XII CONGRESSO DE INICIAÇÃO CIENTÍFICA DA UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

PROPEX
PRÓ-REITORIA DE PESQUISA E EXTENSÃO

PIBITI/CNPq-UFCG 2015

PULSAROPENSTACK-AD - INTRODUÇÃO AO PROCESSAMENTO DISTRIBUÍDO DE DADOS UTILIZANDO O HADOOP E OPENSTACK (SAHARA): UMA VISÃO DO USUÁRIO

Marianne Linhares Monteiro[1], Andrey Elísio Monteiro Brito [2]

RESUMO

Diante da rápida evolução tecnológica, constantemente surgem novos dispositivos com capacidade de processamento e comunicação e, consequentemente, novas fontes de dados. Desta forma, uma questão cada vez mais em voga é a necessidade de processamento de grandes massas de dados de maneira eficiente, acessível e simples. Nesse sentido, a ferramenta de Big Data Hadoop se encontra em ascensão. Além da utilização direta do Hadoop, projetos como o Sahara do OpenStack se mostram-se uma excelente alternativa tanto para usuários mais experientes, quanto para novos utilizadores, oferecendo todo o poder de processamento do Hadoop associado às vantagens do armazenamento em nuvem, uma interface amigável, fácil escalonamento e configuração automática das máquinas virtuais. Neste trabalho, foi feito um estudo a respeito do Hadoop, OpenStack e OpenStack Sahara buscando introduzir essas tecnologias de maneira sucinta e eficaz, resultando tanto em uma referência para a introdução de um usuário a essa tecnologia, como também em um modelo de aplicação para recebimento e armazenamento contínuo de dados para um futuro processamento.

Palavras-chave: big data, computação em nuvem, map reduce.

PULSAROPENSTACK-AD - *DISTRIBUTED PROCESSING OF DATA USING HADOOP AND OPENSTACK(SAHARA): AN USER'S POINT OF VIEW*

ABSTRACT

Given the quick technological evolution, constantly new smart devices appear and consequently new source of data. Therefore an important question is the need of processing big amount of data in a simple, accessible and efficient mode. In these ways the use of the Big Data tool Hadoop is increasing and the new project from OpenStack called Sahara is an excellent choice for both the experient user and the beginner user, offering all the power processing of the most known big data tools associated with the advantages of cloud computing, a friendly interface, easy configuration and scalability. A study was made on Hadoop, OpenStack and OpenStack Sahara targeting to introduce these technologies in a summarized and efficient way, resulting in the production of an user's introduction reference and in an application model for receiving and storage continuous amount of data for a later processing.

Keywords: big data, cloud computing, map reduce.

[1]Aluna do Curso de Ciência da Computação, Departamento de Sistemas e Computação, UFCG, Campina Grande, PB, e-mail: marianne.monteiro@ccc.ufcg.edu.br
[2]Ciência da Computação, Professor Doutor, Departamento de Sistemas e Computação UFCG, Campina Grande, PB, e-mail: andrey@dsc.ufcg.edu.br

INTRODUÇÃO

Devido ao avanço e utilização cotidiana de tecnologias de informação existe uma variedade imensa de dados sendo produzidos de maneira tão rápida quanto se pode imaginar. Os dados mundiais estão crescendo 40% ao ano até a próxima década. Esses dados são criados pelas mais de 2 bilhões de pessoas no mundo, milhões de empresas trabalhando online, e pelos milhões de sensores e aparelhos de comunicação enviando e recebendo informações por toda a internet (TURNER, 2014).

Somando a esse fato o entendimento de que a tecnologia da informação é um recurso importante que permite executar os processos, baixar custos e controlar atividades (SANCHEZ, 2012), hoje, a criação e utilização de uma grande quantidade de dados se estende muito além de grandes companhias como Yahoo, Google e Facebook. Empresas e instituições de vários ramos, enfrentam a pressão de serem competitivas e recorrem a estratégias envolvendo o processamento e obtenção de um grande volume de dados (GE INTELLIGENT PLATFORMS, 2012).

Assim, é notável o aumento da busca e necessidade de ferramentas de Big Data, que podem ser definidas de maneira resumida como tecnologias que possibilitam criar, gerir e manipular dados caracterizados pelo grande volume, variedade e rápido crescimento. Uma destas ferramentas é o Hadoop, um arcabouço de código aberto desenvolvido em Java que utiliza a computação paralela e distribuída para garantir escalabilidade, alta velocidade de processamento e baixo custo

Ao mesmo que tempo que os dados crescem cada vez mais, outra vertente da TI se desenvolve, a computação na nuvem. Há várias definições distintas para a computação na nuvem, um conceito é que as nuvens são grandes repositórios de recursos virtualizados, tais como hardware, plataformas de desenvolvimento e software, que são facilmente acessíveis. Além disto, segundo Vaquero (2008), citado por Borges (2011), estes recursos podem ser configurados dinamicamente de modo a ajustar-se a diferentes cargas de trabalho com a intenção de otimizar sua utilização.

Segundo Dai (2013), serviços de nuvem público irão valer cerca de 40 bilhões de dólares em 2020, além do que a maioria das empresas busca em primeiro lugar, maior agilidade e velocidade, em segundo, custo em um serviço de nuvem. Desta maneira, uma tecnologia em ascensão nesse campo é o OpenStack, um sistema operacional de nuvem de código aberto, que oferece ao usuário controle de grandes conjuntos de recursos computacionais para processamento, armazenamento e comunicação, através de interface web e API's bastante amigáveis e de fácil utilização.

No gráfico 1 podemos observar o crescimento da procura por essas duas tecnologias. O gráfico compara o interesse nessas duas tecnologias com o interesse em uma tecnologia madura chamada Message Passing Interface (MPI)[3].

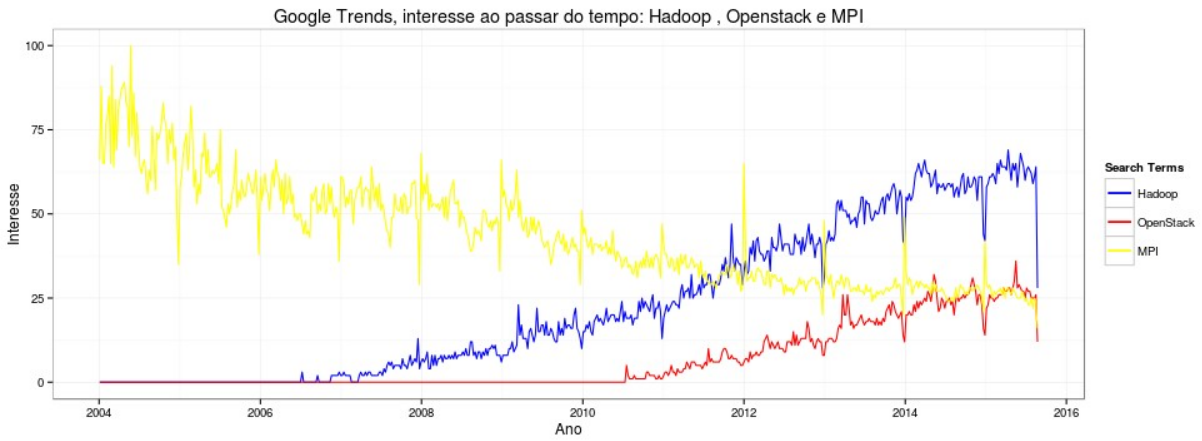

Gráfico 1 - Modelo gráfico que representa o interesse nas tecnologias Hadoop, OpenStack e MPI no Google.

[3]Message Passing Interface (MPI) é uma especificação para comunicação de dados em computação paralela.

Como a procura por ambas as tecnologias se encontra em crescimento e ambas tem características parecidas, porém com vantagens distintas, uma ótima solução para o processamento de Big Data seria utilizar o Hadoop em conjunto com o OpenStack. Essa é justamente a ideia que deu origem ao projeto do OpenStack chamado Sahara.

O Sahara, de maneira resumida, busca prover ao usuário um cluster Hadoop no OpenStack, unindo o que as duas tecnologias tem de melhor, de modo que o resultado final é uma ferramenta capaz de processar grande quantidade de dados através de uma interface de fácil utilização, abstraindo detalhes desnecessários, como configuração interna do usuário. Porém, por ser uma tecnologia recent, há pouco material disponível o que dificulta a iniciação de um novo utilizador.

Este trabalho tem como objetivo introduzir um futuro usuário Hadoop e OpenStack Sahara a estas ferramentas, visando diminuir o tempo de adaptação, além de apresentar um exemplo de aplicação e de códigos que podem ser reutilizados. Este exemplo detalha não só como uma aplicação Hadoop funciona, mas também todo o fluxo de dados, da leitura até o armazenamento dos dados e resultados na nuvem.

REVISÃO BIBLIOGRÁFICA

Hadoop

O Hadoop é um framework de software livre que permite através de um modelo simples de programação, o modelo MapReduce, executar aplicações distribuídas que processam grande quantidade de dados. Foi desenvolvido para ser escalável e ao invés de depender do hardware para garantir alta disponibilidade, o próprio Hadoop tem o poder de detectar e se recuperar de falhas na aplicação (Apache Hadoop).

As vantagens do Hadoop são as seguintes.

- Escalabilidade: petabytes de dados podem ser processados sem problemas.
- Economia: já que o processamento ocorre através máquinas de baixo custo.
- Eficiência: pelo fato do processamento ser distribuído, torna-se extremamente rápido.
- Confiável: há cópias dos dados e remanejamento automático de tarefas em caso de falhas.
- Código aberto: constante correção de erros, apoio de grandes empresas.

O projeto inclui os seguintes módulos (Apache Hadoop).

- Hadoop Common: utilidades comuns para o suporte de outros módulos.
- Sistema de arquivos distríbuidos Hadoop (HDFS) : um sistema de arquivos distribuídos que provê (high-troughput) o uso de muitas fontes computacionais durante um longo perído de tempo para aplicações.
- Hadoop YARN: um arcabouço responsável pelo gerenciamento de recursos disponíveis no cluster.
- Hadoop MapReduce: um sistema baseado no YARN para processamento paralelo de dados.

Neste trabalho, como o foco é a utilização da ferramenta iremos tratar apenas de como é organizado o processamento e armazenamento tratando em conjunto HDFS e o MapReduce que são os dois principais módulos que formam o Hadoop.

O HDFS foi desenvolvido para ser escalável, tolerante a falhas, garantir alto gerenciamento, confiabilidade, usabilidade e desempenho, além de ser capaz de trabalhar em conjunto com MapReduce. O interessante é que ao desenvolver uma aplicação não é necessário se preocupar onde os dados estão, o próprio Hadoop se encarrega dessa tarefa, ou seja, para a aplicação isso é transparente, seria como se os dados estivessem armazenados localmente.

A organização do Hadoop é baseada na arquitetura *Master-Slave* (Mestre-Escravo), onde uma máquina atua como mestre e as demais são escravos (ou trabalhadores). O mestre tem um módulo chamado *NameNode* que controla os arquivos que estão sendo processados e em quantas partes cada arquivo foi dividido, e um *JobTracker* que aceita os jobs MapReduce, manda as tarefas aos escravos, monitora as tarefas e reage a falhas. Nos escravos existem o *DataNode* e o *TaskTracker* responsáveis respectivamente por armazenar e processar os dados. Esta organização está ilustrada na figura 1.

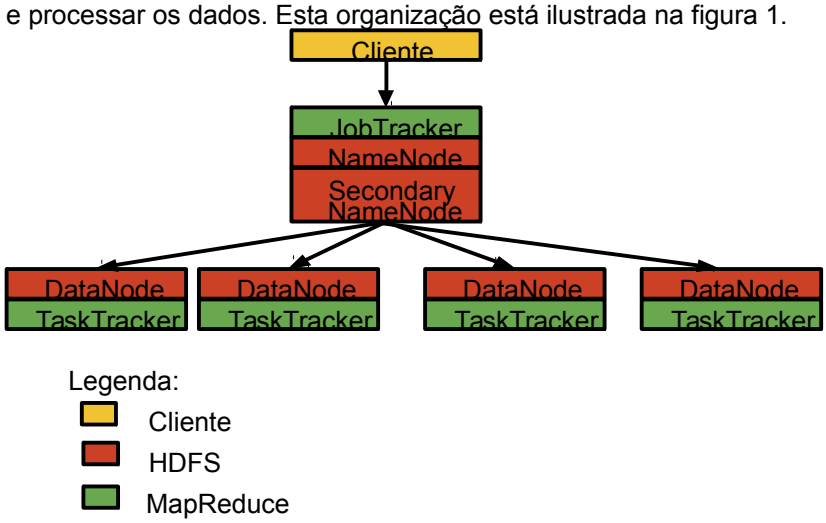

Figura 1 – Resumo da organização do Hadoop

Outro ponto importante para o entendimento da ferramenta é a definição do modelo de programação MapReduce que tem como objetivo facilitar a utilização, já que assim o Hadoop irá cuidar dos detalhes internos, como por exemplo particionamento da entrada e monitoramento de máquinas, além de sua estrutura deixar claro que o processamento é distribuído.

Neste paradigma, o processamento se divide em uma função *map* e uma funçao *reduce*, ambas devem ser definidas pelo usuário. A função *map* recebe um par do tipo *(chave,valor)* e gera um conjunto intermediário de par *(chave,valor)*, a função *reduce* recebe esses pares intermediários e agrega todos os pares que se referem a mesma chave (DEAN, 2004).

A função *map* pode ser executada independentemente em cada par *(chave,valor)*, evidenciando o paralelismo que envolve esse paradigma. De forma similar, a função reduce pode ser executada independentemente de cada chave intermediária (DEAN, 2004).

Para melhor entendimento, segue o exemplo clássico do MapReduce, um contador de palavras e sua respectiva implementação em pseudo-código no modelo convencional estruturado e no MapReduce, além de um diagrama com o exemplo de uma execução no Hadoop. A aplicação é bem simples, porém bastante útil: um arquivo de texto será recebido da entrada e a saída deverá ser a quantidade de aparições de cada palavra do arquivo.

No modelo de programação estruturada, o procedimento é o demonstrado na figura 2 e o pseudocódigo está representado na figura 3.

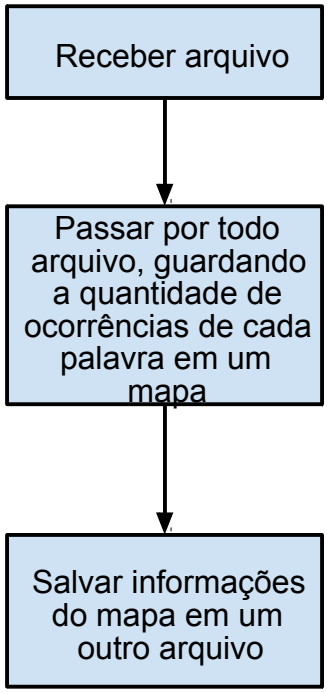

Figura 2 – Diagrama que representa a execução de um aplicação para contar palavras em um arquivo de texto seguindo o paradigma estruturado

```
arquivo_entrada = receberArquivo()

for palavra in arquivo_entrada:
        mapa[palavra]++

arquivo_saida = SalvarInformacoes(mapa)
```

Figura 3 – Pseudocódigo de uma aplicação para contar palavras em um arquivo de texto seguindo o paradigma estruturado.

Já no modelo de programação MapReduce, o procedimento seria o demonstrado na figura 4 e o pseudocódigo está representado na figura 5.

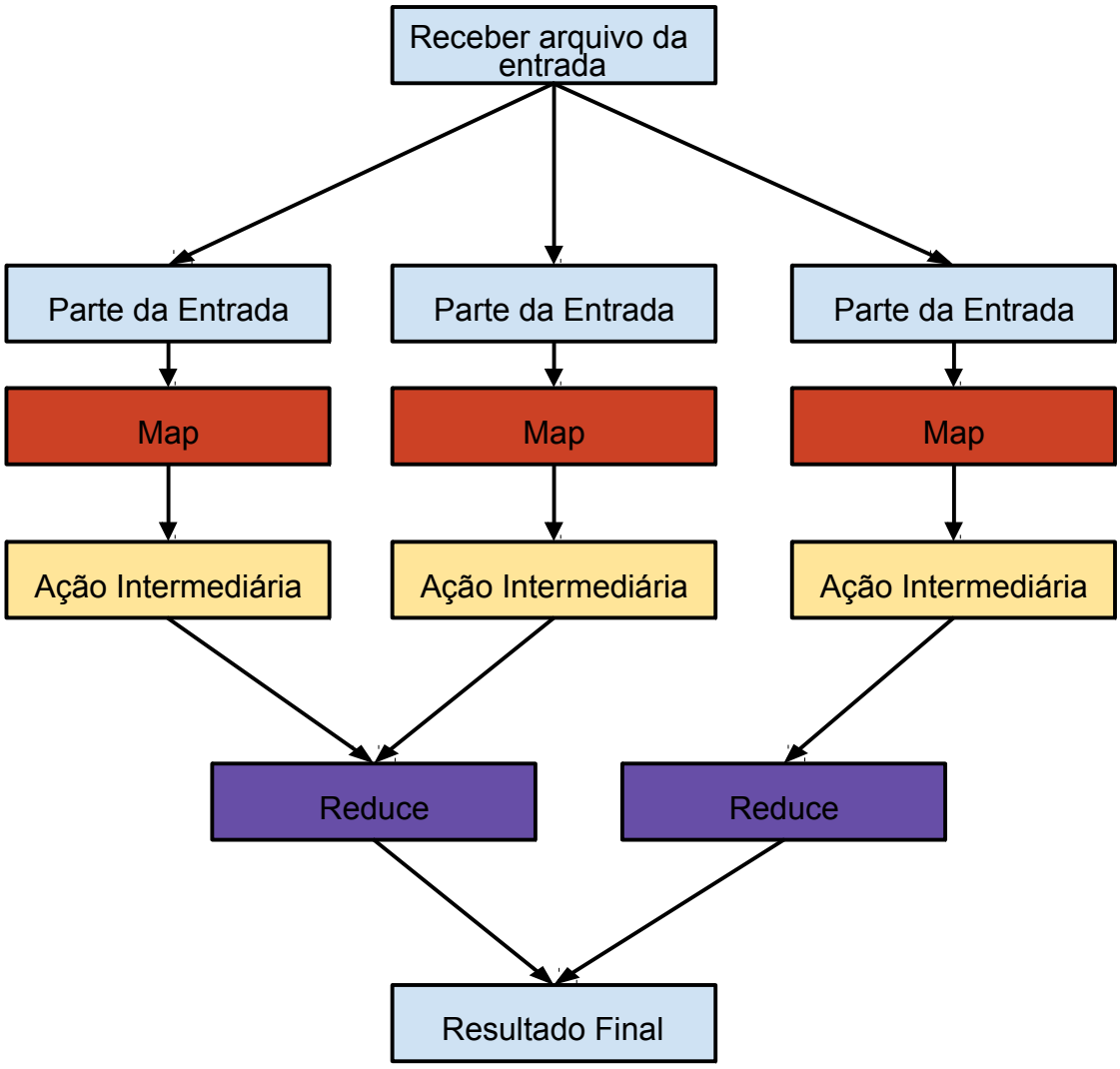

Figura 4 – Diagrama que representa os momentos principais de um aplicação para contar palavras em um arquivo de texto seguindo o paradigma MapReduce.

```
Map(chave, valor) {
   for palavra in valor:
       coletar(palavra, 1)
}

Reduce(chave, valores) {
   int soma_de_aparicoes = 0;
   for valor in valores:
       soma_de_aparicoes += valor
   coletar(chave, valores)
}
```

Figura 5 – Pseudocódigo de uma aplicação para contar palavras em um arquivo de texto seguindo o paradigma MapReduce.

Uma demonstração da execução de uma tarefa MapReduce que conta palavras é representada pelo diagrama na figura 6.

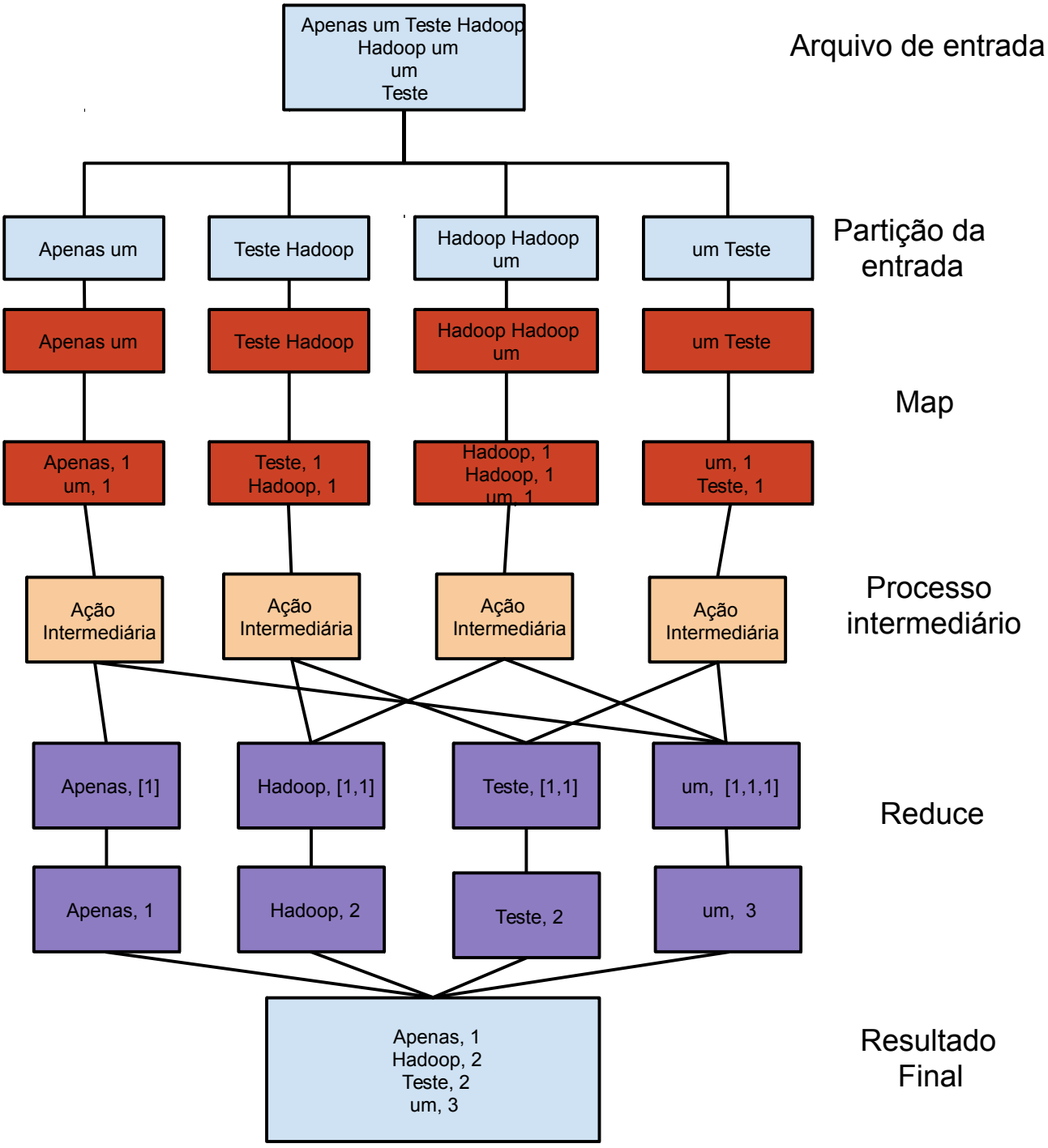

Figura 6 – Diagrama que representa execução de uma tarefa MapReduce que conta palavras de um arquivo de texto.

Kafka

Kafka é um serviço de mensagens distribuído, particionado e replicado. Este sistema provê funcionalidades de um sistema de comunicação, porém com um conjunto de funcionalidades como armazenamento temporário, o qual permite que o consumidor não esteja conectado enquanto o produtor publica mensagens, mas ainda consiga posteriormente recebê-las, e organização por tópicos, seguindo a abordagem *Publish/Subscribe*[4], onde vários produtores podem publicar mensagens em um tópico e todos os consumidores registrados naquele tópico recebem essas mensagens. Kafka também garante escalabilidade, confiabilidade e alta velocidade de processamento.

As principais nomenclaturas específicas do Kafka são:

- Mensagem: um array de bits ordenado.
- Tópico: as mensagens são armazenadas em tópicos.
- Produtor: escreve as mensagens nos tópicos.
- Consumidor: lê as mensagens dos tópicos.

OpenStack

O projeto OpenStack é uma plataforma de computação na nuvem de código aberto para todos os tipos de nuvem, cujo objetivo é ter fácil utilização, extremamente escalável e repleto de diversas ferramentas auxiliares. O OpenStack provê uma solução de Infraestrutura-como-um-Serviço através de um conjunto inter-relacionado de serviços, onde cada um oferece uma API que facilita sua integração (OpenStack).

Na quadro 1 estão os principais serviços e projetos do OpenStack relevantes para este trabalho:

Serviço	Projeto	Descrição
Dashboard	Horizon	Provê um portal web que oferece fácil iteração com os outros serviços
Computação	Nova	Gerencia as máquinas virtuais
Networking	Neutron	Permite comunicação-como-um-serviço para máquinas virtuais
Armazenamento de objetos	Swift	Armazena e recupera objetos via uma API baseada em HTTP
Identificação	Keystone	Provê autenticação e autorização para os outros projetos do OpenStack
Serviço de imagem	Glance	Armazena e recupera imagens de disco de máquinas virtuais
Processamento de dados	Sahara	Provê criação e escalonamento de clusters Hadoop (entre outras ferramentas de processamento de dados)

Quadro 1 – Quadro com os principais serviços do OpenStack referentes a este trabalho.

[4] *Publish/Subscribe* é um padrão de mensagens onde quem envia a mensagem é chamado *publisher* e as mensagens não são enviadas diretamente para seu destino. Quem recebe as mensagens é chamado de *subscriber*.

OpenStack Sahara

O projeto Sahara do OpenStack tem como objetivo oferecer configuração rápida, auto-instalação confiável, escalonamento de cluster e execução de jobs Hadoop. Tais tarefas podem ser executadas tanto através da interface web, o Horizon, como também pela API (MIRANTIS).

A principal vantagem em utilizar o Sahara é o ganho de tempo para criar e escalar clusters, e executar jobs Hadoop, criar manualmente um cluster Hadoop require, instalar o Hadoop em cada instância, configurar a comunicação entre instâncias, especificar *NameNode*, *JobTracker*, *TaskTracker* e *DataNodes*. Utilizando o Sahara todo esse processo é feito de maneira bem mais rápida, onde pode-se simplesmente definir um grupo de nós previamente configurados e lançar o cluster rapidamente (MIRANTIS).

Para melhor entendimento das principais funções realizadas no Sahara na figura 7 é apresentado um diagrama simplificado que apresenta os passos para a criação de um cluster e na figura 8 são representados os passos para executar uma tarefa MapReduce.

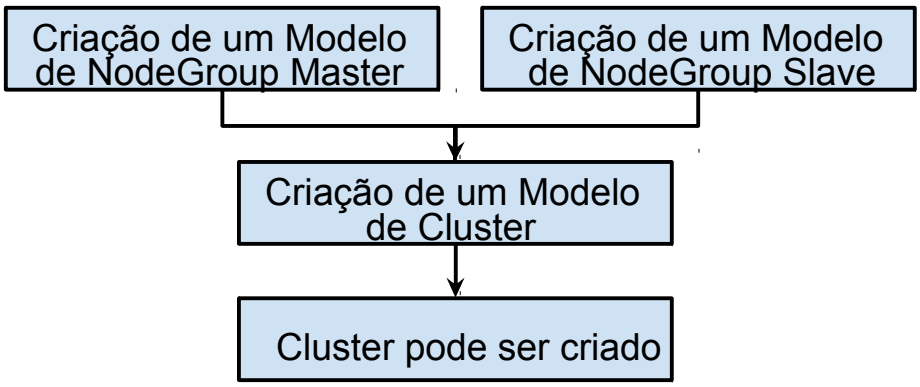

Figura 7 – Diagrama que representa os estados para a criação de um cluster Hadoop no OpenStack Sahara

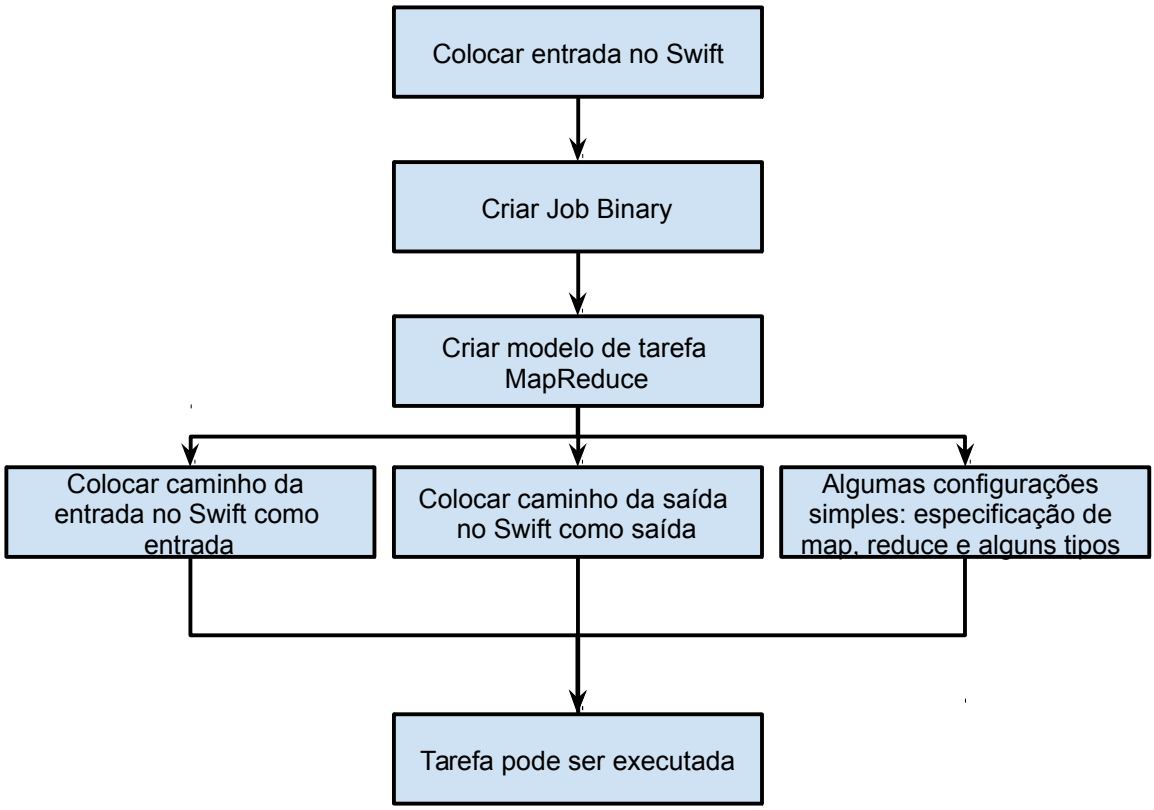

Figura 8 - Diagrama que representa os estados para a execução de uma tarefa MapReduce no OpenStack Sahara

MATERIAIS E MÉTODOS

Diagrama do trabalho

O diagrama da figura 9 apresenta o estudo e as atividades desenvolvidas a respeito das ferramentas abordadas para que este trabalho fosse realizado. Além do estudo das ferramentas, aplicações exemplo foram desenvolvidas, e tutorias apresentados.

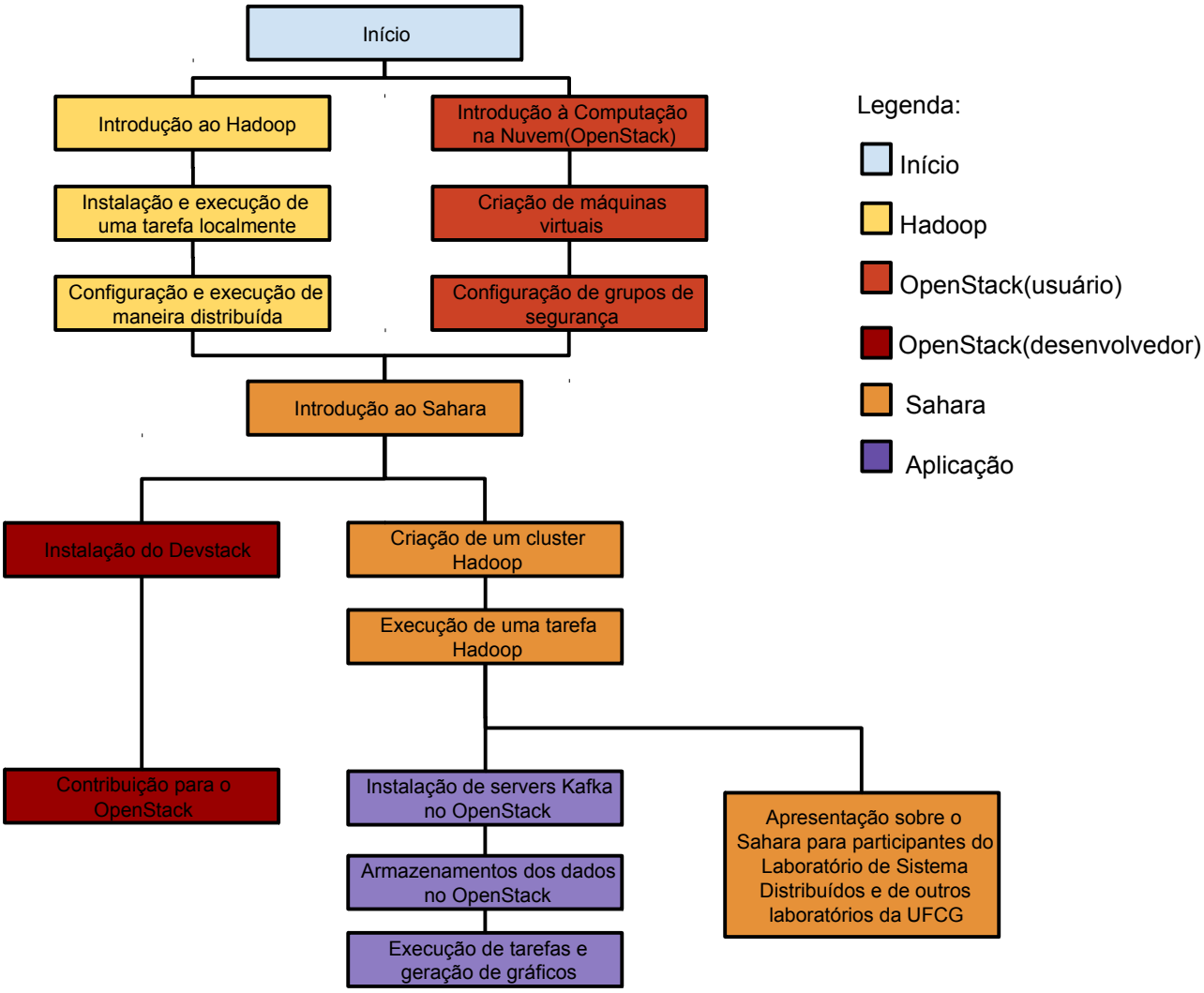

Figura 9 – Diagrama do trabalho desenvolvido para a realização deste estudo.

Detalhes da aplicação

A aplicação consiste em aproveitar dados que estão sendo coletados para um projeto do LSD, chamado SET (Smart Energy Tracker), com o objetivo de um futuro processamento para obter informações úteis, a partir de uma grande quantidade de dados de consumo de energia.

O objetivo do SET é viabilizar a implantação de tecnologias de *smart grids* no Brasil através da adaptação dos medidores existentes e do desenvolvimento de uma plataforma que permitirá recomendações de eficiência energética de forma detalhada e personalizada não só para empresas e indústrias, mas também para a população em geral.

Há alguns sensores coletando continuamente dados de consumo energético e enviando esses dados para um servidor, para obtenção e transporte de informações entre as máquinas foi utilizado o Kafka. Há um produtor recebendo as medições dos sensores e enviando as mensagens para dois tópicos diferentes, onde a mensagem enviada segue o seguinte padrão: “id do sensor; grandeza medida; tempo Unix; valor da medição”, em um dos tópicos, além destas informações há também o nome do aparelho em que a medição ocorreu.

Existem dois consumidores, um para cada tópico, que leem e salvam a mensagem em um arquivo com a data do dia. Ao final do dia o arquivo será colocado no OpenStack Swift, onde poderá ser processado quando necessário. Para esta aplicação foram desenvolvidos três tarefas MapReduce: um contador de mensagens por hora, um contador de mensagens por minuto e um MapReduce que calcula a média das medições por minuto. Como exemplo a figura 9 e figura 10 apresentam uma aplicação decomposta em seus principais métodos, o *map*, o *reduce* e o *main*. Os exemplos completos podem ser encontrados em https://github.com/mari-linhares/Relatorio-PIBITI-FUNTTEL-2015 .

```
        /* Método map: recebe a chave e tipo dos valores especificados na classe Map, um Coletor<tipo_da_chave_de_saída,
        tipo_do_valor_de_saída> e um Reporter. Perceba que o método não tem retorno, no fim do map a mensagem será
coletada.*/
        public void map(LongWritable key, Text value,OutputCollector<Text, IntWritable> output, Reporter reporter)
                                                        throws IOException {
                        /* Em cada map iremos receber uma mensagem e processá-la para retirar a informação requerida */
                String line = value.toString();
                String[] parts = line.split(";");

                        /* Obtemos o id da mensagem, e convertemos o tempo da mensagem para uma formatação mais visual */
                String id = parts[0];
                 long unix = Long.parseLong(parts[2], 10);
                Date date = new Date(unix);
                   SimpleDateFormat sdf = new SimpleDateFormat("HH:MM");
                   sdf.setTimeZone(TimeZone.getTimeZone("GMT-4"));
                   String hour_and_minute = sdf.format(date);

                /* A saída que será coletada será algo:
                                chave: id hora:minuto que mensagem foi enviada pelo sensor
                               valor: quantas vezes vimos a mensagem(1)
                                Sendo a chave e valor dos tipos especifacos na classe. */
                word.set(id + " " + hour_and_minute);
                output.collect(word,one);
}

/*      Método      reduce:      recebe      a      chave      e      um      iterator      dos      tipo      especificados      na      classe      Reduce,      um
        Coletor<tipo_da_chave_de_saída, tipo_do_valor_de_saída> e um Reporter. Perceba que o método não tem retorno,
    no fim do reduce a mensagem será coletada.      */
        public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter)
                                                        throws IOException {
                        /* Como estamos interessados no número de ocorrência das mensagems dessa chave específica iteramos
        no valor do iterator acumulando o valor, para depois coletarmos com os tipos especificados na classe Reduce. */
                int sum = 0;
                 while (values.hasNext()) {
                                sum += values.next().get();
                }
                   output.collect(key, new IntWritable(sum));
}
```

Figura 9 – Código Java do *map* e *reduce* de uma das tarefas desenvolvidas para a aplicação do SET.

```
/* Configuração do Job para utilização das classes feitas acima */
        public static void main(String[] args) throws Exception {

                /* Criando um novo JobConf com a classe principal que criamos */
                JobConf conf = new JobConf(MessageCounterPerMinute.class);
                conf.setJobName("MessageCounterPerHour");

                /* Configuração referente ao que será coletado no Reduce */
                conf.setOutputKeyClass(Text.class);
                conf.setOutputValueClass(IntWritable.class);

                /* Configuração referente ao que será coletado no Map */
                conf.setMapOutputKeyClass(Text.class);
                conf.setMapOutputValueClass(IntWritable.class);


               /* Configuração do Mapper, Combiner e Reducer. Neste caso o Combiner é a própria
                  classe Reduce */
                  conf.setMapperClass(Map.class);
                  conf.setCombinerClass(Reduce.class);
                  conf.setReducerClass(Reduce.class);

                 /* Configuração referente aos tipos de arquivo de entrada e saída, neste caso
                  arquivos de texto */
                  conf.setInputFormat(TextInputFormat.class);
                  conf.setOutputFormat(TextOutputFormat.class);

                  /* Adiciona um caminho a lista de entradas do job mapreduce */
                 FileInputFormat.setInputPaths(conf, new Path(args[0]));
                  /* Adiciona um caminho a lista de saídas do job map-reduce */
                  FileOutputFormat.setOutputPath(conf, new Path(args[1]));

                  /* Execução do job */
                  JobClient.runJob(conf);

        }
}
```

Figura 10 - Código Java do *main* de uma das tarefas desenvolvidas para a aplicação do SET.

RESULTADOS E DISCUSSÕES

Com este trabalho foi obtida uma introdução breve, porém eficaz da utilização das tecnologias Hadoop e OpenStack Sahara, além de aplicar tais conhecimentos e resultar em um modelo para uma aplicação para recebimento e armazenamento contínuo de dados para um futuro processamento. Os resultados específicos foram os seguintes:

1. Exemplo de coleta de dados automatizada do Kafka e inserção destes na nuvem;
2. Exemplos de tarefas de processamento de dados usando Hadoop;
3. Tutoriais, na forma de slides, para execução de tarefas Hadoop no OpenStack Sahara;
4. Exemplos de scripts de visualização para resultados das tarefas de processamento.

As figuras 11, 12 e 13 ilustram saídas de execuções das aplicações exemplo em formato de gráfico para facilitar o entendimento dos resultados.

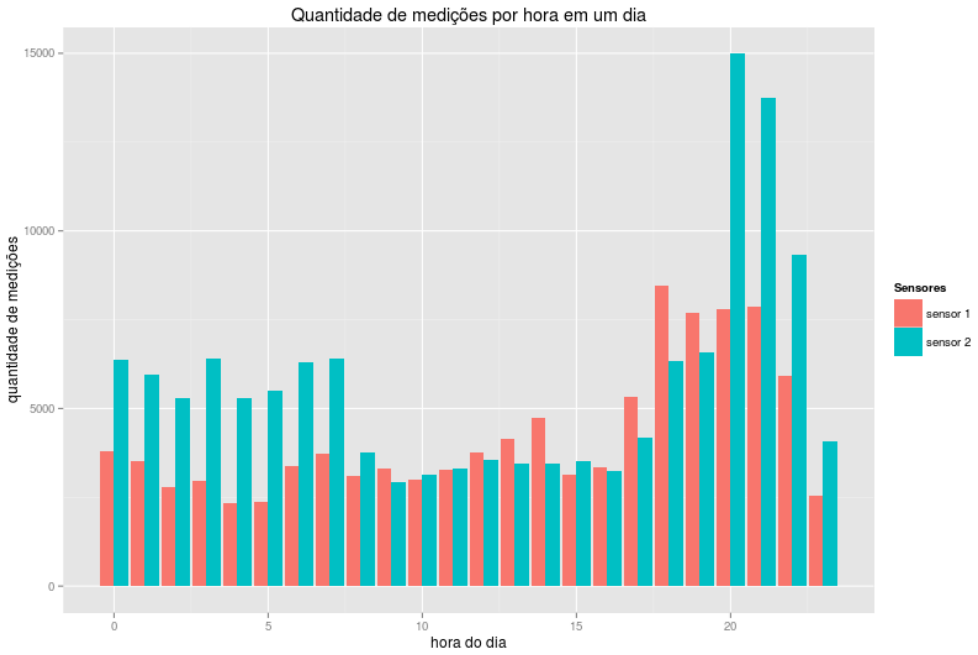

Figura 11 – Gráfico gerado a partir da saída de uma tarefa MapReduce que conta o número de medições por hora em um dia.

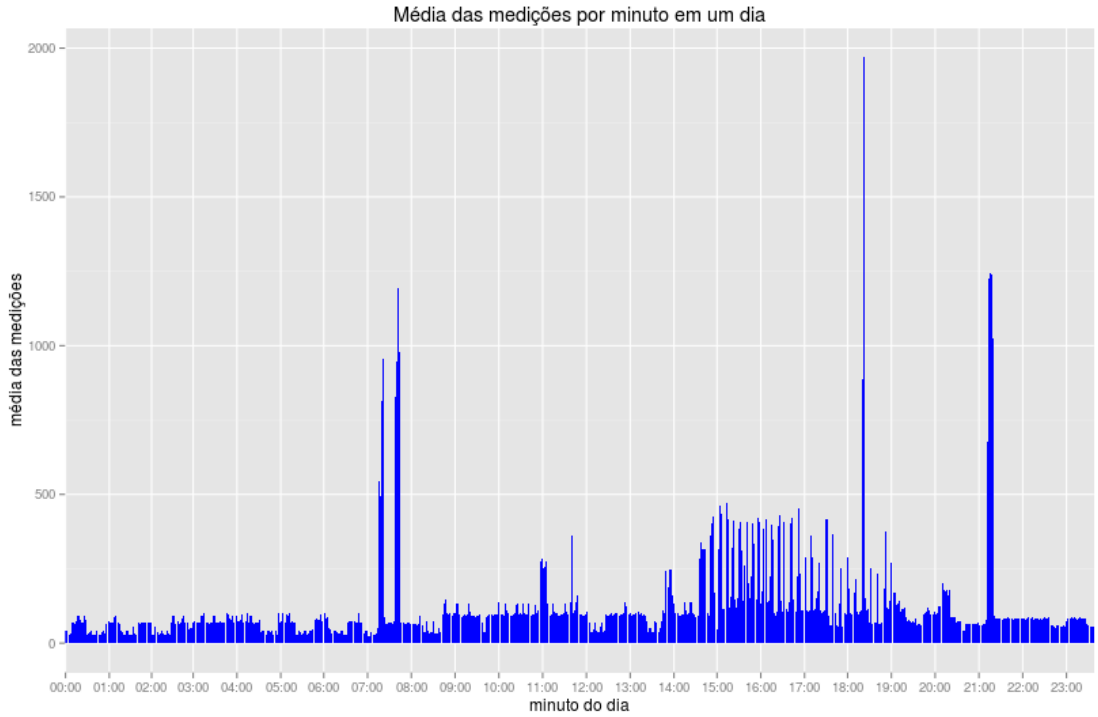

Figura 12 – Gráfico gerado a partir da saída de uma tarefa MapReduce que calcula a média das medições por minuto em um dia

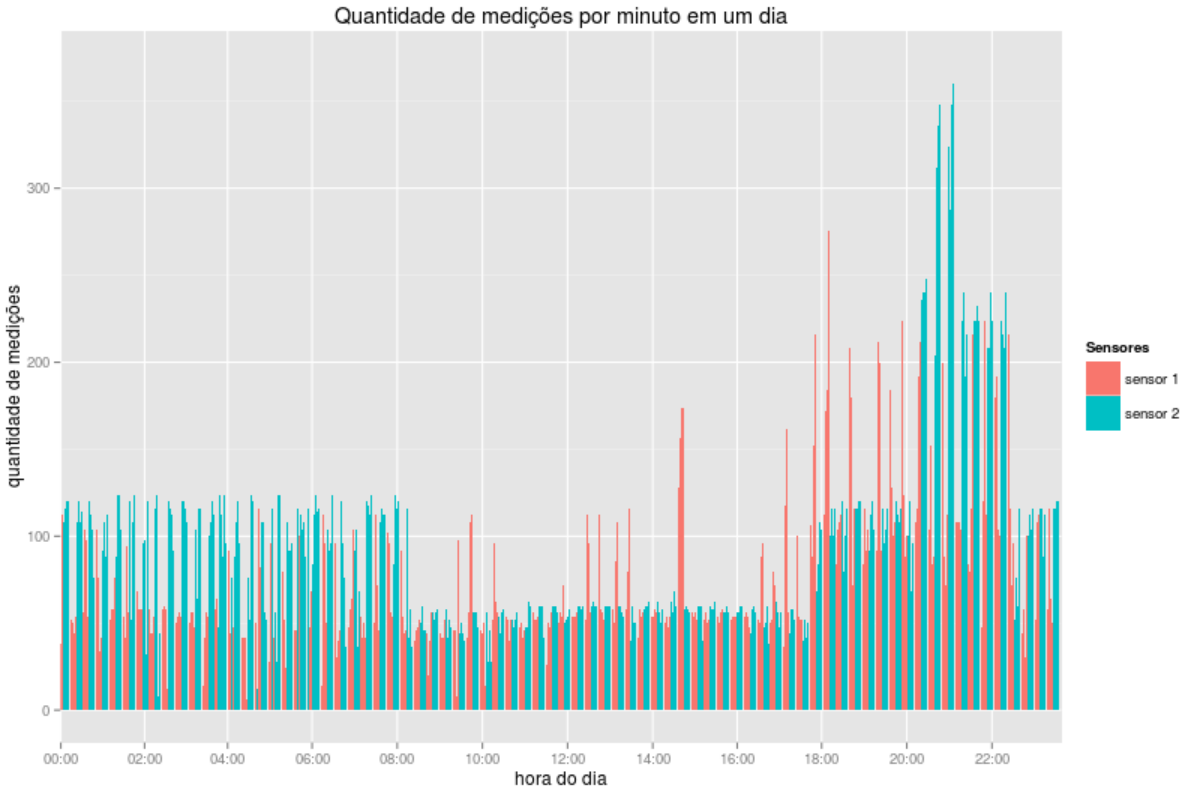

Figura 13 – Gráfico gerado a partir da saída de uma tarefa MapReduce que conta a quantidade de medições por minuto em um dia.

CONCLUSÃO

Conclui-se que tanto o Hadoop, quanto o OpenStack Sahara se mostram tecnologias em ascensão que valem a pena ser estudados, entendidos, e utilizados pelo fato de agregar alto valor a aplicação, devido ao baixo custo, alta velocidade de processamento e facilidade para a realização de tarefas. Os resultados alcançados poderão ajudar outras pessoas do Laboratório de Sistema Distribuídos, assim como parceiros de pesquisa, a implementar suas tarefas de big data com mais facilidade.

AGRADECIMENTOS

Ao CNPq pela manutenção das bolsas, PIBITI e produtividade em pesquisa; ao Laboratório de Sistemas Distribuídos, onde este estudo foi feito, e a todos que contribuíram para este estudo. O presente trabalho foi realizado com apoio do CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil.

REFERÊNCIAS BIBLIOGRÁFICAS

APACHE KAFKA. **Documentação do Kafka.** Disponível em: <http://kafka.apache.org/documentation.html>. Acesso em: 19 agosto 2015

APACHE SOFTWARE FOUNDATION. **Documentação do Apache Hadoop**. Disponível em: <http://Hadoop.apache.org/core/>. Acesso em: 19 de ago. 2015.

BORGES, H. P. et al. **Computação em Nuvem**, Brasil, 2011. Disponível em: <http://livroaberto.ibict.br/handle/1/861>. Acesso em: 19 agosto 2015.

CORDEIRO, D. **Apache Hadoop Conceitos teóricos e práticos, evolução e novas possibilidades**. Disponível em:<http://www.ime.usp.br/~danielc/papers/erad-Hadoop-DanielCordeiro.pdf>.Acesso em: 23 agosto 2015

DAI C.; STATEN J. The True State of Cloud Adoption**.Summit OpenStack**, Hong Kong, Novembro 2013.

DEAN, J; GHEMAWAT, S.; MapReduce: Simplified Data Processing on Large Clusters **OSDI'04: Sixth Symposium on Operating System Design and Implementation**, São Francisco, Dezembro 2004.

GE INTELLIGENT PLATFORMS. **The Rise of Industrial Big Data: Leveraging large time-series data sets to drive innovation, competitiveness and growth—capitalizing on the big data opportunity**. Disponível em: <http://leadwise.mediadroit.com/files/19174the_rise_of_industrial_big_data_wp_gft834.pdf>. Acesso em 19 agosto 2015.

HORTONWORKS. **O que é Apache Hadoop?** . Disponível em: <http://br.hortonworks.com/Hadoop/hdfs/>. Acesso em: 24 de ago. 2015.

MIRANTIS. **OpenStack Data Processing: Sahara**. Disponível em: <https://www.mirantis.com/products/data-processing-sahara/>. Acesso em: 21 agosto 2015.

OpenStack. **Documentação do OpenStack.** Disponível em: <http://www.OpenStack.org/>. Acesso em: 19 de ago. 2015.

SANCHEZ, O. P.; CAPPELLOZZA, A. Antecedentes da adoção da computação em nuvem:efeitos da infraestrutura, investimento e porte**.** ***Revista de Administração. Contemporânea***, Curitiba, v.16, n. 5, 2012. Disponível em: <http://dx.doi.org/10.1590/S1415-65552012000500002>. Acesso em: 19 agosto 2015.

SHAFER, F.; RIXNER, S.; COX, A. L; The Hadoop Distributed Filesystem: Balancing Portability and Performance. **IEEE International Symposium on Performance Analysis of Systems and Software**, White Plains, Março 2010.

TURNER, V. et al. **The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things**. Disponível em:<http://idcdocserv.com/1678>. Acesso em: 21 agosto 2015.

VEGA, J. J. C.; ORTEGA, J. F. C.; AGUILAR, L. J. Conociendo Big Data.**Facultad de Ingeniería**, Tunja, v. 24, n. 38, 2015. Disponível em: <http://revistas.uptc.edu.co/revistas/index.php/ingenieria/article/view/3159>. Acesso em: 19 agosto 2015.