Improving Backfilling by using Machine Learning to Predict Running Times in SLURM

David Glesser

2015-11-19

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*

- Presented this morning in the Resource Management session

Improving Job Scheduling by using Machine Learning

- Machine Learning algorithms can learn odd patterns
- SLURM use 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:

- Uses classic job parameters as input parameters
- Works online (to adapt to new behaviors)
- Uses past knowledge of each user (as each user has its own behaviour)
- Robusts 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

Table 4: Workload logs used in the simulations.

Name	Year	# CPUs	# Jobs	Duration
KTH-SP2	1996	100	28k	11 Months
CTC-SP2	1996	338	77k	11 Months
SDSC-SP2	2000	128	59k	24 Months
SDSC-BLUE	2003	1,152	243k	32 Months
Curie	2012	80,640	312k	3 Months
Metacentrum	2013	3,356	495k	6 Months

Improving Job Scheduling by using Machine Learning

Log	Our algorithm	Backfill	SoA
KTH-SP2	**51.4** (44%)	92.6	63.5 (31%)
CTC-SP2	**20.5** (59%)	49.6	85.8 (-72%)
SDSC-SP2	**75.0** (15%)	87.9	79.4 (10%)
SDSC-BLUE	34.7 (05%)	36.5	**21.0** (42%)
Curie	**27.9** (86%)	202.1	193.5 (04%)
Metacentrum	**84.2** (14%)	97.6	87.2 (11%)

Results on the avereage Stretch ($\frac{real\ running\ time \times waiting\ time}{real\ running\ time}$)

Improving Job Scheduling by using Machine Learning

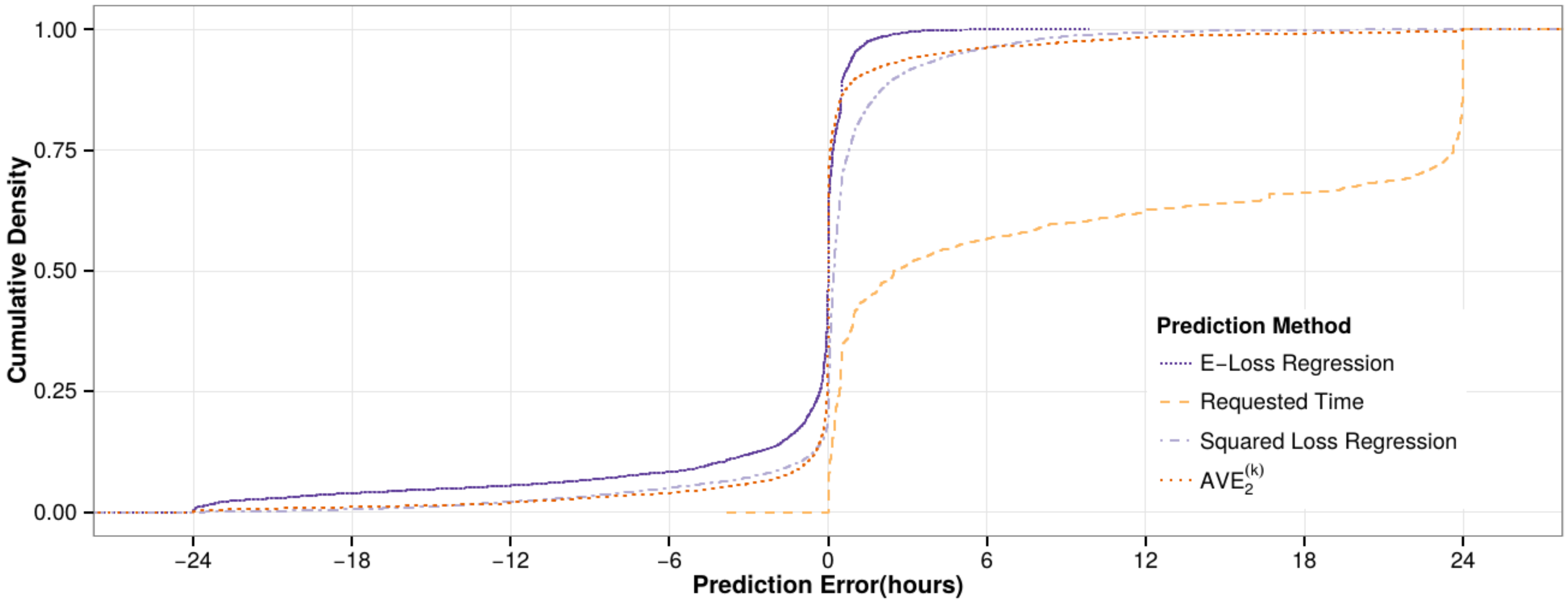


Figure 4: Experimental cumulative distribution functions of prediction errors obtained using the Curie log.

Our algorithm under-estimate more than over-estimate.
This make the backfilling more aggressive (more jobs will be backfilled).

Improving Job Scheduling by using Machine Learning

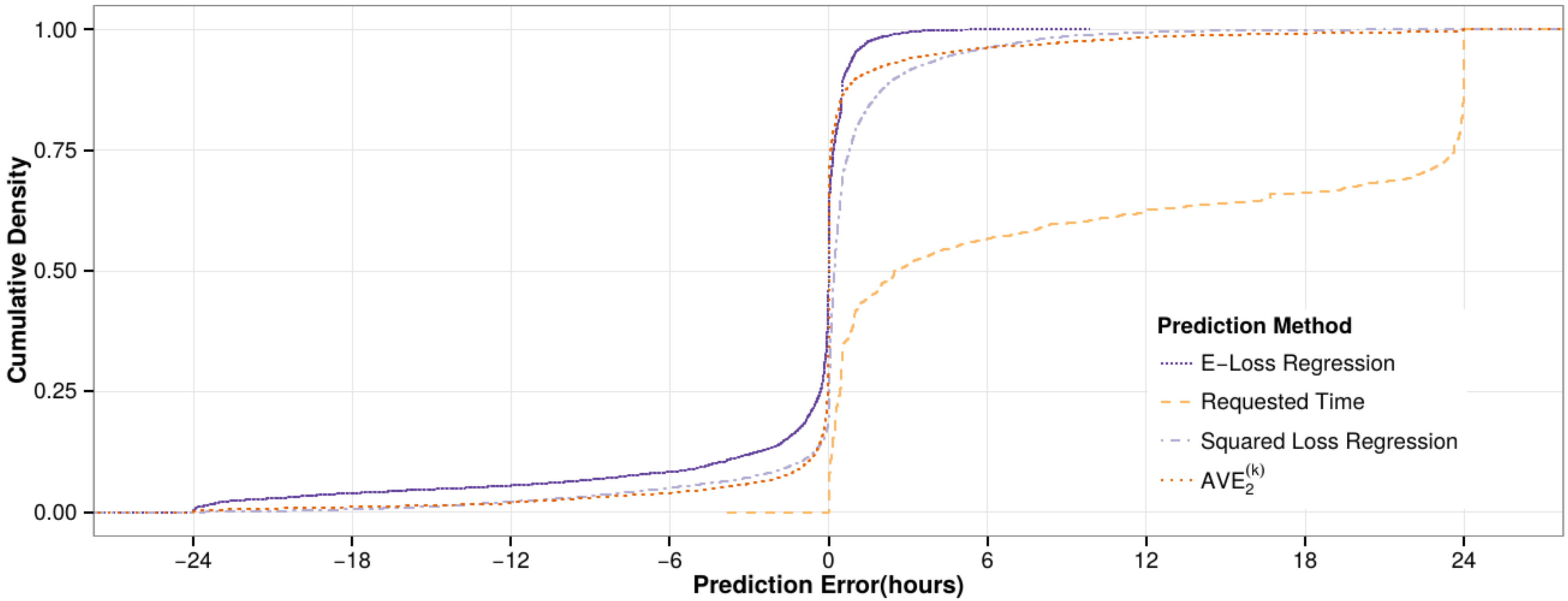


Figure 4: Experimental cumulative distribution functions of prediction errors obtained using the Curie log.

Our algorithm gives the best scheduling performance, but it is not the best at predicting running times !

Improving Job Scheduling by using Machine Learning

Conclusion

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

Implementation in SLURM

- Computation time?
  - O((*#features*)²) for learning and prediction
  - #features=20 in the paper

- No support for time reservations
  - Use of the user estimation for nodes that are reserved in the future

- No estimation of the starting time of the first job
  - Compute an estimation? Don't give it?

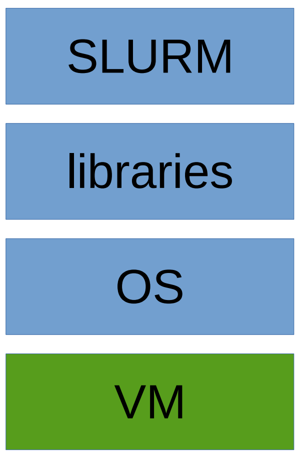
- Impossible to evaluate the implementation
  - Use a Slurm simulator

A Slurm simulator?

Yet an another SLURM simulator

- Previous works:
  - Official Slurm simulator: code is changed, it has to be updated each time a new Slurm is out.
  - Platform emulation: run *sleeps* instead of actual jobs, multiple slurmd per physical node (to emulate bigger cluster than you have access to)

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

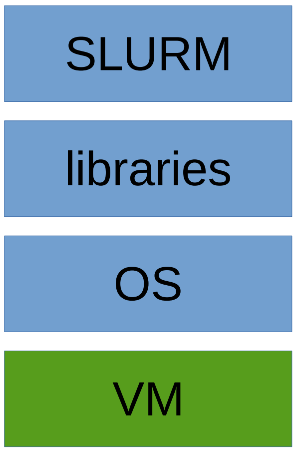
Introducing Simunix, an UNIX simulator

- We implement the "UNIX" API: pthreads, pthread_mutex, gettimeofday, sleep, send, recv…

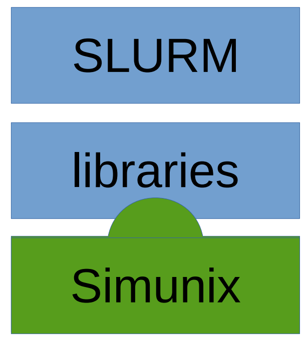
- Use Simgrid framework

  ► 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

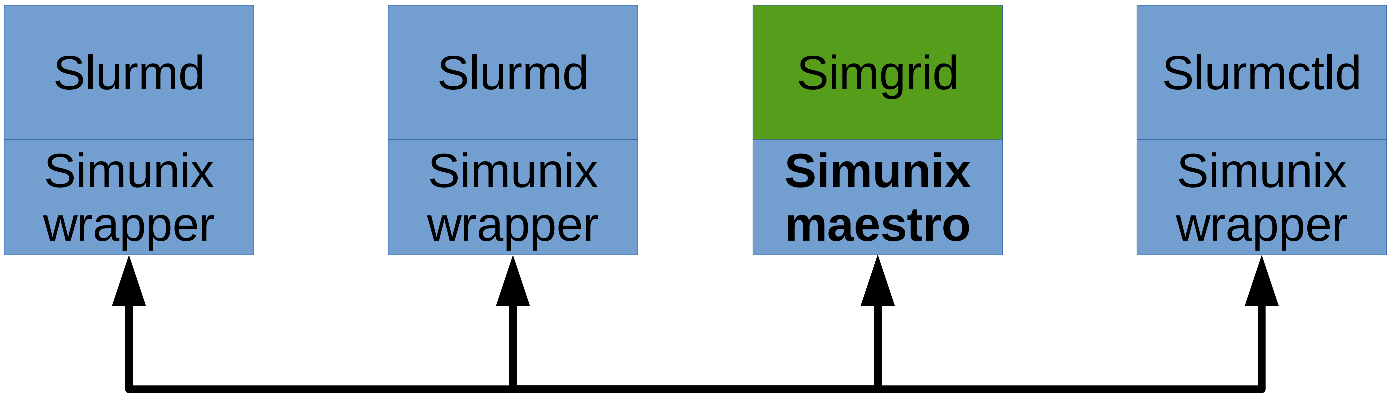
- a framework to design simulators of distributed applications

- Supports:
  - advanced network models
  - energy consumption models
  - I/O models
- Actively developed
- Good practice : they (in)validate their simulator (they explicitly 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?

- 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!)
- Publish!

Improving Job Scheduling by using Machine Learning

Global conclusion

- We can improve the scheduling using machine learning
- Some more works need to be done to support this in Slurm
- Other learning algorithm should also be considered, like Learning2Rank's algorithms

Thanks

2015-05-07