Near Real-time Distributed Temporal Graph Processing

Ben Steer, Alhamza Alnaimi, Félix Cuadrado
Queen Mary University of London
{b.a.steer,a.alnaimi}@se12.qmul.ac.uk, felix.cuadrado@qmul.ac.uk

Abstract

Dynamic graph analysis has gathered significant attention from researchers due to the potential to explore the temporal dimension, together with the existing relations, in a dataset. Supporting this type of computation at scale brings new challenges regarding data management and updates, and efficient execution of operations.

Several recent works have explored some of the challenges behind dynamic graph processing. Kineograph [2] processes evolving graphs generated from streams, and Vaquero et al [4] adapts partitioning to the graphs structural changes. These works model dynamic graphs as a set of snapshots, losing fine-grained temporal information. ImmortalGraph [3] proposes a temporal graph indexing scheme which combines snapshots and individual changes, allowing efficient temporal traversals. This approach assumes graph data has been previously collected and stored, but does not consider computation from streams.

In this work we present a scalable system for performing temporal graph processing on information streams. We address two main challenges; temporal graph analysis beyond snapshots (such as computing temporal shortest paths [1]); and efficient generation and update of temporal graphs from information streams. The proposed solution is built on top of Apache Storm to reduce the initial information stream into a graph, and Apache Spark GraphX for representation and computation on the graph.

Our graph generation component transforms an information stream into a collection of graph updates denoting edge/vertex addition or removal. Temporal graph properties (such as sliding window for edge insertion) are automatically evaluated and added as further graph changes. Processing involves a reduction step, which evaluates the interactions between these commands, to remove conflicting or redundant activities. The result of a reduction is a set of order independent changes which can be applied without conflicts in a deterministic manner. Reduction is run at set intervals, creating micro batches from the new data. The resulting batches maintain the integrity of the final graph, with a small loss of temporal resolution, but reduce the cost of updating the dynamic graph.

The processing system represents the temporal graph as an RDD of edges containing their source, destination and attributes. Creation and removal timestamps are encoded as edge attributes. This enables efficient graph updates, as mapping new time values to an edge is less costly than removal from the data structure. The structure also simplifies temporal graph algorithm computation, as the filter operation allows selecting specific graph time windows.

We have evaluated the system on a cluster of 15 HP Moonshot 1500 nodes with a total of 480GB memory. We processed a stream of 200,000 events per second, seeded from Twitter data. The system computes temporal shortest paths in a similar manner to Pregel's Breadth First Search implementation, but with nodes considering edge intervals of existence to suggest paths to their neighbours.

References

[1] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *Int. Journal of Parallel, Emergent and Distr. Systems*, 27(5), 2012.

[2] R. Cheng, J. Hong, A. Kyrola, Y. Miao, X. Weng, M. Wu, F. Yang, L. Zhou, F. Zhao, and E. Chen. Kineograph: taking the pulse of a fast-changing and connected world. In *ACM Eurosys 2012*, April.

[3] Y. Miao, W. Han, K. Li, M. Wu, F. Yang, L. Zhou, V. Prabhakaran, E. Chen, and W. Chen. Immortalgraph: A system for storage and analysis of temporal graphs. *ACM Trans. on Storage*, 2015.

[4] L. Vaquero, F. Cuadrado, D. Logothetis, and C. Martella. Adaptive partitioning for large-scale dynamic graphs. In *IEEE ICDCS 2014*.

Near Real-time Distributed Temporal Graph Processing

B. A. Steer, A. Alnaimi, F. Cuadrado

Queen Mary University of London

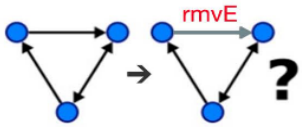
Motivation

Recent systems offer sophisticated solutions to graph processing, but often lack important features or elements.

- Immortalgraph offers temporal graph processing, but lacks the element of parallelism, which may inflict a performance hit [1].
- Kineograph, which allows for graph computation in a distributed environment, only computes and represents snapshots as static points [2].

We, therefore, suggest a framework which allows for truly dynamic graph computation without sacrificing temporal information.

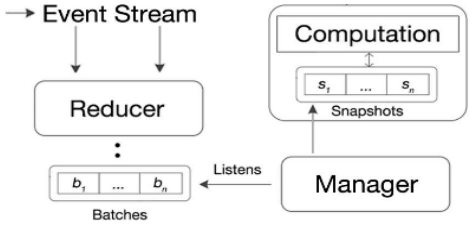
Goals:
1. Allow for queries to run independently.
2. Dynamically evolving graphs.
3. Focus on temporal graph computations.
4. Low performance overhead.



Architecture

The framework architecture consists of *three* main components:

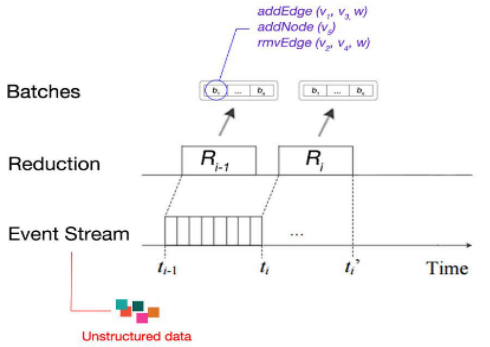
- The 'Reducer' is in charge of converting the event stream (simulated, unstructured data) into its graph representation; in turn producing 1-to-n *microbatches*.
- These batches are listened to and picked up by the 'Manager'; which is in charge of processing the continuous stream of batches, converting graph activities into its temporal representaiton and updating the graph on the fly.
- Consolidated changes of the graph are snapshot every Δ seconds, on which graph computation is executed.



Reducer

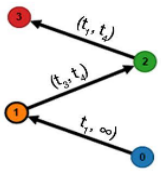
- Filter and convert the event stream into structured graph events.
- Split the events based on their source id. Here we chose hash partioning because of its high scalability to produce balanced partition.
- Remove conflicting events:

addEdge (v_1, v_3, *w*)
addEdge (v_2, v_3, *w*)
rmvNode (v_2)
addEdge (v_2, v_5, *w*) → *addNode* (v_1)
addNode (v_3)
addEdge (v_2, v_5, *w*)



Graph Computation

- Creation and removal timestamps are encoded as edge attributes, allowing for efficient graph updates.



Temporal graph algorithms can then be executed on a snapshot *S*, *e.g*:

Input : A temporal graph G = (V, E) in its edge representation, source vertex x and time range [tα, tω]
Output : The earliest arrival time from x to every vertex v ∈V within [tα, tω]

- Map initial vertex values to *PositiveInfinity*, source vertex to *0*
- for each explored node:
  if n_{value} <= t_{start}:
  $t_{start} + 1$
  elif n_{value} <= t_{end}:
  $n_{value} + 1$

Continuing Work

Having achieved the above functionalities, we aim to further improve the follwing areas;

- Partitioning optimization for graph computation'; here we suggest for the implementation of adaptive partitioning as described in [3].
- Modification of the Spark RDD structure for more efficient updates.

References

[1] Y. Miao, W. Han, K. Li, M. Wu, F. Yang, L. Zhou, V. Prabhakaran, E. Chen, and W. Chen. ImmortalGraph: A System for Storage and Analysis of Temporal Graphs. In *ACM Trans. on Storage 2015*.

[2] R. Cheng, J. Hong, A. Kyrola, Y. Miao, X. Weng, M.Wu, F. Yang, L. Zhou, F. Zhao and Enhong Chen, Kineograph: taking the pulse of a fast-changing and connected world. In *ACM Eurosys 2012*, April.

[3] L. Vaquero, F. Cuadrado, D. Logothetis, and C. Martella. xDGP: A dynamic graph processing system with adaptive partioning. *IEEE ICDCS 2014*.