Accuracy of Aggregation in Peer-to-Peer DBMSs

Norvald H. Ryeng and Kjetil Nørvåg

Department of Computer and Information Science,
Norwegian University of Science and Technology
Sem Sælands vei 7–9, NO–7491 Trondheim, NORWAY
{ryeng,noervaag}@idi.ntnu.no

Abstract. Peer-to-peer routing mechanisms are resilient to churn in the overlay network layer. A major challenge for peer-to-peer database management systems is to provide similar robustness in the data and query processing layer. In this paper we in particular study aggregation queries, and present a new approach to increasing accuracy of such queries.

1 Introduction

A key feature of current peer-to-peer routing mechanisms is resilience to *churn*, the effect of nodes constantly joining and parting from the network. A major challenge in peer-to-peer database management systems is to provide similar robustness in the data and query processing layer. The failure rate of a large, distributed system is such that the system cannot expect all nodes to be accessible at all times, so waiting for disconnected nodes to reconnect is not generally an option. Instead, when nodes fail, query processing has to be based on partial data.

The typical method for doing aggregation in peer-to-peer systems is using a reduction tree. All nodes aggregate over their local data and merge the local partial aggregate with the partial aggregates of all their children. In this way, the result of the aggregation operation propagates up the tree, resulting in the complete result at the root node.

The problem with this approach, is that failure of internal nodes causes loss of data from all nodes below it in the hierarchy. Existing work propose to use replication of the aggregation process to counter this effect. In this paper we study this problem and describe a less costly technique to increase accuracy of aggregation queries.

The organization of the rest of the paper is as follows. In Section 2 we give an overview of related work. In Section 3 we present the general idea. In Section 4 we describe experiments and results. Finally, in Section 5, we conclude and outline issues for further work.

2 Related Work

Distributed hash tables (DHTs) are the underlying technology of several peer-to-peer systems. Popular implementations are Kademlia [1], Chord [2], CAN [3] and Pastry [4].

The PIER query processor [5, 6] and aggregation systems such as Astrolabe [7], SDIMS [8] and TAG [9] allow for efficient aggregation queries, using hierarchical algorithms. The Cougar Project [10] also uses a hierarchical algorithm, but utilizes a hybrid push-pull technique differing from that discussed in this paper.

While most current peer-to-peer systems create reduction trees based on routing paths, an algorithm for creating trees in DHTs is presented in [11], which also discusses how these trees can be used for aggregation and broadcast.

3 Accuracy of Aggregation

The impact of one node failure in a reduction tree depends on the structure of the tree. If the tree is narrow and tall, one node failure results in the loss of data from many nodes. In the pathological case of only one child per node, the average data loss caused by one node failure is 50%. This is reduced by increasing node degree. In the extreme case, the tree consists of only two levels, and one node failure results only in the loss of data from this node. The data loss caused by the failure of a high-level node is documented by Li et al. [11].

The current approach to fighting data loss (as suggested by, e.g., [5] and [11]), is replication. This is easily done by creating several independent reduction trees, replicating the whole aggregation process. Replication is prohibitively costly for large systems, and other approaches should be followed if possible. We propose that more attention is paid to other parameters, especially the degree of nodes.

The tree generated by existing algorithms can be estimated by considering the height of the tree to be the maximum length of the routing path, i.e., the maximum number of hops when doing a lookup. Assuming that all nodes have the same degree and that the tree is completely balanced, the degree can be calculated. The connection between number of nodes, N, degree, k, and height h of the tree is given by $N = \frac{k^{h+1}-1}{k-1}$.

Based on numbers from Stoica et al. [2], we estimate the node degree of a reduction tree based on a Chord network of 10,000 nodes to be approximately 6. Similarly, based on Ratnasamy et al. [3], we esteimate the degree of a reduction tree based on a 4-dimensinal CAN of 130,000 nodes to be 10.

4 Experiments

We simulated a DHT network of 10,000 nodes, storing 100,000 tuples, where we did hierarchical aggregation using different node degrees, with and without replication. The aggregation functions used are the standard SQL functions *sum*, *count*, *avg*, *min* and *max*. 10% of the nodes fail during query processing.

The achieved result, r, is compared to an ideal result, r_i, which is the result we would have got without node failures, and the accuracy of the results are calculated as a distrance metric: $d_{count}(r, r_i) = d_{sum}(r, r_i) = d_{avg}(r, r_i) = \frac{r-r_i}{r_i}$. We also define $d_{min}(r, r_i) = d_{max}(r, r_i) = \frac{r-r_i}{|D_{value}|}$, where D_{value} is the domain of the value attribute.

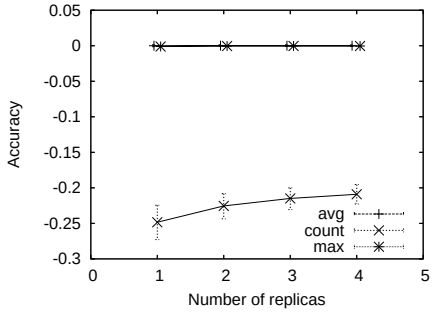

Fig. 1. Accuracy of aggregation functions with different number of replicas.

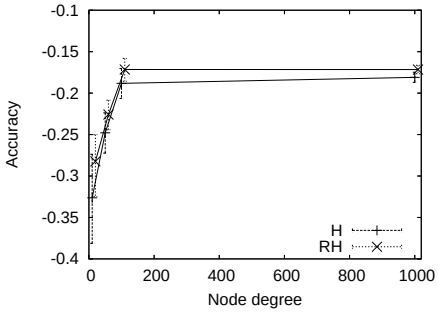

Fig. 2. Accuracy of the count function with different node degrees.

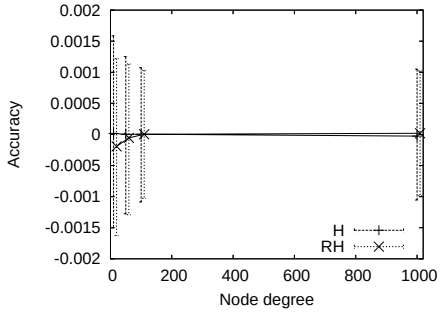

Fig. 3. Accuracy of the avg function with different node degrees.

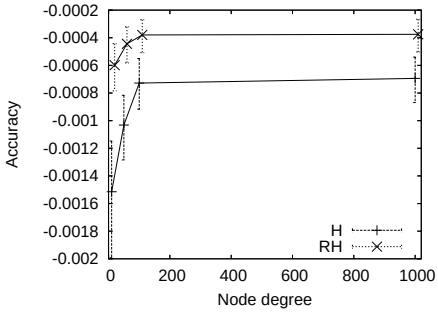

Fig. 4. Accuracy of the max function with different node degrees.

4.1 Results

In all our experiments, the results for *sum* and *min* were very close to those of *count* and *max*, respectively, so these were excluded from the figures.

From Figure 1 we see that the *avg* and *max* functions prove to be quite accurate to start with, whereas the *count* function show that some method to fight data loss is needed. In the rest of the experiments, only single replication is used, so the results compared are from simple hierarchical aggregation (H) and replicated hierarchical aggregation (RH).

Figures 2–4 show the results of varying the degree of nodes in the reduction tree from 10 to 1,000 (using steps 10; 50; 100; 1,000). We can see that the accuracy of *count* queries climb quite steeply from 10 to 100, i.e., from 0.1% to 1% of the number of nodes in the system, but that accuracy does not increase much beyond this number. The RH algorithm performs better than the H algorithm, but for low node degrees, there is more to gain by increasing the degree than by replicating.

We also see that there is little to gain by increasing the degree when computing *avg* queries. The distribution becomes somewhat denser, but not much.

When computing the maximum value, the node degree is important, but there is more to get from simple replication. For the parameters used in our simulations, it is always better to replicate than to increase the degree of internal nodes.

If we compare the results to the estimated node degrees of trees based on routing paths in Chord and CAN given in Section 3, we see that the simulation results indicate that trees based on current DHT implementations are too narrow, and that accuracy could be increased by generating broader trees.

5 Conclusion and Future Work

Our experiments show that varying the node degree in some cases may be more efficient in increasing accuracy than the costly replication, and also that these two methods may be combined to increase accuracy further. The simulations also indicate that there is much to gain from increasing the node degree from that of current implementations.

Several open problems remain. The query processor should be able to use statistics to predict which algorithm and which parameters would suit the query best. This could be combined with a requested level of accuracy to find the most efficient aggregation method to achieve the requested accuracy. Similar studies should also be done on other relational operations.

References

1. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: IPTPS. (2002)
2. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM. (2001)
3. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: SIGCOMM. (2001)
4. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware. (2001)
5. Huebsch, R., Chun, B.N., Hellerstein, J.M., Loo, B.T., Maniatis, P., Roscoe, T., Shenker, S., Stoica, I., Yumerefendi, A.R.: The architecture of PIER: An internet-scale query processor. In: CIDR. (2005)
6. Huebsch, R., Hellerstein, J.M., Lanham, N., Loo, B.T., Shenker, S., Stoica, I.: Querying the internet with PIER. In: VLDB. (2003)
7. Renesse, R.V., Birman, K.P., Vogels, W.: Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. ACM Trans. Comput. Syst. **21** (2003) 164–206
8. Yalagandula, P., Dahlin, M.: A scalable distributed information management system. In: SIGCOMM. (2004)
9. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: A Tiny AGgregation service for ad-hoc sensor networks. SIGOPS Oper. Syst. Rev. **36** (2002) 131–146
10. Yao, Y., Gehrke, J.: Query processing in sensor networks. In: CIDR. (2003)
11. Li, J., Sollins, K., Lim, D.Y.: Implementing aggregation and broadcast over distributed hash tables. SIGCOMM Comput. Commun. Rev. **35** (2005) 81–92