Advances in Intelligent Systems and Computing 1143

Qi Liu · Xiaodong Liu · Lang Li ·
Huiyu Zhou · Hui-Huang Zhao *Editors*

Proceedings of the 9th International Conference on Computer Engineering and Networks

Springer

Advances in Intelligent Systems and Computing

Volume 1143

The series "Advances in Intelligent Systems and Computing" contains publications on theory, applications, and design methods of Intelligent Systems and Intelligent Computing. Virtually all disciplines such as engineering, natural sciences, computer and information science, ICT, economics, business, e-commerce, environment, healthcare, life science are covered. The list of topics spans all the areas of modern intelligent systems and computing such as: computational intelligence, soft computing including neural networks, fuzzy systems, evolutionary computing and the fusion of these paradigms, social intelligence, ambient intelligence, computational neuroscience, artificial life, virtual worlds and society, cognitive science and systems, Perception and Vision, DNA and immune based systems, self-organizing and adaptive systems, e-Learning and teaching, human-centered and human-centric computing, recommender systems, intelligent control, robotics and mechatronics including human-machine teaming, knowledge-based paradigms, learning paradigms, machine ethics, intelligent data analysis, knowledge management, intelligent agents, intelligent decision making and support, intelligent network security, trust management, interactive entertainment, Web intelligence and multimedia.

The publications within "Advances in Intelligent Systems and Computing" are primarily proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

**** Indexing: The books of this series are submitted to ISI Proceedings, EI-Compendex, DBLP, SCOPUS, Google Scholar and Springerlink ****

More information about this series at http://www.springer.com/series/11156

Qi Liu · Xiaodong Liu · Lang Li ·
Huiyu Zhou · Hui-Huang Zhao
Editors

Proceedings of the 9th International Conference on Computer Engineering and Networks

Springer

Editors
Qi Liu
School of Computer and Software
Nanjing University of Information
Science and Technology
Nanjing, Jiangsu, China

Lang Li
College of Computer Science
and Technology
Hengyang Normal University
Hengyang, Hunan, China

Hui-Huang Zhao
College of Computer Science
and Technology
Hengyang Normal University
Hengyang, Hunan, China

Xiaodong Liu
School of Computing
Edinburgh Napier University
Edinburgh, UK

Huiyu Zhou
Department of Informatics
University of Leicester
Leicester, UK

ISSN 2194-5357 ISSN 2194-5365 (electronic)
Advances in Intelligent Systems and Computing
ISBN 978-981-15-3752-3 ISBN 978-981-15-3753-0 (eBook)
https://doi.org/10.1007/978-981-15-3753-0

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

This conference proceedings is a collection of the papers accepted by the CENet2019—the 9th International Conference on Computer Engineering and Networks held on October 18–20, 2019, in Changsha, China.

This proceedings contains four parts: The first part focuses on System Detection and Application (25 papers); second part Machine Learning and Application (26 papers); third part Information Analysis and Application (27 papers); fourth part Communication Analysis and Application (39 papers).

Each part can be used as an excellent reference by industry practitioners, university faculties, research fellows, and undergraduates as well as graduate students who need to build a knowledge base of the most current advances and state-of-practice in the topics covered by this conference proceedings. This will enable them to produce, maintain, and manage systems with high levels of trustworthiness and complexity.

Thanks to the authors for their hard work and dedication as well as the reviewers for ensuring the selection of only the highest-quality papers; their efforts made the proceedings possible.

Nanjing, China — Qi Liu
Edinburgh, UK — Xiaodong Liu
Hengyang, China — Lang Li
Leicester, UK — Huiyu Zhou
Hengyang, China — Hui-Huang Zhao

Contents

System Detection

Design of Collaboration Engine for Large-Scale Heterogeneous Clusters

Hui Zhao and Haifeng Wang

Abstract Aiming at the low utilization rate of intensive computing cores in large heterogeneous clusters and the high complexity of collaborative computing between GPU and multi-core CPUs, this paper aims to improve resource utilization and reduce programming complexity in heterogeneous clusters. A new heterogeneous cluster cooperative computing model and engine design scheme are proposed. The complexity of communication between nodes and cooperative mechanism within nodes is analyzed. Coarse-grained cooperative execution plan is represented by template technology, and fine-grained cooperative computing plan is created by finite automata. The experimental results validate the effectiveness of the collaborative engine. Comparing with the manual programming scheme, it is found that the computational performance loss is less than 4.2%. The collaborative computing engine can effectively improve the resource utilization of large-scale heterogeneous clusters and the programming efficiency of ordinary developers.

Keywords Collaborative computing model · Finite automata · Computing engine · Template technology

1 Introduction

With the development of large data applications, data-intensive computing has become an important area of high-performance computing. GPU general computing is very suitable for data-intensive tasks. Heterogeneous cluster built by GPU and multi-core CPU has become a major data processing solution [1]. It can effectively improve the computing performance and efficiency and has a wide application

H. Zhao · H. Wang
College of Information Science and Engineering, Linyi University, Shuangling Road, Linyi 276002, China

H. Wang (✉)
Shandong Key Laboratory of Network, Research Institute of Linyi University, Shuangling Road, Linyi 276002, China
e-mail: gadfly7@126.com

Q. Liu et al. (eds.), *Proceedings of the 9th International Conference on Computer Engineering and Networks*, Advances in Intelligent Systems and Computing 1143, https://doi.org/10.1007/978-981-15-3753-0_1

prospect [2, 3]. In 2010, "Tianhe 1" supercomputer took the lead in adopting the heterogeneous system of GPU + CPU, using 7168 NVIDIA GPUs. And in the top ten supercomputers in the world, two systems adopt the structure of GPU + CPU. Industry has also begun to use GPU heterogeneous clusters to process large data. Cloud computing service Amazon provides users with GPU computing examples in EC2 cluster to accelerate the performance of oil exploration, meteorological analysis, image processing and other applications. In heterogeneous cluster of GPU, the cooperative computing behavior of GPU and multi-core CPU not only increases the density of computing resources within a single node, but also improves the utilization and efficiency of computing resources.

At present, in the research of GPU and CPU collaborative computing, a large number of data-intensive algorithms have been transplanted to GPU to improve the performance. For example, to solve the problem of subset summation, GPU and CPU multi-core cooperative algorithm is proposed to solve the load balancing between two types of processors and the optimization of communication between nodes and nodes [4]; to query protein sequence data, GPU and multi-core CPU are used to reduce the operation time, mainly to implement two widely used sequence alignment algorithms, SW and NW, and to study the load balancing strategy between cluster nodes. Fine-grained collaborative operation process in nodes [5]; 3-D terrain reconstruction task in UAV perception system belongs to data-intensive real-time computing. Song et al. explored data access operation in GPU-CPU collaborative computation of 3-D reconstruction algorithm and achieved good acceleration effect [6]; proposed CPU/GPU collaborative algorithm for parallel difference algorithm and applied it to mass LIDAR point cloud generation digital. In the elevation model, researchers designed the first-in-first-out queue with concurrent access shared by CPU and GPU to manage tasks and proposed a collaborative framework of parallel collaborative difference [7]; proposed a collaborative parallel computing algorithm between CPU and GPU for local search in optimization algorithm; designed a HySyS prototype system to solve a series of combinatorial optimization problems [8]; Moim implements a collaborative computing system based on MapReduce mode. To decompose the three computing stages of MapReduce, it realizes task allocation and load balancing between GPU and CPU multi-core.

However, there is a lack of research on the commonality of collaborative computing. In collaborative computing programming of GPU and multi-core CPU for different domains, users with different application backgrounds need to write GPU and CPU collaborative computing programs according to business logic. They should not only pay attention to the algorithm logic in their respective domains, but also to the collaborative computing logic. Therefore, the development of GPU collaborative computing program is very difficult, which requires users to have a thorough understanding of thread synchronization, data transmission between GPU and CPU and communication between network nodes. In order to reduce the programming difficulty of GPU collaborative computing, a complete design scheme of collaborative computing engine is proposed in this paper. Firstly, the common problems of collaborative processes in different computing jobs are extracted, and then, the

collaborative logic processes in computing jobs are extracted to reduce the programming complexity of users by reducing repetitive programming. The organizational structure of this paper is as follows: Sect. 2 introduces the main problems to be solved by the collaborative computing engine; Sect. 3 provides the specific scheme for the design of the collaborative computing engine; Sect. 4 summarizes and looks forward to future work.

2 Problem Description

Firstly, the main problems of collaborative computing between GPU and multi-core CPU are analyzed from the perspective of computing system. When GPU heterogeneous cluster processes large data jobs in a distributed way, the input data is divided into several data sets according to load balancing scheduling strategy. Each computing node is responsible for processing the corresponding data, and finally, the final results are summarized. In a single computing node, the data is divided into two parts, which are accomplished by GPU and multi-core CPU, respectively. Therefore, collaborative computing is divided into two parts: intra-node collaboration and inter-node collaboration. The input data of large data processing jobs is loaded into CPU memory by network or external memory. There are three areas in CPU memory: CPU memory area, buffer area and memory area exchanged with GPU. Among them, CPU memory area provides data to CPU computing core; memory area exchanged with GPU is the area where CPU control core transmits data to GPU; computing data is transferred to GPU display by exchange memory, and then computed by GPU and returned to CPU switching memory area; the buffer is paging memory, which is used to aggregate GPU and CPU calculation results in nodes [3]. It can be seen that the cooperative computing in a single node includes inter-thread synchronization, lock operation and complex data communication operation. On the other hand, the process of collaborative computing among nodes is as follows: each node process is responsible for the calculation and collaborative control of the node, and then keeps synchronization and data communication with other nodes in the network. Finally, the global collaborative computing results are summarized. Collaboration among nodes includes process synchronization and complex network communication operations.

Secondly, the problem of collaborative computing is analyzed from the perspective of computing operations. User jobs are logically divided into two parts: domain algorithm logic and collaborative computing logic. The concrete realization forms of domain algorithm logic are GPU kernel function, CPU thread function and CPU data merging process. The design goal of the collaborative computing engine is to extract the collaborative computing logic, so that ordinary users only pay attention to the design of CUDA kernel function, thread function and merging process. After judging the characteristics of large data computing jobs, the cooperative execution plan between CPU computing core and GPU is generated and distributed to heterogeneous cluster nodes for execution. Collaborative execution planning is a coarse-grained activity sequence, which guides the calculation of GPU and multi-core CPU.

3 Cooperative Computing Engine

This section introduces the specific design scheme of the collaborative computing engine and the formal representation of the performance of collaborative computing.

3.1 Design of Cooperative Computing Engine

The architecture of the collaborative computing engine is shown in Fig. 1. The collaborative computing engine is divided into two levels. Coarse-grained collaborative planning is used between computing nodes, while fine-grained collaborative planning is used within computing nodes. Jobs are extracted from job queues, and their types are judged. Cooperative planning templates are imported into template libraries according to job types to achieve basic task allocation and computing data deployment in a distributed network environment, and synchronization and concurrent operations between nodes are specified. On the other hand, fine-grained finite state machine (FSM) is used to generate cooperative execution plan. When a specific event occurs, the state changes are triggered and the corresponding actions or operations are executed to achieve more detailed cooperative computing function within the node.

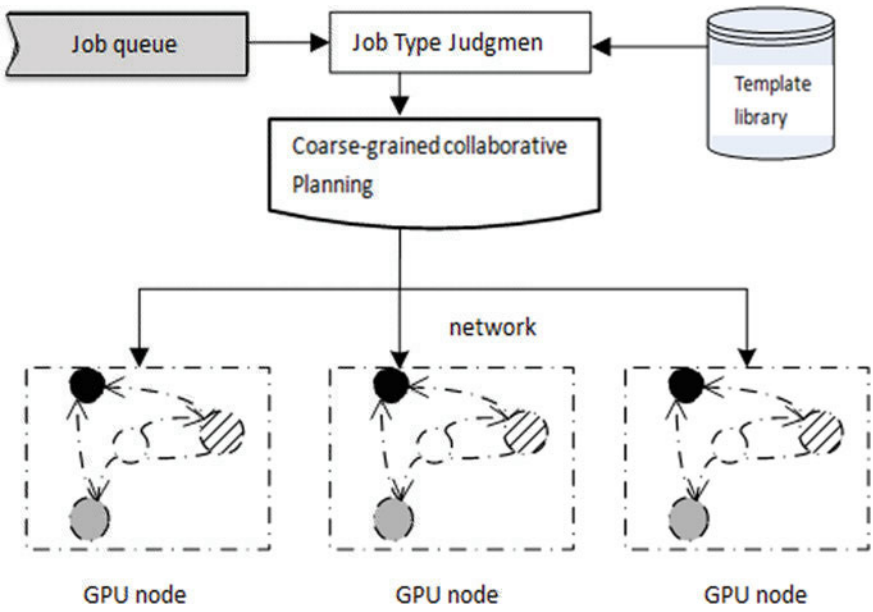

Fig. 1 GPU structure diagram of cooperative computing engine

The design of coarse-grained and fine-grained mixing has the following advantages: The template technology is used to realize the generation function of coarse-grained collaborative planning. Firstly, job types are classified simply, and corresponding collaborative computing models are provided according to different job types. This technology reduces the complexity of collaborative engine and improves the feasibility of implementation.

3.2 Template Cooperative Execution Plan

Template technology is a knowledge expression method based on reuse technology and similarity principle of things. The basic idea is to abstract a framework template from a kind of similar things. Collaborative computing engine extracts common processes from collaborative processes among computing nodes and defines common processes as templates for reuse. Job collaborative computing flow for large data processing is shown in Fig. 2.

Figure 2 shows the cooperative mode of MapReduce batch processing mode. Firstly, the reasonable deployment of input data is realized according to the load balancing strategy, and the data needed for calculation is obtained through the network transmission of each computing node [9]. Then, each computing node asynchronously completes its own computing tasks. The computing process of each node is synchronized [10]. The intermediate results are mixed and synchronized, and finally merged computing is realized. In this collaborative process framework, the differences among different computing jobs are data allocation strategy, data shuffling strategy and computing load balancing.

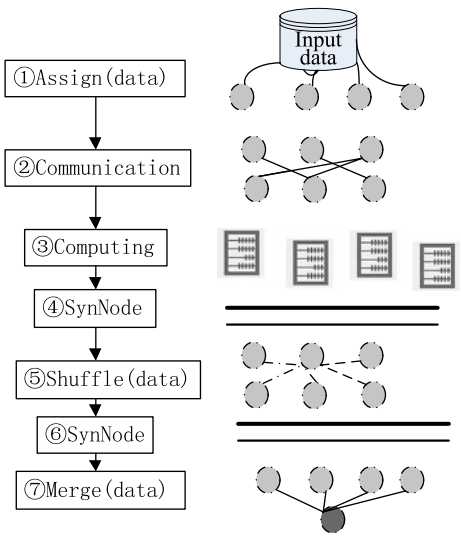

Fig. 2 Inter-node collaborative process framework

3.3 State Machine Execution Plan

Finite state machine is used to generate fine-grained collaboration processes in real time. Finite state machine (FSM) is a mathematical model that represents finite states and the migration and action between states [11]. The CPU, GPU, memory, buffer and GPU display are regarded as different data storage and computing objects. The finite state machine (FSM) models the behavior of each object, describes the state sequence of the object in each computing stage, and how to respond to various external events.

FSM can be expressed as $M = (Q, q_0, \sum, \delta, F)$, where Q is a non-empty finite set of states, $q \in Q$, *A state in which q is M*; q_0 is the start or start state of M; $\sum$ represents the input set of the system; δ is a state transition function; *F is the set of termination states of M*, if $q \in F$, q is the termination state of M. The cooperative operation process in GPU computing nodes is represented by FSM; Table 1 provides the basic state set and system input set.

After modeling the collaborative computing process in nodes with finite state machine, the mapping between collaborative process and finite state machine is needed. Collaborative computing process is composed of several specific operation steps, which have some logical steps, including shunting, aggregation, parallel, jump. In finite state machines, all data-centric states, such as full memory area and empty GPU display memory area, are the relationship between producers and consumers. Therefore, the mapping idea is that every step of collaborative computing is regarded as a state, when one state migrates to another state under certain conditions, it is also considered as a step; when a state has multiple inputs, there must be aggregation in the process of collaboration, from different states to the same state. For example,

Table 1 Symbol table of finite state machine system in node

System state		System input	
Symbol	Meaning	Symbol	Meaning
S_0	Ready state	x_1	Start threads or receive synchronization signals
S_1	GPU computing end	x_2	Terminate thread or task queue empty
S_2	GPU computing end	x_3	Terminate thread or task queue empty
S_3	GPU full display area	x_4	Transfer data to equipment
S_4	CPU full memory area	x_5	Loading data from external memory
S_5	GPU display memory space	x_6	Start GPU computing
S_6	Cache full	x_7	Memcpy data replication operation
S_7	Cache space	x_8	Memcpy data replication operation
S_8	Calculation failure	x_9	Abnormal, fault events
S_9	Calculation successful	x_{10}	Calculate startup events
S_{10}	Mission queue full	x_{11}	Task entry operation
S_{11}	Mission queue empty	x_{12}	Merge computing end

when a thread receives synchronization signals and the thread computation ends, the thread enters the ready state; when a state has multiple outputs, there must be shunting and concurrency in the collaborative process, which can be migrated from the current state to different states. From a technical point of view, a class is designed according to the finite state machine to generate fine-grained cooperative execution plan in real time, but the judgment of state migration will cause some performance loss.

4 Experiments and Analysis

In order to verify the effectiveness of the function of the collaborative computing engine and the performance loss of the algorithm for generating the collaborative execution plan, a typical large data computing job was selected to carry out the experiment. GPU heterogeneous cluster with four computing nodes in the Laboratory Bed, Network is Gigabit Ethernet, GPU Computing Node is configured as GPU Tesla K40 with global memory 12 GB. CPU is Intel Xeon E5-2660 8 core, 2.2 GHz, 32 GB Memory and Hard disk 2T. The operating system is Ubuntu 14.04, CUDA 7.5. Selecting large-scale microblog analysis jobs and Sort jobs in Hibench [12] as collaborative computing jobs in heterogeneous clusters of GPU, microblog data analysis operation can flexibly control data size and belongs to typical data-intensive operation, which is very suitable for GPU computing; Sort job is an I/O-intensive large data processing job, whose input data, intermediate data and output data are the same size. In the experiment, the judgment part of computing job type is simplified, and the corresponding template is used directly for two kinds of jobs, while collaborative computing within nodes is the execution plan generated by engine. (Collaborative engine abbreviated as CE). The benchmark procedure for experimental comparison is written by more skilled personnel (short for MC).

The actual computing time of microblog data analysis is relatively short. Table 2 shows that with the increase of microblog data size, the performance loss caused by the computational complexity of finite state machines in nodes decreases gradually. For such flow computing jobs with large input data and small output data, the performance loss can be controlled within 3%. The data scale of Sort job is large and the actual calculation time is relatively long. There are many types of cooperative state switching within nodes, so the performance loss of cooperative computing engine is between 3.1 and 4.2%. As can be seen from Table 3, the performance loss

Table 2 Performance comparison table of microblog data analysis

Data size (GB)	4	6	12	20
MC (s)	1.288	2.096	3.040	3.984
CE (s)	1.325	2.148	3.109	4.067
Performance loss (%)	2.9	2.5	2.3	2.1

Table 3 SortPerformance comparison table

Data size (GB)	40	63	85	105	126	150
MC (s)	41.21	55.22	102.36	126.52	156.31	190.28
CE (s)	42.47	57.15	106.04	131.70	162.87	198.27
Performance loss (%)	3.1	3.5	3.6	4.1	4.2	4.2

increases with the increase of data size in Port. Through performance tracking, it is found that because Sort belongs to I/O-intensive large data computing operations [13], the memory data area, GPU display data area and memory buffer area in the node will appear the phenomenon of local data waiting for synchronization lock, so the performance loss is larger than that of microblog data analysis.

Typical large data computing jobs verify that the collaborative computing engine can effectively generate the collaborative execution plan. The results of microblog data analysis and Sort are consistent with the result of experts. The performance loss caused by co-generation logic can be controlled to the acceptable range of ordinary users.

5 Summary and Prospect

In heterogeneous GPU clusters, multi-core CPU is a computational resource that is easily overlooked. CPU/GPU cooperative computing mode increases the computing resource density within a single node and can effectively improve the performance of parallel computing. Cooperative computing engine generates corresponding collaborative execution process for user jobs, which reduces the difficulty of user writing collaborative computing logic. The function of collaborative computing engine is validated by data-intensive flow computing tasks and I/O-intensive Sort tasks. The performance loss of collaborative computing is also controlled within acceptable range. However, when a finite state machine generates fine-grained cooperative operations within a node, the problem of multiple threads competing for lock resources is obvious in I/O-intensive tasks, resulting in greater performance loss than common tasks. In order to solve this problem, probabilistic finite state machine (PFSM) is introduced into the design of cooperative generation within nodes. Empirical probability is used to solve the problem of multi-state transition conflict.

Acknowledgements This work was supported in part by the Shandong Province Key Research and Development Program of China (No. 2018GGX101005), the Shandong Province Natural Science Foundation, China (No. ZR2017MF050).

References

1. Zhang, F., Zhai, J.D., He, B.S., et al.: Understanding co-running behavior on integrated CPU/GPU architectures. IEEE Trans. Parallel Distrib. Syst. **28**(3), 905–918 (2017)
2. Zhang, H., Zhang, L.B., Wu, Y.J.: Large-scale graph data processing based on multi-GPU platform. J. Comput. Res. Dev. **55**(2), 273–288 (2018)
3. Li, T., Dong, Q.K., et al.: Research on parallel computing mode of GPU task based on thread pool. Chin. J. Comput. **41**(10), 2175–2192 (2018)
4. Wan, L.J., Li, K.L., Li, K.Q.: A novel cooperateive accelerated parallel two-list algorithm for solving the subset-sum problem on a hybrid CPU-GPU cluster. J. Parallel Distrib. Comput. **97**, 112–123 (2016)
5. Zhou, W., Cai, Z.X., et al.: A multi-GPU protein database search model with hybrid alignment manner on distributed GPU clusters. Concurr. Comput. **30**(8), 1–13 (2018)
6. Song, W., Zou, S.H., et al.: A CPU-GPU hybrid system of environment perception and 3D terrain reconstruction for unmanned ground vehicle. J. Inf. Process. Syst. **14**(6), 1445–1456 (2018)
7. Wang, H.Y., Guan, X.F., Wu, H.Y.: A cooperative parallel spatial interpolation algorithm for CPU/GPU heterogeneous environment. Geomat. Inf. Sci. Wuhan Univ. **42**(12), 1688–1695 (2017)
8. Vidal, P., Alba, E., Luna, F.: Solving optimization problems using a hybrid systolic search on GPU plus CPU. Soft. Comput. **21**, 3227–3245 (2017)
9. Mengjun, X., Kyoung-Don, K., Can, B.: Moim: a multi-GPU mapreduce framework. In: 16th International Conference on CSE, 1279–1286 (2013)
10. Guo, M.S., Zhang, Y., Liu, T.: Research advances and prospect of recognizing textual entailment and knowledge acquisition. Chin. J. Comput. **40**(4), 889–909 (2017)
11. Shan, J.H., Zhang, L., et al.: Extending timed abstract state machines for real-time embedded software. Acta Sci. Nat. Univ. Pekin. (2019). https://doi.org/10.13209/j.0479-8023.2019.005
12. Huang, S., Huang, J., et al.: The Hibench benchmark suite: characterization of the mapreduce-based data analysis. In: IEEE International Conference on Data Engineering Workshops vol. 74, pp. 41–51 (2010)
13. Osama, A.A., Muhammad, J.I., et al.: Analyzing power and energy efficiency of bitonic mergesort based on performance evaluation. IEEE Access **6**, 42757–42774 (2018)

Research on Multi-stage GPU Collaborative Model

Longxiang Zhang and Haifeng Wang

Abstract GPU heterogeneous cluster is extensively utilized in the field of data analysis and processing. Nevertheless, research and studies on collaborative activity model in computing elements of GPU heterogeneous clusters are still inadequate. To conduct research on GPU and multi-core CPU cooperative computing from a theoretical perspective, a multi-stage cooperative computing model (p-DCOT) is established. Bulk synchronous parallel (BSP) model is the core of p-DCOT. Cooperative computing is divided into three layers: data layer, computing layer and communication layer. Computing and communication are described and formalized by matrix. Lastly, representative computing examines the effectiveness of model and parameter analysis. The collaborative computing model finds out the collaborative computing in big data analysis and processing.

Keywords Collaborative model · Load balancing · Big data computing · GPU cluster

1 Introduction

Large-scale data analysis and processing changes the development of high-performance computing. The emergence of high-performance computing platforms has changed from computing platforms to professional computing systems, which processes the big data in different industries [1]. The thread mechanism of GPU is capable of processing data-intensive calculation. Heterogeneous cluster of GPU and multi-core CPU create new opportunities for big data analysis and processing, which enhance calculation performance and improve efficiency [2]. An increasing number of research and studies on the role of GPU heterogeneous cluster in big data analysis and processing are carried out. Due to dynamic data input and random variation of calculation loading, calculation pattern, optimal operation, load balancing, reliability

L. Zhang · H. Wang (✉)
School of Information Science& Engineering, Linyi Univesity, Shuangling Road, Linyi Shandong 276002, China
e-mail: gadfly7@126.com

Q. Liu et al. (eds.), *Proceedings of the 9th International Conference on Computer Engineering and Networks*, Advances in Intelligent Systems and Computing 1143, https://doi.org/10.1007/978-981-15-3753-0_2

and expansibility of GPU have been negatively affected[3, 4]. For this reason, the systems encounter various limitations. Related theories on cooperative computing of big data analysis and processing are inadequate [5–7]. In this paper, theoretical models analyze the general calculation rules and extensive forms of MapReduce, thereby conducting optimization studies. The studies focus on cooperative computation models of different computation patterns and establish a new theoretical model at multiple stages and levels. Moreover, as the theoretical tool of discussing GPU and multi-core cooperative computing, primary model parameters are analyzed and validity of models is verified.

2 Problem Description

Assume GPU heterogeneous clusters include n nodes, and every node also involves CPU and GPU. If multi-core CPU has m cores, cybernetics cores are in interactive operation with GPU [8]. The remaining $m - 1$ are computing cores. Computing cores are divided into cooperative computing core (CCC) and standalone computing core (SCC). When it comes to big data processing and computing, cooperative computing cores and GPU divide data set, and complete cooperative and concurrent computing. Standalone computing cores are in charge of reduce or data aggregation within nodes. From this perspective, in GPU heterogeneous computing system, cooperative computing occurs among and within nodes. Therefore, computing and communicating process become more complicated.

3 Cooperative Computing Model

3.1 Overview of Cooperative Computing Model

As is shown in Fig. 1, a big data computing task is completed in p stages. Each stage is divided into data layer, computing layer and communication layer. On data layer, cybernetics cores of CPU complete distributed mapping. On computer layer, cybernetics cores of CPU, computing cores and GPU conduct concurrent operation. On communication layer, data aggregation occurs among distributed nodes. Intermediate results of computing layer are transmitted and aggregated, in order to prepare data for the next stage of computing.

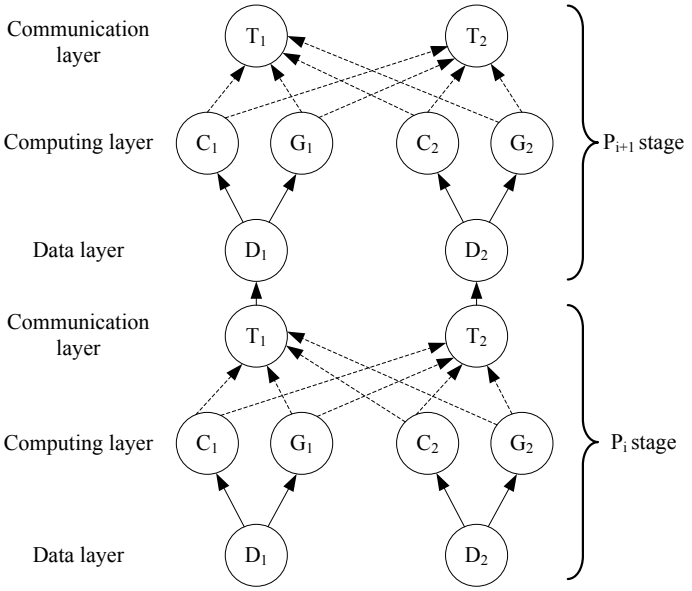

Fig. 1 Cooperative computing process of heterogeneous system

3.2 Formalized Description

Cooperative calculation model is described by matrix. Formalized expression of p-DCOT model is shown as follows:

$$\text{p-DCOT} = D((C+O)T)^p = D(C_1+O_1)T_1 \ldots D(C_p+O_p)T_p \tag{1}$$

Equation (1) shows the p phase of iteration. The expansion equation of each computing phase is shown as follows. Formalization of the i stage:

$$\begin{aligned} D(C_i+O_i)T_i = {} & \beta\left[\, D_1 \; D_2 \; \ldots \; D_n \,\right] \\ & \times \left(\delta_i W_g \begin{vmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & c_{ni} \end{vmatrix} + (1-\delta_i) W_c \begin{vmatrix} o_1 & 0 & 0 & 0 \\ 0 & o_2 & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & o_{ni} \end{vmatrix} \right) \\ & \times \begin{bmatrix} t_{1,1} & t_{1,2} & \ldots & t_{1,ni+1} \\ t_{2,1} & t_{2,2} & \ldots & t_{2,ni+1} \\ \ldots & \ldots & \ldots & \ldots \\ t_{ni,1} & t_{ni,2} & \ldots & t_{ni,ni+1} \end{bmatrix} \end{aligned} \tag{2}$$

Equation (2) demonstrates: cooperation of n_i computing nodes in the i phase of calculation. The diagonal elements of diagonal matrix C_1 to C_{ni} indicate the concurrence computing of GPU nodes. The diagonal elements of diagonal matrix O_1 to O_{ni} represent the concurrence computing of GPU cooperative computing cores. Matrix addition shows the convergence of GPU computing and CPU multi-core computing. Parameters δ_i represent loading and allocation factors of GPU and CPU in the nodes. This parameter determines the data size for GPU and CPU to process in each node. Parameter β_i is the loading and allocation factor that maps computing tasks to different machines. Weight matrix W_g and W_c represent the parameter matrix of GPU and CPU performance differences. Average matrix T represents the point-to-point communication status among the nodes. Element t_{ij} represents the telecommunication relationship between node n_i of the i phase and node n_j of $i + 1$.

3.3 Parameter Analysis

1. **GPU and CPU loading ratio δ in single computing node**

Assume that L is the data loading for single node. In single nodes, there are R cooperative computing cores and GPU. Peak performance of each cooperative computing core is P_{cpu} (assume they are isomorphic computing models. Cooperative computing cores have the same peak performances). Peak performance of GPU is P_{GPU}. Obtain the loading ratio of GPU and CPU δ in the same locking range. Equation (3) shows the completion time of whole computational nodes:

$$\begin{cases} T_{\text{total}} = \max(T_{\text{gpu}}, T_{\text{cpu}}) \\ T_{\text{cpu}} = \frac{L \times \delta}{R \times P_{\text{cpu}}} \\ T_{\text{gpu}} = \frac{L(1-\delta)}{P_{\text{gpu}}} \end{cases} \tag{3}$$

In order to achieve the minimum completion time T_{total} for the whole computing nodes, cooperative computing cores and GPU computing time should be equal to each other, which is shown in Eq. (4):

$$\frac{L \times \delta}{R \times P_{\text{cpu}}} = \frac{L(1-\delta)}{P_{\text{gpu}}} \tag{4}$$

Equation (4) determines the loading ratio δ of GPU and CPU

$$\delta = \frac{R}{P_{\text{gpu}}/P_{\text{cpu}} + R} \tag{5}$$

In practical application, accelerate cooperative computing cores of GPU and CPU to replace peak performance ratio, i.e., $s = P_{\text{gpu}}/P_{\text{cpu}}$. Next, Eq. (5) is derived from approximation of Eq. (6).