CS112 Midiendo Escalabilidad en Supercomputadores

Fabrizio Vásquez, Isaac Trinidad, Aguirre Alvaro, Medina Eduardo
Departamento de Computer Science, UTEC
Lima, Perú

***Resumen*—Las supercomputadoras hoy en día permiten realizar predicciones más exactas, como en la economía , climatología , cosmología entre otras muchas ramas del ámbito científico. Sin embargo, el software que desarrolla muchas de estas predicciones puede que no sea el mejor. En ese mismo sentido, el presente documento tratará de evaluar distintos resultados de diversos códigos para evaluar su escalabilidad a fin de brindar un mejor contexto en la toma decisiones en el uso en computadores de alto rendimiento.**

I. INTRODUCCION

LA computación de alto rendimiento (High Performance Computing) es una práctica que implica la utilización de poder computacional agregado para resolver problemas complejos que requieran un rendimiento mucho mayor al que podría ofrecer una computadora de hogar u oficina. La solución de problemas a gran escala en áreas como la ciencia e ingeniería se debe a las tecnologías computacionales que se emplean en esta práctica, entre las cuales se encuentran el uso de clusters y la computación en paralelo. Para la solución de dichos problemas hoy en día no basta con poseer un ordenardor que alcanze el grado de Petaflops, sino que el software desarrollado para su ejecución , debe ser igual de eficiente que su contraparte física, proporcionando un grado de escalabilidad, es decir que puede adaptarse frente a cambios en la carga que puede soportar, como lo son: millones de usuarios realizando peticiones a un servidor, control de tráfico o problemas que conllevan grandes cálculos como simulaciones en el campo de la astrofísica.

II. OBJETIVOS

En el presente proyecto se plantearon los siguientes objetivos:

- Proponer y desarrollar un método de estandarización para la leída de información.
- Investigar algún método para medir la eficiencia y error empleando los resultados provistos por la ejecución de códigos de terceros.
- Indicar el grado de escalabilidad y error en los resultados de los diferentes códigos provistos y una posible interpretación.

III. MARCO TEÓRICO

En el desarrollo e implementación del software se utilizaron dos conceptos para determinar cuan escalable son los códigos, estos son: Escalamiento fuerte , eficiencia de escalamiento fuerte, desviación Estándar.

III-A. ESCALAMIENTO FUERTE (Strong Scaling)

En el caso de los resultados provistos, el tamaño del problema permanece fijo pero aumenta el número de elementos de procesamiento. Esto se usa como justificación para los programas que tardan mucho en ejecutarse . En una escala fuerte, se considera que un programa escala linealmente si el speed-up es igual al número de elementos de procesamiento utilizados [3].

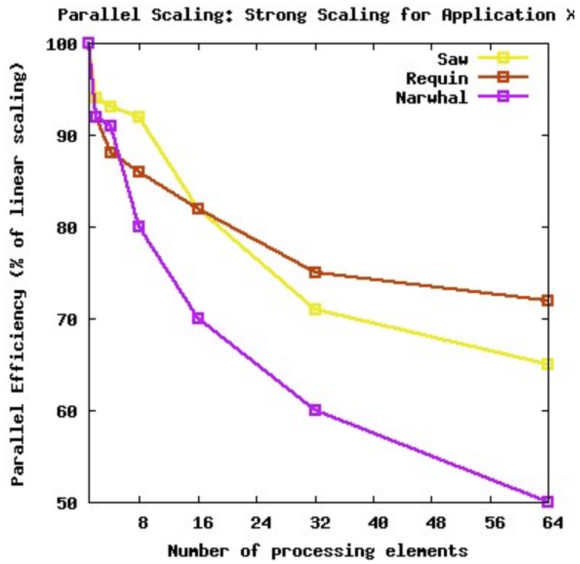

Figura 1. Escalado fuerte: escalamiento fuerte para la aplicación x[4].

III-B. EFICIENCIA DE ESCALAMIENTO FUERTE

Ruby Lee en 1980 definió varios parámetros para evaluar el cálculo paralelo. A continuación se muestra la definición de dichos algunos parámetros.

Sea O(n) el número total de operaciones elementales realizadas por un sistema con n elementos de proceso, y T(n) el

tiempo de ejecución en pasos unitarios de tiempo. En general, T(n) ¡O(n) si los n procesadores realizan más de una operación por unidad de tiempo, donde $n \geq 2$. Supongamos que T(1) = O(1) en un sistema mono-procesador, lo cual supone que IPC = 1 (IPC= Instrucciones Por Ciclo). El factor de mejora del rendimiento (speed-up, aceleración o rapidez) se define como:

$$S_n = \frac{T_1}{T_n} \tag{1}$$

La eficiencia del sistema para un sistema con n procesadores se define como:

$$E_n = \frac{S_n}{n} = \frac{T_1}{n * T_n} \tag{2}$$

III-C. DESVIACIÓN ESTÁNDAR

La desviación estándar es la medida de dispersión más común, que indica qué tan dispersos están los datos con respecto a la media [2]. Mientras mayor sea la desviación estándar, mayor será la dispersión de los datos. En ese sentido, se utilizará dicha medida para determinar el error del promedio de eficiencia obtenido previamente. La fórmula aplicada en el software fue el siguiente:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_1 - \mu)^2} \qquad , donde\ \mu = \frac{1}{N}\sum_{i=1}^{N} X_i$$

IV. METODOLOGIA

La metodología para la evaluación, si un código se considera escalable o no es llevado de la mano con la eficiencia, que viene una comparación del grado de speed-up conseguido frente al valor máximo. Dado que $1 \leq S(n) \leq n$, tenemos $\frac{1}{n} \leq E(n) \leq 1$
Siendo:

E(n) = 0, cuando todo el programa se ejecuta en un único procesador de forma secuencial.

E(n) = 1, cuando todos los procesadores están siendo utilizados durante el periodo de ejecución.

A partir de ello se evalua el porcetanje de escalabilidad lineal antes mencionado.

V. PROCEDIMIENTO

El procedimiento parte con elección del formato al cual se va estandarizar la información para su posterior uso. En este caso, se definió el formato .csv. En el desarrollo del código en lenguaje python se filtro lo necesario a usar para determinar la escalabilidad con los conceptos antes mencionados. Cabe señalar que fue necesario crear un archivo adicional con el mismo formato para verificar el correcto funcionamiento del algoritmo. Posteriormente se utilizo librerías gráficas de python como matplotlib para visualizar de una mejor manera los resultados de las operaciones de tiempos, eficiencia finalmente escalabilidad.

VI. RESULTADOS

Se presenta información relevante en tablas, al someter el resultado de algunos códigos provistos a nuestro software que estandariza la información en ella a formato .csv filtrando información según los conceptos antes mencionados, el software se desarrollo en python 3.6.

Numero de Procesos	Tiempo	Eficiencia
1	11244.12	1.0
2	10212.973	0.5505
4	8783.32	0.32
8	6022.1	0.23342
16	4012.1	0.1752

Tabla I
PROBLEMA N-CUERPOS

Numero de Procesos	Tiempo	Eficiencia
2	288.67	0.5
3	201.00	0.4787
4	166.00	0.4347
5	155.67	0.3709

Tabla II
ÍNDICE-INVERTIDO - 1 GB

Numero de Procesos	Tiempo	Eficiencia
2	502.00	0.5
3	335.67	0.4985
4	293.67	0.4274
5	245.00	0.4098

Tabla III
ÍNDICE INVERTIDO - 2 GB

Numero de Procesos	Tiempo	Eficiencia
2	502.00	0.5
3	335.67	0.4649
4	293.67	0.4478
5	245.00	0.4046

Tabla IV
ÍNDICE INVERTIDO - 3 GB

Número de Procesos	Tiempo	Eficiencia
1	4.91	1.0
2	2.64	0.9299
3	2.12	0.772
4	1.75	0.7014
5	1.72	0.5709
6	1.67	0.49
7	1.78	0.3941
8	1.89	0.3247
9	2.05	0.2661
10	2.28	0.2154
11	2.57	0.1737
12	2.72	0.1504

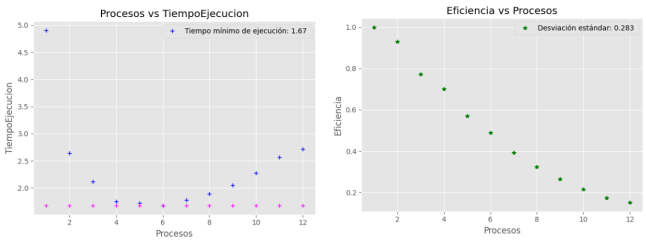

Figura 2: To graph-1

Número de Procesos	Tiempo	Eficiencia
1	20.34	1.0
2	11.12	0.9146
3	8.61	0.7875
4	7.52	0.6762
5	7.14	0.5697
6	7.24	0.4682
7	7.32	0.397
8	7.84	0.3243
9	8.45	0.2675
10	9.12	0.223
11	10.22	0.1809
12	10.95	0.1548

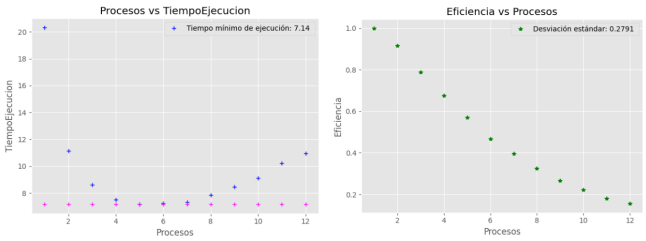

Figura 3: To graph-4

Número de Procesos	Tiempo	Eficiencia
1	30.19	1.0
2	16.24	0.9295
3	12.51	0.8044
4	10.67	0.7074
5	10.66	0.5664
6	10.51	0.4788
7	10.65	0.405
8	11.68	0.3231
9	12.39	0.2707
10	13.81	0.2186
11	15.24	0.1801
12	16.14	0.1559

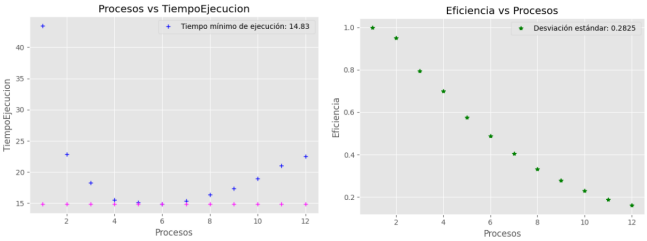

Figura 4: To graph-4

Número de Procesos	Tiempo	Eficiencia
1	50.00	1.0
2	26.62	0.9391
3	20.67	0.8063
4	17.95	0.6964
5	17.74	0.5637
6	17.42	0.4784
7	18.67	0.3826
8	18.85	0.3316
9	19.94	0.2786
10	21.67	0.2307
11	23.78	0.1911
12	25.62	0.1626

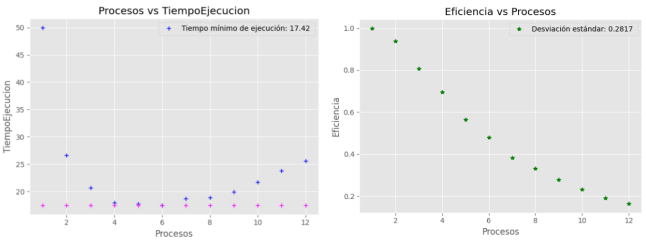

Figura 5: To graph-10

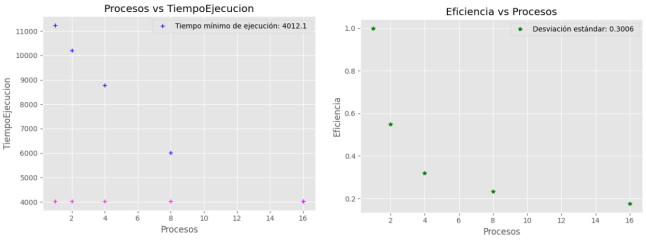

Figura 6: Indice Invertido 1GB

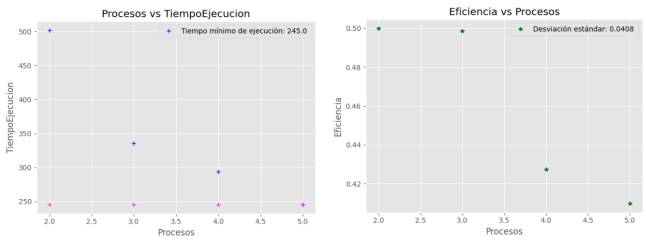

Figura 7: Indice Invertido 2GB

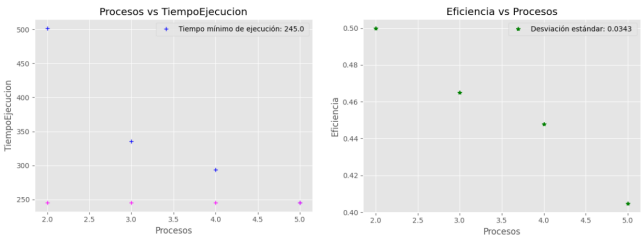

Figura 8: Indice Invertido 3GB

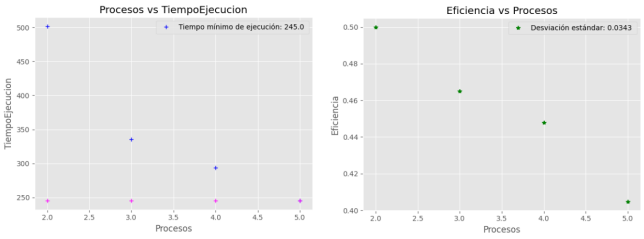

Figura 9: Simulador de Pandemia 50x50

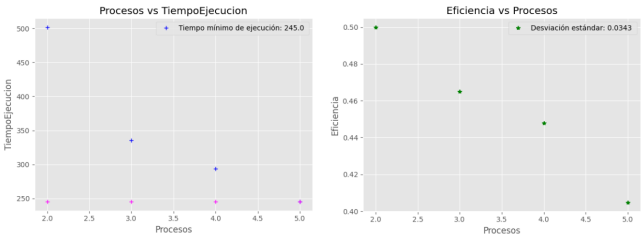

Figura 10: Simulador de Pandemia 100x100

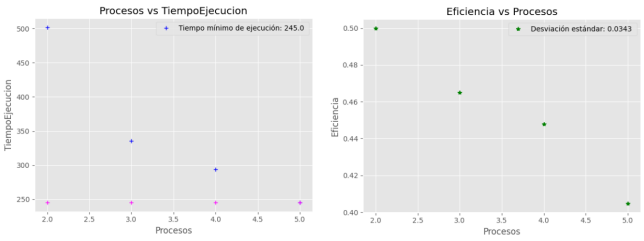

Figura 11: Simulador de Pandemia 200x200

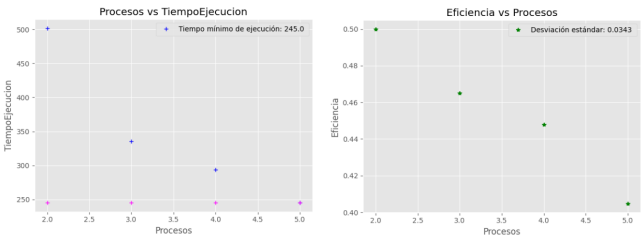

Figura 12: Simulador de Pandemia 400x400

Número de Procesos	Tiempo	Eficiencia
2	0.0420	0.5
4	0.0238	0.4403
8	0.0205	0.2557
16	0.0132	0.1988

Tabla V
SIMULADOR DE PANDEMIA 50X50

Número de Procesos	Tiempo	Eficiencia
2	0.171	0.5
4	0.101	0.4221
8	0.082	0.263
16	0.055	0.1941

Tabla VI
SIMULADOR DE PANDEMIA 100X100

Número de Procesos	Tiempo	Eficiencia
2	0.656	0.5
4	0.371	0.442
8	0.325	0.2528
16	0.227	0.181

Tabla VII
SIMULADOR DE PANDEMIA 200X200

Número de Procesos	Tiempo	Eficiencia
2	2.64	0.5
4	1.51	0.4346
8	1.32	0.25
16	0.94	0.1756

Tabla VIII
SIMULADOR DE PANDEMIA 400X400

Código	Promedio de Eficiencia	Desviación
N-Cuerpos	0.456	0.3006
Indice Invertido 1GB	0.446	0.0494
Indice Invertido 2GB	0.459	0.0408
Indice Invertido 2GB	0.454	0.0343
Tograph 1	0.499	0.283
Tograph 4	0.497	0.279
Tograph 6	0.503	0.284
Tograph 8	0.509	0.2825
Tograph 10	0.505	0.282
Simulador Pandemia 50x50	0.3487	0.1249
Simulador Pandemia 100x100	0.345	0.1219
Simulador Pandemia 200x200	0.3439	0.1312
Simulador Pandemia 400x400	0.340	0.132

VII. ANALISIS DE RESULTADOS

Como se puede apreciar en los resultados los códigos varían de escalabilidad, como por ejemplo, el código de Indice Invertido que se ejecuto con una carga de 1GB ,2GB y 3GB con porcentajes de escalamiento lineal de 44.6 %, 45.9 % y 45.9 %, con un error del 4.94 %, 4.08 % y 3.43 % respectivamente. Por otro lado encontramos el código Tograph 1,4,6,10 y sus derivaciones de carga, parten con un escalamiento lineal de 49.9 %, 50.3 %, 50.9 % y 50.5 % y un error del 28.3 %, 27.9 %, 28.4 % y 28.25 % respectivamente. Finalmente el código de Simulador de Pandemia en rangos de 50x50, 100x100, 200x200 y 400x400 presenta un escalamiento lineal de 34.87 %, 34.50 %, 34.39 % y con un error 12.49 %, 12.19 %, 13.12 % y 13.20 %.

VIII. CONCLUSION

En conlusión , podemos indicar que la escalabilidad varía en las distintas pruebas desarrolladas. Observándose que uno de los códigos con mejor escalamiento lineal es el de Indice

Invertido , ya que si bien el escalamiento que presenta no es el mejor de los 3, sí muestra un muy bajo porcentaje de error a diferencia del resto, proporcionándonos así, un mejor contexto para la implementación de dicho código en computadores de alto rendimiento.

- No basta con poseer un supercomputador para ejecutar tareas complejas, si el software a ejecutar no es escalable tampoco será el tiempo que demande su resultado.

REFERENCIAS

[1] Kai Hwang. *Advanced computer architecture*: Parallelism, scalability,programmability. McGraw-Hill, 1993.

[2] John Gurland and Ram C. Tripathi (1971), *A Simple Approximation for Unbiased Estimation of the Standard Deviation*: The American Statistician 25 (4): 30-32 .

[3] Measuring Parallel Scaling Performance(2016)[Internet].Disponible en https://www.sharcnet.ca/help/index.php/MeasuringParallelScalingPerformanceStrongScaling .

[4] Introduction to HPC: Scaling tests and profiling for efficient HPC[Internet].Disponible en https://pdc-support.github.io/hpc-intro/10-scaling/ .

[5] PI2-Scalability: Software de escalabilidad en python 3.6.Disponible en https://github.com/FabrizioVx/PI2Scalability.