NSF Workshop on Social Networks and Mobility in the Cloud

Arlington, VA

February 23, 2012 - February 24, 2012

Workshop Participants:

Amr El Abbadi, UC Santa Barbara
Divyakant Agrawal, UC Santa Barbara
Sihem Amer-Yahia, QCRI, Qatar
Walid Aref, Purdue University
Michael J. Carey, UC Irvine
Panos K. Chrysanthis, University of Pittsburgh
Anhai Doan, University of Wisconsin and @WalmartLabs
Sameh Elnikety, Microsoft Research
Christos Faloutsos, Carnegie Mellon University
John Greer, National Geospatial-Intelligence Agency
Le Grunewald, NSF
Hakan Hacigümüs, NEC Labs
Yuxiong He, Microsoft Research
Panos Ipeirotis , New York University
James Kang , National Geospatial-Intelligence Agency
Tim Kraska, UC Berkeley
Alexandros Labrinidis, University of Pittsburgh
Laks V.S. Lakshmanan, University of British Columbia
Monica S. Lam, Stanford University
Wang-Chien Lee, Pennsylvania State University
Kristen LeFevre, Google
Justin J. Levandoski, Microsoft Research
Jimmy Lin, University of Maryland College Park
Ashwin Machanavajjhala, Yahoo! Labs
Vahab Mirrokni, Google
Alan Mislove, Northeastern University
Mohamed Mokbel, University of Minnesota
Mario Nascimento, University of Alberta
Suman Nath, Microsoft Research
Jennifer Neville, Purdue University
Raghu Ramakrishnan, Yahoo! Research
Jagan Sankaranarayanan, NEC Labs
Cyrus Shahabi, University of Southern California
Shashi Shekhar, University of Minnesota
Evimaria Terzi, Boston University
Raju Vatsavai, Oak Ridge National Laboratory
Cong Yu, Google

Organization Committee:

Program Chairs:

Amr El Abbadi, UC Santa Barbara
Christos Faloutsos, Carnegie Mellon University
Mohamed Mokbel, University of Minnesota

Steering Committee:

Sihem Amer-Yahia, QCRI, Qatar
Sameh Elnikety, Microsoft Research
Hakan Hacigümüs, NEC Labs
Suman Nath, Microsoft Research
Jagan Sankaranarayanan, NEC Labs
Cong Yu, Google

Table of Contents

Infocosm: Towards Collective Decision Making in Mobile Social Networks

Divyakant Agrawal
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106-5110, USA
agrawal@cs.ucsb.edu

ABSTRACT

As Internet based services and mobile computing become ubiquitous, society becomes increasingly reliant on these communication media to accomplish critical and time-sensitive tasks such as information dissemination and collaborative decision-making. The dependence on these media magnifies the damage caused from their disruption, whether by malicious intent or natural disaster. For instance, in the event of a natural disaster, such as the earthquake in Haiti or the tsunami in Japan, disruption of the centralized mobile and Internet infrastructures impedes information spread and often leads to chaos, both amongst the victims as well the aid providers. Decentralized and ad-hoc mechanisms for information dissemination and decision-making are paramount to help restore order. We propose Infocosm, a mobile social network that utilizes direct device communication to enable group decision-making, or consensus, without reliance on global communication services. Infocosm focuses on minimizing the system resources, to prolong the lifetime of the power constrained devices, by minimizing communication overhead, computational complexity, and persistent storage size. Infocosm provides a simple and intuitive system to enable large-scale coordination amongst non-expert users. Due to the mobility of the users in Infocosm, limited range of point-to-point communication, and the ad-hoc nature of the infrastructure, all the participants in the system cannot communicate with each other. Estimating the number of participants (or a global count) itself becomes a challenge and hence, traditional notions of consensus or quorum based protocols for agreement cannot be used. We, therefore, rely of threshold and time limit based approaches to reach an agreement. In this presentation, we will explore various heuristics and models to estimate group participation to aid users in reconciling consensus.

1. MOTIVATION

From Tahrir Square to Wall Street, new technologies, such as social networks and mobile computing devices, are enabling people to quickly organize in a decentralized manner. Social networks are unintentionally serving as groupware to synchronize and facilitate human interactions [4]. The phenomenon of information diffusion and influence in social networks has been the interest of recent research and modeling. In an abstract sense, all popular social networks enable a user to express an idea and subsequently propagate the idea through a network of peers. This straightforward diffusion of information and the simple interface has enabled people to organize in a lightweight manner which is essential to facilitate large scale group interaction. However, this approach is not entirely decentralized, as these tools rely on Internet services to act as a centralized coordinator for user messages throughout a network. In the event of a natural disaster or an administration turning adversarial to curb a movement, Internet access may become limited or unavailable. Several recent events demonstrate this scenario. With the 2010 earthquake in Haiti, rescue works relied on text messaging (SMS) to coordinate efforts, as cell phone networks strained under failure and overloading [6, 7]. Planned protests at the Bay Area Rapid Transit (BART) in San Francisco, USA resulted in cellular service being cut in order to stave off protests [1]. Lastly, and perhaps most infamously, Hosni Mubarak's government shut down Internet access to Egypt in an attempt to thwart not only social network coordination, but also privacy filters, such as Tor, that bypass censorship firewalls [3].

Despite the availability of a centralized service, certain actions still require coordination of a large group, whether it be a protest of conditions or organizing humanitarian efforts. Without a single point of communication, planned actions can become disjoint and unclear, and result in a reduced effectiveness. We define a problem in the described environment as how can a system disseminate an idea, or proposal, amongst a network of peers in order to ascertain user's intention and determine an expected outcome. As knowledge and context of the proposal is required to determine success, a proposer should specify a tipping point (*quorum value*), which is the number of users that need to agree in order to achieve a consensus. In this context, we use consensus as the consent of a specified group size for a proposed value, and not the stringent definition of consensus where all non-faulty processes agree on a single value. Due to the lack of a central authority and due to the mobility of the users, the number of users that will observe a given ballot is unknown. As a result, classical notions of quorums, group size estimation, or consensus in a static distributed system [8, 12] cannot be applied directly in this problem setting. Since a consistent view of the number of users that will observe a proposal is not known, an application specific quorum value threshold is needed to approximate consensus. Quorum value is application specific since the number of people required to organize a meeting at a database conference is very different compared to the number required to effectively organize a city protest.

The group decision making problem framed in a disconnected environment becomes technically challenging. Users that are mobile, are likely to have different views on the state of the ballot due to the lack of a single point of truth and observing intentions of disjoint peers. Divergent views need to be reconciled between users, so an approximate view can be consistently determined in order to have similar state when a proposal expires. The proposals may be critical for users, so the state of the proposal should be persisted beyond volatile memory. *The reconciliation of disconnected replicas,* *efficient persistent storage, managing concurrent proposals, and an understanding of consensus call for a database system solution to this timely problem*.

The reliance on a centralized cellular or Internet access does not prevent many modern mobile devices from communicating with its peers at a large scale. Many smartphones have capabilities to directly communicate with other mobile devices within a limited range, including IEEE 802.11, Bluetooth, or Ultra Wide Band. These communication media provide the ability to discover peers within a few hundred feet allowing for the construction of a mobile peer-to-peer (**P2P**) network to exchange information [12]. Leveraging these networks and motivated by the need for decentralized organizational tools, we introduce Infocosm, a mobile P2P database that enables group decision making without relying on centralized services. While mobile P2P databases for disaster situations or military applications have been proposed before, specifics were not provided. Moreover, our focus is on group decision making which is beyond communication overlays [11]. The name Infocosm symbolizes our goal to create an aggregate global view of information composed of smaller pieces of information that can be difficult to piece together and individually cannot represent the global picture. In this presentation we will highlight the need for Infocosm and the research challenges associated with enabling group decision making in a P2P mobile environment. We will also present a brief overview of a research prototype that is being developed at UC Santa Barbara to highlight some of the implementation challenges in developing mobile social network technologies.

2. PRELIMINARIES

Infocosm is a database system running on mobile devices that communicates in a P2P fashion with other mobile devices in its vicinity. In Infocosm, a user can propose a question (the **proposal**), optionally accompanied by a suggested answer (the **value**). The proposal is broadcast to all users within the proposer's direct communication range. Since the communication range is limited and the users are mobile, a proposal might eventually be relayed to areas where the original user initiating the proposal is not present. We use the term **proposer** for the user that introduces the ballot to a set of peers who have not received the proposal earlier. The proposer can therefore be the original initiator or a relay node. The mobile agents forward this proposal, allowing users to agree with the proposal, suggest a new value, or reject the proposal. New values are only allowed if none of the proposed valued have reached the tipping point. The users that respond non-negatively to a proposal are **subscribers** to the proposal. Every proposal has an associated **expiration time** after which it is no longer valid; a proposal is spread until its expiration time is reached. The proposal is encapsulated within a **ballot** that contains the proposal, the suggested value, the expiration time, the proposer's unique ID, the minimum number of users to achieve a consensus, and the set of users who have accepted each of the ballot's potential values. All subscribers are notified of the proposal's **outcome** either when a value reaches the tipping point or when the proposal expired. The outcome of a proposal is the values which received votes above a predetermined threshold. Since Infocosm is designed to operate in disconnected modes where all users are mobile and a centralized single point of truth for the ballot may not exist where data is incomplete, the vote counts are associated with error bounds.

Figure 1 describes a sample ballot for the proposal. The ballot is expressed in JSON format for the ease of exposition; Infocosm stores and transmits a ballot in a compact compressed binary format. In this example the proposer suggests that at least twenty participants are required to achieve the tipping point. The

```
ballot{
 proposal: "Gather to demand face-to-face
            PC meetings for SIGMOD",
 expiration: "2012-05-24 10:00:00",
 suggestedValue: "Canyon Room 5/24 11:00",
 quorumValue: 20,
 userAccepts: {
   "Canyon Room 5/24 11:00" : [31083,
      13091, 38919, 900941, 109381],
   "Mesa Hall 5/24 14:00" : [13134]
 }
 key: 107074168843,
 proposerId: 31480
}
```

Figure 1: A sample ballot to demand conference review changes.

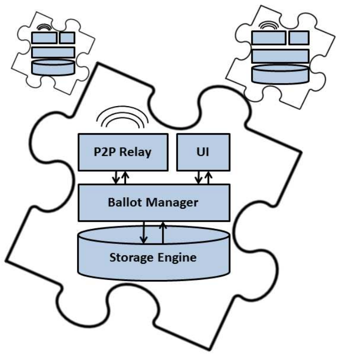


Figure 2: Internal Infocosm Components.

snapshot of Figure 1 captures a scenario where an alternative value has also been proposed. The ballot lists both the proposed values: `Canyon Room at 11 am` that has six acceptors and `Mesa Hall 5/24 14:00` has a single vote. For brevity, the error values and divergence in versions is not shown, but Infocosm tracks it internally. The ballot is broadcast to all peers within range. Each peer with an active Infocosm instance will notify its user of a new ballot. The user can take action on the ballot by either *accepting* the ballot, *relaying* it, *proposing* a new value, or *ignoring* the ballot. The proposer for this ballot is notified of this user's intention. An *accept* signifies that the peer agrees with the proposal, and will relay the proposal to all future peers. A *relay* signifies that the peer will not commit to the proposed value but will subscribe to the ballot and rebroadcast to future peers. If the peer disagrees with the suggested value of the ballot, an alternate value can be proposed. *Ignore* states that the user rejects the ballot, and that Infocosm should ignore all future messages regarding this ballot. The proposer adds all accepted users and proposed values to the ballot, makes the ballot locally persistent in Infocosm, and rebroadcasts the updated ballot to all users in range. On receipt of a message, a peer determines if the ballot contains new information or should be ignored. Additionally, a peer does not rebroadcast a ballot with an identical state (i.e. no new acceptors or values) to peers already aware of the ballot.

3. SYSTEM DESIGN

Infocosm is composed of four major components shown in Figure 2. The *storage engine* provides persistence for the observed ballots and the actions taken by the user. The *P2P relay* notifies the ballot manager of new peers observed, incoming ballots, and ballot proposal responses. The relay also manages proposal broadcasts and responses from the ballot manager. The *UI* module interfaces the ballot manager with the user to acquire responses about proposals and notify with updates on the subscribed ballots. The *ballot manager* coordinates all components, decides what information to relay between components, ignore and broadcast ballot information to new peers, and inform the user about the status and outcome of a ballot.

Several challenges arise in designing an efficient ballot manager. Battery life, network bandwidth, and storage capacities are limited and many optimizations are needed to minimize Infocosm's footprint and resource consumption. An instance of Infocosm on each node stores detailed information about a ballot so that the ballot manager can appropriately orchestrate coordination between the components. However, since network communication is expensive, peers only exchange a compressed ballot header. This compressed header should be sufficient to determine if additional information is needed to synchronize the ballot views of two nodes.

The frequency at which ballots are transmitted must also be optimized. While a great deal of literature exists on gossip protocols and the properties of epidemic communication in mobile environments, additional context can be leveraged in communication protocols [12]. This context can include the expiration time, the mobility patterns, and popularity of a proposal. Mobility patterns will be especially important when a small subset of the peers are mobile, while majority remain static and are less likely to interact with new peers.

Due to the decentralized nature, timestamps alone cannot accurately determine if peers have a consistent view of a ballot. As users move and are disconnected, their versions of the ballot might diverge. As a result, Infocosm must address important research challenges in reconciliation of the divergent ballots, and their respective counts, eliminating duplicates from the counts, as well as tracking the lineage of the ballot versions as they diverge. For instance, a user U_j receives the ballot from another user U_i. After accepting the proposal, U_j moves to the vicinity of U_k and passes the ballot. Now, if U_k moves into the vicinity of U_i, U_i must be able to eliminate the duplicate counts through the lineage of divergent versions. Storing and exchanging the full accepted user set to track lineage is expensive in terms of bandwidth and computation. A naïve optimization involves comparing a hash an ordered set of users (such as bloom filters); however, this approach incurs a high cost to merge large sets and is also an approximate set membership. Additionally, storing and exchanging large sorted sets can be too costly for storage and computation on a device where resources are finite. We are currently exploring using a combination of approaches, using probabilistic data structures, such as bloom filters, to determine set membership quickly, or utilize multi-sets for quick merges and rely on sampling or sketch based techniques to estimate of set membership [5, 9, 13]. Alternatively, Infocosm can leverage de-duplication techniques for comparing large sets and quickly identifying differences in the data set [2]. Lastly, if a large set of peers have frequent interactions, such as conference attendees, a coordinated checkpoint can also be constructed.

Most importantly, Infocosm should also be able to reason about the confidence of consensus beyond ballots reaching a designated threshold of votes. However, due to the uncertain nature of the network topology and communication, true consensus cannot be achieved due to the impossibility of agreement and validity. Often traditional distributed systems notions, like consistency, assume a static number of nodes. Many of these assumptions were reexamined with a rise in popularity of dynamic systems, where the number of nodes varies over time. Research into *Group Size Estimation* and dynamic system modeling will guide efforts into building robust models of consensus for a mobile environment [8, 10]. Since the mobile agents differ from dynamic systems with spatial and temporal patterns, Infocosm considers empirical observations about the mobility of a user and the churn and mobility of peers when modeling consistency. Finally, a feedback mechanism on the outcome is requested from the proposer, in order to reinforce models that accurately stated which proposed value achieved consensus.

In addition to the above mentioned challenges, Infocosm must also address multiple issues, such as privacy of the users, potentially intermittent centralized sources of truth, malicious behavior, and trustworthiness. Discussion of all these challenges and possible approaches to overcome these challenges will be the subject of our presentation at the workshop.

4. REFERENCES

[1] Washington Post: BART San Francisco cut cell services to avert protest. `http://wapo.st/tMomtM`, 2011.

[2] D. Bhagwat, K. Eshghi, D. D. E. Long, and M. Lillibridge. Extreme binning: Scalable, parallel deduplication for chunk-based file backup. In *MASCOTS'09*, pages 1–9, 2009.

[3] The Telegraph : How Egypt shut down the internet. `http://tgr.ph/vidllq`, 2011.

[4] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some issues and experiences. *Commun. ACM*, pages 39–58, 1991.

[5] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31:182–209, September 1985.

[6] IEEE Spectrum : Why Haiti's Cellphone Networks Failed. `http://bit.ly/tnud7v`, 2010.

[7] Reuters : Cell phones and radios help save lives after Haiti earthquake. `http://reut.rs/vU1gMh`, 2010.

[8] D. Kostoulas, D. Psaltoulis, I. Gupta, K. P. Birman, and A. J. Demers. Active and passive techniques for group size estimation in large-scale and dynamic distributed systems. *J. Syst. Softw.*, pages 1639–1658, October 2007.

[9] A. Metwally, D. Agrawal, and A. El Abbadi. Why go logarithmic if we can go linear?: Towards effective distinct counting of search traffic. In *EDBT*, pages 618–629, 2008.

[10] A. Mostéfaoui, M. Raynal, C. Travers, S. Patterson, D. Agrawal, and A. El Abbadi. From static distributed systems to dynamic systems. In *SRDS*, pages 109–118, 2005.

[11] S. Shekhar and H. Xiong, editors. *Encyclopedia of GIS*. Springer, 2008.

[12] O. Wolfson, B. Xu, H. Yin, and H. Cao. Search-and-discover in mobile p2p network databases. In *ICDCS*, pages 65–, 2006.

[13] K. Wu, Y. Xiao, J. Li, and B. Sun. A distributed algorithm for finding global icebergs with linked counting bloom filters. In *ICC'08*, pages 2757–2761, 2008.

Managing and Sharing Experiences on the Social Web

Sihem Amer-Yahia
Qatar Computing Research Institute, Doha, Qatar
syahia@qf.org.qa

NSF Workshop on Social Networks and Mobility in the Cloud
http://dmlab.cs.umn.edu/SocialMobileCloud/

Abstract

Mobile phones have transformed the ways people interact. Last minute plans for physically getting together are made possible by the ability to call each other, check each others' whereabouts and decide to meet up. A recent generation of mobile applications is having similar effects on virtual social networks. Those applications combine a user's social network and location to encourage socializing, provide recommendations in a given geographic area, and engage users in online and offline activities. With that, instant, unplanned and useful socializing has become possible. The social breadcrumbs gathered in that process can be used to build a rich repository of geo-tagged information and personal preferences and leveraged to offer new online experiences that go beyond *atomic content consumption.* In this position paper, we discuss opportunities and challenges that arise when managing and sharing such experiences.

1 Introduction

Hundreds of location-based social networks, services and advertising applications are being proposed on mobile devices. Those applications exploit geographic co-location and social networks to encourage socializing and content consumption. Yet, content recommendation solutions tend to focus on suggesting one piece of content (e.g., news articles, photos, travel destinations, products) that is endorsed by a large-enough number of users. In a world of mobility and virtual social networking, new recommendation opportunities that go beyond mainstream solutions are appearing. In this position paper, we propose a paradigm shift for social search and recommendation that makes use of the millions of social breadcrumbs left online by users and their social network, to *recommend experiences* instead of individual content items.

The list of applications that leverage geo-location and social networking keeps growing. For example, *Aka-aki* shows common friends and interests on a mobile display and also from people in one's neighbourhood, city or region. *Ask Around* and *lockChalk* let a user view, join and share real-time conversations happening nearby. *liin* enables mobile and online location-aware content, community and commerce. *Poki* lets one find out where friends are and track them on Google Maps and Google Earth. *blumapia* is a boating mobile social network with geo-tagged content sharing including photos. *flaik* combines location-based services with social networking, to pinpoint the location of individual skiers, deliver daily run statistics, and enable skiers to share their day with friends and families. *Foodspotting* allows users to share recommendations by taking a picture of their food, saying what it was and where they found it. *Trapster* users share the location of police speed traps. It uses the phone's GPS and Internet to alert users as they approach reported traps.

Social breadcrumbs left by users' actions and events together form their experience. In a mobile environment, breadcrubms are geo-located and time-stamped and can thus be gathered, stored, queried and recommended with different semantics. Moreover, making those experiences readily available will enable sharing and discussing them with others thereby engaging users more socially and helping them refine their own experiences and extract values from others'.

2 Challenges

Shifting the focus from managing and recommending atomic content items to experiences raises a number of challenging questions. First, the notion of experience requires to *gather and connect the actions* undertaken by a large number of users during a time period. Second, the relevance of an experience to a user needs to account not only for the user's current activities (e.g., recommending a swimming pool after a hairdresser's is pointless), the user's affinities with others (e.g., gathering shoestore recommendations from certain friends can be useless), but also the time of day, and the user's geographic radius. Finally, experiences could be queried and explored in different ways. We argue that ranked lists are not the best paradigm for exploring others' experiences and discuss alternative explorations.

2.1 Gathering Experiences

Gathering experiences requires to continuously aggregate the paths of a large number of users. The sheer volume of data generated by individual users raises new opportunities for indexing and compressing such data. User paths have two unique characteristics: location and time. Indexing partial paths by users in the same geographic area and for different time intervals will speed up retrieval. In [1], we explored multiple user clusterings based on overlap in actions and social networks (in that work, an action represented a user tagging a URL in Delicious). This clustering needs to be revisited to account for co-location.

2.2 Experience Relevance

Different experiences appeal to different users. Moreover, the relevance to an experience to a user depends on the place, time and affinities of a user with those who "own" that experience. An interesting experience may be one formed by a set of actions none of which a user experienced before or one that overlaps with previous experiences a user liked. Hence, ranking experiences requires to re-think relevance to incorporate these new dimensions.

2.3 Finding Experiences

Several querying models can be designed to inquire about existing experiences at different times and places. We discuss three different approaches that were developed in different contexts and identify the challenges behind adapting them to the problem of finding experiences.

In [2], we used a graph model to represent user itineraries in a city that were extracted from Flickr photos taken by them at different points of interest (POIs) in the city. User photo streams were aggregated into a POI graph that became readily available for querying. We proposed an adaptation of the Orienteering algorithm to construct itineraries given a start point, an end point and a time budget. This approach could be adapted to account for time of day and user affinities when determining the relevance of an experience to a user.

Experiences can also be queried on the fly while users are on the move. In [4] we formalized interactive itinerary planning as an iterative process where, at each step: (1) the user provides feedback on a proposed set of POIs and the the system further selects a new set of POIs, with optimal utility, to solicit feedback for, at the next steps. This iterative process stops when the user is satisfied with the recommended itinerary. We showed that the problems of POI selection and of itinerary computation are both NP-complete and develped heuristics for helping users to construct itineraries on the fly. An interesting challenge here is to incorporate the physical co-location of one's friends in determining the next set of POIs with an objective function that maximizes socializing. In other terms, the system should account for the proximity of members of a user's social network in selecting the set of places a user could go to next and showing who will be there. That way, the user can choose or not to meet co-located friends.

One factor that comes into play when visualizing a set of experiences is their similarity/diversity ratio, i.e., the user is likely to prefer exploring many experiences that are different enough to be interesting as a whole but overlap enough to constitute alternatives to the same choice. While in other contexts such as Web search and recommendations [3, 5], diversity aims to minimize overlap between query results, here, the challenge is to achieve a good balance between similarity and dissimilarity.

References

[1] S. Amer-Yahia, M. Benedikt, L. V. S. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. *PVLDB*, 1(1):710–721, 2008.

[2] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In M. H. Chignell and E. Toms, editors, *HT*, pages 35–44. ACM, 2010.

[3] S. B. Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu. Constructing and exploring composite items. In A. K. Elmagarmid and D. Agrawal, editors, *SIGMOD Conference*, pages 843–854. ACM, 2010.

[4] S. B. Roy, G. Das, S. Amer-Yahia, and C. Yu. Interactive itinerary planning. In S. Abiteboul, K. Böhm, C. Koch, and K.-L. Tan, editors, *ICDE*, pages 15–26. IEEE Computer Society, 2011.

[5] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In M. L. Kersten, B. Novikov, J. Teubner, V. Polutin, and S. Manegold, editors, *EDBT*, volume 360 of *ACM International Conference Proceeding Series*, pages 368–378. ACM, 2009.

Extensible Context-aware Stream Processing on the Cloud: Rationale and Challenges

Walid G. Aref [1]
Department of Computer Science, Purdue University, West Lafayette, Indiana
aref@cs.purdue.edu

1. Rationale and Challenges for Massive Data Stream Processing on the Cloud

The ubiquity of mobile devices, location services, and sensor pervasiveness, e.g., as in smart city initiatives, call for scalable computing platforms and massively parallel architectures to process the vast amounts of the generated streamed data. Cloud computing provides some of the features needed for these massive data streaming applications. For example, the dynamic allocation of resources on an as-needed basis addresses the variability in sensor and location data distributions over time. However, today's cloud computing platforms lack very important features that are necessary in order to support the massive amounts of data streams envisioned by the massive and ubiquitous dissemination of sensors and mobile devices of all sorts in smart-city-scale applications. To support massive data stream processing, cloud platforms need to be extended in the following directions:

1.1 Hybrid memory-based and disk-based processing: Cloud platforms are typically designed for data-intensive disk-based applications. For massive data streaming applications, disks need to be replaced by distributed main-memory buffers at all the various layers. For example, distributed file systems or distributed disk-based databases are to be replaced by a memory-based online data stream acquisition layer that continuously receives streamed inputs via network sockets and stores them into distributed memory buffers. Disk storage has to be eliminated from the various other layers of massively parallel systems. For example, in a Map/Reduce scenario [4], the communication layer between the mappers and the reducers also has to be memory-based. It is important to point out that even for massive data streaming applications; combined memory- and disk-based platforms have to be integrated to answer user queries. For example, moving objects in the form of spatiotemporal data streams that are processed in memory will also need access to a road network that is stored in the disk-based platform. A typical continuous query on the moving objects' data streams would also have to access the disk-based road-network data. Therefore, a hybrid stream- and disk-based parallel architecture is needed for these applications.

1.2 Dynamic rate-based load-balancing and multi-stream partitioning: Typically, to achieve load balancing, distributed files or tables are partitioned based on their sizes and on the computing power of each of the participating machines, i.e., a more powerful machine claims a bigger portion of the data to process. In massively parallel streaming, the equivalent is to perform rate-based partitioning, i.e., the data streams with high rates are to be split into multiple processing units to avoid buffer overflows and load shedding. Additionally, slow streams needs to be bundled together to avoid under-utilization of resources. Therefore, three load balancing operations are speculated for data streams, namely, split, bundle, and migrate.

1.3 Fault-tolerance and stream replication: In batch cloud systems, data is replicated multiple times to guarantee availability of disk-data despite of disk, machine, and rack failures [e.g., as in [3]). In a massive data streaming cloud, most likely, systems cannot afford to redundantly replicate streamed data multiple times as data is continuously arriving. Novel approaches have to be developed to achieve fault tolerance for massive streaming environments.

1.4 Continuous and progressive window-based processing: In contrast to a batch mode of operation, cloud platforms have to process streamed data continuously using some notion of system pulse or beat, where between two pulses; newly arriving streamed data that gets accumulated into the memory buffers during the previous pulse are consumed and processed while more data arrives. Because data is infinite, some notion of scoping or windowing is needed. Distributed and reliable memory-based maintenance of window states and intermediate results that are accessible to all cloud processing units across pulses is necessary to guarantee correctness and progressive output that builds upon that intermediate state.

1.5 Online data stream summarization and analytics: In addition to the online processing of the massive data streams, data summarization and online analysis of the streamed data have to take place at various spatial and temporal granularities. This online data analytics component will have access to the memory-based streamed data and their summaries as well as to the disk-based related data, e.g., the road-network data. Phenomena detection, tracking, prediction, and emergency response are example functionalities at this level.

2. Rationale and Challenges for Extensible Context Awareness

Extensibility architectures have been proposed in the early 1980's to address the needs of emerging applications, e.g., as in extensible database servers. Although useful, these extensibility features are not adequate as they do not capture a key sensitive parameter in social networks, and database and web search applications, which is the "context" of the user or

[1] Walid G. Aref's research is supported in part by the National Science Foundation under Grants III-1117766, IIS-0964639, and IIS-0811954.

query issuer. For example, in location-service architectures, the location of the user or the query issuer is a sensitive parameter. In privacy-aware "Hippocratic" database systems [1], the identity of the user or query issuer is a sensitive parameter. In temporal databases, the time the query is issued is a sensitive parameter. In a location-aware social network, both the location and the identity of the user are sensitive parameters. Instead of tailoring a system for each sensitive parameter or each combination of sensitive parameters, we can make the system aware of the notion of "contexts" or "sensitive parameters" related to the user or query issuer. By using "contexts" as an abstraction, we can eliminate the need for tailoring specialized engines for each new context. Devising a general context-aware server will eliminate the need for the costly tailoring of a specialized server, e.g., a location server, a temporal DBMS, or a combined Hippocratic location-aware database server, since space, time, and identity of both the user or query issuer and the underlying database objects are treated as contexts. One can instantiate a new specialized server by defining appropriate contexts using context definition and manipulation languages. An extensible context-aware system should provide:

2.1 User or query-issuer contexts: These are high-level interfaces to define the user's or query issuer's contextual interfaces. This context reflects the situation of the query issuer, e.g., the query issuer's location, the time the query is issued, the identity of the query issuer, or even the temperature or the weather conditions surrounding the query issuer. A Context-aware server should be able to make use of these contexts when responding to a user's request.

2.2 Objects' reciprocal contexts: These are high-level interfaces to define the reciprocal contexts of the database objects. These object contexts will reflect on or reciprocate the user's or query issuer's registered contexts, e.g., the location of the database objects, the time duration of an object (or when the object can be available for querying, e.g., as in a restaurant's opening hours when a user is asking for a close-by restaurant), and the identity of the object (or the ids of the query issuers or classes of query issuers that are allowed to access the object as in privacy-aware database systems).

2.3 Binding mechanisms: The extensible context-aware server should have high-level interfaces that dynamically bind the contexts of the query issuer with those of the database objects. For example, binding the location and identity of the query issuer to the location and privacy profiles of social network objects can realize a location-aware social network.

2.4 Performance: The declarative definition of contexts in context-aware systems can have a strong impact provided that performance of these systems is competitive to those of tailored systems. The efficient realization of context-aware database management systems is one of the vital challenges. Related to the issue of performance is that of indexing. In contrast to building tailored indexing methods, e.g., for the spatial locations of objects, or for temporal intervals, can one construct generalized indexing techniques for contexts?

2.5 Dynamic context profiles: In many application scenarios, changes take place in the contexts, e.g., some active contexts may become inactive, inactive ones may become active, or new contexts get introduced. Another form of change is that the contextual values themselves within a context may change, e.g., the surrounding temperature may change or the location of a moving object may change, etc. These changes may affect the query being executed. This is similar in spirit to mid-query reoptimization [7]. However, the difference is that when the contexts change, the system may need to augment the query being executed by additional predicates that reflect the changes in the contexts.

3. Where We Stand

M3 [2] is a prototype data streaming system that is being realized at Purdue using Hadoop [3] and that eliminates all of Hadoop's disk layers, including the distributed file system (HDFS), and the disk-based communication layer between the mappers and the reducers. So far, M3 realizes features 1.1-1.4 above except that as of now, M3 handles only streaming data and does not handle queries that mix streaming with disk-based data. M3 extends Hive [9] to support streaming and continuous SQL querying using SyncSQL-like extensions [6]. Context awareness as an extensible vehicle has been demonstrated separately in Chameleon [5] based on extensions to PostgreSQL [8]. We are currently studying performance issues and general context-based indexing techniques as a step towards realizing context awareness in M3.

4. References

[1] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, "Hippocratic Databases", VLDB'02, Hong Kong, China 2002.
[2]A.M. Aly, A. Sallam, B.M. Gnanasekaran, L.-V. Nguyen-Dinh, W.G. Aref, M. Ouzzani, A. Ghafoor, "M3: Stream Processing on Main-Memory MapReduce". Demo Paper, IEEE ICDE, April 2012.
[3] "Apache hadoop: http://hadoop.apache.org/."
[4] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters", OSDI'04, pp. 137–150, December 2004.
[5] H.G. Elmongui, W.G. Aref, M.F. Mokbel, "Chameleon: Context-Awareness inside DBMSs", ICDE'09, pp. 1335-1338, April 2009.
[6] T.M. Ghanem, A.K. Elmagarmid, P.A. Larson, and W.G. Aref, "Supporting views in data stream management systems," ACM TODS, 2010.
[7] N. Kabra, D. Dewitt, "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans", ACM SIGMOD'98, Seattle, WA 1998.
[8] "PostgreSQL. www.postgresql.org/".
[9] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive - a warehousing solution over a MapReduce framework," PVLDB, 2009.

ASTERIX: Put Your Social Data Here!

Michael J. Carey
Computer Science Department, UC Irvine
mjcarey@ics.uci.edu

1 ASTERIX Objectives

The ASTERIX project at UC Irvine [4] began in early 2009 with the goal of combining and extending ideas drawn from semistructured data management [3], parallel database systems [10], and first-generation data-intensive computing platforms [9] to create a new, highly-scalable, semistructured information management system. ASTERIX is targeting use cases related to archiving, querying, and analyzing semistructured data drawn from Web sources, including sources such as Twitter, social networking sites, news sites, blogs, and so on.

Today's "Big Data" stack has formed around open-source software including Hadoop [1], its HDFS file system, and also various key-value stores [8]. Early clients of this stack primarily used the MapReduce programming model; today, the majority of its use is via declarative languages like Pig and HiveQL. Rather than adopting or modifying the current Hadoop stack to add features, the ASTERIX project is asking "What if we'd actually meant to design an open software stack with records at the bottom and a higher-level language API at the top?" We have thus been rethinking the layers [6] while taking care not to lose important attributes such as open-source availability, non-monolithic layers/components, access to file-based external data as well as system-managed data, fault-tolerant job execution and storage, and automatic data placement and rebalancing as data and machines come and go.

2 The ASTERIX System

Figure 1 provides an overview of the ASTERIX system. Data management in ASTERIX is based on a semistructured data model that can support use cases ranging from strict, table-like data collections with predictable types to flexible and more complex data where very little is known a priori and the instances in data collections are variant and self-describing. The ASTERIX data model (ADM) design was based on taking data concepts from JSON and adding additional primitive types as well as borrowing collection types from object databases [7]. The ASTERIX query language (AQL) was designed to manipulate semistructured ADM data. It was influenced by XQuery FLWOR expressions [12] while leaving XQuery's significant "XML baggage" behind.

While designing and building ASTERIX, three reusable software layers emerged; these layers are summarized in Figure 2. The bottom-most layer is a data-intensive runtime system called Hyracks [5]. The topmost layer of the ASTERIX stack is the ASTERIX system itself, a full parallel information management system that supports both native storage and indexing of data as well as access to external data residing in a distributed file system. Between these two layers lies Algebricks, a model-agnostic, algebraic "virtual machine" for parallel query processing and optimization. Algebricks is the target for AQL query compilation, but it can also be a target for other declarative data languages (e.g., we currently have Facebook's HiveQL language running on top of Algebricks). In addition, we are experimenting with graph data analysis and a Pregel-like programming API built on Hyracks to support it.

Given the target use cases, incoming data will often be dirty or noisy, making fuzzy search a key feature of ASTERIX. Analyzing such data to make recommendations or to identify sub-populations and trends in social networks requires matching data based on set-similarity measures. AQL supports such fuzzy joins over large data sets and executes them in parallel [11]. As the ASTERIX system is aimed at supporting data drawn from (or pushed to ASTERIX from) around the Web, including the increasingly popular and important mobile Web, ASTERIX includes built-in support for location-based (i.e., geospatial) data as well as support for automatic data ingestion via a new ASTERIX feature called datafeeds. Datafeeds channel data coming from continuous Web sources (such as Twitter and RSS-based news services) into affiliated ASTERIX datasets for either immediate or later searching and analysis.

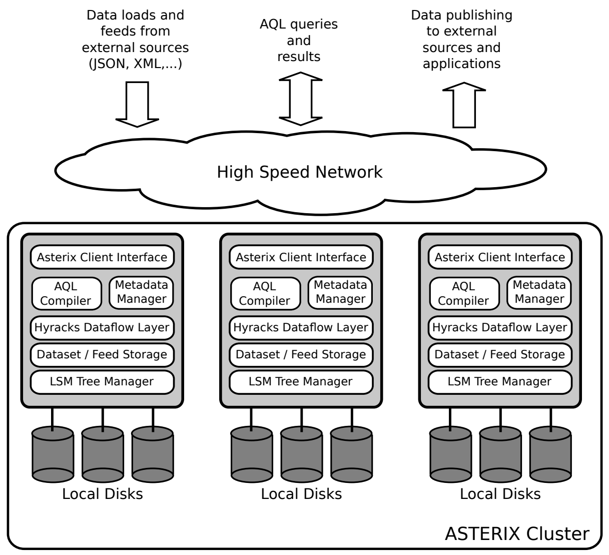


Figure 1: ASTERIX system overview

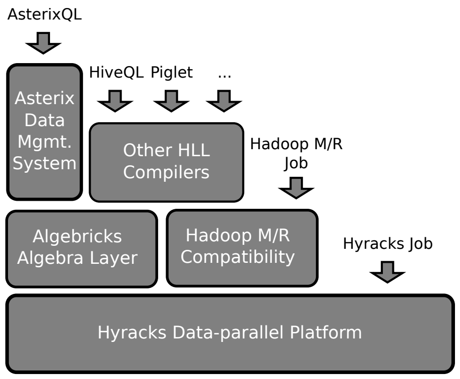


Figure 2: ASTERIX software stack

3 ASTERIX Status

We are 2.5 years into our initial 3-year, NSF-sponsored ASTERIX effort [2]. The Hyracks layer is now fairly mature and has been shown to significantly outperform Hadoop and other open-source alternatives on data-intensive computing problems on clusters with up to 200 nodes (with 800 disks and 1600 cores). The Algebricks layer emerged as a result of work to support both AQL and HiveQL on Hyracks; we have measured 3-8x performance gains in preliminary experiments comparing HiveQL on Algebricks with Hive itself on Hadoop. A group at Yahoo! Labs is using Hyracks (and planning to use Algebricks) as a platform for work on data-parallel machine learning over very large data sets. The ADM/AQL layer, *a.k.a.* ASTERIX proper, is now able to run parallel queries including lookups, large scans, parallel joins, and parallel aggregrates efficiently for data stored in a partitioned LSM B+ tree storage component and indexed via LSM B+ tree and LSM-based R-tree indexes. The system's external data access and datafeed features are also running in the lab. We are planning to offer a first open-source release of ASTERIX and its component layers sometime during the latter part of 2012, and we are actively looking for early partners who might like to try the ASTERIX software out on their favorite "Big Data" problems.

References

[1] Apache Hadoop website. `http://hadoop.apache.org`.

[2] Asterix project website. `http://asterix.ics.uci.edu/`.

[3] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[4] A. Behm, V. R. Borkar, M. J. Carey, R. Grover, C. Li, N. Onose, R. Vernica, A. Deutsch, Y. Papakonstantinou, and V. J. Tsotras. Asterix: towards a scalable, semistructured data platform for evolving-world models. *Distributed and Parallel Databases*, 29(3):185–216, 2011.

[5] V. R. Borkar, M. J. Carey, R. Grover, N. Onose, and R. Vernica. Hyracks: A flexible and extensible foundation for data-intensive computing. In *ICDE*, pages 1151–1162, 2011.

[6] V. R. Borkar, M. J. Carey, and C. Li. Inside "Big Data management": Ogres, onions, or parfaits?". In *Proc. EDBT 2012 Conf.*, March 2012.

[7] R. Cattell, editor. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.

[8] R. Cattell. Scalable SQL and NoSQL data stores. *SIGMOD Rec.*, 39:12–27, May 2011.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI '04*, pages 137–150, December 2004.

[10] D. DeWitt and J. Gray. Parallel database systems: the future of high performance database systems. *Commun. ACM*, 35(6):85–98, 1992.

[11] R. Vernica. *Efficient Processing of Set-Similarity Joins on Large Clusters*. Ph.D. Thesis, Computer Science Department, University of California-Irvine, 2011.

[12] XQuery 1.0: An XML query language. http://www.w3.org/TR/xquery/.

Resource- and Human-constrained Data Processing

Panos K. Chrysanthis & Alexandros Labrinidis
Advanced Data Management Technologies Laboratory
Department of Computer Science, University of Pittsburgh
panos@cs.pitt.edu & labrinid@cs.pitt.edu

Motivation

In the last 20 years, the mobile data management research community has produced techniques and solutions to effectively support data processing under the resource constraints inherent to mobile devices, e.g., communication bandwidth, battery, and size of the display. These solutions assumed small (user-device) interactions and the existence of a support infrastructure for sharing data processing between the mobile device and stationary computing devices. The emergence of the Computing Cloud, which has the potential to facilitate the services that comprise the assumed mobile support infrastructure, is making these solutions more feasible.

The hypothesis of this position paper is that the proliferation of mobile devices, and in particular smart phones, and the emergence of the social networks that generate content which is location-specific and potentially time-sensitive, require new solutions that go beyond just handling of the resource constraints to effectively handle large dynamic interactions, while preventing information overload for users.

The New Scalability Challenge

The "traditional" challenge characterizing mobile data management is that data processing in this environment has been (severely) resource-constrained. Although human cognitive capacity (referred to as *human processing* from now on) has also been a constraint in the past, social networks, combined with advances in mobile device technology, have made human processing an equally important challenge in mobile data management. So, the quest is not only in content efficiently to the user (given resource constraints), but doing so in a way to allow the user to process it (in a meaningful way). Context-awareness is playing a big role in identifying relevant content, but is not enough. The new challenge is how to leverage existing techniques and develop new ones, in order to perform (collaborative) data processing in this environment, **given both the inherent resource constraints and also the human processing limitations,** and to do so by taking advantage of the Computing Cloud.

Towards a Solution

We believe that the driving principle in addressing the new scalability challenge is the examination of the trade-offs imposed by the resource- and human- constrained processing. For instance, when choosing what content to push to the mobile devices, we need to consider both the bandwidth/energy requirements and the relevance of the content to the user (i.e., to prevent both energy depletion and mental overload). We believe the solution to this challenge to be at the intersection/integration of three research thrusts.

Materialized views as a mechanism for personalization and controlling information overloading

The idea is to build on the powerful notion of views which provide a means to present different users with different portions of the database, based on the users' perspective (i.e., context, and preferences). In order to support flexible query processing in meeting the needs of the users and also support asynchronous and disconnected communication to minimize energy consumption, one can explore customization and localization properties of the materialized views. Further, we propose to use the view maintenance options to personalize information sharing among users in a social network. Such personalization should consider both the types of content that are downloaded (i.e., be the most relevant for the specific user) and also the data availability/freshness preferences of the users (i.e., by allowing finer grain of control in maintaining consistency, controlling inconsistencies, and considering the ensuing trade-offs).

Such personalized view maintenance [1] can be supported by a cloud computing infrastructure which offers differentiated levels of view maintenance service (view holders) through multi-tenancy in a scalable way.

Computing Cloud as a Mobile Support Infrastructure
View maintenance is typically an expensive proposition [2] and it is becoming increasingly more expensive given the high incoming data rates, for example, due to real-time data feeds from sensor networks, from the Web, or from user-submitted data (on social networks). This necessitates sharing of (personalized view maintenance) computation between mobile devices and the cloud. This type of sharing can increase efficiency (esp. in terms of resource consumption), enable higher levels of sophistication at the cloud (given the computing power), and allow for larger volumes of data to be considered in tandem with user preferences (and possibly aggregated/filtered out at the cloud and not pushed to the mobile clients). The ability to only get notifications of events and data of interest is generalized through the notion of continuous queries, which are implemented through a Data Stream Management System (DSMS) [3, 4].

All these benefits are made available in an elastic way and in a distributed fashion, whereas the push-based data processing paradigm is implemented via a DSMS as a service (DSMSaaS) deployment and reliability is provided by means of relaxed transactional semantics (both at the cloud and the mobile device [5]). Although DSMS or DSMSaaS are often sufficient to handle simple interactions and single-user data processing requests, complex interactions and processing requirements involving multiple users (i.e., the result of collaboration) need a new paradigm. This is the paradigm supported by continuous workflows, i.e, the product of the "marriage" of traditional workflow systems and DSMS [6, 7].

Continuous Workflows as a facilitator of social network (open) collaborations
Being able to work collaboratively increases efficiency and generates better results (decisions) and innovative ideas. Social networks enable such collaboration (for fun and social gain), beyond the confinements of institutional or enterprise barriers. On the other hand, social-network-based collaborations share many similarities with virtual enterprises, effectively supported by workflows which define roles and tasks for each participant in the collaboration, as well as defining the steps needed for content to be developed, deployed, and shared. Collaborations in this context typically have real-time characteristics (in terms of both data and tasks) and involve continuous (i.e., never-ending) data exchanges. For this reason, we propose to implement continuous workflows as a cloud-hosted service (CWaaS), similarly to DSMSaaS. Successful deployment of CWaaS will require consideration of timeliness (by addressing scheduling) and quality of service and results (by addressing uncertainty, ranking of the answers and annotations).

Conclusions

In this position paper, we identified human- and resource-constrained data processing as the main challenge behind the promise of combining social networking, mobility, and the cloud. We proposed to decompose this challenge by utilizing Materialized views as a mechanism for personalization and controlling information overloading, using Computing Cloud as a Mobile Support Infrastructure, and by employing Continuous Workflows as a facilitator of social network (open) collaborations.

References

[1] Susan Weissman Lauzac, Panos K. Chrysanthis, Personalizing Information Gathering for Mobile Database Clients, Proc. of the 17th ACM Annual Symposium on Applied Computing (SAC'02), pp. 49-56, Mar 2002

[2] Alexandros Labrinidis, Nick Roussopoulos, Exploring the tradeoff between performance and data freshness in database-driven Web servers, The VLDB Journal, 13(3):240-255, Sep 2004.

[3] Mohamed A. Sharaf, Panos K. Chrysanthis, Alexandros Labrinidis, Kirk Pruhs, Algorithms and Metrics for Processing Multiple Heterogeneous Continuous Queries, ACM Transactions in Database Systems (TODS), 33(2):5.1-5.44, Mar 2008.

[4] Thao N. Pham, Lory Al Moakar, Panos K. Chrysanthis, Alexandros Labrinidis, DILoS: A Dynamic Integrated Load Manager and Scheduler for Continuous Queries, The Sixth International Workshop on Self-Managing Database Systems (SMDB'11), pp. 10-15, Apr 2011.

[5] Panos K. Chrysanthis, Krithi Ramamritham, Synthesis of Extended Transaction Models Using ACTA, ACM Transactions on Database Systems (TODS), 19(3):450-491, Sep 1994.

[6] Panayiotis Neophytou, Panos K. Chrysanthis, Alexandros Labrinidis, Towards Continuous Workflow Enactment Systems, Proc. of the 4th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'08), Nov 2008.

[7] Panayiotis Neophytou, Panos K. Chrysanthis, Alexandros Labrinidis, CONFLuEnCE: CONtinuous workFLow ExeCution Engine, Proc. of the 30th ACM International Conference on Management of Data (SIGMOD'11), pp. 1311-1314, Jun 2011

Analyzing and Integrating Social Media

AnHai Doan
University of Wisconsin-Madison and @WalmartLabs

I am interested in analyzing and integrating social media data at the semantic level, then providing such services on the cloud. This interest stems from my work at Wisconsin, Kosmix, and @WalmartLabs. Kosmix was a social media startup in the Bay Area. It was bought in 2011 by Walmart and converted into @WalmartLabs, a research and development lab that analyzes social and mobile data for e-commerce.

Semantic services on the cloud

By social media I mean data such as tweets, blogs, and Facebook updates. A lot of work has analyzed such data, but at the *keyword* level, to answer questions such as "how many tweets mention the word "Obama" today?". In contrast, I want to analyze and integrate such data at the *semantic* level, to answer questions such as "how many tweets mention President Obama today?". To answer this question, I would need to recognize that tweets that mention "Obama", "the pres", "BO", "the messiah", etc. all refer to the same person. In other words, I want to infer entities and relationships from the raw social media data, then leverage them to provide useful services.

Numerous examples of semantic analysis and integration of social media have been studied. Here's a small sample:

- **Information extraction and entity disambiguation**: Given a tweet such as "mel crashed his car. maserati is gone", recognize that "mel" is a person name and that "maserati" is a car name. Further, recognize that "mel" here refers to the person entity Mel Gibson, the Hollywood actor, and not some other Mel.
- **Event discovery**: Find interesting events in the Twittersphere. Global events include Japanese earthquake and the bin Laden assassination. Local events include a planned protest in a square in Syria and a book fair in Mountain View, California.
- **Event monitoring**: Once an event has been found, find all tweets related to that event and display them in a continuously rolling fashion, as they appear.
- **Statistics gathering**: How many tweets mentioned Mitt Romney in the past three hours? What is the overall sentiment of Florida voters with respect to Newt Gingrich in the past two days?

I am interested in developing such semantic services, and then deploying them on the cloud, for companies and end users. For example, an end user may be interested in monitoring all tweets related to an upcoming book fair in Mountain View. He or she can go to a Web site provided by us, define the book fair event, then subscribe to it. When we find any tweet related to this event, we will automatically send the tweet to the user.

Research directions

At Kosmix and @WalmartLabs we have studied many of the above problems. Our experience suggests the following research challenges.

Traditional integration challenges: We must build a giant knowledge base of entities and relationships, along the line of Wikipedia and Freebase, then use this knowledge base to analyze and integrate social media. For example, given the tweet "mel crashed his car", we can recognize that "mel" is a possible person name, because it appears as a person name in our knowledge base. Further, we can match "mel" into the Mel Gibson node in the knowledge base, thereby performing entity disambiguation.

While critical, developing this knowledge base raises numerous challenges. First, we must integrate data from numerous sources, such as IMDB, Wikipedia, Freebase, Musicbrainz, TripAdvisor, etc. into a coherent whole. What is a good end-to-end ETL methodology to integrate such data? How can we use big data techniques (such as Hadoop) in such integration, so that we can scale up to terabytes of data? Events in social media often unfold at real-time speed, in seconds or minutes. How can we refresh the ETL pipeline quickly, so that a change in a raw data source can be reflected in the knowledge base in seconds? Today there has been relatively little research on these challenges.

Social expansion challenges: Once the giant knowledge base has been built using "traditional" Web data (such as IMDB, Wikipedia, Musicbrainz), we must expand it with entities, relationships, and events emerging from the social media sphere. For example, we must continuously add Twitter users and Facebook users, as we discover them, into the knowledge base. We must discover interesting events in the Twittersphere, and add those to the knowledge base too.

If we view the knowledge base as our "understanding of the world" that can be used to help us analyze and integrate social data, then clearly this "understanding" must involve not just "old" entities, relationships, and events (that we find in the traditional Web data), but also "new" entities, relationships, and events that have just emerged in social media. Such social expansion of the knowledge base raises interesting challenges, such as how to match social media users (e.g., a Twitter user account) to person entities already existing in the knowledge base, and how to discover interesting events in the Twittersphere.

Social context challenges: Not only do we have to expand the knowledge base "socially", by adding "social" entities and relationships, but we also have to add social context to each node in the knowledge base. To see this, consider again the tweet "mel crashed his car". Given just this tweet, we can't really tell that "mel" here refers to Mel Gibson, because there is no keyword indicative of Mel Gibson in the tweet, such as "actor", "Oscar", "scandal", and "Hollywood". On the other hand, if we know that Mel Gibson has just crashed his car, and that in the past three hours, most tweets related to Mel Gibson mentioned words such as "crash", "car", and "maserati", then we can match "mel" to Mel Gibson with high confidence.

The above example suggests that for each node in the knowledge base (which can be visualized as a giant graph), we have to maintain a *social context*, a set of words that are most indicative of that node in the past few hours in the social media. The social context is the key that allows us to perform semantic analysis such as entity disambiguation. The challenge is how to construct these social contexts accurately, to maintain them on a very large scale, for hundreds of millions of nodes, and to ensure very low latency.

Crowdsourcing and human computing challenges: In building the knowledge base, expanding it with social elements, and adding social contexts, we will have to use not just algorithms, but humans as well, in a crowdsourcing fashion. How to crowdsource effectively is a major challenge.

Further, once we have deployed semantic services on the cloud, how to effectively engage the end users is also a major challenge. For example, when a user goes to our site to define the book event in Mountain View, what should we ask the user to do, what kind of information should he or she provide, so that we can effectively find tweets that are related to that event for the user?

Scaling challenges for big and fast data: Clearly we will have to deal with not just big data, but also fast data, such as high-speed streams of tweets, Facebook updates, Foursquare checkins, etc. MapReduce has proven to be an effective paradigm to write and execute big data programs. Can we develop a similar paradigm for fast data?

When we deploy semantic services on the cloud, we will have potentially thousands to millions of users taking advantage of the services (e.g., defining an event and monitoring the event). This will severely exacerbate the above big data and fast data challenges.

System Support for Managing Large Graphs in the Cloud

Sameh Elnikety and Yuxiong He
Microsoft Research

1 Motivation

Large graphs are at the heart of online social networks and many other applications including routing in road networks and online collaboration systems. In this paper, we argue that a novel distributed infrastructure is needed to manage and query large graphs to meet the demands of these applications.

In a public social network, a node represents an entity such as a person, event, or photo. An edge represents a binary relationship between two nodes indicating for example friendship, participation at an event or appearance in a photo. Both nodes and edges may have a set of attributes because they model real world entities and interactions. The resulting social graph is challenging to manage: It is too large to manage on a single server, there are frequent updates and users want to pose ad-hoc queries. For example, a user may ask “which photos include me and my friends X and Y”, “how I am connected to person Z”, or “who among my friends is attending this event”.

Private social networks pose similar challenges. For example, Codebook [BPZ10] is a social network of software developers and their software artifacts within an enterprise. Codebook manages a large graph modeling software developers with their organizational hierarchy, source code (including files and functions and their revisions), bug reports, and design documents. Such data are gathered from source code repositories and the employee database. Codebook allows engineers to ask “who resolved this bug”, and “who built this feature”, and “who will be impacted if I change this source code function”.

2 Limitations of Current Graph Systems

In contrast to batch graph processing systems, which have important applications such as social network analytics and webpage raking, we focus here on online systems that answer interactive queries within a few hundred milliseconds.

Current graph management systems offer limited functionality to answer the queries mentioned in Section 1, which include both reachability queries and graph pattern matching. Those types of queries are not suited for relational engines [ADJ87] for several reasons: (1) Some queries are recursive (e.g., reachability queries), (2) nodes and edges are accessed in pattern not suitable for disk-based data structures, and (3) pattern queries require an excessive number of join operations with large intermediate results. This motivates graph systems to build their specialized graph engines.

Existing graph systems lack declarative query languages. For example, neo4j, which is among the most popular centralized graph systems, provides a navigational interface, where programmers need to write programs, rather than declarative queries. Distributed graph engines such as Google Pregel [MAB+10] and Microsoft Surfer [CWHY10] accept programs that execute at graph nodes and send messages across graph edges, focusing more on batch processing.

3 Cloud Infrastructure

Cloud computing offers two opportunities. First, the availability of a large number of servers allows using large main memories to host the topology of large graphs, enabling online query processing. This is important because processing queries over disk-based graphs is too slow for interactive queries. Second, multiple sources of information can be aggregated into a multi-graph, enabling richer queries. A user query can access data from several social networks: Facebook and LinkedIn have portions of the user social network.

4 Where We Stand

We outline several problems which we are investigating and invite our colleagues to reshape and solve them.

4.1 Graph Query Languages

The interface of graph system should be a declarative query language to allow users to write queries rather than navigational programs. We find that the majority of graph queries can be expressed as regular expressions or graph patterns. Both types of queries can be expressed declaratively, allowing execution engines to optimize their processing.

4.2 Execution Engines

Graphs are processed on a cluster of servers. Hybrid execution engines, which use both a graph engine and a relational engine, offer important advantages. A graph engine maintains the graph topology in main memory to answer reachability queries, and a relational engine manages node and edge attributes to retrieve predicated graph elements.

4.3 Isolation

Graph operations are dominated by traversals with more reads than writes. Multi-version concurrency control models offer clear correctness semantics as well as low overhead for read dominated workloads. In particular, generalized snapshot isolation [EPZ05] extends conventional snapshot isolation to distributed systems in a manner that allows graph traversals to neither block or be blocked by updates while providing serializability [BHEF11].

4.3 Query Optimization

With a declarative query language, a query optimizer can generate efficient execution plans customized for the managed graph instance to visit fewer nodes. Mid-query re-optimization [KD98] and budget-based techniques [BPS11] seem effective for traversing scale-free graphs [BB03].

Graph indexes and materialized views are active areas of research but they are developed in isolation and have not been integrated into graph systems. Current optimizers do not fully exploit them.

References

[ADJ87] R. Agrawal, S. Dar, H.V. Jagadish. "Direct Transitive Closure Algorithms: Design and Performance Evaluation." VLDB 1987.

[BB03] Albert-László Barabási, Eric Bonabeau. "Scale-Free Networks." Scientific American May 2003.

[BHEF11] Mihaela Bornea, Orion Hodson, Sameh Elnikety, Alan Fekete. "One-Copy Serializability with Snapshot Isolation under the Hood." ICDE 2011.

[BPS11] Matthias Bröcheler, Andrea Pugliese, V. S. Subrahmanian. "A budget-based algorithm for efficient subgraph matching on Huge Networks". GDM 2011

[BPZ10] Andrew Begel, Khoo Yit Phang, Thomas Zimmermann. "Codebook: Discovering and Exploiting Relationships in Software Repositories." ICSE 2010.

[CWHY10] Rishan Chen, Xuetian Weng, Bingsheng He, Mao Yang. "Large Graph Processing in the Cloud." SIGMOD 2010.

[EPZ05] Sameh Elnikety, Fernando Pedone, Willy Zwaenepoel. "Database Replication Using Generalized Snapshot Isolation." SRDS 2005.

[KD98] Navin Kabra, David DeWitt. "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans." SIGMOD 1998.

[MAB+10] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, Grzegorz Czajkowski. "Pregel: a System for Large-Scale Graph Processing." SIGMOD 2010.

[SEHK12] Mohamed Sarwat, Sameh Elnikety, Yuxiong He, Gabriel Kliot. "Horton: Online Query Execution Engine for Large Distributed Graphs (Demo)". ICDE 2012.

Labor in the cloud: Designing incentives for sharing performance data

Panos Ipeirotis

New York University

Going beyond the provision of "traditional" computing services through the cloud, there is another cloud service that is changing everyday life: the emergence of crowdsourcing (or "labor as a service") as a way to get immediate access to human intelligence, from within a computer system. This idea has changed the way that computer systems are designed and implemented. While this immediate access to human intelligence allows for innovative applications, at the same time we see questions that are not typically encountered when designing purely computational systems.

Separation of task execution and recruiting

For example, a crucial question is how should we abstract and separate the different functionalities provided by crowdsourcing platforms? What are the basic services that a platform should provide? Today, we see a separation of the recruitment functionality from the user-interface and task-handling functionality: Most services abstract away from the "labor channel" and then build their own "application logic" on top. The major example of a recruiting service is Mechanical Turk, and other companies (oDesk, Samasource, etc) are also providing similar labor channels. On the other hand, task handling is done by the employer under a separate interface, almost always ignoring any task handling functionality provided by the labor platform.

The tragedy of the commons in crowdsourcing systems

While this is a natural separation of specialties and focus, this separation of labor channel from task execution also creates the following problem: Each task-handling participant evaluates the recruited employers and knows their performance. However, there is no incentive to share back this information with the labor channel. Providing feedback is effectively a public good. Feedback benefits others but not the one who provides it. Combining this public good properties with a competitive environment, where each service is competing for access to the best workers, creates an environment where effectively nobody has the incentives to share private knowledge about the worker performance. Who wants to tell competitors who are the most trusted and reliable employees?

This is a highly suboptimal solution. First, when there is no public information about the performance of each worker, all employers need to devise their own tests and measurements, and learn by trial-and-error who the workers are that provide high quality services. To make the parallel with computing services, we would have to run benchmarks on every cloud service before even starting executing anything of interest. Combining that with the fact that workers have limited capacity, it is understandable why an employer does not want to share this information with others. Second, even two competitors may end up being better off by sharing information: If two employers have information about, say, 50% of the workforce, they could share information with each other and have information about all workers, saving each other the cost of testing.

Unfortunately, sharing such valuable information generates a prisoner's dilemma situation. While sharing is a better solution for both parties, it is even better for someone to back off and wait for others to share. Or even worse, a malicious employer may give incorrect information to others, in order to feed false information to competitors and lead them to hire incompetent or malicious workers.

While this setting is currently limited to crowdsourcing services, we can see a similar problem emerging in distributed cloud systems, where resources are contributed by a variety of participants and are not controlled strictly by a single provider (Amazon, Google, etc.)

The research challenge

What are the structures that can encourage and incentivize truthful revealing of reputation by the employers who have evaluated workers? Clearly, the worker that provides a public reputation feedback should expect something in return, and not just see others free-ride on this information.

A potential approach is to get employers to post feedback in an "escrow" system which others can probe/search but not browse. In other words, you need the identifying information about the worker whose profile is requested, you cannot just query for "give me the best workers". This ensures that someone can get information only for workers for whom there was some interaction, not for the global pool of workers. Incentive-wise, this means that you need to be participating in the system to get access to performance data. To ensure that there are no abuses of the probing privilege, a tit-for-tat mechanism (e.g., 10 queries, after posting information for a single worker) can mitigate such concerns.

Of course, this does not ensure that the employers will be truthful. In fact, the incentives are to post incorrect information, in order to fool the competitors and lead them to hire incorrect workers. One potential avenue towards fixing this problem is to examine the level of agreement when evaluating a single worker: If an employer provides the same feedback as other good employers, then the information should be trusted. If not, then the information is not reliable and the employer should not be rewarded with access to the pool of information about the workers.

We can augment this simple model to account for factors such as worker reliability (reliable workers get same scores from everyone, unreliable are difficult to label correct), worker focus (scores depend on area of focus), time, and other factors: We can simply leverage all the recent work on managing noisy workers in crowdsourced environments. This can generate a reputation system in which there is a "tit for tat" system of contribution and employers actually benefit by contributing to this common pool of semi-public feedback.

Alternative approaches can also be explored, trying to answer a broader question: How can we share valuable information only with others that contribute back valuable information?

Position Paper: Extracting Geospatial Information from Social Media

James M. Kang, Ashley Holt, John Greer
National Geospatial-Intelligence Agency
7500 GEOINT Drive, Springfield VA 22150

The past decade has seen the sudden attention to and assimilation of social media technologies all over the world. Social media (e.g., Twitter, Facebook) has been referenced as one of the major driving forces of the recent Arab Spring spreading across the middle east (e.g., see Stefanidis et al. 2011). The immense amount of data generated from social media provides opportunities for researchers to understand and extract useful information about social networks, cultures, sentiments, and the transmission of ideas through populations.

In recent years, social media technologies (e.g., Twitter and Facebook) are also beginning to embrace the usefulness of having a spatial reference along with the messages. Combining Geospatial information within social media may identify not only location, but also local culture, "slang" terms and place names unique to that location.

Utilizing geospatial location and natural language processing techniques may help in extracting some spatial information. For example, Cheng et al. utilized geospatially referenced Twitter information to geo-locate messages based on content [Cheng 2010]. Building on this work, Cano et al. leveraged temporal information in social media for identifying relationships between geographic location and the social constructs associated with place, which may change over time.

There exist several main challenges when extracting geospatial information from social media; for example the user-contributed information may not have any geospatial identifiable text (e.g., geo-coordinate) within it. In addition, the massive amount of social media produced daily causes significant computational challenges to process the datasets. Extracting and processing geospatial information from these massive datasets in near real-time requires state-of-the-art large-scale computing methods, including cloud computing.

References

[Cano 2011] Cano, A.E., A. Varga, and F. Ciravegna (2011). Volatile Classification of Point of Interests based on Social Activity Streams. Procedings of the 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011.

[Cheng 2010] Cheng, Z., Caverlee, J., and Lee, K. (2010) You are Where You Tweet: A Content-Based Approach to Geo-locating Twitter Users. CIKM'10 Conference Procedings, October 26–30, 2010.

[Stefanidis 2011] Stefanidis, T., A. Crooks, and J. Radzikowski (2011). Harvesting ambient geospatial information from social media feeds. GeoJournal, DOI 10.1007/s10708-011-9438-2.

Supporting Collaborative Recommendation Services in the Pervasive Computing Era

Wang-Chien Lee, Pennsylvania State University, email: wlee@cse.psu.edu

1 Recommendation Services in Pervasive Computing Era

With the wide deployment of wireless network infrastructure and the success of many popular mobile devices, the era of pervasive computing has emerged as part of our daily life. Via those devices, mobile users nowadays are able to get connected with friends, find useful information for various needs, conduct business/tasks/shopping conveniently, and participate in activities from anywhere, any time. To embrace the growing population of mobile users, many websites and services have enhanced their mobile web support and created new service features. For example, conventional e-commerce sites such as Amazon and Netflix facilitate mobile shopping and video watching. Additionally, powered by the ever-growing capability of smart mobile devices and the advances of Web 2.0 technology, Frickr and Bikely allow mobile users to easily acquire information about happenings in their surroundings or about themselves (e.g., photographs and biking trajectories) for sharing with their friends or even the public.

Personalization has long been a buzzword on the Web. As users become more proficient in their use of the Web, the one-size-fits-all approach for web service provisioning no longer satisfies users' expectation on web experience. To improve their customer satisfactory and boost potential sales, a growing number of websites and web services, e.g., Amazon and Netflix, have incorporated personalized recommendation techniques to enhance the on-line experience of their users. For example., when a user is searching for product information regarding an item of interests, the service provider may promote a number of relevant items as recommendation to the user, e.g., an extra battery or tripod are recommended to a user who buys a camera. The arrival of pervasive computing era brings great opportunities for provisioning of recommendation services to mobile users. For example, mobile advertisement/coupons, which can be seen as a variant of recommendation, from nearby stores may be offered when a user enters a shopping mall. Meanwhile, these opportunities also brings new technical challenges since simply sending all the ads and offers from nearby stores to perspective customers is not going to be effective. Even worse, mistargeted advertisement may annoy mobile users and result in customer complains.

2 Mobile and Social Computing for Recommendation as a Cloud Service

The rapid technological development of cloud computing and data centers in the past few years has provided a much needed service infrastructure for various pervasive computing applications. Due to a wide spectrum of applications, we envisage recommendation as an important cloud service (like the intelligent personal assistant Siri on iphone) for supporting various activities of mobile users.

Recommender systems have attracted a lot of attention from the industry and academic research communities in the past decade and become a core technology for many e-commerce sites. The basic idea behind most recommender systems is to exploit the best matches between user preference and product profile in order make the most effective recommendation to targeted users. Capturing *user preference* accurately is essential for recommender systems as people participate in events or select items (e.g., books, places, etc) mainly based on their interests/preferences for the items. Thus, recommender systems explore user preferences in various ways to recommend matched objects. For example, content-based filtering techniques recommend items to a user by matching item content (including item description, tags or other attributes) with the user's explicitly maintained personal interests/preferences, while collaborative filtering techniques recommend items to a user by exploiting the preferences of *like-minded users* over items. While collaborative filtering and content-based recommendation techniques both have their own strengths and weakness, e.g., the content-based techniques are constrained in the scope of recommendations while the collaborative filtering techniques require a large amount of information on a user in order to make accurate recommendations, we see great research potential in the collaborative filtering approach due to the growth of mobile and social computing technology in supporting collaborative recommendations. Notice that collaborative recommendations are enabled by collecting and analyzing a large amount of information on users behaviors, activities or preferences for predicting what users will like based on their behavior/preference similarity to other users. Without even understanding the item content, recommendation of various complex items can be made accurately based on preference/behavior (i.e., previous selections) of "similar" users. Social and mobile computing technologies are envisaged to support the social crowd wisdom of similarly behaved users.

- Taking into account more preference and behavior information of users in the recommendation process can intuitively enhance the effectiveness of recommendation services. The smart mobile devices, carried and used by their owners, can be used to collect valuable data in order to dictate the preference and behavior information of users. Note that the information is useful for understanding the on-going activities/context in decision making process of users and thus valuable for making effective recommendation. This information, if voluntarily shared by users, can also be aggregated to identify users with similar behavior (i.e., like-minded users) for collaborative recommendation.
- In addition to personal behavior information, the social relationship among users can be exploited to alleviate the *cold start* problem faced by collaborative recommendation. Specifically, the problem concerns the issue that the system cannot draw an effective recommendation for those users whom it has not yet gathered sufficient information. Based on the observed *homophily* and *social influence* phenomena exhibited among friends, not only the behavior similarity but also the social influences from friends can be incorporated to explore the social wisdom in the recommendation process.

3 Technical Challenges

Enabling collaborative recommendation services is a very challenging task due to the dynamic nature of user behaviors and social relationship as well as various issues such as accuracy, data sparsity, scalability, privacy, etc. Efficient collection of personal behavior data from a large mobile user population and transform them into useful social wisdom for collaborative recommendation is very challenging. The following are a number of technical areas that require research effort:

- Data Collection – collaborative filtering techniques rely on a large amount of information on user preference and behavior to make effective recommendations. Thus, collecting user preference and behavior data is essential. In addition to explicitly ask for rating information of items from users, mobile devices can be used to implicitly keep track of the times, places, duration, and responses when users are accessing/examining some items. Moreover, other factors such as the emotion perception of users may be considered for making recommendations. How to explore the various sensors equipped in mobile devices and various types of applications/information accessed via mobile devices to collect useful information is an important task.
- Crowd Sourcing – due to the data sparsity and cold start problems inherently faced in collaborative filtering techniques, creating the much needed "crowd wisdom" is very important to get the service started. In addition to provide incentives or devise some interesting applications (e.g., games) to attract users, tools to facilitate crowd sourcing to the mobile users, e.g., CrowdDB, are useful.
- Preference Mining – to explore the crowd wisdom of mobile users, it is important to develop data mining techniques for efficient processing of large-scale data collected from mobile users to capture their preference. Probabilistic latent classes modeling techniques can be employed to mine and capture user preferences to support collaborative recommendation.
- Social Behavior Modeling and Mining – to better understand the behavior similarity and social influence of friends, data mining techniques that consider both social friendship information along with collected user behavior data need to be developed. Quantitative measure of the behavior similarity and social influence among friends may provide theoretical basis for capturing the social behaviors in decision making process and thus are very useful for collaborative recommendations.
- Scalability – developing efficient algorithms under the cloud framework to provide effective realtime recommendation services for the large population of mobile users is a very challenging yet important task. Algorithms that can exploit the rich resources in the cloud infrastructure and incrementally adapt the mining result to the dynamic changes in user behavior and social data are desirable.
- Privacy Issue – as collaborative recommendation is enabled by collecting preferences and behavior information from many users, privacy preservation of collected user data is a major concern. Thus, it is desirable develop effective techniques for the above-discussed technical areas without directly using sensitive private user data.

In summary, collaborative recommendation is envisaged as an important cloud service for supporting various activities of mobile users. Mobile and social computing techniques for enabling collaborative recommendation services on the cloud may provide technological advances and great benefits to applications in the pervasive computing era.

Private Data Exchange

Ashwin Machanavajjhala, Yahoo! Research, Santa Clara, CA, USA
mvnak@yahoo-inc.com

1 Introduction

The personal information that users generate on the Web, social networks and that is continuously tracked via mobile phones can enable new science and can be used to provide valuable services to users. However, most of this information is tracked and stored in vaults by private industries. Health data is vaulted away in hospital and insurance databases, while search data is exclusively owned by Web search companies. This leads to many problems – 1) there is no mechanism for individual to own and get remunerated for their valuable data, 2) there is no ways users can track what is being done with their data and 3) scientific endeavors that might want to join e.g., medical with Web data are not possible since the data is not available in one place.

I envision a future where users retain ownership of their personal information and store it in one secure location on the cloud. I call this the Private Data Exchange (PDX). Not only can users now track all their data, they can allow third parties to write applications on top of this data, and can "sell" their data to either companies in health care, social networks, or to research e.g., from the Census in return for either monetary compensation or services. I believe that with the advances in cloud technology, privacy mechanisms and our understanding of markets, this vision can indeed become a reality.

2 Private Data Exchange

A *Private-Data Exchange*, or PDX, is a system where individuals store all their personal information in one place on the cloud, and registered applications and service providers, e.g., insurance companies, advertisers, or researchers, can negotiate with the individuals to use their personal information in return for monetary or service-based incentives. Such a system would help users manage and monitor their personal data in a single place, track breaches of privacy, and allow them to be remunerated for sharing their information. It would also help build novel services by integrating disparate kinds of information. PDX would have the following key components: (a) a secure cloud-based storage infrastructure, where users can store and retrieve their personal information, (b) mechanisms for application developers to write queries (including long-standing continuous queries) over user data, (c) a query optimizer, which tracks the requests for data from various service providers, and generates the right answer based on prior queries being answered and the user's privacy settings, (d) a privacy negotiator, that allows users to trade-off utility for privacy with service providers, and finally (e) an auditor, which ensures that personal information is accessed according to correctly negotiated terms of use. Building such a system has many interesting research challenges. While there is ongoing work on some of the research questions, like secure cloud storage [1], and privacy auditing [2], I outline below a selected set of research questions related to privacy and big-data management that would provide the necessary set of tools and concepts to realize this vision.

3 Research Questions

• Privacy Fundamentals Research

In order to understand the limits of the envisioned Private Data Exchange, the following key fundamental questions in privacy must be answered. Recent research [3, 4, 5] has shown that state-of-the-art privacy definitions assume a "worst case" adversary that are either unrealistic, leading to poor utility, or do not represent the worst case for data where individuals are correlated, leading to privacy breaches. Hence, current privacy mechanisms are not applicable to realistic data arising on the Web and social networks, and in continuous monitoring applications. Moreover, results like the No Free Lunch Theorem [3] show that there is no single privacy definition for all kinds of data. For instance, it was shown that existing privacy mechanisms (like differential privacy) only work for data without correlations, and either leak information or provide no utility in social network related data sharing. Hence, finding the right privacy definitions and mechanisms for social networks and continuous data collection is an important open problem. More ambitiously one can think of building a customizable privacy compiler tool that can generate domain specific mechanisms with formal privacy guarantees.

Next, an obstacle for the adoption of current state-of-the-art privacy definitions is that it is hard for an individual or an application developer to understand what sensitive information maybe breached to a realistic adversary. It would be nice to build tools that, given a proposed mechanism for data sharing, automatically identifies examples of possible privacy breaches. This tool can then elicit user feedback as to which breaches are tolerable and which are not, and accordingly can tune the privacy mechanism. Finally, integrating utility-theoretic notions of privacy into the prevalent mathematical worst-case notions would be required to better understand how individuals trade-off utility for privacy.

• **Big-Data Management Challenges**
The envisioned PDX system brings forth many data management challenges, including those important in other settings as well, in building and managing systems that can efficiently and securely store varied forms of personal information. First, feed-following [6] is becoming a very important pattern of real-time data access. For instance, a user in a social networking, an emergency responder, or a health provider-facing application may want to follow the "feed" of information generated by another user. Queries are then answered on the pertinent set of feeds – e.g., latest k urls tagged by a user's friend on a social network, or the set of users driving at speed $> 85mph$. One can envision queries that need probabilistic inference over an individual's data, e.g., the set of individuals with $P[\text{heart attack } > .8]$ based on a model over an individual's medical history and physical activity. At the same time, unintended recipients must not learn anything about the user. Some key research questions in this problem are outlined in [6].

A second interesting problem is that of view materialization on the cloud for applications that continuously monitor aggregate statistics over a population of individuals. As the size of raw data increases, it is important to intelligently materialize views over the raw data. This is all the more critical if the access must be privacy preserving – every time personal information is accessed, there is a potential for further information disclosure, thus motivating the need for privacy-aware views.

• **Novel Applications**
Today, individuals share immense amounts of information about themselves online. These include unstructured posts on social networking sites, and activity tracked via continuous monitoring mobile applications. Integrating such information can provide valuable cues about population demographics (with applications to the Census), and about individual and population health (e.g. Google Flu). One concrete research problem is reducing the costs and effort of performing a Census by utilizing the unstructured information on the Web and social networks, as well as employing crowd-sourcing techniques where the other sources do not have sufficient coverage.

References

[1] S. Bajaj and R. Sion. Trusteddb: A trusted hardware based outsourced database engine. *PVLDB*, 4(12):1359–1362, 2011.

[2] D. Garg, L. Jia, and A. Datta. Policy auditing over incomplete logs: Theory, implementation and applications. In *ACM Conference on Computer and Communications Security*, October 2011.

[3] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 international conference on Management of data*, SIGMOD '11, pages 193–204, New York, NY, USA, 2011. ACM.

[4] A. Machanavajjhala, J. Gehrke, and M. Götz. Data publishing against realistic adversaries. *Proc. VLDB Endow.*, 2:790–801, August 2009.

[5] A. Machanavajjhala, A. Korolova, and A. D. Sarma. Personalized social recommendations: accurate or private. *Proc. VLDB Endow.*, 4:440–450, April 2011.

[6] A. Silberstein, A. Machanavajjhala, and R. Ramakrishnan. Feed following: the big data challenge in social applications. In *Databases and Social Networks*, DBSocial '11, pages 1–6, New York, NY, USA, 2011. ACM.

Rethinking web content distribution in the social media era

Alan Mislove, Northeastern University

We are witnessing the beginnings of a shift in the patterns of content creation and exchange over the web. Previously, web content—including web pages, images, audio, and video—was primarily created by a small set of entities and was delivered to a large audience of web users. However, recent trends such as the rise in popularity of online social networking; the ease of content creation using digital devices like smartphones, cameras, and camcorders; and the ubiquity of Internet access have democratized content creation. Now, individual Internet users are creating content that makes up a significant fraction of Web traffic [3, 9].

As a result, compared to content shared over the web just a few years ago, content today is generated by a large number of users located at the edge of the network, is of more uniform popularity, and exhibits a workload that is governed by the social network. Unfortunately, existing content distribution architectures—built to serve more traditional workloads—are ill-suited for these new patterns of content creation and exchange. For example, web caches have been shown to exhibit poor performance on social networking content [5, 18], due to the more uniform popularity of content, causing many online social networking sites have begun to move away from content distribution networks (CDNs) and towards highly-engineered in-house delivery solutions [10, 11, 15].

Changing workload patterns

To more closely examine these trends, we examine a data set on the photos exchanged by 63,731 users from the New Orleans Facebook regional network [14]. Because data on photo *views* is not available, we use photo comments as a proxy for views (i.e., if a user has commented on a photo, they must have viewed it). Crawling the news feed in a manner similar to previous work [16], we discovered information on a total of 1,068,787 comments placed on 816,508 different photos. While we only analyze this dataset due to lack of space, we have found similar results on other social networks.

Content is created at the edge We first explore *where* the emerging content being exchanged over the web is being created. Today, the rapid adoption of smartphones, digital cameras, digital camcorders, and professional-quality music and video production software, combined with the low cost of broadband Internet service, has greatly eased content creation by individual users. Significantly more news articles are written by bloggers than news organizations [12], more photos are shared on online social networks [8] than on professional photography websites [1], and much of the content shared on YouTube, the most popular video-sharing site, is created by end users [6, 7] empowered by the ubiquity of webcams.

Content is of more uniform popularity We now explore the *popularity distribution* of the content in emerging workloads, relative to previous workloads. To do so, we examine the popularity of photos on Facebook, comparing it to the popularity distribution to that observed in studies of traditional web workloads [4]. We note one primary distinction with respect to traditional workloads: The Facebook workload contains a significantly lower exponent of the Zipf distribution (0.44, compared to 0.64 to 0.83 [4]), implying less emphasis on popular items and resulting in a more uniform popularity distribution and a significantly longer, fatter tail (Figure 1).

Exchange is governed by the social network We turn to explore *how* users are locating content. In particular, we explore the degree to which the exchange of content is governed by the structure of the

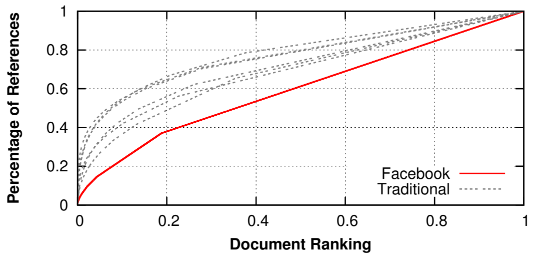


Figure 1: Cumulative distribution of Facebook views compared to five traditional web workloads [4].

social network. To do so, we calculate the fraction of comments on photos that come from the local social network of the uploader. The result of this analysis is that over 28.3% of the comments are placed by friends of the uploader, and at least 89.1% are placed by friends or friends-of-friends (compared to expected values of 0.04% and 0.30%, respectively, were the placement random). This indicates that users are significantly more interested in the content that is uploaded by their friends and friends-of-friends.

Exchange has significant geographic locality Finally, we explore the connection between content exchange and *geographic locality*. Using our Facebook data set, we find that 32.9% of the friends of New Orleans users are also in the New Orleans network; similar findings have been observed in other regional networks [16]. However, if we examine the fraction of content exchange that occurs between New Orleans network users, we observe that 51.3% of comments are placed by other users within the New Orleans regional network, even though only 32.9% of the friendship relationships lie within the network. This indicates that the significant geographic locality already present in social networks is present to an even greater degree in the content exchange that occurs over these networks [17].

Rethinking content distribution

The content that is increasingly being shared on the web today is created at the edge of the network, but is exchanged using centralized infrastructure. The usefulness of existing techniques on this workload is declining [5, 17, 18]: For example, caching the most popular 10% of the items in traditional workloads would satisfy between 55% [4] and 95% [2] of the requests; in our social network workload from the previous section, such a cache would only satisfy 27% of the requests. This also affects the ability to use CDNs, which similarly work best for popular content.

We therefore propose to work towards more decentralized content exchange over the web. While some have suggested decentralizing the provider's data center architecture [17] into many regional data centers, this requires significant changes and expense for the provider. Instead, we propose to focus on retaining the centralized provider architecture of today, while attempting to decentralize content exchange when possible.

In ongoing work, we are building WebCloud, a content distribution system designed to support the workloads present in existing online social networking websites. WebCloud works by recruiting users' web browsers to help serve content to other users and is compatible with the web browsers and web sites of today. Due to the geographic locality that often exists between friends in online social networks [13, 16], content exchange in WebCloud often stays within the user's local Internet Service Provider (ISP), thereby providing a bandwidth savings for both the site and the ISP. As a result, by deploying WebCloud, OSNs such as Facebook would enjoy most of the benefits of large centralized CDNs with lower costs and their users would benefit from faster service.

References

[1] 4,000,000,000 ≪ Flickr Blog. `http://blog.flickr.net/en/2009/10/12/4000000000/`.

[2] M. Arlitt and T. Jin. Workload Characterization of the 1998 World Cup Web Site. *IEEE Network*, 14(3), 2000.

[3] Alexa Top 500 Global Sites. `http://www.alexa.com/topsites`.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. *INFOCOM*, 1999.

[5] G. Cormode and B. Krishnamurthy. Key Differences between Web 1.0 and Web 2.0. *First Monday*, 13(6), 2008.

[6] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. *IMC*, 2007.

[7] X. Cheng, C. Dale, and J. Liu. Statistics and Social Network of YouTube Videos. *IWQoS*, 2008.

[8] Facebook Statistics. `http://www.facebook.com/press/info.php?statistics`.

[9] Facebook and YouTube dominate workplace traffic and bandwidth. `http://www.scmagazineuk.com/facebook-and-youtube-dominate-workplace-traffic-and-bandwidth/article/168082/`.

[10] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. YouTube Traffic Characterization: A View from the Edge. *IMC*, 2007.

[11] N. Kennedy. Facebook's photo storage rewrite. `http://www.niallkennedy.com/blog/2009/04/facebook-haystack.html`.

[12] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cyle. *KDD*, 2009.

[13] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *PNAS*, 102(33), 2005.

[14] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in Online Social Networks. *WSDM*, 2010.

[15] P. Vajgel. Needle in a haystack: efficient storage of billions of photos. `http://www.facebook.com/note.php?note_id=76191543919`.

[16] C. Wilson, B. Boe, A. Sala, K. P.N. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. *EuroSys*, 2009.

[17] M. P. Wittie, V. Pejovic, L. Deek, K. C. Almeroth, and B. Y. Zhao. Exploiting Locality of Interest in Online Social Networks. *CoNEXT*, 2010.

[18] M. Zink, K. Suh, Y. Gu, and J. Kurose. Watch Global, Cache Local: YouTube Network Traffic at a Campus Network - Measurements and Implications. *MMCN*, 2008.

Information Filtering Meets Mobility[1]

Mario A. Nascimento
Dept. of Computing Science
University of Alberta, Canada
mario.nascimento@ualberta.ca

ABSTRACT

The underlying premise in this position paper is that mobility patterns have been relatively underexplored from a data management perspective. More specifically, we discuss how to explore mobility patterns from a large set of users in order to allow proactive discovery and/or suggestion of spatiotemporal events of interest to users. Towards that goal, and considering the semantics of such an application in particular, we suggest three lines of work: (1) how to cluster trajectories, (2) how to find a representative trajectory for trajectory clusters, and (3) how to index trajectories.

1. MOTIVATION

It should be fairly safe to assume that people often move about according to patterns. For instance, the way from home to school/work (and vice-versa) likely follows a few established routes and schedules. A relative large number of people already carry smart-phones with embedded GPS. Ignoring relevant privacy issues for the sake of argumentation, one can envision these devices being used to proactively record one's movements (virtually) all the time. Let us call this type of data Spatio-Temporal Trajectory Patterns (STTPs).

Assuming the above, one can envision the following applications, to mention only a few. Knowing the users' interests (e.g., from their online activities) and their STTPs it becomes feasible to plan beforehand which, when, and where specific ads would be more interesting to be offered, based on the users' preferences and likely location, and without the need to perform (expensive) real-time tracking. "Collective" offers, e.g., instant e-coupons, could be offered to groups of users based on their STTPs, in order to take advantage of their probable proximity to the businesses offering the coupons. Along the same lines, in the context of social networks, targeted e-coupon offers could be sent to groups of online friends who share the same, or reasonably close, STTPs. Another potential application could be to proactively help friends in sharing car rides and/or optimizing them. Finally, users could also be advised, on a need-to-know basis, of road constructions, detours or accidents that may affect their typical daily drive or commute to work, school, etc. It is also conceivable to use STTPs to derive typical behavior of roads (e.g., driving speed as a function of the time of the day), which may be statistically more reliable and up-to-date than potentially stale historical data.

To realize the applications above, let us be inspired by traditional (text-based) information filtering [1]. In information filtering users have a standing query profile that is continuously checked against new data items arriving in a data stream. Once a data item is deemed relevant with respect to a user's profile, that user is somehow alerted about the data item, e.g., a news article. In this case, the user is rather passive as opposed to the typical case where the user pro-actively issues queries. Considering the context just discussed above, one can imagine the situation where the "query" is the user's trajectories and the data stream is a series of spatiotemporal events. This calls for efficient means to efficiently uncover events of interest for users in the context of their STTPs. In the next section we deal with a set of tasks that, collectively, can lead to a solution towards this problem.

We note that an important piece needed for materializing the idea above is learning a person's interest. If we consider the amount and quality of existing work on targeted advertisement based on one's online activities, we are certain that very effective and efficient ways to solve this problem do exist and could, eventually, be re-used. Hence we do not discuss this issue any further in the remainder of this paper.

2. A PROPOSED ROAD MAP

A critical task in addressing the STTP-based information filtering application stated above is to check every event of potential interest against the stored STTPs of all users. A naïve approach, i.e., comparing all STTPs against all events is clearly not practical. To perform this task efficiently we envision the need to (1) determine clusters of trajectories, (2) find representative trajectories for the determined clusters, and (3) index those representative trajectories. In this way, events can be checked against a smaller set of indexed trajectories, thus improving efficiency while maintaining effectiveness. Each of those three components is discussed in turn next.

2.1 Clustering Trajectories

Finding clusters requires one to have a notion of distance between the objects being clustered. In the context of STTPs, it is not sufficient to consider only the distance with respect to the spatial component of a trajectory; the temporal component is also just as relevant. Thus a first task is to determine a suitable distance function that accounts for both temporal and spatial components of trajectories. For that, sophisticated distance functions for time series could to be considered, keeping in mind the need for superior robustness with respect to scaling and shifting, e.g., extending the work presented in [4].

There has been work done in trajectory mining/clustering that we can use as a starting point, e.g., the work by Giannotti et al [5]. In that work however, the authors assume the existence of "locations of interest" which are relevant for all trajectories collectively, not on a per-user basis, as we would need to do in the context of users' STTPs. C.S. Jensen and his team/colleagues

[1] This work has been partially supported by NSERC Canada.

have done other works that may inspire good ideas. For instance, one deals with discovering convoys [6]. A convoy is defined as "a group of objects that have traveled together for some time"; in our case we are looking for STTPs induced by single individuals. Another work deals with constructing accurate routes from GPS data [2], but does not address issues necessary in the context of this proposal, such as identifying patterns of trajectories.

2.2 Determining Representative Trajectories

Once trajectory clusters have been determined, the next task is to determine a representative trajectory for each cluster. At the information filtering stage these representative trajectories alone, thus a relatively small set, will be used to determine the relevance of an event with respect to the STTPs of a group of users (clusters). There are a few possibilities to accomplish this, from computing an "average" trajectory to re-using an existing one that minimizes the error with respect to the others.

In both tasks above one must consider that STTPs are bound to change over time, for instance, in case of road constructions or collective change in behavior (e.g., during a long holiday weekend). Therefore, cost-effective means to keep large clusters of STTP, as well as their representatives, up-to-date and consistent, needs to be investigated as well. We are not aware of any previous research on how to solve this problem.

2.3 Indexing Trajectories

The third task is to index the obtained representative trajectories. The potentially large number of events, actually a stream thereof, needs to be checked against every representative trajectory, which is bound to be a very large dataset. Hence, this task must be performed very efficiently using an index. Again, a considerable amount of research has been done in the topic of trajectory indexing, including some more suitable for DBMS integration, e.g., [8, 9], based on the well-known R-tree.

All those need to be considered, but it is likely that a new indexing structure, based on the semantics of the application at hand, need to be designed. By semantics of the application we note that not only high scalability, a typical requirement, is important, but sustainable high throughput is also essential. A possible venue for work, that is relatively underexplored, is related to the use of Flash-based Disks (SSDs) [7].

3. ENTER THE CLOUD …

All tasks above can be accomplished in a "typical" centralized computing model. However, it is only natural to (re)consider all of them within the realm a computing cloud. For instance, one possible research venue is the massive parallelization of the proposed indices, e.g., using the Map-Reduce paradigm, e.g., [3].

Cloud storage itself is another issue to be considered. One cannot ignore that, underlying the development of all tasks above, one must explicitly address the fact that fairly large and dynamic trajectory datasets will need to be stored, as well as manipulated in order to facilitate the clustering (mining) and indexing tasks. Although one can initially envision most of the processing being done offline, a practical system needs to be able to ingest streaming data to better accommodate changes in the identified patterns and/or in the events of interest in a timely and likely distributed manner.

4. FURTHER WORK

There are several directions in which the research outlined here can be extended. For instance, it would be also interesting to determine common sub-trajectories among STTPs. This would be of use, for instance, for city officials interested in minimizing traffic disruption in case of road construction or in order to maximize the return of expansion investments. Another application would be choosing time and location for road-side electronic advertisement. These types of applications would greatly benefit from the framework developed for the research being suggested here.

Another direction for work, although not directly related to the above application, is the use of STTPs for hop-wise routing of queries and their answers; some works have already considered using user-carried mobile devices as sensors, e.g., [9], which can be seen as either data sources and sinks. For instance, one can issue a query about a given location, and obtain readings from a sensor that has just been (or will shortly be) at the location of interest. This would ensure a fresh answer at relatively low cost provided that the user is able to tolerate some query latency. We are currently using STTPs for finding encounter patterns and then use those patterns for optimizing energy cost or data latency in hop-wise query/data routing.

Finally, a perhaps more visionary idea is to use mobile devices, e.g., smartphones, themselves as components of a cloud. In fact, considering for instance that Cisco estimates that there will be "more than 7 billion mobile devices globally by 2015" and considering that the "total internet traffic will more than quadruple by 2014"[2], this is a path to be considered. Certainly much more progress is needed towards energy management, privacy and data security, but with the ever-increasing capabilities of such devices, this possibility should not be overlooked.

5. REFERENCES

[1] N.J. Belkin et al: Information Filtering and Information Retrieval: Two Sides of the Same Coin? CACM 35(12): 29-38 (1992).

[2] A. Brilingaite et al: Enabling routes as context in mobile services. GIS 2004: 127-136.

[3] A. Cary et al: Experiences on Processing Spatial Data with MapReduce. SSDBM 2009: 302-319

[4] Y. Chen et al: SpADe: On Shape-based Pattern Detection in Streaming Time Series. ICDE 2007: 786-795.

[5] F. Giannotti et al: Trajectory pattern mining. KDD 2007: 330-339.

[6] H. Jeung et al: Discovery of convoys in trajectory databases. PVLDB 1(1): 1068-1080 (2008).

[7] M. Sarwat et al: FAST: A Generic Framework for Flash-Aware Spatial Trees. SSTD 2011: 149-167.

[8] D. Pfoser, C.S. Jensen: Trajectory Indexing Using Movement Constraints. GeoInformatica 9(2): 93-115 (2005).

[9] S. Rasetic, et al: A Trajectory Splitting Model for Efficient Spatio-Temporal Indexing. VLDB 2005: 934-945.

[10] Sasank R. et al: MobiSense - mobile network services for coordinated participatory sensing. ISADS 2009: 231-236.

[2]`http://www.telegraph.co.uk/technology/internet/9051590/50-billion-devices-online-by-2020.html`

A Case for CACE

Suman Nath

Microsoft Research

Recent years have seen two significant trends in the computing landscape: an increasing availability of sensors-integrated smartphones and the Cloud to which phones can be continuously connected to. This provides an opportunity for a novel class of applications, which we call *Context-Aware Cloud-Edge* (CACE) applications. Such applications distribute over many smart *edge-devices* (e.g., smartphones) and the Cloud, and react based on the operating conditions of users, their social states, and the surrounding environment. A canonical example is a *friend-finder* app on a phone (e.g., the *Loopt* app on iPhone, Android, and Windows Phone) that notifies a user whenever any of her friends are near her current location. Other examples include *connected cars*[1] that provides location-aware telematics and data analytics over a large number of cars connected to the Cloud, location-aware coupon services (e.g., *GeoQpons* on Android and iPhone) that notify a user when she is close to a business offering coupons that she might like, mobile multiplayer games (e.g., *iMobsters* on Android) that monitor and react on players' mutual interactions and status, and in-car dashboard apps that provide real-time route and gas station recommendations.

There are two main ingredients in these applications: smart edge-devices and the Cloud, both of which have become ubiquitous by recent technological developments. We believe that many more such applications will appear in coming years. Therefore, it makes sense to build a common platform that captures the common ingredients and allows an application developer to quickly prototype a new application. Building such a platform, however, requires addressing several challenges.

Reliable Context Inference. Physical context is at the core of CACE applications. Some recent works have shown how to efficiently use sensors on smart phones to infer various context attributes such as a user's location, transportation mode, social states, etc. However, existing solutions apply only to a small set of context attributes and many of the solutions are not robust enough to be used as simple plug-and-play. Providing robust and simple plug-and-play solutions for a rich collection of context attributes can enable many new useful applications.

Program Specification. Writing a Cloud-Edge application is itself oftentimes non-trivial. The usual practice is to write them in standard procedural or object-oriented languages such as Objective C, Java, or C#. This puts a high burden on application developers, as they often need to implement a significant part of the core application logic (e.g., filtering and correlating data feeds from various devices), which is often distributed in nature, from the scratch. Further, both the data source and the application logic usually have a significant temporal component. Specifying and processing distributed temporal logic can be hard even for experienced developers. A simple declarative programming framework can significantly simply this task.

[1]`http://thinkd2c.wordpress.com/2011/04/07/microsoft-and-toyota-connected-car-in-the-cloud/`

Efficient execution. It is challenging to tune a Cloud-Edge application to make efficient use of resources such as energy and communication bandwidth of edge-devices. Currently, application developers implement various ad hoc resource optimization techniques such as duty cycling, offloading expensive computation the Cloud, compressing data during transmission, etc. However, implementing such optimizations is often nontrivial. The situation aggravates when an application needs to compare or correlate data from multiple edge devices. For such applications, resource optimization involves deciding what computation to push, and to which edge device. Such computation placement decisions require solving optimization problems involving various factors such as the network topology, rates of the data streams, data upload and download costs, pairs of streams to correlate, etc. Moreover, since these parameters can change over time, the decision needs to be dynamically updated. The complexity of implementing such optimizations often outweighs the cost of developing the core functionality of an application.

Privacy. Context-aware applications require users to release their contexts, which raises serious privacy concerns. Recent works have proposed solutions based on suppressing contexts or releasing modified (e.g., generalized) contexts. However, in many scenarios, these techniques do not necessarily prevent an adversary from inferring sensitive contexts. Suppression itself can leak information. Released nonsensitive contexts can reveal information about sensitive context to an adversary who knows temporal correlation between sensitive and nonsensitive contexts. A formal framework to guarantee privacy against such strong adversary is necessary for widespread adoption of context-aware applications.

At Microsoft Research, we have been working on various solutions to address these challenges [1, 2, 3, 5, 4, 6]. For details, please check `http://research.microsoft.com/~sumann`.

References

[1] Hossein Ahmadi, Nam Pham, Raghu Ganti, Tarek Abdelzaher, Suman Nath, and Jiawei Han. Privacy-aware regression modeling of participatory sensing data. In *ACM SenSys*, 2010.

[2] Badrish Chandramouli, Joris Claessens, Suman Nath, Ivo Santos, and Wenchao Zhou. RACE: Real-time applications over cloud-edge. In *ACM SIGMOD*, 2012.

[3] Michaela Goetz, Suman Nath, and Johannes Gehrke. MaskIt: Privately releasing user context streams for personalized mobile applications. In *ACM SIGMOD*, 2012.

[4] Alexandra Meliou, Wolfgang Gatterbauer, Suman Nath, and Dan Suciu. Tracing data errors with view-conditioned causality. In *ACM SIGMOD*, 2011.

[5] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *ACM SIGMOD*, 2010.

[6] Arsalan Tavakoli, Aman Kansal, and Suman Nath. On-line sensing task optimization for shared sensors. In *ACM/IEEE IPSN*, 2010.

Sampling from Social Activity Networks to Understand User Behavior and Systems

Jennifer Neville
Departments of Computer Science and Statistics
Purdue University
West Lafayette, IN 47907
neville@cs.purdue.edu

Online social activity and interaction is becoming embedded into the fabric of our society. From electronic communication (e.g., email, IMS) to social media (e.g., blogs, wikis) to online content sharing (e.g., facebook, flicker, youtube)—we are currently undergoing an explosive growth in the manner and frequency in which people interact online, both with each other and with content.

The ability to collect and analyze large-scale, complex datasets has recently transformed the fields of computational biology and physics—and the explosive growth of social interactions online offer the potential for social science to undergo a similar transformation. Since the traces of electronic activity, including the production and consumption of content, provides a wealth of data that is much more extensive (and more cost effective to collect/observe) than what has previously been studied in social science domains, it could be used to vastly improve our understanding of interpersonal behavior, social processes, and decision making. At the same time, from a systems perspective, the use patterns in social systems (i.e., structure of content and traffic) may be quite different from other networks/traffic on the Internet. The online activity data can thus be used to understand social behavior as it relates to both Internet traffic and load on the infrastructure of online social networks, in order to drive the development of appropriate models, tools and systems to manage and maintain online content and interactions.

However, the size and scope of online interaction data make it impractical to collect and study *complete* datasets. In 2009, for example, Facebook reported that the number of chat messages had exceeded one billion per day. Thus, network *sampling* methods are critical to selecting a subset of the data for study. Much of the previous research on social network sampling has focused on algorithm development, with the aim of accurately and efficiently select nodes/edges/subgraphs from a single large graph (Leskovec & Faloutsos 2006; Hubler *et al.* 2008; Ribeiro & Towsley 2010; Maiya & Berger-Wolf 2011; Ahmed *et al.* 2011).

For example, consider an input graph $G = (V, E)$ of size $n = |V|$. Then the goal is for the sampling algorithm to select a subgraph $G_s = (V_s, E_s)$ with a subset of the nodes ($V_s \subset V$) and/or edges ($E_s \subset E$), such that $|V_s| = \phi n$, where $\phi < 1$ is the sampling fraction. In some cases, the aim is to use G_s to estimate parameters of the full graph (e.g., degree distribution). In other cases, the aim is to have G_s be a *representative* subgraph from the the full graph. Since a complete graph is rarely available for evaluation, the proposed sampling methods are typically assessed by measuring the similarity between characteristics of selected sample and those of the input graph, which is inevitably a sample itself. Key technical challenges that have been investigated include:

- How to sample when the data are heterogeneous and interdependent (e.g., networks are sparse, but heavy-tailed with clustering) (Leskovec & Faloutsos 2006; Hubler *et al.* 2008; Maiya & Berger-Wolf 2011).
- How to sample without knowledge of the full graph (e.g., users are only visible through queries) (Ribeiro & Towsley 2010).
- How to sample in a dynamic environment when there are not enough resources to store the full graph (e.g., in graph *streams*) (Ahmed *et al.* 2011).

However, these efforts have generally not considered the larger issue of how sampling impacts the analysis and understanding of social processes and performance of social systems (e.g., the performance of a new routing protocol for an OSN system, or the accuracy of a viral marketing model). In particular, they have focused more on preserving properties of the network structure, rather than on providing accurate assessment of the properties of processes *overlaid* on the network structure. Although in some cases, preserving aspects of the network topology in a sample may be *sufficient* to accurately estimate the characteristics of processes overlaid on the network, it may not be *necessary*, nor may it be the only manner in which we can accurately estimate performance.

Moreover, there has been relatively little attention paid to developing the theoretical foundation for sampling from *network processes* that would drive the investigation of these types of questions. For example, if the aim is use G_s to evaluate performance of a process $f(.)$ on a larger graph G (where $n >> m$), then the algorithm evaluation should assess sample "representativeness" by estimating an empirical distribution for the process overlaid on the generated samples $\hat{P}(f(G_m))$ and compare it to process in the original graph $P(f(G))$. For example, if $f(.)$ is a diffusion process that models the spread of information in a social network, then we would like our evaluation of f in G_s to accurately reflect the diffusion properties of f that would be observed in the full graph G. However, to begin to formulate and assess sampling algorithms in this manner, we need a more precise description, and better understanding, of graph *populations*—both with respect to the distribution of possible worlds and their dynamics/evolution over time.

A *statistical population* is typically defined as the set of all items that one wishes to study. When the object of study is an entire network, the population should be defined as a set of networks of a specified size (e.g., n), or the set of networks that could be generated by the same underlying process that created the input network G. In practice, we can rarely observe multiple networks from the same social network domain. There is only one Facebook friendship graph, one Flicker graph—although we can down-sample many smaller networks from these large networks, we cannot measure a second, independent instance. Instead, these networks correspond to *complex systems* evolving over time. Therefore, it is more reasonable to define the population through the process that underlies the formation of the networks. Although it is still an open question as to how to model the *generative* processes of network structure probabilistically, this will be critical to the investigation of sampling methods and their impact on subsequent network analysis.

For example, since many graph characteristics are *not independent* of graph size, it is not clear what structure in the smaller subgraphs will give an accurate estimate of the performance in larger graphs (as they evolve over time). Consider the case where the original network consists of n nodes and we construct a 10% sample (i.e., $|V_s| = 0.1n$). Let $|E_o|$ and $|E_s|$ be the number of edges in the original and the sampled network respectively, and let the density in the original network be $\frac{|E_o|}{n(n-1)}$. Then if we match the density in the sampled graph: $\frac{|E_s|}{0.1n(0.1n-1)} = \frac{|E_o|}{n(n-1)}$, the number of sample edges will be: $|E_s| \simeq 0.1^2|E_o| < 0.1|E_o|$. This shows the dependency of graph metrics on graph size—the number of nodes grows linearly, but the number of possible edges grows quadratically (in n). In G, the average degree is $\bar{d}_o = \frac{|E_o|}{n}$. On the sample subgraph G_s, if the density is equal to that of the original graph, then the average degree will be underestimated $\bar{d}_s = \frac{|E_s|}{0.1n} \simeq \frac{0.1|E_o|}{n}$. Similarly, if we aim to capture the original average degree in the sample, then the density will be overestimated. It is not clear which metric to optimize to select "better" sample graphs for evaluation of performance.

Moreover, since none of the recent work on graph sampling includes an explicit definition of the population of interest or a description of the set of events under consideration, this has led to *subjective* evaluations of algorithm performance where the similarity of the sample to the original graph is used as an indirect proxy for representativeness. *Representativeness* of a sample subgraph G_s should be measured through the likelihood of G_s given the underlying process that generated G. The primary assumption with the current proxy evaluation is that when the sampled network exhibits graph metrics similar to the original input network, then the sample is "close" to the mode of the distribution. However, since the underlying distribution is not formally defined, it is not clear whether this assumption holds in practice. Moreover, when the statistics do not match exactly (as is most often the case), a secondary assumption is that "closer" implies "more" representative. Again, it is not clear whether this holds for real world network processes (e.g., we do not know how much variance is expected in real-world network systems).

To develop a better understanding of how sample structure affects the analysis of behavior and performance in larger networks, more attention needs to be paid to the exploration of correlations among graphs properties, how they evolve given a specific generative mechanism, and how to model probability distributions over graph processes. We note that current statistical models of graphs focus on modeling graphs of a specific size (i.e., the number of nodes is fixed). Size independent graph models exist (e.g., Lovász's graphon), but the current state of the art does not address sparse graphs, nor are there any formal notions of the statistical properties of graph *processes* (e.g., stationarity).

Network sampling is critical for analyzing online social interaction data, both for system development and for investigation and refinement of social theories. We note that since almost *every* network dataset is a sample of network data, the sampling method can impact the accuracy of analysis even when researchers have not explicitly considered *how* to sample. The key aspect of the data—the relationships among users, content, and applications—is also the characteristic that makes it difficult to guarantee unbiased "representative" samples, since local dependencies combine in complex ways to produce global structure. Thus, in order to drive both the advancement of computational social science and the development of robust and reliable social computing systems, more research needs to focus on developing:

- A formal framework for sampling from heterogeneous, partially-observed, interdependent data.
- An understanding of various network characteristics and their dependencies.
- Probabilistic models of dynamic graph processes, that can model network structure as it evolves over time.
- An analysis of the impact of sample *representativeness* on the investigation of social processes and/or system protocols overlaid on the networks.

References

Ahmed, N.; ; Neville, J.; and Kompella, R. 2011. Network sampling via edge-based node selection with graph induction. Technical Report 11-016, CS Dept, Purdue University.

Hubler, C.; Kriegel, H.-P.; Borgwardt, K. M.; and Ghahramani, Z. 2008. Metropolis algorithms for representative subgraph sampling. In *ICDM*.

Leskovec, J., and Faloutsos, C. 2006. Sampling from large graphs. In *SIGKDD*, 631–636.

Maiya, A. S., and Berger-Wolf, T. Y. 2011. Benefits of bias: Towards better characterization of network sampling. In *SIGKDD*.

Ribeiro, B., and Towsley, D. 2010. Estimating and sampling graphs with multidimensional random walks. In *ACM SIGCOMM Internet Measurement Conference*.

Seamless Mobile Services in the Cloud

Jagan Sankaranarayanan Hakan Hacıgümüş
NEC Laboratories America
10080N Wolfe Rd SW#3-350
Cupertino, CA 95014
Email: {jagan,hakan}@sv.nec-labs.com

***Abstract*—The mobility of today is defined by the multitude of apps, which while working in isolation, can achieve a variety of tasks for the mobile user. The mobility of tomorrow is envisioned as one where mobile apps work together by sharing information to create a seamless mobile experience, where the focus is the mobility of the user but not the device. The ultimate goal of mobility is to create a seamless mobile experience, which can be achieved by mobile apps sharing data actively with one another in a cloud ecosystem.**

I. Introduction

The ultimate goal of mobility is to ensure that the experience of a mobile user is a rich one. This means that individual apps (i.e., mobile services) should interact with the mobile user as if they exist in an ecosystem whose collective objective is to ensure that the mobile user can interact with the digital world in a *seamless* fashion. Mobile users frequently change their context as they navigate in their fast-paced daily lives, as described in an example scenario below. The desire is to ensure that mobile users stay connected to the digital world through mobile services as they move forth from one context to another – regardless of the kind of mobile device and sometimes even without a mobile device, which is largely irrelevant here. In this environment, the main constraints of mobility are the limited time, patience, and attentive span of the mobile user who is *on the go*. Although there have been recent efforts to significantly advance the capabilities of mobile devices to improve the user experience, more substantial improvements in user experience can be achieved if individual apps are cognizant to the limitations of the mobile user. In some sense, we want to move beyond traditional arguments that solely attribute the challenges of mobility to the limitations of the mobile device, but instead focus on how apps can provide a much better user experience. Apps that constantly adapt to the current context of the mobile users are said to exhibit "*seamless mobility*" [1].

The idea of seamless mobility is best described by a use case scenario, given in Figure 1. Our scenario begins with a user receiving an email confirmation of acceptance of a paper to a conference. The email app pulls out the event information and populates the calendar app with the relevant information. Later when the mobile user invokes an airline booking app to purchase tickets, the app has access to the conference details, using it to determine if the user will be traveling, which airport the user is flying to/from etc. The mobile user's interaction with the airline ticket booking app is the first instance of a seamless mobile experience, which happened because the

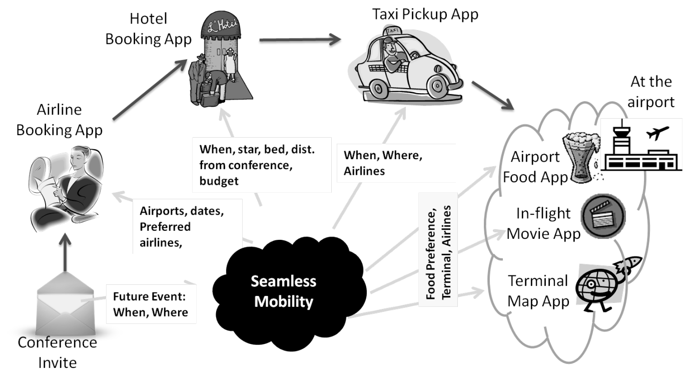


Fig. 1. A case scenario of mobile apps working together to create a seamless mobile experience for a user

calendar app shared information with the airline booking app to make the airline reservation process intuitive for the user. Of course, such a sharing arrangement between the two apps can never be perfect all the time as the user may want to travel on different dates, or use another airline; all these issues can be resolved by suitable user interface options on the app. Moreover, the goal of this example is not delve into the virtues of incorporating context into apps, which is quite evident, but instead motivate the utility of apps sharing information. Continuing with our example, next if the user invokes a hotel reservation app, it has access to the conference details, dates of travel, air ticket, and other information from the calendar and airline reservation apps, and tries to present the user with hotel options that are proximate to the conference venue as well as being on the same dates as the conference. Subsequently, when the user invokes an app to reserve a taxi ride to the airport, the app has all the required information such as the place to pick up and drop as well as relevant parts of the user's itinerary. When the user is at the airport, and looks for restaurants to dine, the app shows options from the same terminal where the user is currently located, but not showing those that the user did not prefer in the past.

The seamless mobility in our scenario is achieved by apps working together to help the mobile user achieve tasks on the go. The apps in our scenario greatly benefited from sharing information with one another. The mobility as we envision is far from what currently exists. Even though there is a rich variety of apps on mobile devices, they generally do not talk to one another let alone share information to create a seamless mobile experience. The communication between apps is *adhoc* in the sense that there is no real support to enable them.

For example, Yelp, which is a restaurant review app, allows users to post their reviews of restaurants using the Facebook app, which is a popular social networking service. As far as we can tell, the communication between Yelp and Facebook apps is facilitated by Yelp accessing the Facebook API. The problem with this setup is that it is unidirectional and not scalable in the sense that every app needs to implement the API individually. In general, APIs are expensive to create and maintain, not to mention that they may not be expressive enough for most sharing needs between apps. Moreover, as apps are often hosted in the same cloud infrastructure, there are other less expensive ways of enabling sharing of data between apps hosted in the cloud. While acknowledging the enormous security and privacy implications of apps sharing personal information of the mobile user, these issues are beyond the scope of this paper.

II. Sharing Service in the Cloud

Mobile apps can be viewed as front ends driven from remote services, which are typically hosted on the cloud. For example, a weather app on a mobile device is essentially a front end that queries a data store on the cloud infrastructure for the weather conditions at a certain zip code. Platform as a Service (PaaS) [2] provides hosting, processing and querying of data for any mobile app that wishes to use its services. A PaaS is the most appropriate place to build a service for sharing as it typically hosts data from several other apps. Sharing between the apps can be provided as a service with little or no overhead to the apps that use it. Sharing, in our context, adds value to all the parties involved by providing access to richer information on the mobile user. In the PaaS setting, mobile apps that use the PaaS to host their databases are referred to as *tenants*. Usually a *PaaS provider* hosts several tenants in the same cloud infrastructure. In other words, the PaaS provider usually resorts to *multitenancy* for good resource usage and spreading of the operation cost among several tenants.

There are several ways of enabling sharing between tenants in the cloud. The sharing in the cloud is usually between a tenant t, who is the *owner* of the data and another tenant, referred to as a *consumer*, who wants access to t's data. Consider a scenario of two apps, say App-A and App-B, hosted on the same cloud infrastructure that agree to share data. In particular, let us only consider the case of App-A, who is the data owner, agreeing to share data with App-B who is the consumer. For instance, App-A could be a calendar service, while App-B could be an airline ticket booking service that wants to query calendar appointments to determine if the user is traveling in the near future.

If App-A wants to share some of its data with App-B, in a traditional scenario, App-A would create an API and share the details of the API with App-B. The advantage of this model is that App-A is *loosely coupled* with App-B in the sense that App-A is free to change its data layout without really affecting App-B as long as the API is suitably updated. However, App-A must setup the necessary infrastructure to create the API as well as keep updating it whenever its data layout changes. An extreme solution would be if App-A allows App-B to access its data directly. The drawback of this arrangement is that it leads to a tight coupling between App-A and App-B. Moreover, if App-B is a *hard-hitter* (i.e., issues queries at a high rate) of App-A's data, which would lead App-A to have poor access on its own data. The PaaS provider can setup a *materialized* shared space for App-B, which ensures that App-B's access on the shared space will not significantly affect App-A's queries. Of course, materialized shared space takes up storage and is expensive to maintain so this solution, while attractive, must be used intelligently.

We therefore need a sharing service offering for mobile apps (e.g., COSMOS [1]) with the goal of supporting seamless mobility by enabling wide scale sharing between apps. To ensure that all the tenants get an acceptable level of service, in spite of sharing the infrastructure with several others, tenants will negotiate *Service Level Agreements* (SLA) with the PaaS provider. SLA is a contract that describes the level of service a tenant requires on the data hosted with the PaaS. For example, an SLA could specify that the tenant would pay 10 cents for queries responded within 300ms, while the tenant would penalize the PaaS $1 if the execution time for the query exceeds 300ms. The PaaS provider, whose objective is to maximize profits, enables sharing between apps while ensure that the tenants do not miss their SLA deadlines too often as that results in a loss of revenue.

III. Concluding Remarks

This paper laid out a vision of mobility, where apps collaborate to create seamless mobility for the mobile user. As mobile users have a common identity across all the mobile apps, sharing information between apps can lead to the development of interesting services as well as a much richer experience for the mobile user. From an intuitive point of view, sharing creates rich data and the utility of rich data can be readily seen by considering the following three classes of apps:

- Free apps: These apps are usually supported by advertisements. A richer data on the users means more targeted advertisements.
- Paid apps: Richer data in turns means more compelling features, which could be a good incentive for users to pay for these apps.
- Enterprise apps: Sharing support in the enterprise cloud means reduced enterprise silos.

Therefore, there is a natural incentive for mobile apps to share information with one another, even with the current limited availability of services to aid sharing. We believe that by building support for large-scale sharing between mobile apps, we can usher in an era of seamless mobility.

References

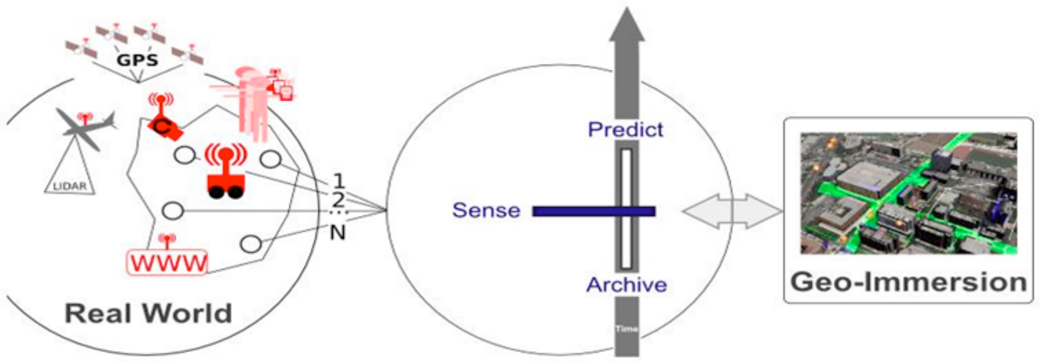
[1] J. Sankaranarayanan, H. Hacigumus, and J. Tatemura, "COSMOS: A platform for seamless mobile services in the cloud," in *Proceedings of the International Conference on Mobile Data Management (MDM)*, vol. 1, Jun. 2011, pp. 303–312.

[2] Hacıgümüş, J. Tatemura, W. Hsiung, H. J. Moon, O. Po, A. Sawires, Y. Chi, and H. Jafarpour, "CloudDB: One size fits all revived," in *IEEE World Congress on Services (SERVICES)*, Miami, FL, Jul. 2010.

Geo-Immersion: A Killer-App for Cloud Computing, Social-Networks and Mobile Computing

Cyrus Shahabi
University of Southern California
Integrated Media Systems Center
Los Angeles, CA 90089
shahabi@usc.edu
http://imsc.usc.edu

Since July 2010, at the University of Southern California's (USC) Integrated Media Systems Center (IMSC), a graduated National Science Foundation (NSF) Engineering Research Center (ERC), we have been focusing on a new geo-socio-temporal computing paradigm, termed *Geo-Immersion*. Geo-Immersion enables humans to capture, model and integrate real-world data into a geo-realistic virtual replica of the world for immersive data access, querying and analysis. It encompasses research from many interesting areas including cloud computing, social-networks and mobile computing. To put simply, the main theme of Geo-Immersion is to *blend the real and virtual worlds*.



Geo-Immersion is a much broader concept than that of the previous fields of augmented-reality, virtual-reality, etc. The reason is that the focuses of these past concepts were mainly on the computer-graphics and visualization aspects of Geo-Immersion. However, the more exciting and emerging topics are now the fusion of human behaviors in these two worlds. Hence, Geo-Immersion is more than a research topic and I would go as far as to categorize it as a new *computing paradigm*. Let me elaborate.

The main task of the first generation of computers was "computation", for example, computing differential equations. This changed in late 1960′s with the advent of ARPANET's university network where the task of "communication" was added to the major tasks of computers. In fact, computers were still performing computation in order to enable communication, but computing went to the background for the seamless support of the new task. The third generation computers, in early 1990′s, enabled "information access" through the Web. This time, communication took the backseat in support of information-access.

I believe that the next-generation of computers will be tasked to blend the real world with the virtual world, i.e., Geo-Immersion. We already witness this through the excitements over location-based-services, social-networks, participatory-sensing (crowdsourcing), and cloud computing. This new paradigm uses the four dimensions of *what, when, where and who*, which enable people to naturally operate in a hybrid virtual-real world. After all, human brain is wired to operate in time and space. For example, I am obsessed with planning my future based on where and when. Then why not use the same concepts in the virtual world to both operate more naturally (by transparently accessing information, communicating and computing) and better integrate the real-world data, phenomena and observations into the virtual world.

To illustrate my point and also the relationship of Geo-Immersion with the topics of this workshop, consider two prominent uses of computers by public: social networking and mobile-apps (e.g., on smart-phones or tablets). I consider social networking as bringing the *real* world and its social fabric to the *virtual* world. Meanwhile, mobile-apps bring the *virtual* world and its efficiency and flexibility to the *real* world. This is the fusion envisioned by Geo-Immersion.

The applications of Geo-Immersion are plenty: urban security, disaster management and rescue-response, military intelligence, urban planning and real-estate, intelligent transportation, simulation and training, public health in urban area, sustainable design, etc. At IMSC, we have been working on some of these applications. To further illustrate the concept and its relationship to this workshop, I briefly explain our work in intelligent transportation.

In the past two years, as part of IMSC's contract with Los Angeles Metropolitan Transportation Authority (MTA), we have been given access to a very large-scale and high-resolution (both spatial and temporal) transportation data from LA County road network. This dataset includes traffic flows recorded by under-pavement loop detectors, police reports, videos and images from CCTV cameras, and operational public transit data such as passenger counts and buses' locations. We have **virtualized** these **real world** datasets by developing an end-to-end system called *TransDec* (for Transportation Decision-making). The backend of TransDec is a **cloud platform**; in particular, all the data (except for videos) are saved to Microsoft Azure storage space. In order to allow processing of queries on such a large data set efficiently, the data is aggregated to create sketches for supporting predefined set of spatial and temporal queries. A Microsoft StreamInsight server, which resides on the Azure AppFabric, handles this aggregation process. The sketches are then written to SQLAzure. Other than infrastructural benefits, this eliminates communication and data-transfer costs to a cloud storage platform. The collection, its refinement, and required geostreaming queries of the traffic data are also implemented on Microsoft's StreamInsight. In other projects, we showed that other relevant datasets (e.g., accidents, traffic reports) can also be contributed by people using their smart phones and through their **social-networks**. Finally, the front-end of TransDec is a mash-up (developed on both Microsoft Bing and Google Map) that is accessible through desktop and **mobile platforms**. For example, we are developing a next-generation route-planning application for smart-phones dubbed ClearPath. ClearPath can quickly find the fastest way to get from Point A to Point B, by taking real-time and future traffic congestion into consideration. ClearPath will save commuters time and money, and make delivery businesses more efficient.

In conclusion, I believe blending the real and virtual worlds through Geo-Immersion, is a killer application for cloud computing, social-networks and mobile computing. It is a new computing paradigm that dominates the way the public utilizes computers in recent years (through their mobile apps and social networks) and has many useful real world applications. It also encompasses research from many interesting areas such as multimedia, participatory-sensing, privacy, trust, web, geospatial and temporal data management, etc. But more importantly, it brings up new fundamental research challenges in computer and social sciences to study the fusion of human behaviors in the real and virtual worlds. For example, how would one blends social-networks (represented as a graph) with geospatial (represented as 2D or 3D space) and temporal (represented as points or intervals) spaces? Is it possible to derive social-networks by analyzing people's movements in time and space? At IMSC, we have just started scratching the surface of this transformative paradigm. But above all, this paradigm enables people to "connect" across time and space. Isn't this what humanity is all about after all?

Spatial Big-Data Challenges Intersecting Mobility and Cloud Computing

Shashi Shekhar, Michael R. Evans, Viswanath Gunturi, KwangSoo Yang

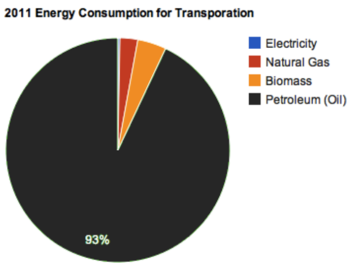
{shekhar,mevans,gunturi,ksyang}@cs.umn.edu
Computer Science & Eng. Faculty, University of Minnesota
200 Union Street S.E. #4192, Minneapolis, MN 55455

1. **INTRODUCTION**

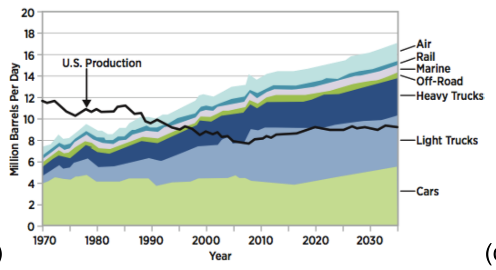
Mobility is efficient, safe and affordable travel in our cities, towns and other places of interest [52]. Mobility services, e.g., routing and navigation, are a set of ideas and technologies that facilitate understanding the geo-physical world, knowing and communicating relations to places in that world, and navigating through those places. The transformational potential of mobility services is already evident. From Google Maps [17] to consumer Global Positioning System (GPS) devices, society has benefited immensely from mobility services and technology. Scientists use GPS to track endangered species to better understand behavior, and farmers use GPS for precision agriculture to increase crop yields while reducing costs. We've reached the point where a hiker in Yellowstone, a biker in Minneapolis, and a taxi driver in Manhattan know precisely where they are, their nearby points of interest, and how to reach their destinations.

Increasingly, however, the size, variety, and update rate of mobility datasets exceed the capacity of commonly used spatial computing and spatial database technologies to learn, manage, and process the data with reasonable effort. Such data is known as Spatial Big Data (SBD). We believe that harnessing SBD represents the next generation of mobility services. Examples of emerging SBD datasets include temporally detailed (TD) roadmaps that provide speeds every minute for every road-segment, GPS trace data from cell-phones, and engine measurements of fuel consumption, greenhouse gas (GHG) emissions, etc. SBD has transformative potential. For example, a 2011 McKinsey Global Institute report estimates savings of "about $600 billion annually by 2020" in terms of fuel and time saved [26,29] by helping vehicles avoid congestion and reduce idling at red lights or left turns. Preliminary evidence for the transformative potential includes the experience of UPS, which saves millions of gallons of fuel by simply avoiding left turns (Figure 1(a)) and associated engine idling when selecting routes [26]. Immense savings in fuel-cost and GHG emission are possible if other fleet owners and consumers avoided left-turns and other hot spots of idling, low fuel-efficiency, and congestion. Ideas advanced in this paper may facilitate '*eco-routing*' to help identify routes that reduce fuel consumption and GHG emissions, as compared to traditional route services reducing distance travelled or travel-time. It has the potential to significantly reduce US consumption of petroleum, the dominant source of energy for transportation (Figure 1(b)). It may even reduce the gap between domestic petroleum consumption and production (Figure 1(c)), helping bring the nation closer to the goal of energy independence.

(a)



(b)



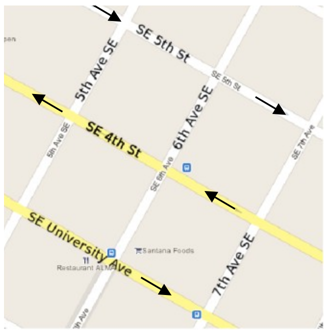
(c)

Figure 1: (Left) UPS avoids left-turns to save fuel [26]. (Middle) Petroleum is dominant energy source for US Transportation [49]. (Right) Gap between US petroleum consumption and production levels is large and growing [3,9]. (Best in color)

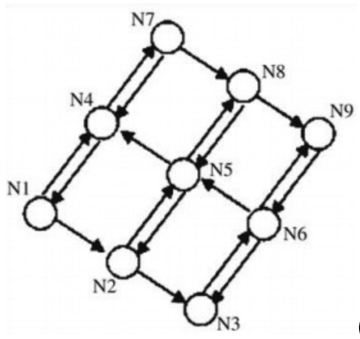
However, SBD raises new challenges for the state of the art in spatial computing for mobility services such as routing. First, it requires a change in frame of reference, moving from a global snapshot perspective to the perspective of the individual object traveling through a road network. Second, SBD increases the impact of the partial nature of traditional route query specification. It significantly increases computation cost due to the tremendous growth in the set of preference functions beyond travel-distance and travel-time to include fuel consumption, GHG emissions, travel-times for thousands of possible start-times, etc. Third, the growing diversity of SBD sources makes it less likely that single algorithms, working on specific spatial datasets, will be sufficient to discover answers appropriate for all situations. Other challenges include geo-sensing, privacy, prediction, etc.

2. TRADITIONAL MOBILITY SERVICES

Traditional mobility services utilize digital road maps [18, 31, 33, 42]. Figure 2(a) shows a physical road map and Figure 2(b) shows its digital, i.e., graph-based, representation. Road intersections are often modeled as vertices and the road segments connecting adjacent intersections are represented as edges in the graph. For example, the intersection of `SE 5th Ave' and `SE University Ave' is modeled as node N1. The segment of `SE 5th Ave' between `SE University Ave' and `SE 4th Street' is represented by the edge N1-N4. The directions on the edges indicate the permitted traffic directions on the road segments. Digital roadmaps also include additional attributes for road-intersections (e.g., turn restrictions) and road-segments (e.g., centerlines, road-classification, speed-limit, historic speed, historic travel time, address-ranges, etc.) Figure 2(c) shows a tabular representation of the digital road map. Additional attributes are shown in the node and edge tables respectively. For example, the entry for edge E1 (N1-N2) in the edges table shows its speed and distance. Such datasets include roughly 100 million (108) edges for the roads in the U.S.A. [31].



(a)



(b)

Nodes

NID
N1
N2
N3
N4
N5
N6
N7
N8
N9

Edges

EID	From	To	Speed	Distance
E1	N1	N2	35mph	0.075mi
E2	N1	N4	30mph	0.075mi
E3	N2	N3	35mph	0.078mi
E4	N2	N5	30mph	0.078mi
E5	N3	N6	30mph	0.077mi
E6	N4	N1	30mph	0.075mi
E7	N4	N7	30mph	0.078mi
E8	N5	N2	30mph	0.078mi
...	...	...	...	...

(c)

Figure 2: Current representations of road maps as directed graphs with scalar travel time values. (Left) Example road map. [17] (Middle) Graph Representation. (Right) Tabular representation of digital road maps.

Route determination services [28, 44], abbreviated as routing services, include the following two services: best-route determination and route comparison [40]. The first deals with determination of a best route given a start location, end location, optional waypoints, and a preference function. Here, choice of preference function could be: fastest, shortest, easiest, pedestrian, public transportation, avoid locations/areas, avoid highways, avoid toll ways, avoid U-turns, and avoid ferries. Route finding is often based on classic shortest path algorithms such as Dijktra's [23], A* [8], hierarchical [19, 20, 41, 43], materialization [37, 39, 41], and other algorithms for static graphs [4, 6, 7, 12, 14, 34, 38]. Shortest path finding is often of interest to tourists as well as drivers in unfamiliar areas. In contrast, commuters often know a set of alternative routes between their home and work. They often use an alternate service to compare their favorite routes using real-time traffic information, e.g., scheduled maintenance and current congestion. Both services return route summary information along with auxiliary details such as route maneuver and advisory information, route geometry, route maps, and turn-by-turn instructions in an audio-visual presentation media.

3. EMERGING SPATIAL BIG DATA

SBD are significantly more detailed than traditional digital roadmaps in terms of attributes and time resolution. In this subsection we describe three representative sources of SDB that may be harnessed in next generation routing services.

Spatio-Temporal Engine Measurement Data: Many modern fleet vehicles include rich instrumentation such as GPS receivers, sensors to periodically measure sub-system properties, and auxiliary computing, storage and communication devices to log and transfer accumulated datasets [21, 22, 27, 30, 46, 47]. Engine measurement datasets may be used to study the impacts of the environment (e.g., elevation changes, weather), vehicles (e.g., weight, engine size, energy-source), traffic management systems (e.g., traffic light timing policies), and driver behaviors (e.g., gentle acceleration/braking) on fuel savings and GHG emissions.

These datasets may include a time-series of attributes such as vehicle location, fuel levels, vehicle speed, odometer values, engine speed in revolutions per minute (RPM), engine load, emissions of greenhouse gases (e.g., CO2 and NOX), etc. Fuel efficiency can be estimated from fuel levels and distance traveled as well as engine idling from engine RPM. These attributes may be compared with geographic contexts such as elevation changes and traffic signal patterns to improve understanding of fuel efficiency and GHG emission.

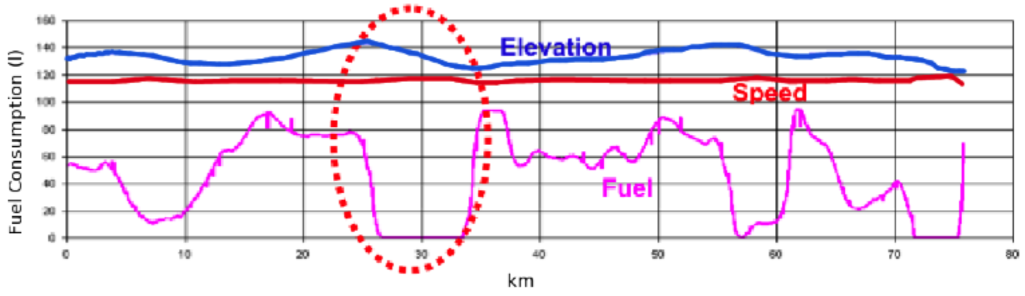
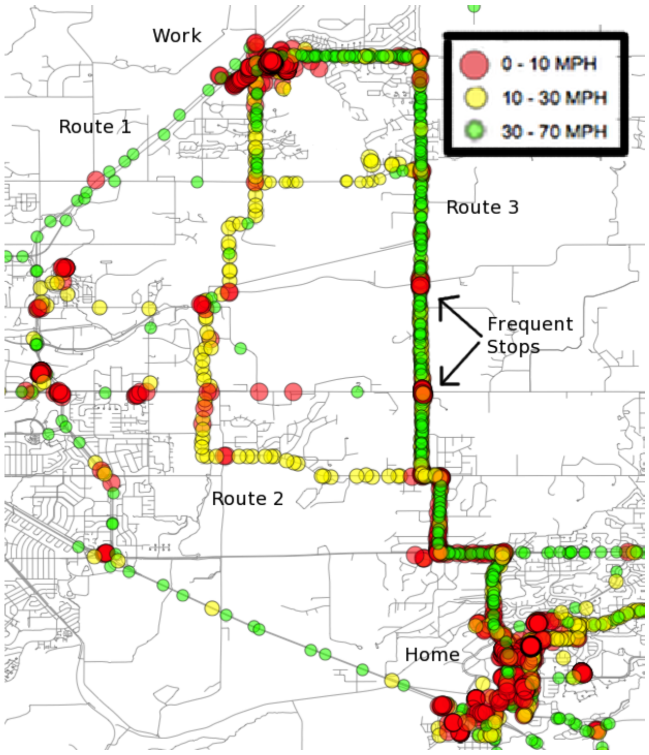


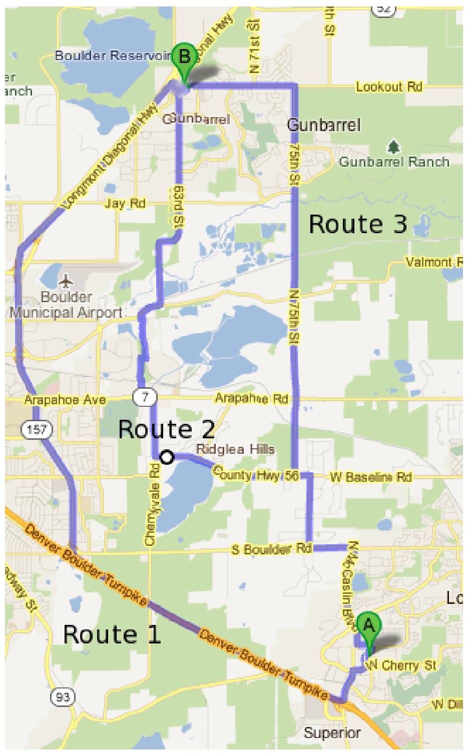
Figure 3: Engine measurement data improves understanding of fuel consumption [5]. (Best in color)

For example, Figure 3 shows heavy truck fuel consumption as a function of elevation from a recent study at Oak Ridge National Laboratory [5]. Notice how fuel consumption changes drastically with elevation slope changes. Fleet owners have studied such datasets to fine-tune routes to reduce unnecessary idling [1, 2]. It is tantalizing to explore the potential of this dataset to help consumers gain similar fuel savings and GHG emission reduction. However, these datasets can grow big. For example, measurements of 10 engine variables, once a minute, over the 100 million US vehicles in existence [11, 45], may have 10^{14} data-items per year.

GPS Trace Data: A different type of data, GPS trajectories, is becoming available for a larger collection of vehicles due to rapid proliferation of cell-phones, in-vehicle navigation devices, and other GPS data logging devices [15, 54] such as those distributed by insurance companies [51]. Such GPS traces allow indirect estimation of fuel efficiency and GHG emissions via estimation of vehicle-speed, idling and congestion. They also make it possible to make personalized route suggestions to users to reduce fuel consumption and GHG emissions. For example, Figure 4 shows 3 months of GPS trace data from a commuter with each point representing a GPS record taken at 1 minute intervals, 24 hours a day, 7 days a week. As can be seen, 3 alternative commute routes are identified between home and work from this dataset. These routes may be compared for idling, which are represented by darker (red) circles. Assuming the availability of a model to estimate fuel consumption from speed profile, one may even rank alternative routes for fuel efficiency. In recent years, consumer GPS products [15, 48] are evaluating the potential of this approach.



(a)



(b)

Figure 4: A commuter's GPS tracks over three months reveals preferred routes. (Best in color)

Historical Speed Profiles: Traditionally, digital road maps consisted of centerlines and topologies of the road networks [16, 42]. These maps were used by navigation devices and web applications such as Google Maps [17] to suggest routes to users. New datasets from companies such as NAVTEQ [31] use probe vehicles and highway sensors (e.g., loop detectors) to compile travel time information across road segments for all times of the day and week at fine temporal resolutions (seconds or minutes). This data is applied to a profile model, and patterns in the road speeds are identified throughout the day. The profiles have data for every five minutes, which can then be applied to the road segment, building up an accurate picture of speeds based on historical data. Such TD roadmaps contain much more speed information than traditional roadmaps. Traditional roadmaps (Figure 2(a)) have only one scalar value of speed for any given road segment (e.g., EID 1). In contrast, TD roadmaps may potentially list speed/travel time for a road segment (e.g., EID 1) for thousands of time points (Figure 5(a)) in a typical week. This allows a commuter to compare alternate start-times in addition to alternative routes. It may even allow comparison of (start-time, route) combinations to select distinct preferred routes and distinct start-times. For example, route ranking may differ across rush hour and non-rush hour and in general across different start times. However, TD roadmaps are big and their size may exceed 10^{13} items per year for the 100 million road-segments in the US when associated with per-minute values for speed or travel-time. Thus, industry is using speed-profiles, a lossy compression based on the idea of a typical day of a week, as illustrated in Figure 5(b), where each (road-segment, day of the week) pair is associated with a time-series of speed values for each hour of the day.

In the near future, values for the travel time of a given edge and start time will be a distribution instead of scalar. For example, analysis of GPS tracks may show that travel-time for a road-segment is not unique, even for a given start-time of a typical week. Instead, it may consist of different values (e.g., 1, 2, 3 units), with associated frequencies (e.g., 10, 30, 20). Emergence of such SBD may allow comparison of routes, start-times and (route, start-time) combinations for statistical distribution criteria such as mean and variance. We also envision richer temporal detail on many preference functions such as fuel cost. Other emerging datasets include those related to pot-holes [35], crime reports [36], and social media reports of events on road networks [50].

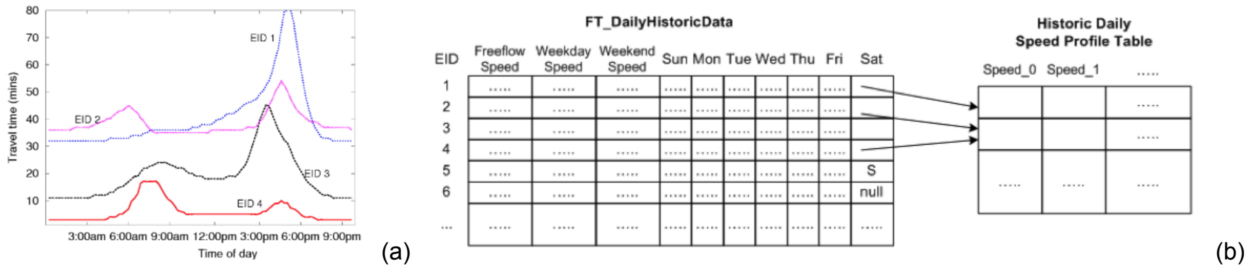


Figure 5: Spatial Big Data on Historical Speed Profiles. (Left) Travel time along four road segments over a day. (Right) Schema for daily historic speed data. (Best in color)

4. NEW CHALLENGES

New generation of mobility services (e.g., Eco-routing) leveraging SBD raise significant new challenges for state of the art spatial computing. First, it requires a change in frame of reference from a snapshot perspective to the perspective of the individual traveling through a transportation network [53]. For instance, consider the new temporally detailed (TD) roadmaps providing historical travel-time (or speed) for each road-segment for every distinct minute of a week. Consider a person sitting in a vehicle and moving along a chosen path in a TD roadmap. She would experience a different road-segment and its historical speed as well as traversal-time at different time-intervals, which may be distinct from the start-time.

Second, the growing diversity of SBD significantly increases computational cost because it magnifies the impact of the partial nature and ambiguity of traditional routing query specification. Typically, a routing query is specified by a starting location and a destination. Traditional routing services would identify a small set of routes based on limited route properties (e.g., travel-distance, travel-time (historical and current)) available in traditional digital roadmap datasets. In contrast, SBD face orders of magnitude richer information, more preference functions (e.g., fuel efficiency, GHG emission, safety, etc.) and correspondingly larger sets of choices. New questions thus arise in context of eco-routing: What is the computational structure of determining routes that minimize fuel consumption and GHG emissions? Does this problem satisfy the assumptions behind traditional shortest-path algorithms (e.g., stationary ranking of alternative routes assumed by a dynamic programming principle)? For example, temporally detailed roadmaps can potentially provide a distinct route for every possible start-time, even when we just consider travel-time. This raises an optimality challenge of correctly determining the fastest route corresponding to each start-time, since ranking of candidate routes might vary with time of day (rush hour vs. non-rush hour). It also raises a representation challenge to summarize potentially large sets of routes in the result. There is also the computational challenge of efficiently determining a large collection of routes (e.g., one for each start time and preference function) by identifying and reducing unnecessary computations perhaps leveraging current cloud computing paradigm (e.g., map reduce) or via novel custom cloud computing paradigms, tentatively called spatial cloud computing.

Third, the tremendous diversity of SBD sources substantially increases the need for diverse solution methods. For example, methods for determining fuel efficient routes that leverage engine measurement and GPS track datasets may be quite different from algorithms to identify minimal travel-time routes for a given start-time exploiting TD roadmaps. In addition, SBD data (e.g., TD roadmaps, GPS-tracks and engine-measurement datasets) differ in coverage, roadmap attributes and statistical details. For example, TD roadmaps cover an entire country, but provide mean travel-time for a road-segment for a given start-time in a week. In contrast, GPS-track and engine-measurements have smaller coverage to well-travelled routes and time-periods, but may provide a richer statistical distribution of travel-time for each road-segment, perhaps revealing newer patterns such as seasonality. New algorithms are likely to emerge as new SBD become available and as a result, a new, extensible, architecture will be needed to rapidly integrate new datasets and associated algorithms.

Fourth, another challenge area is in the use of geospatial reasoning and SBD in sensing and inference across space and time. Multiple tradeoffs (including those arising in privacy considerations) can come to the fore with attempts to sense and draw inferences from stable or mobile sensors. New challenges arise from crowd-sourced sensors. For example, the ubiquity of mobile phones presents an incredible opportunity for gathering information about all aspects of our world and the people living in it [24]. Already research has shown the potential for mobile phones with built-in motion detectors carried by everyday users to detect earthquakes mere seconds after they begin [13]. Navigation companies frequently utilize mobile phone records to estimate traffic levels on busy highways [50]. How can computers efficiently utilize this prevalent sensing power of mobile phones without drastically impacting battery life or personal privacy concerns? This raises many computer science questions related to sensor placement, configuration, etc.

Fifth, privacy of geographic information inside SBDs is an important challenge. While location information (GPS in phones and cars) can provide great value to users and industry, streams of such data also introduce spooky privacy concerns of stalking and geo-slavery [10]. Computer science efforts at obfuscating location information to date have largely yielded negative results. Thus, many individuals hesitate to indulge in mobile commerce due to concern about privacy of their locations, trajectories and other spatio-temporal personal information [25]. Spatio-temporal computing research is needed to address many questions such as the following: "whether people reasonably expect that their movements will be recorded and aggregated..."? [32]. How do we quantify location privacy in relation to its spatio-temporal precision of measurement? How can users easily understand and set privacy constraints on location information? How does quality of location-based service change with variations in obfuscation level?

Sixth, SBD can also be used to make predictions about a broad range of issues including the next location of a car driver, the risk of forthcoming famine or cholera, or the future path of a hurricane. Such predictions would challenge the best of machine learning and reasoning algorithms, including directions with geospatial time series data. Many current techniques assume independence between observations and stationarity of phenomena. Novel techniques accounting for spatial auto-correlation and non-stationarity may enable more accurate predictions. How can new techniques remain computationally efficient while incorporating auto-correlation and non-stationarity while remaining computationally efficient?

5. CONCLUSIONS

Increasingly, mobility datasets are of a size, variety, and update rate that exceed the capability of spatial computing technologies. This paper addresses the emerging challenges posed by such datasets, which we call Spatial Big Data (SBD), specifically as they apply to mobility services (e.g., transporation and routing). SBD examples include trajectories of cell-phones and GPS devices, vehicle engine measurements, temporally detailed (TD) road maps, etc. SBD has the potential to transform society. A recent McKinsey Global Institute report estimates that personal location data could save consumers hundreds of billions of dollars annually by 2020 by helping vehicles avoid congestion via next-generation mobility services such as eco-routing. Eco-routing may leverage various forms of Spatial Big Data to compare routes by fuel consumption or greenhouse gas (GHG) emissions rather than total distance or travel-time.

However, the envisaged SBD-based next-generation mobility services pose several challenges for current routing techniques. First, SBD requires a change in frame of reference, moving from a global snapshot perspective to the perspective of an individual object traveling through a transportation network. Second, SBD magnifies the impact of partial information and ambiguity of traditional routing queries specified by a start location and an end location. For example, traditional routing identifies a unique (or a small set of) route(s), given historical and current travel-times. In contrast, SBD may identify a much larger set of solutions, e.g., one route each for thousands of possible start-times in a week, significantly increasing computational costs. Third, SBD challenges the assumption that a single algorithm utilizing a specific dataset is appropriate for all situations. The tremendous diversity of SBD sources substantially increases the diversity of solution methods. For example, methods for determining fuel efficient routes leveraging engine measurement and GPS track datasets may be quite different from algorithms used to identify minimal travel-time routes exploiting temporally detailed roadmaps. Newer algorithms will be needed as new SBD becomes available, creating demand for a flexible architecture to rapidly integrate new datasets and associated algorithms. Other challenges include geo-sensing, privacy, prediction, etc.

6. ACKNOWLEDGEMENTS

We would like to thank Eric Horvitz (Microsoft), the Computing Community Consortium (CCC), Hillol Kargupta (UMBC), Erik Hoel (ESRI), Oak Ridge National Labs (ORNL) and the US-DoD for their helpful comments and support.

7. REFERENCES

1. American Transportation Research Institute (ATRI). ATRI and FHWA release bottleneck analysis of 100 freight significant highway locations. http://www.atri-online.org/index.php?option=com_content&view=article&id=248&Itemid=75, 2010.
2. American Transportation Research Institute (ATRI). Fpm congestion monitoring at 250 freight significant highway location: Final results of the 2010 performance assessment. http://www.atri-online.org/index.php?option=com_content&view=article&id=303:250-freight-significant-locations, 2010.
3. Austin Brown. Transportation Energy Futures: Addressing Key Gaps and Providing Tools for Decision Makers. Technical report, National Renewable Energy Laboratory, 2011.
4. J. Booth, P. Sistla, O. Wolfson, and I.F. Cruz. A data model for trip planning in multimodal transportation systems. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pages 994–1005. ACM, 2009.
5. G. Capps, O. Franzese, B. Knee, MB Lascurain, and P. Otaduy. Class-8 heavy truck duty cycle project final report. ORNL/TM-2008/122, 2008.
6. E. Chan and J. Zhang. Efficient evaluation of static and dynamic optimal route queries. Advances in Spatial and Temporal Databases, pages 386–391, 2009. Springer. LNCS 5644.
7. T.S. Chang. Best routes selection in international intermodal networks. Computers & operations research, 35(9):2877–2891, 2008. Elsevier.

8. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms. MIT Press, 2001.
9. Davis, S.C. and Diegel, S.W. and Boundy, R.G. Transportation energy data book: Edition 28. Technical report, Oak Ridge National Laboratory, 2010.
10. J. Dobson and P. Fisher. Geoslavery. Technology and Society Magazine, IEEE, 22(1):47–52, 2003.
11. Federal Highway Administration. Highway Statistics. HM-63, HM-64, 2008.
12. D. Frigioni, M. Ioffreda, U. Nanni, and G. Pasqualone. Experimental analysis of dynamic algorithms for the single. Journal of Experimental Algorithmics (JEA), 3:5, 1998. ACM.
13. M. Faulkner, M. Olson, R. Chandy, J. Krause, K. Chandy, and A. Krause. The next big one: Detecting earthquakes and other rare events from community-based sensors. In Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on, pages 13–24. IEEE, 2011.
14. Daniele Frigioni, Alberto Marchetti-Spaccamela, and Umberto Nanni. Semidynamic algorithms for maintaining single-source shortest path trees. Algorithmica, 22(3):250–274, 1998.
15. Garmin. http://www.garmin.com/us/.
16. Betsy George and Shashi Shekhar. Road maps, digital. In Encyclopedia of GIS, pages 967–972. Springer, 2008.
17. Google Maps. http://maps.google.com.
18. Erik G. Hoel, Wee-Liang Heng, and Dale Honeycutt. High performance multimodal networks. In Advances in Spatial and Temporal Databases, pages 308–327, 2005. Springer. LNCS 3633.
19. G.R. Jagadeesh, T. Srikanthan, and K.H. Quek. Heuristic techniques for accelerating hierarchical routing on road networks. IEEE Transactions on Intelligent Transportation Systems, 3(4):301–309, 2002.
20. Ning Jing, Yun-Wu Huang, and Elke A. Rundensteiner. Hierarchical optimization of optimal path finding for transportation applications. In Proceedings of the fifth international conference on Information and knowledge management (CIKM), pages 261–268, 1996. ACM.
21. H. Kargupta, J. Gama, and W. Fan. The next generation of transportation systems, greenhouse emissions, and data mining. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1209–1212. ACM, 2010.
22. H. Kargupta, V. Puttagunta, M. Klein, and K. Sarkar. On-board vehicle data stream monitoring using minefleet and fast resource constrained monitoring of correlation matrices. New Generation Computing, 25(1):5–32, 2006. Springer.
23. Jon Kleinberg and Eva Tardos. Algorithm Design. Pearson Education, 2009.
24. A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In Proceedings of the 7th international conference on Information processing in sensor networks, pages 481–492. IEEE Computer Society, 2008.
25. J. Krumm. A survey of computational location privacy. Personal and Ubiquitous Computing, 13(6):391– 399, 2009.
26. Joel Lovell. Left-hand-turn elimination. http://goo.gl/3bkPb, December 9, 2007. New York Times.
27. Lynx GIS. http://www.lynxgis.com/.
28. M. Mabrouk, T. Bychowski, H. Niedzwiadek, Y. Bishr, JF Gaillet, N. Crisp, W. Wilbrink, M. Horhammer, G. Roy, and S. Margoulis. Opengis location services (openls): Core services. OGC Implementation Specification, 5:016, 2005. Citeseer.
29. J. Manyika et al. Big data: The next frontier for innovation, competition and productivity. McKinsey Global Institute, May, 2011.
30. MasterNaut. Green Solutions. http://www.masternaut.co.uk/carbon-calculator/.
31. [76] NAVTEQ. www.navteq.com.

32. New York Times. Justices Say GPS Tracker Violated Privacy Rights. http://www.nytimes.com/2012/01/24/us/police-use-of-gps-is-ruled-unconstitutional.html, 2011.
33. OpenStreetMap. http://www.openstreetmap.org/.
34. Michalis Potamias, Francesco Bonchi, Carlos Castillo, and Aristides Gionis. Fast shortest path distance estimation in large networks. In Proceedings of the 18th ACM conference on Information and knowledge management, CIKM '09, pages 867–876, 2009.
35. Pothole Info. Citizen pothole reporting via phone apps take off, but can street maintenance departments keep up? http://goo.gl/cGl3B, 2011.
36. SafeRoadMaps. Envisioning Safer Roads. http://saferoadmaps.org/.
37. Hanan Samet, Jagan Sankaranarayanan, and Houman Alborzi. Scalable network distance browsing in spatial databases. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD '08, pages 43–54, 2008.
38. P. Sanders and D. Schultes. Engineering fast route planning algorithms. In Proceedings of the 6th international conference on Experimental algorithms, pages 23–36. Springer-Verlag, 2007.
39. J. Sankaranarayanan and H. Samet. Query processing using distance oracles for spatial networks. Knowledge and Data Engineering, IEEE Transactions on, 22(8):1158–1175, 2010. IEEE.
40. J.H. Schiller and A. Voisard. Location-based services. Morgan Kaufmann, 2004.
41. S. Shekhar, A. Fetterer, and B. Goyal. Materialization trade-offs in hierarchical shortest algorithms. In Advances in Spatial Databases, pages 94–111. Springer, 1997.
42. S. Shekhar and H. Xiong. Encyclopedia of GIS. Springer Publishing Company, Incorporated, 2007.
43. Shashi Shekhar, Ashim Kohli, and Mark Coyle. Path computation algorithms for advanced traveller information system (atis). In Proceedings of the Ninth International Conference on Data Engineering, April 19-23, 1993, Vienna, Austria, pages 31–39. IEEE Computer Society, 1993.
44. Shashi Shekhar, Ranga Raju Vatsavai, Xiaobin Ma, and Jin Soung Yoo. Navigation systems: A spatial database perspective. In Location-Based Services, pages 41–82. Morgan Kaufmann, 2004.
45. Daniel Sperling and D. Gordon. Two billion cars. Oxford University Press, 2009.
46. TeleNav. http://www.telenav.com/.
47. TeloGIS. http://www.telogis.com/.
48. TomTom. TomTom GPS Navigation. http://www.tomtom.com/, 2011.
49. U.S. Energy Information Adminstration. Monthly Energy Review June 2011. http://www.eia.gov/totalenergy/data/monthly/.
50. Waze Mobile. http://www.waze.com/.
51. Wikipedia. Usage-based insurance — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Usage-based_insurance&oldid=464698702, 2011. [Online; accessed 15-December-2011].
52. Shrank, D.L., Lomax, T.J., The 2011 Urban Mobility Report, Texas Transportation Institute, September 2011.
53. Venkata M.V. Gunturi, Ernesto Nunes, KwangSoo Yang and Shashi Shekhar, A Critical-Time-Point Approach to All-start-time Lagrangian Shortest Paths: A Summary of Results. Proc. of 12th International Symposium on Spatial and Temporal Databases (SSTD '11), Minneapolis, MN. 2011.
54. Changqing Zhou, Dan Frankowski, Pamela J. Ludford, Shashi Shekhar, Loren G. Terveen: Discovering personal gazetteers: an interactive clustering approach. 12th ACM International Workshop on Geographic Information Systems, ACM-GIS 2004, November 12-13, 2004, Washington, DC, USA. GIS 2004:266-273.

Towards Generalized Centrality Measures with Applications to Information Networks

Evimaria Terzi & Azer Bestavros & Dora Erdos & Vatche Ishakian
Computer Science Department, Boston University, Massachusetts

The problem of identifying "valuable" or "central" nodes in a given network has long been recognized as important by researchers and practitioners alike. There exists an abundance of measures that associate each node with an individual centrality score; the higher the score of a node the more central its position in the network. Link-analysis algorithms and node-centrality measures try to capture this intuition [1, 5, 6, 8, 10]. While existing, commonly used centrality measures give intuition about the relative value of individual nodes, they are not useful in assessing the *collective value of groups of nodes*, which is not necessarily nor typically reflected by the sum of the values of the nodes in the group. For example, assume that the centrality of a group of nodes is defined as the total number of shortest paths passing through at least one node in the group. In this case, a set of nodes with high group centrality may not necessarily include nodes with high individual centrality scores. A relatively small number of recent studies considered such combinatorial notions of nodes' importance in conjunction with specific problems, including advertisement strategy design [4], virus containment [7], and shortest-path distance approximation [9].

The goal of this paper, is to present the blueprints of a research agenda, which explores expressive notions of *group centrality* and develops algorithmic techniques, which enable the implementation and evaluation at scale of an arsenal of tools for use by researchers and practitioners. Next, we present some indicative applications, which guide and will also benefit from the development of this agenda.

Content de-duplication in information-flow networks: Consider flow networks where data items propagate to the network nodes. Examples of such data items include updates in social networks, news flowing through interconnected RSS feeds and blogs, measurements in sensor networks, route updates in ad-hoc networks. Oftentimes, such propagation lacks coordination: nodes relay information they receive to neighbors, independent of whether or not these neighbors received the same information from other sources. This uncoordinated data dissemination may result in significant, yet unnecessary, communication and processing overheads, ultimately reducing the utility of information networks. To alleviate the negative impacts of this *information multiplicity* phenomenon, we propose that a subset of nodes, that we call *filters*, carry out additional information de-duplication functionality. The strategic placement of filters will determine the extent of information multiplicity and ultimately the level of user satisfaction. In this context, the central nodes correspond to the selected filters. Observe that the placement of the filters does not affect the information that nodes receive; it only reduces the multiplicity of received copies. Our preliminary work [2] indicates that this problem is NP-hard, however efficient approximation algorithms exist. Knowledge of the network structure as well as network-sampling methods can further benefit the performance of these algorithms so that they also handle large datasets.

Information gathering in information-flow networks: Consider the problem of identifying the minimum set of (or best fixed number of) nodes to use for capturing all of (or most of) the information propagating through a flow network from a set of sources to a set of destinations. Since information does not propagate in such networks through shortest (or even single) paths, this problem reduces to the identification of the set of nodes that collectively lie on all (or most) paths in the network. Although existing applications dictate such centrality definitions, the computational complexity of the task of finding the set of such nodes is extremely high – after all, there are exponentially many paths! In our recent work [3], we studied how the computational complexity of this problem is affected by the structure of the underlying flow network graph (e.g., tree, acyclic graph etc). For many of these cases were polynomial (approximation) algorithms exist it is interesting to explore the type and the power of accurate sampling techniques.

Effective advertising strategies in navigational networks: The analysis of navigational patterns – governed by an underlying access network – is instrumental for identifying the set of nodes on which to place an advertisement (ad) for maximal exposure. Studies have shown that the number of times a person is exposed to an ad in a short period of time correlates with response probability; this number is known as the *effective frequency* (`http://en.wikipedia.org/wiki/Effective_frequency`). In this context, an interesting problem is the following: assuming an effective frequency of ℓ, what set of k nodes in an access network should be selected for ad placement? Observe that the effective-frequency parameter requires that the ad messages be placed so that users encounter them in (almost) consecutive pages in a given browsing session. In fact, we can extend such centrality definition even further: instead of requiring ad messages to be placed on a set of neighboring nodes, we can impose the requirement that they are placed on strongly connected subgraphs of the underlying access network. Both of these formulations (as well as others that we cannot fully present due to space limitations) give rise to new algorithmic challenges, and – perhaps more importantly from a broader impact perspective – to new types of advertisement strategies.

All the examples presented above, give rise to new combinatorial notions of centrality. The ability to solve such problems is useful both for researchers and practitioners. We believe that the development of centrality-as-a-service is an interesting and important direction for this line of research. Taking bibliography data as an example, one can develop a tool that finds central authors or central papers within a particular scientific domain.

References

[1] S. Brin, R. Motwani, L. Page, and T. Winograd. What can you do with a web in your pocket? *IEEE Data Eng. Bull.*, 21(2):37–47, 1998.

[2] D. Erdos, V. Ishakian, A. Lapets, E. Terzi, and A. Bestavros. The filter-placement problem and its application to minimizing information multiplicity. In *International Conference on Very Large Databases (VLDB)*, 2012.

[3] V. Ishakian, D. Erdos, E. Terzi, and A. Bestavros. A framework for the evaluation and management of network centrality. In *Siam Data Mining Conference (SDM)*, 2012.

[4] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[5] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *SODA*, pages 668–677, 1998.

[6] R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, 2001.

[7] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007.

[8] M. E. J. Newman. The mathematics of networks. *The New Palgrave Encyclopedia of Economics*, 2008.

[9] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis. Fast shortest path distance estimation in large networks. In *CIKM*, pages 867–876, 2009.

[10] S. Wasserman and K. Faust. *Social Network Analysis.* Cambridge University Press, 1994.

Mining Social Media for Rapid Disaster Response

Ranga Raju Vatsavai, Budhendra Bhaduri
Computational Sciences and Engineering Division
Oak Ridge National Laboratory

Background

Accurate damage assessment due to major natural and anthropogenic disasters is becoming critical due to increasing human and economic losses. This increase in loss of life and severe damages can be attributed to the growing population, and human migration and settlements in disaster prone regions of the world. Rapid damage assessment and dissemination of accurate information is critical for creating an effective emergency response. Remote sensing and geographic information systems (GIS) based techniques and tools are playing an important function in disaster damage assessment and reporting activities. Remote sensing data plays a critical role in disaster mapping of human settlements, which range from delineation of effected population areas to the assessment of structural damages to buildings and critical infrastructures. Previous studies have shown that the remote sensing technology has been most widely utilized in mapping and monitoring of hazards and identification of damages and effects of disasters. Remote sensing is also useful in (near) real time assessment of damages due to floods, forest fires, and other temporal phenomena. However, for supporting a comprehensive decision support system, it may require integration of several technologies includes remote sensing, GIS, modeling and simulation systems, and information extracted from social media.

Social Media and its Role in Disaster Response

In addition to the traditional imagery and vector data, rapid damage assessments can highly benefit from the voluntary geographic information (VGI) and social media. It is clear that social media has become a powerful tool for disaster response efforts. Recent advances in Internet technologies and innovations have given raise to new social networks like twitter, Facebook, YouTube. In addition, advances in computer technology have given raise to new and powerful devises like smart touch phones and tablets equipped with GPS. These devices not only allow faster dissemination of social media data, but also allow creation of data can be used to post immediately to news websites, twitter, Facebook and the like. Therefore it is highly beneficial to use social media data in rapid damage assessment. Especially, such data can greatly benefit ground-truth collection that is required to build accurate supervised machine learning models for change and damage detection to the critical infrastructure and natural resources. However, social media can also be misused which will greatly impact damage assessments and emergency response. New algorithms are needed to integrate social media data into the rapid damage assessment workflows, with an eye for uncertainty and ambiguity. At ORNL we are investigating ways to: (i) efficiently mine social media data sources for extracting facts relevant to disasters, (ii) quantify uncertainty and disambiguate location information extracted from social media, (iii) integrate information mined from the social media with remote sensing and GIS data, and (iv) dissemination of disaster related information for rapid response.

Beyond Simple Parallelism: Challenges for Scalable Complex Analysis over Social Data

Cong Yu
Google Research, New York, NY
congyu@google.com

ABSTRACT

With the increasing popularity of social platforms such as Facebook, twitter and Google+, applications built on those platforms are increasingly relying on the rich social activity data generated by their users to deliver personalized experiences. Those so-called social applications perform various analysis tasks to gather insights from the data and, to handle the data at large scale, adopt parallel programing paradigms for those tasks, MapReduce being the most notable model. In this position paper, we describe the challenges facing sophisticated analysis tasks where simple parallelization no longer works and pose a few questions for future research.

1. INTRODUCTION

Users' social activities on the Web are becoming a rich and critical information source for many applications that are pivoted around providing personalized user experience based on the social data. With the Facebook platform leading the pack with over 800 million users, and other social platforms such as twitter and Google+[1] also reaching hundreds of millions of users, those social applications are empowered with enormous amount of data[1] that allow them to perform various analysis tasks. Those tasks chew over hundreds of terabytes of data on a daily basis and therefore require scalable tools. Over the last few years, MapReduce[4] has emerged as the most popular computing paradigm for parallel, batch-style, analysis of large amount of data. Simplicity is one of the main reasons that MapReduce has become so popular, but it also introduces challenges for tasks that are not embarrassingly parallel[6]. For example, while counting the number of users with different profile features (e.g., age or location) is trivially parallelizable, a more sophisticated analysis of counting distinct number of users over all possible feature combinations can be very challenging. In this paper we discuss the challenges facing the MapReduce model for those non-trivial tasks.

The rest of the paper is organized as follows: Section 2 provides a brief introduction to the MapReduce model. Section 3 presents example tasks that are difficult to do using simple MapReduce programs and highlights the challenges. Section 4 poses a few questions for future research.

2. MAPREDUCE

MapReduce is a shared-nothing parallel data processing paradigm that consists of two main operations that the programmer can customize: *Map* and *Reduce*. Each **map operation** processes one record r from the input and produces zero or more $\langle key, val\rangle$ pairs using the customized map function. Each **reduce operation** processes a single key and all the vals associated with that key to produce zero or more output records using the customized reduce function. The critical aspect of the MapReduce model is that each map operation is independent from each other, and as a result, the entire input can be distributed across many machines (called *mappers*) and be processed in parallel. Similarly, the reduce operations can also be distributed across many machines (called *reducers*). The shuffling of vals into groups of the same key is accomplished by the *Shuffle* phase, which occurs between the *Map* and the *Reduce* phases, and is critical to the overall performance of the MapReduce system.

[1]According its IPO filing on Feb 1, 2012: Facebook sees 2.7 billion likes and comments per day.

Table 1 illustrates a simple example of counting the occurrences of each character within an input set of strings.

Map Input	Map Output	Shuffle Phase	Reduce Input	Reduce Output
hello	$\langle h,1\rangle$, $\langle e,1\rangle$		$\langle h,\{1,1\}\rangle$	$\langle h,2\rangle$
	$\langle l,2\rangle$, $\langle o,1\rangle$	...	$\langle e,\{1,1\}\rangle$	$\langle e,2\rangle$
home	$\langle h,1\rangle$, $\langle o,1\rangle$	...	$\langle l,\{2\}\rangle$	$\langle l,2\rangle$
	$\langle m,1\rangle$, $\langle e,1\rangle$		...	...

Table 1: Counting Characters in MapReduce.

3. CHALLENGES

Not all analysis tasks can be easily accomplished in MapReduce. Mappers and reducers are commodity machines and therefore have limited memories and disk space. Furthermore, the overhead of launching mappers or reducers and the communication cost between jobs can be quite significant. As a result, for any given problem, there are three critical conditions that a solution must meet for a MapReduce program to solve it efficiently. The first condition is **single worker friendliness**, i.e., the workload of a single mapper or reducer needs to be within the capacity of a single machine. While each map operation is usually manageable by a mapper[2], a reduce operation can be troublesome if there are too many values associated with a single key, due to either data skew or inherent characteristics of the problem. In fact, this is often the main issue for many MapReduce programs dealing with real life data. In Section 3.1, we provide such a challenging problem.

The second condition is **limited critical operations**, i.e., the total number of required invocations of any user provided critical operations should be linear to the input size[3]. A critical operation is an expensive piece of user code that must be executed many times to solve the problem. As an example, to compute average user visit duration for each profile groups, *averaging two weighted numbers* is a critical operation in the straight forward solution. Since this operation is invoked once for each input user, this solution satisfies

[2]A single record exceeding the capacity of a machine is rare.
[3]This implies that the output size is also linear to the input size, which is almost always true for practical analysis tasks.

the critical operation condition. In Section 3.2, we provide a concrete problem whose naive solution does not satisfy this condition and is therefore difficult to solve in MapReduce.

The third condition is **bounded iterations**, i.e., the total number of required map-shuffle-reduce iterations should be constant[4]. This issue is of particular importance in graph analysis where many graph problems require iterative solutions that might take a non-trivial number of MapReduce iterations, such as personalized PageRank computation[3]. Many researchers in the theory community have started to look into this—interested readers can start with those studies that aim to model the MapReduce paradigm[5, 7].

3.1 Challenging Aggregations

While most aggregation analyses fit the MapReduce model, many aggregations also turn out to be rather challenging. One prominent example is large scale cube computation with non-algebraic measures (e.g., distinct counts), which was first presented in [8]. Specifically, given a user activity log, a location hierarchy and a topic hierarchy (the latter two being derived from the user and his/her activity in the log), we want to compute `volume` and `reach` of all cube groups whose `reach` is above a certain threshold, and identify those with unusually high `reach` compared with their `volume`. Here, a cube group is defined as any combination of (location, topic) pairs, such as (DC, politics) or (all, all), the latter denoting the group of all users and topics. The measure `volume` is defined as the number of tuples in the group, while `reach` is defined as the number of unique users performing those activities. This analysis is inspired by the need to identify activities that are performed infrequently by the users, but cover a large number of users: these are often missed by traditional frequency based analyses because of their relative low volume, even though they have impact on a disproportionally large user population.

Naive solutions to the problem turn out to be very challenging for MapReduce due to the violation of the first condition: single worker friendliness. Specifically, without careful optimization, a single reduce operation for computing reach can be inundated with a large number of user IDs (the entire input in the worse case) and drag down the whole job.

3.2 Challenging Joins

Traditionally a hard problem in MapReduce, join processing using MapReduce has seen significant progresses made by a few recent studies. In particular, [9] proposed a general framework for processing database-style theta-joins using statistics gathered from a join matrix to guide the distribution of the map and reduce operations, while [10] solved a particularly important problem of Jaccard distance similarity join using a multi-iteration MapReduce pipeline with optimizations highly tuned for the problem.

However, there are many real world complex joining problems where the above techniques don't apply and as such they remain challenging to solve by the MapReduce model. One prominent example is k-nearest-neighbor joining over billions of objects with complex similarity functions, a problem we are currently investigating[2] for social networks. Those similarity functions are either too difficult to reason about because they are user provided C++ functions or simply impossible to know because they are computed using a machine learning model. As a result, naive solutions can not avoid performing $O(N^2)$ number of distance computations, violating the second condition: limited critical operations.

[4]This is a loose condition—the number of iterations can be, for example, logarithmic to the input size as long as the base factor is very large such that, in any reasonable practical settings, it is a small number.

4. QUESTIONS

We believe the above two examples are only the tip of the iceberg of many more problems that are challenging for the MapReduce model. We ask the following questions and hope other researchers can join us in seeking the answers. First, what are the core characteristics of those problems that make them so hard for MapReduce, i.e., is there a "complexity theory" for MapReduce? Second, for problems whose naive solutions violate one or more conditions discussed above, are there principled ways for identifying "good" solutions? If such principles exist, a declarative language on top of MapReduce can potentially be used to produce such MapReduce solutions, an approach us database researchers are quite familiar with. Finally, are there alternative computing models that are more suitable than MapReduce for those large scale analysis tasks?

5. CONCLUSION

In this position paper, we described challenges involved in solving large scale data analysis problems that are not straight-forwardly suited for the MapReduce paradigm. We posed some questions whose answers can get us closer to solving those challenging problems with or without the MapReduce model.

6. ACKNOWLEDGEMENT

We thank Sihem Amer-Yahia (QCRI), Boulos Harb (Google Research), Herald Kllapi (University of Athens), Arnab Nandi (The Ohio State University) for collaborations and discussions.

7. REFERENCES

[1] http://plus.google.com/.
[2] manuscript under preparation.
[3] B. Bahmani, K. Chakrabarti, and D. Xin. Fast personalized pagerank on mapreduce. In *SIGMOD*, 2011.
[4] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, 2004.
[5] J. Feldman, S. Muthukrishnan, A. Sidiropoulos, C. Stein, and Z. Svitkina. On the complexity of processing massive, unordered, distributed data. *CoRR*, abs/cs/0611108, 2006.
[6] I. T. Foster. *Designing and building parallel programs - concepts and tools for parallel software engineering.* Addison-Wesley, 1995.
[7] H. J. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *SODA*, 2010.
[8] A. Nandi, C. Yu, P. Bohannon, and R. Ramakrishnan. Distributed cube materialization on holistic measures. In *ICDE*, 2011.
[9] A. Okcan and M. Riedewald. Processing theta-joins using mapreduce. In *SIGMOD*, 2011.
[10] R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using mapreduce. In *SIGMOD*, 2010.