https://doi.org/10.1051/itmconf/20224702047

FPGA-based heterogeneous acceleration study for multidimensional cubing

Yuan Meng, *Jun* Yang*, and *Jun* Li

School of Information Science and Engineering, Yunnan University, Kunming, China

Abstract. Today's information processing not only faces an explosion of data volume and data dimensions, but also has to meet the growing user requirements for timeliness. The increase in the dimensionality of Kylin brings about the problem of dimensional explosion of multidimensional cubes, which puts great pressure on disk and network transmission. To solve the problem of dimensional explosion when building multidimensional cubes in Kylin, this paper proposes a bottom-up three-layer architecture that links from the hardware compression acceleration kernel layer to the query engine layer through the software driver layer. The software-driven layer is implemented through JNI, dynamic link libraries, and global shared resource pools to link the hardware-accelerated kernel layer to the query engine layer. Finally, comparative experiments on the performance of multidimensional cube building are conducted for heterogeneous clusters and normal clusters. The experimental results show that the hardware- accelerated cluster obtains a 3.7 times speedup ratio in build time and a 2 times speedup ratio in average query time, which can alleviate the IO pressure brought by Kylin during the dimensional explosion.

Keywords: Deflate, FPGA, OpenCL, Kylin, OLAP.

1 Introduction

Although Kylin can perform multidimensional analysis of complex and massive data with sub-second latency, it also has some problems. Figure 3 (a) shows the flowchart of building a Cube in Kylin layer by layer. Each layer in the diagram consists of several MapReduce tasks, which are executed serially between each layer.

This build process has more read and write operations on HDFS due to intermediate files between each Map task and Reduce task, and intermediate files between each layer of Map-Reduce tasks, which need to be cached on HDFS, resulting in an inefficient overall Cube build process. The FPGA implementation of the compression algorithm, with good parallelism, has a performance that is several times better than software compression. Using the FPGA-implemented Deflate algorithm to compress the intermediate files of Map can effectively relieve the pressure of the cluster facing dimensional explosion.

* Corresponding author: junyang@ynu.edu.cn

2 Related work

Apache Hadoop is an open source big data processing and storage platform [1]. The Map node maps the entire problem into several subproblems, and the Reduce phase merges and filters the solutions of the subproblems in the Map phase. However, for IO-intensive tasks, the Map task requires frequent reading and writing of data between hard disks and memory, and between different machines. The frequent IO operations make the computing performance of the cluster receive a great limitation. In order to solve this one problem, Spark designed a model based on in-memory computing, Spark's RDD operator uses memory for storage and computation, avoiding the impact of frequent disk reads and writes on system performance [2].

The horizontal scaling of distributed computing platform means, increasing the number of cluster nodes; while the vertical scaling of distributed computing platform means extending the computing capacity of a single node by integrating more co-processors (e.g., FPGA, GPU, etc.) in a single node [3]. Heterogeneous computing platforms, i.e., platforms implemented by vertically scaling the computational capacity of individual nodes as mentioned above. The second half of this section describes the current state of research on heterogeneous computing platforms.

Chen et al. added FPGAs to the Hadoop cluster to filter the data stored in the Hadoop distributed file system, and finally experimentally compared the system time overhead of FPGA acceleration over the native Hadoop system with increasing data volume at 1GB, 10GB, 30GB, and 50GB [4]. Heterogeneous optimizations have also been proposed for the Spark distributed framework, and Huang proposed a scalable GATK parallelization acceleration scheme [5].Chen et al. achieved a modest scaling of 2.6x by implementing a next-generation DNA sequencing application [6].

Huanget al proposed a framework for deploying FPGA gas pedals in distributed environments (e.g., Yarn and Spark) [7]. The proposed framework is used to implement four different types of applications on a CPU-FPGA cluster. Although a generic API is provided to access FPGA resources, a layer of cluster management is added on top of Spark, which is dedicated to FPGA resources. As a result, this approach increases the overhead of the data flow stack and limits the performance of the system. Huang used FPGA technology to accelerate the write data process and resource scheduling algorithm of Hadoop [8].

3 Design and Implementation of Cubing‘s Acceleration

3.1 Design and Implementation of Deflate FPGA Core

Figure 2 shows the overall architecture of the hardware compression acceleration kernel layer. The main functions of the KernelPool are to initialize, manage and maintain the OpenCL context and kernel resources; to implement algorithms such as Huffman tree building, which is not suitable for FPGA development boards (e.g., Huffman tree building process is serial and difficult to implement in parallel). Transfer the data to be compressed to the FPGA. Receive the compressed data from the FPGA; And send the compressed data to DeflateKernel via Kafka.

DeflateKernel does the most time-consuming LZ77 coding tasks and is written in the C-like language provided by OpenCL. The compiler maps the DeflateKernel to a program that can run on the board based on the BSP provided by the FPGA developer. The kernel program is then burned to run on the development board.

$$\boldsymbol{CompressionRatio} = \frac{\boldsymbol{input\_size} - \boldsymbol{output\_size}}{\boldsymbol{input\_size}} \times \mathbf{100\%} \tag{1}$$

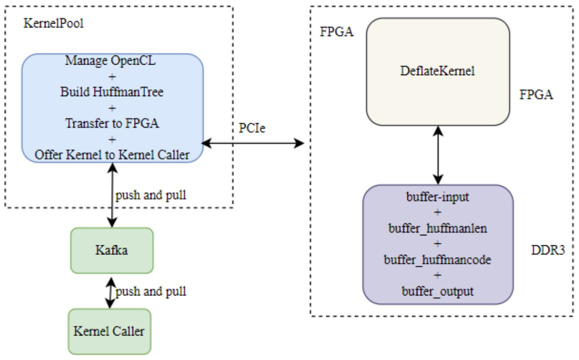
KernelPool
Manage OpenCL + Build HuffmanTree + Transfer to FPGA + Offer Kernel to Kernel Caller
PCIe
push and pull
Kafka
push and pull
Kernel Caller
FPGA
DeflateKernel
FPGA
buffer-input + buffer_huffmanlen + buffer_huffmancode + buffer_output
DDR3

Fig. 2. Architecture diagram of the hardware compression acceleration layer.

Experiments on the compression ratio of the FPGA compression cores implemented in this paper based on the Calgary standard evaluation set are floating from 2.01 to 2.62. The average compression ratio is 2.336. Comparing with previous similar studies, it is known that Alterna OpenCL implemented a compression ratio of 2.17 [9] and Verilog's scheme achieved a compression ratio of 2.18 [10]. Compared to CPU schemes, Intel's compression ratio is 2.18 [11].

The results of the compression speed comparison experiments are shown in Table 2. On the Calgary dataset, Enwik8 and CanterBury datasets, the FPGA-based implementation of the hardware compression kernel compresses 8 times faster than the CPU implementation of the Deflate algorithm, with a 9 times speedup ratio on Enwik7 and a slightly worse performance of 7 times on Silesia's dataset.

Table 2. Compression speed comparison test results.

	Throughput (MB/s)		Speedup
	Software	FPGA	
Calgary	366	2938	8.03
Canterbury	348	2788	8.01
Enwik7	332	3002	9.04
Enwik8	357	2899	8.12
Silesia	373	2786	7.4

3.2 Integration of Hadoop and DeflateKernel

Figure 3(a) shows the problem with Kylin's Cubing build engine, which generates a large amount of intermediate data. The intermediate files are stored in HDFS. Compressing the intermediate files of Map can reduce the amount of data read and write to disk and the amount of data transferred over the network, thus reducing IO time and improving cluster performance. Figure 3(b) illustrates the layer-by-layer construction process of Cube under acceleration. Unlike Figure 3(a) , the intermediate files output from the Map task are compressed, which can effectively relieve the pressure on disk and network reads and writes, thus achieving the purpose of accelerating the Map-Reduce tasks in the Hadoop cluster.

3.3 Design and implementation of by layer cubing

The star model constructed by selecting dwd_fact_order_detail as the fact table and dwd_dim_base_province, dim_sku, and dim_user_info_his three dimension tables is illustrated. Where the dimensions are selected as region [reg], province [pro], gender [gen] and brand [tm]. All dimensions on it are aggregated, and there are 16 Cuboid in total, as shown in Figure 4.

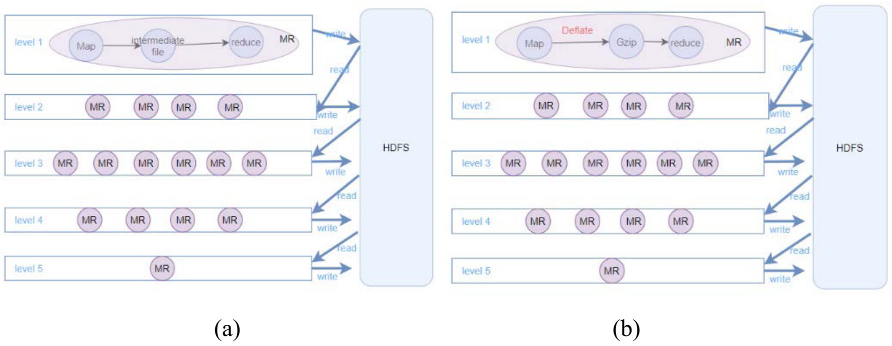

Fig. 3. Traditional Cubing and Accelerated Cubing.

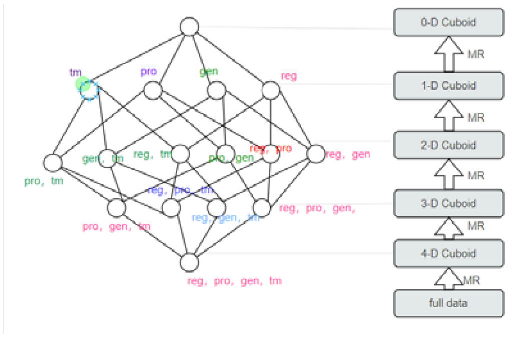

Fig. 4. By layer Cubing.

There are four combinations of [tm], [pro], [gen], and [reg] in one dimension; six combinations in two dimensions; four combinations of [tm, pro ,gen], [tm, gen, reg] , [pro, reg, gen], and [tm, pro, reg] in three dimensions; and one each in zero and four dimensions. The multidimensional cuboid is constructed layer by layer as follows: the original data is input, run a round of Map-Reduce task to construct Base Cuboid, i.e., 4-D Cuboid. Then 4-D Cuboid is used as input to start the next round of Map-Reduce task, because there are four dimensions, four Cuboids will be generated. The next process is similar in principle, and the 3-D Cuboid is used as input to get six 2-D Cuboids. until finally the 0-D Cuboid is obtained.

4 Experimental result and analysis

Figure 5 shows the average build time of the Cube for the two scenarios. The average100 is the Cube build time per 100 thousand data volume for 5 experiments using a weighted average. The Cube has the advantage of building more layers of MR tasks, generating more intermediate files, and calling the hardware compression core multiple times. However, you can see that the data volume from 100000 to 200000 does not take more than 2x, but from 200000 to 500000 and from 100000 to 400000 is significantly more than 2.5x and 4x, indicating that the performance decreases as the data volume increases and the cluster load increases. The speedup ratio decreases with increasing data volume because the Cube construction needs to start more Map tasks as the data size becomes larger.

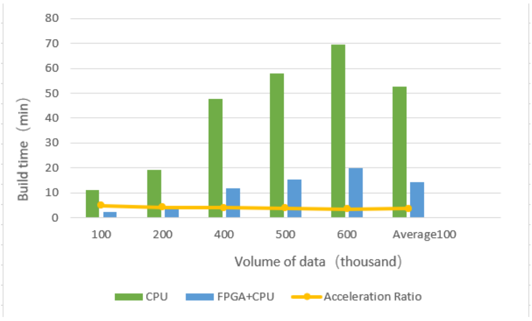

Fig. 5. Build time comparison results.

At this stage, there is only one FPGA accelerator board in the experimental environment used in this paper. When the accelerator board is occupied, Map tasks that do not obtain acceleration resources are waiting for the release of acceleration resources, and the potential of the cluster accelerable can be improved by adding an accelerator board. The cube building model based on query logs proposed in this paper is 3.7159 times longer than the cube building time of hardware and software co-acceleration in terms of first building time. the average SQL query time is tested by writing 35 SQL statements for testing. The average query time was recorded in the log for each experiment, and the "Average Query Time" was calculated by calling Kylin's Restful API.

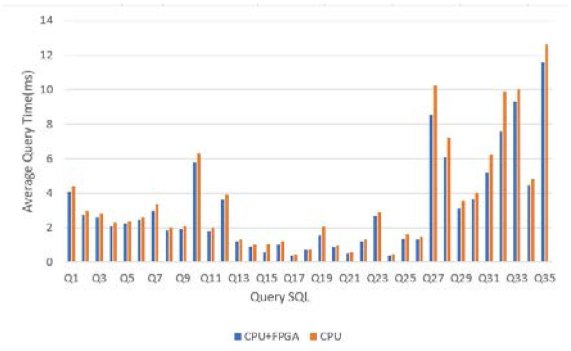

Fig. 6. Average Query time comparison results.

From Figure 6, we can see that the average query time of the cluster with hardware and software co-acceleration is slightly better, but the acceleration is not obvious, because the query principle of Kylin reads precomputed Cube from HBase, and the average acceleration rate is 1.98.

References

1. Kumar Nitin. Big Data Using Hadoop and Hive [M].Mercury Learning and Information: 2021-03-24.
2. Daniel C. M. de Oliveira, Ji Liu, Esther Pacitti, H. V. Jagadish. Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments [M].Morgan & Claypool Publishers: 2019-05-13.
3. D. Singh and C. K. Reddy, A survey on platforms for big data analytics [J], Journal of Big Data, 2014, issue 2, pp.8

4. Cheng, Zhang, et al. Research on efficient data filtering technology based on FPGA [J]. Microelectronics and Computers, 2017, 34(12)130-133+137.DOI:10.19304/j.cnki.issn1000-7180.2017.12.027.
5. Huang., Design and implementation of a parallel acceleration scheme for GATK gene analysis software [D].Huazhong University of Science and TechnoLogy,2019.DOI:10.27157/d.cnki.ghzku.2019.004595.
6. Chen Y-T, et al. When Spark meets FPGAs: a case study for next-generation DNA sequencing acceleration. In: 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16); 2016.
7. Huang M , Wu D , Yu C H , et al. Programming and Runtime Support to Blaze FPGA Accelerator Deployment at Datacenter Scale. Proc ACM Symp Cloud Comput, 2016:456-469.
8. Huang, Research on YARN heterogeneous cluster management method based on FPGA acceleration [D]. Huazhong University of Science and Technology, 2017.
9. Mohamed S, Abdelfattah, Andrei Hagiescu, Deshanand Singh. Gzip on a Chip: High Performance Lossless Data Compression on FPGAs using OpenCL[C]// International Workshop on Opencl. ACM, 2014:4.
10. Craft D J. A fast hardware data compression algorithm and some algorithmic extensions [J]. IBM Journal of Research and Development, 1998, 42(6): 733-746.
11. Gopal V, Guilford J, Feghali W, et al. High Performance DEFLATE on Intel Architecture Processors (2011) [J].