Desarrollo y Sintonización Automática de Aplicaciones Paralelo/Distribuidas

Caymes-Scutari Paola[1,2], Tardivo María Laura[2], Méndez-Garabetti Miguel[2], Cía Flavia Carolina[2], Vega-Hissi Francisco Javier[2], Diamante Diego[2], Bianchini Germán[2]

[1]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

[2]Laboratorio de Investigación en Cómputo Paralelo/Distribuido
Departamento de Ingeniería en Sistemas de Información
Facultad Regional Mendoza/Universidad Tecnológica Nacional

Rodríguez 273 (M5502AJE) Mendoza

Tel. +54 261 5244579

pcaymesscutari@frm.utn.edu.ar, lauratardivo@dc.exa.unrc.edu.ar, miguelmendezgarabetti@gmail.com, ciacarolina@yahoo.com.ar, aehtiopicus@gmail.com seba_dmza@hotmail.com, gbianchini@frm.utn.edu.ar

Resumen

El cómputo paralelo se encuentra en continua expansión dada la potencia y la velocidad con la que pueden obtenerse los resultados. Desafortunadamente, el desarrollo de aplicaciones paralelas constituye una tarea no trivial, dado que involucra una serie de aspectos adicionales a los meramente funcionales que inciden directamente en la eficiencia y en la calidad de los resultados esperados, tanto en lo que respecta a precisión como a tiempo de respuesta. Cuando se trata de un usuario no experto, los efectos negativos tienden a potenciarse dada la falta de experiencia y habilidad para subsanar los problemas. Es por ello que esta línea de investigación aborda el desarrollo de un entorno para el desarrollo automático y la sintonización automática de aplicaciones paralelas que haga transparente el proceso de resolución del problema y la paralelización de la solución, mediante la instanciación de *problem solvers*. El usuario sólo clasifica y especifica el problema a resolver, mientras que la herramienta encapsula la resolución del problema. Como primer paso se trata la clase de problemas cuya resolución se lleva a cabo mediante algoritmos genéticos [Mic92].

Palabras clave: HPC, Cómputo Paralelo, Desarrollo Automático, Sintonización, Problem Solver

Contexto

La línea de I+D presentada en este artículo es coordinada y desarrollada en el ámbito del LICPaD (Laboratorio de Investigación en Cómputo Paralelo/Distribuido) de la Universidad Tecnológica Nacional – Facultad Regional Mendoza – Departamento de

Ingeniería en Sistemas de Información (UTN-FRM-DISI). Dicha línea está presente en el proyecto de investigación titulado “Desarrollo Automático de Aplicaciones Paralelo/Distribuidas basadas en Algoritmos Genéticos”, el cual cuenta con financiamiento tanto de la UTN como de la FRM.

Introducción

En los últimos años, los avances tecnológicos han propiciado un acelerado incremento en el rendimiento de los sistemas computacionales, muchos de ellos basados en algún tipo de paralelismo, coadyuvando a la creciente demanda de alto rendimiento para el tratamiento de problemas de gran tamaño y/o complejidad [Buy+99, Sta06]. No obstante, las aplicaciones para entornos paralelo/distribuidos son complejas de diseñar, las herramientas de programación paralelo/distribuida presentan diversas limitaciones, la ejecución de las aplicaciones no siempre hace un uso eficiente de los recursos computacionales involucrados y su optimización o sintonización muchas veces está fuera del alcance incluso del usuario disciplinar. El desarrollo de aplicaciones debe realizarse de una forma específica que permita su ejecución en un sistema paralelo/distribuido, lo que puede resultar una tarea complicada especialmente para usuarios no expertos dado que involucra el uso o aplicación de conceptos puramente informáticos como por ejemplo la concurrencia, los modelos de programación paralelo/distribuida (como Master/Worker o Pipeline [Mat+04]), las comunicaciones y/o diferentes tipos de middleware. Además, el desarrollo de aplicaciones distribuidas conlleva un conjunto de aspectos adicionales a los meramente algorítmicos que deben ser considerados para obtener un adecuado comportamiento, a saber la técnica de descomposición, la granularidad, el grado de concurrencia, el balanceo de carga y la escalabilidad, entre otros [Gra+03]. A lo largo de los años han surgido diferentes aproximaciones para facilitar la tarea del usuario, como la utilización de librerías que encapsulan primitivas de comunicación (como PVM [Gei+94] o MPI [Gro+96, Sni+96]) o la utilización de esqueletos que implementan patrones de comportamiento paralelo [Bac+95, Kul01, Mac+02, Ser+02]. Cada una de estas opciones ofrece facilidades a diferentes niveles pero a la vez presentan ciertas restricciones y requieren del usuario un cierto grado de conocimiento de los conceptos relacionados con concurrencia y paralelismo para llevar a cabo una correcta implementación del programa. Algunas aproximaciones afrontan de alguna manera el análisis de concurrencia para simplificar y/o asistir la tarea del usuario no-experto, pero aún requieren cierto grado de conocimiento y experiencia (como por ejemplo [Stu+03, You+92, Ter07]). Si bien estas herramientas constituyen un medio para mejorar el diseño, no proveen un mecanismo explícito para diseñar y/o desarrollar programas concurrentes y/o paralelos. Esto se debe a que el paralelismo que pueda lograrse en un programa depende mucho de la naturaleza del problema, lo cual dificulta la generalización cuando de herramientas automáticas se trata. Por otro lado, debido a que el rendimiento constituye un aspecto clave, tanto la calidad de la aplicación creada como el comportamiento de la misma en función del conjunto de datos de entrada y/o del estado del entorno de ejecución deben ser contemplados ya que pueden influir drásticamente en la utilidad del programa. A lo largo de los años se ha utilizado una serie de índices para evaluar el rendimiento de las aplicaciones, tales como el *speedup*, la

escalabilidad, la eficiencia, o el balanceo de carga [Wil+05]. Sin embargo, tales índices proporcionan una ponderación general de alguna característica particular del programa, pero ninguno proporciona información específica ni de utilidad para modificar y en consecuencia mejorar el comportamiento del programa. Es por ello que resulta habitual recurrir al proceso de sintonización. El proceso de sintonización u optimización es un proceso que implica varias fases y que permite buscar y solucionar problemas concretos en un programa para asegurar que no existen cuellos de botella durante la ejecución de la aplicación. En primer lugar, durante la fase de monitorización debe registrarse información acerca del comportamiento de la aplicación. En segundo lugar se ejecuta el análisis de la información. El análisis del rendimiento permite hallar cuellos de botella, deducir sus causas y determinar las acciones necesarias para su eliminación. Finalmente, debe realizarse la sintonización de la aplicación a través de la implementación de los cambios apropiados en el código para solucionar los problemas y mejorar la performance [Buy+99]. Aún cuando las herramientas existentes para el análisis y sintonización de aplicaciones representan un importante recurso de gran ayuda y utilidad para simplificar y asistir de alguna manera la tarea del usuario, existe una problemática asociada a la amplia variedad de opciones, tecnologías y aproximaciones tanto para el desarrollo de aplicaciones como para la sintonización de las mismas. En dicho escenario, el usuario continúa necesitando un alto grado de conocimiento para decidir cuál o cuáles de las herramientas serán las más adecuadas según las características de la aplicación y el entorno de ejecución. Por otro lado, en general cada una de las herramientas utiliza un cierto conjunto de formalismos propios que el usuario se ve obligado a aprender o modificar dependiendo de las características particulares de cada una de sus aplicaciones y de las herramientas que sea viable utilizar. Aquel usuario que no domina los conceptos de concurrencia y paralelismo queda ajeno a todo potencial de esta tecnología. En este contexto, es clara la necesidad de contar con herramientas que integren el proceso de desarrollo desde un nivel básico juntamente con el proceso de sintonización de aplicaciones paralelo/distribuidas, de modo que la intervención del usuario se vea simplificada.

Por todo lo expuesto, la línea de investigación presentada en este trabajo abarca dos grandes aspectos de la computación paralelo/distribuida: el desarrollo y la sintonización combinados en un mismo entorno. La arquitectura general del entorno se basa en tres módulos para cada *problem solver*. En primer lugar, una interface con el usuario que de soporte para la especificación de la aplicación. El segundo módulo o traductor, es responsable de traducir la información provista por el usuario a fin de instanciar el esqueleto del *problem solver*. Finalmente, el módulo de sintonización permite incorporar capacidades de adaptación de la ejecución a la aplicación. La sintonización de las aplicaciones resultará transparente al usuario, ya que durante la etapa de desarrollo –y de acuerdo a las especificaciones brindadas por el usuario– la aplicación se prepara automáticamente para poder ser sintonizada de acuerdo a las características que presente y al entorno de ejecución.

Líneas de Investigación y Desarrollo

La línea de investigación presentada en este trabajo posee tres ejes principales: el desarrollo automático, la sintonización automática y los métodos de optimización (o *problem solvers*). En lo que hace al desarrollo automático, involucra los aspectos de reutilización y generación de código, así como también la amigabilidad con el usuario, a fin de ofrecer una herramienta de uso sencillo. Por su parte, la sintonización automática permite dotar a la aplicación paralela generada con la capacidad de adaptar su ejecución a las condiciones del ambiente sin la necesidad de que intervenga el usuario. Ambas componentes (desarrollo y sintonización) dependen del *problem solver* que esté bajo consideración, para lo cual se requiere una familiarización previa con cada *problem solver*, de manera que se identifique su funcionamiento y dependencias, sus oportunidades o posibilidades de paralelización, y sus problemas de rendimiento a ser sintonizados.

Resultados y Objetivos

En la actualidad se cuenta con la infraestructura general del entorno. Como se ha mencionado anteriormente, el primer *problem solver* abordado ha sido el correspondiente a Algoritmos Genéticos, para el cual ya se cuenta tanto con la interface de usuario como con el módulo traductor, y se están incorporando las propiedades de sintonización automática. Al concluir la validación y depuración de los desarrollos alcanzados hasta el momento, se dará paso al tratamiento, diseño y desarrollo de otros *problem solver* que puedan adicionarse al entorno, de modo de ampliar de forma incremental su espectro de posibilidades para ofrecer soporte a las necesidades de los diferentes tipos de problemas (como búsqueda tabú, *simulated annealing*, evolución diferencial, etc. [Tal09]).

Formación de Recursos Humanos

Esta línea de investigación cuenta con la dirección de la Dra. Paola Caymes Scutari y la codirección del Dr. Germán Bianchini, quienes llevan adelante las tareas relacionadas con la planificación, administración y desarrollo del proyecto, así como también la formación de los alumnos de grado y postgrado que integran el LICPaD. En lo que hace a estudiantes de doctorado del LICPaD, esta línea de investigación cuenta con la participación de la Lic. María Laura Tardivo (cuyo plan de tesis doctoral versa específicamente dentro de esta línea de investigación) y del Ing. Miguel Méndez Garabetti, quienes cursan el doctorado en Ciencias de la Computación de la Universidad Nacional de San Luis, y a partir de abril del corriente año se incorporarán al programa de becas de CONICET (tipo I), realizando sus tareas doctorales en el marco del LICPaD. Además, se cuenta con la colaboración de tres estudiantes de grado de la carrera de Ingeniería en Sistemas de Información (Flavia Carolina Cía, Héctor Mazzucotelli y Diego Diamante) y de un alumno de la Universidad Católica (Javier Vega Hissi).

Referencias

[Bac+95] Bacci B., Danelutto M., Orlando S., Pelagatti S., Vanneschi M., “P3L: A structured high level programming language and its structured support”, Concurrency: Practice and Experience, 7(3):225-255, 1995.

[Buy+99] Buyya R. et al, "High Performance Cluster Computing – Architectures and Systems (Volume 1)", Prentice Hall, 1999.

[Gei+94] Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R., Sunderam V., "PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Network Parallel Computing", MIT Press. Cambridge, MA, 1994.

[Gra+03] Gramma A., Gupta A., Karypis G., Kumar V., "Introduction to Parallel Computing (2º Ed)", Pearson Addison Wesley. 2003.

[Gro+96] Groop W., Lusk E., Doss N., Skjellum A., "A high-performance, portable implementation of the MPI message passing interface standard", Parallel Computing, volume 22-6, pp.789-828, septiembre de 1996.

[Kul01] Kulkarni S., "An intelligent framework for Master-Worker applications in a dynamic metacomputing environment", Computer Science Department. University of Wisconsin – Madison. 2001.

[Mac+02] MacDonald S., Anvik J., Bromling S., Schaeffer J., Szafron D., Tan K., "From patterns to frameworks to parallel programs", Parallel Computing 28(12):1663-1684.

[Mat+04] Mattson T., Sanders B., Massingill B., "Patterns for Parallel Programming", Addison-Wesley, 2004.

[Mic92] Michalewicz Z., "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag, Berlin Heidelberg.

[Ser+02] Sérot J., Ginhac D., "Skeletons for parallel image processing: an overview of the SKIPPER project", Parallel Computing 28(12):1685-1708, 2002.

[Sni+96] Snir M., Otto S., Huss-Lederman S., Walker D., Dongarra J., "MPI: The Complete Reference". The MIT Press. Cambridge Massachusetts. London. England. 1996.

[Sta06] Stallings W., "Organización y Arquitectura de Computadores". Pearson Prentice Hall. 2006.

[Stu+03] Stuijk S., Ypma J., Basten T., "CAST - A Task-Level Concurrency Analysis Tool", 9th Annual Conference of the Advanced School for Computing and Imaging. ACSD 2003, pg 237-238. IEEE Computer Society.

[Tal09] Talbi E., "Metaheuristics – From Design to Implementation". Wiley. 2009.

[Ter07] Tersteeg A., "Application Concurrency Audit Tool: CFinder", http://softwarecommunity.intel.com/articles/eng/1613.htm. 2007.

[Wil+05] Wilkinson B., Allen M., "Parallel Programming - Techniques and Applications Using Networked Workstations and Parallel Computers", Pearson Prentice Hall. Second Edition. 2005.

[You+92] Young M., Taylor R., Forester K., Levine D., Brodbeck D., "A Concurrency Analysis Tool Suite: Rationale, Design and Preliminary Experience", Technical Report, TR-128-P. Software Engineering Research Center, Purdue University, Oct. 1992.