Accelerating Database Joins Using a General Purpose GPU

Kevin Angstadt, Ed Harcourt
Department of Mathematics, Computer Science, and Statistics
St. Lawrence University
Canton, NY

Overview

- Terminology
- Computer Architecture
- Database Joins
- GPU Implementation
- Results

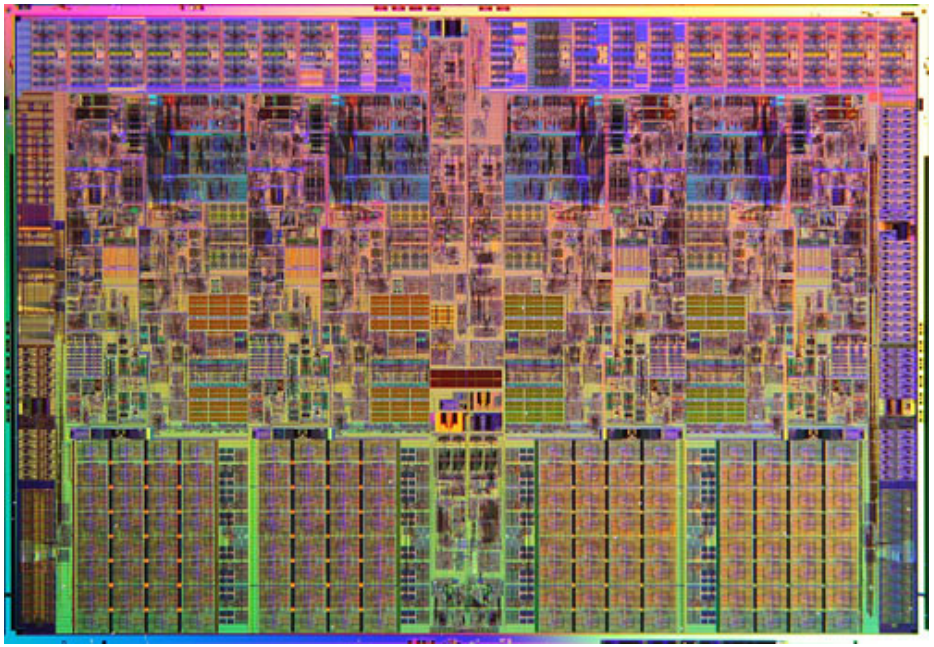
Terminology

- CPU
  - Central Processing Unit
  - Most computation done here
  - Uses RAM to store information
  - **"Host"**

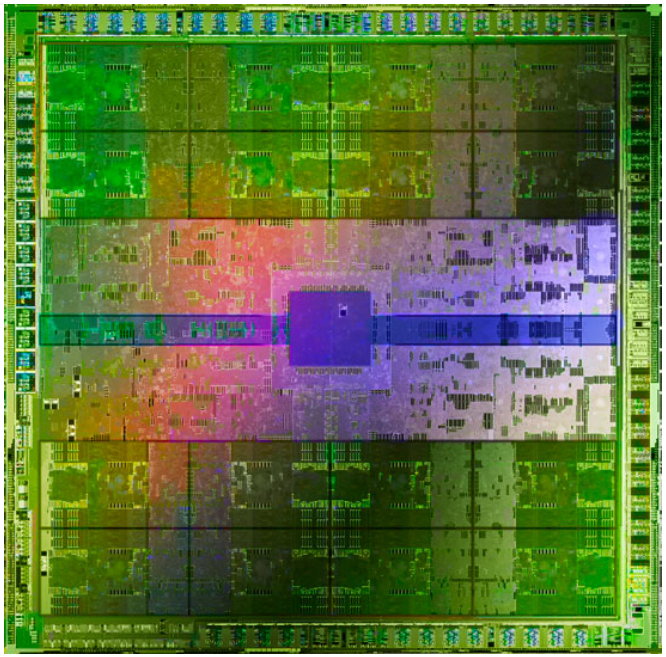
- GPU
  - Graphics Processing Unit
  - Separate card used for displaying computer graphics
  - Built-in Memory
  - **"Device"**

Computational Power

Which of these two processors (from the same era) has more computational power?



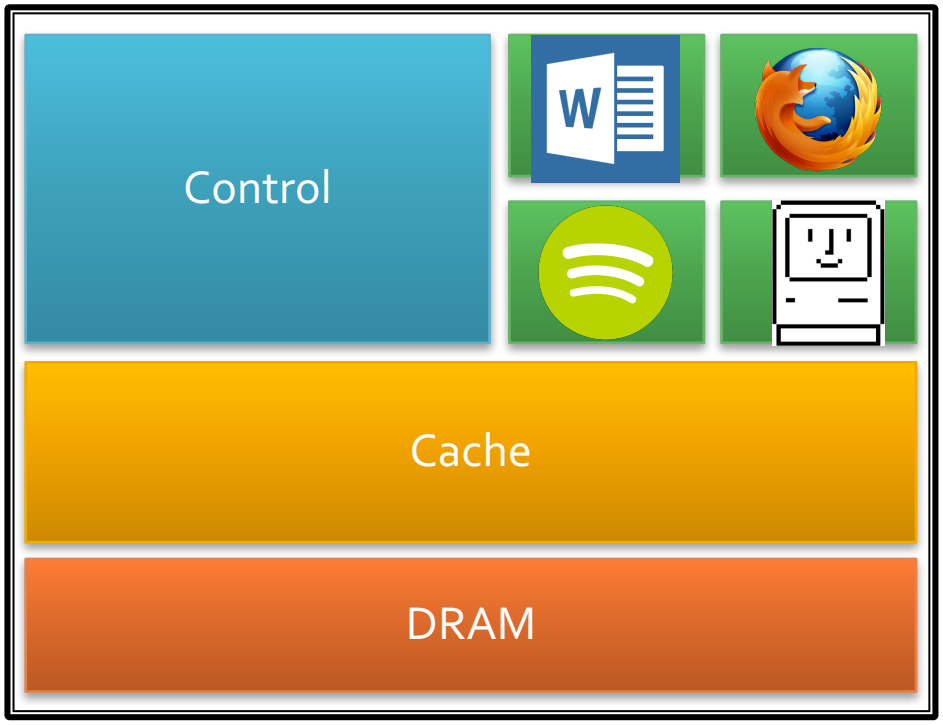
Intel Core® i7-920 CPU



nVidia Fermi GPU

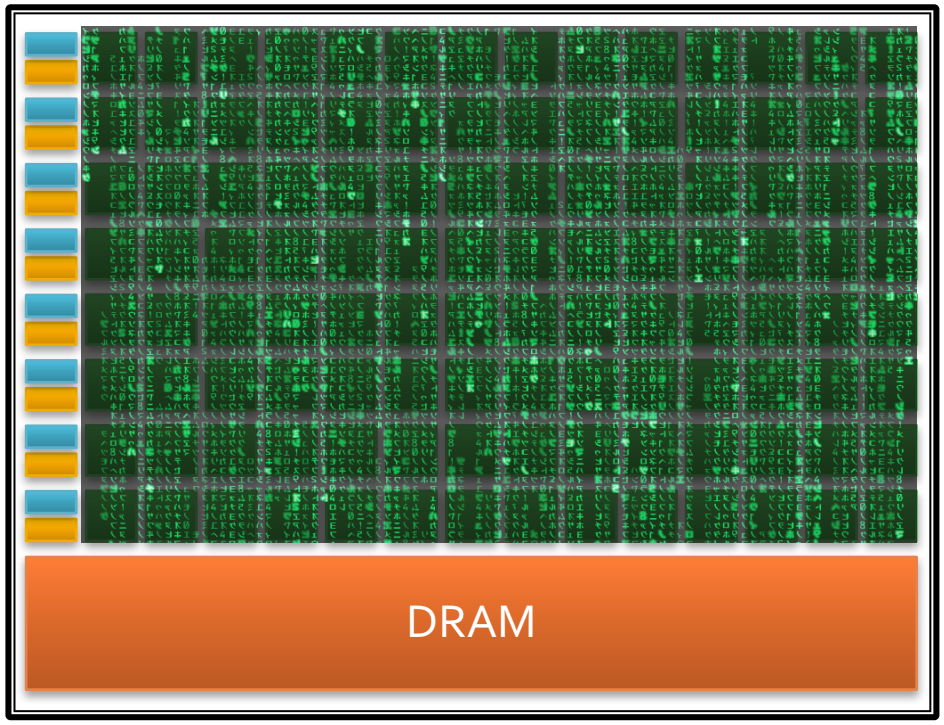
CPU vs. GPU Architecture

CPU



MIMD

GPU



SIMD

Database Joins

- Common Database operation
- Combines data rows from two tables

course_number	course_title
CS140	Intro to Computer Programming
CS220	Computer Organization
CS321	Computer Networking
CS340	Software Engineering
CS364	Programming Languages

Table **A**

course_number	course_professor
CS140	Ed Harcourt
CS140	Lisa Torrey
CS140	Choong-Soo Lee
CS220	Ed Harcourt
CS321	Choong-Soo Lee
CS340	Lisa Torrey
CS364	Lisa Torrey

Table **B**

Database Joins

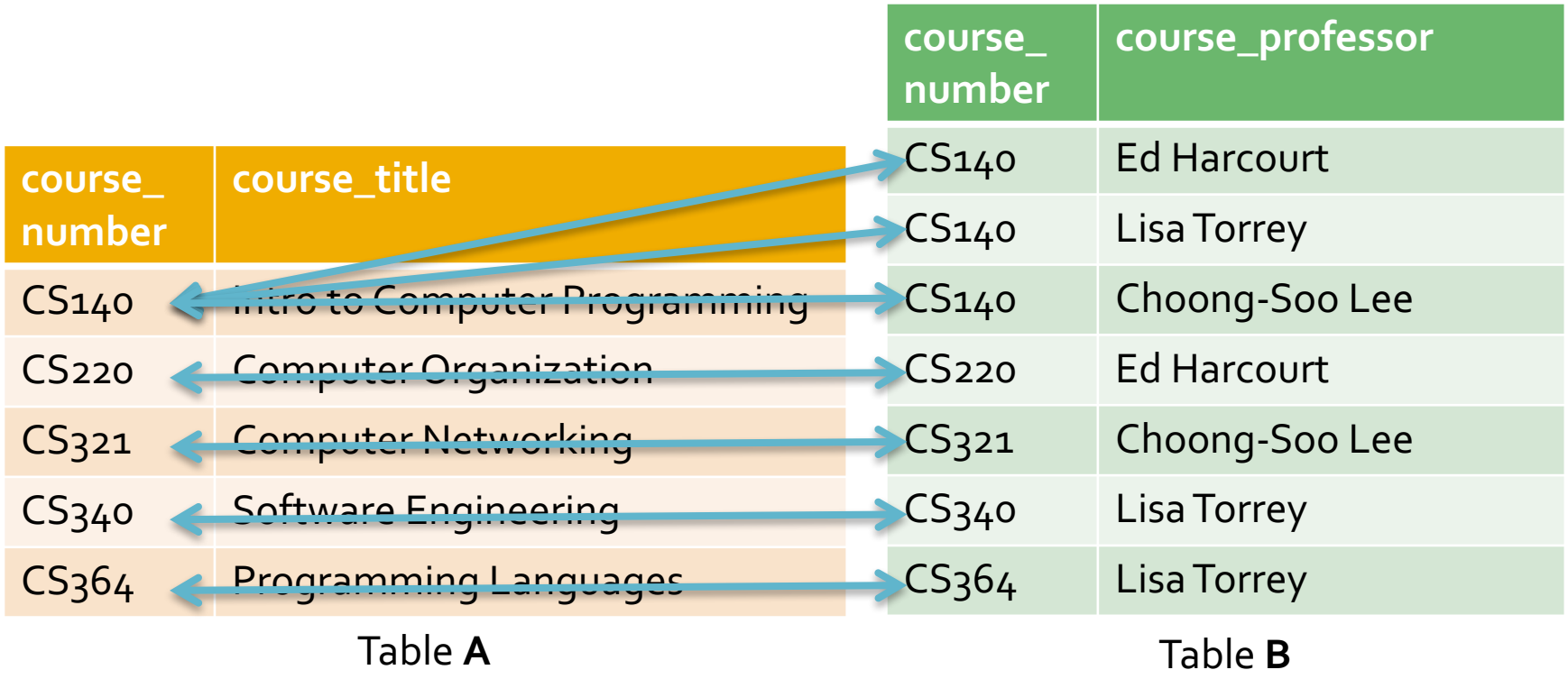
- Cartesian Product: the set of all possible pairs (a, b) where $a \in A$ and $b \in B$. (cross product)

$$A = \{1, 2, 3\} \qquad B = \{x, y, z\}$$

$$\begin{array}{lll} (1, x) & (1, y) & (1, z) \\ (2, x) & (2, y) & (2, z) \\ (3, x) & (3, y) & (3, z) \end{array}$$

Database Joins

- Cartesian Product has superfluous rows
- Restrict result with predicates
- `A.course_number=B.course_number`



course_number	course_title
CS140	Intro to Computer Programming
CS220	Computer Organization
CS321	Computer Networking
CS340	Software Engineering
CS364	Programming Languages

Table **A**

course_number	course_professor
CS140	Ed Harcourt
CS140	Lisa Torrey
CS140	Choong-Soo Lee
CS220	Ed Harcourt
CS321	Choong-Soo Lee
CS340	Lisa Torrey
CS364	Lisa Torrey

Table **B**

GPU Implementation

- nVidia Cuda API
  - Provides toolkit needed to run code on GPU
  - C-style language
- Extends "Virginian" Database
  - Demonstrated speedup for single-table queries
  - UVA; NEC Labs, New Jersey
- Virtual Machine-based Implementation
  - Similar to SQLite Database
  - Used by programs, such as Firefox

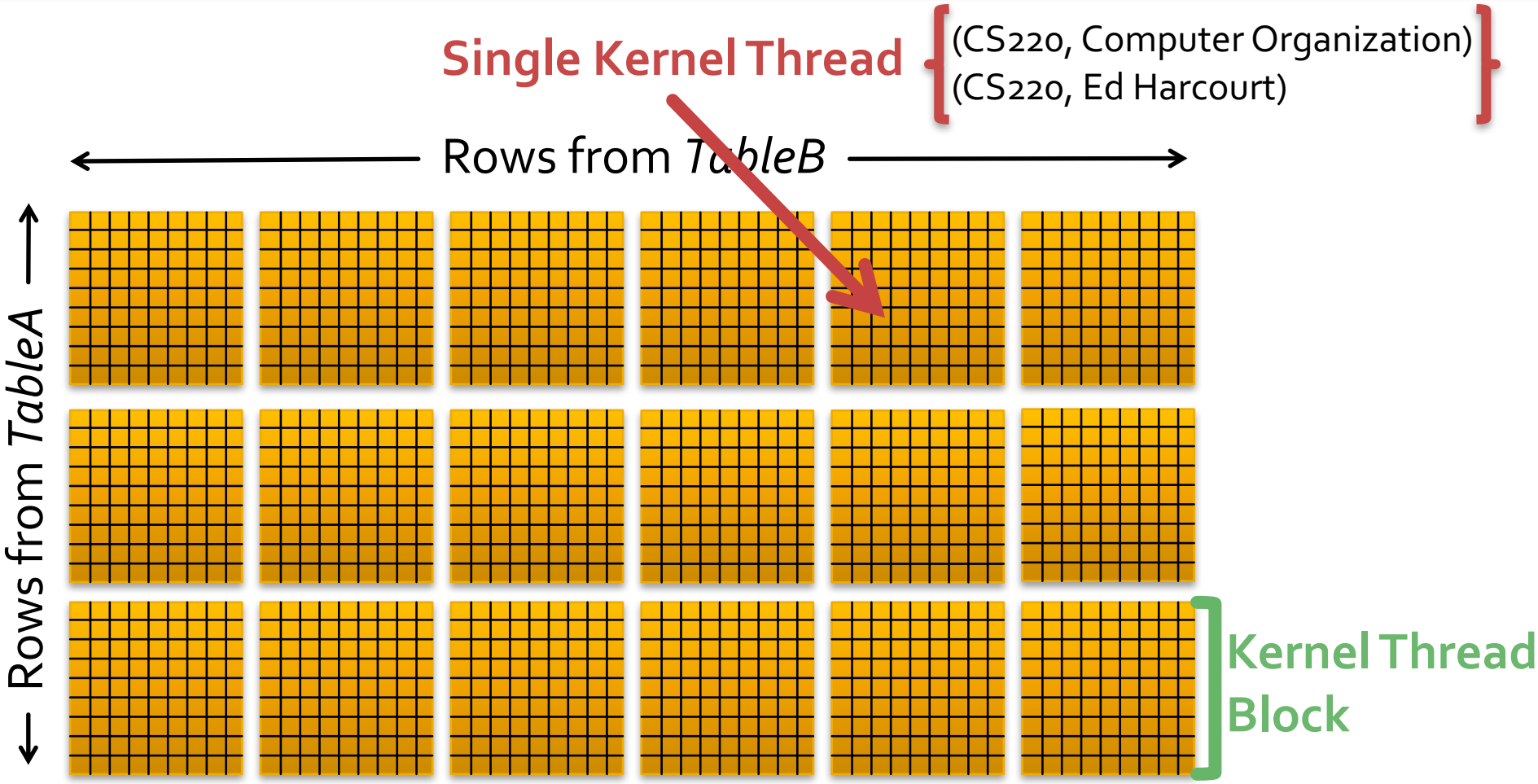
Step 1: Compile SQL

```
SELECT A.course_number, A.course_title, B.course_professor
    FROM A,B WHERE A.course_number=B.course_number;
```

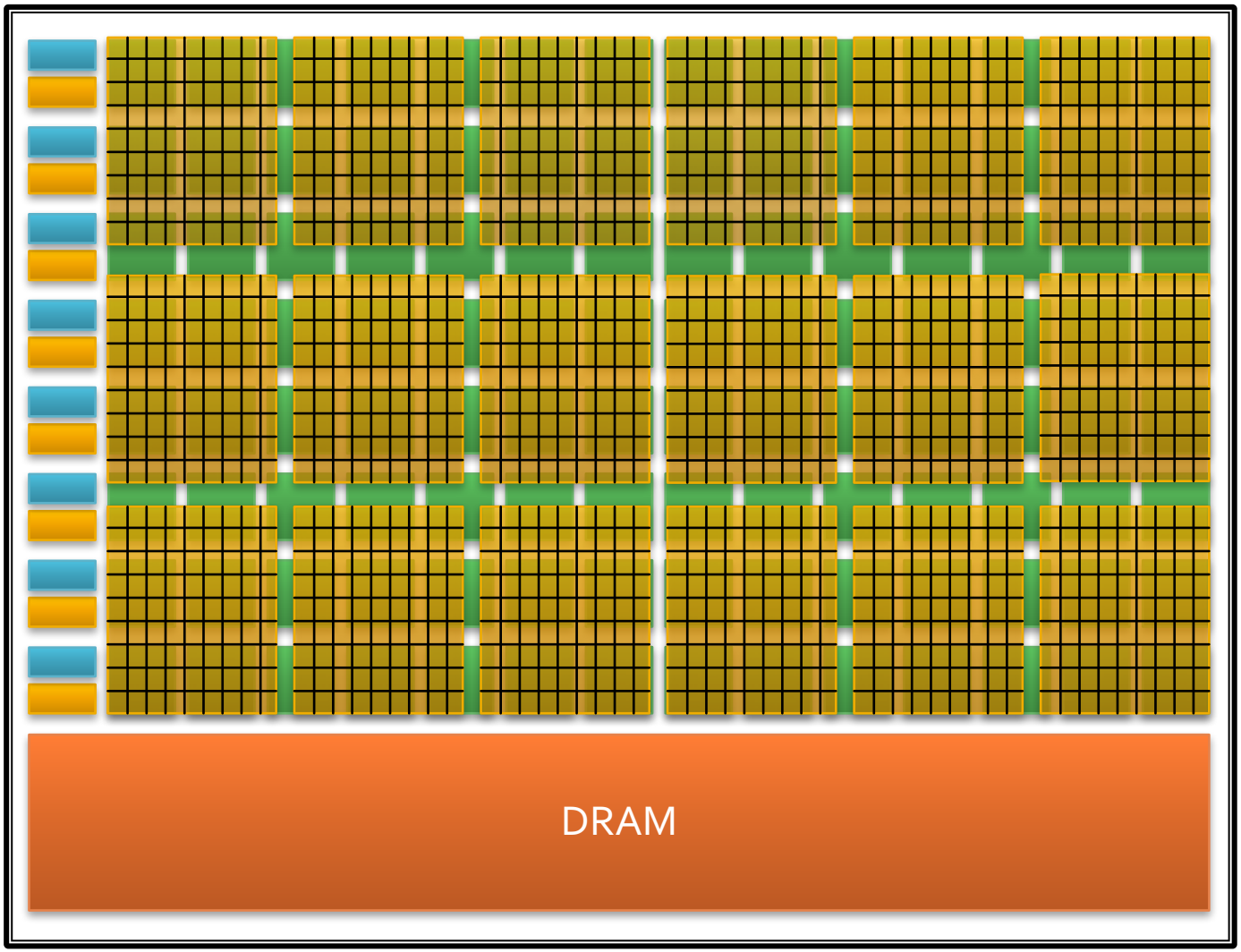
```
 0: Table          0    0    0              0
 1: Table          1    0    1              0
 2: ResultColumn   0    0    0             id
 3: ResultColumn   0    0    0       uniformi
 4: ResultColumn   0    0    0       normali5
 5: Parallel       0    0   16              0
 6: Column         3    0    1              0
 7: Integer        0   60    0              0
 8: Le             3    0   14              0
 9: Column         4    1    0              0
10: Integer        1    0    0              0
11: Lt             4    1   13              1
12: Invalid        0    0    0              0
13: Rowid          2    0    0              0
14: Result         2    3    0              0
15: Converge       0    0    0              0
16: Finish         0    0    0              0
```

Compute for each row

Step 3: Set up GPU Kernel



GPU Kernels

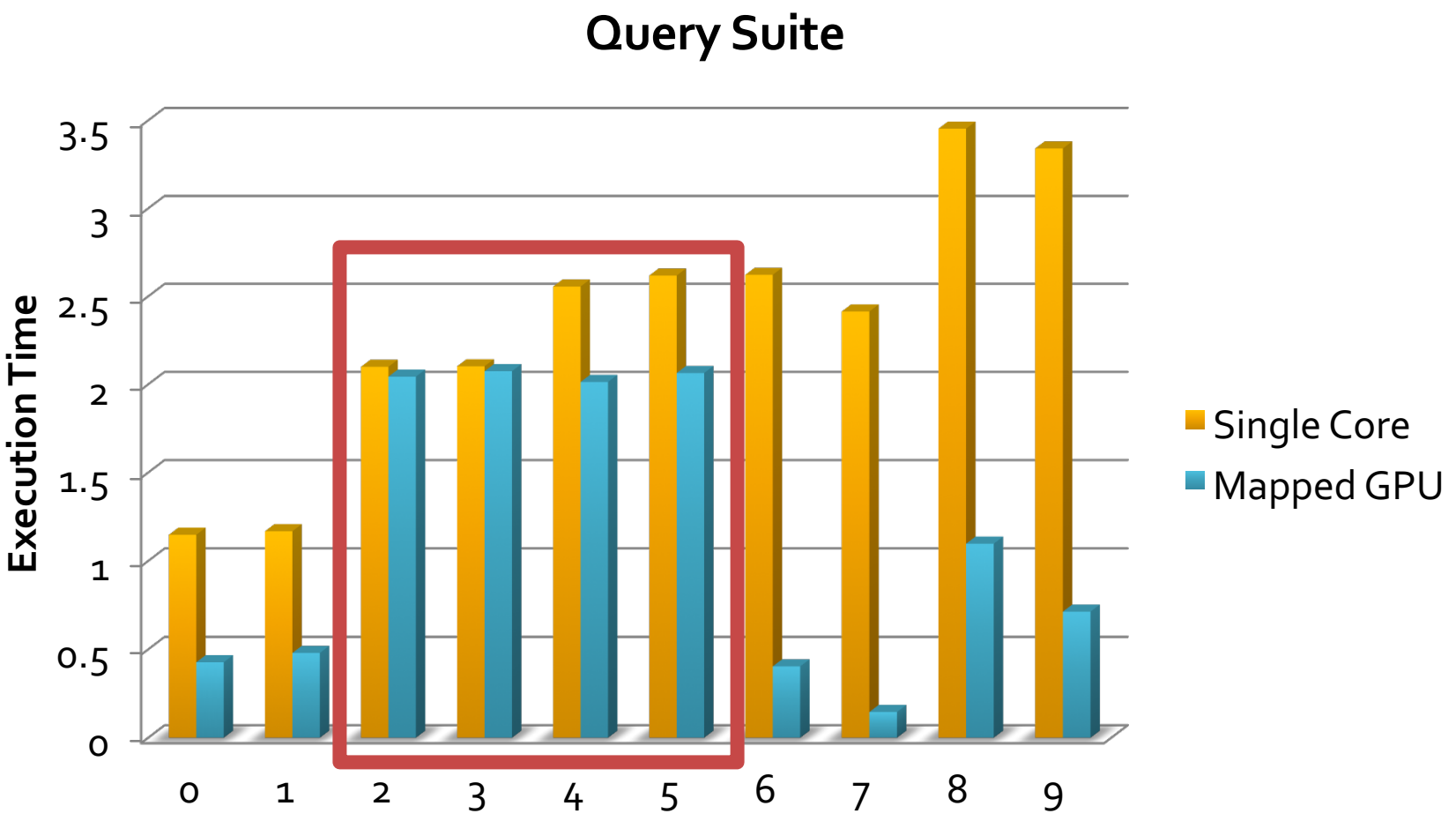


Step 4: Execute

Test Machine

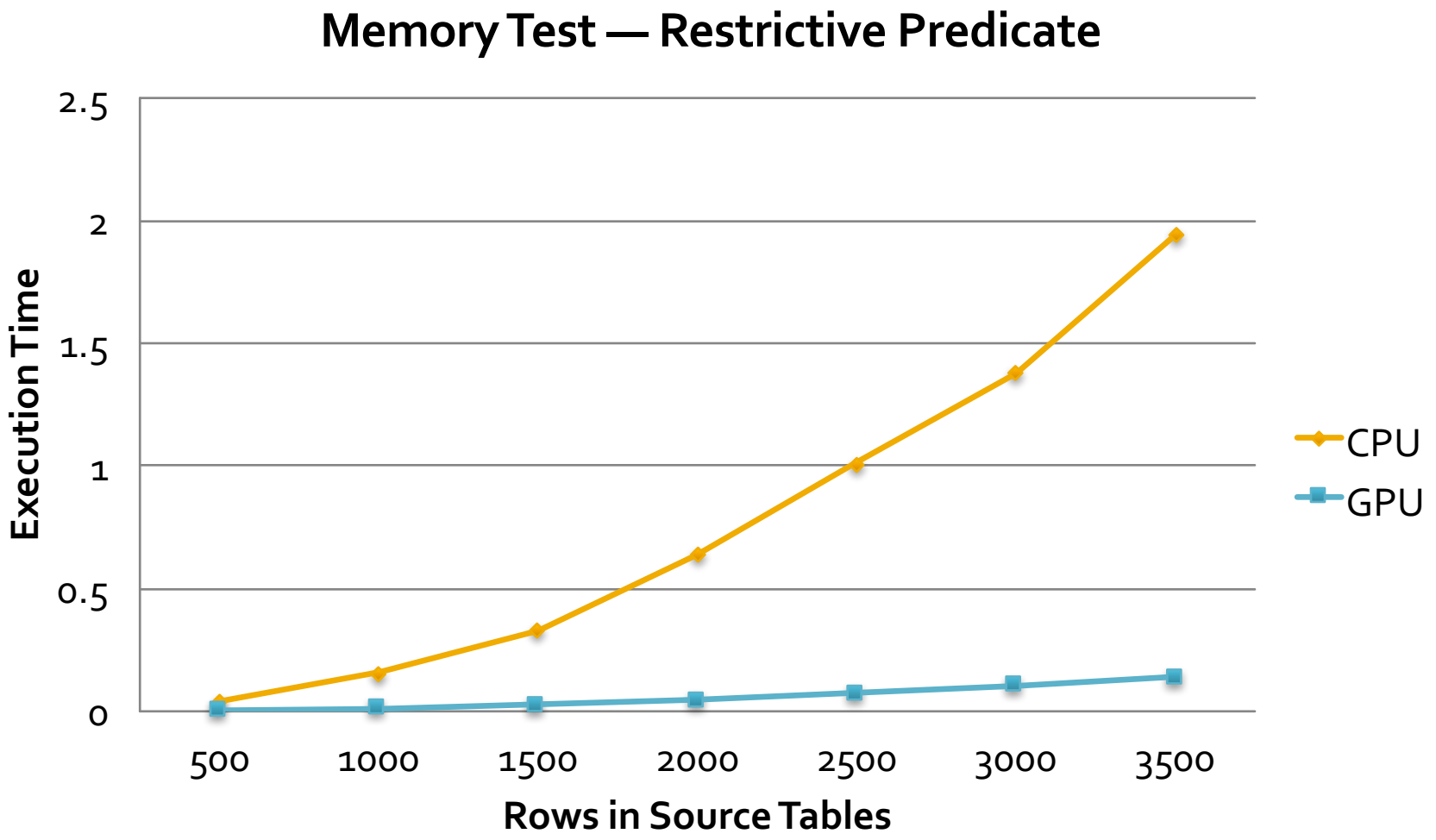
- Intel Core® i7 920 CPU
  - Eight hardware threads @ 2.66 GHz
  - 12 GB RAM
  - Linux 3.2.0-61
- nVidia GTX460 GPU (x2)
  - 336 Cuda Cores
  - 1 GB Memory

Results


Query Suite
Execution Time
3.5
3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8
9
Single Core
Mapped GPU

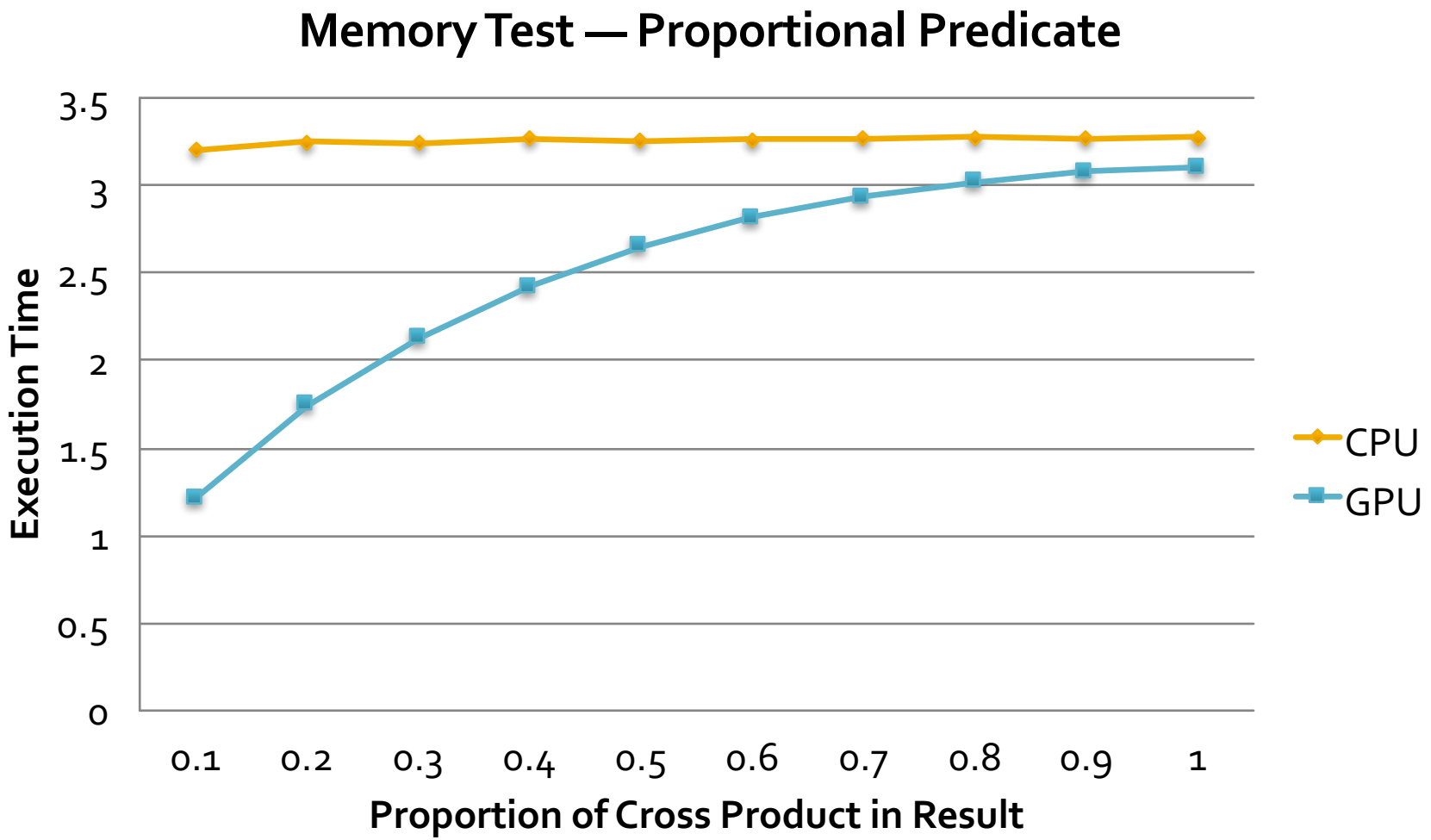
Results

Increase source table size; Small result table size

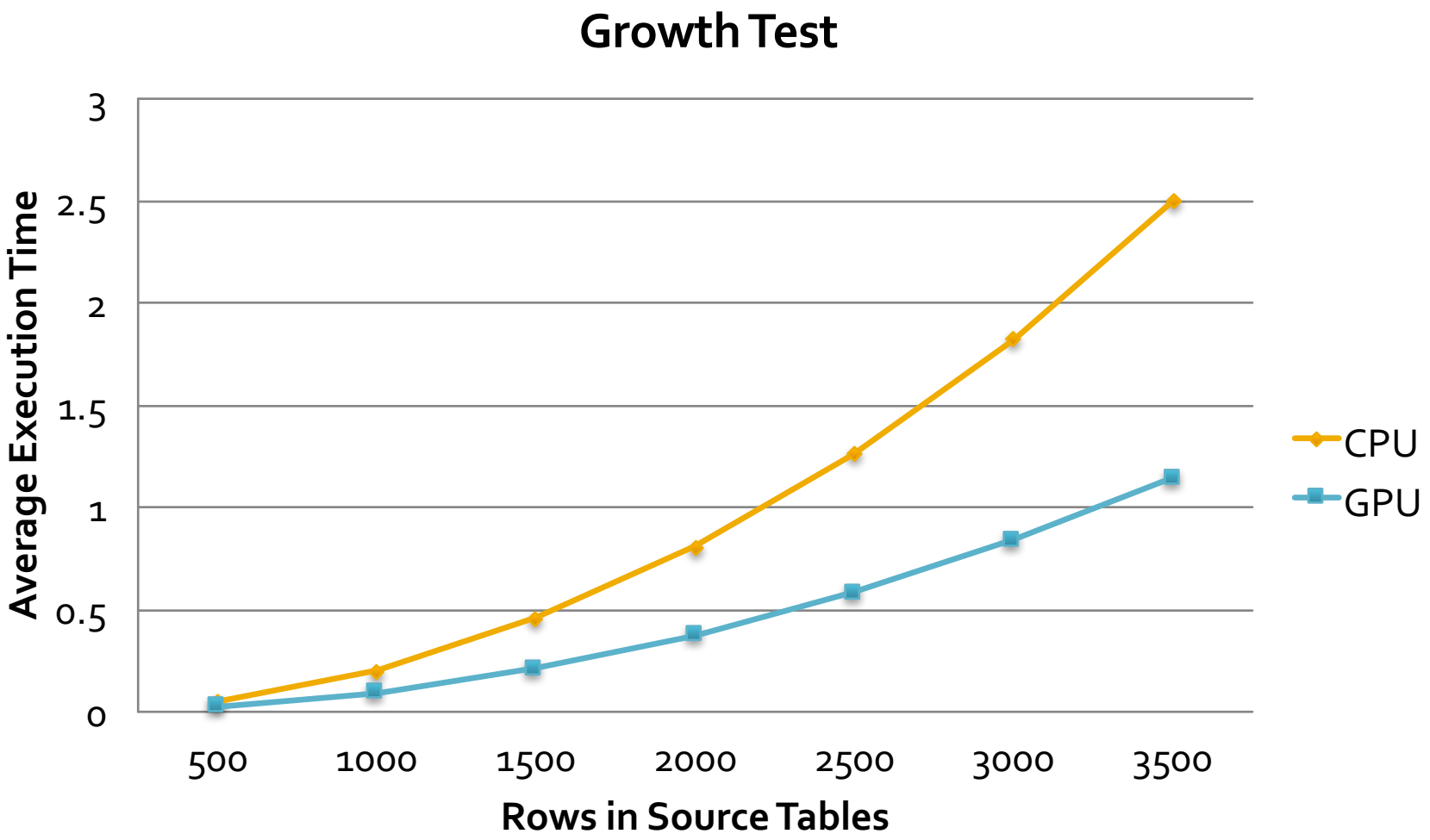


Results

Constant source table size; Increase result table size



Average Speedup



Conclusions

- Memory is the bottleneck
  - Small result tables = Faster execution time
- For *reasonable* queries, the GPU is more efficient
- On average, queries execute **TWICE** as fast on GPU

References

[1] P. Bakkum and S. Chakradhar. Efficient data management for GPU databases. Technical report, NEC Laboratories America, Princeton, NJ.

[2] P. Bakkum and K. Skadron. Accelerating SQL database operations on a GPU with Cuda. In Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units, GPGPU '10, pages 94–103, New York, NY, USA, 2010. ACM.

[3] P. Bakkum and K. Skadron. Accelerating SQL database operations on a GPU with Cuda: Extended results. Technical Report CS-2010-08, University of Virginia Department of Computer Science, May 2010.

[4] R. Fang, B. He, M. Lu, K. Yang, N. K. Govindaraju, Q. Luo, and P. V. Sander. GPUQP: Query co- processing using graphics processors. In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD '07, pages 1061–1063, New York, NY, USA, 2007. ACM.

[5] B. He, M. Lu, K. Yang, R. Fang, N. K. Govindaraju, Q. Luo, and P. V. Sander. Relational query coprocessing on graphics processors. ACM Trans. Database Syst., 34(4):21:1–21:39, Dec. 2009.

[6] B. He, K. Yang, R. Fang, M. Lu, N. Govindaraju, Q. Luo, and P. Sander. Relational joins on graphics processors. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, pages 511–524, New York, NY, USA, 2008. ACM.

[7] M. Pietron, P. l Russek, and K. Wiatr. Accelerating select where and select join queries on a GPU. Computer Science, 14(2):243, 2013.

[8] J. Sanders and E. Kandrot. CUDA by Example: An Introduction to General-Purpose GPU Programming. Pearson Education, 2010.

Questions?