Performance engineering
Project kick-off

April 7th , 2022

A.L.Varbanescu@uva.nl

To do

- Objective
- Process
- Example
- Q&A

- Project selection

Objective

- Building a toolbox for performance engineering on modern architectures

- Modern architectures =
  - Parallel processors (multi-core CPUs)
  - GPUs
  - Xeon Phi (Knights Corner)
  - Clusters
- Toolbox = set of tools to use
  - For you!

50% of your grade!

So … opportunity to practice

- The full performance engineering experience
  - Select and analyze an application
  - Select and analyze the architecture/machine
  - Performance measurement, analysis, modeling
  - Performance tuning
  - Iterate till goal is achieved
  - Results analysis

Practical matters.

Project plan (tentative)

- Week 1 (today): Introduction + examples
- Week 2: Prototype 0: reference implementation
- Week 3: Experimental setup
- Week 4: First model, maybe some optimizations (Prototype 1 (!)= Prototype 0)
- *Mid-term! (week 5)*
- Week 6: Prototypes 2,3,4,…n: Performance analysis, modeling, tweaking
- Week 7: Performance prediction for Prototypes 1 and n
- Week 8: Did you meet the performance requirements?

Week 1: Project kick-off

- Decide on the application
  - By Thursday, 14/4, noon-ish + submit on Canvas
    - The sooner, the better …
- Decide on performance requirements
  - By default: best that can be achieved
  - Exceptions: application-specific + justified
- Decide on the group
  - At most 4 people
- Start on the reference implementation

Weeks 1-2: Reference implementation

- Typically, sequential code
  - Can be yours, can be taken from books, can be taken from Github
  - Make sure you understand the implementation
  - Keep it simple!
- Reason about the next steps

Week 3: experimental setup + performance analysis

- Make sure you build a comprehensive experimental setup
  - Enough datasets
  - Enough platforms (if needed)
  - Enough parameters
- In-depth performance analysis
  - Profiling
  - Bottleneck analysis

Weeks 4-5: model + optimization

- First performance model for the application
- Might fix some of the performance challenges
  - Thus, optimize the sequential code
- Consider prediction
  - What model? What to predict?

Week 6: Parallel prototype(s)

- Design parallel solution
  - Which architecture?
  - Which algorithm?

- Revise the experimental setup
  - New parameters for the performance engineering/tuning?
  - More performance measurement

- Performance analysis
  - New bottlenecks, New ideas for improvement => new prototypes

- Build/Revise performance model(s)

Week 7: Performance prediction [?]

- Improve models for prediction
- Use the model you have
  - Calibrated
  - Updated to the latest prototype
- Attempt performance prediction
  - Check versus measurements
  - Interpret the results

Week 8: Performance engineering analysis

- What was the final result?
- How did the process go?
- What was the most difficult?
- What was the most “entertaining”?
- What did you have to “search” the most?
- What would you do different now?
- What are guidelines that you take out of this work?
- …

Week 5: Project report + presentation

- Report (50%):
  - Skeleton of the report (will be published):

    Optional!

    - Introduction
      - Application description, performance requirements
      - Approach (algorithm, restrictions, etc.)
    - Related Work (if any)
    - Design+implementation+analysis “Reference”
      - Requirements + design
      - Implementation in pseudocode
      - Experimental setup and analysis of the results
      - Model
    - Proposed “Prototype I”
      - Parallel algorithm + parallel architecture
    - Plan for the rest of the work
- Presentation time: 5-7 minutes
  - Q&A: +2 minutes

Week 8: Project report + presentation

- Report (add the remaining 50%):
  - Week 5 content +
  - Proposed “Prototype i”
    - Parallel algorithm + parallel architecture
    - Performance analysis
    - Performance modeling
  - Overall results
  - Performance prediction
  - Process analysis and lessons learned
  - Conclusion, future work
- Presentation time: 10-12 minutes
  - Q&A: +2 minutes

Rules

- Can use any reference code
  - But you must understand it thoroughly
- MUST implement your own prototypes
- Can use tutorials
  - But *must* reimplement the code yourself
- Can use any tools
- Can use any models/modeling techniques
- Can use any machine
- Must submit code (all versions) + report + final presentation

Submissions & deadlines

- Project idea
  - Optional, by 14/4 at noon
    - Follows discussion in class on the idea in the first lab
    - Submit via canvas or email
    - Feedback provided asap, FCFS
- Project mid-term report
  - Optional, by 13/05 at noon
    - Submit via canvas or email
    - Feedback provided asap, FCFS
- Project, final report, presentation, code
  - Mandatory, by 2/06 at noon (talk) and by 3/06, 20:00 (report)

Examples from previous years

- Ray tracing
- Label propagation
- All-pairs shortest path
- Eve Online
- RushHour
- Gradient descent optimization
- Barnes-Hut on the GPU
- Graph matching

Other ideas

- **benchmarking** suites: SPECAccel, NAS, Rodinia, Parsec, Lonestar or LonestarGPU and provide optimized version of those applications.
- take a look at the **examples** in CUDA SDK or in the OpenMP examples
- take a **difficult problem** - e.g., graph isomorphism - and try to provide algorithms - sequential or parallel - to solve it.
- take a **simple problem** - convolution, edge detection, 2D or 3D stencils, triangle counting, etc - and try to super-optimize your solution.
- take a **real-life problem** - game of life, simulations of some kind, fluid dynamics, data clustering, embedding for visualization - and come up with parallel versions for them, using CPUs or GPUs.
- take **data-dependent problems** - sparse-matrix operations, graph processing operations - and focus on performance analysis and potentially providing different algorithms for different (types) of data.
- take known **difficult parallel problems** - community detection, graph coloring, label propagation - and try to understand how you can still get performance from parallelism and what are the modelling challenges.
- take a **problem you had to solve in a previous course** - the PMMS heat dissipation, histogram computation, sorting.