Sindbad: A Location-based Social Networking System

Mohamed Sarwat*, Jie Bao*, Ahmed Eldawy*, Justin J. Levandoski†, Amr Magdy*, and Mohamed F. Mokbel*
*Dept. of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455
†Microsoft Research, Redmond, WA 98052-6399
*{sarwat,baojie,eldawy,amr,mokbel}@cs.umn.edu, †justin.levandoski@microsoft.com

ABSTRACT

This demo presents Sindbad; a location-based social networking system. Sindbad supports three new services beyond traditional social networking services, namely, *location-aware news feed*, *location-aware recommender*, and *location-aware ranking*. These new services not only consider social relevance for its users, but they also consider spatial relevance. Since location-aware social networking systems have to deal with large number of users, large number of messages, and user mobility, efficiency and scalability are important issues. To this end, Sindbad encapsulates its three main services inside the query processing engine of PostgreSQL. Usage and internal functionality of Sindbad, implemented with PostgreSQL and Google Maps API, are demonstrated through user (i.e., web/phone) and system analyzer GUI interfaces, respectively.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Design, Management, Performance, Algorithms

Keywords

Social Networking, Recommender Systems, News Feed, Spatial Rating, Spatial Message

1. INTRODUCTION

This demo presents Sindbad: a location-based social networking system built with an open-source database management system. Sindbad distinguishes itself from existing social networking systems (e.g., Facebook [4] and Twitter [10]) as it injects location-awareness within every aspect of social interaction and functionality in the system. For example, posted messages in Sindbad have inherent spatial extents (i.e., spatial location and spatial range) and users receive friend news feed based on their locations and the spatial extents of messages posted by their friends. The functionality of Sindbad is fundamentally different from current incarnations of location-based social networks (e.g., Facebook Places [5], Foursquare [7]). These existing location-based social networks are strictly built for mobile devices, and only allow users to receive messages about the whereabouts of their friends (e.g., Foursquare "check-ins" that give an alert that "your friend Alice has checked in at restaurant A"). Sindbad, on the other hand, takes a broader approach that marries functionality of traditional social-networks with location-based social scenarios (e.g., friend news posts with spatial extents, location-influenced recommendations) [3]. Thus, Sindbad is appropriate for both traditional social networking scenarios (e.g., desktop-based applications) as well as location-based scenarios (e.g., mobile-based applications).

Users of Sindbad can select their friend list as well as getting listed as friends to other users in a same way like traditional social networking systems. In addition, users can post (spatial) messages and/or rate (spatial) objects (e.g., restaurants), which will be seen by their friends. Once a user logs on to Sindbad, the user will see an incoming *location-aware* news feed posted by the user friends. Sindbad has a *location-aware news feed* module [2], named *GeoFeed*. For any user, GeoFeed efficiently retrieves the relevant messages from her friends based on the user location and the message spatial extents. Geofeed minimizes the total system overhead for delivering the location-aware news feed to the user while guaranteeing a certain response time for each user to obtain the requested location-aware news feed.

Sindbad users can receive *location-aware* recommendation about spatial items, e.g., restaurants, or non-spatial items, e.g., movies. To this end, Sindbad is equipped with a *location-aware recommender* module, called *LARS* [8]. For any user, LARS efficiently suggests (spatial) items based on users locations, items locations, and previous ratings by user friends. Sindbad produces recommendations by employing a user partitioning and travel distance penalty techniques in order to deliver relevant recommendations to its users. In addition to both GeoFeed and LARS, Sindbad adopts a *location-aware ranking* module that efficiently selects the top-k relevant objects produced from either the location-aware news feed or the recommender system modules.

A major part of Sindbad is built inside PostgreSQL; an open-source DBMS. Hence, Sindbad: (a) takes advantage of the scalability provided by the DBMS, and (b) is able to employ early pruning techniques inside the DBMS engine, which yields an efficient performance for the news feed, recommendation, and ranking functionalities. Moreover, Sindbad provides an RESTful [6] web API so that it would allow a wide variety of applications to easily communicate with Sindbad and make use of its unique features. The system is demonstrated using both a web and smart phone (i.e., Android) applications that we built on top of Sindbad particularly for demonstration purpose. Moreover, the system internals are demon-

SIGMOD '12, May 20–24, 2012, Scottsdale, Arizona, USA.

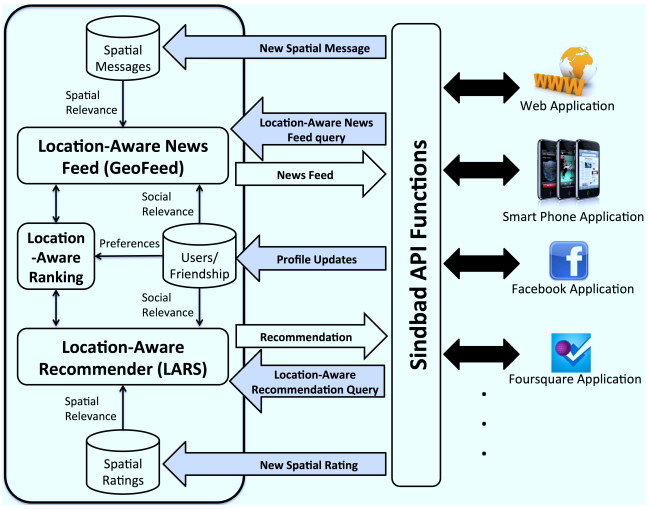

Figure 1: Sindbad System Architecture.

strated through an administrator-like interface that shows the different system parameters as well as general statistics about Sindbad.

The rest of the paper is organized as follows: Section 2 gives the system architecture for Sindbad. The main three components of Sindbad, namely, *location-aware news feed (GeoFeed)*, *location-aware recommender system (LARS)*, and *location-aware ranking*, are discussed in Sections 3, 4, and 5, respectively. Finally, the demonstration scenario is described in Section 6.

2. SINDBAD ARCHITECTURE

Figure 1 depicts the Sindbad system architecture that consists of three main modules, namely, *location-aware news feed* (GeoFeed), *location-aware ranking*, and *location-aware recommender* (LARS), and three types of stored data, namely, spatial messages, user profiles, and spatial ratings. The communication between Sindbad and the outside world is held through RESTful web API interface (named *Sindbad API Functions* in Figure 1). The API functions facilitates building a wide variety of applications (e.g., web applications, smart phone applications) on top of Sindbad. As shown in Figure 1, Sindbad API functions can also be used to complement the functionality of existing social networking websites e.g., Facebook, and turn their news feed and recommendation to be location-aware. Sindbad can take five different types of input (i.e., through the API interface): profile updates, a new message, a new rating, a location-aware news feed query, and a location-aware recommender query. The actions taken by Sindbad for each input is described as follows:

Profile updates. As in typical social networking systems, Sindbad users can update their personal information, their friend list, or accept a friend invitation from others.

A new message. Users can post spatial messages to be seen by their friends, if relevant. A spatial message is represented by the tuple: (*MessageID*, *Content*, *Timestamp*, *Spatial*), where *MessageID* and *Content* represent the message identifier and contents, respectively, *Timestamp* is the time the message is generated, while *Spatial* indicates the spatial range for which the message is effective. The message is deemed relevant to only those users who are located within its spatial range.

A new rating. Sindbad users can give location-aware (spatial) ratings to various items in a scale from one to five. Location-aware (spatial) ratings can take any of these three forms: (1) *Spatial ratings for non-spatial items*, represented as a four-tuple (*user*, *userLocation*, *rating*, *item*); for example, a user located at home rating a book, (2) *Non-spatial ratings for spatial items*, represented as a four-tuple (*user*, *rating*, *item*, *itemLocation*); for example, a user with unknown location rating a restaurant with an inherent location, and (3) *Spatial ratings for spatial items*, represented as a five-tuple (*user*, *userLocation*, *rating*, *item*, *itemLocation*); for example, a user at his/her office rating a restaurant with an inherent location.

Location-aware news feed queries. Once a Sindbad user logs on to the system, a location-aware news feed query is fired to retrieve the relevant news feed, i.e., messages posted by the user's friends that have spatial extents covering the location of the requesting user. Details of the execution of the location-aware news feed query will be discussed in Section 3. The output of the *location-aware news feed* module (GeoFeed) will be processed further by the *location-aware ranking* module to get only the top-k news feed based on the spatial and social relevance, which will be returned to the user as the requested news feed. Details of the *location-aware ranking* module will be described in Section 5.

Location-aware recommendation queries. Sindbad users can request recommendations of either spatial items (e.g., restaurants, stores) or non-spatial items (e.g., movies) by explicitly issuing a location-aware recommendation query. The *location-aware recommender* module (LARS) suggests a set of items based on: (a) the user location (if available), (b) the item location (if available), and (c) ratings previously posted by either the user or the user's friends. Details of LARS will be discussed in Section 4. Similar to location-aware news feed queries, the output of LARS goes through the *location-aware ranking* module to select only the top-k items based on both spatial and social relevance.

3. LOCATION-AWARE NEWS FEED

Although news feed functionality is widely available in all social network systems [9], these systems select the relevant messages either based on the message timestamp or some importance criteria that ignores the spatial aspect of posted messages. Thus, users may miss several important messages that are spatially related to them. For example, when a traveling user logs on to a social network site, the user would like to get the news feed that match his/her new location, rather than receiving the most recent (non-spatial) news feed. The same concept applies for users who continuously log onto the system from the same location, yet have a large number of friends. It is of essence for such users to limit their news feed to the messages related to their location. Examples of the location-aware news feed returned by Sindbad include a message about local news, a comment about a local store, or a status message targeting friends in a certain area.

The main idea of the *location-aware news feed* module (GeoFeed) is to abstract the location-aware news feed problem into one that evaluates a set of location-based point queries against each friend in a user's friend list that retrieves the set of messages issued that overlap with the querying user's location. The *location-aware news feed* is equipped with three different approaches for evaluating each location-based query: (1) *spatial pull approach*, in which the query is answered through exploiting a spatial index over the messages posted by the friend, (2) *spatial push approach*, in which the query simply retrieves the answer from a pre-computed materialized view maintained by the friend, and (3) *shared push approach*, in which the pre-computation and materialized view maintenance are shared among multiple users. Then, the main challenge of GeoFeed is to decide on when to use each of these three approaches for a query.

A better response time calls for using the *spatial push* approach for all location-aware news feed queries issued to Sindbad. In this case, all location-aware news feed are pre-computed. However,

this approach results in tremendous system overhead since a massive number of materialized views must be maintained. On the other hand, favoring system overhead may result in executing more queries using the *spatial pull* approach as no views needs to be maintained. However, this approach may result in a long query response time for users who have a large number of friends, since they will suffer a long delay when retrieving their news feed. Sindbad takes these factors into account when deciding on which approach to use to evaluate each query in a way that minimizes the system overhead and guarantees a certain user response time. Sindbad is equipped with an elegant decision model that decides upon using these approaches in a way that: (a) minimizes the system overhead for delivering the location-aware news feed, and (b) guarantees a certain response time for each user to obtain the requested location-aware news feed. More details about GeoFeed decision model are provided in [2].

4. LOCATION-AWARE RECOMMENDER

In general, recommender systems make use of community opinions to help users identify useful items from a considerably large search space (e.g., Amazon inventory, Netflix movies). Community opinions are expressed through explicit ratings represented by the triple (*user*, *rating*, *item*) that represents a user providing a numeric rating for an item (e.g., movie). Unfortunately, ratings represented by this triple ignore the fact that both users and (some) items are spatial in nature. Unlike traditional recommendation techniques that assume the (non-spatial) rating triple (*user*, *rating*, *item*), the *location-aware recommender* module (LARS) in Sindbad supports a taxonomy of three types of location-based ratings: (1) *Spatial ratings for non-spatial items*, represented as a four-tuple (*user*, *ulocation*, *rating*, *item*), where *ulocation* presents the user location, (2) *Non-spatial ratings for spatial items*, represented as a four-tuple (*user*, *rating*, *item*, *ilocation*), where *ilocation* presents the item location, and (3) *Spatial ratings for spatial items*, represented as a five-tuple (*user*, *ulocation*, *rating*, *item*, *ilocation*), where *ulocation* and *ilocation* present the user and item locations, respectively.

Sindbad produces recommendations using *spatial user ratings for non-spatial items* by employing a user partitioning technique that exploits the user location embedded in the ratings. This technique uses an adaptive pyramid structure to partition ratings by their user location attribute into spatial regions of varying sizes at different hierarchies. Then, for a querying user located in a region $\mathcal{R}$, we apply an existing collaborative filtering technique [1] that utilizes only the ratings located in $\mathcal{R}$. The challenge, however, is to determine whether all regions in the pyramid must be maintained in order to balance two contradicting factors: scalability and locality. Maintaining a large number of regions increases locality (i.e., recommendations unique to smaller spatial regions), yet adversely affects system scalability because each region requires storage and maintenance of a collaborative filtering data structure (i.e., model) necessary to produce recommendations. The pyramid dynamically adapts to find the right pyramid shape that minimizes storage overhead, i.e., increases scalability, without sacrificing locality.

Sindbad produces recommendations using *non-spatial user ratings for spatial items* by employing a travel penalty technique that accounts for item locations embedded in the ratings. This technique favors recommendation candidates the closer they are in travel distance to a querying user. Sindbad employs an efficient query processing framework capable of terminating early once it discovers that the list of recommended items cannot be altered by processing more candidates. Finally, to produce recommendations using *spatial ratings for spatial items*, Sindbad employs both the user partitioning and travel penalty techniques to address the user and item locations associated with the ratings. More details about LARS are given in [8].

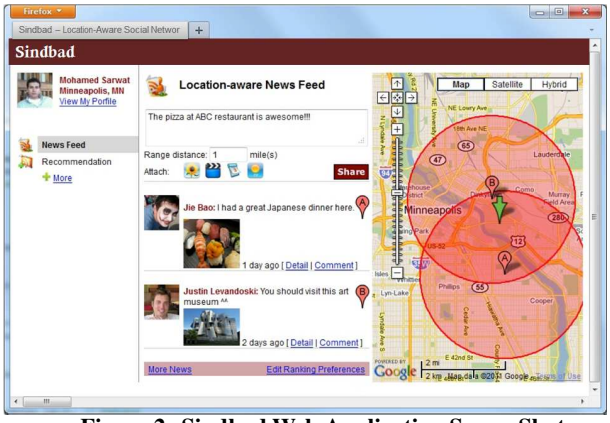

Figure 2: Sindbad Web Application ScreenShot

5. LOCATION-AWARE RANKING

Sindbad users may have different preferences over messages from the location-based news feed or recommender system. For example, a traveling user may be more interested in messages that were issued close to her current locations. On the other hand, the stationary user maybe more interested in the most recently issued messages. Moreover, due to the large volume of messages submitted to Sindbad and the user's limited viewing capability (e.g., 40 messages for the web page and 20 messages for mobile application), Sindbad provides a *location-aware ranking* module that is responsible for ranking the results coming out of the *location-aware news feed* module and *location-aware recommender* module, based on the user's preferences.

Instead of ranking all objects, Sindbad *location-aware ranking* module encapsulates the user's ranking preferences within the query processor to improve the response time for the user. Moreover, if the user is continuously online, Sindbad location-aware module is responsible for keeping either the news feed or the recommended items correctly ranked (i.e., continuously evaluating the ranking function) as the user moves or if the user changes her preferences.

6. DEMONSTRATION SCENARIO

We hooked up Sinbad with Foursquare (by means of Foursquare APIs) so that the demo attendee can use his/her existing Foursquare account (if they already have one) to interact with Sindbad. In case the demo attendee is not already registered as a Foursquare user or does not want to sign up in Sindbad, we have prepared several preset user accounts for demonstration purpose. Moreover, we have collected data (e.g., restaurants, Foursqure users preferences...) for the city of Scottsdale in Arizona, where SIGMOD takes place this year. This will increase the interaction with the demo attendees by providing location-aware news feed to SIGMOD attendees or recommending them locations (e.g., restaurants, bars...) in Scottsdale.

Web/Phone Application. We demonstrate Sindbad using both web (see Figure 2) and Android phone (see Figure 3) interfaces which communicate with the system using Sindbad API Interface. Both applications are integrated with the Google Maps API, and can be accessed through standard web browsers. The demo attendee can choose to log onto Sindbad using one of several preset user accounts, or using his own existing Foursquare account, or even by creating a new user account on the fly for the demonstration purpose. When the user logs on to Sindbad, the application displays

(a) Location-Aware News Feed (b) Location-Aware Recommender

Figure 3: Sindbad Android Phone Application ScreenShots

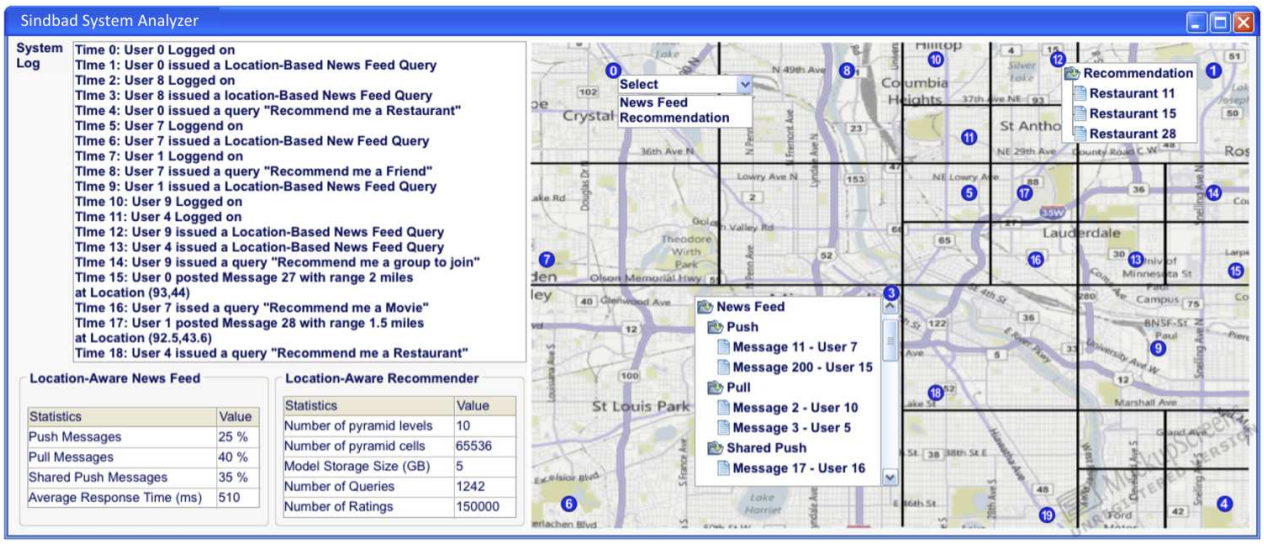

Figure 4: Sindbad System Analyzer ScreenShot

the relevant news for the user on the map. Each message is associated with a circle representing the range of each message. The demo attendee may "change" locations by dragging and dropping the green arrow on the map, to see how the news feed will change accordingly. The user may also submit a geo-tagged message to the system. For instance, the user shares a message "The pizza at ABC restaurant is awesome" with a range distance of one mile. Moreover, The demo attendee can also ask Sindbad for recommendations (e.g., Restaurants in Scottsdale where SIGMOD takes place) by clicking on the recommendation link on the left-hand side of the web interface or by clicking on the location-aware recommendation button in the mobile app. The user then enters the type of object he is interested in (e.g., restaurant, theaters, stores). a spatial range in miles, and also the number of recommended items to be returned to him and then presses the *Recommend* button (see Figure 3(b)). The recommended items are then shown on the map.

Sindbad system analyzer. The Sindbad analyzer shows the system behavior while different operations are applied to Sindbad (see Figure 4). The GUI shows a map of the city of Minneapolis, in which users that are currently logged in and are moving in the area, are shown on the map in blue circles. The demo attendee can select a user to track by clicking on the corresponding blue circle. At this point, a drop down menu pops up giving two options: (1) News feed: when selecting this option, a location-based news feed query is issued to Sindbad, and the result is shown as a list of messages; each message is associated with the user friend that posted the message as well as the approach used to retrieve this message (i.e., pull, push, shared-push) (2) Recommendation: when selecting this option, a location-based recommendation query is issued to Sindbad, and the result is shown as a list of items recommended to the user. When the user moves, the news feed and recommendation results are updated accordingly. The analyzer also shows a log of Sindbad operations (e.g., user logins, queries, posted messages). The analyzer gives some statistics about GeoFeed (e.g., percentage of push, pull, or shared-push messages) and LARS modules (e.g., number of ratings, number of queries, and number of pyramid levels).

7. ACKNOWLEDGEMENT

This work is supported in part by the National Science Foundation under Grants IIS-0811998, IIS-0811935, CNS-0708604, IIS-0952977 and by a Microsoft Research Gift.

8. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *TKDE*, 17(6):734–749, 2005.

[2] J. Bao, M. F. Mokbel, and C.-Y. Chow. GeoFeed: A Location-Aware News Feed System. In *ICDE*, 2012.

[3] C.-Y. Chow, J. Bao, and M. F. Mokbel. Towards Location-based Social Networking Services. In *ACM SIGSPATIAL-LBSN*, 2010.

[4] Facebook. http://www.facebook.com/.

[5] The Facebook Blog, "Facebook Places": http://tinyurl.com/3aetfs3.

[6] R. T. Fielding and R. N. Taylo. Principled Design Of The Modern Web architecture. In *ICSE*, 2000.

[7] Foursquare: http://foursquare.com.

[8] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. LARS: A Location-Aware Recommender System. In *ICDE*, 2012.

[9] A. Silberstein, J. Terrace, B. F. Cooper, and R. Ramakrishnan. Feeding Frenzy: Selectively Materializing User's Event Feed. In *SIGMOD*, 2010.

[10] Twitter. http://www.twitter.com/.