Faculdade Estácio – ESTACIO – Big Data e otimização de recursos: implantando funções de MapReduce para melhorar a gestão de gastos de internet em setor público

José Alberto Perdigão Júnior*
Curso: Ciência de Dados e Big Data Analytics
Pólo: Via Corpvs
Orientador: Profª. Beatriz Acampora

RESUMO

Com a crise financeira pela qual o Brasil está passando, cada vez mais o governo lança mão de cortes de verbas e despesas em diversos setores públicos. No entanto, estes setores devem enxugar ao máximo seus gastos para que possam se manter. Este artigo sugere a aplicação de técnicas de programação paralela baseadas no paradigma de programação MapReduce para fazer a identificação de gastos de internet pela Administração Pública, além de propor uma forma de consolidação dessas informações de maneira que permita a fácil visualização de disparidades encontradas no grande volume de dados apresentados. A solução proposta foi testada em um estudo de caso executado na Procuradoria Geral do Estado do Ceará (PGE-CE). Os resultados obtidos apontam que tais técnicas se constituem de promissoras ferramentas para gestão e otimização de gastos da internet, para setores que necessitem controle do tráfego gerado pelos seus colaboradores.

Palavras-chaves: Otimização de gastos, Big Data.

1. Introdução

O Ministério do Planejamento, Desenvolvimento e Gestão (acesso em: 01 mai 2018) informou que, no primeiro semestre de 2018, as despesas de custeio administrativo do Governo Federal totalizaram R$ 3,3 bilhões. Dentre as três categorias de dispêndios com custeio que obtiveram redução de gastos, de um total de oito categorias, comparado com o ano de 2017, a categoria de Gastos com Comunicação e Processamento de Dados sofreu uma diminuição de despesas de 50,9% em termos reais, ou seja, descontando a inflação. Isso reflete o esforço do governo federal na busca da otimização do gasto público, que tem por meta continuar contribuindo para a tarefa de reduzir o custo da máquina pública.

Seguindo esse esforço do governo federal para otimização de gastos, a PGE-CE também tem buscado contribuir com esta redução de gastos. No cenário atual deste órgão a internet utilizada é paga mediante consumo trafegado, ou seja, tudo referente a acesso a sites, download de arquivos através de sites, visualização de vídeos online etc., sendo esse consumo medido pela quantidade de gigabytes trafegados. Até então a PGE-CE não possuía visibilidade detalhada de seus custos com a internet, onde informações básicas como "que setores/colaboradores mais utilizam a internet" ou "quanto está sendo gasto em acessos a determinados sites" não estavam disponíveis de forma clara aos gestores, demonstrando um descontrole com relação ao seu consumo.

A solução para esse tipo de problema pode ser viabilizada através de softwares de captura de tráfego de rede e da aplicação de técnicas de tratamentos de dados que permitam a estruturação da informação, capturada por esses softwares, de forma clara e elucidativa para os gestores. Para isso, é necessário que sejam desenvolvidas ferramentas capazes de processar o grande volume de dados gerado e que permitam uma visualização consolidada dessas informações, facilitando assim a identificação de informações que se desviem da normalidade.

Atualmente, segundo deRoos, Zikopoulos, Brown, et al. (2014) “processamentos intensivos de dados baseados na infraestrutura do Hadoop e no paradigma de programação MapReduce têm se apresentado como ótimas soluções para esses e outros tipos de problemas advindos da necessidade de processamento e tratamento de grandes volumes de dados”.

O trabalho em questão propõe uma metodologia de tratamento de informação para a obtenção de visões mais esclarecedoras sobre os gastos de internet do órgão governamental em questão. O objetivo é a aplicação de técnicas de programação baseadas no paradigma de programação MapReduce para fazer a identificação de um conjunto pré-determinado de tráfego de internet, além de propor uma forma de consolidação dessas informações de maneira que permita a fácil visualização de informações encontradas no grande volume de dados apresentados. Esse mecanismo propiciará uma melhor avaliação dos gastos do órgão, permitindo com que os gestores possam entender melhor como o órgão está empregando seus recursos.

O restante desse artigo está organizado da seguinte forma: a seção 2 faz uma revisão sobre Big Data e o modelo de programação MapReduce; já a seção 3 descreve a metodologia proposta; enquanto que as seções 4 e 5 apresentam o estudo de caso desenvolvido e os resultados obtidos. Finalmente, a seção 6 faz a conclusão do trabalho.

2. BIG DATA E O MODELO DE PROGRAMAÇÃO MAPREDUCE

> Big Data é um campo dedicado à análise, processamento e armazenamento de grandes coleções de dados que frequentemente se originam de fontes diferentes. Aborda requisitos distintos, como a combinação de vários conjuntos de dados não relacionados, o processamento de grandes quantidades de dados não estruturados e a colheita de informações ocultas de forma sensível ao tempo (ERL, KHATTAK e BUHLER, 2016, p. 19).

Os dados nos ambientes Big Data geralmente são acumulados numa empresa através de aplicativos, sensores e fontes externas, e podem ser usados por aplicativos empresariais diretamente ou podem ser armazenados em um data warehouse para enriquecer os dados existentes lá. De acordo com Erl, Khattak e Buhler (2016) “os resultados obtidos através do processamento de Big Data podem levar a uma ampla gama de *insights* e benefícios, tais como: otimização operacional, registros mais detalhados, melhoria da tomada de decisões, descobertas científicas etc.”.

O gerenciamento e processamento desses dados vêm sendo cada vez mais complexo devido a sua escala de volume e variedade. No entanto, grandes soluções e práticas de dados são normalmente requeridas quando as tecnologias tradicionais de análise, processamento e armazenamento de dados e técnicas são insuficientes.

Para suprir essa insuficiência, o MapReduce surge como um paradigma de programação que permite realizar processamento massivo de dados em paralelo através de um grande conjunto de servidores (cluster). Derivado do Google MapReduce, é altamente escalável e confiável, e é baseado no princípio de dividir-e-conquistar, que fornece tolerância a falhas internas e redundância. Ele divide um grande problema em uma coleção de problemas menores, onde cada um pode ser resolvido rapidamente, permitindo execuções duplicadas ou re-execuções das tarefas de *map* em caso de falhas ou desequilíbrio de carga, sem afetar os resultados da computação.

Segundo Tannir (2014) “o modelo MapReduce consiste de duas funções: *map* e *reduce*, definidas pelo usuário, que lidam principalmente com pares de dados chave/valor”. Essas funções são executadas sequencialmente em um cluster, e a saída da fase *map* torna-se a entrada para a fase *reduce*. A função *map* lê um par de chaves/valores, aplica o código específico do usuário e produz resultados chamados resultados intermediários. Então, esses resultados intermediários são agregados pelo código *reduce* do usuário, que emite os resultados finais. Entre as operações de Map e Reduce existe uma fase chamada “*Shuffle and Sort*”, que ocorre de forma automatizada, onde os

pares chave/valor, gerados pelo *map*, são distribuídos para o *reduce* ordenados pela chave, de forma que a função *reduce* já receba os pares de maneira ordenada pela chave.

> A entrada para um aplicativo MapReduce é organizada por registros, conforme a especificação de entrada, que produzirá pares chave/valor, cada um dos quais sendo um par <k1, v1>. A função *map* é aplicada a todos os registros de entrada um por um, e para cada registro é gerada uma lista de zero ou mais pares chave/valor intermediários, isto é, registros <k2, v2>. Então todos os registros <k2, v2> são coletados e reorganizados para que os registros com as mesmas chaves (k2) sejam colocados juntos em um registro <k2, list(v2)>. A função *reduce* é chamada uma vez para cada chave distinta do resultado do *map*, registros <k2, list(v2)>, e para cada registro, a função *reduce* gera a saída de zero ou mais <k2, v3> pares. Todos os pares <k2, v3> são reunidos ao resultado final (Tannir, 2014, p. 8).

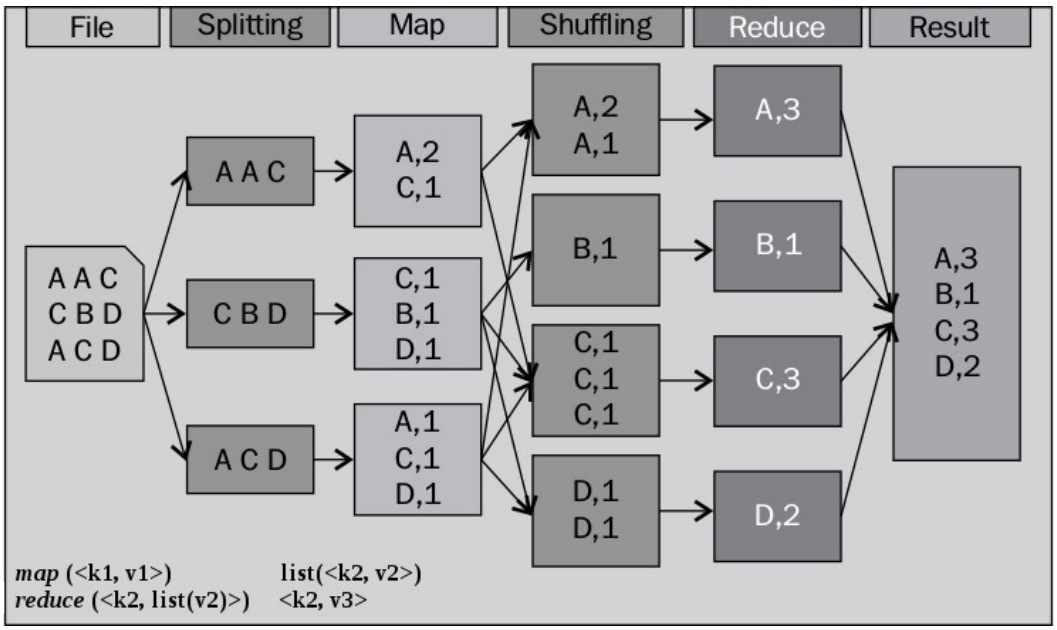

Figura 1. Modelo de execução do MapReduce

3. METODOLOGIA PROPOSTA

O objetivo da metodologia ora proposta é a utilização dos conceitos de programação MapReduce, utilizando-se de softwares de licença livre, a fim de tornar viável o processamento dos grandes volumes de dados gerados pelo programa Squid. A metodologia apresenta uma forma de analisar as descrições textuais do tráfego de rede, que são apresentados em arquivos de log, possibilitando assim a consolidação de uma série de informações a respeito desse tráfego. Como resultado, espera-se obter, a partir de informações textuais, um conjunto de dados estruturados que quantificam o tráfego gerado, sendo apresentados posteriormente através de uma *dashboard*.

A arquitetura da solução está dividida em quatro partes. A primeira é composta pela ferramenta Squid, a qual gera os logs de tráfego de internet a serem analisados. A segunda é composta pelas funções de *map* e *reduce*, que são responsáveis pela qualificação do tráfego através da aplicação de regras de identificação e pela agregação do tráfego identificado, a fim de se obter informações que tracem o perfil de uso da internet feita pela PGE-CE. A terceira parte é composta por um script Python, executado de forma opcional, que faz acesso ao servidor LDAP (*Samba – Active Directory*), previamente existente no órgão, através do utilitário *ldapsearch*, e consulta os dados referentes aos setores dos usuários a fim de se obter informações mais detalhadas. A quarta refere-se à *dashboard* que consumirá os dados obtidos pelas etapas anteriores, armazenados em um

banco de dados. A Figura 2 apresenta a arquitetura da solução proposta, sendo que, as subseções seguintes detalharão as ferramentas e as atividades representadas na figura.

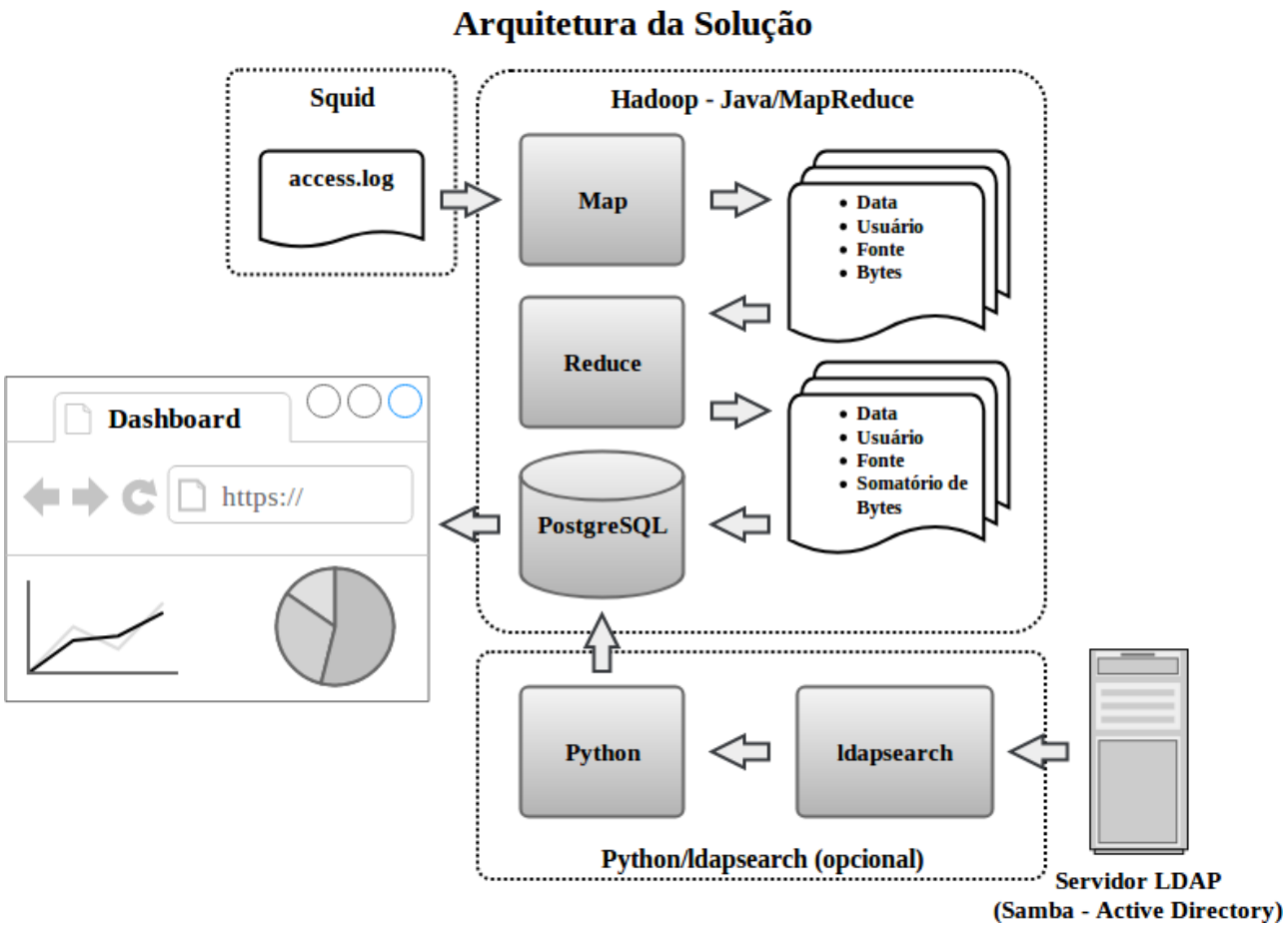

Figura 2. Arquitetura da solução proposta

3.1. Squid

> O Squid é um servidor proxy que suporta HTTP, HTTPS, FTP e outros. Ele reduz a utilização da conexão e melhora os tempos de resposta fazendo cache e reutilizando requisições frequentes de páginas web numa rede de computadores. Squid otimiza o fluxo de dados entre o cliente e o servidor para melhorar o desempenho e armazena em cache o conteúdo usado com frequência para economizar largura de banda (Time oficial do Squid. Acesso em: 02 mar. 2018).

O Squid é um software consolidado e com um histórico longo e de evolução constante. É hoje cada vez mais utilizado para auxílio na redução de custo e controle de navegação em grandes redes. Seu arquivo de log é bem estruturado e padronizado, o que facilita a extração de dados para posterior análise. Segundo o Time oficial do Squid (Acesso em: 02 mar. de 2018) "é recomendado o uso do formato de log nativo do Squid devido à maior quantidade de informações disponibilizadas para posterior análise". Portanto, uma entrada no *access.log* geralmente consiste em (pelo menos) 10 colunas separadas por um ou mais espaços:

(1) Hora (*time*) – segundos decorridos desde um *Unix epoch* (01 de janeiro de 1970). Este é o tempo quando o Squid começou a gravar a transação.

(2) Duração (*duration*) – tempo de resposta em milissegundos.
(3) Endereço do cliente (*client address*) – endereço IP do cliente.
(4) Códigos de resultado (*result codes*) – combinação entre o status das requisições do Squid e o código do status do HTTP.
(5) Tamanho (*bytes*) – tamanho da resposta encaminhada ao cliente, incluindo Cabeçalhos HTTP.
(6) Método de requisição (*request method*) – método de requisição para se obter um objeto.
(7) URL – a URL requisitada.
(8) Usuário (*user*) – pode conter a identidade do usuário para o cliente solicitante. Caso a identidade do usuário não esteja disponível, um "-" será registrado.
(9) Código de hierarquia (*hierarchy code*) – combinação do status da hierarquia do Squid e do endereço IP ou do nome do *peer* de seu cache.
(10) *Tipo* (*type*) – tipo de conteúdo do objeto, conforme informado no cabeçalho de resposta HTTP.

O Squid registra em seu log uma linha para cada objeto requisitado pelos clientes. A Figura 3 apresenta um fragmento do arquivo *access.log*.

```
1521203661.983    163 192.168.30.5 TCP_MISS/200 46808 GET http://portal.estacio.br/ alberto
HIER_DIRECT/177.184.128.202 text/html
1521203662.026     12 192.168.30.5 TCP_CLIENT_REFRESH_MISS/200 2658 GET http://portal.estacio.br/
css/menuNew.min.css alberto HIER_DIRECT/177.184.128.202 text/css
1521203662.027     13 192.168.30.5 TCP_CLIENT_REFRESH_MISS/200 1183 GET http://portal.estacio.br/
css/home_style.css alberto HIER_DIRECT/177.184.128.202 text/css
1521203662.028     12 192.168.30.5 TCP_CLIENT_REFRESH_MISS/200 1585 GET http://portal.estacio.br/
js/cookies.js alberto HIER_DIRECT/177.184.128.202 application/javascript
1521203662.029     11 192.168.30.5 TCP_CLIENT_REFRESH_MISS/200 1038 GET http://portal.estacio.br/
js/prepage.js alberto HIER_DIRECT/177.184.128.202 application/javascript
1521203662.030     18 192.168.30.5 TCP_CLIENT_REFRESH_MISS/200 7501 GET http://portal.estacio.br/
css/font-awesome.min.css alberto HIER_DIRECT/177.184.128.202 text/css
```

Figura 3. Fragmento do arquivo *access.log*

Para este trabalho foi configurado no Squid a rotação mensal de seu arquivo de log, ou seja, os registros de cada mês são armazenados em um arquivo referente àquele mês, a fim de que estes dados fossem mais bem organizados e que o tamanho do arquivo de log fosse limitado, maximizando assim seu tempo de processamento.

3.2 Hadoop

Conforme Tannir (2014) “dentre as várias implementações de MapReduce (como Esfera, Starfish, Riak, etc.), Hadoop é a implementação Java mais popular de código aberto do modelo de programação MapReduce proposto pelo Google”.

Segundo Apache Software Foundation (Acesso em: 02 mar. 2018) “a biblioteca de software Apache Hadoop é uma estrutura que permite o processamento distribuído de grandes conjuntos de dados em clusters de computadores usando modelos de programação simples”. Ele é projetado para atender milhares de máquinas, cada uma oferecendo computação local e armazenamento, além de detectar e lidar com falhas na camada do aplicativo, oferecendo assim um serviço altamente disponível em um cluster de computadores, cada um dos quais podendo ser propenso a falhas.

Ainda de acordo com Apache Software Foundation (Acesso em: 02 mar. 2018), o projeto inclui os seguintes módulos:

- Hadoop Common: os utilitários comuns que suportam os outros módulos do Hadoop.
- Hadoop Distributed File System (HDFS™): um sistema de arquivos distribuído que fornece acesso de alto throughput aos dados da aplicação.
- Hadoop YARN: uma framework para agendamentos de jobs e gerenciamento de recursos de cluster.
- Hadoop MapReduce: um sistema baseado em YARN para processamento paralelo de grandes conjuntos de dados.

Além disso, para Prajapati (2013), o Hadoop é usado de três modos diferentes:

- O modo autônomo: Neste modo, você não precisa iniciar nenhum Hadoop daemons. Em vez disso, basta chamar *~/Hadoop-directory/bin/hadoop* que irá executar uma operação Hadoop como um único processo Java. Isso é recomendado para fins de teste. Este é o modo padrão e você não precisa configurar qualquer outra coisa. Todos os daemons, como NameNode, DataNode, JobTracker e TaskTracker são executados em um único processo Java.
- O pseudo modo: Neste modo, você configura o Hadoop para todos os nós. Uma Máquina Virtual JAVA (JVM) separada é gerada para cada um dos componentes ou daemons Hadoop como mini cluster em um único host.
- O modo completo distribuído: Neste modo, o Hadoop é distribuído por várias máquinas. Hosts dedicados são configurados para os componentes Hadoop. Portanto, processos separados da JVM estão presentes para todos os daemons.

Segundo deRoos, Zikopoulos, Brown et al. (2014) "Hadoop é comumente utilizado para casos de uso envolvendo análise de logs". Para este trabalho foi utilizado o modo autônomo do Hadoop para efeito de testes. Seu código escrito em Java será descrito mais adiante.

3.2.1 Funções *map*

As funções *map* são responsáveis pela categorização das informações. Elas recebem como entrada um arquivo-texto, o *access.log* gerado pelo Squid, separando as informações recebidas e as tratando de forma individualizada.

Nessa etapa são analisados os tráfegos gerados. O objetivo dessa fase é fazer a identificação do tráfego que será processado. Essa identificação se dá através da aplicação de regras de identificação, estabelecidas por especialistas.

Ao final do processo, as funções de *map* geram registros em que cada linha possui as seguintes informações:

- Data – data do tráfego gerado. Esse campo será utilizado como chave na atividade de reducer.
- Usuário – usuário que gerou o tráfego. Esse campo será utilizado como chave, juntamente com a Data, na atividade reducer.
- Fonte – a fonte de dados consultada, como por exemplo, um site da internet. Esse campo será utilizado como chave, juntamente com a Data e o Usuário, na atividade reducer.
- Bytes – quantidade de bytes trafegados. Esse campo será utilizado para o cálculo dos valores estatísticos do tráfego, durante a fase de "*reduce*".

Como apresentado na seção 2, a função de *map* deve gerar pares de chave/valor, para serem passados para a função de *reduce*. No caso da solução proposta, a chave é representada pela data (dia/mês/ano) juntamente com o nome do usuário e a fonte consultada, e o campo valor é formado pela quantidade de Bytes, cabendo à função *reduce* separar, interpretar e processar esses dados.

3.2.2 Funções *reduce*

As funções r*educe* recebem como entrada os registros gerados pelas funções de *map* e são responsáveis por fazer os cálculos estatísticos, que indicarão o tráfego gerado pelos usuários.

Nessa fase do processamento, a Data, Usuário e Fonte são agrupados, permitindo assim o cálculo de algumas métricas sobre os tráfegos identificados. A saída gerada por essa fase é composta por um conjunto de linhas cuja configuração é apresentada abaixo:

- Data – data do tráfego gerado.
- Usuário – indica o usuário que gerou o tráfego.
- Fonte – indica a fonte de dados consultada.
- Bytes – indica o somatório de bytes gerados pelo Usuário/Data.

Essas informações fornecem um parâmetro de referência sobre a quantidade de tráfego gerado na rede da instituição, tanto de internet no geral, quanto de sites específicos quando configurados, permitindo que possam ser feitas comparações entre os gastos realizados diariamente, mensalmente e anualmente por todos os usuários.

3.3 Python/ldapsearch

> Python é uma linguagem de programação interpretada de alto nível para programação de propósito geral. Criado por Guido van Rossum e lançado pela primeira vez em 1991, o Python possui uma filosofia de design que enfatiza a legibilidade do código e uma sintaxe que permite que os programadores expressem conceitos em menos linhas de código, principalmente usando espaços brancos significativos. Ele fornece construções que permitem uma programação clara em escalas pequenas e grandes. Possui um sistema de tipo dinâmico e gerenciamento automático de memória, além de suportar múltiplos paradigmas de programação, incluindo orientado a objetos, imperativo, funcional e processual, e possuir uma ampla e abrangente biblioteca padrão (Python Software Foundation. Acesso em: 07 mar. 2018).

Segundo Kalb (2016) “Python é amplamente usado como uma linguagem de script por administradores de computadores, que a usam para capturar e reproduzir sequências de comandos em computadores”. “Por ser uma linguagem de fácil leitura, com uma curva de aprendizado e tempo de desenvolvimento reduzidos, quando comparado com linguagens como C/C ++ e Java” de acordo com Mueller (2018), Python demonstrou ser uma ótima ferramenta no apoio a criação de scripts.

O utilitário *ldapsearch* é uma interface *shell* utilizada para localizar e recuperar entradas de diretório. Ele abre uma conexão ao servidor especificado usando o nome e senha distinta especificados e localiza entradas com base em um filtro de pesquisa específico. Ele emite solicitações de pesquisa para um diretório Lightweight Directory Access Protocol (LDAP) e exibe o resultado como texto LDIF Data Interchange Format (LDIF). Suas muitas opções permitem executar diferentes tipos de operações de busca, desde a recuperação de entrada simples até buscas avançadas que envolvem referências de segurança ou diretório.

Este utilitário foi utilizado, juntamente com o Python, com o propósito de recuperar os dados dos colaboradores cadastrados na rede LDAP (*Samba – Active Directory*) já existente, possibilitando assim a obtenção de seus setores, auxiliando na geração de informações referentes aos setores do órgão.

3.4 PostgreSQL

Para The PostgreSQL Global Development Group (acesso em: 07 mar. 2018),

> o PostgreSQL é um poderoso sistema de banco de dados relacional *free open source*. Possui mais de 15 anos de desenvolvimento ativo e uma arquitetura comprovada que lhe valeu uma forte reputação de confiabilidade, integridade de dados e correção. Ele é compatível com ACID (Atomicidade, Consistência, Isolamento, Durabilidade) e possui integridade transacional. É executado em todos os principais sistemas operacionais, incluindo Linux, UNIX (AIX, BSD, HP-UX, macos, Solaris) e Windows. Possui interfaces de programação nativas para C/C ++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outros, e documentação excepcional.

O PostgreSQL foi utilizado para se obter o controle das operações MapReduce sobre o access.log e armazenar as informações geradas.

Quando um *job* do Hadoop é executado, antes das funções de *map* e *reduce* serem executadas, é verificado no banco de dados os parâmetros necessários para a execução do *job* (custo, fonte etc) e o último registro do arquivo *access.log* processado, a fim de que, a cada execução de um *job,* não sejam processados registros que já foram utilizados em *jobs* executados anteriormente.

Na função de *map* é feito o controle dos registros lidos do arquivo, onde ao seu final é guardado na memória o tempo do último registro lido pela função, e ao final do *job* esse tempo é cadastrado no banco de dados.

Na função de *reduce* é feito no banco de dados o cadastro do tráfego calculado para os usuários.

Todas as operações no banco de dados são executadas em um bloco de transação para cada *job* executado, sendo comitado ao final do *job*. Caso ocorra algum erro na transação, ou falha no sistema, o banco de dados efetuará um *rollback* das informações cadastradas, garantindo a integridade dos dados.

3.5 Dashboard

Segundo Wikipédia (acesso em: 09 abr. 2018) "*dashboard* refere-se a uma tela sob a forma de um painel, composta de instrumentos virtuais onde são associadas variáveis a serem monitoradas, além de gráficos que mostram a evolução destas variáveis, fornecendo uma representação ilustrada do desempenho dos negócios em toda a organização".

Responsável por consumir os dados do banco de dados e gerar as informações necessárias à gestão. Neste trabalho, é composta de duas linguagens principais, o PHP e o JavaScript, que recuperam as informações do banco de dados e as apresentam em forma de tabelas, gráficos etc., em um *web* s*ite*. A seguir são apresentadas as características destas duas linguagens.

Para The PHP Group (Acesso em: 09 abr. 2018) "PHP (um acrônimo recursivo para *PHP: Hypertext Preprocessor"*, originalmente *Personal Home Page*) é uma linguagem de script projetada especificamente para o desenvolvimento web. É focado em atuar no lado do servidor e capaz de gerar conteúdo dinâmico, podendo ser embutido dentro do HTML". De acordo com Ullman (2011) "seu propósito principal é de implementar soluções web velozes, simples e eficientes, tendo como características principais a velocidade e robustez, a portabilidade, a facilidade de aprendizado e uso, além de ser *open-source* e ser a ferramenta mais popular disponível para construção de sites dinâmicos".

Para Ullman (2012),

> *JavaScript* é uma das linguagens de programação mais usadas atualmente, encontrada em quase todas as páginas da Web. Foi concebida para ser uma linguagem de programação script com orientação a objetos baseada em protótipos, de tipagem fraca e dinâmica, implementada como parte dos navegadores web para que scripts pudessem ser executados

do lado do cliente. Tem como propósito primário ser uma tecnologia do lado do cliente para melhorar a exeperiência do usuário.

3.6 O Processo de execução

Inicialmente foi criado um *script* (*script_trafego_internet.sh*, seção 4.2.6) contendo todos os passos para a execução da solução apresentada. Então este *script* foi adicionado ao agendador de tarefas do Sistema Operacional (*cron*), a fim de que ele fosse executado de tempos em tempos, gerando dados em um curto espaço de tempo, com o intuito de tornar a informação o mais atual possível.

Para que o processo de execução seja iniciado, é necessário ter o ambiente previamente preparado efetuando os seguintes passos:

(1) Efetuar a carga do arquivo *access.log* pelo programa Squid*,* contendo as informações dos tráfegos de internet. Após devidamente configurado, este processo é realizado sempre que algum usuário utiliza a internet.
(2) Gerar o arquivo *.jar* (programa Java) a ser utilizado pelo Hadoop, contendo as funções de *map* e r*educe*.
(3) Inserir os registros necessários no banco de dados, descritos na seção 4.2.4, para o funcionamento correto das funções de *map* e *reduce*.

Realizado o preparo do ambiente, o *script* é executado e é dado o início ao processamento dos dados. O arquivo *access.log* é enviado ao Hadoop, o qual é processado. Em seguida é executado o arquivo "*.jar"* pelo Hadoop sobre o arquivo processado, onde serão executados os *jobs* contendo as funções de *map* e *reduce* para cada registro desse arquivo.

Na execução dos *jobs* são analisados os textos descritivos de cada tráfego, onde são aplicandas regras de identificação de tráfego, a fim de identificar o tráfego de internet, de sites específicos e de economia gerada. Inicialmente a configuração foi desenvolvida para gerar relatórios de gastos com acesso à internet como um todo, gastos com acesso a sites como Facebook, Youtube e Instagram, podendo ser configurado para outros sites específicos, além da economia gerada, identificada pelo acesso ao cache e/ou bloqueio aos sites.

Após esta etapa é executado o *script* Python/ldapsearch para atualizar o setor de cada usuário cadastrado no banco de dados. Ele consiste de uma consulta à base LDAP através do utilitário *ldapsearch,* retornando os setores dos usuários cadastrados na rede LDAP. Logo após, faz uma comparação destes usuários com os já cadastrados no banco de dados e atualiza seus setores, caso não possuam cadastrados. A execução do *script* Python/ldapsearch torna-se opcional, visto que foi somente utilizado para se obter os setores dos usuários para enriquecer as informações a serem apresentadas, podendo ser omitido ou substituído por outra forma de se obter os setores.

Neste momento os dados já estarão armazenados no banco de dados prontos para serem consultados pela *dashboard*, que através de consultas SQL simples, por meio da linguagem de programação PHP, retorna as informações desejadas em forma de gráficos em uma web site.

Toda a estrutura de arquivos e diretórios utilizados será mostrada na seção seguinte.

4. ESTUDO DE CASO

Para testar a metodologia proposta, foi aplicado um estudo de caso nos dados de tráfego de rede gerados pelo programa Squid na Procuradoria Geral do Estado do Ceará (PGE-CE).

Esta seção demonstra o estudo de caso desenvolvido e está dividida em duas subseções, a primeira apresenta a infraestrutura utilizada no estudo de caso, enquanto que a segunda descreve a implantação das tecnologias utilizadas.

4.1 Infraestrutura utilizada

O estudo de caso foi desenvolvido no ambiente de computação da própria PGE-CE. O ambiente era composto por um servidor, virtualizado pelo RHEV (Red Hat Virtualization), com Linux Ubuntu 14.04.5 LTS, um Processador Intel Core i7 2.4 GHz com oito núcleos, 8 GB de memória, 21 GB de armazenamento em disco SAS de 7200 RPM e plataforma de 64 bits.

Optou-se por essa estrutura, pois de acordo com deRoos, Zikopoulos, Brown et al. (2015) esta estrutura contempla os recursos de software e hardware necessários para a execução de aplicações que utilizam Hadoop.

4.2 Implantação das tecnologias utilizadas

O trabalho em questão utiliza as seguintes tecnologias: Squid, Java, Apache Hadoop, PostgreSQL, PHP/JavaScrip, Python e ldapsearch, sendo essas duas últimas opcionais, pois referem-se unicamente ao cadastro de setores dos usuários do órgão. A seguir são listadas as instalações das ferramentas no sistema operacional Linux Ubuntu, assim como os códigos da solução.

4.2.1 Squid

> Squid está disponível em vários formatos (arquivos fonte compactados, código fonte em um sistema de controle de versão, pacotes binários, como RPM, DEB e assim por diante) em seu site oficial (http://www.squid-cache.org/), vários espelhos Squid em todo o mundo e repositórios de software de quase todos os sistemas operacionais populares. Squid também é fornecido por muitas distribuições Linux / Unix (Saini, 2011, p. 9).

Instalação:

```
$ sudo apt-get update
$ sudo apt-get install squid3
```

4.2.2 Java

> Java é usado em bilhões de dispositivos em todo o mundo. De aplicativos móveis a softwares de desktop, o Java capacita as maiores empresas e os menores dispositivos pessoais. Alunos, profissionais de TI e qualquer um que esteja considerando uma carreira em programação descobrirão que precisam aprender Java (Payne, 2018, p. xix).

Segundo Oracle (Acesso em: 11 abr. 2018) “a plataforma Java possui dois componentes principais: a API (Application Programming Interface) Java, que é uma biblioteca de linhas de comandos Java, e a Java Virtual Machine (JVM) que interpreta o código Java na linguagem de

máquina". Java é independente de plataforma, gratuito e está disponível no seguinte endereço: https://www.java.com/pt_BR/ (Acesso em: 11 abr. 2018).

Instalação:

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
$ java -version
        java version "1.8.0_151"
        Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
        Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

Os códigos de *map* e *reduce* em Java podem ser visualizados no repositório de códigos *GitHub https://github.com/AlbertoPerdigao/tcc-ciencia_de_dados-alberto_perdigao* na pasta "Projeto Java-MapReduce". Os parâmetros de conexão com o banco de dados foram omitidos, devendo ser configurados de acordo com a infraestrutura utilizada.

4.2.3 Hadoop

Abaixo segue sua instalação detalhada, tendo como pré-requisito o Java 6, ou uma versão posterior, previamente instalado. Hadoop está disponível em http://hadoop.apache.org/releases.html#Download (Acesso em: 11 abr. 2018).

```
$ wget http://ftp.unicamp.br/pub/apache/hadoop/common/hadoop-2.8.3/hadoop-2.8.3.tar.gz
$ tar -xvf hadoop-2.8.2.tar.gz
$ sudo mv hadoop-2.8.2 /usr/local/hadoop

$ readlink -f /usr/bin/java | sed "s:bin/java::"
        /usr/lib/jvm/java-8-oracle/jre/
$ vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh
        . . .
        #export JAVA_HOME=${JAVA_HOME}
        export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
        . . .
$ /usr/local/hadoop/bin/hadoop version
        Hadoop 2.8.3
        . . .
```

4.2.4 PostgreSQL

O PostgreSQL pode ser baixado no seguinte endereço: https://www.postgresql.org/download/ (Acesso em: 11 abr. 2018). A seguir é demonstrado sua instalação.

```
$ sudo apt-get install postgresql-9.6 postgresql-client-9.6 postgresql-contrib-9.6 libpq-dev
```

Antes da criação dos objetos no banco de dados será mostrado o diagrama entidade-relacionamento das tabelas para maior entendimento da aplicação. O modelo é descrito na Figura 4.

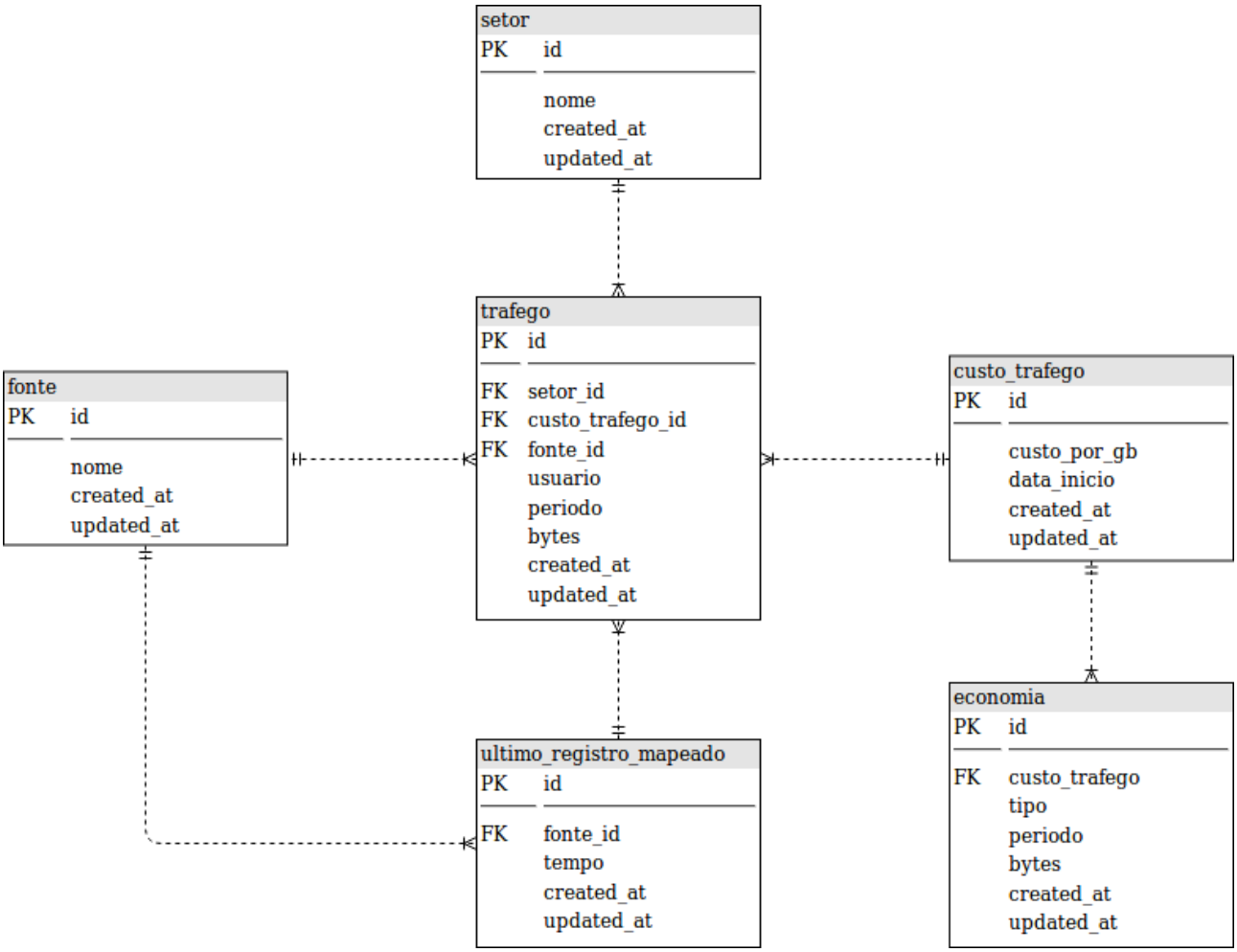

Figura 4. Diagrama entidade-relacionamento

Como visto na Figura 4, o banco de dados da aplicação é composto por seis tabelas, a serem descritas:

- trafego – tabela que armazena os dados do tráfego, como o usuário que gerou o tráfego, o período do tráfego, a quantidade de bytes trafegados, o setor do usuário, o custo do tráfego e a fonte consultada (site).
- custo_trafego – armazena o custo por gigabytes do tráfego a partir de uma data de início. O custo do tráfego é calculado respeitando a vigência do custo. Sempre será utilizado o custo com a maior data de início dentro do período corrente. Esta tabela tem a necessidade de ser inserido um custo inicial para o correto funcionamento da aplicação.
- economia – tabela que armazena os dados do tráfego com relação à economia, ou seja, todo tráfego que foi gerado com acesso ao cache ou bloqueado. É definida pelo custo do tráfego, o tipo de acesso (acesso ao cache ou acesso bloqueado), o período do tráfego e a quantidade de bytes trafegados.
- ultimo_registro_mapeado – armazena o tempo do último registro mapeado pela aplicação. Esta tabela é utilizada para que a cada execução da aplicação sejam computados apenas os valores dos registros cujo tempo seja maior que o cadastrado. A cada execução da aplicação

é armazenado o tempo do último registro mapeado pelo Hadoop, com o intuito de que na próxima execução, os registros sejam computados a partir do último registro da execução anterior, evitando assim a repetição de processamento dos registros e recadastro de valores.

- fonte – refere-se às fontes de pesquisa, como a internet como um todo, a economia de internet, e os sites consultados. Os seguintes valores devem estar previamente cadastrados para o correto funcionamento da aplicação: “Internet”, “Internet Economia” e “Outras Fontes”. Estes parâmetros servem para computar o tráfego e para controle da aplicação.
- setor – armazena os setores da empresa.

O script de criação dos usuários, tabelas e schema no banco de dados encontra-se no repositório *GitHub https://github.com/AlbertoPerdigao/tcc-ciencia_de_dados-alberto_perdigao* na pasta “Script Banco de Dados”.

4.2.5 Utilitário ldapsearch

Sua documentação encontra-se no endereço https://docs.oracle.com/cd/E19957-01/816-6400-10/lsearch.html (Acesso em: 11 abr. 2018). Abaixo segue sua instalação.

```
$ apt-get install ldap-utils
```

4.2.6 Python

O Python pode ser baixado do endereço https://www.python.org/ (Acesso em: 11 abr. 2018). Segue sua instalação:

```
$ sudo apt-get install python2.7
```

Instalação do psycopg2 (módulo de acesso ao postgres no python):

```
$ sudo apt-get install python-psycopg2
```

O código do script Python para cadastrar os setores dos usuários no banco de dados pode ser visto no repositório G*itHub https://github.com/AlbertoPerdigao/tcc-ciencia_de_dados-alberto_perdigao* na pasta “Script Python”.

4.2.7 PHP/JavaScript

Para que a dashboard possa ser visualizada em um site web é necessário realizar a instalação de um seridor web. Para este trabalho foi decidido utilizar o Apache (download em https://www.apache.org/. Acesso em: 11 abr. 2018). Abaixo segue sua instalação, assim como a do PHP, disponível em https://secure.php.net/ (acesso em: 11 abr. 2018).

```
$ sudo apt-get update
$ sudo apt-get install apache2
$ sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

Os códigos em PHP/JavaScript para criação da *dashboard* são apresentados no repositório G*itHub* *https://github.com/AlbertoPerdigao/tcc-ciencia_de_dados-alberto_perdigao* na pasta "Dashboard PHP-JavaScript". Para que fosse obtida uma melhor apresentação visual foi utilizado um *template* (download em https://www.creative-tim.com/product/light-bootstrap-dashboard) baseado na ferramenta Bootstrap – uma *toolkit* de código aberto para desenvolvimento com HTML, CSS (*Cascading Style Sheets*) e JavaScript (https://getbootstrap.com/. Acesso em: 11 abr. 2018). Os parâmetros de conexão com o banco de dados foram omitidos, devendo ser configurados de acordo com a infraestrutura utilizada.

4.2.8 Criação dos diretórios, arquivos de apoio e script de execução

A Figura 5 mostra a estrutura de arquivos e diretórios utilizada.

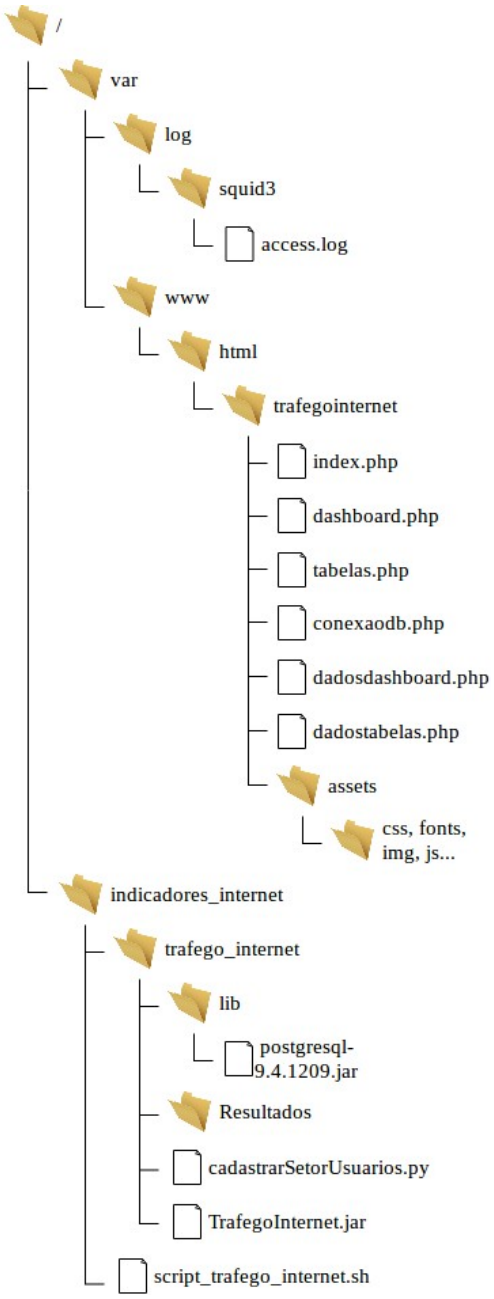

Figura 5. Estrutura de arquivos e diretórios

O script *script_trafego_internet.sh*, o qual executa todo o processamento da solução e armazena o resultado no banco de dados, pode ser visualizado no repositório G*itHub https://github.com/AlbertoPerdigao/tcc-ciencia_de_dados-alberto_perdigao* na pasta "Script Execuçao".

A seguir é demonstrado o cadastro do script no agendador de tarefas (Cron), definido para ser executado de quinze em quinze minutos:

```
$ crontab -e
        */15 * * * * /indicadores_internet/script_trafego_internet.sh
```

5. RESULTADOS

O processo analisou o total de quatro arquivos de logs gerados nos meses de janeiro a abril de 2018, sendo cada arquivo composto em média de 30 milhões de registros textuais e 4 GB de tamanho.

Foi averiguado que no quinto dia de cada mês o tempo gasto em média para se concluir o processamento do access.log, com 5 milhões de registros e 0,7 GB de tamanho, foi de 6 minutos. No décimo quinto dia de cada mês o tempo gasto em média para se concluir o processamento, com 15 milhões de registros e 2 GB de tamanho, foi de 8 minutos. Já ao final de cada mês, com o arquivo completo, o tempo registrado foi em média de 20 minutos. Esses valores foram obtidos com apenas um processamento para cada arquivo, ou seja, sendo processados todos os registros existentes em cada arquivo nos intervalos de datas citados.

Com o agendamento do script *script_trafego_internet.sh* (seção 4.2.8) para ser rodado de quinze em quinze minutos, os registros a serem processados para cada execução diminuíram consideravelmente, tendo em média neste intervalo de tempo um total de aproximadamente 42 mil registros, logo o tempo de processamento também diminuiu. No quinto dia de cada mês, cada processamento efetuado (com intervalos de 15 minutos) foi concluído num tempo médio de 2 minutos (arquivo com média de 0,7 GB de tamanho). No décimo quinto dia de cada mês, cada processamento efetuado foi concluído com um tempo de 6 minutos em média (arquivo com média de 2 GB de tamanho). Já no último dia de cada mês, cada processamento efetuado foi concluído num tempo médio de 15 minutos com o arquivo de tamanho de 4 GB em média.

Processamento único do access.log

Dia	Tamanho (GB)	Quantidade de registros processados (milhões)	Tempo (minutos)
5	0,7	5	6
15	2	15	8
30	4	30	20

Processamento do access.log com intervalos de 15 minutos

Dia	Tamanho (GB)	Quantidade de registros processados (mil)	Tempo (minutos)
5	0,7	42	2
15	2	42	6
30	4	42	15

Figura 6. Tabelas contendo os tempos de execução da solução

O aumento gradual do tamanho do arquivo access.log aumentou o tempo de processamento da aplicação. Contudo os resultados demonstraram a viabilidade da solução proposta, onde os dados apresentados puderam ser atualizados em intervalos de quinze minutos, disponibilizando informações à gestão da PGE em um intervalo de tempo satisfatório para suas tomadas de decisão.

Como resultado do processamento dos logs mensais, puderam ser obtidas, através da dashboard (Figuras 6 e 7), as seguintes informações:

- Top 5 de gastos com a internet ou outras fontes por setor e por usuário (podendo ser filtrados por setor, fonte, dia, mês e ano). Nesses gráficos são apresentados os cinco setores e/ou usuários que possuem o maior gasto.
- Histórico mensal de gastos com a internet e outras fontes (podendo ser filtrado por ano).
- Histórico diário de gastos com a internet e outras fontes (podendo ser filtrado por mês e ano).
- Economia gerada pelo bloqueio a sites e acesso ao cache (podendo ser filtrados por dia, mês e ano). Apresentam as informações referentes a todos os setores.
- Histórico mensal de economia (podendo ser filtrado por ano).
- Tabelas de gasto com a internet ou outras fontes por setor e por usuário (podendo ser filtradas por setor, fonte, dia, mês e ano). Nelas são mostrados os gastos juntamente com as quantidades de gigabytes trafegados por cada setor e/ou usuários.

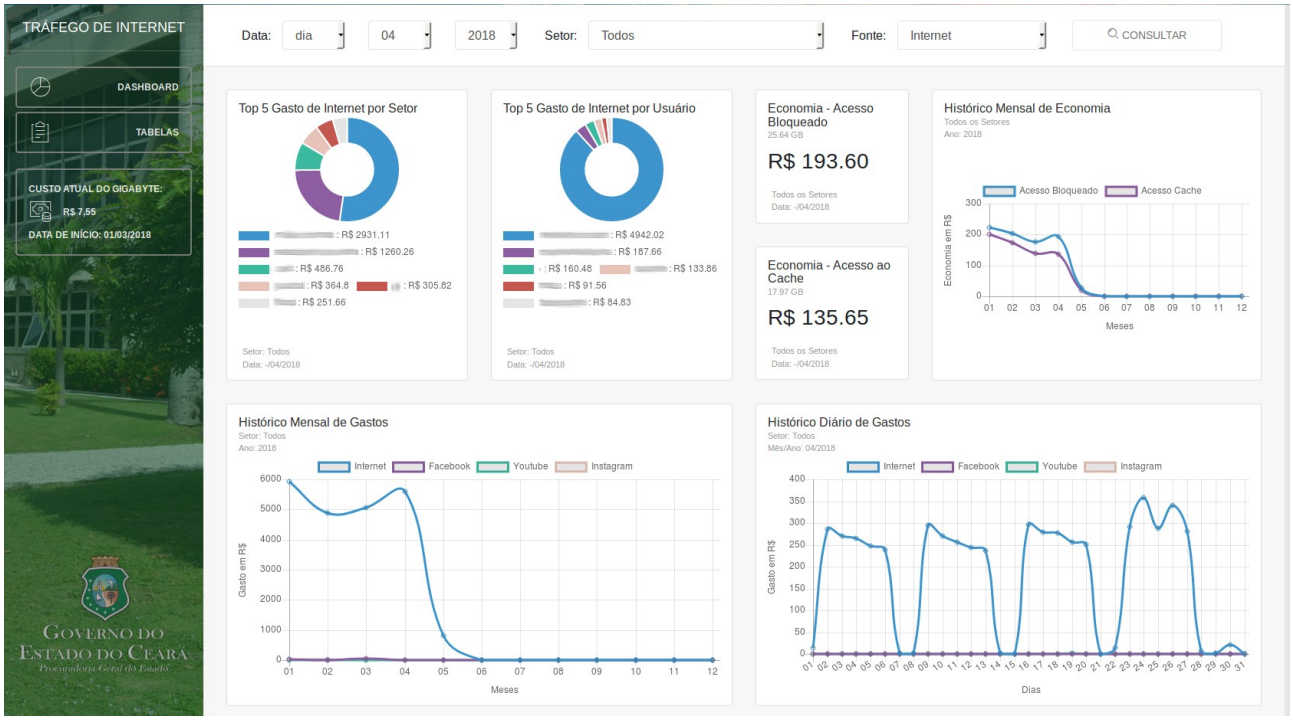

Figura 6. Dashboard – gráficos

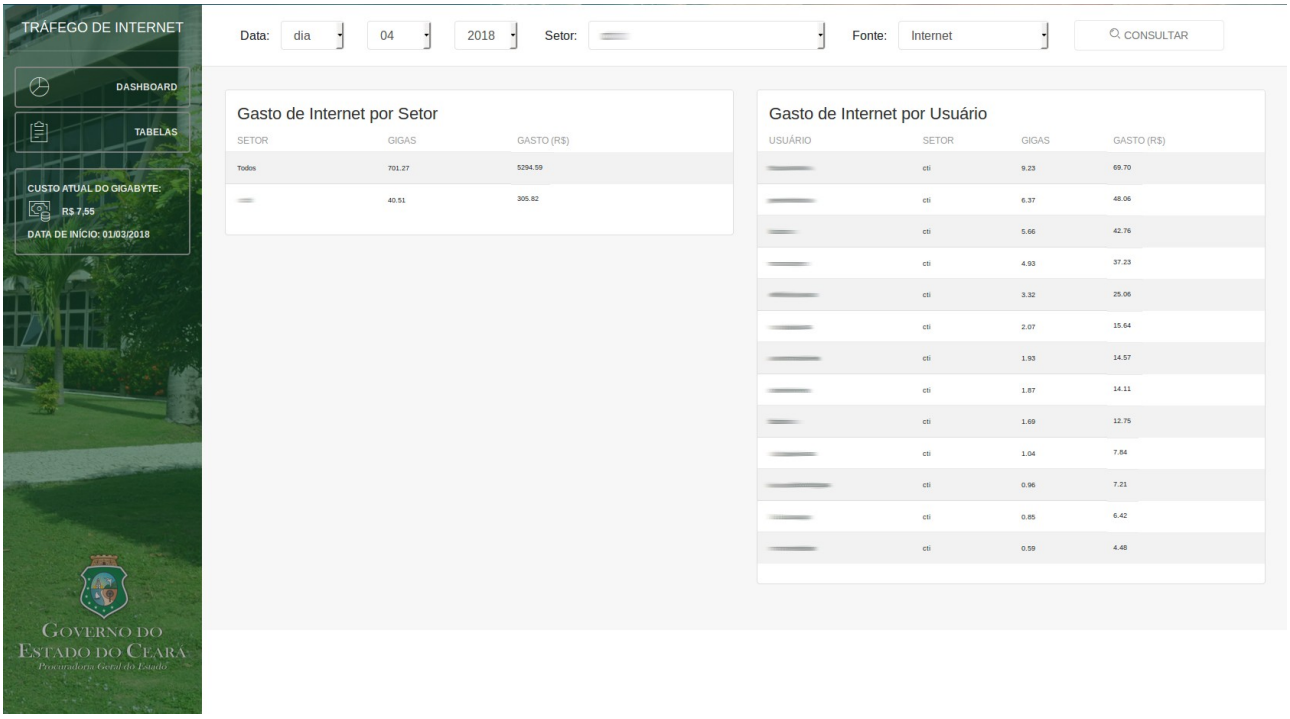

Figura 7. Dashboard – tabelas

Em outros cenários, caso as informações necessitem ser apresentadas de forma ainda mais rápida, uma possível solução para o aumento de desempenho da solução, referente ao aumento do tamanho do arquivo access.log, seria configurar a rotação do arquivo access.log para um tempo menor que um mês, como, por exemplo, para 15 dias, onde, de acordo com os testes aplicados, seu tamanho, e consequentemente seu tempo de execução, diminuiriam pela metade. Outra possível solução seria utilizar o modo completo distribuído do Hadoop, onde servidores adicionais dividiriam a carga de dados entre eles, diminuindo o tempo processamento.

6. CONCLUSÃO

Este trabalho apresentou uma metodologia para se obter informações relevantes a gastos de internet em órgão público.

A solução proposta tinha como objetivo a identificação e quantificação do tráfego de internet gerado na infraestrutura de rede da PGE-CE.

A metodologia sugerida faz uso de técnicas de processamento de dados com a utilização do paradigma de programação MapReduce através do sistema de processamento Hadoop.

A proposta apresentada foi testada para um conjunto de milhões de registros, reunidos em arquivos referentes a cada mês do ano, decorrentes de acessos à internet gerados pelos usuários da rede da PGE-CE. Esse conjunto de dados foi composto por registros capturados no ano de 2018 entre os meses de janeiro a maio.

Como resultado, obteve-se uma série de informações relacionadas aos gastos de internet, os quais auxiliaram os gestores em suas tomadas de decisões. A metodologia proposta também permitiu a identificação da economia gerada quando um site era acessado diretamente do cache ou bloqueado, não fazendo a utilização da internet, o que permitiu uma melhor configuração do Squid para otimização dos acessos, diminuindo os gastos.

Pode-se concluir que a utilização de técnicas de processamento de dados através do paradigma de programação MapReduce são ferramentas de grande valia para questões ligadas à

área de gerência, que normalmente requerem informações rápidas, mais apuradas e de qualidade, oriundas de vários meios diversos.

Como trabalhos futuros, pretende-se desenvolver novas técnicas para obtenção de informações mais apuradas que permitam a identificação dos sites mais acessados pelos usuários/setores da PGE. Além disso, pretende-se utilizar o modo completo distribuído do Hadoop, onde os dados poderão ser distribuídos para hosts diferentes, aumentando assim o poder de processamento e diminuindo o tempo de execução da solução.

7. REFERÊNCIAS

ERL, Thomas; KHATTAK, Wajid; BUHLER, Paul. **Big Data Fundamentals** Concepts, Drivers & Techniques. United States of America: Prentice Hall, 2016.

TANNIR, Khaled. **Optimizing Hadoop for MapReduce** Learn how to configure your Hadoop cluster to run optimal MapReduce jobs. Birmingham, UK: Packt Publishing, 2014.

DEROOS, Dirk; ZIKOPOULOS, Paul C.; BROWN, Bruce et al. **Hadoop for Dummies**. New Jersey: John Wiley & Sons, Inc., 2014.

TIME OFICIAL SQUID. Disponível em: <http://www.squid-cache.org/>. Acesso em: 02 mar. 2018.

APACHE SOFTWARE FOUNDATION. Disponível em: <http://hadoop.apache.org/>. Acesso em: 02 mar. 2018.

PRAJAPATI , Vignesh. **Big Data Analytics with R and Hadoop** Set up an integrated infrastructure of R and Hadoop to turn your data analytics into Big Data analytics. Birmingham, UK: Packt Publishing, 2013.

PYTHON SOFTWARE FOUNDATION. About. Disponível em: <https://www.python.org/>. Acesso em: 07 mar. 2018.

MUELLER, John Paul. **Beginning Programming with Python for dummies**. 2nd ed. Hoboken, New Jersey: John Wiley & Sons, Inc. 2018.

KALB, Irv. **Learn to Program with Python**. California, USA: Apress, 2016.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. About. Disponível em: <http://www.postgresql.org/>. Acesso em: 07 mar. 2018.

WIKIPEDIA. Disponível em: <https://pt.wikipedia.org/wiki/Painel_de_bordo>. Acesso em: 09 abr. 2018.

THE PHP GROUP. Disponível em: <https://secure.php.net/>. Documentation. Acesso em: 09 abr. 2018.

ULLMAN, Larry. **PHP for the Web**. Fourth Edition. Berkeley, CA: Peachpit Press. 2011.

ULLMAN, Larry. **Modern JavaScript** Develop an Design. Berkeley, CA: Peachpit Press. 2012.

SAINI, Kulbir. **Squid Proxy Server 3.1** Beginner's Guide Improve the performance of your network using the caching and access control capabilities of Squid. Birmingham, UK: Packt Publishing, 2011.

ORACLE. Java. Disponível em: <https://www.oracle.com/br/index.html>. Acesso em: 11 abr. 2018.

SUN MICROSYSTEMS, INC. The ldapsearch Tool. Disponível em: <https://docs.oracle.com/cd/E19957-01/816-6400-10/lsearch.html>. Acesso em: 11 abr. 2018.

PAYNE, Bryson. **Learn Java the Easy Way** A Hands-On Introduction to Programming. San Francisto, CA: No Starch Press, 2018.

BRASIL. Ministério do Planejamento, Desenvolvimento e Gestão. **Governo divulga boletim trimestral de custeio administrativo**. Disponível em: <http://www.planejamento.gov.br/noticias/governo-divulga-boletim-trimestral-de-custeio-administrativo>. Acesso em: 01 mai. 2018.