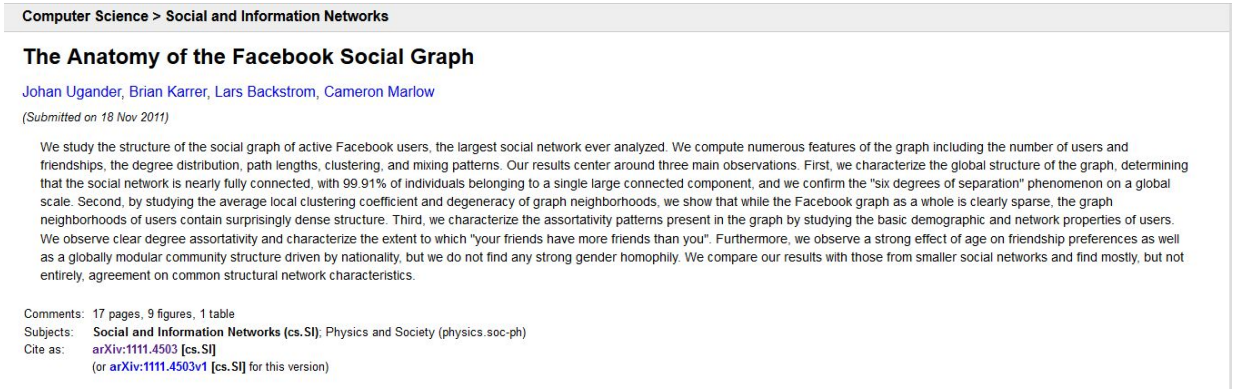
social hash

an assignment framework for optimizing distributed systems operations in social networks

the problem

- All user visible data on Facebook is maintained in a single directed graph called *the Social Graph*
- Contains millions of vertices and trillions of edges
- Consumes hundreds of petabytes of storage space
- Largest social network analyzed so far:



Computer Science > Social and Information Networks

The Anatomy of the Facebook Social Graph

Johan Ugander, Brian Karrer, Lars Backstrom, Cameron Marlow

(Submitted on 18 Nov 2011)

We study the structure of the social graph of active Facebook users, the largest social network ever analyzed. We compute numerous features of the graph including the number of users and friendships, the degree distribution, path lengths, clustering, and mixing patterns. Our results center around three main observations. First, we characterize the global structure of the graph, determining that the social network is nearly fully connected, with 99.91% of individuals belonging to a single large connected component, and we confirm the "six degrees of separation" phenomenon on a global scale. Second, by studying the average local clustering coefficient and degeneracy of graph neighborhoods, we show that while the Facebook graph as a whole is clearly sparse, the graph neighborhoods of users contain surprisingly dense structure. Third, we characterize the assortativity patterns present in the graph by studying the basic demographic and network properties of users. We observe clear degree assortativity and characterize the extent to which "your friends have more friends than you". Furthermore, we observe a strong effect of age on friendship preferences as well as a globally modular community structure driven by nationality, but we do not find any strong gender homophily. We compare our results with those from smaller social networks and find mostly, but not entirely, agreement on common structural network characteristics.

Comments: 17 pages, 9 figures, 1 table
Subjects: **Social and Information Networks (cs.SI)**; Physics and Society (physics.soc-ph)
Cite as: arXiv:1111.4503 **[cs.SI]**
(or **arXiv:1111.4503v1 [cs.SI]** for this version)

the problem

- Information presented to users is a result of dynamically generated queries over this graph
- The Social Graph must generate answers to over a billion queries a second!
- Distributed system design for implementing the systems supporting the social graph can have immense impact
- Designing and implementing such a system is a non-trivial task
- One of the biggest problems is: *how do we assign objects to components in a distributed system in an efficient, scalable, and robust manner?*
- E.g, assigning user request to computer servers or data records to storage subsystems.

the problem

- Assignment constraints:
  - Minimal average query response time
  - Load balance components
  - Assignment stability
  - Fast lookup
- There is combinatorial explosion in the relationship between social networks and query requests, so finding a good assignment is hard
  - NP-hard for many objectives!
- Target optimization goal might violate the other conditions above, so there is a tradeoff.

challenges

- The main challenges are:
  - Scale
  - Effects of similarity on load balance
  - Heterogeneous and dynamic set of components
  - Dynamic workload
  - Dynamic graph (addition and removal of objects)
- The magnitude and relevance of these might vary depending on the distributed system
- Nonetheless, all of these pose serious challenges
- Coming up with a good solution can clearly have significant impact

introducing: social hash

- Social Hash is their proposed solution in this paper
- It is a framework for producing, serving, and maintaining assignment of objects to components in order to optimize operations on large social networks
- It allows us to trade-off between these conflicting objectives above
- They also show that it gives a practical solution to the challenges listed above (at least in the Facebook Social Graph context)
- They show that it has notable impact on the performance and resource utilization over the previous strategies implemented by Facebook

previous work

- Their work depends heavily on graph partitioning and hypergraph partitioning
- The objective in their paper is edge locality and fan-out, which correspond to edge-cut and hyperedge-cut in social network terminology (both well studied)
- For graph partitioning, recall survey and work discussed in my previous presentation
- Many graph partitioning systems available online, including Metis
- There is a Giraph-based approach called Spinner, which is close to this paper
    - Spinner application was optimizing batch processing systems (such as Giraph itself) via increased edge locality
    - This paper's graph partitioning system is embedded in the Social Hash framework

previous work

- There is also literature on hypergraph partitioning, which is a generalization of the graph partitioning problems, and thus harder
- There is a parallel solution for hypergraph partitioning called PHG
- The previous paper by Ugander et al. discusses partitioning large graphs to optimize for Facebook infrastructure
- Stein et al. have considered a theoretical application of partitioning for Facebook infrastructure

previous work

- Other research has considered data replication in combination with partitioning for sharding data for online networks
  - Pujol et al. look at low fan-out configurations via replication of data between hosts
  - Wang et al. look minimizing fan-out by random replication and query optimization

the present work

- This paper is different from some of the previous papers since it presents a realized framework integrated into production system at Facebook
- Their approach, at least in how it relates to Facebook's operating environment and workload, is unique to this paper
- Some technical novelty
    - Two stage-approach, to be discussed
    - First to use edge-cut based graph partitioning techniques for making routing decisions to reduce cache miss rates
    - Focus on bipartite graph partitioning based on prior access patterns in unique way

main idea

- Assignment framework must address both optimization and adaptation objectives
- Assignment of objects to components is done in two steps, and allows for the joint objective
- *Static Assignment Step:* each object is assigned to a group
  - A group is a conceptual entity representing a cluster of objects
  - There are many more groups than components
  - Assignment is based on optimizing a given objective function
    - E.g, when assigning HTTP requests to computer clusters, we might want to minimize chache miss rates
  - We want to reassign objects to groups only periodically and offline

main idea

- *Dynamic Assignment:* each group is assigned to a component
  - Based on input from system monitors and system administrators so as to rapidly and dynamically respond to changes in the system and workload
  - Able to accommodate components going on or offline to keep component loads well-balanced
- **Key idea:** decouple optimization in the static step and separately do dynamic adaptation
- Their procedure is able to shift emphasis between the optimization and adaptation objectives at will by using the parameter n = |groups|/|components|
- Can set n on a per-application basis

social hash architecture

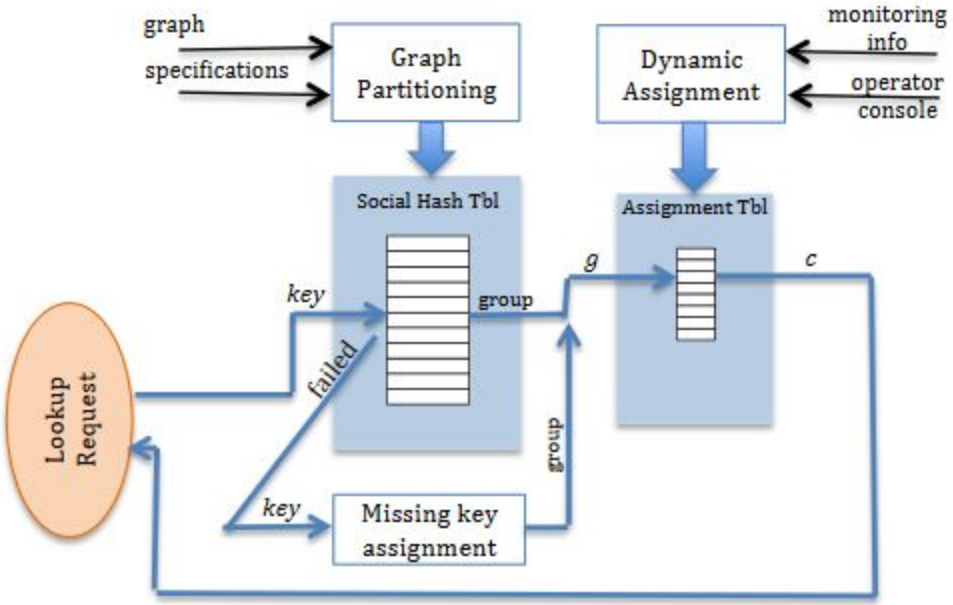


Figure 2: Social Hash Architecture

social hash architecture

- Static assignment algorithm generates a static mapping from objects to groups
  - outputs (key, group) pairs called Social Hash Table
- Dynamic assignment shifts groups among components to balance the load
  - Outputs (group, component) pairs called the Assignment Table
- When a client looks up which component assignment of object, they go through both stages
- If there is a missing key (say new user), then the Missing Key Assignment rule assigns the object to a group on the fly
- These new keys are eventually incorporated into the Social Hash Table by the static partitioning algorithm

static assignment algorithm

- Uses graph partitioning in order to group the objects into groups
- Heuristic
  - Begin with a balanced assignment of objects to groups (say random)
  - For each object v, record the group that gives the optimal assignment for v to minimize the objective function, fixing all other assignments
  - Repeat this for each object (in parallel)
  - Swap as many reassignments as possible under size constraint (in parallel)
  - Repeat until convergence or you reach the number of iterations
- This procedure manages to produce high quality results for the graphs underlying Facebook operations in a fast and scalable manner

dynamic assignment

- Primary goal here is to keep component load well balanced despite changes in access patterns and infrastructure
- The specific load balancing strategy used for Social Hash framework may vary on a per-application basis, due to factors including:
  - Accuracy in predicting future loads
  - Dimensionality of loads
  - Group transfer overhead
  - Assignment memory

social hash for facebook's web traffic routing

- Objective: to improve the efficiency of large cache services
- Demonstrate effectiveness of Social Hash with two applications
    - Assign HTTP requests to individual computer clusters with the goal of minimizing the memory based cache miss rate
    - Assign data record to storage subsystems with the goal of minimizing the number of storage subsystems that need to be accessed on a multi-get fetch requests
- Both have been in production at Facebook for over a year, with over 78% of Facebook's stateless web traffic routing occurring with this framework!

some results

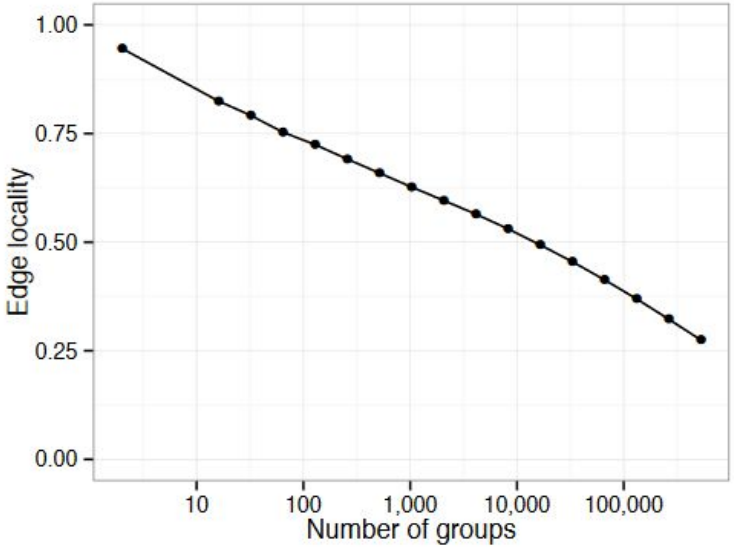


Figure 3: Edge locality (fraction of edges within groups) vs. the number of groups for Facebook's friendship graph.

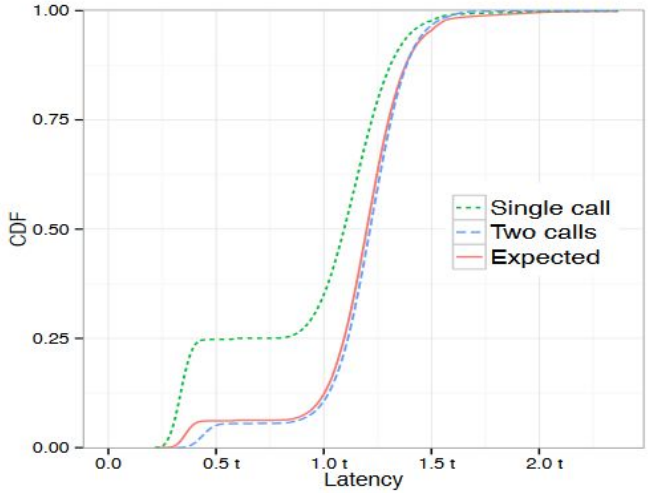


Figure 6: Cumulative distribution of latency for a single request, two requests in parallel, and the expected distribution from two independent samples from the single request distribution, where t is the average latency of a single call

.

some results

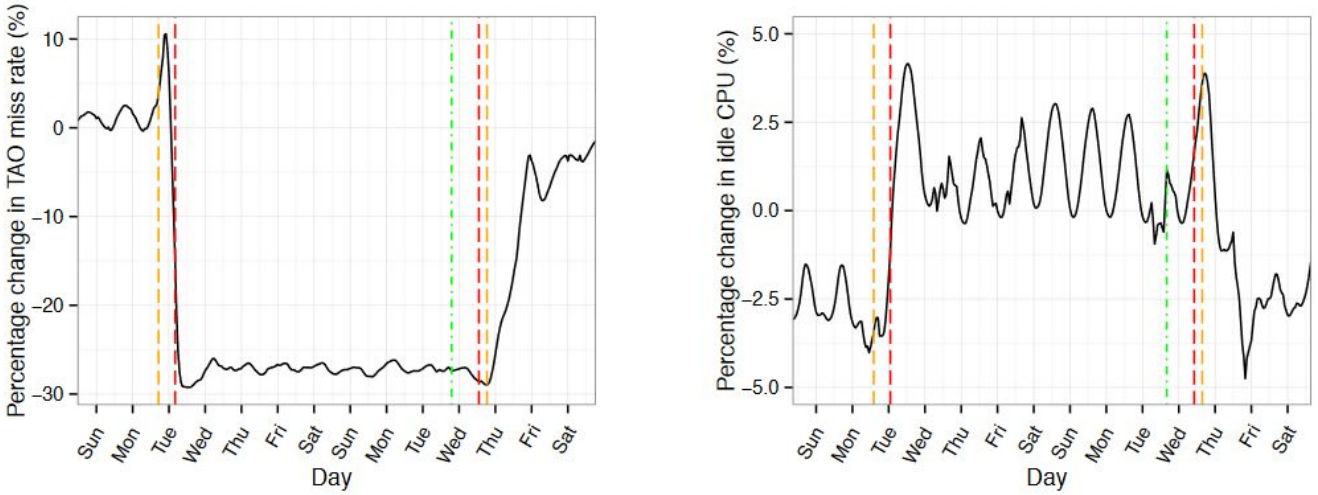


Figure 4: Percentage change in TAO miss rate (left, where lower is better) and CPU idle rate (right, where higher is better) on the Social Hash cluster relative to the cluster with random assignment. Area between red dashed lines: period of the test. Orange dashed lines: traffic shifts. Green dot-dash line: Social Hash Table is updated. The values on the days traffic was shifted (Tuesday and Wednesday, respectively) are not representative

issues and concerns

- Optimized for social networks and might not work for every distributed system
  - Would like to see this implemented for other social networks as well
  - Assumes that you can beneficially group together objects
  - Assumes graph must be reasonably sparse
  - The graph cannot change too rapidly
  - Having many missing keys would create a huge overhead and the approach wouldn’t work

future work

- Their Social Hash framework is elegant and easy to explain but it’s not hard to construct examples where it would not perform well
- Improve the graph partitioning algorithm
- Use machine learning to use query patterns to improve performance
- Incorporate geo-locality considerations for the HTTP routing optimization
- Incorporate alternative replication schemes for further reducing fanout in storage sharded systems
- Dynamic assignment seems a bit opaque

future work

- Fully characterize or at least better understand the optimization-adaptation tradeoff
- As discussed, there are are many avenues to explore with respect to overlapping graph partitions
- Thoughts?

thank you!