The validation problem on distributed heterogeneous computing platforms: simulation, modeling, observation . . . ?

Arnaud Legrand

Laboratoire de l’Informatique du Parallélisme
École Normale Supérieure de Lyon, France

April 18, 2004
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
Scheduling parallel applications is already a challenging problem on simple homogeneous platforms. On an heterogeneous one, it is even more complicated. Even when an optimal solution to a scheduling problem can be found in polynomial time, small modifications of the underlying assumptions (e.g. addition of non-zero network latencies) often render the problem NP-hard:

\[\leadsto \text{low complexity heuristics}\]

Questions:

- How to compare two different heuristics?
- How to study the flaws of the modeling?
Scheduling parallel applications is already a challenging problem on simple homogeneous platforms. On an heterogeneous one, it is even more complicated.

Even when an optimal solution to a scheduling problem can be found in polynomial time, small modifications of the underlying assumptions (e.g. addition of non-zero network latencies) often render the problem NP-hard:

\[ \leadsto \text{low complexity heuristics} \]

Questions:

- How to compare two different heuristics?
- How to study the flaws of the modeling?
Scheduling parallel applications is already a challenging problem on simple homogeneous platforms. On an heterogeneous one, it is even more complicated. Even when an optimal solution to a scheduling problem can be found in polynomial time, small modifications of the underlying assumptions (e.g. addition of non-zero network latencies) often render the problem NP-hard:

\[ \leadsto \text{low complexity heuristics} \]

Questions:

- How to compare two different heuristics?
- How to study the flaws of the modeling?
Scheduling parallel applications is already a challenging problem on simple homogeneous platforms. On an heterogeneous one, it is even more complicated. Even when an optimal solution to a scheduling problem can be found in polynomial time, small modifications of the underlying assumptions (e.g. addition of non-zero network latencies) often render the problem NP-hard:

\[ \leadsto \text{low complexity heuristics} \]

Questions:

- How to compare two different heuristics?
- How to study the flaws of the modeling?
Scheduling parallel applications is already a challenging problem on simple homogeneous platforms. On an heterogeneous one, it is even more complicated. Even when an optimal solution to a scheduling problem can be found in polynomial time, small modifications of the underlying assumptions (e.g. addition of non-zero network latencies) often render the problem NP-hard:

\[ \leadsto \text{low complexity heuristics} \]

Questions:

- How to compare two different heuristics?
- How to study the flaws of the modeling?
Scheduling parallel applications is already a challenging problem on simple homogeneous platforms. On an heterogeneous one, it is even more complicated. Even when an optimal solution to a scheduling problem can be found in polynomial time, small modifications of the underlying assumptions (e.g. addition of non-zero network latencies) often render the problem NP-hard:

\[ \leadsto \text{low complexity heuristics} \]

Questions:

- How to compare two different heuristics?
- How to study the flaws of the modeling?
Scheduling parallel applications is already a challenging problem on simple homogeneous platforms. On an heterogeneous one, it is even more complicated. Even when an optimal solution to a scheduling problem can be found in polynomial time, small modifications of the underlying assumptions (e.g. addition of non-zero network latencies) often render the problem NP-hard:

\[ \leadsto \text{low complexity heuristics} \]

Questions:

- How to compare two different heuristics?
- How to study the flaws of the modeling?
Real experiments . . .

Traditional parallel computers are somehow stable and rather easy to model. Modern computing platforms are increasingly distributed and often span multiple administrative domains.

▶ Resource availability fluctuations makes it impossible to conduct repeatable experiments for relatively long running applications.

▶ The number of platform configurations that can be explored is limited.
Real experiments . . .

Traditional parallel computers are somehow stable and rather easy to model. Modern computing platforms are increasingly distributed and often span multiple administrative domains.

- Resource availability fluctuations makes it impossible to conduct repeatable experiments for relatively long running applications.
- The number of platform configurations that can be explored is limited.
Real experiments . . .

Traditional parallel computers are somehow stable and rather easy to model. Modern computing platforms are increasingly distributed and often span multiple administrative domains.

- Resource availability fluctuations makes it impossible to conduct repeatable experiments for relatively long running applications.

- The number of platform configurations that can be explored is limited.
Traditional parallel computers are somehow stable and rather easy to model.
Modern computing platforms are increasingly distributed and often span multiple administrative domains.

- Resource availability fluctuations makes it *impossible to conduct repeatable experiments* for relatively long running applications.
- The number of platform configurations that can be explored is *limited*. 
Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are repeatable, configurable, and generally fast. But...

▶ No standard: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computer and network behavior are no longer valid for modern platforms.

▶ Need for realistic and more complex models than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non identical and unreliable processors and links?
Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are repeatable, configurable, and generally fast. But...

- No standard: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern platforms.

- Need for realistic and more complex models than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non identical and unreliable processors and links?
Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are **repeatable**, **configurable**, and generally **fast**. But...

- No standard: "throw-away" simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern plat- 

- Need for realistic and more complex models than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource available-

How to model a distributed computing platform made of thousands of non identical and unreliable processors and links ?
Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are repeatable, configurable, and generally fast. But...

- No standard: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern platforms.

- Need for realistic and more complex models than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non identical and unreliable processors and links?
Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are **repeatable**, **configurable**, and generally **fast**. But...

- No standard: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern platforms.

- Need for realistic and more complex models than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non-identical and unreliable processors and links?
or simulation

Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are repeatable, configurable, and generally fast. But...

- **No standard**: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern platforms.

- **Need for realistic and more complex models than the one used for designing algorithms.** The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non identical and unreliable processors and links?
Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are repeatable, configurable, and generally fast. But...

- **No standard**: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern platforms.

- **Need for realistic and more complex models** than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non-identical and unreliable processors and links?
Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are repeatable, configurable, and generally fast. But...

- **No standard**: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern platforms.

- **Need for realistic and more complex models** than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non identical and unreliable processors and links?
Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are repeatable, configurable, and generally fast. But...

- **No standard**: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern platforms.

- **Need for realistic and more complex models** than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non identical and unreliable processors and links?
... or simulation

Simulation has been used extensively as a way to evaluate and compare scheduling strategies as simulation experiments are repeatable, configurable, and generally fast. But...

▶ No standard: “throw-away” simulators make it difficult to reproduce results. This lack of standard simulation procedure and software was somewhat justifiable when the simulation models in use were simplistic but traditional models and assumptions about computing platforms are no longer valid for modern platforms.

▶ Need for realistic and more complex models than the one used for designing algorithms. The assumption that the behavior of the computing platform is perfectly predictable also needs to be revisited as modern platforms exhibit dynamic resource availabilities.

How to model a distributed computing platform made of thousands of non identical and unreliable processors and links?
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
Network Simulators

**Goal:**
- Understanding networks behavior, routing protocols, QoS, ...
- Identifying limitations of network protocols and developing improvements.

- Requires a precise simulation of the movement of packets along the network links: NS [ns, BBE\textsuperscript{+99}], DaSSF [LN01], OMNeT++ [OMN].

Inadequate

We are interested by the network behavior as it is experienced by an application.

- Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).
- Adding CPU resources to model applications using the network is labor-intensive.
- External background load is generally done by using additional random connections, hence a longer simulation time.
Network Simulators

**Goal**: ▶ understanding networks behavior, routing protocols, QoS, ...

▶ identifying limitations of network protocols and developing improvements.

requires a precise simulation of the movement of packets along the network links: NS [ns, BBE+99], DaSSF [LN01], OMNeT++ [OMN].

Inadequate

We are interested by the network behavior as it is experienced by an application.

▶ Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).

▶ Adding CPU resources to model applications using the network is labor-intensive.

▶ External background load is generally done by using additional random connections, hence a longer simulation time.
Network Simulators

Goal:

- understanding networks behavior, routing protocols, QoS, ...
- identifying limitations of network protocols and developing improvements.

requires a precise simulation of the movement of packets along the network links: NS [ns, BBE+99], DaSSF [LN01], OMNeT++ [OMN].

Inadequate

We are interested by the network behavior as it is experienced by an application.

- Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).
- Adding CPU resources to model applications using the network is labor-intensive.
- External background load is generally done by using additional random connections, hence a longer simulation time.
Network Simulators

Goal:

▶ understanding networks behavior, routing protocols, QoS, . . .

▶ identifying limitations of network protocols and developing improvements.

requires a precise simulation of the movement of packets along the network links: NS [ns, BBE+99], DaSSF [LN01], OMNeT++ [OMN].

Inadequate

We are interested by the network behavior as it is experienced by an application.

▶ Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).

▶ Adding CPU resources to model applications using the network is labor-intensive.

▶ External background load is generally done by using additional random connections, hence a longer simulation time.
Network Simulators

**Goal:**
- understanding networks behavior, routing protocols, QoS, ...
- identifying limitations of network protocols and developing improvements.

leads to requires a precise simulation of the movement of packets along the network links: NS [ns, BBE+99], DaSSF [LN01], OMNeT++ [OMN].

**Inadequate**

We are interested by the network behavior as it is experienced by an application.

- Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).
- Adding CPU resources to model applications using the network is labor-intensive.
- External background load is generally done by using additional random connections, hence a longer simulation time.
Network Simulators

**Goal:**
- understanding networks behavior, routing protocols, QoS, ...
- identifying limitations of network protocols and developing improvements.

requires a precise simulation of the movement of packets along the network links: NS [ns, BBE+99], DaSSF [LN01], OMNeT++ [OMN].

**Inadequate**

We are interested by the network behavior as it is experienced by an application.

- Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).
- Adding CPU resources to model applications using the network is labor-intensive.
- External background load is generally done by using additional random connections, hence a longer simulation time.
Network Simulators

Goal:
► understanding networks behavior, routing protocols, QoS, ...
► identifying limitations of network protocols and developing improvements.

requires a precise simulation of the movement of packets along the network links: NS [ns, BBE+99], DaSSF [LN01], OMNeT++ [OMN].

Inadequate

We are interested by the network behavior as it is experienced by an application.

► Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).

► Adding CPU resources to model applications using the network is labor-intensive.

► External background load is generally done by using additional random connections, hence a longer simulation time.
Network Simulators

**Goal:**
- understanding networks behavior, routing protocols, QoS, ...
- identifying limitations of network protocols and developing improvements.

leadsto

requires a precise simulation of the movement of packets along the network links: NS [ns, BBE+99], DaSSF [LN01], OMNeT++ [OMN].

Inadequate

We are interested by the network behavior as it is experienced by an application.

- Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).
- Adding CPU resources to model applications using the network is labor-intensive.
- External background load is generally done by using additional random connections, hence a longer simulation time.
Network Simulators

Goal: ▶ understanding networks behavior, routing protocols, QoS, ...
▶ identifying limitations of network protocols and developing improvements.

requires a precise simulation of the movement of packets along the network links: NS [ns, BBE+99], DaSSF [LN01], OMNeT++ [OMN].

Inadequate

We are interested by the network behavior as it is experienced by an application.

▶ Due to their highly detailed simulation models, most network simulators induce long simulation times (e.g. they implement the TCP stack).
▶ Adding CPU resources to model applications using the network is labor-intensive.
▶ External background load is generally done by using additional random connections, hence a longer simulation time.
Platform Emulation

A few examples:

MicroGrid [SLJ+00] (UCSD)
- The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
- Network simulation is handled through DaSSF [LN01]
- No external load for the network.

PANDA [KBM+02] (Amsterdam)
- Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
- No external load for processors.

The code is run for real ⇨ too slow, too “precise”, too difficult for simple tests or the design phase.
Platform Emulation

A few examples:

**MicroGrid [SLJ+00] (UCSD)**
- The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
- Network simulation is handled through DaSSF [LN01]
- No external load for the network.

**PANDA [KBM+02] (Amsterdam)**
- Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
- No external load for processors.

The code is *run for real* → too slow, too “precise”, too difficult for simple tests or the design phase.
Platform Emulation

A few examples:

MicroGrid [SLJ+00] (UCSD)

- The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
- Network simulation is handled through DaSSF [LN01]
- No external load for the network.

PANDA [KBM+02] (Amsterdam)

- Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
- No external load for processors.

The code is run for real \(\rightarrow\) too slow, too “precise”, too difficult for simple tests or the design phase.
Platform Emulation

A few examples:

MicroGrid [SLJ+00] (UCSD)

- The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
- Network simulation is handled through DaSSF [LN01]
- No external load for the network.

PANDA [KBM+02] (Amsterdam)

- Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
- No external load for processors.

The code is run for real \(\leadsto\) too slow, too “precise”, too difficult for simple tests or the design phase.
Platform Emulation

A few examples:

MicroGrid [SLJ+00] (UCSD)
▶ The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
▶ Network simulation is handled through DaSSF [LN01]
▶ No external load for the network.

PANDA [KBM+02] (Amsterdam)
▶ Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
▶ No external load for processors.

The code is run for real ~ too slow, too “precise”, too difficult for simple tests or the design phase.
Platform Emulation

A few examples:

MicroGrid \[\text{SLJ}^+00\] (UCSD)

- The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
- Network simulation is handled through DaSSF [LN01]
- No external load for the network.

PANDA \[\text{KBM}^+02\] (Amsterdam)

- Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
- No external load for processors.

The code is *run for real* \(\Rightarrow\) too slow, too “precise”, too difficult for simple tests or the design phase.
Platform Emulation

A few examples:

**MicroGrid [SLJ⁺00] (UCSD)**
- The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
- Network simulation is handled through DaSSF [LN01]
- No external load for the network.

**PANDA [KBM⁺02] (Amsterdam)**
- Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
- No external load for processors.

The code is *run for real* → too slow, too “precise”, too difficult for simple tests or the design phase.
Platform Emulation

A few examples:

MicroGrid [SLJ⁺00] (UCSD)

- The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
- Network simulation is handled through DaSSF [LN01]
- No external load for the network.

PANDA [KBM⁺02] (Amsterdam)

- Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
- No external load for processors.

The code is *run for real* → too slow, too “precise”, too difficult for simple tests or the design phase.
Platform Emulation

A few examples:

MicroGrid $[\text{SLJ}^+\text{00}]$ (UCSD)
- The computing platform is mapped onto a fast cluster: a fraction of CPU is allocated to each process according to the speed and the load of the simulated host.
- Network simulation is handled through DaSSF $[\text{LN01}]$
- No external load for the network.

PANDA $[\text{KBM}^+\text{02}]$ (Amsterdam)
- Two-level grid (High speed LAN or slow WAN) and no processor heterogeneity: one-to-one mapping of the computing platform on a cluster; virtual inter-cluster links are artificially slowed down.
- No external load for processors.

The code is *run for real* $\leadsto$ too slow, too “precise”, too difficult for simple tests or the design phase.
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

**SIMGRID**, a modular trace-based simulator

Obtaining a realistic platform model
  - Random topology
  - Real topology
  - Getting traces

Conclusion
History:

- Application Level Scheduling (AppLeS): to a given application corresponds a given scheduler. Many students have been working on scheduling on the grid with specific needs.

- From these experiences, Henri Casanova (UCSD) designed a minimal set of low-level basic functions essential for building a simulator that uses traces: SG (SimGrid v.1)

- MSG is a simulator built on top of SG and adapted to the study of non-centralized scheduling (SimGrid v.2). Simulation is described in terms of communicating processes.

Strong points:

- Ability to use complex and realistic platforms.

- Fast simulations: ratio \( \frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6} \).
History:

- Application Level Scheduling (AppLeS): to a given application corresponds a given scheduler. Many students have been working on scheduling on the grid with specific needs.

- From these experiences, Henri Casanova (UCSD) designed a minimal set of low-level basic functions essential for building a simulator that uses traces: SG (SIMGRID v.1)

- MSG is a simulator built on top of SG and adapted to the study of non-centralized scheduling (SIMGRID v.2). Simulation is described in terms of communicating processes.

Strong points:

- Ability to use complex and realistic platforms.

- Fast simulations: ratio \( \frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6} \).
History:

- Application Level Scheduling (AppLeS): to a given application corresponds a given scheduler. Many students have been working on scheduling on the grid with specific needs.

- From these experiences, Henri Casanova (UCSD) designed a minimal set of low-level basic functions essential for building a simulator that uses traces: SG (SIMGRID v.1)

- MSG is a simulator built on top of SG and adapted to the study of non-centralized scheduling (SIMGRID v.2). Simulation is described in terms of communicating processes.

Strong points:

- Ability to use complex and realistic platforms.

- Fast simulations: ratio \( \frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6} \).
History:

- Application Level Scheduling (AppLeS): to a given application corresponds a given scheduler. Many students have been working on scheduling on the grid with specific needs.

- From these experiences, Henri Casanova (UCSD) designed a minimal set of low-level basic functions essential for building a simulator that uses traces: SG (SIMGRID v.1).

- MSG is a simulator built on top of SG and adapted to the study of non-centralized scheduling (SIMGRID v.2). Simulation is described in terms of communicating processes.

Strong points:

- Ability to use complex and realistic platforms.

- Fast simulations: ratio $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$. 
History:

- Application Level Scheduling (AppLeS): to a given application corresponds a given scheduler. Many students have been working on scheduling on the grid with specific needs.
- From these experiences, Henri Casanova (UCSD) designed a minimal set of low-level basic functions essential for building a simulator that uses traces: SG (SimGrid v.1)
- MSG is a simulator built on top of SG and adapted to the study of non-centralized scheduling (SimGrid v.2). Simulation is described in terms of communicating processes.

Strong points:

- Ability to use complex and realistic platforms.
- Fast simulations: ratio \( \frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6} \).
History:

- Application Level Scheduling (AppLeS): to a given application corresponds a given scheduler. Many students have been working on scheduling on the grid with specific needs.
- From these experiences, Henri Casanova (UCSD) designed a minimal set of low-level basic functions essential for building a simulator that uses traces: SG (SimGrid v.1)
- MSG is a simulator built on top of SG and adapted to the study of non-centralized scheduling (SimGrid v.2). Simulation is described in terms of communicating processes.

Strong points:

- Ability to use complex and realistic platforms.
- Fast simulations: ratio $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$. 
History:

- Application Level Scheduling (AppLeS): to a given application corresponds a given scheduler. Many students have been working on scheduling on the grid with specific needs.

- From these experiences, Henri Casanova (UCSD) designed a minimal set of low-level basic functions essential for building a simulator that uses traces: SG (SimGrid v.1)

- MSG is a simulator built on top of SG and adapted to the study of non-centralized scheduling (SimGrid v.2). Simulation is described in terms of communicating processes.

Strong points:

- Ability to use complex and realistic platforms.

- Fast simulations: ratio $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$. 
SG: objects

A trace is a time-stamped series of values.
Two different types: resources and tasks.

SG RESOURCE Name, availability trace (CPU, bandwidth), time access trace (latency), sharing policy (sequential, shared, TCP).

SG TASK Name, amount of work

SG allows to create those objects and to schedule a task on a resource.

▶ Starting a transfer of $S$ bytes on a resource at time $t_0$ requires $T$ units of time with $T$ s.a.:

$$\int_{t_0}^{t_0+T} B(t) dt = S$$

▶ On shared resources, all task get an amount of power proportional to their priority.
SG: objects

A trace is a time-stamped series of values.

Two different types: resources and tasks.

**SG_Resource** Name, availability trace (CPU, bandwidth), time access trace (latency), sharing policy (sequential, shared, TCP).

**SG_Task** Name, amount of work

SG allows to create those objects and to schedule a task on a resource.

- Starting a transfer of $S$ bytes on a resource at time $t_0$ requires $T$ units of time with T.s.a.:
  $$\int_{t=t_0+L(t_0)}^{t_0+T} B(t) dt = S$$

- On shared resources, all task get an amount of power proportional to their priority.
**SG : objects**

A trace is a time-stamped series of values.

Two different types: resources and tasks.

**SG_Resource** Name, availability trace (CPU, bandwidth), time access trace (latency), sharing policy (sequential, shared, TCP).

**SG_Task** Name, amount of work

SG allows to create those objects and to schedule a task on a resource.

- Starting a transfer of $S$ bytes on a resource at time $t_0$ requires $T$ units of time with $T$ s.a.:
  \[
  \int_{t=t_0+L(t_0)}^{t_0+T} B(t) \, dt = S
  \]

- On shared resources, all task get an amount of power proportional to their priority.
SG : objects

A trace is a time-stamped series of values. Two different types: resources and tasks.

**SG_Resource** Name, availability trace (CPU, bandwidth), time access trace (latency), sharing policy (sequential, shared, TCP).

**SG_Task** Name, amount of work

SG allows to create those objects and to schedule a task on a resource.

- Starting a transfer of $S$ bytes on a resource at time $t_0$ requires $T$ units of time with $T$ s.a.:
  \[
  \int_{t=t_0+L(t_0)}^{t_0+T} B(t) dt = S
  \]

- On shared resources, all task get an amount of power proportional to their priority.
A trace is a time-stamped series of values. Two different types: resources and tasks.

**SG_Resource** Name, availability trace (CPU, bandwidth), time access trace (latency), sharing policy (sequential, shared, TCP).

**SG_Task** Name, amount of work

SG allows to create those objects and to schedule a task on a resource.

- Starting a transfer of $S$ bytes on a resource at time $t_0$ requires $T$ units of time with $T$ s.a.:

\[
\int_{t=t_0+L(t_0)}^{t_0+T} B(t) dt = S
\]

- On shared resources, all task get an amount of power proportional to their priority.
SG : objects

A trace is a time-stamped series of values.
Two different types: resources and tasks.

**SG_Resource** Name, availability trace (CPU, bandwidth), time access trace (latency), sharing policy (sequential, shared, TCP).

**SG_Task** Name, amount of work

SG allows to create those objects and to schedule a task on a resource.

- Starting a transfer of $S$ bytes on a resource at time $t_0$ requires $T$ units of time with $T$ s.a.:

  $$\int_{t=t_0+L(t_0)}^{t_0+T} B(t) \, dt = S$$

- On shared resources, all task get an amount of power proportional to their priority.
SG: objects

A trace is a time-stamped series of values.
Two different types: resources and tasks.

**SG_Resource** Name, availability trace (CPU, bandwidth), time access trace (latency), sharing policy (sequential, shared, TCP).

**SG_Task** Name, amount of work

SG allows to create those objects and to schedule a task on a resource.

- Starting a transfer of $S$ bytes on a resource at time $t_0$ requires $T$ units of time with $T$ s.a.:
  $$\int_{t=t_0+L(t_0)}^{t_0+T} B(t) dt = S$$

- On shared resources, all task get an amount of power proportional to their priority.
Using some traces

Ordonnanceur

$P_1$

$P_2$
Using some traces

Ordonnanceur

Schedule($T_1, P_1$)
Schedule($T_2, P_2$)
Simulate(10 s)

$P_1$

$P_2$
Using some traces

Ordonnanceur

Schedule($T_1$, $P_1$)
Schedule($T_2$, $P_2$)
Simulate(10 s)
Using some traces

GetPrediction($P_1$)
GetPrediction($P_2$)
Schedule($T_3,P_1$)
Simulate(ANY_TASK)

Ordonnanceur
Using some traces

GetPrediction($P_1$)
GetPrediction($P_2$)
Schedule($T_3$, $P_1$)
Simulate(ANY_TASK)
Using some traces
Using some traces

Ordonnanceur

Simulate()
Using some traces

Ordonnanceur

$P_1$

$P_2$
Using some traces
Using some traces
PSTS\textsubscript{im}, a centralized scheduler

Given a master, some slaves, and a set of independent tasks that may share some input files.

- Some communication tasks are scheduled on network links and some computation tasks are scheduled on CPUs while respecting the dependencies.
- The simulation is run until some task is finished. From the state of the platform, the scheduling algorithm schedules some new tasks, a.s.o.
**PSTS\textsubscript{im}, a centralized scheduler**

Given a master, some slaves, and a set of independent tasks that may share some input files.

- Some communication tasks are scheduled on network links and some computation tasks are scheduled on CPUs while respecting the dependencies.

- The simulation is run until some task is finished. From the state of the platform, the scheduling algorithm schedules some new tasks, a.s.o.
PSTS\textsubscript{im}, a centralized scheduler

Given a master, some slaves, and a set of independent tasks that may share some input files.

- Some communication tasks are scheduled on network links and some computation tasks are scheduled on CPUs while respecting the dependencies.

- The simulation is run until some task is finished. From the state of the platform, the scheduling algorithm schedules some new tasks, a.s.o.
PSTS$_{\text{im}}$, a centralized scheduler

Given a master, some slaves, and a set of independent tasks that may share some input files.

- Some communication tasks are scheduled on network links and some computation tasks are scheduled on CPUs while respecting the dependencies.

- The simulation is run until some task is finished. From the state of the platform, the scheduling algorithm schedules some new tasks, a.s.o.
Store & Forward, WormHole, TCP

How to model a file transfer along a path?
How to model a file transfer along a path?

Store & Forward, WormHole, TCP
How to model a file transfer along a path?

Store & Forward, WormHole, TCP

Store & Forward: bad model for contention
How to model a file transfer along a path?

\[ S \]

\[ l_1 \]

\[ l_2 \]

\[ l_3 \]
How to model a file transfer along a path?

WormHole: computation intensive (packets), not that realistic
How to model a file transfer along a path?

\[ \forall l \in \mathcal{L}, \quad \sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leq c_l, \]

Analytical model
How to model a file transfer along a path?

\[ \forall l \in L, \sum_{r \in R \text{ s.a. } l \in r} \rho_r \leq c_l, \]

Max-Min Fairness [BG87] maximize

\[ \min_{r \in R} \rho_r. \]
How to model a file transfer along a path?

\[ \forall l \in \mathcal{L}, \quad \sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leq c_l, \]

Max-Min Fairness [BG87] maximize \( \min_{r \in \mathcal{R}} \rho_r \).

Proportional Fairness [Kel97] maximize \( \sum_{r \in \mathcal{R}} \rho_r \log(\rho_r) \).
How to model a file transfer along a path?

∀ \( l \in \mathcal{L} \),
\[
\sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leq c_l,
\]

Max-Min Fairness [BG87] maximize \( \min_{r \in \mathcal{R}} \rho_r \).

Proportional Fairness [Kel97] maximize \( \sum_{r \in \mathcal{R}} \rho_r \log(\rho_r) \).

MCT minimization maximize \( \sum_{r \in \mathcal{R}} \frac{1}{\rho_r} \).
How to model a file transfer along a path?

\[
\forall l \in \mathcal{L}, \quad \sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leq c_l,
\]

**Max-Min Fairness** [BG87] maximize \(\min_{r \in \mathcal{R}} \rho_r\).

**Proportional Fairness** [Kel97] maximize \(\sum_{r \in \mathcal{R}} \rho_r \log(\rho_r)\).

**MCT minimization** maximize \(\sum_{r \in \mathcal{R}} \frac{1}{\rho_r}\).

**TCP behavior** [CM02] Close to max-min. In MSG: max-min + bound by \(\frac{1}{RTT}\).
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.

MSG abstractions:

Agents some code, private data, and the location at which it executes;

Locations a computational resource, a number of mailboxes that enable communication with other agents, and private data that can be only accessed by agents at the same location;

Task an amount of computing, a data size, and private data;

Path a set of network links used to transfer a task from a location to another location;

Channel mailbox number.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.

MSG abstractions:

**Agents** some code, private data, and the location at which it executes;

**Locations** a computational resource, a number of mailboxes that enable communication with other agents, and private data that can be only accessed by agents at the same location;

**Task** an amount of computing, a data size, and private data;

**Path** a set of network links used to transfer a task from a location to another location;

**Channel** mailbox number.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.

MSG abstractions:

**Agents** some code, private data, and the location at which it executes;

**Locations** a computational resource, a number of mailboxes that enable communication with other agents, and private data that can be only accessed by agents at the same location;

**Task** an amount of computing, a data size, and private data;

**Path** a set of network links used to transfer a task from a location to another location;

**Channel** mailbox number.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.

MSG abstractions:

**Agents** some code, private data, and the location at which it executes;

**Locations** a computational resource, a number of mailboxes that enable communication with other agents, and private data that can be only accessed by agents at the same location;

**Task** an amount of computing, a data size, and private data;

**Path** a set of network links used to transfer a task from a location to another location;

**Channel** mailbox number.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.

MSG abstractions:

**Agents** some code, private data, and the location at which it executes;

**Locations** a computational resource, a number of mailboxes that enable communication with other agents, and private data that can be only accessed by agents at the same location;

**Task** an amount of computing, a data size, and private data;

**Path** a set of network links used to transfer a task from a location to another location;

**Channel** mailbox number.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.

MSG abstractions:

Agents some code, private data, and the location at which it executes;

Locations a computational resource, a number of mailboxes that enable communication with other agents, and private data that can be only accessed by agents at the same location;

Task an amount of computing, a data size, and private data;

Path a set of network links used to transfer a task from a location to another location;

Channel mailbox number.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.

\[
P_0 \quad P_1 \quad P_2 \quad P_3 \quad P_4
\]

- Grid Operating System
- Grid Information Service
- Forecasting Service

SG : Event Manager
Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.
Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.
Non-centralized scheduling?

Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.
Basic MSG functions

Object creation
- MSG_host_create
- MSG_link_create
- MSG_process_create
- MSG_task_create

Agent basic actions
- MSG_task_get
- MSG_task_put
- MSG_task_execute

Agent additional actions
- MSG_task_Iprobe
- MSG_process_sleep
- MSG_process_suspend
- MSG_process_resume
Basic MSG functions

Object creation
- MSG_host_create
- MSG_link_create
- MSG_process_create
- MSG_task_create

Agent basic actions
- MSG_task_get
- MSG_task_put
- MSG_task_execute

Agent additional actions
- MSG_task_Iprobe
- MSG_process_sleep
- MSG_process_suspend
- MSG_process_resume
Basic MSG functions

Object creation
- MSG_host_create
- MSG_link_create
- MSG_process_create
- MSG_task_create

Agent basic actions
- MSG_task_get
- MSG_task_put
- MSG_task_execute

Agent additional actions
- MSG_task_Iprobe
- MSG_process_sleep
- MSG_process_suspend
- MSG_process_resume
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
Building a platform is a pain

Realistic platforms are complex and building such platforms is generally fastidious since it requires to create a large number of elements:

- Hosts
- Links
- Routing
- Traces

Different ways to automatically build a platform

- Random topology
- Real topology
- Getting traces
Building a platform is a pain

Realistic platforms are complex and building such platforms is generally fastidious since it requires to create a large number of elements:

- Hosts
- Links
- Routing
- Traces

Different ways to automatically build a platform

- Random topology
- Real topology
- Getting traces
Building a platform is a pain

Realistic platforms are complex and building such platforms is generally fastidious since it requires to create a large number of elements:

- Hosts
- Links
- Routing
- Traces

Different ways to automatically build a platform

- Random topology
- Real topology
- Getting traces
Building a platform is a pain

Realistic platforms are complex and building such platforms is generally fastidious since it requires to create a large number of elements:

- Hosts
- Links
- Routing
- Traces

Different ways to automatically build a platform

- Random topology
- Real topology
- Getting traces
Building a platform is a pain

Realistic platforms are complex and building such platforms is generally fastidious since it requires to create a large number of elements:

- Hosts
- Links
- Routing
- Traces

Different ways to automatically build a platform

- Random topology
- Real topology
- Getting traces
**Flat models**

**Brain-dead**\(^N\) dots are randomly chosen (using a uniform distribution) in a square. Then they are randomly connected with a uniform probability \(\alpha\).

**Waxman** [Wax88] Dots are randomly placed on a square of side \(c\) and are randomly connected with a probability \(P(u, v) = \alpha e^{-d/(\beta L)}\), \(0 < \alpha, \beta \leq 1\) where \(d\) is the Euclidean distance between \(u\) and \(v\) and \(L = c\sqrt{2}\). The edge number increases with \(\alpha\) and the edge length heterogeneity increases with \(\beta\).

**Exponential** Dots are randomly placed and are connected with a probability \(P(u, v) = \alpha e^{-d/(L-d)}\).

**Locality** [ZCD97] This model is due to Zegura. Dots are randomly placed and are connected with a probability

\[
P(u, v) = \begin{cases} 
\alpha & \text{if } d < L \times r \\
\beta & \text{if } d \geq L \times r 
\end{cases}
\]
Flat models

**Brain-dead** $N$ dots are randomly chosen (using a uniform distribution) in a square. Then they are randomly connected with a uniform probability $\alpha$.

**Waxman** [Wax88] Dots are randomly placed on a square of side $c$ and are randomly connected with a probability $P(u, v) = \alpha e^{-d/(\beta L)}$, $0 < \alpha, \beta \leq 1$ where $d$ is the Euclidean distance between $u$ and $v$ and $L = c\sqrt{2}$. The edge number increases with $\alpha$ and the edge length heterogeneity increases with $\beta$.

**Exponential** Dots are randomly placed and are connected with a probability $P(u, v) = \alpha e^{-d/(L-d)}$.

**Locality** [ZCD97] This model is due to Zegura. Dots are randomly placed and are connected with a probability

$$P(u, v) = \begin{cases} \alpha & \text{if } d < L \times r \\ \beta & \text{if } d \geq L \times r \end{cases}.$$
Flat models

Brain-dead $N$ dots are randomly chosen (using a uniform distribution) in a square. Then they are randomly connected with a uniform probability $\alpha$.

Waxman [Wax88] Dots are randomly placed on a square of side $c$ and are randomly connected with a probability $P(u, v) = \alpha e^{-d/(\beta L)}$, $0 < \alpha, \beta \leq 1$ where $d$ is the Euclidean distance between $u$ and $v$ and $L = c\sqrt{2}$. The edge number increases with $\alpha$ and the edge length heterogeneity increases with $\beta$.

Exponential Dots are randomly placed and are connected with a probability $P(u, v) = \alpha e^{-d/(L-d)}$.

Locality [ZCD97] This model is due to Zegura. Dots are randomly placed and are connected with a probability

$$P(u, v) = \begin{cases} \alpha & \text{if } d < L \times r \\ \beta & \text{if } d \geq L \times r \end{cases}.$$
Flat models

Brain-dead $N$ dots are randomly chosen (using a uniform distribution) in a square. Then they are randomly connected with a uniform probability $\alpha$.

Waxman [Wax88] Dots are randomly placed on a square of side $c$ and are randomly connected with a probability $P(u, v) = \alpha e^{-d/(\beta L)}$, $0 < \alpha, \beta \leq 1$ where $d$ is the Euclidean distance between $u$ and $v$ and $L = c\sqrt{2}$. The edge number increases with $\alpha$ and the edge length heterogeneity increases with $\beta$.

Exponential Dots are randomly placed and are connected with a probability $P(u, v) = \alpha e^{-d/(L-d)}$.

Locality [ZCD97] This model is due to Zegura. Dots are randomly placed and are connected with a probability

$$P(u, v) = \begin{cases} 
\alpha & \text{if } d < L \times r \\
\beta & \text{if } d \geq L \times r 
\end{cases}$$
Flat models

Brain-dead $N$ dots are randomly chosen (using a uniform distribution) in a square. Then they are randomly connected with a uniform probability $\alpha$.

Waxman [Wax88] Dots are randomly placed on a square of side $c$ and are randomly connected with a probability $P(u, v) = \alpha e^{-d/(\beta L)}$, $0 < \alpha, \beta \leq 1$ where $d$ is the Euclidean distance between $u$ and $v$ and $L = c\sqrt{2}$. The edge number increases with $\alpha$ and the edge length heterogeneity increases with $\beta$.

Exponential Dots are randomly placed and are connected with a probability $P(u, v) = \alpha e^{-d/(L-d)}$.

Locality [ZCD97] This model is due to Zegura. Dots are randomly placed and are connected with a probability

$$P(u, v) = \begin{cases} 
\alpha & \text{if } d < L \times r \\
\beta & \text{if } d \geq L \times r.
\end{cases}$$
Node placement

Uniform
Node placement

Heavy Tailed
Faloutsos brothers [FFF99] have analyzed the topology at the AS level and have established power-laws describing this topology. The rank $r_v$ of a note $v$ is its index in the order of decreasing degree.
Faloutsos brothers [FFF99] have analyzed the topology at the AS level and have established power-laws describing this topology. The rank $r_v$ of a note $v$ is its index in the order of decreasing degree.
Faloutsos brothers [FFF99] have analyzed the topology at the AS level and have established power-laws describing this topology. The rank $r_v$ of a note $v$ is its index in the order of decreasing degree.

**Power Lax (rank exponent).**

Given a graph, the degree $d_v$ of a node $v$ is proportional to the rank of the node $r_v$ to the power of a constant $R$.

$$d_v \propto r_v^R$$

<table>
<thead>
<tr>
<th></th>
<th>Nov. 97</th>
<th>Apr. 98</th>
<th>Dec. 98</th>
<th>Router 95</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>0.81</td>
<td>0.82</td>
<td>0.74</td>
<td>0.48</td>
</tr>
</tbody>
</table>
What about Power Laws?

Incremental growth and affinity lead to Power Laws [BA99]). Nodes are incrementally added. The probability that \(v\) is connected to \(u\) depends on \(d_u\):

\[
P(u, v) = \frac{d_u}{\sum_k d_k}
\]
What about Power Laws?

Incremental growth and affinity lead to Power Laws [BA99]). Nodes are incrementally added. The probability that \( v \) is connected to \( u \) depends on \( d_u \):

\[
P(u, v) = \frac{d_u}{\sum_k d_k}
\]
What about hierarchy?

Top-Down

**N-level [ZCD97]** Starting from a connected graph, at each step, a node is replaced by another connected graph (Tiers, GT-ITM).

**Transit-stub [ZCD97]** 2-levels of hierarchy and some additional edges (GT-ITM, BRITE).
What about hierarchy?

**Top-Down**

**N-level [ZCD97]** Starting from a connected graph, at each step, a node is replaced by another connected graph (Tiers, GT-ITM).

**Transit-stub [ZCD97]** 2-levels of hierarchy and some additional edges (GT-ITM, BRITE).
Let us check two Power-Laws

Interdomain November 1997
Let us check two Power-Laws

GT-ITM flat?
Let us check two Power-Laws

Waxman (BRITE)
Let us check two Power-Laws

Transit-Stub (GT-ITM)
Let us check two Power-Laws

Barábasi Albert (BRITE)
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Distortion
- Resilience
- Eccentricity distribution
- Eigenvalues distribution
- Set cover size, ...

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Distortion
- Resilience
- Eccentricity distribution
- Eigenvalues distribution
- Set cover size, ...

Observation [TGJ+02]:
- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Distortion
- Resilience
- Excentricity distribution
- Eigenvalues distribution
- Set cover size, ...

Observation [TGJ⁺02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seems to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Resilience
- Distortion
- Excentricity distribution
- Eigenvalues distribution
- Set cover size, ...

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Resilience
- Distortion
- Excentricity distribution
- Eigenvalues distribution
- Set cover size, ...

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Resilience
- Distortion
- Excentricity distribution
- Eigenvalues distribution
- Set cover size, ...

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Resilience
- Distortion
- Eccentricity distribution
- Eigenvalues distribution
- Set cover size, ...

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Resilience
- Distortion
- Excentricity distribution
- Eigenvalues distribution
- Set cover size, ...

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Distortion
- Resilience
- Eccentricity distribution
- Eigenvalues distribution
- Set cover size, ... 

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Distortion
- Resilience
- Excentricity distribution
- Eigenvalues distribution
- Set cover size, . . .

Observation [TGGJ⁺02]:

- AS-level and router-level have similar characteristics
  - Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Distortion
- Resilience
- Excentricity distribution
- Eigenvalues distribution
- Set cover size, . . .

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
Are these measurements meaningful?

Some of the power laws observed by the Faloutsos brothers are correlated. What kind of measurements can be used?

- Expansion
- Distortion
- Resilience
- Excentricity distribution
- Eigenvalues distribution
- Set cover size, . . .

Observation [TGJ+02]:

- AS-level and router-level have similar characteristics
- Degree-based generators are significantly better at representing large scale properties of the Internet than structural ones.
- Hierarchy seem to arise from degree-based generators.
I’m lost... what should I do then?

- For a **10 000** nodes platform, degree-based generators seem to give good results.

- For a **100** node platform, power laws do not make sense and structural generators may be more appropriate.

  We need some additional informations (e.g. routing, bandwidth, latency, sharing capacity, ...).

**Idea 1:** use a structural generator (e.g. Tiers) with a simple edge classification scheme (LAN/MAN/WAN) and annotate with some real measurements.

**Idea 2:** use a network mapper.
I’m lost... what should I do then?

▶ For a 10 000 nodes platform, degree-based generators seem to give good results.
▶ For a 100 node platform, power laws do not make sense and structural generators may be more appropriate.

We need some additional informations (e.g. routing, bandwidth, latency, sharing capacity, ...).

Idea 1: use a structural generator (e.g. Tiers) with a simple edge classification scheme (LAN/MAN/WAN) and annotate with some real measurements.

Idea 2: use a network mapper.
I’m lost... what should I do then?

- For a 10 000 nodes platform, degree-based generators seem to give good results.
- For a 100 node platform, power laws do not make sense and structural generators may be more appropriate.

We need some additional informations (e.g. routing, bandwidth, latency, sharing capacity, ...).

Idea 1: use a structural generator (e.g. Tiers) with a simple edge classification scheme (LAN/MAN/WAN) and annotate with some real measurements.

Idea 2: use a network mapper.
I’m lost... what should I do then?

- For a **10 000** nodes platform, degree-based generators seem to give good results.
- For a **100** node platform, power laws do not make sense and structural generators may be more appropriate.

  We need some additional informations (e.g. routing, bandwidth, latency, sharing capacity, ...).

**Idea 1:** use a structural generator (e.g. Tiers) with a simple edge classification scheme (LAN/MAN/WAN) and annotate with some real measurements.

**Idea 2:** use a network mapper.
I’m lost... what should I do then?

- For a 10 000 nodes platform, degree-based generators seem to give good results.
- For a 100 node platform, power laws do not make sense and structural generators may be more appropriate.

We need some additional informations (e.g. routing, bandwidth, latency, sharing capacity, ...).

**Idea 1:** use a structural generator (e.g. Tiers) with a simple edge classification scheme (LAN/MAN/WAN) and annotate with some real measurements.

**Idea 2:** use a network mapper.
Which level of the network?

- Level 1: unreachable, too close to hardware
- Level 2: still too close to hardware, proprietary tools, SNMP not always usable, ...
- Level 3: the application level, usable in a grid context, still allow to guess layer 2 informations

Efficient Network View

- Developed at UCSD by Gary Shao
- Designed to improve master/slave scheduling. → master point of view (tree)
- Only relies on user level tools (python, ssh, traceroute, ...)

Network Mapping for Platform Simulation
Which level of the network?

- **Level 1**: unreachable, too close to hardware
- **Level 2**: still too close to hardware, proprietary tools, SNMP not always usable,…
- **Level 3**: the application level, usable in a grid context, still allow to guess layer 2 informations

**Efficient Network View**

- Developed at UCSD by Gary Shao
- Designed to improve master/slave scheduling. ↳ master point of view (tree)
- Only relies on user level tools (python, ssh, traceroute, …)
Network Mapping for Platform Simulation

Which level of the network?

- Level 1: unreachable, too close to hardware
- Level 2: still too close to hardware, proprietary tools, SNMP not always usable,
- Level 3: the application level, usable in a grid context, still allow to guess layer 2 informations

Efficient Network View

- Developed at UCSD by Gary Shao
- Designed to improve master/slave scheduling. master point of view (tree)
- Only relies on user level tools (python, ssh, traceroute, ... )
Network Mapping for Platform Simulation

Which level of the network?

- Level 1: unreachable, too close to hardware
- Level 2: still too close to hardware, proprietary tools, SNMP not always usable,…
- Level 3: the application level, usable in a grid context, still allow to guess layer 2 informations

Efficient Network View

- Developed at UCSD by Gary Shao
- Designed to improve master/slave scheduling. master point of view (tree)
- Only relies on user level tools (python, ssh, traceroute, …)
Network Mapping for Platform Simulation

Which level of the network?

- Level 1: unreachable, too close to hardware
- Level 2: still too close to hardware, proprietary tools, SNMP not always usable, ...
- Level 3: the application level, usable in a grid context, still allow to guess layer 2 informations

Efficient Network View

- Developed at UCSD by Gary Shao
- Designed to improve master/slave scheduling.  \( \leadsto \) master point of view (tree)
- Only relies on user level tools (python, ssh, traceroute, ...)
Network Mapping for Platform Simulation

Which level of the network?

- Level 1: unreachable, too close to hardware
- Level 2: still too close to hardware, proprietary tools, SNMP not always usable, ...
- Level 3: the application level, usable in a grid context, still allow to guess layer 2 informations

Efficient Network View

- Developed at UCSD by Gary Shao
- Designed to improve master/slave scheduling. master point of view (tree)
- Only relies on user level tools (pyhton, ssh, traceroute, ... )
Network Mapping for Platform Simulation

Which level of the network?

- Level 1: unreachable, too close to hardware
- Level 2: still too close to hardware, proprietary tools, SNMP not always usable, ...
- Level 3: the application level, usable in a grid context, still allow to guess layer 2 informations

Efficient Network View

- Developed at UCSD by Gary Shao
- Designed to improve master/slave scheduling. master point of view (tree)
- Only relies on user level tools (python, ssh, traceroute, ... )
Network Mapping for Platform Simulation

Which level of the network?

- Level 1: unreachable, too close to hardware
- Level 2: still too close to hardware, proprietary tools, SNMP not always usable,…
- Level 3: the application level, usable in a grid context, still allow to guess layer 2 informations

Efficient Network View

- Developed at UCSD by Gary Shao
- Designed to improve master/slave scheduling. \( \sim \) master point of view (tree)
- Only relies on user level tools (python, ssh, traceroute, …)
ENV: methodology

the-doors

any host
ENV : methodology

the-doors

any host

the-doors

cluster
ENV : methodology

the-doors

any host

the-doors

cluster

cluster
ENV : methodology

The-doors

Any host

Cluster

Cluster

Cluster
Running ENV on the ENS network
Running ENV on the ENS network
Running ENV on the ENS network

```xml
<NONE type="ENV_Switched">
  <LABEL name="sci0" />
  <PROPERTY name="ENV_base_BW" value="32.65" units="Mbps" />
  <PROPERTY name="ENV_base_local_BW" value="32.29" units="Mbps" />
  <MACHINE name="sci1.popc.private" />
  <MACHINE name="sci2.popc.private" />
  <MACHINE name="sci3.popc.private" />
  <MACHINE name="sci4.popc.private" />
  <MACHINE name="sci5.popc.private" />
  <MACHINE name="sci6.popc.private" />
</NETWORK>
```
ENV limitations

- Firewall ~ fusion
- Precision, time limitation, experimental thresholds
- Master/slave oriented
  - Asymmetric routing
  - Inter-cluster links
ENV limitations

- Firewall → fusion
- Precision, time limitation, experimental thresholds
- Master/slave oriented
  - Asymmetric routing
  - Inter-cluster links
ENV limitations

- Firewall → fusion
- Precision, time limitation, experimental thresholds
- Master/slave oriented
  - Asymmetric routing
  - Inter-cluster links
Gathering traces

Network Weather Service

- Developed at UCSB
- Provides accurate data on a meta-computing platform
- Forecasting on links and processors performances
- Almost automatized deployment from the ENV output [LQ04].
Gathering traces

Network Weather Service

▶ Developed at UCSB
▶ Provides accurate data on a meta-computing platform
▶ Forecasting on links and processors performances
▶ Almost automatized deployment from the ENV output [LQ04].
Gathering traces

Network Weather Service

- Developed at UCSB
- Provides accurate data on a meta-computing platform
- Forecasting on links and processors performances
- Almost automatized deployment from the ENV output [LQ04].
Gathering traces

Network Weather Service

- Developed at UCSB
- Provides accurate data on a meta-computing platform
- Forecasting on links and processors performances
- Almost automatized deployment from the ENV output [LQ04].
Modeling hubs, switches and routers

Master

Link A

Hub

Link B Link C

Slave A Slave B
Modeling hubs, switches and routers
Modeling hubs, switches and routers
Modeling hubs, switches and routers
Modeling hubs, switches and routers
Modeling hubs, switches and routers

Bandwidth of Link A cannot be measured by NWS
Modeling hubs, switches and routers

Diagram:
- Master
- Router
- Slave A
- Slave B
- Link A
- Link B
- Link C
- Link B'
- Link C'

Equation:
$A \gg B + C$
Modeling hubs, switches and routers 
Outline

Scheduling on an heterogeneous environment: validation

Simulation: a brief state of the art

SIMGRID, a modular trace-based simulator

Obtaining a realistic platform model
  Random topology
  Real topology
  Getting traces

Conclusion
# A few remarks

**SimGrid cannot:**

- help you to figure out what is going to be the duration of a real application
- model accurately the behavior of a computing platform
- help you to fix some experimental thresholds
- help you to debug an already existing code

**but SimGrid rather can:**

- help you to compare two algorithms
- help you to study the robustness of your algorithm in a noisy environment
- be used to design adaptive thresholds strategies and test them against a wide variety of environments
- help you to test and debug your algorithms before the real implementation
## A few remarks

<table>
<thead>
<tr>
<th><strong>SimGrid</strong> cannot:</th>
<th>but <strong>SimGrid</strong> rather can</th>
</tr>
</thead>
<tbody>
<tr>
<td>help you to figure out what is going to be the duration of a real application</td>
<td>help you to compare two algorithms</td>
</tr>
<tr>
<td>model accurately the behavior of a computing platform</td>
<td>help you to study the robustness of your algorithm in a noisy environment</td>
</tr>
<tr>
<td>help you to fix some experimental thresholds</td>
<td>be used to design adaptive thresholds strategies and test them against a wide variety of environments</td>
</tr>
<tr>
<td>help you to debug an already existing code</td>
<td>help you to test and debug your algorithms before the real implementation</td>
</tr>
</tbody>
</table>
**A few remarks**

<table>
<thead>
<tr>
<th><strong>SimGrid cannot:</strong></th>
<th><strong>but SimGrid rather can</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>help you to figure out what is going to be the duration of a real application</td>
<td>help you to compare two algorithms</td>
</tr>
<tr>
<td>model accurately the behavior of a computing platform</td>
<td>help you to study the robustness of your algorithm in a noisy environment</td>
</tr>
<tr>
<td>help you to fix some experimental thresholds</td>
<td>be used to design adaptive thresholds strategies and test them against a wide variety of environments</td>
</tr>
<tr>
<td>help you to debug an already existing code</td>
<td>help you to test and debug your algorithms before the real implementation</td>
</tr>
</tbody>
</table>
### A few remarks

<table>
<thead>
<tr>
<th><strong>SimGrid cannot:</strong></th>
<th><strong>but SimGrid rather can</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>help you to figure out what is going to be the duration of a real application</td>
<td>help you to compare two algorithms</td>
</tr>
<tr>
<td>model accurately the behavior of a computing platform</td>
<td>help you to study the robustness of your algorithm in a noisy environment</td>
</tr>
<tr>
<td>help you to fix some experimental thresholds</td>
<td>be used to design adaptive thresholds strategies and test them against a wide variety of environments</td>
</tr>
<tr>
<td>help you to debug an already existing code</td>
<td>help you to test and debug your algorithms before the real implementation</td>
</tr>
</tbody>
</table>
## A few remarks

<table>
<thead>
<tr>
<th><strong>SimGrid cannot:</strong></th>
<th><strong>but SimGrid rather can</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>help you to figure out what is going to be the duration of a real application</td>
<td>help you to compare two algorithms</td>
</tr>
<tr>
<td>model accurately the behavior of a computing platform</td>
<td>help you to study the robustness of your algorithm in a noisy environment</td>
</tr>
<tr>
<td>help you to fix some experimental thresholds</td>
<td>be used to design adaptive thresholds strategies and test them against a wide variety of environments</td>
</tr>
<tr>
<td>help you to debug an already existing code</td>
<td>help you to test and debug your algorithms before the real implementation</td>
</tr>
</tbody>
</table>
## A few remarks

**SimGrid** *cannot*:  
help you to figure out what is going to be the duration of a real application  
model accurately the behavior of a computing platform  
help you to fix some experimental thresholds  
help you to debug an already existing code

But **SimGrid** *rather* *can*:  
help you to compare two algorithms  
help you to study the robustness of your algorithm in a noisy environment  
be used to design adaptive thresholds strategies and test them against a wide variety of environments  
help you to test and debug your algorithms before the real implementation
## A few remarks

<table>
<thead>
<tr>
<th><strong>SimGrid cannot:</strong></th>
<th><strong>but SimGrid rather can</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>help you to figure out what is going to be the duration of a real application</td>
<td>help you to compare two algorithms</td>
</tr>
<tr>
<td>model accurately the behavior of a computing platform</td>
<td>help you to study the robustness of your algorithm in a noisy environment</td>
</tr>
<tr>
<td>help you to fix some experimental thresholds</td>
<td>be used to design adaptive thresholds strategies and test them against a wide variety of environments</td>
</tr>
<tr>
<td>help you to debug an already existing code</td>
<td>help you to test and debug your algorithms before the real implementation</td>
</tr>
</tbody>
</table>
## A few remarks

<table>
<thead>
<tr>
<th><strong>SimGrid cannot:</strong></th>
<th><strong>but SimGrid rather can</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>help you to figure out what is going to be the duration of a real application</td>
<td>help you to compare two algorithms</td>
</tr>
<tr>
<td>model accurately the behavior of a computing platform</td>
<td>help you to study the robustness of your algorithm in a noisy environment</td>
</tr>
<tr>
<td>help you to fix some experimental thresholds</td>
<td>be used to design adaptive thresholds strategies and test them against a wide variety of environments</td>
</tr>
<tr>
<td>help you to debug an already existing code</td>
<td>help you to test and debug your algorithms before the real implementation</td>
</tr>
</tbody>
</table>
Always keep in mind

- Modeling is the art of tradeoff: trying to model everything is hopeless and it may be worse than a plain modeling.
- Platform \( \neq \) software deployment: the physical topology and the logical organization of the application are two distinct concepts.
- If you are working with DAGs and perfectly centralized scheduling (i.e. with Gantt Charts) then you should use SG.
- If many scheduling actions may occur independently, then use MSG. If you fail to express something with MSG, just wonder what you would do if you had to implement it for real.
- SIMGRID is still under development… 😊

http://gcl.ucsd.edu/simgrid/
Always keep in mind

- Modeling is the art of tradeoff: trying to model everything is hopeless and it may be worse than a plain modeling.
- Platform $\neq$ software deployment: the physical topology and the logical organization of the application are two distinct concepts.
- If you are working with DAGs and perfectly centralized scheduling (i.e. with Gantt Charts) then you should use SG.
- If many scheduling actions may occur independently, then use MSG. If you fail to express something with MSG, just wonder what you would do if you had to implement it for real.
- SIMGRID is still under development... 😊

http://gcl.ucsd.edu/simgrid/
Always keep in mind

- Modeling is the art of tradeoff: trying to model everything is hopeless and it may be worse than a plain modeling.
- Platform ≠ software deployment: the physical topology and the logical organization of the application are two distinct concepts.
- If you are working with DAGs and perfectly centralized scheduling (i.e. with Gantt Charts) then you should use SG.
- If many scheduling actions may occur independently, then use MSG. If you fail to express something with MSG, just wonder what you would do if you had to implement it for real.
- SIMGRID is still under development... 😊
  http://gcl.ucsd.edu/simgrid/
Always keep in mind

- Modeling is the art of tradeoff: trying to model everything is hopeless and it may be worse than a plain modeling.
- Platform ≠ software deployment: the physical topology and the logical organization of the application are two distinct concepts.
- If you are working with DAGs and perfectly centralized scheduling (i.e. with Gantt Charts) then you should use SG.
- If many scheduling actions may occur independently, then use MSG. If you fail to express something with MSG, just wonder what you would do if you had to implement it for real.
- SimGrid is still under development... 😊
  
  http://gcl.ucsd.edu/simgrid/
Always keep in mind

► Modeling is the art of tradeoff: trying to model everything is hopeless and it may be worse than a plain modeling.

► Platform $\neq$ software deployment: the physical topology and the logical organization of the application are two distinct concepts.

► If you are working with DAGs and perfectly centralized scheduling (i.e. with Gantt Charts) then you should use SG.

► If many scheduling actions may occur independently, then use MSG. If you fail to express something with MSG, just wonder what you would do if you had to implement it for real.

► SimGRID is still under development... 😊

http://gcl.ucsd.edu/simgrid/
A.L. Barabási and R. Albert. 
Emergence of scaling in random networks. 
Available at [http://www.nd.edu/~networks/Papers/science.pdf](http://www.nd.edu/~networks/Papers/science.pdf).

Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala. 
Improving simulation for network research. 
Available at [http://citeseer.nj.nec.com/bajaj99improving.html](http://citeseer.nj.nec.com/bajaj99improving.html).
D. Bertsekas and R. Gallager.  
*Data Networks.*  

Henri Casanova and Loris Marchal.  
A network model for simulation of grid application.  

Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos.  
On power-law relationships of the internet topology.  

Thilo Kielmann, Henri E. Bal, Jason Maassen, Rob van Nieuwpoort, Lionel Eyraud, Rutger Hofman, and Kees Verstoep.  
Programming environments for high-performance grid computing: the albatross project.  
F. Kelly.
Charging and rate control for elastic traffic, 1997.
Available at http://citeseer.nj.nec.com/kelly97charging.html.

Jason Liu and David M. Nicol.

A. Legrand and M. Quinson.
Automatic deployment of the Network Weather Service using the Effective Network View.

The network simulator - ns-2.
http://www.isi.edu/nsnam/ns.

Objective modular network testbed.
The microgrid: a scientific tool for modeling computational grids.
In *Supercomputing*, 2000.

Network topology generators: Degree-based vs structural.
Available at http://citeseer.nj.nec.com/tangmunarunkit02network.html.

Bernard M. Waxman.
Routing of Multipoint Connections.

Ellen W. Zegura, Kenneth L. Calvert, and Michael J. Donahoo.
A quantitative comparison of graph-based models for Internet topology.


Available at [http://citeseer.nj.nec.com/zegura97quantitative.html](http://citeseer.nj.nec.com/zegura97quantitative.html).