A Distributed Platform to Ease the Development of Recommendation Algorithms on Large-Scale Graphs

Alejandro Corbellini
ISISTAN Research Institute, CONICET-UNCPBA
Campus Universitario, Tandil (B7001BBO), Buenos Aires, Argentina
acorbellini@isistan.unicen.edu.ar

Abstract

The creation of novel recommendation algorithms for social networks is currently struggling with the volume of available data originating in such environments. Given that social networks can be modeled as graphs, a distributed graph-oriented support to exploit the computing capabilities of clusters arises as a necessity. In this thesis, a platform for graph storage and processing named Graphly is proposed along with GraphRec, an API for easy specification of recommendation algorithms. Graphly and GraphRec hide distributed programming concerns from the user while still allowing fine-tuning of the remote execution. For example, users may customize an algorithm execution using job distribution strategies, without modifying the original code. GraphRec also simplifies the design of graph-based recommender systems by implementing well-known algorithms as "primitives" that can be reused.

1 Introduction

The creation of novel and more effective recommendation algorithms for social networks is currently facing major challenges regarding data processing. This can be explained by two facts. Firstly, since social networks grown in popularity, the volume of data originating from these platforms has grown in size as well. Secondly, the development of most experimental recommendation algorithms for social networks are implemented as single-machine, single-threaded applications [Durand *et al.*, 2013; Wang *et al.*, 2014; Guo and Lu, 2007].

In the field of distributed graph processing frameworks, there are many execution models and architectures [Malewicz *et al.*, 2010; Low *et al.*, 2012] but are mainly in-memory which, for small clusters imposes a hard limit on scalability. On the other hand, graph databases[1][2] provide persistence, but usually do not provide execution facilities neither custom task distribution.

[1]Titan, http://thinkaurelius.github.io/titan/

[2]Neo4J, http://neo4j.com/

Therefore, in our view, there is a pressing need for a new support that takes advantage of distributed data stores and provides abstractions to (optionally) allow users to exploit low-level tuning execution mechanisms, while additionally offering a simple graph traversal API tailored to recommendation algorithms. The thesis proposes Graphly, a distributed graph database and processing platform that simplifies the creation of distributed graph algorithms by providing a simple query API for graphs, while supporting fine tuning of task execution. This fine tuning is provided through mapping strategies, a non-invasive way of distributing queries through a computer cluster.

GraphRec extends Graphly's query API by providing well-know recommendation algorithms for graphs, e.g., HITS [Najork, 2007], SALSA [Najork, 2007], WhoToFollow [Gupta *et al.*, 2013], among others such as [Armentano *et al.*, 2012]. Together, Graphly and GraphRec provide a platform that can also be of great use for the recommendation systems research community to develop and evaluate novel recommendation algorithms on real-sized graphs. Furthermore, the mapping strategies provided by Graphly allow to adjust these algorithms to the cluster being used.

The remainder of this document is organized as follows. Section 2 introduces Graphly and GraphRec. Section 3 illustrates how mapping strategies affect the execution of recommendation algorithms. Finally, Section 4 elaborates some conclusions and future work.

2 Graphly and GraphRec

Graphly is a graph database and processing platform targeted to small, heterogeneous cluster setups. Besides providing tools to store and query a persistent graph database, Graphly implements *job mapping strategies* that allow tuning the algorithm's execution to the current cluster setup (e.g. round robin, location aware).

GraphRec is an extension of Graphly's query API that includes several well-known algorithms as built-in primitives from which novel algorithms can be built upon. A usage example of the SALSA algorithm [Najork, 2007] in GraphRec would be:

```
g.v(T).as(GraphRec).salsa("a","h",100)
```

Where the *as* operation casts the current query to a GraphRec query, and then, the *salsa* operation is executed during 100

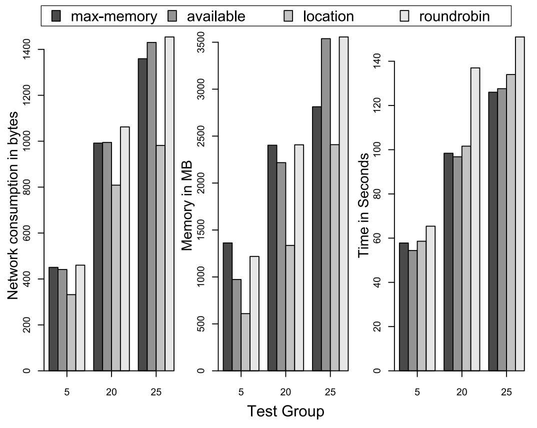

Figure 1: SALSA using different mapping strategies.

iterations. The authority and hub values are stored on each vertex under the *a* and *h* properties respectively. The *salsa* operation updates the authority and hub scores according to SALSA algorithm:

```
1: forall $v$ in $V_{div}$
2:     $auth(v) = \sum_{u \to v} \sum_{u \to w} \frac{auth(w)}{out(u) in(w)}$
3:     $hub(v) = \sum_{v \to u} \sum_{w \to u} \frac{hub(w)}{in(v) out(w)}$
4: endfor
```

3 Experiments

As part of a more extensive experimental evaluation, we tested the SALSA implementation in GraphRec on an 8-node cluster with different job mapping strategies and 3 groups of users: a 5-user group, a 25-user group and a 50-user group. The first strategy is a Location-Aware strategy that maps vertices to where they are located. The second and third strategies uses the maximum and the currently available RAM memory measures respectively to map vertices. Finally, the Round Robin metric simply divides vertices equally among cluster nodes. Figure 1 shows the results in terms of recommendation time, network usage and memory usage. From the results, we argue that strategies can be helpful on different scenarios. For example, if the amount of RAM memory is not the same for each node, a memory-based strategy might scale better for huge datasets but will perform poorly in terms of network usage. Moreover, if the cluster is shared among different users, a dynamic memory strategy might work better than a static strategy. On the other hand, if the cluster uses an slow network, e.g. Wi-Fi, a location-aware strategy provides better performance but does not take into account the capabilities of each node.

4 Conclusions

The main contribution of this work is a support platform for creating new graph-based recommendation algorithms that hides distributed concerns while still allowing fine tuning of the underlying distributed execution.As future work, the inclusion of other non-invasive strategies like job-stealing and caching strategies are planned. Another natural extension of this work is the creation of hybrid strategies, e.g. a Location-Aware strategy that takes into account memory or CPU characteristics or both to split jobs. Currently the division of work is implemented through a fork-join algorithm, but the inclusion of alternative processing models for graphs, e.g. Pregel [Malewicz *et al.*, 2010], is being implemented. Providing implementations of commonly used algorithms in both models (Fork-Join and Pregel) will help to compare the impact of model selection on the algorithm and how mapping strategies affect response time and cluster usage on different execution models.

References

[Armentano *et al.*, 2012] M. Armentano, D. Godoy, and A. Amandi. Topology-based recommendation of users in micro-blogging communities. *Journal of Computer Science and Technology*, 27(3):624–634, 2012.

[Durand *et al.*, 2013] G. Durand, N. Belacel, and F. LaPlante. Graph theory based model for learning path recommendation. *Information Sciences*, 251:10–21, 2013.

[Guo and Lu, 2007] X. Guo and J. Lu. Intelligent e-government services with personalized recommendation techniques. *International Journal of Intelligent Systems*, 22(5):401–417, 2007.

[Gupta *et al.*, 2013] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. WTF: The who to follow service at Twitter. In *Proceedings of the 22th International World Wide Web Conference (WWW 2013)*, pages 505–514, Rio de Janeiro, Brazil, 2013.

[Low *et al.*, 2012] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. Hellerstein. Distributed GraphLab: A framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

[Malewicz *et al.*, 2010] G. Malewicz, M. H. Austern, A. J. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 International Conference on Management of Data (SIGMOD '10)*, pages 135–146, Indianapolis, IN, USA, 2010.

[Najork, 2007] Marc a. Najork. Comparing the effectiveness of hits and salsa. *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07*, page 157, 2007.

[Wang *et al.*, 2014] X. Wang, J. Ma, and M. Xu. Group recommendation for Flickr images by 4-order tensor decomposition. *Journal of Computational Information Systems*, 10(3):1315–1322, 2014.