Distributed Graph Processing:

Theoretical TLAV vs. Pregel

PETAR JOVANOVIC

Introduction

Distributed graph processing

Thinking Like a Vertex (TLAV) framework

- iterative execution of a user-defined vertex program over vertices of the graph
    - Vertices pass messages to adjacent vertices   **Send message to just adjacent vertex**
- Two views: **vertices view** and **edges view**
- Synchronous (BSP)
    - Computation is based on **supersteps**, which serve as sync barriers

Pregel – most famous implementation of BSP TLAV framework

- Besides vertices and edges, it also maintains a triplets view

Attribute of origin and destination node
triple view avoid useless things,,,optimization

Theoretical TLAV

Apply: also known as vertex program, applies a user-defined **function** to each vertex in parallel; meaning that the function specifies the behavior of a single vertex v at a particular superstep S. On the first iteration, the vertex program is invoked on all vertices and the pre-defined message is passed. On subsequent iterations, the vertex program is only invoked on those vertices that receive messages.

Scatter: also known as send message, sends messages to other vertices, such that those vertices will receive the messages in the next superstep S+1.

Gather: receives and reads messages that are sent to a node v from the previous superstep S – 1 and apply the function. This function must be commutative and associative.

Pregel

Pregel is essentially a message-passing interface constrained to the edges of a graph

Implementation of the Apply / Gather / Scatter

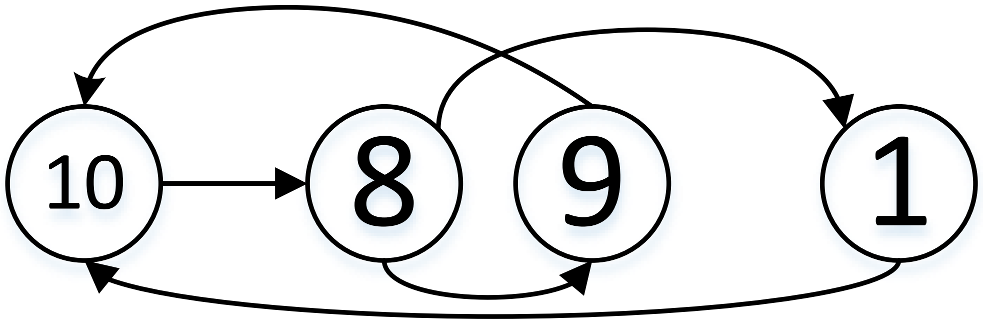
- **VProg** (corresponding to the **Apply** phase): in the case of the first superstep, the vertex value, otherwise applies the **function** over the vertex value and the received message and sends the result.
- **sendMsg** (corresponding to the **Scatter** phase): by accessing the **triplets view*** *checks if it the current vertex value can **change** the destination vertex value, and if so, it sends the current value to it. Otherwise, does not send anything.*

  This is the function that is used to generate messages to be used in the next step of the iteration.
- **merge** (corresponding to the **Gather** phase): receives messages from previous supersteps and applies the **function**.

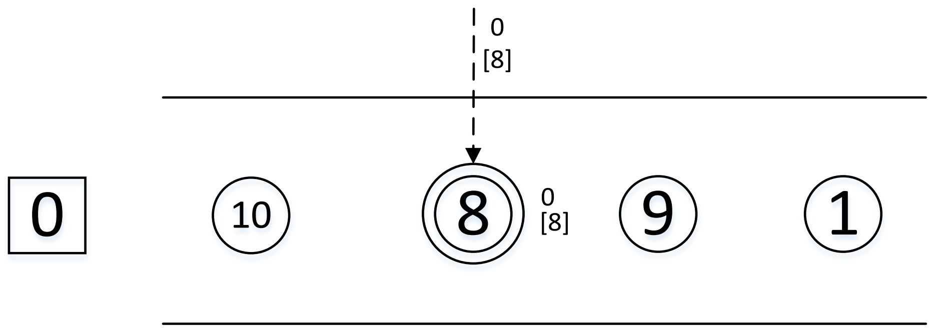
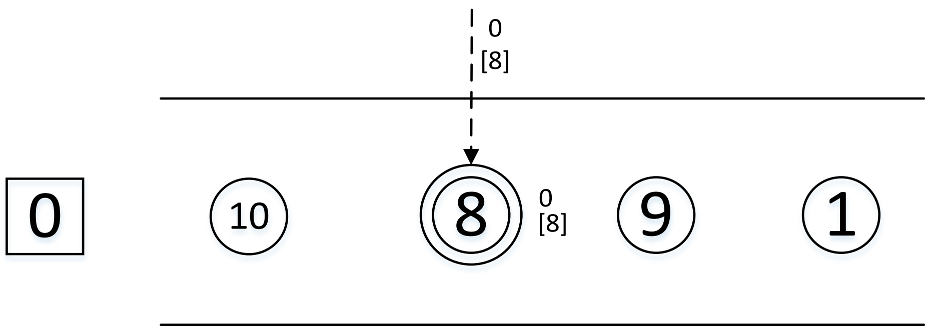
  function to pick one message out of a group of messages sent to the same node (vertex).

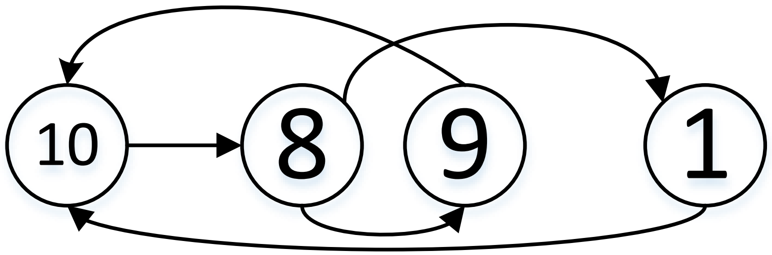
*** Triplets view:** in addition to the vertex and edge views of the property graph, GraphX also exposes a **triplet view**. The triplet view joins the vertex and edge properties yielding an RDD that contains instances of the EdgeTriplet class. EdgeTriplet class extends the Edge class by adding the **srcAttr** and **dstAttr** members which contain the **source** and **destination** properties, respectively.

Example: Maximal value

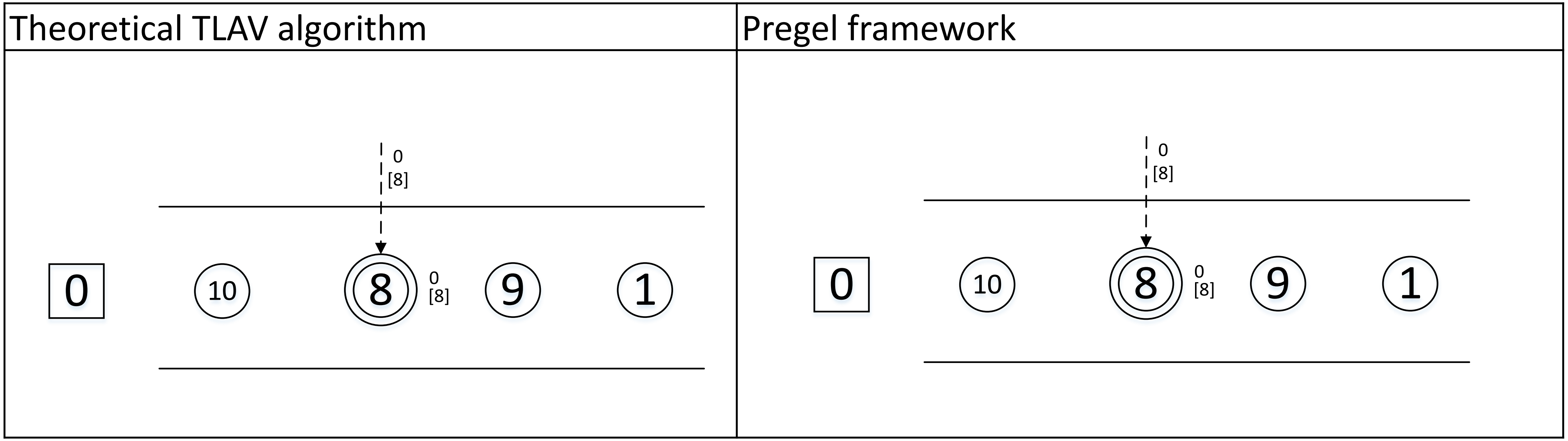


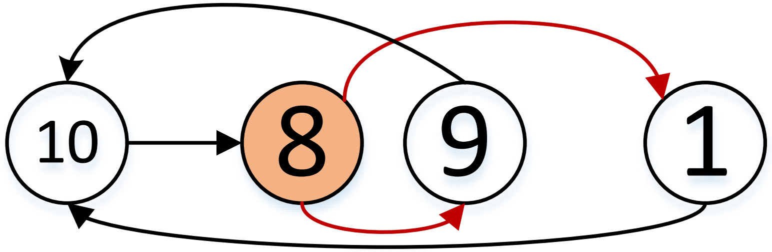
Example: Maximal value

Theoretical TLAV algorithm	Pregel framework
	

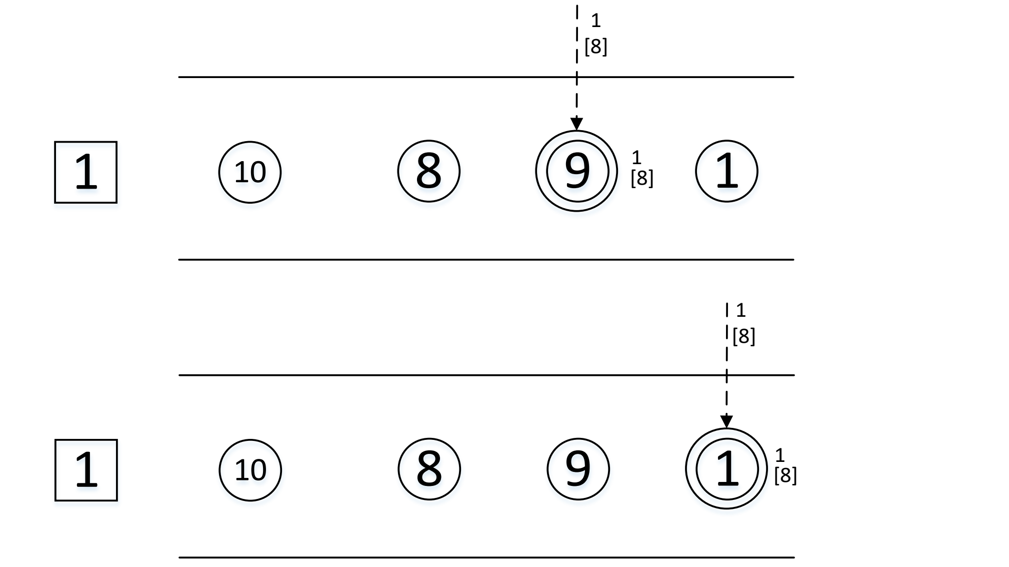
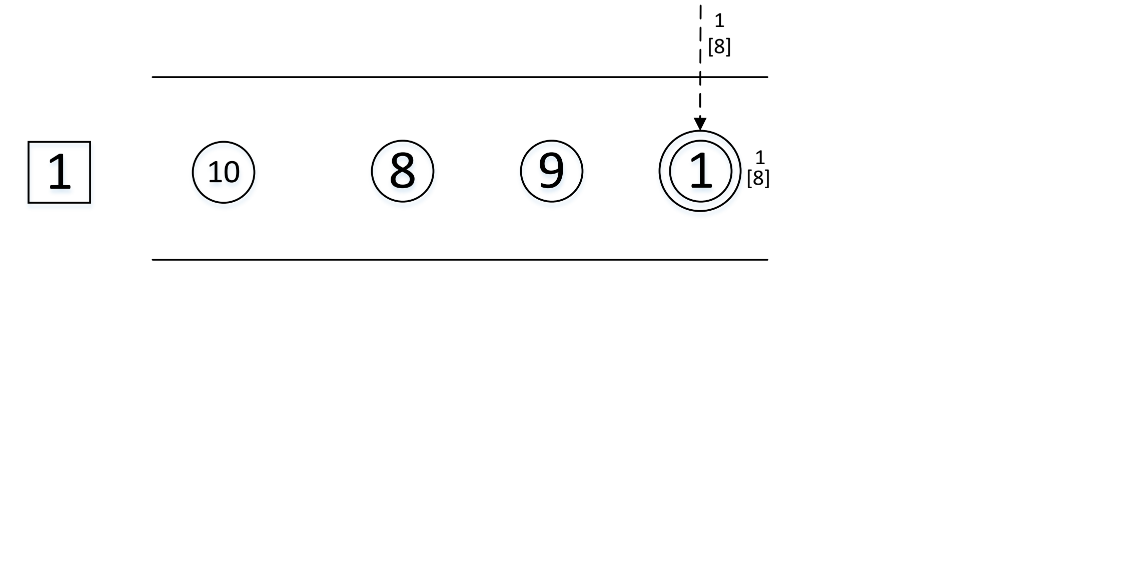


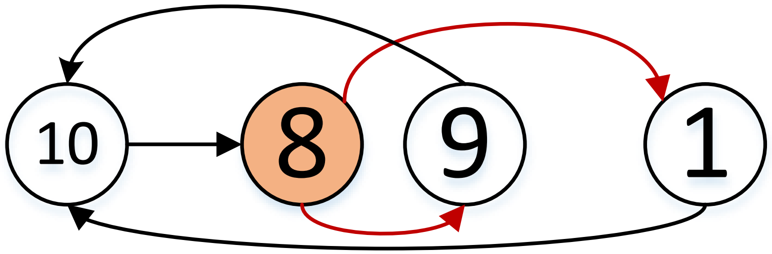
Example: Maximal value



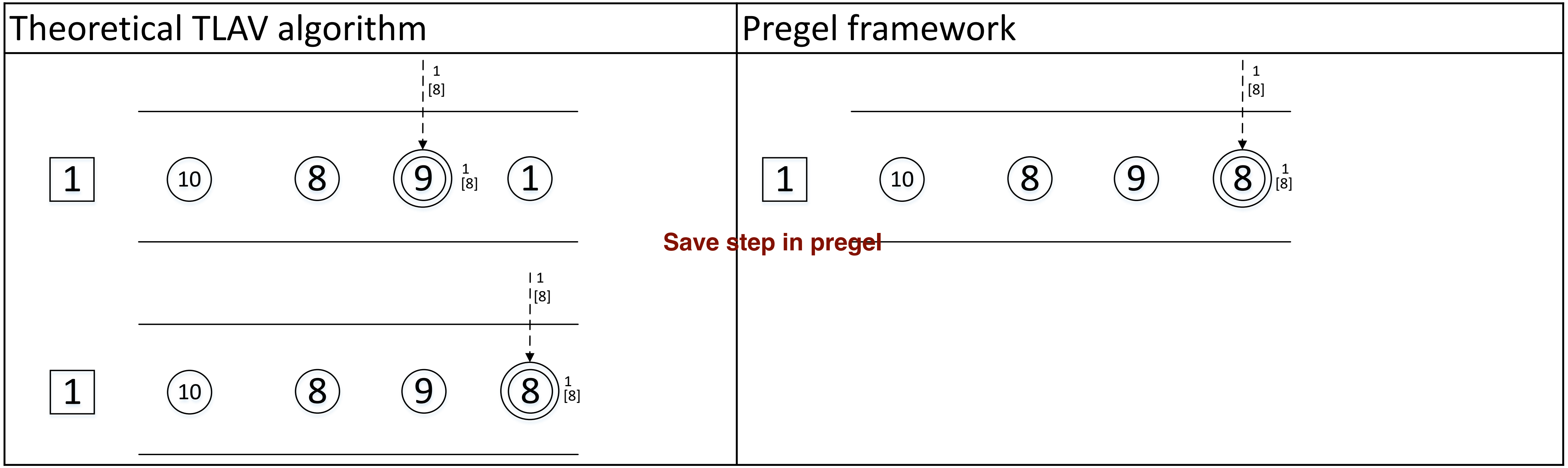


Example: Maximal value

Theoretical TLAV algorithm	Pregel framework
	



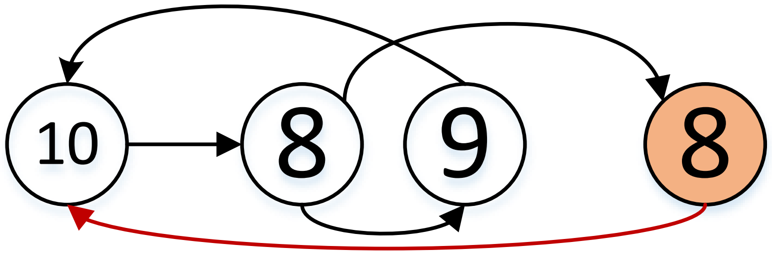
Example: Maximal value



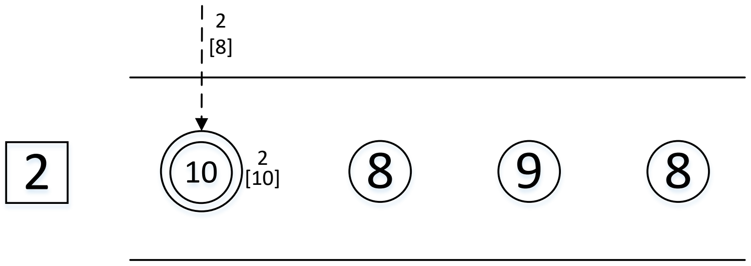
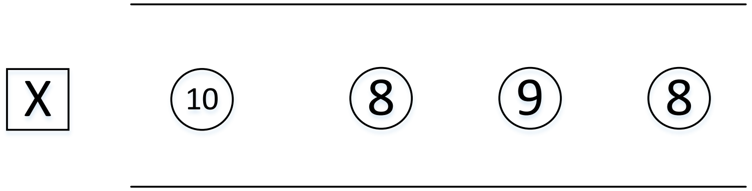
Theoretical TLAV algorithm

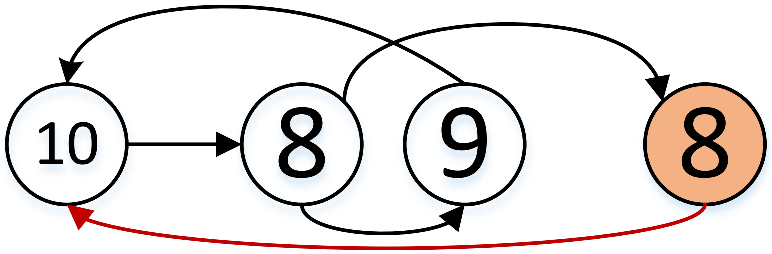
Pregel framework

Save step in pregel



Example: Maximal value

Theoretical TLAV algorithm	Pregel framework
	



Conclusions

Pregel and GraphX implement the theoretical TLAV framework by maintaining the three views:

- **Vertices**, **edges**, **triplets** (source & destination attributes).

+ Triplets view saves us from unnecessary message sending by allowing vertices to check the destination value.

- Maintenance of the triplets view has additional overhead, especially in the case of evolving graph topologies.

Thank you!

Questions?