Citation for published version (APA):

DOI:
10.1145/3184558.3186934

Document status and date:
Published: 23/04/2018

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Tink: A Temporal Graph Analytics Library for Apache Flink

Wouter Lightenberg, Yulong Pei, George Fletcher, Mykola Pechenizkiy
Eindhoven University of Technology
wouter@onziehtbaar.net, {y.pei, g.h.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:

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 experience 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].

Tink is built on top of the DataSet API of Apache Flink. It makes use of Gelly and focuses on interval graphs in which every edge 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 1: The Tink library in the Apache Flink stack (adapted from [1]).
The functionality of Tink consists of four components: (1) Graph creation: Several temporal graph algorithms have been implemented including shortest path earliest arrival time (SSSTPEAT) [12]. (4) Graph metrics: Graph algorithms: Several temporal graph algorithms have been implemented such as the shortest path fastest path (SSSTFPFP) [12].

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 (SSSTFPFP) [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 with 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 algorithms by using Tink’s APIs.

REFERENCES