**Load Shedding in Mobile Systems with MobiQual**

**Abstract:**

In location-based, mobile continual query (CQ) systems, two key measures of quality-of-service (QoS) are: freshness and accuracy. To achieve freshness, the CQ server must perform frequent query reevaluations. To attain accuracy, the CQ server must receive and process frequent position updates from the mobile nodes. However, it is often difficult to obtain fresh and accurate CQ results simultaneously, due to 1) limited resources in computing and communication and 2) fast-changing load conditions caused by continuous mobile node movement. Hence, a key challenge for a mobile CQ system is: How do we achieve the highest possible quality of the CQ results, in both freshness and accuracy, with currently available resources? In this paper, we formulate this problem as a load shedding one, and develop MobiQual-a QoS-aware approach to performing both update load shedding and query load shedding. The design of MobiQual highlights three important features. 1) Differentiated load shedding: We apply different amounts of query load shedding and update load shedding to different groups of queries and mobile nodes, respectively. 2) Per-query QoS specification: Individualized QoS specifications are used to maximize the overall freshness and accuracy of the query results. 3) Low-cost adaptation: MobiQual dynamically adapts, with a minimal overhead, to changing load conditions and available resources. We conduct a set of comprehensive experiments to evaluate the effectiveness of MobiQual. The results 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 that lack the QoS-awareness properties of MobiQual, as well as the solutions that perform query-only or update-only load shedding.

**Algorithm Used:**

Quality **l**oss **b**ased **c**lustering algorithm

**System Architecture:**

****

**Existing System:**

To the best of our knowledge, none of the existing work has exploited the potential of performing load shedding to maximize the application-level freshness and accuracy of mobile queries. In contrast to existing work on scalable query processing and indexing techniques, MobiQual provides a QoSaware framework for performing both update load shedding and query load shedding, in order to provide highly accurate and fresh query results, even under limited resources or overload conditions. Moreover, as a complementary solution, MobiQual can easily take advantage of existing query processing and indexing techniques.

**Proposed System:**

In this paper, we present MobiQual − a resource-adaptive and QoS-aware load shedding framework for mobile CQ systems. MobiQual is capable of providing high-quality query results by dynamically determining the appropriate amount of update load shedding (discarding certain location update messages) and query load shedding (skipping some query re-evaluations) 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 sheddingcapability that is configuring *query re-evaluation periods* and *update inaccuracy thresholds* for achieving high overall QoS with respect to both freshness and accuracy.

Another salient feature of MobiQual design is its ability to perform dynamic update load shedding and query load shedding according to changing workload characteristics and resource constraints, and its ability to reduce or avoid severe performance degradation in query result quality under such conditions. Mobi- Qual employs *query grouping* and *space partitioning* techniques to reduce the adaptation time required for re-configuring the system in response to high system dynamics, such as the number of queries, the number of mobile nodes, and the evolving movement patterns.

**Module Description:**

1. Load Shedding in Mobile CQ Systems
2. The MobiQual Approach
3. QoS-aware update load shedding
4. QoS-aware query load shedding

**Load Shedding in Mobile CQ Systems:**

In a mobile CQ system, the CQ server receives position updates from the mobile nodes through a set of base stations and periodically evaluates the installed continual queries (such as continual range or nearest neighbor queries) over the last known positions of the mobile nodes.3 Since the mobile node positions change continuously, motion modeling, 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.

**The MobiQual Approach:**

The MobiQual system aims at performing dynamic load shedding to maximize 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.

**QoS-aware update load shedding:**

We use inaccuracy thresholds from motion modeling as control knobs to adjust the amount of update load shedding to be performed, where the same amount of increase in inaccuracy thresholds for different geographical regions brings differing amounts of load reduction and QoS degradation with respect to 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*.

**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. We refer to the load shedding that uses query re-evaluation periods to maximize the average freshness of the query results under the QoS specifications as *QoS-aware query load shedding*.

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. As illustrated in 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 per-query QoS specifications to capture each query’s different tolerance to staleness and inaccuracy.

# System Configuration:-

# H/W System Configuration:-

# Processor - Pentium –III

Speed - 1.1 Ghz

RAM - 256 MB(min)

Hard Disk - 20 GB

Floppy Drive - 1.44 MB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor - SVGA

# S/W System Configuration:-

* Operating System :Windows95/98/2000/XP
* Application Server : Tomcat5.0/6.X
* Front End : HTML, Java, Jsp,Servlet,Ajax
* Scripts : JavaScript.
* Server side Script : Java Server Pages.
* Database : MsAccess
* Database Connectivity : JDBC.