Task Scheduling in Graphic Processing Units Heterogeneous With Density of The Graph

Xuecheng Wang1*, Jinquan Ma1, Chunsheng Yue1 and Di Yang1

1Departments of information systems engineering, PLA Strategic Support Force Information Engineering University, 450002, Zheng Zhou, China
*Corresponding author’s e-mail: iota9411@foxmail.com

Abstract. An efficient scheduling algorithm is significant to achieve high performance in heterogeneous consisted by the central processing units (CPU) and the graphic processing unit (GPU). However, most of the scheduling algorithms are not suitable to meet the present demands of the heterogeneous. According to the characteristics of the direction acyclic graph, the direction acyclic graph nodes density list scheduling algorithm (DAG-NDLS) was proposed. The algorithm had two important phase, the first phase was task prioritizing, selected the ready task with highest priority, defined by the summation of the node density and the transmission between the node and child nodes, and the second phase was processor selection, selected the processors for the tasks with the earliest finish time minimum. In the end, compared the performances of the proposed algorithm with the heterogeneous earlier finish time algorithm and the high performance task scheduling algorithm, the results of the proposed algorithm were the best in case of the efficiency and utilization.

1. Introduction

In the field of high performance computing (HPC), the heterogeneous consisted by the central processing units and the graphic processing units (CPU+GPU) are the mainstream for the next generation heterogeneous. Unfortunately, to face the high transmission latency and the high demands of computational efficiency, common scheduling algorithms work poorly on the CPU+GPU heterogeneous.

The classical task scheduling algorithms include the dynamic scheduling and static scheduling [1]. And the dynamic scheduling algorithm is a kind of real-time scheduling so that a lot of computing resources are wasted at the time of execution, on the contrary, the static scheduling is more stable and easier. What is more, the list scheduling heuristics [2]-[5], clustering heuristics [6][7], task duplication heuristics [8][9] and guided random search based scheduling algorithm [10][11], all of them are kinds of static scheduling algorithms played an important role on the particular platforms. Furthermore, the list scheduling heuristics are the most widely used on various HPC platforms with the lowest complexity. The heterogeneous earlier finish time (HEFT) algorithm [5] and the high performance task scheduling algorithm (HPS) [12] are belong to list scheduling algorithm, and outperform almost other algorithms in case of complexity and efficiency. However, both of them work poorly with the high intercommunication latency and the consistent model [13] of the CPU+GPU heterogeneous. Hence, according to the characteristics of the CPU+GPU heterogeneous and the feature of the direction acyclic graph (DAG), the direction acyclic graph nodes density list scheduling algorithm (DAG-NDLS) was proposed. And the algorithm is also a kind of list scheduling algorithm, and traversed the DAG hierarchical. The DAG-NDLS had three main steps. The first step was layering for the DAG by topological sorted. The second step was sorting the task hierarchical with the task priority assigned with
the summation of graph density and the amount of transmission from the node to its child nodes. In the third step, the tasks were mapped to the processors with the minimum completion time. As a result, the intercommunication latency is decreased by the efficient execution order constructed by the three steps.

Evidently, the performances of the DAG-NDLS were best, which were compared with the HEFT and the HPS on various experiments, and the result had shown that the DAG-NDLS had the maximizing utilization and minimizing completion time.

2. Abstraction and Modelling
A scheduling system model consists of a heterogeneous environment, applications and performance metrics for scheduling. And then, we will cover the scheduling system separately.

2.1. Hardware Models
This section presents what the CPU+GPU heterogeneous platforms are, and the figure of the hardware structure is shown as fig1.

![Figure 1. The model of the hardware structure](image)

There are two kinds of ways to implement multi-GPUs communication. One way is communicating with each other by the PCIE bus, and the other way is communicating through the CPU. And the latter way is the mainstream for the limited resources of PCIE bus. However, the latency of the latter way is higher than the former. And the high latency makes a large number of common scheduling algorithms useless.

2.2. Model of the Task Priority
The application was represented by the DAG, the DAG of the DAG-NDLS was defined as \( \text{DAG} = [V, E, C, TC] \), where the \( V \) denotes the aggregate of the nodes, which are equal to the tasks in the DAG, where the \( E \) denotes the aggregate of the edges, and the \( E_{ij} \in E \) denotes the edge from the \( V_i \) to \( V_j \), where the \( C \) is the aggregate of the computation of the nodes and the \( TC \) is the aggregate of the amount transmission to the child nodes. Furthermore, the processors of the DAG-NDLS was defined as \( P \), where the \( P_i \) denotes the \( i \)th processors, and the computation of the \( P_i \) is defined as \( P_{Ci} \).

Furthermore, the priority of a node \( V_i \) in the DAG-NDLS was assigned with the summation of the transmission of the \( V_i \) and density of the \( V_i \). And the priority \( Pr \) was defined by

\[
Pr = ND_{Vi} + TC_{Vi}
\]

(1)

The \( ND_{Vi} \) is the density of the \( V_i \). And the density of the \( V_i \) was defined by

\[
ND_{Vi} = EN_{Vi}/VN_{Vi}^2
\]

(2)

The \( EN_{Vi} \) and the \( VN_{Vi} \) are the amount of the edges and the nodes of the \( V_i \) graph, respectively. And the \( V_i \) graph is a child graph, the \( V_i \) as the only entry node of the graph, and finding all of the nodes and edges on the path from the entry node to the exit node as the child graph nodes and edges.

2.3. Metrics
The performance of the algorithm was judged by following metrics, the Improved Scheduling Length Ratio (ISLR) and the Variance of the Amount of Task on the Processors (VATP). And the ISLR was based on the SLR proposed by the literature [13] defined by
The MakeSpan(Solution) denotes the total execution time of the Solution algorithm, and the SNE is the summation of the cost of the nodes and the edges defined by

$$SNE = \sum_{i} C_i + \sum_{i} TC_i$$ (4)

The VATP was defined by

$$\text{VATP} = \frac{\sum_{i} (\text{Task}_i - \overline{\text{Task}})^2}{\text{PN}}$$ (5)

The Task$_i$ is the amount of the task on the $i$th processor, the $\overline{\text{Task}}$ is the average of the task on the processors and the $\text{PN}$ is the amount of the processors. By the way, the less, the better for both metrics.

3. The Algorithm of The DAG-NDLS

The DAG-NDLS is a kind of list scheduling algorithm. It is aims to maximizing utilization and minimizing completion time. And the DAG was defined by the adjacency matrix $M$, where the $M_{ij}$ are the elements of the $M$, which denotes the connection relationship between the $V_i$ and $V_j$, and the values of the $M_{ij}$ are defined by

$$M_{ij} = \begin{cases} 1 & \exists E_{ij} \in E \\ 0 & \text{otherwise} \end{cases}$$ (6)

And there were three main steps for the DAG-NDLS. Firstly, the DAG was layered by the topological sort. And the topological sort found all of the nodes $V_i$, that $V_i \in \left\{ V | \sum_{i} M_{ij} = 0 \right\}$, then saved them in the same aggregate, defined as $V_{\text{avo}}$, where the $k$ denotes the layers. And updated the $M$, set the $M_{ij} = 0$, for $j = 0, 1, 2 \ldots$ and the $i$ is the subscript of the $V_i \in V_{\text{avo}}$. And then, iterated again and again until all of the elements of the $M$ changed into zero.

The second step was setting the priority, and sorting the nodes by decreasing order of priority for every layer. Moreover, the DAG-NDLS set the nodes priority according to the equation(1). And the nodes priority were constructed by the nodes density and the nodes transmission. The value of the node density were computed by the equation(2), and the values of the $EN$ and the $VN$ were calculated by traversing the node graph in downwards direction starting from the node as an entry node and traversing until the exit node had not came. And the nodes transmission were saved in the nodes information. And the nodes lists for every layer were generated by sorting the nodes by decreasing order of priority.

The third step was mapping the processors for the nodes. It put constrains to map the node to the appropriate processors and aims to maximizing utilization. Mapping the nodes to the processors had to follow the nodes list. Started with the highest priority node, $V_i = \max_{V_i \in V_{\text{avo}}} \left\{ V_i \in V_{\text{avo}} \right\}$, for every time, compared the completion time of the selected node to processors and found that the minimizing completion time processor, $P_i = \arg \{ \min \{ \text{CT}(V'_i) \} \}$, and where the $\text{CT}(V'_i)$ is the minimizing completion time of the $V'_i$, and then, mapped the node to it. And the $\text{CT}(V'_i)$ is defined by

$$\text{CT}(V'_i) = \max_{P_i \in P} \{ R(V'_i), R(P_i) \} + \text{TC} / \text{PC}_i$$ (7)

And where the $R(V'_i)$ and the $R(P_i)$ are the earliest ready time of the $V'_i$ and the $P_i$, respectively.

Undoubtedly, it is helpful for the three steps to decrease the searching time, resolve the load balancing problems and improve the overall performance.
4. Simulation Results
This section presents the simulation results. We compared the proposed algorithm to the HEFT and the HPS on the classical DAG and a set of random DAGs, and evaluated the results with two different metrics, ISLR and VATP.

4.1. Classical Direction of Graphs
To validate that the DAG-NDLS worked well than the HEFT and the HPS, compared the performance of the three algorithm on the classical DAG proposed by the literature [5]. And the time of the nodes executed on the processors are displayed in table1. The classical DAG is showed as fig2. The Node ID is the name of the nodes in the DAG, and the P1, P2 and P3 are the processors name, the AverTime is the average time of the task executing on the three processors. According to the three algorithm, we obtained three schedules displayed on the fig3, And the result of the ISLR and the VATP are shown in the table2 and table3, respectively.

![Figure 2. The classical DAG.](image)

| Node ID | P1   | P2   | P3   | AverTime |
|---------|------|------|------|----------|
| V1      | 0.48 | 0.55 | 0.93 | 0.65     |
| V2      | 89.07| 102.13| 174.09| 121.76   |
| V3      | 73.65| 84.45| 143.95| 100.68   |
| V4      | 56.81| 65.15| 111.05| 77.67    |
| V5      | 97.23| 111.49| 190.05| 132.92   |
| V6      | 57.17| 65.56| 111.05| 77.92    |
| V7      | 7.80 | 8.95 | 15.25| 10.67    |
| V8      | 90.05| 103.25| 176.00| 123.18   |
| V9      | 62.19| 71.31| 121.55| 85.01    |
| V10     | 33.14| 38.00| 64.77| 45.30    |

According to the simulation results presented by fig3, the DAG-NDLS obtained the shortest schedule lengths, is equal to 363.74. And then, calculated the value of the \( SNE \) by the equation(4), took the values of the \( \text{MakeSpan(Solution)} \) for the three algorithms and the value of the \( SNE \) into equation(3) to get the values of the ISLR for the three algorithms and counted them in the table2. And took the amount of the nodes mapped to the processors into equation(5) to get the values of the VATP and counted them in table3. The simulation results showed that the DAG-NDLS is faster than the HEFT and the HPS by 5 percent and 8 percent, respectively with the same VATP.
Figure 3. Scheduling result of the classical DAG.

Table 2. The ISLR of the HEFT algorithm, HPS algorithm and DAG-NDLS algorithm on the classical DAG

| ALGORITHM | HEFT       | HPS        | DAG-NDLS  |
|-----------|------------|------------|-----------|
| ISLR      | 0.3044     | 0.3140     | 0.28678   |

Table 3. The VATP of the HEFT algorithm, HPS algorithm and DAG-NDLS algorithm

| ALGORITHM | P1 | P2 | P3 | VATP |
|-----------|----|----|----|------|
| HEFT      | 5  | 3  | 2  | 1.56 |
| HPS       | 5  | 3  | 2  | 1.56 |
| DAG-NDLS  | 5  | 3  | 2  | 1.56 |

4.2. Random direction of graphs

This section compared with respect to various graphs size. And the randomly generated application graphs were considered with various graphs size. Furthermore, the results of the set of experiments are shown in the fig4 and fig5 for the ISLR and the VATP, respectively. By the way, we counted the probability of the scheduling success, which one was fastest on the test, the one was scheduling success. And the probability is shown in fig6. According to the simulation results, it was found that compared with the HEFT and the HPS, the DAG-NDLS gets the largest range, the highest efficiency and the maximizing utilization for the processors.

Figure 4. Scheduling results for ISLR with randomly graph
5. Conclusion
As the CPU+GPU heterogeneous platforms become mainstream, an efficient scheduling of the tasks of an application on the available processors is the important key factor for achieving high performance. And the high throughput of the CPU+GPU heterogeneous platforms is well suited to the computation needs of intensive computing, however, the efficiency of the platforms is limited firmly by the latency of the intercommunication. Hence, the DAG-NDLS algorithm was proposed. And the model of the nodes priority was proposed based on the characteristics of the DAG and the transmission of the application so that it was able to meet the high latency. And as the set of experiments have shown, the DAG-NDLS significantly outperforms the other algorithms in terms of both efficiency and the two metrics, the ISLR and the VATP. And the DAG-NDLS was demonstrated to be a viable way for application scheduling problems on the CPU+GPU heterogeneous with its stable performance.

References
[1] DU Xiaoli, JIANG Changjun, XU Guorong, DING Zhijun. (2006) A Grid DAG Scheduling Algorithm Based on Fuzzy Clustering. JOURNAL OF SOFTWARE, 17(11):2277-2288.
[2] MU. (2010) Advanced list scheduling heuristic for task scheduling with communication contention for parallel embedded systems. Science China(Information Sciences), 53(11):2272-2286.
[3] Mu P, Nezan J F, Raulet M. (2013) A List Scheduling Heuristic with New Node Priorities and Critical Child Technique for Task Scheduling with Communication Contention. Lecture Notes in Electrical Engineering, 73:217-236.
[4] Alebrahim S, Ahmad I. (2017) Task scheduling for heterogeneous computing systems. Journal of Supercomputing, 73(6):2313-2338.

[5] Topcuoglu H, Hariri S, Wu M Y. (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel & Distributed Systems, 13(3):260-274.

[6] Shin I, Easwaran A, Lee I. (2008) Hierarchical Scheduling Framework for Virtual Clustering of Multiprocessors[C]// Euromicro Conference on Real-Time Systems. IEEE Computer Society:181-190.

[7] Tsai Y L, Liu H C, Huang K C. (2015) Adaptive dual-criteria task group allocation for clustering-based multi-workflow scheduling on parallel computing platform. Journal of Supercomputing, 71(10):3811-3831.

[8] Atef A, Hagras T, Mahdy Y B, et al. (2017) Lower-bound complexity algorithm for task scheduling on heterogeneous grid. Computing, 99(11):1125-1145.

[9] Zhang W, Hu Y, He H, et al. (2017) Linear and dynamic programming algorithms for real-time task scheduling with task duplication. Journal of Supercomputing, 1(1):1-16.

[10] Zhang L, Chen Y, Sun R, et al. (2008) A Task Scheduling Algorithm Based on PSO for Grid Computing. International Journal of Computational Intelligence Research, 4(1):37-43.

[11] Jiang Y S, Chen W M. (2015) Task scheduling for grid computing systems using a genetic algorithm[J]. Journal of Supercomputing, 71(4):1-21.

[12] Ilavarasan E, Thambidurai P, Mahilmman R. (2005) High Performance Task Scheduling Algorithm for Heterogeneous Computing System[C]// International Conference on Algorithms and Architectures for Parallel Processing. Springer-Verlag:193-203.

[13] Reda N M, Tawfik A, Marzok M A, et al. (2014) Sort-Mid tasks scheduling algorithm in grid computing. Journal of Advanced Research:S209012321400143X