SimGrid and Batsim Overview

Millian Poquet

2021-03-16

Study distributed systems and applications

Many distributed systems in use today (HPC, Clouds...) and tomorrow (Edge, Fog?)

Complex platforms with many issues (energy, fault tolerance, scheduling, scalability, heterogeneity...)

Methodological approaches

- Direct experimentation (real applications on real platforms)
- Simulation (application prototypes on platforms models)
- Something in between (emulation, partial simulation...)

Building simulators from scratch is risky

How useful is a simulator whose results cannot be trusted?

- Models validated?
- Implementation tested?
- Model instantiation evaluated?

Doing it thoroughly may take (dozens of) years!

Using SimGrid (or any validated simulation frameworks) helps a lot

- Thoroughly validated models
- Thoroughly tested implementation
- Model instantiation responsibility is still on you

Overview

Simulation framework around distributed platforms and applications

Main use cases

- Develop digital twins of distributed applications
- Evaluate various platform topologies/configurations
- Prototype systems or algorithms

Key features

- Sound/accurate models: theoretically and experimentally evaluated
- Scalable: fast models and implementations
- Usable: LGPL, linux/mac/windows, C++ Python and Java

Overview (2)

Numbers

- Exists since early 2001, development still very active
- $\approx$ 200k lines of C/C++ code
- $\approx$ 32k commits
- Used in at least 532 scientific articles

Community

- 4 main developers
- Many power users (current/previous PhD. students...)
- Get help easily (documentation, mailing list, irc, mattermost...)

Architecture

How to build your simulator?

- Use one of the SimGrid interfaces
- Link the SimGrid library with your code

Available interfaces

- SMPI: `smpicc`/`smpirun` on your real MPI code
- S4U: write your own simulator (actors, messages), C++ C or Python
- MSG: older brother of S4U, C or Java
- MC: verify properties on your application *model* (model is code)

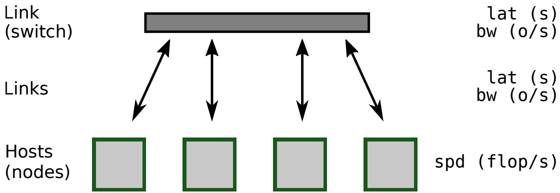
Platform and network models

Platform = graph of hosts and links

Hosts : computational resources
- Speed (FLOP per second)

Links : network resources (cables, switches, routers...)
- Latency (seconds)
- Bandwidth (bytes per second)

Several network models available
- Fast flow-level: slow start, TCP congestion, cross-traffic
- Constant time: a bit faster (unrealistic)
- Packet-level: NS-3 binding



Actors, computations and communications

Actors

- One of the simulation *actors* — AKA agent, thread, process...
- Executes user-given code on a Host
- User-given code may contain SimGrid calls

Main SimGrid calls

- Compute x flops on current host
- Send x bytes to an actor/host/mailbox
- Yield (just interrupt control flow)

S4U simulator example (Python)

```python
from simgrid import Actor, Engine, Host, this_actor

def sleeper():
    this_actor.info("Sleeper started")
    this_actor.sleep_for(1)
    this_actor.info("I'm done. See you!")

def master():
    this_actor.execute(64)
    actor = Actor.create("sleeper", Host.current(), sleeper)
    this_actor.info("Join sleeper (timeout 2)")
    actor.join(2)

if __name__ == '__main__':
    e = Engine(sys.argv)
    e.load_platform(sys.argv[1])
    Actor.create("master", Host.by_name("Tremblay"), master)
    e.run()
```

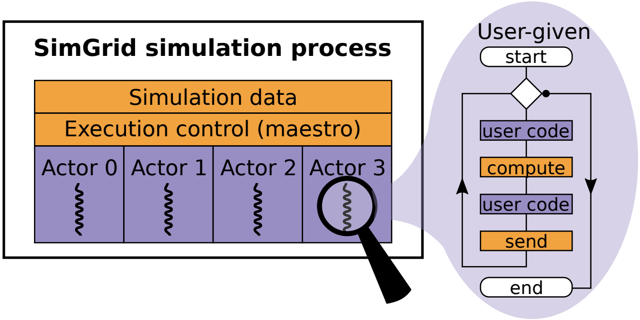
Actor execution model

Main points

- mutual exclusion on actors
- *maestro* dictates who run (deterministic)
- SG calls ≈ syscalls
  - interruption points inside user-given functions

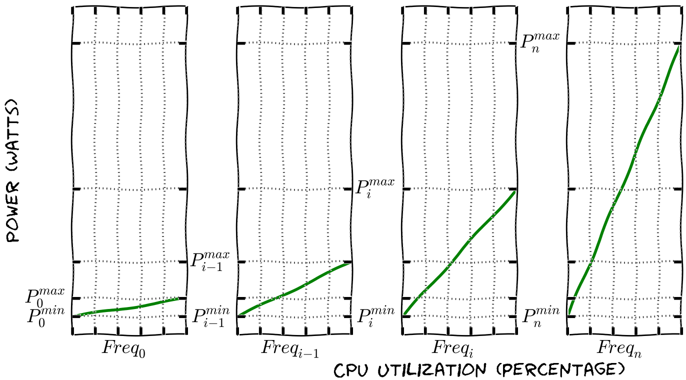
Various implementations

- pthread: *easy* debug, slow
- asm: blazing fast
- ucontext, boost context...



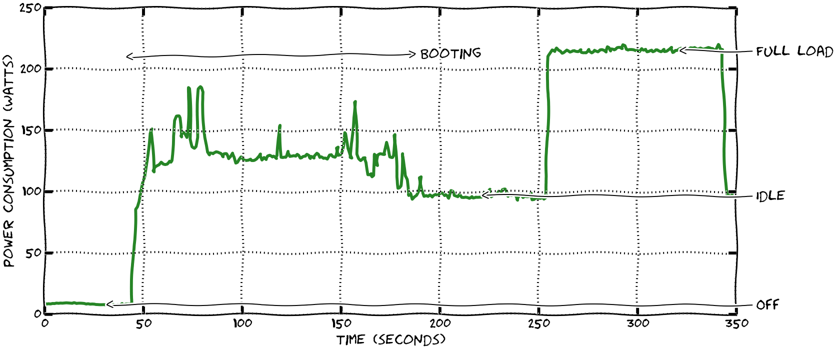
Energy model (DVFS)

- Modern CPUs can reduce computation speed to save energy
- Power states: levels of performance. *Governors* pick them
- SimGrid: Manually switch pstates, which change the flop rate
- For one pstate, consumption = linear function of CPU use



Energy model (ON/OFF)

ON $\leftrightarrow$ OFF takes time (seconds) and energy (Joules)



- Not easy for the noise: everybody wants something specific
- SimGrid provides basic mechanisms, you have to help yourself
- Switching ON/OFF is instantaneous

Real usage example: StarPU's digital twin

StarPU

- Task programming library/runtime for hybrid architectures
- Input: graph of tasks (using StarPU library or OpenMP)
- Input: CPU/GPU/both implementation for each task
- Executes your application with optimized scheduling

How to do this digital twin?

- Copy/paste StarPU's code
- Use SimGrid actors and computations/communications calls
  (working prototype within a few days)
- (Do some optimizations — e.g., replace real code by performance models)

How is it used?

- Test/tune performance of scheduling algo/parameters on many simulated platforms
- Scalability tests at low energy footprint

Overview

Resource management simulator built on top of SimGrid

Main use cases

- Analyze and compare online scheduling algorithms
- Workload/platform dimensioning

Key features

- Prototype scheduling algorithms in any programming language
- Or use real schedulers (done on OAR and K8s, prototypes for flux/slurm)
- Several job models (tunable level of realism) without deep SimGrid knowledge

Overview (2)

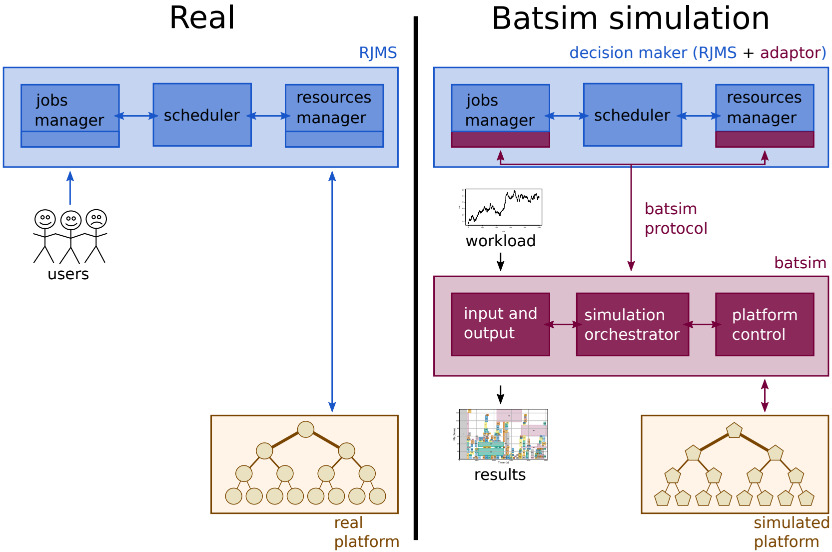
Numbers

- Exists since 2015
- $\approx$ 9k lines of C++ code
- $\approx$ 2k commits

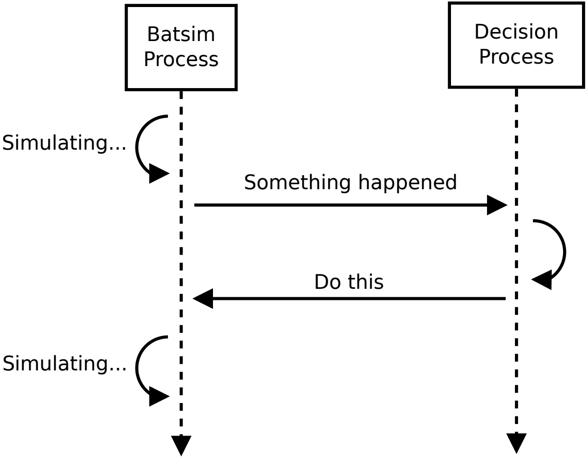
Community

- 1-2 main developers at the same time
- Mostly used by PhD. students/interns from scientific labs so far
- Get help easily (documentation, mailing list, mattermost)

Architecture



Protocol



Classical scheduling events

- Job submitted
- Job finished

Resource management decisions

- Execute job j on $M = \{1, 2\}$
- Shutdown $M = \{3, \ldots, 5\}$

Simulation/monitoring control

- Call scheduler at $t = 120$
- How much energy used?
- How much data moved?

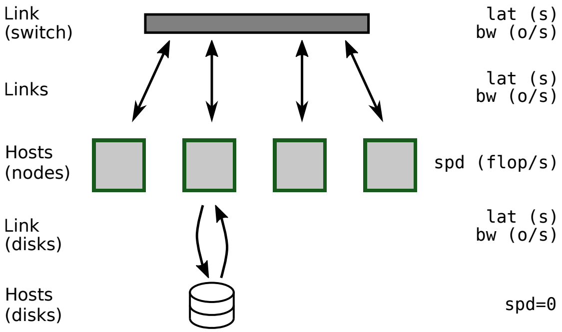
Platform

SimGrid platform + some sugar

RJMS internals on *master* host

Disks modeled as speed=0 hosts

- Enables parallel task use



Jobs and profiles

Jobs : scheduler view

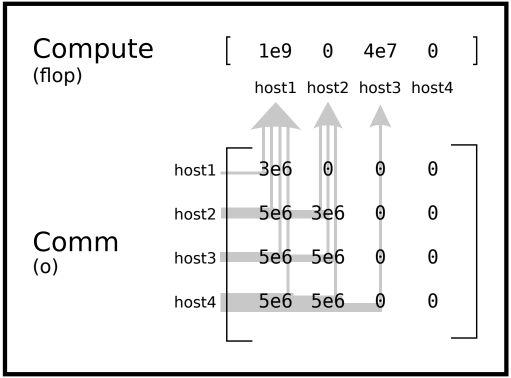
- User resource request
- (Walltime)
- Simulation profile

Profiles : simulator view

- How to simulate the app?

Profile types

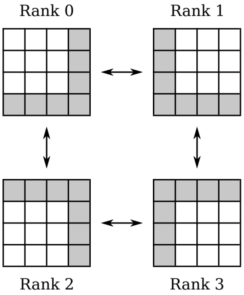
- Fixed length
- Parallel task
- MPI trace replay
- Sequence
- Convenient shortcuts
    - IO transfers (alone)
    - IO transfers (along- task)



Sequence

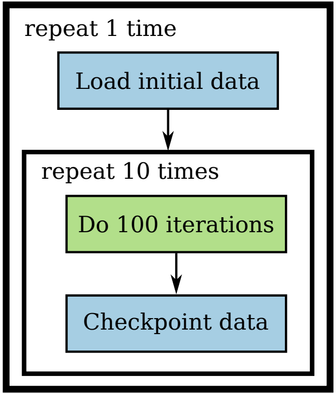
Application model example: Stencil with checkpoints

1. Loads data from parallel filesystem
2. Iteration: local computations, exchange data with neighbors
3. Every 100 iterations: dump checkpoint on parallel file system
4. Stop after 1000 iterations.



Profile example

- Bundle 100 iterations in 1 parallel task



Application model example: Stencil with checkpoints (code)

```
{ "initial_load": {
    "type": "parallel_homogeneous_pfs",
    "bytes_to_read": 67108864,
    "bytes_to_write": 0,
    "storage": "pfs" },
  "100_iterations": {
    "type": "parallel",
    "cpu": [   1e9,    1e9,    1e9,    1e9],
    "com": [     0, 819200, 819200,      0,
            819200,      0,      0, 819200,
            819200,      0,      0, 819200,
                 0, 819200, 819200,      0] },
  "checkpoint": {
    "type": "parallel_homogeneous_pfs",
    "bytes_to_read": 0,
    "bytes_to_write": 67108864,
    "storage": "pfs" },
  "iterations_and_checkpoints": {
    "type": "composed",
    "repeat": 10,
    "seq": ["100_iterations", "checkpoint"] },
  "imaginary_stencil": {
    "type": "composed",
    "repeat": 1,
    "seq": ["initial_load", "iterations_and_checkpoints"] }
}
```

Ecosystem and Usage

Ecosystem

- Set of scheduling algorithms (C++, Python, Rust, D, Perl...)
- Tools to generate platforms and workloads
- (Interactive) tools to visualize/analyze Batsim results
- Tools to help experiments (environment control, execution...)

Already used to study

- Energy/temperature related scheduling heuristics
- Big data / HPC convergence (best effort Spark jobs within HPC cluster) with distributed file system (HDFS)
- Evolving jobs with parallel file system + burst buffers

Conclusion

Take home message

- Simulation is a precious tool to study distributed systems/applications
- SimGrid: 20 years of model (in)validation and optimizations
- Give SimGrid a try, it may save you a lot of time
- Batsim: Specialized SimGrid use around resource management

Thanks! :)

- `millian.poquet@inria.fr`
- `https://framateam.org/simgrid/channels/town-square`
- `https://framateam.org/batsim/channels/town-square`