Hacia un Modelo Genérico de Aplicaciones Paralelas

Cecilia M. Lasserre[1], Adelina García[1], Nilda M. Pérez Otero[1], Abigail R. N. Verazay[1], Marcelo Pérez Ibarra[1], Silvia A. Nolasco[1], Jorge G. Martínez[1]

[1] Grupo de Ingeniería de Software, Facultad de Ingeniería, Universidad Nacional de Jujuy, 4600, Jujuy, Argentina
lasserre@arnet.com.ar, adelinagarcia_jujuy@hotmail.com, nilperez@gmail.com, abigailrn@gmail.com, cmperezi@gmail.com, sil_ale107@hotmail.com, jorgegerman@live.com.ar

Abstract. En este trabajo se presenta el proceso de reformulación del modelo preliminar plasmado en CluSim. Estudios realizados sobre un conjunto de aplicaciones master/worker condujeron a la caracterización de los patrones de cómputo y comunicación asociados a programas paralelos. Para la contrastación del nuevo modelo se utilizó una aplicación distinta del conjunto usado para la caracterización. Los satisfactorios resultados obtenidos, permiten concluir que el proceso de modelado utilizado es adecuado generar a posteriori un modelo genérico de aplicaciones paralelas.

Keywords: master/worker, modelado de aplicaciones, simulación, CluSim.

1 Introducción

Durante los últimos años, el Grupo de Investigación de Ingeniería de Software (GIS) de la Facultad de Ingeniería (FI) de la Universidad Nacional de Jujuy (UNJu), Argentina, se ha centrado en el estudio de aplicaciones del Cómputo de Altas Prestaciones (HPC), particularmente, en el análisis del rendimiento de aplicaciones paralelas que se ejecutan en clusters de computadoras. Hasta el momento, el GIS desarrolló un modelo preliminar basado en una arquitectura cluster, que se plasmó en CluSim (Cluster Simulator) un simulador de clusters para aplicaciones paralelas [1].

CluSim es un framework basado en OMNeT++ (plataforma abierta para el desarrollo de simuladores [2]) que permite simular distintas configuraciones de un cluster para aplicaciones HPC, implementando los módulos de la arquitectura de Cómputo de Altas Prestaciones de forma parametrizable y configurable [1]. De este modo, CluSim permitiría analizar el impacto de las distintas configuraciones de un sistema paralelo en el rendimiento de aplicaciones HPC, haciendo posible determinar qué configuraciones resultan más adecuadas para cada tipo de aplicación.

Las primeras versiones de CluSim sólo emulaban los patrones de comunicación, sin considerar el tamaño de paquetes, asociados a una aplicación de producto de matrices implementada según el paradigma master/worker [3]. Versiones más recientes de CluSim incorporan el framework INET (basado también en OMNeT++) que simula las características físicas de la red de interconexión de un cluster [4].

Entre las líneas de investigación propuestas en trabajos anteriores, se estableció extender la funcionalidad inicial de CluSim para desarrollar un entorno de simulación robusto y flexible que se adapte fácilmente a los paradigmas de programación paralela pipeline, divide/conquer y SPMD. Siguiendo esta línea, el GIS dirigió sus esfuerzos a la generalización del modelo original de CluSim a fin que éste contemplara los distintos tipos de aplicación paralela. Con esto en mente, se seleccionaron sendos conjuntos de aplicaciones correspondientes a los paradigmas master/worker, SPMD, pipeline y divide/Conquer. En este trabajo se presenta el estudio realizado sobre aplicaciones master/worker y la caracterización obtenida como resultado de este estudio.

El resto del trabajo se estructura de la siguiente forma: en el apartado 2 se presentan algunos modelos de análisis de rendimiento basados en el comportamiento de aplicaciones paralelas, en el apartado 3 se describe el proceso de modelado aplicado en este trabajo y el modelo de aplicaciones propuesto, en el apartado 4 se expone la contrastación del modelo y una aplicación de prueba, y se discuten los resultados obtenidos. Finalmente en el apartado 5 se presentan las conclusiones del trabajo y posibles líneas futuras. Por último, en el apartado 6 se indican las referencias utilizadas.

2 Antecedentes

El análisis de rendimiento permite evaluar y predecir el coste computacional de la ejecución de aplicaciones paralelas en entornos paralelos. Este análisis permite determinar la forma correcta de funcionamiento de una aplicación en un sistema particular, de modo que se consiga la mayor productividad posible.

Una de las principales herramientas del análisis de rendimiento la constituyen los modelos. Un modelo es una abstracción que permite obtener una descripción concisa de un sistema, centrándose en los aspectos más relevantes del problema estudiado [5].

Si bien, no existe una clasificación generalizada, los modelos de rendimiento de sistemas paralelos pueden dividirse en modelos arquitecturales y modelos de aplicaciones. Los modelos arquitecturales describen de forma genérica el comportamiento de un sistema, en tanto que, los modelos de aplicaciones describen el comportamiento de las aplicaciones paralelas al ejecutarse en un determinado sistema.

En la literatura es posible encontrar varios modelos orientados a la evaluación del rendimiento que analizan el comportamiento de aplicaciones paralelas. Considerando esto, a continuación se describen algunos modelos y herramientas cuyos fundamentos se aplicaron en el desarrollo del presente trabajo.

El entorno TIA (Tools for Instrumentation and Analysis) es una herramienta de análisis de rendimiento en sistemas paralelos, que define una metodología basada en el análisis estadístico de los datos generados mediante una instrumentación ligera del código fuente de una aplicación paralela. TIA proporciona un mecanismo de modelado automático que permite enfocar los esfuerzos del usuario en el diseño experimental y en el análisis de los resultados. En particular, esta metodología requiere que el usuario elija adecuadamente el conjunto de métricas y parámetros de rendimiento que deben ser considerados en la instrumentación y en el proceso de

modelado. Tanto la instrumentación adecuada del código como la obtención del modelo de rendimiento son procesos automáticos, y el resultado final es un modelo de rendimiento en función de las métricas y de los parámetros considerados inicialmente [5].

En [6] se presenta el desarrollo de una herramienta de análisis de prestaciones para aplicaciones paralelas de paso de mensajes que permite extraer conocimiento sobre la estructura de los programas, aún sin acceso a los fuentes ni a los algoritmos correspondientes. Esta herramienta se divide en dos componentes: un módulo que funciona al tiempo de ejecución de las aplicaciones, obteniendo trazas según requerimientos del usuario, y una aplicación del módulo anterior que analiza las trazas recogidas, obteniendo conocimiento sobre la naturaleza algorítmica de las aplicaciones ejecutadas y sobre los recursos utilizados. Con este conocimiento la herramienta permite realizar predicciones bajo condiciones diferentes a las de ejecución original.

El modelo TTIGHA [7] permite representar aplicaciones paralelas que se ejecutan sobre arquitecturas heterogéneas, considerando que la heterogeneidad está dada por procesadores y la red de comunicación. El modelo permite representar las aplicaciones a través de grafos que los algoritmos de mapping utilizan para asignar las diferentes tareas que componen una aplicación, teniendo en cuenta sus características, a cada uno de los procesadores de la arquitectura.

3 Modelado de Aplicaciones

Como se mencionó, existen diversos modelos que permiten representar el comportamiento de aplicaciones paralelas a fin de estudiar distintos aspectos de rendimiento de los sistemas paralelos.

Para formular modelos de aplicación, lo adecuado sería experimentar con el sistema real midiendo sus características mediante herramientas específicas. Las técnicas de análisis de rendimiento permiten registrar y determinar los valores correspondientes a métricas (por ejemplo, tiempo de ejecución) o parámetros del sistema (por ejemplo, número de procesadores) que se utilizan para evaluar el rendimiento. En virtud de ello, las técnicas de análisis de rendimiento pueden clasificarse en función del momento en el que se registran las mediciones o sucesos (métodos de muestreo o métodos accionados por eventos) o en función de cómo se registran dichas mediciones o sucesos (perfiles o trazas).

Estas técnicas se usan normalmente para extraer los parámetros que luego se utilizarán en los modelos analíticos, en las simulaciones, para validar un modelo de rendimiento determinado o para ser utilizado en módulos de predicción [8].

La instrumentación de la aplicación, técnica correspondiente al método accionado por eventos, consiste en insertar instrucciones en el código a fin de registrar el comportamiento dinámico durante la ejecución. De acuerdo a la instrumentación aplicada, ésta puede ser de código fuente, de traductores código a código, de interposición de librerías o de forma dinámica [5].

Las trazas obtenidas mediante la instrumentación del código permiten estudiar el comportamiento dinámico de la aplicación y determinar las propiedades algorítmicas subyacentes a la sucesión de eventos de la aplicación, con lo que se consigue [6]:

- Obtener información acerca de la distribución de datos y los patrones de comunicación de la aplicación.
- Identificar las estrategias de resolución de problemas utilizada por la aplicación.
- Identificar problemas en el balance de carga, de modo que se pueda sugerir mapeos alternativos o reubicación dinámica de procesos.
- Ayudar a predecir prestaciones en sistemas propuestos, reales o hipotéticos.

La Fig. 1 ilustra el proceso seguido en este trabajo para la formulación de modelos de aplicaciones.

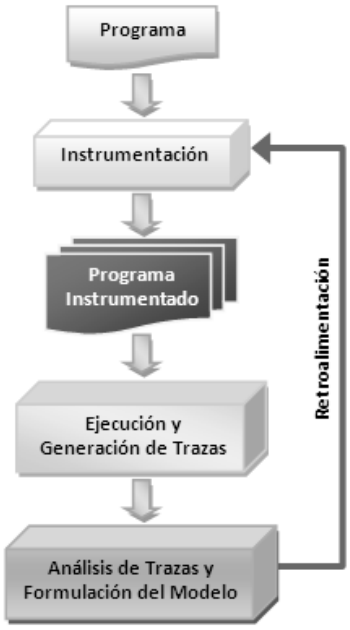


Fig. 1. Proceso de modelado.

A partir del modelo preliminar plasmado en CluSim, este trabajo se enfoca en la reformulación de un modelo de aplicación que describe los patrones de cómputo y comunicación asociados a programas implementados con el paradigma master/worker. Para la reformulación se aplicaron:

- el método accionado por eventos para el registro de los sucesos del programa,
- la instrumentación por interposición de librerías, utilizando la librería MPE de MPICH y

- la generación de trazas de ejecución para la obtención de mapas del comportamiento dinámico de las aplicaciones analizadas.

3.1 Reformulación del Modelo

En la versión original de CluSim se definieron los siguientes parámetros para el modelado del comportamiento del producto de matrices [3]:

1) **t_ini**: tiempo que transcurre entre el *send* de aviso de inicio de tarea y *send* del bloque de la primera matriz.
2) **t_matrices**: tiempo que transcurre entre el *send* del bloque de la primera matriz y el *send* del bloque de la segunda matriz.
3) **t_nodos**: tiempo que transcurre entre el envío de tareas a los *workers*.
4) **t_computo**: tiempo de cómputo de la tarea.
5) **t_aviso**: tiempo que transcurre entre *send* de aviso y *send* de resultados del *worker[i]* al *master*.
6) **t_trabajo**: tiempo que transcurre entre *recv* de resultados del *worker* y *send* de una nueva tarea.

La Fig. 2 ilustra un segmento de traza de la aplicación objetivo en la que se indican los tiempos considerados en el análisis.

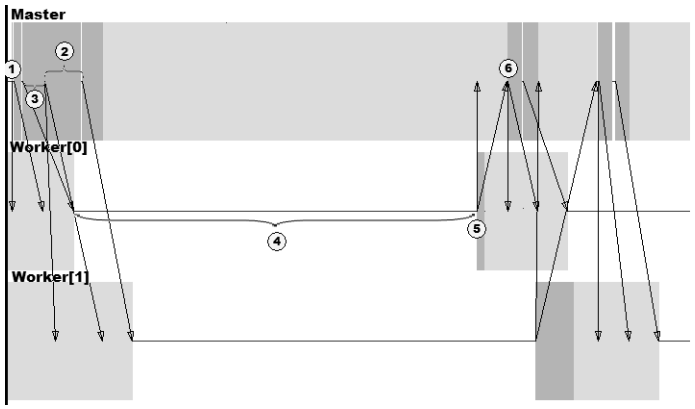


Fig. 2. Segmento de una traza de la aplicación producto de matrices.

Si bien los experimentos iniciales mostraron que este modelo reproducía satisfactoriamente el comportamiento de la aplicación en un cluster homogéneo [1, 3], e incluso en uno heterogéneo [9], experimentos más recientes reflejaron ciertas limitaciones al modelar el comportamiento de otras aplicaciones master/worker.

Esto evidenció el carácter específico del modelo, mostrando la necesidad de ajustarlo para otorgarle un mayor nivel de abstracción, contemplando sólo los aspectos más significativos del comportamiento de aplicaciones paralelas.

Con el objetivo de alcanzar un modelo más genérico, se utilizaron sobre un grupo de aplicaciones sencillas de tipo master/worker las técnicas descriptas al inicio de este apartado. Así, se obtuvo un nuevo conjunto de parámetros consiguiendo un mayor nivel de abstracción.

El nuevo modelo contempla los siguientes tiempos de cómputo y comunicación (parámetros):

1) **t_computo_inicial_master**: tiempo que transcurre entre el inicio de la aplicación y el primer envío de trabajo a los *workers*.
2) **t_computo_final_master**: tiempo que transcurre entre la última recepción de trabajo completado, hasta la finalización de la aplicación.
3) **t_nodos**: tiempo que transcurre entre el envío de tareas a los *workers*.
4) **t_computo_nodo**: tiempo de cómputo de la tarea.
5) **t_envio_trabajo**: tiempo de transferencia del trabajo del *master* a un *worker*.
6) **t_envio_resultado**: tiempo de transferencia del resultado de la tarea asignada de un *worker* al *master*.

La Fig. 3 presenta un segmento de traza en el que se identifican los tiempos considerados en el modelo reformulado.

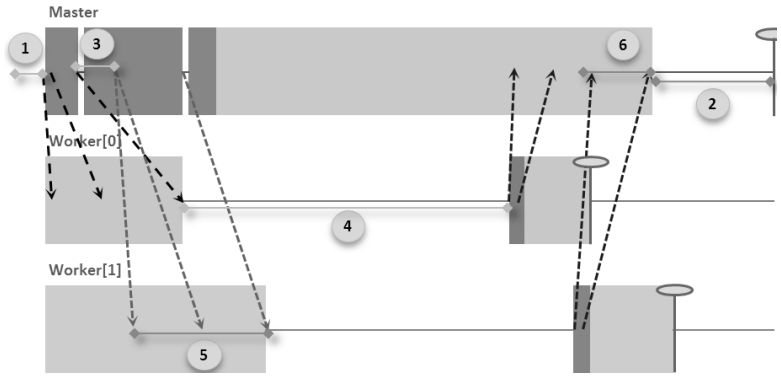


Fig. 3: Segmento de una traza de la aplicación producto de matrices para el nuevo modelo.

Definidos los parámetros del modelo, es preciso identificar su comportamiento. A continuación se presentan los experimentos realizados a tal fin.

3.2 Identificación de las distribuciones de los parámetros

Para 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 10.04
- Librería de Paso de Mensajes: MPICH2
- Librerías Adicionales: librería MPE y visualizador Jumpshot.

Las siguientes aplicaciones, tipo master/worker, se utilizaron en el proceso de experimentación:

- Suma de Arreglos.
- Producto de Matrices con asignación dinámica.

La Tabla 1 muestra los escenarios de experimentación utilizados.

Tabla 1. Escenarios de experimentación.

Aplicación	Características	Cluster (N° de workers)
Suma de arreglos	500000 elementos	2, 4 y 8
Producto de matrices con asignación dinámica	matriz 1600x1600 en bloques 100	2, 4 y 8
	matriz 1600x1600 en bloques 200	2, 4 y 8
	matriz 1600x1600 en bloques 400	2, 4 y 8

Por un lado, como resultado del análisis de la representación gráfica (utilizando la herramienta *Jumpshot*) de las trazas de ejecución, se identificaron los patrones de comportamiento de estas aplicaciones, a partir de los que se definieron los parámetros del modelo (indicados en la sección 3.1).

Por otro lado, el análisis del detalle de tiempos asociados a los eventos de envío y recepción de mensajes (contenido en las trazas) permitió establecer un conjunto de distribuciones de probabilidad que podrían representar el comportamiento de los parámetros identificados. Así, para seleccionar la distribución de cada parámetro del modelo, se tomó aquella que, con un significativo grado de ajuste, era común a los resultados generados por ambas aplicaciones [10, 11].

La Tabla 2 presenta los parámetros del nuevo modelo y las distribuciones de probabilidad asociadas.

Tabla 2. Parámetros y distribuciones del nuevo modelo.

Parámetro	Distribución de Probabilidad
t_computo_inicial_master	No identificada
t_computo_final_master	No identificada
t_nodos	Lognormal
t_computo_nodo	Lognormal
t_envio_trabajo	Weibull
t_envio_resultado	Lognormal

Los parámetros t_computo_inicial_master y t_computo_final_master no pudieron ser asociados a alguna distribución de probabilidad, ya que los datos obtenidos como resultado de los experimentos ejecutados fueron insuficientes para proporcionar información significativa sobre ellos. Por ello, se prevé realizar más experimentaciones que permitan determinar apropiadamente las distribuciones correspondientes a los parámetros previamente indicados.

En el siguiente apartado se presenta la contrastación del modelo con el análisis de una aplicación master/worker a fin de validar las hipótesis planteadas.

4 Contrastación del modelo

Para contrastar el modelo se utilizó una aplicación de búsqueda y ordenación de arreglos implementada según el paradigma master/worker. Esta aplicación se ejecutó considerando los mismos escenarios de experimentación que se utilizaron para definir los parámetros del modelo y las distribuciones asociadas.

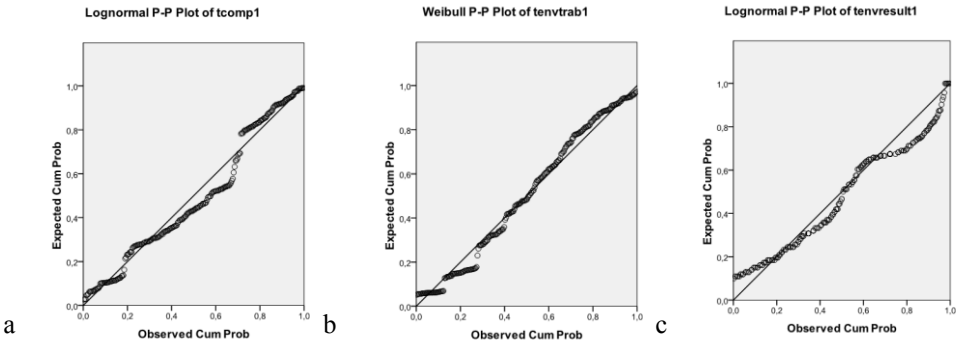
Para analizar las características de la aplicación utilizada en la validación, se aplicó nuevamente la instrumentación del código (por interposición de librerías) y se generaron las trazas de ejecución correspondientes para procesarlas y determinar el patrón de sucesión de eventos (envío y recepción de mensajes).

En la contrastación se verificó que la distribución propuesta por el modelo para cada parámetro estuviera entre las distribuciones de mejor ajuste. La Tabla 3 resume los resultados obtenidos

Tabla 3. Distribución de probabilidad de los tiempos de la aplicación de prueba.

Tiempos	Distribución de Probabilidad
t_computo_inicial_master	No identificada
t_computo_final_master	No identificada
t_nodos	Lognormal
t_computo_nodo	Lognormal
t_envio_trabajo	Weibull
t_envio_resultado	Lognormal

Al comparar los resultados presentados en las Tablas 2 y 3 puede observarse que los tiempos del modelo y de la aplicación de prueba siguen las mismas distribuciones de probabilidad. Las Fig. 4.a, 4.b y 4.c permiten apreciar el ajuste de los parámetros de la aplicación de búsqueda y ordenación a las distribuciones propuestas en el modelo.



a b c

Fig. 4: (a) Distribución de t_computo_nodo, (b) distribución de t_envio_trabajo, (c) distribución de t_envio_resultado.

5 Conclusiones y Trabajos Futuros

En vista de los resultados obtenidos puede concluirse que:

- Para lograr un modelo más abstracto, los parámetros del modelo están definidos según los tiempos de comunicación y de cómputo.
- Se evidenció la necesidad de utilizar *benchmarks* y aplicaciones de cómputo intensivo para poder calibrar con precisión los parámetros del modelo.
- Es preciso estudiar la relación entre los factores variables del sistema paralelo y los parámetros del modelo.
- Si bien el modelo intenta comprender un espectro amplio de aplicaciones, alcanzar un mayor nivel de abstracción tiene como costo una menor precisión en la representación del sistema real.

Respecto a las líneas de investigación abiertas:

- Entendiendo que la metodología aplicada en la reformulación del modelo permitió obtener resultados positivos, se prevé continuar la extensión del modelo para contemplar los paradigmas SPMD, pipeline y divide/conquer.
- Integrar a CluSim el modelo reformulado y validarlo utilizando *benchmarks*.
- Formular un modelo arquitectural para CluSim que permita independizar las características de la máquina paralela, de las de la aplicación que se simule.
- Desarrollar una herramienta que permita analizar la traza de una aplicación y generar la entrada (parámetros y distribuciones) adecuada para CluSim.

6 Referencias

1. Pérez Ibarra, C. M., Valdiviezo, L. M., Pérez Otero, N. M., Liberatori, H. P., Rexachs, D, Luque, E., Lasserre, C. M.: CLUSIM: Simulador de Clusters para Aplicaciones de Cómputo de Altas prestaciones basado en OMNeT++. XVI Congreso Argentino de Ciencias de la Computación. Morón, Buenos Aires (2010)
2. Varga, A.: The OMNeT++ Discrete Event Simulation System. Proceedings of the European Simulation Multiconference (ESM'2001). (2001)
3. Valdivezo, L. M., Pérez Otero, N. M., Pérez Ibarra, C. M., Lasserre, C. M.: Caracterización de una aplicación paralela con distintas configuraciones en CluSim. Investigaciones en Facultades de Ingeniería del NOA – 2010. ISSN 3367-5072. Ed. EdiUNJu. S. S. de Jujuy, Jujuy. pp. 499-504. (2010)
4. García, A., Pérez Otero, N. M., Pérez Ibarra, C. M., Lasserre, C. M.: Simulación de Clusters: Integración de INET a CluSim. XVII Congreso Argentino de Ciencias de la Computación. La Plata, Buenos Aires. (2011).
5. Rodríguez Martínez, D.: Modelado Analítico del Rendimiento de Aplicaciones en Sistemas Paralelos. Tesis Doctoral. Departamento de Electrónica e Computación. Universidade de Santiago de Compostela. (2011)
6. Grosclaude, E. Caracterización de Aplicaciones de Paso de Mensajes con Técnicas de Caja Negra. Trabajo Final de Especialidad de Cómputo de Altas Prestaciones. Universidad Nacional de La Plata. Buenos Aires. (2012)
7. De Giusti, L. Mapping sobre Arquitecturas Heterogéneas. Tesis Doctoral. Universidad Nacional de La Plata. Buenos Aires. (2008)
8. Blanco, V.: Análisis, predicción y visualización del rendimiento de métodos iterativos en HPF y MPI. Tesis Doctoral. Departamento de Electrónica e Computación. Universidade de Santiago de Compostela. (2002).
9. Lasserre, C. M., Pérez Ibarra, C. M., Valdiviezo, L. M., Verazay, A. R. N., Quispe, G. L., Nolasco, S. A., Chosco, Víctor H., Pérez Otero, N. M.: Adaptación de CluSim a cluster heterogéneos. ISSN 1853-7871. Editorial Científica Universitaria. Catamarca, Argentina. (2011).
10. Devore, J. L.: Probabilidad y Estadística para Ingeniería y Ciencias (6° edición). (Trad. Sánchez Fragoso, F.). México D.F.: México, International Thomson Editores. (2005)
11. Walpole, R. E.: Probabilidad y Estadística para Ingenieros (6° edición). (Trad. Cruz, R.). México: Prentice-Hall Hispanoamericana. (1999)