Reproducible Stencil Compiler Benchmarks Using PROVA!

Danilo Guerrera, Helmar Burkhart, Antonio Maffia

University of Basel
danilo.guerrera@unibas.ch

PMMA 16

Thu, Jun 23, 2016
Overview

1. Reproducibility in Science
   - Problem, System, Method
   - Reproducibility Levels

2. PROVA!
   - Current Version
   - Environment Management

3. Use Cases
   - Wave
   - Molecular Dynamics

4. Conclusions
Reproducibility - A Science Principle

“Non-reproducible single occurrences are of no significance to science.” (Karl Popper The Logic of Scientific Discovery 1934/1959)
Goals of the PROVA! Project

*try, prove, convince*

- Taxonomy for Reproducibility Levels
Goals of the PROVA! Project

*try, prove, convince*

- Taxonomy for Reproducibility Levels
- Performance Engineering Support
Goals of the **PROVA!** Project

*try, prove, convince*

- Taxonomy for Reproducibility Levels
- Performance Engineering Support
- Best Practice Demonstrator
Goals of the PROVA! Project

try, prove, convince

- Taxonomy for Reproducibility Levels
- Performance Engineering Support
- Best Practice Demonstrator
- Modern HPC Education
Goals of the PROVA! Project

*try, prove, convince*

---

**STENCIL PROBLEM REPOSITORY**

**RE-EXPERIMENT REPPLICATE REPEAT**

**EXPERIMENT WORKSPACE & TOOLS**

**PERFORMANCE MODEL**

**REPRODUCIBILITY PORTABILITY COMPLETENESS**

**PARALLEL SYSTEM 1**

---

1. STENCIL PROBLEM REPOSITORY
2. RE-EXPERIMENT REPPLICATE REPEAT
3. PERFORMANCE REPOSITORY
4. PARALLEL SYSTEM 1
5. PERFORMANCE MODEL
6. ANALYZE INTERPRETE

---

**LAWS PREDICTION SCALABILITY**
Problem / Method / System

A computational problem is solved by an algorithmic method on a compute system.
Problem / Method / System

A computational problem is solved by an algorithmic method on a compute system.

- **Problem**: specification of the problem including characteristic parameters.
Problem / Method / System

A computational problem is solved by an algorithmic method on a compute system.

- **Problem**: specification of the problem including characteristic parameters.
- **Method**: description of the algorithmic approach used to tackle the problem.
A computational problem is solved by an algorithmic method on a compute system.

- **Problem**: specification of the problem including characteristic parameters.
- **Method**: description of the algorithmic approach used to tackle the problem.
- **System**: representation of the compute environment (both hardware and software), on which an experiment is run.
Micro- and Macro-Experiments

System

Method

micro-experiment 1

Problem

Sys1

Met1

Pro1
Micro- and Macro-Experiments

- Sys1
- Met1
- Met2
- Met3
- Problem
- Pro1
- micro-experiment 1
- micro-experiment 2
- micro-experiment 3
- macro-experiment
Reproducibility Levels

- **Repetition**: re-running the original *micro* - or *macro-experiment* without any variation of the parameters, should drive to the same results and a certain level of credibility is guaranteed (completeness of documentation)
Reproducibility Levels

- **Repetition**: re-running the original *micro- or macro-experiment* without any variation of the parameters, should drive to the same results and a certain level of credibility is guaranteed (*completeness* of documentation)

- **Replication**: is related to the system hosting an experiment. An experiment should not be bound to a specific compute environment (*portability*)
Reproducibility Levels

- **Repetition**: re-running the original *micro-* or *macro-experiment* without any variation of the parameters, should drive to the same results and a certain level of credibility is guaranteed (*completeness* of documentation)

- **Replication**: is related to the system hosting an experiment. An experiment should not be bound to a specific compute environment (*portability*)

- **Re-experimentation**: if changing the methods drives to the same outputs, the scientific approach is proven (*correctness* of the approach)
Functionalities Needed - Support Given

- Collaboration Support: git
- Software Management: EasyBuild, LMod
- Experiment Reproduction
- Experiment Portability
- Performance Modeling Support
- Visualization
PROVA! - Current Version of the Architecture
Lmod modules

- developed at TACC\(^1\)
- user’s environment can be changed dynamically through modulefiles
- manages the PATH
- a modulefile contains information on how to run a particular application or provide access to a particular library

\(^1\)https://www.tacc.utexas.edu/research-development/tacc-projects/lmod
Scientific Software Management and Build Via EasyBuild²

- a flexible framework for building/installing (scientific) software
- fully automates software builds
- keeps track of the versions
- consistent software stack
- allows for easily reproducing previous builds
- keep the software build recipes/specifications simple and human-readable
- supports co-existence of versions/builds via dedicated installation prefix and module files
- enables sharing with the HPC community

²http://hpcugent.github.io/easybuild/
Lmod + EasyBuild

- Easily install new software as module
- Clean environment for all of the users
- Keep track of the software installed
- Possibility to attach to an experiment a subset of modules
  - Export source code + environment
Web Application

Remote Environment 1

- Front-end machine
- Framework
- Scheduler
- Parallel machine
- Cluster
- Workspaces
  - Workspace Scientist1
  - Workspace Scientist2
- File Storage
- NFS

Use Cases

- Web Application
- Remote Environment 1
- Cluster
- File Storage
- Parallel machine
- Workspaces
  - Workspace Scientist1
  - Workspace Scientist2

Conclusions
# Simple 3D Wave

**University of Basel 2016**

<table>
<thead>
<tr>
<th><strong>Problem</strong></th>
<th>Calculate a 3-D wave equation of $200^3$ elements (IEEE single precision arithmetic) in 100 timesteps</th>
</tr>
</thead>
</table>
| **System**  | **SW**: OpenMP 4.0, GCC 4.9.2, PATUS 0.1.4, PLUTO 0.10  
**HW**: 1 node  
- CPU: 2x AMD Opteron 6274 "Bulldozer" 16-Core, 2.2 GHz, 12 MiB L3 cache, 4 NUMA domains  
- RAM: 256 GiB  
- OS: Ubuntu 14.04.4, Kernel 3.8.0-38 |
| **Method**  | 1. Naive OpenMP implementation with NUMA aware initialization (16 FLOPS)  
2. DSL + auto-tuning with PATUS (20 FLOPS)  
3. Polyhedral model with PLUTO (16 FLOPS) |
# Simple 3D Wave

**University of Erlangen 2016**

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate a 3-D wave equation of $200^3$ elements (IEEE single precision arithmetic) in 100 timesteps</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SW</strong>: OpenMP 4.0, GCC 4.9.2, PATUS 0.1.4, PLUTO 0.10</td>
</tr>
<tr>
<td><strong>HW</strong>: 1 node</td>
</tr>
<tr>
<td>• CPU: 2x Xeon 5650 &quot;Westmere&quot; 6 cores + SMT, 2.66 GHz, 12 MiB Shared Cache per chip, 2 NUMA domains</td>
</tr>
<tr>
<td>• RAM: 24 GB (DDR3-1333)</td>
</tr>
<tr>
<td>• OS: CentOS 6.7, Kernel 2.6.32-573.7.1.el6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Naive OpenMP implementation with NUMA aware initialization (16 FLOPS)</td>
</tr>
<tr>
<td>2. DSL + auto-tuning with PATUS (20 FLOPS)</td>
</tr>
<tr>
<td>3. Polyhedral model with PLUTO (16 FLOPS)</td>
</tr>
</tbody>
</table>
Simple 3D Wave (2)

Performance Comparison of Project: Wave3D
Parameters (X_MAX Y_MAX Z_MAX): 200 200 200

Implemented Method
- openMP
- patus
- pluto

GFlop/s vs Number of Threads
Simple 3D Wave (3)

Performance Comparison of Project: likwid_comparison
Parameters (X_MAX Y_MAX Z_MAX): 200 200 200

Implemented Method
omp
omp_node
Demo

https://repro-hpc.dmi.unibas.ch
PROVA!: Adaptation Effort

- How to create a methodType?
- What is the effort?
  - What knowledge is needed?
  - How much time to invest in it?
PROVA!: Adaptation Effort - Molecular Dynamics

- Create a method descriptor
PROVA!: Adaptation Effort - Molecular Dynamics

1. Create a method descriptor

```json
{
    "name": "GROMACS-5.0.5",
    "eb_modules": [
        "GROMACS/5.0.5-foss-2016a-hybrid-noGPU"
    ],
    "version": "5.0.5",
    "comment": "GROMACS, a molecular dynamics package primarily designed for biomolecular systems such as proteins and lipids, based on the foss-2016a toolchain, compiled with hybrid OpenMP and Open MPI, without CUDA support"
}
```
PROVA!: Adaptation Effort - Molecular Dynamics

- Create a method descriptor
- Create (if not existing) the easyconfigs for the needed modules
PROVA!: Adaptation Effort - Molecular Dynamics

- Create a method descriptor
- Create (if not existing) the easyconfigs for the needed modules

```python
name = 'GROMACS'
version = '5.0.5'
versionsuffix = '−hybrid−noGPU'

homepage = 'http://www.gromacs.org'

description = '''GROMACS is a versatile package to perform molecular
dynamics, i.e. simulate the Newtonian equations of motion for systems
with hundreds to millions of particles.'''

toolchain = {'name': 'foss', 'version': '2016a'}
toolchainopts = {'openmp': True, 'usempi': True}

sources = [SOURCELOWER_TAR_GZ]

builddependencies = [
    ('CMake', '3.4.3'),
    ('libxml2', '2.9.2')
]

dependencies = [(
    'Boost', '1.59.0', '−Python−2.7.11'))
# explicitly disable CUDA support
configopts = '−DGMX_GPU=OFF'
moduleclass = 'bio'
```
PROVA!: Adaptation Effort - Molecular Dynamics

- Create a method descriptor
- Create (if not existing) the easyconfigs for the needed modules
- Create compile (Makefile) and run scripts
PROVA!: Adaptation Effort - Molecular Dynamics

- Create a method descriptor
- Create (if not existing) the easyconfigs for the needed modules
- Create compile (Makefile) and run scripts
- Install it in the tool
PROVA!: Adaptation Effort - Molecular Dynamics

- Create a method descriptor
- Create (if not existing) the easyconfigs for the needed modules
- Create compile (Makefile) and run scripts
- Install it in the tool
- Create a project and use it! ³

³Thanks to Florent Hedin http://www.chemie.unibas.ch/~hedin/
### PROVA!: Adaptation Effort - Molecular Dynamics

#### REAL CYCLE AND TIME ACCOUNTING

On 2 MPI ranks, each using 8 OpenMP threads

<table>
<thead>
<tr>
<th>Computing</th>
<th>Num Ranks</th>
<th>Num Threads</th>
<th>Call Count</th>
<th>Wall time (s)</th>
<th>Giga–Cycles total sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain decomp.</td>
<td>2</td>
<td>8</td>
<td>960</td>
<td>7.755</td>
<td>272.988 2.2</td>
</tr>
<tr>
<td>DD comm. load</td>
<td>2</td>
<td>8</td>
<td>39</td>
<td>0.001</td>
<td>0.019 0.0</td>
</tr>
<tr>
<td>Neighbor search</td>
<td>2</td>
<td>8</td>
<td>961</td>
<td>10.492</td>
<td>369.350 2.9</td>
</tr>
<tr>
<td>Comm. coord.</td>
<td>2</td>
<td>8</td>
<td>18240</td>
<td>2.674</td>
<td>94.139 0.7</td>
</tr>
<tr>
<td>Force</td>
<td>2</td>
<td>8</td>
<td>19201</td>
<td>229.409</td>
<td>8075.973 64.3</td>
</tr>
<tr>
<td>Wait + Comm. F</td>
<td>2</td>
<td>8</td>
<td>19201</td>
<td>3.283</td>
<td>115.568 0.9</td>
</tr>
<tr>
<td>PME mesh</td>
<td>2</td>
<td>8</td>
<td>19201</td>
<td>67.117</td>
<td>2362.756 18.8</td>
</tr>
<tr>
<td>NB X/F buffer ops.</td>
<td>2</td>
<td>8</td>
<td>55681</td>
<td>8.287</td>
<td>291.731 2.3</td>
</tr>
<tr>
<td>Write traj.</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>0.029</td>
<td>1.035 0.0</td>
</tr>
<tr>
<td>Update</td>
<td>2</td>
<td>8</td>
<td>19201</td>
<td>4.215</td>
<td>148.368 1.2</td>
</tr>
<tr>
<td>Constraints</td>
<td>2</td>
<td>8</td>
<td>19201</td>
<td>18.922</td>
<td>666.110 5.3</td>
</tr>
<tr>
<td>Comm. energies</td>
<td>2</td>
<td>8</td>
<td>961</td>
<td>0.039</td>
<td>1.372 0.0</td>
</tr>
<tr>
<td>Rest</td>
<td></td>
<td></td>
<td></td>
<td>4.639</td>
<td>163.315 1.3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>356.862</strong></td>
<td><strong>12562.724 100.0</strong></td>
</tr>
</tbody>
</table>
# PROVA!: Adaptation Effort - Molecular Dynamics

## Breakdown of PME mesh computation

<table>
<thead>
<tr>
<th>Task</th>
<th>Core t (s)</th>
<th>Wall t (s)</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PME redist. X/F</td>
<td>2</td>
<td>8</td>
<td>38402</td>
</tr>
<tr>
<td>PME spread/gather</td>
<td>2</td>
<td>8</td>
<td>38402</td>
</tr>
<tr>
<td>PME 3D–FFT</td>
<td>2</td>
<td>8</td>
<td>38402</td>
</tr>
<tr>
<td>PME 3D–FFT Comm.</td>
<td>2</td>
<td>8</td>
<td>38402</td>
</tr>
<tr>
<td>PME solve Elec</td>
<td>2</td>
<td>8</td>
<td>19201</td>
</tr>
</tbody>
</table>

| Time:                  | 5701.124   | 356.862    | 1597.6 |
| (ns/day)               | (hour/ns)  |            |        |
| Performance:           | 9.298      | 2.581      |        |
| Finished mdrun on rank 0 Sat Jun 18 11:53:30 2016 |
Conclusions

- Reproducibility needs to be emphasized in the performance modeling.
- Repeatability of an experiment only possible if precise description of experiment is given: Problem, System, and Method.
- Repeatability: World-wide access to experiments through Internet feasible (security and authentication mechanisms essential).
- Replication and re-experimentation: harder to achieve but not impossible.
Future Work

Short term:
- Jobs: no clue about when the job finishes its execution
- Homogeneity of nodes: libraries and sw are installed on a shared NFS so all the nodes must be homogeneous to run such sw
- Experiment is run as a block: bad resource usage
- Installation of the modules is simply delegated to EasyBuild
- Visualization is not so powerful

Mid term:
- Provenance of the experiments
- Collaborative Performance Engineering
- Integrate Performance Models to Evaluate Performance Outputs
- Towards a Science Gateway
Interested?

https://prova.io