An Effective Methodology for Reproducible Research on Dynamic Task-Based Runtime Systems

Luka Stanisic and Arnaud Legrand  
luka.stanisic@inria.fr

Inria, Bordeaux Sud-Ouest, France  
CNRS, Grenoble, France

SIAM Conference on Parallel Processing for Scientific Computing  
Paris, April 12, 2016
Challenges of Experimental Studies in HPC

What your research supposedly looks like:

Data → Data Acquisition → Prototype → Controller → Computer

What your research actually looks like:

Figure 1. Experimental Diagram

Figure 2. Experimental Mess
Challenges of Experimental Studies in HPC

- Large, hybrid, prototype hardware/software (hard to control)
- Costly experiments with numerous parameters
- Non-deterministic executions (makespans, traces, ...)
- Workflows specific to the studies (hardly applicable in general)

→ difficult to make research results reproducible
Reproducibility in HPC Community

Reproducing results of others is not in the culture

- Novelty is more rewarded than consolidating existing results
- Hardware and software evolve very quickly
- Reproducible research is not mandatory

Why bother?
Reproducibility in HPC Community

Reproducing results of others is not in the culture

- Novelty is more rewarded than consolidating existing results
- Hardware and software evolve very quickly
- Reproducible research is not mandatory

Why bother?

Motivation for reproducible research

- Model \( \neq \) Reality (mistakes are easily done)
- Compare with others (or even with your previous solutions)
- Build on previous work
1934: Karl Popper introduces the notion of falsifiability and crucial experiment and puts reproducing the work of others at the core of science.

Reproducibility of experimental results is the hallmark of science.

[Drummond, 2009]
1934: Karl Popper introduces the notion of falsifiability and crucial experiment and puts reproducing the work of others at the core of science

Reproducibility of experimental results is the hallmark of science

[Drummond, 2009]

⚠️ Terminology varies ⚠️

Repetition, replication, variation, reproduction, corroboration [Feitelson, 2015]

Replicability

Precisely replicate exactly what someone else has done, recreating their artifacts

(same results)

Reproducibility

Recreate the spirit of what someone else has done, using your own artifacts

(same scientific conclusions)

Inspired by Andrew Davison (AMP Workshop on Reproducible research)
Inspired by Roger D. Peng’s lecture on reproducible research, May 2014
Inspired by Roger D. Peng’s lecture on reproducible research, May 2014.
Inspired by Roger D. Peng’s lecture on reproducible research, May 2014
Inspired by Roger D. Peng’s lecture on reproducible research, May 2014
"Tricky" and "Easy" refer to parallel computer scientist use cases.

Inspired by Roger D. Peng’s lecture on reproducible research, May 2014
1. Reproducible Study of Dynamic Task-based Application
2. Strengthening Reproducibility of Experiments with Simulation
3. Conclusion
Is it possible to perform a *clean, coherent, reproducible study* of HPC applications executed on top of dynamic task-based runtime systems?
qr_mumps (CNRS-IRIT Toulouse)

- **Tree parallelism**: nodes in separate branches can be treated independently
- **Node parallelism**: large nodes can be treated by multiple processes

StarPU (Inria Bordeaux)

- Dynamic runtime for hybrid architectures (CPU, GPU, MPI)
- Opportunistic scheduling of a task graph guided by performance models
- Features dense, sparse and FMM applications
Overview of Experimental Results

Riri machine with 10 cores

Riri machine with 40 cores

Native

State
- Do_subtree
- Activate
- Panel
- Update
- Assemble
- Deactivate
- Idle

CPU7
CPU6
CPU5
CPU4
CPU3
CPU2
CPU1
CPU0

Time [ms]

0 10,000 20,000 30,000 40,000 50,000

Makespan [s]

0 100 200 300 400 500

Type
- Native
Key Elements of Reproducible Research

- Publishing data/code/article
- Replicable article
- Logging activity
- Capturing meta-data

- Logging and backuping data
- Mastering environment
- Controlling experiments
- Organizing code and data

“Tricky” and “Easy” refer to parallel computer scientist use cases

http://starpu-simgrid.gforge.inria.fr/
Is it possible to perform a clean, coherent, reproducible study of HPC applications executed on top of dynamic task-based runtime systems?

- Yes, but reproducing experiment results and understanding performance remains tricky.
Outline

1. Reproducible Study of Dynamic Task-based Application

2. Strengthening Reproducibility of Experiments with Simulation

3. Conclusion
Alleviating the Tricky Part

Hard to reproduce Native experiments

Measured Data

Analytic Data

Computational Results

Figures

Tables

Published Article

Text

Scientific Question

Protocol (Design of Experiments)

Experiment Code (workload injector, VM recipes, ...)

Nature/System/...

Presentation Code

Analysis Code

Processing Code

Easy part

“Easy” part

Author

Reader
Alleviating the Tricky Part

Easy part

How this part can be improved?

Author

Measured Data

Analytic Data

Computational Results

Presentation Code

Figures

Tables

Numerical Summaries

Published Article

Text

Nature/System/...

Experiment Code
(workload injector, VM recipes, ...)

Processing Code

Analysis Code

Presentation Code

Scientific Question

Protocol
(Design of Experiments)
Possible Approaches

**Equations**
- Biased machine
- Biased code
- Quick trends but simplistic
- Replicable results

**Mini Application**
- Actual machine
- Biased code
- Discrete bias
- Non replicable performance
# Possible Approaches

## Equations
- Biased machine
- Biased code
- Quick trends but simplistic
- Replicable results

## Mini Application
- Actual machine
- Biased code
- Discrete bias
- Non replicable performance

## Simulation
- Biased machine (but configurable)
- Actual code
- Continuous bias (tunable accuracy)
- Replicable executions (performance, bugs)
Simulating Dynamic Task-based Application

**qr_mumps (CNRS-IRIT Toulouse)**

- Tree parallelism: nodes in separate branches can be treated independently
- Node parallelism: large nodes can be treated by multiple process

**StarPU (Inria Bordeaux)**

- Dynamic runtime for hybrid architectures (CPU, GPU, MPI)
- Opportunistic scheduling of a task graph guided by performance models
- Features dense, sparse and FMM applications
Simulating Dynamic Task-based Application

**qr_mumps (CNRS-IRIT Toulouse)**

- Tree parallelism: nodes in separate branches can be treated independently
- Node parallelism: large nodes can be treated by multiple process

**StarPU (Inria Bordeaux)**

- Dynamic runtime for hybrid architectures (CPU, GPU, MPI)
- Opportunistic scheduling of a task graph guided by performance models
- Features dense, sparse and FMM applications

**SimGrid (Inria Grenoble, Lyon, Nancy, ...)**

- Scalable simulation framework for distributed systems
- Tunable accuracy: actual execution of scheduler, tasks replaced by delays
Devised Workflow: StarPU + SimGrid

Calibration

Performance Profile

Run once!
Devised Workflow: **StarPU + SimGrid**

**Calibration**

- StarPU

**Performance Profile**

- Run once!

**Simulation**

- StarPU
- SimGrid

**Quickly Simulate Many Times**
Overview of Simulation Accuracy

Riri machine with 10 cores

Riri machine with 40 cores

Comparing traces

Native

SimGrid

State
- Do_subtree
- Activate
- Panel
- Update
- Assemble
- Deactivate
- Idle

CPU7
CPU6
CPU5
CPU4
CPU3
CPU2
CPU1
CPU0

CPU7
CPU6
CPU5
CPU4
CPU3
CPU2
CPU1
CPU0

Time [ms]
0
10,000
20,000
30,000
40,000
50,000
Is it possible to perform a clean, coherent, reproducible study of HPC applications executed on top of dynamic task-based runtime systems?

- Yes, but reproducing experiment results and understanding performance remains tricky.

- Task-based applications can be efficiently and accurately handled through simulation.
Outline

1. Reproducible Study of Dynamic Task-based Application

2. Strengthening Reproducibility of Experiments with Simulation

3. Conclusion
General methodology for **reproducible** experimental studies

Strengthened reproducibility with **simulation**

Task-based modularity allows for **fast** and **accurate** simulation

**Validated** approach on:
- Heterogeneous, distributed systems
- Chameleon, qr_mumps, ScalFMM, soon PasTiX

**Same** approach could be applicable to any task-based application

**Your code?**
Thank you!

More on reproducibility

- Anyone can check and try to reproduce this work:
  
  http://starpu-simgrid.gforge.inria.fr/

- Minimal example of our methodology:
  
  http://github.com/stanisic/RR_example/

- Ongoing series of webinars on reproducible research:
  
  http://github.com/alegrand/RR_webinars/