MapReduce vs Bancos de Dados Paralelos no cálculo de medidas de centralidade em grafos

Fabiano da Silva Fernandes[1], Eduardo Javier Huerta Yero[1]

[1]Mestrado em Ciência da Computação – Faculdade de Campo Limpo Paulista (FACCAMP). Rua Guatemala, 167 - Jardim América – 13.231-230 - Campo Limpo Paulista / SP – Brasil

contato@fabianofernandes.adm.br, huerta@cc.faccamp.br

Abstract. *In recent years we have witnessed an unprecedented explosion on the amount of data digitally available. A significant part of this data can be modeled using graphs, particularly data related to social networks. This article compares the effectiveness of two paradigms: the MapReduce model and the Parallel System Database when calculating eccentricity measures (radius and diameter) for a graph. Preliminary results indicate that Parallel System Databases are better suited to solve this problem than MapReduce-based solutions.*

Resumo. *Os últimos anos têm visto uma explosão sem precedentes na quantidade de dados gerados. Muitos destes dados podem ser modelados através de grafos, principalmente aqueles provenientes de redes sociais. Este artigo analisa a efetividade de dois paradigmas: o modelo MapReduce e os Sistemas de Gerenciamento de Bancos de Dados Paralelos (SGBD paralelo) para cálculo de medidas de centralidade em grafos (raio e diâmetro). Resultados preliminares indicam que os SGBD paralelos conseguem calcular as medidas de centralidade de forma mais eficiente do que as soluções baseadas em MapReduce.*

1. Introdução

Durante a última década a quantidade de dados disponíveis digitalmente tem aumentado de forma significativa. Esta nova realidade trouxe consigo enormes desafios computacionais que, por sua vez, têm produzido um espectro de soluções capazes de lidar com grandes massas de dados. Dentre elas destacam-se os Sistemas de Gerenciamento de Bancos de Dados (SGBD) paralelos e implementações do modelo MapReduce.

MapReduce é um modelo apresentado pelo Google [Dean and Ghemawat, 2008] e que, apesar da sua ampla adoção, tem sofrido duras críticas vindas da comunidade de bancos de dados. Em particular Stonebraker em [Stonebraker, Abadi, DeWitt, & Madden, 2010] compara o modelo MapReduce com SGBDs paralelos, uma solução já existente há algumas décadas e também capaz de lidar eficientemente com grandes volumes de dados. Os resultados apresentados por Stonebraker indicam que, para problemas tradicionais que podem ser eficientemente representados usando o modelo relacional, um SGBD paralelo tem melhor desempenho do que a solução equivalente utilizando MapReduce.

Neste cenário de explosão de dados disponíveis os grafos têm sido bastante utilizados como ferramenta de modelagem e representação. O modelo relacional nunca foi considerado particularmente bem sucedido para processar grafos, o que inclusive gerou uma quantidade considerável de iniciativas de bancos de dados especificamente projetados para lidar com eles. Para o processamento de grafos grandes, a escolha tem recaído no modelo MapReduce, como mostra o algoritmo HEDA *(Hadoop-based Exact Diameter Algorithm)* [Nascimento and Murta, 2012], do qual trataremos neste artigo. Não há relatos na literatura sobre o comportamento de SGBD paralelos tratando específicamente com grafos considerados grandes.

Este trabalho compara o modelo MapReduce e um Sistema de Gestão de Banco de Dados Paralelo no processamento de grafos grandes. Para este fim utilizamos as medidas de centralidade, que exigem calcular a distância entre todos os pares de nós do grafo e geram, portanto, um problema computacionalmente complexo [Del Vecchio, Lima, Galvão, & Loures, 2009].

O trabalho está organizado como a seguir. A seção 2 descreve brevemente o modelo MapReduce e apresenta o algoritmo HEDA, baseado neste modelo e utilizado nos experimentos. A seção 3 descreve o algoritmo utilizado no SGBD parelo. A seção 4 mostra os resultados experimentais. A seção 5 analisa dos resultados obtidos, e a seção 6 apresenta as conclusões do trabalho.

2. Modelo MapReduce – Algoritmo HEDA

O modelo MapReduce consiste essencialmente em duas funções básicas: *map()*, que recebe como entrada um conjunto de pares de chave/valor e gera um novo conjunto intermediário de pares chave/valor e *reduce()*, que recebe como entrada o conjunto de pares intermediários gerado pela função *map()* (previamente agrupados de forma tal que todos os pares com a mesma chave são entregues à mesma instância da função) e que gera o conjunto final de pares chave/valor.

Neste trabalho utilizamos o algoritmo HEDA, baseado no modelo MapReduce. Este algoritmo tem como objetivo objetivo calcular o diâmetro e raio de grafos. Baseado em busca em largura, ele resolve o problema conhecido como SSSP (Single Source Shortest Path) usando cada nó do grafo como ponto de partida para calcular a distância entre todos os pares de nós do grafo.

3. Cálculo de medidas de centralidade em um SGBD paralelo

O algoritmo para o cálculo de medidas de centralidade de um grafo usando um SGDB paralelo consiste em uma sequência de consultas para encontrar o menor caminho entre todos os pares de vértices do grafo usando uma estratégia de expansão dos vértices adjacentes similar à utilizada no algoritmo de Dikjstra.

As consultas estão divididas em duas partes. A primeira é responsável por encontrar o menor caminho entre todos os pares de vértices (Algoritmo 1), enquanto a segunda parte é responsável pelo cálculo do raio e diâmetro do grafo e é mostrada na (Algoritmo 2).

O Algoritmo 1 começa criando 3 tabelas temporárias. A cada passo estas tabelas temporárias são utilizadas para armazenar as distâncias parciais já calculadas entre os pares de vértices que já foram processados. O cálculo dos menores caminhos é realizado

entre a linha 8 e a linha 11. A linha 12 armazena os valores computados na iteração corrente. Quando não há mais inserções de resultados na tabela *TEMP RESULT* o algoritmo é interrompido, pois a partir desse momento não haverá mais alterações nos resultados. O Algoritmo 2 extrai o raio e o diâmetro a partir das distâncias calculadas no Algoritmo 1.

Algoritmo 1: Cálculo do menor caminho entre todos os pares de vértices

```
   Data: G = Grafo
   Result: D = Menor Caminho entre Todos os Pares de Vértices de G
 1 CREATE NEW TEMPORARY RELATION NEW TEMP ;
 2 CREATE NEW TEMPORARY RELATION OLD TEMP ;
 3 CREATE NEW TEMPORARY RELATION TEMP RESULT ;
 4 NEWTEMP ← G;
 5 i ← 0;
 6 UPDATE NEW TEMP SET timestep=1 ;
 7 while TEMP RESULT muda tamanho do
 8     OLDTEMP ← NEWTEMP;
 9     R ← G join OLD TEMP distance + 1, timestep+=1 ;
10     R' ← SELECT * R WHERE R.timestep=i+1;
11     INSERT INTO NEW TEMP SELECT * FROM R' ;
12     INSERT INTO TEMP RESULT SELECT v1, v2, distance, timestep FROM R'
13 end
14 return SELECT v1, v2, MIN(distance) FROM TEMP RESULT;
```

Algoritmo 2: Extração do diâmetro e o raio do grafo

```
   Data: D = Menor Caminho entre Todos os Pares de Vértices
   Result: raio, diametro
1  R ← SELECT D.v1, MAX(D.distance) AS dist FROM D ;
2  R' ← SELECT MAX(dist)as diameter, MIN(dist)as radius FROM R ;
3  return r ← R'.radius,  d ← R'.diameter;
```

4. Projeto dos experimentos

Os experimentos realizados neste trabalho utilizaram um dos ambientes computacionais oferecidos pelo Grid'5000 [Balouek and Quesnel, 2013], composto por 45 máquinas com 2 Processadores Intel Xeon Dual Core, 32GB de memória RAM, e um disco rígidos de 350GB SATA II. Todas as máquinas receberam o Sistema Operacional Linux CentOS 5.9. O Java JDK versão 1.7.0_51, Apache Hadoop, versão 1.2.1. e o SGBD paralelo Pivotal Greenplum Database, versão 4.2.2.

Foram utilizados 4 grafos para comparar o desempenho das duas soluções testadas. Dois deles são grafos reais, obtidos da topologia da rede de colaboração da *High Energy Physics Theory* disponível no *dataset* da Universidade de Stanford. Os outros dois grafos são grafos sintéticos gerados a partir do mesmo gerador do trabalho [Kang, Tsourakakis, Appel, Faloutsos, & Leskovec, 2011], baseado no modelo de Erdös-Rényi. A Tabela 1 apresenta a relação dos grafos e quantidade de arestas que cada um possui.

Tabela 1. Grafos utilizados nos experimentos

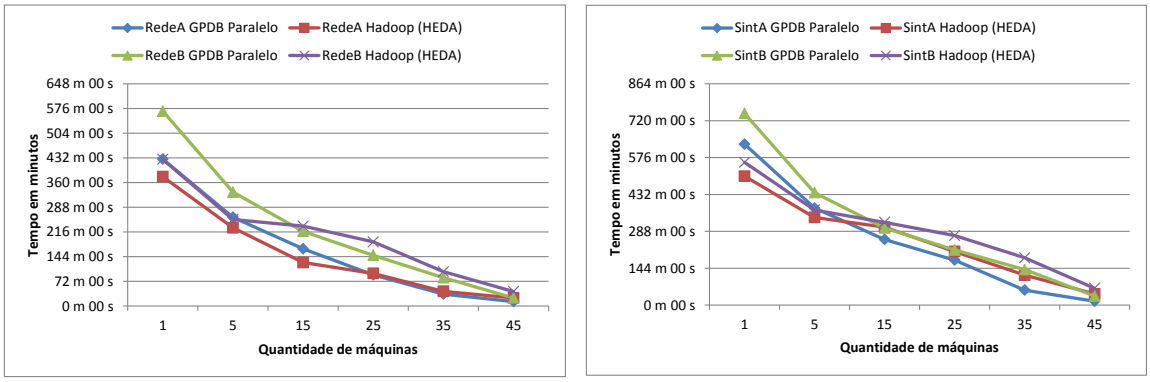
Identificador	Tipo de Grafo	Vértices	Arestas
RedeA	Rede de Colaboração	9.877	51.971
RedeB	Rede de Colaboração	12.008	237.010
SintA	Sintético	20.000	200.000
SintB	Sintético	100.000	300.000

No SGBD paralelo os dados foram segmentados pelos vértices de origem, utilizando função *hash*, de forma tal que os dados relacionados a um determinado vértice estivessem localizados no mesmo nó do cluster. No caso do Hadoop, os grafos são representados por dois arquivos em formato texto gravados no HDFS. Neste caso não é possível especificar explicitamente como particionar os dados, como fizemos no SGBD paralelo. Em vez disso, o Hadoop HDFS particiona os arquivos em blocos, cria

repetições para cada bloco e, em seguida, distribui aleatoriamente todos esses blocos entre as várias máquinas do *cluster*.

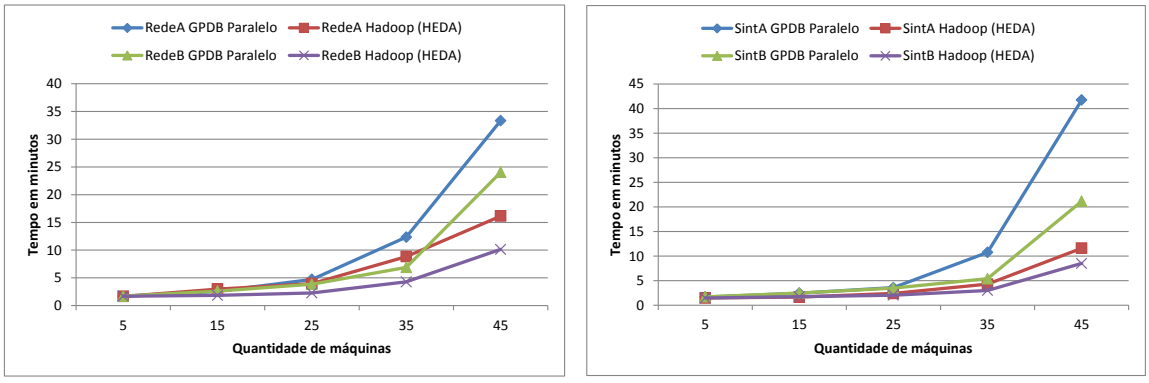
A Figura 1 mostra os tempos de execução do cálculo da centralidade para os grafos reais RedeA e RedeB e para os grafos sintéticos SintA e SintB. Ressaltamos que nos tempos apresentados não foram considerados o tempo de carregamento dos grafos. Para ilustrar melhor o comportamento de cada plataforma, a Figura 2 mostra o *speedup* de cada uma delas para cada configuração testada.

O HEDA se mostra em ambos os casos a opção mais eficiente quando usamos poucos computadores. O SGBD paralelo mostra maior capacidade para gerenciar eficientemente uma maior quantidade de recursos computacionais (Figura 1). Para 45 computadores, os resultados obtidos pelo SGBD paralelo são aproximadamente 47% melhores do que o HEDA.



Grafos Reais **Grafos Sintéticos**

Figura 1. Tempos de execução para os grafos RedeA e RedeB



Grafos Reais **Grafos Sintéticos**

Figura 2. Speedup para os grafos Reais e Sintéticos.

Estes dados comprovam que o SGDB paralelo é mais escalável, pois aproveita de forma mais efetiva os novos recursos computacionais disponibilizados. O gráfico da Figura 2 parece indicar que esta diferença irá se acentuar em configurações com mais computadores. Vale a pena ressaltar o *speedup* maior que 40 para o grafo SintA usando 45 computadores, um valor próximo do ótimo.

5. Análise dos resultados

Há várias razões que contribuem para que o SGBD paralelo obtenha melhores tempos de execução do que o Hadoop ao calcular as medidas de centralidade nos grafos. Uma delas é o fato do SGBD paralelo fazer p*arsing* dos dados uma única vez, no momento do carregamento, enquanto o MapReduce força todas as tarefas *map* e *reduce* a fazer o *parsing* dos dados. Outra vantagem do SGBD paralelo está no uso de técnicas de *pipelining* para comunicar os resultados obtidos por um operador para o próximo. O MapReduce "materializa" estes resultados intermediários no HDFS, incorrendo portanto, em uma maior sobrecarga. Por fim, o SGBD paralelo cria o plano de execução da consulta antes de começar sua execução, o que permite um escalonamento mais eficiente dos recursos computacionais. No caso do MapReduce o escalonamento de tarefas é feito em tempo de execução, com os computadores recebendo novas tarefas na medida em que terminam as anteriores.

De forma geral, os resultados obtidos mostram que o MapReduce sacrifica desempenho para conseguir objetivos como tolerância a falhas, adaptabilidade a mudanças no ambiente de execução. O SGBD paralelo, por outro lado, foca em extrair o melhor desempenho possível dos recursos computacionais disponíveis, ainda que ao custo de uma arquitetura mais complexa e rígida.

6. Conclusões e trabalhos futuros

Os resultados mostram que, apesar do SGBD paralelo ser relativamente lento com poucos computadores, ele se torna a melhor opção para executar o cálculo da centralidade em grafos quando a quantidade de computadores disponíveis aumenta, chegando a obter tempos até 66% menores que o HEDA para o grafo SintA usando 45 computadores. Os trabalhos futuros incluem executar experimentos em um *cluster* maior, e utilizando grafos maiores, com o objetivo de verificar se o comportamento observado nestes testes se mantém para configurações maiores.

Referências

Balouek, D., Lèbre, A., & Quesnel, F. (2013). Flauncher and DVMS -- Deploying and Scheduling Thousands of Virtual Machines on Hundreds of Nodes Distributed Geographically. Finalist of the IEEE International Scalable Computing Challenge (SCALE 2013).

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplied Data Processing on Large Clusters. ACM Digital Library, 107-113.

Del Vecchio, R., Lima, L., Galvão, D., & Loures, R. (2009). Medidas de Centralidade da Teoria dos Grafos aplicada a Fundos de Ações no Brasil. XLI Simpósio Brasileiro de Pesquisa Operacional, Porto Seguro.

Nascimento, J. P., & Murta, C. (2012). Um algoritmo paralelo em Hadoop para Cálculo de Centralidade em Grafos Grandes. XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Ouro Preto: SBC.

Stonebraker, M., Abadi, D., DeWitt, D., & Madden, S. (2010). MapReduce and parallel DBMSs: friends or foes? Communications of the ACM. 53(1):64-71.