Simulación de sistemas distribuidos de gran escala

Félix García Carballeira
Grupo de Arquitectura de Computadores
Universidad Carlos III de Madrid
felix.garcia@uc3m.es

Objetivo

- Comprender las ventajas que aportan las técnicas de simulación cuando se quiere realizar investigación y experimentar sobre sistemas distribuidos de gran escala y las ventajas que aporta el entorno de simulación SimGrid

Contenido

- Motivación
- Entorno de simulación SimGrid
- Ejemplos de modelos de simulación

¿Por qué es importante la simulación?

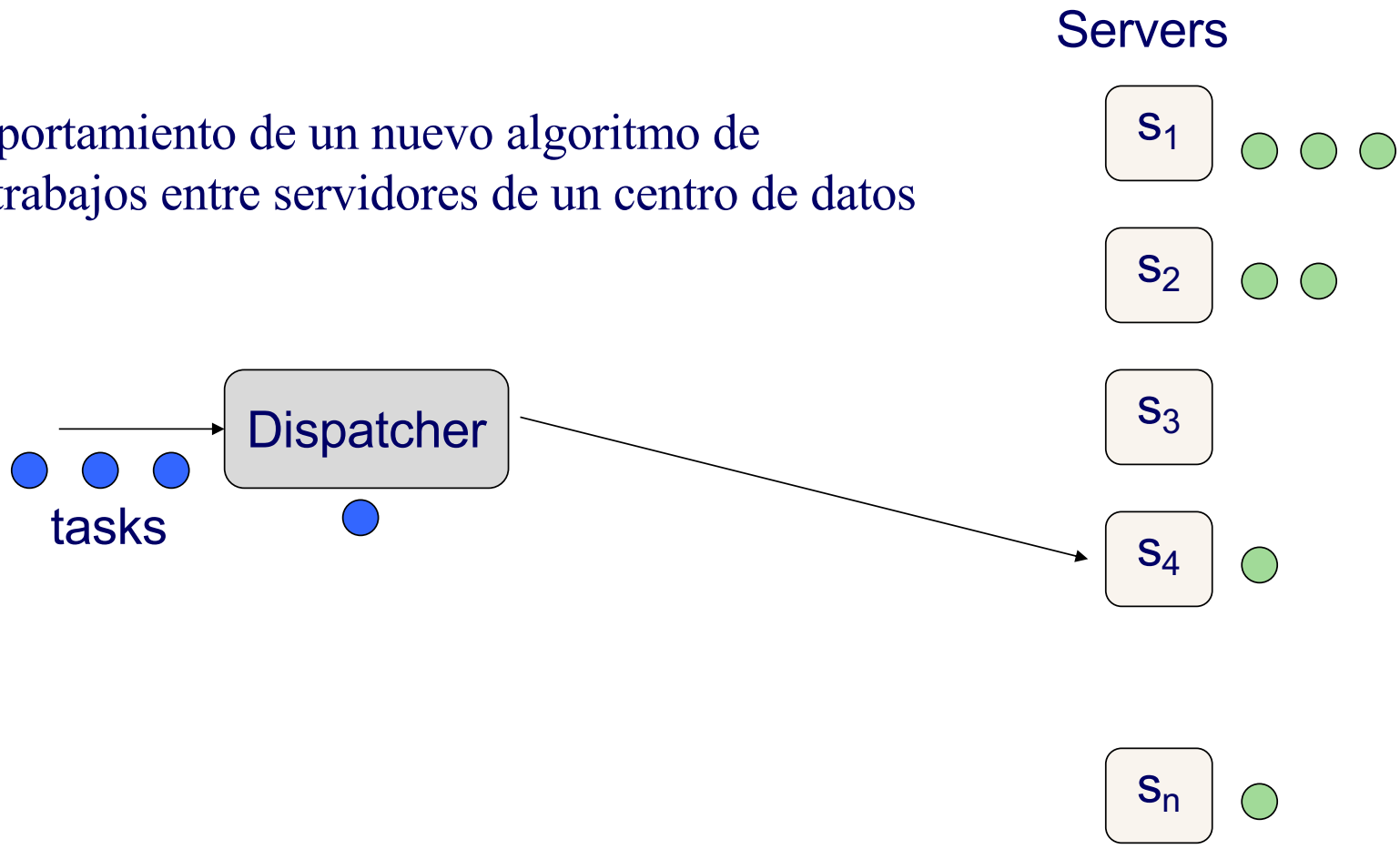
- Conocer el comportamiento de un sistema cuando no se dispone de él bajo una determinada carga de trabajo
- Ejemplos:
  - Conocer el comportamiento de un determinado algoritmo de planificación
  - Comparar el funcionamiento de dos algoritmos de distribución de carga
  - Conocer el comportamiento de un sistema de ficheros distribuido

Motivación

- Conocer el comportamiento de un nuevo algoritmo de distribución de trabajos entre servidores de un centro de datos

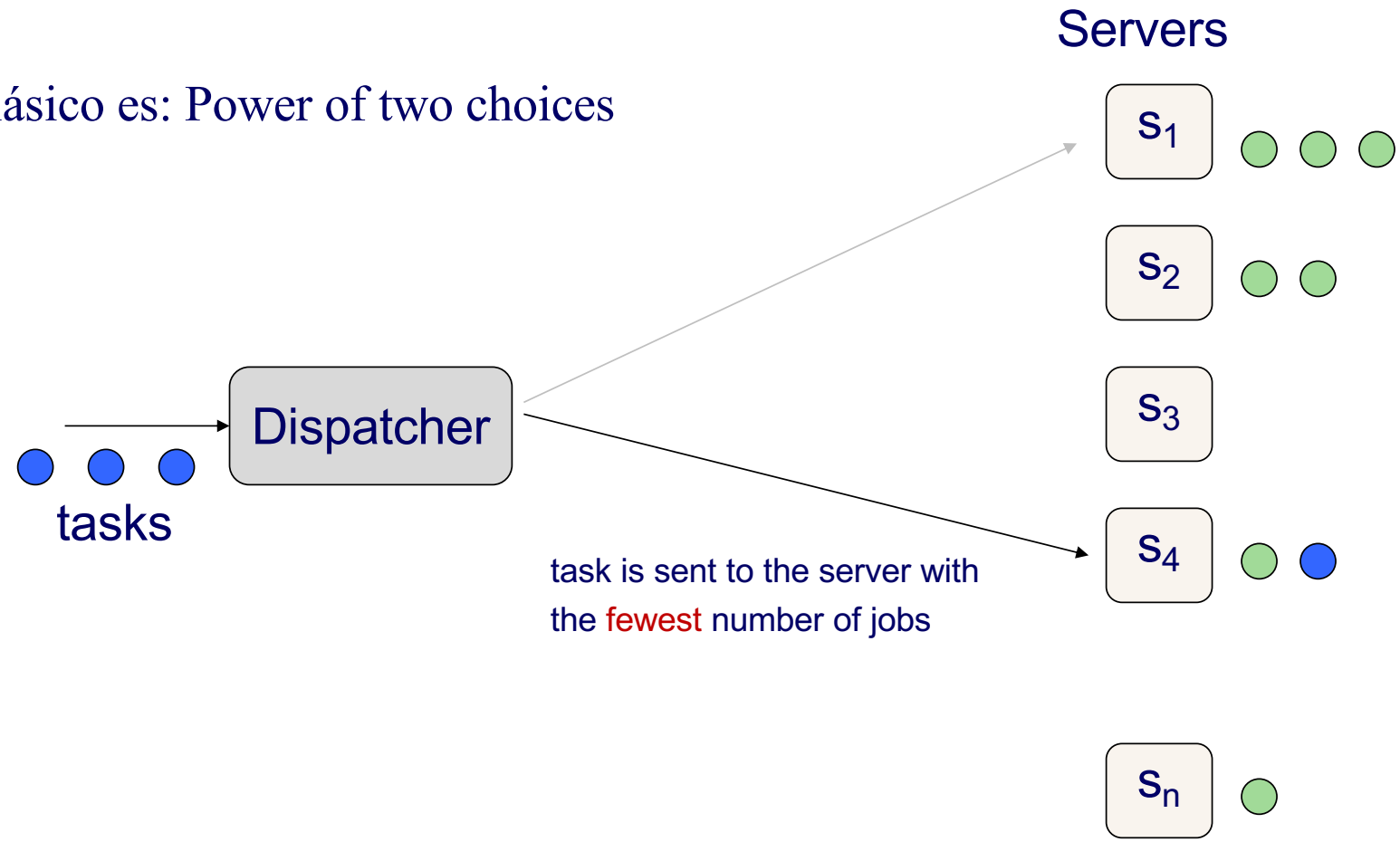
Motivación

- Conocer el comportamiento de un nuevo algoritmo de distribución de trabajos entre servidores de un centro de datos



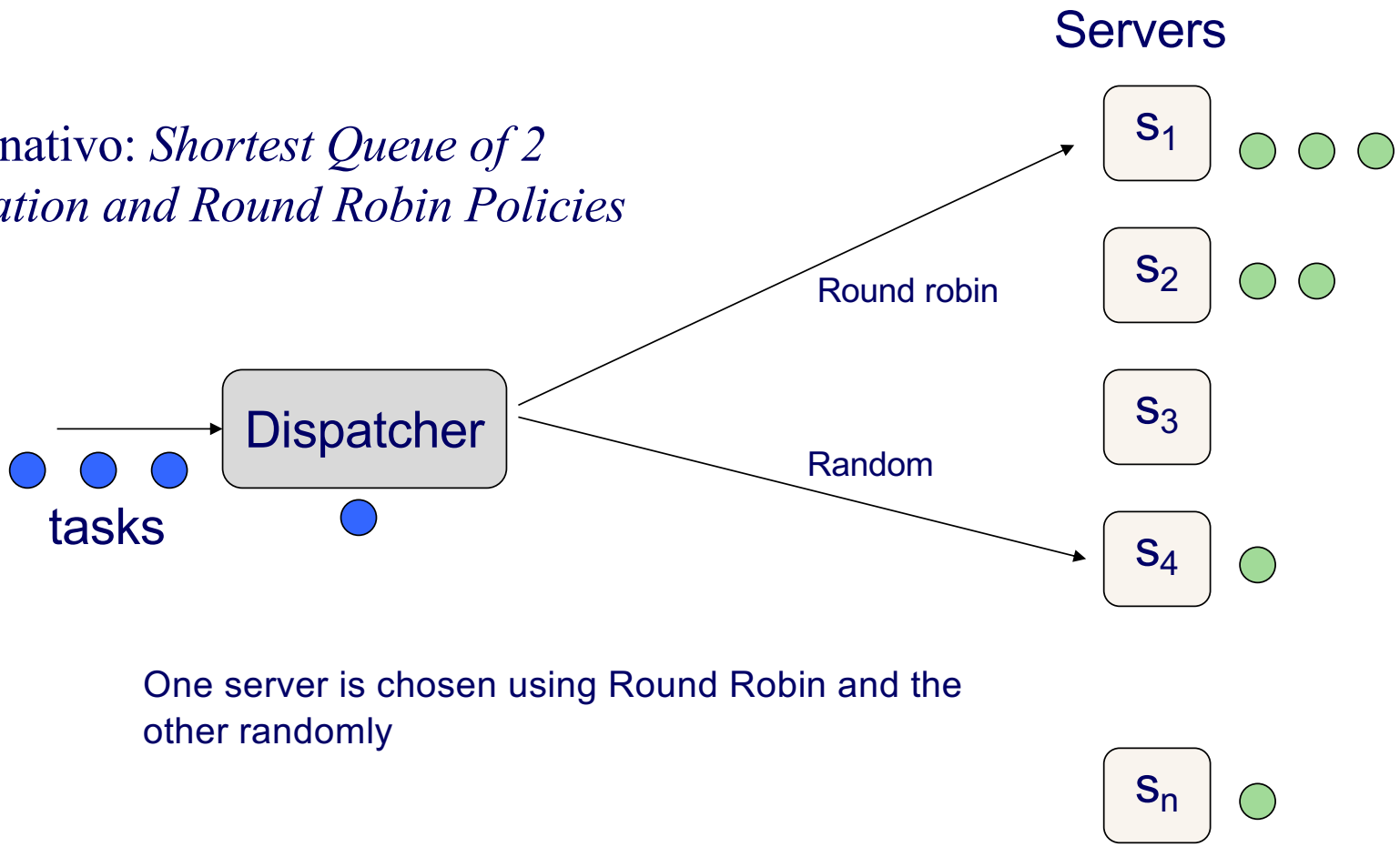
Motivación

- Un algoritmo clásico es: Power of two choices load balancing



Motivación

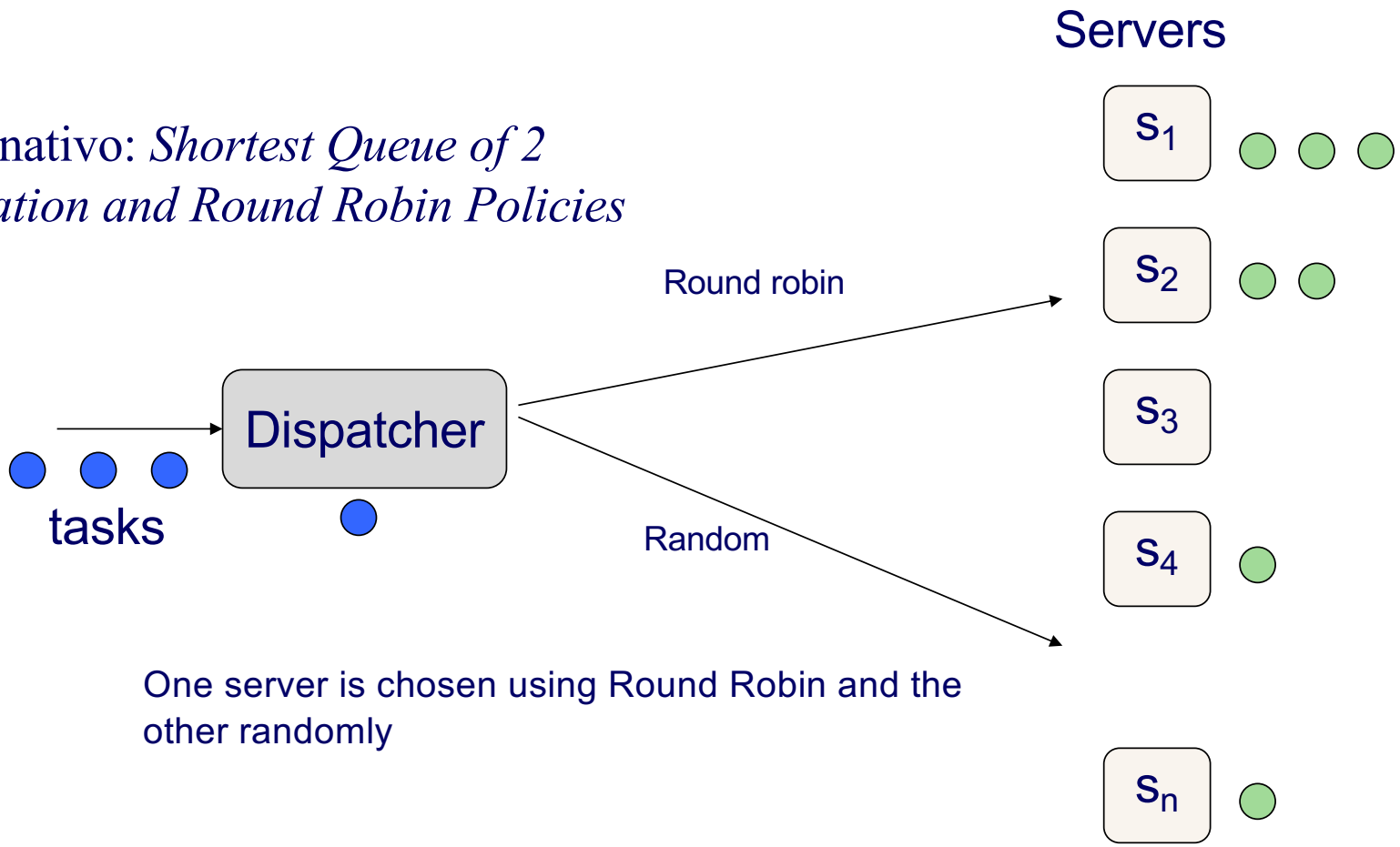
- Algoritmo alternativo: *Shortest Queue of 2 with Randomization and Round Robin Policies*



One server is chosen using Round Robin and the other randomly

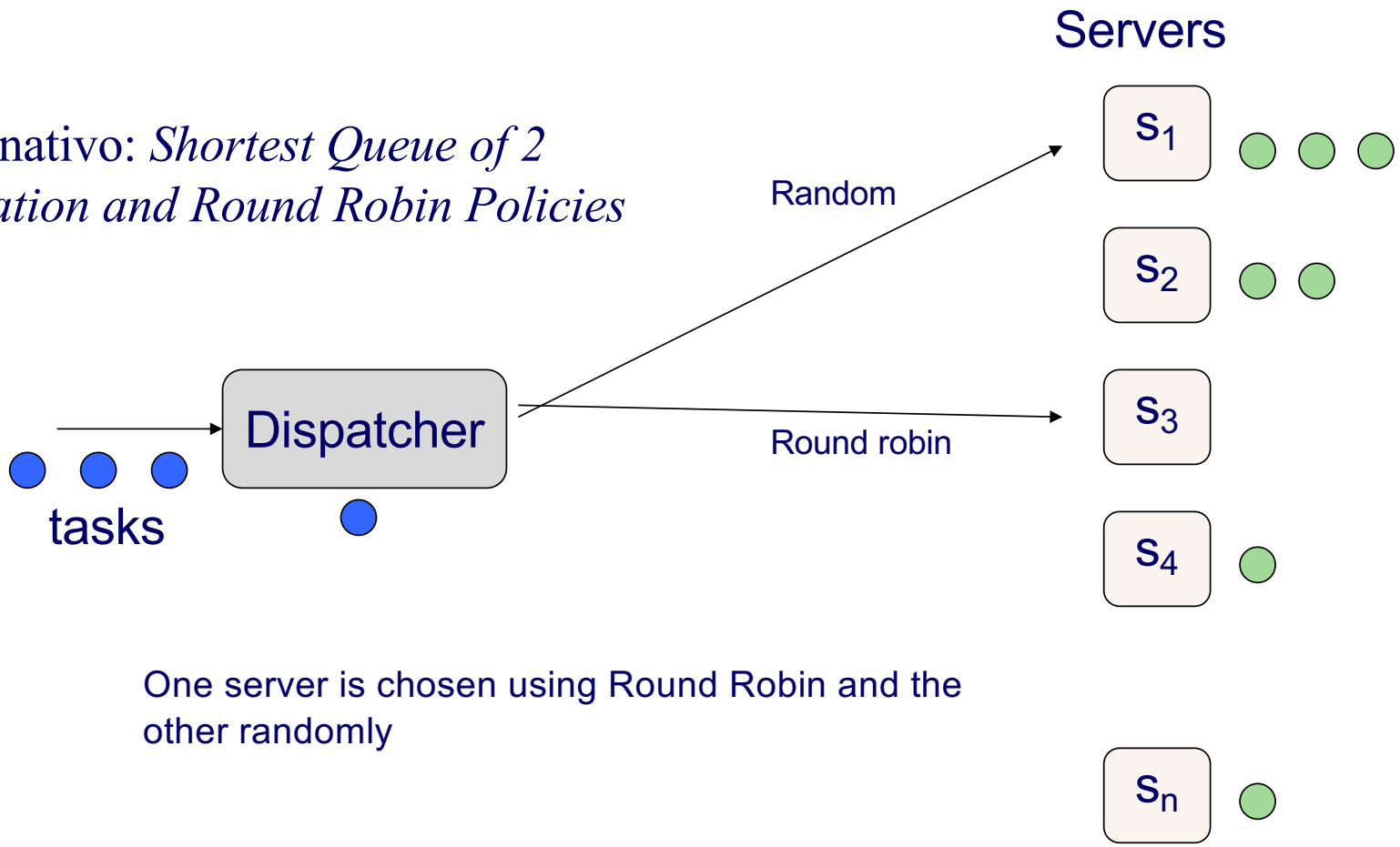
Motivación

- Algoritmo alternativo: *Shortest Queue of 2 with Randomization and Round Robin Policies*



Motivación

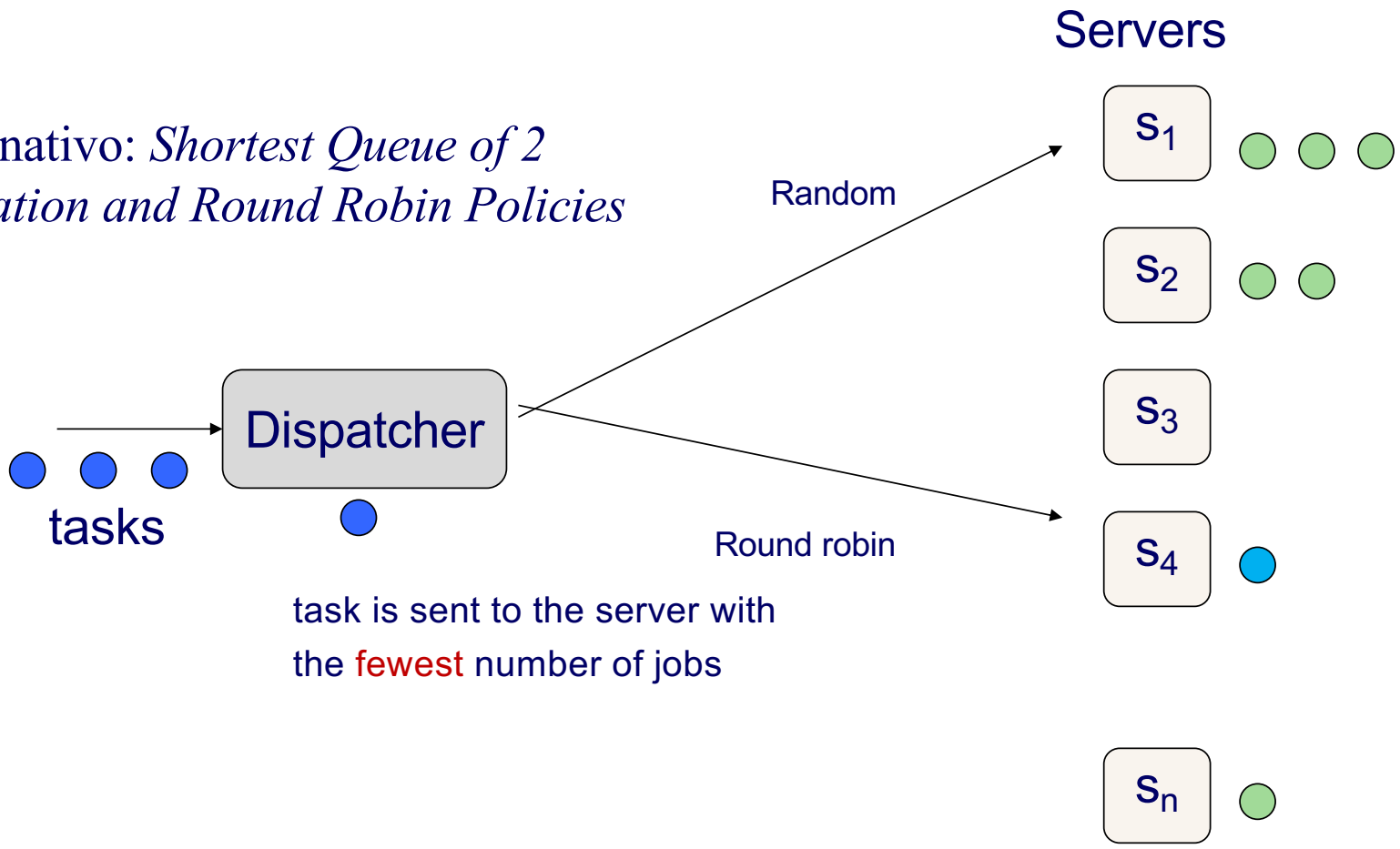
- Algoritmo alternativo: *Shortest Queue of 2 with Randomization and Round Robin Policies*



One server is chosen using Round Robin and the other randomly

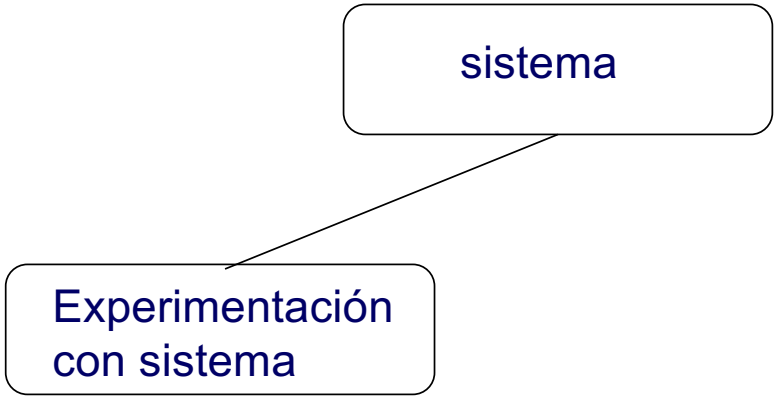
Motivación

- Algoritmo alternativo: *Shortest Queue of 2 with Randomization and Round Robin Policies*

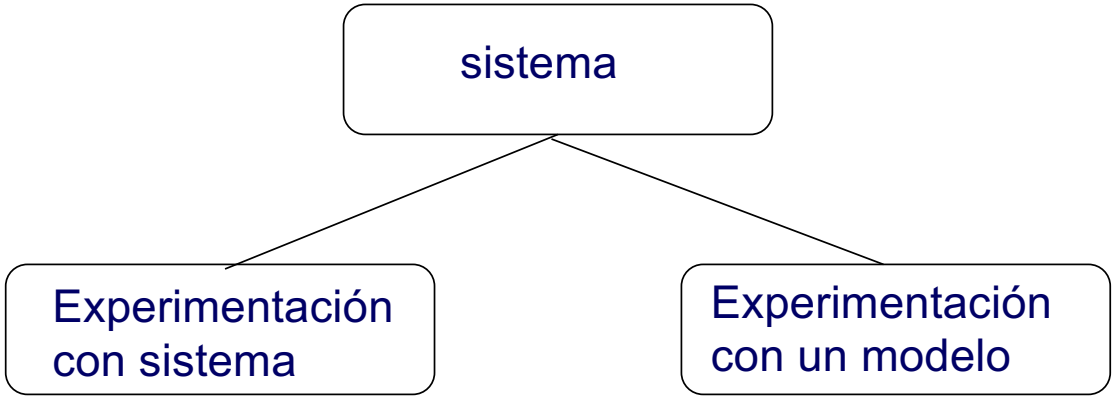


Enfoques de estudio

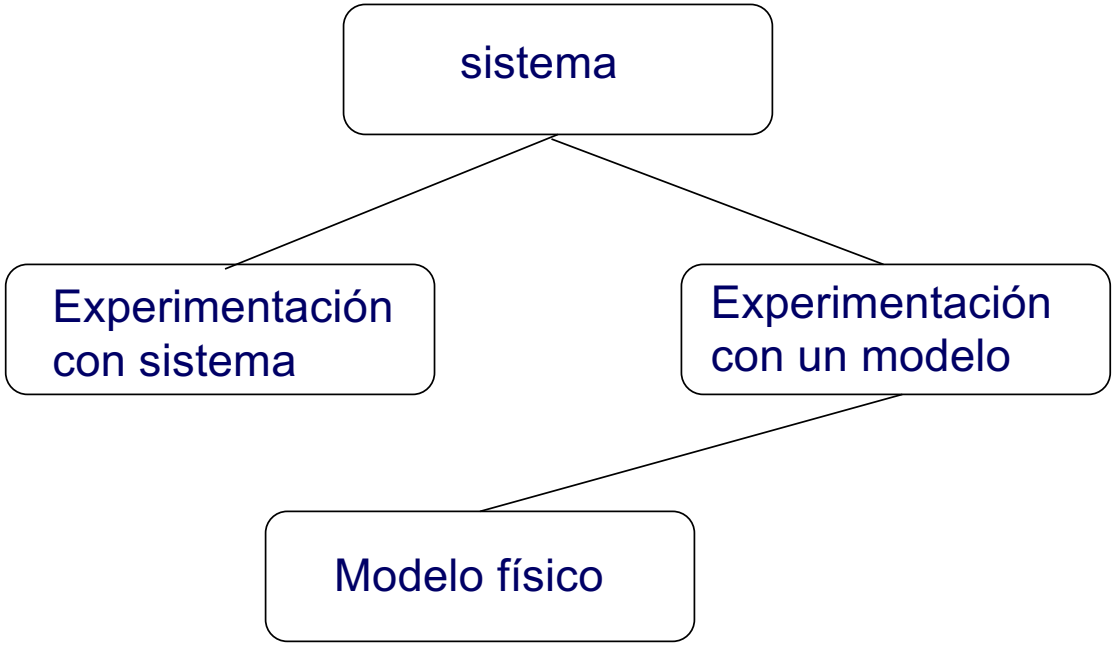
Enfoques de estudio



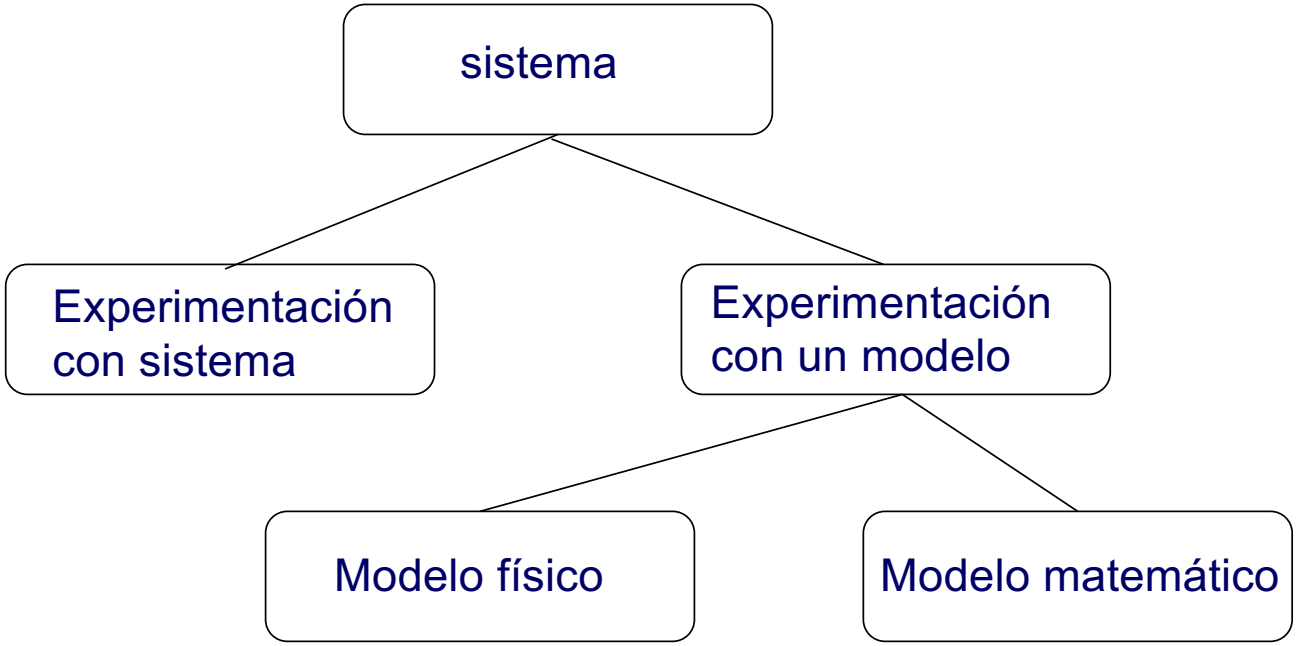
Enfoques de estudio



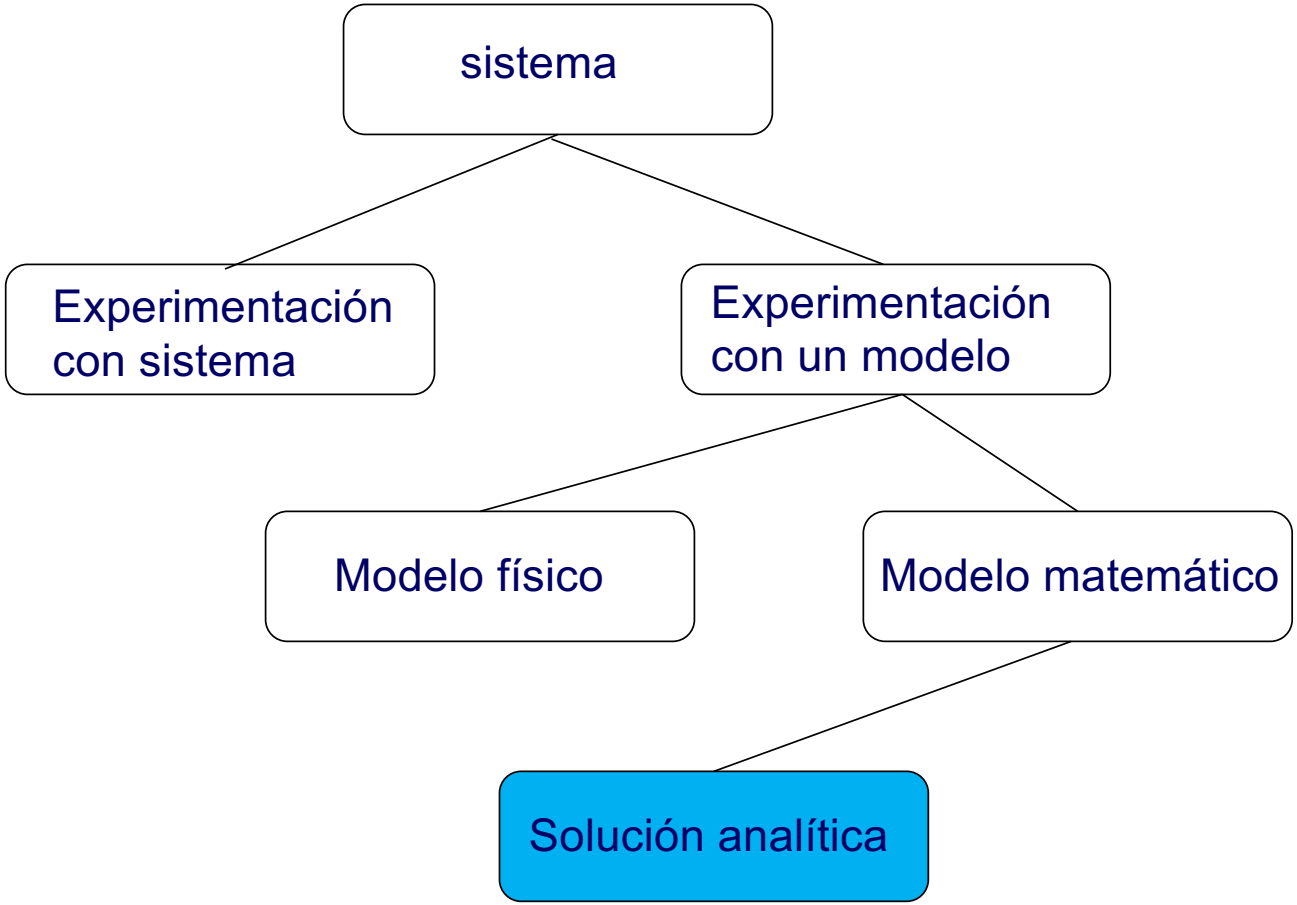
Enfoques de estudio



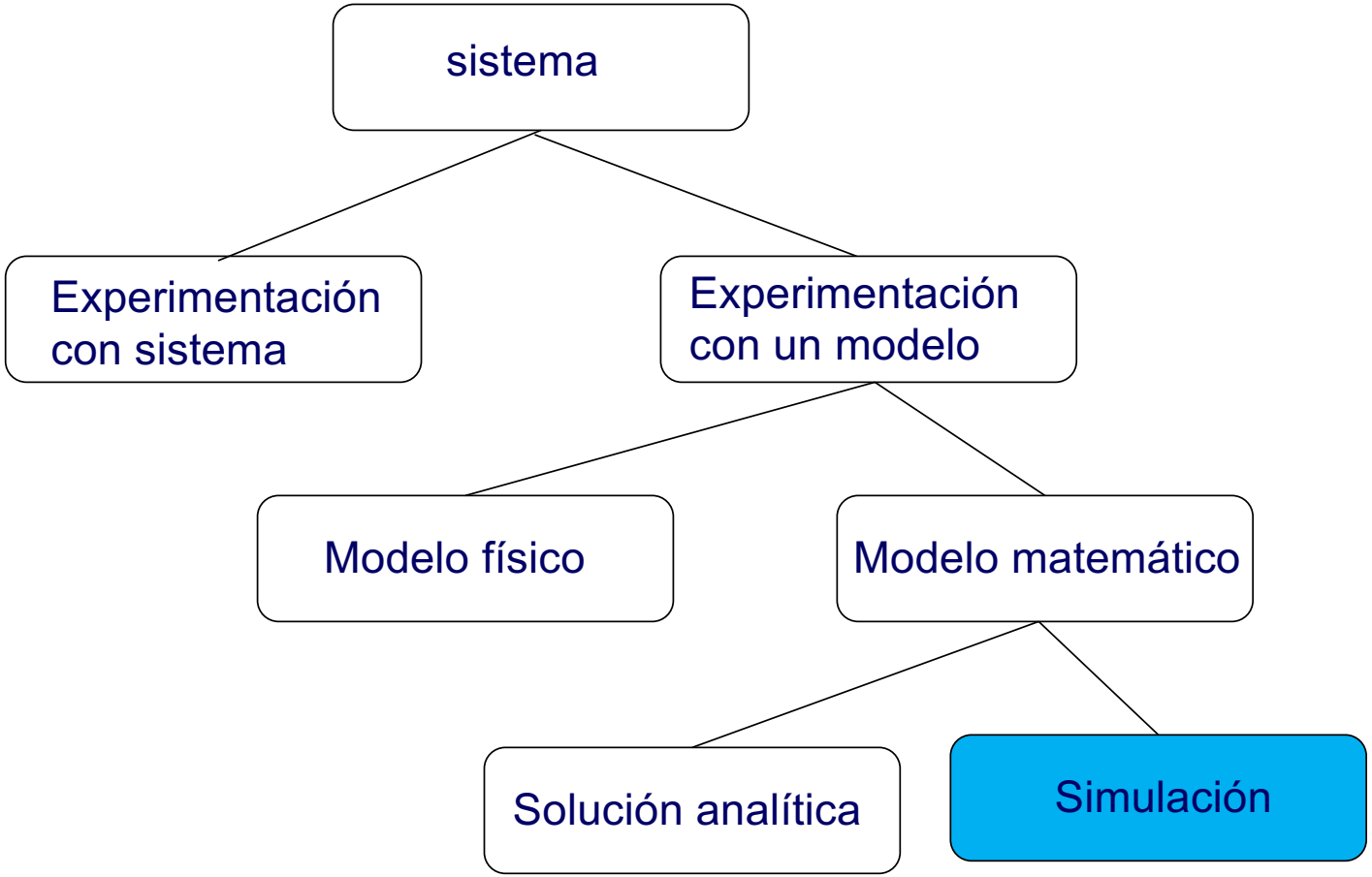
Enfoques de estudio



Enfoques de estudio



Enfoques de estudio para el ejemplo



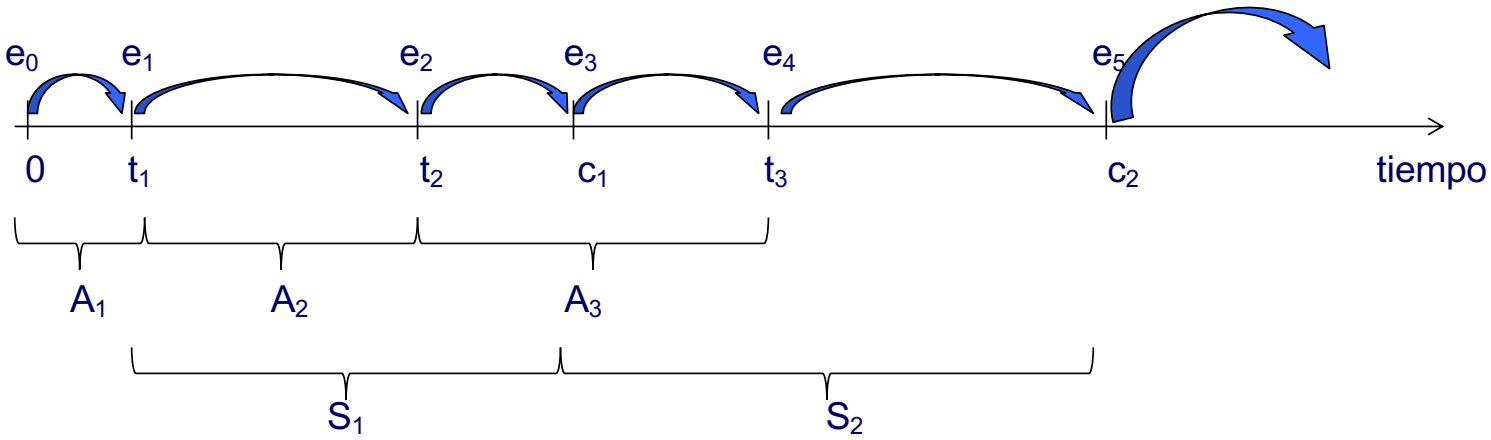
Simulación de eventos discretos

- Modela un sistema cuyo estado global cambia en función del tiempo
- El estado global se actualiza cada vez que ocurre un evento
- Características
  - Dinámica
  - Estocástica
  - Discreta
  - Tiempo discreto

Características

- Dinámica: el estado del sistema varía con el tiempo
- Estocástica
  - Entradas y/o relaciones se modelan como variables aleatorias
  - Las salidas de la simulación deben tratarse como variables aleatorias
  - Ejemplo: la llegada de peticiones a un servidor Web no es determinista, sigue una determinada función de distribución
- Discreta: las variables de estado cambian de valor en instantes separados en el tiempo
  - El número de paquetes recibidos cambia cuando llega un paquete
  - El número de procesos en una cola de ejecución cambia cuando llega un nuevo proceso para ejecutar
- Tiempo discreto: Las variables cambian en un conjunto discreto/numerable de instantes de tiempo

Eventos típicos



- e_i= evento del sistema
  - t_i = tiempo de llegada de una petición, tarea, trabajo, etc.
  - c_i= t_i+D_i+S_i= tiempo de servicio de una petición, tarea, etc
- A_i = t_j-t_i = tiempo entre llegadas
- S_i= tiempo de servicio para la petición, tarea, trabajo, etc.
- D_i= tiempo en la cola del servidor que atiende peticiones, etc.

Algoritmo de simulación

```
Inicializar las variables de estado
Tiempo = 0
Obtener el primer evento
while ((no haya mas eventos) AND (tiempo < máximo)) {
        avanzar el tiempo
        Obtener/eliminar el siguiente evento de la lista
        Procesar el evento {
                Actualizar el estado global
                Actualizar las estadísticas de la simulación
                Generar nuevos eventos y añadirlos a la lista
                de eventos
        }
}
Imprimir resultados
```

Generación de eventos

- Dirigidas por traza
  - Se registran las trazas de eventos que ocurren en un sistema real y se alimenta el simulador con las mismas
- Dirigidas por una distribución aleatoria
  - Los eventos y entradas son generadas a partir de una determina función de distribución
  - Ejemplo: la llegada de trabajos para su ejecución se puede modelar como la cantidad de tiempo que transcurre entre dos llegadas consecutivas. Esta cantidad se puede modelar con una función de distribución exponencial

Generación de valores aleatorios

- Un generador de números pseudoaleatorios genera una secuencia de valores z_i, que se aproximan a la distribución U(0,1)
- Los generadores de números pseudoaleatorios parten de una semilla inicial z0
  - Distintos valores de la semilla generan distintas secuencias de números

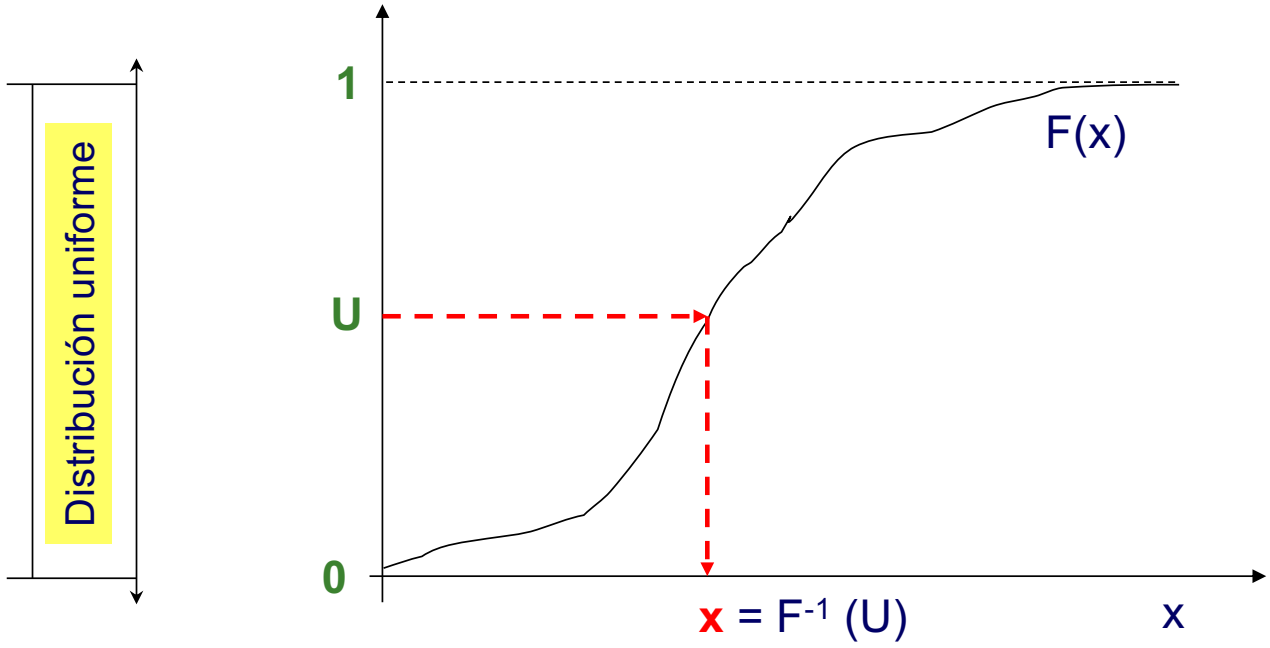
Requisitos de un generador de números pseudoaleatorios

- Aleatoriedad
  - Muestras aproximadas a U(0,1)
  - Sin correlación entre las muestras
- Eficiencia
  - Mide el tiempo y memoria necesaria para la generación
- Periodo máximo
  - Número de muestras que se pueden obtener antes de repetir la secuencia
  - El generador *Mersenne twister* tiene un periodo de $2^{19937}-1$
- Secuencia reproducible
  - Para poder repetir exactamente la misma simulación

Generación de variables aleatorias

- ¿Cómo generar variables aleatorias de otras funciones de distribución a partir de una muestra de variables aleatorias con distribución U(0,1)?
- Métodos
  - Método de la transformada inversa
  - Método de la composición
  - Método de la convolución
  - Método de aceptación-rechazo

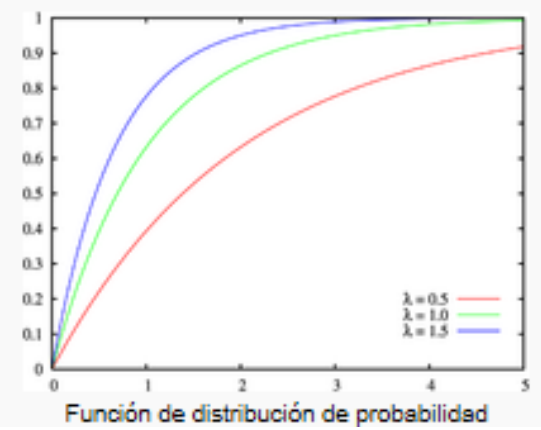
Método de la transformada inversa



Ejemplo: función de distribución exponencial

- Si X es una variable aleatoria exponencial con parámetro λ:
- Función de distribución:

$$F(x)=\begin{cases}1-e^{-\lambda x} & \text{si } x \geq 0\\ 0 & \text{si } x<0\end{cases}$$



Función de distribución de probabilidad

- Media y varianza

$$E[X]=\frac{1}{\lambda} \qquad V(X)=\frac{1}{\lambda^2}$$

Ejemplo: función de distribución exponencial

- Si *u* es una muestra de una variable aleatoria U(0,1)
- Entonces:

$$x = -\frac{1}{\lambda}\ln(1-u)$$

- *x* es una muestra de una variable aleatoria exponencial con parámetro λ

Entornos y lenguajes de simulación

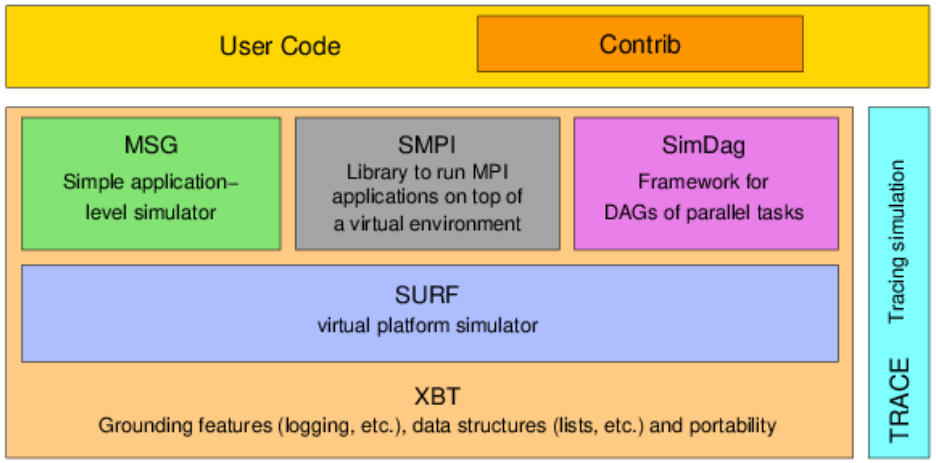
- OMNet++
- PARSEC
- Desmo-J
- JavaSim
- Adves
- OPNET
- Arena
- Simio
- Simscript
- Simulink
- Simgrid

Introducción a SIMGRID

- SIMGRID: *Versatile Simulation of Distributed Systems*
- *https://simgrid.org/*
- Permite la simulación de sistemas distribuidos de gran escala
  - Grids, Clouds, HPC, P2P, aplicaciones MPI, …

Introducción a SIMGRID (versión 3)

- Ofrece un conjunto de funcionalidades para simular aplicaciones distribuidas en entornos distribuidos heterogéneos



  - MSG: modela y simula aplicaciones que intercambiar mensajes
  - SimDag para grafos de aplicaciones paralelas
  - SMPI para código MPI

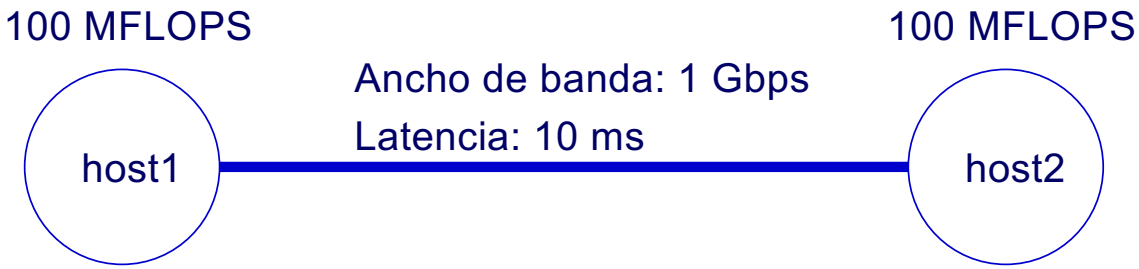
Instalación

https://simgrid.org/

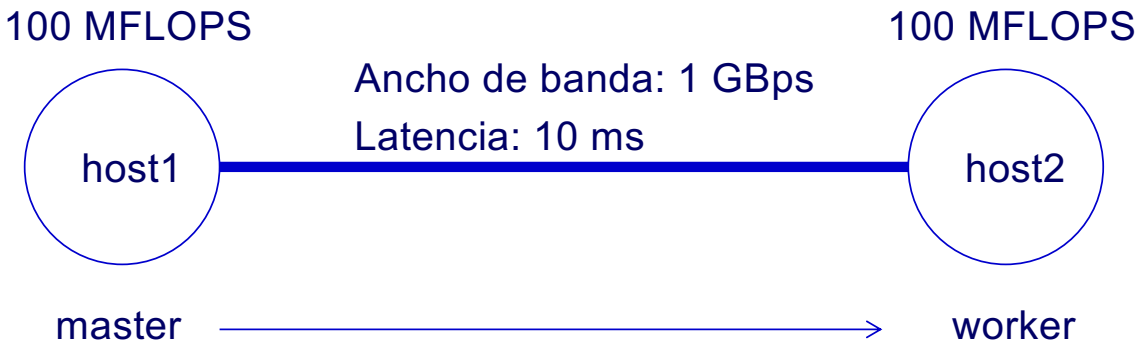
MSG

- MSG: simulación de procesos secuenciales concurrentes que intercambian mensajes
- Principales abstracciones:
  - Agente: algún código, con datos, que ejecuta en un host
  - Task: cantidad de trabajo a realizar y datos a intercambiar
  - Host: localización en la que ejecutan los agentes
  - Mailbox: lugar donde se envían los mensajes y reciben los mensajes
    - Independientes de la posición de la red
    - Los mensajes se envía a un *mailbox* y se reciben de un *mailbox*
    - Se identifican como strings
      - Funcionamiento síncrono o asíncrono

Ejemplo


100 MFLOPS
100 MFLOPS
Ancho de banda: 1 Gbps
Latencia: 10 ms
host1
host2

Ejemplo

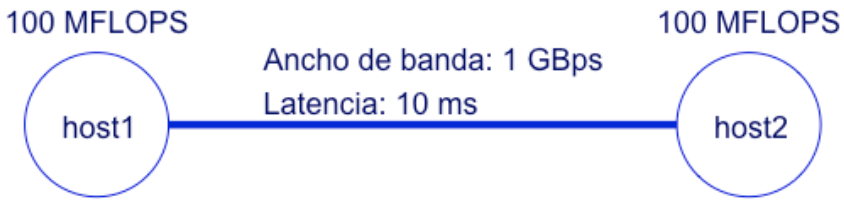


100 tareas

- Tamaño: 100000 bytes
- Duración del cálculo: 50 MFLOPS (cantidad de procesamiento en flops)

Descripción de la plataforma: platform.xml

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM
"http://simgrid.gforge.inria.fr/simgrid/simgrid.dtd">
<platform version="4.1">

<AS id="AS0" routing="Full">
        <host id="host1" speed="1E8"/>
        <host id="host2" speed="1E8"/>

        <link id="link1" bandwidth="1GBps" latency="10ms"/>

        <route src="host1" dst="host2">
                <link_ctn id="link1"/>
        </route>

</AS>
</platform>
```



Descripción del despliegue: deployment.xml

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM
"http://simgrid.gforge.inria.fr/simgrid/simgrid.dtd">
<platform version="4.1">

<!-- The master process -->

    <process host="host1" function="master">
                <argument value="100"/><!--argv[1]:#tasks-->
                <argument value="1"/><!--argv[2]:#workers-->
    </process>

<!-- The workers -->
        <process host="host2" function="worker">
                <argument value="0"/>
        </process>
</platform>
```

Definición del master

```
#define task_comp_size 50000000
#define task_comm_size 100000

int master(int argc, char *argv[])
   int i;
   char mailbox[256];    char buf[256];
   msg_task_t task = NULL;

   number_of_jobs = atoi(argv[1]);    number_of_workers = atoi(argv[2]);

   for (i = 1; i <= number_of_jobs; i++) {// Round Robin
      sprintf(mailbox, "worker-%d", i % number_of_workers);
      sprintf(buf, "Task_%d", i);

      task = MSG_task_create(buf, task_comp_size, task_comm_size, NULL);

      MSG_task_send(task, mailbox);  // mailbox en este caso es ”worker-0”
   }
    for (i = 0; i < number_of_workers; i++) {
       sprintf(mailbox, "worker-%d", i % number_of_workers);
       msg_task_t finalize = MSG_task_create("finalize", 0, 0, 0);
       MSG_task_send(finalize, mailbox);
  }
  return 0;
}
```

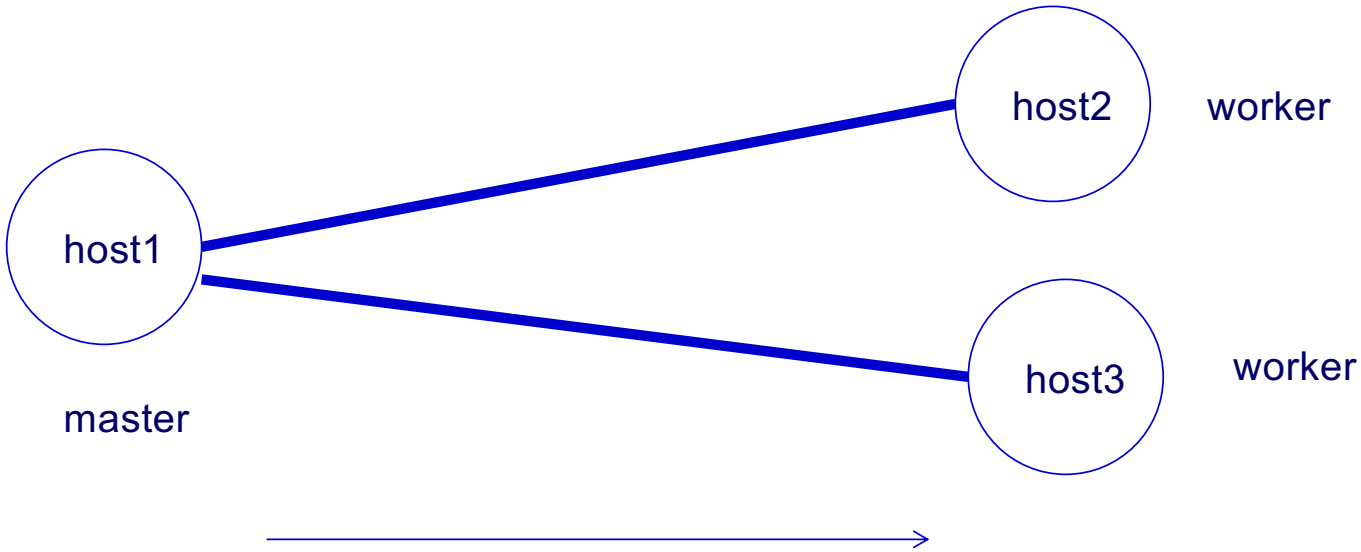
Definición del worker

```
int worker(int argc, char *argv[]) {
   msg_task_t task;
   int errcode;
   char mailbox[80];
   int id = atoi(argv[1]);

   sprintf(mailbox,"worker-%d",id);
   while(1) {
       errcode = MSG_task_receive(&task, mailbox);
       xbt_assert(errcode == MSG_OK, "MSG_task_get failed");

       if (!strcmp(MSG_task_get_name(task),"finalize")) {
           MSG_task_destroy(task);
           break;
       }
       XBT_INFO("Processing '%s'", MSG_task_get_name(task));
       MSG_task_execute(task);
       XBT_INFO("'%s' done", MSG_task_get_name(task));
       MSG_task_destroy(task);
   }
   return 0;
}
```

Función `main()`

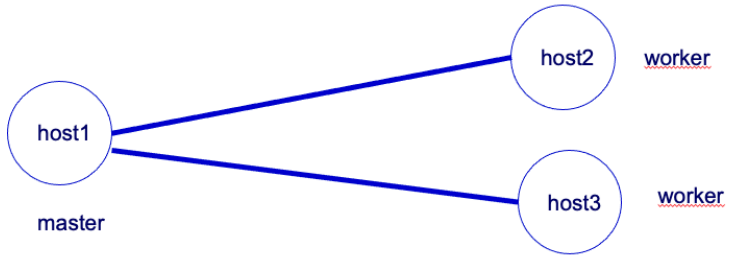
```
/** Main function */
int main(int argc, char *argv[])
{

   MSG_init(&argc,argv);

   /* Declare all existing agent, binding their name to their function */
   MSG_function_register("master", &master);
   MSG_function_register("worker", &worker);

   /* Load a platform instance */
   MSG_create_environment(argv[1]);

   /* Load a deployment file */
   MSG_launch_application(argv[2]);

   MSG_main();
   XBT_INFO("Simulation took %g seconds",MSG_get_clock());
}
```

Modificación



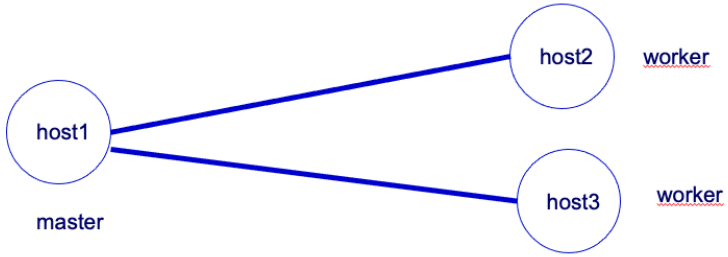
Nueva plataforma

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM
"http://simgrid.gforge.inria.fr/simgrid/simgrid.dtd">
<platform version="4.1">

<AS id="AS0" routing="Full">
        <host id="host1" speed="1E8"/>
        <host id="host2" speed="1E8"/>
        <host id="host3" speed="1E8"/>

        <link id="link1" bandwidth="1GBps" latency="10ms"/>
        <link id="link2" bandwidth="1GBps" latency="10ms"/>

        <route src="host1" dst="host2">
                <link_ctn id="link1"/>
        </route>

        <route src="host1" dst="host3">
                <link_ctn id="link2"/>
        </route>
</AS>
</platform>
```



Nuevo despliegue

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM
"http://simgrid.gforge.inria.fr/simgrid/simgrid.dtd">
<platform version="4.1">

<!-- The master process -->
        <process host="host1" function="master">
                <argument value="100"/><!--argv[1]:#tasks-->
                <argument value="2"/><!--argv[2]:#workers-->
        </process>

<!-- The workers -->
        <process host="host2" function="worker">
                <argument value="0"/>
        </process>
        <process host="host3" function="worker">
                <argument value="1"/>
        </process>
</platform>
```



Código sin fichero de despliegue

```
int main(int argc, char *argv[]){
   MSG_init(&argc,argv);

   MSG_function_register("master", &master);
   MSG_function_register("worker", &worker);

   /* Load a platform instance */
   MSG_create_environment(argv[1]);

   char**argv1=xbt_new(char*,4);          argc = 3;
   argv1[0] = bprintf("%s","master");
   argv1[1] = bprintf("%d",100);
   argv1[2] = bprintf("%d",1);
   argv1[3] = NULL;

   MSG_process_create_with_arguments("master", master, NULL,
                                     MSG_get_host_by_name("host1"), argc, argv1);

   char**argv2=xbt_new(char*,2);       argc = 2;
   argv2[0] = bprintf("%s","server");
   argv2[1] = bprintf("%d",0);
   argv2[2] = NULL;
   MSG_process_create_with_arguments("worker", worker, NULL,
                                     MSG_get_host_by_name("host2"), argc, argv2);

   MSG_main();
   XBT_INFO("Simulation took %g seconds",MSG_get_clock());
}
```

Descripción de plataformas en SimGrid

- Plataforma en SimGrid: archivo XML que describe:
  - Hosts con su potencia de cómputo
  - Enlaces con sus anchos de banda y latencia
  - Rutas entre hosts

```xml
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid/simgrid.dtd">
<platform version="4.1">
  <zone id="first zone" routing="Full">
    <!-- the resources -->
    <host id="host1" speed="1Mf"/>
    <host id="host2" speed="2Mf"/>
    <link id="link1" bandwidth="125MBps" latency="100us"/>
    <!-- the routing: specify how the hosts are interconnected -->
    <route src="host1" dst="host2">
      <link_ctn id="link1"/>
    </route>
  </zone>
</platform>
```

Elementos que se pueden describir

- AS (*autonomous system*): unidad que contiene recursos y que define el encaminamiento entre ellos
- Recursos que incluye un AS:
  - *Host*
  - *Link*
  - *Router*
  - *Cluster*: hosts interconectados por una red dedicada

Especificación de un host

```
<host id="host1”
      speed="1000000000”
      [state=“ON”]
      [availability_file=“host.trace”]
      [core=“4”]
```

- `id`: identificador del host
- `speed`: potencia en Flops
- `availability_file`: ficher de traza asociado
- `state`: estado inicial del host (ON/OFF)
- `core`: número de cores

Fichero de disponibilidad

```
<platform version="4">
  <host id="bob" speed="500Gf" availability_file="bob.trace" />
</platform>
```

Example of "bob.trace" file

```
PERIODICITY 1.0
  0.0 1.0
  11.0 0.5
  20.0 0.8
```

- En el instante 0: el host proporciona 500 Mflops
- En el instante 11.0: entrega 250 Mflops
- En el instante 20.0: entrega el 80%, 400 Mflps
- El proceso se repite

Energía

```
<host id=”host1" speed="100.0Mf,50.0Mf,20.0Mf" >
       <prop id=”watt_per_state" value="95.0:200.0,
       93.0:170.0, 90.0:150.0" />
</host>
```

- Define tres valores de potencia: 100, 50 y 20 Mflops
- Para cada valor se define un par con el consumo de potencia en watios cuando el procesador está libre y cuando está a maximo carga

```
#include "simgrid/plugins/energy.h”
....
sg_host_energy_plugin_init();
...
sg_host_get_consumed_energy(host);
```

Enlaces de red

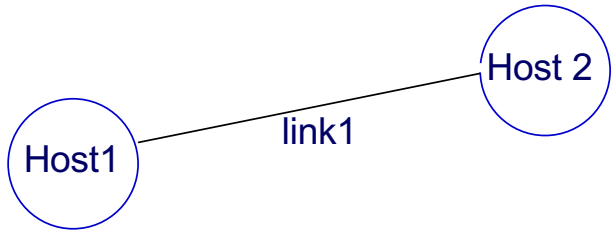
```
<link id="enlace_1"
       bandwidth="125MBps"
       latency="0.01"
      [sharing_policy="SHARED"]   />
```

Unit	Meaning	Duration in seconds
ps	picosecond	10^{-12} = 0.000000000001
ns	nanosecond	10^{-9} = 0.000000001
us	microsecond	10^{-6} = 0.000001
ms	millisecond	10^{-3} = 0.001
s	second	1
m	minute	60
h	hour	60 * 60
d	day	60 * 60 * 24
w	week	60 * 60 * 24 * 7

- `id`: identificador del enlace
- `bandwidth`: ancho de banda en bytes/s
- `latency`: laencia del enlace
- `sharing_policy_`
  - `SHARED` (por defecto): Los flujos comparten el ancho de banda disponible
  - `FATPIPE`: Cada flujo obtiene todo el ancho de banda del enlace

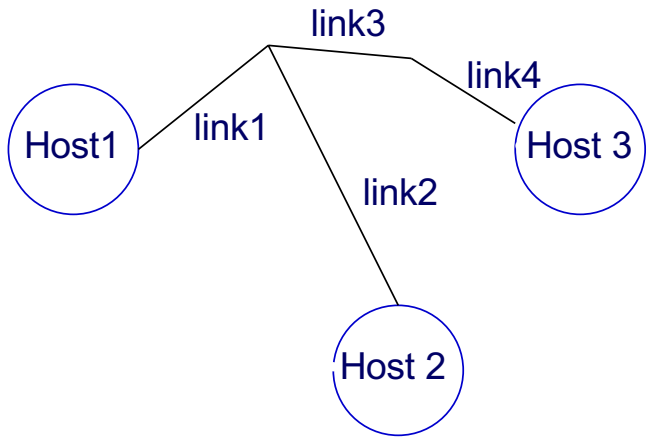
Rutas

```
<host id="host1" speed="1E8"/>
<host id="host2" speed="1E8"/>
<link id="link1" bandwidth="1Gbps" latency="100us"/>

<route src="host1" dst="host2">
     <link_ctn id="link1"/>
</route>
```



Rutas con múltiples saltos

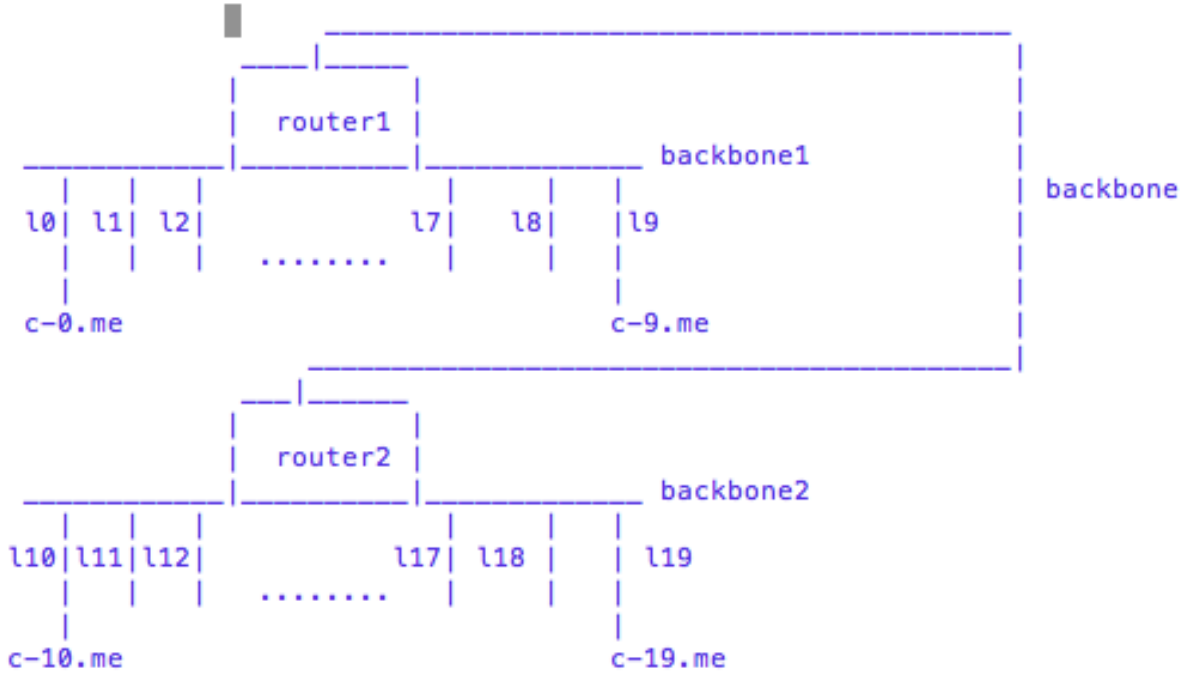
```
<host id="host1" speed="1E8"/>
<host id="host2" speed="1E8"/>
<host id="host2" speed="1E8"/>

<link id="link1" bandwidth="1E6" latency=”100ms"/>
<link id="link2" bandwidth="1E6" latency=”500us"/>
<link id="link3" bandwidth=“2E6" latency=”150ms"/>
<link id="link4" bandwidth=“2E6" latency=”100ms"/>

<route src="host1" dst="host3">
    <link_ctn id="link1"/>
    <link_ctn id="link3"/>
    <link_ctn id="link4"/>
</route>

<route src="host2" dst="host3">
    <link_ctn id="link2"/>
    <link_ctn id="link3"/>
    <link_ctn id="link4"/>
</route>
```



Especificaciones de routers

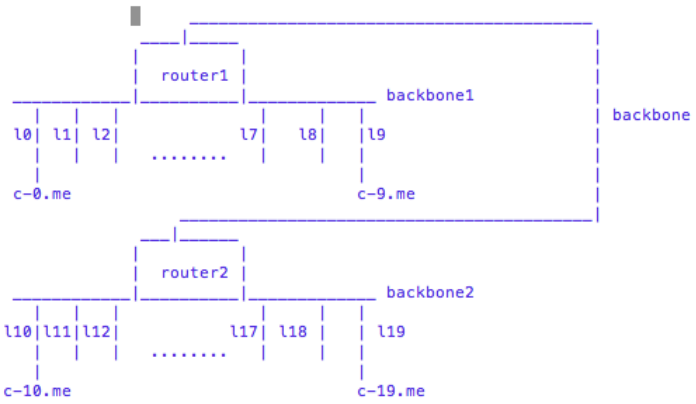
```
<router id="R1">
<router id="R2">

<route src="A" dest="R1">
   <link_ctn id="link1"/>
</route>

<route src="R1" dest="B">
   <link_ctn id="link2"/>
</route>

<route src="R1" dest="C">
   <link_ctn id="link3"/>
</route>
```

Definición de clusters



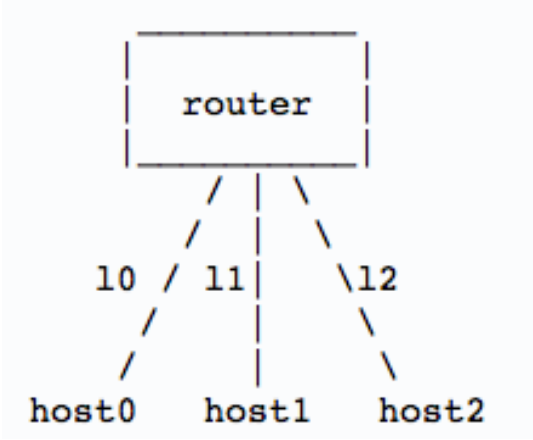
Definición de clusters

```
<AS id="AS0" routing="Full">
        <cluster id="my_cluster_1" prefix="c-" suffix=".me" radical="0-9"
                speed="1Gf" bw="125MBps" lat="50us" bb_bw="2.25GBps"
                bb_lat="500us" />

        <cluster id="my_cluster_2" prefix="c-" suffix=".me" radical="10-19"
                speed="1Gf" bw="125MBps" lat="50us" bb_bw="2.25GBps"
                bb_lat="500us" />

        <link id="backbone" bandwidth="1.25GBps" latency="500us" />

        <ASroute src="my_cluster_1" dst="my_cluster_2"
                 gw_src="c-my_cluster_1_router.me"
                 gw_dst="c-my_cluster_2_router.me">
                <link_ctn id="backbone" />
        </ASroute>
</AS>
```

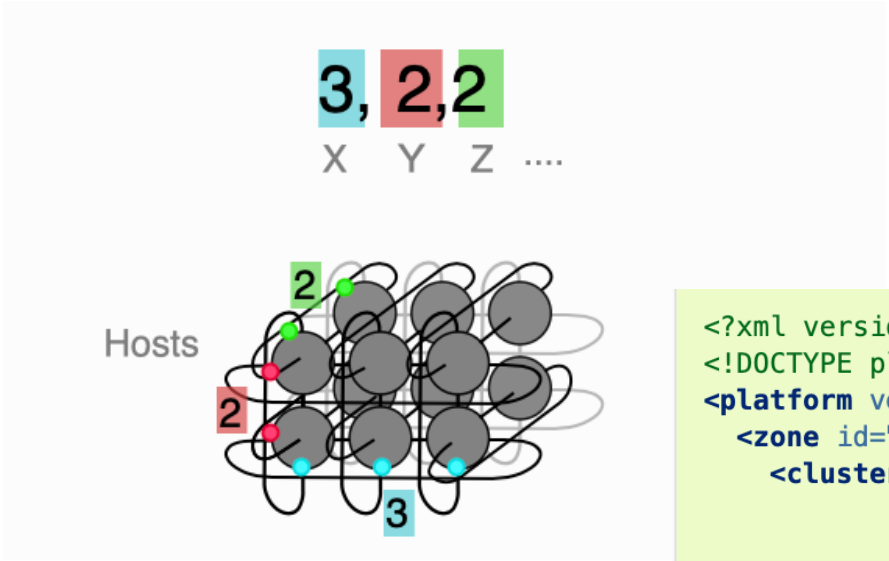


Otro tipo de cluster

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid/simgrid.dtd">
<platform version="4.1">
  <zone id="AS0" routing="Full">
          <cluster id="my_cluster_1" prefix="" suffix="" radical="0-262144"
                   speed="1Gf" bw="125MBps" lat="50us"/>
  </zone>
</platform>
```

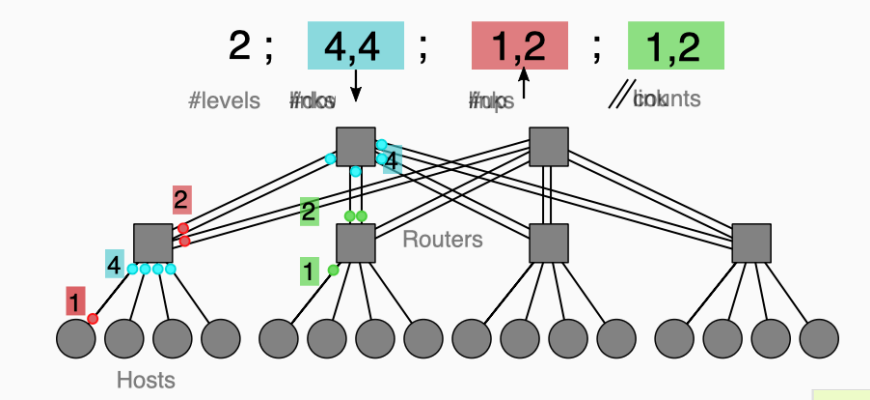


Torus Cluster




```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "https://simgrid.org/simgrid.dtd">
<platform version="4.1">
  <zone id="world" routing="Full">
    <cluster id="bob_cluster" topology="TORUS" topo_parameters="3,2,2"
             prefix="node-" radical="0-11" suffix=".simgrid.org"
             speed="1Gf" bw="125MBps" lat="50us"
             loopback_bw="100MBps" loopback_lat="0"/>
  </zone>
</platform>
```

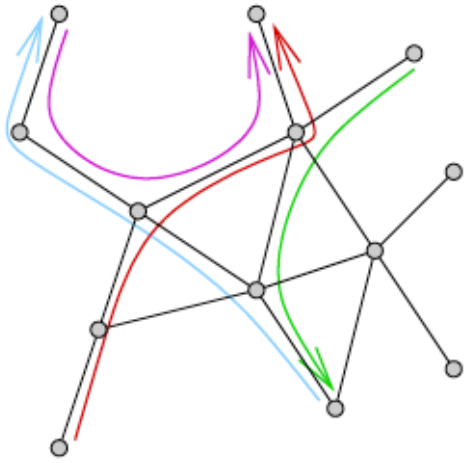
Fat-Tree Cluster




```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "https://simgrid.org/simgrid.dtd">
<platform version="4.1">
  <zone id="world" routing="Full">
    <cluster id="bob_cluster"
             prefix="node-" radical="0-15" suffix=".simgrid.org"
             speed="1Gf" bw="125MBps" lat="50us"
             topology="FAT_TREE" topo_parameters="2;4,4;1,2;1,2"
             loopback_bw="100MBps" loopback_lat="0" />
  </zone>
</platform>
```

Modelización básica de la red

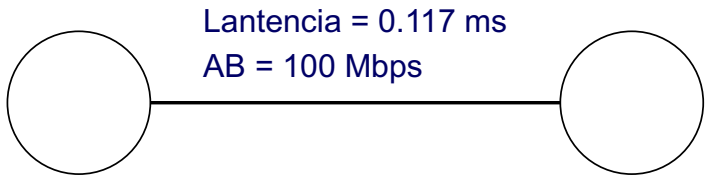
$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

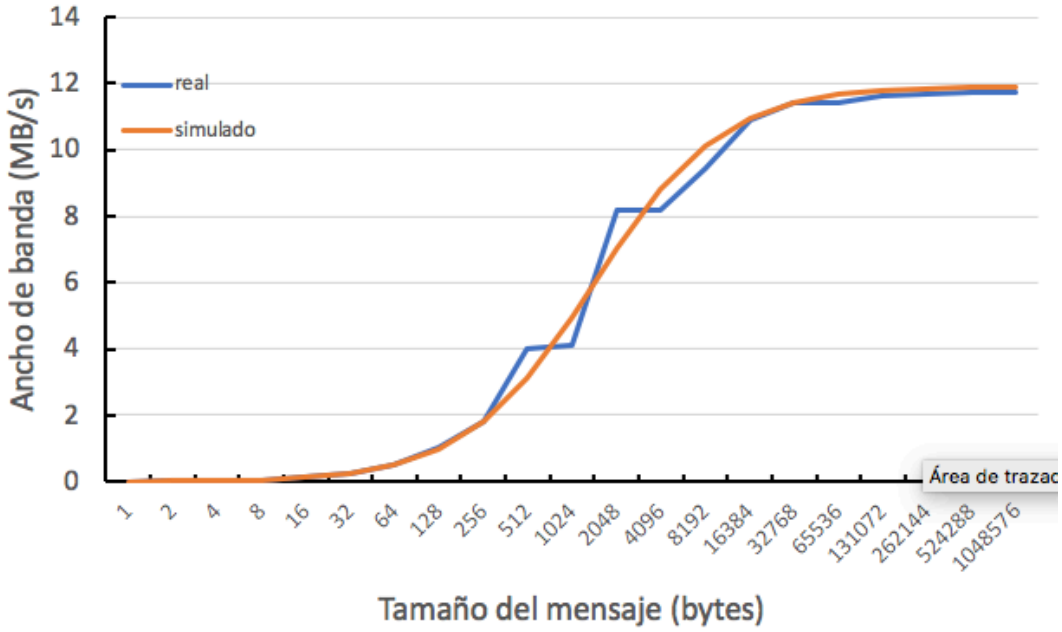


Uso de modelos analíticos
para una rápida simulación

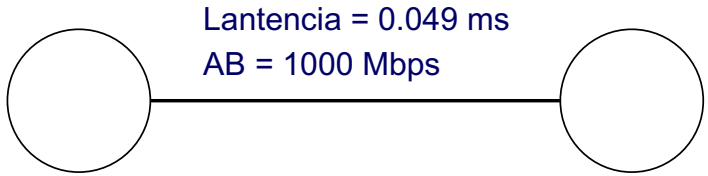
Velho, Pedro, Schnorr, Lucas Mello, Casanova, Henri, Legrand, Arnaud.
On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations.
ACM Transactions on Modeling and Computer Simulation (TOMACS), 2013. WWW doi:10.1145/2517448

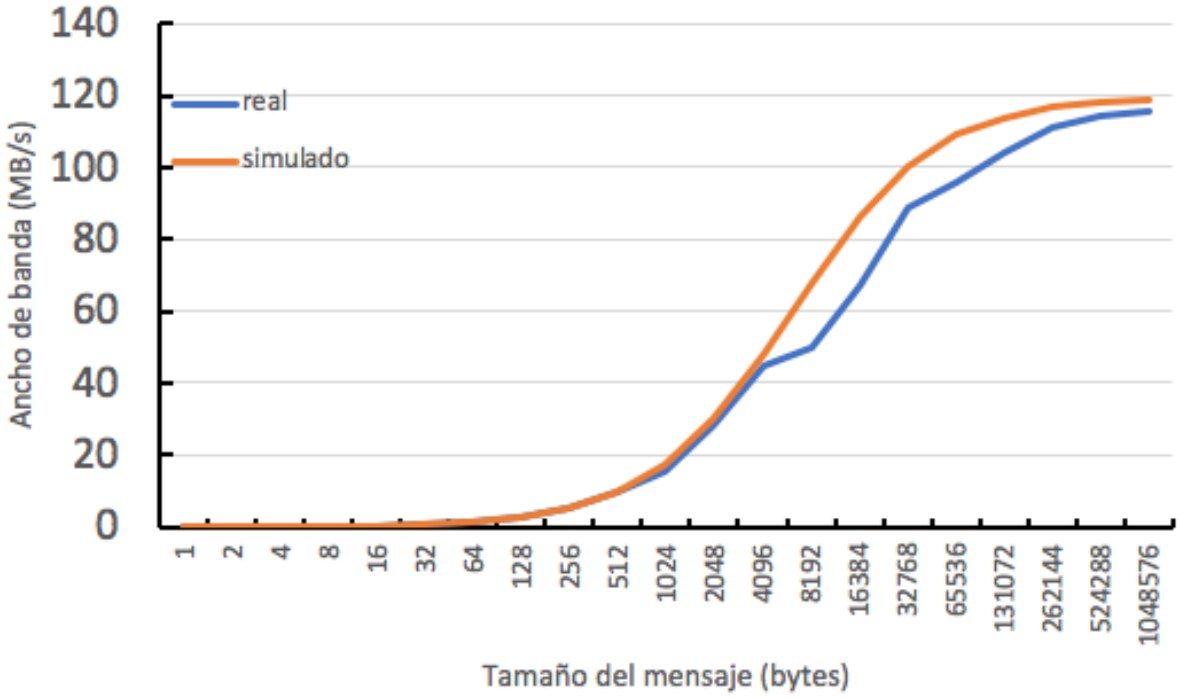
Precisión del modelo de red





Precisión del modelo de red





Tipos de comunicación

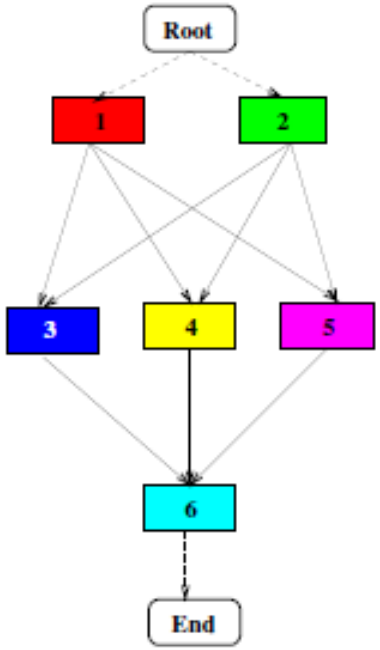
- Comunicaciones síncrona:
  - Buzones síncronos (por defecto)
    - Emisor y receptor deben coincidir en el tiempo
  - Buzones asíncronos
    - MSG_mailbox_set_async(nombre);
    - El emisor en el send espera a que el mensaje llegue al host destino (hay tiempo de red) aunque no espera el proceso receptor
- Comunicación asíncrona (MSG_task_isend, MSG_task_ireceive)
  - El emisor continua su ejecución (tiempo de ejecución 0)
  - El receptor continua su ejecución si no hay mensajes

Modelización de la CPU

- Una CPU tiene:
  - Una potencia en flop/sec
  - Un número de *cores*
- Una tarea requiere *size* flops
- El tiempo para ejecutar la tarea será:

$$\frac{size}{pow}\,\text{sec}$$

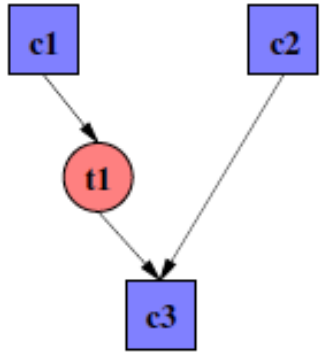
SimDag



- Permite crear DAG de tareas
  - Vértices: tareas
  - Arcos: relaciones de precedencia entre tareas

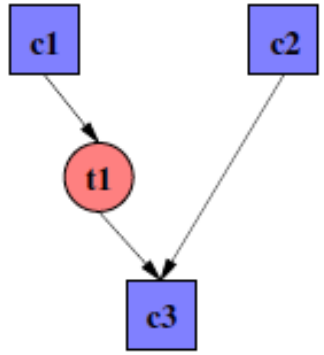
Ejemplo

- DAG formado por:
  - Tres tareas secuencias:
    - c1 procesa 1e9 flops
    - c2 proceso 5e9 flops
    - c3 procesa 2e9 flops
  - Una única transferencia
    - t1 de 5e8 bytes entre c1 y c3



Especificación del ejemplo

```
int main(int argc, char **argv) {
  SD_task_t c1, c2, c3, t1;

  SD_init(&argc, argv);

  c1 = SD_task_create_comp_seq("c1", NULL, 1E9);
  c2 = SD_task_create_comp_seq("c2", NULL, 5E9);
  c3 = SD_task_create_comp_seq("c3", NULL, 2E9);

  t1 = SD_task_create_comm_e2e("t1", NULL, 5e8);

  SD_task_dependency_add ("c1-t1", NULL, c1, t1);
  SD_task_dependency_add ("t1-c3", NULL, t1, c3);
  SD_task_dependency_add ("c2-c3", NULL, c2, c3);

  SD_exit();
  return 0;
}
```



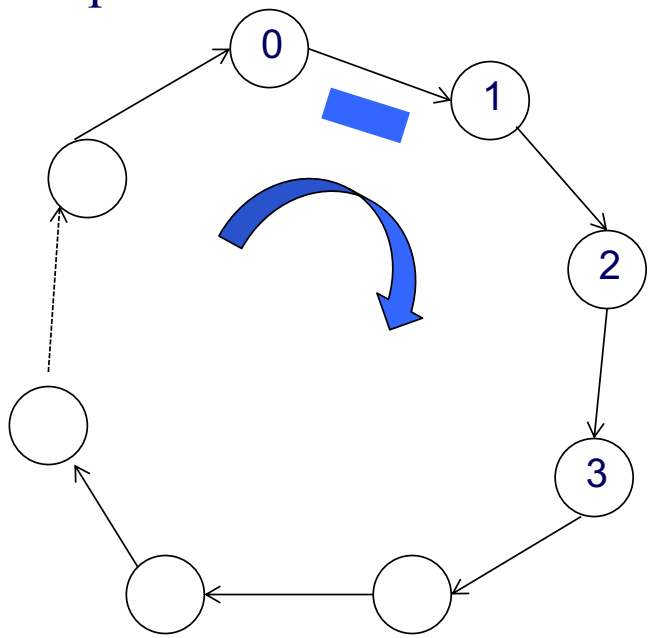
SMPI

- Permite la ejecución de aplicaciones MPI en SimGrid
  - Permite la experimentación reproducible de código MPI
  - Probar código MPI sobre una plataforma simulada

  - https://simgrid.github.io/SMPI_CourseWare/topic_getting_started/platforms/

Escalabilidad

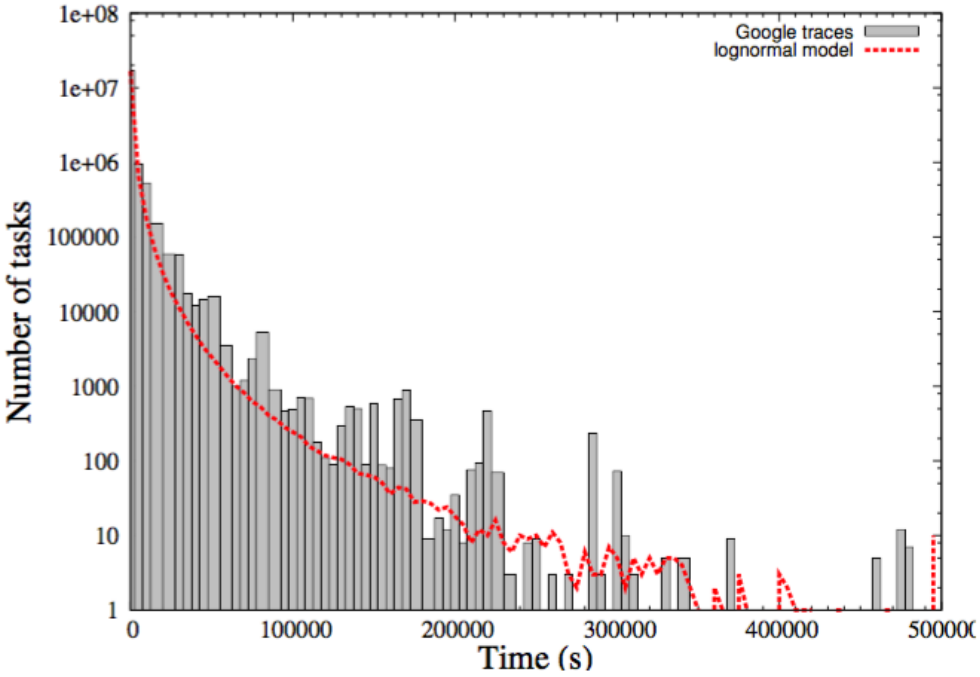
- 500 000 procesos conectados por una red de 1 Gbps
  - Paquete de 10000 bytes
  - Tiempo simulado: 2h 20 min.
  - Tiempo de simulación: 66 s.
  - Memoria: 8 GB
- 1 000 000 de procesos:
  - Tiempo simulado: 3h
  - Tiempo de simulación: 95 s
  - Memoria: 12 GB



Aplicación a Power of two choices load balancing algorithm

- Simular cluster de servidores, generador de tareas y dispatcher
- Conocer la carga de trabajo
- Validar el simulador realizado

Carga de trabajo



Log-normal aprox:

σ=1.42

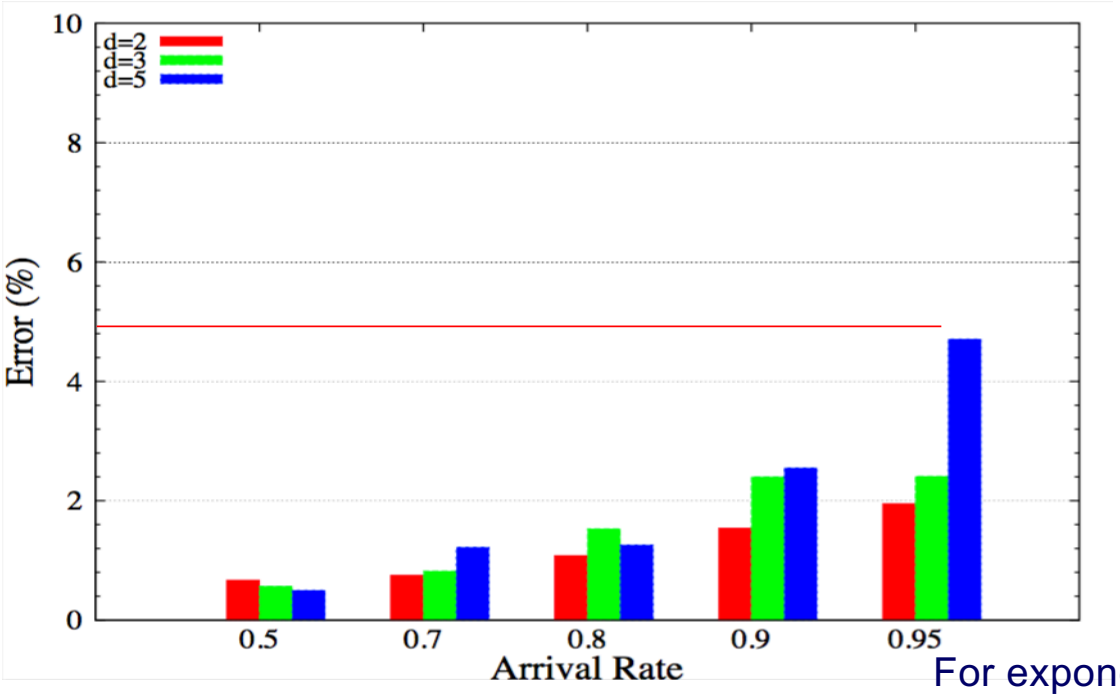
μ=6.25

- Distribution of tasks execution time (Google Datacenter traces vs lognormal characterization) [7]

[7] G. Da Costa, L. Grange, I. De Courchelle *Modeling and Generating large-scale Google-like Workload*. In International Workshop on Resilience and/or Energy-aware techniques for High-Performance Computing. Nov 2016, Hangzhou, China. 2016

Validación del simulador

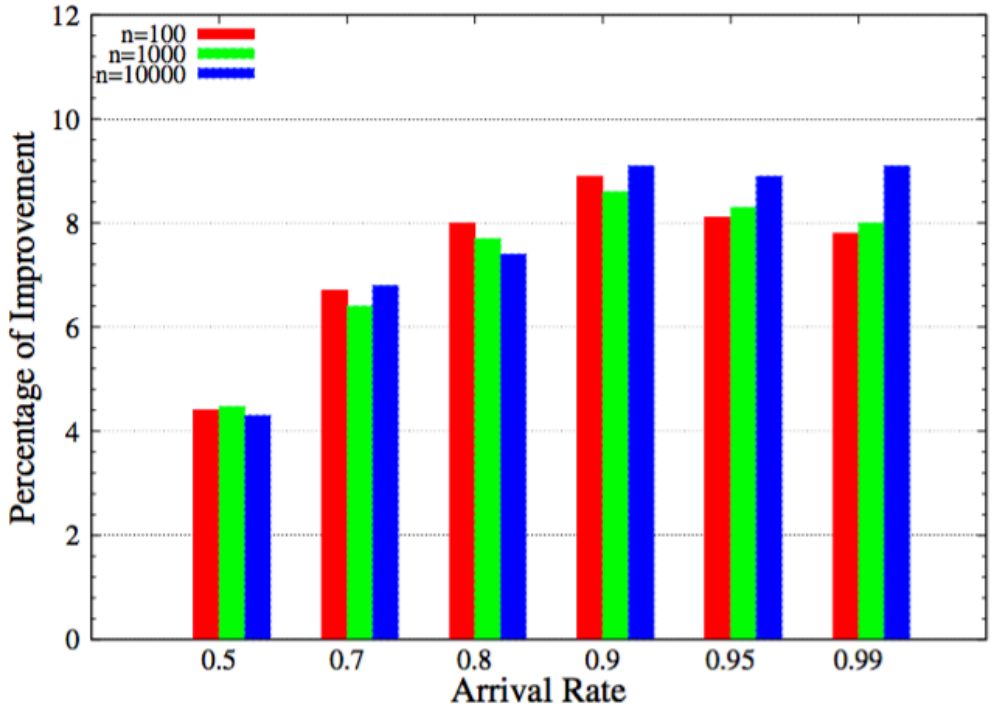
- Relative error (%) comparing the simulation results to the average response time for the Power of two choices:



$$W_{SQ(d)} = \frac{1}{\lambda}\sum_{i=1}^{\infty}\lambda^{\frac{d^i-1}{d-1}} = \sum_{i=1}^{\infty}\lambda^{\frac{d^i-d}{d-1}}$$

For exponential services time

Porcentaje de mejora de SQ-RR(d) vs SQ(d) for Google workload



F. Garcia-Carballeira, A. Calderon, and J. Carretero, "Enhancing the power of two choices load balancing algorithm using round robin policy," *Cluster computing*, pp. 1-14, 2020

Ejemplos de uso de SimGrid

- S. Alonso-Monsalve, F. Garcia-Carballeira, and A. Calderon, "ComBos: A complete simulator of Volunteer Computing and Desktop Grids," *Simulation Modelling Practice and Theory*, vol. 77, pp. 197-211, 2017
- S. Alonso-Monsalve, F. Garcia-Carballeira, and A. Calderon, "Analyzing the Performance of Volunteer Computing for Data Intensive Applications," in The 2016 International Conference on High Performance Computing & Simulation (HPCS 2016). The 14th Annual Meeting, 2016.
- Alonso-Monsalve, F. Garca-Carballeira, and A. Calderón, "A heterogeneous mobile cloud computing model for hybrid clouds," Future generation computer systems, 2018
- S. Alonso-Monsalve, F. Garcia-Carballeira, and A. Calderon, "Fog Computing Through Public-Resource Computing and Storage," in 2nd International Conference on Fog and Edge Mobile Computing (FMEC2017), 2017
- E. del Pozo and F. Garcia-Carballeira, "A generic simulator for edge computing platforms," in *2020 International Conference on High Performance Computing & Simulation (HPCS). barcelona, march 2021*, 2021.

Complete Simulator of BOINC Infrastructures (ComBoS)

- https://arcos-combos.github.io
- Simulador de una infraestructura completa de computación voluntaria basada en BOINC

- S. Alonso-Monsalve, F. Garcia-Carballeira, and A. Calderon, "ComBos: A complete simulator of Volunteer Computing and Desktop Grids," *Simulation Modelling Practice and Theory*, vol. 77, pp. 197-211, 2017

- S. Alonso-Monsalve, F. Garcia-Carballeira, and A. Calderon, "Analyzing the Performance of Volunteer Computing for Data Intensive Applications," in The 2016 International Conference on High Performance Computing & Simulation (HPCS 2016). The 14th Annual Meeting, 2016.

Computación voluntaria

Volunteer PC

1.- Download input files

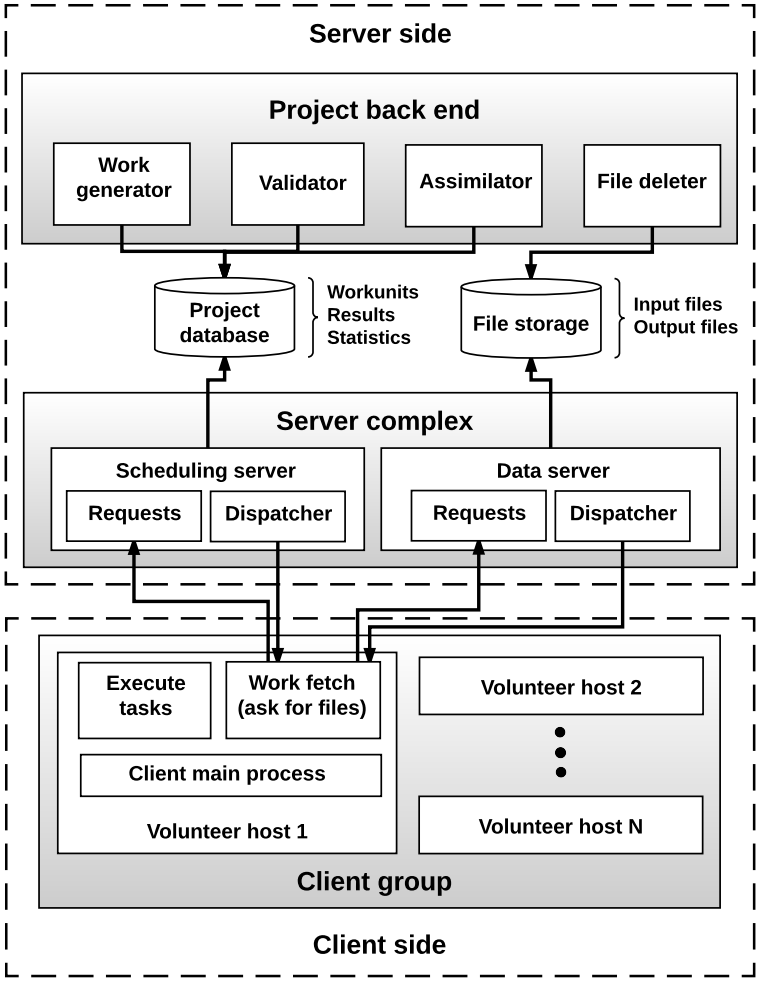
2.- Compute the application

3.- Upload output files

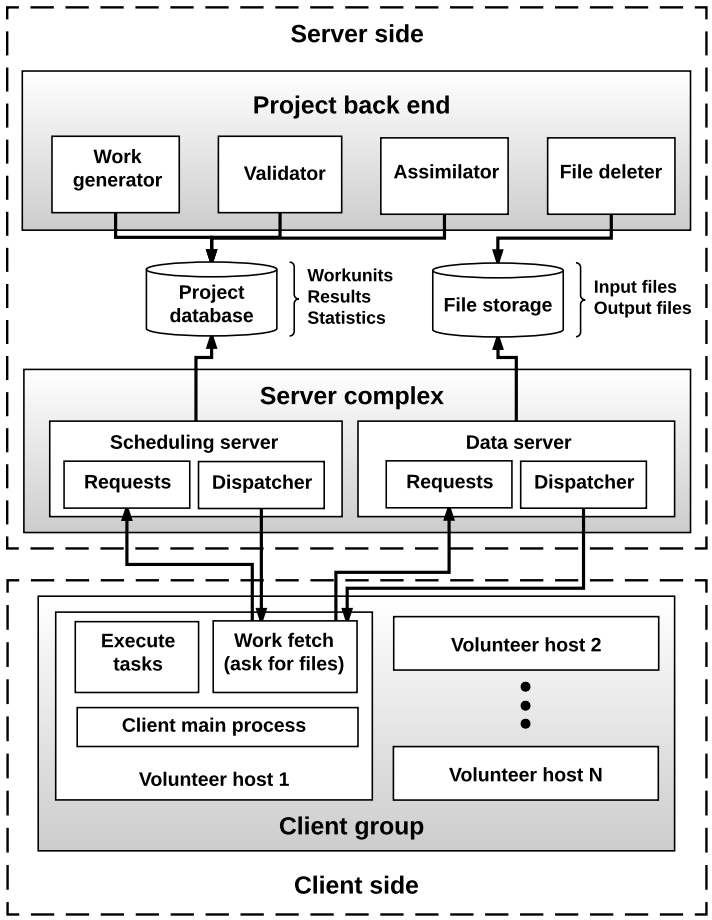
4.- Report results

Project servers

Arquitectura de ComBoS

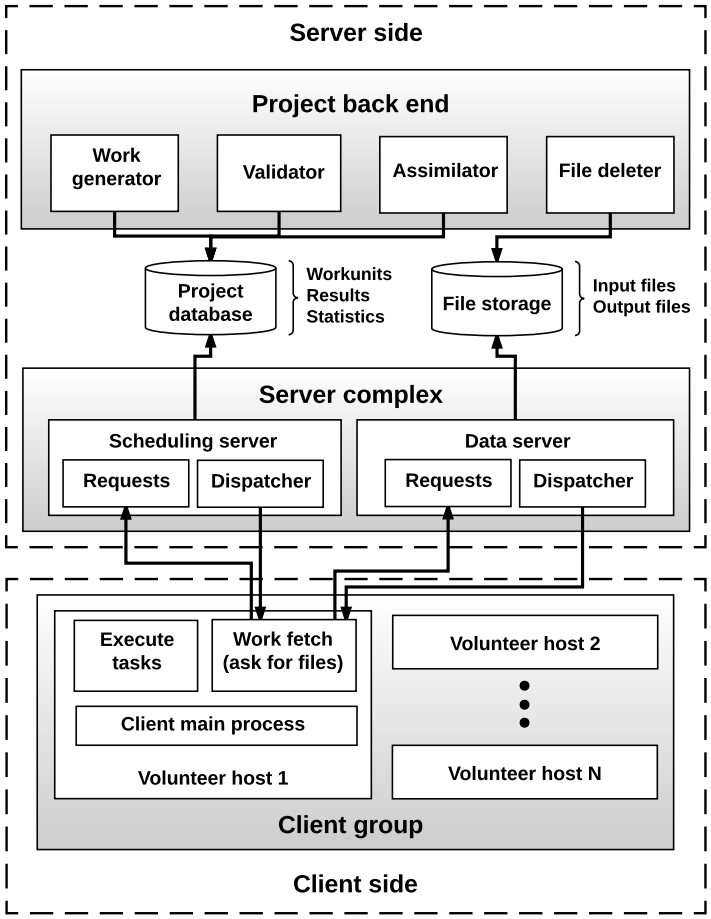


Información para la simulación



- Server side
  - number of scheduling servers
  - number of data servers
  - bandwidth of disks
  - CPU power of servers (GigaFLOPS)
  - input file size
  - task deadline
  - …

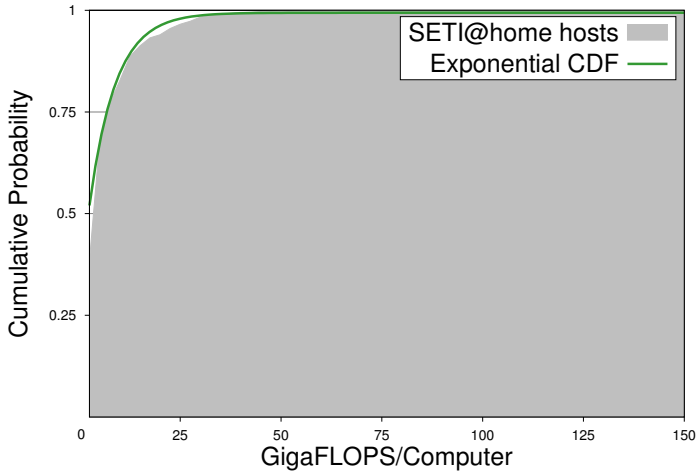
Información para la simulación



- Client side
  - number of groups
  - number of volunteer nodes per group
  - number of projects attached per volunteer in a group
  - bandwidth and latency of the network
  - CPU power model
  - Availability and unavailability model

Definición de la potencia de CPU

- Statistical model



- Traces

CPU model	Number of computers	Avg. cores/computer	GFLOPS/core	GFLOPs/computer
Intel(R) Core(TM) i5-5675R CPU @ 3.10GHz [x86 Family 6 Model 71 Stepping 1]	47	4.00	6.41	25.66
Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz [x86 Family 6 Model 71 Stepping 1]	18	8.00	6.14	49.11
Intel(R) Core(TM) i5-5575R CPU @ 2.80GHz [x86 Family 6 Model 71 Stepping 1]	34	4.00	5.74	22.96
Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz [x86 Family 6 Model 94 Stepping 3]	114	7.95	5.51	43.79
Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz [x86 Family 6 Model 60 Stepping 3]	149	7.95	5.42	43.10
Intel(R) Core(TM) i7-5557U CPU @ 3.10GHz [x86 Family 6 Model 61 Stepping 4]	40	4.00	5.40	21.61
Intel(R) Core(TM) i5-5287U CPU @ 2.90GHz [x86 Family 6 Model 61 Stepping 4]	61	4.00	5.39	21.55
Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz [x86 Family 6 Model 60 Stepping 3]	109	4.00	5.26	21.03
Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz [x86 Family 6 Model 94 Stepping 3]	41	4.00	5.24	20.97
Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz [x86 Family 6 Model 60 Stepping 3]	10	8.00	5.10	40.84
Intel(R) Core(TM) i5-4670 CPU @ 3.40GHz [x86 Family 6 Model 60 Stepping 3]	140	4.00	5.08	20.33
Intel(R) Core(TM) i5-3570K CPU @ 3.40GHz [x86 Family 6 Model 58 Stepping 9]	11	4.00	4.99	19.97
Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz [x86 Family 6 Model 60 Stepping 3]	33	4.00	4.96	19.84

Validación

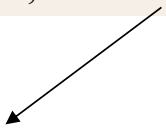
- Compare the simulation results with the statistics of real projects (BOINCstats)
  - SETI@home
  - LHC@home
  - Einstein@home
- Availability model used in the simulation developed in

  J.-M. V. D. P. A. Bahman Javadi, Derrick Kondo, "**Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home,**" *Parallel and Distributed Systems, IEEE Trans- actions*, vol. 22, pp. 1896–1903, 2011.

- Hosts power: model obtained from the 3,900,000 hosts that participate in SETI@home
- ADSL networks and Internet backbone of 10 Gbps

Resultados de la validación (GigaFLOPS)

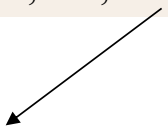
		BOINCstats	ComBoS	
Project	Active Hosts	GigaFLOPS	GigaFLOPS	Error
SETI@home	175,220	864,711	865,001	0.03%
Einstein@home	68,338	1,044,515	1,028,172	1,16%
LHC@home	15,814	7,521	7,392	1,7%



Results published in BOINCstats about real projects

Resultados de la validación (créditos/día)

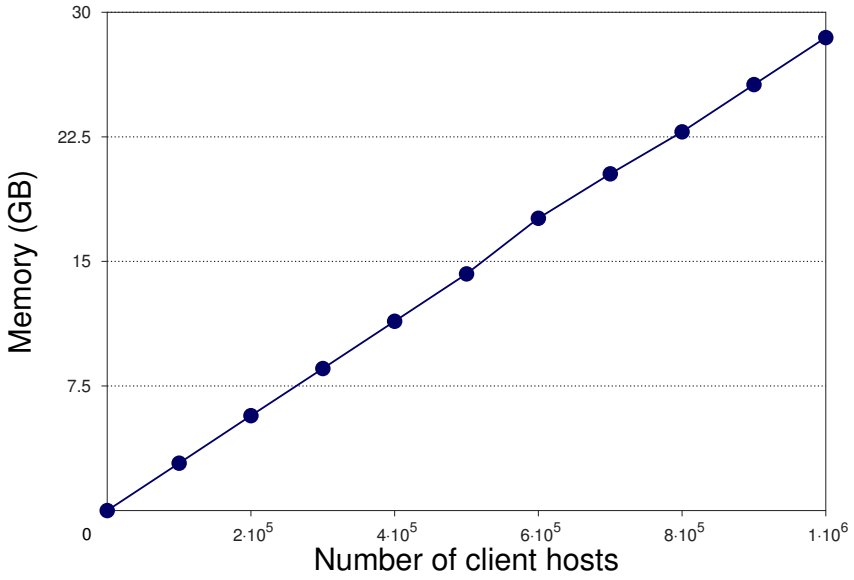
		BOINCstats	ComBoS	
Project	Active Hosts	Credits/day	Credits/day	Error
SETI@home	175,220	171,785,234	168,057,478	2.1%
Einstein@home	68,338	208,902,921	205,634,486	1.56%
LHC@home	15,814	1,504,214	1,393,932	7.3%



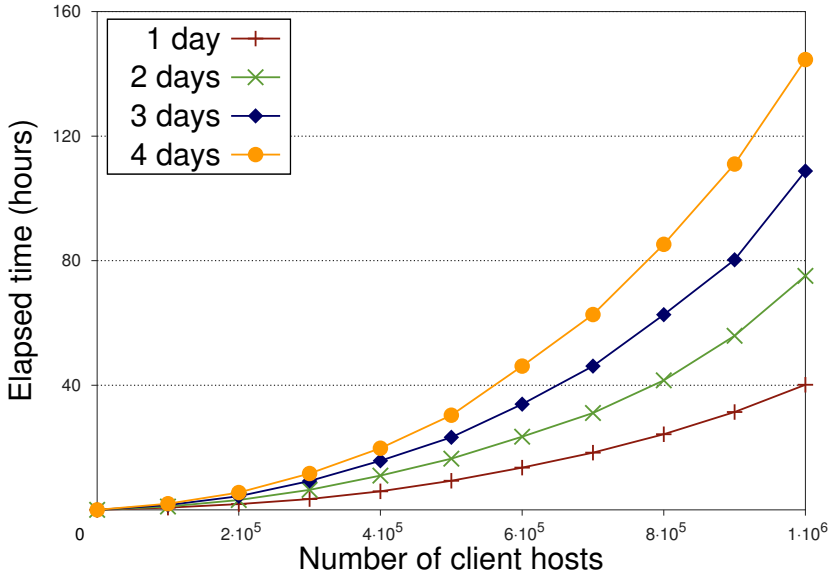
Results published in BOINCstats about real projects

Crédito: medida del trabajo realizado por un cliente. Una máquina de 1 GigaFLOP ejecutando durante todo el día produce 200 créditos.

Rendimiento de ComBoS



Memory usage



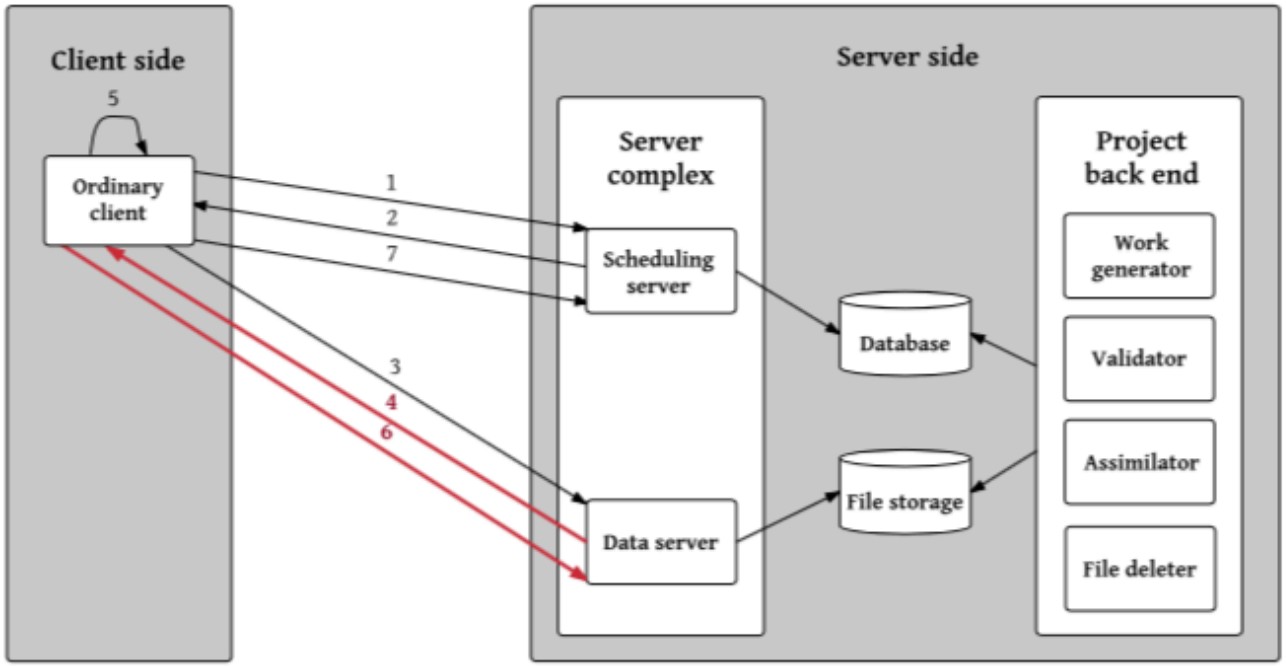
Execution time

Modelo de computación voluntaria para aplicaciones intensivas en datos

- Modelo de computación voluntaria que utiliza nodos como voluntarios de datos
- Los voluntarios de datos participan como el resto de nodos pero el software que se descargan actúa como un servidor de ficheros

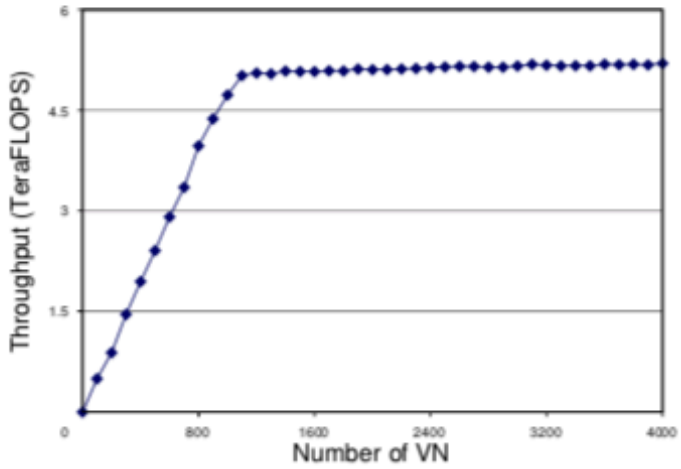
- S. Alonso-Monsalve, F. Garcia-Carballeira, and A. Calderon, "A new volunteer computing model for data-intensive applications," Concurrency and computation: practice and experience. 2017
- S. Alonso-Monsalve, F. Garcia-Carballeira, and A. Calderon, "Improving the Performance of Volunteer Computing with Data Volunteers: A Case Study with the ATLAS@ home Project," in ICA3PP: 16th International Conference on Algorithms and Architectures for Parallel Processing, 2016,

Modelo original de BOINC

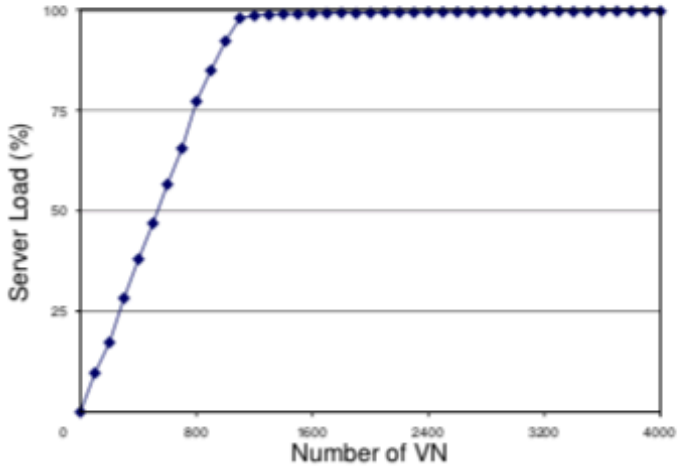


Evaluation: ATLAS@home project

Current performance:

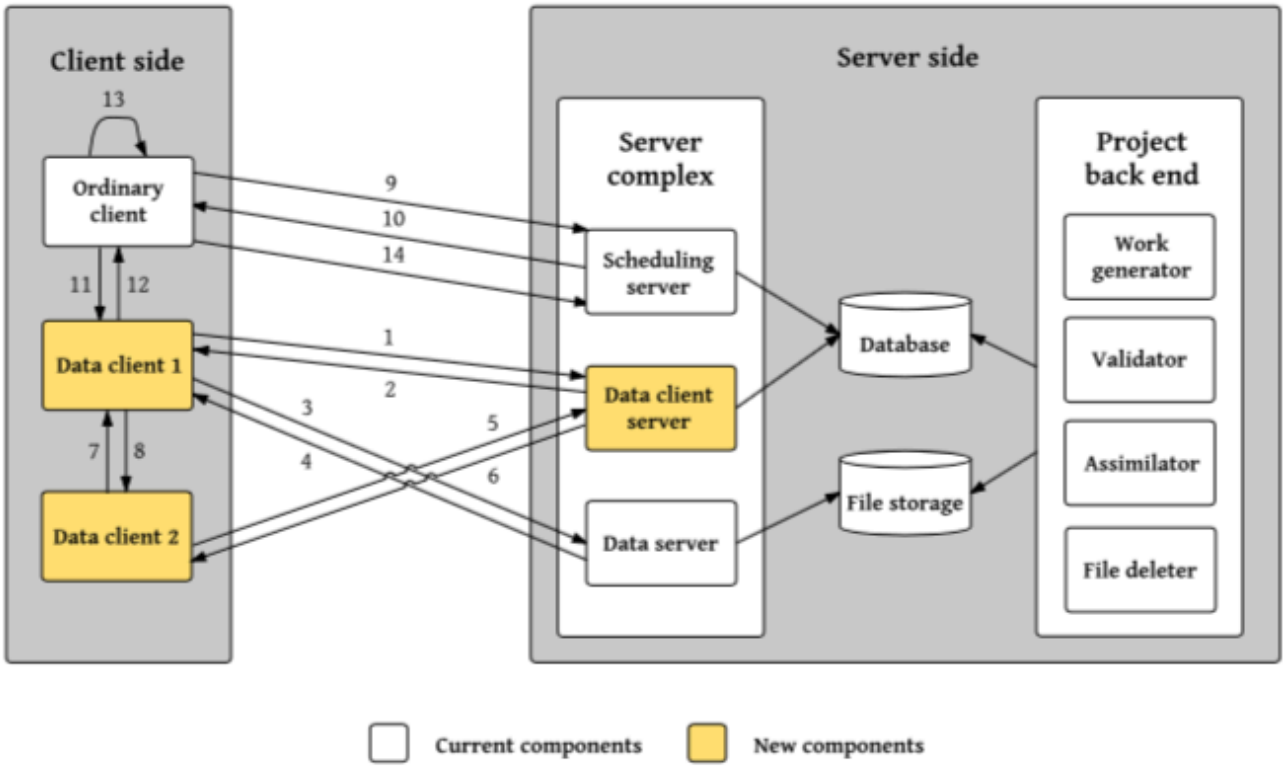


(a) Throughput



(b) Server load

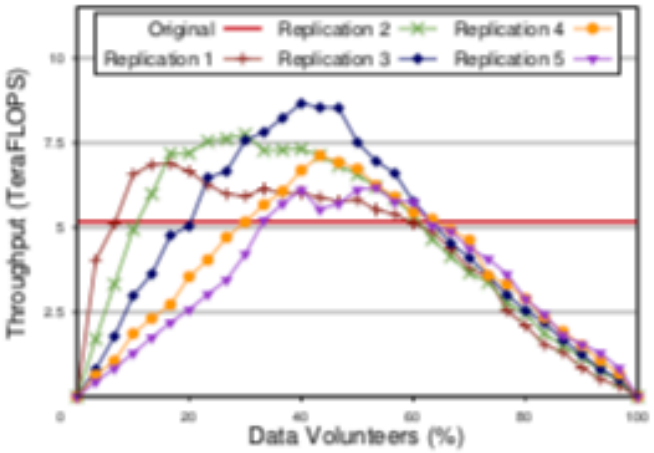
Modelo de BOINC alternativo



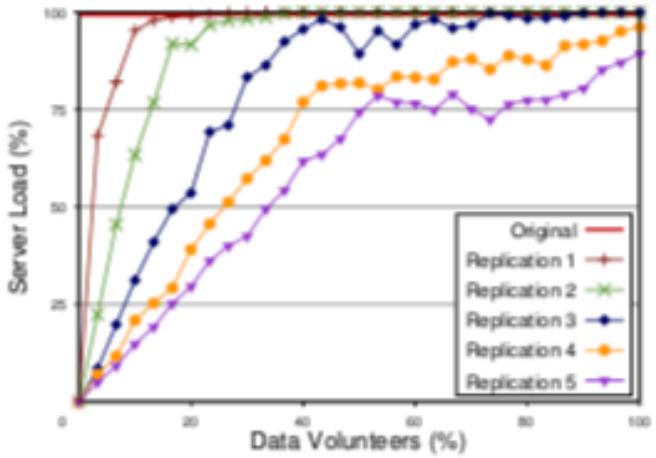
Evaluation: ATLAS@home project

Using volunteer nodes. Case study 1:

- 3,200 active volunteer hosts.



(c) Throughput



(d) Server load

Simulación de sistemas distribuidos de gran escala

Félix García Carballeira
Grupo de Arquitectura de Computadores
Universidad Carlos III de Madrid
felix.garcia@uc3m.es