# Feedback on Experimentation

Millian Poquet

millian.poquet@inria.fr

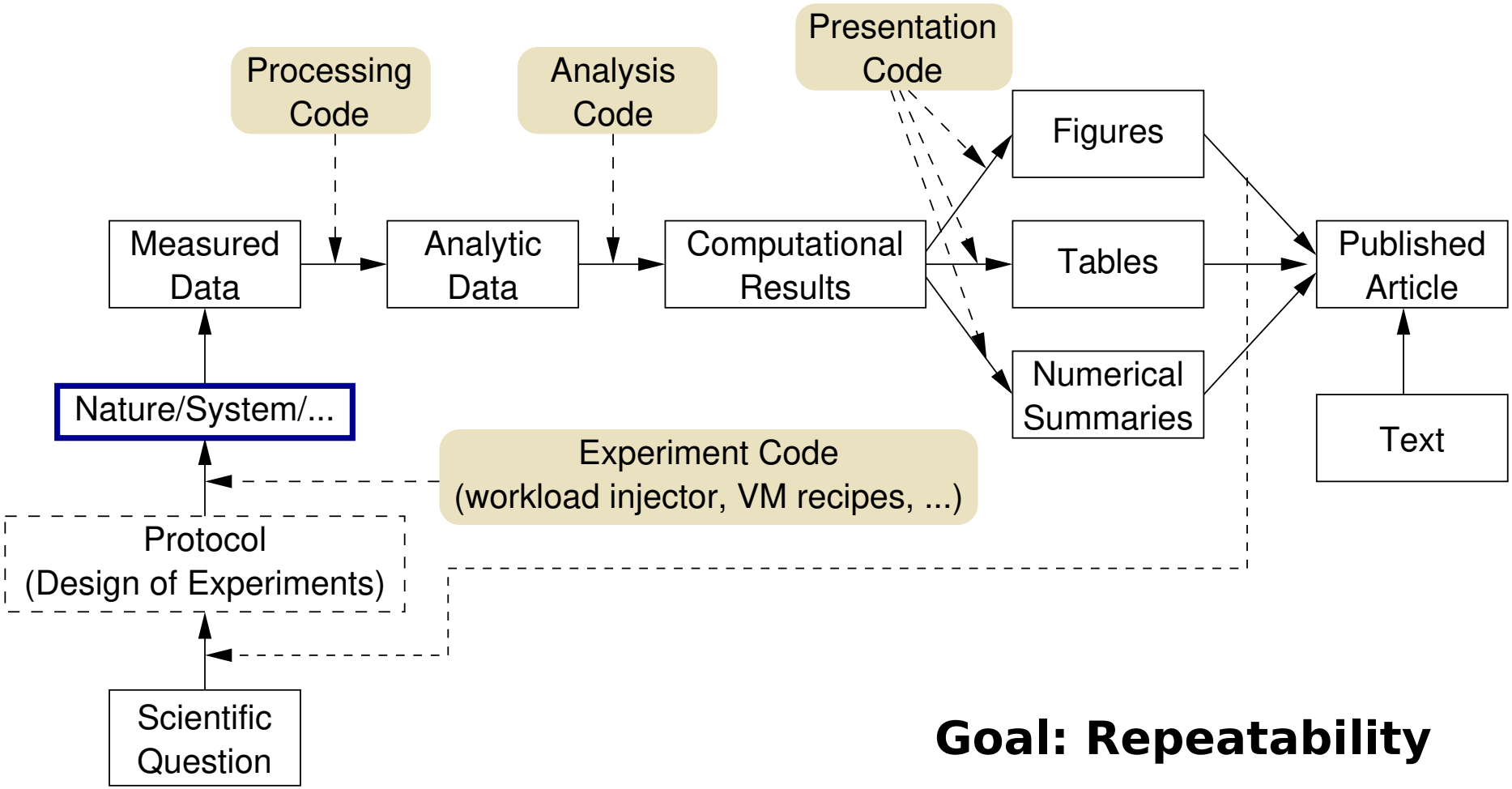
# What is this session about?

Overview of technical choices to run an experiment.

Plan

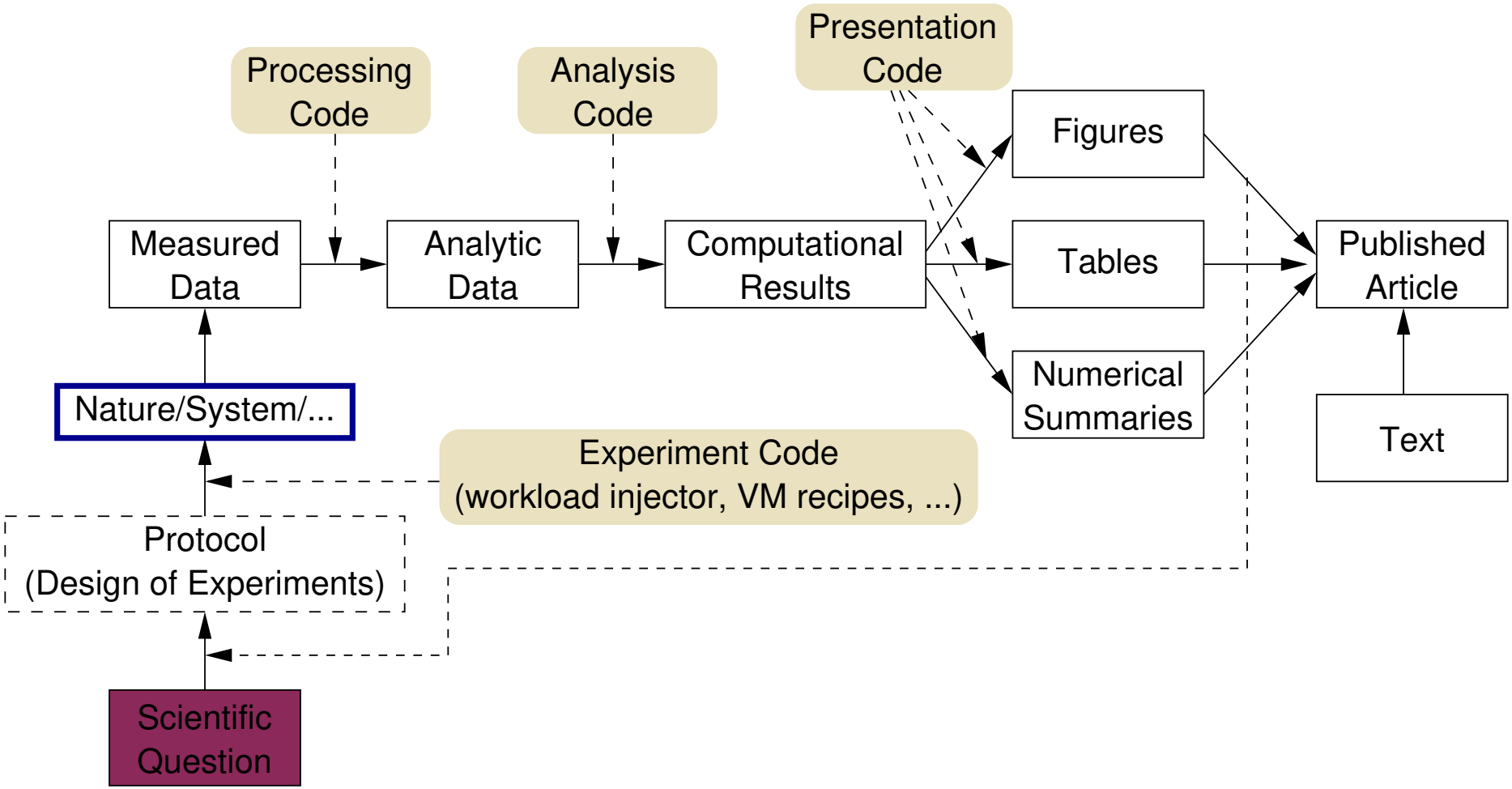
- *Short* presentation
- Discussion

# General workflow (courtesy of Arnaud Legrand)



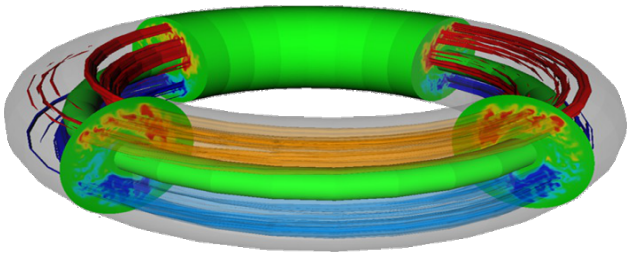

**Goal: Repeatability**

# Science

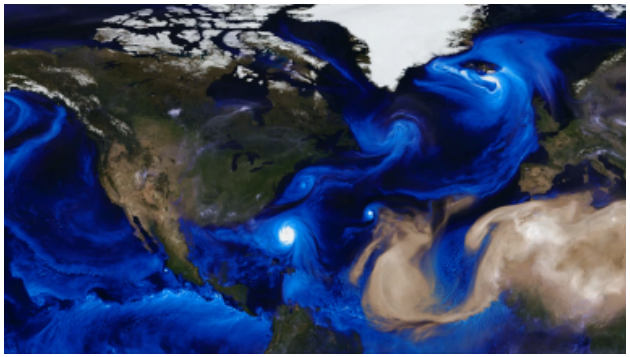

Processing Code
Analysis Code
Presentation Code
Measured Data
Analytic Data
Computational Results
Figures
Tables
Numerical Summaries
Published Article
Text
Nature/System/...
Experiment Code (workload injector, VM recipes, ...)
Protocol (Design of Experiments)
Scientific Question

# High Performance Computing

Sunway Taihulight

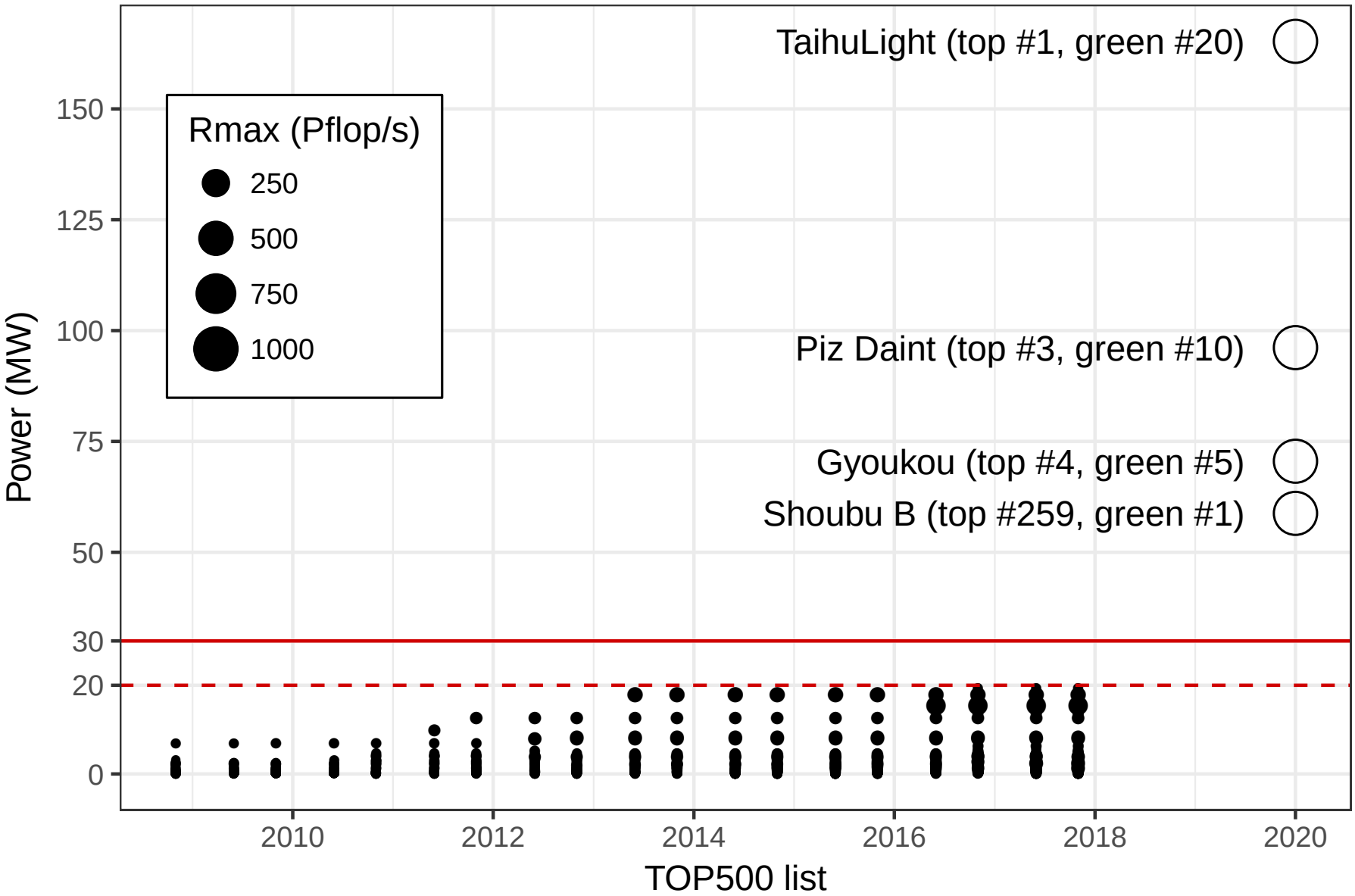


Tokamak plasmas (Gysela5D)

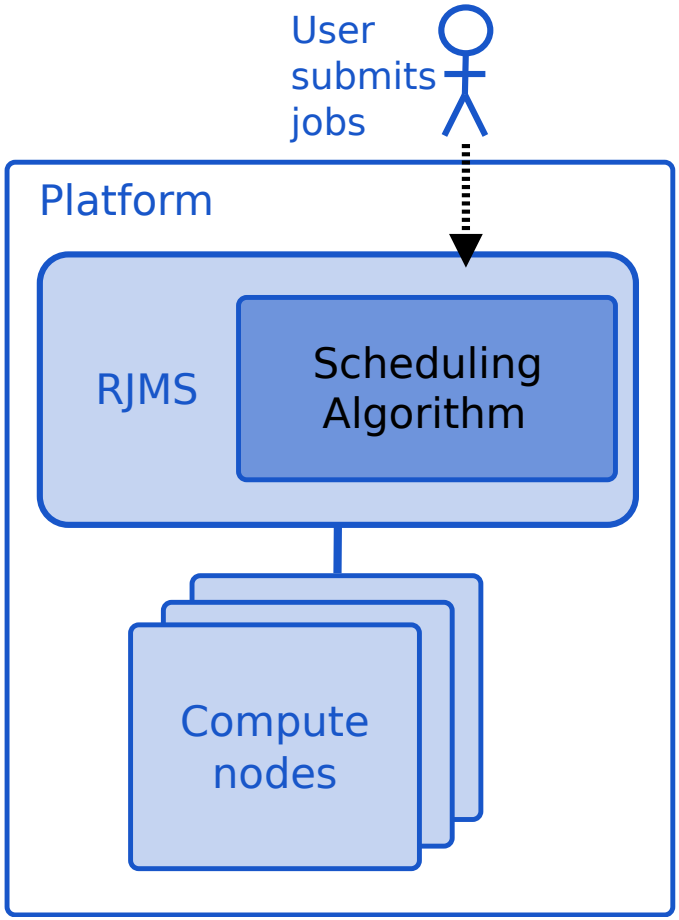


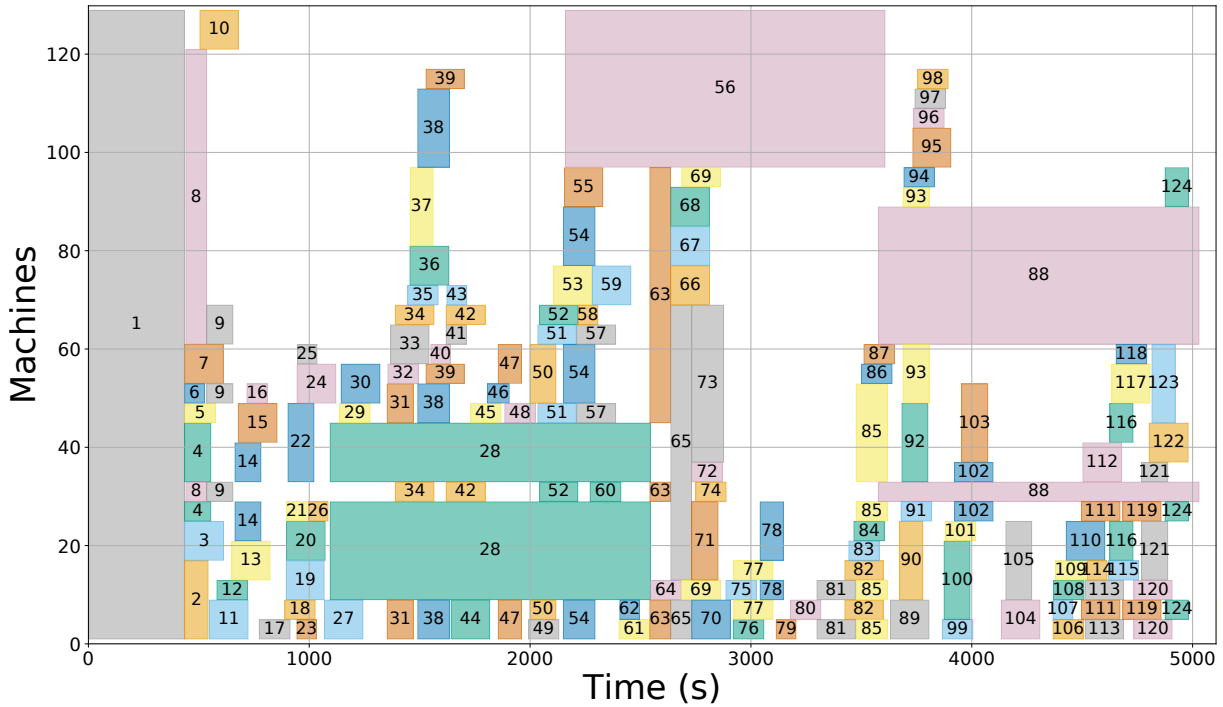
Weather prediction (GEOS)

# Problem: Power consumption (naive) projection

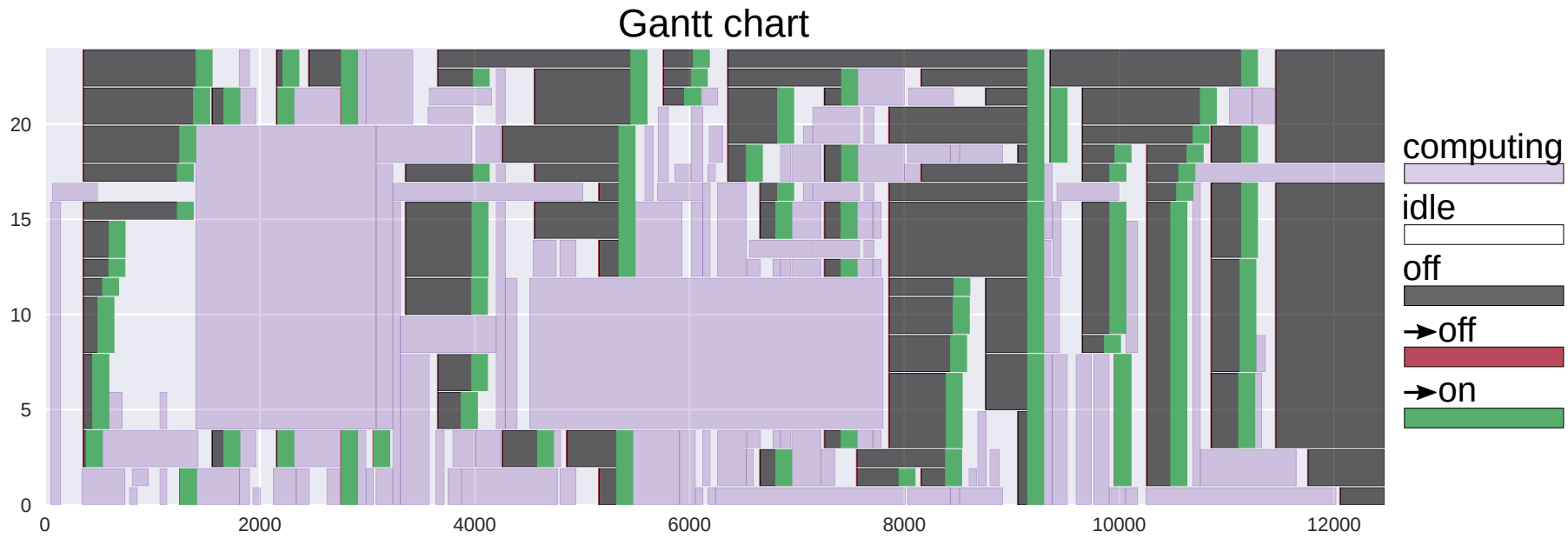


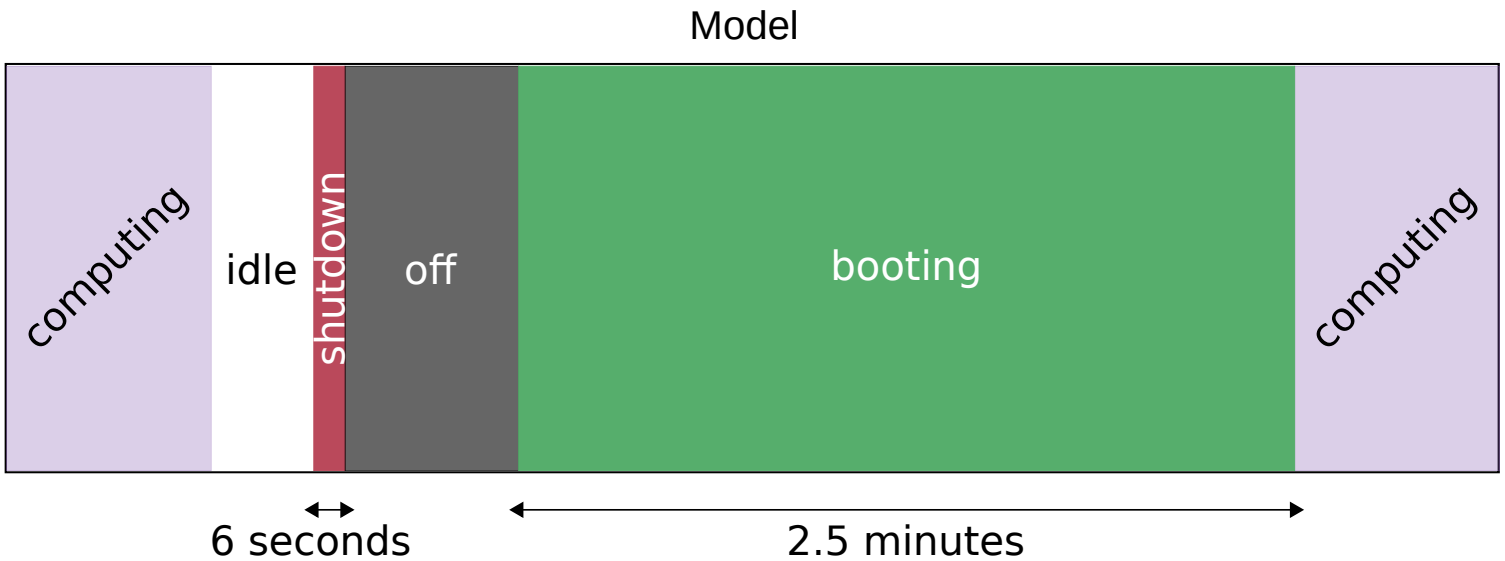
# Lever: Resources and Jobs Management System (RJMS)



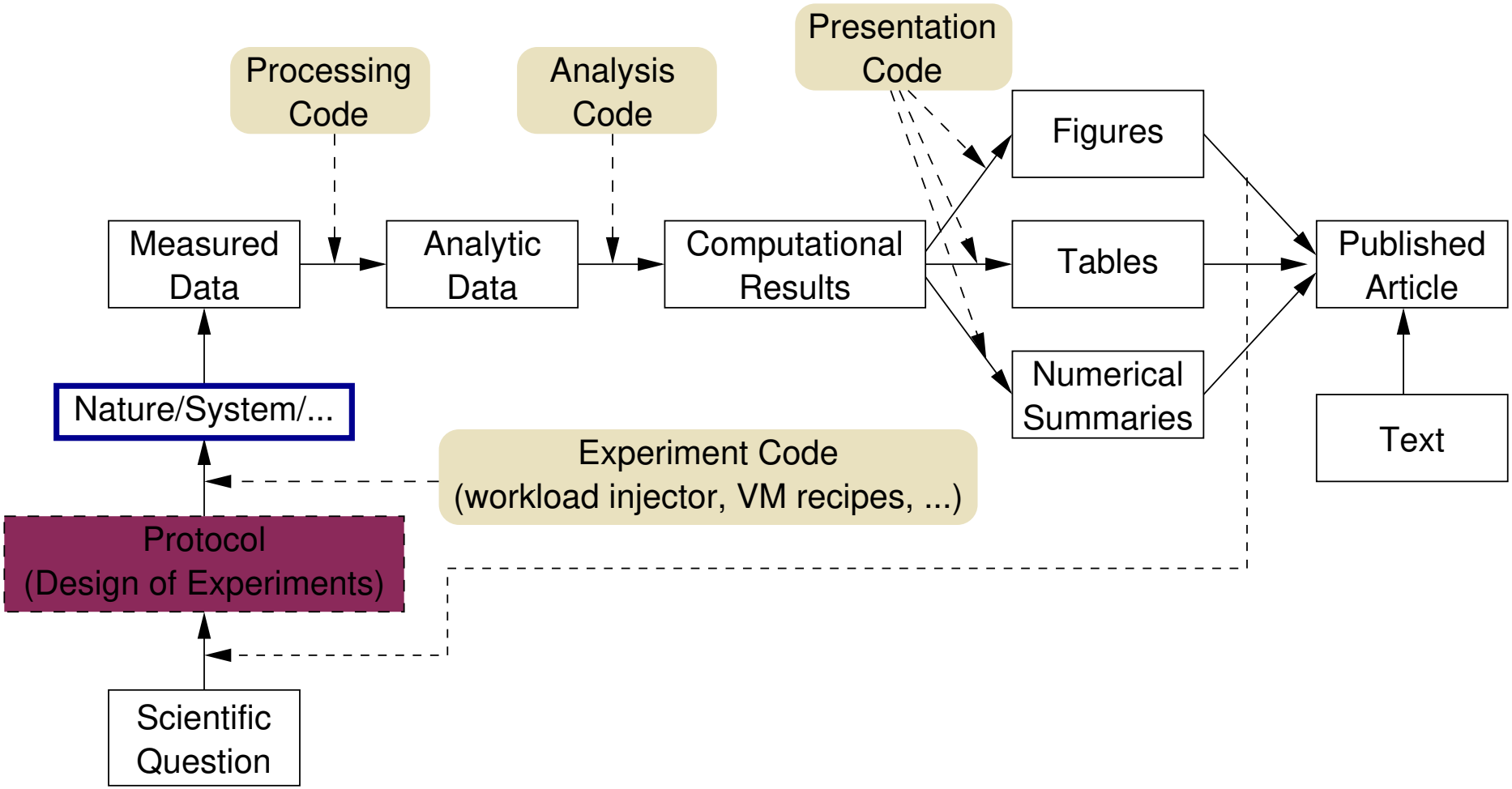



# Study: Node shutdown policies



# General workflow



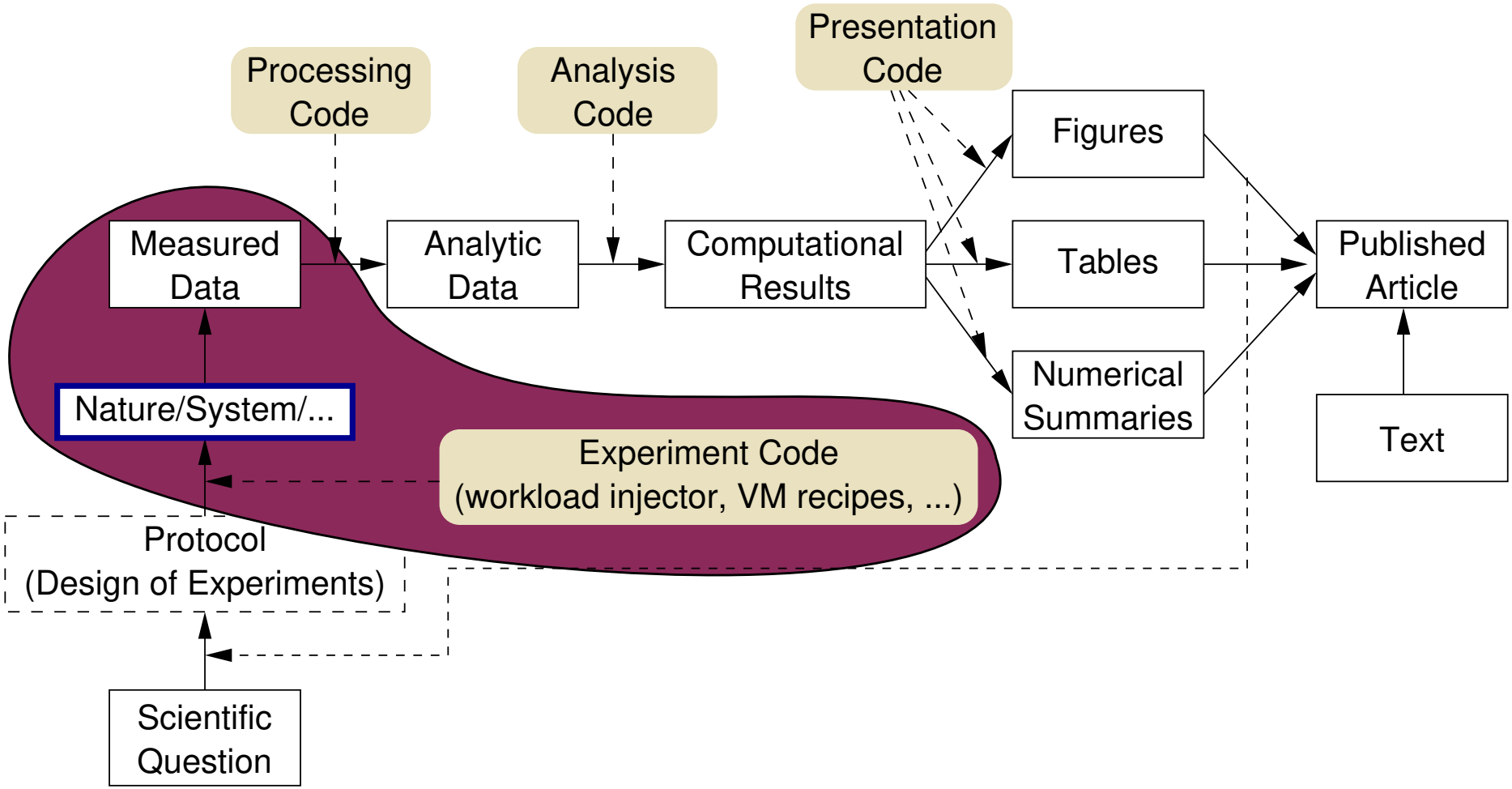
# Protocol in a nutshell

Assess node shutdown policies in simulation.

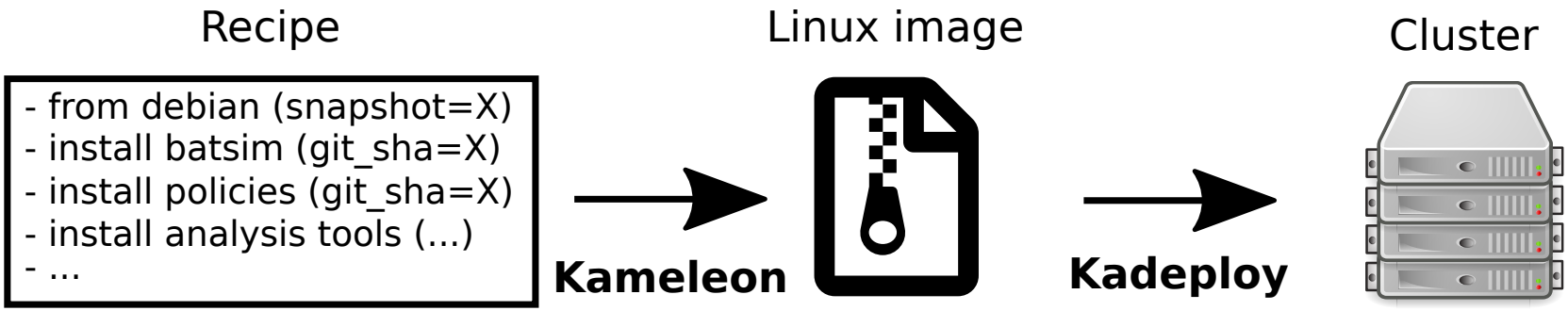
- Batsim
- (energy, job QoS)
- Various policies combined with EASY
  - Various parameters for each policy
- Replay existing HPC traces

Tune parameters **iteratively**.

# Obtaining raw data



# Reproducible environment (Bulldozer approach)

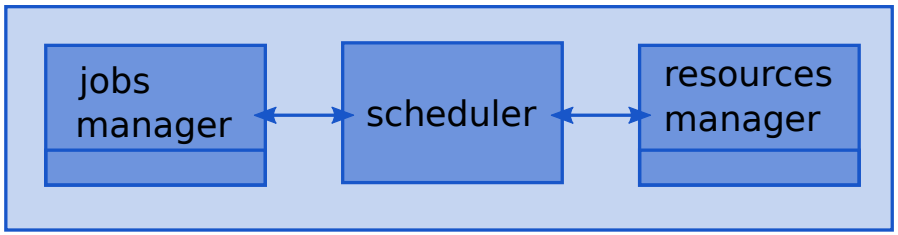


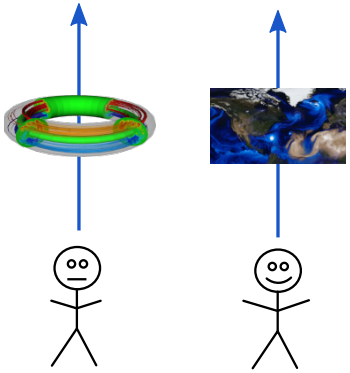

Documentation: `http://kameleon.imag.fr`

# Batsim simulation overview

## Real

RJMS (SLURM, OAR, PBS...)

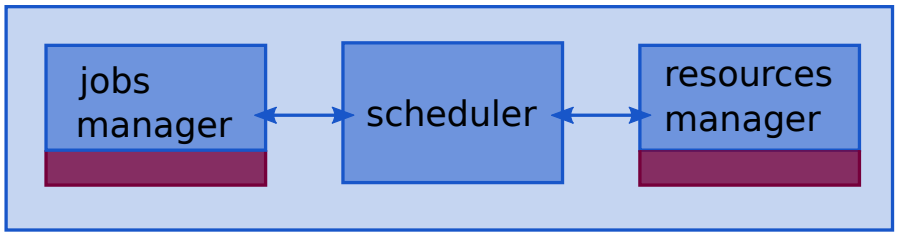


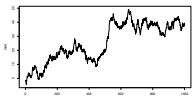



real platform

## Batsim simulation

decision maker (RJMS + adaptor)

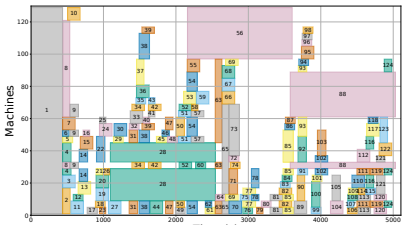


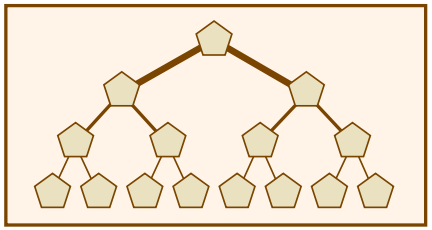
workload

batsim protocol

batsim




results



simulated platform

# Required software to run 1 instance

Simulator: Batsim (version)

- SimGrid (version)
- ZMQ, RapidJSON, Boost... (+versions)

Algorithms: Batsched (version)

- ZMQ, RapidJSON, Boost, Redis, GMP... (+versions)

Execution manager: Ad hoc Python script (version)

- Python interpreter, standard lib, PyPI libs (+versions)

# Execute1 Overview

Input:

- Batsim command (string)
- Scheduler command (string)
- Timeouts (float[])

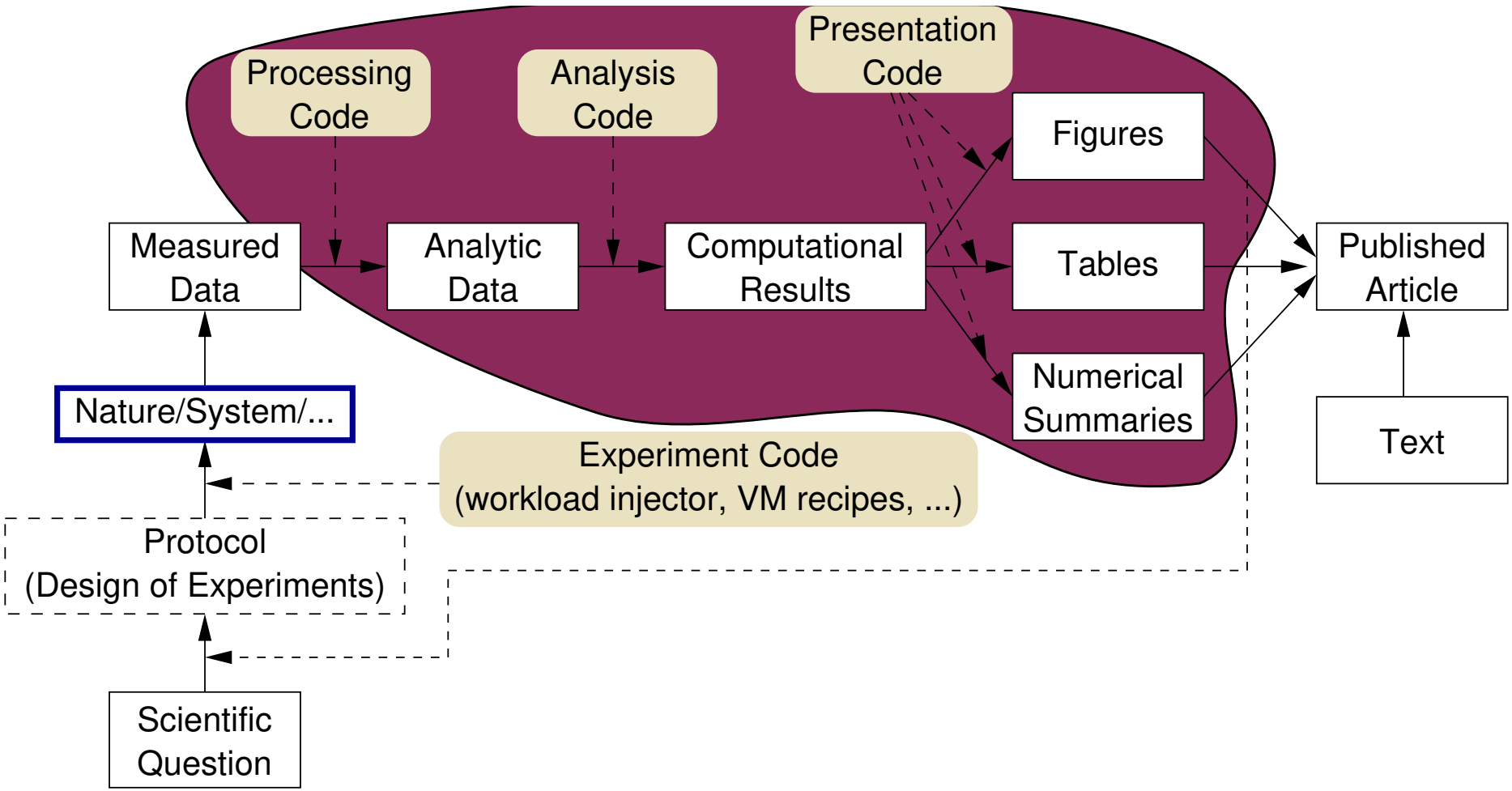
Output:

- Batsim output (CSV files)
- Scheduler internal traces (CSV files)

**Should prevent process side effect.**

- Processes may crash
- A timeout may be reached (infinite loop, deadlock...)

# Data analysis



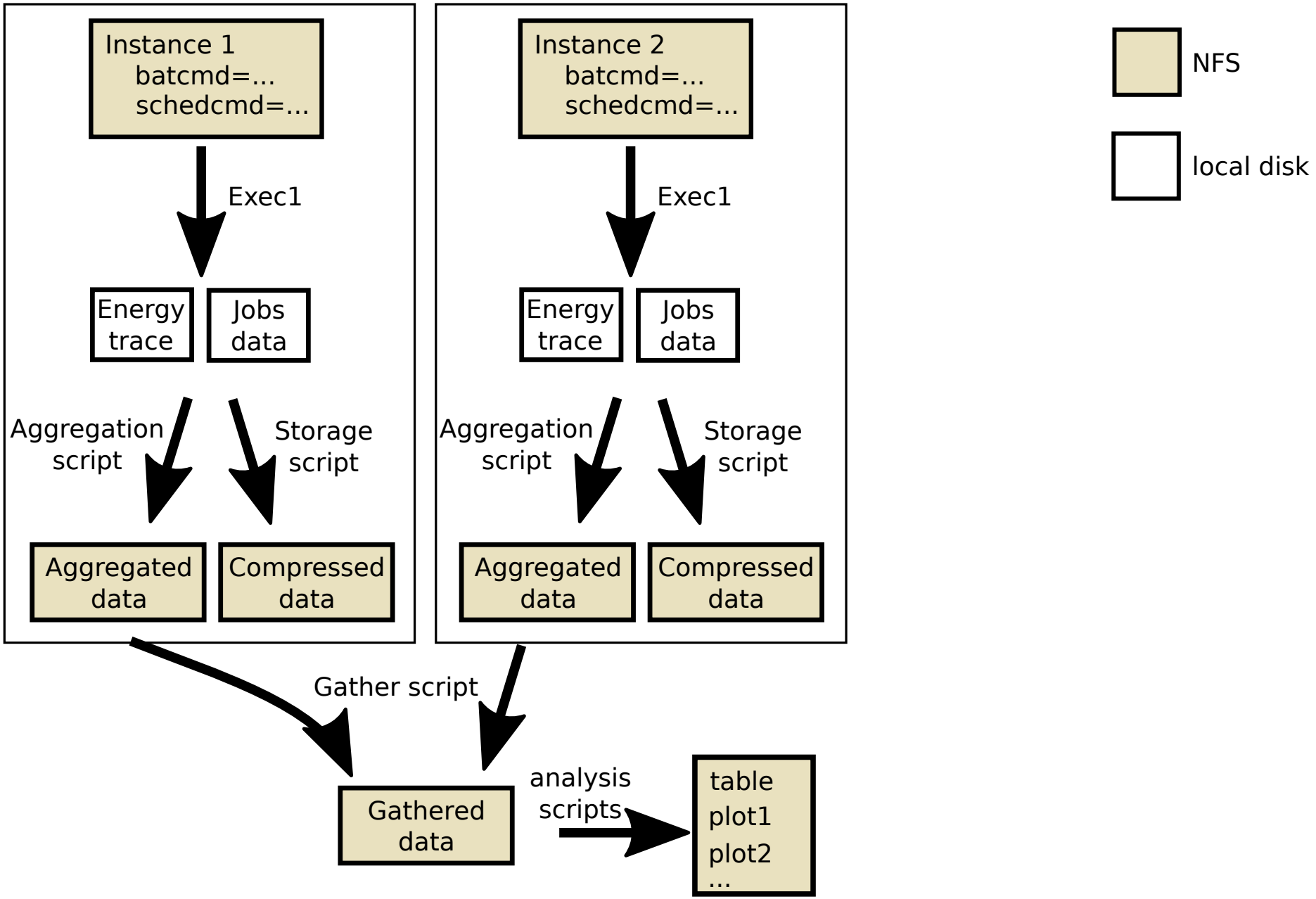
# Data analysis

Usual desires.

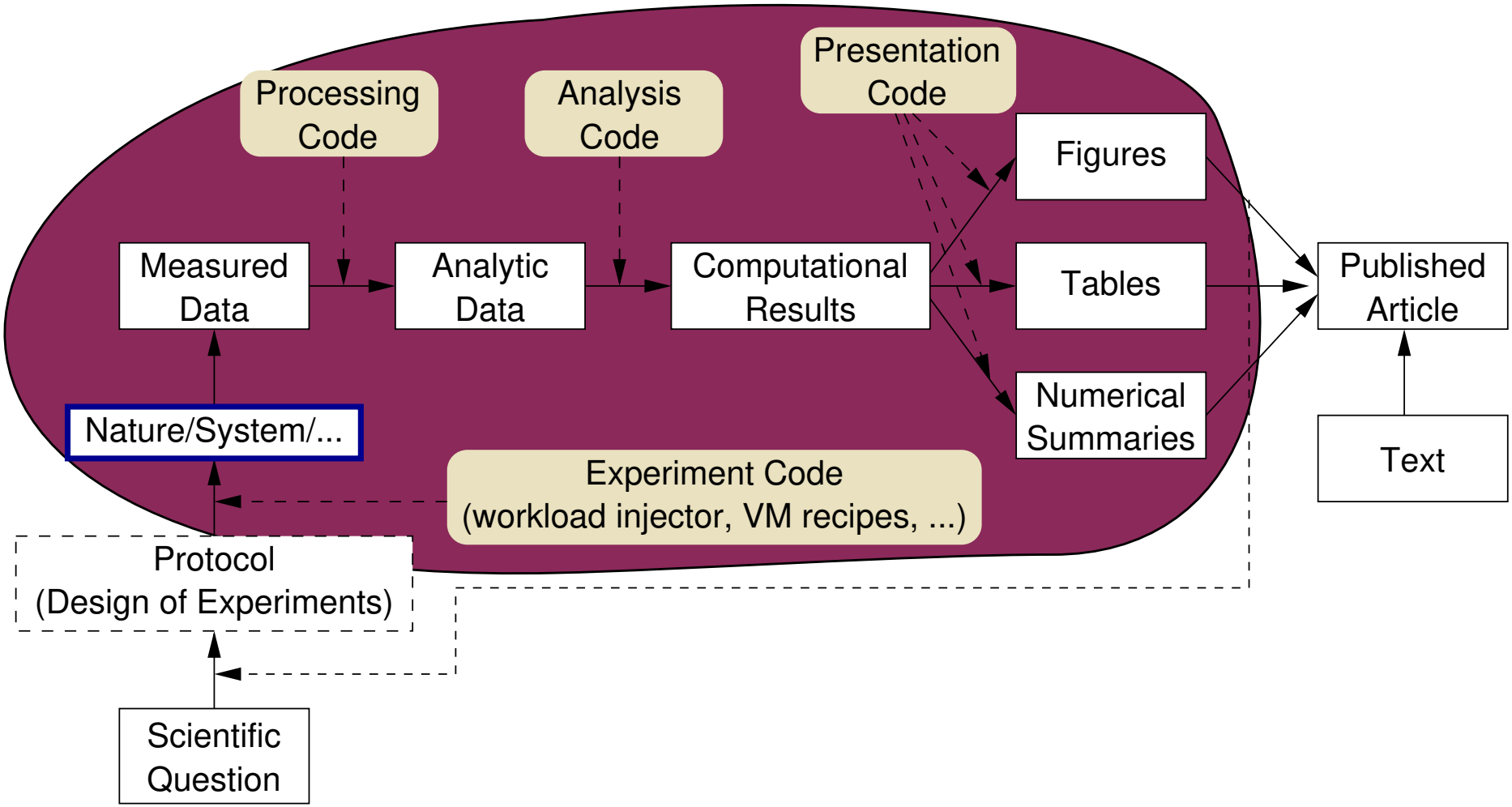
- Modifiability (easy to change/create a plot)
- Language-agnostic (R for plots, Python for algos...)

In our case.

- Simulation output $\simeq$ 320 Go $\rightarrow$ (in situ) aggregation


Instance 1
batcmd=...
schedcmd=...
Exec1
Energy trace
Jobs data
Aggregation script
Storage script
Aggregated data
Compressed data
Instance 2
batcmd=...
schedcmd=...
Exec1
Energy trace
Jobs data
Aggregation script
Storage script
Aggregated data
Compressed data
Gather script
Gathered data
analysis scripts
table
plot1
plot2
...
NFS
local disk

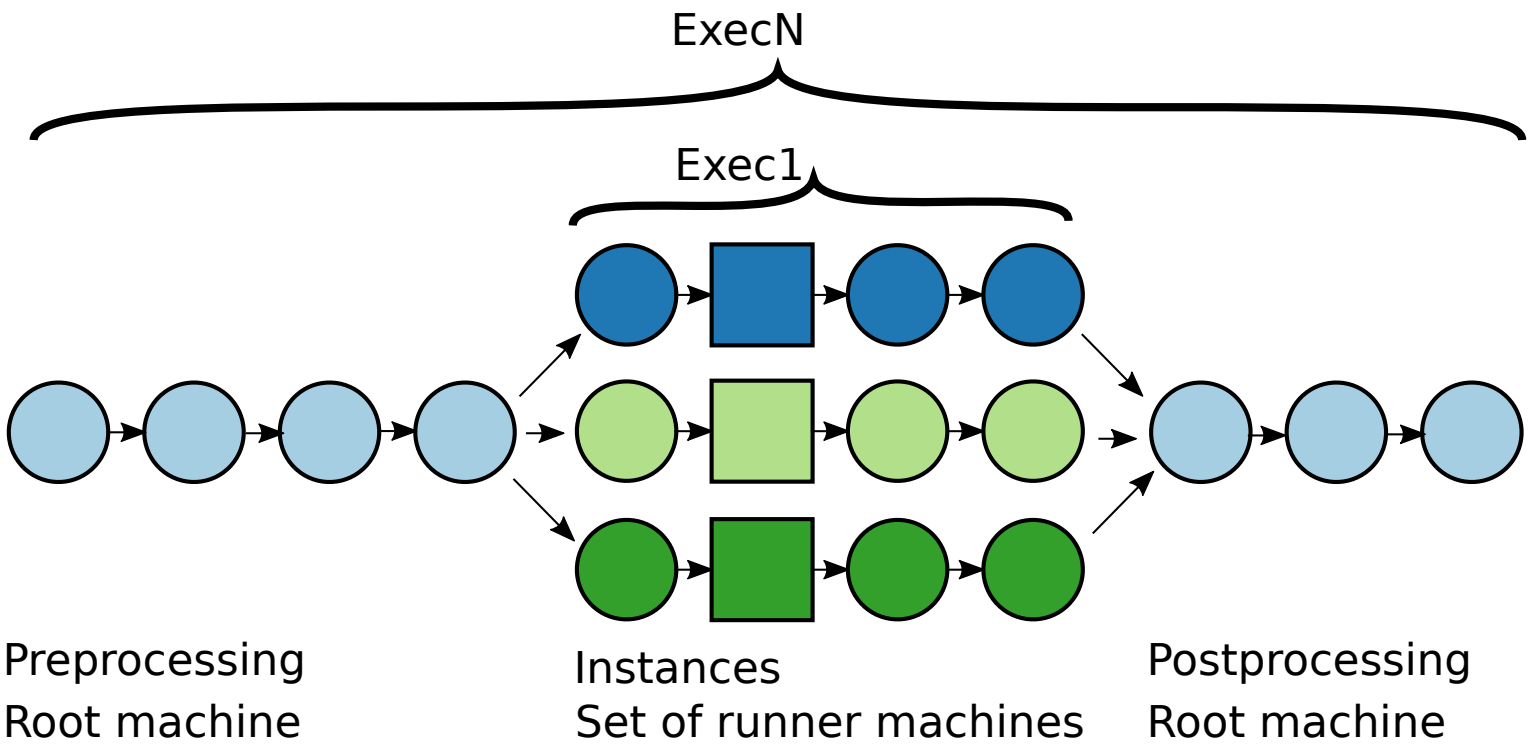
# Monolithic Approach



# Monolithic Approach

One file to rule them all.

1. Get / generate simulation inputs (platforms, workloads...)
2. Generate simulation instances (union of cartesian products)
3. Run instances + local computations
4. Gather all data
5. Analyze gathered data



# Exec* scripts

Goal

- Batsim specific
- Genericity: Experiments and tests

Technology history

- 1: Execo (instance db, process management, remote exec...)
- 2: Execo (instance db) + instance db hacks + py3
- 3: 2 + coroutines

Current status

- Unmaintained.
- subset(exec1) → robin

# Discussion

Conclusion

- Reproducible? OK.
- Convenient local/cluster run? OK.
- Caching system? Yes + manual.
- Easy to update params/scripts? Yes.
- Easy to update software env? =/
- Maintainable? =/

Alternatives?

- Nix/Guix (example)
- CWL, Swift