Project 2 : HPCG and Singularity

MAISY DUNLAVY

This report presents an investigation into the scaling of the High-Performance Conjugate Gradient (HPCG) benchmark across both Ahmdal's and Gustafson's scaling paradigms as well as GPU and CPU implementations for escalating problem sizes. Leveraging Ahmdal's scalability model, the this report explores the limitations imposed by serial portions of the algorithm, focusing on how these impact overall performance with increasing system resources. Conversely, Gustafson's scalability model is employed to examine the performance improvements achievable by scaling problem sizes alongside system resources. Comparing HPCG performance on CPUs and GPUs reveals distinctive computational efficiencies and why there is increasing importance placed on GPU programming. These insights contribute to advancing the understanding of high-performance computing strategies and their practical applications.

ACM Reference Format:
Maisy Dunlavy. 2024. Project 2 : HPCG and Singularity. 1, 1 (December 2024), 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

INTRODUCTION

High-Performance Conjugate Gradient (HPCG) is a benchmark designed to evaluate the performance of modern supercomputers and high-performance computing (HPC) systems. Unlike traditional benchmarks that primarily focus on floating-point operations per second (FLOPS), HPCG assesses a broader range of system capabilities, including communication, memory bandwidth, and latency. By simulating the iterative solution of a sparse linear system using the Conjugate Gradient method, HPCG provides valuable insights into the real-world performance of HPC systems, helping researchers and engineers optimize their designs for a wide range of scientific and engineering applications.

METHODS

RESULTS

Ahmdal's scaling underscores the limitations imposed by the sequential portions of the algorithm on overall performance improvements with increased parallelization. Named after computer architect Gene Amdahl, this scaling model posits that the speedup of a parallelized computation is fundamentally constrained by the fraction of the algorithm that must be executed serially. In the context of HPCG, which involves iterative sparse linear system solutions, certain computational steps inherently remain sequential, thus limiting the scalability achievable with additional processing elements. As such, Ahmdal's scaling highlights the importance of identifying and optimizing these serial bottlenecks to maximize the benefits of parallel execution on modern high-performance computing architectures. Figure 1 shows my results from Ahmdal's scaling. These results were not what I origionally thought they would be, which I suspect is due to me choosing too small of a matrix size to keep constant (32x32x32). We see a dip in performance for 16 CPUs, then a performance increase that hits its peak at 32 CPUs which is the max number of CPUs on a single node. Once we started multinode runs, there was a dropoff in performance.

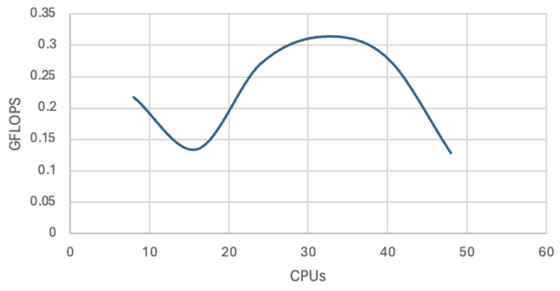

Fig. 1. Ahmdals's scaling

Gustafson's scaling emphasizes the scalability of problem sizes alongside expanding computational resources. Unlike Ahmdal's law, Gustafson's model suggests that larger problems can be efficiently addressed with an increasing number of processors. In the domain of HPCG, this scaling concept implies that as the count of CPU cores or threads rises, more substantial problem sizes can be tackled effectively within reasonable timeframes. Consequently, the Gustafson approach encourages parallelism exploration to leverage the full computational potential of modern multicore CPU architectures, facilitating the resolution of more intricate problems and driving advancements in scientific and engineering simulations. Figure 2 illustrates the results I got for Gustafson's scaling. I started with a problem size of 16x16x16 with 16 CPU's and worked up to 64x64x64 with 64 CPUs. Increasing the resrouces along with the problem size showed marginal increased at first, but once I got to a problem size large enough that I had to use multiple nodes for, I saw a huge performance increase.

When comparing CPU and GPU performance in terms of GFLOP output results for the High-Performance Conjugate Gradient (HPCG) benchmark, distinct differences emerge due to the architectural disparities between the two processing units. Generally, GPUs tend to

Author's address: Maisy Dunlavy.

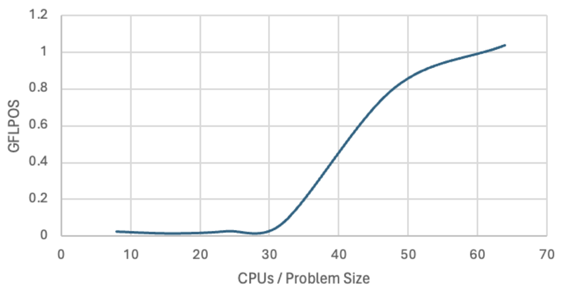

Fig. 2. Gustafson's scaling

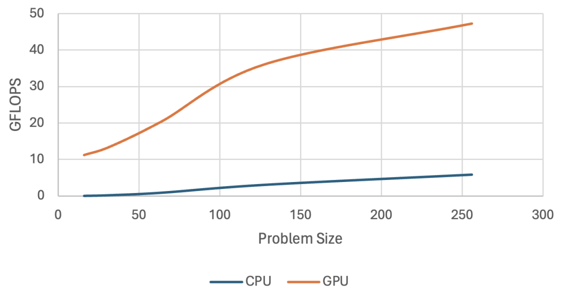

Fig. 3. CPU vs GPU output as a function of problem size

outperform CPUs in terms of raw floating-point operations per second (GFLOPs) due to their massively parallel architecture optimized for data-parallel workloads like matrix operations. GPUs excel at executing large batches of computations simultaneously, leveraging thousands of cores to achieve high throughput.

The GPU version of HPCG that we used was through a singularity image that NVIDIA distributes. Using this Singularity image ensures consistency and reproducibility across different GPU architectures. To use this image, we did a singularity pull and copied over the .sif file, .dat file and run script from the image to our run directory on Hopper. Due to limited resrouces, I chose to write a quick slurm script and submit my test runs with batch jobs rather than interactive runs and set up the mail option for slurm so I got notified when any jobs finished. I did run into some issues with file paths that I was able to fix running on the debug queue. After I resolved those, I started to get errors pertaining to memory, so I switched over to running on the condo queue. Many of my jobs were killed halfway through, so my results are not as tourough as I would like. I intended to explore much larger problem sizes and see how close I could get to the theoretical max output for GPUs on the condo queue, but my jobs were getting killed so frequently that I had to cut my scope down to a 256x256x256 problem size.

CONCLUSIONS

The data from the Ahmdal's scaling study underscores the influence of serial components within the algorithm on overall performance enhancements with increased parallelization. As the number of CPUs is increased, there's an initial rise in GFLOPS, indicating some parallel efficiency. However, this ascent eventually plateaus, and in some cases, diminishes, suggesting that further parallelization is impeded by the sequential aspects of the algorithm. For instance, with a problem size of 32 x 32 x 32, the GFLOPS peak at 0.314247 with 32 CPUs before leveling off or declining with additional CPUs, illustrating the constraints on parallel scalability in accordance with Ahmdal's law.

The data from the Gustafson's scaling study offers valuable insights into the scalability of parallel computing systems. Unlike Ahmdal's law, which highlights the limitations imposed by serial portions of an algorithm, Gustafson's approach emphasizes the scalability of problem sizes with increasing computational resources. As the number of CPUs is increased, there is a discernible trend of increasing GFLOPS performance, indicating a positive correlation between parallelization and computational throughput. The GFLOPS exhibit a notable rise, particularly as the CPU count increases from 32 to 64, reaching 1.04024 GFLOPS with 64 CPUs. This trend underscores the potential for achieving significant performance gains by scaling problem sizes alongside computational resources, as envisioned by Gustafson's scalability model.

The comparison of CPU to GPU runs demonstrates why GPU programming is becoming increasingly important. As function of problem size, the GPU version of the code had substancially better performance at every problem size. The gap between CPU and GPU output only increased as problem size grew, making it increasingly clear that for larger scale programs with minimal unserializable code, CPUs do not compare in terms of perofrmance. This disparity in performance can be attributed to the highly parallel architecture of GPUs, optimized for data-parallel workloads like dense linear algebra computations inherent in HPCG. Despite the impressive performance of GPUs, it's essential to consider factors beyond raw computational power, such as energy efficiency, cost-effectiveness, and compatibility with existing infrastructure and software ecosystems.

In conclusion, the CPU to GPU comparisons for HPCG highlight the distinct advantages of GPU accelerators in terms of computational performance and scalability for dense linear algebra computations. As GPU technology continues to evolve and mature, coupled with advancements in software optimization and parallel computing techniques, GPUs are poised to play an increasingly pivotal role in accelerating a wide range of scientific and engineering applications, including those represented by the HPCG benchmark.

AUTHOR CONTRIBUTIONS

Maisy Dunlavy did all the parts