Dynamic Near Data Processing Framework for SSDs

Gunjae Koo*, Kiran Kumar Matam*, Te I†, H.V. Krishna Giri Nara*, Jing Li‡, Hung-Wei Tseng†, Steven Swanson‡, Murali Annavaram*

*University of Southern California
†North Carolina State University
‡University of California, San Diego
Conventional Storage = Cheap Passive Devices

**Conventional storage devices**

- **Slow, limited bandwidth** (SATA 150 ~ 600 MB/s)
- **Passive devices** (read, write, erase)

*Figures from Intel and Western Digital*
Storage in Modern Server Systems

Storage devices for Big Data

- Huge volumes of data → slow, slower, much slower
- Data movement is critical for performance
NVM-based storage devices

- No seek time, higher bandwidth over PCIe
- Potential to be active systems

* Figures from Intel
Intelligent Storage

NVM-based storage devices

- No seek time, higher bandwidth (PCIe)
- Potential to be active systems

* Figures from Intel

SSD Processor

DRAM

NAND flash packages

* Figures from Intel
Near Data Processing (NDP)

- **CPU**
- **Storage interface**
- **Storage Processor (SP)**
- **Data computation @ host**
- **Data transfer from storage**
- **Internal**
- **External (host – storage)**
Near Data Processing (NDP)

W/O NDP
- Data computation @ host
- Data transfer from storage

With NDP
- Data computation @ storage
Near Data Processing (NDP) on SSDs

- **Host**: CPU
- **Storage interface**: Connection between Host and Storage
- **SP**: Storage Processor
  - Garbage collection
  - Wear-leveling

**Without NDP (W/O NDP)**
- Data computation @ host
- Data transfer from storage
- Internal

**With NDP**
- Data computation @ storage
- Data transfer from storage
Obstacles to in-SSD processing

- Less powerful embedded processor
- Dynamic computation resource availability

Summarizer: Dynamic NDP framework for SSD
Summarizer – Basic Concept

Monitoring resources
Summarizer – Basic Concept

Monitoring resources

Host

CPU

Storage interface

AP

Database
Host CPU

User Applications / Operating Systems

NVMe Host Driver

Storage Interface (PCIe / NVMe)

SSD Firmware

I/O Controller (NVMe command decoder)

Flash Translation Layer (FTL)

Request queue

Response queue

SSD Embedded Processors

SSD SoC Interconnection

Flash Controller

DRAM Controller

NAND Flash

SSD DRAM

Host Memory

SQ

CQ

User Functions

Task Controller

Summarizer

TQ
Summarizer – Initialization (Function Offloading)

Host Memory

SQ  CQ

Host CPU

User Applications / Operating Systems

Storage Interface (PCIe / NVMe)

SSD Firmware

I/O Controller (NVMe command decoder)

Flash Translation Layer (FTL)

SSD SoC Interconnection

Flash Controller

NAND Flash

SSD DRAM

New NVMe command

NVMe Host Driver

INIT (foo)

User Functions

f#1  foo()

Task Controller

TQ

Function registration

Function offloading
Summarizer – Computation (Dynamic mode)

Host CPU

User Applications / Operating Systems

Storage Interface (PCIe / NVMe)

New NVMe command

RD&PROC( LBA, foo)

New NVMe command decode

I/O Controller (NVMe command decoder)

Flash Translation Layer (FTL)

RD&PROC(PPA, foo)

Request queue

Response queue

SSD SoC Interconnection

Task Controller

User Functions

f#1  foo()

f#2  goo()
Summarizer – Computation (Dynamic mode)
Summarizer – Computation (Dynamic mode)

Host CPU

User Applications / Operating Systems

NVMe Host Driver

Host Memory

Storage Interface (PCIe / NVMe)

SSD Firmware

I/O Controller (NVMe command decoder)

Flash Translation Layer (FTL)

RD&PROC(PPA,foo)

Request queue

Response queue

RD&P(PPA1,foo)

SSD SoC Interconnection

Flash Controller

DRAM Controller

NAND Flash

Page data

Summarizer

Task Controller

Register in TQ

buf1, foo

User Functions

f#1

foo1()

f#2

goo()
Summarizer – Computation (Dynamic mode)

- Host CPU
  - User Applications / Operating Systems
  - NVMe Host Driver

- Host Memory
  - SQ
  - CQ

- Storage Interface (PCIe/NVMe)

- SSD Firmware
  - I/O Controller (NVMe command decoder)
  - Flash Translation Layer (FTL)
    - RD&PROC(PPA,foo)
  - Flash Controller
  - DRAM Controller
  - NAND Flash
  - Page data

- SSD SoC Interconnection

- Task Controller
  - Task Controller (CC)
  - Request queue
  - Response queue
  - RD&PROC(PPA,foo)
  - RD&P(PPA1,foo)

- Summarizer
  - User Functions
    - f#1
    - foo()
    - f#2
    -goo()
Evaluation Platform

- LS2085a intelligent SSD development platform
- ARM cores running FTL and **Summarizer** firmware
- FPGA implementing NAND flash controller
- PCIe Gen. 3 4x lanes for host communication
Evaluation Platform

- LS2085a intelligent SSD development platform
- ARM cores running FTL and *Summarizer* firmware
- FPGA implementing NAND flash controller
- PCIe Gen. 3 4x lanes for host communication
Evaluation - Performance

TPC-H Query6

- **SDD time**
- **Host time**

Static workload offloading

Static workload offloading
Evaluation - Performance

TPC-H Query6

![Chart showing SDD time and Host time for CPU only processing and SSD only processing in static and dynamic scenarios.](chart.png)

- **CPU only processing (baseline)**
- **SSD only processing**

Static
- 0
- 0.2
- 0.4
- 0.6
- 0.8
- 1

Dynamic
Evaluation - Performance

TPC-H Query 6

- SDD time
- Host time

Summary: Dynamic Offloading

Static

0
0.2
0.4
0.6
0.8
1

Dynamic
TPC-H Query6

**Evaluation - Performance**

- **SDD time**
- **Host time**

**SSD processing + transfer time**
(Internal + external + In-SSD processing)

**Host CPU processing time**

<table>
<thead>
<tr>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>0.8</td>
<td>1</td>
</tr>
</tbody>
</table>

- SSD processing + transfer time (internal + external + In-SSD processing)
- Host CPU processing time
Evaluation - Performance

TPC-H Query6

Execution time normalized to baseline (CPU only)

<table>
<thead>
<tr>
<th>Static</th>
<th>0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>0.8</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dynamic
Evaluation - Performance

TPC-H Query6

Execution time (normalized to baseline)

SDD time  □ Host time

Static

0.2
0.4
0.6
0.8
1

Dynamic

0
1
2
3
4

0.2
0.4
0.6
0.8
1
Evaluation - Performance

Execution time (normalized to baseline)

- CPU only
  - SDD time: 0.70
  - Host time: 0.30
- Dynamic
  - SDD time: 0.60
  - Host time: 0.24
Evaluation - Performance

- **W/O NDP**
  - Data computation @ host
  - External (host – storage)
  - Internal

- **With NDP**
  - Data computation @ storage
  - Data transfer from storage

### Static vs. Dynamic Chart Title
- **SDD time**
- **Host time**

#### CPU only
- **Static**
- **Dynamic**

#### W/O NDP
- **Data computation @ host**
- **Data transfer from storage**

#### With NDP
- **Data computation @ storage**
Evaluation - Performance

TPC-H Query 6

Performance degraded by static NDP

SDD time - Host time

Static

Dynamic
Evaluation - Performance

TPC-H Query 6:
- Execution time (normalized to baseline)
- SDD time vs Host time
- Dynamic vs Static
- Improvement: 16%

TPC-H Query 1:
- Execution time (normalized to baseline)
- SDD time vs Host time
- Dynamic vs Static
- Improvement: 10%

TPC-H Query 14:
- Execution time (normalized to baseline)
- SDD time vs Host time
- Dynamic vs Static
- Improvement: 20%

String Similarity Join:
- Execution time (normalized to baseline)
- SDD time vs Host time
- Dynamic vs Static
- Improvement: 7%
Better embedded processor is cost effective
Design Exploration – Higher Internal Bandwidth

Embedded processor performance

- TPC-H Query 6
- TPC-H Query 1
- TPC-H Query 14
- String Similarity Join
- Average
Summarizer is a cost effective NDP solution with powerful storage processors.
Conclusion

✓ **Dynamic NDP framework for SSDs**
  - Opportunistically enables in-SSD processing
  - Page-level NDP control
  - Automatic workload partitioning

✓ **Summarizer programming model**
  - Evaluation on the real development platform
  - Explored design space for future SSDs
Thank you

Summarizer: Trading Communication with Computing Near Storage (MICRO ‘17)

(We thank to Dell EMC for supporting the SSD development board)