DynamoGraph:
Large-scale Temporal Graph Processing and its Application Scenarios
# Outline

## Introduction
- Motivation
- Addressed Application Areas
- Hypothesis

## Related Work Preliminaries
- Temporal Graphs
- Large Graphs
- Distributed Computing

## DynamoGraph
- Partitioning Strategies
- Temporal Maps
- Distributed Processing
When people are networked, their power multiplies geometrically. [...] They can reach out and instantly tap the power of other machines [and] people, essentially making the entire network their computer.

— Scott McNeely (Sun Microsystems)
Motivation


Graphs have a Temporal Dimension

- **Static snapshots** of a graph give an **incomplete** view

- Many **systems** modelled by graphs are in fact **highly dynamic**
  - Social networks
  - Politics / communication
  - Biological processes
  - WWW / the Internet
Real-World Networks are Large

They …

- … exceed the **memory capacity** of a single computer
- … cannot be **visualised** with traditional graph vis methods
- … are not **feasible to process** with traditional (sequential) tools and algorithms
- … keep growing

**Examples from internetlivestats.com**
- Facebook: 1,818,823,208
- Websites: 1,141,592,432

28th of January, 2017
Hypothesis

• **Temporal Graph Partitioning**
  traditional graph partitioning schemes can be used

• **Distributed Temporal Graph Storage**
  self contained (temporal) vertex representation allows for mobile data storage

• **Distributed Temporal Graph Processing**
  distributed code, executed locally near the data can be used to process over very large datasets
Temporal Graphs

Graph $G$ is a pair $(V, E)$ where $V$ denotes the set of vertices and $E$ denotes the set of edges between any $v, e \in V$

A temporal graph $T$ can be interpreted as a set of graphs $T = \{G_1, G_2, G_3, \ldots, G_t\}$ where each $G_x = (V_x, E_x)$ $G_x$ is called a static snapshot at time $x$

And $G_{im..in}$ a selection of multiple $G_x$ from $T$ is a static snapshot for a timespan

---

Graph Databases

Popular graph databases picked up temporal support only recently

- Extension on the **time-stamped edges** concept (Kempe, Kleinberg, Kumar 2002)

- Introduction of **intermediate temporal vertexs** allows **efficient** temporal queries

  get all edges at time \( T_2 \)

- Focus is on **distributed storage** but not processing

---

Distributed Matrix Processing

• (Sparse) **adjacency matrices** are a popular in-memory model for graphs

• Many **popular graph algorithms** have **efficient implementations** for matrices

• Distributed matrix processors (multiplicators) can be used to cope with large scale graphs

• Temporal aspect hard to integrate

---

Distributed Graph Processing

- Dominantly implementations on top of Big Data systems
- Data-set is living on a distributed file-system
- Graph processing jobs are implemented as MapReduce jobs
- Most popular: Pregel, by Google but with Open Source implementations (Giraph, GPS, etc.)
- Temporal aspects not covered in framework

---

G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A System for Large-Scale Graph Processing. ACM SIGMOD, 2010.
Temporal Partitioning Strategies

Growth in $V$ typically faster than along the temporal dimension

- host1
- host2
- host3
Structural Partitioning Strategies
Vertices as Temporal Maps

- **Multiple versions** of the same vertex are stored in a map
- **Insert** operations require a **insert time** to be specified
- **Read** operations over **timeframes** often requires to **resolve conflicts**
- Vertices are **self-contained**, thus mobile
Pregel w/ Temporal Extensions

- Each vertex $v_n$ has its own processor and memory.
- Developers specify a compute function $c$ which
  - is repeatedly executed in parallel for all vertices in the graph
  - has access only to vertex local memory
  - can send messages to other vertices
  - is restricted to access data from timespan $t$ only
  - in which the vertex can vote to halt execution

$\text{mv} := \min(\text{mv}, \text{msgs})$
if $\text{mv} == v_n$ then
  voteToHalt()
else
  neighbours.send($\text{mv}$)
Extensions over Pregel

- **Temporal filtering**: in each iteration only data from a defined timespan is available.

- **Global memory**: all vertices can read and write a local copy of global memory; developers can define:
  - **conflict resolution** strategies for this global memory and
  - an **initialisation** function

- Write back, job-chaining, triggered execution

---

Pregel Computation Cycles

- Execution of a Pregel job is controlled by a central instance.

- The central instance repeatedly instructs all active vertices to execute \( c \); each of these cycles is called a superstep.

- A counter of active vertices is kept.

- If a vertex votes to halt in \( c \) it is marked inactive and omitted from computation in consecutive supersteps.

- Unless the vertex receives a message in which case it is again marked as active.

Prototype Architecture

- Provides an **execution environment** for temporal Pregel jobs
- Individual **worker nodes** manage **storage** and **processing** for multiple graph partitions (slots)
- Client application can use API to **manipulate** and **import data** and to **upload** and **execute code**

Bulk Synchronous Parallel Execution

- Executing jobs in DynamoGraph produce the shown processing patterns.

- Typically as more and more vertices become inactive, local computation and messaging phases become shorter.
Label Propagation Community Detection

• Originally published by (Raghavan et al. 2007) as an option for near linear time community detection $O(m)$

• In an iterative process each vertex $v$ in the graph observes its neighbours’ community labels

• The vertex $v$ gets assigned the community label most often found among its neighbours

• Algorithm allows for efficient distributed implementation in Pregel
Label Propagation Community Detection

\[
\begin{align*}
\text{while } s &< \text{MAX IT} \text{ and } g[lc] > 0 \\
&\text{execute } \text{compute} \text{ for each vertex} \\
&\text{compute}(v, msgs, g, t, s)
\end{align*}
\]

1. \(cl := \text{count\_max}(msgs)\)
2. \(\text{if } cl \text{ not set then} \)
3. \(\text{let } cl \text{ be the vertices name}\)
4. \(\text{else} \)
5. \(g[lc] := g[lc] + 1\)
6. \(\text{for each neighbour } n \text{ of } v \text{ do} \)
7. \(m(n, cl)\)

\(cl \ldots \text{community label}\)
\(g \ldots \text{global memory}\)
\(lc \ldots \text{labels changed}\)
Label Propagation Community Detection

(a) Initialized $t = 0$

(b) $t = 1$

(c) $t = 2$

(d) $t = 3$
Outlook: Dynamic Algorithms

- **Dynamic** graph **algorithms** allow to **update** computed **metrics** as the graph changes.

- With a temporal graph data structure, dynamic graph algorithms can be mimicked by using results from an **earlier timeframe** $t_n$ to compute the results for $t_{n+1}$.

- See the inserted ? vertex and the behaviour of the presented community detection.
Evaluation Setup

• **Private Cloud** installation @TK based on openstack

• During experiments the stack was reserved exclusively

• 2 compute vertexs with a total of: 48 Intel Xeon 3.47 Ghz CPUs, 288 GB RAM, and 2 TB local storage

• Linux, KVM hypervisor, memory ballooning, enterprise grade GBit switch
PageRank over Enron Dataset

- Email database from Enron scandal
- Available to researches as maildir database
- Executing PageRank in a timeframe from Jan 2002 - Dec 2003 results in **expected behaviour**:
  - More **processors** result in **faster execution**
  - Adding **workers** also adds **networking overhead** thus provides comparably inferior speedup

---

PageRank over Click Dataset

- **HTTP headers (clicks)** recorded by an edge-router Sept 2006 - May 2010

- Custom, anonymised, binary file format, ~3 TB gzip, ~13 TB uncompressed

- Preprocessed to a ~93 million edges graph that contains only top domains as vertices at monthly resolution

- **PageRank** computed in sliding window of six month

- **Processing reaches** a plateau, stalls observed
Discussion

- **Distributed processing** shows clear **advantage** in the described usage scenarios but significant **difference** in scalability between social and technical network becomes visible.

**technical net.**
Several vertices
degree edge degree
1000 - 100,000

**email**
max degree
~2000
Selected Case / Political Network

- Temporal graphs can be used to observe political networks
- Label propagation community detection used to determine whether a person belongs to coalition or to opposition
- Visualisation shows discussion in Austrian parliament automatically extracted from transcripts

Future Work

- **Visualisation** and **tooling** are in **prototypical state** and far from being used in production environments.

- Many spots for **performance improvements** identified: **vertex-cuts** over edge-cuts *(Gonzales et al. 2012)*, **decoupling profile data** from network data.

- **Dynamic notion of time** vs. fixed size time resolution in *DynamoGraph*

- Support **incremental processing** for **dynamic graph algorithms**
Conclusion / Contribution

• **Closes a gap** between the *theoretical work* found for *temporal graphs* and *applied computer science* concepts for *large-scale graph processing*.

• Resulted in an **Open Source prototype implementation** called *DynamoGraph* which has been shown to be generally technically feasible and scalable.

• **Provides a platform** for continued *application oriented research* in this area (see case studies).

• 7 peer reviewed and 2 invited publications.
DynamoGraph: Large-scale Temporal Graph Processing and its Application Scenarios
Matthias Steinbauer
31st of January, 2017

Matthias Steinbauer
matthias.steinbauer@jku.at
slides available at https://steinbauer.org/
List of Relevant Publications


conf.  Matthias Steinbauer and Gabriele Kotsis, **Platform for General-Purpose Distributed Data-Mining on Large Dynamic Graphs** presented at the 22nd Workshops on Enabling Technologies Infrastructure for Collaborating Enterprises, Hammamet, Tunisia, 2013.


conf.  Matthias Steinbauer and Gabriele Anderst-Kotsis, **Using DynamoGraph: Application Scenarios for Large-scale Temporal Graph Processing**, presented at the 17th International Conference on Information Integration and Web-based Applications & Services, Brussels, Belgium, 2015.

workshop  Matthias Steinbauer and Gabriele Anderst-Kotsis, **DynamoGraph: A Distributed System for Large-scale, Temporal Graph Processing, its Implementation and First Observations**, Temporal Web Analytics Workshop at WWW2016, Montréal, Canada, 2016.

journal  Matthias Steinbauer and Gabriele Anderst-Kotsis, **DynamoGraph: Extending the Pregel Paradigm for Large-scale Temporal Graph Processing**, International Journal on Grid and Utility Computing, Volume 7, No. 2, 2016.
