StreamBox-HBM
Stream Analytics on High Bandwidth Hybrid Memory

Hongyu Miao, Purdue ECE; Myeongjae Jeon, UNIST; Gennady Pekhimenko, UToronto;
Kathryn S. McKinley, Google; Felix Xiaozhu Lin, Purdue ECE

http://xsel.rocks/p/streambox
Timely processing of streaming data

High Throughput & Low Latency!

On 100+ GB memory
Hybrid Memory: 3D Memory + DRAM

DRAM
- Larger capacity, but lower bandwidth

3D Memory
- Higher bandwidth, but smaller capacity
- NO latency benefit (Unlike cache: SRAM+DRAM)
- Same as DRAM without high parallelism or sequential access
- As cache of DRAM? → Poor performance...
**Can hybrid mem speed up stream analytics?**

Yes! **StreamBox-HBM**

- The **first** stream engine optimized for 3D memory + DRAM on real hardware
- Achieves the **best** reported throughput on single node (win-avg:110MRec/s)
- Speeds up stream analytics by **7x**

![Diagram showing throughput vs. number of cores]

- **Throughput Mrec/s**
- **TopK Per Key**
- **3D + DRAM in-mem-index**
- **3D as cache full-records**
- **7x speedup**
Challenges

1. Hash Grouping performs poorly on 3D memory
2. 3D memory is capacity limited
3. How to dynamically map streaming data to hybrid mem?
Challenge 1: Hash Grouping performs poorly on 3D memory

- Operators: computations consume/produce streams
- Pipeline: a graph of streaming operators

• Data Grouping
  • A set of very common and expensive operators that reorganize records
  • Hash with random access in existing engines → Performs poorly on 3D memory...
Challenge 2: 3D memory is capacity limited

- Streaming data
  - High data volume (100+ GB)

- 3D Memory
  - Capacity limited (~16 GB)

- 3D memory is NOT large enough to hold all streaming data....
Challenge 3: managing two types of memory

• How to **dynamically** map data/operators to two types of memory?

Unbounded data

Various queries

Ingestion → Window → Groupby key → Average per key → Top Key

**What to map?**

**Where to map?**

Hybrid memory: benefit & limitation

3D Memory: 16 GB, 375 GB/s

DRAM: 96 GB, 80 GB/s
StreamBox-HBM Solutions

1. Hash grouping performs poorly on 3D memory
   - Solution 1: Use high parallel Sort for grouping

2. 3D memory is capacity limited
   - Solution 2: Only use 3D memory to store in-memory indexes

3. How to manage two types of memory?
   - Solution 3: Balance two limited resource with a single knob
Solution 1: Parallel Sort for Grouping

Known duals of Grouping: Hash vs. Sort
  • DRAM: Hash is the best [VLDB’09, VLDB’13, SIGMOD’15]
  • Contribution: 3D memory reverses the debate. Sort outperforms Hash.

Sort is worse than Hash on algorithmic complexity
  • $O(N\log N)$ vs. $O(N)$

Yet, Sort outperforms Hash after we exploit all:
  • Abundant memory bandwidth
  • High task parallelism
  • Wide SIMD (avx512)

Solution 1: Parallel Sort for Grouping

Throughput

Mem bandwidth

Sort outperforms Hash on 3D memory
Solution 1: Parallel Sort for Grouping

Sort outperforms Hash on 3D memory
Solution 1: Parallel Sort for Grouping

Throughput

Sort outperforms Hash on 3D memory

Mem bandwidth
Solution 1: Parallel Sort for Grouping

Throughput

Sort outperforms Hash on 3D memory

Mem bandwidth
Solution 1: Parallel Sort for Grouping

Sort outperforms Hash on 3D memory
Solution 2: Only use 3D memory for in-memory index

- Smaller
- Faster
- More efficient
- K Swapping

Index <key, pointer>

Full Records <key, key1, v1, v2, v3...>

Streaming data

Minimize the use of precious 3D mem’s capacity while exploit high bandwidth
Solution 3: balance two limited resources
Solution 3: balance two limited resources

High pressure on 3D Memory capacity
Solution 3: balance two limited resources

High pressure on 3D Memory capacity ➔ indexes on DRAM
Solution 3: balance two limited resources

3D Memory

16 GB

Cores

3D stacked Capacity

80 GB/s

Pressure rebalanced
Solution 3: balance two limited resources

High pressure on DRAM bandwidth
Solution 3: balance two limited resources

High pressure on DRAM bandwidth ➔ more indexes on 3D memory
Solution 3: balance two limited resources

3D Memory

16 GB

Cores

3D-stacked Capacity

80 GB/s

DRAM Bandwidth

Pressure rebalanced
Solution 3: balance two limited resources

High pressure on both... → reach hardware limit → limit data ingestion
Other optimizations

• Customized memory allocator
• Customized task scheduler for high pipeline and data parallelism
• High parallel merge-sort kernels using avx-512
• Dynamically handle key changes
• Parallel aggregation
• Co-design RDMA ingestion with memory management and task scheduling
• Task parallelism to utilize all CPU cores
• ...
StreamBox-HBM Implementation

• Based on our prior work StreamBox [USENIX ATC’17]

• Implement on real hardware (Intel KNL) with RDMA network
  • 61K lines of C++11, of which 38K lines are new
  • Open source: http://xsel.rocks/p/streambox

16GB 3D memory
96GB DRAM
64 cores @1.3GHz

Ninja Developer Platform (KNL)

Mellanox ConnectX-2

40Gb/s

Evaluation

• Comparing to widely used stream analytics engine
• Validating our key system designs
StreamBox-HBM is 10x faster than Flink

Throughput MRec/s

# Cores

5-10x

Benchmark: Yahoo Stream Benchmark.
Output delay: 1 second
Poor performance without any key designs
In-mem-index performs better than full-record

Using in-mem index
3D memory boosts performance

Throughput Mrec/s

TopK Per Key

Using 3D memory

- 3D as cache in-mem-index
- DRAM only in-mem-index
- 3D as cache full-records

# cores
**SW** better manages hybrid memory than **HW**

**SW manages hybrid memory**

---

**TopK Per Key**

Throughput Mrec/s

- **3D + DRAM in-mem-index**
- **3D as cache in-mem-index**
- **DRAM only in-mem-index**
- **3D as cache full-records**

---

**# cores**
Performance improve with all system designs

Using all key system designs

TopK Per Key

Throughput Mrec/s

# cores
StreamBox-HBM

The first stream engine optimized for 3D Memory + DRAM on real hardware

1. Grouping with Sort
   - Hash ➔ Sort
   - Abundant memory
   - High parallelism
   - Wide SIMD (avx512)
   - Sequential access

2. In-memory index in 3D Memory
   - Index <key, pointer>
   - Full Records <key, k2, v1, v2, v3…>
   - Streaming data

3. Mng hybrid mem

Exploit high bandwidth
Minimize use of capacity
Balance limited resources

http://xsel.rocks/p/streambox
Lessons on exploiting 3D memory + DRAM

Apps
- High task parallelism
- Wide SIMD (avx512)
- Sequential mem access
- Packed data structure

Runtime
- Thread pool + custom task scheduler
- Custom mem allocator

OS kernel
- Cheap VM (huge page)
- RDMA network bypass kernel, free CPU

Hybrid Memory
- 3D Memory
- 375 GB/s
- 16 GB
- Cores
- 80 GB/s
- 96 GB
- DRAM

Thank You!