UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"Diseño de un Framework para la planificación de tareas preemptive en sistemas embebidos heterogéneos"

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

José Antonio Ayala Barbosa

DIRECTOR DE TESIS:

Dr. Paul Erick Méndez Monroy

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

Ciudad Universitaria, CDMX a Septiembre 2020

Índice general

Índice de Figuras

Índice de Tablas

Resumen

Capítulo 1

Introducción

Capítulo 2

Antecedentes

El objetivo de este capítulo es introducir los conceptos de: 1) sistemas en tiempo real; 2) tipos de ejecución de tareas; 3) el algoritmo por defecto de los sistemas en tiempo real 4) sistemas embebidos heterogéneos; 5) arquitecturas de hardware y software de tarjetas gráficas; y 6) cómputo de propósito general en en unidades de procesamiento de gráficos.

2.1. Sistemas en tiempo real

Los sistemas en tiempo real son sistemas de cómputo cuyas tareas deben actuar dentro de limitaciones de tiempo precisas ante eventos en su entorno. Por lo que el comportamiento del sistema depende, no solo del resultado del cálculo, sino también del momento (tiempo) en que se produce [1].

2.1.1. Tipos de ejecución de tareas

Existen dos tipo de ejecución de tareas, las ***preemptive***, donde es necesario interrumpir temporalmente una tarea que está realizando un sistema de cómputo, para darle la oportunidad a otra con mayor prioridad, con el compromiso de reanudar la rezagada más adelante, y las ***non-preemptive*** donde se requiere que termine la tarea actual para que posteriormente inicie una con mayor prioridad.

2.1.2. Algoritmos de planificación

Earliest Deadline First (EDF) es un algoritmo óptimo de planificación para sistemas de tiempo real, y acepta tareas en modo preemptive. Es un algoritmo muy extendido en sistemas en tiempo real debido a su optimalidad teórica en el campo no-preemptive, pero al momento de implementarlo en un planificador preemptive, el resultado puede acarrear un exceso de ejecución si se toma el peor caso [2]. Por ello es necesario buscar alternativas de algoritmos que tengan un mejor desempeño en tareas específicas.

2.2. CPU

La unidad de procesamiento central o CPU es un procesador de propósito general, lo que significa que puede hacer una variedad de cálculos, pero esta diseñado para realizar el procesamiento de información en serie, consta de pocos núcleos de propósito general. Aunque se pueden utilizar bibliotecas para realizar concurrencia y paralelismo, el hardware *per se* no tiene esa implementación.

2.3. GPU

La unidad de procesamiento gráfico o GPU es un procesador especializado para tareas que requieren de un alto grado de paralelismo. La tarjeta gráfica en su interior puede contener una cantidad de núcleos de un orden de cientos hasta miles de unidades que son más pequeñas y que por ende, individualmente realizan un menor número de operaciones. Esto hace que la GPU esté optimizada para procesar cantidades enormes de datos pero con programas más específicos[7]. Lo más común al utilizar la aceleración por GPU es ejecutar una misma instrucción a múltiples datos para aprovechar su arquitectura.

2.3.1. Manycore y Multicore

Es necesario destacar que los *manycore* y los *multicore* son utilizados para etiquetar a los CPU y los GPU, pero entre ellos existen diferencias. Un core de CPU es relativamente más

potente, está diseñado para realizar un control lógico muy complejo para buscar y optimizar la ejecución secuencial de programas. En cambio un core de GPU es más ligero y está optimizado para realizar tareas de paralelismo de datos como un control lógico simple enfocándose en la tasa de transferencia (*throughput*) de los programas paralelos. Con aplicaciones computacionales intensivas, las secciones del programa a menudo muestran una gran cantidad de paralelismo de datos. Las GPU se usan para acelerar la ejecución de esta porción código. Cuando un componente de hardware que está físicamente separado de la CPU y se utiliza para acelerar secciones computacionalmente intensivas de una aplicación, se le denomina acelerador de hardware. Se puede decir que las GPU son el ejemplo más común de un acelerador de hardware.

2.3.2. Arquitectura Pascal

2.3.2.1. Memoria unificada

La memoria unificada proporciona un único espacio de direcciones virtuales para la memoria de la CPU y GPU, permitiendo la migración transparente de datos entre los espacios de direcciones virtuales completos tanto de la tarjeta gráfica como del procesador. Esto simplifica la programación en GPUs y su portabilidad ya que no es necesario el preocuparse por administrar el intercambio de datos entre dos sistemas de memoria virtual diferentes[3].

2.3.2.2. Computación preemptive

Permite que las tareas de cómputo se reemplacen con granularidad a nivel de instrucción, en lugar de bloque de subprocesos, evitando el funcionamiento prolongado de aplicaciones que monopolizan el sistema y no dejan ejecutar terceras tareas. Obteniendo así, que las tareas puedan ejecutarse todo el tiempo que requieran ya sea para procesar grandes volúmenes de datos o qué esperen a que ocurran varias condiciones, mientras otras aplicaciones son computadas concurrentemente[3].

2.3.2.3. Balanceo de carga dinámico

En versiones anteriores de las tarjetas, la asignación de recursos en las colas de cálculos y de gráficos debía decidirse antes de la ejecución, por lo que, una vez que se lanzaba la tarea, no era posible reasignarla sobre la marcha. Un problema añadido que existía, era que si una de las colas se quedaba sin trabajo antes que la otra no podía iniciar un nuevo trabajo hasta que ambas colas terminen completamente[4]. La arquitectura Pascal introdujo el soporte para balanceo de carga dinámico [5], ayudando a la aceleración del cómputo de tareas asíncronas.

2.3.2.4. Operaciones atómicas

Las operaciones atómicas de memoria frecuentemente son importantes el cómputo de alto rendimiento ya que permiten que los hilos concurrentemente lean, escriban y modifiquen variables compartidas. La arquitectura Pascal nos permite realizar estas operaciones pero ahora con la ventaja de trabajar sobre memoria unificada.

2.3.3. GPGPU

El computo de proposito general en unidades de procesamiento de graficos o GPGPU es utilizado para acelerar el procesamiento realizado tradicionalmente por la CPU unicamente, donde la GPU actua como un coprocesador que puede aumentar la velocidad del trabajo [6].

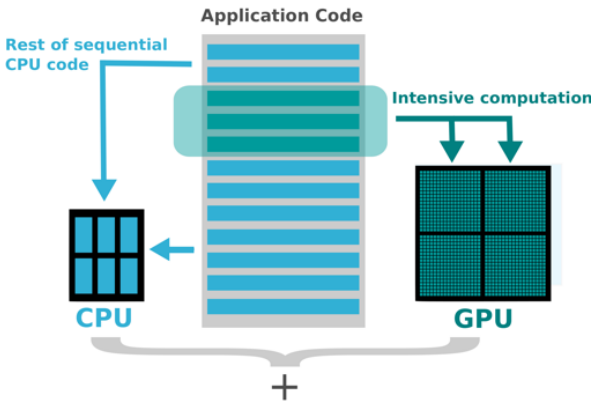

Figura 2.1. *Aceleración de programas en GPUs[7].*

La unificación de los espacios de memoria facilita el GPGPU ya que no hay necesidad de transferencias explícitas de memoria entre el host y el dispositivo.

2.4. Sistemas embebidos

Un sistema embebido es un sistema de cómputo diseñado para realizar tareas dedicadas, donde el mayor retos es realizar tareas específicas donde la mayoría de ellas tengan requerimientos de tiempo real [8].

2.4.1. Sistemas embebidos heterogéneos

En los últimos años los sistemas embebidos han ido demandando nuevas características debido a su rápida adopción en el mercado. Con lo que surge el desarrollo de sistemas embebidos heterogéneos, donde está contemplado realizar una gran cantidad de cómputo pero con una gran eficiencia tanto energética como en espacio.

Actualmente la empresa NVIDIA tiene en su catálogo sistemas embebidos heterogéneos con un gran soporte y bibliotecas para el cómputo de alto rendimiento. Dichos sistemas cuentan con la arquitectura pascal de última generación [9], la cual permite compartir memoria entre CPU y GPU.

2.4.1.1. Jetson TX2

Debido a que la mayoría de las GPU en sistemas embebidos no son de naturaleza preemptive, es importante programar los recursos de GPU de manera eficiente en múltiples tareas [10] ya sea de planificación o memoria, lo que permite pensar en un framework que ayude a la administración de sus características.

Caracteristicas del sistema

aa

	Descripción
Arquitectura	NVIDIA Pascal GPU
CPU	2 Denver 64-bit CPUs + Quad-Core A57 Complex
Memoria	8 GB L128 bit DDR4 Memory 32 GB eMMC 5.1 Flash Storage
	Connectivity to 802.11ac Wi-Fi and Bluetooth-Enabled Devices
	10/100/1000BASE-T Ethernet

Tabla 2.1. *Especificaciones del sistema.*

2.5. Resumen

Capítulo 3

Trabajo Relacionado

Anexos

Bibliografía

[1] G. C. Butazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications.* Springer Science Business Media, 2011.

[2] S. Heath, *Embedded systems design.* EDN Series For Design Engineers, 2003.

[3] NVIDIA, *NVIDIA Tesla P100 The Most Advanced Datacenter Accelerator Ever Built Featuring Pascal GP100, the World's Fastest GPU*, NVIDIA Corporation, 2016.

[4] R. Smith, "The nvidia geforce gtx 1080 gtx 1070 founders editions review: Kicking off the finfet generation." [Online]. Available: https://www.anandtech.com/show/10325/the-nvidia-geforce-gtx-1080-and-1070-founders-edition-review/9

[5] J. P. HURTADO V., "Análisis a fondo: Arquitectura gpu nvidia pascal – diseñada para la velocidad," 2016. [Online]. Available: https://www.ozeros.com/2016/05/analisis-a-fondo-arquitectura-gpu-nvidia-pascal-disenada-para-la-velocidad/

[6] NVIDIA, "Nvidia sobre la computación de gpu y la diferencia entre gpu y cpu," 2018. [Online]. Available: https://la.nvidia.com/object/what-is-gpu-computing-la.html

[7] ——, "Computación acelerada: Supera los desafíos más importantes del mundo." [Online]. Available: https://la.nvidia.com/object/what-is-gpu-computing-la.html

[8] M. Bertogna and S. Baruah, "Limited preemption edf scheduling of sporadic task systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 579–591, 2010.

[9] C. Hartmann and U. Margull, "Gpuart - an application-based limited preemptive gpu real-time scheduler for embedded systems," *Journal of Systems Architecture*, 2018.

[10] NVIDIA, "Nvidia jetson tx2: High performance ai at the edge." [Online]. Available: www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/