CERES
Center for Unstoppable Computing

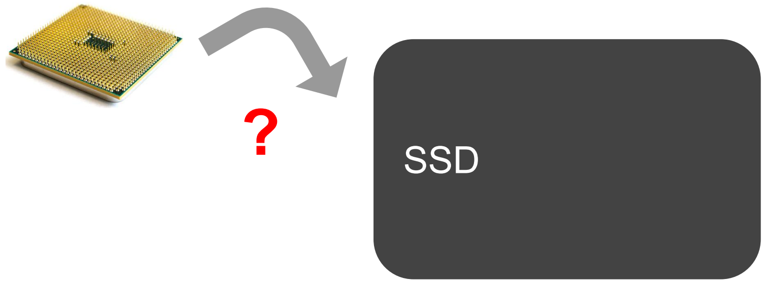
THE UNIVERSITY OF CHICAGO

ASSASIN: Architecture Support for Stream Computing to Accelerate Computational Storage

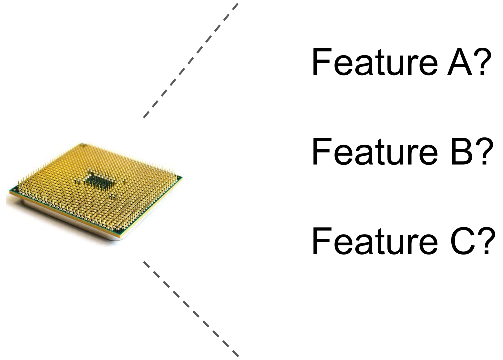
Chen Zou[1], Andrew A. Chien[1,2]
[1]Department of Computer Science, University of Chicago
[2]Mathematics and Computer Science Division, Argonne National Laboratory
{chenzou@, achien@cs.}uchicago.edu
people.cs.uchicago.edu/~aachien/lssg/research/

Lightning: Systematically Architect Compute Inside SSD

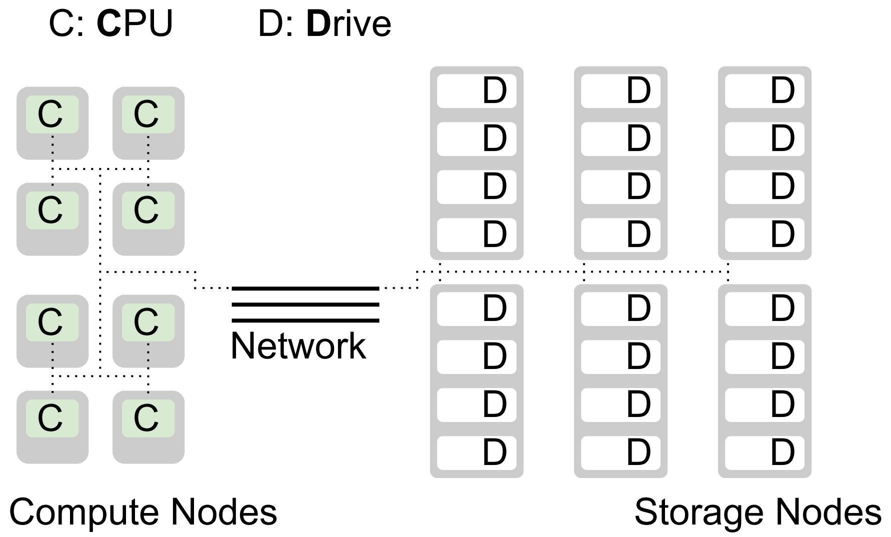
How to integrate compute into SSD?



What should the compute look like?



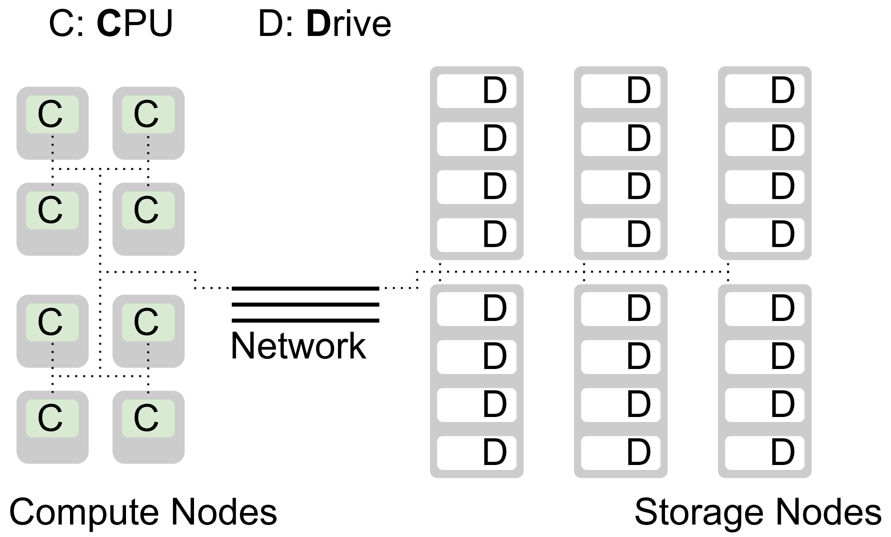
Datacenter Storage



Datacenter Storage

Disruptive Storage technology:

- Flash, NVM, …
- Continuous bandwidth scaling
- Data center network bottleneck

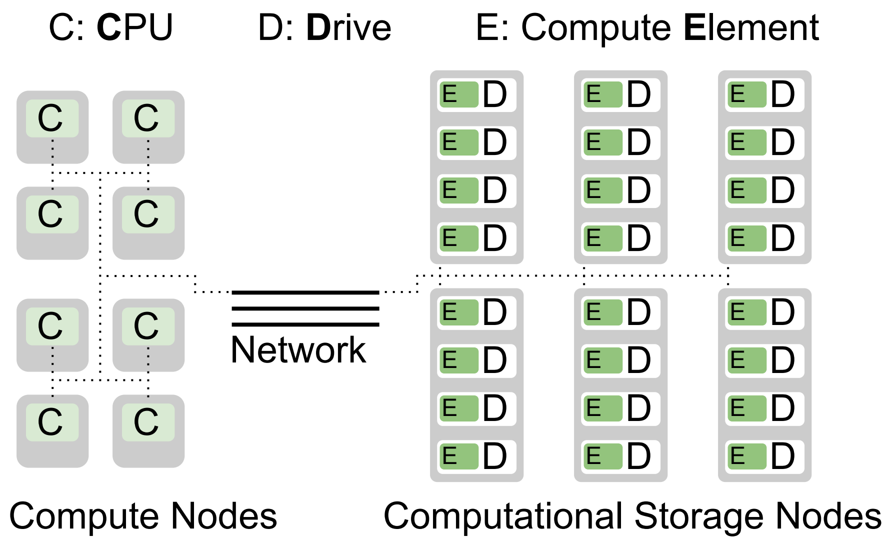


Datacenter Storage

Disruptive Storage technology:

- Flash, NVM, …
- Continuous bandwidth scaling
- Data center network bottleneck

Computational Storage:



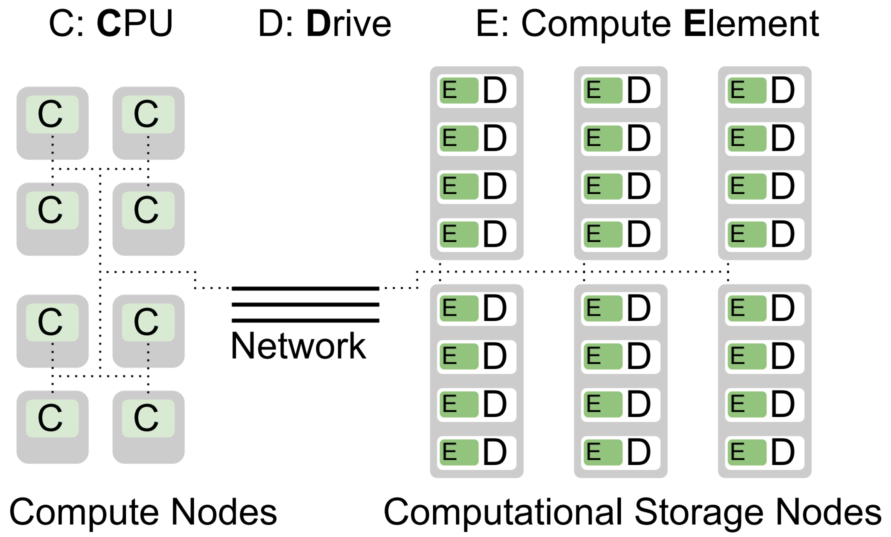
Datacenter Storage

Disruptive Storage technology:

- Flash, NVM, …
- Continuous bandwidth scaling
- Data center network bottleneck

Computational Storage:

- Offload functions in/near storage to keep up with storage bandwidth
- Filesystem, Data Analytics, …



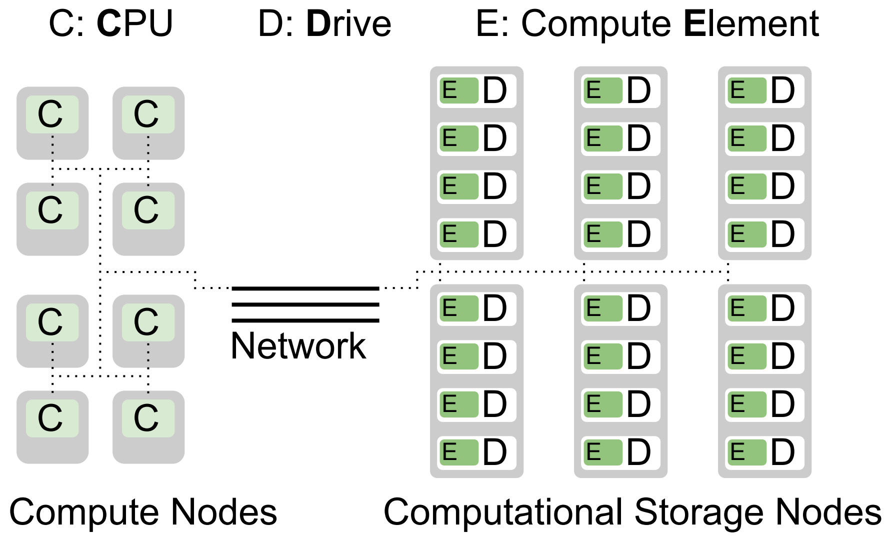
Datacenter Storage

Disruptive Storage technology:

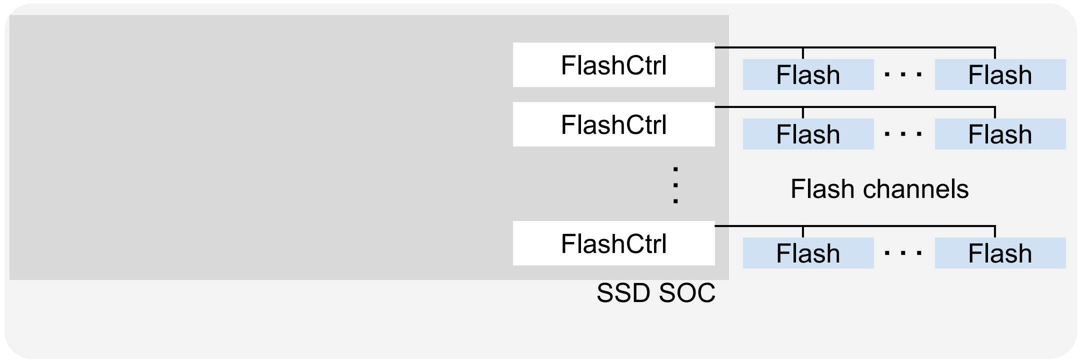
- Flash, NVM, …
- Continuous bandwidth scaling
- Data center network bottleneck

Computational Storage:

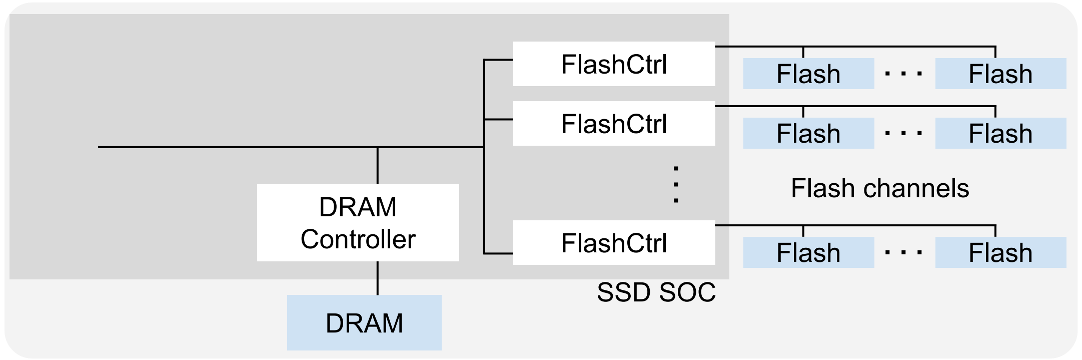
- Offload functions in/near storage to keep up with storage bandwidth
- Filesystem, Data Analytics, …
- Largely reduce data moved to CPUs



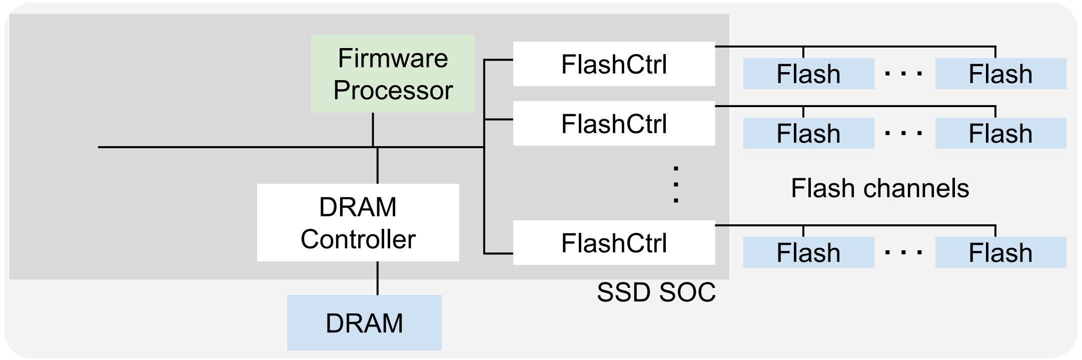
Single Computational SSD



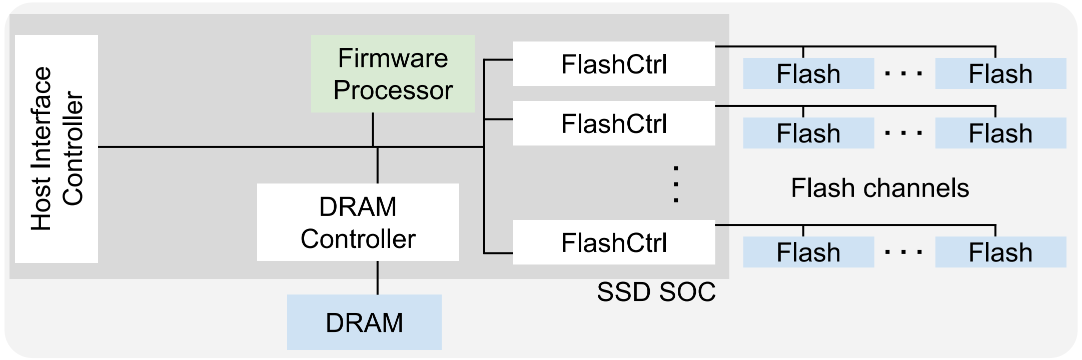
Single Computational SSD



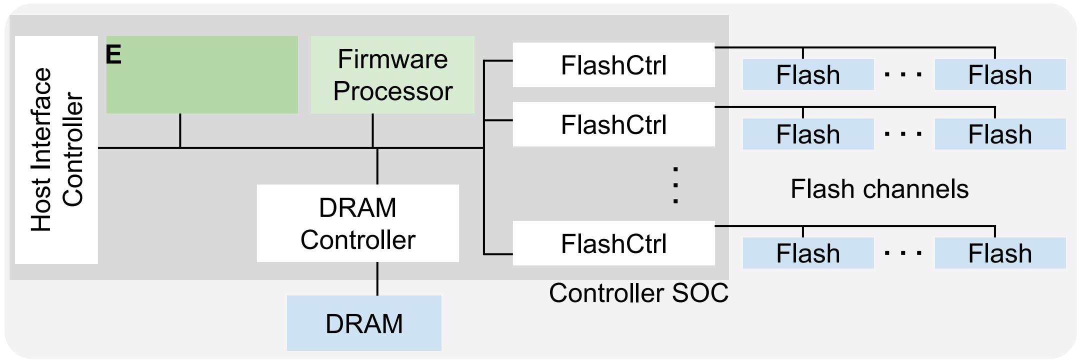
Single Computational SSD



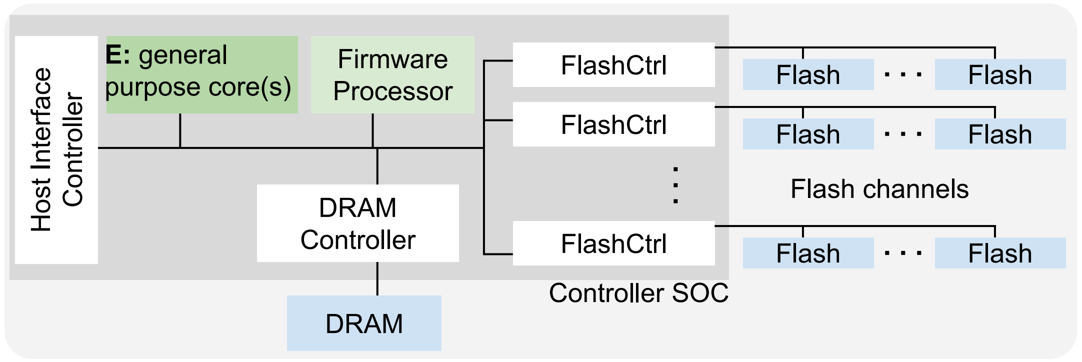
Single Computational SSD



Single Computational SSD



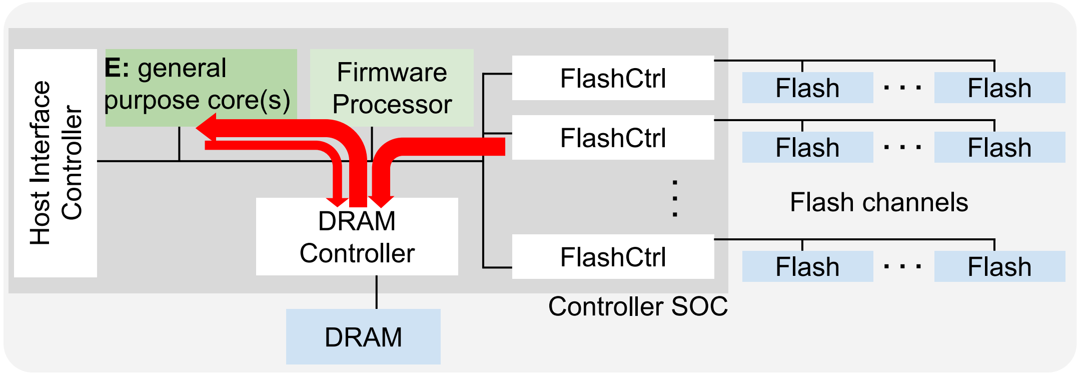
Single Computational SSD



State-of-the-art generic computational SSD architecture

Captures design of [YourSQL, Biscuit, Summarizer, IceClave, BlockIF]

Problem: Memory Wall Inside Computational SSD



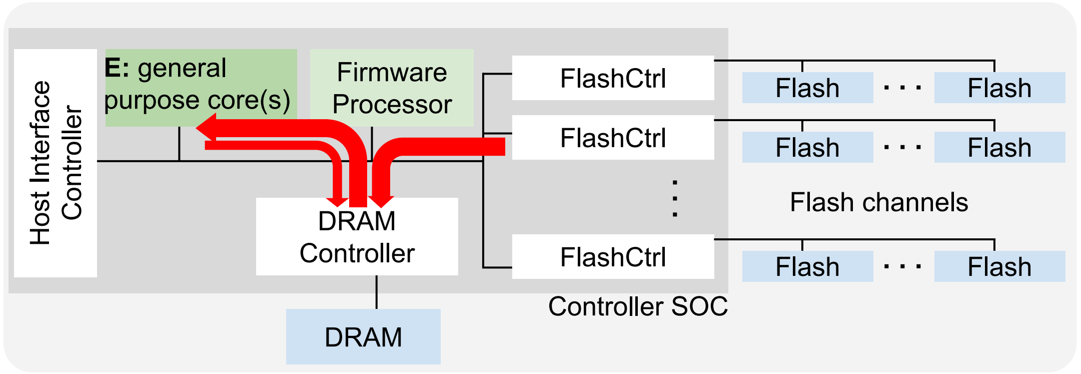
Data flows Flash => DRAM => ComputeElement=> DRAM

High DRAM traffics

Caches don’t help (because of workload)

+Management traffic for Firmware/FTL

Problem: Memory Wall Inside Computational SSD



With increasing Flash BW:
SSD DRAM bottleneck!

An 8-channel flash array of 12.8 GB/s would require at least 25.6 GB/s DRAM bandwidth.
> DDR4 channel
>> LPDDR4 (in SSD products)

Data flows Flash => DRAM => ComputeElement=> DRAM

High DRAM traffics

Caches don’t help (because of workload)

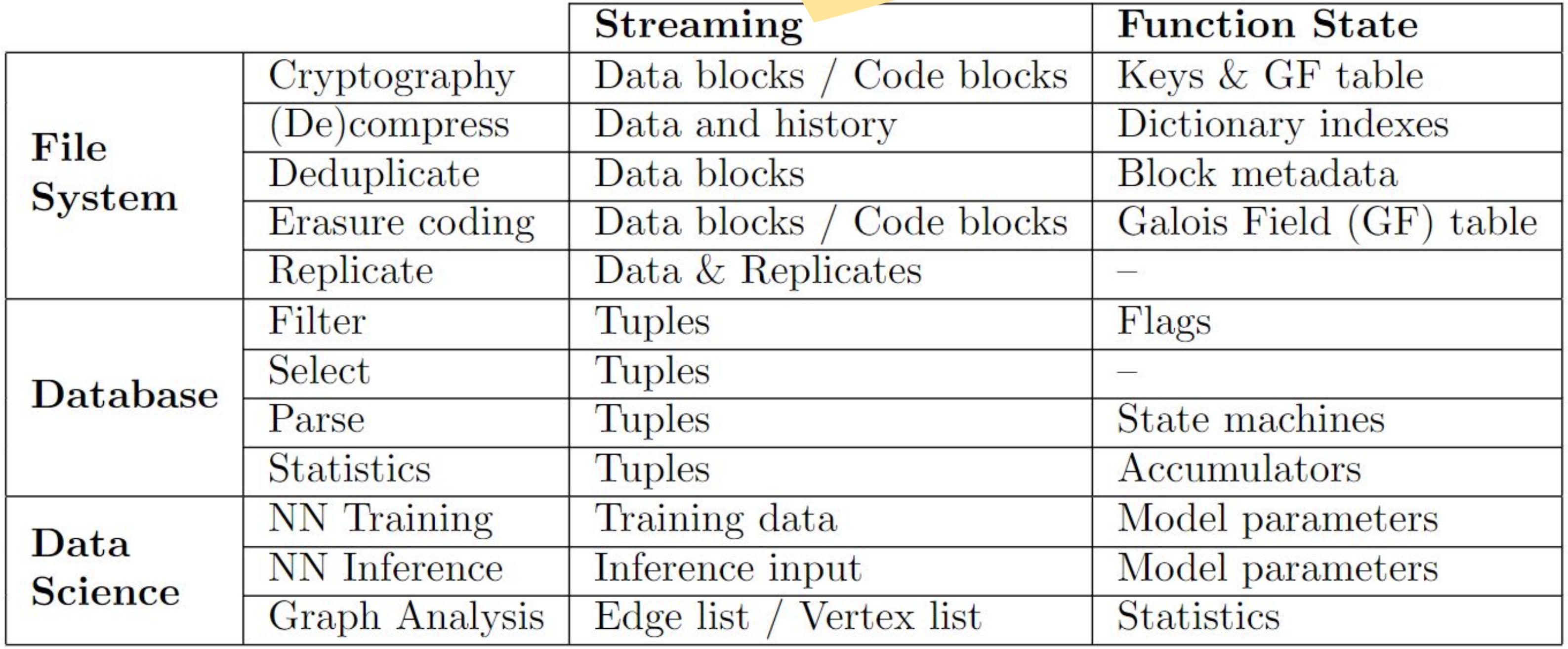
+Management traffic for Firmware/FTL

How to eliminate bottleneck and deliver high performance?

Offloads are often Streaming Operators

via systematic survey of common storage applications and offloads in industry and research.
[see paper]

Little Temporal Reuse

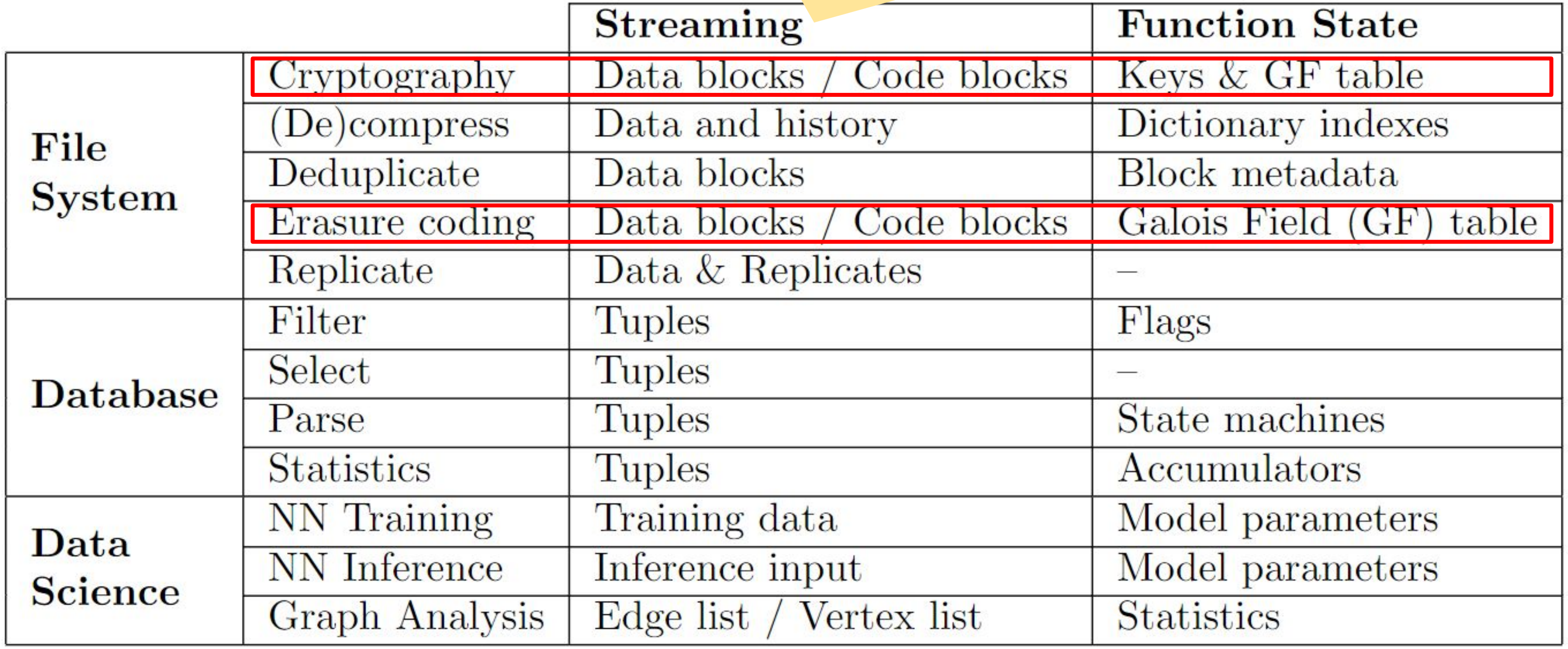


		Streaming	Function State
File System	Cryptography	Data blocks / Code blocks	Keys & GF table
	(De)compress	Data and history	Dictionary indexes
	Deduplicate	Data blocks	Block metadata
	Erasure coding	Data blocks / Code blocks	Galois Field (GF) table
	Replicate	Data & Replicates	–
Database	Filter	Tuples	Flags
	Select	Tuples	–
	Parse	Tuples	State machines
	Statistics	Tuples	Accumulators
Data Science	NN Training	Training data	Model parameters
	NN Inference	Inference input	Model parameters
	Graph Analysis	Edge list / Vertex list	Statistics

Offloads are often Streaming Operators

via systematic survey of common storage applications and offloads in industry and research.
[see paper]

Little Temporal Reuse

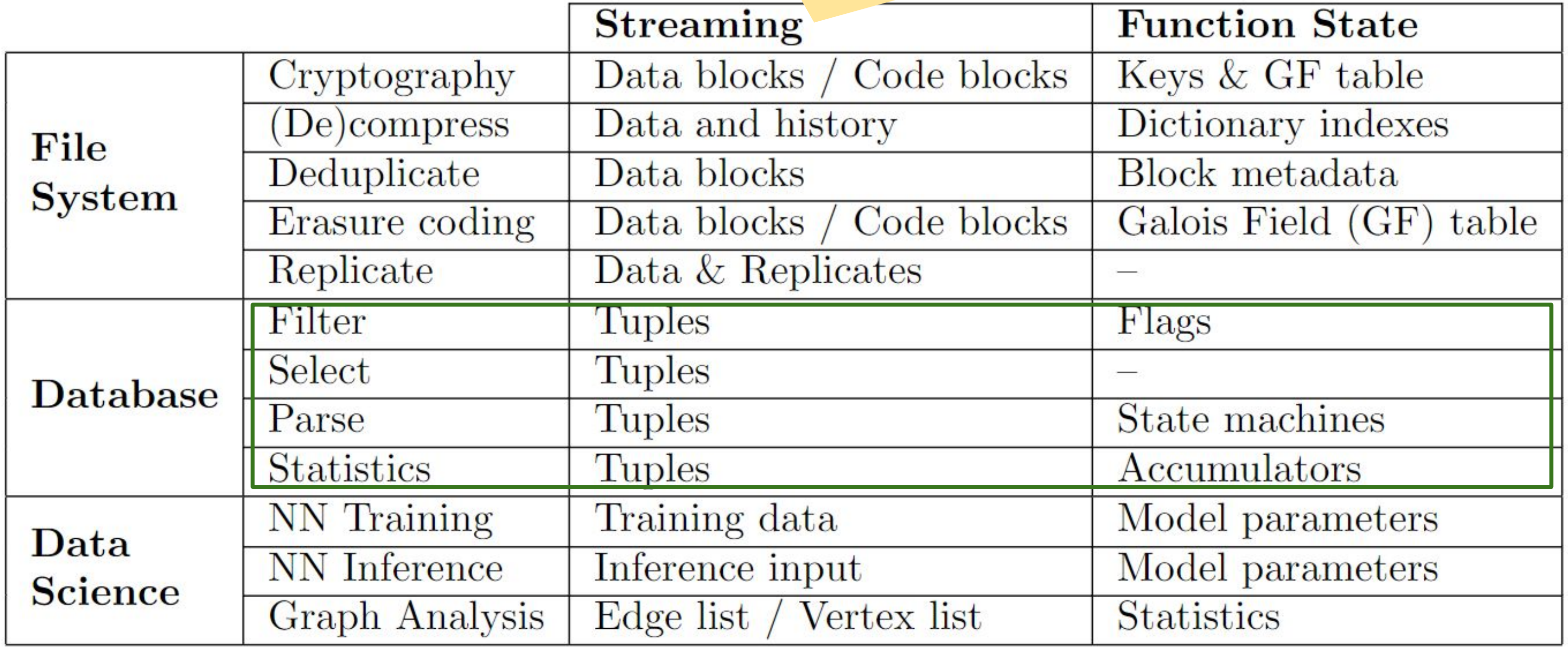


		Streaming	**Function State**
File System	Cryptography	Data blocks / Code blocks	Keys & GF table
	(De)compress	Data and history	Dictionary indexes
	Deduplicate	Data blocks	Block metadata
	Erasure coding	Data blocks / Code blocks	Galois Field (GF) table
	Replicate	Data & Replicates	–
Database	Filter	Tuples	Flags
	Select	Tuples	–
	Parse	Tuples	State machines
	Statistics	Tuples	Accumulators
Data Science	NN Training	Training data	Model parameters
	NN Inference	Inference input	Model parameters
	Graph Analysis	Edge list / Vertex list	Statistics

Offloads are often Streaming Operators

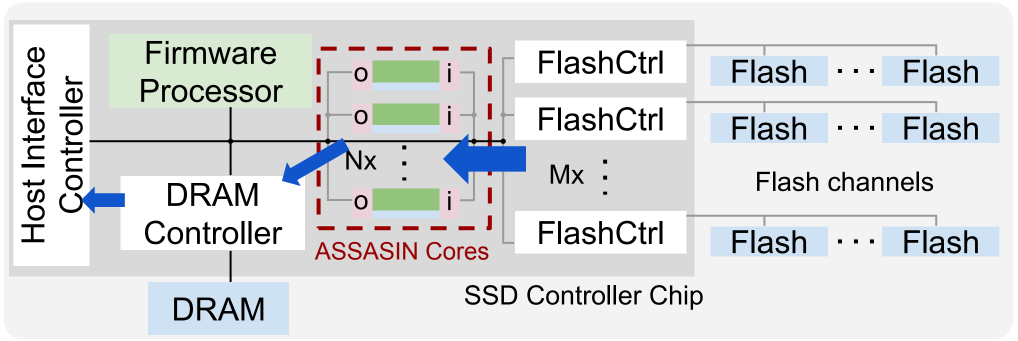
via systematic survey of common storage applications and offloads in industry and research. [see paper]

Little Temporal Reuse



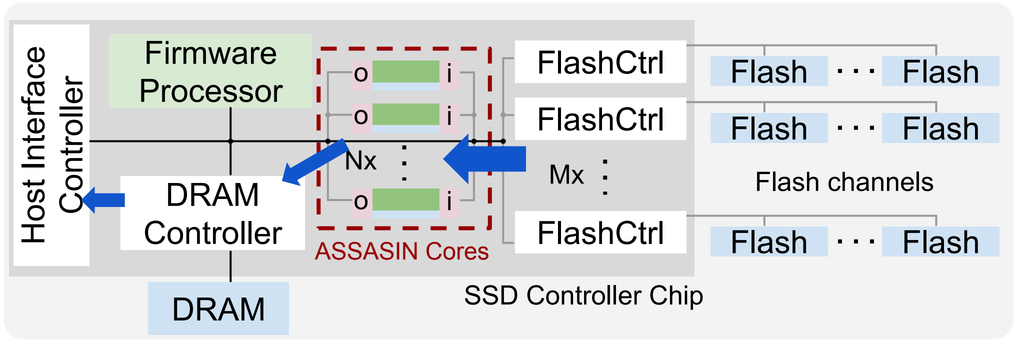
		Streaming	**Function State**
File System	Cryptography	Data blocks / Code blocks	Keys & GF table
	(De)compress	Data and history	Dictionary indexes
	Deduplicate	Data blocks	Block metadata
	Erasure coding	Data blocks / Code blocks	Galois Field (GF) table
	Replicate	Data & Replicates	–
Database	Filter	Tuples	Flags
	Select	Tuples	–
	Parse	Tuples	State machines
	Statistics	Tuples	Accumulators
Data Science	NN Training	Training data	Model parameters
	NN Inference	Inference input	Model parameters
	Graph Analysis	Edge list / Vertex list	Statistics

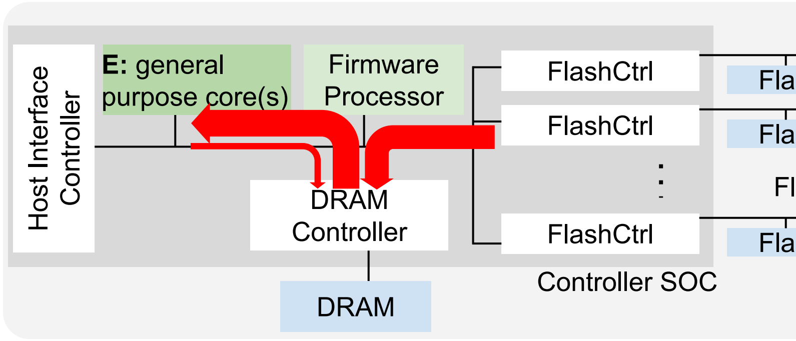
Architecture **S**upport for **S**tream computing to **A**ccelerate computat**I**o**N**al **S**torage



ASSASIN computes offloads on flash data directly. Data is not first staged in DRAM. Eliminates DRAM Bottleneck.

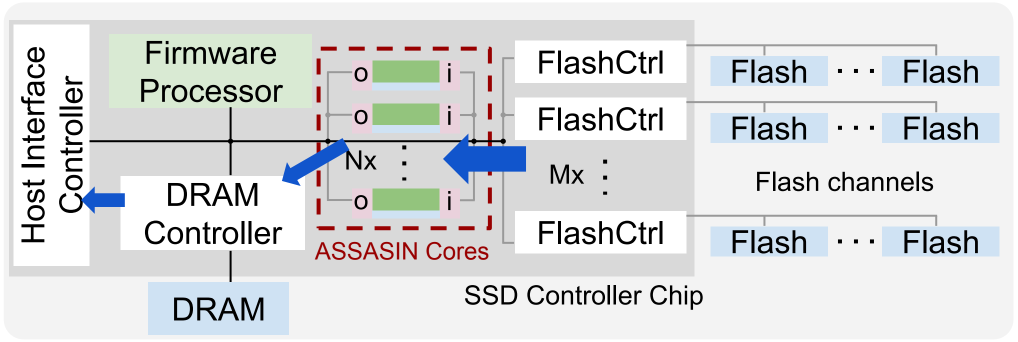
Architecture **S**upport for **S**tream computing to **A**ccelerate computat**I**o**N**al **S**torage





ASSASIN computes offloads on flash data directly. Data is not first staged in DRAM. Eliminates DRAM Bottleneck.

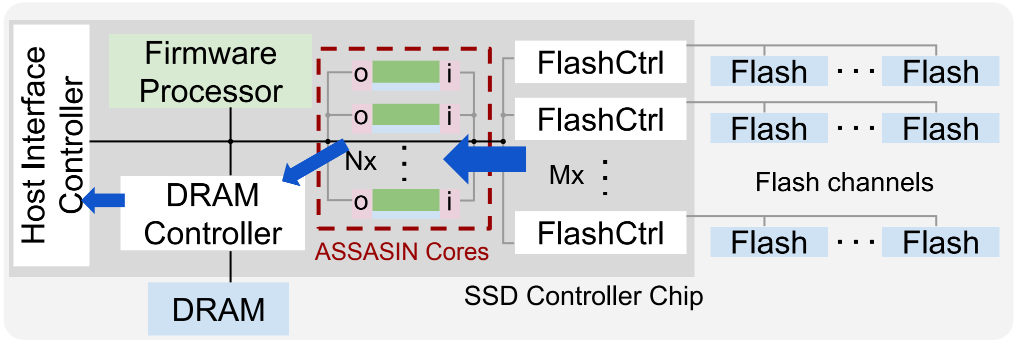
Architecture **S**upport for **S**tream computing to **A**ccelerate computat**I**o**N**al **S**torage

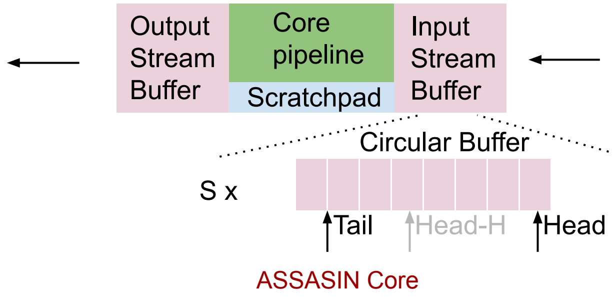


ASSASIN computes offloads on flash data directly. Data is not first staged in DRAM. Eliminates DRAM Bottleneck

Flexible interconnect enables teaming ASSASIN cores for higher bandwidth and core utilization.

Architecture **S**upport for **S**tream computing to **A**ccelerate computat**I**o**N**al **S**torage





ASSASIN computes offloads on flash data directly. Data is not first staged in DRAM. Eliminates DRAM Bottleneck

Flexible interconnect enables teaming ASSASIN cores for higher bandwidth and core utilization.

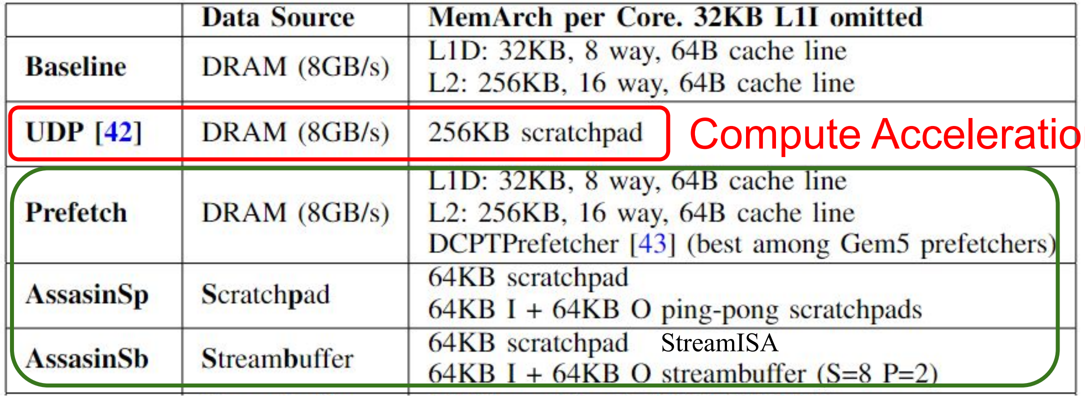
ASSASIN core pulls/pushes data through streambuffer, computes on it directly

StreamISA (see paper) provides efficient stream pointers management .

Offload computation gets higher-performance data access.

Evaluation Methodology

Computational SSD Configurations



	Data Source	MemArch per Core. 32KB L1I omitted
Baseline	DRAM (8GB/s)	L1D: 32KB, 8 way, 64B cache line L2: 256KB, 16 way, 64B cache line
UDP [42]	DRAM (8GB/s)	256KB scratchpad
Prefetch	DRAM (8GB/s)	L1D: 32KB, 8 way, 64B cache line L2: 256KB, 16 way, 64B cache line DCPTPrefetcher [43] (best among Gem5 prefetchers)
AssasinSp	Scratchpad	64KB scratchpad 64KB I + 64KB O ping-pong scratchpads
AssasinSb	Streambuffer	64KB scratchpad StreamISA 64KB I + 64KB O streambuffer (S=8 P=2)

Compute Acceleration

Memory Architecture Innovation

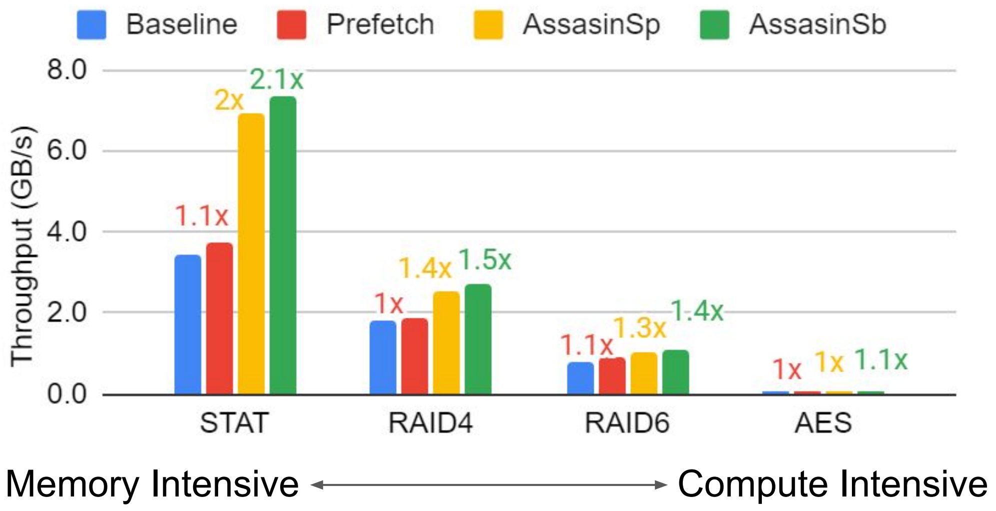
Hybrid Simulation

- Gem5 RISCV model
  - Compute, DRAM, Stream Buffers
  - 8 RISCV cores
  - LPDDR5 8GB/s DRAM
- MQSim no interface Flash Model
  - Storage access
  - 8 channels of flashes
  - 1GB/s per channel

Workloads

- File system functions
- Database functions

FileSystem Offloads: ASSASIN up to 2x Speedup

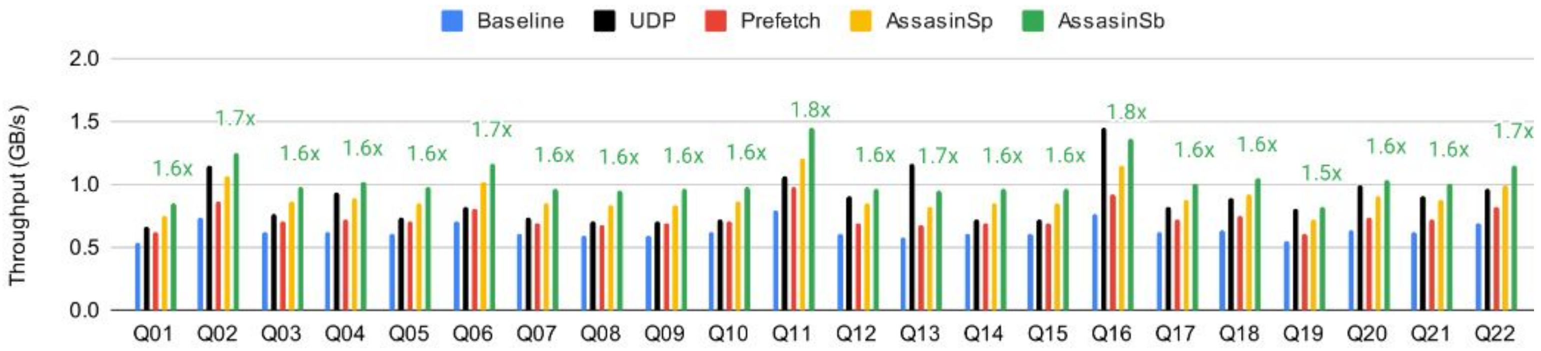


Prefetch aggravates the DRAM bottleneck. Minimal speedup.

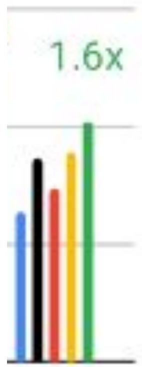
ASSASIN benefits offload more when offloads are memory -intensive. Up to **2x speedup** for **eliminating the memory wall**.

ASSASIN core features: StreamISA and streambuffer brings **10% further speedup** for AssasinSb over AssasinSp

Database offloads - combined operators (**P**arse, **S**elect, **F**ilter)



22 TPC-H queries Scale factor = 10



GeoMean

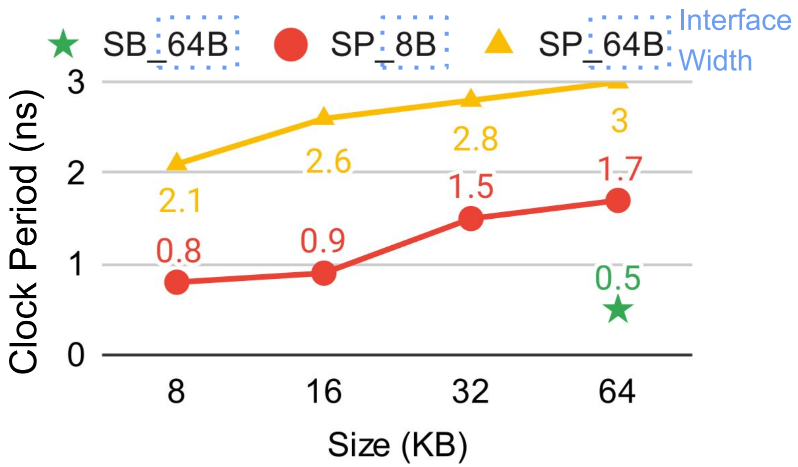
PSF Database offload is moderately compute-intensive

Compute acceleration (UDP) achieves 1.3x speedup over baseline (Geomean)

Memory-hierarchy innovation (ASSASIN) match and exceeds compute acceleration

- AssasinSp matches performance by avoiding DRAM bottleneck
- AssasinSb improves 18% further with streambuffer and StreamISA.

Achievable Clock Period

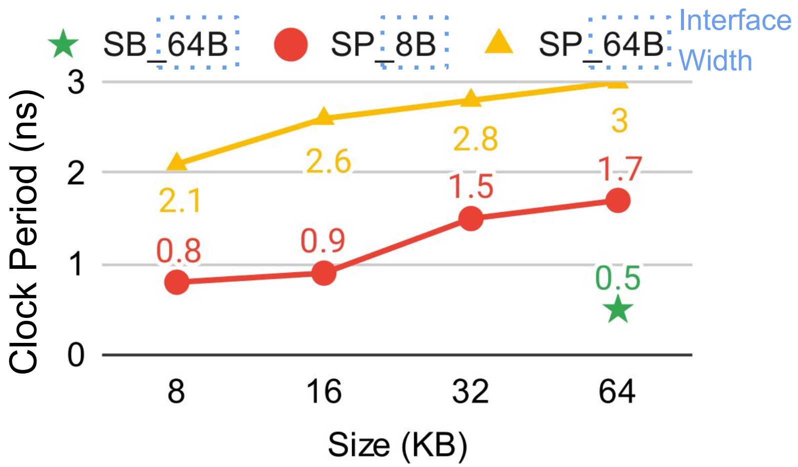


SB=**S**tream**B**uffer SP=**S**cratch**P**ad
Streambuffer precisely prefetches the head, so AssasinSb enjoys 11% shorter clock period.

Scratchpad limited by large MUXes.
AssasinSp suffers 30% performance degradation.

SAED14nm technology

Achievable Clock Period

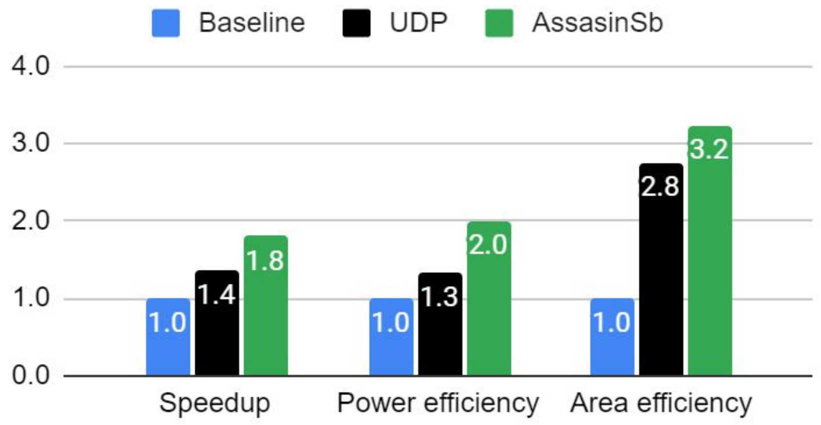


SB=**S**tream**B**uffer SP=**S**cratch**P**ad
Streambuffer precisely prefetches the head, so AssasinSb enjoys 11% shorter clock period.

Scratchpad limited by large MUXes.
AssasinSp suffers 30% performance degradation.

SAED14nm technology

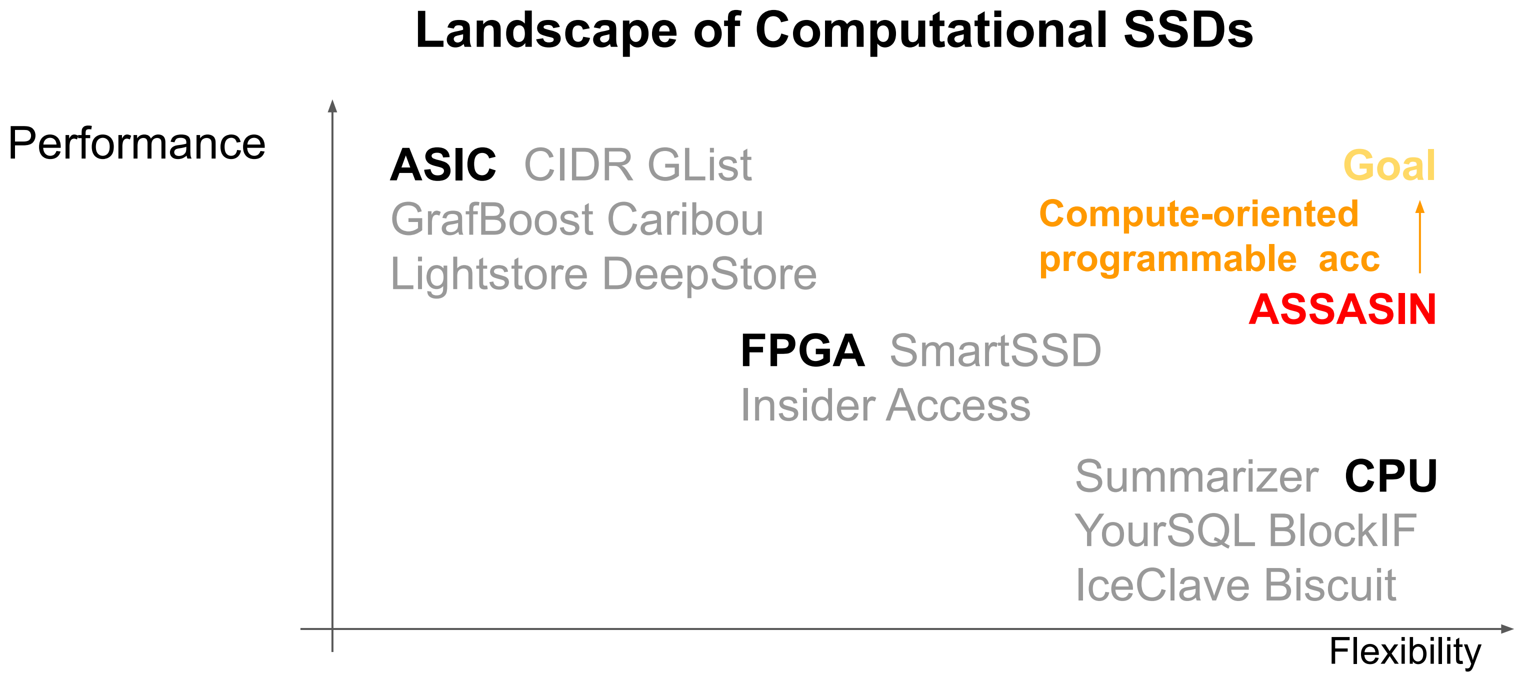
Power and Area efficiency



Shorter clock period further improves ASSASIN

1.8x speedup, 3.0x power efficiency, 3.2x area efficiency

Related Work & Future Work



Summary

Literature survey: Most computational storage offloads are dominated by streaming access

The ASSASIN optimizes for streaming access

- ASSASIN SSD architecture
    - integrates compute directly on flash data streams
    - eliminates the SSD DRAM bottleneck
- ASSASIN core architecture:
    - Streambuffers, StreamISA(integrated stream management)
    - Low-latency, energy, high bandwidth access

Benefits to offloads: **1.5-2.4x speedup. 3x power efficiency. 3.2x area efficiency**

See the paper for:

- ASSASIN programming model
- How ASSASIN enables independent FTL and thus generality
- ASSASIN's performance scalability
- ASSASIN robust performance with data layout skew