Computación heterogénea, el siguiente nivel

Avances y retos para la computación actual

Julian Eduardo Ayala Camacho

-Abstrac: [The development of this document is focused on studying and analyzing the different challenges agreed for heterogeneous computing, these processing based on performance and seek evoluvionar the concepts established throughout the history of computing. This study is carried out starting from the tools found in the market and the different types that can be accessed.]

Introducción

Al pasar los años, la humanidad se ha interesado por el desarrollo de tecnologías que le permitan tener una calidad de vida más alta, esto se refleja tanto en ámbitos comunicativos como la invención del teléfono fijo y celular, como los avances que se han presentado en la medicina los cuales permiten la detección de enfermedades a tiempo salvando y previendo muertes masivas. Este interés no es infundado, ya que desde los principios de la humanidad el ser humano se ha visto inmerso en el descubrimiento e invención de sistemas o herramientas que le ayudaran a sobrepasar los porvenires del nuevo mundo.

Esta reflexión se hace en pro de recalcar uno de los tópicos más importantes a tratar, el ultimo avance en computación el cual cambia el paradigma permitiendo subir las expectativas en cuanto a rendimiento y velocidad para este campo.

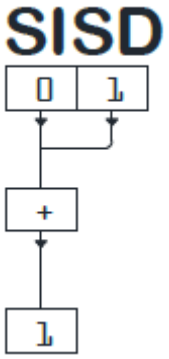
Hasta hace unos años la computación por sistema paralelo tenía el dominio de presencia ya que este sistema solo contaba con procesadores de un solo fabricante, limitando el rendimiento a un solo tipo de procesador aplicado varias veces, en cambio para el tema a tratar se tiene como principal ventaja sobre su predecesor, el uso permitido de diferentes tipos de procesadores los cuales, a través de su estudio detallado en este documento, combinan sus diferentes características para elevar el rendimiento de la maquina en cuestión y permitir la reducción de costos lógicos y físicos del sistema. Como toda nueva invención presenta unos retos a cumplir, heredados de la computación anterior, sumandos a los que se presentan mediante va avanzando la tecnología, estos retos presentes en su mayoría para rendimiento y procesamiento.

J.Ayala, Universidad Industrial de Santander (UIS), Santander, Colombia, jeac@gmail.com

I.Vista cronológica de computación heterogénea

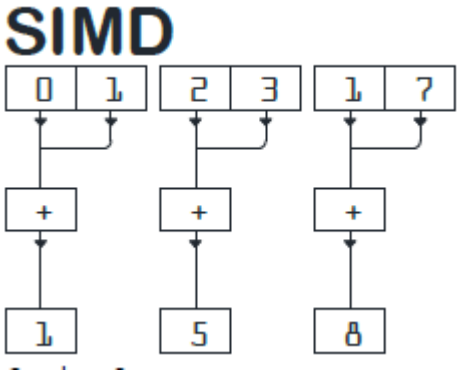
1.1 Después de observar tantos cambios evolutivos en la informática, nos parece poco impresionante estos sucesos que se presentan frecuentemente en nuestra época, pero para esta ocasión estudiaremos el ultimo avance presentado en computación, el cual propone el uso de diferentes arquitecturas de proceso en el mismo computador, este avance se denomina computación heterogénea.

Para tener un completo entendimiento del tema es necesario diferenciar entre computación paralela y heterogénea, en la cual para la paralela se utilizaban varias unidades centrales de procesamiento(CPU) con sistemas operativos y aplicaciones que permiten la operación en simultaneo.



Cuadro 2

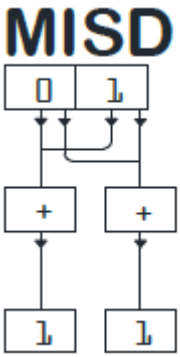
La arquitectura que se utilizaba mayormente por las primeras CPU´s se denominó SISD, la cual ejecuta una instrucción sobre un dato, está desarrollada con algoritmos y conocimientos recopilados en los años sesenta.
Para esta misma época se tiene registrada la aparición de los primeros computadores vectoriales los cuales incluían una CPU capaz de procesar varios datos a la vez. A continuación, se muestra un ejemplo de los primeros casos de arquitectura SIMD, la cual ejecuta una misma instrucción sobre varios datos simultáneamente.



Cuadro 1

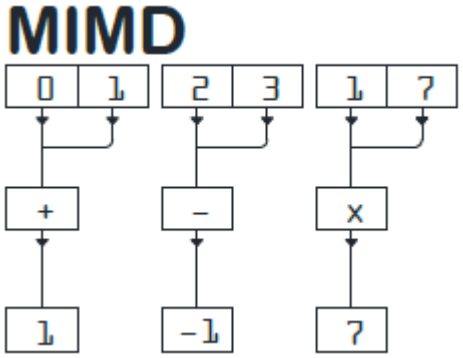
AL epasar vemos que la arquitectura menos usada en la computacion generalista, es la

MISD, en la cual varias instrucciones actúan en forma simultanea sobre un mismo dato, aunque es la menos usada está presente en sistemas de alta seguridad para comprobar las operaciones por redundancia.



Cuadro 3

Contraste a esto tenemos la arquitectura mas utilizada en la computacion generalista, esta se denomina MIMD en la cual, varias instrucciones actúan de forma simultanea sobre varios datos.



Cuadro 4

En esta se reflejan tanto en la topología de los más grandes sistemas de nodos presentes en supercomputadoras de hoy en dia, como a nivel de microprocesador multinucleo los cuales están presentes en nuestras Pc´s. Para estos microprocesadores mencionados se observa que estan conformados por varios nucleos de arquitectura SISD los cuales trabajan en paralelo y cuentan con acceso independiente a los datos, por lo que se podría decir que internamente la CPU es MIMD, siempre y cuando el acceso a la memoria suele ser atravez de una sola via por lo que hay una “serializacion” del sistema.

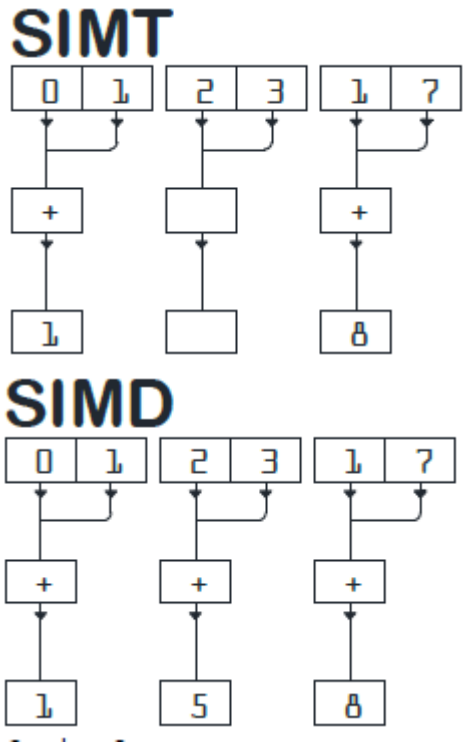
Despues de este breve repaso por las arquitecturas mas econtradas en el mercado, ninguna de ella se puede considerar computación heterogénea en el entido que actualmente se le da al término, esto se presenta porque tienen el mismo tipo de unidad de procesamiento general, son un sistema homogéneo.

Ahora pasamos a definir lo que son los sistemas de procesamiento grafico, GPU los cuales tienen sus inicios con Evan & Sutherland, donde se fusionaron con Silicon Graphics(SGI) cuyos diseños, el Reality Engine basado en Intel i860, fueron pioneros en la vectorizacion por hardware del tratamiento de graficos en 3D. OPEN GL el cual es de vital importancia para el desarrollo de las GPU estándar tiene sus orígenes en SGI y este ha contribuido a el modelado del

hardware utilizado para la búsqueda de la optimización.

II. En una visat al mercado de hoy en dia vemos que el desarrollo de estas tecnologías esta liderado por [3]NVIDIA y AMD, llamado antes [4]ATI. Para sus inicios ambos fabricantes comenzaron con GPUs muy enfocadas a la optimización de todo el proceso de bits, texturas, pixeles, etc.… todo esto para llegar a una arquitectura generalista la cual ha permitido aumentar la capacidad sin aumentar la complejidad.

La principal diferencia encontrada entre ambas empresas, NVIDIA y AMD, es el uso por parte de NVIDIA de arquitecturas SMIT, la cual consiste en una actuación simultanea de una instrucción sobre muchos “threads”, a esto se le dio el nombre de CUDA.
Esta diferencia se hace notable al observar la comparacion entre SIMD y SIMT



Esta diferencia se marca en que SIMT puede saltarse la ejecución en algunos de los procesos en un determiando paso, esto tiene grandes consecuencias para el programa permitiéndole la gestión de la divergencia en los procesos de forma muy similar a la tradicional o en serie la cual se presenta en la arquitectura SISD. Esta ventaja en la facilidad de la programación hace que las GPUs de NVIDIA sean el camino ganador hacia un buen rendimiento.

Con la información recolectada a travez de la realización de este documento, vemos que existen diferentes tipos de arquitecturas y que nuestro ordenador siempre utiliza dos de ellas, una dedicada a los procesos generales y la otra

para el procesamiento de tareas graficas.

El avance mas significativo que se puede resaltar, también con el cual se asumen los retos establecidos para esta computación, esta en la capacidad de poder usar la CPU de la tarjeta grafica (GPU) para los procesos generales, antes de esto se había generado un paralelismo con CPUs iguales, lo cual se cambió para hacer trabajar conjuntamente las CPUs de forma diferente, unas dedicadas al funcionamiento en serie, las CPUs, y las otras dedicadas al funcionamiento en paralelo, las GPUs.

A pesar de que internamente se cuentan con varios nucleos de procesamiento, aun consideramos la CPU como arquitectura en serie, esto se presenta como uno de los restos para esta computación ya que esta consideración se hace por las diferentes restricciones en la gestión de la memoria y porque su paradigma de programación es tradicionalmente la programación en serie. Estos procesadores generan paralelismo el cual viene de la ejecución simultanea de varios subprocesos o threads en serie que se van sincronizando entre ellos de maneras diferentes según sea el subproceso a tratar.

Una de las ventajas con la cual se asumen retos para esta computación es la capacidad de que tiene la GPU para procesar problemas con grandes series de datos, tanto enteros como de punto flotante. Esta ventaja le permite contar con ahorros en el tiempo de ejecución con hasta dos ordenes de magnitud, 100x incluso 400x, para los casos mas favorables.

La computación heterogénea brinda la capacidad al usuario de tener potencias de cálculo hasta ahora solo alcanzadas por supercomputadores, esto se debe por las excelentes relaciones entre precio-rendimiento y consumo-rendimiento de los actuales sistemas GPU y también por el enfoque generalista de las GPUs las cuales permiten su programación de una forma mas sencilla.

III. Ahora bien, basados en los conceptos fundamentales de la computación heterogenea los cuales radican en la diferencia entre programación en serie y paralelo vistos en el párrafo anterior y en la potencia de cálculo ofrecida por la GPU, veremos un ejemplo en cual muestra se la diferencia en el rendimiento de las programaciones en serie y paralelo generando una misma tarea para realizar de las dos maneras.

Suma de un vector con 4 elementos.

Suma en serie. Para la siguiente prueba se codifica un programa en C:

```
sumavectores( int *a, int *b, int *c, int n)
     for (i = 0 ; i < n ; i++)
          c[i] = a[i] + b[i];
     return;
};
```

La tarea comienza cuando el compilador envía a la CPU las instrucciones para realizarla, estas se muestran en la siguiente figura:

Cuadro 5

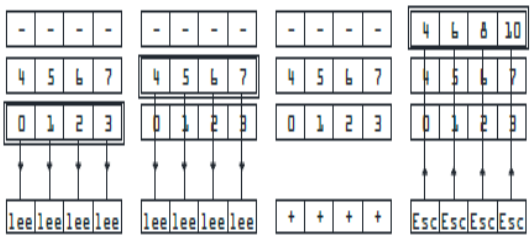
Despues de observar los resultados vemos que son necesarios 16 ciclos de trabajo, distribuidos en 8 de lectura, 4 de suma y 4 de escritura, los cuales completan la tarea asiganada.

Ahora bien probamos para sistemas en paraelo, para esto usamos un código codificado en CUDA C:

```
_global_ sumavectores( int *a, int *b, int *c){
     int i = threadIdx.x;
     c[i] = a[i] + b[i];
     return;
};
```

Revisando el código vemos que antes de que comience su ejecución, el programador se asegura de que cada thread ejecutado sobre un nucleo de la GPU le llegue una copia del programa, dejando al compilador encargado de la mayor parte de la ejecución de la tarea. La variable threadIdx.x es con la cual cada thread se identifica y con esta se accede a los distintos elementos del vector ordenando.

Al ver paso a paso los resultados obtenidos en el cuadro 6, vemos que se ejecutan cuatro ciclos de trabajo distribuidos en dos ciclos de lectura de datos, uno de suma y otro de escritura de datos.



Cuadro 6

Al comparar ambos resultados, tanto en serie como en paralelo

es fácil observar que la relación de mejora es bastante amplia pues lo que acabamos de aprender es extrapolable a conjuntos con mayor número de datos. Esta diferencia radica en la posibilidad de adecuar nuestros datos y procesos al procesamiento en paralelo aplicado a la segunda iteración.

En este ejemplo es claro observar uno de los retos pactados para el estudio de la computación heterogénea y es que el rendimiento hace parte importante de su surgimiento y principal razón de desarrollo.

Pero para tener un estudio total veremos cuáles son sus demás retos, características de estos y posibles soluciones.

- Eficiencia computacional:
  Este propone aprovechar eficazmente todo el hardware disponible. A pesar de la enorme capacidad de las maquinas estudiadas, mo es posible usar el 100% del hardware.
  Como vimos anteriormente las nuevas arquitecturas implican aplicar nuevos paradigams de programación los cuales no disponen de herramientas de desarrollo tan elaboradas como las de pre-acelerador.
  Para abordar este reto tomamos las siguientes consideraciones:
  - Entendemos que no todas las aplicaciones son adaptables a todas las plataformas disponibles
  - Es necesario el buen uso de las herramientas en los SDKs, como debuggers, profilers y manuales de buenas practicas
  - Utlizar las herramientas GPGPU como Ocelot, Par4All, rCuda, GPGPU, entre otros.

- Eficiencia energética: Esta prsenta como reto minimizar el cosumo manteniendo el rendimiento ya que el incremento de FLOPS implica mayor consumo eléctrico al igual que mayores emisiones de calor.
  Las siguientes tablas muestran los valores para medir la eficiencia: MFLOP/W (Lista Green500)

Green500 Rank	MFLOPS/W
1	2,499.44
2	2,351.10
3	2,142.77
4	2,121.71
5	2,102.12

Tabla.1

Computer*	Total Power (kW)
Beacon - Appro GreenBlade GB824M, Xeon E5-2670 8C 2.600GHz, Infiniband FDR, Intel Xeon Phi 5110P	44.89
SANAM - Adtech ESC4000/FDR G2, Xeon E5-2650 8C 2.000GHz, Infiniband FDR, AMD FirePro S10000	179.15
Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x	8,209.0
Todi - Cray XK7 , Opteron 6272 16C 2.100GHz, Cray Gemini interconnect, NVIDIA Tesla K20 Kepler	129.00
JUQUEEN - BlueGene/Q, Power BQC 16C 1.600GHz, Custom interconnect	1,970.0

Tabla 1.1

Para afrontar este reto se aprovecha de forma eficiente el hardware, esto se logra aplicando un acelerador el cual bien aprovechado genera relaciones *(coste/rendimiento) y (consumo/rendimiento)* muy favorables, estas mencionadas atrás en el documento. A esto se le suma la utilización de muchos nodos de bajo consumo los cuales se logran con la utilización de procesadores tipo Atom, ARM: tecnología diseñada para sistemas empotrados y móviles.

- Herramientas de desarrollo: En la actualidad la mayoría de los programas aceleardos con GPUs se basan en la plataforma de NVIDIA: CUDA.
  Esta plataforma nos ofrece, rendimiento, amplia comunidad de usuarios y un ecosistema de aplicaciones, pero el factor que se pretende mejorar son las limitaciones presentes en las tarjetas NVIDIA.

Para afrontar este reto se utiliza una herramienta llamada OpenCL

Esta herramienta contiene un estándar para el desarrollo de aplicaciones paralelas el cual es soportado en la mayoría de procesadores actuales, esta presencia en esta herramienta proporciona portabilidad al código aplicado.
Una de las soluciones propuestas por esta herramienta para los efectos negativos generados en el rendimiento se utiliza una opción que es llamada “tunear” la cual permite que el desarrollador optimice el código para hardware el cual lo vuelve mas eficiente.

La ultima herramienta confirmada por NVIDIA es CUDA 4.0, la cual se ha confirmado que está enfocada a la memoria, esto se observa en el avance clave de CUDA 4.0 que adopta un modelo de memoria plano. Este concepto hace la vida del programador mas sencilla y este sienta las bases para la optimización de bajo nivel en la gestión de memoria la cual genera mejoras en la velocidad.

IV. Conclusiones

Después de analizar la información encontrada sobre computación heterogénea y ver como su evolución ha estado ligada a la evolución de la tecnología, vemos que ha aumentado la presencia de fabricantes y tipos de hardware en el mercado, esto sumado a las nuevas aquitecturas cada vez mas complejas, las cuales generan nuevos retos para los que se tienen amplios recursos disponibles como herramientas, documentación y redes de usuarios disponibles para tratar los mismos.

Al asumir estos retos se tienen objetivos claros, el ahorro de tiempo entre proceso y calculo, el cual se basa en reducir el tiempo de procesamiento para una tarea y generar respuestas interactivas al usuario las cuales representan bajar los tiempos de cumplimiento de una tarea a tal punto que las respuestas sean inmediatas.

Referencias

- Anon, (2016). [online] Available at: http://www.famaf.unc.edu.ar/grupos/GPGPU/RIO2013/clase4.pdf [Accessed 20 Aug. 2016].
- Es.slideshare.net. (2016). *Computación Heterogénea: Aplicaciones y Modelado de Rendimiento*. [online] Available at: http://es.slideshare.net/unlopez/computacin-heterognea-aplicaciones-y-modelado-de-rendimiento [Accessed 20 Aug. 2016].
- Es.wikipedia.org. (2016). *Computación heterogénea*. [online] Available at: https://es.wikipedia.org/wiki/Computaci%C3%B3n_heterog%C3%A9nea [Accessed 20 Aug. 2016].
- Atc.ugr.es. (2016). [online] Available at: http://atc.ugr.es/pages/personal/propia/alberto_prieto/conferencias_pdfs/retos_ingeniaria_computadores/! [Accessed 20 Aug. 2016].
- Anon, (2016). [online] Available at: http://www.nlhpc.cl/wp-content/uploads/2012/09/3_APUs.pdf [Accessed 20 Aug. 2016].
- Cobo, J. (2013). *Computación Heterogénea (Potencia GPU+CPU) Heterogeneous Computing (GPU+CPU Power).* [online] ResearchGate. Available at: https://www.researchgate.net/publication/259295149_Computacion_Heterogenea_Potencia_GPUCPU_Heterogeneous_Computing_GPUCPU_Power [Accessed 20 Aug. 2016].