Debunking the 100x GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU

Presenter: Victor Lee
victor.w.lee@intel.com

Throughput Computing Lab, Intel Architecture Group

GPUs is 10 – 100x faster than CPUs

Truth or Myth ???

Background

Performance of applications critically depends on two resources provided by processors – compute and bandwidth

- **Compute does the work**
- **Bandwidth feeds the compute**

Background

Well optimized applications are compute or bandwidth bounded

For compute bound applications:

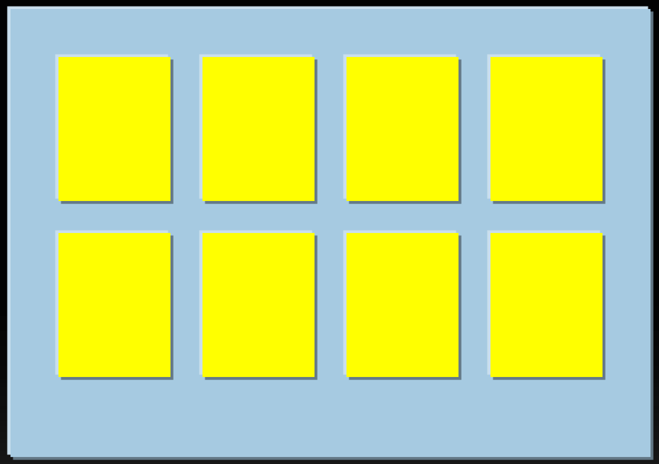
*Performance = Arch efficiency * Peak Compute Capability*

For bandwidth bound applications:

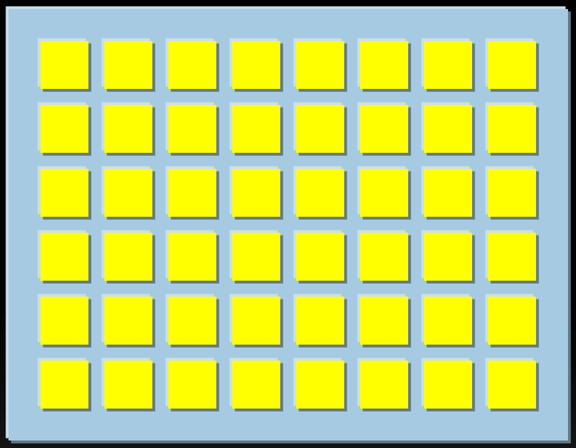
*Performance = Arch efficiency * Peak Bandwidth Capability*

Background

Chip A



Chip B



$Perf_A$ = Eff_A * $Peak_A$(Comp or BW)

$Perf_B$ = Eff_B * $Peak_B$(Comp or BW)

$$Speedup\frac{B}{A} = \frac{Perf_B}{Perf_A} = \frac{Eff_B}{Eff_A} * \frac{Peak_A(Comp/BW)}{Peak_B(Comp/BW)}$$

Background

Core i7 960

- Four OoO Superscalar Cores, 3.2GHz
- Peak SP Flop: 102GF/s
- Peak BW: 30 GB/s

GTX 280

- 30 SMs (w/ 8 In-order SP each), 1.3GHz
- Peak SP Flop: 933GF/s*
- Peak BW: 141 GB/s

Assuming both Core i7 and GTX280 have the same efficiency:

	Max Speedup: GTX 280 over Core i7 960
Compute Bound Apps: (SP)	933/102 = 9.1x
Bandwidth Bound Apps:	141/30 = 4.7x

* 933GF/s assumes mul-add and the use of SFU every cycle on GPU

GPUs is 10 – 100x faster than CPUs

Truth or Myth ???

Outline

- Throughput Workloads
- Performance Measurements
- Architecture Analysis
- Conclusion

Outline

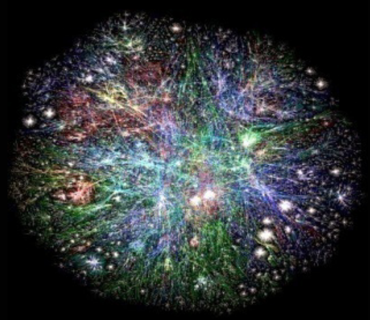
- Throughput workloads
- Performance Measurements
- Architecture Analysis
- Conclusion

Throughput workloads

- About processing a large amount of data in a given amount of time

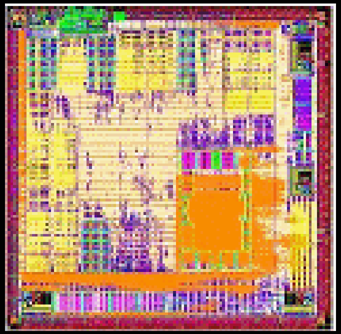
- Characteristics:
  - Workloads with plenty of data level parallelism
  - Fast response time for all data processed vs. a single data processed

Examples of Throughput Apps



Bioscience, astronomy

EDA

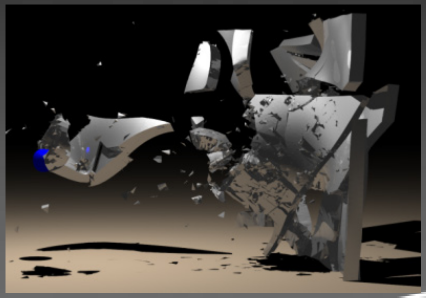


Financial Services

Virtual World

Computational Medicine

Hollywood Physics



Throughput Benchmarks

Applications	Domain
SGEMM	HPC
SAXPY	HPC
SpMV	HPC
FFT	HPC
Monte Carlo	Financial Services
Histogram	EDA
Bilateral	Image Processing
Convolution	Image Processing
Ray Casting	Medical Imaging
Constraint Solver	DCC (Physical Simulation)
GJK	DCC (Physical Simulation)
LBM	DCC (Physical Simulation)
Sort	Database
Search	Database

Throughput Benchmarks

Applications	DLP processed by	Main limiter
SGEMM	Threads / SIMD	Compute
SAXPY	Threads / SIMD	Bandwidth
SpMV	Threads / SIMD (Gather)	Bandwidth
FFT	Threads / SIMD	Compute
Monte Carlo	Threads / SIMD	Compute
Histogram	Threads / SIMD (Atomic)	Compute
Bilateral	Threads / SIMD	Compute
Convolution	Threads / SIMD	Compute
Ray Casting	Threads / SIMD (Gather)	Compute
Constraint Solver	Threads / SIMD (Gather)	Compute
GJK	Threads / SIMD (Gather)	Compute
LBM	Threads / SIMD	Bandwidth
Sort	Threads / SIMD (Gather)	Compute
Search	Threads / SIMD (Gather)	Compute

Outline

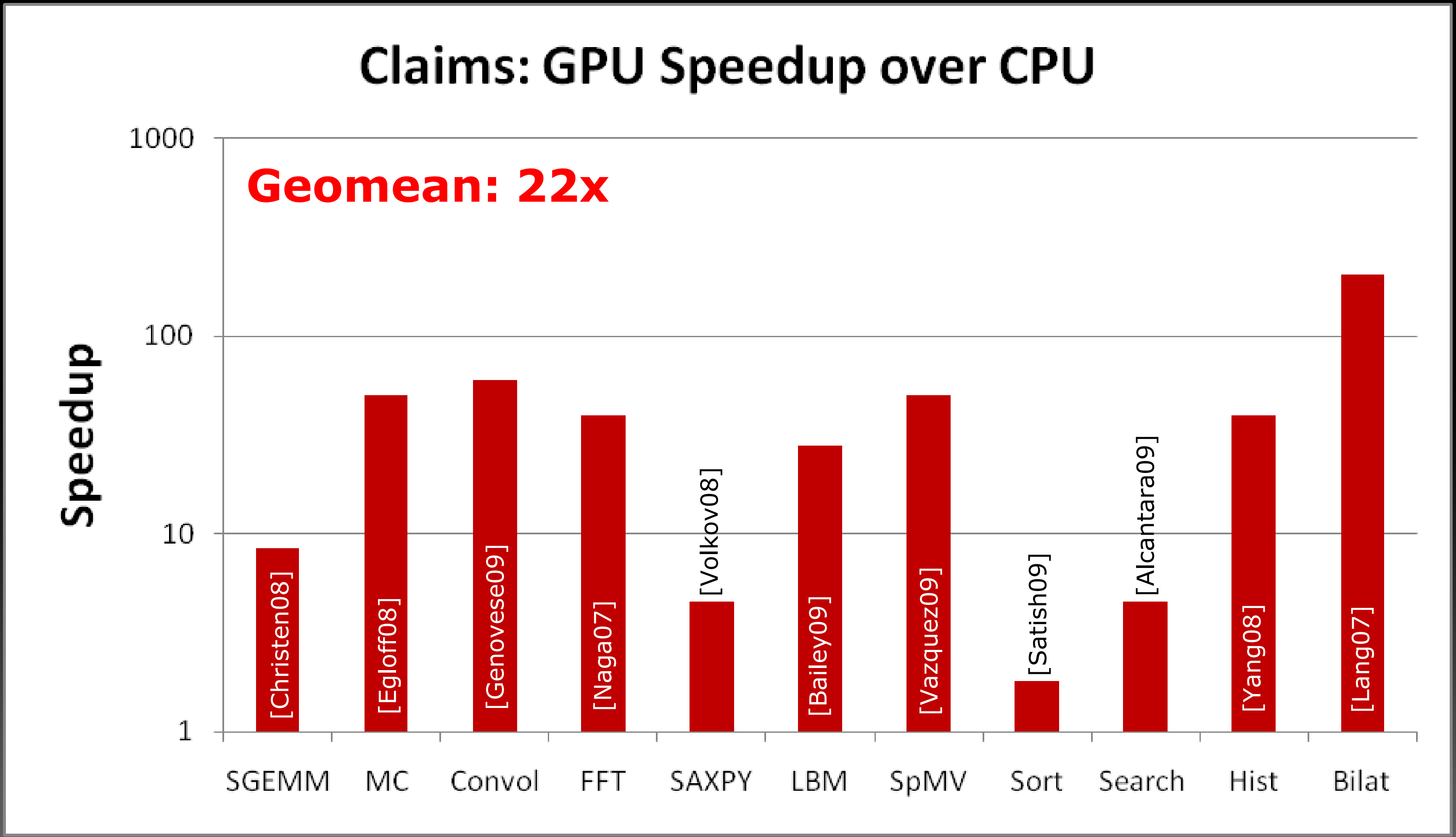
- Throughput workload characteristics
- Performance Measurements
- Architecture Analysis
- Conclusion

Methodology

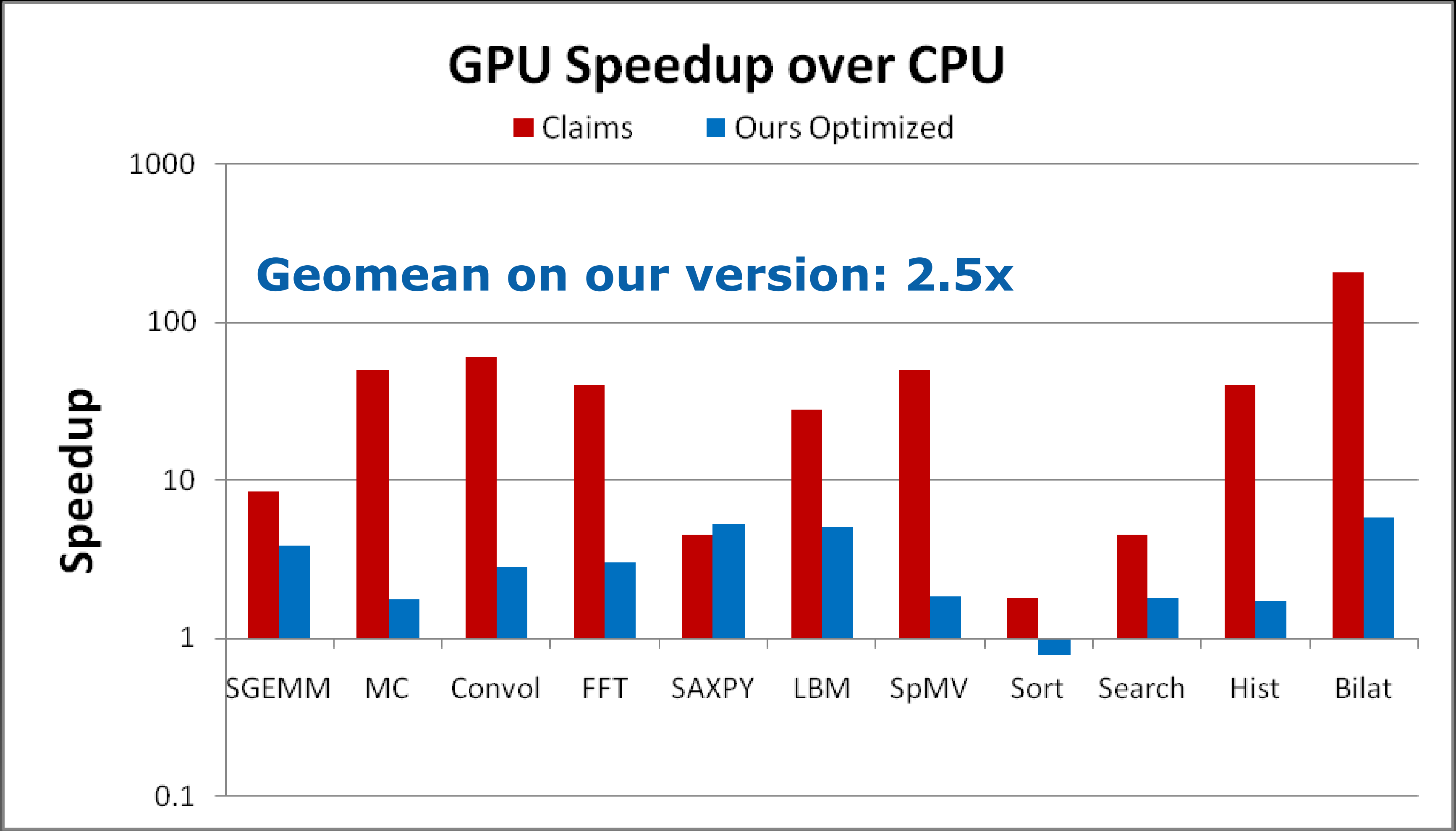
- Start with previously best published code / algorithm
- Validate claims by others
- Optimize BOTH CPU and GPU versions
- Collect and analysis performance data

Note: Only computation time on the CPU and GPU is measured. PCIe transfer time and host application time are not measured for GPU. Including such overhead will lower GPU performance

What was claimed



What we measured

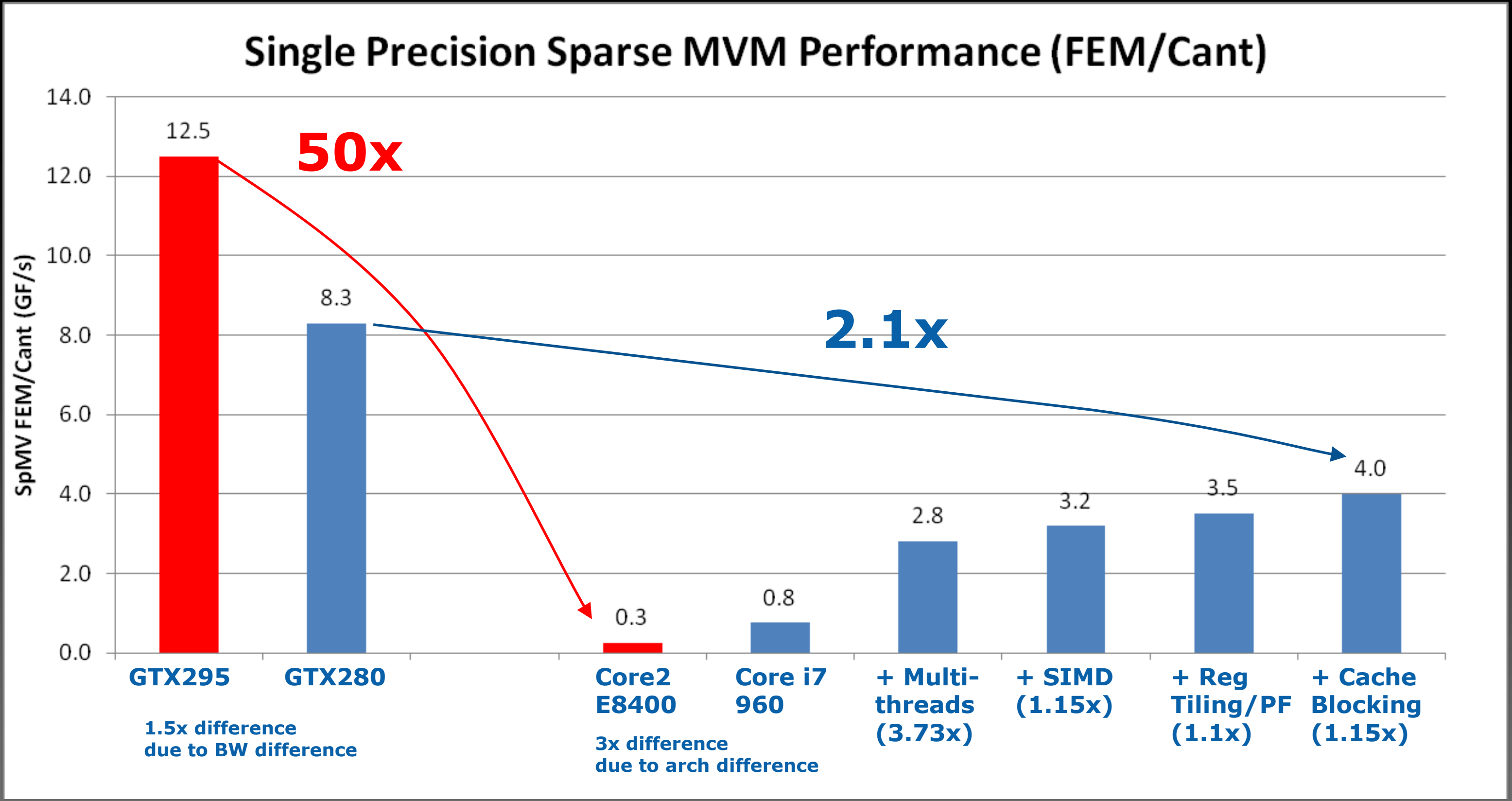


Apps.	SGEMM	MC	Conv	FFT	SAXPY	LBM	Solv	SpMV	GJK	Sort	RC	Search	Hist	Bilat
Core i7-960	94	0.8	1250	71.4	16.8	85	103	4.9	67	250	5	50	1517	83
GTX280	364	1.4	3500	213	88.8	426	52	9.1	1020	198	8.1	90	2583	475

Case Study: Sparse MVM

How a 50x claim becomes 2x

- [Vazquez09]: GTX295: ~12.5GF/s, Core 2 Duo E8400: ~ 0.25GF/s
- Our results: GTX280: 8.3GF/s, Core i7 960: 4.0GF/s



What went wrong

- CPU and GPU are not contemporary
- All attention is given to GPU coding
- CPU version is under optimized
  - E.g. Not use multi-threading
  - E.g. Not use common optimizations such as cache blocking

Outline

- Throughput workload characteristics
- Performance Measurements
- Architecture Analysis
- Conclusion

Performance Analysis

Core i7 960

- Four OoO Superscalar Cores, 3.2GHz
- Peak SP Flop: 102GF/s
- Peak BW: 30 GB/s

GTX 280

- 30 SMs (w/ 8 In-order SP each), 1.3GHz
- Peak SP Flop: 933GF/s*
- Peak BW: 141 GB/s

Assuming both Core i7 and GTX280 have the same efficiency:

	Max Speedup: GTX 280 over Core i7 960
Compute Bound Apps: (SP)	933/102 = 9.1x
Bandwidth Bound Apps:	141/30 = 4.7x

* 933GF/s assumes mul-add and the use of SFU every cycle on GPU

Performance Analysis

- **Compute-bound**
  - SGEMM, Conv, FFT: Single-Precision (2.8x – 3.0x)
  - MC: Double-Precision (1.8x)
- **Bandwidth-bound**
  - SAXPY, LBM: Main Memory (5.0x – 5.3x)

GPUs are much less compute efficient than CPUs but are slightly more bandwidth efficient

Performance Analysis

- **Compute-bound**
  - SGEMM, Conv, FFT: Single-Precision (2.8x – 3.0x)
  - MC: Double-Precision (1.8x)
- **Bandwidth-bound**
  - SAXPY, LBM: Main Memory (5.0x – 5.3x)
- **Advantage of Cache (reduce BW gap)**
  - SpMV: Bandwidth-bound (2.1x)
  - Sort, Search, RC: Compute-bound (0.79x - 1.8x)

Performance Analysis

- **Compute-bound**
  - SGEMM, Conv, FFT: Single-Precision (2.8x – 3.0x)
  - MC: Double-Precision (1.8x)
- **Bandwidth-bound**
  - SAXPY, LBM: Main Memory (5.0x – 5.3x)
- **Advantage of Cache (reduce BW gap)**
  - SpMV: Bandwidth-bound (2.1x)
  - Sort, Search, RC: Compute-bound (0.79x - 1.8x)
- **Synchronization issue on GPU (reduce compute gap)**
  - Hist: Parallel Reduction (1.7x)
  - Solv: Global Barrier (0.52x)

Performance Analysis

- **Compute-bound**
  - SGEMM, Conv, FFT: Single-Precision (2.8x – 3.0x)
  - MC: Double-Precision (1.8x)
- **Bandwidth-bound**
  - SAXPY, LBM: Main Memory (5.0x – 5.3x)
- **Advantage of Cache (reduce BW gap)**
  - SpMV: Bandwidth-bound (2.1x)
  - Sort, Search, RC: Compute-bound (0.79x - 1.8x)
- **Synchronization issue on GPU (reduce compute gap)**
  - Hist: Parallel Reduction (1.7x)
  - Solv: Global Barrier (0.52x)
- **Advantage of Fixed Function for GPU (increase compute gap)**
  - Bilat: Transcendental Operations (5.7x)
  - GJK: Texture Sampler Hardware (15x)

Outline

- Throughput workload characteristics
- Performance
- Case studies
- Architecture Analysis
- Conclusion

Conclusion

1. GPUs are NOT orders of magnitude faster than CPUs
   - In many cases, they are architecturally less efficient than CPU

2. Problems with previous work
   - Processors of comparison are not contemporary
   - Lack of architecture specific optimizations

3. Architecture features are important for throughput computing
   - Caches are good for reducing external bandwidth requirement
   - Fast synchronization and fixed function are useful for some apps

Acknowledgements

Throughput Computing Lab

Pradeep Dubey, Yen-Kuang Chen, Jatin Chhugani, Michael Deisher, Michael Espig, Christopher J Hughes, Changkyu Kim, Daehyun Kim, Anthony D Nguyen, Satish Nadathur Rajagopalan, Mikhail Smelyanskiy

Intel Architecture Group

Srinivas Chennupaty, Per Hammarlund, Ronak Singhal

Thank You!

- Visit our website: http://tcl.intel-research.net/