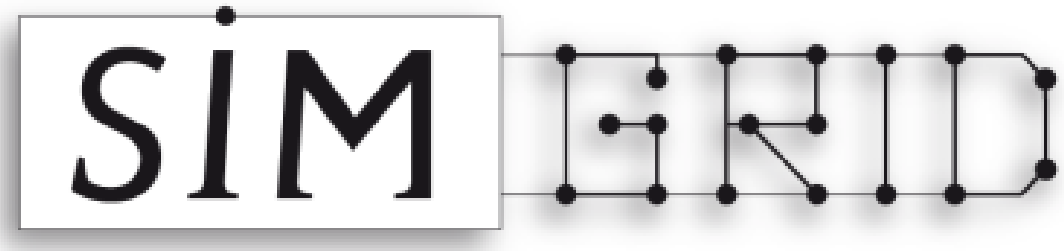
The SimGrid Framework for Research on Large-Scale Distributed Systems - Second Part

Martin Quinson (Nancy University, France)
Arnaud Legrand (CNRS, Grenoble University, France)
Henri Casanova (Hawaii University at Manoa, USA)

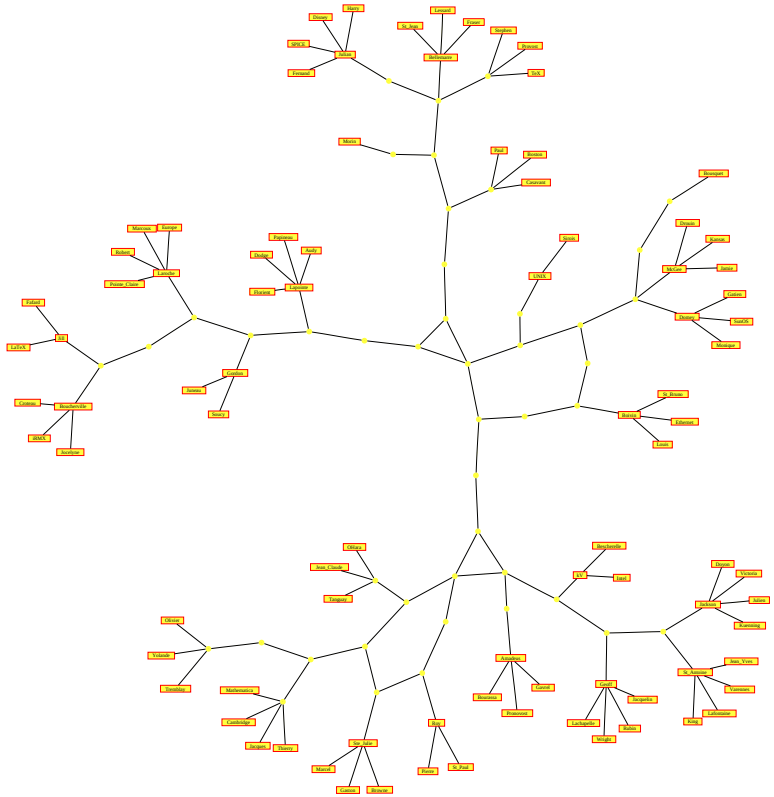
Presented By:
Pedro Velho (Grenoble University, France)

simgrid-dev@gforge.inria.fr



Scheduling Problem

- Heterogeneous platform
- Homogeneous tasks
- Master/Worker approach
- Use SimGrid
  - Model the scheduler
  - Evaluate performance of heuristics



"Given n identical independent tasks that are initially hold by the master, what is the best way to distribute them to the p other computers ?"

SimGrid Installation 1/2

All files and links you need you can find here
`http://mescal.imag.fr/membres/pedro.velho/workshop-simgrid.html`

1. In links section download SimGrid 3.3
   `http://gforge.inria.fr/frs/?group_id=12`
2. Get the file named
   `simgrid-3.3.3.tar.gz`
3. Decode/decompress the tar ball file
   `tar zxvf simgrid-3.3.3.tar.gz`
4. Get to the new created directory
   `cd simgrid-3.3.3`

SimGrid Installation 2/2

1. Configure SimGrid for your architecture, select an installation path
   `./configure --prefix=/simgrid/installation/path`
2. Compile SimGrid
   `make`
3. Install SimGrid
   `make install`
4. Add SimGrid libraries to your library path
   `export LD_LIBRARY_PATH=/simgrid/installation/path/lib`

Any doubt try the SimGrid FAQ

`http://simgrid.gforge.inria.fr/doc/faq.html`

Compiling/Running - Master/Slave Model

1. Download brainless version
   `http://mescal.imag.fr/membres/pedro.velho/src.tgz`
2. Decode/decompress the tar ball file
   `tar zxvf src.tgz`
3. Get to the new created directory
   `cd src`
4. Open Makefile in a text editor, set properly the SimGrid installation path
   `INSTALL_PATH=/simgrid/installation/path`
5. Compile it
   `make`
6. Run it
   `./mslave`

Required Files

- Main file: `mslave.c`
- Master/Slave functions: `sched_struct.c`,`sched_struc.h`
- Makefile: `Makefile`

SimGrid Master/Slave Example (1/2)

- Write the Code of your Agents

```
int master(int argc, char **argv) {
for (i = 0; i < number_of_tasks; i++) {
 t=MSG_task_create(name,comp_size,comm_size,data);
 sprintf(mailbox,"worker-%d",i % workers_count);
 MSG_task_send(t, mailbox);
}
```

```
int worker(int ,char**){
sprintf(my_mailbox,"worker-%d",my_id);
while(1) {
  MSG_task_receive(&task, my_mailbox);
  MSG_task_execute(task);
  MSG_task_destroy(task);
}
```

- Detail your Experiment Platform

XML Platform File

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "surfxml.dtd">
<platform version="2">
<host name="host1" power="1E8"/>
<host name="host2" power="1E8"/>
<link name="link1" bandwidth="1E6"
                   latency="1E-2"  />
<route src="host1" dst="host2">
  <link:ctn id="link1"/>
</route>
...
</platform>
```

XML Deployment File

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "surfxml.dtd">
<platform version="2">
<!-- The master process -->
<process host="host1" function="master">
 <argument value="10"/><!--argv[1]:#tasks-->
 <argument value="1"/><!--argv[2]:#workers-->
</process>
<!-- The workers -->
<process host="host2" function="worker">
  <argument value="0"/></process>
</platform>
```

SimGrid Master/Slave Example (2/2)

- Glue things together

```
int main(int argc, char *argv[ ]) {
  /* Bind agents' name to their function */
  MSG_function_register("master", &master);
  MSG_function_register("worker", &worker);

  MSG_create_environment("my_platform.xml");   /* Load a platform instance */
  MSG_launch_application("my_deployment.xml"); /* Load a deployment file */

  MSG_main(); /* Launch the simulation */

  INFO1("Simulation took %g seconds",MSG_get_clock());
}
```

- Compile your code (linked against `-lsimgrid`), run it and enjoy

Question

In the current implementation, the master starts satisfying requests only when there are enough pending requests.

- Propose and try other request selection strategies:
  - fastest CPU
  - slowest CPU
  - higher Bandwidth
  - grater CPU/Bandwidth ratio
  - Other
- Evaluate different application parameters
  - Number of tasks
  - Task size
  - Other

Useful files

- `test.sh`
- `deployment_generic.xml`
- `deployment_generic_smarter.xml`