INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 4 ISSUE 2 (2018) PAGES 3156 - 3162
RECEIVED : 05.03.2018 PUBLISHED : 10.04.2018

April 10, 2018

EFFICIENT AGGREGATION AND NON-BLOCKING INSTANCE ON LARGE DOMAINS

Ms.B.Keerthi, M.Tech.IT.,Student,
Prof. M.Nandhini, M.Tech., Assistant Professor,
Prof. R. Srinivasan, M.E, (Ph.D.), Professor and Head,
Department of Information Technology,
P.S.V College of Engg. & Tech., Krishnagiri - 635108.

ABSTRACT

Business Intelligence (BI) is recognized as one of the most important IT applications in the coming big data era. In recent years, non-uniform memory access (NUMA) has become the de-facto architecture of multiprocessors on the new generation of enterprise servers. Such new architecture brings new challenges to optimization techniques on traditional operators in BI. Aggregation, for example, is one of the basic building blocks of BI, while its processing performance with existing hash-based algorithms scales poorly in terms of the number of cores under NUMA architecture. In this paper, we provide new solutions to tackle the problem of parallel hash based aggregation, especially targeting at domains of extremely large cardinality. We propose a NUMA-aware radix partitioning (NaRP) method which divides the original huge relation table into subsets, without invoking expensive remote memory access between nodes of the cores. We also present a new efficient aggregation algorithm (EAA), to aggregate the partitioned data in parallel with low cache coherence miss and locking costs. Theoretical analysis as well as empirical study on an IBM X5 server prove that our proposals are at least two times faster than existing method.

INTRODUCTION

Aggregation, such as sum, average, count, etc., is a traditional and fundamental building block in BI systems. The efficiency of aggregation is crucial to systems with critical response time requirement. In practice, aggregation operator is usually invoked on a large dataset containing millionsor even billions of records.It usually works with a group-by operator, which divides the dataset into groups based on specified attributes. The cardinality of the result groups could be extremely large. In our real application with Shanghai Stock Exchange, for example, the group-by operator is executed on daily transactions based on the user accounts, involving aggregations over a huge domain with millions of stock trading participants in China.

The contributions of this paper are listed as follows:

1) We proposed a new NUMA-affined radix partitioning algorithm which follows the characteristics of NUMA architectures and achieves both intra-node and inter-node load balance.

2) We proposed an efficient aggregation algorithm (EAA) which effectively avoids the cache coherence miss and reduces the locking cost through elaborate scheduling based on range locking.

3) Theoretical analysis and extensive experimental studies show the efficiency and effectiveness of our newly proposed methods.

EXISTING SYSTEM

Business Intelligence (BI) is becoming more and more prevailing as it helps enterprises to make valuable decisions based on their archived data sets. Existing systems affiliating such BI applications, including HANA, Greenplum and Aster, are emerging at an amazing pace.The architecture of modern multi-core server has evolved so quickly in recent years to deal with the well-known frequency wall, power wall and memory wall Such architecture changes lead to shift from traditional memory access methods, i.e. uniform memory access (UMA) in the chip multiprocessor systems (CMP) and symmetric multiprocessor systems (SMP). While its processing performance with existing hash-based algorithms scales poorly in terms of the number of cores under NUMA architecture. Due to sensing imprecision and environmental interferences, the sensor readings are usually noisy.The cache capacity miss problem stated above can be nicely solved by hash-partitioning the input data on the group-by attributes such that the total size of the hash buckets corresponding to all keys in a single partition is small enough to reside in cache. However, the hash partitioning process becomes the performance issue if the number of partitions exceeds the number of cachelines and TLB entries.

PROPOSED SYSTEM

Here provide new solutions to tackle the problem of parallel hash based aggregation, especially targeting at domains of extremely large cardinality. Here proposed a new NUMA-affined radix partitioning algorithm(NaRP) which follows the characteristics of NUMA architectures and achieves both intra-node and inter-node load balance

Here proposed an efficient aggregation algorithm (EAA) which effectively avoids the cache coherence miss and reduces the locking cost through elaborate scheduling based on range locking.Range Based Aggregation (RBA), forces the partitions with overlapping ranges to be aggregated by a single instance. As all partitions with the same range will be processed by a single parallel instance, locking is not necessary. Unfortunately, this method introduces expensive remote memory accesses in NUMA architectures. Node independent aggregation (NIA), allocates each NUMA node a private hash table and lets each parallel instance aggregate on the local private hash table, which will be merged together finally. As the ranges within each NUMA node do not overlap, providing that an entire partition will be aggregated by a single instance, the cache coherence miss is avoided and locking is not needed. However, the overhead 0of merging private hash tables is very expensive for large group-by cardinality.

Algorithm 1

PAS for dynamic workload balance

Input :
Sensor Node **S1,S2,**Clusters**…..SnC1,C2,…..**Basestation**Cn B1,B2,**Begin**…Bn**for**.**each **S1.S2,**tuple**….C1,C2,S**for**….**each**Cn C2,C2…..**tuple**B1,CnB2,,** do**…**Load**Bn** Balance. It is worthwhile toemphasize other benefits of PAS to data. First, and do not need to communicate with

each other, or to write to the same part of the memory pool. Consequently, they do not interfere with each other in terms both workflow and efficiency

Second, if more instances take over the workloads, the partitioning process is independent of each other, after the computation of the insert cursors. Third, the cost of calculating insert counts in for pre-allocation is very efficient,as SIMD instructions are employed and sequential memory access of the original data could leverage hardware cache pre-fetching.

In the passes except the first one, as the repartitioning of each input partition from previous round is independent, we use task queue model to improve the load balancing between instances.In the task queue model, each input partition is modeled as a job and all input partitions are kept in a job queue. An instance picks up the first partition in the job queue, finishes the partition before his next job pickup at the

ARCHITECTURE

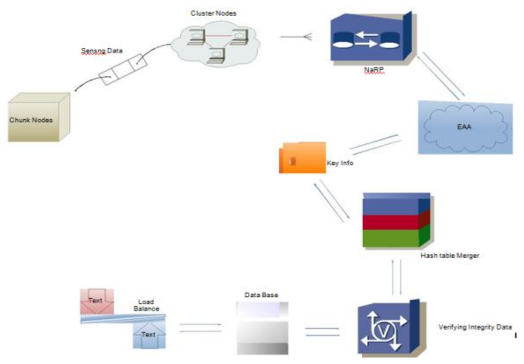

FIGURE 1:ARCHITECTURE

Efficient Aggregation

Our EAA only uses a single global hash table, since the merging of private hash tables under our problem setting is too expensive to afford.But we have new optimization techniques to minimize the cache coherence misses and locking cost.EAA applies range locking, in which there are global mute locks, namely range locks, shared by all the parallel instances.The range lock corresponds to the range that the partition on any node has Before any instance aggregates a partition into the global hash table.It should first obtain the range lock to prevent other parallel instances from aggregating the partitions with the same range.

SYSTEM MODULE

1. Chunk Nodes and Data Pruning
2. NUMA Aware Radix Partioning
3. Efficient Aggregation

4. Verifying Integrity Data

5. PAS Dynamic Workload Balance

1. Chunk Nodes and Data Pruning

The extensive number of research work in this area has appeared in the literature. Due to the limited energy budget available at sensor nodes, the primary issue is how to develop energy-efficient techniques to reduce communication and energy costs in the networks. Approximate-based data aggregation techniques have also been proposed. The idea is to tradeoff some data quality for improved energy efficiency.

2. NUMA Aware Radix Portioning

New optimizations based on the characteristics of NUMA architectures maximize the parallelism in partitioning to balance intra-core workload and intra-node workload, which results in a dramatic reduction on partitioning latency.Local parallel instances running on a NUMA node are responsible for partitioning the local data and maintaining the intermediate partitioning results in local memory. There is no data sharing or communication across NUMA nodes. the overall structure of the new parallel partitioning method follows the standard radix partitioning algorithm

3. Efficient Aggregation

The first NUMA-aware parallel aggregation algorithm, to maximize the utility of multiple cores under NUMA architectures. Here propose a three aggregations Shared aggregation (SA) allows parallel instances to share a global hash table, and parallel updates are serialized by executing locks. Independent aggregation (IA) allocates a private hash table for each parallel instance to store its aggregation results individually, and finally merges all the private hash tables to complete the aggregation. Hybrid aggregation (HA) assigns a private hash table of specific size (based on the cache size) to each parallel instance, to maintain the partial aggregation results. When overflow happens to the private hash table, the additional aggregation results are directly flushed into the global hash table.

4. Verifying Integrity Data

To verify the scalability of the algorithms, we sample the transaction table under different sampling rate to generate variant datasets with desirable group-by cardinalities To perform in-network query processing, a routing tree is often formed among sensor nodes and the base station. , we measure the total amount of data transmission as the performance metrics. Notice that, response time is another important metrics to evaluate query processing algorithms in wireless sensor networks

5. PAS Dynamic Workload Balance

The PAS strategy and task queue model can be easily extended to inter-node load balancing by allowing the parallel instances in the NUMA node to help other NUMA nodes after they have finished the processing of all the local data. To minimize the remote memory access and the synchronization overhead, cross-node workload migration is enabled, only when one NUMA node has finished its radix partition for all the passes. The remote memory access penalties during inter-node balancing can be further reduced by the following strategies. First, let the idle NUMA nodes prefer to take over the workload from the nearest nodes (with one hops NUMA distance). Second, before the partitions the assigned data chunk into the remote pre-allocated partitions, it first feeds the partitions into local cache through efficient sequential scan such that the subsequent writes to those partitions can avoid expensive remote random memory access.

CONCLUSION

Here discuss the in-memory aggregation computation of big data with large group-by cardinality on modern servers under NUMA architectures. We point out the limitations of the state-of-the-art solutions on poor performances and scalability, due to the expensive costs of cache coherence miss and locking in NUMA architectures. We thus contribute a new NUMA-aware radix partitioning algorithm to improve the data partitioning efficiency, by optimizing the operations based on the characteristics of NUMA and including new load balancing strategies. We also design a novel efficient parallel aggregation algorithm, featuring in avoiding the cache coherence miss and minimizing the locking cost. We provide theoretical analysis on the performance of these proposals. The experiment results confirm that our new aggregation algorithm is at least two times faster than other state-of the- art algorithms by testing on real and synthetic datasets on .

We believe that the key ideas of avoiding cache coherence miss and synchronization in this paper can be extended to processing other database operations as well as many real-time analysis applications in NUMA architectures. Online aggregation of fast data stream, for example, is one of the possible extensions. By applying our implementation strategies appropriately, the efficiency of the existing methods could be dramatically improved.

FUTURE ENHANCEMENTS

The aggregation of data sets received from clusters other than to be merged with base station in order to compute the finally. A model-driven approach was proposed in to balance the confidence of the query answer against the communication cost in the network. Moreover, continuous top-k queries for sensor networks have been studied. In addition, a distributed threshold join algorithm has been developed for top-k queries. These studies, considering no uncertain data, have a different focus from our study.

REFERENCES

[1] Garter. (2010). Gartner taps predictive analytics as next big business intelligence trend [Online]. Available: http://www.enterpriseappstoday. com/business-intelligence /gartner-taps-predictiveanalytics- as- next-big-business-intelligence-trend.html

[2] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach. San Mateo, CA, USA: Morgan Kaufmann, 2011.

[3] P. A. Boncz, M. L. Kersten, and S. Manegold, “Breaking the memory wall in MonetDB,” Commun. ACM, vol. 51, no. 12, pp. 77–85, Dec. 2008.

[4] J. Cieslewicz and K. Ross, “Adaptive aggregation on chip multiprocessors,” in Proc. 33rd Int. Conf. Very Large Data Bases, 2007, pp. 339–350.

[5] R. Epstein, “Techniques for processing of aggregates in relational database systems,” Dept. Comput. Sci., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/ERL, 1979.

[6] G. Graefe, “Query evaluation techniques for large databases,” ACM Comput. Surv., vol. 25, no. 2, pp. 73–169, 1993.

[7] D. DeWitt, S. Ghandeharizadeh, D. Schneider, A. Bricker, H. Hsiao, and R. Rasmussen, “The gamma database machine project,” IEEE Trans. Knowl. Data Eng., vol. 2, no. 1, pp. 44–62, Mar. 1990.

[8] D. Bitton, H. Boral, D. DeWitt, and W. Wilkinson, “Parallel algorithms for the execution of relational database operations,” ACM Trans. Database Syst., vol. 8, no. 3, pp. 324–353, 1983.

[9] A. Shatdal and J. Naughton, “Adaptive parallel aggregation algorithms,” ACM SIGMOD Record, vol. 24, no. 2, pp. 104–114, 1995.

[10] J. Cieslewicz, K. Ross, K. Satsumi, and Y. Ye, “Automatic contention detection and amelioration for data-intensive operations,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 483–494.

[11] Y. Ye, K. Ross, and N. Vesdapunt, “Scalable aggregation on multicore processors,” in Proc. 7th Int. Workshop Data Manage. New Hardware, 2011, pp. 1–9.

[12] P. Boncz, S. Manegold, and M. L. Kersten, “Database architecture optimized for the new bottleneck: Memory access,” in Proc. VLDB Conf., 1999, pp. 54–65.

[13] C. Kim, T. Kaldewey, V. Lee, E. Sedlar, A. Nguyen, N. Satish, J. Chhugani, A. Di Blas, and P. Dubey, “Sort vs. hash revisited: Fast join implementation on modern multi-core CPUs,” Proc. VLDB Endowment, vol. 2, no. 2, pp. 1378–1389, 2009.

[14] J. Cieslewicz and K. A. Ross, “Data partitioning on chip multiprocessors,” in Proc. 4th Int. Workshop Data Manage. New Hardware, 2008, pp. 25–34.

[15] C. Balkesen, J. Teubner, G. Alonso, and M. Ozsu, “Main-memory hash joins on multi-core CPUs: Tuning to the underlying hardware,” in Proc. IEEE Int. Conf. Data Eng., 2013, pp. 362–373.

[16] Intel, "An introduction to the intel quickpath interconnect," ser. Document Number: 320412-001US, Jan. 2009.

[17] D. Molka, D. Hackenberg, R. Schone, and M. Muller, "Memory performance and cache coherency effects on an intel Nehalem multiprocessor system," in Proc. 18th Int. Conf. Parallel Archit. Compilation Techn., 2009, pp. 261–270.

[18] Intel, "Intel xeon processor e7- 8800/4800/2800 product families," ser. Document Number: 325120-001, Apr. 2011.

[19] J. Torrellas, H. Lam, and J. Hennessy, "False sharing and spatial locality in multiprocessor caches," IEEE Trans. Comput., vol. 43, no. 6, pp. 651–663, Jun. 1994.

[20] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes, "Samplingbased estimation of the number of distinct values of an attribute," in Proc. 21th Int. Conf. Very Large Data Bases, 1995, vol. 95, pp. 311–322