CLOUD COMPUTING APPLICATIONS

Pregel - Part 1

Roy Campbell & Reza Farivar

Introduction

- Infrastructure for graph processing – expensive to design
- Can we use MapReduce?
  - Inefficient because the graph state must be stored at each stage of the graph algorithm, and each computational stage will produce much communication between stages
- Single computer and library approach – not scalable
- Use existing shared memory parallel graph algorithms – no fault-tolerance

Vertex-Oriented

- Based on BSP model
- Provides directed graph to Pregel
- Runs your computation at each vertex (processor)
- Repeats until every computation at each vertex votes to halt
- Pregel returns directed graph as a result

Primitives

- Vertices – first class
- Edges – not first class
- Both vertices can be created and destroyed

Pregel Organized via C++ API

- Supersteps S
- Application code subclasses Vertex, writes a Compute method
- Can get/set Vertex value
- Can get/set outgoing edges values
- Can send/receive messages
- Reads messages sent to V in superstep S-1. Sends messages to other vertices that will be received at superstep S+1; modifies state of V and its outgoing edges

C++ API

- Message passing
- No guaranteed message delivery order
- Messages delivered exactly once
- Can send a message to any node
- If destination doesn’t exist, user’s function is called