HetCache: Synergising NVMe Storage and GPU acceleration for Memory-Efficient Analytics

Hamish Nicholson, Aunn Raza, Periklis Chrysogelos, Anastasia Ailamaki
The Broken Pillars of Fast Analytics

- Memory is cheap
  - Memory is (relatively) expensive

- Cache hits are key to performance
  - NVMe array bandwidth competitive with memory bandwidth

- NUMA is insignificant compared to persistent storage access
  - Increasing accelerator heterogeneity

Storage must be workload & hardware aware
Heterogeneous Hierarchies have Multiple Transfer Paths

1. DRAM to GPU (32 GB/s)
   - Eagerly transfer pages to GPU-memory
   - Byte-addressable access by GPU

2. NVMe to DRAM (86 GB/s, block)

3. NVMe to GPU-memory to GPU (32GB/s, block)

Data routing requires optimizing for path BW & granularity
NVMe BW Saturates CPU Throughput

- Data intensive
- Data intensive in-memory
- Processing intensive
- Processing intensive in-memory

Query Execution Time (s) vs. Storage Bandwidth GB/s

- Data intensive
- Processing intensive

GPU needs caching to mitigate interconnect bottleneck

CPU

- DRAM
- NVMe storage
- >100 GB
- 10-100 TB

GPU

- Memory
- NVMe storage
- <80 GB
- 10-100 TB

24-core AMD EPYC 7413
NVIDIA A40, PCIe 4.0 x16
1-12x PCIe 4.0 x4 NVMe SSB1000
Block Storage Wastes Interconnect BW

SELECT T.c FROM T WHERE T.a < 50 AND T.b > 42
Block Storage Wastes Interconnect BW

SELECT T.c FROM T WHERE T.a < 50 AND T.b > 42

Staged SemiLazy Transfers

GPU Pipeline
3 block / T

Byte addressability minimizes overfetching
Transfer Path Depends on Workload and Hardware

Delay data transfers until first access

*Chrysogelos et. al [VLDB 2019]
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs

Delay data transfers until first access

*Chrysogelos et. al [VLDB 2019]
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents

---

Delay data transfers until first access

*Chrysogelos et. al [VLDB 2019]*
Transfer Path Depends on Workload and Hardware

Delay data transfers until first access

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Chrysogelos et. al [VLDB 2019]
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

```
1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Chrysogelos et. al [VLDB 2019]*

Delay data transfers until first access

Table T

HetCache

Router*

mem-move*

aggregate

filter

router

aggregate

logical scan

A

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

B

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

C

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

Delay data transfers until first access

*Chrysogelos et. al [VLDB 2019]
1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Chrysogelos et al [VLDB 2019]

Delay data transfers until first access
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Chrysogelos et. al [VLDB 2019]

Delay data transfers until first access
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

Delay data transfers until first access

*Chrysogelos et. al [VLDB 2019]
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

Delay data transfers until first access

*Chrysogelos et. al [VLDB 2019]
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. $\text{mem-move}$ consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Delay data transfers until first access*

*Chrysogelos et al [VLDB 2019]*
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

Delay data transfers until first access

*Chrysogelos et. al [VLDB 2019]
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Chrysogelos et. al [VLDB 2019]

Delay data transfers until first access
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Chrysogelos et. al [VLDB 2019]

Delay data transfers until first access
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Chrysogelos et. al [VLDB 2019]

Delay data transfers until first access
Delay data transfers until first access

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

*Chrysogelos et. al [VLDB 2019]
Transfer Path Depends on Workload and Hardware

1. Logical scan emits page IDs
2. Route based on cache contents
3. mem-move consults HetCache on:
   - Locations of in-memory copies
   - Preferred location to cache, if any

Delay data transfers until first access

*Chrysogelos et. al [VLDB 2019]
Experimental Setup

- **Hardware**
  - 24-core AMD EPYC 7413
  - NVIDIA A40, PCIe 4.0 x16
  - 12x PCIe 4.0 x4 NVMe, 7GB/s each

- **Software**
  - Proteus: Hybrid CPU-GPU analytical engine

- **Benchmark**
  - Star Schema Benchmark. (SF 1000)
  - ~96GB(Q1.x – 3.x) working set per query
Combining Transfer Paths for GPU

Enabling sub-page accesses via DRAM staging => up to 45% faster

24-core AMD EPYC 7413
NVIDIA A40 48GB, PCIe 4.0 x16
8x (NVMe, PCIe 4 x4)
SSB SF=1000
Memory Efficient CPU-GPU Execution

- DRAM-resident
- NVMe-resident
- HetCache

Execution Time (s)

Data intensive (Q1.3)
- 10 GB GPU memory
- 76 GB CPU memory

Processing intensive (Q3.1)
- 10 GB GPU memory
- 0 GB CPU memory

25% - 100% less DRAM used for inputs

24-core AMD EPYC 7413
NVIDIA A40 48GB, PCIe 4.0 x16
8x (NVMe, PCIe 4 x4)
96GB working set
Storage BW is approaching DRAM BW

- (Near) in-memory performance on larger-than-DRAM datasets
  - Granularity and processing throughput-aware data placement

- Efficient interconnect use for NVMe-GPU transfers
  - Stage selectively accessed data in DRAM for GPUs

- Storage systems must be hardware & workload aware

Thank You!