Tool of the day: Shell scripting

Single-thread performance

Multi-thread performance
Bits and pieces

- Don’t have a project? Let’s fix that very soon
- HW5: soon
- HW6: due today
- Dec 5: Last day of regular class
- Dec 12: Legislative Day
- Dec 17/18/19: Project presentations
- Don’t have grade reports for HW1...4? Talk to me
Tool of the day: Shell scripting

Single-thread performance

Multi-thread performance
Shell scripting

Demo time
Shell scripting

All you ever wanted to know about scripting:

- `man bash`
Tool of the day: Shell scripting

**Single-thread performance**
- How about actually doing work?
- Compilers and what they do to your code

**Multi-thread performance**
Recap

Single-thread performance recap:

- CPU bits
  - Bus, Register File, ALU, Memory Interface, Machine language
- Memory hierarchy
  - Latency, bandwidth
  - Caches: lines, associativity
  - Locality, working set
- Pipelines
  - Dependencies
  - Branch predictor
  - Software pipelining, loop unrolling
Tool of the day: Shell scripting

**Single-thread performance**
- How about actually doing work?
- Compilers and what they do to your code

**Multi-thread performance**
Remember SIMD?

Credit: Kayvon Fatahalian (Stanford)
Remember SIMD?

GPU Idea #2
Amortize cost/complexity of managing an instruction stream across many ALUs
→ SIMD

Credit: Kayvon Fatahalian (Stanford)
Recall simple processing core

Execute

Context

Same principle works well on CPUs, too!

Software Single-thread performance Multi-thread performance

GPU Idea #2

Amortize cost/complexity of managing an instruction stream across many ALUs

→ SIMD
Remember SIMD?

GPU Idea #2

Amortize cost/complexity of managing an instruction stream across many ALUs

→ SIMD

Credit: Kayvon Fatahalian (Stanford)
Remember SIMD?

GPU Idea #2

Amortize cost/complexity of managing an instruction stream across many ALUs

→ SIMD

Same principle works well on CPUs, too!

Credit: Kayvon Fatahalian (Stanford)
Talking to SIMD

Ways of expressing SIMD:
- Not at all (-ftree-vectorizer-verbose=2, pray)
- “Implicit” (OpenCL workgroups)
- “Explicit” (many ways)

OpenCL is also one of the saner ways of expressing explicit vectorization. (even on the CPU)

Other ways:
- “Intrinsics”: mm256_hadd_ps
- GCC extensions
- ispc
CL vector demo
Outline

Tool of the day: Shell scripting

**Single-thread performance**

How about actually doing work?

Compilers and what they do to your code

**Multi-thread performance**
Inside a compiler

Two subsequent stages agree upon a data exchange format “Intermediate Representation”– often a little like assembly. Almost always more complicated: “Passes” include optimizers, . . .

http://llvm.org/demo/
Inside a compiler

- Preprocessor
- Parser
- Code generator
- Assembler
- Linker
  (Dynamic Linker)

Two subsequent stages agree upon a data exchange format—"Intermediate Representation"—often a little like assembly. Almost always more complicated: "Passes" include optimizers, . . .

http://llvm.org/demo/

Software  Single-thread performance  Multi-thread performance
Inside a compiler

- Preprocessor
- Parser
- Code generator
- Assembler
- Linker
  - (Dynamic Linker)

Two subsequent stages agree upon a data exchange format

“Intermediate Representation” – often a little like assembly

Almost always more complicated:
“Passes” include optimizers, …

http://llvm.org/demo/
Register allocator:
• Important
• Complicated

Failure: ‘Register Spill’

Not dramatic on the CPU (L1 is fast)

Very dramatic on the GPU
Compilers and the register file

Register allocator:
  • Important
  • Complicated

Failure: ‘Register Spill’

Not dramatic on the CPU (L1 is fast)

Very dramatic on the GPU

Demo

Registers most effective when data can be reused many times
Pointer aliasing demo
Pointer aliasing demo

Not the only thing to go wrong with pointers...
Alignment

Match base address of:

- Single word: double, float
- SIMD vector
- Larger structure

To:

- Natural word size
- Vector size
- Cache line

Matched structure
Alignment

Match base address of:

- Single word: double, float
- SIMD vector
- Larger structure

To:

- Natural word size
- Vector size
- Cache line

Matched structure

Software  Single-thread performance  Multi-thread performance
Alignment

Match base address of:

- Single word: double, float
- SIMD vector
- Larger structure

To:

- Natural word size
- Vector size
- Cache line

Matched structure

OK
Alignment

Match base address of:
- Single word: double, float
- SIMD vector
- Larger structure

To:
- Natural word size
- Vector size
- Cache line

“Bad”
Alignment

Match base address of:
- Single word: double, float
- SIMD vector
- Larger structure

To:
- Natural word size
- Vector size
- Cache line
Alignment

Match base address of:
- Single word: double, float
- SIMD vector
- Larger structure

To:
- Natural word size
- Vector size
- Cache line

Matched structure

(matched base address)

Software | Single-thread performance | Multi-thread performance
Alignment

Match base address of:
- Single word: double, float
- SIMD vector
- Larger structure

To:
- Natural word size
- Vector size
- Cache line

Matched structure

Comes in two flavors:
- Actual alignment
  \texttt{malloc} \rightarrow \texttt{posix\_memalign}
- Compiler-known alignment
  \texttt{float \_attribute\_ \((\text{aligned (64)})\) *a}
Alignment

Match base address of:

- Single word: double, float
- SIMD vector
- Larger structure

To:

- Natural word size
- Vector size

Comes in two flavors:

- Actual alignment
  - `malloc → posix_memalign`
- Compiler-known alignment
  - `float __attribute__((aligned (64))) *a`

No difference on Sandy Bridge

More difference on other machines
(e.g. AMD Opteron)
Alignment

Match base address of:

- Single word: `double, float`
- SIMD vector
- Larger structure

To:

- Natural word size
- Vector size
- Cache line

Comes in two flavors:

- Actual alignment
  
  `malloc → posix_memalign`

- Compiler-known alignment
  
  `float __attribute__((aligned (64))) *a`

No difference on Sandy Bridge

More difference on other machines
(e.g. AMD Opteron)

Brief demo
Other compiler optimizations

More techniques:

- Inlining (see HW6)
- Unrolling
- Vectorization

Many of these need tunable parameters. From where?

- \(-\text{march}=\text{native} \quad -\text{mtune}=\text{native}\)
- Profile-Guided Optimization
From the horses’ mouth

- **AMD Optimization Manual**
  - Good source-level C part at the beginning
- **Intel Optimization Manual**
  - Dual audience: Compiler writers, users

Grab bag of good practices:

- Use indices rather than pointers (easier to reason about)
- Extract common subexpressions
- Make functions static
- Use `const`
- Avoid store-to-load dependencies
Outline

Tool of the day: Shell scripting

Single-thread performance

Multi-thread performance
  Memory-related
Difference to single-thread?
Multi-thread performance

Difference to single-thread?

**Memory System** is (about) the only shared resource.

All ‘interesting’ performance behavior of multiple threads has to do with that.
Outline

Tool of the day: Shell scripting

Single-thread performance

Multi-thread performance
  Memory-related
Multiple threads

Threads v. caches demo
Questions?

?
Image Credits

- Pebbles: sxc.hu/topfer