Journal of Scientific and Technical Research
14 (2) 2024, 01-03
http://www.publishingindia.com/jstr

Application Runtime Prediction for Scheduling CPU-GPU: Analysis

Suman Goyat[1*] and Shri Kant[2]
[1]Research Scholar, Sharda University, Greater Noida, Uttar Pradesh, India
[2]Professor Center for Cyber Security and Cryptology, Department of Computer Science and Engineering, School of Engineering and Technology, Sharda University, Greater Noida, Uttar Pradesh, India
[*]Corresponding Author: goyat1606026@gmail.com

Abstract: In high performance computing large number of applications are executed over heterogeneous devices like CPU, GPU, and combination of more than one type of processors. Application when assigned to particular processor run-time information is recorded which is a measure-describing factor in application for task scheduling. Predicting this information prior to execution is the focus, which further distinguishes implemented approaches. It is not guaranteed that model will provide exact information but the predicted value is not always the similar to actual runtime for a specific task in distributed environment. In proposed work, here is the analysis of predicting runtime in advance so that it is clear and correct which device is suitable for which task in particular application. The predicted time is compared to actual runtime and accuracy is improved by running same task over and over again on the supposed device. The results show that historical information can be stored on the basis of prediction model which gives tolerable accuracy for the further yield of reducing waiting time and increasing processor utilization.

Keywords: Graphical processing unit, High performance computing, Load sharing, Prediction modeling, Regression.

I. Introduction

This paper contributes for creation of Historical database, which have the information about the runtime of particular application which provide support for scheduling decision. Scheduling is the basic requirement for the successful implementation and utilization of distributed systems. Even the most dominant, high performance and powerful computing environments need appropriate scheduling so as to proficiently and effectively assist the users. Also, contemporary advances in computing environments have presented new challenges to schedulers such as issues of scheduling parallel jobs amongst which there are no dependencies on a cluster of distributed processors. Further, in a distributed computing environment, the heterogeneity of new systems and modifications in execution environment many a times need dedicated scheduling techniques for every resource group. Owing to these reasons, many researchers have been looking for scheduling methods to improve and optimize the performance of distributed system. Thus, Scheduling require prior knowledge of execution timeline for the particular task whether that is compute intensive or data intensive. Here prediction model is analyzed which further explains database can be created of application with the highly accurate runtime prediction.

Linear regression is applied here for predicting runtime of particular application over CPU/GPU.

II. Details of Existing Dynamic Load Sharing Strategy

Authors have worked in this field and gained advantages over energy consumption and load sharing when the scheduler is able to decide which approach is used for reducing waiting time and reducing total turnaround time of an application. Here is description of few of work which was dedicated to making scheduling decision. Dynamic scheduling gained popularity from researchers and industries due to their high performance and complexity in fabrication operations [1]. High performance scheduling encounters significant challenges during Heterogenous computing in a distributed environment in which a task is allocated during the runtime, i.e., dynamic scheduling [2]. Dynamic scheduling also significantly supports the heterogeneous parallelism and distributed environment. He developed a dynamic task scheduler to reduce the makespan in a distributed environment. The authors reported that the dynamic task scheduler could improve the overall makespan by 31% by optimising unbalanced workload [3]. Further, various studies have presented distinct dynamic scheduling algorithms to achieve high performance during resource management [4, 5, 6] . In recent years, researchers have focused on heterogenous processors such as central processing unit (CPU) and graphical processing unit (GPU) to meet the performance challenges related to resource utilisation, execution time, latency [7, 8].

III. Support of Algorithm for Scheduling in Particular Application

A research was conducted on CPU-GPU architectures using parallel and accurate k-means algorithm. For CPU and GPU, the author proposed parallel optimization techniques for the k-means algorithm. Mainly the author took into consideration a two-step summation method with package processing to take hold of the effect of rounding errors that may happen during the phase of apprising cluster centroids. These tryouts on artificial and real-world datasets comprising of millions of examples display a speedup up to seven for the k-means repetition time on GPU versus 20/40 CPU threads using AVX units. This facilitates in attaining double-precision correctness with single-precision processing [9].

A research conducted on Autonomous Driving Platform which is considered to be cutting-edge embedded system applications. The research focused on performance and memory footprint optimization via CPU-GPU memory management. The research stated that for DNNs-driven workload, technologies like unmanned drone and self-driving cars are dependent on integrated CPU-GPU platforms. Here, DNNs-driven workload includes discernment and further highly parallel components. The present research has been focused on to assess the unseen performance consequence of GPU memory management approaches of combined CPU-GPU architecture. To work in this direction, the research applied and far assessed the proposed system prototype on the NVIDIA Jetson TX2, Drive PX2, and Xavier AGX platforms. Herein, both Rodinia benchmark suite and two real-worlds case studies of drone software and independent driving software [10].

IV. Related Work Done

Researcher proposed forecast model after taking into account the past real execution time of each process set, they compute the average time of each individual set of processes by taking into account variables like the total number of processes, the sum of their execution times, and the number of threads each process is assigned. Next, they assess each set of processes prediction times for the CPU and GPU by taking into account variables including the number of processes, their weight, and the average time of the prior set of processes. The GPU is assigned resources based on the process's maximum predicted duration [11].

V. Methodology

DHSGC is the non-pre-emptive approach and sequential steps are as follows: The algorithm is: The majority of applications will operate more quickly when allocated to the GPU, which is determined based on past database data, out of the two accessible devices. Utilizing insights from parallel computing to design a new, extremely effective scalable scheduling system. Our goal is to analyze the scheduling of the application framework using statistics and dynamics. Additionally, this will enable the continued application of novel approaches in the planning and realization of heterogeneous environment goals. The program used to create historical datasets is called MSI Afterburner. Here is a table that we took a few applications over in our previous work to generate a historical database over which we have to make predictions.

Table I: Runtime Information

Application	Actual Runtime (Time in ms) CPU
DCT	503
FloydWarshall	527
SimpleConv	481
QuasiRandSeq	441
Coloring Of Graph	100
AMG Solver	350
DwtHaar1D	924

VI. Runtime Prediction Model

When an application is run with the same input size after it has been assigned to a device with a specific set of inputs at least once, the scheduler utilizes the average application time to anticipate the runtime of that device. The scheduler uses a linear least-squares computation to anticipate the outcome when an application has been run with different inputs on a specific device at least twice and is then run again with a new input. This calculation takes two parameters into account: the input sizes and the prior runtimes. The actual runtimes and the expected runtimes are displayed in a bar graph. For the applications with more than 50 trials, the average runtimes are indicated by the height of the left-side bars. The bars on the right side indicate the expected time following the scheduler's training. We were able to use the anticipated runtimes for scheduling purposes because they were within the actual minimum and maximum runtimes in our studies.

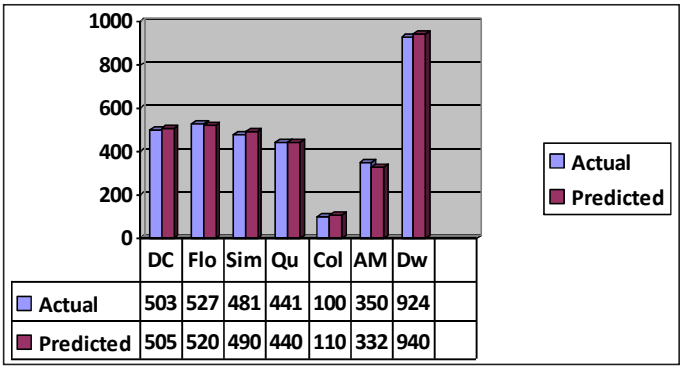

Fig. 1: Graphical Representation of Runtime

After prediction we are required to train and test our scheduler on the basis of reducing the waiting time of process in distributed environment. We used the identical applications as in our benchmark to train the scheduler for our trials, but we varied the input sizes. In order to standardize the results for Fig. 2, we additionally selected a sample fixed set of input sizes to test with the database following each training session. Applications are first allocated to the CPU if available and to the GPU if not, based on little or no past data. The scheduler performs poorly in the initial runs when compared to a GPU-only scheduling approach, but it quickly improves. As the database gathers information on every application it encounters, it continuously undergoes cross-training, and the scheduler gains advantages each time it encounters a new application than once, irrespective of the size of the dataset. For this group of applications, the scheduler always outperforms a CPU-only solution. The data used in the results section below was collected after five training runs.

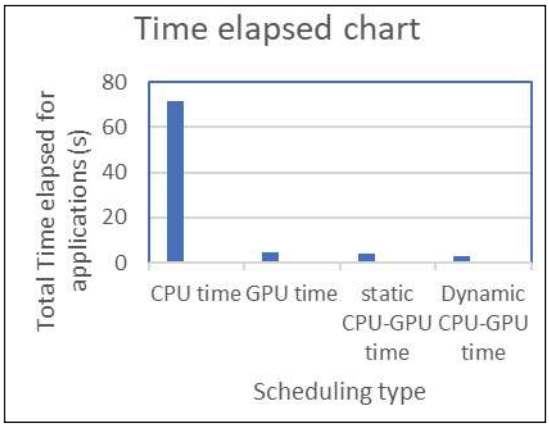

Fig. 2: Scheduler Decision

VII. Conclusion

The prediction modelling can be helpful for making decision regarding selection of scheduling approach whether that is task scheduling or processor scheduling. It is used to predict CPU or GPU load to prevent overloading and ensure optimal performance. Here the results show the accuracy is attained approx. 90%. In future it can be improved by selecting the different application data size and platform for completing on heterogeneous environment.

References

[1] B. Firme, J. Figueiredo, J. M. C. Sousa, and S. M. Vieira, "Agent-based hybrid tabu-search heuristic for dynamic scheduling," *Engineering Applications of Artificial Intelligence*, vol. 126, 2023, doi: https://doi.org/10.1016/j.engappai.2023.107146.

[2] H. Tayeb, B. Bramas, A. Guermouche, and M. Faverge, "MulTreePrio: Scheduling task-based applications for heterogeneous computing systems," *Conference in Parallelism, Architecture and Systems,* France, 2022, pp. 5-8.

[3] H. Tayeb, B. Bramas, M. Faverge, A. Guermouche, and D. Tasks, "Dynamic tasks scheduling with multiple priorities on heterogeneous computing systems," *HAL Open Science*, *hal-044986*, 2024.

[4] Y. Chen, S. Liu, Y. Chen, and X. Ling, "A scheduling algorithm for heterogeneous computing systems with edge cover queue," *Knowledge-Based Systems*, vol. 265, 2023, doi: https://doi.org/10.1016/j.knosys.2023.110369.

[5] H. Hafsi, H. Gharsellaoui, and S. Bouamama, "Genetically-modified multi-objective particle swarm optimization approach for high-performance computing workflow scheduling," *Applied Soft Computing*, vol. 122, 2022, doi: https://doi.org/10.1016/j.asoc.2022.108791.

[6] N. Sehgal, S. Bansal, and R. K. Bansal, "A comparative analysis of dynamic scheduling algorithms for enhanced resource management in homogeneous and heterogeneous fog computing environments," *Procedia Computer Science*, vol. 230, 2023, doi: https://doi.org/10.1016/j.procs.2023.12.110.

[7] J. Fang, K. Zhou, M. Zhang, and W. Xiang, "Resource scheduling strategy for performance optimization based on heterogeneous CPU-GPU platform," *Computers, Materials and Continua*, vol. 73, no. 1, 2022, doi: https://doi.org/10.32604/cmc.2022.027147.

[8] M. Amaris, R. Camargo, D. Cordeiro, A. Goldman, and D. Trystram, "Evaluating execution time predictions on GPU kernels using an analytical model and machine learning techniques," *Journal of Parallel and Distributed Computing*, vol. 171, 2023, doi: https://doi.org/10.1016/j.jpdc.2022.09.002.

[9] G. He, S. Vialle, and M. Baboulin, "Parallel and accurate k-means algorithm on CPU-GPU architectures for spectral clustering," *Concurrency and Computation: Practice and Experience*, 2021, doi: https://doi.org/10.1002/cpe.6621.

[10] S. Bateni, Z. Wang, Y. Zhu, Y. Hu, and C. Liu, "Co-optimizing performance and memory footprint via integrated cpu/gpu memory management, an implementation on autonomous driving platform," *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2020, pp. 310-332.

[11] B. N. Chandrashekhar, M. Mohan, and V. Geetha, "Forecast model for scheduling an HPC application on CPU and GPU architecture," in *2023 3rd International Conference on Intelligent Technologies (CONIT),* IEEE, Jun. 2023, pp. 1-5.