HPC Informe Tarea 2

Pablo Messina

June 2019

1 Resumen de la solución

NOTA: Todo el código de la tarea de puede encontrar en `https://github.com/PabloMessina/HPC_Homeworks/tree/master/T2`.
Muy resumidamente, mi algoritmo funciona de la siguiente manera:

- El proceso root (rank=0) se encarga de leer el input del usuario y broadcastear dicho input a los demás procesos (en adelante, los *workers*)
- Luego el root ejecuta un BFS (breadth-first search) **serial** bien rápido sobre el árbol de soluciones con el fin de encontrar suficientes nodos para repartir entre los workers. Como resultado se obtiene una *cola* de soluciones parciales (nodos que no son hojas en el árbol de soluciones). Si se da la casualidad que durante este corto BFS el root ya encontró una solución, entonces se notifica a los workers que no hay nada que hacer y se imprime la respuesta de inmediato. Si no, se ejecutan los puntos a continuación.
- Posteriormente el root particiona la cola lo más equitativamente posible en *chunks* y envía a cada worker un *chunk* para trabajar.
- Cada worker entonces comienza a explorar los sub-árboles de soluciones que le tocaron usando backtracking (equivalente a hacer DFS con podas). Al mismo tiempo, cada cierto número de iteraciones cada worker ejecuta un MPI_Test() para chequear si es que algún otro worker ya encontró una solución antes.
- En paralelo, el root comienza a esperar los resultados obtenidos por los workers. Para ello se ejecuta un loop que itera tantas veces como workers hay. En cada iteración se ejecuta un MPI_Recv() bloqueante, pero la gracia está en que se acepta cualquier worker por medio del argumento MPI_ANY_SOURCE. Tan pronto como se reciva una solución encontrada, el root notifica de manera **asíncrona** a todos los workers que alguien ya encontró una solución, utilizando para ello la función MPI_Ibcast(). De esta manera los workers que siguen trabajando eventualmente ejecutarán un MPI_Test() y se darán cuenta que ya está resuelto el problema y se detendrán.

- Finalmente, una vez que el root ya recibió respuestas de parte de todos los workers, se procede a imprimir la respuesta y terminar.

1.1 Experimentos y Resultados

Usé 4 archivos de prueba. Un par con N=28 (uno sin solución y otro con solución única) y otro par con N=30 (lo mismo, sin solución y con solución única). Los resultados se pueden ver en las figuras 1 y 2.

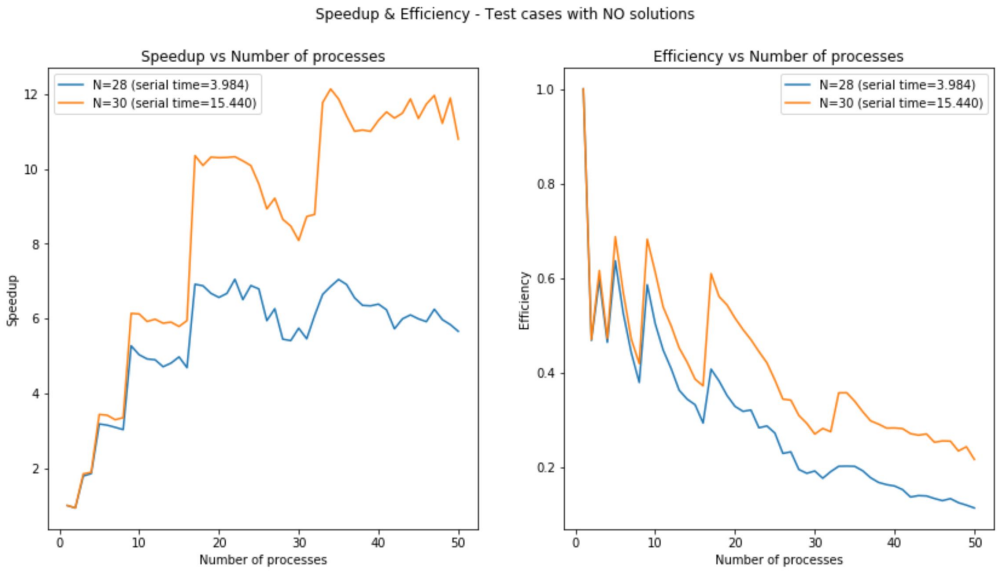

Figure 1: Speedup y Eficiencia en casos de prueba **sin** solución

En la Figura 1 tenemos los resultados de speedup y eficiencia para los casos de prueba sin solución. Notamos que aumentar la cantidad de procesos tiende a mejorar el speedup, alcanzando un speedup máximo mayor a 12 con un poco más de 30 procesos para N=30. Comparando N=28 y N=30 notamos que las formas de las curvas de speedup son relativamente similares pero la escala es mayor para N=30. Este resultado tiene mucho sentido ya que al aumentar el N la cantidad de trabajo paralelizable (exploración de soluciones) aumenta y por lo tanto este adquiere un tamaño relativo más grande respecto a la parte serial. No obstante, la eficiencia dista de ser perfecta. La eficiencia tiende a ir a la baja con más procesos, llegando a niveles pésimos de en torno a 0.2 con 50 procesos. De todas maneras notamos que la curva de eficiencia para N=30 es mejor que la curva para N=28, lo que concuerda con la intuición de que para N más grandes hay más trabajo paralelizable y por ende tener más workers se aprovecha de mejor manera.

La Figura 2 nos muestra las curvas de speedup y eficiencia en los casos de prueba con una única solución. En general son curvas muy similares al caso

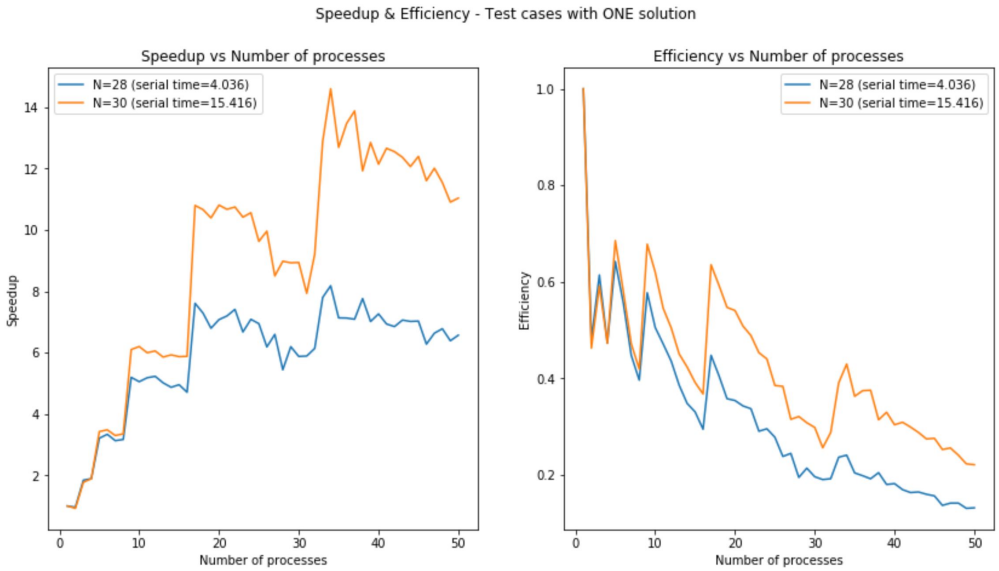

Figure 2: Speedup y Eficiencia en casos de prueba con **una sola** solución

anterior, por lo que todo lo dicho anteriormente se aplica acá también. Un detalle a destacar es el hecho que el speedup máximo alcanzado aquí fue de casi 15 para N=30 y con aproximadamente 32-33 procesos. Esta ligera mejora en el speedup probablemente se deba al factor de interrupción de ejecución prematura que ocurre cuando un proceso encuentra una solución y el root notifica al resto que ya pueden terminar.