

# Simulating Applications for Research in Simulation Applications for Research

Martin Quinson (LORIA–Nancy University, France)  
(thanks to Mario Lassnig *et Al.*)



Nancy-Université



# Simulating Applications for Research in Simulation Applications for Research

La simulation d'applications pour la recherche  
en applications de simulation pour la recherche

Martin Quinson (LORIA–Nancy University, France)  
(thanks to Mario Lassnig *et Al.*)

## Context

# Simulating Applications for Research in Simulation Applications for Research

## Context

# Simulating Applications for Research in Simulation Applications for Research

## Simulation Applications for Research

- ▶ Monte-Carlo Simulations last a large fraction of grids computation time
- ▶ (that's what I understand from the use of computers by scientists)

## Context

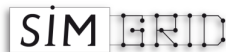
# Simulating Applications for Research in Simulation Applications for Research

## Simulation Applications for Research

- ▶ Monte-Carlo Simulations last a large fraction of grids computation time
- ▶ (that's what I understand from the use of computers by scientists)

## Simulating Application

- ▶ In CS, comparing heuristics on real platform challenging, labor-intensive
- ▶ Classical to simulate them on virtual systems
- ▶ Several tools exist for that, I work on one of them

The logo for SIM GRID. The word "SIM" is in a white box with a black border. The word "GRID" is in a black box with a white border. The letters are in a stylized, blocky font.

## Context

# Simulating Applications for Research in Simulation Applications for Research

## Simulation Applications for Research

- ▶ Monte-Carlo Simulations last a large fraction of grids computation time
- ▶ (that's what I understand from the use of computers by scientists)

## Simulating Application

- ▶ In CS, comparing heuristics on real platform challenging, labor-intensive
- ▶ Classical to simulate them on virtual systems
- ▶ Several tools exist for that, I work on one of them



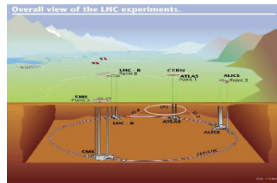
## Claim: Simulating Production Infrastructures may Help

- ▶ Experimentation on the infrastructure challenging (or simply impossible)
- ▶ Simulation controllable, easy to setup & fast
- ▶ SimGrid's realism & scalability good enough for your first try (at least)

# ATLAS: Example of such an infrastructure

## The Large Hardon Collider (LHC)

- ▶ European Scientific Instrument
- ▶ High energy physics particle accelerator
- ▶ Petabytes of data are captured during runs
- ▶ Data processed afterward  
(run Monte-Carlo simulations to assess models)



The ATLAS experiment needs an Infrastructure Handling the Data

# Agenda

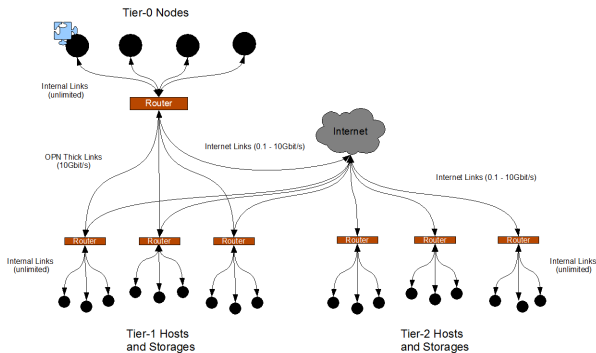
- The ATLAS Experiment
  - Overview
  - The DQ2 Infrastructure
  - DQ2 Use Cases
  - DQ2 Experimental Challenge
- Distributed Computing Experimentation in CS
  - Classical Methodologies
  - Simulation in Computer Science
- SimGRID
  - Overview
  - Simulation Models
  - Accuracy Assessment
  - Scalability Assessment
- SimGRID and Production Grids
  - Simulating Data-Intensive Applications
  - The Simterpose Project
- Conclusion



# DQ2 Use Cases

## Data Movements in DQ2

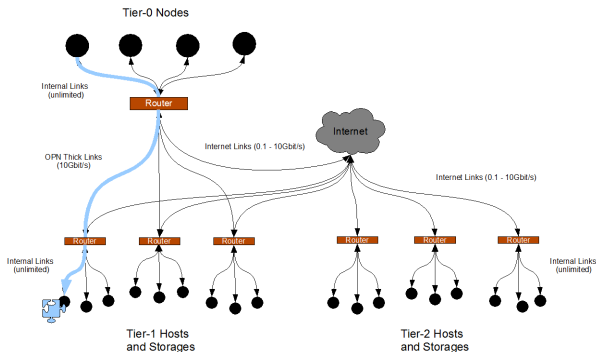
### 1. Data produced at CERN



# DQ2 Use Cases

## Data Movements in DQ2

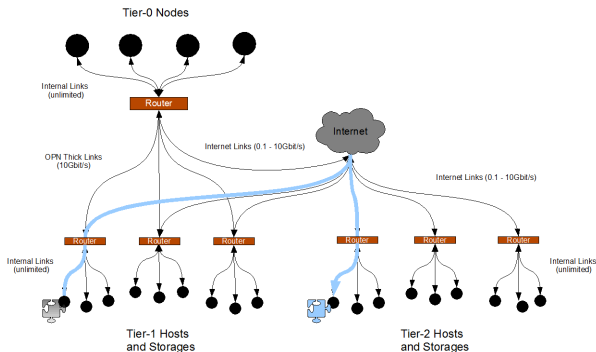
1. Data produced at CERN
2. Data stored on Tier-1



# DQ2 Use Cases

## Data Movements in DQ2

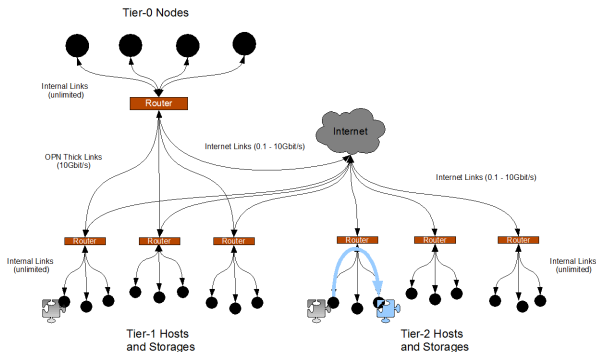
1. Data produced at CERN
2. Data stored on Tier-1
3. Duplicated on Tier-2 (on user's request)



# DQ2 Use Cases

## Data Movements in DQ2

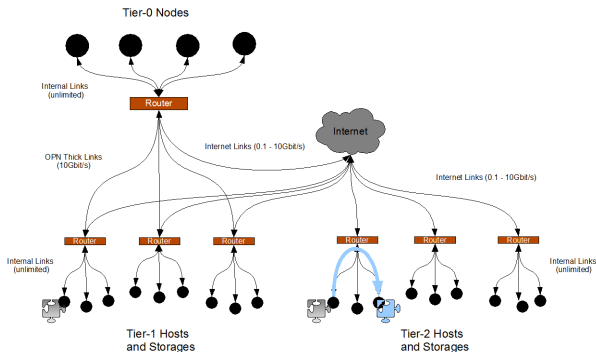
1. Data produced at CERN
2. Data stored on Tier-1
3. Duplicated on Tier-2 (on user's request)
4. Processed within Tier-2



# DQ2 Use Cases

## Data Movements in DQ2

1. Data produced at CERN
2. Data stored on Tier-1
3. Duplicated on Tier-2 (on user's request)
4. Processed within Tier-2



## Scalability Challenges

- ▶ Files are very large (over 100Mb; 2Gb or more are common)
- ▶ Basic processing induce 30M operations daily
- ▶ User requests induce  $\approx 2M$  operations daily
- ▶ Evaluations should consider one month of operation

# Assessing and Improving DQ2: Experimental Challenges

## No such infrastructure was ever built before

- ▶ As new as cloud infrastructure, but completely different
- ▶ No central control over the infrastructure
- ▶ Multi-national consortium; resources provided by different organizations
- ▶ This is **the** Data Grid.

## This is a production infrastructure

- ▶ Users simply won't tolerate you messing with the infrastructure
- ▶ Impossible (!) to shut down the service to experiment an idea
- ▶ Experiments and improvements need to be non-intrusive

# Assessing and Improving DQ2: Experimental Challenges

## No such infrastructure was ever built before

- ▶ As new as cloud infrastructure, but completely different
- ▶ No central control over the infrastructure
- ▶ Multi-national consortium; resources provided by different organizations
- ▶ This is **the** Data Grid.
- ▶ Hot Research Topic

## This is a production infrastructure

- ▶ Users simply won't tolerate you messing with the infrastructure
- ▶ Impossible (!) to shut down the service to experiment an idea
- ▶ Experiments and improvements need to be non-intrusive
- ▶ But how to test ideas and codes before deployment???

# Agenda

- The ATLAS Experiment
  - Overview
  - The DQ2 Infrastructure
  - DQ2 Use Cases
  - DQ2 Experimental Challenge
- Distributed Computing Experimentation in CS
  - Classical Methodologies
  - Simulation in Computer Science
- SimGRID
  - Overview
  - Simulation Models
  - Accuracy Assessment
  - Scalability Assessment
- SimGRID and Production Grids
  - Simulating Data-Intensive Applications
  - The Simterpose Project
- Conclusion

# Classical Experimental Methodologies

## Analytical works?

- ▶ Some purely mathematical models exist
- 😊 Allow better understanding of principles (impossibility theorems)
- 😞 Theoretical results are difficult to achieve (without unrealistic assumptions)
- ⇒ Most published research in the area is experimental

## Real-world experiments?

- 😊 Eminently *believable* to demonstrate the proposed approach applicability
- 😞 Very time and labor consuming; 😞 Reproducibility issues
- ⇒ Most published results rely on simulation or emulation

## Simulation and emulation?

- 😊 Solve most issues of real-world experiments (fast, easy, unlimited and repeatable)
- 😞 Validation issue (amongst others)
- ⇒ Tools validity must be carefully assessed

# Simulation in Computer Science

## Microprocessor Design

- ▶ A few standard “cycle-accurate” simulators are used extensively  
<http://www.cs.wisc.edu/~arch/www/tools.html>

⇒ Possible to reproduce simulation results

## Networking

- ▶ A few established “packet-level” simulators: NS-2, DaSSF, OMNeT++, GTNetS
- ▶ Well-known datasets for network topologies
- ▶ Well-known generators of synthetic topologies
- ▶ SSF standard: <http://www.ssfnet.org/>

⇒ Possible to reproduce simulation results

## Large-Scale Distributed Systems?

- ▶ No established simulator up until a few years ago
- ▶ Most people build their own “ad-hoc” solutions

Naicken, Stephen *et Al.*, *Towards Yet Another Peer-to-Peer Simulator*, HET-NETs'06.

From 141 P2P sim.papers, 30% use a custom tool, 50% don't report used tool

# Simulation in Parallel and Distributed Computing

- ▶ Used for decades, but under drastic assumptions in most cases

## Simplistic platform model

- ▶ Fixed computation and communication rates (Flops, Mb/s)
- ▶ Topology either fully connected or bus (no interference or simple ones)
- ▶ Communication and computation are perfectly overlappable

## Simplistic application model

- ▶ All computations are CPU intensive (no disk, no memory, no user)
- ▶ Clear-cut communication and computation phases
- ▶ Computation times even ignored in Distributed Computing community
- ▶ Communication times sometimes ignored in HPC community

## Straightforward simulation in most cases

- ▶ Fill in a Gantt chart or count messages with a computer rather than by hand
- ▶ No need for a “simulation standard”

# Large-Scale Distributed Systems Simulations?

## Simple models justifiable at small scale

- ▶ Cluster computing (matrix multiply application on switched dedicated cluster)
- ▶ Small scale distributed systems

## Hardly justifiable for Large-Scale Distributed Systems

- ▶ **Heterogeneity** of components (hosts, links)
  - ▶ **Quantitative:** CPU clock, link bandwidth and latency
  - ▶ **Qualitative:** ethernet vs myrinet vs quadrics; Pentium vs Cell vs GPU
- ▶ **Dynamicity**
  - ▶ **Quantitative:** resource sharing  $\rightsquigarrow$  availability variation
  - ▶ **Qualitative:** resource come and go (churn)
- ▶ **Complexity**
  - ▶ **Hierarchical systems:** grids of clusters of multi-processors being multi-cores
  - ▶ **Resource sharing:** network contention, QoS, batches
  - ▶ Multi-hop networks, non-negligible latencies
  - ▶ Middleware overhead (or optimizations)
  - ▶ Interference of computation and communication (and disk, memory, etc)

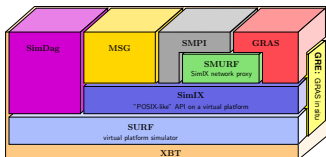
# The SimGrid Project (Hawai'i, Grenoble, Nancy)

## History

- ▶ Created originally for HPC, now used in Desktop Grid, P2P, etc.
- ▶ Original goal: scheduling research  $\leadsto$  need for speed (users do parameter sweep)
- ▶ HPC quality criteria: makespan  $\leadsto$  accuracy not negligible

## SimGRID in a Nutshell

- ▶ SimGRID is 10 years old: we explored several architectures, models, etc
- ▶ Many genericity hooks: modular, multi-API, multi-model  $\leadsto$  multi-community?



## Current Work

- ▶ Pushing scalability limits (targeting P2P)
- ▶ Validating the models

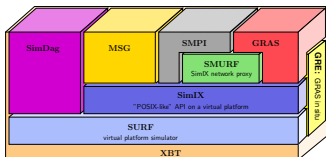
# The SimGrid Project (Hawai'i, Grenoble, Nancy)

## History

- ▶ Created originally for HPC, now used in Desktop Grid, P2P, etc.
- ▶ Original goal: scheduling research  $\leadsto$  need for **speed** (users do parameter sweep)
- ▶ HPC quality criteria: makespan  $\leadsto$  **accuracy** not negligible

## SimGRID in a Nutshell

- ▶ SimGRID is 10 years old: we explored several architectures, models, etc
- ▶ Many genericity hooks: modular, multi-API, multi-model  $\leadsto$  multi-community?



## Current Work

- ▶ Pushing scalability limits (targeting P2P)
- ▶ Validating the models

## Claim

- ▶ Could help assessing production infrastructures

Let's try to convince you of that

# Under the Hood: Simulation Models (1/2)

## Modeling CPU

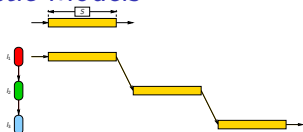
- ▶ Resource delivers  $pow$  flop / sec; task require  $size$  flop  $\Rightarrow$  lasts  $\frac{size}{pow}$  sec
- ▶ Simple (simplistic?) but more accurate become quickly intractable

## Modeling Single-Hop Networks

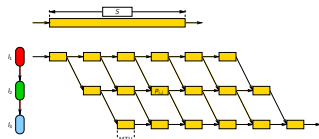
- ▶ Simplistic model:  $T = \lambda + \frac{size}{\beta}$ ;
- ▶ Better model: use  $\beta' = \min(\beta, \frac{W_{max}}{RTT})$   
(this accounts for TCP windowing algorithms)

# Modeling Multi-Hop Networks

## Simplistic Models



Store & Forward

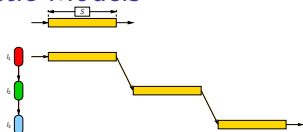


Wormhole

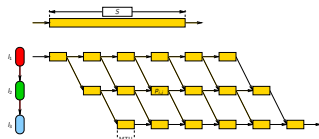
😊 Easy to implement; ☹ Not realistic (TCP Congestion omitted)

# Modeling Multi-Hop Networks

## Simplistic Models



Store & Forward



Wormhole

😊 Easy to implement; ☹ Not realistic (TCP Congestion omitted)

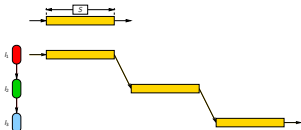
## NS2 and other packet-level

▶ Study the path of each and every network packet

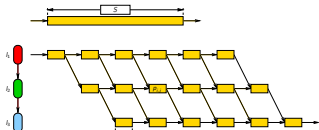
😊 Realism commonly accepted; ☹ Sloooooow

# Modeling Multi-Hop Networks

## Simplistic Models



Store & Forward



Wormhole

😊 Easy to implement; ☹️ Not realistic (TCP Congestion omitted)

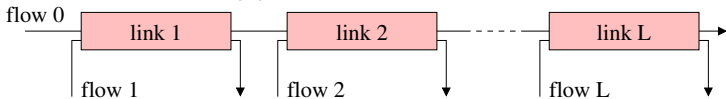
## NS2 and other packet-level

▶ Study the path of each and every network packet

😊 Realism commonly accepted; ☹️ Sloooooow

## Fluid Models:

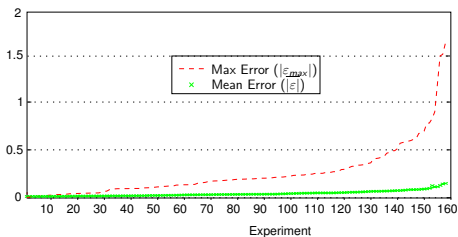
▶ Data streams  $\approx$  fluids in pipes



😊 Fast, Rather well studied; ☹️ Resource sharing; Realism not naturally clear

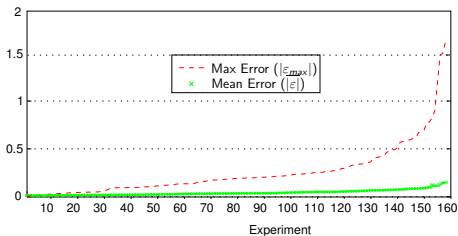
# Validation experiments on random platforms

- ▶ 160 Platforms (generator: BRITE)
- ▶  $\beta \in [10, 128]$  MB/s;  $\lambda \in [0; 5]$  ms
- ▶ Flow size:  $S=10$ MB
- ▶ #flows: 150; #nodes  $\in [50; 200]$
- ▶  $\overline{|\varepsilon|} < 0.2$  (i.e.,  $\approx 22\%$ );  
 $|\varepsilon_{max}|$  still challenging up to 461%



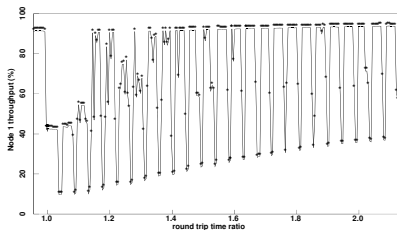
# Validation experiments on random platforms

- ▶ 160 Platforms (generator: BRITE)
- ▶  $\beta \in [10,128]$  MB/s;  $\lambda \in [0;5]$  ms
- ▶ Flow size:  $S=10$ MB
- ▶ #flows: 150; #nodes  $\in [50;200]$
- ▶  $|\overline{\varepsilon}| < 0.2$  (i.e.,  $\approx 22\%$ );  
 $|\varepsilon_{max}|$  still challenging up to 461%



## Maybe the error is not SimGrid's

- ▶ Big error because GTNetS multi-phased
- ▶ Seen the same in NS3, emulation, ...
- ▶ **Phase Effect:** Periodic and deterministic traffic may resonate [Floyd&Jacobson 91]
- ▶ Impossible in Internet (thx random noise)



~ We're adding random jitter to continue SimGRID validation

# Scalability Assessment

## Master/Workers on amd64 with 4Gb

| #tasks    | Context mechanism | #Workers |      |       |       |        |        |
|-----------|-------------------|----------|------|-------|-------|--------|--------|
|           |                   | 100      | 500  | 1,000 | 5,000 | 10,000 | 25,000 |
| 1,000     | ucontext          | 0.16     | 0.19 | 0.21  | 0.42  | 0.74   | 1.66   |
|           | pthread           | 0.15     | 0.18 | 0.19  | 0.35  | 0.55   | *      |
|           | java              | 0.41     | 0.59 | 0.94  | 7.6   | 27.    | *      |
| 10,000    | ucontext          | 0.48     | 0.52 | 0.54  | 0.83  | 1.1    | 1.97   |
|           | pthread           | 0.51     | 0.56 | 0.57  | 0.78  | 0.95   | *      |
|           | java              | 1.6      | 1.9  | 2.38  | 13.   | 40.    | *      |
| 100,000   | ucontext          | 3.7      | 3.8  | 4.0   | 4.4   | 4.5    | 5.5    |
|           | pthread           | 4.7      | 4.4  | 4.6   | 5.0   | 5.23   | *      |
|           | java              | 14.      | 13.  | 15.   | 29.   | 77.    | *      |
| 1,000,000 | ucontext          | 36.      | 37.  | 38.   | 41.   | 40.    | 41.    |
|           | pthread           | 42.      | 44.  | 46.   | 48.   | 47.    | *      |
|           | java              | 121.     | 130. | 134.  | 163.  | 200.   | *      |

\*: #semaphores reached system limit  
(2 semaphores per user process,  
System limit = 32k semaphores)

## Extensibility with UNIX contextes

| #tasks    | Stack size | #Workers |        |         |         |
|-----------|------------|----------|--------|---------|---------|
|           |            | 25,000   | 50,000 | 100,000 | 200,000 |
| 1,000     | 128Kb      | 1.6      | †      | †       | †       |
|           | 12Kb       | 0.5      | 0.9    | 1.7     | 3.2     |
| 10,000    | 128Kb      | 2        | †      | †       | †       |
|           | 12Kb       | 0.8      | 1.2    | 2       | 3.5     |
| 100,000   | 128Kb      | 5.5      | †      | †       | †       |
|           | 12Kb       | 3.7      | 4.1    | 4.8     | 6.7     |
| 1,000,000 | 128Kb      | 41       | †      | †       | †       |
|           | 12Kb       | 33       | 33.6   | 33.7    | 35.5    |
| 5,000,000 | 128Kb      | 206      | †      | †       | †       |
|           | 12Kb       | 161      | 167    | 161     | 165     |

## Scalability limit of GridSim

- ▶ 1 user process = 3 java threads (code, input, output)
  - ▶ System limit = 32k threads
- ⇒ at most 10,922 user processes

†: out of memory

# Agenda

- The ATLAS Experiment
  - Overview
  - The DQ2 Infrastructure
  - DQ2 Use Cases
  - DQ2 Experimental Challenge
- Distributed Computing Experimentation in CS
  - Classical Methodologies
  - Simulation in Computer Science
- SimGRID
  - Overview
  - Simulation Models
  - Accuracy Assessment
  - Scalability Assessment
- SimGRID and Production Grids
  - Simulating Data-Intensive Applications
  - The Simterpose Project
- Conclusion

# Package Evaluation

During Summer 2009, 2 interns @CERN tried to simulate DQ2

- ▶ They evaluated GridSim and SimGrid for that
- ▶ I contacted them (*afterward* ;) and report verbatim their findings here

## SimGrid

- ▶ Based on pure C
- 😊 Fast execution time; low memory consumption; scalable
- ☹ Lacking in some functionality; High level of abstraction

## GridSim (Buyya *et Al.*)

- ▶ Java-based
- 😊 Highly developed; internal logging of network traffic; easier to use; Packet-based
- ☹ Slow execution time; bad memory consumption; not scalable

# Evaluation of Grid computing simulation packages

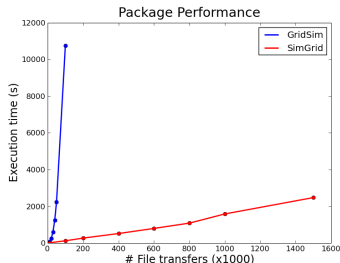
## Experiment description

- ▶ Attempted to simulate one day on grid (1.5 million file transfers)
- ▶ Recaping final requirements:
  - ▶ Basic processing induce 30M operations daily
  - ▶ User requests induce  $\approx 2$ M operations daily
  - ▶ Evaluations should consider one month of operation

## Findings

**GridSim** exponential in CPU time with increasing transfers

**SimGrid** linear in CPU Time with increasing transfers



# Adding Missing Models to SimGrid

## Facts

- ▶ SimGRID is the only grid simulation package able to deal with DQ2's scale
- ▶ Data storage elements are not handled in SimGRID

# Adding Missing Models to SimGrid

## Facts

- ▶ SimGRID is the only grid simulation package able to deal with DQ2's scale
- ▶ Data storage elements are not handled in SimGRID

## Simulating Data-Intensive Application

- ▶ We submitted a project to the IDG/G5K call to bootstrap a collaboration
- ▶ Consortium: DQ2@CERN, AIGorille@INRIA Nancy, ProGGres@CC.IN2P3
- ▶ Goals:
  - ▶ CERN help SimGrid providing a data storage resource model
  - ▶ SimGrid help modeling such a large infrastructure
- ▶ Benefits:
  - ▶ SimGrid gains a new resource type and becomes usable for data grid too
  - ▶ CERN gets the tool fitted to its needs

# Adding Missing Models to SimGrid

## Facts

- ▶ SimGRID is the only grid simulation package able to deal with DQ2's scale
- ▶ Data storage elements are not handled in SimGRID

## Simulating Data-Intensive Application

- ▶ We submitted a project to the IDG/G5K call to bootstrap a collaboration
- ▶ **Consortium:** DQ2@CERN, AIGorille@INRIA Nancy, ProGGres@CC.IN2P3
- ▶ **Goals:**
  - ▶ CERN help SimGrid providing a data storage resource model
  - ▶ SimGrid help modeling such a large infrastructure
- ▶ **Benefits:**
  - ▶ SimGrid gains a new resource type and becomes usable for data grid too
  - ▶ CERN gets the tool fitted to its needs
- ▶ **Requested Funding:** 2-3 meetings, 4k €
- ▶ **Envisioned Projects:**
  - ▶ (not ANR since CERN is not french)
  - ▶ An INRIA's ARC (a post-doc at least)
  - ▶ A European project, or become part of a bigger European project

# Agenda

- The ATLAS Experiment
  - Overview
  - The DQ2 Infrastructure
  - DQ2 Use Cases
  - DQ2 Experimental Challenge
- Distributed Computing Experimentation in CS
  - Classical Methodologies
  - Simulation in Computer Science
- SimGRID
  - Overview
  - Simulation Models
  - Accuracy Assessment
  - Scalability Assessment
- SimGRID and Production Grids
  - Simulating Data-Intensive Applications
  - The Simterpose Project
- Conclusion

# SimGrid use limitation

## Main limitation of SimGrid today

- ▶ You have to write your application using its interfaces
- ▶ Impossible to reuse it on real life

## Some partial solution exist

- ▶ GRAS allows you to reuse the code written in SG on real life
  - ☹ you still have to learn a new API
- ▶ SMPI allows you to run MPI code in SG
  - ☹ not anyone use MPI

## It ought to be a better solution

- ▶ How could I just launch my application on “virtual platform”?
- ▶ This project is dubbed **simterpose** (interposing a simulator)

# Simterpose: presentation

Goal: Allowing to use SimGrid on unmodified distributed applications

## Why? Motivations

- ▶ Test your code in reproducible way
- ▶ Dimension your hardware to fit your application
- ▶ Process folding (debug in the train – w/o GSM)

## How? what's needed?

- ▶ Intercept any interaction with the system (send, receive, gettimeofday)
- ▶ Report them into the simulator
- ▶ Reflect simulated reality into real one  
slow down the process by the given amount of time, return simulated clock value

# Simterpose: presentation

Goal: Allowing to use SimGrid on unmodified distributed applications

## Why? Motivations

- ▶ Test your code in reproducible way
- ▶ Dimension your hardware to fit your application
- ▶ Process folding (debug in the train – w/o GSM)

## How? what's needed?

- ▶ Intercept any interaction with the system (send, receive, gettimeofday)
- ▶ Report them into the simulator
- ▶ Reflect simulated reality into real one  
slow down the process by the given amount of time, return simulated clock value

## But how technically??

- ▶ `#define send(a,b,c) sg_send(a,b,c)`
- ▶ PTRACE (trace processes as gdb does) ▶ Library injection (LD\_PRELOAD)
- ▶ Valgrind (Code injection in binary before running it)
- ▶ Real virtual machine (qemu, xen) ▶ Java Agents (way to profile Java programs)

# Simterpose: presentation

**Goal:** Allowing to use SimGrid on unmodified distributed applications

## Why? Motivations

- ▶ Test your code in reproducible way
- ▶ Dimension your hardware to fit your application
- ▶ Process folding (debug in the train – w/o GSM)

## How? what's needed?

- ▶ Intercept any interaction with the system (send, receive, gettimeofday)
- ▶ Report them into the simulator
- ▶ Reflect simulated reality into real one  
slow down the process by the given amount of time, return simulated clock value

## But how technically??

- ▶ `#define send(a,b,c) sg_send(a,b,c)`
- ▶ PTRACE (trace processes as gdb does) ▶ Library injection (LD\_PRELOAD)
- ▶ Valgrind (Code injection in binary before running it)
- ▶ Real virtual machine (qemu, xen) ▶ Java Agents (way to profile Java programs)

## Connexion to Production Grids

- ▶ SimGlite project submitted: Assess gLite stack partially using this approach

# Agenda

- The ATLAS Experiment
  - Overview
  - The DQ2 Infrastructure
  - DQ2 Use Cases
  - DQ2 Experimental Challenge
- Distributed Computing Experimentation in CS
  - Classical Methodologies
  - Simulation in Computer Science
- SimGRID
  - Overview
  - Simulation Models
  - Accuracy Assessment
  - Scalability Assessment
- SimGRID and Production Grids
  - Simulating Data-Intensive Applications
  - The Simterpose Project
- Conclusion

# Conclusion

## Production Grids are challenging to assess

- ▶ Very hard (impossible?) to experiment on production platforms
- ▶ Using evaluation platforms may help, but Simulating too (Cost/Trust tradeoff)

# Conclusion

## Production Grids are challenging to assess

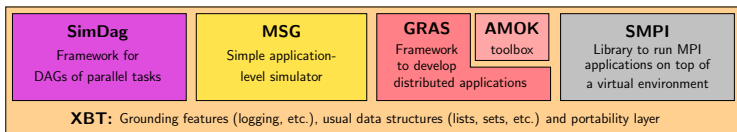
- ▶ Very hard (impossible?) to experiment on production platforms
- ▶ Using evaluation platforms may help, but Simulating too (Cost/Trust tradeoff)

## SimGRID could reveal helpful here (too)

- ▶ **Multi-Community:** Initially HPC; already used in Desktop Grids; P2P soon
- ▶ **Provides Interesting Models:** fast and shown accurate
  - ▶ When chasing SimGRID accuracy limits, we found packet-level ones
  - ▶ 30,000 requests/sec (and counting) in Master/Workers classical example
- ▶ **Is Configurable:** Platform, Deployment, Workload and Code not intermixed
- ▶ **Enjoys a solid user community:** 130 members on -user; grounded >40 papers

Any questions?

# User-visible SimGrid Components



## SimGrid user APIs

- ▶ **Specialized APIs:** Designed for a specific community, genericity not a goal
  - ▶ **SimDag:** model applications as DAG of (parallel) tasks
  - ▶ **SMPI:** simulate MPI codes
- ▶ **Generic APIs:** allow to express Concurrent Sequential Processes (CSP)
  - ▶ **MSG:** study heuristics, get quickly some performance evaluation charts
  - ▶ **GRAS:** develop real applications, studied and debugged in simulator
- ▶ (+XBT: grounding toolbox easing C coding)

## Argh, you really expect me to code in C?!

- ▶ Java bindings to MSG exist, other are planned (Python, C++, SimDag)
- ▶ Some bad sides of C avoided: feature-rich toolbox w/o dependency, portable

# SimGrid Usage Workflow: the MSG example (1/2)

## 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>
```

# SimGrid Usage Workflow: the MSG example (2/2)

## 3. 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());  
}
```

## 4. Compile your code (linked against -lsimgrid), run it and enjoy

### Executive summary, but representative

- ▶ Similar in others interfaces, but:
  - ▶ glue is generated by a script in GRAS and automatic in Java with introspection
  - ▶ in SimDag, no deployment file since no CSP
- ▶ Platform can contain trace informations, Higher level tags and Arbitrary data
- ▶ In MSG, applicative workload can also be externalized to a trace file

# So, what is the model used in SimGrid?

“`--cfg=network_model`” command line argument

- ▶ `CM02`, `LV08`  $\rightsquigarrow$  MaxMin fairness (give a fair share to everyone)
- ▶ `Vegas`  $\rightsquigarrow$  Vegas TCP fairness (Lagrangian approach)
- ▶ `Reno`  $\rightsquigarrow$  Reno TCP fairness (Lagrangian approach)
- ▶ By default: `LV08`
- ▶ Example: `./my_simulator --cfg=network_model:Vegas`

## CPU sharing policy

- ▶ Default MaxMin is sufficient for most cases
- ▶ `cpu_model:ptask_L07`  $\rightsquigarrow$  model specific to parallel tasks

## Want more?

- ▶ `network_model:gtnets`  $\rightsquigarrow$  use Georgia Tech Network Simulator for network Accuracy of a packet-level network simulator without changing your code (!)
- ▶ Plug your own model in SimGrid!!
- ▶ Other models are currently cooking (constant time, last-mile, etc.)

# Running real-world experiments

- 😊 Eminently *believable* to demonstrate the proposed approach applicability
- 😞 Very time and labor consuming
  - ▶ Entire application must be functional
  - ▶ Parameter-sweep; Design alternatives
- 😞 Choosing the right testbed is difficult
  - ▶ My own little testbed?
    - 😊 Well-behaved, controlled, stable
    - 😞 Rarely representative of production platforms
  - ▶ Real production platforms?
    - ▶ Not everyone has access to them; CS experiments are disruptive for users
    - ▶ Experimental settings may change drastically during experiment (components fail; other users load resources; administrators change config.)
- 😞 Results remain limited to the testbed
  - ▶ Impact of testbed specificities hard to quantify  $\Rightarrow$  collection of testbeds...
  - ▶ Extrapolations and explorations of “what if” scenarios difficult (what if the network were different? what if we had a different workload?)
- 😞 Experiments are uncontrolled and unrepeatable
  - ▶ No way to compare fairly alternatives back-to-back (even if disruption is part of the experiment)
  - ▶ Difficult for others to reproduce results even if this is the basis for scientific advances!

# Running real-world experiments

- 😊 Eminently *believable* to demonstrate the proposed approach applicability
- 😞 Very time and labor consuming
  - ▶ Entire application must be functional
  - ▶ Parameter-sweep; Design alternatives
- 😞 Choosing the right testbed is difficult
  - ▶ My own little testbed?
    - 😊 Well-behaved, controlled, stable
    - 😞 Rarely representative of production platforms
  - ▶ Real production platforms?
    - ▶ Not everyone has access to them; **CS experiments are disruptive for users**
    - ▶ Experimental settings may change drastically during experiment (components fail; other users load resources; administrators change config.)
- 😞 Results remain limited to the testbed
  - ▶ Impact of testbed specificities hard to quantify  $\Rightarrow$  collection of testbeds...
  - ▶ Extrapolations and explorations of “what if” scenarios difficult (what if the network were different? what if we had a different workload?)
- 😞 Experiments are uncontrolled and unrepeatable
  - ▶ No way to compare fairly alternatives back-to-back (even if disruption is part of the experiment)
  - ▶ **Difficult for others to reproduce results** even if this is the basis for scientific advances!

# Simulation

## 😊 Simulation solves these difficulties

- ▶ No need to build a real system, nor the full-fledged application
- ▶ Ability to conduct controlled and repeatable experiments
- ▶ (Almost) no limits to experimental scenarios
- ▶ Possible for anybody to reproduce results

# Simulation

## 😊 Simulation solves these difficulties

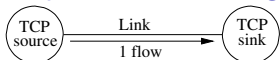
- ▶ No need to build a real system, nor the full-fledged application
- ▶ Ability to conduct controlled and repeatable experiments
- ▶ (Almost) no limits to experimental scenarios
- ▶ Possible for anybody to reproduce results

## Simulation in a nutshell

- ▶ *Predict aspects of the behavior of a system using approximate model of it*
- ▶ **Model:** Set of objects defined by a state  $\oplus$  Rules governing the state evolution
- ▶ **Simulator:** Program computing the evolution according to the rules
- ▶ **Wanted features:**
  - ▶ **Accuracy:** Correspondence between simulation and real-world
  - ▶ **Scalability:** Actually usable by computers (fast enough)
  - ▶ **Tractability:** Actually usable by human beings (simple enough to understand)
  - ▶ **Instanciability:** Can actually describe real settings (no magical parameter)
  - ▶ **Relevance:** Captures object of interest

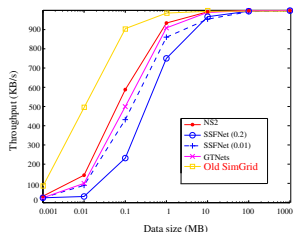
# Validation experiments on a single link

## Experimental settings

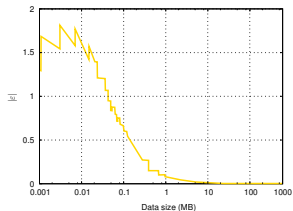


- ▶ Compute achieved **bandwidth as function of S**
- ▶ Fixed  $L=10\text{ms}$  and  $B=100\text{MB/s}$

## Evaluation Results

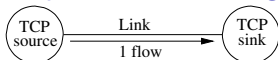


- ▶ Packet-level tools don't completely agree
- ▶ SSFNet TCP\_FAST\_INTERVAL bad default
- ▶ GTNetS is equally distant from others
- ▶ Old SimGrid model omitted slow start effects



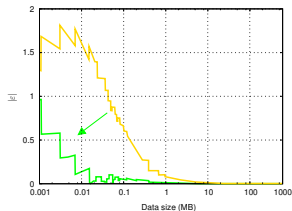
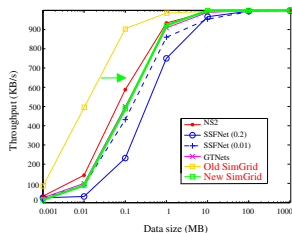
# Validation experiments on a single link

## Experimental settings



- ▶ Compute achieved **bandwidth as function of S**
- ▶ Fixed  $L=10\text{ms}$  and  $B=100\text{MB/s}$

## Evaluation Results



- ▶ Packet-level tools don't completely agree
- ▶ SSFNet TCP\_FAST\_INTERVAL bad default
- ▶ GTNetS is equally distant from others
- ▶ Old SimGrid model omitted slow start effects
- ⇒ Statistical analysis of GTNetS slow-start
- ~ Better instantiation of MaxMin model  
 $\beta'' \sim .92 \times \beta'$ ;  $\lambda \sim 10.4 \times \lambda$
- ▶ Resulting validity range quite acceptable

| S                  | $ \overline{\epsilon} $ | $ \epsilon_{max} $ |
|--------------------|-------------------------|--------------------|
| $S < 100\text{KB}$ | $\approx 12\%$          | $\approx 162\%$    |
| $S > 100\text{KB}$ | $\approx 1\%$           | $\approx 6\%$      |

# Simulation speed: **SimGrid's faster than packet-level sims.**

200-nodes/200-flows network sending **1MB** each

| # of flows | GTNetS          |                             | SimGrid         |                             |
|------------|-----------------|-----------------------------|-----------------|-----------------------------|
|            | Simulation time | <i>simulation simulated</i> | Simulation time | <i>simulation simulated</i> |
| 10         | 0.661s          | 0.856                       | 0.002s          | 0.002                       |
| 100        | 7.649s          | 7.468                       | 0.137s          | 0.140                       |
| 200        | 15.705s         | 11.515                      | 0.536s          | 0.396                       |

200-nodes/200-flows network sending **100MB** each

| # of flows | GTNetS          |                             | SimGrid         |                             |
|------------|-----------------|-----------------------------|-----------------|-----------------------------|
|            | Simulation time | <i>simulation simulated</i> | Simulation time | <i>simulation simulated</i> |
| 10         | 65s             | 0.92                        | 0.001s          | 0.00002                     |
| 100        | 753s            | 8.08                        | 0.138s          | 0.00142                     |
| 200        | 1562s           | 12.59                       | 0.538s          | 0.00402                     |

## Conclusion

- ▶ GTNetS execution time linear in both data size and #flows
- ▶ SimGrid only depends on #flows
- ▶ (plus, GTNetS clearly outperforms NS2)

# Simterpose: approaches

Hard part: interception. How to intercept calls to system and libraries?

- ▶ `#define send(a,b,c) sg_send(a,b,c)`
  - 😊 quite easy to do (SMPI does so)
  - ☹ recompilation is mandatory (thus, need source code); ☹ C only
- ▶ PTRACE (trace processes as gdb does)
  - 😊 Seamless
  - ☹ syscalls only (one may want to follow pthread calls); ☹ reputed slow
- ▶ Library injection (LD\_PRELOAD under linux)
  - ▶ The system linker use your symbols in preference to classical ones
  - ☹ Only library calls, not syscalls (but anyone uses libcs' wrappers)
  - 😊 Seamless, it could even trick the JVM?
- ▶ Valgrind (Code injection in binary before running it)
  - 😊 Seamless, would trick the JVM; ☹ Slow (x40 for empty valgrind tools)
- ▶ Real virtual machine (qemu, xen, etc)
  - 😊 Seamless, would trick the JVM; ☹ Slow, huge memory requirements
- ▶ Java Agents (the classical way of spying Java programs for profiling)
  - 😊 Seamless, path quite well paved; ☹ Java only