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

Loris Marchal

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

Colorado State University - April 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

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

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

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

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

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

Conclusion

Scheduling on an heterogeneous environment

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:

$\rightsquigarrow$ *low complexity heuristics*

Questions:

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

Scheduling on an heterogeneous environment

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:

$\rightsquigarrow$ *low complexity heuristics*

Questions:

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

Scheduling on an heterogeneous environment

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:

$\leadsto$ *low complexity heuristics*

Questions:

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

Scheduling on an heterogeneous environment

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:

$\rightsquigarrow$ *low complexity heuristics*

Questions:

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

Scheduling on an heterogeneous environment

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:

$\leadsto$ *low complexity heuristics*

Questions:

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

Scheduling on an heterogeneous environment

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:

$\rightsquigarrow$ *low complexity heuristics*

Questions:

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

Scheduling on an heterogeneous environment

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:

$\leadsto$ *low complexity heuristics*

Questions:

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

Real experiments . . .

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 . . .

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 . . .

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*.

. . . 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

  forms.
- 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

  forms.

- 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

  forms.

- 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 ?

. . . 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
  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 ?

. . . 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 ?

. . . 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 ?

. . . 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 ?

. . . 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

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, DaSSF, OMNeT++.

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, DaSSF, OMNeT++.

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.

$\rightsquigarrow$ requires a precise simulation of the movement of packets along the network links: NS, DaSSF, OMNeT++.

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.

$\rightsquigarrow$ requires a precise simulation of the movement of packets along the network links: NS, DaSSF, OMNeT++.

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, DaSSF, OMNeT++.

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, DaSSF, OMNeT++.

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, DaSSF, OMNeT++.

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, DaSSF, OMNeT++.

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, DaSSF, OMNeT++.

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 (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
- No external load for the network.

PANDA (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 (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
- No external load for the network.

PANDA (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* $\rightsquigarrow$ too slow, too "precise", too difficult for simple tests or the design phase.

Platform Emulation

A few examples:

MicroGrid (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
- No external load for the network.

PANDA (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 (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
- No external load for the network.

PANDA (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* $\rightsquigarrow$ too slow, too "precise", too difficult for simple tests or the design phase.

Platform Emulation

A few examples:

MicroGrid (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
- No external load for the network.

PANDA (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* $\rightsquigarrow$ too slow, too "precise", too difficult for simple tests or the design phase.

Platform Emulation

A few examples:

MicroGrid (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
- No external load for the network.

PANDA (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* $\rightsquigarrow$ too slow, too "precise", too difficult for simple tests or the design phase.

Platform Emulation

A few examples:

MicroGrid (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
- No external load for the network.

PANDA (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 (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
- No external load for the network.

PANDA (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* $\rightsquigarrow$ too slow, too "precise", too difficult for simple tests or the design phase.

Platform Emulation

A few examples:

MicroGrid (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
- No external load for the network.

PANDA (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

Conclusion

History:

- Application Level Scheduling (AppLeS) : to a given application corresponds a given scheduler. $\rightsquigarrow$ 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 $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$.

SimGrid

History:

- Application Level Scheduling (AppLeS) : to a given application corresponds a given scheduler. $\rightsquigarrow$ 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 $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$.

SimGrid

History:

- Application Level Scheduling (AppLeS) : to a given application corresponds a given scheduler. $\rightsquigarrow$ 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 $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$.

SimGrid

History:

- Application Level Scheduling (AppLeS) : to a given application corresponds a given scheduler. $\rightsquigarrow$ 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 $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$.

SimGrid

History:

- Application Level Scheduling (AppLeS) : to a given application corresponds a given scheduler. $\rightsquigarrow$ 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 $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$.

SimGrid

History:

- Application Level Scheduling (AppLeS) : to a given application corresponds a given scheduler. $\rightsquigarrow$ 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 $\frac{\text{simulation time}}{\text{simulated time}} \approx 10^{-6}$.

SimGrid

History:

- Application Level Scheduling (AppLeS) : to a given application corresponds a given scheduler. $\rightsquigarrow$ 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.:

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

- On shared resources, all tasks 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 tasks 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 tasks 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 tasks 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 tasks 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 tasks 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

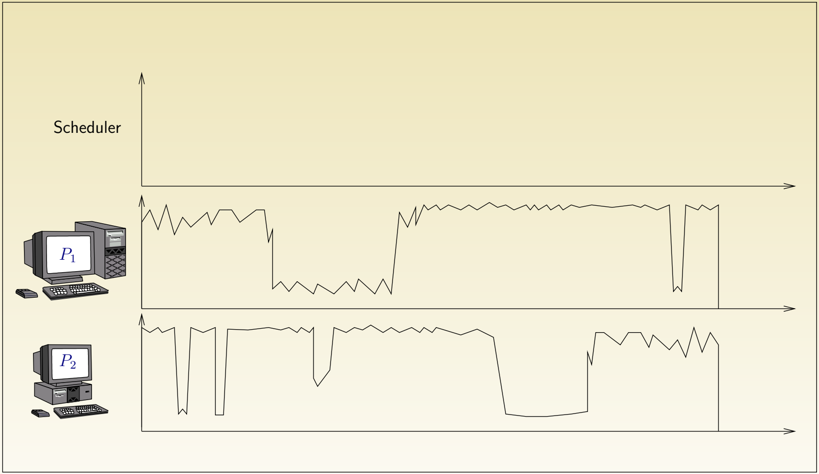
SG allows to create those objects and to schedule a task on a resource.

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

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

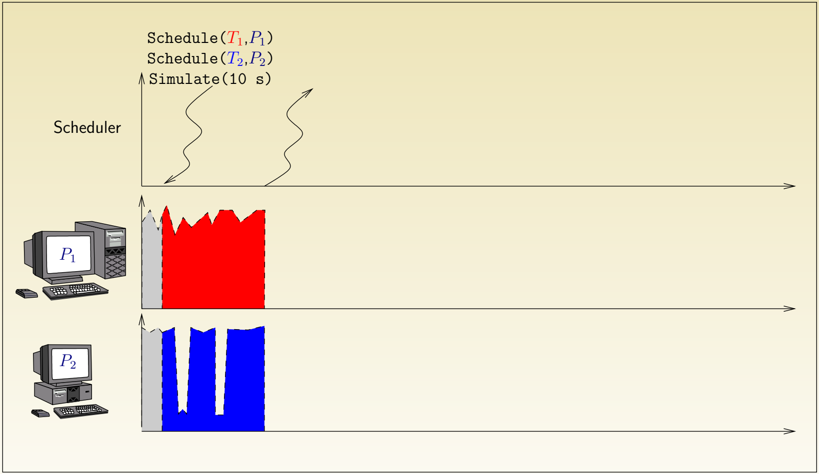
- On shared resources, all tasks get an amount of power proportional to their priority.

Using some traces

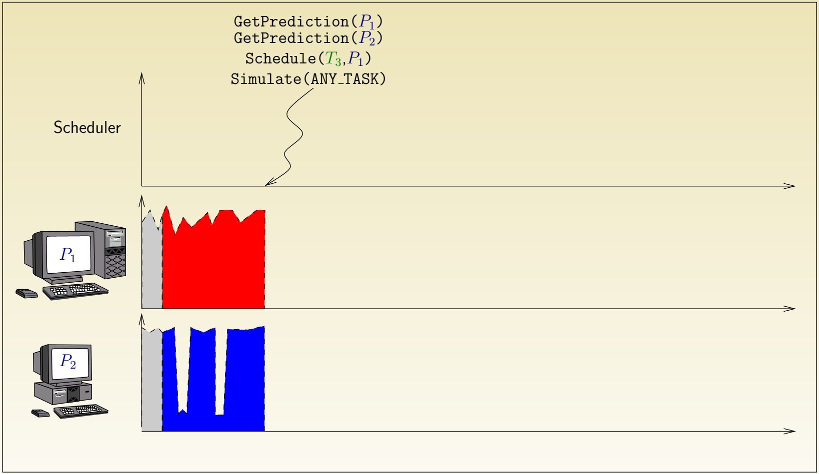


Using some traces

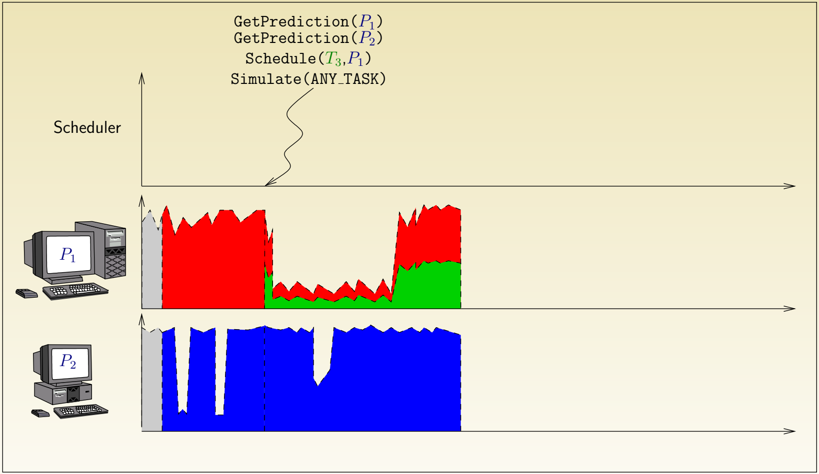
Using some traces



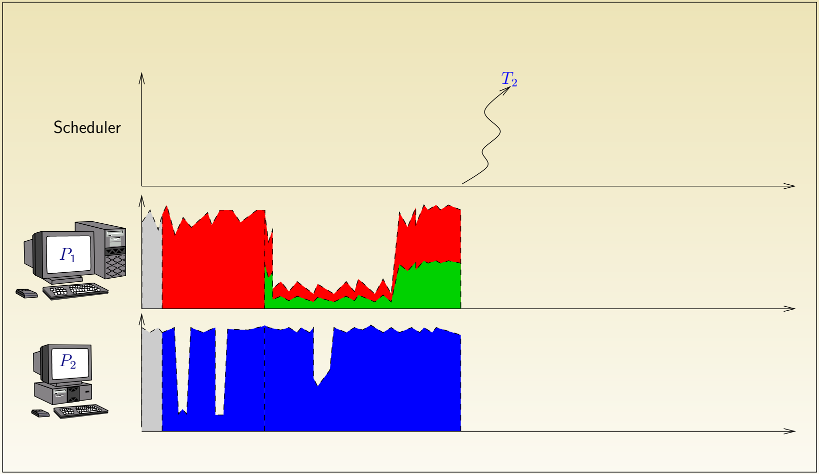
Using some traces



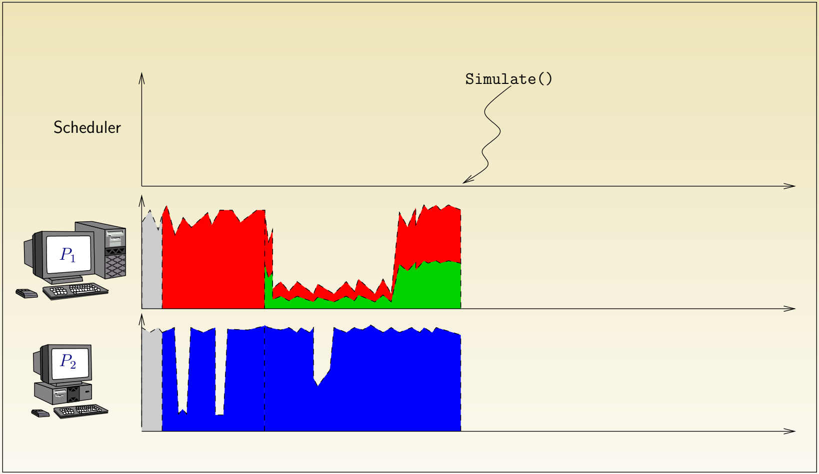
Using some traces



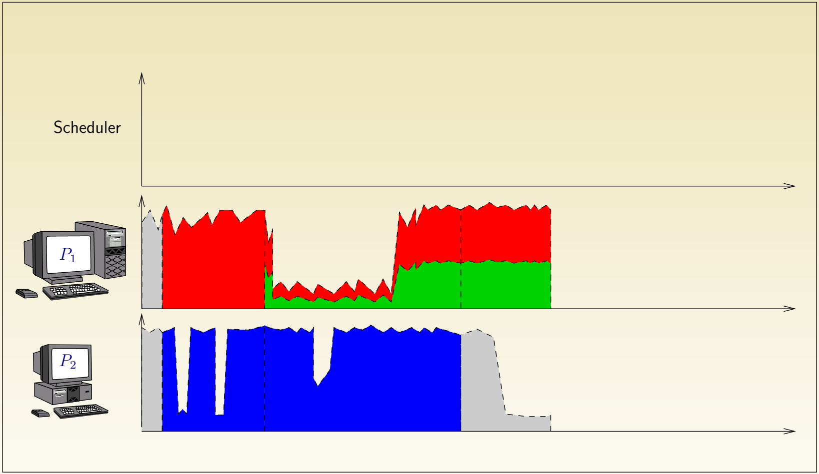
Using some traces



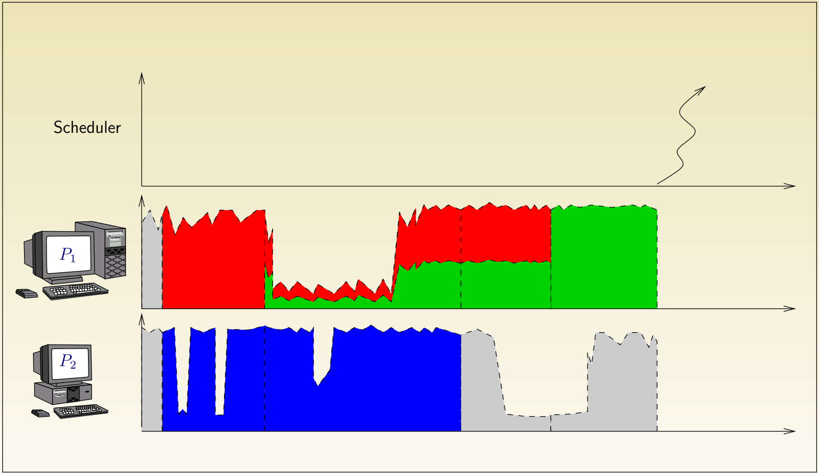
Using some traces



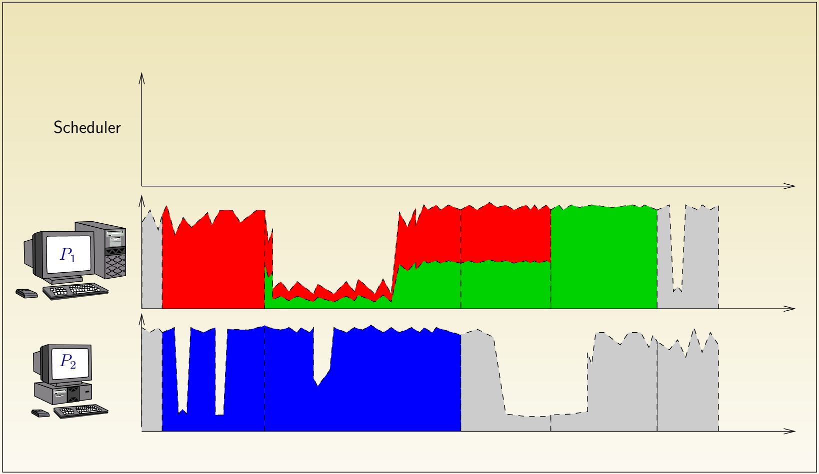
Using some traces



Using some traces



Using some traces

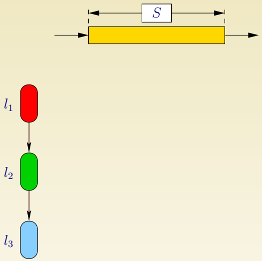


Store & Forward, WormHole, TCP

How to model a file transfer along a path?

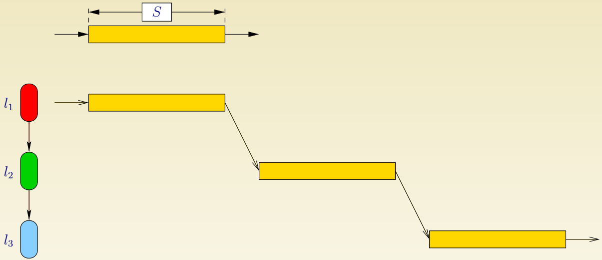
Store & Forward, WormHole, TCP

How to model a file transfer along a path?



Store & Forward, WormHole, TCP

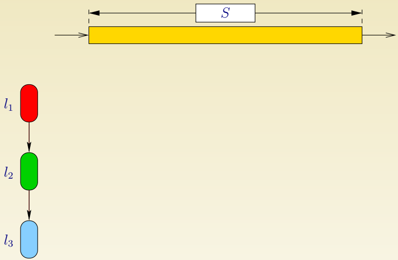
How to model a file transfer along a path?



Store & Forward : bad model for contention

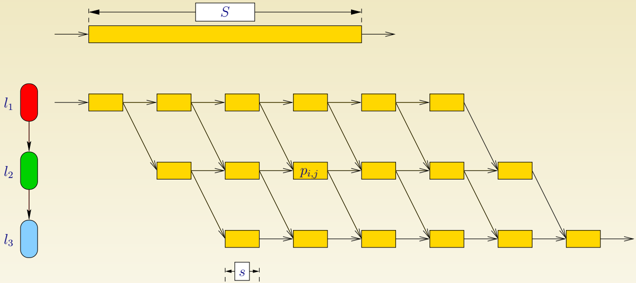
Store & Forward, WormHole, TCP

How to model a file transfer along a path?



Store & Forward, WormHole, TCP

How to model a file transfer along a path?



WormHole : computation intensive (packets), not that realistic

Store & Forward, WormHole, TCP

How to model a file transfer along a path?

$$\forall l \in \mathcal{L}, \quad \sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leqslant c_l,$$

Analytical model

Store & Forward, WormHole, TCP

How to model a file transfer along a path?

$$\forall l \in \mathcal{L}, \quad \sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leqslant c_l,$$

Max-Min Fairness maximize $\min_{r \in \mathcal{R}} \rho_r$.

Store & Forward, WormHole, TCP

How to model a file transfer along a path?

$$\forall l \in \mathcal{L}, \qquad \sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leqslant c_l,$$

Max-Min Fairness maximize $\min_{r \in \mathcal{R}} \rho_r$.

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

Store & Forward, WormHole, TCP

How to model a file transfer along a path?

$$\forall l \in \mathcal{L}, \quad \sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leqslant c_l,$$

Max-Min Fairness maximize $\min_{r \in \mathcal{R}} \rho_r$.

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

MCT minimization maximize $\sum_{r \in \mathcal{R}} \frac{1}{\rho_r}$.

Store & Forward, WormHole, TCP

How to model a file transfer along a path?

$$\forall l \in \mathcal{L}, \quad \sum_{r \in \mathcal{R} \text{ s.a. } l \in r} \rho_r \leqslant c_l,$$

Max-Min Fairness maximize $\min_{r \in \mathcal{R}} \rho_r$.

Proportional Fairness 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 Close to max-min. In MSG : max-min + bound by $1/RTT$

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;

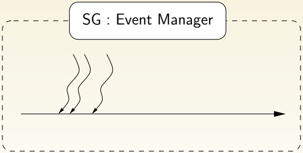
Task an amount of computing, a data size, and private data;

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

Channel mailbox number.

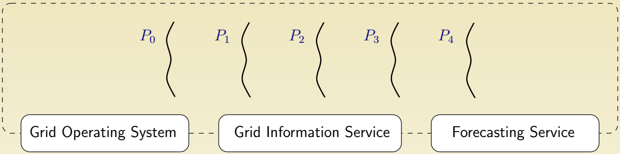
Non-centralized scheduling ?

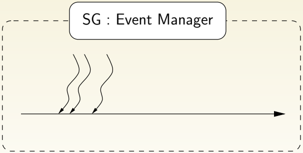
Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.



Non-centralized scheduling ?

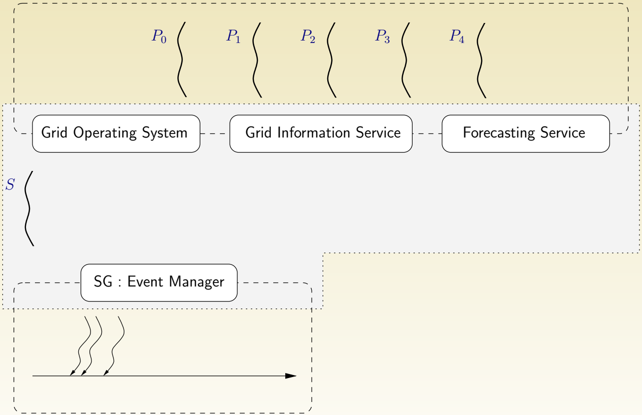
Centralized scheduling do not scale and SG is not well suited to study such scheduling policies.





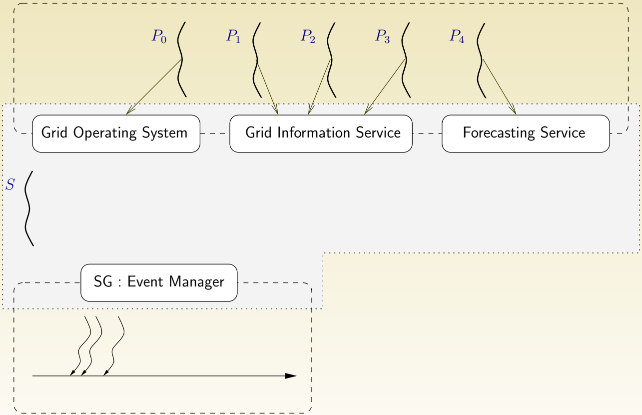
Non-centralized scheduling ?

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



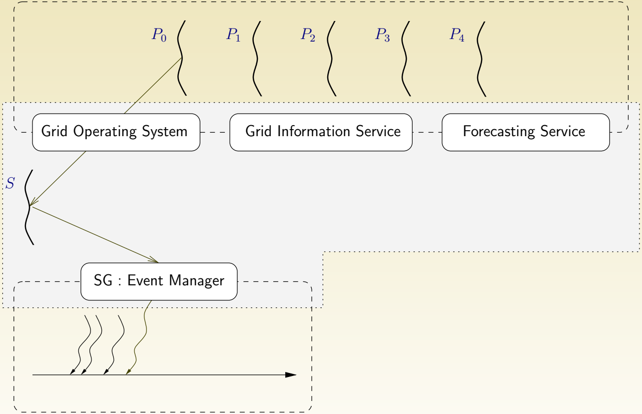
Non-centralized scheduling ?

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



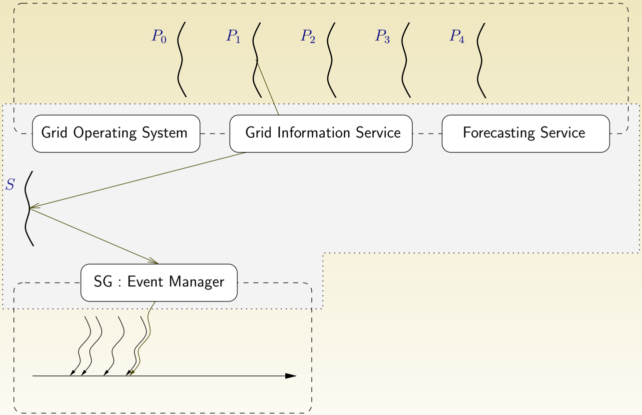
Non-centralized scheduling ?

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



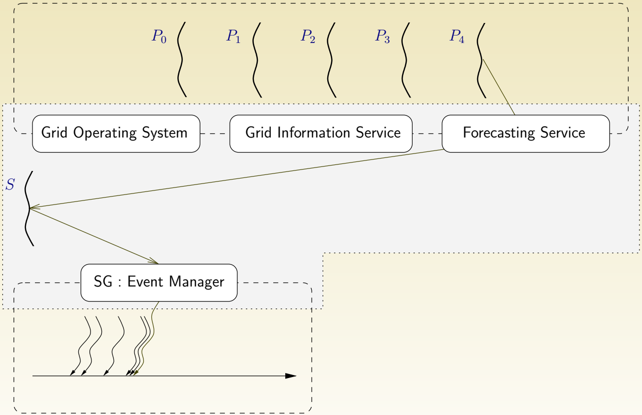
Non-centralized scheduling ?

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



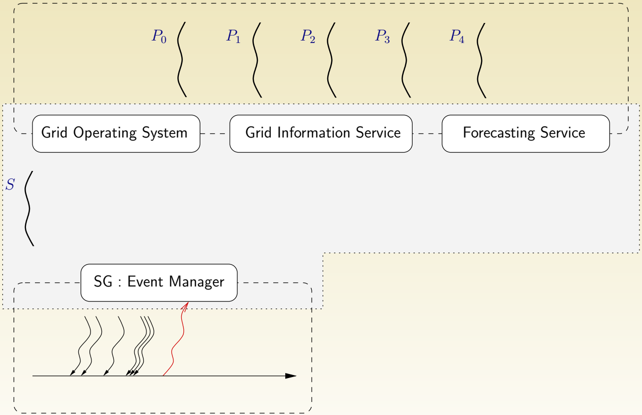
Non-centralized scheduling ?

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



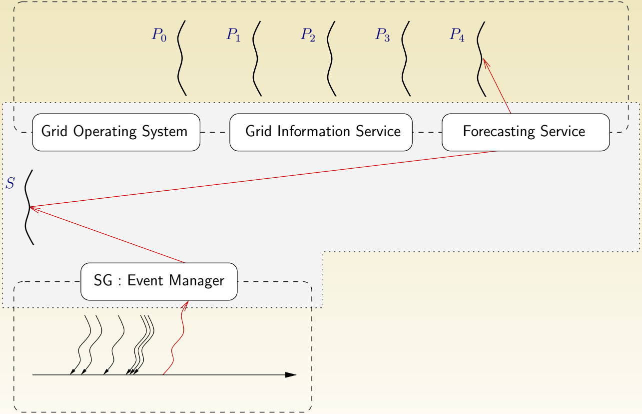
Non-centralized scheduling ?

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



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_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_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_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

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 α.

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

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

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

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

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 α.

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

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

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

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

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 α.

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

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

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

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

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 α.

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

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

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

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

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 α.

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

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

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

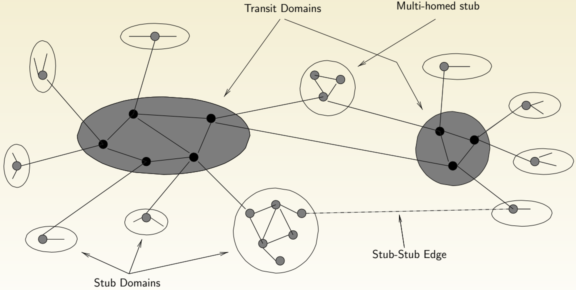
$$P(u,v) = \begin{cases} \alpha & \text{if } d < L \times r \\ \beta & \text{if } d \geqslant L \times r \end{cases}.$$

Hierarchical Model

Top-Down

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

Transit-stub 2-levels of hierarchy and some additional edges (GT-ITM, BRITE).

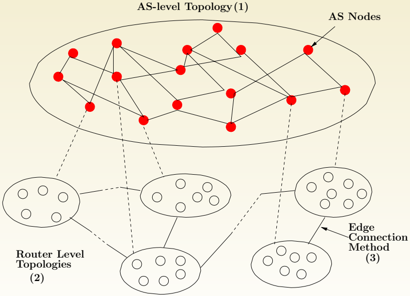


Hierarchical Model

Top-Down

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

Transit-stub 2-levels of hierarchy and some additional edges (GT-ITM, BRITE).



Network Mapping for Platform Simulation

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 (pyhton, ssh, traceroute, . . .)

Network Mapping for Platform Simulation

Efficient Network View

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

Network Mapping for Platform Simulation

Efficient Network View

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

Network Mapping for Platform Simulation

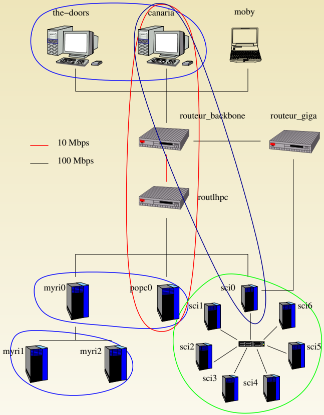
Efficient Network View

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

Gathering traces

Network Weather Service

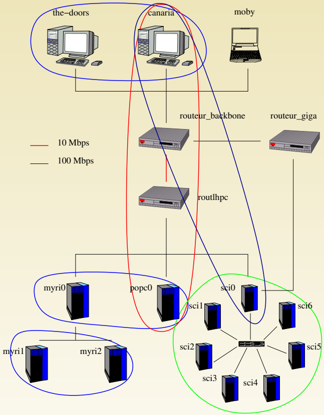
- Developed at UCSB
- Provides accurate data on a meta-computing platform
- Forecasting on links and processors performances
- Almost automatized deployment from the ENV output.



Gathering traces

Network Weather Service

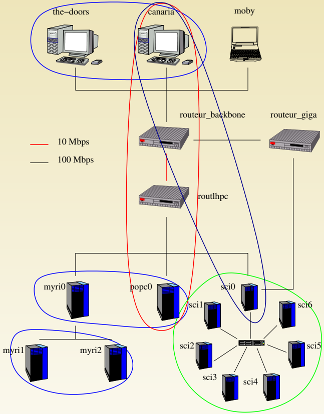
- Developed at UCSB
- Provides accurate data on a meta-computing platform
- Forecasting on links and processors performances
- Almost automatized deployment from the ENV output.



Gathering traces

Network Weather Service

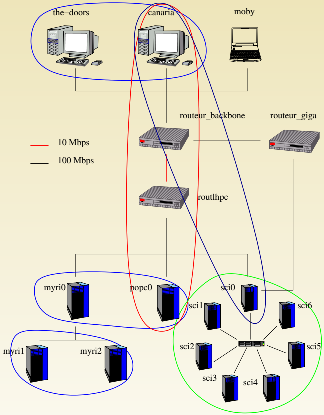
- Developed at UCSB
- Provides accurate data on a meta-computing platform
- Forecasting on links and processors performances
- Almost automatized deployment from the ENV output.



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.



Outline

A few remarks

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

A few remarks

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

A few remarks

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

A few remarks

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

A few remarks

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

A few remarks

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

A few remarks

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

A few remarks

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

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.
- 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.
- 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.
- 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.
- 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/

Bibliography

Henri Casanova.
Simgrid: A toolkit for the simulation of application scheduling.
In *Proceedings of the IEEE Symposium on Cluster Computing and the Grid (CCGrid'01)*. IEEE Computer Society, May 2001.
Further on Simgrid at http://gcl.ucsd.edu/simgrid.

Henri Casanova, Arnaud Legrand, and Loris Marchal.
Scheduling distributed applications: the simgrid simulation framework.
In *Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*. "IEEE Computer Society Press", may 2003.

A. Legrand and M. Quinson.
Automatic deployment of the Network Weather Service using the Effective Network View.
In *HPGC'2004; High Performance Grid Computing workshop*. IEEE Computer Society Press, 2004.