Proceedings of the 6th World Congress on Intelligent Control
and Automation, June 21 - 23, 2006, Dalian, China

The Study of Parallel K-Means Algorithm

Yufang Zhang, Zhongyang Xiong, Jiali Mao and Ling Ou

Department of Computer Science, Chongqing University,

Chongqing, China, 400044

zhangyf@cqu.edu.cn

Abstract **- Clustering analysis plays an important role in scientific research and commercial application. K-means algorithm is a widely used partition method in clustering. As the dataset's scale increases rapidly, it is difficult to use K-means to deal with massive amount of data. A parallel strategy is incorporated into clustering method and a parallel K-means algorithm is proposed. For enhancing the efficiency of parallel K-means, dynamic load balance is introduced. Data parallel strategy and Master/Slave model are adopted. The experiments demonstrate that the parallel K-means has higher efficiency and universal use.**

Index terms - Parallel Clustering. the K-means Algorithm. PVM.

I. The Status Of Parallel Clustering

For processing massive data or distributed data extensively, a high efficient clustering data mining method is needed. Only use the increment clustering cannot improve the efficiency radically. It is hoped to enhance clustering efficiency for adopting parallel processing technology.

Parallel data mining technology involves parallel architecture and parallel algorithm. A typical parallel clustering at least contains the following three steps.

Partition: Partition the dataset into p sub-datasets and send them to each processor, where p is the number of processors presently used.

Computing: Execute clustering algorithm in every processor for its local dataset. The clustering algorithm running on each processor can be different.

Integrating: Integrate the local clustering results returned from each processor and form to an entire result.

A number of authors have worked in the area of parallel clustering. Partition clustering algorithm was paralleled with the shrink structure of VLSI [1]. SIMD Computing model based on super cube structure and parallel hierarchical clustering algorithm based on PRAM were described. [2], [3] A distributed clustering algorithm that adopted parallel genetic algorithm with thick granularity to obtain the optimum partition based on SIMD was depicted. [4] A parallel center clustering algorithm utilizing master-slave programming model based on PC COW was realized [5], which holding 80% usage of computing resource. Aimed at fixed architecture of static network and the limited communication ability, the running time of parallel algorithm was improved by wider bus system not more processors [6, 7]. A parallel K-means based on MPI and SPMD model [8] was realized, which using data parallel strategy potentially and achieved linear accelerate ratio approximately and linear retractility. This algorithm was applied into Beowulf COW; in it each node owns two processors [9]. The parallel realization of LBG and ELBG in commercial huge computer that composed of one speed Ethernet connect with 32 heterogeneous nodes was characterized, [10, 11] which adopted master-slave programming model and used PVM to communicate between nodes.

Above all reveal that the parallel architecture includes parallel computer, reconfigurable parallel processing system and COW system and so on.

II. Parallel Strategy

A. Data Parallel

Data parallel is also called data division. The task is to divide the data set reasonably and distribute the division data into processor and in turn each processor computes the allocated data parallel. There are two type data parallelisms. One is based on record and the other is attribute. The former scatters the records to each processor and the latter is to allocate the attribute sequences.

For each record, the probability of executing data mining algorithm is the same when distributed randomly. This made allocating the data equably is possible. In the real parallel data mining application, data partition based on record is usually adopted. If the attribute division is improper, the data relation will be broken and the mining accuracy is also tampered.

B. Task Parallel

Task parallel is that decomposing the entire solving procedure into several sub-procedures and dispatching them into different processor. It is also has two approaches to realize task parallel. One is based on divide-and conquer strategy. This divides the task first and assigns the sub-task to an appointed processor. The other one is based on task queue. It allocates the sub-task to present usable processor dynamically. For task parallel, all data are shared. The executing manner is either using multiple processors to finish the same task or using multiple processors to complete the different one. Which one is chosen depends on the detailed application and data structure. Whether which parallel manner is adopted, the load balance problem should be considered. If the data allocated unevenly, the load may be unbalanced.

C. Combination Data Parallel with Task Parallel

In many cases, there is needed to unite data parallel and task parallel together because any single one cannot solve the solution properly. In the data parallel mode, all processors execute the same task together, while in the task parallel mode different processor executes the different task. When combining data parallel with task parallel, an effective method is depicted follows.

After dividing the data reasonably and distributing sub-data into each processor, sub-task is assigned to an appointed processor according to the data characteristic allocated in each processor. When sub-task running on each processor is finished, integrates the results returned from sub-task. Here both data parallel and task parallel are adopted. The benefit is that data exchange is happened only before and after executing task. The task is not interrupted and the communication overhead is reduced.

III. K-MEANS PARALLEL ALGORITHM

The iteration of K-means algorithm mostly concentrates on [2] that one is distance computing between sample and cluster center, the other is sum and average operation when new cluster center is computed. The computational complexity of these two parts is $O(nkt)$ and $O(nt)$ respectively , where n is the total number of objects, k is the number of clusters, and t is the number of iterations. The task of distance computing will be increased with the accretion of dataset. Parallel technology can be used to fulfill these two iteration computings and also reduce the memory demand.

A. The Idea of Parallel Cluster

NOW，network of workstations, and PVM[1] are used to build the computing environment. The strategy is Master/slave and data parallel. Main program runs on the host, which takes charge of distributing dataset, adjusting load, gathering the cluster results after each iterative operation and producing new cluster center. The slave programs deal with the allocated data. Message transfer routine of PVM is used to realize data transfer among processors. Data distribution is self-adapted so as to load balance dynamically.

At the beginning, the master processor divides the sample and sends them accompany with cluster center to each slave one. Each slave processor receives the dataset with size of SN where N is the entire data size and P is the slave number and satisfies $SN < \frac{N}{P}$ usually. Each slave processor executes clustering operation for received data and returns the clustering results to the master. The master partitions a new sub-dataset and sends it to the slave. This is continued until there are no more data in the master and all the results are returned. So far an iterative process is completed. If the value difference between twice error-squared function is smaller than ε where $\varepsilon \le 10^{-6}$ then the algorithm is ended. Otherwise the master processor computes the new cluster center again and a new iterative procedure begins.

B. Dynamic Load Balance

For realizing data segment availability, which more the segment number more the communication and less dispose time of each processor, there is needed to consider segment number and communication tradeoff. The inherent characteristic of K-means and different disposal ability of processors are considered. If adopted static division data, data deflection may be produced and some processors will be in idle. The higher data deflection is that disposed data are near similar. This brings difference of processing data quantity and destroys load balance ultimately. So the available method is that after slave completed computing for assigned sub-dataset, it initiative applies for the next sub-dataset that has the same size from the master until there are no more data to be allocated. This balances the load better.

C. The Algorithm of Parallel K-means

The detailed description of parallel K-means algorithms with master and slave portions are figured out.

Master :

```
Read dataset from file;
Select initial centers;
do
{
    arr_current=0;
    recv_patch_count=0;
    for(i=0; i<nhost-1; i++)
    {
        for(j=arr_current; j<arr_current+SN; j++)
           // iteration for initial partition dataset
        {
            xs[j-arr_current]=x[j];
            ys[j-arr_current]=y[j];
        }
        arr_current=arr_current+SN;
        machine_no=i+1;
        pvm_initsend(PvmDataDefault);
           // send cluster center to each slave
        pvm_pkdouble(cx1,CN,1);
        pvm_pkdouble(cy1,CN,1);
        pvm_pkdouble(xs,SN,1);    // transmit sub-data to slave
        pvm_pkdouble(ys,SN,1);
        pvm_pkint(&machine_no,1,1);
        pvm_send(tid[machine_no],1);
    }
    compjc=1;
          // store the difference of error-squared functions
    while(recv_patch_count<patch_amount)
    {
        cc=pvm_recv(-1,2);
                          // receive cluster result from slave
            if (cc>0)
            {
              pvm_upkint(fs,SN,1);
              pvm_upkint(&pnr,1,1);
               pvm_upkdouble(sum_mean,k,1);
                pvm_upkint(&machine_no,1,1);
            }
            recv_patch_count++;
            for(i=0;i<SN;i++)
            {
                if(i+pnr*SN>=N)
                    break; // save the transferred result
                f[i+pnr*SN]=fs[i];//
            }
            // send the remaining data to the slave
          if(arr_current<N)
          {
               arr_current=devide_remain_dataset();
                           //split remain dataset
               pvm_initsend(PvmDataDefault);
                           // send the sub-dataset to the slave
               pvm_pkdouble(xs,SN,1);
               pvm_pkdouble(ys,SN,1);
               pvm_send(tid[machine_no],1);
          }
    }      // here iterative operation is completed once
     jc=compute_new_jc();
                      // compute the error square function
     compjc=fabs(jc1-jc);
     jc1=jc;
     jc=0;
     compute_new_centers(); // compute new cluster center
 }while(compjc>1e-6);
```

Slave:

```
ptid = pvm_parent();
sla_tid = pvm_mytid();
while(TRUE)
{
        cc = pvm_recv(ptid, 3);    // receive cluster center
        if (cc>0)
        {
            pvm_upkdouble(cx,CN,1);
            pvm_upkdouble(cy,CN,1);
        }
        for(i=0;i<CN;i++)      cf[i]=i+1;
          cc=pvm_recv(ptid,1);      // receive data
        if (cc>0)
            {
                pvm_upkdouble(x,SN,1);
                pvm_upkdouble(y,SN,1);
                pvm_upkint(&machine_no,1,1);
             }
            clusterdata();    // cluster operation
            sla_tid=pvm_mytid();
           pvm_initsend(PvmDataDefault); // return the result
           pvm_pkint(sum_mean, k ,1);
           pvm_pkint(fs,SN,1);
           pvm_pkint(&machine_no,1,1);
           pvm_send(ptid,2);
}
```

Ⅳ. Experiment Results

All experiments are executed on the COW system constituted by LINUX REDHAT 8.0 and PVM3.4.3, computers with Intel P4 1.7GHz、256MB RAM connected by 100M/10M Ethernet card.

The experimental data comes from a real medicine sales data. Before clustering, each medicine is not belonged to any cluster and the class number is zero. Two important attributes quantity and price are selected. Each kind of medicine sales data can be denoted as (X, Y, f), where X is sales quantity, Y is the price and f stands for sort mark. The results of serial and parallel K-means are almost the same. The clustering results of 100M quantity data are drawn in table I

TABLE I
CLUSTERING RESULTS

Type	Cluster center	Quantity
Unsalable	1.464871, 15566.371560	436
Common	2.427712, 3440.377419	5013
Salable	42.484051, 153.873550	94551

After clustering manipulation, all medicines are set to unsalable, common and salable class respectively. This accords with the real situation. Table II gives the running time iterative time and the accelerate ratio of serial compared with parallel K-means algorithm.

TABLE II
THE RESULTS COMPARED WITH SERIAL AND PARALLEL K-MEANS ALGORITHM

N D=2	Iterative time		Executing time (second)		Acceleration ratio
	Serial	Parallel	Serial	Parallel	
100k	15	14	1	15	0.067
200k	15	14	3	24	0.125
300k	15	14	5	39	0.128
400k	15	14	6	40	0.15
500k	15	14	8	55	0.145
600k	15	14	11	76	0.1447
700k	15	14	15	84	0.1785
800k	15	14	141	106	1.33
900k	15	14	190	102	1.863
1000k	15	14	212	110	1.927

There is no accelerate ratio when dataset scale N between 100K and 700K. This is because the communication time is larger than computing time. After N reaches 800K, computing time is larger than communication time, the real accelerate ratio is produced.

Ⅴ. Conclusion

A parallel K-means algorithm is put forward. For parallel manner data parallel strategy and Master/Slave programming model are adopted. For enhancing parallel K-means algorithm efficiently, dynamic load balance is also used. The experiments verify the correctness of proposed parallel K-means and also tell us presented parallel K-means has higher efficiency and general usage.

REFERENCES

[1] L., Jain, A, A VLSI systolic architecture for pattern clustering. IEEE Trans. Pattern Anal, Machine Intel, 7 (1), 80－89, 1985

[2] Li, X., Fang, Z, Parallel clustering algorithms. Parallel Compute,11, 275－290,1989

[3] Clark F.Olson: Parallel Algorithms for Hierarchical Clustering. Parallel Computing,21:1313-1325,1995N. K. Ratha, A. K. Jain, and M. J. Chung：Clustering using a coarse-grained parallel Genetic Algorithm: A Preliminary Study. Proceedings of the 1995 Computer Architectures for Machine Perception, 1995,pp. 331-338

[4] Bin Zhang, Meichun Hsu: Scale Up Center-Based Data Clustering Algorithms by Parallelism. HPL-2000-6 January,2000

[5] Horng-Reg Tsai, Shi-Jinn Horng: Parallel Clustering Algorithms on a Reconfigurable Array of Processors with Wider Bus Networks. IEEE, 0-8186-8227, Feb.1997

[6] Chin-Hsiung Wu, Shi-Jinn Horng, Yi-wen Chen: Designing scalable and efficient parallel clustering algorithms on arrays with reconfigurable optical buses. Image and Vision Computing 18(2000) 1033-1043

[7] Inderjit S.Dhillon, Dharmendra S.Modha: A Data-Clustering Algorithm On Distributed Memory Multiprocessors.

[8] Matthew C.Siegel: A Parallelization of the Batch K-means Clustering Algorithm. Dec 02,2002

[9] Giuseppe Patane, Marco Russo: Parallel Clustering on a Commodity Supercomputer. IEEE,0-7695-0619,April 2000

[10] Giuseppe Patane, Marco Russo: Distributed unsupervised learning using the multisoft machine. Information Sciences 143 181-196,2002

[11] Hazem M.Abbas, Mohamed M.Bayoumi: Parallel codebook design for vector quantization on a message passing MIMD architecture. Parallel Computing 28:1079-1093,2002

[12] Sanpawat Kantabutra: Parallel K-Means Clustering Algorithm on NOW. Sep. 1999.

[13] B.Boutsinas, T.Gnardellis: On distributing the clustering process. Pattern Recognition Letters 23 999-1008,2002

[14] D.Foti, D.Lipari, C.Pizzuti: Scalable Parallel Clustering for Data Mining on Multicomputers.