MobiQual: QoS-aware Query and Update Load Shedding in Mobile CQ Systems

Sk. Mahaboob basha
M.Tech (CSE) Student
Dept of CSE
QISCET, India
Skmahboob.basha@gmail.com

R. Lakshmi Tulasi
Professor& HOD
Dept of CSE
QISCET, India
ganta.tulasi @gmail.com

Abstract—**Position updates and query reevaluations are two predominant, costly components of processing location-based, continual queries (CQs) in mobile systems. To obtain high-quality query results, the query processor usually demands receiving frequent position updates from the mobile nodes. However, processing frequent updates often causes the query processor to become overloaded, under which updates must be dropped randomly, bringing down the quality of query results, negating the benefits of frequent position updates. The design of MobiQual highlights three important features. (1) Different amounts of query and update load shedding are applied to different groups of queries and mobile nodes, respectively. (2) The overall freshness and accuracy of the query results are maximized with individualized QoS specifications. (3) MobiQual dynamically adapts, with a minimal overhead, to changing load conditions and available resources. We show that, through a careful combination of update and query load shedding, the MobiQual approach leads to much higher freshness and accuracy in the query results in all cases, compared to existing approaches.**

Keywords— Load shedding, Query, MobiQual, QOS.

I. INTRODUCTION

The proliferation of mobile devices and advances in wireless communications are creating an increasing interest in rich, value-added location-based services[LBSs], which are expected to form an important part of the future computing environments that will seamlessly integrate into our lives [1]. A recent example from the industry is the Google Ride Finder [2] service, which provides mobile users with the capability to employ CQs to monitor nearby taxi services. Other Examples include resource management, such as transportation services, fleet management, mobile games, and battlefield coordination.

A key challenge for LBSs is a scalable location monitoring system capable of handling large number of mobile nodes and processing complex queries over their positions. Although several mobile continual query (CQ) systems have been proposed to handle long-running location monitoring tasks in a scalable manner [3], [4], the focus of these works is primarily on efficient indexing and query processing techniques, not on accuracy or freshness of the query results. To produce high- quality query results, the query processor usually demands receiving frequent position updates from the mobile nodes. However, receiving and processing frequent updates often causes the query processor to become overloaded.

Accuracy or inaccuracy is defined based on the amount of mobile node position errors found in the query results at the time of query re-evaluation. It mainly depends on the frequency of position updates received from the mobile nodes. we can also use a higher level concept to measure accuracy, such as the amount of containment errors found in the query results, including both false positives (inclusion errors) and false negatives (exclusion errors), by using position update errors for accuracy measure will provide higher level of precision. This is primarily because by utilizing the amount of node position errors as the accuracy measure, one can easily bound the inaccuracy by a threshold-based position reporting scheme [5], [6].

Freshness or staleness refers to the age of the query results since the last query reevaluation. It is dependent on the frequency of query re-evaluations performed at the server. As mobile nodes continue to move, there are further deviations in mobile node positions after the last query reevaluation. However, such deviations are not attributed to inaccuracy. Hence, freshness can be seen as a metric capturing the post query reevaluation deviations in mobile node positions. It is important to note that higher freshness does not necessarily imply higher accuracy and vice versa.

Both accuracy and freshness are similar to timeliness and completeness respectively in the web information monitoring domain. The CQ server must reevaluate the continual queries more frequently, requiring more computing resources in order to provide Fresh query results . Similarly, to attain more accurate query results, the CQ server must receive and process position updates from the mobile nodes in a higher rate, demanding communication as well as computing resources. However, it is almost impossible for a mobile CQ system to achieve 100% fresh and accurate results due to continuously changing positions of mobile nodes. A key challenge therefore is: How do we achieve the highest possible quality of the query results in both freshness and accuracy, in the presence of changing available resources and changing workloads of location updates and location queries?

In this paper, we propose a resource adaptive and QoS-aware load shedding framework called MobiQual for mobile CQ systems. In this frame work, we are combining both update load shedding and query load shedding . MobiQual is capable of providing high-quality query results by dynamically determining the appropriate amount of update load shedding and query load shedding to be performed according to the application-level QoS specifications of the queries. An obvious advantage of combining query load shedding and update load shedding within the same framework is to empower MobiQual with differentiated load shedding capability, that is configuring query reevaluation periods and update inaccuracy thresholds for achieving high overall QoS with respect to both freshness and accuracy.

MobiQual design has the ability to perform dynamic update load shedding and query load shedding according to changing workload conditions and resource constraints, and has the ability to avoid or reduce severe performance deprivation in query result. MobiQual employs query grouping and space partitioning techniques to reduce the adaptation time required for reconfiguring the system in response to high system dynamics, such as the number of queries, the number of mobile nodes, and the evolving movement patterns.

In contrast to the existing work on scalable query processing and indexing techniques, even under limited resources or overload conditions MobiQual provides a QoS-aware framework for performing both update load shedding and query load shedding, in order to provide highly accurate and fresh query results.

II. LOAD SHEDDING IN MOBILE CQ SYSTEMS: DESIGN OVERVIEW

In a mobile CQ system, the CQ server receives position updates from the mobile nodes through a set of base stations (see Figure 1) and periodically evaluates the installed continual queries (such as continual range or nearest neighbor queries) over the last known positions of the mobile nodes. Since the mobile node positions change continuously, motion modeling [5], [6] is often used to reduce the number of updates sent by the mobile nodes. The server can predict the locations of the mobile nodes through the use of motion models, albeit with increasing errors. Mobile nodes generally use a threshold to reduce the amount of updates to be sent to the server and to limit the inaccuracy of the query results at the server side below the threshold. Smaller thresholds result in smaller errors and higher accuracy, at the expense of a higher load on the CQ server. This is because a larger number of position updates must be processed by the server, for instance, to maintain an index. When the position update rates are high, the amount of position updates is huge and the server may randomly drop some of the updates if resources are limited. This can cause unbounded inaccuracy in the query results. In MobiQual, we use accuracy-conscious update load shedding to regulate the load incurred on the CQ server due to position update processing by dynamically configuring the inaccuracy thresholds at the mobile nodes.

Major problem for the CQ server is to keep the query results up to date by periodically executing the CQs over the mobile node positions. More frequent query re-evaluations leads to increased freshness in the query results, also at the expense of a higher server load. Given limited server resources, when the rate of query re-evaluations is high, the amount of queries to be re-evaluated is vast and the server may randomly drop some of the re-evaluations, causing stale query results (low freshness).

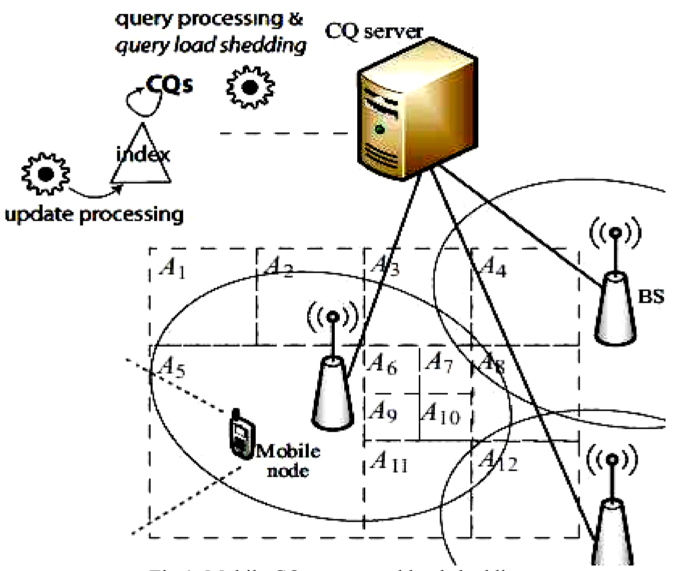

Fig 1: Mobile CQ system and load shedding.

MobiQual utilizes QoS aware query load shedding to control the load incurred on the CQ server due to query reevaluations by configuring the query re-evaluation periods. In general, the total load due to evaluating queries and processing position updates dominates the performance and scalability of the CQ server and thus should be delimited by the capacity of the CQ server. Furthermore, the time-varying processing demands of a mobile CQ system entails that update and query load shedding should be dynamically balanced and adaptively performed in order to match the current workload with the server's capacity, while meeting the accuracy and freshness requirements of queries.

III. THE MOBIQUAL APPROACH

By dynamically determining the amount of load shedding to be performed MobiQual maximizes the overall quality of the query results, based on per-query QoS specifications and subject to processing capacity constraints. The QoS specifications are defined based on two factors: accuracy and freshness. In MobiQual, the QoS specifications are used to decide on not only how to spread out the impact of load shedding among different queries, but also how to find a balance between query load shedding and update load shedding. The main idea is to apply differentiated load shedding to adjust the accuracy and freshness of queries. Namely, load shedding on position updates and query re-evaluations is done in such a way that the freshness and accuracy of queries are non-uniformly impacted.

From the perspective of query load shedding, we make two observations to show that non-uniform freshness in the query results can increase the overall the overall QoS. First, different geographical regions have different numbers of mobile nodes and queries. Second, different queries have different tolerance to position errors in the query results.

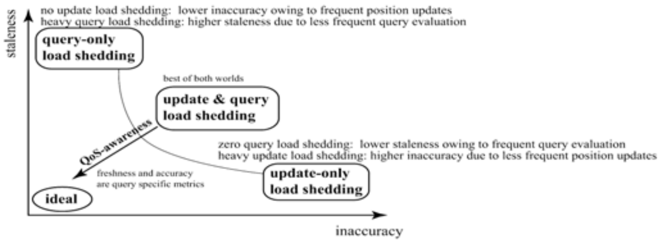
staleness
no update load shedding: lower inaccuracy owing to frequent position updates
heavy query load shedding: higher staleness due to less frequent query evaluation
query-only load shedding
best of both worlds
update & query load shedding
QoS-awareness
zero query load shedding: lower staleness owing to frequent query evaluation
heavy update load shedding: higher inaccuracy due to less frequent position updates
freshness and accuracy are query specific metrics
update-only load shedding
ideal
inaccuracy

Fig 2: QoS-aware update load shedding and QoS-aware query load shedding.

This means that shedding more updates from a region with a higher density of mobile nodes and a lower density of queries can bring a higher reduction on the update load and yet have a smaller impact on the overall query result accuracy. We refer to the load shedding that adjusts the inaccuracy thresholds based on the densities of mobile nodes and queries to maximize the average accuracy of the query results under the QoS specifications as QoS-aware update load shedding.

Similar to update load shedding, we make two observations regarding query load shedding to show that non-uniform freshness in the query results can increase the overall QoS of the mobile CQ system: (1) Different queries have different costs in terms of the amount of load they incur. (2) Different queries have different tolerance to staleness in the query results. Thus it is more effective to shed load (by sacrificing certain amount of freshness) on a costly query than an inexpensive one. This is especially beneficial if the costly query happens to be less severe on freshness, based on its QoS specification. In MobiQual we employ QoS-aware query load shedding: We use query re-evaluation periods as control knobs to perform query load shedding, where the same amount of increase in query reevaluation periods for different queries brings differing amounts of load reduction and QoS degradation with respect to freshness.

MobiQual dynamically maintains a throttle fraction, which defines the amount of load that should be retained. It performs both update load shedding and query load shedding to control the load of the system according to this throttle fraction, while maximizing the overall quality of the query results. MobiQual not only strikes a balance between freshness and accuracy by employing both query and update load-shedding, but also improves the overall quality of the results by utilizing perquery QoS specifications to capture each query's different tolerance to staleness and inaccuracy.

A. Problem Formalization

The objective of the combined load shedding problem is to maximize the $\Psi = \frac{1}{m}(\Psi_v + \Psi_u)$

We now restate the processing constraint by combining the load due to query re-evaluation and update processing. Let zv denote the fraction of the query load retained for a given set of re-evaluation periods {Pj}. We have:

$$z_v = \frac{\sum_{j=1}^{k} f_c(C_j)/P_j}{\sum_{q \in Q} f_c(\{q\})/\tau_{\vdash}}.$$

Similarly, let zu denote the fraction of the update load retained for a given set of inaccuracy thresholds {Δi}. We have:

$$z_u = \frac{\sum_{i=1}^{l} n_i \cdot f_r(\Delta_i)}{n \cdot f_r(\epsilon_{\vdash})}.$$

With these definitions, we can state the processing constraint as follows:

$$z_v + z_u \cdot \gamma \leq z \cdot (1 + \gamma).$$

The parameter γ in the above equation represents the cost of performing update processing. For the ideal case, the query reevaluation costs 1 unit, whereas the update processing costs γ belongs to (0,1] units. γ is not a system-specified parameter and is calculated as follow:

Let U be the observed cost of update processing and V be the observed cost of query reevaluation during the last adaptation period. Then we have

$$\gamma = \frac{U/z_u}{V/z_v}$$

IV. SOLUTION OUTLINE

There are three functional components in the MobiQual system: reduction, aggregation, and adaptation:

- Reduction -- includes the algorithm for grouping the queries into k clusters and the algorithm for partitioning the geographical space of interest into regions. The query groups are incrementally updated when queries are installed or removed from the system. The space partitioning is recomputed prior to the periodic adaptation.

- Aggregation -- involves computing aggregate QoS functions for each query group and region. The aggregated QoS functions for each query group represent the freshness aspect of the quality. The aggregated QoS functions for each region represent the accuracy aspect of the quality. We argue that the separation of these two aspects is essential to the development of a fast algorithm for configuring the reevaluation periods and the inaccuracy thresholds to perform adaptation. QoS-aggregation is repeated only when there is a change in the query grouping or the space partitioning.

- Adaptation is performed periodically to determine:

- the throttle fraction belongs to[1,0], which defines the amount of load that can be retained relative to the load of providing perfect quality .
- the setting of reevaluation periods belongs to[1,k] and
- the setting of inaccuracy thresholds belongs to[1,l] .

The latter two are performed with the aim of maximizing the overall QoS. The computation of the throttle fraction is performed by monitoring the performance of the system and adjusting z in a feedback loop

A. Aggregating the QoS Function

The aim of QoS aggregation is to associate an aggregate function for each query group, and an aggregate function for each region , such that the overall QoS of the system, denoted by Ψ, is maximized. We define

$$\Psi = \frac{1}{m} \sum_{q \in Q} \mathcal{S}_q(\tau_q, \epsilon_q),$$

where m is the total number of queries and Sq(Tq; ϵq) denotes the QoS specification for query q and can be defined as follows:

$$\mathcal{S}_q(\tau_q, \epsilon_q) = \alpha_q \cdot \mathcal{V}_q(\tau_q) + (1 - \alpha_q) \cdot \mathcal{U}_q(\epsilon_q).$$

In other words, Sq(Tq; ϵq)is a linear combination of the freshness QoS function Vq(Tq) and the accuracy one Uq(ϵq). The parameter αqϵ [0, 1], called freshness weight, is used to adjust the relative importance of the two components, freshness and accuracy. Vq(Tq) and Uq(ϵq) are nonincreasing positive functions, where lower staleness bound of Qos function Vq is 1 and lower inaccuracy bound of Qos function Uq is 1. Since the query groups are non-overlapping, we have the following:

$$\mathcal{V}^*_j(P_j) = \sum_{q \in C_j} \alpha_q \cdot \mathcal{V}_q(P_j).$$

We approximate the Vq functions using piecewise linear functions of k equal-sized segments along the input domain [T├,T┤]. This enables us to represent the aggregate QoS functions as piecewise linear functions of k segments as well. Fig. 3 gives an example of aggregating two piecewise linear functions of four segments each.

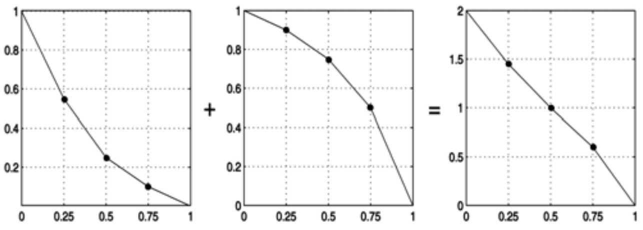

Fig. 3 Example of aggregating

V. CONCLUSION

In this paper, we have presented MobiQual, a load shedding system aimed at providing high-quality query results in mobile continual query systems. MobiQual has three unique properties. First, it uses per-query QoS specifications that characterize the tolerance of queries to staleness and inaccuracy in the query results, in order to maximize the overall QoS of the system. Second, it effectively combines query load shedding and update load shedding within the same framework, through the use of differentiated load shedding concept. Finally, the load shedding mechanisms used by MobiQual are lightweight, enabling quick adaption to changes in the workload, in terms of the number of queries, number of mobile nodes, or their changing movement patterns. Through a detailed experimental study, we have shown that the MobiQual system significantly outperforms approaches that are based on query-only or update-only load shedding, as well as approaches that do combined query and update load shedding but lack the differentiated load shedding elements of the MobiQual solution, in particular, the query grouping and space partitioning mechanisms.

There are several interesting issues for future work. MobiQual should be able to dynamically adjust the values of the l (number of shedding regions) and k (number of query groups) parameters as the workload changes. An overestimated value for these parameters means lost opportunity in terms of minimizing the cost of adaptation, whereas an underestimated value means lost opportunity in terms of maximizing the overall QoS. In this paper, we have shown that the time it takes to run the adaptation step is relatively small compared to the adaptation period in most practical scenarios. This means that relatively aggressive values for l and k could be used to optimize for QoS without worrying about the cost of adaptation. We leave it as a future work to adapt these parameters dynamically.

REFERENCES

[1] C. Science and T. Board. IT Roadmap to a Geospatial Future. The National Academics Press, November 2003.

[2] Google RideFinder home page. http://labs.google.com/ridefinder, Febuary 2006.

[3] B. Gedik, K.-L. Wu, P. S. Yu, and L. Liu, "Processing moving queries over moving objects using motion adaptive indexes," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 5, 2006.

[4] H. Hu, J. Xu, and D. Lee, "A generic framework for monitoring continuous spatial queries over moving objects," in Proceedings of ACM SIGMOD, 2005.

[5] O. Wolfson, P. Sistla, S. Chamberlain, and Y. Yesha, "Updating and querying databases that track mobile units," Distributed and Parallel Databases, vol. 7, no. 3, 1999.

[6] Civilis, C. S. Jensen, and S. Pakalnis, "Techniques for efficient road-network-based tracking of moving objects," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 5, 2005.

AUTHORS PROFILE

Mrs. R.Lakshmi Tulasi currently working as Professor & HOD in Dept of CSE. She received her B.Tech from Bapatla Engineering College with Distinction and M.Tech from JNTU Anantapur. Currently she is pursuing Ph.D from JNTU, Hyderabad. She has 12 years of teaching experience and published many papers at various international journals and conferences. Her Research areas includes Computer networks, Computer Organization and Information retrieval Systems.

Sk.Mahaboob Basha, pursuing M.Tech (CSE) from QIS College of Engineering and Technology, Ongole Affiliated to JNTU, Kakinada. His Interested areas includes Computer networks, Computer Organization and Mobile Computing.