

Programming Parallel Computers

Index

Material

Introduction · Why parallelism? · Programming modern CPUs · Programming modern GPUs · Course idea and prerequisites

Chapter 1: Role of parallelism

- **Why do we need parallelism?** · But what about performance? · After 2000 · New kind of performance · Example: a massively parallel university
- **How to exploit parallelism?** · Creating potential for parallelism and realizing it

Chapter 2: Case study

- **Introduction** · The shortcut problem · Remarks · Interface · Example
- **Version 0: Baseline** · Platform · Benchmark instances · Results
- **Version 0: Assembly code** · Seeing the assembly code · Finding the relevant place · How to interpret it · Interactive assembly
- **OpenMP** · About threads · OpenMP: multithreading made easy · OpenMP parallel for loops · Scheduling directives · Warning! Stay safe! · Back to our application · Results
- **Version 1: Linear reading** · What is the source of the problem? · But why was macOS slower? · How to fix it? · Results
- **Version 1: Assembly code** · Interpretation · Latencies and throughputs · Dependency chain · Interactive assembly
- **Version 2: Instruction-level parallelism** · Independent operations · Instruction-level parallelism is automatic · Implementation · Notes · Results
- **Version 2: Assembly code** · But what about vaddss and %xmm0? · Analysis · Interactive assembly
- **Version 3: Vector instructions** · Vector registers · Some terminology · Vector types in C++ code · Warning: proper memory alignment needed · Implementation · Results · Vectors of doubles
- **Version 3: Assembly code** · Comparison with V1 · Interactive assembly
- **Version 4: Reuse data in registers** · Opportunities for data reuse · Implementation · Results
- **Version 4: Assembly code** · Interactive assembly
- **Version 5: More register reuse** · Basic idea · Choosing the right permutations · Implementation · Results
- **Version 5: Assembly code** · Analysis · Interactive assembly
- **Version 6: Prefetching** · Implementation · Assembly code · Results
- **Version 7: Better use of cache memory** · Cache memory hierarchy · Improving reuse · Shorter rows · Results

Chapter 3: Multithreading with OpenMP

- **Introduction** · Basic multithreading construction: parallel regions · Critical sections · Shared vs. private data · Other shared resources
- **OpenMP parallel for loops** · It is just a shorthand
- **OpenMP parallel for loops: waiting** · Interaction with critical sections · No waiting before a loop
- **OpenMP parallel for loops: scheduling** · Dynamic loop scheduling
- **Parallelizing nested loops** · Challenges · Good ways to do it · Wrong way to do it, part 1 · Wrong way to do it, part 2
- **Hyper-threading** · Hyper-threading helps to keep the CPU busy · When it might help · When it might not help
- **OpenMP memory model: manipulating shared data** · Temporary view vs. memory · Memory model is an abstraction · An example · Rules of thumb · Granularity · Atomic operations
- **More useful features: thread numbers and tasks** · Do-it-yourself parallel for · Controlling the number of threads · Single thread only · Tasks
- **Examples of useful OpenMP constructions** · Divide and conquer — bottom up · Divide and conquer — top down

Chapter 4: GPU programming

- **Introduction** · How much performance in theory? · How much performance in practice?
- **Getting started with CUDA** · CUDA toolkit · Programming model · Multidimensional grids and blocks
- **Version 0: Baseline** · Threads and blocks · GPU side · CPU side · Results
- **Version 0: OpenCL** · GPU side · CPU side · Compilation and linking
- **Version 1: Better memory access pattern** · What does not work · Problem · Solution · Results
- **Version 1: OpenCL**
- **Version 2: Reuse data in registers** · Memory access pattern · Kernel · Kernel for padding · CPU side · Results
- **Version 2: OpenCL** · Results
- **Version 3: Reuse data in shared memory** · Shared memory · Shared memory as a cache · Memory access pattern · Implementation · Results · Are we happy now?
- **Version 3: OpenCL**

Lectures

- **Week 1** · Lectures · Topics covered
- **Week 2** · Lectures · Topics covered · Additional recommended reading
- **Week 3** · Lectures · Topics covered · Additional material
- **Week 4** · Lectures · Topics covered · Additional material
- **Week 5** · Lectures · Topics covered
- **Week 6** · Lectures

Links to external resources · Hardware · OpenMP · SIMD · CUDA · OpenCL · Rust programming language · Low-level programming techniques

Acknowledgments · Copyright and license

Index