Cod. 8471 | Ciencia de los ordenadores | 1203.26 Simulación

Aplicación de la computación paralela con unidad procesadora de gráficos para el análisis de un sistema mecánico ferroviario

Parallel computing application with graphics processor unit for analysis of a mechanical railway system

Alejandro Bustos-Caballero, Higinio Rubio-Alonso, Eduardo Corral-Abad y Juan-Carlos García-Prada
Universidad Carlos III de Madrid (España)

DOI: http://dx.doi.org/10.6036/8471

Muchos ordenadores incorporan tarjetas gráficas avanzadas que presentan características muy atractivas para el cómputo de problemas de ingeniería que precisan un gran número de operaciones, funcionando como un coprocesador de procesos gráficos u operaciones de coma flotante (gráficos en videojuegos, simuladores o renderizado). Si la unidad central de procesamiento (CPU) de un ordenador convencional actual dispone de 4 a 8 núcleos de proceso, la unidad de procesamiento gráfico (GPU, *Graphics Processor Unit*) dispone de cientos o miles de núcleos de proceso. La arquitectura de las CPUs está orientada a la eficacia de las tareas del sistema operativo y, principalmente, al procesamiento secuencial mientras que las GPUs, aunque ejecutan tareas más básicas, poseen una arquitectura orientada a realizar operaciones en coma flotante, manejando múltiples tareas simultáneamente, en paralelo [1].

Número de iteraciones	CUDA® C++®		C/C++®		MATLAB® CUDA®		MATLAB®	
	Tiempo (ms)	Factor-X	Tiempo (ms)	Factor-X	Tiempo (s)	Factor-X	Tiempo (ms)	Factor-X
201	0,58 ± 0,002	1,00	2,25 ± 0,44	3,87	3,52 ± 0,11	6048,81	4,56 ± 1,89	7,84
601	0,92 ± 0,002	1,00	6,85 ± 0,37	7,46	9,87 ± 0,16	10754,34	10,73 ± 0,09	11,69
1101	1,37 ± 0,002	1,00	12,25 ± 0,44	8,91	17,97 ± 0,18	13077,19	18,81 ± 0,14	13,69
2501	1,38 ± 0,02	1,00	22,35 ± 0,49	16,17	43,71 ± 1,61	31621,25	34,44 ± 5,27	24,91
5001	1,39 ± 0,002	1,00	56,00 ± 0,46	40,16	101,80 ± 2,73	73013,90	52,82 ± 4,18	37,88
10001	2,69 ± 0,003	1,00	111,90 ± 0,55	41,55	297,06 ± 14,55	110292,64	86,12 ± 5,11	31,97
20001	4,03 ± 0,006	1,00	232,95 ± 6,39	57,74	764,86 ± 54,91	189591,03	171,45 ± 9,46	42,50
100001	17,50 ± 0,97	1,00	1233,35 ± 5,67	70,49	--	--	908,15 ± 96,65	51,90
500001	81,14 ± 0,23	1,00	7507,65 ± 60,91	92,52	--	--	4974,47 ± 248,12	61,31
1000000	165,67 ± 0,20	1,00	14687,80 ± 210,34	88,66	--	--	9527,56 ± 382,29	57,51

Tabla 1: Resultados del cálculo de la cinemática del mecanismo ferroviario

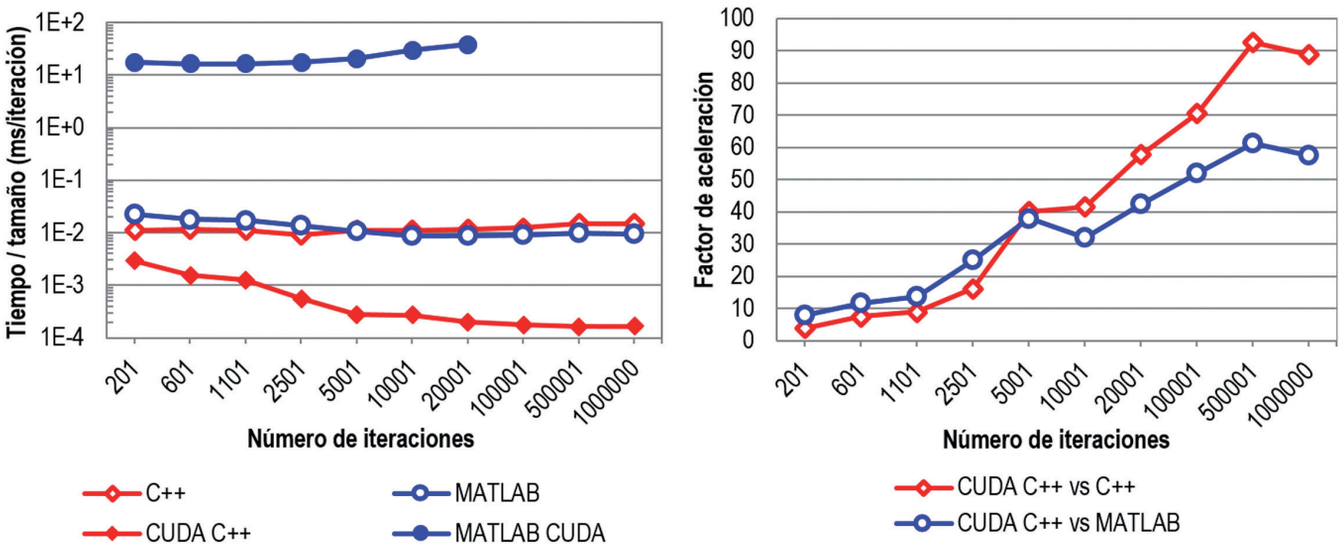

Fig. 1: Evolución de los tiempos de ejecución por iteración y del Factor de aceleración

En los últimos años, ingenieros de todos los ámbitos han utilizado las GPUs en sus investigaciones [2-3]. Siguiendo esta línea, en este trabajo se estudiará la viabilidad de aplicar la tecnología de procesamiento con GPU en diferentes procesos de análisis de un sistema mecánico. Se aplicará la tecnología de procesamiento con GPU, a través de la herramienta CUDA®, al cálculo de la cinemática y de la pseudo-optimización de un mecanismo ferroviario para controlar el movimiento de lazo. El análisis de los tiempos de computación necesarios para realizar los cálculos permitirá detectar las fortalezas y debilidades de la computación paralela con GPU [4].

Las ecuaciones del modelo cinemático se obtienen utilizando el método de Raven [5] y se implementan en diferentes lenguajes de programación: de forma secuencial en C++® y MATLAB® y de forma paralela en CUDA® C++® y MATLAB® con CUDA®. En el modelo de pseudo-optimización propuesto se aproxima la trayectoria del mecanismo a una trayectoria de referencia. El parámetro que evalúa el objetivo a conseguir es el SFA (*Scale Factor Amplitude*), basado en la distancia de Hausdorff [6]. El algoritmo que calcula el mejor índice SFA se desarrolla en dos versiones: una secuencial sobre MATLAB® y otra paralela sobre CUDA® C++®. Por último, se implementa un algoritmo mixto, combinación de los dos procesos (cálculo simultáneo del modelo cinemático y la pseudo-optimización).

Los resultados del modelo cinemático están recogidos en la Tabla 1 y muestran los tiempos de cálculo de los diferentes programas para un número de operaciones incremental. Los tiempos de cálculo menores se obtuvieron en CUDA® C++®. Además, se comprobó que el tiempo de cálculo por iteración en CUDA® C++® se reducía al aumentar el número de iteraciones por lo que el factor de aceleración aumentaba respecto al modelo cinemático implementado en C++® y en MATLAB®, alcanzándose reducciones temporales de hasta 60 veces respecto a MATLAB® y 90 veces respecto a C++® (Fig. 1). También se ha evidenciado que MATLAB® con CUDA® es la opción más lenta de todas.

Los tiempos de ejecución resultantes del cálculo del modelo de pseudo-optimización fueron similares en CUDA® C++® y MATLAB®. Esto se debe al efecto negativo que la constante transferencia de datos entre CPU y GPU repercute en la velocidad de cálculo.

Los resultados de los tiempos de ejecución del proceso mixto, combinación de los dos anteriores, mostraron que la resolución del algoritmo es del orden de 10 veces más rápido sobre CUDA® C++® que sobre MATLAB®.

En definitiva, el uso de la computación paralela con GPU puede ser una valiosa herramienta para reducir los costes de computación en la resolución de problemas de ingeniería que conlleven un gran número de repeticiones. Sin embargo, cuando el proceso de cálculo implica el intercambio frecuente de datos entre GPU y CPU disminuye su ventaja computacional.

Los autores agradecen al Gobierno español la financiación de este trabajo con el proyecto MAQ-STATUS DPI2015-69325-C2-1-R.

REFERENCIAS

[1] Ghorpade J, Parande J, Kulkarni M, et al. "GPGPU processing in CUDA architecture". Advanced Computing: An International Journal (ACIJ), 2012. Vol.3-1, p.105-120. DOI: http://dx.doi.org/10.5121/acij.2012.3109.

[2] Tasora A, Negrut D, Anitescu M. "GPU-based parallel computing for the simulation of complex multibody systems with unilateral and bilateral constraints: an overview" En: Arczewski K. Multibody Dynamics. Computational Methods and Applications. Varsovia: Springer, 2011. p.283-307.

[3] Garcia-Blas J, Abella M, Isailaa F, et al. "Surfing the optimization space of a multiple-GPU parallel implementation of a X-ray tomography reconstruction algorithm". Journal of Systems and Software. Septiembre 2014. Vol. 95 p. 166–175 DOI: http://dx.doi.org/10.1016/j.jss.2014.03.083.

[4] Bustos A, Rubio H, Corral E et al. "Parallel computing used to solve a railway mechanical system". DYNA New Technologies, Enero-Diciembre 2017, vol. 4, no. 1, p.[18 p.]. DOI: http://dx.doi.org/10.6036/NT8288

[5] Rubio H, Bustos A, Castejón C, et al. "Tool for the Analysis of New Skills Biped Pasibot". En: Petuya V, Pinto C, Lovasz EC (ed). New Advances in Mechanisms, Transmissions and Applications. Springer, 2014. Vol. 17, p. 173-181. ISBN 978-94-007-7484-1.

[6] Guerra C, Pascucci V. "Line-based object recognition using Hausdorff distance: from range images to molecular secondary structures". Image and Vision Computing. 2005. Vol. 23-4 p. 405-415. DOI: http://dx.doi.org/10.1016/j.imavis.2004.11.002.