GMAPREDUCE: A SELF-ADAPTION MAPREDUCE FRAMEWORK BASED ON GRANULAR COMPUTING*

Junbo Zhang, Tianrui Li, Fei Teng, Chuan Luo

School of Information Science and Technology,
Southwest Jiaotong University, Chengdu 610031, China
E-mail: {jbzhang,luochuan}@my.swjtu.edu.cn, {trli, fteng}@swjtu.edu.cn

In this paper, we propose a self-adaption MapReduce framework based on granular computing (GrC), named gMapReduce. In the MapReduce model, the input data is partitioned into many data blocks which is the key step for the following parallel processing. The number of data blocks depends on the size of the block. It means the block size will affect the total running time. According to the proposed gMapReduce model, we design two algorithms, naive and advanced, for finding the appropriate granule. Both two algorithms can find the appropriate size of data block, thereby accelerating the running process effectively.

Keywords: Granular Computing; MapReduce; Hadoop; Cloud Computing.

1. Introduction

Granular computing (GrC) emerges as a new and rapidly growing paradigm of information processing, and has received much attention in recent years. GrC is an umbrella term to cover any methodologies, theories, techniques, and tools that make use of information granules in complex problem solving.[1] MapReduce, by Google, is a scalable and fault-tolerant data processing tool that enables to process a vast amount of data in parallel with many low-end computing nodes.[2,3] Hadoop is one of the most popular MapReduce runtime system.[4]

In MapReduce, the input data is partitioned into many data blocks which is the key step for the following parallel processing. We desire to

*This work is supported by the National Science Foundation of China (Nos. 60873108, 61175047, 61100117, 61202043), and NSAF (No. U1230117). the Fundamental Research Funds for the Central Universities (No. SWJTU12CX098), the Science and Technology Planning Project of Sichuan Province (No. 2012RZ0009), China, and the Fostering Foundation for the Excellent Ph.D. Dissertation of Southwest Jiaotong University 2012.

know how the block size affects the running time. We only change the block size and keep other arguments constant. The running time varies with the variation of block size as shown in Figure 2(a). From the point of GrC, the *block size* can be viewed as a *granule*. It performs the best when the block size is 32MB. The number of blocks is equal to the size of input data divided by the size of the block. It means that the smaller size, the more blocks. There are 493 and 2 blocks when the block size is equal to 1MB and 256MB, respectively. When there are more blocks, it would increase the I/O time because the cluster processes more times repeatedly. Conversely, the working slots in the cluster would be idle when there are fewer blocks. In this experiment, 14 map slots are idle when the block size equals to 256MB.

1.1. *Problem description*

Before describing the problem, we first give the following notation.

Notation:

D: input data;

F: specific function, *e.g.*, WordCount;

P: computing resource, *e.g.*, working slots in Map phase;

$G = \{g_1, g_2, \cdots, g_n\}$: a finite set of granules;

i: granule index;

T: the function of running time.

Given the input data D, the specific function F, the working cores P, and a finite set of granules $G = \{g_1, g_2, \cdots, g_n\}$, it aims to find the appropriate granule $g_\kappa \in G$ so as to minimize the running time:

$$\arg\min_{g_\kappa} T(D, F, P, g_\kappa) \tag{1}$$

Example 1.1. The example in Section 1.1 can be described as: D is the input data, whose size is 493MB; F is the specific function, *i.e.*, WordCount in Hadoop; P is the computing resource, *i.e.*, working slots in Map phase $P = 16$; $G = \{1MB, 2MB, ..., 256MB\}$. According to Figure 2(a), we find the appropriate granule $g_\kappa = 32MB$.

2. A GrC-based MapReduce Framework

2.1. *The MapReduce model*

The MapReduce model always includes the following stages: (1) Map phase; (2) Sort/Shuffle phase; (3) Reduce phase.

Definition 2.1. In MapReduce, the total running time is denoted as

$$T_{all} = T_m + T_r + T_o \tag{2}$$

where T_m is the running time in Map phase; T_r is the running time in Reduce phase; T_o is the running time in other phases, *e.g.*, sort/shuffle, pass intermediate data, output result.

2.2. *The gMapReduce model*

In this subsection, we give a GrC-based MapReduce framework, called gMapReduce, as shown in Figure 1.

The difference between the gMapReduce model and MapReduce is the size of the data block. As we know, in MapReduce, the block size is set by user, which is always static, fixed and experience-based. However, the idle resource of the running cluster varies with time. The fixed block size can not match the cluster perfectly. Therefore, we propose the gMapReduce model, which can give a dynamic and learning-based block size.

The gMapReduce model includes the following steps:

(1) Sample the data with different granules;
(2) Upload the data blocks;
(3) Start MapReduce Jobs for each block;
(4) Record the running time of each Job;
(5) Select the appropriate granule which satisfies the objective function.

Here, the objective function is the total running time. Assume that the granule g_i is selected and the corresponding running time is t_i. Hence, the number of total blocks is $nb_i = \left\lceil \frac{D}{g_i} \right\rceil$. Assume that p cores are idle in the cluster. Then, the repeat times is $\left\lceil \frac{nb_i}{p} \right\rceil$. Therefore, the estimated time is $et_i = t_i \times \left\lceil \frac{nb_i}{p} \right\rceil$. The granule g_κ will be selected which satisfies $et_\kappa = \min_{g_i \in G}\{et_i\}$. The detailed algorithm is outlined in Algorithm 1.

Algorithm 1 is a naive greedy algorithm for finding the appropriate granule. When we compare the estimated time and the actual time, the result is not ideal. The main reason is that it does not consider the initial time of the Job. Hence, we design an advanced greedy algorithm for finding the appropriate granule. Before it, we present the new definition of the total runtime as follows.

Definition 2.2. In MapReduce, the total running time is denoted as

$$T_{all} = T_i + T_m + T_r + T_o \tag{3}$$

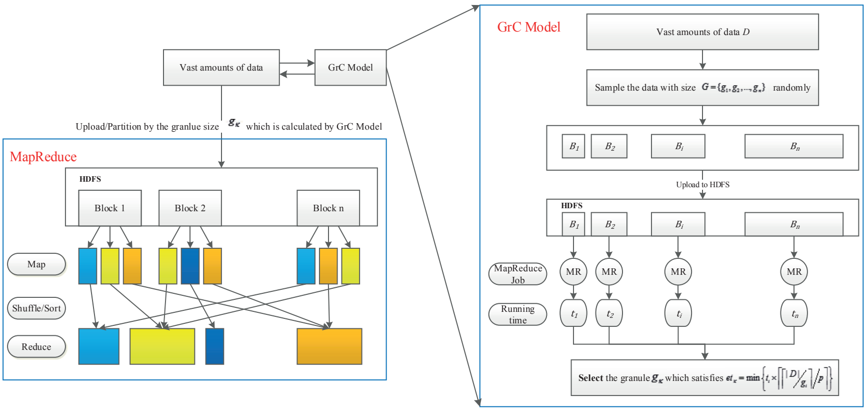

Vast amounts of data
GrC Model
Upload/Partition by the granlue size g_κ which is calculated by GrC Model
MapReduce
HDFS
Block 1
Block 2
Block n
Map
Shuffle/Sort
Reduce
GrC Model
Vast amounts of data D
Sample the data with size $G=\{g_1,g_2,\ldots,g_n\}$ randomly
B_1
B_2
B_i
B_n
Upload to HDFS
HDFS
MapReduce Job
MR
Running time
t_1
t_2
t_i
t_n
Select the granule g_κ which satisfies $et_\kappa = \min\{t_i \times \lceil \lceil |D|/g_i \rceil / p \rceil\}$

Fig. 1. The gMapReduce model

where T_i is the initialization time; T_m is the running time in Map phase; T_r is the running time in Reduce phase; T_o is the running time in other phases, *e.g.*, sort/shuffle, pass intermediate data, output result.

Algorithm 1: A naive greedy algorithm

```
   Input: $G = \{g_1, g_2, \cdots, g_n\}$; $D$; $p$.
   Output: The appropriate granule: $g_\kappa$
1  begin
2      for $g_i \in G$ do
3          $B_i = S(D, g_i)$; // Sample the data with size $g_i$ from $D$
4          $t_i = \mathrm{RunJob}(B_i)$; // Record the running time of $B_i$
5          $nb_i = \left\lceil \frac{|D|}{g_i} \right\rceil$; // Compute the number of blocks
6          $et_i = t_i \times \left\lceil \frac{nb_i}{p} \right\rceil$; // Estimated time when partitioning data with $g_i$
7      Select the granule $g_\kappa$ which satisfies $et_\kappa = \min_{g_i \in G}\{et_i\}$;
8      Output $g_\kappa$.
```

Algorithm 2 first tests the initialization time of the Job. To test the initialization time, a Job with empty input data set is submitted to the cluster, then the running time is recorded. The following steps but the function of the estimated time are the same with Algorithm 1. The function of the estimated time is changed to $(t_i - t_\emptyset) \times \left\lceil \frac{nb_i}{p} \right\rceil + t_\emptyset$, where $t_\emptyset$ is the initialization time of the Job.

Algorithm 2: An advanced greedy algorithm

```
    Input: $G = \{g_1, g_2, \cdots, g_n\}$; $D$; $p$.
    Output: The appropriate granule: $g_\kappa$
 1  begin
 2      Let $B_\emptyset = \emptyset$; // Set the data $B_\emptyset$ with empty set
 3      $t_\emptyset = \mathrm{RunJob}(B_\emptyset)$;
 4      for $g_i \in G$ do
 5          $B_i = S(D, g_i)$; // Sample the data with size $g_i$ from $D$
 6          $t_i = \mathrm{RunJob}(B_i)$; // Record the running time of $B_i$
 7          $nb_i = \left\lceil \frac{|D|}{g_i} \right\rceil$; // Compute the number of blocks
 8          $et_i^A = (t_i - t_\emptyset) \times \left\lceil \frac{nb_i}{p} \right\rceil + t_\emptyset$; // Estimated time
 9      Select the granule $g_\kappa$ which satisfies $et_\kappa = \min_{g_i \in G}\{et_i^A\}$;
10      Output $g_\kappa$.
```

3. Experimental Analysis

To test the proposed algorithms, the experiments are carried out on two different Hadoop clusters, which are outlined in Table 1.

Table 1. The description of the clusters

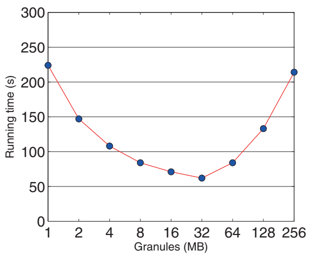
	Small cluster	Large cluster
Hadoop version	1.0.1	1.0.1
JDK version	1.7.0_5	1.7.0_5
Operating system	Ubuntu 12.04	CentOS 6.2
CPU	Inter(R) Core(TM) i7-2670QM	AMD Opteron(TM) Processor 2376
Clock frequency	2.20GHz	2.3GHz
Total working cores	4	32
Total Map slots	2	16
Total Reduce slots	2	16

We set the set of granules $G = \{1, 2, 4, 8, 16, 32, 64\}$ with the unit MB. Table 2 shows the estimated time of two algorithms on both clusters. In the small cluster, both two algorithms find the appropriate granule is 64MB. However, in the large cluster, they find the appropriate granule is 32MB. It shows that different clusters have different appropriate granules as different software and hardware configurations.

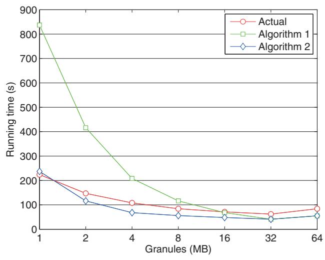
To verify the difference between Algorithms 1 and 2, we give a comparison of actual running time and estimated time of Algorithms 1 and 2 in the large cluster as shown in Figure 2(b). It is obvious to show that the curve of the estimated time of Algorithm 2 is close to that of actual running time. Therefore, Algorithm 2 has better estimate than Algorithm 1.

Table 2. Estimated Time

Small cluster				Large cluster			
g_i	t_i	et_i	et_i^A	g_i	t_i	et_i	et_i^A
1	28	6916	1996	1	27	837	237
2	27	3321	881	2	26	416	116
4	30	1860	640	4	26	208	68
8	26	806	206	8	29	116	56
16	29	464	164	16	34	68	48
32	39	312	172	32	41	**41**	**41**
64	49	**196**	**136**	64	55	55	55



(a)



(b)

Fig. 2. (a) Running time with different granules (block sizes). (b)A comparison of actual running time and estimated time of Algorithms 1 and 2.

4. Conclusion

In this paper, we proposed the gMapReduce model and designed two algorithms for finding the appropriate granule. Both algorithms can find the appropriate block size, and the advanced algorithm has better performance than the naive one, thereby accelerating the running process effectively. However, there still exist two problems. One is how to give the set of granules. In this paper, we present a simple set of granules in our experiments. Another is how to estimate the running time exactly. We will research on these two aspects in the future.

References

1. J. Yao, A. Vasilakos and W. Pedrycz, *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* **PP**, 1 (2013).
2. J. Dean and S. Ghemawat, *Communications of the ACM* **51**, 107(January 2008).
3. K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung and B. Moon, *SIGMOD Rec.* **40**, 11(January 2012).
4. T. White, *Hadoop: The Definitive Guide*, 2nd edn. (O'Reilly Media / Yahoo Press, 2010).