UCL
Université catholique de Louvain

INGI2145 Cloud Computing

Lab 5: Graph processing using Apache Giraph

Muhammad Bilal

Problems with Hadoop MapReduce

- Iterative algorithms are cumbersome and inefficient
- Intermediate results need to persist on HDFS
- Each iteration is separate MR job (scheduled separately)

Apache Giraph

Apache Giraph

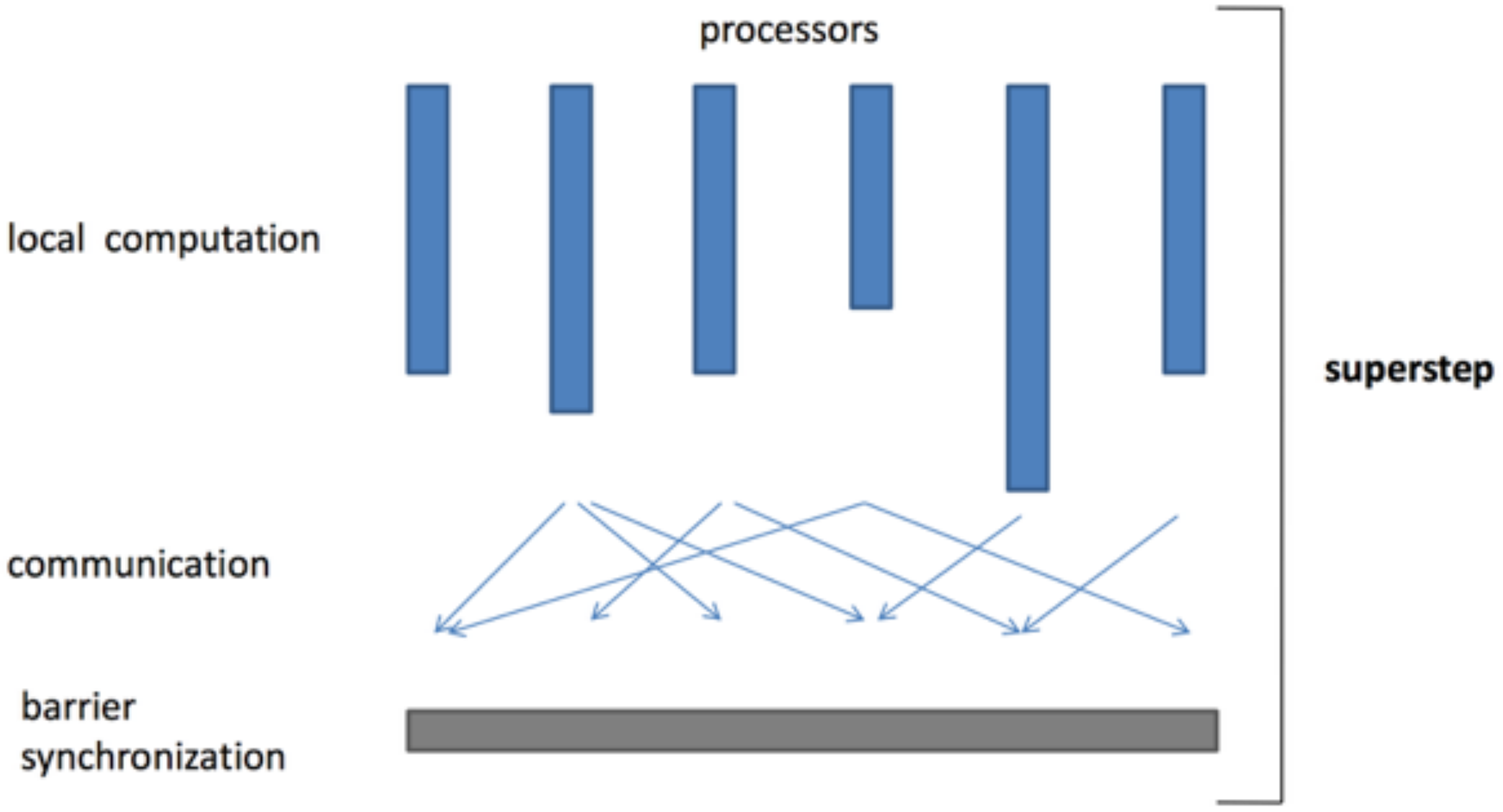
- Scalable, fault-tolerant implementation of graph-processing algorithms on Apache Hadoop
- Initially inspired by Google's Pregel framework
- Vertex-centric approach
- Bulk Synchronous Parallelism programming model
- Replaces map reduce with a single mapper only job
- Data partition done once

https://www.safaribooksonline.com/blog/2014/02/10/intro-apache-giraph/

Apache Giraph

- Giraph algorithms consists of iterative execution of “super-steps”
- Super steps consists of:
  - Message reception
  - Aggregation
  - Vertex/Edge property update
  - Message transmission
- Synchronization phase

Apache Giraph



http://researcher.ibm.com/researcher/files/us-heq/Large%20Scale%20Graph%20Processing%20with%20Apache%20Giraph.pdf

Apache Giraph

- Giraph the Vertex Object. Vertex<I,V,E,M>
  - I is Vertex ID type
  - V is Vertex Value type
  - E is Edge Value type
  - M is Message data type

Example: N Hop Neighbor count

```
// Initialization in superstep 0
if (getSuperstep() == 0) {
   vertex.setValue(new VertexDataStructure(0l,new HashSet<Long>()));
   for (Edge<LongWritable, FloatWritable> edge : vertex.getEdges()) {
     Message m = new Message(vertex.getId().get(),3,0l);
     sendMessage(edge.getTargetVertexId(), m);
     LOG.debug("Sending message "+ m + " to " + edge.getTargetVertexId());
   }
  }
```

Example: N Hop Neighbor count

```
// Process messages and change state
Long nodeCount = vertex.getValue().vertexValue.get();
 for (Message message : messages) {
   if(message.srcId.equals(vertex.getId())){
       LOG.debug("Reply received" + message);
       if(!received.contains(message.senderId.get())){
           received.add(message.senderId.get());
           LOG.debug(received + " " +vertex.getId() );
           nodeCount++;
       }
   }else if(message.hopCount.get()>0){
       sendMessage(message.srcId,new Message(message.srcId.get(),
               message.hopCount.get()-1,vertex.getId().get()));
        for (Edge<LongWritable, FloatWritable> edge : vertex.getEdges()) {
              sendMessage(edge.getTargetVertexId(), new Message(message.srcId.get(),
               message.hopCount.get()-1,vertex.getId().get()));
       }
   }
}
VertexDataStructure vDS = new VertexDataStructure(nodeCount,received);
vertex.setValue(vDS);
vertex.voteToHalt();
```

Example: N Hop Neighbor count

```
 // Process messages and change state
 Long nodeCount = vertex.getValue().vertexValue.get();
  for (Message message : messages) {
    if(message.srcId.equals(vertex.getId())){
        LOG.debug("Reply received" + message);
        if(!received.contains(message.senderId.get())){
            received.add(message.senderId.get());
            LOG.debug(received + " " +vertex.getId() );
            nodeCount++;
        }
    }else if(message.hopCount.get()>0){
        sendMessage(message.srcId,new Message(message.srcId.get(),
                message.hopCount.get()-1,vertex.getId().get()));
         for (Edge<LongWritable, FloatWritable> edge : vertex.getEdges()) {
               sendMessage(edge.getTargetVertexId(), new Message(message.srcId.get(),
                message.hopCount.get()-1,vertex.getId().get()));
        }
    }
}
VertexDataStructure vDS = new VertexDataStructure(nodeCount,received);
vertex.setValue(vDS);
vertex.voteToHalt();
```

Example: N Hop Neighbor count

```
 // Process messages and change state
 Long nodeCount = vertex.getValue().vertexValue.get();
  for (Message message : messages) {
    if(message.srcId.equals(vertex.getId())){
        LOG.debug("Reply received" + message);
        if(!received.contains(message.senderId.get())){
            received.add(message.senderId.get());
            LOG.debug(received + " " +vertex.getId() );
            nodeCount++;
        }
    }else if(message.hopCount.get()>0){
        sendMessage(message.srcId,new Message(message.srcId.get(),
                message.hopCount.get()-1,vertex.getId().get()));
         for (Edge<LongWritable, FloatWritable> edge : vertex.getEdges()) {
               sendMessage(edge.getTargetVertexId(), new Message(message.srcId.get(),
                message.hopCount.get()-1,vertex.getId().get()));
        }
    }
}
VertexDataStructure vDS = new VertexDataStructure(nodeCount,received);
vertex.setValue(vDS);
vertex.voteToHalt();
```

Let’s take a look at other parts

Lab Exercise

Shortest Path Computation

Page Rank

Up Next

- Hands-on with Apache Spark

Apache Giraph

- MasterCompute class:
  - Master’s compute always runs before the slaves’
  - aggregators are registered here

- Worker Context:
  - Allows execution of user code on a per-worker basis
  - There’s one WorkerContext per worker
  - Methods for pre/post superstep operations