SLIM: A Scalable Location-Sensitive Information Monitoring Service

Bhuvan Bamba[♠], Kun-Lung Wu[♣], Buğra Gedik[♢], Ling Liu[†]
[♠] Oracle America Inc.
[†] College of Computing, Georgia Institute of Technology
[♢] Dept. Computer Engineering, Bilkent University
[♣] IBM T.J. Watson Research Center
bhuvan.bamba@oracle.com, klwu@us.ibm.com, bgedik@cs.bilkent.edu.tr, lingliu@cc.gatech.edu

***Abstract*—Location-sensitive information monitoring services are a centerpiece of the technology for disseminating content-rich information from massive data streams to mobile users. The key challenges for such monitoring services are characterized by the combination of spatial and non-spatial attributes being monitored and the wide spectrum of update rates. A typical example of such services is "alert me when the gas price at a gas station within 5 miles of my current location drops to $4 per gallon". Such a service needs to monitor the gas price changes in conjunction with the highly dynamic nature of location information. Scalability of such location sensitive and content rich information monitoring services in the presence of different update rates and monitoring thresholds poses a big technical challenge. In this paper, we present SLIM, a scalable location sensitive information monitoring service framework with two unique features. First, we make intelligent use of the correlation between spatial and non-spatial attributes involved in the information monitoring service requests to devise a highly scalable distributed spatial trigger evaluation engine. Second, we introduce single and multi-dimensional safe value containment techniques to efficiently perform selective distributed processing of spatial triggers to reduce the amount of unnecessary trigger evaluations. Through extensive experiments, we show that SLIM offers high scalability for location-sensitive, content-rich information monitoring services in terms of the number of information sources being monitored, number of users and monitoring requests.**

I. Introduction

Advancements in wireless communication technology and cloud computing are enabling many JIT Web services for delivery of contextual information to mobile users at the right time and the right place. Other technological advances like large-scale deployment of sensor networks and information delivery systems have led to the availability of large amounts of context-aware location-sensitive information. Such information is typically delivered in the form of data streams that arrive continuously, rapidly and in real time [1], [2]. A mobile information monitoring system is characterized by a large number of such data streams, some delivering location of a large number of mobile users and others delivering location-sensitive monitored data. Examples of such location-sensitive monitored data may include gas prices at gas stations, traffic conditions at major junctions or pollution levels in different parts of a city. In such an environment, *spatial triggers* provide useful means of allowing users to express their location-specific information needs.

In this paper, we introduce SLIM[1], a service for efficiently monitoring information streams comprised of spatial as well as non-spatial data in a distributed fashion. Users express their information needs in terms of spatial triggers which involve spatial and non-spatial predicates. For example, a user may install a trigger of the form *"alert when within one mile of gas station G and the price of gas is below $4"*. The central server receives location and other data updates from a possibly large number of sources to determine the opportune moment to activate triggers. As opposed to current works which focus on efficient evaluation of different types of continuous queries (e.g. kNN query, range query) based on spatial predicates alone ([3], [4], [5]), we consider a service receiving a large number of location-sensitive attributes from multiple data sources. Introduction of non-spatial predicates renders existing solutions incapable of handling the problem efficiently. However, it presents an opportunity to perform optimizations beyond those possible with spatial attributes alone.

Spatial trigger processing requires meeting three demanding objectives: (i) *high accuracy*, which ensures no triggers are missed, (ii) *server scalability*, which guarantees that the information monitoring server scales to a large number of triggers, growing base of mobile users and a large number of data streams, and (iii) *immediate evaluation of updates* in order to activate triggers at the earliest possible moment to allow users to react.

Solutions in the streaming domain provide optimizations which lead to approximate answers [6] or delayed evaluation as in the case of batch processing [7]. Both are unable to meet our objectives of immediate evaluation with 100% accuracy. A simple solution processes each and every data update on arrival to determine if any triggers are activated. However, this *in-transit processing* approach hurts the scalability of the system. Load shedding approaches [8] drop data updates randomly or selectively but cannot guarantee 100% accuracy.

Our approach takes into account interaction between various attributes which facilitates selective update evaluation against installed trigger information, dropping updates which have

[1]SLIM is an acronym for Scalable Location-sensitive Information Monitoring

zero probability of activating any installed triggers. In order to achieve 100% accuracy while selectively evaluating data updates, we introduce the concept of *safe containment*. Safe value containers are computed for each data attribute in SLIM allowing the service to drop updates which lie within their respective safe value containers without evaluation. Additionally, safe value containers are communicated back to relevant data sources, seeking their cooperation in the monitoring process and allowing for communication cost savings. However, safe value container computation leads to additional processing cost at the server.

In order to meet this challenge, we present efficient algorithms for safe value container computation for single-dimensional and multi-dimensional data. Our experimental evaluation shows that server scalability is enhanced multiple times by deployment of safe containment techniques. Mobile clients will also benefit in terms of energy and bandwidth consumption by monitoring their position within their respective two-dimensional safe value containers.

The rest of the paper is structured as follows. Section II provides a motivating example. Section III introduces the fundamental concept of safe containment followed by the service overview in Section IV. This is followed by a description of our safe value container computation algorithms (Section V). An experimental evaluation using a real world road network is presented in Section VI. Section VII provides a brief discussion of related work followed by the conclusion in Section VIII.

II. Problem Motivation

We motivate the optimization opportunities presented by addition of less dynamic non-spatial attributes to the information monitoring problem in mobile services with the aid of an example. Figure 1 displays mobile users *A, B* and *C* with installed spatial triggers on gas station *G*. For future reference, *A, B, C and G* are termed as *objects* in the system. Each user specifies spatial as well as non-spatial predicate conditions for her spatial trigger. For example, the spatial trigger for user *A* requires activation when the *user is within one mile of the gas station G* and *gas price is below $4*. Similarly, users *B* and *C* specify predicate conditions for their spatial triggers. Each object in the system will have an associated monitored attribute. For example, A, B and C have location as a monitored attribute; whereas gas station G has the gas price as a monitored attribute.

For the sake of exposition, we assume that the rate at which the pricing data stream is delivered is half the rate of the location data stream. The location data stream comprises of location information between time instants t_1 and t_4 (at constant interval) as shown in figure 1. Similarly, pricing data is received at the server at time instants t'_2 ($t_2 < t'_2 < t_3$) and t'_4 ($t'_4 > t_4$). A simple approach would process the data updates as and when they are received. However, it is unnecessary for the server to process all received data updates.

Firstly, we observe that user *A* lies outside the spatial region associated with her installed trigger at time instants t_1, t_2 and t_4. If it was possible for the service to determine that user *A* cannot be positioned within the spatial trigger region at these time instants, the location data for the user at these time instants can be dropped. Alternately, the user can save on energy and bandwidth costs by not sending updates to the server at these time instants. Secondly, we observe that the pricing data at time instant t'_2 does not satisfy the constraint for any of the installed triggers. If the server can determine that the pricing data constraint for any of the installed triggers cannot be satisfied at t'_2, this update may be dropped. Thirdly, the position update for all mobile users at time instant t_3 can be dropped if the service recognizes the fact that though the spatial constraints are satisfied for all the triggers, the non-spatial constraint cannot be satisfied due to previous pricing data update at t'_2. Lastly, the pricing data update at time instant t'_4 may be dropped, even though it satisfies the non-spatial constraints for the trigger installed by user *C*, if the server recognizes that none of the users satisfy the spatial constraints for their respective triggers at this time instant. With these observations in mind, we define the concepts associated with safe containment followed by an overview of the SLIM service.

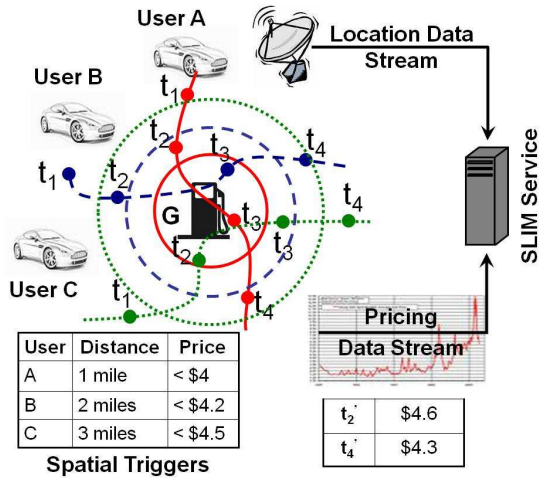


Fig. 1: Motivating Example

III. Safe Containment

The concept of safe containment can be applied to incoming updates ensuring that updates with zero probability of activating triggers may be dropped without processing. Otherwise, trigger evaluation needs to be performed to determine if any of the triggers may be activated by an incoming data update.

Definition 1: (**Safe Value Container**) A safe value container $\psi(o, s)$ is defined for each object o for any monitored attributes s relevant to the object o. As long as the value of the monitored attribute lies within its safe value container no dependant triggers can be activated by updates to this attribute value. For example, $\psi(A, l)$ represents the two-dimensional safe value container for user A (in Figure 1) based on its current location l. Similarly, $\psi(G, p)$ represents the single-dimensional safe value container for the gas station G based on the current value of the monitored attribute, gas price p.

Definition 2: (**Inclusive Trigger Set**) When a safe value container $\psi(o,s)$ overlaps the predicate range of a set of relevant triggers $\mathcal{T}^I$, $\mathcal{T}^I$ is defined as the inclusive trigger set. Note that any future data updates $v(s) \in \psi(o,s)$ can still be dropped without trigger processing, however, updates for other attributes may activate an inclusive trigger $\in \mathcal{T}^I$. Section V-A exemplifies the concept of inclusive triggers.

The goal of safe containment is to minimize the number of trigger evaluations performed. We identify the following requirements for safe value container computation.

Lightweight Construction: Safe value container computation needs to be performed for each object o with respect to monitored attributes s delivering data relevant to the object o, thus making it imperative that safe value container computation is performed quickly in order to avoid server overload.

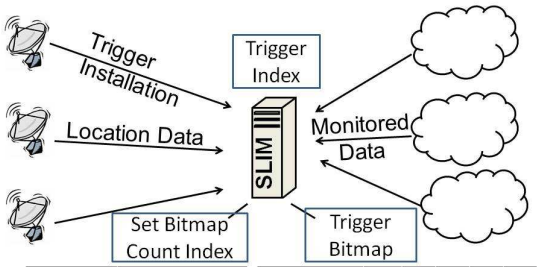
Fast Containment Check: The containment check needs to determine if the value of monitored attributes s for object o at time t, $v(s,t)$, lies within the current safe value container $\psi(o,s)$ of the object.

Maximum Extent: As long as the attribute value $v(s,t)$ lies within the safe value container $\psi(o,s)$, the data update can be dropped. The goal of minimizing the number of trigger evaluations can be achieved by maximizing the probability of the data value staying within the safe value container. For example, a safe value container of largest possible area around the current position of a mobile user would allow the user to stay within its safe value container for maximum time.

IV. SLIM Service Overview

This section provides a brief overview of the SLIM service and the data structures maintained at the server (Figure 2). The spatial triggers are installed at the server and indexed using the relevant indexing techniques [9], [10]. We maintain a main-memory index on the trigger identifiers with each trigger being associated with a bitmap. The length of the bitmap is equivalent to the number of non-spatial data sources delivering information to the service. For a service which has one million triggers installed and receives information from 10 different sources, this translates to a 10MB main memory structure. Bits are set for a data source whose value satisfies the corresponding trigger predicates (inclusive triggers).

Safe value containers are computed by the trigger processing server and communicated back to the data source. Non-spatial data sources deliver data to the trigger processing server only when the current attribute value moves outside the safe value container and a new container needs to be computed. Mobile clients are not required to report their location unless the non-spatial attribute bitmap is set for at least one relevant trigger. This leads to large savings in wireless communication and energy costs for mobile clients. The trigger processing service is responsible for informing mobile clients when the bitmap is completely set for at least one relevant trigger for the client using the *set bitmap count index*. The service returns the safe value container for the mobile client if no spatial predicates are satisfied.



USER ID	# Set Triggers
1765341	0
9804342	2
1345098	1

TRIGGER ID	S_1	S_2	S_3	S_4	S_5
1987543	1	0	1	0	1
3876543	1	1	1	1	0
4156096	1	1	1	1	1

Fig. 2: SLIM Service Overview

The next section describes the safe value container computation algorithms which forms the basis for our service scalability.

V. Safe Value Container Computation

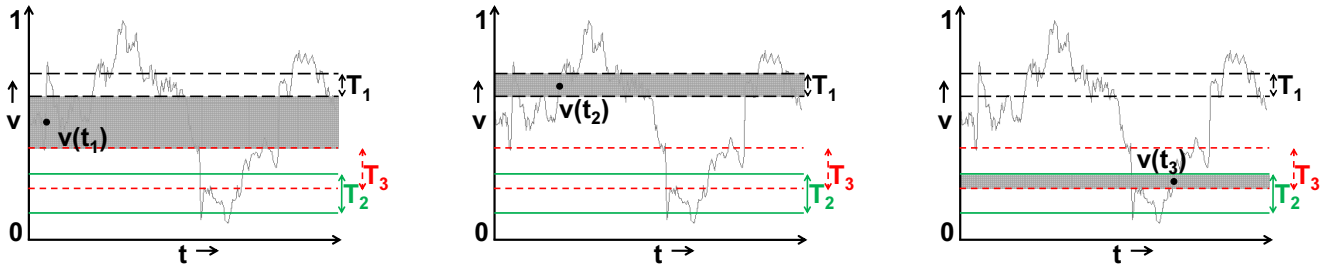
A. Single-Dimensional Safe Value Containers

We define single-dimensional safe value containers $\psi(o,s)$ for each object o and single-dimensional monitored attribute s relevant to o. Such safe value containers can be represented using a set of value ranges $[L_1^1, H_1^1], ..., [L_p^1, H_p^1]$ where p $\geq$ 1, such that, as long as the attribute value lies within one of the value ranges no triggers relevant to o can be activated. Figure 3 displays the plot of a single-dimensional attribute v varying with time t. The predicate conditions for three different installed triggers T_1, T_2 and T_3 over the attribute v are displayed by horizontal lines. We identify three separate cases for safe value container computation.

Case I: When attribute value lies outside any relevant trigger range, the safe value container is identified as the range $[l,h]$, such that, no relevant triggers can be activated as long as the attribute value lies within this range. Figure 3(a) shows an example where the value of the attribute at time t_1, $v(t_1)$, lies in $[l,h] = [T_3.high, T_1.low]$. This safe value container remains valid till the attribute value lies between $T_3.high$ and $T_1.low$.

Case II: When attribute value lies within a relevant trigger range, the bounds of the predicate condition associated with the trigger form the safe value container and the trigger is an inclusive trigger. As long as future attribute values lie within this range no triggers can be activated. However, updates to other relevant attributes may result in activation of the *inclusive trigger*. Figure 3(b) shows an example scenario where the attribute value $v(t_2)$ lies within the trigger range of T_1 and the safe value container is identified as $[l,h] = [T_1.low, T_1.high]$. Trigger T_1 is an inclusive trigger in this scenario.

Case III: When attribute value lies within multiple trigger ranges, the safe value container is identified as the *minimal intersection* of the trigger ranges and these triggers form the *inclusive trigger set*. Figure 3(c) shows an example scenario where the attribute value $v(t_3)$ lies within the trigger range of T_2 as well as T_3. The safe value container is identified as $[l,h]$



(a) (b) (c)

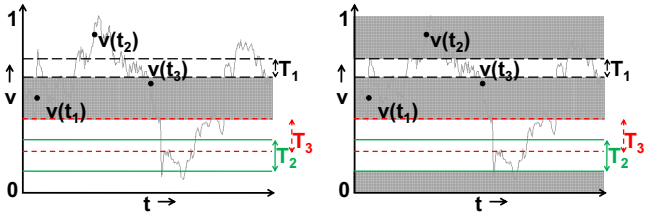
Fig. 3: Single-dimensional safe value container computation scenarios. (a) Case I: attribute value lies outside any relevant trigger regions, (b) Case II: attribute value lies inside a relevant trigger region and (c) Case III: attribute value lies within the intersection of multiple relevant trigger regions.

$= [T_3.low, T_2.high]$. Triggers T_2 and T_3 form the inclusive trigger set.

We consider two different flavors for safe value container computation for single-dimensional data: single and multiple. Multiple safe value container computation is more expensive compared to single safe value container computation, however, it leads to *fewer trigger evaluations compared to single safe value containers when the rate of change of data values is high and update frequency is low* as shown in our experiments. We discuss the algorithm for single safe value container computation and depict the differences with multiple safe value containers with the aid of an example (refer to [11] for details).

Figure 4 displays an example scenario where the source generates data updates $v(t_1)$, $v(t_2)$ and $v(t_3)$ at time instants t_1, t_2 and t_3. Consider the single safe value container in Figure 4(a). The data source being considered here delivers data which has a *high rate of change* and *low update frequency*. The value $v(t_2)$ at time instant t_2 lies outside the current safe value container. Hence, the safe value container is invalidated and a new safe value container is computed. At time instant t_3, the data value again lies within the safe value container associated with $v(t_1)$; as this safe value container has been previously invalidated on receiving data update $v(t_2)$, the safe value container associated with $v(t_1)$ is recomputed as the new safe value container at time instant t_3. Figure 4(b) displays multiple safe value containers, represented by shaded stripes, which avoid the unnecessary recomputation of safe value containers described in the above scenario. The values $v(t_1)$, $v(t_2)$ and $v(t_3)$ lie inside one of the multiple safe value containers at each time instant. Note that multiple safe value containers can only be computed when current attribute value lies outside all relevant trigger predicate range values.

In order to limit the safe value container computation costs, we consider a small number of triggers in the vicinity of the current attribute value. The attribute value range is divided into a number of blocks and the range block $[r_l, r_h]$ within which the current attribute value lies is computed. Next, we retrieve the set of triggers $\mathcal{T}'$ relevant to object o intersecting the range block $[r_l, r_h]$. If the set of intersecting triggers $\mathcal{T}'$ is empty, the entire range block is returned as the safe value container. Otherwise, we proceed to compute the safe value container dependent on the intersecting trigger bounds as for the three cases above.



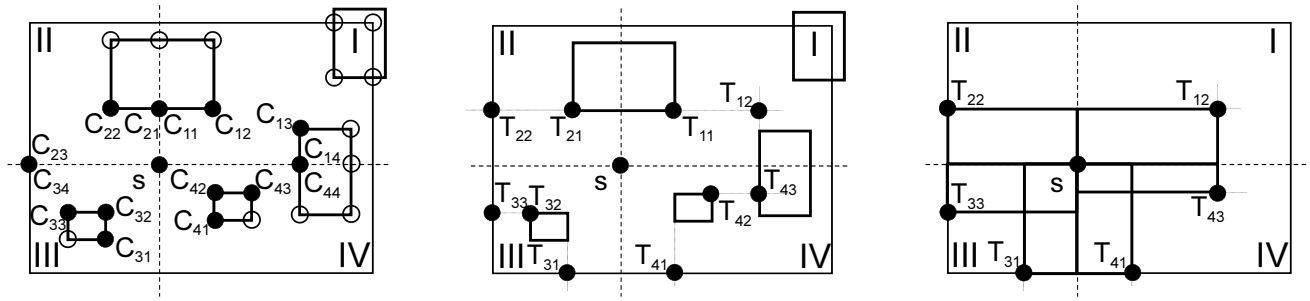
(a) (b)

Fig. 4: Single Safe Value Container vs. Multiple Safe Value Containers

B. Multi-dimensional Safe Value Containers

Data streams may deliver multi-dimensional data instead of single-dimensional data, thus requiring computation of multi-dimensional safe value containers. For example, the data stream delivering location updates provides two-dimensional data (x, y) denoting the current location of the mobile user. Section III defined the requirements for safe value container characteristics. Multi-dimensional safe value containers may have some additional requirements like *compact representation*. For example, two-dimensional safe value containers for location updates need to be communicated back to the mobile user over a wireless channel which may lead to significant communication costs and energy consumption on the mobile client. Additionally, mobile clients need to track their location within the safe value container. A rectangular-shaped two-dimensional safe value container requires only two points (bottom-left and top-right) for representation and it is computationally efficient to detect a point inside a rectangle. We now outline the procedure for computation of two-dimensional safe value containers. The underlying concepts may be applied to extend this approach to safe value container computation for higher dimensions.

We describe a process for two-dimensional safe value container computation similar to skyline point computation [12], [13]. The computed skyline points form the corner points of the rectangular safe value container in two-dimensional space. In order to reduce the safe value container computation costs, relevant triggers in the vicinity of the current value (a_1, a_2) are considered. This is achieved by overlaying a grid structure over the two-dimensional space associated with the data being delivered by the source. The algorithm accepts



(a) Candidate Point Set (b) Tension Point Set (c) Component Rectangles

Fig. 5: Two-dimensional Safe Value Container Computation

the two-dimensional data value (a_1, a_2) and the current two-dimensional grid cell in which the data value resides as inputs. The set of triggers intersecting this two-dimensional grid cell are considered for safe value container computation. In case no relevant triggers intersect the grid cell, the entire two-dimensional cell is returned as the safe value container. Otherwise, the algorithm proceeds to calculate the safe value container by applying the concept of dominating points and skyline computation as mentioned earlier.

The algorithm partitions the two-dimensional grid cell into four quadrants with (a_1, a_2) as the origin. We define a set of *candidate points* and a set of *tension points* for each quadrant. The candidate point set is the set of points which can potentially form a corner point of the rectangular safe value container. Tension points are obtained from the set of candidate points by ensuring that only points that form largest possible rectangular regions not overlapping the spatial region associated with any relevant trigger are selected.

The set of candidate points is determined as follows. Firstly, the spatial region corner for each relevant trigger is selected as a candidate point in its appropriate quadrant. For triggers which do not completely lie inside the grid cell, the intersection points of the cell boundary and the trigger spatial region are considered as candidate points instead of the corner points which fall outside the grid cell. Secondly, for trigger spatial regions which intersect the x-axis or y-axis of the coordinate axes with origin at (a_1, a_2), we also consider points of intersection of the triggers with the axes as candidate points. The algorithm trims the set of candidate points in the next step. Firstly, in case multiple candidate points in a quadrant intersect the x-axis (or y-axis), all candidate points other than the point on the x-axis (or y-axis) closest to the origin are removed from the candidate point set. If no intersecting points are present on the x-axis, the point of intersection of the x-axis and the cell is added to the candidate point set. Further, we remove points which *fully dominate* any other point from the candidate set. A point P_1 is said to fully dominate point P_2 if $abs(P_1.x) > abs(P_2.x)$ and $abs(P_1.y) > abs(P_2.y)$. Finally, the points are sorted according to increasing distance of the x-coordinate from the origin. Points with the same x-coordinate are arranged in order of decreasing distance of y-coordinate from origin.

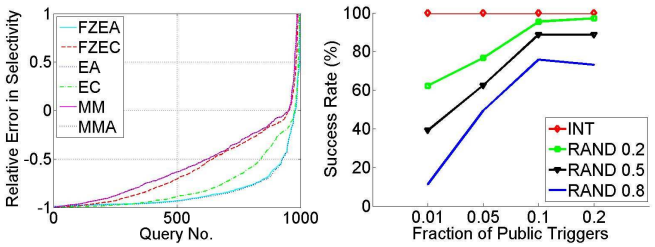
The set of candidate points is then processed to obtain the set of tension points. Each tension point T_{Qi}, where $Q \in \{1, 2, 3, 4\}$ represents the quadrant the point belongs to, is assigned the same x-coordinate as the corresponding candidate point C_{Qi}. T_{Qi} is assigned the same y-coordinate as that of C_{Qi-1}, or T_{Qi-1} if T_{Qi} and T_{Qi-1} have the same x-coordinate. The y-coordinate of T_{Q1} is set as either the top bound of the cell or the y-coordinate of a candidate point intersecting the y-axis if any.

The set of tension points form the opposite corner (opposite to the origin) of the set of candidate *component rectangles* in each quadrant. The final safe value container is composed of the intersection of the component rectangles from each quadrant. As opposed to an optimal solution which enumerates every possible combination of component rectangles and computes metrics for each combination thus taking quartic time, our approach performs greedy decisions. We refer readers to our technical report [14] for a detailed discussion of the various heuristics.

Figure 5 shows an example of our safe value container computation approach. The candidate point set for the given scenario is as shown in Figure 5(a). The black dots represent the candidate points, whereas the hollow dots represent points which are trimmed from the candidate point set as explained above. Figure 5(b) displays the set of tension points obtained from the candidate point set. Figure 5(c) displays the component rectangles formed by selecting a few of the tension points. The final rectangular safe value container is composed out of these component rectangles. Extension of this approach to larger dimensional spaces may lead to expensive computations. For example, three-dimensional spaces would require the processing of relevant points in each octant of the coordinate system. However, intuitively SLIM will benefit more for higher dimensionality data in terms of lower computation costs when compared to the gain in performance experienced for single or two-dimensional data simply due to lower selectivity of the predicate ranges for the triggers in higher dimensions. Our experimental evaluation is limited to two-dimensional spaces and we observe these trends for single-dimensional and two-dimensional data.

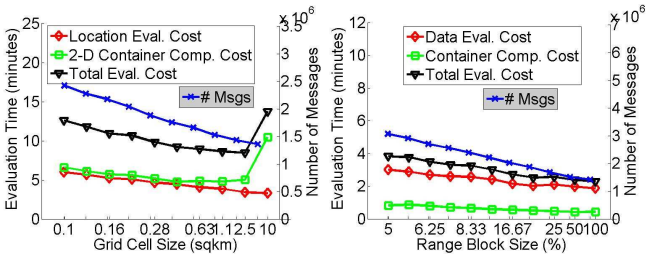
VI. Experimental Evaluation

In this section, we evaluate the performance of the SLIM system to exhibit the benefits of our safe containment techniques. We benchmark the performance of the SLIM approach



(a) (b)

Fig. 6: Scalability of in-transit processing and random dropping with varying fraction of public triggers



(a) (b)

Fig. 7: (a) Results with varying grid cell size for two-dimensional safe value containers. (b) Results with varying range block size for single-dimensional safe value containers.

against three different approaches: in-transit processing (*INT*) which processes all data updates in the order of their arrival, random update dropping (*RND*) and the safe value container approach applied to the spatial attribute (*SR*) alone. The INT approach displays the inability of an approach which processes data on arrival to scale with the increasing number of users/triggers in the system. The RND approach displays the inability of a random update dropping approach in activating triggers with high accuracy. We study the performance of these techniques based on the following metrics: 1) *Computation costs*. These are measured as a combination of trigger evaluation costs and safe value container computation costs. 2) *Communication costs*. The system aims at reducing client-to-server communication costs, especially in a wireless environment where high communication costs also lead to high energy consumption on the mobile client. 3) *Trigger activation success rate*. We aim to achieve 100% trigger activation success rate. The sequence of triggers which should be activated are determined by the in-transit processing approach. We relax the real-time processing requirements for this approach in order to determine the trigger activation sequence.

A. Experimental Setup

We simulate the proposed mobile information monitoring system using an event-based simulator. The simulator generates a trace of vehicles moving on a real-world road network using maps available from the National Mapping Division of the U.S. Geological Survey [15]. Vehicles are randomly placed on the road network, according to traffic densities determined from the traffic volume data in [16], ensuring appropriate traffic densities on different road types. We use a map of Atlanta and surrounding regions, which covers an area around 1000 km^2 in expanse, to generate the trace. Our experiments use traces generated by simulating vehicle movement for a period of one hour with results averaged over a number of such traces. Default values allow us to simulate the movement of a set of 10,000 vehicles with each vehicle generating location updates with a period of 1 second.

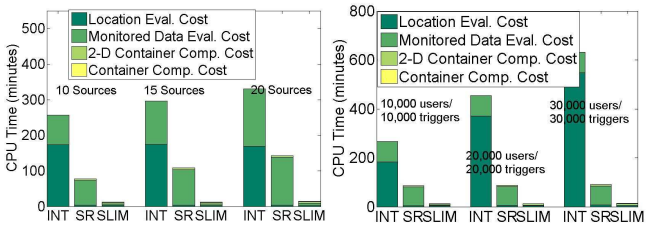
The other part of the simulator models multiple data sources generating data relevant to the locations of interest. A default set of 10,000 locations of interest are distributed over the entire map with multiple hotspots. The data arrival at the information monitoring server is modeled as a poisson process; hence, the interarrival times are exponentially distributed. A maximum of twenty data sources are considered in the system with a different set of attributes monitored at each location of interest. The default trigger information consists of a set of 10,000 triggers with private, shared and public triggers installed in the system determining the number of triggers relevant to each client. We consider a dynamically changing trigger set where triggers are inserted and deleted periodically.

B. Experimental Results

1) Limitations of Other Approaches: This experiment is designed to expose the limitations of the in-transit processing and random dropping approaches. We vary the fraction of public triggers from 0.01 to 0.2 and study the performance of the in-transit processing approach (INT) and the random dropping approach for drop probabilities of 0.2, 0.5 and 0.8 (RND 0.2, RND 0.5, RND 0.8). When the fraction of public triggers is 0.01, we have around 101 relevant triggers per subscriber in the system. On increasing this to 0.2, we have more than 2000 relevant triggers per subscriber installed at the server.

Figure 6(a) plots the CPU time for each approach as we vary the fraction of public triggers. It can be observed from the figure that the in-transit approach is not at all scalable even for the lowest fraction of public triggers as it requires nearly 200 minutes of CPU time to process data received over 60 minutes. Figure 6(b) displays the success rate for each approach and shows that this approach has 100% success rate. However, this approach will start dropping updates due to the heavy load on the server. A simple alternative is to randomly drop updates as they are received at the server. Figure 6(a) shows the CPU time required for random dropping approaches with different drop probabilities. The CPU load reduces as we increase the drop probability for the random dropping approach. However, even with very high drop probability of 0.8 the system is not scalable for higher fraction of public triggers. For low fraction of public triggers the success rate is unacceptable (Figure 6(b)); hence, this approach fails spectacularly for a personalized service. Safe value container approaches rectify the situation by allowing the system to determine and drop updates which do not activate relevant triggers.

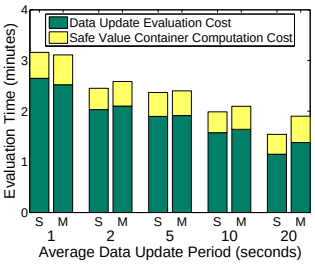
2) Determining Range Block Size and Grid Cell Size: Figure 7(a) displays the evaluation time and the number of update messages received at the server for a location data stream utilizing the two-dimensional safe value container computation algorithm. A total of 60 million update messages are



(a) (b)

Fig. 8: Scalability comparison for in-transit, spatial safe value container and SLIM with (a) varying number of monitored sources. (b) varying number of users and triggers.

Average Update Period (seconds)	Single SVC (# messages in millions)	Multiple SVC (# messages in millions)	Gap (%)
1	1.749	1.749	0
2	1.643	1.640	0.18
5	1.666	1.638	1.56
10	1.622	1.558	3.95
20	1.490	1.403	5.84



(a) (b)

Fig. 9: Performance comparison of single vs. multiple safe value containers with varying update period.

generated for the duration of this experiment. The figure shows the results for evaluation time as we vary the grid cell size from 0.1 km^2 to 10 km^2, plotted on the left y-axis. There are two costs involved in the system: two-dimensional safe value container computation cost and the location update evaluation cost for mobile clients positioned outside their current safe value container.

We can observe from the figure that the safe value container computation cost first slowly decreases as we increase the grid cell size. This is due to lower number of safe value container computations being performed for larger grid cell sizes. As the grid cell size is increased, a larger number of triggers in the vicinity of the user are considered, allowing for computation of larger safe value containers. This leads to lower number of messages being processed at the server for larger grid cell sizes as mobile clients stay within the safe value container for longer time periods. The number of messages processed are plotted on the right y-axis. However, with increasing grid cell sizes, each safe value container computation becomes more expensive as it considers a larger number of trigger regions. For the largest grid cell size of 10 km^2, each safe value container computation is expensive enough to outweigh the effect of lower number of safe value container computations being performed. The location update evaluation costs decline with increasing grid cell sizes as fewer number of messages are evaluated at the server. The overall computation costs are lowest for a grid cell size of 2.5 km^2 as visible from the figure. Figure 7(b) plots the evaluation times and the number of messages as we vary the range block size for single-dimensional data on the left y-axis. The safe value container computation costs steadily decline as we increase the range block size. Again, this is due to fewer number of messages being processed at the server with increasing range block size, as plotted on the right y-axis. Larger range block sizes allow us to compute safe value containers of larger extent which leads to fewer number of data updates being processed at the server. Even though the cost of each safe value container computation increases with increasing range block size, the overall safe value container computation costs slowly decline. Data update evaluation costs also decrease with increasing range block size. Range block size of 50% has lowest total processing costs.

3) System Scalability: Our mobile system is designed with the goal of achieving 100% trigger activation success rate. The following experiment considers only those approaches which can meet this criteria and studies the performance of SLIM in comparison with in-transit processing (INT) and the safe value container approach applied to spatial data alone (SR). As can be seen from Figure 8(a), the SLIM approach outperforms in-transit processing as well as the SR approach as we increase the number of monitored sources from 10 to 20. The SR approach does not handle the increasing number of monitored data updates effectively. The SLIM approach shows that interaction of spatial and non-spatial data attributes further enhances system scalability. SLIM outperforms in-transit processing by a margin of around 25 times and the SR approach by a margin of 7-10 times. Due to interaction between attributes from different data sources, the SLIM approach exhibits scalability even as we increase the number of data sources. The SR approach fails to exploit the effects of this interaction between multiple attributes.

Figure 8(b) displays the effect of increasing the number of users and triggers (both from 10,000 to 30,000 with 1% of the triggers as public) in the system while keeping the number of monitored sources constant at 10. Note that location update evaluation costs increase due to larger number of users for the INT approach. The SR approach is also scalable here as the monitored data evaluation costs remain the same. However it requires a large number of evaluations to be performed for the monitored data updates. SLIM is able to drop a large fraction of the monitored data updates and performs 25-45 times better than the INT approach. SLIM also performs 6-8 times better than the SR approach, the margin between the approaches decreases with increasing number of users and triggers.

4) Single (S) vs. Multiple (M) Safe Value Containers: This set of experiments provides a comparison of the performance of the two approaches for safe value container computation for single-dimensional data, namely, single and multiple safe value container computation. Figure 9(a) displays the number of messages processed by each approach as we increase the update period. At high data frequency (low update period), the gap between the single and multiple approaches is nominal. However, as we increase the update period, a significant gap appears between the number of updates processed by the approaches. This is due to the data values *jumping between* the multiple safe value containers leading to significantly larger number of dropped updates. Figure 9(b) displays the evaluation times for each approach as we vary the average update period. Note that the multiple safe value container

approach does not perform significantly worse than the single safe value container approach, although the container computation times for this approach are a little higher. Similar results are experienced as we increase the rate of change of data values [11].

VII. Related Work

An event-based location reminder system has been advocated by many human computer interaction projects [17], [18], [19]. In the realm of information monitoring, event-based systems have been developed to deliver relevant information to users on demand [20]. The SLIM system also needs to deal with the complexity of monitoring continuously changing data from multiple data sources in order to trigger relevant alerts in a non-intrusive manner. [21] provides a brief survey on continuous distributed monitoring. Different flavors of monitoring distributed data have been proposed [22], [23] but none aim to utilize interaction between multiple attributes. In the field of location-based queries, periodic reevaluation approach is commonly used for continuous monitoring of moving objects [24], [25]. The SLIM system differs from this work in two aspects. Firstly, our installed spatial triggers do not demand periodic evaluation or reevaluation like continuous queries; instead they require one-shot evaluation which should result in a notification being sent to the user when the trigger activation conditions are satisfied. Secondly, unlike the SLIM system, none of the previous work explores interaction between different attributes for information processing in mobile systems. Streaming data processing optimizations include batch processing methods [7] or approximation processing [6] to handle the large amounts of data. Fortunately, our problem defines a finite number of objects for which an infinite amount of continuously evolving information is being delivered. This makes it possible to store data corresponding to each object in the form of its safe value container.

VIII. Conclusion

We have presented the SLIM service as an efficient solution for the mobile information monitoring problem in presence of non-spatial attributes. This paper makes three important contributions towards efficiently solving the information monitoring problem in presence of spatial as well as non-spatial attributes. First, we show that the addition of less dynamic non-spatial attributes to the mobile information mix provides opportunities to enhance system scalability beyond what is possible with spatial attributes alone. Second, we use safe containment which allows us to perform selective processing of data updates by seeking cooperation from data sources in the information monitoring process but provides savings in communication costs. Last but not the least, we conduct extensive experimental evaluation for a real world road network-based simulator which shows that our approach makes the service scalable.

Acknowledgement

The last author is partially supported by grants from NSF CISE NetSE and SaTC program and an IBM faculty award.

References

[1] L. Golab and M. Özsu, "Issues in Data Stream Management," *ACM SIGMOD Record*, vol. 32, no. 2, pp. 5–14, 2003.
[2] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, "Query Processing, Approximation, and Resource Management in a Data Stream Management System," in *CIDR*, 2003.
[3] H. Hu, J. Xu, and D. Lee, "A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," in *ACM SIGMOD*, 2005, pp. 479–490.
[4] G. Iwerks, H. Samet, and K. Smith, "Continuous K-Nearest Neighbor Queries for Continuously Moving Points with Updates," in *VLDB*, 2003, pp. 512–523.
[5] X. Xiong, M. Mokbel, and W. Aref, "SEA-CNN: Scalable Processing of Continuous k-Nearest Neighbor Queries in Spatio-Temporal Databases," in *ICDE*, 2005, pp. 643–654.
[6] S. Viglas, J. Naughton, and J. Burger, "Maximizing the Output Rate of Multi-way Join Queries over Streaming Information Sources," in *VLDB*, 2003, pp. 285–296.
[7] H. Lim, J. Lee, M. Lee, K. Whang, and I. Song, "Continuous Query Processing in Data Streams using Duality of Data and Queries," in *ACM SIGMOD*, 2006, pp. 313–324.
[8] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, "Load Shedding in a Data Stream Manager," in *VLDB*, 2003, pp. 309–320.
[9] D. Comer, "The Ubiquitous B-tree," *ACM Computing Surveys*, vol. 11, pp. 121–137, 1979.
[10] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," in *ACM SIGMOD*, 1990, pp. 322–331.
[11] B. Bamba, K.-L. Wu, B. Gedik, and L. Liu, "Scalable Information Monitoring Using Distributed Spatial Triggers," Georgia Institute of Technology, Tech. Rep., 2012.
[12] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An Optimal and Progressive Algorithm for Skyline Queries," in *SIGMOD*, 2003.
[13] K. Tan, P. Eng, and B. Ooi, "Efficient Progressive Skyline Computation," in *VLDB*, 2001.
[14] B. Bamba, L. Liu, A. Iyengar, and P. S. Yu, "Safe Region Techniques for Fast Spatial Alarm Evaluation," Georgia Institute of Technology, Tech. Rep., 2008.
[15] "U.S. Geological Survey," http://www.usgs.gov.
[16] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *MobiSys*, 2003, pp. 31–42.
[17] P. Ludford, D. Frankowski, K. Reily, K. Wilms, and L. Terveen, "Because I Carry My Cell Phone Anyway: Functional Location-Based Reminder Applications," in *SIGCHI Conference on Human Factors in Computing Systems*, 2006, pp. 889–898.
[18] T. Sohn, K. Li, G. Lee, I. Smith, J. Scott, and W. Griswold, "Place-Its: A Study of Location-Based Reminders on Mobile Phones," in *UbiComp*, 2005.
[19] S. Kim, M. Kim, S. Park, Y. Jin, and W. Choi, "Gate Reminder: A Design Case of a Smart Reminder," in *Conference on Designing Interactive Systems*, 2004, pp. 81–90.
[20] L. Liu, C. Pu, and W. Tang, "WebCQ - Detecting and Delivering Information Changes on the Web," in *CIKM*, 2000, pp. 512–519.
[21] G. Cormode, "Continuous Distributed Monitoring: A Short Survey," in *First International Workshop on Algorithms and Models for Distributed Event Processing*, 2011, pp. 1–10.
[22] I. Sharfman, A. Schuster, and D. Keren, "A Geometric Approach to Monitoring Threshold Functions over Distributed Data Streams," *ACM Transactions on Database Systems*, vol. 32, no. 4, pp. 23–38, 2007.
[23] L. Huang, M. Garofalakis, J. Hellerstein, A. Joseph, and N. Taft, "Toward Sophisticated Detection with Distributed Triggers," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*. ACM, 2006, pp. 311–316.
[24] C. Jensen, D. Lin, and B. Ooi, "Query and Update Efficient B+-Tree based Indexing of Moving Objects," in *VLDB*, 2004, pp. 768–779.
[25] X. Yu, K. Pu, and N. Koudas, "Monitoring k-Nearest Neighbor Queries over Moving Objects," in *ICDE*, 2005, pp. 631–642.