Challenges and solutions in simulating clouds

Marc FRINCU
Associate Professor, PhD
West University of Timisoara
Faculty of Mathematics and Computer Science
Department of Computer Science

cHiPSet Training School
Bucharest, Romania, 21-23 September 2016

Distributed systems

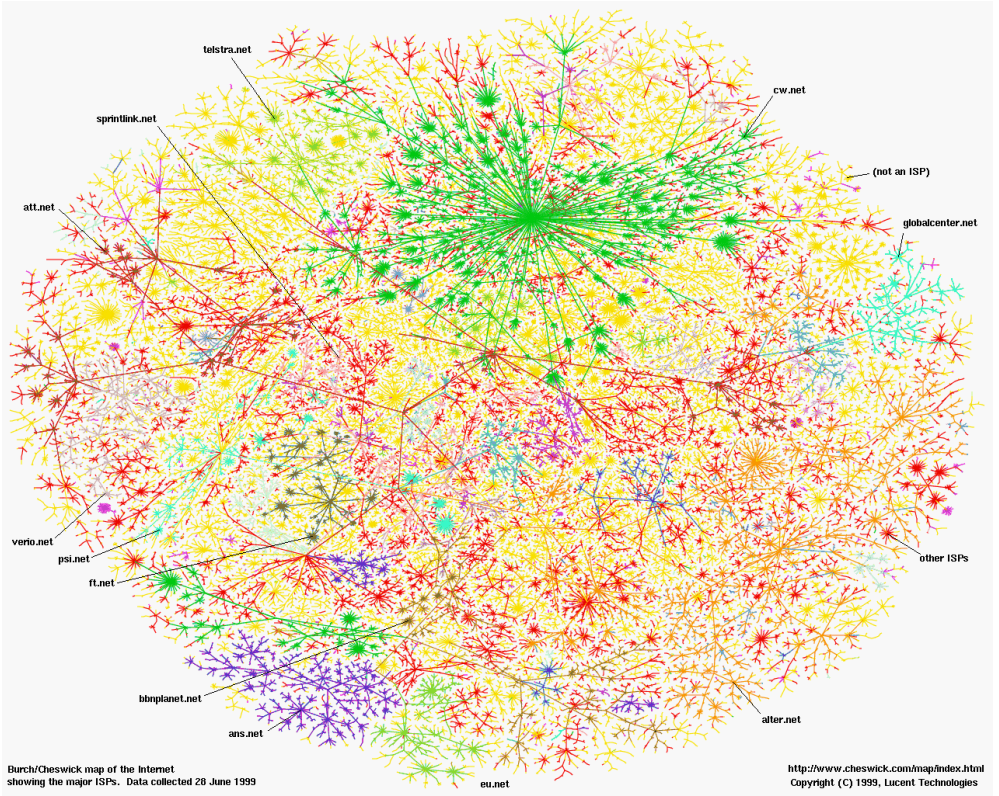
- *"A collection of (probably heterogeneous) automata whose distribution is transparent to the user so that the system appears as one local machine. This is in contrast to a network, where the user is aware that there are several machines, and their location, storage replication, load balancing and functionality is not transparent. Distributed systems usually use some kind of client-server organization."* – FOLDOC
- *"*
- *A Distributed System comprises several single components on different computers, which normally do not operate using shared memory and as a consequence communicate via the exchange of messages. The various components involved cooperate to achieve a common objective such as the performing of a business process."* – Schill & Springer

- Main characteristics
  - Components:
    - Multiple spatially separated individual components
    - Components posses own memory
    - Cooperation towards a common objective
  - Resources:
    - Access to common resources (e.g., databases, file systems)
  - Communication:
    - Communication via messages
  - Infrastructure:
    - Heterogeneous hardware infrastructure & software middleware

Distributed systems

- These definitions do not define the insides of a distributed system
  - Design and implementation
  - Maintenance
  - Algorithmics (i.e., protocols)

Facebook social network graph.



The Internet color coded by ISPs.

A working definition

- *"A distributed system (DS) is a **collection of entities**, each of which is **autonomous**, **programmable**, **asynchronous** and **failure-prone**, and which **communicate** through an **unreliable** communication medium, appearing to the end user as a **single** coherent system."*

- Key terms
  - Entity = process on a device (PC, server, tablet, smartphone)
  - Communication medium = wired or wireless network

Parallel, concurrent, and distributed computing

- **Parallelism**
  - Perform multiple tasks **at the same time**
  - True parallelism requires distribution on multiple processors/cores/machines
  - Can range from many core to multi processor to many computer on shared or distributed memory
- **Concurrency**
  - **Computations** with **multiple threads**
  - Can exploit hardware parallelism but it is inherently related to the software need (i.e., react to different asynchronous events)
  - Concurrency becomes parallelism if parallelism is real (one thread per processor/core/machine) not virtual
- **Distributed computing**
  - Related to **where the computation physically resides**
    - Distributed algorithm is executed on multiple CPUs, connected by networks, buses or any other data communication channel
  - Computers are connected by communication links on distributed memories
    - Rely fundamentally on message passing
  - Usually part of the goal
    - If resources are geographically spread than the system is inherently distributed

Parallel vs. distributed computing

- Is *distributed* computing a subset of *parallel* computing?
- Not an easy answer
- In favor
  - Distributed computing is parallel computing on geographically spread machines
    - distributed → parallel → concurrent computing
- Against
  - They address different issues
    - Distributed computing is focused on issues related to computation and data distribution
    - Parallel computing **does not** address problems such as partial failures
    - Parallel computing **focuses** on tightly coupled applications

Parallel vs. distributed systems

	Parallel Systems	**Distributed Systems**
Memory	**Tightly coupled shared** memory UMA, NUMA	**Distributed** memory Message passing, RPC, and/or used of distributed shared memory
Control	**Global clock** control SIMD, MIMD	**No global clock** control Synchronization algorithms needed
Processor interconnection	Order of **Tbps** Bus, mesh, tree, mesh of tree, and hypercube (-related) network	Order of **Gbps** Ethernet(bus), token ring and SCI (ring), myrinet(switching network)
Main focus	**Performance** Scientific computing	Performance(**cost and scalability**) **Reliability/availability** **Information/resource sharing**

Source: http://courses.washington.edu/css434/slides/w03w04/Fundamentals.ppt

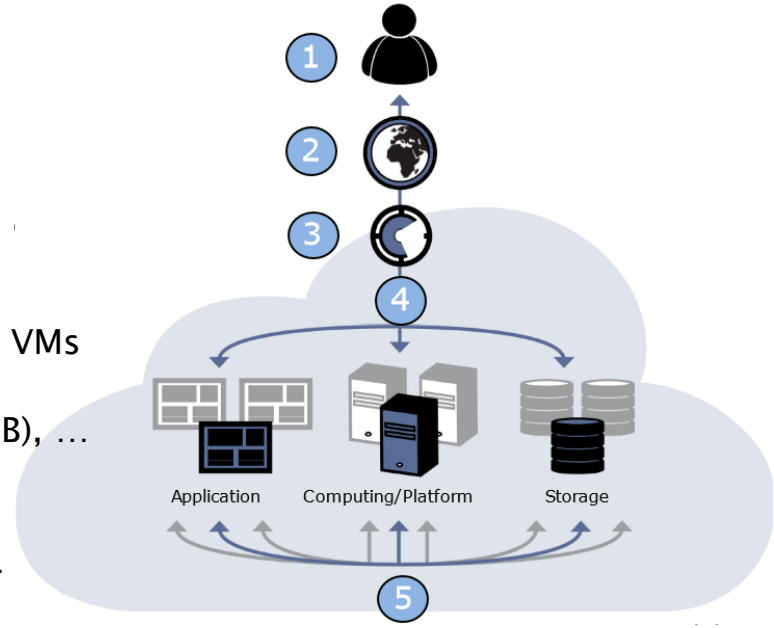
Types of distributed systems

- **Cluster computing**
  - Centralized management of resources which are available as orchestrated shared services
- **Grid computing**
  - Collection of resources from different places working together to reach a common goal
  - Loosely coupled
  - Heterogeneous
  - Geographically dispersed
- **Cloud computing**
  - On demand virtualized access to geographically distributed resources
    - Elastic
    - Pay-per-use (economic model)
    - Multiple layers of abstraction (IaaS, PaaS, SaaS, DaaS)

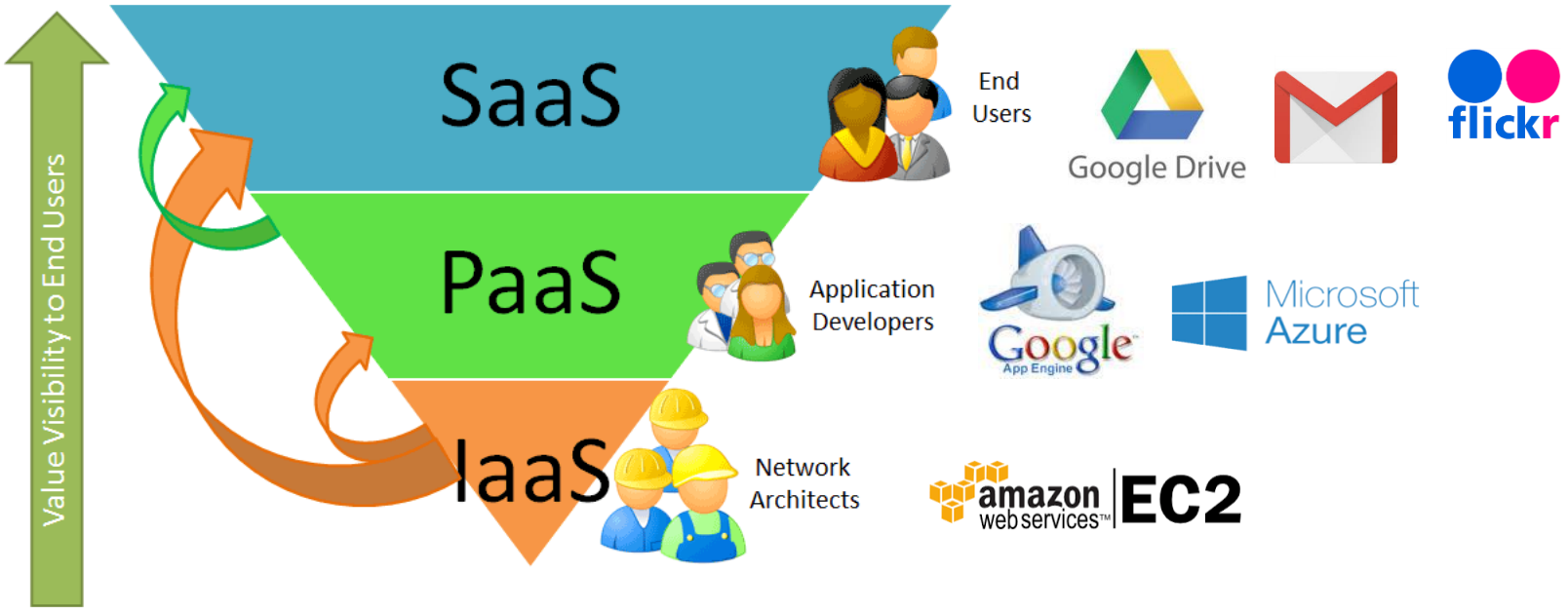
Cloud computing

- "*Cloud computing is a **model** for enabling ubiquitous, convenient, on-demand network **access** to a shared pool of **configurable** computing resources (e.g., networks, servers, storage, applications, and services) that can be **rapidly** provisioned and released with **minimal** management effort or service provider interaction.*" (NIST)

- **Key characteristics**
  1. On demand access
     - Storage, computational, network, applications
  2. Broad network access
  3. Pay-per-use policy
     - Per hour, per minute, per Gb, per request
  4. Resource pooling
     - Virtually unlimited resources
  5. Rapid elasticity
     - Add/remove VM cores and memory, add/remove VMs
  6. New programming paradigms
     - MapReduce, Hadoop, NoSQL (Cassandra, MongoDB), …
  7. Data intensive nature
     - MBs have become TBs, PBs, ….
       - Daily logs, web data, scientific data, …



Cloud stack


Value Visibility to End Users
SaaS
PaaS
IaaS
End Users
Google Drive
flickr
Application Developers
Google App Engine
Microsoft Azure
Network Architects
amazon web services
EC2

How to conceive and optimize clouds and their applications?

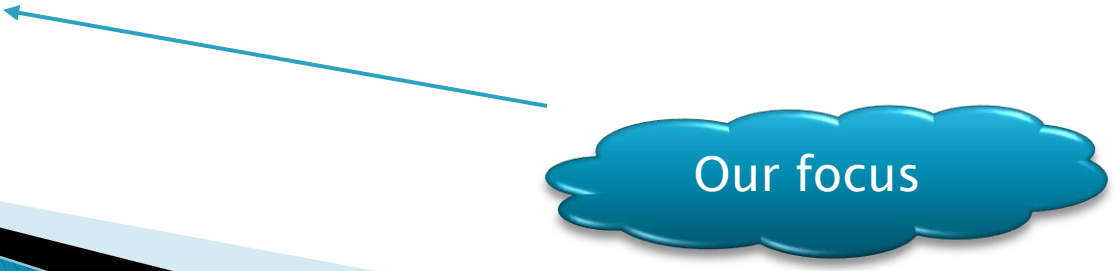
Objectives

- **Correction**: absence of crash, race conditions, deadlocks and other defects, …
- **Performance**: makespan, throughput, economics, energy, ...

See http://simgrid.gforge.inria.fr/tutorials/simgrid-101.pdf for more details

Assessing cloud applications

- Correction study → formal methods
  - **Tests**: unable to provide definitive answers
  - **Model-checking**: exhaustive and automated exploration of state space
- Performance study → experimentation
  - **Math**: often not sufficient to fully understand these systems
  - **Experimental facilities**: real applications on real platform
  - **Emulation**: real applications on synthetic platforms
  - Simulation: prototypes of applications on system models

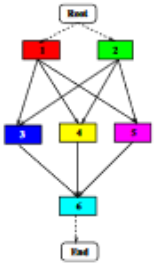


Why simulations?

- **Theoretical studies** are not enough
  - Clouds are highly complex
    - Heterogenous, dynamic behavior, complex platforms, exascale
    - Big Data (volume, variety, velocity, veracity)
- **Simulation** is the *fastest* path from idea to data
- Comfortable for users
  - Get preliminary results from partial implementations
  - Experimental campaign with thousands of runs within the week
  - Test your scientific idea without dealing with technical subtleties
- Challenges for the simulators
  - **Validity**: Get realistic results (controlled experimental bias)
  - **Scalability**: Simulate fast enough problems big enough
  - **Associated tools**: result analysis, settings generation, …
  - **Applicability**: should simulate what is relevant to users

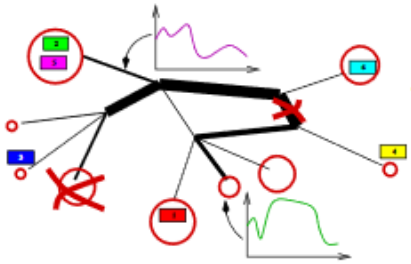
Simulation flow

Idea to test



+

Experimental setup

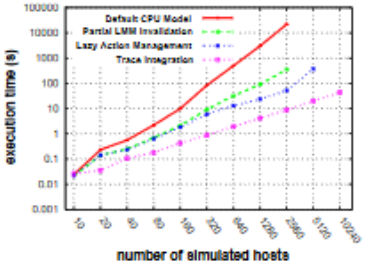


+

Simulation Model

⇒

Scientific results

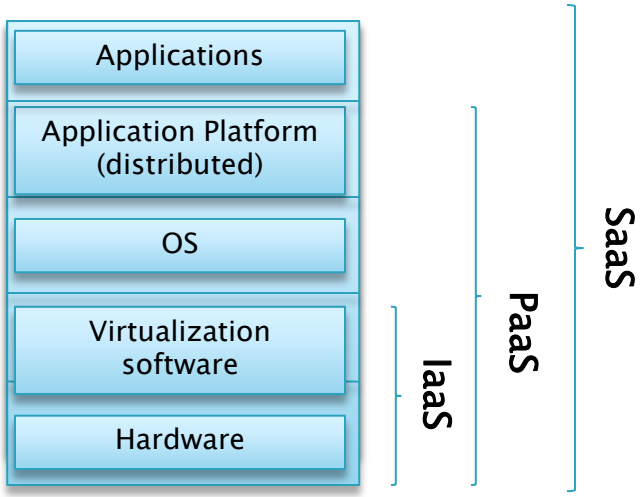


Requirements for a sound simulation

- **Reproducible results**: read a paper, reproduce the results and improve
- **Standard tools** that grad students can learn quickly
- Current **practice** is quite **different**
  - Experimental **settings not detailed** enough in literature
  - Many short-lived simulators with **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, . . .

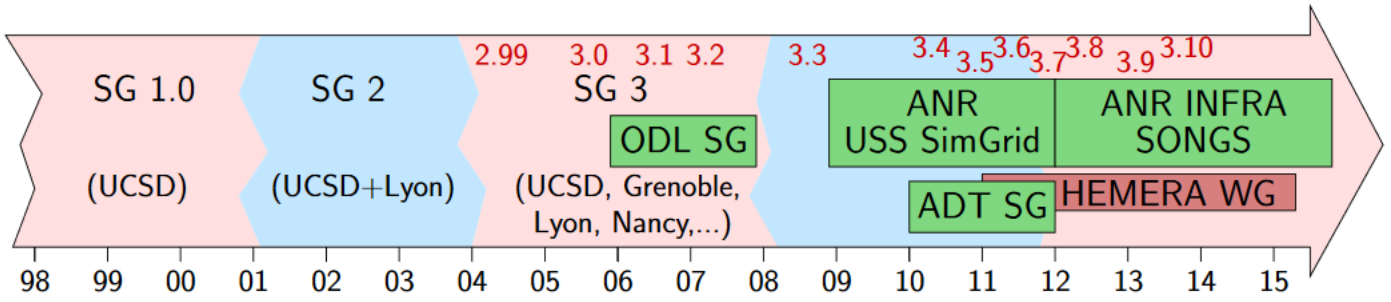
Cloud simulators

- *Remember the cloud stack?*
- **Models** for
  - **Hardware**
    - Network
    - CPU
    - Memory
  - **Virtualization**
    - Hypervisor
  - **OS**
    - OS level abstractions
    - Resource management
  - **Application platform**
    - Cloud level management
  - **Applications**
    - Processes



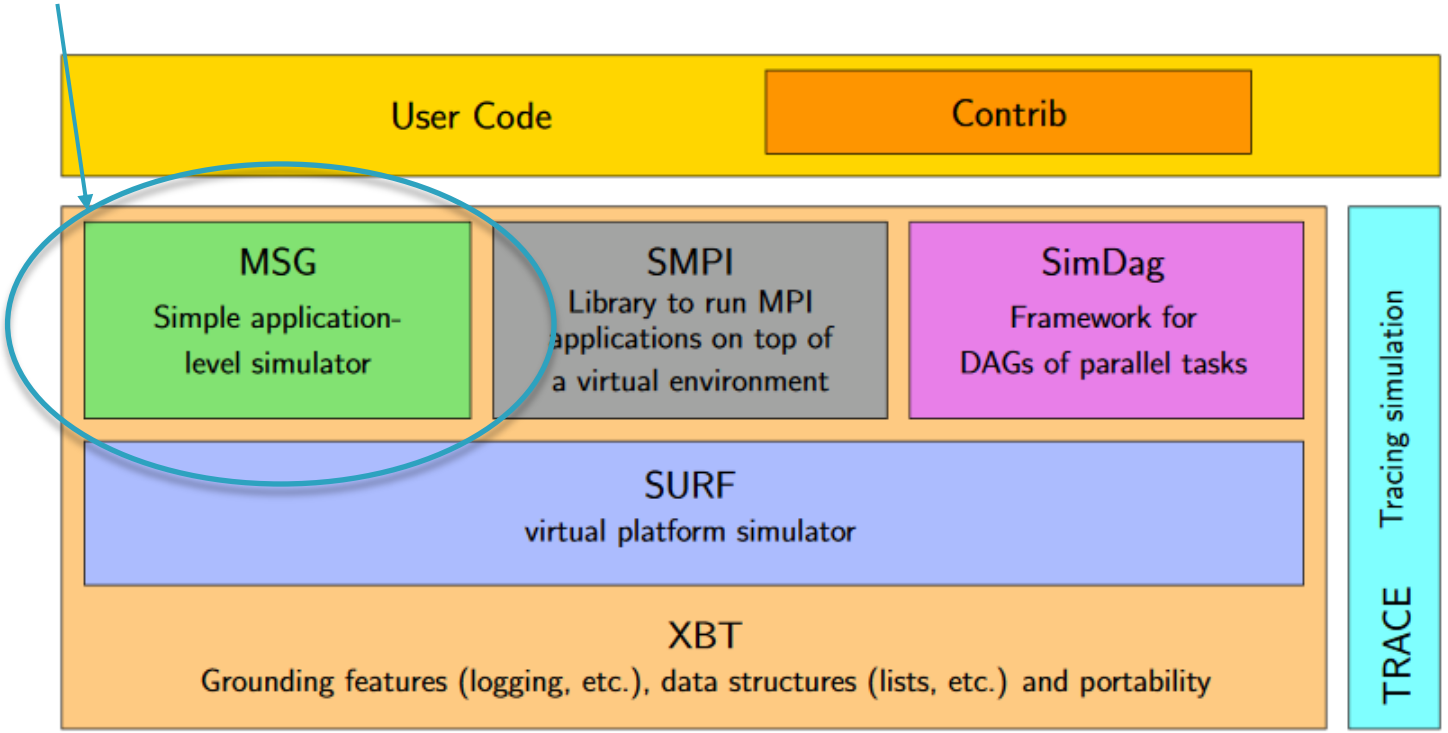
Simgrid as a cloud simulator

- **Scientific instrument**
  - **Versatile**: Grid, P2P, HPC, Volunteer Computing and others
  - **Sound**: Validated, Scalable, Usable; Modular; Portable
  - **Open**: Grounded +100 papers; 100 members on simgrid-user@; LGPL
- **Scientific object** (and lab)
  - Allows **comparison of network models** on non-trivial applications
  - Experimental **Model-Checker**; full Emulator under way



Simgrid's internal stack

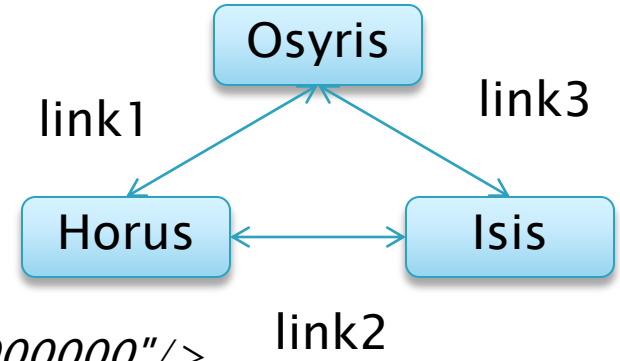
Our focus today



Hardware modeling

- CPU model
  - Define CPU in flops and process size in flops
- Network model
  - Many models, but most realistic
    - Max-Min fairness model between network flows
      - **Objective**: maximize the minimum flow
      - **Equilibrium**: increasing any flow will decrease others
    - Accounts for contention, slow-start, TCP congestion, cross-traffic effects
  - Define network bandwidth and latency, and process bandwidth requirements

Hardware topology



- Static topology

```
<platform version="3">
  <AS  id="AS0"  routing="Full">
    <host id="Horus" core="4" power="8095000000"/>
    <host id="Osyris" core="4" power="8095000000"/>
    <host id="Isis" core="4" power="8095000000"/>

    <link id="link1" bandwidth="125000000" latency="0.000100"/>
    <link id="link2" bandwidth="125000000" latency="0.000100"/>
    <link id="link3" bandwidth="125000000" latency="0.000500"/>

    <route src="Horus" dst="Osyris"><link_ctn id="link1"/></route>
    <route src="Horus" dst="Isis"><link_ctn id="link2"/></route>
    <route src="Osyris" dst="Isis"><link_ctn id="link3"/></route>

  </AS>
</platform>
```

- Dynamic traces for network bandwidth and host CPU speed

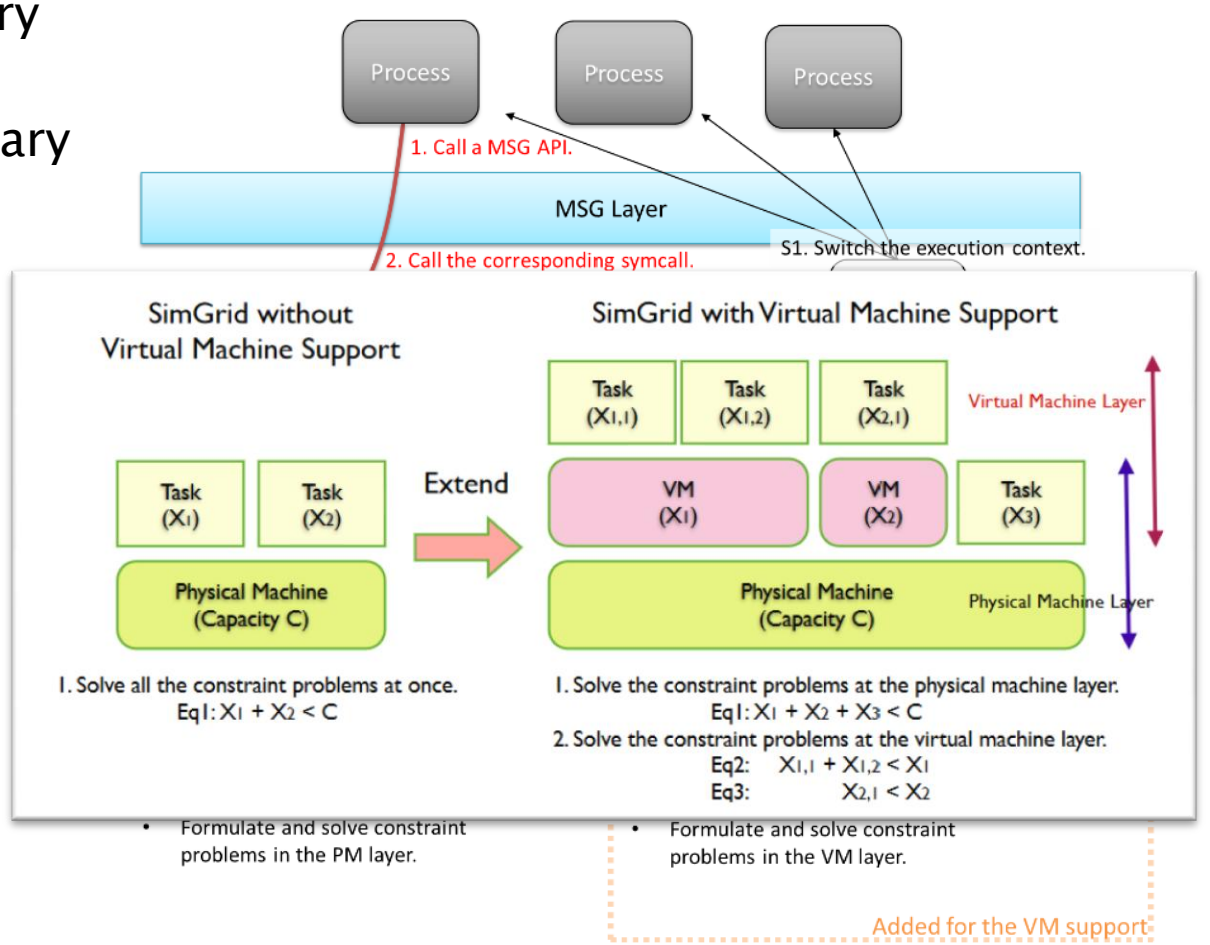
Hypervisor modeling

- **Similar API** as the one for using physical machines
- **VMs controllable** similarly as in the real world
  - Start/suspend/resume/shutdown/migrate
- **Live migration**
  - **Precopy algorithm**
    - Copy memory pages while VM is still running
    - Slow downtime: few ms to secs
  - "*This model **correctly** calculates the **migration time** as well as the **migration traffic**, taking account of resource contention caused by other computations and data exchanges within the whole system. This allows user to **obtain accurate results** of dynamic virtualized systems.*"

See http://simgrid.gforge.inria.fr/contrib/clouds-sg-doc.php

VMs in Simgrid

- VMs are seen as ordinary tasks by the host
- Tasks see VMs as ordinary hosts
- **Resource allocation**
  - Solve first at host level
  - Then solve at VM level

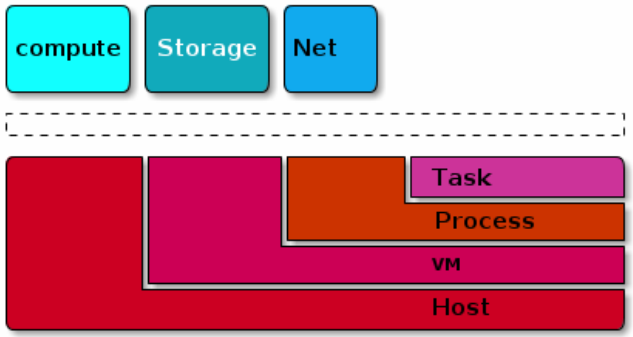


See https://hal.inria.fr/hal-01197274

Application platform modeling

- Cloud platforms
  - **EC2 model** as reference
    - Functions similar to the EC2 API
  - **Users handle instances** not VMs
    - Many instance types
    - Billing models
    - Storage, transfer, compute
    - Automatic VM placement on physical machines

See http://schiaas.gforge.inria.fr/



Cloud “topology”

```
<clouds version="1">
 <cloud  id="myCloud">
  <storage id="myStorage" engine="org.simgrid.schiaas.engine.storage.rise.Rise">
   <config controller="Horus"/>
  </storage>

  <compute engine="org.simgrid.schiaas.engine.compute.rice.Rice">
   <config controller="Horus"
                    image_storage="myStorage"
                    image_caching="PRE“
                    inter_boot_delay="10"/>

   <instance_type id="small" core="1" memory="1000" disk="1690"/>
   <instance_type id="medium" core="2" memory="1000" disk="1690"/>
   <instance_type id="large" core="4" memory="1000" disk="1690"/>

   <image id="myImage" size="1073741824"/>

   <host id="Osyris"/>
   <host id="Isis"/>
  </compute>
 </cloud>
</clouds>
```

Application modeling

- Applications as **processes** that contain a certain number of tasks to execute, have CPU and network requirements and can be executed on several machines

```
<process host="Horus"  function="cloud.schiaas.Master">
    <argument value="10"/>       <!-- Number of tasks -->
    <argument value="5e10"/>  <!-- Computation size of tasks -->
    <argument value="1000000"/>   <!-- Communication size of tasks -->
    <argument value="10"/>  <!-- Number of slave processes -->
 </process>
```

OS modeling

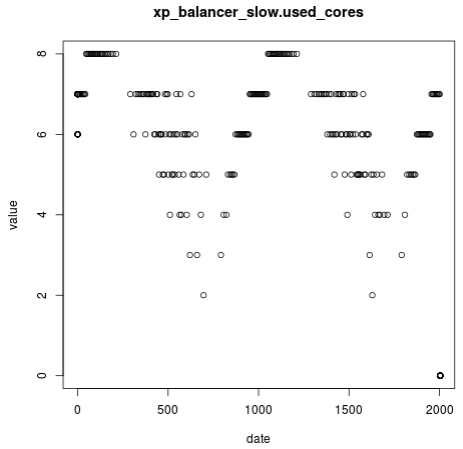
- *We left this one for the end*

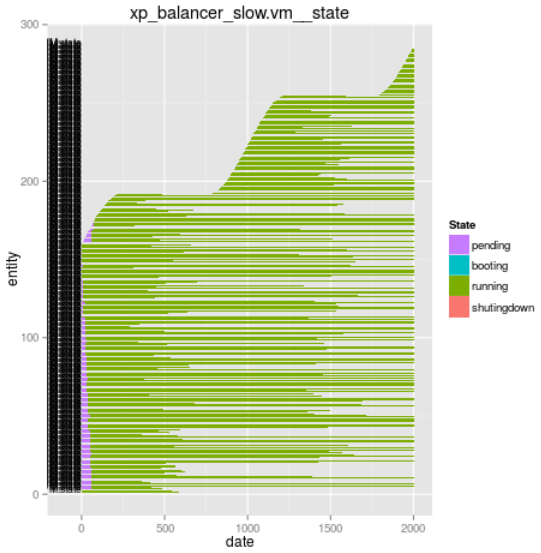
- **Resource management**
  - Cloud (virtualized) level
    - Tasks to VMs
  - Physical level
    - Processes to physical machines
    - VMs to physical machines
  - Users can implement their own **scheduling algorithms**
    - Design, test, validate
    - Load injectors
  - At cloud level by default (built on Simgrid)
    - **RICE**: Reduced Implementation of Compute Engine
    - **RISE**: Reduced Implementation of Storage Engine
    - Tasks2VM: Simschlouder http://schiaas.gforge.inria.fr/javadoc/simschlouder/index.html
    - VMs2Hosts: Schiaas http://schiaas.gforge.inria.fr/javadoc/schiaas/index.html

```
<cloud  id="myCloud">
  <scheduler controller="controller"
             delay="100"
             type="balancer"
             name="org.simgrid.schiaas.engine.compute.scheduler.simplescheduler.SimpleScheduler"/>
</cloud>
```

Simulation visualization

- Scriptable visualization
  - Scalable tools
- Right information
  - Platform and applicative visualizations
- Right representation
  - Gantt charts, spatial representations, tree-graphs
- Easy navigation in space and time
  - Selection, aggregation, animation
- Easy trace comparison





- R, Excel, custom built applications

Thank you!

Questions?