PERMON in material engineering applications

Marek Pecha, Jiří Tomčala, David Horák, Martin Čermák, Jakub Kružík, Radim Sojka

October 31, 2017

permon.it4i.cz   am.vsb.cz   IT4Innovations national supercomputing center it4i.cz

Outline

- Artificial intelligence in Industry 4.0
  Predicting quality of steel for low temperature applications
- Fatigue simulations on HPC systems
  PragTic software parallelization

Industry 4.0?

Industry 4.0 ?= 4th industrial revolution ?= Work 4.0

- Digitalization of manufacturing processes
- Simulations of processes
- Autoconfiguration and autodiagnosis of systems

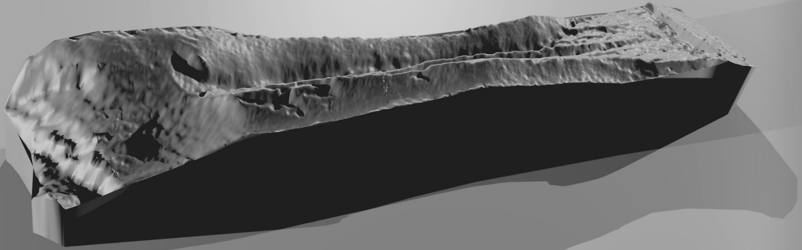
Concept

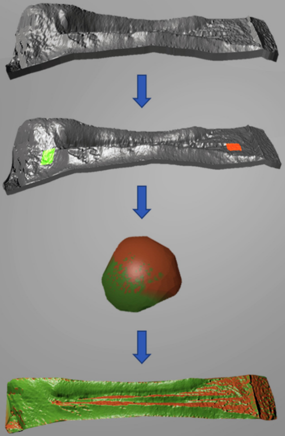
- Cyber physical system
- Internet of things
- Internet of services
- Digital economy

PERMON

ground truth supervised learning

detections of brittle and ductile fractures





the material engineering application employs

PermonSVM machine learning tool

achieved accuracy 93.25%

Predicting quality of steel for low temperature applications

Eastern Siberia - Pacific Ocean oil pipeline.

Predicting quality of steel for low temperature applications

- Drop weight-tear test
- Developed in the early 1960s (Battelle Memorial Institute).
- Tests characterisations of materials.
- Aimed at avoiding brittle fractures.

Predicting quality of steel for low temperature applications

How to evaluate the test?

- Expert analysis

- Artificial intelligence analysis

Predicting quality of steel for low temperature applications

Workflow (learning)

- IO operations
  - Point cloud from LIMESS scan
  - Point cloud from high resolution depth maps - in development for RT
- 3D surface reconstruction
- Determination characteristic (human input)
- Learning
  - Supervized learning: Ground truth learning from expert analysis - SVM
  - Unsupervized learning: PAM, PAM++, start-fix (own algorithm), other Lloyd type algorithms, EM (in development), spectral method (in development)
  - Evaluation of results (partially done)
- Building model (SVM, TensorFlow) - in development

Predicting quality of steel for low temperature applications

- Vision and learning parallelization:
  - MPI
  - OpenMP
  - MPI + OpenMP (in development)
  - OpenCL and/or CUDA for computer vision in RT (in development)

- Libraries:
  - Computer vision: VTK, OpenCV
  - Learning libraries: PermonSVM (+PermonQP), PermonLloyd
  - Support libraries: PETSc, Boost

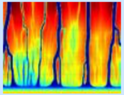
PERMON software on GitHub

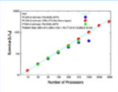
GitHub Downloads (PermonQP & PermonSVM)

- http://github.com/it4innovations/permon
- http://github.com/it4innovations/permonsvm

PERMON software on PETSc webpage

PETSc/Tao





Portable, Extensible Toolkit for Scientific Computation

- **Home**
- **Annual Meetings**
- **Download**
- **Features**
- **Documentation**
  - **Manual pages and Users Manual**
  - **Citing PETSc**
  - **Tutorials**
  - **Installation**
  - **SAWs**
  - **Changes**
  - **BugReporting**
  - **CodeManagement**
  - **FAQ**
  - **License**
- **Applications/Publications**
- **Miscellaneous**
- **External Software**
- **Developers Site**

Toggle Search

The current version of PETSc is 3.8; released September 26, 2017.

PETSc, **pronounced PET-see** (the S is silent), is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It supports MPI, and GPUs through CUDA or OpenCL, as well as hybrid MPI-GPU parallelism. PETSc (sometimes called PETSc/Tao) also contains the Tao optimization software library.

- Scientific applications that use PETSc
- Features of the PETSc libraries (and a recent podcast)
- Linear system solvers accessible from PETSc
- Related packages that use PETSc
  - PermonSVM support vector machines and PermonQP quadratic programming
  - MOOSE - Multiphysics Object-Oriented Simulation Environment finite element framework, built on top of libMesh and PETSc
  - SLEPc - Scalable Library for Eigenvalue Problems
  - COOLFluiD - CFD, plasma and multi-physics simulation package
  - Fluidity - a finite element/volume fluids code
  - OpenFVM - finite volume based CFD solver
  - OOFEM - object oriented finite element library
  - libMesh - adaptive finite element library
  - FEniCS - sophisticated Python based finite element simulation package
  - Firedrake - sophisticated Python based finite element simulation package
  - DEAL.II - sophisticated C++ based finite element simulation package
  - PHAML - The Parallel Hierarchical Adaptive MultiLevel Project
  - Chaste - Cancer, Heart and Soft Tissue Environment
  - PyClaw - A massively parallel, high order accurate, hyperbolic PDE solver
  - PetIGA - A framework for high performance Isogeometric Analysis

PragTic software parallelization

Fatigue simulations on HPC systems

- Mechanical fatigue is caused by repeated service loading
- It leads to the reduction of its load-carrying capacity and then to the fatigue failure
- Experimental verification is very expensive and time consuming

Fatigue simulations on HPC systems

The Fatigue Damage Software (PragTic)

- Developed at CTU in Prague by Jan Papuga et al.
- Predicts mechanical fatigue failure by generating large - scale computational simulations
- Major feature is the multiaxial fatigue analysis
- Sequence implementation

Fatigue simulations on HPC systems

From desktops to HPC

- Compile 20 year old source code on HPC system
- Task: optimization & parallelization

Fatigue simulations on HPC systems

Code optimization

- I/O elimination
  it caused that data stays just in the memory
- replacing the old temporary buffers class data_vector with standard C++ std::vector class
- required changes to hundreds of lines throughout the whole PragTic source code
- **maximum observed speedup: 60**

Fatigue simulations on HPC systems

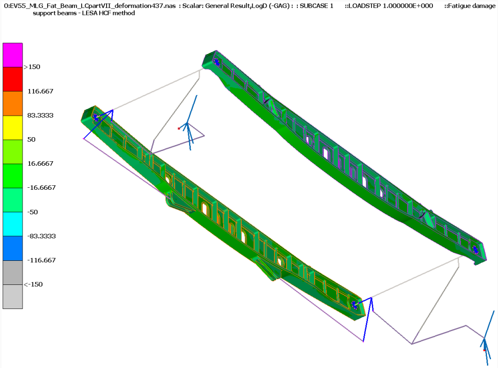
Parallelization

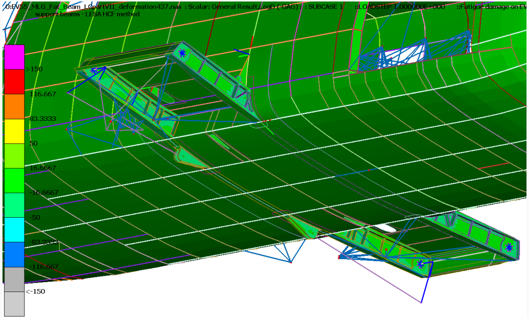
- parallelization using MPI technology
- in the preprocessing the master process distributes a particularsubset of nodes to every process
- post-processing merges partial results into one result file
- **maximum observed speedup: 36**

Maximum observed optimization & parallelization speedup: 150.

Fatigue simulations on HPC systems

Fatigue simulations on HPC systems

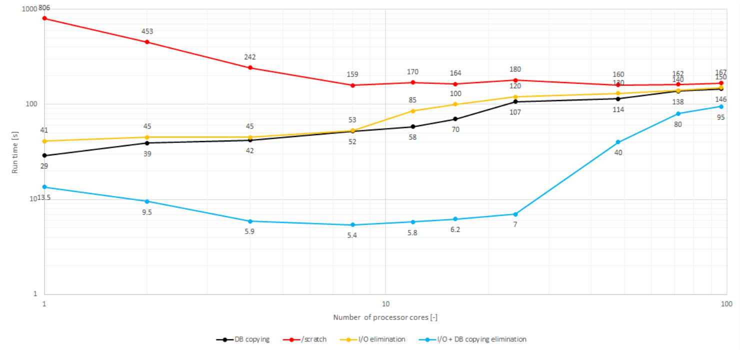




EV-55 gear landing

Fatigue simulations on HPC systems

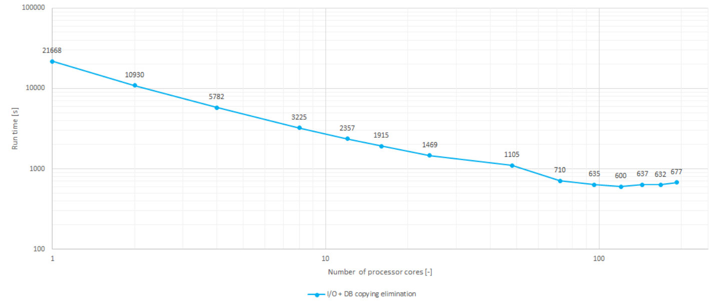
Run times for the EV-55 airplane landing gear example (239,628 nodes). Simulated mechanical loading: taxiing on the runway $(1 + 0.05 * \sin(t))$. Method: Bergmann.



Maximum overall speedup: 150.

Fatigue simulations on HPC systems

Run times for the EV-55 airplane landing gear example (239,628 nodes). Simulated mechanical loading: taxiing on the runway (1 + 0.05*sin(t)). Method: Wang & Brown.



Maximum speedup: 36.

Thank you for your patience and attention

Visit us on **permon.it4i.cz.**