PREGEL: A SYSTEM FOR LARGE-SCALE GRAPH PROCESSING.

Authors: Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, L., Leiser, N., Czjkowski, G.

Speaker: Chong Li

Department: Applied Health Science

Program: Master of Health Informatics

OUTLINE

- Term explanation
- Motivation & Introduction
- Computation Model
- System Implementation
- Experiment
- Conclusion & Future Work
- Application

TERM EXPLANATION

- Graph Database: a storage system that uses graph representations for data where each node represents an entity with unique id, type and properties.

- Superstep: iteration that is used for graph algorithm in Pregel . It can be viewed as sort of a barrier for parallel-y executing entities.

MOTIVATION & INTRODUCTION

--- WHO ARE THEY??

MOTIVATION & INTRODUCTION

Yes?

Daddies?

Larry Page& Sergey Brin, 2 geniuses brought a surprise to this world in 1998:

-Google

MOTIVATION & INTRODUCTION

-- 70 offices in more than 40 countries
-- Products include search tools, security tools, map-related products, etc.
-- More and more information is collected and stored in geographically different offices.

MOTIVATION & INTRODUCTION

- 80% of google distributed computation is based on MapReduce (Google Map, Google Translate, etc).
- --can take advantage of locality of data, processing it on or near the storage assets in order to reduce the distance over which it must be transmitted

MOTIVATION & INTRODUCTION

Challenges faced by MapReduce:

- Many practical computing problems concern large-scale graphs- such as shortest path. MapReduce, however :

  - A lot of I/O due to passing the entire state of the graph from one stage to the next.

  - Too many iterations are needed for parallel graph processing

MOTIVATION & INTRODUCTION

- Need for a scalable distributed solution with features of :

--Scalable and Fault-tolerant platform

--API with flexibility to express arbitrary graph algorithm

--Vertex centric computation (Think like a vertex) –pg.14

MOTIVATION & INTRODUCTION

- Need for a scalable distributed solution with features of :

--Scalable and Fault-tolerant platform

--API with flexibility to express arbitrary algorithm

--Vertex centric computation (Think like a vertex)

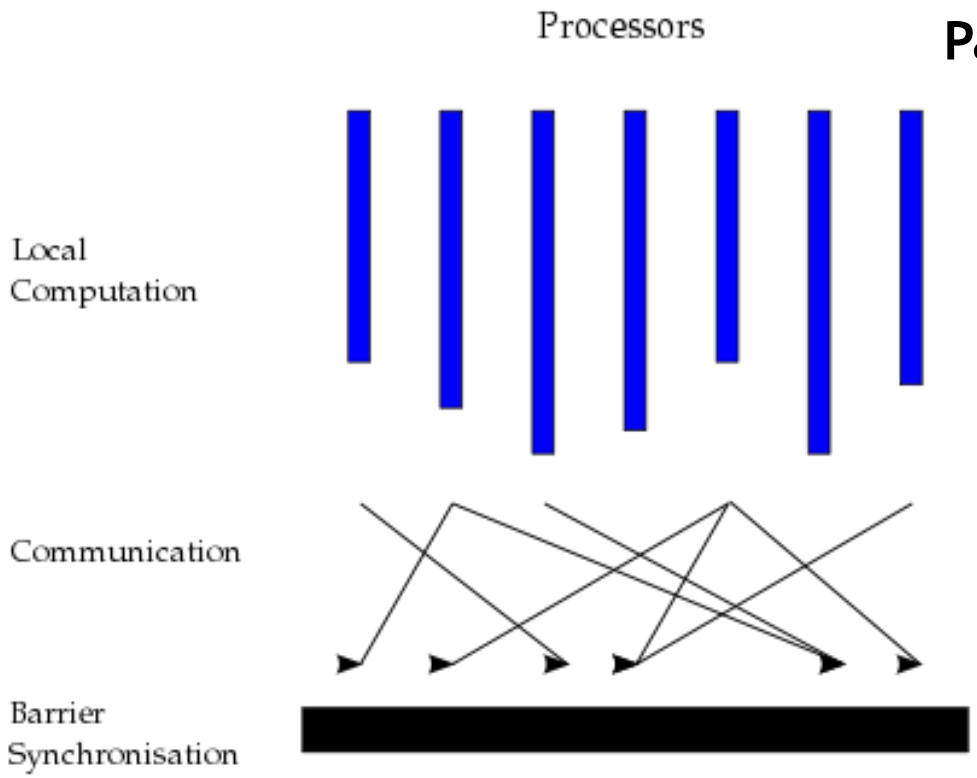
INTRODUCTION- PREGEL

- Pregel is a system for large-scale graph processing. It provides a fault-tolerant framework for the execution of graph algorithms in parallel over many machines.
- Pregel model retains worker state (the same worker is responsible for the same set of nodes) across iteration, the graph can be loaded in memory once and reuse across iterations.
- Pregel only sends local computed result over the network, which implies the minimal bandwidth consumption.

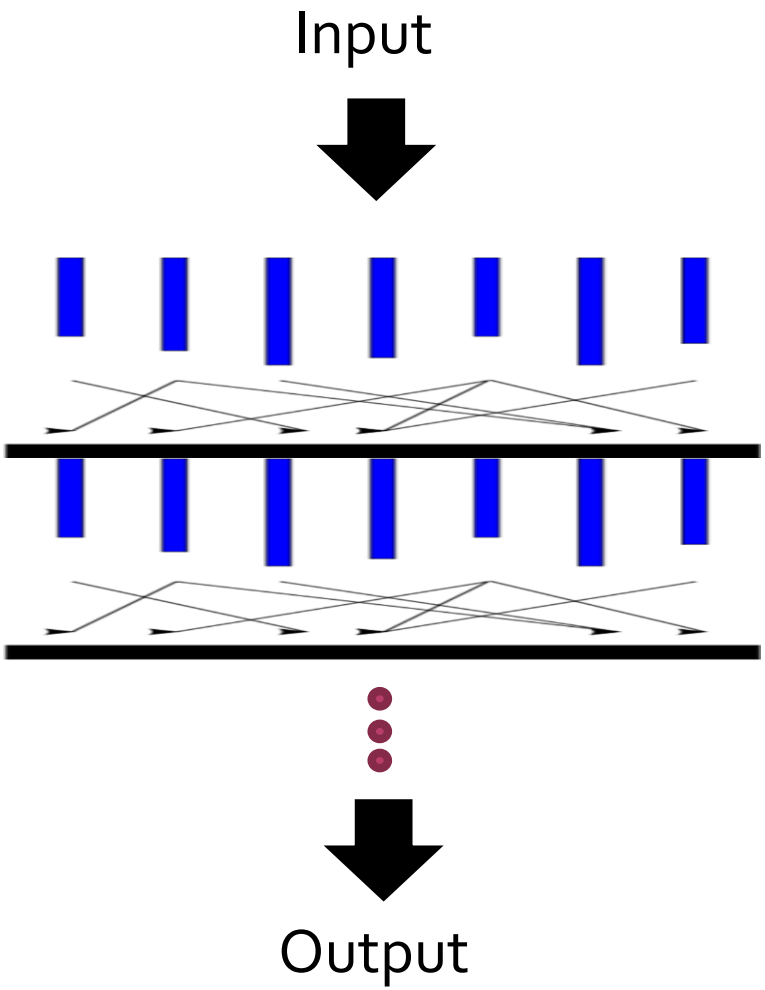
Note: Pregel is not a database because no key-value store or any new means of storing is used in this Google product.

COMPUTATION MODEL

Bulk Synchronic Parallel model (BSP)



COMPUTATION MODEL



Supersteps
(a sequence of iterations)

COMPUTATION MODEL

- In Superstep: the vertices compute in parallel
  - Each vertex
    - Receives messages sent in the previous superstep
    - Executes the same user-defined function
    - Modifies its value or values of its outgoing edges
    - Sends messages to other vertices (to be received in the next superstep)
    - Mutates the topology of the graph
    - Votes to halt if it has no further work to do

--Vertex centric computation

COMPUTATION MODEL

Vertex State Machine

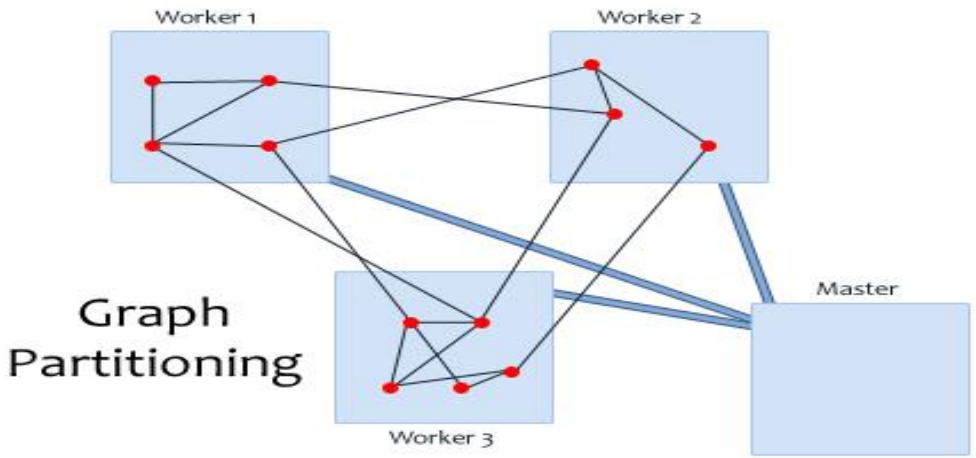
- Termination condition
  - All vertices are simultaneously inactive
  - There are no messages in transit

SYSTEM IMPLEMENTATION

- Pregel system also uses the master/worker model
  - Master
    - Maintains worker
    - Recovers faults of workers
    - Provides Web-UI monitoring tool of job progress
  - Worker
    - Processes its task
    - Communicates with the other workers
- Persistent data is stored as files on a distributed storage system (such as GFS or BigTable)
- Temporary data is stored on local disk

SYSTEM IMPLEMENTATION

1. Many copies of the program begin executing on a cluster of machines
2. Master partitions the graph and assigns one or more partitions to each worker
3. Master also assigns a partition of the input to each worker
   - Each worker loads the vertices and marks them as active



SYSTEM IMPLEMENTATION

4. The master instructs each worker to perform a superstep
   - Each worker loops through its active vertices & computes for each vertex
   - Messages are sent asynchronously, but are delivered before the end of the superstep

   Note: This step is repeated as long as any vertices are active, or any message is in transit

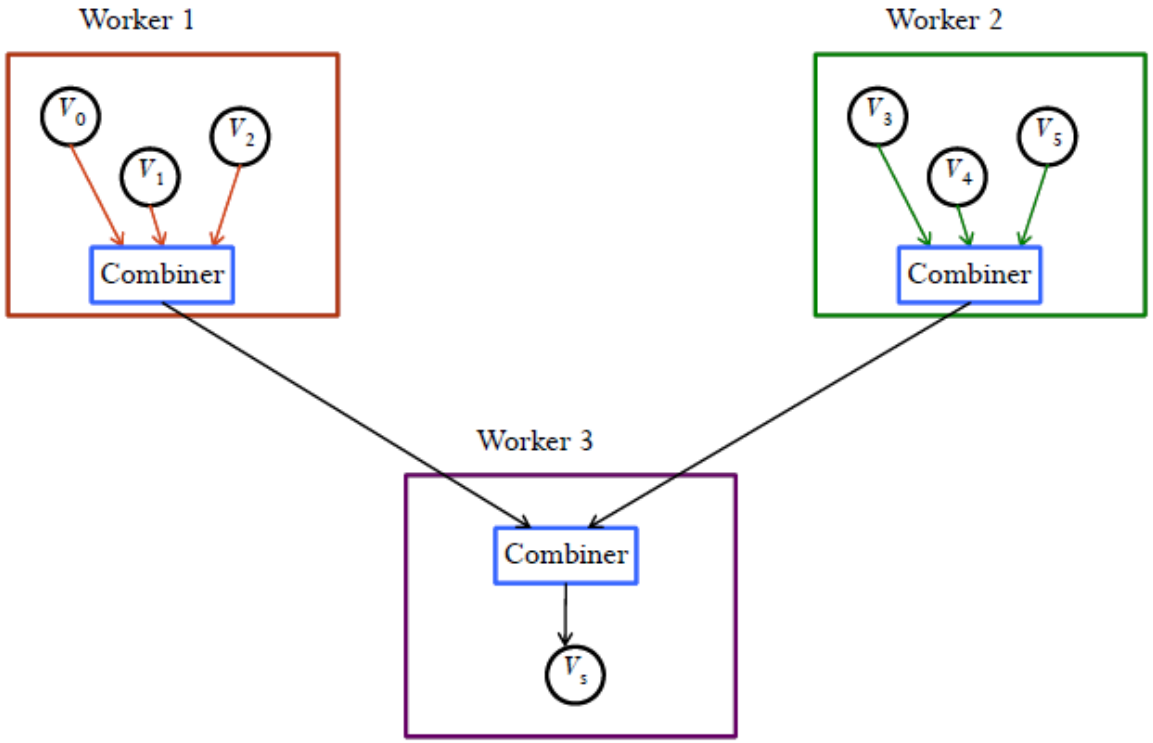
5. After the computation halts, the master may instruct each worker to save its portion of the graph

SYSTEM IMPLEMENTATION- FAULT TOLERANCE

- Checkpointing
  - The master periodically instructs the workers to save the state of their partitions to persistent storage system
    - e.g., Vertex values, edge values, incoming messages
- Failure detection
  - Using regular “ping” messages
- Recovery
  - The master reassigns graph partitions to the currently available workers
  - The workers all reload their partition state from most recent available checkpoint

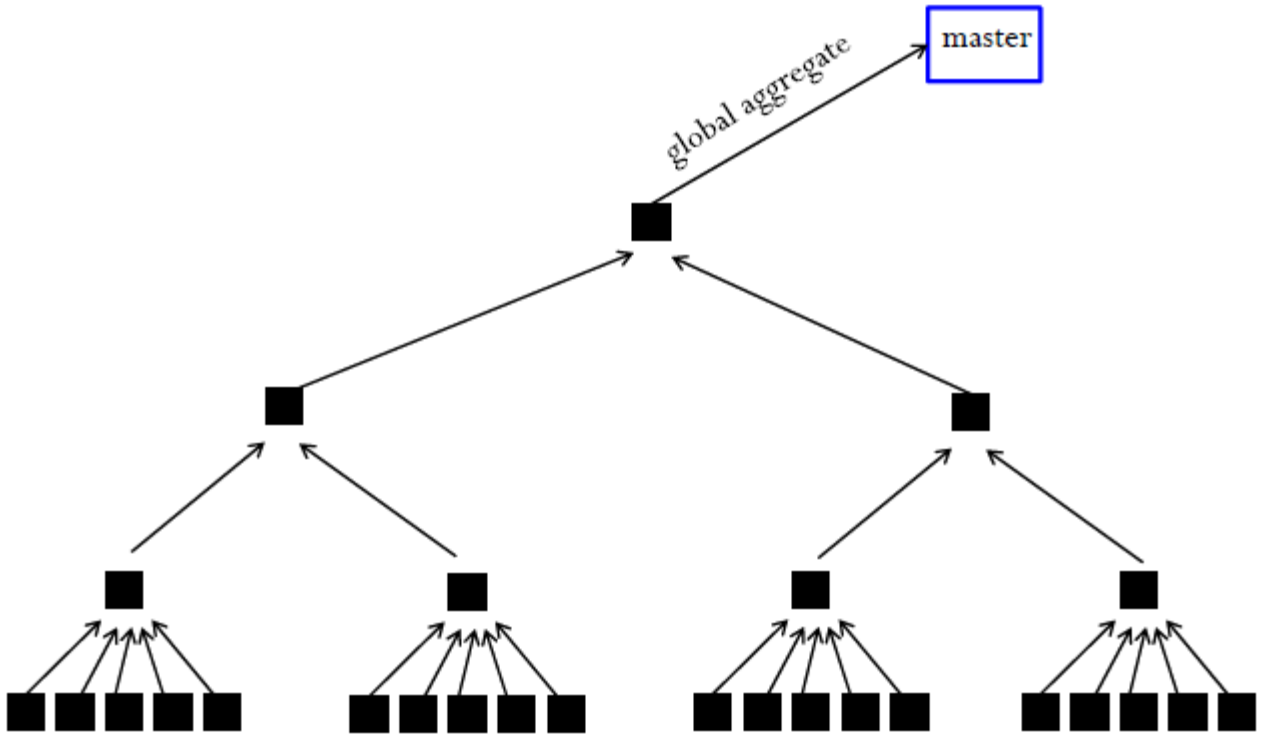
SYSTEM IMPLEMENTATION-COMBINER (NOT ENABLED DEFAULT, API)

- Worker can combine messages reported by its vertices and send out one single message
- Reduce message traffic and disk space

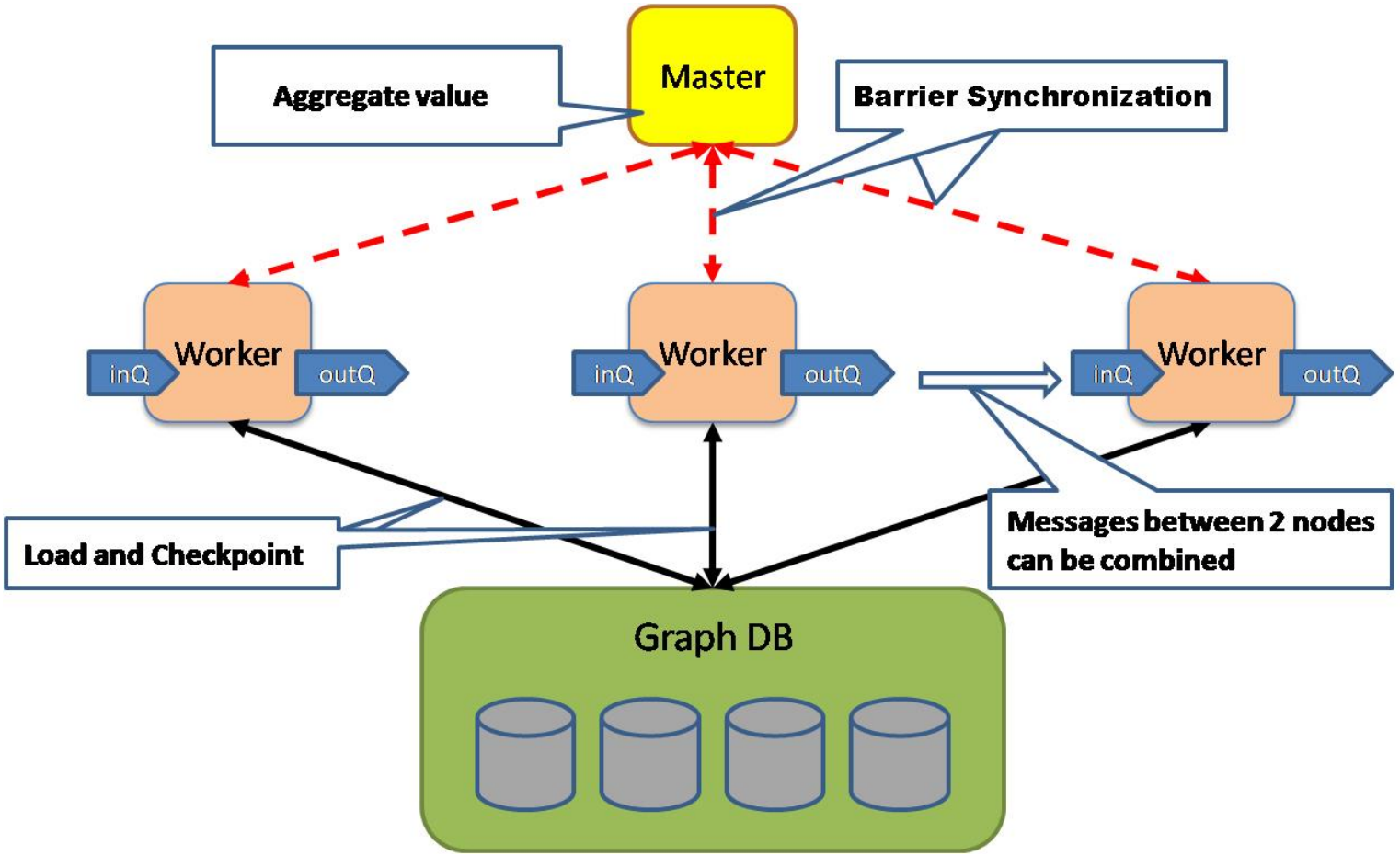


SYSTEM IMPLEMENTATION-AGGREGATOR (DEFAULT, API)

- Used for global communication, global data and monitoring



SYSTEM IMPLEMENTATION



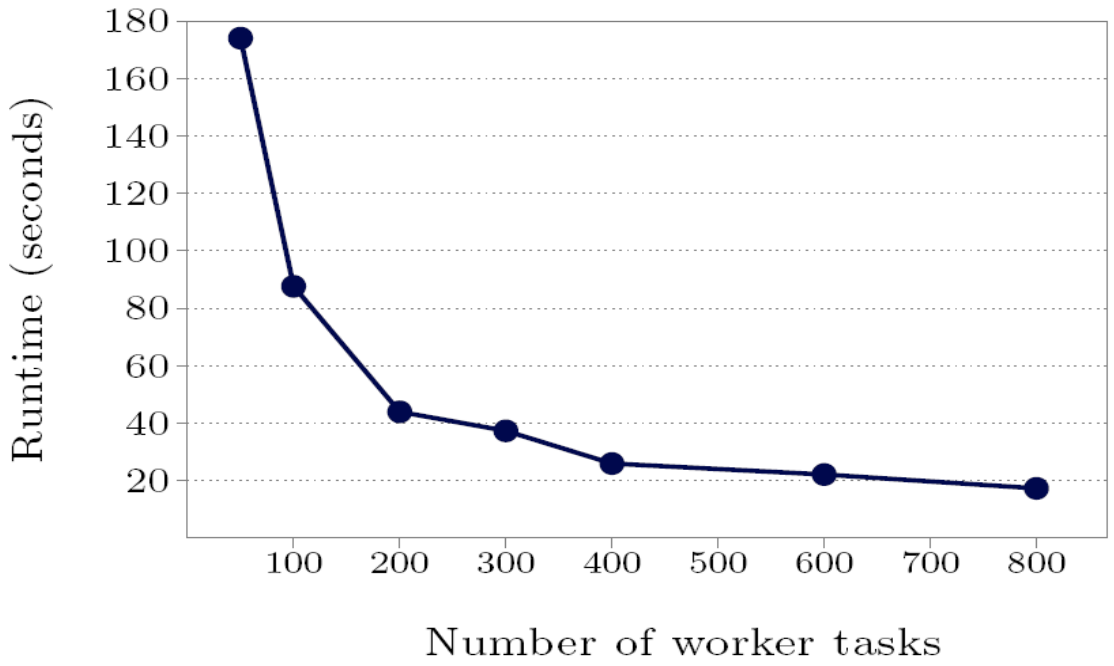
EXPERIMENT

- Environment
  - H/W: A cluster of 300 multicore commodity PCs
  - Data: binary trees, log-normal random graphs (general graphs)

- Naïve SSSP implementation (single-source shortest path)
  - The weight of all edges = 1
  - No checkpointing- because of short runtime

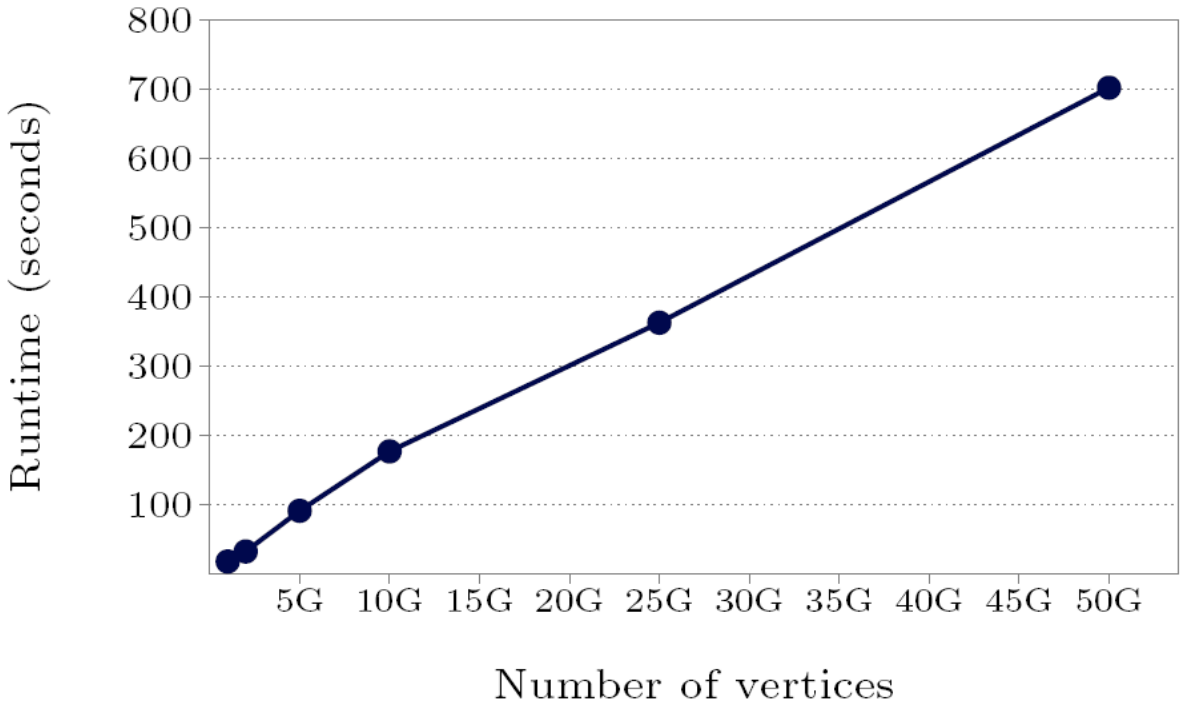
EXPERIMENT

- SSSP – 1 billion vertex binary tree: varying # of worker tasks



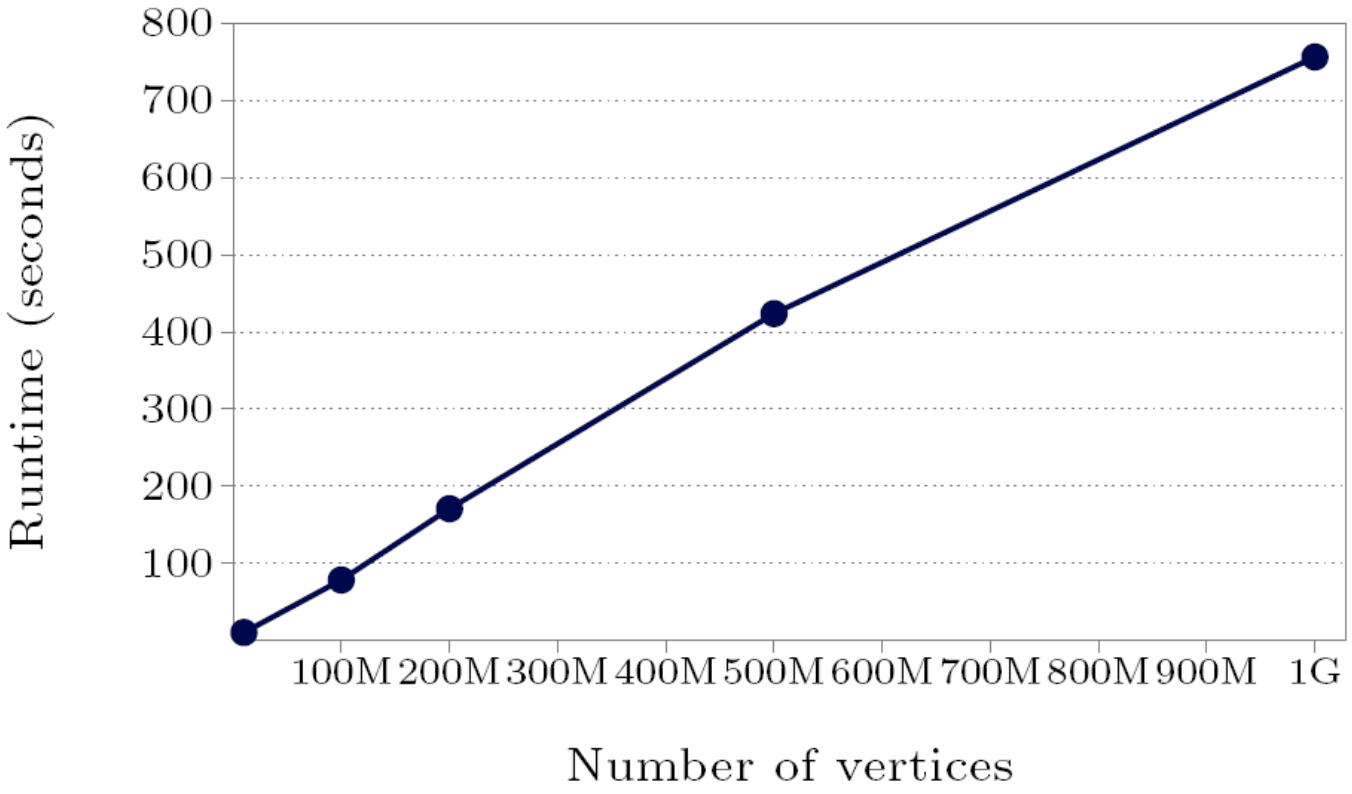
EXPERIMENT

- SSSP – binary trees: varying graph sizes on 800 worker tasks



EXPERIMENT

- SSSP – Random graphs: varying graph sizes on 800 worker tasks



CONCLUSION & FUTURE WORK

- Pregel is a scalable and fault-tolerant platform with an API that is sufficiently flexible to express arbitrary graph algorithms

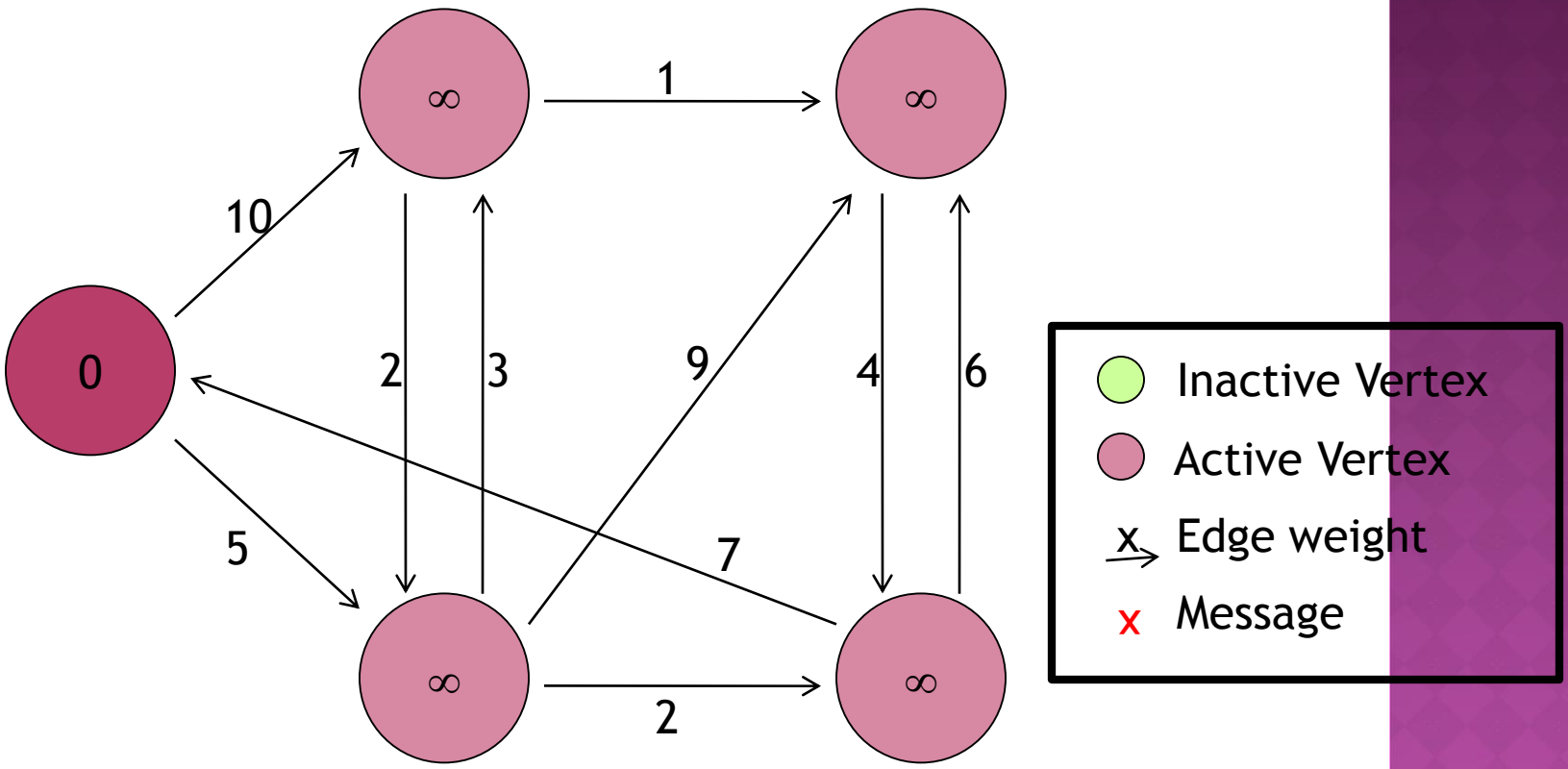
- Future work
  - Relaxing the synchronicity of the model
    - Not to wait for slower workers at inter-superstep barriers
  - Assigning vertices to machines to minimize inter-machine communication
  - Caring dense graphs in which most vertices send messages to most other vertices

AN APPLICATION IN PREGEL

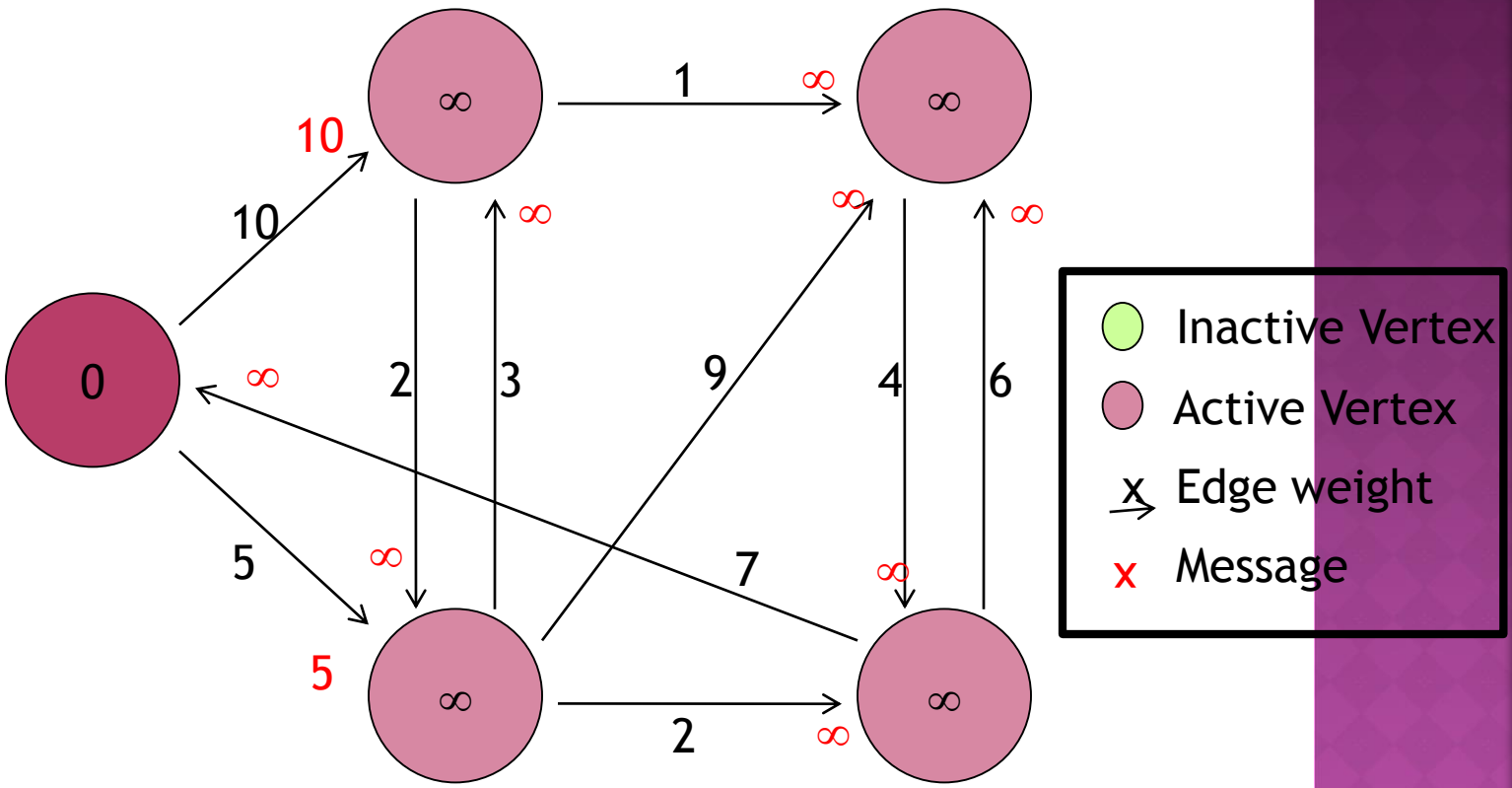
- Single Source Shortest Path
  - Find shortest path from a source node to all target nodes

```
class ShortestPathVertex
    : public Vertex<int, int, int> {
  void Compute(MessageIterator* msgs) {
    int mindist = IsSource(vertex_id()) ? 0 : INF;
    for (; !msgs->Done(); msgs->Next())
      mindist = min(mindist, msgs->Value());
    if (mindist < GetValue()) {
      *MutableValue() = mindist;
      OutEdgeIterator iter = GetOutEdgeIterator();
      for (; !iter.Done(); iter.Next())
        SendMessageTo(iter.Target(),
                      mindist + iter.GetValue());
    }
    VoteToHalt();
  }
};
```

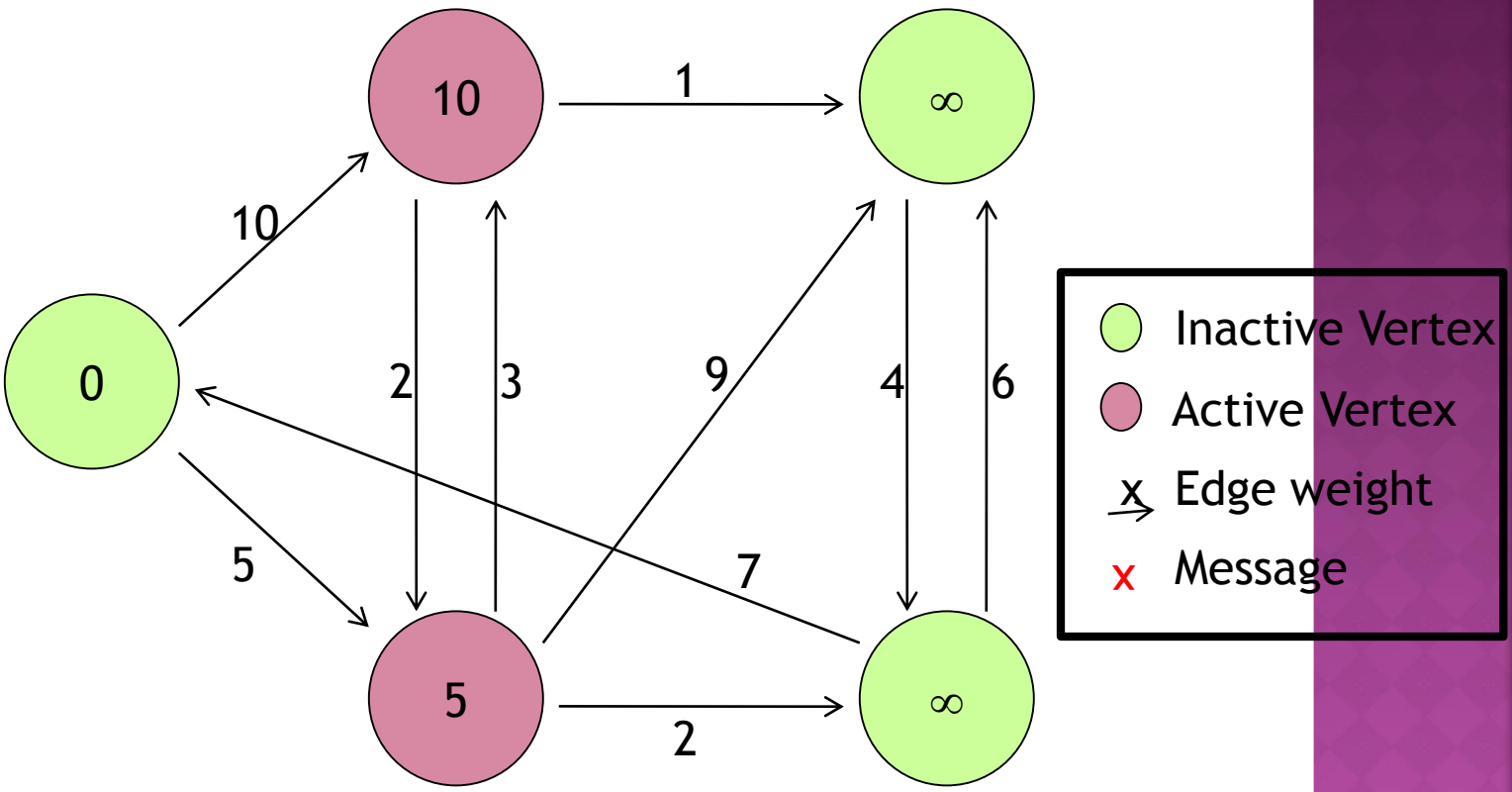
EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



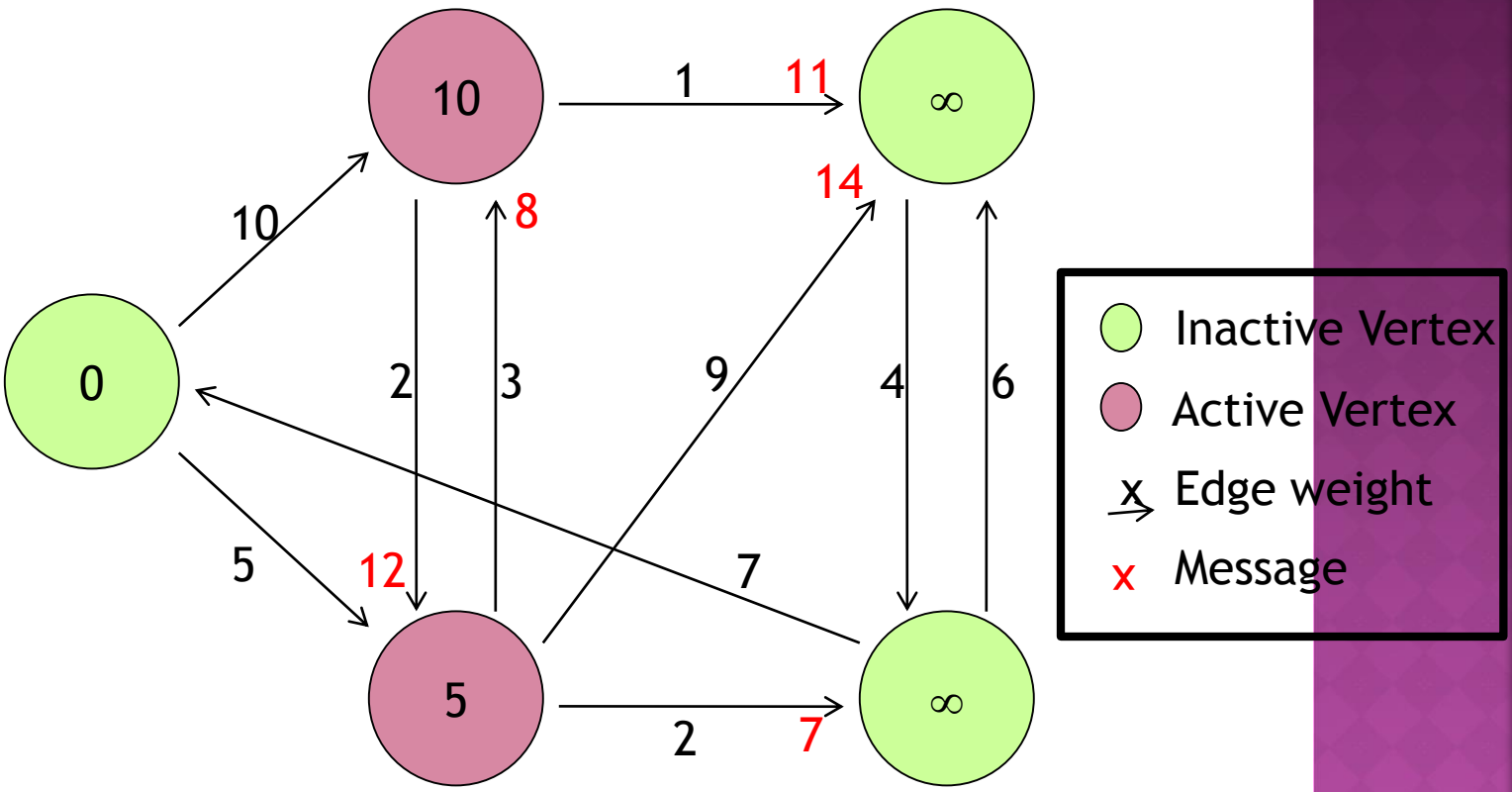
EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



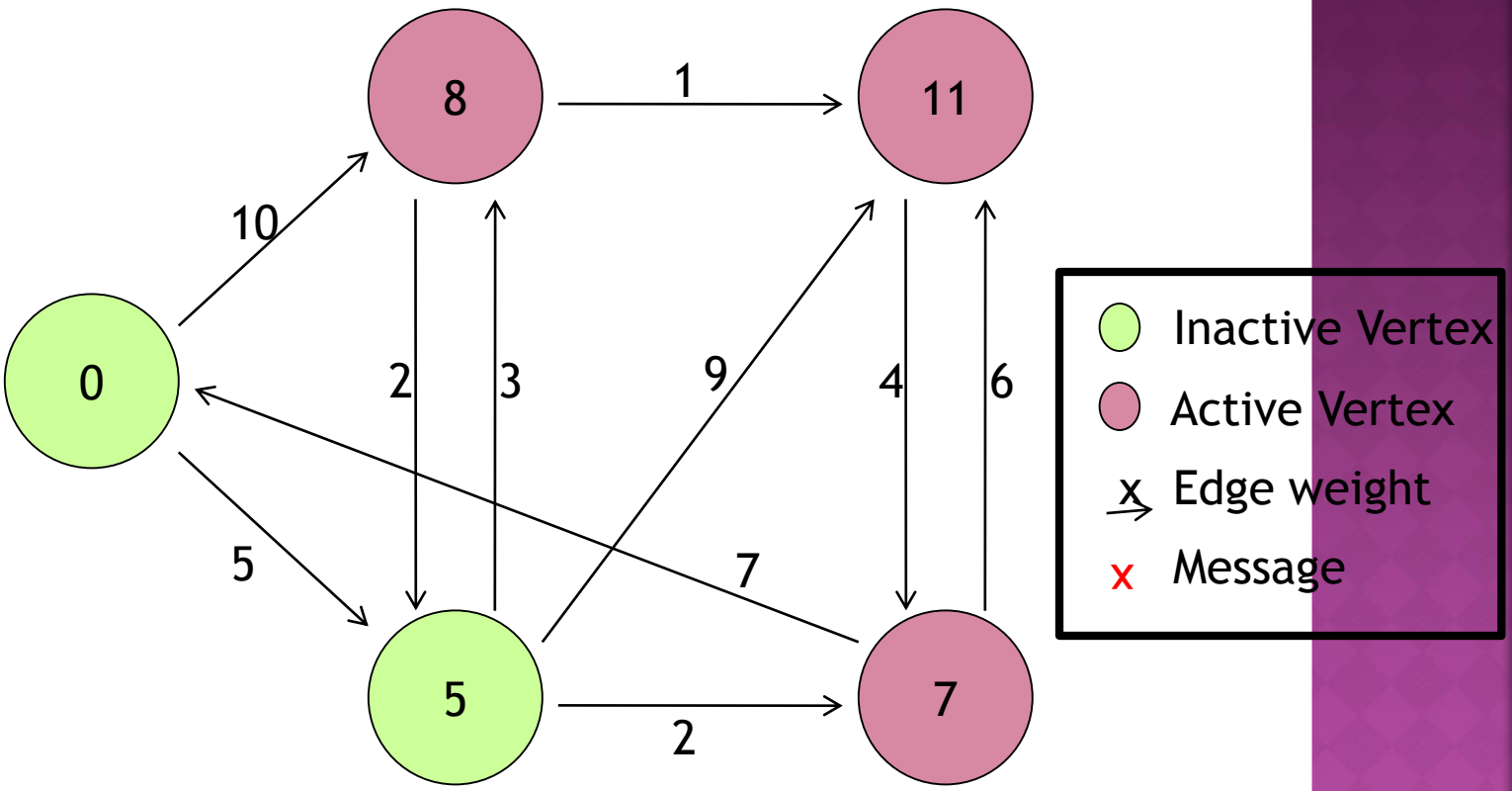
EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



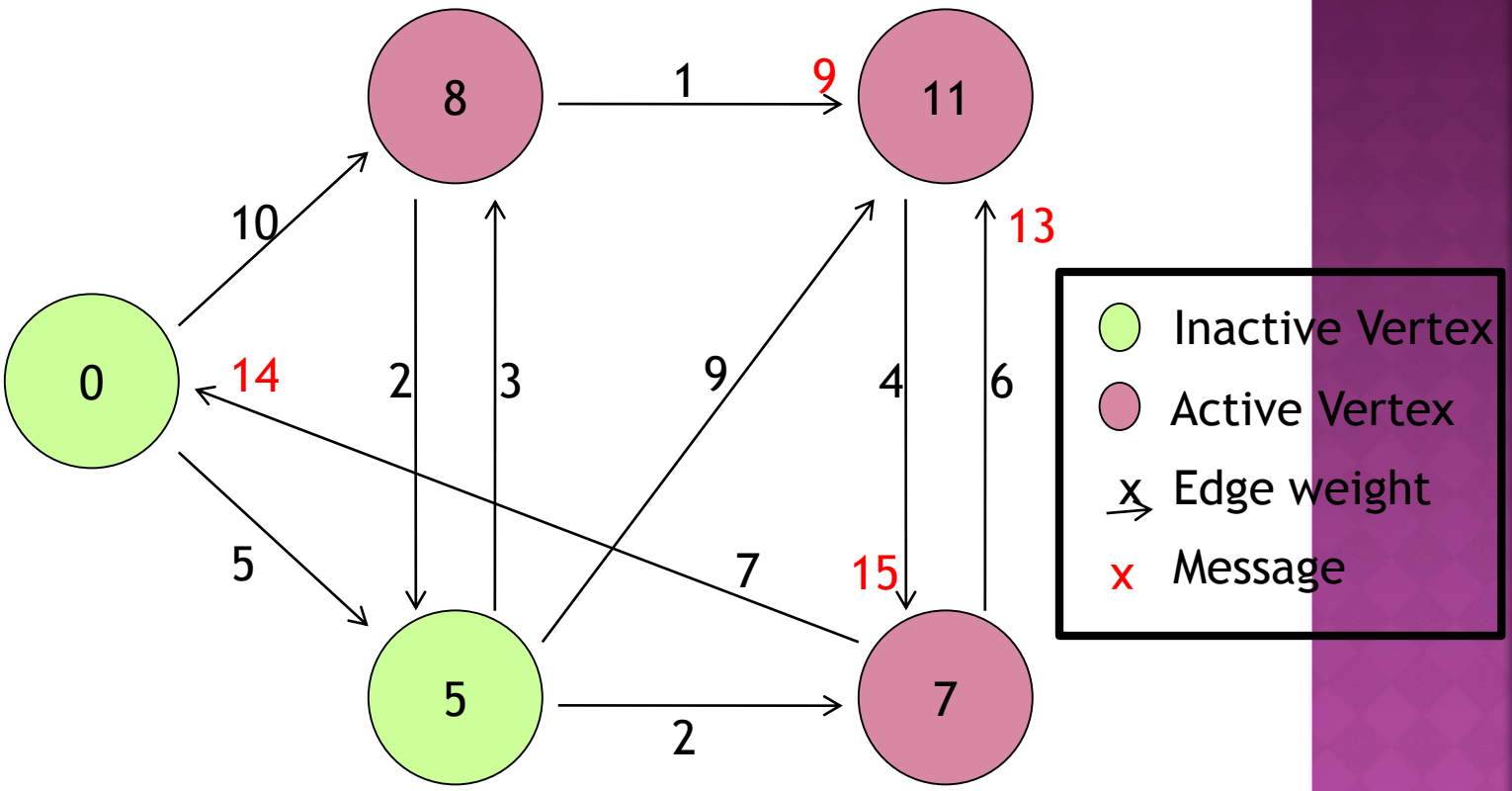
EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



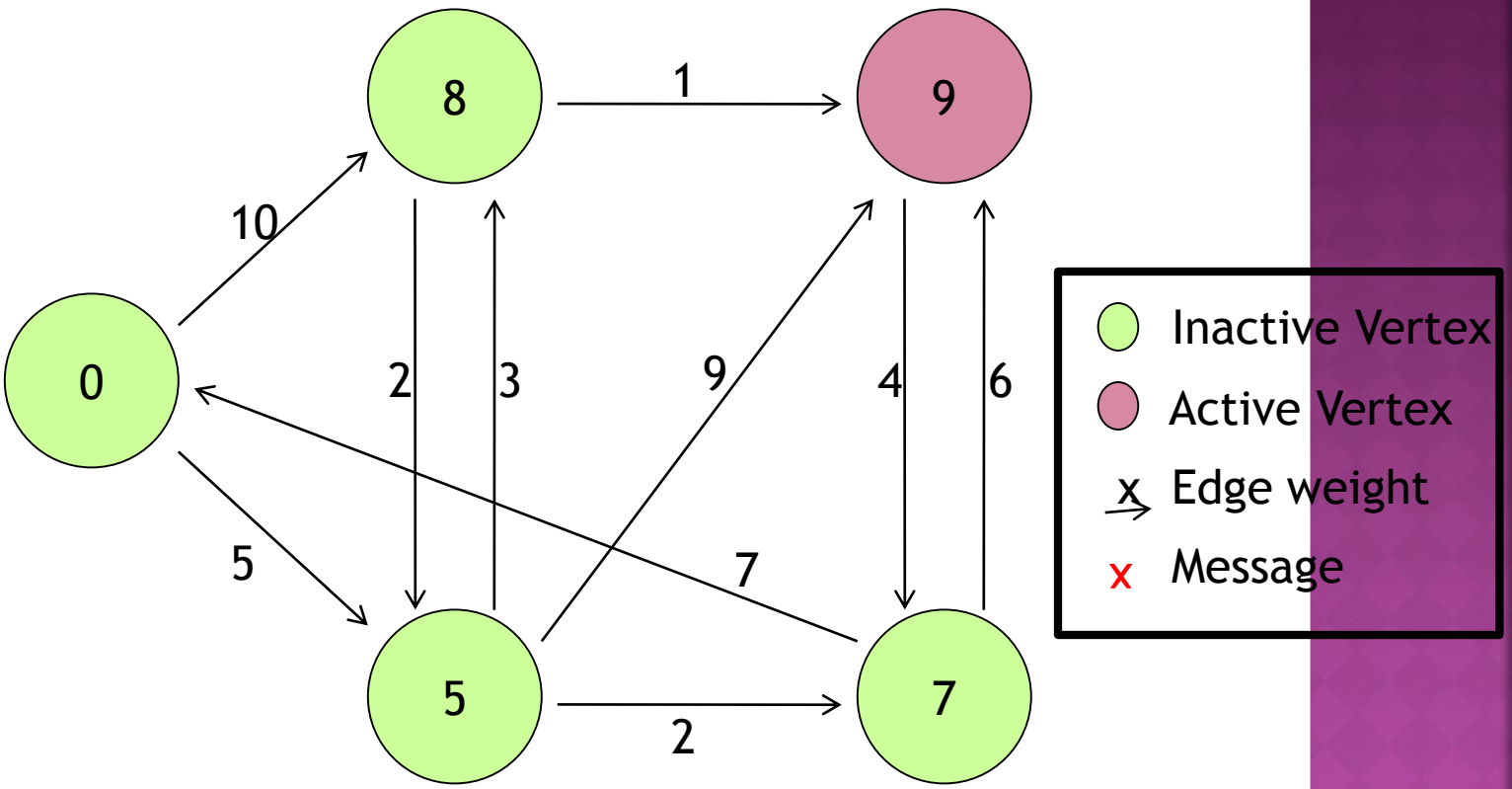
EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



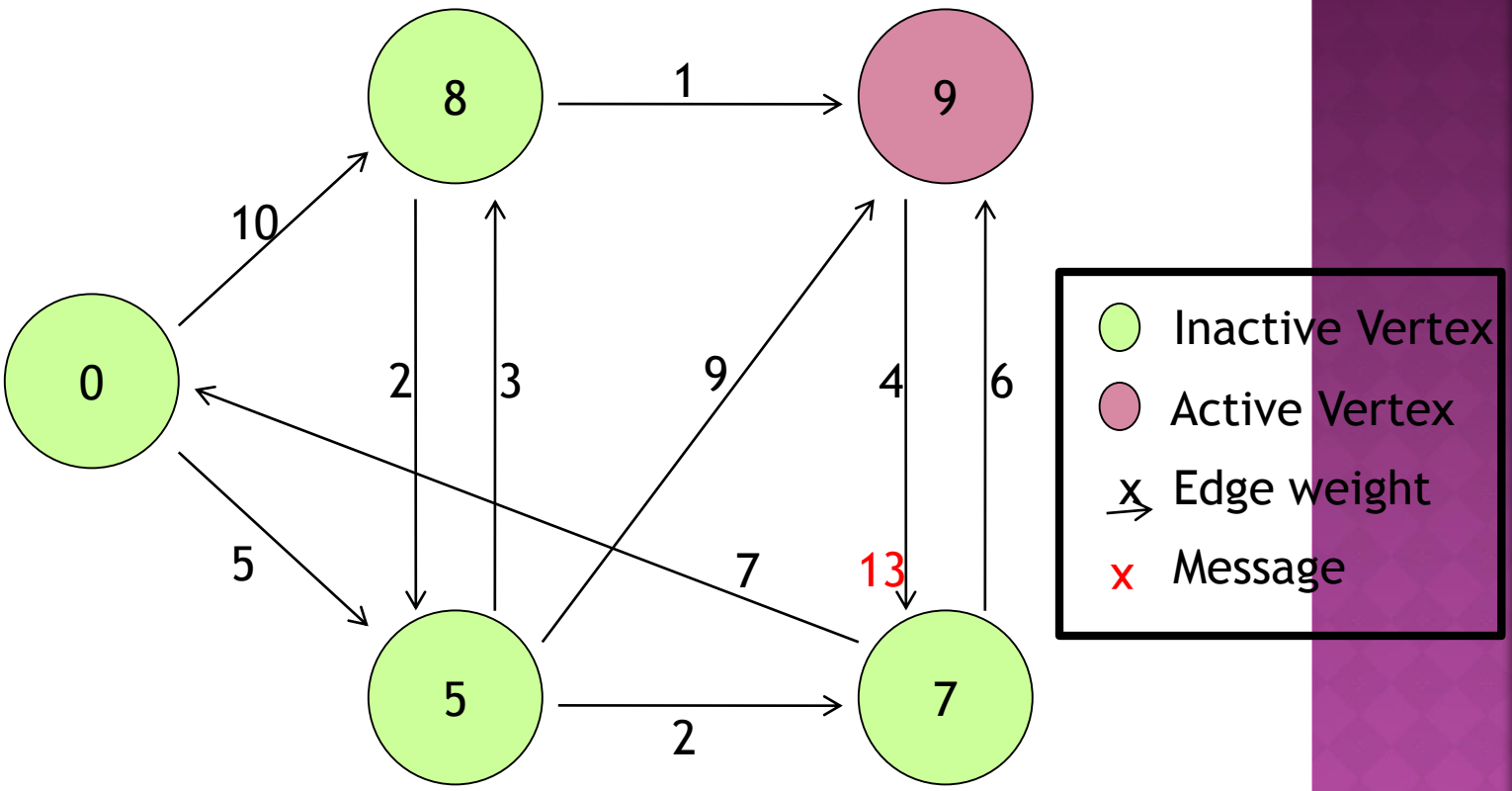
EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



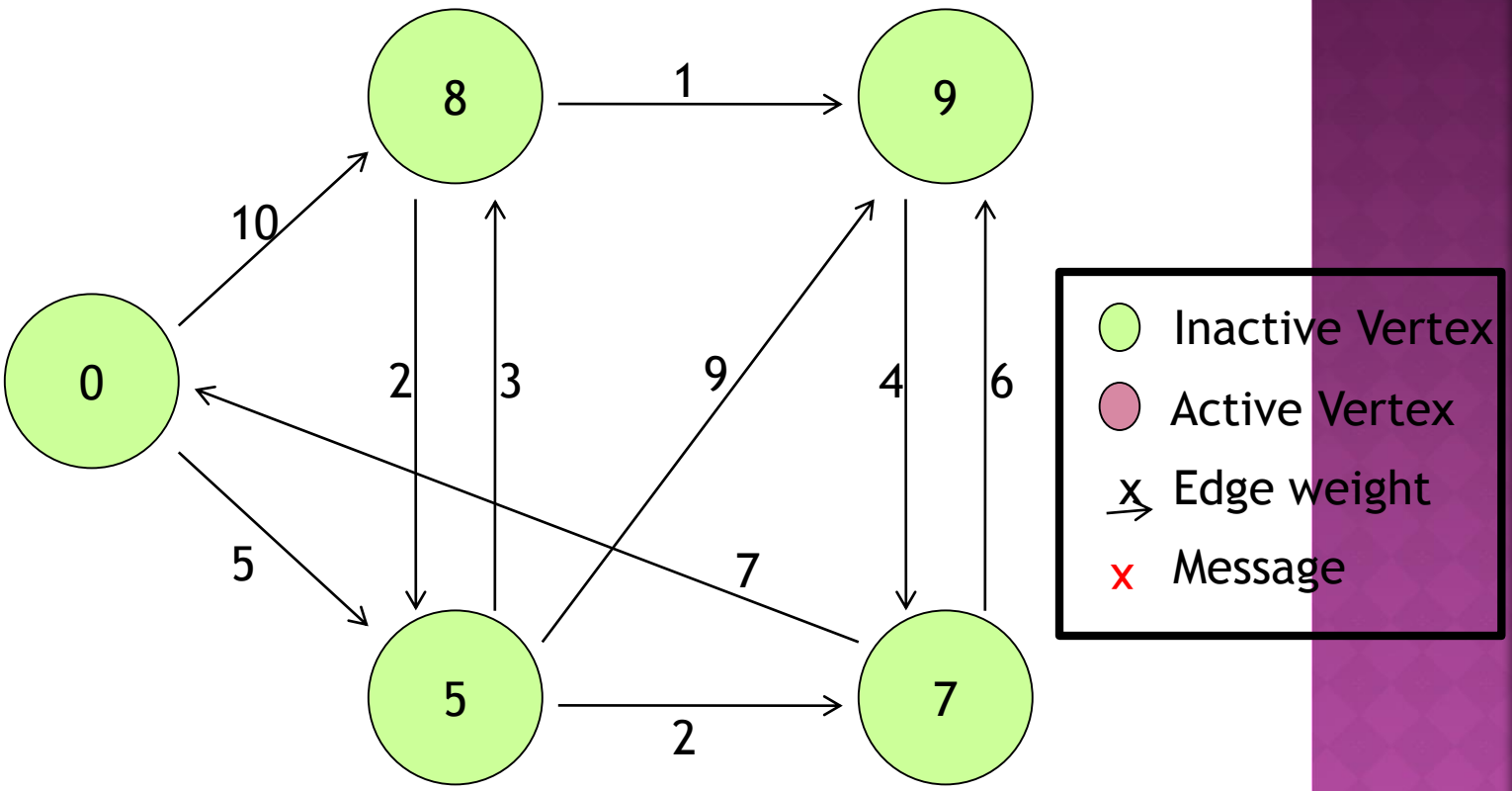
EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



EXAMPLE: SSSP – PARALLEL BFS IN PREGEL



THANK YOU!

--Any question?