Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

CE-4303 Arquitectura de Computadores 1

Profesor: Ing. Luis Chavarría Zamora

Luis Pedro Morales Rodríguez - 2017089395

Taller 1

Investigación

1. Basado en este paper, explique y contraste las principales leyes de escalabilidad vigentes a la fecha (***apartado 1.1 Laws of Scalability***)

R/

1. **Ley de Amdahl (1967):**

Describe el comportamiento del aumento en la velocidad o rendimiento de un sistema computacional que se paraleliza parcialmente; es decir, que hay una porción de sus tareas que deben de ejecutarse secuencialmente. La ley se describe mediante la ecuación:

$$S_{thruput}(N) = \frac{N}{1-\sigma(N-1)},$$

donde $0 \leq \sigma \leq 1$ es la porción del tiempo invertido en las tareas que no pueden ser paralelizadas y ***N*** es la cantidad de unidades o hilos utilizados en la paralelización. Esta ley describe las limitaciones en el aumento del rendimiento paralelizado puesto a que la función se satura a un valor aproximado de σ^{-1} para valores de ***N*** "muy grandes" [1].

2. **Ley de Gustafson (1988):**

Esta ley tiene un enfoque mucho más optimista en el que se asume que el impacto de las tareas secuenciales $0 \leq \sigma \leq 1$ se reduce conforme se incrementa el valor de ***N,*** lo que describe un aumento del rendimiento ilimitado al añadir más unidades de procesamiento [1].

$$S_{thruput}(N) = N + (1 - N)\sigma,$$

3. **Ley de Gunther (1993):**

Se centra en la posibilidad de que, para sistemas grandes, los costos de coordinación entre unidades debido al aumento del tamaño del sistema puedan ser mayores que la mejora aportada por las unidades añadidas. El rendimiento de sistemas con un número de unidades mayor a un cierto valor crítico N_c, va a disminuir conforme se aumente N. Esto se describe por la ecuación de la Ley de Escalabilidad Universal (USL):

$$S_{thruput}(N) = \frac{N}{1 + \sigma(N-1) + kN(N-1)},$$

donde σ es ahora llamada contención (tiempo perdido en la cola por recursos compartidos) y el parámetro adicional κ considera el retraso de coherencia (sobrecostos de coordinar muchas unidades) [1].

2. Basado en el suite de punto flotante de SPEC (Standard Performance Evaluation Corporation), explique tres de esos benchmarks (indique lo que realiza y lo que mide o su propósito y objetivo)

R/

- ***503.bwaves_r***

Este benchmark está diseñado para aplicaciones en el área de CFD (Computational Fluid Dynamics). Simula numéricamente ondas de choque en un flujo laminar viscoso tridimensional transitorio. El algoritmo implementado es un solucionador no factorizado de las ecuaciones de Navier-Stokes utilizando el algoritmo Bi-CGstab, que resuelve sistemas de ecuaciones lineales no simétricas de manera iterativa [2].

La naturaleza transitoria del flujo y el solucionador iterativo hacen que bwaves sea un problema difícil de validar. En SPEC CPU®2017, esto se ha abordado comparando tres salidas diferentes [2]. Estas son:

1. La norma L2 del vector dq(l.i.j.k) después del último paso de tiempo.
2. El residuo para la convergencia después de cada paso de tiempo.
3. La suma acumulativa de iteraciones para la convergencia en cada paso de tiempo.

- **511.povray_r**

POV-Ray es una aplicación de trazado de rayos de código abierto y gratuita. El benchmark renderiza una imagen de 2560 x 2048 píxeles de un tablero de ajedrez, con las piezas ubicadas en la posición inicial. La imagen de la escena renderizada se guarda como un archivo Targa (.tga).

Esta imagen de salida se compara con la imagen de referencia utilizando la utilidad *imagevalidate* de SPEC®, que calcula el índice de similitud estructural (SSIM) entre grupos correspondientes de píxeles de 8x8 en cada imagen. El índice SSIM tiene un rango de -1 (diferencia máxima) a 1 (idénticas). El benchmark requiere que todos los índices SSIM calculados sean mayores que 0.996 para que la ejecución se considere exitosa [3].

- ***544.nab_r***

544.nab_r se basa en Nucleic Acid Builder (NAB), que es una aplicación de modelado molecular que realiza cálculos intensivos en coma flotante que ocurren comúnmente en computación de ciencias naturales. Los cálculos van desde "dinámica molecular" relativamente no estructurada hasta álgebra lineal relativamente estructurada.

La entrada de 544.nab_r consiste en moléculas con diferentes números de átomos. Para simular una molécula, se requieren dos archivos. Uno es un archivo de texto en formato PDB que especifica las coordenadas atómicas (x, y, z). El otro es un archivo de texto en formato PRM que especifica, entre otras características, el "campo de fuerza" que indica cómo interactúan los átomos entre sí. Las salidas son resúmenes del estado de la simulación de dinámica molecular cada 100 iteraciones y al final [4].

3. Explique en qué consiste el benchmark de CoreMark. Interprete los resultados de la Tabla 3 de este enlace.

R/

El benchmark CoreMark® de EEMBC mide el rendimiento de microcontroladores (MCUs) y unidades centrales de procesamiento (CPUs) utilizados en sistemas integrados. CoreMark contiene implementaciones de los siguientes algoritmos: procesamiento de listas (buscar y ordenar), manipulación de matrices (operaciones comunes de matrices), máquina de estados (determinar si una secuencia de entrada contiene números válidos) y CRC (comprobación de redundancia cíclica). Está diseñado para ejecutarse en dispositivos desde microcontroladores de 8 bits hasta microprocesadores

de 64 bits. Al ejecutar CoreMark se obtiene un puntaje único que permite a los usuarios realizar comparaciones rápidas entre procesadores [5].

Intepretando los resultados de la tabla, para CoreMark con 1 iteración, se ejecutan 306242 instrucciones RISC-V en la iteración del programa RISC-V compilado por GNU GCC, mientras que se ejecutan 442604 instrucciones ARMv6-M en el programa compilado por ARMCC. Si la instrucción ARMv6-M se interpreta directamente como una instrucción RISC-V, se requieren 1963327 instrucciones RISC-V, y en promedio, una instrucción ARMv6-M requiere 4.44 instrucciones RISC-V para compilarse. Sin embargo, después de optimizar las banderas, 442604 instrucciones ARMv6-M requieren 654065 instrucciones RISC-V, y en promedio, cada instrucción ARMv6-M requiere 1.48 instrucciones RISC-V. Finalmente, una instrucción ARM requiere tan solo 1.11 instrucciones RISC-V después de optimizar las banderas e instrucciones de salto, disminuyendo el conteo de instrucciones a 492766.

Referencias

[1] H. Hamann and A. Reina, “Scalability in Computing and Robotics,” 2021.

[2] “503.bwaves_r.” https://www.spec.org/cpu2017/Docs/benchmarks/503.bwaves_r.html (accessed Aug. 14, 2023).

[3] “511.povray_r.” https://www.spec.org/cpu2017/Docs/benchmarks/511.povray_r.html (accessed Aug. 14, 2023).

[4] “544.nab_r.” https://www.spec.org/cpu2017/Docs/benchmarks/544.nab_r.html (accessed Aug. 14, 2023).

[5] “CPU Benchmark – MCU Benchmark – CoreMark – EEMBC Embedded Microprocessor Benchmark Consortium.” https://www.eembc.org/coremark/ (accessed Aug. 14, 2023).