Optimising the High-Performance Linpack Benchmark on Two Different Clusters

ADRIAN ABEYTA, University of New Mexico, USA
ROGER DAVIS, University of New Mexico, USA
ANDREI POPA-SIMIL, University of New Mexico, USA

In High-Performance Computing (HPC), the Linpack software package is considered the standard measure by which to compare the throughput of supercomputer clusters. Our team was tasked with optimizing this benchmark on two distinct clusters provided by the University of New Mexico's Center for Advanced Research Computing (CARC). We accomplish this by varying several independent parameters and running the application across a small number of nodes. We find that there is a tight range of P, Q, NB, and matrix size values that maximizes the floating point operations per second (FLOPs) of both the Hopper and Wheeler clusters. Particularly, P and Q are close in size, with NB being a scalar multiple of these dimensions, as well as an optimal problem size value in an N-size matrix that is related to the total amount of memory available in a resource allocation. This work is crucial in determining the best input sizes and load balancing configuration for any scientific research conducted by UNM and the wider community.

ACM Reference Format:
Adrian Abeyta, Roger Davis, and Andrei Popa-Simil. 2024. Optimising the High-Performance Linpack Benchmark on Two Different Clusters. 1, 1 (May 2024), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

INTRODUCTION

With a wide-range of architectures and hardware, there are no two supercomputing clusters exactly the same. It is the objective of the HPC engineer to not only manage these components, but to also provide performance characteristic information to all of its users. To make efficient use of the enormous power of these machines, knowing the optimal input size and work distribution is essential. It is with this goal in mind that our team conducted empirical tests on two separate computing clusters housed at the University of New Mexico's Center for Advanced Research Computing to determine these parameters with an established, widely-accepted high-performance benchmark.

The High-Performance Linpack Benchmark accepts many parameter inputs, but our team focused on four of the most consequential in terms of performance. NB is the most important of these values. It dictates how the larger problem size is broken up into smaller blocks to be loaded into main memory and distributed among nodes and threads. In our benchmark runs, we discovered that this parameter has a large impact on the FLOPs, or floating point operations, per second throughput of each cluster. This value tends to be in the 100-200 range, with different values found for Hopper and Wheeler. The next most effective parameters are P and Q, which are instead related to the interconnection between nodes. Being that both clusters use Infiniband technology with the Message Passing Interface protocol, we were aware of a general range of values and ratios to start with. We kept this to a ratio of either 1:1, 1:2 or otherwise with Q being a small scalar multiple of P. Also, to keep performance optimal, $P * Q$ = total # processors. This is to ensure the size of the sub-matrix processing blocks are in line with the subsection of data that is distributed across nodes. Finally, the least consequential but still valuable parameter is the problem size. In these experiments, this is the N-square dimension size of the input matrices. To follow the recommendations for HPL, we used a standard formula for this value to ensure we use the most main memory available per node while still leaving enough room for operating system processes to avoid memory swapping to disk:

$$N \approx 0.80 * \sqrt{(\text{memory size in GB} * 1024 * 1024 * 1024 * \#\text{ nodes})/8} \quad (1)$$

METHODS

Our group started by using the provided resources from to determine the theoretical maximum attainable Gflops of the CPUs on the Wheeler and Hopper servers, and used the commands `qgrok` and `lscpu` to determine the number of cores and amount of memory for each node of the servers.

Server	CPU (Xeon)	TMax	MaxMem	ActMem
Hopper	Gold 6226R	486.4 Gflops	1TB	93GB
Wheeler	X5550	683.5 Gflops	144GB	43GB

Table 1. Server Hardware Specifications

We familiarized ourselves with parameter_sweep_array.slurm by modifying it to include the matrix dimension and P and Q variables from HPL_params.csv, and modified HPL_template.dat to accept those values and facilitate array testing, and running simple tests to confirm we could modify the variables as intended. We set N to 30,000 and 40,132 as our base value as advised by Ryan Scherbarth, tested a variety of NB values from 32 to 256 on Wheeler and determined that there was a marked decrease in performance above 228, then checked our findings on Hopper by testing every multiple of 16 from 196 to 588.

Appreciative thanks to Dr. Matthew Fricke and the team at UNM's CARC for providing us with the tools, knowledge, guidance, and resources to carry out these experiments.
Authors' addresses: Adrian Abeyta, University of New Mexico, Albuquerque, NM, 87131, USA; Roger Davis, University of New Mexico, Albuquerque, NM, 87131, USA; Andrei Popa-Simil, University of New Mexico, Albuquerque, NM, 87131, USA.

XXXX-XXXX/2024/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Next, we took drastically different approaches to testing the P and Q variables for each server. On Hopper, based on information from class discussion and our initial meeting with Ryan Scherbarth, we performed a short series of runs on a single node allowing 16 tasks to confirm our expectation that the best P and Q values were factors of T, tasks, the total sum of cores used across all nodes, in this case $T = 16$. On Wheeler, Andrei Popa-Simil tested all positive values for P, Q, on 2 nodes with $T = 16$, where $P * Q <= T$, the results of this test can be seen in **figure heatmap** 4.

Lastly, we scaled up our N value and number of used nodes on the servers to observe how the performance changed as it neared the theoretical maximum of the portion of the servers available to us. We determined that the servers have hard limits of 4 and 50 nodes for Hopper and Wheeler respectively, and that requesting 4 exclusive nodes from Hopper generally required more than a day of waiting to begin running. We also noticed that fewer non-HPL users were competing for server use during Saturday and Sunday. We set up tests for N values from our base, increasing by 15,000 and 4800 to 350,000, then by larger steps up to 500,000 for both servers, but found that the attempts either ran into time limits or afoul of system memory killers before attaining those higher values. On Wheeler, we were able to run more experiments and close in on a value near the one projected from our formula, but the turnaround time on Hopper was too long for us to be as precise in our final value.

RESULTS

Our initial runs on Wheeler with varying NB.

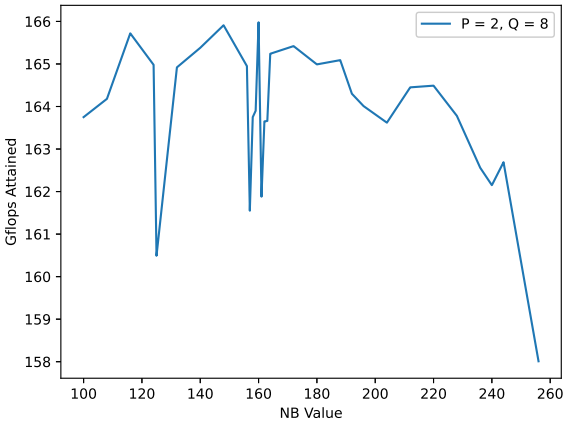

Fig. 1. Change in performance from changes in NB value on Wheeler for $N = 30,000$. Highly irregular, peaks at $NB = 160, 165.98$ Gflops. Sharply drops at $NB > 228$.

Our subsequent runs on Hopper.

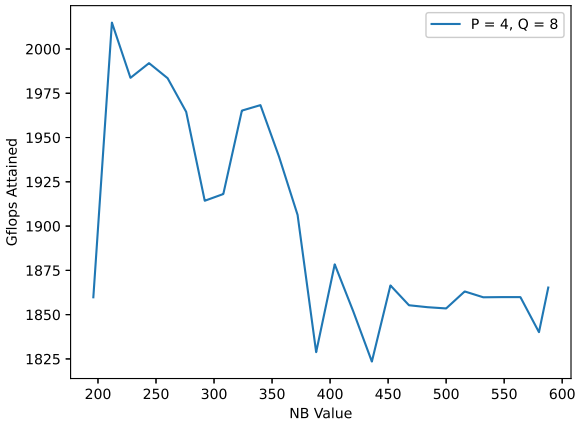

Fig. 2. Change in performance from changes in NB value on Hopper for $N = 96,000$. Comparably erratic to the results from Wheeler, NB values higher than 300 maintain the overall decreased performance.Peaks at $NB = 212, 201.49 Gflops$.

Comparing results led us to determine 196 as the tentative optimal base NB value for Hopper and 116 for Wheeler, with the potential of needing to increase them linearly once we found a sufficiently high N.

Our direct check of the factor pairs of $T = 16$ on Hopper.

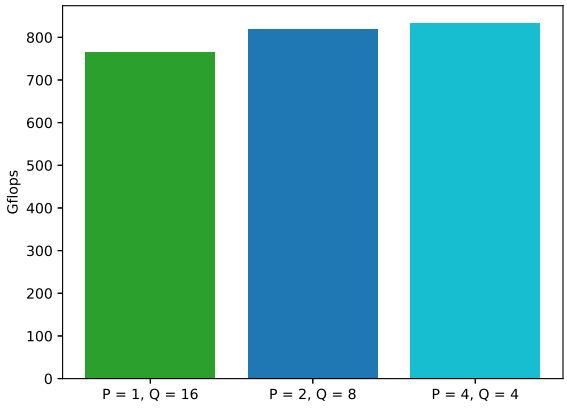

Fig. 3. Minor but noticeable increase in performance as the difference in the values decreases, especially between a difference of 15 and 6.

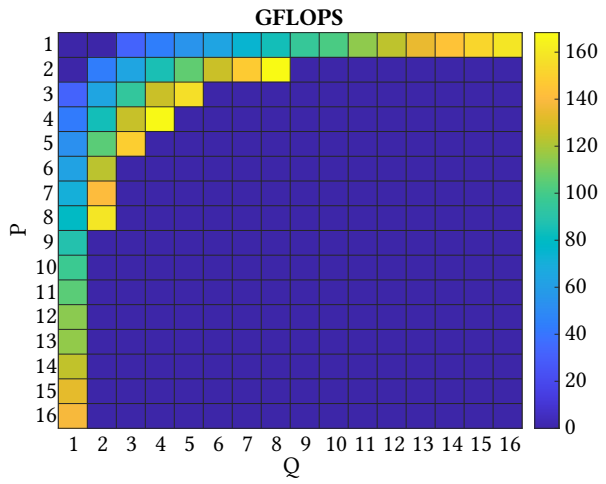

Fig. 4. **PQ Heatmap.** Above is a heatmap showing combinations of P and Q when run on 2 nodes with 8 cores each. As that means 16 total tasks the following holds true $P * Q <= 16$. Every PQ combination that results in $P * Q > 16$ fail to run and resulted in a 0 (dark blue). Additionally, very bad PQ combinations such as 1x1 took too long to plot and were terminated as obviously bad values. Notice an arc of best PQ combinations (yellow), this arc shows best PQ combinations for this node configuration. Here $P = 2$ and $Q = 8$ is the best combination.

After observing, the data it became clear that: *a*) having a $Q >= P$ provided improved results; *b*) that $P * Q < T$ could still provide usable results; and *c*) the closer $P * Q$ was to T the better the performance. These three properties could reduce our testing range for P and Q as using all 50 nodes on Wheeler gives 1434 possible combinations. The 17 total P and Q for 50 nodes is shown below. These tests resulted in $P = 9$ and $Q = 44$ giving us the best results to push off of. After further experimentation on Wheeler we found that $NB = 96$ was the best for 50 nodes with 8 cores each.

Our tests of scaling N and the number of nodes used on Hopper.

Our penultimate tests on Hopper with 4 exclusive nodes

P	Q	Gflops
4	100	2.50e+03
5	80	2.68e+03
6	66	2.66e+03
7	57	2.72e+03
8	50	2.73e+03
9	44	2.80e+03
10	40	2.74e+03
11	36	2.74e+03
12	33	2.44e+03
13	30	2.61e+03
14	28	2.64e+03
15	26	2.45e+03
16	25	2.37e+03
17	23	2.51e+03
18	22	2.54e+03
19	21	2.50e+03
20	20	2.52e+03

Table 2. PxQ for 50 Nodes on Wheeler at NB=104, N=40132. P=9 and Q=44 provide highest Gflops.

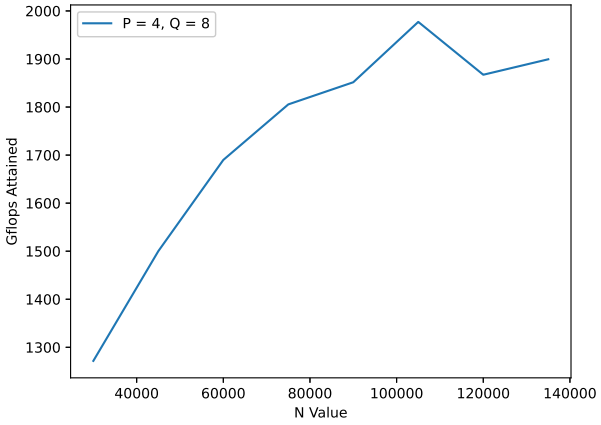

Fig. 5. Change in performance from changes in N value on Hopper for $NB = 196$. Semi-logarithmic progression, with an erratic increase near the maximum value. Peaks at 105,000.

Our tests scaling N on Wheeler

Our testing with P and Q confirmed that the largest N the system could handle would attain the highest performance,

Server	NB	N	P	Q	Results
Hopper	196	178000	8	16	5.4088e+03
Wheeler	96	383620	9	44	3.9591e+03

Table 3. Optimal Achieved Results

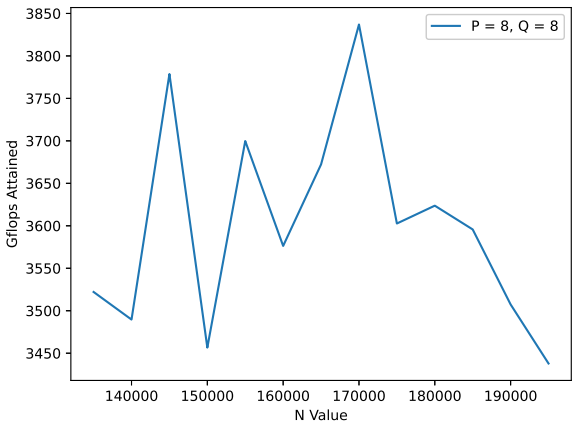

Fig. 6. Change in performance from changes in N value on Hopper for $NB = 196$, N value increased by 5,000. Highly erratic, arguably oscillatory behavior with a seemingly sustained decline at $N >= 185,000$. Peaks at $N = 170,000$

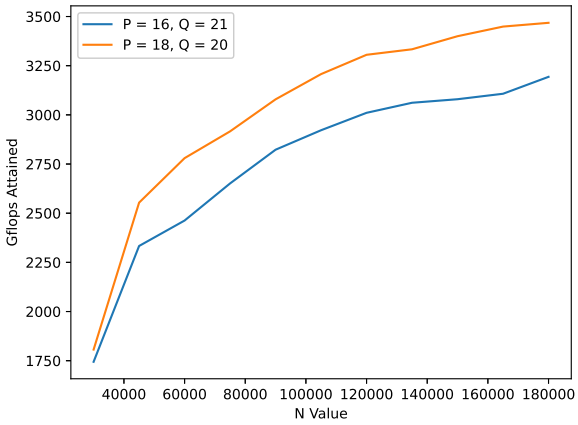

Fig. 7. Change in performance from changes to N value on Wheeler with $NB = 116$ for two different P and Q pairs. Semi-logarithmic pattern present in both data sets, $N >= 210,000$ resulted in the processes either reaching the time limit of thirty minutes or attaining critical memory use and being killed, returning no results either way

CONCLUSIONS

It was not surprising to us that larger N values provided for greater performance overall, as this allows the cluster to operate on a large enough problem size to reach its maximum throughput after surpassing any overhead that takes place by initializing the matrices in memory and distributing the workload across nodes. Looking at the graphs of N versus performance in Gflops, larger and larger values continued to give us greater ultimate throughput, while maintaining below the upper limit for available total memory in a particular node configuration.

On the other hand, we made several discoveries about the other three parameters we studied in these experiments. Both Hopper and Wheeler clusters exhibited marked fluctuations in performance by varying the NB block size parameter. Wheeler's performance profile was especially interesting in relation to this value, as it shows a wild reduction in Gflops attained at various intermediate values until it drops off at an asymptotic value near 260. As our initial motivation for this research had indicated, large values of NB will not be optimal, and this appears to agree with this notion for Wheeler and Hopper. In our sweeps we also discovered that this value should be equivalent to the total number of CPUs allocated per each run.

As for values P and Q, these had noticeable differences with variable values with a large influence on Gflop performance as few combinations provided good results and other non-optimal combinations provided significant drops in Gflops. As shown in our results, we diverged from the recommendations to keep them fairly equal while maintaining $P <= Q$ to take advantage of row-major array caching being compiled in C. We found the the formula for determining PQ values in the ideal arc would be to make a list $P = 1 : \sqrt{(tasks)}$ and then $Q_i = floor(tasks/P_i)$. This is not guaranteed to be more optimal than using the factors of N but makes for a interesting additional avenue of optimization to test.

Overall, despite Hopper being a newer cluster with faster processors and more memory per node, Wheeler actually performed almost equivalently. We can attribute this to the fact that Hopper appears to have higher general utilization during our experiments. If we were to have exclusive control and access to the Hopper nodes, we could probably have attained even higher maximum performance. For the time being, we were only able to allocate up to 3 nodes per parameter sweep with the time constraint given to us, while Wheeler could be almost fully allocated.

AUTHOR CONTRIBUTIONS

{Adrian Abeyta} ran initial tests on the Hopper cluster, determined its optimal base value for NB, and ran additional sweeps to find that this value is in the best range for throughput performance. They also researched and found a formula for the optimal problem size N as it relates to available memory. Lastly, they wrote the abstract, introduction, and conclusion, as well as editing the LaTeX file for submission.

{Roger Davis} wrote the code that produced 5, 6, 2, 3, 7, and 1. Additionally he wrote the majority of the Methods and Results sections, and contributed to editing the LaTeX file for submission.

{Andrei Popa-Simil} wrote the code that produced 4 and 2. Additionally, wrote sections in Methods, Results and Conclusions. Additional work done in excel sheet "Wheeler HPL Tests.xlsx" from which optimal Wheeler values for NB,P,Q, and N were generated and tested. Ran about 1300 tests on Wheeler.