Meta Volante 1a: Implementación HPL

Samuel Palacios, Simón Marín, David Alsina, Isabel Piedrahíta
Ingeniería de Sistemas, Universidad EAFIT
Medellín, Colombia
Matemáticas aplicadas y ciencias de la computación, Universidad del Rosario
Bogotá, Colombia

sdpalaciob@eafit.edu.co
smaring1@eafit.edu.co
davidsa.florez@urosario.edu.co
ipiedrahiv@eafit.edu.co

***Abstract*— El presente documento tiene por objetivo demostrar el proceso de ejecución de HPL en un cluster de 2 nodos con la intención de obtener un desempeño de como mínimo el 80% con respecto al máximo desempeño posible de la máquina calculado de manera teórica. A continuación se expondrá el entorno de ejecución, el máximo desempeño teórico de la máquina, las optimizaciones realizadas y aquellas que potencialmente serían útiles pero no pudieron ser implementadas por el equipo de trabajo en el marco de tiempo dispuesto.**

I. ENTORNO DE EJECUCIÓN

1.1 **Hardware:**

- **Cantidad de nodos:** 2
- **Procesadores por nodo:** 2×CPU Intel® Xeon® CPU. E5-2670, 8 núcleos por procesador, hyper-threading desactivado.
- **InfiniBand:** ConnectX3 con driver mlx4.
- **Memoria por nodo:** 16 x 4GB DIMM DDR3 1333 MT/s.

1.2 **Software:**

- **HPL:** 2.3 [1]
- **OS:** CentOS 8.2 [2]
- **BLAS:** OpenBLAS [3], MKL
- **MPI:** MPICH [4], Intel MPI
- **NFS:** NFSv3 [5]

II. MÁXIMO DESEMPEÑO POSIBLE

Teniendo en cuenta las especificaciones del entorno de ejecución previamente dispuestas, procedemos a calcular el máximo desempeño teórico (R_{peak}) de la siguiente manera [6]:

$$R_{peak} = (\#nodos)(\frac{\#cpus}{nodo})(\frac{\#nucleos}{cpu})(\frac{ciclos}{segundo})(\frac{FL}{ci}$$

$$R_{peak} = (2)(2)(8)(2.9 \times 10^{9})(8) = 742.4\,GF$$

Conociendo el R_{peak} de nuestra máquina y sabiendo que en promedio un HPL se optimiza para alcanzar entre un 50%-80% del R_{peak} , podemos calcular que el resultado de HPL deseado es entre 332.8 GFLOP/s y 532.48 GFLOP/s, sin embargo el equipo de trabajo aspira a conseguir un resultado como mínimo de 530 GFLOP/s mediante las optimizaciones planteadas.

III. OPTIMIZACIONES IMPLEMENTADAS

A. Comunicación mediante InfiniBand

La primera optimización que se planteó fue mejorar la comunicación entre los nodos mediante el uso de la red InfiniBand. Esto se hizo con la intención de disminuir la latencia de la red, aumentar el ancho de banda y disminuir el gasto de la CPU causado por la transferencia de datos entre nodos de una misma red. Si bien esto no está directamente relacionado con la implementación de HPL es esencial para evitar cuellos de botella posteriormente, ya que la velocidad de paso de mensajes es una de las grandes limitantes de la velocidad de un programa distribuido. Para lograr esto se instalaron los drivers de Mellanox directamente a partir del código fuente compilado contra el kernel usando las siguientes banderas durante la instalación: --speed-up-kmp --enable-opensm --basic --hpc --upstream-libs. Posteriormente se configuró la red de infiniband ib0 con nmtui y se probó su desempeño con qperf.

Habiendo implementado únicamente este cambio, y usando MPICH y OpenBLAS para correr el benchmark, el desempeño de HPL fue de 1.6169×10^{-2}GFLOPs, es decir, un 0.00217% del R_{peak}.

B. Archivo de configuración

Posteriormente se hizo tuning de manera manual, usando como guía la calculadora de HPL sourceforge [7]. Era claro que este iba a ser el ítem con el mayor impacto

sobre el desempeño de HPL, por lo que se hicieron varias pruebas con configuraciones diferentes. Los cambios principales que se hicieron fueron:

- N: Descubrimos que puede ocupar en nuestro caso hasta un 88% de la memoria RAM sin tener que hacer SWAP, en nuestro caso esto corresponde a un N de 104832.
- P y Q: Se modificaron para acoplarse al número de procesadores disponibles, para mejorar el desempeño de seleccionaron experimentalmente los valores de 4 y 8 para P y Q respectivamente.
- NB: Para este valor se eligió el 224, para calcular este valor se siguió la recomendación de la calculadora además de algunos experimentos.

Con estos cambios se alcanzó un desempeño de 5.3074×10^{2}GFLOPs, este valor supera la meta auto-impuesta del equipo de trabajo y corresponde a un 79.7% del R_{peak}.

C. Utilización de la OneAPI de Intel

Dada nuestra arquitectura *(procesadores Intel® Xeon® CPU)*, para poder generar una comunicación MPI mejor adaptada y más eficiente optamos por usar la suite de OneAPI de intel que tiene optimizaciones específicas para estas CPUs, dicha suite incluye el compilador de intel MPI , y la implementación de BLAS Math Kernel Library *(MKL),* en la utilización de Intel MPI optamos por usar la herramienta mpitune_fast, con el fin de hacer autotunig de MPI y mejorar los resultados.

Adicionalmente decidimos correr el benchmark con la implementación de intel, debido a que así tendría mayor eficiencia. Se usó el HPL de intel con los siguientes valores:

- N: Gracias a Vincent Arcila notamos que es posible correr un N más alto que el anteriormente considerado, por lo que usamos un N de 125440.
- P y Q: Se modificaron para acoplarse a la implementación de intel, se eligieron 2 y 2 para P y Q respectivamente.
- NB: Para este valor se eligió el 256, ya que era el recomendado por la configuración de Intel.

Debido a restricciones de tiempo esta optimización se corrió en simultáneo con la próxima, por lo que no hay resultados de su efecto aislado en el performance de HPL.

B. Optimizaciones para sistemas NUMA

Para aumentar el rendimiento también es posible realizar binding, de forma tal que un proceso sólo pueda ser ejecutado en un solo núcleo físico. Esto reduciría el tiempo de acceso a datos, ya que no sería necesario recuperar datos de diferentes unidades de procesamiento. Se implementó usando numad descargado con el manejador de archivos por defecto de CentOS. De esta forma realizamos binding de procesos y obtuvimos de esta forma un desempeño de 637.951 GFLOPs, que equivale a casi el 86% del R_{peak}. Para el desarrollo de este fragmento fue indispensable la ayuda de Vincent Arcila, quien nos explicó cómo se utilizan estos recursos.

IV. OPTIMIZACIONES PLANTEADAS A FUTURO

A. Autotuning con SLURM

Todos los experimentos del archivo de configuración fueron realizados manualmente, de forma tal que no fueron tantos como habría sido ideal. Para obtener mejores resultados consideramos que sería útil realizar autotuning de HPL con una manejador de tareas como SLURM, de esta forma mientras el equipo de trabajo realiza otras actividades podrían estar corriendose pruebas, para hallar así la mejor configuración posible. Esta optimización no se realizó en esta ocasión debido a la complejidad del proceso de instalación y configuración de SLURM.

V. CONCLUSIONES

Usando todas las técnicas de optimización expuestas en el presente informe logramos un desempeño del 85.9%, sin embargo el principal contribuyente a este resultado fue el tuning al archivo de configuración de HPL, por lo que consideramos que realizando auto-tuning riguroso podríamos llegar a tener un desempeño de hasta el 95%.

La colaboración entre equipos también fue fundamental para alcanzar los resultados obtenidos. La mayor parte del equipo no tenía conocimientos previos de administración de servidores Linux, por lo que la ayuda de Vincent Arcila fue indispensable para solucionar algunos de los errores que no eran comúnmente encontrados, y por ende no podrían ser fácilmente investigados.

REFERENCIAS

[1] J. D. A. C. P. L. Antoine Petitet, Clint Whaley, “Hpl 2.3 - a portable implementation of the high-performance linpack.” [Online]. Available: https://www.netlib.org/benchmark/hpl/

[2] “CENTOS,” 2021. [Online]. Available: https://www.centos.org/

[3] Zhang Xianyi, Martin Kroeker, “OpenBLAS,” 2021. [Online]. Available: http://www.openblas.net/

[4] “MPICH,” 2021. [Online]. Available: https://www.mpich.org/

[5] Red Hat Customer Portal, “RHEL 8 Managing File Systems, CHAPTER 4. EXPORTING NFS SHARES,” 2021. [Online].

Available: https://access.redhat.com/documentation/en-us/red hat enterprise linux/8/html/managing file systems/exporting-nfs-shares managingfile-systems#configuring-an-nfsv4-only-server exporting-nfs-shares

[6] ASC Community, “The Student Supercomputer Challenge Guide”, 2018. https://doi.org/10.1007/978-981-10-3731-3

[7] Sourceforge, “Top500 HPL Calculator,” n.d. [Online] Available: http://hpl-calculator.sourceforge.net/