Extensions of BG for Testing and Benchmarking Alternative Implementations of Feed Following*

Sumita Barahmand, Shahram Ghandeharizadeh and Dobromir Montauk

Database Laboratory Technical Report 2014-01

Computer Science Department, USC

Los Angeles, California 90089-0781

{barahman, shahram}@usc.edu, dobromirv@google.com

May 19, 2014

Abstract

Social networking sites are cloud service providers for person-to-person communication. Feed following is a key service where a member follows activity streams of other members and entities such as fan pages. A follower receives a personalized fusion of streams produced by those followed. There are alternative definitions of the quality of feed displayed to a follower that might be impacted by the highly variable fan-out of the follows graphs. This makes both an implementation of feed following and a validation of its correctness as big data challenges.

This paper presents extensions of an interactive social networking benchmark named BG with feed following actions. An implementation of these actions is agnostic to a data store and can be customized for different data models and designs. We illustrate this extensibility feature of BG by evaluating two alternative implementations of feed following using an industrial strength relational database management system and a document store.

A Introduction

Social networking sites such as Google+, Facebook and Twitter enable their members to produce events and display the events generated by other members and entities, typically their friends or those that they follow. For example, the Home Stream of Google+ [9] displays posts that have been shared with a member[1]. The displayed content might be shared specifically with the member or the circle the member belongs to, produced by a celebrity or an entity that the member is following, or shared publicly. Feed following is

*This research was supported in part by a 2013 Google Ph.D. Fellowship.

[1]These include posts shared with a member and those the member is interested in.

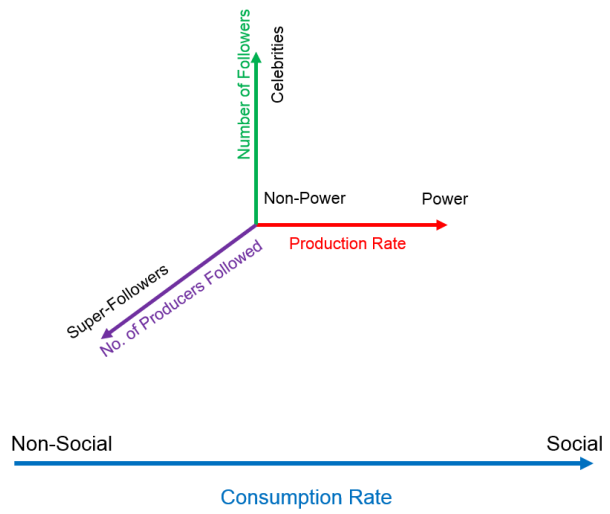


Figure 1: Dimensions used for characterizing member behavior in a social networking system.

an interactive action with users expecting the response time of both producing and consuming events to be short. An implementation must also support a large number of concurrent users producing and consuming events. The quality of the feed displayed to a member is important and must consider what events to show, in what order and when. This makes conceptualizing and modeling feed following for an application challenging. Many of today's social networking sites customize the events displayed to ensure they are relevant, interesting, and timely to a member. Both the highly variable fan-out of the follows graph along with its dynamically changing structure (e.g., a member stops following a celebrity) make an implementation of feed following challenging [13]. One may introduce different designs and implementations to address these challenges [14, 12, 1]. In addition, once an implementation is available, validating the correctness of the implementation and its execution introduces additional challenges [6]. A benchmark that quantifies the performance tradeoffs of these alternatives and evaluates their correctness would enable an experimentalist to obtain insights into the behavior of these designs to introduce improved algorithms.

The primary contribution of this paper is an extension [8] of the BG benchmark [3] with two new actions that quantify the performance of an implementation for feed following. This includes changes at both a conceptual and a physical representation of benchmark data while preserving the semantics of BG's existing eleven actions. At a conceptual level, we made two changes. First, we extended BG's data model with additional entities and relationships to support two feed following actions, see Section B. Second, we identified the different dimensions that impact the performance of feed following, see discussions of Figure 1 below. At a physical level, we made three changes. First, we modified BG's generator module [2] to generate data and populate the new conceptual schema with different social graphs. Second, we extended BG's validation module [6] to verify the correctness of different designs and implementations for the feed following actions.

Term	Definition
Consumer	Members who retrieve their news feed.
Producer	Members or Pages who share resources with their followers. These resources will be posted on their followers' news feed.
Social	Members with a high consumption rate who access their news feed frequently.
Non-Social	Members with a low consumption rate who do not access their news feed frequently.
Power	Members and Pages with a high production rate who share resources with their followers frequently.
Non-Power	Members and Pages with a low production rate who do not share resources with their followers frequently.
Super-Follower	Members following more than 1,000 producers (members or pages).
Celebrity	Pages with more than 1,000,000 followers.
Active	Members with both a high production and high consumption rate.
Super-Doc	A Celebrity followed by followers whom are following many other Celebrities.

Table 1: Member behavior in social networks.

Finally, we incorporated the metrics that impact the different dimensions of feed following and modeled them as parameters controlled by a configuration file.

The behavior of members (see Table 1) is characterized along four dimensions [2], see Figure 1:

- Production rate: is the number of events produced and resources shared by a member in t units of time. A member who produces events at a high rate is termed as a Power member and one who does not produce events frequently is termed as a Non-Power member.

- Consumption rate: is the number of times a member accesses her news feed in t units of time. A member who accesses her feed infrequently is referred to as a Non-Social member and one who retrieves her feed very frequently is referred to as a Social member.

- Number of producers followed (fan-out): is the total number of members and pages (see Table 3) a member is following. This includes the number of friends for a member in a symmetric relationship. Members following a large number of other members/pages ($> 1,000$) are referred to as Super-Followers.

- Number of followers (fan-in): is the total number of members following a member or a page. This includes the number of friends for a member in a symmetric relationship. Members/pages followed by a large number of members ($> 1,000,000$) are considered as Celebrities.

Using BG, an experimentalist may control the size of a social graph and its following-follower fan-out, the frequency of change in the structure of the social graph, rate at which events are produced and consumed, and the amount of load. BG quantifies the observed response time and throughput along with a measure of how well the displayed feed matched the requirements of an application. To the best of our knowledge, this is the first paper to introduce a formal benchmark for evaluating feed following.

Database parameters	
M	Number of members in the database.
P	Number of pages in the database.
ϕ	Number of friends per member.
ρ	Number of resources per member.
ι	Number of followers per page.
ϱ	Number of pages followed by each member.

Table 2: BG's database parameters and their definitions.

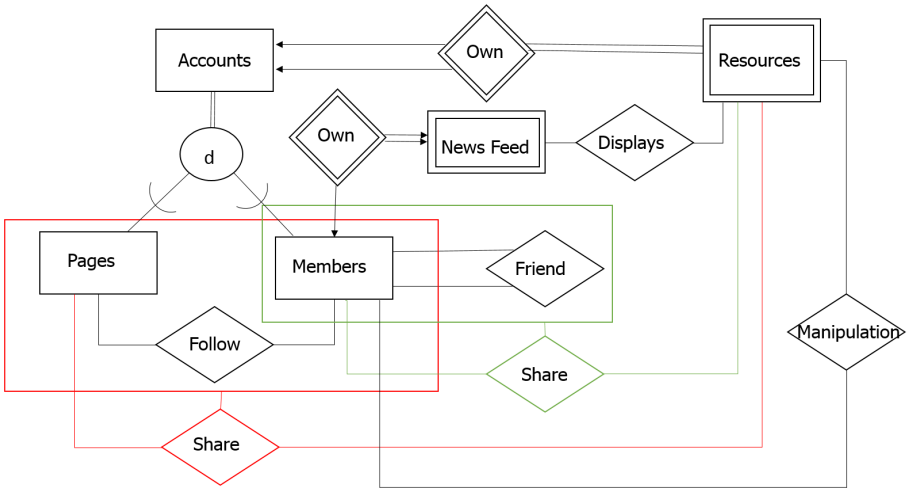


Figure 2: Conceptual data model of BG's database including feed following actions.

The rest of this paper is organized as follows. In Section B, we introduce BG's feed following actions and present the conceptual and physical design of BG's social graph. In Section C we describe how BG validates the correctness of the feed following actions. Section D describes an implementation of these actions based on a pull and push [14] paradigm using an industrial strength relational database management system named[2] SQL-X and a document store named MongoDB, reporting on their behavior with different mixes of actions and follower fan-out as a function of the imposed load. We provide brief future research directions in Section F.

B Conceptual Data Model

This section describes BG's data model in support of two new actions: View News Feed (VNF) and Share Resource (SR) actions. We start with a conceptual design of data and its reduction to two logical data models: relational and JSON.

Figure 2 shows the conceptual data model of BG's database using the Entity-Relationship (ER) data

[2]Due to licensing restrictions, we cannot disclose the identity of this product and its vendor.

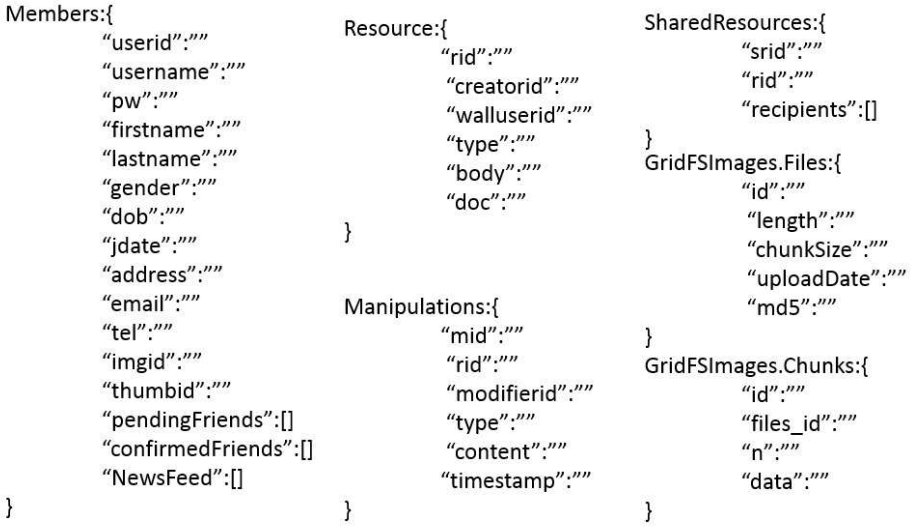


Figure 3: JSON-Like data model of BG's database.

model. The Account entity set is a generalization of Members and Pages entity sets. Pages are special topics such as business, celebrities, brands and etc. that share resources with their followers who are Members. (See Table 3 for a list of terms and their definitions.) The Member entity set consists of accounts registered with a social network that belong to individual people. Its attributes include a unique identifier and a number of string attributes such as firstname, lastname and others. The number of these attributes and their lengths are configuration parameters and can be adjusted to generate different database sizes. Each member profile is configured with a 12 KB profile image and a 2 KB thumbnail image. Typically, thumbnails are displayed when listing friends of a member and the profile image is displayed when visiting a member's profile. Similar to Pages, Members can also share resources with their followers which are other Members who are their friends. These two sharing relationships are captured using the two "Share" relationship sets shown in Figure 2. A member may either extend an invitation to or be friends with another member. Figure 2 captures this using the "Friend" relationship set and uses an attribute to differentiate between invitation and friendships[3]. An Account may "own" resources such as images, a posted question, a technical manuscript, etc. These entities are grouped in one set named "Resources". The existence of a resource depends on it being "owned" by an account. Hence, Resources is a weak entity set and the participation of a resource in the "own" relationship is mandatory. An account, either a Member or a Page, may post a resource, say an image, on the profile of another account, represented as a ternary relationship between two accounts and a resource. In this relationship, the two accounts might be the same account where the account is posting the resource on her/its own profile. A member may comment on a resource. This is implemented using the "Manipulation" relationship set. A member may restrict the ability to comment on a resource only to her

[3]An alternative captures them in two different relationships "Invite" and "Friend", respectively

Members(userid,username,pw,firstname,lastname,gender,dob,jdate,ldate,address,tel,email,profileimage,thumbnail)
Friend(userid1,userid2,status)
Resources(rid,creatorid,wallUserid,type,body,doc)
Manipulation(mid,modifierid,rid,resourceCreatorid,timestamp,type,content)
SharedResources(srid,rid,creatorid)
SharedResourceRecipients(srid,userid)
NewsFeed(userid,srid,rid,creatorid)

Figure 4: Relational data model of BG's database.

friends.

A member owns a News Feed entity. The News Feed entity for a member displays the top k events shared by Pages followed by the member or shared by her friends. This is captured using the "Displays" relationship set.

Figures 3 and 4 show the logical design of the ER diagram with both MongoDB's JSON-like and relational data models. An experimentalist builds a database by specifying the number of members (M) in the social network, number of friends per member (ϕ), number of pages (P), the number of followers for each page (ι), number of pages followed by each member (ϱ) , and resources per account (ρ), see Table 2. We have simplified our social graph by assuming ϕ, ρ and ϱ are the same for all members; and assuming all fan pages have an equal number of followers and resources. Using these assumptions, BG constructs the social graph only if the following conditions hold true:

1. $\varrho \times \iota \leq M$
2. $(P \times \iota)/\varrho \leq M$
3. $P \leq M$
4. $\varrho \leq P$
5. $\iota \leq M$

BG first inserts the members, their friendship relationships and their resources into the data store. Next, it inserts the pages, creates the following relationships and inserts the page resources. In addition to a uniform distribution, some of the relationships might be generated using a skewed distribution. For example, one may use a Zipfian distribution with either (a) exponent 0.99 to model a uniform distribution that assigns an equal number of friendships to each member , or (b) exponent smaller than 0.99 to model a skewed distribution with a few members having a significantly higher number of frineds than others. For example, with 0.27 as the exponent of the Zipfian distribution, 20% of members hold more than 60% of all friendships in the social graph [16, 4].

Term	Definition
Account	Any registered profile with a social network which can be either a member or a page.
Action	A logical social operation implemented by a web page and invoked by a mouse click.
Inter-arrival time	Idle time between two socialite sessions emulated by one thread.
Member	It is an account registered with a social network system that belongs to an individual person. Members can perform all BG's 13 social actions. They can be friends with other members and follow pages.
Page	It is an account registered with a social network that is used to connect people (members) to a topic, which can be a company, celebrity, brand, etc. Pages can be followed only by members, do not follow anyone and do not have any friends.
Resource	An entity that a socialite may browse and post a comment on, e.g., an image.
Session	A sequence of actions by a socialite.
SR	Share Resource action emulated by BG
Socialite	A member/page engaged in a social session.
Think time	Idle time between actions in a session.
VNF	View News Feed action emulated by BG

Table 3: Social networking terms and their definitions.

In the following we describe two feed following actions[4] supported by BG. Section D presents two implementations of these actions and shows how BG can be used to reason about their tradeoffs.

Share Resource, SR: Each member (page) in BG's social graph can share a resource she (it) owns either publicly or with a list of specific members. If shared publicly, then the resource will be available for all the members who are friends with the owner of the resource (are following the page). If shared specifically with a list of members, then the resource is made available to those members only.

When an experimentalist creates a BG database with pages, BG requires each member to follow a fixed number of pages (ϱ) and each page to consist of a fixed number of resources (ρ). These resources are created on the page's own wall and can be shared publicly with all the followers of the page or specifically with a list of them. Share Resource (SR) action enables a socialite to share resources she owns with all her followers or a subset of them. In Section D, we analyze two implementations of this action using MongoDB. With the first implementation, Pull, every time a resource is shared, a new record for the shared item is created [15]. This record maintains the resource information as well as some meta information such as a list of followers allowed to see the shared resource (recipients in Figure 3). The second implementation, Push, extends Pull as follows. It maintains a News Feed entity for each follower and inserts the shared resource in this entity [15]. Each is similar to a materialized view [10, 11] that is uptodated incrementally by either appending entries in response to either an SR or Accept Friend Request action, or deleting entries in response to a Thaw Friendship action.

The SR action requires the memberid of either the member or the page who emulates the action, the resourceid for the resource owned by the member which is going to be shared with followers (the resource

[4]For a list of all actions emulated by BG and their implementation see [3].

is owned by the socialite performing the action) and a list of followers allowed to see the shared item. If this list is set to -1 then the resource is shared with all the followers of the resource owner. This is the only action that can be emulated by pages. BG can also be configured to issue $r\%$ of SR actions by pages and $(1 - r\%)$ of them by members. The value of r is configurable and can be provided to BG as an input.

Both the number of followers per page (ι) and the number of friends per member (ϕ) may impact the throughput observed with different data stores using workloads consisting of SR actions. For example with the Push approach, every time a celebrity (modeled as a page with more than 1,000,000 followers) shares a resource, the News Feed entities for all its followers need to be updated, see Section D for more details.

View News Feed, VNF: Each member of BG owns a News Feed entity. The VNF action enables a socialite to retrieve her news feed and display the top k resources shared with the member. These resources may be shared publicly or privately with the member by other members/pages she is following. The definition of top k is configurable. This action requires the memberid of the Member emulating the action, and the value for k and returns a list of k resources satisfying the order required by the application, as the events in the member's news feed.

An implementation of this technique may use the design of the Share Resource action as follows. With the first one, Pull, upon a VNF action, the pages followed by the member and her friends are queried. Next, all the resources shared by these members/pages which are either shared publicly or specifically with the member are retrieved. Finally the top k criteria is applied to limit the number of shared resources to k and the final k resources are returned as the events displayed on the member's News Feed. With the second implementation, Push, the events shared with a member (publicly or privately) are maintained in a structure (News Feed entity) and updated upon an SR action. So a VNF action only retrieves this News Feed structure for each member emulating the action.

The performance of VNF is impacted by the number of friends per member (ϕ) and number of pages followed by each member (ϱ). With a larger value of ϕ and ϱ for a member, VNF will require retrieving a larger amount of data to compute the top k resources to display for the member's feed and that results in slower service times.

C Validation

As discussed in Section B the View News Feed (VNF) action of BG, retrieves the top k most recent resources shared with a member by her followers. These resources are shared using the Share Resource (SR) action supported in BG. A resource can either be shared publicly with all the members following the resource creator or can be shared specifically with a list of members following the resource creator. Conceptually BG is aware of the initial state of feed for every member, so it changes the value of each feed upon any related SR action by adding the shared resource to the member's feed. A related SR action for member i's feed is

Operation id	Type	Data item	Start	End	Value
DWL1-1	Write	Member1	0	3	1
DWL1-2	Write	Resource1	0	3	-1
DWL2-1	Write	Member1	0	5	2
DWL2-2	Write	Resource2	0	5	2,3
Read1	Read	Member2	4	6	1,2

Table 4: Example log records for BG.

one that is issued by Account j followed by member i. Account j can either be a Member or a Page. For each VNF action, BG enumerates all the relevant SR actions and computes a list of acceptable feed values (list of acceptable resources). If the data store returns a value other than what is expected then unpredictable data is observed and the returned results are not correct.

During the benchmarking phase BG generates two write log records for each SR action and one read log record for each VNF action. The first write log record for an SR action contains the start and end time stamp of the action, the memberid/pageid of the member/page issuing the action and the resourceid of the resource being shared. The second log record also contains the start and end time stamp for the action, the resourceid of the resource being shared and the list of followers the resource is shared with. If the list of followers is set to -1 the resource is shared publicly with all members following this member/page. The read log record contains start and end time stamp for the VNF action, the memberid for the member retrieving her feed and the list of resourceids that are displayed on her feed. BG is also aware of the initial set of accounts followed by a member. In addition, it logs the changes made to the list of members followed by a member. During validation upon a read log record for a VNF action, it computes the list of accounts followed by the member performing the VNF action at the time of read using the start and end time stamp logged for it. It then finds all the resources shared by these accounts till that point of time. It only retrieves the list of resources that are either shared publicly with all followers or are specifically shared with this member. It then compares this list with the list of resources retrieved from the data store for the member's feed. As some SR actions may overlap with the VNF action, BG comes up with a set of acceptable lists containing resourceids. If the feed retrieved form the data store is not a subset of this computed set, then the data store has produced unpredictable data. This is best illustrated with an example. Consider the five log record in Table 4. Assume Member2 is only following Member1 and Member1 owns Resources 1 and 2.

DWL1-1 and DWL-2 belong to the same SR action. DWL1-1 indicates that Member1 shares Resource1 and DWL1-2 indicates that Resource1 is shared publicly (value = -1). Similarly DWL2-1 and DWL2-2 are created by the same SR action. DWL2-1 indicates that Member1 shares Resource2 and DWL2-2 indicates that Resource2 is shared with Member2 and Member3. Read1 indicates that Member2 retrieves her feed and observes Resource1 and Resource2 in it. BG's validation for Read1, finds all the members

followed by Member2 (in this example only Member1). Then retrieves all the SR actions initiated by the members Member2 is following and completed before the read or overlap with it (in this example DWL1-1 and DWL1-2 completed before the read and DWL2-1 and DWL2-2 overlap with it). Then among those it retrieves the ones that either share a resource publicly (DWL1) or share it specifically with Member2 (DWL2) and uses this to compute a set of acceptable resourceid lists for Member2's feed, in this example 1 and 1,2. If Read1 had a value equal to 1,3 then unpredictable data was observed.

D Implementation

This section presents two alternative implementations for VNF and SR actions of BG. We demonstrate that the structure of the social graph, member activity level (exponent θ in D-Zipfian) and the load imposed on the data store impact the performance results. One may use the results provided in this section to develop a new algorithm which targets each of these dimensions for an improved performance.

D.1 Pull

A trivial implementation of displaying a feed is to re-compute it every time a member invokes a VNF action. when a producer generates an event using the SR action of BG, its attributes such as the time of creation and list of recipients (with public sharing, this is -1, and for private sharing the list contains the memberids of those members the resource is shared with) are stored into the data store.

When a member retrieves her feed using BG's VNF action, first a list of all producers followed by her is retrieved, next the top k resources shared by them (either publicly or specifically with this member) are computed for display (the definition of top k may be the k recent events, or the k most relevant events among the others.).

With the pull approach, modifications to friendship relationships incur no additional overhead. For example, assume Member A stops following Member B. The next time Member A retrieves her news feed, a list of producers followed by her is retrieved. This list will no longer contain Member B. Next all resources shared by these producers, either publicly or specifically with Member A, are queried. As Member B is no longer in the list of producers followed by Member A, the resources shared by Member B will not be displayed in Member A's feed.

In addition, with social networking applications that allow modification to generated events such as shared resources, once a producer updates an event produced previously (e.g. edit her status message in Facebook), the next time her followers access their feed, the updated version of the event will be retrieved and displayed on their feed.

With a Pull approach, for a workload consisting of only feed following actions (SR and VNF actions), a higher percentage of SR action, reduces the performance of the data store. This is because higher percentage

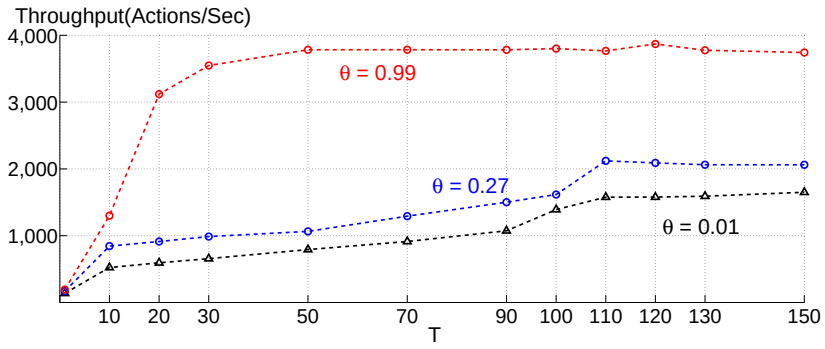


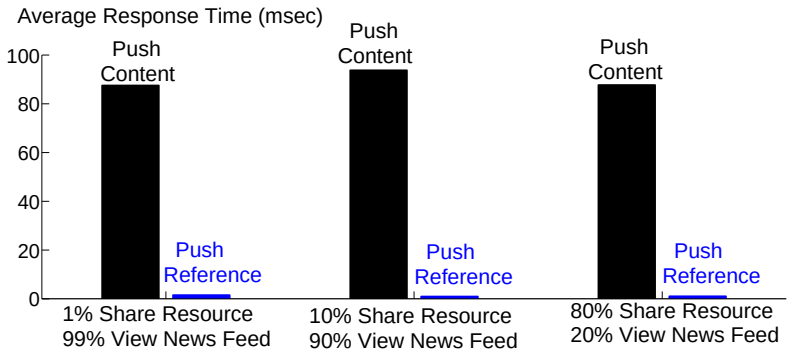
Figure 5: Impact of member activity distribution (θ in D-Zipfian) on MongoDB's performance using a Pull architecture. $M = 10,000$, $P = 100$, $\iota = 1,000$, $\varrho = 10$, $\phi = 100$, $\rho = 10$. The workload consists of 1% SR and 99% VNF actions. Pages issue 1% of the SR actions in all workloads.

of SR actions (writes) results in a larger database as the workload executes. This may result in a higher response time for VNF actions, as now the related queries are issued on a larger data set size, see Figure 5. In addition, the number of members/pages followed by a member impacts the performance of the VNF action with the Pull paradigm. For members with a large fan-out (following a large number of members/pages), the VNF action will observe a higher service time as it will retrieve a larger number of shared resources.

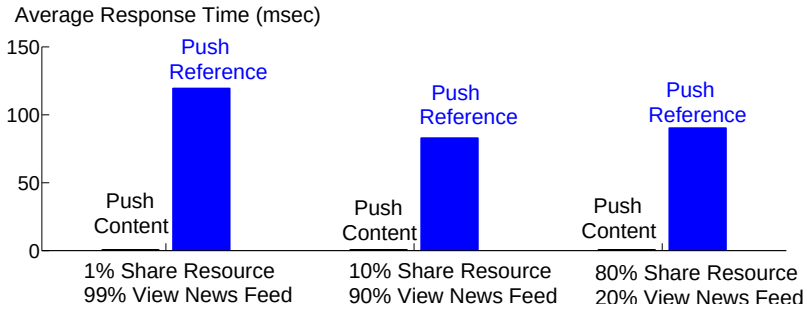
Figure 5, shows that the member activity distribution (identified by exponent θ of the D-Zipfian distribution) also has an impact on the performance of Pull. With a more skewed distribution, the feed for the follower's of Power producers (see Table 1) will consist of a larger number of shared resources. As the VNF action retrieves the top k most recent shared resources for a member, sorting a larger number of events for these members becomes more expensive and reduces the performance of the system.

D.2 Push

With the Push approach, the feed for every member is pre-computed and stored in the data store similar to a materialized view [10, 11]. It is maintained up to date in the presence of SR actions. Every time a producer generates an event by invoking BG's SR action, the list of members following that producer is queried, and the shared resource is pushed (added) to these members' feed. Now the feed for a member is always constructed, up to date and available for her upon request without any additional queries. The disadvantage of this approach is obvious when the relationships between consumers and producers change. Now if a consumer decides to stop following a producer, all the events generated by that producer need to be removed from the consumer's feed. On the other hand if the consumer decides to follow a new producer, the events generated by that producer need to be retrieved and merged with the events already present in the member's news feed. For example, if the friendship between Member A and Member B is thawed, all resources shared by B should be removed from A's feed and vice versa. And if Member B and Member C



6.a : Average Response Time for BG's SR Action



6.b : Average Response Time for BG's VNF Action

Figure 6: Average response time of MongoDB for BG's feed following actions with two Push alternatives: Push Content and Push Reference. $T = 1$, $M = 10,000$, $P = 100$, $\phi = 100$, $\rho = 10$,$\iota = 1,000$, $\varrho = 10$, $\theta = 0.99$.

become friends, all resources shared by B should be added to C's feed and vice versa.

The Push approach is more effective for Social members (see Table 1) who constantly retrieve their feed by invoking VNF actions. For these members, using a Pull approach which recomputes the member's feed upon every request is not ideal. However, for Non-Social members who do not follow many producers, reconstructing the feed upon every SR action may be wasteful work and reduce the performance of the system.

Figures 3 and 4 show the data model used for a Push architecture with MongoDB and SQL-X. There are multiple ways of implementing this data model. For example, one may push either the resources shared by the producer along with all its attributes into each consumer's feed (*Push Content*) or only the references for these resources to each consumer's feed (*Push Reference*). Figure 6 shows the average response time for generating events (BG's SR action) with the Push Content is higher than that with Push Reference with MongoDB. On the other hand, the roles are reversed when considering the VNF action. This is because with Push Reference the VNF action must first retrieve a list of references to all shared resources and then retrieve the resources themselves for display. One may use BG to reason about alternatives and their tradeoffs.

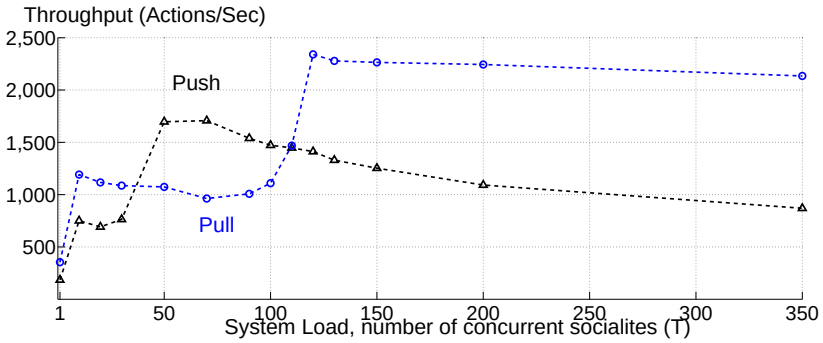


Figure 7: Performance of Pull vs. Push with MongoDB for a High (11% Write) Mixed workload. $M = 10,000$, $P = 100$, $\iota = 1,000$, $\varrho = 10$, $\phi = 100$, $\rho = 10$ and $\theta = 0.27$. Pages issue 1% of the SR actions in all workloads.

E Evaluation

One may use BG to rate a data store by specifying SLA. This computes a single value, Social Action Rating (SoAR) to compare different data stores with one another, see [7, 5]. Alternatively, one may use BG to quantify the throughput (actions/second) observed with different strategies and data stores as a function of the imposed system load, T. We employ this approach to gain insight into the performance of Push and Pull strategies. All experimental results are obtained using one BGClient and an instance of a data store deployed on two different PCs connected using a Gigabit switch and networking cards.

In the first experiment we studied the performance of Pull versus Push with MongoDB for a workload consisting of both feed following actions and other BG actions such as actions that modify friendship relationships: Invite Friend (IF), Accept Friend Request (AFR), Reject Friend Request (RFR), Thaw Friendship (TF). This workload consists of 10% friendship modification actions, 1% SR and 89% VNF action.

For this workload with $\theta = 0.27$ and a low load ($T < 50$) Pull performs better than Push, see Figure 7. This is because Push needs to construct the feed for every member upon every update which introduces an additional overhead. With medium load ($50 < T < 110$) Push performs better than Pull. This is because Push constructs the feed for every member and as the percentage of VNF action is higher than the SR action, Push results in a better response time for retrieving feed for consumer's following Active producers, producers with a high production and consumption rate (see Table 1), and a better overall performance compared to Pull. With a high load ($T > 110$), once again Pull performs better than Push as with an increased number of updates the overhead of constructing member feed increases (as Push retrieves the entire feed and then computes the top k).

In our next set of experiments we set the number of friends per member to 10, 100 and 1,000 and evaluated the performance of Pull and Push architectures for a fixed workload with MongoDB. For all values of ϕ, Pull performs better than Push as it eliminates the overhead of pre-computing member's feed.

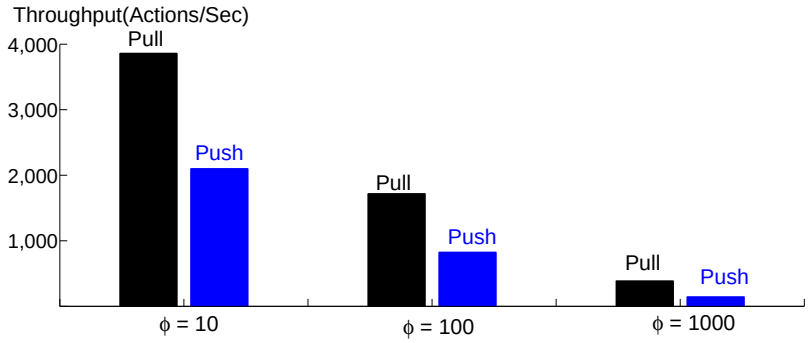
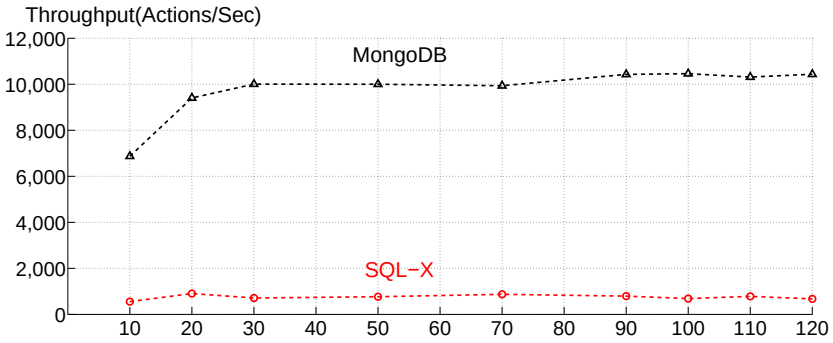
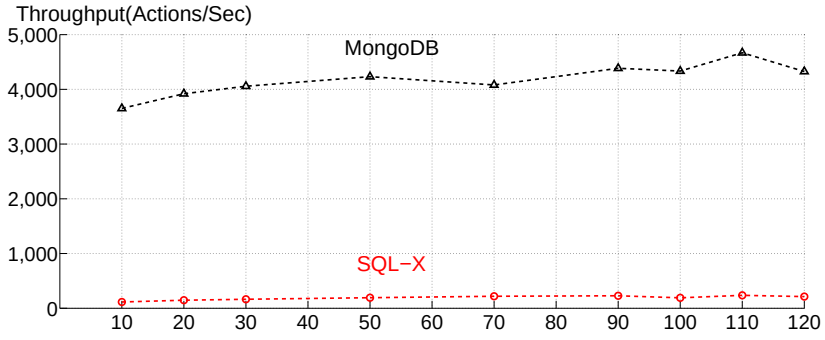


Figure 8: Impact of modifying the number of friends per member (ϕ) on the performance of Pull and Push with MongoDB for a workload consisting of 1% SR and 99% VNF action. $M = 10,000$, $P = 100$, $\iota = 1,000$, $\varrho = 10$, $\rho = 10$ and $\theta = 0.27$. Pages issue 1% of the SR actions in all workloads.



9.a : 1% Share Resource, 99% View News Feed



9.b : 10% Share Resource, 90% View News Feed

Figure 9: Performance of MongoDB and SQL-X with Pull for two workloads. $M = 100,000$, $P = 100$, $\phi = 100$, $\rho = 10$, $\iota = 10,000$, $\varrho = 10$ and $\theta = 0.99$. Pages issue 1% of the SR actions in all workloads.

But as shown in Figure 8 the difference between the alternatives decreases as the value of ϕ increases. This is because with a larger value for ϕ, a member follows a larger number of producers and the resources shared by them need to be retrieved and sorted to compute the member's feed which reduces Pull's performance.

Finally in the last set of experiments we studied the behavior of MongoDB and SQL-X, a relational data store for three different workloads with the Pull (Figure 9 shows the result for two of them. However, the results observed for all three workloads show the same trends.) architecture. As shown in this figure, the performance of MongoDB is superior to that of SQL-X. With SQL-X, the disk of the node hosting the data store becomes the bottleneck, with MongoDB the CPU of the node hosting the data store becomes the bottleneck[5].

F Future Research

BG includes a validator that computes the quality of results produced by the VNF action. Its current design requires the set of feeds displayed to a member to correspond to the set of events produced by those she follows. This validator is ideal for ensuring the correctness of an implementation of the SR and VNF actions. A future research direction is to enable an experimentalist to specify arbitrary validators that rate the quality of feeds displayed to a member in response to the VNF action. For example, in Facebook, news feed displays the latest headlines generated by a member's top friend and pages followed by her. One may provide different definitions of the term *latest*. A challenge is to develop a validator that would quantify how accurately the VNF action with a specific definition is implemented [6].

References

[1] X. Bai, F. P. Junqueira, and A. Silberstein. Cache Refreshing for Online Social News Feeds. In *CIKM*, 2013.

[2] S. Barahmand. Benchmarking Interactive Social Networking Actions, Ph.D. thesis, Computer Science Department, USC, 2014.

[3] S. Barahmand and S. Ghandeharizadeh. BG: A Benchmark to Evaluate Interactive Social Networking Actions. *Proceedings of 2013 CIDR*, January 2013.

[4] S. Barahmand and S. Ghandeharizadeh. D-zipfian: A Decentralized Implementation of Zipfian. In *Proceedings of the Sixth International Workshop on Testing Database Systems*, DBTest '13, 2013.

[5] S. Barahmand and S. Ghandeharizadeh. Expedited Rating of Data Stores Using Agile Data Loading Techniques. In *Proceedings of the 22Nd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '13, pages 1637–1642, 2013.

[5] This is because MongoDB utilizes the entire 16 GB of memory available to the PC hosting the data store while SQL-X was configured with a 4 GB buffer pool.

[6] S. Barahmand and S. Ghandeharizadeh. Benchmarking Correctness of Operations in Big Data Applications. In *IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2014.

[7] S. Barahmand, S. Ghandeharizadeh, and J. Yap. A Comparison of Two Physical Data Designs for Interactive Social Networking Actions. In *Proceedings of the 22Nd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '13, 2013.

[8] S. Ghandeharizadeh and S. Barahmand. A Mid-Flight Synopsis of the BG Social Networking Benchmark. In *4th Workshop on Big Data Benchmarking (WBDB)*, 2013.

[9] Google+. Content That Appears on Your Home Page, http://support.google.com/plus/answer/1269165?hl=en.

[10] A. Gupta and I. S. Mumick. Maintenance of Materialized Views: Problems, Techniques, and Applications. *IEEE Data Eng. Bull.*, 18(2):3–18, 1995.

[11] A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal*, 10.

[12] F. P. Junqueira, V. Leroy, M. Serafini, and A. Silberstein. Shepherding Social Feed Generation with Sheep. In *SNS*, 2012.

[13] A. Silberstein, A. Machanavajjhala, and R. Ramakrishnan. Feed Following: The Big Data Challenge in Social Applications. In *DBSocial*, pages 1–6, 2011.

[14] A. Silberstein, J. Terrace, B. F. Cooper, and R. Ramakrishnan. Feeding Frenzy: Selectively Materializing Users' Event Feeds. In *SIGMOD Conference*, 2010.

[15] A. Silberstein, J. Terrace, B. F. Cooper, and R. Ramakrishnan. Feeding frenzy: Selectively materializing users' event feeds. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 831–842, New York, NY, USA, 2010. ACM.

[16] J. Yap, S. Ghandeharizadeh, and S. Barahmand. An Analysis of BG's Implementation of the Zipfian Distribution, USC DBLAB Technical Report 2013-02, http://http://dblab.usc.edu/Users/papers/zipf.pdf, 2013.