FADU

Storage: The Quiet Engine of AI

Heechul (Fletcher) Chae
FADU, Product Planning

The Shift From Training to Inference

- Generative AI has moved from research to mainstream use, creating billions of inference events daily.
- By 2030, ~75% of AI compute will be inference

— design data centers for high-volume inference traffic.

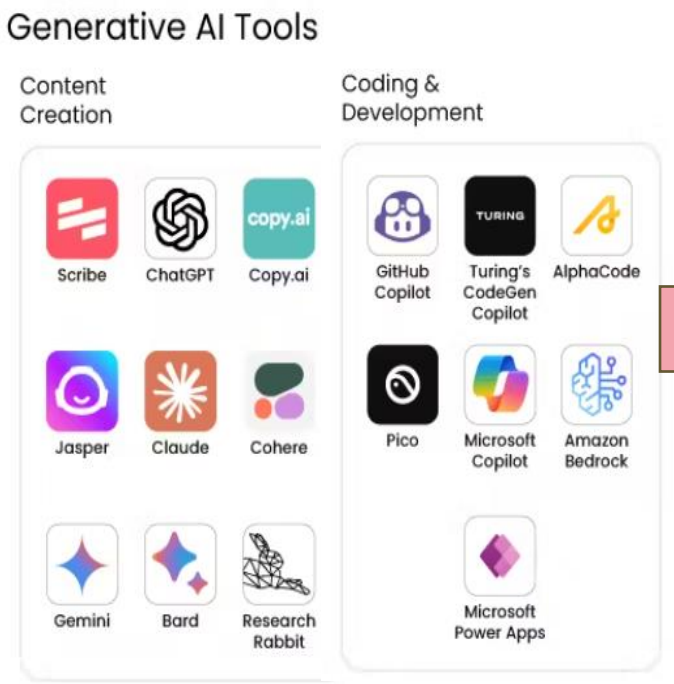
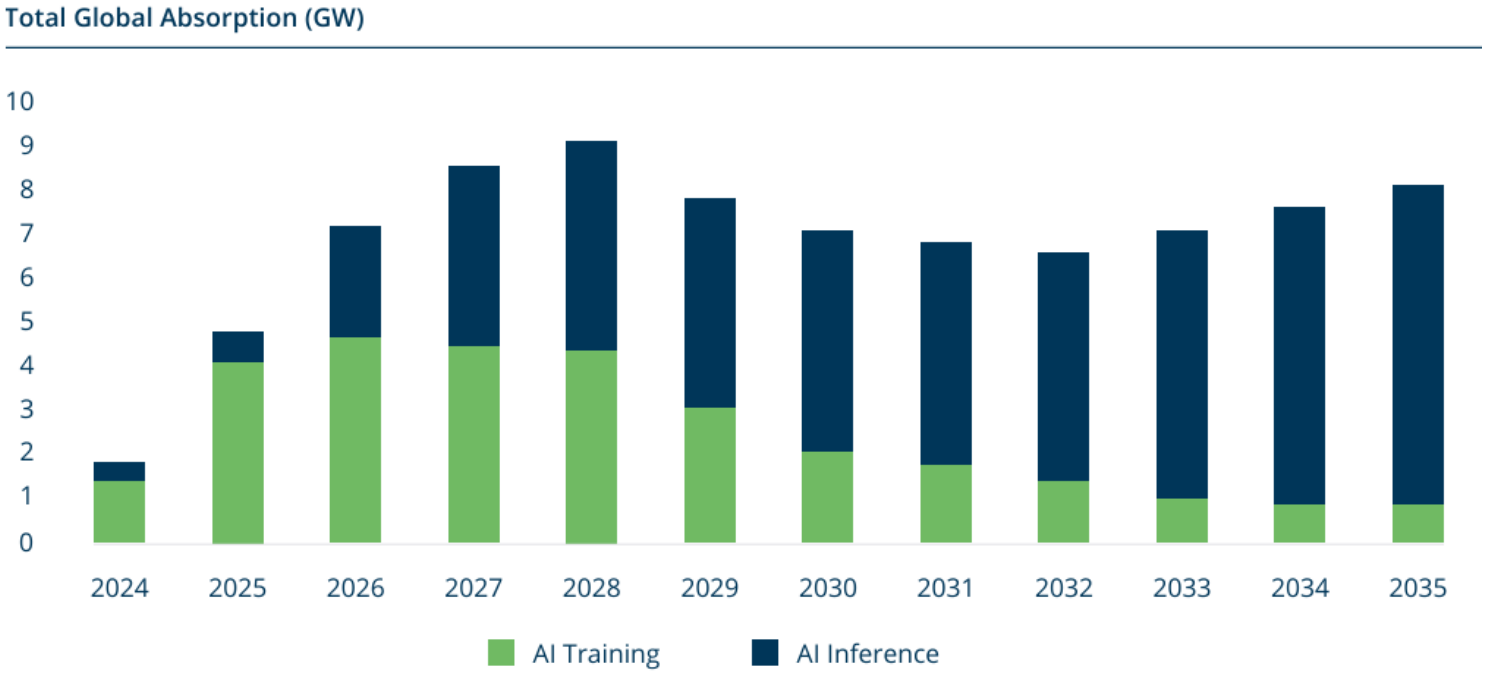


Figure 10: **Approximately 75% of Future AI Compute Demand to Come From Inference by 2030**

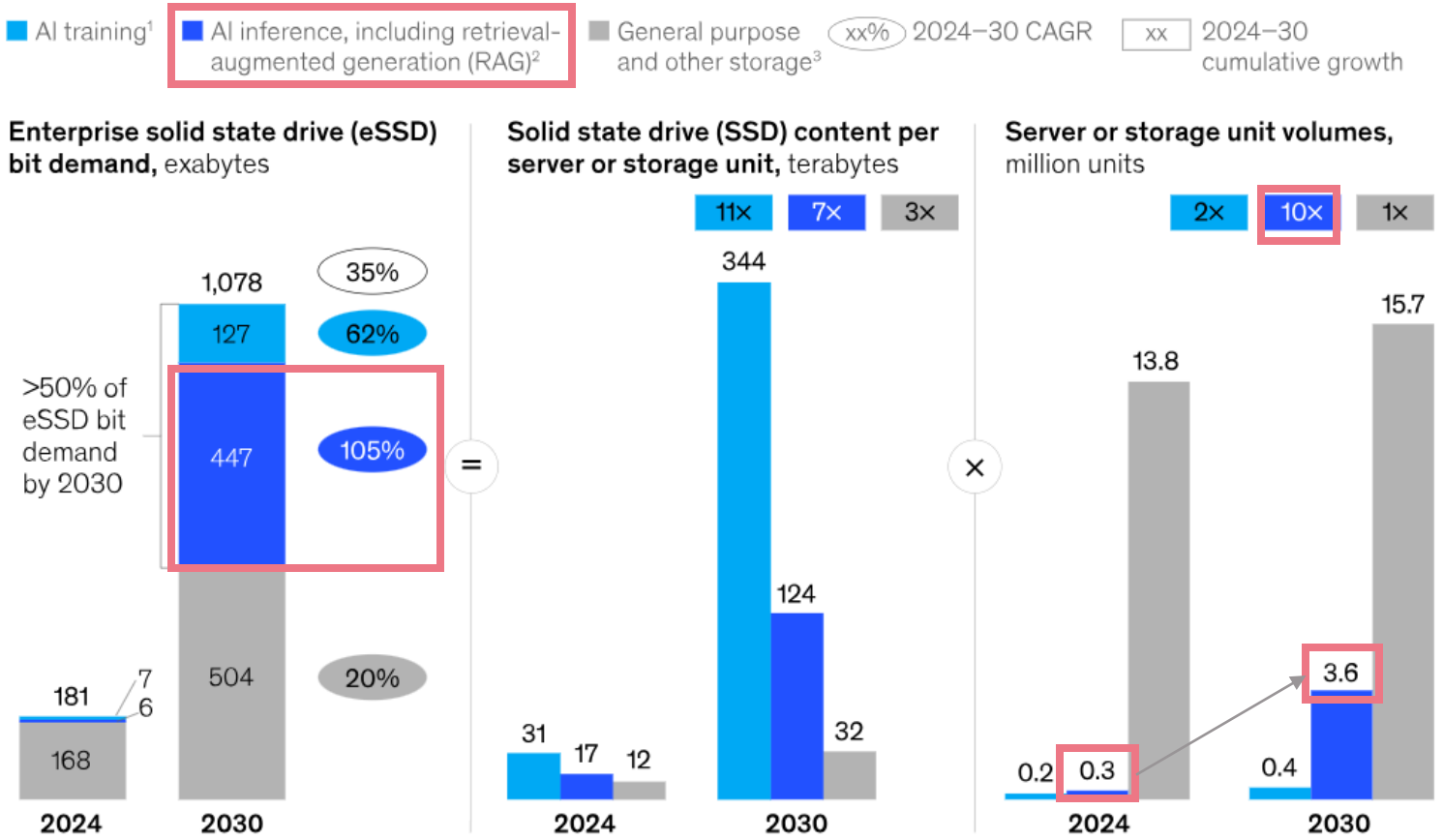


Source: Brookfield internal research.

※Brookfield_Building_the_Backbone_of_AI.pdf

Inference Server (w RAG) is Driving Explosive Data Demand

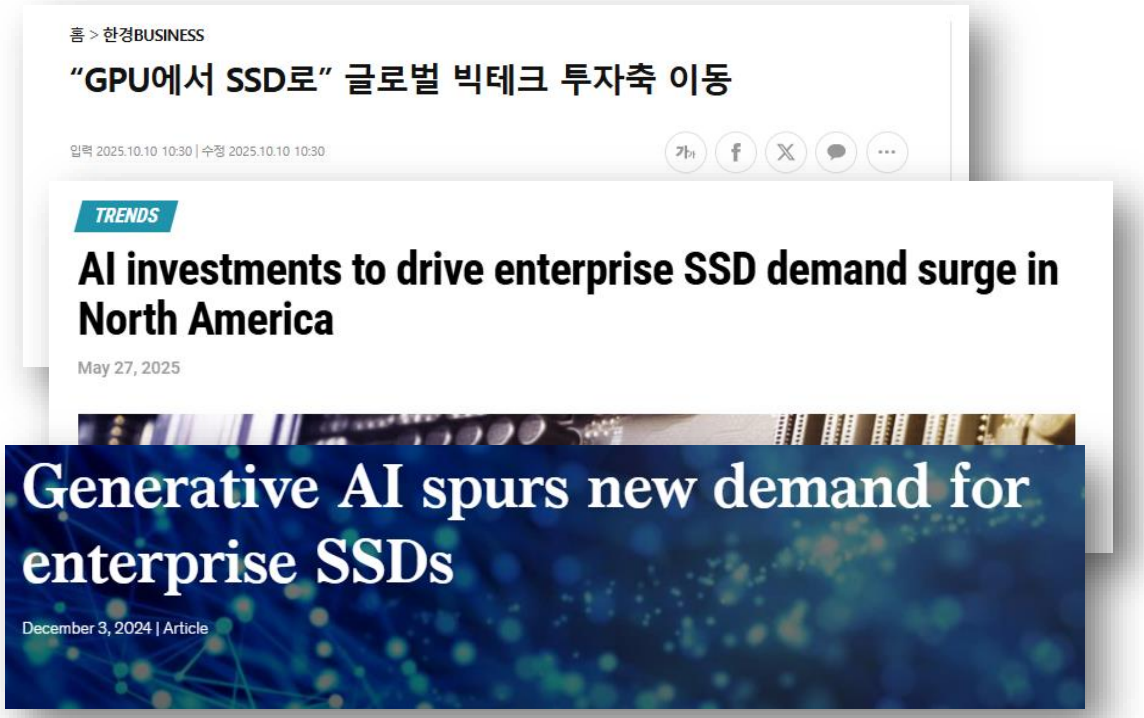
- AI inference infra(GenAI) data demand will grow ~105% CAGR through 2030—over 3.5× training.
- Main driver: RAG—via vector DBs (indexing, replication, low-latency fetch).



※Generative AI spurs new demand for enterprise SSDs | McKinsey

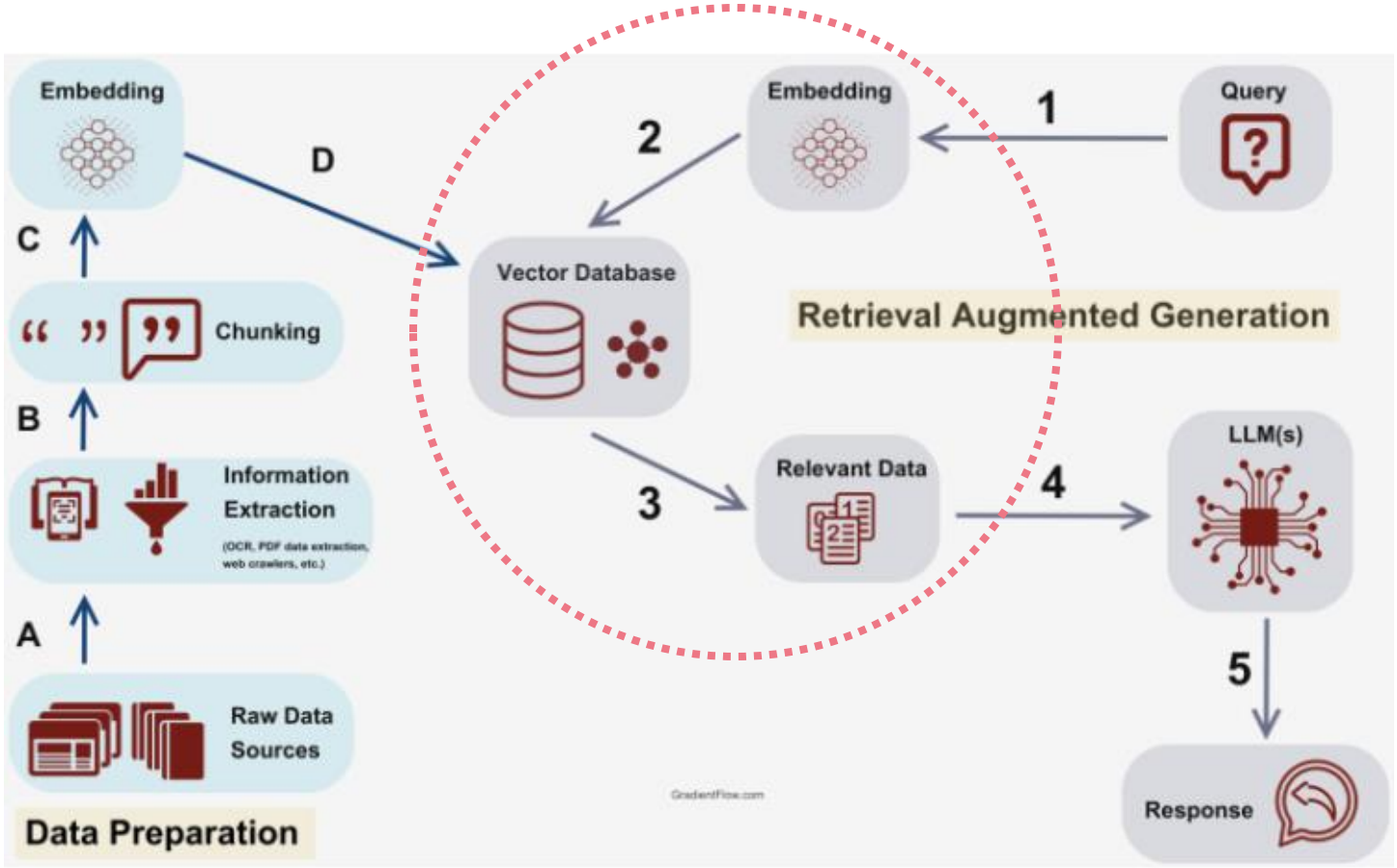
Capital Flows Toward eSSDs in the GenAI Supercycle

- Investment and attention are converging on eSSDs for the inference era.
- It signals a need for specialized eSSDs—not bulk HDDs or commodity drives—like HBM for training.



RAG Makes Storage the Bottleneck in Inference (1)

- Vector search with query embeddings is crucial for **inference QoS**.
- To make vector search faster, **caching key pieces** in HBM and local SSD.



Vector DB Caching Effects

Tail Latency Suppression

Eliminate PCIe/DRAM round-trips

Remove CPU scheduling jitter

→ p99/p999 Stabilized

QPS Increase ↑

GPU massive parallel search

(1000s~10000s queries batched)

→ No memory bottleneck

Recall Cost Mitigation

Increase nprobe (IVF)

efSearch (HNSW)

→ Gentle latency growth

Data Copy Cost ↓

Candidate → Distance → Re-rank

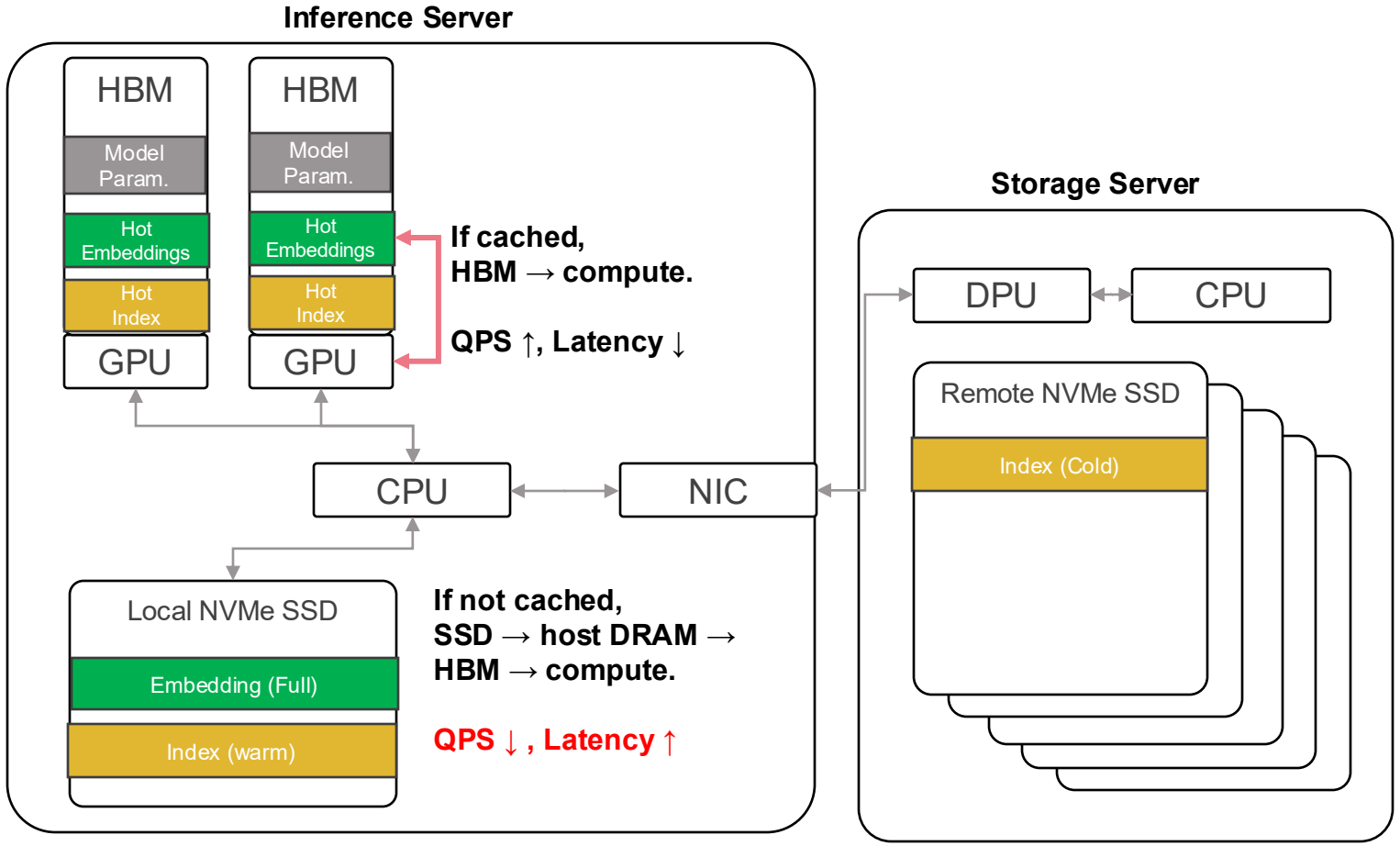
All in HBM pipeline

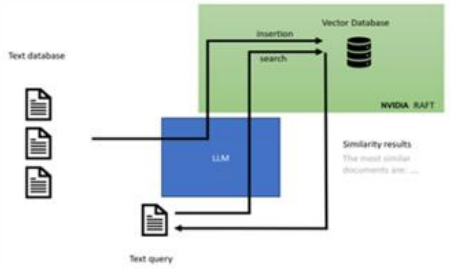
→ Minimize memory bounce

RAG Makes Storage the Bottleneck in Inference (2)

- As the vector DB grows, caching gets complex - causing OOM(Out-of-Memory)
- As I/O shifts from HBM↔GPU to NVMe SSD, SSD BW becomes the bottleneck

— this is why storage is critical for inference.





RAG/VectorDB (>600GB)
ANN algos on embeddings

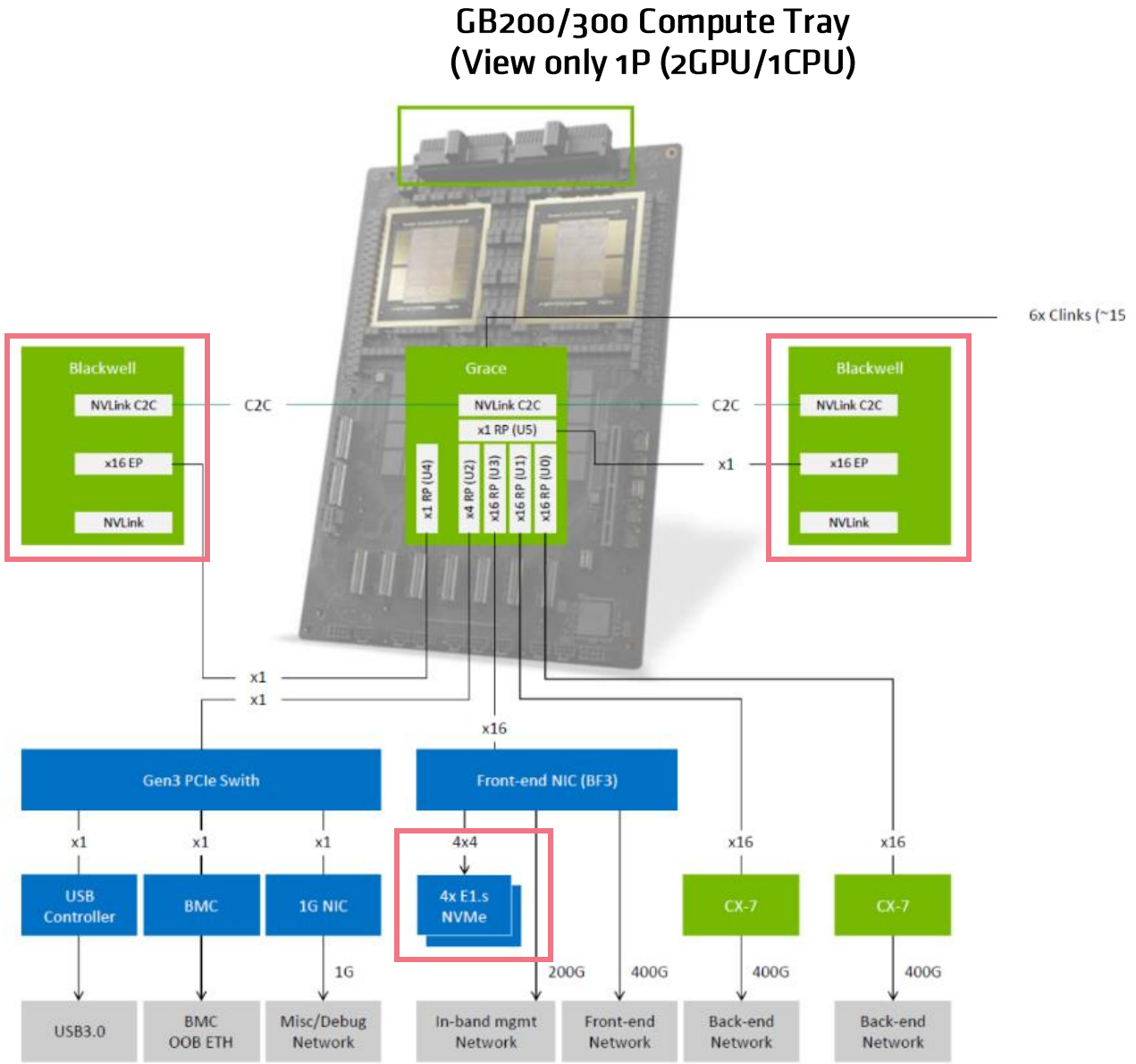
cuVS

Vector Search (up to 40PB)
specialized algos on
embeddings and files

cuVS

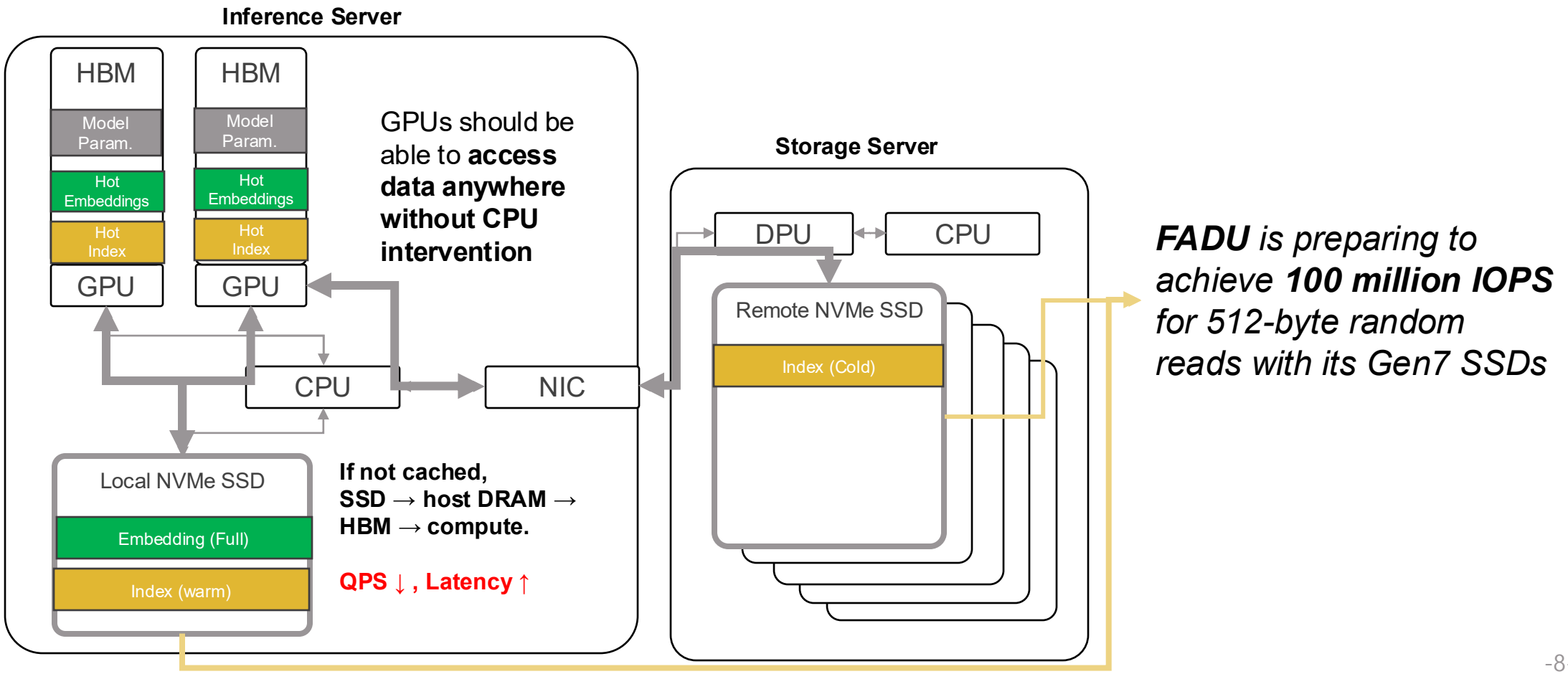
RAG Workloads Storage Must Face

- **Volume**
  - **Provide O(100M) IOPs!**
    - Gen5 PCIe (x16 lane): 50 GB/s
    - IOPS per GPU
      = PCIe bandwidth / xfer xsize
      = 50 GB/s / 512B
      = 100 MIOPs/GPU
  - 예) Blackwell 기준 (2 GPU to 4 NVMe)
    IOPS per GPU = 200 MIOPS / GPU
    = 100 MIOPS / NVMe (Can be required)
- **Size**
  - **Under 512Bytes**
  - Embeddings 512B-4KB,
    graph structure could be 8B-128B
- **Pattern**
  - **Sparse (Random)**

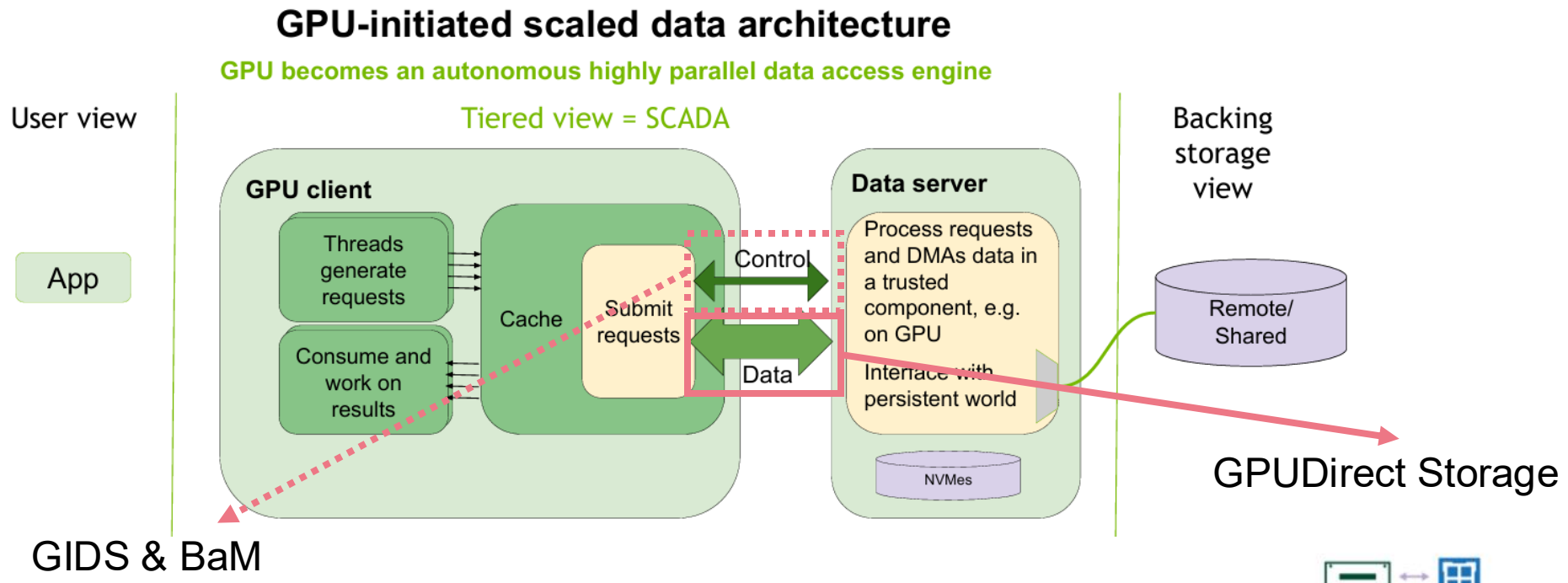


What to solve?

1. (Infrastructure Developers) Optimize the path from GPUs to NVMe SSDs.
2. (Storage Developers) Make SSDs can handle the I/O rate generated by the GPU.



Related work – Optimizing GPU-to-NVMe SSD Paths



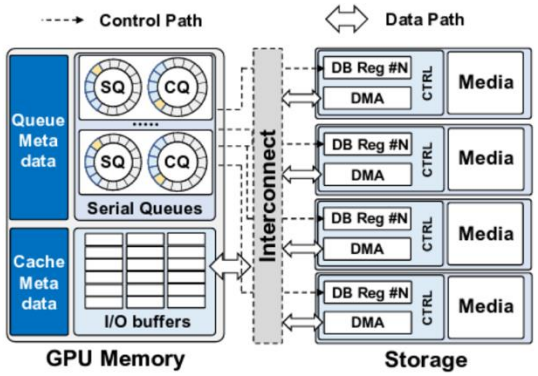
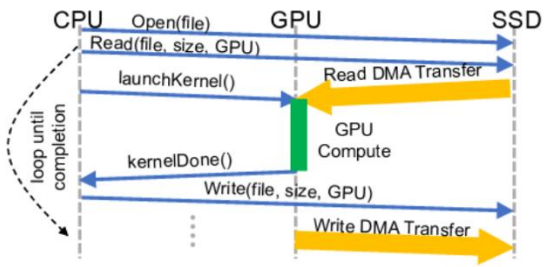
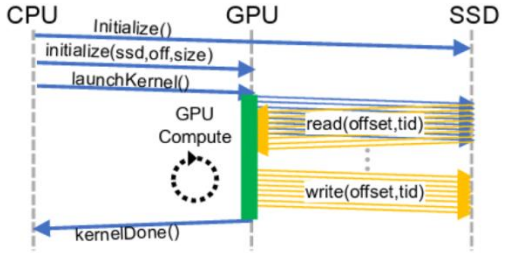


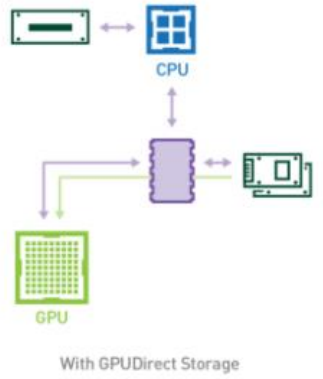
Figure 1: Logical view of BaM design.



(a) CPU-Centric Model.



(b) BaM Model.



Future works

1. **Remove local SSDs from the inference server**
   - Will the network (e.g., the NIC) become the bottleneck?
   - Do we need a new interface to the remote storage server?

2. **A complete implementation of GPU-initiated storage access**
   - GPU-side NVMe command–level access is possible;
   - However, without filesystem-level direct access, aren't we still dependent on the CPU?

Random Read Maximization for AI Storage

Tech to overcome NAND Channel Bottleneck

EuiJin.Kim, FADU Firmware

Contents

Random Read IOPS Trend for AI Storage

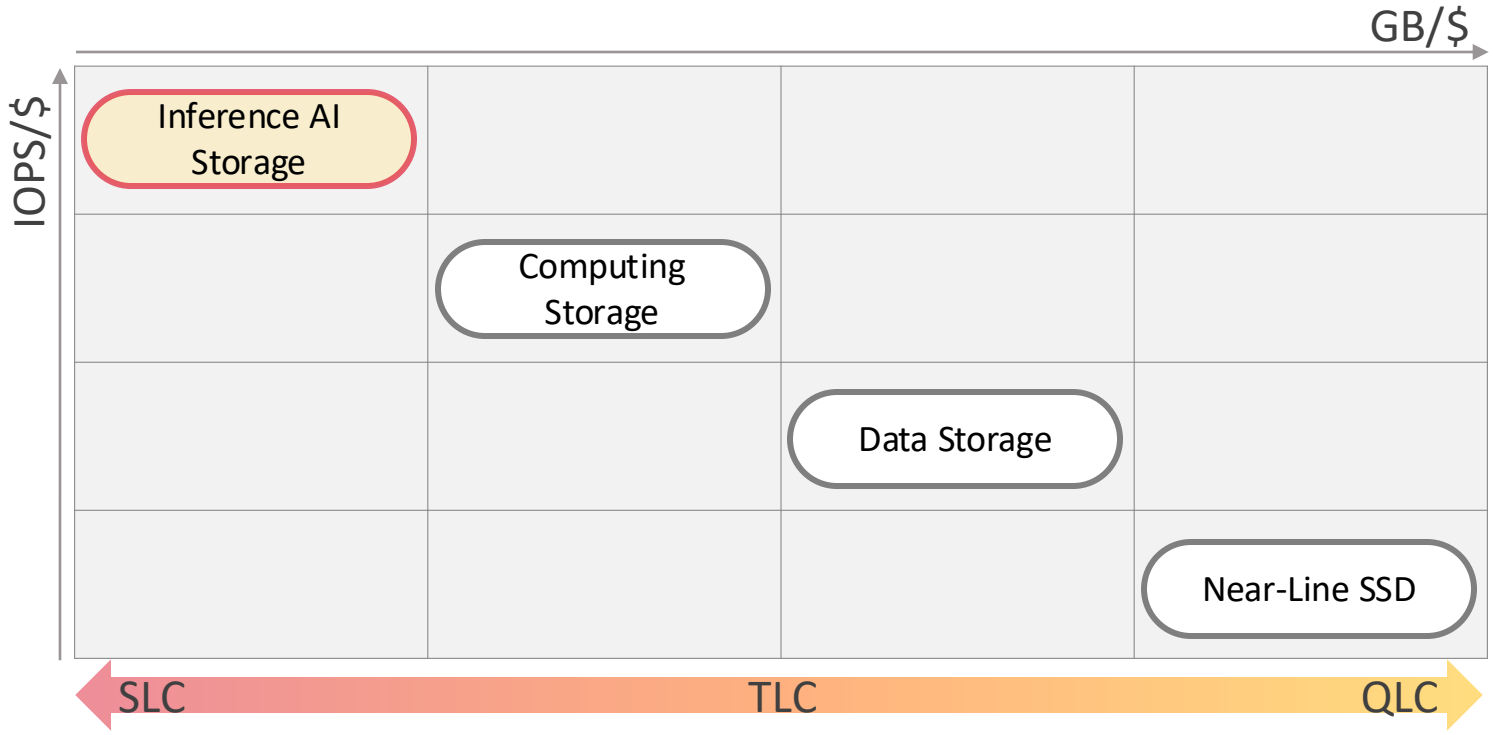
Demand for Inference AI Storage

- IOPS / $
- IOPS / Watt
- Fine-Grained Random Read Intensive
  - embeddings 512B~4KB
  - graph structure could be 8B~128B

SSD Tech Trend for Inference AI Storage

Focused on “Random Read IOPS” & “Watt”

- 512Byte Block Size
  - More IOPS via fixed speed I/F
- SLC NAND
  - Lowest Cell Read Time
  - Lowest Cell Read Power



Target Random Read IOPS for Future Solution

Maximized BW & IOPS on PCIe Gen7

Block size	Ideal Read BW	Ideal Read IOPS
4KB	55.7 GB/s	13.6M IOPS
512Byte	52.7 GB/s	102.9M IOPS

x 7.5

Target Random Read IOPS for Future Solution

Maximized BW & IOPS on PCIe Gen7

Block size	Ideal Read BW	Ideal Read IOPS
4KB	55.7 GB/s	13.6M IOPS
512Byte	52.7 GB/s	102.9M IOPS

x 7.5

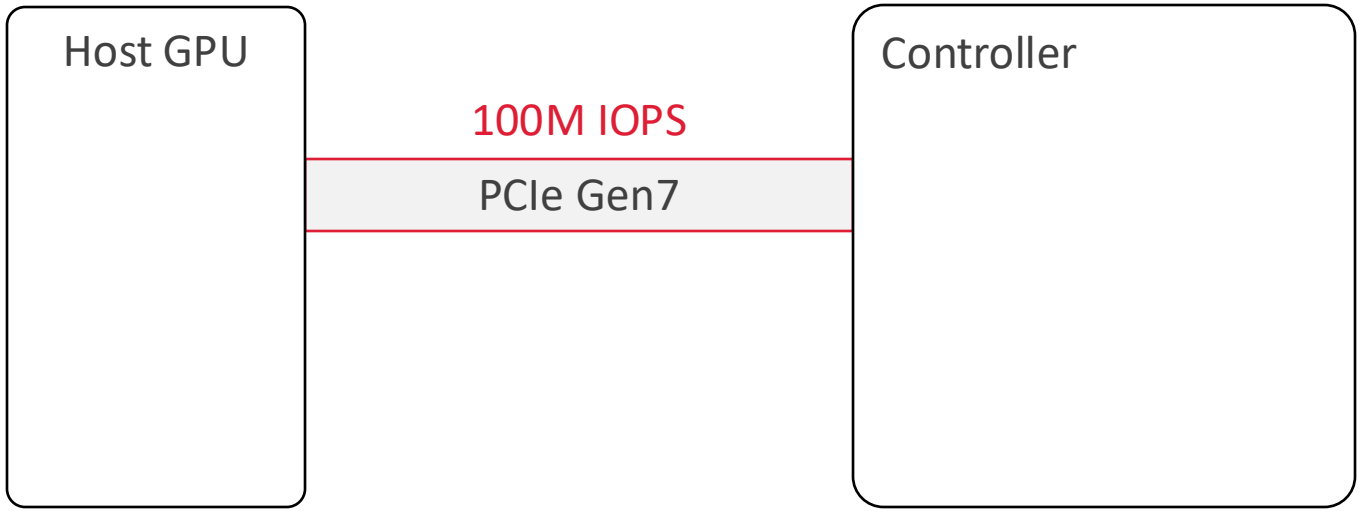
Target IOPS for Future Solution

100M IOPS !!

To achieve 100 Million IOPS of Random Read

System capability to maximize Random Read IOPS

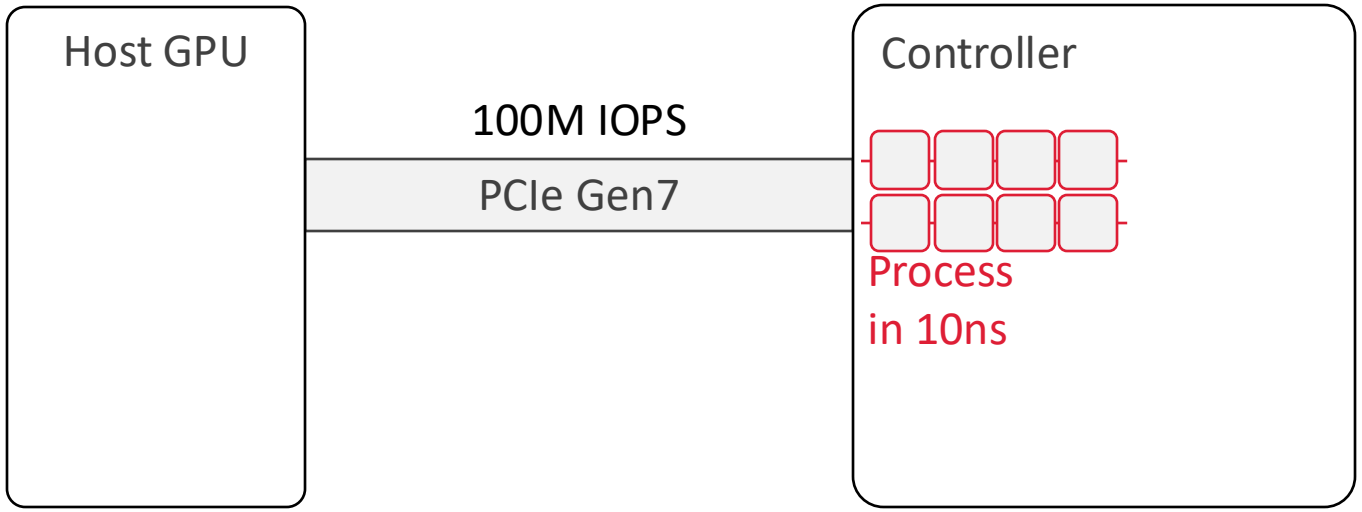
- Fully utilized PCIe Gen7 throughput



To achieve 100 Million IOPS of Random Read

System capability to maximize Random Read IOPS

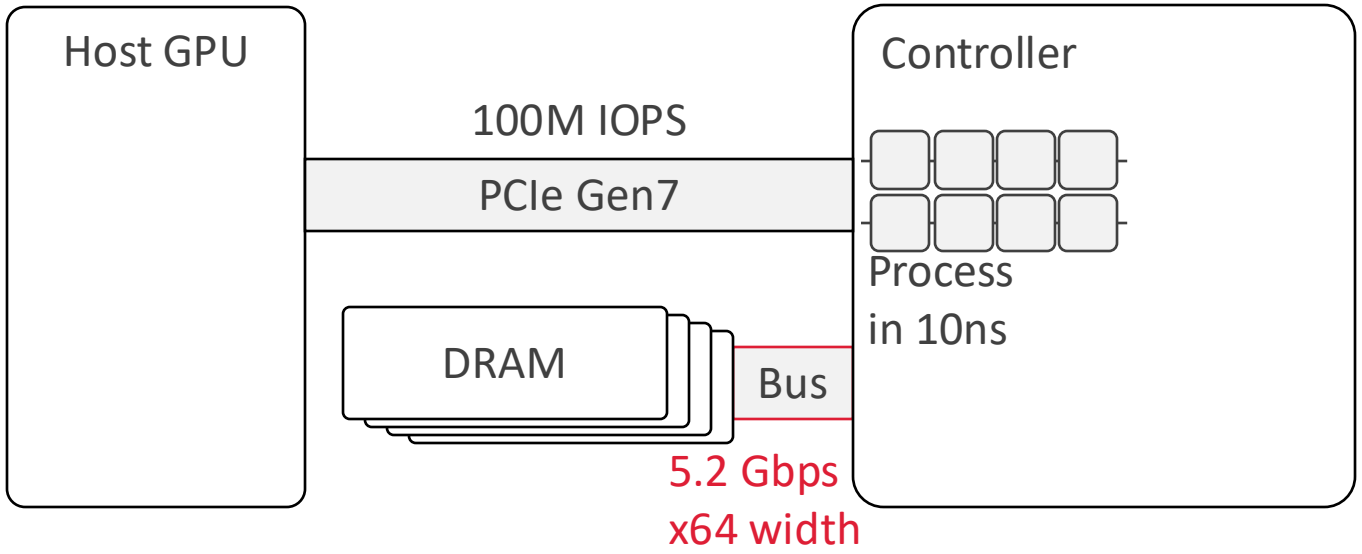
- Fully utilized PCIe Gen7 throughput
- Controller processing capability to handle single I/O in 10ns



To achieve 100 Million IOPS of Random Read

System capability to maximize Random Read IOPS

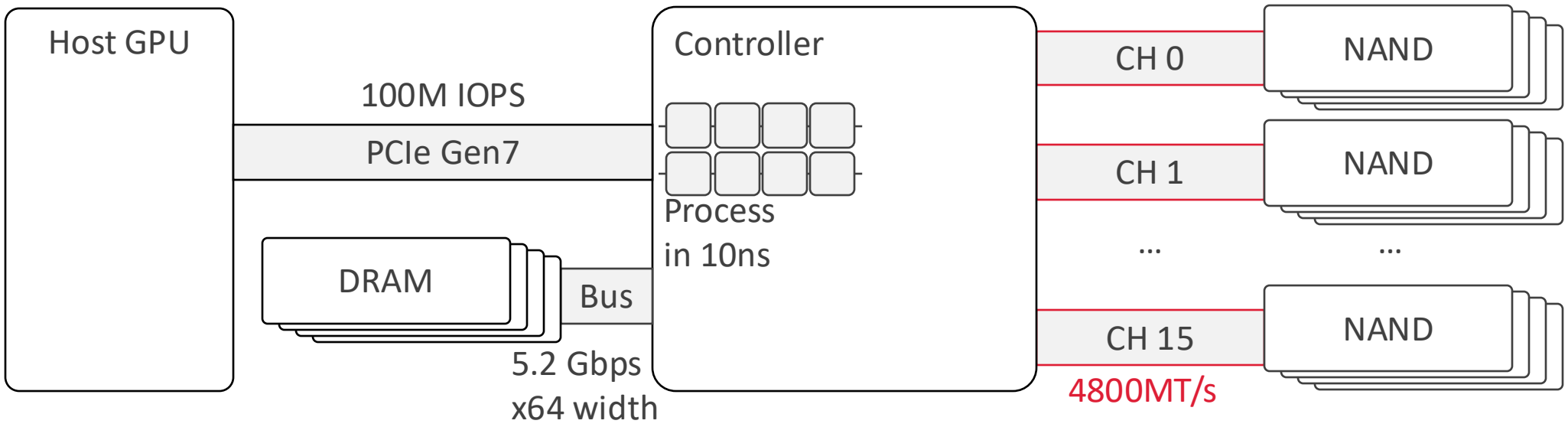
- Fully utilized PCIe Gen7 throughput
- Controller processing capability to handle single I/O in 10ns
- DRAM bus throughput for 100M metadata access



To achieve 100 Million IOPS of Random Read

System capability to maximize Random Read IOPS

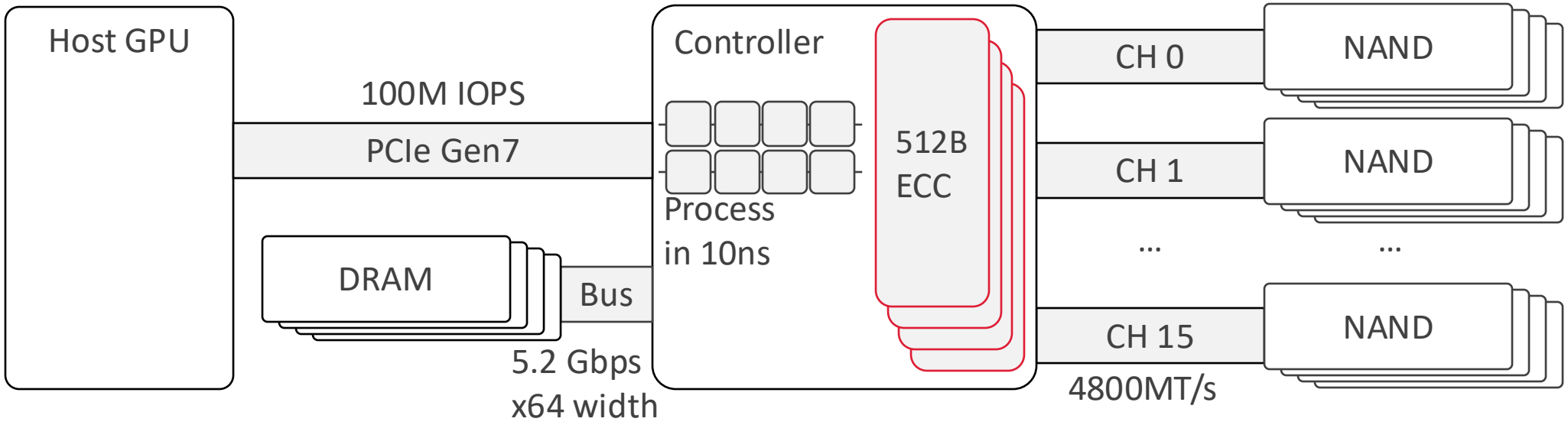
- Fully utilized PCIe Gen7 throughput
- Controller processing capability to handle single I/O in 10ns
- DRAM bus throughput for 100M metadata access
- Practical solution with 16 NAND Channels x 8 Dies which support 4800MT/s



To achieve 100 Million IOPS of Random Read

System capability to maximize Random Read IOPS

- Fully utilized PCIe Gen7 throughput
- Controller processing capability to handle single I/O in 10ns
- DRAM bus throughput for 100M metadata access
- Practical solution with 16 NAND Channels x 8 Dies which support 4800MT/s
- 512Byte ECC Engine – Customized for AI workload

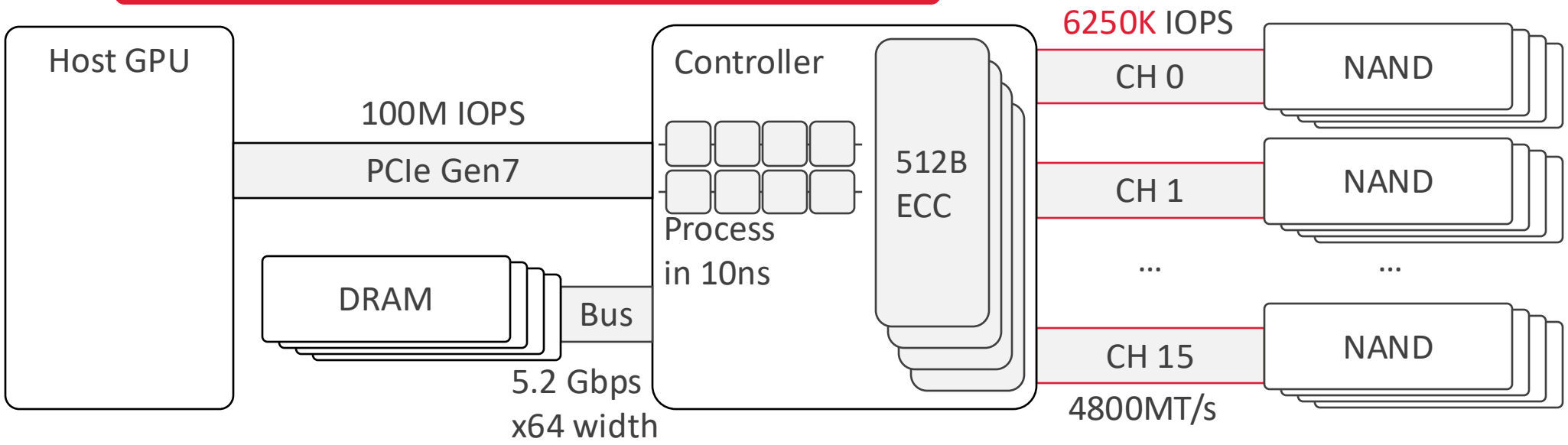


To achieve 100 Million IOPS of Random Read

System capability to maximize Random Read IOPS

- Fully utilized PCIe Gen7 throughput
- Controller processing capability to handle single I/O in 10ns
- DRAM bus throughput for 100M metadata access
- Practical solution with 16 NAND Channels x 8 Dies which support 4800MT/s
- 512Byte ECC Engine – Customized for AI workload
- 6.25M IOPS per Channel *(= 100M IOPS / 16 Channels)*
  - Required effective BW of single channel: 3600MT/s
    - Required channel efficiency: 75%

It is hurdle to get a CH efficiency of 75% or higher!!



To achieve 100 Million IOPS of Random Read

System capability to maximize Random Read IOPS

- Fully utilized PCIe Gen7 throughput
- Controller processing capability to handle single I/O in 10ns
- DRAM bus throughput for 100M metadata access
- Practical solution with 16 NAND Channels x 8 Dies which support 4800MT/s
- 512Byte ECC Engine – Customized for AI workload
- 6.25M IOPS per Channel *(= 100M IOPS / 16 Channels)*
  - Required effective BW of single channel: 3600MT/s
    - Required channel efficiency: 75%

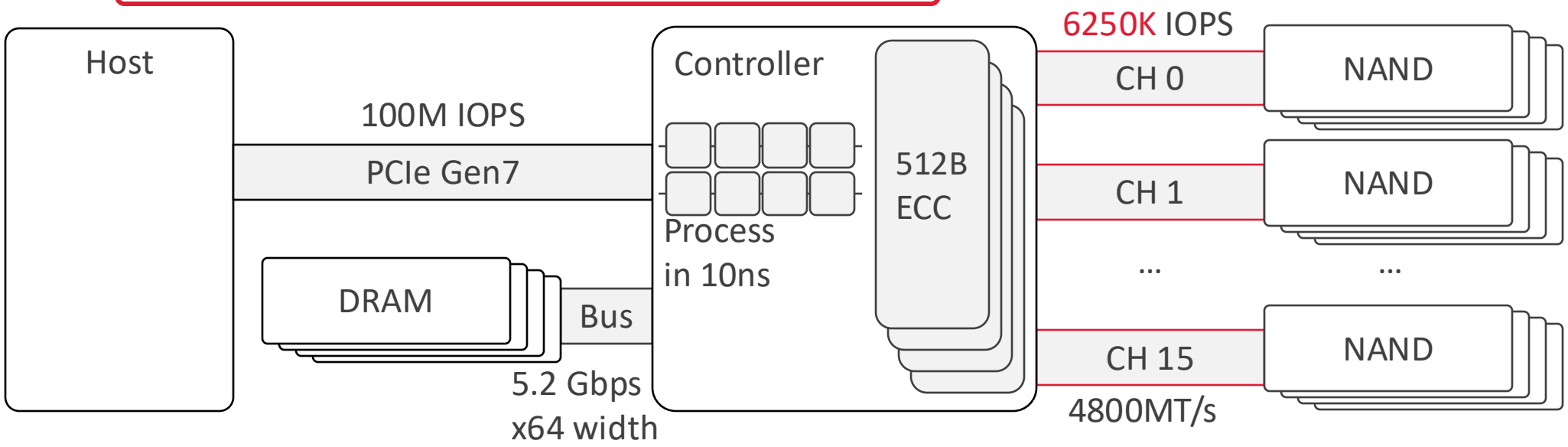
It is hurdle to get a CH efficiency of 75% or higher!!

The easiest way to achieve throughput is to *"increase the # of channels"*

The more channels,

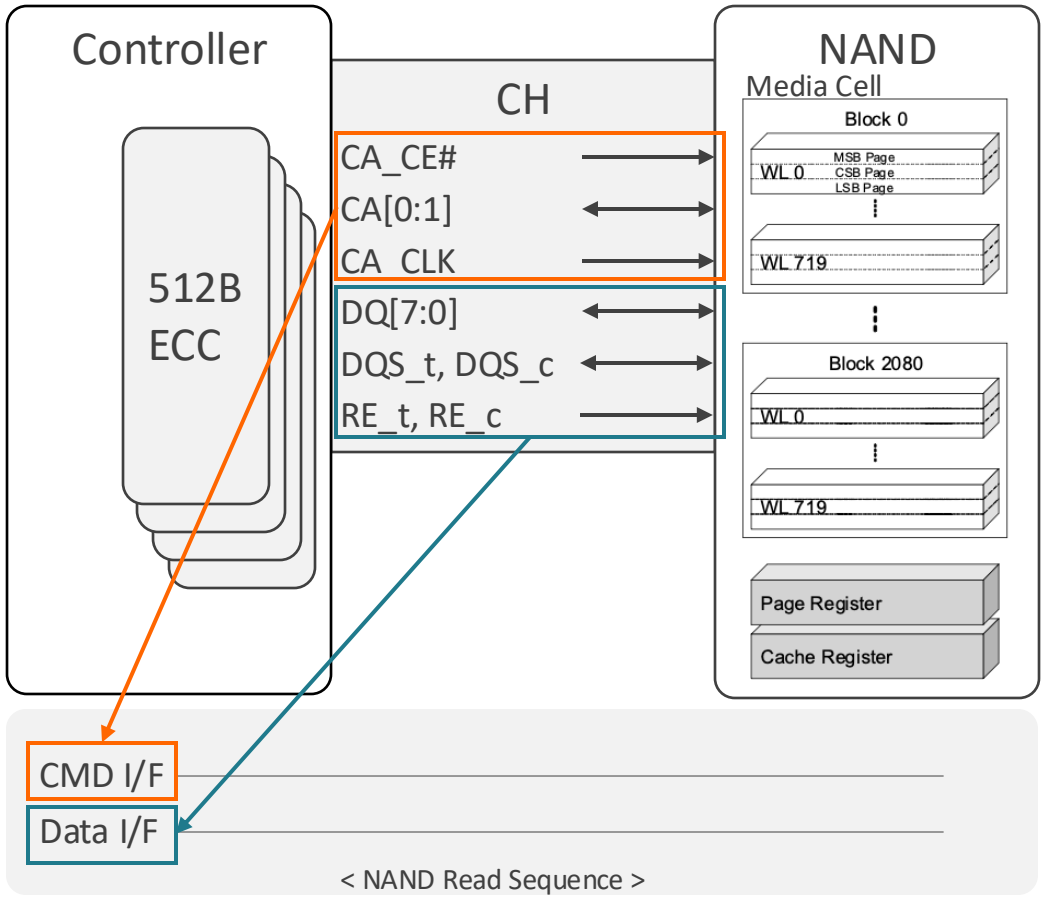
- the higher the throughput can be achieved
- the larger the chip size
- the less installation space

➔ **Find a way to solve the issue, while practically staying within 16 channels.**



Background: NAND Read Operation Flow

How NAND Read works & Interact with controller *(on SCA Protocol)*

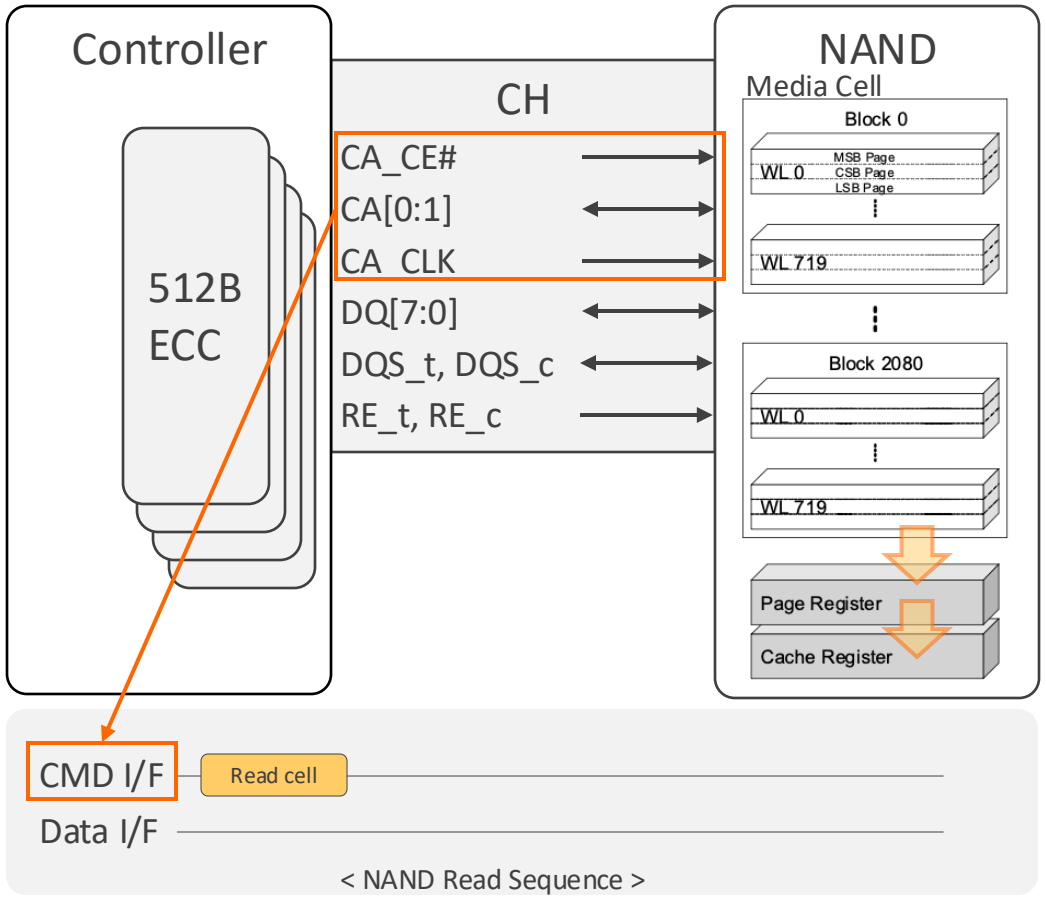


*1. SCA: Separated Command Address

Background: NAND Read Operation Flow

How NAND Read works & Interact with controller *(on SCA Protocol)*

1. Read Cell Command *via CMD I/F*
   - Controller designates the cell address to read
   - NAND status: changed to busy
   - Sensing Read data from Cell
   - Load data to page register
   - Copy data from page register to cache register
   - NAND status: changed to ready after finishing job

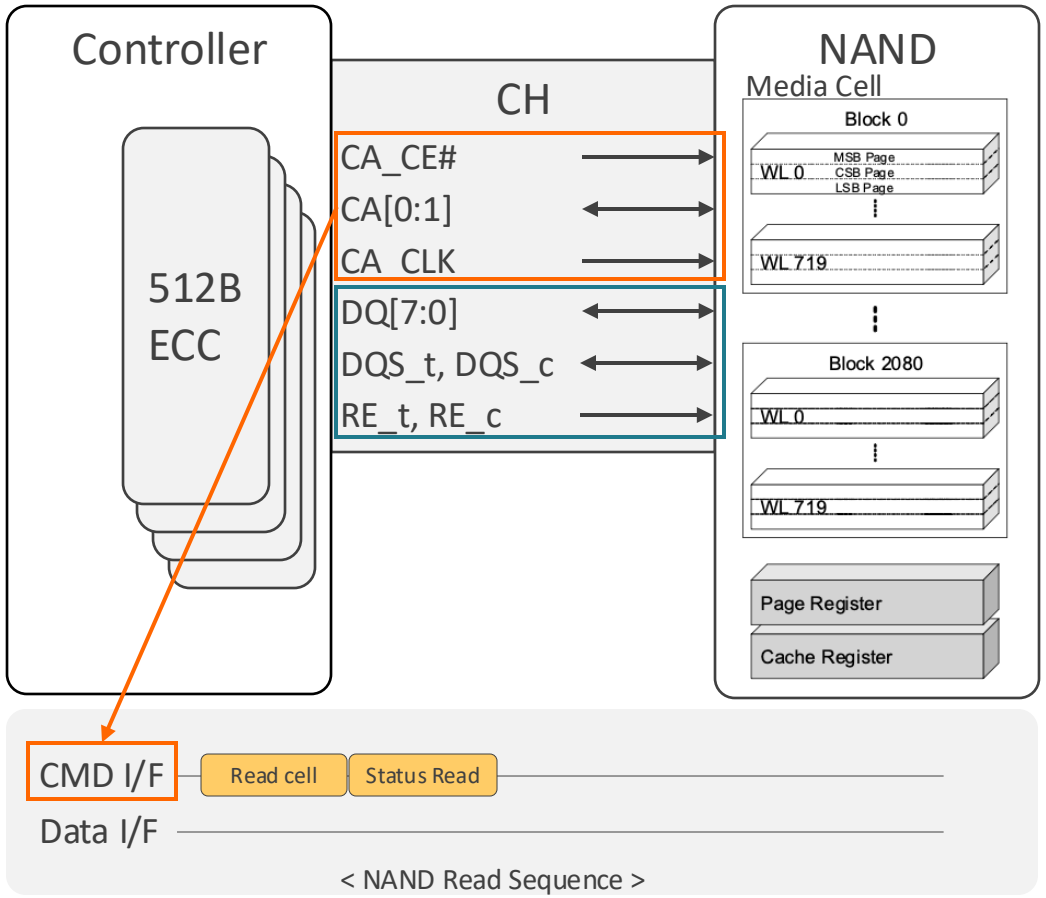


**1. SCA: Separated Command Address*

Background: NAND Read Operation Flow

How NAND Read works & Interact with controller *(on SCA Protocol)*

1. Read Cell Command *via CMD I/F*
   - Controller designates the cell address to read
   - NAND status: changed to busy
   - Sensing Read data from Cell
   - Load data to page register
   - Copy data from page register to cache register
   - NAND status: changed to ready after finishing job
2. Status Read Command *via CMD I/F*
   - Controller check NAND is ready to read-out data
   - NAND returns current status

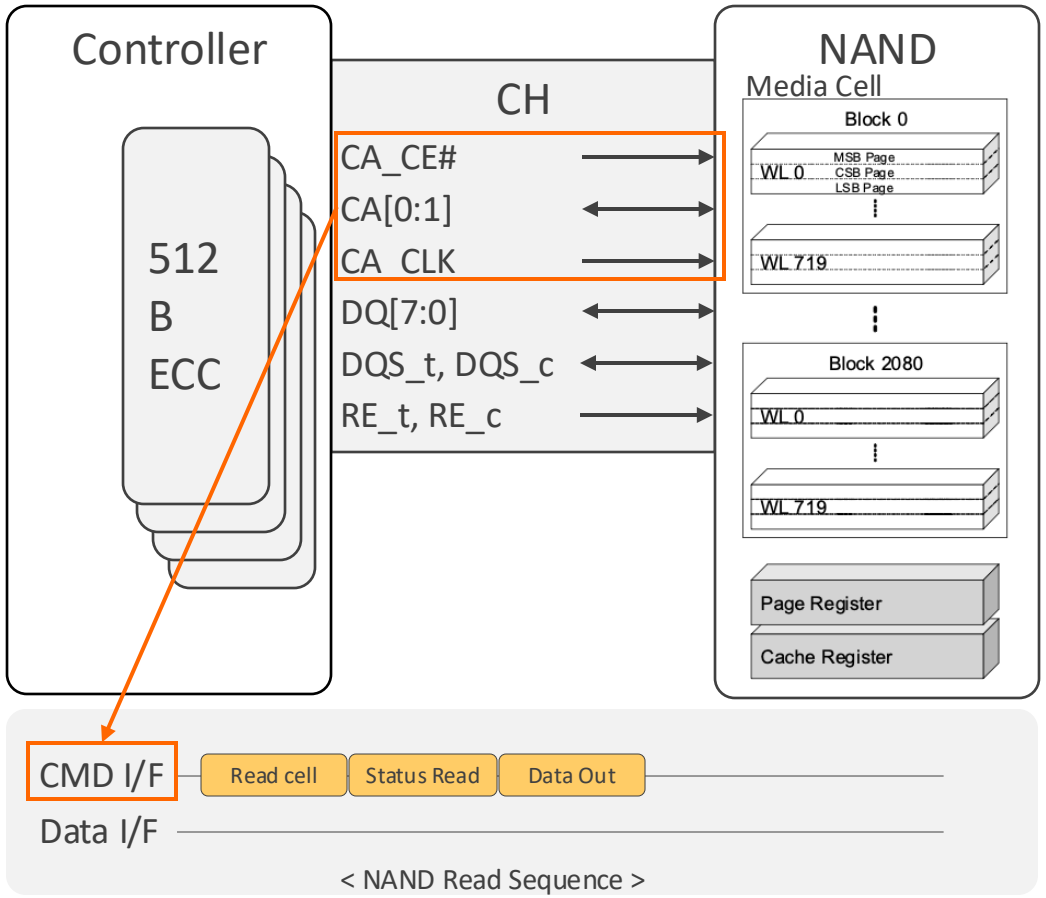


**1. SCA: Separated Command Address*

Background: NAND Read Operation Flow

How NAND Read works & Interact with controller *(on SCA Protocol)*

1. Read Cell Command *via CMD I/F*
   - Controller designates the cell address to read
   - NAND status: changed to busy
   - Sensing Read data from Cell
   - Load data to page register
   - Copy data from page register to cache register
   - NAND status: changed to ready after finishing job
2. Status Read Command *via CMD I/F*
   - Controller check NAND is ready to read-out data
   - NAND returns current status
3. Random data-out command *via CMD I/F*
   - The controller notifies the NAND to prepare to toggle out data.

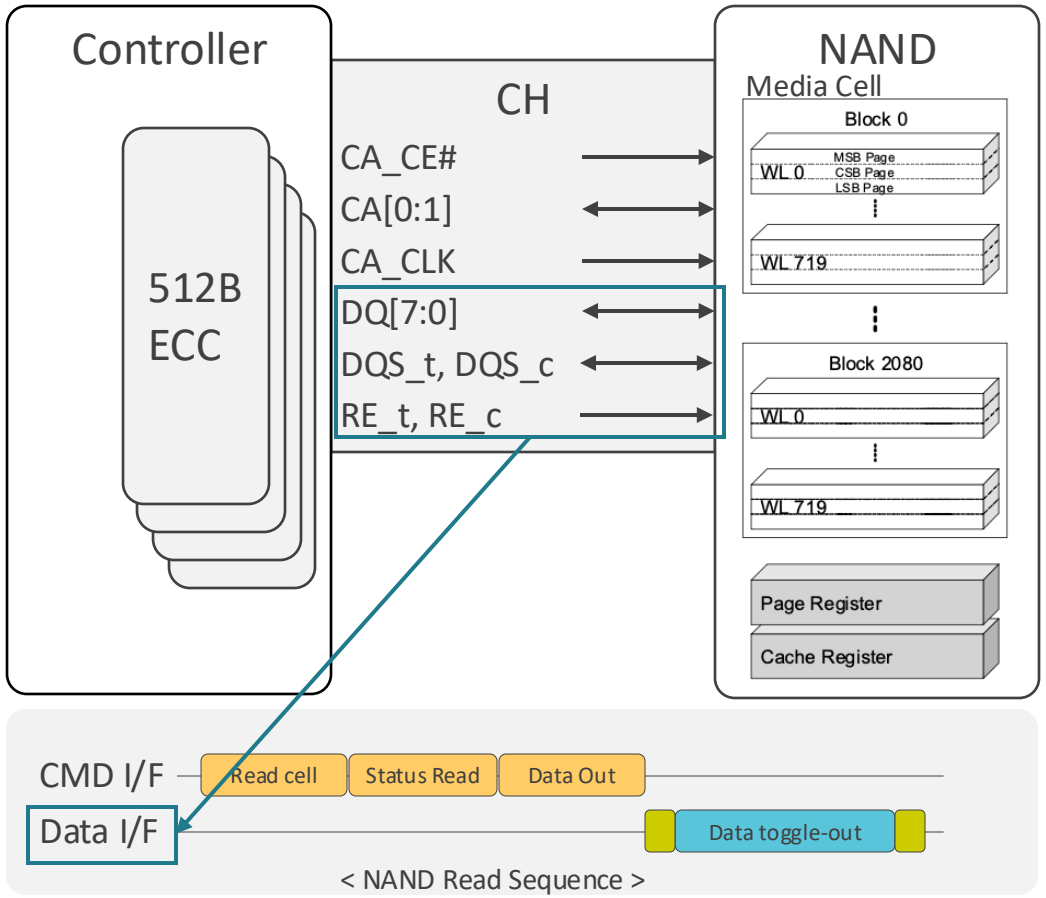


**1. SCA: Separated Command Address*

Background: NAND Read Operation Flow

How NAND Read works & Interact with controller *(on SCA Protocol)*

1. Read Cell Command *via CMD I/F*
   - Controller designates the cell address to read
   - NAND status: changed to busy
   - Sensing Read data from Cell
   - Load data to page register
   - Copy data from page register to cache register
   - NAND status: changed to ready after finishing job
2. Status Read Command *via CMD I/F*
   - Controller check NAND is ready to read-out data
   - NAND returns current status
3. Random data-out command *via CMD I/F*
   - The controller notifies the NAND to prepare to toggle out data.
4. Data Toggle Out *via Data I/F*
   - Controller toggles RE to NAND
   - NAND toggle-out data in cache register via NVDDR I/F (DQ[7:0])

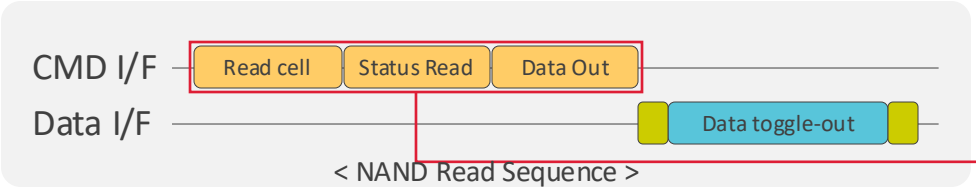


**1. SCA: Separated Command Address*

Why the NAND Channel is the Bottleneck

Factors that reduce channel efficiency

1. Slow improvement of the CMD I/F speed with legacy manner



NAND Interface Spec		Data I/F Speed	Data I/F			CMD I/F	CH Efficiency	
			Data Transfer Time		Pre-amble / Post-amble	CMD&ADDR Transfer		
			4KB	512B			4KB	512B
ONFi 2.0	NV-DDR	200 MT/s	23040 ns	2880 ns	90ns	1us	95%	73%
ONFi 3.0	NV-DDR2	400 MT/s	11520 ns	1440 ns	90ns	1us	91%	57%
ONFi 4.0	NV-DDR3	1200 MT/s	3840 us	480 us	90ns	1us	78%	31%
ONFi 5.0	NV-LPDDR4	3600 MT/s	1280 us	160 us	90ns	500ns	93%	32%
ONFi 5.0	Enhanced	4800 MT/s	960 ns	120 ns	90ns	500ns	91%	24%

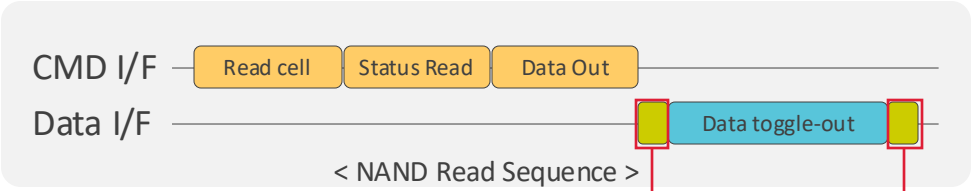
SCA*1

*1. SCA: Separated Command Address

Why the NAND Channel is the Bottleneck

Factors that reduce channel efficiency

1. Slow improvement of the CMD I/F speed with legacy manner
2. Fixed overhead for data interfaces reliability
   - Pre-amble, Post-amble



NAND Interface Spec		Data I/F Speed	Data I/F			CMD I/F	CH Efficiency	
			Data Transfer Time		Pre-amble / Post-amble	CMD&ADDR Transfer		
			4KB	512B			4KB	512B
ONFi 2.0	NV-DDR	200 MT/s	23040 ns	2880 ns	90ns	1us	95%	73%
ONFi 3.0	NV-DDR2	400 MT/s	11520 ns	1440 ns	90ns	1us	91%	57%
ONFi 4.0	NV-DDR3	1200 MT/s	3840 us	480 us	90ns	1us	78%	31%
ONFi 5.0	NV-LPDDR4	3600 MT/s	1280 us	160 us	90ns	500ns	93%	32%
ONFi 5.0	Enhanced	4800 MT/s	960 ns	120 ns	90ns	500ns	91%	24%

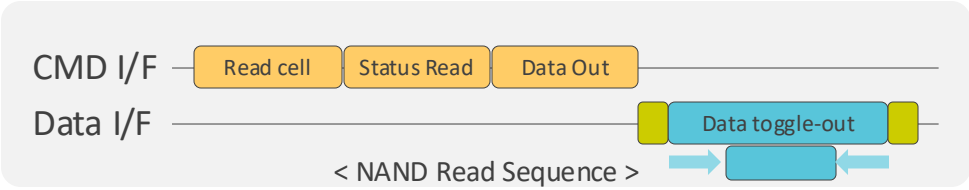
SCA*1

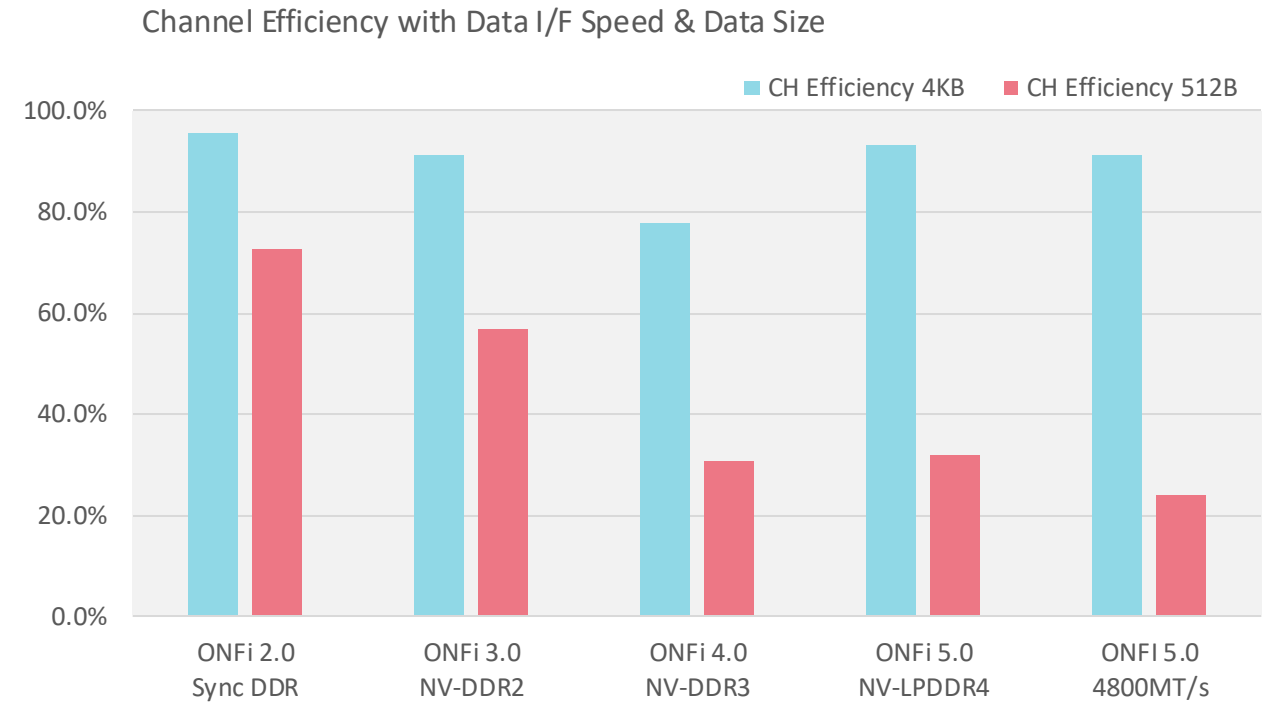
*1. SCA: Separated Command Address

Why the NAND Channel is the Bottleneck

Factors that reduce channel efficiency

1. Slow improvement of the CMD I/F speed with legacy manner
2. Fixed overhead for data interfaces reliability
   - Pre-amble, Post-amble
3. The smaller the data size, the relatively higher the fixed cost





NAND Interface Spec		Data I/F Speed	Data I/F		CMD I/F		CH Efficiency	
			Data Transfer Time		Pre-amble / Post-amble	CMD&ADDR Transfer		
			4KB	512B			4KB	512B
ONFi 2.0	NV-DDR	200 MT/s	23040 ns	2880 ns	90ns	1us	95%	73%
ONFi 3.0	NV-DDR2	400 MT/s	11520 ns	1440 ns	90ns	1us	91%	57%
ONFi 4.0	NV-DDR3	1200 MT/s	3840 us	480 us	90ns	1us	78%	31%
ONFi 5.0	NV-LPDDR4	3600 MT/s	1280 us	160 us	90ns	500ns	93%	32%
ONFi 5.0	Enhanced	4800 MT/s	960 ns	120 ns	90ns	500ns	91%	24%

Fixed Cost

SCA*1

*1. SCA: Separated Command Address

Points to optimize for 100M IOPS

Goal to optimize

- Channel Efficiency of 75% or higher
- 6250K IOPS per Channel (= 160 ns per single read command)
  - CMD I/F occupancy time < 160 ns
  - Data I/F occupancy time < 160 ns

2 Blockers

1. CMD I/F time: 500ns > 160 ns
2. Data I/F time: 210ns > 160 ns

NAND Interface Spec		Data I/F Speed	Data I/F			CMD I/F	CH Efficiency	
			Data Transfer Time		Pre-amble / Post-amble	CMD&ADDR Transfer		
			4KB	512B			4KB	512B
ONFi 5.0	Enhanced	4800 MT/s	960 ns	120 ns	90ns	500ns	91%	24%

NAND Channel Optimization – Data I/F

Limitation

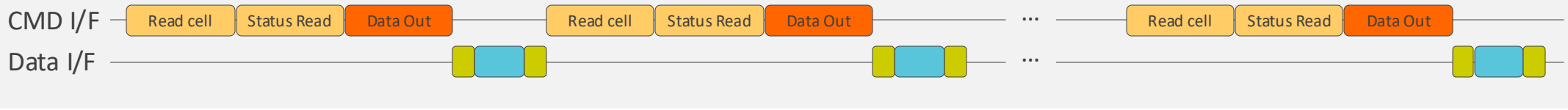
Pre-amble & Post-amble time is fixed

Approach to solve problem

Reducing the fixed overhead ratio

Data size	① Data-out Time	② Pre-amble +Post-amble time	Overhead Ratio (= ② / (① + ②))	Data I/F efficienty
512B	119ns	90ns	43%	57%
4K	955ns	90ns	8.6%	91.4%
16K	3820ns	90ns	2.3%	97.7%

AS-IS



TO-BE

NAND Channel Optimization – Data I/F

Limitation

Pre-amble & Post-amble time is fixed

Approach to solve problem

Reducing the fixed overhead ratio

Data size	① Data-out Time	② Pre-amble +Post-amble time	Overhead Ratio (= ② / (① + ②))	Data I/F efficiency
512B	119ns	90ns	43%	57%
4K	955ns	90ns	8.6%	91.4%
16K	3820ns	90ns	2.3%	97.7%

Improvement Effect

Data I/F

- Overhead sharing

AS-IS

TO-BE

NAND Channel Optimization – Data I/F

Limitation

Pre-amble & Post-amble time is fixed

Approach to solve problem

Reducing the fixed overhead ratio

Data size	① Data-out Time	② Pre-amble +Post-amble time	Overhead Ratio (= ② / (① + ②))	Data I/F efficiency
512B	119ns	90ns	43%	57%
4K	955ns	90ns	8.6%	91.4%
16K	3820ns	90ns	2.3%	97.7%

Improvement Effect

Data I/F

- Overhead sharing

CMD I/F

- Reduction in the number of Data Out commands

AS-IS

TO-BE

NAND Channel Optimization – Data I/F

Limitation

Pre-amble & Post-amble time is fixed

Approach to solve problem

Reducing the fixed overhead ratio

Data size	① Data-out Time	② Pre-amble +Post-amble time	Overhead Ratio (= ② / (① + ②))	Data I/F efficiency
512B	119ns	90ns	43%	57%
4K	955ns	90ns	8.6%	91.4%
16K	3820ns	90ns	2.3%	97.7%

Improvement Effect

Data I/F

- Overhead sharing

CMD I/F

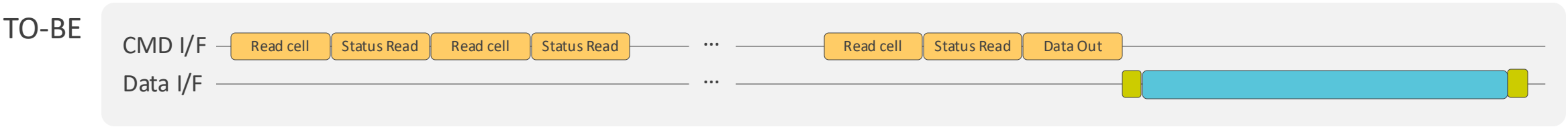
- Reduction in the number of Data Out commands

AS-IS

TO-BE

Current NAND does not support this behavior

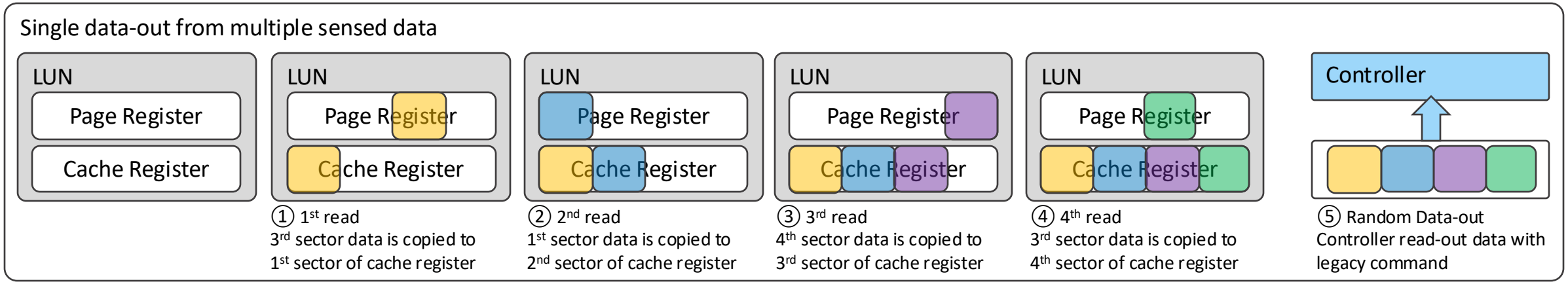
NAND Channel Optimization – Data I/F



Required New NAND Command To-Be:

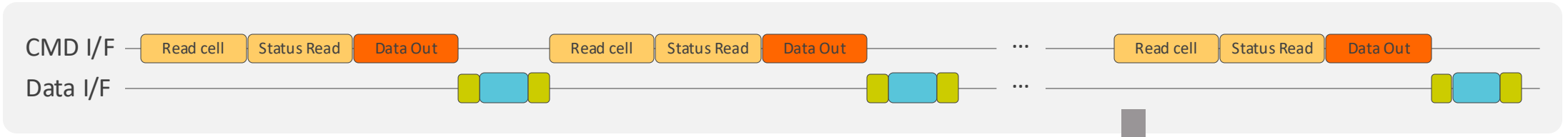
Read & Flexible load to cache register

- After cell sensing, data is copied to cache register with source & destination offset
- Cache register is utilized like data FIFO
- Improve the constraint requiring loading to a fixed column address between the page register and cache register



NAND Channel Optimization – CMD I/F 1st

Origin



Data I/F Opt.

Is it still needed?

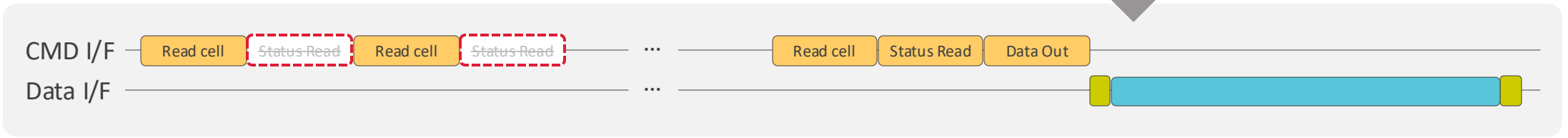
Expansion of Idea

- Why status read is always required?
  - Command input is not allowed during busy ← Requires periodic status read
- There is no need to immediately perform data-out after reading the cell.
  - The tasks that can be performed within the NAND if tasks are queued.
  - Only last status read is required before ‘Data-out’

➔ NAND read command queueing

- Eliminate the status read command overhead

TO-BE



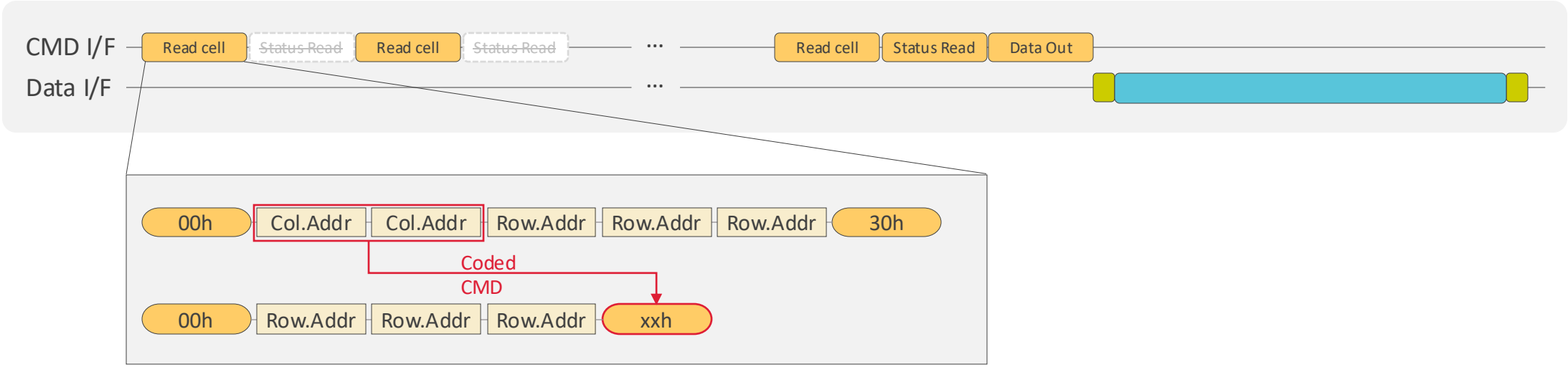
NAND Channel Optimization – CMD I/F 2nd

Idea to reduce CMD I/F overhead

Coded CMD for Limited Column Address Usage

- The usage case for NAND column addresses is aligned to ECC units.
- 2Byte information is not required.
- Legacy confirm command(30h) is replaced by coded command include column address info(ex. 30h~4Fh)

➔ Reduction of CMD I/F occupancy time to send read command



Conclusion

- The customized SSD market is rapidly growing.
  - To meet the requirements of the custom SSD market, improvements are needed at each layer.
- SSD solution & NAND vendors must jointly enhance the customized solutions.
  - FADU is engaging to advance the future solutions with multiple NAND suppliers .
- The NAND Legacy CMD I/F is somewhat unsuitable for AI SSD.
  - The fixed overhead ratio will continue to increase.
- Long-term, the NAND Command Interface requires innovative improvements.

The SSD Expert