Simunix, a large scale platform simulator

David Glesser

Adrien Faure

2015-11-19
Yet another Slurm simulator

• We want to:
  • Test new developments
  • Test new clusters
  • Test new configurations
  • Develop machine learning algorithms
Yet another slurm simulator

- Previous works to test slurm:
  - Production cluster
  - Small dev cluster
  - Virtual machines
  - Multiple-slurmd (multiple slurm nodes per real node)
  - Slurm simulator
Yet an another slurm simulator

• Previous works to test slurm:
  • Production cluster   impossible!
  • Small dev cluster   too small!
  • Virtual machines   too heavy!
  • Multiple-slurmd   limited! (eg. network contention)
  • Slurm simulator   limited! (eg. the code of slurm is modified)
Our solution

- We simulate the underlying platform, not slurm!
The architecture

- **Slurmd**
  - Simunix interceptor

- **Slurmd**
  - Simunix interceptor

- **Simgrid**
  - Remote simgrid

- **Slurmctld**
  - Simunix interceptor

---

Thrift RPC
The architecture

- **Slurmd**: Simunix interceptor
- **Slurmd**: Simunix interceptor
- **Simgrid**: Remote simgrid
- **Slurmctld**: Simunix interceptor

Thrift RPC
The architecture – interceptors

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

- Then, they traduce UNIX calls to calls to the simulaltor
The architecture – interceptors

How to intercept function calls?
• 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)
The architecture – interceptors

How to intercept function calls?
• 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:

```c
GOT[“sleep@libc“] = &simunix_sleep;
GOT[“time@libc“] = &simunix_time;
...
```
The architecture – Thrift RPC

- *Apache Thrift* is an open-source library to build RPC client and server
- From a custom language, it generates files to build the client and the server
- We use it to transfer calls from the interceptors to the simulator
- *Remmote-simgrid* is our code to use thrift with simgrid
The architecture

- **Slurmd**
  - Simunix interceptor

- **Slurmd**
  - Simunix interceptor

- **Simgrid**
  - Remote simgrid

- **Slurmctlld**
  - Simunix interceptor

---

Thrift RPC
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)
How to start a simulation?
How to start a simulation?

Specific to slurm:
• A slurm installation
• A slurm configuration

Specific to Simgrid:
• A platform XML file (that describe the platform)
• A deployment XML file (that describe who will run where)

Specific to the experiment:
• A script to launch commands (sinfo, srun...)
Demo!
Some experiments already done

- Scalability: we are limited to ~50 nodes

(Bug spotted, fixed soon)
Some experiments already done

- Scalability: we are limited to ~50 nodes
- Run different slurm version (16.05.3
Some experiments already done

- Scalability: we are limited to ~50 nodes
- Run different slurm version (16.05.3, 15.08.9)
Some experiments already done

- Scalability: we are limited to ~50 nodes
- Run different slurm version (16.05.3, 15.08.9 and 2.6.9)
Some experiments already done

- Scalability: we are limited to ~50 nodes
- Run different slurm version (16.05.3, 15.08.9 and 2.6.9)
- Speed: 10 nodes doing nothing for 4 days run in 1m30
Some experiments already done

- Scalability: we are limited to ~50 nodes
- Run different slurm version (16.05.3, 15.08.9 and 2.6.9)
- Speed: 10 nodes doing nothing for 4 days run in 1m30
- Interceptor size: about 20Mo added to each binary
Future work

*Short term future works*

- Remove bugs (scalability, srun)
- Ease of installation (provide a docker)
- Ease of use (4 files to start an experiment?!)
- Support databases
- Publish it (GPL)

*Long term future works*

- How close are we to the reality?
- Optimize
- Intercept /proc and related calls
- Intercept IPMI or RAPL calls (to support energy)
- Use it on other batch schedulers
Thanks