Temporal graph analytics on Apache Flink Stateful Functions

Speculative Red Hat Collaboratory Project

Vasiliki (Vasia) Kalavri
vkalavri@bu.edu
https://sites.bu.edu/casp/
Modeling the world as a graph

Social networks

Transportation networks

Actor-movie networks

The web
If you like “Inside job” you might also like “The Bourne Identity”

What’s the cheapest way to reach Zurich from London through Berlin?

These are the top-10 relevant results for the search term “graph”
• Which services are reachable from service “User Timeline”? 

https://github.com/delimitrou/DeathStarBench/tree/master/socialNetwork
• Which services are reachable from service “User Timeline”?
• Which services are **frequently** on the critical path?
• Which services are reachable from service “User Timeline”?
• Which services are frequently on the critical path?
• Which request paths contributed to increased latency around the time when a SLA violation occurred?

https://github.com/delimitrou/DeathStarBench/tree/master/socialNetwork
Which services are reachable from service “User Timeline”?  
Which services are frequently on the critical path?  
Which request paths contributed to increased latency around the time when a SLA violation occurred?  
Are there any disconnected services right now?
• Which services are reachable from service “User Timeline”?
• Which services are frequently on the critical path?
• Which request paths contributed to increased latency around the time when a SLA violation occurred?
• Are there any disconnected services right now?

Temporal graph queries

https://github.com/delimitrou/DeathStarBench/tree/master/socialNetwork
Modeling temporal graphs with edge events

A purchase, a movie rating, a like on an online post, a bitcoin transaction, a message sent from one service to another
Modeling temporal graphs with edge events

A purchase, a movie rating, a like on an online post, a bitcoin transaction, a message sent from one service to another
Challenging requirements for existing systems

<table>
<thead>
<tr>
<th>Queries</th>
<th>Access patterns</th>
<th>Workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Point-in-time</td>
<td>• Full-graph</td>
<td>• Iterative analytics</td>
</tr>
<tr>
<td>• Continuous</td>
<td>• Neighborhood-local</td>
<td>• Path traversals</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Frequent updates</td>
</tr>
</tbody>
</table>
Challenging requirements for existing systems

**Queries**
- Point-in-time
- Continuous

**Access patterns**
- Full-graph
- Neighborhood-local

**Workloads**
- Iterative analytics
- Path traversals
- Frequent updates

Graph DBs (e.g. Neo4j)
Challenging requirements for existing systems

**Queries**
- Point-in-time
- Continuous

**Access patterns**
- Full-graph
- Neighborhood-local

**Workloads**
- Iterative analytics
- Path traversals
- Frequent updates

Graph DBs (e.g. Neo4j)
Batch graph processing systems (e.g. Apache Giraph)
Challenging requirements for existing systems

Queries
- Point-in-time
- Continuous

Access patterns
- Full-graph
- Neighborhood-local

Workloads
- Iterative analytics
- Path traversals
- Frequent updates

Graph DBs (e.g. Neo4j)

Batch graph processing systems (e.g. Apache Giraph)

Data stream processing systems (e.g. Apache Flink)
Bridging the gap: Temporal graph analytics on Flink Stateful Functions

- Actor-like programming model
  - Built-in support for streams
  - Flexibility in expressing iterative, path, and search queries

- Decoupling of computation and state
  - Support for concurrent graph updates and queries
  - Ingestion and analytics can scale independently

- Flexible deployment
  - Embedded compute for low latency
  - Serverless remote compute for scalability
Statefun as a dataflow

Ingress source

Router flatmap

Dispatcher operator

Feedback operator

keyBy(address)

Fn1
Fn2
Fn3
Fn4

keyBy(address)

Local feedback
The function dispatcher operator

1. Extract function type
2. Load function
3. Load state
4. Invoke function
5. Update state
6. Output message
Deployment options: Embedded

- Functions are run in the JVM and are directly invoked with the messages and state access.
- Very performant but updates to functions require updating the Flink cluster.
Deployment options: Co-located

- A Flink TaskManager interacts with one Function process sitting “next to it”.
  - Deploy pods consisting of a Flink container and the function side-car container; the two communicate via the pod-local network.
  - Supports different languages but it cannot scale the state and compute parts independently.
Deployment options: Remote

- Physical \textit{separation} - logical \textit{co-location}.
- The state/messaging tier (i.e., the Flink processes), and the function tier are deployed, managed, and scaled independently.
Prototype system architecture

Ingress

Graph stream (Kafka)

Query stream (Kafka)

Egress

Flink Cluster

Partition 1

Partition 2

Partition k

Results stream

VertexStorageFn

QueryHandlerFn

IterCoordinatorFn

ConnectivityFn

ShortestPathsFn

GNNInferenceFn

PageRankFn

Extract query-specific state

Temporal graph view

Convergence?
Design considerations & challenges

- How to express queries as interacting functions?
- How to represent graph state on RocksDB?
  - Implications on concurrency and latency
- How to represent temporal neighborhoods?
  - Query vs. update time
- When to compute embedded vs. remotely?
Representing temporal graphs in a KV store

- AFSF *co-partitions* messages and function state
- To ensure consistency, only one *function invocation per key* can be active at any point in time
Representing temporal graphs in a KV store

- (+) single-lookup access to neighboring subgraph
- (-) concurrency limited by #partitions
Representing temporal graphs in a KV store

- (+) maximum concurrency
- (-) multi-lookup access to neighboring subgraph
Storing a temporal neighborhoods in RocksDB

`{(4,4), (3,6)}`
Storing a temporal neighborhoods in RocksDB

srcID

(destID, time)

{(4,4), (3,6)}
Storing a temporal neighborhoods in RocksDB

Deserialize neighborhood on lookup
Serialize neighborhood on update
Storing a temporal neighborhoods in RocksDB

Query and update only the relevant bin(s)
Read latency (100 connectivity queries)

CDF

128 interval bins

Adjacency list

- TM_32
- iTM_128
- LL
- PQ
- P99
- P90

ns
Update latency (100 connectivity queries)

CDF

128 interval bins

Adjacency list

ns
Ongoing and next steps

- Extending the library with more algorithms
- Evaluating embedded vs. remote function invocation
- Large-scale experiments on realistic scenarios
  - Micro-service traces collected from Jaeger
- Preparing a first open-source release
Project Team

**PhD students**
- Emmanouil Kritharakis
- Sonia Horchidan

**Master students**
- Vivek Unnikrishnan
- Shekhar Sharma
- Sihan Chen

**Faculty**
- Paris Carbone
- Vasiliki Kalavri
Temporal graph analytics on Apache Flink Stateful Functions

Speculative Red Hat Collaboratory Project

Vasiliki (Vasia) Kalavri
vkalavri@bu.edu
https://sites.bu.edu/casp/