MARS

Processing-In-Memory Acceleration of Raw Signal Genome Analysis Inside the Storage Subsystem

Melina Soysal

Konstantina Koliogeorgi   Can Firtina   Nika Mansouri Ghiasi

Rakesh Nadig   Haiyu Mao   Geraldo F. Oliveira   Yu Liang   Klea Zambaku

Mohammad Sadrosadati   Onur Mutlu

SAFARI   ETH zürich

Outline

Background

Motivation and Goal

MARS

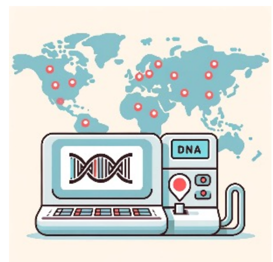
Evaluation

Conclusion

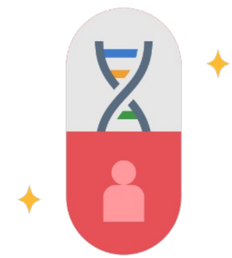
Genome Sequence Analysis

- Study of genome to understand **genetic variation, species and evolution** has led to groundbreaking advances:

Evolutionary biology



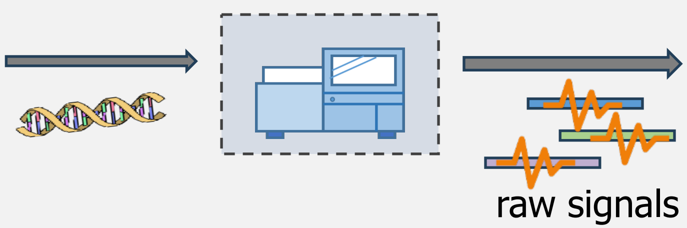
Outbreak tracing



Personalized medicine

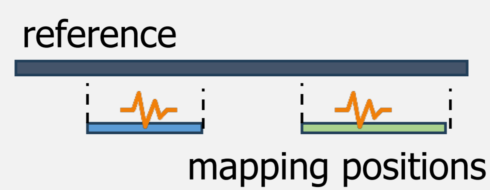
- Genome analysis typically starts with **sequencing** and **mapping**

Sequencing



generates small DNA fragments

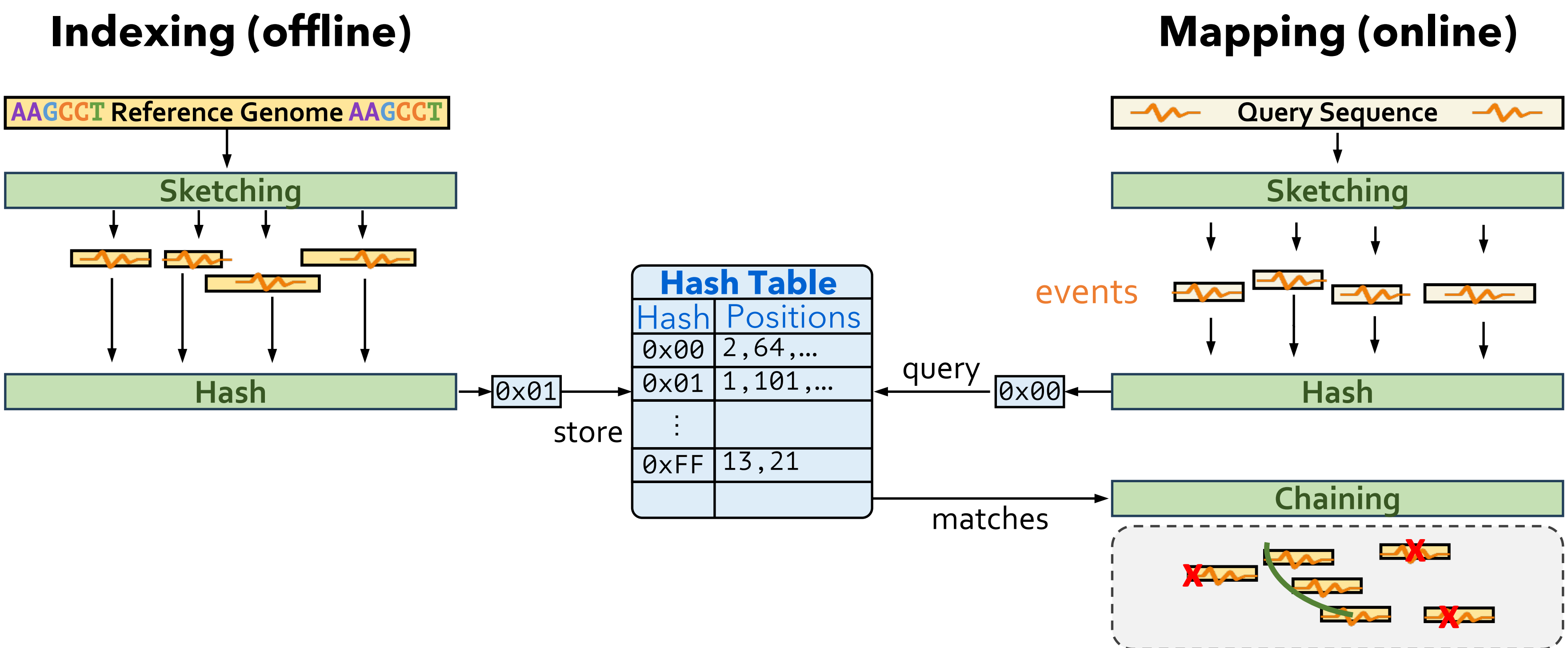
Mapping



aligns signals to reference genome

Raw Signal Genome Analysis (RSGA)

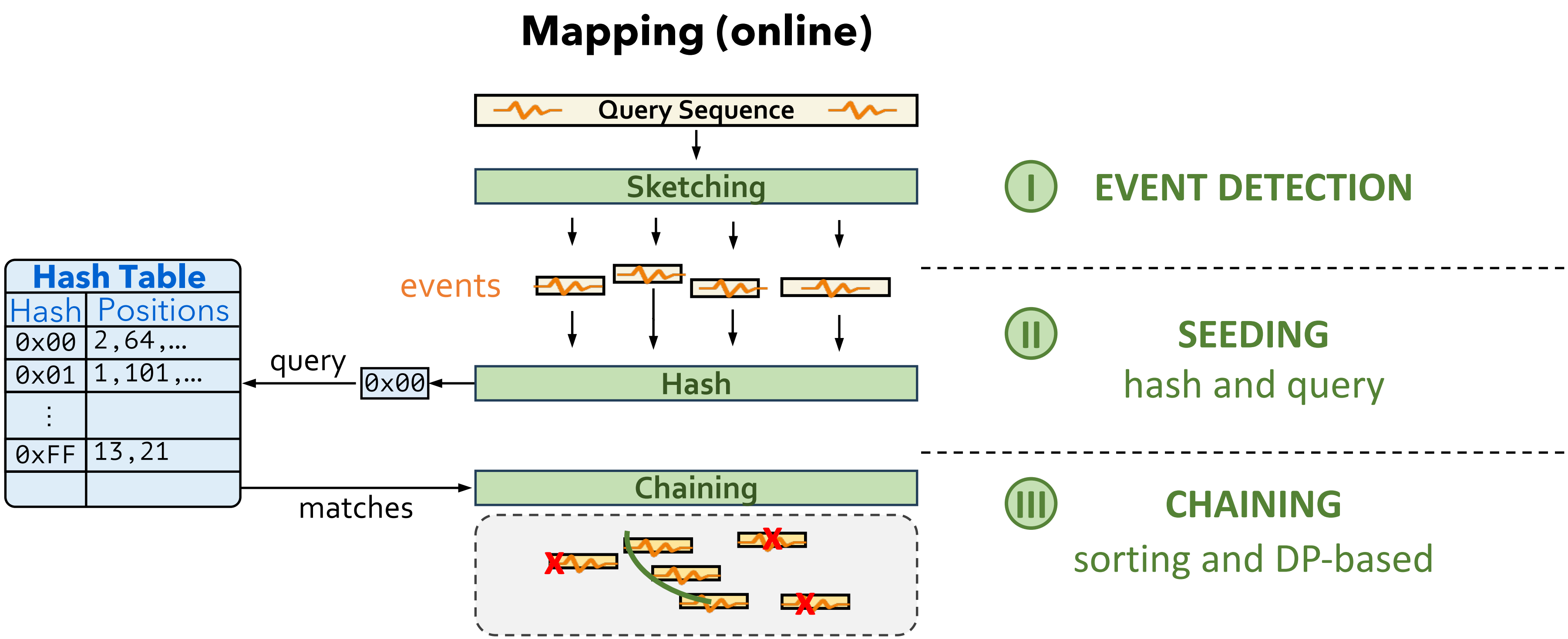
- Raw Signal Genome Analysis (RSGA)
  - novel approach that operates *directly* on **raw electrical signals**
  - exploits **high throughput** of modern sequencing technologies
  - enables **real-time** analysis
- High Level Overview of RSGA Read Mapping



Deep Dive on mapping following

Raw Signal Genome Analysis (RSGA)

- Raw Signal Genome Analysis (RSGA)
  - novel approach that operates *directly* on **raw electrical signals**
  - exploits **high throughput** of modern sequencing technologies
  - enables **real-time** analysis
- High Level Overview of RSGA Read Mapping



Outline

Background

Motivation and Goal

MARS

Evaluation

Conclusion

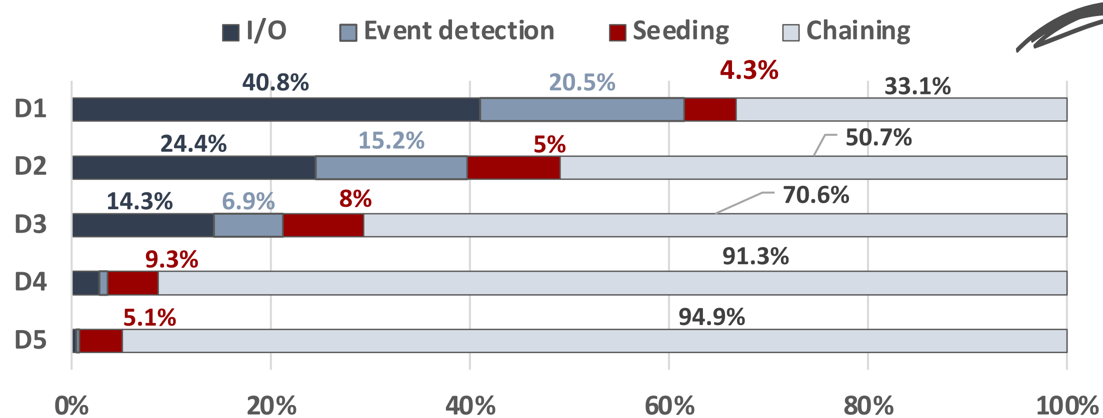
Motivation

Sequencing throughput is growing rapidly
Modern sequencers can generate 100Ks of signal samples per second

RSGA struggles to keep up with increasing real-time requirements
Software pipelines cannot keep up with the pace of raw signal generation

There is a critical need for acceleration

Goal: Identify key performance bottlenecks in the RSGA pipeline
Target the most time- and energy-intensive steps for hardware acceleration

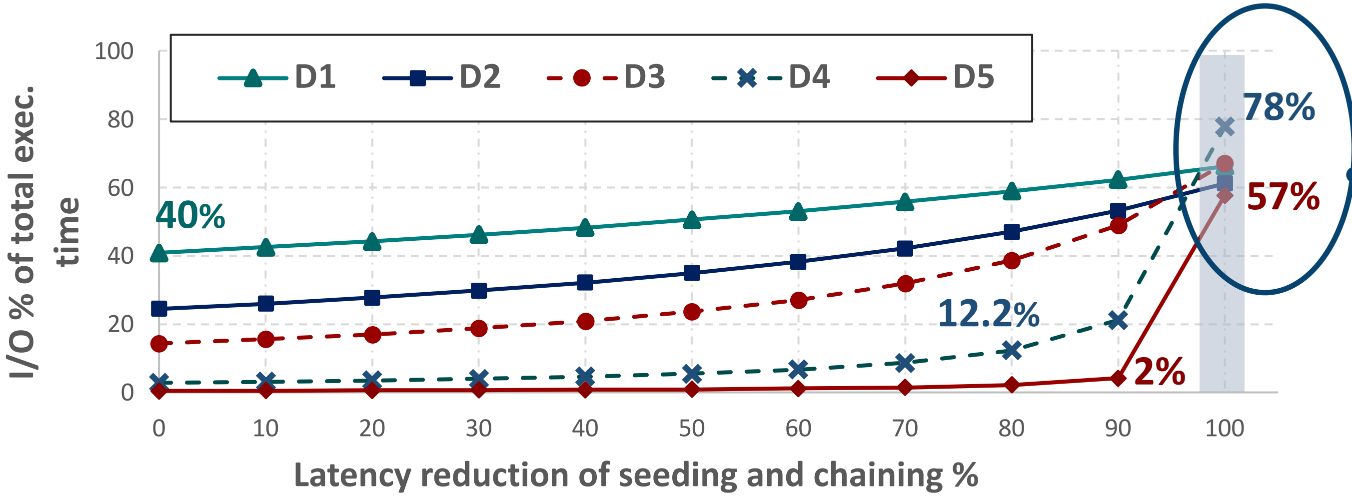


Chaining & **seeding** consistently dominate runtime

Event detection & **I/O overhead** are also non-negligible

Impact of I/O data movement

We model **I/O data movement overhead** when accelerating commonly targeted steps in the pipeline: **seeding** and **chaining**



As compute latency is reduced, **I/O** becomes the **dominant contributor** to total execution time

I/O data movement from SSD becomes dominant overhead as compute steps are accelerated in RSGA

Our Goal

*Design a **high-performance** and **energy-efficient** system for RSGA by addressing both the **I/O data movement** and **computational bottlenecks***

Outline

Background

Motivation and Goal

MARS

Evaluation

Conclusion

MARS: PIM Acceleration of RSGA Inside Storage

- **First In-Storage-Processing system for *RSGA***
- **Key Idea:** Enable multiple Processing-In-Memory paradigms within the SSD and leverage high SSD-internal bandwidth to create a highly-parallel and energy-efficient system for RSGA
- Co-design between **targeted software modifications** and **dedicated hardware components** within the **storage system**

Targeted software modifications to optimize RSGA for in-storage executions

- Tailored to SSD **parallelization capabilities and architecture**

Processing-In-Memory within the SSD to accelerate RSGA steps

- Processing **using** the SSD-internal DRAM
- Processing **near** the SSD-internal DRAM

MARS Genome Analysis Workflow

Software modifications to reduce **computational workload** and **intermediate storage requirements**

Filtering Techniques

- Discard candidates unlikely to result in valid mapping
- Includes frequency and vote-based filtering

Arithmetic Conversion Techniques

- Convert floating-point to fixed-point representations (16 bits)
- Reduces memory footprint and computational cost

Challenge: Preserve accuracy in the presence of raw signal noise

- Requires careful placement of both techniques in the pipeline
- Involves fine-tuning filter parameters

Impact of SW Modification on Accuracy

We conduct an accuracy analysis to show the benefits of our SW improvements and evaluate the usage of fixed-point arithmetic

- **Ground Truth**

  Basecalling (Dorado) and **Read Mapping** (minimap2)

- **Datasets:** SARS-CoV-2, E.coli, Yeast, Green Algae, Human

- **Comparison points:**
  - **RawHash2 (RH2)** *[Firtina+, Bioinformatics'24]*: state-of-the-art RSGA algorithm
  - **MS-CPU$_{\text{Float}}$**: RSGA + filtering techniques
  - **MS_CPU$_{\text{Fixed}}$**: RSGA + filtering + arithmetic conversion techniques

Accuracy Evaluation

	D1 *SARS-CoV-2*			D2 *E.coli*			D3 *Yeast*			D4 *Green Algae*			D5 *Human HG001*		
	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1
RH2	0.9868	0.8735	0.9267	0.9573	0.9009	0.9282	0.9862	0.8412	0.9079	0.9691	0.7015	0.8139	0.8949	0.4054	0.5582
MS-CPU$_{Fixed}$	0.9917	0.9694	0.9803	0.9854	0.9574	0.9712	0.9533	0.9643	0.9588	0.9125	0.9166	0.9141	0.8723	0.6318	0.7300
MS-CPU$_{Float}$	0.9939	0.9796	0.9867	0.9893	0.9616	0.9753	0.9551	0.9655	0.9603	0.9254	0.9438	0.9354	0.8763	0.6729	0.7612

Key Observations

Accuracy Evaluation

	D1 *SARS-CoV-2*			D2 *E.coli*			D3 *Yeast*			D4 *Green Algae*			D5 *Human HG001*		
	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1
RH2	0.9868	0.8735	0.9267	0.9573	0.9009	0.9282	0.9862	0.8412	0.9079	0.9691	0.7015	0.8139	0.8949	0.4054	0.5582
MS-CPU_{Fixed}	0.9917	0.9694	0.9803	0.9854	0.9574	0.9712	0.9533	0.9643	0.9588	0.9125	0.9166	0.9141	0.8723	0.6318	0.7300
MS-CPU_{Float}	0.9939	0.9796	0.9867	0.9893	0.9616	0.9753	0.9551	0.9655	0.9603	0.9254	0.9438	0.9354	0.8763	0.6729	0.7612

Key Observations

- MS-CPU **improves recall and F1-score** over RH2

  *Our filtering techniques are **effective in eliminating ambiguous and redundant** candidate matches & allow pipeline to **focus on more informative regions** that are more likely to represent the correct match*

Accuracy Evaluation

	D1 *SARS-CoV-2*			D2 *E.coli*			D3 *Yeast*			D4 *Green Algae*			D5 *Human HG001*		
	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1
RH2	0.9868	0.8735	0.9267	0.9573	0.9009	0.9282	0.9862	0.8412	0.9079	0.9691	0.7015	0.8139	0.8949	0.4054	0.5582
MS-CPU$_{Fixed}$	0.9917	0.9694	0.9803	0.9854	0.9574	0.9712	0.9533	0.9643	0.9588	0.9125	0.9166	0.9141	0.8723	0.6318	0.7300
MS-CPU$_{Float}$	0.9939	0.9796	0.9867	0.9893	0.9616	0.9753	0.9551	0.9655	0.9603	0.9254	0.9438	0.9354	0.8763	0.6729	0.7612

Key Observations

- MS-CPU **improves recall and F1-score** over RH2

  *Our filtering techniques are **effective in eliminating ambiguous and redundant** candidate matches & allow pipeline to **focus on more informative regions** that are more likely to represent the correct match*

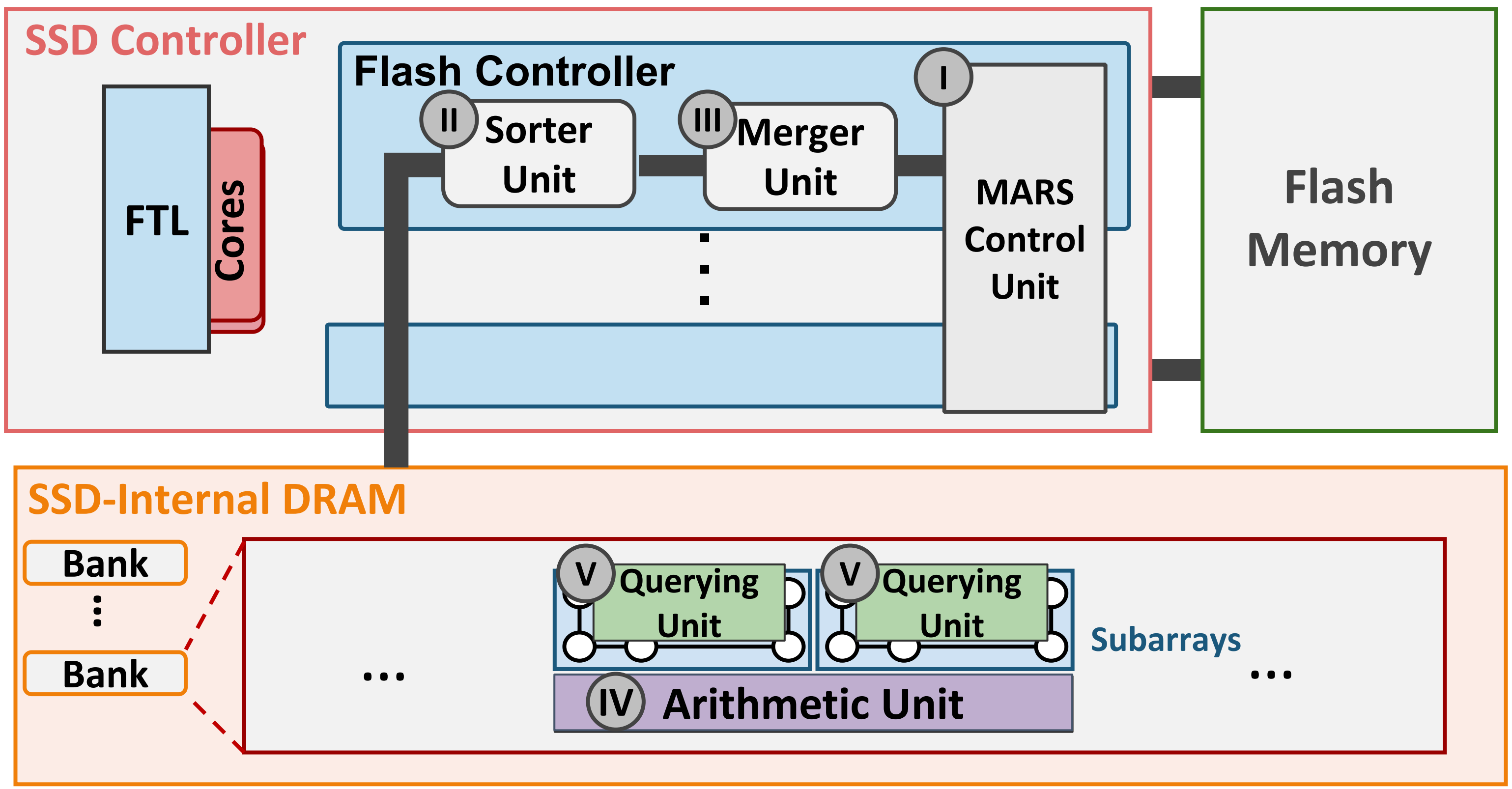
- Using **fixed-point arithmetic** in MS-CPU only leads to minimal accuracy loss

  *We apply **earlier normalization & quantization** to ensure fixed-point arithmetic has no significant impact on our results*

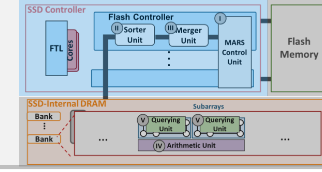
MARS Architecture & System - Overview

In-Storage-Processing system for *RSGA*

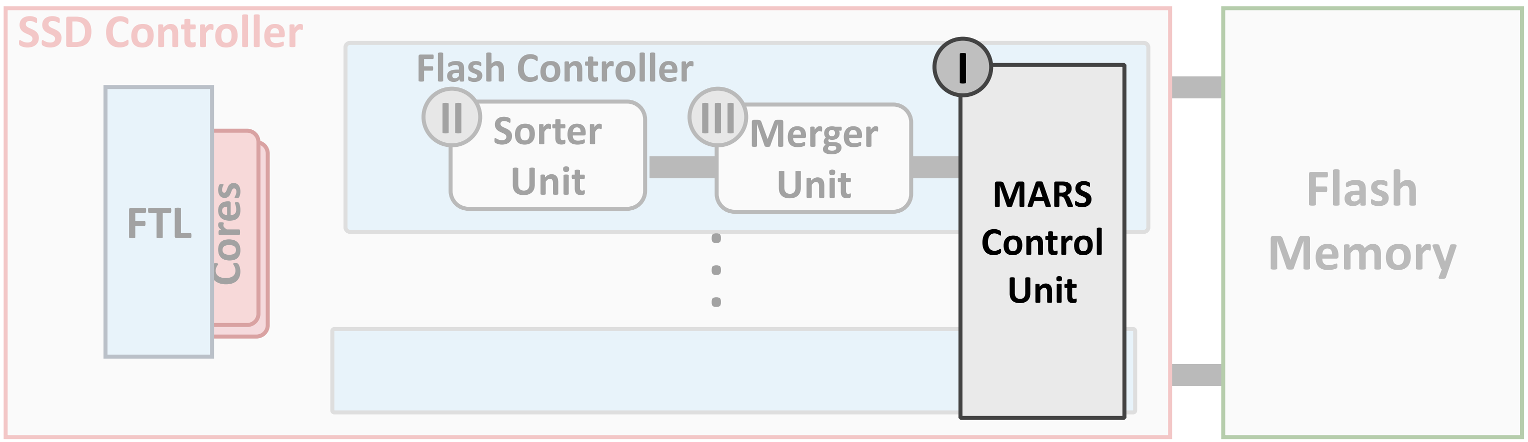
- **Conventional Mode:** SSD performs standard SSD storage operations
- **Accelerator Mode:** SSD executes the full RSGA pipeline without host intervention



MARS Architecture & System (I/IV)



SSD Controller Components

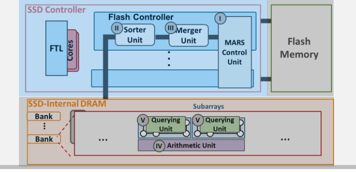


I

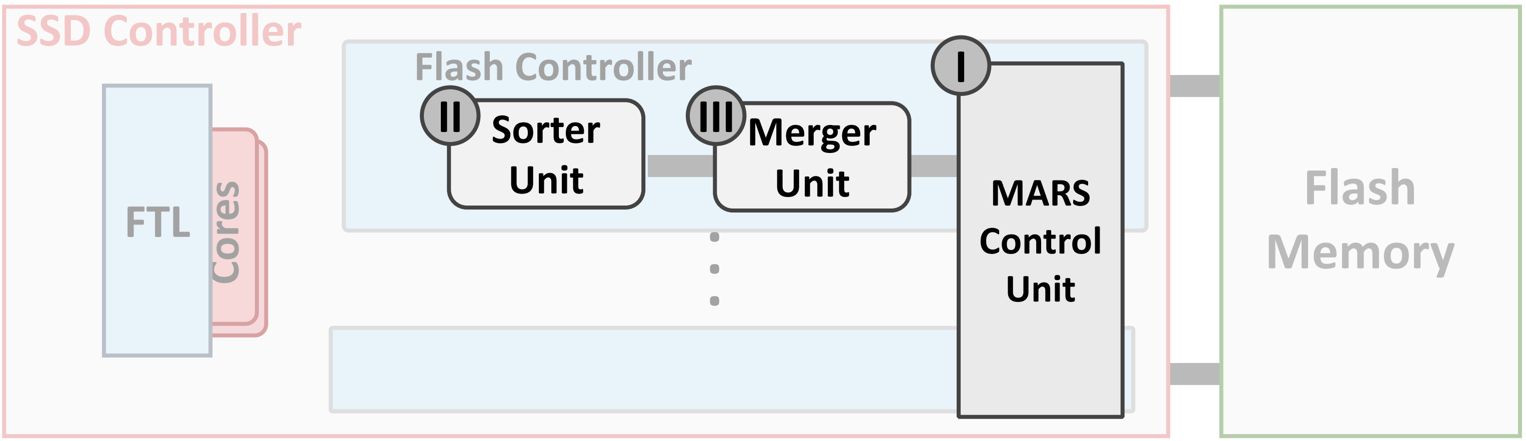
MARS Control Unit

Finite State Machine that orchestrates pipeline execution, control and data flow

MARS Architecture & System (II/IV)



SSD Controller Components



I MARS Control Unit

Finite State Machine that orchestrates pipeline execution, control and data flow

II Sorter Unit

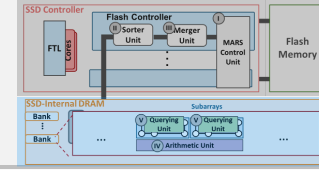
Performs parallel sorting of sequences using dedicated logic

III Merger Unit

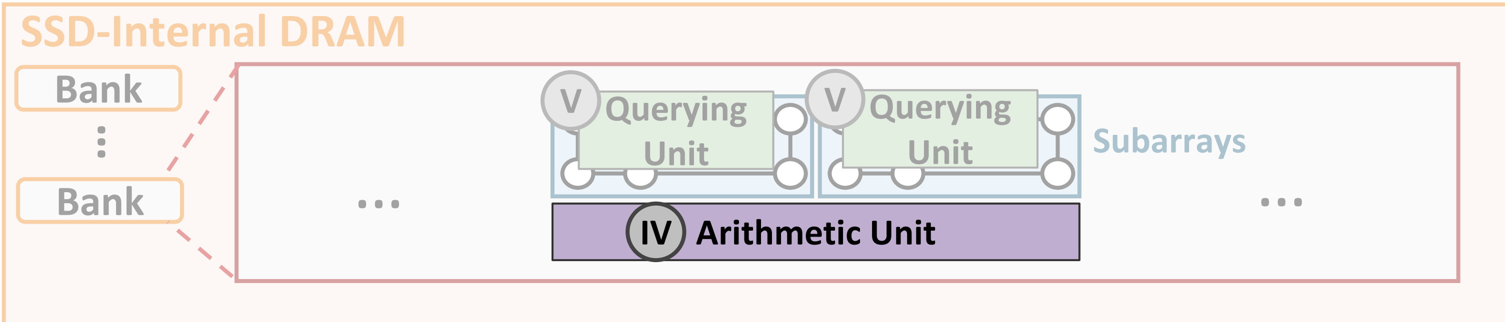
Combines sorted subsequences into longer sorted ones

Paradigm: Implements Processing Near DRAM

1 Sorter-Merger pair per flash controller



SSD-Internal DRAM Components



IV

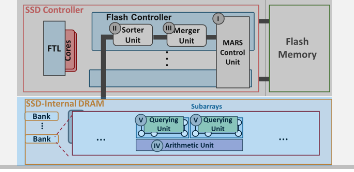
Arithmetic Unit

Performs arithmetic and logical operations required for RSGA

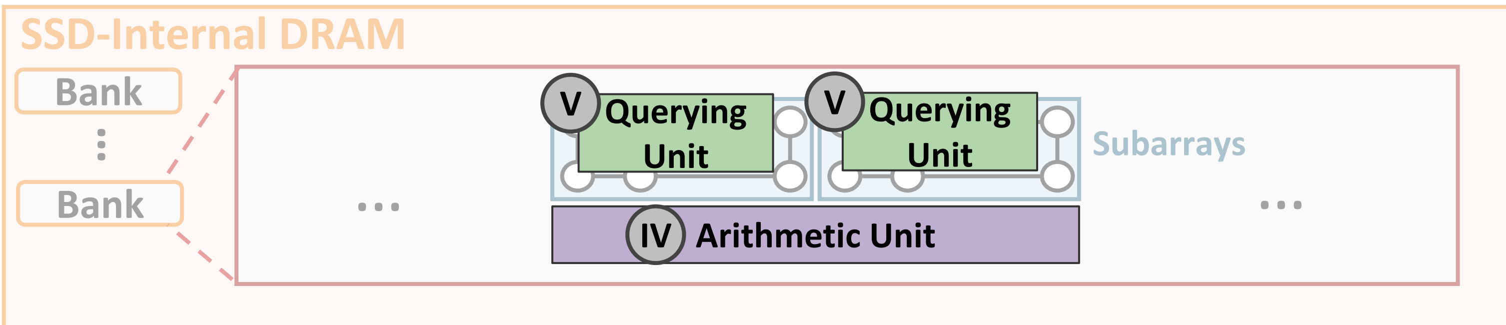
Paradigm: Implements Processing Near DRAM

Design: Based on the FULCRUM architecture for leveraging DRAM subarray proximity for low-latency parallel arithmetic operations

1 per two subarrays



SSD-Internal DRAM Components



IV

Arithmetic Unit

Performs arithmetic and logical operations required for RSGA

Paradigm: Implements Processing Near DRAM

Design: Based on the FULCRUM[1] architecture for leveraging DRAM subarray proximity for low-latency parallel arithmetic operations

1 per two subarrays

V

Querying Unit

Performs efficient, in-memory hash table lookups for seeding

Paradigm: Implements Processing Using DRAM

Design: Based on the PLUTO[2] architecture leveraging DRAM row activation for massively parallel hash-table lookup operations

1 per subarray

1: [Lenjani, HPCA'20]; 2: [Ferreira+, MICRO'22]

Mapping RSGA Workflow to MARS Architecture

1 Break down RSGA steps into **fine-grained tasks**

- Arithmetic, querying and sorting operations

Mapping RSGA Workflow to MARS Architecture

1 Break down RSGA steps into **fine-grained tasks**

- Arithmetic, querying and sorting operations

2 Implement MARS as a **pipeline**

- Each task mapped to a dedicated compute unit
  - e.g., multiplications mapped to arithmetic unit

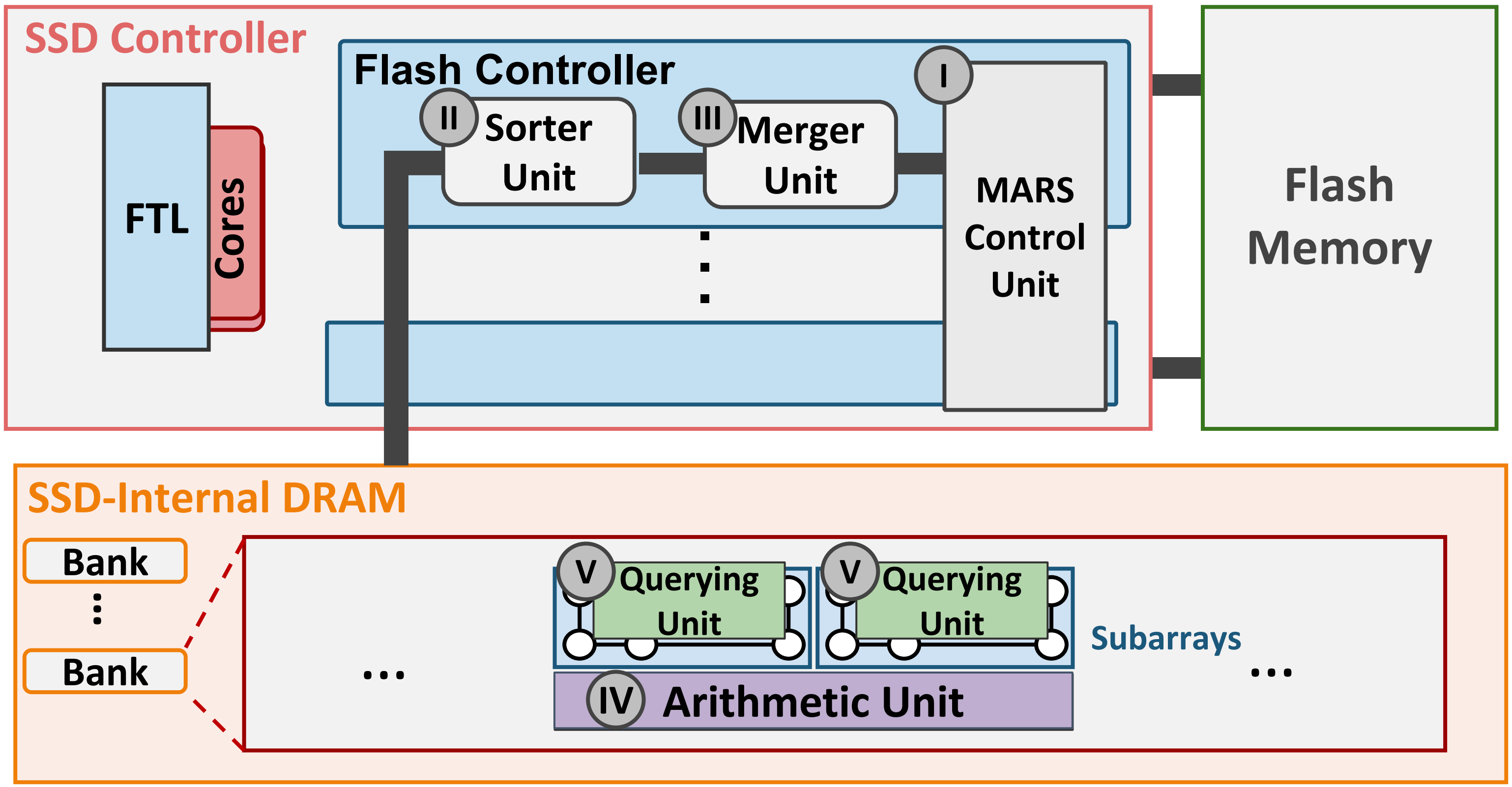
Mapping RSGA Workflow to MARS Architecture

1. Break down RSGA steps into **fine-grained tasks**
   - Arithmetic, querying and sorting operations

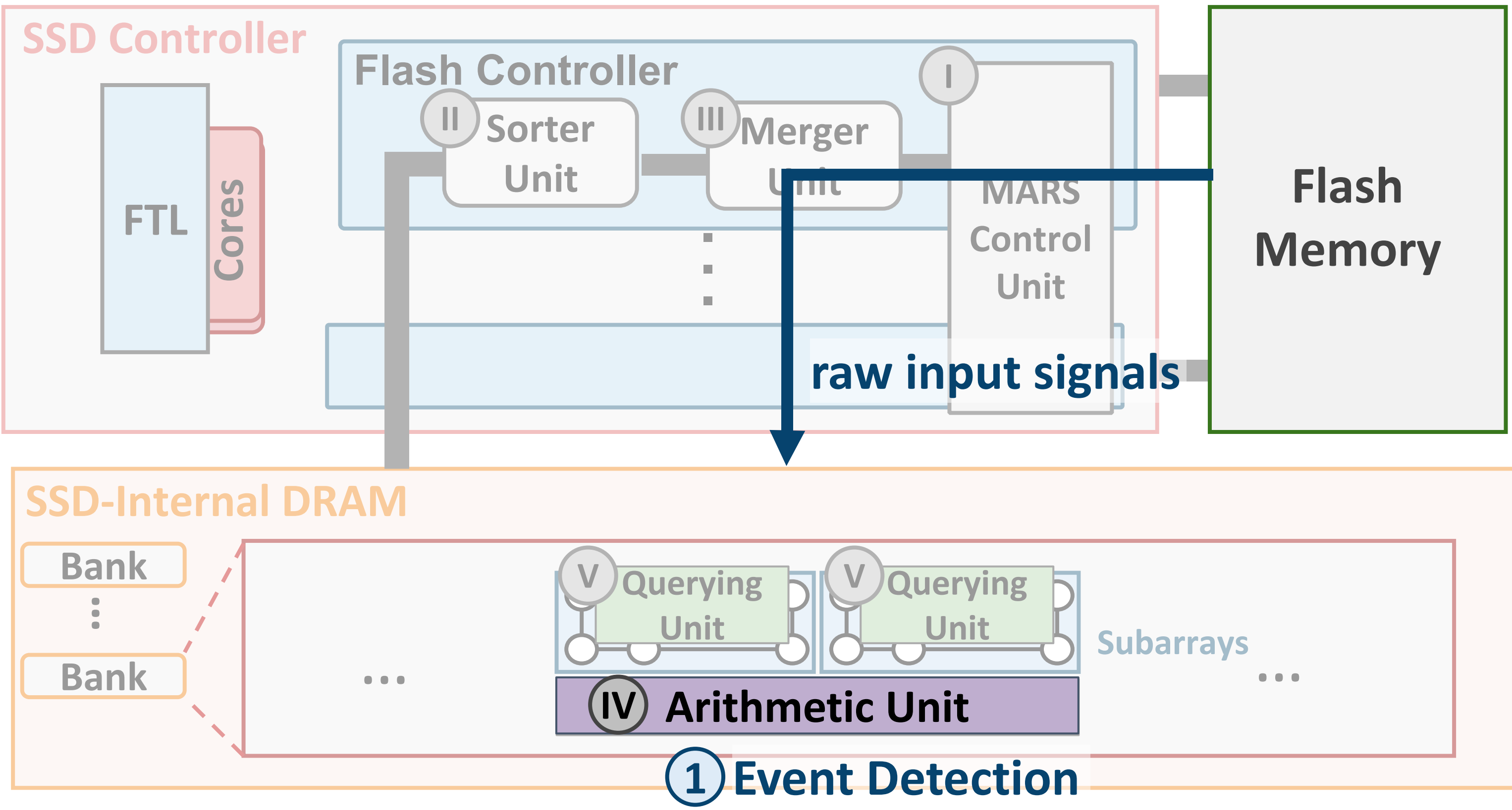
2. Implement MARS as a **pipeline**
   - Each task mapped to a dedicated compute unit
     - e.g., multiplications mapped to arithmetic unit

3. MARS Control Unit encodes pipeline into a **Finite State Machine**
   - Defines a fixed execution sequence
   - Dynamically triggers tasks when input data is ready

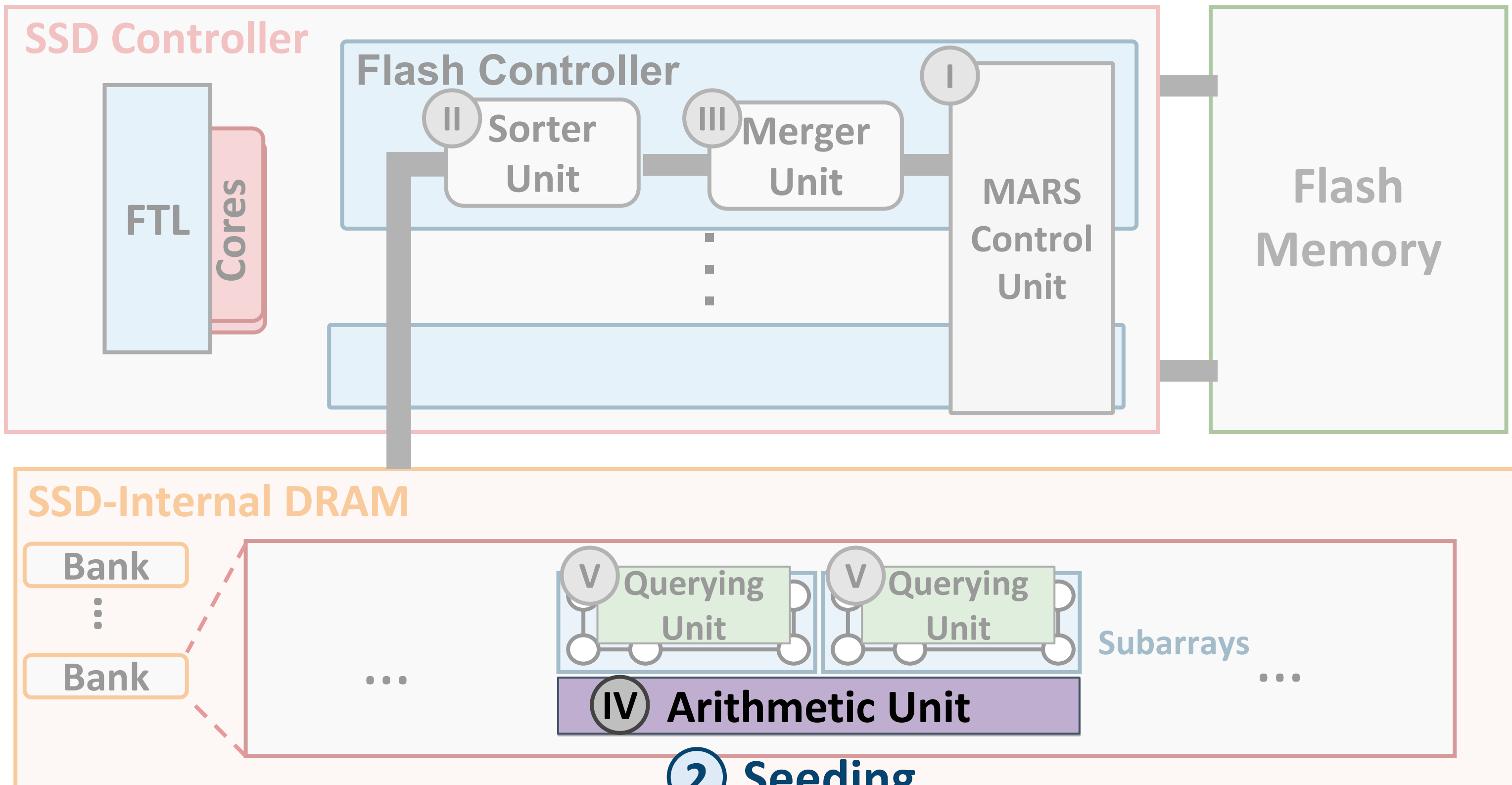
Control and Data Flow



Control and Data Flow (I/VI)



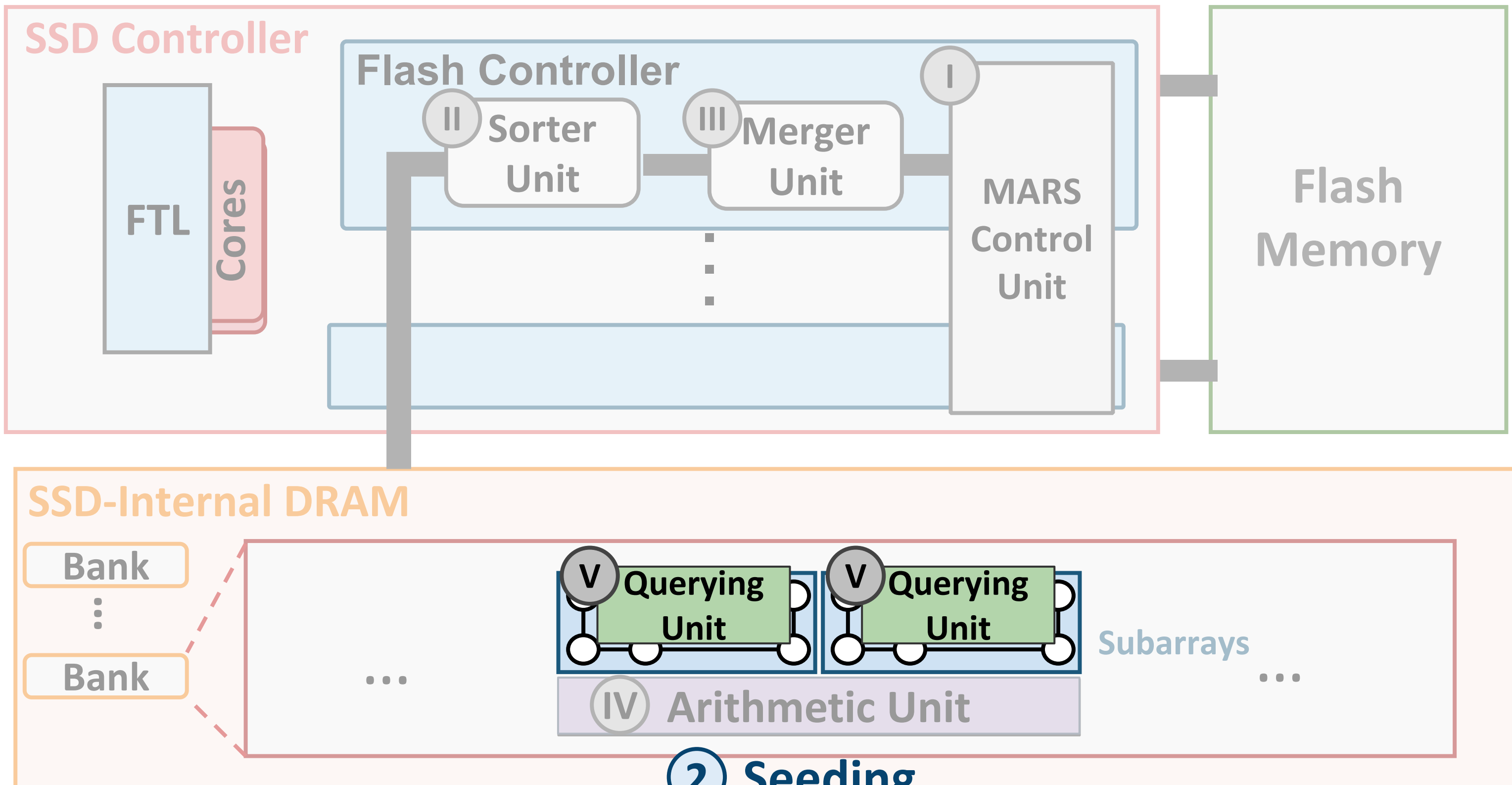
Control and Data Flow (II/VI)



2 **Seeding**

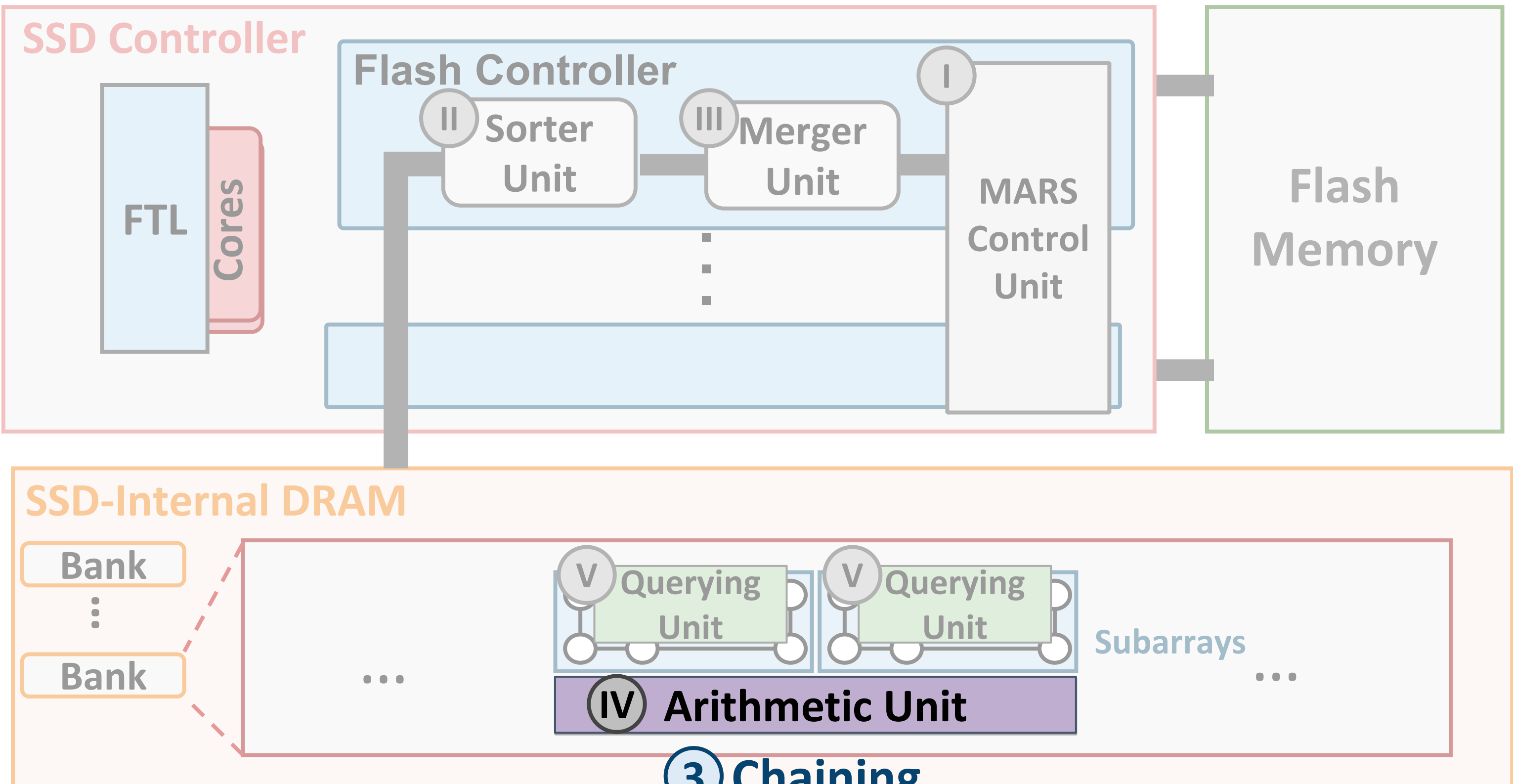
(Hash-value generation & filtering)

Control and Data Flow (III/VI)



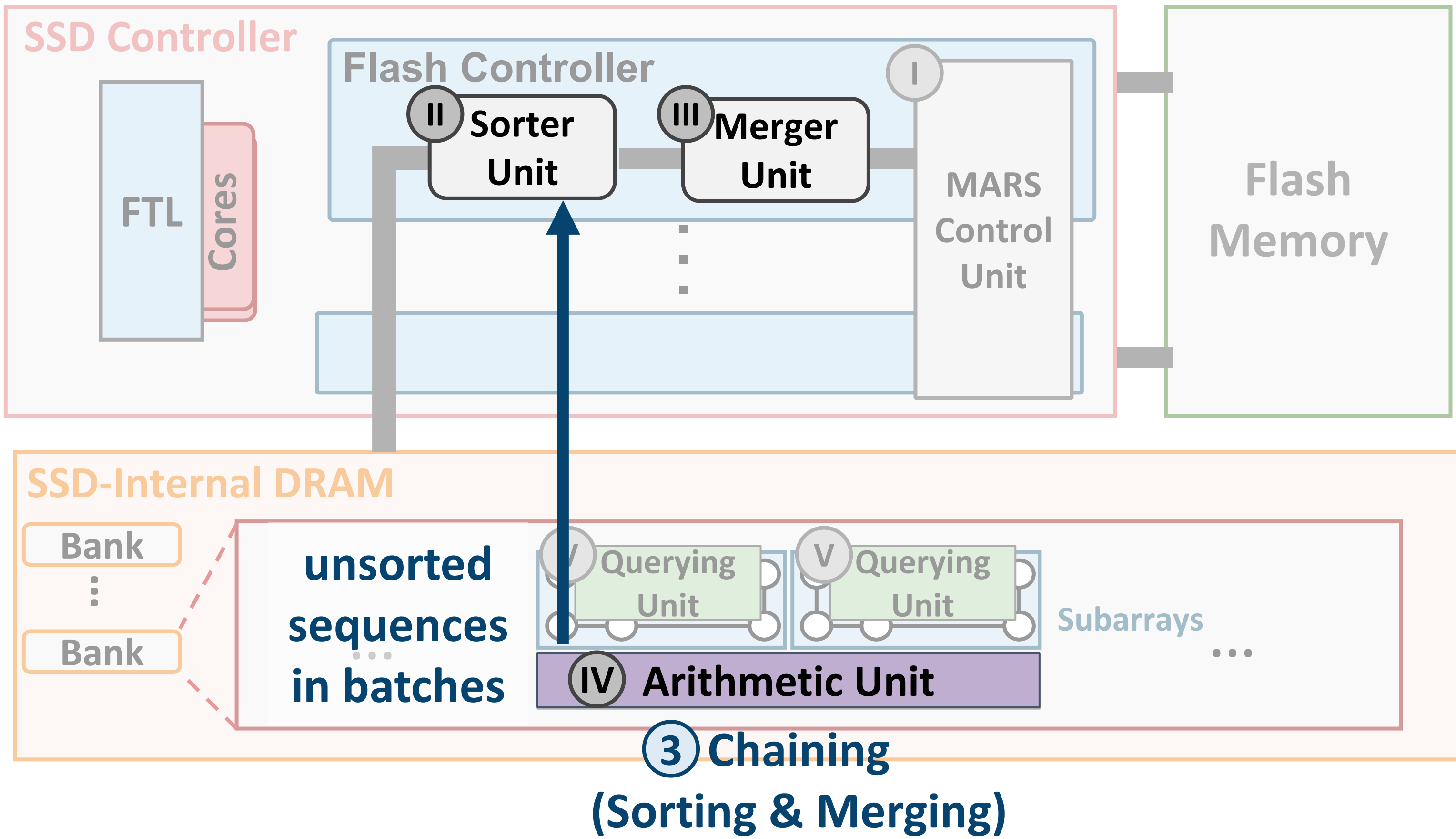
2 Seeding
(Hash-table querying)

Control and Data Flow (IV/VI)

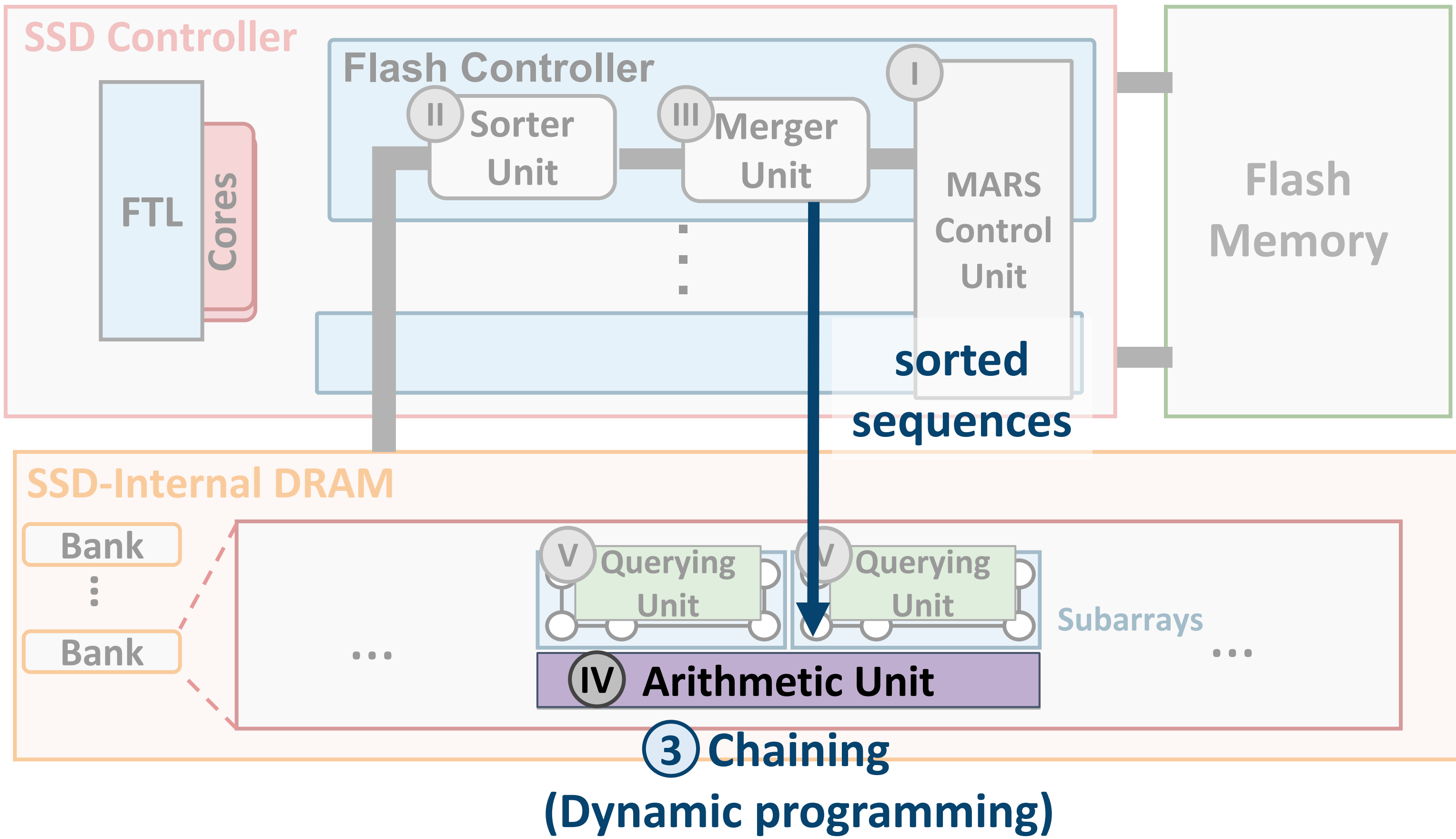


3 Chaining
(Create batches of sequences)

Control and Data Flow (V/VI)



Control and Data Flow (VI/VI)



Computational Units & System Integration

- Arithmetic Unit
  - Processing-Near-Memory
  - Add Logic near the subarrays
  - Column-Selection Latches

- Querying Unit
  - Processing-Using-Memory
  - Add Custom Hash-key Match logic

More details on the Mechanism and System Integration

can be found in the paper

- Sorter and Merger Unit
- Inside the SSD controller
- Processing-Near-Memory
- Bitonic Sorter and Merger

Outline

Background

Motivation and Goal

MARS

Evaluation

Conclusion

Evaluation Methodology (I/II)

We compare MARS against four systems for our performance and energy analysis

BC	GenPIP	RH2	MARS-CPU	MARS
Conventional pipeline with GPU-based **basecalling** (dorado) and CPU-based **read mapping** (minimap2)	**NVM-based hardware-accelerated** pipeline for basecalling & read mapping *[MICRO'22]*	State-of the art **CPU-based RSGA pipeline** *[Bioinformatics'24]*	MARS **software-only RSGA** pipeline executed on CPU	Full MARS **hardware-accelerated** RSGA system inside the **storage subsystem**

Basecalling-based pipeline (BC, GenPIP) | RSGA-based pipeline (RH2, MARS-CPU, MARS)

Evaluation Methodology (II/II)

We use the following hardware setup and simulate elements of our system setup

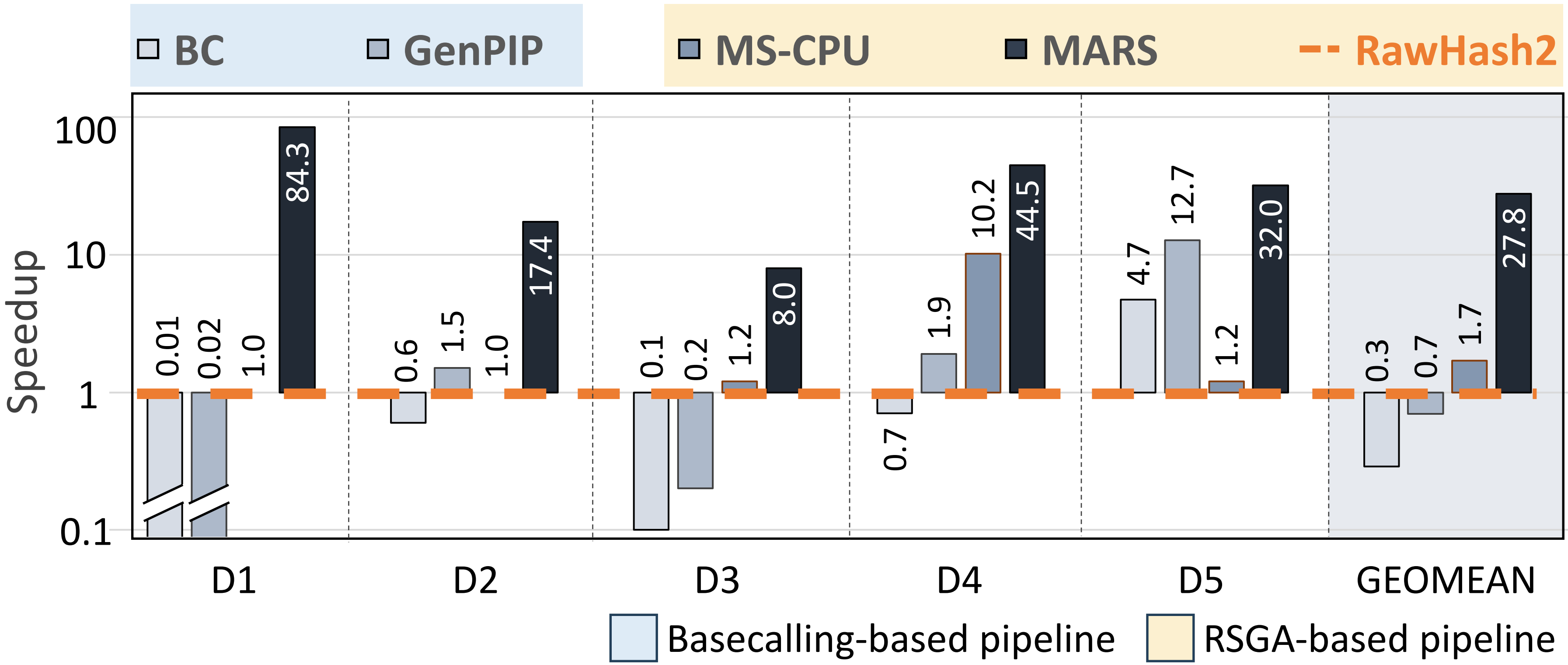
Hardware Components

- **CPU-based baseline**
  - state-of-the-art server with two AMD® EPYC® CPUs with 64 physical cores
  - 1-TB DDR4 DRAM
  - NVIDIA RTX A6000 GPU *(for the BC baseline)*
- **SSD configurations**
  - 8 channels with each 8 flash chips, PCIe 4.0 interface
  - 4GB LPDDR4 DRAM with 16 banks and 512 subarrays

Simulators Used

- Synopsis Design Compiler - for accelerators
- MQSim *[Tavakkol+, FAST'18] -* for SSD-internal operations
- CACTI7 *[Kim+, CAL'15] -* for SSD-internal LPDDR4 DRAM

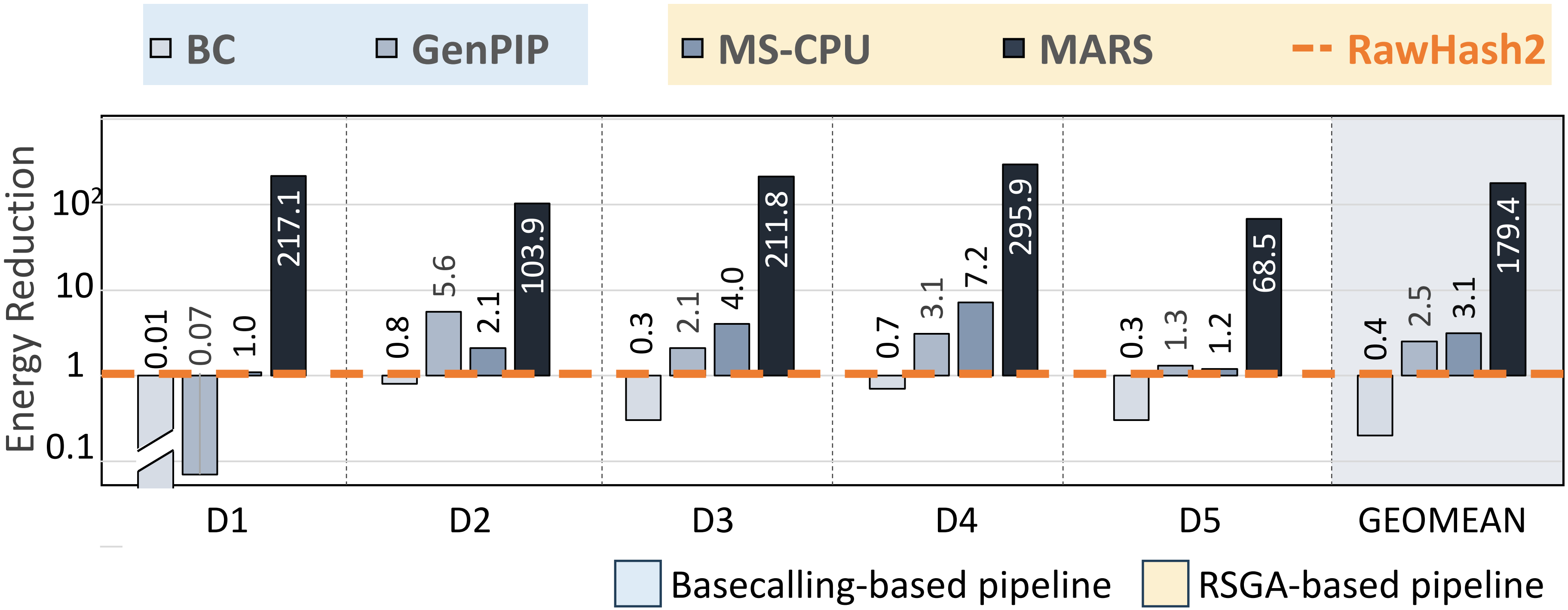
Performance Evaluation



Key Observations

- MARS outperforms all other baselines across all datasets
- MARS outperforms GenPIP adopting PIM outside storage
  We fundamentally solve the I/O data movement overhead problem
- MARS-CPU provide speedup of Ø1.7x over RH2
  Our SW modifications are effective in reducing the computational load

Energy Evaluation



Key Observation

- MARS reduces energy consumption over all other baselines across all datasets

  Improvement stems from

  (1) reduced execution time

  (2) fundamentally addressing the I/O data movement overhead

Outline

Background

Motivation and Goal

MARS

Evaluation

Conclusion

Conclusion

MARS

The *first* **in-storage processing** system for *end-to-end* raw signal genome analysis enables through SW improvements and a HW system inside the storage system

High accuracy

matches basecalling-based pipelines

improves F1 scores over **state-of-the-art RSGA pipeline**

Improves performance

93× over **state-of-the-art software-based basecalling read mapping** pipeline

40× over **state-of-the-art hardware-accelerated basecalling read mapping** pipeline

28× over **state-of-the-art software-based RSGA read mapping** pipeline

Reduces energy consumption

427× over **state-of-the-art software-based read mapping** pipeline

72× over **state-of-the-art hardware-accelerated read mapping** pipeline

180× over **state-of-the-art software-based RSGA read mapping** pipeline

MARS

Processing-In-Memory Acceleration of Raw Signal Genome Analysis Inside the Storage Subsystem

Melina Soysal

Konstantina Koliogeorgi Can Firtina Nika Mansouri Ghiasi

Rakesh Nadig Haiyu Mao Geraldo F. Oliveira Yu Liang Klea Zambaku

Mohammad Sadrosadati Onur Mutlu

SAFARI ETH zürich

BACKUP SLIDES

BACKUP – To be improved

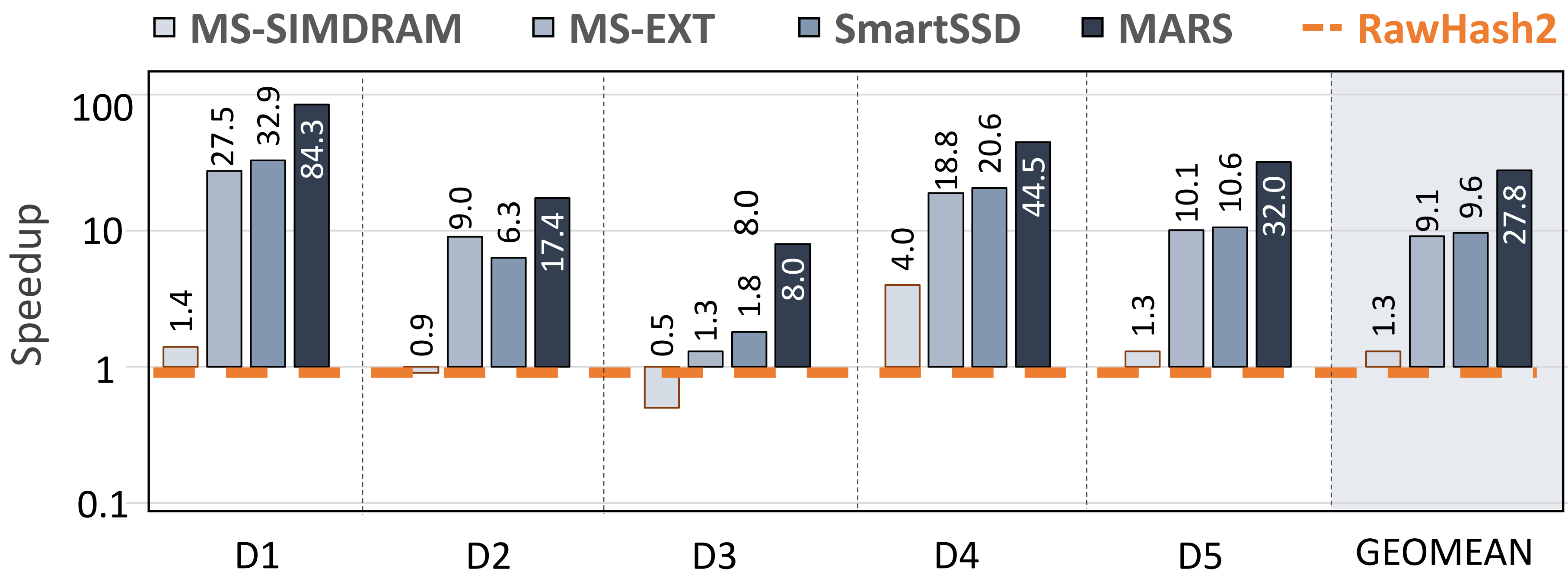
Ablation Study - Methodology

We conduct an ablation study to find the most suitable system setup of MARS

MARS	MARS-SIMDRAM	SmartSSD	MARS-EXTERNAL
Full MARS **hardware-accelerated** RSGA system inside the **storage subsystem**	MARS with **bit-serial PUM-based arithmetic units** instead near-DRAM accelerators	MARS with offloaded sorting using an **external FPGA near the SSD**, replacing in-SSD logic	MARS with **conventional SSD**, sorting on **ASIC near CPU** and **external PIM-based DRAM**

RSGA-based pipeline

Ablation Study - Performance Evaluation



Key Observations

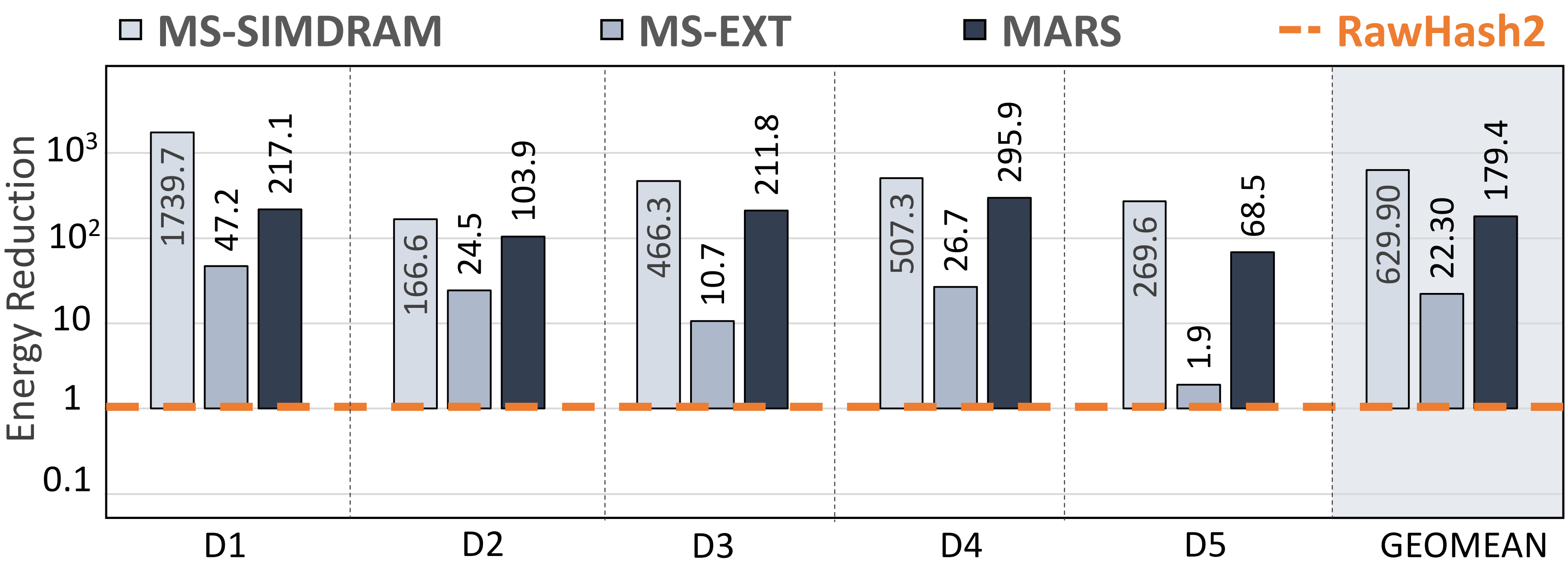
- MARS outperforms all ablation variants across all datasets
- MS-EXT lags behind due to high I/O data movement to storage-external compute

  MS-EXT is not able to fundamentally tackle the I/O data movement overhead

- SmartSSD performs moderately, but is limited due to FPGA interface BW

  MARS as only system combining high compute with fundamentally addressing I/O data movement overhead

Ablation Study - Energy Evaluation



Key Observation

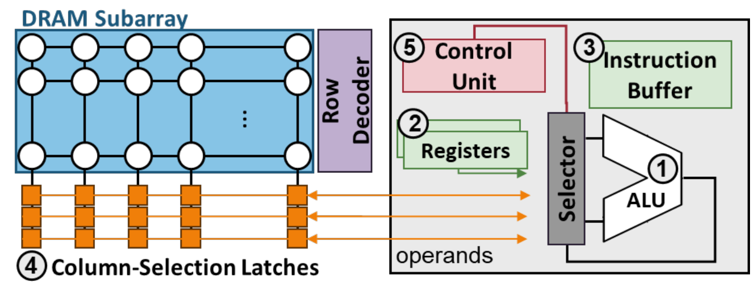
- MS-SIMDRAM achieves highest energy reduction, but lowest performance among the ablation variants

  Small bit-serial arithmetic unit trades off energy consumption for performance

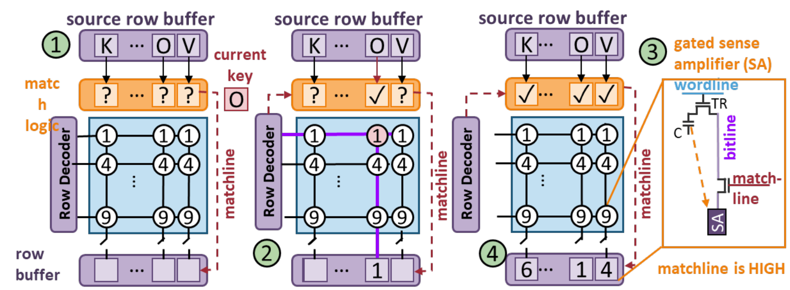
- MS-EXT incurs higher energy cost due to off-chip data movement and increased host involvement
- MARS provides the best performance-energy trade-off

Computational Units

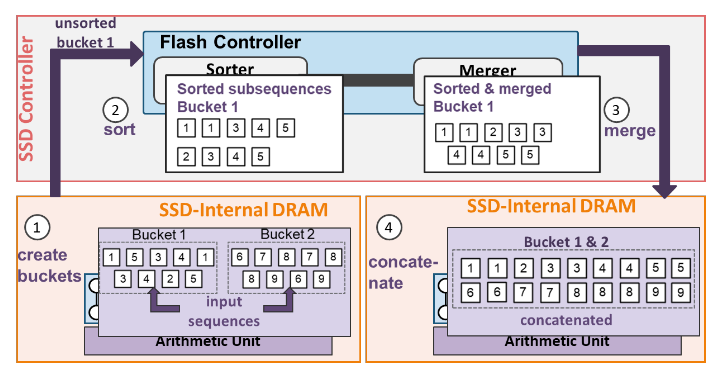
- Arithmetic Unit
  - Processing-Near-Memory
  - Add Logic near the subarrays
  - Column-Selection Latches



- Querying Unit
  - Processing-Using-Memory
  - Add Custom Hash-key Match logic and Gated Sense Amplifiers



- Sorter and Merger Unit
- Inside the SSD controller
- Processing-Near-Memory
- Bitonic Sorter and Merger



MARS detailed workflow