Versatile Simulation of Distributed Systems

Frédéric Suter
On behalf of the SimGrid Team

January 23, 2015
**What is Science?**

**Doing Science = Acquiring Knowledge**

- **Experimental Science**
  - Thousand years ago
  - Observations-based
  - Can describe

- **Theoretical Science**
  - Last few centuries
  - Equations-based
  - Can understand

- **Computational Science**
  - Nowadays
  - Compute-intensive
  - Can simulate

\[
\frac{\partial}{\partial x_j} \left( \frac{\partial \Phi}{\partial x_i} \right) = \frac{\partial}{\partial x_i} \left( \frac{\partial \Phi}{\partial x_j} \right)
\]
Use of Computers in Modern Science

High Throughput Computing (HTC)

- Huge amount of data
- Millions of jobs on “cheap” computers
- Higgs Boson discovery

High Performance Computing (HPC)

- Huge parallel simulation
- Finely tuned expensive machine
- Comparison with observations
What is Computer Science (in this context)?

- Deal with huge amount of data
  - Data placement, distribution, replication
  - Provenance, mining

- Heterogeneous distributed infrastructures
  - Job and workflow scheduling
  - Scientific gateways, Cloud

- Tightly coupled parallel applications
  - Performance analysis ~ Optimization
  - Resilience, algorithms

- Massively parallel machines
  - High Performance Network interconnects
  - GPUs, Many-Cores, low-power nodes
SimGrid Scope: Distributed Systems


- *Computational science* infrastructure: Massive / Federated systems
- **Main issues:** Be TOP’500 #1 / compatibility, trust, accountability

Cloud Computing

- Large infrastructures underlying commercial Internet (eBay, Amazon, Google)
- **Main issues:** Optimize costs; Keep up with the load (flash crowds)

P2P Systems

- Exploit resources at network edges (storage, CPU, human presence)
- **Main issues:** Volatility (churn); Network locality; Anonymity

Production systems, with **hard to assess characteristics**

- **Correction:** absence of crash, race conditions, deadlocks, and other defects
- **Performance:** makespan, economics, energy, . . . . ← main focus of SimGrid
Studying Distributed Applications

Correction Study $\sim$ Formal Methods

- **Tests:** Do not provide definitive answers

Performance Study $\sim$ Experimentation

- **Maths:** Often not sufficient to fully understand systems
Studying Distributed Applications

Correction Study $\sim$ Formal Methods

- Tests: Do not provide definitive answers
- Model-Checking: Exhaustive and automated exploration of state space

Performance Study $\sim$ Experimentation

- Maths: Often not sufficient to fully understand systems

- Experimental Facilities: Real applications on Real platform (in vivo)

- Simulation: Prototypes of applications on system’s Models (in silico)
Studying Distributed Applications

Correction Study $\leadsto$ Formal Methods
- **Tests**: Do not provide definitive answers
- **Model-Checking**: Exhaustive and automated exploration of state space

Performance Study $\leadsto$ Experimentation
- **Maths**: Often not sufficient to fully understand systems

- **Experimental Facilities**: Real applications on Real platform (*in vivo*)
- **Emulation**: Real applications on Synthetic platforms (*in vitro*)
- **Simulation**: Prototypes of applications on system’s Models (*in silico*)
Studying Distributed Applications

Correction Study \(\leadsto\) Formal Methods

- **Tests:** Do not provide definitive answers
- **Model-Checking:** Exhaustive and automated exploration of state space

Performance Study \(\leadsto\) Experimentation

- **Maths:** Often not sufficient to fully understand systems

**Claim:** Simulation is both sound and convenient

- Less simplistic than proposed theoretical models
- Easier and faster than experimental platforms

- **Experimental Facilities:** Real applications on Real platform \((in\ vivo)\)
- **Emulation:** Real applications on Synthetic platforms \((in\ vitro)\)
- **Simulation:** Prototypes of applications on system’s Models \((in\ silico)\)
What is Simulation Good for?

Fastest Path from Idea to Results

- Get results from partial implementations
- Run thousands of experiments within a week
- Test the scientific idea without bothering with details

Easiest way to study distributed applications/systems

- Centralization: Distribution is only simulated
- High Reproducibility: Everything is controlled
- Clairvoyance: Observe the hidden behaviors
- What-if analysis: Measure the impact of condition changes
- Eco-Friendly: No resource waste for debug and tests
Computational Science of Distributed Systems?

Requirements for a Scientific Approach

- Reproducible results: read a paper, reproduce the results and improve
- Standard tools that Grad students can learn quickly

Current practice in the field is quite different

- Experimental settings not detailed enough in literature
- Many short-lived simulators; few sound and established tools
  - Grid/Cloud: OptorSim, GridSim, GroudSim, CloudSim, iCanCloud, ...
  - Volunteer Computing: SimBA, EmBOINC, SimBOINC, ...
  - P2P: PeerSim, P2PSim, OverSim, ...
  - HPC: PSINS, LogGOPSIm, BigSim, MPI-SIM, ...
  - ...

SimGrid Team – Versatile Simulation of Distributed Systems
SimGrid: a Versatile Simulation Toolkit

Scientific Instrument and Scientific Object

- Developed for 15 years (personal contribution since 2009)
- Simulate real and abstract programs
- Validated, Scalable, Usable, Modular, Portable
- Comparison of network and middleware performance models
SimGrid History

1998-2001 Baby steps: Factorize some code between PhD students in scheduling
2001-2003 Infancy: CSP and improved models
2003-2008 Teenage: Performance, validity, multi-APIs
2008-2011 Maturation: Scope increase to P2P; visualization
2012-: Taking the world over :)
   ▶ Further scope increase to HPC and Cloud
   ▶ Added methodologies: emulation, verification
   ▶ Mature ecosystem and community

Further details:
- SG 1.0 (UCSD)
- SG 2 (UCSD+Lyon)
- SG 3 (UCSD, Grenoble, Lyon, Nancy,...)
- ODL SG
- ANR USS SimGrid
- ANR INFRA SONGS
- ADT SG
- HEMERA WG
Quick Overview of Internals Organization

User-visible SimGrid Components

- **SimDag**: Framework for DAGs of parallel tasks
- **MSG**: Simple application-level simulator
- **SMPI**: Library to run MPI applications on top of a virtual environment

**XBT**: Grounding features (logging, etc.), data structures (lists, etc.), portability

- **MSG**: heuristics as Concurrent Sequential Processes (Java/Ruby/Lua bindings)
- **SimDag**: heuristics as DAG of (parallel) tasks
- **SMPI**: simulate real applications written using MPI

SimGrid is Strictly Layered internally

- **MSG**: User-friendly syntaxic sugar
- **Simix**: Processes, synchro (SimPOSIX)
- **SURF**: Resources usage interface
- **Models**: Action completion computation
Simulation Validity

SotA: Models in most simulators are either simplistic, wrong or not assessed

- **PeerSim**: discrete time, application as automaton;
- **GridSim**: naive packet level or buggy flow sharing
- **OptorSim, GroudSim**: documented as wrong on heterogeneous platforms

SimGrid provides several network models

- **Fast flow-based** model, toward realism and speed (by default)  
  Accounts for Contention, Slow-start, TCP congestion, Cross-traffic effects
- **Constant time**: A bit faster, but no hope of realism
- **Coordinate-based**: Easier to instantiate in P2P scenarios
- **Packet-level**: NS3 bindings
- **Controlled by command line switches** (exact comparison on a given application)
Max-Min Fairness between Network Flows

Computing the sharing between flows

- **Objective function**: maximize \( \min_{f \in F} (\rho_f) \) [Massoulié & Roberts 2003]
- **Equilibrium**: increasing any \( \rho_f \) decreases a \( \rho'_f \) (with \( \rho_f > \rho'_f \))
- (actually, that’s a simplification of our real objective function)

Efficient Algorithm

1. Search for the bottleneck link \( l \) so that: \( \frac{C_l}{n_l} = \min \left\{ \frac{C_k}{n_k}, \ k \in \mathcal{L} \right\} \)
2. This sets the share of any flow \( f \) on this link: \( \rho_f = \frac{C_l}{n_l} \)
3. Update all \( n_k \) and \( C_k \) to remove these flows; Loop until all \( \rho_f \) are fixed

\[
\begin{align*}
x_1 & \leq \text{Power}_\_\text{CPU}_1 \quad (1a) \\
x_2 + x_3 & \leq \text{Power}_\_\text{CPU}_2 \quad (1b) \\
\rho_1 + \rho_2 & \leq \text{Capacity} \_\text{link}_1 \quad (1c) \\
\rho_1 + \rho_3 & \leq \text{Capacity} \_\text{link}_2 \quad (1d)
\end{align*}
\]
Max-Min Fairness Example

Homogeneous Linear Network

- All links have the same capacity $C$
- Each of them is limiting. Let’s choose link 1

\[
\begin{align*}
C_1 &= C & n_1 &= 2 \\
C_2 &= C & n_2 &= 2 \\
\rho_0 &= \\
\rho_1 &= \\
\rho_2 &= 
\end{align*}
\]
Max-Min Fairness Example

Homogeneous Linear Network

- All links have the same capacity $C$
- Each of them is limiting. Let’s choose link 1
  \[\rho_0 = C/2 \text{ and } \rho_1 = C/2\]

\[
\begin{align*}
C_1 &= C & n_1 &= 2 \\
C_2 &= C & n_2 &= 2 \\
\rho_0 &= C/2 \\
\rho_1 &= C/2 \\
\rho_2 &=
\end{align*}
\]
Max-Min Fairness Example

Homogeneous Linear Network

- All links have the same capacity $C$
- Each of them is limiting. Let’s choose link 1
  \[ \rho_0 = \frac{C}{2} \text{ and } \rho_1 = \frac{C}{2} \]
- Remove flows 0 and 1; Update links’ capacity

\[ C_1 = 0 \quad n_1 = 0 \]
\[ C_2 = \frac{C}{2} \quad n_2 = 1 \]
\[ \rho_0 = \frac{C}{2} \]
\[ \rho_1 = \frac{C}{2} \]
\[ \rho_2 = \]
Max-Min Fairness Example

Homogeneous Linear Network

- All links have the same capacity $C$
- Each of them is limiting. Let’s choose link 1
  - $\rho_0 = C/2$ and $\rho_1 = C/2$
- Remove flows 0 and 1; Update links’ capacity
- Link 2 sets $\rho_1 = C/2$. 
- We are done computing the bandwidths $\rho_f$

SimGrid Implementation is **Efficient**
- Lazy updates, Trace integration, preserving Cache locality
Back on the internals

User’s APIs

Handling of concurrent user’s actions

Computation of resource sharing & actions’ progress

App. spec. as concurrent code

Concurrent processes

Condition variables

SIMIX

Activities

Resource capacities and interconnections specification

SURF

Variables

Resource Capacities

MSG

SMPI

SIMDAG

App. spec. as task graph

435
372
530
530
664
50
245
245
\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ x_n \]
\[ \leq C_{L_2} \]
\[ \leq C_{P_1} \]
\[ \leq C_{L_1} \]
\[ \leq C_{L_m} \]
What do others in grid or cloud simulation?

Naive flow models documented as wrong

<table>
<thead>
<tr>
<th>Setting</th>
<th>Expected Output</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B = 100$</td>
<td>$B = 100$</td>
<td></td>
</tr>
<tr>
<td>$B = 20$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What do others in grid or cloud simulation?

Naive flow models documented as wrong

What do others in grid or cloud simulation?

Naive flow models documented as wrong

What do others in grid or cloud simulation?

Naive flow models documented as wrong

**Setting**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$B = 100$</td>
<td>$B = 100$</td>
<td>$B = 20$</td>
</tr>
</tbody>
</table>

**Expected Output**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$B = 100$</td>
<td>$B = 100$</td>
<td>$B = 20$</td>
</tr>
</tbody>
</table>

**Output**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$B = 100$</td>
<td>$B = 100$</td>
<td>$B = 20$</td>
</tr>
</tbody>
</table>


Validation by general agreement

"Since SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed." – CloudSim, ICPP'09
What do others in grid or cloud simulation?

Naive flow models documented as wrong


Validation by general agreement

“Since SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers, bugs that may compromise the validity of the simulation have been already detected and fixed.” – CloudSim, ICPP’09

Buggy flow model (GS 5.2, 11/2010). Similar issues with naive packet-level models
SimGrid Network Model for MPI

Measurements

Hybrid Model

Asynchronous ($k \leq S_a$)    Detached ($S_a < k \leq S_d$)    Synchronous ($k > S_d$)

Fluid model: account for contention and network topology

SimGrid Team – Versatile Simulation of Distributed Systems
And this actually works!

- **Sweep3D**: Simple Application (but not trivial) predicted in all details
- Graphene (16 procs), OpenMPI, TCP, Gigabit Ethernet achieved without overfitting :)

SimGrid Team – Versatile Simulation of Distributed Systems
Reality often . . . surprizing

TCP collapse

- NAS CG on Graphene, with 128 processes
- Highly congested → TCP reduce the emissions
- When speed reaches 0, it timeouts after 200ms, resets, and start over
- (TCP_RTO should help alleviating this bug – but doesn’t)

We could model these effects

- but actually, you want to fix reality
- We wanted to understand the systems with models, what a success!
SimGrid Scalability

Simulation Versatility should not hinder Scalability

- Two aspects: Big enough (large platforms) ⊕ Fast enough (large workload)

Versatile yet Scalable Platform Descriptions

- Hierarchical organization in ASes
  - cuts down complexity
  - recursive routing
- Efficient on each classical structures
  - Flat, Floyd, Star, Coordinate-based
- Allow bypass at any level
- Grid’5000 platform in 22KiB
  - (10 sites, 40 clusters, 1,500 nodes)
- King’s dataset in 290KiB
  - (2,500 nodes, coordinate-based)
Comparison to GridSim

A master distributes 500,000 fixed size jobs to 2,000 workers (round robin)

<table>
<thead>
<tr>
<th></th>
<th>GridSim</th>
<th>SimGrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network model</td>
<td>delay-based model</td>
<td>flow model</td>
</tr>
<tr>
<td>Topology</td>
<td>none</td>
<td>Grid5000</td>
</tr>
<tr>
<td>Time</td>
<td>1h</td>
<td>14s</td>
</tr>
<tr>
<td>Memory</td>
<td>4.4GB</td>
<td>165MB</td>
</tr>
</tbody>
</table>

Volunteer Computing settings

- Loosely coupled scenario as in Boinc
- SimGrid: full modeling (clients and servers), precise network model
- SimBA: Servers only, decisions based on simplistic markov modeling

SimGrid shown 25 times faster
How big and how fast? (2/3 – P2P)

- Scenario: Initialize Chord, and simulate 1,000 seconds of protocol
- Arbitrary Time Limit: 12 hours (kill simulation afterward)

### Largest simulated scenario

<table>
<thead>
<tr>
<th>Simulator</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oversim (OMNeT++)</td>
<td>10k</td>
<td>1h40</td>
</tr>
<tr>
<td>Oversim (simple)</td>
<td>300k</td>
<td>10h</td>
</tr>
<tr>
<td>PeerSim</td>
<td>100k</td>
<td>4h36</td>
</tr>
<tr>
<td>SimGrid (flow-based)</td>
<td>10k</td>
<td>130s</td>
</tr>
<tr>
<td>SimGrid (delay-based)</td>
<td>300k</td>
<td>32mn</td>
</tr>
<tr>
<td>SimGrid (flow-based)</td>
<td>2M*</td>
<td>6h23</td>
</tr>
<tr>
<td>SimGrid (delay-based)</td>
<td>2M</td>
<td>5h30</td>
</tr>
</tbody>
</table>

* 36GB = 18kB/ process (16kB for the stack)

- Orders of magnitude more scalable than state-of-the-art P2P simulators
- Precise model incurs a ≈ 20% slowdown, but accuracy is not comparable
- Also, parallel simulation (faster simulation at scale); Distributed sim. ongoing

SimGrid Team – Versatile Simulation of Distributed Systems
Simulating a binomial broadcast

Model:
- SimGrid: contention + cabinets hierarchy
- \textsc{LogGoPSim}: simple delay-based model

Results:
- SimGrid is roughly 75% slower
- SimGrid is about 20% more fat (15GB required for $2^{23}$ processors)

Genericity of SimGrid data structures \Rightarrow slight overhead

**BUT**

Scalability \nRightarrow loss of realism
What about workflows?

SimDag is the API of choice

1. Describe DAGs
2. Describe resources
3. Write scheduling heuristics
What about workflows?

SimDag is the API of choice

1. Describe DAGs (possibly in the DAX format)
2. Describe resources
3. Write scheduling heuristics

For each task do
Select resource
Schedule task
end do

Workflow
Scheduling
Simulator
What about workflows?

X workflows

Y platforms

Z Heuristics

For each task do
  Select resource
  Schedule task
  end do

SimDag is the API of choice

1. Describe DAGs (possibly in the DAX format)
2. Describe resources
3. Write scheduling heuristics

DAX format support

- Actually DAXes from the workflow archive
  - Jobs include a runtime attribute
  - Do not reflect the executable workflow
  - Lacks of support of the auxiliary info
- Easy to use: sd_daxload("my_dax_file.xml")
  - Return a dynamic array of SimDag tasks
Visualizing SimGrid Simulations

- Visualization scriptable: easy but powerful configuration; Scalable tools
- Right Information: both platform and applicative visualizations
- Right Representation: gantt charts, spatial representations, tree-graphs
- Easy navigation in space and time: selection, aggregation, animation
- Easy trace comparison: Trace diffing (still partial ATM)
Dynamic Verification with SimGrid

Verifying safety and liveness properties

- Works on real C code, using Dwarf to introspect state
- Explicitely explores the execution graph
- DPOR-based reduction techniques (safety only) or State equality reduction
- Mostly suited for bug finding (no certification)

Current state

- Usable in MSG and SMPI (in C)
- Found *wild* bugs in medium-sized programs (Chord protocol)
- Verified collectives of MPICH (in minutes)

Ongoing work

- Verify larger applications
- Ensure send determinism (for checkpointing)
Practical Trust onto SimGrid?

- Internal code base rather complex because of hacks for versatile efficiency

Continuous Integration

- Current version tested every night
- 450 integration tests; 10,000 unit tests; 70% coverage
- 2 SimGrid configurations on 10 Linux versions
- Performance regression testing soon operational

Release tests

- Windows and Mac considered as additional release goals
- Actually works on all Debian arch.: hurd, kfreebsd, mips, arm, ppc, s390

This is free software anyway

- The code base is currently LGPL (probably soon GPL)
- Come, check it out and participate! (5 of 25 committers not affiliated to us)
**Take Away Messages**

SimGrid will prove helpful to your research

- **Versatile:** Used in several communities (scheduling, Grids, HPC, P2P, Clouds)
- **Accurate:** Model limits known thanks to validation studies
- **Sound:** Easy to use, extensible, fast to execute, scalable to death, well tested
- **Open:** LGPL; User-community much larger than contributors group
- **Around since 15 years, and ready for at least 15 more years**

Welcome to the Age of (Sound) Computational Science

- **Discover:** [http://simgrid.org/](http://simgrid.org/)
- **Learn:** several 101 tutorials, user manual, and examples
- **Join:** user mailing list, #simgrid on irc.debian.org
  We even have some open positions ;)

SimGrid Team – Versatile Simulation of Distributed Systems