Understanding GPUs Through Benchmarking

Mike Houston

Stanford University

Introduction

- **Key areas for GPGPU**
  - Memory latency behavior
  - Memory bandwidths
  - Upload/Download
  - Instruction rates
  - Branching performance
- **Chips analyzed**
  - ATI X1900XTX (R580)
  - NVIDIA 7900GTX (G71)
  - NVIDIA 8800GTX (G80)

GPUBench

- **An open-source suite of micro-benchmarks**
  - GL (we'll be using this for the talk)
  - DX9 (alpha version)
- **Developed at Stanford to aid our understanding of GPUs**
  - Vendors wouldn't directly tell us arch details
  - Behavior under GPGPU apps different than games and other benchmarks
- **Library of results**

http://graphics.stanford.edu/projects/gpubench/

Memory latency

- **Questions**
  - Can latency be hidden?
  - Does access pattern affect latency?

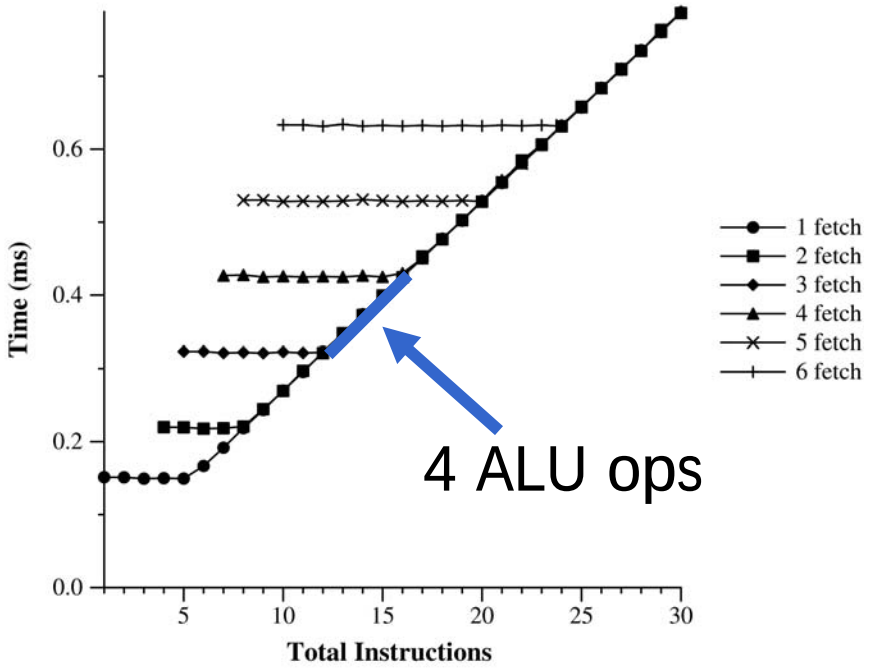
Methodology

- **Try different numbers of texture fetches**
  - Different access patterns:
    - Cache hit – every fetch to the same texel
    - Sequential – every fetch increments address by 1
    - Random – dependent lookup with random texture
- **Increase the ALU ops of the shader**
- **ALU ops *must* be dependent to avoid optimization**

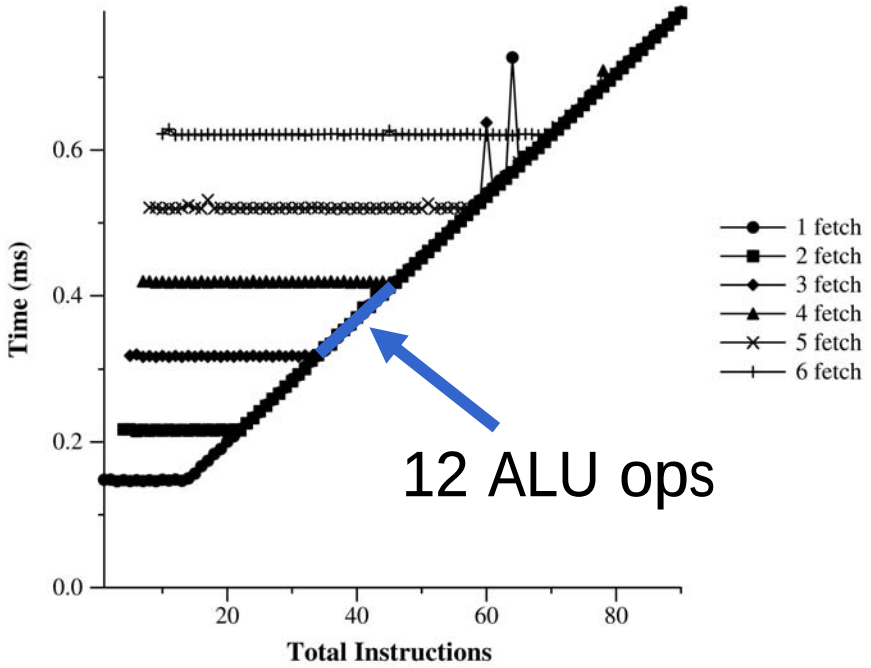
- **GPUBench test: fetchcost**

Fetch cost – ATI – cache hit

X1900XTX has 3X the ALUs per pipe



ATI X1800XT

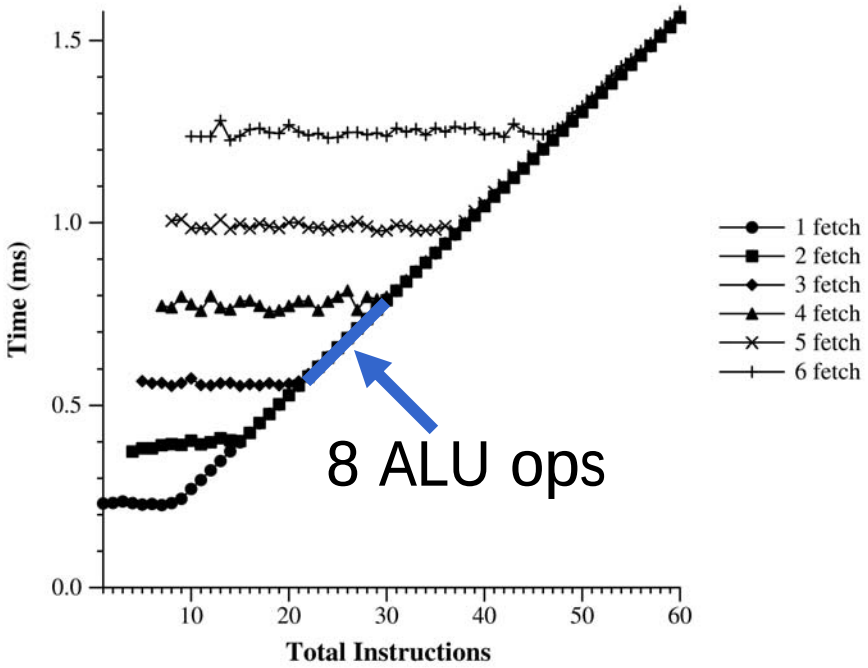


ATI X1900XTX

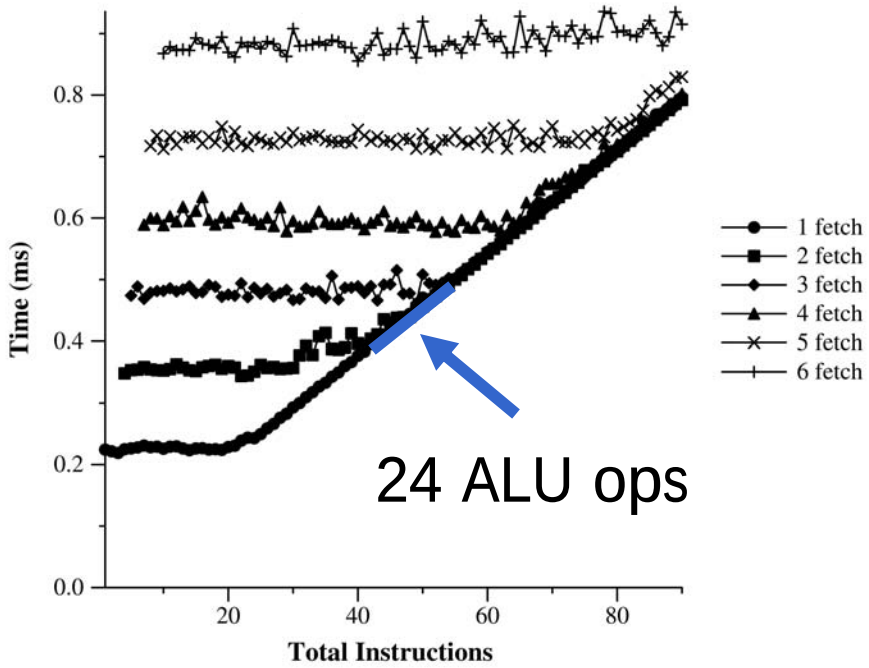
Cost = max(ALU, TEX)

Fetch cost – ATI – sequential

X1900XTX has 3X the ALUs per pipe



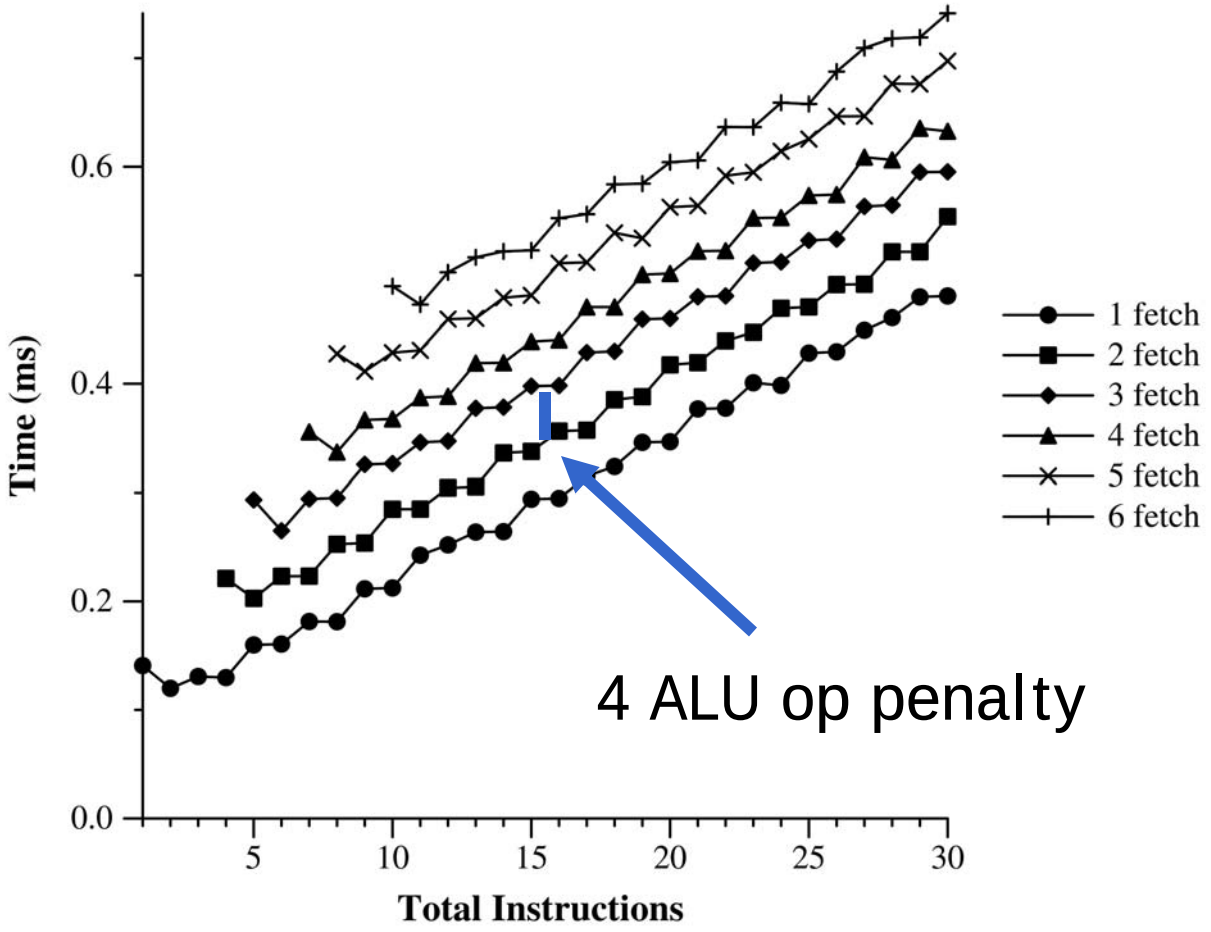
ATI X1800XT



ATI X1900XTX

Cost = max(ALU, TEX)

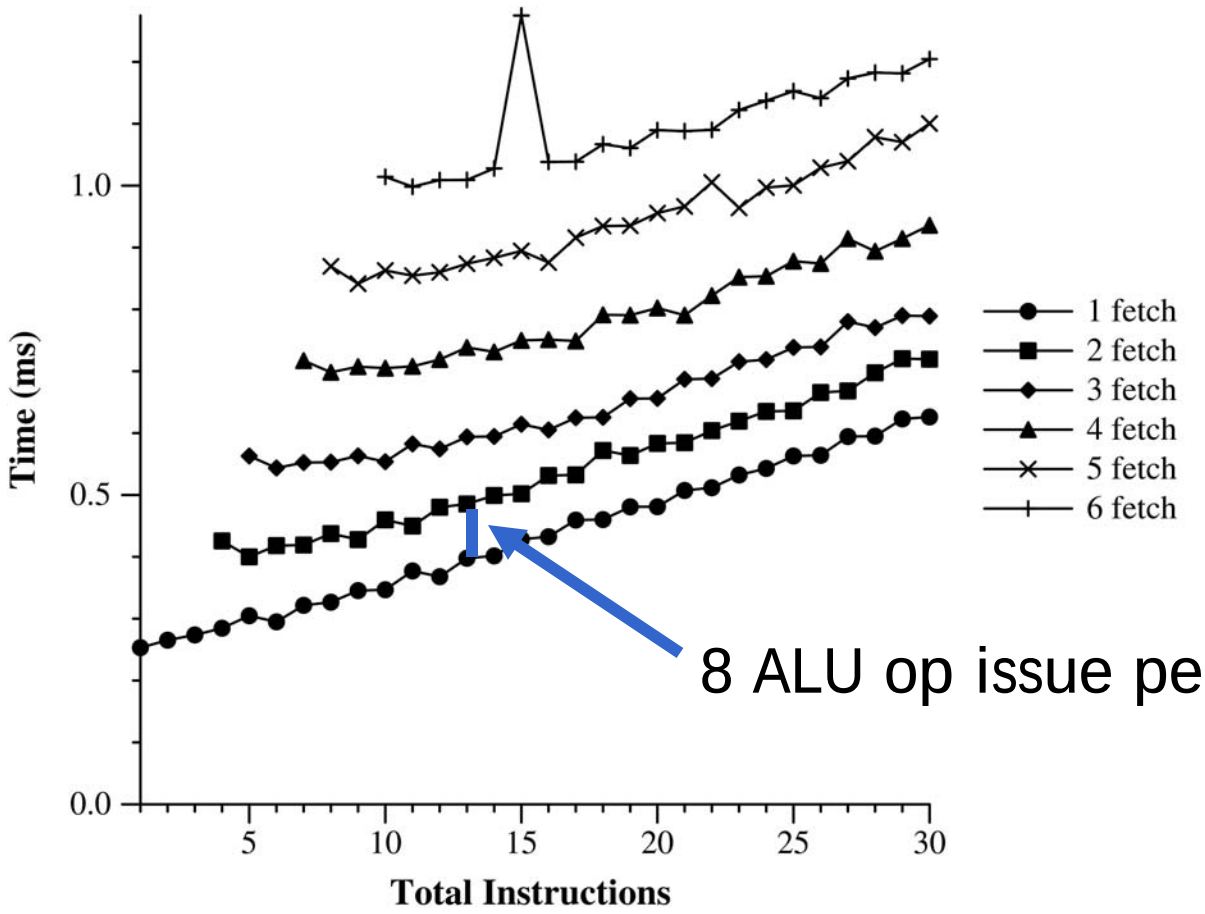
Fetch cost – NVIDIA – cache hit



Cost = sum(ALU, TEX)

NVIDIA 7900 GTX

Fetch cost – NVIDIA – sequential

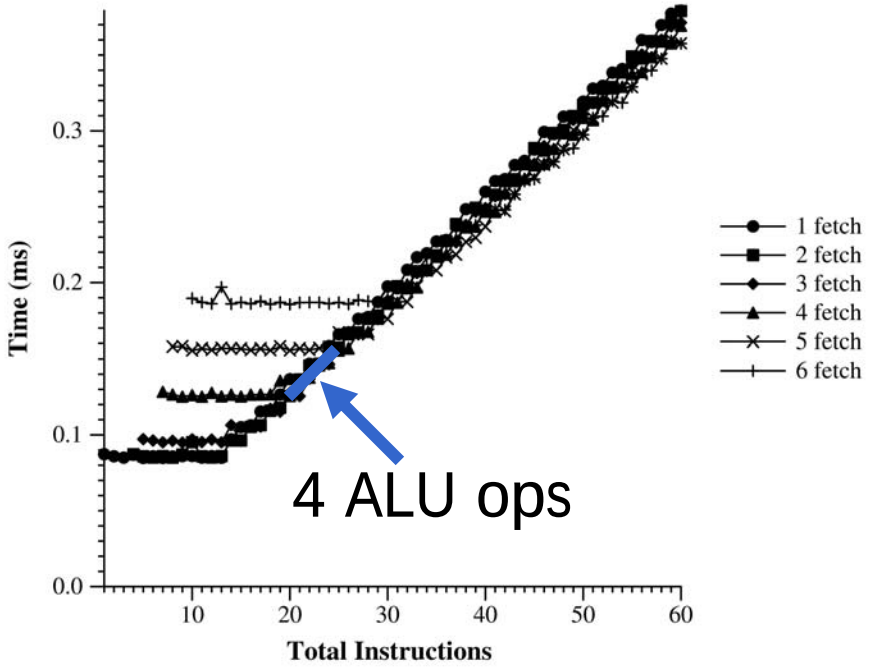


Cost = sum(ALU, TEX)

NVIDIA 7900 GTX

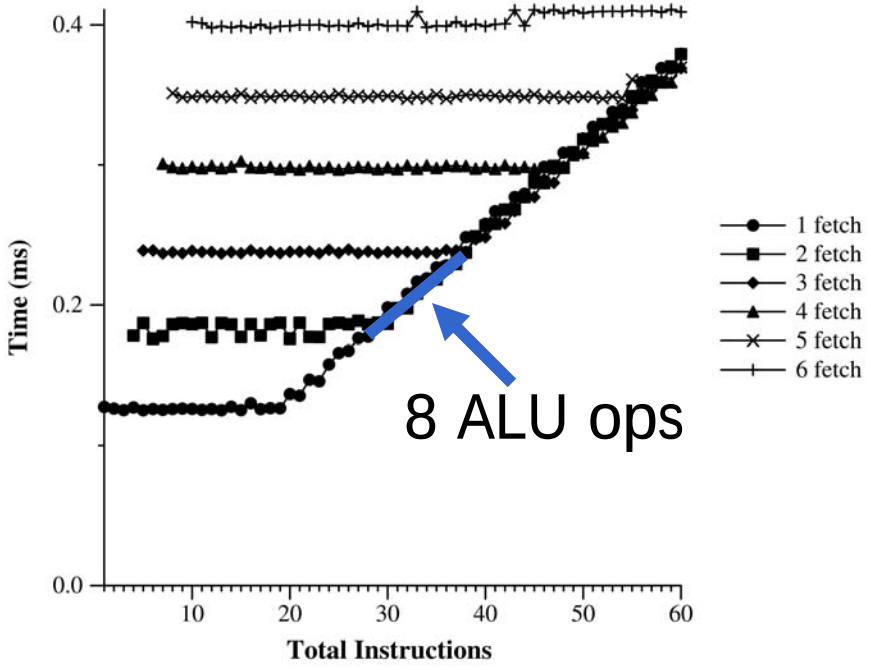
Fetch cost – NVIDIA 8800 GTX

Cache



NVIDIA 8800 GTX

sequential



NVIDIA 8800 GTX

Cost = max(ALU, TEX)

Bandwidth to ALUs

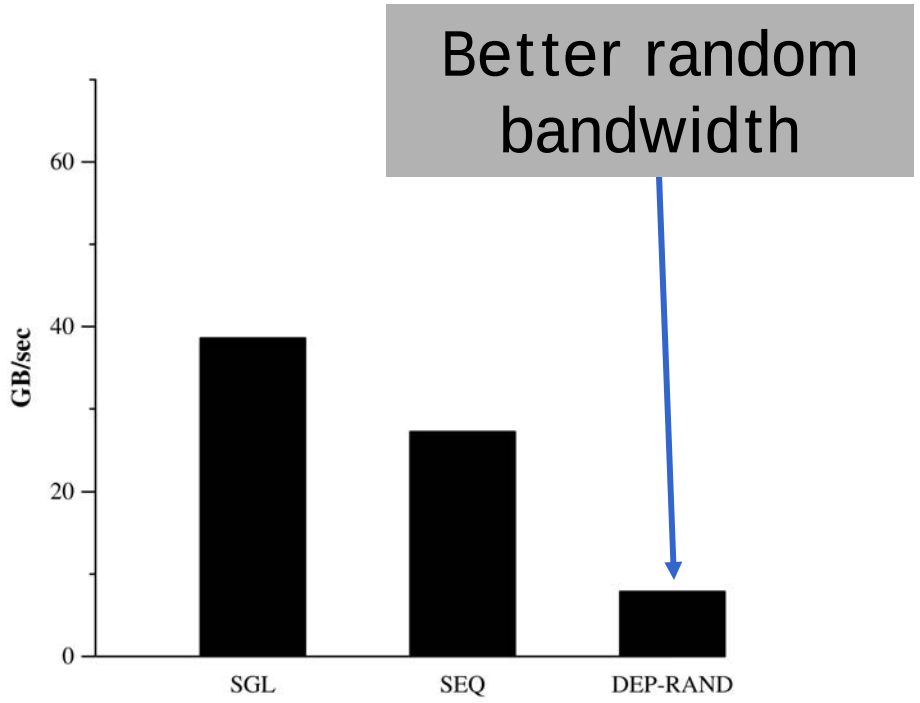
- **Questions**
  - Cache performance?
  - Sequential performance?
  - Random-read performance?

Methodology

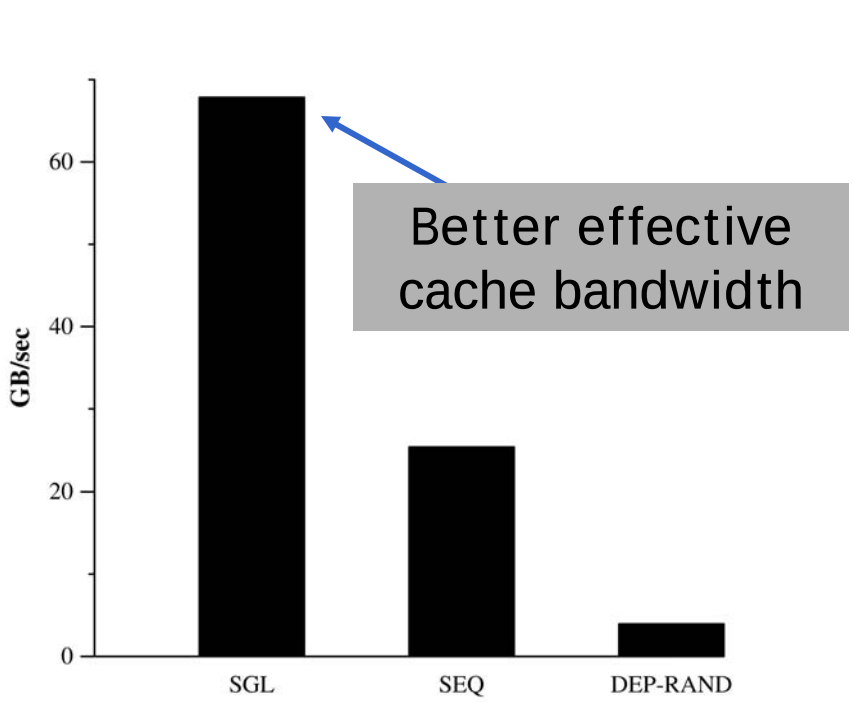
- **Cache hit**
  - Use a constant as index to texture(s)
- **Sequential**
  - Use fragment position to index texture(s)
- **Random**
  - Index a seeded texture with fragment position to look up into input texture(s)

- **GPUBench test: inputfloatbandwidth**

Results



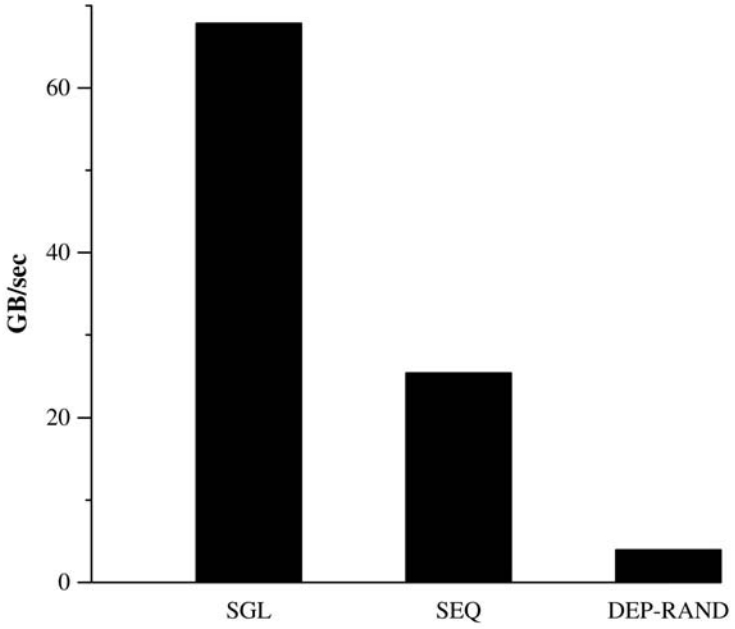
ATI X1900XTX



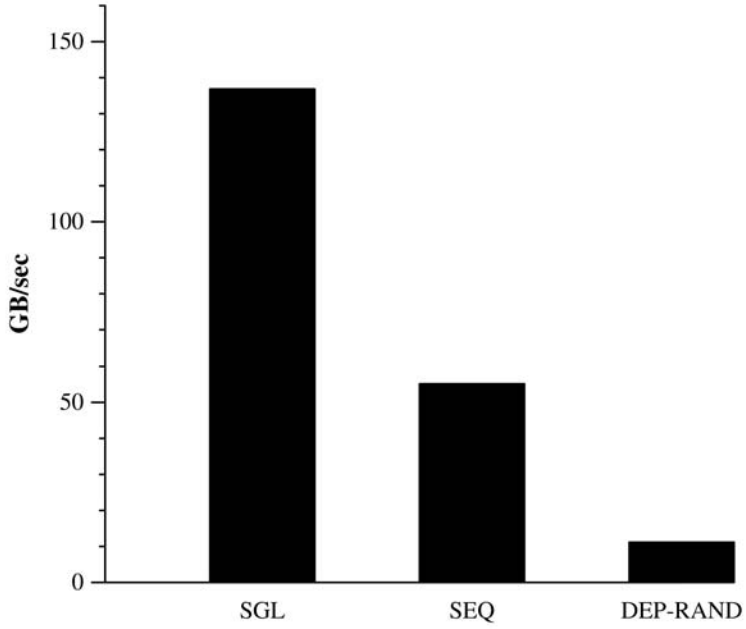
NVIDIA 7900GTX

Sequential bandwidth (SEQ) about the same

Results



NVIDIA 7900GTX



NVIDIA 8800GTX

2X bandwidth of 7900GTX

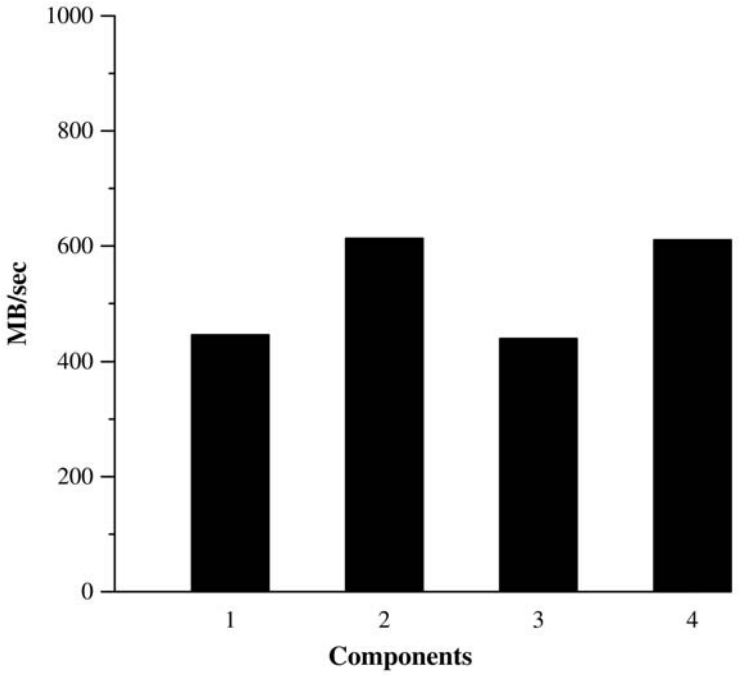
Off-board bandwidth

- **Questions**
  - How fast can we get data on the board (download)?
  - How fast can we get data off the board (readback)?

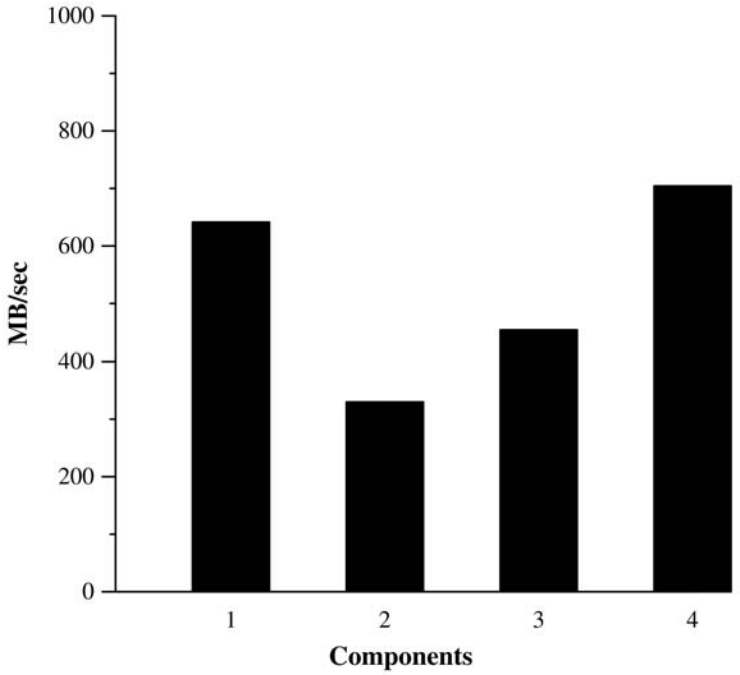
- **GPUBench tests:**
  - download
  - readback

Download

Host to GPU is slow



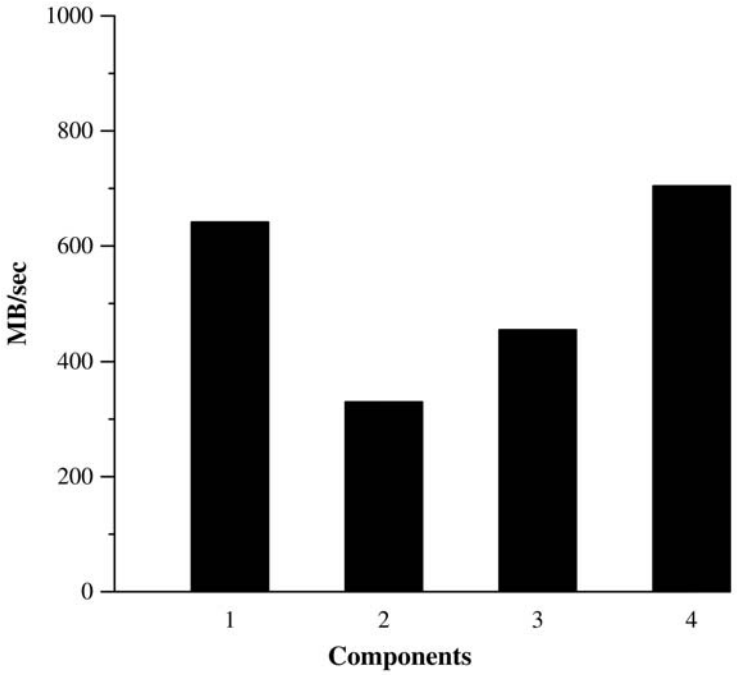
ATI X1900XTX



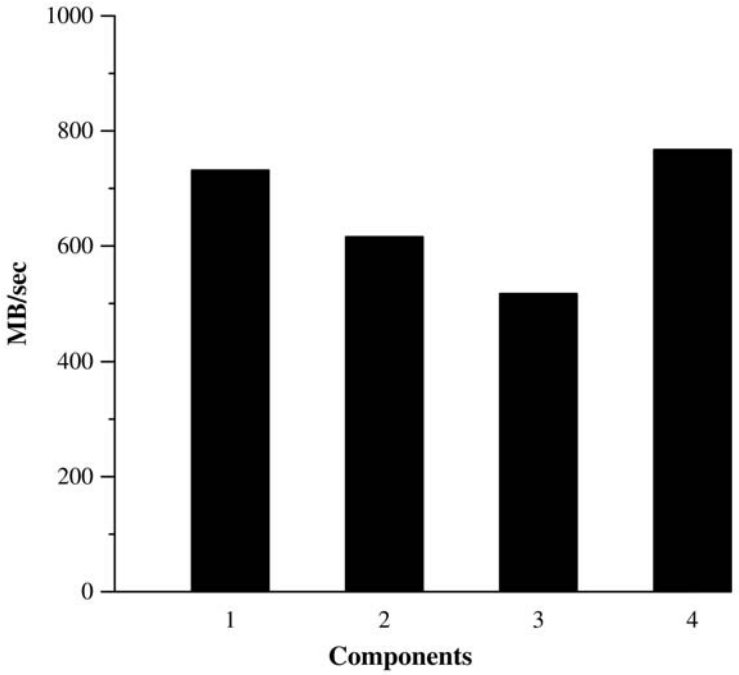
NVIDIA 7900GTX

Download

Next generation not much better…



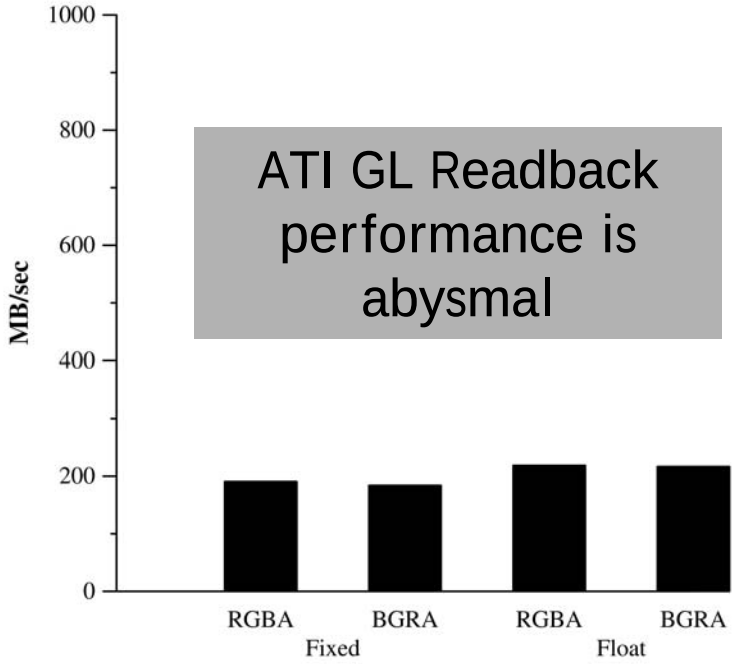
NVIDIA 7900GTX



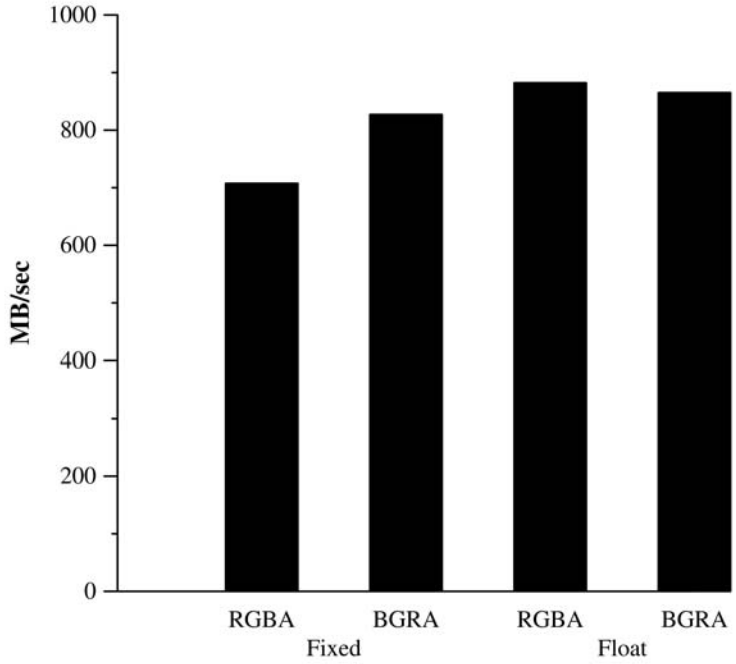
NVIDIA 8800GTX

Readback

GPU to host is slow



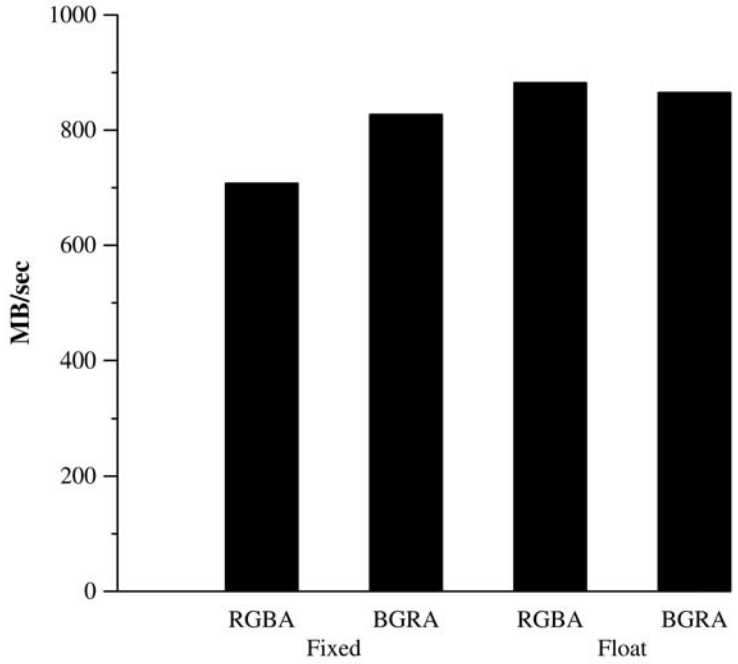
ATI X1900XTX



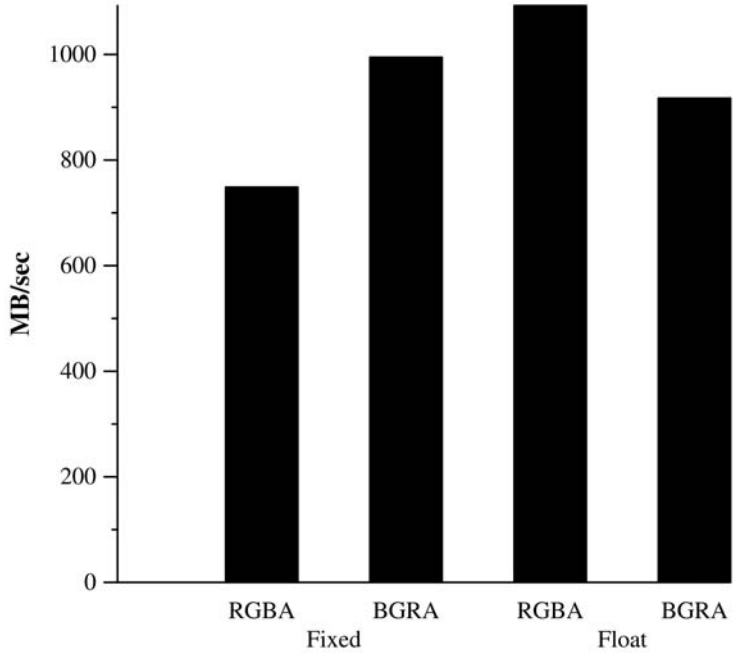
NVIDIA 7900GTX

Readback

Next generation not much better…



NVIDIA 7900GTX



NVIDIA 8800GTX

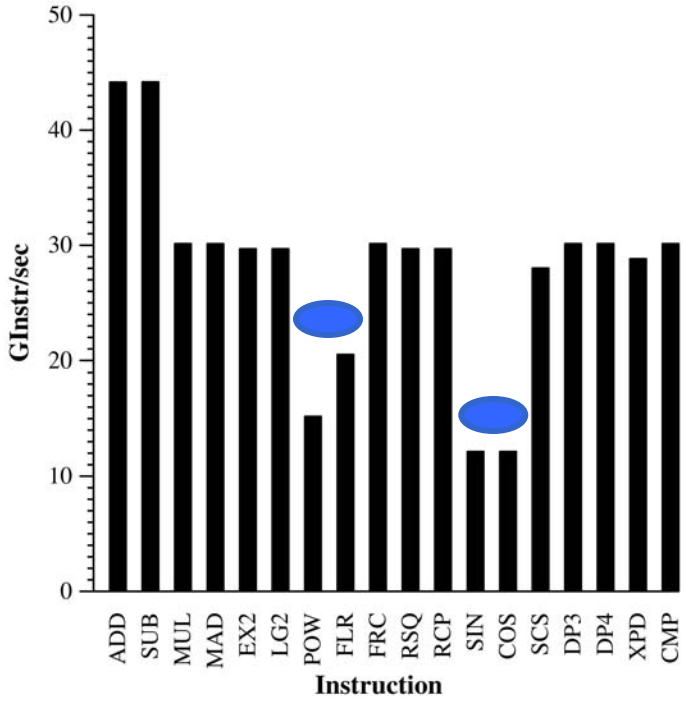
Instruction Issue Rate

- **Questions**
  - What is the raw performance achievable?
  - Do different instructions have different costs?
  - Vector vs. scalar issue differences?

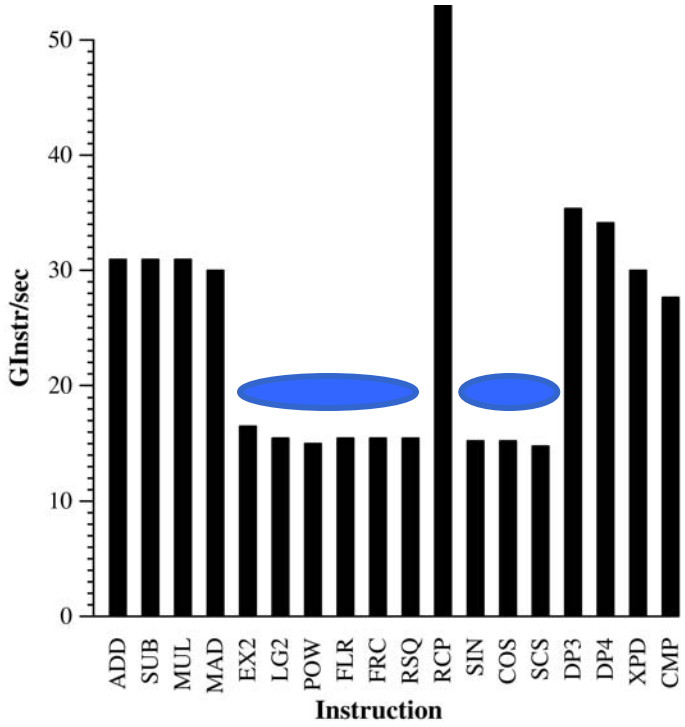
Methodology

- **Write *long* shaders with dependent instructions**
  - >100 instructions
  - All instructions dependent
    - But try to structure to allow for multi-issue
- **Test float1 vs. float4 performance**
- **GPUBench tests:**
  - instrissue

Results - Vector issue



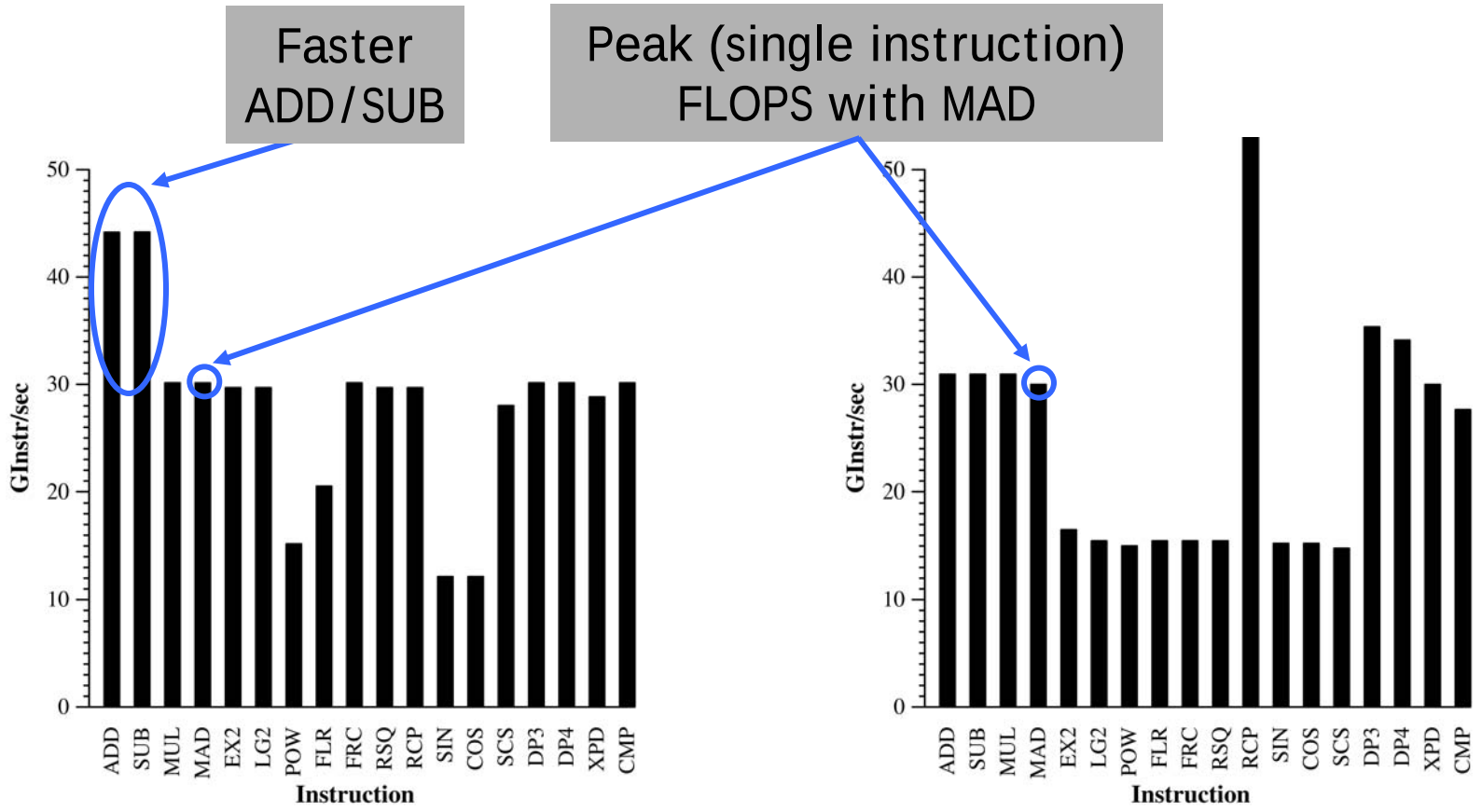
ATI X1900XTX



NVIDIA 7900GTX

= More costly than others

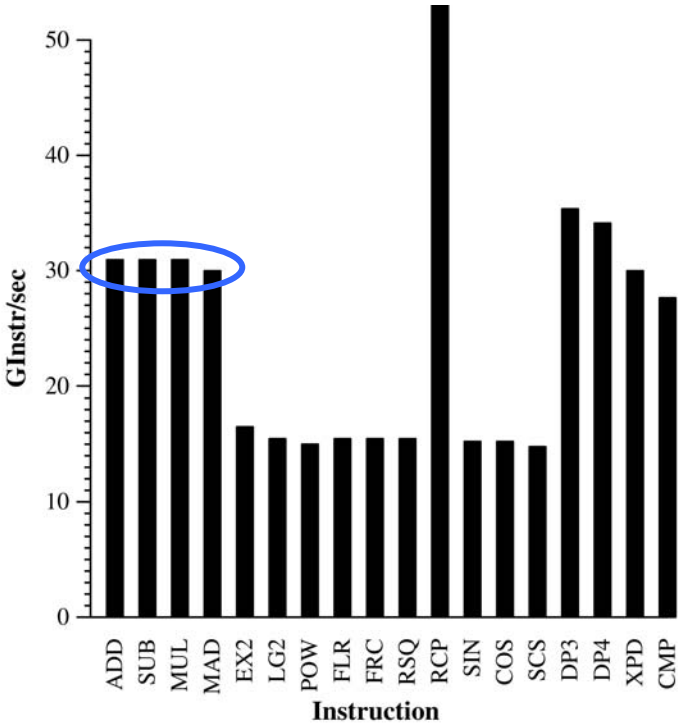
Results - Vector issue



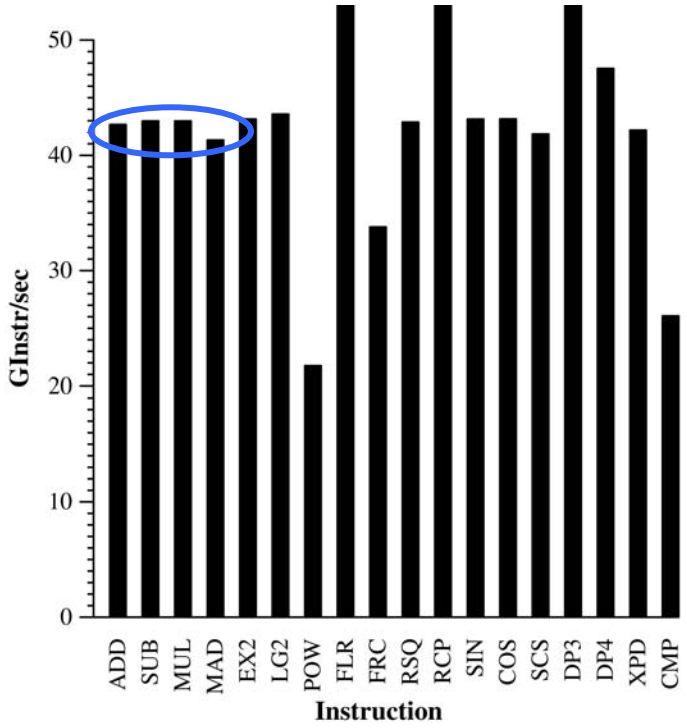
ATI X1900XTX

NVIDIA 7900GTX

Results - Vector issue



NVIDIA 7900GTX

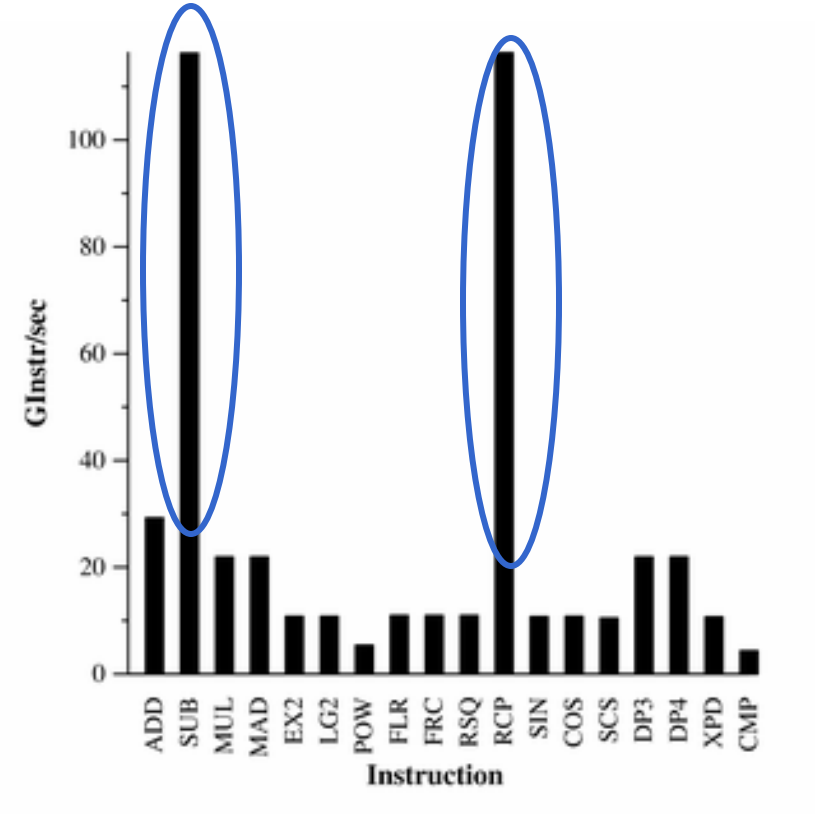


NVIDIA 8800GTX

8800GTX is 37% faster (peak)

When benchmarks go wrong…

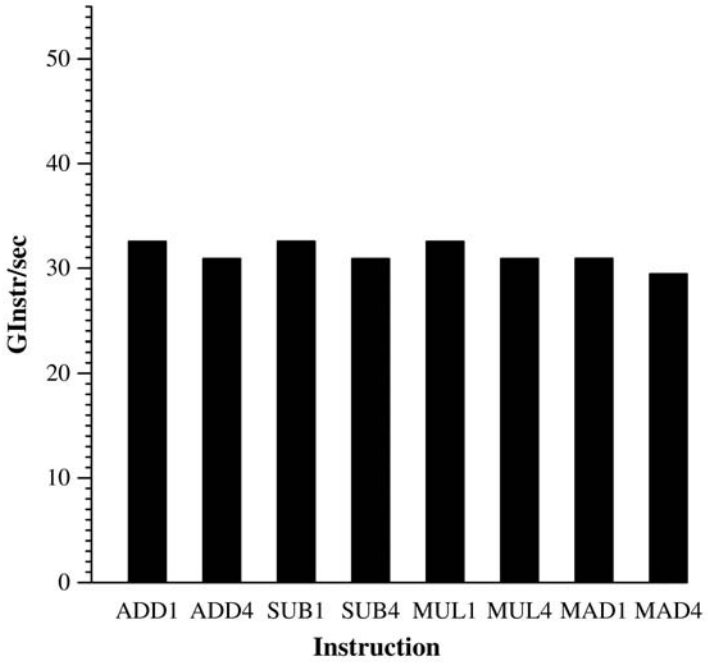
- **Smart compilers subverting testing and optimizing away shaders. Bug found in previous subtract test. No clever way to write RCP test found yet…**
  Always sanity check results against theoretical peak!!!



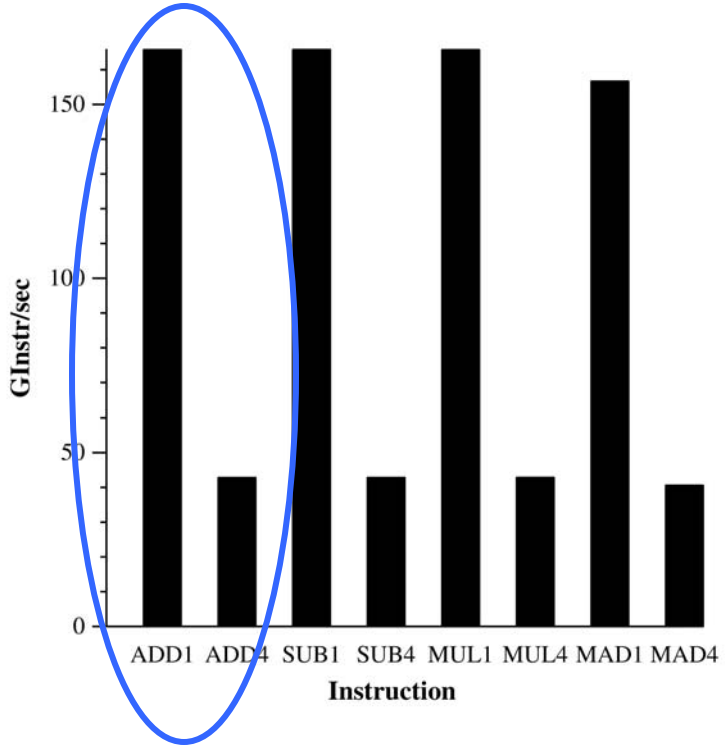
NVIDIA 7800GTX

GPUBench 1.2

Results – Scalar issue



NVIDIA 7900GTX



NVIDIA 8800GTX

8800GTX is a scalar issue processor

Branching Performance

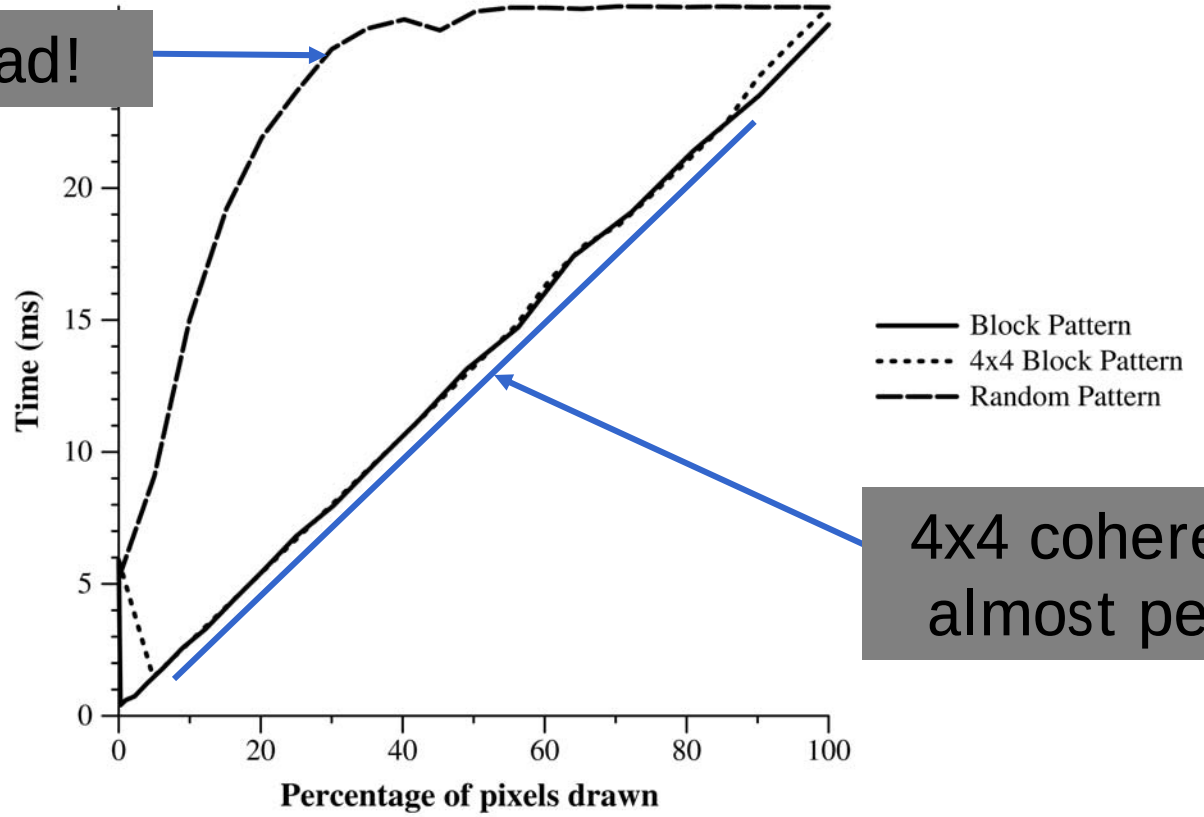
- **Questions**
  - Is predication better than branching?
  - Is using “Early-Z” culling a better option?
  - What is the cost of branching?
  - What branching granularity is required?
  - How much can I really save branching around heavy computation?

Methodology

- **Early-Z**
  - Set a Z-buffer and compare function to mask out compute
  - Change coherence of blocks
  - Change sizes of blocks
  - Set differing amounts of pixels to be drawn
- **Shader Branching**
  - If{ do a little }; else { LOTS of math}
  - Change coherence of blocks
  - Change sizes of blocks
  - Have differing amounts of pixels execute heavy math branch
- **GPUBench tests:**
  - branching

Results – Early-Z - NVIDIA

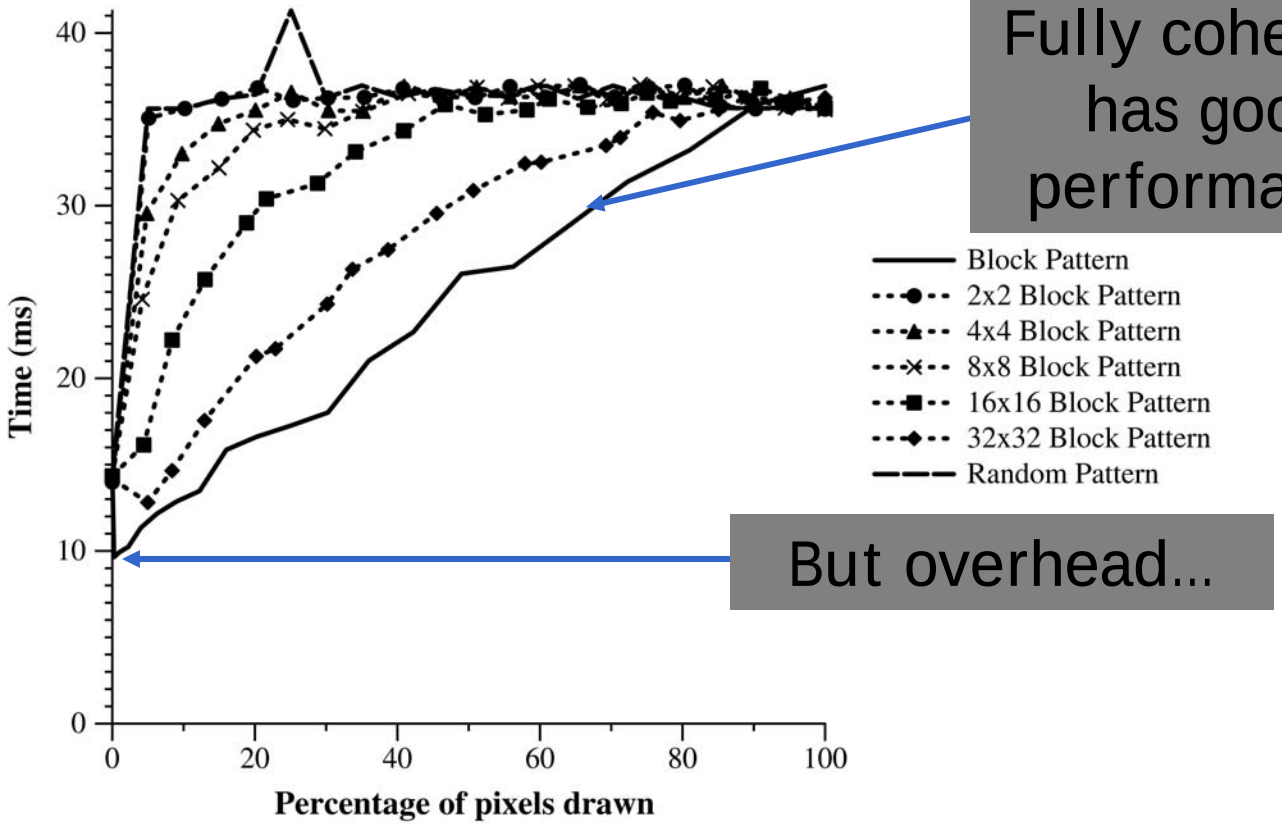
Random is bad!



4x4 coherence is almost perfect!

NVIDIA 7900GTX

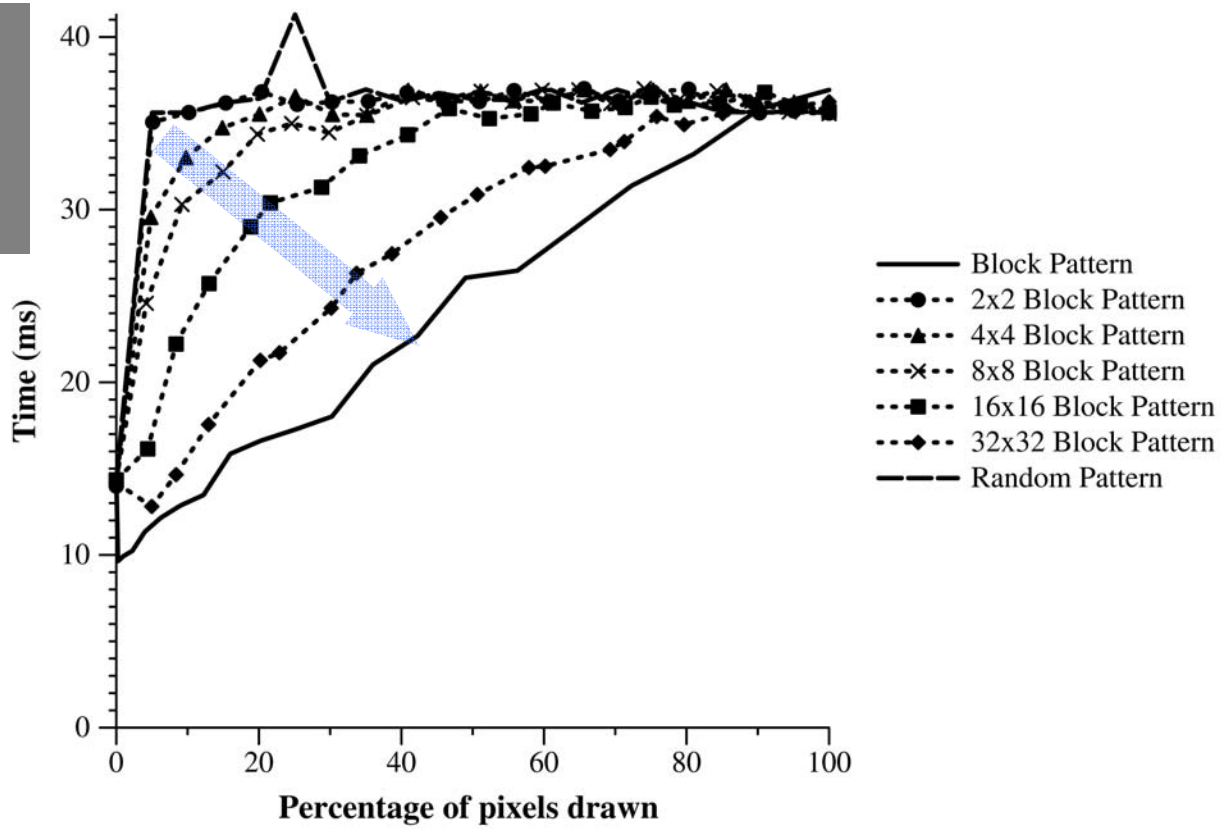
Results – Branching - NVIDIA



NVIDIA 7900GTX

Results – Branching - NVIDIA

Performance increases with branch coherence

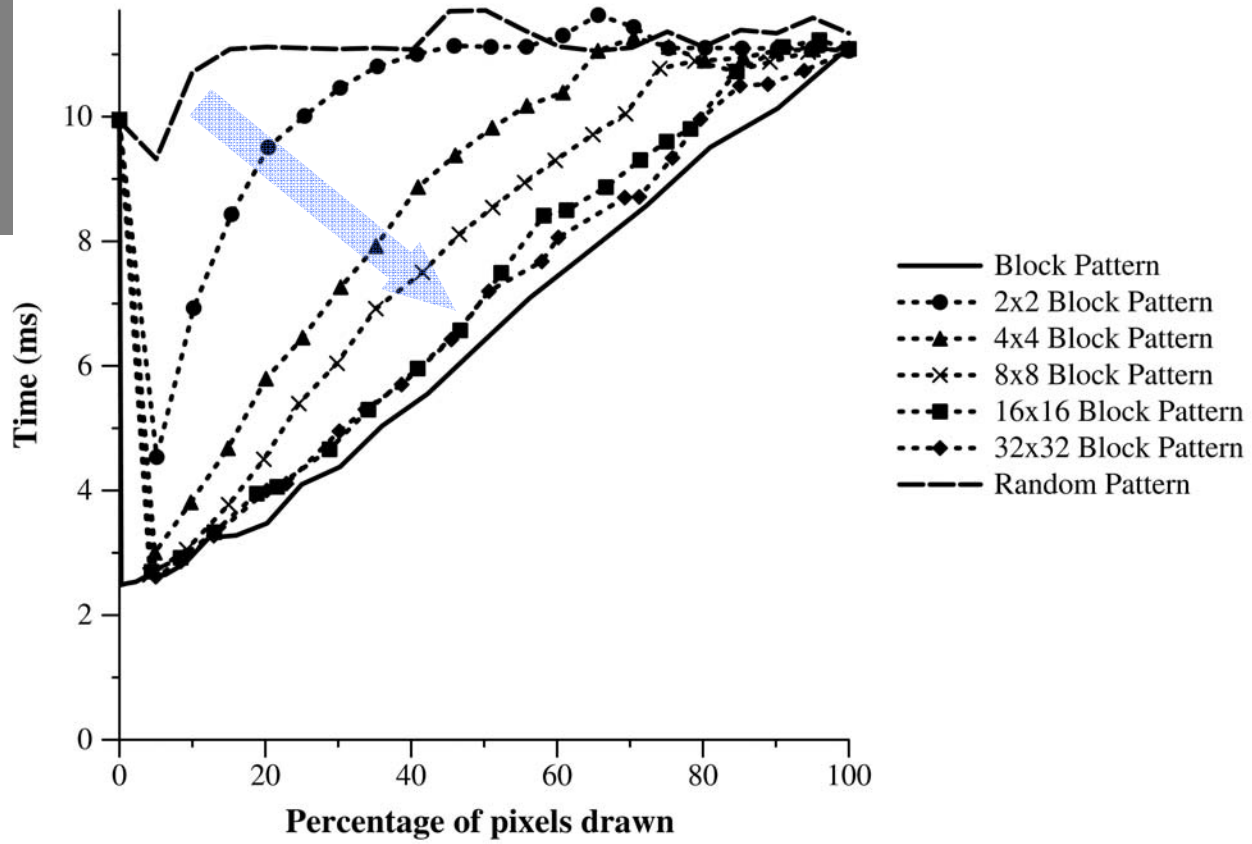


NVIDIA 7900GTX

Need > 32x32 branch coherence

Results – Branching - NVIDIA

Performance increases with branch coherence



NVIDIA 8800GTX

Need > 16x16 branch coherence
(Turns out 16x4 is as good as 16x16)

Summary

- **Benchmarks can help discern app behavior and architecture characteristics**
- **We use these benchmarks as predictive models when designing algorithms**
  - Folding@Home
  - ClawHMMer
  - CFD
- **Be wary of driver optimizations**
  - Driver revisions change behavior
    - Raster order, scheduler, compiler