CLUSIM: Simulador de Clusters para aplicaciones de cómputo de altas prestaciones basado en OMNeT++

C. Marcelo Pérez Ibarra[1], Luis M. Valdiviezo[1], Nilda M. Pérez Otero[1], Héctor P. Liberatori[1], Dolores Rexachs[2], Emilio Luque[2], and Cecilia M. Lasserre[1]

[1] Facultad de Ingeniería - Universidad Nacional de Jujuy (UNJu)
[2] Departamento de Arquitectura de Computadores y Sistemas Operativos
Universidad Autónoma de Barcelona (UAB)
{cmperezi,lmvaldiviezo,classerre,nilperez}@fi.unju.edu.ar
{dolores.rexachs,emilio.luque}@uab.es

Resumen El Grupo de Ingeniería de Software de la Facultad de Ingeniería de la UNJu tiene entre los objetivos de su proyecto de investigación "Definir la configuración adecuada de tolerancia a fallos para diferentes tipos de aplicaciones HPC, teniendo en cuenta los requerimientos de rendimiento y prestaciones del usuario". En este paper se presenta el simulador CLUSIM basado en OMNeT++ que permite simular de forma parametrizable distintas configuraciones de un cluster para aplicaciones HPC. CLUSIM fue pensado como base de un simulador de sistemas de tolerancia a fallos en HPC. En su versión inicial permite simular aspectos de comunicación para aplicaciones tipo Master/Worker. Para llevar a cabo la validación de CLUSIM se consideraron tanto aspectos prestacionales (tiempo de ejecución) como los archivos de trazas generados por el simulador y por las aplicaciones utilizadas en la experimentación. El proceso de validación permitió concluir que, a pesar de las simplificaciones del modelo de simulación, el comportamiento de CLUSIM con otras configuraciones se mantendrá cercano al real.

Keywords: Cómputo de Altas Prestaciones, High Performance Computing, Simulación de HPC, OMNeT++.

1. Introducción

Los sistemas de Cómputo de Altas Prestaciones (High Performance Computing - HPC) se usan para desarrollar software en una gran cantidad de campos, incluyendo física nuclear, simulación de accidentes, procesamiento de datos de satélites, dinámica de ruidos, modelado del clima, bioinformática y modelado financiero. La gran variedad de organizaciones científicas, gubernamentales y comerciales presentes en esta lista ilustra el creciente predominio e impacto de las aplicaciones de HPC en la sociedad moderna [1].
Actualmente el uso de estos entornos se está introduciendo en aplicaciones comerciales como el correo electrónico, servidores web, bases de datos, etc.; lo que

puede suponer el arranque definitivo de los entornos distribuidos de cómputo en la resolución de problemas de diversa índole.
Para estudiar científicamente el comportamiento de sistemas del mundo real a menudo se realizan una serie de suposiciones acerca de cómo trabajan éstos. Estas suposiciones, que usualmente toman la forma de relaciones matemáticas o lógicas, constituyen un modelo que se utiliza para intentar comprender el comportamiento del sistema real.
Si las relaciones que componen el modelo son lo suficientemente simples, es posible usar métodos matemáticos (tales como álgebra, cálculo o teoría de la probabilidad) para obtener una información exacta de las cuestiones de interés; a esto se le llama solución analítica. Sin embargo, la mayoría de los sistemas del mundo real son demasiado complejos y normalmente sus modelos no pueden evaluarse analíticamente. Una posible solución es estudiar dichos sistemas mediante simulación. En una simulación se utiliza la computadora para experimentar numéricamente con un modelo, de forma que con los resultados obtenidos se realice una estimación de las características del sistema.
La simulación, junto con la teoría de modelos y el cómputo masivo, permiten estudiar aplicaciones de la ciencia que hasta el momento tenían un escaso tratamiento como por ejemplo la predicción de terremotos, descubrimiento del nacimiento de galaxias, simulación de reacciones nucleares, avances en genética, etc. Esto se ha conseguido gracias a la utilización de entornos cluster, multicluster y grid como paradigma de resolución de problemas en cómputo de altas prestaciones.
En la actualidad, el uso de modelos de simulación computacional en HPC es cada vez más frecuente, ya sea como ayuda al diseño y modelado de prestaciones [2], como para explorar arquitecturas o aplicaciones [3], [4] o como una herramienta de predicción de tráfico [5].
De igual modo, disponer de un simulador de clusters permitiría analizar el impacto de distintas configuraciones de un sistema paralelo en el rendimiento de aplicaciones de HPC. De este modo, será posible determinar las configuraciones más adecuadas para cada tipo de aplicación (intensivas en cómputo, intensivas en comunicación). Por ello, uno de los objetivos del Grupo de Ingeniería de Software (GIS) de la Facultad de Ingeniería (FI) de la Universidad Nacional de Jujuy (UNJu) es el desarrollo de un simulador que realice las funciones indicadas.
El objetivo principal de este paper es presentar un framework basado en OMNeT++ [6], [7] que permita simular distintas configuraciones de un cluster para aplicaciones de cómputo de altas prestaciones, implementando los módulos de la arquitectura de cómputo de altas prestaciones de forma parametrizable y configurable. Este framework es la base, en un trabajo futuro, de un simulador para el análisis y comprensión de sistemas de tolerancia a fallos en HPC. Es decir, este paper refleja la primera etapa en el desarrollo de un simulador que permitirá:

- la creación y prueba de nuevas políticas en sistemas de altas prestaciones que no están disponibles físicamente;

- el análisis del comportamiento del sistema (desbalanceo de carga, cuellos de botellas causados por fallos) mediante la inyección de diferentes patrones de fallos;
- ayudar en el proceso de toma de decisiones respecto a la inclusión y configuración de sistemas tolerantes a fallos.

El resto de este documento está organizado como sigue. En la sección 2, se comentan algunos de los trabajos relacionados que se estudiaron. En la sección 3 se describe el desarrollo del simulador CLUSIM y en la sección 4 el proceso de experimentación y validación. Finalmente, en la sección 5 se presentan las conclusiones y trabajos futuros.

2. Trabajos Relacionados

En la literatura se encuentra muchos trabajos centrados en simular grandes redes y aplicaciones de HPC. Sin embargo, la mayoría de estos simuladores de redes están centrados en arquitecturas específicas. En [8] se presenta un simulador de Redes de Área de Almacenamiento (SANs) que permite trabajar tanto con trazas de tráfico reales como sintéticas, y simula fallos en enlaces y *switches*, canales virtuales, diferentes algoritmos de ruteo, etc. SIMLAB [9] es otro entorno de simulación para SANs que fue implementado en C++ y permite modelar discos rígidos, nodos de ruteamiento e interfaces de redes. Tiene como objetivo ayudar en el desarrollo y verificación de algoritmos distribuidos para la red PRESTO (SAN que soporta la entrega de datos en tiempo real) ayudando a decidir qué algoritmos son más adecuados y eficientes para ese tipo de red. PARSEC [10] es un entorno de simulación de eventos discretos, que, mediante un compilador mejorado de C++ permite simular entidades y constructores de mensajes de comunicación entre entidades. SIMCAN [11] es un entorno de simulación para grandes redes complejas de almacenamiento que permite simular estas redes y sus subsistemas subyacentes correspondientes (I/O, *Networking*, etc.).
También se encontraron varios trabajos que simulan aplicaciones MPI. Por ejemplo, en [12] se presenta a MPI-SIM, un simulador paralelo diseñado para predecir el rendimiento de aplicaciones MPI y MPI-IO ya existentes. El objetivo de MPI-SIM es predecir el rendimiento de estos programas en función de características de arquitectura tales como número de procesadores, latencias en la comunicación, algoritmos de caché, etc. En [13] se presenta el prototipo de un simulador que consiste en un enfoque híbrido entre la ejecución de una aplicación paralela en modo *stand-alone* y la simulación de una red que se utiliza para el paso de mensajes de MPI. Este prototipo necesita que la aplicación simulada se ejecute en el mismo (o casi el mismo) hardware que se desea simular (por ejemplo, igual CPU a la misma velocidad, igual subsistema de memoria, etc.)
También es posible encontrar entornos de simulación de redes de propósito general que permiten crear diferentes configuraciones de redes, con diferentes tipos de nodos, *switches*, topologías, protocolos, etc. Ejemplos de éstos son OPNET Modeler (http://www.opnet.com/ y OMNeT++ (http://www.omnetpp.org/).

Del análisis de la gran variedad de simuladores hallados, SIMCAN [14] se consideró, en primera instancia, como alternativa para el desarrollo del simulador para aplicaciones de cómputo de altas prestaciones con tolerancia a fallos, debido a su flexibilidad de configuración y a que permite simular el comportamiento de aplicaciones paralelas. Pero debido a que SIMCAM no es de dominio público, se optó por un desarrollo *ad hoc* basado en el framework OMNeT++. De allí surge CLUSIM cuyo desarrollo se describe a continuación.

3. CLUSIM

El estudio del comportamiento de aplicaciones paralelas en clusters de computadoras está limitado no sólo por cuestiones físicas (por ejemplo, complejidad en los cambios de configuración) sino también por cuestiones económicas (por ejemplo, impacto económico asociado a la parada de un cluster). Por ello, un simulador que permita parametrizar la configuración de un cluster resulta una herramienta útil para evaluar y predecir el impacto de diferentes configuraciones en el rendimiento de distintos tipos de aplicaciones paralelas.
El simulador CLUSIM se desarrolló adaptando y combinando módulos del entorno OMNeT++ que permiten configurar los principales componentes de una red de comunicaciones (nodos, protocolos, paso de mensajes, etc). En su primera etapa de desarrollo, CLUSIM sólo simula aspectos de comunicación para aplicaciones tipo Master/Worker sin considerar el tamaño de los mensajes. Para este desarrollo se utilizó la versión 4.0 de OMNET++.

3.1. Entorno de Simulación OMNeT++

OMNeT++ es una herramienta pública de modelado y simulación, basada en componentes modulares y con un entorno de simulación de arquitectura abierta y con un fuerte soporte de GUI, creado por András Varga en el año 2001 en la Universidad Técnica de Budapest. Es un entorno de simulación discreto [6].
Su principal área de aplicación es la simulación de redes de comunicaciones, y debido a que su arquitectura es genérica y flexible, se utilizó exitosamente en el modelado de redes cableadas e inalámbricas, modelado de protocolos, evaluación de aspectos rendimiento en sistemas software complejos, validación de arquitecturas de hardware, etc.
OMNeT++ provee una arquitectura modular basada en componentes (módulos) programados en C++, que se pueden ensamblar para obtener componentes y modelos más complejos utilizando un lenguaje de alto nivel (NED). Aunque OMNeT++ no es un simulador en sí, actualmente está ganando popularidad como una plataforma de simulación de redes tanto en la comunidad científica como en la industria.
OMNeT++ proporciona las herramientas básicas para realizar simulaciones, pero por sí mismo no brinda ningún componente específico para la simulación de redes de computadoras, simulaciones de colas, simulaciones de arquitectura de sistemas o cualquier otra área. Lo que OMNeT++ proporciona es una librería

de clases C++ que permite la creación de componentes de simulación como módulos simples y canales; también, proporciona la infraestructura para reunir las simulaciones de estos componentes y configurarlos en el lenguaje NED o en archivos tipo ini. Además proporciona interfaces en las que se puede observar y manipular el tiempo de ejecución o entornos para la simulación como Tkenv, Cmdenv; herramientas para facilitar la creación de simulaciones y evaluación de resultados.
Para realizar una simulación en OMNeT++ se pueden seguir 6 pasos básicos, que en general, pueden ser aplicados para el modelado.

1. Un modelo de OMNeT++ está formado por componentes que se comunican entre sí intercambiando mensajes. Estos módulos pueden ser anidados, es decir, que varios módulos pueden agruparse todos juntos y formar un módulo compuesto (compound module). Cuando se crea el modelo, se genera un mapa del sistema en una jerarquía de módulos comunicados entre sí.
2. Se define la estructura del modelo en lenguaje NED. Los archivos NED se pueden editar en cualquier editor de texto o en GNED, el editor gráfico de OMNeT++.
3. Los componentes activos del modelo (módulos simples) se programan en C++, utilizando los modelos de simulación y librería tipo class.
4. Se escribe un archivo `omnetpp.ini` para conservar la configuración de OMNeT++ y los parámetros del modelo creado. Un archivo config puede describir diversas simulaciones con parámetros diferentes.
5. Se crea el programa de simulación y se ejecuta. Al hacer esto se crea un vínculo entre el código de OMNeT++ y la interfaces de usuario que proporciona OMNeT++.
6. Los resultados de la simulación se escriben en un vector y un archivo escalar de salida. Se pueden utilizar Plove y Scalars para visualizarlos.

En el siguiente apartado se describe brevemente cómo se siguieron estos pasos en el desarrollo de CLUSIM.

3.2. Desarrollo de CLUSIM

Se creó un archivo de topología `.ned` que consta de un nodo *master* y un arreglo de nodos *workers* conectados entre ellos (*master* y *workers*) por un canal que extiende de la clase `DatarateChannel`, el canal de comunicación genérico de OMNeT++.
A continuación se programó en C++ el comportamiento de los nodos que extienden de la clase `cSimpleModule`, módulo básico simple de OMNeT++ donde se consideraron aspectos tales como tiempos entre envío de mensajes, tiempos de cómputo.
En una tercera etapa se creó un archivo `.ini` para cada una de las configuraciones que se querían simular (ver apartado Validación) asignando valores a los parametros cantidad de nodos, cantidad de bloques, distribuciones de tiempos, etc. Finalmente, se ejecutaron cada una de las simulaciones generando un archivo de trazas para cada una, los que fueron utilizados en el proceso de validación.

4. Experimentación y validación

Para llevar a cabo la experimentación se utilizó el cluster del Laboratorio Universitario de Tecnologías Informáticas (LUTI) de la FI de la UNJu, que cuenta con 10 PC's con las siguientes características (hardware y software):

- Procesador Intel Core 2 Duo
- 2 GB de RAM
- 120 GB de disco duro
- Placa Ethernet 10/100M
- Sistema Operativo: Linux Ubuntu 9.04
- Librería de Paso de Mensajes: MPICH2
- Librerías Adicionales: librería MPE y visualizador Jumpshot.

La librería MPE (MPI Parallel Environment) incluida en MPICH2 permite, entre otras utilidades, recopilar información acerca del comportamiento de programas MPI. La herramienta Jumpshot permite visualizar los archivos de traza generados con la librería MPE.
Las siguientes aplicaciones paralelas de tipo Marter/Worker se utilizaron en el proceso de experimentación:

- Producto de Matrices con asignación estática.
- Producto de Matrices con asignación dinámica.

En el siguiente subapartado se indican los procedimientos seguidos en la experimentación.

4.1. Experimentación

Las aplicaciones paralelas seleccionadas se compilaron con las opciones `-llmpe` y `-lmpe` que permiten enlazar programas MPI con las librerías MPE:

```
mpicc mm-static.c -o estatica -llmpe -lmpe
mpicc mm-dynamic.c -o dinamica -llmpe -lmpe
```

Al compilar los archivos de esta forma se obtiene, tras la ejecución, un archivo `.clog2` que contiene información acerca de los mensajes (tiempos de envío, tiempos de recepción, tamaño de mensajes, etc.) enviados y recibidos por los procesos de la aplicación.
Por un lado, las aplicaciones paralelas se ejecutaron considerando distintas configuraciones del cluster (2, 4 y 8 nodos). Para la aplicación `estatica` se utilizaron matrices de 500x500 elementos, mientras que las dimensiones de las matrices utilizadas en la aplicación `dinamica` fueron 600x600, en bloques de 200x200.
Por otro lado, se escribieron los archivos de configuración del simulador que se corresponden con los escenarios reales de experimentación. Con estos archivos y las cargas reales (obtenidas a partir del análisis de las trazas) se lanzaron las simulaciones en CLUSIM. Análogamente a lo realizado con las aplicaciones reales, se generaron los archivos de trazas (`.elog`) asociados a cada una de las simulaciones.

4.2. Validación

Las trazas generadas por la librería MPE, además de hacer posible la obtención de las cargas reales, se utilizaron para la validación del simulador. Para ello, se utilizaron los 12 archivos de traza obtenidos de las ejecuciones descriptas en el apartado anterior:

Cluster real	Cluster simulado
▪ estatica_2nodos.slog2	▪ estatica_2nodos.elog
▪ estatica_4nodos.slog2	▪ estatica_4nodos.elog
▪ estatica_8nodos.slog2	▪ estatica_8nodos.elog
▪ dinamica_2nodos.slog2	▪ dinamica_2nodos.elog
▪ dinamica_4nodos.slog2	▪ dinamica_4nodos.elog
▪ dinamica_8nodos.slog2	▪ dinamica_8nodos.elog

A fines ilustrativos, la Figura 1 muestra la traza correspondiente al producto de matrices con asignación dinámica ejecutada sobre 8 nodos, y en la Figura 2 se presenta la traza generada al simular la aplicación análoga en CLUSIM. De

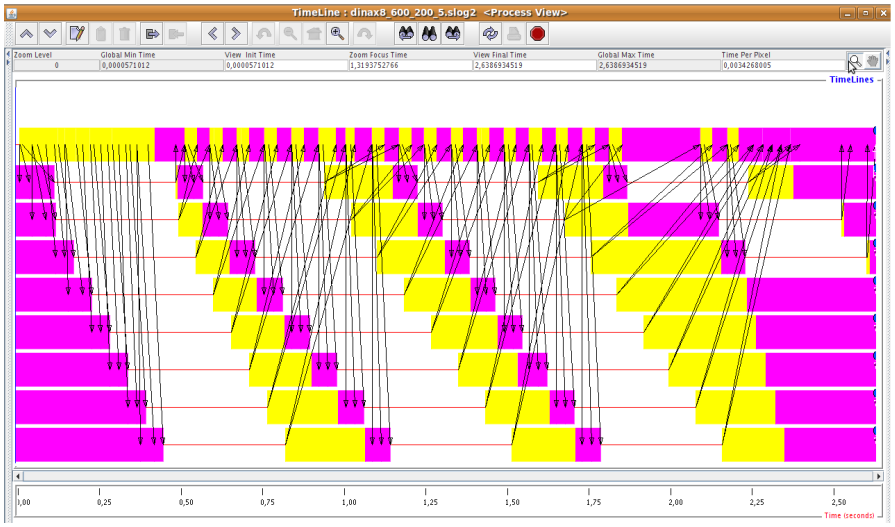

Figura 1. Traza correspondiente al producto de matrices con asignación dinámica ejecutada sobre 8 nodos.

lo que puede observarse en las Figuras 1 y 2 se desprende que la ejecución en el cluster real y en el simulador generan patrones de comunicación similares, esta constatación realizada con diferentes aplicaciones permite validar los aspectos funcionales del simulador.

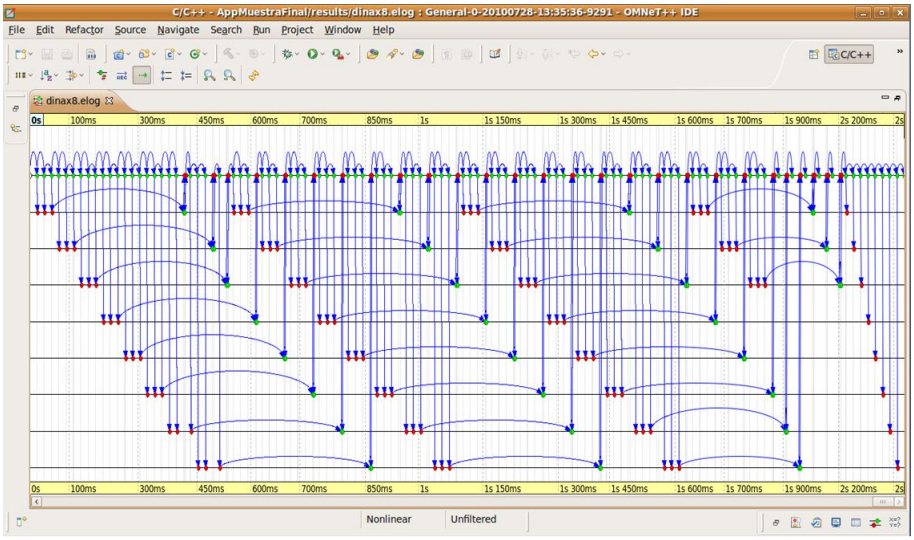

Figura 2. Traza correspondiente al producto de matrices con asignación dinámica sobre 8 nodos simulada en CLUSIM.

Otro aspecto considerado en la validación de CLUSIM fueron los aspectos prestacionales, para ello se constrastó el tiempo de ejecución reportado por las aplicaciones reales y el simulador.
La Figura 3 contrasta los tiempos de ejecución correspondientes al producto de matrices, con asignación estática, ejecutado en 2, 4 y 8 nodos del cluster, con los tiempos de ejecución generados por el simulador. La Figura 4 contrasta los

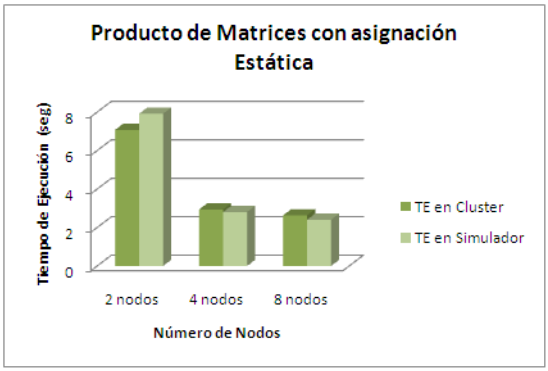

Figura 3. Producto de Matrices con asignación Estática.

tiempos de ejecución correspondientes al producto de matrices, con asignación

dinámica, ejecutado en 2, 4 y 8 nodos del cluster, con los tiempos de ejecución generados por el simulador. Los tiempos de ejecución reales y simulados que se

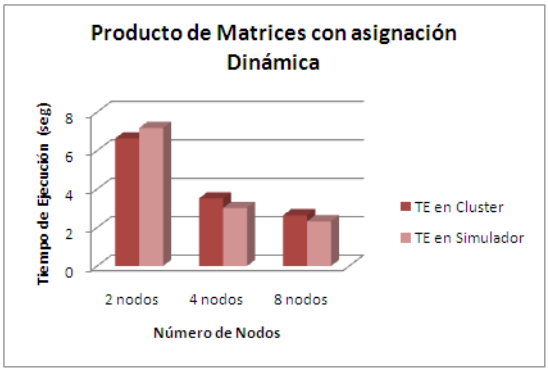

Figura 4. Producto de Matrices con asignación Dinámica.

muestran en las Figuras 3 y 4 son aproximadamente iguales, a pesar de las simplificaciones del modelo de simulación.

5. Conclusiones y Trabajos Futuros

Este artículo presentó el simulador CLUSIM y su proceso de validación. La validación permitió determinar que si bien CLUSIM aún se encuentra en una etapa experimental, es capaz de emular un comportamiento aproximado al de un cluster real.
Como ya se mencionó, la actual versión de CLUSIM realiza la simulación de aplicaciones tipo Master/Worker, sin considerar el tamaño de los mensajes. Por ello, se espera que las futuras versiones de este simulador incluyan más características de la arquitectura física mediante el uso del framework INET (paquete del entorno de simulación OMNeT++ que contiene modelos para varios protocolos: UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF y otros), lo que permitirá obtener resultados más exactos.
Además, considerando los resultados obtenidos hasta ahora de las simulaciones realizadas, puede suponerse que CLUSIM escalará en forma adecuada cuando utilice cargas sintéticas. Esta característica es muy deseable al momento de utilizar el simulador en experimentos de diferentes configuraciones de arquitecturas tolerantes a fallos.

Referencias

1. Jeffrey C. Carver. Third international workshop on software engineering for high performance computing (hpc) applications. In *ICSE COMPANION '07: Compa-*

nion to the proceedings of the 29th International Conference on Software Engineering, page 147, Washington, DC, USA, 2007. IEEE Computer Society.

2. Wolfgang E. Denzel, Jian Li, Peter Walker, and Yuho Jin. A framework for end-to-end simulation of high-performance computing systems. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
3. S. D. Hammond, G. R. Mudalige, J. A. Smith, S. A. Jarvis, J. A. Herdman, and A. Vadgama. Warpp: a toolkit for simulating high-performance parallel scientific codes. In *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
4. Cyriel Minkenberg and Germán Rodriguez. Trace-driven co-simulation of high-performance computing systems using omnet++. In *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
5. Mustafa M. Tikir, Michael A. Laurenzano, Laura Carrington, and Allan Snavely. Psins: An open source event tracer and execution simulator for mpi applications. In *Euro-Par '09: Proceedings of the 15th International Euro-Par Conference on Parallel Processing*, pages 135–148, Berlin, Heidelberg, 2009. Springer-Verlag.
6. András Varga. The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference (ESM'2001)*, June 2001.
7. András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems workshops*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
8. Xavier Molero, Federico Silla, Vicente Santonja, and José; Duato. Modeling and simulation of storage area networks. *Modeling, Analysis, and Simulation of Computer Systems, International Symposium on*, 0:307, 2000.
9. Petra Berenbrink, André Brinkmann, and Christian Scheideler. Simlab - a simulation environment for storage area networks. In *In Workshop on Parallel and Distributed Processing (PDP*, pages 227–234, 2001.
10. Rajive Bagrodia, Mineo Takai, Yu an Chen, Xiang Zeng, and Jay Martin. Parsec: A parallel simulation environment for complex systems. *IEEE Computer*, 31:77–85, 1998.
11. Alberto Nuñez, Javier Fernandez, Jose D. Garcia, Laura Prada, and Jesús Carretero. Simcan: a simulator framework for computer architectures and storage networks. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
12. Rajive Bagrodia, Ewa Deelman, and Thomas Phan. Parallel simulation of large-scale parallel applications. *Int. J. High Perform. Comput. Appl.*, 15(1):3–12, 2001.
13. Rolf Riesen. A hybrid mpi simulator. In *CLUSTER*, 2006.
14. Alberto Núñez, Javier Fernández, Jose D. Garcia, Félix Garcia, and Jesús Carretero. New techniques for simulating high performance mpi applications on large storage networks. *J. Supercomput.*, 51(1):40–57, 2010.