StormRider: Harnessing "Storm" for Social Networks

Vaibhav Khadilkar, Murat Kantarcioglu, and Bhavani Thuraisingham
The University of Texas at Dallas
Richardson, Texas, USA
vvk072000@utdallas.edu, muratk@utdallas.edu, bxt043000@utdallas.edu

ABSTRACT

The focus of online social media providers today has shifted from "content generation" towards finding effective methodologies for "content storage, retrieval and analysis" in the presence of evolving networks. Towards this end, in this paper we present StormRider, a framework that uses existing cloud computing and semantic web technologies to provide application programmers with automated support for these tasks, thereby allowing a richer assortment of use cases to be implemented on the underlying evolving social networks.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems*; D.1.3 [**Programming Techniques**]: Concurrent Programming—*Distributed Programming*

1. INTRODUCTION

The rise of social media applications has turned the once privileged realm of web authoring and publishing into a commonplace activity. This has led to an explosion in the amount of user-generated content online. The main concern for social media providers is no longer "content generation" but finding effective methodologies for "content storage, retrieval and analysis". There has been a significant amount of research (see for *e.g.* [1]) that addresses this issue. However, existing work views a network as a series of snapshots, where a snapshot represents the state of a network in a given time period. Therefore, different network operations need to be individually performed over each snapshot. In reality, online social networks are continuously evolving entities and therefore, network operations should be automatically performed as they evolve. Moreover, viewing the problem from this perspective allows us to create a solution that supports advanced, real-world use cases such as the following: (a) Tracking the neighborhood of a given node. This use case is relevant in law enforcement, for example to track the activities of potential criminals/terrorists. (b) Being able to store and access prior snapshots of a network for auditing and verification tasks. Such a use case is relevant in healthcare, for example in tracing the medical history of a patient.

In this paper, we present StormRider, a framework that uses a combination of cloud-based and semantic web-based tools to allow the storage, retrieval and analysis of evolving social networks. In addition, users can perform these operations on networks of their choice by creating custom-built implementations of the interfaces provided in StormRider. The StormRider framework makes use of the following existing tools as basic building blocks: (i) The Storm framework allows StormRider to automatically store, query and analyze data as the underlying network evolves over time. Storm was selected because it is a realtime computation system that guarantees message processing and is scalable, robust and fault-tolerant. (ii) The Jena-HBase framework [2] allows the storage of network data in a RDF representation as well as to query the data using SPARQL. (iii) Apache HBase was used to construct materialized views that store metadata related to nodes in the network. These views allow faster analytics to be performed on the network.

Our contributions: StormRider provides the following novel contributions: (i) The Jena-HBase framework facilitates the of use several semantic web features with social networks such as application of reasoning algorithms, reification, *etc.* (ii) The ability to store, query and analyze evolving networks through the use of novel algorithms (for *e.g.*, approximation algorithms for centrality estimation) implemented in Storm. (iii) Application programmers are provided with simple interfaces through which they can interact with social networks of their choice.

2. STORMRIDER ARCHITECTURE

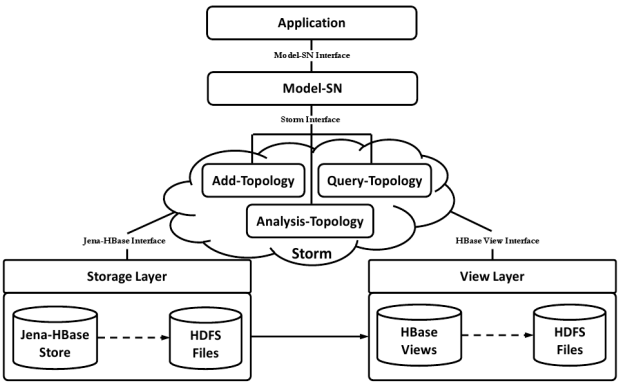

Figure 1: StormRider - Architectural Overview

Figure 1 presents an architectural overview of StormRider. User applications interact with an abstract social network model (Model-SN) which translates high-level user-defined network operations (*viz.* store, query and analyze) into low-level operations on the underlying network representations

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

used by StormRider. The low-level operations are implemented as Storm topologies and are designed to support evolving social networks. A Storm topology represents a graph of computation, where nodes contain the logic of the computation while links between nodes denote how data is passed from one node to another. Storm internally interfaces with the Storage Layer (Jena-HBase), through the Jena-HBase Interface, and the View Layer (HBase tables used as materialized views), through the HBase View Interface, to execute topologies on the underlying networks.

The Storage Layer, composed of Jena-HBase, is used to store networks in a RDF representation in a cloud-based framework. The storage of networks in RDF when combined with topologies defined in Storm allows us to support realistic uses cases such as those given previously (examples (a) and (b) in the Introduction) through the use of concepts such as property-path queries and reification. For additional details about Jena-HBase, an interested reader is referred to our detailed technical report [2]. The View Layer is used to store metadata about nodes that make up a network. The metadata is mainly used to facilitate a speed-up in performance during the analysis of a network.

Additional details of the architecture along with a detailed description of sample Add-, Query- and Analyze-Topologies for the Twitter network are given in [3]. Note that these topologies are only provided as examples with the StormRider framework. Consequently, an application programmer needs to define custom topologies based on their requirements to interact with networks they want to examine.

3. EXPERIMENTAL EVALUATION

As stated earlier, the sample topologies in StormRider have been implemented for Twitter. The Add-Topology is used to add data to the Storage layer as well as to update node-related information in the View layer. The Analyze-Topology is then used to compute degree, closeness and betweenness centrality using the metadata from the View layer. Some of these metrics require shortest path computations which we perform using the landmark-based approximation technique [4]. As a part of our experimental evaluation, we evaluated the effectiveness of this method *vs.* the exact method given in [5] for computing closeness and betweenness centrality on a maximum of 500K Twitter users. The number of nodes in the landmarks set was set to: (total no. of users)/100, where the factor 100 was randomly selected, while the elements in the landmarks set were selected as the top-k nodes with the highest degree. Finally, each experiment was conducted along the following dimensions: (i) Approximation Error: This metric measures the accuracy of StormRider in computing the centrality value *vs.* the exact method and is computed as, $|\hat{l} - l|/l$ where l is the actual centrality value and $\hat{l}$ is the approximation. (ii) Execution Time: This metric measures the time required to perform the approximate and exact calculations of the centrality values. The time for the approximate case is computed as a sum of both, the time required to update the views and the time required to perform the actual centrality computation.

Figure 2 presents the experimental results for the computation of closeness centrality. The graph on the left clearly shows that the magnitude of the approximation error is small even when the number of users increases from 100K to 500K. By observing the graph on the right, we see that the execution time of StormRider's approximate method is faster than

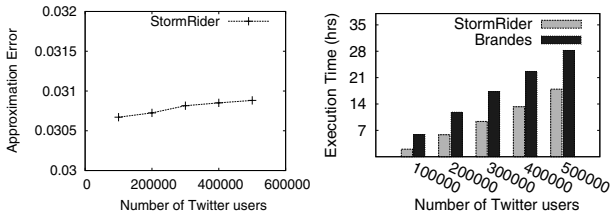

Figure 2: Performance of StormRider for Closeness Centrality computation

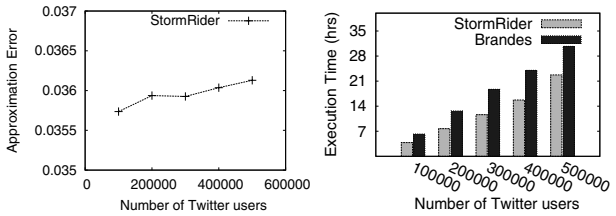

Figure 3: Performance of StormRider for Betweenness Centrality computation

the exact method. This is expected, since the exact algorithm performs k single-source-shortest-path (SSSP) computations while the approximate algorithm requires lesser SSSP computations which leads to a lower overall execution time (where k is the number of nodes in the network).

Figure 3 presents experimental results for betweenness centrality computation, which are very similar to the results given in Figure 2. As before, the graph on the left shows that the magnitude of the approximation error is small even when the number of users is varied while the graph on the right shows that the time required to perform betweenness centrality in StormRider is much better than the exact method.

4. CONCLUSION

In this paper we have presented StormRider, a framework that uses a novel combination of existing cloud computing and semantic web technologies to allow "storage, retrieval and analysis" of evolving online social networks, thus enabling support for several new, realistic use cases.

5. ACKNOWLEDGEMENTS

This work was partially supported by The Air Force Office of Scientific Research MURI Grants FA-9550-08-1-0265 and FA-9550-08-1-0260, National Institutes of Health Grant 1R01LM009989, National Science Foundation (NSF) Grant Career-CNS-0845803, and NSF Grants CNS-0964350, CNS-1016343. We thank Dr. Robert Herklotz for his support.

6. REFERENCES

[1] G. Erétéo, M. Buffa, F. Gandon, and O. Corby. Analysis of a real online social network using semantic web frameworks. In *ISWC*, pages 180–195, 2009.

[2] V. Khadilkar, M. Kantarcioglu, P. Castagna, and B. Thuraisingham. Jena-HBase: A Distributed, Scalable and Efficient RDF Triple Store. Technical report, 2012. `http://www.utdallas.edu/~vvk072000/Research/Jena-HBase-Ext/tech-report.pdf`.

[3] V. Khadilkar, M. Kantarcioglu, and B. Thuraisingham. StormRider: Harnessing "Storm" for Social Networks. Technical report, 2012. `http://www.utdallas.edu/~vvk072000/Research/StormRider/tech-report.pdf`.

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

[5] U. Brandes. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.