Towards reproducibility of experiments

Lucas Nussbaum
lucas.nussbaum@loria.fr

Excerpts from two recent PhD theses done using Grid'5000:

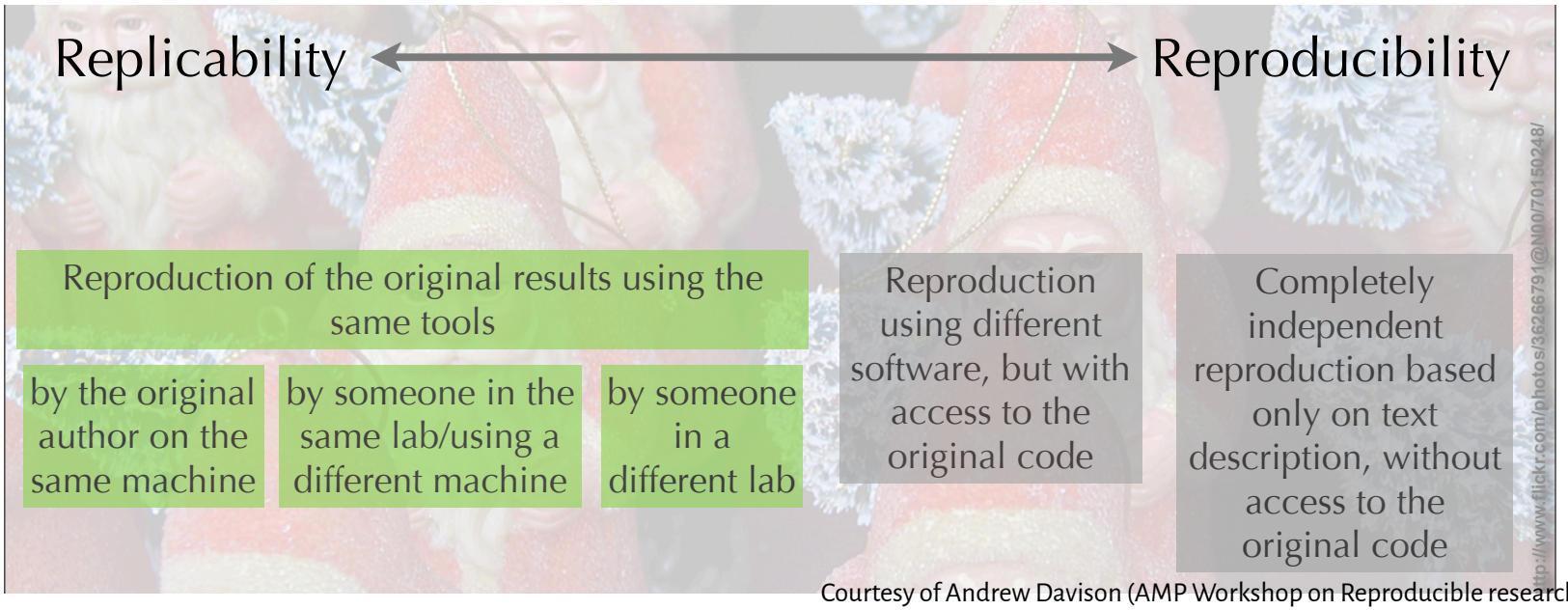
- Tomasz Buchert. *Managing large-scale, distributed systems research experiments with control-flows*. Directed by Lucas Nussbaum and Jens Gustedt. Defended on 2016-01-06.
- Cristian Ruiz. *Methods and Tools for Challenging Experiments on Grid'5000: a use case on electromagnetic hybrid simulation*. Directed by Olivier Richard and Thierry Monteil. Defended on 2014-12-15.

Introduction

- Great testbeds: Grid'5000, Chameleon, CloudLab, GENI, Fed4FIRE, etc.
- Unclear how to use them $\rightsquigarrow$ best practices?

Introduction

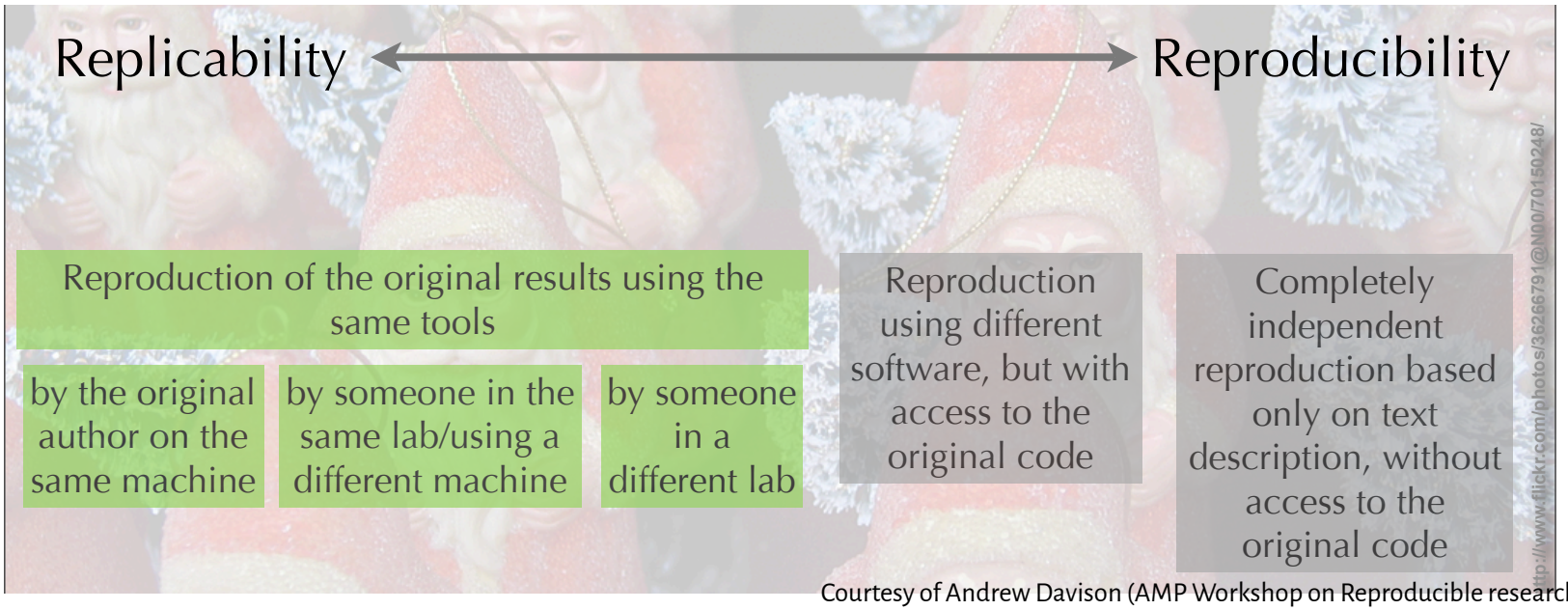
- Great testbeds: Grid'5000, Chameleon, CloudLab, GENI, Fed4FIRE, etc.
- Unclear how to use them ⇝ best practices?
- Ultimate goals:
  - Replicable experiments
  - Reproducible experiments



Courtesy of Andrew Davison (AMP Workshop on Reproducible research)

Introduction

- Great testbeds: Grid'5000, Chameleon, CloudLab, GENI, Fed4FIRE, etc.
- Unclear how to use them ⇝ best practices?
- Ultimate goals:
  - Replicable experiments ⇝ easy if we can fully automate
  - Reproducible experiments ⇝ extended description & provenance



Courtesy of Andrew Davison (AMP Workshop on Reproducible research)

Introduction

- Great testbeds: Grid'5000, Chameleon, CloudLab, GENI, Fed4FIRE, etc.
- Unclear how to use them ⇝ best practices?
- Ultimate goals:
  - Replicable experiments ⇝ easy if we can fully automate
  - Reproducible experiments ⇝ extended description & provenance
- This talk: an overview of two recent & complementary PhD theses
  - Tomasz Buchert. *Managing large-scale, distributed systems research experiments with control-flows*. Directed by Lucas Nussbaum and Jens Gustedt. 2016-01-06.

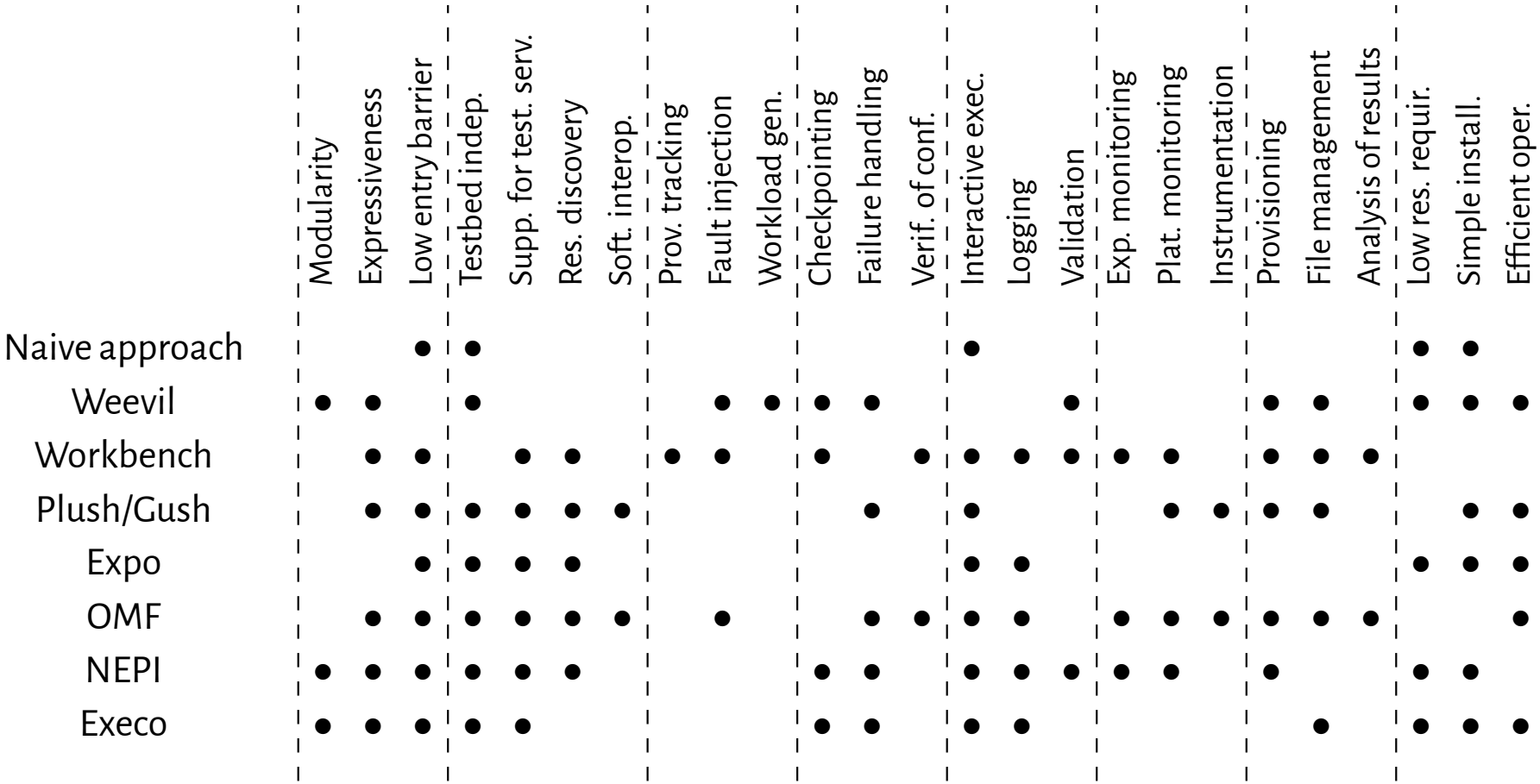
    ⇝ **XPFlow: Experiment management using workflows**
  - Cristian Ruiz. *Methods and Tools for Challenging Experiments on Grid'5000: a use case on electromagnetic hybrid simulation*. Directed by Olivier Richard and Thierry Monteil. 2014-12-15.

    ⇝ **Kameleon: reconstructable software appliances**

XPFlow: Experiment management using workflows

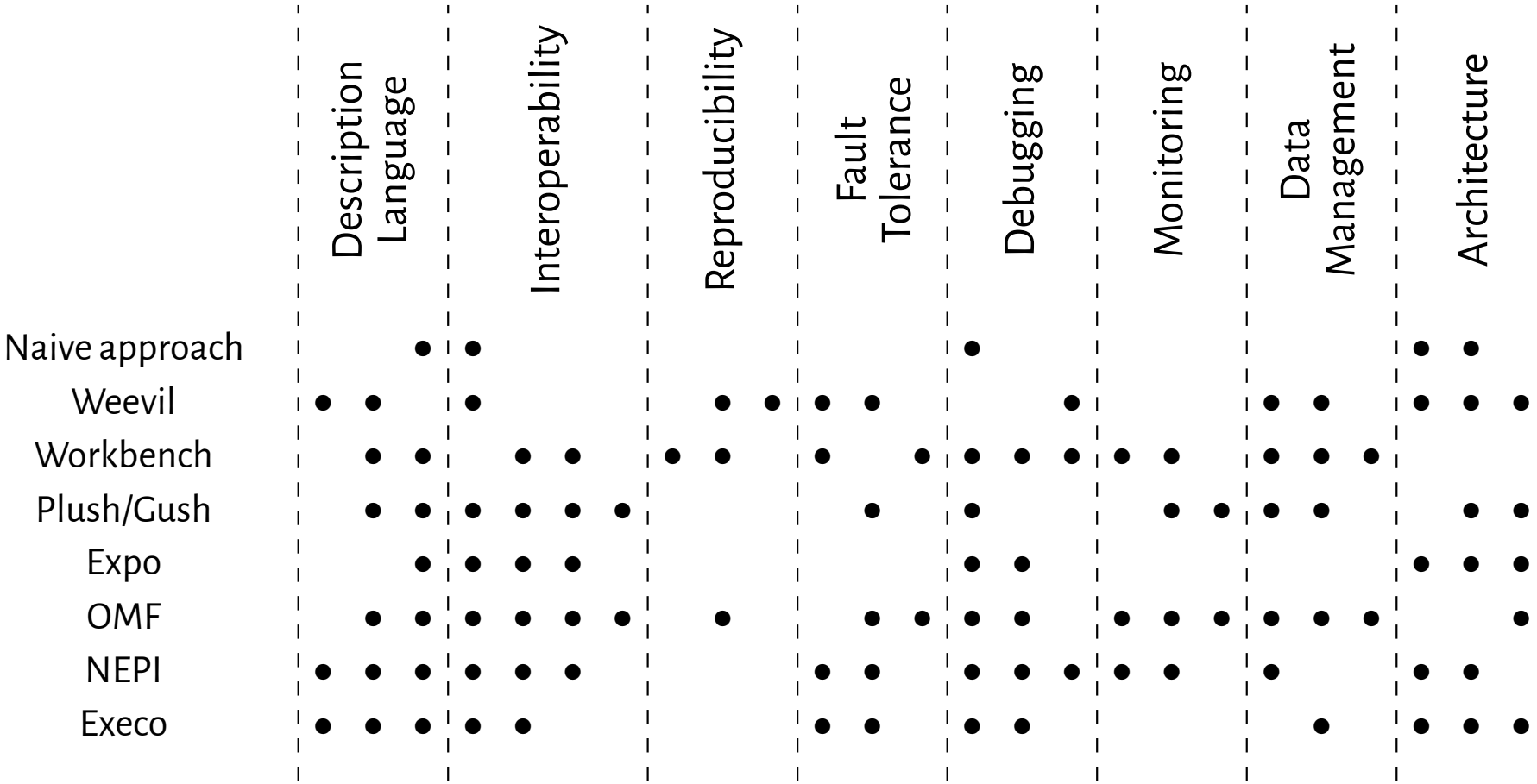
(Tomasz Buchert. *Managing large-scale, distributed systems research experiments with control-flows*)

Survey of experiment management tools[1]



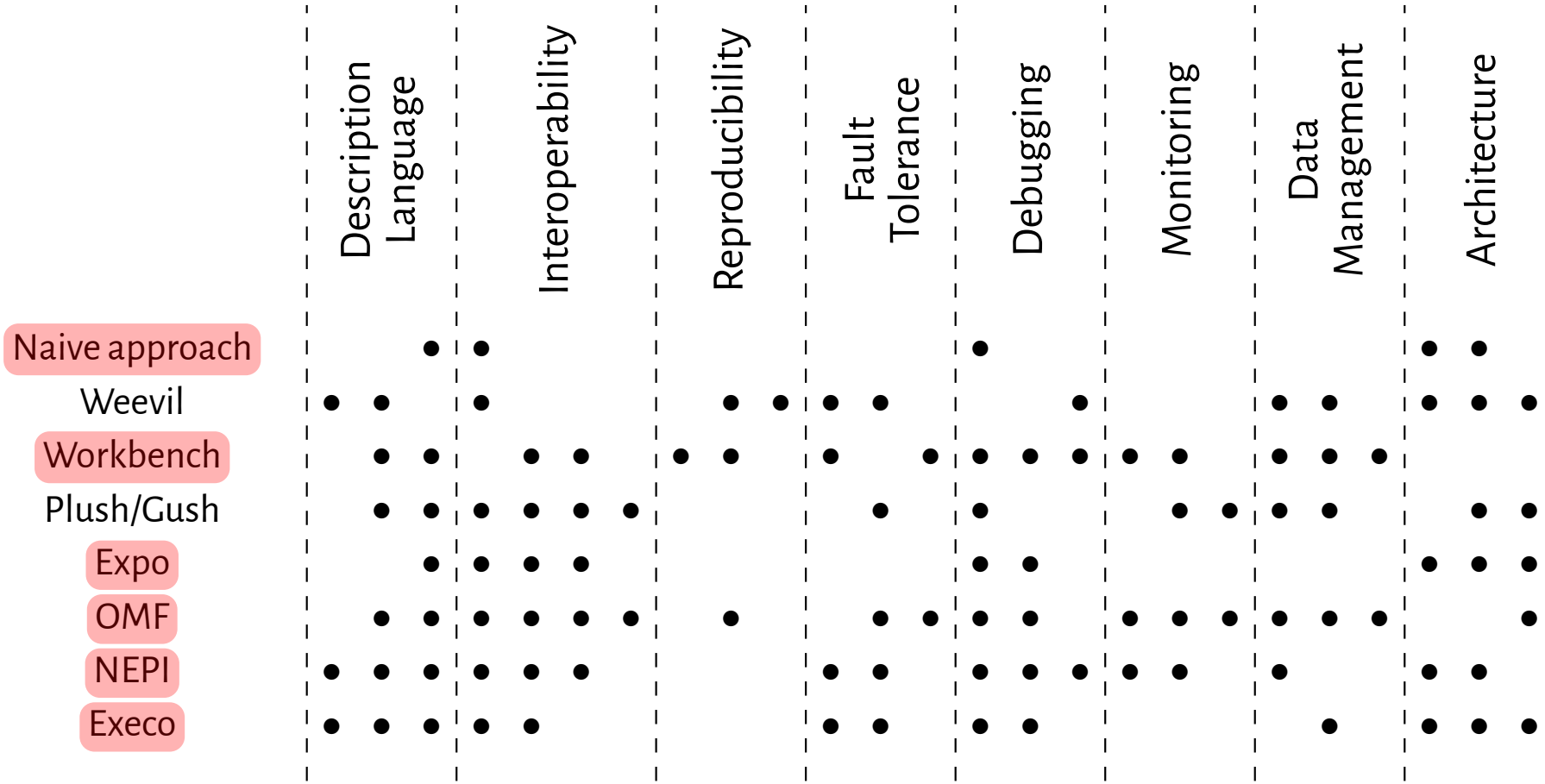
[1] Tomasz Buchert et al. "A survey of general-purpose experiment management tools for distributed systems". In: *Future Generation Computer Systems* 45 (2015), pp. 1–12.

Survey of experiment management tools[1]



[1] Tomasz Buchert et al. "A survey of general-purpose experiment management tools for distributed systems". In: *Future Generation Computer Systems* 45 (2015), pp. 1–12.

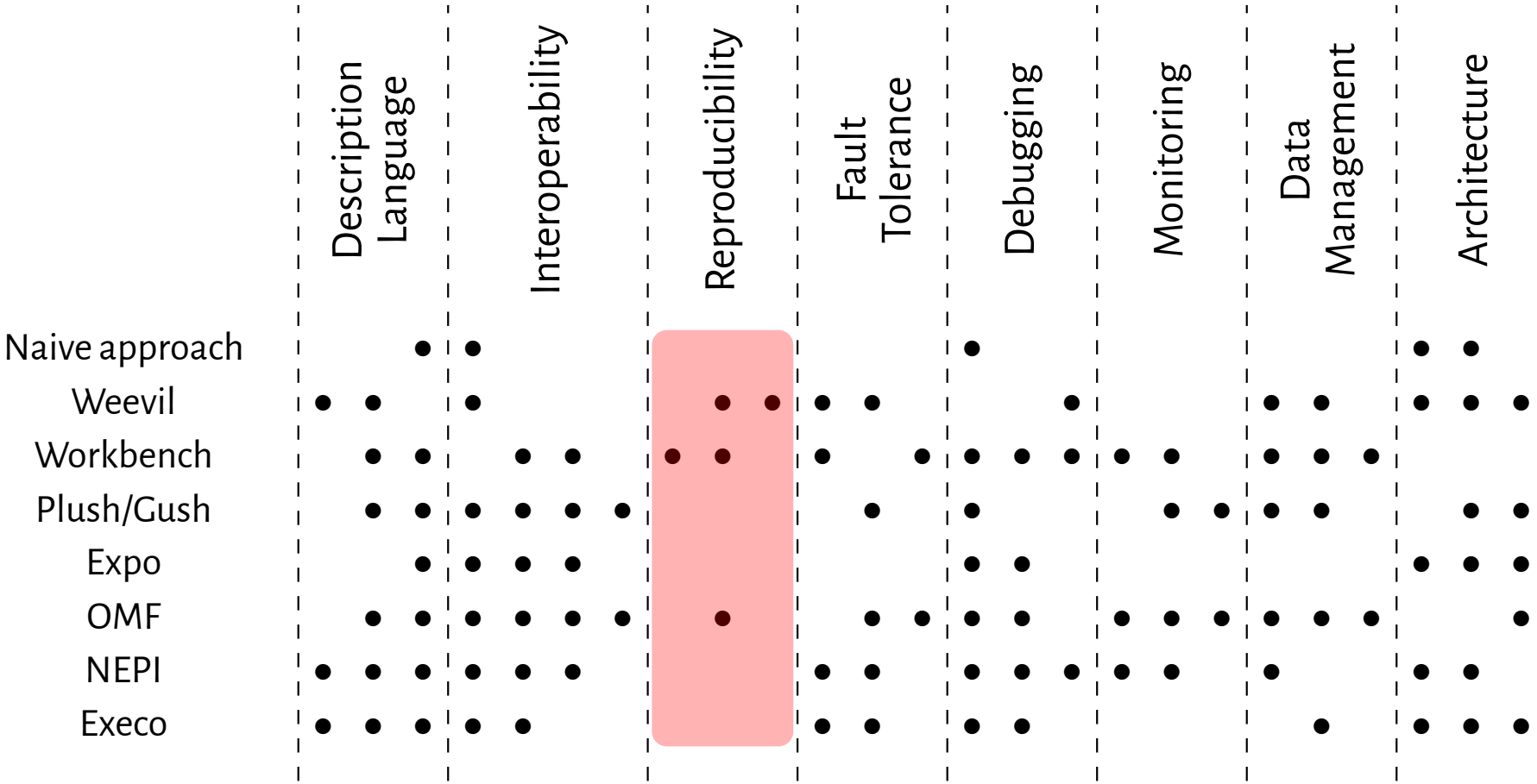
Survey of experiment management tools[1]



Most tools use **imperative descriptions** of experiments (vs declarative)

[1] Tomasz Buchert et al. "A survey of general-purpose experiment management tools for distributed systems". In: *Future Generation Computer Systems* 45 (2015), pp. 1–12.

Survey of experiment management tools[1]



Support for Reproducibility is almost **nonexistent**

[1] Tomasz Buchert et al. "A survey of general-purpose experiment management tools for distributed systems". In: *Future Generation Computer Systems* 45 (2015), pp. 1–12.

Business Process Management (BPM)

- a successful methodology to manage complex processes
- focused on **business processes**

Business process:

a set of activities performed in coordination
in organizational and technical environment to achieve a business goal

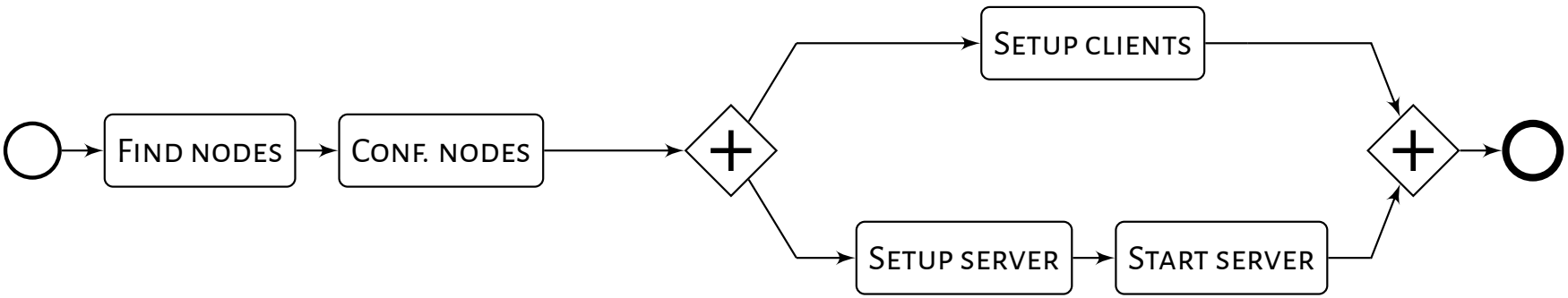
Examples:

- administrative tasks in an organization
- production in a factory
- delivering products

The goal of BPM $\rightarrow$ formalize and understand business processes

Can Business Process Management be used to manage complex experiments?

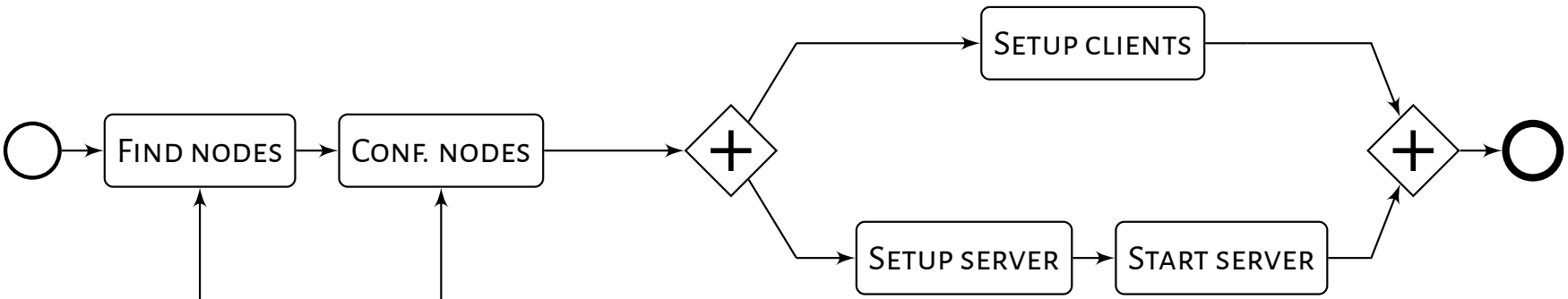
XPFlow: Experiment workflows in a DSL[2]




```
process :experiment do
    find_nodes
    configure_nodes
    parallel do
        setup_clients
        sequence do
            setup_server
            start_server
        end
    end
end
```

[2] Tomasz Buchert, Lucas Nussbaum, and Jens Gustedt. "A workflow-inspired, modular and robust approach to experiments in distributed systems". In: *CCGrid*. 2014.

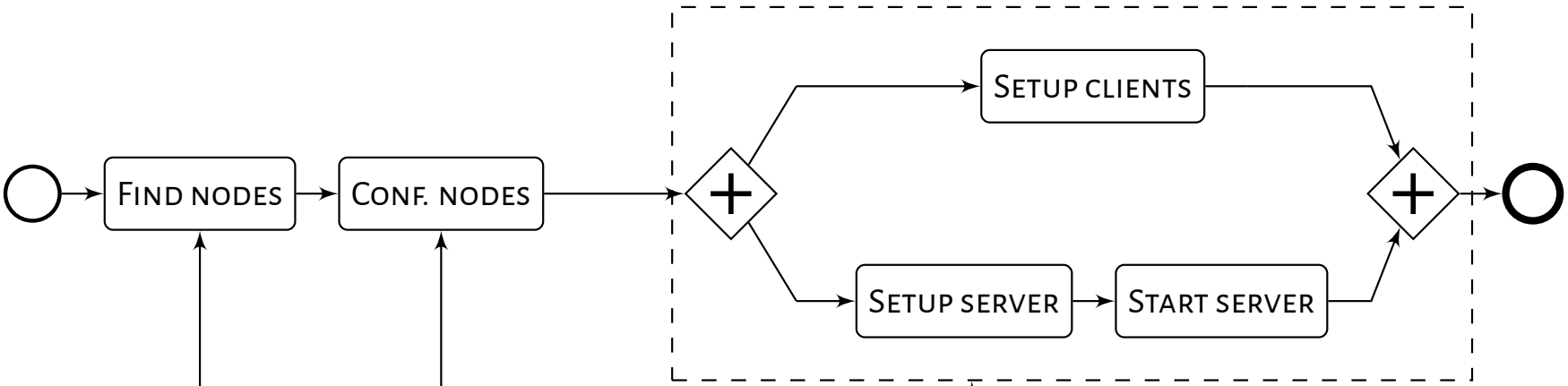
XPFlow: Experiment workflows in a DSL[2]




```
process :experiment do
    find_nodes
    configure_nodes
    parallel do
        setup_clients
        sequence do
            setup_server
            start_server
        end
    end
end
```

[2] Tomasz Buchert, Lucas Nussbaum, and Jens Gustedt. "A workflow-inspired, modular and robust approach to experiments in distributed systems". In: *CCGrid*. 2014.

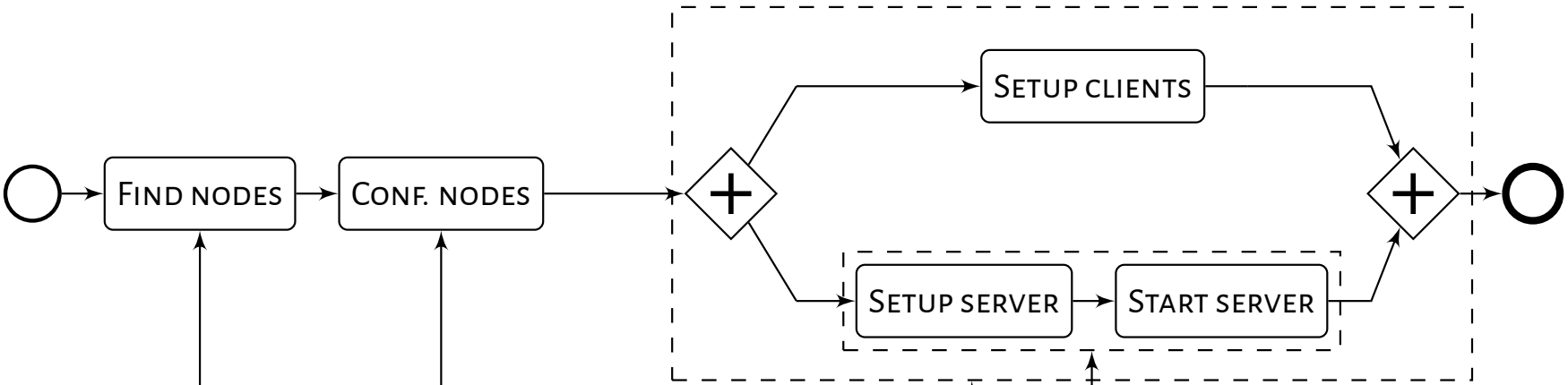
XPFlow: Experiment workflows in a DSL[2]




```
process :experiment do
  find_nodes
  configure_nodes
  parallel do
    setup_clients
    sequence do
      setup_server
      start_server
    end
  end
end
```

[2] Tomasz Buchert, Lucas Nussbaum, and Jens Gustedt. "A workflow-inspired, modular and robust approach to experiments in distributed systems". In: *CCGrid*. 2014.

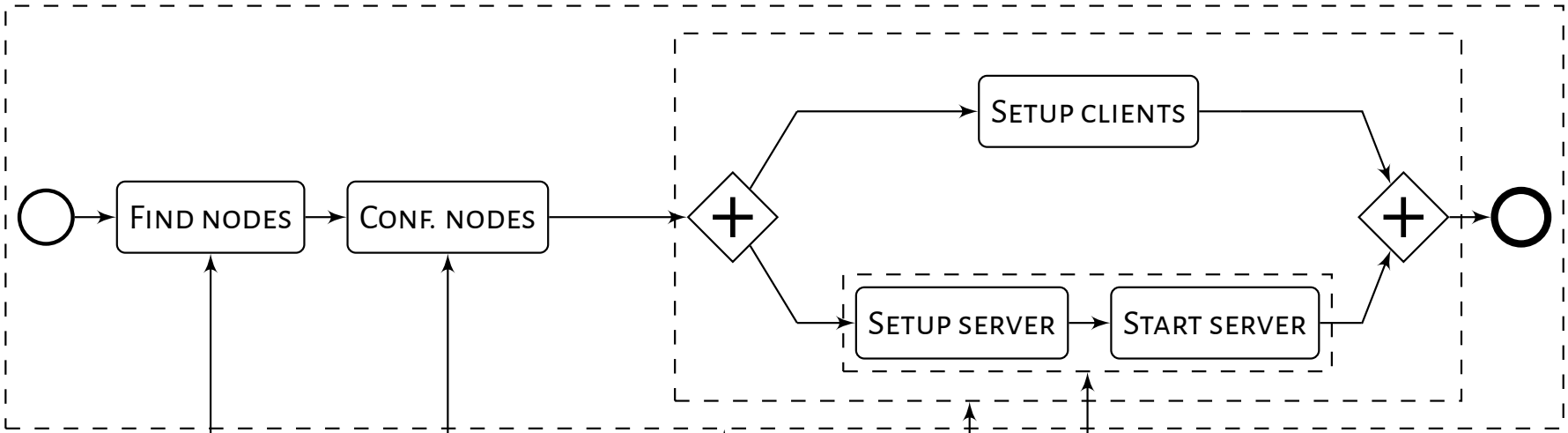
XPFlow: Experiment workflows in a DSL[2]




```
process :experiment do
  find_nodes
  configure_nodes
  parallel do
    setup_clients
    sequence do
      setup_server
      start_server
    end
  end
end
```

[2] Tomasz Buchert, Lucas Nussbaum, and Jens Gustedt. "A workflow-inspired, modular and robust approach to experiments in distributed systems". In: *CCGrid*. 2014.

XPFlow: Experiment workflows in a DSL[2]




```
process :experiment do
    find_nodes
    configure_nodes
    parallel do
        setup_clients
        sequence do
            setup_server
            start_server
        end
    end
end
```

[2] Tomasz Buchert, Lucas Nussbaum, and Jens Gustedt. "A workflow-inspired, modular and robust approach to experiments in distributed systems". In: *CCGrid*. 2014.

Evaluation of XPFlow

Three case studies:

1. **performance evaluation** of an HTTP server
2. **large-scale comparison** of command execution methods
3. evaluation of data distribution techniques **under varying conditions**

Focus on:

- **how** these experiments are done
- **what** can be improved
- less so on the results

Conducted with Grid'5000

Case study II

Evaluation of **parallel command execution methods**:

- ClusterShell (sliding window approach)
- TakTuk 2 (tree topology with arity = 2)
- TakTuk 3 (tree topology with arity = 3)

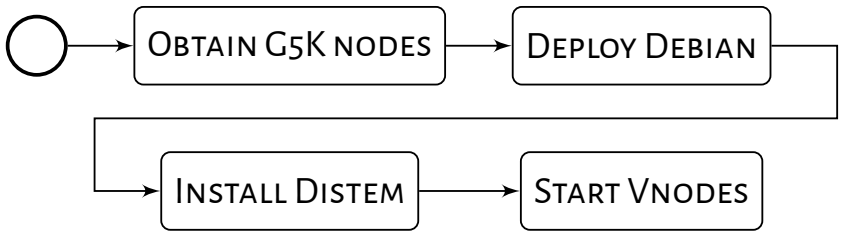
Focus → **scalability** and **failure handling**

Scalability pushed to extreme:

- lightweight containers (with Distem[3])
- high-performance network (InfiniBand)
- efficient network overlay (VXLAN)

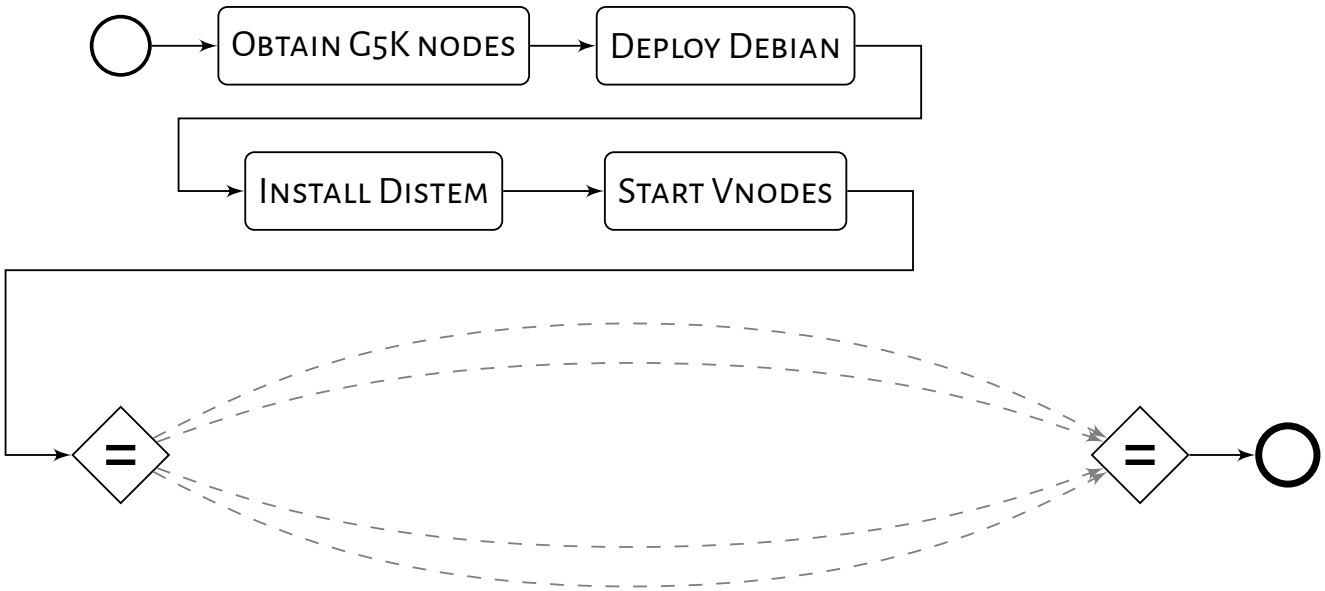
[3] Luc Sarzyniec et al. "Design and Evaluation of a Virtual Experimental Environment for Distributed Systems". In: *PDP2013*. IEEE, 2013.

Experiment workflow



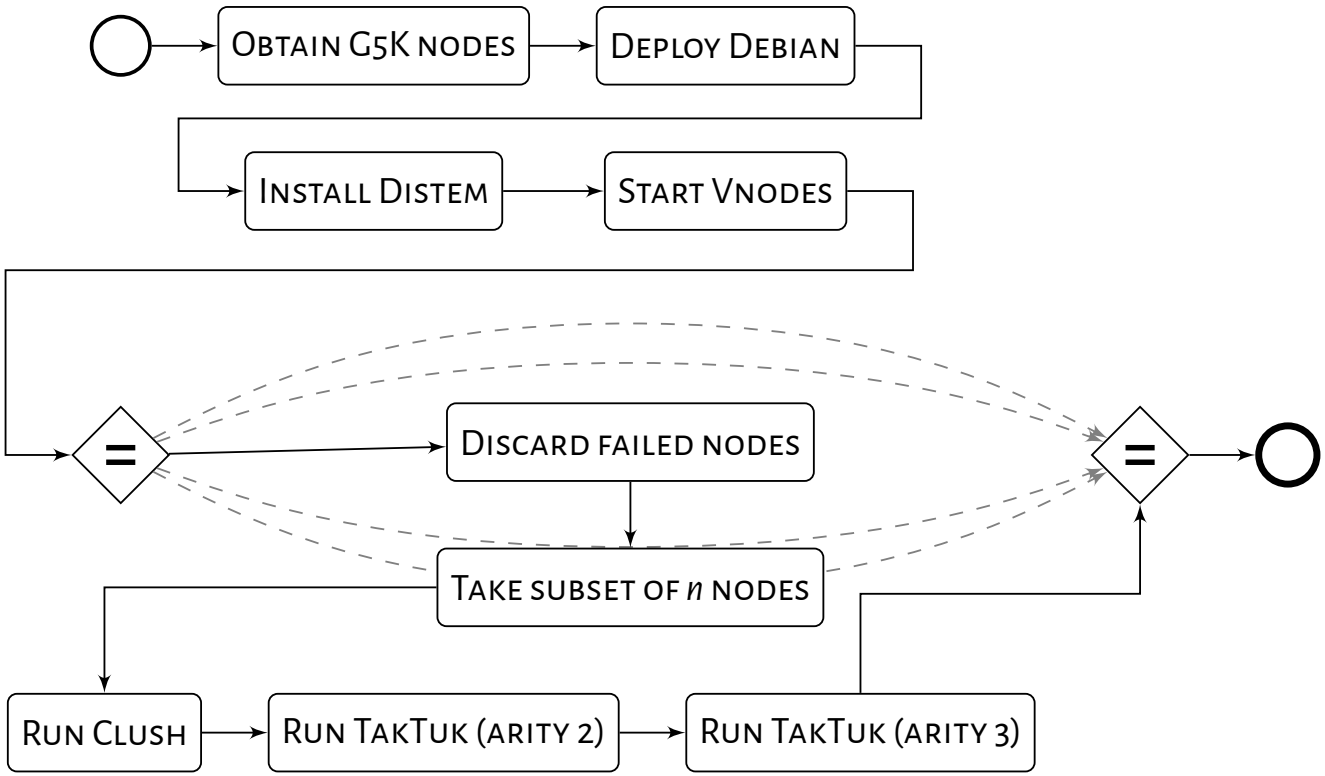
Basic, reusable setup

Experiment workflow



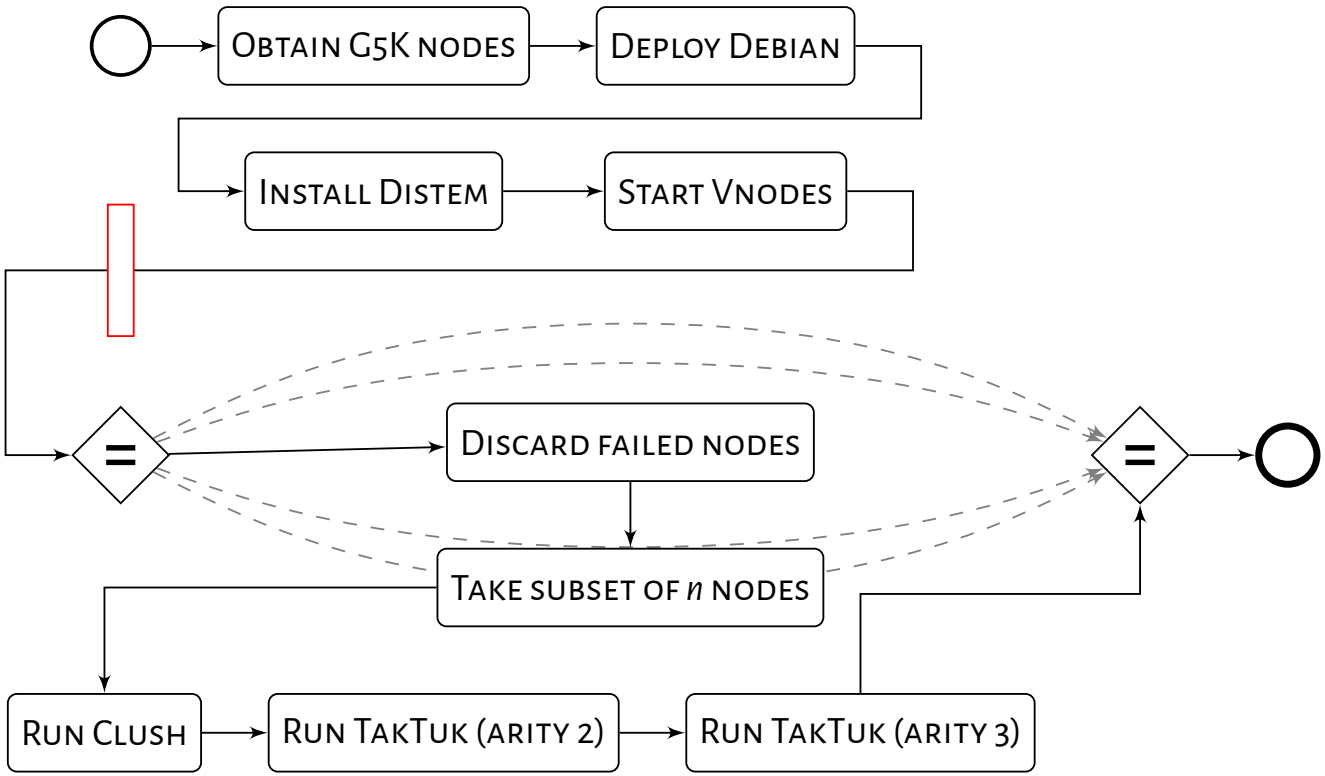
Experimental pattern (foreach)

Experiment workflow



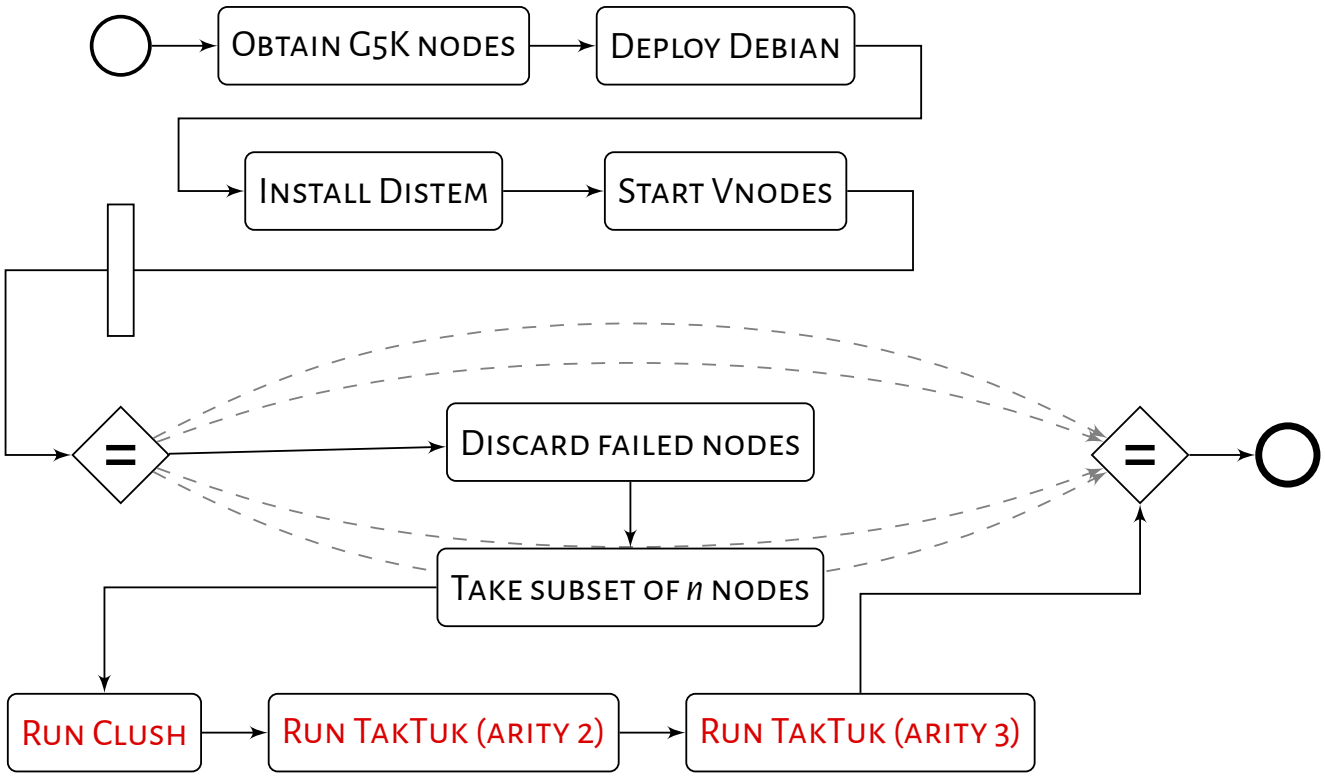
Main experiment (evaluation of the methods)

Experiment workflow



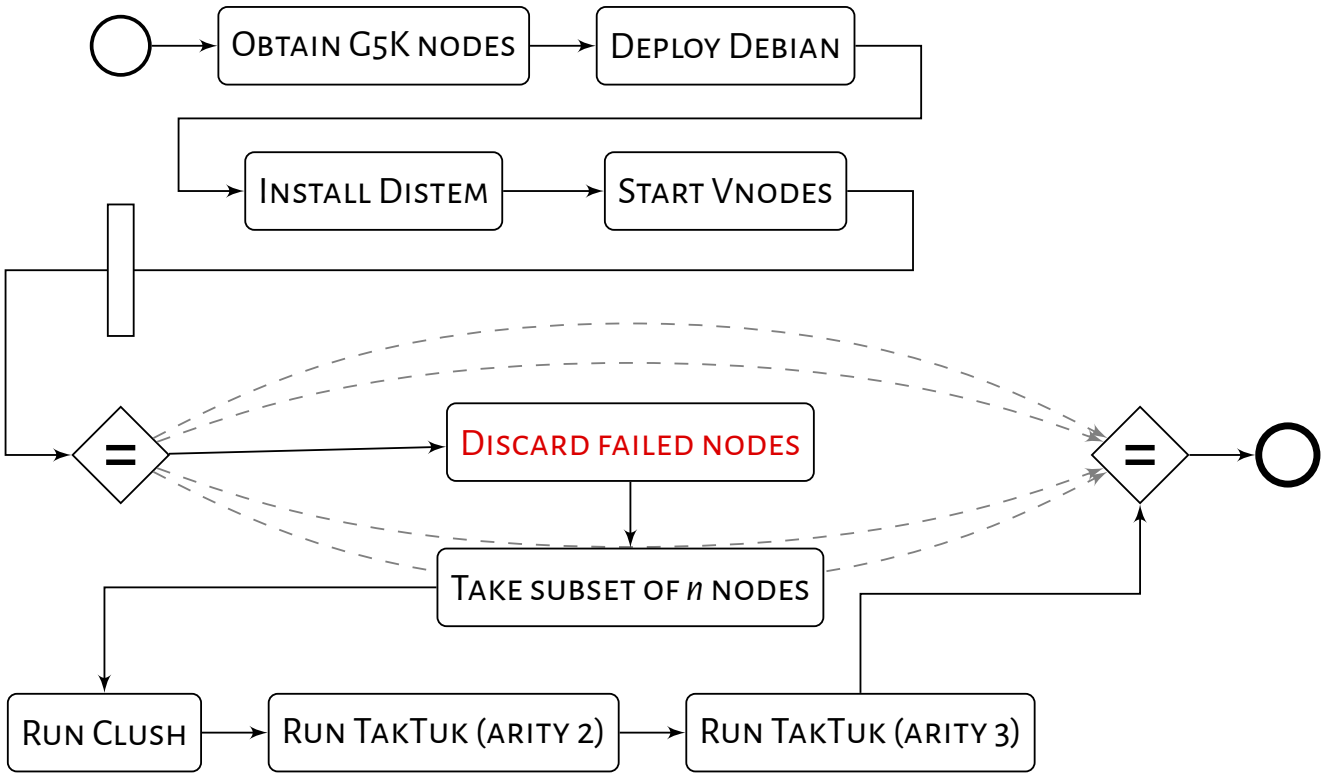
Checkpointing on the workflow level

Experiment workflow



And on the level of stored results

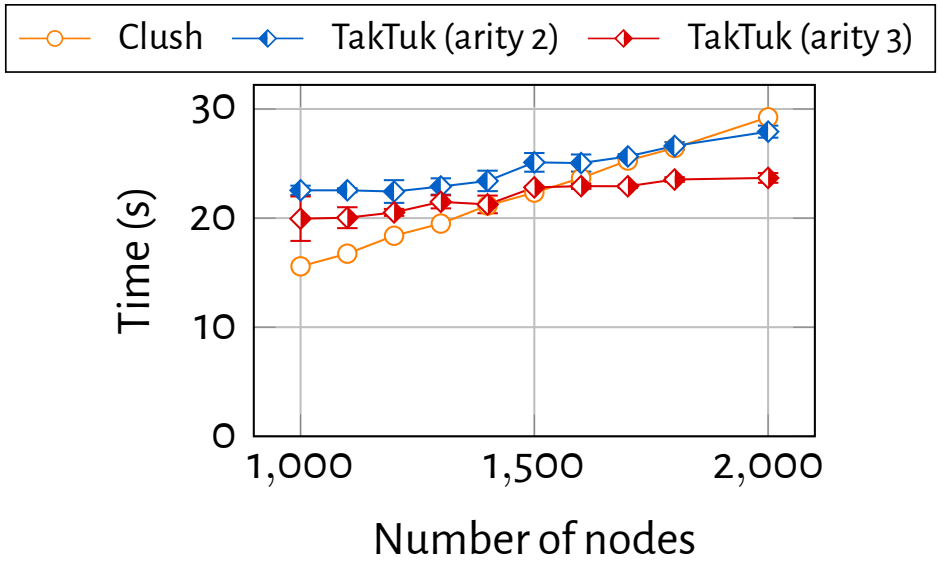
Experiment workflow



Custom, failure-handling activity

Results

- Experiment:



TakTuk more efficient for more than 2000 nodes[4]

- Case study:
  - up to almost **40 000** virtual nodes used
  - heavy use of **checkpointing** (saved time)
  - integration with **external tools** (Distem emulator)

[4] Tomasz Buchert, Emmanuel Jeanvoine, and Lucas Nussbaum. "Emulation at Very Large Scale with Distem". In: *Scale (CCGrid)*. 2014.

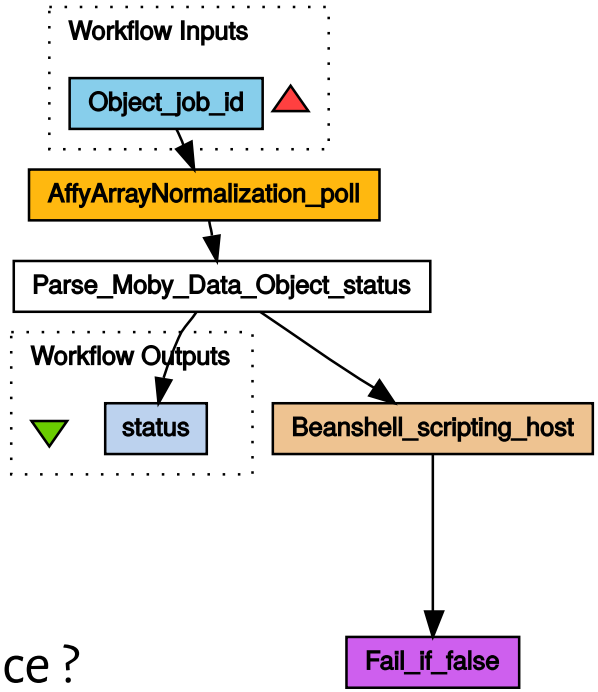
Provenance collection

Provenance rarely collected in distributed systems experiments or BPM:

- often **manual**
- **technically challenging** (scalability issues)
- hard to **interpret/query afterwards**
- potentially harmful impact → *observer effect*

Collected by **scientific workflow systems**:

- by tracing how **data** is **transformed**
- a well-defined **model** (directed, acyclic graphs)



data provenance = provenance ?

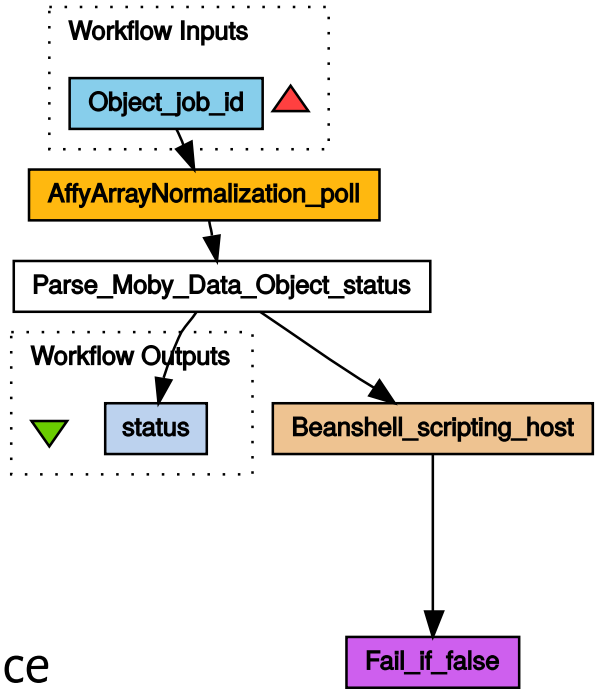
Provenance collection

Provenance rarely collected in distributed systems experiments or BPM:

- often **manual**
- **technically challenging** (scalability issues)
- hard to **interpret/query afterwards**
- potentially harmful impact $\rightarrow$ *observer effect*

Collected by **scientific workflow systems**:

- by tracing how **data** is **transformed**
- a well-defined **model** (directed, acyclic graphs)



data provenance $\neq$ provenance

Three types of provenance

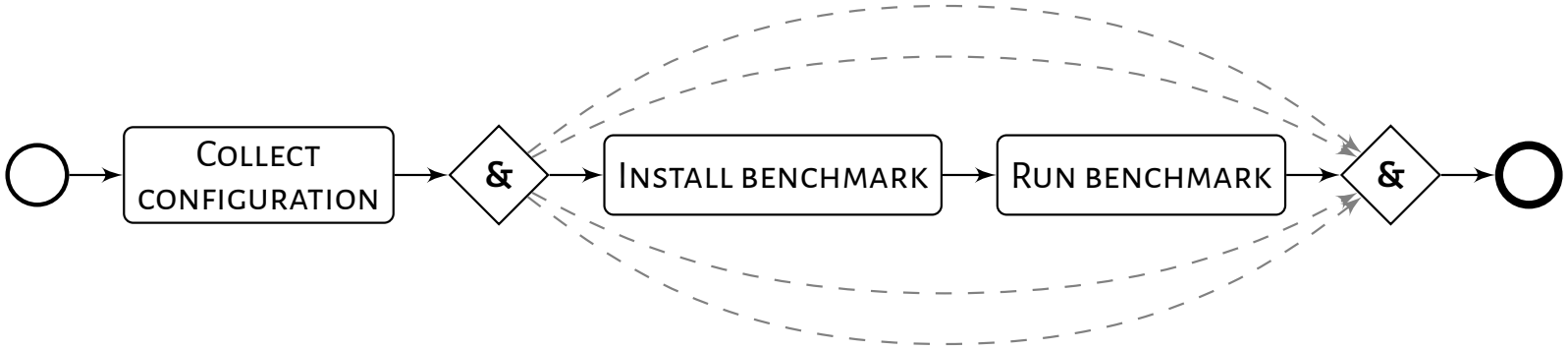
Our proposal → new classification into **three** types:

- **description provenance** – experiment description (textual or graphical)
- **data provenance** – data produced by the experiment
- **process provenance** – runtime and causal information

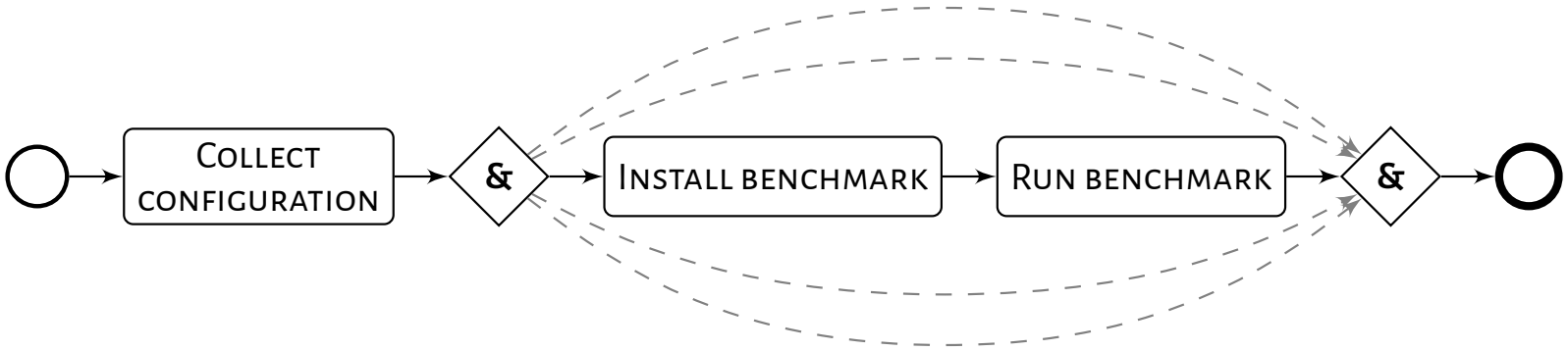
Why the new classification?

- DS experiments rarely about **data**
- focus on the **runtime behavior** of a system
- different structure, optimal storage, graphical representation

Questions about provenance



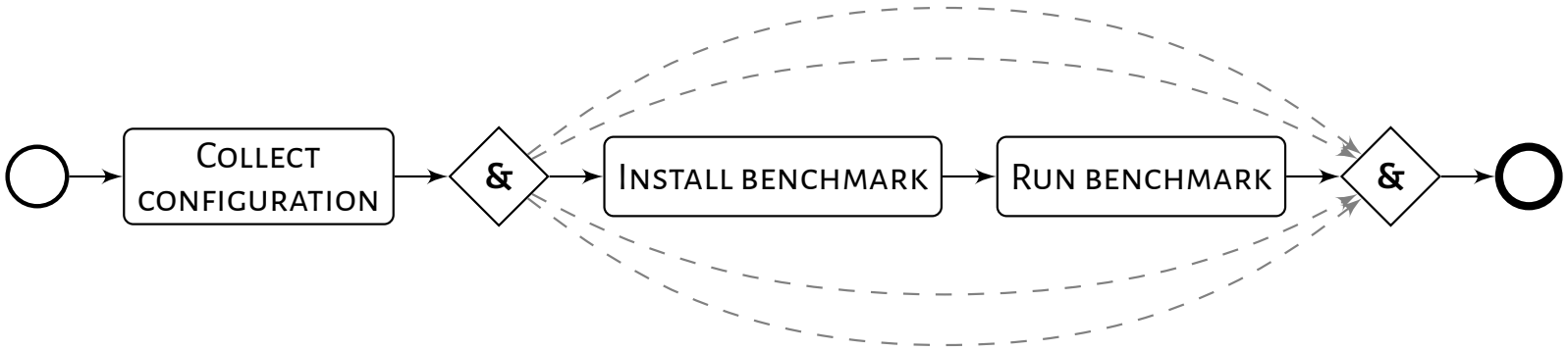
Questions about provenance



Questions about description provenance:

- What is the description of the experiment?
- What are the dependencies between activities?
- What is the authorship of activities?

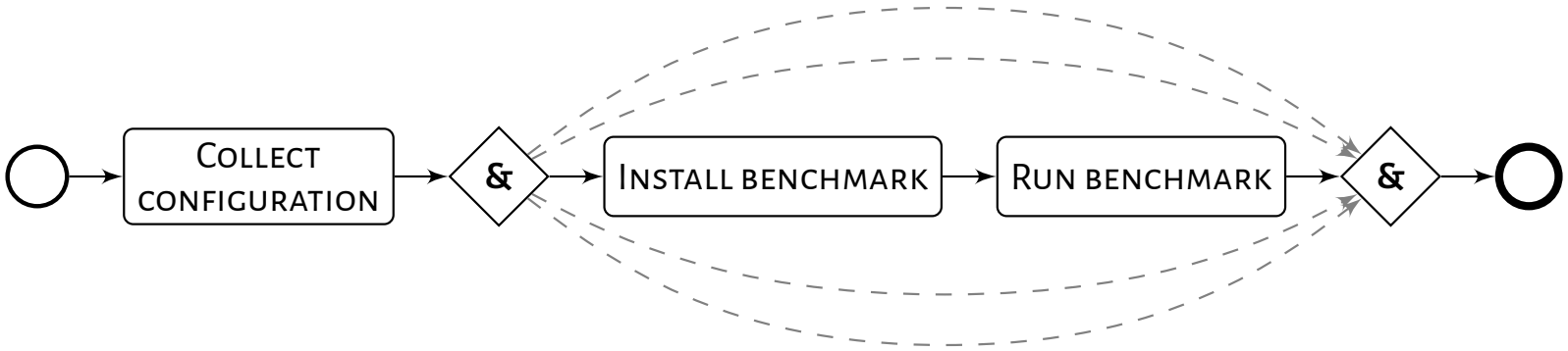
Questions about provenance



Questions about process provenance:

- What is the execution time of the Collect configuration activity?
- How many nodes executed the parallel loop?
- Which node finished Run benchmark last?

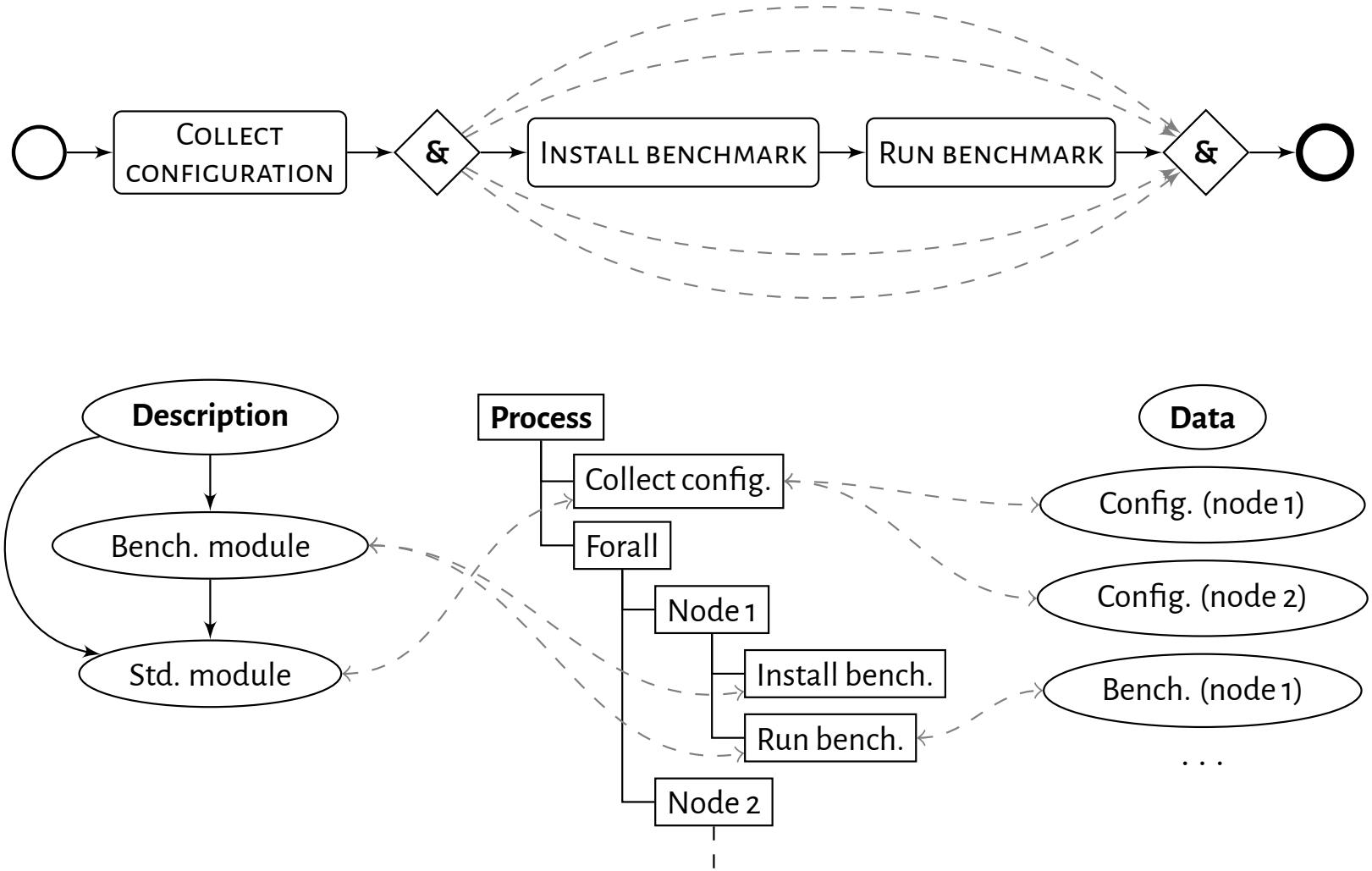
Questions about provenance



Questions about data provenance:

- What are the CPUs of the nodes?
- What are the results of the benchmark?
- What is the network topology?

Automatic provenance collection implemented in XPFlow



Conclusions

This work explored the use of

BPM workflows patterns as a model for representing the large-scale experiments in research on distributed systems

1. Analyzed the existing solutions to:
   - construct an evaluation framework
   - uncover important missing functionality and trends
2. Implemented and evaluated the workflow-based approach that:
   - is a robust, fault-tolerant and scalable approach to experiment control
   - is on a par with the best current approaches (feature-wise)
3. Proposed and evaluated a new classification of provenance that:
   - makes the collection of provenance an automatic task
   - is the first work on provenance in control-flows

Kameleon: reconstructable software appliances

(Cristian Ruiz. *Methods and Tools for Challenging Experiments on Grid'5000: a use case on electromagnetic hybrid simulation*)

Provide an experiment's software environment?

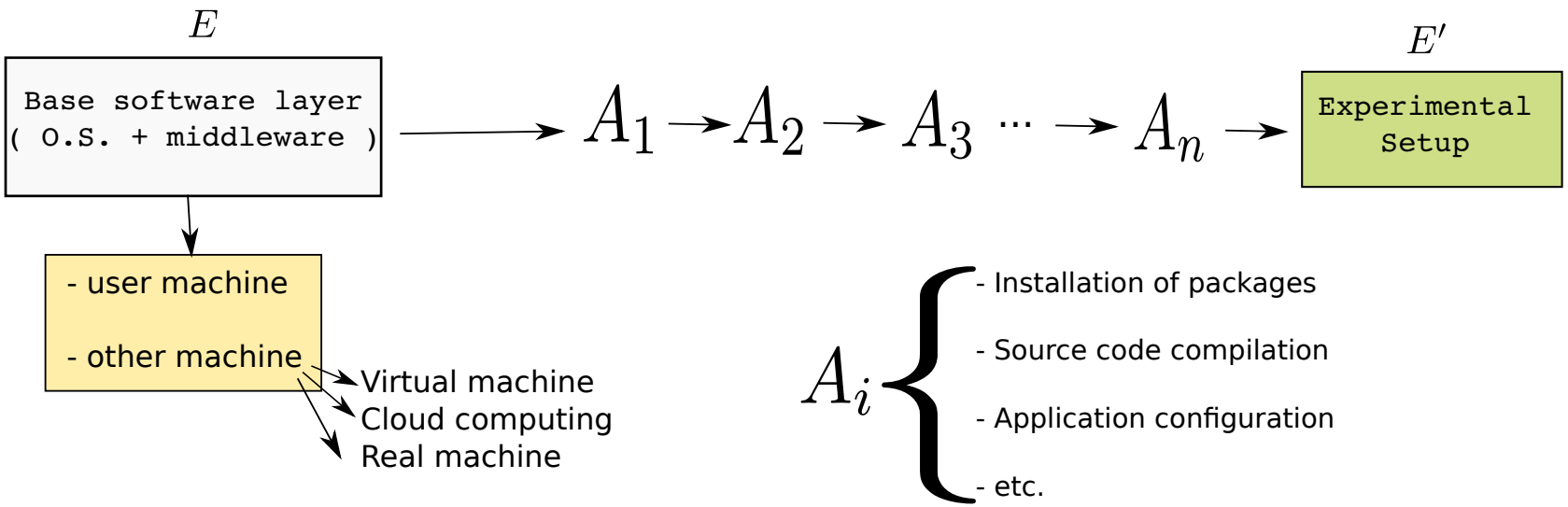
Provide an experiment's software environment?

- Source code of the main application?
  - Might fail to build
  - Might provide different performance (toolchain, libs)

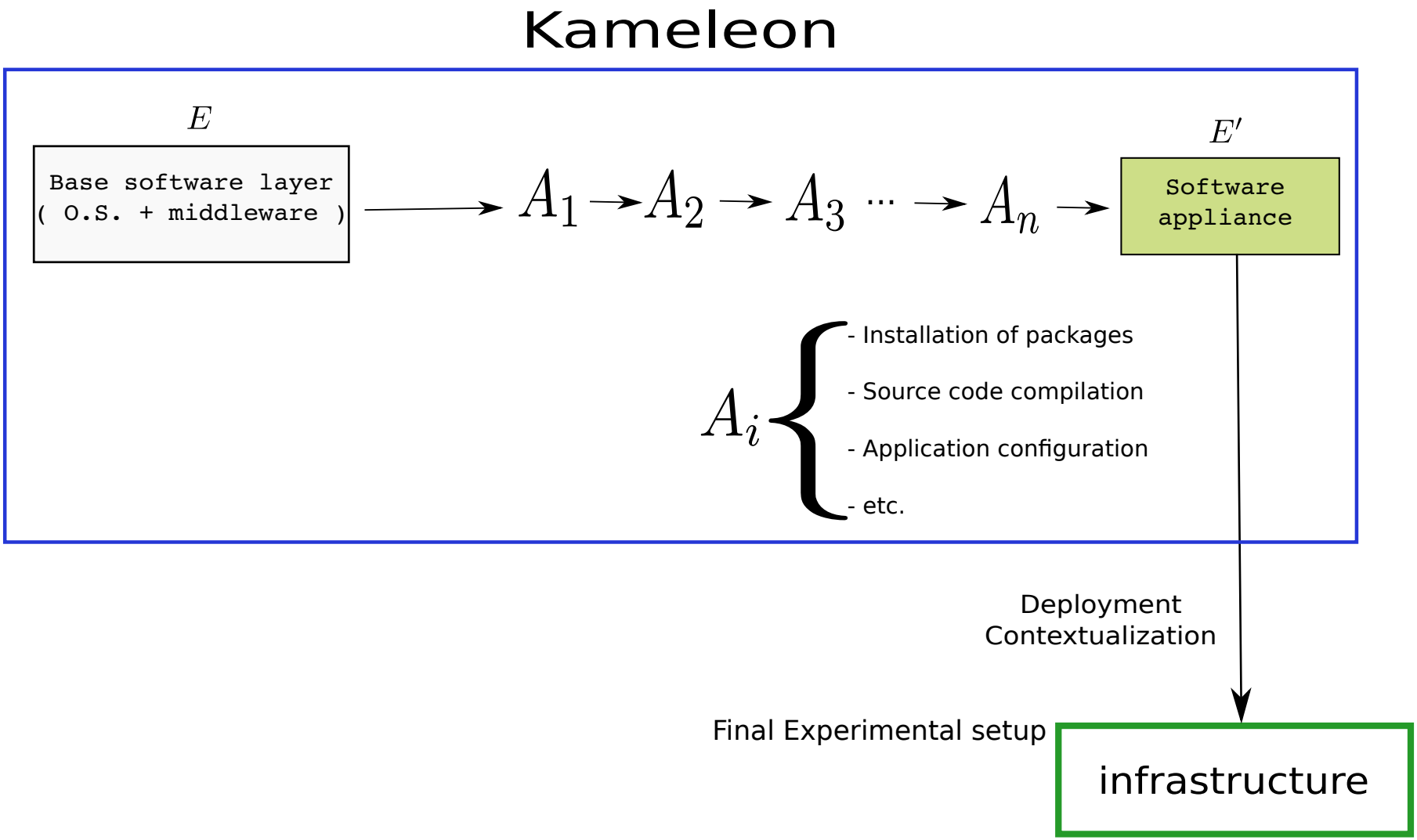
Provide an experiment's software environment?

- Source code of the main application?
  - Might fail to build
  - Might provide different performance (toolchain, libs)
- System images?
  - Don't know how they were built, or the exact ingredients
  - Not the preferred form for further modification
  - Just have the final result: binary code, not source code
  - Sometimes specific to a given environment, cannot be adapted

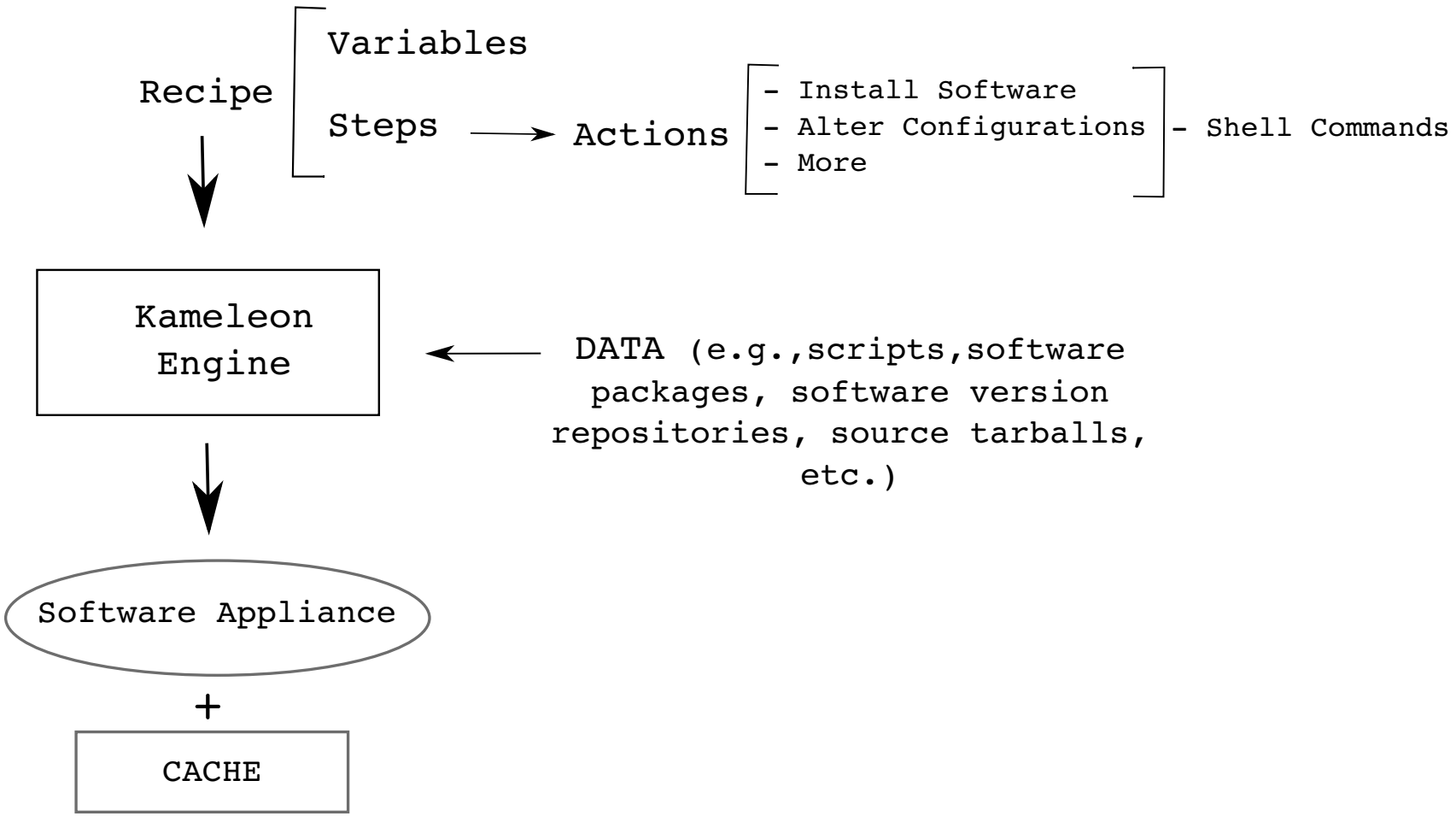
Kameleon: Creation process of an experimental setup



Kameleon: Creation process of an experimental setup

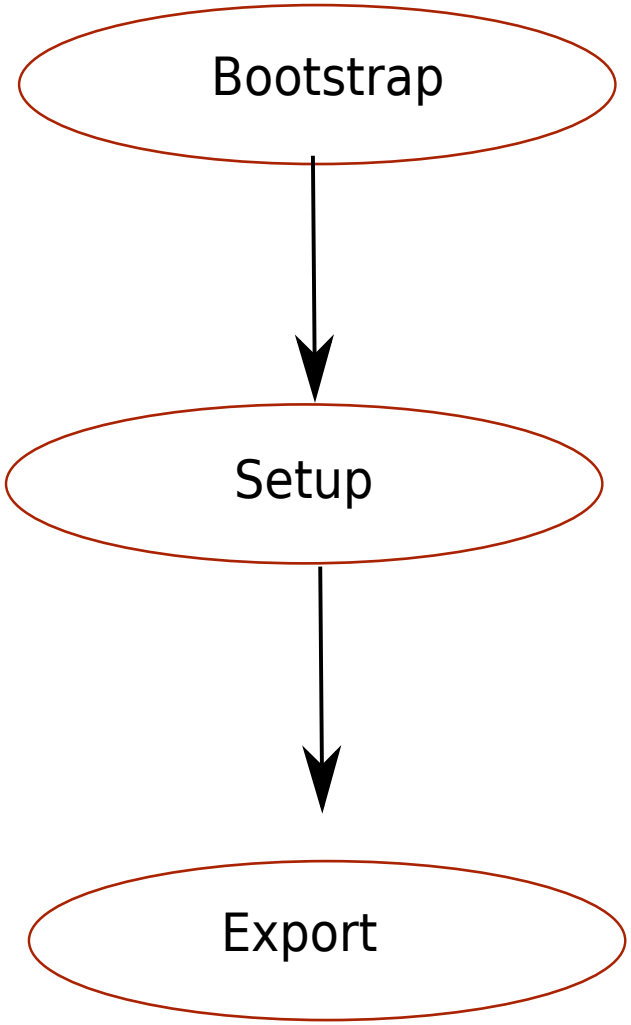


Kameleon overview



Kameleon recipe

```
Bootstrap:
  - start_virtualbox
  - install_requirements:
      - packages: debootstrap extlinux
  - debootstrap:
Setup:
  - configure_kernel
  - configure_keyboard:
     - layout: "us,fr,de"
  - configure_network
Export:
  - virtualbox_save_appliance:
     - output: "$${kameleon_recipe_name}"
     - save_as_ova
  # - save_as_qcow2
```



Step and microstep

Each step contains a list of microsteps that in turn contain a list of commands (YAML)

```
- hadoop_directory:
    - exec_in: |
               mkdir -p /app/hadoop/tmp
               chown hduser:hadoop /app/hadoop/tmp
               chmod 750 /app/hadoop/tmp

- config_hadoop:

    - local2in:
               - $$kameleon_recipe_dir/data/hadoop-env.sh
               - /usr/local/hadoop/conf/hadoop-env.sh

    - local2in:
               - $$kameleon_recipe_dir/data/core-site.xml
               - /usr/local/hadoop/conf/core-site.xml

- formating_filesystem:
    - exec_in: /usr/local/hadoop/bin/hadoop namenode -format
```

Extension mechanism

Base recipe

```
global:

  distrib: debian
  release: wheezy
  arch: amd64

  virtualbox_memory_size: 768
  virtualbox_ssh_port: 55423
  virtualbox_os_type: Debian_64
  apt_repository: http://ftp.debian.org/debian/

  in_context:
    cmd: ssh -F $$ssh_config_file $${kameleon_recipe_name} -t /bin/bash
    workdir: /root/kameleon_workdir
    proxy_cache: 10.0.2.2`

bootstrap:
  - enable_checkpoint
  - prepare_virtualbox
  - start_virtualbox
  - install_requirements:
    - packages: parted e2fsprogs debootstrap extlinux
  - initialize_disk
  - debootstrap:
  - switch_context_virtualbox

setup:
  # Install
  - install_software:
    - packages: >
        debian-keyring ntp sudo less vim bash-completion curl less acpic
  - configure_kernel
  - install_bootloader
  # Configuration
  - configure_system:
  - configure_keyboard:
    - layout: "us,fr,de"
  - configure_network

export:
  - virtualbox_save_appliance:
    - output: "$$kameleon_cwd/$${kameleon_recipe_name}"
    - save_as_ova
```

Extended recipe

```
extend: virtualbox/debian7.yaml

global:

bootstrap:
  - "@base"
setup:
  - "@base"
  - install_software:
    - packages: g++ make openssh-server openmpi-bin build-essential fort77
  - install_atlas:
    - repository: http://sourceforge.net/projects/math-atlas/files/Stable/
    - version: "3.10.1"
  - install_hpl:
    - repository: "http://www.netlib.org/benchmark/hpl/"
    - version: "2.1"
    - hpl_makefile: "$$kameleon_recipe_dir/data/Make.Linux"

export:
  - "@base"
```

Persistent cache to archieve reconstructability

A persistent cache mechanism brings the two followings advantages:

- Data can always be retrieved.
- The software versions will be exactly the same.

Implementation

- Kameleon uses Polipo [a] which is a tiny and lightweight web proxy to cache most of the packages that comes from the network.
- Ad hoc approaches for: revision control repository, users files.

[a] `http://www.pps.univ-paris-diderot.fr/~jch/software/polipo/`

Conclusions

Of this second PhD:

- Kameleon: software appliance builder that guarantees **reconstructability** and **supports reproducibility**
  - Persistent cache makes experimental setups **reconstructable at any time**
- **Now used to generate Grid'5000 reference images**

Of this talk:

- We are not just building a testbed: we are exploring and designing tools that supports best practices regarding reproducible research