The 29th IEEE International Symposium on High-Performance Computer Architecture (HPCA-29)

NvWa: Enhancing Sequence Alignment Accelerator Throughput via Hardware Scheduling

Yewen Li[1,2], Xueqi Li[1,✉], Ruihao Gao[1,2], Wanqi Liu[1,2], Guangming Tan[1,2]

[1]State Key Lab of Processors, ICT, CAS

[2]University of Chinese Academy of Sciences, Beijing, China

INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

University of Chinese Academy of Sciences

XPU-driven Research on Sys and Arch.

Presentation Outline

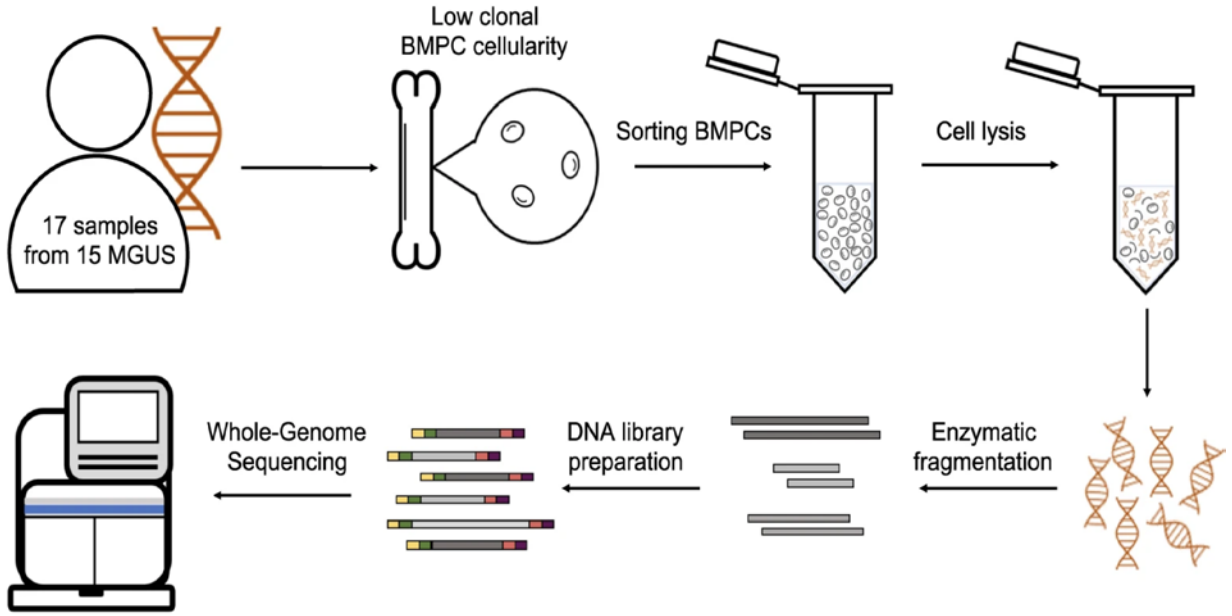
- Background
- Motivation and Challenges
- NvWa Accelerator
- Evaluation
- Conclusion

Background: Genome Sequencing

Genome

Source: NIH Genome

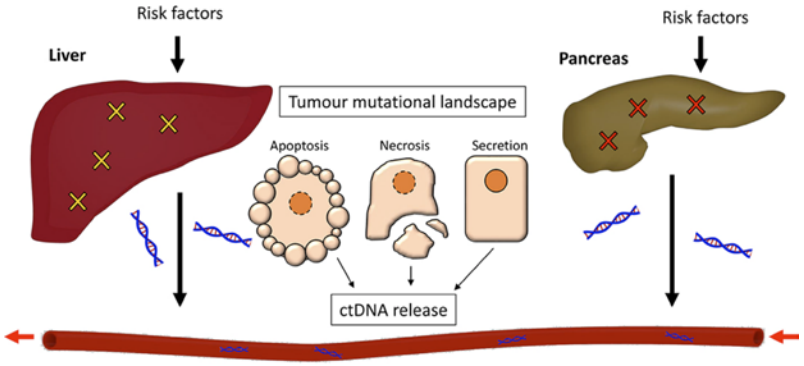
Genome Sequencing



Source: Bénedith Oben et al.[Nature Communication'21]

Background: The Importance of Genome Sequencing

Detecting Genome Mutations

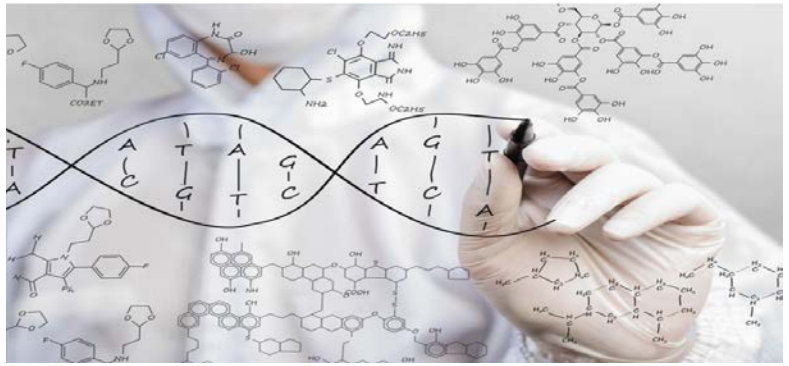


Source: Alistair James Rice [Frontier'19]

Personal Medicine

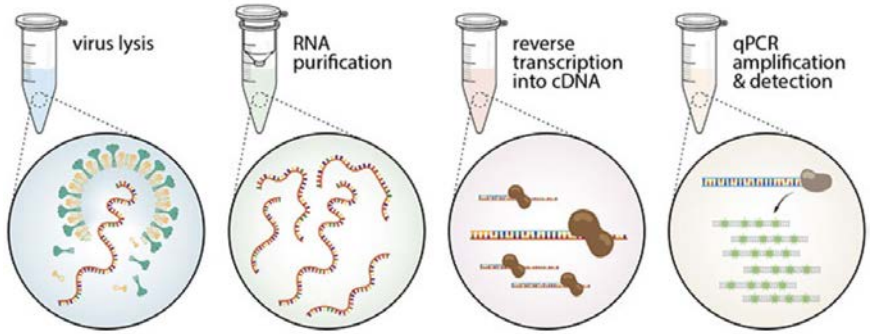
Source: NIH News in Health

Genetic Research



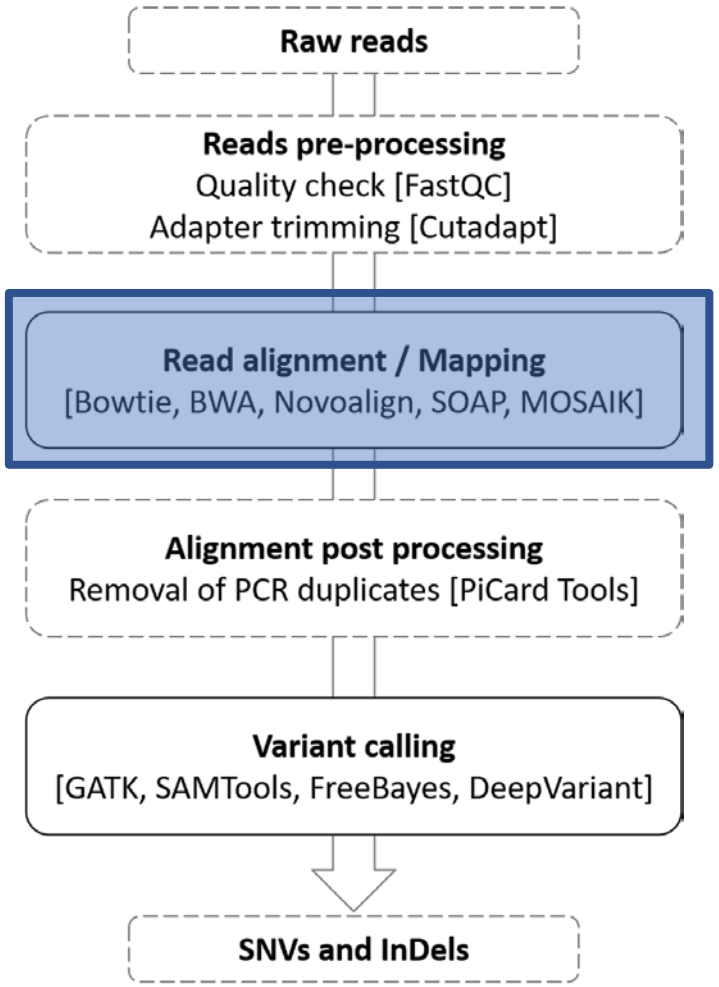
Source: Undergraduate Genetics Lecture and Lab Offered Online

COVID-19 Detection



Source: How do Coronavirus Disease (COVID-19) Tests Work?

Background: Sequence Alignment (aka. Read Alignment)



Source: Manojkumar Kumaran et al. [BMC Bioinformatics'20]

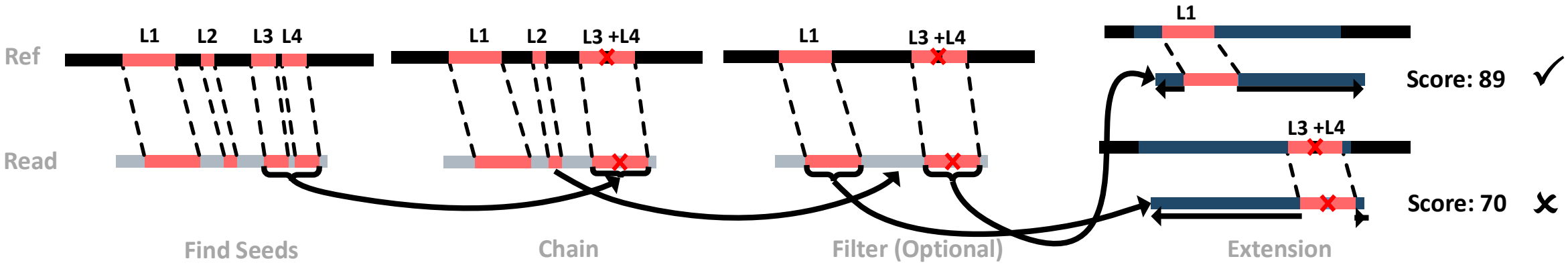
The essential step in genome analysis pipeline.

300 hours when aligning the NA12878 dataset[1].

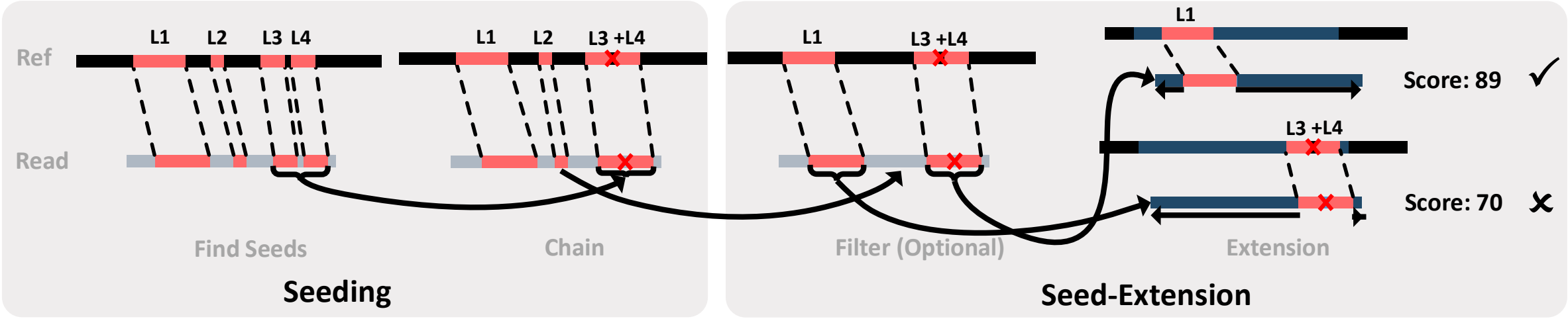
30% execution time in the genome analysis pipeline[2].

[1] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," arXiv:1303.3997 [q-bio], May 2013.
[2] T. J. Ham et al., "Genesis: A Hardware Acceleration Framework for Genomic Data Analysis," in ISCA, Los Angeles, CA, 2020.

Background: Sequence Alignment Pipeline



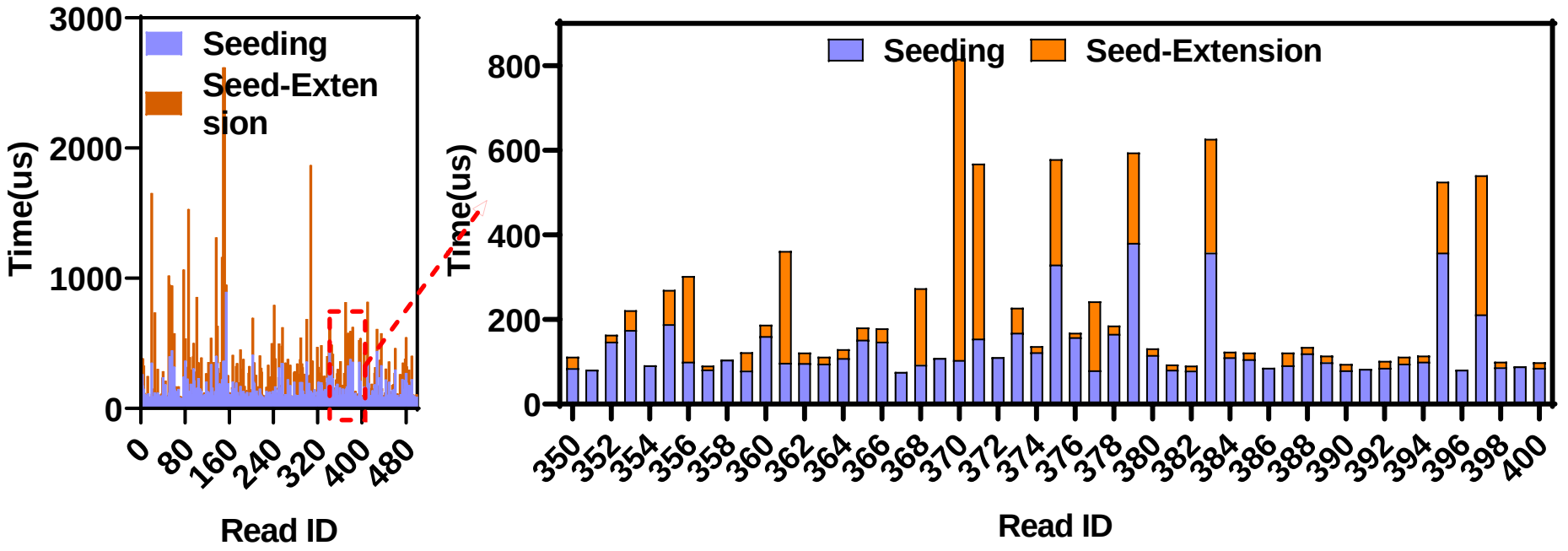
Background: Sequence Alignment Pipeline



Presentation Outline

- Background
- Motivation and Challenges
- NvWa Accelerator
- Evaluation
- Conclusion

Motivation and Challenges: Key Observation

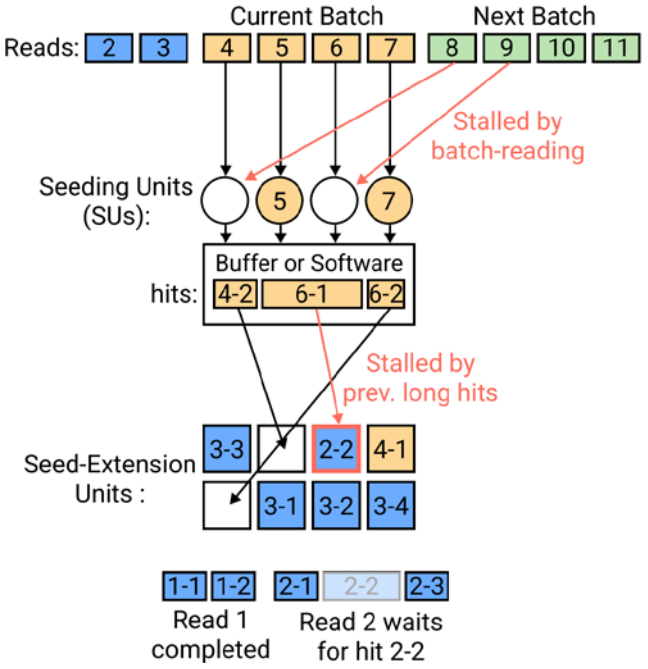


Diversity Problem

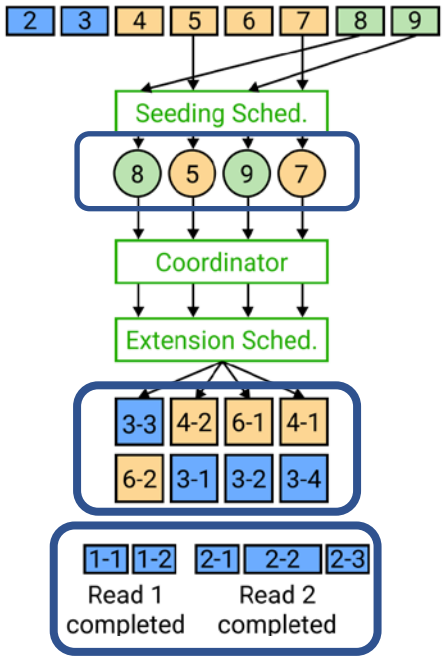
The proportion of two phases is varied

The total execution time of each read is also varied.

Motivation and Challenges: With Ideal Scheduler?



Traditional approach



Traditional approach with hardware schedulers

Benefit-1: SU 1 and 3 can load read 8 and read 9.

Benefit-2: EU 2 and EU 5 can execute hit 4-2 and 6-2, long hit 2-2 can be executed efficiently.

Benefit-3: Read 2 is not stalled by hit 2-2.

We argue that multi-phase scheduling is a promising solution to this problem.

Motivation and Challenges: Three Challenges

Challenge-1: Seeding Termination Diversity

How to schedule the idle SUs efficiently when considering the execution time of SUs is sensitive to the input reads.

Challenge-2: Extension Scales Diversity

How to avoid the under-utilization and workload imbalance of the EUs when the hardware design is specified in advance.

Challenge-3: Hit Characteristics Diversity

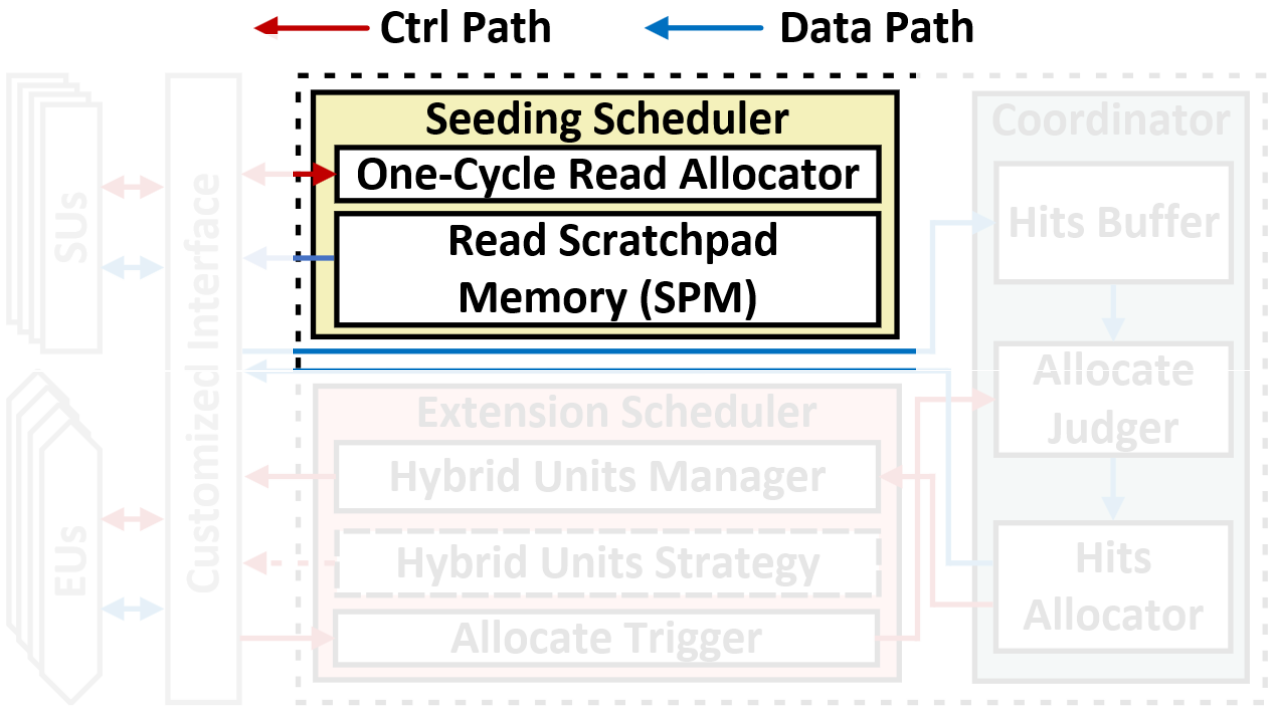
How to cache and analyze hit in a lightweight way since the number and features of the hits generated by each SUs are varied.

Presentation Outline

- Background
- Motivation and Challenges
- NvWa Accelerator
- Evaluation
- Conclusion

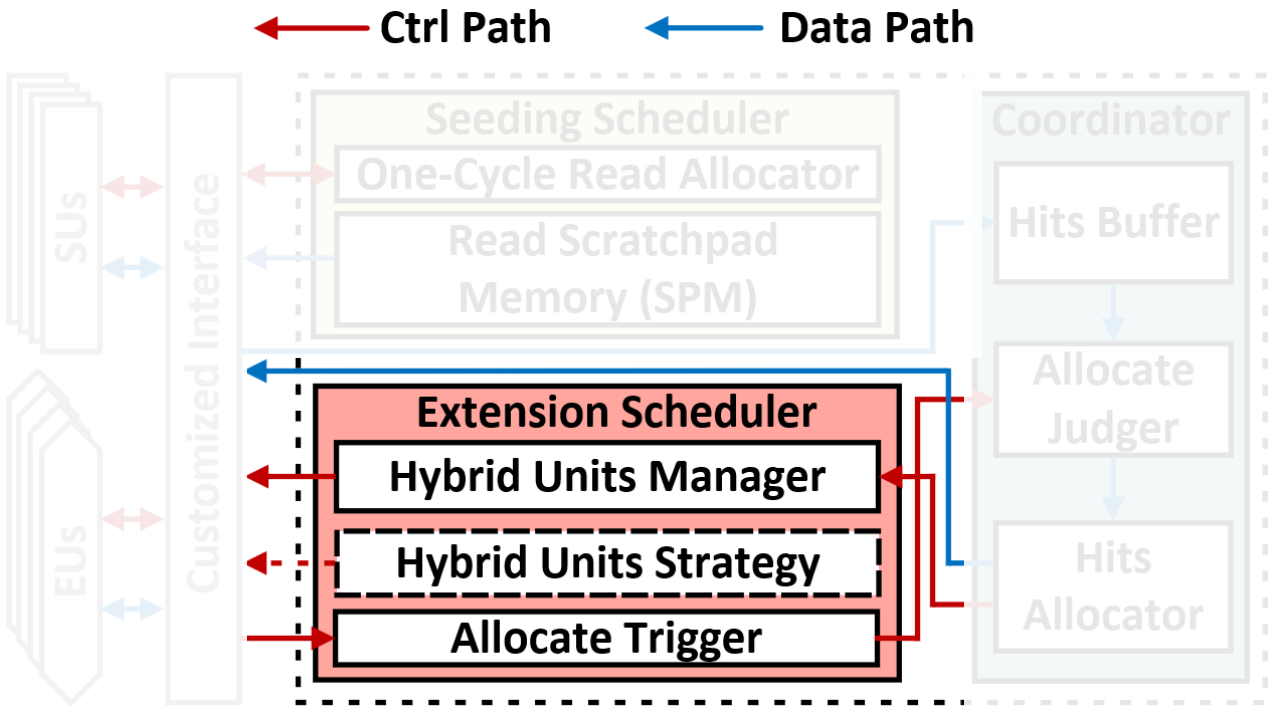
NvWa Accelerator: Architecture Overview

- **(Solving Challenge-1) Seeding Scheduler.**
  - Inefficiency of previous scheduling strategy.
  - Propose one-cycle scheduling architecture.
- **(Solving Challenge-2) Extension Scheduler.**
  - Latency model for EUs latency.
  - Propose hybrid-unit strategy.
- **(Solving Challenge-3) Coordinator.**
  - Two coordinate problem.
  - Propose micro-architecture design.
- **Other Components.**
  - SUs and EUs are based on previous work.
  - Unified Interface for flexibility.



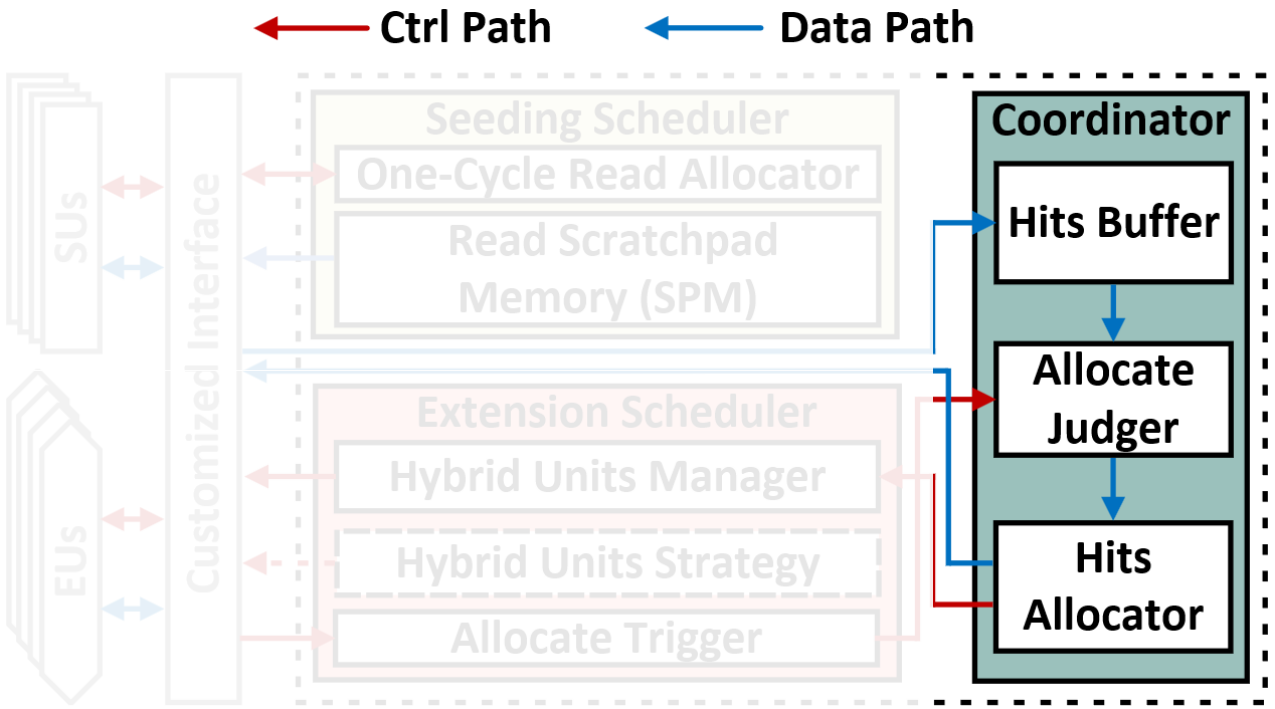
NvWa Accelerator: Architecture Overview

- **(Solving Challenge-1) Seeding Scheduler.**
  - Inefficiency of previous scheduling strategy.
  - Propose one-cycle scheduling architecture.
- **(Solving Challenge-2) Extension Scheduler.**
  - Latency model for EUs latency.
  - Propose hybrid-unit strategy.
- **(Solving Challenge-3) Coordinator.**
  - Two coordinate problem.
  - Propose micro-architecture design.
- **Other Components.**
  - SUs and EUs are based on previous work.
  - Unified Interface for flexibility.



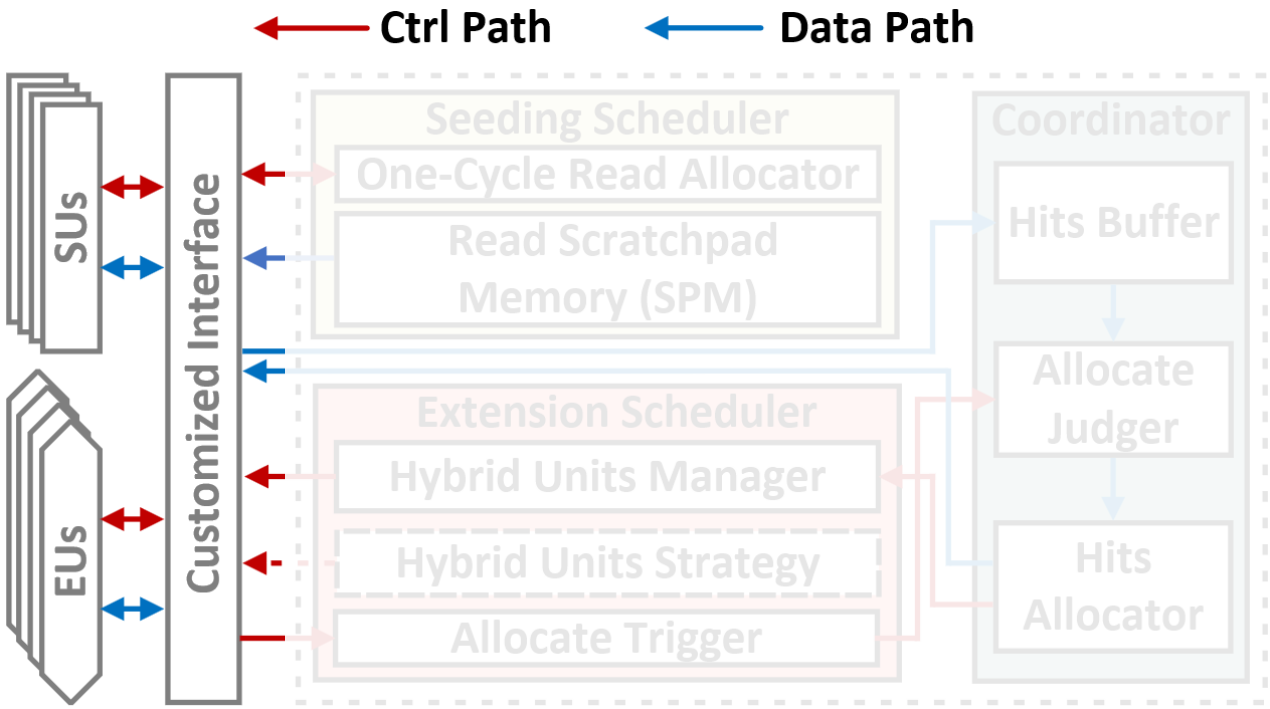
NvWa Accelerator: Architecture Overview

- **(Solving Challenge-1) Seeding Scheduler.**
  - Inefficiency of previous scheduling strategy.
  - Propose one-cycle scheduling architecture.
- **(Solving Challenge-2) Extension Scheduler.**
  - Latency model for EUs latency.
  - Propose hybrid-unit strategy.
- **(Solving Challenge-3) Coordinator.**
  - Two coordinate problem.
  - Propose micro-architecture design.
- **Other Components.**
  - SUs and EUs are based on previous work.
  - Unified Interface for flexibility.



NvWa Accelerator: Architecture Overview

- **(Solving Challenge-1) Seeding Scheduler.**
  - Inefficiency of previous scheduling strategy.
  - Propose one-cycle scheduling architecture.
- **(Solving Challenge-2) Extension Scheduler.**
  - Latency model for EUs latency.
  - Propose hybrid-unit strategy.
- **(Solving Challenge-3) Coordinator.**
  - Two coordinate problem.
  - Propose micro-architecture design.
- **Other Components.**
  - SUs and EUs are based on previous work.
  - Unified Interface for flexibility.



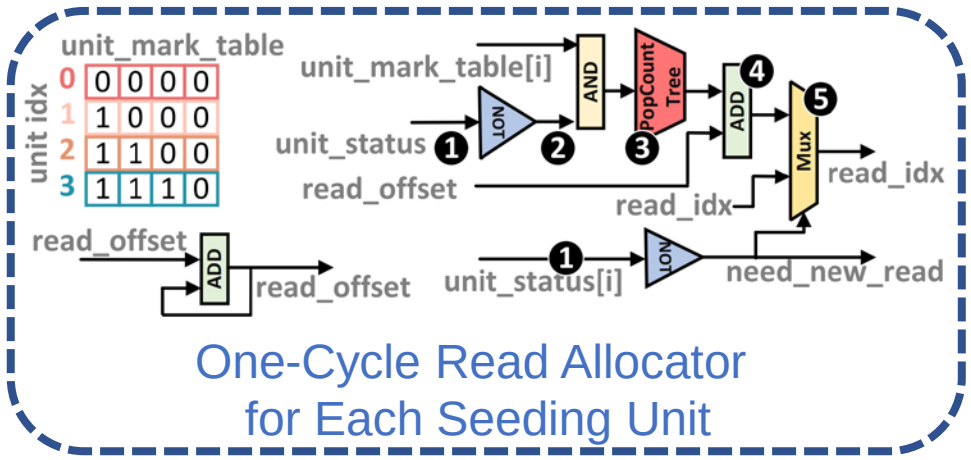
Seeding Scheduler

Read-in-Batch Strategy

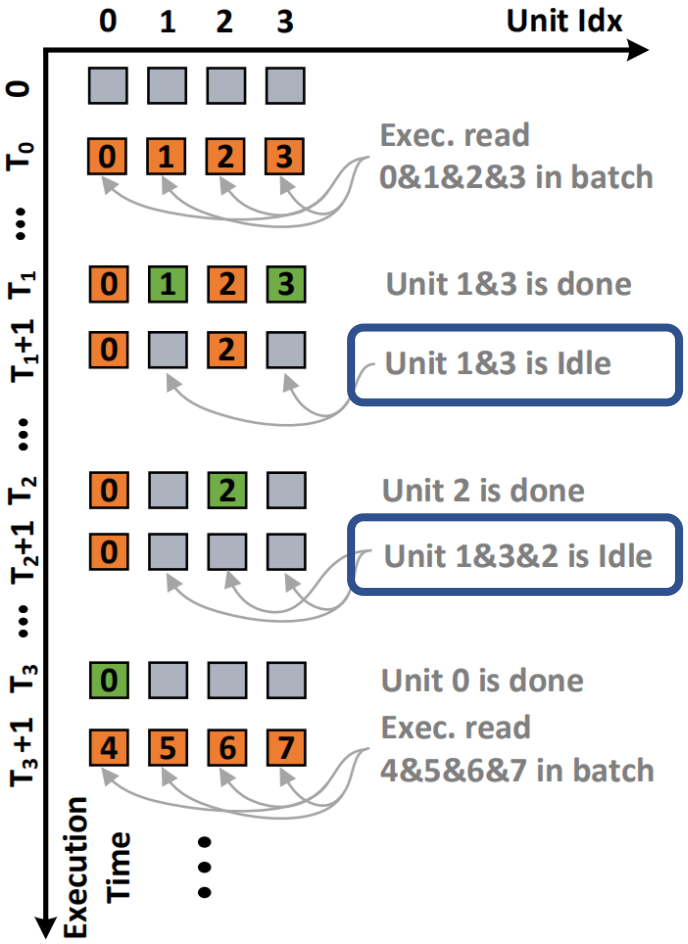
- Used by GenAx and ERT.
- Low seeding unit utilization.

One-Cycle Allocation Strategy

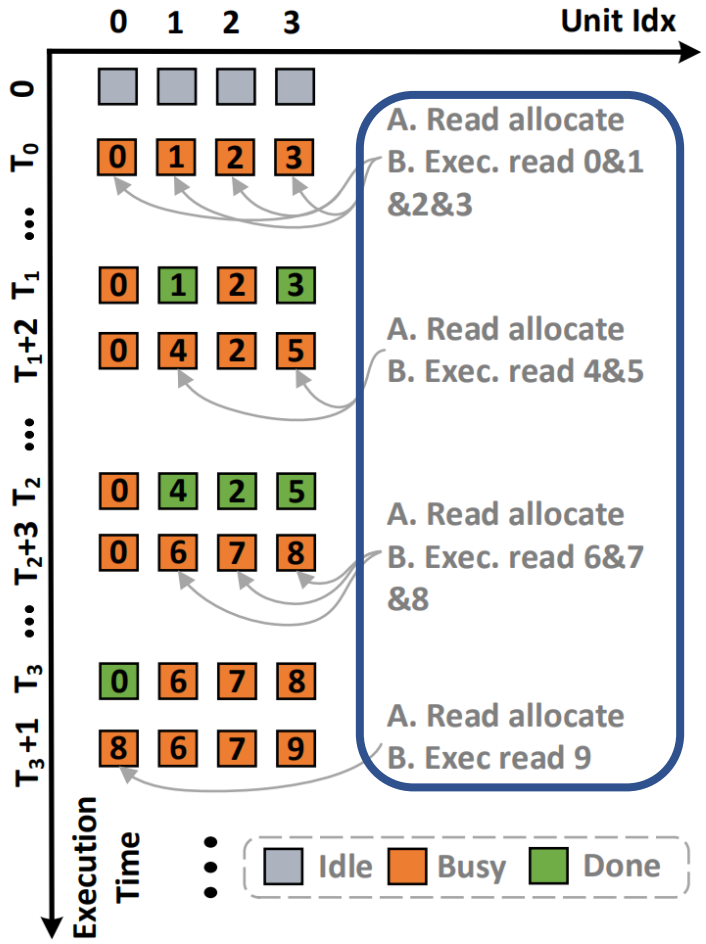
- 95% seeding unit utilization.
- Very simple logic.



One-Cycle Read Allocator for Each Seeding Unit

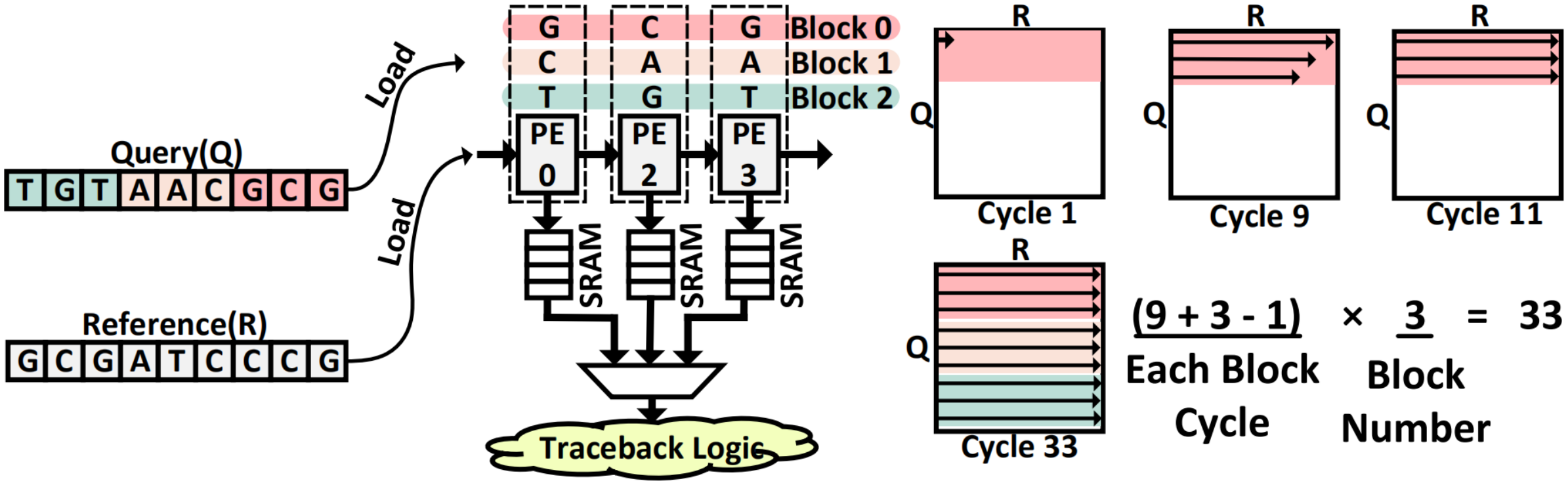


Read-in-Batch Strategy



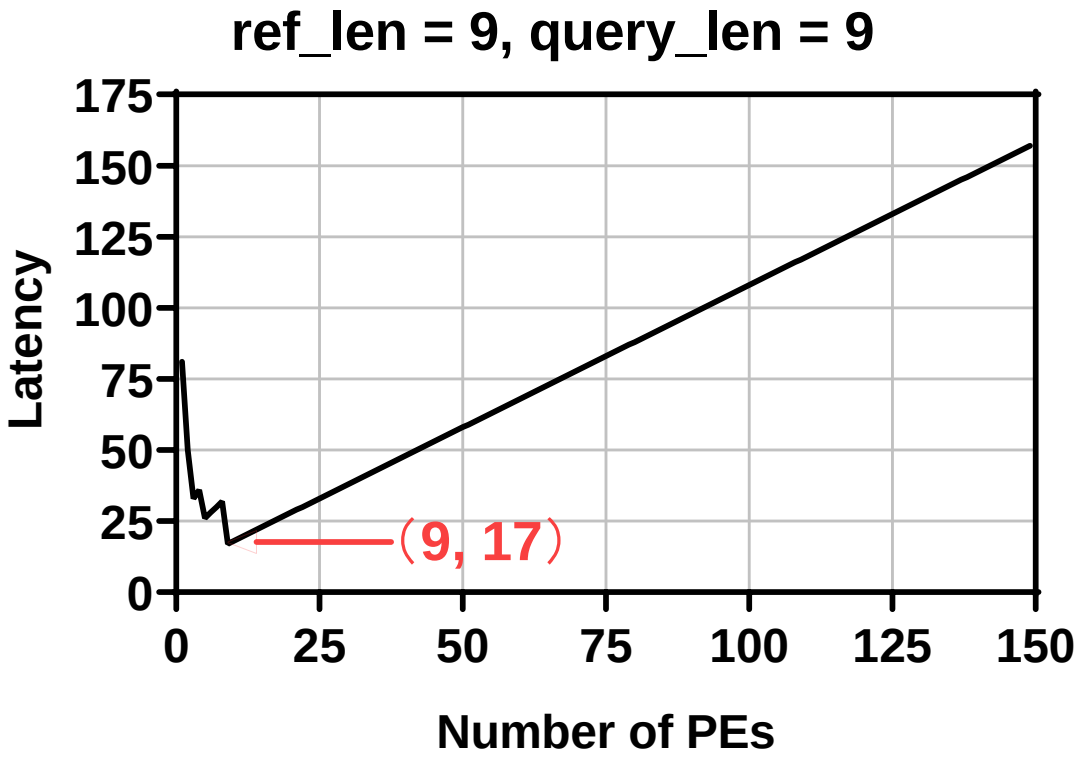
One-Cycle Allocation Strategy

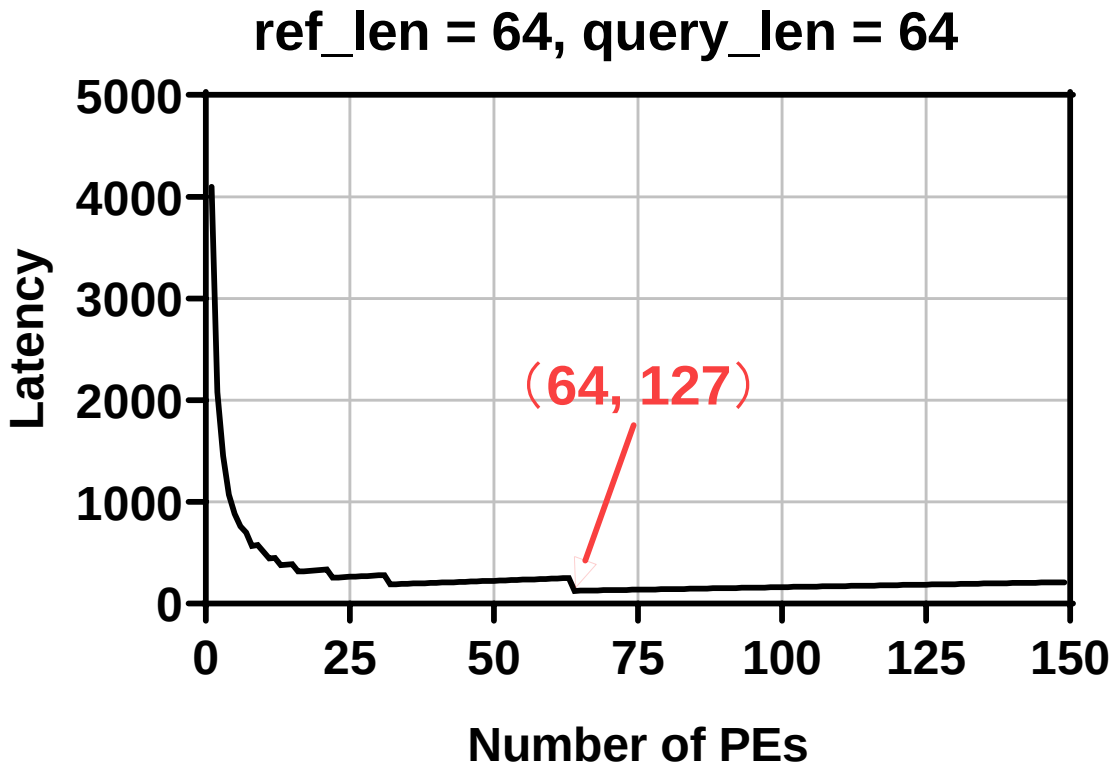
Extension Scheduler: Workflow of Systolic Array



$$Latency = (ref\_len + pe\_num - 1) \times \left\lceil \frac{query\_len}{pe\_num} \right\rceil$$

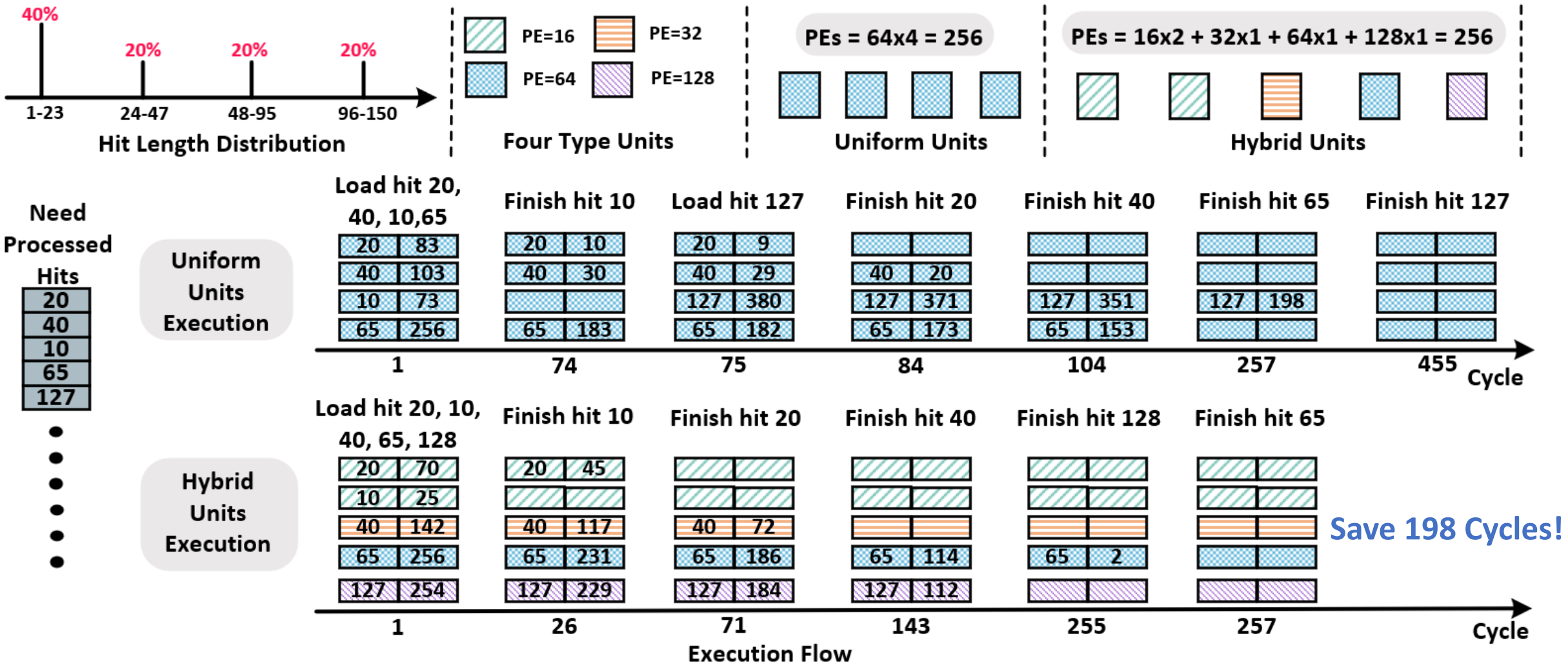
Extension Scheduler: Execution Latency vs. Hit Length





The latency is optimal when the hit length and the PE number are close.

Extension Scheduler: Hybrid Unit Strategy



Coordinator: Two Scheduling Problems

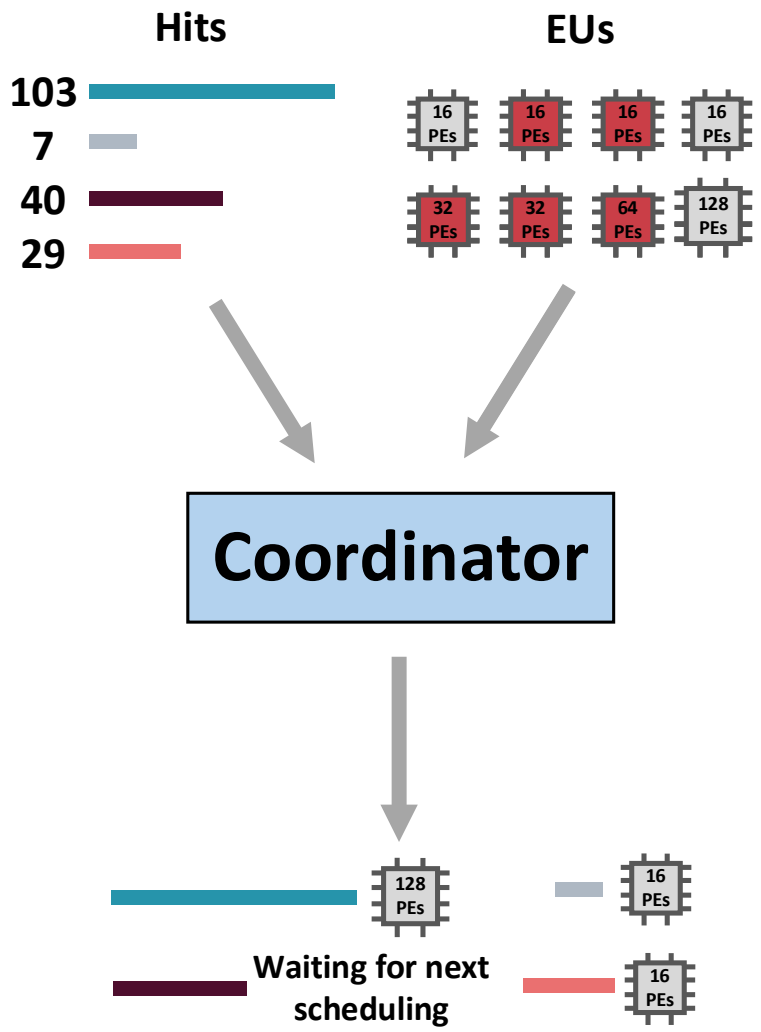
Hits Fragmentation Problem and Solution

- **Problem:** The allocated failed hits will be kept in hits buffer, which leads to a fragmentation problem.
- **Solution:**
  - Define the allocated hits offset to know where to read the hits in the subsequent scheduling.
  - Sort to move the unprocessed hits to the end of the data batch.

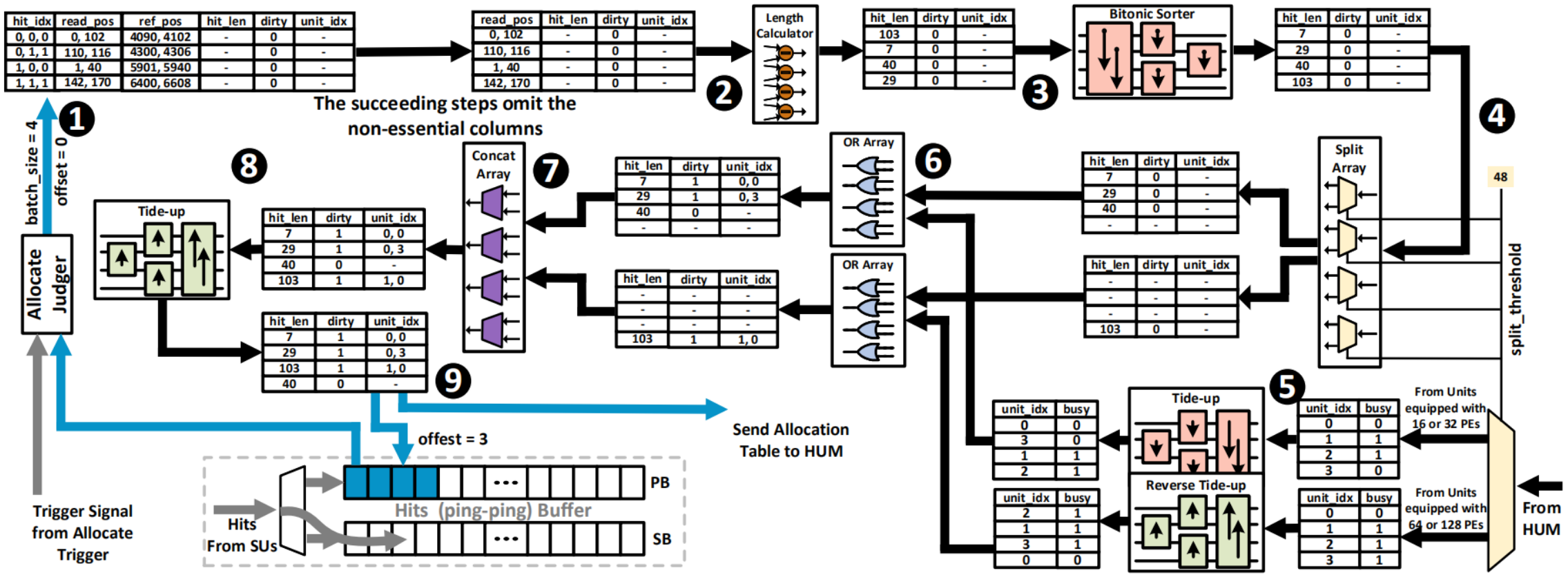
Hits Allocation Problem and Solution

- **Problem:** How to get a good tradeoff in execution latency and parallelism?
- **Solution:**
  - Group the EUs into several groups according to its PE numbers.
  - A low-latency greedy allocation strategy.

Toy Example



Coordinator: Micro-architecture



About 5~10 cycles

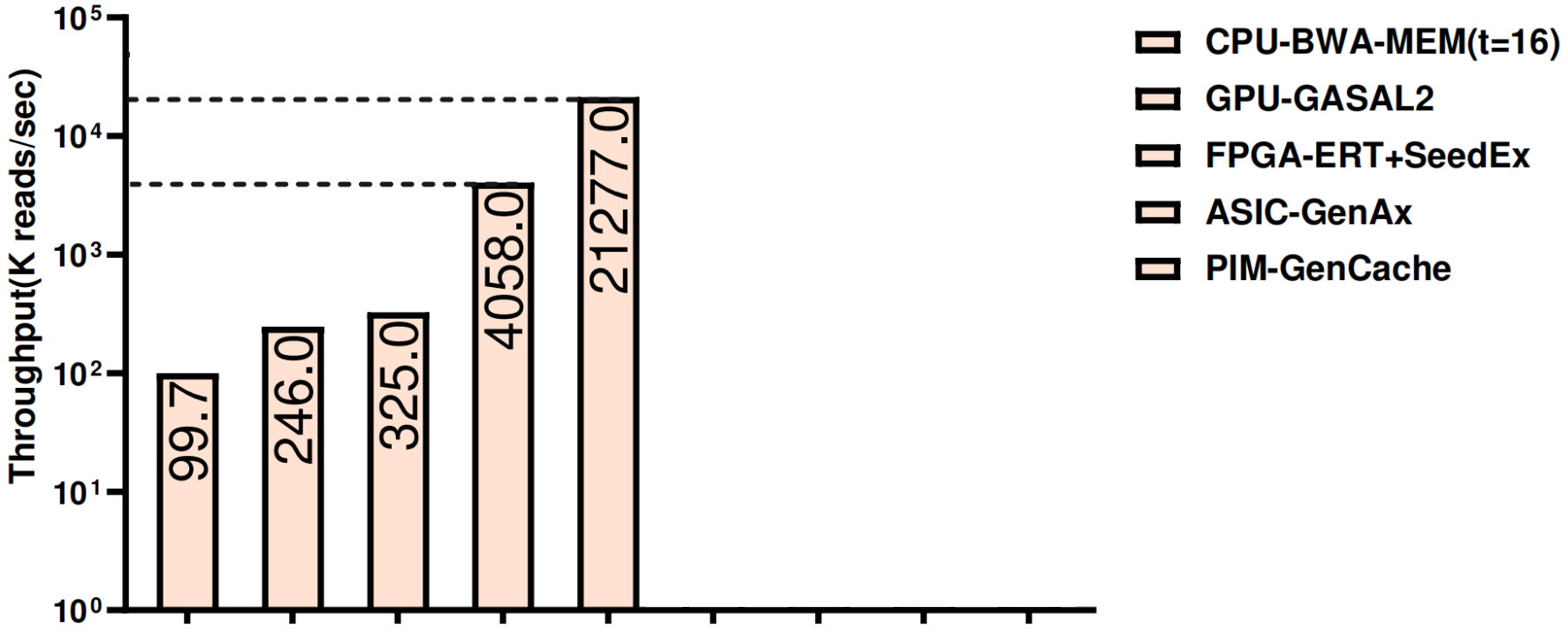
Presentation Outline

- Background
- Motivation and Challenges
- NvWa Accelerator
- Evaluation
- Conclusion

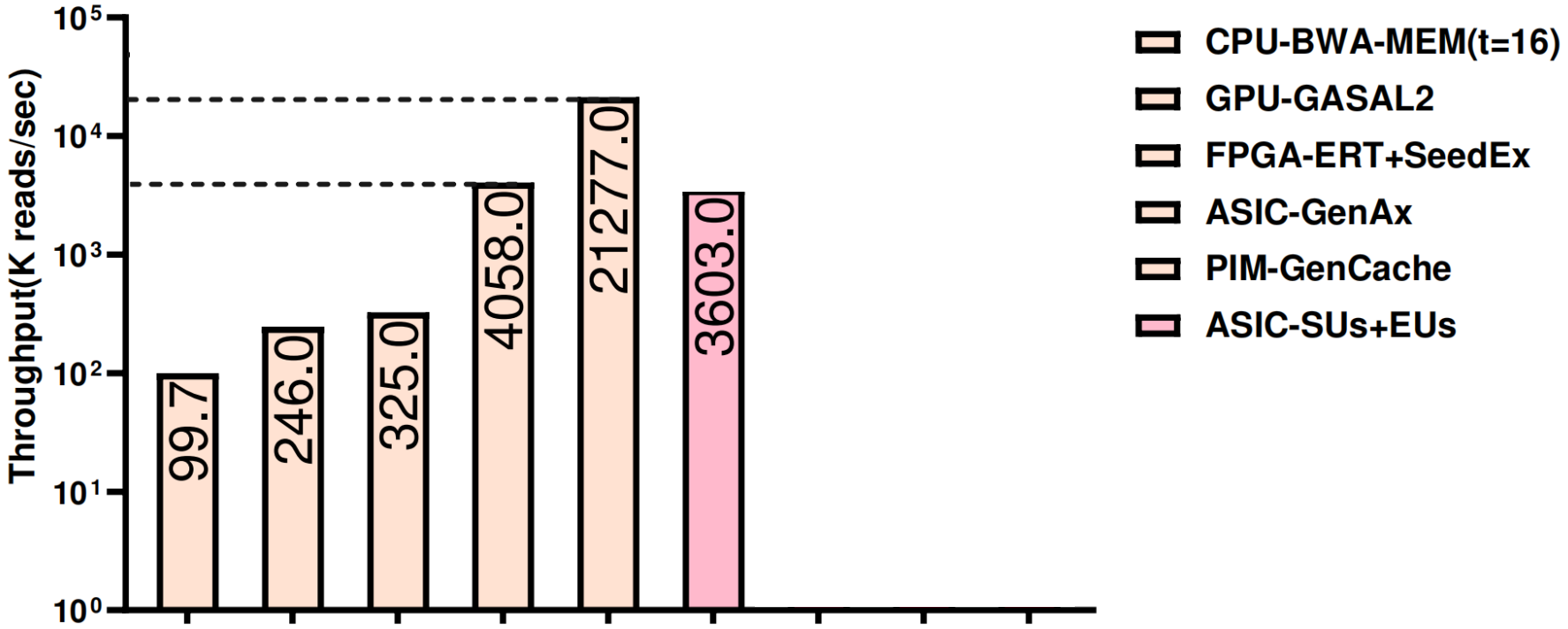
Evaluation: Methodology

- **Architecture Simulation:**
  - System Implementation: cycle-accurate and execution-driven simulator using Python
  - Key Components: write Chisel3 and generate Verilog
  - CAD Tools: Synopsys Design Compiler with the SIMC 14 nm standard VT library
  - Memory: Ramulator
- **Baseline:**
  - CPU: BWA-MEM (16 cores @ 2.10GHz + 128GB DDR4)
  - GPU: GASAL2 (NVIDIA A100 GPU)
  - FPGA: SeedEx[MICRO'20]+ERT[ISCA'21]
  - ASIC: GenAx[ISCA'18]
  - PIM: GenCache[MICRO'19]
- **Datasets:**
  - Platinum NA12878 genome dataset
  - Five datasets generated by DWGSIM and NCBI reference genome

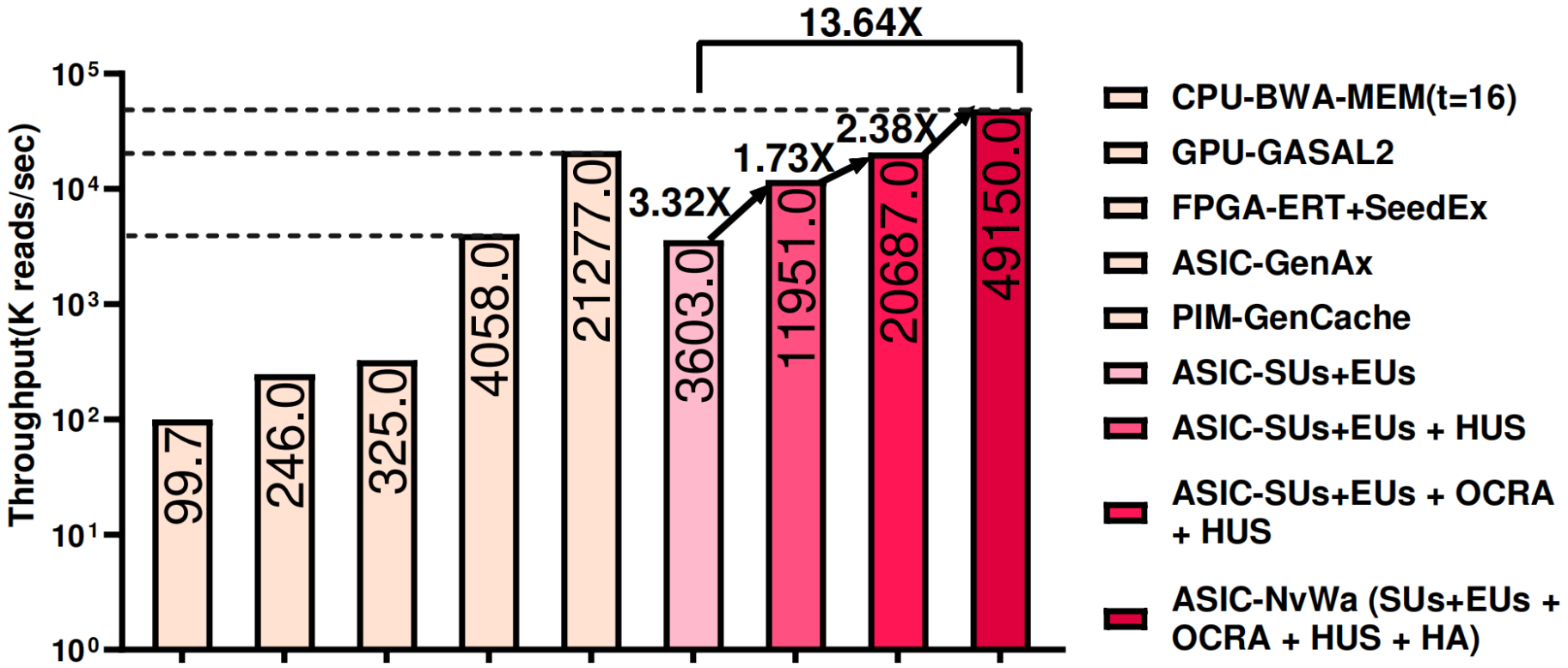
Evaluation: Throughput



Evaluation: Throughput



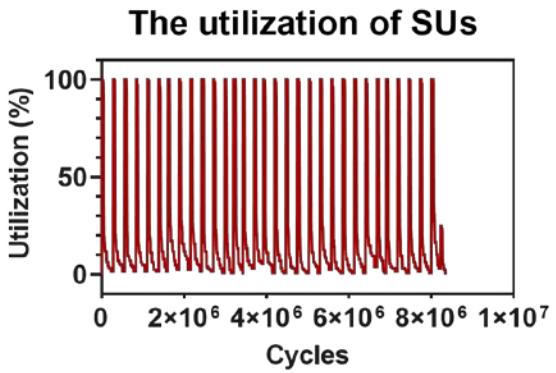
Evaluation: Throughput

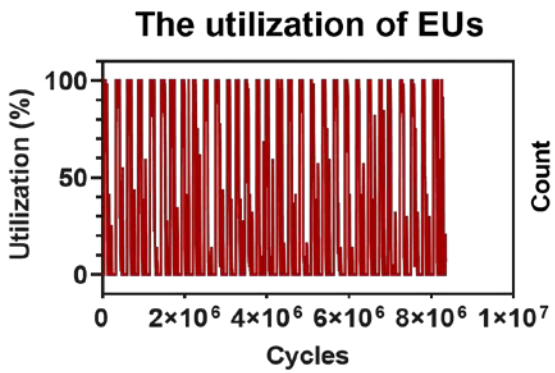


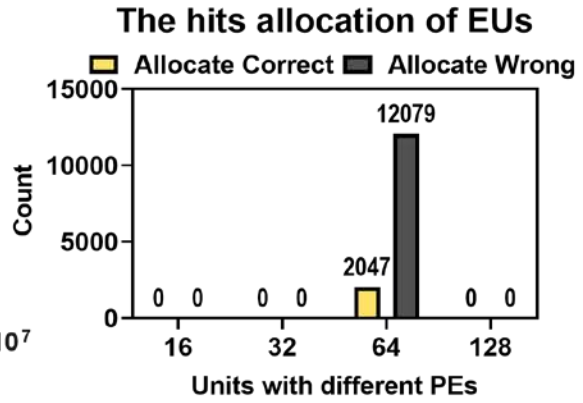
1. NvWa provides 2.30x speedup when compared with SOTA accelerators.
2. Three strategy provides 3.32x, 1.73x, and 2.38x, respectively.

Evaluation: Resource Utilization

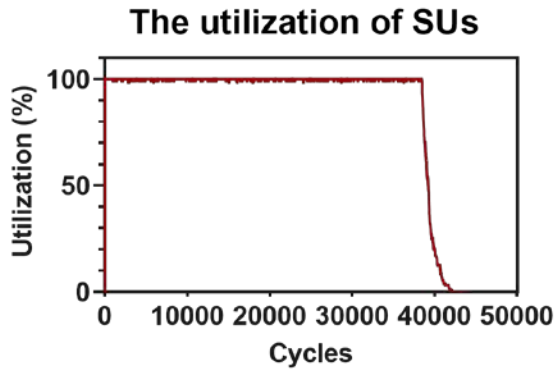
Traditional Approach (SUs+EUs)

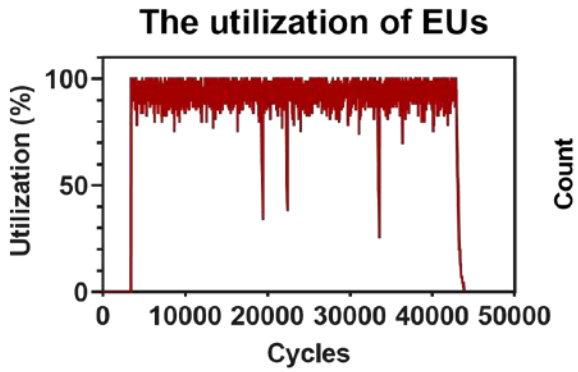


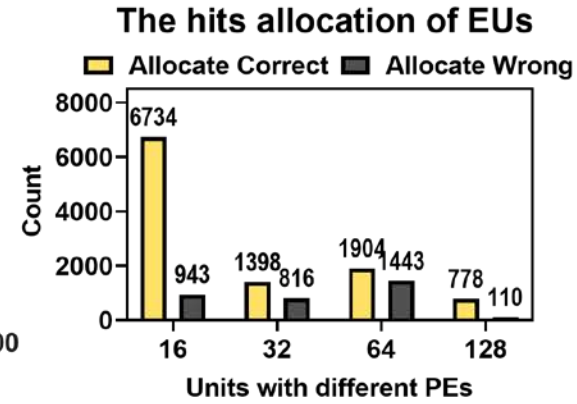




NvWa (SUs+EUs+Schedulers)







NvWa provides high utilization (97.1%, 85.4%) and high correct allocation (87.7%).

Evaluation: Power Consumption

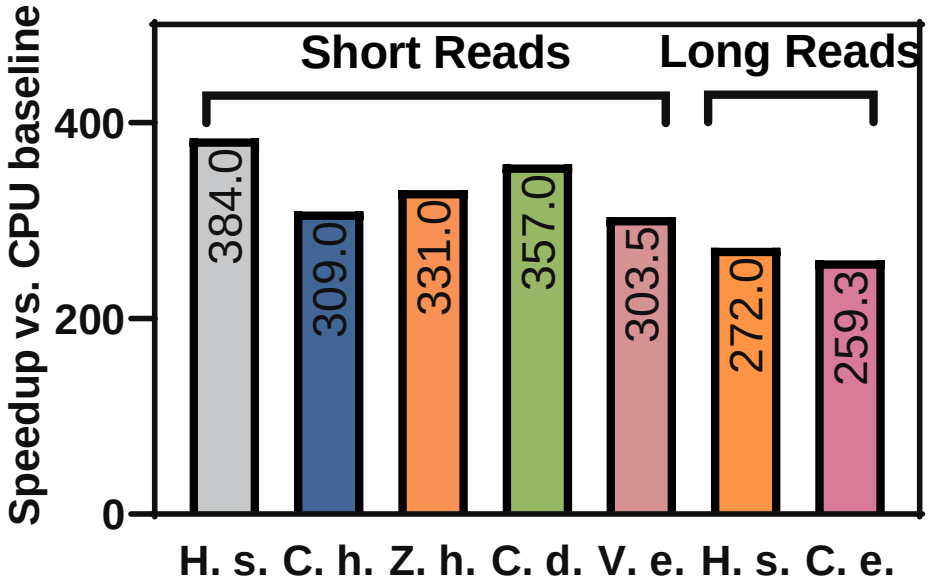
Module	Category	Area(mm^2)	Power(W)
SUs[1]	Logic	0.5	0.36
	Table SRAM	2.16	0.71
EUs	Logic	1.62	0.30
	Table SRAM	21.15	3.614
Seeding Scheduler	SPM	0.13	0.04
	Logic	0.1	0.072
Extension Scheduler	Table SRAM	0.065	0.021
	Logic	0.23	0.165
Coordinator	SRAM Buffer	0.782	0.257
	Logic	0.273	0.215
Total	N/A	27.009	5.754

1. Three key components only have 1.58 mm^2 and 0.77W.

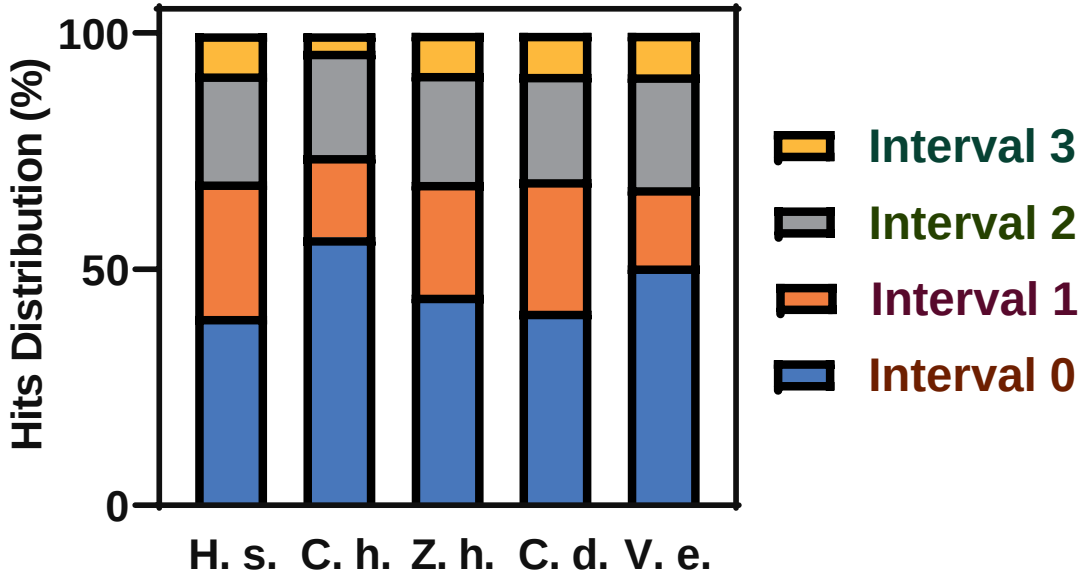
2. NvWa provides 5.85x energy reduction when compared with SOTA accelerators.

Evaluation: Sensitivity Analysis

The performance on multiple datasets



The hits distribution percentage of multiple datasets.



NvWa shows considerable performance gain on multiple datasets(384x~259x).

Presentation Outline

- Background
- Motivation and Challenges
- NvWa Accelerator
- Evaluation
- Conclusion

Conclusion

NvWa leverages multi-phase scheduling mechanisms and architectures to enhance the throughput of sequence alignment accelerators.

- We first identify an execution diversity problem that appears in the sequence alignment process, which limits the end-to-end sequence alignment throughput.

- NvWa aids the accelerator design for the multi-phase scheduling mechanisms and architectures with the followings:
  - **Seeding Scheduler** is targeted to realize the efficient execution of the seeding units.
  - **Extension Scheduler** is designed to maximize the efficient execution of extension units.
  - **Coordinator** buffers the hits finished by the SUs and allocates them reasonably to the idle EUs.

- NvWa can provide 493x, 200x, 12.11x, 2.30x speedup and, 14.21x, 5.60x, 4.34x, 5.85x energy saving over a 16-thread BWA-MEM, GASAL2, GenAx, and GenCache.

NvWa: Enhancing Sequence Alignment Accelerator Throughput via Hardware Scheduling

Thanks for your attention

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES
SKLP
处理器芯片全国重点实验室
State Key Lab of Processors, ICT, CAS
中国科学院大学
University of Chinese Academy of Sciences
XPU-driven Research on Sys and Arch.