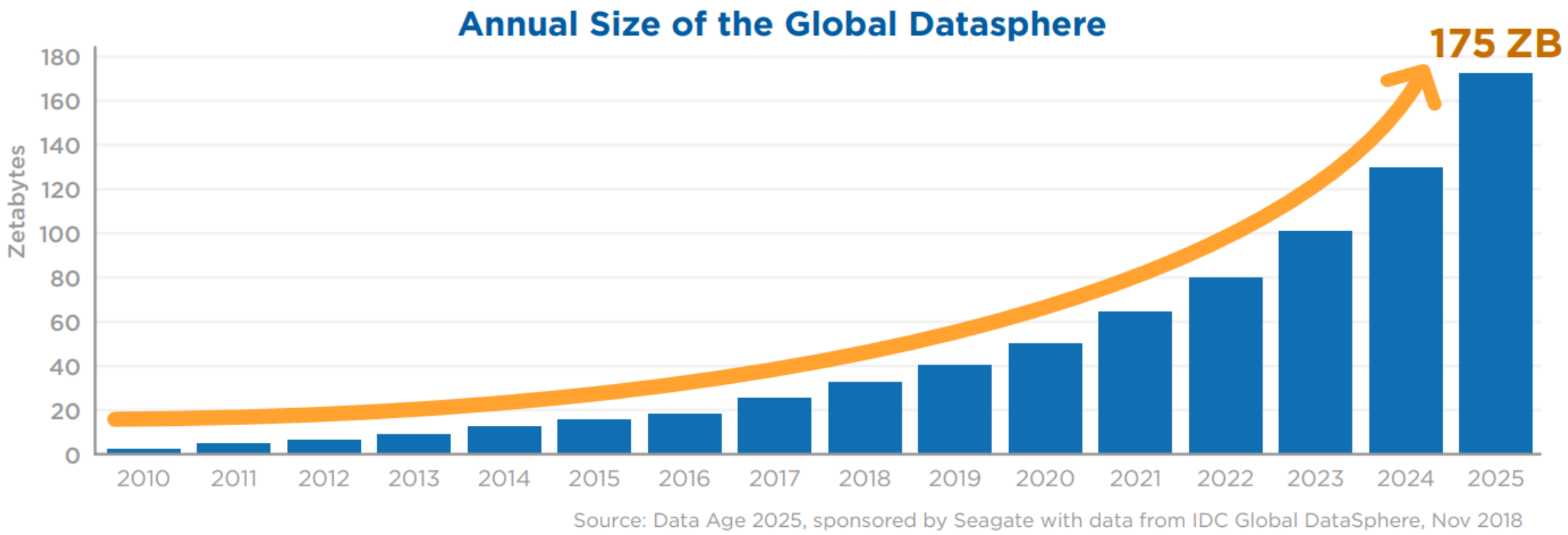
GPUKV: An Integrated Framework with KVSSD and GPU Through P2P Communication Support

Min-Gyo Jung[1], Chang-Gyu Lee[1], Donggyu Park[1], Sungyong Park[1], Jungki Noh[2], Woosuk Chung[2], Kyoung Park[2], Youngjae Kim[1]

[1]Sogang University, Seoul, Republic of Korea, [2]SK hynix

The Era of Data Overflow

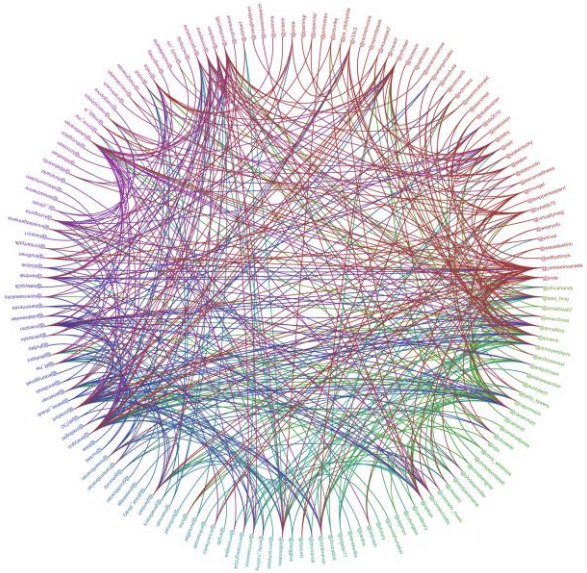


The Era of Data Overflow

How can we handle data?

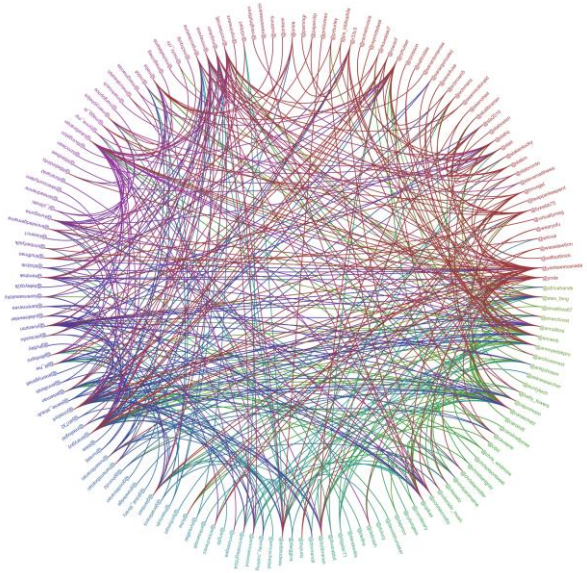
Data-intensive applications

Data-intensive applications

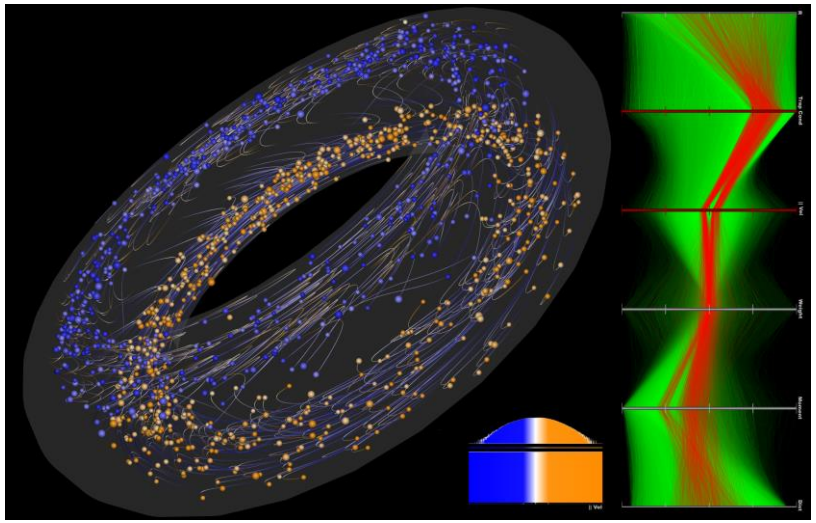


Graph Processing

Data-intensive applications

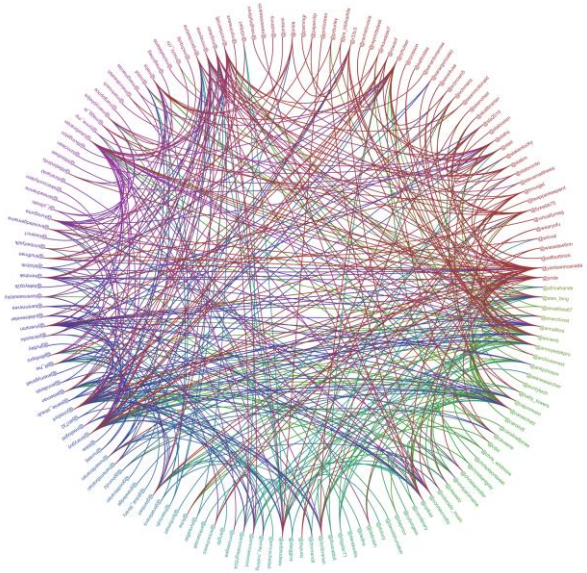


Graph Processing

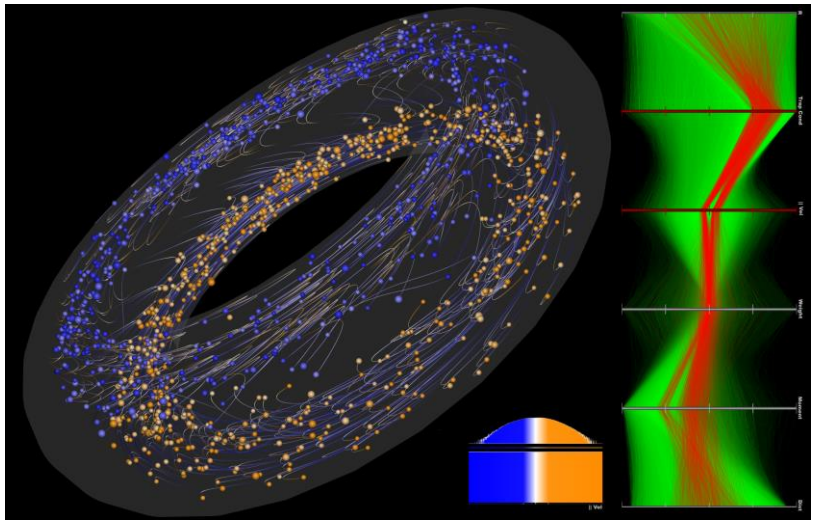


Scientific Simulations

Data-intensive applications



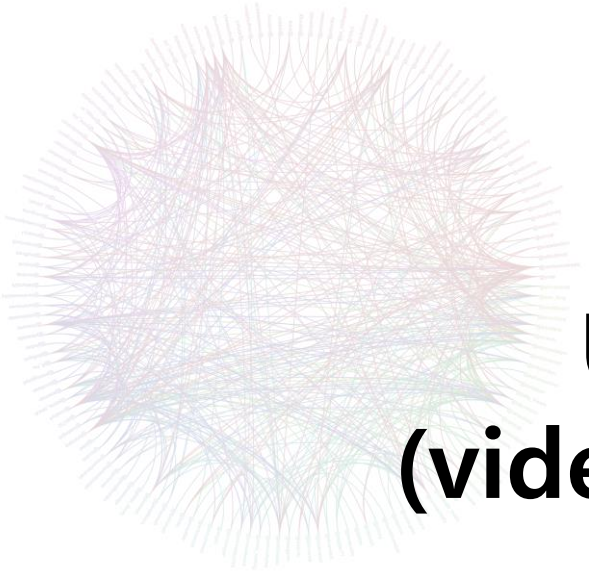
Graph Processing

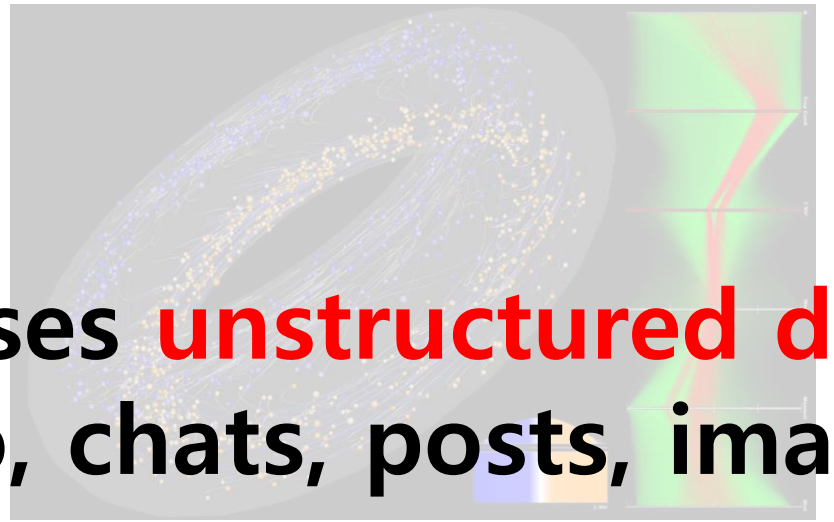


Scientific Simulations

Deep Learning

Data-intensive applications





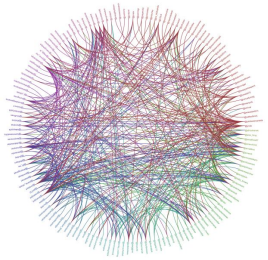
Uses unstructured data
(video, chats, posts, images, …)

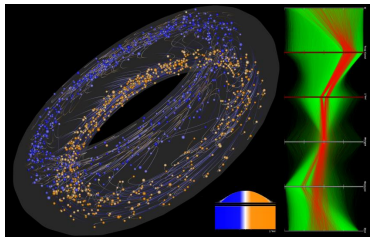
Graph Processing

Scientific Simulations

Deep Learning

Data-intensive applications





+

cassandra

redis

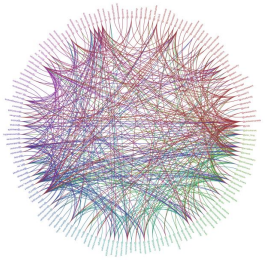
RocksDB

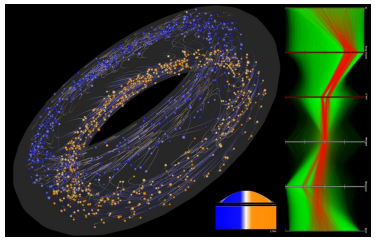
DynamoDB

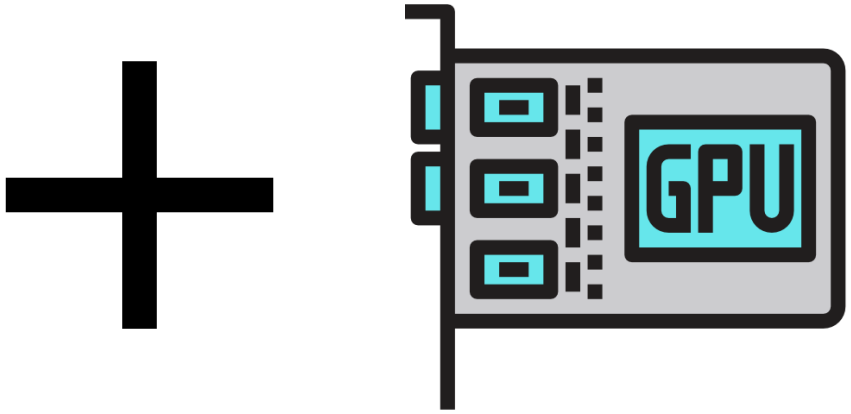
amazon web services

Key-Value Stores

Data-intensive applications meet GPUs







Massive Parallelism

Boost data-intensive applications

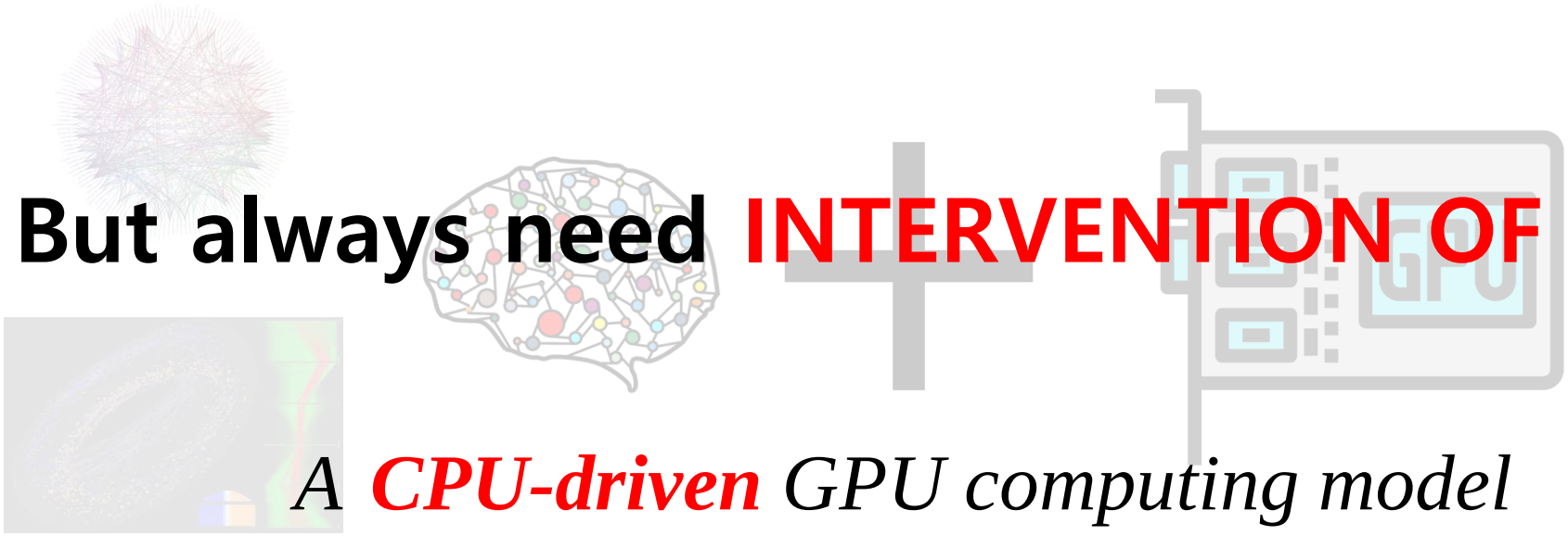
Data-intensive applications meet GPUs

But always need INTERVENTION OF CPU

Massive Parallelism

Boost data-intensive applications

Data-intensive applications meet GPUs



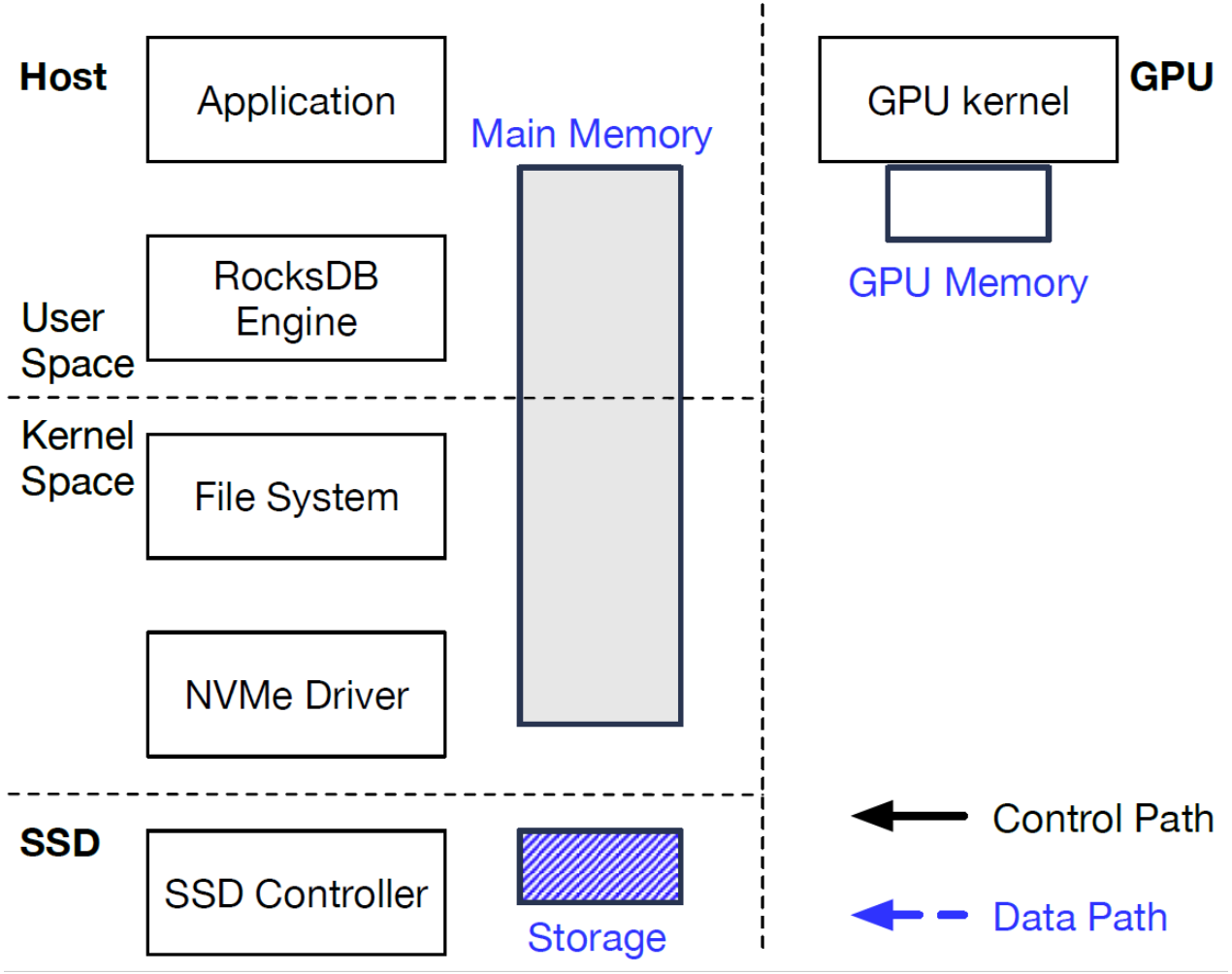
But always need INTERVENTION OF CPU

*A **CPU-driven** GPU computing model*

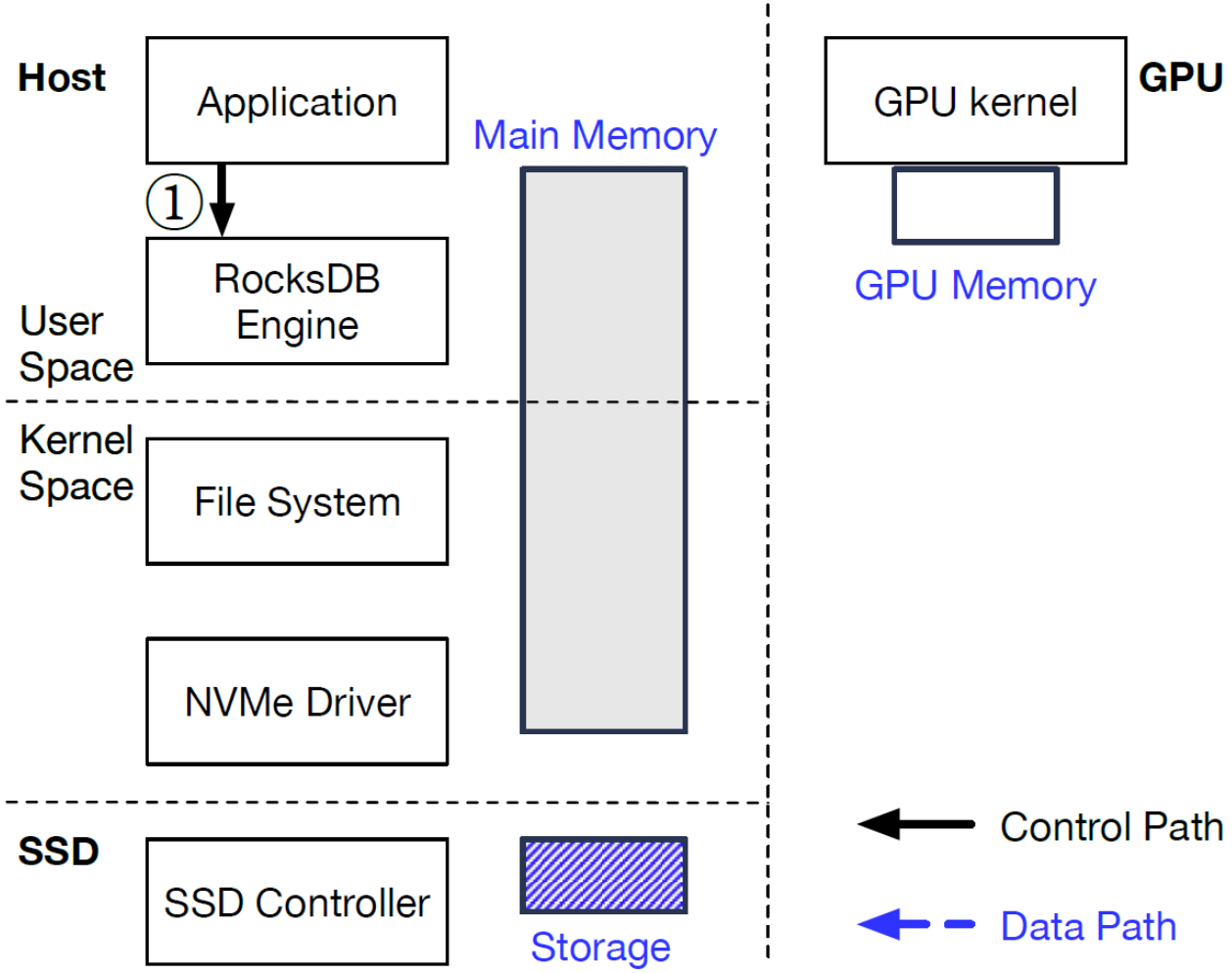
Massive Parallelism

Boost data-intensive applications

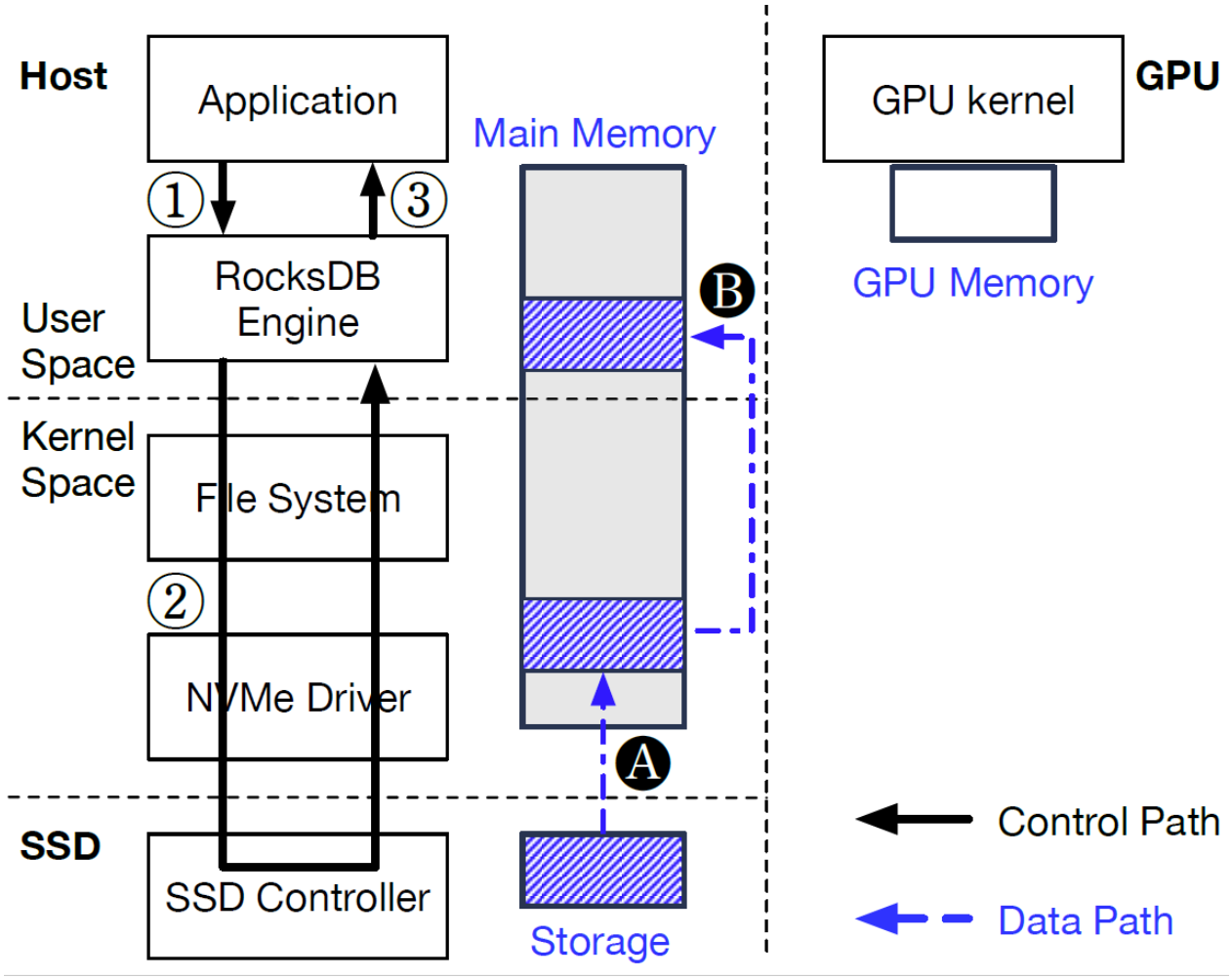
When Data-intensive applications meet GPUs



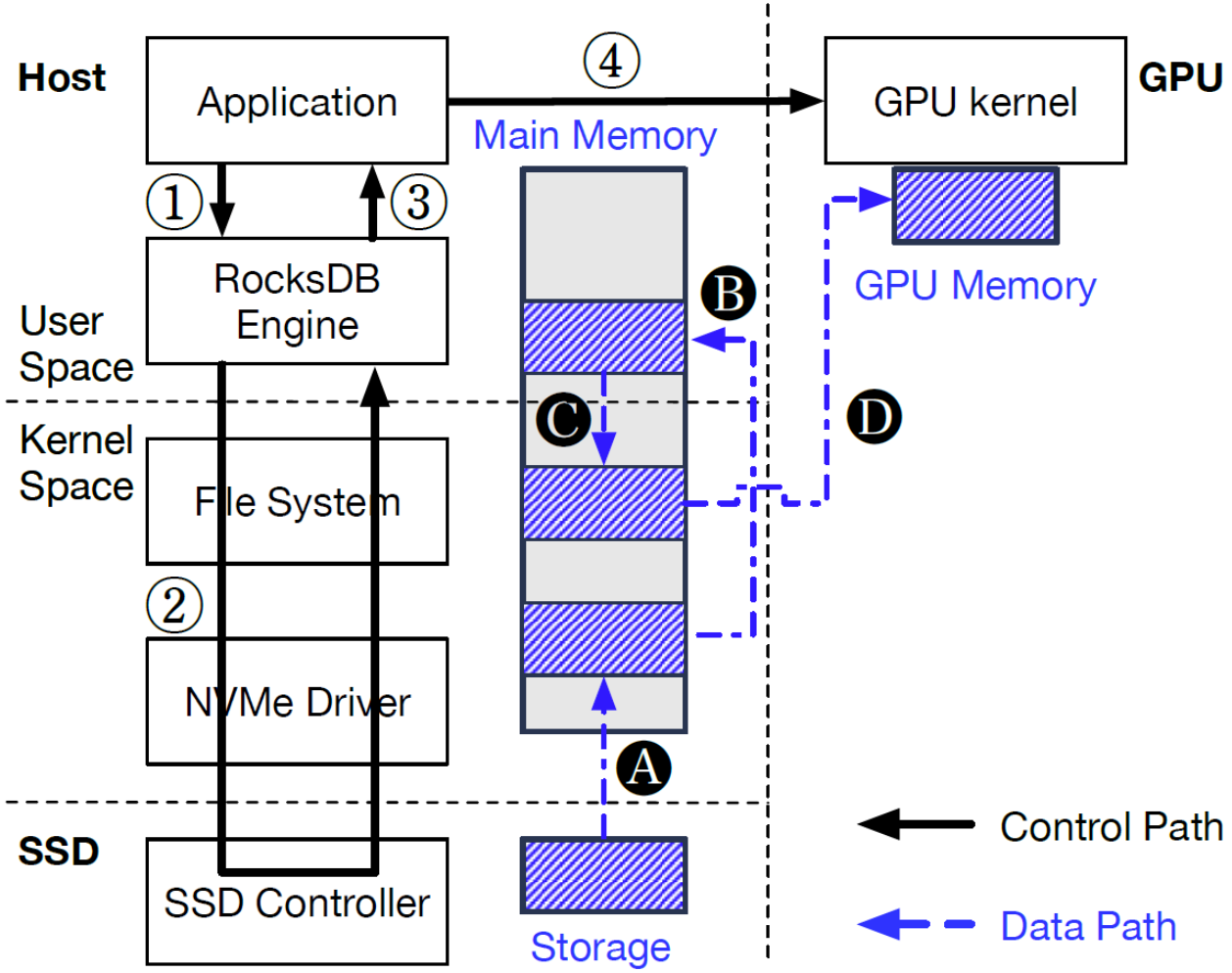
When Data-intensive applications meet GPUs



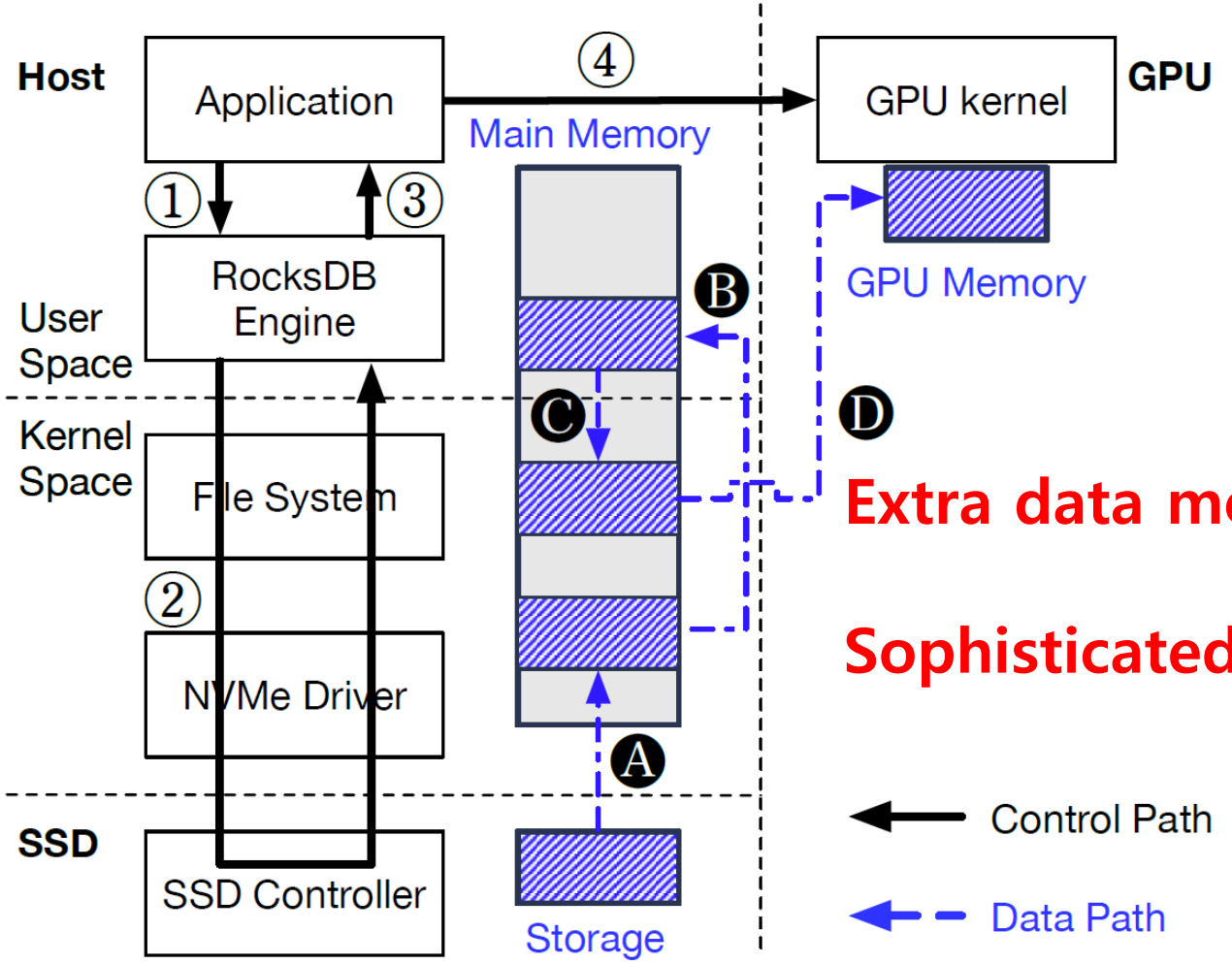
When Data-intensive applications meet GPUs



When Data-intensive applications meet GPUs



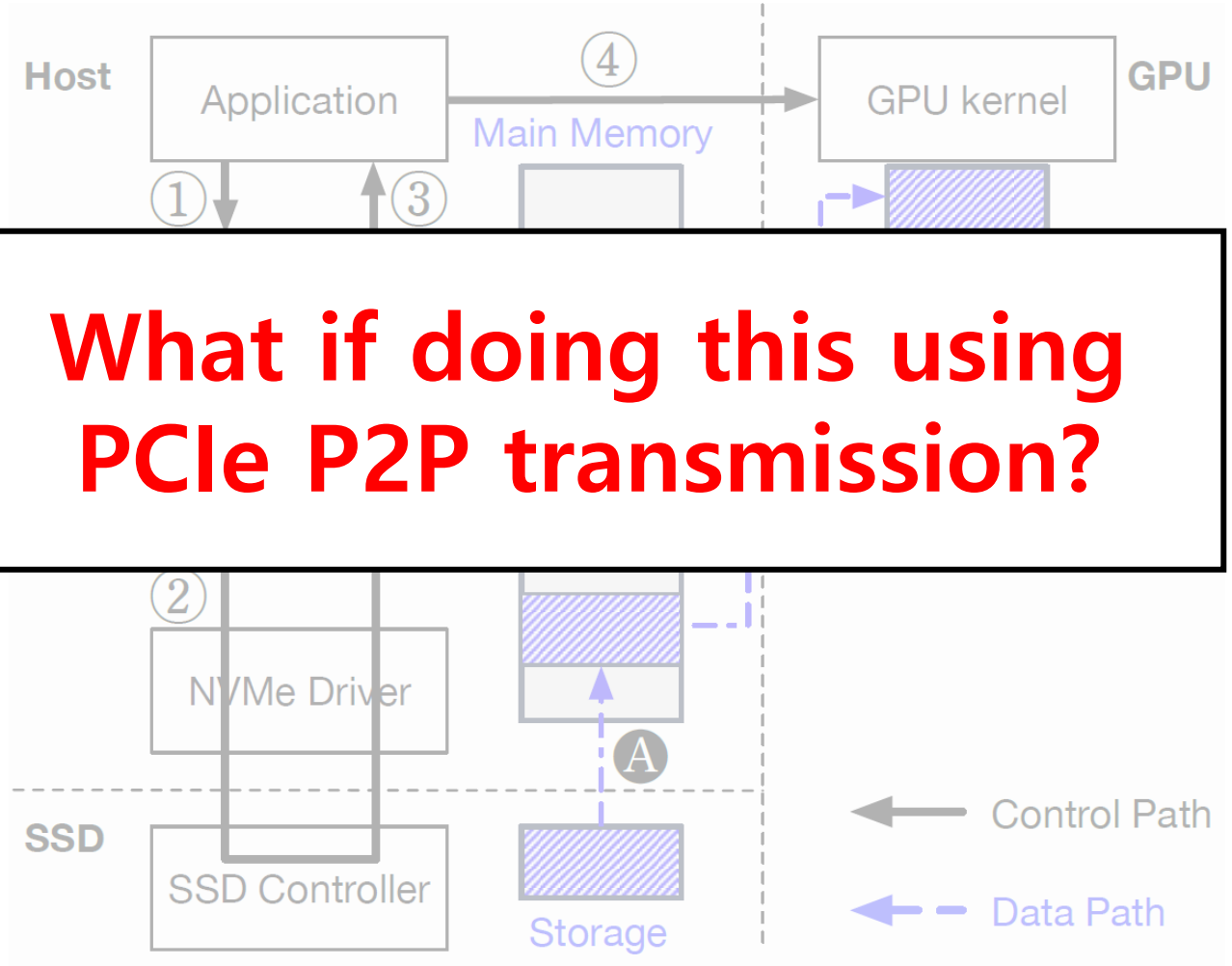
When Data-intensive applications meet GPUs



Extra data movement

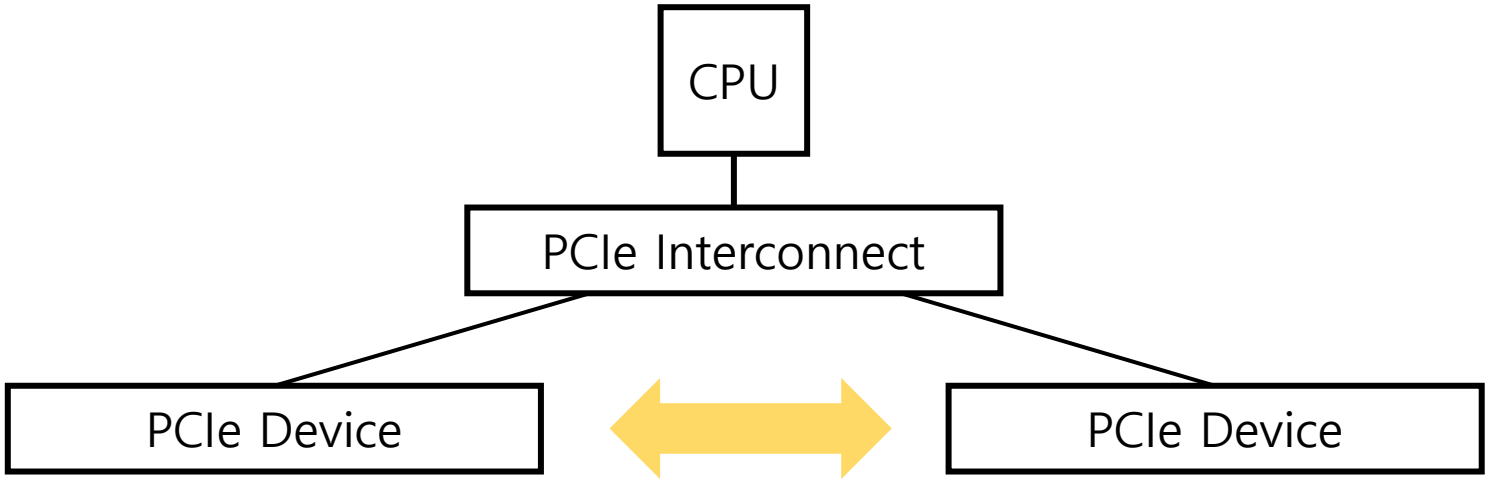
Sophisticated control path

When Data-intensive applications meet GPUs



What if doing this using PCIe P2P transmission?

PCIe P2P Communication



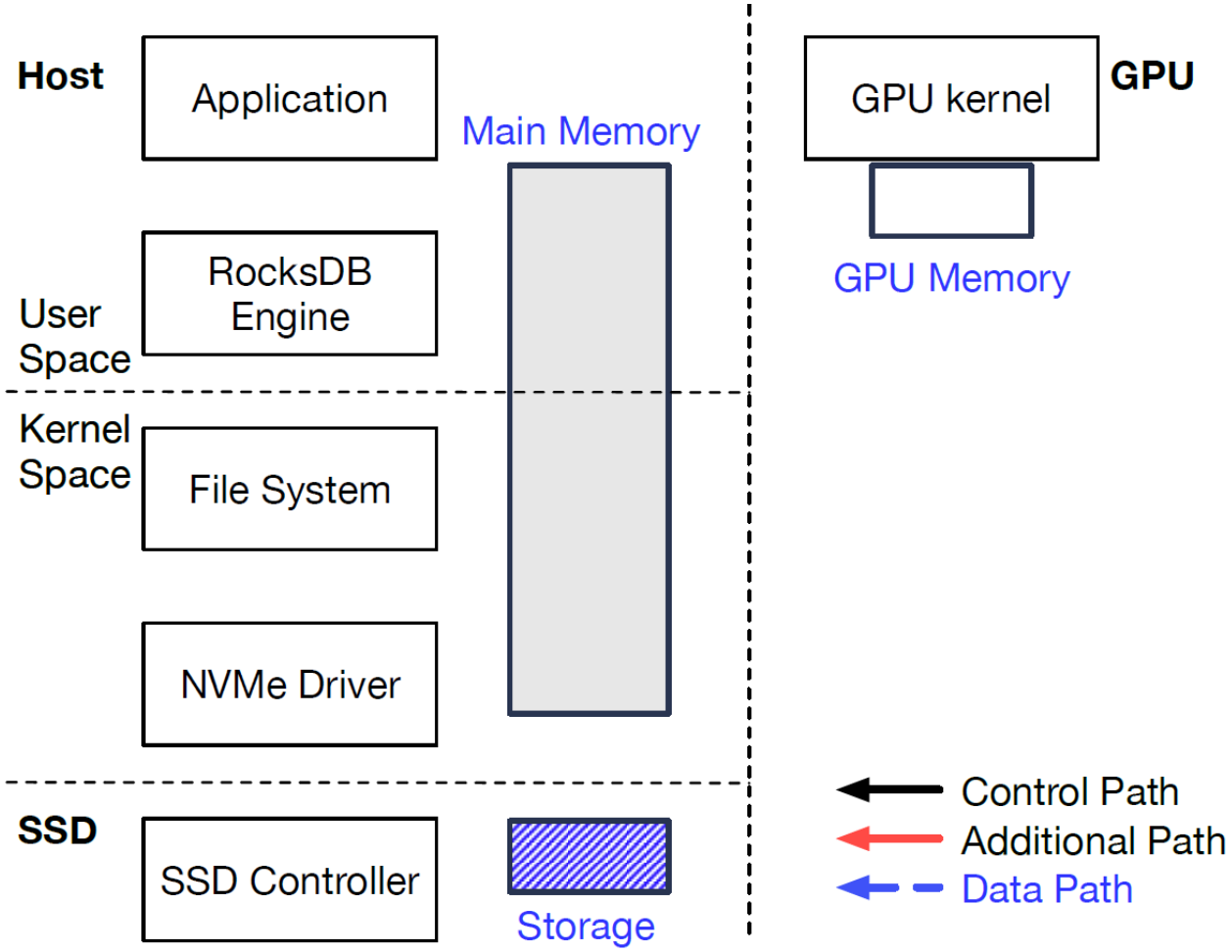
Direct Data Transfer

NVIDIA GPUDirect RDMA
AMD's DirectGMA

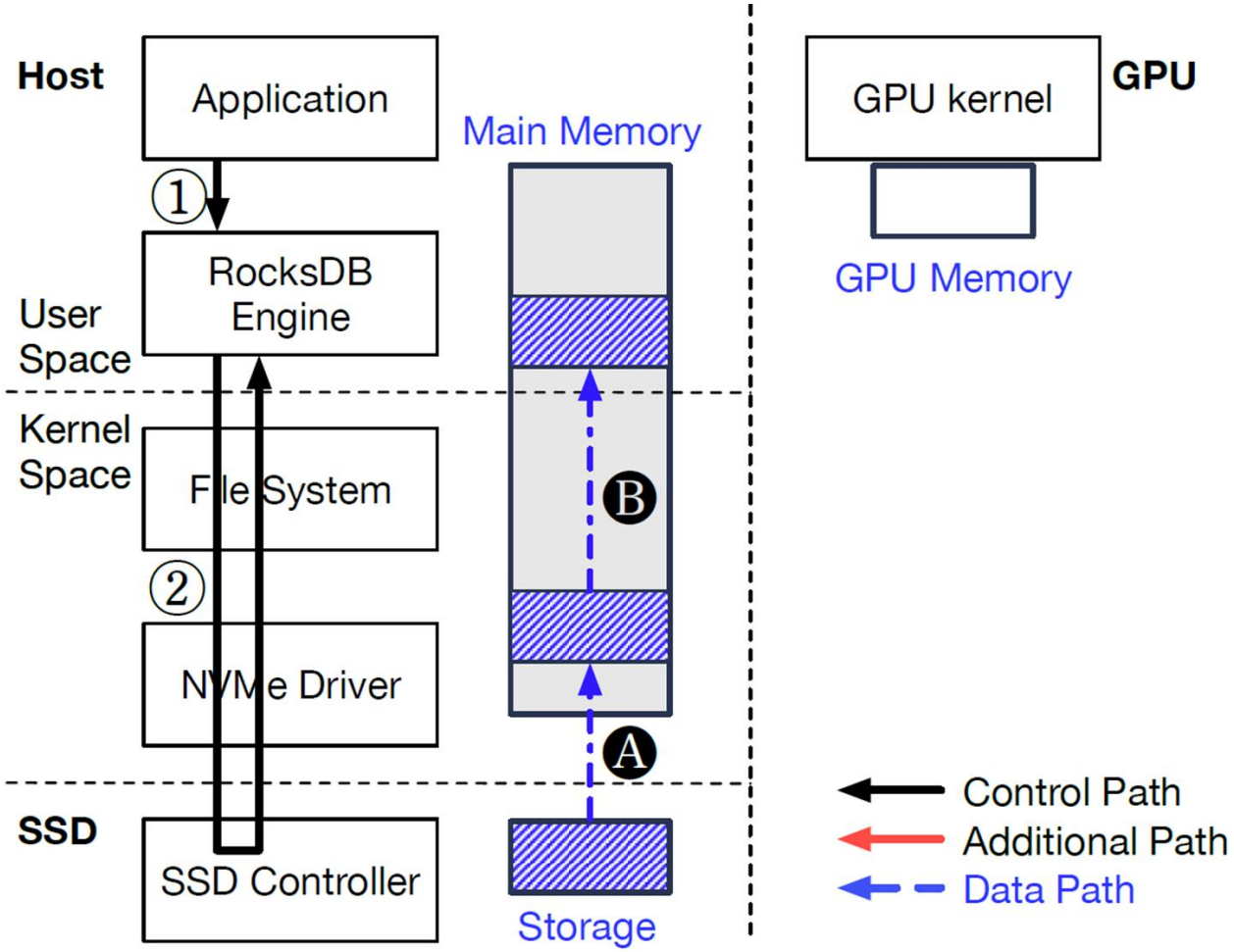
PCIe P2P Communication

- ❑ **GPUs are PCIe Devices**
- ❑ **NVMe SSDs use PCIe Interconnect**
- ❑ **Related works: SPIN [USENIX ATC `17], NVMMU [PACT `15], Morpheus [SIGARCH `16], HippogriffDB [VLDB `16], …**

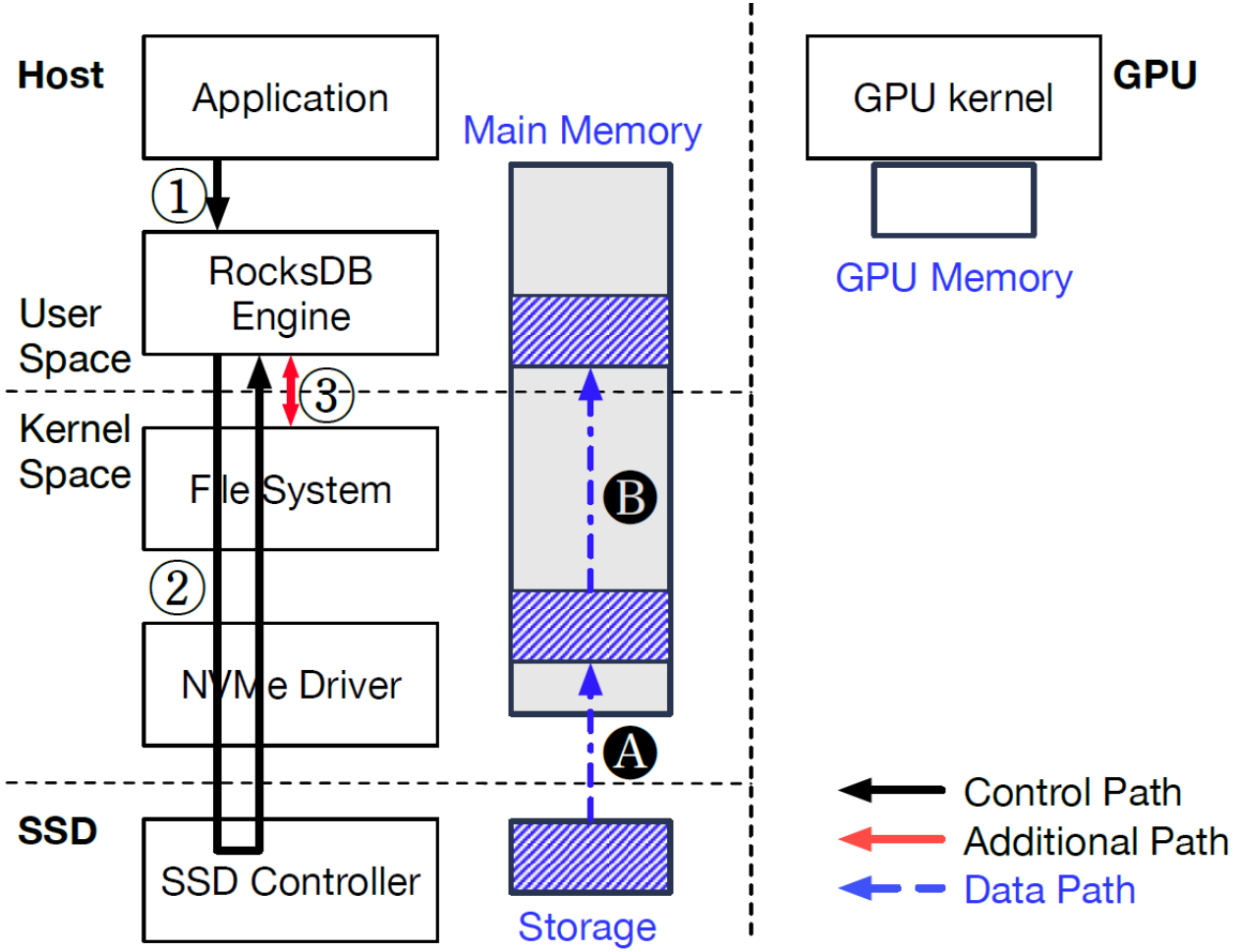
Data Transfer using PCIe P2P



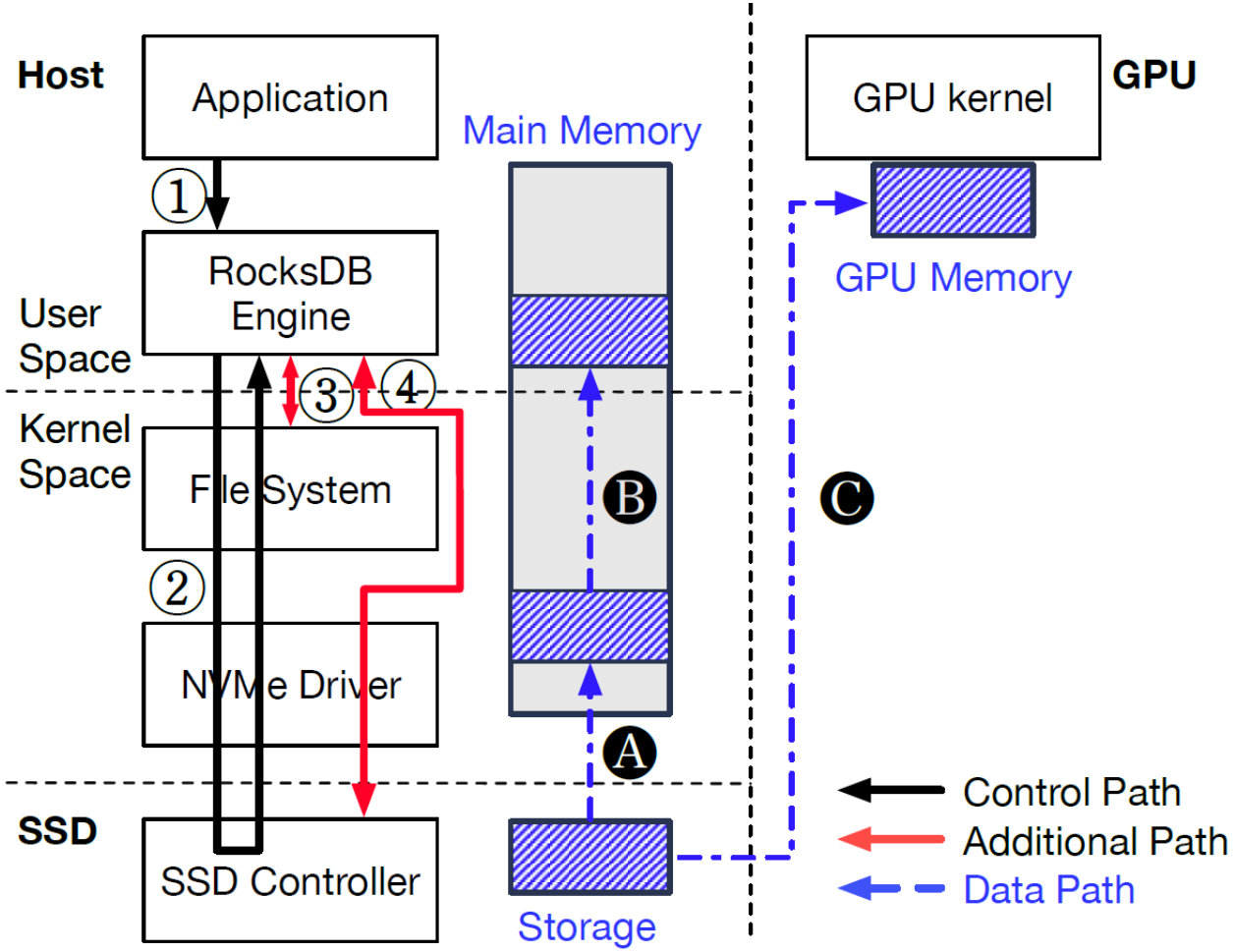
Data Transfer using PCIe P2P



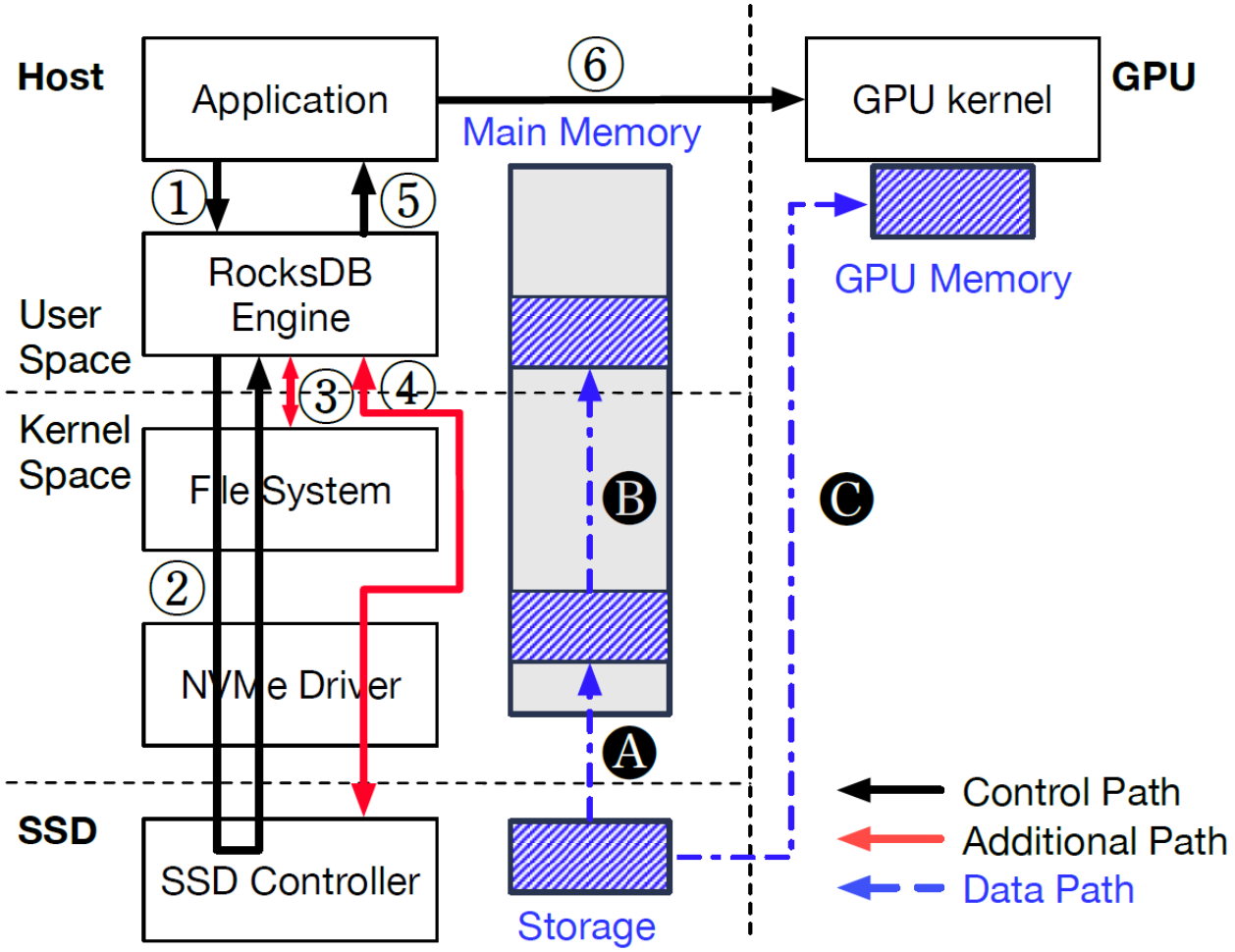
Data Transfer using PCIe P2P



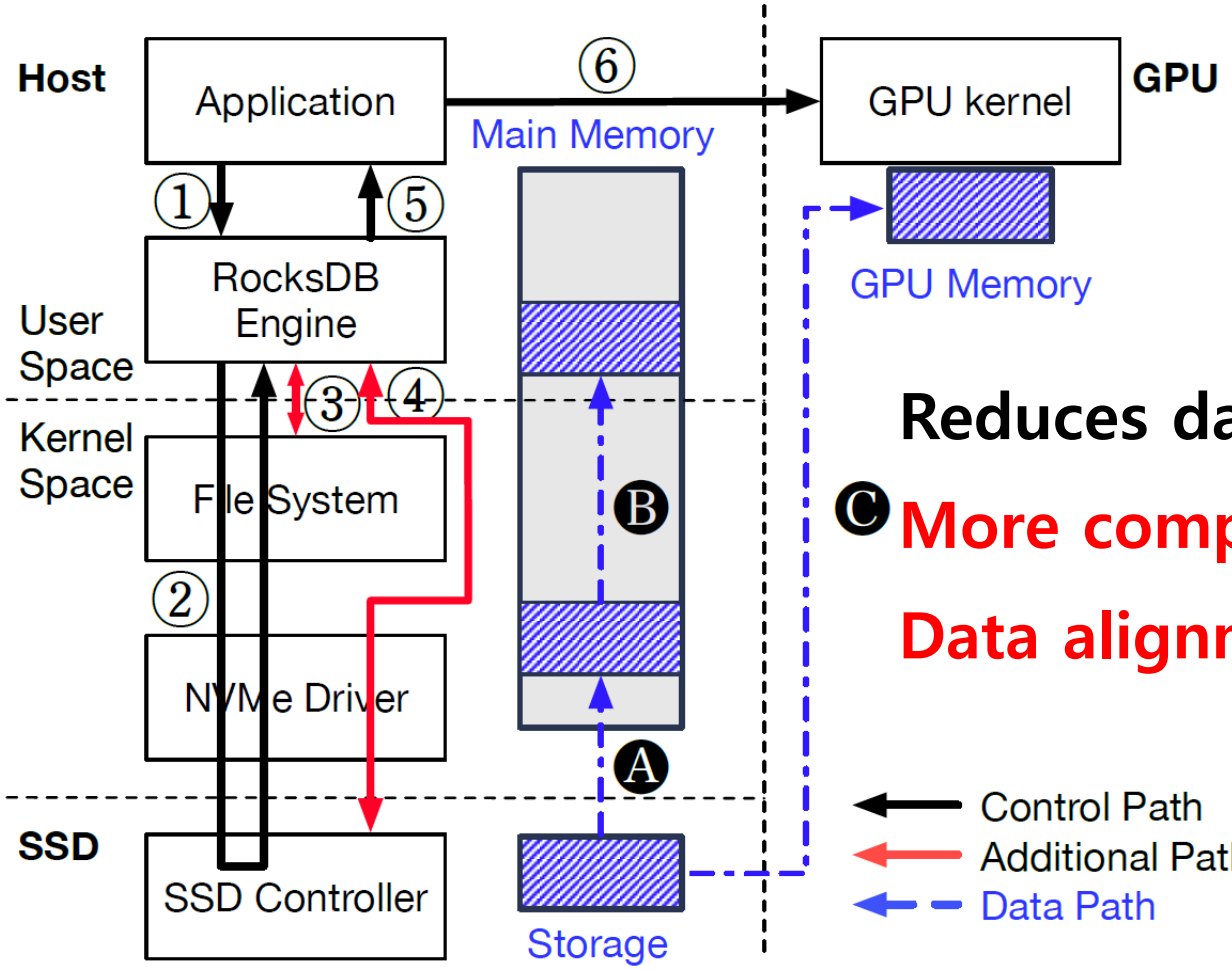
Data Transfer using PCIe P2P



Data Transfer using PCIe P2P



Data Transfer using PCIe P2P



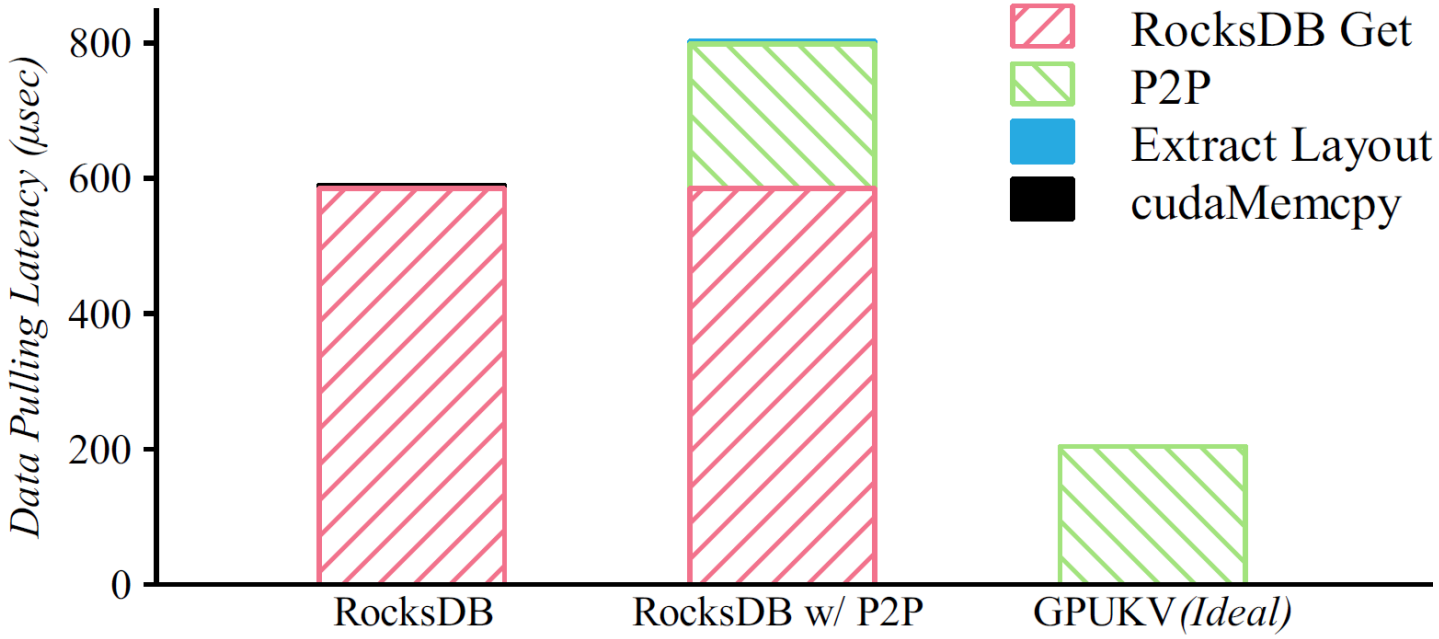
Reduces data movement

C **More complicated control path**

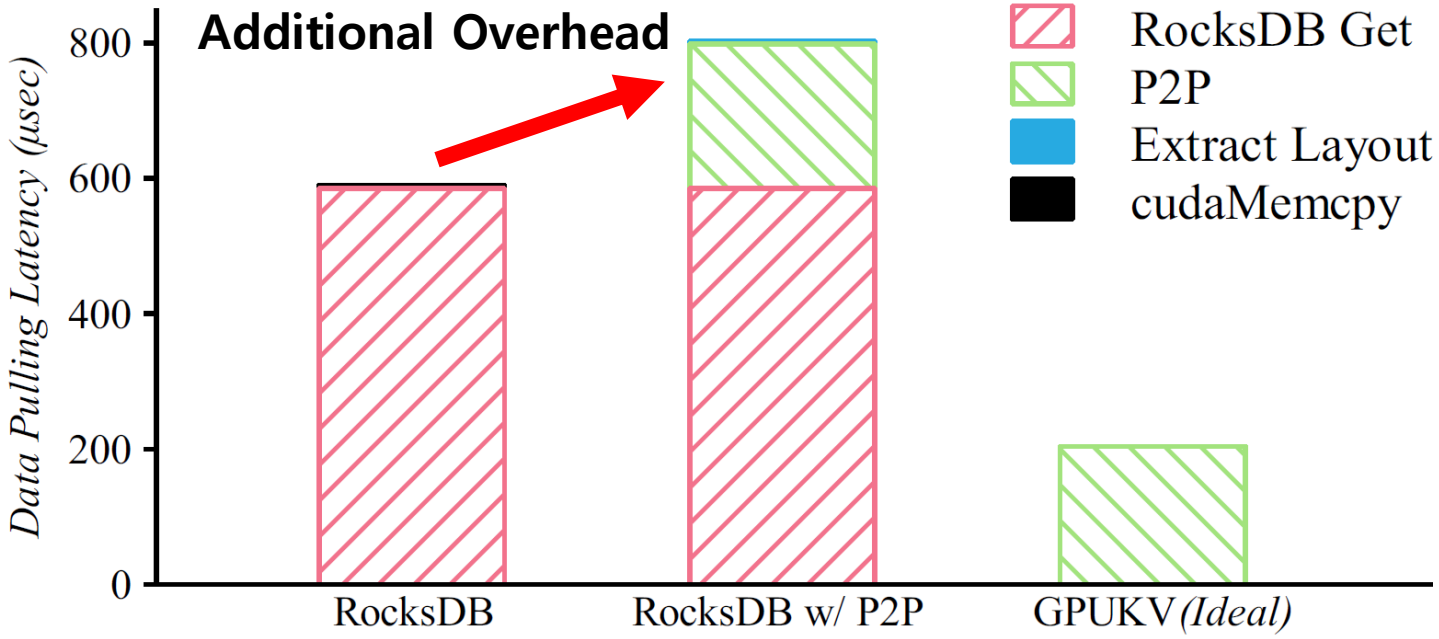
Data alignment for P2P

Does it really work?

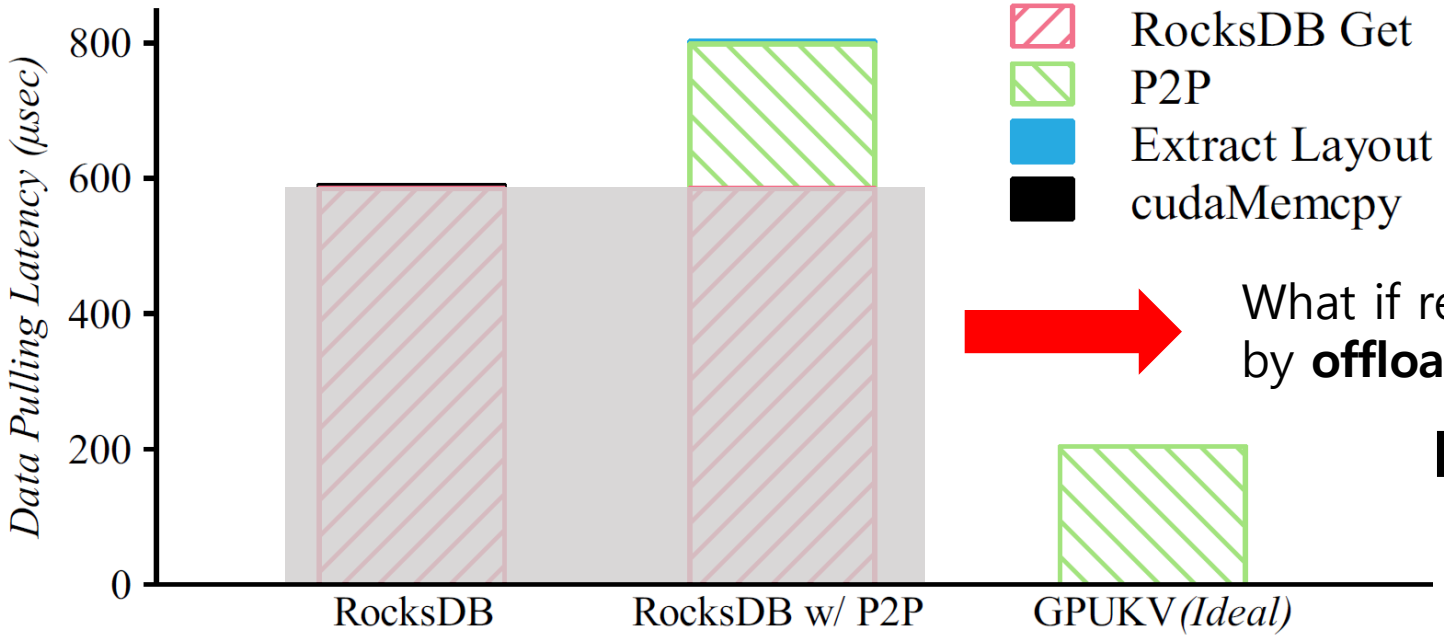
Does it really work?



Does it really work?



Does it really work?



What if remove overhead by **offloading** RocksDB part?

KVSSD

Let's totally remove **intervention of CPU!**

*A **GPU-driven** GPU computing model*

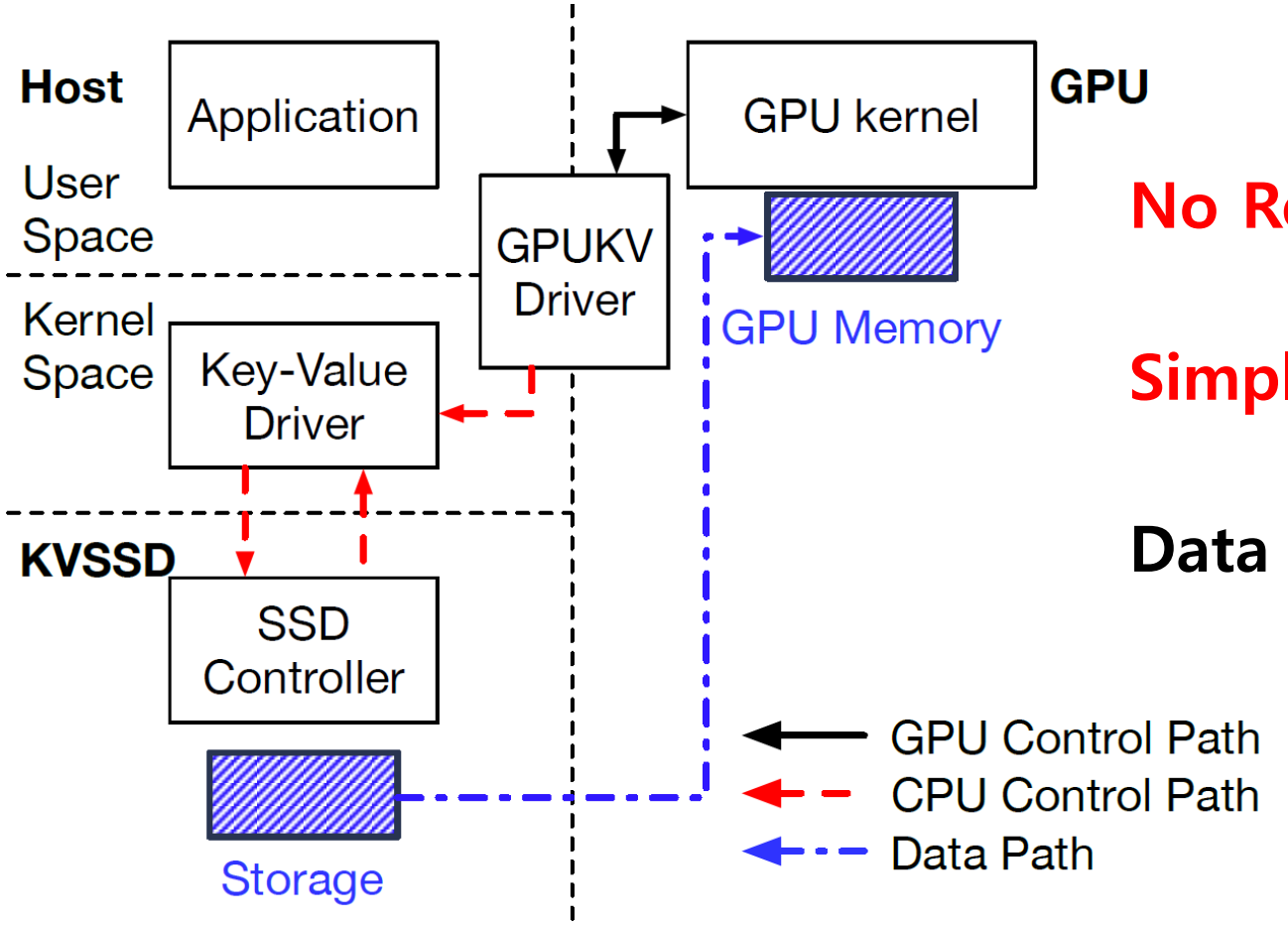
Our Contributions

1. **Key-Value Store Abstraction to GPU**

2. **Key-Value SSD**

3. **P2P Communication between KVSSD and GPU**

Architecture of GPUKV



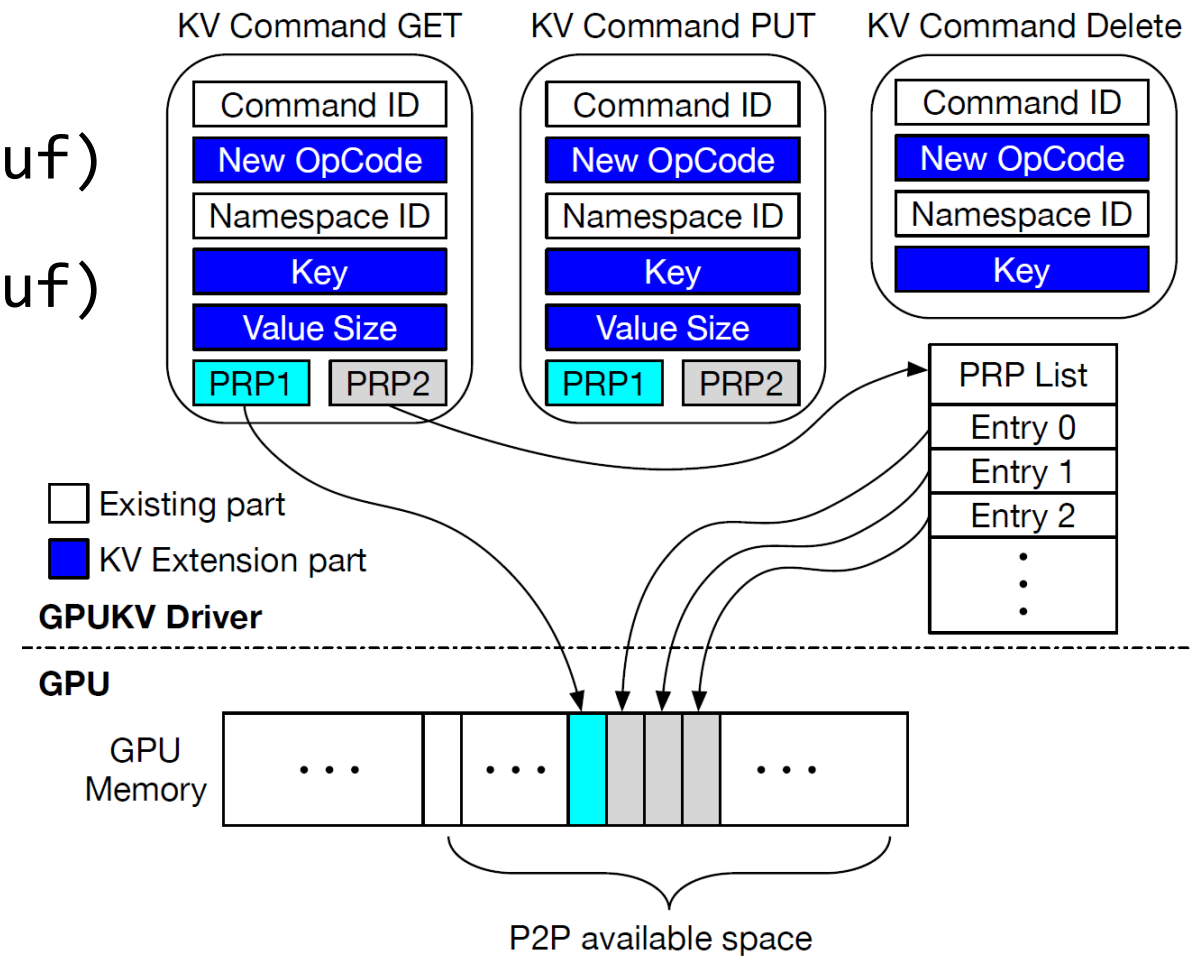
No Redundant data copy

Simple and short Control Path

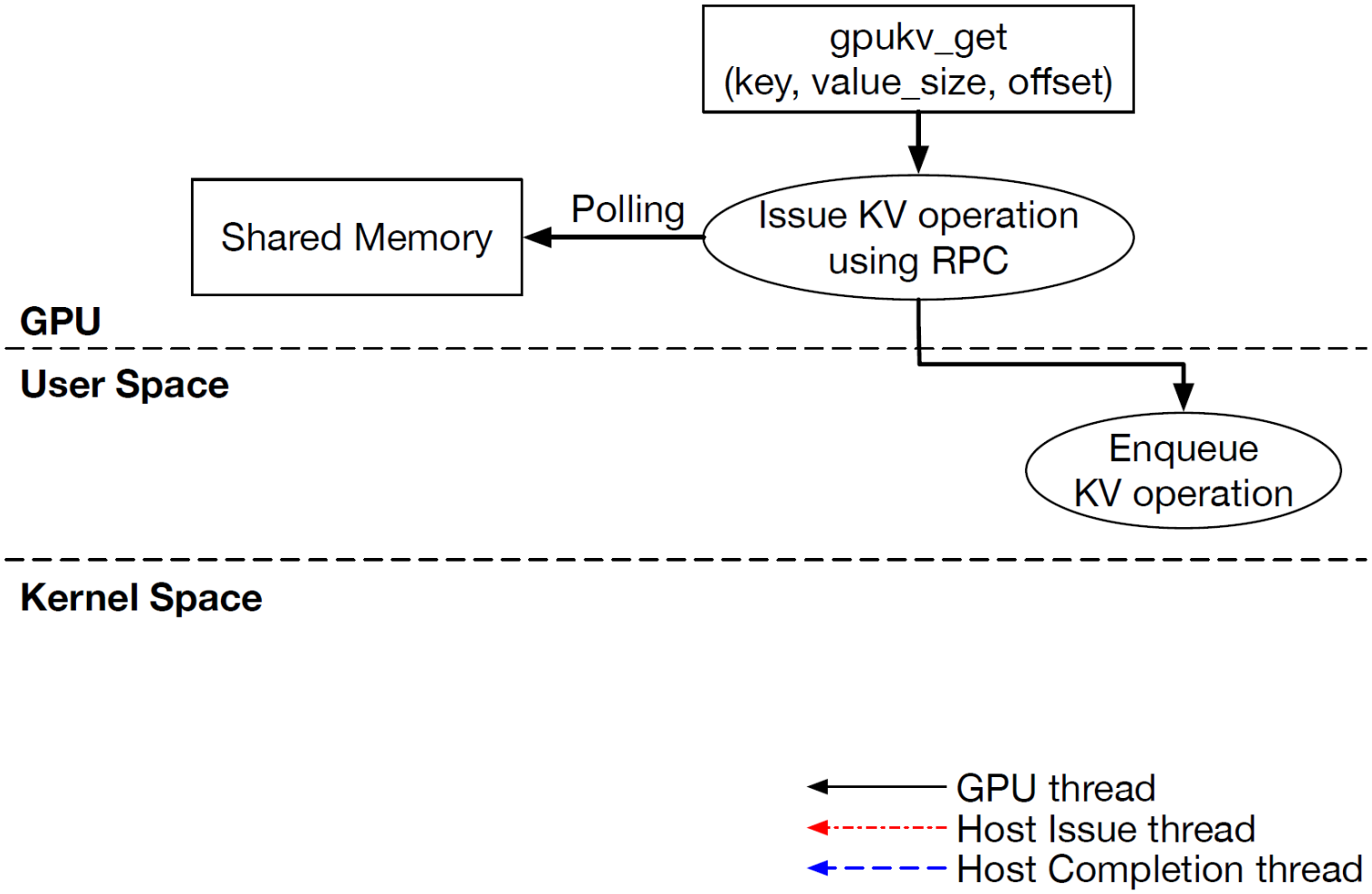
Data request from GPU itself

Key-Value command extension

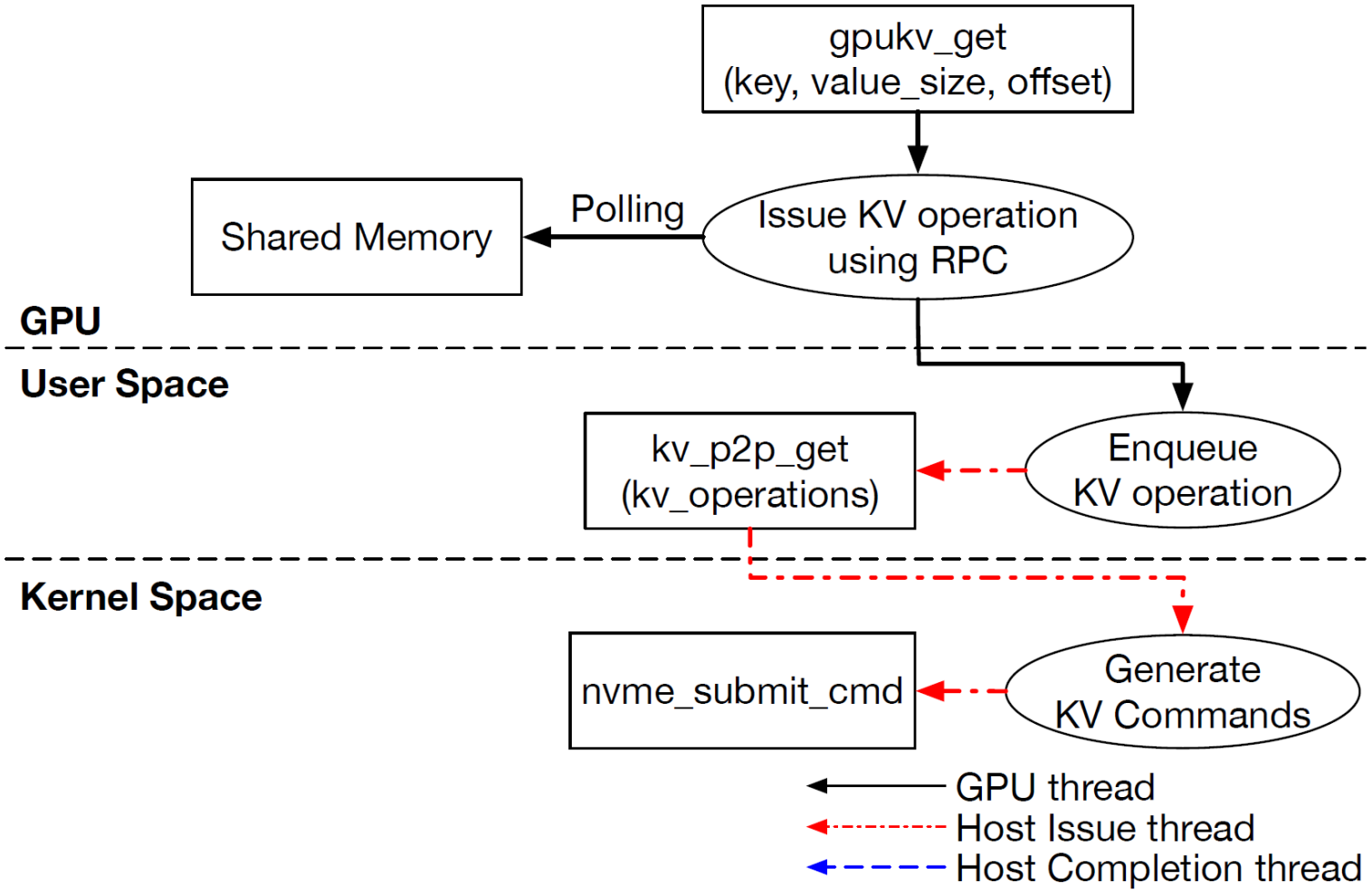
- **Support three commands**
  - `gpukv_get(key, val_size, buf)`
  - `gpukv_put(key, val_size, buf)`
  - `gpukv_delete(key)`



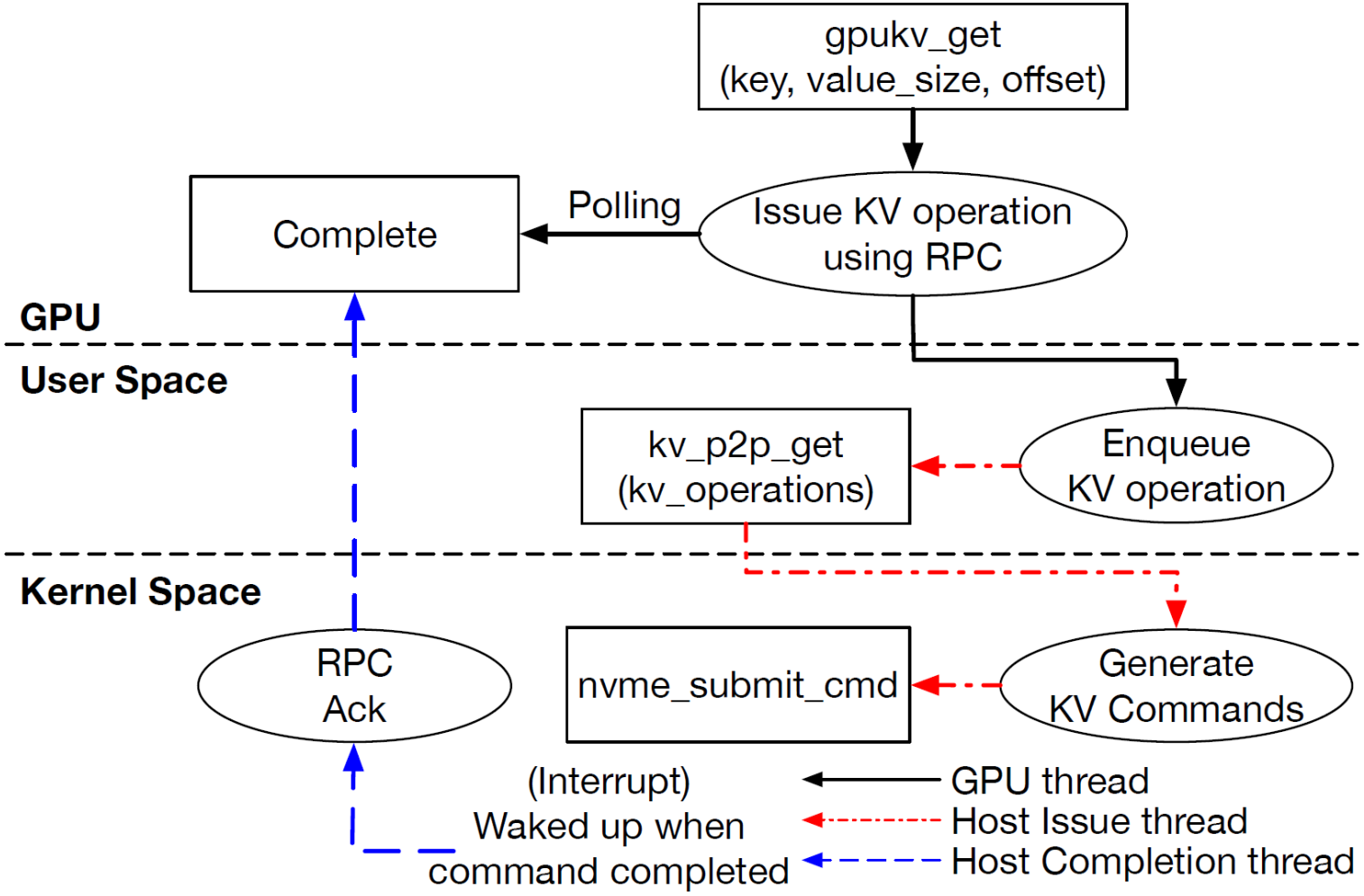
Functional Flow of GPUKV



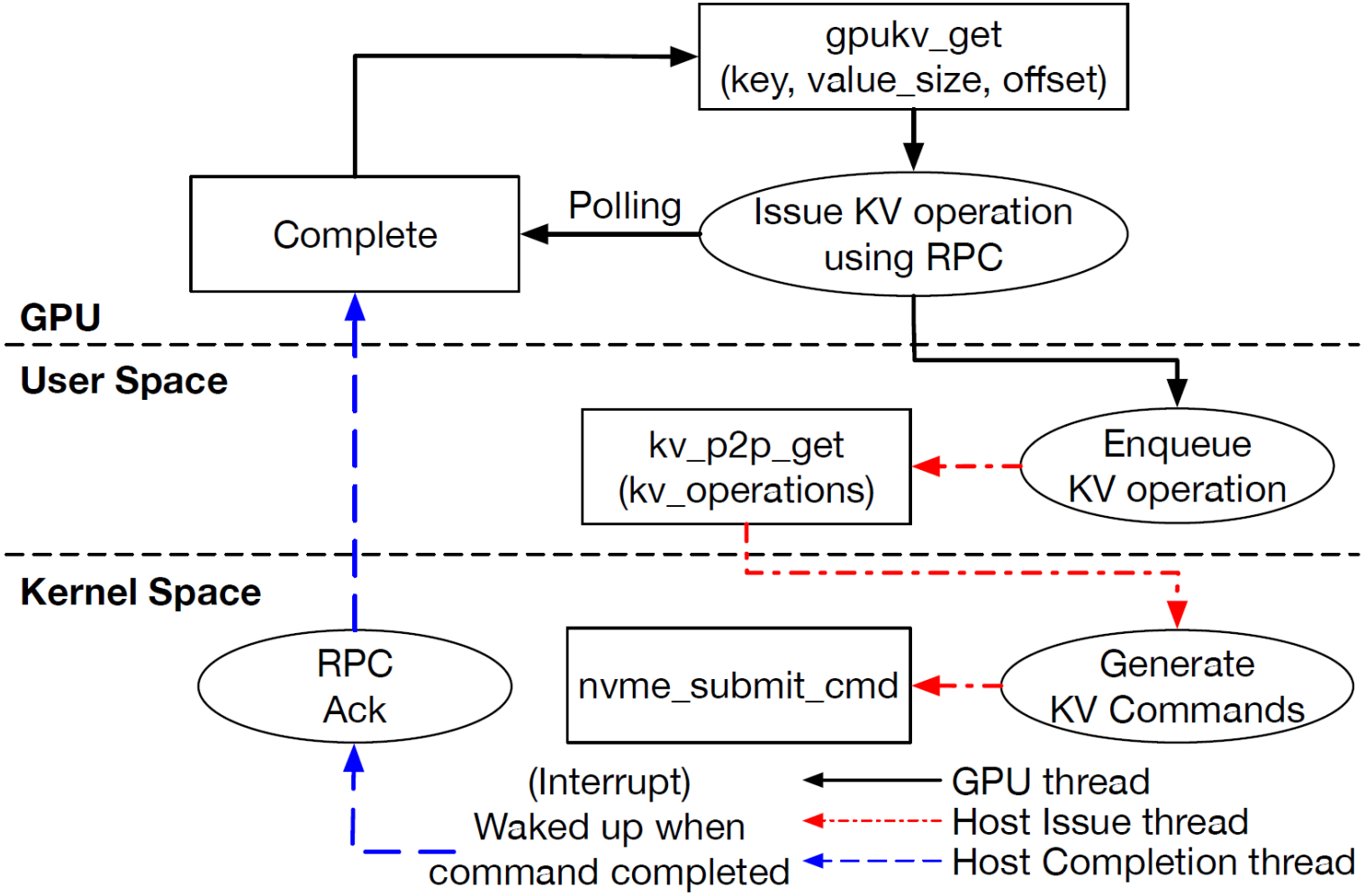
Functional Flow of GPUKV



Functional Flow of GPUKV



Functional Flow of GPUKV



System setup

- **KVSSD: Cosmos+ OpenSSD Platform**
- **GPU: NVIDIA Quadro P4000**
- **Host machine:**
  - i5-4690
  - DDR3 4GB x2
  - Linux kernel 4.1.52
  - KVSSD and GPU

Evaluation Setup

- **Two different configurations about Host**
  - Evaluate the performance and resource use efficiency
  - *low-end server*: 2 cores, 800MHz
  - *high-end server*: 4 cores, 3.5GHz

Workloads

- ***Streaming* workload**
  - **Predictable** data access pattern
  - Can **prefetch** the next input dataset
  - ex) Page Rank (PR)
- ***Dynamic* workload**
  - **Unpredictable** data access pattern
  - **Cannot prefetch** the next input dataset
  - ex) Breadth-First Search (BFS)

Experimental Results

- **Synthetic workload**
  - Streaming & Dynamic
- **Realistic workload (GARDENIA [JETC `19] graph processing workload)**
  - Streaming (Page Rank) & Dynamic (BFS)
- **CPU utilization**
  - Streaming pattern of Synthetic workload

System Comparisons

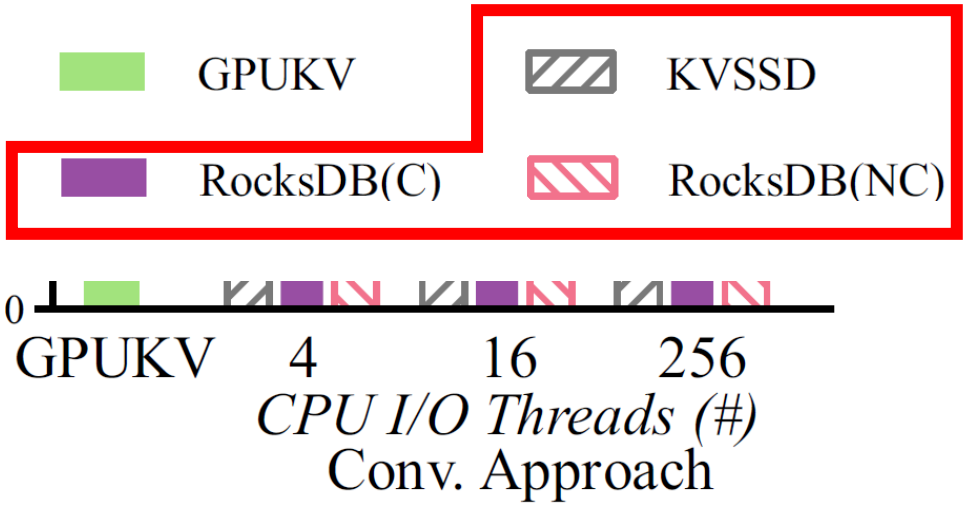
- **GPUKV**
  - **Our system**. Bypass file system using **KVSSD**. Use **P2P** for data transfer.
- **KVSSD**
  - Cannot use P2P. Bypass file system using **KVSSD**. Use ***cudaMemcpy*** for data transfer.
- **RocksDB(C)**
  - **RocksDB runs on the Ext4**. Transfer data using ***cudaMemcpy***.

    Enable RocksDB's **cache** and do **buffered I/O**.
- **RocksDB(NC)**
  - **RocksDB runs on the Ext4**. Transfer data using ***cudaMemcpy***.

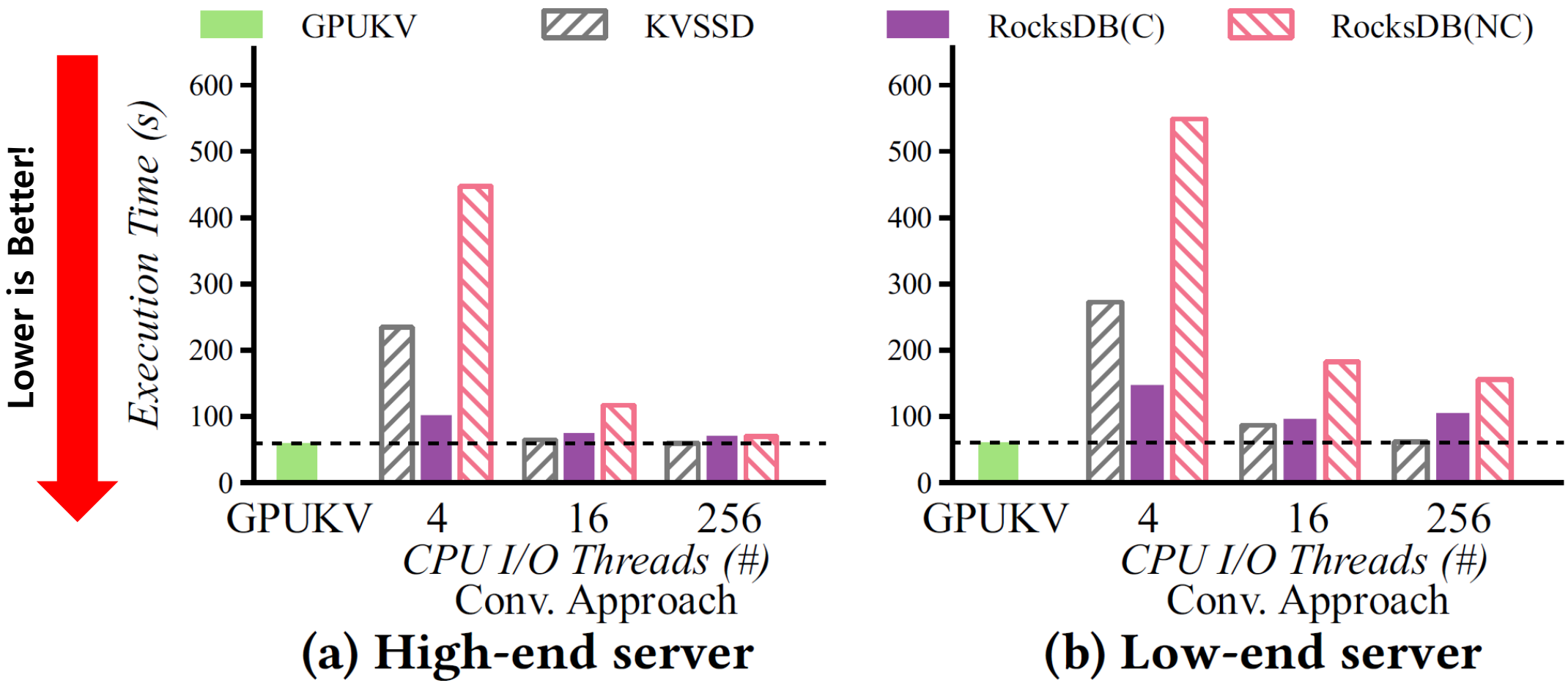
    Disable RocksDB's **cache** and do **directIO**.

Workload Analysis

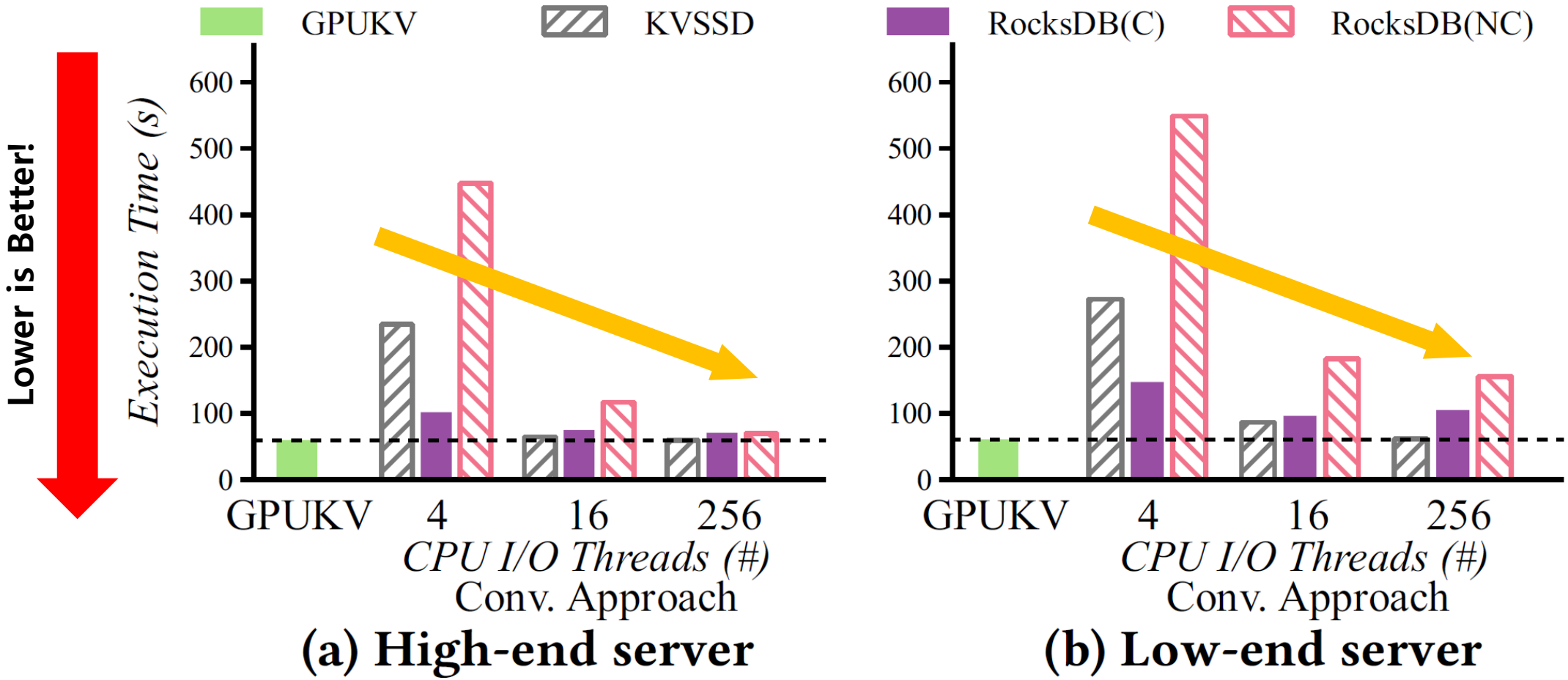
- **GPUKV: 2 I/O Command handling threads**
  - Issue thread / Completion thread
- **Conventional Approach (KVSSD, RocksDB(C), RocksDB(NC))**
  - Increasing CPU I/O threads → Accelerates data movement, Increases CPU use



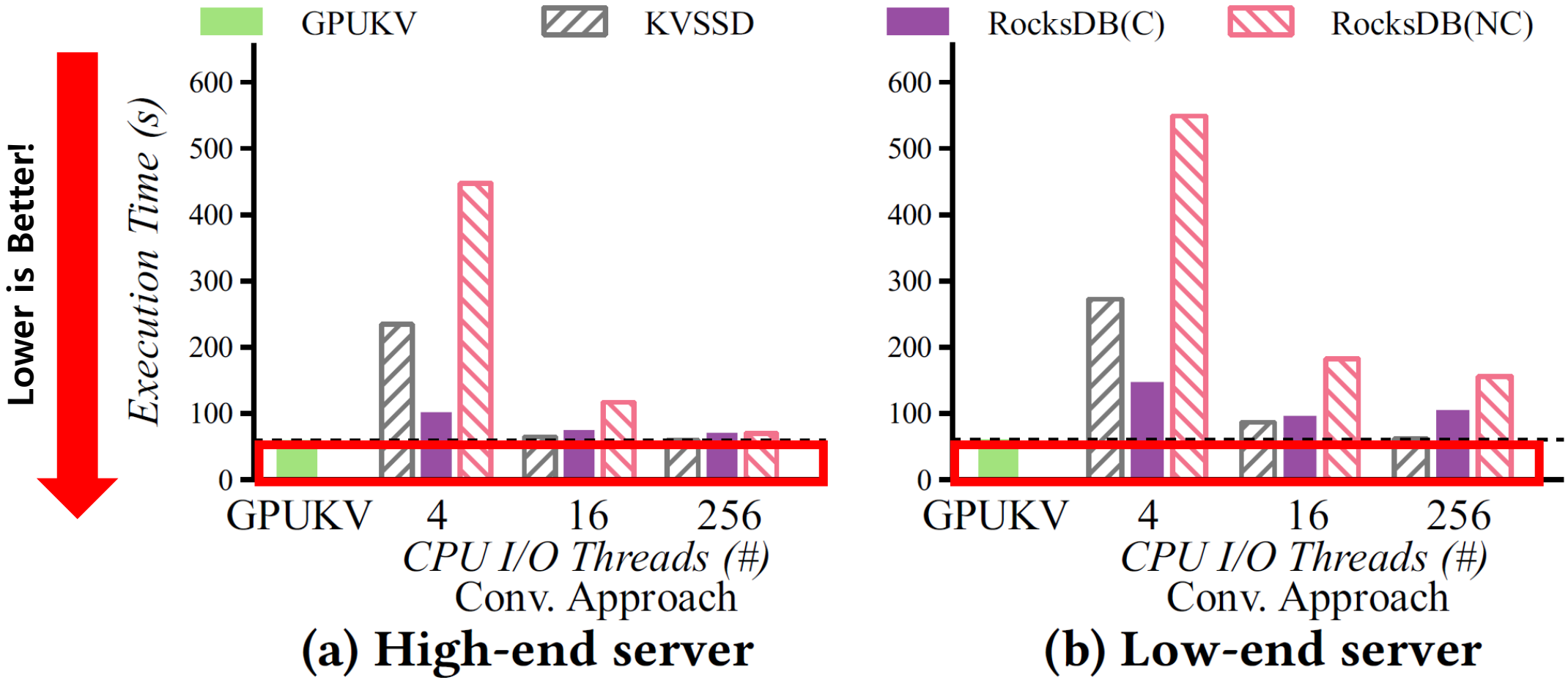
Synthetic workload (Streaming)



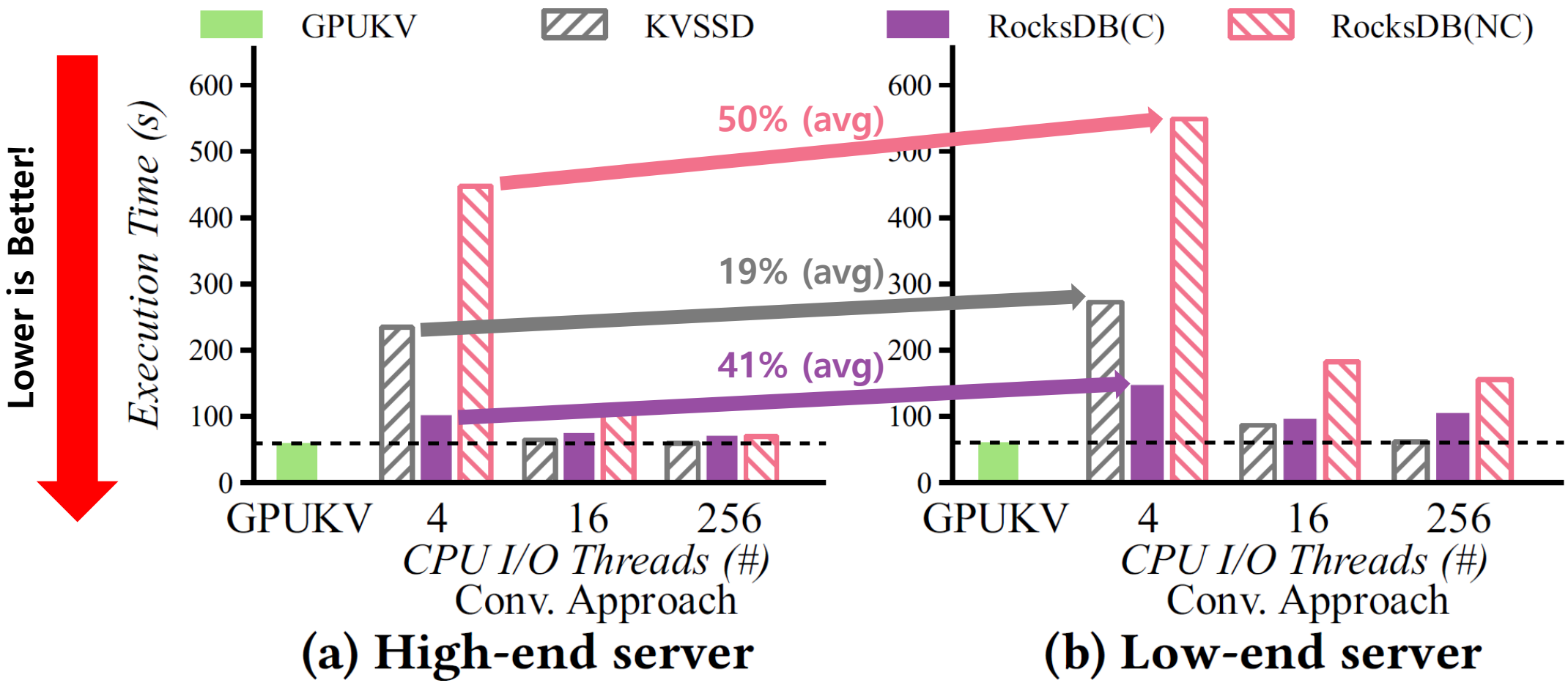
Synthetic workload (Streaming)



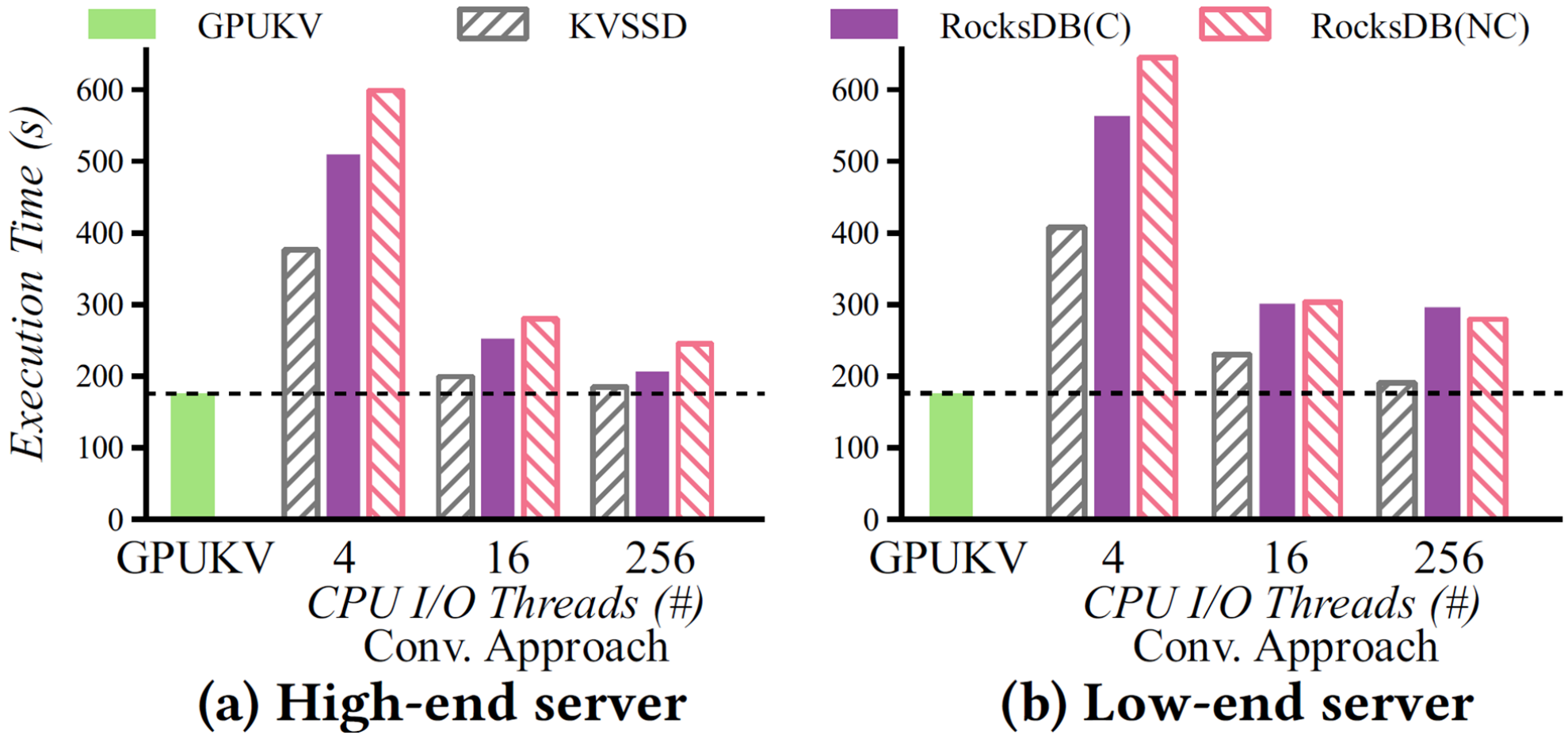
Synthetic workload (Streaming)



Synthetic workload (Streaming)



Synthetic workload (Dynamic)



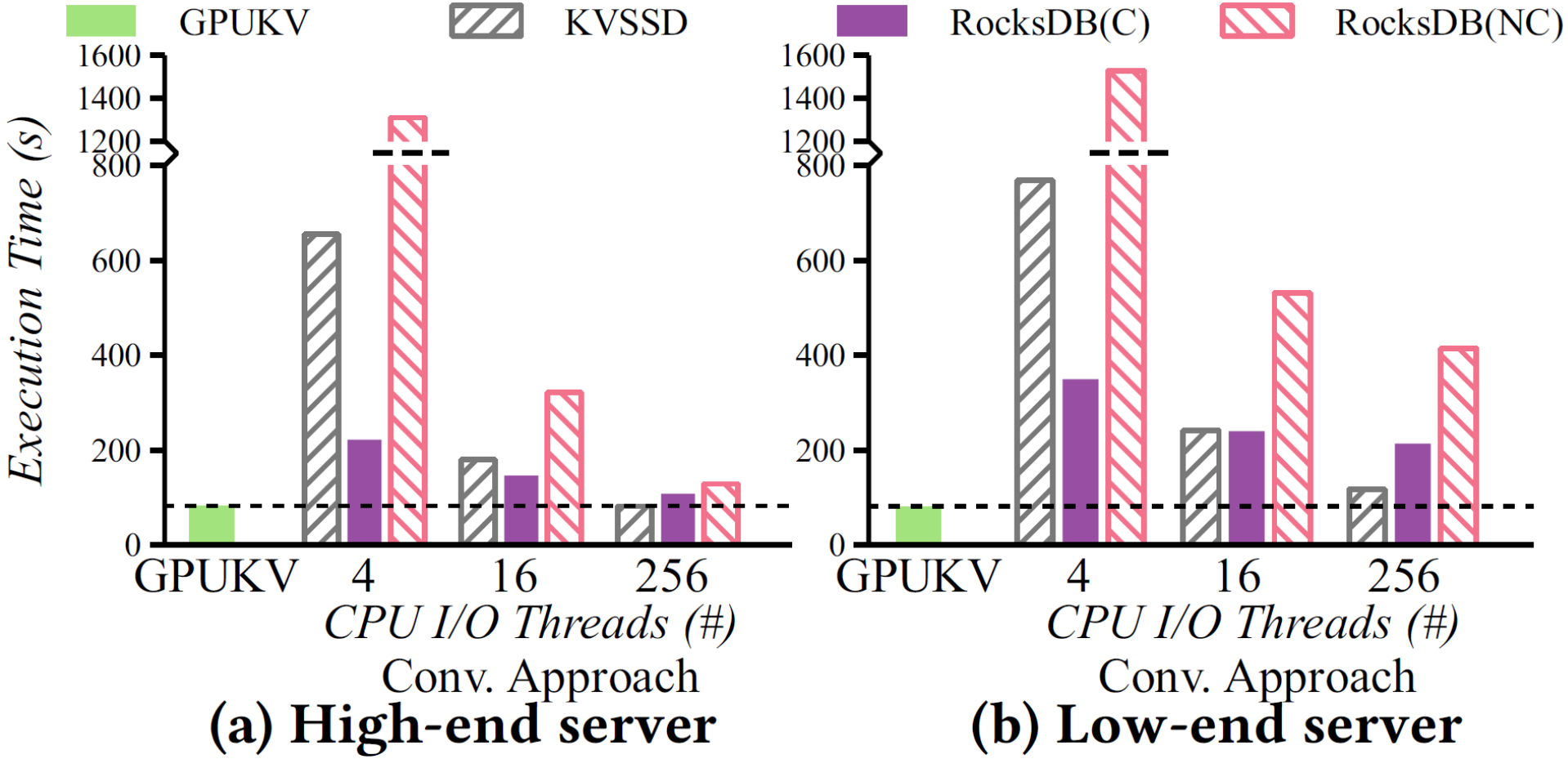
Synthetic workload

- **GPUKV**
  - Best Performance at both CPU setup

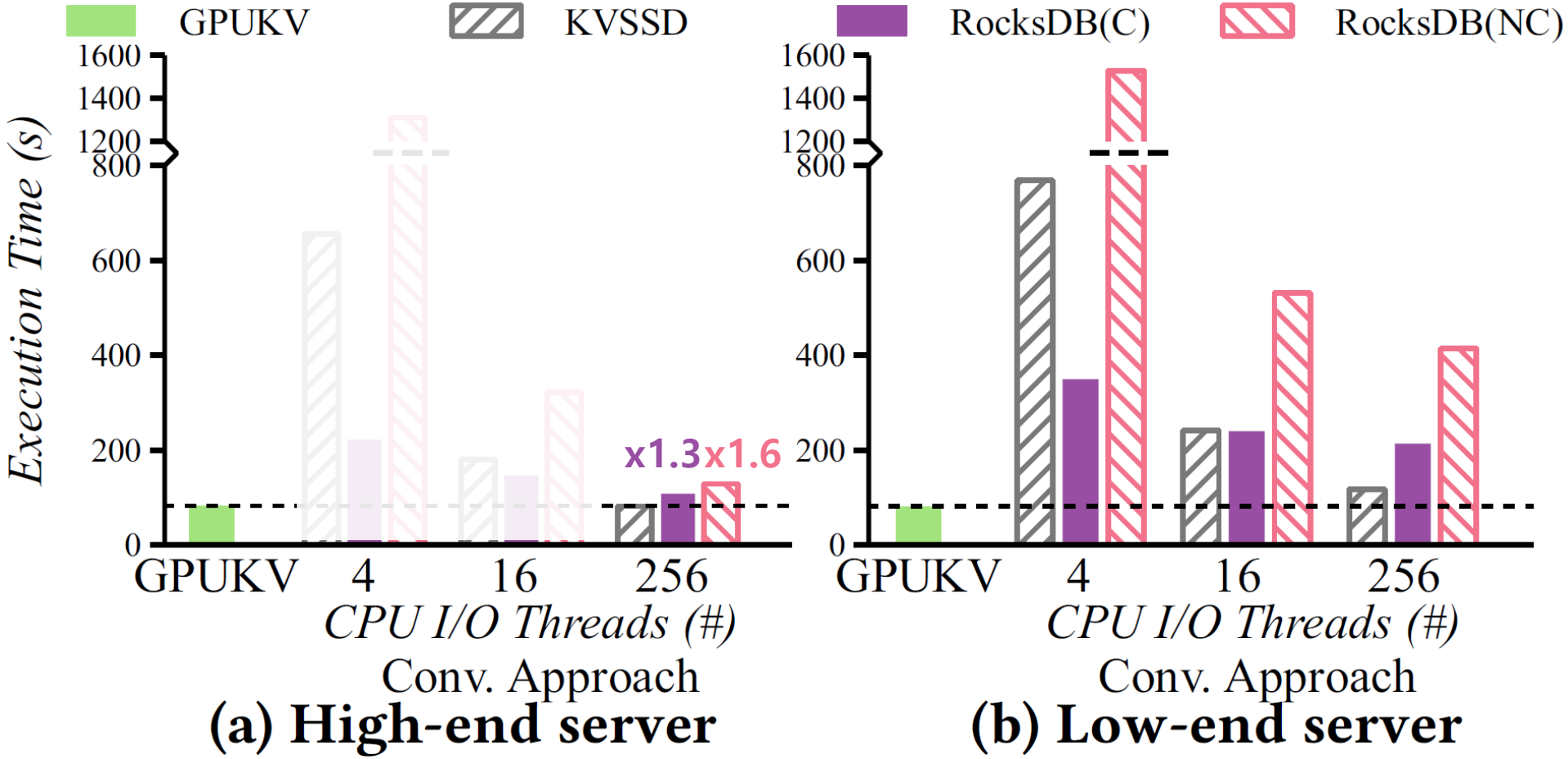
- **CPU-driven GPU computing models**
  - Requires a large number of I/O threads for performance

- **GPUKV enables GPU-driven computing with very little depen dence no the host's CPU**

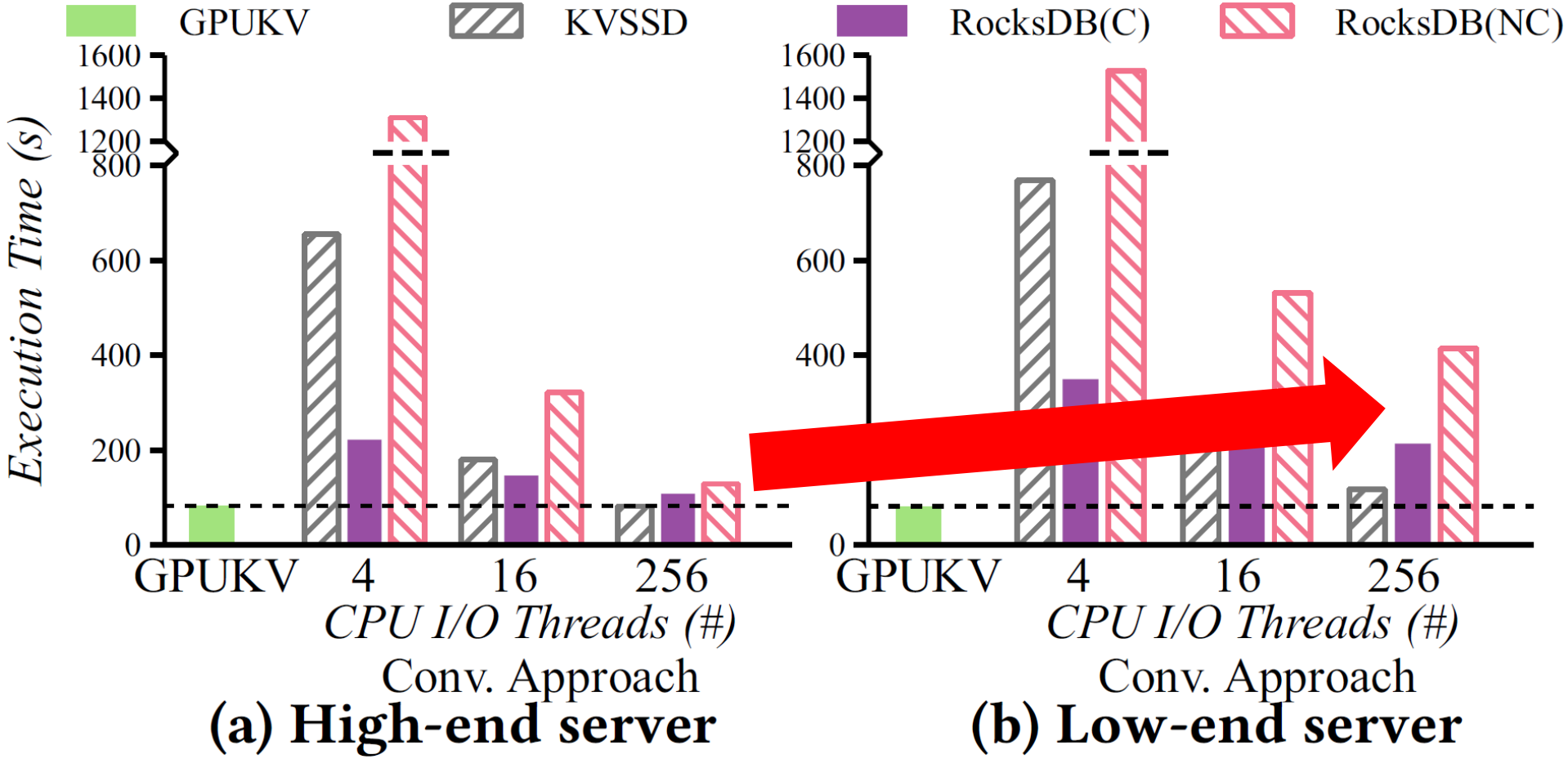
Realistic workload (Streaming - PR)



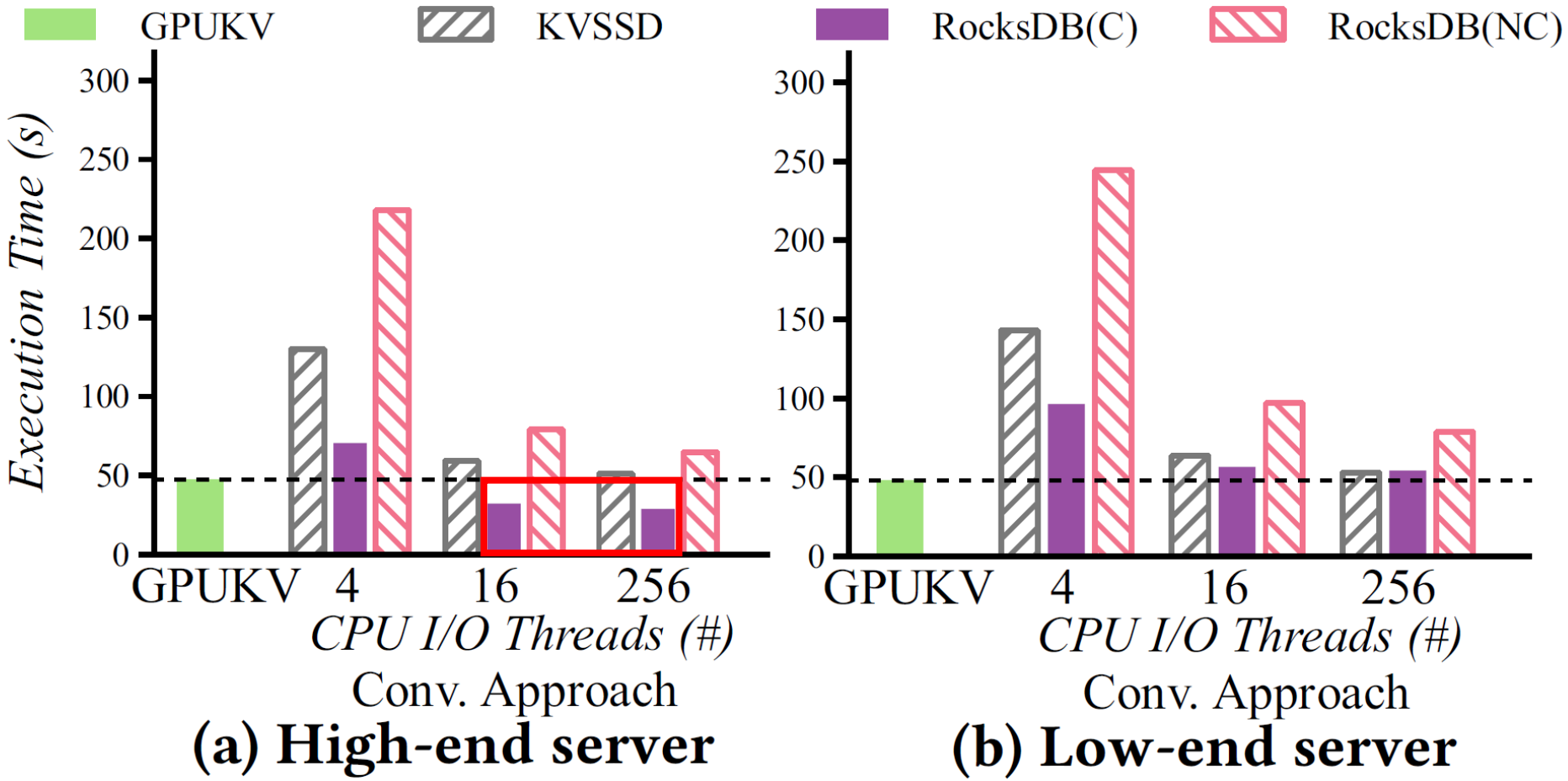
Realistic workload (Streaming - PR)



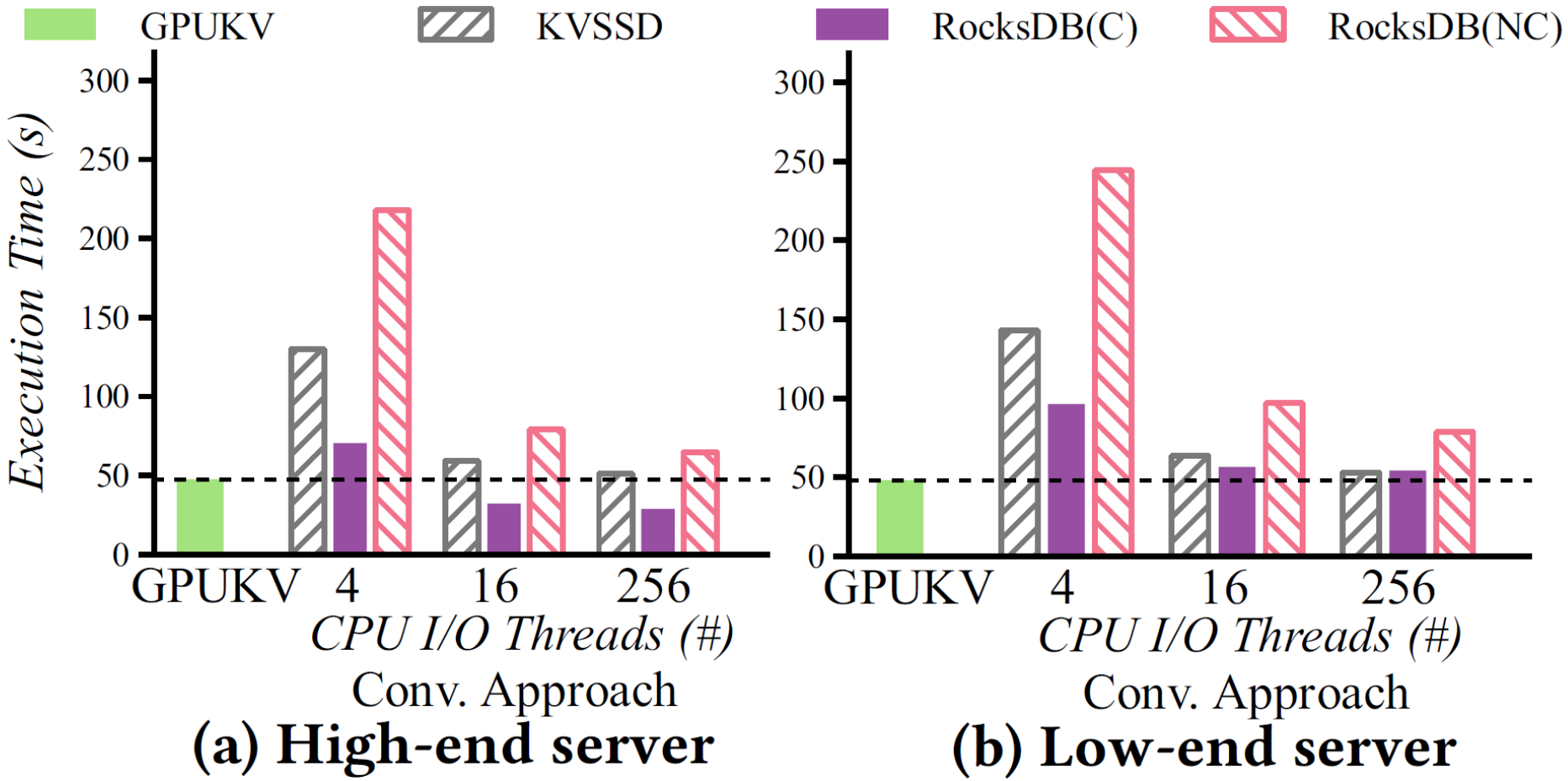
Realistic workload (Streaming - PR)



Realistic workload (Dynamic - BFS)



Realistic workload (Dynamic - BFS)

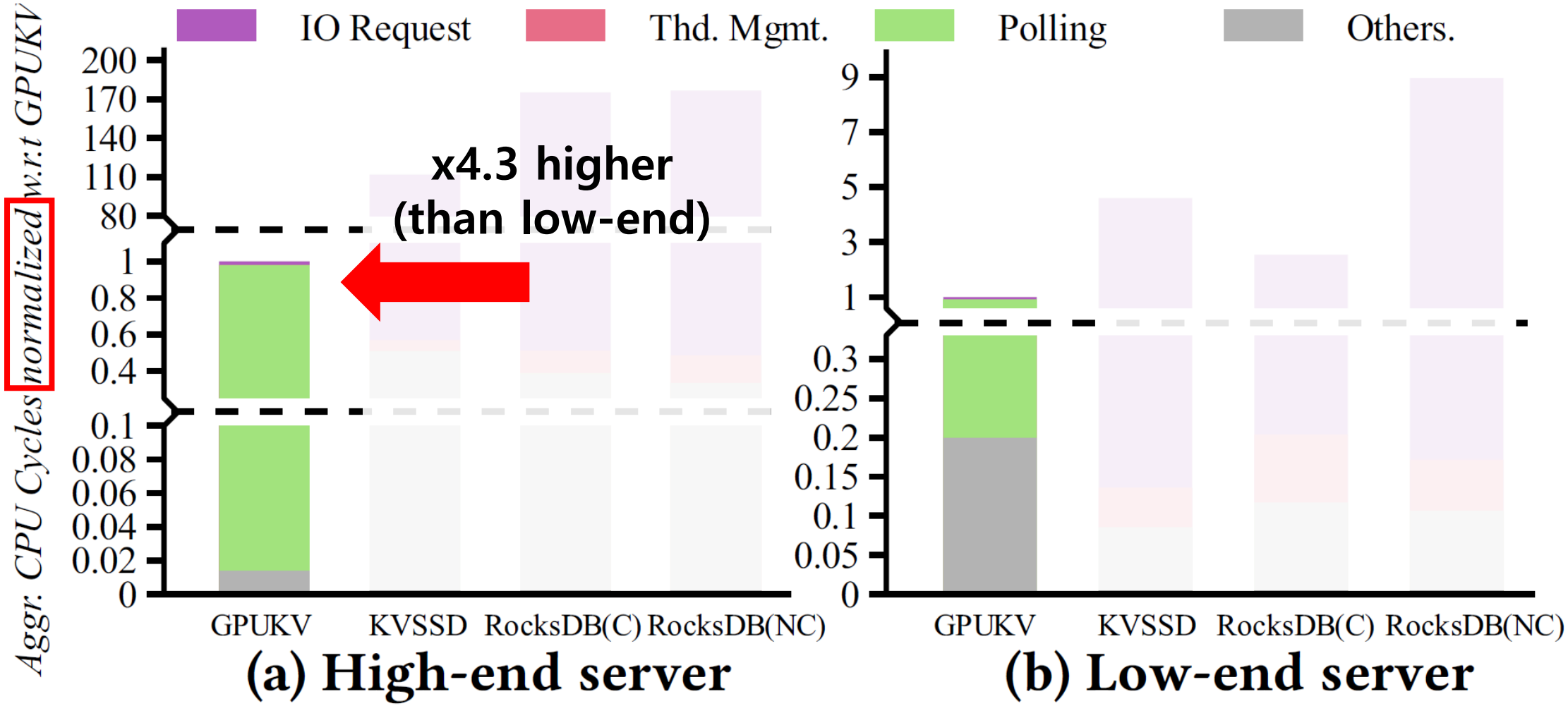


CPU utilization

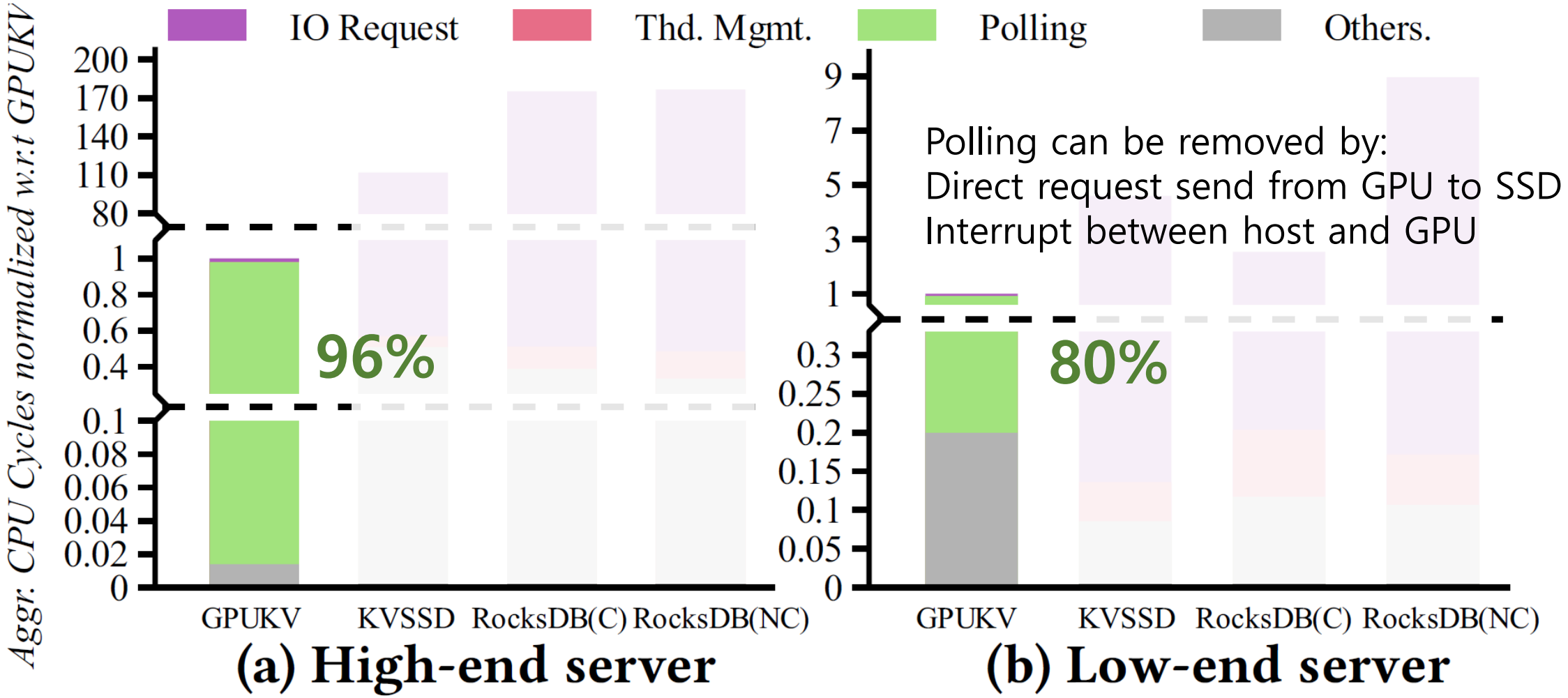
- GPUKV's performance is not affected by the host's CPU performance

- CPU-driven GPU-computing models are affected by the host's CPU performance

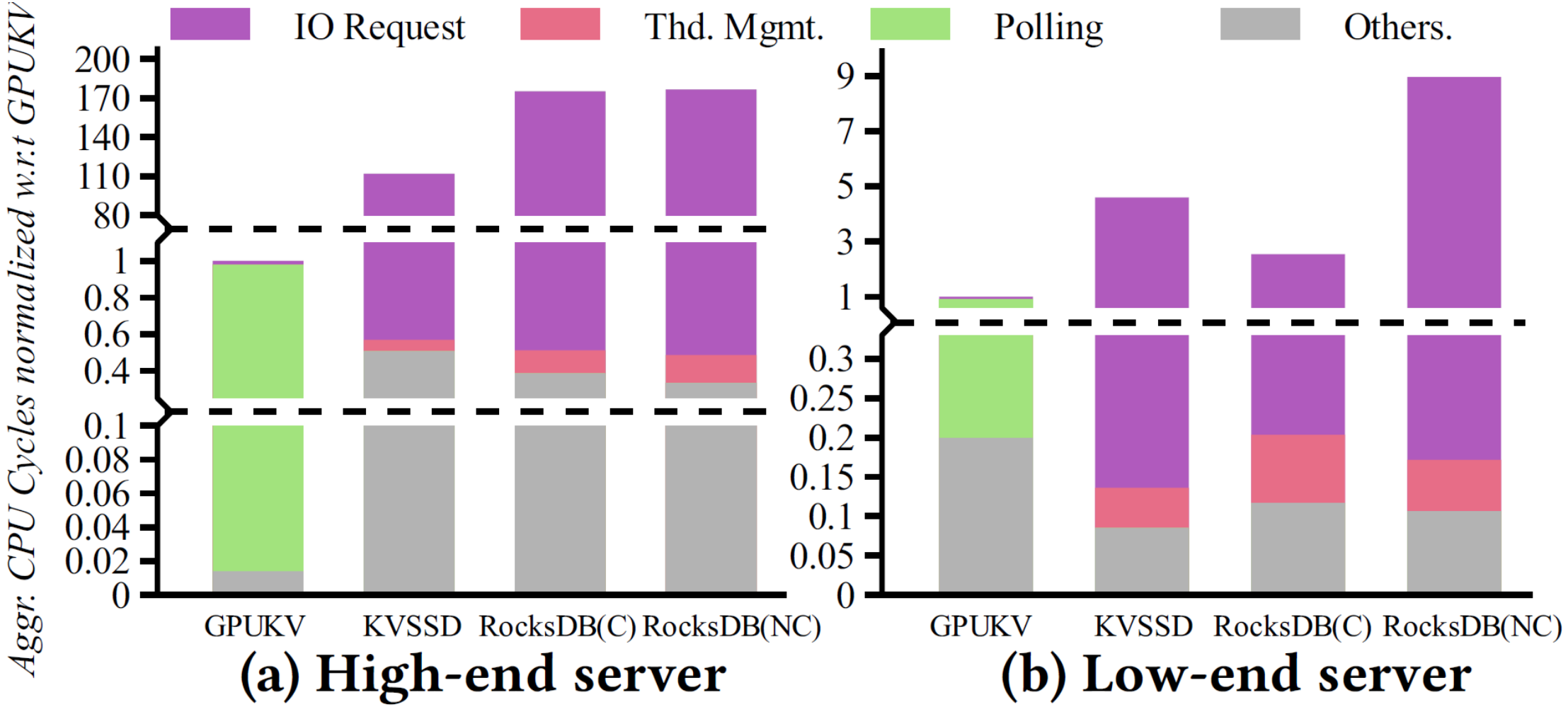
CPU utilization



CPU utilization



CPU utilization



Summary and Conclusion

- ❑ **GPUKV suggests a GPU-driven computing model.**
- ❑ **GPUKV consumes lowest CPU cycles and shows best performance than conventional CPU-driven GPU computing approaches.**
- ❑ **GPUKV can perform the workload fastest while consuming minimal host resource.**

Thank you!
Q & A

Min-Gyo Jung

jmg7173@u.sogang.ac.kr

Sogang University, Seoul, Republic of Korea