Performance engineering in the PeCoH project

H. Stüben, K. Himstedt, N. Hübbe, S. Schröder, M. Kuhn,
J. Kunkel, T. Ludwig, S. Olbrich, M. Riebisch

Performance Engineering Workshop
TU Dresden
26 March 2019

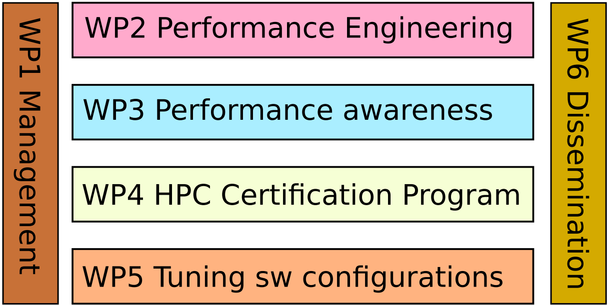
PeCoH is supported by Deutsche Forschungsgemeinschaft (DFG) under grants LU 1335/12-1, OL 241/2-1, RI 1068/7-1

Overview

Partners

- computer science at Universität Hamburg
  - *Scientific Computing*
  - *Scientific Visualization and Parallel Processing*
  - *Software Engineering*
- supporting HPC centres
  - DKRZ – Deutsches Klimarechenzentrum
  - RRZ – Regionales Rechenzentrum der Universität Hamburg
  - TUHH RZ – Rechenzentrum der TU Hamburg

Work packages and topics



Performance engineering

Goals

- improve productivity
- bring software engineering closer to HPC

Topics

- determination of concepts
- benefit of data analytics
- benefit of in-situ visualization
- compiler-assisted development
- code co-development

Software engineering techniques in HPC

Goal: motivate HPC users to

- use an integrated development environment (IDE) *(eclipse)*
- use the IDE for debugging
- employ automated testing (unit testing)

Interesting tool found

- *Visual Studio Code* (open source)
  - plugins for: bash, Fortran, . . .
  - full screen debugging based on *gdb*

Performance awareness

Motivation

- HPC hardware and its operation are costly
- resources are requested in abstract terms
  - compute time, storage/archive capacity
- limited feedback on resource utilization

→ users and even experts are often not aware of cost

Goals

- Raise performance awareness by providing cost feedback

→ reduce overall cost and increase scientific output

→ help to make decisions on optimization effort

Approach and tasks

- model cost of resources (storage, compute, . . .)
- integrate cost models into workload manager
- deploy feedback tools on production systems

Cost models

Simple model

- devide total machine cost by node hours

Refined model

- procurement costs: compute, storage, infrastructure
- operational costs: energy, service, staff

Example data

- compute: 0.33 € to 0.47 € (per node hour)
- disk: 12.80 € (per month and TB)

Write-up

https://wr.informatik.uni-hamburg.de/_media/research/projects/pecoh/d3_1-and-d3_3-modelling-hpc-usage-costs.pdf

Cost modelling: a simple example

Question: What is the value of optimization?

Assumptions

- unoptimized run needs 10,000 node hours
- the optimizing scientist costs 60 k€ per year

Example alternatives

1. run code as is
2. spend an hour to make code run 2% faster
3. spend a day to make code run 5% faster

Cost modelling: a simple example

Question: What is the value of optimization?

Assumptions

- unoptimized run needs 10,000 node hours
- the optimizing scientist costs 60 k€ per year

Example alternatives

1. run code as is
2. spend an hour to make code run 2% faster
3. spend a day to make code run 5% faster

Answer: alternative 2 leads to lowest cost

- saving: 200 node hours ≈ 66€
- investment: one working hour ≈ 36€
- total cost: 1. ≈ 3300€, 2. ≈ 3270€, 3. ≈ 3423€

Feedback on costs of HPC usage

We studied practical options to give feedback

- compute Time $\rightarrow$ SLURM epilogue
- online storage $\rightarrow$ daily/monthly reporting
- archive space $\rightarrow$ instrumentation of archiving commands

Scripts that use cost models were implemented

- script 1: job cost estimation
  - read a cost model configuration
  - analyse SLURM jobs accordingly
- script 2: statistical analysis of finished jobs
  - computes averages, std-deviations and quantiles

Feedback from DKRZ user group meetings

- most user representatives don't find it necessary to provide cost information for every batch job
- some users would like to be able to convert between compute and storage resources according to their cost

HPC Certification / *"HPC-Führerschein"*

Motivation

- HPC-Führerschein
  (corresponds to a *Golf Proficiency Certificate* in Singapore)
  - provide HPC beginners with basic skills required for using HPC clusters
  - check success by self testing
- HPC certification program
  - provide HPC teaching material at all levels
  - establish HPC certificates (like other IT certificates)
  - *HPC-Certification Forum* started

$\rightarrow$ `http://hpc-certification.org`

Approach and tasks

- listing and classification of competences
- development of a certification program
- creation of workshop material
- provision of an online tutorial
- enabling an online examination

Listing and classification of HPC competences

Main topics

- general HPC knowledge
- performance engineering
- software engineering
- use of the HPC environment

Roles

- tester (running programs)
- builder (compiling programs)
- developer (writing programs)

Levels

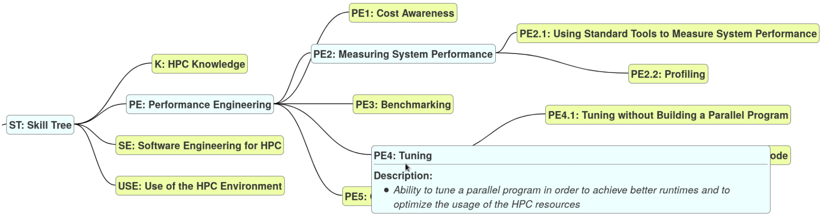
- basic, intermediate, expert

Result

- `https://www.hhcc.uni-hamburg.de/files/hpccp-concept-paper-180601.pdf`

Skill tree

The main topics are subdivided into a *skill tree:*



→ `https://www.hhcc.uni-hamburg.de/en/hpc-certification-program/hpc-skill-tree.html`

Views

Our framework allows creation and selection of *views* on the skill tree, e.g.

- basic tester
- (intermediate) developer
- groups of users (chemists, climate researches, . . .)
- analogy: targets and dependences in *Makefiles*

Content production workflow

Markdown

- easy to use lightweight markup language, widely used for documentation purposes (e.g. on GitHub)
- plain text editor is sufficient
- supports formulae, syntax-highlighting, tables, hyperlinks, inclusion of images
- content of a single skill is based on a list of Markdown files

Extensible Stylesheet Language Transformations (XSLT)

- XSLT-programs generate Makefiles for Pandoc from skill tree (XML) and contents (Markdown)

Pandoc

- convertions between many markup formats
- used to convert .md-skill content files to .html, .pdf, .tex

Tuning of packaged software

Goals

- provide settings for tuning parameters
  - runtime settings
    - e.g. $TMPDIR, process placement, thread number
  - compile time settings
    - e.g. compiler flags, libraries

Approach and tasks

- determination of tuning possibility (from manuals)
- setup of realistic use cases (cooperation with users)
- benchmarking (with use cases)
- documentation (success stories)

Tuning statistical computations with R

Compile time settings

- use OpenBLAS or MKL (slightly better than OpenBLAS)
- -O3 delivered best performance
- no benefit from *profile guided optimization*
- use at least simple parallelization via `foreach()`

Tuning results for R

Use case A: “R Benchmark 2.5” [Simon Urbanek]

- mix of matrix operations (cross product, eigenvalues) and other parts (recursions, loops)
- single process speedup: ≈ 4 using MKL
- parallel speedup with OpenMP: $\approx 15\,\%$

Use case B: parallelization of regression analysis

- speedup ≈ 30 on 4 nodes $\times$ 16 cores (64 cores)

Use case C: analysis of satellite night images

- parallelization with `foreach()`
- speedup: 126 on 32 nodes $\times$ 4 cores (128 cores)
  no domain decomposition $\rightarrow$ every process has all data

Climate Data Inferface (CDI) optimization

Problem: reading of compressed files with CDI is slow

Analysis: slowdown due to libaec and GribAPI

- GribAPI is hard to change

Solution: parallelization of decoding

Result:

- significant speedup on compressed data
- slight slowdown on uncompressed data
  - caches become less effective, overheads are significant
- decoding is concurrent to further processing steps

Measurements

Test case: copy data from one file to another

Memory size: 113GB (double), file size: 28GB (short), compressed: 13GB

	compressed		uncompressed	
	time	throughput	time	throughput
sequential input	334s	340$^{MB}/_{s}$	66s	1710$^{MB}/_{s}$
→disk access	5s	2600$^{MB}/_{s}$	25s	1120$^{MB}/_{s}$
→decoding	329s	343$^{MB}/_{s}$	41s	2756$^{MB}/_{s}$
parallel input	66s	1710$^{MB}/_{s}$	89s	1269$^{MB}/_{s}$
→disk access	27s	481$^{MB}/_{s}$	57s	491$^{MB}/_{s}$
→overheads	41s	2756$^{MB}/_{s}$	33s	3424$^{MB}/_{s}$
parallel decoding	462s	244$^{MB}/_{s}$	141s	801$^{MB}/_{s}$

Cache effects heavily influence many values

Overheads are mostly extra `memcpy()` calls

PeCoH web pages

HHCC – Hamburg HPC Competence Center

- `https://www.hhcc.uni-hamburg.de`

Scientific computing group

- `https://wr.informatik.uni-hamburg.de/research/projects/pecoh/start`

Conclusion

- PeCoH brings Hamburg HPC centers closer together
- broad range of topics
- most results are in certification and training
  - topics were structured
  - framework for producing training material was developed
  - writing material is in progress