A New Load Balancing Scheme on Heterogeneous Database Cluster

GONG Weihua  WANG Yuanzhen

ABSTRACT In heterogeneous database cluster, the performance of load balancing is closely related to the computing capabilities of heterogeneous nodes and the different types of workloads. Thus, a method is introduced to evaluate the load status of nodes by the weighted load values with consideration of both the utilization of different resources and the workload types in a load balancer and an efficient and dynamic load balancing scheme is proposed for OLTP(online transaction processing) workloads to maximize the utilization of distributed resources and achieve better performance, which need not collect the feedback of load information from the lower nodes and effectively keeps from the data skew. The simulation results for OLTP services gained by TPC-C tool show that the dynamic weighted balancing policy leads to sub-linear throughput speedup and keeps the heterogeneous cluster well balanced.

KEYWORDS heterogeneous cluster; OLTP; load balancing

CLC NUMBER TP393

Introduction

Although the development of cluster has gained more and more advantages of low-cost, high-performance and high-availability computing, efficient resource usage such as CPU, memory and I/O is still a key to achieve better performance in cluster systems. Especially for heterogeneous high-throughput cluster, the differences in processor speeds, architectures and memory capacities can significantly influence system performance. Thus the purpose of load balancing is to reduce the execution time of applications and improve the utilization of the resources.

The issues of load balancing in cluster have been extensively studied in the literature in recent years, which include some important factors system architectures[1-4], algorithms[5-7], resource managements[8-15] and so on. The related works about them are introduced as follows.

Firstly, a new load balancing architecture that can deal with heterogeneous tasks was presented in Reference [1], and the study in Reference [2] gives a middle tier architecture based on dynamic agent system for heterogeneous cluster, which provides non-preemptive task scheduling dynamically. Xu et al.[3] propose a task management and scheduler system in cluster which is mainly for remote computing service, but the drawback is that to keep the balance of tasks there is need to know some parameters in advance, not transparent to users. Reference [4] evaluated the performance of a load balancing middleware which improves the scalability of distributed applications. Secondly, a static scheduling scheme, which only considers CPU heterogeneity to coordinate parallel jobs and local processes, was proposed in Reference [5], and Reference [6] presented a new I/O-aware load-balancing scheme to improve overall performance of a dis-
tributed system with a general and practical workload including I/O activities, whereas dynamic load balancing for cluster computing in Reference [7] was achieved by combining the advantages of loop scheduling strategies.

Additionally, many studies also address the problems of resource managements and data allocation. Some of them have primarily considered the resource of CPU, memory, and I/O, or a combination of them, for instance, Zhu et al. considered both CPU and I/O resource requirements, while Zhang et al. discussed CPU-memory (CM) based load balancing scheme.

Finally, different types of workloads and services are also closely related to the load balancing, the studies in References [6, 15] only discussed I/O intensive workloads, and a new I/O-aware load-balancing scheme known as weighted average load with preemptive migration (WAL-PM) was presented in Reference [17]. Chen et al. take the contents/services types into consideration for balancing workloads of scalable web server clusters, and Reference [16] considered load sharing policies for memory intensive workloads on heterogeneous cluster.

In this paper, we focus on mixture workloads of CPU intensive and I/O intensive applications just like OLTP workloads in heterogeneous database cluster, and present an efficient and dynamic load balancing scheme to achieve better performance, the major objective is to minimize the response time of each individual transaction and to maximize the system throughput of OLTP service by effectively using the distributed resources such as CPU and I/O.

1 Load architecture of heterogeneous database cluster

In this section we will describe the overall architecture of a load balancer in heterogeneous cluster. The heterogeneity in database cluster only refers to the variations of CPU powers, memory and I/O capacities, but the variations of operating systems, network interfaces among the nodes. Each node of the cluster runs an autonomous DBMS, interconnected by a fast Ethernet network, just as illustrated in Fig. 1.

Wu et al. [15] have showed a load management system for file access based on heterogeneous shared-disk clusters, while in this model the database cluster shares nothing. The load balancer is a middle tier architecture, which mainly consists of three components: meta data, scheduler and aggregation unit. The meta data unit of the middleware stores the data distribution rules in database cluster. And the scheduler component is responsible for assigning the client’s requests to execute on corresponding nodes according to the information stored in meta data, which mainly adopts two processing methods with partitioning operation and replicating operation. The partitioning operation is introduced if the service processing requests or data exceed the capacity of a single server node, while the replication service is commonly employed to improve the system availability and provide load sharing. The aggregation unit is mainly used to process the results returned from each node and to provide uniform and complete results for client requests, since partial results that come from multiple data partitions may need to be aggregated in this component before being delivered to external users. The three components are our kernel parts in the load balancer, and the framework can also keep the concurrent transactions consistent, but the issues of concurrency and coherency control protocol are orthogonal to our concerns.

In this architecture, the load balancer supports
standard API or interfaces such as ODBC 3.0 or JDBC to communicate with upper layer and bottom autonomous DBMS, and the tables distributed among the nodes are categorized into two types: replicated tables and partitioned tables. The replicated tables are used to execute read-only or infrequent update operations and the data are duplicated and distributed among all nodes while the partitioned tables are usually executed for update-intensive transactions and the data are horizontally distributed in nodes by hybrid-range partitioning method which combines the merits of the hash and range partitioning schemes. When the imbalanced workloads cause the performance decreasing, the balancer usually transfers some data from the overloaded nodes to other under-loaded nodes so as to share some workloads, at the same time the requests that need update the transferring data are joined in the queue waiting to be executed.

2 Dynamic load balancing scheme

This load balancing method is a little similar to the WAL-PM for I/O intensive workload and the weighted average load with remote execution (WAL-RE) for memory intensive workload, but they only consider the monolithic workload and homogeneous environment, whereas in a heterogeneous cluster system for mixed workloads, the load balancing scheme is more complex and variable. Hence we will propose a dynamic load balancing scheme, taking multiple resources and mixed workloads into consideration. The improvement of throughput is closely related to the resource utilization and the kind of workloads, in other words, the load index of each node in cluster is composed of composite usage of different resources including CPU, memory, and I/O, which are contributed to different workloads, and the compute capability of a node is a weight function of load index, the total weighted value can be calculated with

\[ W(i) = \lambda_1(i) \times W_{CPU}(i) + \lambda_2(i) \times W_{RAM}(i) + \lambda_3(i) \times W_{I/O}(i) \quad (1) \]

where \( \sum_{j=1}^{3} \lambda_j(i) \geq 1 \), and the parameter \( \lambda_j(i) \) is the dynamic percentage that the workload employs the type of resource denoted by \( j \) on node \( i \). The parameter \( \lambda_j(i) \) can be obtained from the following formula

\[ \lambda_j(i) = \frac{R_j(i)}{R} \quad (2) \]

where the numerator \( R_j(i) \) represents the number of requests, which is distributed on node \( i \) and consume the type of resource \( j \) on the node. And the denominator \( R \) is the total requests distributed among the cluster system. The above two parameters are tracked and calculated by the scheduler that pre-processes and analyzes the structure of requests so as to know where to execute and what resources to be used according to the meta data. The value of \( \lambda_j(i) \) changes dynamically as the number of the requests increases, while some requests maybe need to spend the multiple resources of node \( i \), they will be repeatedly counted in Eq. (2) for different \( \lambda_j(i) \). As a result, the accumulated sum of \( \lambda(i) \) may be larger than 1.

In Eq. (1), the other three variables such as \( W_{CPU}(i), W_{RAM}(i) \) and \( W_{I/O}(i) \) are the relative values compared with the base node which is equal to the average computing capacity of heterogeneous cluster and given by \( W = \phi(V, M, B) \), where \( W \) denotes the standard computing capacity of the base node, and the method for calculating \( V, M \) and \( B \) will be given later. The relative computing capacity of CPU is got from \( W_{CPU}(i) = \frac{V(i)}{\bar{V}}, \) where \( \bar{V} = \frac{\sum_{i=1}^{n} V(i)}{n}, \) and \( V(i) \) is the CPU speed measured by millions of instructions per second (MIPS). In the same way, we can get the relative value of memory by \( W_{RAM}(i) = \frac{M(i)}{\bar{M}}, \) where \( \bar{M} = \frac{\sum_{i=1}^{n} M(i)}{n}, \) and \( M(i) \) represents the available memory size (Mb). Likewise the I/O capacity is given by \( W_{I/O}(i) = \frac{B(i)}{\bar{B}}, \) where the value of \( \bar{B} \) is defined as \( \bar{B} = \frac{\sum_{i=1}^{n} B(i)}{n}, \) and \( B(i) \) is the average I/O access rate of disk on node \( i \). From Eqs. (1) and (2), we know that the weighted value of a node
represents the comprehensive computing capability for some workload in heterogeneous configurations, and the utilization of different resources is closely related to the types of the requests. Thus, the performance of a node is determined by both the computing capacity of different resources and the types of requests.

Before presenting this dynamic load balancing scheme, we suppose that every node has a load threshold value denoted by $\theta(i)$ which is determined by the weighted computing capability of different resources when the node achieves saturation point in some workload. If the value of dynamic weighted load, for example $W(i)$, exceeds the threshold value on node $i$, i.e. $W(i) > \theta(i)$, which shows that the node is overloaded, then part of the data distributed on the node should be transferred to other under-loaded nodes so as to alleviate the workload of the overloaded node correspondingly, and the load balancer will chose the maximal value in the set of $d(i)$, which is given by

$$d(i) = \begin{cases} \theta(i) - W(i), & \text{if } \theta(i) > W(i) \\ 0, & \text{else} \end{cases} \quad (i = 1, 2, \ldots, n)$$

The maximal $d(i)$ represents the node $i$ with the lightest workload in cluster, and part of the data on overloaded node should be transferred to the node $i$. When processing the requests parallel begins, the load balancer calculates the load information of all nodes dynamically, and loops to check the difference set continuously in every interval according to Eq. (3). At the same time the set is dynamically changed as the requests increase. When the value of $d(i)$ is equal to 0, it shows that the node is overloaded and need to share the workload with other under-loaded nodes.

In addition, the types of workloads also have much impact on the utilization of different resources. Previous studies such as Qin et al. have presented an I/O-aware load balancing scheme for I/O intensive applications. Now we will take the OLTP workloads for example to discuss this load balance scheme. The OLTP workloads consist of large volumes of update-intensive and small transactions which belong to typical CPU-I/O-intensive workloads. Therefore, the weighted load value of each node for OLTP service is mainly influenced by CPU and I/O resources, and then the value of $\lambda_3(i)$ is equal to 0 in Eq. (1), it means the derived formula is $W(i) = \lambda_1(i) \times W_{CPU}(i) + \lambda_3(i) \times W_{IO}(i)$. We use above derived formula to calculate value of $W(i)$, and the threshold value of $\theta(i)$ will be given in the experiments of next section.

The scheme that aim to improve the efficiency of heterogeneous cluster by the dynamic load balancing has some advantages. Firstly, our load balancing schemes not only focus on the resource utilization but also consider the types of workloads, we make full use of both CPU and I/O resources on different nodes to achieve dynamic balancing. Secondly, the load balancer only needs to know the configurations of the critical resources which does not change frequently such as CPU, memory and I/O in advance, and then to calculate the load information dynamically. Additionally, all requests are scheduled and distributed to the appropriate nodes processing through the balancer, the load balancer does not have to collect the feedback of load information from the lower nodes, it not only reduces network costs but also makes full use of heterogeneous resources efficiently. Lastly, the dynamic load balancing scheme does not incur data skew and effectively avoids the status of "hot" spots.

3 Experiments and performance analysis

To evaluate the performance of the load balancing scheme for OLTP services in heterogeneous cluster, we measure an improved version of the transaction processing council’s TPC-C benchmark, which is an OLTP benchmark that simulates an order-entry environment. The TPC-C testing is a mixture of read-only and update intensive transactions including five types of transactions: new-order (45%), payment (43%), order-status (4%), delivery (4%), stock-level (4%). The two former types of
transactions include many shortages and updating intensive transactions, and the others are mostly read-only transactions that mainly cost more CPU resource. Thus we can approximately consider that about 88% of the total workloads are I/O intensive transactions and only 12% of the workloads cost CPU resource in TPC-C testing.

In our experiments, the heterogeneous database cluster consists of 15 nodes. Each node has a combination of multiple resources such as CPU, memory, disks, run an autonomous SQL Server 2000 database, and is connected via a 100 Mb/s switched Ethernet. Additionally, the database scheme of the TPC-C benchmark consists of nine tables with various sizes and organizations and has an important feature of natural distribution of data, eight of the tables are well suited for partitioned distribution using hybrid-range strategy based on the primary key attributes, and only the item table can be easily replicated across all the nodes because it is a read-only table and has static data. The system architecture is illustrated in Fig. 1, and all the heterogeneous configurations of nodes and the average configuration of the base node are listed in Table 1.

Table 1 Hardware configurations of heterogeneous database cluster

| Type | Number | CPU   | Memory/Mb | I/O/(Mb/s) | \( \theta(i) \) |
|------|--------|-------|-----------|------------|---------------|
| I    | 4      | PIV   | 1.6 G     | 512        | 100           | 1.46          |
| II   | 1      | PIV   | 1.6 G     | 256        | 33            | 0.58          |
| III  | 4      | CIII  | 1.0 G     | 384        | 65            | 0.97          |
| IV   | 3      | PIV   | 2.0 G     | 128        | 33            | 0.65          |
| V    | 3      | AMD 800 M | 512     | 66         | 0.95          |
| Base node | V=I. 36 G | M=384   | B=66.27   | W=1         |

In the above table, the threshold values of different types of configurations are calculated according to the formula \( \theta(i) = 0.21 \times W_{CPU}(i) + 0.88 \times W_{IO}(i) \) when the node achieves saturated points, at this case the ratio of the workloads consuming different resources is fixed. For the different thresholds obtained above, we find that I/O resources make up the major contributions in simulating OLTP services. The relative computing capability of different configurations compared with base node is illustrated in Fig. 2. Obvioulsy, we observe that the first type is the fastest configuration, for the reason that the total sum of three kinds of resources is larger than the others. We will measure the maximal throughput on the fastest node and make it as the performance reference of throughput speedup. During the test, we find when the node can provide some fixed available memory for OLTP workloads, more memory size will have little help to improve the performance. So the heterogeneity of memory is not considered much for the OLTP simulation.

We use the dynamic load balancing scheme for heterogeneous cluster as stated in Section 2 to achieve load balancing. As previous studies have declared that speedup is one of the most important metrics to evaluate the performance of a balancing cluster system, which indicates how much faster a task can be run by increasing the degree of parallelism with more nodes, we adopt another improved speedup measurement called throughput speedup \((S(n))\) which is equal to the sum of throughput on n nodes divided by the throughput on the fastest node in the same interval, that is given by

\[
S(n) = \frac{\sum_{j=1}^{n} P(j)}{\max_{i} P(i)}
\]

where \( P(i) \) denotes the throughput of node \( i \). The throughput speedup also reflects the increase of the performance due to the increase of the number of nodes. Firstly, we measure the actual total throughput by TPC-C testing tool when the load balancer is at the cases of balanced and imbalanced schemes in heterogeneous database cluster as the heterogeneous nodes increasing, and then the throughput speedup is calcu-
lated according to the Eq. (4) and the corresponding curves are described in Fig. 3. Additionally, we also evaluate the throughput speed-up in the ideal homogeneous database cluster composed of base nodes. In order to compare more conveniently, we assume that the homogeneous cluster is composed of the same number of nodes as the heterogeneous cluster, and the workloads as well as data are evenly distributed in homogeneous cluster to achieve load balancing.

As shown in Fig. 3, the trends of throughput speedup in the homogeneous cluster are sub-linear increasing with the increase of the homogeneous nodes. In homogeneous cluster the network communication overheads also have some impact on the performance of system as more and more nodes are joined. Therefore the throughput speedup curve depicted in broken line is not ideally increasing linearly. However, in heterogeneous cluster the overall performance of load balancer manifests itself very well when adopting this dynamic load balancing scheme, and the throughput speedup is asymptotic to the sub-linear increasing trend of homogeneous cluster. To some extent, we can take the total throughput of homogeneous cluster as ideal simulated result even considering the influence of network communication overheads, thus the throughput speedup curve described with dotted real line in balanced heterogeneous cluster is nearly close to the ideal results of homogeneous cluster and the dynamic load balancing scheme achieves the optimal utilization of heterogeneous resources. But if there is no load balancing scheme in heterogeneous cluster, the load imbalance will lead to lower processing efficiency of overloaded nodes and worse throughput speedup just as illustrated in Fig. 3. Above all, the results of TPC-C testing show that this dynamic load balancing scheme in heterogeneous cluster not only maximizes the utilization of different resources but also scales up well.

4 Conclusions

In this paper we present a study on dynamic load balancing scheme for OLTP application which is a mixture of CPU intensive and I/O intensive workloads in heterogeneous database cluster. Since the resource utilization such as CPU, memory and I/O is closely related to the system performance, we develop a middleware system to balance the workloads on heterogeneous nodes with different processing capabilities. In order to evaluate the influence of all kinds of resources in different nodes under different types of workloads, we introduce the weighted load values for different nodes that consider both the utilization of different resources and the workload types, and propose an efficient and dynamic load balancing scheme to maximize the resource utilization and to keep from the data skew. When the distribution of workloads is imbalanced among heterogeneous nodes, the load balancer will transfer part of the hot data from the overloaded nodes to other under-loaded nodes, which alleviates the workloads correspondingly. The advantages of improving the efficiency of heterogeneous cluster by this load balancing scheme have also been stated in Section 2. The simulation results for OLTP services testing with TPC-C tool show that the dynamic weighted balancing policy keeps the heterogeneous cluster well balanced and avoids the hot spots effectively. Furthermore, we also observe that the balanced heterogeneous cluster system has the sub-linear throughput speedup and the good scale-up.

REFERENCES

[1] Nishikawa H, Steenkise P (1993) A general archi-
tecture for load balancing in a distributed-memory environment [C]. The 13th International Conference on Distributed Computing Systems, Pittsburgh, Pennsylvania, USA

[2] Rajagopalan A, Hariri S (2000) An agent based dynamic load balancing system [C]. The International Workshop on Autonomous Decentralized Systems, Chengdu, China

[3] Xu Fei, Huang Dazheng, Yang Guangwen (2003) TMSS; a task management and scheduler system in cluster for remote computing service [C]. The 17th International Conference on Advanced Information Networking and Applications, Xi’an, China

[4] Balasubramanjan J, Schmidt D C, Dowdy L, et al. (2004) Evaluating the performance of middleware load balancing strategies [C]. The 8th IEEE International Enterprise Distributed Object Computing Conference, Monterey, California

[5] Du Xing, Zhang Xiaodong (1997) Coordinating parallel processes on networks of workstations [J]. Journal of Parallel and Distributed Computing, 46(2):125-135

[6] Qin Xiao, Jiang Hong, Zhu Yifeng, et al. (2003) A dynamic load balancing scheme for I/O-intensive applications in distributed systems [C]. The 2003 International Conference on Parallel Processing Workshops, Kaohsiung, Taiwan

[7] Balasubramanjam M, Barker K, Banicescu I, et al. (2004) A novel dynamic load balancing library for cluster computing [C]. The ISPDC/HeteroPar ’04, Cork, Ireland

[8] Xiao Li, Chen Songqing, Zhang Xiaodong (2002) Dynamic cluster resource allocations for jobs with known and unknown memory demands [J]. IEEE Transactions on Parallel and Distributed Systems, 13(3):223-240

[9] Abawajy J H, Dandamudi S P (2003) Scheduling parallel jobs with CPU and I/O resource requirements in cluster computing systems [C]. The 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation, Orlando, FL, USA

[10] Zhang Xiaodong, Qu Yanxia, Xiao Li (2000) Improving distributed workload performance by sharing both CPU and memory resource [C]. The 20th International Conference on Distributed Computing Systems, Taipei, Taiwan

[11] Lee L, Schaeuermann P, Vingralek R (2000) File assignment in parallel I/O systems with minimal variance of service time [J]. IEEE Transactions on Computers, 49(2):127-140

[12] Chen Zungshi, Chen Kuolian (2004) Balancing workload based on content types for scalable web server clusters [C]. The 18th International Conference on Advanced Information Networking and Application, Fukuoka, Japan

[13] Thanalapati T, Dandamudi S (2001) An efficient adaptive scheduling scheme for distributed memory multicomputers [J]. IEEE Transactions on Parallel and Distributed Systems, 12(7):758-768

[14] Zhu Haican, Yang Tao, Zheng Qi, et al. (1998) Adaptive load sharing for clustered digital library servers [C]. The 7th International Symposium on High Performance Distributed Computing, Chicago, Illinois, USA

[15] Smirni E, Reed D A (1997) Workload characterization of input/output intensive parallel applications [C]. The 9th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Malo, France

[16] Xiao Li, Zhang Xiaodong, Qu Yanxia (2000) Effective load sharing on heterogeneous networks of workstations [C]. The 14th International Parallel and Distributed Processing Symposium, Cancun, Mexico

[17] Qin Xiao, Jiang Hong, Zhu Yifeng, et al. (2003) Boosting performance for I/O-intensive workload by preemptive job migration in a cluster system [C]. The 15th Symposium on Computer Architecture and High Performance Computing, Sao Paulo, Brazil

[18] Transaction Processing Performance Council. TPC benchmark C standard specification revision 5.3 [OL], http://www.tpc.org/tpcc/spec/tpcc_current.pdf

[19] Ghaderifarizadeh S, Dewitt J D (1990) Hybrid-range partitioning strategy: a new declustering strategy for multiprocessor database machines [C]. The 15th VLDB Conference, Brisbane, Australia

[20] Wu Changxun, Burns R (2004) Achieving performance consistency in heterogeneous clusters [C]. The 13th IEEE International Symposium on High Performance Distributed Computing, Hawaii, USA