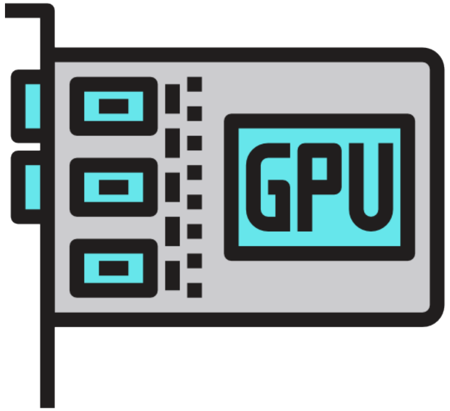
GPUKV: Towards a GPU-Driven Computing on Key-Value SSD

Min-Gyo Jung†, Chang-Gyu Lee†, Donggyu Park†, Sungyong Park†, Youngjae Kim†
Jungki Noh‡, Woosuk Chung‡, Kyoung Park‡

†Sogang University, Seoul, Republic of Korea, ‡SK hynix

Why is Key-Value Store + GPU important?



+

cassandra

redis

RocksDB

DynamoDB

amazon web services

GPU

Massive Parallelism

Boost data-intensive applications

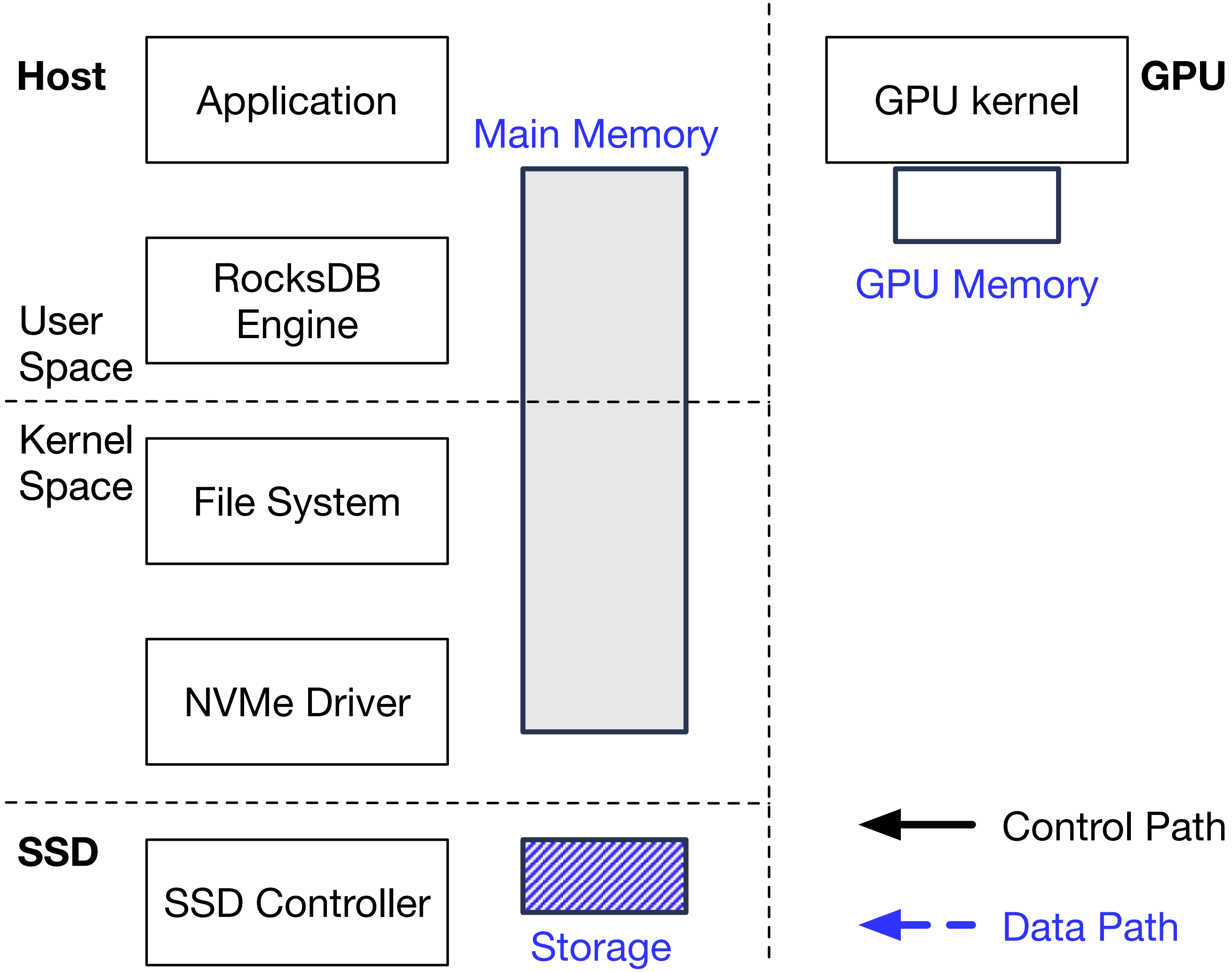
Key-Value Store

Good to store unstructured data

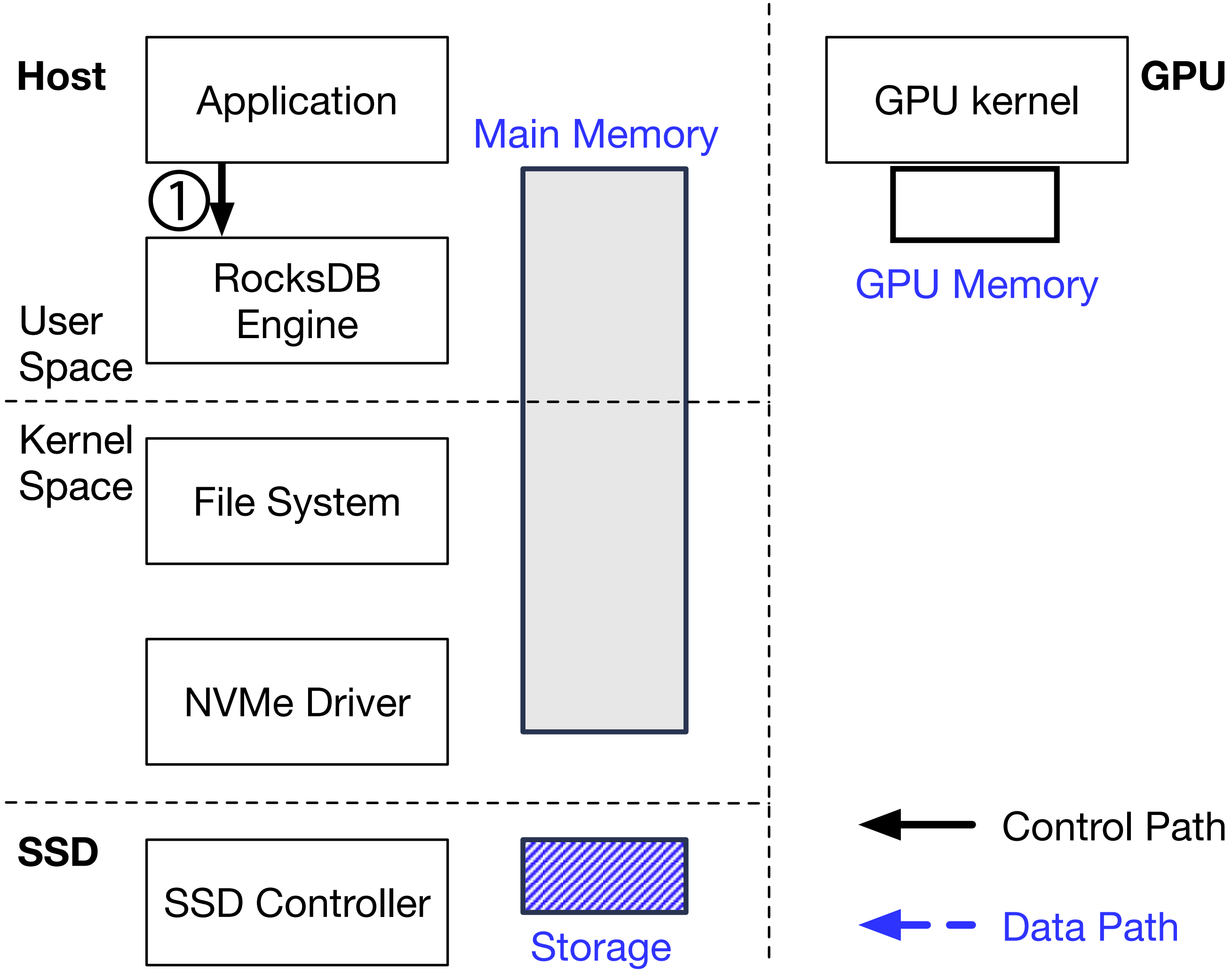
Widely used for storing big data

More **powerful performance** and **usability** for data-intensive applications
e.g. Map-Reduce, Graph Processing, Data Analysis …

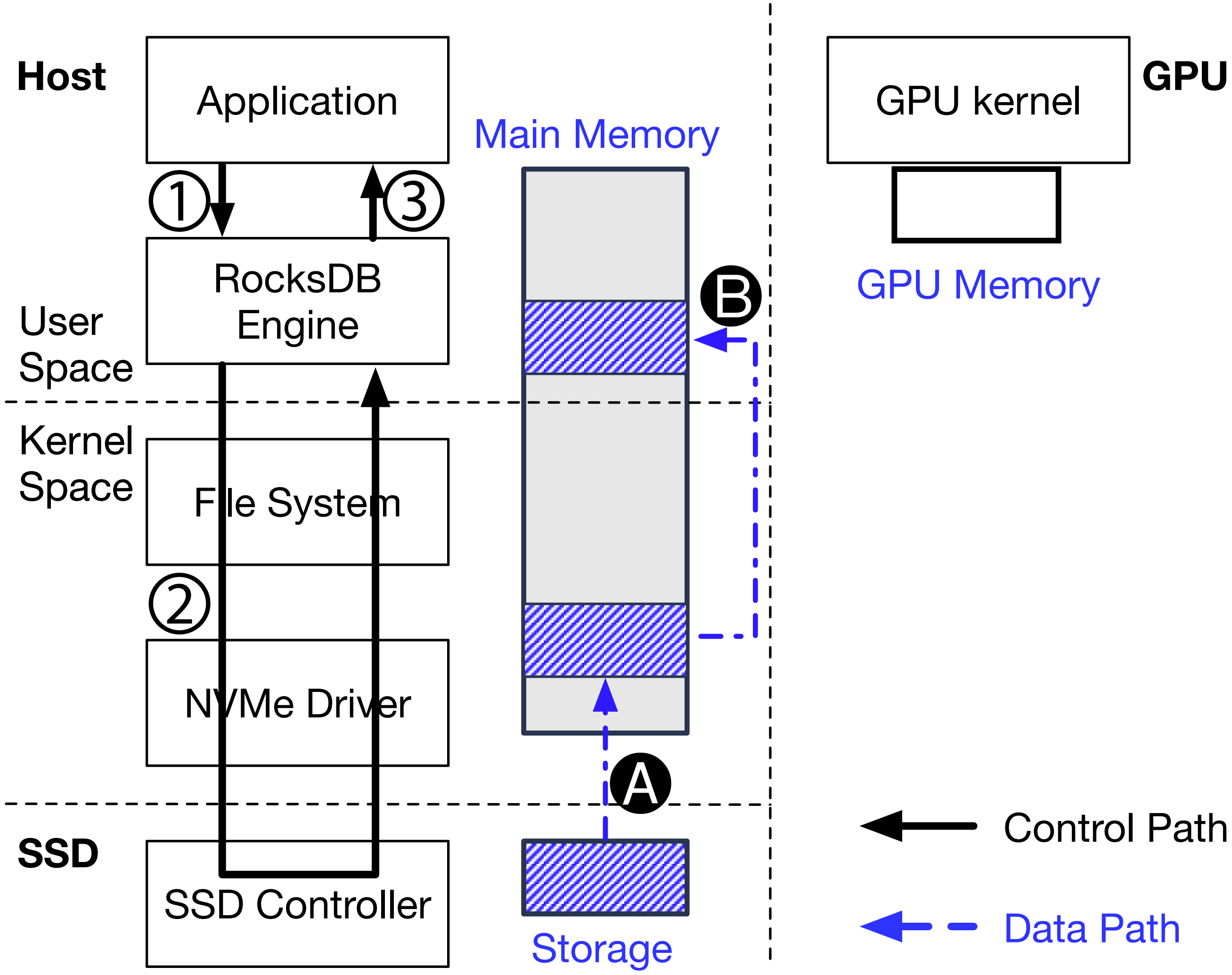
Data Transfer Flow from Key-Value Store to GPU



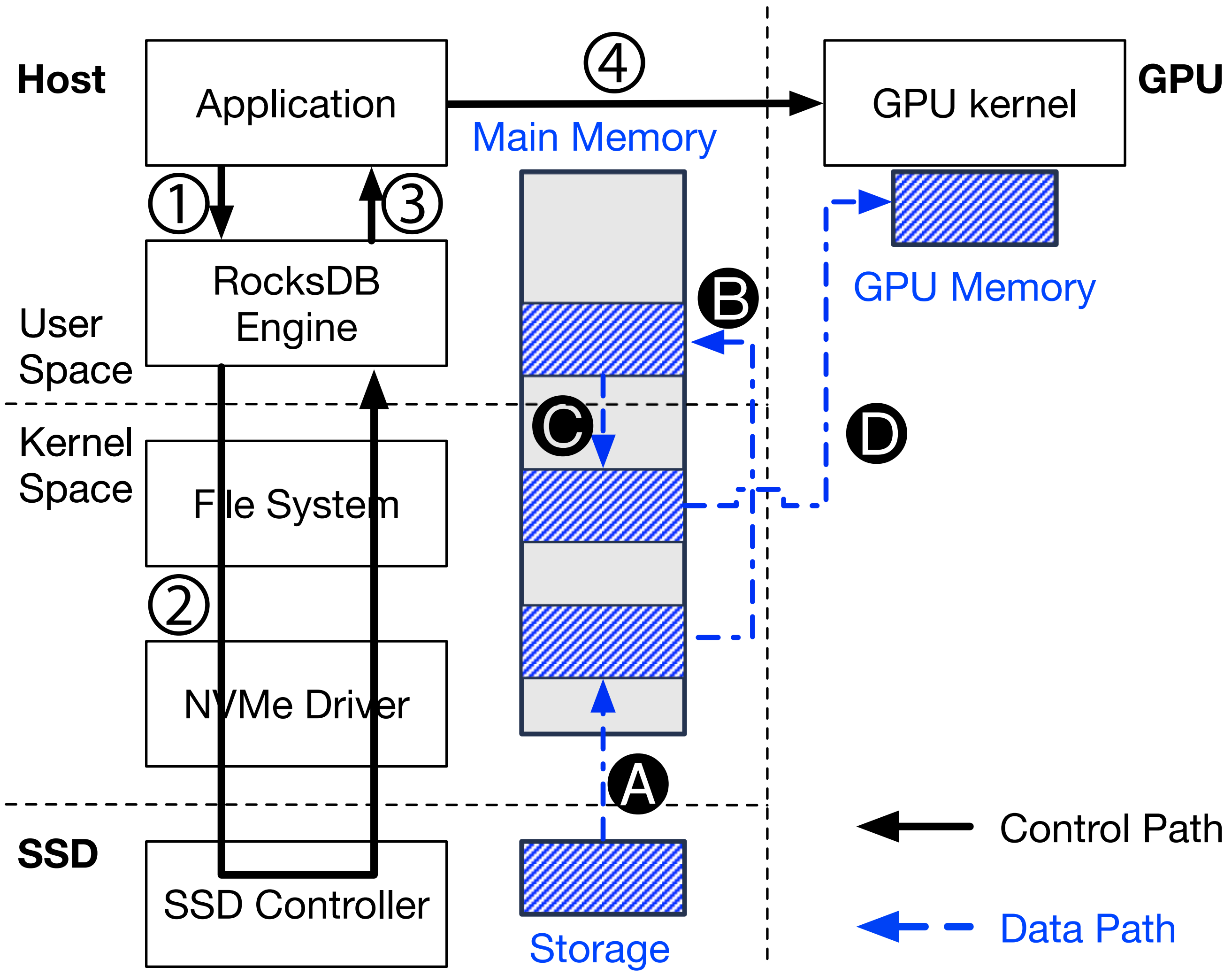
Data Transfer Flow from Key-Value Store to GPU



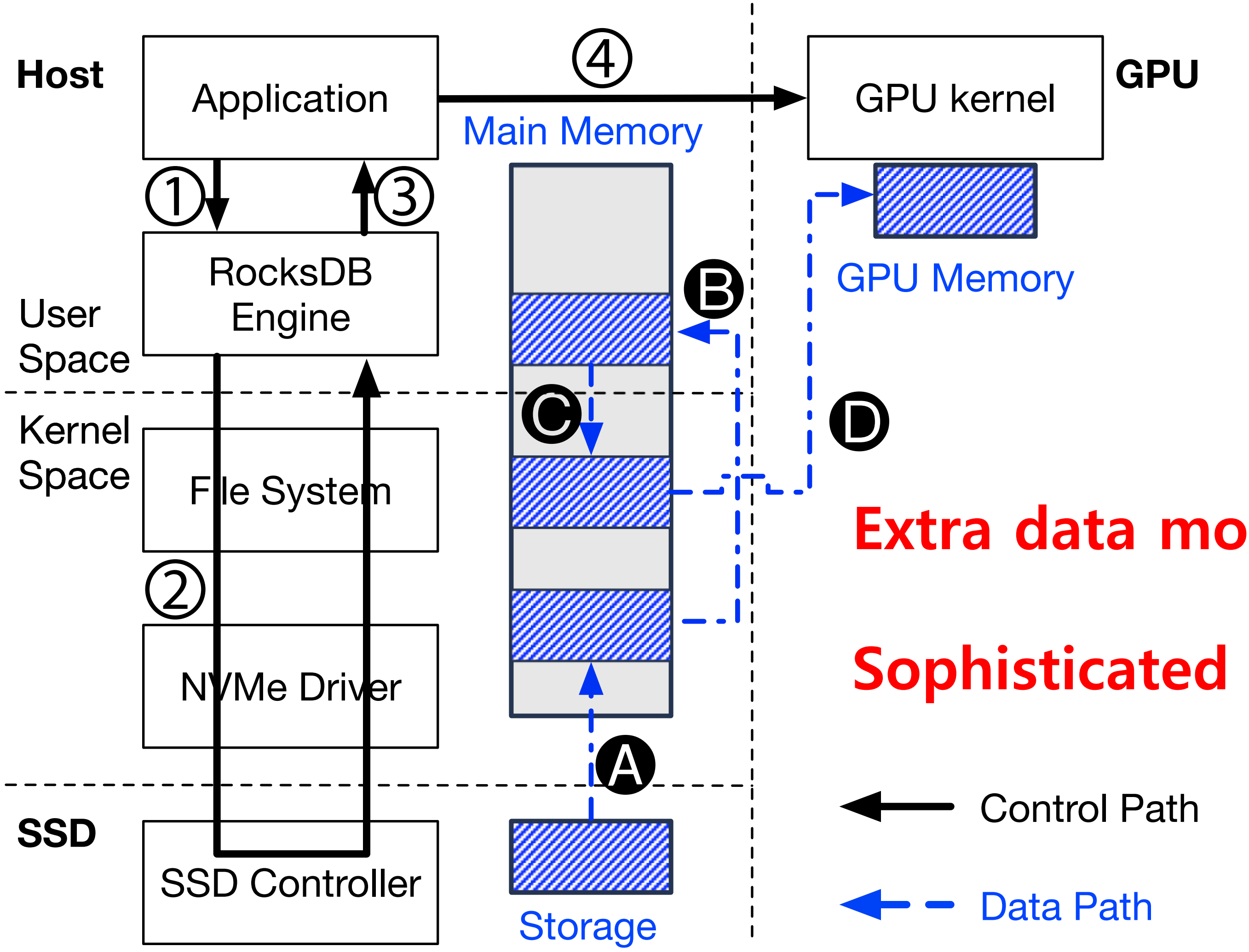
Data Transfer Flow from Key-Value Store to GPU



Data Transfer Flow from Key-Value Store to GPU



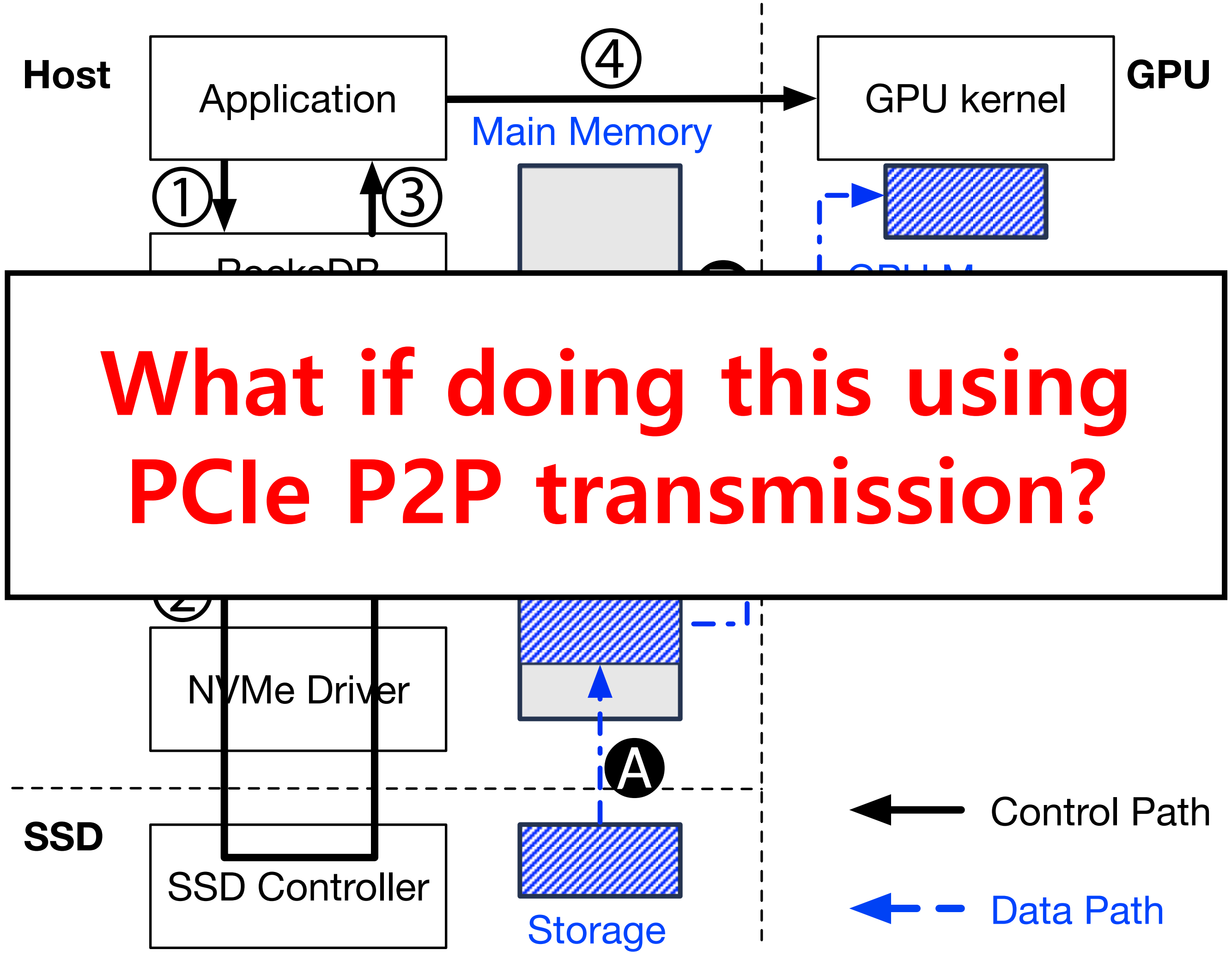
Data Transfer Flow from Key-Value Store to GPU



Extra data movement

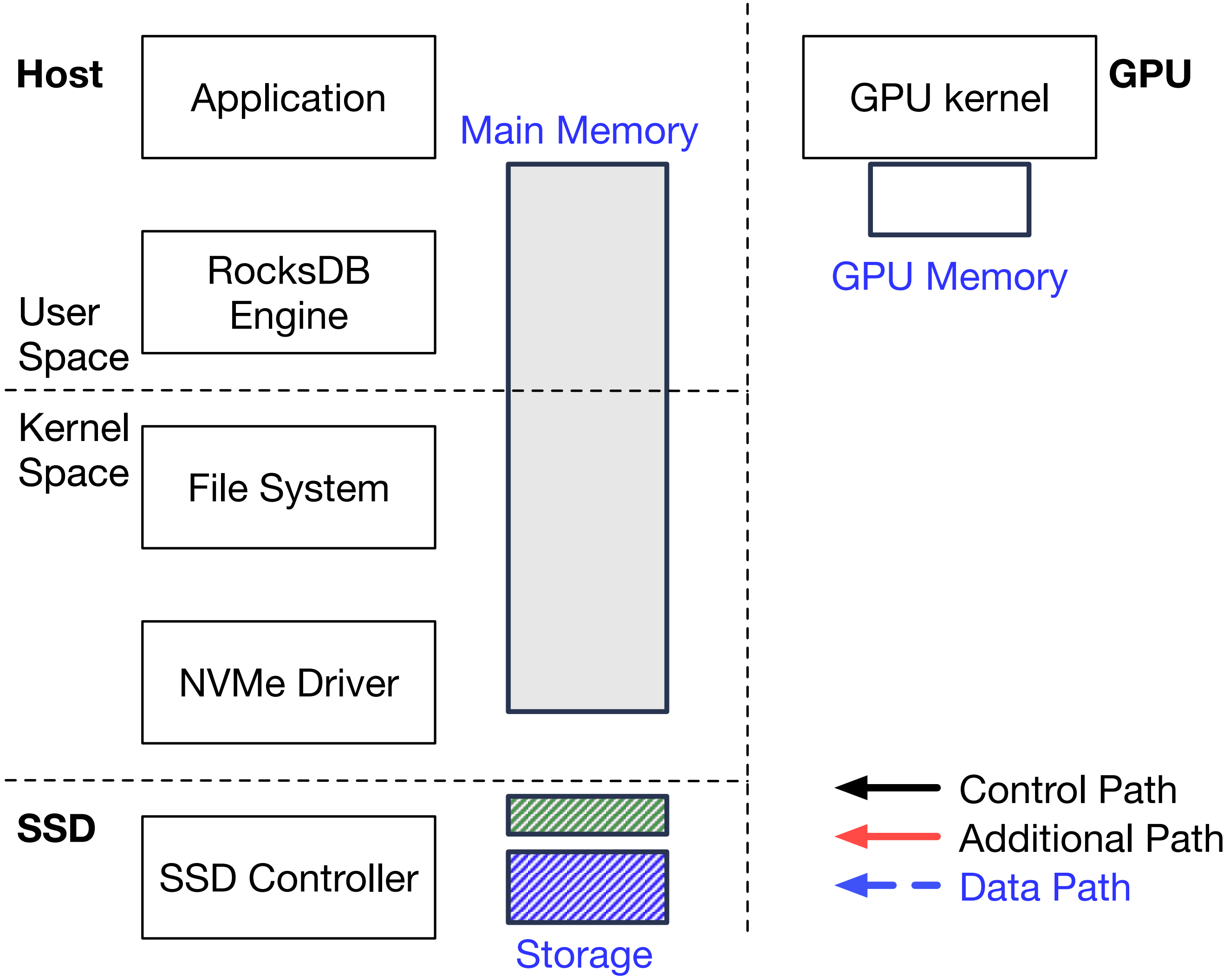
Sophisticated control path

Data Transfer Flow from Key-Value Store to GPU

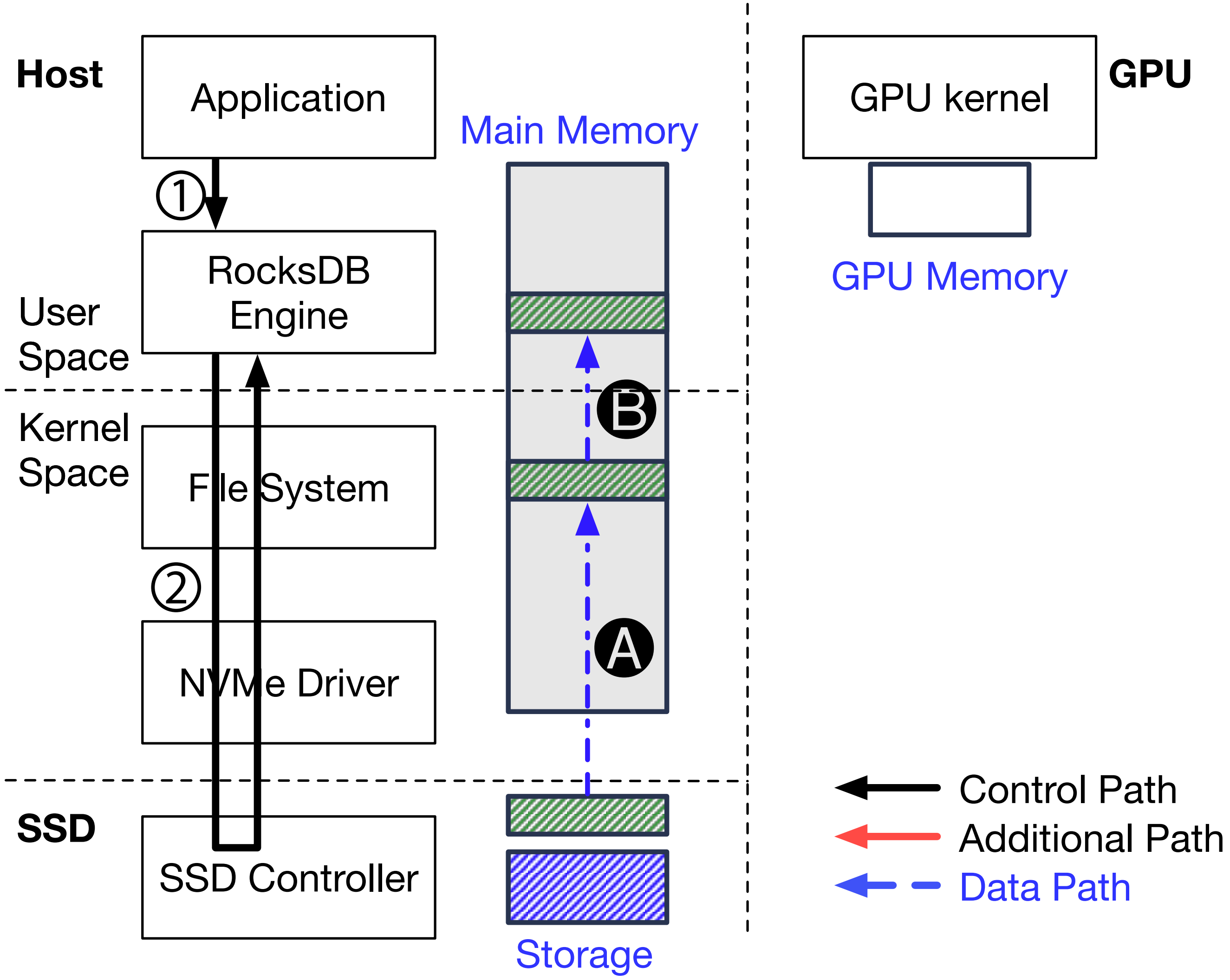


What if doing this using PCIe P2P transmission?

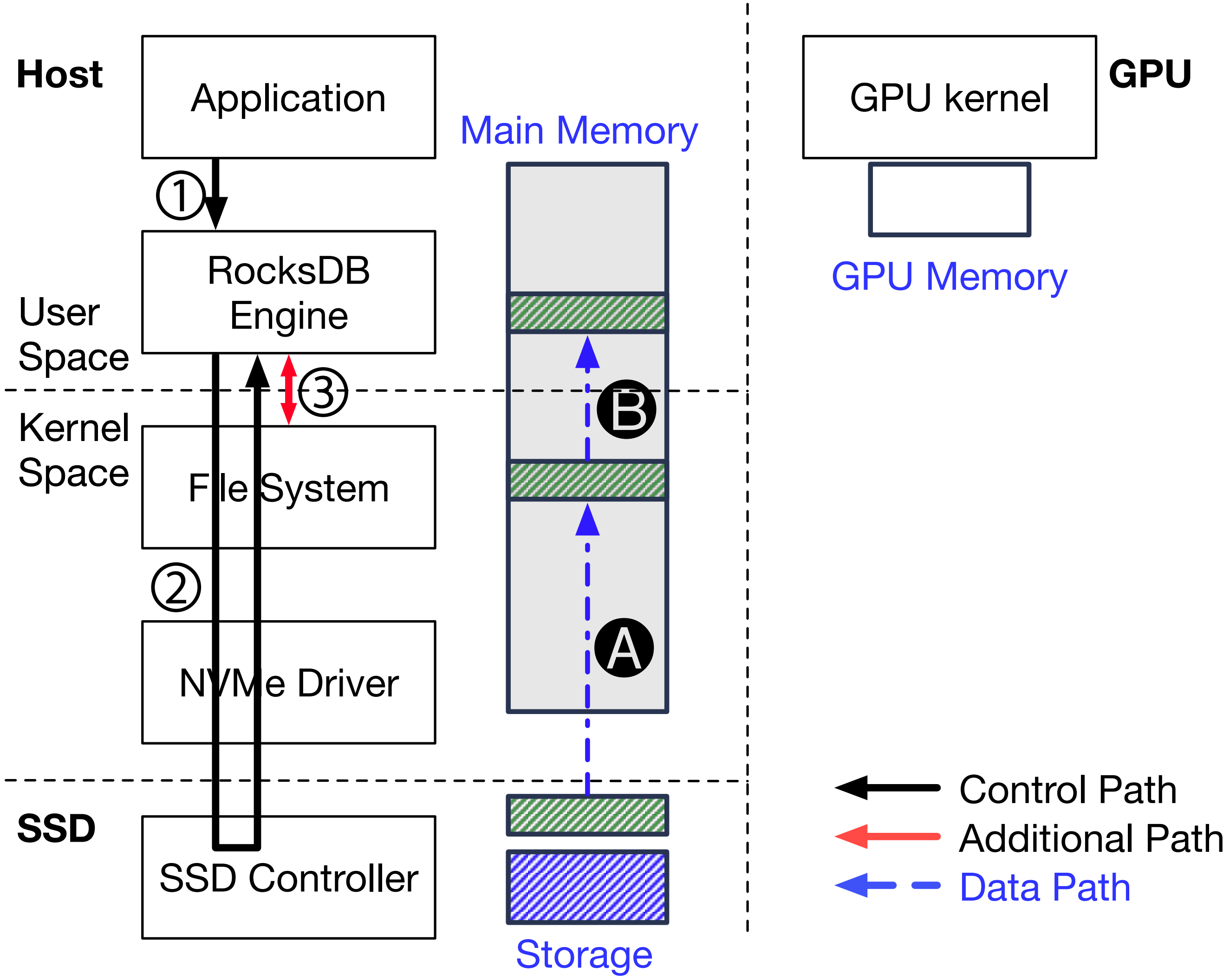
Data Transfer Flow when transferring using P2P



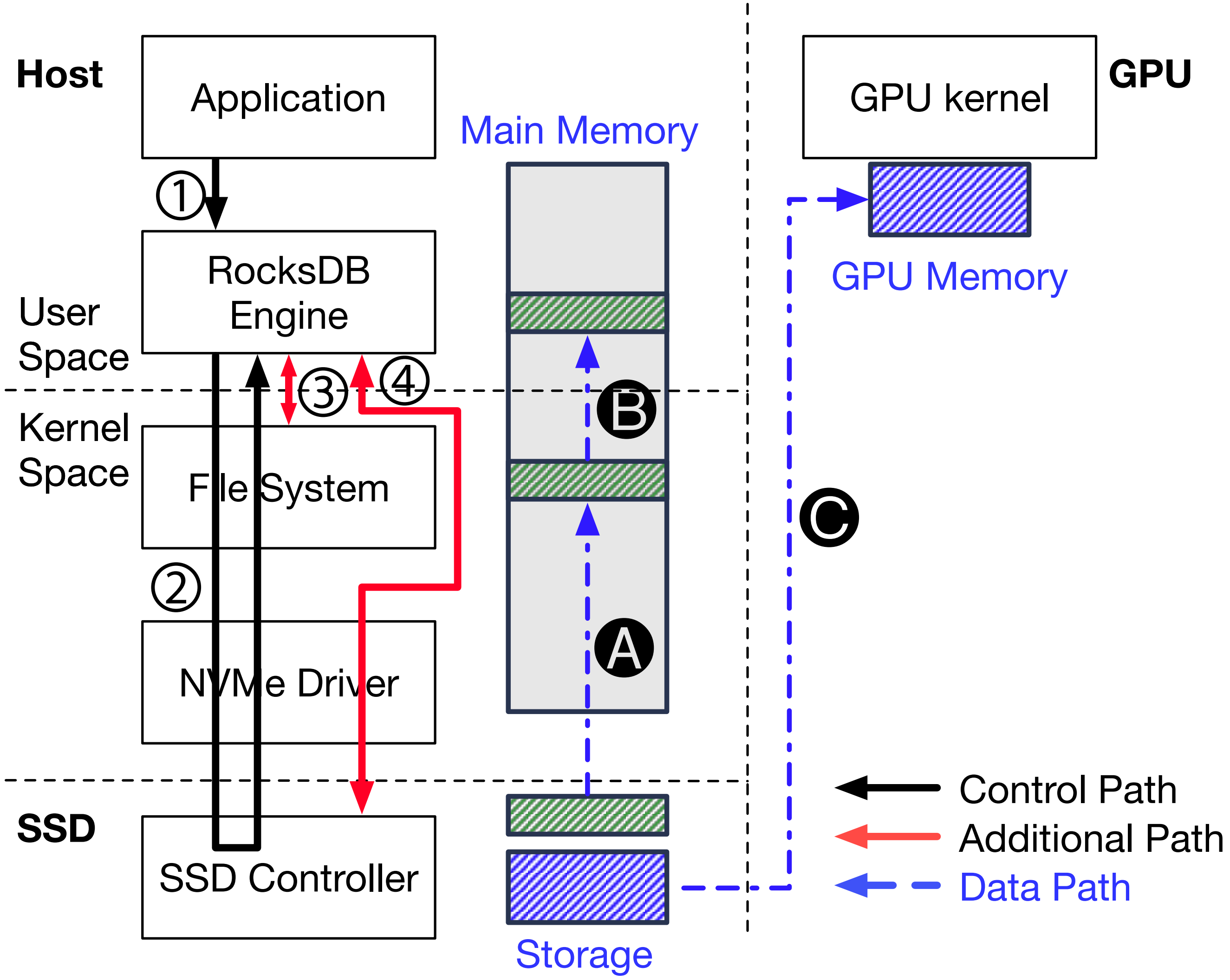
Data Transfer Flow when transferring using P2P



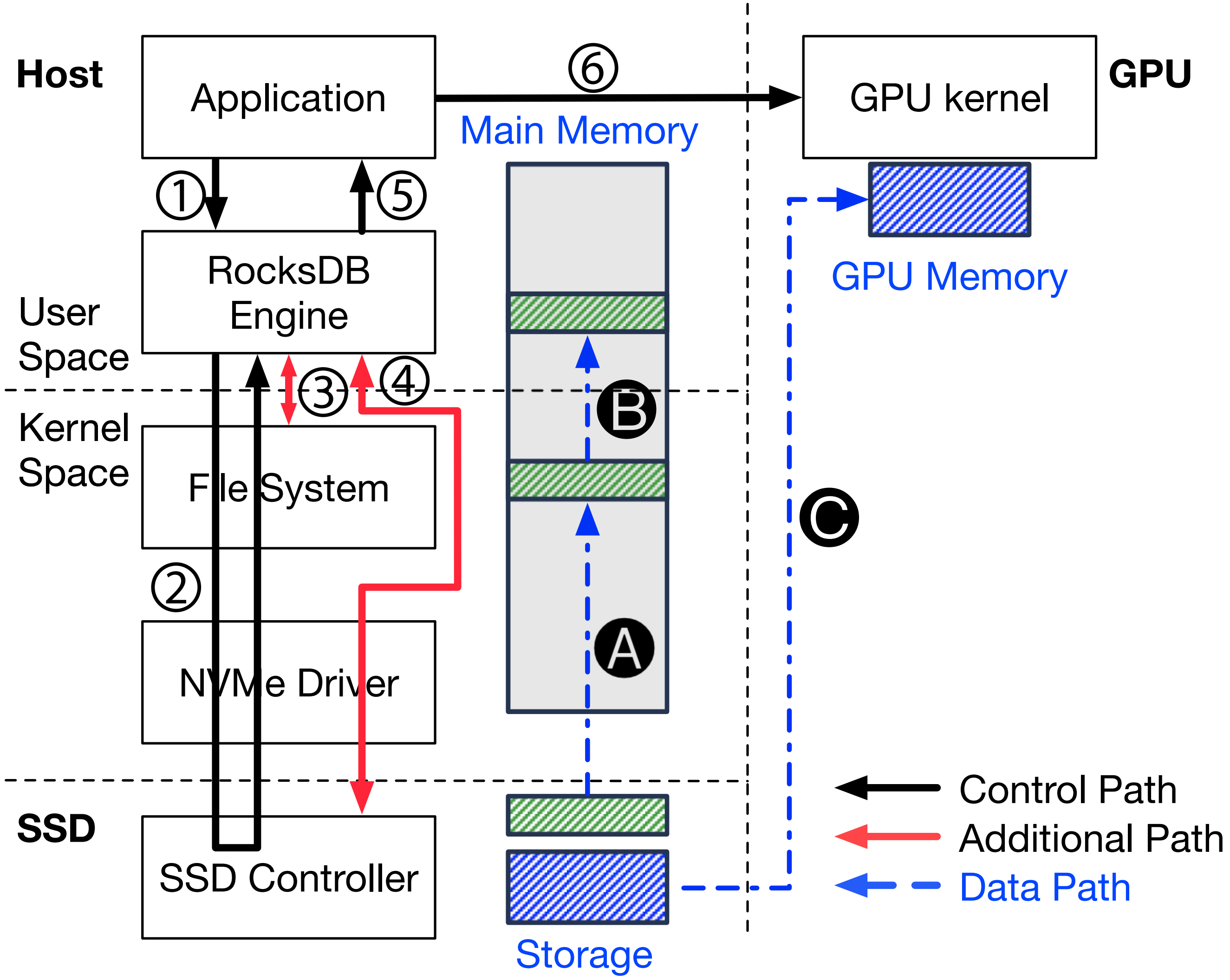
Data Transfer Flow when transferring using P2P



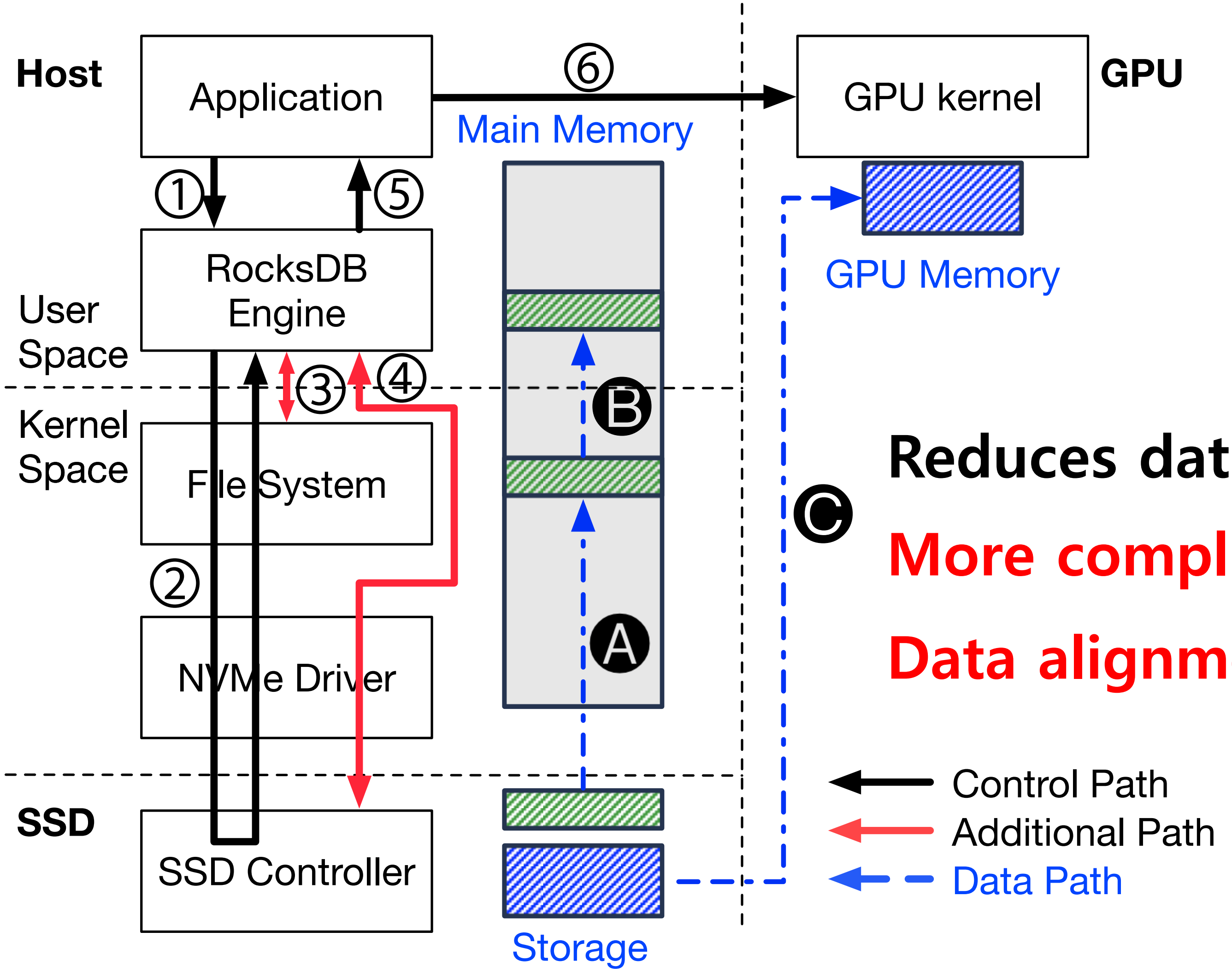
Data Transfer Flow when transferring using P2P



Data Transfer Flow when transferring using P2P



Data Transfer Flow when transferring using P2P



Reduces data movement

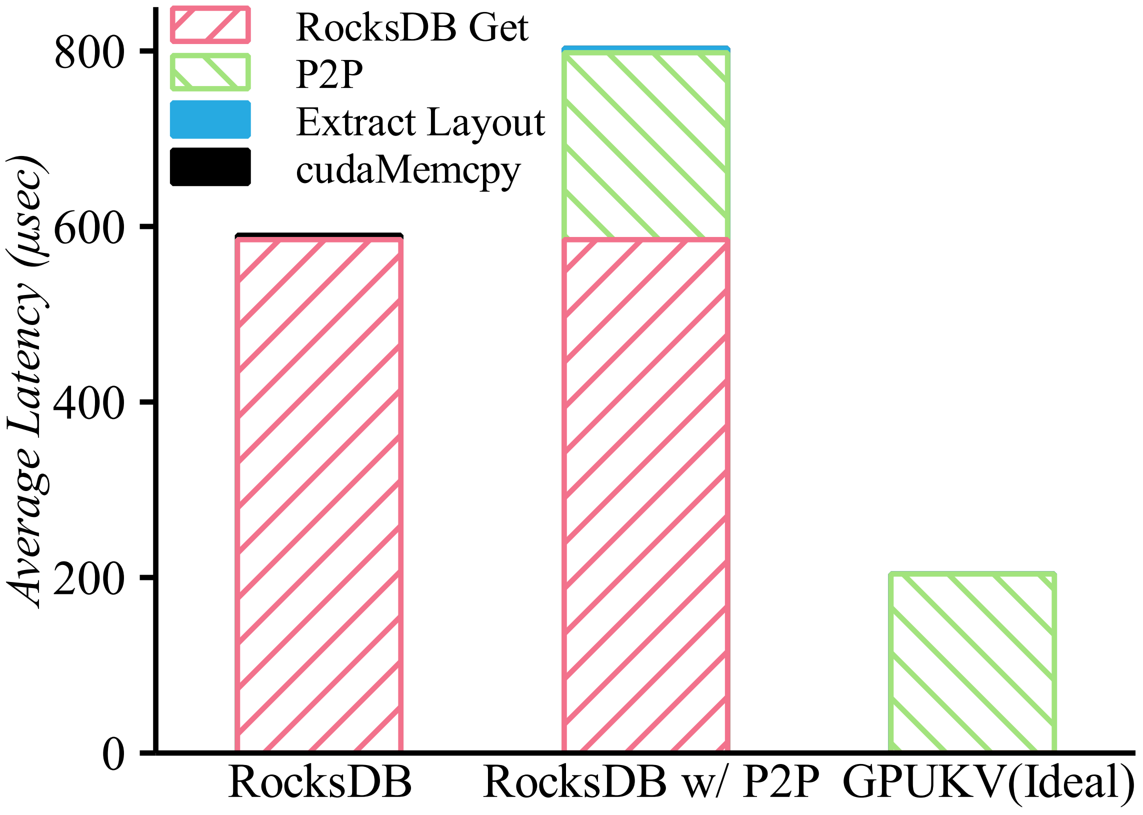
More complicated control path

Data alignment for P2P

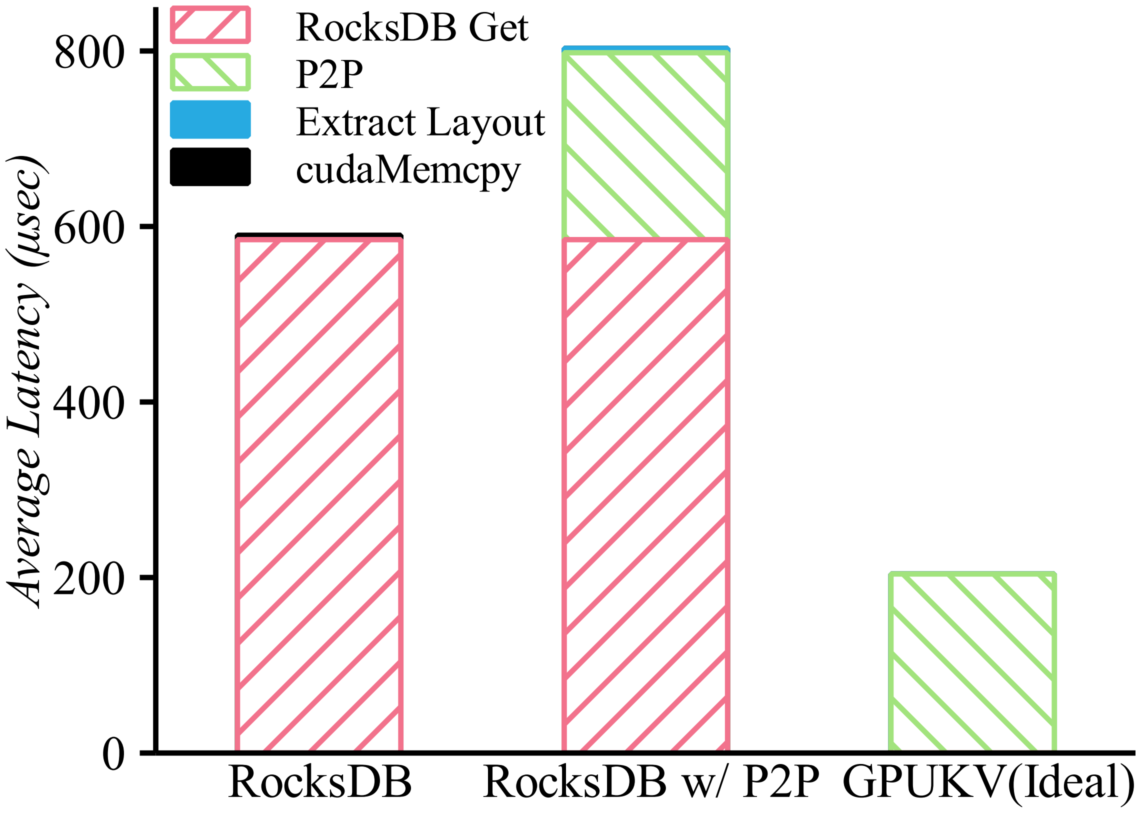
What does GPUKV suppose to do?

- **GPU-driven computing model**
  - GPU issues IO bypassing host architectures
- **Reduce data movement using PCIe P2P**
  - Data storage ↔ Accelerator (GPU)
  - Save wasting memory bus bandwidth
- **Simple control path**
  - Implementing Key-Value store at SSD, reduce complex control paths

Data Transfer Latency Breakdown

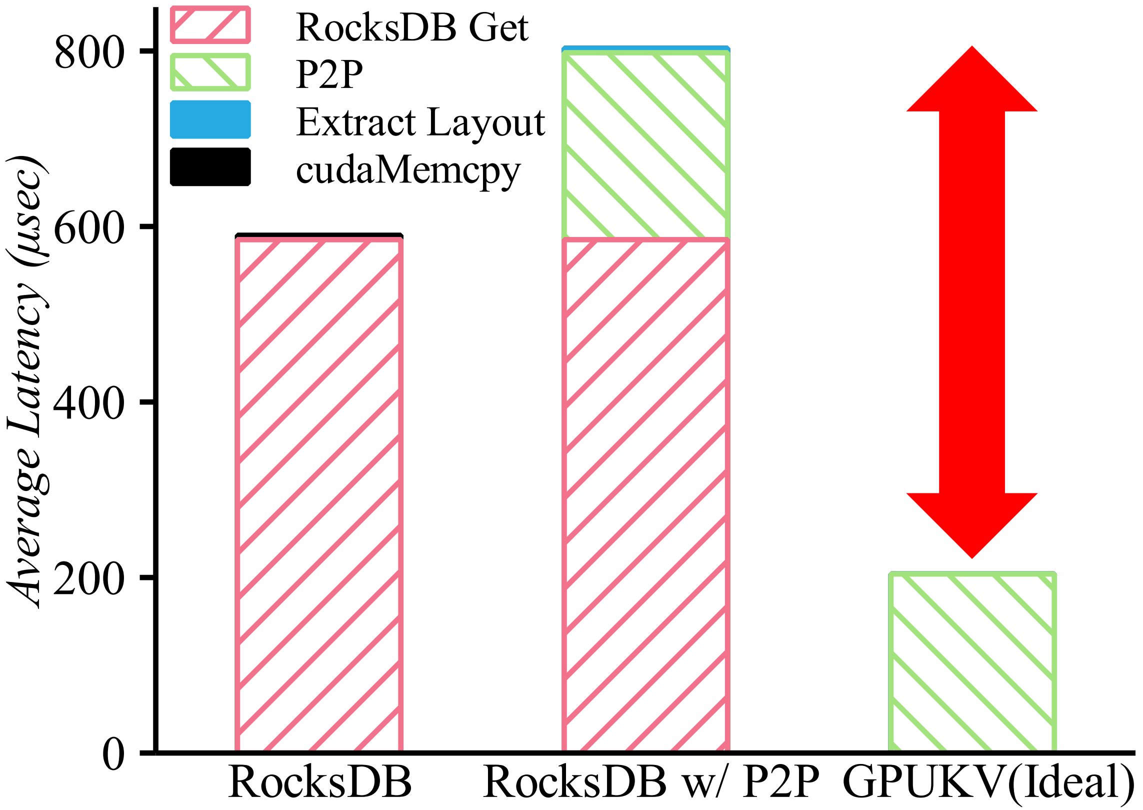


Data Transfer Latency Breakdown



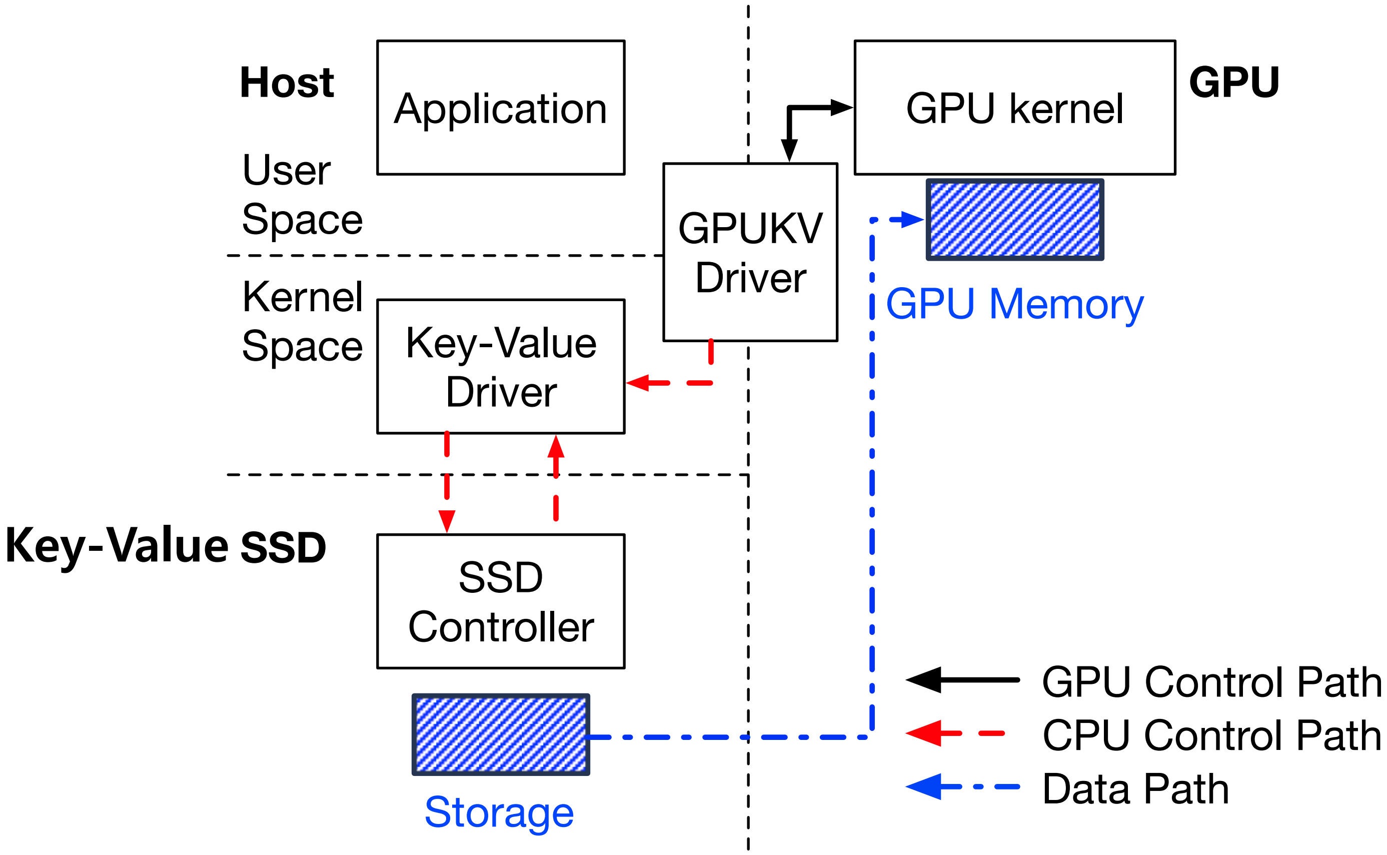
In ideal case, **GPUKV** only needs data transfer latency

Data Transfer Latency Breakdown



GPU-driven Computing is necessary!

GPUKV's Data Transfer Flow



No Redundant data copy

Simple and short Control Path

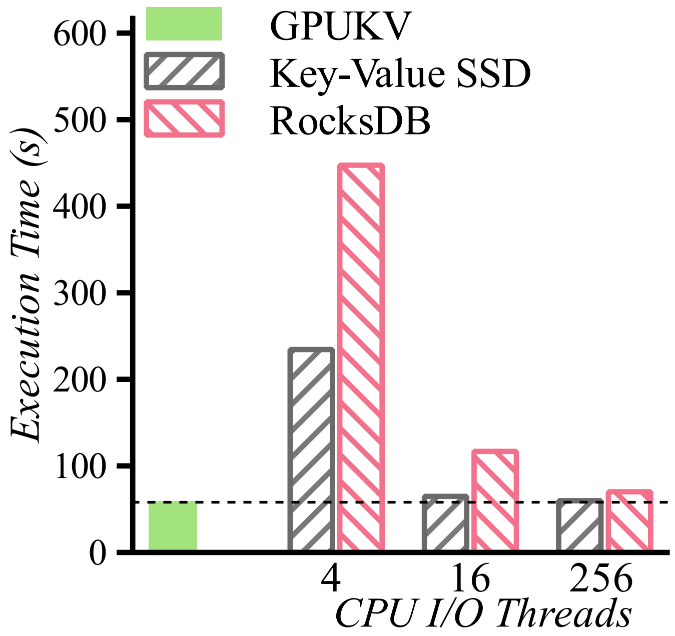
Data request from GPU itself

Preliminary Results: Synthetic Workloads

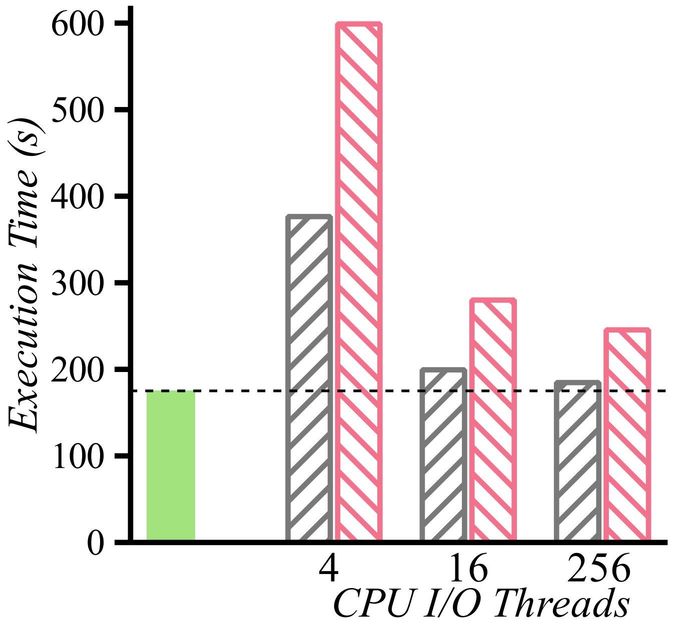
- **Streaming workload** ($W_{streaming}$)
  - Predictable data access pattern
  - The next dataset needed by GPU kernel can be prefetched

- **Dynamic workload** ($W_{dynamic}$)
  - Unpredictable data access pattern
  - The next dataset GPU kernel needs cannot be prefetched
  - Only can be loaded when current GPU kernel finishes.

Preliminary Results: Synthetic Workloads

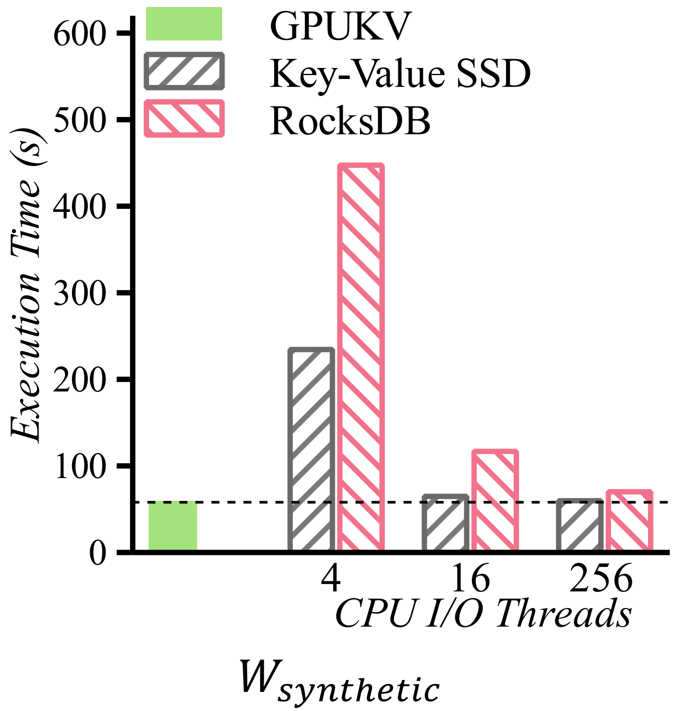


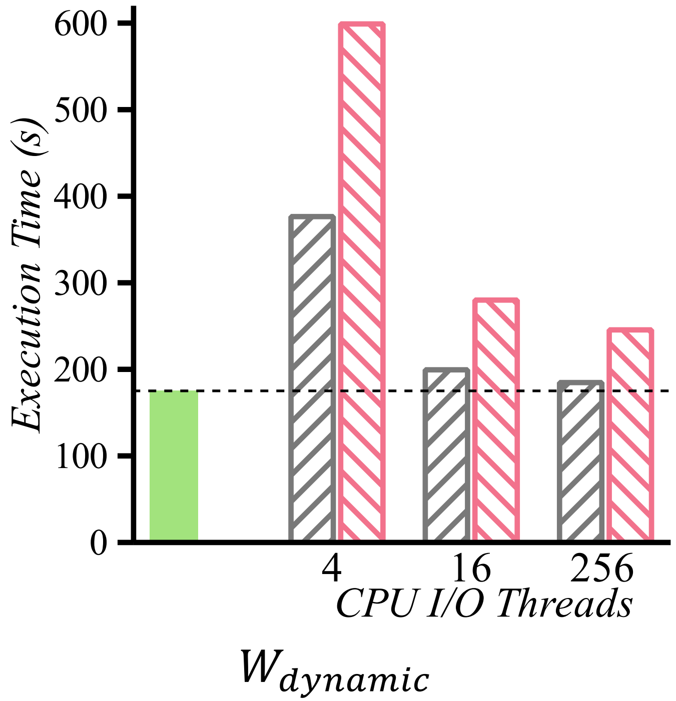
$W_{synthetic}$



$W_{dynamic}$

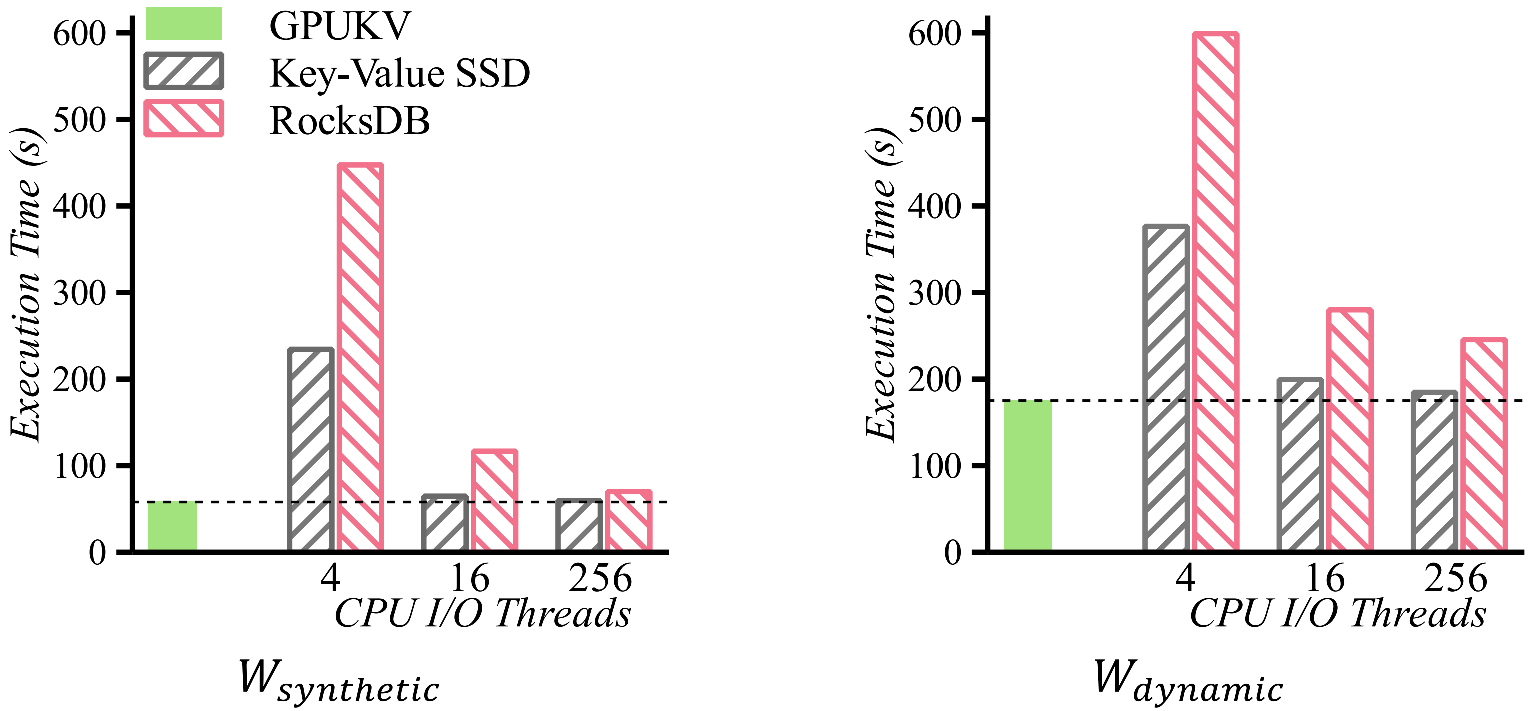
Preliminary Results: Synthetic Workloads





Conventional way: Need powerful host resources

Preliminary Results: Synthetic Workloads



$W_{synthetic}$

$W_{dynamic}$

Our approach – GPUKV:

Always shows best performance with only 1 I/O thread

Barely requires host resource

GPUKV: Towards a GPU-Driven Computing on Key-Value SSD

Min-Gyo Jung†, Chang-Gyu Lee†, Donggyu Park†, Sungyong Park†, Youngjae Kim†
Jungki Noh‡, Woosuk Chung‡, Kyoung Park‡

†Sogang University, Seoul, Republic of Korea, ‡SK hynix

jmg7173@u.sogang.ac.kr

서강대학교
SOGANG UNIVERSITY

SK hynix