Journal of Computers Vol. 31 No. 3, 2020, pp. 86-99
doi:10.3966/199115992020063103007

# A Reliable Resource Scheduling Approach with Dataflow Natural Attribute Priority

Yu-Ling Fang, Qing-Kui Chen*, Jing-Juan Wang

University of Shanghai for Science and Technology, No.516 Jungong Road, Shanghai 200093, China
forwardfyl@163.com, chenqingkui@usst.deu.cn, wjj9209@163.com

Received 18 October 2018; Revised 18 February 2019; Accepted 9 April 2019

Abstract. CPU-GPU cluster computing systems are widely used to process large-scale data in various fields owing to its famous thousands of computing cores and high computation intensiveness. However, its high performance is always accompanied by high power consumption, even further resulting in reduced reliability and instantaneous failure. In this paper, a reliable cluster Resource Scheduling approach with dataflow Natural Attribute Priority (RSNAP) is proposed, and it is designed to increase the reliability of collaborative computing nodes. On the basis of the idea of genetic algorithm, it searches for an optimal task-processor allocation scheme with considering the characteristics of tasks and processors (GPUs). For tasks that have data dependencies, we reduce the load that needs to be performed by adjusting the frequency and voltage of the GPU to further reduce node power consumption without affecting the natural distribution of data in the nodes. With less loads, GPU power consumption pressure has been eased, further enhancing system reliability. While for tasks that have no dependencies, we improve system reliability through dynamic task migration. The experiment results show that RSNAP can reduce GPU power consumption and improve reliability of collaborative computing.

Keywords: cluster computing, frequency and voltage, natural attribute priority, power consumption, reliability, resource scheduling

## 1 Introduction

During the last decades, the CPU-GPU collaborative computing systems had been widely used in scientific and commercial fields such as massive amounts of data mining, Interest of Things and cloud computing [1-3]. Especially, with the development of deep learning, the computing systems are more and more important in artificial intelligence application [4]. As a major component, Graphics Processing Units (GPUs) are increasingly applied to accelerate computing. Since 1999, GPUs' main manufacturer NVIDIA released its first consumer GPUs – GeForce 256. It has now been updated to the fifth generation Pascal, and the sixth generation Volta will soon be available in 2018. The renewal and breakthrough of GPUs meets the requirements of big data processing. In the newest Top500 list [5], there are more than 50% supercomputer systems employing GPUs (including AMD GPUs) as accelerators. Take the world-famous, almost undefeated AlphaGo as an example, its latest version has nearly three hundred GPUs [6]. GPUs assist it get faster computing speed in real time.

It's no doubt that more processors will obtain higher performance. The ranked first supercomputer "Sunway TaihuLight" has more than ten million computing cores, and its Linpack Performance can reach to 93,014.6 TFlop/s. Therefore, it can solve computational challenges in different areas with huge performance advantages, for example, simulating the return path of "Tian gong-1". However, the high performance always is accompanied by high power consumption, and it is about 15.37 million W per hour. The high power consumption not only brings tremendous pressure to the heat dissipation, but also costs more money. Therefore, power consumption and reliability problem also attract more and more attentions.

* Corresponding Author

Many studies have focused on improving the energy efficiency and reliability of GPU computing systems [7-9]. The first researcher [7] classified energy techniques on the basis of different ideas. Moreover, it compared GPU energy efficient with FPGA's, and given the guidance for other researchers. The work in [8] presented a migration framework based on a virtualized environment for improving system reliability, and it was demonstrated to provide low migration overheads through four application kernels. Task migration requires not only additional idle GPU resources but also no dependencies between data streams. So, it does not apply to all applications, especially for dataflow of applications that have spatial or time-dependent. While for tasks with dependencies, researchers proposed different energy efficiency optimization methods. The work in [9] proposed power consumption efficient work assignment method for single application on a GPU system, and it minimized energy consumption by adjusting processor's frequency and coordinating inter-processor work assignment. Meanwhile, a synthesized source code optimization method was studied by the work [10], and it evaluated 128 versions and found the optimal result. The work in [11-12] presented the Dynamic Voltage Frequency Scaling (DVFS) techniques to adjust GPU performance and power efficiency. The former surveyed the state-of-the-art GPU DVFS characteristics, and summarized research works on GPU power and performance models. It verified that GPU DVFS has significant potential for energy saving and improving system reliability. The latter presented a lot of literature research and analysis on DVFS techniques, and it found that DVFS not only work solely but also work with other techniques, such as load balancing and task mapping.

In this paper, we mainly focus on the reliability problem of cluster computing system that caused by high power consumption and long running applications. For tasks with data dependencies, based on Genetic Algorithm (GA) and heuristic idea, we propose a reliable RSNAP. Further, in order to improve reliability and computing persistent, we adjust GPU frequency and voltage to reduce workload. Meanwhile, for tasks without data dependencies, we introduce the dynamic task migration for improving system performance. In order to calculate and analyze the power consumption of a single GPU or GPU, we use a GPU Cluster Power Consumption Calculation (GCPC) model based on wireless sensor networks.

The rest of this paper is organized as follows. In section 2, related work is reviewed. Section 3 presents our basic theoretical models. In Section 4, RSNAP is outlined to reduce GPU power consumption and improve the system reliability. Experiments results and analyses are given in Section 5. Finally, we conclude this paper in Section 6.

## 2 Related Work

As we known, the task scheduling is a NP-hard problem [13], and the selected best solution is always approximately optimal. Therefore, heuristic and meta-heuristic algorithms are always applied to solve this problem. The work in [14] presented two parallel algorithms to solve the independent tasks scheduling problem. It combined the advanced Min-min heuristic and parallel cellular GA to take advantage of the massive parallelism of the GPU for improving performance. Hamano et el. [15] proposed a scheme to optimize overall energy consumption by scheduling runtime and energy of each scheduling decision, and it improved energy effectively based on its acceleration factor. An effective reliability-driven task scheduling model was built based on weibull distribution, and it could measure the task reliability of GPU cluster with arbitrary networks architecture [16]. Most of task scheduling problems depend on heuristic thinking or GA. The work in [17] presented two developed genetic algorithms that combined heuristic principles and load balance to solve different task scheduling problems, respectively. The work in [18] proposed a task scheduling scheme using a multiple priority queues genetic algorithm, and it exploited the ideas of evolutionary-based and heuristic-based to improve system performance. In addition, GA also plays an important role in solving some common problems, such as P-Median problem [19].

GA can be used to address task scheduling problems in different fields. The work in [20] proposed a powerful and improved genetic algorithm to optimize task scheduling solutions in cloud computing. It exploited the advantages of evolutionary genetic algorithm and heuristic approached to obtain the optimal scheduling solution. In order to verify the validity of proposed solution, it presented a behavioral modelling method based on the idea of detection. Also, it used NuSMV and PAT model as checkers to verify its behavioral models. The work in [21] established a new scheduling algorithm based on double-

fitness adaptive algorithm-job spanning time and load balancing genetic algorithm, and it adopted greedy algorithm to initialize the population.

In addition to its extensive application in cloud computing, heuristic scheduling algorithm is also applied to the computing system of multi-core processors. The work in [22] presented a meta-heuristic algorithm to solve static task scheduling for different processors in heterogeneous computing system. It introduced new operators to guarantee sample variety and improve its performance. The work in [23] provided an idea to reach an optimal makespan. It utilized resources at maximum by converging its algorithm to the optimal solution in the shortest time. The work in [22] also proposed a genetic-based algorithm to solve static task scheduling for processors in large-scale computing environment. It replaced the random initial population with some known initial populations, and obtained a better performance. Moreover, it lowered repetitions with relatively optimized methods.

In this paper, we use the idea of natural evolution and heuristic development of genetic algorithms to combine the characteristics of GPUs and tasks to find an approximately optimal task assignment scheme by maximizing cluster resource utilization.

### 2.1 CPU-GPU Collaborative Computing Cluster

Generally, a GPU contains at least hundreds of computing units (CUDA cores), and each of them can execute a thread in parallel. In CUDA, the CPU is regarded as a master and mainly focuses on complex logical transaction and serial computing; the GPU is the coprocessor and perform intensive parallel kernels which are assigned by its CPU, as shown in the left of Fig. 1. The two parts work together to ensure that the entire calculation is completed successfully. Each kernel launch plenty of threads to be executed in parallel, and all threads of a kernel execute the same code but with numerous data, that is Single Instruction Multiple Thread (SIMT) mode. However, since Kepler architecture, a thread supports two concurrent instructions simultaneously, that is Multiple Instruction Multiple Thread (MIMT) mode [24].

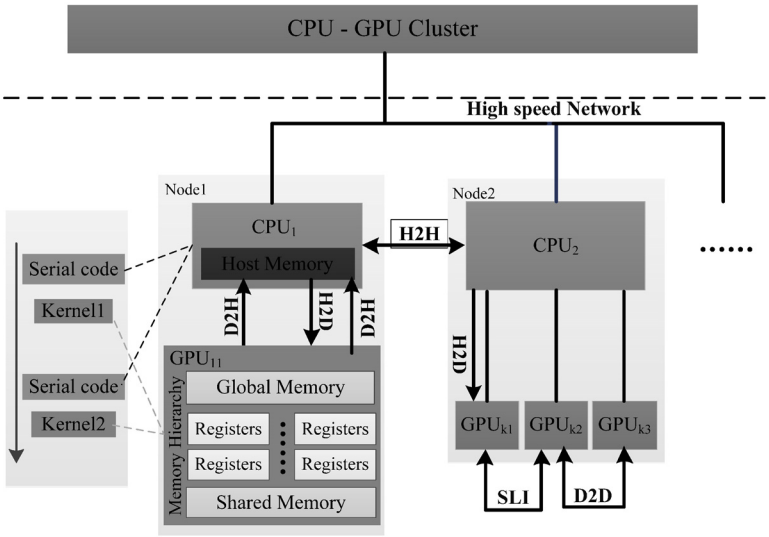


**Fig. 1.** Cluster model

In this paper, we regard a GPU cluster as our computing system for large-scale data processing. For a single node, the CPU and GPU (or GPUs) are physically connected through PCI-Express (PCIe) bus and logically connected through the Compute Unified Device Architecture (CUDA) programming model, as shown in Fig. 1, and different GPUs are connected by peer-to-peer mechanism without needing the PCIe [25]. So, the peer-to-peer technique can greatly reduce the communication overhead between different GPUs in the same node. For different nodes in the cluster, they are connected through high speed network.

In Fig. 1, the cluster has multiple nodes, and each node has different number of GPUs. Among them, the Node1 has only one GPU, and Node2 has three GPUs. For different nodes, the CPU and their GPU

(GPUs) work together via PCIes. In Node1, CPU1 and GPU11 transfer data to each other by calling the cudaMemcpy () API.

### 2.2 Dataflow Natural Attribute

In this paper, the dataflow that needs to be processed comes from the collection of sensors in the Internet of Things (IoT) and video steams that is monitored in real time [26], and there is a close dependence between these dataflow in different areas, such as time-dependent and space-dependent. The temporal and spatial dependency distributions are natural attributes of the dataflow [27], as shown in Table 1. Therefore, taking into account the coupling of dataflow, without changing the natural distribution of them, researchers reduced the GPU energy consumption through DVFS instead of migrating loads to other processors.

**Table 1.** Data dependency relationship

| Name | Instance | Type |
|---|---|---|
| Write after read | a=1; b=a; | Time dependency |
| Write after write | a=1; a=2; | Time dependency |
| Read after write | a=b; b=1; | Space dependency |

In the above three cases, as long as the reordering or migration of some instructions, it will affect the program execution results. Therefore, in order to guarantee the correctness of the program, we reduce the migration of subtasks and use DVFS to reduce cluster energy consumption and improve reliability. What's more, the data dependencies are limited to operations within a single processor or a single thread.

## 3 Basic Theoretical Models

In this section, we present the task-processor allocation model, energy consumption model of the collaborative computing cluster.

### 3.1 Task-processor Allocation Model

In a collaborative computing cluster, it has multiple nodes and each of them contains one CPU and at least one GPU. All of them are used to perform a large task collaboratively, and meanwhile they should follow certain allocation principles. Regardless of the CPU type, the main factor of affecting task assignment and execution is the GPU type. Table 2 lists the related parameters of GPUs used in our cluster.

**Table 2.** Basic parameters of different GPUs

| GPU | Global memory | Shared memory | Register files | CUDA cores | Compute capability | TDP |
|---|---|---|---|---|---|---|
| GTX 480 | 1.5 GB | 48 KB | 32 KB | 480 | 2.0 | 250W[1] |
| GTX 670 | 2 GB | 48 KB | 64KB | 1344 | 3.0 | 170W |
| GTX 680 | 2 GB | 48 KB | 64KB | 1536 | 3.0 | 195W |
| GTX 970 | 4 GB | 96KB | 64KB | 1664 | 5.2 | 177W |

Table 2 shows that different GPUs have different computing resources, memory resources and compute capability, and all of them will affect the task assignment. Therefore, the following relationships should be satisfied before determining the task assignment scheme in our cluster:

$$\begin{cases} \sum N_i M_{GPU_i} \geq M_{task} \\ N_{GPUs} \geq N_{sub-tasks} \end{cases}. \quad \textbf{(1)}$$

[1] It is the CUDA cores' TDP announced by NVIDIA, and the GPU's TDP is bigger than 250W.

In Eq.(1), $M_{GPU_i}$ denotes the memory size of $GPU_i$, and $i$ represents the type of GPU, such as 480, 680. $N_i$ denotes the number of $GPU_i$. The equation means that the memory resources of GPU cluster are more than the memory of the total task, and the number of GPUs is bigger than the number of sub-tasks.

### 3.2 Power Consumption Model

The work in [28] proposed an effective power and energy consumption measurement requirements for supercomputers, and some of them have been used by the Green500 and Top500. An empirical power model for the GPU was developed, and it calculated dynamic power by predicting execution times. Moreover, it also modelled the relationship between power consumption and GPU temperature [29]. The work in [30] statistically analyzed and modelled the GPU power by exploiting the characteristics of performance, workloads. None of these studies can directly obtain the real-time power consumption of a single node.

According to the original calculation definition of power consumption $P = UI$, we design and develop the GCPC model to directly obtain its real-time power. The power consumption of s single node can be expressed by Eq.(2), and the detail proof process is presented in [26],

$$P_{cn_i}(t_i) = U_{cn_i} \cdot I_{idl} + U_{cn_i} i_{cn_i}^{dny}(t_i), \tag{2}$$

and the cluster power consumption can be expressed by Eq.(3)

$$P_{cluster} = \sum_{i=0}^{k-1} P_{cn_i}. \tag{3}$$

### 3.3 Reliability Evaluation Model

System reliability represents the ability of a node to complete a defined function within specified conditions and time. If the node fails to complete the prescribed function, it is called a failure. There are multiple processors in our cluster, and their reliability varies with the processor type. In order to better measure the reliability of the cluster, we introduce the same reliability indictor – Mean Time Between Failures (MTBF) [31]. According to our previous work, the MTBF can be expressed by Equation (4).

$$MTBF = n \cdot \frac{1}{N_0} \cdot \sum_{i=1}^{N_0} t_i. \tag{4}$$

Where $n$ denotes the number of GPUs, and $N_0$ is the number of failure times. Working hours were $t_1, t_2, \cdots, t_{N_0}$ of each time, respectively. So we can calculate the MTBF of different nodes through a large number of experiments.

## 4 Algorithm Design and Implementation

In this section, we will first present the example and motivation of the propose RSNAP. Then, the optimal task-processor allocation scheme is determined by exploiting the natural evolutionary ideas of GA. Finally, through adjusting GPU frequency and voltage reduces power consumption. Meanwhile, for tasks without data dependencies use dynamic task migration to guarantee cluster performance.

### 4.1 Example and Motivation

In our practical application, no matter how many GPUs of a node are executing long-running task, the whole node may be down due to high power consumption. Here, take a node with three GTX970 GPUs as an example, the first two GPUs (GPU1 and GPU2) are connected by scalable link interface (SLI), and all of them support peer-to-peer transmission technique. So, Data transfer between GPUs does not need to go through the corresponding CPU. In this node, we do the following tests:

(1) GPU1 is full loaded and the other two are idle. After running for a long time (t1), the node is in normal working condition;

(2) GPU 3 is full loaded and the other two are idle. After running for a long time (t2), the node crashed;

(3) Both GPU1 and GPU2 are full loaded and GPU3 are idle. After running for a long time (t3), the node crashed;

(4) All the GPUs are full loaded, and the node crashed soon (t4).

**Table 3.** GPUs in different tests

| | Test 1 | | | Test 2 | | |
|---|---|---|---|---|---|---|
| | GPU1 | GPU2 | GPU3 | GPU1 | GPU2 | GPU3 |
| Load (%) | 51 | 49 | 0 | 0 | 0 | 99 |
| Power (W) | 106.2 | 131 | 20.71 | 19.116 | 65 | 173.23 |
| Temperature (°C) | 49 | 61 | 31 | 31 | 51 | 95 |
| | Test 3 | | | Test 4 | | |
| | GPU1 | GPU2 | GPU3 | GPU1 | GPU2 | GPU3 |
| Load (%) | 99 | 99 | 0 | 99 | 99 | 99 |
| Power (W) | 152.4 | 167.15 | 57.38 | 138.6 | 159.12 | 123.9 |
| Temperature (°C) | 84 | 94 | 49 | 72 | 94 | 56 |

The test application is OpenPose [32], and it has t1 > t2 > t3 > t4. Compare the first two tests, we found that although GPU1 is fully loaded in its original state, it assigns half of the task into GPU2 by SLI automatically. Therefore, the node is safe without any failure. But, all the other tests have instantaneous failures. That is because the higher utilization and computation intensiveness are prone to produce excessively high system temperature and power consumption, and finally result in processor failure. Therefore, we consider reducing processor load to reduce power consumption and improve reliability.

### 4.2 Original Allocation Approach

In this section, we will select the optimal task-processor allocation scheme with considering fitness function in GA.

During the selection phase, we initial population, that is, all allocation solution in the cluster. It randomly combines tasks and GPUs to get multiple allocation solutions. All of them can be regarded as the parents for selecting better offsprings. Then, the parents of a GA evolve largely by crossover and mutations. In our approach, it generate new offsprings by crossover operation with be limited by fitness function. The fitness function of the collaborative computing cluster is expressed by Eq.(5) ,

$$\begin{cases} C_{GPU_i} \geq T_{C_j}, M_{GPU_i} \geq T_{M_j} \\ Max\left(\sum T_{C_j} / \sum C_{GPU_i}\right) \\ Max\left(\sum T_{M_j} / \sum M_{GPU_i}\right) \end{cases} . \quad \textbf{(5)}$$

Where $T_{C_j}$ and $T_{M_j}$ represent the amount of computation and data size needed for different subtasks, respectively. $C_{GPU_i}$ and $M_{GPU_i}$ denote the compute resource and memory resource of $GPU_i$ in the cluster. Sometimes, a subtask even requires multiple processors. Under the constraint of Eq.(5), we search for the optimal solution with maximal resources utilization and minimum execution time with the guidance of GA.

### 4.3 Natural Attribute Priority Scheduling Approach

#### 4.3.1 Natural Attribute Priority Based on GA

The proposed RSNAP is inspired by GA [23], which is the most suitable heuristics algorithm that can be used to find the near optimal solution of hard problems, such as task scheduling problem, traveler

salesman problem [17]. In recognition of task scheduling, we simulated the biological evolution process to find the best task-processor allocation scheme.

Generally, the parent generation produces offspring often needs the following operations: selection, crossover and mutation. The better-performing individuals are singled out as parents in selection process, and they are the original task-processor allocation schemes in this paper. Then, in crossover process, the parents exchange information themselves to produce offspring, that is, new task-processor allocation schemes by task scheduling. By crossover, we can obtain better allocation results. Mutation is the process that only a single individual mutates into a new one. In addition, we also need fitness computing in this paper, and the fitness parameters are power consumption and temperature and so on.

Unlike other traditional search methods, GA has multiple search points simultaneously, and each of them corresponds to one task assignment determined in Section 4.2. Meanwhile, these assignments are represented by a sequence of symbols, and all the sequences are called chromosomes in GA. Among them, each chromosome has a fitness value, which is evaluated by an objective function (fitness function) value. In this paper, the fitness value is listed in Table 3. Using fitness function, we can obtain better chromosomes, that is, better task assignment in the cluster.

In summary, all allocation schemes are the original population of RSNAP, and we search for the optimal scheme based on the idea of GA and heuristic idea.

#### 4.3.2 Initial Population

In our cluster, the scheduling problem of resources and tasks is the mapping problem of tasks and processors, aiming at reducing the cluster power consumption while ensuring performance. In Section 4.2, we developed the original task-processor allocation scheme based on the task type and processor type in the cluster. All possible original schemes are the initial population of the algorithm, and each chromosome represents a task-processor allocation scheme.

#### 4.3.3 Optimal Allocation Based on Fitness Value

In this section, we combine fitness value of GA and heuristic idea to obtain the best task-processor assignment scheme of the cluster. In Section 4.2, we had the original task-processor allocation scheme with simple restricted conditions, and the scheme is the parent of GA. Then, the parents will produce adaptable offsprings by crossing, mutating, etc. In previous work, we analyzed the usage of computing and storage resources by different kernels for GPUs and defined the types of kernels: Compute-Bound (CB) type, Memory-Bound (MB) type and Balanced (B) type. At the same time, we also analyzed how to allocate different types of kernels to make better use of cluster resources when there are different levels of GPUs in the cluster. Among them, the CB type kernel requires higher shared memory resource; the B type kernel requires lower shared memory and register files resources; the MB type kernel requires higher shared memory and register files resources. Therefore, we can translate these rules into fitness values and combine the heuristic ideas to develop crossover and mutation rules to seek the optimal offsprings, that is, the optimal task-processor allocation scheme.

For optimal task-processor allocation scheme, we propose a Heuristic-based Task-Processor Scoring (HTPS) method as the fitness value to search an optimal mapping. The HTPS evaluates existing assignments and gives corresponding values; if the score is below the threshold, it is redistributed and graded according to relevant rules until the highest score in the entire cluster is found. Among them, the fitness value is calculated in Table 4.

**Table 4.** Grading rules

| Kernel Type | Compute capability | | |
|---|---|---|---|
| | High | Middle | Low |
| M | 1 | 0 | -1 |
| B | -1 | 0 | 1 |
| C | 1 | 1 | -1 |

We repeated the above processes until we found the highest fitness value, that is, the optimal task-processor allocation scheme.

#### 4.3.4 Power-based Task Scheduling

Having determining the optimal task-processor allocation scheme in Section 4.3.3, each sub-task is fixedly stored in the memory medium of the corresponding GPU according to its natural attribute. Without affecting the data distribution, we reduce the GPU's real-time energy consumption by adjusting its voltage and frequency in Linux OS. The work in [33] presented the relationship between dynamic power with frequency and voltage of CMOS circuit, and it can be expressed as:

$$P_{dyn}=ACV^2F=KV^2F\ . \tag{6}$$

Where $P_{dyn}$ denotes GPU's dynamic power consumption, $A$ and $C$ denotes switch activity factor and capacitance, respectively, and they are constants for a specified GPU. $K=AC$ is the constant coefficient. $V$ represents supply voltage and represents GPU's core frequency. Therefore, GPU power consumption changes in proportion to their changes.

For each GPU, we search for all possible frequency and corresponding voltage values. When adjusts its frequency, the voltage also changes. Then record the current to calculate real-time power consumption by Eq.(1). The detail process is described in Algorithm 1.

**Algorithm 1.** Reliable resource scheduling algorithm

**Input:** $(V,F)$, GPU power consumption

**Output:** new$(V,F)$, GPU power consumption

1. **for** a designated type $GPU_i$ in the cluster **do**
2. Search for all supported $(V,F)$ combination and record them to table $V_i$
3. **for** each $(V,F)$ combination **do**
4. Calculate $P_{dyn}$ by Eq.(1) and record $P_{dyn}$ and temperature $T$ to table $P_i$
5. **end for**
6. **if** $GPU_i$ has $P_{real}\approx P_{max}$ and $T_{real}\approx T_{max}$ in table $P_i$ then
7. Reduce $(V,F)$ until $P_{real}\approx 0.7P_{max}$ and $T_{real}\le 0.7T_{max}$
8. **if** $P_{real}$ or $T_{real}$ continue to decrease with the new $(V,F)$ then
9. Increase $(V,F)$ until $P_{real}\approx 0.9P_{max}$ and $T_{real}\approx 0.9T_{max}$
10. **else**
11. Keep the current $(V,F)$ level
12. **end if**
13. **end if**
14. Repeat step 6 to 13 when and return to maximum
15. **end for**
16. Return the new $(V,F)$

In Algorithm 1, the original voltage and frequency combination $(V,F)$ is the input, and each GPU is fully loaded. When the GPU's energy consumption and temperature rise to the threshold value, we adjust $F$ and $V$ to reduce real-time power consumption and temperature. Meanwhile, the fan speed remains unchanged. After GPU returns to its stable working state with lower power consumption, we can continue to increase them for improving GPU performance again. The above processes are shown in step 6 to step 13. Step 14 repeats these operations dynamically based on GPU status during task execution.

Using this Algorithm, although GPU may sacrifice some of its performance due to lower core frequencies, the reliability of the entire cluster has been improved. Further, global optimization is achieved by reducing local optimal.

### 4.4 Dynamic Taskflow Migration

As we reduce the energy consumption of equipment by reducing the frequency and voltage of GPU, the system performance will be affected to some extent. Therefore, we introduce a Dynamic Taskflow Migration Approach (DTMA) to reduce GPU power consumption without affecting its performance. In DTMA, we monitor the power consumption, temperature, performance, etc. of different GPUs in real time. When any indicator exceeds its threshold and there is no dependency between the dataflow, we migrate some of the subtasks to other available GPUs to ease the task pressure of the current device. Hence, taskflow migration is able to further guarantee system performance and reliability. During the task execution phase, there are multiple indicators to determine whether the sub-taskflow is migrated to another device, such as GPU temperature, energy consumption and data dependencies. Here, we briefly introduce our DTMA with performance as an example.

When a GPU performance changes, it will affect task execution efficiency. Therefore, in order to ensure real-time of the computing system, we monitor the GPUs' performance and dynamically adjust their workloads. As long as the performance of the current GPU is less than the predetermined threshold, we will carry DTMA. The prerequisite for task migration is that there are available computing and memory resources.

Based on GPU execution efficiency, we divide GPU's performance into three levels: P0, P1and P2. Among them, P0 is zero, P2 means the maximum performance, so P1is the migration threshold (P2 > P1 > P0). Then the new taskflow reallocation scheme (TRAS) can be described in Algorithm 2.

**Algorithm 2.** Dynamic task migration approach

**Input:** Original allocation
**Output:** new task-processor allocation

1. **for** each GPU in the cluster **do**
2. Check its performance $P_{real}$ and compare it with predetermined threshold $P_2$
3. **if** $P_{real} < P_2$ and without data dependencies
4. Migration 30% sub-taskflow to an available GPU in the cluster
5. **else**
6. Keep the original allocation approach
7. **end if**
8. Repeat step 2 to step 7
9. **end for**
10. Return new allocation approach

In this section, the sub-taskflow is migrated to other GPUs according to GPU performance changes. Through step 2 to step 7, we can ensure the real-time of task execution. Here, we briefly analyze the relationship between performance changes and task migration. In our next paper, we will elaborate on the relationship between device temperature, system energy consumption and dynamic task migration.

## 5 Experiment Evaluation

To demonstrate the effective of the proposed approach, we firstly present our experiment environment in this section, followed by the detail results analysis.

### 5.1 Experiment Environment

The computing cluster includes multiple CPU-GPU collaborative computing nodes, as shown in Table 5.

**Table 5.** CPU-GPU cluster configuration information

| Node | CPU | GPU type | GPU No. |
|---|---|---|---|
| 1 | A 4-core Inter i5-3470 CPU @3.20GHz 4G Mem 500G HD | GTX 480 | 1 |
| 2 | A 2-core Inter Duo CPU E8400 @3.00GHz 8G Mem 500G HD | GTX 670 | 1 |
| 3 | A 4-core Inter i5-760 CPU @2.80GHz 8G Mem 500G HD | GTX 680 | 2 |
| 4 | A 8-core Inter i7-4790 CPU @3.60GHz 32G Mem 1T HD | GTX 970 | 3 |

Different nodes are assigned to different tasks, and they are listed in Table 6. We fully considered the characteristics of tasks and GPUs when select the optimal task-processor allocation approach.

**Table 6.** Task-processor allocation approach

| Node | Task | Application | Benchmark |
|---|---|---|---|
| 1 | Gaussian | Basic Function | [34] |
| 2 | Matrixmul | Basic Function | [35] |
| 3 | K-means | Clustering | [31] |
| 4 | OpenPose | Deep Learning | [32] |

We modify the source code of these algorithms and increase the amount of data as much as possible. Among them, K-means is more complex than Gaussian and Matrixmul, and it is assigned to a node with two GPUs. OpenPose is the first real-time multi-person system to jointly detect human body, facial keypoints, and it requires powerful computing capability and memory space for multi-channel video processing. Therefore, it is executed by Node4.

### 5.2 Data Dependency Analysis

We analyze the data dependencies in Table 6 in terms of time and space, respectively, and the results are shown in Fig. 2.

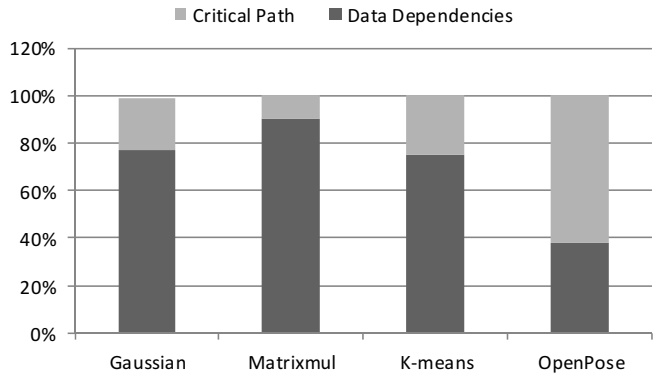


**Fig. 2.** Data dependencies analysis

In this figure, the critical path is the longest path that does not contain wait states or dependencies events, and the data dependencies in the red section of the figure represent the dependency ratios of different tasks. Among them, the OpenPose has the maximum, so it means that subtask migration can seriously undermine the integrity of the program. Meanwhile, the other three programs have different levels of data dependencies.

### 5.3 Execution Time Analysis

We compare the performance of different applications with and without RSNAP. The workload of the same application in both cases is unchanged, meanwhile, in Baseline test, we make each node runs 24 hours with full load. When the system temperature or energy consumption rises to 95% of the maximum, we back up the current calculation results in Baseline. In addition, nodes may crash due to high GPU power consumption. Under the circumstances, we will restart the node and re-execute from the backup point. The execution results are shown in Fig. 3.

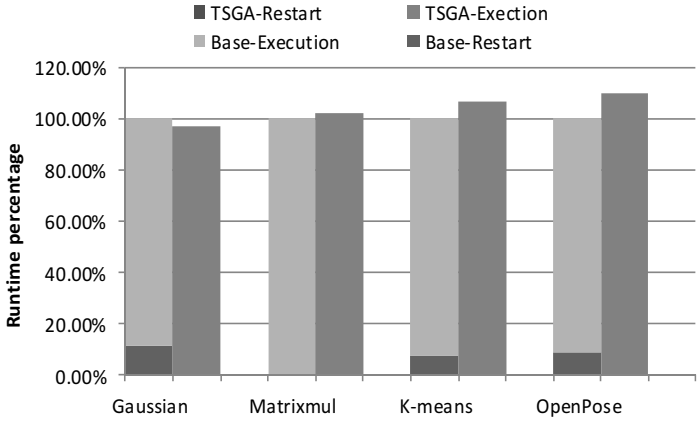


**Fig. 3.** Execution time comparison with and without RSNAP

In Fig. 3, the restart overhead includes three parts: node reboot time, GPU recovery time and application rollback to the last backup point time, as shown in Fig. 4.

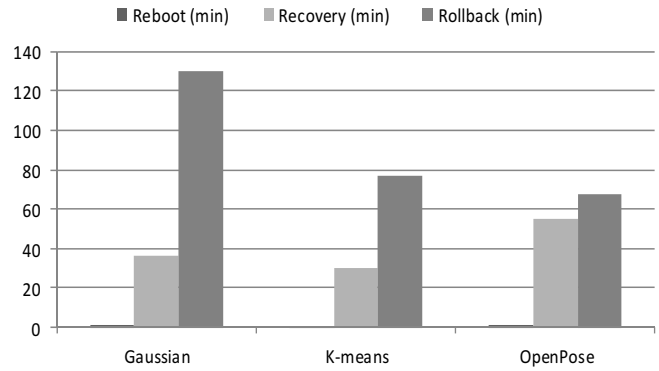


**Fig. 4.** Restart overhead distribution

Among them, the Gaussian and K-means execution node crashed one time, and OpenPose execution node crashed two times owing to high power consumption. The restart overhead of Gaussian is longest, because its rollback overhead more than 8% of total execution time. Although OpenPose node crashed two times, its rollback overhead is less than others.

From the two figures, we can see that although the proposed RSNAP reduces GPU's frequency and voltage, it does not always lead to an increase in execution time. For example, Gaussian's runtime in RSNAP is only 96.7% of the baseline. That is because there is a long time between backup point and node crash point, and it needs to be re-executed this part of the task after the node restarts, as shown in Fig. 3. As for Maxtrimul, it is executed in GTX670, and during its runtime, the node did not crash due to high power consumption. So, its Restart overhead is zero. For K-means and OpenPose, they have longer execution time in RSNAP than in Baseline.

### 5.4 Power Consumption Analysis

We also compared the maximal power consumption with and without RSNAP, as shown in Fig. 5.

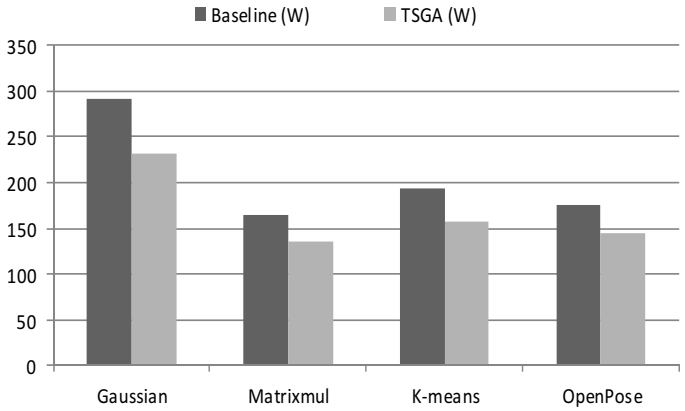


**Fig. 5.** Power analysis with and without RSNAP

In Fig. 5, we see that nodes power consumption drops significantly after using RSNAP. In Baseline, nodes execute applications at full load, and their power consumption keeps rising with the execution time. The higher the power consumption, the more likely the instantaneous failure is. Under the monitoring of our GCPC model, as long as the power consumption is greater than 90% of the maximum, we reduce the GPU frequency and voltage while keeping the fan speed constant. Finally, it not only reduces GPU power consumption but also avoids transient failures.

### 5.4 Reliability Analysis

The proposed RSNAP can effectively avoid the GPU transient failure caused by high power consumption and improve the system reliability. Table 7 shows failures information of them.

**Table 7.** Failures analysis

| Node | Gaussian | Matrixmul | K-means | OpenPose |
|---|---|---|---|---|
| Baseline | 1 | 0 | 1 | 2 |
| RSNAP | 0 | 0 | 0 | 0 |

## 6 Conclusions

In this paper, it proposes a resource scheduling approach with natural attribute priority based on genetic algorithm and improves the reliability of CPU-GPU collaborative computing clusters. Firstly, it follows the idea of GA and searches for an approximate optimal task-processor allocation scheme under the condition of fully considering the characteristics of GPUs and tasks. Then, for the problem of high power consumption causing node crashed in large-scale computing, we present a reliable scheduling approach by adjusting GPU frequency and voltage to reduce its workloads. Further, less workload will produce less power and avoid instantaneous failure. However, it will affect the execution performance owing to the reduced frequency and voltage. Therefore, for task without data dependencies, we use dynamic taskflow migration approach to guarantee system performance and reduce power consumption. Finally, the system reliability is improved.

In future work, we attempt to apply our approach to more fields and combine the fault-tolerant methods to obtain more reliable cluster system.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (Nos. 61572325 and 60970012); Ministry of Education Doctoral Fund of Ph.D. Supervisor of China (No. 20113120110008); Shanghai Key Science and Technology Project in Information Technology Field (Nos. 14511107902 and 16DZ1203603); Shanghai Leading Academic Discipline Project (No. XTKX2012); Shanghai Engineering Research Center Project (Nos. 630 GCZX14014 and C14001).

## References

[1] C.H. Nadungodage, Y. Xia, J.J. Lee, M. Lee, C.S. Park, GPU accelerated item-based collaborative filtering for big-data applications, in: Proc. IEEE International Conference on Big Data, 2013.

[2] D.A. Reed, J. Dongarra, Exascale computing and big data, Communications of the ACM 58(7)(2015) 56-68.

[3] W. Ai, K. Li, C. Chen, J. Peng, K. Li, DHCRF: a distributed conditional random field algorithm on a heterogeneous CPU-GPU cluster for big data, in: Proc. IEEE International Conference on Distributed Computing Systems, 2017.

[4] X.-W. Chen, X. Lin, Big data deep learning: challenges and perspectives, IEEE access 2(2014) 514-525.

[5] E. Strohmaier, J. Dongarra, H. Simon, M. Martin TOP500 LISTS. <https://www.top500.org/lists/top500/>, 2018 (18.03.15).

[6] DeepMind DISCOVER MORE ABOUT ALPHAGO. <https://deepmind.com/research/alphago/>, 2018 (18.04.06).

[7] S. Mittal, J.S. Vetter, A survey of methods for analyzing and improving GPU energy efficiency, ACM Computing Surveys (CSUR) 47(2)(2015) 19.

[8] S. Xiao, P. Balaji, J. Dinan, Q. Zhu, R. Thakur, S. Coghlan, H. Lin, G. Wen, J. Hong, W. Feng, Transparent accelerator migration in a virtualized GPU environment, in: Proc. the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), 2012.

[9] G. Wang, X. Ren, Power-efficient work distribution method for CPU-GPU heterogeneous system, in: Proc. Parallel and Distributed Processing with Applications (ISPA), 2010 International Symposium on, 2010.

[10] J. Coplin, M. Burtscher, Effects of source-code optimizations on GPU performance and energy consumption, in: Proc. the 8th Workshop on General Purpose Processing using GPUs, 2015.

[11] X. Mei, Q. Wang, X. Chu, A survey and measurement study of GPU DVFS on Energy conservation, Digital Communications & Networks 3(2)(2016) 89-100.

[12] A. Mishra, N. Khare, Analysis of DVFS techniques for improving the GPU energy efficiency, Open Journal of Energy Efficiency 4(4)(2015) 77-86.

[13] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, WH Free. Co., New York, NY, 1990.

[14] F. Pinel, B. Dorronsoro, P. Bouvry, Solving very large instances of the scheduling of independent tasks problem on the GPU, Journal of Parallel & Distributed Computing 73(1)(2013) 101-110.

[15] T. Hamano, T. Endo, S. Matsuoka, Power-aware dynamic task scheduling for heterogeneous accelerated clusters, in: Proc. IEEE International Symposium on Parallel & Distributed Processing, 2009.

[16] X. Tang, K. Li, G. Liao, An effective reliability-driven technique of allocating tasks on heterogeneous cluster systems, Cluster Computing 17(4)(2014) 1413-1425.

[17] F.A. Omara, M.M. Arafa, Genetic algorithms for task scheduling problem, J Parallel Distr Com 70(1)(2010) 13-22.

[18] Y. Xu, K. Li, J. Hu, K. Li, A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, Information Sciences 270(6)(2014) 255-287.

[19] B.F. Albdaiwi, H.M.F. Aboelfotoh, A GPU-based genetic algorithm for the p-median problem, Journal of Supercomputing 73(5)(2016) 1-24.

[20] B. Keshanchi, A. Souri, N.J. Navimipour, An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing, Journal of Systems & Software 124(February 2017)(2016) 1-21.

[21] T. Wang, Z. Liu, Y. Chen, Y. Xu, X. Dai, Load balancing task scheduling based on genetic algorithm in cloud computing, in: Proc. IEEE International Conference on Dependable, Autonomic and Secure Computing, 2014.

[22] M. Akbari, H. Rashidi, S.H. Alizadeh, An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems, Engineering Applications of Artificial Intelligence 61(2017) 35-46.

[23] S.G. Ahmad, C.S. Liew, E.U. Munir, F.A. Tan, S.U. Khan, A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems, Journal of Parallel & Distributed Computing 87(C)(2016) 80-90.

[24] N.G. GTX, 680: The fastest, most efficient GPU ever built, Whitepaper, NVIDIA 2012.

[25] S. Cook, CUDA Programming: A Developer's Guide to Parallel Computing with GPUs, Waltham, MA, 2013.

[26] Y. Fang, Q. Chen, N.N. Xiong, D. Zhao, J. Wang, RGCA: A Reliable GPU Cluster Architecture for Large-Scale Internet of Things Computing Based on Effective Performance-Energy Optimization, Sensors 17(8)(2017) 1799.

[27] J. Zhu, Z. Nie, J.R. Wen, B. Zhang, W.Y. Ma, Simultaneous record detection and attribute labeling in web data extraction, in: Proc. Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.

[28] T. Scogland, J. Azose, D. Rohr, S. Rivoire, N. Bates, D. Hackenberg, Node variability in large-scale power measurements: perspectives from the Green500, Top500 and EEHPCWG, in: Proc. the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015.

[29] S. Hong, H. Kim, An integrated GPU power and performance model, in: Proc. ACM SIGARCH Computer Architecture News, 2010.

[30] X. Ma, M. Dong, L. Zhong, Z. Deng, Statistical power consumption analysis and modeling for GPU-based computing, in: Proc. ACM SOSP Workshop on Power Aware Computing and Systems (HotPower), 2009.

[31] X. Wei, M. Hu, T. Peng, M. Jiang, Z. Wang, X. Qin, PRODA: improving parallel programs on GPUs through dependency analysis, Cluster Computing 22(1)(2019). DOI:10.1007/s10586-017-1295-4

[32] Z. Cao, T. Simon, S.E. Wei, Y. Sheikh, Realtime multi-person 2D pose estimation using part affinity fields. <https://arxiv.org/abs/1611.08050>, 2016.

[33] T.D. Burd, R.W. Brodersen, Energy efficient CMOS microprocessor design, in: Proc. Hawaii International Conference on System Sciences, 1995.

[34] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. Mclachlan, A. Ng, B. Liu, P.S. Yu, Top 10 algorithms in data mining, Knowledge & Information Systems 14(1)(2008) 1-37.

[35] C.W. Tsai, C.F. Lai, M.C. Chiang, L.T. Yang, Data mining for internet of things: a survey, IEEE Communications Surveys & Tutorials 16(1)(2014) 77-97.