Tink: A Temporal Graph Analytics Library for Apache Flink

Wouter Lightenberg, Yulong Pei, George Fletcher, Mykola Pechenizkiy

Eindhoven University of Technology

wouter@onzichtbaar.net, {y.pei.1, g.h.l.fletcher, m.pechenizkiy}@tue.nl

ABSTRACT

We introduce the Tink library for distributed temporal graph analytics. Increasingly, reasoning about temporal aspects of graph-structured data collections is an important aspect of analytics. For example, in a communication network, time plays a fundamental role in the propagation of information within the network. Whereas existing tools for temporal graph analysis are built stand alone, Tink is a library in the Apache Flink ecosystem, thereby leveraging its advanced mature features such as distributed processing and query optimization. Furthermore, Flink requires little effort to process and clean the data without having to use different tools before analyzing the data. Tink focuses on interval graphs in which every edge is associated with a starting time and an ending time. The library provides facilities for temporal graph creation and maintenance, as well as standard temporal graph measures and algorithms. Furthermore, the library is designed for ease of use and extensibility.

ACM Reference Format:

Wouter Lightenberg, Yulong Pei, George Fletcher, Mykola Pechenizkiy. 2018. Tink: A Temporal Graph Analytics Library for Apache Flink. In *The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France*. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3184558.3186934>

1 INTRODUCTION

Analytics on temporal graphs has recently attracted increasing interest from research communities [7]. For example, in a social network, time windows on the edges of the graph play a fundamental role in the propagation of information within the network. Temporal graph analytics is an important and ubiquitous task with many application domains, and hence has experienced recent intensive study, e.g., [9, 12]. Although different systems have been developed to analyze temporal graphs, two major limitations exist: (1) most systems are stand-alone, and hence their data structures and algorithms are diverse and it is difficult to reuse code and compare different systems; (2) most systems focus on time instances rather than intervals so it is difficult to generalize them to analyze different types of temporal graphs, e.g., temporal property graph model introduced in Section 2. To solve these problems, we have recently introduced Tink, a temporal graph analytics library [8].¹ Tink is built on top of the DataSet API of Apache Flink². It makes use of Gelly³ and focuses on interval graphs in which every edge

¹<https://github.com/otherwise777/Tink>

²<https://flink.apache.org/>

³<https://ci.apache.org/projects/flink/flink-docs-master/dev/libs/gelly/>

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW'18 Companion, April 23–27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5640-4/18/04.

<https://doi.org/10.1145/3184558.3186934>

Figure 1: The Tink library in the Apache Flink stack (adapted from [1]).

is associated with a starting time and an ending time. Flink is one of the most representative systems for distributed big data analytics. Gelly is the graph processing engine of Flink built on top of the DataSet API. Gelly provides methods to create, transform and modify graphs, as well a library of graph algorithms including PageRank and label propagation. The Stack of Flink is displayed in Figure 1, where Tink at the top-right corner.

Related Work. Tink differs from existing tools for temporal graph analytics on three aspects: it is the only one built on top of an existing general-purpose data processing framework, the only one supporting time intervals instead of time instances and the only one does not use snapshot storage. For instance, Chronos is a storage and execution engine designed and optimized specifically for running in-memory iterative graph computation on temporal graphs [6]. Kineograph provides a distributed platform for incoming data to construct a continuously changing graph and the difference of Kineograph is the separation of graph updates and computations [4]. Graphstream’s main focus is not temporal graph analytics, but it is one of the larger open-source projects that enables these analyses to be built upon [5].

2 TEMPORAL PROPERTY GRAPH MODEL

The Tink data model is an extension of the well-established property graph model, where a graph instance consists of a finite set of labelled nodes, a set of labelled directed edges between these nodes, and both nodes and edges are associated with finite sets of property-values [2]. Our data model extends this model such that each edge has an associated time window. An example of a temporal property graph model is shown in Figure 2.

Formally, given an infinite set of (node and edge) labels L , an infinite set of property names P , an infinite set of values V and an infinite set of time windows $T = \{[i, j] \mid i, j \in \mathbb{N}, i \leq j\}$, a *temporal property graph* is a tuple $(N, E, \rho, \lambda, \sigma, \tau)$ where

- N is a finite set of vertices; E is a finite set of edges;
- $\rho : E \rightarrow (N \times N)$ is a function from edges to source-target node pairs;

Figure 2: Temporal property graph model example where nodes represent people and edges represent their relationships in time.

- $\lambda : (N \cup E) \rightarrow \text{SET}(\mathbf{L})$ is a function from vertices and edges to sets of labels (i.e., $\text{SET}(\mathbf{L})$ is the set of all finite subsets of $\mathbf{L}$);
- $\sigma : (N \cup E) \times \mathbf{P} \rightarrow \mathbf{V}$ is a partial function from the property names of vertices and edges to their values; and,
- $\tau : E \rightarrow \mathbf{T}$ is a function from edges to time windows.

In our example graph of Figure 2, we have that $\lambda(15) = \emptyset$, $\rho(15) = (3, 6)$, $\lambda(14) = \{Knows, Likes\}$, $\sigma(3, sex) = Male$, and $\tau(14) = [5, 6]$.

3 SYSTEM DESIGN

The functionality of Tink consists of four components: (1) Graph creation: a temporal graph can be created from text sources, collections and Flink’s datasets. (2) Graph mutations: temporal graphs can be evolved, i.e. adding and removing edges and vertices. (3) Graph algorithms: Several temporal graph algorithms have been implemented including shortest path earliest arrival time (SSSTPEAT) and shortest path fastest path (SSSTFPF) [12]. (4) Graph metrics: Several temporal centrality metrics have been implemented including temporal betweenness and temporal closeness [7]. The library is designed for ease of use and with support for ease of extensibility.

Most of the current algorithms in the library use the Signal/Collect model [10]. In this model a vertex produces messages (signal) and updates its value based on the messages it receives (collect). In every iteration we create signals from the nodes and collect them. Each iteration is called a superstep. If there are no more messages to be collected or the maximum number of iterations has been reached the algorithm will converge.

4 EXPERIMENTS

To verify and demonstrate the functionality of Tink, we conduct experiments on both synthetic and real datasets [8]. First, we run tests on Tink for performance w.r.t different numbers of cores. Secondly, we inspect the results on the execution of a graph with different interval distributions.

Datasets. For synthetic dataset, we generated several datasets with gMark, a state of the art graph generator [3]. For real dataset, Facebook message graph [11] has been used. Note that for synthetic

data the starting time is randomly chosen between 1 and 100, and for real data the starting time is the timestamp associated with edges. Four different distributions, i.e., constant, normal, uniform and Zipfian, have been used to generate the interval duration for edges in both datasets.

Results. For *parallelization* experiment, Tink is made to be run on different machines and different cores simultaneously. We test the SSSTPEAT algorithm on Facebook messages graph with the different interval distributions. The results show that the algorithm runtime decreases as more cores are added and it converges around 4 cores. At 7 cores there is a small bump in the performance, this is probably because of the machine having 8 cores and that several cores are in use for background processes.

The results of *interval distributions* show that the directed graphs are always faster than the undirected graphs. This can be explained by Tink's implementation of directed graphs. In Tink an undirected graph is a directed graph in which the edges are duplicated and reversed. The result is a graph twice the size which has to be processed. When comparing the different distributions we only see slight differences like the *constant 0* being the slowest, this can be explained by the nature of the SSSTPEAT algorithm which will converge faster if there are no nodes left to visit, if every node has a constant interval time of 0 there are many nodes to be visited before the algorithm converges, thus resulting in a longer duration.

5 CONCLUSIONS

We introduced Tink, a library for scalable temporal graph analytics. We developed a number of novel algorithms to enable an efficient and scalable temporal graph analytics in a distributed environment. We demonstrated the feasibility and usefulness of our approach by performing ease-of-use and empirical studies. Further, we demonstrated the ease of development of temporal analytics algorithms by using Tink’s APIs.

REFERENCES

- [1] 2018. Intro to Apache Flink. (2018). <https://flink.apache.org/introduction.html>
- [2] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. 2017. Foundations of modern query languages for graph databases. *ACM Comput. Surv.* 50, 5 (2017), 68:1–68:40.
- [3] Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George HL Fletcher, Aurélien Lemay, and Nicky Adavokaat. 2017. gMark: schema-driven generation of graphs and queries. *IEEE TKDE* 29, 4 (2017), 856–869.
- [4] Raymond Cheng et al. 2012. Kinograph: taking the pulse of a fast-changing and connected world. In *ECCS'12*. ACM, 85–98.
- [5] Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. 2007. Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In *ECCS'07*.
- [6] Wentao Han, Youshan Miao, Kaiwei Li, Ming Wu, Fan Yang, Lidong Zhou, Vijayan Prabhakaran, Wenguan Chen, and Enhong Chen. 2014. Chronos: a graph engine for temporal graph analysis. In *ECCS'14*. ACM, 1.
- [7] Petter Holme. 2015. Modern temporal network theory: A colloquium. *Eur Phys J B* 88 (2015), 234–264.
- [8] Wouter Ligtenberg. 2017. Tink, a temporal graph analytics library for Apache Flink. *Eindhoven University of Technology* (2017).
- [9] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in Temporal Networks. In *WSDM*. 601–610.
- [10] Philip Stutz, Abraham Bernstein, and William Cohen. 2010. Signal/collect: graph algorithms for the (semantic) web. *ISWC'10* (2010), 764–780.
- [11] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. 2009. On the evolution of user interaction in facebook. In *WOSN'09*. ACM, 37–42.
- [12] Huanhuan Wu, James Cheng, Yiping Ke, Silu Huang, Yuzhen Huang, and Hejun Wu. 2016. Efficient Algorithms for Temporal Path Computation. *IEEE TKDE* 28, 11 (2016), 2927–2942.