UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUÍS FELIPE POLO

Aplicação Big Data para Predição de Tempo de Viagens de Táxi

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Cláudio Fernando Resin Geyer
Co-orientador: Profa. Dra. Renata de Matos Galante

Porto Alegre
2017

RESUMO

Em uma iniciativa para melhorar acessibilidade, transparência e prestação de contas da cidade, os legisladores de Nova York aprovaram em 2012 uma legislação sobre *Open Data*. O projeto *NYC Open Data* disponibiliza dados gerados por diversos órgãos públicos da cidade. Qualquer pessoa pode utilizar esses dados para realizar pesquisas e desenvolver aplicações. Como parte desse projeto, a *New York City Taxi Limousine Commission* disponibiliza dados de mais de um bilhão de viagens de táxi realizadas na cidade de Nova York desde 2009. Este trabalho propõe uma aplicação que cruza dados de corridas de táxi com dados climáticos para identificar a influência de diferentes variáveis climáticas no tempo de viagem. A aplicação faz uma predição da duração de uma corrida de táxi a ser realizada baseada no histórico de corridas similares, as variáveis que definem a similaridade são a data/hora inicial, data/hora de chegada, local de origem, local de destino e informações climáticas (chuva, neve e vento). Para fazer essa predição, foi utilizado um modelo que inicialmente agrupa corridas similares geograficamente e temporalmente, e depois utiliza regressão para definir a influência das variáveis climáticas e estimar o tempo. Devido ao grande volume dos conjuntos de dados utilizados, é necessária uma implementação paralela e execução da aplicação em um ambiente com maior capacidade de processamento para obter um bom desempenho. A implementação é feita utilizando o modelo de programação MapReduce. Os experimentos para avaliação dos resultados e do desempenho são feitos utilizando Hadoop em um cluster na plataforma Microsoft Azure.

Palavras-chave: Big Data. MapReduce. Open Data. Trânsito. Predição de Tempo.

Big Data Application For Taxi Trip Time Predicion

ABSTRACT

As an initiative to improve the accessibility, transparency, and accountability of City government, New York City lawmakers passed an Open Data legislation in 2012. The NYC Open Data project provides data generated by various New York City public agencies. Anyone can use this data to research and develop applications. As part of this project, the New York City Taxi Limousine Commission provides information covering over 1 bilion taxi trips in the city from January 2009 through December 2016. This work proposes an application that uses taxi trip data and weather data to identify the influence of weather variables on travel time. The application predicts the travel time of a future trip based on similar trips, the variables that define similarity are starting time, arrival time, origin, destination, and weather information (rain, snow and wind). First, similar trips are grouped based on geographical and temporal information, and then a regression model is used to define the influence of weather variables and estimate time. The application is implemented based on the MapReduce programming model. Experiments to evaluate results and performance are done using Hadoop in a cluster on the Microsoft Azure platform.

Keywords: Big Data, MapReduce, Open Data, Traffic, Trip Time Prediction.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Com a revolução digital das décadas recentes, houve uma explosão na capacidade de armazenar, transmitir e manipular grandes quantidades de dados. Até o ano 2000, apenas um quarto de toda a informação armazenada no mundo era digital, o resto estava em papel, filme fotográfico e outras mídias analógicas. Hoje, menos de 2% do total de informações armazenadas no mundo não é digital (CUKIER; SCHONBERGER, 2013).

A exploração dessa grande quantidade de dados permite descobrir padrões e tendências que até então não estavam ao nosso alcance, ligar e relacionar diferentes aspectos de sistemas para entender melhor seu comportamento, tomar decisões, e resolver problemas complexos (WU et al., 2013). São abertas novas perspectivas em diversas áreas de pesquisa: política, comunicação, medicina, ecologia, meteorologia, entre outras.

Tecnologias inteligentes e Big Data têm sido cada vez mais utilizadas para resolver problemas urbanos relacionados com poluição, trânsito e infraestrutura. Muitas cidades já começaram a desenvolver uma variedade de serviços e aplicações para melhorar a experiência urbana para moradores e turistas.

Governos de vários níveis têm criado projetos de *Open Data*, são projetos que disponibilizam publicamente dados de diversos departamentos de governos. A maioria das definições de *Open Data* incluem dois conceitos básicos: os dados devem estar publicamente disponíveis para qualquer pessoa usar e devem ser licenciados de uma maneira que permita reuso. Esses dados devem ser relativamente fáceis de usar e em geral devem ser gratuitos ou ao menos ter um custo muito baixo. O acesso a dados em um formato em que podem ser processados computacionalmente permite a criação de uma série de aplicações proporcionando um melhor entendimento dos serviços prestados pelas agências governamentais e melhorando a vida das pessoas nessas áreas urbanas (BROEK; HUIJBOOM, 2011).

Em uma iniciativa para melhorar acessibilidade, transparência e prestação de contas da cidade, os legisladores da cidade de Nova York aprovaram em 2012 uma legislação sobre *Open Data*. O projeto *NYC Open Data* disponibiliza dados gerados por diversos órgãos públicos da cidade. Qualquer pessoa pode utilizar esses dados para realizar pesquisas e desenvolver aplicações.

Como parte desse projeto, a *New York City Taxi and Limousine Comission* regularmente disponibiliza dados das corridas de táxi realizadas na cidade. Os registros contêm informações como local de origem e destino, hora inicial e final, distância, preço,

quantidade de passageiros, forma de pagamento, etc. Estão disponíveis cerca de 200 GB de dados com informações de viagens realizadas a partir de janeiro de 2009, totalizando pouco mais de 1 bilhão de registros.

Trabalhando com esses dados, Wang et al. (2013) propõem um modelo de predição de tempo para corridas de táxi com base no histórico de viagens em condições similares. Com base nas coordenadas geográficas - origem e destino - e na hora inicial da corrida, as condições de tráfego do momento da viagem são estimadas e é feita a predição de tempo. A aplicação recebe como entrada informações da corrida a ser realizada - a hora inicial, a origem e o destino - e gera como saída uma predição de tempo para o deslocamento. A aplicação não especifica a rota, apenas estima o tempo necessário.

Há vários cenários em que esse tipo de aplicação pode ser útil. Por exemplo, uma pessoa precisa pegar um táxi para pegar um voo que parte no dia seguinte às 17h. Como ela não está dirigindo, uma rota específica é menos importante, a única preocupação é saber quanto tempo vai demorar o deslocamento para decidir qual o melhor horário de partida. Outro exemplo, é relacionado a medidas de eficiência do tráfego da cidade. Uma métrica utilizada por pesquisadores da área é o tempo médio que as pessoas gastam todos dias para ir e voltar do trabalho, chamado de tempo de migração pendular (BUN; MCDONALD, 2003). Para estimar o tempo de migração pendular de uma cidade, a habilidade de prever o tempo de deslocamento entre quaisquer dois pontos da cidade é crucial.

Além de condições geográficas e temporais, outro fator que pode afetar significativamente o fluxo de trânsito é o clima (ZHAO; CHIEN, 2012). Em condições climáticas adversas, a velocidade diminui e surgem congestionamentos. Adicionando informações climáticas a um modelo de predição de tempo é possível melhorar a precisão da estimativa (NOOKALA, 2006).

Neste trabalho, inicialmente é feita uma implementação paralela da aplicação proposta por Wang et al. (2013). Depois, busca-se melhorar a precisão das estimativas adicionando informações climáticas ao modelo. Os registros das viagens de táxi são cruzados com informações históricas de clima. Desta forma, para cada registro de corrida de táxi são adicionadas informações climáticas do momento em que ela foi realizada. As variáveis adicionadas são chuva, neve e vento. Para estimar a influência dessas variáveis no tempo da corrida é utilizado um modelo de regressão polinomial, como proposto por Stern, Shah and Goodwin (2013). A aplicação é implementada de forma paralela e as execuções para avaliação de desempenho são feitas utilizando um cluster na plataforma

Microsoft Azure.

Os resultados dos experimentos confirmam que a estimativa de tempo se torna mais precisa quando dados climáticos são adicionados ao modelo. O erro médio diminui de 130 para 113 segundos, uma diminuição de 15,5%. Também foram obtidos ganhos de desempenho com a paralelização da aplicação. Por exemplo, a execução do treinamento da regressão com uma entrada de 32 GB dura cerca de 72 minutos quando feita em uma única máquina, mas quando executada em 8 máquinas leva cerca de 13 minutos.

1.1 Organização do trabalho

O restante do texto está estruturado da seguinte forma: o segundo capítulo apresenta os conceitos necessários para a compreensão do trabalho. O terceiro capítulo apresenta o modelo da solução proposta. O quarto capítulo descreve detalhes da implementação e as estratégias utilizadas para a paralelização da aplicação. No quinto capítulo, os resultados de todos os experimentos realizados são apresentados e analisados. Por fim, o último capítulo relata as conclusões e possibilidades de trabalhos futuros.

2 TECNOLOGIAS E CONCEITOS

Neste capítulo serão apresentados os conceitos e tecnologias que servem de base para a compreensão do trabalho. Os itens 2.1, 2.2 e 2.3 se referem a conceitos de Big Data, MapReduce e Hadoop. O item 2.4 descreve o conceito de *Open Data*. O item 2.5 explica o conceito de regressão, que será usado no modelo apresentado no capítulo 3. E por fim, 2.6 trata sobre trânsito, clima e modelos para predição de tempo de deslocamento.

2.1 Big Data

Um dos principais desafios computacionais da atualidade é armazenar, gerenciar, analisar e extrair valor da grande quantidade de dados existente. Segundo a IBM, 90% de toda informação global foi criada nos últimos dois anos. Atualmente, cerca de 2,5 quintilhões de bytes são produzidos por dia. Com dispositivos móveis, redes sociais, sensores e tecnologias emergentes, a tendência é que o crescimento se acelere ainda mais (IBM, 2017).

O Oxford English Dictionary define Big Data como: "Dados de um tamanho muito grande, cuja manipulação e gerenciamento trazem significativos desafios."

O McKinsey Global Institute usa a seguinte definição para Big Data:

> Big Data se refere a conjuntos de dados cujo tamanho vai além da capacidade que ferramentas de software convencionais têm para analisar, armazenar e gerenciar esses dados. Essa definição é intencionalmente subjetiva e incorpora uma definição do quão grande um conjunto de dados precisa ser para ser considerado Big Data. Não se define Big Data em termos do conjunto de dados ser maior que um determinado número de gigabytes ou terabytes. Assume-se que, conforme a tecnologia avança ao longo do tempo, o tamanho dos conjuntos de dados considerados Big Data também aumente (MCKINSEY, 2011, p. 1).

Em resumo, Big Data é um conjunto de dados tão grande que é difícil processar usando estruturas de armazenamento e técnicas de software convencionais.

2.1.1 Características de Big Data

São associados 5Vs ao termo Big Data: volume, variedade, velocidade, veracidade e valor. Apesar de não usar o termo Big Data, Laney (2001) foi o primeiro a associar

volume, variedade e velocidade a grandes volumes de dados. Volume se refere à quantidade de dados gerados e armazenados. Variedade se refere ao tipo e a natureza dos dados. Velocidade se refere à velocidade com que dados precisam ser gerados e/ou processados para atender a demanda.

Hoje em dia, veracidade e valor também são associados a Big Data. Veracidade se refere à precisão e confiabilidade dos dados. Valor se refere ao potencial dos dados em contribuir para a resolução de problemas (DEMCHENKO; NGO; MEMBREY, 2013).

2.1.2 Processamento Batch vs. Real Time

O tipo de processamento pode ser online ou offline. No processamento offline, chamado de processamento batch, os dados são primeiramente armazenados e posteriormente processados em lotes. No processamento online, ou real time, os dados vão sendo processados conforme vão chegando e os resultados são produzidos em um intervalo de tempo não muito grande.

2.1.3 Frameworks

Devidos às restrições físicas das arquiteturas convencionais, a computação distribuída e paralela é impulsionada como alternativa para aplicações Big Data. O processamento é normalmente realizado em clusters e grids computacionais, que utilizando computadores comuns conseguem grande poder de processamento a um custo relativamente baixo.

Porém, como dividir uma tarefa em subtarefas e então executá-las em paralelo não é trivial, os frameworks Big Data buscam fazer várias abstrações para ajudar o programador a armazenar, analisar e processar os dados. Os mais usados atualmente são Hadoop, Spark, Flink e Storm (INOUBLI et al., 2017). Hadoop, utilizado neste trabalho, será apresentado na seção 2.3.

2.2 MapReduce

MapReduce é um modelo de programação paralela criado pela Google para o processamento distribuído de grandes volumes de dados. Em um programa no modelo Ma-

pReduce, o programador deve apenas especificar as funções *map* e reduce. A função *map* transforma a entrada de dados em um par chave-valor e gera um conjunto de pares chave-valor intermediários. A função *reduce* processa os valores intermediários associados à mesma chave e emite um novo par chave-valor (DEAN; GHEMAWAT, 2008).

Esse modelo foi inspirado pelas primitivas *map* e *reduce* presentes em Lisp e outras linguagens funcionais; programas escritos dessa forma podem ser facilmente paralelizados e executados em clusters. O ecosistema do MapReduce é responsável por todas as tarefas relacionadas à paralelização da computação: particionamento dos dados, escalonamento de tarefas, comunicação entre nós, tolerância a falhas, etc. (DEAN; GHEMAWAT, 2008).

2.2.1 Modelo de Programação

A computação recebe um conjunto de entrada com pares chave-valor e produz um conjunto de saída com pares chave-valor. O usuário do framework MapReduce expressa a computação como duas funções: *map* e reduce.

A função *map* recebe como entrada um par chave-valor e produz um conjunto de pares também no formato chave-valor. Uma função de Hash agrupa todos os valores associados à mesma chave e os repassa para a função reduce.

A função *reduce* recebe como entrada uma chave e conjunto de valores relacionadas a essa chave. Esses dados são processados conforme a função definida pelo usuário, em geral produzindo um conjunto menor de valores (DEAN; GHEMAWAT, 2008).

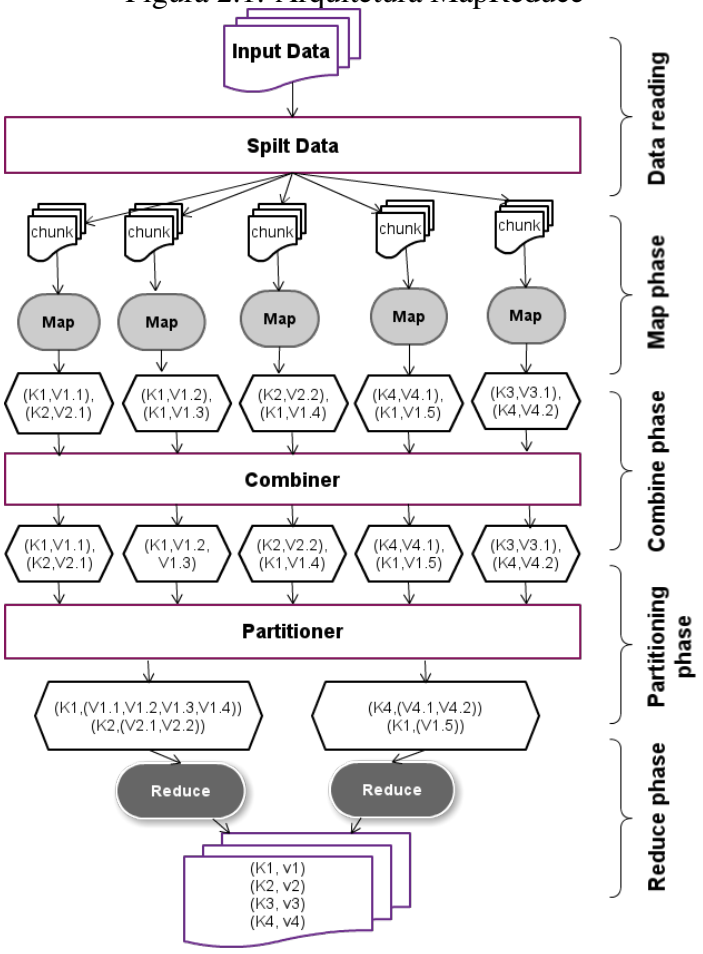
2.2.2 Fluxo de Execução

Os passos da execução de um programa MapReduce, como apresentados na Figura 2.1, são os seguintes (INOUBLI et al., 2017):

- Leitura de dados: A entrada é dividida em vários blocos, chamados chunks. Cada chunk será processado por uma instância da função map;
- *Map*: para cada entrada, a função é executada e gera um conjunto intermediário de zero ou mais pares chave-valor;
- *Combine*: essa etapa agrupa todos os pares chave-valor intermediários associados a uma mesma chave;

- Partição: os resultados são distribuídos para as funções de reduce; todos os pares com mesma chave são enviados para uma mesma instância de reduce;
- *Reduce*: com os pares chave-valor agrupados por chaves, a função *reduce* utiliza os valores para computar o resultado final.

Figura 2.1: Arquitetura MapReduce

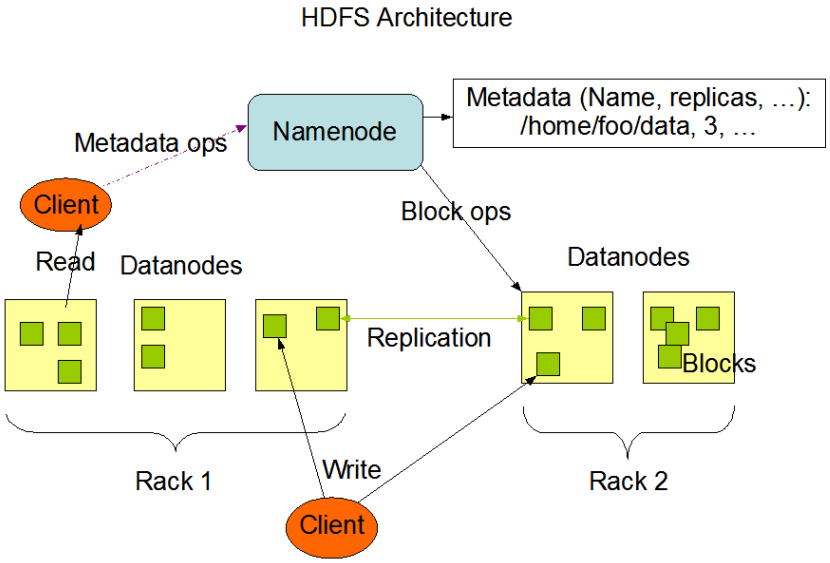


Fonte: Inoubli et al., (2017, p. 4)

2.3 Hadoop

O Hadoop é um framework para armazenamento distribuído e processamento de Big Data que utiliza o modelo de programação MapReduce. Os principais componentes do Hadoop são o *Hadoop Distributed File System (HDFS)* para armazenamento de dados e o Hadoop MapReduce, uma implementação do modelo MapReduce proposto pela Google.

Figura 2.2: Arquitetura do HDFS



Fonte: HADOOP (2013)

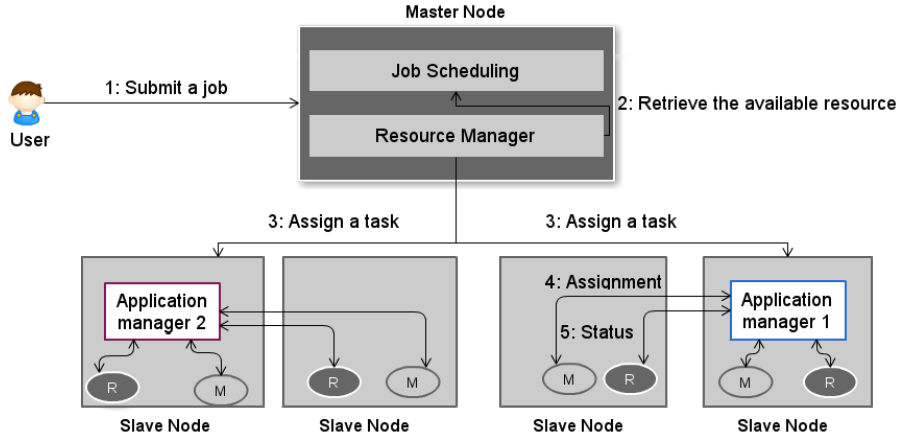
2.3.1 HDFS

O *Hadoop Distributed File System* (HDFS) é uma implementação de código aberto do Google File System - sistema de arquivos distribuído do MapReduce. O HDFS provê um sistema de arquivos distribuído para armazenamento de grandes arquivos de uma maneira eficiente e confiável (TOSHNIVAL, 2014).

A arquitetura do HDFS, representada pela Figura 2.2, consiste em uma arquitetura mestre-escravo com o Namenode como mestre e diversos Datanodes como escravos. O Namenode é responsável por alocar espaço físico para armazenar grandes arquivos enviados pelo cliente HDFS. Se o cliente quer acessar dados do HDFS, ele manda uma requisição para o Namenode. O Namenode procura a localização no seu sistema de index e envia o endereço para o cliente. O Namenode retorna para o cliente HDFS os metadados (nome do arquivo, localização, etc.) relacionados aos arquivos armazenados. Um Namenode secundário salva periodicamente o estado do Namenode. Se o Namenode falha, o Namenode secundário assume automaticamente (INOUBLI et al., 2017).

Outras características importantes para o bom desempenho do HDFS são o balanceamento e o Rack Awareness (WHITE, 2010). O sistema de arquivos busca sempre manter os dados distribuídos de forma balanceada entre os Datanodes, redistribuindo os dados se necessário. O Rack Awareness é um recurso utilizado pelo HDFS para identificar os Datanodes pertencentes ao mesmo rack e assim distribuir as réplicas de forma mais inteligente.

Figura 2.3: Arquitetura Yarn



Fonte: Inoubli et al., (2017, p. 8)

2.3.2 Hadoop MapReduce

O Hadoop MapReduce é uma implementação em Java do modelo MapReduce criado pela Google. Existem duas versões principais do Hadoop MapReduce (INOUBLI et al., 2017). Na primeira versão, chamada de MRv1, o Hadoop era baseado em dois componentes. O primeiro é o Task Tracker, que supervisiona a execução das funções *map* e *reduce*. O outro é o Job Tracker, que representa a parte mestre da arquitetura e permite gerenciamento de recursos, escalonamento e monitoramento de jobs. O Job Tracker supervisiona e gerencia os Task Trackers. Na segunda versão, chamada de Yarn, demonstrada na Figura 2.3, as duas principais funcionalidades do Job Tracker foram divididas em dois daemons: um Resource Manager e um Application Master por aplicação. O Resource Manager recebe e executa jobs MapReduce. O Application Master obtém os recursos do Resource Manager e trabalha com os Node Managers para rodar e monitorar as tarefas. O Node Manager é responsável por disparar e gerenciar containers em um nodo. Containers executam as tarefas especificadas pelo Application Master. No Yarn, o gerenciador de recursos substitui o Job Tracker e o Node Manager substitui o Task Tracker (LI et al., 2015).

2.4 Open Data

Segundo a *Open Knowledge Foundation*, o termo *Open Data* se relaciona aos seguintes conceitos (MOLLOY, 2011):

- Disponibilidade e acesso: os dados devem estar disponíveis como um todo e a um custo razoável para acesso, preferencialmente através de download pela internet. Os dados também devem estar disponíveis em formato conveniente e modificável;
- Reuso e redistribuição: os dados devem ser disponibilizados em termos que permitem reuso e redistribuição incluindo cruzamento com outros datasets;
- Participação universal: todas as pessoas podem usar, reusar e redistribuir. Não deve haver restrições por áreas de atuação nem contra grupos ou pessoas. Por exemplo, não são permitidas restrições contra uso comercial ou para propósito específicos, como licenças somente para fins educativos.

Instituições públicas ao redor do mundo têm cada vez mais adotado políticas relacionadas à abertura de dados para promover mais transparência e maior prestação de contas aos cidadãos. Órgãos públicos produzem grandes quantidades de dados, ao incentivar o uso, reuso e livre distribuição desses dados, governos promovem inovação e criação de negócios, e serviços voltados aos cidadãos (BROEK; HUIJBOOM, 2011).

2.5 Regressão Linear

Análise de regressão é uma técnica estatística para estimar a relação entre duas ou mais variáveis. O objetivo é analisar a relação entre uma variável dependente e uma ou mais variáveis independentes e formular a relação funcional entre essas variáveis (UYANIK; GULER, 2013).

A regressão é chamada de linear quando as relações entre as variáveis são modeladas através de combinações lineares das variáveis independentes. A regressão linear com uma variável independente é chamada de regressão linear simples; com mais de uma variável independente é chamada de regressão multilinear (UYANIK; GULER, 2013).

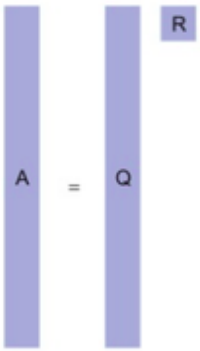
A maneira mais comum para estimar os coeficientes de uma regressão linear é usando o método dos mínimos quadrados. O método visa minimizar a soma dos quadrados das distâncias entre os dados estimados e os dados observados, maximizando o ajuste do modelo aos ajustes observados (ADJOUT; BOUFARES, 2015).

2.5.1 Mínimos Quadrados e Decomposição QR

Será utilizada a decomposição QR para a resolução do problema dos mínimos quadrados de um sistema Ax = b, onde b representa os valores da variável dependente, A representa os valores das variáveis independentes e x representa os coeficientes a serem estimados.

A decomposição QR de uma matriz A de tamanho m x n é: A = QR onde Q é uma matriz m x n ortogonal e R é uma matriz n x n superior triangular.

Figura 2.4: Decomposição QR



Uma das formas de se obter as matrizes Q e R é utilizando a transformação de Householder. Na transformação de Householder aplica-se Q como uma matriz de reflexão sobre algum vetor, de forma que todos os seus elementos, com exceção do primeiro, se anulem após a aplicação de Q. Depois de computadas as matrizes Q e R, os valores de x são calculados resolvendo o sistema $Rx = Q^T b$. Esse sistema pode ser resolvido facilmente, visto que R é triangular superior.

2.5.2 Regressão polinomial

Regressão polinomial é uma forma de regressão em que a relação entre uma variável independente x e uma variável dependente y é modelada como um polinômio de grau n em x. Portanto, a regressão polinomial modela uma relação não linear entre a variável dependente e as variáveis independentes (FOX, 2008).

Apesar de modelar uma relação não-linear, a regressão polinomial é considerada um caso especial de regressão multilinear. Na regressão linear, as variáveis independentes são tratadas como valores fixos e a linearidade é uma restrição apenas nos coeficientes da regressão. Portanto, do ponto de vista estatístico o problema é linear, no sentido em que

os parâmetros a serem estimados são lineares (FOX, 2008).

2.6 Estimativa de tempo de deslocamento baseada em dados históricos

Grande parte da literatura da área é voltada para a estimativa de tempo entre uma origem e um destino baseada em uma rota, isto é, os dados possuem vários pontos intermediários, permitindo dividir o trajeto em vários segmentos. A abordagem para predição de tempo de um trajeto baseada em uma rota busca estimar o tempo de deslocamento em cada segmento do caminho e depois soma os tempos de todos os segmentos para estimar o tempo total da rota (WANG et al., 2013).

Os dados usados para estimar o tempo em cada segmento são coletados por sensores em rodovias ou aparelhos de GPS instalados nos carros (TREIBER; KESTING, 2013). *Loop traffic detectors* são sensores instalados em rodovias capazes de detectar quando um veículo está passando sobre ele. (PETTY et al., 1998), (ZIA et al., 2001), (RICE; ZWET, 2004), (WANG; PAPAGERGIOU, 2005) propõem métodos que utilizam dados desses sensores para inferir a velocidade em cada segmento e assim estimar o tempo de cada segmento da rota. Dados de *floating car* são coordenadas com um timestamp coletadas a partir de aparelhos GPS instalados nos carros. (TOSSAVAINEM et al., 2008) e (CINTIA et al., 2013) propõem modelos que utilizam esses dados para estimar a velocidade em cada segmento.

Para os casos em que não há informações sobre a trajetória, (DEAN, 1999), (KANOULAS et al., 2006), (GONZALES et al., 2007), (YUAN et al., 2010) propõem inferir uma rota de acordo com algum critério, como caminho mais curto ou rota mais rápida, e depois estimar os tempos de cada segmento da mesma forma que os modelos que utilizam dados com informações da trajetória.

Wang et al. (2013) mostram que, quando se trabalha com uma quantidade muito grande de dados históricos, é possível chegar a resultados bastante precisos sem precisar inferir uma rota. O modelo proposto por Wang et al. (2013) estima o tempo entre origem e destino trabalhando apenas com os dados de viagens realizadas em condições similares. A abordagem é baseada na seguinte ideia: suponha que se deseja estimar quanto tempo leva para ir do ponto A até o ponto Z às 23:43 em um sábado. Se existirem milhares de viagens realizadas exatamente nessas condições, então uma média do tempo dessas corridas pode ser usada como estimativa. Mas mesmo trabalhando com uma grande quantidade de dados, o número de viagens com mesma origem, destino, hora e dia da semana pos-

sivelmente não será grande o suficiente para uma estimativa precisa. Para resolver esse problema, é proposto modelar a dinâmica de trânsito calculando a velocidade temporal referencial, que permite levar em consideração todas as viagens do ponto A até o ponto Z independentemente do horário em elas foram realizadas. A velocidade referencial define quantas vezes mais rápida ou mais lenta seria a viagem se ela fosse realizada em outro horário. Mais especificamente, para duas viagens P e Q realizadas em horários diferentes, mas com origem e destino similares, é possível estimar o tempo de P ajustando o tempo da viagem Q com a velocidade temporal referencial. Esse método será explicado em detalhes no capítulo 3.

2.6.1 Influência do clima nas condições de tráfego

Condições climáticas adversas podem ter um significativo impacto no trânsito, diminuindo a velocidade média dos veículos e gerando congestionamentos. Em casos de chuva, neve ou vento, o fluxo de trânsito piora e o tempo de viagem aumenta.

Tsapakis, Cheng and Bolbol (2012) analisaram os impactos de diferentes níveis de chuva, neve e temperatura no trânsito da cidade de Londres. O estudo mostra que o impacto de chuva e neve varia em função de sua intensidade. O tempo de deslocamento médio de um ponto a outro na cidade aumenta entre 0,1 e 2,1% em caso de chuva fraca; 1,5% e 3,8% para chuva média; 4% e 6% em caso de chuva forte. Neve fraca aumenta o tempo de deslocamento médio entre 5,5% e 7,6%, enquanto em uma forte nevasca o tempo varia de 7,4% a 11,4%. Variações de temperatura têm efeito desprezível no tempo médio de deslocamento.

Usando dados de GPS coletados na Dinamarca, Andersen and Torp (2016) mostram que a neve pode aumentar em até 27% o tempo de viagem e que vento muito forte pode aumentar em até 19% o tempo de viagem.

Os dados desses estudos mostram que uma estimativa de tempo para um deslocamento em uma cidade pode ganhar mais precisão levando em consideração variáveis climáticas.

Nookala (2006) propõe adicionar informações do clima a um modelo de predição de tempo. O trabalho mede a correlação da velocidade do vento, umidade do ar, temperatura do ar e temperatura da superfície com o tempo do trajeto.

Stern, Shah and Goodwin (2013) chegam a melhores resultados utilizando modelos de regressão para estimar a relação entre nível de precipitação, vento e o tempo de

deslocamento em Washington DC. O estudo mostra que equações de regressão em sua forma quadrática - regressão polinomial, portanto - obtém melhores resultados e são a melhor opção para relacionar o tempo da corrida às variáveis utilizadas nesse modelo.

2.7 Considerações Finais

Neste capítulo foram apresentados os principais conceitos relacionados ao trabalho. As seções sobre Big Data, MapReduce e Hadoop tratam sobre conceitos que são importantes devido ao grande volume dos datasets utilizados e a necessidade de paralelizar a computação. Depois, são apresentados conceitos de regressão e de predição de tempo que serão utilizados no próximo capítulo, que define o problema e descreve o modelo proposto para resolução.

3 MODELO

Este capítulo descreve o modelo utilizado para a predição de tempo de uma corrida de táxi baseada em dados históricos.O problema é definido na seção 3.1. Em 3.2, é apresentado o conceito de similaridade geográfica. A dinâmica das condições de tráfego é modelada na seção 3.3. A seção 3.4 descreve a utilização de um modelo de regressão para definir o impacto das condições climáticas no tempo de viagem.

3.1 Definição do problema

Uma viagem Pi é definida como uma 8-upla (Oi, Di, Si, Li, Ti, Ri, Wi, Ni) que consiste em uma coordenada de origem Oi, uma coordenada de destino Di, uma data e hora inicial Si, uma distância Li, um tempo Ti, um nível de chuva Ri, uma velocidade de vento Wi e uma quantidade de neve Ni.

A partir de uma base de dados de viagens, dada uma viagem Q = (Oq, Dq, Sq, Ri, Wi, Ni), o objetivo é estimar o tempo de duração Tq de uma viagem com origem em Oq, destino em Dq, início na data/hora Sq, nível de chuva Ri (em cm), velocidade do vento Wi (em m/s) e quantidade de neve Ni (em cm) usando as informações da base de viagens com informações do clima.

É feita uma predição de tempo inicial usando o modelo de Wang et al. (2013). Esse modelo leva em consideração dados históricos relativos à origem, ao destino e ao momento que a corrida foi realizada para estimar o tempo de duração.

Para estimar a influência das variáveis climáticas na velocidade, é utilizado um modelo de regressão polinomial. As variáveis climáticas usadas são vento, chuva e neve. O resultado dessa etapa é usado para melhorar a estimativa inicial e gerar o resultado final.

3.2 Similaridade geográfica

Uma viagem Pi é considerada vizinha de q se origem e destino de Pi são geograficamente próximos da origem e destino de q. Então, o conjunto de vizinhos de uma viagem q é definido da seguinte forma: $\eta(q) =$ distância$(Oi, Oq) <= \tau$ e distância$(Di, Dq) <= \tau$.

A distância(x, y) é a distância euclidiana entre as coordenadas x e y. Um τ maior

gera um conjunto maior de vizinhos e então com mais dados a estimativa poderia ser mais precisa. Porém, um τ maior implica em menor similaridade entre as viagens, o que poderia introduzir erros na predição. Segundo os experimentos de Wang et al. (2013), o valor de τ que apresentou melhores resultados foi de 3 metros.

Com essa definição de vizinhos, uma abordagem inicial seria considerar a média de todas as viagens vizinhas como uma estimativa, mas as condições de tráfego variam muito em diferentes momentos do dia. Então é necessário modelar a dinâmica das condições de tráfego.

3.3 Dinâmica das condições de tráfego

A velocidade média de deslocamento varia muito conforme o momento em que a viagem é feita. Uma viagem na madrugada, por exemplo, é muito mais rápida em comparação com as realizadas nos horários de pico. Portanto, para estimar o tempo de deslocamento de uma corrida realizadas às 2h usando uma viagem vizinha p realizada às 17h, a ideia é diminuir proporcionalmente o tempo de p, já que uma corrida às 2h normalmente é muito mais rápida que uma corrida às 17h. Para ajustar a velocidade de todas as viagens vizinhas é definido um fator de escala. Com $V(s)$ indicando a velocidade média em um determinado intervalo de tempo, o fator de escala é definido como:

$$Si \approx \frac{Vi}{Vq} \approx \frac{V(Si)}{V(Sq)}$$

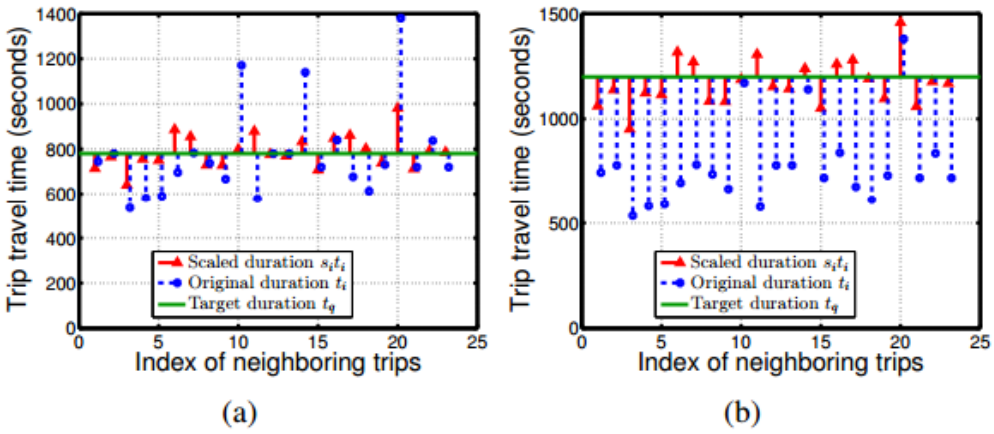
A equação que define o fator de escala é uma aproximação, pois o modelo utiliza uma pequena distância τ para extrair as viagens vizinhas.

A Figura 3.1 demonstra a efetividade do fator de escala. O objetivo é estimar o tempo de uma viagem q. O tempo de viagem Ti das viagens vizinhas é representado como um círculo azul, enquanto o tempo corrigido pelo fator de escala é representado pelo triângulo vermelho. A linha verde representa o tempo real de q. Em (a), o tempo das viagens vizinhas tem grande variação, pois elas são de intervalos de tempo diferentes com condições de tráfego diferentes. O fator de escala reduz com sucesso a variação em relação ao tempo da viagem q. Em (b), a viagem q ocorreu em horário de pico, então seu tempo difere bastante de grande parte das suas viagens vizinhas. O fator de escala reduz com sucesso a diferença de tempo entre as viagens históricas e a viagem q. Considerando

esse fator de escala, o tempo de viagem será estimado da seguinte forma:

$$Tq = \frac{1}{|\eta(q)|} \sum_{Pi \epsilon \eta(q)} (Ti . \frac{V(Si)}{V(Sq)})$$

Figura 3.1: Fator de Escala da Velocidade

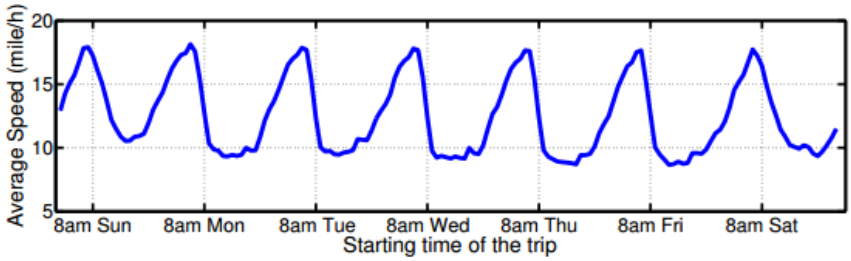


Fonte: WANG et al. (2013, p. 5)

3.3.1 Velocidade referencial relativa

Conforme é possível observar na Figura 3.2, a velocidade de uma corrida segue um forte padrão conforme a hora e dia da semana que ela é realizada. Todos os dias de semana têm padrões de tráfego semelhante às 8h da manhã, mas nos finais de semana o trânsito não é tão grande como o usual.

Figura 3.2: Velocidade média conforme horário e dia da semana



Fonte: WANG et al. (2013, p. 4)

Para calcular a velocidade referencial relativa, inicialmente a semana é dividida em T intervalos, onde T depende da periodicidade assumida. Por exemplo, se o intervalo

for de 1 hora, T = 24 * 7 = 168. A velocidade média do intervalo n é a média das velocidades de todas as corridas realizadas nesse intervalo. Para um intervalo i, um conjunto de viagens desse intervalo *S(Pi)*, a velocidade referencial é calculada como segue:

$$Vi = \frac{1}{|S(Pi)|} \sum_{Pj \epsilon S(Pi)} (\frac{Lj}{Tj})$$

Um problema desse modelo é que ele não é capaz de estimar com precisão a velocidade relativa de dias com condições de tráfego que não seguem o padrão, como feriados. Para resolver esse problema, Wang et al. (2013) utilizam o modelo ARIMA (*Autoregressive integrated moving average*) para capturar as condições de tráfego em diferentes intervalos de tempo. O modelo não será usado neste trabalho devido ao fato de que nesse modelo um cálculo para um termo t depende do resultado termo $t-1$ e portanto não é muito paralelizável.

Uma maneira que esse problema poderia ser solucionado seria criando um parâmetro de entrada informando que o dia não segue o padrão semanal e horário das condições de tráfego. Por exemplo, para o caso de 25 de dezembro, o modelo levaria em consideração apenas dados de corridas realizadas em 25 dezembro para calcular a velocidade referencial. Para o dia do trabalho, que nos Estados Unidos é comemorado na primeira segunda-feira de setembro, apenas dados de corridas realizadas na primeira segunda-feira de setembro seriam levados em consideração.

3.4 Clima

Tu, Lint and Zuylen (2007) mostram que em períodos de maior tráfego, como o início da manhã e o final da tarde, o impacto de condições adversas de clima é menos significativo que durante a noite ou fora dos horários de pico. A piora das condições do clima afeta menos o trânsito já congestionado, enquanto fora dos horários de pico, o trânsito não está congestionado e, portanto, mais sensível a efeitos externos.

Baseado nessas informações, este trabalho utiliza um modelo de regressão polinomial para estimar o quanto a velocidade varia em relação à velocidade média do horário calculada na seção 3.4. As variáveis independentes da regressão são a velocidade da corrida, a velocidade média do horário, nível de chuva, velocidade do vento e nível de neve. A variável dependente é o fator de escala para a velocidade. Esse fator de escala indica

quantas vezes mais lenta ou mais rápida é a velocidade dadas determinadas condições climáticas em comparação com a velocidade média do horário.

Para o conjunto de treinamento, o fator de escala é definido como a velocidade da corrida dividida pela velocidade média do horário. A partir deste conjunto de treinamento os coeficientes da regressão são estimados usando a decomposição QR.

Na execução da aplicação, o fator de escala para a velocidade é calculado em função dos dados de entrada de clima, a velocidade da corrida de acordo com a estimativa inicial de tempo e a velocidade média do horário. A partir da velocidade corrigida por esse fator de escala é gerada a estimativa de tempo final.

Foram testadas diversas equações de regressão a fim de definir a que mais acrescenta precisão ao modelo, os resultados serão apresentados no capítulo 5.

3.5 Considerações Finais

Neste capítulo foi apresentado um modelo para uma aplicação que estima o tempo de uma viagem de táxi com base em dados históricos. No próximo capítulo serão descritos detalhes de uma implementação paralela deste modelo.

4 PROTÓTIPO

Este capítulo descreve a implementação da aplicação conforme modelo apresentado no capítulo 3. A seção 4.1 descreve os dados utilizados e o cruzamento entre os dados de históricos de clima e táxi. Em 4.2 é apresentada a implementação do modelo de estimativa de tempo baseada no histórico de viagens similares. O item 4.3 descreve a implementação do modelo de regressão para estimar a influência do clima.

A aplicação foi escrita em Python utilizando o modelo de programação MapReduce e a API Hadoop Streaming. Hadoop streaming vem com a distribuição padrão do Hadoop e permite o desenvolvimento de *mappers* e *reducers* em qualquer linguagem de programação. *Mappers* e *reducers* leem a entrada de stdin e escrevem a saída em stdout. Para as operações de álgebra linear foi utilizada a biblioteca NumPy, que oferece diversas funções matemáticas para operações de álgebra linear com matrizes.

4.1 Dados

A implementação foi feita baseada em dados de corridas de táxi e de clima da cidade de Nova York no período de 2009 a 2016.

4.1.1 Dados das viagens de táxi

Os dados das corridas de táxi utilizados são os disponibilizados pela *New York City Taxi Limousine Commission (NYC TLC)*. São cerca de 230 GB de dados com informações de corridas de táxi realizadas entre janeiro de 2009 e dezembro de 2016, totalizando aproximadamente 1,3 bilhão de viagens.

Todas as corridas realizadas em um mês são disponibilizadas em um arquivo csv. Cada linha desse arquivo é um registro correspondente a uma corrida de táxi. Cada registro tem informações como data/hora do início e do final da corrida, coordenada inicial e final, distância, número de passageiros, preço, tempo da corrida, etc. Devido a questões de privacidade dos passageiros e dos motoristas, informações de GPS com pontos intermediários não são divulgadas.

Recentemente a *NYC TLC* passou a também divulgar dados de *For Hire Vehicules* (dados de aplicativos de transporte individual como Uber, Lyft, etc.), porém esses regis-

tros não têm todas informações que seriam necessárias para integrá-los ao trabalho, então não foram utilizados.

4.1.2 Dados climáticos

Os dados climáticos são disponibilizados pelo Mesowest, projeto da Universidade de Utah que provê acesso a dados climáticos históricos ou momentâneos. Na cidade de Nova York, há dados de três estações que monitoram o clima:

- *KNYC (New York City, Center)*, localizada nas coordenadas geográficas 40.78333N -73.96667W (Central Park);
- *KLGA (New York, La Guardia)*, localizada nas coordenadas geográficas 40.77917N -73.88000W (Aeroporto La Guardia);
- *KJFK (New York, Kennedy)*, localizada nas coordenadas geográficas 40.63915N -73.76393W (Aeroporto John F. Kennedy).

Para o período entre janeiro de 2009 e dezembro de 2016, existem cerca de 70.000 registros por estação, totalizando cerca de 240.000 registros. Cada registro de dados referentes ao clima tem informações como temperatura, chuva, vento, neve em um determinado horário.

4.1.3 Cruzamento dos dados

Foi feito um cruzamento dos dados para inserir informações nos registros das corridas de táxi sobre o clima do momento que elas foram realizadas. Foram considerados os dados da estação mais próxima do ponto médio entre origem e destino no horário médio entre início e fim da corrida.

Para isso, inicialmente a distância entre o ponto médio da corrida e cada estação é calculada. Depois são extraídos valores de data/hora inicial e data/hora final, e o horário médio da corrida é calculado. A partir desse horário médio é buscado o registro climático com horário mais próximo no arquivo de dados climáticos da estação selecionada. Como as informações de clima estão ordenadas cronologicamente, a busca é feita com um algoritmo similar a uma busca binária - a diferença aqui em relação à busca binária é que se busca o valor mais próximo e não uma chave específica - e a saída gerada é um registro

de corrida de táxi com todas as informações que ele já possuía e mais as informações do registro de clima encontrado adicionadas.

4.2 Estimativa com base no histórico de viagens

É feito um pré-processamento para calcular a velocidade média de cada um dos T intervalos de tempo de uma semana. O *mapper* recebe um registro de uma viagem de táxi e gera como saída um par chave-valor com a chave representando um intervalo de tempo e o valor representando a velocidade da corrida. O *reducer* calcula a média para cada um dos intervalos. O valor de T utilizado foi de 672, portanto são calculadas as velocidades médias de intervalos de 15 minutos (uma semana = 10.080 minutos = 672 * 15).

A estimativa de tempo da corrida baseada em dados históricos também é implementada utilizando MapReduce. Dada uma viagem de entrada q, para cada corrida do conjunto de dados, a origem é comparada com a origem de q e o destino é comparado com o destino de q. Se ambas as distâncias forem menor do que o parâmetro de entrada τ, essa viagem tem seu tempo relativo calculado utilizando o fator de escala definido conforme as velocidades médias dos horários de ambas as corridas. O *mapper* gera como saída um valor fixo na chave (todas as saídas devem ir para o mesmo reducer) e o tempo relativo da viagem como valor. O *reducer* recebe todos os tempos e calcula a média, gerando a estimativa de tempo inicial. Na sequência, é utilizada a equação de regressão para gerar a estimativa final.

4.3 Regressão Polinomial

Para resolver o problema dos mínimos quadrados e estimar os coeficientes da regressão, será utilizada uma variação da decomposição QR, a decomposição *Tall and Skinny QR* (TSQR), que é a mais adequada para o conjunto de dados utilizado no trabalho, uma matriz com um número de linhas muito maior que o número de colunas.

4.3.1 Decomposição TSQR

A decomposição TSQR é baseada no algoritmo *communication avoiding QR*, proposto por Demmel et al. (2008). Uma abordagem que evita comunicação é a ideal para

ambientes de computação em grid, onde o *overhead* de comunicação é muito mais custoso que a computação em processadores individuais.

A principal ideia por trás da decomposição TSQR é calcular pequenas fatorações QR em blocos de linhas da matriz original e combinar as matrizes R resultantes. Em seguida, é feita outra rodada de fatorações QR, esse processo é repetido até que a matriz R final seja computada. A matriz ortogonal Q nunca é formada explicitamente, o que não é necessário para resolver um problema de mínimos quadrados.

4.3.2 Paralelização da decomposição TSQR

A paralelização da decomposição TSQR e sua implementação no ambiente MapReduce foram feitas conforme proposto por Constantine and Gleich (2011).

Supondo uma matriz A com 8n linhas e n colunas particionada como segue:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \\ \boldsymbol{A}_4 \end{bmatrix}$$

Neste particionamento, cada Ai é uma matriz *2n x n*. Assume-se que cada matriz Ai é pequena o suficiente tornando viável a sua decomposição QR utilizando apenas um processador. A fatoração de cada uma das matrizes Ai resulta na seguinte fatoração de A:

$$\boldsymbol{A} = \underbrace{\begin{bmatrix} \boldsymbol{Q}_1 & & & \\ & \boldsymbol{Q}_2 & & \\ & & \boldsymbol{Q}_3 & \\ & & & \boldsymbol{Q}_4 \end{bmatrix}}_{8n \times 4n} \underbrace{\begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \boldsymbol{R}_3 \\ \boldsymbol{R}_4 \end{bmatrix}}_{4n \times n}$$

Essa fatoração, no entanto, ainda não é uma fatoração QR de A. Mas a fatoração QR de A pode ser obtida com a fatoração da matriz R computada no passo anterior (CONSTANTINE; GLEICH, 2011):

$$A = \overbrace{\underbrace{\begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & Q_3 & \\ & & & Q_4 \end{bmatrix}}_{8n \times 4n}}^{=Q} \underbrace{\tilde{Q}}_{4n \times n} \quad \underbrace{\tilde{R}}_{n \times n}$$

A mesma ideia pode ser aplicada se a matriz R fatorada no passo anterior for muito grande para ser fatorada por um único processador:

$$\underbrace{\begin{bmatrix} R_1 \\ R_2 \\ \hline R_3 \\ R_4 \end{bmatrix}}_{4n \times n} = \underbrace{\left[\begin{array}{cc} Q_5 & \\ \hline & Q_6 \end{array}\right]}_{4n \times 2n} \underbrace{\begin{bmatrix} R_5 \\ \hline R_6 \end{bmatrix}}_{2n \times n}.$$

Esse exemplo mostra que a única operação necessária é a fatoração QR de A é a fatoração QR da matriz *2n x n* acima. Seguindo essa ideia, é possível fatorar QR com o máximo de paralelismo:

$$A = \overbrace{\underbrace{\begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & Q_3 & \\ & & & Q_4 \end{bmatrix}}_{8n \times 4n} \underbrace{\left[\begin{array}{cc} Q_5 & \\ \hline & Q_6 \end{array}\right]}_{4n \times 2n} \underbrace{Q_7}_{2n \times n}}^{=Q} \quad \underbrace{R_7}_{n \times n}$$

4.3.3 TSQR Serial

Demmel et al. (2008) demonstram que as fatorações QR dos blocos de uma matriz A podem ser feitas em qualquer ordem. Por exemplo, supondo que as linhas da matriz A são lidas uma de cada vez e também que há capacidade para armazenar $3n$ linhas de A e computar a fatoração QR para essas linhas armazenadas. Essa sequência de fatorações QR produz a matriz R:

$$A_1 = Q_1 R_1; \begin{bmatrix} R_1 \\ A_2 \end{bmatrix} = Q_2 R_2; \begin{bmatrix} R_2 \\ A_3 \end{bmatrix} = Q_3 R_3; \begin{bmatrix} R_3 \\ A_4 \end{bmatrix} = Q_4 R_4$$

O algoritmo que faz essa computação deve primeiramente ler e armazenar as 2n linhas de $A1$. Então a fatoração QR desse bloco pode ser computada, produzindo uma matriz *n x n R1*. Em seguida, é feita a leitura das $2n$ linhas correspondentes à $A2$. Neste momento, o buffer está cheio, mas é feita uma compressão para uma nova matriz $R2$ através de uma segunda fatoração QR. O processo continua até que não reste mais linhas em A. Em resumo, a computação é a seguinte:

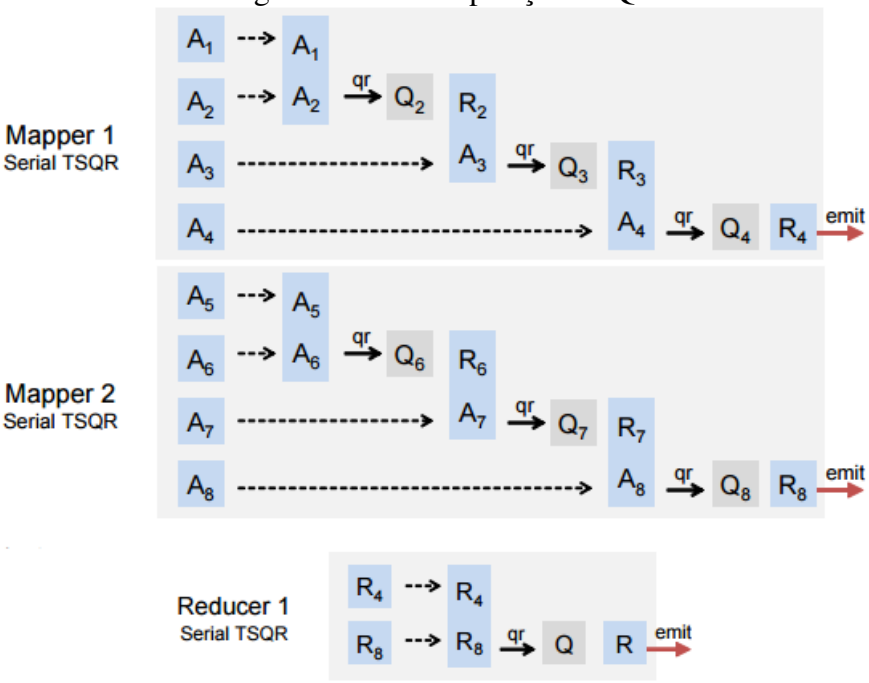
$$A = \overbrace{\underbrace{\begin{bmatrix} Q_1 & & \\ & I_{2n} & \\ & & I_{2n} \\ & & I_{2n} \end{bmatrix}}_{8n \times 7n} \underbrace{\begin{bmatrix} Q_2 & & \\ & I_{2n} & \\ & & I_{2n} \end{bmatrix}}_{7n \times 5n} \underbrace{\begin{bmatrix} Q_3 & \\ & I_{2n} \end{bmatrix}}_{5n \times 3n} \underbrace{Q_4}_{3n \times n}}^{=Q} \underbrace{R}_{n \times n}$$

4.3.4 Implementação MapReduce TSQR

Para implementar a decomposição TSQR em uma arquitetura MapReduce, são combinadas as duas abordagens apresentadas anteriormente: uma computação serial e uma computação totalmente paralela da fatoração TSQR.

Cada *mapper* executa uma fatoração TSQR serial, processando uma linha da matriz de cada vez. Depois de processar todas as linhas do bloco, é gerado como saída a matriz R referente a essas linhas. As fatorações QR internas são feitas utilizando a função linalg.qr da biblioteca NumPy, que faz a decomposição através de transformações de Householder.

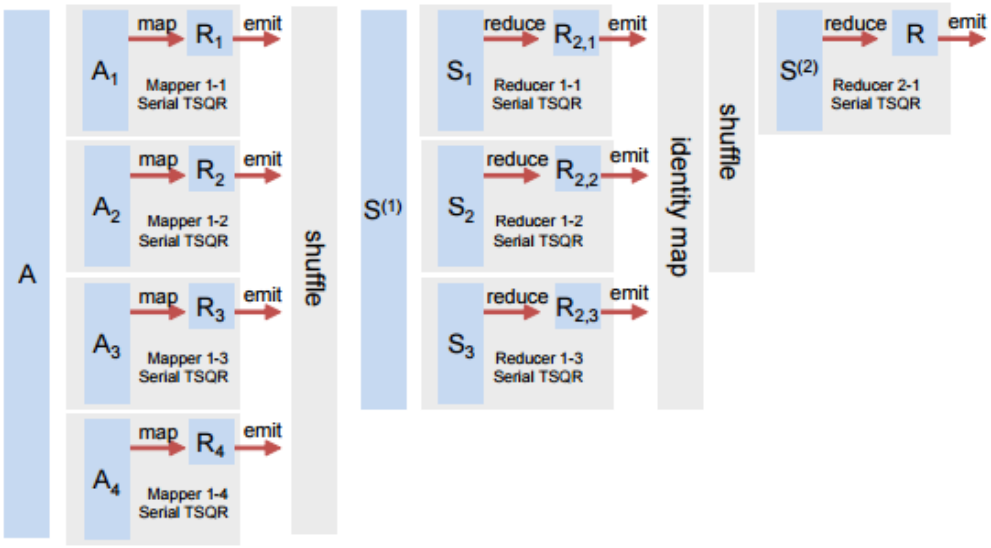
Figura 4.1: Decomposição TSQR



Fonte: Constantine et Gleich (2011, p.45)

Quando esse programa MapReduce roda com apenas um *reducer*, a saída desse *reducer* é a matriz R final. Utilizar muitos *mappers* e apenas um *reducer*, entretanto, reduz oportunidades de paralelismo, pois o *reducer* tem que computar a saída de todos os *mappers*, e essa pode ser uma longa computação serial. Para otimizar essa computação, é usada uma abordagem com mais de uma iteração. Utilizando mais de uma iteração, a saída dos *mappers* é enviada para vários *reducers*. E a saída desses *reducers* é matriz de entrada para uma nova fatoração TSQR. Essa iteração utiliza um *identity mapper* para eliminar trabalho extra relacionado ao processamento da saída do *reduce* anterior. O *identity mapper* é um *mapper* que implementa a função matemática identidade, ou seja, envia diretamente para a saída os pares chave-valor de entrada. Quando a iteração tiver apenas um *reducer*, é gerada a matriz R final.

Figura 4.2: Decomposição TSQR com mais de uma iteração



Fonte: Constantine e Gleich (2011, p. 47)

O último passo para estimar os coeficientes da equação é resolver o sistema linear $Rx = Q^T b$. Essa implementação não forma explicitamente a matriz Q, porém é possível calcular $Q^T b$ utilizando a mesma ideia apresentada anteriormente para paralelizar a decomposição QR. Considerando a matriz *A 4n x n* do exemplo acima e $b = [b_1^T, b_2^T, b_3^T, b_4^T]$, com decomposição QR apresentada nessa seção, temos:

$$\mathbf{Q^T b} = \mathbf{Q_7^T} \begin{bmatrix} \mathbf{Q_5^T} & & \\ & \mathbf{Q_6^T} & \end{bmatrix} \begin{bmatrix} \mathbf{Q_1^T} & & & \\ & \mathbf{Q_2^T} & & \\ & & \mathbf{Q_3^T} & \\ & & & \mathbf{Q_4^T} \end{bmatrix} \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \mathbf{b_3} \\ \mathbf{b_4} \end{bmatrix}$$

Em um ambiente MapReduce, isso significa que os valores de b estão armazenados com as linhas correspondentes de A. Desta forma, é possível formar $Q^T b$ adicionando $Q_i^T b_i$ na saída de cada pequena fatoração QR. Depois de obtida a matriz R e o vetor $Q^T b$, o sistema é resolvido utilizando a função linalg.solve da biblioteca NumPy.

4.4 Considerações Finais

Neste capítulo foi descrita a implementação de uma aplicação paralela do modelo descrito no capítulo 3. O cálculo da velocidade média de cada hora e o treinamento da regressão são pré-processados. Em tempo de execução, são computados os vizinhos da viagem de entrada. Todos os vizinhos têm seu tempo de viagem corrigido pelo fator

de escala - calculado a partir das velocidades médias previamente pré-processadas - e a estimativa de tempo inicial é calculada. Em seguida, é utilizada a equação gerada pela regressão linear para gerar a estimativa final.

O próximo capítulo descreve os experimentos realizados e apresenta os resultados relativos à precisão do modelo e ao desempenho da aplicação.

5 EXPERIMENTOS E RESULTADOS

Neste capítulo são apresentados os resultados dos experimentos realizados. Na seção 5.1 é avaliada a precisão do modelo. Em 5.2 é feita a análise de desempenho com foco nos aspectos da paralelização da aplicação, os experimentos foram realizados em um cluster na plataforma Azure da Microsoft.

5.1 Avaliação do modelo

Com o propósito de verificar a melhoria da estimativa do modelo de Wang et al. (2013) são utilizadas as mesmas métricas para avaliação do resultado: erro absoluto médio (EAM), erro relativo médio (ERM), mediana do erro absoluto (MedEA) e mediana do erro relativo (MedER).

Foram testadas diversas equações de regressão a fim de definir a que mais acrescentaria precisão ao modelo, os melhores resultados são apresentados na tabela 5.1. A primeira linha apresenta os resultados do modelo de Wang et al. (2013), que não utiliza qualquer informação climática. Nas linhas seguintes são apresentados os resultados do modelo com variáveis climáticas usando diferentes equações de regressão. Nas equações abaixo, f representa o fator de escala da velocidade, m é a velocidade média do horário, v é a velocidade da viagem, c é o nível de chuva, n é a quantidade de neve e w é a velocidade do vento.

Tabela 5.1: Avaliação do modelo

Modelo	EAM (s)	ERM	MedEA (s)	MedER
Wang et al. (2013)	130,3502	0,1978	83,283	0,1633
$f = x_1m + x_2v + x_3^2c + x_4^2n + x_5w$	115,566	0,1751	78,489	0,1539
$f = x_1m + x_2v + x_3c + x_4^2n + x_5^2w$	114,378	0,1733	77,724	0,1524
$f = x_1m + x_2v + x_3^2c + x_4n + x_5^2w$	112,992	0,1712	76,908	0,1508
$f = x_1m + x_2v + x_3^2c + x_4^2n + x_5^2w$	112,794	0,1709	76,041	0,1491

O erro calculado considerando a mediana é significativamente menor devido a algumas viagens com tempo muito diferente do normal, causado por situações anormais de trânsito, como acidentes. Conforme é possível observar na tabela 5.1, com a equação $f = x_1m + x_2v + x_3^2c + x_4^2n + x_5^2w$, em comparação com o modelo de Wang et al. (2013), o erro médio é 15,56% menor e a mediana do erro é 9,52% menor. O modelo proposto por este trabalho diminui mais o erro médio que a mediana do erro devido ao fato que leva em consideração situações adversas de clima, que são responsáveis por parte das viagens

com tempo muito diferente do normal. Ainda assim, a mediana do erro é menor que o erro médio devido a situações anormais de trânsito causadas por qualquer outro motivo que não seja o clima.

5.2 Avaliação de desempenho

Esta seção tem o objetivo de medir os ganhos obtidos com a paralelização da implementação. São avaliados os resultados do treinamento da regressão, que é a parte da aplicação que demanda maior capacidade de processamento. Foram feitas diversas execuções para medir o desempenho com base na variação dos seguintes atributos:

- Número de nós (1, 2, 4, 6, 8);
- Tamanho do conjunto de dados (1 GB, 8 GB, 16 GB, 32 GB);
- Tamanho do bloco (64 MB, 256MB, 512MB).

5.2.1 Ambiente de execução

Os experimentos foram conduzidos em um cluster na plataforma Microsoft Azure. O cluster é formado por computadores com processador Intel Xeon® E5-2673 v3 (Haswell) de 2.4 GHz (com a Intel Turbo Boost Technology 2.0 pode chegar a até 3.1 GHz), 8GB de memória RAM, sistema operacional Ubuntu 16 e Hadoop 2.7.

O número de nós utilizados varia conforme o experimento. Todos os resultados são a média de 5 repetições. O número de repetições é baixo devido a limitações na utilização de recursos da Azure.

5.2.2 Critérios de avaliação

Para avaliar o desempenho com base na variação do número de máquinas, mantendo constante o tamanho da entrada, foram calculados o *speedup* e a eficiência. O *Speedup* avalia o aumento de desempenho obtido pela aplicação quando executada por mais de um processador, é definido como *Sp = T1 / Tp* (tempo de execução com um processador sobre tempo com p processadores). Já a eficiência indica a efetiva utilização dos processadores, é definida como *Ep = Sp / p* (*speedup* sobre número de processadores).

Para avaliar a escalabilidade, o número de máquinas é mantido constante e o tamanho da entrada varia. Uma aplicação é dita escalável quando demonstra a capacidade de manter a mesma eficiência à medida que o número de processadores e a complexidade do problema aumentam proporcionalmente.

Também são avaliados os resultados da variação do tamanho do *split*. *Split* é uma partição da entrada enviada a um *mapper*. O tamanho do *split* determina o número de *mappers* disparados pelo Hadoop.

5.2.3 Análise dos resultados

Para medir o desempenho da aplicação foram utilizados conjuntos de dados de 1 GB, 8 GB, 16 GB e 32 GB. Para cada conjunto de dados foram feitos experimentos utilizando 1, 2, 4, 6 e 8 máquinas. Nas figuras 5.1 a 5.4 são apresentados os gráficos com tempo de execução em função do tamanho da entrada e do número de nós, *speedup*, eficiência e tempo de execução em função do tamanho do *split*. Detalhes sobre cada experimento estão descritos no apêndice A.

Figura 5.1: Tempo de execução

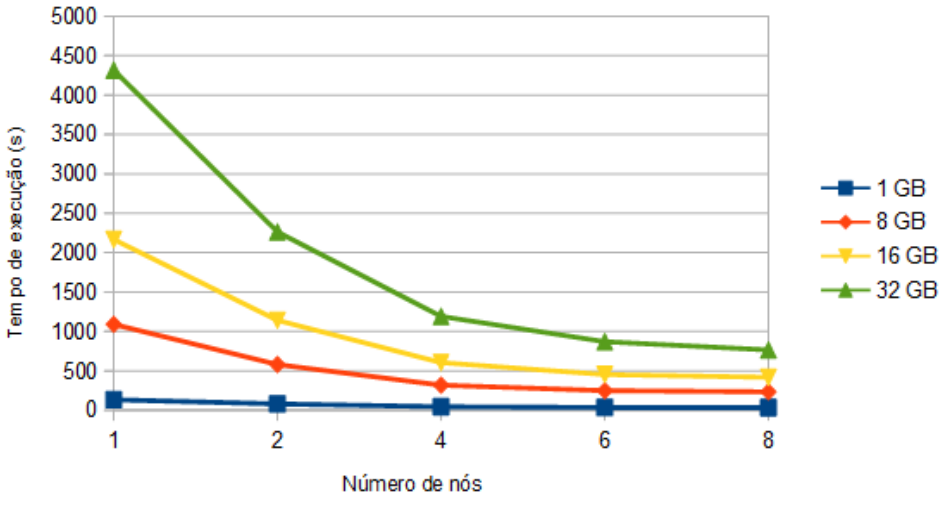
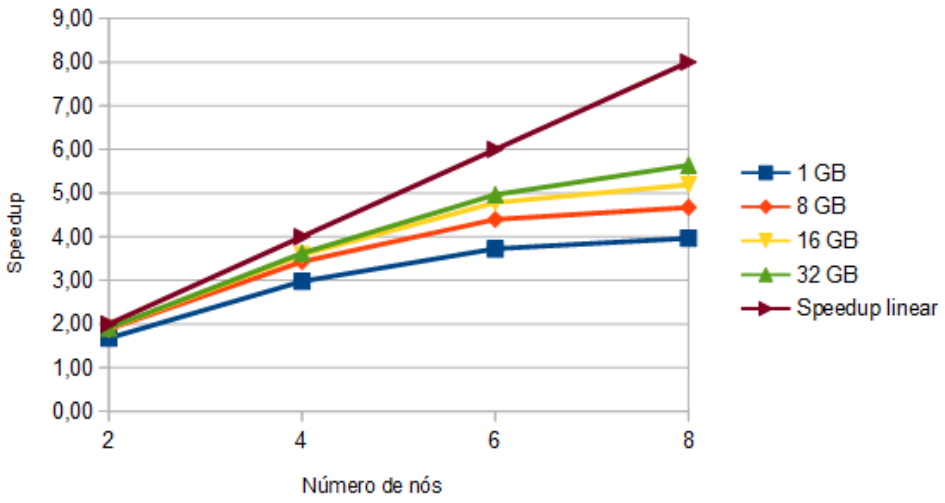
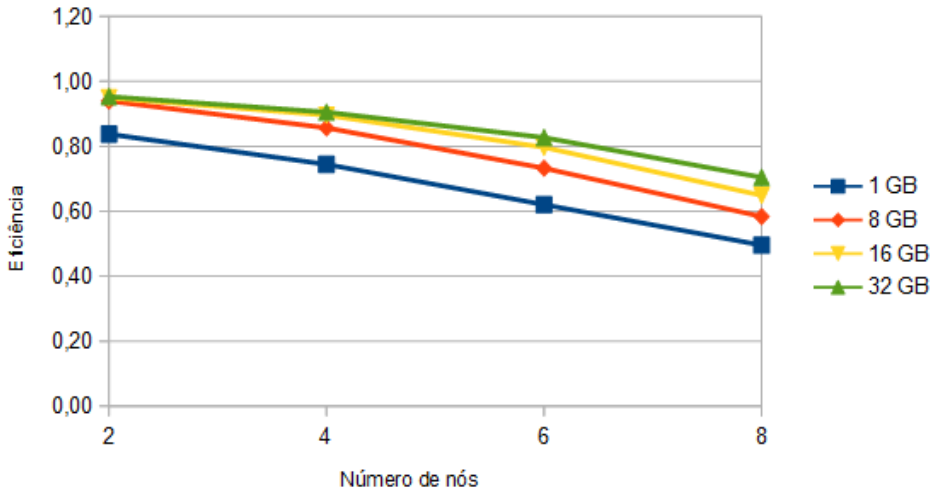


Figura 5.2: *Speedup*



Na Figura 5.1 são exibidos os tempos de execução de acordo com o tamanho da entrada e o número de nós utilizados. A Figura 5.2 mostra o *speedup* dessas mesmas execuções. Analisando essas duas figuras, nota-se uma diminuição do tempo de execução conforme mais nós são adicionados. Por exemplo, a execução com uma entrada de 16 GB leva 4316 segundos quando feita com apenas um nó e 870 segundos quando executada em 6 nós, um *speedup* de 4,96. O *speedup* foi calculado em relação à aplicação paralela executada em um nó do cluster. Se fosse calculado em relação à aplicação sequencial, sem qualquer *overhead* provocado pela utilização do Hadoop, seria um pouco menor.

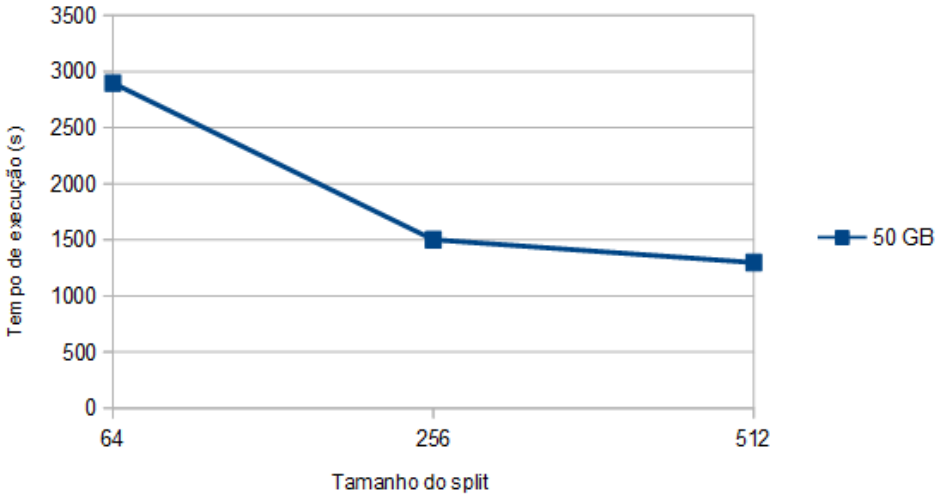
Figura 5.3: Eficiência



O principal fator que define a escalabilidade da aplicação é a abordagem com mais de uma iteração descrita na seção 4.3.4 e ilustrada na Figura 4.2. O número de

mappers cresce proporcionalmente com o aumento do tamanho dos dados de entrada. Na abordagem com apenas uma iteração, em um caso com muitos *mappers*, o *reducer* teria que fazer uma longa computação serial com as saídas de todos os *mappers*. Com mais de uma iteração, essa computação é paralelizada e isto garante a escalabilidade da aplicação.

Figura 5.4: Tempo x tamanho do split



Na Figura 5.4 são observadas reduções de tempo significativas com o aumento do tamanho do *split*. Isso se deve ao fato de que utilizar splits de tamanho maior reduz o *overhead* do Hadoop pois é necessário gerenciar um número menor de *mappers*.

Esta seção mostra que é possível diminuir consideravelmente o tempo de execução através de paralelização, Além disso, um ganho ainda maior de desempenho pode ser obtido se a mesma aplicação for programada em alguma linguagem mais eficiente que Python. Se for escrita em Java, pode ser utilizada a implementação padrão do Hadoop, que possui menos *overhead* que o Hadoop Streaming.

5.3 Considerações Finais

Neste capítulo foram apresentados os resultados dos experimentos realizados. Os resultados obtidos confirmam que o modelo de predição de tempo pode ser melhorado se levar em consideração informações do clima e que a aplicação pode ser executada em muito menos tempo devido à paralelização.

No próximo capítulo serão feitas considerações finais sobre o trabalho e sobre trabalhos futuros.

6 CONCLUSÃO

Este trabalho propõe uma aplicação que cruza dados de viagens de táxi com dados climáticos para identificar a influência de algumas variáveis climáticas no tempo de viagem. A aplicação proposta faz uma estimativa de tempo de uma viagem a ser realizada baseada em dados históricos de viagens realizadas em condições similares.

Uma estimativa inicial é feita utilizando o modelo de Wang et al. (2013), que leva em consideração viagens similares geograficamente e temporalmente. Em seguida, é utilizado um modelo de regressão para estimar influência das variáveis climáticas no tempo da corrida e gerar a estimativa final.

A aplicação foi desenvolvida em Python. A programação foi feita utilizando o modelo MapReduce para que fosse possível a execução em paralelo. Para o uso de Python com Hadoop foi utilizada a API Hadoop Streaming.

Diversos experimentos foram realizados para verificar a precisão e o desempenho da aplicação. Os resultados confirmam que é possível melhorar a estimativa de uma viagem de táxi quando variáveis climáticas são adicionadas ao modelo de predição. Conforme foi demonstrado na seção 5.1, o erro médio diminui 15,56% e a mediana do erro diminui 9,52% quando adicionamos variáveis climáticas ao modelo. Além disso, também é mostrado que o tempo de exeucão diminui quando a execução é feita em mais máquinas.

6.1 Trabalhos futuros

Conforme mencionado na seção 3.1, os dados de viagens realizadas por aplicativos de transporte individual não foram utilizados neste trabalho. O motivo é que as únicas informações disponíveis são o horário e local inicial da viagem. Caso esses dados passem a ser disponibilizados também com horário final, coordenada de destino e distância da viagem poderiam ser integrados ao trabalho.

REFERÊNCIAS

ADJOUT, M.; BOUFARES, M. Parallel implementation of multiple linear regression algorithm based on mapreduce. 2015.

ANDERSEN, O.; TORP, K. A data model for determining weather's impact on travel time. 2016.

BROEK, T. V. den; HUIJBOOM, N. Open data: an international comparison of strategies. 2011.

BUN, L.; MCDONALD, J. Determinants of commuting time and distance for seoul residents: The impact of family status on the commuting of women. 2003.

CONSTANTINE, P.; GLEICH, D. Tall and skinny qr factorizations in mapreduce architectures. 2011.

CUKIER, K.; SCHONBERGER, V. Big data: A revolution that will transform how we live, work, and think. 2013.

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. 2008.

DEMCHENKO, Y.; NGO, C.; MEMBREY, P. Architecture framework and components for the big data ecosystem. 2013.

DEMMEL, J. et al. Communication-avoiding parallel and sequential qr. 2008.

IBM. 10 key marketing trends for 2017. 2017.

INOUBLI, W. et al. An experimental survey on big data frameworks. 2017.

LANEY, D. 3d data management: Controlling data volume, velocity, and variety. 2001.

NOOKALA, L. Weather impact on traffic conditions and travel time prediction. 2006.

STERN, A.; SHAH, V.; GOODWIN, L. Analysis of weather impacts on traffic flow in metropolitan washington dc. 2013.

TSAPAKIS, I.; CHENG, T.; BOLBOL, A. Impact of weather conditions on macroscopic urban travel times. 2012.

TU, H.; LINT, J. W. C. V.; ZUYLEN, J. V. The impact of adverse weather on travel time variability of freeway corridors. 2007.

UYANIK, G.; GULER, N. A study on multiple linear regression analysis. 2013.

WANG, H. et al. A simple baseline for travel time estimation using large-scale trip data. 2013.

WHITE, T. Hadoop the definitive guide. 2010.

WU, X. et al. Data mining with big data. 2013.

ZHAO, L.; CHIEN, S. Analysis of weather impact on travel speed and travel time reliability. 2012.

APÊNDICE A — EXPERIMENTOS

Tabela A.1: Tempos de execução, *speedup* e eficiência

Quantidade de dados (GB)	Número de nós	Tempo de execução (s)	*Speedup*	Eficiência
1	1	134	-	-
1	2	80	1,68	0,84
1	4	45	2,98	0,74
1	6	36	3,72	0,62
1	8	34	3,96	0,50
8	1	1090	-	-
8	2	580	1,88	0,94
8	4	318	3,43	0,86
8	6	248	4,40	0,73
8	8	233	4,67	0,58
16	1	2167	-	-
16	2	1142	1,90	0,95
16	4	605	3,58	0,90
16	6	453	4,78	0,80
16	8	418	5,19	0,65
32	1	4316	-	-
32	2	2264	1,91	0,95
32	4	1192	3,62	0,91
32	6	870	4,96	0,83
32	8	766	5,64	0,70