Effective Reproducible Research with Org-Mode and Git

Luka Stanisic and Arnaud Legrand
firstname.lastname@imag.fr

CNRS/Inria/Univ. of Grenoble, Grenoble, France

REPPAR, Porto
August 26, 2014
Experimenting in the Wild

Experiments in HPC: Even Worse!

- Using large, distributed, hybrid, prototype hardware/software
- Measuring execution times (makespans, traces, ...)
- More parameters, very costly and hard to reproduce
Reproducing in the Wild

In HPC Community

Reproducing results of others is not in the culture
- Novelty is more rewarded than consolidating existing results
- Excuse #1: Hardware and software evolve very quickly
- Excuse #2: Reproducible research is not mandatory

Why bother?

Motivation for Reproducible Research
- Mistakes are easily done (compiler, OS, . . .)
- Compare with others (or even with your previous solutions. . .)
- Build on others’ work
Reproducible Research: Challenging Issues

Experimental Machine Setup

1. Platform accessibility
2. Setting up environment
3. Conducting experiments

Often Neglected Aspects

1. Data and code accessibility
2. Provenance tracking
3. Documenting
4. Extendability
5. Replicable analysis

Several tools developed to ease specific experimental workflows

Our approach: use a lightweight combination of existing generic tools
Outline

1. Case Studies
2. Reproducible Research Workflow with Git (Live Demo)
3. Provenance Tracking with Org-mode (Live Demo)
4. Pros, Cons and Open Questions
Case Study #1: CPU Cache Performance

Studying CPU caches on various Intel and ARM micro-architectures

Issues

Platform related
- Non-standard OS, tools, ARM architectures → minimal dependencies
- Unstable environment setup → need to track many information
- Limited resources → no complex workflow

Application related
- Numerous, ever-growing number of input parameters
- Incrementally growing set of experimental results → organize, browse, compare
Case Study #2: StarPU+SimGrid

Simulating dynamic task-based runtime executed on hybrid (CPUs + GPUs) platforms

Issues

Platform related
- Prototype hardware with evolving libraries (CUDA, OpenCL, ...)
- Partial access to environment setup → need to track many information

Application related
- Complex evolving source codes → need to manage external repositories
- Incrementally growing set of experimental results → organize, browse, compare
Outline

2 Reproducible Research Workflow with Git (Live Demo)

3 Provenance Tracking with Org-mode (Live Demo)

4 Pros, Cons and Open Questions
Pros

- Fast and efficient for daily usage (loose safety belts: `git status`, `branch organization`, `labbook`, ...)
- No perfect replicability but good level of confidence in reproduction
- Not limited to specific use cases
- Good provenance tracking (easy to extend and explore)
- Writing reproducible articles becomes a natural step

StarPU+SimGrid Euro-Par Paper

Anyone can check and try to reproduce this work

http://dx.doi.org/10.6084/m9.figshare.928338
**Conclusion (2/2)**

**Cons**
- Git + Emacs + Org-mode + R + ... = steep learning curve
- Git is not designed for storing large data files
  - Sometimes crash during transfers
  - Checkouts are slow
  - Free public git servers are not meant for such workload
- Managing several external source repositories can be cumbersome

**Open Questions**
- Multiple developers and experimenters working in parallel
- Ideally our whole git repository would be public (e.g., github)

Does this really help external researchers?
article
analysis
data
experimentation
source code
2 Branches

```
src
```

```
data
```
Source Modifications

src data
New Branch for Experimentation

![Diagram showing connections between 'src', 'exp', and 'data']
Analyzing Results
New Experiment
Possible Conflicts

Cherry-Pick

Revert

Source modifications

src

data