PeCoH: HPC Skill Tree and Content Production Workflow

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

Workshop on HPC-training, -education and -documentation
RRZ Universität Hamburg
July 31, 2019

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

Overview

Performance Conscious HPC (PeCoH)

Three Hamburg compute centers involved

- German Climate Computing Center / Deutsches Klimarechenzentrum (DKRZ)
- Regional Computing Center / Regionales Rechenzentrum der Universität Hamburg (RRZ)
- Computer Center of Hamburg University of Technology / RZ der Technischen Universität Hamburg (TUHH RZ)

Three Scientific Institutions at Universität Hamburg involved

- Scientific Computing Group
- Scientific Visualization Group
- Software Construction Methods Group

PeCoH: Major Project Goals

- Raising the users' awareness for performance
- Tuning of packaged and user-developed software
- Bringing software engineering closer to HPC
- Development of a cost model embedded into SLURM
- Efficient use of HPC resources by well-trained users
- Reduced efforts for user support

HPC Certification / *"HPC-Führerschein"*

HPC-Führerschein

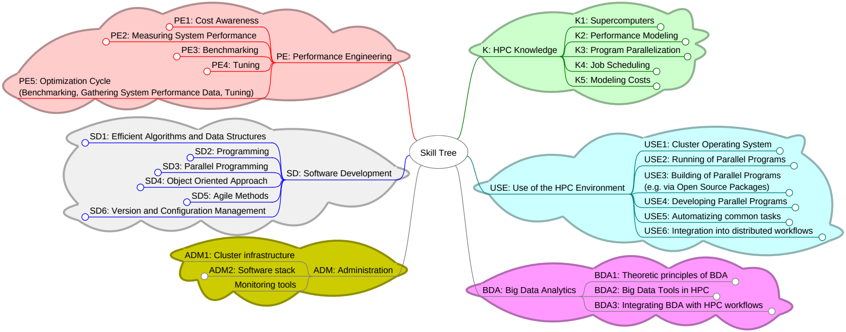
- Provides basic skills required for using HPC clusters
- Includes learning material
- Success is checked by self testing

International HPC Certification Program

- We bootstrapped the *HPC-Certification Forum* (HPC-CF) to sustain the activities → `http://hpc-certification.org`
  - HPC-CF is an independent body
  - Curates curriculum (all skill levels)
  - Establishes generally accepted HPC certificates
- Does not include learning material

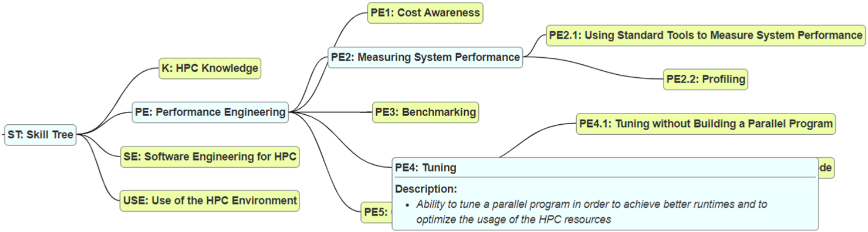
Outline

Representing HPC Competences by Skills



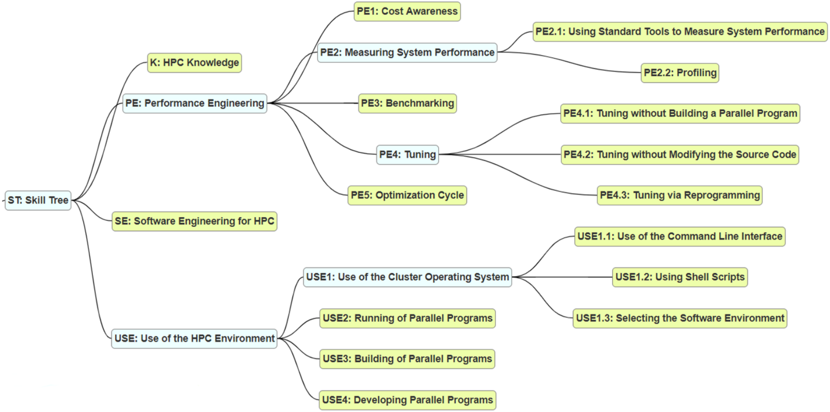
First Two Levels of the Current Skill Tree

Classification of HPC Competences



- Skills close to the root: Generic
- Skills at leaf level: Specific
  - Granularity: 1.5 to 4h of learning material per leaf
- Skill tree acts as a database
  - Implementation is based on XML
  - Corresponding XML Schema (XSD) assures consistency

Why Do We Use a Tree?



- Skills are generally built upon one another
- Skills depend on sub-skills

Current Skill Tree Statistics

There are 6 major branches at level 1

- HPC Knowledge (K)
- Performance Engineering (PE)
- Software Engineering / Software Development (SE / SD)
- Use of the HPC Environment (USE)
- Big Data Analytics (BDA) (recently added)
- Administration (ADM) (recently added)

Skills at level 2: $\approx$ 31; at level 3: $\approx$ 50; at level 4: $\approx$ 5

Skills at the leaf level: $\approx$ 66

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 dependencies from sub-skills

  Analogy: targets and dependencies in a *Makefile*

Views on the Skill Tree

Additional attributes

- 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
  - ...

Sets of Skills Can Easily Be Bundled

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

Available soon via Hamburg HPC Competence Center (HHCC): https://www.hhcc.uni-hamburg.de/

Outline

Challenge

Requirements to be met

- Support of various media types / target formats
  - Screen device for e-learning
  - Printer device for tutorials and handouts
- No “duplication” of content files
- Use of a common source format for content files to produce
  - HTML for browsable learning material, presentation slides
  - TeX, PDF for printed tutorials, handouts, presentation slides
- Integration with the skill tree database (XML)
- Automated build process after changing files

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

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
```

Conclusions

PeCoH

- has a broad range of topics
- this talk: HPC topics classification and content production

HPC Skill Tree

- suitable to classify HPC competences
- supports building of new skills by reusing its subtrees
- contains no learning material itself

Content Production Workflow

- merges the skill tree with content
- automates the transformation process (screen, printer)
- successfully used for the material produced so far
  - ca. 20 skills consisting of ca. 50 content files altogether