Toward GPUs being mainstream in analytic processing

An initial argument using simple scan-aggregate queries

Jason Power || Yinan Li || Mark D. Hill
Jignesh M. Patel || David A. Wood
<powerjg@cs.wisc.edu>

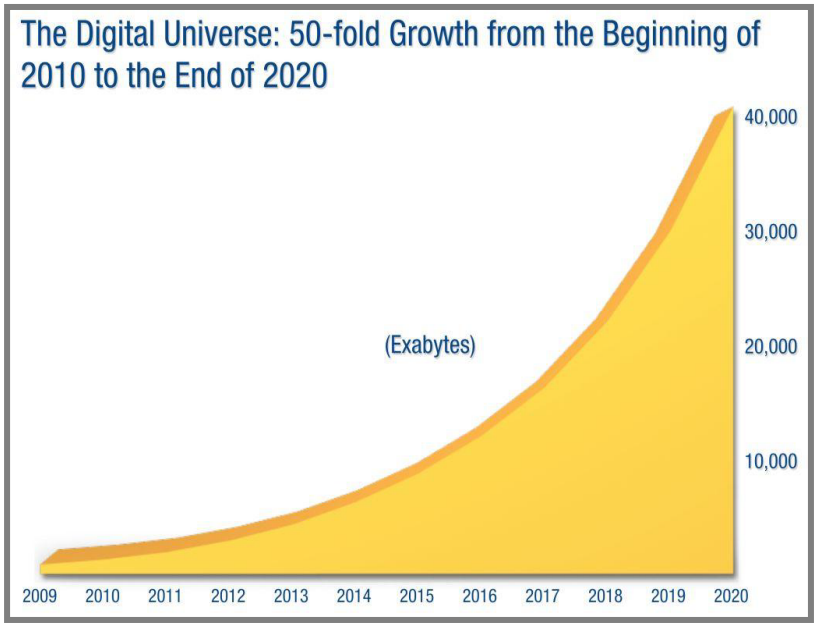
DaMoN 2015

Summary

- GPUs are energy efficient
  - Discrete GPUs unpopular for DBMS
  - New integrated GPUs solve the problems
- Scan-aggregate GPU implementation
  - Wide bit-parallel scan
  - Fine-grained aggregate GPU offload
- Up to 70% energy savings over multicore CPU
  - Even more in the future

Analytic Data is Growing

- Data is growing rapidly
- Analytic DBs increasingly important



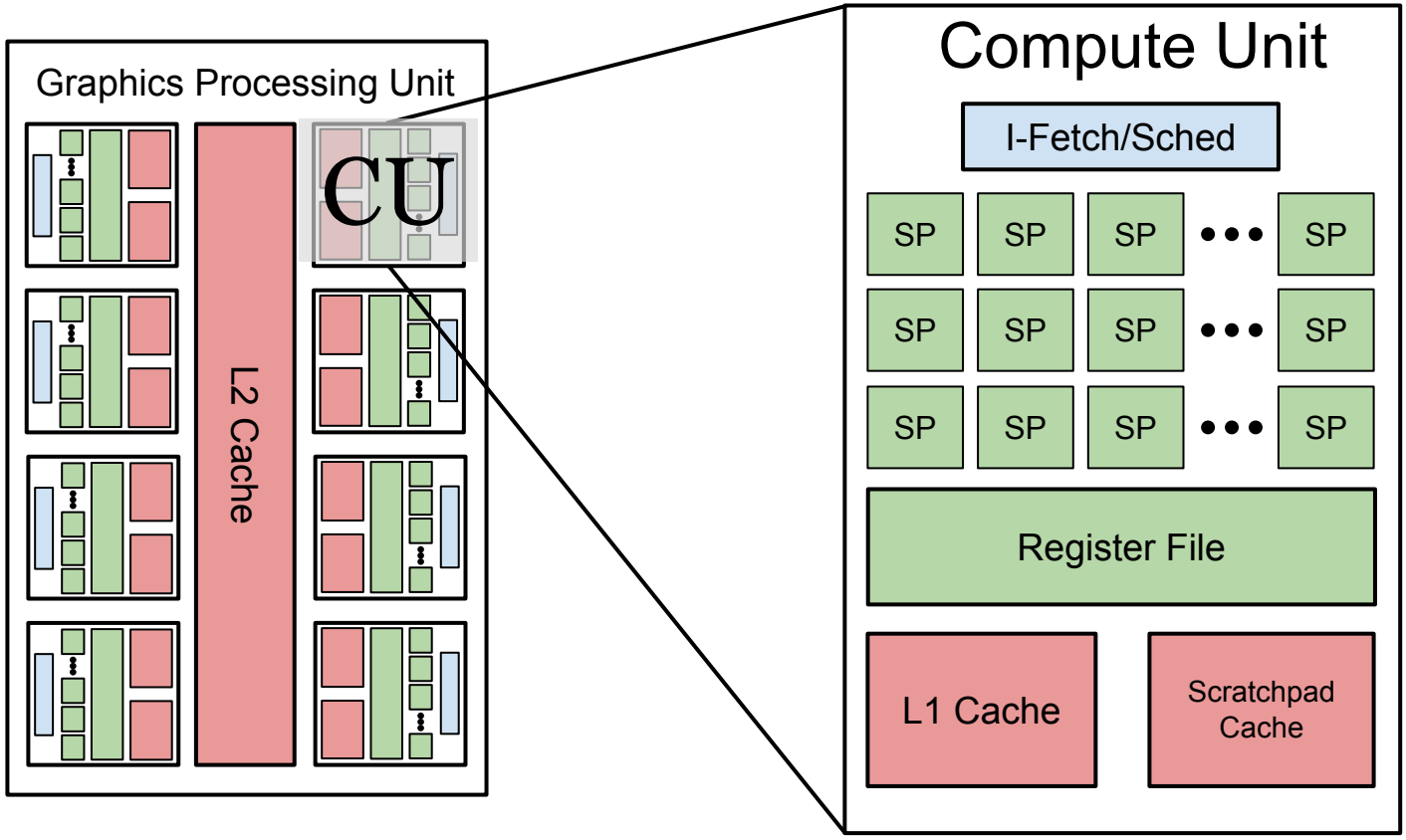
Source: IDC's Digital Universe Study. 2012.

Want: **High performance** Need: **Low energy**

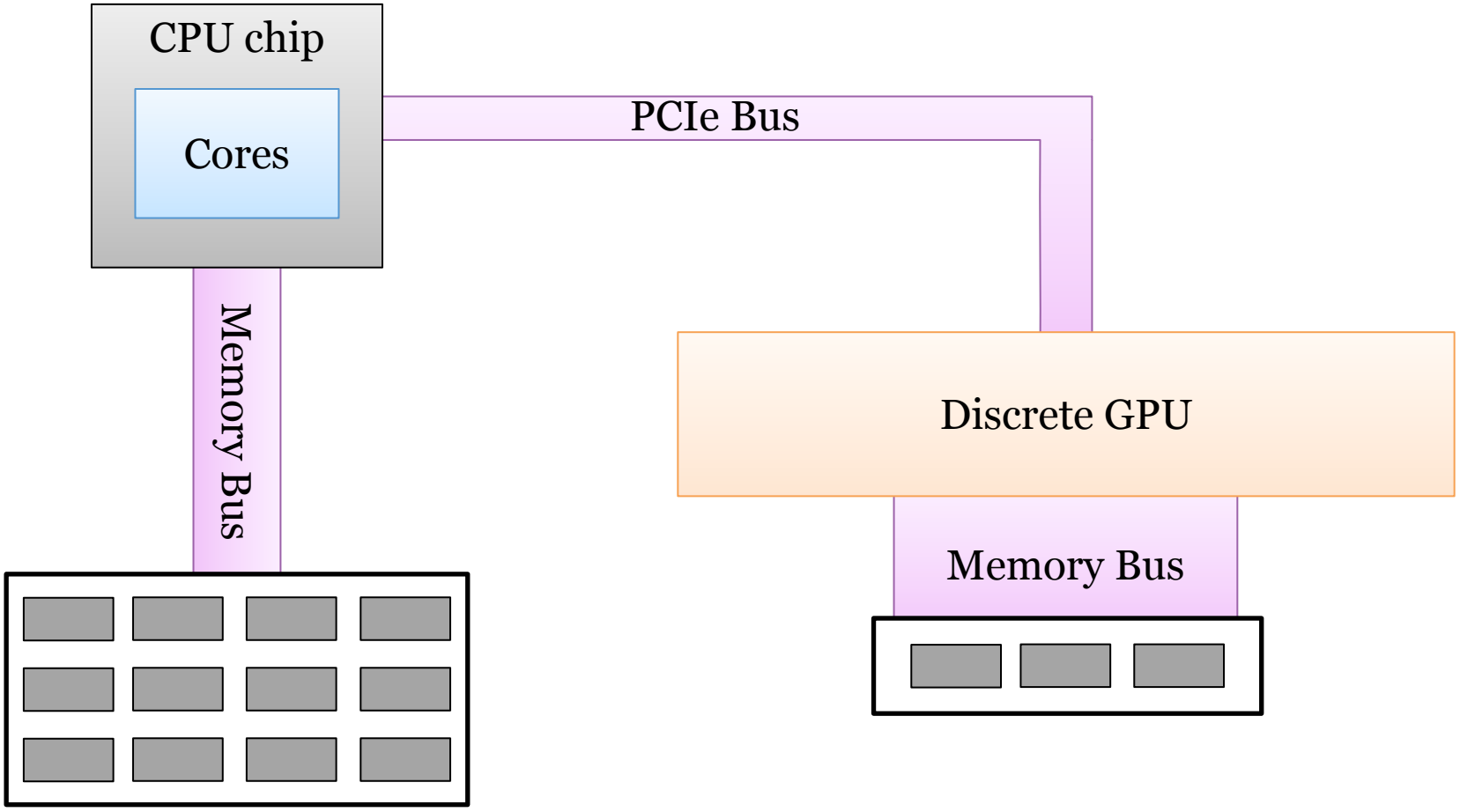
GPUs to the Rescue?

- GPUs are becoming more general
  - Easier to program
  - Integrated GPUs are everywhere
- GPUs show great promise [Govindaraju '04, He '14, He '14, Kaldewey '12, Satish '10, and many others]
  - Higher performance than CPUs
  - Better energy efficiency
- Analytic DBs look like GPU workloads

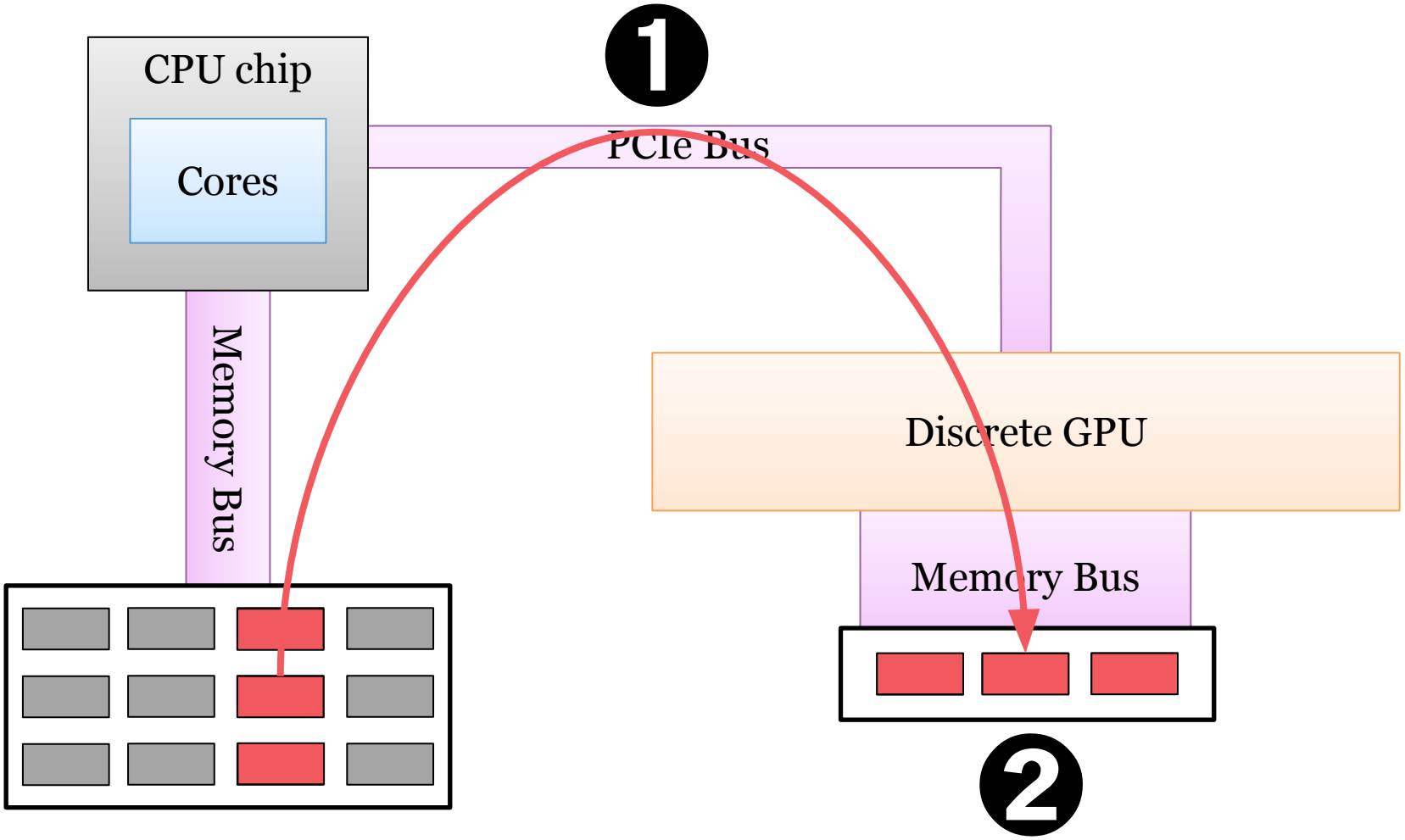
GPU Microarchitecture



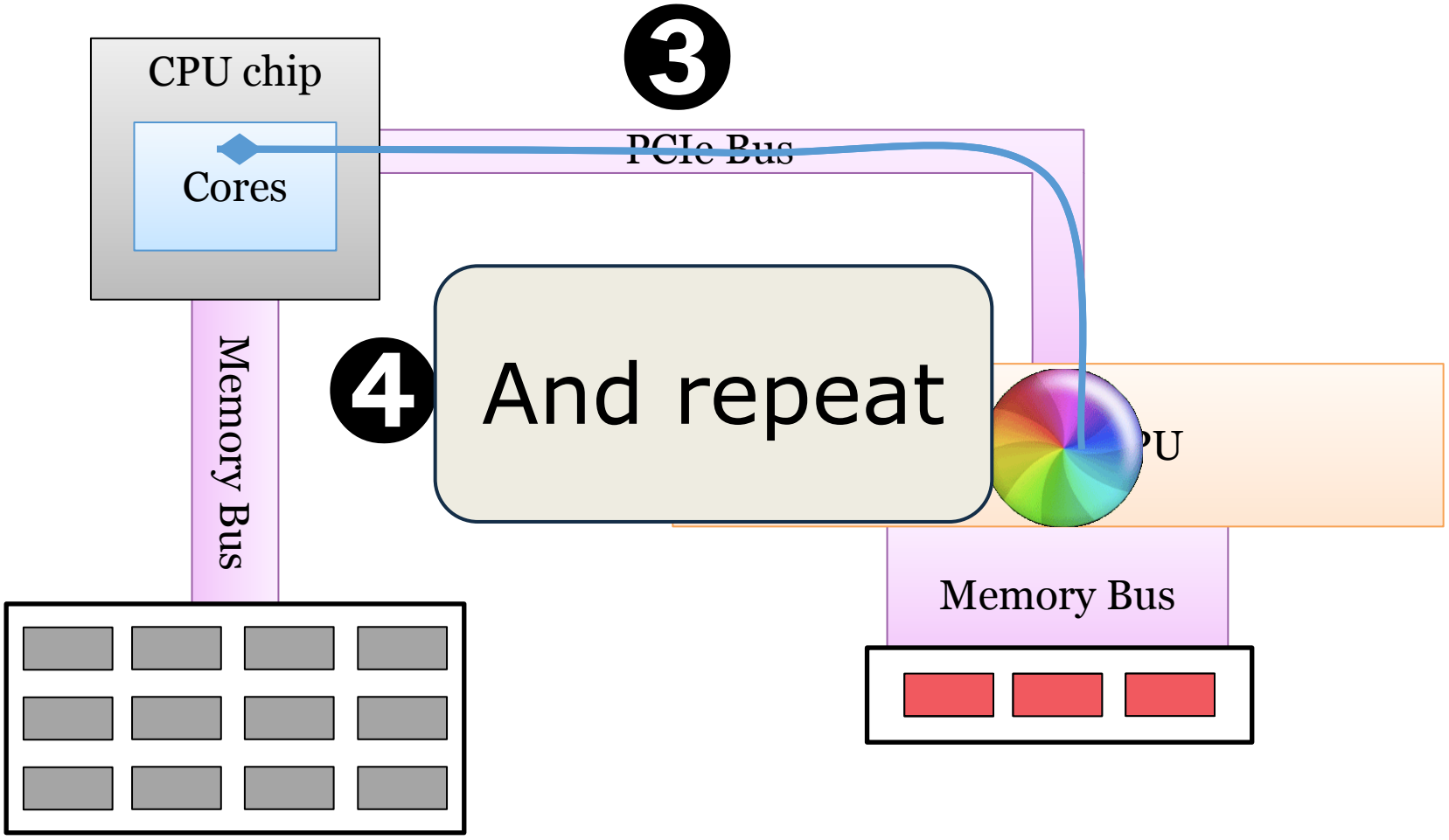
Discrete GPUs



Discrete GPUs



Discrete GPUs



Discrete GPUs

❶ Copy data over PCIe

- Low bandwidth
- High latency

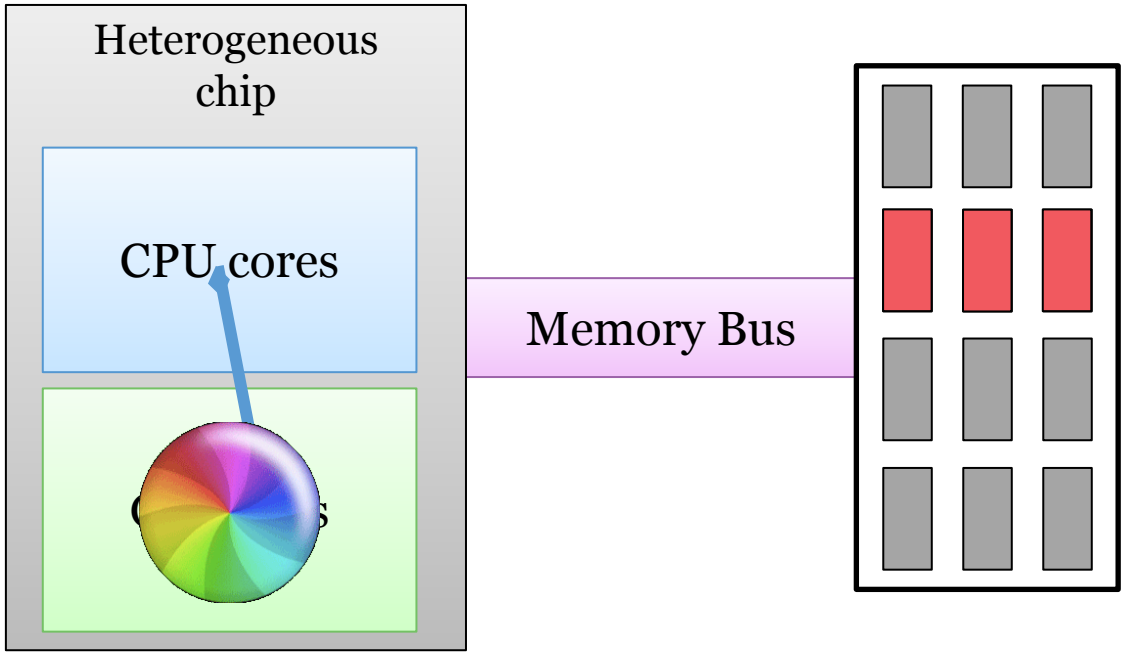
❷ Small working memory

❸ High latency user→kernel calls

❹ Repeated many times

98% of time spent not computing

Integrated GPUs



Heterogeneous System Arch.

- API for tightly-integrated accelerators
- Industry support
  - Initial hardware support today
  - HSA foundation (AMD, ARM,Qualcomm, others)
- No need for data copies
  - Cache coherence and shared address space
- No OS kernel interaction
  - User-mode queues

Outline

- Background
- **Algorithms**
  - **Scan**
  - **Aggregate**
- Results

Analytic DBs

- Resident in main-memory
- Column-based layout
- WideTable & BitWeaving [Li and Patel '13 & '14]
  - Convert queries to mostly scans by pre-joining tables
  - Fast scan by using sub-word parallelism
  - Similar to industry proposals [SAP Hana, Oracle Exalytics, IBM DB2 BLU]
- Scan-aggregate queries

Running Example

Shirt Color	Shirt Amount
2	1
2	3
1	1
2	5
3	7
0	2
3	1
1	4
3	2

Color	Code
Red	0
Blue	1
Green	2
Yellow	3

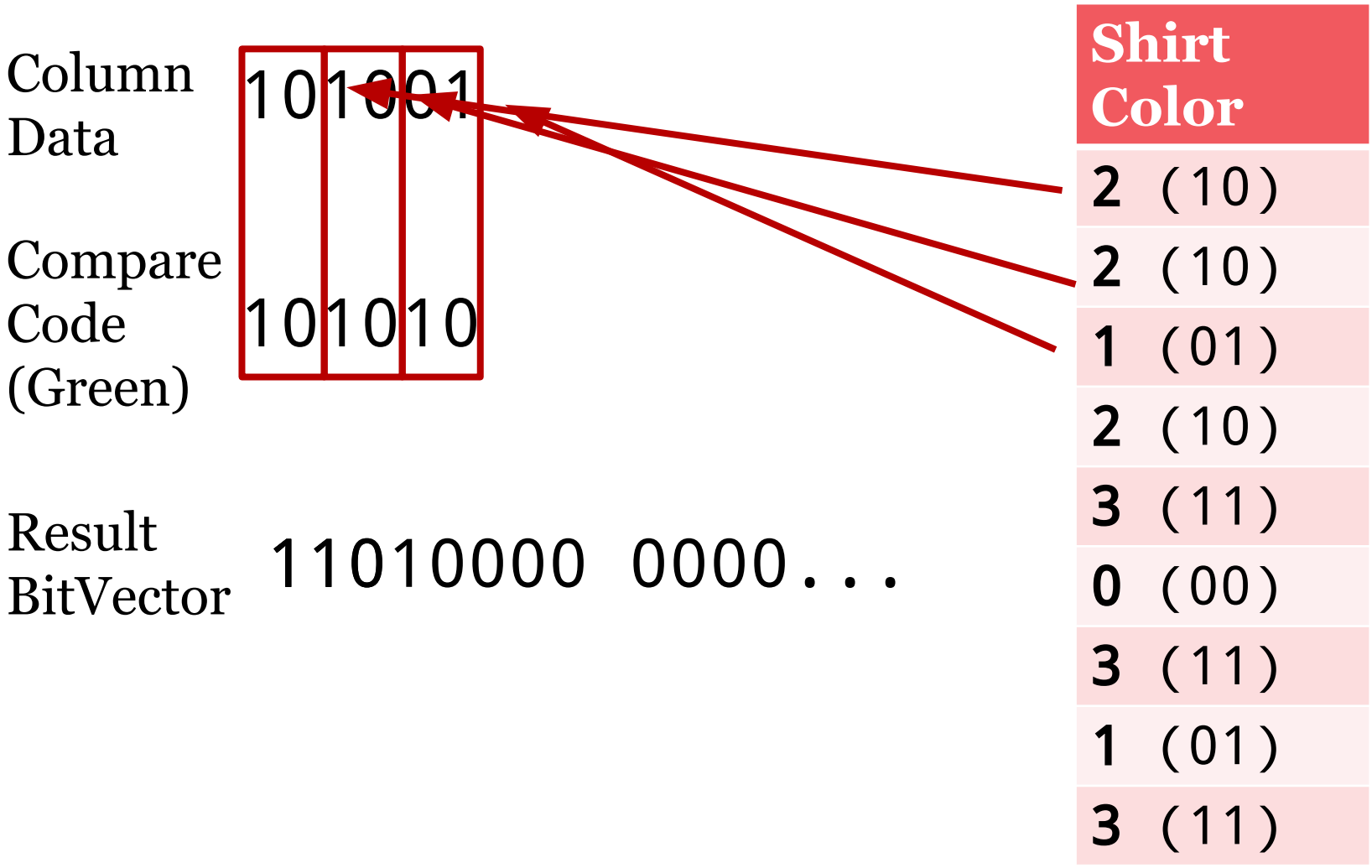
Running Example

Shirt Color	Shirt Amount
2	1
2	3
1	1
2	5
3	7
0	2
3	1
1	4
3	2

Count the number of **green** shirts in the inventory

1. **Scan** the color column for **green** (2)
2. **Aggregate** amount where there is a match

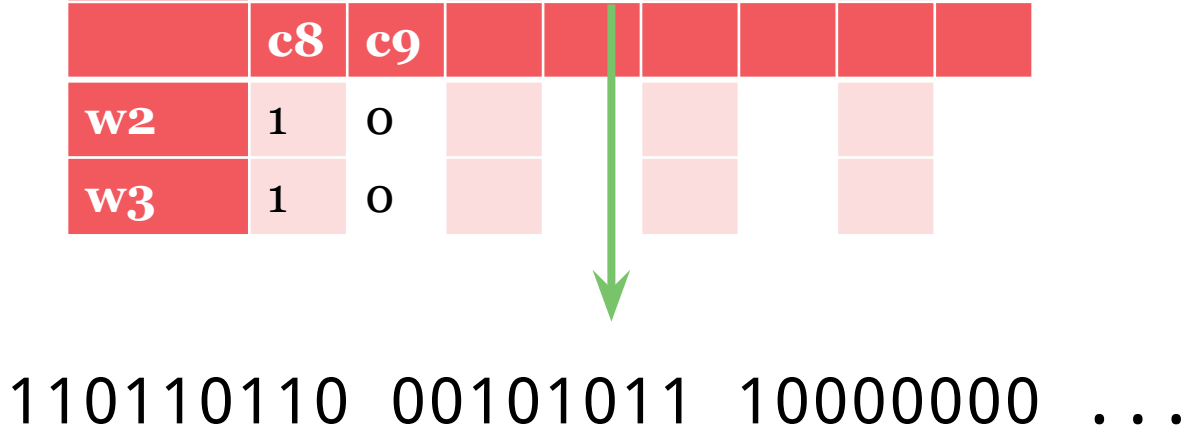
Traditional Scan Algorithm



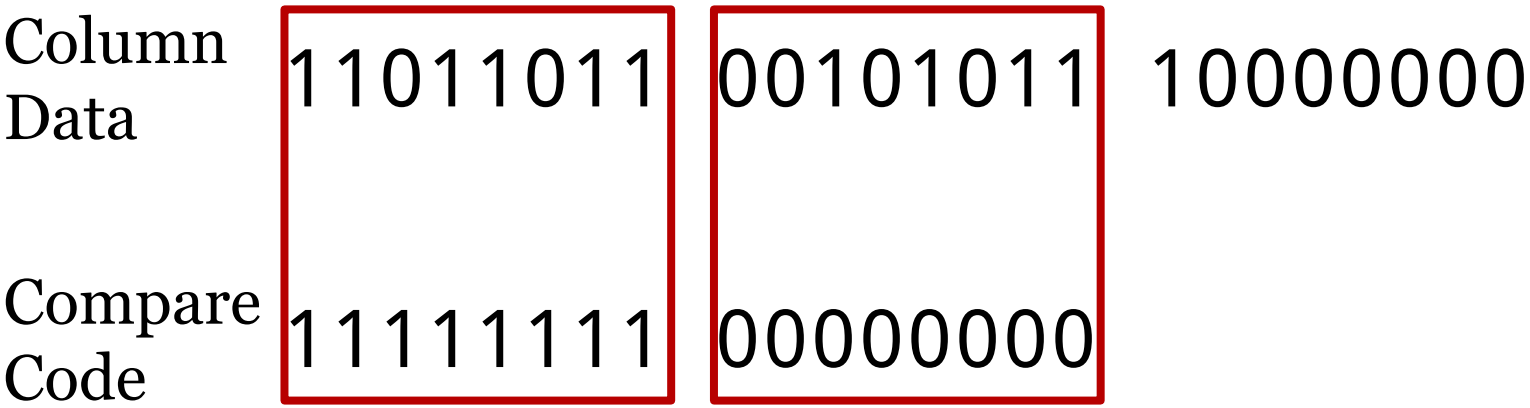
Vertical Layout

	Color
c0	2 (10)
c1	2 (10)
c2	1 (01)
c3	2 (10)
c4	3 (11)
c5	0 (00)
c6	3 (11)
c7	1 (01)
c8	3 (11)
c9	0 (00)

word	c0	c1	c2	c3	c4	c5	c6	c7
w0	1	1	0	1	1	0	1	0
w1	0	0	1	0	1	0	1	1
	c8	c9						
w2	1	0						
w3	1	0						



CPU BitWeaving Scan



CPU width: 64-bits, up to 256-bit SIMD

GPU BitWeaving Scan

Column Data	11011011 00101011 10000000
Compare Code	11111111 11111111 11111111
Result BitVector	11010000 0000...

GPU width: 16,384-bit SIMD

GPU Scan Algorithm

- GPU uses very wide "words"
  - CPU: 64-bits or 256-bits with SIMD
  - GPU: 16,384 bits (256 lanes × 64-bits)
- Memory and caches optimized for bandwidth
- HSA programming model
  - No data copies
  - Low CPU-GPU interaction overhead

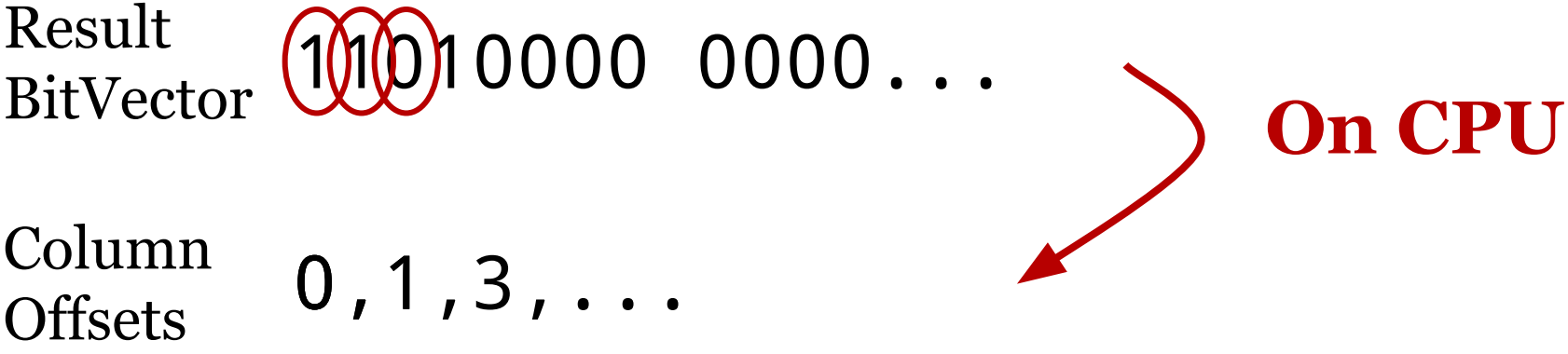
CPU Aggregate Algorithm

Result BitVector 11010000 0000...

Result 1+3+5+...

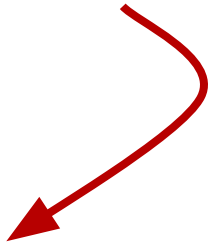
Shirt Amount
1
3
1
5
7
2
1
4
2

GPU Aggregate Algorithm



GPU Aggregate Algorithm

Column Offsets `0,1,3,...`



On GPU

Result `1+3+5+...`

Shirt Amount
1
3
1
5
7
2
1
4
2

Aggregate Algorithm

- Two phases
  - Convert from BitVector to offsets (on CPU)
  - Materialize data and compute (**offload to GPU**)
- Two group-by algorithms (see paper)
- HSA programming model
  - Fine-grained sharing
  - Can offload subset of computation

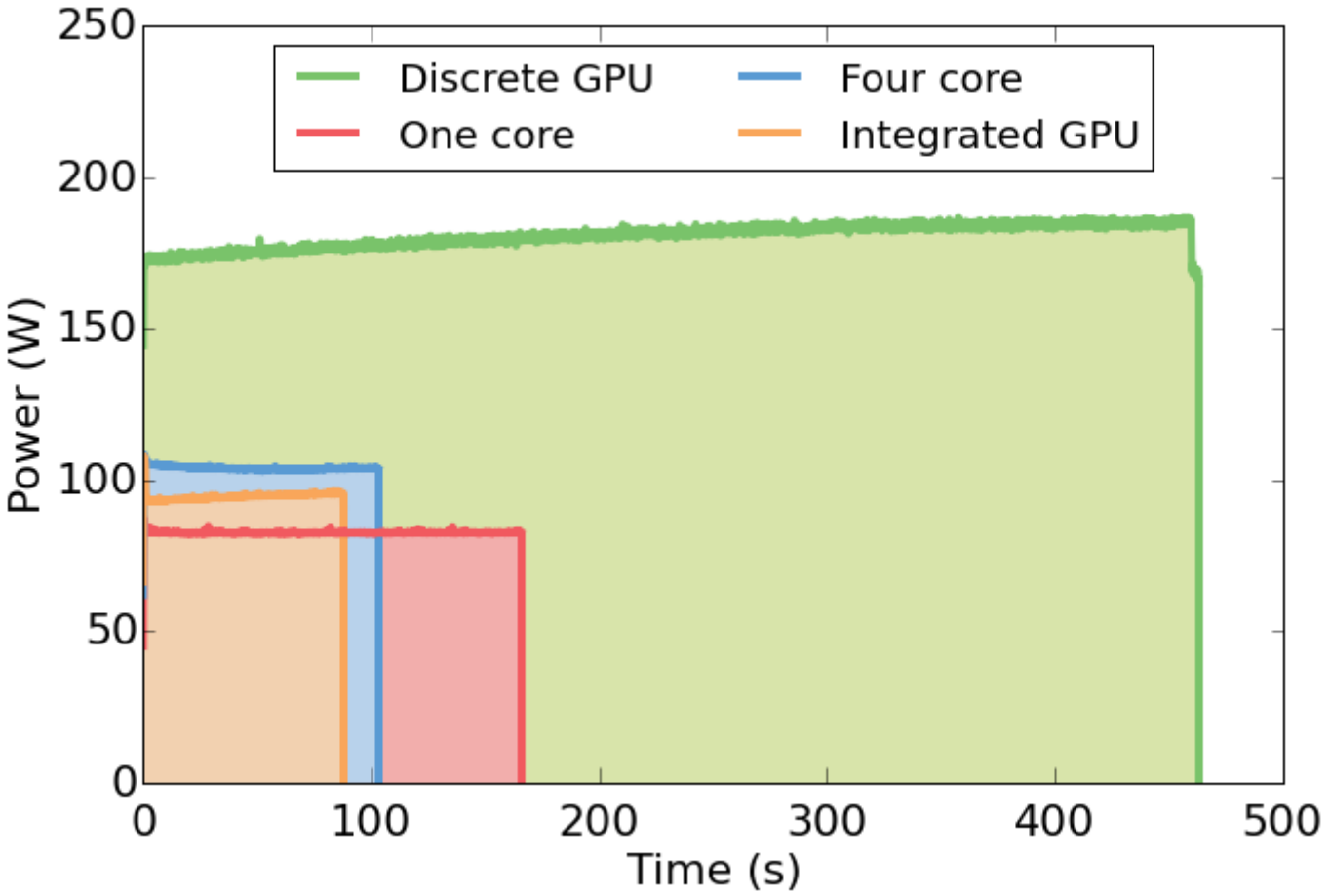
Outline

- Background
- Algorithms
- **Results**

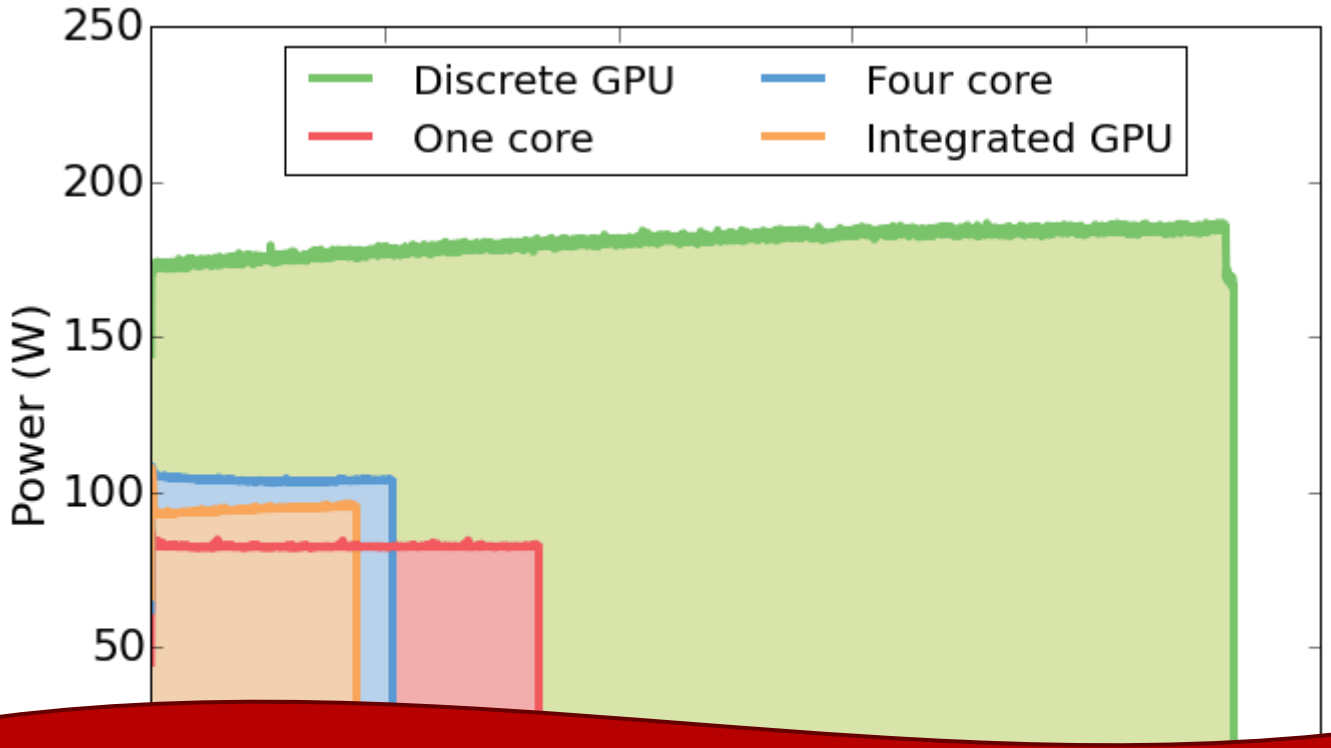
Experimental Methods

- AMD A10-7850
  - 4-core CPU
  - 8-compute unit GPU
  - 16GB capacity, 21 GB/s DDR3 memory
  - Separate discrete GPU
- Watts-Up meter for full-system power
- TPC-H @ scale-factor 10

Scan Performance & Energy



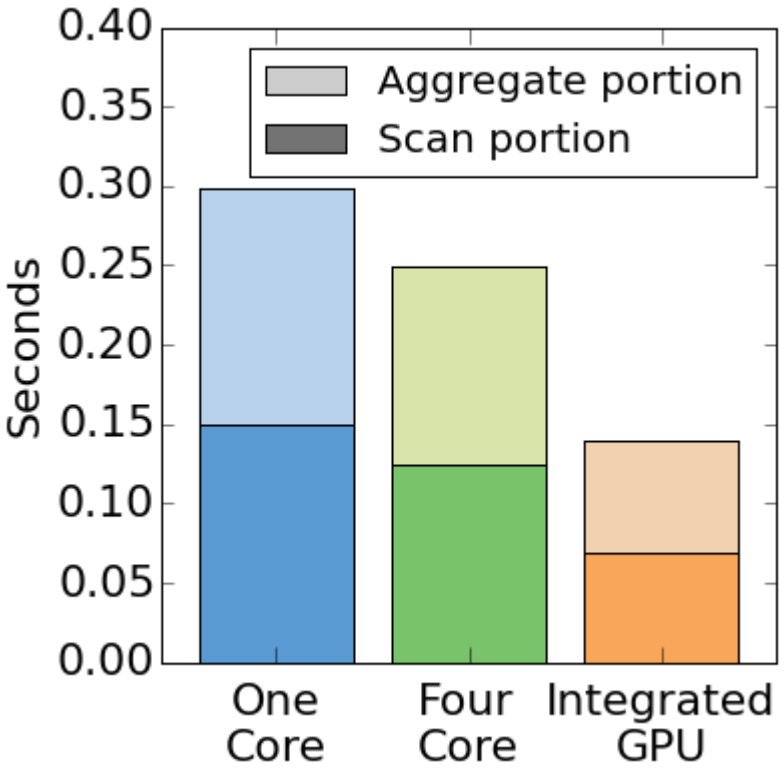
Scan Performance & Energy



Takeaway:
Integrated GPU most efficient for scans

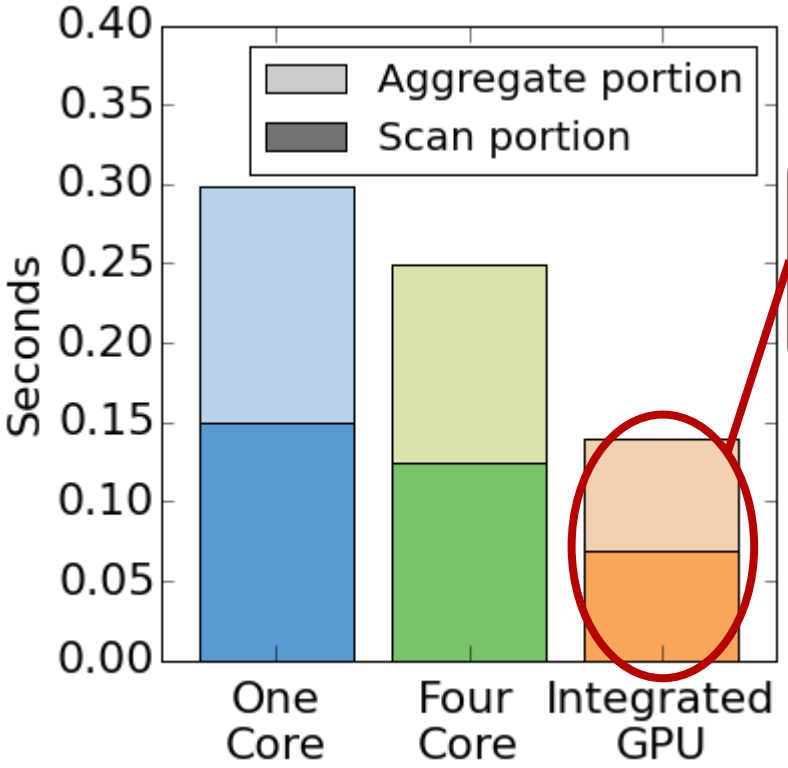
TPC-H Queries

Query 12 Performance

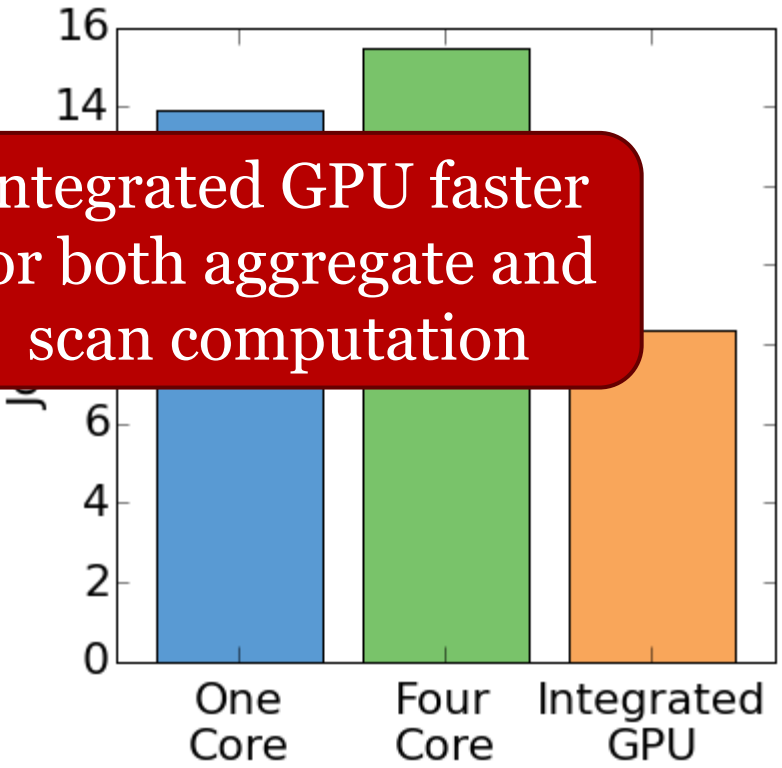


TPC-H Queries

Query 12 Performance



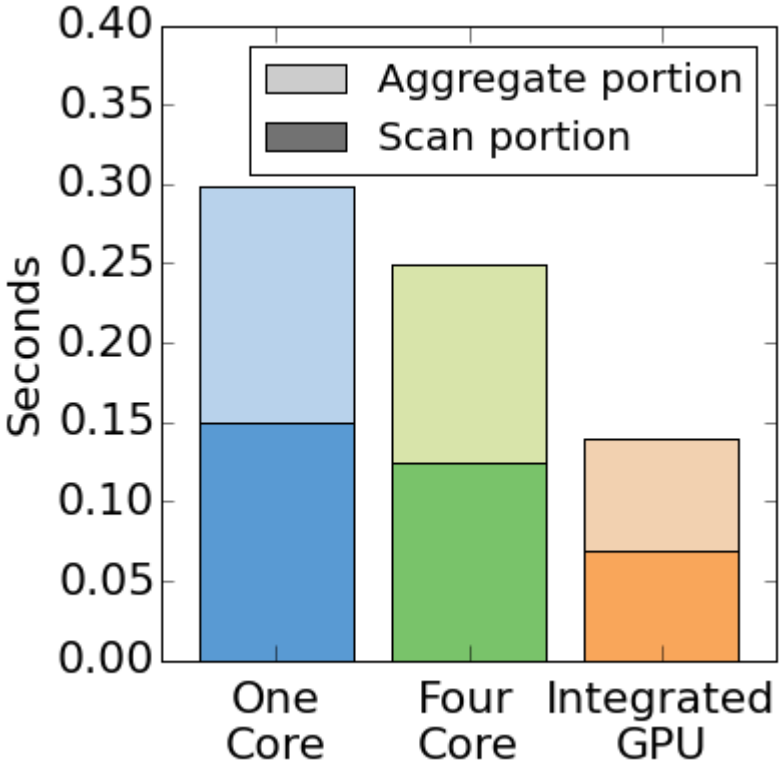
Query 12 Energy



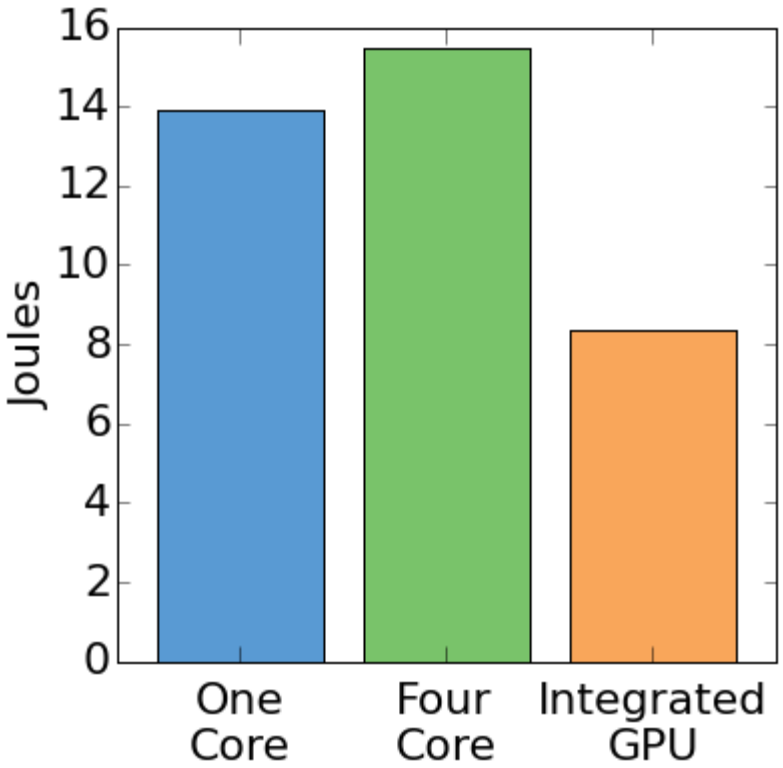
Integrated GPU faster for both aggregate and scan computation

TPC-H Queries

Query 12 Performance

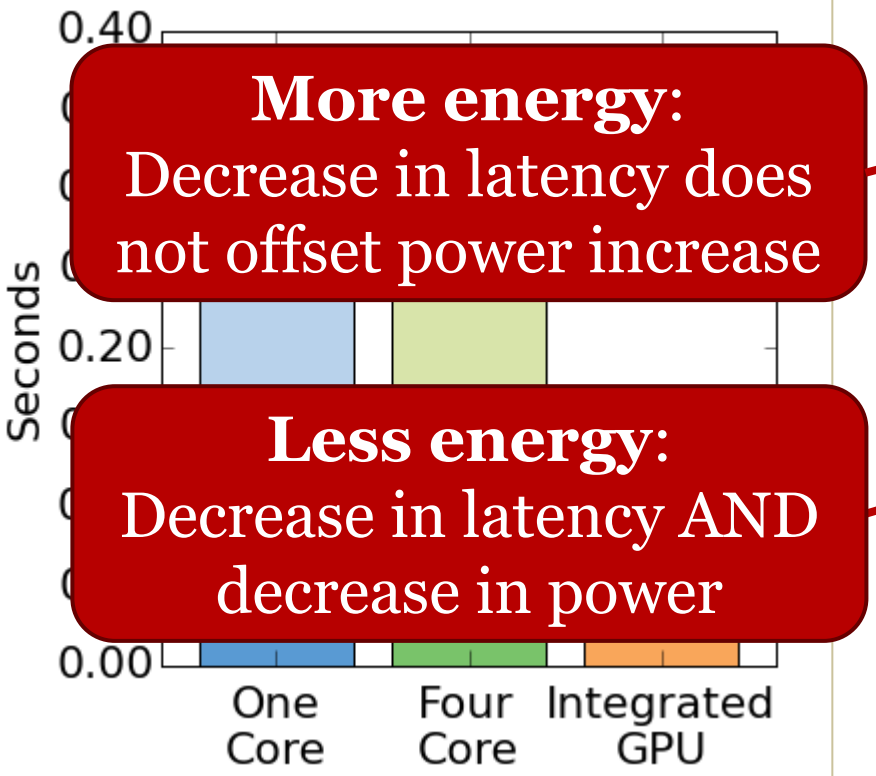


Query 12 Energy



TPC-H Queries

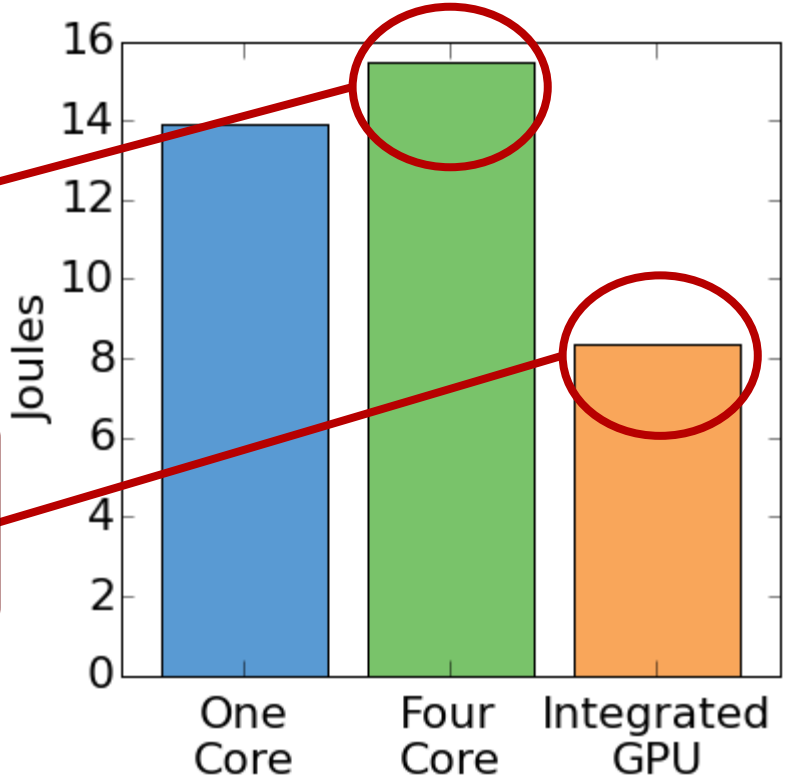
Query 12 Performance



More energy: Decrease in latency does not offset power increase

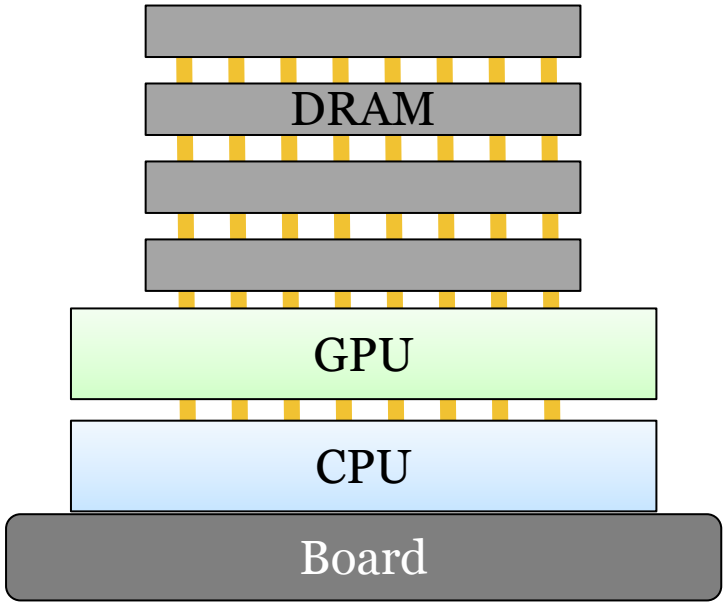
Less energy: Decrease in latency AND decrease in power

Query 12 Energy



Future Die Stacked GPUs

- 3D die stacking
- Same physical & logical integration
- Increased compute
- Increased bandwidth



Power et al.
Implications of 3D GPUs on the Scan Primitive
SIGMOD Record. Volume 44, Issue 1. March 2015

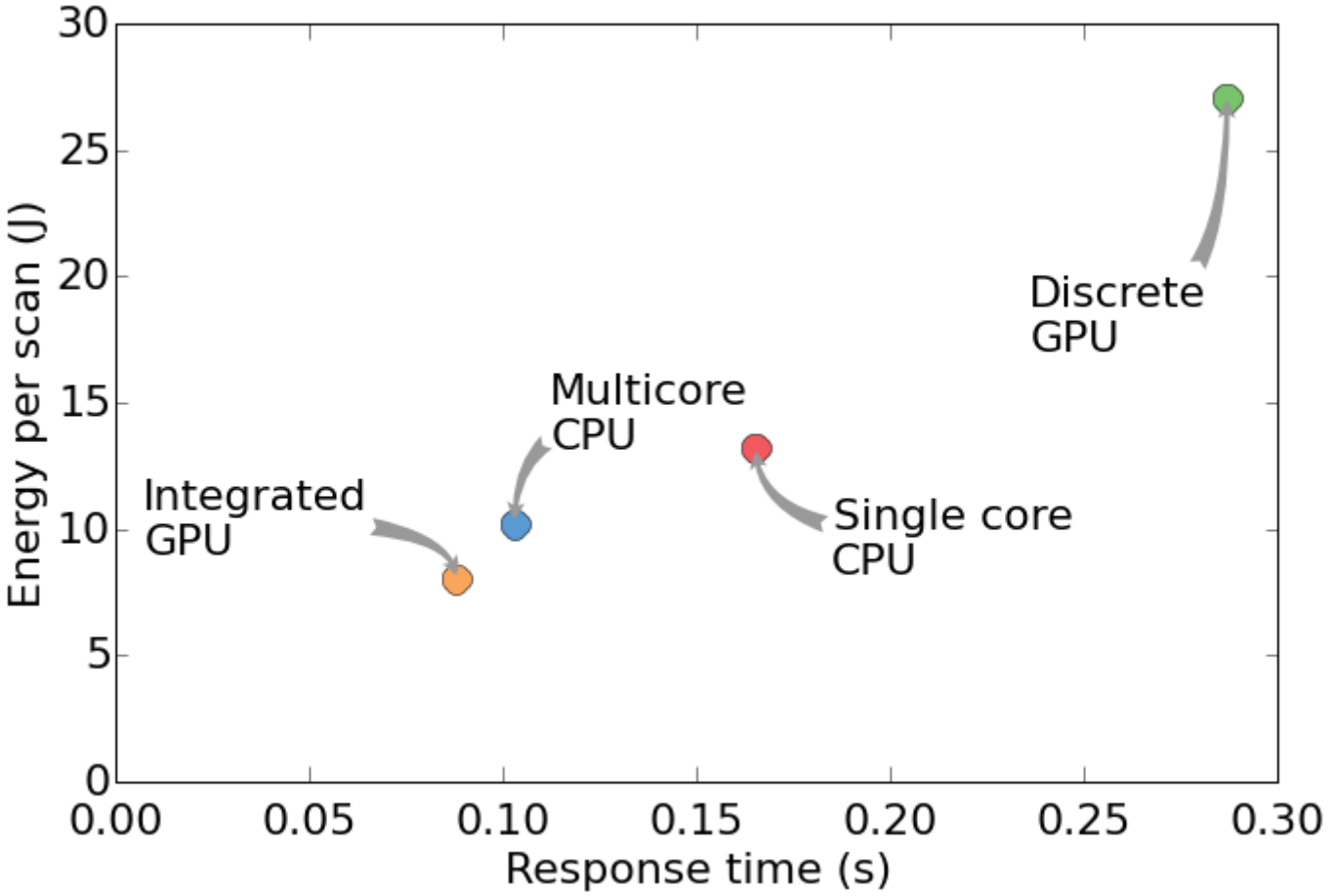
Conclusions

	Discrete GPUs	Integrated GPUs	3D Stacked GPUs
Performance	High ☺	Moderate	High ☺
Memory Bandwidth	High ☺	Low ☹	High ☺
Overhead	High ☹	Low ☺	Low ☺
Memory Capacity	Low ☹	High ☺	Moderate

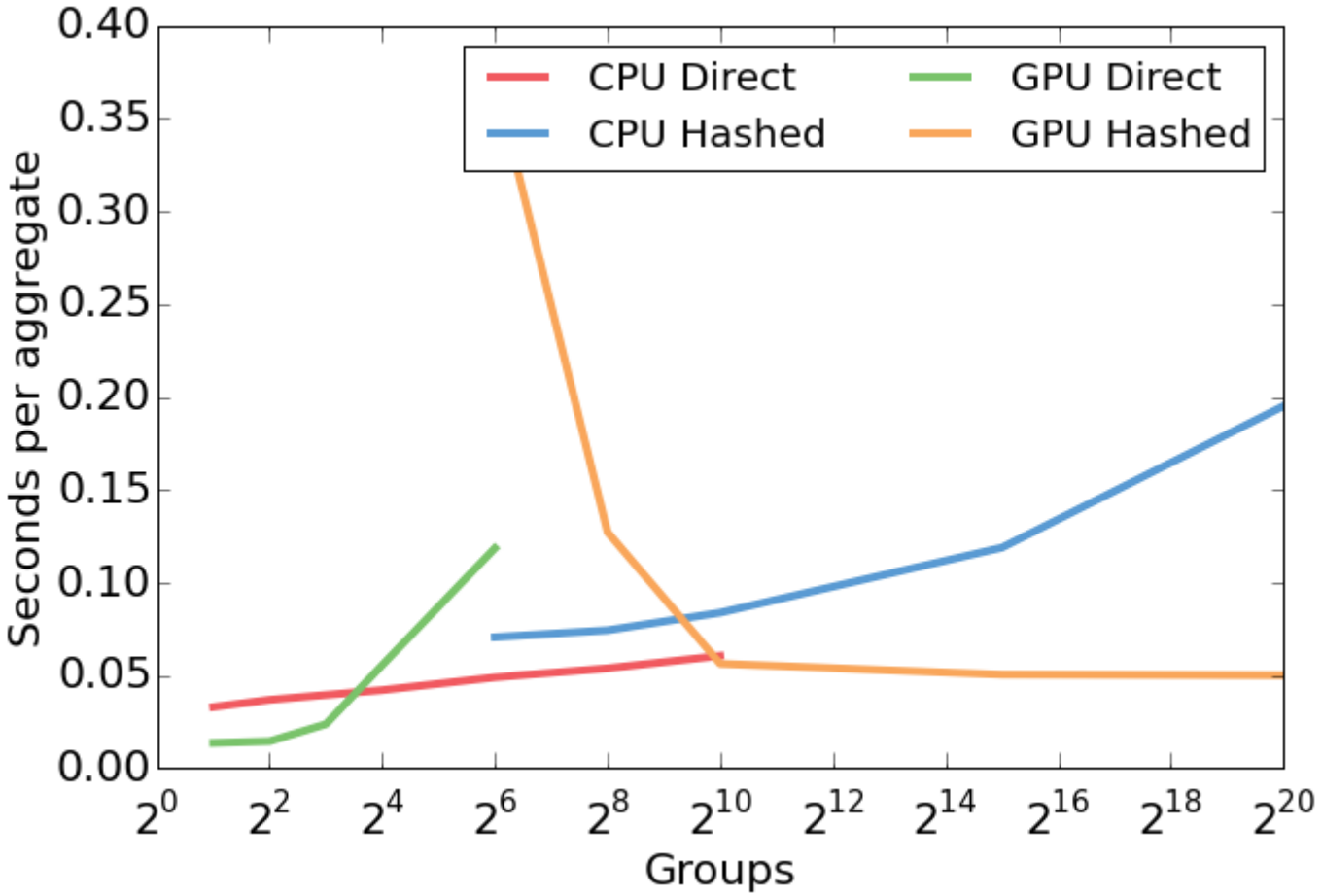
HSA vs CUDA/OpenCL

- HSA defines a heterogeneous architecture
  - Cache coherence
  - Shared virtual addresses
  - Architected queuing
  - Intermediate language
- CUDA/OpenCL are a level above HSA
  - Come with baggage
  - Not as flexible
  - May not be able to take advantage of all features

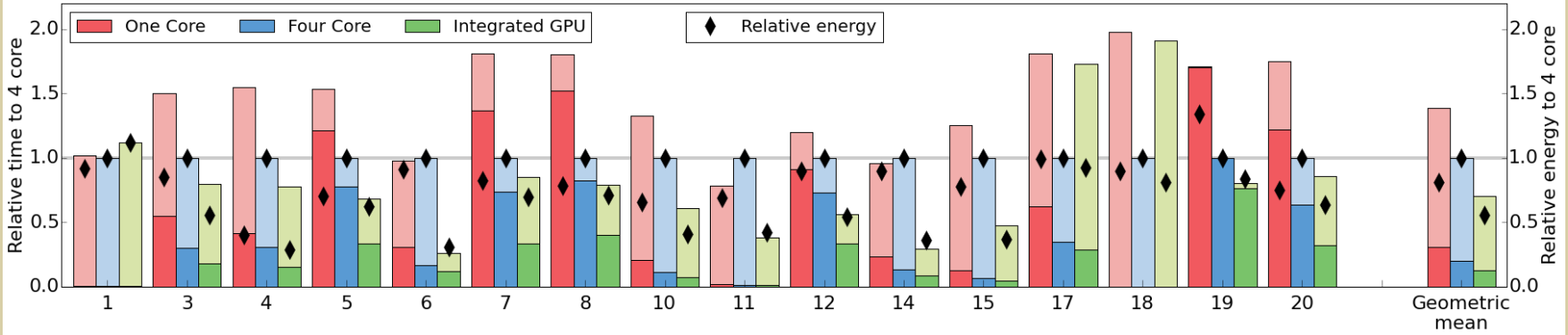
Scan Performance & Energy



Group-by Algorithms

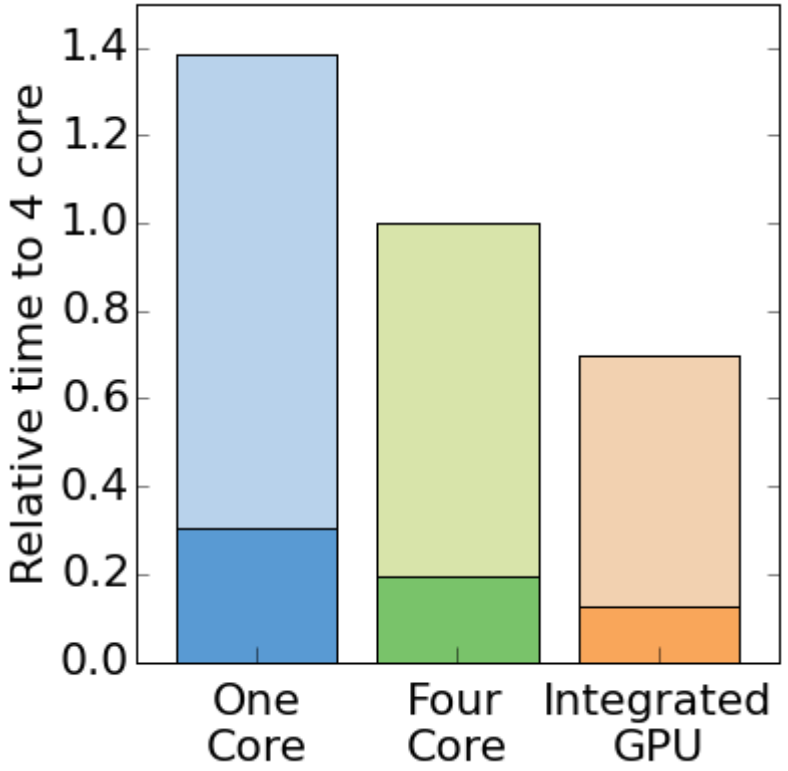


All TPC-H Results

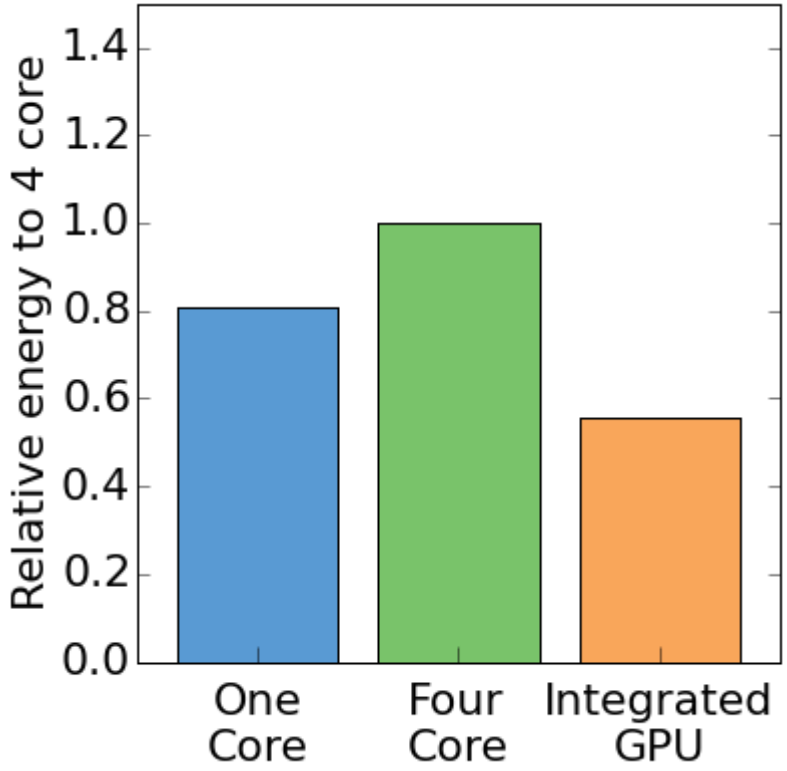


Average TPC-H Results

Average Performance



Average Energy



What's Next?

- Developing cost model for GPU
  - Using the GPU is just another algorithm to choose
  - Evaluate exactly when the GPU is more efficient
- Future "database machines"
  - GPUs are a good tradeoff between specialization and commodity

Conclusions

- Integrated GPUs viable for DBMS?
  - Solve problems with discrete GPUs
  - (Somewhat) better performance and energy

- Looking toward the future…
  - CPUs cannot keep up with bandwidth
  - **GPUs perfectly designed for these workloads**