Operaciones basicas matriciales

Wilson Lopez Rubi, Oscar Vega Morua
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Arquitectura de computadores
I semestre 2018

***Abstract*—The main objective of this project is to run a parallel program in a cluster Kabre with different configurations of nodes, to know the structure of a super computer through a simple example programmed with openmpi.**

1. Introduction

El proyecto tiene como objetivo el desarrollo de un simple programa paralelizado con la biblioteca Openmpi , este programa se debe ejecutar en el cluster del Cenat , denominado Kabré con la configuración de 2,3,4,5 nodos y de esta forma realizar una comparación entre los tiempos de ejecución del programa y entender la diferencia entre los equipos de trabajo que normalmente se utilizan con respecto a una supercomputadora. Para el desarrollo del programa paralelizado se decide crear un pequeño programa en C++ que realiza el calculo y operaciones básicas entre matrices (suma , resta y multiplicación)

2. Descripción de equipo

En esta sección detallamos las características de el equipo de trabajo que normalmente utiliza una persona , esto es un una computadora personal con especificaciones estándares del mercado.

- Laptop DELL 5559
- RAM: 16GB DDR3 1.6Ghz/s
- CPU:Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
- GPU: Radeon HD R5 M330
- Sistema Operativo:Ubunto 17.10

3. Selección del programa

Es este apartado se detallara el motivo de la selección del programa , las matrices son una de las estructuras mas importantes en la computación , sin embargo el recorrer una matriz es una operación bastante cara para el procesador , pues en la mayoría de los lenguajes para poder recorrer una matriz se debe realizar un doble ciclo , uno para recorrer las filas y otro para recorrer las columnas , a pesar de que las computadoras pueden realizar estas funciones un tanto rápido , cuando el tamaño de las matrices aumenta el peso con el que debe luchar el procesador aumenta también por lo que el tiempo que requiere para procesar esas tareas es algo ,se considera que el utilizar este tipo de tareas tan pesadas es la manera mas adecuada de ver las diferencias entre tiempo que podemos tener entre el equipo de trabajo y el cluster.

4. Descripción del programa

El programa genera dos matrices de tamaño 500x500 de manera incremental , es decir la matriz va insertando los números de forma ordenada y con un contador, posteriormente se realiza una suma , una resta y una multiplicación entre las las dos matrices generadas y calculamos el tiempo de ejecución de cada una de las operaciones.

5. Ley Gustafson

La ley de Gustafson es una ley en ciencia de la computación que establece que cualquier problema suficientemente grande puede ser eficientemente paralelizado. La ley de Gustafson está muy ligada a la ley de Amdahl, que pone límite a la mejora que se puede obtener gracias a la paralelización, dado un conjunto de datos de tamaño fijo, ofreciendo así una visión pesimista del procesamiento paralelo

6. Ley Moore

La ley de Moore establece que la velocidad del procesador o el poder de procesamiento total de las computadoras se duplica cada doce meses.

7. Ley Amdahl

La ley de amdahl establece que La mejora obtenida en el rendimiento de un sistema debido a la alteración de uno de sus componentes está limitada por la fracción de tiempo que se utiliza dicho componente. Y establece la siguiente formula:

$$T_m = T_a \cdot \left((1 - F_m) + \frac{F_m}{A_m} \right)$$

Figure 1. Ley de amdhal

8. OpenMPI

OpenMPI es una implementación más de “caso común” del estándar MPI, tanto en términos de uso y soporte de red. Aún así, OpenMPI es una implementación de alta calidad del estándar, que tiene contribuidores como AMD, Nvidia, Cisco Systems, entre otros. Además, el administrador de procesos de OpenMPI (ORTE de Open Run-Time Environment) es bastante bueno y no tiene debilidades.

9. Elección del lenguaje

A pesar de que C es un lenguaje bastante robusto y compatible con la biblioteca de openMPI , se decidió realizar el programa en C++ debido a que tenemos mas conocimiento de ese lenguaje y mas experiencia.

10. Lecturas

En la presente se presentan las tablas de las lecturas realizadas durante la ejecución del programa en el cluster con la respectiva cantidad de nodos, cabe destacar que todas las medidas se realizaron en milisegundos para facilitar la compresion las lecturas.

10.1. Local

En esta sección podemos ver las medidas del programa corriendo de manera local en nuestro equipo de trabajo , podemos ver que el tiempo de ejecución es altamente mayor que el que se registra en las ejecuciones paralelizadas.

Local	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5	Corrida 6	Corrida 7	Corrida 8	Corrida 9	Corrida 10	Promedio
Suma	15	18	19	17	18	11	17	19	18	18	17
Resta	3	6	6	6	5	4	7	8	8	6	5.9
Multiplicacion	773	857	776	777	794	780	782	780	795	780	789.4

Figure 2. Local

10.2. 2 Nodos

Como en todos los demás casos la multiplicación siempre es la operación que toma mas tiempo pues es la mas tediosa y la resta la que menos , en este caso la resta toma un promedio muy estándar de 3 milisegundos y la mayor variación la encontramos en la multiplicación.

2 Nodos	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5	Corrida 6	Corrida 7	Corrida 8	Corrida 9	Corrida 10	Promedio
Suma	5	4	4	5	4	4	5	4	5	4	4.4
Resta	3	3	3	3	3	3	3	3	3	3	3
Multiplicacion	86	86	88	82	82	83	86	82	87	86	84.8

Figure 3. 2 nodos

10.3. 3 Nodos

Con tres nodos el patrón no cambia pero vemos como los tiempos se reducen sin embargo no de una manera muy significativa.

3 Nodos	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5	Corrida 6	Corrida 7	Corrida 8	Corrida 9	Corrida 10	Promedio
Suma	3	4	3	3	3	3	3	4	3	3	3.2
Resta	2	2	2	2	2	2	2	2	2	2	2
Multiplicacion	76	84	78	81	79	81	80	82	79	77	79.7

Figure 4. 3 nodos

10.4. 4 Nodos

En esta tabla se ven las ejecuciones con 4 nodos , estas son mas rápido que todos sus anteriores , de 3 , 2 y local , sin embargo podemos notar que con la suma el cambio es mínimo con respecto a su anterior.

4 Nodos	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5	Corrida 6	Corrida 7	Corrida 8	Corrida 9	Corrida 10	Promedio
Suma	3	4	3	5	3	4	3	3	4	3	3.5
Resta	1	1	1	1	1	1	1	1	1	1	1
Multiplicacion	75	75	68	73	75	74	75	69	74	76	73.4

Figure 5. 4 nodos

10.5. 5 Nodos

Esta es la configuración mas rápida que se probo , muestra una gran mejora con respecto a todas las demás configuraciones.

5 Nodos	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5	Corrida 6	Corrida 7	Corrida 8	Corrida 9	Corrida 10	Promedio
Suma	2	3	3	4	3	2	3	4	4	3	3.1
Resta	1	1	1	1	1	1	1	1	1	1	1
Multiplicacion	63	69	62	60	64	61	64	61	60	67	63.1

Figure 6. 5 nodos

11. Análisis general

Esto es una comparación entre el promedio de todas las ejecuciones y la cantidad de nodos que empleamos , como podemos ver la velocidad de ejecución se disminuye entre mas paralelizado esta la ejecución esto refleja el poder de procesamiento de las súper computadoras y como estas puede ser muy útiles y necesarias para tareas muy complejas y pesadas.

	Loca	2 Nodos	3 Nodos	4 Nodos	5 Nodos
Suma	17	4.4	3.2	3.5	3.1
Resta	5.9	3	2	1	1
Multiplicacion	789.4	84.8	79.7	73.4	63.1

Figure 7. Promedio general de ejecuciones

12. Conclusiones

- Después de todas las ejecuciones podemos concluir que la ejecución con mas nodos en el cluster es la que da mejor rendimiento , pues registra tiempos de ejecución bastante cortos.
- La operación de resta de matrices es la que le toma menor tiempo de ejecución al sistema pues en todas las tablas registra un muy corto periodo de ejecución.
- No todo tipo de código es paralelizable solamente algunos que requieren de mucho uso del procesador , un ejemplo de ello es la iteración.

- EL procesamiento de matrices es una operación bastante demandante para el procesador pues requiere recorrer los datos muchas veces.
- Algunos procesos pequeños son mas fáciles y rápidos de realizar en un equipo de uso cotidiano , pues el tiempo que toma dividir el proceso y posteriormente volverlo a unir toma mas que el mismo proceso en como tal.

References

Millot, D., Muller, A., Parrot, C., Silber-Chaussumier, F. (2009, September). From OpenMP to MPI: first experiments of the STEP source-to-source transformation tool. In ParCO 2009: International Conference on Parallel Computing: Mini-Symposium" Parallel Programming Tools for Multi-core Architectures" (pp. 669-676). IOS Press.

Chien, A. A., Reddy, U. S., Plevyak, J., Dolby, J. (1996, March). ICC++-a C++ dialect for high performance parallel computing. In International Symposium on Object Technologies for Advanced Software (pp. 76-95). Springer, Berlin, Heidelberg.