Paralelismo - 23GIIN

Actividad 3 – Portafolio

Analisis de rendimiento de arquitecturas y aplicaciones paralelas

Gagliardo Miguel Angel

17 de Diciembre de 2023

1) Investigue, baje, compile y ejecute al menos un benchmark para linux (o windows, como última opción) y evalúe el desempeño de varios ordenadores:

- Describe qué hace el benchmark
- Explique y muestre las métricas que evalúa el benchmark (uso de CPU, MIPS, FLOPS, memoria, etc.) y muestre una tabla que compare ambos ordenadores (o en el mismo ordenador en varios sistemas operativos) de acuerdo a esas métricas

El programa o benchmark elegido es **linkpack,** que es es un benchmark que mide la performance de una computadora a traves de un problema de algebra lineal simple. Configura una matriz aleatoria **A** de tamaño N = 1000, y un vector **B** del lado derecho que es el producto de **A** y un vector **X** de todos los 1:

- La primer tarea consiste en calcular una factoriuzacion LU de **A**
- La segunda tarea es usar la factorizacion LU para resolver el sistema lineal:
  - **A * X = B**

El mismo se puede descargar de:

- Codigo: https://shorturl.at/hrHM9
- Script: https://shorturl.at/bjBS4

El numero de operaciones de coma flotante necesarias para estas dos tareas es aproximadamente: **Operaciones = 2 * N * N * N / 3 + 2 * N * N**.

Por lo tanto, la calificacion “Mflops”, MegaFlops o millones de operaciones de punto flotante por segundo, se puede encontrar como: **MFlops = Operaciones / (CPU * 1.000.000)**

En una computadora determinada, si ejecuta el punto de referencia para una secuencia de valores crecientes de N, el comportamiento de la clasificación MegaFLOPS variará a medida que pase por tres zonas principales de comportamiento:

- Una zona “ascendente”, ya que ni la memoria caché local ni el procesador se ven afectados.
- Una zona “plana”, donde el procesador se ve desafiado, es decir, que está funcionando al máximo rendimiento.
- Una zona de “decadencia”, donde la memoria caché local también se ve desafiada, es decir, la matriz es tan grande que la memoria caché no es lo suficientemente grande como para mantener los datos necesarios lo suficientemente cerca del procesador para que siga funcionando a la máxima velocidad.

Para instalarlo, podemos ejecutar el script mencionado mas arriba, aunque como podemos ver, no es nada mas que una compilacion utilizando GCC:

```
x-1 ~/Documents/VIU/03 - Tercero/23GIIN - Paralelismo/UC3 [master L|+ 343…5
16:00 $ cat linpack_bench.sh
#! /bin/bash
#
gcc -c -Wall linpack_bench.c
if [ $? -ne 0 ]; then
  echo "Compile error."
  exit
fi
#
gcc linpack_bench.o -lm
if [ $? -ne 0 ]; then
  echo "Load error."
  exit
fi
#
rm linpack_bench.o
#
chmod ugo+x a.out
mv a.out ~/binc/linpack_bench
#
echo "Normal end of execution."
```

En mi caso lo hice manualmente, y luego ejecutando el archivo compilado **a.out**:

```
✔ ~/Documents/VIU/03 - Tercero/23GIIN - Paralelismo/UC3 [master L|+ 343…5]
16:00 $ gcc -c -Wall linpack_bench.c
✔ ~/Documents/VIU/03 - Tercero/23GIIN - Paralelismo/UC3 [master L|+ 343…5]
16:00 $ gcc linpack_bench.o -lm
✔ ~/Documents/VIU/03 - Tercero/23GIIN - Paralelismo/UC3 [master L|+ 343…5]
16:01 $ ./a.out
16 December 2023 04:01:13 PM

LINPACK_BENCH
  C version

  The LINPACK benchmark.
  Language: C
  Datatype: Double precision real
  Matrix order N              = 1000
  Leading matrix dimension LDA = 1001

     Norm. Resid      Resid           MACHEP         X[1]          X[N]

        6.491510        0.000000    2.220446e-16        1.000000        1.000000

      Factor     Solve      Total     MFLOPS       Unit      Cray-Ratio

   0.497516   0.001647   0.499163  1339.575783   0.001493   8.913625

LINPACK_BENCH
  Normal end of execution.

16 December 2023 04:01:14 PM
```

Lamentablemente no pude testearlo en varios sistemas operativos en el mismo computador dado que no utilizo windows, solo Linux.

2) Sea un conjunto de N= 1000 operaciones a distribuir para ser realizadas entre p = 10 procesadores. El tiempo promedio para realizar cada operación puede ser estimado en t=1; el tiempo mínimo t_{min}= 0.5 y el tiempo máximo t_{max} = 2:

(a) Suponiendo balance de carga estático, es decir que cada procesador ya tiene acceso a sus operaciones (no hay comunicación para repartir la carga, ni transmitir resultados), calcule el tiempo total de ejecución paralela y la aceleración en el caso más desfavorable.

(b) Suponiendo un balance de carga dinámico perfecto, calcule el tiempo total de ejecución paralela y la aceleración. Para esto considere que la búsqueda de trabajo por parte de los procesadores les insume un tiempo (sobrecarga) tg= 0.1 y que el trabajo les es entregado por paquetes (chunks) de 2 operaciones.

Para resolver el punto **a)**, al considerar **el caso mas desfavorable** se tiene que el tiempo para realizar cada operacion sera el tiempo maximo: $t_1 = t_{1max} = 2$.

Ademas, si son n = 1000 las operaciones a distribuir entre p = 10 procesadores, cada procesador realizara $n_1 = 100$ operaciones, con lo cual **el tiempo total de ejecucion paralela** sera $t_p = n_1 * t_1 = 100 * 2 = 200$.

Para calcular la aceleracion se emplea la ecuacion: $A = \frac{tp}{ts}$

Donde t_p es el tiempo de procesamiento en paralelo calculado, y t_s el tiempo de procesamiento secuencial, el cual seria igual al numero de operaciones, por tanto: n = 1000 = t_s = 1000, dado que serian todas las operaciones ejecutadas por un 1 unico procesador.

Por lo tanto, la aceleracion sera: $A = \frac{ts}{tp} = \frac{1000}{200} = 5$

En el punto **b)** debe realizarse el mismo analisis, considerando en este caso balance de carga estático. La diferencia fundamental es que el tiempo de operacion de cada procesador no sera el tiempo maximo asumido para el punto anterior, sino que podra considerarse el tiempo promedio t_1.

Sabemos por lo mencionado antes, que cada procesador realizara n_1 = 100 operaciones en tandas de a 2 (chunks) y que cada busqueda de datos le requiere al ordenador un tiempo t_g = 0.1

Con estos datos se adopta la siguiente expresion para calcular el tiempo de procesamiento en paralelo: $tp = \frac{tg * n1}{chunks} + n1 * t1 = 150$

Dado que el tiempo secuencial no depende de los cambios introducidos en el punto **b)** se tendra que **ts = 1000** y la aceleracion sera: $A = \frac{ts}{tp} = \frac{1000}{150} = 9.52$

3) Considere un algoritmo para sumar 128 números reales (ai, i=1, ... 128). Suponiendo que una suma consume 1ms y que la transferencia de datos (hasta 16 números reales) consume 1ms, calcule la aceleración y la eficiencia para:

(a) una estrategia maestro-esclavo (el maestro participa en las sumas parciales) y

(b) una estrategia jerárquica.

Elabora una tabla para mostrar la aceleración y la eficiencia para 1, 2, 4, 8 y 16 procesadores para cada estrategia de paralelización. Intente generalizar el modelo para el cálculo de la aceleración y la eficiencia para cada estrategia de paralelización.

Antes de pasar en limpio las tablas, un par de comentarios sobre los calculos considerados aqui:

1. Tiempo secuencial: t_s = 127; dado que son 127 sumas las que se ejecutan en 1 solo procesador, vale tanto para (a) como para (b)
2. Cantidad de operaciones: M = 128; En nuestro caso, cantidad de sumas
3. Cantidad de procesadores: P = {1, 2, 4, 8, 16}; Segun corresponda
4. Aceleracion: $A_c = \frac{Ts}{Tp}$
5. Eficiencia = $E_p = \frac{Ac}{P}$
6. Costo Computacional: $Fp = \frac{Ep}{Tp} = \frac{Ep * Ac}{Ts}$

(a) Para estrategia maestro-esclavo:

Procesos (M)	Procesadores (P)	T_{serial}	T_{comm}	T_{comp}	$T_{paralelo}$	Aceleracion (A)	Eficiencia (E_p)	Costo Computacional (F_p)
128	1	127	0	127	127	1	1	0.01
128	2	127	4	64	69	1.84	0.92	0.01
128	4	127	2	34	37	3.43	0.86	0.02
128	8	127	1	22	24	5.29	0.66	0.03
128	16	127	1	22	24	5.29	0.33	0.01

Teniendo en cuenta que:

- $t_p = t_{comp} + t_{comm} = (\frac{M}{P}) - 1 + P - 1 + P * Tcomm + 1$
- $t_{comp} = \frac{M}{P} - 1 + P - 1$
- $t_{comm} = \frac{\frac{M}{P}}{16}$; Dividimos por 16 dado que es la cantidad de numeros reales a transferir

(b) Para estrategia jerarquica:

Procesos (M)	Procesadores (P)	T_{serial}	T_{comm}	T_{comp}	$T_{paralelo}$	Aceleracion (A)	Eficiencia (E_p)	Costo Computacional (F_p)
128	1	127	0.00	127.00	127.00	1.00	1.00	0.01
128	2	127	1.00	63.30	65.30	1.94	0.97	0.01
128	4	127	2.00	31.60	34.60	3.67	0.92	0.03
128	8	127	3.00	15.90	19.90	6.38	0.80	0.04
128	16	127	4.00	8.20	13.20	9.62	0.60	0.05

Teniendo en cuenta que:

- $t_p = t_{comp} + t_{comm} + 1 = (\frac{M}{P}) - 1 + \log(P) + Tcomm + 1$
- $t_{comp} = \frac{M}{P}$ - 1 + log(P)
- $t_{comm} = \log_2(P)$

4) Suponga que un programa presenta una fracción secuencial que toma 1 seg y una parte paralelizable que toma 1000 segs en un procesador. Calcule la aceleración y la eficiencia si el programa se ejecuta con: (a) 100 procesadores; (b) 500 procesadores.

Parte paralelizable: **1000/1001 = 0.9999**

Parte secuencial: **1/1001 = 0.0009**

Segun Ley de Amdahl: $A = \frac{p}{(1+(p-1)*f)}$

Por lo tanto:

- Con 100 procesadores: $A(100) = \frac{100}{(1+(100-1)*0.0009)} = \frac{100}{(1+99*0.0009)} = 91.82$
- Con 500 procesadores:

$$A(500) = \frac{500}{(1+(500-1)*0.0009)} = \frac{500}{(1+499*0.0009)} = 345.04$$

REFERENCIAS

- Manual de la asignatura 23GIIN – Paralelismo: https://shorturl.at/coFLW

- Linpack Benchmark: https://shorturl.at/mqtLW