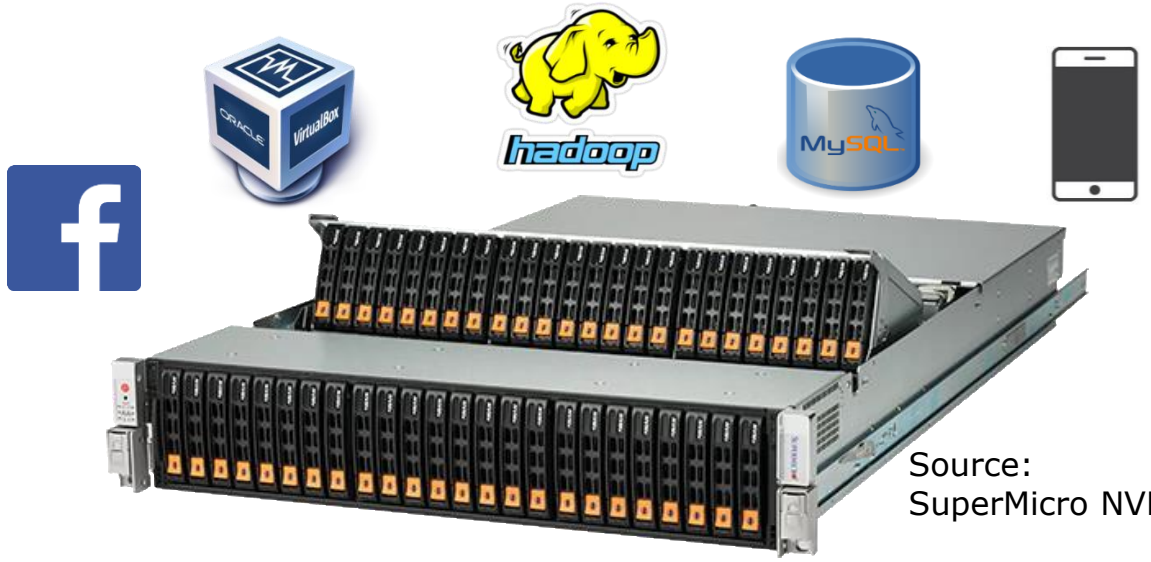
# CIDR: A Cost-Effective In-Line Data Reduction System for Terabit-per-Second Scale SSD Arrays

Mohammadamin Ajdari[1], Pyeongsu Park[2], Joonsung Kim[2],
Dongup Kwon[2], and Jangwoo Kim[2]

[1]Dept. of Computer Science and Engineering, POSTECH
[2]Dept. of Electrical and Computer Engineering, Seoul National University

# Era of High Performance Storage Servers

- **Modern SSD array (node) : 100+ GB/s, 100+ TB**
  - Products by Intel, DELL EMC, Pure Storage, SmartIOPS, …




Source:
SuperMicro NVMe Platform

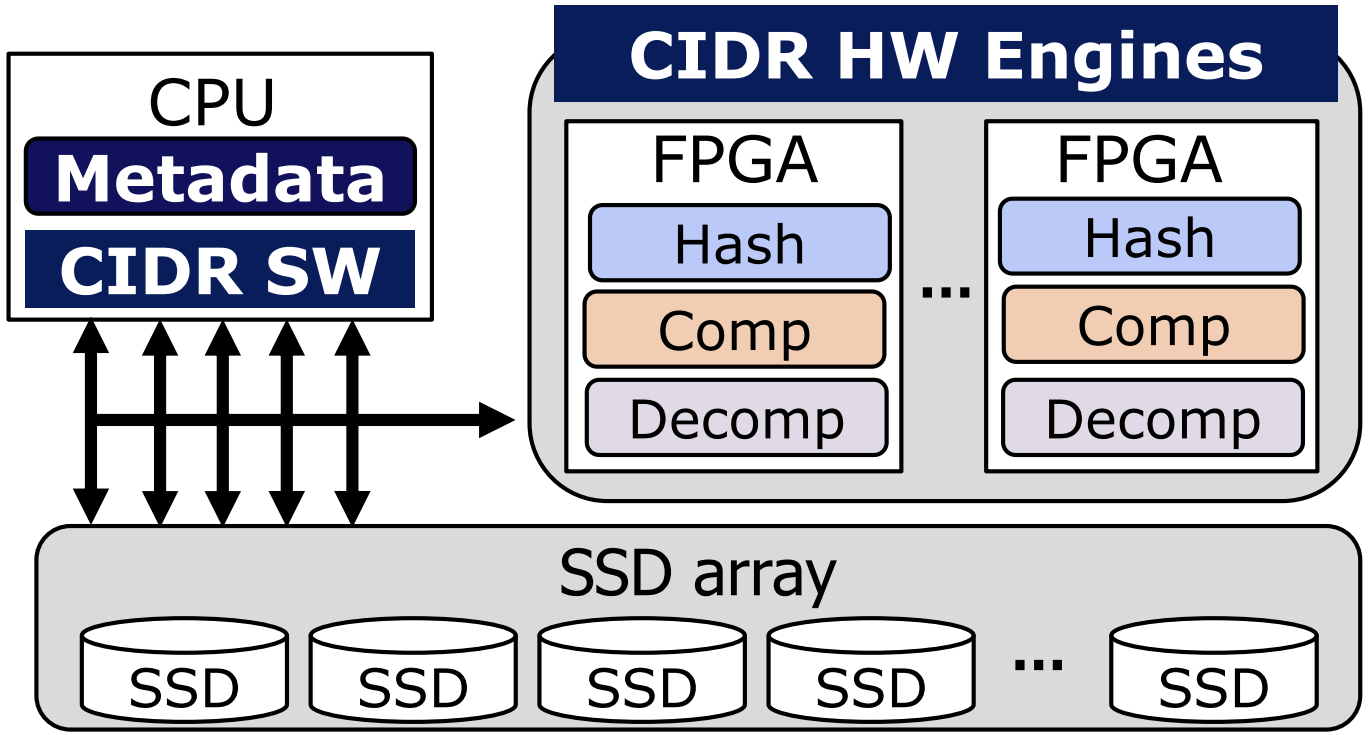
→ An increasing number of fast SSDs per server

Data reduction becomes cost critical!

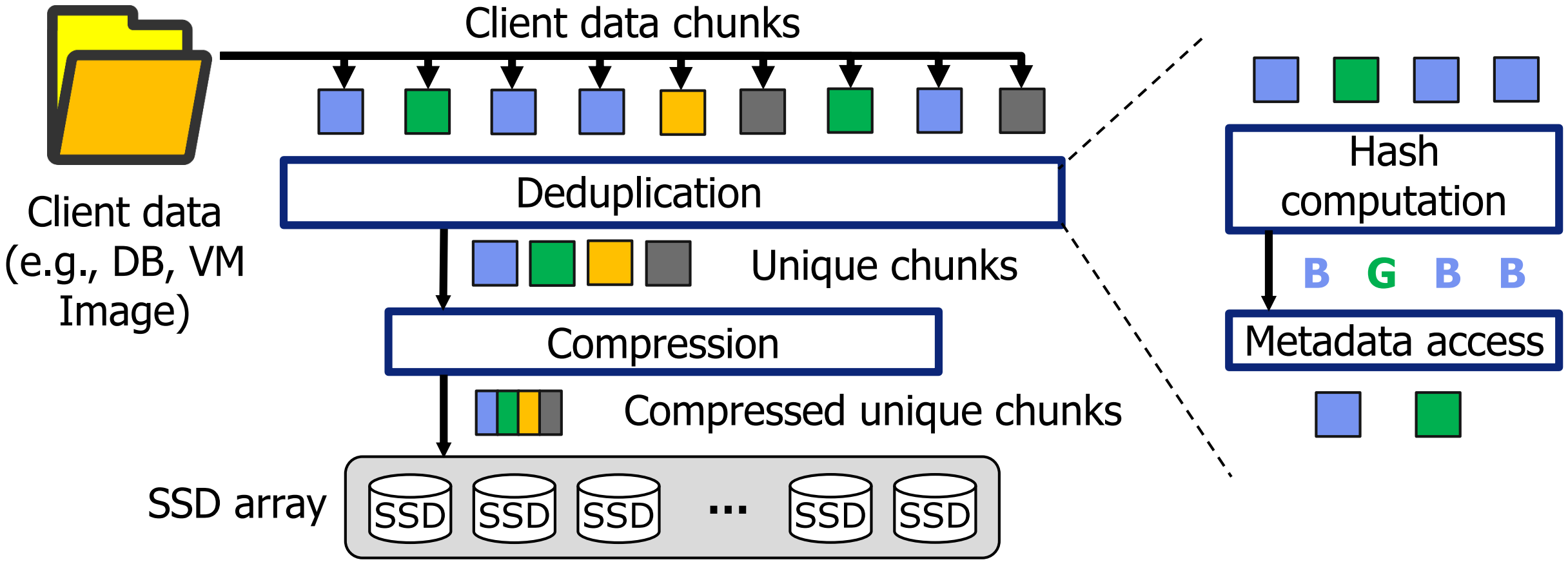
**Storage systems face challenges due to high SSD costs!**

# CIDR: Cost-effective Inline Data Reduction

- **Scalable data reduction system for modern SSD arrays**
  - Acceleration through **a scalable FPGA array**
  - **SW/HW orchestration** for flexibility and efficiency

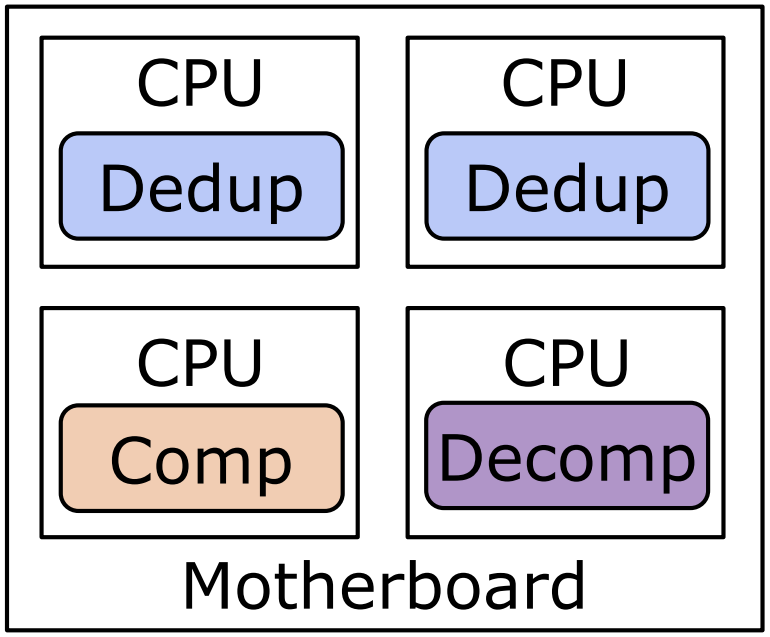


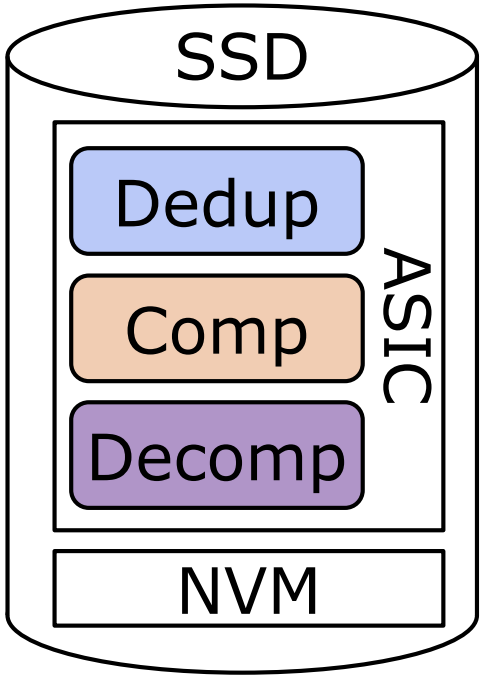
# Data Reduction Basics



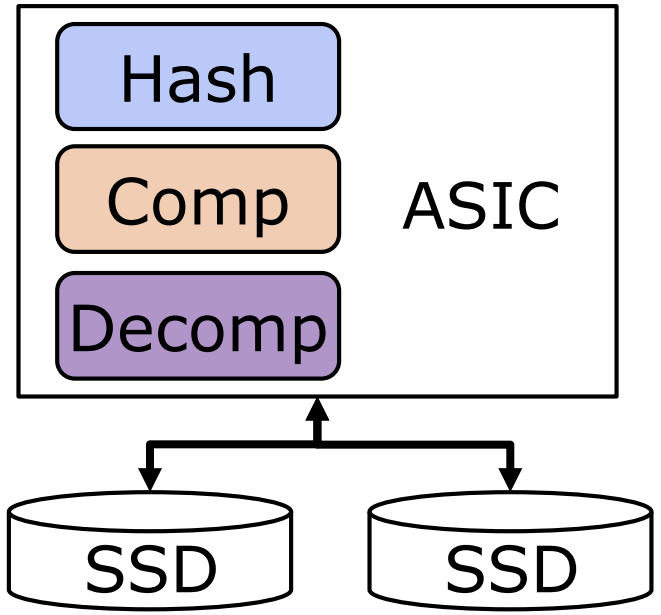

**Deduplication + compression → 60-90% data reduction**

# Existing Approaches




Intra-SSD




Dedicated ASIC

**SW-based**

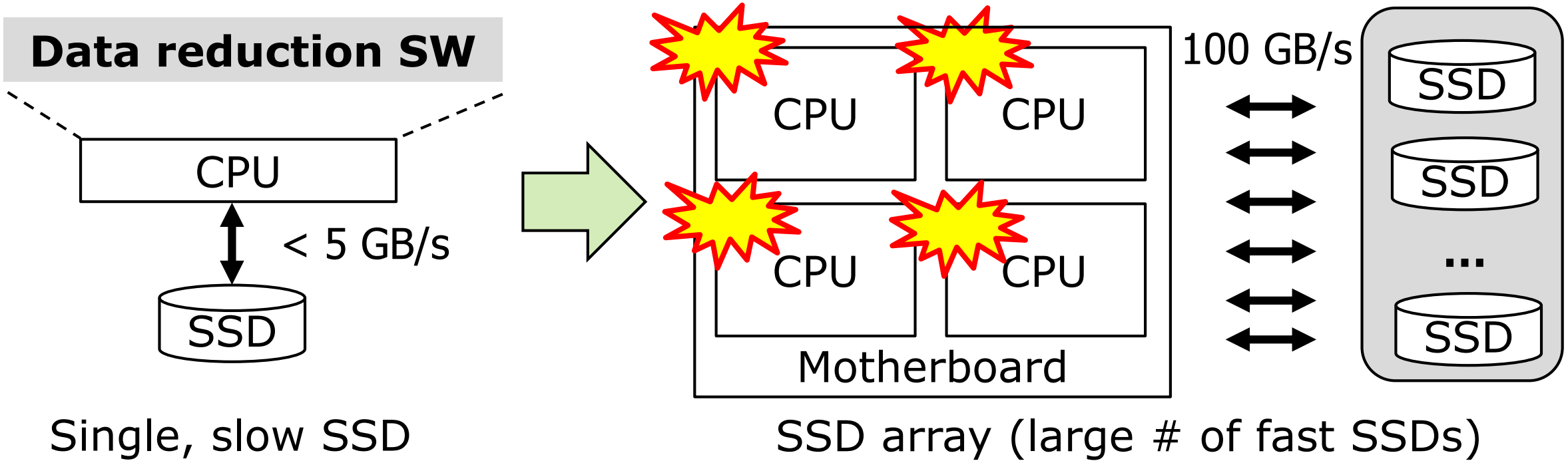
**HW acceleration**

# 1. Limitations of SW-Based Reduction

- **CPUs run optimized data reduction operations**

**(-) Low throughput scalability due to CPU bottleneck**

- Many compute-intensive operations & limited # of CPU sockets




Single, slow SSD

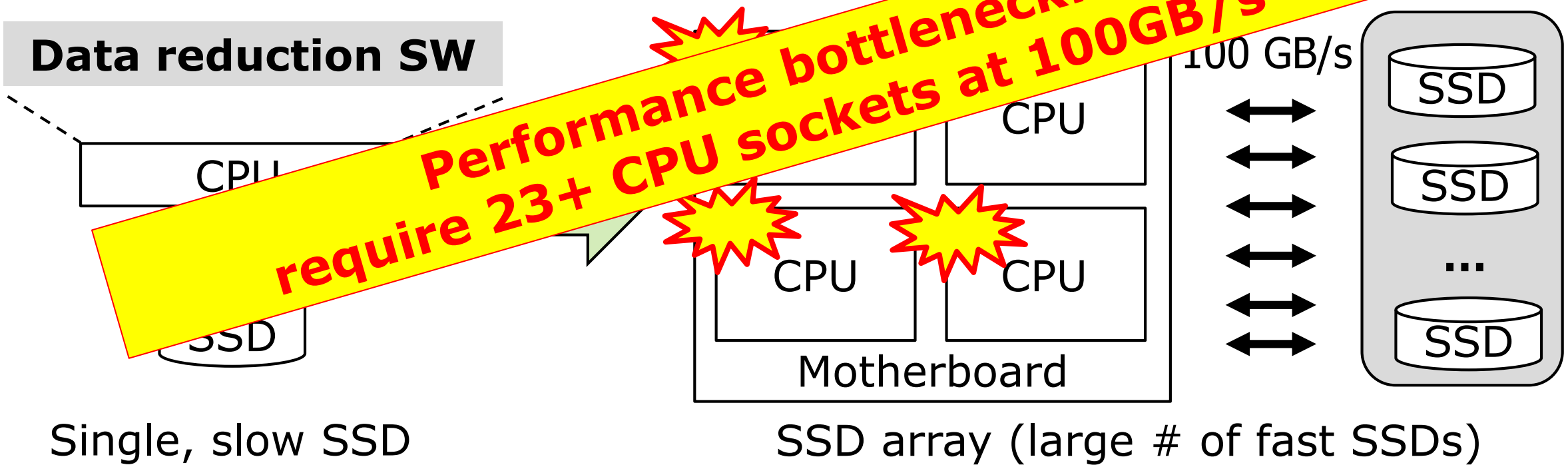
SSD array (large # of fast SSDs)

# 1. Limitations of SW-Based Reduction

- **CPUs run optimized data reduction operations**

**(-) Low throughput scalability due to CPU bottleneck**

– Many compute-intensive operations & limited # of [illegible]s

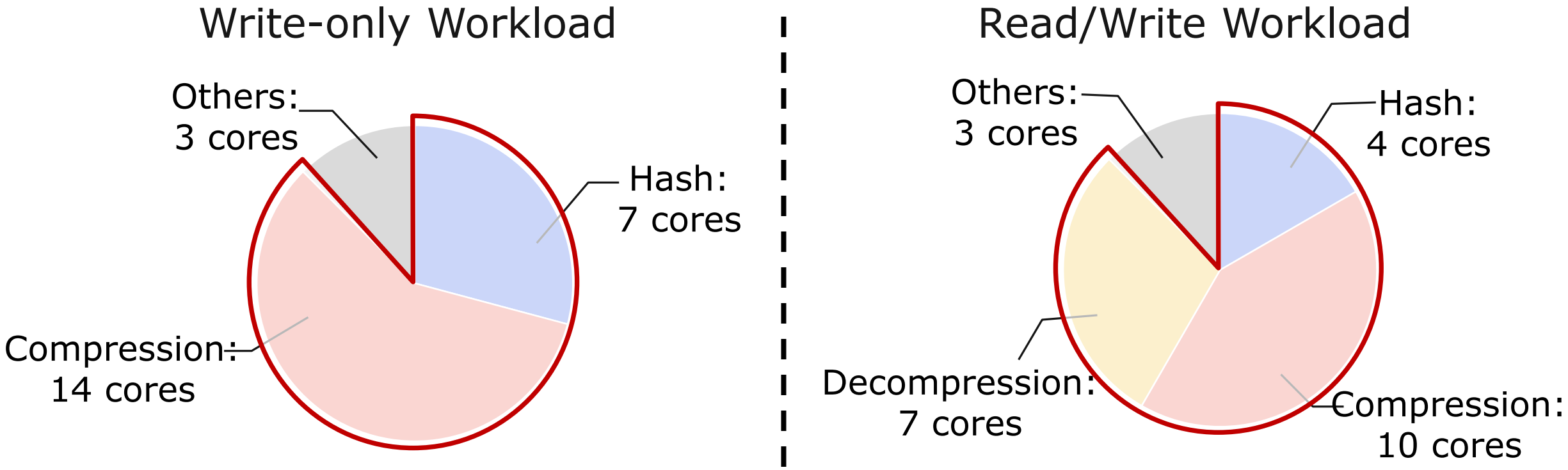



Single, slow SSD

SSD array (large # of fast SSDs)

# Heavy Computations on CPUs

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



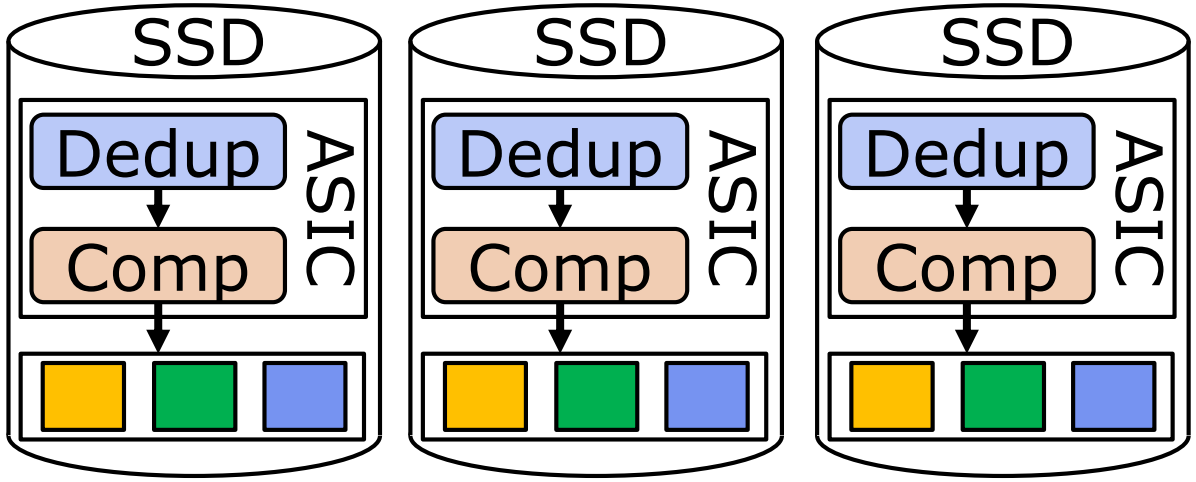

**90 % of CPU-intensive operations → hardware acceleration**

# 2. Limitations of Intra-SSD HW Acceleration

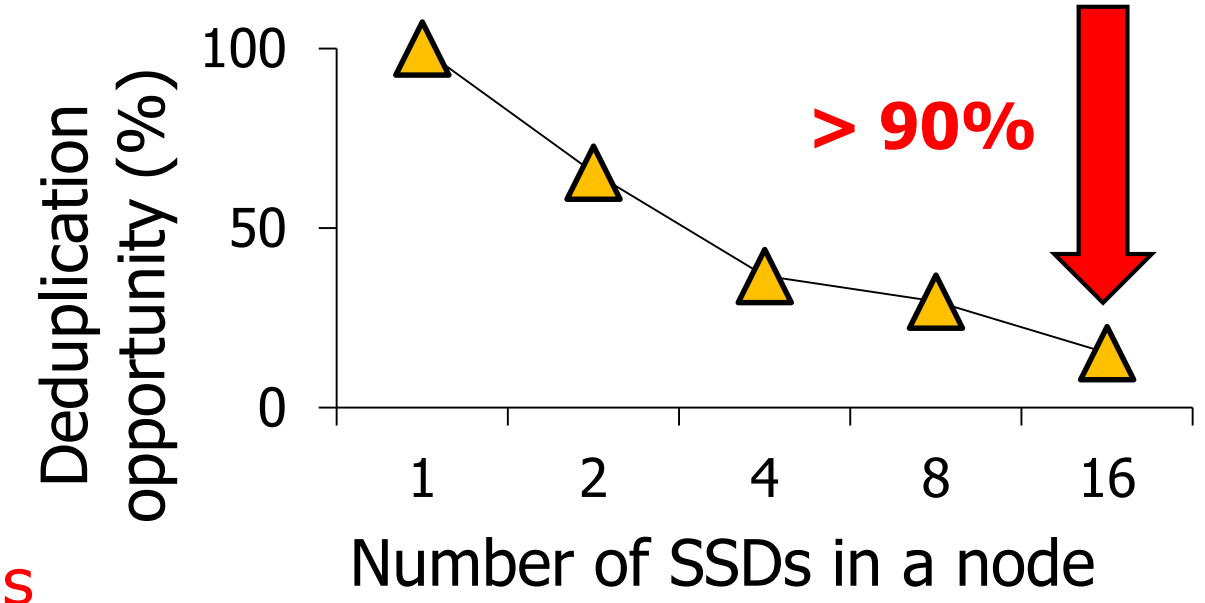
- **Accelerators in each SSD for scalable throughput**

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

- Decentralized metadata management




Cannot detect duplicates in other SSDs

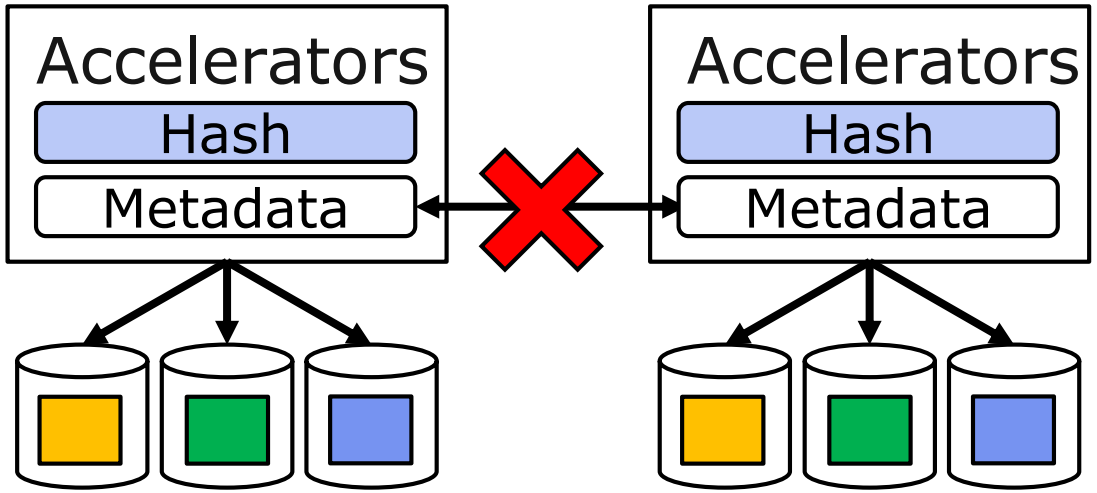


# 3. Limitations of Dedicated HW Acceleration

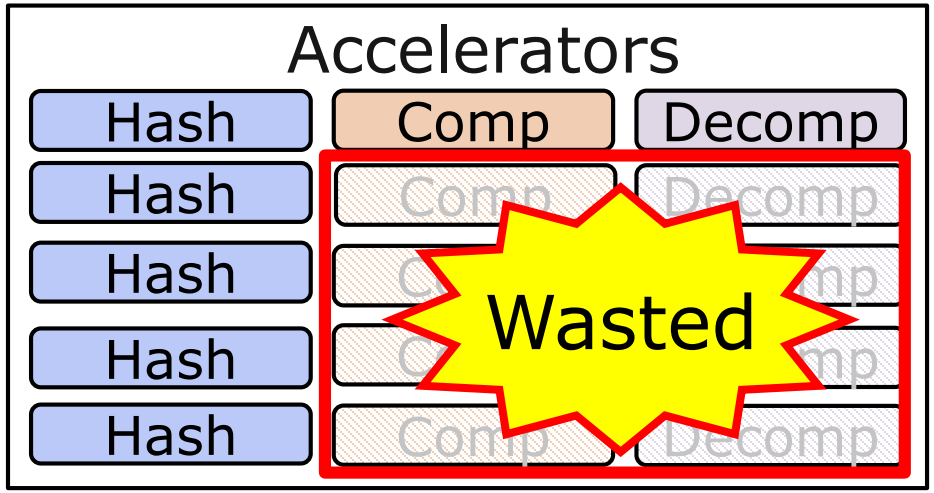
- **"Shared-nothing" accelerators, separated from SSDs**

**(-) Low device utilization due to fixed provisioning**

- Fixed, inflexible resource overprovisioning for the worst-case




No SW support
for device orchestration
(e.g., centralized metadata)




Many wasted resource
(e.g., many duplicates &
write-intensive workload)

# 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 |

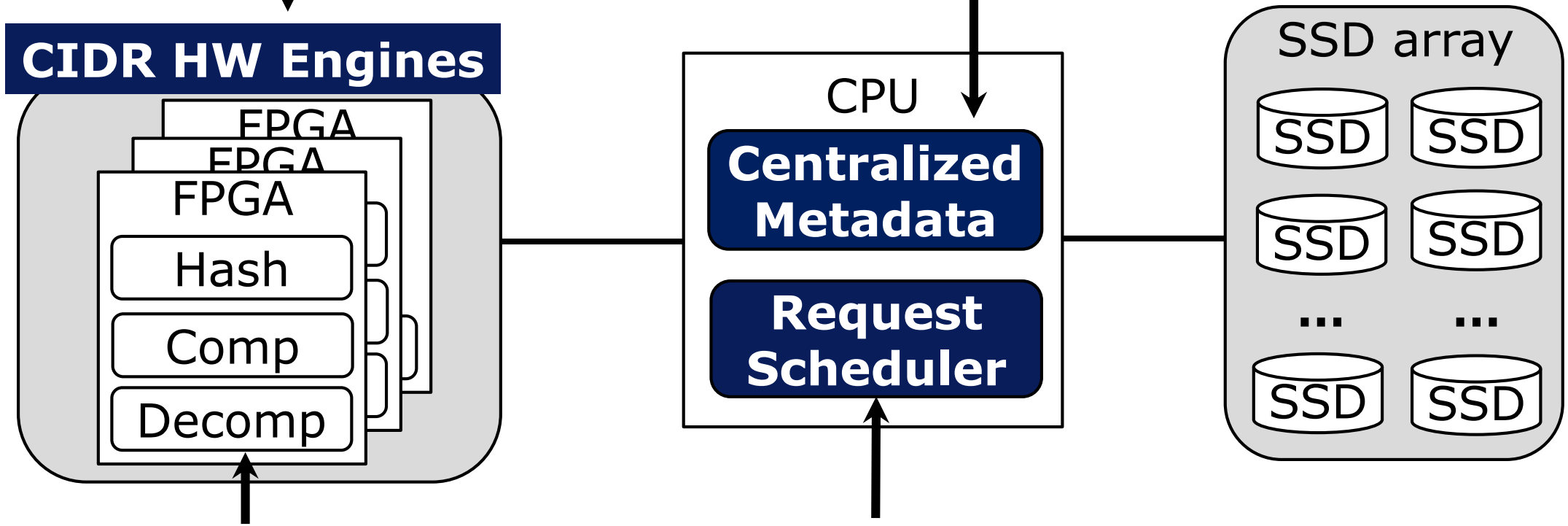
# Index

- Motivation
- **CIDR: A Cost-Effective In-line Data Reduction**
  - **Key ideas**
  - Architecture details
- Evaluation
- Conclusion

# Four Key Ideas of CIDR

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

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

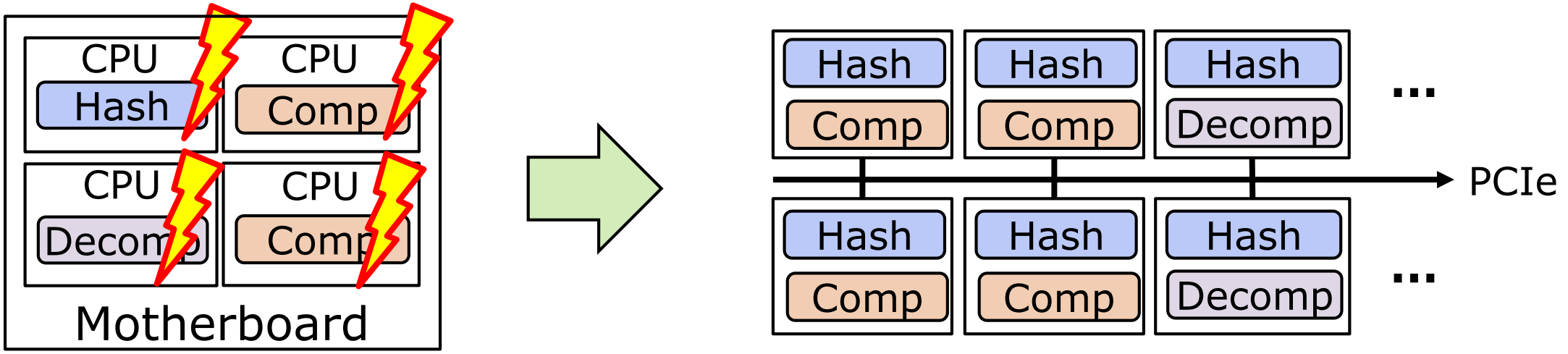



3. Long-term FPGA reconfig
⇒ **Efficient device utilization**

4. Short-term request scheduler
⇒ **Efficient device utilization**

# Key Idea #1: Scalable FPGA Array

- **Deploy a CPU-free, scalable FPGA accelerator array**
  - Offload CPU-heavy operations to FPGA engines
  - Deploy "multiple" accelerators on a scalable PCIe tree (c.f., 4 for CPUs)



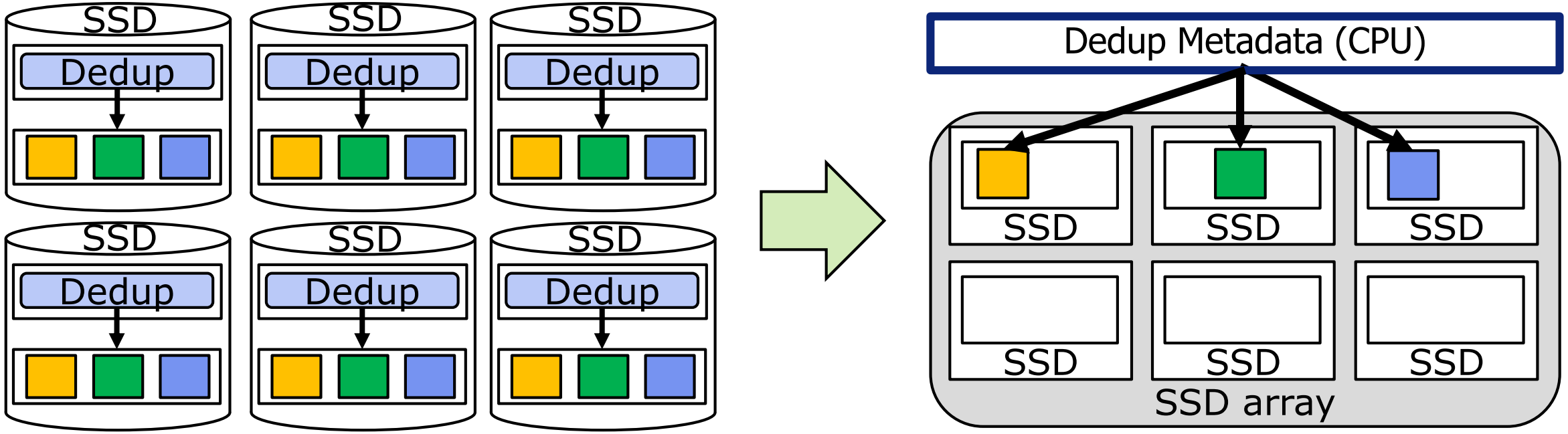

Limited # of slow CPUs

Fast, scalable FPGA array

**"Higher throughput" with a scalable FPGA array!**

# Key Idea #2: Centralized Table Management

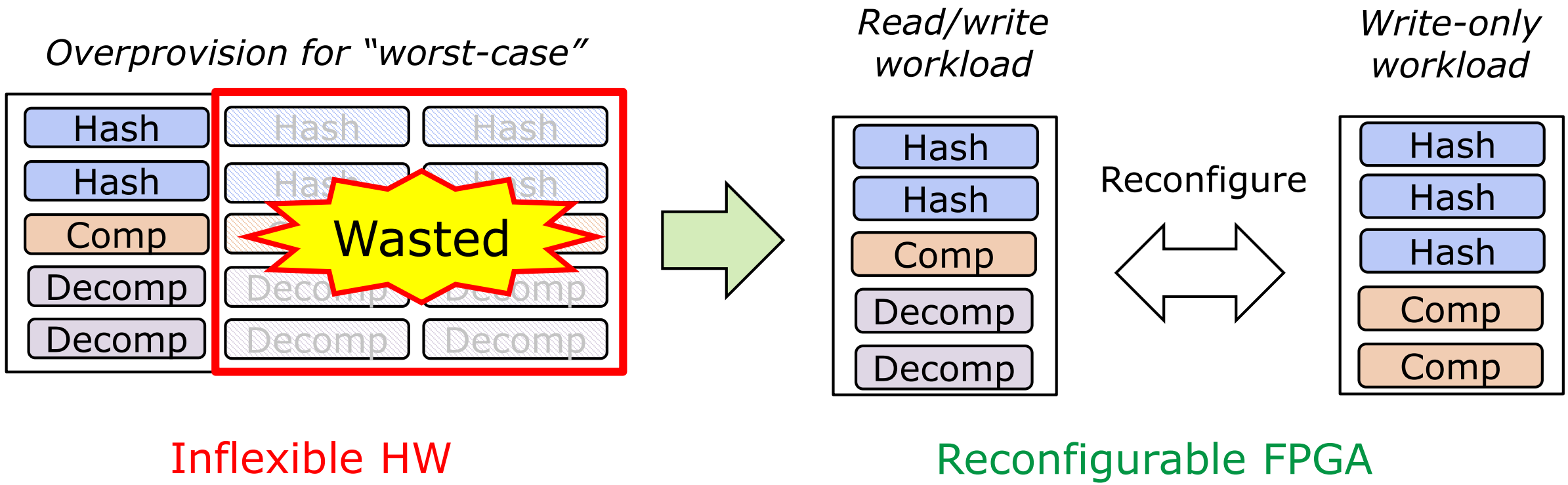
- **Detect all duplicate chunks in a large SSD array**




**"High data reduction" with the centralized table!**

# Key Idea #3: Long-Term FPGA Reconfig

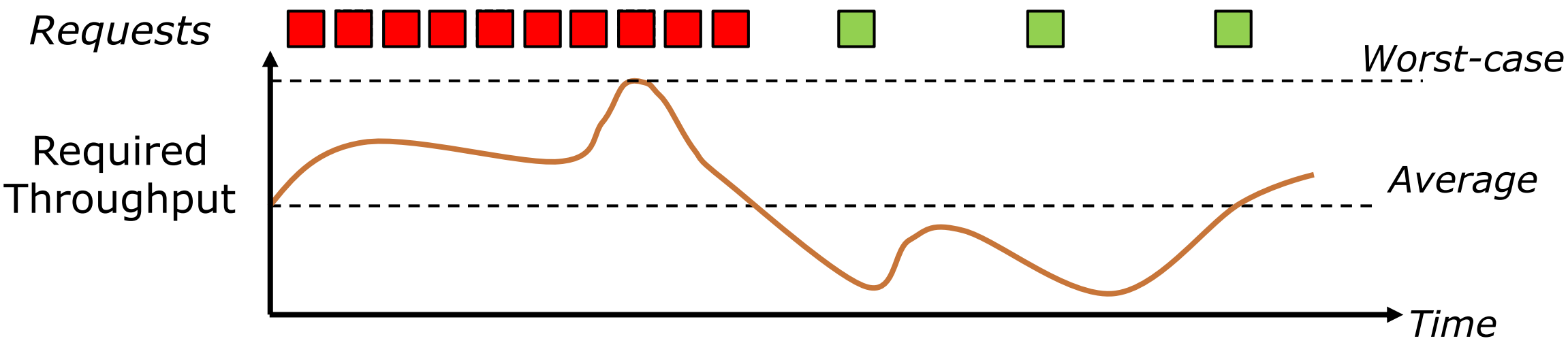
- **Reconfigure FPGAs to workload's average behavior**




**"Minimal HW resources" with reconfigurable FPGAs!**

# Key Idea #4: Short-term Request Scheduler

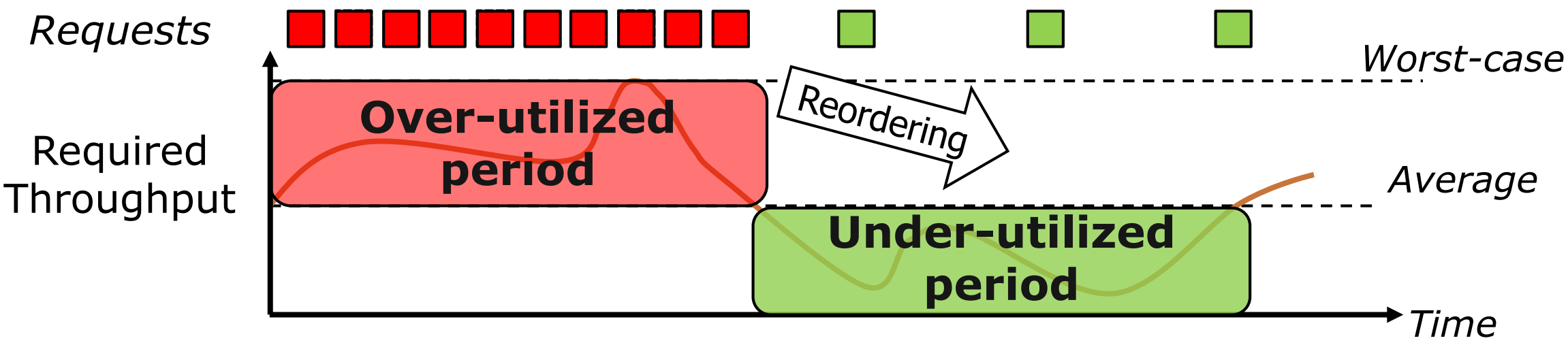
- **Schedule requests considering available HW resources**
  - Shift the load of over-utilization period to under-utilization period




**"High resource utilization" with smart request scheduling!**

# Key Idea #4: Short-term Request Scheduler

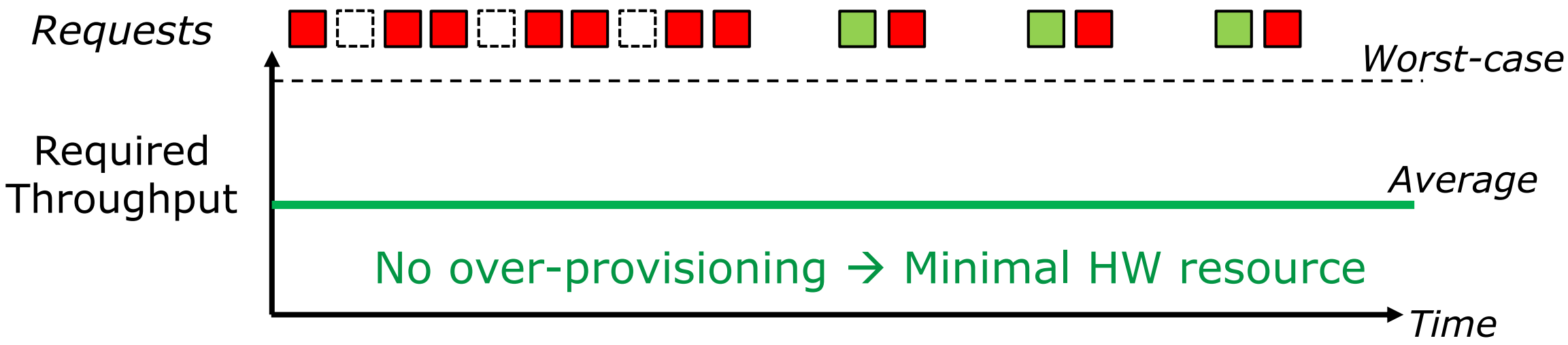
- **Schedule requests considering available HW resources**
  - Shift the load of over-utilization period to under-utilization period




**"High resource utilization" with smart request scheduling!**

# Key Idea #4: Short-term Request Scheduler

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

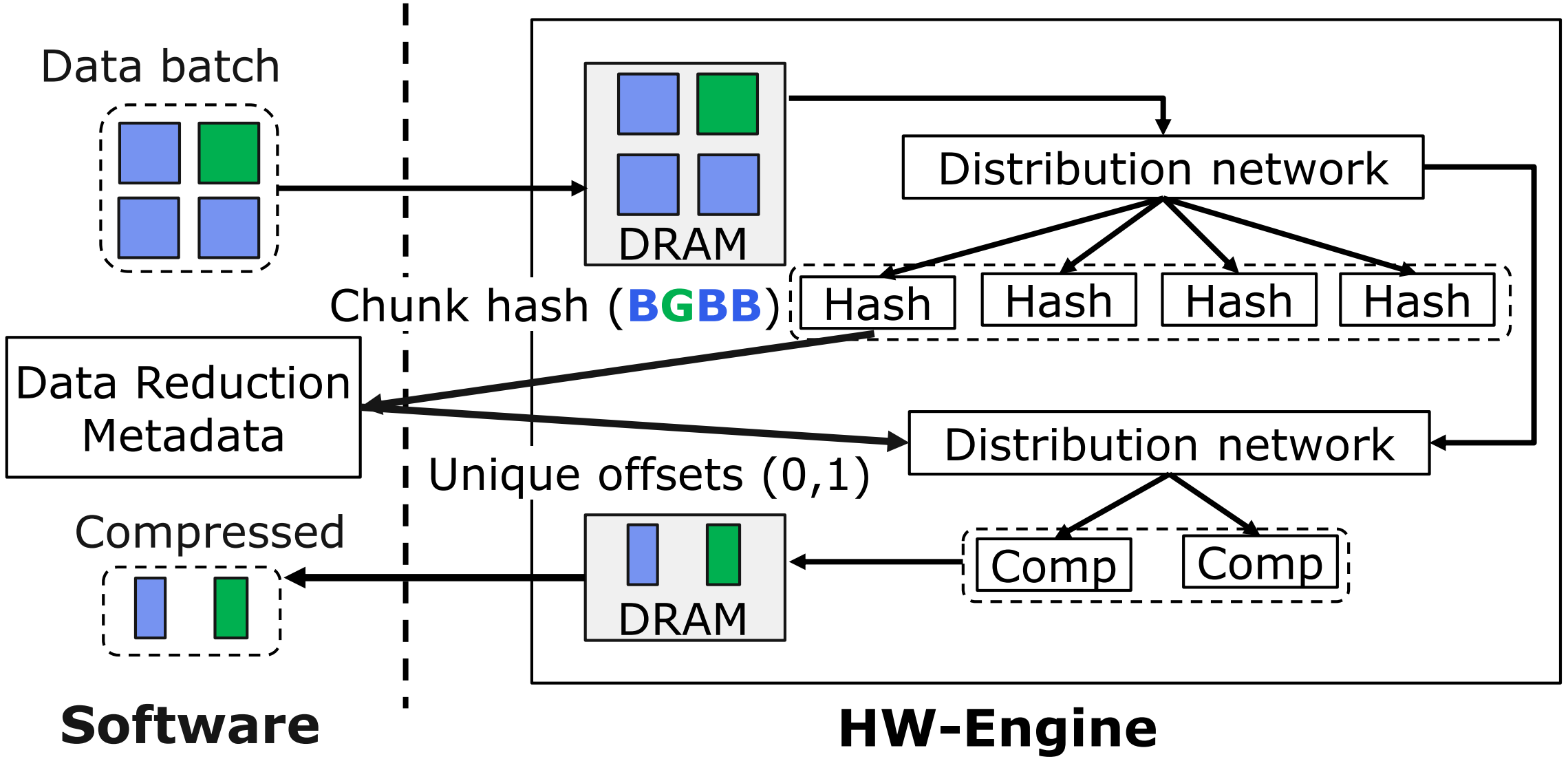



**"High resource utilization" with smart request scheduling!**

# Index

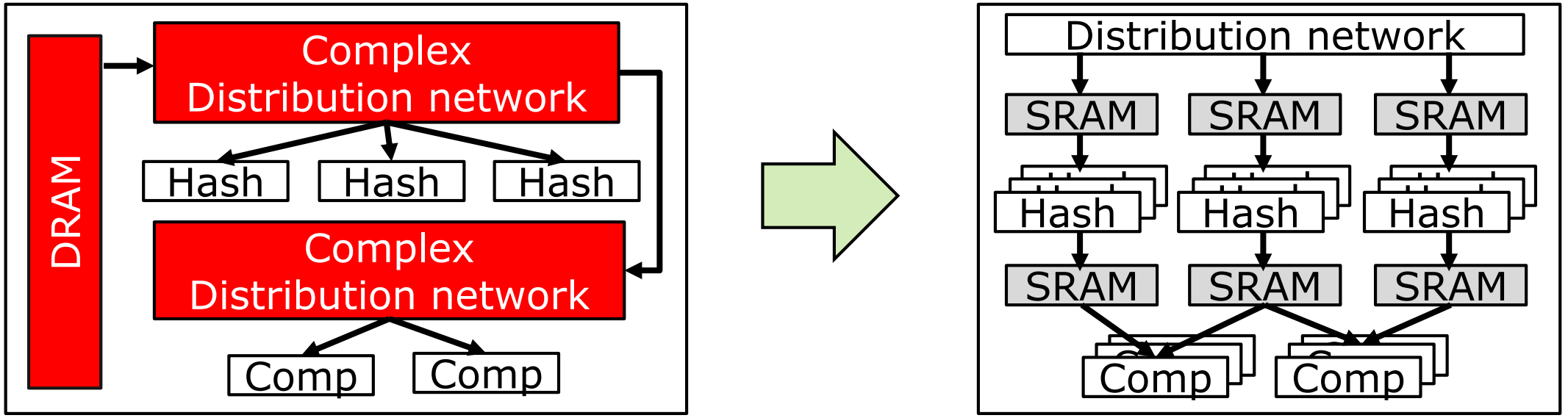
- Motivation
- **CIDR: A Cost-Effective In-line Data Reduction**
  - Key ideas
  - **Architecture details**
- Evaluation results
- Conclusion

# CIDR: Basic Write Flow



# Opt #1: Simplified, Minimized HW Structure

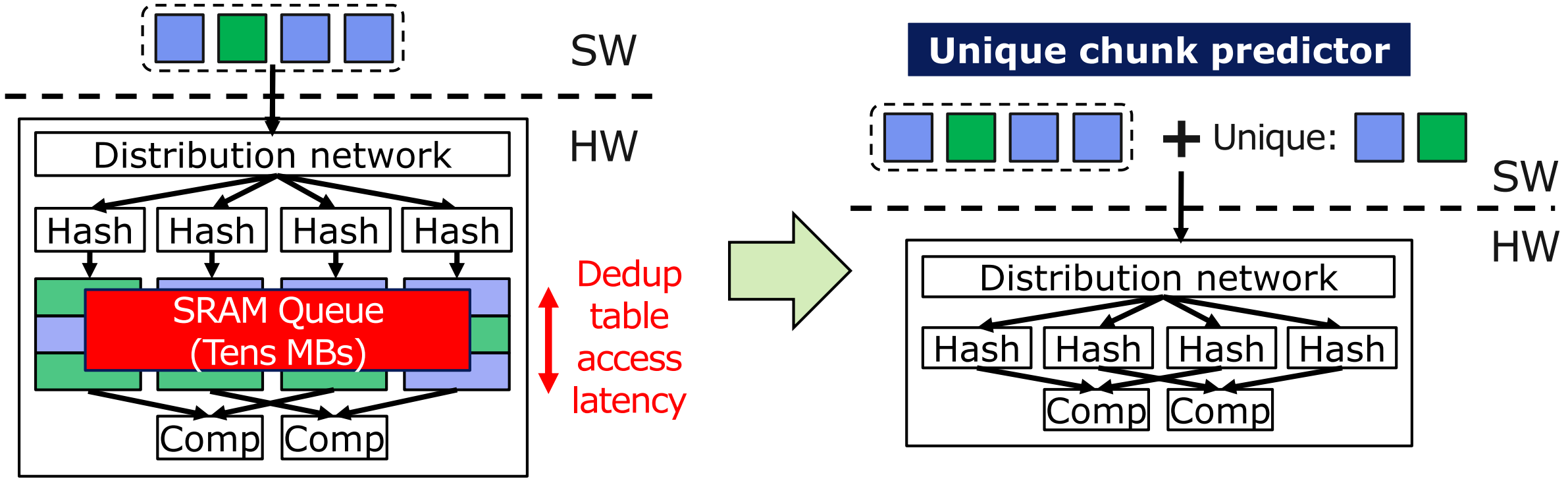
- Use SRAM buffer instead of slow DRAM buffer
- Cluster units to make distribution network simple




**Reduce overhead of data buffering & distribution network**

# Opt #2: Fast Initiation of Compression

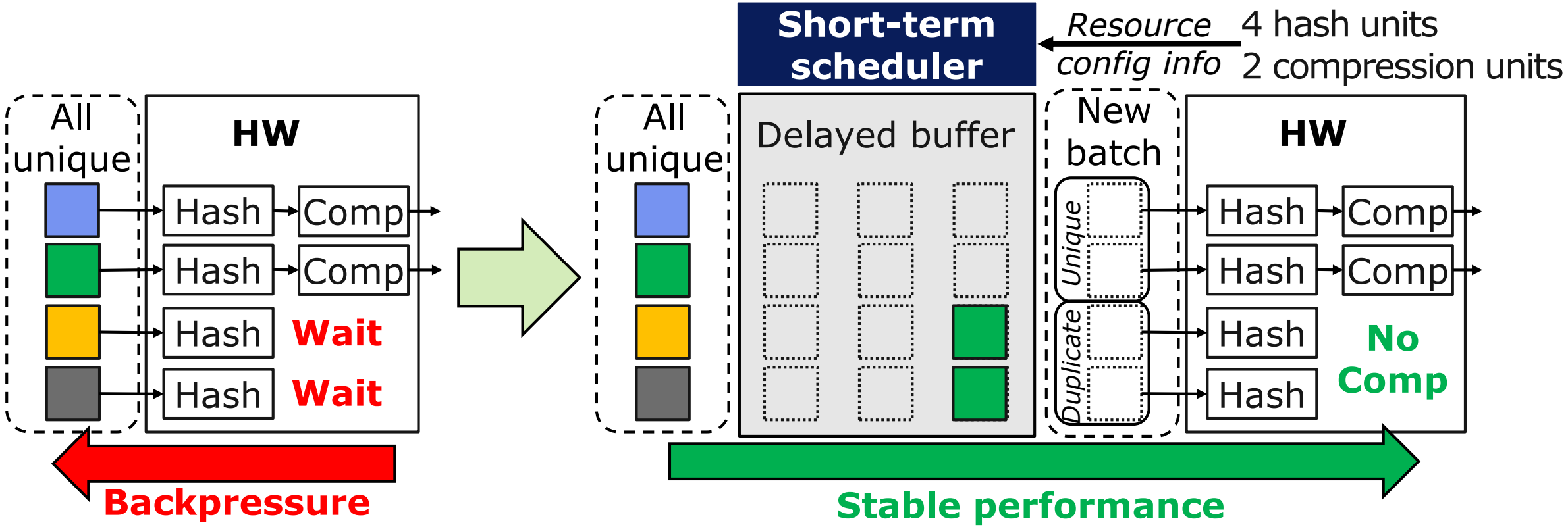
- **Remove the large SRAM queue by predicting uniqueness**




**Eliminate SRAM queueing by initiating compression early**

# Opt #3: Short-Term Request Scheduling

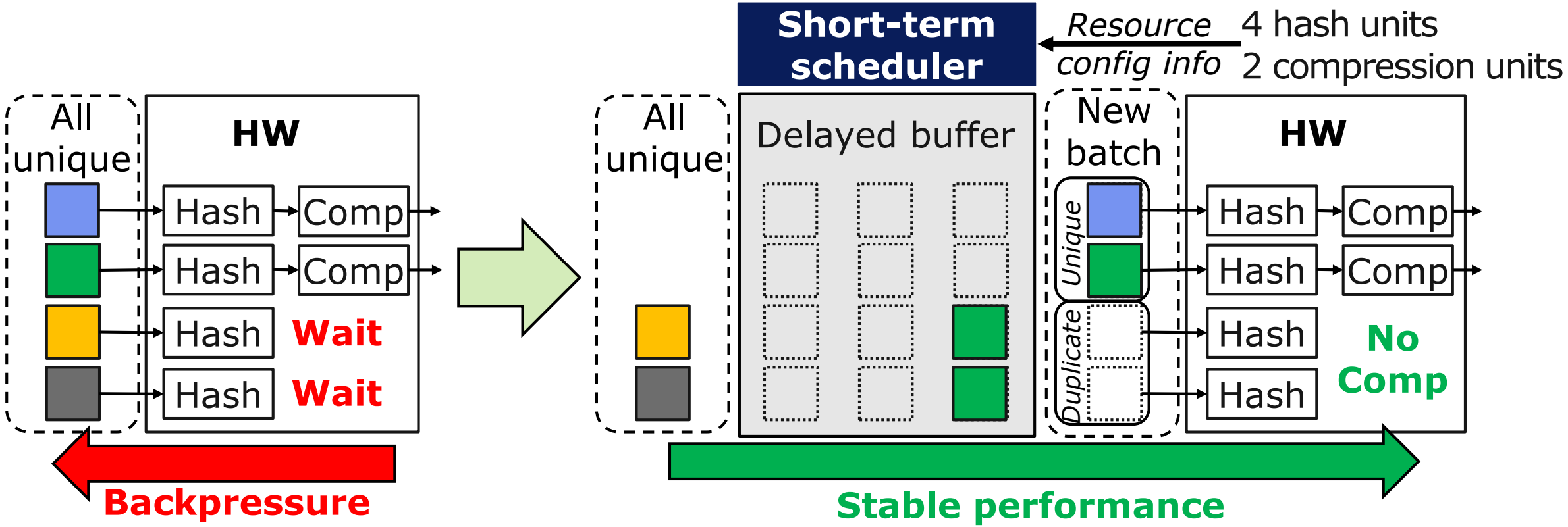
- **Delay over-subscribed requests into the delayed buffer**




**Match the workload behavior to the FPGA configuration**

# Opt #3: Short-Term Request Scheduling

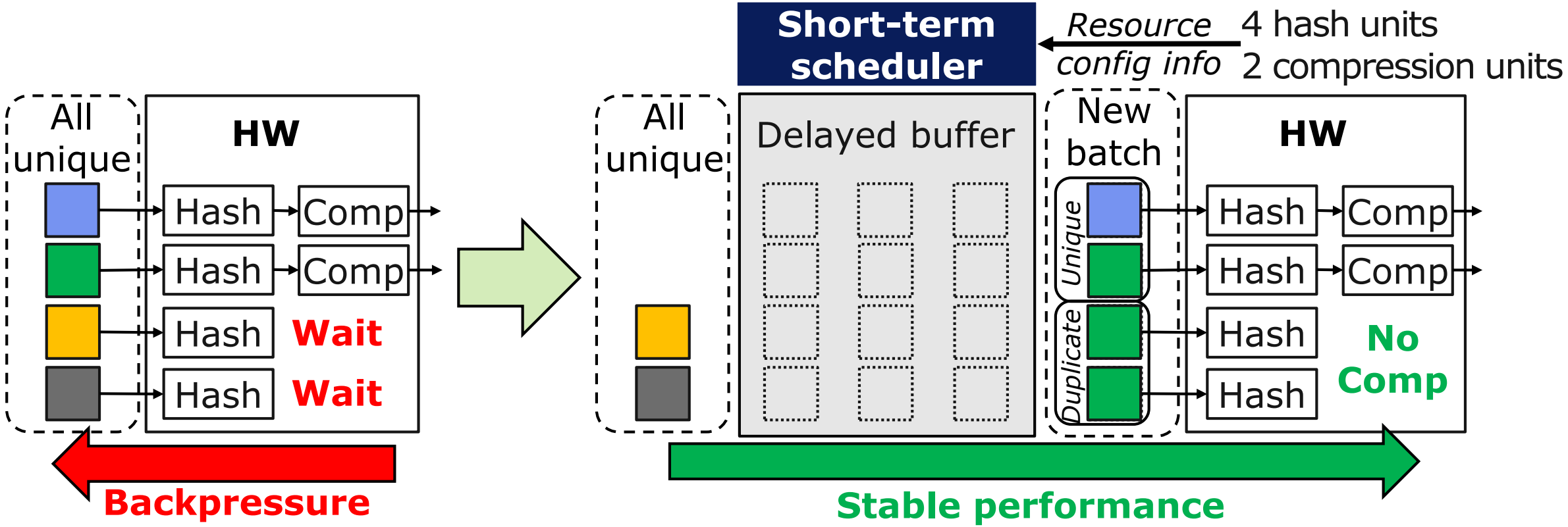
- **Delay over-subscribed requests into the delayed buffer**




**Match the workload behavior to the FPGA configuration**

# Opt #3: Short-Term Request Scheduling

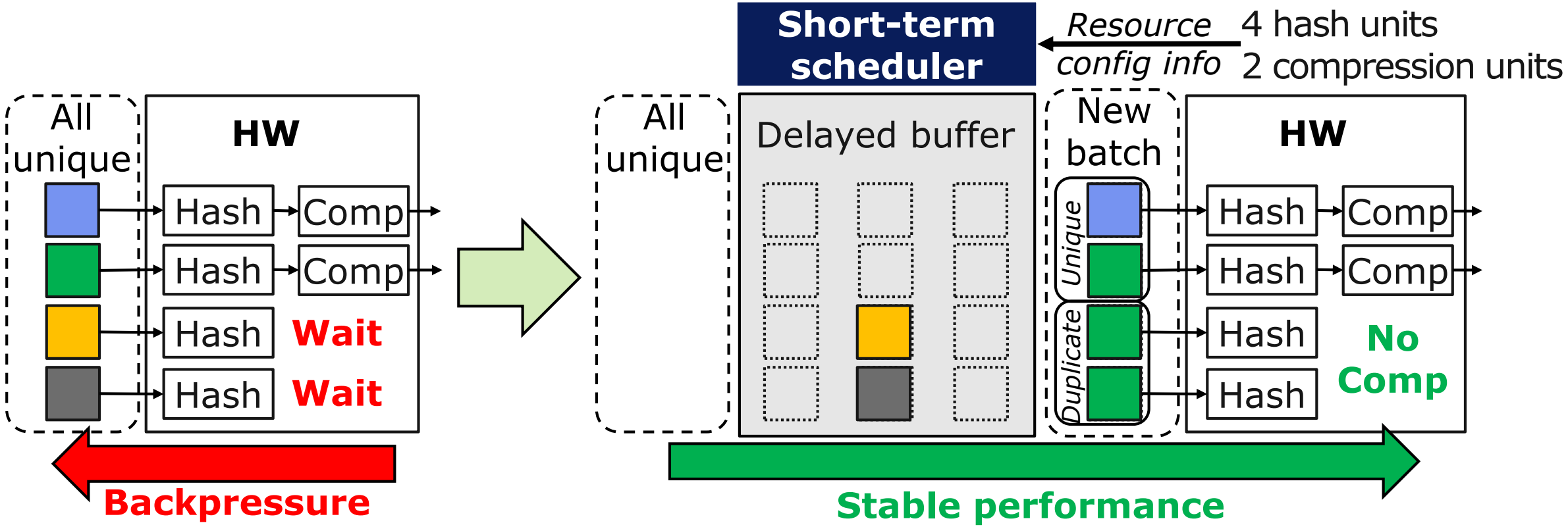
- **Delay over-subscribed requests into the delayed buffer**




**Match the workload behavior to the FPGA configuration**

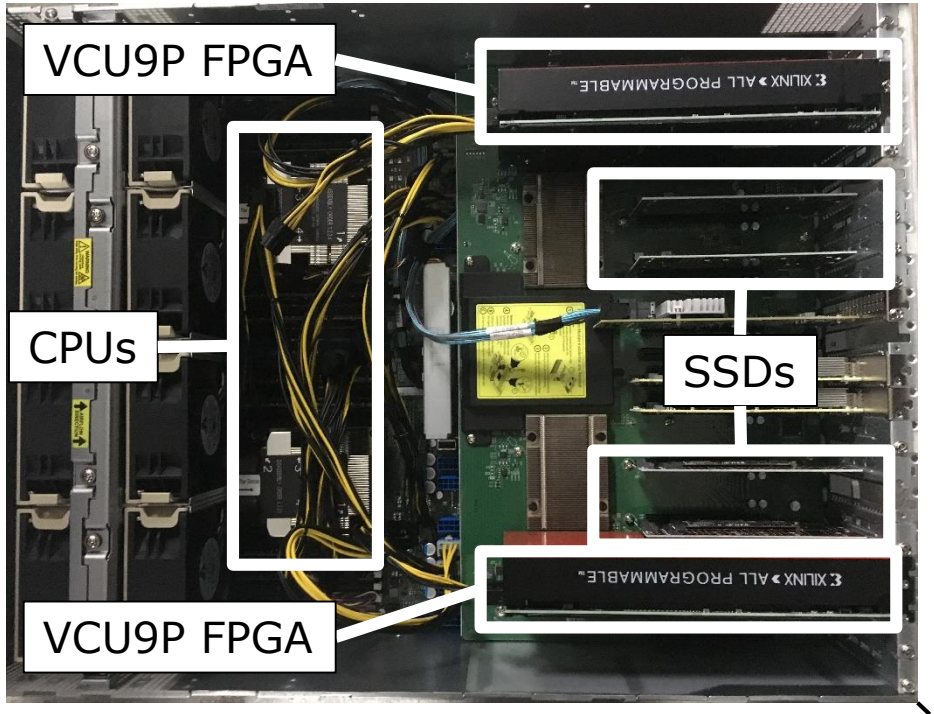
# Opt #3: Short-Term Request Scheduling

- **Delay over-subscribed requests into the delayed buffer**




**Match the workload behavior to the FPGA configuration**

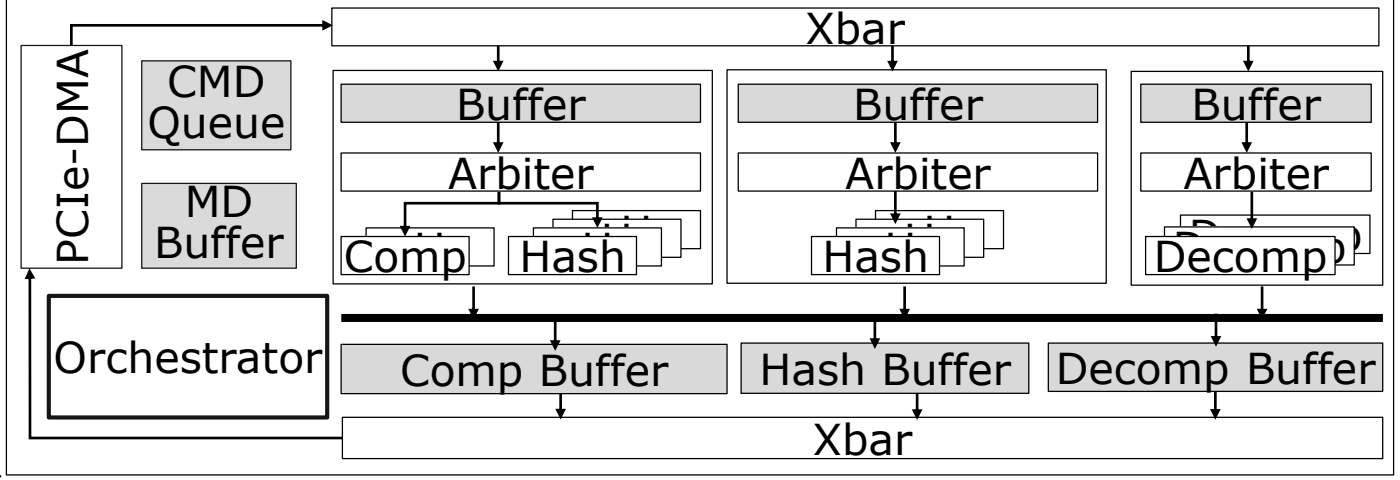
# CIDR Detailed System Architecture




## CIDR SW Support

- Buffer management
- Client request buffer
- Unique chunk predictor
- Opportunistic batch maker ↔ Delayed chunk buffer
- Data reduction table management ↔ Data reduction tables
- Chunk store management

## CIDR HW Engine




**We prototyped an example CIDR system on a real machine!**

# Index

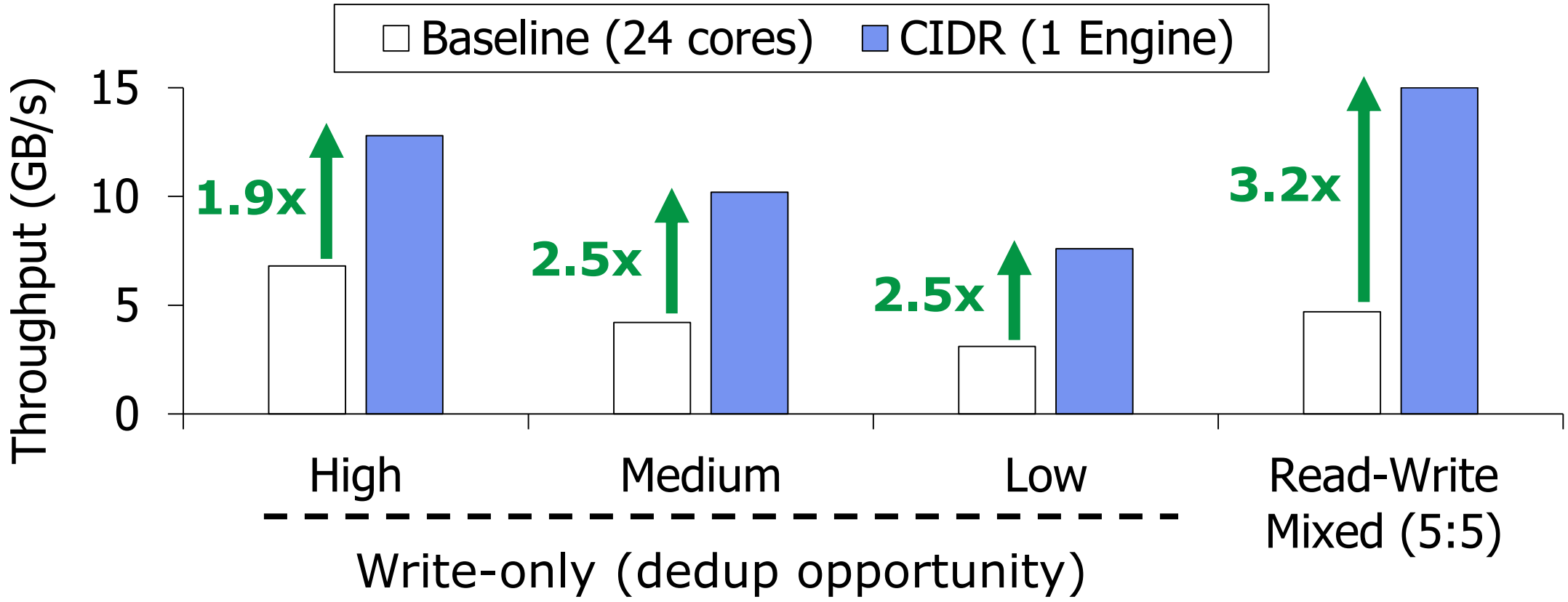
- Motivation
- CIDR: A Cost-Effective In-line Data Reduction
  - Key ideas
  - Architecture details
- **Evaluation**
- Conclusion

# Evaluation Methodology

- **Baseline: optimized SW in the Linux kernel**
  - Highly optimized SW using Intel ISA-L and SW pipelining
- **2 CIDR HW Engines and dual socket CPUs**
  - VCU9P FPGA (less than 33% LUTs utilized)
- **Workloads with different characteristics**
  - 3 write-only and 1 mixed read/write workloads generated by Vdbench
  - Real IO traces for dynamic workload behavior (refer to the paper)

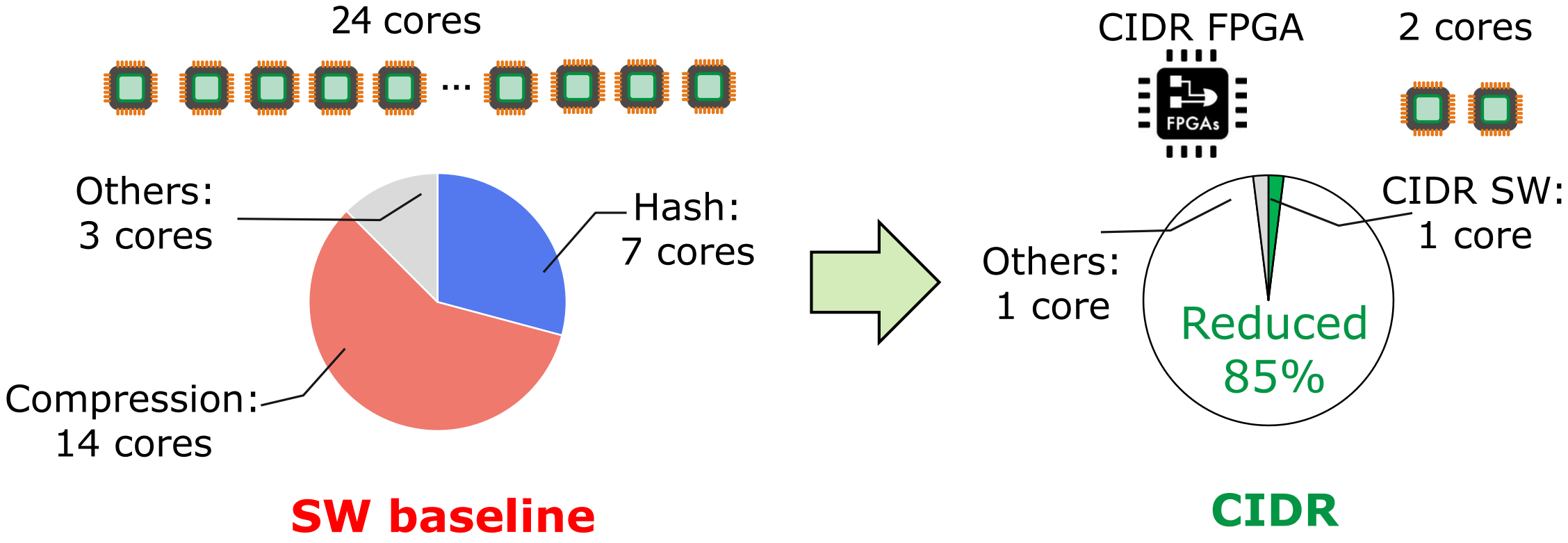
# CIDR's High Throughput (Single FPGA)

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



# CIDR's Low CPU Utilization

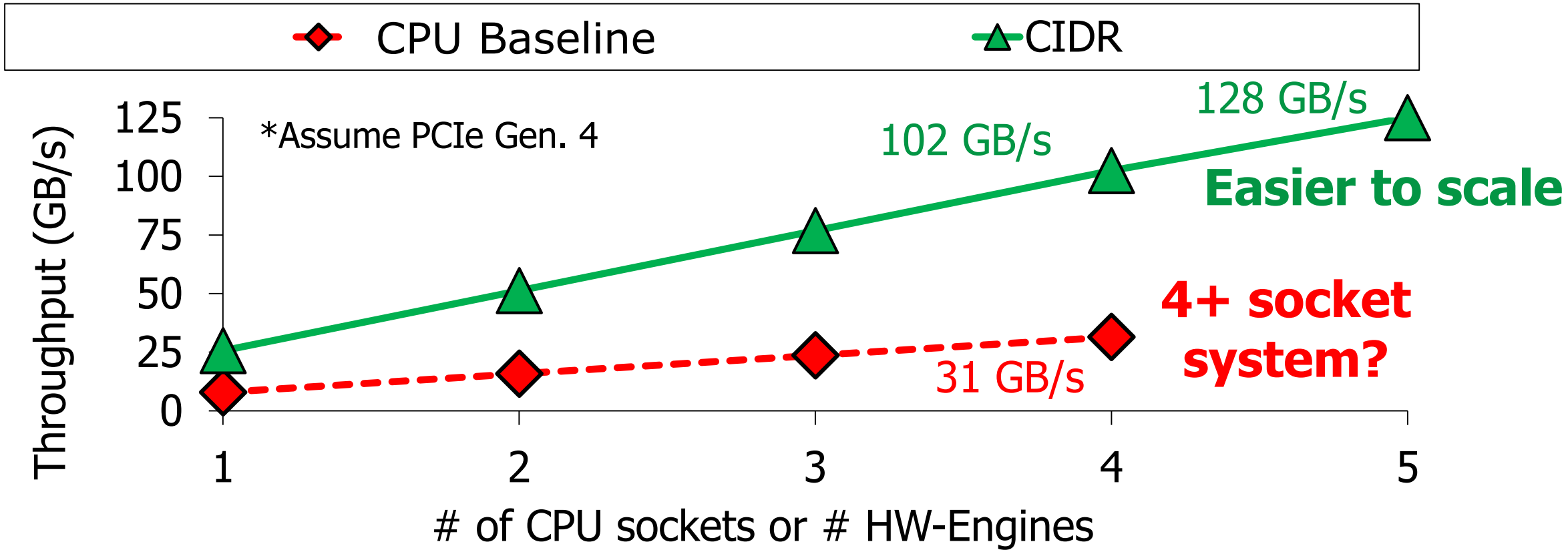
- **Comparison at the same throughput**




**Enables extreme throughput scalability**

# CIDR's High Throughput Scalability

- **Scalable FPGA array for higher throughput**



# Conclusion

- **CIDR co-optimizes SW/HW for efficient reduction**
  - Array of scalable PCIe-attached FPGAs
  - Centralized metadata management on CPUs
  - Novel SW/HW orchestration mechanism
- **CIDR significantly outperforms the CPU baseline**
  - Scalable performance for **1+ Tbps** with multiple HW-Engines
  - **85+ %** CPU utilization reduction
- **More practical details are in the paper**