Procesamiento paralelo y distribuido. Fundamentos y aplicaciones.

R. Marcelo Naiouf, Armando E. De Giusti, Laura C. De Giusti, Franco Chichizola
Instituto de Investigación en Informática LIDI (III-LIDI)
Facultad de Informática – UNLP
{mnaiouf, degiusti, ldgiusti, francoch}@lidi.info.unlp.edu.ar

CONTEXTO

Esta línea de Investigación es parte del Proyecto "Algoritmos Distribuidos y Paralelos. Aplicación a Sistemas Inteligentes y Tratamiento Masivo de Datos" del Instituto de Investigación en Informática LIDI acreditado por la UNLP.

RESUMEN

La temática central de esta línea de investigación y desarrollo la constituye el estudio de los temas de procesamiento paralelo y distribuido, tanto en lo referente a los fundamentos como a las aplicaciones. Esto incluye los problemas de software asociados con la construcción, evaluación y optimización de algoritmos paralelos y distribuidos sobre arquitecturas multiprocesador.

Los temas de interés abarcan aspectos de fundamentos tales como paralelización de algoritmos, modelos de computación paralela, paradigmas paralelos, métricas del paralelismo, escalabilidad, balance de carga, predicción y evaluación de performance sobre diferentes clases de arquitecturas (homogéneas y heterogéneas) de soporte (cluster, multicluster, grid).

Asimismo, se trabaja en la concepción de aplicaciones paralelas numéricas y no numéricas específicas sobre grandes volúmenes de datos y cómputo intensivo.

Palabras clave: *Sistemas paralelos. Algoritmos paralelos y distribuidos. Clusters. Grid. Balance de carga. Evaluación de performance.*

1. INTRODUCCION

El procesamiento paralelo y distribuido se ha convertido en un área de gran importancia dentro de la Ciencia de la Computación, produciendo en muchos casos profundas transformaciones [1][2][3][4][5][6].

Un área de investigación es la especificación, transformación, optimización y evaluación de algoritmos distribuidos y paralelos. Interesa la optimización de soluciones en función de distintos modelos de arquitectura y el estudio de métricas de complejidad y eficiencia. Esto incluye el desarrollo de procesos paralelos, la transformación de algoritmos secuenciales en paralelos, y las métricas de evaluación de performance sobre diferentes plataformas de soporte (de hardware y software). En esta línea de I/D la mayor importancia está dada en los *algoritmos paralelos*, y en los métodos utilizados para su construcción y análisis de performance [7][8][9].

Si bien el paralelismo resulta un concepto intuitivo (utilizar múltiples procesadores para resolver problemas, en general de complejidad creciente y para obtener resultados con mayor velocidad), los fundamentos que subyacen a los mecanismos y soportes de paralelización presentan numerosas variantes. Un problema puede tener distintas formulaciones paralelas y la eficiencia de cada una dependerá de la arquitectura de soporte y el algoritmo específico que se desarrolle.

Numerosas áreas científicas y de la industria requieren la utilización de procesamiento paralelo y distribuido para la resolución de aplicaciones de cómputo intensivo. Entre ellas pueden citarse simulaciones, modelización, optimización discreta, análisis molecular, monitoreo de contaminación, búsquedas en árboles, aprendizaje en redes neuronales, tratamiento de imágenes, visión por computadora, reconocimiento de patrones, procesamiento de consultas en bases de datos, etc. [10][11][12][13][14][15].

La creación de algoritmos paralelos/distribuidos, o la transformación de un algoritmo secuencial en paralelo, se encuentra lejos de ser un proceso directo y está influida por la arquitectura física. En algunos casos el costo del paralelismo puede ser alto en términos del esfuerzo de programación: debe pensarse en la aplicación de técnicas nuevas reescribiendo totalmente el código secuencial, y las técnicas de debugging y tuning de performance no se extienden fácilmente al mundo paralelo [10][12][13][14]

Un *sistema paralelo* es la combinación de un algoritmo paralelo y la máquina sobre la cual éste se ejecuta; ambos factores poseen numerosas variantes y de un adecuado "matching" entre ambos depende el éxito de la aplicación. Respecto de los algoritmos, pueden ser especificados utilizando una diversidad de paradigmas (cliente/servidor, pipeline, dividir y

conquistar, SPMD); otra forma de clasificarlos es por la utlización de paralelismo de datos o de control. Por el lado de las arquitecturas, si bien todas poseen más de un procesador, pueden diferir en varias dimensiones tales como el mecanismo de control, la organización del espacio de direcciones, la granularidad de los procesadores, la red de interconexión, la sincronicidad, y la clase de procesadores utilizados (homogéneos o heterogéneos) [16][17].

Las arquitecturas para procesamiento paralelo han evolucionado, y en la actualidad las redes de computadoras constituyen una plataforma de cómputo paralelo muy utilizada por sus ventajas en términos de la relación costo/rendimiento. La noción de sistema distribuido como máquina paralela es común a las denominaciones redes de computadoras, NOW, redes SMP, clusters, multiclusters y grid. En estos casos, se deben identificar las capacidades de procesamiento, interconexión, sincronización y escalabilidad [18].

El concepto de multicluster es una generalización que permite que redes dedicadas a una aplicación paralela se interconecten y puedan cooperar en un algoritmo, compartiendo parcialmente recursos e incrementando la potencia de cómputo. Esto implica clusters dedicados interconectados, a diferencia del concepto de *grid computing* en que cada procesador puede realizar otras tareas independientes del algoritmo paralelo compartido, brindando alta disponibilidad de procesamiento y/o de almacenamiento [19][20][21][22][23][24][25][26]. En estos casos, existe un bajo grado de acoplamiento de los procesadores y un bajo rendimiento de la red de interconexión.

La caracterización y estudio de rendimiento del soporte de comunicaciones es de especial importancia para la predicción y optimización de performance de los algoritmos paralelos, así como la homogeneidad o heterogeneidad de los procesadores que componen la arquitectura [27][28]

Es importante referirse a un algoritmo paralelo mencionando el modelo de computación para el que fue diseñado. Uno de los objetivos en la definición de un modelo de computación es la posibilidad de *predicción de performance* que brinde el mismo. La computación monoprocesador se benefició por la existencia de un modelo teórico simple (RAM), que hizo posible desarrollar algoritmos y establecer correctitud y performance esperada de manera relativamente independiente de la máquina específica.

Al tratar las máquinas paralelas se encuentran un gran número de modelos, aunque no tan simples y precisos como RAM, y ninguno intenta servir como modelo para todas las clases de máquinas paralelas. Si bien existen varios modelos, BSP es uno de los más exitosos: es un modelo semi-asincrónico, en el cual los procesadores trabajan asincrónicamente L unidades de tiempo y luego son sincronizados por un mecanismo de hardware. Sin embargo, esta sincronización impide la superposición de cómputo y comunicaciones, lo que para algunos autores restringe en exceso el conjunto de algoritmos disponible. Algunas propuestas alternativas dieron origen a variantes que intentan aliviar el sincronismo explícito de BSP y predecir aceptablemente software asincrónico.

La diversidad de opciones en los sistemas paralelos complica el análisis de performance, ya que los ejes sobre los cuales pueden compararse dos sistemas son varios. En el mundo serial se puede realizar la evaluación a través de los requerimientos de tiempo y espacio del programa. En las aplicaciones paralelas, hay otra cantidad de medidas que pueden ser de interés, y esas métricas están ligadas tanto al algoritmo como a la arquitectura paralela.

La performance obtenida en el sistema paralelo está dada por una compleja relación en la que intervienen factores como el tamaño del problema, la arquitectura, la distribución de procesos en procesadores, la existencia o no de un algoritmo de balance de carga, etc.

Existen un gran número de métricas para evaluar sistemas paralelos. Entre las más conocidas se encuentran el tiempo de ejecución paralelo, el speedup (ganancia efectiva en velocidad de cómputo usando más de un procesador) y eficiencia (que denota el uso efectivo de los recursos de cómputo). Pero existen otras medidas que pueden ser útiles tales como costo, overhead paralelo, grado de concurrencia, escalabilidad, isoeficiencia, etc. [29]

El tema de la *escalabilidad*, y su relación con la función de isoeficiencia, es de importancia dado que permiten capturar las características de un algoritmo paralelo y de la arquitectura en la que se lo implementa. Permite testear la performance de un programa paralelo sobre pocos procesadores y predecir su performance en un número mayor. También permite caracterizar la cantidad de paralelismo inherente en un algoritmo, y puede usarse para estudiar el comportamiento de un sistema paralelo con respecto a cambios en parámetros de hardware tales como la velocidad de los procesadores y canales de comunicación.

El objetivo primario del paralelismo es reducir el tiempo de ejecución y hacer uso eficiente de los recursos de cómputo. La manera de asignar o *mapear* procesos lógicos a procesadores físicos es fundamental para la eficiencia: el uso desigual

(desbalance) de los procesadores puede degradar fuertemente la eficiencia del procesamiento paralelo. El *balance de carga* es un aspecto central del cómputo paralelo y consiste en, dado un conjunto de tareas que comprenden un algoritmo y un conjunto de computadoras donde ejecutarlas, encontrar el mapeo de tareas a computadoras que resulte en que cada una tenga una cantidad de trabajo que demande aproximadamente el mismo tiempo.

Un mapeo que balancea la carga de trabajo de los procesadores incrementa la eficiencia global y reduce el tiempo de ejecución. Este objetivo es particularmente complejo si los procesadores (y las comunicaciones entre ellos) son heterogéneos, y deben tenerse en cuenta las distintas velocidades.

El problema de asignación es NP-completo para un sistema general con *n* procesadores, y por lo tanto la tarea de encontrar una asignación de costo mínimo es computacionalmente intratable salvo para sistemas muy pequeños. Por esto pueden utilizarse enfoques alternativos como la relajación, el desarrollo de soluciones para casos particulares, la optimización enumerativa, o la optimización aproximada (que brindan soluciones subóptimas aunque aceptables) [30][31].

En algunos casos el tiempo de cómputo asociado con una tarea dada puede determinarse "a priori". En tales circunstancias, se puede realizar el mapeo antes de comenzar la computación (balance de carga *estático*). Para una clase importante y creciente de aplicaciones, la carga de trabajo para una tarea particular puede modificarse en el curso del cómputo y no puede estimarse de antemano; en estos casos el mapeo debe cambiar durante el cómputo (balance de carga *dinámico*), realizando etapas de balanceo durante la ejecución de la aplicación.

El balance estático, en general, es de menor complejidad que el dinámico, pero también menos versátil y escalable. Conceptualmente los métodos dinámicos requieren alguna forma de mantener una visión global del sistema y algún mecanismo de análisis para la migración de procesos y/o datos. Si bien potencialmente pueden mejorar la performance global de la aplicación redistribuyendo la carga entre los elementos de procesamiento, esta actividad produce overhead de comunicación y requiere espacio extra para mantener la información.

No puede establecerse un método efectivo y eficiente en todos los casos. Siempre la elección depende de la aplicación y la plataforma de soporte, y en muchos casos es necesario adaptar o combinar métodos existentes para lograr buena performance [30][32][33]

Es de interés la evaluación de performance de distintas clases de aplicaciones sobre las arquitecturas disponibles, de modo de adecuar los resultados teóricos de las métricas a la realidad. Esto se basa en que muchos sistemas paralelos no alcanzan su capacidad teórica, y las causas de esta degradación son muchas y no siempre fáciles de determinar. El análisis permite estudiar el impacto que tienen algunos de estos factores sobre las implementaciones, y adecuar las métricas clásicas a las mismas. En particular, interesa estudiar la influencia de las estrategias de distribución de procesos y datos, y la carga (estática o dinámica) asignada a cada procesador sobre el speedup, la eficiencia y la escalabilidad.

Desde el punto de vista de la relación costo/rendimiento, el cómputo paralelo implementado sobre arquitecturas distribuidas ha ganado rápidamente espacio en el campo de las aplicaciones reales dado el bajo costo de los procesadores y estaciones de trabajo estándares junto con su alto rendimiento. Si bien existe una amplia gama de posibilidades estudiadas y publicaciones con todo tipo de aplicaciones resueltas, también se acepta que es necesario continuar con la investigación en esta área [34].

Desde el punto de vista algorítmico se deben considerar varios factores, entre los cuales se pueden mencionar: heterogeneidad, bajo rendimiento de las comunicaciones (que impacta en la granularidad de las aplicaciones paralelas), adaptación de las estaciones de trabajo para cómputo paralelo, etc.

Entre las aplicaciones de interés se encuentran las numéricas y no numéricas, el tratamiento de imágenes y video, reconocimiento de patrones en secuencias de ADN (para trazabilidad, análisis de paternidad, abigeato, determinación de enfermedades), bases de datos distribuidas, sistemas inteligentes, data mining, etc.

2. LINEAS DE INVESTIGACION y DESARROLLO

- Algoritmos paralelos. Paralelización de algoritmos secuenciales. Optimización de algoritmos. Diseño de algoritmos.
- Especificación de procesos concurrentes y paralelos.
- Lenguajes y bibliotecas de comunicaciones para procesamiento paralelo y distribuido.
- Modelos de computación paralela. Predicción de performance en algoritmos paralelos.
- Paradigmas de programación paralela.
- Patrones de cómputo paralelo.

- Sistemas paralelos como combinación de software y arquitectura.
- Métricas del paralelismo. Speedup, eficiencia, rendimiento, isoeficiencia, balance de carga, etc.
- Escalabilidad de algoritmos paralelos en arquitecturas de distribuidas.
- Estudio de la complejidad de algoritmos paralelos.
- Análisis (teórico y práctico) de los problemas de migración y asignación óptima de procesos y datos a procesadores. Migración dinámica.
- Balance de carga estático y dinámico. Técnicas de balanceo de carga.
- Implementación de soluciones sobre diferentes modelos de arquitectura homogéneas y heterogéneas (clusters, multiclusters y grid). Ajuste conceptual del modelo de software al modelo de hardware, de modo de optimizar el sistema paralelo.
- Evaluación de performance de las soluciones paralelas.

3. RESULTADOS OBTENIDOS/ESPERADOS

- Desarrollar y optimizar algoritmos paralelos sobre los diferentes modelos de arquitectura multiprocesador. En particular, a problemas de tratamiento masivo de datos (secuencias de ADN, imágenes, video, bases de datos).
- Evaluar la eficiencia, rendimiento, speedup y escalabilidad de las soluciones propuestas.
- Investigar la migración de aplicaciones paralelas conocidas a esquemas multicluster y grid.
- Mejorar y adecuar las técnicas disponibles para el balance de carga (estático y dinámico) entre procesos.
- Estudiar los modelos de predicción/evaluación de performance con diferentes paradigmas de interacción entre procesos, en esquemas multicluster y grid. Proponer las adecuaciones necesarias.
- Estudiar la aplicación de mecanismos de tolerancia a fallas en arquitecturas de cluster, multicluster y grid.
- Formar recursos humanos en los temas del Subproyecto, incluyendo tesinas de grado y tesis de postgrado.

4. FORMACION DE RECURSOS HUMANOS

Existe cooperación con grupos de otras Universidades del país y del exterior. Dentro de la temática de la línea de investigación y desarrollo se espera concluir 4 tesis de doctorado que se encuentran en curso, 2 tesis de maestría, y al menos 3 Tesinas de Grado de Licenciatura.

5. BIBLIOGRAFIA

[1] J. Basney, M. Livny. "Deploying a High Throughput Computing Cluster". R. Buyya Ed., High Performance Cluster Computing: Architectures and Systems, Vol. 1, Prentice-Hall, Upper Saddle River, NJ, USA, pp. 116-134, 1999.

[2] Vijay K. Garg. “Elements of Distributed Computing”. Wiley-IEEE Press, 2002.

[3] M.L. Liu. “Distributed Computing: Principles and Applications”. Addison Wesley; 1st edition, 2003.

[4] A. Grama, A. Gupta, G. Karypis, V. Kumar. "Introduction to Parallel Computing", Pearson Addison Wesley, 2nd Edition, 2003.

[5] Vijay K. Garg. “Concurrent and Distributed Computing in Java”. Wiley-IEEE Press, 2004.

[6] Hagit Attiya, Jennifer Welch. “Distributed Computing: Fundamentals, Simulations, and Advanced Topics”, (Wiley Series on Parallel and Distributed Computing). Wiley-Interscience; 2nd edition, 2004.

[7] Kenneth A. Berman, Jerome L. Pau. “Algorithms: Sequential, Parallel, and Distributed”. Course Technology; 1st edition, 2004.

[8] Barry Wilkinson, Michael Allen. “Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers (2nd Edition)”. Prentice Hall, 2004.

[9] L. Ridgway Scott, Terry Clark, Babak Bagheri. “Scientific Parallel Computing”. Princeton University Press , 2005

[10] S. Akl. “Parallel Computation. Models and Methods”, Prentice-Hall, Inc., 1997.

[11] R. Miller, Q. F. Stout. “Algorithmic Techniques for Networks of Processors”, CRC Handbook of Algorithms and Theory of Computation, M. J. Atallah, ed, 1998.

[12] G. Andrews. “Foundations of Multithreaded, Parallel and Distributed Programming”, Addison Wesley, 2000

[13] C. Leopold. "Parallel and Distributed Computing. A survey of Models, Paradigms, and Approaches", Wiley Series on Parallel and Distributed Computing. Albert Zomaya Series Editor, 2001

[14] H. F. Jordan, G. Alaghband, H. E. Jordan. "Fundamentals of Parallel Computing", Prentice Hall, 2002

[15] www.EMNet.org. Sitio WEB del European Expertise in Biocomputing.

[16] K. Hwang, "Advanced Computer Architecture. Parallelism, Scalability, Programmability", McGraw Hill, 1993.

[17] D. Sima, T. Fountain, P. Kacsuk. "Advanced Computer Architectures. A Design Space Approach", Addison Wesley Longman Limited, 1997.

[18] T. Anderson, D. Culler, D. Patterson, NOW Team, "A Case for NOW (Networks of Workstations)", IEEE Micro, 15(1), 1995, pp. 54-64.

[19] Ahmar Abbas. "Grid Computing: Practical Guide To Technology & Applications (Programming Series)". Charles River Media; 1st edition, 2003.

[20] IEEE Task Force on Cluster Computing (www.ieeetfcc.org)

[21] F. Berman, G. Fox & A. Hey (Eds). "Grid Computing: Making The Global Infrastructure a Reality". John Wiley & Sons, 2003.

[22] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. "The data Grid: Towards and Architecture for the Distributed Mannagement and Analysis of Large Scientific data Sets". Journal of Network and Computer Applications, 2001. pp. 187-200.

[23] Ian Foster, Carl Kesselman. "The Grid 2: Blueprint for a New Computing Infrastructure". (The Morgan Kaufmann Series in Computer Architecture and Design). Morgan Kaufmann; 2nd edition, 2003.

[24] Zoltan Juhasz, Peter Kacsuk & Dieter Kranzlmuller (Eds), "Distributed and Parallel Systems: Cluster and Grid Computing". (The Intl Series in Engineering and Computer Science). Springer; 1st edition, 2004

[25] Daniel Minoli. "A Networking Approach to Grid Computing". Wiley-Interscience, 2004.

[26] Vladimir Silva. "Grid Computing For Developers" (Programming Series). Charles River Media; 1st edition, 2005.

[27] Goldman. "Scalable Algorithms for Complete Exchange on Multi-Cluster Networks". CCGRID'02, IEEE/ACM, Berlin, pp. 286-287, 2002.

[28] C. Kurmann, F. Rauch, M. Stricker. "Cost/Performance Tradeoffs in Network Interconnects for Clusters of Commodity PCs". Technical Report 391, Swiss Federal Institute of Technology Zurich, Institute for Computer Systems, January 2003

[29] X-H Sun. "Scalability versus Execution Time in Scalable Systems". Journal of Parallel and Distributed Computing, Number 12, 2002, pp 173-192

[30] C. Bohn, G. Lamont. "Load Balancing for Heterogeneous Clusters of PCs", Future Generation Computer Systems, Elsevier Science B.V., Vol 18, 2002, pp 389-400

[31] A. Cortés, A. Ripoll, F. Cedó, M.A. Senar, E. Luque. "An Asynchronous and Iterative Load Balancing Algorithm for Discrete Load Model", Journal of Parallel and Distributed Computing. Academic Press., Vol 62, 2002, pp 1729-1746

[32] F. Baiardi, S. Chiti, P. Mori, L. Ricci. "Integrating Load Balancing and Locality in the Parallelization of Irregular Problems", Future Generation Computer Systems, Elsevier Science B.V., Vol 17, 2001, pp 969-975

[33] Z. Lan, V. Taylor, G. Bryan. "A Novel Dynamic Load Balancing Scheme for Parallel Systems", Journal of Parallel and Distributed Computing, Vol 62, Number 12, December 2002, pp 1763-1781

[34] Timothy Mattson, Beverly Sanders, Berna Massingill. "Patterns for Parallel Programming". Addison Wesley Professional, 2004.