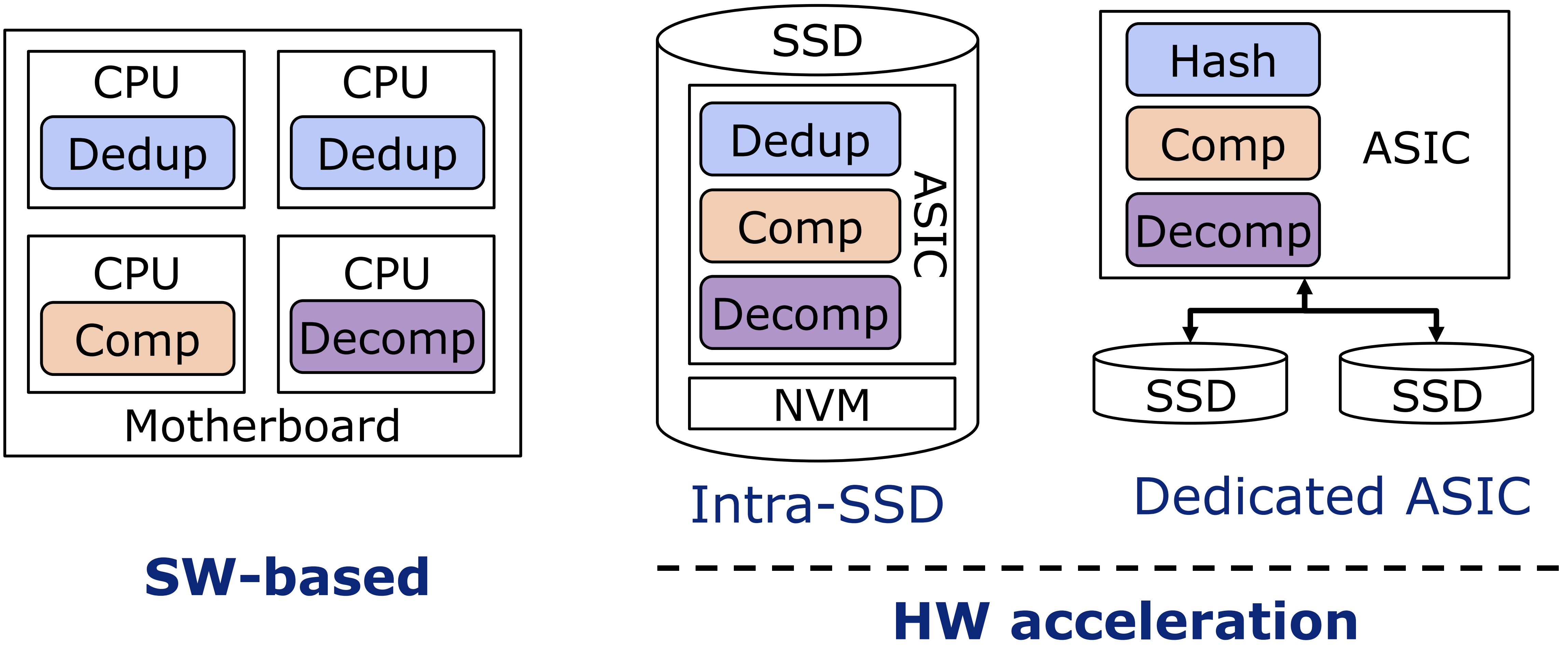
92% less CPU utilization

40% Less chip power consumption

Index

- **Background**
  - Storage Systems and Trends
  - Basics of Data Reduction Techniques
- **Proposing New Data Reduction Architecture**
  - Deduplication for slow SSD Arrays
  - **Deduplication and Compression for fast SSD Arrays**
  - Optimizing for Ultra-scalability & more Workload Support
- **Conclusion**

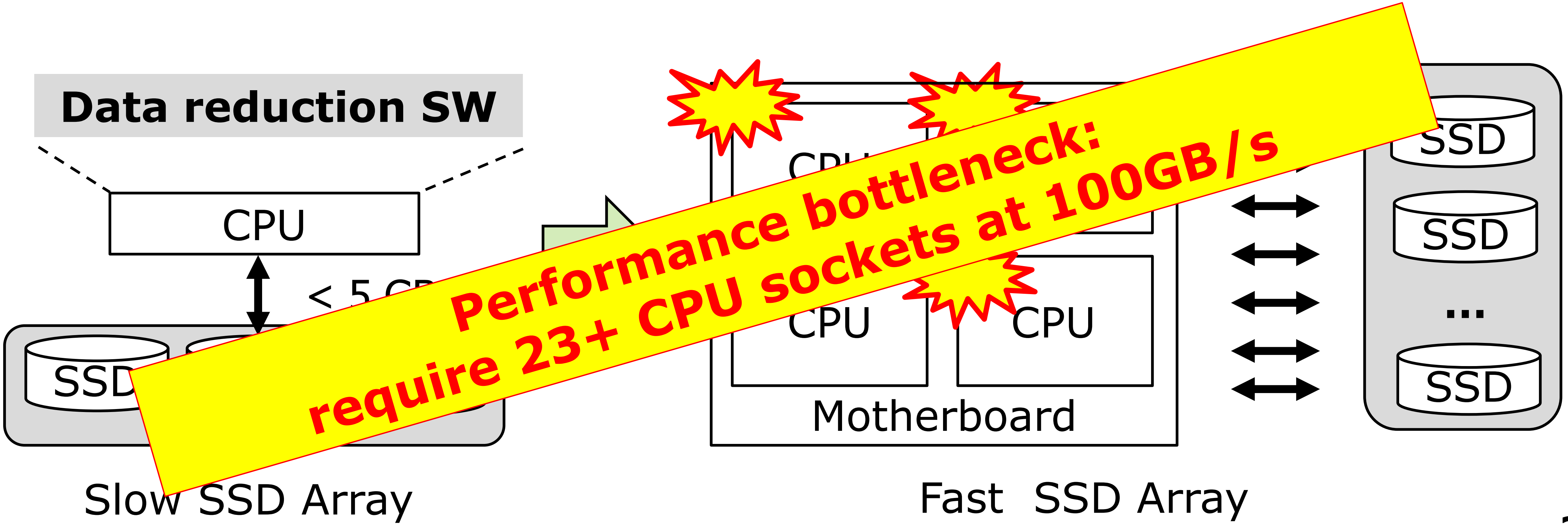
Existing Approaches



1. SW-Based Deduplication & Compression

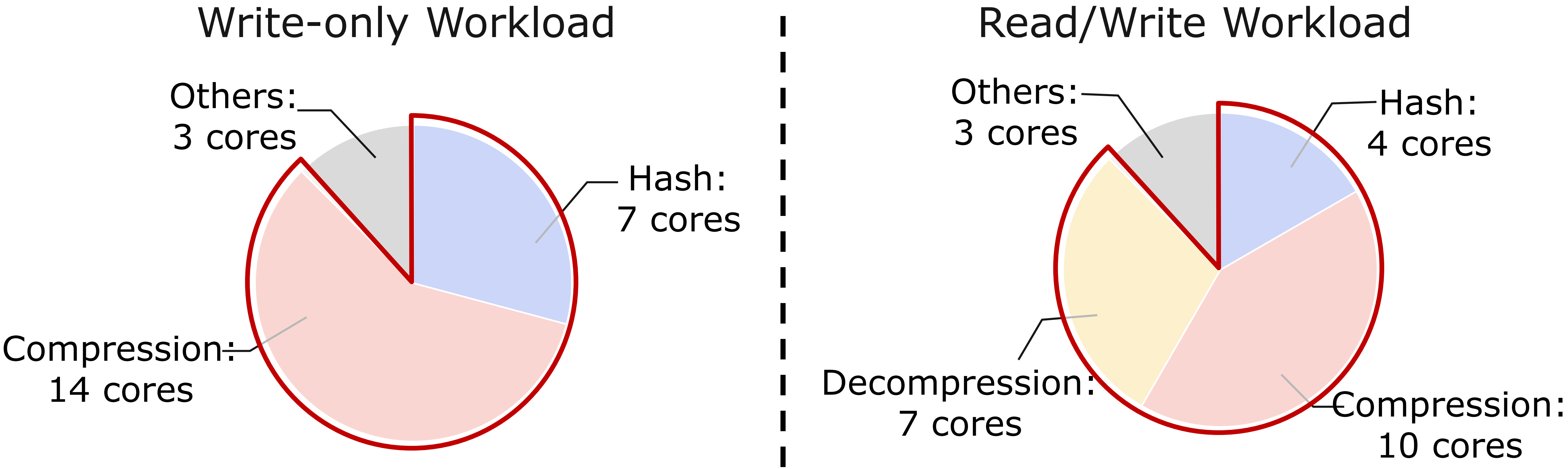
- **Optimized SW (Intel ISA-L) scales for slow SSD array**

(-) Low throughput scalability for a high-end SSD array



Heavy Computations on CPUs

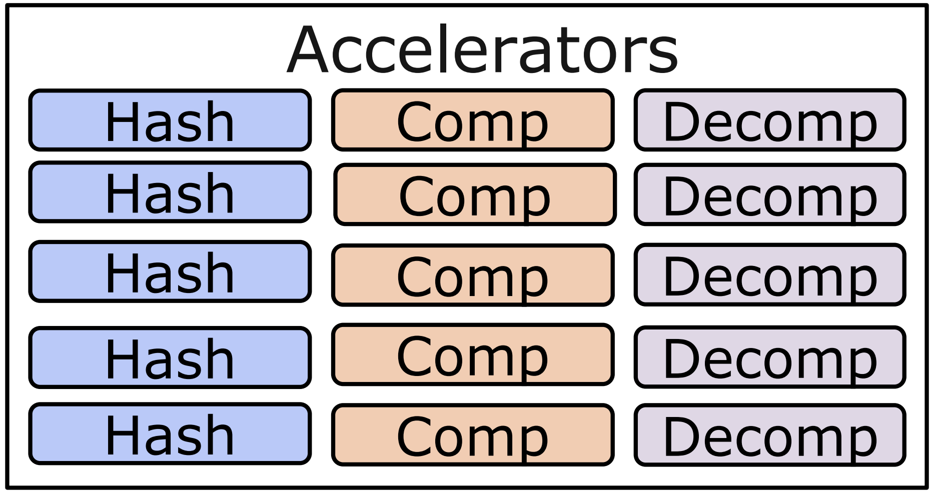
- **Profiled CPU utilization on a 24-core machine**



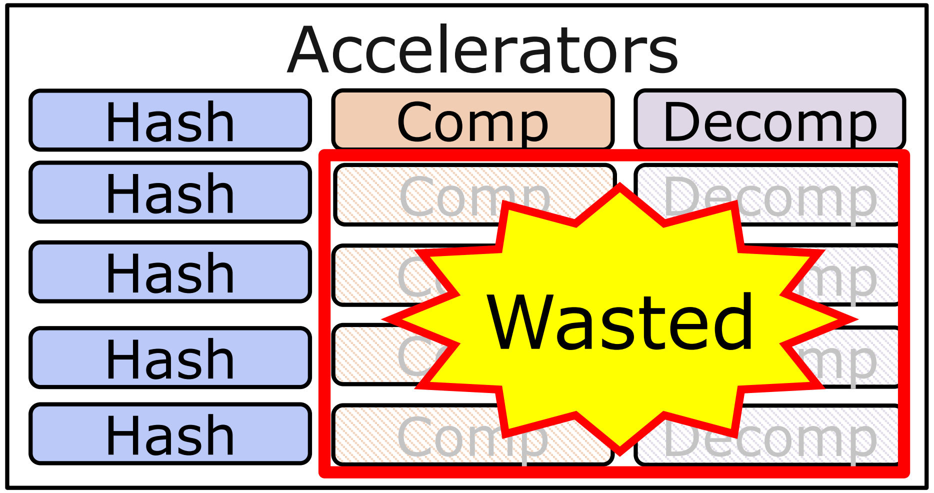
90 % of CPU-intensive operations → hardware acceleration

2. Dedicated HW Acceleration

- **Hardware design is inflexible**
- **Overprovision resources for the worst-case workload**



Overprovisioned design
(worst-case scenarios)



Required Design for example workload
(Write-intensive + many duplicates)

Low device utilization due to fixed provisioning

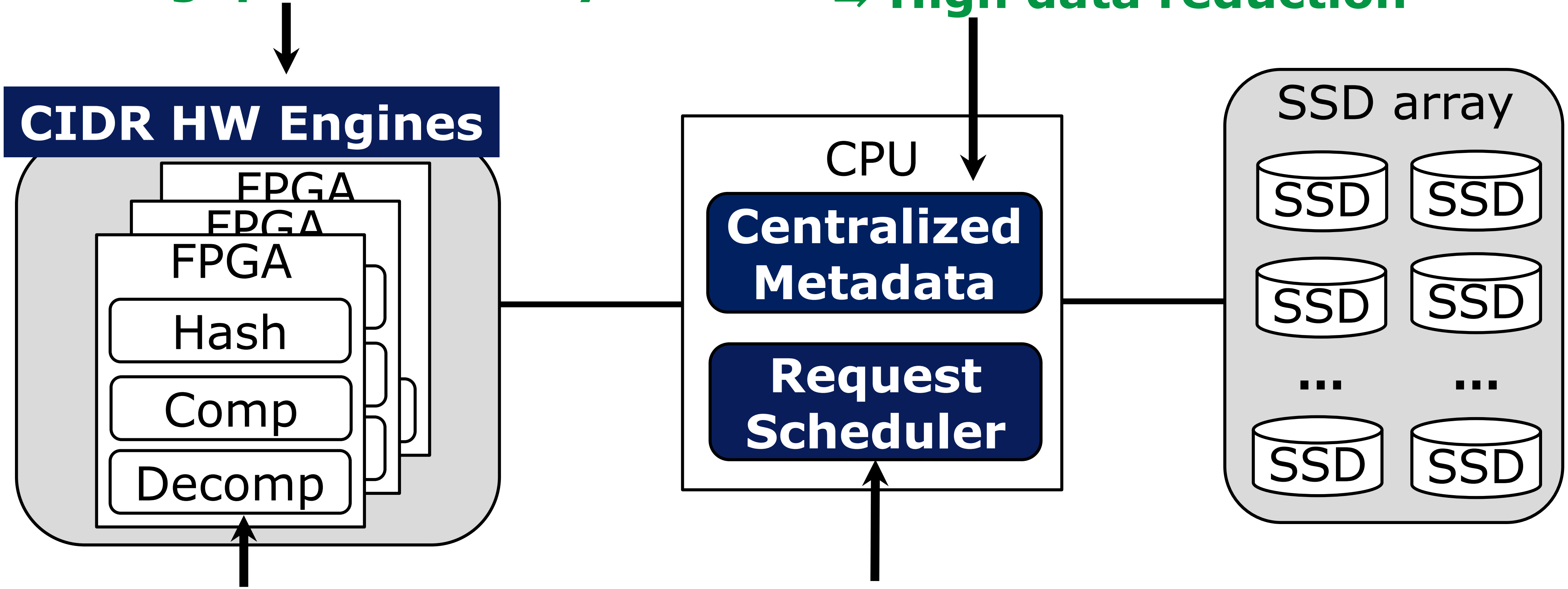
CIDR: Design Goals

	SW	Intra-SSD	Dedicated HW	CIDR
1 Throughput scalability	X	O	O	O
2 High data reduction	O	X	△	O
3 Efficient device utilization	O	X	X	O

CIDR: Key Ideas

1. Scalable FPGA array
⇒ **Throughput scalability**

2. Centralized table management
⇒ **High data reduction**



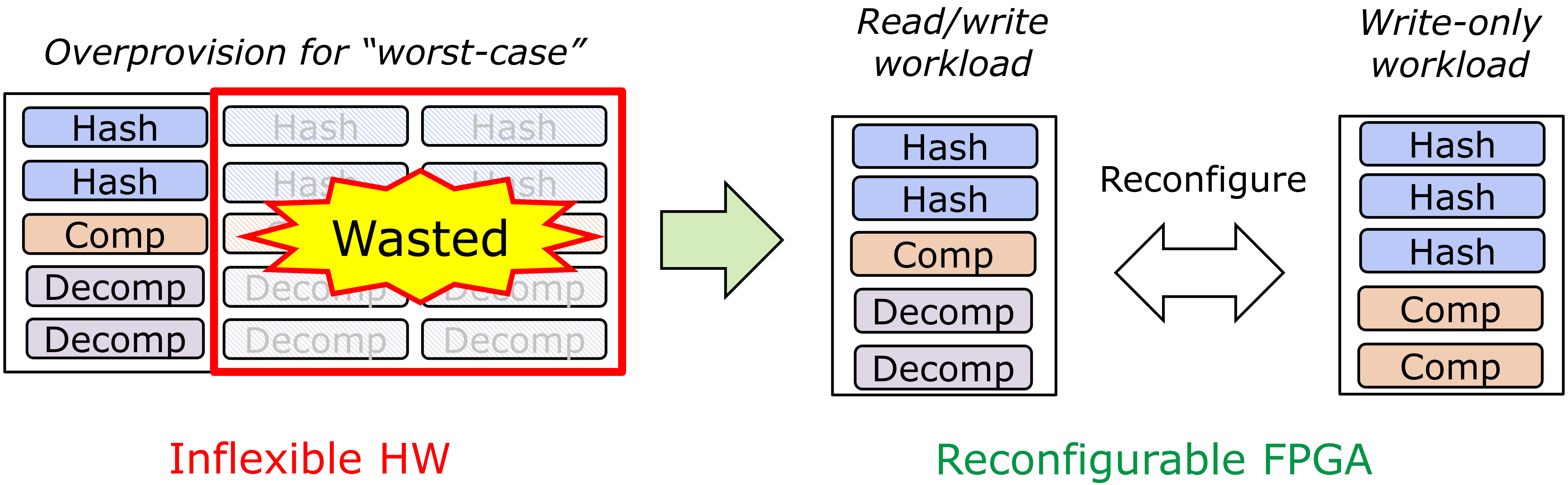
3. Long-term FPGA reconfig

4. Short-term request scheduler

⇒ **Efficient device utilization**

Key Idea #3: Long-Term FPGA Reconfig

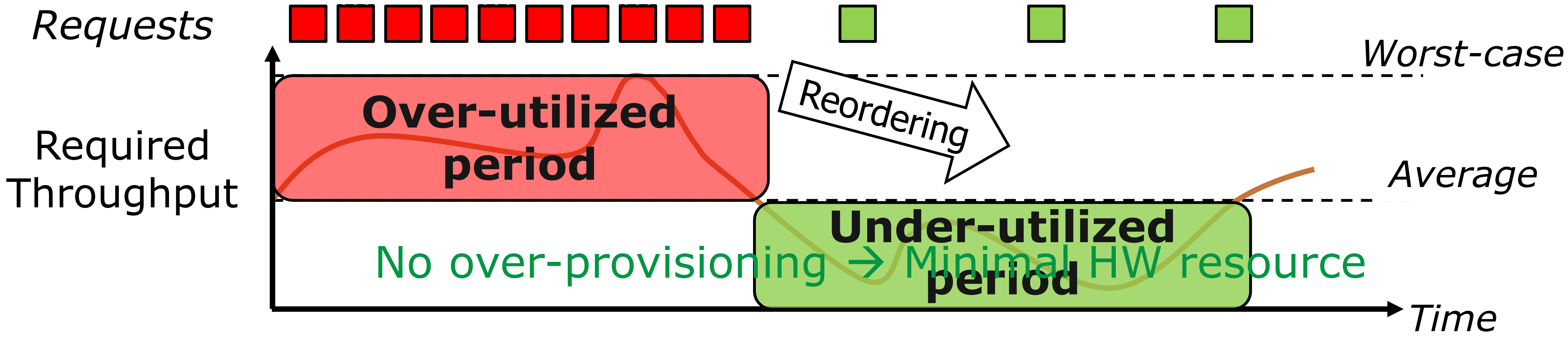
- **Reconfigure FPGAs to workload's average behavior**



"Minimal HW resources" with reconfigurable FPGAs!

Key Idea #4: Short-term Request Scheduler

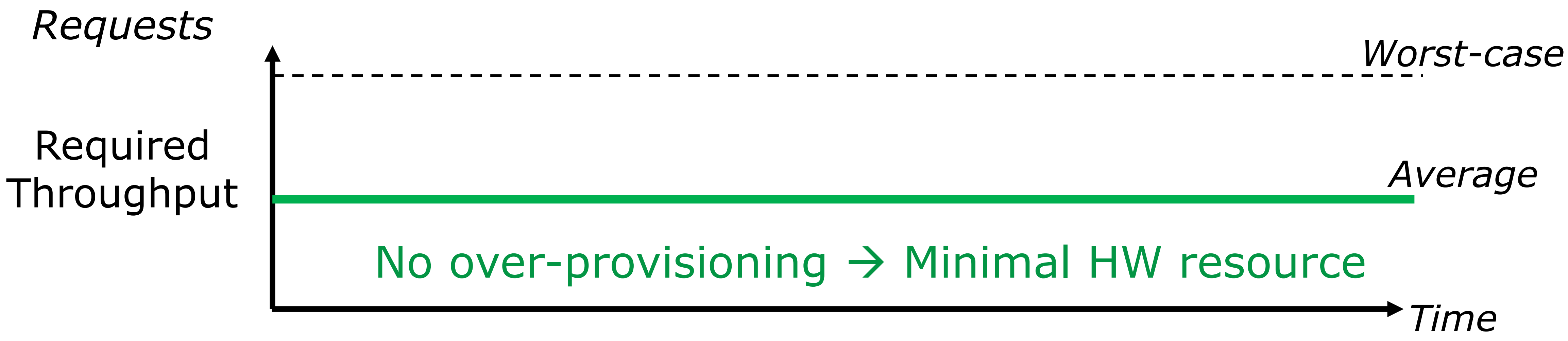
- **Schedule requests considering available HW resources**
  - Shift the load of over-utilization period to under-utilization period



"High resource utilization" with smart request scheduling!

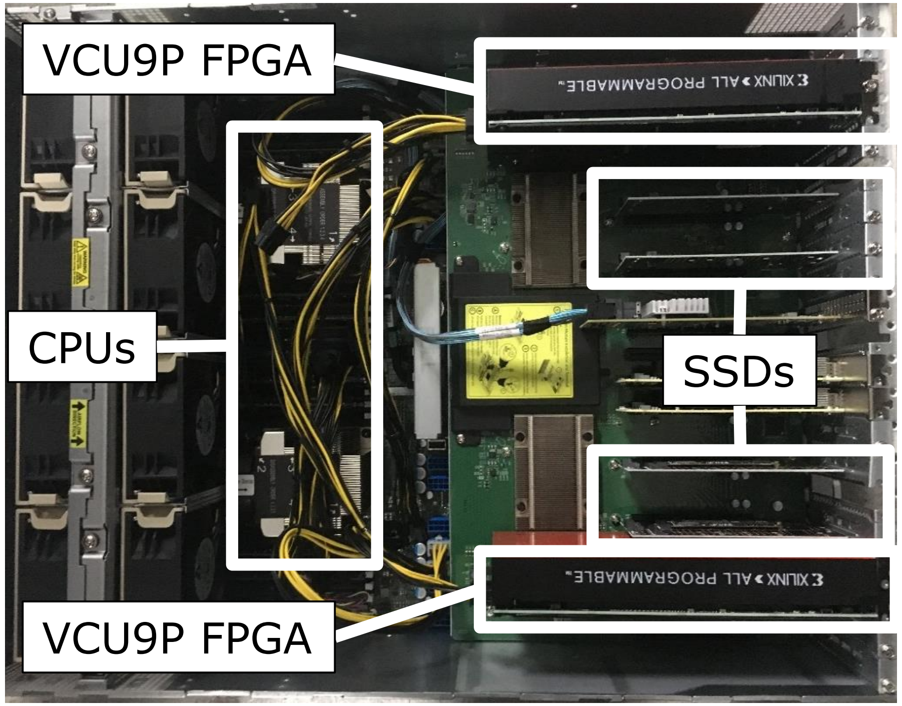
Key Idea #4: Short-term Request Scheduler

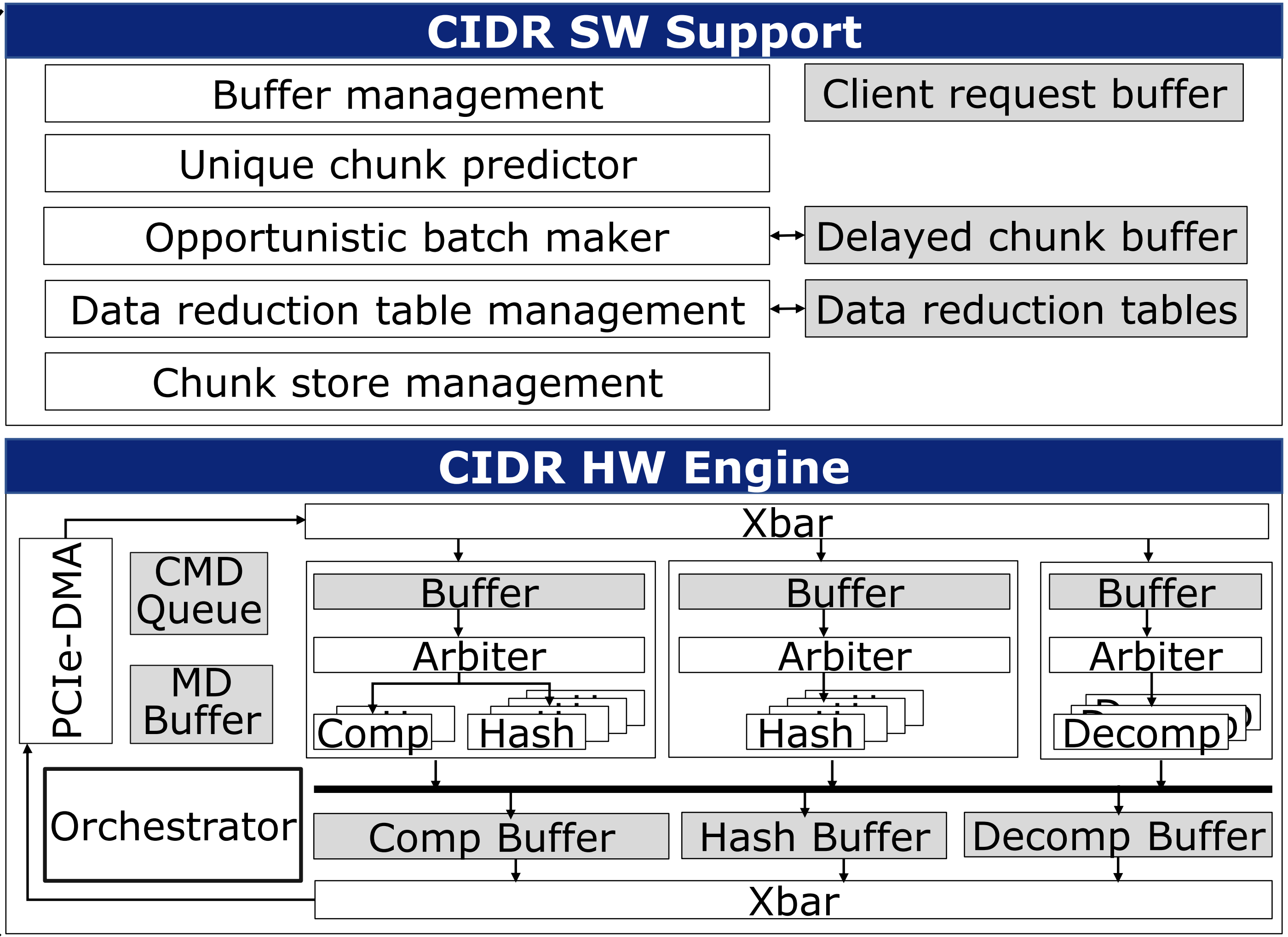
- **Schedule requests considering available HW resources**
  - Shift the load of over-utilization period to under-utilization period



"High resource utilization" with smart request scheduling!

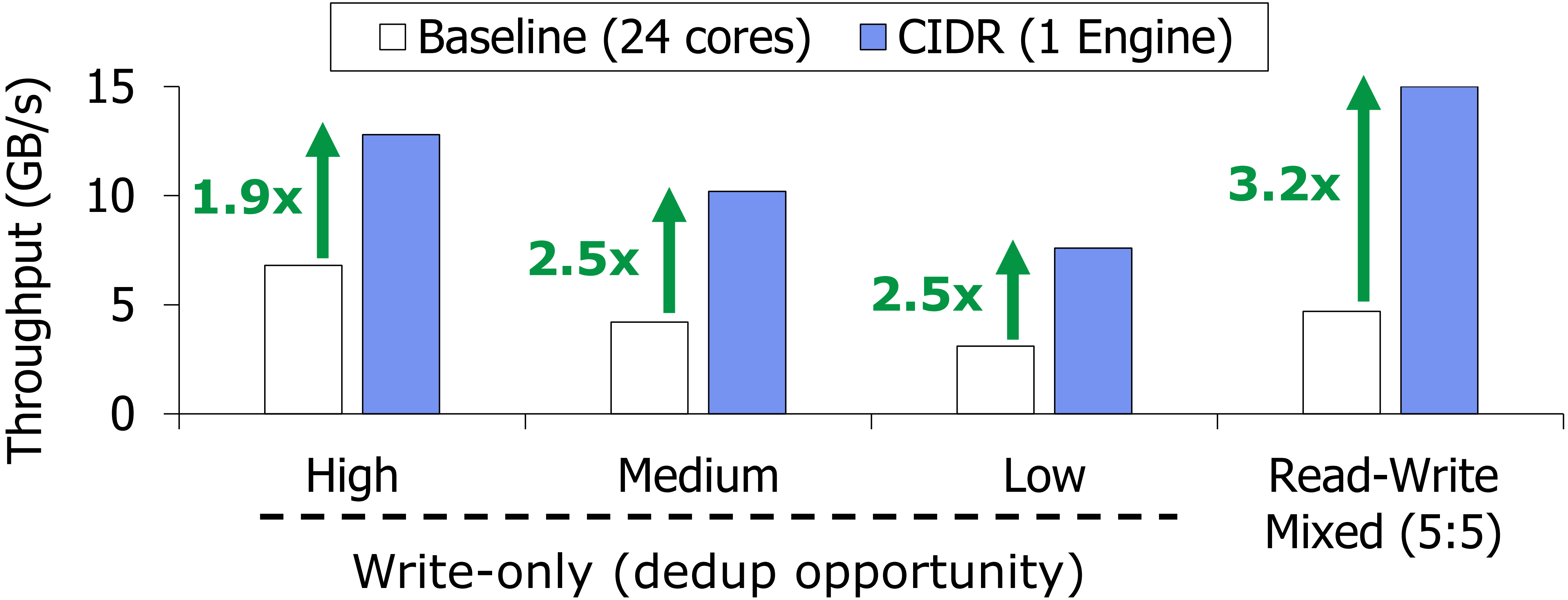
CIDR: Detailed System Architecture





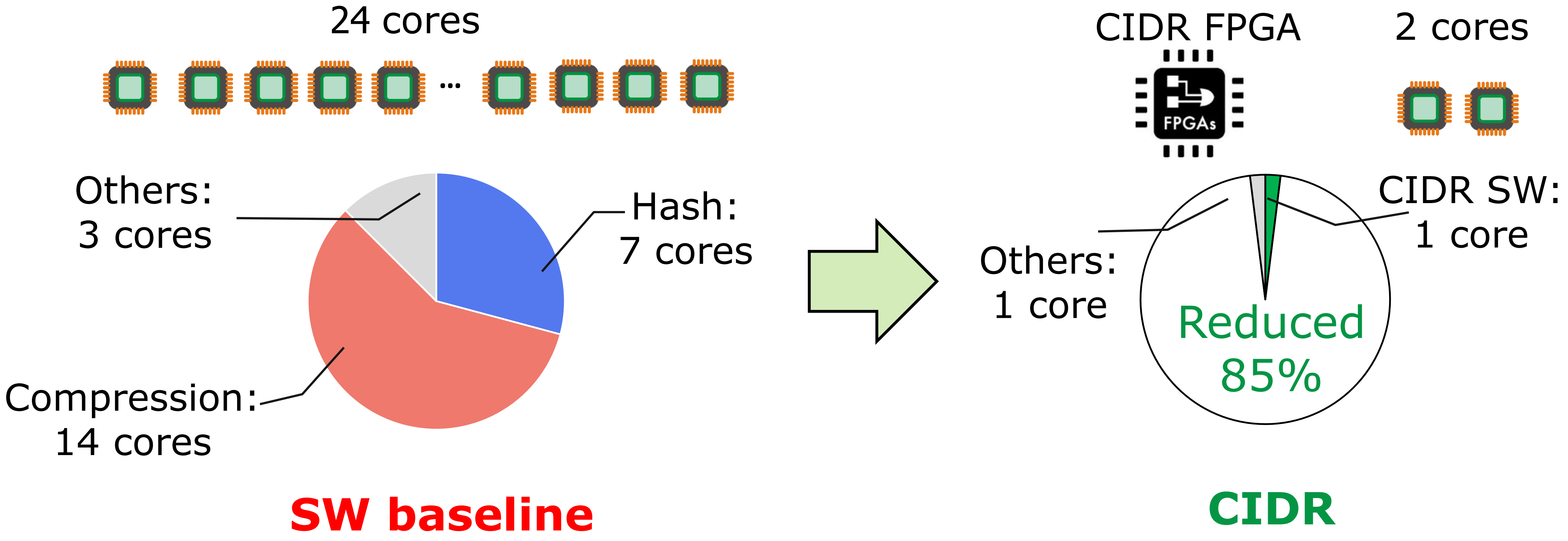
CIDR's High Throughput (Single FPGA)

- **Hardware acceleration with HW/SW optimizations**



CIDR's Low CPU Utilization

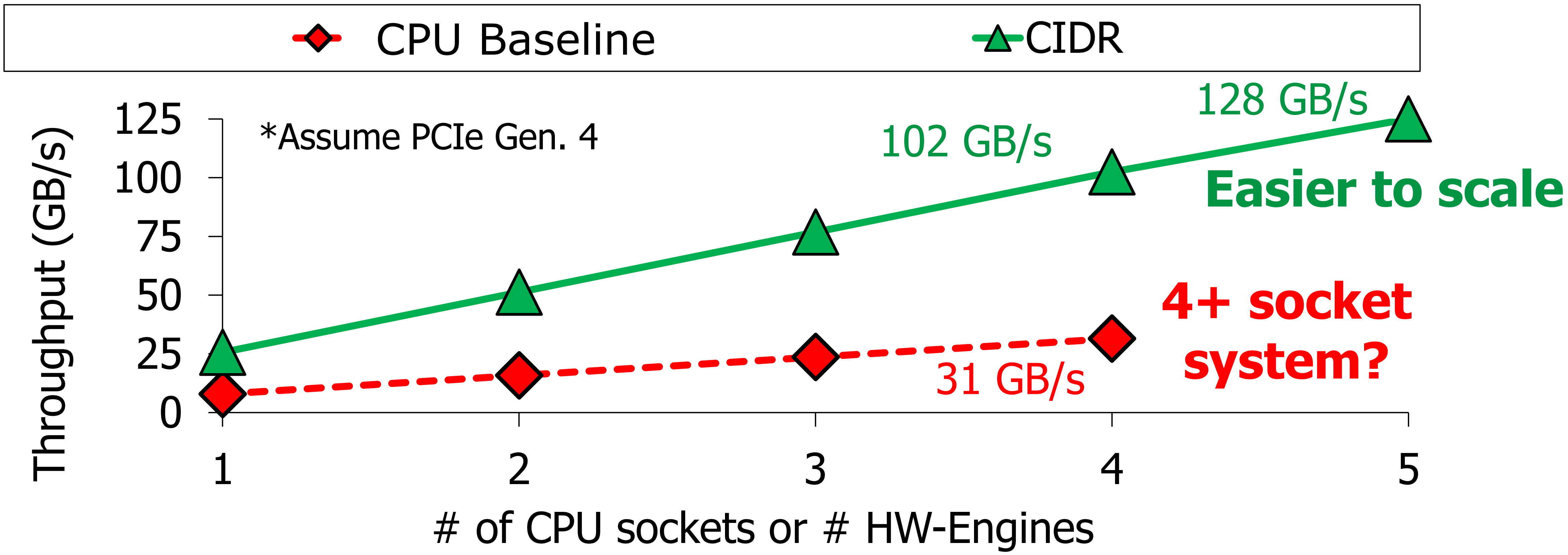
- **Comparison at the same throughput**



Enables extreme throughput scalability

CIDR's High Throughput Scalability

- **Scalable FPGA array for higher throughput**



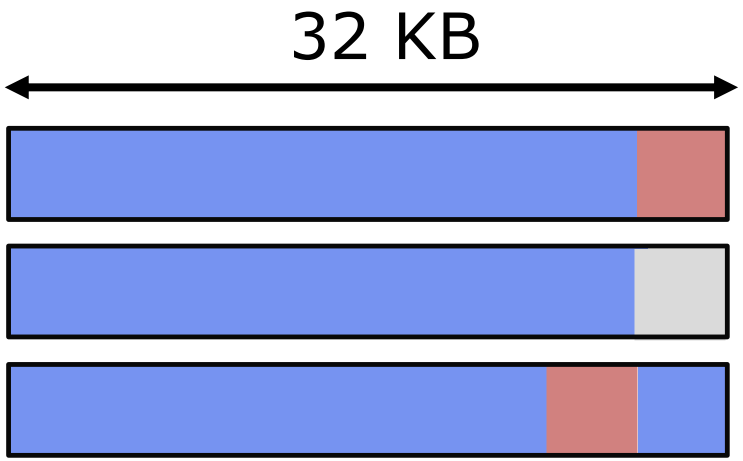
Index

- **Background**
  - Storage Systems and Trends
  - Basics of Data Reduction Techniques
- **Proposing New Data Reduction Architecture**
  - Deduplication for slow SSD Arrays
  - Deduplication and Compression for fast SSD Arrays
  - **Optimizing for Ultra-scalability & Workload Support**
- **Conclusion**

Why Small Chunking?

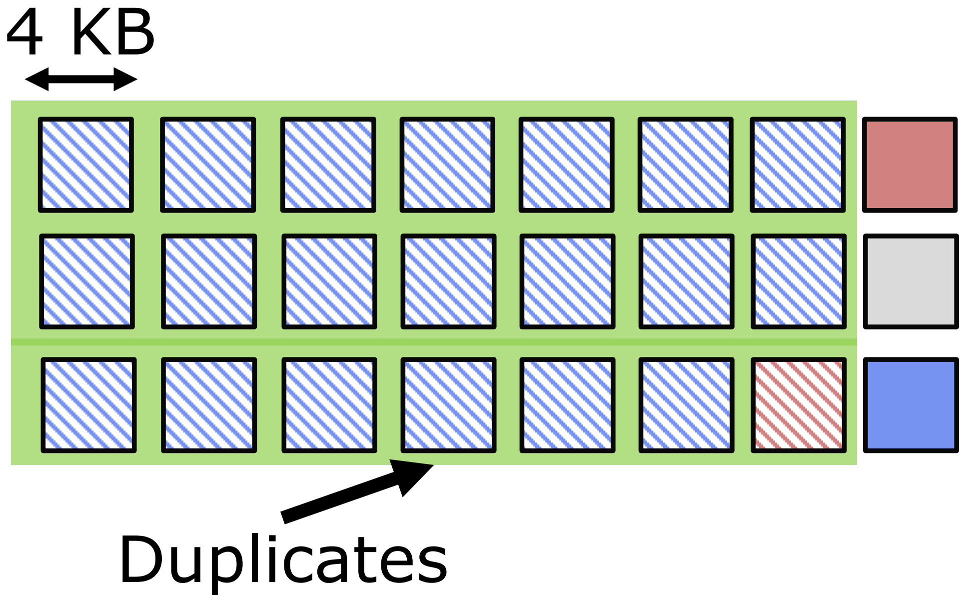
- **Small chunking can detect more duplicates**

Large chunking (CIDR)



(-) Small # of duplicates
(-) High RMW overheads
(17x IO overhead in FIU traces)

Small chunking

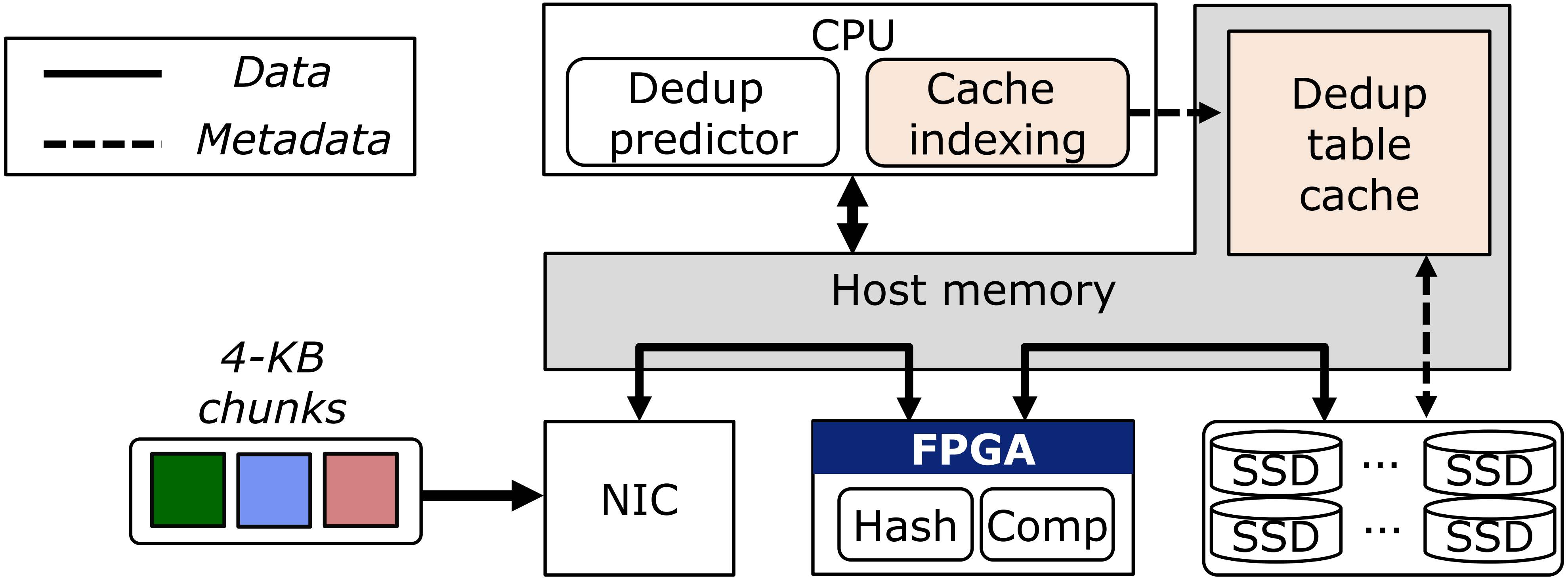


(+) Large # of duplicates
(+) Supports more workloads

Increase the cost-effectiveness of storage servers

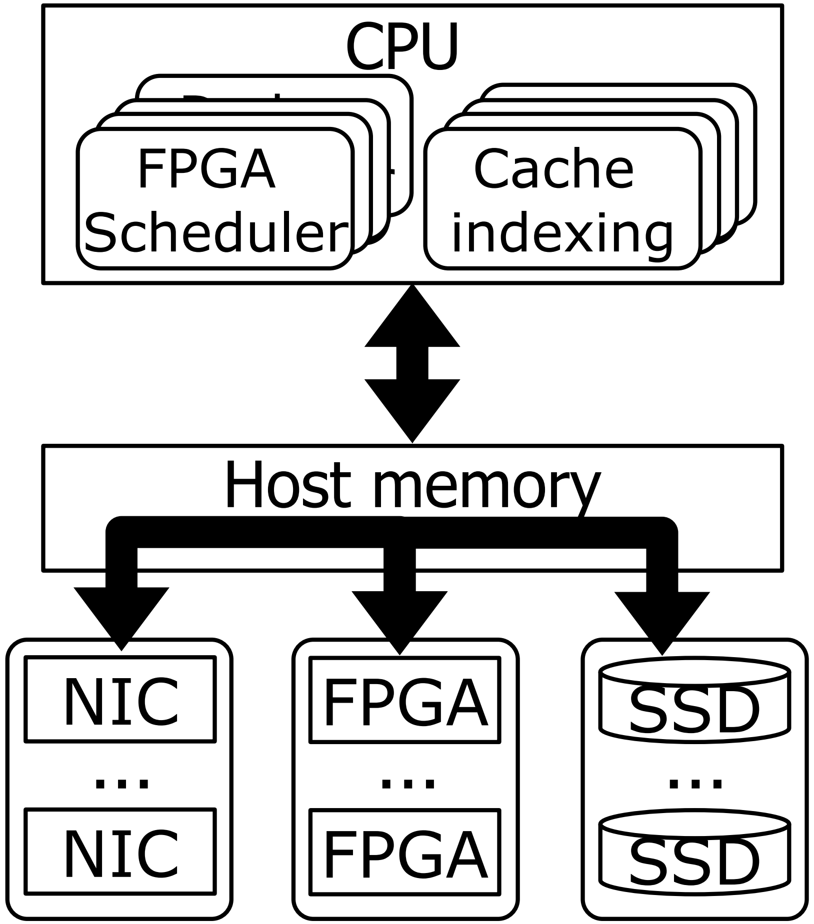
CIDR+: As the New Baseline

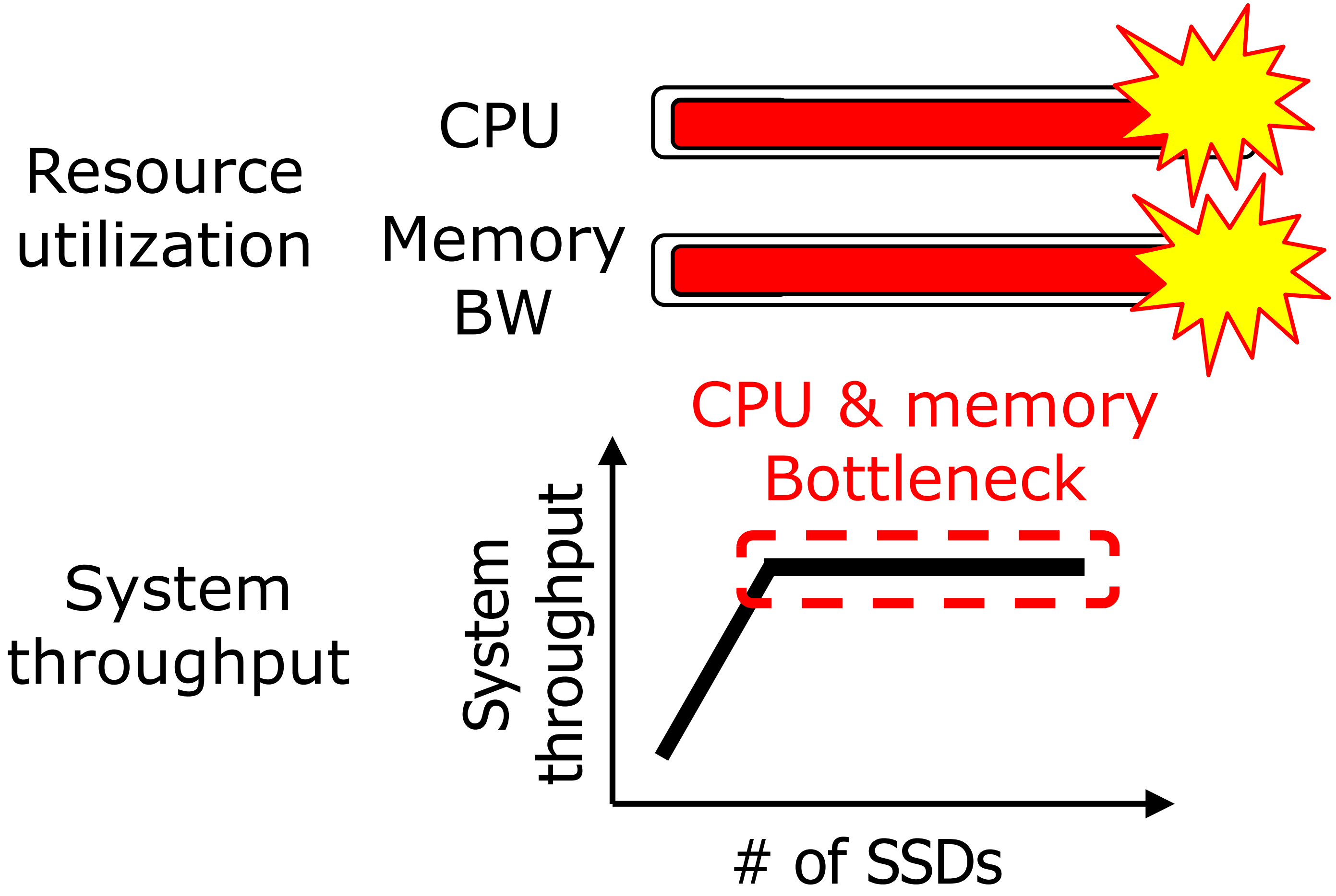
- **CIDR with dedup table cache to support small chunking**



Now, we can analyze the performance bottleneck of small-chunking data reduction!

Limited Scalability of Baseline

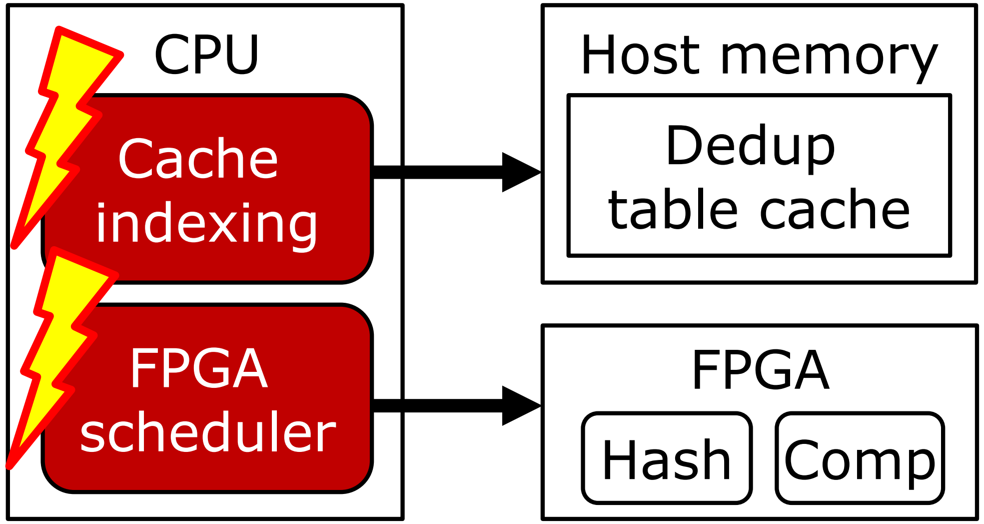


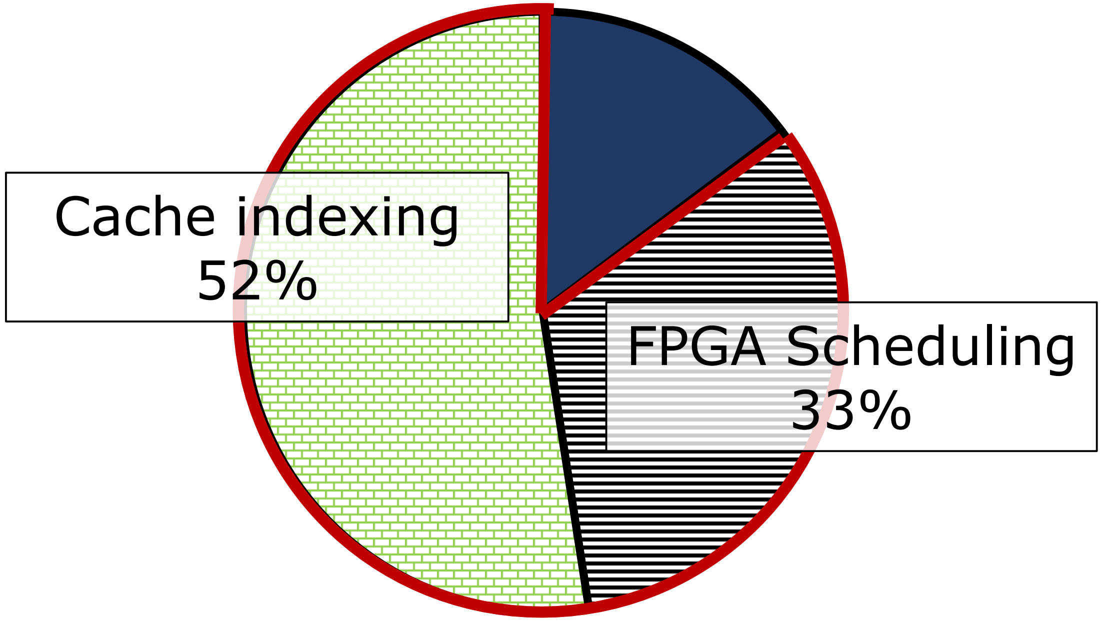


System throughput saturates due to high memory and CPU overhead

Why is "CPU" the Bottleneck?

- **Higher throughput → more indexing & FPGA scheduling**



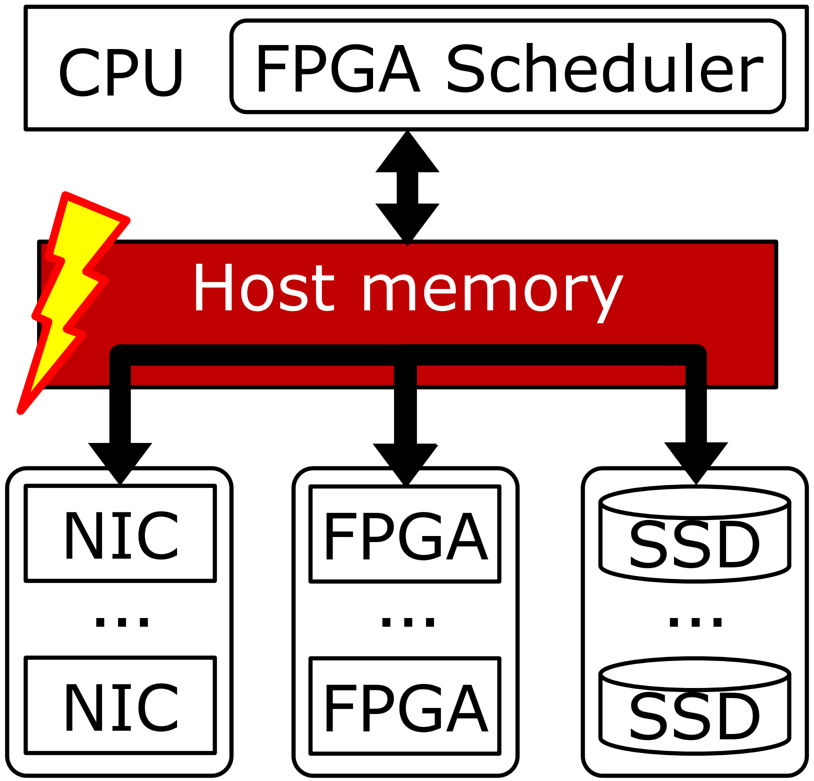


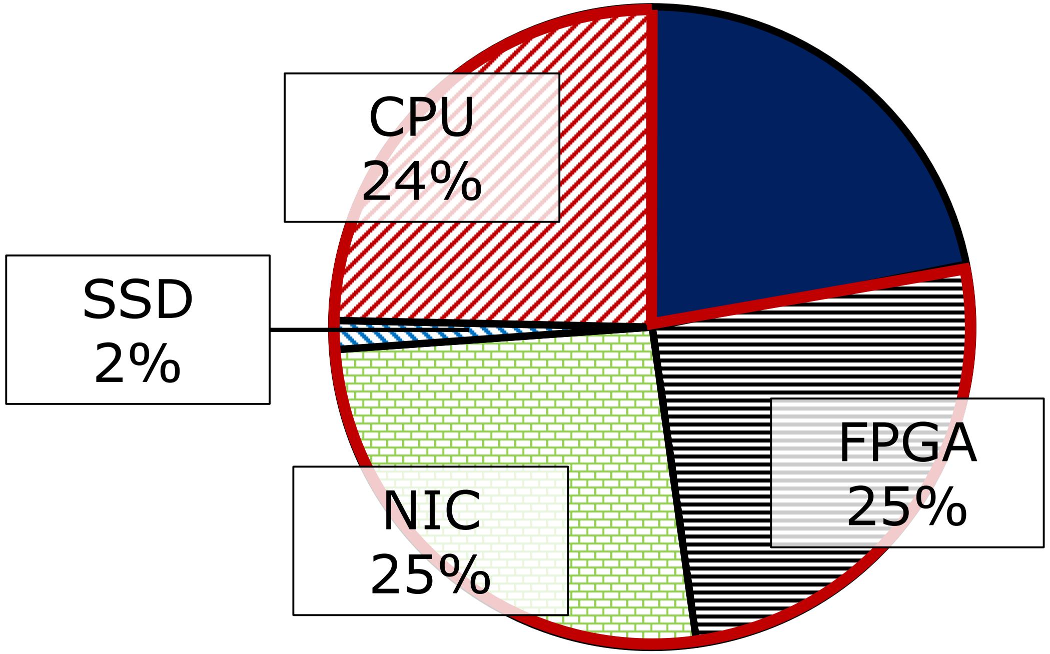
CPU utilization

At scale, the two operations take many CPU cycles!

Why is "Memory" the Bottleneck?

- **Higher throughput → higher rate of data movements**

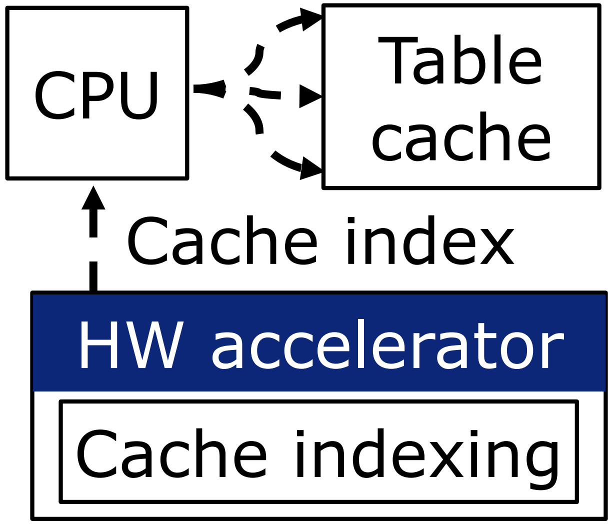




Memory BW utilization

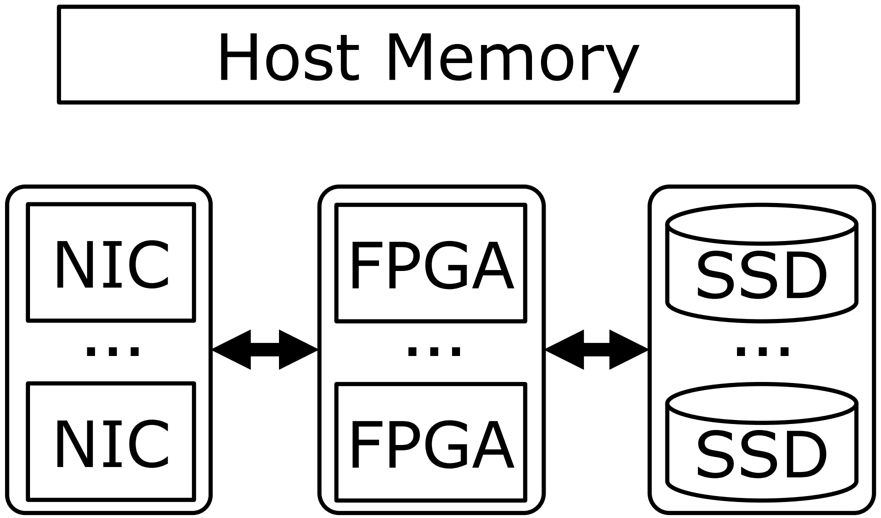
Data movements consume most memory BW!

Three Key Ideas of FIDR



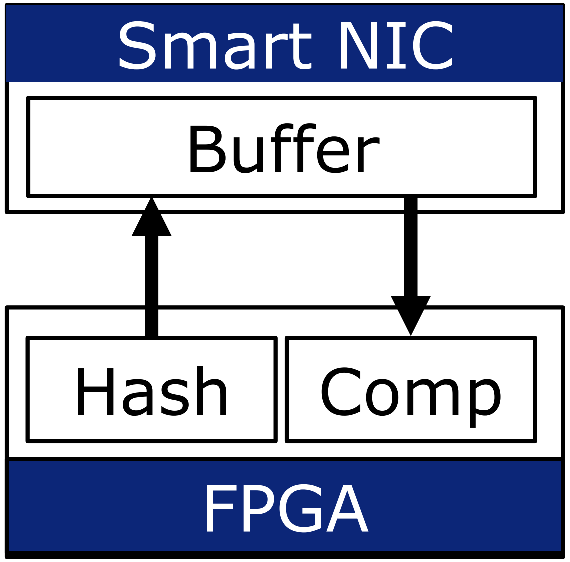
1. Cache indexing acceleration

(+) Reduced CPU overhead



2. Direct D2D communication

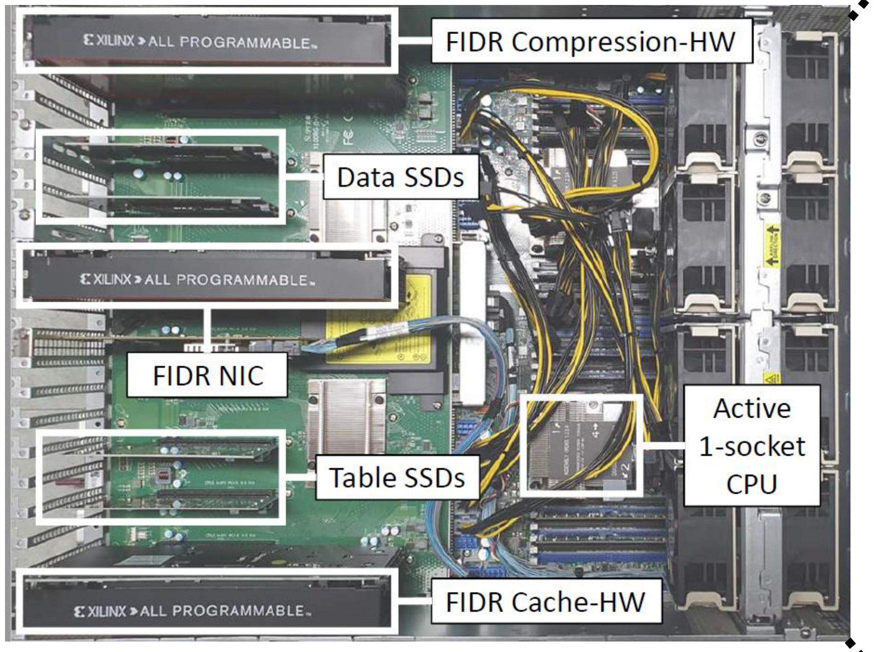
(+) Minimal memory pressure

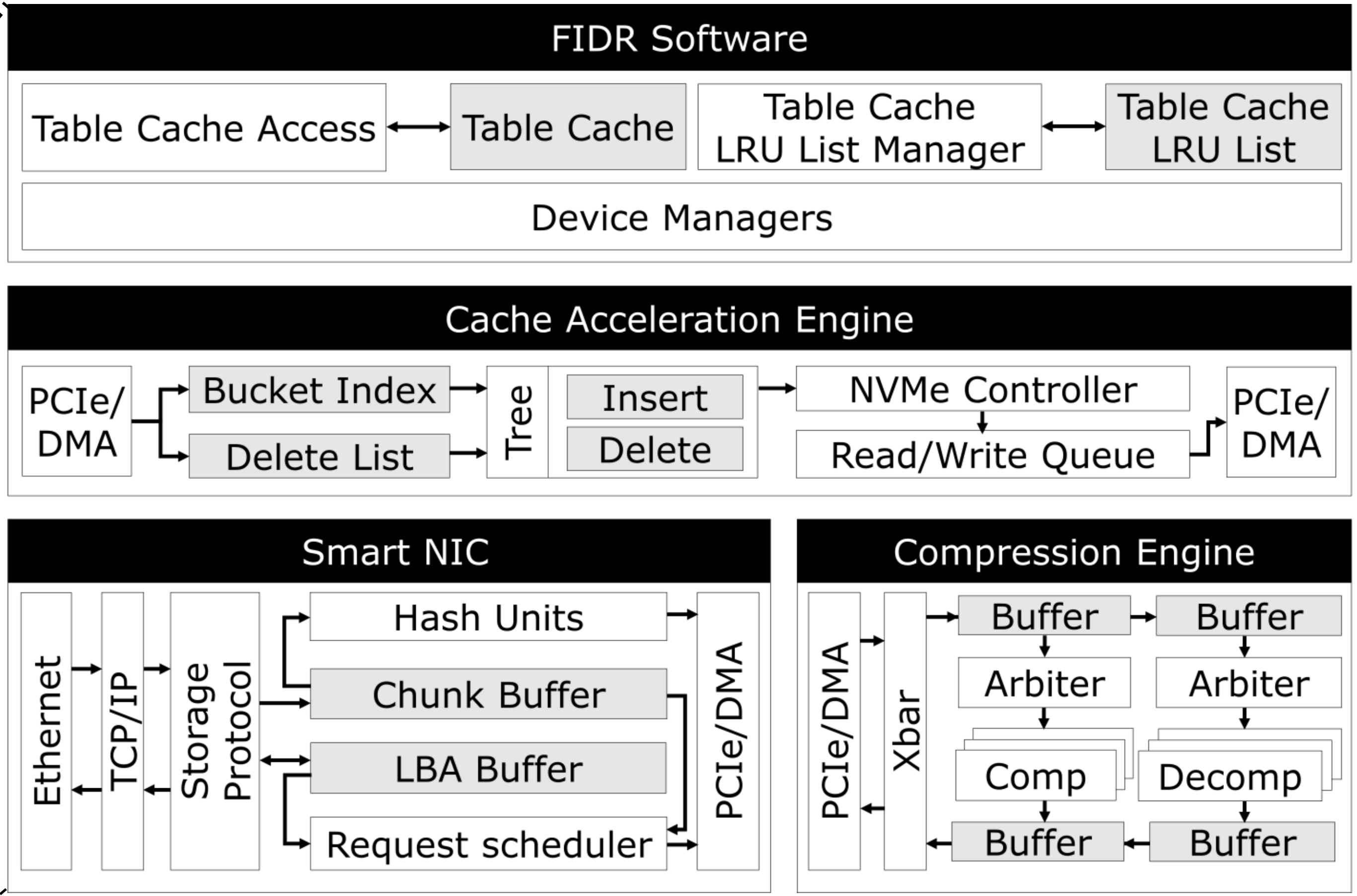


3. NIC-assisted pipelining

(+) Reduced CPU/memory overhead

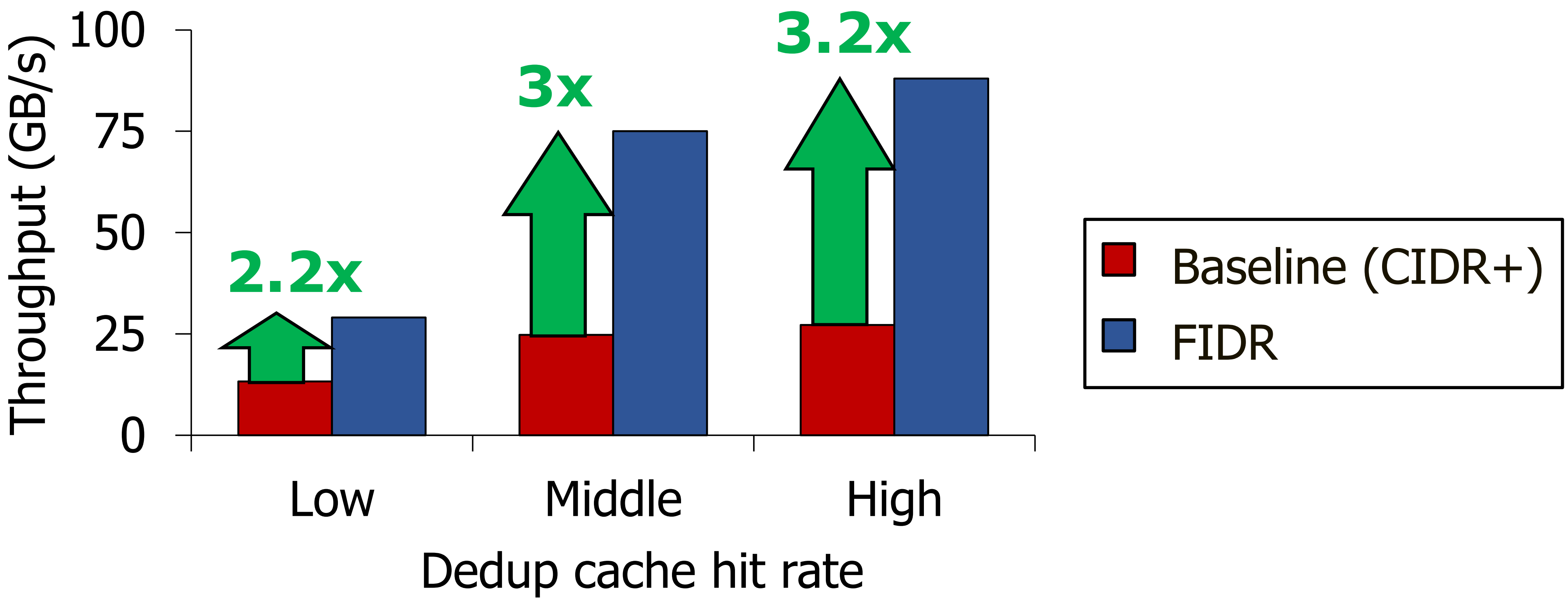
FIDR Prototype





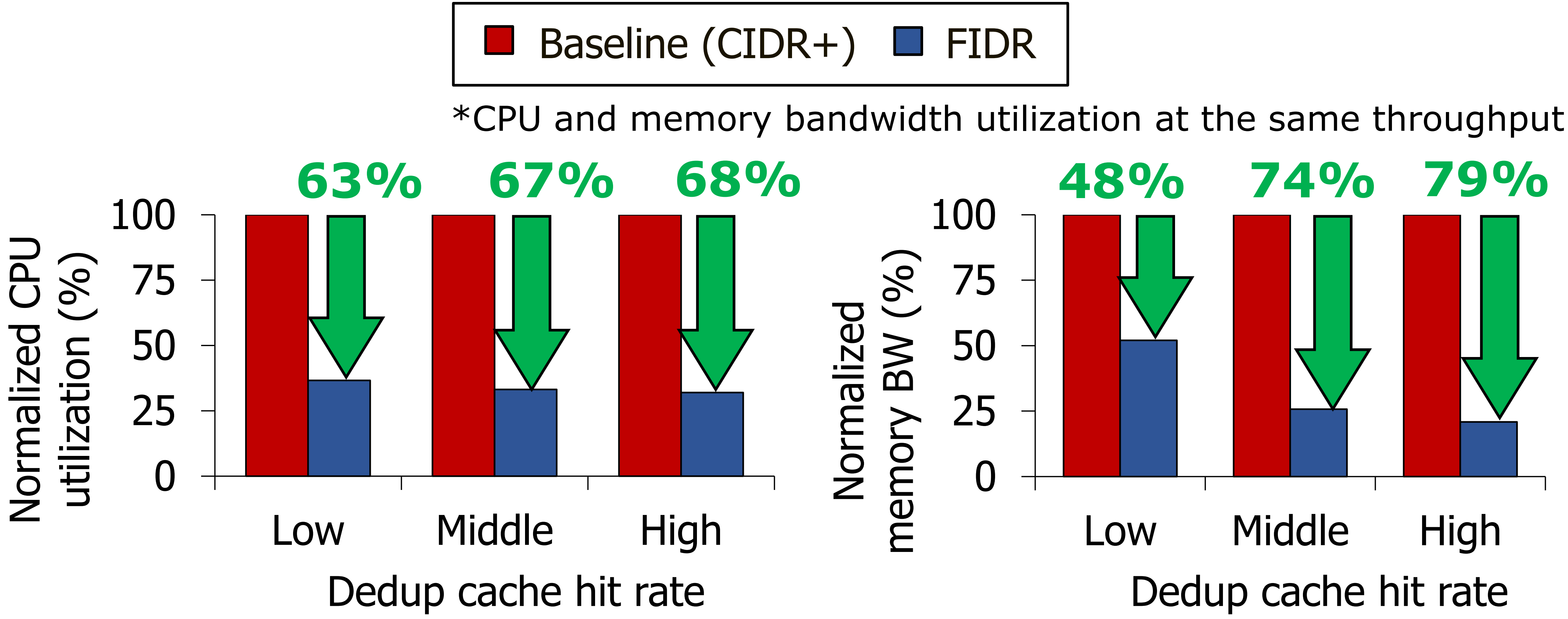
- Three VCU1525 FPGAs for a NIC, a CIDR engine, and a Cache engine
- Four Samsung 970 Pro SSDs, Intel E5-2650 v4 CPU

FIDR's High Scalability



FIDR scales up to 80GB/s throughput while CIDR+ suffers from CPU/memory bottleneck

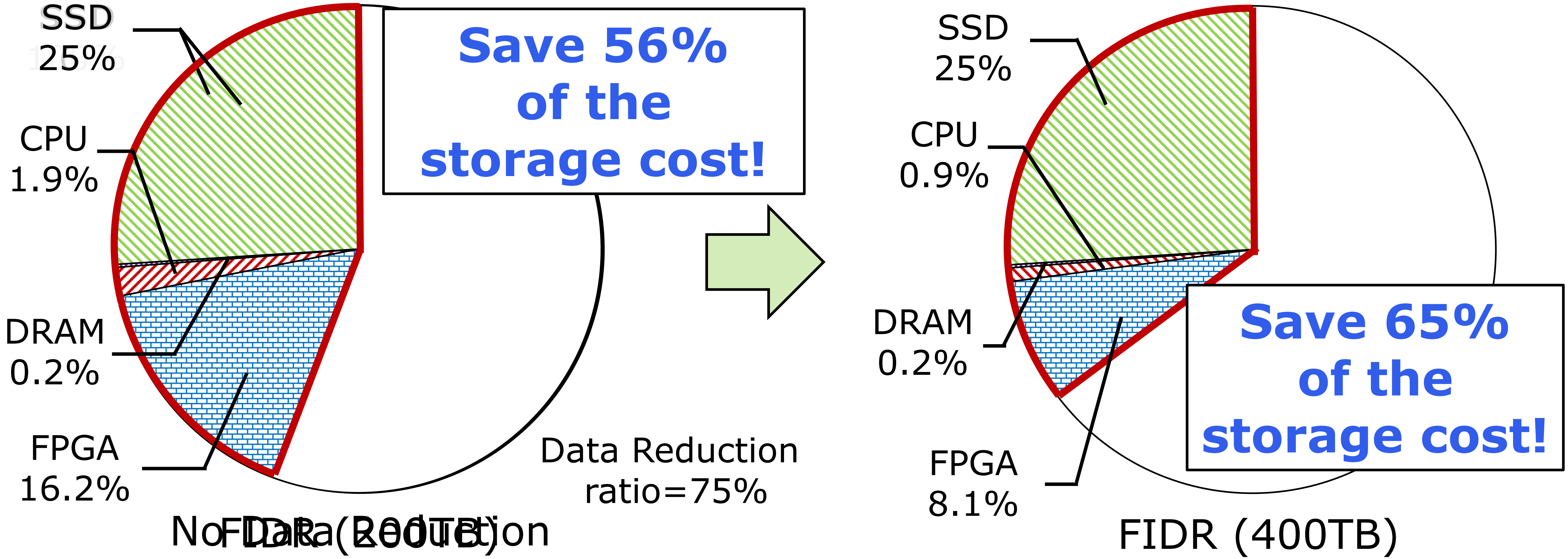
FIDR's Efficient System Resource Usage



FIDR utilizes CPU and memory BW more efficiently!

FIDR’s Cost-effectiveness

- **Cost saving = reduced SSD cost – additional HW cost**

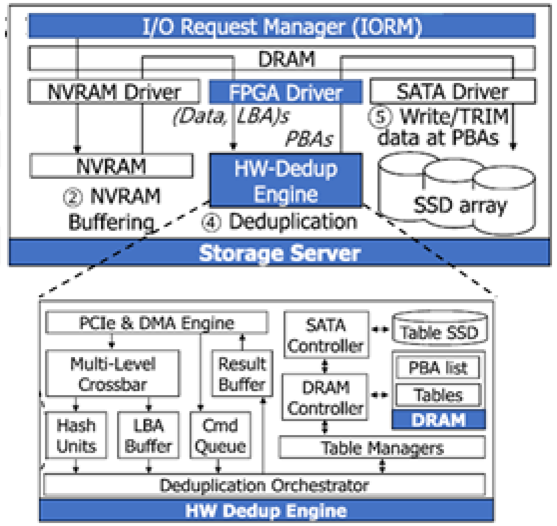


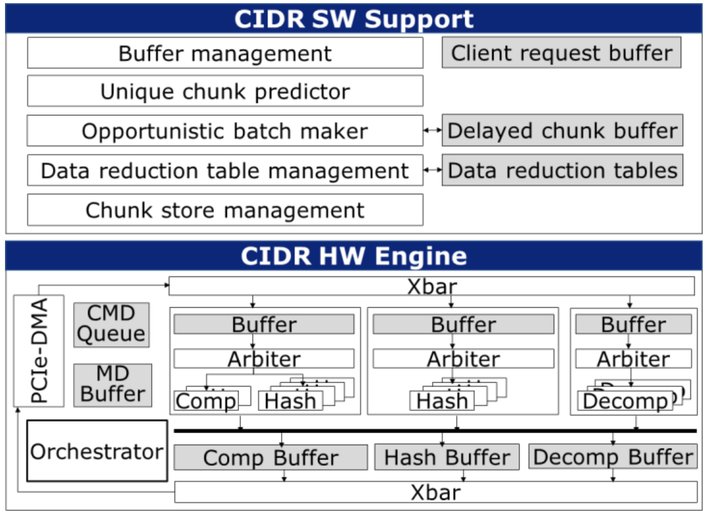
FIDR’s cost-effectiveness is higher with larger storage size

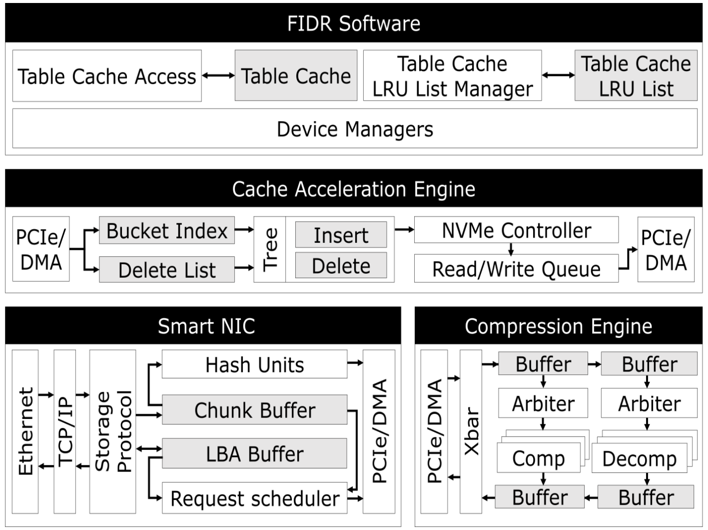
Conclusion

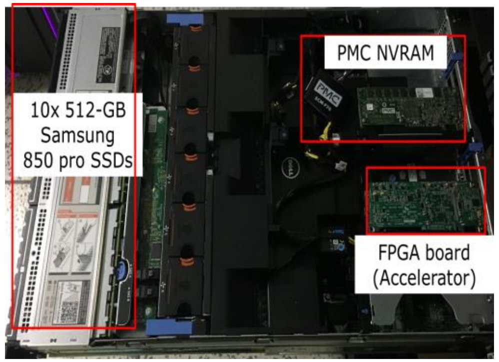
- **Lack of scalability of existing data reduction approaches**
  - High CPU utilization (SW approach)
  - Low data reduction or low device utilization (Hardware approaches)
- **Proposed a scalable HW/SW architecture**
  - Almost **an order of magnitude faster** than optimized SW
  - **Minimal utilization of CPU & memory BW**
  - **Efficient HW** accelerator usage **& 59.3% less storage costs**
- **Scalable to multi-Tbps and PB capacity SSD arrays**

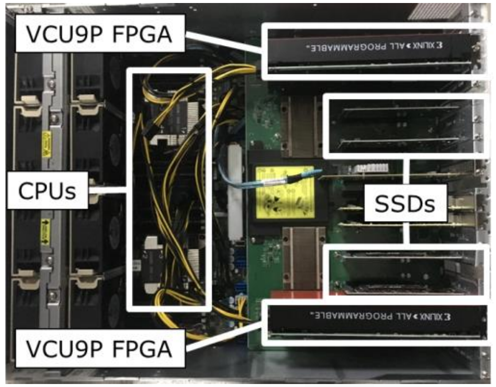
Thank you

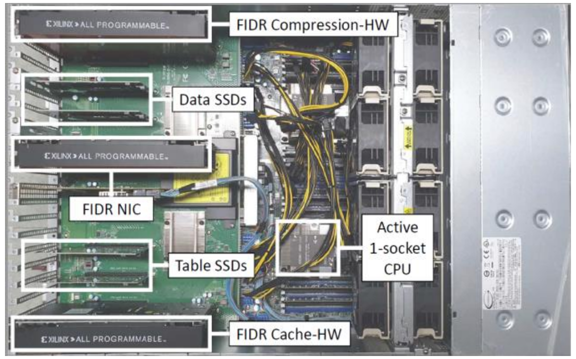












Any Questions?