Algorithm Engineering: Gliederung

Markus Pawellek
markuspawellek@gmail.com

February 5, 2018

1 C++-Basics

- use C++ Reference (cppreference.com) and the official standard
- C++ is based on different paradigms
- use cmake to organize projects
- use `nullptr` instead of `NULL`
- C++ random number generation
- prefer `std::cout` over `std::printf`
- include headers that you are using and no headers you are not using
- do not write code you do not use, make everything as simple as possible
- automize the processes using tools like clang-tidy, clang-format, cpplint, etc.
- code is written for humans not for the computer
- use Google C++ style guide
- do not use variable-length arrays
- measuring time
- read arguments from the command line
- avoid using `unsigned`-qualifier
- avoid manually managing memory with `new` and `delete`
- rely on RAII-principle
- prefer `enum class` over `enum`
- use `constexpr` instead of `enum`
- pass an argument by `const &`
- for readability one declaration per line
- for teamwork everything should be written in English with ASCII-characters

2 Motivation

- worst-case behavior and discrepancies between theory and practice
- cycle of development and algorithm engineering
- What is algorithm engineering?
- Why Algorithm Engineering?
- Experiment with Algorithms
- Paths to Glory
- Engineering Aspects
- Robert Pike Rules
- Unix Philosophy

3 Version Control with Git

- Distributed Version Control with Git
- How Git stores Data
- Interacting with Git
- Creating a repository
- Adding content
- Status
- Showing Modifications
- Committing
- Commit Message
- Unstaging
- Untracking content
- Checkout
- Branching
- Merging
- Branching and Merging
- Commit History
- Tagging
- Interact with Remote Repositories
- Managing remote repositories
- cloning a repository
- fetching from remote repositories
- Git reference

4 Building Software

- configure, build and install
- guidelines for building software
- make and makefiles
- autotools

5 CMake

- Interest over time
- cmake
- using cmake
- example
- `cmake -build .`
- `cmake_minimum_required`
- `project`
- `add_subdirectory`
- `find_package`

6 High-Performance Motivation

- realistic environment
- Moore's Law
- more Cores
- wider vector units
- consequences for developers
- outlook and history
- xeon phi architecture
- chip layout
- core structure
- hardware threads
- cache organization
- prefetching
- vector processing units
- performance metrics
- reality of parallelization
- example
- amdahl's law
- conclusion

7 sofware testing

- what is software testing
- why test software
- testing levels: unit tests, integration tests

7.1 testing tools

- unit testing with cmake
- continuous integration (CI) using gitlab or github
- catch - test driven development in c++: key-features, examples, basics, catch-sections, bdd-style test cases

7.2 what makes good unit tests

- Step 0 - write tests
- properties
- correctness
- readability
- completeness
- demonstrability
- resiliance: ordering, nonhermeticity, deep dependence
- recap: what is the goal?

8 Caches

- cache structure
- cache attached to core
- cache lines
- cache miss
- intel xeon phi cache
- example
- writing to cache
- example
- communication
- cache coherency
- false sharing
- example
- scaling
- how to avoid false sharing
- summary

9 Parallelism

- multiple levels of parallelism
- vectorization
- current vectorization hardware
- vector registers
- history
- why vectorization
- compile will do?
- how do we vectorize?
- intrinsics review and do it yourself
- steps to vectorization
- vector-loop requirements
- how to think of auto-vectorization
- how to instruct the compiler
- challenge: loop dependencies: read after write, write after read
- aliasing
- aliasing and compilers
- resolving dependencies
- example
- caches and alignment
- alignment for vectorization
- alignment on 512-bit wide registers
- how to align: tell the compiler about it
- what happens without alignment
- alignment - multi-dimensional data structures
- padding
- strided access
- streaming stores
- example
- structure-of-arrays versus array-of-structures
- cache-blocking
- blocking-basics, blocking-principle, loop-splitting, loop-interchange
- blocking versus non-blocking
- example: matrix transposition
- example: matrix multiplication
- vectorization guidelines

10 multiple levels of parallelism

- dot product sequential
- dot product vectorized
- dot product `std::thread`
- dot product OpenMP

10.1 OpenMP

- history
- OpenMP
- OpenMP-Process
- basics
- library
- example: hello world
- execution model
- communication
- parallel directive
- how many threads
- work-sharing, for-directive
- example: for-directive
- sections-directive, example
- single-directive, example
- synchronization, example
- critical-directive, example
- what does critical do internally
- atomic-directive, example
- tasking: concept, directive, clauses, example
- when do tasks get executed
- taskyield directive
- openmp environment