Challenge Benchmarks That Must Be Conquered to Sustain the GPU Revolution

Emily Blem

Matt Sinclair

Karu Sankaralingam

University of Wisconsin-Madison

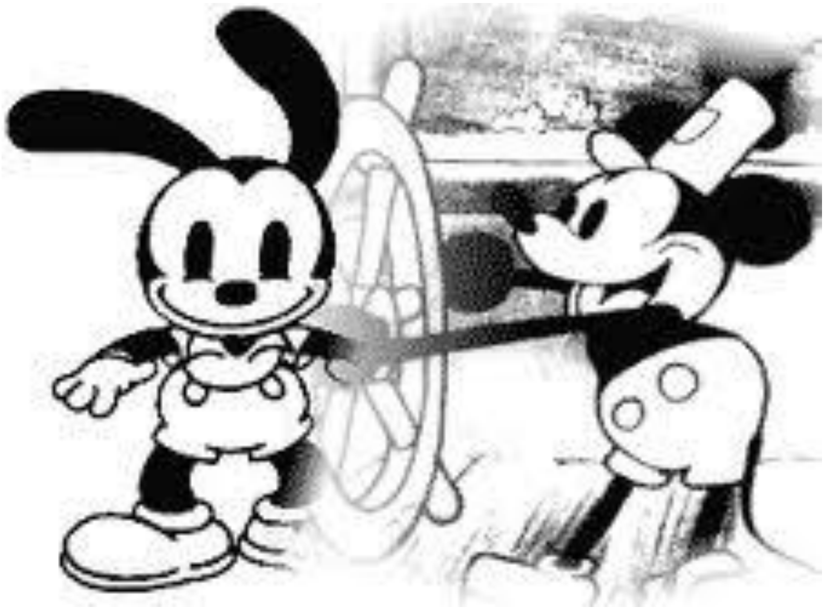
Department of Computer Sciences

Vertical Research Group

Let’s begin by thinking about a mouse.

Walt Disney Co. in the beginning …

- Walt Disney originally decided to be an animator.
- His initial successes came in the 1920’s and 1930’s.
- He was doing very well, and wasn’t forced to expand into other areas…



Walt Disney Co. as we know it.

Motivation

- GPUs are very good at data parallel programs.
- However, just like Walt Disney Co., for them to continue to grow, they need to expand.
- In this paper we find benchmarks that currently do not perform well on GPUs, but **could** perform well.

Executive Summary

- We have identified 19 challenge benchmarks.

- Our analysis suggests that there is no simple tweak to get them to perform well on GPUs.

Outline

- Introduction
- **Identifiying Challenge Benchmarks**
- Bottlenecks
- Case Studies
- Conclusions

Identifying Challenging Benchmarks

- Searched common GPU benchmark suites:
  - Rodinia
  - GPGPU-Sim
  - SHOC
  - Others
- Wrote some of our own from the PARSEC suite.

- **Goal**: Identify benchmarks from these suites that perform poorly on GPUs.

Classifying Benchmarks as Challenging

- For all benchmarks that perform at **≤ 40% of peak effective GPU IPC**.
  - We classify these benchmarks as **challenging**.
- What is effective IPC?
  - IPC calculated using only useful instructions per cycle (i.e. ignoring masked instructions).
- We use a Tesla C1060-like configuration & GPGPU-Sim version 2.1.1b.

The Challenging Benchmarks

- From GPGPU-Sim (5/14):
  - WP, NN, N-Queens, Mummer, BFS
- From Rodinia (10/20):
  - SC, SRAD1, Backprop, Heartwall, HW Tracking
  - CFD, BFS, NN, NW, Myocyte
- PARSEC:
  - Fluidanimate, Swaptions
- Others:
  - S3D (SHOC)
  - Mummer++

Outline

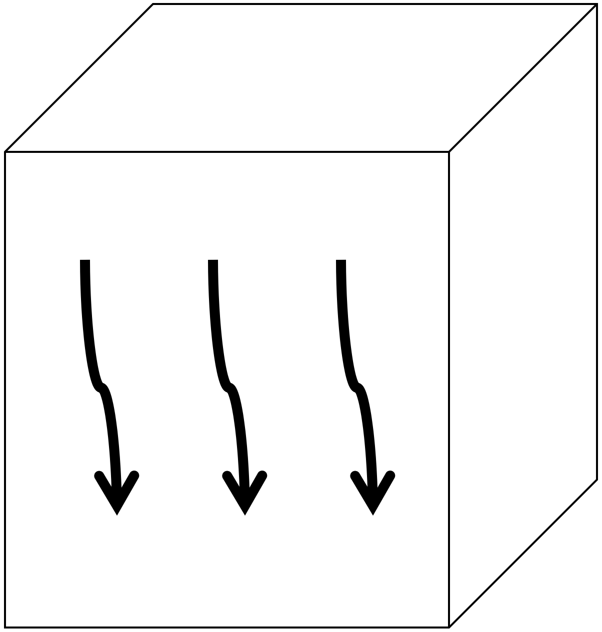
- Introduction
- Identifying Challenge Benchmarks
- **Bottlenecks**
- Case Studies
- Conclusions

GPU Bottleneck Categories

- Available Parallelism
- Control Flow
- Memory Access

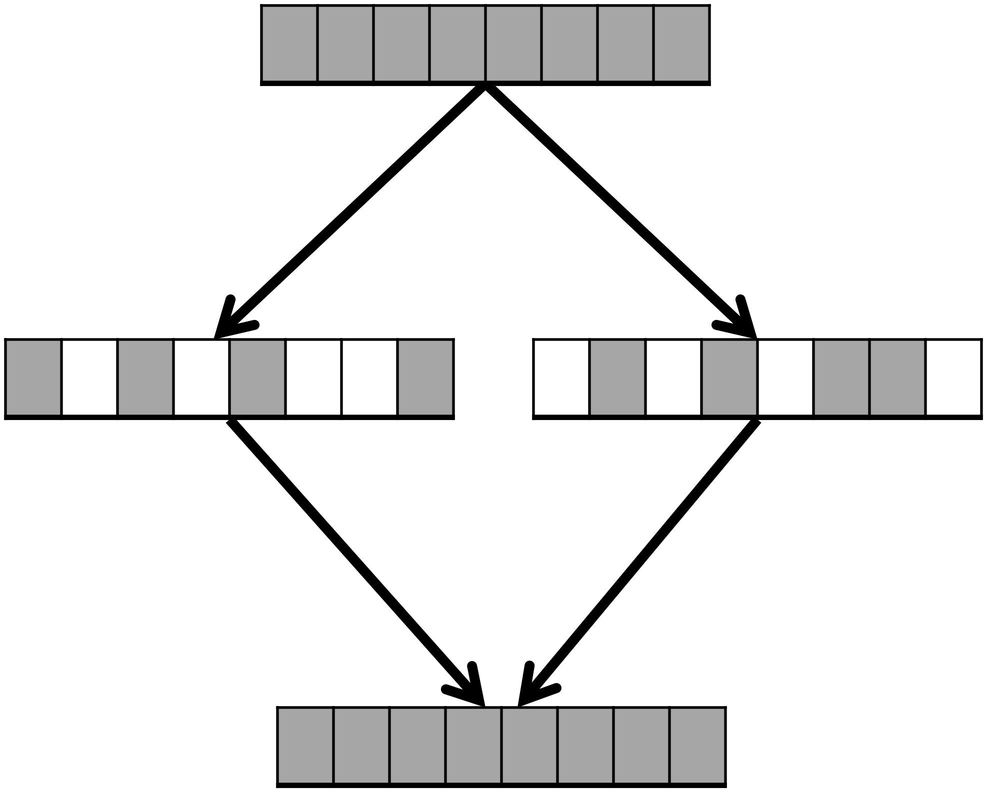
Available Parallelism

- Limited by:
  - Fraction of algorithm that is parallelizable.
- Subcategories:
  - Block Parallelism (BP)
  - Thread Parallelism (TP)
- 12/38 kernels.



Control Flow

- Limited By:
  - Thread divergence.
  - Serial execution (due to atomics, barriers, etc.).
- Subcategories:
  - Few active threads per warp (WP)
  - Single active thread per warp (ST)
- 21/38 kernels.



Memory Access

- Limited by:
  - Lack of caching
  - Heavy cache contention.
  - For lightly threaded benchmarks, GPUs can't effectively hide latency of accesses.
- Subcategories:
  - Memory Bandwidth (BW)
  - Long Latency of Memory Access (LAT)
- 19/38 kernels.

Performance Impact of Bottlenecks

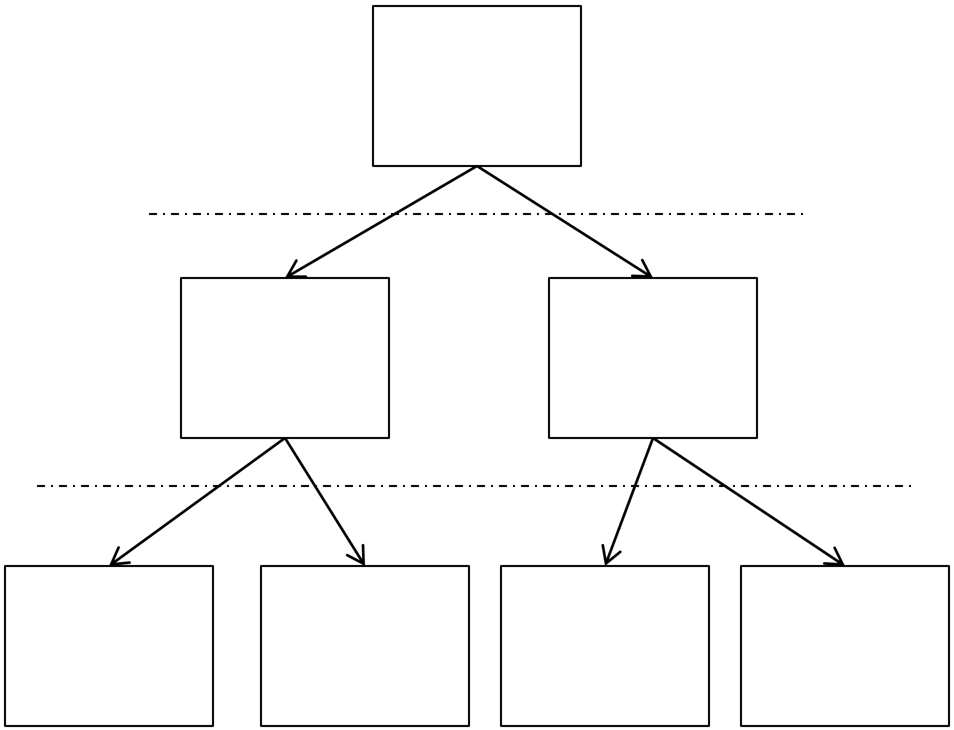
- 32/38 kernels reach peak machine efficiency after bottlenecks are removed.
  - Some require up to **5** bottlenecks be removed before reaching peak.
  - Kernels that do **not** reach peak are limited by synchronization.
- Need to remove different bottlenecks for each benchmark to reach peak efficiency.
- **Benchmarks require a 19x geometric mean speedup to reach peak machine efficiency.**

Outline

- Introduction
- Identifying Challenge Benchmarks
- Bottlenecks
- **Case Studies**
  - **BFS (Rodinia)**
  - **Fluidanimate**
- Conclusions

Case Study: BFS (Rodinia)

- 2 kernels:
  1. Marks which nodes are visited.
  2. Marks children as next; updates costs of nodes.



- 1 thread for each node in the tree, but **only** a few threads do useful work.
  - Little locality in accesses.

BFS Con't

Metric	**Kernel 1**
Effective IPC	4.9
Average Threads/Warp	10
Serialization	25%
Memory Access Coalesced	56%
DRAM Bandwidth (GB/s)	70
Stalled for Memory	76%
Bottlenecks	WP, ST, LAT

Case Study: Fluidanimate

- The fluidanimate GPU implementation requires many calls to global memory to access values.
- Also exhibits thread divergence and register pressure.
- CPU synchronization between each stage in the computation due to lack of efficient global GPU synchronization mechanism.

Fluidanimate Con’t

Metric	**Kernel 4**
Effective IPC	0.1
Average Threads/Warp	3
Serialization	51%
Memory Access Coalesced	3%
DRAM Bandwidth (GB/s)	13
Stalled for Memory	40%
(All) Bottlenecks	WP, BP, LAT, ST

Modeled speedups after removing bottlenecks

- We explored different design improvements to improve GPGPU performance.
  - Just adding additional cores or isolating a single bottleneck is not sufficient.

Thus, we look at pairs of design changes.

- Results: (N/35 kernels)
  - Group X: Near peak IPC after any design pair introduced (12).
  - Group Y: Need specific design pair to get near peak IPC (10).
  - Group Z: Don’t reach peak IPC even after multiple pairs (13).
  - **No single technique to help all benchmarks.**

Outline

- Introduction
- Identifying Challenge Benchmarks
- Bottlenecks
- Case Studies
- **Conclusions**

Conclusions

- We've introduced a set of challenging benchmarks
  - These benchmarks represent the issues future GPUs need to overcome to allow GPUs to become more general-purpose.

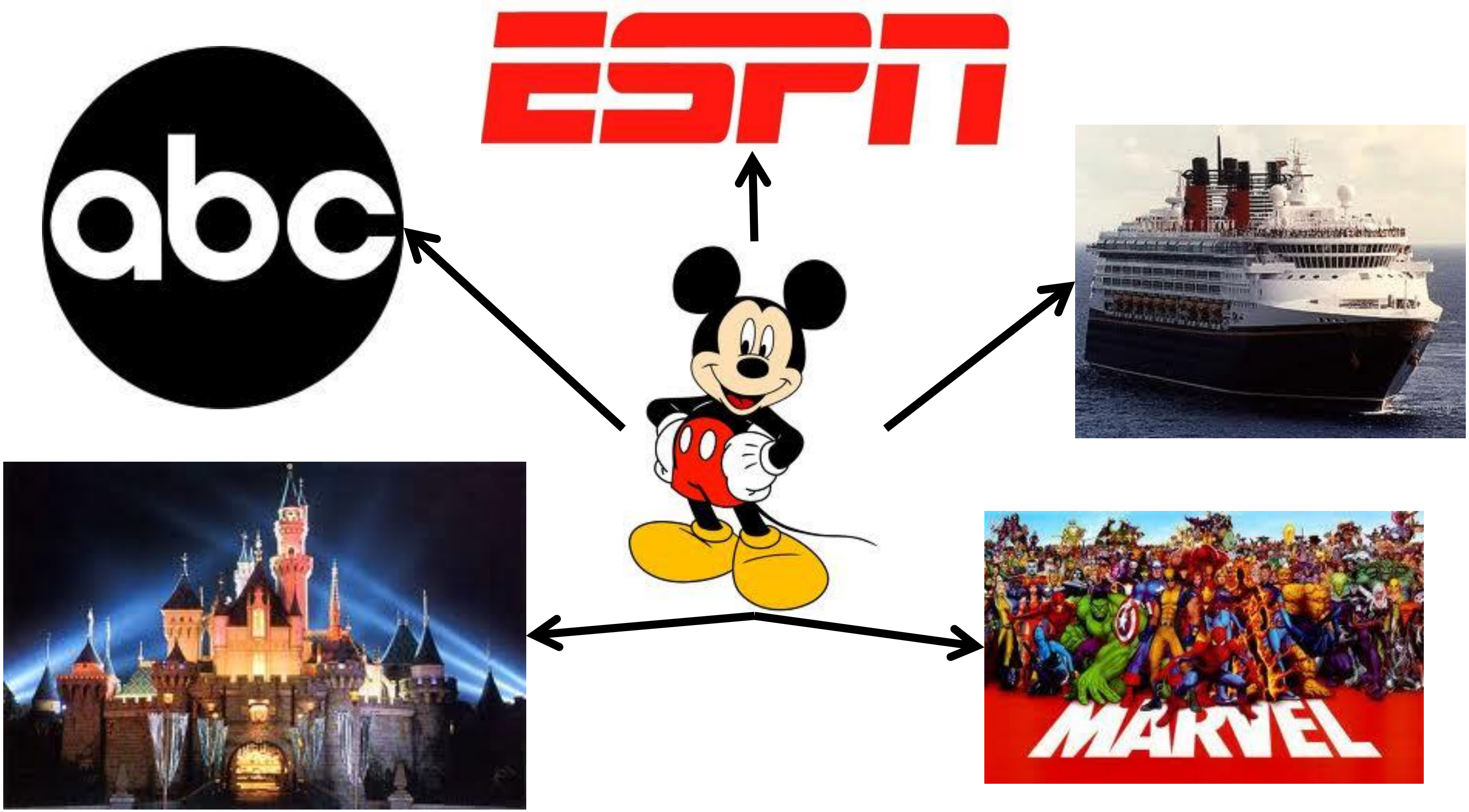
- We've also explored the bottlenecks for these benchmarks and highlighted how alleviating them will affect performance.
  - Many changes need to be made to the GPU architecture
  - **This is a hard problem, 1 or 2 techniques are not sufficient.**

Questions?

Paper available at cs.wisc.edu/vertical/

Backup Slides

By solving these challenges, GPUs can continue to expand.



Case Study: Neural Network

- The neural network executes by calling a series of layers, which update the weights of the nuerons.
- Varying number of threads per layer to account for varying number of neurons.
  - Never more than 3000 threads per layer.
- All nuerons **access global memory** when updating their values and passing them to the next layer.

Neural Network Con't

Metric	**Kernel (Layer) 2**
Effective IPC	12
Average Threads/Warp	25
Serialization	0%
Memory Access Coalesced	90%
DRAM Bandwidth (GB/s)	64
Stalled for Memory	65%
(All) Bottlenecks	BW

Case Study: Mummer++

- Kernel is attempting to align genomes
- Very limited number of threads (256)
- Lots of divergence within the kernel because we're using lots of conditionals in the pairing process.
- Most of references are to global memory.

Mummer++ Con't

Metric	**Kernel 4**
Effective IPC	0.3
Average Threads/Warp	8
Serialization	37%
Memory Access Coalesced	77%
DRAM Bandwidth (GB/s)	52
Stalled for Memory	58%
(All) Bottlenecks	WP, BP, ST

BFS Alternate Data

Metric	**Kernel 1**	**Kernel 2**
Effective IPC	4.9	104.3
Average Threads/Warp	10	27
Serialization	25%	4%
Memory Access Coalesced	56%	97%
DRAM Bandwidth (GB/s)	70	34
Stalled for Memory	76%	33%
Bottlenecks	WP, ST, LAT	LAT

The changes with Fermi

- Fermi additions:
  - Local L1 and shared L2 caching.
  - More SPs per SM (doubles effective peak IPC)
  - **This is a step in the right direction.**
- We performed the same hardware profiling study on a Tesla C2050.
- **Result:** Challenge benchmarks were only sped up **1.5x**.
  - Limited parallelism and significant thread divergence are still problems.