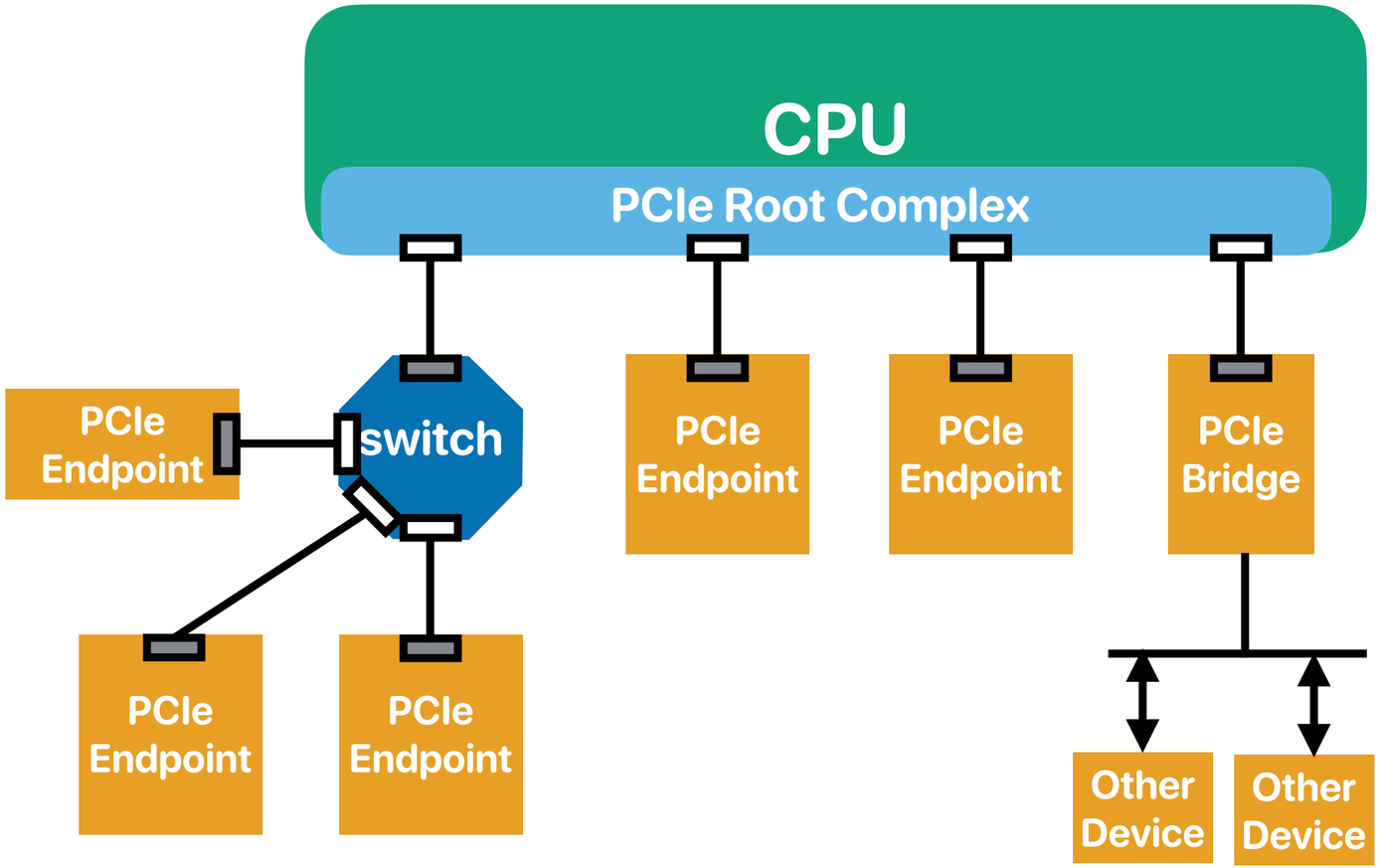
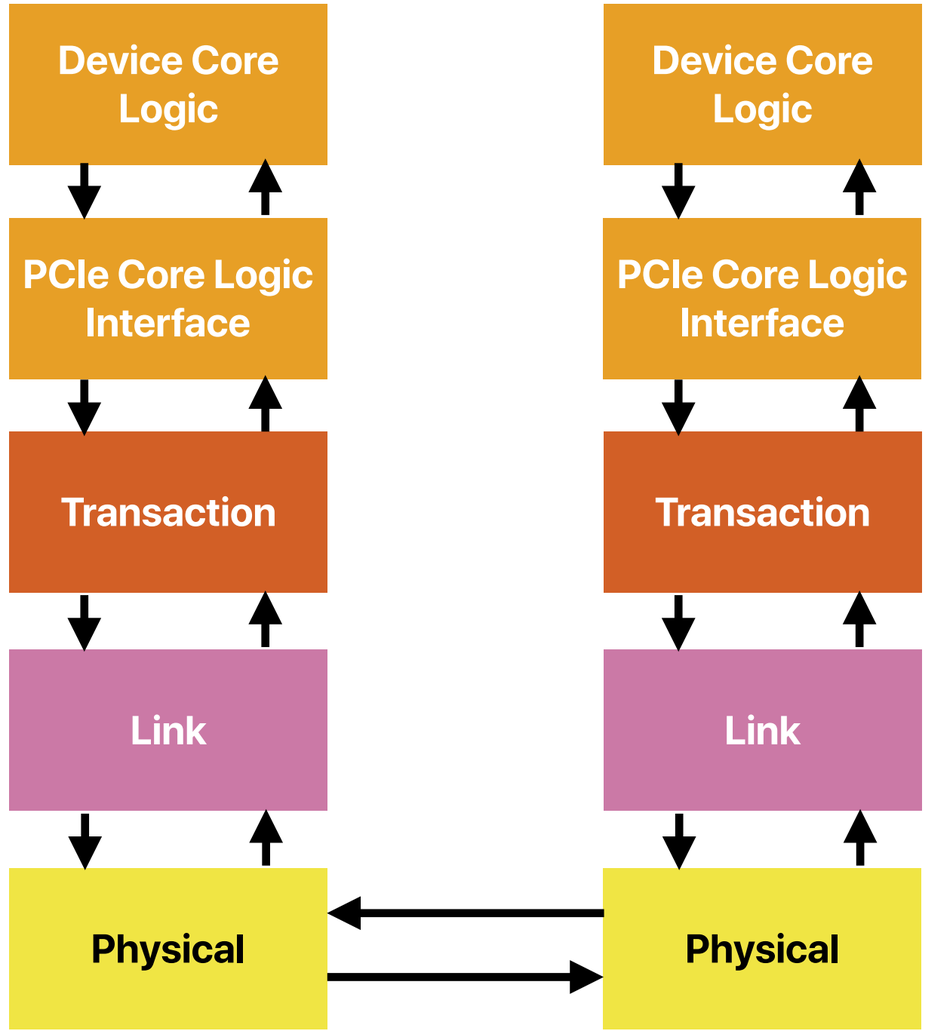
In-Storage Processing

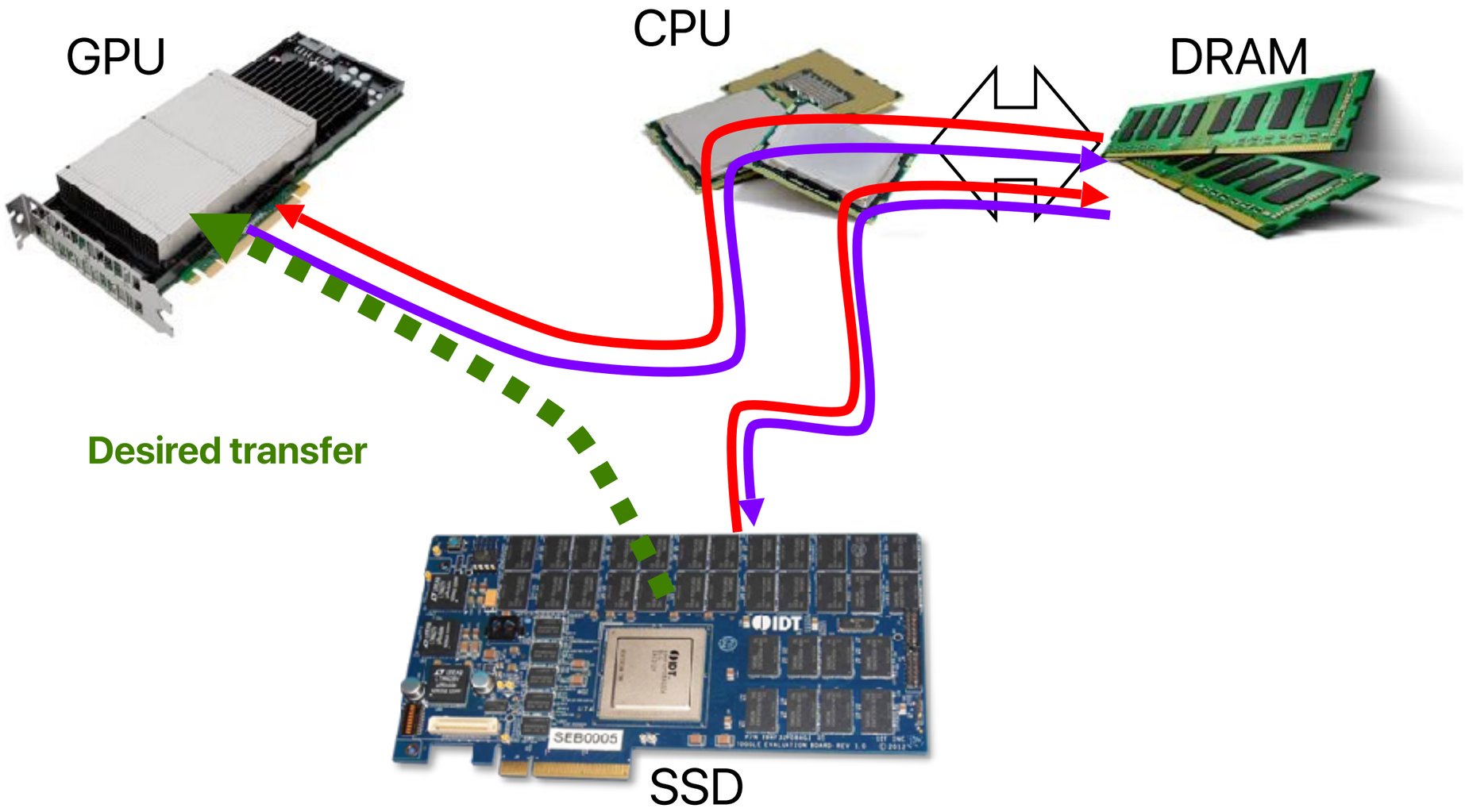
Hung-Wei Tseng

Recap: PCIe "Interconnect"

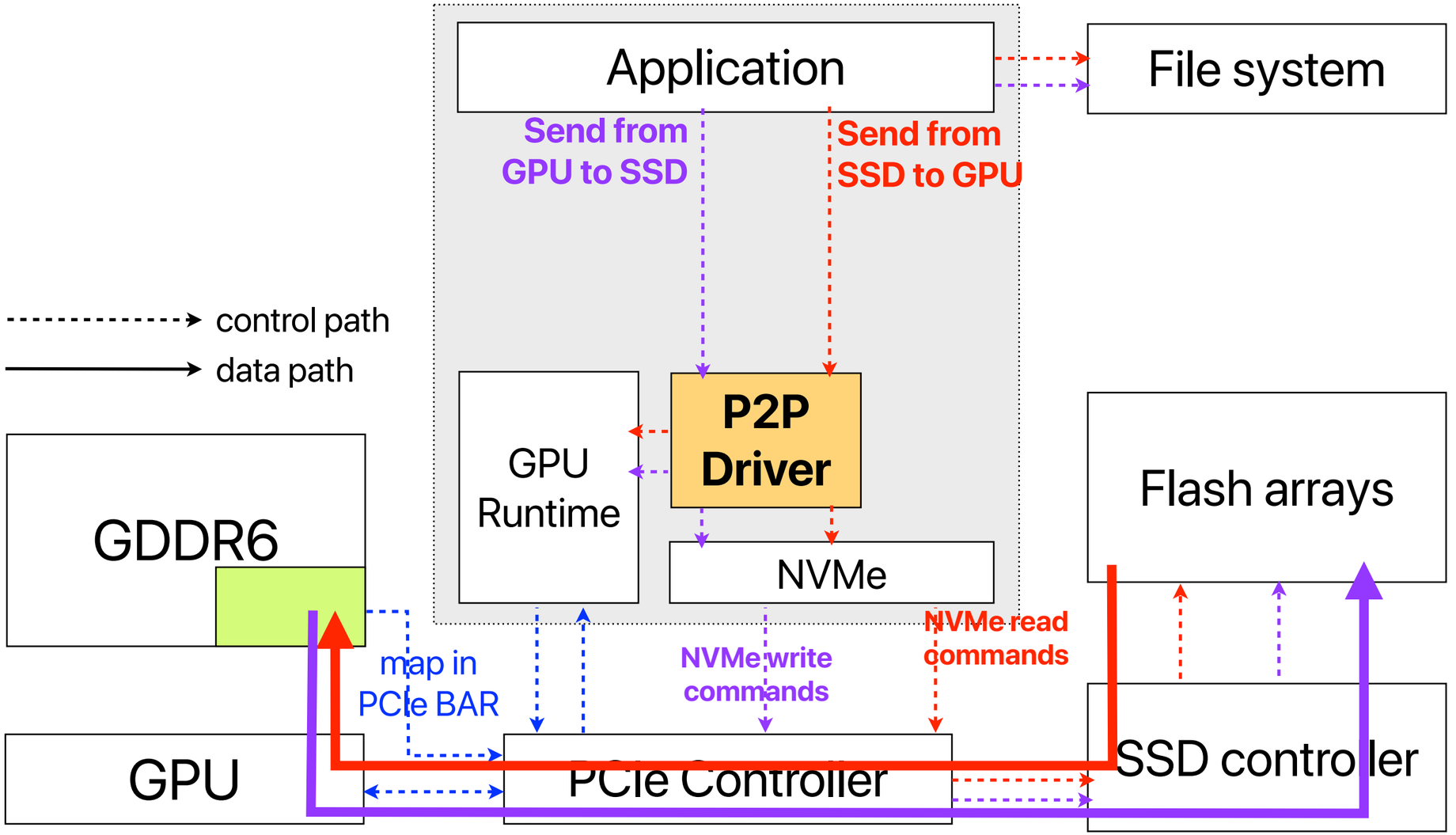




GPU-accelerated architecture



Recap: P2P between GPU and SSD



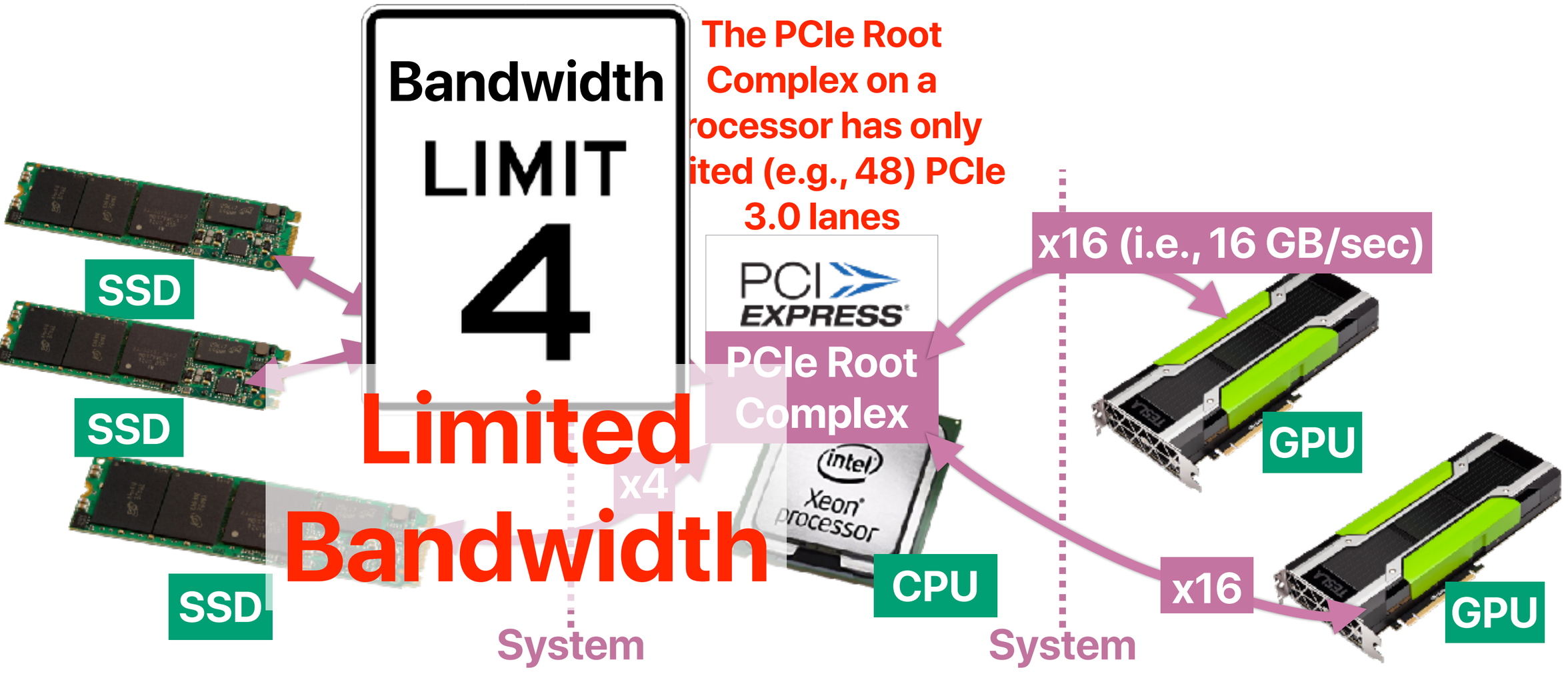
Recap: why doesn't GPUDirect fly?

- Limited performance gain (30% at most) with the demand of code modifications
- Performance may get worse in multiprogrammed workloads
  - No write buffering — longer time to switch out a GPU kernel
  - No page caching — longer time to switch in a GPU kernel or reading files

Outline

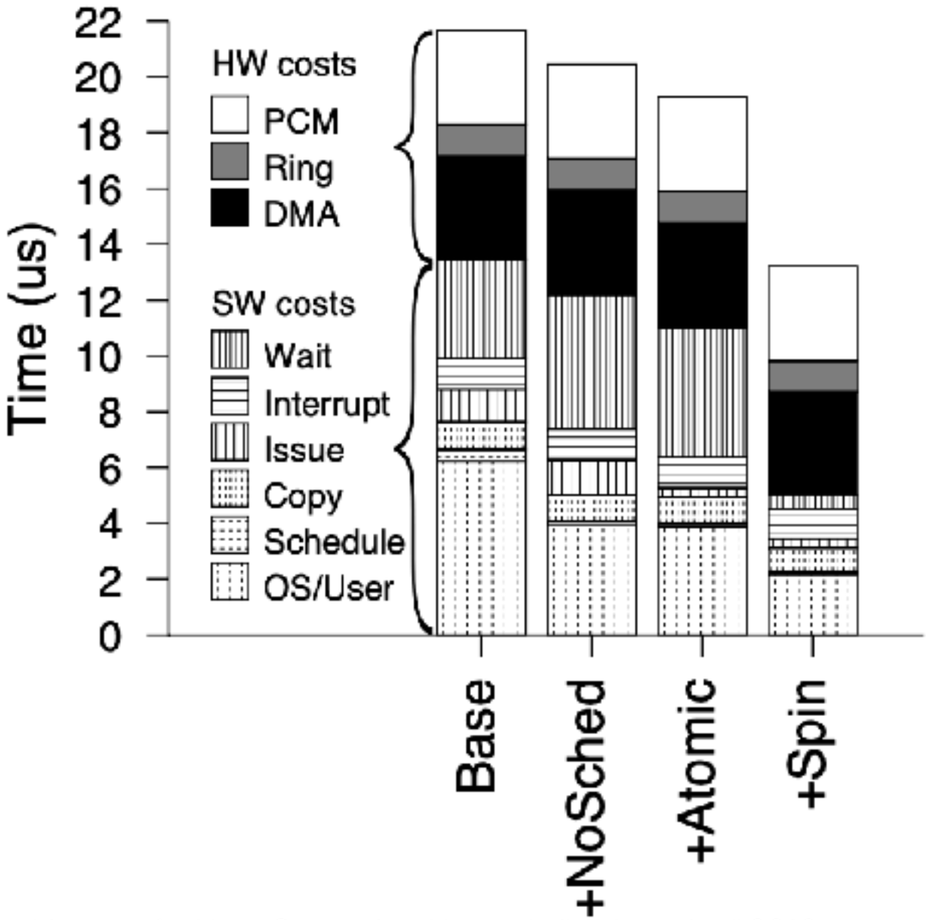
- The formula for Near/In-Storage Processing
- Cases of ISP

Limited Interconnect Bandwidth



Software overhead

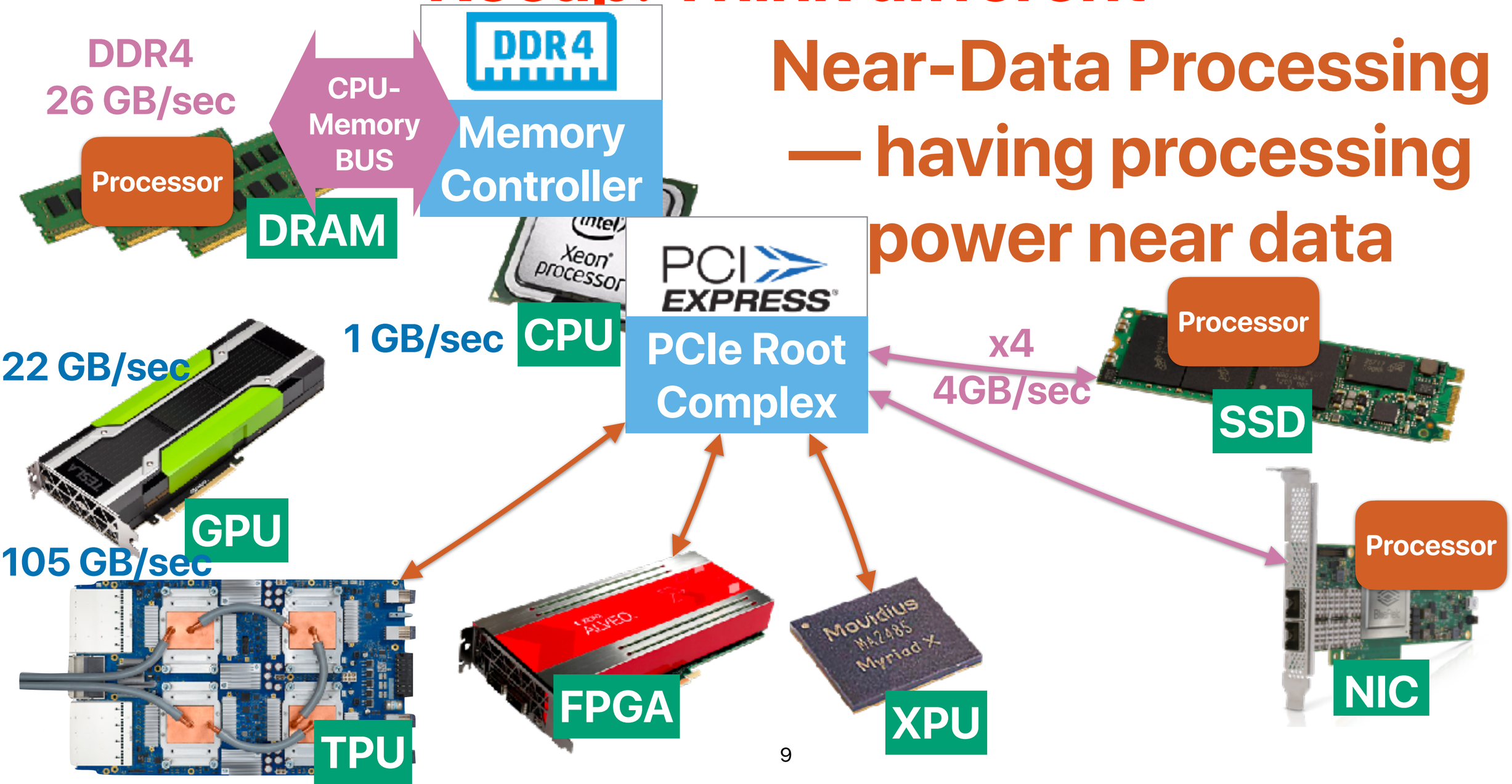
Label	Description	Baseline latency (μs) Write	Read
OS/User	OS and userspace overhead	1.98	1.95
OS/User	Linux block queue and no-op scheduler	2.51	3.74
Schedule	Get request from queue and assign tag	0.44	0.51
Copy	Data copy into DMA buffer	0.24/KB	-
Issue	PIO command writes to Moneta	1.18	1.15
DMA	DMA from host to Moneta buffer	0.93/KB	-
Ring	Data from Moneta buffer to mem ctrl	0.28/KB	-
PCM	4 KB PCM memory access	4.39	5.18
Ring	Data from mem ctrl to Moneta buffer	-	0.43/KB
DMA	DMA from Moneta buffer to host	-	0.65/KB
Wait	Thread sleep during hw	11.8	12.3
Interrupt	Driver interrupt handler	1.10	1.08
Copy	Data copy from DMA buffer	-	0.27/KB
OS/User	OS return and userspace overhead	1.98	1.95
Hardware total for 4 KB (accounting for overlap)		8.2	8.0
Software total for 4 KB (accounting for overlap)		13.3	12.2
File system additional overhead		5.8	4.2



A. M. Caulfield, A. De, J. Coburn, T. I. Mollow, R. K. Gupta and S. Swanson, "Moneta: A High-Performance Storage Array Architecture for Next-Generation, Non-volatile Memories," 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, 2010, pp. 385-395, doi: 10.1109/MICRO.2010.33.

Recap: Think different

Near-Data Processing — having processing power near data



Why does ISP/SmartSSD make sense?

The "Winning Formula" of In-Storage Processing

Conventional Model

ISP Model

$$\frac{DataVolume_{raw}}{Bandwidth_{device}} + \frac{DataVolume_{raw} \times ComputationIntensity_{NDP}}{ComputationThroughput_{device}} + \frac{DataVolume_{afterNDP}}{Bandwidth_{device2host}} + \frac{DataVolume_{afterNDP} \times ComputationIntensity_{afterNDP}}{ComputationThroughput}$$

$$<= \frac{DataVolume_{raw}}{Bandwidth_{device2host}} + \frac{DataVolume_{raw} \times ComputationIntensity}{ComputationThroughput}$$

The ISP processor must have efficient internal access to the device

The ISP computation cannot take too long

The ISP should reduce data volume

The ISP should offload the processor

Opportunities of SmartSSDs

Multi-channel to optimize bandwidth

Flashtec™ NVMe2032 and NVMe2016 Controllers

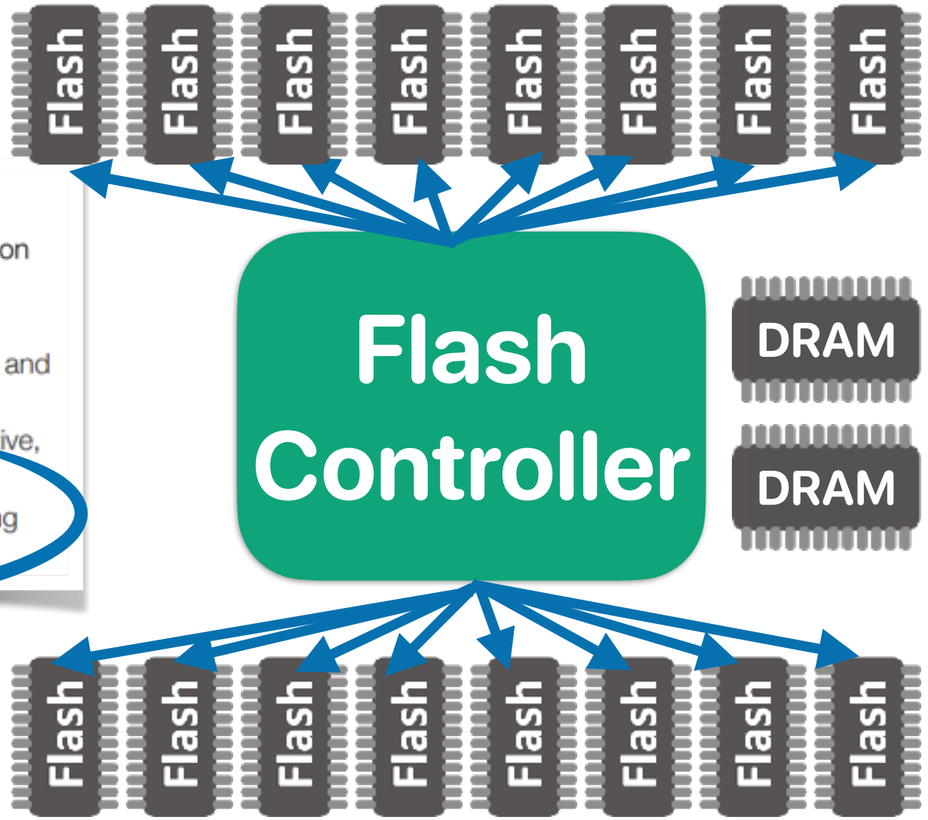
32- and 16-Channel PCIe Flash Controller Pr

Summary

The Flashtec™ 2nd generation NVMe Controller Family enables the
and data centers to realize the highest performance SSDs utilizing n
technologies. Combining world-class capacity and flexibility, the Flas
reliable choice. The Flashtec NVMe2032 and NVMe2016 controllers
Express (NVMe) host interface and are optimized for high-performan
operations, performing all flash management operations on-chip an
processing and memory resources.

Features

- Flashtec NVMe2032 controller can achieve up to 1 million random read IOPS on 4 KB operations
- Up to 20 TB Flash capacity using 256 GB Flash
- SLC, MLC, Enterprise MLC, and TLC Flash with toggle and ONFI interface
- PCIe Gen 3 x8 or dual independent PCIe Gen 3 x4 (active, active/standby) host interface
- 16 and 32 independent Flash channels, each supporting up to 8 CE



Each ~500MB/sec
8 channels == 4GB/sec
16 channels == 16 GB/sec

Micron

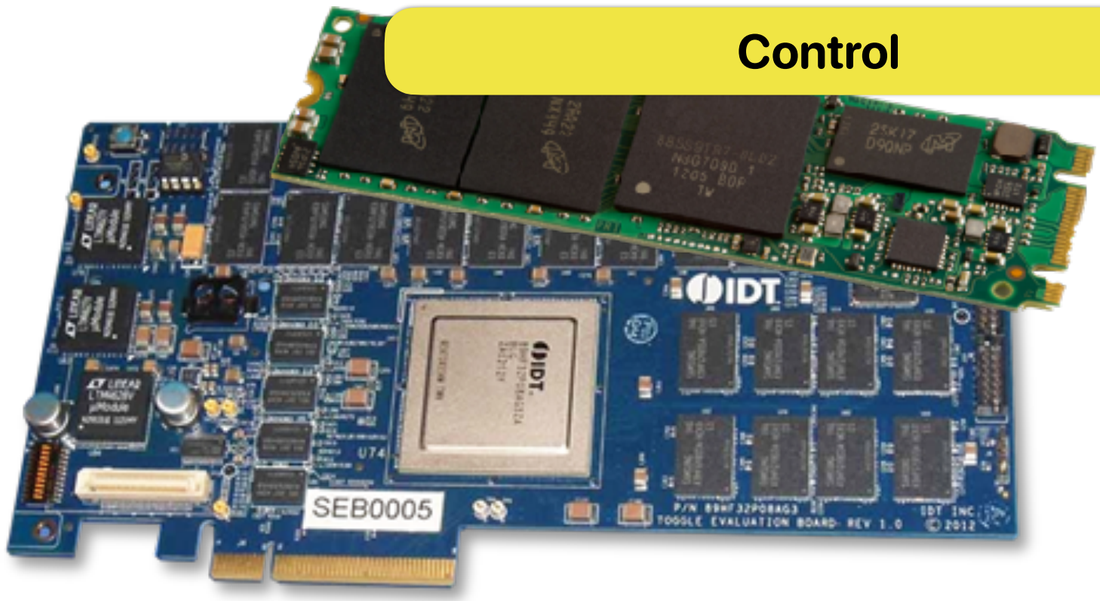
- Organization
  - Page size x8: 18,592 bytes (16,384 + 2208 bytes)
  - Block size: 2304 pages, (36,864K + 4968K bytes)
  - Plane size: 4 planes x 504 blocks
  - Device size: 512Gb: 2016 blocks; 1Tb: 4032 blocks; 2Tb: 8064 blocks; 4Tb: 16,128 blocks; 8Tb: 32,256 blocks

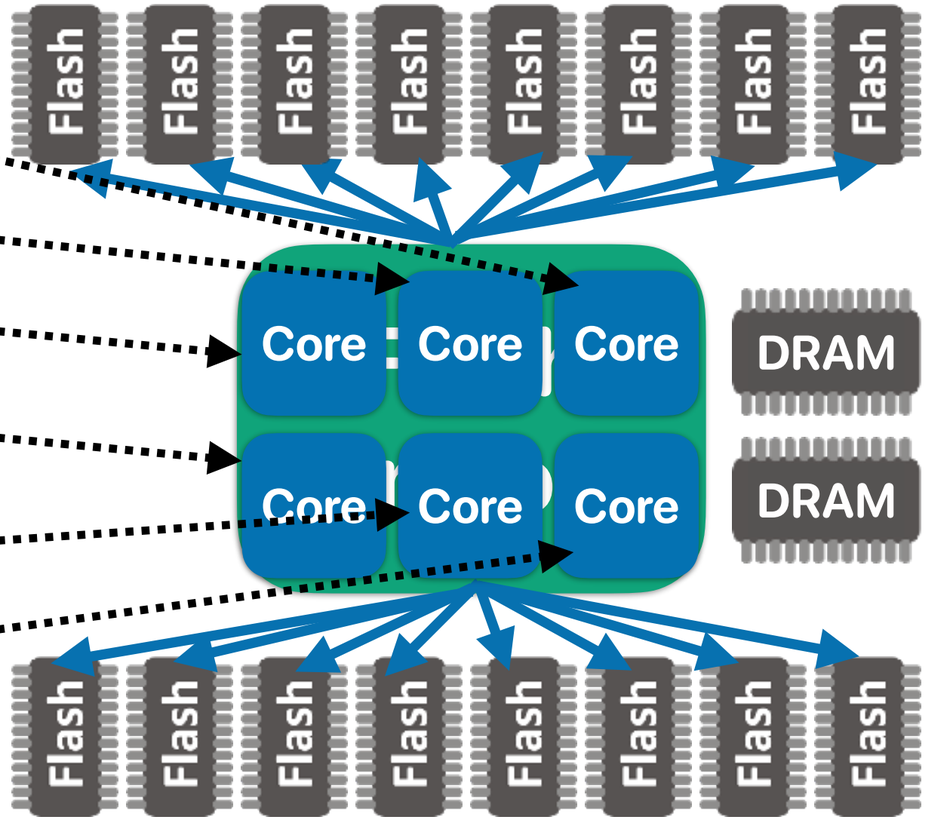
TLC 512Gb-8Tb NAND – Array Characteristics

is suspended or ongoing					
READ PAGE operation time without/with V_{PP}	t_R	57/41	60	µs	2, 12
SNAP READ operation time	t_{RSNAP}	27	37	µs	
Cache read busy time	t_{RCBSY}	11	60	µs	2, 8, 12

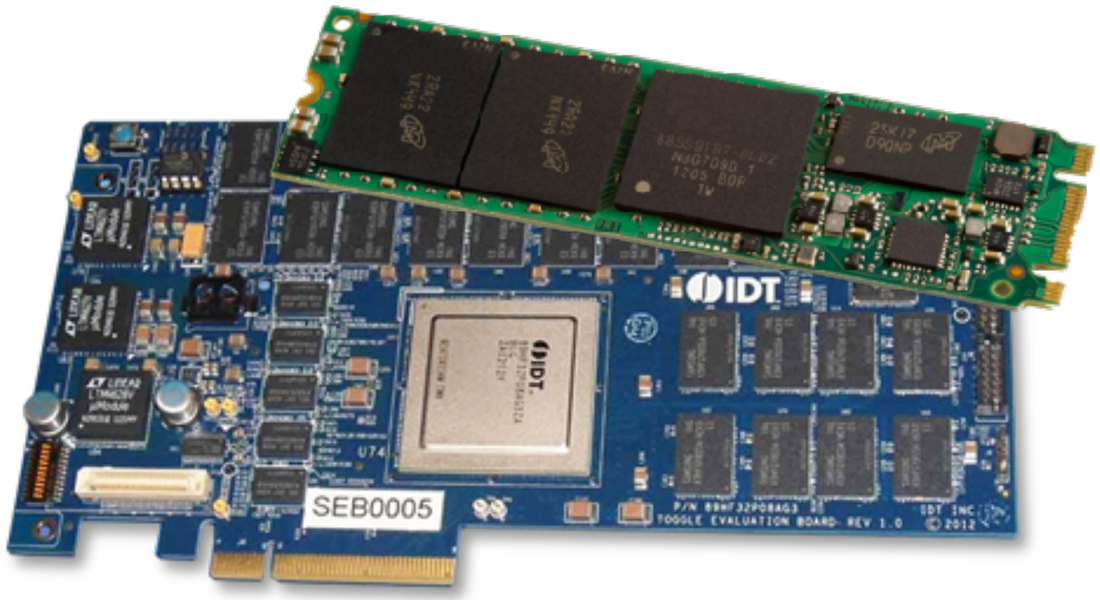
SSDs already have processors inside

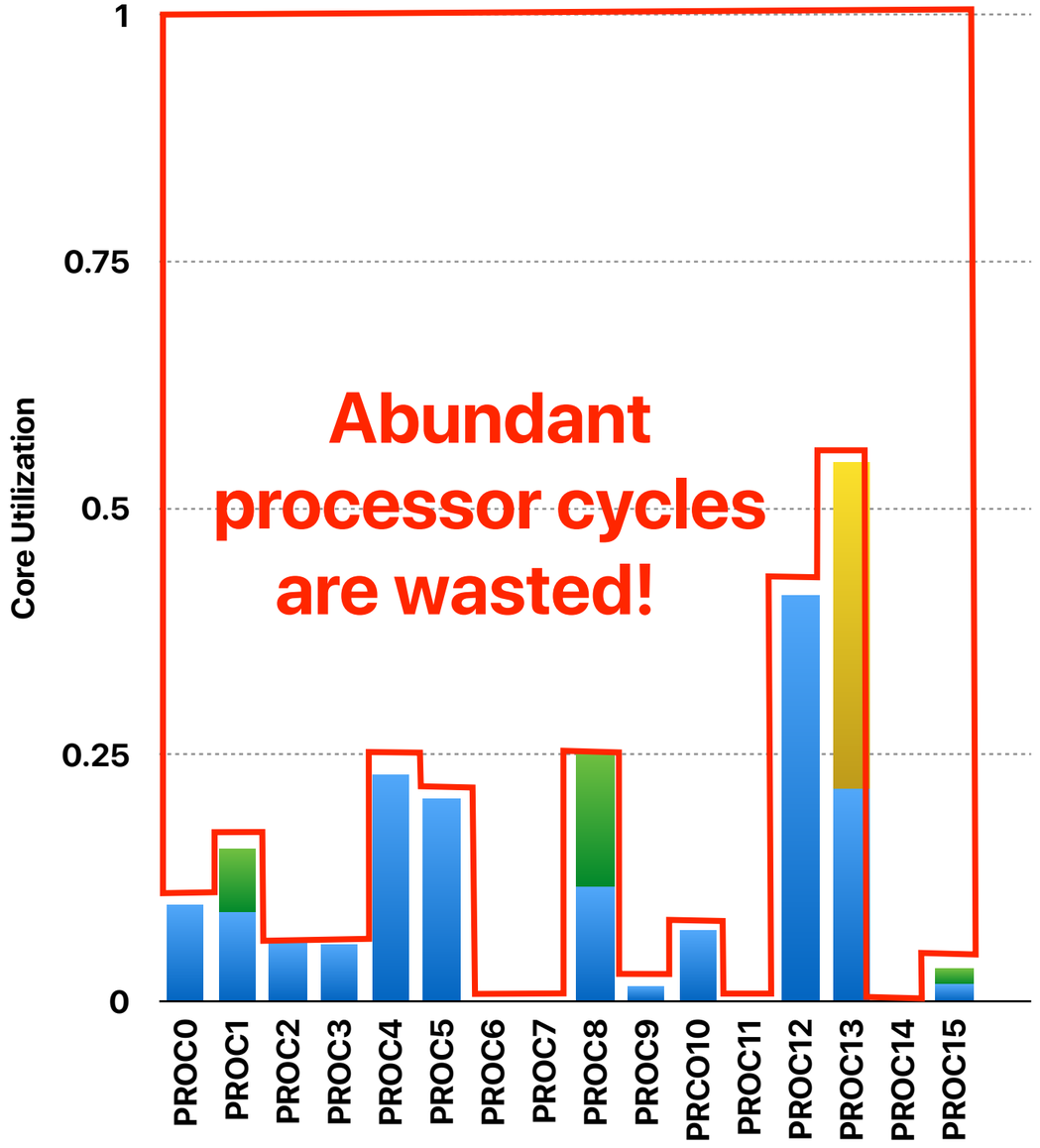
- Parse NVMe/SATA/PCIe commands
- Address mapping/lookup
- Wear-leveling/Garbage Collection
- Read
- Program
- Control





Are we using them “efficiently”?



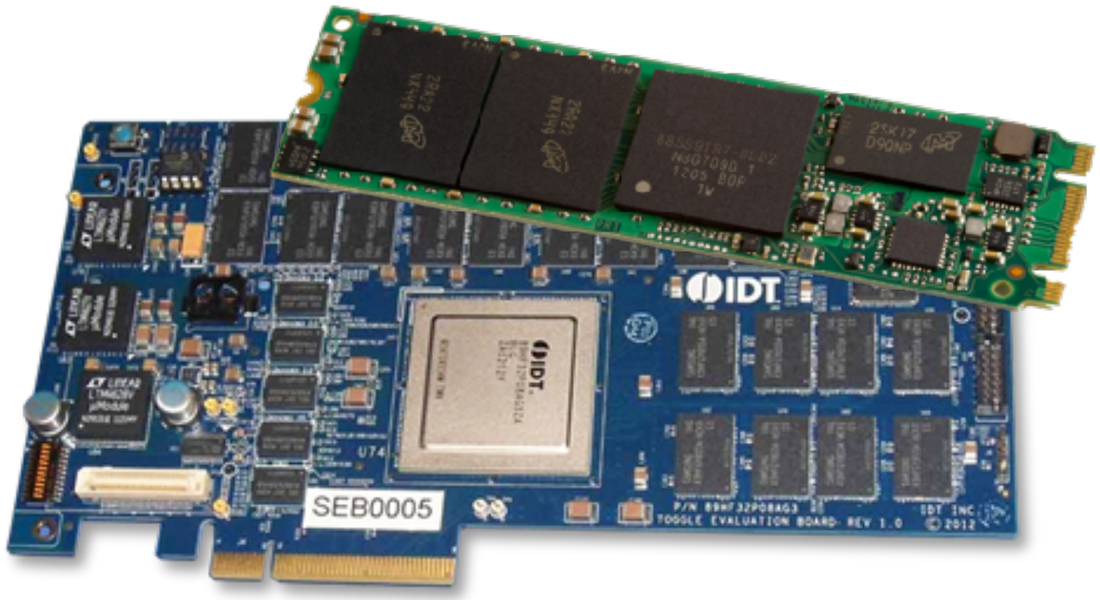


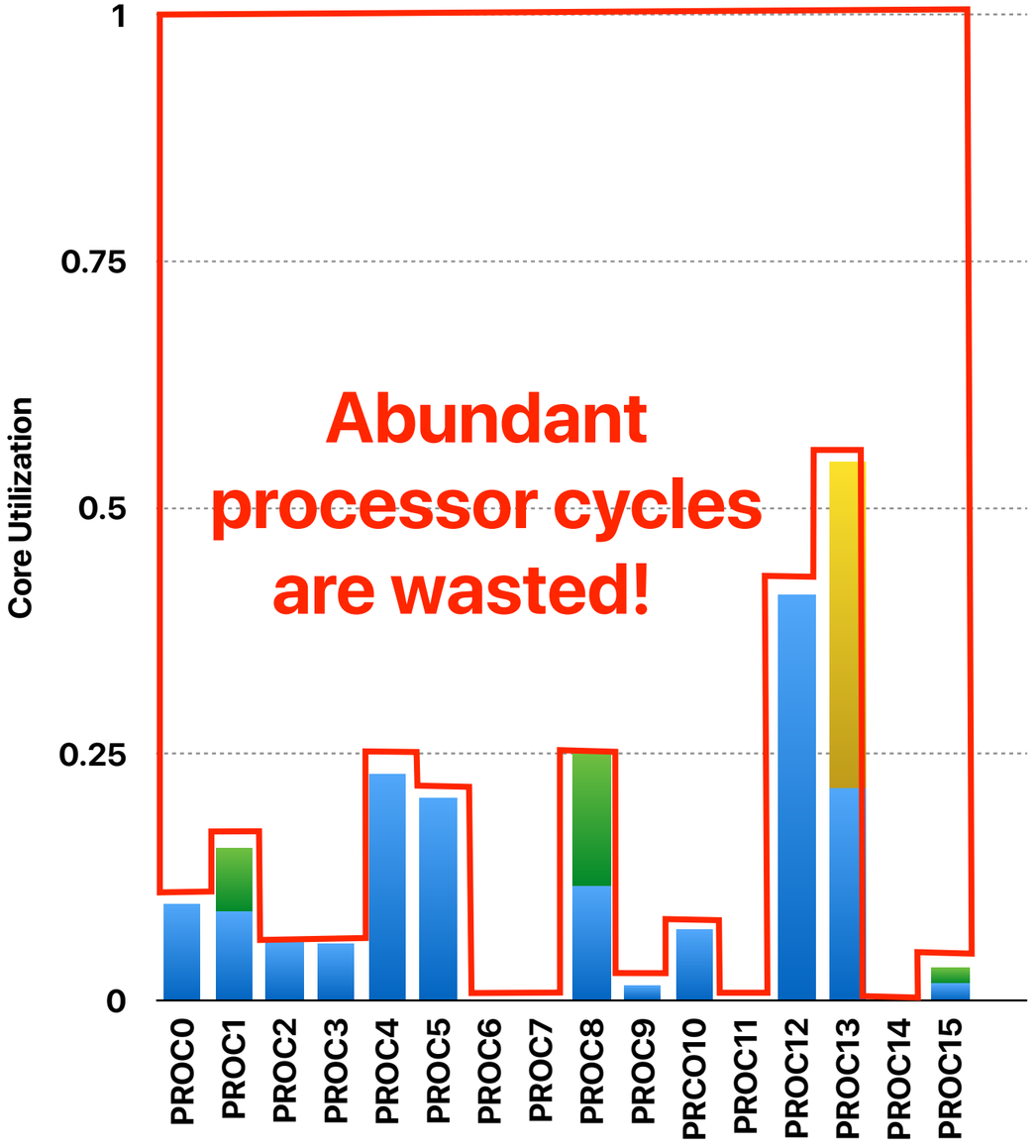
3:00

Why are we under utilizing these processors?

SSDs already have processors inside

- Firmware does not rely on OS
- Utilization is not optimized as well
- We can “tweak” and “extend” the firmware to perform computation!

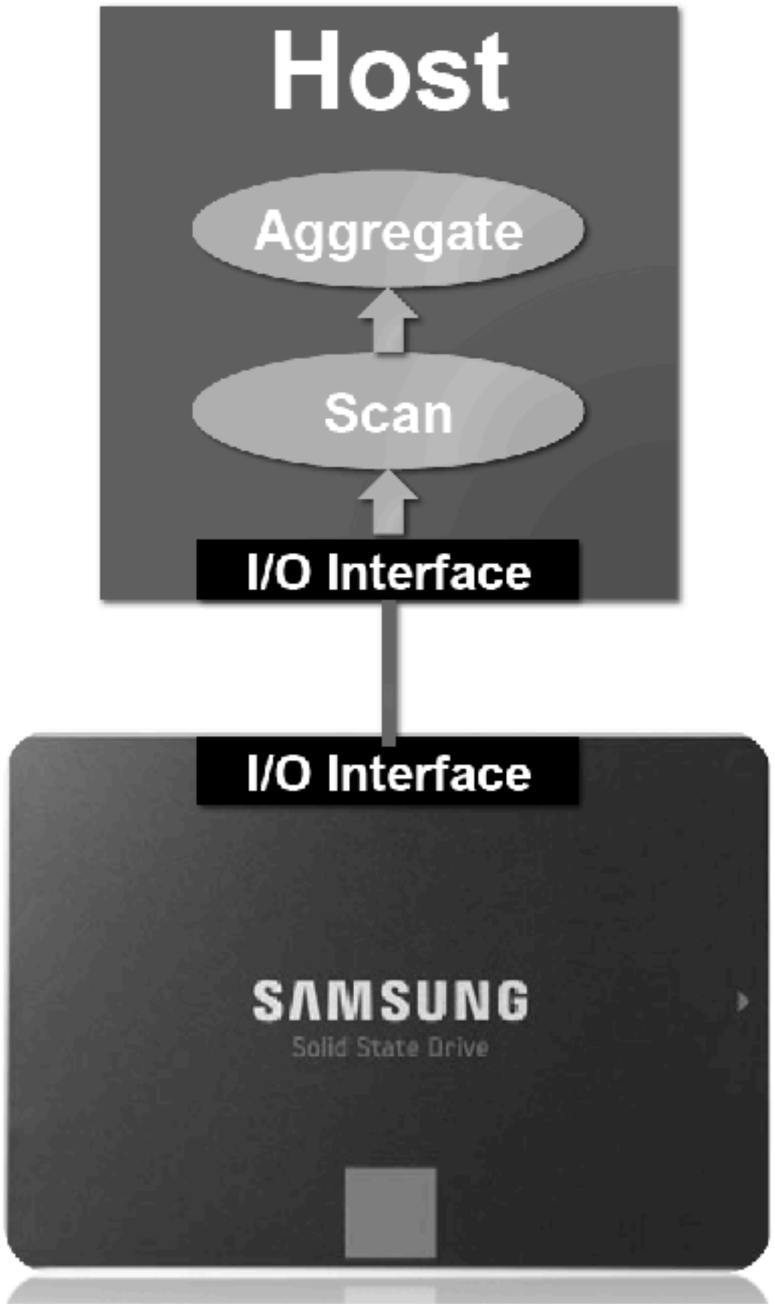




Query processing on smart SSDs: opportunities and challenges.

Jaeyoung Do, Yang-Suk Kee, Jignesh M. Patel, Chanik Park, Kwanghyun Park, and David J. DeWitt

In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)



Query Processing on Smart SSDs , SIGMOD 2013

Change in the System

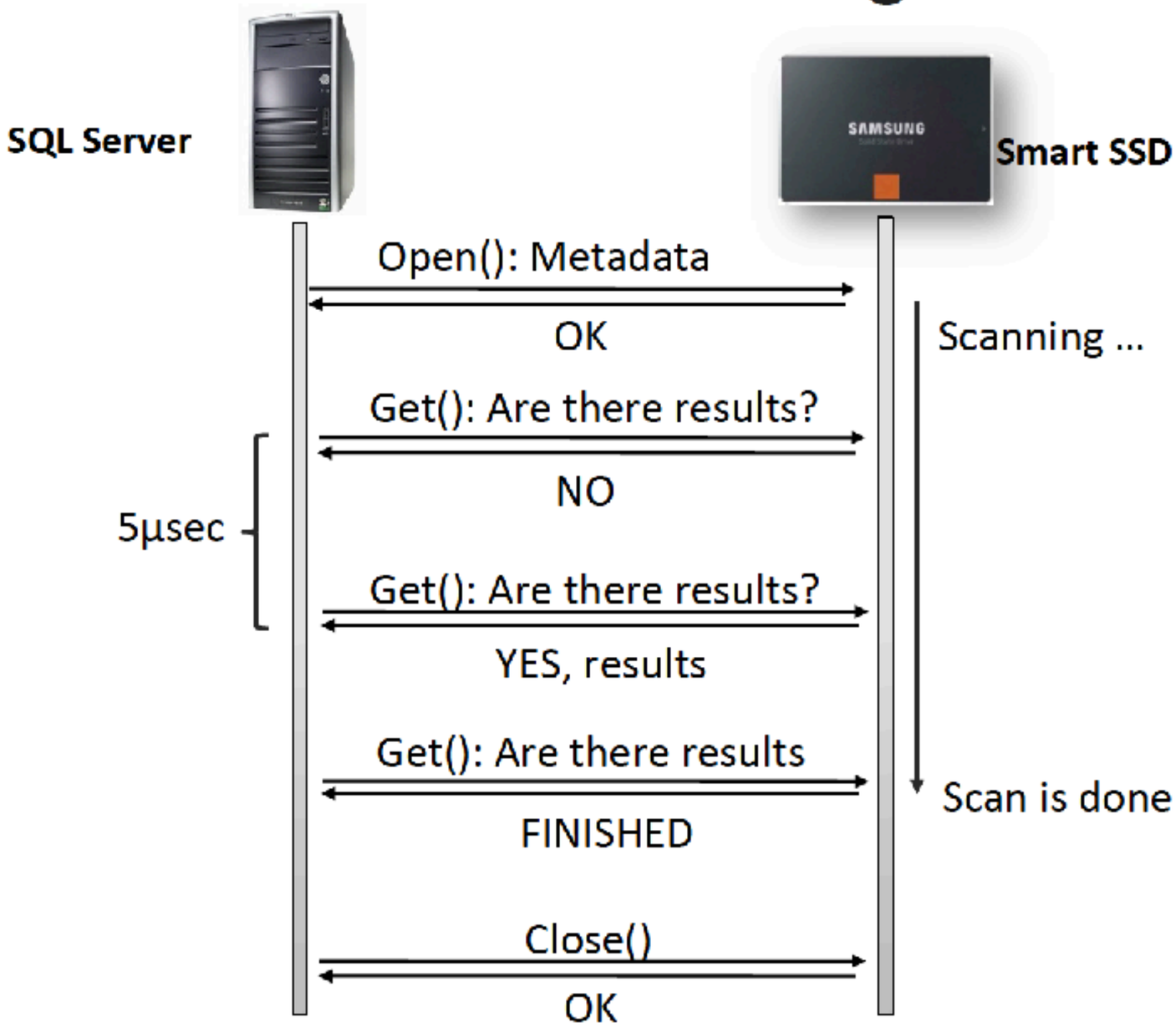
Samsung Smart SSD

- **New Smart SSD APIs** to run database operations inside the SSD
- Open(), Get(), Close(), SendToHost()

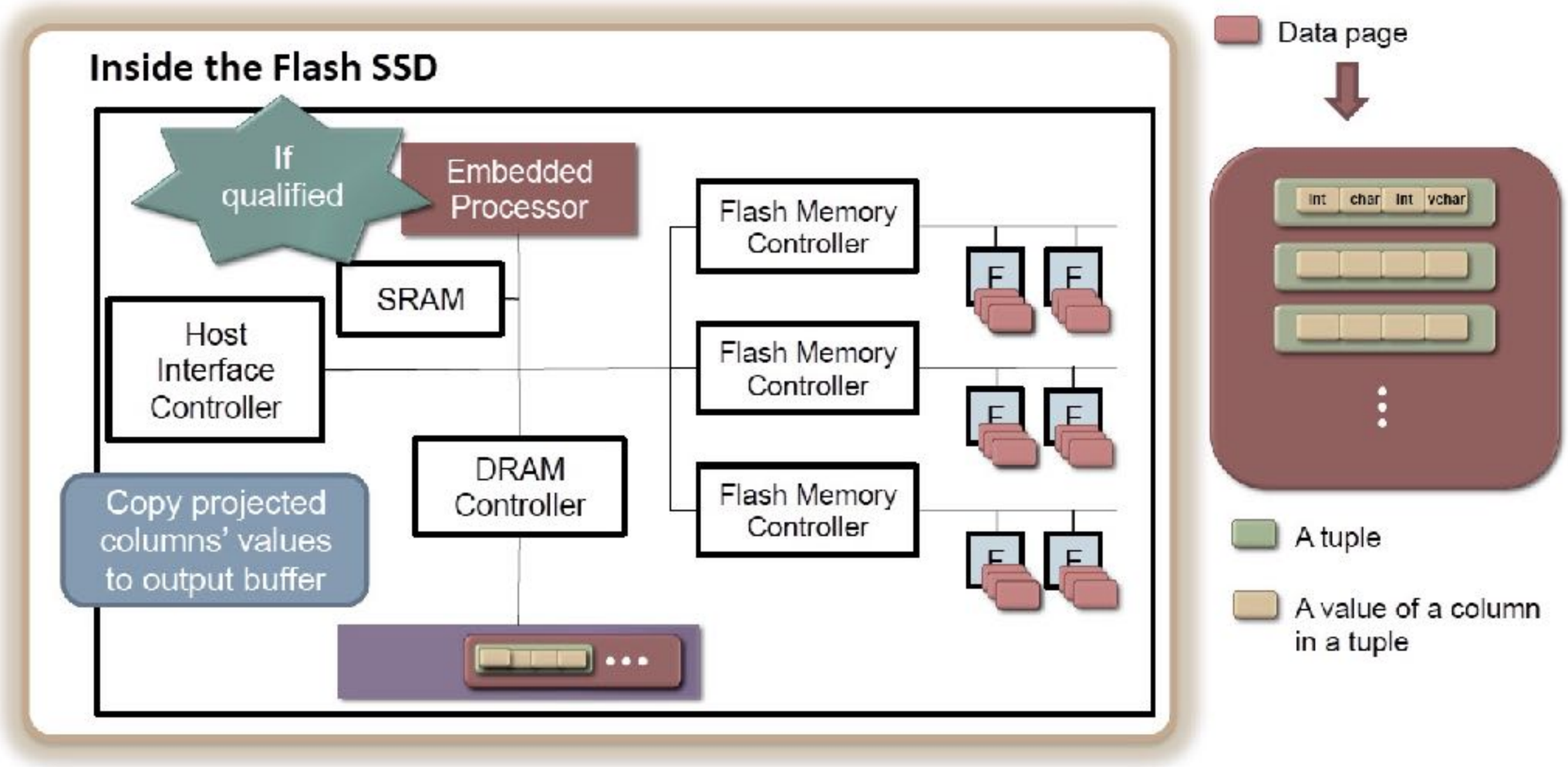
SQL Server

- Modified the query processor to hard-code selected Smart SSD plans
- Implement Smart SSD executors

Communication Flow – e.g. Scan



Data Format in SSD

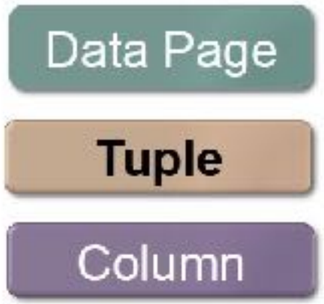


Data is read in tuples from the table

Query Processing on Smart SSDs , SIGMOD 2013

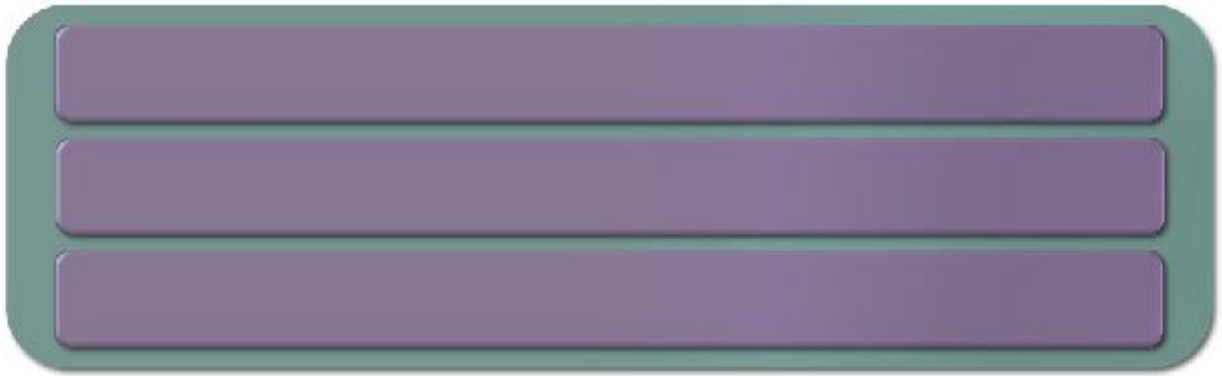
Data Format in SSD

- NSM layout



Tuple based organization

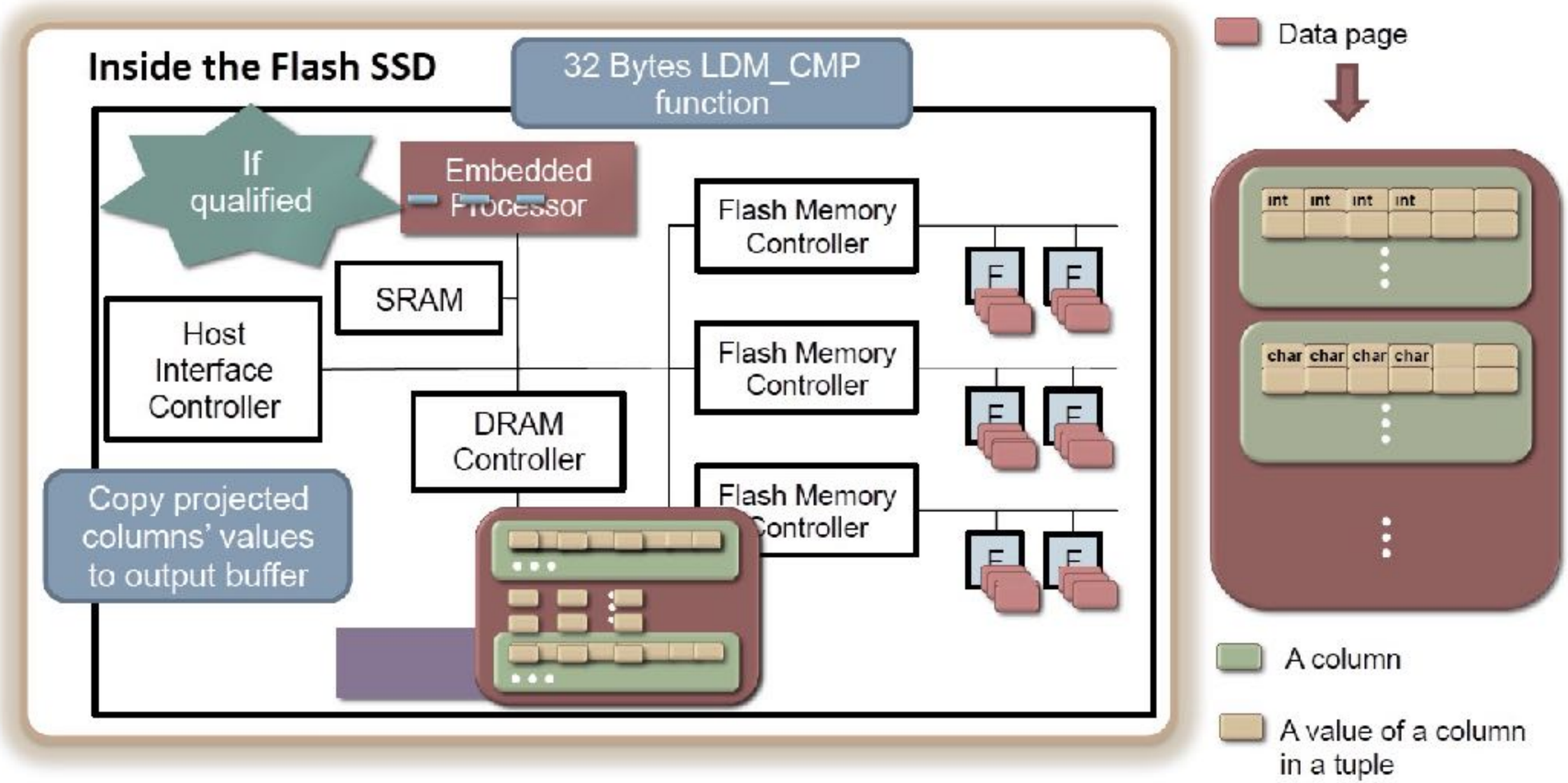
- PAX layout



Column based organization

Operations are performed on values in the same column

Column Format in SSD



Operations are performed on values in the same column

System Configuration

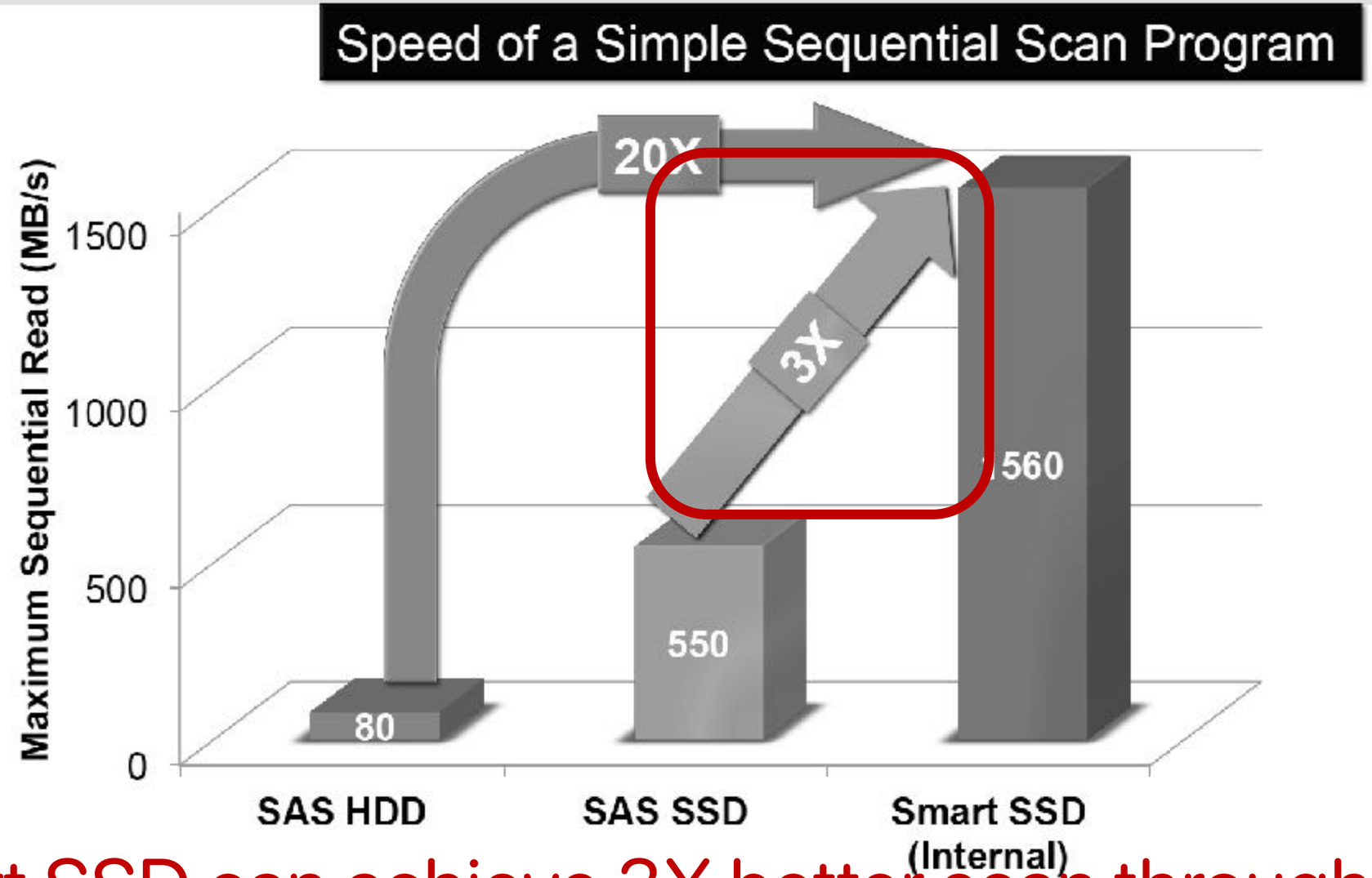
SQL server (Host machine)
64-bit Windows 7
32GB DRAM (24GB dedicated to the DBMS)
2.66 GHz Dual Quad Core Intel Xeon5430 32KB L1 cache, 6MB L2 cache
Two 7.5 RPM SATA HDDs one for the OS and one for the Log
LSI four-port SATA/SAS 6Gbps HBA (host bus adapter)

Flash SSD
Samsung 400GB SAS SSD
Two ARM Cores
16KB SRAM for each core
256KB DRAM for each core
1MB DRAM output buffer

HDD
HP 146GB 10K RPM SAS HDD

SSD cores are lightweight, DRAM capacity is small

Upper Bound on Expected Performance Gains



Smart SSD can achieve 3X better scan throughput than normal SSD

Query Processing on Smart SSDs , SIGMOD 2013

Evaluated Data Sets

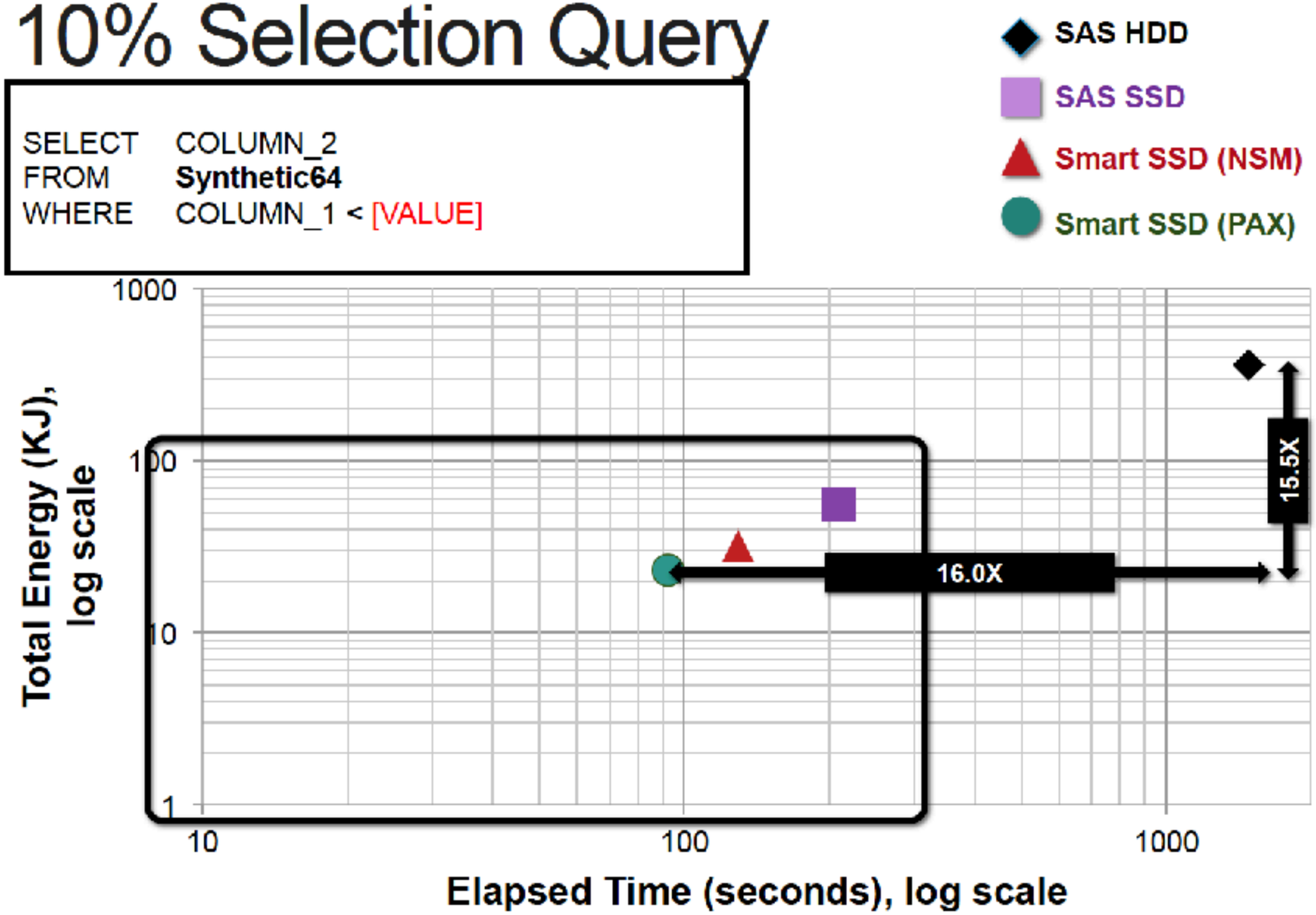
Each table has a number of integer-typed columns

Table	# Columns	Table Size (GB)	# tuples (millions)	# tuples/ page
Synthetic 4	4	10	400	323
Synthetic16	16	30	400	109
Synthetic64	64	110	400	29

Synthetic64 has the largest data size

10% Selection Query

```
SELECT   COLUMN_2
FROM     Synthetic64
WHERE    COLUMN_1 < [VALUE]
```

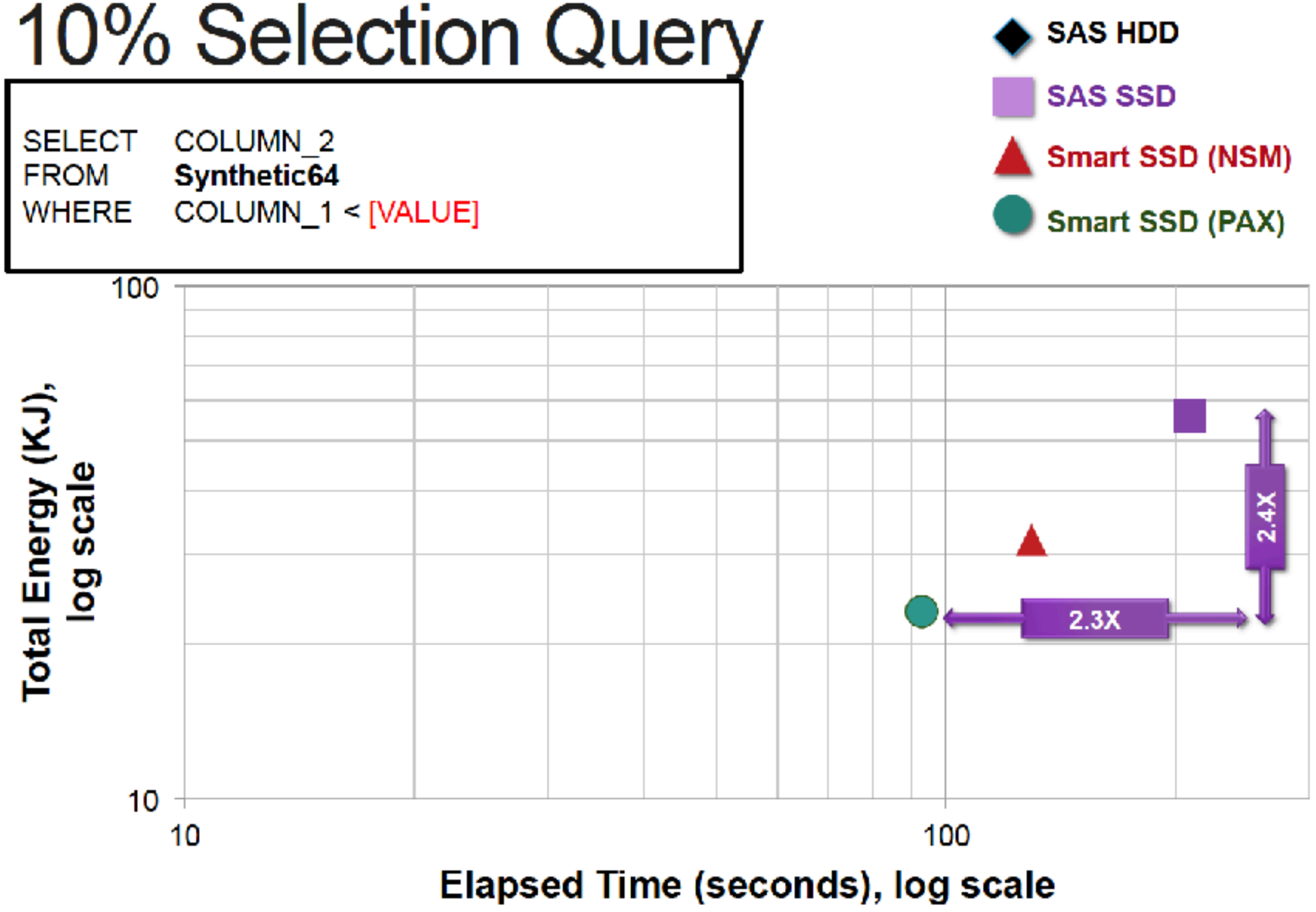


16X better energy and performance than disk

Query Processing on Smart SSDs , SIGMOD 2013

10% Selection Query

```
SELECT   COLUMN_2
FROM     Synthetic64
WHERE    COLUMN_1 < [VALUE]
```



2.5X better energy and performance than SSD

Query Processing on Smart SSDs , SIGMOD 2013

3:00

What do you expect the performance of SmartSSDs on Synthetic 4 of the selection query?

SELECT SecondColumn FROM SyntheticTable WHERE FirstColumn < [VALUE]

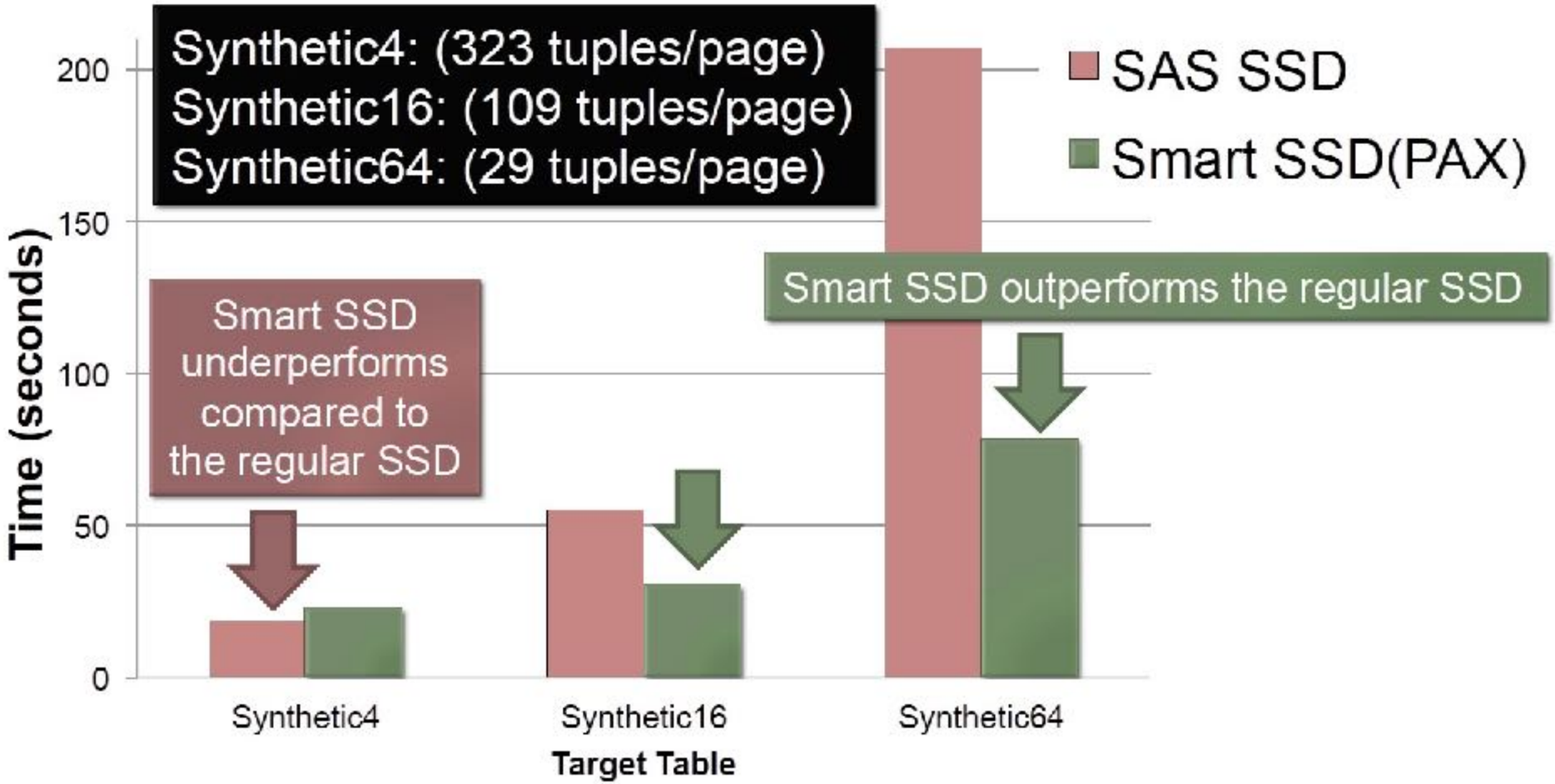
Each table has a number of integer-typed columns

Table	# Columns	Table Size (GB)	# tuples (millions)	# tuples/ page
Synthetic 4	4	10	400	323
Synthetic16	16	30	400	109
Synthetic64	64	110	400	29

Is SmartSSD always a good idea?

Effect of the # of Tuples on a Page

```
SELECT COLUMN_2, FROM [Synthetic_table] WHERE COLUMN_1 < [VALUE]
```

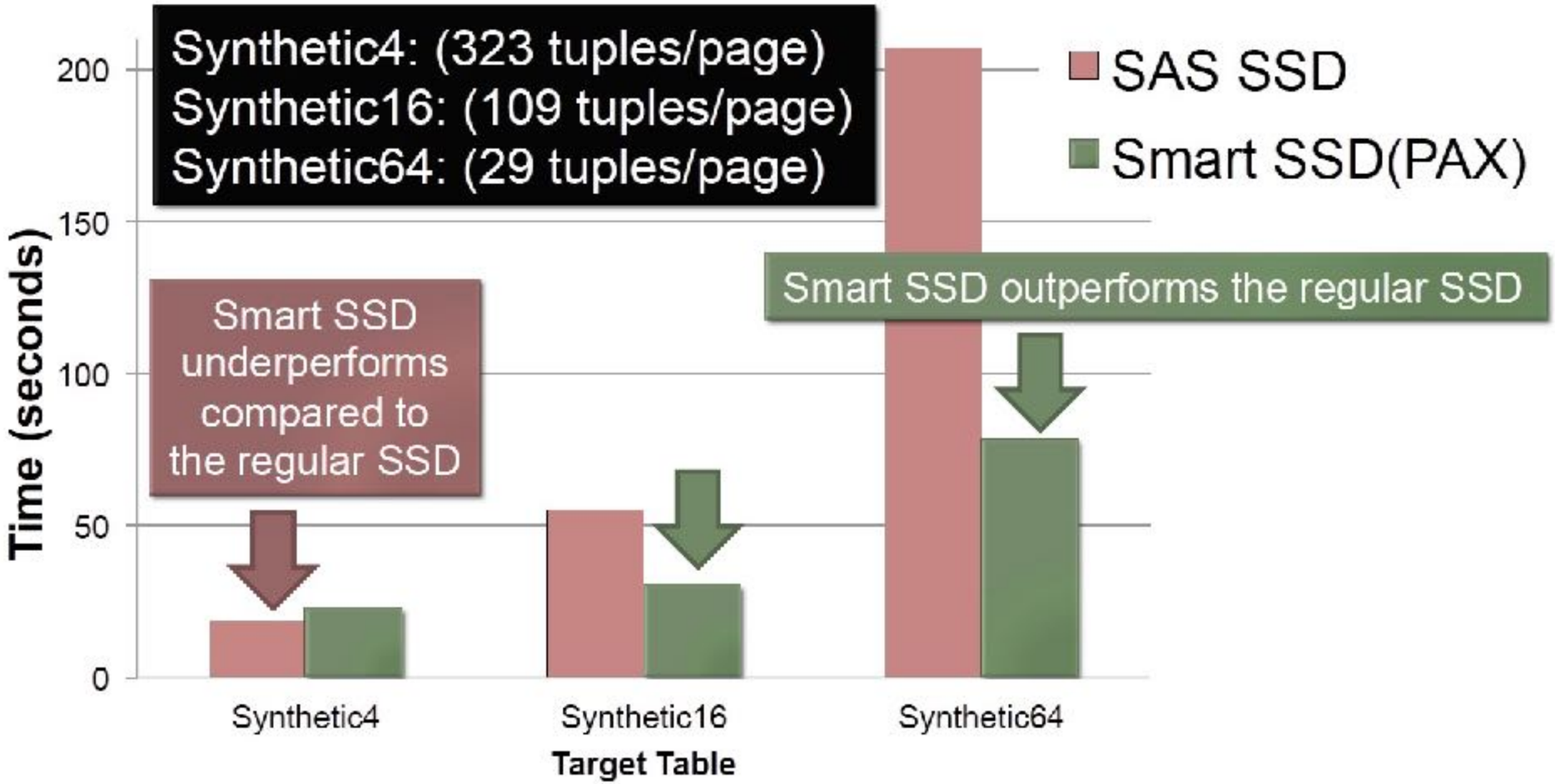


Benefit depends on the dataset size

Query Processing on Smart SSDs , SIGMOD 2013

Effect of the # of Tuples on a Page

```
SELECT COLUMN_2, FROM [Synthetic_table] WHERE COLUMN_1 < [VALUE]
```



Benefit depends on the dataset size

Query Processing on Smart SSDs , SIGMOD 2013

The "Winning Formula" of Near-Data Processing

Conventional Model

ISP Model

The compute intensity is higher in synthetic 4 as there are more tuples per unit of data

$$\frac{DataVolume_{raw}}{Bandwidth_{device}} + \frac{DataVolume_{raw} \times ComputationIntensity_{NDP}}{ComputationThroughput_{device}} + \frac{DataVolume_{afterNDP}}{Bandwidth_{device2host}} + \frac{DataVolume_{afterNDP} \times ComputationIntensity_{afterNDP}}{ComputationThroughput}$$

$$<= \frac{DataVolume_{raw}}{Bandwidth_{device2host}} + \frac{DataVolume_{raw} \times ComputationIntensity}{ComputationThroughput}$$

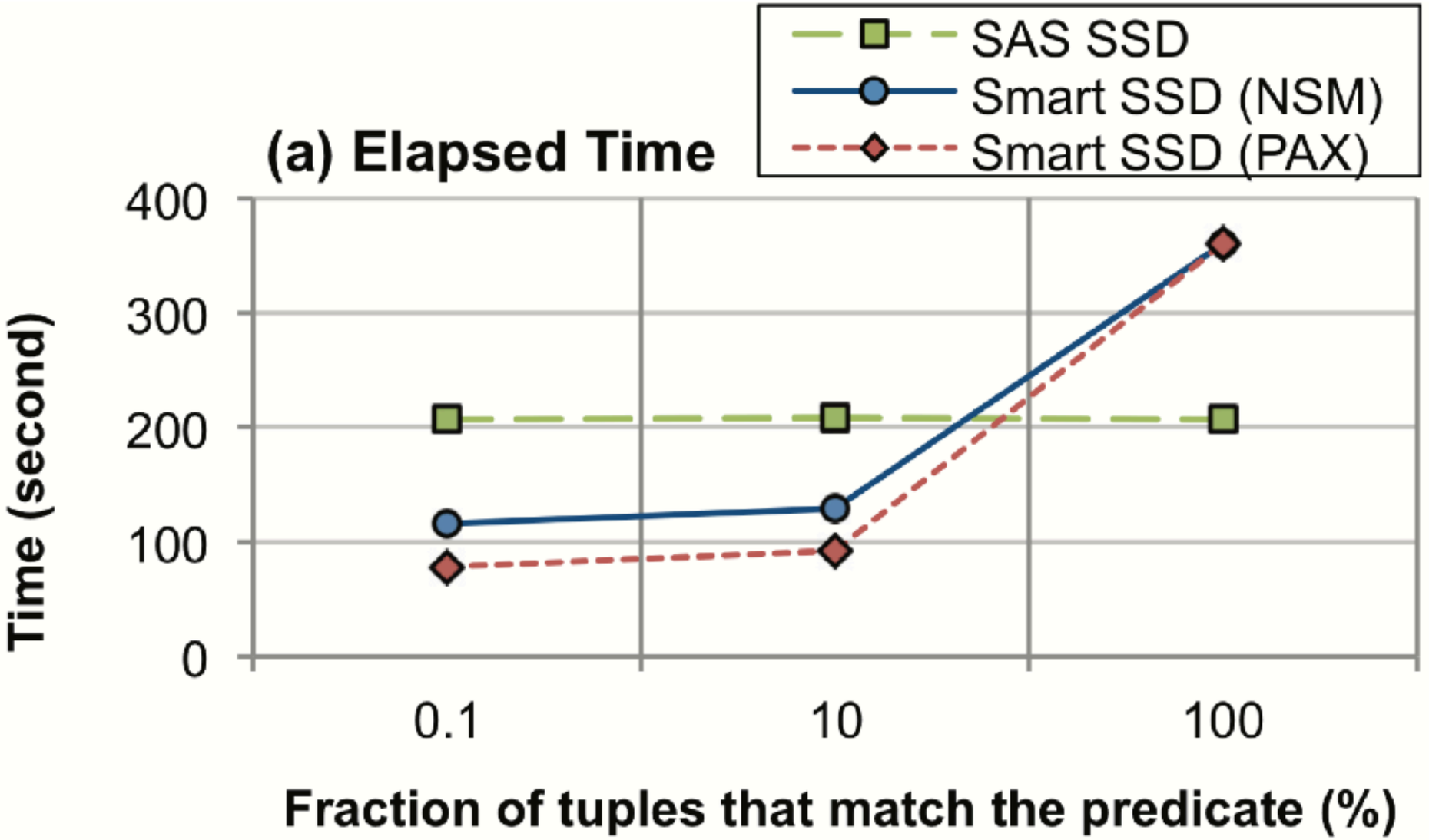
The ISP processor must have efficient internal access to the device

The ISP computation cannot take too long

The ISP should reduce data volume

The ISP should offload the processor

SSD becomes a bottleneck



Memory and compute in SSD is a bottleneck

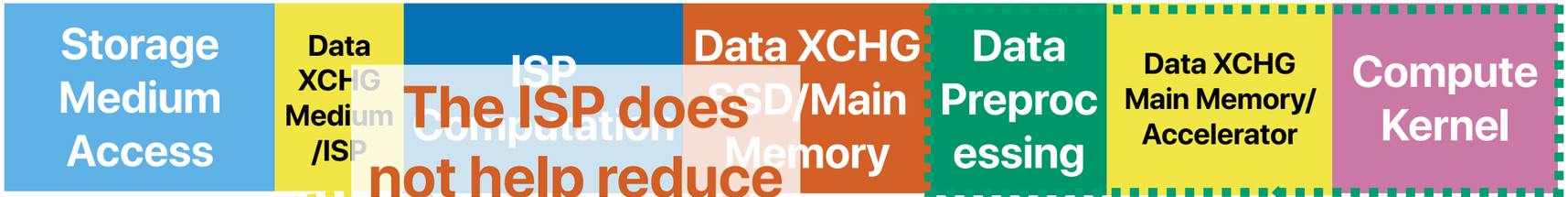
3:00

Why the performance of using SmartSSD is worse if the selectivity is high?

Recap: The "Winning Formula" of In-Storage Processing

Conventional Model

ISP Model



The ISP does not help reduce the volume

$$\frac{DataVolume_{raw}}{Bandwidth_{device}} + \frac{DataVolume_{raw} \times ComputationIntensity_{NDP}}{ComputationThroughput_{device}} + \frac{DataVolume_{afterNDP}}{Bandwidth_{device2host}} + \frac{DataVolume_{afterNDP} \times ComputationIntensity_{afterNDP}}{ComputationThroughput}$$

$$<= \frac{DataVolume_{raw}}{Bandwidth_{device2host}} + \frac{DataVolume_{raw} \times ComputationIntensity}{ComputationThroughput}$$

The ISP processor must have efficient internal access to the device

The ISP computation cannot take too long

The ISP should reduce data volume

The ISP should offload the processor

PRODUCT BRIEF

AMD XILINX

SmartSSD® COMPUTATIONAL STORAGE DRIVE

OVERVIEW

The Samsung SmartSSD computational storage drive (CSD) is the industry's first adaptable computational storage platform. It empowers a new breed of software developers to easily build innovative hardware-accelerated solutions in familiar high-level languages. The SmartSSD dramatically accelerates data intensive applications by 10X or more by pushing compute to where the data lives.

At the heart of the SmartSSD is the Xilinx Adaptive Platform, a powerful toolkit which enables customers to create customized, scalable applications to solve a broad range of data center problems.

CHALLENGE

Turning Big Data Into Fast Data

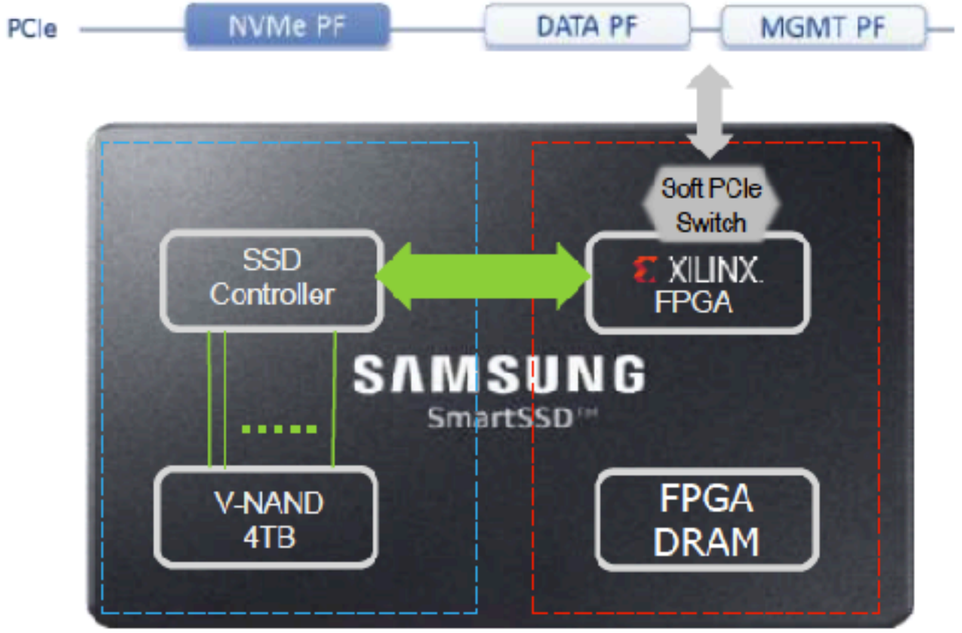
The global datasphere will grow from 45 zettabytes in 2019 to 175 by 2025, and nearly 30% of the world's data will need real- time processing. This data explosion provides tremendous opportunity for business insights, but the sheer volume

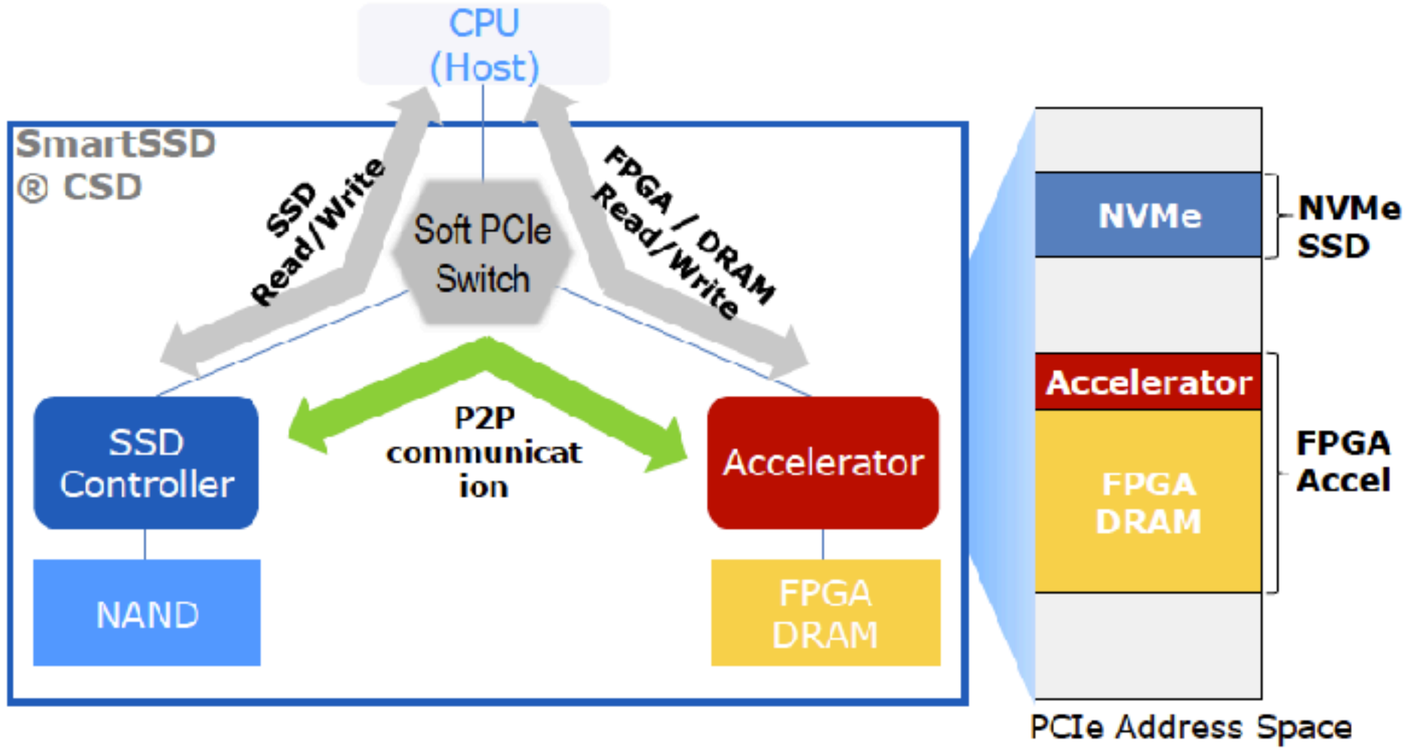
TARGET APPLICATIONS

- AI/ML Inference
- Big Data Analytics

SmartSSD® CSD HW Architecture

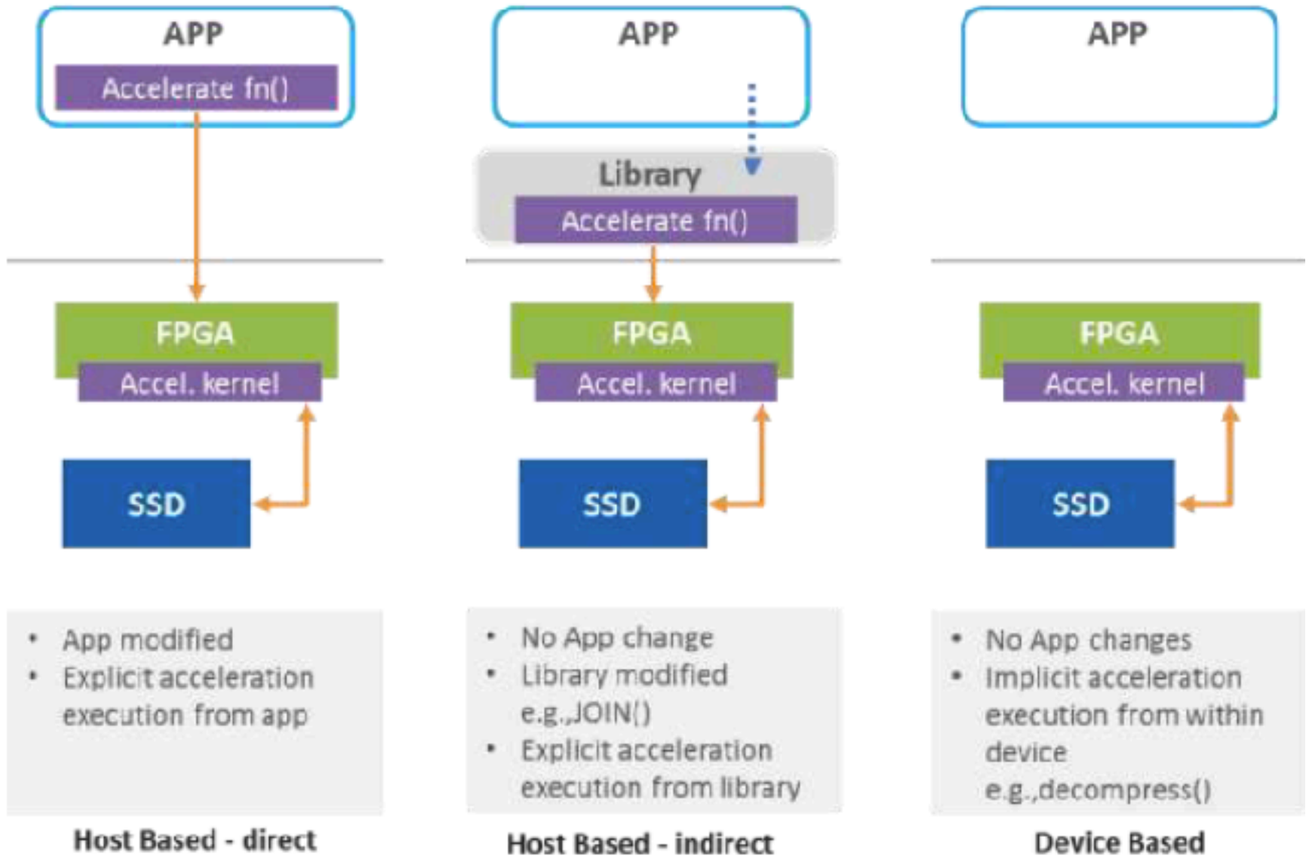
- Peer-to-peer (P2P) communication enables unlimited concurrency
  - SSD:Accelerator data transfers use internal data path
    - Save precious L2:DRAM Bandwidth (Compute Nodes) • Scale without costly x86 frontend (Storage Nodes)
    - Avoid the unnecessary funneling and data movement of standalone accelerators
  - FPGA DRAM is exposed to Host PCIe address space
    - NVMe commands can securely stream data from SSD to FPGA peer-to-peer





Developing on SmartSSD® CSD

- Frameworks supported by partners
  - Spark, Kafka available
  - FFmpeg coming
  - Many more in development
- Supported OSes
  - Linux
  - FreeBSD
  - Windows Server
- Ease of porting for SDAccel OpenCL developers
- Vivado-friendly for RTL developers



SmartSSD® OpenCL Programming in 5 Steps

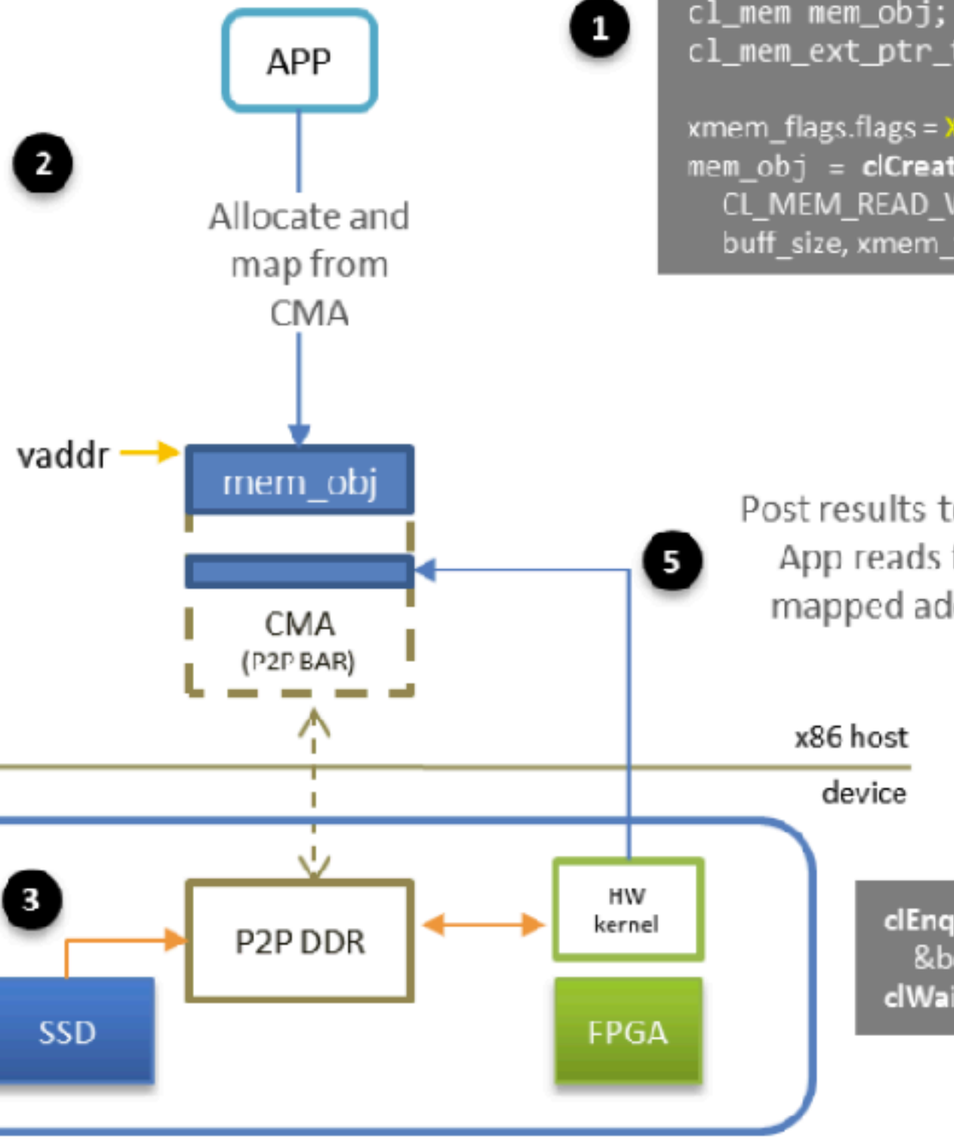
- Secure, P2P data movement
  - Data moves in/out of SSD only under control of storage stack (NVMe)

```
cl_mem mem_obj;
cl_mem_ext_ptr_t xmem_flags = {0};

xmem_flags.flags = XCL_MEM_EXT_P2P_BUFFER;
mem_obj = clCreateBuffer(dev_cxt,
  CL_MEM_READ_WRITE | CL_MEM_EXT_PTR_XILINX,
  buff_size, xmem_flags, &err);
```

1

```
void *vaddr;

vaddr = clEnqueueMapBuffer(cmd_queue, cl_buff,
  CL_TRUE, host_access, 0, buff_size, 0, NULL, NULL,
  &err);
```

2




```
bytes_read = read(fd, vaddr, buff_size);
```

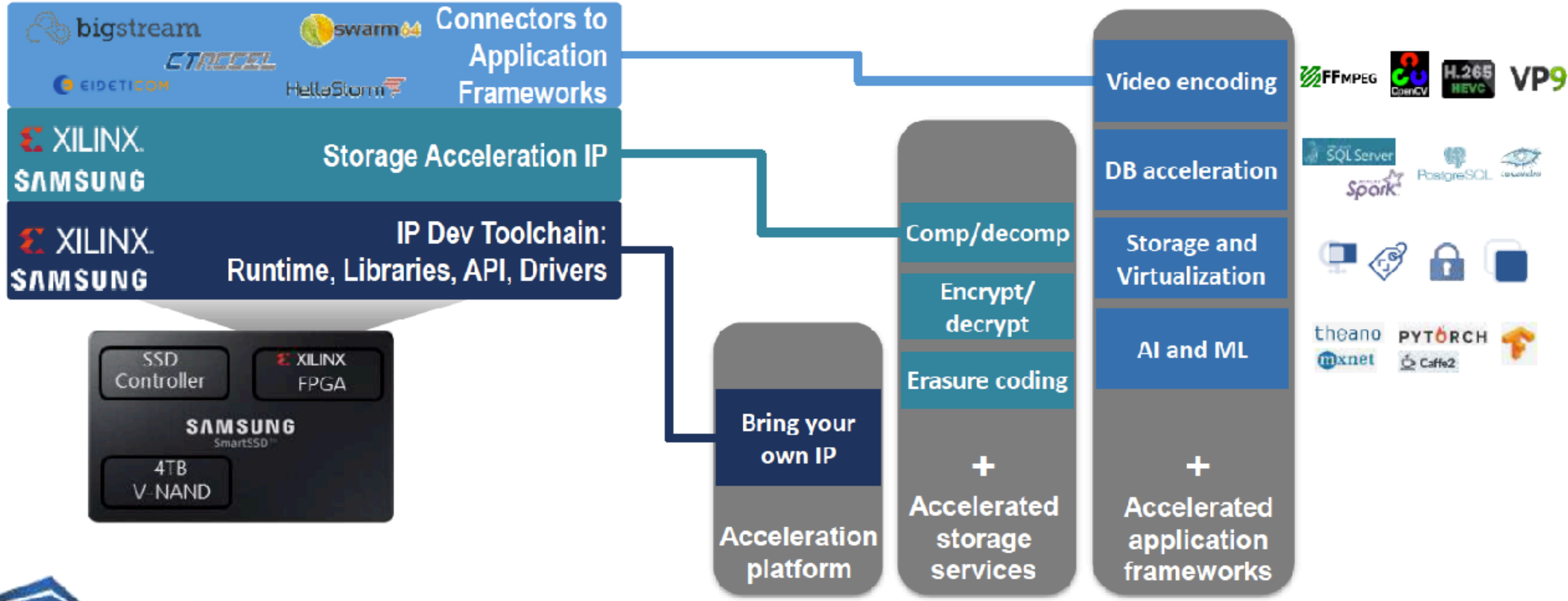
Initiate peering transfers from SSD

4

```
clEnqueueTask(cmd_queue, hw_kernel, 0, NULL,
  &bc_event);
clWaitForEvents(1, &bc_event);
```

SmartSSD® CSD Use Cases and Ecosystem

- **Storage Services:** Comp/Decomp, Encryp/Decrypt, Metadata management, Erasure Coding,
- **Real-time Analytics & Biz Intelligence:** DB Query (Spark, PostgreSQL, ..), Log analytics, genomics, physics
- **Rich Media and ML:** transcoding, live streaming, object detection



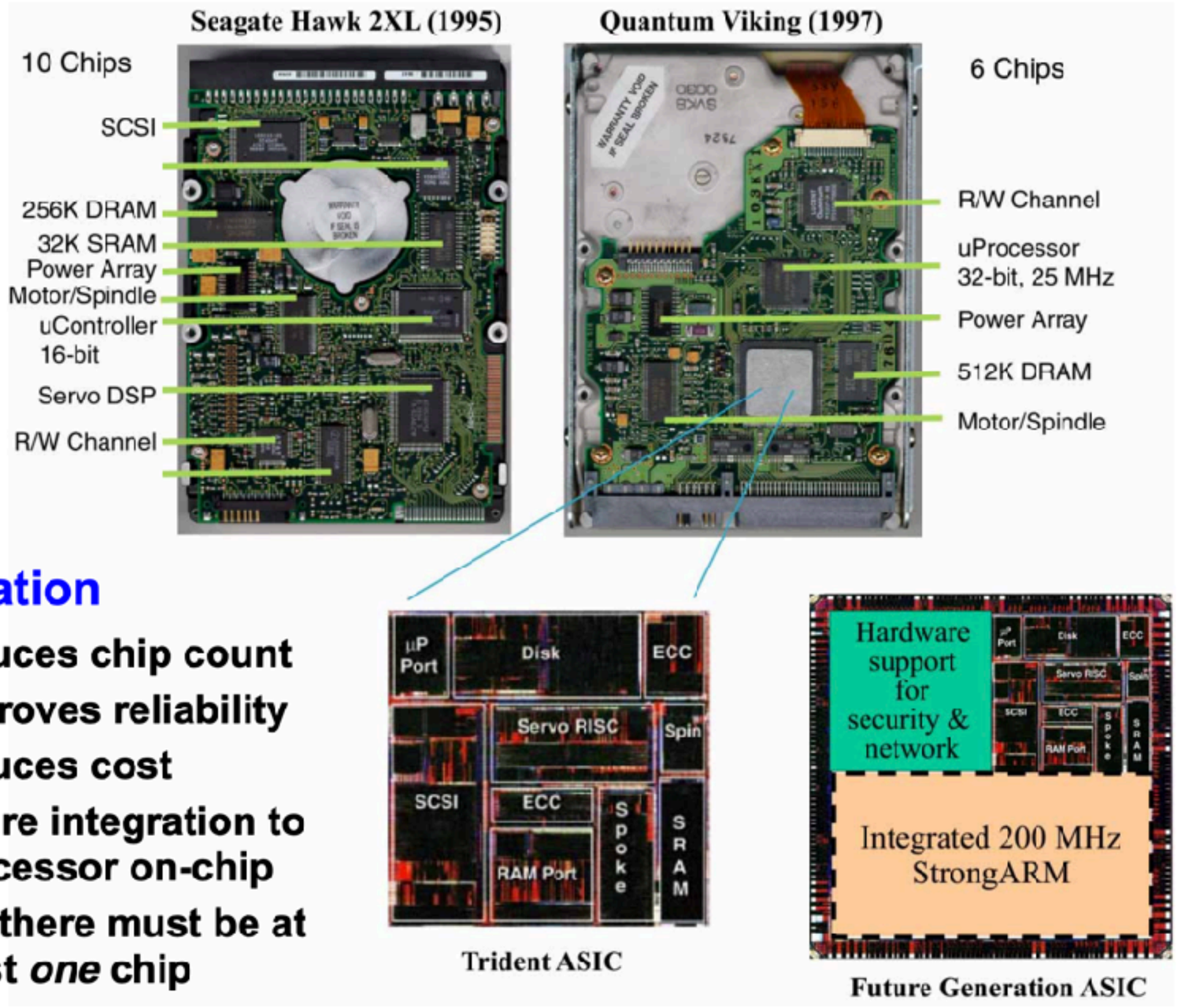
In-storage processing isn’t a new idea…

Active disks: programming model, algorithms and evaluation

Anurag Acharya, Mustafa Uysal, and Joel Saltz.
In Proceedings of the eighth international conference on Architectural support for programming languages and operating systems (ASPLOS VIII), 1998

http://www.cs.cmu.edu/~riedel/defense/talk99_ActiveDisksForDatabases.pdf

Evolution of Disk Drive Electronics

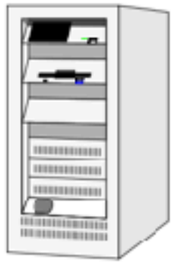


http://www.cs.cmu.edu/~riedel/defense/talk99_ActiveDisksForDatabases.pdf

Opportunity

TPC-D 300 GB Benchmark, Decision Support System

Database Server

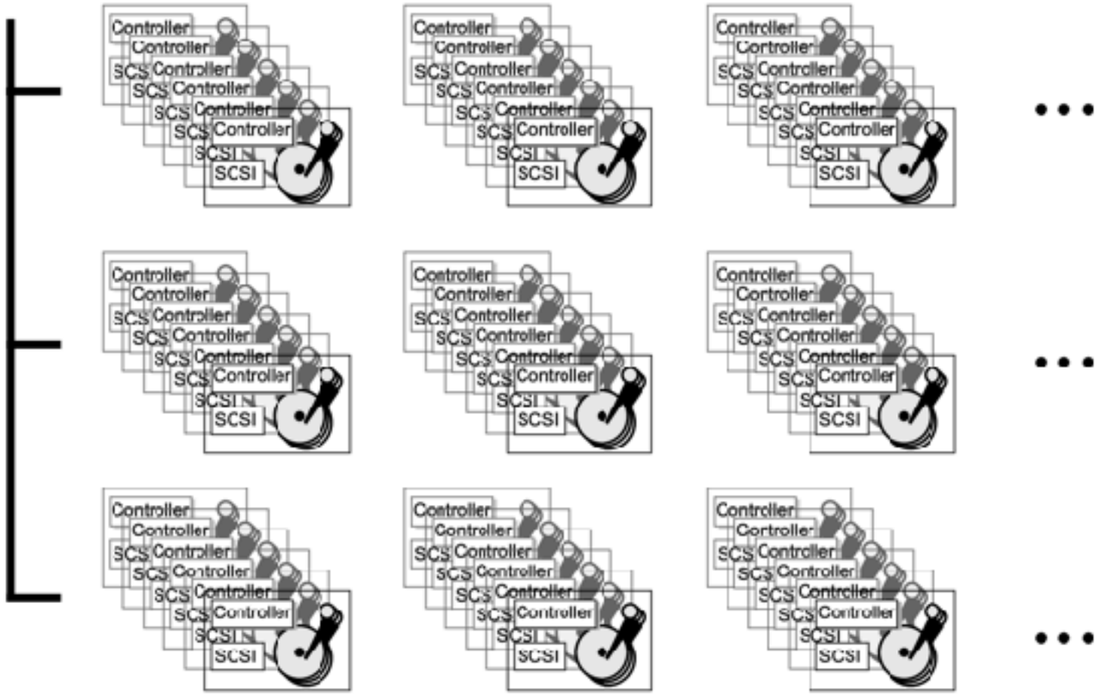


Digital AlphaServer 8400 = 7,344 total MHz
- 12 x 612 MHz 21164
- 8 GB memory
- 3 64-bit PCI busses — *3 x 266 = 798 MB/s*
- 29 FWD SCSI controllers — *29 x 40 = 1,160 MB/s*

Storage
- 520 rz29 disks
- 4.3 GB each
- 2.2 TB total

= 104,000 total MHz
(with 200 MHz drive chips)

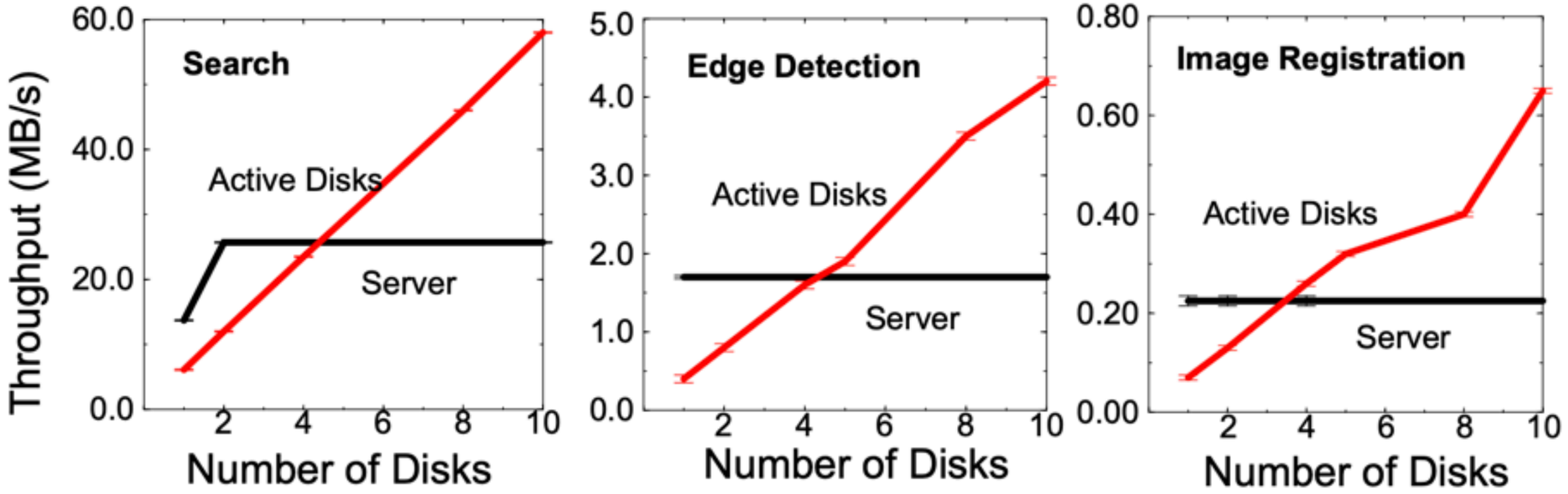
= 5,200 total MB/s
(at 10 MB/s per disk)



http://www.cs.cmu.edu/~riedel/defense/talk99_ActiveDisksForDatabases.pdf

Performance with Active Disks

application	input	computation (inst/byte)	throughput (MB/s)	memory (KB)	selectivity (factor)	bandwidth (KB/s)
Search	k=10	7	28.6	72	80,500	0.4
Frequent Sets	s=0.25%	16	12.5	620	15,000	0.8
Edge Detection	t=75	303	0.67	1776	110	6.1
Image Registration	-	4740	0.04	672	180	0.2



3:00

Why “active disks” did not work out in real practice?

Why isn't ActiveDisk working?

Recap: The "Winning Formula" of In-Storage Processing

Conventional Model

ISP Model

$$\frac{DataVolume_{raw}}{Bandwidth_{device}} + \frac{DataVolume_{raw} \times ComputationIntensity_{NDP}}{ComputationThroughput_{device}} + \frac{DataVolume_{afterNDP}}{Bandwidth_{device2host}} + \frac{DataVolume_{afterNDP} \times ComputationIntensity_{afterNDP}}{ComputationThroughput}$$

$$<= \frac{DataVolume_{raw}}{Bandwidth_{device2host}} + \frac{DataVolume_{raw} \times ComputationIntensity}{ComputationThroughput}$$

The ISP processor must have efficient internal access to the device

The ISP computation cannot take too long

The ISP should reduce data volume

The ISP should offload the processor

Why Active Disks did not work out?

- Computation still dominates at that era
- Magnetic disks do not really have too much to do with “internal bandwidth”
- Independent advances like distributed storage is similar to active disk
- New features at added cost may not be used by many and convincing system vendors to implement support is hard
- Business model — hard disk drives focus on low cost per unit of data

3:00

Can we summarize when will ISP/NDP work? Do you think there is a future of ISP/NDP?

Is ISP/NDP always a good idea?

When can ISP fail?

Conventional Model

ISP Model

- The ISP computation does not help offload computation
- The ISP computation does not help reduce data volume
- The ISP computation does not facilitate data preprocessing
- The ISP computation is compute intensive

Challenges of ISP

- Programming is still an issue
  - Do you want to program at the level of FPGAs?
- Applications
  - Low compute intensity
  - Volume reduction after pre-processing
- Hardware capability and cost
  - What kind of processors can we have in the storage device?

Example — Varifocal Storage✳

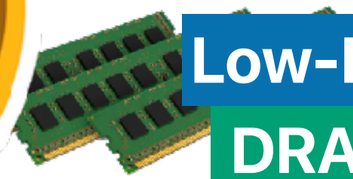
✳ Yu-Ching Hu, Murtuza Lokhandwala, Te I and Hung-Wei Tseng. Dynamic Multi-Resolution Data Storage. In the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019 (Best paper nominee)

Accelerators in General-Purpose Computers

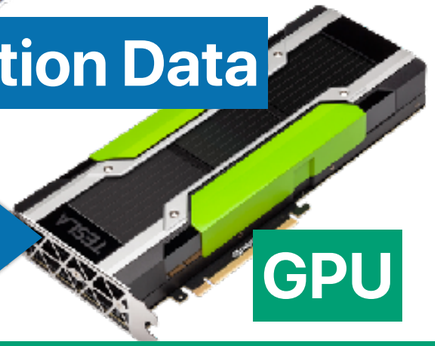
Retrieve Raw Data

Adjust Data Resolutions

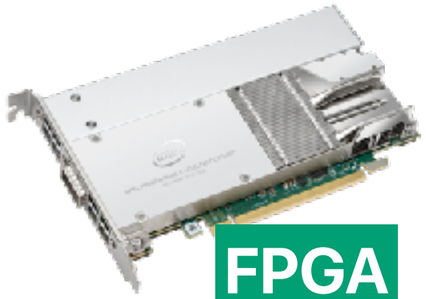
Low-Resolution Data



DRAM

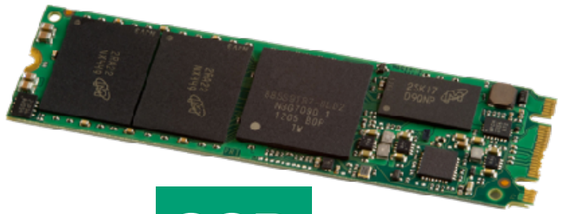


GPU



FPGA

Raw Data



SSD

CPU

Compute

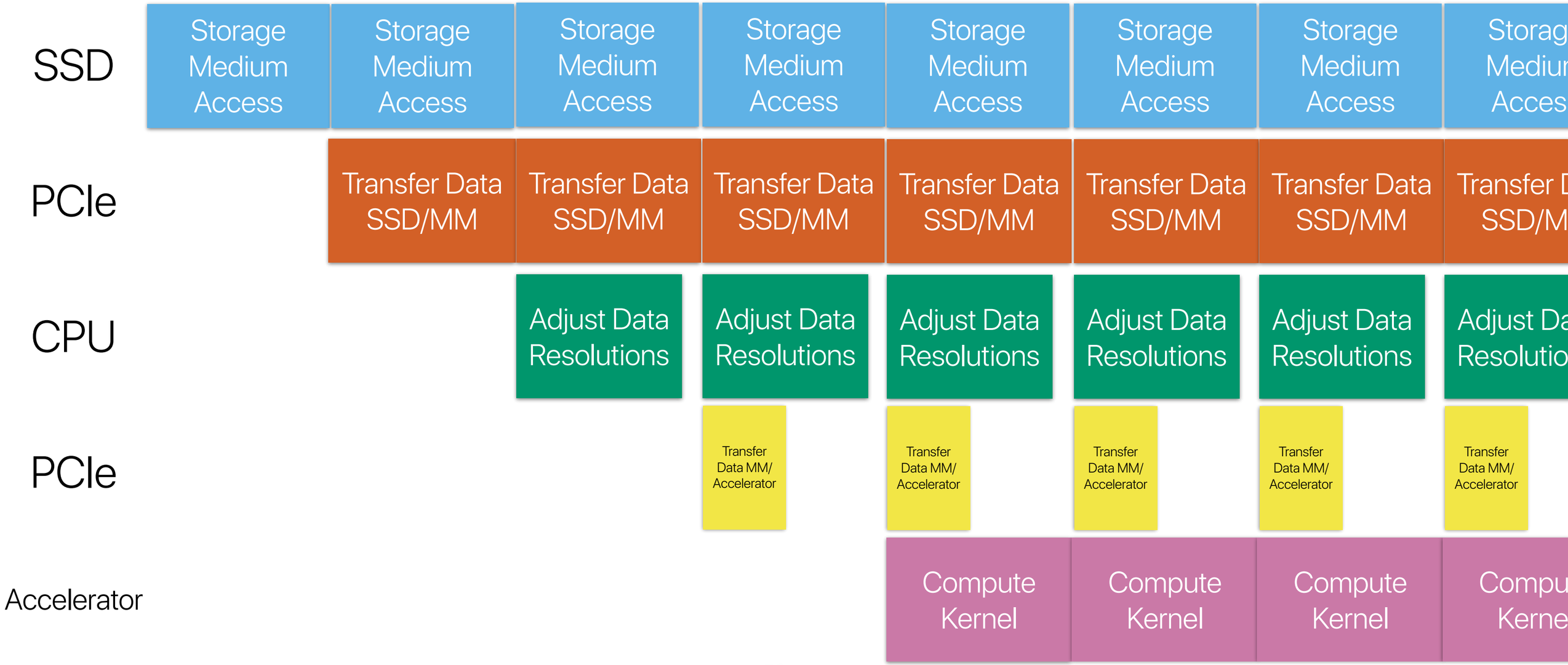
TPU

System Interconnect (e.g., PCIe)

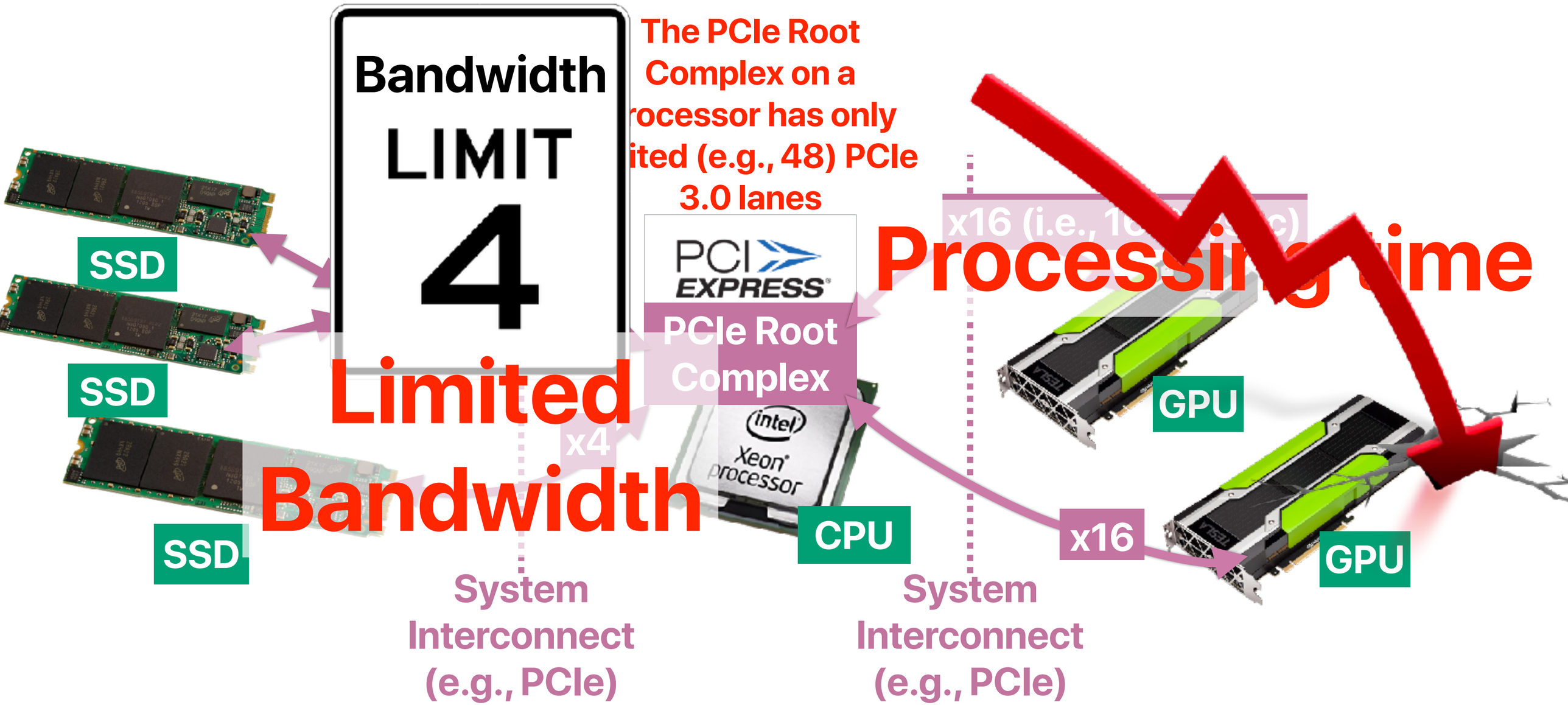
Quality Control

(e.g., PCIe)

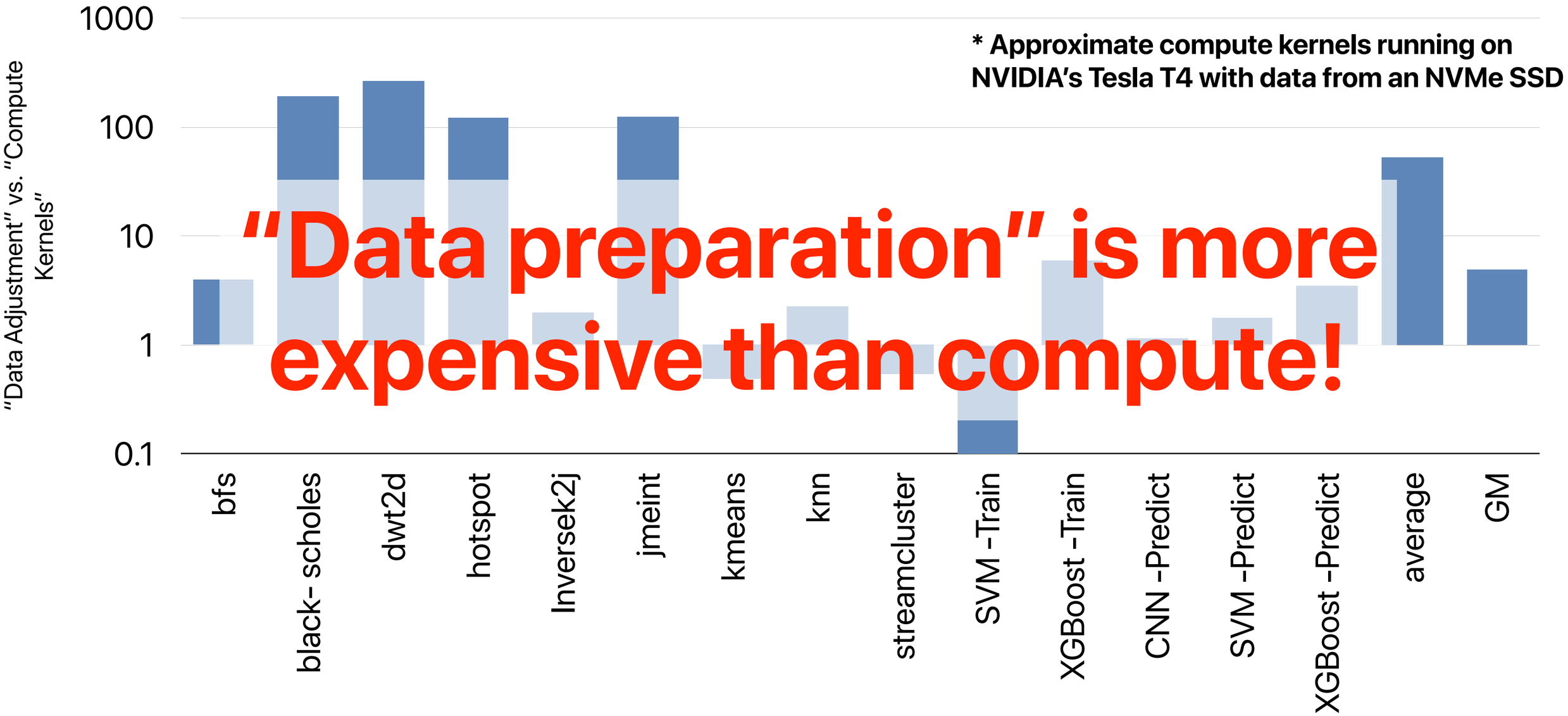
Data processing pipeline in modern computers



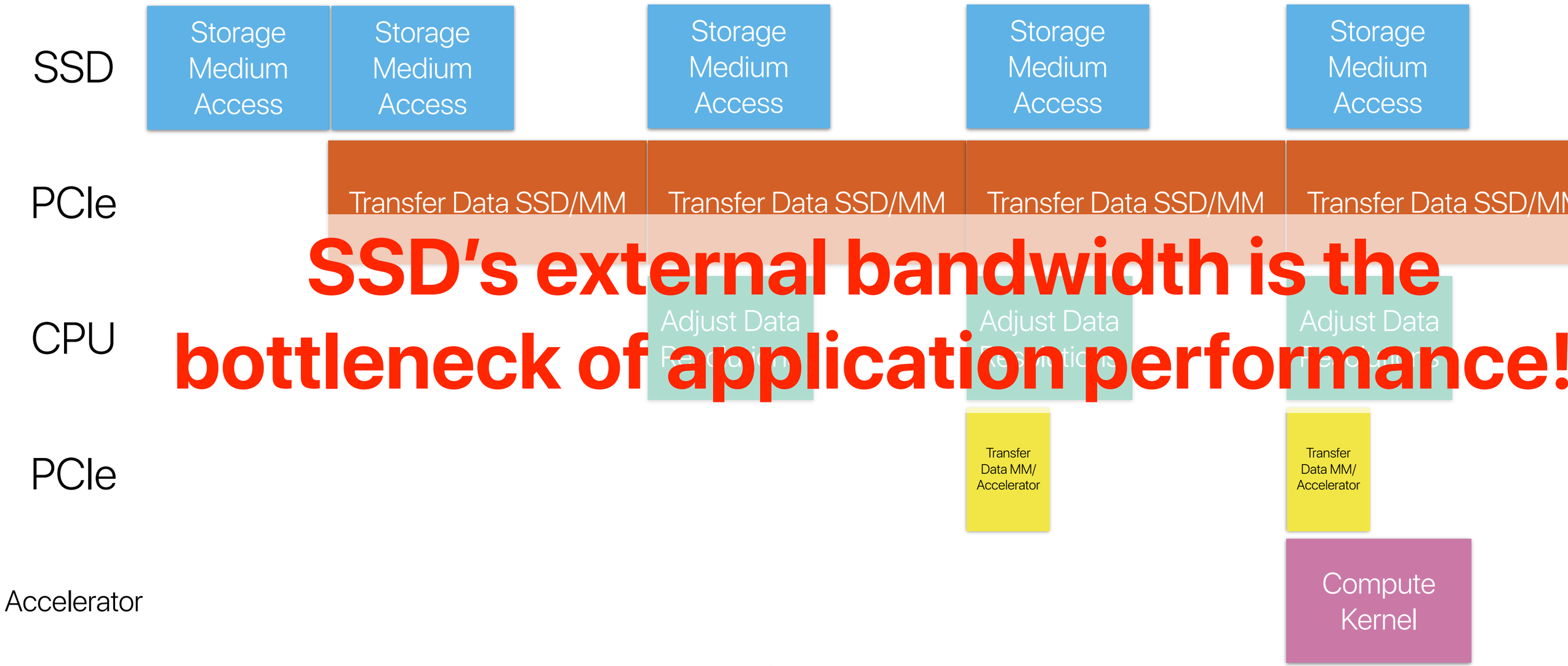
Limited Interconnect Bandwidth



Data Preparation Can be 100x More Than Compute Kernels



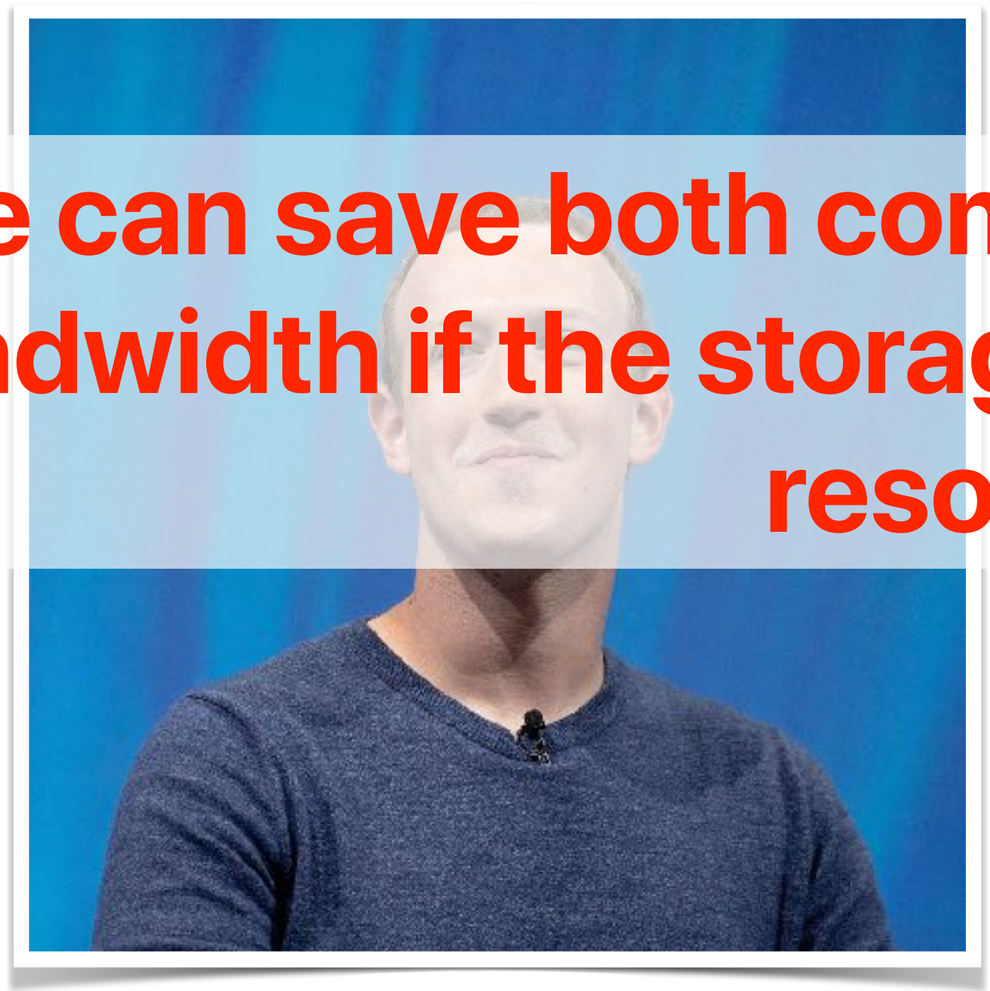
The “actual” pipeline …

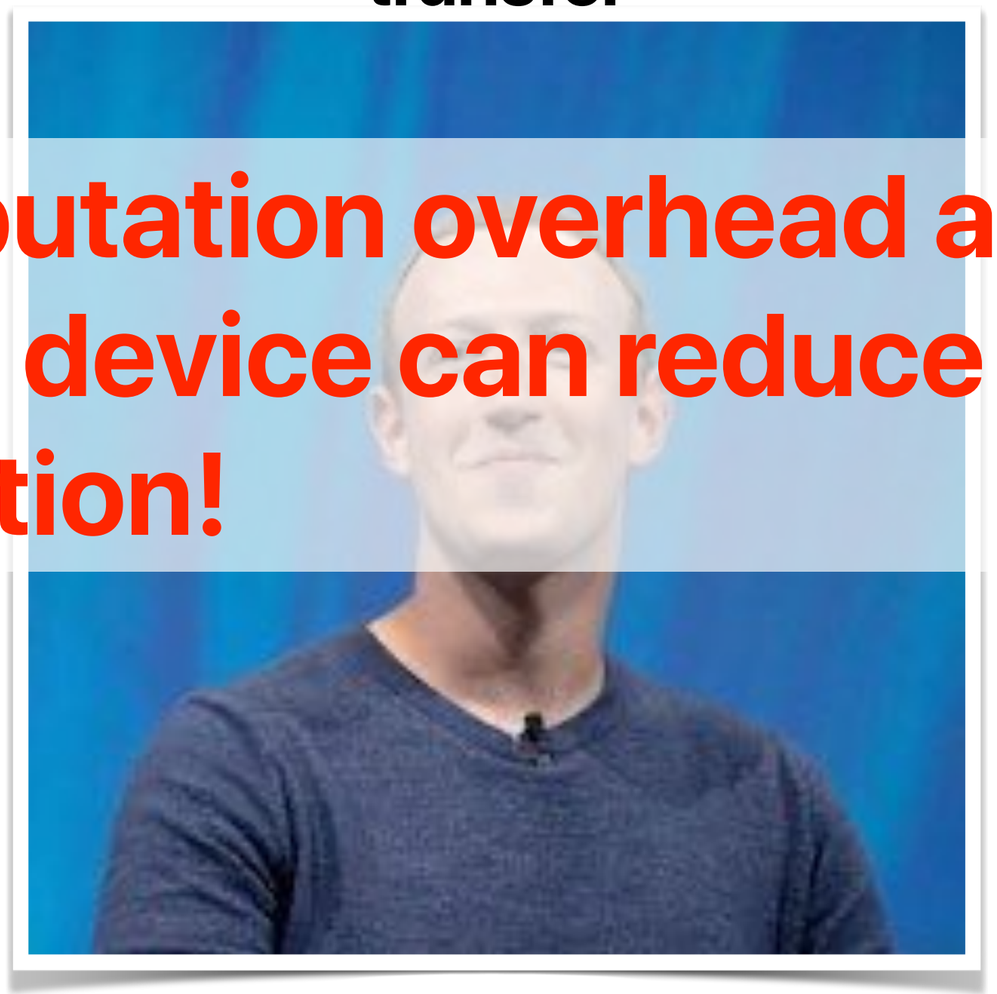


SSD’s external bandwidth is the bottleneck of application performance!

We don’t need full resolution in many cases

Only need 25% bytes to transfer





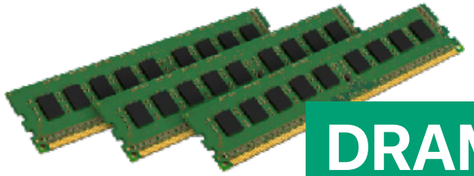
We can save both computation overhead and bandwidth if the storage device can reduce the resolution!

Varifocal Storage: A Holistic System Architecture

Retrieve Raw Data

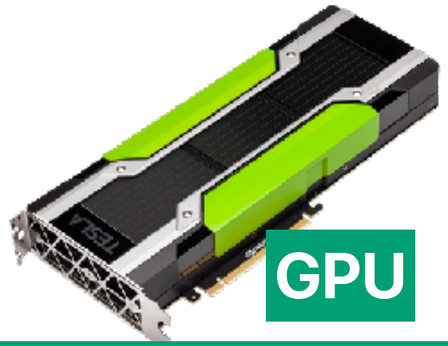
Adjust Data Resolutions

Varifocal Storage (VS)

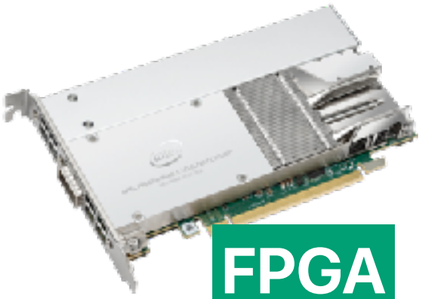


DRAM

CPU



GPU



FPGA

Compute

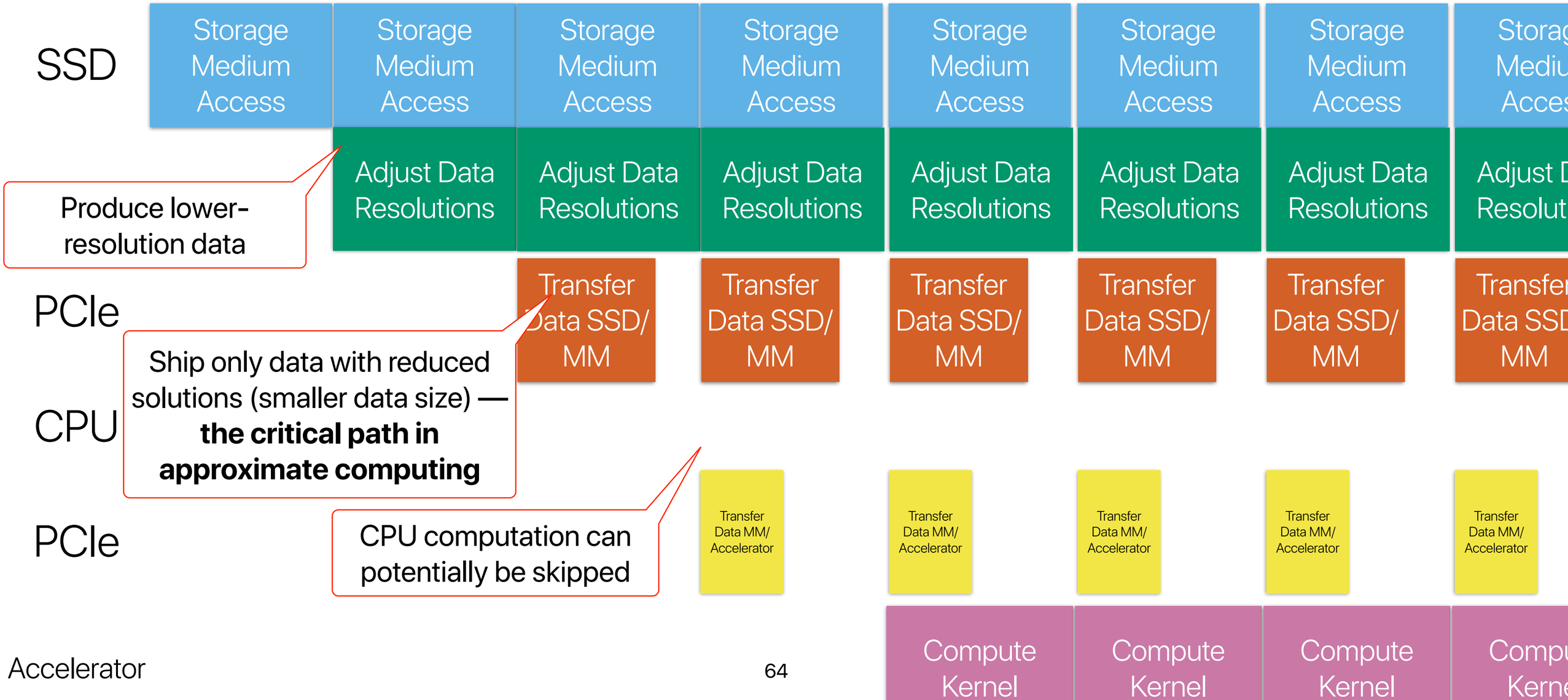
TPU

System Interconnect (e.g., PCIe)

Quality Control

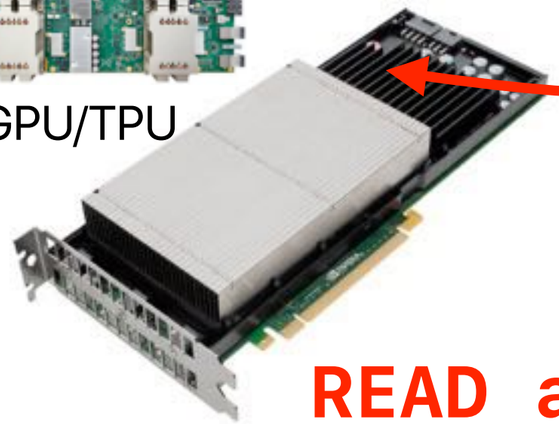
(e.g., PCIe)

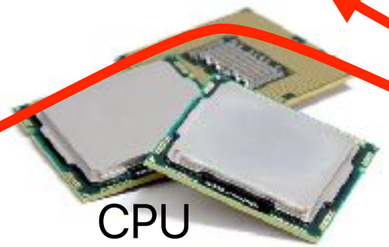
Data processing with ISP



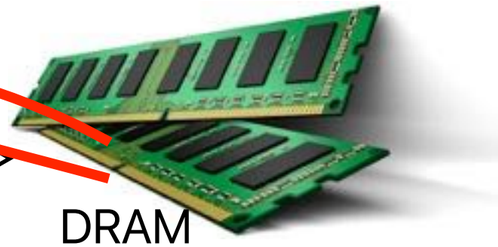
The “concept” of Varifocal Storage

GPU/TPU



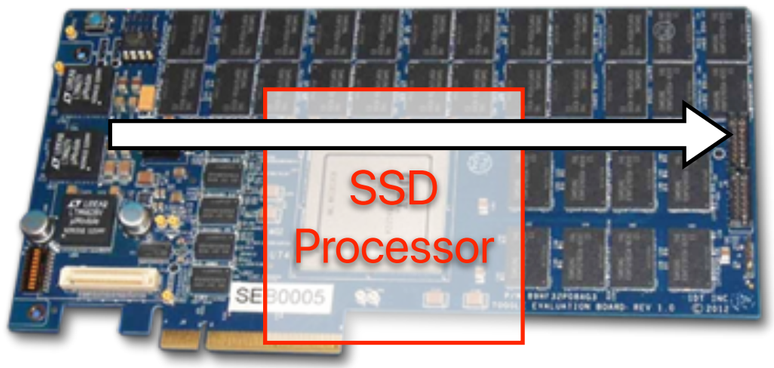


CPU



DRAM

`READ address(mark_zuckerberg), reduce(25%)`



extension of
NVMe interface

The “Winning Formula” of Near-Data Processing

- The NDP computation can help offload computation
- The NDP computation can help reduce data volume
- The NDP computation can facilitate data preprocessing
- The NDP computation itself cannot be too slow

$$\frac{8GB}{8GBps} + \frac{8GB \times 0.25}{2GOPS} + \frac{2GB}{2.5GBps} + \frac{2GB \times 1}{1000GOPS} = 2.4$$

$$\frac{DataVolume_{raw}}{Bandwidth_{device}} + \frac{DataVolume_{raw} \times ComputationIntensity_{NDP}}{ComputationThroughput_{device}} + \frac{DataVolume_{afterNDP}}{Bandwidth_{device2host}} + \frac{DataVolume_{afterNDP} \times ComputationIntensity_{afterNDP}}{ComputationThroughput}$$

$$<= \frac{DataVolume_{raw}}{Bandwidth_{device2host}} + \frac{DataVolume_{raw} \times ComputationIntensity}{ComputationThroughput}$$

$$\frac{8GB}{2.5GBps} + \frac{8GB \times 1}{1000GOPS} = 3.2$$

NVMe

- The standard of PCIe SSD devices now
  - Provides multiple command queues to better support multithreading hardware
  - Allows more parallelism inside the SSD
- The "payload" of a PCIe packet

VS_READ

REDUCE/QUTIZATION/
PACKING/SAMPLING

PARAMETER

0	8	16	32	48	63
OPCODE	FLAGS	command id	Namespace ID		
reserved					
metadata					
PRP1					
PRP2					
Start LBA					
length		control	Dataset management		
Reference tag			App tag	App mask	

Programming model of Varifocal Storage

```
int setup(int argc, char **argv)  {
    // Skip – the rest of code ...
    int infile;

    // VS: Declare VS variables
    struct vs_operator op[1];
    struct vs_feedback fb[1];

    // Open a file descriptor
    infile = open(filename, O_RDONLY, "0600");

    // Read precise data from the file descriptor
    read(infile, &npoints,   sizeof(int));
    read(infile, &nfeatures, sizeof(int));

    // Skip – some other initialization code ...

    // VS: set parameters for desired operator
    // PACKING(default)/PACKING_AF(autofocus)/VS_IF(iFilter)
    op[0].op = PACKING;
    op[0].resolution = HALF;

    // Skip – some other initialization code ...
    // VS: apply the desired VS operator for the file
    vs_setup(infile, &op, “%f”);
    // VS: read data processed by the VS operator
    vs_read(infile, buf, npoints*nfeatures*sizeof(float), &fb);
    // VS: disable the usage of VS operator for the file
    vs_release(infile);
    // Skip – the rest of code ...
    // VS: use approximate kernel if the operator succeed
    if(fb[0].resolution == op[0].resolution)
        cluster_approximate(…);
    else
        cluster(…);
    // Skip – the rest of code ...
}
```

```
void test_distributed_page_rank(char* graphfilename,
int num_ofVertex, int num_ofEdges, int iterations)
{
    FILE *fin;
    vs_stream input_stream;
    void **arg_list;
    fin = fopen(graphfilename, "r");
    vs_input_stream = vs_stream_create(fin);
    Edge *edge_array = (Edge *)malloc(
                              sizeof(Edge)*num_ofEdges);
    inputApplet(input_stream, edge_array);
    // The rest of code ...

}
```

```
StorageApp int inputApplet
(ms_stream ssd_input_stream, void *edge_array)
{
    Edge ssd_edge_array[4096];
    int i = 0;
    while(vs_scanf(ssd_input_stream, "%d %d",
                   &ssd_edge_array[i%4096].first,
                   &ssd_edge_array[i%4096].second)==2)
    {
        i++;
        if(i % 4096 == 0)
        {
            vs_memcpy(edge_array, ssd_edge_array,
                      sizeof(Edge)*4096);
            edge_array = sizeof(Edge)*4096;
        }
    }
    vs_memcpy(edge_array, ssd_edge_array,
                          sizeof(Edge)*(i%4096));
    return i;
}
```

Programmer has to implement

Traditional Quality Control

Raw Data

Lower-resolution

25% Resolution

Compute

Test

Tiger

Reject Result

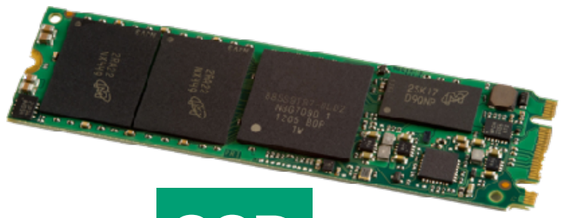
Better-resolution

50% Resolution

Re-compute

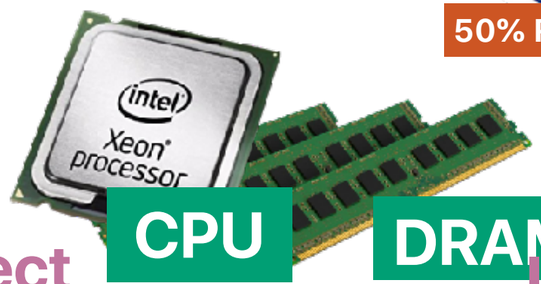
Test

Bear



SSD

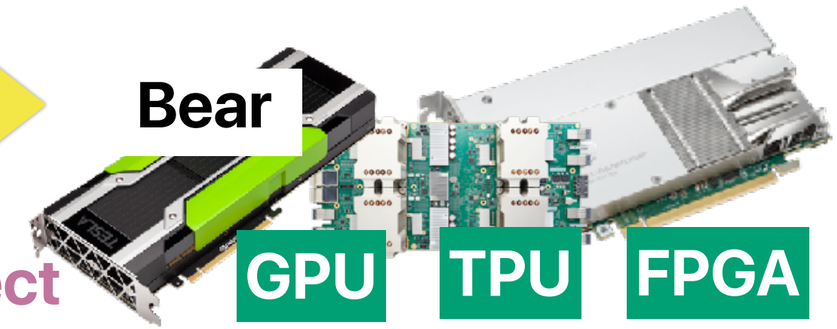
System Interconnect (e.g., PCIe)



CPU

DRAM

System Interconnect (e.g., PCIe)

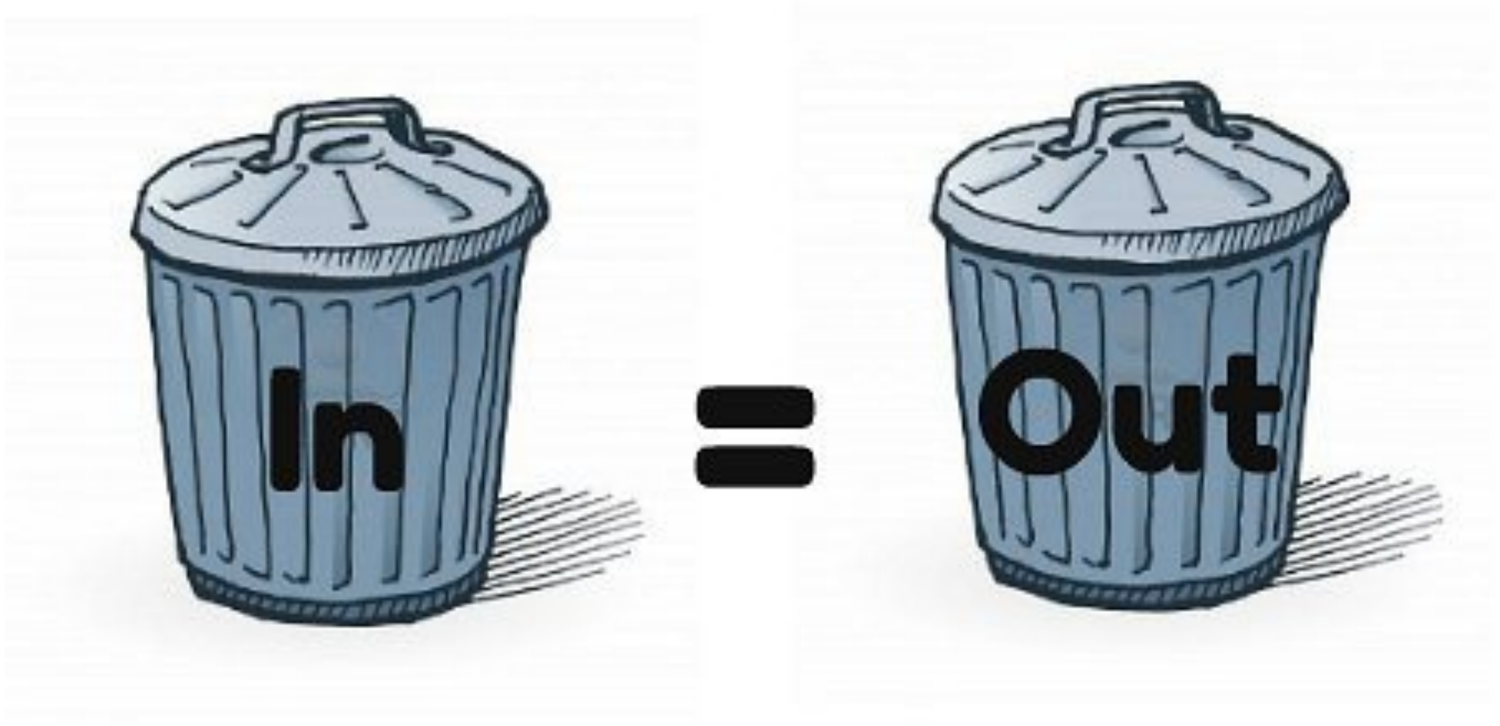


GPU

TPU

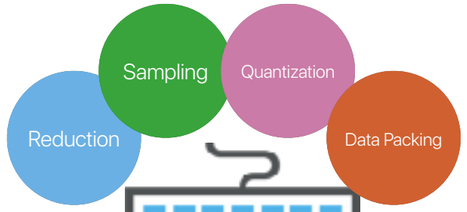
FPGA

Garbage in, garbage out



Quality Control

Autofocus



Raw Data

Lower-resolution

25% Resolution

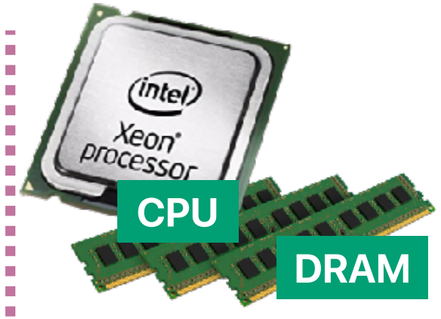
Reject if it becomes garbage

Better-resolution

50% Resolution

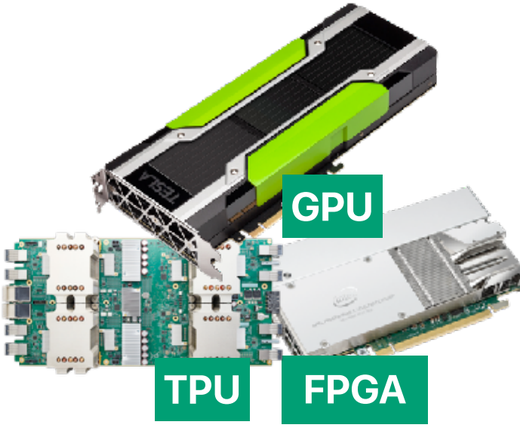
Use this resolution for the same dataset later

Varifocal Storage (VS)



CPU

DRAM



GPU

TPU

FPGA

Pass!

Compute

Test

Bear

System Interconnect (e.g., PCIe)

System Interconnect (e.g., PCIe)

iFilter

Autofocus on Operator A

25% Resolution

Autofocus on Operator B

45% Resolution

Autofocus on Operator C

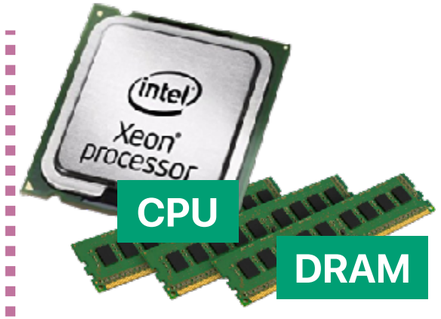
50% Resolution

Varifocal Storage (VS)

Pass!

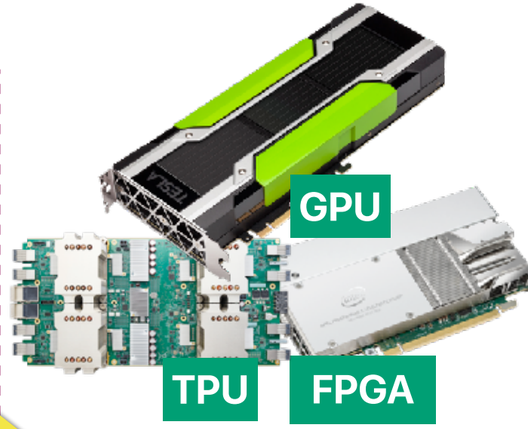
Use this resolution/ operator for the same dataset later

Pass!



CPU

DRAM



GPU

TPU

FPGA

Send Smaller One & Compute

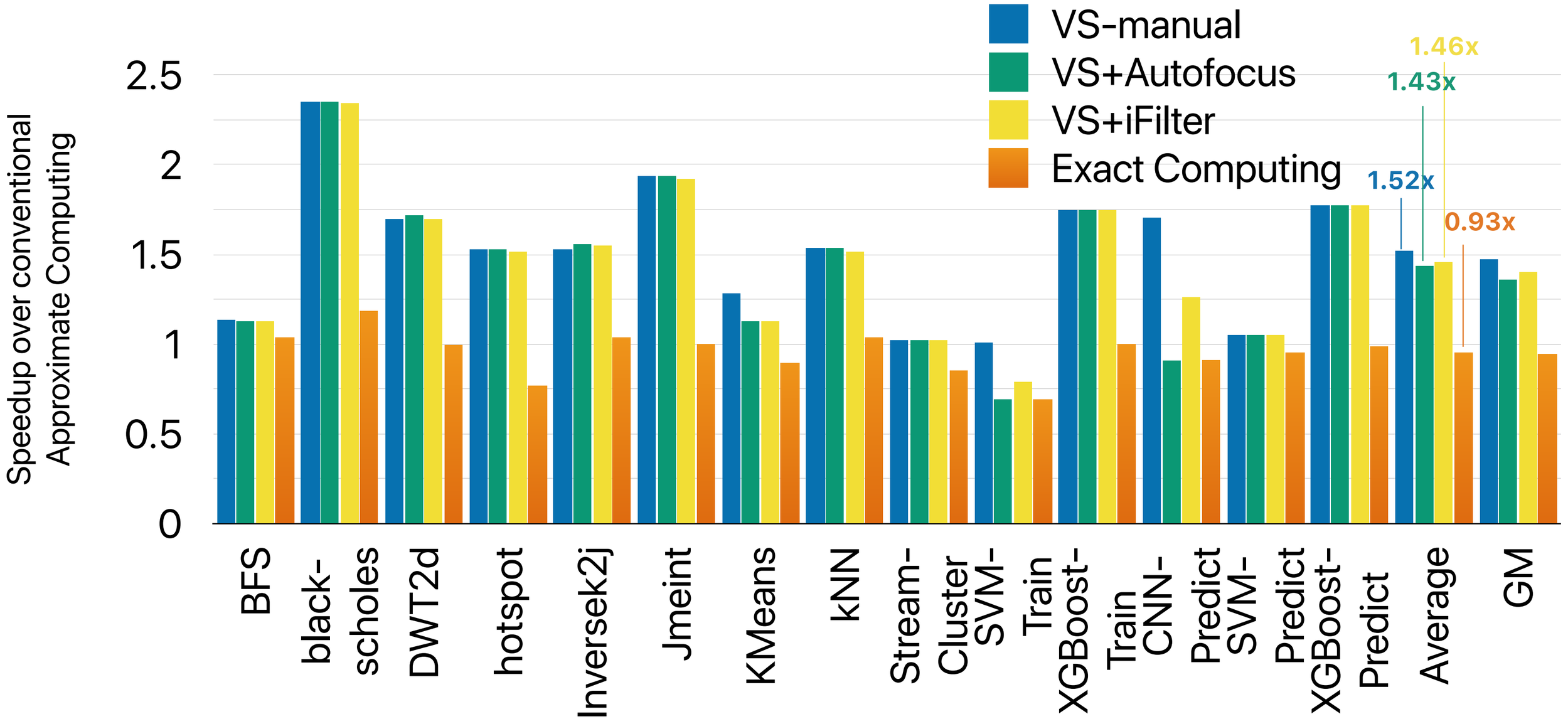
Test

Bear

System Interconnect (e.g., PCIe)

System Interconnect (e.g., PCIe)

Performance of Varifocal Storage (VS)



Announcement

- What to expect in the final presentation
  - Brief recap of the why — 3-5 minutes
  - What has been tried — 5-7 minutes
  - What concrete experimental results have been measured — 5-7 minutes
    - You need to have convincing experimental setup
    - You need to have several result figures (i.e., bar charts)
  - What insights have you learned that would guide future research — 3-5 minutes

Electrical
Computer Science
Engineering

277

つづく