InterIMAGE Cloud Platform: Em direção à arquitetura de uma plataforma distribuída e de código aberto para a interpretação automática de imagens baseada em conhecimento

Rodrigo da Silva Ferreira [1]
Dário Augusto Borges Oliveira [1]
Patrick Nigri Happ [1]
Gilson Alexandre Ostwald Pedro da Costa [1]
Raul Queiroz Feitosa [1,2]
Cristiana Bentes [1,2]

[1] Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio
Caixa Postal 38097 - 22451-900 – Gávea – Rio de Janeiro - RJ, Brasil
rsilva@ele.puc-rio.br, darioaugusto@gmail.com, {patrick,gilson,raul}@ele.puc-rio.br

[2] Universidade do Estado do Rio de Janeiro - UERJ
20550-900 – Maracanã – Rio de Janeiro - RJ, Brasil
raul@ele.puc-rio.br, cris@eng.uerj.br

Abstract. In this paper we introduce the architecture of a new open-source, knowledge-based, distributed platform for automatic image interpretation named InterIMAGE Cloud Platform (ICP). ICP is designed for distributed processing and is able to handle very large volumes of data using clusters of low-cost computers. Thus, it is possible to understand ICP as a functional and architectural redesign of the first version of InterIMAGE, moving towards a scalable solution for dealing with the very high-resolution remote sensing imagery currently available. As well as many recent applications dealing with large-scale data processing, the architecture aforementioned envisages the usage of the Hadoop framework, which offers a highly scalable and reliable platform for storing and processing massive data sets in computer clusters. Additionally, the usage of a high-level data-flow language (Pig Latin) offers the possibility to extend ICP seamlessly. Lastly, ICP provides an intuitive way to embed knowledge into the system through semantic networks and operator graphs, which express the processing flow of the interpretation model. It is important to remark that ICP is also intended to work as a web service, providing the execution of the platform on cloud computing services. Preliminary experiments presented in this paper validate ICP concepts and show the scalable nature of the system.

Palavras-chave: remote sensing, distributed systems, object-based image analysis, geographic information systems, sensoriamento remoto, sistemas distribuídos, análise de imagens baseada em objetos, sistemas de informações geográficas.

1. Introdução

O aumento da disponibilidade de imagens de sensoriamento remoto de altíssima resolução nas últimas décadas permitiu a criação de um novo espectro de aplicações, mas acabou também por gerar novos desafios relacionados à capacidade de processamento computacional de tamanho volume de dados. De fato, o crescimento das resoluções espacial, espectral e temporal das imagens de sensoriamento remoto resultou em uma crescente demanda por soluções escaláveis para problemas de interpretação de imagens (Vatsavai, 2013).

Esta não é uma particularidade da área de análise de imagens. Áreas como processamento de linguagem natural (Exner et al., 2014) e análise de *big data* (Shang et al., 2013) também estão lidando com os desafios decorrentes da análise e do processamento de grandes conjuntos de dados.

*Cluster*s de computadores de baixo custo têm se apresentado como uma das principais ferramentas para lidar com este tipo de problema. Nesse contexto, o paradigma de programação MapReduce (Dean e Ghemawat, 2004), mais especificamente a sua versão de

código livre incluída no *framework* Hadoop (Hadoop, 2014), fornece uma plataforma escalável, confiável e de baixo custo para processar e armazenar grandes quantidades de dados em *cluster*s.

Neste trabalho, propomos uma nova plataforma distribuída para a análise automática de imagens chamada *InterIMAGE Cloud Platform* (ICP). Esta plataforma resulta de uma reformulação do sistema InterIMAGE (Costa et al., 2010), concebida para processar grandes conjuntos de dados em um *cluster* de computadores físicos ou virtuais utilizando Hadoop. Desta forma, é possível executar o ICP na nuvem contratando servidores e equipamentos de terceiros sem a necessidade de arcar com os custos de montagem e manutenção da infraestrutura de *hardware*. Neste caso, além dos custos estarem associados apenas ao tempo de processamento, existe a possibilidade de inclusão dinâmica de novos servidores virtuais à infraestrutura corrente.

A arquitetura do ICP é inspirada por dois trabalhos recentes. O primeiro tem como objetivo habilitar o *framework* Hadoop para trabalhar com dados espaciais (Eldawy et al., 2013) e o outro propõe um sistema de *data warehouse* para dados espaciais baseado em Hadoop com aplicação em análise de imagens de citopatologia (Aji et al., 2013).

Nossa abordagem difere dos trabalhos anteriores pela flexibilidade e generalidade de sua arquitetura. Ainda que a aplicação-alvo considerada durante a concepção do ICP fosse o sensoriamento remoto, a arquitetura se propõe a apoiar a resolução de problemas em diversas áreas de conhecimento, como biometria e medicina.

Este artigo apresenta inicialmente uma breve descrição dos *frameworks* Hadoop (Seção 2) e Pig (Seção 3). A seguir, detalhamos a arquitetura proposta (Seção 4) e os seus principais componentes: camada de interpretação (Seção 5); camada de linguagem (Seção 6); e camada MapReduce (Seção 7). Na seção 8, apresentamos os resultados dos experimentos realizados e finalizamos com conclusões e indicações de trabalhos futuros.

2. Hadoop

Hadoop é um *framework* para o desenvolvimento de aplicações que processam de forma paralela e distribuída grandes quantidades de dados em *clusters* de computadores de uma maneira confiável e tolerante a falhas.

Neste *framework*, a distribuição de trabalho e de dados entre as máquinas é realizada de forma automática e eficiente, aproveitando o potencial de paralelismo existente nos vários núcleos das máquinas do *cluster* (Hadoop, 2014). Seu funcionamento é baseado em dois principais conceitos: o sistema de arquivos distribuído (HDFS – *Hadoop Distributed File System*) e o modelo de programação MapReduce.

O HDFS (Ghemawat et al., 2003) divide grandes arquivos de dados em blocos que são controlados por diferentes nós do *cluster*. Cada processo em execução em um determinado nó do *cluster* processa então um desses blocos, que representa um subconjunto dos dados de entrada. O escalonamento dos processos no *cluster* é realizado automaticamente e leva em consideração a localização dos dados. Os processos são levados até os dados, evitando transferências desnecessárias e, consequentemente, produzindo um melhor desempenho.

O modelo MapReduce envolve a execução de *jobs* geralmente compostos por três fases principais: *map*, *shuffle* e *reduce* (Gates, 2011). Na fase *map*, o processamento é realizado em cada registro da entrada separadamente, definido por um par chave-valor. Na fase *shuffle*, os dados são aglomerados de acordo com a chave utilizada, ordenados e então distribuídos em máquinas diferentes para a fase *reduce*. Registros que contenham a mesma chave são enviados para o mesmo *reducer* a fim de realizar o processamento final combinando os diferentes resultados e gerando a saída.

Existem dois tipos de nós que controlam o processo de execução dos *jobs*: um *job tracker* e diversos *task trackers*. O primeiro é o responsável por escalonar e controlar as tarefas e

recursos em execução, enquanto o segundo se encarrega da execução das tarefas de *map* ou de *reduce*, além de reportar o progresso da execução (White, 2012).

Apesar de representar um nível de abstração relativamente simples, a programação em MapReduce pode se mostrar não-trivial quando se deseja tratar problemas mais complexos. Assim, diferentes alternativas surgiram de forma a tornar esta tarefa mais simples e intuitiva, como os *frameworks* Pig (Pig, 2014) e Hive (Hive, 2014). Neste trabalho, utilizamos o Pig, que será brevemente descrito a seguir.

3. Pig

O Pig é um *framework* que fornece um motor para a execução de fluxos de dados de forma paralela em Hadoop. Para tanto, inclui uma linguagem de alto nível, conhecida como Pig Latin, que permite aos desenvolvedores criar programas de processamento paralelo em Hadoop de forma simples e intuitiva. A linguagem inclui operadores para muitas das operações de dados tradicionais (*join*, *sort*, *filter*, etc), além de permitir que usuários desenvolvam suas próprias funções para leitura, processamento e escrita de dados (Gates, 2011).

O *framework* processa os *scripts* em *Pig Latin* e os compila automaticamente em *jobs* MapReduce. Estas características fornecem maior facilidade de programação, possibilidade de otimização na criação de *jobs* MapReduce e extensibilidade através das UDFs (*User Defined Functions*), que são funções definidas pelos usuários.

4. Arquitetura Proposta

O ICP foi concebido como uma plataforma para interpretação automática de imagens que fornece ferramentas para a criação de modelos de interpretação através da definição de redes semânticas e grafos de operadores.

A plataforma provê uma solução distribuída para a execução de grandes volumes de dados através da divisão do processamento entre os nós dos *cluster*s. Para tanto, o dado deve ser dividido de acordo com as configurações de *hardware* disponíveis. As imagens, por exemplo, são separadas em *tiles* de mesmo tamanho (quando possível) que são distribuídos entre os *jobs* MapReduce.

A arquitetura do ICP é composta por três camadas: interpretação, linguagem e MapReduce (Figura 1). As seções seguintes descrevem essas camadas e como elas interagem entre si.

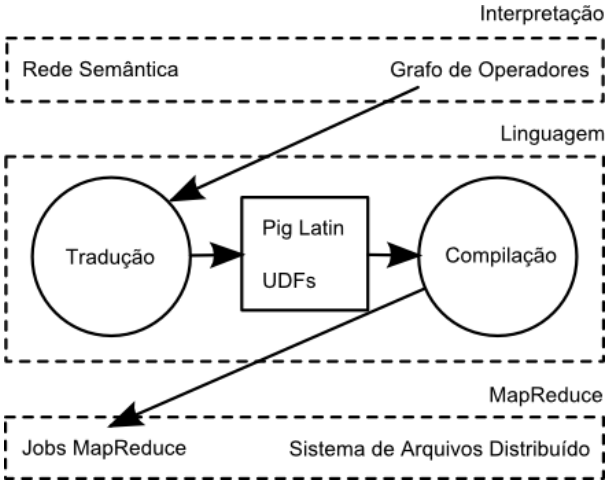

Figura 1. Arquitetura do ICP.

5. Camada de Intepretação

A camada de interpretação compreende todos os componentes que são necessários para a criação de um modelo de interpretação. Para construir um modelo de interpretação no ICP, um usuário deve tipicamente definir uma rede semântica e um grafo de operadores. As seções seguintes descrevem esses dois componentes.

5.1 Rede Semântica

A *rede semântica* representa conceitos ou classes que se espera encontrar (classificar) durante a execução de um modelo de intepretação. O usuário pode construir mais de uma rede semântica por projeto de interpretação (e.g., uma para classes de cobertura e outra para classes de uso do solo), permitindo a construção de modelos complexos e flexíveis.

É interessante ressaltar que, nesta nova proposta, o papel da rede semântica mudou em relação ao sistema original. Naquela versão, além de representar uma hierarquia de conceitos, a rede semântica definia também a lógica de intepretação, ou seja, a sequência de execução de operadores de processamento de imagens durante a interpretação. No ICP, a rede semântica representa apenas conhecimento declarativo, enquanto que o conhecimento procedimental é definido através de um novo componente chamado grafo de operadores, apresentado a seguir.

5.2 Grafo de Operadores

O *grafo de operadores* é um grafo orientado que define de forma inequívoca os operadores utilizados em um processo de interpretação e suas relações. Ele define, portanto, a sequência de passos necessária para se executar um modelo de interpretação. No grafo, os nós representam os operadores, que estão relacionados principalmente com a geração, filtragem e validação de hipóteses associadas às classes definidas na rede semântica, e as arestas representam a estrutura do grafo, indicando a sequência de execução dos operadores (Figura 2).

Existem dois tipos de operadores: operadores autocontidos e conjuntos de regras.

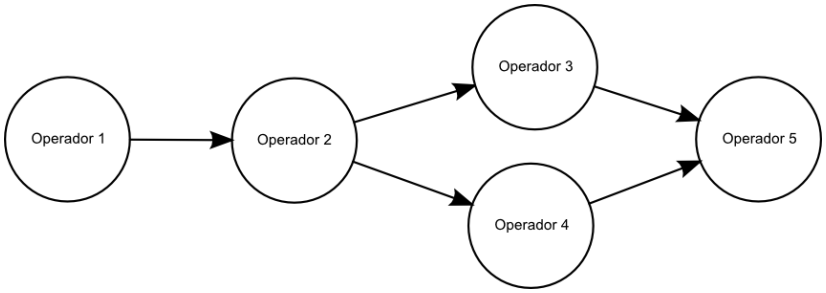

Figura 2. Grafo de operadores

Operadores autocontidos

Operadores autocontidos são geralmente programas executáveis ou funções fornecidas por bibliotecas externas que recebem um conjunto de parâmetros e retornam o resultado correspondente. A comunicação entre dois operadores (definida por uma aresta do grafo) é controlada pela interface dos operadores. O ICP permitirá a integração de bibliotecas externas e *scripts* criados por desenvolvedores através do uso de UDFs. Alguns operadores autocontidos típicos são, por exemplo, segmentadores e classificadores.

Conjuntos de Regras

Os *conjuntos de regras* podem ser considerados como operadores personalizados. Usuários que não são desenvolvedores e precisam de um operador específico, que não esteja disponível como um operador autocontido, podem se utilizar de uma interface gráfica intuitiva para definir um conjunto de regras que eles pretendem aplicar sobre os dados em um

determinado momento do processo de interpretação. Essas regras compreendem operações de filtro, cálculo de atributos espaciais e espectrais, operações espaciais, seleção de atributos, classificação, etc. Através dos conjuntos de regras, a plataforma fornece aos usuários uma ferramenta para construir graficamente seus próprios operadores e adicioná-los ao modelo de interpretação.

6. Camada de Linguagem

Na camada de linguagem, o grafo de operadores definido pelo usuário é traduzido para uma linguagem intermediária, Pig Latin, e então compilada em vários *jobs* MapReduce que são enviados para serem executados no *cluster*.

Existem algumas vantagens na utilização desta abordagem. Primeiro, o fato de que um *script* em Pig Latin descreve um grafo direcionado faz dele uma escolha muito adequada para representar o grafo de operadores. Segundo, a linguagem Pig Latin é extensível através do uso de UDFs. Essas funções permitem a integração de bibliotecas externas e de *scripts* criados por desenvolvedores em diferentes linguagens como Java, Jython, JavaScript, Ruby, Groovy e Python. Funções criadas em uma dessas linguagens podem ser facilmente integradas ao *framework*.

Importante notar que esta camada fornece aos desenvolvedores uma maneira simples e eficiente de incluir novas funcionalidades, que serão executadas de forma distribuída, sem ter que lidar com as complexidades de se programar diretamente para MapReduce.

Outra vantagem é que durante a compilação de um *script* Pig Latin, o *framework* Pig analisa todo o fluxo de processamento em busca de possíveis otimizações antes de enviar os processos para o *cluster*. Esse procedimento geralmente leva a um melhor desempenho do que o de uma programa escrito sobre o MapReduce diretamente.

7. Camada MapReduce

A camada MapReduce representa o *framework* Hadoop e se ocupa do armazenamento distribuído dos dados, bem como da execução dos *jobs* MapReduce no *cluster*. Além disso, o Hadoop fornece mecanismos para lidar com falhas em computadores do *cluster* e para gerenciar a comunicação entre eles, proporcionando uma plataforma confiável e de alta disponibilidade para trabalhar com grandes conjuntos de dados.

8. Resultados Experimentais

Com o objetivo de validar e avaliar o desempenho da arquitetura proposta, foram executados experimentos utilizando um *cluster* virtual (em uma nuvem comercial). Uma imagem SPOT-5 de 20883x17883 pixels cobrindo a região de Nairóbi, Quênia, foi utilizada.

O experimento foi realizado para dois diferentes tamanhos de *tile* (256x256 e 512x512 pixels) com a variação da quantidade de nós do *cluster* (3, 5, 10, 20 e 40). Para o tamanho de *tile* igual a 256x256, foram gerados 5822 *tiles*; e para 512x512, foram gerados 1512 *tiles*. A interpretação consiste em um grafo com apenas um nó que realiza uma operação de segmentação. Os *tiles* são enviados para o *cluster* e representam o conjunto de dados de entrada. No *cluster*, a segmentação se dá de forma paralela internamente em cada *tile*.

Nestes experimentos, implementou-se uma versão do segmentador multiresolução proposto por Baatz e Schäpe (Baatz e Schäpe, 2000). Os segmentos de borda ainda não estão sendo tratados propriamente (i.e., a segmentação paralela gera artefatos nas bordas dos tiles), apesar de já estarmos trabalhando em algumas abordagens para resolver a questão.

É interessante também ressaltar que *no framework* Hadoop, uma das máquinas é a mestre, onde apenas o *job tracker* é executado. Assim, o processamento propriamente dito do ICP é realizado nas outras máquinas, através dos *task trackers*. Por conta disso, o paralelismo real

alcançado é sempre o número de máquinas descontada a máquina mestre. O ambiente de execução e os resultados são apresentados a seguir.

8.1 Ambiente de Execução

O ambiente de execução consiste numa combinação de serviços para processamento em *cluster* em uma nuvem comercial. Foi utilizado um repositório no Amazon S3 para armazenar os dados de entrada e de saída, bem como os programas e bibliotecas necessários para execução do sistema.

O serviço Amazon EMR (Elastic MapReduce) foi utilizado para a criação do *cluster*. Neste serviço, é possível configurar máquinas que já possuam o Hadoop e o Pig instalados. As máquinas do *cluster* podem ser configuradas no AWS de acordo com o desejo do usuário. Para os testes executados, foram utilizadas máquinas do tipo *m1.xlarge*, que são máquinas de 64 bits, *quadcore*, com 15 GB de RAM e 4 discos de 420 GB. Parâmetros de configuração do Pig e do Hadoop podem também ser facilmente alterados.

O fluxo de dados ocorre da seguinte maneira. O grafo de operadores armazena o *script* Pig no *bucket* do Amazon S3 e então esse *script* é lido pelo Amazon EMR. Ali, o *framework* Pig processa o *script* e o traduz em *jobs* MapReduce que são então executados no *cluster* de forma distribuída. Uma vez terminado o processamento, o resultado é armazenado novamente no Amazon S3.

8.2 Resultados

A figura 3 apresenta os resultados dos experimentos. É interessante notar que o tempo de processamento quase não se alterou com a mudança no tamanho do *tile*, o que sugere que o eventual ganho de processamento ao se segmentar um *tile* menor é compensado pelo *overhead* de ter que se disparar um número maior de processos.

Tomando-se o tamanho de *tile* igual a 512x512 como exemplo, temos com 3 máquinas um tempo de processamento de 4288 segundos (~1h11min). Esse tempo de processamento cai para 2243 (~37 min) quando são utilizadas 5 máquinas. Com 10 máquinas, o tempo de processamento é de 1059 (~17 min). Para 20 e 40, os tempos de processamento foram 434 (~7min) e 243 (~4 min) respectivamente.

É possível notar que nas últimas três configurações do *cluster* a inclinação da curva é reduzida. De fato, o ganho de processamento tende a diminuir gradativamente a partir de um determinado número de máquinas. O tamanho da imagem é um dos principais fatores a influenciar a localização deste ponto. Para a imagem em questão, a utilização de um número de máquinas maior do que 10 ou 20 pode ser desvantajosa. Entretanto, a utilização de uma maior quantidade de máquinas se justifica para imagens maiores.

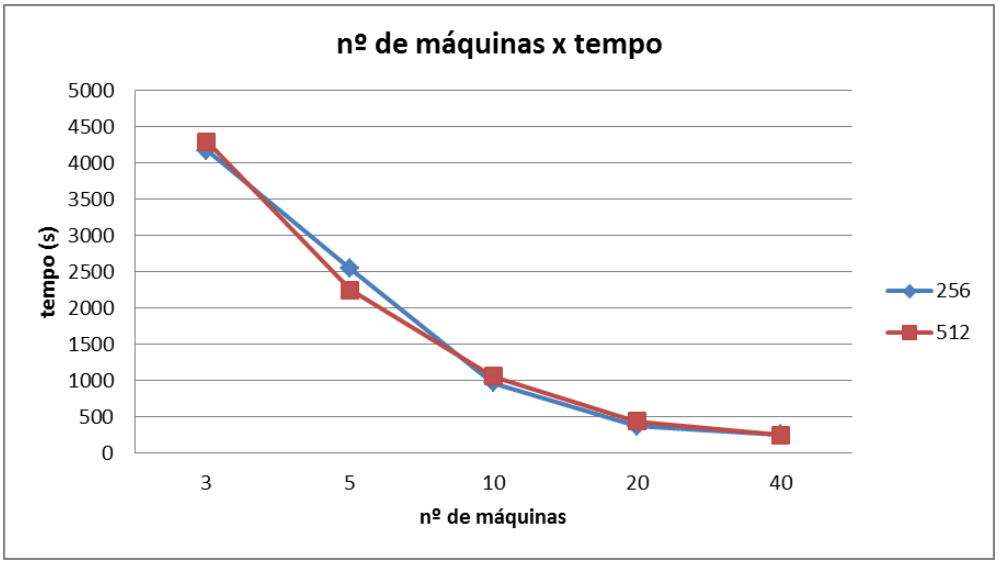

Figura 3. Número de máquinas x tempo.

9. Conclusões

Este artigo descreve de forma breve a arquitetura do ICP, uma plataforma que fornece um ambiente computacional distribuído, acoplado a uma interface gráfica intuitiva, que representa um passo adiante em direção a soluções robustas de código aberto para as demandas de grandes volumes de dados em interpretação de imagens.

Os experimentos descritos no artigo tiveram como objetivo provar o funcionamento da arquitetura proposta e demonstrar a escalabilidade da plataforma. Diferentemente das arquiteturas correntes, como a do InterIMAGE original, o sistema não apresenta um limite fixo de memória, bastando acrescentar máquinas ao cluster para tratar um maior volume de dados. Desta forma, é possível verificar o potencial escalável do ICP para reduzir o tempo de processamento quando se processa grandes volumes de dados.

Além disso, o ICP apresenta uma plataforma robusta e flexível que permite a modelagem de projetos de interpretação de imagem complexos. O especialista modela o seu conhecimento através da definição de uma rede semântica e de um grafo de operadores. Esses operadores podem tanto ser funções fornecidas por bibliotecas externas, como regras específicas, definidas pelo próprio usuário.

Uma vez definido o projeto de interpretação, a execução acontece de forma distribuída sem a necessidade de nenhum conhecimento específico do usuário sobre como o paralelismo será realizado no *cluster*.

A plataforma, que se encontra atualmente em desenvolvimento, deve funcionar como um serviço *web*, não havendo necessidade de o usuário estar de posse de uma infraestrutura de *hardware* avançada. No entanto, a plataforma também estará disponível para execução em *clusters* físicos, proprietários.

Agradecimentos

Os autores agradecem o financiamento fornecido pela FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro), pelo CNPq (Conselho Nacional de Desenvolvimento e Pesquisa) e pela CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) no escopo do Programa Ciências Sem Fronteiras, e pelo FP7 (Seventh Framework Programme) no escopo do projeto TOLOMEO.

Referências Bibliográficas

Aji A., Wang F., Vo H., Lee R., Liu Q., Zhang X., Saltz J., 2013, Hadoop-GIS: a high performance spatial data warehousing system over MapReduce. Proceedings of the VLDB Endowment, Volume 6, Issue 11: 1009-1020.

Baatz, M., Schäpe, A. 2000. Multiresolution Segmentation – an optimization approach for high quality multi-scale image segmentation. In: Strobl/Blaschke/Griesebner (editors): Angewandte Geographische Informationsverarbeitung XII, Wichmann-Verlag, Heidelberg, pp. 12-23.

Costa, G.A.O.P.; Feitosa, R.Q.; Fonseca, L.M.G.; Oliveira, D.A.B.; Ferreira, R. S.; Castejon, E. F. Knowledge-Based Interpretation of Remote Sensing Data with the InterIMAGE System: Major Characteristics and Recent Developments. In: 3rd International Conference on Geographic Object-Based Image Analysis (GEOBIA 2010), Gehnt. Enshede: ITC, 2010. v.XXXVII.

Dean, J, Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. Google Labs, OSDI; 2004 137–150.

Eldawy A., Mokbel M.F., 2013, A demonstration of SpatialHadoop: an efficient MapReduce framework for spatial data. Proceedings of the VLDB Endowment, Volume 6, Issue 12: 1230-1233.

Exner, P., Nugues, P., 2014, KOSHIK: A large-scale distributed computing framework for NLP. In 3rd International Conference on Pattern Recognition Applications and Methods (pp. 464-470).

Gates, A. Programming Pig, O'Reilly, 2011.
Hadoop. Disponível em: <http://hadoop.apache.org>. Acesso em: 09.out.2014.

Ghemawat, S., Gobioff, H., Leung, S, 2003, The Google File System. 9th ACM Symposium on Operating Systems Principles, Lake George, NY, October, 2003.

Hadoop. Disponível em: <https://hadoop.apache.org>. Acesso em: 09.out.2014.

Hive. Disponível em: <https://hive.apache.org>. Acesso em: 09.out.2014.

Pig. Disponível em: <https://pig.apache.org>. Acesso em: 09.out.2014.

Shang W., Jiang Z.M., Hemmati H., Adams B., Hassan A.E., Martin P., 2013, Assisting developers of big data analytics applications when deploying on Hadoop clouds. In Proceedings of the 2013 International Conference on Software Engineering (ICSE '13). IEEE Press, Piscataway, NJ, USA, 402-411.

Vatsavai R.R., 2013, Object based image classification: state of the art and computational challenges. In Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data (BigSpatial '13). ACM, New York, NY, USA, 73-80.

White, T. Hadoop: The Definitive Guide. O'Reilly, 2012.