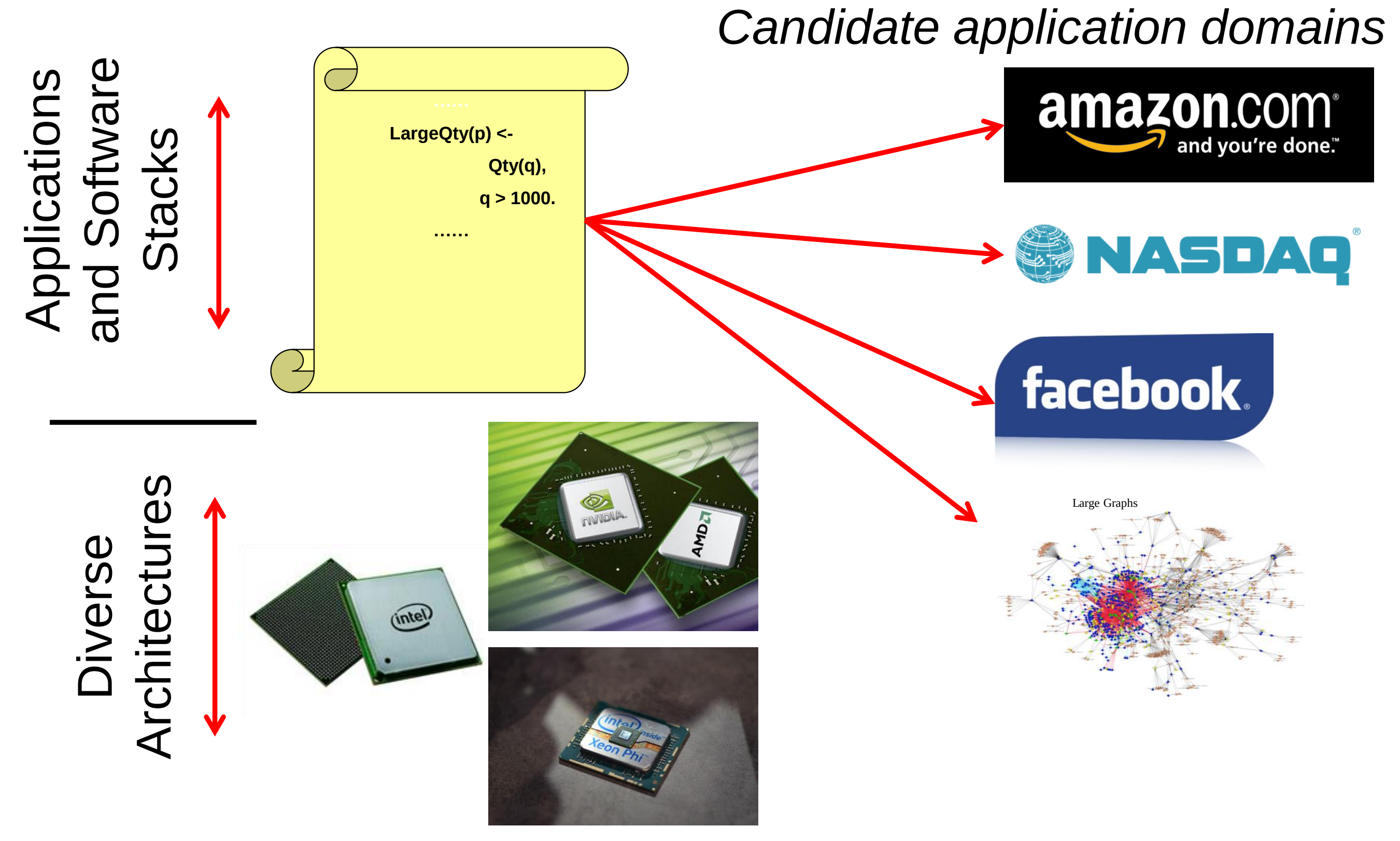
A Portable Relational Algebra Library for High-Performance Data-Intensive Query Processing

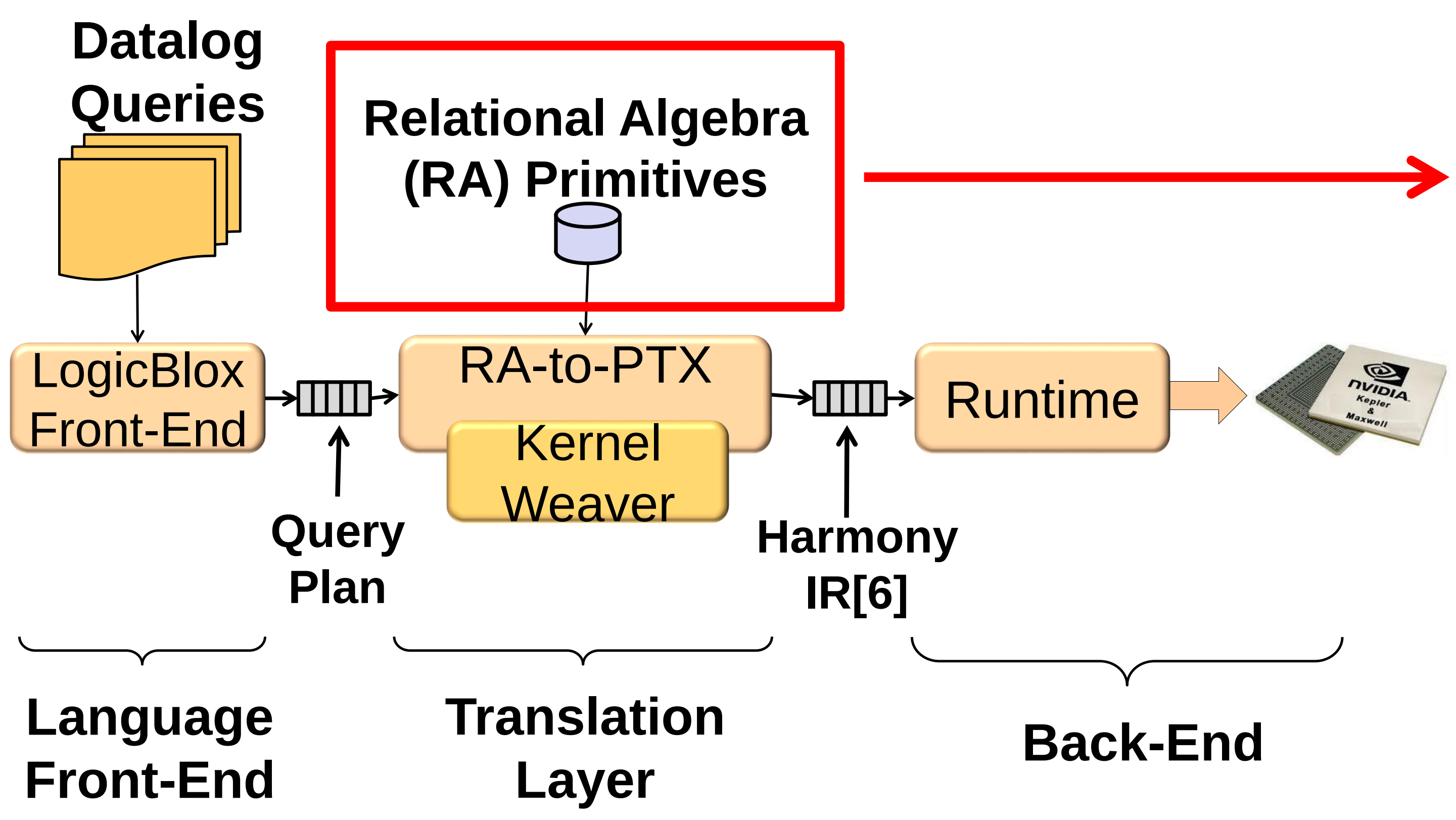
Ifrah Saeed, Haicheng Wu, and Sudhakar Yalamanchili
Georgia Institute of Technology

CHALLENGES


Candidate application domains
Applications and Software Stacks
LargeQty(p) <-
Qty(q),
q > 1000.
......
amazon.com and you're done.
NASDAQ
facebook
Large Graphs
Diverse Architectures

Relational Computations Over Massive Unstructured Data

OUR SOLUTION



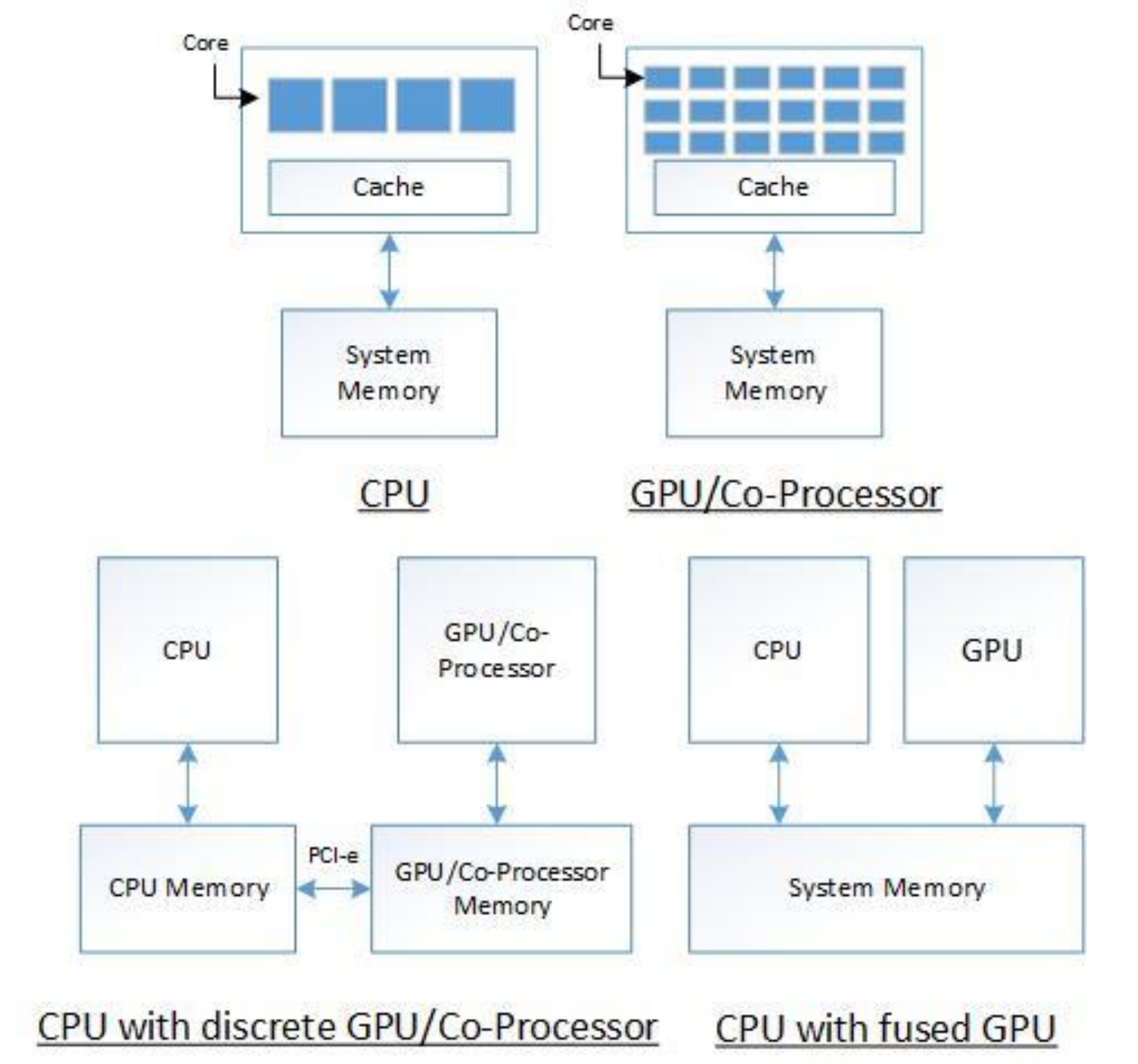
Red Fox Platform Diagram

Primitives include:

- PROJECT	- REDUCE
- SELECT	- REDUCE BY KEY
- INNER JOIN	- UNIQUE
- SET Family	- SORT
- PRODUCT	- Math (+, -, *)

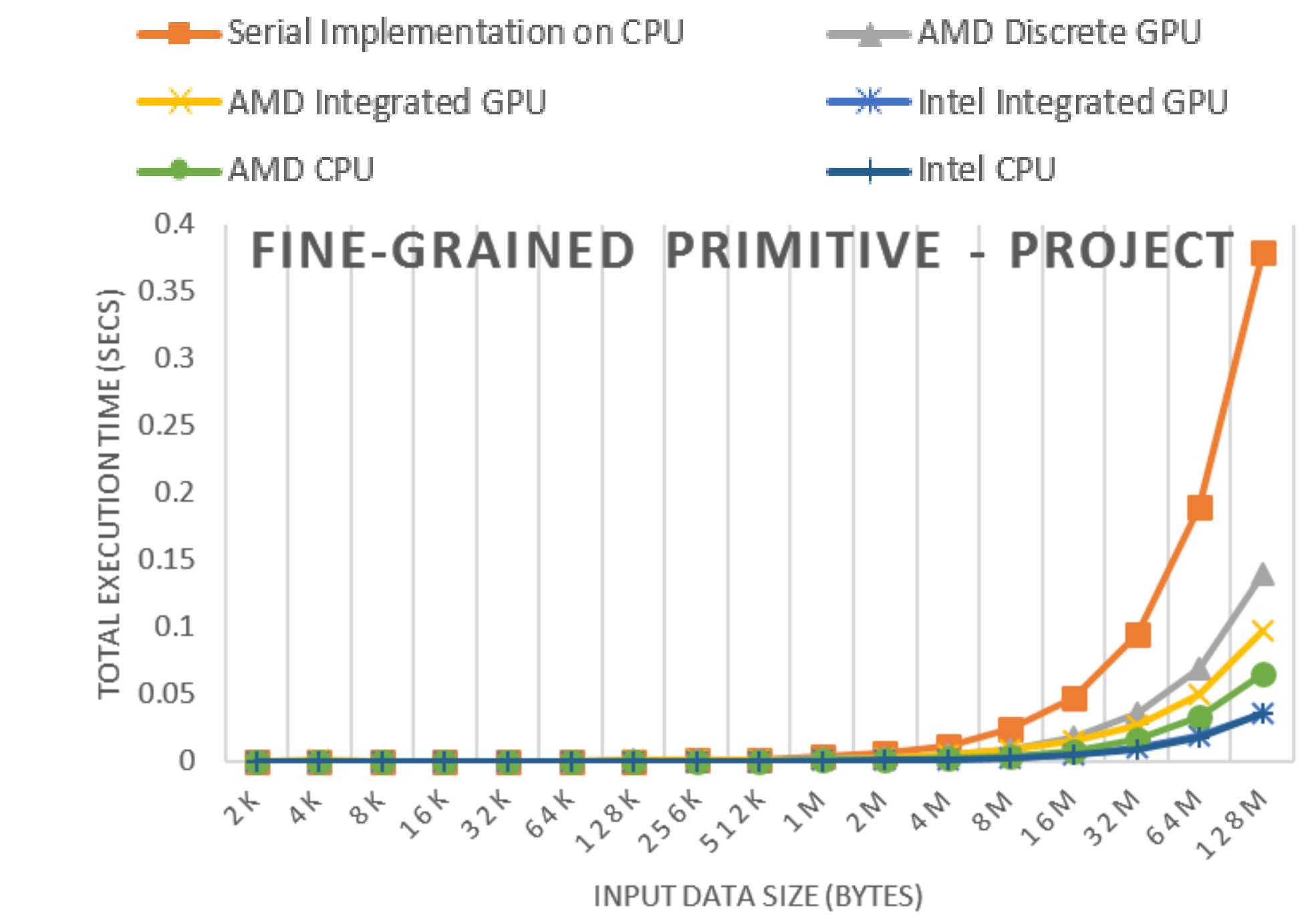
Portability is achieved using ***OpenCL.***

Work in Progress: Execution of the primitives on Intel Xeon Phi

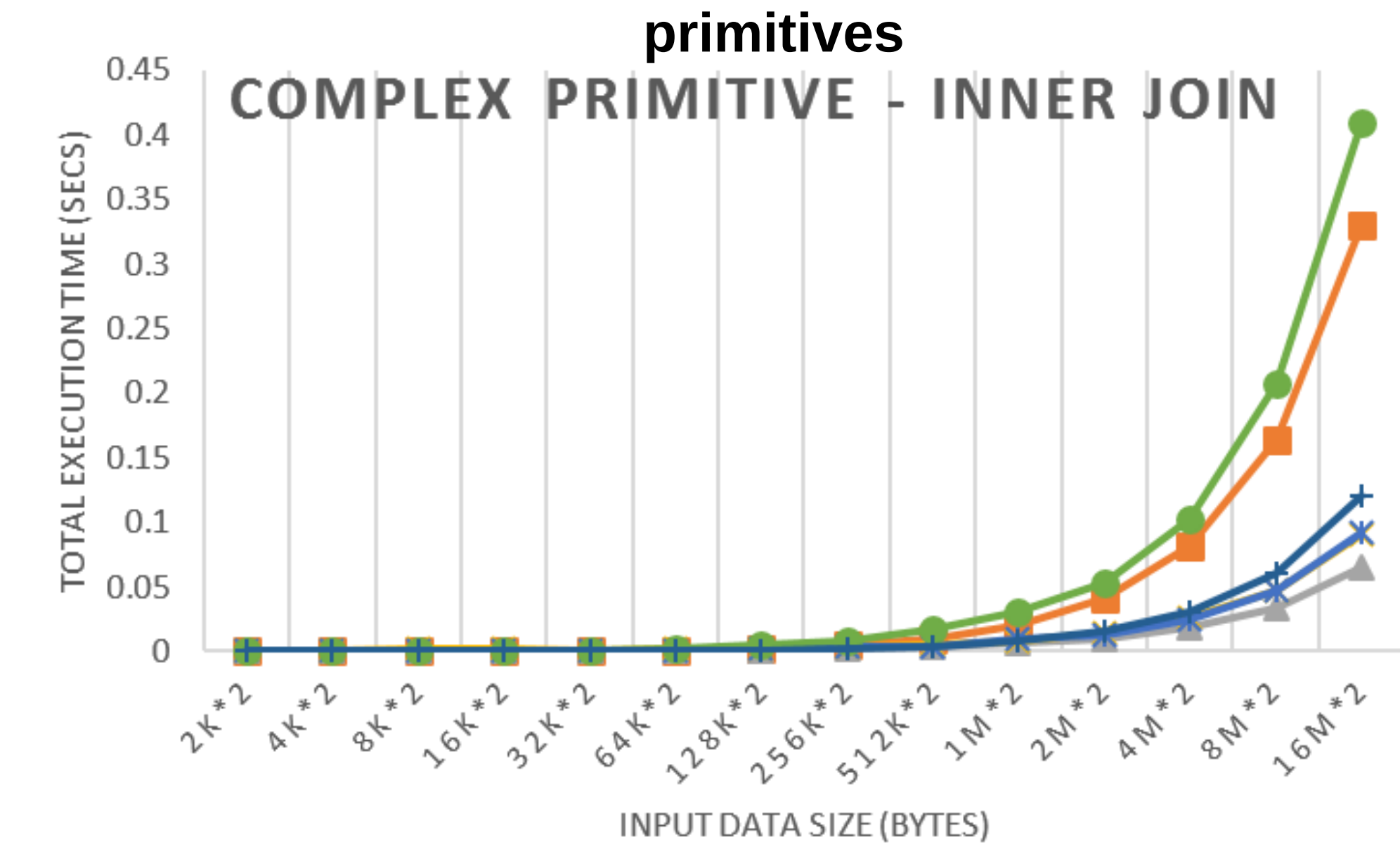


Parallel Architectures

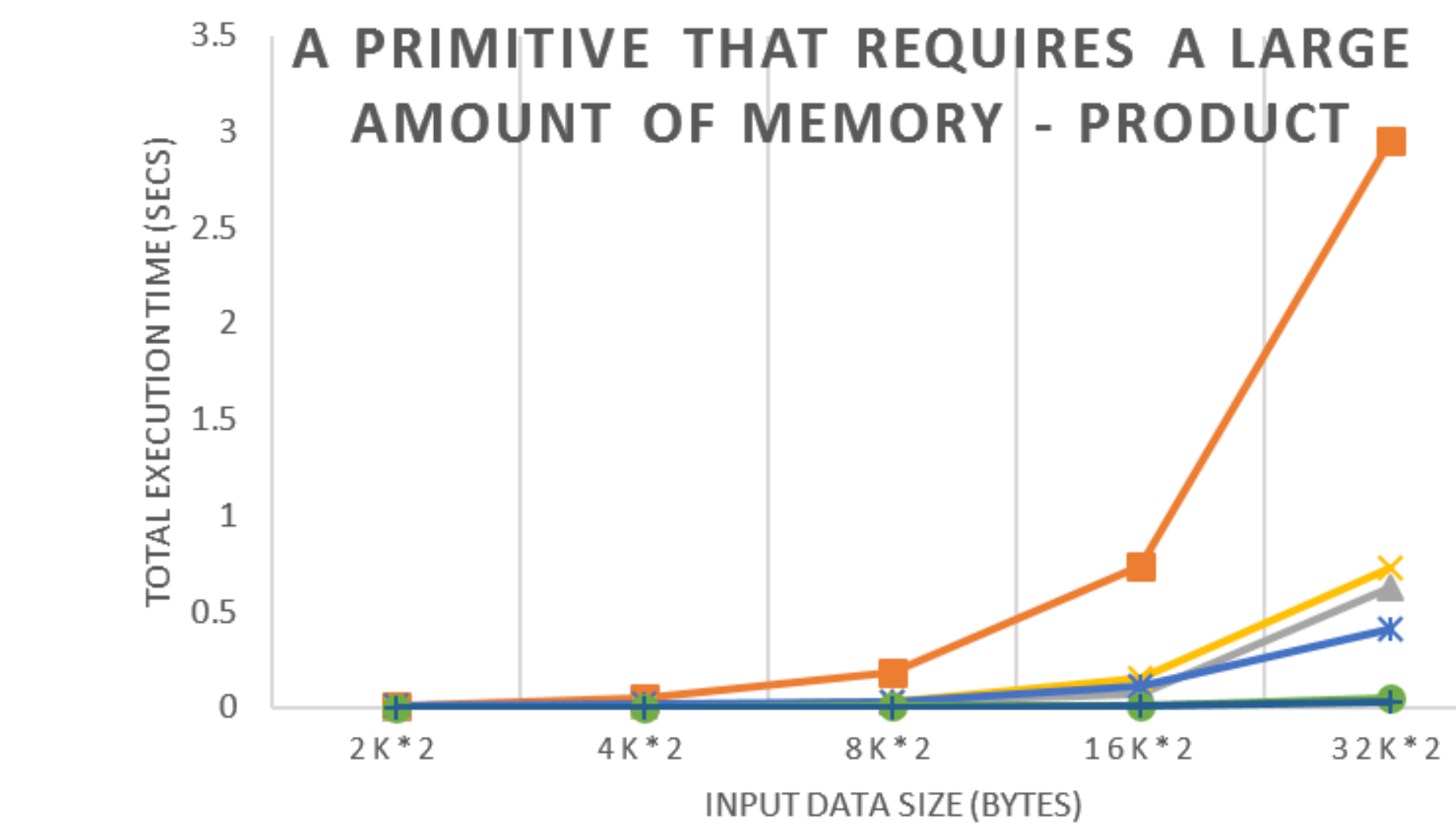
PERFORMANCE EVALUATION



Due to data transfer overhead, integrated GPUs and CPUs perform better than discrete GPUs for fine-grained primitives

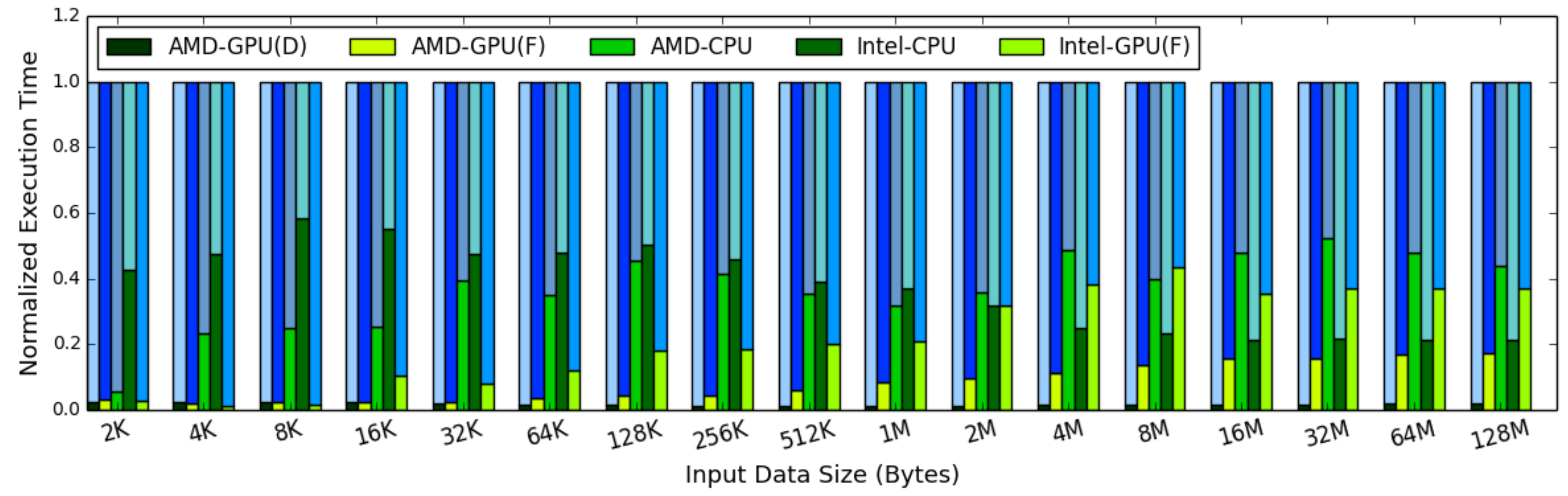


Due to higher parallelism and higher memory bandwidth, discrete GPUs outperform integrated devices for complex primitives

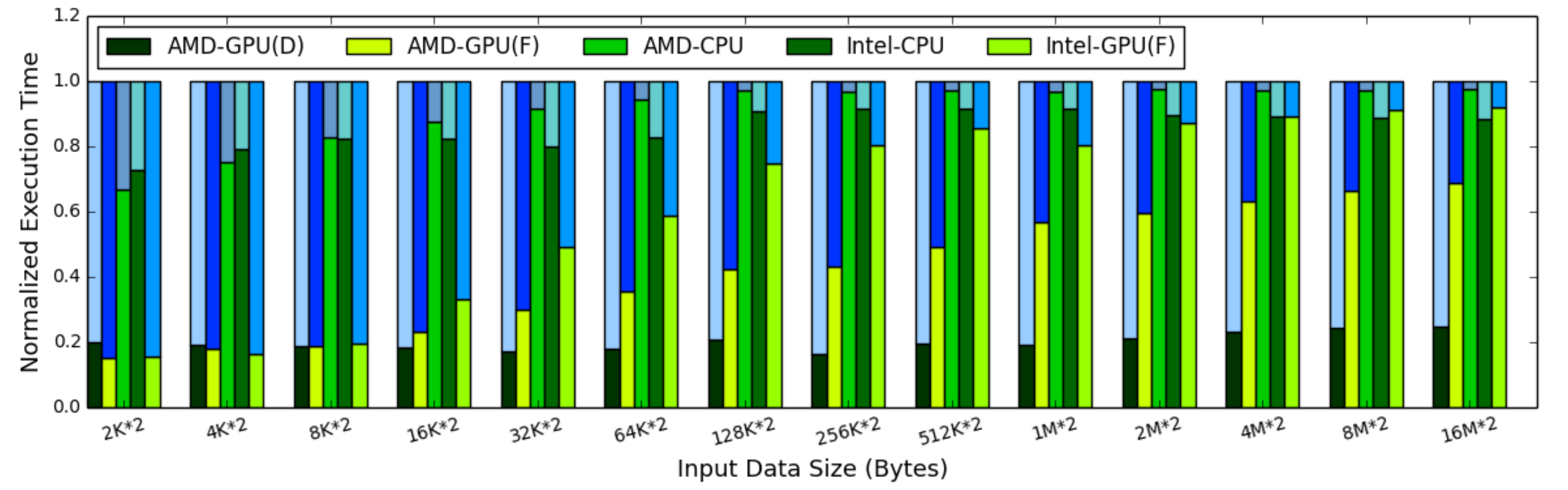


When data fits in large CPU caches, CPUs perform better than other devices

Percentage of time spent in actual computation and data transfer between the host and OpenCL device

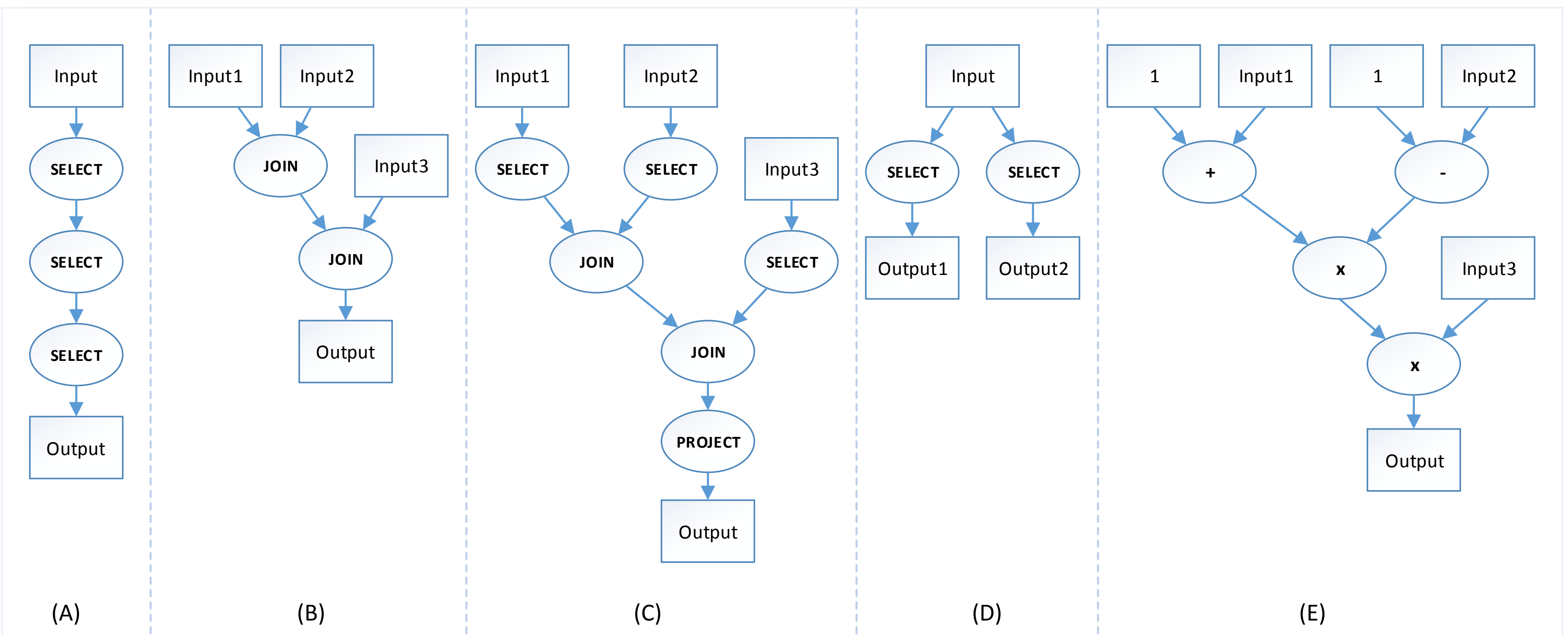


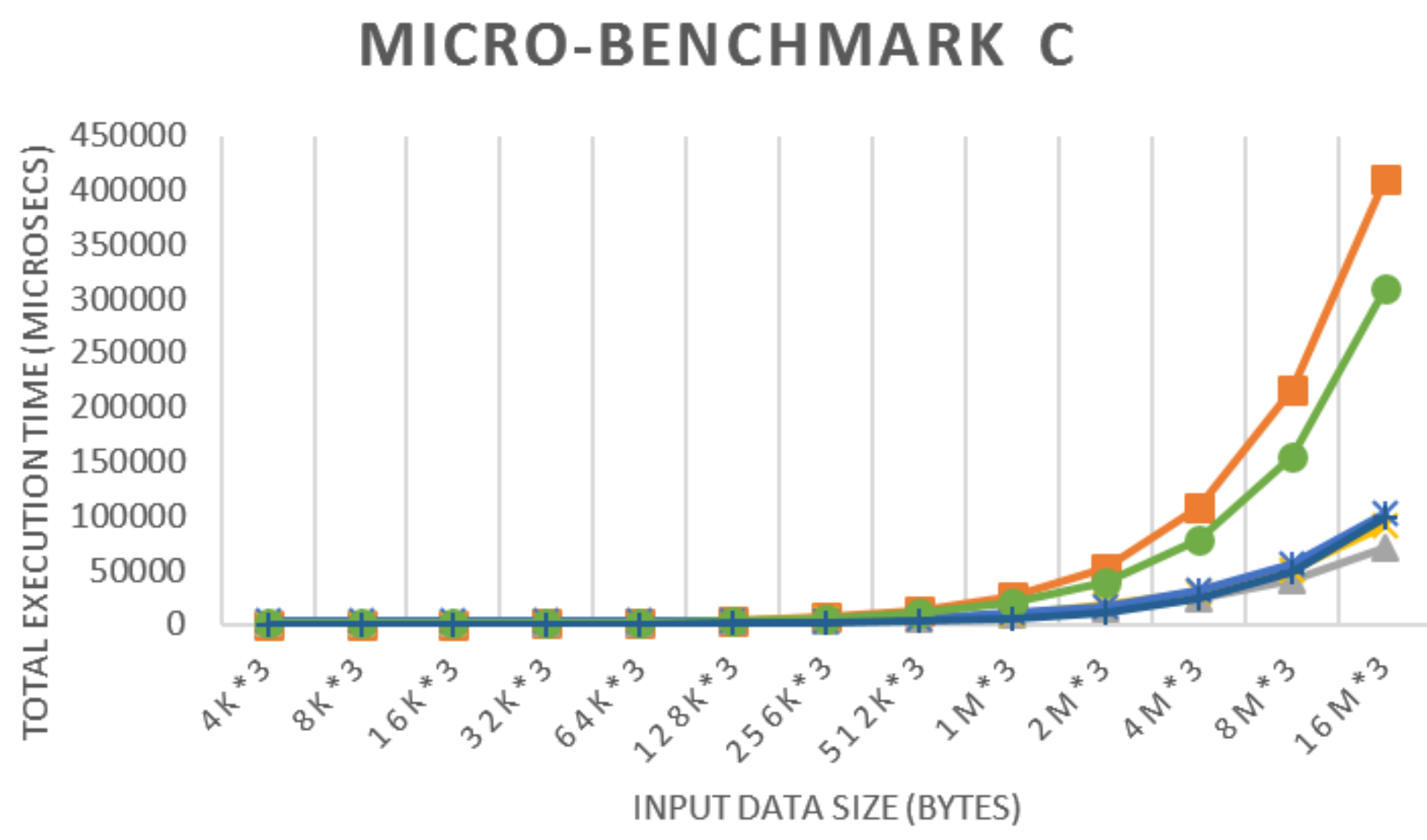
For fine-grained primitive, PROJECT, majority of the time is spent on transferring data between the host and the device

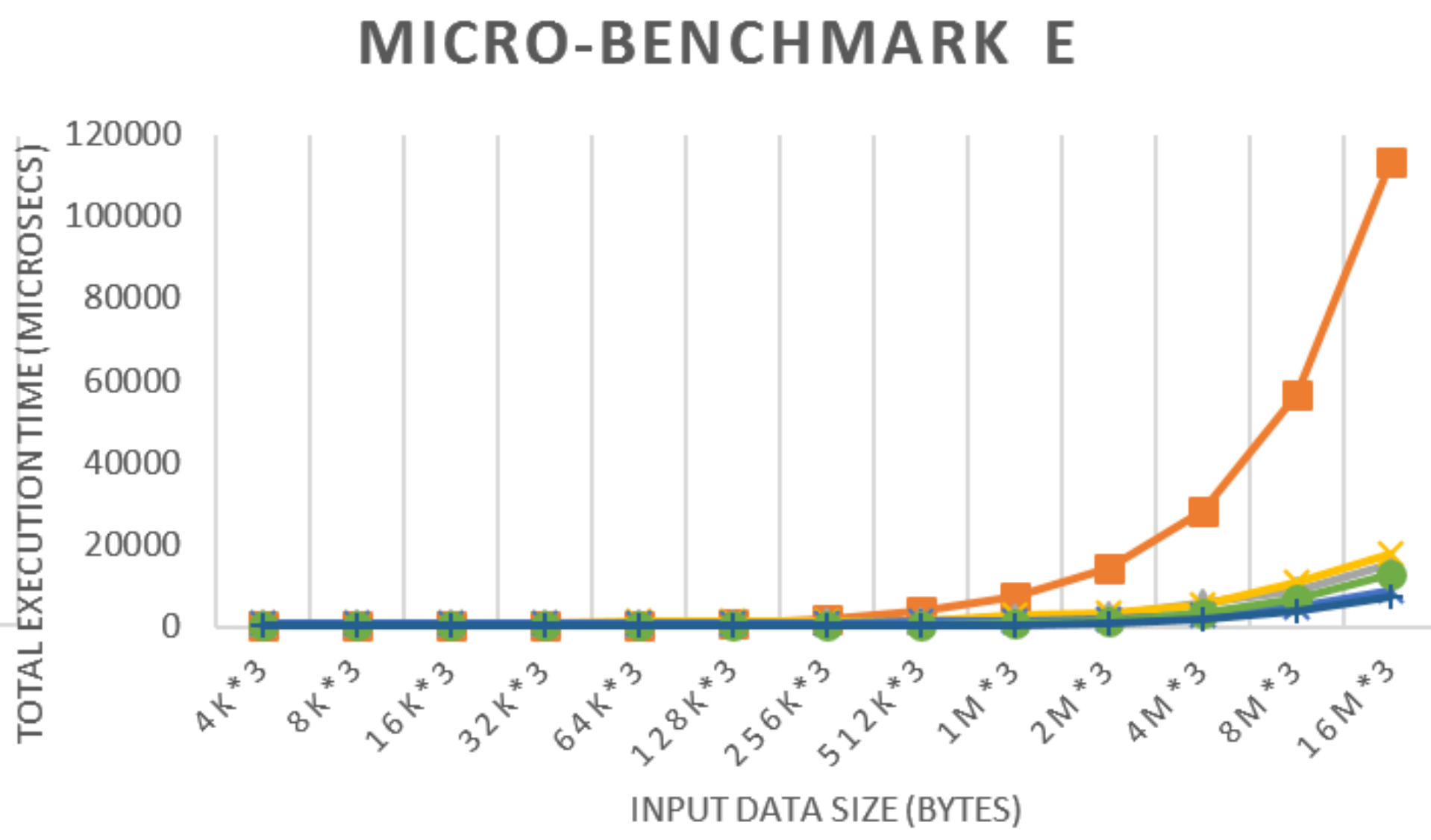


Even for a complex kernel, INNER JOIN, data transfer is a performance bottleneck as most of the total computation time is spent on transferring data

TPC-H Micro-benchmarks







For complex micro-benchmarks (A), (B), (C), and (D), the discrete GPU takes the minimum execution time (the most complex one (C) is shown here).

For the simple micro-benchmark (E), the inclusion of the data transfer and the kernel launch time degrades the performance of discrete GPUs.

CONCLUSION

- Need for a portable library containing primitives that are required to execute data-intensive relational queries.
- When multiple devices are available in the system
  - Schedule fine-grained primitives on the integrated GPU
  - Schedule complex primitives on the discrete GPU
  - Schedule primitives with cache sensitive footprints and large caches requirement on the multicore CPU

References

[1] Independent Oracle Users Group. “A New Dimension to Data Warehousing: 2011 IOUG Data Warehousing Survey.”
[2] He, Lu, Yang, Fang, Govindaraju, Luo, Sander. “Relational query co-processing on graphics processors.” TODS, 2009.
[3] Diamos, Wu, Wang, Lele, Yalamanchili. “Relational Algorithms for Multi-Bulk-Synchronous Processors.” PPoPP 2013.
[4] Wu, Diamos, Wang, Cadambi, Yalamanchili, “Optimizing Data Warehousing Applications for GPUs Using Kernel Fusion/Fission.” PLC 2012.
[5] Wu, Diamos, Cadambi, Yalamanchili, “Kernel Weaver: Automatically Fusing Database Primitives for Efficient GPU Computation.” MICRO 2012.
[6] Wu, Diamos, Sheard, Aref, Baxter, Garland, Yalamanchili, “Red Fox: An Execution Environment for Relational Query Processing on GPUs.” CGO 2014
[7] Zhang, He, He, Lu, “Omnidb: Towards portable and efficient query processing on parallel CPU/GPU architectures.” VLDB 2013