Partitioning Social Networks for Time-dependent Queries

Berenice Carrasco Yi Lu
Department of ECE
University of Illinois
carrasc4@illinois.edu, yilu4@illinois.edu

Joana M. F. da Trindade
Department of Computer Science
University of Illinois
trindade@illinois.edu

Abstract

The most common type of queries in online social networks are news feeds of friends' recent activities. These queries involve the retrieval of multiple small records generated by different users in the network, and the results are time dependent. Hash-based horizontal partitioning of data results in accesses at multiple servers, which significantly affects throughput and response time. Partitioning of social network data is difficult because of the power-law degree distribution of the friendship graph, and the time dependency of queries and user activities. The power-law degree distribution results in a tremendous amount of extra storage for replication-based partitions, and the time dependency makes query-driven partitioning ineffective. We propose to partition not only the spatial network of social relations, but also in the time dimension so that users who have communicated in a given period are grouped together. We build an activity prediction graph to capture relationships with strong activity and serve as the basis for partitioning. New nodes occurring in the current period are added greedily. We test the partitioning results with emulation of Facebook page downloads, and show that our algorithm achieves 10 times better data locality than hash-based horizontal partitioning algorithms. We show the quality of activity prediction by observing that the algorithm with prediction achieves 80% data locality of that with perfect knowledge of the current period.

1. Introduction

The fast growth of online social networks has prompted interest in building a scalable architecture that caters to social network queries [Curino 2010, de Candia 2007, Lakshman 2010, Pujol 2010], the most common of which are news feeds of friends' recent activities. Scaling is important to both new and existing social network sites: Facebook has reached five hundred million registered users in 2010 [Wikipedia 2011] and Twitter grew by 1382% between Feb and Mar 2009 [Media 2009]. The queries of online social networks differ significantly from traditional web applications: The highly personalized content results in small queries involving multiple records generated by different users in the network [Benevenuto 2009, Schneider 2009]; data are highly interconnected due to the community structure among users [Newman 2006] whose popularity and structure vary over time [Viswanath 2009].

Most online social networking sites use hash-based horizontal partitioning of data for its simplicity, where the base unit of partitioning is a row of a table. Twitter, a microblogging social network with around 175 million users, developed Gizzard [Kallen 2006], which implements range partitioning. Cassandra [Lakshman 2010] is a distributed storage system developed by Facebook, which uses consistent hashing of user IDs for better performance with incremental and dynamic data. Amazon implemented Dynamo [de Candia 2007] for storing and retrieving user shopping carts with a modified consistent hashing that addresses the non-uniformity of load distribution.

Hash-based horizontal partitioning techniques require the access of multiple small records located on different servers. It has been reported that distributed queries reduce performance compared to local queries [Curino 2010, Pujol 2010]. In particular, Curino *et. al.* reported that using local queries doubles the throughput [Curino 2010]. Retrieving small records from distributed servers also increases communication overhead as the payload of packets is small compared to the header.

However, local queries are difficult to achieve in online social networks due to the power-law degree distribution of the friendship graph [Newman 2002] and the time-varying nature of queries and underlying social networks, which make most queries non-repeatable.

1. Power-law graph. It is well-known that balanced partitioning of power-law graphs is a difficult problem [Fjallstrom 1998]. The friendship graph of social networks, where two users are connected if they are friends, has a power-law distribution and a small fraction of nodes has very large degrees. Partitioning the graph into disjoint parts results in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SNS'11 April 10, 2011, Salzburg, Austria.

Copyright © 2011 ACM 978-1-4503-0634-8/11/04. . . $10.00

many queries accessing multiple partitions [Hamilton 2008], and replication approaches require a large amount of extra storage. While the recently proposed replication algorithm, SPAR [Pujol 2010], significantly outperforms other algorithms in terms of replication overhead, the average number of extra replicas still reaches 7 for the Facebook data with 512 servers, and a small portion of user data even need to be replicated on *all* servers.

2. Time-dependent queries and networks. An example of time-dependent queries is a refresh of a user's Facebook home page to retrieve the most recent news feeds of his / her friends' activities: At different times, the set of most recent messages retrieved is different. Such a query will never be repeated exactly due to its time dependency. As a result, approaches that exploit past query patterns do not handle time-dependent queries well. In addition, the underlying social network also varies over time. Friends are added and deleted, and an existing friendship link varies in its interaction frequency, which makes partitioning solely based on friendship graphs insufficient.

1.1 Our Contribution

We propose to partition not only the spatial network consisting of user nodes, but also along the *time* dimension. That is, the partition will not place all messages between user 1 and 2 on the same server, but will divide these messages according to their time stamps and place messages within a particular time range on the same server. While the friendship graph considered by SPAR is power-law with a heavy tail, the graph corresponding to messages within a time range, say a month, has a much lighter tail. The phenomenon is observed in [Viswanath 2009] and the latter graph is referred to as the *activity network*. As a result, partitioning activity networks produces balanced partitions with much fewer cross-edges and reduces the need for replication. It also adapts to the time-varying nature of queries and networks. The idea is illustrated in Figure 1.

Figure 1 shows a small network with 7 user nodes (squares). The messages exchanged between user i and j are indexed by (i, j) and represented by a message node (circles). In period 1, users 1 and 2 exchanged 100 messages while users 2 and 3 did not interact; In period 2, users 1 and 2 did not interact, but users 2 and 3 exchanged 100 messages. When replication is used to ensure one-hop locality on the friendship network such as in [Pujol 2010], all 200 messages are replicated once as illustrated in Figure 1(b). However, when activity networks of different periods are partitioned, no replication is needed, as illustrated in Figure 1(c).

Our algorithm identifies strong links from past traces and build an activity prediction graph (APG) for partitioning messages of the current period. In this paper, we focus on the quality of prediction and data locality for a single period. Our contributions are as follows:

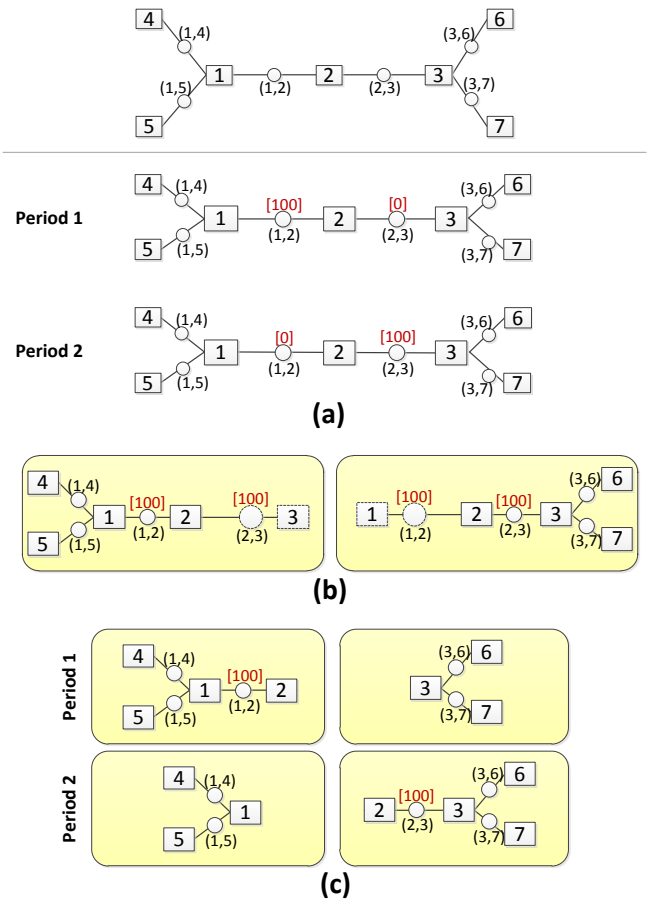

Figure 1. Example of a small social network to be partitioned in two servers. (a) Communication pattern in the network over different periods of time. Number in square bracket is the total number of messages sent between the corresponding users. (b) Partitioning the friendship graph with replications. A total of 200 messages are replicated. (c) Partitioning the activity networks for different periods of time. No replication is needed. .

- We construct an activity prediction graph (APG) that takes into consideration the interactions on real online social networks. That is, a user is able to view all activities initiated and received by a friend, possibly with some user who is not a friend. This necessitates the use of the *two-hop network*.
- We evaluate the partitioning result with queries mimicking real Facebook page downloads: A query requests for a fixed number of most recent messages in the user's two-hop network. Each user has different access frequencies, which we assume to be proportional to his / her activities.
- Our algorithm produces 10 times better data locality than hash-based horizontal partitioning algorithms, as shown in Section 4.2. APG produces high-quality prediction as our algorithm (with prediction) achieves 80% data

locality of the same algorithm with perfect knowledge of the current period.

Unlike previous work where all past queries are treated equally when partitioning the graph, we discount past queries by their time stamps so that activities taking place long time ago have little effect on the partitioning of the current period. Moreover, the edges are weighted in our graph, corresponding to strong and weak links among users. We do not guarantee that all users have their data locally. Instead, we aim to provide *frequent* users with high data locality, which speeds up a large proportion of queries with much less storage than algorithms with stronger locality guarantees. The space-time tradeoff might not be favorable if a replica is created for data accessed with very low frequency. Although experiments in this paper do not involve replicas, replicas can be added appropriately to improve data locality. When used in conjunction with a caching system, the algorithm can also be used to partition data in the cache.

For this paper, we use data from the Facebook New Orleans network between Jan 2005 and Dec 2006 provided by Viswanath *et al*. It contains 8643 users and 69836 wall posts. This is the same set of data used in [Viswanath 2009]. Note that wall posts are the main medium of interaction during this period. The size of the network crawled is much smaller than the size of entire Facebook as the access of activity data has much more stringent requirement than that of friendship data. For instance, one needs to be a member of the network in order to view activities in the network. We did not use the data from Twitter and Orkut, which are also available to the public, as they do not contain time stamp information. We tested the partitioning of a single period with queries mimicking Facebook page download. Placement of data across periods to minimize number of time ranges is beyond the scope of this paper.

The rest of the paper is organized as follows. Section 2 explains the distribution and evolution of activity networks. Section 3 defines the activity prediction graph used for partitioning. Section 4 presents the evaluation results from emulation of Facebook page refreshes.

2. Activity Networks and Link Decay

Activity networks consider actual interaction between users rather than mere friendship information. In a friendship network, links have equal weights. They are added whenever a friendship relation is declared between two users and remain in the network once created. In an activity network, however, links have different weights that change over time, and links can be added and removed based on real interaction frequency in the period. Activity networks were shown to display very different structural properties than the friendship network [Chun 2008, Wilson 2009]. For instance, it was shown in [Viswanath 2009] that the average degree of the activity network constructed from the New Orleans network on Facebook is much lower than that of the corresponding friendship network (only 12.2% of the social links showed any wall post activity), and that the maximal degree in the activity network are around 100. This motivates our algorithm to partition the activity network, as high data locality can potentially be achieved with few replicas.

It was also shown in [Viswanath 2009] that the links in the activity network tend to come and go over time, and the strength of ties exhibits a decreasing trend as the link ages. In order to see how past queries should affect the partitions for the current period, we plot the proportion of links in Dec 2006 that are present in a past period in Fig. 2.

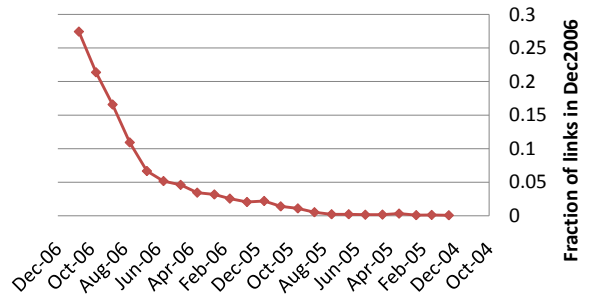

Figure 2. Fraction of links in Dec 2006 that are present in the past period.

Observe that the temporal correlation decreases significantly with the distance from the current period. This motivates us to weight an interaction by a decay factor so that interactions taking place in the remote past have less effect on the partitioning for the current period.

3. Graph Representation

We define the activity prediction graph (APG) and computation of its weights. Let G be a graph on vertex sets U and V where U is the set of users in the social network and V is the set of interactions, or messages, between a pair of users. A vertex in V always has degree 2 and connects to the two user vertices that are involved in the interaction. Fig. 3 shows an example of a graph with 6 user vertices and 5 interaction vertices.

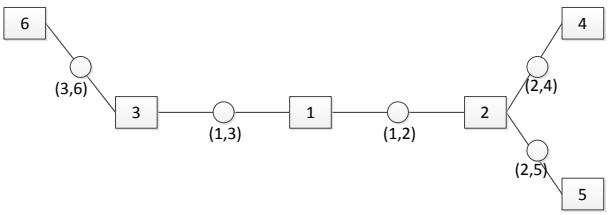

Figure 3. Graph with 6 user vertices and 5 interaction vertices.

In order to quantify the impact of partitioning at a particular edge, we need the following concepts.

3.1 Two-Hop Networks

A two-hop network of user i includes vertices whose content is visible to user i. A user can view the interactions between him and his friends, and all interactions initiated or received by all his friends. We define a hop to be the traversal from one user vertex to a message vertex and to another user vertex in G. For instance, user 1 is one hop from user 2, while user 3 is two hops from user 2. As each user vertex always connects to a message vertex and message vertices always have degree 2, each hop contains a message vertex.

The messages initiated or received directly by user i are on the one-hop network centered at user i, while the messages initiated or received by all friends of user i reside on the two-hop network centered at user i. For instance, the set of user vertices $\{2, 3, 4, 5, 6\}$ and the edges connecting them constitutes the two-hop network centered at user 1.

We also define the two-hop network centered at a message vertex to be the union of the one-hop networks of its initiators and receivers. For instance, the set of user vertices $\{1, 2, 3, 4, 5\}$ and the edges connecting them constitutes the two-hop network centered at message vertex $(1, 2)$. It is important that we consider two-hop networks as each message can be accessed by any user on its two-hop networks.

3.2 Vertex Weights and Frequent Users

Define the weight of each message vertex to be the discounted message frequency,

$$w_{i,j} = C \sum_{k=1}^{K} f_k \, n_{i,j}^k,$$

where K is the total number of past periods considered, C is a scaling constant, $n_{i,j}^k$ is the number of messages exchanged between users i and j in month k and f_k is the decay factor computed on a monthly basis for month k defined as follows. Let L_k be the set of links in month k and L be the set of links in the current month. Let $|\cdot|$ denote cardinality of a set, then

$$f_k = \frac{|L_k \cap L|}{|L_k|}.$$

The weight of each message vertex is used for computing balanced partitions as it is a prediction of the number of messages at each link based on past queries. Balancing the weight of message vertices produces partitions, each of which has an equal share of data storage.

All user vertices are assigned weight 0 in this paper. We associate a frequency m_i to each user i. Let

$$m_i = \sum_{(i,j) \in G} w_{i,j},$$

that is, the sum of all weights on message vertices connected to user vertex i. This corresponds to the predicted value of amount of interaction user i is capable of in the current period. The frequency m_i will be used for computing the edge weights. We do not consider the balance of accesses across partitions in this paper, but it can be readily integrated by assigning weights to user vertices proportional to its frequency.

3.3 Edge Weights

We define edge weights to quantify the effect of "cutting" an edge, which corresponds to storing a message with either the initiator or the receiver, but not both. Let the edge weight between user vertex i and edge vertex (i, j) be $e_{i,j}^i$ and that between user vertex j and edge vertex (i, j) be $e_{i,j}^j$. Let $G_{i,j}^2$ be the two-hop network centered at message vertex (i, j) and G_i^2 be the two-hop network centered at user vertex i. We need the following quantities:

1. Sum of all message weights in the two-hop network, D_i. For each user vertex, we define

$$D_i = \sum_{(i,j) \in G_i^2} w_{i,j}.$$

This is the total weight of messages accessible to user i. For user 1 in Fig. 4,

$$\begin{aligned} D_1 &= w_{3,6} + w_{1,3} + w_{1,2} + w_{2,4} + w_{2,5} \\ &= 22 + 46 + 71 + 5 + 10 = 154. \end{aligned}$$

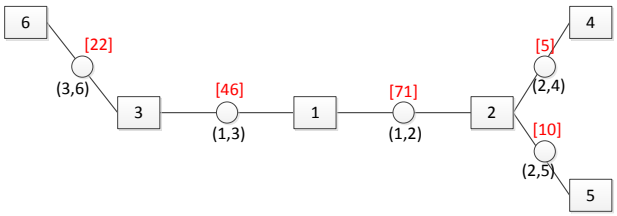

Figure 4. Graph with 6 user vertices and 5 message vertices, with weights on each message vertex.

2. Remote message weights for user k given the edge between user vertex i and edge vertex (i, j) is cut, $W_{(i,j),i}^k$. When an edge belonging to the two-hop network is cut, the user will need to access some of the messages from a remote partition. Let $W_{(i,j),i}^k$ be the total weight of all messages in a remote partition from user k when the edge between user vertex i and edge vertex (i, j) is cut. In Fig. 4,

$$\begin{aligned} W_{(1,2),1}^1 &= w_{1,2} + w_{2,4} + w_{2,5} \\ &= 71 + 5 + 10 = 86 \end{aligned}$$

and

$$W_{(1,2),1}^3 = w_{1,2} = 71.$$

We are ready to define the edge weights. Let

$$e_{i,j}^i = \sum_{k \in G_{i,j}^2} m_k \frac{W_{(i,j),i}^k}{D_k},$$

which is the sum of fraction of remote message weights weighted by user frequency.

4. Evaluation

We fix the test period to be Dec 2006. There are a total of 13948 messages in this period. We use the activity prediction network to partition the storage of messages in the test period. The details are described in Section 4.1.

To test the quality of prediction and the resulting partitioning, we emulate the user activity on social networks. For each message posted in December 2006, we trigger one query each for the initiator and receiver of the message, based on the assumption that activity of a user is proportional to the number of messages he generates or receives. Each query involves the downloading of the most recent 6 messages in the two-hop network of the user. We present the result on data locality for each query in Section 4.2.

4.1 Graph Partitioning

We use $C = 12$ and $K = 23$ to construct the APG. We experimented with other values of C and K and the result is not sensitive to the change in the values. We use KMETIS, a software program from the METIS library [Karypis 2009], to partition the APG. KMETIS uses a multilevel k-way min-cut algorithm to produce partitions that balance vertex weights in each partition and minimize edge weights across partitions.

For a message whose corresponding node is present in the APG, it is stored in the given partition. For a message not predicted by the APG, we use the following simple algorithm:

1. If both the initiator and receiver of the message exist in the APG, but no previous message exists, store the message with the user with a smaller value of D, as the new message will contribute a larger fraction of this user's future query.
2. If exactly one of the initiator and receiver of the message exists in the APG, say user node i, store the new message in the same partition as node i.
3. If neither the initiator nor the receiver exists in the APG, store the new message to the partition with the least number of messages.

The values D_i are updated for each user i as new messages are stored in each partition.

4.2 Evaluation of Data Locality

We take the set of messages posted in Dec 2006, and trigger one query each for the initiator and receiver of the message. The query involves the retrieval of the most recent 6 messages in the two-hop network of the user. We choose the number of messages to be 6 as our data set is relatively small: the data in Dec 2006 contains a total of 13948 messages.

We compare our algorithm to two hash-based horizontal partitioning algorithms. These are the algorithms used in commercial online social networks. The first algorithm, hash_p1, hashes the initiator ID of a message. As a result, all messages generated by the same user are grouped in one partition. The second algorithm, hash_p1p2, hashes the unordered initiator-receiver pair of a message. All messages exchanged between a particular pair of users are grouped in one partition. We compare the experiments with different number of partitions up to 20. We did not experiment with a larger number of partitions as there are only 8640 active users for Dec 2006, and we are considering the locality of messages in a two-hop network.

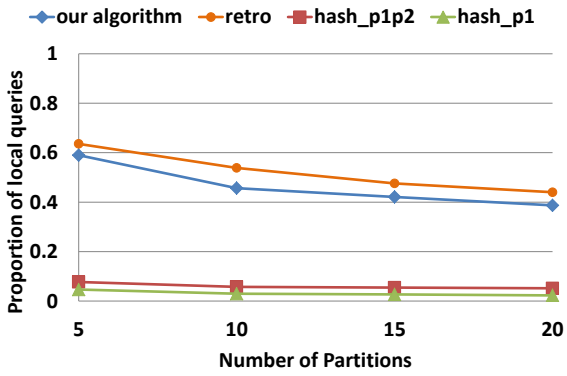

Figure 5. Proportion of queries that access only 1 partition. Comparison of our algorithm with hashing the initiator ID (hash_p1) and the unordered initiator-receiver pair (hash_p1p2).

Fig. 5 compares the proportion of queries that have all 6 most recent messages in a single partition for the three algorithms. With 5 partitions, our algorithm produces 60% of all queries with all 6 messages in one partition whereas both hashing algorithms have only 5% of all queries in one partition. With 20 partitions, our algorithm achieves 40% of local queries as some two-hop networks need to be cut to keep the balance of the data storage, which is still 14 times better than the hashing algorithms, each achieving 2.8% and 2.6% of local queries.

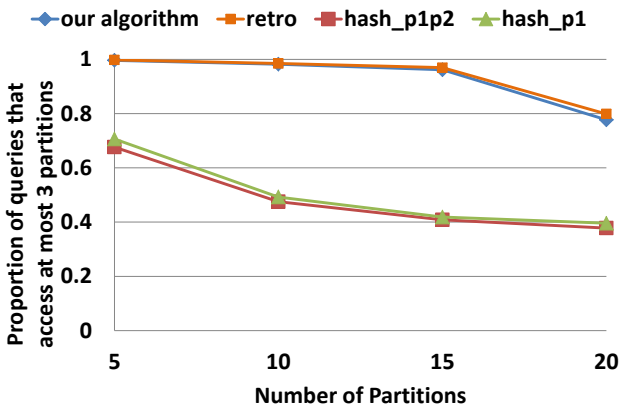

Figure 6. Proportion of queries that access at most 3 partition. Comparison of our algorithm with hashing the initiator ID (hash_p1) and the unordered initiator-receiver pair (hash_p1p2).

Fig. 6 compares the proportion of queries that have all 6 most recent messages in at most 3 partition. A small amount of replication can be used to keep these queries local. For all numbers of partitions, more than 80% of queries access at most 3 partitions with our algorithm. The proportion is very close to 1 up to 15 partitions. For the hashing algorithm, while 71% of all queries access at most 3 partitions when there are a total of 5 partitions, the fraction decreases to 40% when there are a total of 20 partitions.

5. Related Work

There has been significant interest in efficient data back-ends of online social networks recently. We compare and contrast our approach with related work in the area.

Distributed Hashing: Most online social networks rely on distributed hashing [de Candia 2007, Lakshman 2010] to partition data in a scalable way, but it can lead to poor performance due to lack of data locality, and create issues like the 'multi-get' hole for Facebook. Our algorithm is shown to produce better data locality than distributed hashing in this paper.

Partitioning and Replication: Pujol *et. al.* proposed SPAR [Pujol 2010] to partition and replicate the graph to achieve one-hop locality. While outperforming existing algorithms, the average number of extra replicas is large, requiring several times more storage than simple hashing. This is due to the power-law nature of social networks and is difficult to overcome. The situation will be worse if two-hop networks are considered.

Workload-driven Approach: Curino *et. al.* proposed Schism that partitions data based on query patterns. It works well where queries are static and repeated many times. However, it will not be able to predict future queries in social networks, when both data and the network are changing over time.

6. Conclusion and Future Work

We proposed an algorithm to partition social network data based on activity levels of relationships at different times. We built an activity prediction graph for partitioning messages of the current period and produced significantly improved data locality than consistent hashing. Future work includes placement of data across different periods so that an active user has most of his data in one partition.

Acknowledgement

We would like to thank Alan Mislove of Rice University for providing the data on Facebook New Orleans network, and the anonymous reviewers for very helpful comments.

References

[Benevenuto 2009] Fabrício Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virgílio Almeida. Characterizing user behavior in online social networks. In *IMC '09*, pages 49–62, New York, NY, USA, 2009. ACM.

[Chun 2008] H. Chun, H. Kwak, Y-H. Eom, Y-Y. Ahn, S. Moon, and H. Jeong. Online social networks: Sheer volume vs social interaction. *Proc. of IMC*, 2008.

[Curino 2010] C. Curino, E. Jones, Y. Zhang, and S. Madden. Schism: a workload-driven approach to database replication and partitioning. *Proc. of the VLDB Endowment*, 3, 2010.

[de Candia 2007] G. de Candia, D. Hastorun, M. Jampani, G. Kakulapati, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. *Proc. of 21st ACM SIGOPS*, pages 205–220, 2007.

[Fjallstrom 1998] P.-O. Fjallstrom. Algorithms for graph partitioning: A survey. *Linkoping Electronic Articles in Computer Information Science*, 1998.

[Hamilton 2008] J. Hamilton. Scaling LinkedIn. Website, 2008. URL `http://perspectives.mvdirona.com/2008/06/08/ScalingLinkedIn.aspx`.

[Kallen 2006] N. Kallen, R. Pointer, E. Ceaser, and J. Kalucki. Introducing gizzard, a framework for creating distributed datastores. *Twitter Engineering Website*, 40, April 2006. URL `http://engineering.twitter.com/2010/04/introducing-gizzard-framework-for.html`.

[Karypis 2009] G. Karypis and V. Kumar. Metis: Unstructured graph partitioning and sparse matrix ordering system. 2009. URL `http://www.cs.umn.edu/ metis`.

[Lakshman 2010] A. Lakshman and P. Malik. Cassandra: A decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44, April 2010.

[Media 2009] N. Media. Growth of twitter. Website, 2009. URL `http://blog.nielsen.com/nielsenwire/online mobile/twitters-tweet-smell-of-success/`.

[Newman 2002] M. Newman, D. Watts, and S. Strogatz. Random graph models of social networks. In *Proc. of the National Academy of Sciences*, volume 99 (Suppl 1), 2002.

[Newman 2006] M. E. J. Newman. Modularity and community structure in networks. volume 103, pages 8577–8582, June 2006.

[Pujol 2010] J.M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The little engine(s) that could: scaling online social networks. *Proc. of the ACM SIGCOMM 2010 Conference*, 40, October 2010.

[Schneider 2009] Fabian Schneider, Anja Feldmann, Balachander Krishnamurthy, and Walter Willinger. Understanding online social network usage from a network perspective. In *IMC '09*, pages 35–48, New York, NY, USA, 2009. ACM.

[Viswanath 2009] B. Viswanath, A. Mislove, M. Cha, and K.P. Gummadi. On the evolution of user interaction in facebook. *Proc. of the 2nd ACM SIGCOMM WOSN*, 3, 2009.

[Wikipedia 2011] Wikipedia. List of social networking websites. Website, 2011. URL `http://en.wikipedia.org/wiki/List_of_social_networking_websites`.

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