Um Servidor Escalável para Bases Massivas de Dados Geográficos
Disciplina de Projeto e Análise de Algoritmos

Leandro da Silva Santos, Ricardo Augusto Rabelo Oliveira e Tiago Garcia de Senna Carneiro
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email:{leandrosilva,rabelo,tiago}@iceb.ufop.br

***Resumo*—Sistemas de Informação Geográfica com dados de alta resolução em escala global trabalham com dados na faixa dos terabytes. Para viabilizar o acesso a essa quantidade de dados é necessário elaborar um servidor que trabalhe sobre essa massa de dados em tempo real. Características como um sistema de arquivo distribuído, processamento paralelo e alto grau de indexação são necessárias para construir algoritmos eficazes para o tratamento desta escala de informação. Nesta linha, respectivamente, o *framework* Hadoop fornece soluções como HDFS, MapReduce e HBase similares a infraestrutura do Google Maps.**

***Keywords*-servidor, Hadoop, MapReduce, HBase.**

I. INTRODUÇÃO

Hoje, na área de Sistemas de Informação Geográfica (SIG), há a necessidade de viabilizar o acesso em tempo real a uma quantidade massiva de dados (matricial, vetorial e textual) para clientes de sistemas móveis e estações de trabalho como mostrado na Figura 1.

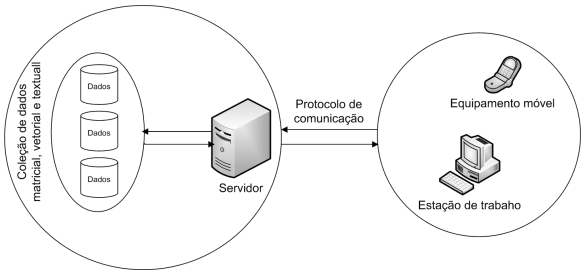

Figura 1. Arquitetura cliente-servidor.

A arquitetura proposta deverá fornecer soluções que podem ser divididas em dois grupos. No primeiro, deverá fornecer soluções na área de armazenamento e processamento de dados na escala dos terabytes. No segundo, deverá prover acesso ao dado a uma quantidade massiva de clientes em tempo real.

O escopo do artigo é propor uma estrutura de armazenamento e processamento de dados de natureza matricial, neste caso, imagens de satélite.

A. Armazenamento

A estrutura de armazenamento adotada será o software livre Hadoop [1] *Distributed File System* (HDFS) sistema de arquivo distribuído baseado modelo *Google File System* (GFS) [2]. O HDFS [3] é um sistema de arquivo distribuído construído sobre uma arquitetura mestre-escravo que propicia alta disponibilidade, alta largura de banda e escabilidade sobre *commodities* (computadores) de baixo custo [4].

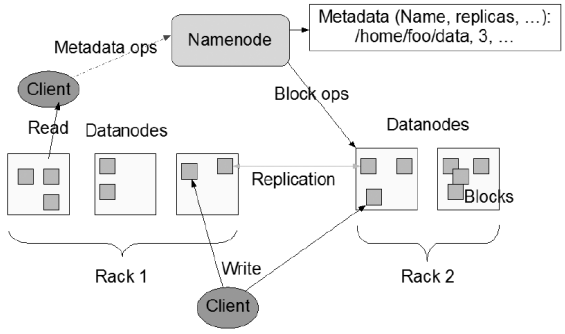

Figura 2. Arquitetura do sistema de arquivo distribuído HDFS [5].

O HDFS disponibiliza operações e uma estrutura de diretórios semelhantes ao sistema POSFIX, porém prioriza operações em lote e uma alta transferência de dados [5]. A arquitetura HDFS, mostrada na Figura 2, é composta por duas entidades básicas: Namenode e Datanode.

O Namenode é o servidor principal responsável pelo gerenciamento do espaço de nomes do sistema de arquivo e por regular o acesso dos clientes aos arquivos [5]. O Datanode é responsável pela leitura e escrita das requisições feitas pelos clientes ao sistema de arquivo [5]. Tal modelo possibilita a descentralização da operações de escrita e leitura visto que o Namenode apenas informa a qual Datanode o cliente deve solicitar a operação. A disponibilidade do dado é garantida pela replicação do mesmo nos Datanodes [5].

A estrutura de banco de dados adotada será o *software* livre HBase. O HBase é um banco de dados semi-estruturado que se apóia sobre o HDFS [6]. Este banco de dados foi

implementado seguindo o modelo proposto pelo banco de dados BigTable [7] do Google.

O HBase possibilita o acesso em tempo real a dados na escala dos terabytes [6]. Sobre o HBase é possível adotar um modelo de processamento paralelo dos dados como mostrado na Subseção I-B. O modelo de dados do HBase como mostrado na Figura 3 é composto por uma tripla de valores do tipo <linha,coluna,tempo>.

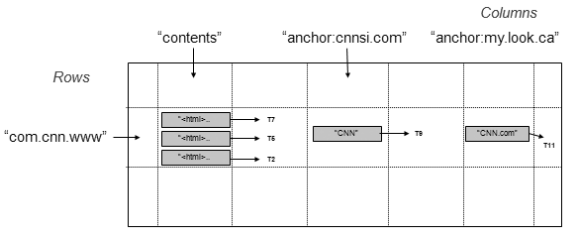

Figura 3. Modelo de dados básico do sistema HBase [7].

As principais características deste modelo podem ser descritas como:

- Armazenamento orientado a colunas;
- Foco na leitura das colunas;
- Nome de colunas em dois níveis;
- Controle de versões de células pelo campo *time*;
- Não suporta múltiplas operações sobre a linha;
- Não suporta o modelo relacional.

B. Processamento

Para o processamento de uma grande quantidade de dados é necessário um alto grau de paralelismo. Neste enfoque a estrutura de programação paralela adotada será o MapReduce definido pelo Google [8] e implementado como *software* livre pelo Hadoop.

No modelo de programação MapReduce os detalhes de paralelismo são ocultados do usuário. Para implementar um programa MapReduce é necessário apenas a definição de duas funções: Map e Reduce [9].

A função Map processa um par $(chave, valor)$ e produz um conjunto de valores intermediários de pares $(chave, valor)$ [8]. A função Redce é responsável pela agregação de todos os valores intermediários associados a uma chave como mostrado na Figura 4.

No modelo de programação paralela do MapReduce, Figura 5, existem dois atores: *Master* e *Worker*. O *Master* é responsável por atribuir as funções de *Map* e *Reduce* às *commodities* (servidores) [11]. O Worker Map será o responsável por executar a função de *Map* sobre o *split* (fragmento do dado de entrada). O *Worker Reduce* será responsável por executar a função de *Reduce* sobre os dados produzidos pelo *Worker Map* [11]. No Hadoop a nomeclatura *Master* e *Worker* corresponde, respectivamente, a *Job Tracker* e a *TaskTracker*.

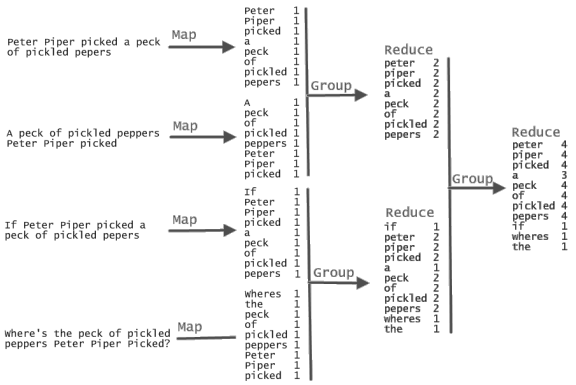

Figura 4. Exemplo de execução de um programa MapReduce. Contador de palavras [10].

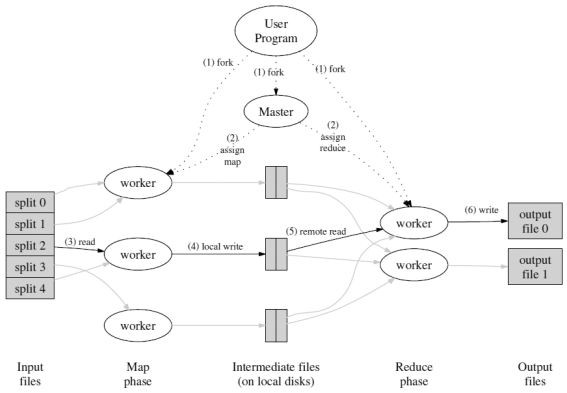

Figura 5. Modelo de programação MapReduce [11].

II. Métodos

A metodologia adotada para o armazenamento e processamento do dado de entrada visa implementar um modelo similar ao utilizado pelo Google Maps. O modelo de armazenamento e processamento pode ser dividido em duas etapas: refinamento e indexação do dado.

A. Refinamento

O modelo de refinamento adotado, Figura 6, busca transformar o dado bruto, imagem de satélite, num sistema de visualização piramidal derivado num conjunto de blocos. Neste sistema há um aumento da definição do dado à medida que se aproxima da base da pirâmide. Um aspecto importante deste modelo está na definição do número de níveis da pirâmide de visualização. Cada nível é subdividido em 4^n blocos, sendo n o número do nível (zero é o nível inicial). Cada bloco tem uma dimensão de 256x256 pixels.

A implementação adotada deste modelo de refinamento exige que o dado de entrada esteja no sistema de referência espacial EPSG:4326 descrito na Tabela I. O dado adquirido tem seu sistema de referência espacial modificado para EPSG:900913 descrito na Tabela I. E, em seguida, ocorre

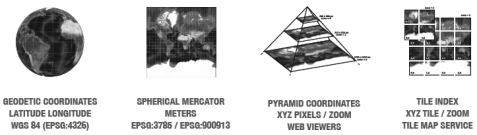

Figura 6. Modelo de processamento do dado. [12]

a construção do modelo piramidal de visualização e a segmentação de cada nível em 4^n blocos.

Tabela I
SISTEMA DE REFERÊNCIA ESPACIAL

	EPSG:4326	EPSG:900913
Projeção:	Latlong	Mercator (1SP)
Elipisoide:	WGS84	WGS84
Datum:	WGS84	WGS84

A transformação do dado bruto para o modelo de visualização piramidal ficará a cargo da biblioteca de tratamento de dados geográficos GDAL [13] (Geospatial Data Abstraction Library). A biblioteca GDAL [14] contém funções que contribuiram para a elaboração do modelo de visualização piramidal por propiciar a manipulação de dados matriciais em diversos formatos (TIFF/GeoTIFF, MrSID, ECW, JPEG2000, JPEG e PNG) [13].

B. Indexação

O modelo de indexaçào adotado utiliza-se da estrutura de dados *QuadTree* [15] para indexar os blocos resultantes da geração do modelo de visualização piramidal como mostrado na Figura 7.

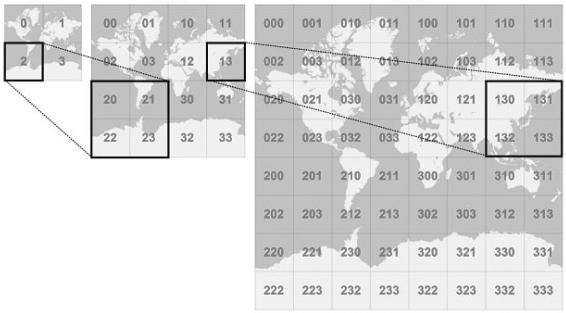

Figura 7. Modelo de indexação utilizando a QuadTree [16].

III. ANÁLISE DE COMPLEXIDADE

A complexidade de tempo e de espaço para o refinamento do dado bruto está relacionada ao número de nível gerados para a construção do modelo de visualização piramidal. Sendo n o número de níveis a complexidade pode ser expressa por $O(n)$;

REFERÊNCIAS

[1] "Welcome to apache hadoop!" http://hadoop.apache.org/, visitado em 20/06/2011.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, October 2003.

[3] C. Lam, *Hadoop in Action*. Manning Publications, 12 2010.

[4] D. Moise, G. Antoniu, and L. Bougé, "Improving the hadoop map/reduce framework to support concurrent appends through the blobseer blob management system," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10, 2010, pp. 834–840.

[5] D. Borthakur, *The Hadoop Distributed File System: Architecture and Design*, The Apache Software Foundation, 2007.

[6] D. Borthakur, J. Gray, J. S. Sarma, K. Muthukkaruppan, N. Spiegelberg, H. Kuang, K. Ranganathan, D. Molkov, A. Menon, S. Rash, R. Schmidt, and A. Aiyer, "Apache hadoop goes realtime at facebook," in *Proceedings of the 2011 International Conference on Management of data*, ser. SIGMOD '11. New York, NY, USA: ACM, 2011, pp. 1071–1080.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06)*, 2006.

[8] R. Lämmel, "Google's mapreduce programming model - revisited," *Sci. Comput. Program.*, vol. 68, October 2007.

[9] J. Dean and S. Ghemawat, "Mapreduce: a flexible data processing tool," *Commun. ACM*, vol. 53, pp. 72–77, January 2010.

[10] Tarn, "Map-reduce on mongo," http://sharpthinking.com.au/author/tarn.aspx, visitado em 04/07/2011.

[11] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, January 2008.

[12] K. P. Pridal, "Tiles à la google maps: Coordinates, tile bounds and projection - conversion to epsg:900913 (epsg:3785) and epsg:4326 (wgs84)," http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/, visitado em 11/07/2011.

[13] "Gdal." in *Encyclopedia of GIS*, S. Shekhar and H. Xiong, Eds. Springer, 2008, p. 329.

[14] "Gdal - geospatial data abstraction library," http://www.gdal.org/, visitado em 06/07/2011.

[15] R. E. Loke and J. M. H. du Buf, "Segmentation of range images in a quadtree." ser. Lecture Notes in Computer Science, F. J. P. López, A. C. Campilho, N. P. de la Blanca, and A. Sanfeliu, Eds., vol. 2652. Springer, 2003, pp. 428–436.

[16] "Bing maps tile system," http://msdn.microsoft.com/en-us/library/bb259689.aspx, visitado em 03/07/2011.