PeCoH – Performance Conscious HPC: Status

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

8. HPC-Status-Konferenz der Gauß-Allianz
RRZE Erlangen
9 October 2018

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

General Information About PeCoH

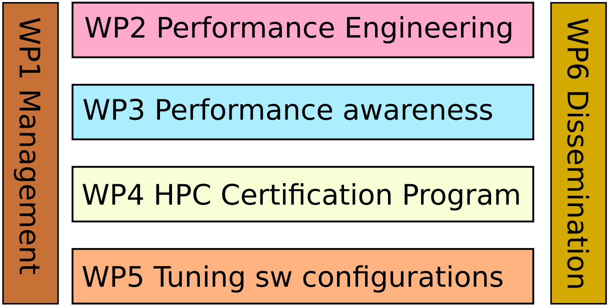
Partners

- 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

Key facts

- Started: 03/2017 (Month 20 now)
- Hired: 03/17 (1 FTE), 06/17 (2/3 FTE), 02/18 (1/3 FTE)

Work Packages and Topics



Outline

Performance Engineering

Goals

- Identify suitable concepts to improve productivity
- Assess benefit of concepts
- Implement selected concepts (co-design with users)

Tasks

1. Identification of concepts
2. Benefit of data analytics
3. Benefit of in-situ visualization
4. Compiler-assisted development
5. Code co-development (includes SWE methods)

Status

1. Identification of concepts (ongoing)
   - Created draft of the deliverable
   - Described benefit assessment
   - Explored SWE methods (benefit analysis to complete)
   - Ongoing: collection of related work (best practices)
2. Benefit of data analytics (pending in plan)
3. Benefit of in-situ visualization (pending in plan)
4. Compiler-assisted development (ongoing)
   - Explored translation of OpenMP to MPI via LLVM
   - Investigated error detection via static code analysis
5. Code co-development (ongoing)
   - Investigated SWE methods for scientific computing

Example: Software Engineering Concepts – Overview

Goal

- Analyse benefit from software engineering practices
  - Practices to efficiently create, maintain and reuse code
  - Assess potential benefit and practicability with scientists

Programming Concepts for HPC

Programming Best Practices for HPC

Software Configuration Management

Software Quality

Agile Software Development

Documentation

Example: Agile Development for Scientific Computing

- Similar challenges as in industry software engineering
  - Not all requirements are known upfront
- New or evolving theories add new system functionalities
  - Agile practices guide software evolution
- Agile practices help scientists to
  - facilitate responsiveness to change, e.g. test new theories
  - allow flexibility and collaboration during development
  - test new and evolving requirements thoroughly
  - achieve an appropriate level of software quality
- Studies show successful application of agile practices[1], [2]

[1] *Erskine et al.:* A Literature Review of Agile Practices and Their Effects in Scientific Software Development

[2] *Sletholt et al.:* What do we know about Scientific Software Development's Agile Practices?

Example: Agile Software Development - Contents

Goal

Identify agile practices that are useful and applicable for scientific software development

- **Test-driven Development and Agile Testing**
  - Automated testing, performance & regression testing
  - Developing test strategies for scientific programs
  - Test frameworks for scientific programs
- **Extreme Programming (XP)**
  - Pair programming, system metaphor, small releases, continuous process, refactoring
- **SCRUM**
  - Sprint, Backlog, Planning, Standup Meeting, Proj. Velocity

Outline

Performance Awareness

Motivation

- Supercomputer hardware and operation is costly
- Users request resources in abstract concepts
  - Compute time, storage capacity, archive capacity
- Users have limited feedback on resource utilization

⇒ Users and even experts are mostly unaware of costs

Goals

Raise performance awareness by providing cost feedback

⇒ put focus of RD&E on relevant inefficiencies

⇒ reduce overall costs and increase scientific output

Approach and Tasks

1. Modeling costs of resources (storage, compute, ...)
2. Integrating of cost models into workload manager
3. Deploying feedback tools on production systems
4. Analyzing data and exploring benefit

Status

1. Modeling costs of resources (storage, compute, ...) (done)
   - Various cost models are defined
   - D3.1: Modelling HPC Usage Costs
2. Integration of cost models into workload manager (done)
   - Software is written to analyze jobs based on the models
     - D3.2 Code for the integration of cost models
   - Designed integration into existing user portal (at DKRZ)
3. Deploying feedback tools (ongoing)
   - Discussed the approach with the DKRZ user-group
   - Awaiting decisions to roll-out tools to production
4. Analyzing data and exploring benefit (started)
   - Apply the cost models to investigate statistics on Mistral

Cost Models

Refined model

- Split procurement costs into compute, storage, infr.
- Consider operational costs: staff, energy, ...
- Utilization of resources (e.g., 50% means 2x costs)
- Configurable parameters in a file

Example data (derived from public information)

- Compute: 0.33 € to 0.47 € (per node hour)
- Storage (online): 12.80 € (per month and TB)
- Storage (offline): 0.68 € (per month and TB)

Cost Modelling: A Trivial Example

Experiment: How much is optimization worth?

Assumptions: Unoptimized run needs 10,000 node hours, the optimizing scientist costs 60 k€ per year

Example alternatives

1. Run code as is (unoptimized)
2. Spend an hour to make code run 2% faster
3. Spend a day to make code run 5% faster

Cost Modelling: A Trivial Example

Experiment: How much is optimization worth?

Assumptions: Unoptimized run needs 10,000 node hours, the optimizing scientist costs 60 k€ per year

Example alternatives

1. Run code as is (unoptimized)
2. Spend an hour to make code run 2% faster
3. Spend a day to make code run 5% faster

Answer: 2. leads to lowest costs

- Saving 200 node hours ≈ 66€
- Investment one working hour ≈ 36€

Total costs: 1. ≈ 3300€, 2. ≈ 3270€, 3. ≈ 3423€

Feedback on Costs of HPC Usage

We investigated practicable options to give feedback

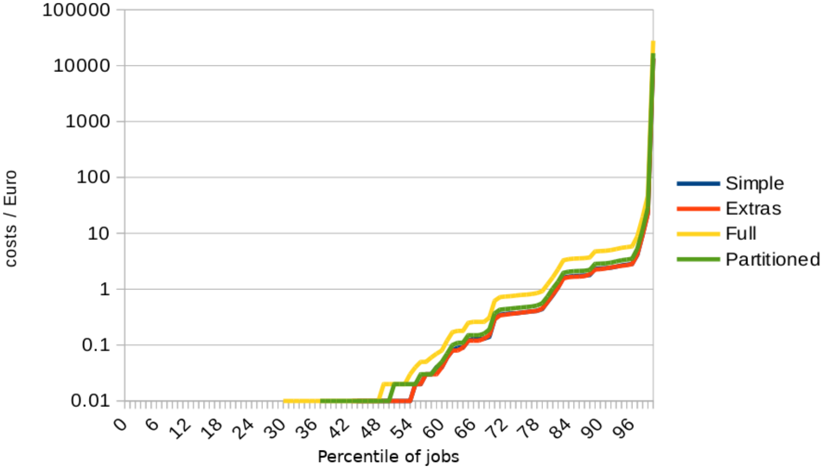
- Compute Time $\Rightarrow$ SLURM epilogue
- Online Storage $\Rightarrow$ daily/monthly reporting
- Archive Space $\Rightarrow$ instrumentation of archiving commands

Implemented scripts for compute cost models

- Script 1: Job cost estimation
  - Read a cost model configuration
  - Analyse SLURM jobs accordingly
  - May run as job epilogue or perform post-mortem analysis
- Script 2: Statistical analysis of finished jobs
  - Computes means, std-devs, and quantiles of costs factors
- Usable by anyone with any cost model

Exemplary Job Cost Statistics

Statistic derived from a day of jobs on DKRZ Mistral supercomputer, using different cost models



Developed Software for SLURM

- Implemented new feature for `scontrol`:
  - Problem: `scontrol` output is impossible to parse safely
    - Job epilogues are very likely to make system vulnerable
  - Solution: Extended `scontrol` for easy and safe usage
  - Status: Proposed, but still unmerged and pending
  - *Patch is available from the link below*
- Developed job epilogue using feature above
  - Reads **cost model** from file and analyzes current job
  - Can run post-mortem without superuser privileges
- Developed script to compute statistics
  - Uses the same cost model input as the epilogue
  - Analyzes data provided by `sacct`
- Docker based test environment available

Outline

HPC Certification Program

Motivation

- Users do often not possess the right level of training
  - Inefficient usage of systems, frustration, lost potential
  - Good training saves compute time and costs!
- Learning is not easy
  - Users need to understand beneficial knowledge for tasks
  - Teaching of different data centers is hard to compare
- Data center has difficulties to verify the skills of users

HPC Certification Program

Goals

- Standardize HPC knowledge representation
- Supporting navigation and role-specific knowledge maps
- Establish certificates attesting knowledge

Approach and Tasks

1. Classification of competences
2. Development of a certification program
3. Creation of workshop material
4. Providing an online tutorial
5. Enabling an online examination

Status

1. Classification of competences (done)
   - Developed schema, technical representation, and content
2. Development of a certification program (done)
   - D4.1: An HPC Certification Program Proposal
   - We started the HPC-Certification Forum
     - Global activity, sustains development of certification
3. Creation of workshop material (ongoing)
   - Developed workflow for public sharing of material
   - Summarized existing work from local centers
   - Some basic material; towards: D4.2: Workshop material
4. Providing an online tutorial (ongoing)
   - Created workflow to create tutorial from material
5. Enabling an online examination (ongoing)

Classification of HPC competences

- HPC skills are generally built upon one another
  - Skills are depending on sub-skills ⇒ tree structure
  - References to skills are possible
- Tree of HPC skills
  - Database for the HPC certification program
  - Implementation is based on XML
  - Corresponding XML Schema (XSD) assures consistency
- Additional attributes are used to describe:
  - Level of a skill (Basic, Intermediate, Expert)
  - Suitability for a user role (Tester, Builder, Developer)
  - Suitability for a scientific domain (Chemistry, Physics, ...)
- Skill tree supports different views on the content
- Live Demo

Considerations

- Granularity of skill descriptions
  - Too fine $\Rightarrow$ content of a skill is predefined at leaf level
  - Too coarse $\Rightarrow$ no help for structuring the material
  - Actual skill tree contains 76 skills
- Certificate definition
  - Bundles a set of skills
  - A users' HPC qualification is certified by successful exams
- Separation of skill, certificates and content provider
  - Similar to the concept of a high school graduation exam
  - Learning material can be provided by different institutions
  - Teachers can add a badge on material: this "trains XYZ"
- Support flexible usage (views on skill tree)
  - Institutions can derive new skill tree with own groups
    e.g. users in weather/climate, single program, testers
  - Realized via JavaScript (and JSON config files)

Outline

Tuning of Software Configurations

Goals

- Tune typically used software packages in Tier 3 centers
  - Explore best high-level configuration
    - Examples: Compiler flags, libraries
  - Adjusting runtime settings
    - Examples: $TMPDIR, process placement, thread number

Approach and Tasks

1. Determination of tuning possibility (from literature)
2. Setup of realistic use cases (cooperation with users)
3. Benchmarking (with use cases)
4. Documentation (success stories)

Status

Use-cases executed cross all tasks

- Several use-cases for the statistical tool R
  - Optimizing compiler options measured with R-benchmark
  - Parallelization of rlassoEffects-regression function
  - Parallelization of satellite image analysis

Tasks

1. Determination of tuning possibility (ongoing)
2. Setup of realistic use cases (ongoing)
3. Benchmarking (ongoing)
4. Documentation (ongoing)

Findings

Generic

- Use OpenBLAS or MKL (minimal better than OpenBLAS)
- -O3 already delivered best performance (PGO: no benefit)
- Use at least simple parallelization via `foreach()`

- Use case A: "R Benchmark 2.5" (Simon Urbanek)
  - Mix of matrix operations (cross-product, eigenvalues) and algorithmic parts (recursion, loops)
  - Speedup: ca. 4 using MKL
  - Hardly any additional speedup by parallelization via OMP_NUM_THREADS (only ca. 15%)

Findings

- Use case B: Parallelization of the rlassoEffects-function (regression analysis)
  - Speedup (reasonable problem size): ca. 30 using 64 cores (4 nodes / 16 cores each)
- Use case C: Analyzing satellite night images
  - Support user to parallelize the program using `foreach()` (co-development)
  - Speedup: ca. 126 using 128 cores (32 nodes / 4 cores)

Outline

Dissemination

Goals

- Establishing the Hamburg HPC Competence Center
- Collection of success stories (to motivate users)
- Creating a knowledge base
  - A "Google" for linking to trustworthy data center material

Tasks

1. Webpage
2. Success stories
3. Knowledge base

Status

Tasks

1. Webpage (done)
   - HHCC webpage is integrated into University CMS `https://www.hhcc.uni-hamburg.de/`
2. Success stories (ongoing)
   - Started a repository on the web page
3. Knowledge base (ongoing)
   - Student machine learning project crawling data
   - Explored ChatBot feature as alternative "search"

Activities

- Several meetings with ProfiT-HPC at DKRZ
- Discussion with ProPE team about certification program
- Handout at SC17 (November 2017)
- Handout at ISC 2017
- Several meetings/vid.call of the HPC certification forum
  `https://www.hpc-certification.org`
- Project posters at ISC-HPC 2017, ISC-HPC 2018
- Talk “Towards an HPC Certification Program” at SC 2018
  Workshop on Best Practices for HPC Training and Education
- See our annual Report D4.1 for more details

Outline

Summary

PeCoH

- brings Hamburg data centers closer together
- researches new strategies
  - Understanding cost-efficiency as feedback mechanism
  - Managing competences (HPC Certification program!)
  - Easing navigation of knowledge
- applies established techniques
  - Estimating and exploring emerging concepts benefit
  - Collecting / utilizing best-practises
  - Tuning of software packages

Backup

Example: Agile Software Development - Motivation

Current situation in scientific software development

- Scientific software often lacks quality
- SE best practices rarely adopted
- Mindset: conduct science, do not invest in development
- There is a lack of knowledge about requirements and testing principles[3]

[3] *D.F. Kelly:* A Software Chasm: Software Engineering and Scientific Computing

Storage Cost Models

- Storage costs (idle data) cannot be accounted for jobs
- Disk storage and tape archives must be distinguished
- Feedback on disk storage should be continuous:
  "Your X GiB of data on disk cost roughly Y € per month"
- Feedback on archive storage should be immediate:
  "Archiving this data for 10 years will cost about Z €"

Content production workflow (simplified)

- Markdown
  - Easy to use lightweight markup language, widely used for documentation purposes (e.g. on GitHub)
  - Plain text editor is sufficient
  - Supports formulas, syntax-highlighting of source code, tables, hyperlinks, including of images, ...
  - Content of each single skill is based on a set of Markdown-files
- Pandoc-Tool
  - Converts a variety of markup formats ("swiss-army knife")
  - Used to convert .md-skill content files to .html, .pdf, .tex
- Extensible Stylesheet Language Transformations (XSLT)
  - Transforms XML files to other formats
  - XSLT-Programs are used to generate Makefiles with corresponding Pandoc calls based on the skill tree data