International Journal of Database Theory and Application
Vol.7, No.4 (2014), pp. 37-46
http://dx.doi.org/10.14257/ijdta.2014.7.4.04

Automatic Data Distribution in Large-scale OLTP Applications

*Xiaoyan Wang[1, 2, 3], Xu Fan[1, 3], Jinchuan Chen[1] and Xiaoyong Du[1, 3]**

[1]Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), MOE, Beijing, 100872, China
[2]School of Information and Electrical Engineering, Ludong University, Yantai, 264025, China
[3]School of Information, Renmin University of China, Beijing, 100872, China
{wxy, b8flowerfire, jcchen, duyong}@ruc.edu.cn

Abstract

With the dramatic increase of data volume, automatic data distribution has been one of the key techniques and intractable problem for distributed systems. This work summarizes the problem of data distribution and abstracts it as a general triangle model called DaWN. Based on data and workload analysis, it presents a novel strategy called ADDS for automatic data distribution in OLTP applications. According to the results of a series of experiments over TPC-C datasets and transactions, the proposed approach shows effective improvements.

Keywords: *automatic data distribution; data placement; data partitioning; OLTP*

1. Introduction

As data volumes growing, the basic strategy of big data management is to push computing to data rather than moving large amounts of them [1]. It is a big issue to distribute data carefully in order to obtain high performance in terms of availability and scalability. Data distribution is used to partition a series of data into disjoint data partitions, and allocate them into different data nodes. Although there have been some works on data distribution, most of them are manual on some special scenario or mainly focus on processing static data. Besides, with the trend of automation, it is a fairly challenging issue to have an automatic data distribution solution which could have high system throughput with low processing cost.

In this work, we summarize and abstract the problem of data distribution as a general triangle model called DaWN, and propose a novel strategy called ADDS for automatic data distribution in OLTP applications. ADDS is based on data and workload schema analysis, and could automatically distribute both static and incremental data. We also conduct a series of experiments on TPC-C to show both the efficiency and effectiveness of our proposed strategy.

The remainder of this paper is organized as follows: Section 2 presents the problem of data distribution and its triangle model called DaWN, Section 3 proposes a novel strategy called ADDS for automatic data distribution, Section 4 reports our observations and analysis in the experiments, while Section 5 presents related works in data distribution, and finally, Section 6 briefly summarizes this work and discusses works in the future.

2. Problem Description

To illustrate the concerned problem, consider the following motivating scenario: Given a data center with many workloads, a data center administrator needs to build a database server

ISSN: 2005-4270 IJDTA

configuration that consists of many nodes. A critical question is how to distribute the static and incremental data to suitable nodes so that workloads could access them locally.

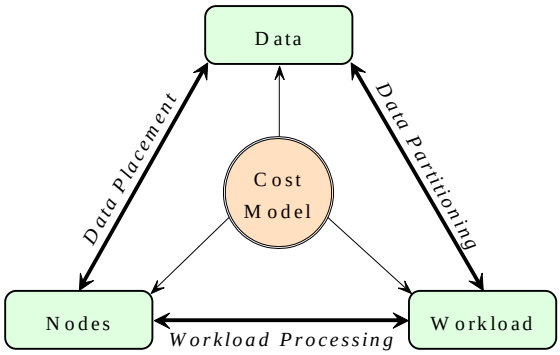

Figure 1. Dawn the Triangle Model of Data Distribution

Data distribution is a process of consolidation, during which data, workload and node perform data partitioning, data placement and workload processing under the constraints of a certain cost model. From the view of the whole system, the reasonableness of data distribution not only affects local access efficiency, but also restricts global queries and transaction processing efficiency. Based on studies in this field, we propose a triangle model called DaWN (stands for **Da**ta, **W**orkload and **N**odes) for data distribution, which is as shown in Figure 1. It uses data, workload and nodes as the vertexes of the triangle separately, and encodes data partitioning, data placement and workload processing as the sides to describe the relationships between each other. As different systems have different business requirements, DaWN uses cost model as the central hub to obtain performance goals under certain resource limitations.

Definition 1. Data Distribution. Given a database $\boldsymbol{D} = \{R_1, R_2, \ldots, R_{|\boldsymbol{D}|}\}$, a workload of transactions $\boldsymbol{W} = \{T_1, T_2, \ldots, T_{|\boldsymbol{W}|}\}$, a storage bound $\boldsymbol{NS} = \{B_1, B_2, \ldots, B_{|N|}\}$ of the working node set $\boldsymbol{N} = \{N_1, N_2, \ldots, N_{|N|}\}$, find an distribution solution for $\boldsymbol{D}$ such that the size of data distributes fits in each B_z, and the overall cost of $\boldsymbol{W}$ is minimized. Here,

- R_x ($1 \le x \le |\mathbf{D}|$) is a predefined table schema in database D,
- T_y ($1 \le y \le |\mathbf{W}|$) is a predefined transaction with an access frequency AF_y in workload W,
- B_z ($1 \le z \le |\mathbf{S}|$) is the storage bound of data node N_z.

The basic distribution strategy is considered to maximize the usage of the obtained information mentioned above in the whole process. What need to be mentioned is that part of them might not be available for an application. It is supposed that the more useful information properly used for data distribution, the better the effect is. Still, a trade-off for different solution goals in a given application could not be avoided.

3. An Automatic Data Distribution Solution

According to the research and analysis in [2], most of the transactions in current massive OLTP applications are usually fast running and short-lived, which need to be quickly responded. Meanwhile, most of them only touch a small subset of data with strong semantic dependency, and do not need to take full table scanned or execute large distributed joins.

Furthermore, in practical applications, they are usually repetitive executed as predefined in transaction templates or stored procedures.

3.1. Solution Overview

For a well-designed OLTP application, the basic information of the system, including data schema, workload pattern and their utility features, and the number of data nodes, is supposed to be obtainable before data distribution. We focus on the applications in which the query pattern is steady and could be obtained with access frequencies for all of the transactions within it. Also, we assume that the data schema is well defined.

The input of a distribution process would include a database, a representative workload, and the number of nodes. In order to have a proper automatic data distribution solution, it is then fairly important to maximize the utilization of potential information implicit in these inputs. Then the strategy is expected to output a distribution solution that could minimize the overall expected cost of running the workload. For the workloads in an OLTP application, the expected cost is directly related to the number of distributed transactions, which will be used in this paper as the measure for overall performance. The whole architecture of the designed ADDS (**A**utomatic **D**ata **D**istribution **S**olution) is as shown in Figure 2. It contains two phases, *i.e.*, data partitioning and data placement.

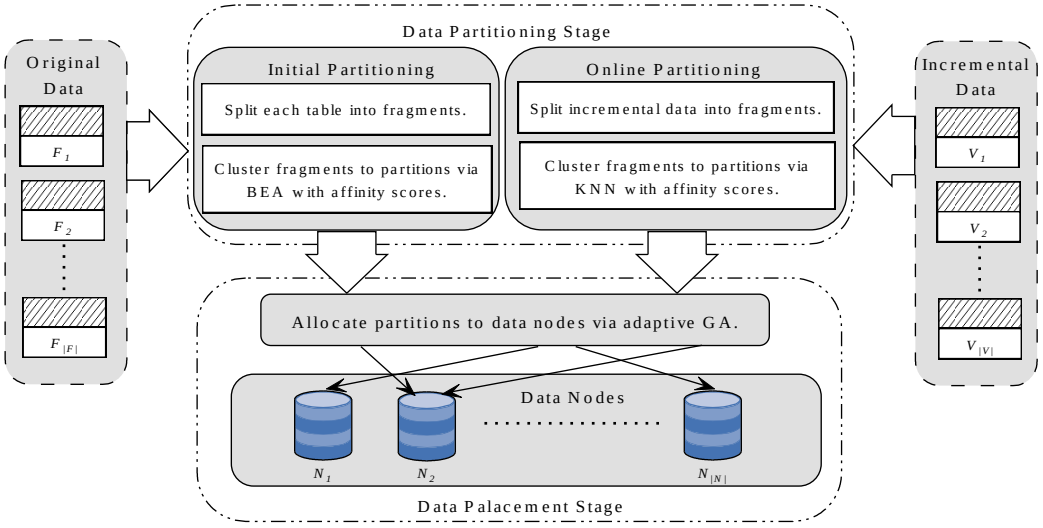

Figure 2. ADDS Architecture

3.2. Data Partitioning

This phrase has 2 parts: initial partitioning for original data and online partitioning for incremental data. In order to have a much more suitable data partitioning method, we use the partitioning strategy completed in [3], which combines advantages of both horizontal and vertical partitioning to have a better effect at a proper fragment granularity level. In initial partitioning, it firstly splits initial data in each table into small fragments, computes the affinity scores between every pair of fragments based on a given workload, and then, clusters them into partitions using BEA [4]. Online partitioning will be invoked periodically by incremental data. It will group newly incremental data into fragments through computing the affinity scores between them and the old ones, and put them into the closest partitions based on kNN [5]. The whole procedure of data partitioning is as shown in Algorithm 1.

ALGORITHM 1 Data Partitioning Procedure

Input: B_P is the partition size limitation, ***F*** *is the dataset in terms of a set of fragments, DU is the usage matrix,* ***D'*** *is all append data,* $\boldsymbol{P}=\{P_1, P_2, ..., P_{|P|}\}$, *and k* is the parameter of kNN.
Output: $\boldsymbol{P_{ini}}$ *is the initial partitioning solution, and* $\boldsymbol{P_{onl}}$ *is the online partitioning solution.*

Procedure: DPartP

//Initial Partitioning Procedure
1. Assign $U=\Phi$, $U'=\boldsymbol{F}$, $i=1$;
2. for (i=1; i<=|P|; i++) {
3. Build the Correlation Matrix *AA* of *U'* and Transform *AA* with BEA
4. Partition *AA into 2 disjoint subsets*: $AA=AA_1 \cup AA_2$, $U=U \cup AA$
5. if $P_i <$ | B_P |, choose a subset of *U* and assign it to *U'*;
6. }
7. Storage the set of P_i as the initial partitioning solution $\boldsymbol{P_{ini}}$ and go to data placement procedure
//Online Partitioning Procedure
8. for (each time incremental data ***D'*** invokes online partitioning) {
9. split ***D'*** into a set of fragments $\boldsymbol{V}=\{V_1,V_2,.....,V_{|V|}\}$
10. for (each V_i in $\boldsymbol{V}$) {
11. Compute the affinity scores between V_i and all of fragment F_j inside the existing partitions
12. find top-k nearest neighbors, say $F_1, F_2,..., F_k$ of V_i
13. choose the partition, say P_l, which contains the largest portion of these *k* nearest neighbors
14. if $P_l <$ | B_P |, put V_i into P_l
15. }
16. Storage the set of P_l as the online partitioning solution $\boldsymbol{P_{onl}}$ and go to data placement procedure
17. }

3.3. Data Placement

In data placement phase, it allocates these partitions generated from both initial and online partitioning via a genetic algorithm until each partition has been assigned to a single working node. The design goal of placement algorithm is to have the data accessed by transactions with high priority, variability, access frequency, and correlation in the same node. As an NP-hard problem [6], this work uses an adaptive genetic algorithm, which is an adaptive search technique based on the principles of natural selection and 'survival of the fittest', to solve it.

(1) **Coding:** In data placement, chromosome is the mapping code of partitions to nodes. Assuming that each partition could be only allocated to one node while each node could allocate multiple partitions, this work takes coding method based on partitions. An individual solution *s* must have each partitions allocated into a corresponding data

node. The chromosome of each *s* has |P| gene segments. The code of the *i* th segment is the mapping of partition P_i. For $\forall N_i \in N$, its code is *(i-1)*. For example, when *|P|=|N|=10*, the chromosome code is {0 1 2 3 4 5 6 7 8 9} for the data placement solution: mapping of partition P_i to N_i.

(2) **Fitness:** The design of fitness function *FF* is related to procedure goals. To achieve requirements in Formula (3) of Work [5], it is designed as shown in Formula (1):

$$FF(x) = \frac{f_{\max}(x) - f(x)}{f_{\max}(x) - f_{\min}(x)} \quad (1)$$

Here, *f(x)* could be obtained from Formula (3) of Work [5], while $f_{max}(x)$ and $f_{min}(x)$ separately present the maximum and minimum value that each individual obtains from Formula (3) of Work [5] in the current generation.

(3) **Selection:** It uses expected value model to dealing with the selection operation. This method is simple while could remain individuals with good fitness into the next generation. Assuming the current group scale is *m*, it firstly calculate the expected value of current generation as $\overline{FF_i} = \frac{1}{m}\sum_{i=1}^{m} FF_i$, and then calculate the expected value of each individual survived in next generation as $\overline{RR_i} = \frac{FF_i}{\overline{FF_i}}$. Then, it could get an integer RR_i which is rounded to the nearest integer by $\overline{RR_i}$. If RR_i = 0, the individual is eliminated. Else, it will have RR_i copies of it in the next generation.

(4) **Crossover:** It is used to have good genes inherited into the next generation and generate more complex and new individuals. According to [7], it uses an adaptive crossover factor PP_c in Formula (2) to balance the good gene maintaining with the high speed searching. Generally, PP_{c1} is set to be 0.9 while PP_{c2} is 0.6.

$$PP_c = \begin{cases} PP_{c1} - \dfrac{(PP_{c1} - PP_{c2})(FF_{\max} - FF_m)}{FF_{\max} - FF_{avg}} & FF_m \geq FF_{avg} \\ PP_{c1} & FF_m < FF_{avg} \end{cases} \quad (2)$$

(5) **Mutation:** It is used to improve the local search ability of GA and maintain the population diversity. According to [7], it uses an adaptive mutation factor PP_m in Formula (3) to balance both of the population diversity and the search randomness. Generally, PP_{m1} is set to be 0.1 while PP_{m2} is 0.001.

$$PP_m = \begin{cases} PP_{m1} - \dfrac{(PP_{m1} - PP_{m2})(FF_{\max} - FF_m)}{FF_{\max} - FF_{avg}} & FF_m \geq FF_{avg} \\ PP_{m1} & FF_m < FF_{avg} \end{cases} \quad (3)$$

(6) **Termination:** Repeat the above operations until this processing has been completed.

As shown in Figure 3, the whole data placement process is divided into 3 parts: initial strategy generation, strategy optimization and eventual strategy generation. It is supposed to have better performance of convergence speed and better stability in the global optimum

result with the same generations. All partitions could then be allocated into data node set by methods mentioned above recursively to finish the placement phase.

4. Experiment Analysis

In this section, we would like to report the results and analysis in conducted experiments.

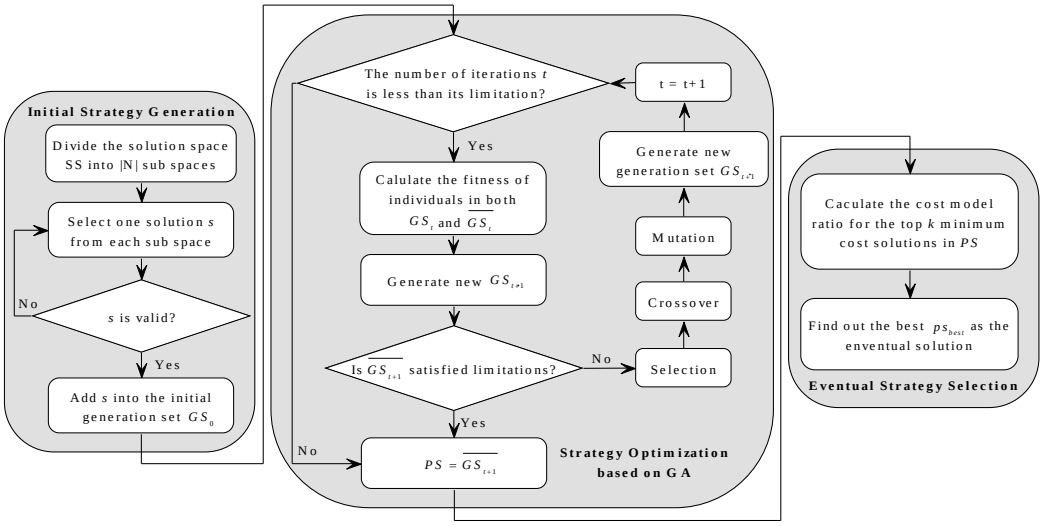

Figure 3. The Process of Data Placement

4.1. Environment and Configuration

In the experiment, we have setup an environment with 8 data nodes, each of which is equipped with a processor of 2.4G * 6 Cores * 2, 48GB memories, 300G SAS disk, and Redhat-release-5server-5.5.0.2 operation system. All of these data are stored in MySQL 5.4.3. The dataset and transactions are generated from TPC-C 5.11[8], which contains 9 tables, 92 attributes, 8 primary keys, 9 foreign keys with 5 different kinds of transactions to illustrate a complete environment where a population of terminal operators executes transactions against a database. According to the hardware condition, the scale factor (the number of Warehouse) of the dataset is set to be 100 at the build-up stage, and 50 incremental during each step until it gets to 500. The number of transactions is generated accordingly, which contains 5 different transactions mixed together with the relative ratios in TPC-C.

4.2. Experiment Results and Analysis

To illustrate the effect in automatic data distribution, we have used 3 different strategies for comparison in the experiments, including Hashing [12], Round-Robin[12] and ADDS described in Section 3. Their effects could be presented in several aspects, including the number of distributed transactions, the workload on partitions, and the average data correlation in portioning. In order to get a steady result and have a much more reliable result analysis, each strategy with relative dataset and workload in a specific distribution strategy has been executed for 5 times completely in the same physical environment.

4.2.1. Number of Distributed Transactions: Figure 5 shows the ratio of distributed transactions generated in the executing process of different distribution strategies. As shown

in Figure 5, ADDS shows less number of distributed transactions than both of Hashing and Round-Robin, with 28.45% reducing in average.

Figure 6 shows the variation trends of distributed transactions in each strategy executing. Hashing and Round-Robin shows the similar features when processing the same dataset and workload. With the increment of data and workload, each of them shows upward trend, while ADDS grows slowly.

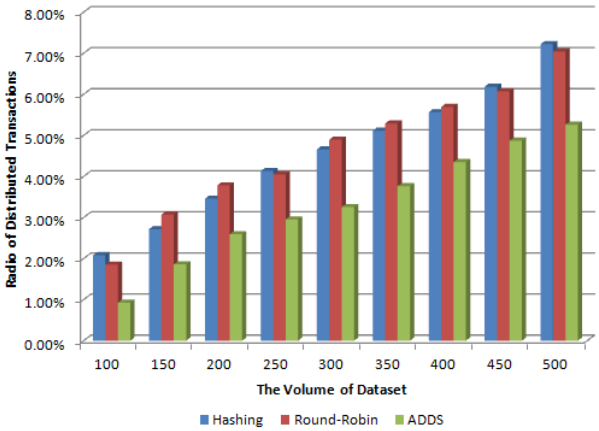

Figure 5. Ratio of Distributed Transactions

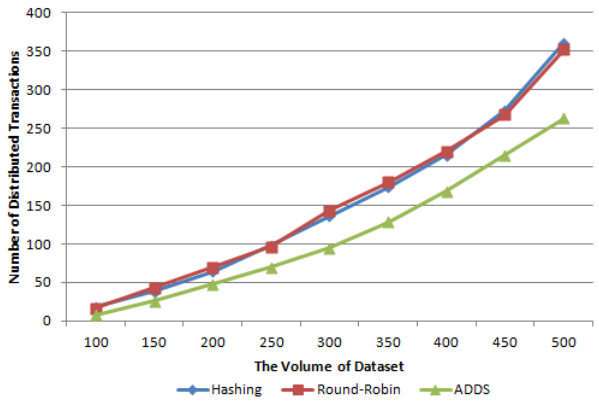

Figure 6. Number of Distributed Transactions

As shown in Figure 5 and 6, Hashing and Round-Robin partitioning has more numbers of distributed transactions at each stage than ADDS does. This is because ADDS take data dependency into consideration, which combined the semantic information in data schema and workload pattern. The result shows that, distribute more relative data would reduce the number of distributed transactions and then improve the system performance to a certain degree.

As indicated in Figure 5 and 6, the performance of ADDS will descend slowly (the number of distributed transactions increases). This is because the structure of existing partitions will be affected by the newly incremental data. In our settings, the volume of the initial database will be doubled and the clustering structure of the original partitions will be greatly changed. In real applications, this big change implies we may have to adjust these partitions, which exceeds the scope of this paper and will be addressed in our future works.

4.2.2. Workload on Partitions: As shown in Figure 7, each working node during the execution of Hashing and Round-Robin needs to process more transactions than ADDS does, thus ADDS shows a slow upward trend with 10.42% less than them in average. This is because a distributed transaction will be served by multiple working nodes. There are fewer distributed transactions in ADDS, which directly reduce the number of transactions per node.

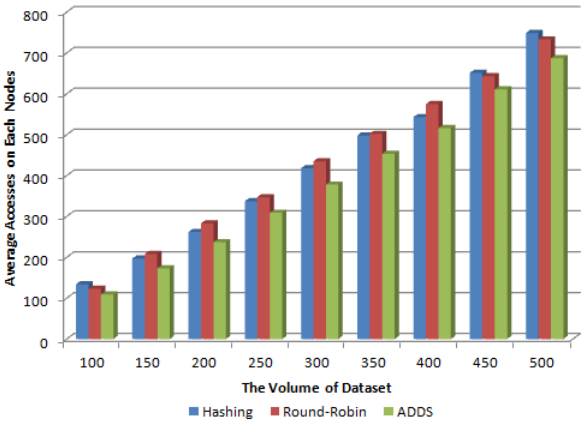

Figure 7. Workload on Partitions

4.2.3. Average Data Correlation: As shown in Figure 8, it is clear that, for ADDS, the fragments inside each partition are closer than Hashing and Round-robin. Meanwhile, both Hashing and Round-Robin shows almost the same correlation features when processing the same dataset and workload. The result shows that, clustering more relative data in a data node would increase the correlation of fragments in the same partitions, which would contributes to the reduction of the number of distributed transactions and then improve the system performance to a certain degree.

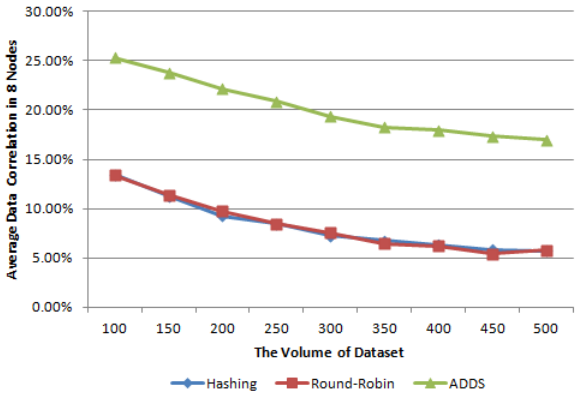

Figure 8. Average Correlation in Each Node

5. Related Works

In the fields of Data Management, data distribution is one of the most famous technologies [9] for platform scalability. Traditional partitioning solutions are hashing, round-robin, *etc.*, This kind of distribution results in low efficiency and high cost during dealing requests from users, especially in OLTP applications, which need to have high performance with low cost.

In recent years, there have emerged some new research works in data distribution. As considering tuples and relationships among them as a super graph and uses Metis [10] to partition it, Schism[11] is much more suitable for the applications with fixed datasets and workloads. As a refinement, work [2] uses LNS [12] to partition and allocate data. If there is incremental data added into the system, both of them need to be recalculated and partitioned, and may lead to re-placement or migration of data, which would cost too much.

Recently, research works in [13, 14] propose their solutions for distribute incremental data. In the view of one-size-fits-all, work [13] gives a one dement ion algorithm, which mixed horizontal and vertical partitioning to handle online requirement in [13]. But it is mainly focus on OLAP applications. Work [14] has applied the partitioning and placement technology into a specific scientific computing environment, which has the feature of workflow. Both of them are not suitable for large-scale OLTP applications as they do not consider much on transaction execution performance.

6. Conclusion

As data amount increasing rapidly and workload requests variously, new challenges have emerged in the field of big data. Based on the analysis of data distribution problem in distributed systems, this work abstracts it as a triangle model called DaWN and proposes an architecture called ADDS for automatic data distribution in large-scale OLTP applications. According to the experiment results, ADDS shows nice overall performance and could reduce the communication cost among nodes.

Also, there are many aspects to be improved: DaWN model needs to be instantiation in different kinds of big data applications, while ADDS strategy need to be refined and adjusted based on the real world requirements. Many other problems should also be taken into account, including various workload balancing, replication utilization, *et al*. These problems are planned to be studied and resolved in the following works.

Acknowledgements

This work is supported by State Key Laboratory of Software Development Environment Open Fund under Grant No.SKLSDE-2012KF-09, and the National Science Foundation of China under Grant No. 61003086. The authors would also like to thank Sa Shi-Xuan Research Center of Big Data Management and Analytics for its supports.

References

[1] Forrester Research Inc., “In-Database Analytics: The Heart Of The Predictive Enterprise”, Forrester Whitepaper (2009)
[2] A. Pavlo, C. Curino and S. Zdonik, “Skew-aware Automatic Database Partitioning in Shared Nothing Parallel OLTP Systems”, Proceedings of the 2012 ACM SIGMOD,Scottsdale, Usa, May 20-24, 2012.
[3] X. Wang, J. Chen, X. Du and X. Fan, “Research on automatic partitioning of incremental data in parallel OLTP systems”, Journal of Frontiers of Computer Science and Technology, vol.7, (2013)iss. 9, pp. 800-810.
[4] W. McCormick, P. Schweitzer and T. White. Problem decomposition and data reorganization by a clustering technique. Operations Research, vol. 20, iss. 5 , (1972), pp. 993-1009.
[5] D. Coomans and D. Massart, “Alternative k-nearest neighbor rules in supervised pattern recognition: Part 1. k-Nearest neighbor classification by using alternative voting rules”, Analytica Chimica Acta, vol. 136, (1982), pp. 15-27.
[6] D. Sacca and G. Wiederhold, “Database partitioning in a cluster of processors”, ACM Transactions on Database Systems, vol. 10, iss. 1, (1985), pp. 29-56.
[7] M. Srinvivas and L. M. Patnaik, “Adaptive probabilities of crossover and mutation in genetic algorithms”, IEEE Transactions on SMC Part B: Cybernetics , vol. 28, iss. 5, (1998), pp. 629-639.
[8] TPC Benchmark™ C, (2012) http://www.tpc.org/tpcc/

[9] R. Buyya, C. Yeo, S. Venugopal, J.Broberg *et al.*, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, vol. 25, iss. 6 , (2009), pp. 599-616.
[10] Metis, (2013) http://glaros.dtc.umn.edu/gkhome/views/metis/index.html
[11] C. Curino, E. Jones, Y. Zhang and S. Madden, "Schism: a Workload-Driven Approach to Database Replication and Partitioning", Proceedings of the VLDB Endowment Singapore; September (2010), pp. 13-17.
[12] E. Danna and L. Perron, "Structured vs. unstructured large neighborhood search: A case study on job-shop scheduling problems with earliness and tardiness costs", Principles and Practice of Constraint Programming, vol. 2833, (2003), pp. 817-821.
[13] A. Jindal and J. Dittrich, "Relax and Let the Database Do the Partitioning Online", Proceedings of BIRTE'11, Seattle, USA, September 2, 2011.
[14] M. L. Gistau1, R. Akbarinia, E. Pacitti, F. Porto *et al.*, "Dynamic Workload-Based Partitioning for Large-Scale Databases", Proceedings of DEXA', Vienna, Australia, vol.12**,** September 3-6, 2012.

Authors

Xiaoyan Wang, is a Ph.D. candidate at Renmin University of China. She received her B.S. degree in Computer Science and Technology from Central South University in 2003 and M.S. degree in Computer Software and Theory from Shandong University in 2006. Her research interests include big data management, database system and intelligent information retrieval.

Xu Fan, is a master candidate at School of Information, Renmin University of China. He received his B.S degree in Information Security from University of Electronic science and Technology of China in 2012. His research interests include multi-core processing and on-line analytical processing (OLAP), etc.

Jinchuan Chen, is an Assistant Professor at the Key Laboratory of Data Engineering and Knowledge Engineering, MOE. He received his B.S degree from Beijing Normal University in 2001, and his M.S. degree from Chinese Academy of Sciences in 2004. He then obtained his PhD from HKPolyU in 2009. His research interests include uncertain data clearing and unstructured data management.

Xiaoyong Du, is a professor and Ph.D. supervisor at Renmin University of China. He received his B.S. degree from Hangzhou University, M.S. degree from Renmin University of China and Ph.D. degree from Nagoya Institute of Technology. His research interests include database system, intelligent information retrieval, and big data analysis.

