Universidad De Granada

Grado ingeniería informática (2017 – 2018)

PROGRAMACIÓN PARALELA

Resumen de la Charla

Trabajo realizado por Antonio Miguel Morillo Chica.

Motivaciones por las que paralelizar.

Desde sus principios la programación secuencial en un único procesador era el orden del día en las computadoras de casa. El número de transistores y la frecuencia a la que estos trabajaban iban en aumento pero, tal y como predijo Moore con sus famosas leyes que se han estudiado en otras asignaturas llegariamos a un tope, es decir, el expacio es finito y el hardware no iba a poder dar más de si. No podemos ir sumando transistores y aumentando las frecuencias por problemas clásicos de la fisica, calor, espacio y rentabilidad económica.

El paralelismo ha surgido para solucionar parte de estos problemas a nivel industrial, además de otras soluciones "más toscas" como usar refrigerantes, ventiladores, etc.

Otro gran avance a sido el uso de trarjetas gráficas que han acelerado el número de operaciones en coma flotante además de poder hacer estas operaciones de forma paralela siendo entre 5 y 10 veces más eficientes que una CPU, hasta que una CPU multicore.

Por último otro gran avance hardware ha sido la velocidad de transmisión de los datos através del bus. Nuevas memorias, nuevos diseños que han aumentado de velocidades que llegaban a los 60GB/s a los 340GB/s. Esto es posible tambien, gracias al avance en la velocidad de cómputo. De nada nos servirían estas velocidades si no somos capaces de computar los datos.

Diseño de algoritmos paralelos.

Es muy importante unbuen conocimiento de un algoritmo y un buen conocimiento del hardware donde se va a ejecutar ese algoritmo ya que de esta forma estaremos en una situación más favorable para exprimir al algoritmo todo su potencial. Tendremos que hacer muchas consideraciones que se resumen en:

- Leyes de Amdahl's: Límite del speedup por el que podemos paralelizar.
- Leyes de Gustafson: Siempre existirán parte no paralelizables.
- La precisión de las operaciones es solo de 4bytes.
- Latencias impuestas e inevitables: Transmisión de datos por bus, por red..

- Mover un datos es demasiado caro.

Ejemplos de uso con multi–y–many–Cores.

Algunas de las formas de trabajar con multi–y–many–Cores es usando herramientas como:

- OpenGL: GPGPU

  Uno de los primeros lenguajes para la programación de GPU que usa primitivas para realizar calculos matematicos. El algoritmo algoritmo es simple, se basa en la multiplicación de matrices. Se pueden usar texturas.

- CUDA and mature programming languages

  CUDA resuelve muchos problemas de OpenGL, es una programación basada en C, muy rápido. Se ejecuta una grid de bloques para los pixel para realizar los carculos de cada bloque independiente aunque los hilos comunes al bloque pueden colaborar.

  Se ha vuelto una herramienta muy usada en el campo academico y AMD intenta competir pero no recibe mucha atención.

- High-level GPU with Jupyter

  - OpenCL: Una nueva herramienta capaz de ser usada en muchos más dispositivos. Es una API en C y con más versatilidad lo que ha obligado a CUDA a implemental soporte a templates. El problema de OpenCL es que hata un ejeplo simple necesita mucho código. Lo que ha provocado el uso de python con jupyter.