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
Background & Motivation

The Era of Data Overflow

Annual Size of the GlobalDatasphere

Source: Data Age 2025, sponsored by Seagate with data from IDC Global DataSphere, Nov 2018
The Era of Data Overflow

How can we handle data?

Annual Size of the Global Datasphere

Source: Data Age 2025, sponsored by Seagate with data from IDC Global DataSphere, Nov 2018
Background & Motivation

Data-intensive applications
Background & Motivation

Data-intensive applications

Graph Processing
Data-intensive applications

Graph Processing

Scientific Simulations
Background & Motivation

Data-intensive applications

Graph Processing
Scientific Simulations
Deep Learning
Background & Motivation

Data-intensive applications

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

Graph Processing
Scientific Simulations
Deep Learning
Background & Motivation

Data-intensive applications

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

Background & Motivation

But always need **INTERVENTION OF CPU**

A **CPU-driven** GPU computing model

Massive Parallelism

Boost data-intensive applications
When Data-intensive applications meet GPUs

Background & Motivation
When Data-intensive applications meet GPUs
Background & Motivation

When Data-intensive applications meet GPUs
When Data-intensive applications meet GPUs
When Data-intensive applications meet GPUs

Background & Motivation

Extra data movement

Sophisticated control path
When Data-intensive applications meet GPUs

Background & Motivation

What if doing this using PCIe P2P transmission?
Background & Motivation

PCle P2P Communication

- CPU
- PCIe Interconnect
- PCIe Device
- PCIe Device

Direct Data Transfer

- NVIDIA GPUDirect
- RDMA
- AMD’s DirectGMA
PCle P2P Communication

- GPUs are PCIe Devices
- NVMe SSDs use PCIe Interconnect

Related works: SPIN [USENIX ATC `17], NVMMU [PACT `15], Morpheus [SIGARCH `16], HippogriiffDB [VLDB `16], ...
Data Transfer using PCIe P2P
Background & Motivation

Data Transfer using PCIe P2P
Data Transfer using PCIe P2P

Background & Motivation
Data Transfer using PCIe P2P

Background & Motivation
Data Transfer using PCIe P2P
Data Transfer using PCIe P2P

Background & Motivation

Reduces data movement

More complicated control path

Data alignment for P2P
Does it really work?
Does it really work?

![Bar chart showing data pulling latency comparison between RocksDB, RocksDB with P2P, and GPUKV (ideal).]
Does it really work?

Background & Motivation

![Additional Overhead Graph](image.png)

- RocksDB
- RocksDB w/ P2P
- GPUKV (Ideal)

- RocksDB Get
- P2P
- Extract Layout
- cudaMemcpy

Data Pulling Latency (μsec)
Does it really work?

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

A **GPU-driven** GPU computing model

KVSSD
Our Contributions

1. Key-Value Store Abstraction to GPU

2. Key-Value SSD

3. P2P Communication between KVSSD and GPU
Design & Implementation

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

- **gpukv_get** (key, value_size, offset)
- Issue KV operation using RPC
- Enqueue KV operation

### Design & Implementation

**Shared Memory**

**GPU**

**User Space**

**Kernel Space**

- GPU thread
- Host Issue thread
- Host Completion thread
Functional Flow of GPUKV

- `gpukv_get` (key, value_size, offset)
- Issue KV operation using RPC
- Shared Memory
  - Polling
  - GPU
- User Space
  - `kv_p2p_get` (kv_operations)
  - Enqueue KV operation
- Kernel Space
  - `nvme_submit_cmd`
  - Generate KV Commands
  - GPU thread
  - Host Issue thread
  - Host Completion thread
Functional Flow of GPUKV

1. GPUKV_get (key, value_size, offset)
2. Issue KV operation using RPC
3. Polling
4. Complete

User Space

1. kv_p2p_get (kv_operations)
2. Enqueue KV operation

Kernel Space

1. RPC Ack
2. nvme_submit_cmd
3. Generate KV Commands

GPU

1. Interrupt
   - Waked up when command completed
2. GPU thread
3. Host Issue thread
4. Host Completion thread
Functional Flow of GPUKV

- **Complete**
  - Polling
  - Issue KV operation using RPC
    - gpu_kv_get (key, value_size, offset)

- **GPU**
  - kv_p2p_get (kv_operations)
  - Enqueue KV operation

- **User Space**
  - RPC Ack
  - Generate KV Commands
    - nvme_submit_cmd

- **Kernel Space**
  - (Interrupt)
    - Waked up when command completed
  - GPU thread
    - Host Issue thread
  - Host Completion thread

Design & Implementation
Evaluation

**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 **directI/O**.
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)

Evaluation

(a) High-end server

(b) Low-end server
Synthetic workload (Streaming)

Evaluation

Lower is Better!

(a) High-end server

(b) Low-end server
Synthetic workload (Streaming)

(a) High-end server

(b) Low-end server

Lower is Better!
Evaluation

Synthetic workload (Streaming)

Lower is Better!

(a) High-end server
(b) Low-end server
Synthetic workload (Dynamic)

Evaluation

(a) High-end server

(b) Low-end server
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 dependence on the host’s CPU**
Realistic workload (Streaming - PR)

Evaluation

(a) High-end server

(b) Low-end server
Realistic workload (Streaming - PR)

Evaluation

(a) High-end server

(b) Low-end server
Realistic workload (Streaming - PR)

(a) High-end server

(b) Low-end server
Realistic workload (Dynamic - BFS)

Evaluation

(a) High-end server

(b) Low-end server
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
Evaluation

CPU utilization

(a) High-end server

(b) Low-end server

x4.3 higher (than low-end)
Evaluation

CPU utilization

(a) High-end server

- GPUKV
- KVSSD
- RocksDB(C)
- RocksDB(NC)

(b) Low-end server

- GPUKV
- KVSSD
- RocksDB(C)
- RocksDB(NC)

Polling can be removed by:
Direct request send from GPU to SSD
Interrupt between host and GPU

96%
80%
Evaluation

CPU utilization

(a) High-end server

(b) Low-end server

IO Request

Thd. Mgmt.

Polling

Others.

Aggr. CPU Cycles normalized wrt GPUKV

GPUKV  KVSSD  RocksDB(C)  RocksDB(NC)  GPUKV  KVSSD  RocksDB(C)  RocksDB(NC)
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