Temporal Reasoning on Large-Scale Graphs

Pietro Daverio[1] , **Hassan Nazeer Chaudhry**[1] , **Alessandro Margara**[1] , **Matteo Rossi**[1]
[1]Politecnico di Milano
pietro.daverio@mail.polimi.it, {hassannazeer.chaudhry, alessandro.margara, matteo.rossi}@polimi.it

Abstract

Many applications adopt graph data structures to model relations between entities, which continuously change over time. Reasoning on the temporal evolution of large-scale graphs is challenging, as it involves studying how events such as the addition or removal of an edge affect the structure and properties of the graph. We present our ongoing research on the definition of abstractions and processing mechanisms to reason on dynamic graphs. We propose a data and processing model and a system architecture that supports them. We discuss preliminary results and show our plans for future work.

1 Background and Motivations

Graph-based data structures model relations between entities in many diverse applicative domains: social networks include "follower" relations among users, e-commerce websites link products to customers who buy and review them, maps connect locations with roads. In most scenarios, entities and relations evolve over time: for instance, social networks can receive thousands of new posts every second. In this context, capturing relevant patterns in the temporal evolution of graphs enables prompt reactions to critical situations. For instance, forecasting how road congestions can propagate over time might help navigation systems suggest alternative routes. Similarly, detecting sudden changes in the reviews of products can help e-commerce websites detect spammers, improve user experience, or optimize advertisement campaigns.

Over the last decade, several approaches have been proposed to perform computations over large-scale graph data structures. They range from graph databases that propose query languages to search for subgraphs having a specific shape (Angles and Gutierrez 2008), to processing abstractions that compute relevant results from graphs, such as vertex-centric abstractions pioneered by Pregel (Malewicz et al. 2010) and now available in most big data platforms (Gonzalez et al. 2014). However, most of these approaches work on static graphs and neglect the temporal dimension. At the same time, composite event recognition (CER) systems offer abstractions and mechanisms to reason about continuous streams of events (Cugola and Margara 2012; Giatrakos et al. 2019). However, most of these systems do not support reasoning about large-scale evolving state as necessary in graph-based applications.

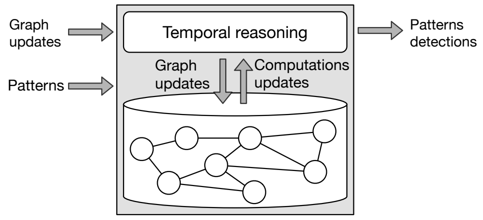

Figure 1: Data and processing model

In summary, devising suitable abstractions and processing techniques to reason on the dynamic evolution of large-scale graphs remains an open problem. The talk will present our ongoing research, summarized in the following Section 2. It will discuss a data and processing model to reason on the temporal evolution of large-scale graphs, present implementation strategies and preliminary results, explore open problems and suitable research directions.

2 Reasoning on Dynamic Graphs

We overview our proposal for integrating graph computations and temporal pattern evaluation to build a system that captures the evolution of large-scale dynamic graphs.

Data and processing models Figure 1 shows a conceptual overview of our proposed data and processing models. A *temporal reasoning* system stores a graph that continuously evolves over time according to a stream of input *graph updates*. Users install *patterns* that predicate over the temporal evolution of the graph, and the system notifies them when some of the installed patterns occur.

We assume vertices and edges to have associated properties, which we denote as their *state*. The input stream contains time-annotated updates: addition and removal of vertices or edges, or updates to the value of their state. Pattern evaluation is triggered by input updates: whenever an update is received, the system evaluates all installed patterns and outputs a notification of detection for each and every pattern that is satisfied. Patterns predicate on the values resulting from *computations* on the graph that take place at different points in time. In our current prototype, computations can derive new values starting from the state of individual vertices/edges or groups of them, for instance using vertex-centric programs to compute values and aggregations

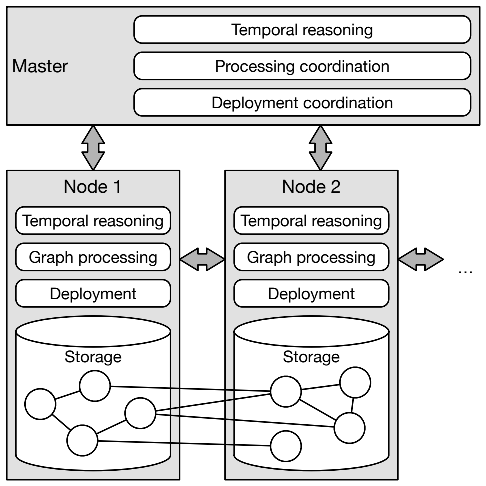

Figure 2: System architecture

to compose them. Patterns capture the temporal evolution of graphs by correlating the results of computations at different points in time.

System infrastructure We implemented our model in a distributed system having the architecture presented in Figure 2. It adopts a master-slave approach where a master receives input updates and coordinates the work of several processing nodes that can be deployed on the same or different physical machines for scalability. In summary, the system comprises the following layers.

Storage. The graph is partitioned across nodes, each of them storing its partition in main memory for improved access time. Nodes store the current state of the vertices and edges in their partition, as well as previous states that might still be useful for pattern detection.

Deployment. The deployment layer is responsible for associating vertices and edges with nodes. Currently, we implement a deployment strategy that balances the number of vertices and edges associated with each node. Arguably, deployment is the most critical component of the architecture, as it influences the distribution of processing and the need for costly inter-node communication and coordination.

Graph processing. This processing layer is responsible for executing graph computations (on the current or on some previous state of the graph). We currently provide a library of computations, including several vertex-centric algorithms, and we plan to extend to different classes of algorithms in the future. During computations, nodes can directly exchange data with each other. If necessary, the master can act as a synchronization point: for instance, vertex-centric computations work in phases which are controlled by the master.

Temporal reasoning. Temporal reasoning correlates the results of graph computations at different points in time and implements the temporal pattern recognition logic. We adopt a hierarchical approach, where individual nodes perform all the steps that are possibile based on their local view of the graph, and the master integrates the contributions of multiple nodes. Temporal reasoning governs the execution of graph processing, starting expensive computations only when strictly necessary for the detection of patterns.

Current results and future work We implemented the architecture in Figure 2 as a JVM-based system on top of an actor-oriented framework. Our preliminary evaluation shows that: (i) most graph processing computations scale linearly with the number of nodes and provide performance on a par with state-of-the-art tools for large-scale static graph processing; (ii) temporal reasoning can boost performance by avoiding computations that cannot lead to pattern detection.

We are currently working to extend the framework in various ways, and in particular: (i) Introduce additional processing abstractions (e.g., subgraph query and retrieval, as implemented in graph databases) to integrate them in the pattern specification language and promote the applicability of the system to a broader range of domains. (ii) Implement pattern rewriting techniques to further optimize pattern evaluation: given the high cost of some graph processing computations, rewriting patterns to limit their probability of execution or the scope of their analysis has large time-saving potentials. (iii) Implement deployment strategies that take into account the structure of the graph to minimize inter-node communication, and migration strategies that consider the evolution of the graph over time.

Conclusions While many application domains can benefit from reasoning on the temporal evolution of graphs, the topic has only been partially addressed in reasearch. Integrating temporal reasoning capabilities and large-scale graph computations presents significant challenges that call for a new class of systems. In this talk, we present ongoing research in this area, discuss the results we achieved and the open challenges for future investigations.

References

Angles, R., and Gutierrez, C. 2008. Survey of graph database models. *ACM Computing Surveys* 40(1):1:1–1:39.

Cugola, G., and Margara, A. 2012. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys* 44(3):15:1–15:62.

Giatrakos, N.; Alevizos, E.; Artikis, A.; Deligiannakis, A.; and Garofalakis, M. 2019. Complex event recognition in the big data era: a survey. *The VLDB Journal*.

Gonzalez, J. E.; Xin, R. S.; Dave, A.; Crankshaw, D.; Franklin, M. J.; and Stoica, I. 2014. Graphx: Graph processing in a distributed dataflow framework. In *Proc. of the USENIX Conf. on Operating Systems Design and Implementation*, OSDI'14, 599–613. USENIX.

Malewicz, G.; Austern, M. H.; Bik, A. J.; Dehnert, J. C.; Horn, I.; Leiser, N.; and Czajkowski, G. 2010. Pregel: A system for large-scale graph processing. In *Proc. of the Int. Conf. on Management of Data*, SIGMOD '10, 135–146. ACM.