Improving Job Scheduling by using Machine Learning

&

Yet an another SLURM simulator

David Glesser, Yiannis Georgiou (BULL)
Denis Trystram(LIG)

Improving Job Scheduling by using Machine Learning

&

Yet an another SLURM simulator

Improving Job Scheduling by using Machine Learning

Improving Backfilling by using Machine Learning to Predict Running Times

- By *Eric Gaussier, David Glesser, Valentin Reis, Denis Trystram*
- In proceedings of SuperComputing 2015

Improving Job Scheduling by using Machine Learning

- Machine Learning algorithms can learn odd patterns
- SLURM uses a backfilling algorithm
- the running time given by the user is used for scheduling, as the actual running time is not known
- The value used is very important

- better running time estimation => better performances

► Predict the running time to improve the scheduling

Improving Job Scheduling by using Machine Learning

- We select a Machine Learning algorithm that:

- Use classic job parameters as input parameters
- Work online (to adapt to new behaviors)
- Use past knowledge of each user (as each user has its own behaviour)
- Robust to noise (parameters are given by humans, jobs can segfault...)

Improving Job Scheduling by using Machine Learning

- We test 128 different algorithms on 6 logs (from the Feitelson Workload Archive) on the Pyss simulator

- A leave-one-out cross validation product give us the best algo that we called *E-Loss*:
  - Online linear regression model
  - Predict that a running time is more than the actual value cost more to the model
  - When we under estimate a running time, we add a fixed value (1min, 5min, 15 min, 30 min…)
  - When we backfill jobs we sort them by shortest first

Improving Job Scheduling by using Machine Learning

- Backfilling performances can be improved by changing running times
- More precise running times does not mean better performances
- Scheduling performances can be increased using basic Machine Learning algorithms

Improving Job Scheduling by using Machine Learning

Ongoing works

- Impelement *E-Loss* in SLURM
- We need a simulator within SLURM to test it
  - Machine Learning algorithms perform best when they have a lot of data to learn from

- Instead of customizing the priority factors by hand, a Machine Learning can do it for you!

Improving Job Scheduling by using Machine Learning

&

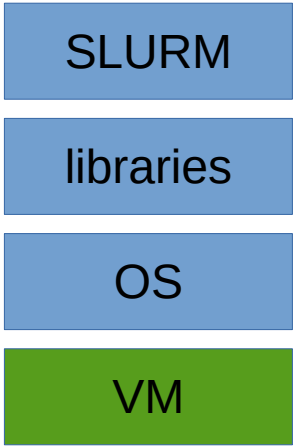
Yet an another SLURM simulator

Yet an another SLURM simulator

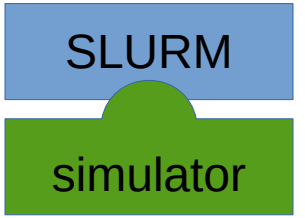
- Previous work: run *sleep*s instead of actual jobs, multiple slurmd per physical node (to emulate bigger cluster than you have access to)

- Why not advancing in time when everybody sleeps?
  - ► Use simulation!

Yet an another SLURM simulator



Virtual Machines
+ perfect behaviour
- heavy and slow
+ No modifications to SLURM



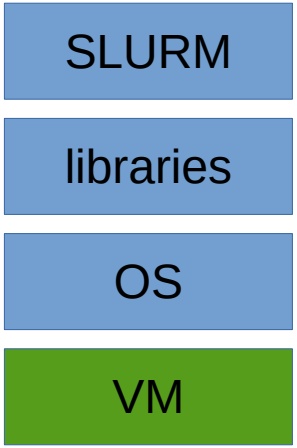
Classic simulators
- no guarantee on the behaviour
+ extra light
- Modifications of SLURM

Yet an another SLURM simulator

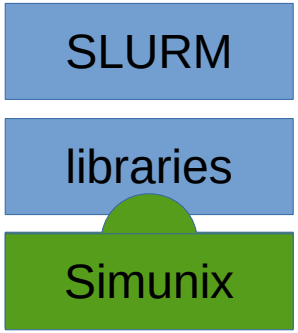
Introducing Simunix, an UNIX simulator

- We implement the "UNIX" API: pthreads, pthread_mutex, gettimeofday, sleep, send, recv…
- Use Simgrid framework
  - ► We can run an unmodified slurm on a simulated cluster

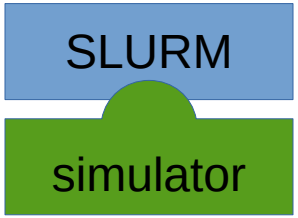
Yet an another SLURM simulator



Virtual Machines
+ perfect behaviour
- heavy and slow
+ No modifications to SLURM



Simunix
+ close behaviour
+ light
+ No modifications to SLURM



Classic simulators
- no guarantee on the behaviour
+ extra light
- Modifications of SLURM

Yet an another SLURM simulator

How to force a binary to use our libraries?

- Change how linking is done!

- The Linux linker load from the system and LD_PRELOAD the needed shared libraries
- It fills the GOT (Global Object Table) with the address of each functions of each libraries
- The compiler compile

```
sleep(10);
```

to

```
GOT[“sleep@libc“](10);
```

(Of course, it's not exactly like this, if you have more question RTFM of the ELF format)

Yet an another SLURM simulator

How to force a binary to use our libraries?

- Change how linking is done!

- At runtime, simunix rewrite the GOT
  - Of the selected binary/libraries
  - Not on the simunix library nor the Simgrid library!
  - Addresses in the GOT are replace by our own functions:

```
GOT[“sleep@libc“] = &simunix_sleep;
GOT[“time@libc“] = &simunix_time;
...
```

Yet an another SLURM simulator

Simgrid

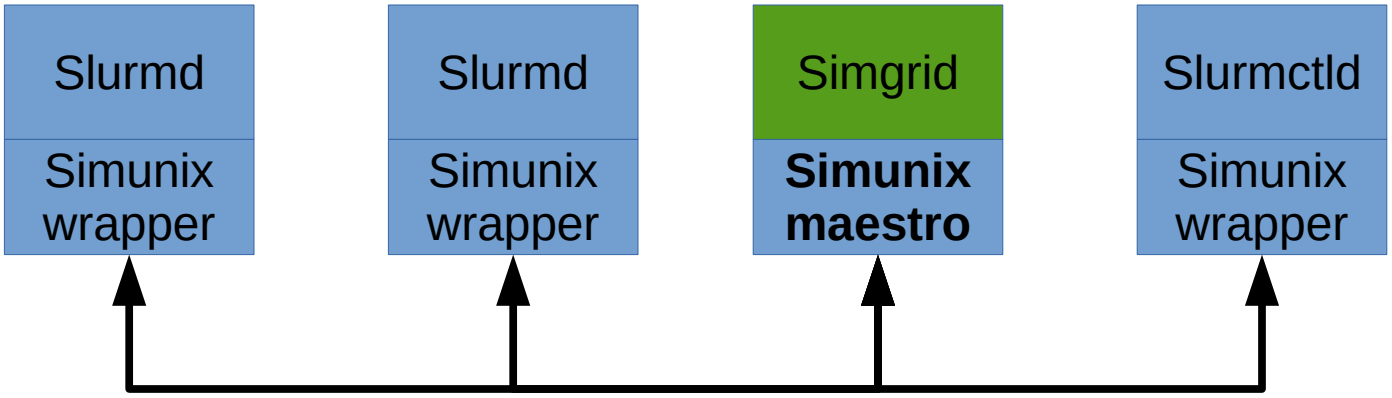
- a framework to design simulators of distributed applications

- Supports:
  - advanced network models
  - energy consumption models
  - I/O models
- Actively developped
- Good practice : they (in)validate their simulator (they explictly give the strengths and weaknesses of their models by testing them and compared them to real runs!)

Yet an another SLURM simulator

How this work?

- Then, each intercepted calls communicate to an independent maestro process



Yet an another SLURM simulator

Current works

- Optimize to simulate 1 year in a reasonable amount of time
- Support more Simgrid features:
  - run simulated apps not just a sleep (network contention…)
  - DVFS and energy
- Try out with other schedulers (every Linux software is compatible!)

Thanks