Metaheurísticas y entornos de ejecución

Carolina Salto[1,2], Gabriela Minetti[1], Carlos Bermudez[1], Hugo Alfonso[1], Cintia Ayala[1]
[1]Laboratorio de Investigación en Sistemas Inteligentes (LISI) - [2] CONICET
Facultad de Ingeniería - Universidad Nacional de La Pampa
Calle 110 Esq. 9 (6360) General Pico - La Pampa - Rep. Argentina
Te. / Fax: (02302) 422780/422372, Int. 6302

Resumen

La contribución de esta línea de investigación es analizar el impacto en el rendimiento de una metaheurística paralela cuando se ejecuta usando un conjunto de recursos de computación heterogéneos. Como entorno de ejecución se consideran procesadores que tengan distintas velocidades, capacidades de memoria y almacenamiento, arquitecturas y sistemas operaticos. Además se contempla el uso de unidades de procesamiento gráficas, que aportan un gran poder de cómputo en la ejecución de algoritmos paralelos.

CONTEXTO

Esta línea de investigación se desarrolla en el marco del proyecto de investigación "Resolviendo problemas complejos con técnicas metaheurísticas avanzadas", dirigido por la Dra. Carolina Salto, y llevado a cabo en el Laboratorio de Investigación de Sistemas Inteligentes (LISI), de la Facultad de Ingeniería de la Universidad Nacional de La Pampa. Además, las tareas se realizan en el marco del proyecto PICTO-2011-0278-UNLPAM "Metaheurísticas aplicadas a problemas de optimización en PyMEs". El grupo de trabajo mantiene desde hace varios años una importante vinculación con investigadores de la Universidad Nacional de San Luis (Argentina) y de la Universidad de Málaga (España) con quienes se han realizado varias publicaciones conjuntas.

INTRODUCCIÓN

El campo de las metaheurística paralelas está en continua evolución como resultado del surgimiento de nuevo hardware y de las necesidades que los investigadores van demostrando. Cuando se trabaja con un algoritmo paralelo, es importante tener en cuenta la plataforma de ejecución en la cual ha sido implementada, debido a que la arquitectura de hardware impacta notablemente en el tiempo requerido para realizar las operaciones, las comunicaciones, las sincronizaciones y la compartición de datos. Hasta la última década, las propuestas clásicas de metaheurísticas paralelas se enfocaban en supercomputadoras y cluster de estaciones de trabajo. Actualmente, el surgimiento de arquitecturas de computación paralela, tal como procesadores multicore, unidades procesamiento gráfico o ambientes grid, proveen nuevas oportunidades para desarrollar técnicas de computación paralelas para mejorar la solución de problemas y disminuir los tiempos de procesamiento requeridos.

La mayoría de los resultados reportados sobre algoritmos distribuidos (AEs) distribuidos asumen que el ambiente de hardware subyacente tiene características idénticas (ambiente homogéneo) teniendo en cuenta no sólo componentes de hardware (procesadores, memoria, redes) sino también software (sistema operativo) [1]. Por otra parte, esta clase de homogeneidad en hardware es muy difícil de mantener en el transcurso del tiempo. Una de las principales razones es que frente a roturas en alguna de las máquinas se hace difícil construir un sistema igual teniendo en cuenta el mismo procesador, arquitectura interna y configuración. Por otra parte, el rápido desarrollo de la tecnología en el diseño de procesadores, redes y almacenamiento de datos sumado al constante decremento de la relación costo/rendimiento, posibilita el uso de nuevos recursos disponibles. Como consecuencia, la coexistencia de equipamiento nuevo y viejo en un ambiente de computación ha dado lugar a la emergencia de plataformas paralelas heterogéneas, las cuales son actualmente muy comunes

en laboratorios, compañías, instituciones, campus, etc. Uno de los primeros trabajos que trata con ambientes de computación heterogéneos y AEs distribuidos se puede hallar en [2]. Trabajos más recientes relacionados con ambientes heterogéneos se pueden encontrar en [3], [7].

Las plataformas multiprocesador son herramientas útiles para el desarrollo e implementación de aplicaciones paralelas de metaheurísticas. La disponibilidad de múltiples recursos integrados en un único dispositivo permite tratar con problemas de optimización que requieren una solución rápida, o en tiempo real, en una plataforma fácil de programar. Las unidades de procesamiento gráfico (GPUs) es un ejemplo representativo de plataformas multiprocesador. Originalmente diseñadas como dispositivos específicos para procesamiento gráfico, se han transformado en plataformas muy poderosas a costos accesibles para utilizar con propósitos de cómputo general. Por esta razón, el estudio de implementaciones de AEs usando GPUs ha crecido a pasos agigantados, ya que ayuda a reducir el tiempo de ejecución de estos algoritmos mediante la explotación del paralelismo masivo de tales dispositivos. La relación costo versus poder de cómputo han posicionado a las GPUs al frente de la próxima generación de infraestructuras de cómputo de alto rendimiento.

LÍNEAS DE INVESTIGACIÓN, DESARROLLO E INNOVACIÓN

En esta sección se describen dos líneas de investigación y desarrollo: (*i*) la relacionada con el aprovechamiento de los distintos recursos hardware disponibles en un laboratorio en la ejecución de una metaheurística y (*ii*) la relacionada con el desarrollo de algoritmos evolutivos para hacer uso de las ventajas en velocidad de procesamiento que brindan las GPUs.

En cuanto a la primera línea de investigación, hemos considerado la ejecución un algoritmo evolutivo distribuido (dEA) [10] en un conjunto de procesadores con rendimientos dispares, desde modernos a viejos. El dEA considerado es un modelo multipoblación (islas) que realizan intercambios de individuos entre las distintas subpoblaciones. Las diferentes subpoblaciones son evolucionadas por procesadores con diferentes velocidades, en consecuencia en un determinado momento cada una se encuentra en diferentes estados de la evolución. Tan pronto una isla ejecutando en un procesador rápido llega al intercambio de soluciones, envía una solución a sus vecinos y continua con su procedimiento de optimización de una manera asíncrona. Las soluciones desde sus vecinos más lentos deberán tomar más tiempo para diseminarse a otras islas. La metodología consiste en calcular las diferencias relativas entre las velocidades de cada procesador. Con esta información, se puede establecer un ordenamiento de los recursos hardware, que puede ayudar a tomar decisiones más informadas respecto de los valores de los parámetros del algoritmo. Esto facilita el desarrollo de un mecanismo que habilita al AE distribuido tratar con diferencias en las características de los procesadores involucrados en la ejecución el mismo. Lo que se busca es contestar los siguientes interrogantes: (*i*) ¿la exactitud del algoritmo evolutivo distribuido se mantiene al usar hardware heterogéneo? y (*ii*) ¿el uso de hardware heterogéneo permite obtener tiempos de ejecución aceptables?.

Relacionado con la segunda línea de investigación, se trabaja en el diseño de algoritmos evolutivos para resolver problemas de optimización utilizando como soporte de ejecución GPUs. El objetivo que se persigue es diseñar un AE que se ejecute completamente sobre una GPU para resolver el problema de corte máximo de un grafo, conocido como MaxCut [6]. Es un problema NP-duro muy conocido y además de su importancia teórica, tiene aplicaciones en varios campos ([4], [11], [8]). Aprovechando las ventajas brindadas por la GPU para acelerar el proceso de optimización se implementaron, utilizando NVIDIA CUDA, todos los operadores del AE para que se ejecuten como *kernel* en la GPU. Es importante analizar el desempeño y la escalabilidad del algoritmo diseñado teniendo en cuenta distintos parámetros (tamaño de población, operadores de variación, probabilidades, entre otros) y también desde el punto de vista del incremento en la complejidad del problema a resolver.

RESULTADOS OBTENIDOS/ESPERADOS

En esta sección se detallan los resultados obtenidos en las distintas líneas de investigación en desarrollo, presentadas en la sección anterior.

Siguiendo un un procedimiento metodológico, se propuso un diseño algorítmico [9] para tratar con la ejecución de un AE distribuido sobre un conjunto de elementos de procesamiento heterogéneos, en cuanto a diferencias en la velocidad de ejecución, la arquitectura, el sistema operativo, etc. Como primer paso se caracteriza la plataforma heterogénea con distintas puntuaciones obtenidas por algoritmos de testeo científicos (tal como Whitstone, Drystone, Livermore Loops, entre otros). Muchas veces ocurre que estos tests no reflejan las operaciones que realizan las aplicaciones, en este caso metaheurísticas, por lo que también se obtiene un ordenamiento de las máquinas considerando los tiempos de ejecución de la metaheurística. Con esta información se determina cuál de los algoritmos de testeo producen el mismo ordenamiento y se establece la relación entre las distintas máquinas. Con esta información, se configuran algunos parámetros del AE distribuido de modo que todas las islas terminen en el mismo momento, sin importar las características del procesador en uso. Además el AE distribuido se mejora con un mecanismo que consiste en modificar la política de migración para que todas las islas reciban un flujo similar de inmigrantes independientemente de la velocidad del procesador sobre el que se esté ejecutando.

El algoritmo distribuido desarrollado fue comparado con un dEA ejecutado en un ambiente de computación homogéneo. Independientemente de las características de la plataforma de ejecución, se obtuvieron resultados similares en varios indicadores de rendimiento tal como la tasa de éxitos y el tiempo de ejecución. La pecularidad es que el dEA ejecutando en un ambiente heterogéneo obtiene buenas soluciones en una menor cantidad de evaluaciones. Por lo tanto, los investigadores podrían usar los elementos de procesamiento dispares disponibles en un laboratorio para la ejecución de algoritmos de optimización.

En la actualidad, nos encontramos abocados al análisis de los efectos e influencia que causa la topología en la comunicación entre las distintas islas, cuando un AE distribuido se ejecuta en una plataforma heterogénea.

Desarrollamos e implementamos un AE para GPU (AG-GPU) [5] para resolver el problema de MaxCut, utilizando tecnología CUDA. Esto permitió conocer y experimentar en el desarrollo de algoritmos de optimización en estos entornos de ejecución paralela. Realizamos un estudio preliminar para obtener una configuración adecuada de los parámetros del algoritmo. Para ello, uno de los estudios consistió en determinar el operador de cruce más adecuado. La representación utilizada para representar el grafo es binaria, para la cual existe una amplia variedad de operadores de cruce, entre ellos el operador de dos puntos y el tradicional operador de un punto.

Del estudio realizado se observó el buen desempeño de AE con operador de dos puntos ya que en el 80 % de las instancias utilizadas obtuvo soluciones de mejor calidad que el AE con operador de un punto. Este mejor desempeño se hace más evidente en las instancias que tienen una menor complejidad evidenciada en una menor densidad de nodos conectados entre sí. Por último, no parece haber una relación directa entre la cantidad de nodos y la calidad de los resultados, ya que los mejores resultados se obtuvieron en instancias que tienen 2000 nodos, mientras que en las que se obtuvieron valores menores tienen 800 nodos.

Otro de los ejes de esta línea de investigación consiste en medir la ganancia de tiempo que se obtiene de ejecutar el algoritmo en paralelo (ambiente GPU) respecto de su ejecución en serie (entorno CPU). Para ello se efectuó la misma experimentación utilizando los mismos parámetros utilizados en la versión desarrollada para GPU, pero ahora con un algoritmo modificado para que pueda correr completamente en un CPU (con un único hilo de ejecución). Con estos resultados se pueden relacionar los mejores tiempos de la versión paralela con los mejores tiempos de su versión serie, y así obtener una medida de rendimiento conocida como speed-up. Los valores de speedup obtenidos entre 1 y 17, y están relacionados con la densidad del grafo de corte que representa cada instancia. Una de las ventajas de nuestra implementación es que se puede ejecutar en cualquier GPU nVidia y plataforma Linux/Windows.

En la actualidad se trabaja en estrategias de optimización específicas de CUDA para sacar el mayor provecho a la unidad de procesamiento gráfico. También nuestra investigación estará orientada a la implementación de algoritmos genéticos paralelos,

en especial bajo el modelo isla, para una plataforma GPU usando la tecnología CUDA.

FORMACIÓN DE RECURSOS HUMANOS

La actividad de formación se recursos humanos está orientada a la formación de becarios de iniciación en la investigación. En este marco, un alumno de la carrera Ingeniería en Sistemas se encuentra desarrollando las actividades planificadas para una Beca de Estímulo a las Vocaciones Científicas, otorgada por el CIN para el periodo 2013-2014. Además se formaron otros dos alumnos avanzados quienes accedieron a Becas de Iniciación en la Investigación otorgadas por la UNLPam. Por otra parte, es interesante resaltar que en el LISI se trabaja con alumnos avanzados en la carrera Ingeniería en Sistemas en temas relacionados a la resolución de problemas de optimización usando técnicas inteligentes y en algoritmos paralelos, con el objeto de guiarlos en el desarrollo de sus tesinas de grado y, también, de formar futuros investigadores.

REFERENCIAS

[1] E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley, 2005.

[2] E. Alba, A.J. Nebro, and J.M. Troya. Heterogeneous computing and parallel genetic algorithms. *Journal of Parallel and Distributed Computing*, 62:1362–1385, 2002.

[3] J. Dominguez E. Alba. A methodology for comparing the execution time of metaheuristics running on different hardware. In Jin-Kao Hao and Martin Middendorf, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 7245 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2012.

[4] M. Anjos and F. Liers. Global approaches for facility layout and vlsi floorplanning. *Handbook of Semidefinite, Cone and Polynomial Optimization: Theory, Algorithms, Software and Applications*, page 32, 2010.

[5] C. Bermudez and C. Salto. Análisis del comportamiento de un ag para gpus. In *XIX Congreso Argentino de Ciencias de la Computación (CACIC 2013)*, 2013.

[6] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85 – 103. 1972.

[7] S. Mostaghim, J. Branke, A. Lewis, and H. Schmeck. Parallel multi-objective optimization using master-slave model on heterogeneous resources. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1981 –1987, 2008.

[8] S. Saha and S. Bandyopadhyay. Automatic mr brain image segmentation using a multiseed based multiobjective clustering approach. *Applied Intelligence*, Online First, June 2010.

[9] C. Salto, , F. Luna, and E. Alba. Distributed evolutionary algorithms in heterogeneous environments. *Eighth International Conference on Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2nd International Workshop on Soft Computing Techniques in Cluster and Grid Computing Systems (SCCG2013)*, 2013.

[10] R. Tanese. Distributed genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 434–439, 1989.

[11] B. Verma and S.Z. Hassan. Hybrid ensemble approach for classification. *Applied Intelligence*, Online, 2009.