Delay Optimization in Quorum Consensus

Xuemin Lin

Abstract. The management of replicated data in distributed database systems is a classic problem with great practical importance. Quorum consensus is one of the popular methods, combined with eager replication, for managing replicated data. In this paper we investigate the problems of delay-optimal quorum consensus. Firstly, we show that the problem of minimizing the total delay (or mean delay) restricted to a ring can be solved in a constant time in contrast to the existing approximation results. Secondly, we show that the problem of minimizing the total delay (or mean delay) is NP-hard. Thirdly, we present an approximate algorithm with an approximate ratio 2; and the approximate algorithm can guarantee the exact solutions for some specific network topology, such as trees and meshes. Finally, we present an improvement on the existing algorithm to solve the problem of minimizing the maximal delay; this reduces the time complexity from \(O(n^3 \log n)\) to \(O(n^3)\) where \(n\) is the number of nodes.

Key Words. Quorum consensus, Replicated data management, Optimizations.

1. Introduction. Replicated data management in distributed databases is a classic problem with great practical importance. Distributed data warehouses and data marts contain a huge amount of replicated data distributed among a number of sites. Therefore, in recent developments [3], [4], [6], [10] of the area there is always a trade-off among system efficiency, data availability, data freshness, and data consistency. Two replicated data management methods are available in the literature: eager and lazy. Eager replication management gives data consistency and the highest data freshness. However, it suffers from system efficiency due to an application of a 2-phase commit protocol [19]. On the other hand, lazy replication management provides high system efficiency but does not necessarily provide data freshness. Moreover, pure lazy replication management does not generally guarantee data consistency. Recent research results [3] reveal that it is best if these two methods can be combined.

To achieve a high system efficiency, “quorum consensus” is often adopted in eager replication. In this paper we investigate the quorum consensus method. A quorum consensus algorithm is based on the design of a “coterie” (to be formally defined in Section 2). In a quorum consensus algorithm, a data processing (read or write) operation can proceed only if permission is granted by a group (quorum—an element in the coterie) of the data copies (sites) over the network. Permission from all copies from any one of the quorums in a coterie is sufficient for an operation to proceed, and to ensure data consistency.

1 Part of this work was done when the author took study leave in the SEEM Department at the Chinese University of Hong Kong (CUHK).
2 School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia. lxue@cse.unsw.edu.au.

Recent developments in quorum consensus are mainly focused on (1) minimizing the total communication costs for processing a given set of transactions, and (2) minimizing the number of remote sites to be communicated while assembling a quorum. A number of quorum consensus protocols [5], [9], [11], [13]–[17], [20], [21] have been developed for these purposes.

Note that in quorum consensus, since messages are sent (possibly by the multicast mechanism [18]) to the multiple nodes in a quorum in order to ensure consistency of the operations, the delays by passing messages through a long distance communication channel in a wide area network can create a bottleneck [19], [8] in the response time. In this paper we investigate the problems of minimizing the “average” (or total) delay and minimizing the “maximal” delay.

These two problems were first investigated in [8]. It provides several algorithms for the problem of minimizing the maximal delay with respect to special classes of network topologies, such as rings, trees, and clustered graphs, while [22] provides a $O(n^3 \log n)$ algorithm to solve the problem generally. Both papers also investigated the problem of minimizing the average (or total) delay. Reference [8] provides an approximate algorithm with approximation ratio 1.25 for rings with uniform links and uniform nodes and provides an exact algorithm for trees, while [22] presents a way to reduce the mean delay when minimizing the maximum delay. A branch-and-bound algorithm is developed in [12] to minimize the mean delay. The experimental results demonstrated that the branch-and-bound algorithm is quite effective for moderate-sized networks, though it runs in exponential time in the worst case. The complexity of minimizing mean (total) delay in general has been left open.

In this paper we firstly show that there is a constant time algorithm that gives an exact solution to the problem of minimizing the average (total) delay for rings with uniform links and nodes in contrast to the approximation result in [8]. Secondly, we show that the problem of minimizing the average (or total) delays is NP-hard. Thirdly, we provide an approximate algorithm with an approximation ratio 2 in general. Note that the average-delay minimization problem, which we study in the paper, is more general than the problem in [8], where each node takes only a unit weight. While the NP-hardness we show in this paper is restricted to the case where each node takes a unit weight, the approximation ratio of our algorithm covers the general case where each node can take an arbitrary weight. Moreover, we can show that our approximate algorithm can guarantee exact solutions for the popular network topologies, such as meshes [19] and trees. These are the principal contributions of the paper. Finally, we show that the complexity of the algorithm in [22], for minimizing the maximum delay, may be reduced to $O(n^3)$ from $O(n^3 \log n)$ where $n$ is the number of nodes.

Note that in [1] and [18] the file allocation has been studied in an on-line environment with the assumption that each read is processed by reading one copy and each write has to be propagated to each copy—read one and write all policy. The duplicated data management, discussed in this paper, assumes that a file allocation is given, and investigates read/write policies instead of the policy of read one and write all. Therefore, the results and techniques in [1] and [18] are not applicable.

The rest of the paper is organized as follows. In the second section we provide the background knowledge, and precisely define the problems. In the third section we present our results for the problem of minimizing the average (or total) delays. The fourth section