PeCoH – Performance Concious HPC
Status 2019

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

9. HPC-Status-Konferenz der Gauß-Allianz
Paderborn Center for Parallel Computing (PC^2)
18 October 2019

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

Overview

WP1 Management

WP2 Performance Engineering

WP3 Performance awareness

WP4 HPC Certification Program

WP5 Tuning sw configurations

WP6 Dissemination

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

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*

Code co-development

- Climate Data Inferface (CDI) optimization
  - factor 5 speep-up for compressed I/O

Performance awareness

Idea: raise performance awareness by providing cost feedback

Approach and tasks

- model cost of resources (storage, compute, . . .)
  - https://wr.informatik.uni-hamburg.de/_media/research/projects/pecoh/d3_1-and-d3_3-modelling-hpc-usage-costs.pdf
- integrate cost models into workload manager
  - https://github.com/pecoh/cost-modelling
- deploy feedback tools on production systems
  - discussion at DKRZ user group meeting

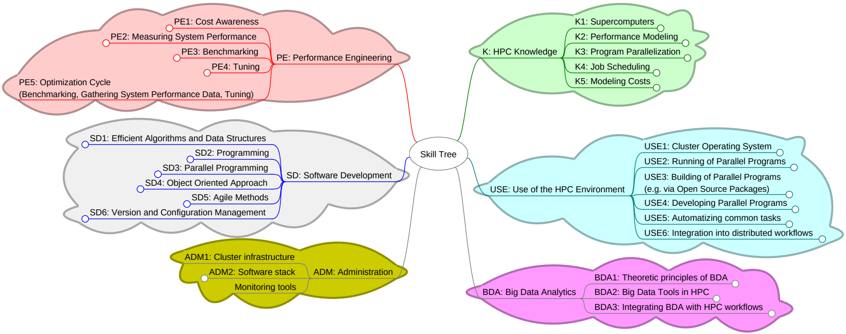
HPC Certification / *"HPC-Führerschein"*

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

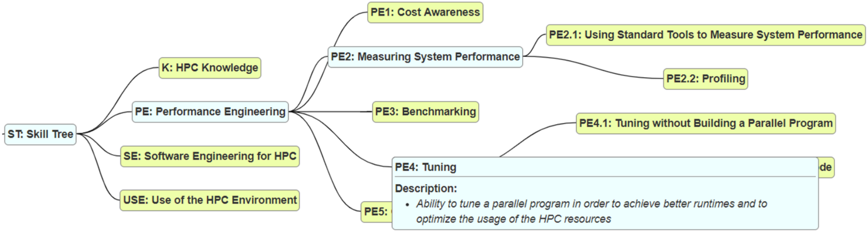
$\rightarrow$ `http://hpc-certification.org`

Representing HPC competences by skills



First two levels of the current skill tree

Classification of HPC competences



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

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

- skills close to the root: *generic*
- skills at leaf level: *specific*
- skill tree acts as a database
  - implementation is based on XML
  - corresponding XML Schema (XSD) assures consistency

Definition of a skill (1)

Each skill consists of

- unique name / ID

  e.g. *Benchmarking / PE3*

- background information
  - motivation

    benchmarking example:
    Benchmarking is essential in the HPC environment to determine speedup and efficiencies of a parallel program
  - main focus

    benchmarking example:
    Benchmarking emphasizes on carrying out controlled experiments to measure the runtimes of parallel programs

...

Definition of a Skill (2)

...

- aim ("What is covered by the skill?")

  benchmarking example:
  comprehending and describing the basic approach of benchmarking to assess speedups and efficiencies of a parallel program

- learning outcomes ("What are the students learning?")

  benchmarking example (extract):
  measuring runtimes (e.g. /usr/bin/time)
  performing experiments using 1, 2, 4, 8, 16, ... nodes
  generating a typical speedup plot
  ...

- list of dependences from sub-skills

  analogy: targets and dependences in a *Makefile*

Views

Additional attributes allow to generate *views* on the skill tree

- educational levels: *basic*, *intermediate*, *expert*
  - *expert* contains *intermediate*
  - *intermediate* contains *basic*
- user roles
  - tester (running programs)
  - builder (compiling and linking programs)
  - developer (writing programs)
- possible extension: scientific domains
  - astrophysicists
  - chemists
  - climate researchers
  - ...

View example: Getting started with HPC Clusters

GSWHC-B Getting Started with HPC Clusters

- K1.1-B System Architectures
- K1.2-B Hardware Architectures
- K1.3-B I/O Architectures
- K2-B Performance Modeling
    - K2.1-B Performance Frontiers ← CURRENT READING POSITION
- K3.3-B Parallelization Overheads
- K3.4-B Domain Decomposition
- K4-B Job Scheduling
- USE1-B Use of the Cluster Operating System
    - USE1.1-B Use of the Command Line Interface
    - USE1.2-B Using Shell Scripts
    - USE1.3-B Selecting the Software Environment
- USE2.1-B Use of a Workload Manager
- PE3-B Benchmarking

Content production workflow challenge

Requirements

- support of various media types / target formats
  - screen device for e-learning
  - printer device for tutorials and handouts
- no “duplication” of content files
- common source format for content files to produce
  - HTML for browsable learning material, presentation slides
  - LaTeX, PDF for printed tutorials, handouts, presentation slides
- integration with the skill tree database (XML)
- automated build process after changing files

Content production workflow solution

Markdown

- easy to use lightweight markup language
- widely used for documentation purposes (e.g. on GitHub)
- supports formulas, syntax-highlighting, tables, hyperlinks, embedding of images, . . .
- content of a single skill: list of Markdown files

XSLT (Extensible Stylesheet Language Transformations)

- XSLT-programs generate Makefiles for Pandoc from skill tree data (XML) and content files (Markdown)

Pandoc

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

Example: Amdahl's Law – target format: HTML

https://www.hhcc.uni-hamburg.de/hpc-certification-program/getting-started-with-hpc-clusters-b/getting-started-with-hpc-clusters-b-y-performance-frontiers-b.html

ABOUT US　PECOH PROJECT　PERFORMANCE　CERTIFICATION　SUCCESS STORIES

General Formulation

The parallelizable part of a program can be presented as some fraction α.

The non-parallelizable, i.e. sequential, part of the program is thus $(1-\alpha)$.

Taking T_1 as total runtime of the program on a single core, regardless how many cores n are available, the sequential runtime part will be $(1-\alpha)T_1$, while the runtime of the parallelizable part of the program will decrease corresponding to the speedup $\frac{\alpha T_1}{n}$.

The speedup (neglecting overheads) is therefore expressed as

$$S_n \leq \frac{T_1}{(1-\alpha)T_1 + \frac{\alpha T_1}{n}} = \frac{1}{(1-\alpha) + \frac{\alpha}{n}}$$

and the limit for the speedup is given by

$$S_\infty := S_{n\to\infty} = \frac{1}{(1-\alpha)}$$

Example: Speedups for a Given Fraction α of Parallelizable Work

α	$n=4$	$n=8$	$n=32$	$n=256$	$n=1024$	$n=\infty$
0.9	3.08	4.7	7.8	9.7	9.9	10
0.99	3.88	7.5	24	71	91	100
0.999	3.99	7.9	31	204	506	1000

Example: Amdahl's Law – target format: LaTeX/PDF

18 / 64 125%

General Formulation

The parallelizable part of a program can be presented as some fraction α.

The non-parallelizable, i.e. sequential, part of the program is thus $(1-\alpha)$.

Taking T_1 as total runtime of the program on a single core, regardless how many cores n are available, the sequential runtime part will be $(1-\alpha)T_1$, while the runtime of the parallelizable part of the program will decrease corresponding to the speedup $\frac{\alpha T_1}{n}$.

The speedup (neglecting overheads) is therefore expressed as

$$S_n \leq \frac{T_1}{(1-\alpha)T_1 + \frac{\alpha T_1}{n}} = \frac{1}{(1-\alpha) + \frac{\alpha}{n}}$$

and the limit for the speedup is given by

$$S_\infty := S_{n\to\infty} = \frac{1}{(1-\alpha)}$$

Table 4: Example: Speedups for a Given Fraction α of Parallelizable Work

α	$n = 4$	$n = 8$	$n = 32$	$n = 256$	$n = 1024$	$n = \infty$
0.9	3.08	4.7	7.8	9.7	9.9	10
0.99	3.88	7.5	24	71	91	100
0.999	3.99	7.9	31	204	506	1000

Example: Amdahl’s Law – source format: Markdown

```
### General Formulation

The parallelizable part of a program can be presented as some
fraction $\alpha$.

The non-parallelizable, i.e. sequential, part of the program is thus $(1 - \alpha)$.

Taking $T_{1}$ as total runtime of the program on a single core,
regardless how many cores ${n}$ are available,
the sequential runtime part will be $(1 - \alpha) T_{1}$,
while the runtime of the parallelizable part of the program will decrease
corresponding to the speedup $\frac{\alpha T_{1}}{n}$.

The speedup (neglecting overheads) is therefore expressed as

$$S_{n} \leq \frac{T_{1}}{(1 - \alpha) T_{1} + \frac{\alpha T_{1}}{{n}}} = \frac{1}{(1 - \alpha) + \frac{\alpha}{{n}}}$$

and the limit for the speedup is given by

$$S_\infty := S_{n \rightarrow \infty} = \frac{1}{(1 - \alpha)}$$

---------- --------- --------- ---------- ----------- ------------ -------------
 $\alpha$   ${n}=4$   ${n}=8$   ${n}=32$   ${n}=256$   ${n}=1024$   ${n}=\infty$
---------- --------- --------- ---------- ----------- ------------ -------------
$0.9$      $3.08$    $4.7$     $7.8$      $9.7$       $9.9$        $10$

$0.99$     $3.88$    $7.5$     $24$       $71$        $91$         $100$

$0.999$    $3.99$    $7.9$     $31$       $204$       $506$        $1000$
---
:Example: Speedups for a Given Fraction $\alpha$ of Parallelizable Work
```

PeCoH workshop

Workshop on HPC-training, -education and -documentation

Universität Hamburg, 30-31 July 2019

- presentations from projects in the DFG-Call *Performance Engineering für wissenschaftliche Software*
  - ProfiT-HPC, ProPE, SES-HPC, PeCoH
- and others
  - Goethe-Universität Frankfurt
  - Hessisches Kompetenzzentrum für Hochleistungsrechnen (HKHLR)
  - Paderborn Center for Parallel Computing (PC^2)
- slides are available at
  - `https://www.hhcc.uni-hamburg.de/pecoh/workshop`

Tuning without modifying the source code

Typical optimization parameters

- runtime options
  - process: pinning/mapping, hyperthreading (on/off)
  - MPI: bcast and reduce algorithms, large scale thresholds
  - application specific options for partitioning, tiling
- compilers
  - vendor: GNU, Intel, PGI
  - version
  - optimization level
  - profile guided optimization (PGO)
- libraries
  - MKL, OpenBLAS
- MPI
  - Intel MPI, Open-MPI

Traditional tuning

Manual approach

- problem: huge search space
- benchmarking all combinations is not possible
- thus: benchmark only promising combinations based on
  - educated guesses and/or time consuming profiling
- requires expert and domain specific knowledge
- however, good combinations might get overlooked

In PeCoH applied to

- several R applications
  - use OpenBLAS or MKL (minimally better than OpenBLAS)
  - -O3 already delivered best performance
  - PGO: no benefit

Using the Black Box Optimizer tool (1)

From the experience with the manual approach we looked for a better solution:

Automatic tuning based on genetic algorithms[1]

- parallel program to tune is a black box for the optimizer
- Black Box Optimizer functionality
  - benchmark a set of parameter combinations (“population”)
  - create next improved population by “crossing” and “mutating” parameter combinations with good benchmark results
  - repeat both steps until a good solution is found

[1] Himstedt, K., S. Köhler, D.P.F. Möller, J. Wittmann. Ein Framework-Ansatz für die simulationsbasierte Optimierung auf High-Performance-Computing-Plattformen. In: J. Wittmann, D.K. Maretis (Hrsg.). *Simulation in Umwelt- und Geowissenschaften*. Workshop Osnabrück 2014. Shaker Verlag. Aachen (2014):109-122.

Using the Black Box Optimizer tool (2)

- advantages
  - generic approach
  - huge search space is drastically reduced
  - no expert knowledge for tuning required
  - easy to use
- in PeCoH applied to automatically tune
  - first experiments
    - π calculation
    - Boolean satisfiability problem (SAT)
  - real applications
    - BQCD
    - Fesom2

Black Box Optimizer results

App	Size of Search Space	Best Environment	Opt Level	PGO	HT	BLAS Lib	Binding, Mapping	Other	Pop. Size	Gen.
π	480	gcc-6.4_openmpi-2.1	-O4	no	yes	–	–	–	20	3
SAT	480	gcc-5.2_impi-5.0.3	-O1	yes	yes	–	–	–	20	1
BQCD	20736	fixed (intel)	fixed (-O3)	fixed (no)	no	–	optimized: decomposition, ppn, threads	BQCD specific	100	7
Fesom2	11520	intel-18_impi	-O3	yes	no	MKL	to core, blocked	MPI options manually found	30	10
Fesom2	262E+9	intel-18_impi	-O3	yes	no	Open BLAS	default, default	MPI options via BBO	150	4

- BBO tuning vs. manual tuning
  - BQCD
    - BBO: 10–15% faster than educated guess
  - Fesom2
    - BBO: settings equivalent to manual tuning were found
- observations
  - latest compiler generation is not always the fastest
  - hyperthreading and PGO are sometimes helpful

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
    - workshop organized
- automatic software tuning
    - Black Box Optimization (BBO)
        - method from *soft computing*
        - successfully applied to HPC applications