Mimir+: An Optimized Framework of MapReduce on Heterogeneous High-Performance Computing System

Nan Hu, Zhiguang Chen, Yunfei Du, and Yutong Lu(✉)

School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
yutong.lu@nscc-gz.cn

Abstract. In this paper, we present an optimized data processing framework: Mimir+. Mimir+ is an implementation of MapReduce over MPI. In order to take full advantage of heterogeneous computing system, we propose the concept of Pre-acceleration to reconstruct a heterogeneous workflow and implement the interfaces of GPU so that Mimir+ can facilitate data processing through reasonable tasks and data scheduling between CPU and GPU. We evaluate Mimir+ via two benchmarks (i.e. the WordCount and large-scale matrix multiplication) on the Tianhe-2 supercomputing system. Experimental results demonstrate that Mimir+ achieves excellent acceleration effect compared with original Mimir.

Keywords: High-performance computing · MapReduce Heterogeneous

1 Introduction

With the continuous development of information technology, the data generated in daily life, industrial productions and scientific researches are exploding. The convergence of high-performance computing and big data processing is becoming a promising solution to efficiently tackle with the massive data.

MapReduce is a programming paradigm popularized by Google [1] which presents a parallel computing model and method for large-scale data processing. Implementations of MR-MPI [5] have given practical and feasible solutions to transplant MapReduce to high-performance computing system. However, MR-MPI suffers from a severe shortcoming which is its simple memory management. In our previous work, we presented Mimir [2] which is an optimized framework based on MR-MPI. Mimir redesigns the execution model to incorporate a number of sophisticated optimization techniques that achieve similar or better performance with significant reduction in the amount of memory used. Nevertheless, we can see that MR-MPI and Mimir mainly perform their calculation in CPUs.

https://doi.org/10.1007/978-3-030-05677-3_18

Among the latest TOP500 list published in June 2018, Summit captured the number one spot with a performance of 122.3 petaflops on High Performance Linpack. Each node of Summit is equipped with two 22-core Power9 CPUs, and six NVIDIA Tesla V100 GPUs. Summit's championship demonstrated the capabilities and potentiality of heterogeneous high-performance computing system. Although MapReduce-MPI and Mimir can implement the MapReduce model well on high-performance computing system, their lack of heterogeneous architecture will cause a problem that the heterogeneous resources cannot be fully utilized.

We continued the work of Mimir and present Mimir+ in this paper. This work targets to promote the calculation speed of Mimir and support heterogeneous GPU acceleration on high-performance computing system.

The remainder of the paper is organized as follows. Section 2 introduces the optimizations of Mimir+. Section 3 describes the experimental environment and results. Other research related to our paper is presented in Sect. 4. We conclude this paper in Sect. 5.

2 Design of Mimir+

In this section, we introduce the main optimizations and designs in Mimir+.

2.1 Heterogeneous Workflow

The original Mimir designs two special objects called KV containers and KMV containers to help manage the intermediate data $<key, value>$ pairs between map phase and reduce phase. Similar to Mimir and MR-MPI, Mimir+ still adopts the KV containers and four basic phases: map, aggregate, convert and reduce. However, in order to further improve the computation speed, we reconstruct a heterogeneous workflow for Mimir+. Specifically, Mimir+ integrates the map phase and the aggregate phase into one process called MAP, and the convert phase and the reduce phase are integrated into another process called REDUCE.

Figure 1 shows the reconstructed workflow of Mimir+. The first thing to do in MAP is to process the input data according to a user-defined callback function. Here, we implement a new interface for GPU to perform the map jobs and users can select whether to perform the calculation on the GPU or on CPU by using the corresponding interfaces. Then, Mimir+ performs the MPI_Alltoallv function to exchange the KVs and stored them in KVCs through an interleaved execution model. When the MAP process ends, the REDUCE process starts and Mimir+ converts $<key, value>$ in KVC to $<key, <value1, value2, value3...>>$ into the KMVC. Mimir+ also has two types of interfaces in reduce phase for users to determine whether they will use GPU to calculate the reduce jobs or not. A user-defined reduce_GPU function implemented in CUDA is needed to start the data processing in GPU and the final output data will be transferred back from GPU memory.

2.2 Design of GPU Acceleration Modules

When we implement the heterogeneous workflow of Mimir+, we can't simply load the map/reduce tasks and data into GPU because GPU is not suitable for receiving fragmented data. Here we propose a concept of Pre-acceleration. Pre-acceleration actually refers to the operations we perform before we use GPU to calculate data. Specifically, operations like data partitioning, data communication and data transmission required before GPU acceleration can all be regarded as a part of Pre-acceleration. In combining the concept of Pre-acceleration, we divide the GPU acceleration process into four modules to achieve an efficient and convenient management. Figure 2 shows a brief architecture of Pre-processing Module, Transmission Module, Calculation Module and Feedback Module.

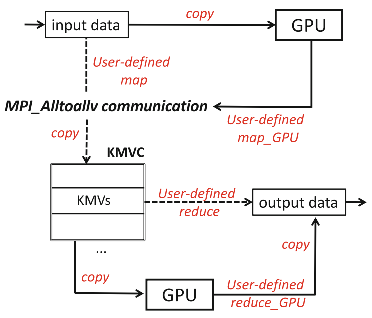


Fig. 1. Workflow of Mimir+

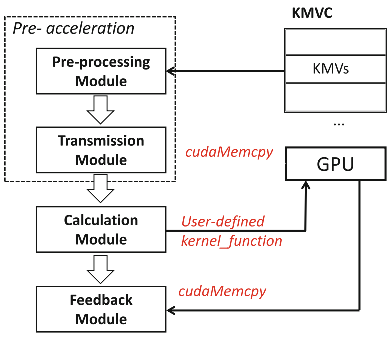


Fig. 2. GPU acceleration modules

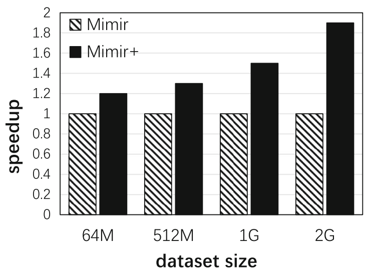
3 Evaluation

In this section, we evaluate the acceleration effect of Mimir+ and compare it with the original Mimir.

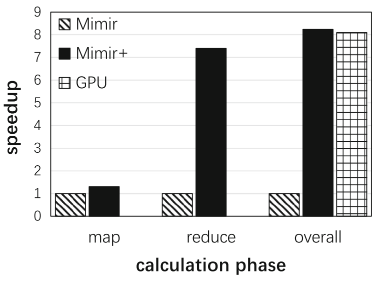
3.1 Performance Comparison

We perform WordCount (WC) and matrix multiplication on 4 nodes of heterogeneous computing system: Tianhe-2. Each node in Tianhe-2 is equipped with a 2-way 8-cores Intel Xeon CPU E5-4640, 128 GB memory, running at 2.40 GHz. The GPU equipped on the node is NVIDIA Tesla K80 GPU with two sets of 12 GB GDDR5 memory (24 GB in total), 4992 stream processors, the memory bandwidth is 240 GB/s. Each node installs a 64-bit Linux 3.10.0 operating system, and we use mvapich2-2.2, gcc-4.8.5 and CUDA 8.0 to conduct the tests on Mimir+.

The results of WC are shown in Fig. 3(a). As we can notice, Mimir+ obtains a comparatively good acceleration effect on WC. With the increase of test data,



(a) WordCount



(b) Matrix Multiplication

Fig. 3. Performance comparison on WordCount and matrix multiplication

the acceleration effect achieved by Mimir+ is becoming more and more obvious. Because the tasks of WordCount in MapReduce do not require intensive computing, the effect of acceleration in Mimir+ is not fully reflected.

Since the tasks of matrix multiplication vary from the map phase and the reduce phase, we tested and compared the two phases separately in the other experiment. Moreover, in order to compare the difference between the calculation of heterogeneous systems and pure parallel GPU computing, we performed the matrix multiplication in GPU alone using CUDA with the same input data and put the result into comparison. The results are shown in Fig. 3(b). In the map phase, because of the little calculation, the effect of acceleration is not good. However, in the reduce phase which contains a huge amount of calculation, Mimir+ achieves a considerable speedup of about 7.4 compared with Mimir. After comparing Mimir+ and Mimir in map and reduce phase, we performed an overall test and the whole framework can achieve a speedup of about 8.1 to 8.3. Nevertheless, the calculations performed in GPU alone achieved a speedup of about 8.5 which is close to Mimir+.

4 Related Work

MapReduce is an extremely popular model and many researches intend to improve the performance of MapReduce jobs on heterogeneous system.

Phoenix [6,8] proposed by Colby Ranger et al. from Stanford University is a MapReduce implementation on shared memory system targeting thread-based parallel programming. Shared memory minimizes indirect costs caused by parallel task spawning and data communication. Mrphi [4] is also a MapReduce implementation optimized for the Intel Xeon phi. Different from these systems, Mimir+ works on large-scale distributed-memory systems.

Mars [3] is a MapReduce implementation totally deployed on GPUs. In Mars, there are a large number of threads running in parallel on the GPUs. Each thread computes a KV pair at a time. To avoid multi-threaded write conflicts, Mars uses a lock-free strategy to ensure that parallel programs are correct, with minimal synchronization costs.

On high performance computing system, Tsoi et al. developed a heterogeneous computing system, Axel [7], which consist of FPGAs and GPUs, and they implemented a MapReduce framework on Axel which significantly promoted the speed of calculation.

5 Conclusion

In this paper, we propose an optimized MapReduce framework on heterogeneous high-performance computing system: Mimir+. This framework inherits the core idea of MR-MPI, reconstructs a heterogeneous workflow and implements the GPU acceleration interfaces so that we can accelerate the data processing of MapReduce jobs and make full use of resources on heterogeneous high-performance computing system. Our results on the Tianhe-2 supercomputer prove that Mimir+, compared to the original Mimir, significantly improves the speed of data processing during the computing phase for data-intensive applications.

Acknowledgments. We are grateful to the anonymous reviewers for their valuable suggestions that will be used to improve this paper. This work is partially supported by National Key R&D Program of China 2017YFB0202201, National Natural Science Foundation of China under U1611261, 61433019, U1435217, 61872392 and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant NO. 2016ZT06D211.

References

1. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. ACM (2008)
2. Gao, T., et al.: Mimir: memory-efficient and scalable mapreduce for large supercomputing systems. In: Parallel and Distributed Processing Symposium, pp. 1098–1108 (2017)
3. He, B., Fang, W., Luo, Q., Govindaraju, N.K., Wang, T.: Mars: a mapreduce framework on graphics processors. In: International Conference on Parallel Architectures and Compilation Techniques, pp. 260–269 (2008)
4. Lu, M., Liang, Y., Huynh, H.P., Ong, Z., He, B., Goh, R.S.M.: MrPhi: an optimized mapreduce framework on Intel Xeon Phi coprocessors. IEEE Trans. Parallel Distrib. Syst. **26**(11), 3066–3078 (2015)
5. Plimpton, S.J., Devine, K.D.: Mapreduce in MPI for large-scale graph algorithms. Parallel Comput. **37**(9), 610–632 (2011)
6. Talbot, J., Yoo, R.M., Kozyrakis, C.: Phoenix++: modular MapReduce for shared-memory systems. In: International Workshop on Mapreduce and ITS Applications, pp. 9–16 (2011)
7. Tsoi, K.H., Luk, W.: Axel: a heterogeneous cluster with FPGAS and GPUS. In: International Symposium on Field-Programmable Gate Arrays, pp. 115–124 (2010)
8. Yoo, R.M., Romano, A., Kozyrakis, C.: Phoenix rebirth: scalable MapReduce on a large-scale shared-memory system. In: IEEE International Symposium on Workload Characterization, pp. 198–207 (2011)