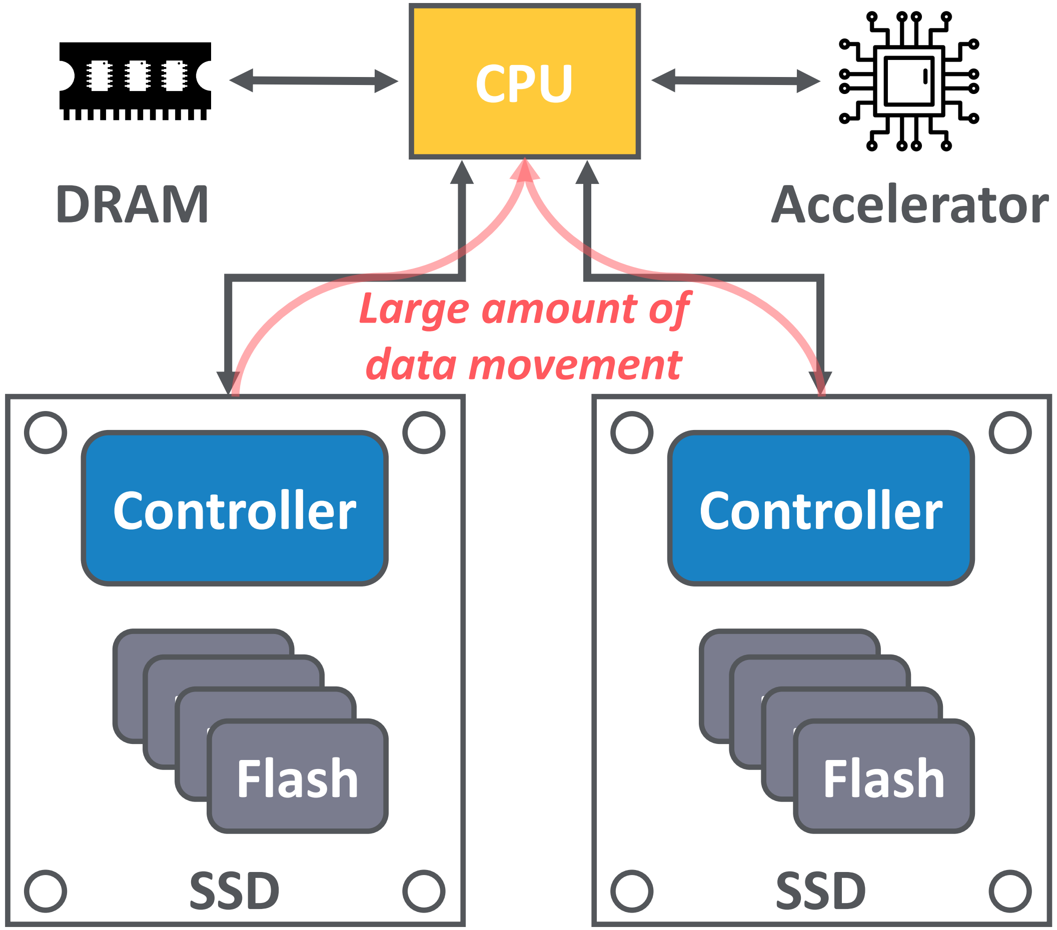
PiF: In-Flash Acceleration for Data-Intensive Applications

Myoungjun Chun[1], Jaeyong Lee[1], Sanggu Lee[1], Myungsuk Kim[2], and Jihong Kim[1]

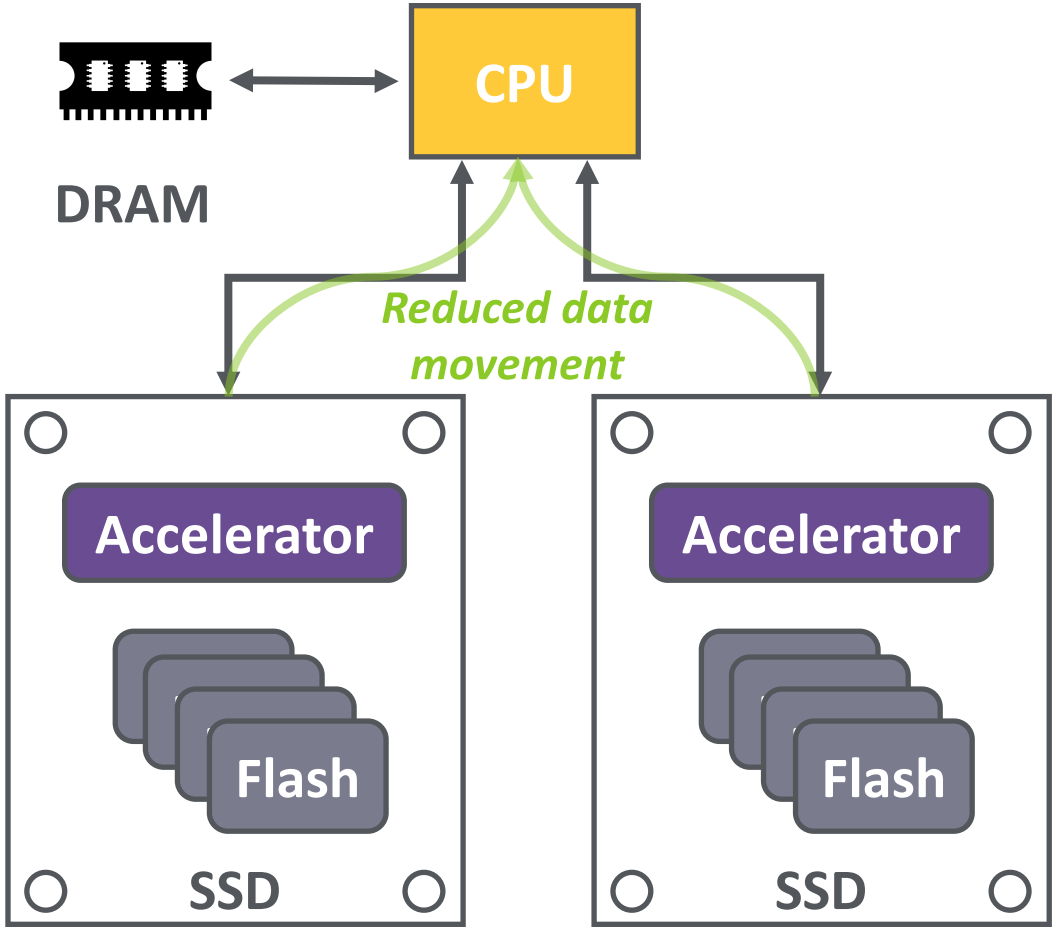
[1]Seoul National University, [2]Kyungpook National University

The 14th ACM Workshop on Hot Topics in Storage and File Systems
June 28, 2022

Processing-in-Storage Architectures

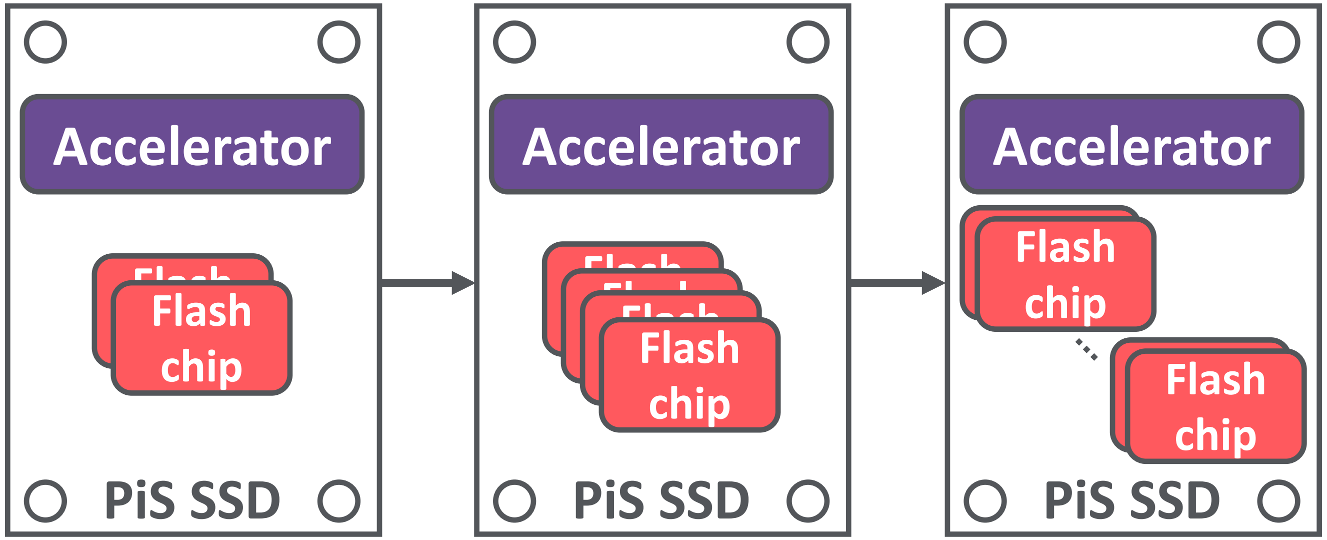


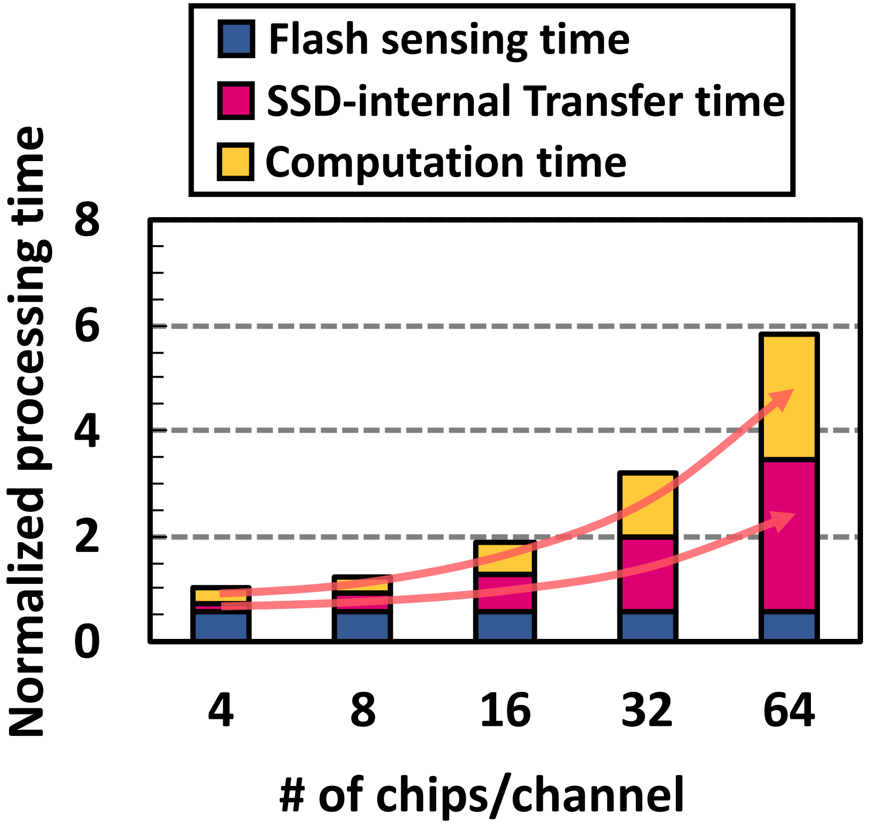
Traditional architecture



Processing-in-Storage (PiS) architecture

Limitations of the PiS Technique





⚠ Large data movements from flash to an accelerator

⚠ Unscalable acceleration capability over the number of flash chips

Outline

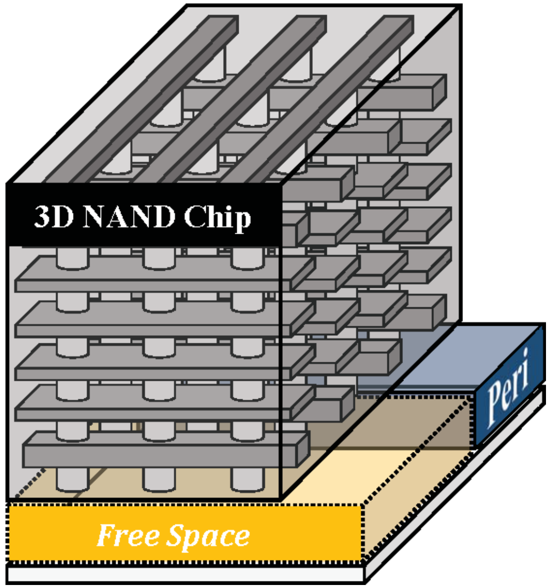
001 Processing-in-Flash Architectures and Challenges

002 Use case: A Pattern Matching Enabled PiF Architecture

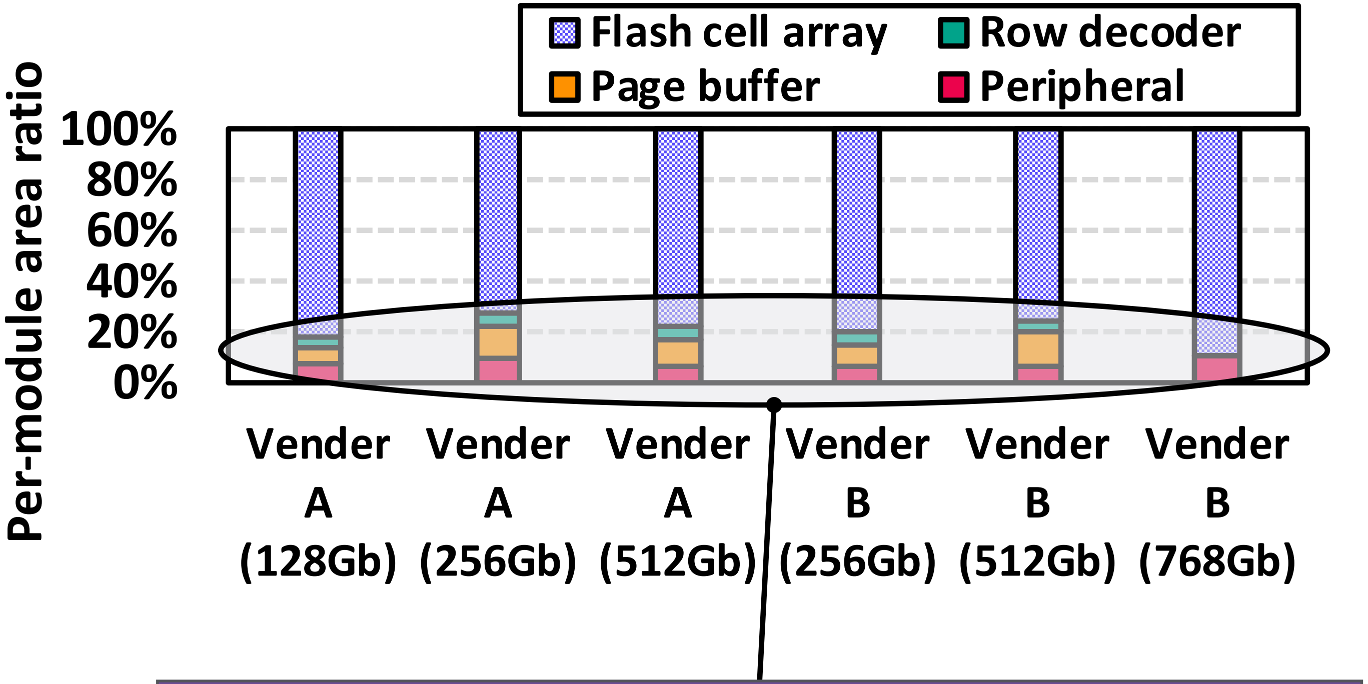
003 Evaluation & Conclusion

An Opportunity for In-Flash Processing

- Exploit the free space of a CoP-based flash chip



A Cell-over-Peri (CoP) flash chip [1]

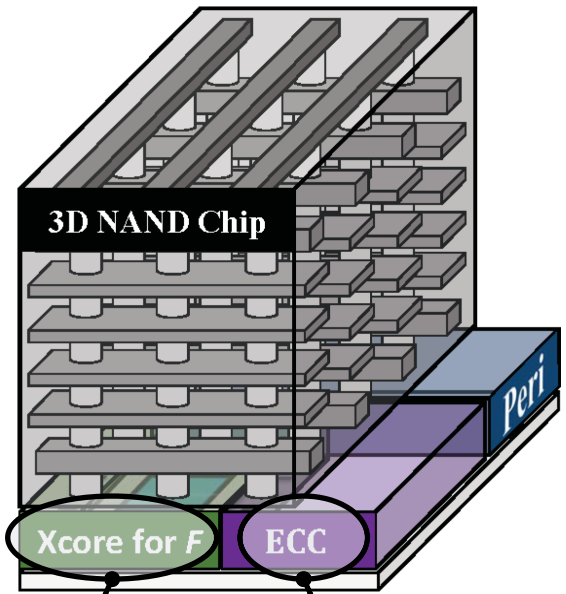


78% usable space at the bottom of CoP chip

[1] J Park, et al. 30.1 A 176-Stacked 512Gb 3b/Cell 3D NAND Flash with 10.8Gb/mm2 Density with a Peripheral Circuit Under Cell Array Architecture. In ISSCC, 2021

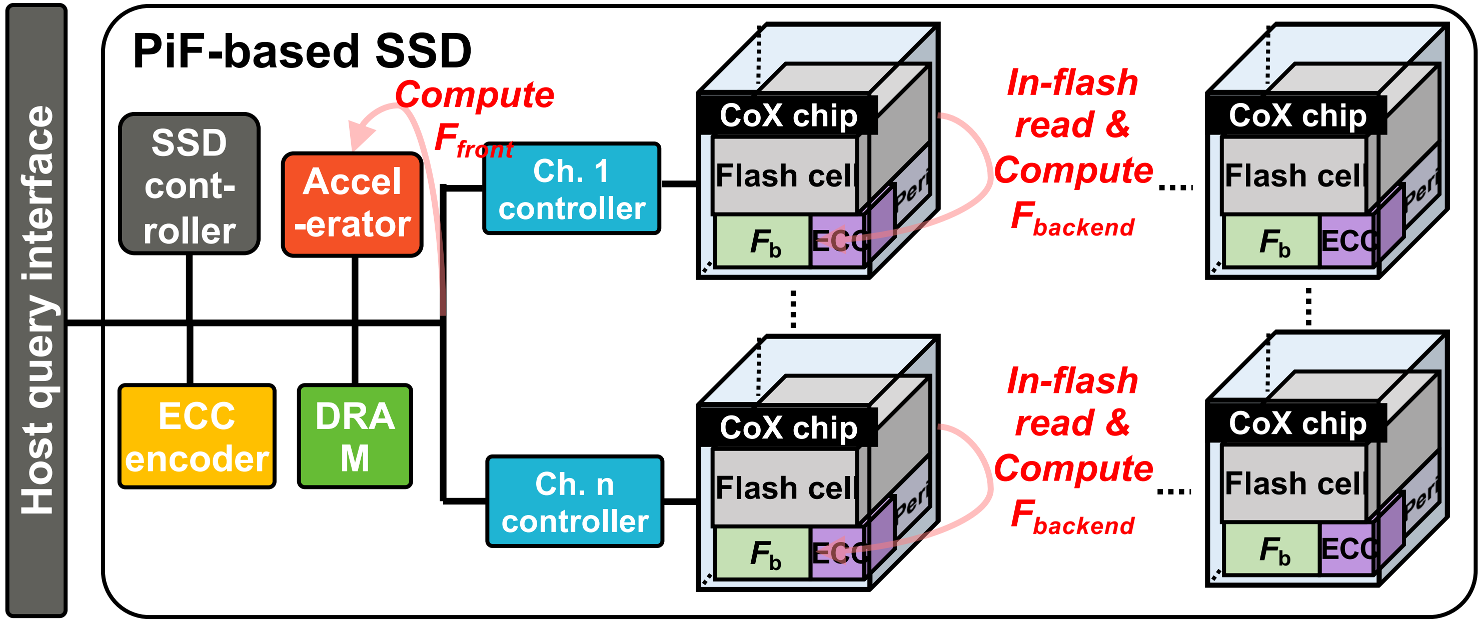
Processing-in-Flash Architectures

A Cell-over-X (CoX) flash chip



In-flash accelerator

In-flash ECC module



Challenges in Designing a PiF SSD

- What is the power budget for CoX flash chips?
  - In typical flash chips, Power$_{program}$ > Power$_{read}$
  - Allocate Power$_{program}$ – Power$_{read}$ to the CoX flash chips

- How can we support reliable in-flash read without a controller-side ECC module?
  - Design a weak but low-complexity ECC module

- What is the ideal computation stage should be offloaded to the CoX flash chip?
  - Requirements of ideal candidates:
    - A large amount of data reduction ratio
    - Suitable for data-parallel processing
    - Low implementation overhead (i.e., under a power/area budget)

CARES

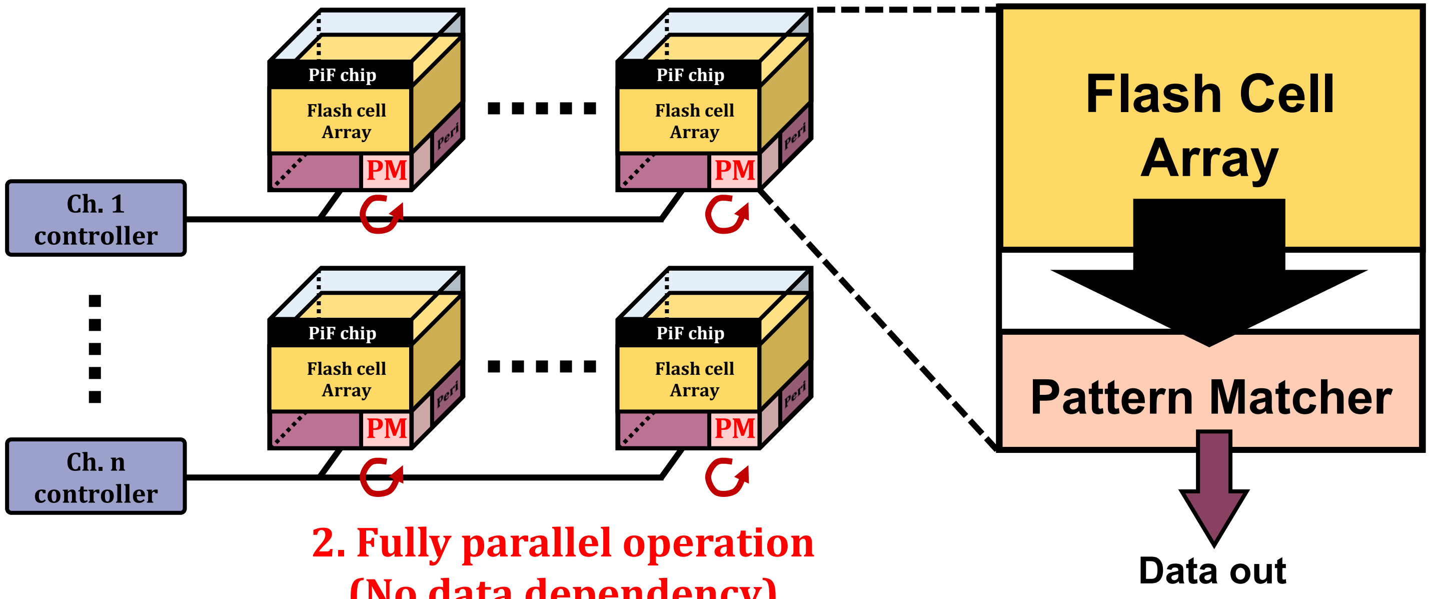
Outline

001 Processing-in-Flash Architectures and Challenges

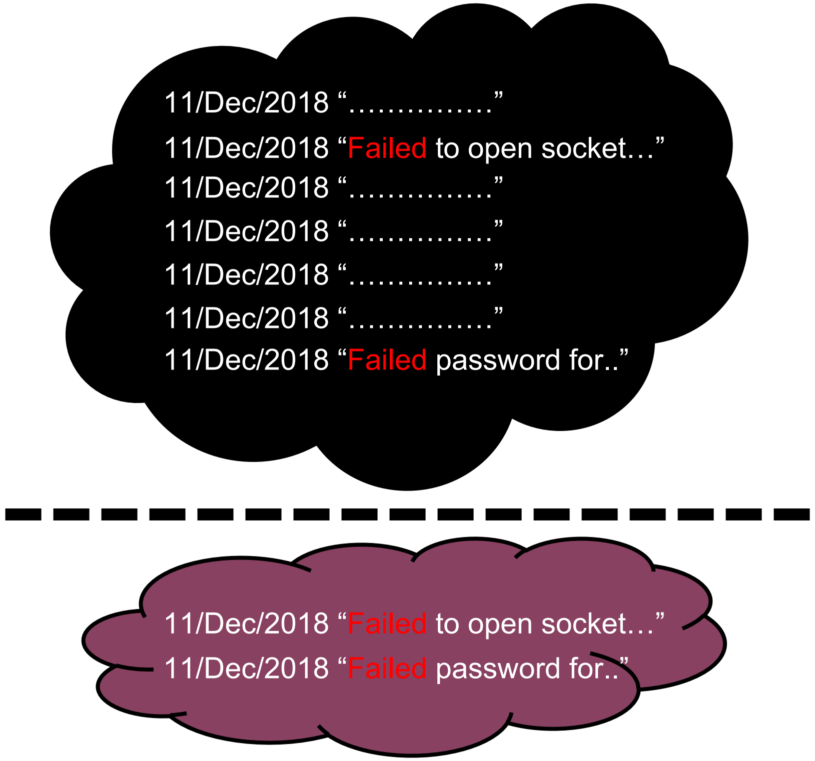
002 Use case: A Pattern Matching Enabled PiF Architecture

003 Evaluation & Conclusion

Use Case: PiF-PM with a Pattern Matcher



2. Fully parallel operation
(No data dependency)



```
11/Dec/2018 "…………….."
11/Dec/2018 "Failed to open socket…"
11/Dec/2018 "…………….."
11/Dec/2018 "…………….."
11/Dec/2018 "…………….."
11/Dec/2018 "…………….."
11/Dec/2018 "Failed password for.."
```

```
11/Dec/2018 "Failed to open socket…"
11/Dec/2018 "Failed password for.."
```

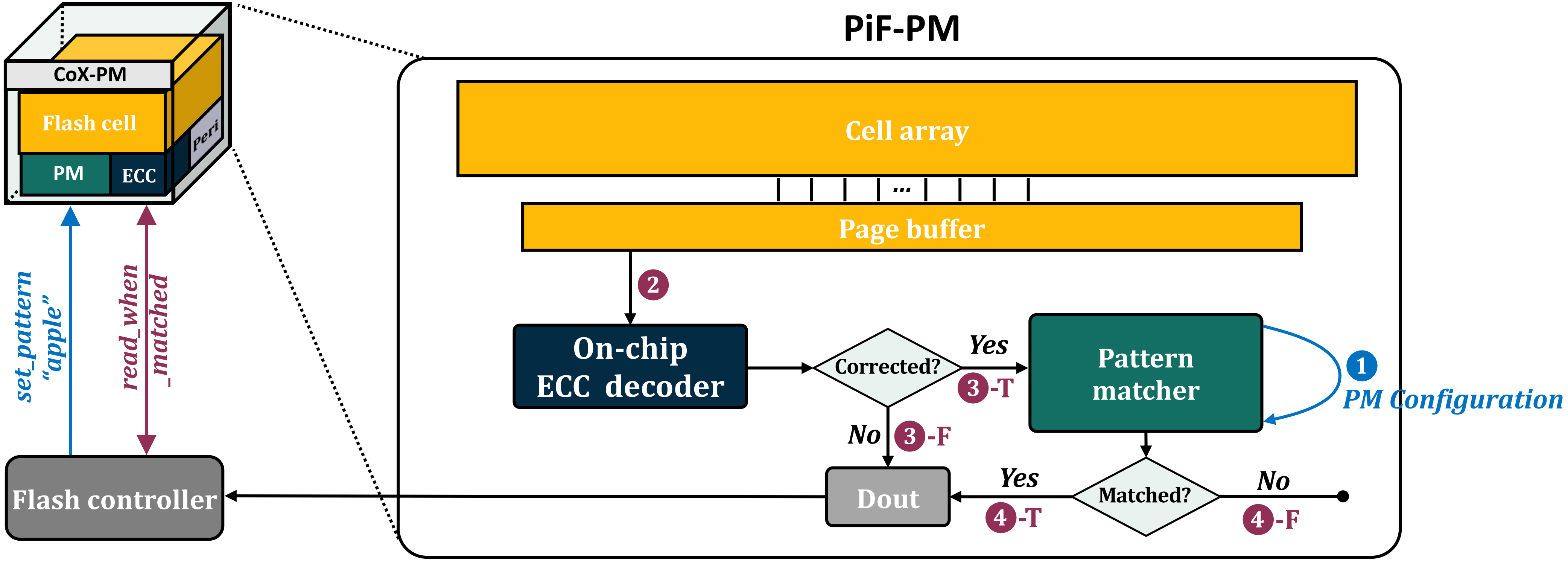
1. High data reduction

3. High applicability

….

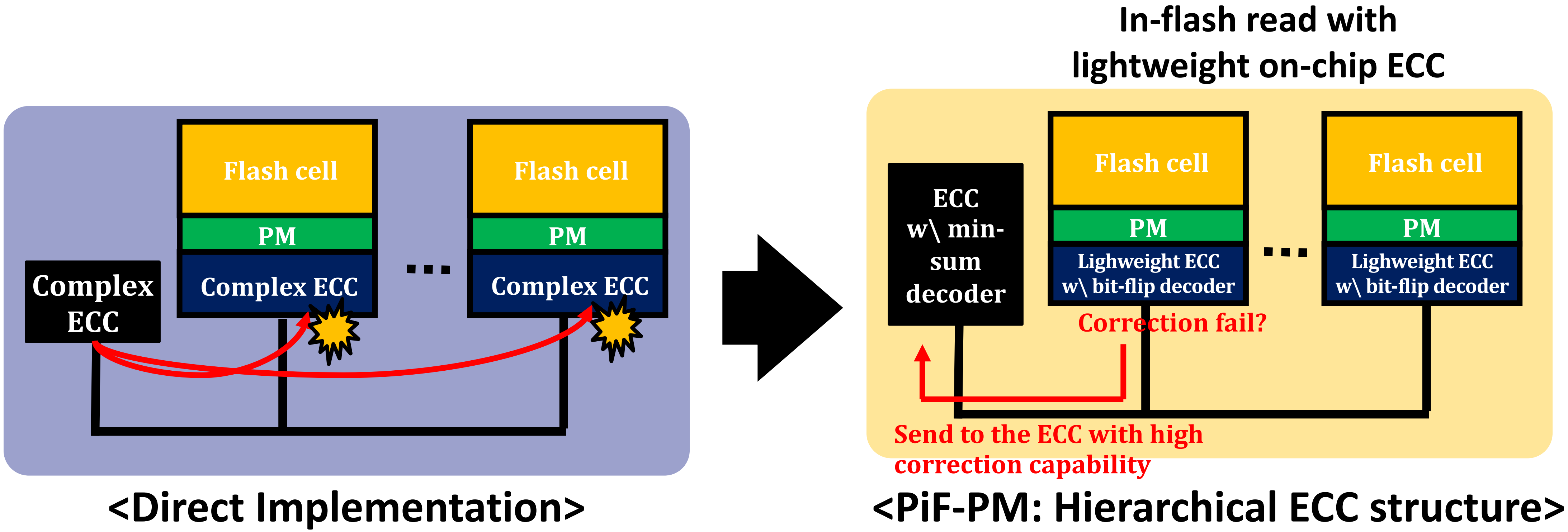
PiF-PM: Operational Overview

- Two additional commands for supporting PiF-PM
  - *set_pattern*: configure the PM to search the specific patterns
  - *read_when_matched*: only output the pages containing specified patterns



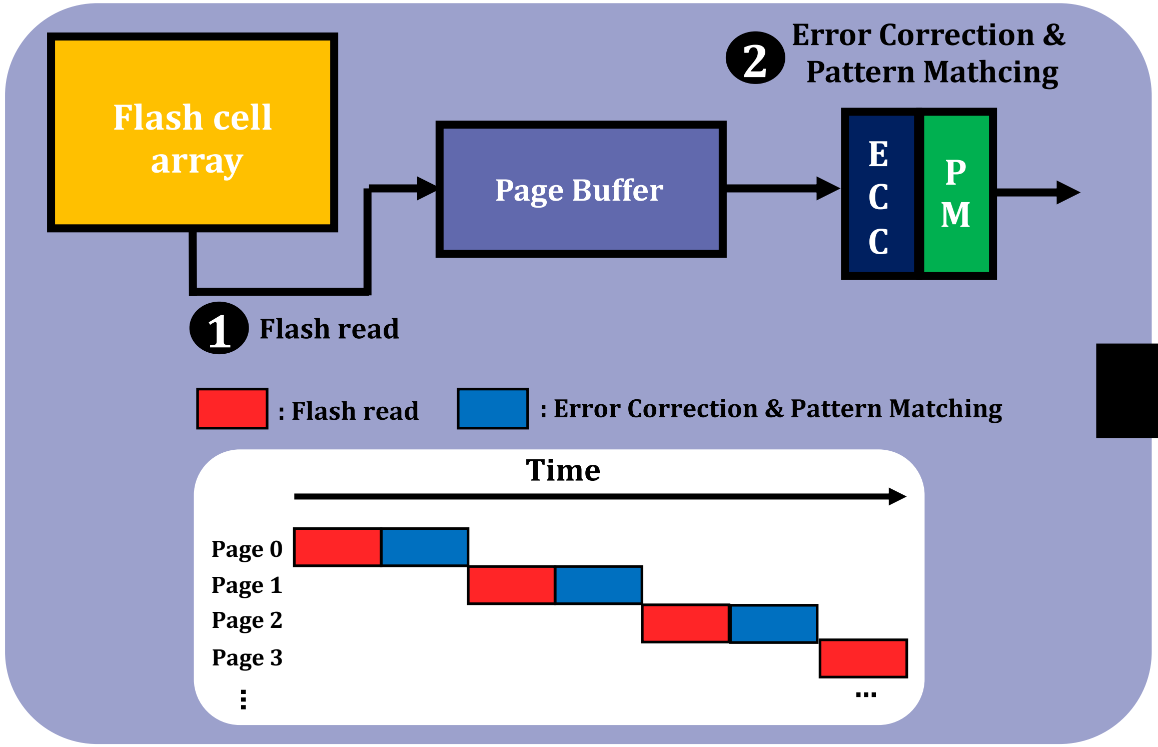
Challenge 1: Reliabile In-flash Read

- Direct implementation of controller-side ECC engine on chip incurs high power & area consumption

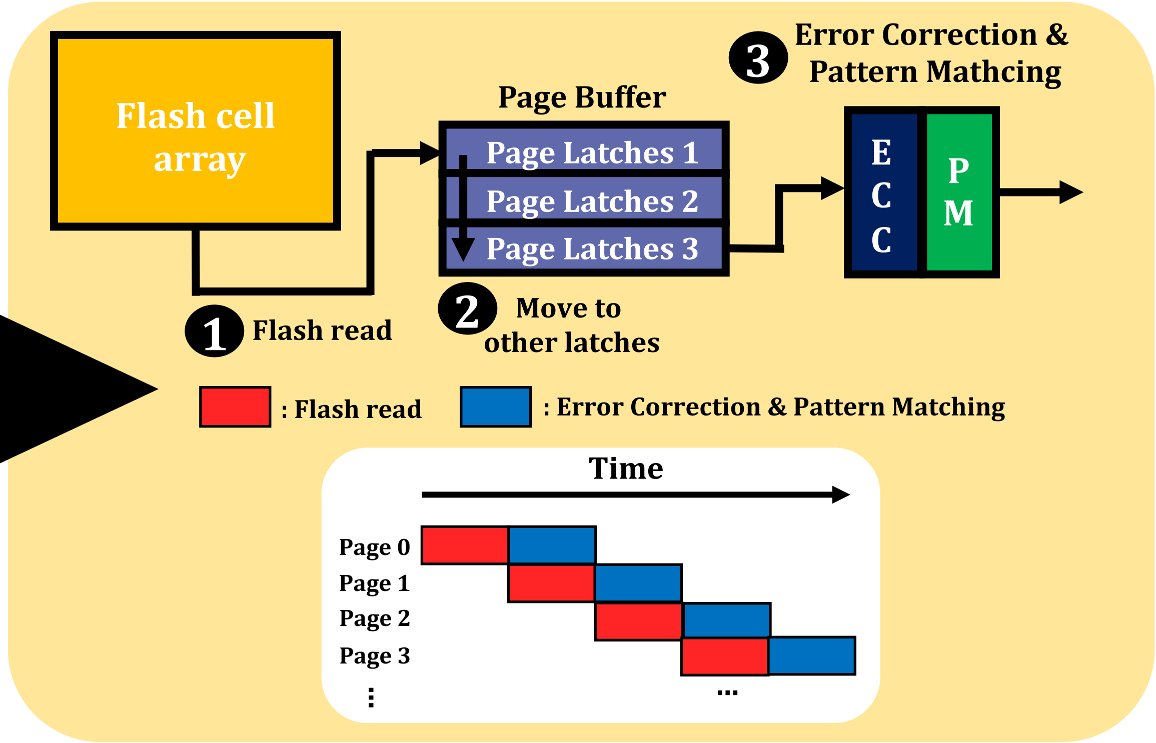


Challenge 2: Bandwith Degradation

- Simple structure: Bandwidth degradation due to extra operations
- Pipelined structure of PiF-PM: Fully exploiting the chip bandwidth by overlapping all operations



<Simple structure>



<PiF-PM: Pipelined structure>

Outline

001 Processing-in-Flash Architectures and Challenges

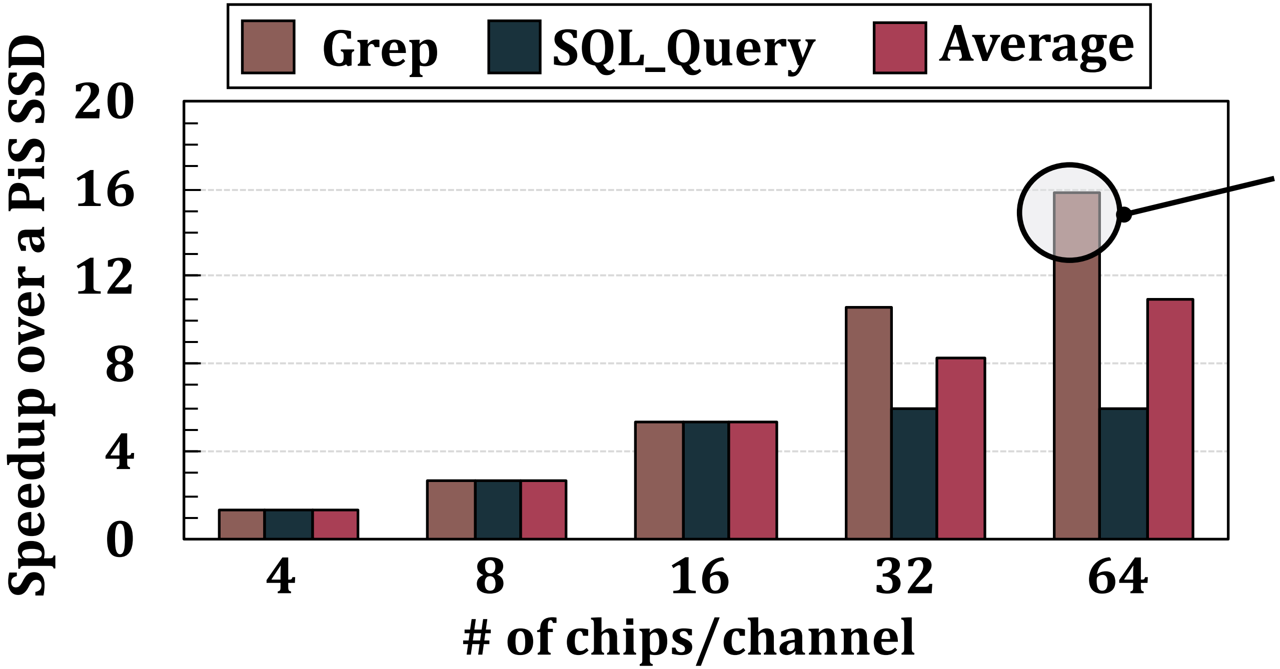
002 Use case: A Pattern Matching Enabled PiF Architecture

003 Evaluation & Conclusion

Experimental Setup

- Evaluation Platform
  - Cosmos+ OpenSSD
- Comparison schemes
  - Baseline: Processing-in-Storage scheme
  - Proposed: Processing-in-flash with CoX-PM
- Workloads
  - Grep
  - SQL_Query (TPC-H benchmark)
- Metrics
  - All values are normalized to baseline

Result 1: Performance Improvement

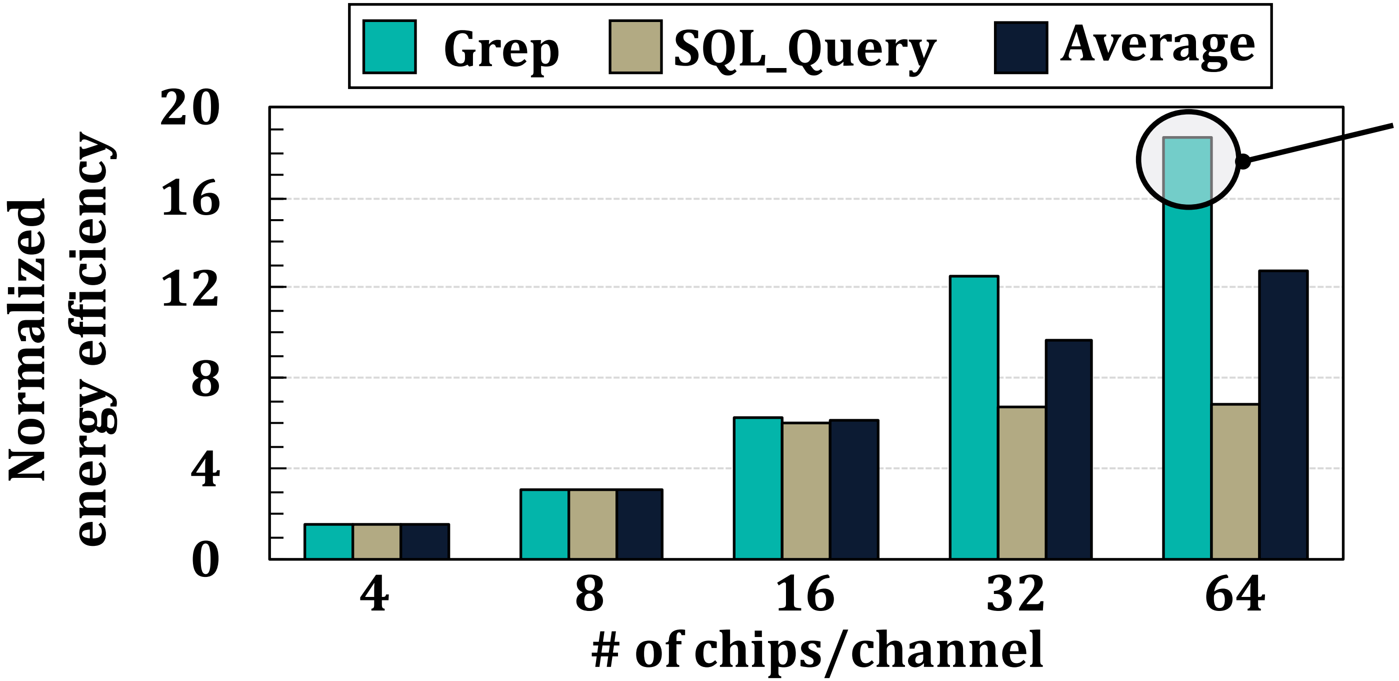


15.9x improvement

Observation 1: Almost scalable performance improvement under varying number of chips/channel

Observation 2: Different performance improvement by the difference in the data reduction ratio (Grep: 93.7%, SQL_Query: 83.3%)

Result 2: Energy Efficiency



18.6x improvement

Observation: Achieved high energy efficiency due to the performance improvement and data transfer reduction along channels

Conclusion

- Investigated the limitation of the existing processing-in-storage scheme by slow internal bandwidth
- Proposed a processing-in-flash (PiF) scheme that moves computation inside flash chips where data are physically present
- Demonstrated that the PiF-based SSD is very promising by outperforming a PiS-based SSD by several times in the execution time/power efficiency

Thank You!