MPI: Laboratorio Computación Distribuida

Paula Andrea Caballero Sepulveda, Mariana Ruge Vargas

Universidad Sergio Arboleda

paula.caballero01@usa.edu.co , mariana.ruge01@usa.edu.co

RESUMEN: Se presenta un análisis sobre el Problema de la Mochila en un entorno maestro-esclavo, con el maestro en una máquina física con Linux y el esclavo en una máquina virtual (VirtualBox) con Ubuntu. Se implementó el algoritmo evaluando el impacto del número de nodos (2 en este caso), además del tiempo de ejecución y uso de memoria. Se usó conexión SSH (Secure Shell) para la comunicación y reporte de pruebas entre maestro y esclavo. Se realizaron pruebas desconectando la red o apagando nodos durante la ejecución, midiendo el impacto en el desempeño. Además, se utilizó Wireshark para analizar tráfico y velocidad de red. Los resultados muestran cómo estos factores afectan la eficiencia del sistema.

I. INTRODUCCIÓN

El Problema de la Mochila se considera un desafío de optimización combinatoria y se considera un problema NP-completo. El presente informe de laboratorio muestra la implementación en un entorno maestro-esclavo, donde el maestro se corre en una máquina real Linux y el esclavo se considera una máquina virtual Ubuntu en VirtualBox comunicado mediante el protocolo Secure Shell (conocido como SSH). Se busca analizar el impacto del número de nodos en tiempo de ejecución y uso de memoria. Esperamos sea de utilidad para futuros análisis.

II. OBJETIVOS

- Evaluar el desempeño del Problema de la mochila en un entorno maestro-esclavo, midiendo su impacto en tiempo de ejecución, uso de memoria y comunicación de red.
- Analizar el impacto de los nodos y la capacidad de la latencia en los recursos por medio de máquinas y su impacto.
- Comparar el rendimiento del algoritmo del Problema de la Mochila en diferentes configuraciones de nodos, identificando variaciones en eficiencia y estabilidad del sistema.
- Evaluar los recursos que se consumen al ejecutar el problema de la Mochila distribuyendo los nodos.

III. MARCO TEÓRICO

Complejidad Computacional

El problema de la mochila 0/1 es NP-completo, lo que implica que no existe un algoritmo polinomial conocido que lo resuelva en todos los casos. Para abordarlo, se han desarrollado diferentes estrategias:

- **Enfoques exactos:** Programación Dinámica y Ramificación y Acotación (Branch & Bound).
- **Enfoques heurísticos:** Algoritmos voraces, recocido simulado y algoritmos genéticos.

Computación Paralela y el Modelo Maestro-Esclavo

Dado que el problema es costoso computacionalmente, una solución eficiente es la paralelización mediante el modelo Maestro-Esclavo. En este esquema, el maestro distribuye tareas entre los esclavos, quienes procesan los datos y retornan resultados.

MPI (Message Passing Interface)

MPI es un estándar de comunicación entre procesos en sistemas distribuidos. En este trabajo se usa `mpi4py`, una implementación de MPI para Python, permitiendo la comunicación eficiente entre procesos en entornos paralelos.

SSH y Virtualización

Para la comunicación entre la máquina real (Linux) y la máquina virtual (Ubuntu en VirtualBox), se emplea el protocolo SSH, que permite la ejecución de procesos remotos en un entorno distribuido.

IV. METODOLOGÍA

En esta sección se describe el montaje realizado y los elementos utilizados en la práctica. Se emplearon máquinas virtuales para simular un entorno distribuido. A continuación, se detallan los elementos requeridos para la implementación:

Software utilizado:

- Sistema operativo host: Linux (Ubuntu).
- Máquina virtual: VirtualBox con Ubuntu.
- MPI (Message Passing Interface) con `mpi4py` instalado en todas las máquinas.
- OpenSSH para la comunicación remota entre nodos.
- Python para la implementación del algoritmo.

Configuración de la red:

- Conexión SSH configurada entre la máquina real y las máquinas virtuales.
- Configuración de `hosts` para permitir la comunicación entre nodos esclavos y el maestro.
- Verificación de la conectividad con `mpirun --hostfile hosts` y pruebas de ejecución remota.

Problema de la mochila

El problema de la mochila, abreviado como KP (por sus siglas en inglés Knapsack Problem), consiste en buscar la mejor solución a una situación análoga al llenar una mochila, con una capacidad dada para soportar un peso determinado, con todo o una parte de un conjunto de objetos, cada uno con un peso y valor específicos (si se quieren priorizar) (Puchingery col., 2010). Los objetos colocados en la mochila deben maximizar el valor total, sin exceder la capacidad máxima de la mochila.

- Su definición formal es la siguiente: supongamos que tenemos n distintos tipos de ítems, que van del 1 al n.

De cada tipo de ítem se tienen qi ítems disponibles, donde qi es un entero positivo que

$$\forall i = 1, n$$

Cada tipo de ítem i puede tener un beneficio asociado dado por vi y un peso (o volumen) wi. Usualmente se asume que los pesos no son negativos. Por otro lado, se tiene una mochila donde se pueden introducir los ítems, que soporta un peso máximo (o volumen máximo) .

El problema consiste en meter en la mochila los ítems de tal forma que se maximice el valor de los ítems que contiene, siempre que no se supere el peso máximo que puede soportar la misma. La solución al problema vendrá dada por la secuencia de variables x1, x2, ..., xn donde el valor de xi indica cuántas copias se meterán en la mochila del tipo de ítem i.

$$max \sum_{i=1}^{n} v_i x_i$$

Inicialización de MPI: Se obtiene el identificador del proceso (`rank`) y el número total de procesos (`size`).

Definición del problema: Se tienen listas de valores y pesos de los objetos, junto con la capacidad máxima de la mochila (`W`).

División del problema:

- El **maestro** (proceso `rank == 0`) divide el problema entre los esclavos.
- Cada **esclavo** recibe una porción del problema y calcula la solución parcial de su segmento.

Cálculo del problema de la mochila:

- Se usa **Programación Dinámica** con una matriz `K` de dimensiones `(n+1) x (W+1)`, donde cada celda representa la mejor solución para una submochila con `i` objetos y peso `w`.
- Cada esclavo calcula una parte de esta matriz y devuelve su resultado.

Recolección de resultados:

- El maestro recibe los valores parciales de los esclavos y los suma para obtener la solución final.
- También mide los **tiempos de ejecución** y calcula estadísticas.

Visualización de Resultados:

- Se imprimen métricas como el tiempo total, el tiempo promedio por nodo, la desviación estándar y los tiempos mínimos y máximos.
- Se generan **gráficas ordenadas en cuadrícula** para visualizar los tiempos de ejecución por nodo.

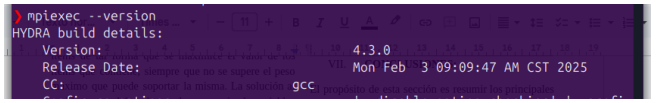
Cada tipo de ítem i puede tener un beneficio asociado dado por vi y un peso (o volumen) wi. Usualmente se asume que los pesos no son negativos. Por otro lado, se tiene una mochila donde se pueden introducir los ítems, que soporta un peso máximo (o volumen máximo) W.

El problema consiste en meter en la mochila los ítems de tal forma que se maximice el valor de los ítems que contiene, siempre que no se supere el peso máximo que puede soportar la misma. La solución al problema vendrá dada por la secuencia de variables x1, x2, ..., xn donde el valor de xi indica cuantas copias se meterán en la mochila del tipo de ítem i.

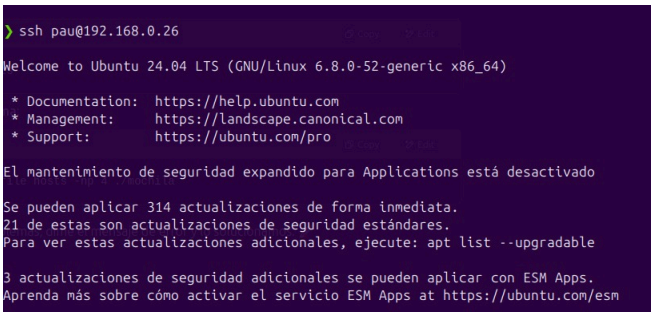
V. RESULTADOS

Versiones de MPI ejecutadas en el esclavo en el maestro

Todos los resultados se ejecutan en la máquina maestro con la complejidad del nodo esclavo.



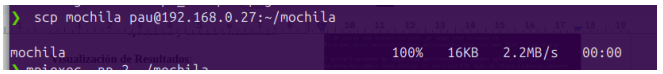
Conexión con la máquina SSH



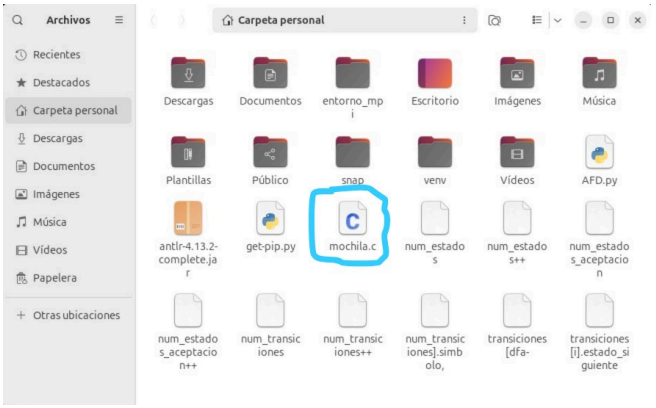
En la siguiente imagen se observa la conexión desde la máquina maestro (mariana-vivobook) a la maquina esclavo (pau@pau-Virtualbox)

Gráfica. 1: título de figura 1

Se realiza la copia en la máquina esclavo



Se visualiza el archivo en el sistema de archivos de la máquina esclavo



Corremos los archivos, se corre tanto en la máquina virtual como en la máquina hosts

```
pau@pau-VirtualBox:~$ ./mochila
./mochila: error while loading shared libraries: libmpi.so.12: cannot open shared
object file: No such file or directory
```

Ejecución en la máquina maestro y en la máquina esclavo

```
❯ mpirun -np 4 ./mochila
Nodo 1: Resultado parcial: 160, Tiempo: 0.000009 segundos
Nodo 2: Resultado parcial: 260, Tiempo: 0.000009 segundos
Nodo 3: Resultado parcial: 260, Tiempo: 0.000005 segundos
Valor máximo que se puede obtener: 680
Tiempo total: 0.0001 segundos
Nodo 1: Resultado parcial: 160, Tiempo: 0.000009 segundos
Nodo 2: Resultado parcial: 260, Tiempo: 0.000009 segundos
Nodo 3: Resultado parcial: 260, Tiempo: 0.000005 segundos
Guardado en tiempos.txt: Nodo 1, Tiempo 0.000009
Guardado en tiempos.txt: Nodo 2, Tiempo 0.000009
Guardado en tiempos.txt: Nodo 3, Tiempo 0.000005
Generando gráfico de boxplot...
```

Dentro de las pruebas también se realizaron gráficos que muestran la eficiencia del paralelismo al correr el archivo.

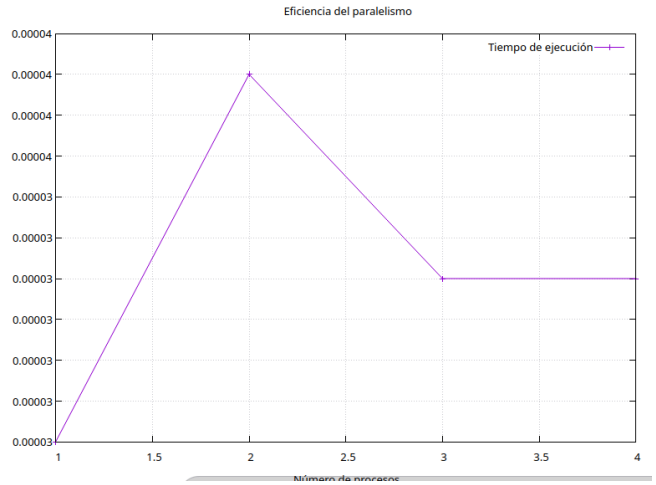


Figura 1.2 Eficiencia del paralelismo al correr el programa del maestro al esclavo.

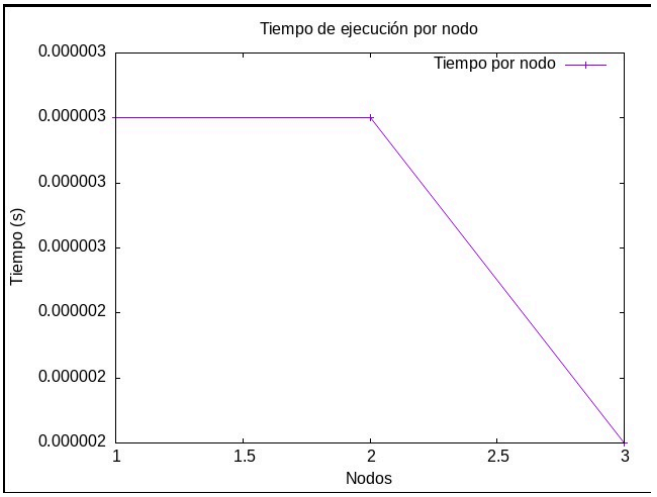


Figura 1.3 Relación del tiempo de ejecución por nodo al ejecutar con recursos de la máquina esclava.

ANÁLISIS DE RESULTADOS

En las siguientes tablas se presentan los resultados de las pruebas al correr los algoritmos desde el nodo maestro al nodo esclavo.

Prueba	Valor que se obtiene	Tiempo (s)	Nodos
1	680	0.0011	4
2	680	0.0003	4
3	260	0.0001	2
4	260	0.0005	2

Según los datos se puede observar que la optimización del tiempo de ejecución y escalabilidad son clave cumpliendo con los principios de la computación distribuida. Se refleja que en sistemas distribuidos, factores como la asignación inicial de tareas, la latencia en la red o incluso el estado momentáneo de los nodos (ej. carga de CPU) impactan el rendimiento, incluso para la misma instancia del problema.

Por ejemplo en la prueba 1 y 2 (valor de 680) contra las pruebas 3 y 4 (valor 260) se puede observar que más capacidad de la mochila aumenta la complejidad computacional, debido al uso de los 4 nodos en las pruebas para mejorar la ejecución de la carga. Mientras que las pruebas 3 y 4 al ser menor la capacidad solamente requieren 2 nodos.

Aunque el número de nodos influye, la reducción del tiempo no es lineal, puesto que en algunas pruebas como la primera y la segunda, se ve un balance de la carga en la segunda ejecución. Si un nodo recibe más subproblemas o tareas que otro se genera un cuello de botella , puesto como se ve en la prueba 4 (0.0005s) un nodo pudo haber sido sobrecargado por la caída abrupta de la eficiencia, por ello podemos decir que en ciertos casos, el costo de coordinar incluso pocos nodos supera las ganancias del paralelismo. Esto resalta un principio clave en computación distribuida: el paralelismo no siempre garantiza aceleración, especialmente si la sobrecarga de gestión (ej. sincronización) no se minimiza.

CONCLUSIONES

- Se observa que la evaluación del desempeño en un entorno maestro-esclavo revela cómo la distribución de tareas impacta el tiempo de ejecución, el uso de memoria y la comunicación de red. La correcta asignación de las cargas de trabajo entre los nodos es esencial para mejorar la eficiencia y minimizar los costos de comunicación.
- La latencia y el manejo eficiente de la comunicación entre nodos son cruciales para determinar cómo estos elementos impactan el rendimiento respecto a los recursos y la distribución de los nodos (overheight).
- Se registra variabilidad en el rendimiento al cambiar la configuración de los nodos muestra cómo las diferencias en la cantidad y disposición de nodos afectan la eficiencia y estabilidad del sistema. Al comparar distintas configuraciones, se pueden identificar cuellos de botella que serán optimizados en futuros estudios.
- Por ello se evidencia la importancia de los recursos de los esclavos, puesto que al distribuir los nodos, el consumo de recursos (como la memoria y la comunicación de red) varía según el número de nodos esclavos. Evaluar estos recursos permite determinar el número óptimo de nodos y cómo el sistema puede escalar sin generar sobrecarga, dependiendo del porcentaje de CPU se pueden procesar las tareas.

REFERENCIAS

Vélez-Díaz, Moreno-Gutiérrez, Martínez-Cervantes y Sánchez-Muñoz. “Boletín Científico :: UAEH”. Universidad Autónoma del Estado de Hidalgo :: UAEH. Accedido el 16 de marzo de 2025. [En línea]. Disponible:

https://www.uaeh.edu.mx/scige/boletin/tlahuelilpan/n6/e2.html?utm_source=chatgpt.com

“Problema de la mochila: 'Algoritmo', 'Ejemplos' | StudySmarter”. StudySmarter ES. Accedido el 16 de marzo de 2025. [En línea]. Disponible:

https://www.studysmarter.es/resumenes/ciencias-de-la-computacion/algoritmos-en-ciencias-de-la-computacion/problema-de-la-mochila/?utm_source=chatgpt.com

Andrew S Tanenbaum and Maarten van Steen, Distributed Systems. Principles and Paradigms. Prentice Hall India, 2008.