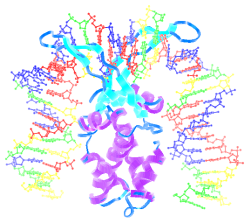
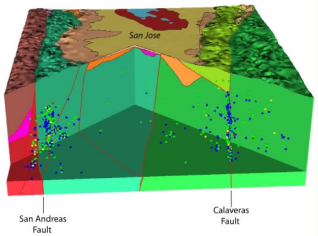
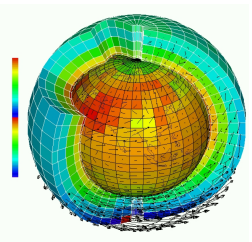
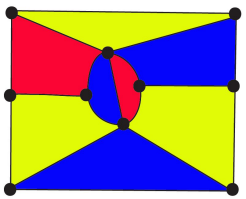
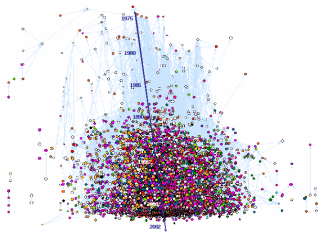
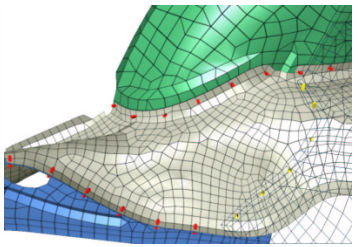
Computational Science *of* Computer Systems

Computational Science

Complex models to understand and predict the studied phenomenons

Large-Scale Distributed Systems

- Complex hierarchies: grid, cluster, multicore
- Massive parallelism: 10^6 cores (10^9 in 2020)
- Heterogeneity (GPU, SOC) ; Dynamicity

Scientific Challenge: Study system **correctness** and **performance**

- Need for adapted scientific instruments, combining approaches
- Modeling to understand system dynamics; Simulation to predict experiments

SimGrid

- Many competitors in each application domain (GridSim, PeerSim, BigSim. . .)

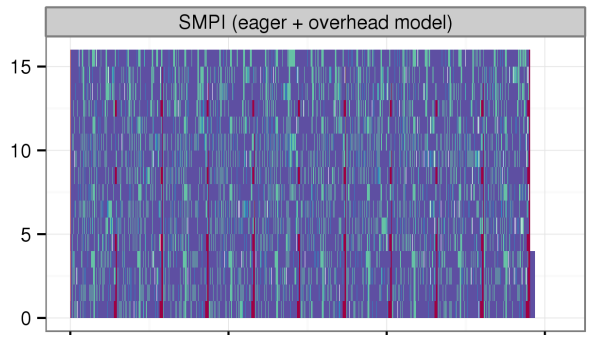
Approach and Originality in Modeling

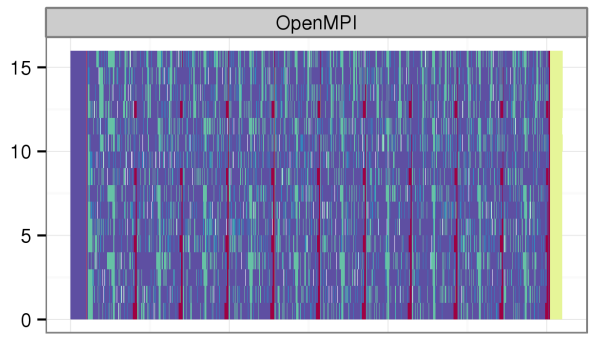
Epistemological stance

- Empirically consider large-scale computer systems as natural objects
- Complexity of these artificial artifacts reaches "natural" levels
- *Ich bin ein physicist*: Focus on modeling errors (more instructive)

Fine and accurate modeling of distributed applications (perfs & semantic)

Simulated

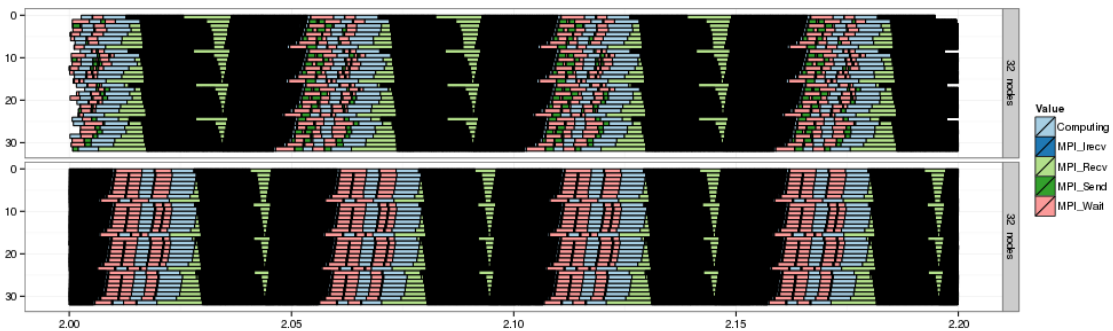


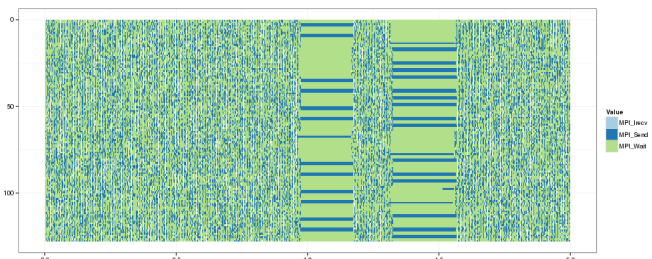


Real

Sweep3D

Reality sometimes sucks





Very long timeouts on pkt loss

Do we want to model it or fix it?

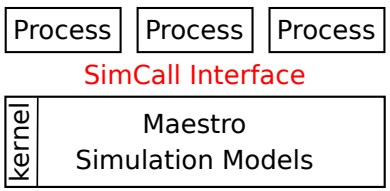
LU (left: 32p (up:real;down sim); right: 128)

Approach and Originality in Simulation

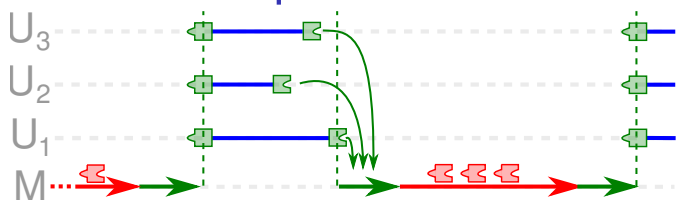
SimGrid is an Operating Simulator

- OS-like internal design, isolating user processes with simcalls

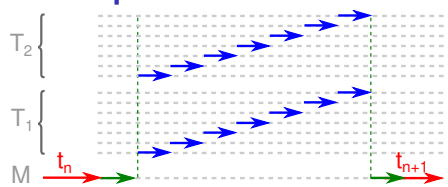
Functional View



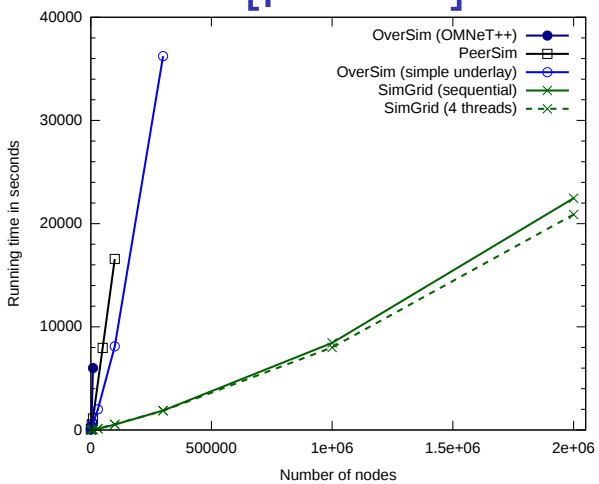
Temporal View



Implementation



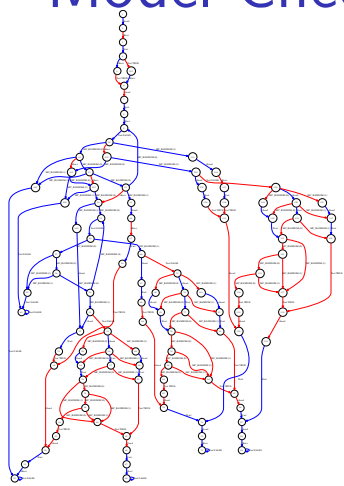
Efficient [parallel] simulation



dPeerSim: 2LP $\rightsquigarrow$ 4h / 16LP $\rightsquigarrow$ 1h

(but only 47s in sequential PeerSim, and 5s with SimGrid :)

Model-Checking (Safety & Liveness)



Exhaustive Chord

(2 processes)

- Aim at bug finding, not assessment
- System State Equality
- + DPOR Reduction
- Soon MPI-complient
- Soon more parallelism
- Soon statistical MC

My tools

Technical Side

- SimGrid has no dependency, works for Linux, Mac and Windows; C/Java
- Grid'5000 for reproducible experiments (both on modelisation and simulation)
- Pajé / Triva visualization tools, R statistical post-processing

Scientific Side

- ANR SONGS: Simulation Of Next Generation Systems
  8 WP: Grid/P2P/Clouds/HPC; Kernel/Models/Analyse/Open Science
  1.8M€ 2012-2016, Grenoble, Lyon, Bordeaux, Nantes, Strasbourg, Nice
  Action d'Envergure Nationale de l'ANR :)
- ANR USS SimGrid: Ultra Scalable Simulation (800k€ 2009-2011)
- Quelques petites choses à la région, en PHC avec la belgique, dans le CPER

Communication Side

- Conf.: IPDPS, HPDC, SC, CCGrid, EuroPar; CAV, Forte; EuroSys, Usenix
- Journals: TPDS, ParCo, JPDC, SPE
- SimGrid Users' Day, SimGrid Working Workshops, SimGrid Sprints

Take Away Messages

SimGrid will prove helpful to your research

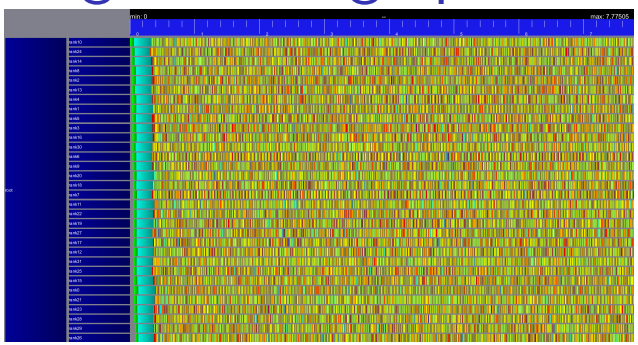
- Versatile: Used in several communities (scheduling, GridRPC, HPC, P2P, Clouds)
- Accurate: Model limits known thanks to validation studies
- Sound: Easy to use, extensible, fast to execute, scalable to death, well tested
- Open: User-community much larger than contributors group; LGPL
  120 publications (110 auteurs distincts, 5 continents), 4 PhD/ 25 commiters, 5 unaffiliated
- Around since over 10 years, and ready for at least 10 more years

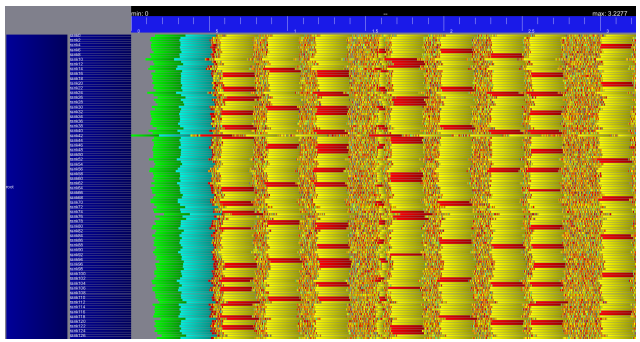
Welcome to the Age of (Sound) Computational Science

- Discover: `http://simgrid.gforge.inria.fr/`
- Learn: 101 tutorials, user manuals and examples
- Join: user mailing list, #simgrid on irc.debian.org
  We even have some open positions ;)

When simulation is better than reality

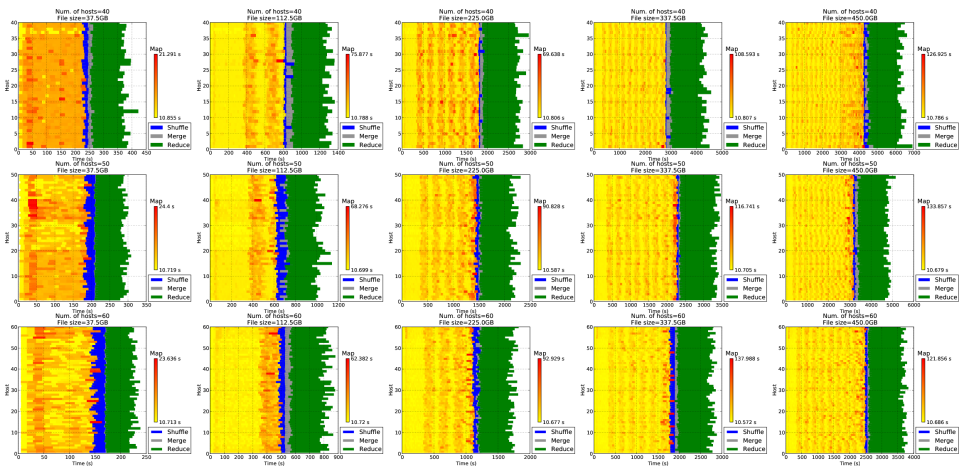
BigDFT on graphene

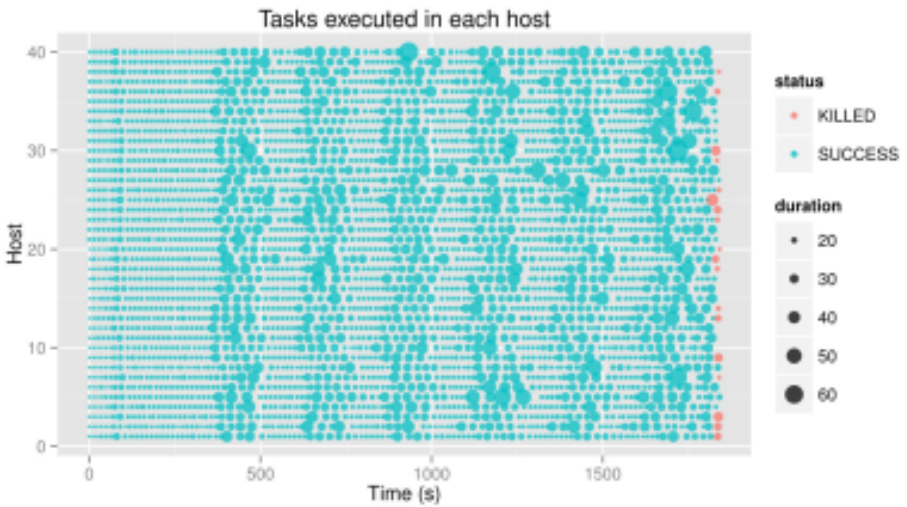




- Hardware bug
- Packet drops & delays
- Fix model or reality?

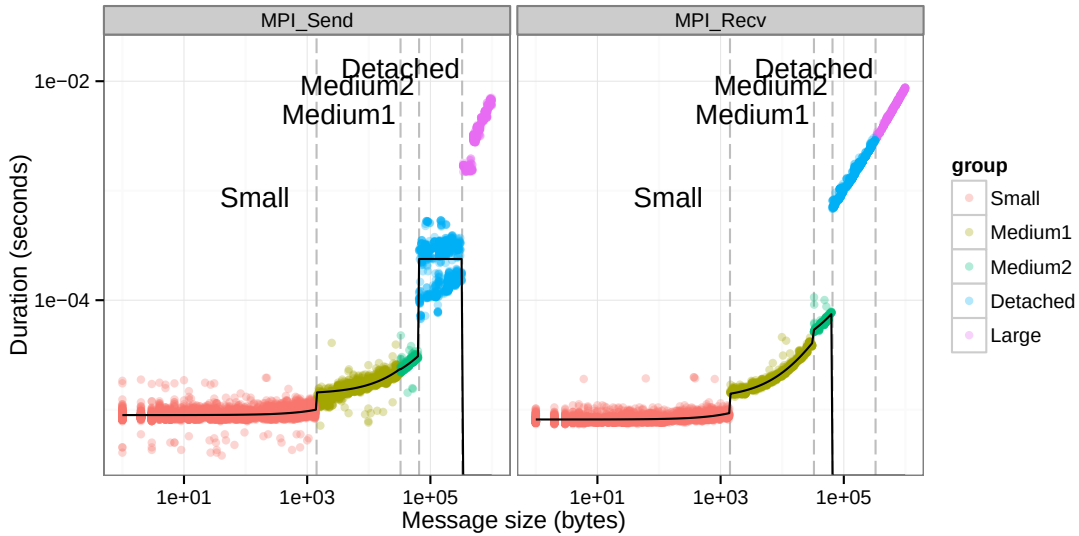
MapReduce on Grid'5000





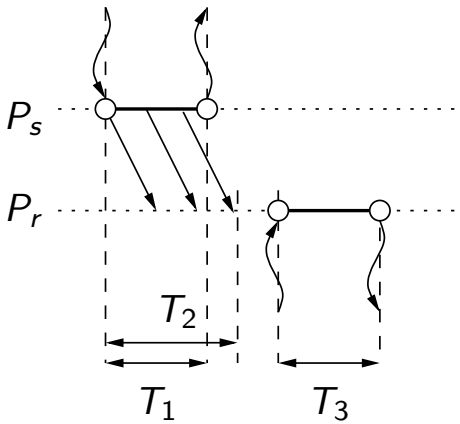
Modeling MPI point-to-point communication

Measurements

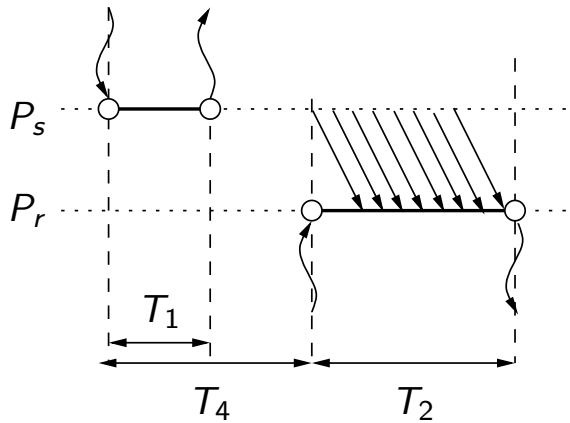


Models

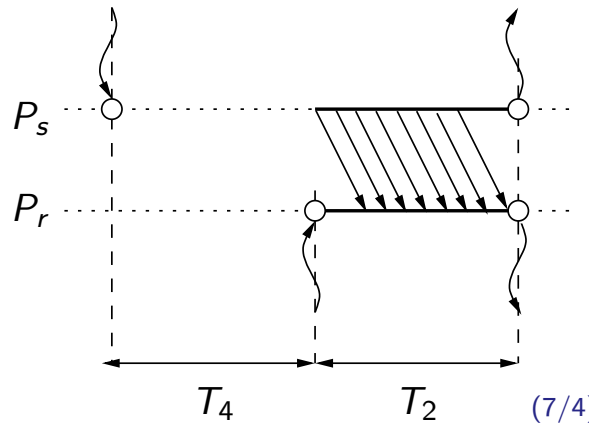
SMPI Asynchronous mode ($k \leq S_a$)



SMPI Detached mode ($S_a < k \leq S_d$)



SMPI Synchronous mode ($k > S_d$)



Open Science and CS^2

Les expériences sont améliorables

- Pas décrites avec assez de précision; le diable est dans les détails
- Problème de reproductibilité, un comble!

Les outils sont améliorables

- Beaucoup de micro-communautés: auteur seul utilisateur
- Peu d'outils passent l'épreuve du temps
- Certains outils sont buggués voire reconnus faux

Les méthodes complémentaires se marient mal

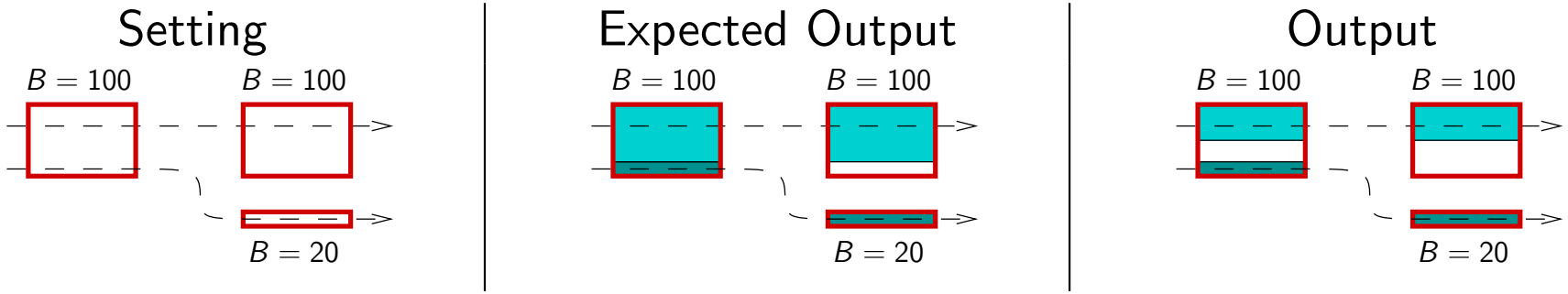
- Formalismes différents (programme $\neq$ prototype $\neq$ modèles formels)
- Outils différents, sans aucune interopérabilité
- Choix de l'outil dicté par les habitudes de l'expérimentateur

Vision: Open Science

- Des outils et méthodes standards pour regagner la reproductibilité
- Idée: SimGrid comme cheval de Troie des *best practices* expérimentales
- Collaboration avec Grid'5000, Distem et tous les autres

Invalidating Simulators from the Litterature

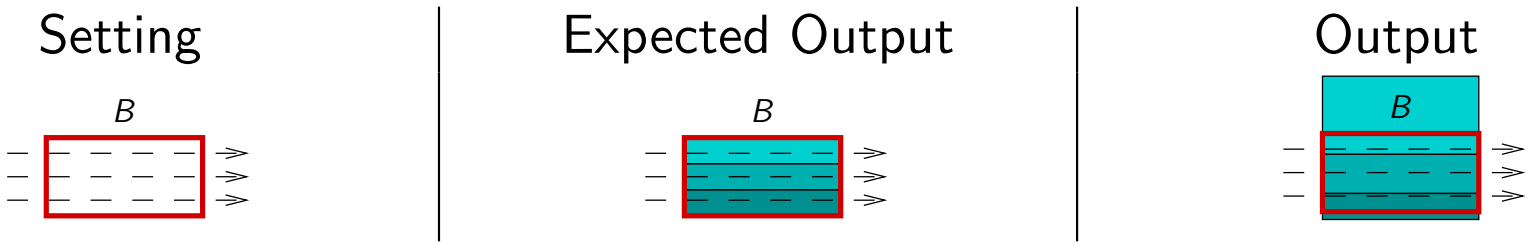
Naive flow models documented as wrong



Known issue in Narses (2002), OptorSim (2003), GroudSim (2011).

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 (GridSim 5.2, Nov. 25, 2010). Similar issues with naive packet-level models.

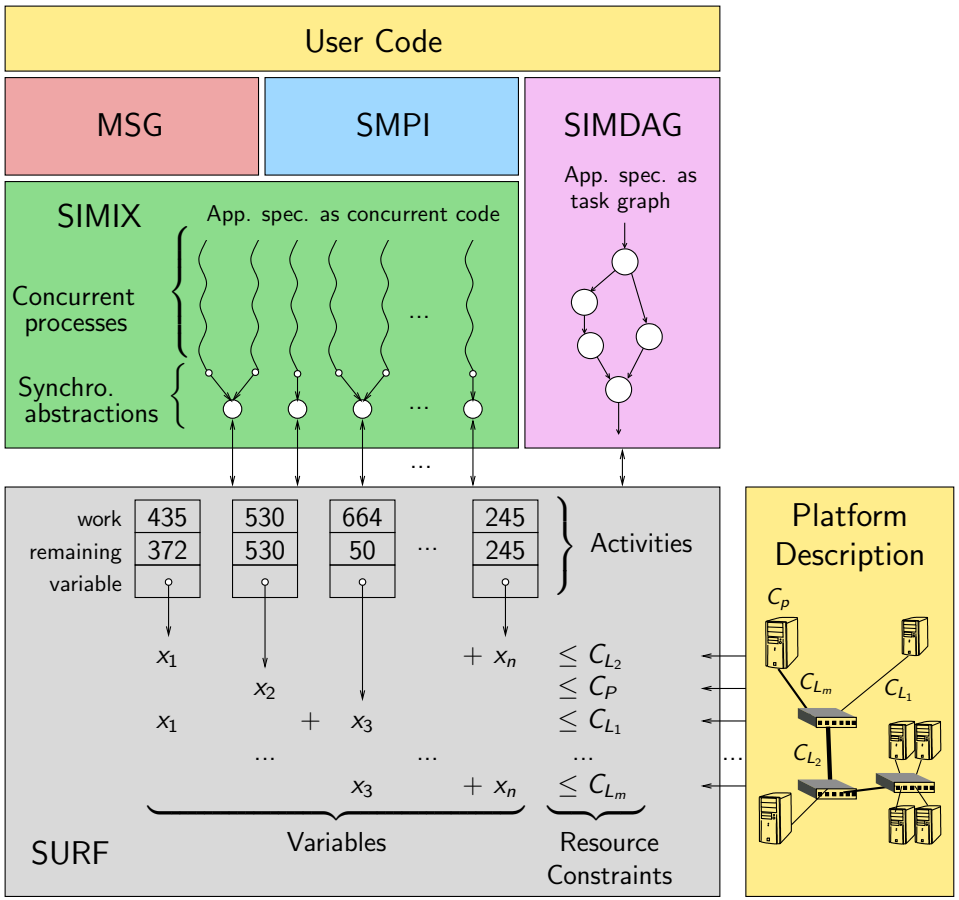
Quick Overview of Internals Organization

User-visible SimGrid Components

- 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 internal layers

- MSG: User-friendly syntactic sugar
- Simix: Processes, synchronizations
- SURF: Resources usage interface
- Models: Compute completion dates



SimGrid Usage example: Master/workers example

1. 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);
}
```

2. Describe your Experiment

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>
```

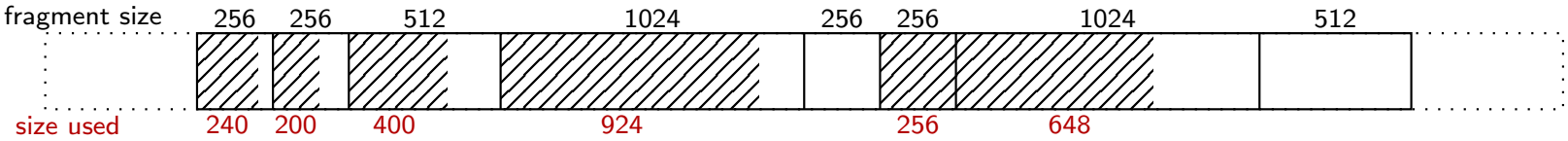
XML Deployment File

```
<?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>
```

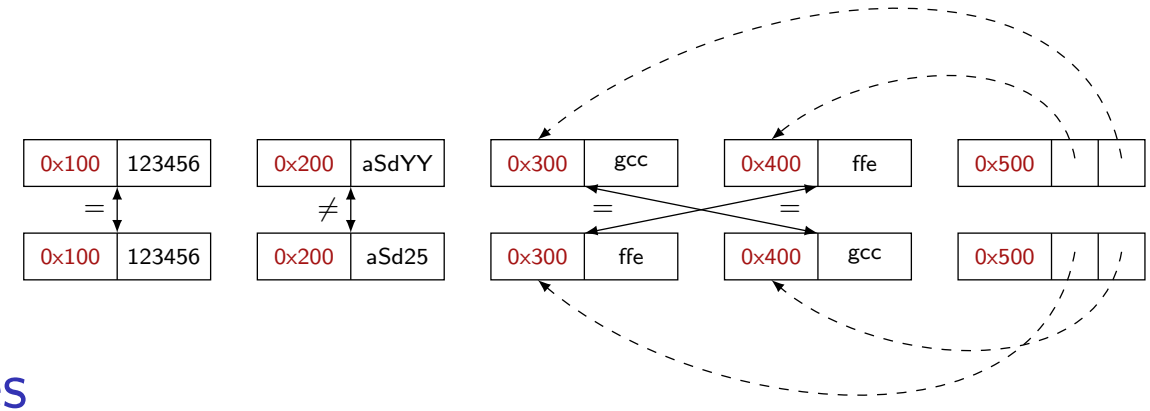
Challenges of System-level State Equality

Over provisioning



Syntactic differences

- In malloc, blocs order can vary without impacting applicative semantic



Padding Bytes

- Data is aligned in memory for efficiency, leaving holes

Irrelevant differences

- Host-related data (pid, files), simulation-related data (time)