International Journal of Computer and Communication Technology

Volume 6 | Issue 4 Article 15

October 2015

USING PREGEL TO CALCULATE PAGE RANK OF A WEBPAGE A BULK SYNCHRONOUS PARALLEL PROGRAMMING APPROACH

PRASHANT RAGHAV
BE, Manipal University SAS Pune,India, prashant_raghav@ymail.com

USING PREGEL TO CALCULATE PAGE RANK OF A WEBPAGE A BULK SYNCHRONOUS PARALLEL PROGRAMMING APPROACH

PRASHANT RAGHAV

BE, Manipal University SAS Pune,India
E-mail:prashant_raghav@ymail.com

Abstract— Although using graphs to represent networks and relationship is not new; the size of network has been dramatically increasing in the past decade so storing the whole graph in one place is almost impossible. Problems arise when processing very large graphs, when visiting billions of highly connected vertices. In such cases a graph can't fit on a single machine, and the implementation resorts to a big batch distributed over a cluster of machines. The graph needs to be broken into multiple partitions and stored at various locations. This resulted in the need for a framework that can work in a Distributed Environment. Also, by breaking the graph into different partitions, we can manipulate the graph in parallel to speed up the processing. Google Pregel provides a simple straightforward solution to the large-scale graph processing problems. While it sounds similar to MapReduce, Pregel is optimized for graph operations by reducing I/O, ensuring data locality, but also preserving processing state between phases. The paper will give an insight of the Pregel approach for large scale graph processing.

The paper will give an overview of PREGEL's architecture and then will explore use of Pregel to solve real time applications such as finding PageRank of a Webpage. The paper will also give an insight on Bulk Synchronous Programming and will showcase how it increases computation speed with just few simple lines of code.

***Keywords**: Pregel, Distributed Systems,BSP*

I. INTRODUCTION

With the advent of Internet, graphical representation has become more common . For example Maps are graph that are diagrammatic representation of an area. Social Networks are graph that describe relationship among people. Transportation routes create graph of physical connections among geographical locations. Computer network topologies, player performance In a match, run rate graph, rise and fall of temperature in a city .Perhaps we can refer web as most pervasive graph where documents are vertices and links are edges. But Graph representation varies according to the type of problem. Like sometime we need graph to get the shortest path between two cities for that there are shortest path computing algorithms, there are many other graph problems like the page rank algorithm or minimum spanning tree problem . 340 million tweets are made per day , 10,000 payment card transactions are made every second around the world. As the size of data is increasing their representation has become difficult. In this paper I will present a simple but powerful framework for distributed computing developed by Google called Pregel[1]. Pregel Is Framework oriented towards graph based algorithm. A graph based algorithms is one which we can express in terms or vertices of a graph and their edges. Example of a problem includes finding the shortest path between a set of points, to check whether the two points in a graph connected. As a concrete example I will describe how we can use Pregel to determine page rank of a web page.

Programming for clusters require the programmer to be aware of the cluster architecture, communication between machines in cluster, consideration of fault tolerance, and so on . Pregel program can be scaled very easily on a large scale computer cluster without requiring a programmer to worry about the details of distributing the computation. Instead the programmers can concentrate on algorithm they want to implement. 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. It provides flexibility to express arbitrary algorithms. You can refer to it as MapReduce re-thought for graph operations.

II. BULK SYNCHRONOUS PROGRAMMING ALGORTITHM:

Pregel architecture is inspired by the bulk Synchronous Parallel model introduced by Leslie *Valient*. BSP[2] is a computational model for the implementation of parallel algorithms. Its aim is to replicate, in parallel computation, the universality of the von Nuemann model, forming a foundation for successful, commercial parallel computing.

The heart of BSP programming is the BSP computer which consist of a processors connected by a communication network. Each processor can have different threads of computation ,which comprises of a series of superstep . Each superstep consist of 3 component

Concurrent computation: Several independent computation occurring asynchronously take place on every participating processor.

Communication: Exchange of data between processes occur in the form of get and put calls.
Barrier Synchronization : When a process reaches the point called barrier , it waits until all other process have finished computation .

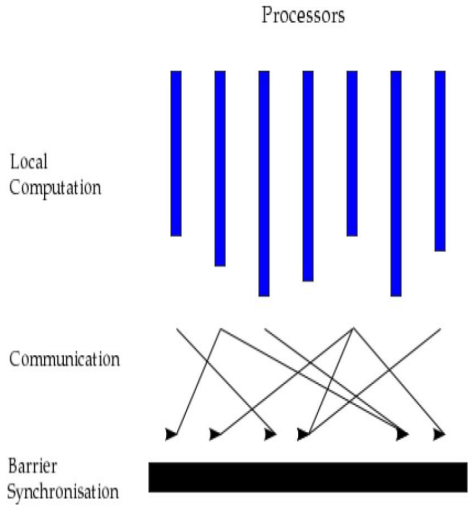

Fig 1:BSP

Each processor executes same algorithm on its data at every superstep,which is a sequence of iteration. User can invoke the defined function at each vertex. Any superstep s will evaluate its state with the message sent from the previous superstep s-1 and the computation made at that step. Superstep S can send message to superstep s+1 . And the cycle goes on. Barrier synchronization occurs when s gets to be s+ 1.

III. PAGE RANK :

PageRank is a link analysis algorithm that is used to determine the importance of a document based on the number of references to it and the importance of the source documents themselves.
A Pregel program takes as input a graph, with many vertices and (directed) edges. The graph might, for example, be the link graph of the web, with the vertices representing web pages, and the edges representing links between those pages. Each vertex is also initialized with a value. For our PageRank example, the value will just be an initial guess for the PageRank.

A. What is a Page Rank ?

PageRank is a numerical value that tells us the importance of a webpage . If the developer has specified a link on one page it is effectively like selecting that page with a vote .The more the counter more the importance of page . The more votes that are cast for a page, the more important the page must be. Also, the importance of the page that is casting the vote determines how important the vote itself is. Google calculates a page's importance from the votes cast for it. How important each vote is taken into account when a page's PageRank is calculated .PageRank is Google's way of deciding a page's importance. It is important because it is one of the factor that helps in determine the page ranking in the search result.

B. How page rank is calculated ?

To calculate page rank, all the incoming links are taken into account both from with in and outside the site .

$$PR(A)=(1-d)+d\bullet(\frac{PR(T_1)}{C(T_1)}+\frac{PR(T_2)}{C(T_2)}+.....+\frac{PR(T_n)}{C(T_n)})$$

1-Tn represents pages linking to page A , 'C' represents number of outbound links that a page. C(1) represents no of outgoing links on page1 has and 'd' is damping factor set to 0.85
We can write as
a page's PageRank = 0.15 + 0.85 * (a "part" of the PageRank of every page that links to it)

Above Equation tells us that a link with PR(5) and 6 outbound links is worth more then a link from a page with PR9 and 100 outbound link. Page Rank of page that links to your page is important but also number of link on that page. The more the no of links lesser the page rank value your page receives from it.

C. Understanding with an Example

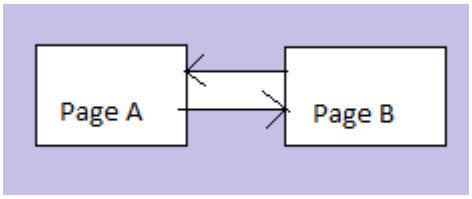

Fig 2: Page Rank

In the Fig each page has one outgoing link so outgoing count is 1 so C(A)=1 ,C(B)=2

Case 1: Initial PR(A) = 1 PR(B)=1
d=0.85
PR(A) = (1-d)+d(PR(B)/1)
PR(B) = (1-d)+d(PR(A)/1)
PR(A) = 0.15 + 0.85*1 = 1
PR(B) = 0.15 + 0.85*1 = 1

Case2: PR(A)=.80 PR(B) =0
d=0.85
PR(A) = (1-d)+d(PR(B)/1)
PR(B) = (1-d)+d(PR(A)/1)
PR(A) = 0.15 + 0.85*0 = .15
PR(B) = 0.15 + 0.85*.15 = .2775

Next Best
PR(A) = 0.15 + 0.85*.2775 = .3858
PR(B) = 0.15 + 0.85*.3858 =. 4779
Next Best
PR(A)=0.15+0.85*.4779 = .5562
PR(B)=0.15+0.85*.5562=.6227
and so on. The numbers just keep going up. But the numbers stop after reaching 1

Case 3: PR(A) = 40 PR(B) = 40

PR(A) = 0.15 + 0.85 * 40 = 34.15

PR(B) = 0.15 + 0.85 * 34.15 = 29.1775

Next Best

PR(A) = 0.15 + 0.85 * 29.1775 = 24.950875

PR(B) = 0.15 + 0.85 * 24.950875 = 21.35824375

Numbers heading down and will stop at 1.

it doesn't matter where you start your guess, once the PageRank calculations have settled down, the "*normalized probability distribution*" (the average PageRank for all pages) will be 1.0

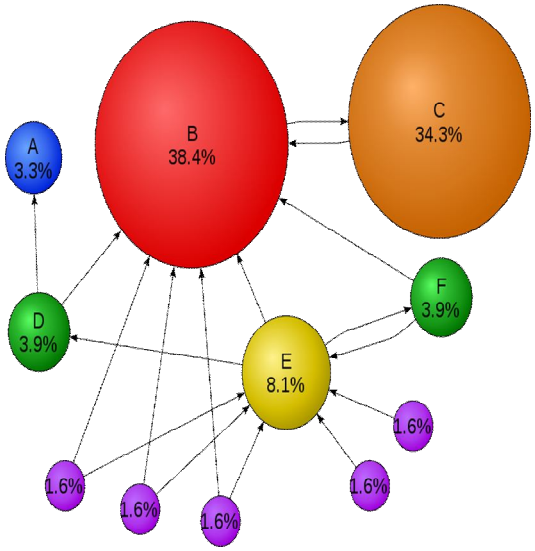

Fig 3:Page Rank Algorithm

IV. PREGEL OVERVIEW

Pregel is a system for large scale Grpah Processing .It allows the developer to write a vertex-centric algorithm for graph processing which means you just have to write a function that receives messages from vertices and sends messages to other vertices do not worry about things as distribution and fault-tolerance.

To create a Pregel program subclass the predefined Vertex class. The user has to override the Compute() method., which is computed at every active vertex in the respected superstep.
Creating a Pregel program typically involves sub classing the predefined Vertex class. The user overrides the virtual Compute() method. This method is the function that is computed for every active vertex in supersteps. Compute() can get the vertex's associated value by GetValue() or modify it using MutableValue**()** Values of edges can be inspected and modified using the out-edge iterator. The class Vertex conatins methods like VoteToHalt() , SendMessageTo() , GetValue(), and const methods superstep() and vertex_id().
Vertex communicates with one another by sending messages.Each message consists of a value and the name of the destination vertex. Any number of messages can be sent in a superstep. A message can be sent to any vertex if it's identifier is known.
Combiners are provided in pregel to prevent message passing overhead by combining messages where ever required.It also provides Aggregators which allow global communication by receiving messages from multiple vertices, combining them and sending the result back to the vertices used mainly in statistics.

V. PAGE RANK IN PREGEL :

1. *Initialization :*

Start each vertex (i.e webpage) off with an estimate for its PageRank[3]. Let's assume it to be 1/(no of vertex). The value of a vertex represents the tentative page rank of the vertex. In each of 30 superstep, each vertex sends its tentative page rank along all of its outgoing edges

2. *Superstep :*

Computation proceeds through a series of superstep
Each Vertex 2 task : 1)update its own value ; (2) send messages to adjacent vertices
The way it updates its own value is by computing a user-specified function which depends on the value of the vertex at the end of the previous superstep, as well as the messages sent to the vertex during the last superstep. Similarly, the messages the vertex sends can depend both on its value and the messages it was sent during the last superstep. From Superstep 1 to 30, each vertex sums up the values arriving on all its messages and sets its tentative page rank to

(1-0.85)/this.getTotalVertices()+0.85*sum;

this.setValue(newRank);

3. *Halting:*

There is an attribute of the vertex that determines whether the vertex is active or not. Initially the value of vertex is Active but can change to Inactive at any superstep. The computation halts when every vertex

is inactive. The paper notes that inactive vertices can be reactivated, although it is a little vague on when this happens.

```
public class PageRankVertex extends Vertex{

 public PageRankVertex(){

 }

public void compute(MessageIterator mi) {
if(this.getSuperStep()>=1){
 double sum=0;
 for(Message m : mi.getMessages()){
 sum+=mi.getMessageValue();
 }
double newRank=(1-
.85)/this.getTotalVertices()+0.85*sum;
          this.setValue(newRank);
       }

       if(this.getSuperStep()<this.getMaxIteration()){
          int numEdges=this.getEdges().size();
          for(Edge e : this.getEdges()){
             this.sendMessage(e,
this.getValue()/numEdges);
          }
       }
   }
```

Fig 4 : Page Rank implementation in JAVA

Web pages (10 anchors per page)	Tasks	Supersteps	Job Execution Time
10 million	17	25	2596.858 seconds
10 million	90	25	551.194 seconds
20 million	35	25	2349.266 seconds
20 million	90	25	1122.242 seconds
30 million	54	25	2636.32 seconds
30 million	90	25	1629.504 seconds

Fig 5: Page Rank Execution time [5]

VI. THE PREGEL INSPIRATION

Just as with MapReduce again, several teams took inspiration from the Pregel paper and started to write their own implementations, or at a minimum refer to Pregel as a basis for comparison with their own approach. And, of course, Pregel is not available to the public. Here is a quick survey of the most visible alternatives[4].

Apache HAMA

Apache Hama is a distributed computing framework based on BS .It was inspired by Google's Pregel but different in the sense that it's purely BSP and common model, not just for graph. What this means for you, a developer willing to work on graph processing, is that you have to build the Pregel API/layer on-top-of it. What Apache Hama provides are the BSP primitives, so messaging between tasks and a synchronization barrier.

Apache Giraph

Developed orignaly by Yahoo ,The Giraph API, as far as one can tell, is **closer to Pregel's** and more complete in this respect, since the project was explicitly started as a clone. Giraph is a Hadoop Map-only job, so you can run it on your existing Hadoop infrastructure, but it provides the API and the middleware of Pregel.

Phoebus

This project is mentioned here because it's also stated as a "Pregel implementation". Phoebus is written in Erlang.

GoldenOrb

Yet another Pregel clone, or used to be. Besides, it was supported by Ravel Data, a company that is probably out of business (its website is down).

VII. CONCLUSION AND FUTRUE WORK

- Pregel is a scalable and fault-tolerant platform with an API that is sufficiently flexible to express arbitrary graph algorithms
- Relaxing the synchronicity of the model
- Not to wait for slower workers at inter-superstep barriers
- Implementing Pregel to use for recommendation system
- Caring dense graphs in which most vertices send messages to most other vertices

REFERENCES

[1] http://dl.acm.org/citation.cfm?id=1807167.1807184J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] http://en.wikipedia.org/wiki/Bulk_synchronous_parallel

[3] http://blog.udanax.org/2010/06/summary-of-google-pregel.html

[4] http://java.dzone.com/news/google-pregel-graph-processingArticle in a journal

[5] http://wiki.apache.org/hama/Benchmarks

◈◈◈

.