International Journal of Advanced Science and Technology
Vol.71 (2014), pp.49-58
http://dx.doi.org/10.14257/ijast.2014.71.05

The Research of Synthesizing Parallel Computing Models with Graph Reduction

Shen Chao [1,2] and Tong Weiqin[1]

[1]*School of Computer Engineering and Science Shanghai University, Shanghai, China*
[2]*Institute of Smart City (Sino-France) Shanghai University, Shanghai, China*
shenchao1010@aliyun.com

Abstract

The demands of data analysis and processing make the parallel computing platforms are continuously developed. But the existing parallel computing models which are the core of platforms present the characteristics of diversification, high pertinence and short cycle. So a synthetic model of supporting flexible platforms urgently needs to be researched. This work researches a synthetic model to shield the heterogeneity of parallel computing models under the theories of λ -calculus, functional language and graph reduction. Based on the MapReduce and BSP models, first of all, the performing principles of models are analyzed. And then the unified modalities of models are found. Finally, the synthetic model having high performance is developed with graph reduction rules, and the experiment results are also shown.

Keywords: *parallel computing, synthetic model, graph reduction, MapReduce, BSP*

1. Introduction

With the rapid development of information era, the amount of data in today's world is constantly increasing at a dramatic rate, and this poses a great challenge to data analysis and processing. Based on Internet, the web search dataset today is much larger than could have probably been imagined ten or even five years ago. Other examples of such large datasets abound, from medical treatment to financial data [1-2]. In order to grasp the development trend of social by analyzing and processing the data with characteristics of Volume, Variety, Value, Velocity and Integrality, data mining has been changed from routine to in-depth. How to make data to produce greater value has become an urgent problem to solve.

The parallel computing now plays a prominent role in the field of data analysis and processing. And many of parallel computing platforms have been developed, but each platform is only appropriate for a particular data type. Such as Hadoop, Storm, Pregel, Hama respectively points to data-intensive data in the form of <key; value> pairs, streaming data, large graph data and scientific computing data. Looking horizontally, with the amount of data type increasing the new computing platforms are constantly appearing, and looking lengthways, parallel computing platforms are constantly evolving.

And further on, efficient parallel computing models are fundamental importance in parallel computing platform field. For example, the platforms of HadoopDB, Pig Latin, Sawzall, Oivos, KPNs, Storm, Spark are all evolved from parallel computing model of MapReduce, and Prege, Hamma are based on BSP model. But, the diversiform of application field, complex of data type and multiform of data processing flow directly lead to parallel computing model shows the characteristics of variety and short cycle, at the same time, thus increased the difficulty of application development and maintenance. There isn't a method of

ISSN: 2005-4238 IJAST

synthesizing parallel computing models to support the flexible and efficient platforms, and then to research the diversiform of data analysis and processing.

Based on the theory of λ -calculus, functional language and graph reduction, this work aims to research a synthesized method to synthesize the models. Section 2 introduces the theory of graph reduction, followed by an introduction to the MapReduce and BSP models in section 3. In section 4 the unified modality of parallel computing models is found, and then according to the reduction rules the synthetic model is produced. Section 5 gives the synthetic model implement framework and experiments result. The final section is the conclusions.

2. Graph Reduction

Graph reduction works with the mathematics foundation of λ -calculus and function model of LNF calculus[3]. So it could be as an execution model of functional language. At the same time, functional language is with λ -calculus and recursive function theory as theoretical basis, so graph reduction also could be interpreted as graphical expression of functional language or λ -calculus.

Based on λ -calculus and functional language, λ-expression could be represented by a graph, and the result of λ-expression is computing by the transformation rules of graph. The work procedure of graph reduction has the advantages of easy to share the public expression, without special structure to save the environment and high efficiency. λ -calculus and functional language are introduced as follows.

2.1. λ -calculus

What is usually called λ -calculus is a collection of several formal systems which are good at researching function definition, application and recursion, based on a notation invented by Alonzo Church in the 1930s[4]. They are designed to describe the most basic ways that operators or functions can be combined to form other operators. Meanwhile λ -calculus provides a definition method and some transformation rules for a function. Anyone computable function could use this form system to express and evaluated.

The transformation rules:

1) δ- transform reduction: bind variables could be replaced by other variables with the restriction rules.

2) β- eliminating reduction: it is used to express the concept of function computing, and the lambda expressions which couldn't perform β- eliminating are called paradigm.

3) η- transformation: form a new equivalence relation.

2.2. Functional Language

Functional language isn't with Von Neumann computer model as the background and makes the performance on computer as a mathematical functional performance. Its most important theoretical basis is λ -calculus, at the same time, the function of λ -calculus could make the function as the parameter and return value [5]. Beyond that functional language program form of pure functions, mainly includes the following characteristics:

1) A functional program hasn’t side effect and its state doesn’t be modified;

2) A functional program could be naturally represented as a tree or a graph;

3) An execution of functional program contains an expression computing;

4) A computing of expression forms of a series steps are called reductions to complete, and each reduction is corresponding to a local transformation of the graph;

5) Functional program has natural parallelism. So reductions could be run in different parts of the graph at the same time;

6) The communication of concurrent reductions between the processors is implicit;

7) When there isn't expression to be reduced, the computing is terminated, and the expression is called paradigm;

8) In the process of realizing functional program the smallest concurrent unit is a simple reduction.

3. The Parallel Computing Models

The parallel computing models of PRAM, BSP, LogP, C3, BDM, MapReduce, MPI, OpenMP have been existed in the parallel computing for a long time. Every one of them has themselves advantages and disadvantages. Now the popular models are MapReduce and BSP, although they have been widely adopted in a wide range of areas, it hasn't been determined whether MapReduce and BSP are better than other models [6-7]. For example, MapReduce doesn't have strong theoretical foundations, it is demanded to establish its relationship to other major parallel computation models such as BSP and PRAM.

The diversity and flexibility of the models reflect the necessity of synthetic model, and the trend of computing model research is synthetic the models of themselves advantages. In currently, the most methods of synthesizing computing models are based on MapReduce and BSP. Tomasz Kajdanowicz *et al.*, [8] research the iteration process for the big data of figures with BSP in the development environment of Hadoop. Mike Spreitzer *et al.*, [9] combine MapReduce with BSP and then generate the mechanism to support for selecting different approaches to process the same type data. Pan Wei *et al.*, [10] embedded BSP model into Map and Reduce phases, so complete the iterative process in the superstep of Map or Reduce phase.

The researching of this work is also based on MapReduce and BSP.

3.1. MapReduce

MapReduce model has been widely adopted in lots of industries to compute massive datasets, and it has been implemented by Hadoop, Phoenix, Disco, Mars, Cell MapReduce, FPMR, Ussop. In which of them Hadoop as a open source implementation platform has been currently used by more than one hundred companies including eBay, IBM, Yahoo!, Facebook, and Twitter, along with a number of universities[11].

MapReduce model has been used to solve a number of non-trivial problems in industry and academia. Its original design was to run on large clusters of being composed by hundreds or thousands of low-end commodity machines, and each machine contains a processor, a fast primary memory and a slower secondary memory. Furthermore, all the machines are connected via an underlying network and all the secondary memories form of a global shared memory, at the same time, each machine could access other machines' secondary memory remotely during synchronization.

MapReduce model computes the data in rounds, and there are two phases of map and reduce in every round, furthermore, every phase is composed of an input, a computation and an output. The input data in form of <key; value> pairs are all stored in global memory [12]. One round is beginning with map phase, only finish map phase the reduce phase could begin. And when map phase is finished the output data is written to share memory as the input of reduce phase before the output data is partitioned by the shuffle and sort step. The output of this phase is used as input of the next phase, so each machine working on reduce phase could read the data is written in map phase during synchronised, and the local primary memory is

cleared before every synchronisation. The work unit of each machine is called task, and the number of tasks which are composed of map tasks and reduce tasks is larger than the number of processors, but the number of reduce tasks is smaller than the number of map tasks. With all the tasks, the master processor controls how the tasks assigned across the other worker processors. Such as in map phase, when a processor finishes computing its task a new map task is assigned to it by master processor, all the map tasks are finished the reduce tasks will begin to perform.

3.2. BSP

BSP is a bridge model for parallel computing and put forward by G. Valiant. Its level structure contains computing phase, whole communication phase and barrier synchronisation phase, vertical structure is consisted by supersteps [13]. A BSP computer can be defined by p processors, each with its local memory, connected via some means of point-to-point communication. BSP model computes the data in supersteps beginning with each processor receives input, and then perform some computation asynchronously, communicate any output at the end. Barrier synchronisation is used at the end of every superstep to synchronise all the p processors. Each processor can communicate directly with every other processor, providing complete control over how the data is distributed between the processors in every superstep. The working principle is shown in Figure 1.

A program runs in BSP model can be measured by the computation time, communication cost for each superstep, and the number of supersteps. BSP model has four parameters (P, L, g, s): P-The number of virtual processors, L-The minimum delay time of executing whole barrier synchronization, or L- Send/receive a packet, g-The network bandwidth of whole communication, s-Computing velocity of CPU [14-15]. Suppose the BSP programs includes S supersteps, the execution time of superstep(i) is that:

$$\text{Max } \{ \text{Wi} \} + \text{Max } \{ \text{hi} * g \} + L$$

Among those, Wi is the number of local operation steps of every processor in this superstep, hi is the number of a processor sending or receiving the message packets in this superstep, the actual values of g and L usually depend on the topology of the network, routing algorithm, channel capacity and communication cycle time and other technology parameters. The entire program execution time is summation of all the supersteps time.

4. Synthesize the Parallel Computing Models

We have given the parallel computing models in an unified modality and defined the rules to be used reducing the theory figures of models, and then the synthetic model based on MapReduce and BSP is produced.

4.1. The Unified Modality of Models

There are lots of differences among the implementation methods of the parallel computing models, furthermore the models shown the characteristics of diversity, short cycle, and quick update. But all the models could be represented with a unified modality which forms of input part, read part, computing part, communication part and output part, and each part is an independent functional program. Based on the working principle of the models, MapReduce and BSP models could be partitioned into the unified modality which is shown in Figure 1.

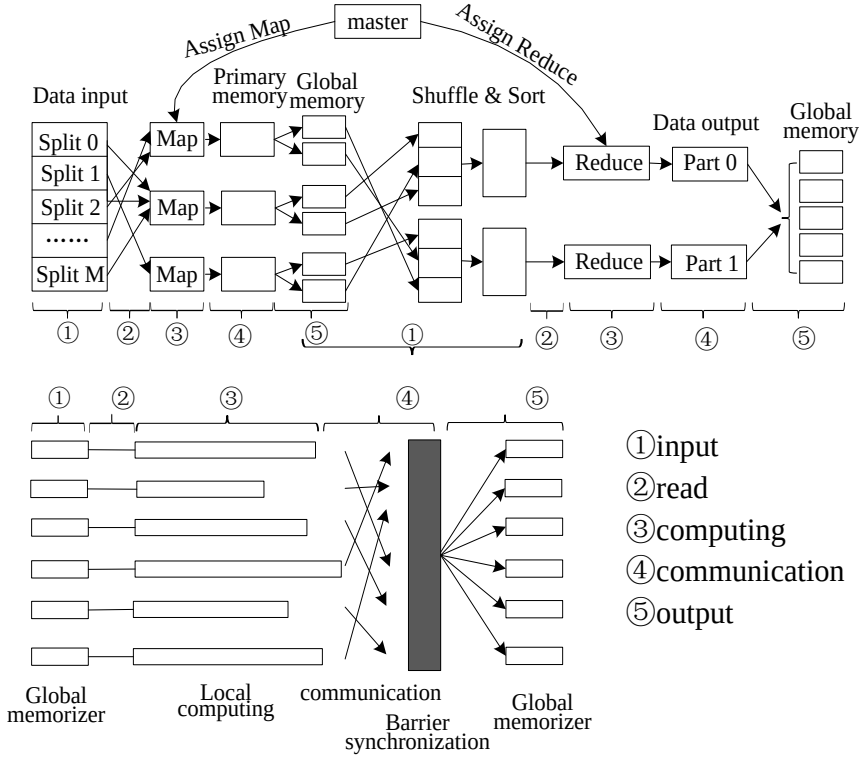


Figure 1. Partition Parallel Computing Models

4.2. The Reduction Rules

Based on functional language and λ-calculus, the unified model could be expressed by λ-expressions and it is shown in Fig.2. MapReduce model could be expressed by λ_{MR}, and the parts of MapReduce in the unified modality could be expressed by $\lambda_{MR\text{-}input}$, $\lambda_{MR\text{-}read}$, $\lambda_{MR\text{-}computing}$, $\lambda_{MR\text{-}communication}$ and $\lambda_{MR\text{-}output}$, furthermore define the function of λ_{MR} as $\lambda_{MR}(\lambda_{MR\text{-}input}, \lambda_{MR\text{-}read}, \lambda_{MR\text{-}computing}, \lambda_{MR\text{-}communication}, \lambda_{MR\text{-}output})$. In the same way, BSP model could be expressed by λ_{BSP}, and the function of λ_{BSP} is $\lambda_{BSP}(\lambda_{BSP\text{-}input}, \lambda_{BSP\text{-}read}, \lambda_{BSP\text{-}computing}, \lambda_{BSP\text{-}communication}, \lambda_{BSP\text{-}output})$. At the same time, because λ-expression could be represented by a graph, the λ-expressions of $\lambda_{MR}(\lambda_{MR\text{-}input}, \lambda_{MR\text{-}read}, \lambda_{MR\text{-}computing}, \lambda_{MR\text{-}communication}, \lambda_{MR\text{-}output})$ and $\lambda_{BSP}(\lambda_{BSP\text{-}input}, \lambda_{BSP\text{-}read}, \lambda_{BSP\text{-}computing}, \lambda_{BSP\text{-}communication}, \lambda_{BSP\text{-}output})$ could be represented by the corresponding graphs of $G_{MR}(g_{MR\text{-}input}, g_{MR\text{-}read}, g_{MR\text{-}computing}, g_{MR\text{-}communication}, g_{MR\text{-}output})$ and $G_{BSP}(g_{BSP\text{-}input}, g_{BSP\text{-}read}, g_{BSP\text{-}computing}, g_{BSP\text{-}communication}, g_{BSP\text{-}output})$. In this work, first of all, the data structure of node in the graphs and the reduction rules are defined, and then λ-expressions are reduced by graph reduction with the rules, finally the graph of synthesize model is expressed.

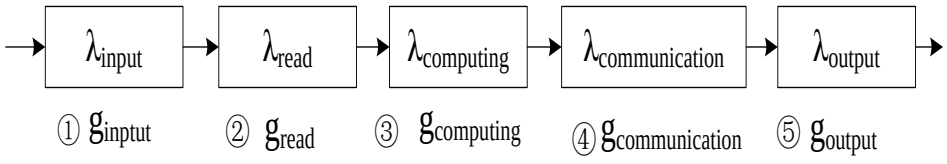


Figure 2. λ-expressions of Unified Model

Definition 1: Graph G ($G_{MR,}$ G_{BSP}) could be extended and the data structure of node in G is represented as four elements group:

< Name, g, In-degree, Out-degree >

Parameter g records the node belongs to which graph, and the graphs contain g_{input}, g_{read}, $g_{computing}$, $g_{communication}$, g_{output}.

In-degree is the number of arc points to the node;

Out-degree is the number of arc starts from the node.

Definition 2: The reduction rule of verifying node.

This rule is to verify the legitimacy of the nodes in G. Illegal node is defined that the node only has out-degrees (unless the root input node) or the node attributes are not the four elements group in definition 1. When there are illegal nodes in G, the graph of G could not be reduced.

Definition 3: The reduction rule of nodes merging.

If two or more nodes have the same function (or name), these nodes will be merge into one new node.

(1) The new node name remains unchanging.

(2) Parameter g records the original nodes belong to which graph (g_{input}, g_{read}, $g_{computing}$, $g_{communication}$, g_{output}).

(3) The new node's in-degree and out-degree are changed under definition 4.

Definition 4: The reduction rules of in-degree and out-degree.

Assuming that, the node of N is the merging node, and the change of in-degree and out-degree could be simulated with the basic models in Figure 3.

(1) In-degree of N increases: graph A and graph B are merged graph C.

(2) In-degree of N chooses the max value among the original nodes with same name: graph A and graph C are merged graph C or graph B and graph C are merged graph C.

(3) out-degree of N increases: graph D and graph E are merged graph F.

(4) out-degree of N chooses the max value among the original nodes with same name: graph D and graph F are merged graph F or graph E and graph F are merged graph F.

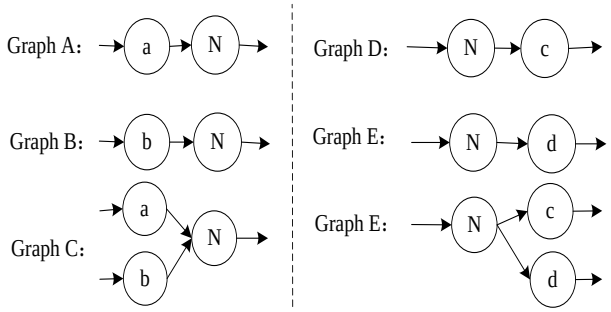


Figure 3. The Basic Graph of in-degree and out-degree

Definition 5: The reduction rule of release useless node.

When the nodes are merged, only one node is effective, other nodes are as useless nodes. But the useless nodes also occupy lots of storage resources. For saving the resources and improving the performance of the reduction, useless nodes need to be released.

4.3. Synthetic Model

Definition 6: (1) $\lambda=\lambda_{MR}+\lambda_{BSP}$: the λ-expression of synthetic model which are formed by parallel computing models.

(2) $G=G_{MR}+G_{BSP}$: the reduction graph of synthetic model.

(3) +: the reduction rules of graph reduction.

With the unified modality of MapReduce and BSP, graph reduction adopts the reduction rules of verification and merging to verify the legitimacy of the nodes and merge the nodes with same name, then changes the values in the four elements group and releases the useless nodes under definition 4 and 5. Finally, the reduction graph of synthetic parallel model is formed and shown in Figure 4.

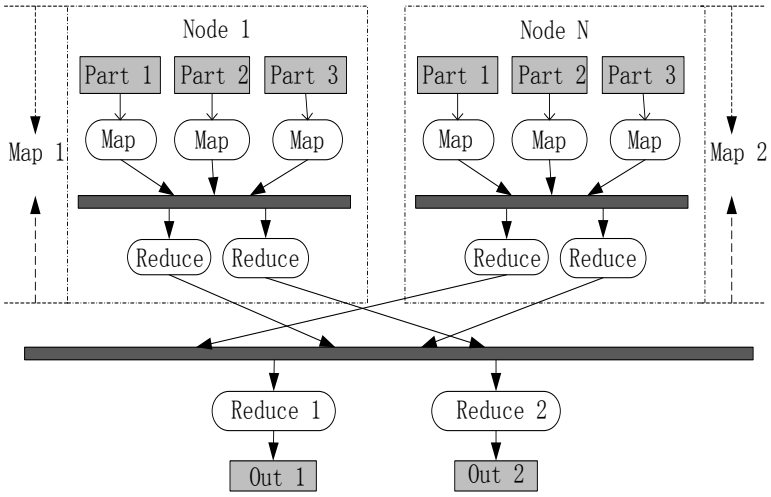


Figure 4. Synthetic Model

Figure 4 shows that under the reduction rules, the synthetic model merges the same functional parts of the original models and selects the superior performance mechanisms from the original models. So the synthetic model involves all the advantages of MapReduce and BSP models, thus makes the synthetic model has higher performance. For example, MapReduce adopts the method of separating computing with communication, and adopts barrier synchronization to perform the interaction between Map and Reduce. Meanwhile, each task only uses the data stored in the local memory. This way not only avoids the interaction between the tasks with same type, but also avoids communication delay and deadlock in computing process. Another model BSP could set checkpoints for fault tolerance, and the correctness and time complexity of the BSP programs also could be forecast.

5. The Implementation Architecture of Synthetic Model and Experiments

We have implemented synthetic model. This section presents the synthetic model architecture and the experimental results.

5.1. Synthetic Model Architecture

Based on the procedure of MapRedce, one MapReduce round could be split into three phases which are map phase, shuffle phase and reduce phase. Map and Reduce phases are separate, and shuffle phase is the data bridge among map phase and reduce phase, meanwhile shuffle phase determines how data is distributed amongst the reduce tasks, but the data assigned to each reduce task is sorted and combined by key in reduce phase.

As shown in Figure 4, in a MapReduce round, each phase of map and reduce contains three parts of reading input data, performing some computation and outputting any results. In a superstep, the BSP model also has the three parts. So map phase or reduce phase can be simulated by a BSP superstep, MapReduce model implies BSP model. Inverse, if the map phase is ignored, each reduce task as a BSP processor, shuffle phase as the BSP input, BSP model could be simulated by MapReduce model, BSP model also implies MapReduce model. Figure 5 shows the implementation architecture of synthetic model.

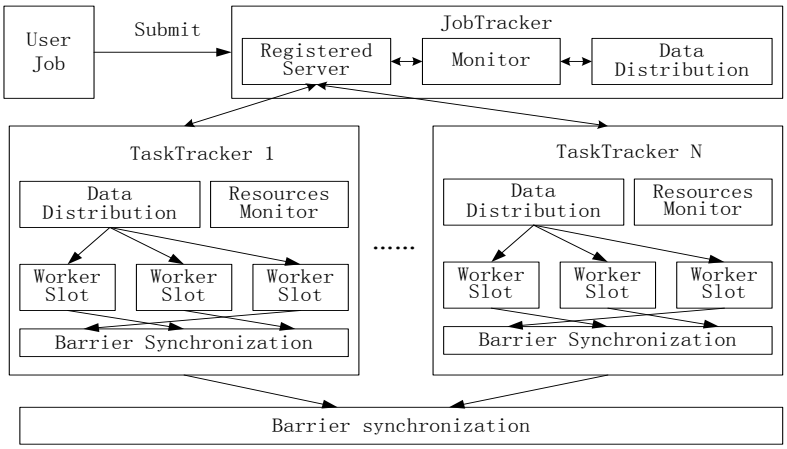


Figure 5. Implementation Architecture of Synthetic Model

Synthetic model architecture is composed of a JobTracker and a set of TaskTrackers. The JobTracker has three components: Registered Server is used to connect with the nodes, The Monitor saves the resources and monitor the tasks state real-time, and the Data Distribution is divide data and then send to TaskTracker through the Registered Server. A TaskTracker of which is in one computing node has four components: Data Distribution is divide data and send to Worker Slots, The Worker Slots present a set of concurrent threads, Resources Monitor is control the resources and Worker Slots state, Barrier synchronization is used to control deadlock of the tasks.

5.2. Experiment Results

In this section, there are two experiments all running in the virtualization cluster which is constituted by three physical nodes. The experiments purpose is count the occurrences number of each word in one file and the experiment data is in the Table 1.

Table 1. Experiment Data

File Name	File 1	File 2	File 3	File 4	File 5
Data (byte)	13664	33534	73898	103085	133514

In the first experiment, the CPU of every node has four cores and one core presents one worker slot. We compute the data by hadoop and synthetic model respectively in same cluster, the result is in Figure 6. The Figure 6 shown that the performance of synthetic model is higher than hadoop, and the growth trend of execution time is slow by the data number growth.

In the second experiment, there are four groups of results respectively represent that the CPU of every node has four cores, three cores, two cores and one core. The results is shown in Figure 7. We could draw out that the execution time is decrease by the cores number of CPU increase and the execution time is increase by the data number increase, but no matter what the cores number is the increase trend of execution time is slow based on the data number average growth.

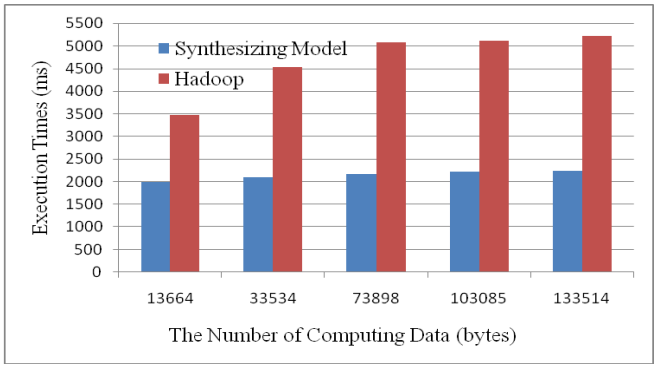


Figure 6. The Execution Results of Hadoop and Synthetic Model

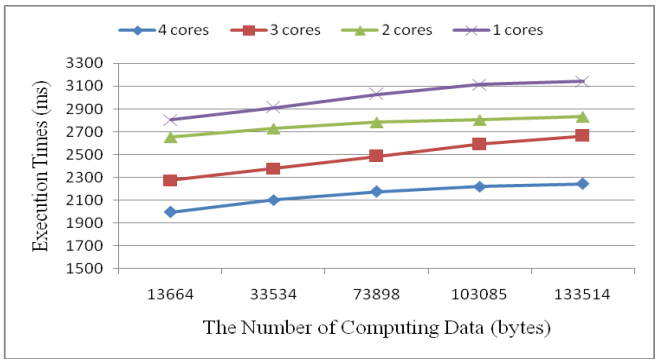


Figure 7. The Execution Results of Different Number of Cores

6. Conclusions and Future Work

This work adopts graph reduction to synthesize parallel computing models could eliminate the diversity and heterogeneity among the models, integrate a variety of high-performance modules of the models, and provide a synthetic model for analyzing and processing large amount and multi-type data. In this paper, we reduce the graphs of lambda-expressions to get the synthetic model, thus with the basis theoretical of lambda-calculation and functional language, and makes the synthetic method of models has strong theoretical foundations.

As future work, we plan to research the resources manager of the synthetic model. We also adopt graph reduction to manage the resources to be dynamic released and applied by the application program. The resources of dynamic reconfiguration play an important role in parallel computing.

Acknowledgements

This work is supported by the follow projects:

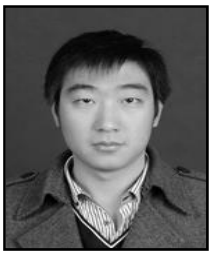
(1)The National High Technology Research and Development Program of China (863 Program) under Grant NO.2013AA 01A 603;

(2)Innovation Action Plan supported by Science and Technology Commission of Shanghai Municipality (No.11511500200);

References

[1] Lynch C, "How do your data grow?", Nature, vol.455, **(2008)**.
[2] Qin Xion-Ppai, Wang Hui-Ju, LI Fu-Rong, LI Cui-Ping, CHEN Hong, ZHOU Xuan, DU Xiao-Yong, WANG Shan, "New Landscape of Data Management Technologies", Journal of Software, vol.24, no.2, **(2012)**, pp.175-197.
[3] Huang Linpeng, Ni Deming, Sun Yongqiang, "CM: A Concurrent Functional Language Based on Miranda", Computer Engineering & Design, no.4, **(1994)**, pp.38-45.
[4] Yang Xiangjin, "Graphic Reduction Calculus in Implementation of Functional Programming", Journal of Computer Research and Development, no.12, **(1993)**, pp.12-25.
[5] Jeff Epstein, Andrew P. Black, Simon Peyton-Jones, "Towards Haskell in the cloud", 4th ACM Haskell Symposium, **(2011)**September 22; Tokyo, Japan, pp.118-129.
[6] G. Jung, N. Gnanasambandam, T. Mukherjee, "Synchronous Parallel Processing of Big-Data", IEEE 5th International Conference on Cloud Computing, **(2012)**June 24-19; Honolulu, USA, pp.811-818.
[7] Pace, Matthew Felice, "BSP vs MapReduce", Procedia Computer Science, Vol.9, **(2012)**, pp.246-255.
[8] Tomasz Kajdanowicz, Wojciech Indyk, Przemyslaw Kazienko, Jakub Kukul, "Comparison of the Efficiency of MapReduce and Bulk Synchronous Parallel Approaches to Large Network Processing", IEEE 12th International Conference on Data Mining Workshops, **(2012)**December 10; Brussels, Belgium, pp. 218-225.
[9] Mike Spreitzer, Malgorzata Steinder, Ian Whalley, "Ripple: Improved Architecture and Programming Model for Bulk Synchronous Parallel Style of Analytics", IEEE 33rd International Conference on Distributed Computing Systems, **(2013)**July 8-11; Philadelphia, USA, pp. 460-469.
[10] PAN Wei, LI Zhan-Huai, WU Sai, Chen Qun, "Evaluating Large Graph Processing in MapReduce Based on Message Passing", JOURNAL OF COMPUTERS, Vol.34, No.10, **(2011),** pp.1768-1784.
[11] Li Jian-Jiang, Jian Cui, Wang Dan, Yan Lin, Huang Yi-Shuang, "Survey of MapReduce Parallel Programming Model", ACTA ELECTRONICA SINICA, Vol.39, No.11, **(2011)**, pp.2635-2642.
[12] Vianna Emanuel, Comarela Giovanni, Pontes Tatiana, Almeida, Jussara, Almeida Virgílio, Wilkinson Kevin, Kuno Harumi, Dayal Umeshwar, "Analytical Performance Models for MapReduce Workloads", International Journal of Parallel Programming, Vol.41, **(2013)**, pp.495–525.
[13] V. Niculescu, "Cost evaluation from specifications for BSP programs", Parallel and Distributed Processing Symposium, **(2006)**April 25-29; Rhodes Island, pp.320-320.
[14] Leslie G. Valiant, "A Bridging Model for Parallel Computation", Communications of the ACM, vol.33, no.8, **(1990)**, pp.103-111.
[15] Abu Salem, F, "A BSP Parallel Model of the Gottfert Algorithm for Polynomial Factorization over F_2", Parallel Processing and Applied Mathematics , vol.3019, **(2003)**, pp. 217-224.

Authors



Shen Chao, he is currently a Ph.D. student of Shanghai University. His primary research interests cover parallel computing, virtualization, and grid computing.

Tong Weiqin, he received his Ph.D. degree from Shanghai Jiao Tong University. He is currently a Professor of Shanghai University. His primary research interests cover High performance computing, embedded system and distributed system.

