
重庆大学
CHONGQING UNIVERSITY

香港城市大學
City University
of Hong Kong

# Exploiting Parallelism in I/O Scheduling for Access Conflict Minimization in Flash-based Solid State Drives

Congming Gao, **Liang Shi**, Mengying Zhao, Chun Jason Xue, Kaijie Wu, Edwin H.-M.Sha

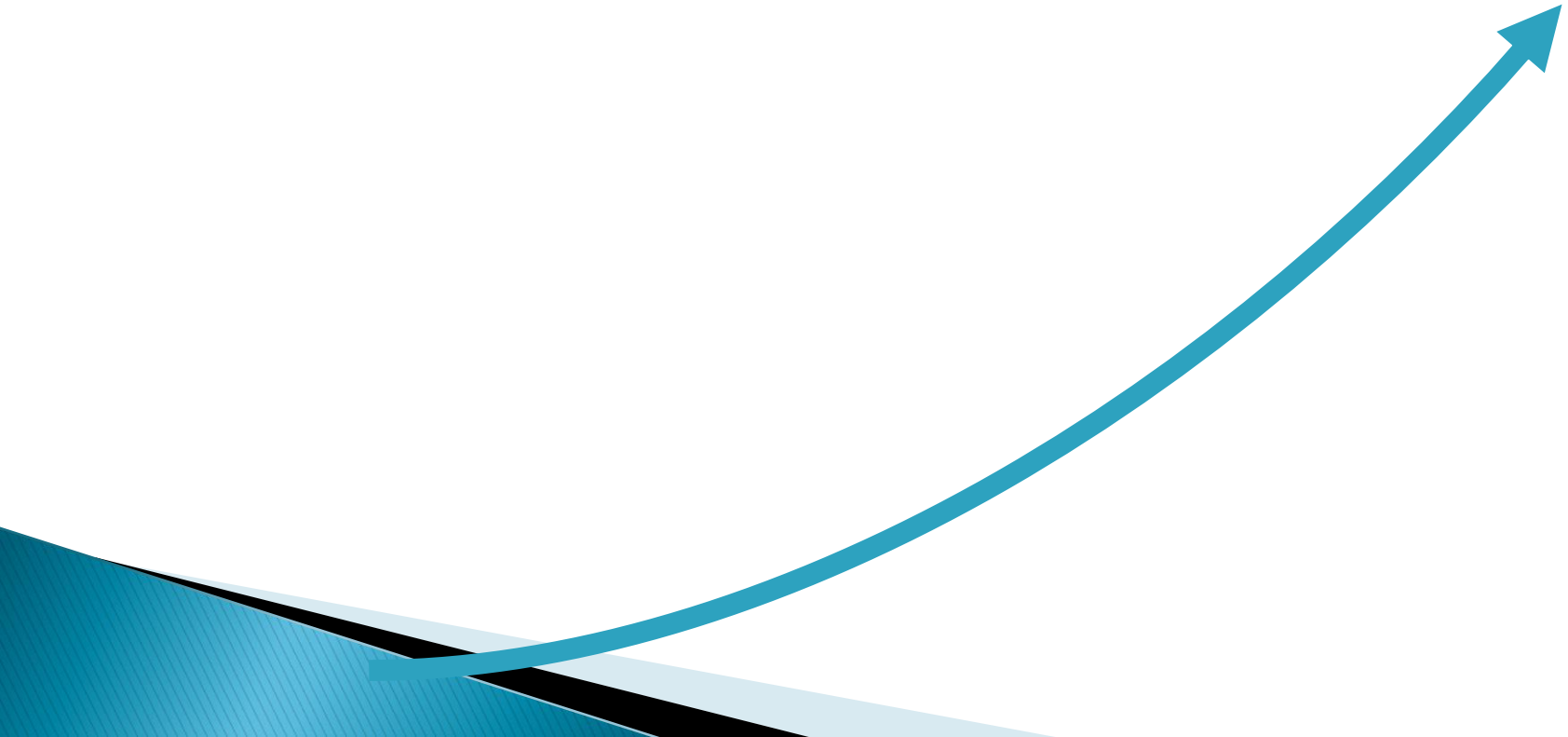
**Chongqing University, China**

**City University of Hong Kong, Kowloon, Hong Kong**

# Outline

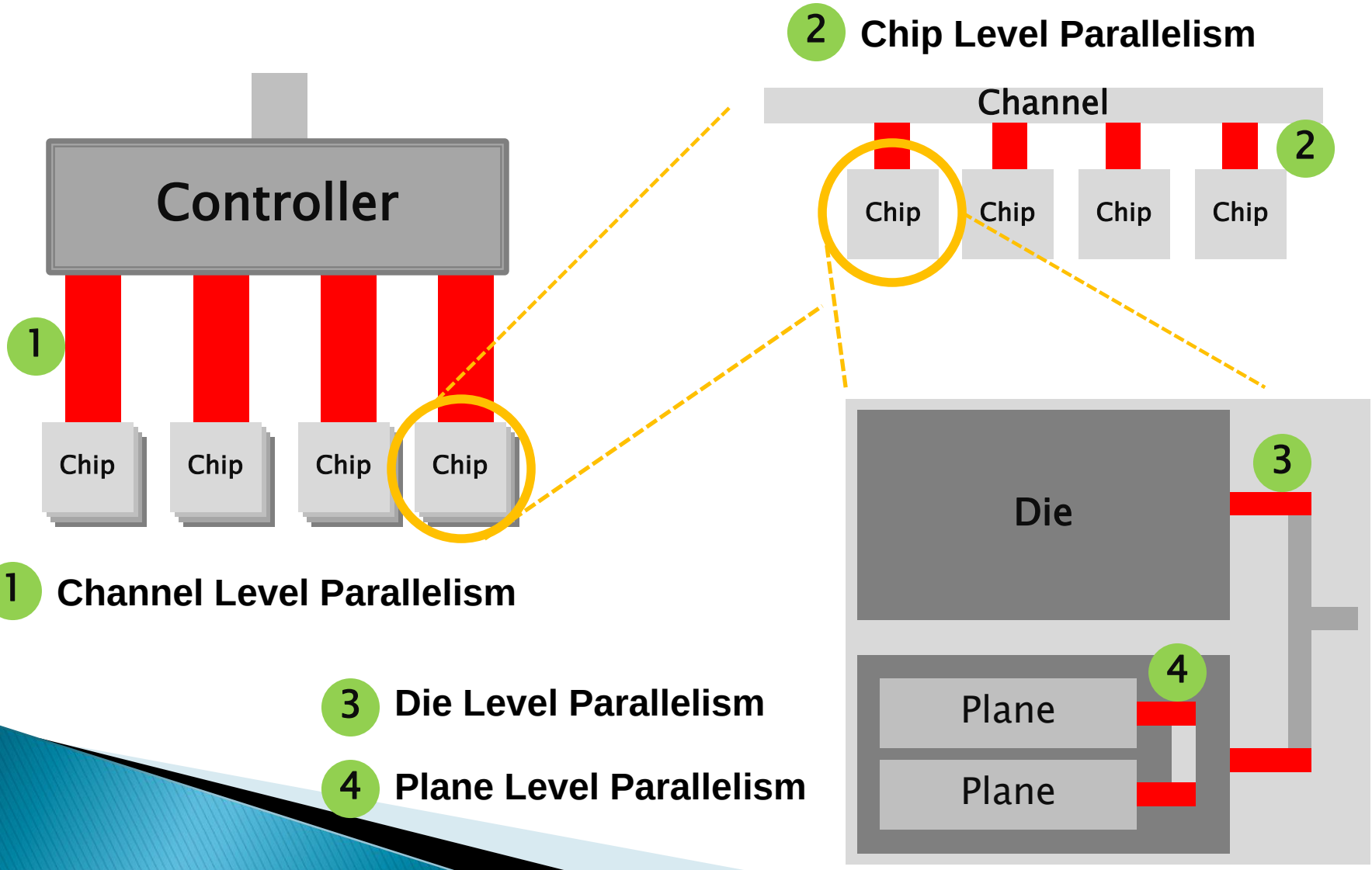
- **Background and Related Work**
- Parallel Issue Queuing For Parallelism Exploration
  - Access Conflict Detection
  - Parallel Issue Queuing (PIQ)
- Experiment And Analysis
- Conclusions

# Flash Development

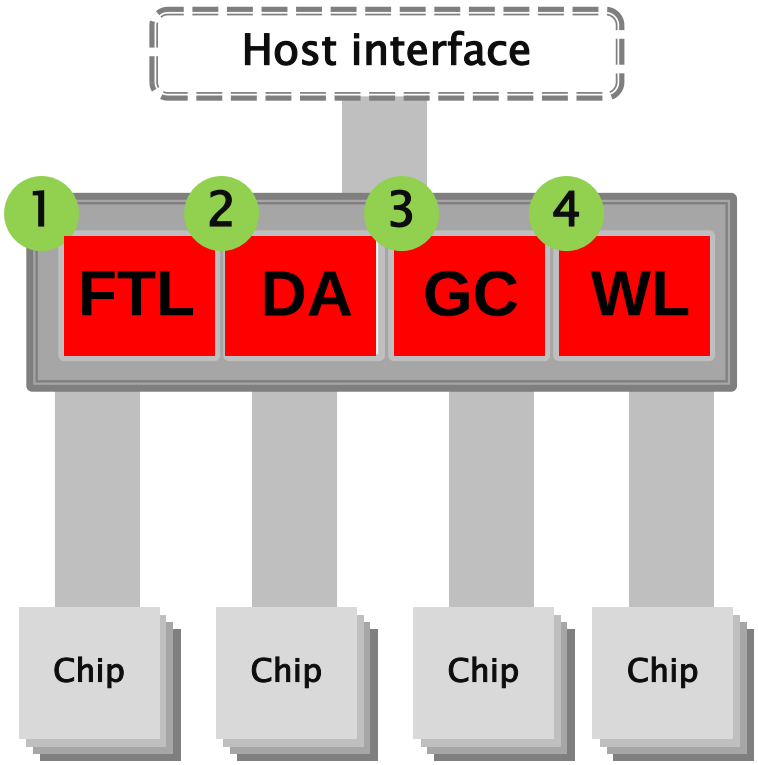


Performance Improvement

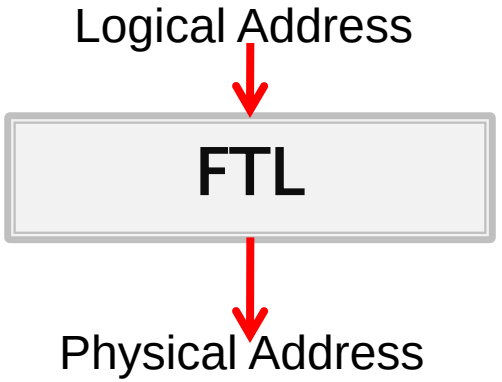
# Parallel Organization



# Controller Design

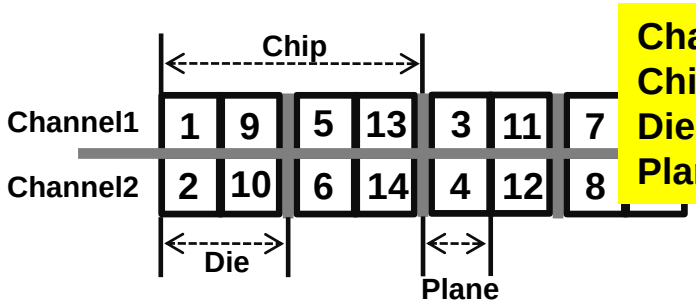



1




2

Data Allocation




**Channel First**
**Chip Second**
**Die Third**
**Plane Last**

[**Jung *et al. USENIX HotStorage'12***]

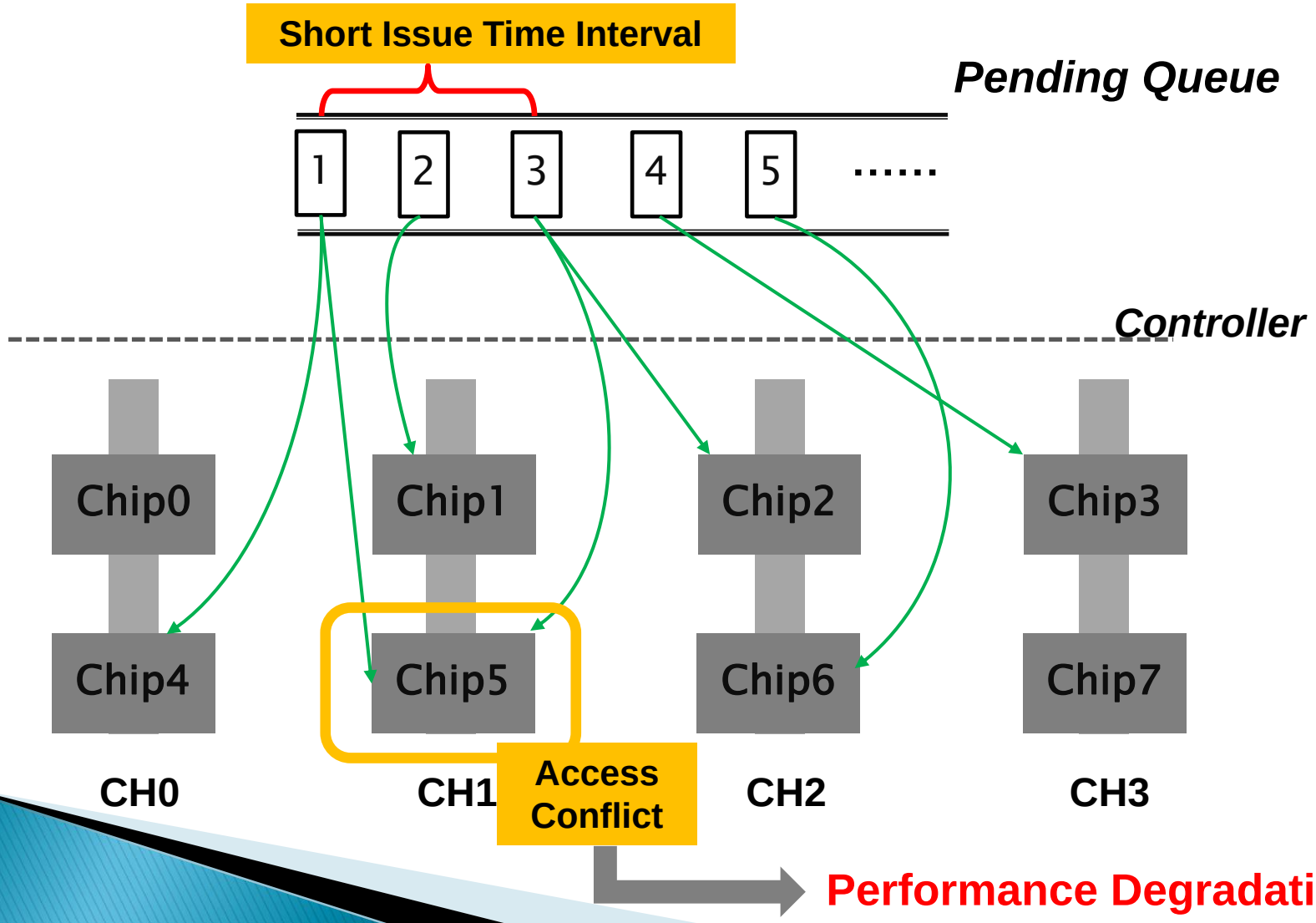
3

4

**Gabage Collection collects the invalid pages;**
**Wear leveling prolongs the flash lifespan**

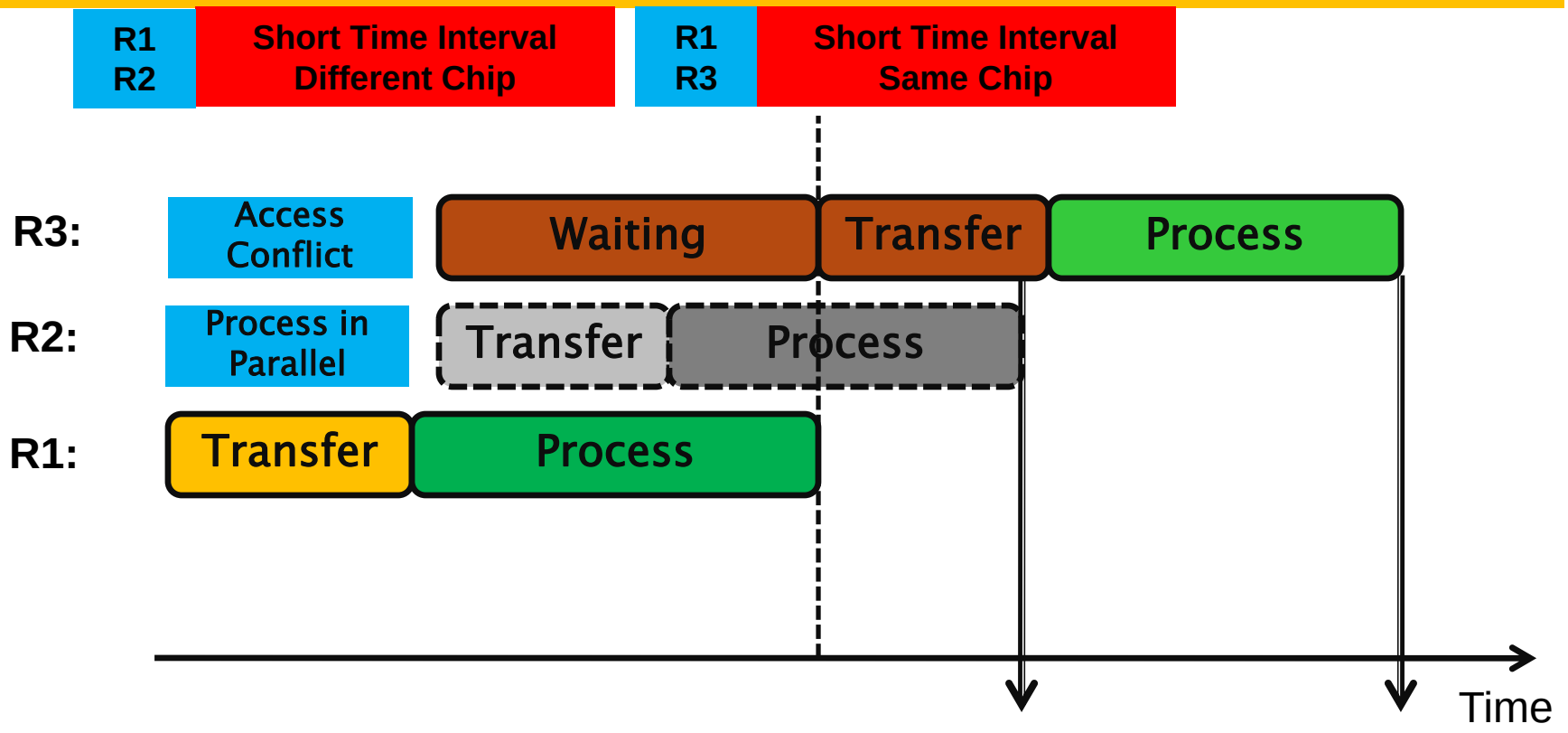
# Traditional I/O Scheduler

**NOOP**



# What is the access conflict?

**Access conflict is highly correlated with the issue time of I/O requests and location of data(Chip Level).**

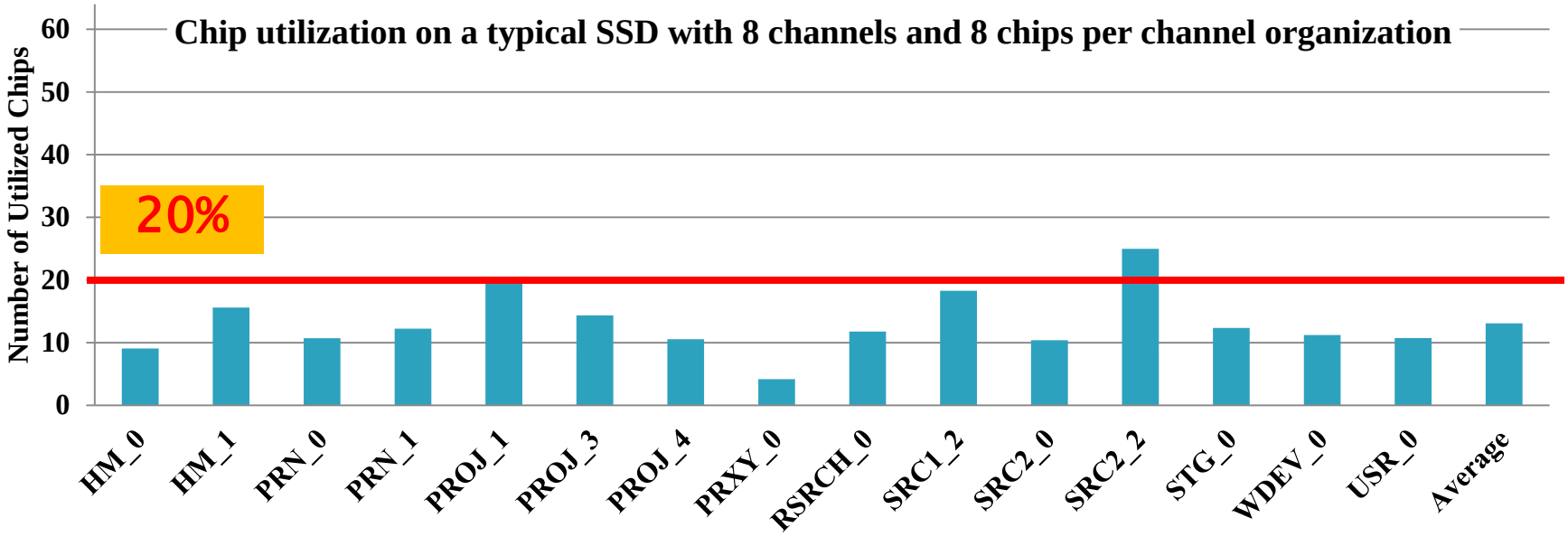



**Performance Degradation!!!**

# Motivation

**NOOP**

The chip utilizations of the various benchmarks are mostly below **20%**

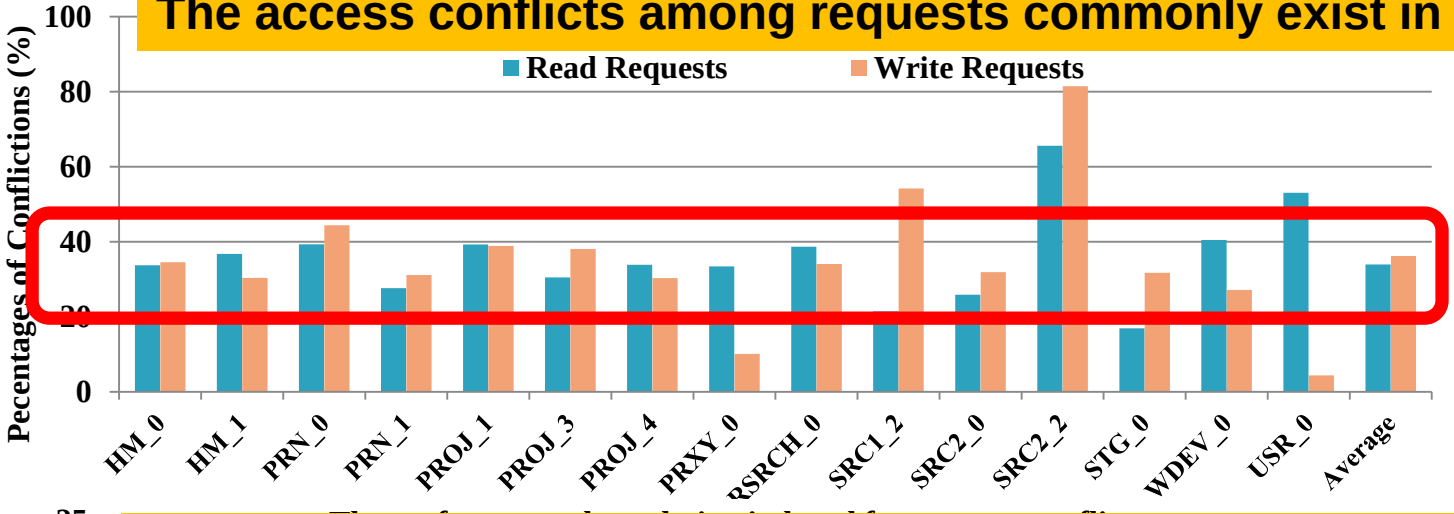



**Poor Chip Utilization!!!**

# Motivation

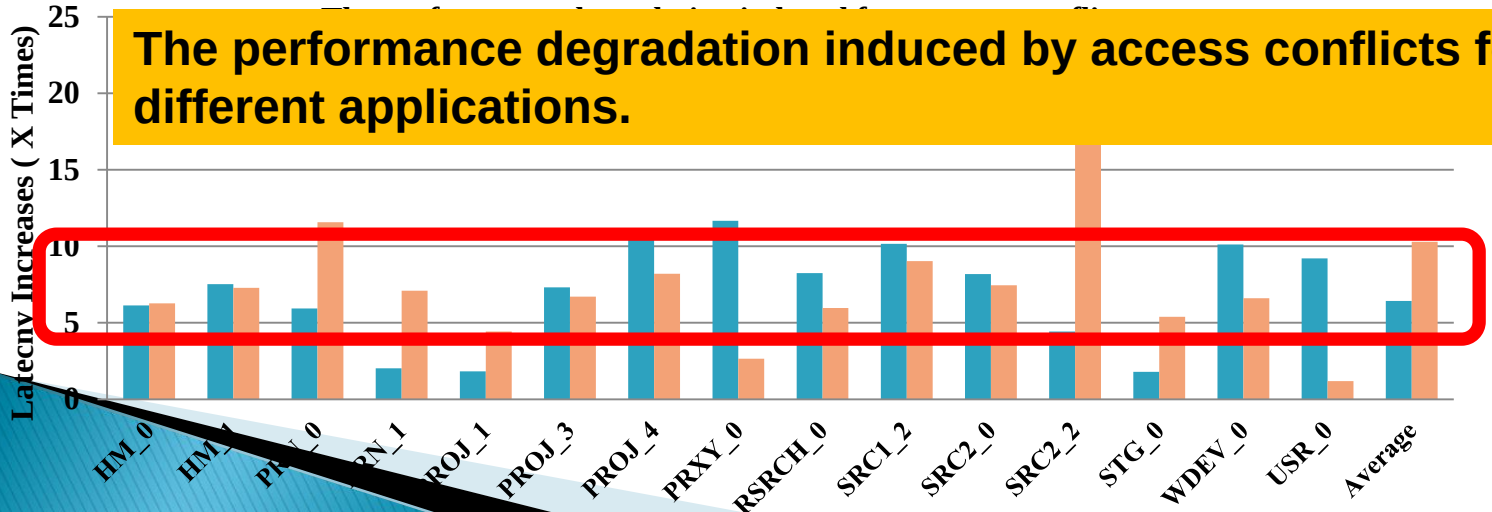
**NOOP**

**The access conflicts among requests commonly exist in applications**




**Matching Pattern**

**The performance degradation induced by access conflicts for different applications.**



# Related Work

## Exploration of Parallelism

1. Seol *et al.* [4] and Park *et al.* [3] proposed to exploit the multi-channels of SSDs from the view of write buffers to improve the write performance.

But, They did not solve the access conflict problem.

2. Jung *et al.* [2] proposed a read resource contention aware approach, physical address queuing under FTL, to issue more I/O requests to the SSDs.

But, PAQ requires hardware modification and only solves the read conflict problem.

3. Chen *et al.* [1] first proposed a buffer cache management approach for SSDs to solve the read conflict problem by exploiting the read parallelism of SSDs.

But, Their works are only able to solve the conflict problem when the conflicts have taken place. Moreover, their works only solve the read conflict problem.


[1] Z. Chen, N. Xiao, and F. Liu. Sac: rethinking the cache replacement policy for ssd-based storage systems. In SYSTOR'12, pages 13:1–13:12, 2012.

[2] M. Jung, E. H. Wilson, III, and M. Kandemir. Physically addressed queueing (paq): improving parallelism in solid state disks. In ISCA'12, pages 404–415, 2012.

[3] S. K. Park, Y. Park, G. Shim, and K. H. Park. Cave: Channel-aware buffer management scheme for solid state disk. In SAC'11, pages 346–353, 2011.

[4] J. Seol, H. Shim, J. Kim, and S. Maeng. A buffer replacement algorithm exploiting multi-chip parallelism in solid state disks. In CASES'09, pages 137–146, 2009.

# Related Work

## Scheduler of SSD

1. Traditional I/O schedulers for HDD include NOOP, Deadline, Anticipate, and Completely Fair Queuing (CFQ) [4].

But, none of them work efficiently on SSDs [1].

2.Kim *et al.*[1] first proposed an I/O scheduler for SSDs with the awareness of read/write interferences. They proposed to bundle write requests and schedule read requests first to reduce the impact of slow write operations on read performance.

3. Park *et al.*[2] and Shen *et al.*[3] proposed two schedulers to achieve fairness among multi-tasks on SSDs.

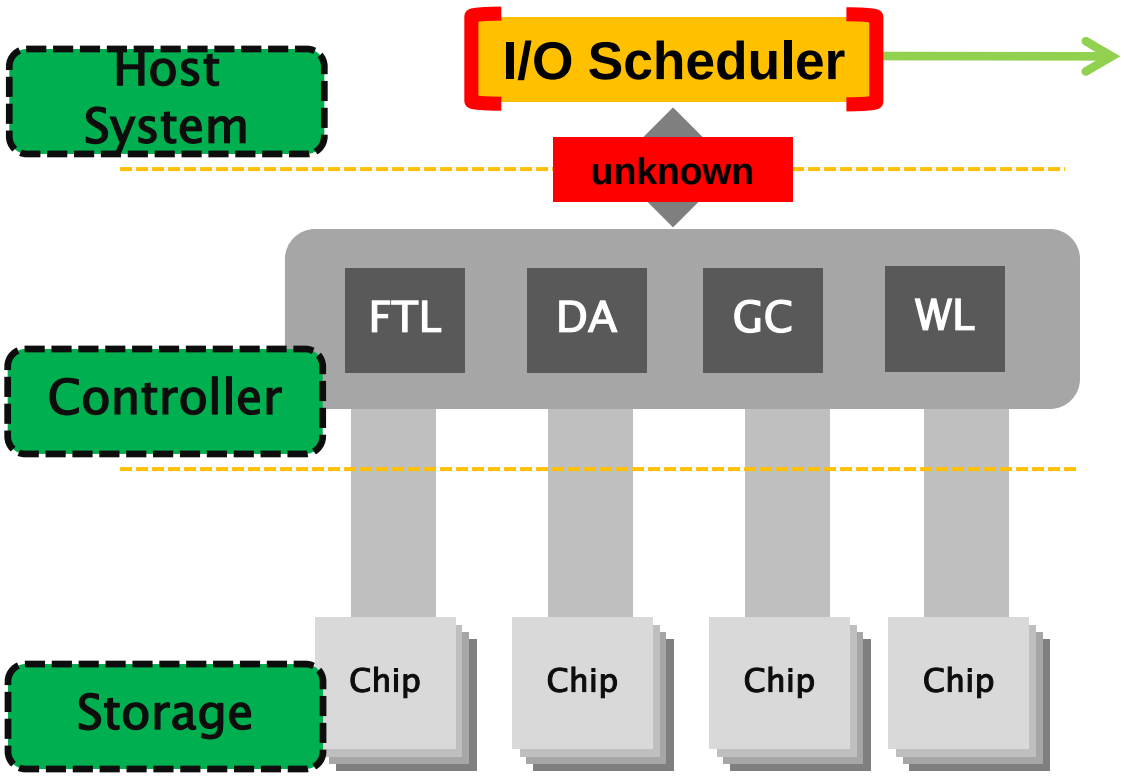
But, Their works are proposed to improve the performance of SSDs from the view of fairness, none of their works proposed to solve the access conflict problem.

[1] J. Kim, Y. Oh, E. Kim, J. Choi, D. Lee, and S. H. Noh. Disk schedulers for solid state drivers. In EMSOFT'09, pages 295–304, 2009.
[2] S. Park and K. Shen. Fios: A fair, efficient flash i/o scheduler. In FAST'12, pages 13–13, 2012.
[3] K. Shen and S. Park. Flashfq: A fair queueing i/o scheduler for flash based ssds. In ATC'13, pages 67–78, 2013.
[4] A. S. Tanenbaum and A. Tannenbaum. Modern operating systems,
volume 2. Prentice hall Englewood Cliffs, 1992.

# Outline

- Background and Related Work
- **Parallel Issue Queuing For Parallelism Exploration**
  - **Access Conflict Detection**
  - Parallel Issue Queuing (PIQ)
- Experiment And Analysis
- Conclusions

# Main idea of PIQ




physical address of data in SSDs is unknown to the host system.

Location of Request Data

Access Conflict Detection

Parallel Issue Queuing(PIQ)

# Access Conflict Detection

**How to get location of data?**

**Channel First**
**Chip Second**
**Die Third**
**Plane Last**

1. **Logical Sector Number(LSN)**
2. **Data Allocation Scheme**[1][2][3][4]

$$chip\_first = \left( \left\lfloor \frac{LSN \times SS}{Page\_Size} \right\rfloor \right) \% N_C$$

$$chip\_last = \left( \left\lceil \frac{(LSN + Sector\_Size) \times SS}{Page\_Size} \right\rceil - 1 \right) \% N_C$$

**Assume:**

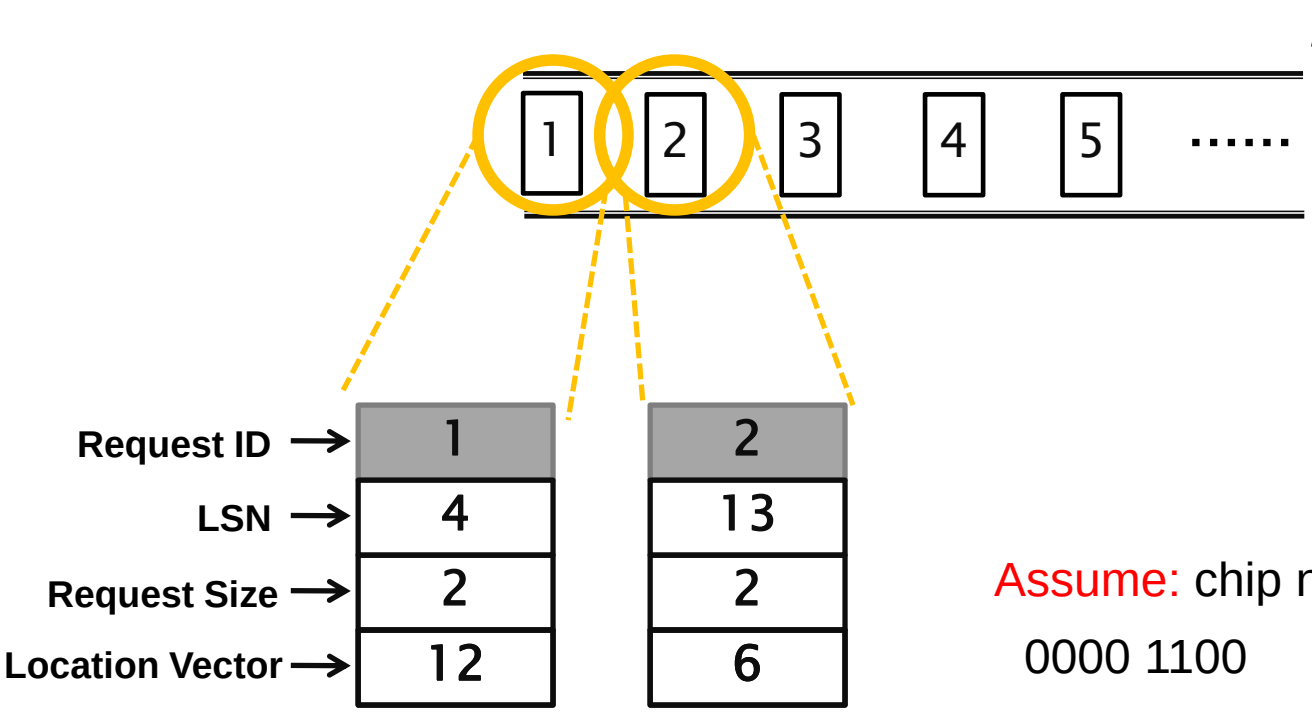
Request=(Type,LSN,Sector_Number,T)=(r,14854,2,0.02332);
SS=512B,Page_Size=4KB,$N_C$=8.

**chip_first=0,chip_last=1;** → **Location Vector** → **V($R_i$)=(11000000)=192**


[1] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren. Exploring and exploiting the multilevel parallelism inside ssds for improved performance and endurance. IEEE Transactions on Computers, 62(6):1141–1155, 2013.
[2] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang. Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity. In ICS'11, pages 96–107, 2011.
[3] M. Jung and M. Kandemir. An evaluation of different page allocation strategies on high-speed ssds. In FAST'12, pages 9–9, 2012.
[4] J.-Y. Shin, Z.-L. Xia, N.-Y. Xu, R. Gao, X.-F. Cai, S. Maeng, and F.-H. Hsu. Ftl design exploration in reconfigurable high-performance ssd for server applications. In ICS'09, pages 338–349, 2009.

# Access Conflict Detection

## Conflict Detection




Assume: chip number=8

0000 1100

0000 0110

---

*Conflict* = & = **0000 0100**

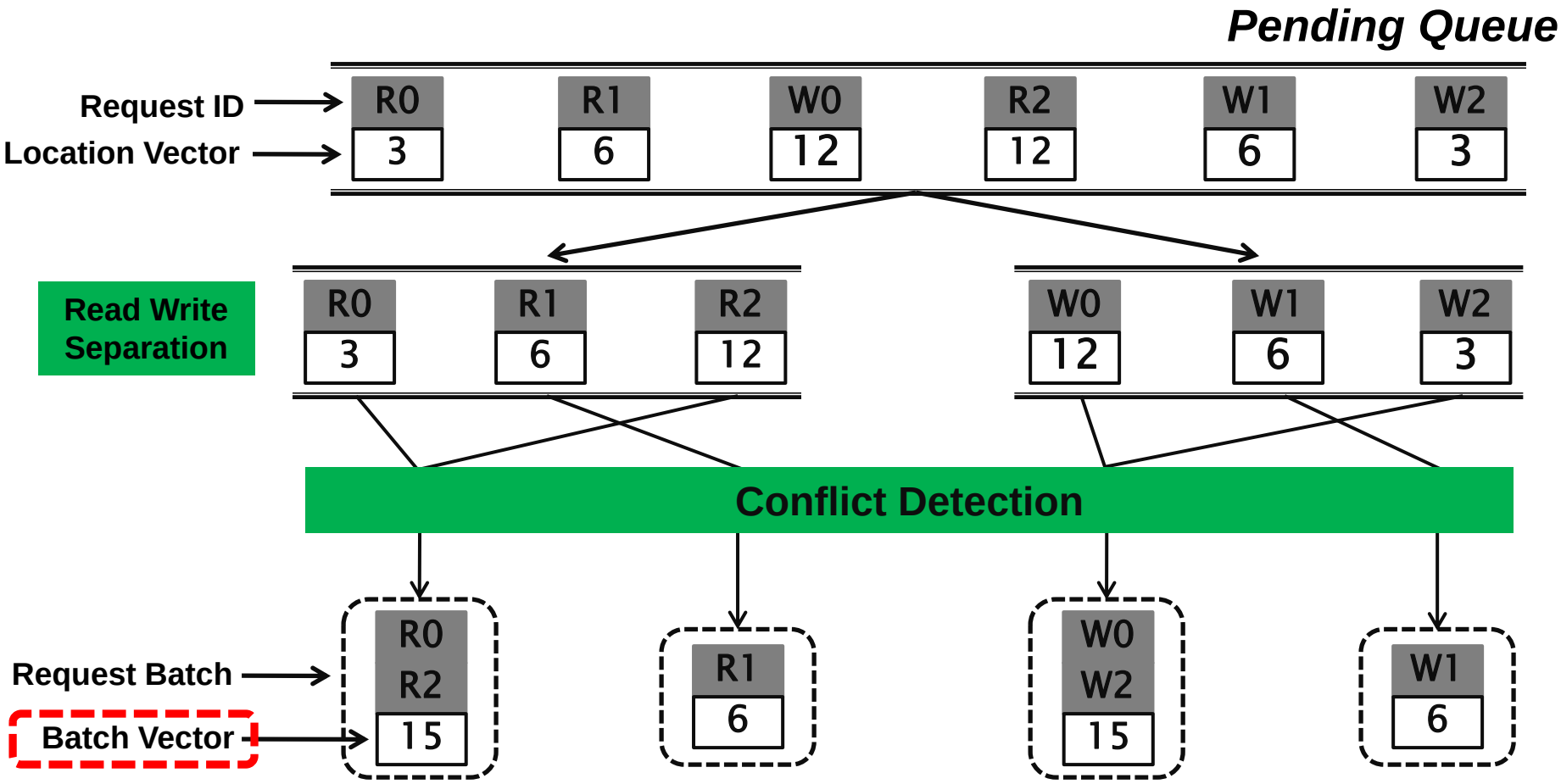
## Conflict Detection

**Access Conflict**

# Outline

- Background and Related Work
- **Parallel Issue Queuing For Parallelism Exploration**
  - Access Conflict Detection
  - **Parallel Issue Queuing (PIQ)**
- Experiment And Analysis
- Conclusions

# Parallel Issue Queuing(PIQ)



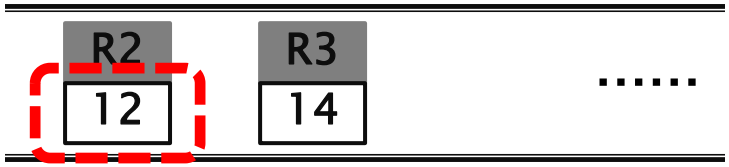

?

**All requests in a batch can be serviced in parallel**
**Batches are issued in the order of creation time**

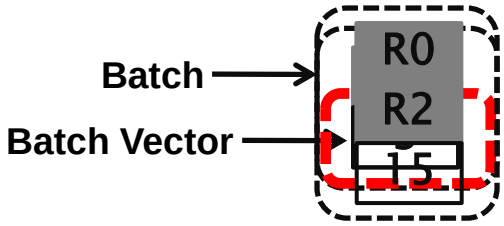
# Parallel Issue Queuing(PIQ)

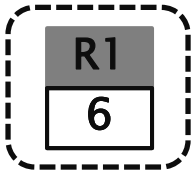
## Batch Conflict Detection

***Read Pending Queue***




## Update Batch




Assume: chip number=8

0000 0011

0000 1100

*Conflict* = 3 & 12 = **0000 0000**

**No Access Conflict**

*New Batch Vector* = 3 | 12 = **0000 1111 = 15**

# Outline

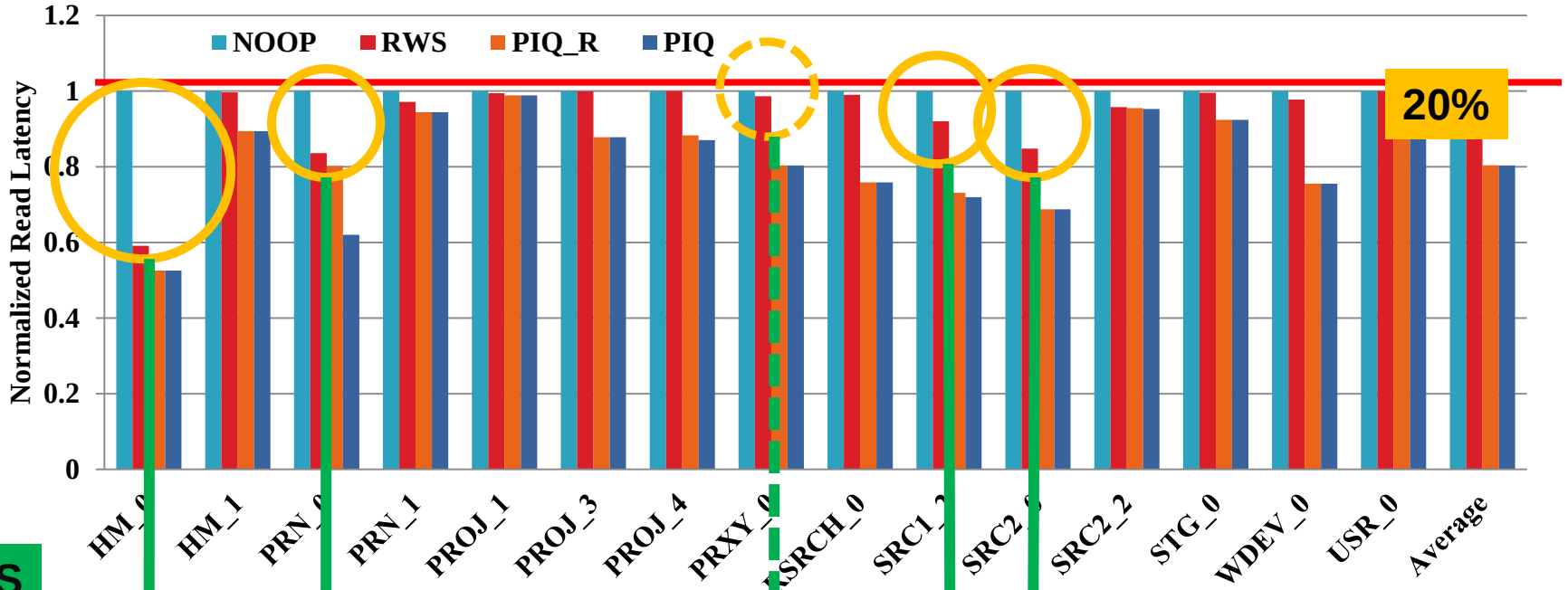
- Background and Related Work
- Parallel Issue Queuing For Parallelism Exploration
  - Access Conflict Detection
  - Parallel Issue Queuing (PIQ)
- **Experiment And Analysis**
- Conclusions

# Experiment Setup

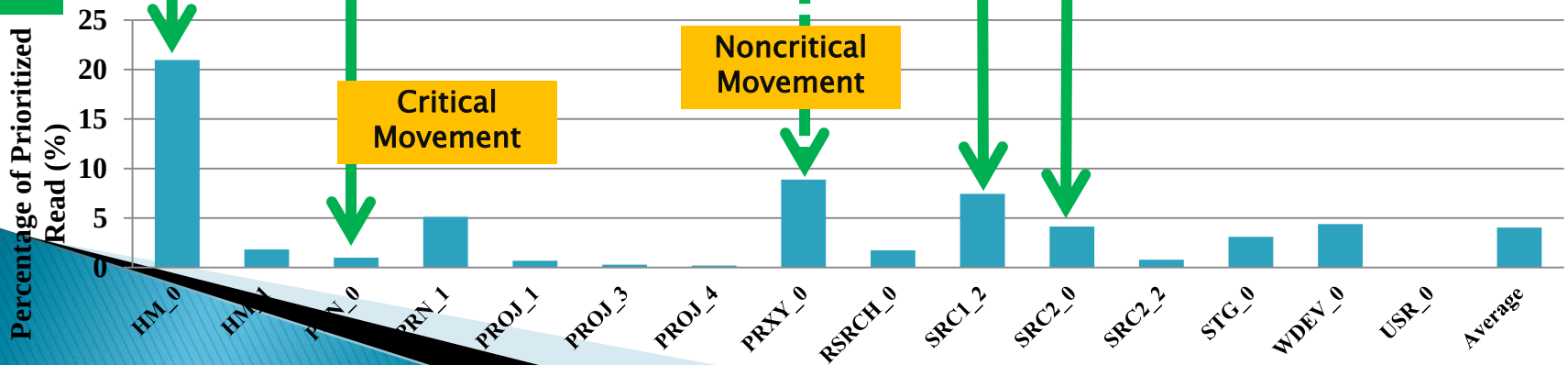
- A trace drive simulator is used to verify the proposed framework.
- Traces include a set of carefully selected MSR Cambridge traces from servers.
- Comparison among: NOOP, RWS, PIQ_R, PIQ_W, PIQ
  - NOOP: Traditional I/O Scheduler
  - RWS: read requests and write requests separated into two sub queues.
  - PIQ_R: read  requests separated into batches.
  - PIQ_W: write requests separated into batches.
  - PIQ: Our proposed I/O scheduler.

# Experiment and Analysis

## Read Performance

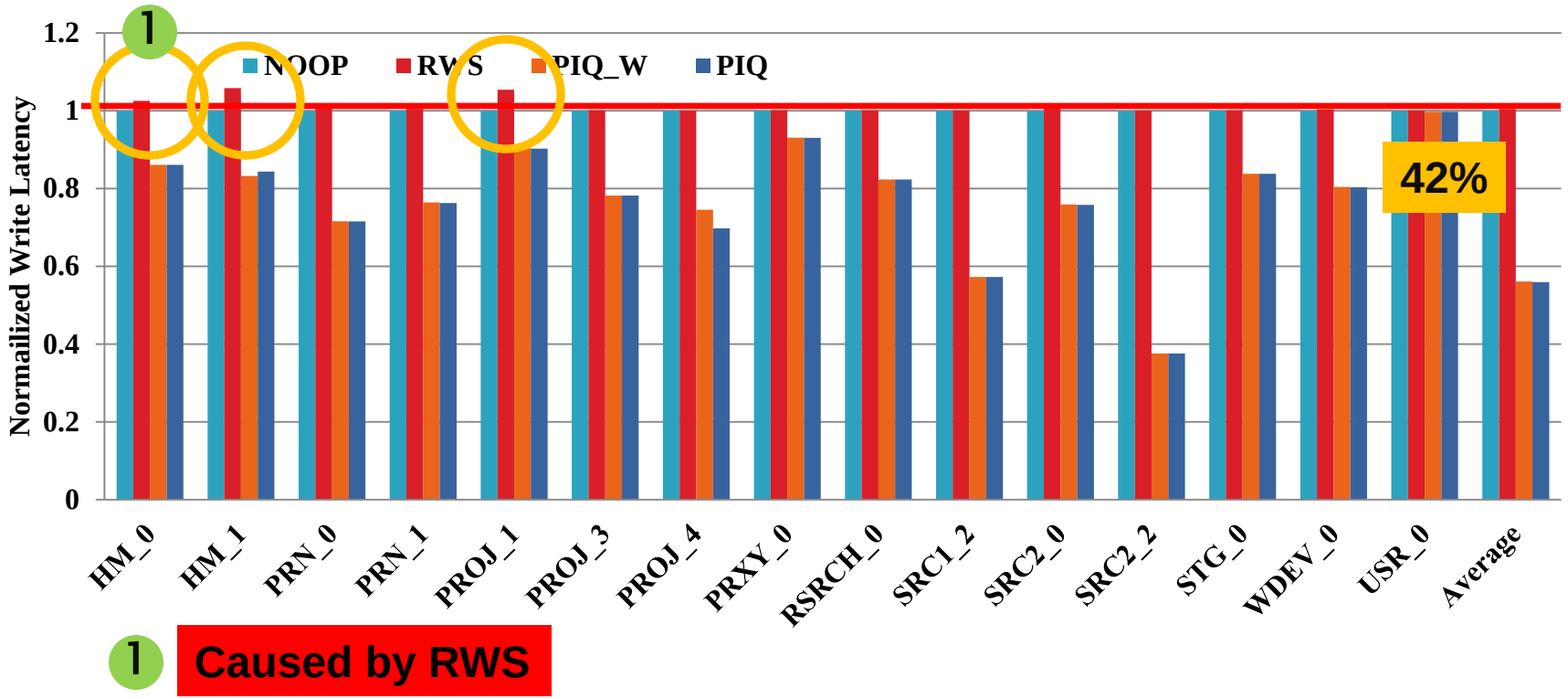



RWS



# Experiment and Analysis

## Write Performance

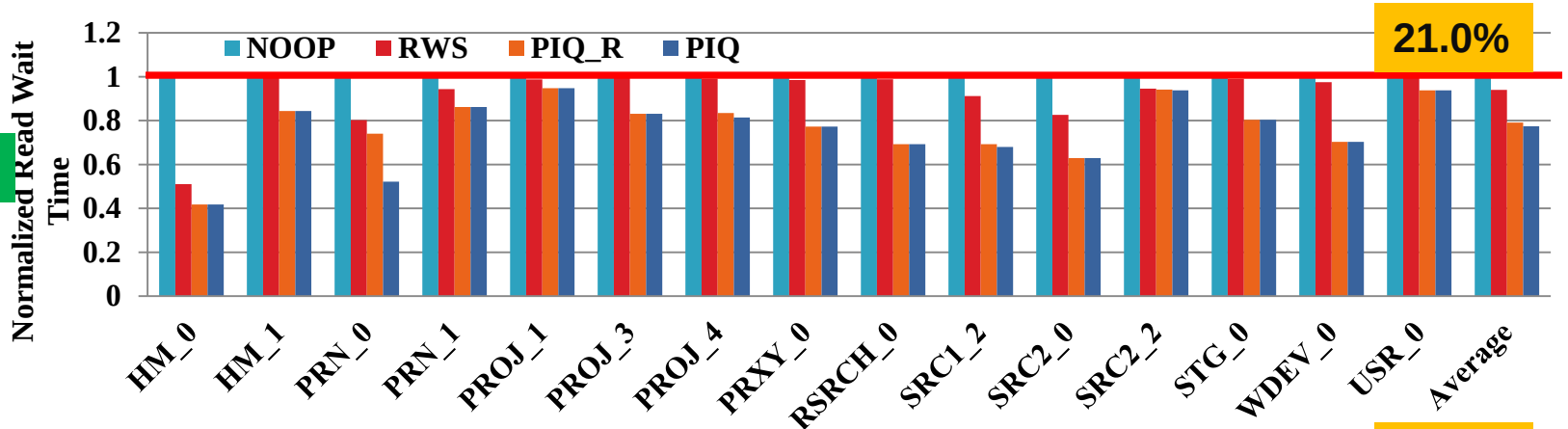



1 **Caused by RWS**

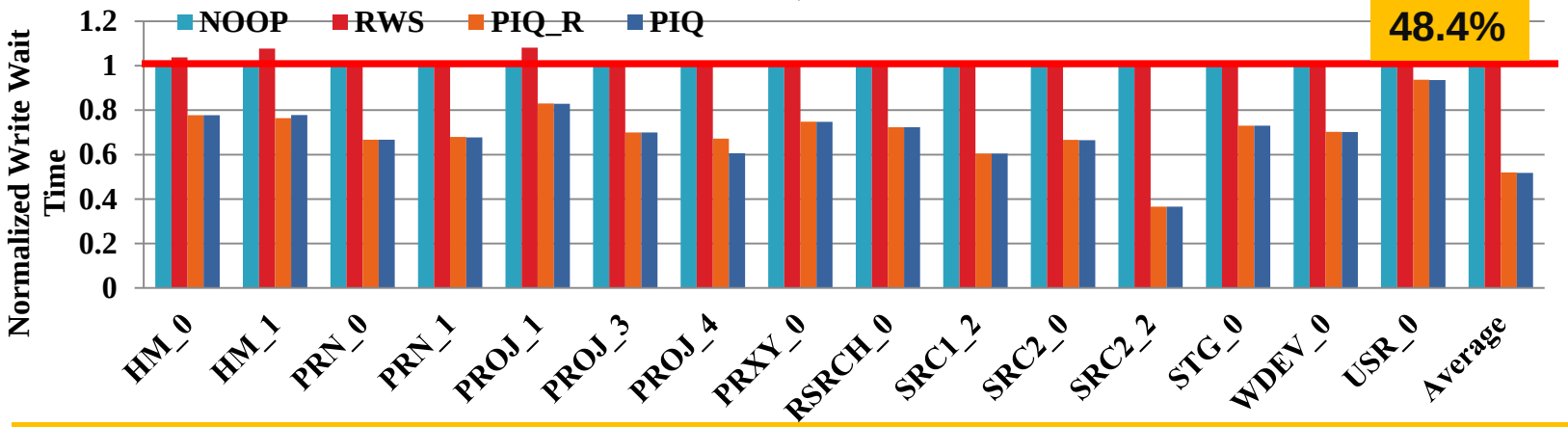
# Experiment and Analysis

## Waiting Time

**Read**



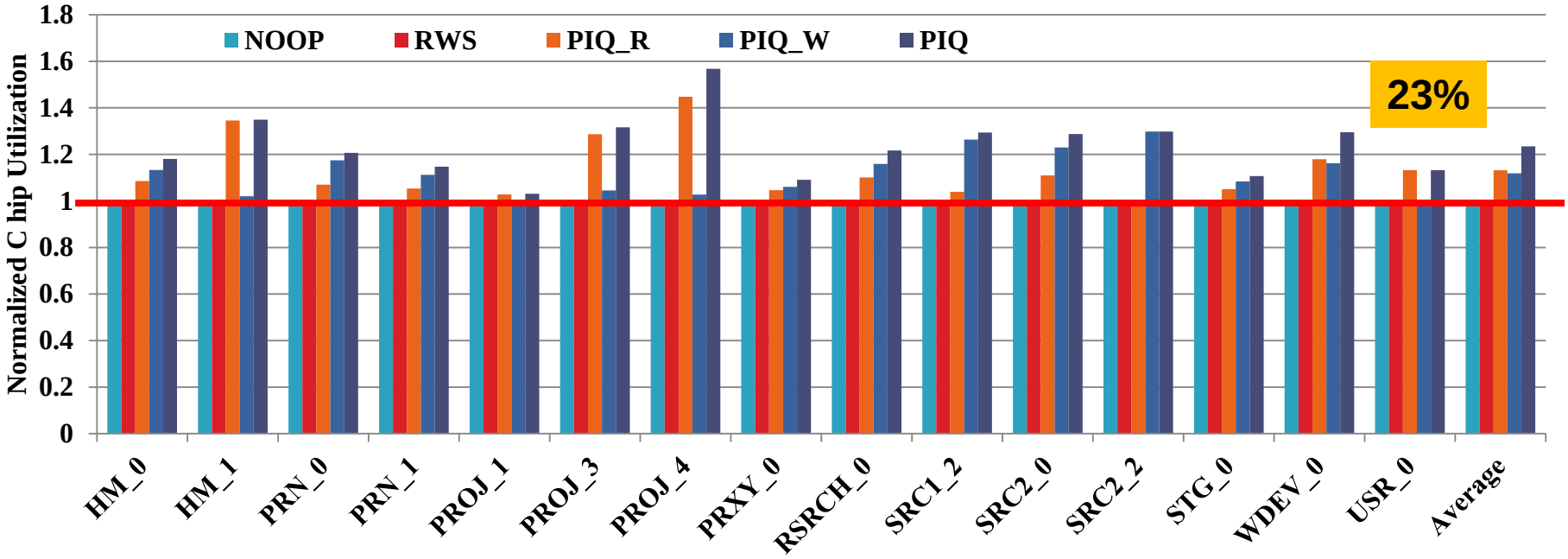

**Write**




The performance improvement is achieved by the reduction of waiting time!

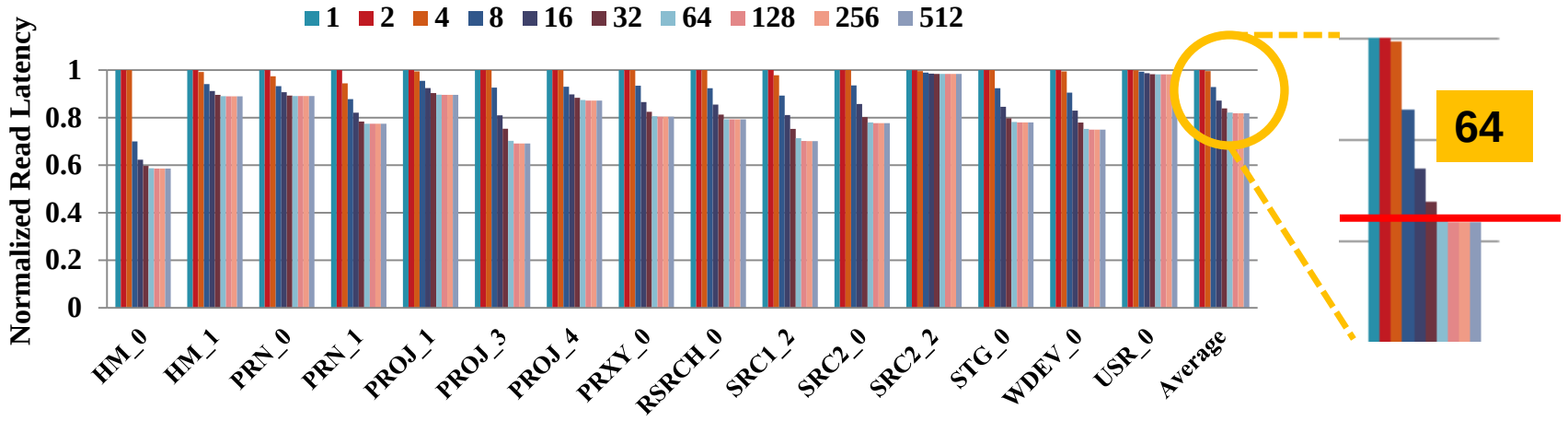
# Experiment and Analysis

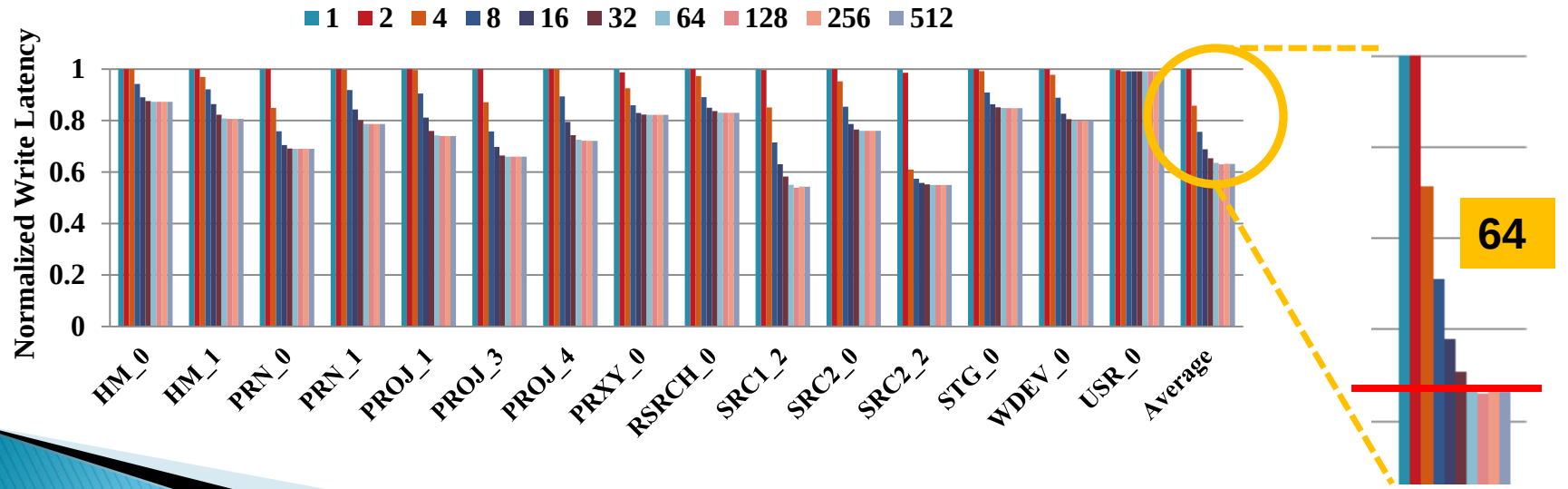
## Chip Utilization



# Experiment and Analysis

**Sensitive Studies** | **I/O queue length is varied from 1 to 512**



# Outline

- Background and Related Work
- Parallel Issue Queuing For Parallelism Exploration
  - Access Conflict Detection
  - Parallel Issue Queuing (PIQ)
- Experiment And Analysis
- **Conclusions**

# Conclusions

- Proposed an I/O scheduler which is designed to reduce the access conflicts of SSDs.
  - Proposed an access conflict detection approach.
  - Proposed a parallel issue queuing approach to exploit the parallelism of SSDs.

- Experiment results show that the proposed approach is very efficient in performance improvement.

# Thank you

Questions?