A Proposal of GPU-based Parallel Optimization

GPU (Graphics Processing Unit) parallel computing uses the graphics processor to make full use of the internal structure of the GPU to improve computing efficiency. At present, people have proposed many models of GPU parallel computing but standard parallel algorithms are still not efficient enough to run on GPU. This proposal is about the GPU based parallel optimization technology. Aiming to make possible solutions to the problem which will let standard parallel algorithms increase the running efficiency on the graphics processing unit. We are considering different approaches to increase the efficiency and make possible projections on each method. Through this proposal we can see optimization on parallel computing can effectively improve the execution efficiency of the algorithm on the GPU, which will significantly save time when running a system.

Current intelligent optimization algorithms are based on the principle of nature. It mainly includes simulated annealing algorithm, genetic algorithm, tabu search algorithm.

Simulated annealing is derived from the principle of solid annealing, which is a general probability algorithm[1]. It is often used to find the approximate optimal solution in a large search space within a certain period of time. It is a random global search algorithm based on a local search algorithm. At the beginning, an initial temperature value needs to be set. The initial temperature will directly affect the result of the algorithm optimization. However, the algorithm implementation efficiency is low since the algorithm requires a higher initial temperature, a slower cooling rate, a lower end temperature, and a sufficient sampling frequency at each temperature[2].

Genetic algorithm is a search algorithm used to solve optimization in computational mathematics, and it is a kind of evolutionary algorithm. It is a multi-point parallel iterative process. Firstly, select the initial life population, and then repeat the following operations in each iteration: 1. Evaluate the fitness of individuals. 2. Select the next species based on the principle of proportionality 3. Change this population. Repeat the process until a certain convergence criterion is met. However, the algorithm may only get local optimum when the fitness function is not properly selected. The number of the initial population is also very important. If the number of the initial population is too large, the algorithm will occupy a lot of system resources; if the number of the initial population is too small, the algorithm is likely to ignore the optimal solution[3].

Tabu search is a modern heuristic algorithm, proposed by Professor Fred Glover of the University of Colorado in the United States around 1986. It is a search method used to escape the local optimal solution[4]. This algorithm, is modeled on human intelligence activities, introduces a data storage structure, avoids repeated searches according to some taboo criteria, so as to ensure the global optimization of search results, and improve the effectiveness and accuracy of the search[5]. The disadvantage of this algorithm is that it is highly dependent on the initial solution. A poor initial solution will reduce the convergence speed of the tabu search, and the searched solution is relatively poor.

To realize the optimization of the GPU parallel algorithm, we designed following plan to to achieve it step by step:
1.Understand the existing trends by analyzing the existing parallel algorithms.
2.Collect the experimental content and results of these technologies.
3.Summarize different ideas of optimization strategies by analyzing the experimental results and gathering the critical points of GPU performance optimization.

We studied the following five typical experimental results of GPU-based modern parallel algorithms. They are the parallel simulated annealing algorithm experiment [6], the tabu search algorithm experiment [7], the genetic algorithm experiment [8], the particle swarm algorithm experiment [9], the BP neural network algorithm experiment [10]. Through comparison, we found that the GPU does not significantly improve the performance when the number of tasks or the iterations is small. As the scale of processing data increases and the iteration scale increases, the GPU acceleration effect is noticeable, and the maximum can reach 1000 times.

We believe that the root cause of the performance difference can be summarized in two aspects through analysis. One is that when the number of data processing is small, the main component of the time of the model is dominated by the data communication time, and several threads cannot be measured. It effectively hides the relatively significant time delay. Second, when the amount of data is large, the main component of the time of the model is dominated by the calculation time, and the parallelism between a large number of multi-threads can considerably hide the data communication time delay. The greater the throughput of data processing, the less time delay, the better the effect.

I. Collaborative optimization

Modern optimization algorithms are continuously developing toward highly diversified, parallelized, and cluster intelligence optimization. With the expansion of CPU multi-core processors and the optimization and improvement of GPU chip integration technology, the algorithm execution architecture will now be more inclined to the CPU/GPU cluster collaboration model. Using multi-CPU/GPU collaborative parallel computing, we can design a reasonable data organization structure and make the multi-GPU processing peak close to the theoretical best performance.

II. Software optimization

Generally speaking, the performance influencing factors for CPU/GPU collaborative parallel computing programs include computing kernel organization method, thread organization method, registers and caches, global memory access characteristics, GPU-CPU synchronization, and GPU-CPU data transmission[11]. Accordingly, we believe that GPGPU program performance optimization has the following directions:

1) Make full use of the GPU's zero-overhead process switching characteristics to ensure sufficient fine-grained parallelism to hide overheads such as memory access latency;
2) Use shared memory to achieve efficient blocks Inter-thread communication, try to avoid using high-latency global synchronization operations (global memory read and write);
3) Study the data locality and memory access mode of the program, make full use of the GPU memory level, especially the use of cache;
4) Design a reasonable data structure, make full use of GPU hardware textures, and accelerate the access of large blocks of read-only data;

III. Hardware optimization

Both CPU and GPU have the problem of storage walls [12]. CPU mainly uses a multi-level storage structure to alleviate this problem, while GPU uses hardware multithreading technology to hide high-overhead memory access latency. Consequently, more efficient parallel algorithms should have the following characteristics:
1) Heterogeneous perception: Design algorithms based on the characteristics of the underlying hardware to maximize the performance of the architecture algorithm combination;
2) High computational intensity: High computational intensity is the high computational efficiency of parallel programs. It is a general requirement, which is especially important for GPUs. Otherwise, the GPU's high floating-point computing performance advantages will not be fully utilized;
3)Less interactions between CPU and GPU (including data transmission overhead and synchronization overhead): The interaction between CPU and GPU is inevitable for collaborative parallel computing. Optimization algorithms should be used to reduce the number of data transmissions and the amount of data and synchronization overhead.

Based on previous analysis, efficient CPU/GPU coordinated computing is a key factor for the performance of heterogeneous hybrid platforms. Therefore, a reasonable coordination method must be determined according to the computing capabilities and execution characteristics of the two to ensure the balance of computing load between CPU and GPU and reduce various interaction overheads.

The potential improvements include the partition model of computing tasks, the scheduling strategy of tasks, and the mapping relationship between computing tasks and hardware resources. Reasonable task division is the basis for efficient collaboration. Also, it is necessary to comprehensively consider multiple factors such as the computing power, calculation amount, communication and data transmission overhead of each computing resource, and divide the entire computing task into a certain number of subtasks with appropriate granularity. Task scheduling schedules various computing subtasks to idle computing resources for execution, so that the entire heterogeneous hybrid system is kept at full load at all times, so as to maximize its computing power.

In this proposal, we summarized the current status of the GPU parallel computation and raised a couple of potential improvements on the GPU-based parallel optimization. Among the three proposals, hardware optimization is hoping to have a breakthrough in the nearest future. It is likely that a better GPU will be powerful and dependent enough to hide the data transmission overhead. At the same time, algorithm innovation will also repay the hardware. We can conclude that the improvements in hardware and software performance will altogether take the parallel efficiency to a higher level.

[1] Simulated Annealing, from https://en.wikipedia.org/wiki/Simulated_annealing
[2] Qingke, Z., Yang, B., Lin, W., Fuxiang, Z.(2012, April). ResearchonParallelM odernOptimizationAlgorithmsUsingGPU. Computer Science, 39(4).
[3] Genetic Algorithm, from https://en.wikipedia.org/wiki/Genetic_algorithm
[4] Tabu Search, from https://en.wikipedia.org/wiki/Tabu_search
[5] Yaqun, C.(2019 Nov.). Research on GPU-based Parallel Optimization Algorithm. SCIENCE & TECHNOLOGY INFORMATION.
[6] Choong, A., Beidas, R., & Zhu, J. (2010, August). Parallelizing simulated annealing-based placement using GPGPU. In 2010 International Conference on Field Programmable Logic and Applications (pp. 31-34). IEEE.
[7] Czapiński, M., & Barnes, S. (2011). Tabu Search with two approaches to parallel flowshop evaluation on CUDA platform. Journal of Parallel and Distributed Computing, 71(6), 802-811.
[8] Shah, R., Narayanan, P., & Kothapalli, K. (2010). GPU-accelerated genetic algorithms. cvit. iiit. ac. In.
[9] Zhou, Y., & Tan, Y. (2009, May). GPU-based parallel particle swarm optimization. In 2009 IEEE Congress on Evolutionary Computation (pp. 1493-1500). IEEE.
[10] Lopes, N., & Ribeiro, B. (2009, September). GPU implementation of the multiple back-propagation algorithm. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 449-456). Springer, Berlin, Heidelberg.
[11] Yang, C., Xue, W., Fu, H., Gan, L., Li, L., Xu, Y., ... & Zheng, W. (2013). A peta-scalable CPU-GPU algorithm for global atmospheric simulations. ACM SIGPLAN Notices, 48(8), 1-12.
[12] Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., ... & Yelick, K. A. (2006). The landscape of parallel computing research: A view from berkeley.