Análisis de desempeño de una Implementación del Algoritmo K-means en CUDA y OMP

Joaquin Jimenez, Raúl Oscar Klenzi, Alejandra Malberti

Instituto de Informática / Departamento de Informática /Facultad de Ciencias Exactas Físicas y Naturales / Universidad Nacional de San Juan

Av. Ignacio de la Roza 590 (O), Complejo Universitario "Islas Malvinas", Rivadavia, San Juan, Teléfonos: 4260353, 4260355 Fax 0264-4234980, Sitio Web: http://www.exactas.unsj.edu.ar

joaquinjimenez33, rauloscarklenzi, amalberti@gmail.com

Resumen

En este trabajo se analiza el desempeño correspondiente a la implementación de un algoritmo de segmentación sobre hardware con procesadores multinúcleo y con Unidades de Procesamiento Gráfico –GPU- basado en la placa de video Nvidia utilizando programación CUDA, y se muestran métricas de ejecución, que miden y comparan su desempeño respecto de su equivalente OpenMP y secuencial.

Se paraleliza el algoritmo de segmentación K-means que se nutre con datos obtenidos del área de la astronomía que consisten en un catálogo descripto por una matriz con filas que representan galaxias y columnas que constituyen sus parámetros o atributos característicos.

También se propone evidenciar el orden complejidad temporal de K-means, que es O(tkn)[1], donde n es el número de objetos, k el número de clusters y t el número de iteraciones. Se harán pruebas cambiando los valores de estas variables, mostrando los cambios en los tiempos de ejecución y en los SpeedUps logrados respecto del código secuencial.

Palabras clave: Segmentación, K-means, CUDA, OpenMP.

Contexto

Petabytes de datos se vierten cada día en redes de computadoras (Intranets), la World Wide Web, y en diferentes dispositivos de almacenamiento de datos masivos provenientes entre otras, de aplicaciones de negocio, de redes sociales, y de experiencias científicas de diversas áreas temáticas entre la que se destaca la astronomía y sobre los cuales se realizan tareas que permiten extraer información y conocimiento subyacente

Este contexto permite elevar la presente propuesta que está contenida en proyecto bianual "Extracción de Conocimiento en Datos Masivos" aprobado por el Consejo de Investigaciones Científicas, Técnicas y de Creación Artística de la Universidad Nacional de San Juan (CICITCA-UNSJ) y sujeto a evaluación externa. Al mismo lo integran docentes y alumnos del Departamento Informática quienes trabajan en conjunto con Docentes del departamento Geofísica y Astronomía, investigadores de CONICET en el proyecto PICT 2010 Bicentenario N°0680 y PIP 2012-2014 GI (Código: 11220110100298), y cuyas tareas se centran en análisis estadísticos que permitan extraer conocimiento, utilizando

catálogos públicos de galaxias, cúmulos de galaxias, quasares, galaxias con núcleos activos, etc. El proceso de investigación exige resultados claros y confiables, siendo necesario recopilar permanentemente grandes cantidades de datos de objetos extragalácticos. Un claro ejemplo es el uso del séptimo relevamiento fotométrico del catálogo Sloan Digital SkySurvey (SDSS-DR7). Este catálogo, se actualiza cada cuatro años y consta de 357 millones de objetos con gran cantidad de parámetros, conformando un total de 18TB. Sobre una fracción de estos datos se aplica el algoritmo de segmentación K-means en hardware con procesadores multinúcleo y GPU Computing permitiendo extraer automáticamente conocimiento y comparar el desempeño de cada implementación.

Introducción

OpenMP

OpenMP [2] es una interfaz de programación de aplicaciones (API) para programación paralela con memoria compartida. Es una extensión para lenguajes como C, C++ y Fortran que añade concurrencia a los programas. Incluye directivas de compilador, palabras reservadas de programación, rutinas de bibliotecas y variables de entorno.

Fue desarrollado en conjunto por desarrolladores de Hardware y de Software, logrando un modelo portable y escalable a través de una interfaz simple. Mediante un modelo de programación basado en la forma fork/join de Unix, hace que una tarea pesada se divida en hilos más livianos.

En este trabajo se usa OpenMP en un entorno multinúcleo, en una CPU core i7 de 8 núcleos lógicos.

CUDA

Una Unidad de Procesamiento Gráfico (GPU) es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante y su utilización, permite aligerar la carga de trabajo de la CPU.

Las GPU se diseñaron con el propósito específico del procesamiento de imágenes y tienen dos características sobresalientes. Primeramente, como el tratamiento de imágenes implica un alto grado de paralelismo, por las operaciones sobre pixeles, las GPU poseen cientos de procesadores shader unificados con una frecuencia de reloj no tan elevada (entre 600MHz y 1000MHz) evitando así su sobrecalentamiento. En segundo lugar, las GPU actuales están optimizadas para cálculo con valores en coma flotante, predominantes en los gráficos 3D[3]. Así, las GPU se pueden usar en tareas de propósito general, sobre todo aquellas que requieren computación intensiva.

Nvidia ha creado CUDA, una plataforma diseñada conjuntamente a nivel de software y de hardware, y hace referencia tanto a un compilador como a un conjunto de herramientas de desarrollo que permiten hacer aplicaciones para GPUs, mediante una extensión del lenguaje de programación C.[4]

Mediante el uso de CUDA, es posible construir aplicaciones paralelas capaces de aprovechar los múltiples cores de una GPU, lo que implica comprender la interacción de datos y de control que se produce entre la CPU y la GPU, dando lugar a un nuevo modelo de programación paralela que combina un enfoque distribuido (la interacción entre CPU y GPU) y uno paralelo (el que se da interno en la GPU).[5]

Minería de Datos

Minería de datos es el proceso de encontrar nuevas y significativas

correlaciones, patrones y tendencias desmenuzando grandes cantidades de datos almacenados en repositorios[6]. Los patrones extraídos deben ser válidos, potencialmente útiles, y entendibles. Es un área multi-disciplinaria, que involucra matemática, bases de datos, inteligencia artificial, heurísticas, visualización de datos y aprendizaje de máquina [1].

Existen diversas técnicas de minería de datos, en este paper se trabaja en particular la técnica de segmentación. El proceso de segmentación consiste en tratar de agrupar un conjunto de objetos o casos de estudio en grupos de objetos similares. Un grupo, es un conjunto de objetos similares entre sí, pero distintos de los objetos de otros grupos.

El objetivo principal es lograr que los objetos de un determinado grupo sean lo más similares posibles en base a algún criterio de similitud, y lo más diferentes de los objetos de otros grupos[1][7].

K-Means es uno de los métodos de segmentación más sencillos y eficientes[6][1]. Se parte de un conjunto de objetos o instancias $D = \{x_1, x_2, \dots, x_n\}$ donde cada $x_i = (x_{i1}, x_{i2}, \dots, x_{ir})$. En esta aplicación, cada objeto es un vector de r atributos, definidos en el campo de los reales. K-means divide a los objetos en k grupos. Cada uno de estos grupos tiene un vector asociado denominado centro del cluster o centroide. Este centroide se obtiene simplemente promediando los objetos del cluster que representa. Dado que existirán k centroides, los cuales son los promedios de los grupos, surge el nombre del algoritmo, K-Means (K medias o K promedios).

Al inicio del algoritmo no hay grupos formados, los centroides se forman tomando k objetos aleatorios. Luego, para cada objeto, se busca el centroide más cercano a él y se determina que el objeto en cuestión pertenece al grupo que dicho centroide representa. Los grupos quedan finalmente conformados por los objetos que contiene y por el centroide que lo representa. Una vez que todos los objetos han sido asignados, se recalcula el centroide promediando todos los objetos del grupo. El proceso se repite hasta alcanzar algún criterio de convergencia, que suele ocurrir cuando se alcanza un número máximo establecido de iteraciones o cuando la cantidad de objetos que ha cambiado de grupo está por debajo de un umbral determinado.

La principal fortaleza del algoritmo es su simplicidad y eficiencia[1]. Su orden de complejidad temporal es O(tkn), con: n número de objetos, k número de clusters y t cantidad de iteraciones. Dado que tanto k como t suelen ser considerablemente menores que n, se considera que el algoritmo tiene complejidad lineal en n.

Líneas de investigación, Desarrollo e Innovación

El algoritmo K-means se implementa en la placa de video Nvidia sobre la que se ejecuta el programa CUDA, obteniéndose métricas de ejecución, que comparan su performance respecto de su equivalente OpenMP y secuencial.

La placa de video utilizada para llevar a cabo las pruebas es la GeForce GT 610, en una máquina Intel Core I7, de 8 núcleos. Esta placa de video posee 48 cores, memoria global de 1024 MB, memoria compartida de 48 KB por bloque y la cantidad de registros por bloque es de 32K registros.

La implementación de K-means utilizada es una versión modificada del código de Serban Giuroiu[8], un egresado de la Universidad de California Berkeley e ingeniero en Google. Este, a su vez se basó en el código y estudios realizados por el Profesor Wei-kengLiao de La Universidad Northwestern, de Estados Unidos. El código de Serban está disponible en: http://serban.org/software/kmeans/

En el enlace se encuentra una descripción general del código, el cual provee de implementaciones de K-means tanto secuencial como paralela en OpenMP, MPI y CUDA. La versión CUDA es la que se usó y mejoró para este informe. Serban Giuroiu concede libertad de trabajar con su código. Esto se aclara en una licencia que se puede descargar desde la misma página.

Resultados y Objetivos

Se muestran los resultados de ejecución del programa secuencial, de la versión OpenMP y CUDA, sobre archivos con distinta cantidad de registros (n) y cluster (k). El Speed Up se calcula dividiendo el tiempo secuencial sobre el tiempo CUDA.

Las gráficas expresan en el eje de abscisas, la cantidad de clusters y en ordenadas el tiempo en segundos. La línea de tiempo de CUDA es azul, la de OMP roja y la secuencial verde.

k	Sec.	OMP	CUDA	Speedup
2	0.2605s	0.2009s	0.1630s	1.59x
4	0.6896s	0.4786s	0.2215s	3.11x
16	10.2199s	6.9149s	0.8746s	11.68x
64	107.6821s	74.5335s	6.4831s	16.6x

Tabla 1. Archivo de28MB, 374669 registros y 7 atributos.

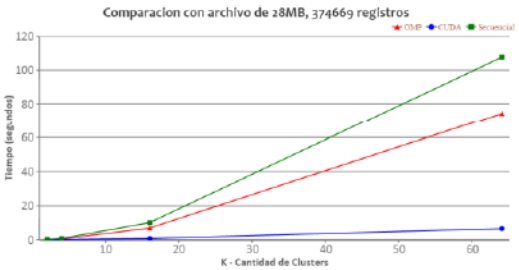

Figura 1. Archivo de28MB, 374669 registros y 7 atributos.

k	Sec.	OMP	CUDA	SpeedUp
2	0,4071	0,3113	0,3842	1,06
4	0,3461	0,2296	0,1843	1,88
16	29,9413	18,6268	3,3844	8,85
64	159,9451	99,2352	11,3810	14,05

Tabla 2. Archivo de 84MB, 867906 registros y 9 atributos.

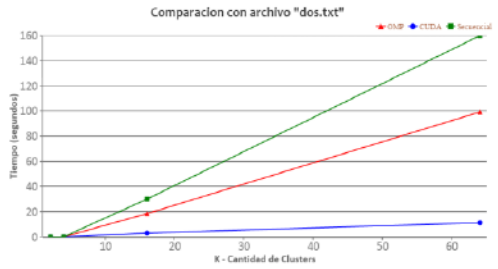

Figura 2. Archivo de 84MB, 867906 registros y 9 atributos.

k	Sec.	OMP	CUDA	SpeedUp
2	0,6021	0,4936	0,4374	1,38
4	0,5146	0,3576	0,2250	2,29
16	15,2952	9,9598	1,4863	10,29
64	239,2600	139,6318	15,0478	15,90

Tabla 3. 140MB, 1301859 registros, 9 atributos.

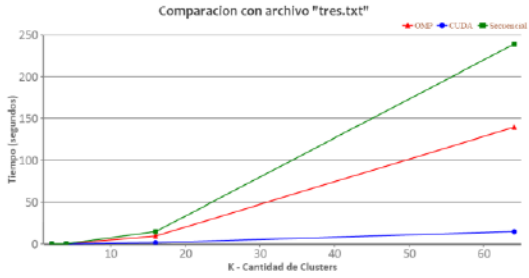

Figura 3. 140MB, 1301859 registros, 9 atributos.

k	Sec.	OMP	CUDA	SpeedUp
2	1,4008	0,8905	1,0247	1,37
4	2,3905	1,4327	0,6454	3,70
16	71,8736	42,0697	6,1892	11,61
64	301,4826	179,4260	20,7529	14,53

Tabla 4. 322MB, 3037670 registros, 9 atr.

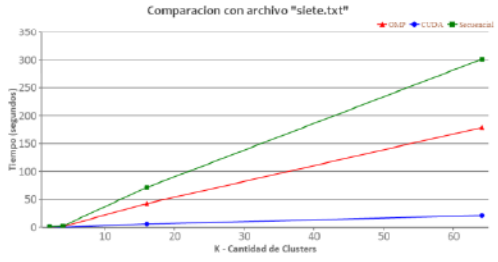

Tabla 5. 322MB, 3037670 registros, 9 atr.

Conclusiones

Los mayores SpeedUp se logran a medida que se aumenta el valor del k, dado que implica mayor cómputo (más centroides con los que comparar los objetos y más trabajo en su recalculo). Los tiempos CUDA son siempre menores

que los OpenMP y estos siempre menores que los secuenciales, constatándose un SpeedUp de hasta 16x, lo cual deja en claro las ventajas computacionales de CUDA. Así mismo se constata desde la linealidad de la gráfica el orden de complejidad temporal O(tkn).

El trabajo deja entrever las ventajas del uso de sistemas paralelos para el procesamiento de grandes volúmenes de datos, evidenciándose las mejoras relativas que se pueden alcanzar respecto del procesamiento secuencial.

El archivo de mayor extensión utilizado contiene algo más de 3 millones de registros, pesa 322MB y representa menos del 1% del total de datos.

Como trabajo a futuro se propone el desarrollo de otros algoritmos de minería de datos en CUDA, e incorporar la utilización de las nuevas generaciones de placas de video y de las buenas características que estas puedan tener.

Formación de Recursos Humanos

En el marco del proyecto, en el que se genera esta propuesta, han culminado recientemente tres trabajos finales de Licenciatura, a la vez que se están desarrollando otros dos específicamente en Ciencias de la Computación destacándose: "Análisis De Algoritmos De Minería De Datos En Datos Masivos". La propuesta temática y méritos del alumno que desarrolla su trabajo final, permitió que se lograra un apoyo institucional a través de una beca de investigación de alumnos avanzados culminada en Agosto de 2014 y conforme se avanzaba en diferentes etapas de la misma permitió exponerlas, a nivel de poster, en la 57 Reunión de la Asociación Argentina de Astronomía Córdoba Set-2014, como así también en el 2º Congreso nacional de Ingeniería Informática y Sistemas de Información (2º CONAIISI) Nov-2014 en la ciudad de San Luis.

También se dirigen trabajos finales de posgrado en el marco de la Maestría de Informática de la Universidad Nacional de La Matanza, que en etapa de redacción de informe final permite publicar, en la edición de marzo, el trabajo "Análisis de Planes de Estudio Mediante Determinación Automática de Pertinencias Sintáctico-Temáticas en Carreras de Informática" en la Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação. Además se dirigen trabajos finales en el marco de la Maestría de Informática de la FCEFN-UNSJ.

Referencias

[1] B. Liu, Web Data Mining, Springer, 2007.

[2] N. Matloff, Programming on Parallel Machines GPU, Multicore, Cluesters and More, 2012.

[3] http://es.wikipedia.org/wiki/ Unidad_de_procesamiento_gráfico.

[4] N. Wilt, The CUDA Handbook, Addison-Wesley, 2013.

[5] M. Ujaldon, *Programando la GPU con CUDA*, Rio Cuarto, 2014.

[6] D. T. Larose, Discovering Knowledge in Data, 2005.

[7] S. S.-S. y. S. Ben-David, Understanding Machine Learning, Cambridge, 2014.

[8] S. Giuroiu, http://serban.org