Resumen del Capítulo 1 del Libro “Programming Massively Parallel Processors”

Alumno: Patrick Xavier Márquez Choque

Cada vez la tecnología avanza ya que existe un aumento en la demanda por parte de los usuarios en mejoras de tiempo y rendimiento, esto provoca que se investiguen y desarrollen los métodos paralelos que como ya se estudio anteriormente son los que sobrepasan a las aplicaciones secuenciales pero existen ciertas dificultades, entre ellas es el hardware ya que el consumo y desperdicio de la energía trae problemas relacionados con la disipación de calor y los dispositivos actuales como las CPU’s y las GPU’s manifiestan estos problemas pero es aquí donde se menciona el término de Programación Paralela Heterogénea.

Se orienta este concepto para la utilización de ambas unidades de procesamiento tanto en CPU como en GPU, pero la larga brecha del rendimiento entre la ejecución paralela y secuencial se está superando poco a poco logrando una alternativa como el uso de las unidades de cómputo GPU para los cálculos intensivos dentro de una aplicación y mientras más trabajo es necesario realizar más veces se tendrá dividir el mismo.

Igualmente más problemas se acercan de manera abrupta como la complicada serie de instrucciones para realizar de manera paralela dentro de una CPU que está optimizada para código secuencial, complicaciones en caso de memorias caché ya que estas no contribuyen significativamente a la mejora del rendimiento entre los cálculos simultáneos y el limitado ancho de banda ya que las velocidades de las aplicaciones están limitadas por las velocidades en que los datos van a ser enviados entre la memoria a los procesadores.

Por lo tanto, se concluye que las GPU’s que están diseñadas y orientadas como unos motores paralelos para la mejora del rendimiento pero que no realizan muy bien algunas tareas de igual manera que las CPU’s por ejemplo en el caso de tareas de ejecución con un número pequeño de operaciones se recomienda la utilización de las CPU ya que en este caso tendrán mejor rendimiento que una GPU.

Aquí es donde es necesario mencionar que la programación paralela masiva se dio por la neta necesidad del aumento de la velocidad que demandan las aplicaciones y que con una buena distribución de estas implementaciones en paralelo con GPU se pueden alcanzar números de hasta 100 veces la aceleración frente a las ejecuciones secuenciales. Pero hay limitaciones dentro de estos métodos de manera similar como se vio con la ley de Amdahl y la ley de Gustafson donde se especifica que la mejora obtenida por el aumento de la velocidad y el rendimiento debido a la complejidad del mismo y la posibilidad de volverlo paralelizable dentro de alguna aplicación está limitada por las propiedades físicas de los propios dispositivos de tal manera que está aceleración no se puede expandir de manera indefinida y esto constituye un límite en todo aumento en la aceleración de la velocidad de las aplicaciones actuales, ya que aunque estas son las suficientemente rápidas siempre esta demanda conlleva una mejora continua pero tenemos que considerar este propio límite.

En un entorno más real con las limitaciones que se presentan tenemos que tener en cuenta que estas aceleraciones se pueden obtener haciendo que la mayoría del tiempo de una aplicación sea totalmente paralelizable pero esto es imposible ya que hay un límite entre las porciones de código de una aplicación se pueda volver paralelizable con procesadores que soporten esta lógica y que definitivamente, también existe un límite entre estas porciones ya que necesariamente existen partes que se tienen que hacer secuencialmente o que debido a su lógica estas serían muy complejas para pasarlas a un contexto paralelizable.

Dentro del libro nos menciona los 3 principales desafíos en la programación paralela, en primer lugar es debido a la complejidad mencionada para desarrollar algoritmos paralelos y sobre todo que se pueda asegurar una mejora en el rendimiento ya que se comprobó que debido al incorrecto diseño del algoritmo paralelo estos tienen incluso peor rendimiento que su contraparte secuencial; en segundo es el neto limite de la velocidad del acceso de memoria ya que como se mencionó anteriormente, estas limitaciones se definen por el número de instrucciones que se quieren realizar en relación con las cantidades de información que se necesitan y por último sería la propia dificultad de los programas paralelos ya que son estos que son más sensibles a los diseños entre los datos y las características de las aplicaciones para las que se utilizan, como por ejemplo un sistema de recuperación de software de tráfico de vehículos puede depender de los datos de entrada que los mismos puedan tener como las zonas geográficas pero de cierta manera los datos que son impredecibles o erráticos afectan en el rendimiento global de esta aplicación y es por eso que se tienen estos problemas dentro del mundo real.

Por último es mencionar que los lenguajes y modelos que existen dentro de la programación paralela son una respuesta para los problemas y son considerados como opciones para brindar un mejor soporte en la creación de código paralelo y son los que se vieron a lo largo de curso como MPI(Message Passing Interface) y OpenMP implementado para sistemas multiprocesador de memoria compartida. Dentro de OpenMP se creó una variación llamada OpenACC con el objetivo de proveer información a los programadores para la abstracción de algoritmos paralelos y tener mejores opciones para el desarrollo de los mismos. En el año 2009 varias de las empresas y marcas como Apple, Intel, AMD y NVIDIA desarrollaron un lenguaje como estándar para la programación paralela llamado OpenCL(Open Computing Language) que de cierta manera es similar a CUDA pero que se desarrollo de extensiones para el manejo del paralelismo, el envió de datos de manera masiva entre procesadores para alcanzar mejor rendimiento entre la programación paralela pero cabe destacar que las extensiones le brindan a los usuarios un entorno amable y familiar con el desarrollo de nuevas características aunque no enfocándose totalmente en que se puedan alcanzar mucha más aceleración en comparación a CUDA.