COMPRENSIÓN DE SISTEMAS PARALELOS

Norma Beatriz Perez[1], Mario M. Berón[1], Pedro R. Henriques[2] y Maria J. Pereira[3]

[1]Departamento de Informática / Facultad de Ciencias Físico Matemáticas y Naturales / Universidad Nacional de San Luis (UNSL), Argentina

Ejercito de los Andes 950, D5700HHW San Luis, +54-0266 4520300 - Int. 2102
{nbperez, mberon}@unsl.edu.ar

[2]Departamento de Informática / Escola de Engenharia / Universidade de Minho Braga, Portugal

Campus de Gualtar, 4710-057 BRAGA, +351-253 604470, Fax: +351-253604471
pedrorangelhenriques@gmail.com

[3]Departamento de Informática / Instituto Politécnico de Bragança, Portugal

Campus de Santa Apolónia, 5300-253 BRAGANÇA, +351-273 303200, Fax: +351 273325 405
mjoao@ipb.pt

Resumen

Las Tecnologías de la Información y la Comunicación (TIC's) ofrecen servicios sorprendentes, que proporcionan acceso permanente a todo tipo de información y aplicaciones que se encuentran alrededor del mundo. En particular, los *sistemas paralelos* son buenos ejemplos de los paradigmas emergentes que deben ser considerados por científicos que se desempeñan en el campo informático. Estos sistemas plantean actividades críticas (sincronización, comunicación, etc.) que repercuten en la eficiencia e utilidades de ellos. Los sistemas con las características mencionadas también están sujetos a tareas de mantenimiento, evolución y migración de software. Para realizar estas actividades los programadores deben comprender el sistema (o programa). Un programador entiende un programa cuando consigue relacionar el dominio del problema con el dominio del programa. Llevar a cabo esta relación no es una tarea sencilla porque implica la construcción de una representación adecuada para cada dominio y la definición de un procedimiento de vinculación de ambas representaciones. Por esta razón, es necesario la elaboración de métodos, técnicas y herramientas que faciliten la comprensión de estos sistemas de forma tal que se puedan reducir los costos y esfuerzos en las tareas descriptas. Lo antes mencionado es la principal línea de investigación que se presenta en este artículo.

Palabras clave: TIC's, sistemas paralelos, comprensión de programa, dominio del problema, dominio del programa

Contexto

La línea de investigación descripta en este artículo, se encuentra enmarcada en el contexto del proyecto de investigación: "*Ingeniería del Software: Aspectos de Alta Complejidad Sensibilidad en el Ejercicio de la Profesión de Ingeniero de Software*", que se desarrolla en la UNSL. Este proyecto es reconocido por el programa de incentivos y es la continuación de diversos proyectos de investigación de gran éxito a nivel nacional e internacional.

1. Introducción

En los últimos años, el mundo ha sido testigo de la revolución en la industria del software y hardware. La fuerza motriz que conduce estos cambios se debe a la necesidad de los usuarios de disponibilizar la información solicitada en el menor tiempo posible. Para esto, se estudian diversas técnicas que son propuestas para mejorar el acceso a datos, diseñar y/o rediseñar métodos que aprovechan las ventajas del software y hardware de hoy.

Los avances de los sistemas paralelos, posiblemente distribuidos en todo el mundo, se reflejan en: (i) La industria, siendo su foco competir en el mercado produciendo mejores productos; (ii) En el campo científico como ser: el procesamiento multimedial (manipulación, correción de video y/o sonido), área de la medicina (tratamiento de drogas, vacunas, exploración del ADN humano, análisis de proteínas); (iii) En el campo de la meteorología y climatología (predicción de desastres naturales inundaciones, terremotos, tsunamis). Son ejemplos de aplicaciones que están en permanente evolución debido a los cambios en los requerimientos del sistema en cuestión; cambios en la tecnología que permiten mayor poder cómputo (respuestas en tiempos aceptables frente a consultas solicitadas por usuarios); manipulación eficiente en estructuras de datos a fin de mejorar la utilización de la arquitectura, etc. Los cambios sociales y económicos que se deben a los avances tecnológicos son muy amplios dado que la información es accesible desde cualquier lugar y en cualquier momento. Esta ubicuidad de la información ha ayudado a reducir la brecha entre los tiempos de respuesta y la complejidad de manipular sistemas paralelos. Gracias a la variedad de métodos existentes que surgen continuamente, hacen que el mundo se transforme, y una vez más, los usuarios de TIC's puedan aprovechar las ventajas de los servicios provistos de manera eficaz e eficiente. Los tipos de sistemas descriptos previamente, como cualquier otro software, están sujetos a actividades de mantenimiento, evolución y migración.

Lo descripto en los párrafos anteriores revela algunos de los puntos más importantes que motivan este artículo. Las investigaciones realizadas se describen de manera sucinta en las siguientes subsecciones.

1. 1 Sistemas Paralelos

El código de estos sistemas puede estar distribuido alrededor del mundo en diferentes procesadores (no necesariamente de características idénticas). Este tipo de sistemas son el foco de esta investigación.

En esencia los sistemas paralelos tienen un modelo [1, 2] y/o infraestructura [3, 4] asociado que soporta el paralelismo. Es importante destacar que la elección del modelo y/o infraestructura la hace el programador, siendo una actividad crítica dado que no hay un modelo y/o infraestructura que funcione en todos los escenarios paralelos. Esto ha sido una de las motivaciones para el desarrollo de un conjunto amplio y diverso de modelos y/o infraestructuras en el campo del paralelismo. Ellos varian en complejidad siendo algunos adaptativos a arquitecturas específicas de hardware. Esto evidencia, que si el programador no realiza una correcta elección del modelo y/o infraestrura, se obtienen algoritmos paralelos poco eficientes, complejos de entender, una incorrecta sincronización y/o comunicación, códigos replicados, etc.

Por otro lado, como el mercado del software y hardware se encuentra en constante evolución debido a los requerimientos de los usuarios es necesario realizar en estos sistemas tareas de mantenimiento,

evolución y migración. Los cambios del software se deben a mundanzas en el:

Hardware: cuando surge la necesidad de migrar a una arquitectura que ofrezca mejores prestaciones que la actual sobre la que corre el sistema paralelo.

Software: cuando se desee realizar tareas de mantenimiento y/o migrar a otro modelo y/o infraestructura. En este caso no es necesario que el programador sea un experto en el modelo y/o infraestructura asociado. Para que esto, sea posible el programador debe disponer de métodos que le permitan comprender el código rápidamente e identificar el modelo y/o infraestructura con que fue diseñado el algoritmo paralelo. Evitando de esta manera costos elevados en recursos humanos y minimizando el factor de tiempo. Para desarrollar e implementar estos métodos se propone emplear la compresión de programas (CP).

1.2. Comprensión de Programas

La CP es una disciplina de la Ingeniería de Software cuyo objetivo es proveer métodos, técnicas y herramientas para facilitar el estudio y entendimiento de programas. La CP se basa en un proceso cognitivo y de ingeniería, cuyo objetivo es facilitar el entendimiento de los sistemas, en particular los sistemas paralelos.

Proceso Cognitivo [5]: se basa en estructuras de la información, estrategias de estudio y análisis de etapas seguidos por los programadores para comprender programas [6]. Los modelos cognitivos (MC) se dividen en componentes. Ellas son:

Conocimiento: (i) Interno: compuesto por el conjunto de conceptos y relaciones que conforman la estructura de conocimiento del programador; y (ii) Externo: cuyos componentes son los nuevos conceptos proporcionados por el sistema en estudio.

Modelo Mental: representación mental que tiene el programador del sistema.

Proceso de Asimilación: describe la estrategia (top-down, bottom-up o híbrida) empleada por el programador para entender programas.

Entender los MC es una tarea relevante porque ellos en cierta manera como un programa comprenden un problema. Son muchos los autores que sostienen que: *un programador entiende un programa cuando él puede localizar las componentes de software utilizadas para producir la salida del sistema* [7, 8]. Es decir, cuando es posible *relacionar el dominio del problema y el dominio del programa*. Para lograr este objetivo se debe:

- Proveer representaciones para los dominios del problema y programa.
- Definir un proceso que permita unir ambas representaciones.

Proceso de Ingeniería: teniendo presente el resultado mencionado por los MC, este proceso debería abordar:

Extracción de la Información: con el objetivo de implementar métodos, técnicas y herramientas basadas en modelos cognitivos y visualización de software se extrae: (i) Información estática (empleando técnicas tradicionales de compilación) y (ii) Información dinámica (empleando: un esquema de instrumentación de código, debugging, profile u otro) del sistema paralelo en estudio. Este artículo, tiene por objetivo proponer una elaboración gradual de las investigaciones descriptas, dividiendo las tareas en etapas, donde en cada etapa se añaden funcionalidades extras que permiten comprender el sistema paralelo. La Figura 1, describe gráficamente esta situación; donde en la primera etapa: un programa paralelo (que es el objeto de estudio) se le aplica un *proceso de matching* que permite identificar el modelo y/o infraestructura subyacente, luego en la segunda etapa: se utiliza la información recopilada del programa (identificación del modelo y/o infraestructura aso-

ciado al sistema en estudio) se elaboran estrategias de comprensión que relaciona el domino del problema con el dominio del programa.

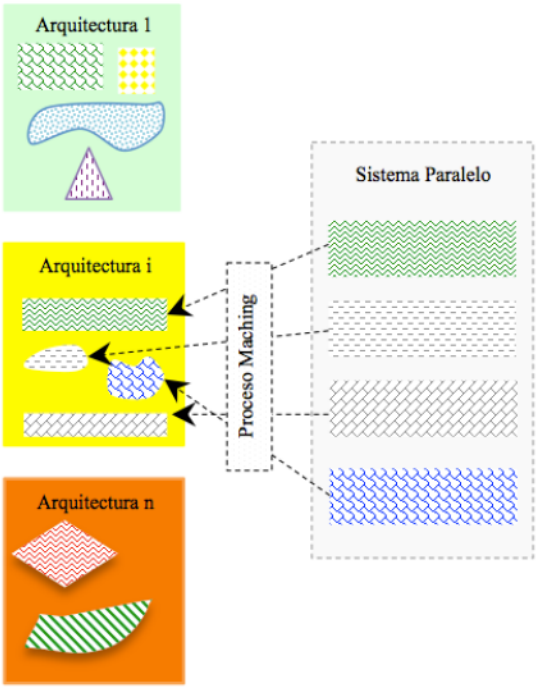

Figura 1: Representación de mapeo entre un programa paralelo y diferentes arquitecturas

Estrategias de Interconexión de Dominios: Una de las formas de facilitar la comprensión de sistemas consiste en relacionar el dominio del problema (es decir el comportamiento del programa) con el dominio del programa (la operación del programa). Esta relación permite que el programador pueda identificar rápidamente componentes de interés del programa sobre los cuales debe concentrar su estudio. Es importante mencionar que los costos involucrados en el mantenimiento, evolución y migración del sistema disminuyen debido a que se reduce el tiempo que necesita el programador para la realización de estas actividades. Esta característica es de relevancia en el contexto de la CP debido a la posibilidad que se presenta para realizar contribuciones ya que son casos excepcionales las herramientas de comprensión que incluyen este tipo de relación.

Visualización de Software: tiene como objetivo mapear ciertos aspectos de software en una o más representaciones multimediales [9, 10, 11]. El desafío principal, cuando la visualización del software esta orientada a la CP, consiste en construir vistas que permitan relacionar el dominio del problema con el dominio del programa. Existe un amplio conjunto de herramientas de visualización de software. Sin embargo, un amplio rango de ellas proponen visualizaciones del dominio del problema (funciones, variables, módulos, etc.) sin considerar dos componentes importantes que son el dominio del problema y su relación con el dominio del programa. Esto deja en evidencia la necesidad de nuevos métodos, técnicas y herramientas que permitan subsanar el problema antes mencionado y que son objeto de esta investigación.

2. Líneas de Investigación, Desarrollo e Innovación

La propuesta de este artículo surge a partir de la experiencia adquirida en la tesis de maestría en donde se investigó temáticas relacionadas con el paralelismo y en particular las infraestructuras aplicadas a programas paralelos. Se visualiza a partir de la investigación realizada la posibilidad de:

- Definir la arquitectura de software de los modelos y/o las infraestructuras de programación paralelas.
- Especificar estrategias que mapeen un programa paralelo cualesquiera en una arquitectura paralela específica.

Lo antes mencionado permitirá beneficiarse de una mayor comprensión del dominio del programa y posibilitará crear un manual de ingeniería de especificaciones de sistemas de modelos y/o infraestructuras paralelas. A partir del conocimiento adquirido utilizando la aproximación antes mencionada se proyecta elaborar estrategias que relacionen el dominio del problema con el dominio del programa

contribuyendo de esta manera a facilitar la comprensión de este tipo de sistemas.

3. Resultados y Objetivos

La elaboración de métodos, técnicas y herramientas de CP implica detectar e implementar los conceptos comunes a disciplinas tales como modelos cognitivos, visualización de software y extracción de la Información. Para alcanzar este objetivo se requiere realizar investígaciones exhaustivas en cada una de esas disciplinas.

En este artículo, se ha conseguido: (i) explorar diferentes modelos y/o infraestructuras asociadas a los sistemas paralelos, (ii) arquitecturas que soportan este tipo de sistema, (iii) Se han analizado algunos métodos que permiten facilitar la CP paralelos, (iii) Se ha estudiado como relacionar, transformar y presentar el dominio del problema con el dominio del programa.

Durante este proyecto se espera realizar un análisis exhaustivo de los ítems antes mencionado y poder, finalmente proveer un método que simplifique la tarea de los programadores a la hora de migrar, mantener y rediseñar sistemas paralelos.

4. Formación de Recursos Humanos

Los trabajos que se han realizando en esta línea de investigación son la base preliminar para el desarrollo de tesis de grado y posgrado. Es importante mencionar que el grupo de investigadores de Argentina como de Portugal se encuentran dedicados a la captura de alumnos de grado y posgrado para la realización de estudios de investigación relacionados con las temáticas presentadas en este artículo. Dichos estudios pretenden fortalecer la relación entre ambas instituciones.

5. Referencias

[1]. Duane Albert. Adams. A computation model with data flow sequencing. Stanford University, USA,1969.

[2]. Leslie G. Valiant. A bridging model for parallel computation. Commun. ACM, 33(8): 103–111, 1990.

[3]. Michael Beynon, Chialin Chang, Umit Catalyurek, Tahsin Kurc, Alan Sussman, Henrique Andrade, Renato Ferreira, and Joel Saltz. Processing large-scale multi-dimensional data in parallel and distributed environments. Parallel Comput, 28(5): 827–859, May 2002.

[4]. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplefied data processing on large clusters. Commun. ACM, 51(1): 107–113, January 2008.

[5]. Margaret A. Storey. A Cognitive Framework for Describinh and Evaluating Software Exploration Tools. PhD thesis, Simon Fraser University, 1998.

[6]. Andrew Walestein. Cognitive Support in Software Engineering Tools: A Distributed Cognitive Framework. PhD thesis, Simon Fraser University, 2002.

[7]. Micheal P. O'Brien. Sofware Comprehension A Review and Research Direction. Technical Report, 2003.

[8]. Tim Tiemens. Cognitive Model of Program Comprehension. Technical Report, 1989.

[9]. Sarita Bassil and Rudolf k. Keller. A Qualitative and Quantitative Evaluation of Software Visualization Tools. Proc. of the IEEE Symposium on Information Visualization, pages 69–75, 2001.

[10]. Chaomei Chen. Information Visualization. Springer Verlag, 2006.

[11]. Marian Petre and Ed de Quincey. A Gentle Overview of Software Visualization. PPIG: Psychology of Programing Interest Group, pages 1–10, 2006.