UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA DE COMPUTAÇÃO

HENRIQUE DE SOUZA SAGIORATTO

APRIMORANDO O DESEMPENHO DE REDES EM AMBIENTES DE NUVEM ATRAVÉS DO ROTEAMENTO POR MÚLTIPLOS CAMINHOS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO

2021

HENRIQUE DE SOUZA SAGIORATTO

APRIMORANDO O DESEMPENHO DE REDES EM AMBIENTES DE NUVEM ATRAVÉS DO ROTEAMENTO POR MÚLTIPLOS CAMINHOS

Improving Network Performance in Cloud Environments Through Multipath Routing

Trabalho de Conclusão de Curso, apresentada ao Curso de Graduação em Engenharia de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau em " Engenharia de Computação"

Orientador: Prof. MSc Luís Cassiano Goularte Rista

Co-orientador:Prof. Dr. Fábio Favarim

PATO BRANCO

2021

HENRIQUE DE SOUZA SAGIORATTO

APRIMORANDO O DESEMPENHO DE REDES EM AMBIENTES DE NUVEM ATRAVÉS DO ROTEAMENTO POR MÚLTIPLOS CAMINHOS

Trabalho de Conclusão de Curso, apresentada ao Curso de Graduação em Engenharia de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau em " Engenharia de Computação"

Data de aprovação: 06 de dezembro de 2021

Luís Cassiano Goularte Rista
Mestrado em Ciências da Computação
Universidade Tecnológica Federal do Paraná

Fábio Favarim
Doutorado em Engenharia Elétrica
Universidade Tecnológica Federal do Paraná

Marcelo Teixeira
Doutorado em Engenharia de Automação e Sistemas
Universidade Tecnológica Federal do Paraná

Eden Ricardo Dosciatti
Doutorado em Engenharia Elétrica e Informática Industrial
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2021

RESUMO

SAGIORATTO, Henrique de Souza. APRIMORANDO O DESEMPENHO DE REDES EM AMBIENTES DE NUVEM ATRAVÉS DO ROTEAMENTO POR MÚLTIPLOS CAMINHOS. 60 f. Trabalho de Conclusão de Curso – Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Pato Branco, 2021.

Embora a computação em nuvem pareça ser algo novo, a sua ideia é relativamente antiga, surgiu na década de 60 a partir das ideias de pioneiros como J.C.R. Licklider e John McCarthy. Desde então sua praticidade, flexibilidade e eficiência, tem cada vez mais chamado à atenção da indústria e da academia. Entretanto um problema que deve ser levado em consideração é a queda de desempenho devido à camada de virtualização, havendo a necessidade de uma grande largura de banda, e o custo para contratar tal recurso. Um dos métodos encontrados seria a agregação de *links* usada para aumentar a largura de banda. Este trabalho propõe um aprimoramento à agregação de *links*, usufruindo do MPTCP para a alocação de recursos em nuvem. Entretanto a implementação de um cenário para tal melhoria é inviável, sem antes ter uma noção de como será o comportamento da rede para a melhoria. Para isso, é indispensável um modelo, capaz de prover um comportamento próximo à realidade, por esse motivo usamos Redes de Petri, uma ferramenta gráfica e matemática, podendo assim avaliar o desempenho da rede perante a tecnologia do MPTCP.

Abstract: *Although cloud computing seems to be something new, its idea is relatively old, it emerged in the 60's from the ideas of pioneers like J.C.R. Licklider and John McCarthy. Since then, its practicality, flexibility and efficiency have attracted more and more attention from industry and academia. However, a problem that must be taken into account is the drop in performance due to the virtualization layer, with the need for a large amount of bandwidth, and the cost of hiring such a resource. One of the methods found would be the aggregation of links used to increase bandwidth. This work proposes an improvement to the aggregation of links, taking advantage of MPTCP for the allocation of resources in the cloud. However, the implementation of a scenario for such improvement is unfeasible, without first having a notion of how the network will behave for the improvement. For this, a model, capable of providing a behavior close to reality, is essential. For this reason, we use Petri nets, a graphical and mathematical tool, thus being able to evaluate the performance of the net with the MPTCP technology.*

Palavras-chave: MPTCP, Redes de Petri, Hadoop, Computação em Nuvem, Escalabilidade

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

QoS	Qualide de Serviço
MPTCP	*Multipath TCP*
GSPN	*Generalized Stochastic Petri Nets* / Redes de Petri Estocásticas Generalizadas
NIST	*National Institute of Standards and Technology*
CPU	*Central Processing Unit*
ANS	Acordo de Nível de Serviço
TCP	*Transmission Control Protocol* / Protocolo de Controle de Transmissão
LTE	*Long Term Evolution*
SYN	*Synchronize* / Sincronizar
ACK	*Acknowledgement* / Confirmação
LACP	*Link Aggregation Protocol*
LXC	*Linux Container*
VM	*Virtual Machine*
KVM	*Kernel-based Virtual Machine*
API	*Application Programming Interface*
HDFS	*Hadoop Distributed File System*

1 INTRODUÇÃO

O conceito de computação em nuvem teve seu inicio na década de 60 com John McCarthy conhecido como o "pai da inteligência artificial"e inventor da programação Lisp, que sugeriu que a computação fosse oferecida como um serviço público, pagando apenas pelo que se usa. Em 1962, Joseph Carl Robnett Licklider encontrou um método para compartilhar dados em escala global, criando assim a rede Arpanet, sendo ambos vistos como pioneiros da computação em nuvem. Tal conceito ficou adormecido por algumas décadas, foi então em 1997 que o termo "computação em nuvem" foi utilizado por Ramnath Chellappa, em uma palestra acadêmica como visto em (VIANA, 2013). Com o crescimento da Internet, a computação em nuvem vem se tornando cada vez mais forte na indústria e na academia.

Os serviços em nuvem apresentam diversas vantagens para os usuários, tais como: previsibilidade e custos mais baixos, proporcionais à qualidade de serviço (QoS) e cargas de trabalho reais; complexidade técnica reduzida graças às interfaces de acesso unificado e administração simplificada; e elasticidade e escalabilidade, proporcionando a percepção de recursos quase infinitos. Por outro lado, o provedor tem que garantir a ilusão de recursos infinitos sob cargas de trabalho dinâmicas e minimizar os custos operacionais associados a cada usuário (SOUSA et al., 2010).

Contudo, a nuvem apresenta uma variabilidade de desempenho bastante elevada (SCHAD et al., 2010), principalmente devido à camada de virtualização, exigindo boa largura de banda, alta taxa de transferência e baixa latência. Devido a isso, pode ser quase inviável sua utilização por aplicativos de alto desempenho, como *big data*, pois não é possível prever como será dado o uso dos serviços pelos provedores de serviços.

Desse modo, se fez necessário uma solução que aumentasse a largura de banda, a taxa de transferência e melhorasse a latência, sem uma grande atualização física, pois isso seria muito caro e geraria transtornos para os provedores de serviços e seus clientes.

Tendo em vista tais problemas, (RISTA et al., 2017) propuseram uma solução

baseada em agregação de *links*, a qual consiste em uma técnica usada para o acoplamento de dois ou mais canais *Ethernet* em paralelo para a fim de se obter um único canal com maior capacidade, aumentando a largura de banda da rede, obtendo assim uma melhor taxa de transferência e latência, sem a necessidade de modificações da infraestrutura.

O modelo de Redes de Petri proposto por (RISTA et al., 2018), visa estimar a utilização da largura de banda da rede em diferentes configurações do sistema como a sobrecarga imposta pela adição de instâncias de nuvem baseadas em *containers*, no contexto de soluções de agregação de *links*. O objetivo principal visa aumentar a largura de banda da rede e, consequentemente, obter melhor taxa de transferência e latência. Além da descrição da estrutura de modelagem, também foi demonstrado um conjunto de experimentos para avaliar as várias configurações diferentes do sistema. Em seguida, a partir de uma infraestrutura de rede real com base nas sugestões do modelo, foram demonstrados também como as estimativas afetariam o desempenho do sistema na prática. Os resultados revelaram que o modelo pode antecipar, com razoável precisão, as características estruturais e comportamentais da computação em nuvem. Assim, a eficiência da rede pode ser melhorada sistematicamente sem a necessidade de modificações de infraestrutura.

Desse mesmo modo para conseguirmos estimar os resultados que foram obtidos com o *Multipath TCP* (MPTCP), é indispensável a criação de um modelo para a simulação de diversas configurações, pois a implementação física de um grande cenário é inviável financeiramente, e testes de diversas configurações é impraticável no quesito tempo. Foi usado então Redes de Petri, uma ferramenta gráfica e matemática (CARDOSO; VALETTE, 1997), podendo-se então prever o desempenho das configurações desejadas.

Para a validação do modelo, é fundamental um cenário de testes capaz de fornecer o comportamento real do sistema. Para isso é apresentado uma série de experimentos, tanto no ambiente nativo quanto no ambiente virtual. Com esses resultados foi possível validar o modelo, e dizer se é capaz de simular um cenário real. Podendo assim realizar diversos cenários de testes, sem a necessidade da implementação, e encontrar prováveis gargalos na rede, ou outros problemas.

1.1 OBJETIVO GERAL

Este trabalho propõe um aprimoramento da rede através do roteamento por múltiplos caminhos, ou seja, através do MPTPC. O MPTCP permite o uso simultâneo das interfaces de rede disponíveis por meio da criação de múltiplos sub-fluxos. Com o MPTCP a aplicação continua acessando a rede como se fosse uma única interface de rede, porém, na verdade os dados são distribuídos pelos sub-fluxos existentes. Os benefícios disso incluem melhor utilização de recursos, melhor rendimento e reação mais suave a falhas.

Entretanto antes de implementar a tecnologia do MPTCP, é necessário um modelo capaz de antecipar o comportamento da rede através das interfaces. Para isso este trabalho também tem por objetivo a criação de um modelo capaz de analisar o impacto que as estratégias do uso do MPTCP podem trazer sobre métricas como desempenho e eficiência da rede. Para poder assim estimar o comportamento da rede sem a necessidade de toda a implementação necessária para isso.

1.2 OBJETIVOS ESPECÍFICOS

- Modelar o comportamento do sistema através de Redes de Petri;
- Comparar o sistema modelado com o sistema real;
- Desenvolver um conjunto de experimentos de modo a verificar se a abordagem proposta é adequada para ambientes de nuvem;
- Avaliar o sistema proposto em uma aplicação de *big data* usando o sistema *Hadoop* em um ambiente baseado em containers.

1.3 JUSTIFICATIVA

O trabalho de (RISTA et al., 2018) apresenta um modelo em Redes de Petri capaz de avaliar a utilização do sistema com o uso de agregação de *links*. Validado com base no trabalho (RISTA et al., 2017), no qual *links* são agregados de forma a prover maior largura de banda para a aplicação, sendo a alocação dos recursos de rede de forma estática.

Como este trabalho aborda um aprimoramento da rede através do MPTCP, o modelo sugerido por (RISTA et al., 2018) não é válido para esse cenário. Sendo

indispensável a criação de um novo modelo para avaliar a utilização do sistema. E para validar tal modelo é fundamental uma série de experimentos, sendo preciso montar o cenário e coletar os resultados. Desse modo podendo chegar a um modelo válido para a simulação de diversos cenários sem a necessidade de implementar fisicamente, gerando uma economia de tempo e financeira.

2 REFERENCIAL TEÓRICO

Esse capítulo tem como objetivo, introduzir aos assuntos mais relevantes e necessários, para uma melhor compreensão do trabalho, apresentando conceitos, funcionamentos e as razões que levaram a escolher tais assuntos.

A computação em nuvem, em um mais alto nível, proporciona ao usuário a sensação de recursos quase infinitos. Entretanto, possui uma variabilidade de desempenho bastante elevada, necessitando de largura de banda, taxa de transferência e latência adequadas. Devido a esse problema (RISTA et al., 2017), sugere uma solução de escalabilidade através de agregação de *links*. Este trabalho tem por interesse apresentar um aprimoramento da rede através de múltiplos caminhos, ou seja, através do *Multipath TCP* (MPTCP). Contudo, para verificar se essa proposta é viável, tanto computacionalmente como em questão de tempo e gasto, é necessário um estudo prévio através de modelos, sendo possível avaliar o desempenho.

Em problemas do mundo real, desenvolver sistemas e a partir deles auferir informações quantitativas e qualitativas sobre suas características pode acarretar em custos e riscos excessivos. Construir e implementar um modelo de simulação, por sua vez, é uma alternativa que pode economizar recursos financeiros e de tempo, bem como minimizar o acontecimento de riscos. De posse desse modelo implementado, poderá ser possível realizar simulações próximas ou equivalentes às funcionalidades do sistema real. Desta forma, a construção do sistema pode ser realizada ao final, tomando como base as informações que forem extraídas dessas simulações.

Dessa forma, sendo um modelo a melhor saída encontrada para prever o funcionamento do MPTCP proposto por este trabalho, se faz então necessário o uso de alguma ferramenta para a construção de tal. Para isso foi feito a escolha de Redes de Petri, se fazendo essencial, antes de tudo, entender seu funcionamento.

2.1 REDES DE PETRI

Segundo (CARDOSO; VALETTE, 1997) a rede de Petri é uma ferramenta gráfica e matemática que se adapta a um grande número de aplicações em que as noções de eventos de evoluções simultâneas são importantes. Esta teoria nasceu da tese *Comunicação com autômatos*, defendida por Carl Adam Petri em 1962 na Universidade de *Darmstadt*, Alemanha.

Entre as aplicações pode-se citar: avaliação de desempenho, análise e verificação formal em sistemas discretos, protocolos de comunicação, controle de fabricação, concepção de software tempo real e/ou distribuído, sistemas de informação (organização de empresas), sistemas de transporte, logística, gerenciamento de base de dados, interface homem-máquina e multimídia (CARDOSO; VALETTE, 1997).

De acordo com (KARTSON et al., 2005) as Redes de Petri podem ser definidas a partir de dois tipos de nós, denominados lugares e transições. Um lugar é representado por um círculo e uma transição por uma barra vertical ou horizontal. Lugares e transições são conectados por arcos. O número de lugares é finito e não pode ser zero, assim como o número de transições. Um arco é direcionado e conecta um lugar a uma transição ou vice-versa. Em outras palavras, Redes de Petri são grafos direcionados, bipartidos e ponderados, que apresentam em sua estrutura uma marcação inicial e tem uma dinâmica associada. Extensões de Redes de Petri foram desenvolvidas para representar uma importante classe de processos dependentes de tempo (temporizados), tais como, canais de comunicação, processamento de código, projetos de hardware e fluxos de trabalho do sistema.

Para (KARTSON et al., 2005) nas GSPN (*Generalized Stochastic Petri Nets*/Redes de Petri Estocásticas Generalizadas), o tempo é representado por uma variável randômica, distribuída de forma exponencial associada a transições temporizadas. Quando o tempo é irrelevante para uma determinada transição, pode-se simplesmente usar transições não temporizadas (ou imediatas).

Formalmente uma GSPN (PETERSON, 1977) é uma sétupla $GSPN = (L, T, \Pi, E, S, M_0, Pe)$, em que:

- $L = \{l_1, l_2, \ldots, l_n\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, \ldots, t_m\}$ é um conjunto finito de transições;

- $\Pi : T \rightarrow \mathbb{N}$ é a função prioridade, em que:

$$\Pi(t) = \begin{cases} \geq 1, & \text{se } t \in T \text{ é imediata;} \\ 0, & \text{se } t \in T \text{ é temporizada.} \end{cases}$$

- $E : (T \times L) \rightarrow \mathbb{N}$ é a função entrada que define a multiplicidade dos arcos direcionados dos lugares às transições;

- $S : (T \times L) \rightarrow \mathbb{N}$ é a função saída que define a multiplicidade dos arcos direcionados das transições aos lugares;

- $M_0 : L \rightarrow \mathbb{N}$ é a marcação inicial da função. M indica o número de marcações em cada lugar, [1] ou seja, define o estado do modelo do sistema;

- $Po : T \rightarrow \mathbb{R}^+$ é a função ponderação que representa o peso das transições imediatas (Pe_t) ou as taxas de transições temporizadas (λ_t), onde:

$$Pe(t) = \begin{cases} Pe_t \geq 0, & \text{se } t \in T \text{ é imediata;} \\ \lambda_t > 0, & \text{se } t \in T \text{ é temporizada.} \end{cases}$$

A relação entre lugares e transições é estabelecida pelos conjuntos ${}^\bullet t$ e $t^\bullet$, definidos como segue.

Definição 1 *Dada uma transição $t \in T$, define-se:*

- ${}^\bullet t = \{l \in L \mid E(t,l) > 0\}$ *como pré-condições de t;*

- $t^\bullet = \{l \in L \mid S(t,l) > 0\}$*como pós-condições de t.*

Um estado de uma GSPN se altera quando uma transição dispara. Somente transições habilitadas podem ser disparadas. Transições *Imediatas* disparam assim que são habilitadas. As definições das regras de *habilitação* e *disparo* para transições são definidas abaixo.

Definição 2 (Regra de Habilitação) *Uma transição $t \in T$ é dita estar habilitada em uma marcação M se e somente se:*

- $\forall l \in {}^\bullet t, M(l) \geq E(t,l)$.

[1] Representados graficamente por pontos pretos que mostram uma marcação em um lugar.

Quando uma transição habilitada dispara, a marcação é removida da entrada para a saída (condições *pré* e *pós*).

Definição 3 (Regra de Disparo) *O disparo de uma transição $t \in T$ habilitado na marcação M leva a uma nova marcação M' de tal modo que* $\forall l \in ({}^\bullet t \cup t^\bullet), M'(l) = M(l) - E(t,l) + S(t,l)$.

Uma GSPN é dita ser *limitada* se houver um limite $k > 0$ no número de marcações em cada lugar. Portanto, garante que o espaço de estados resultante de uma GSPN limitada seja finito.

Quando o número de marcações em cada lugar de entrada l de t é N vezes o mínimo necessário para habilitar t ($\forall l \in {}^\bullet t, M(l) \geq N \times E(t,l)$, onde $N \in \mathbb{N}$ e $N > 1$), a transição é habilitada para disparar mais de uma vez. Nessa situação a transição t é dita estar habilitada com grau $N > 0$.

O disparo de transições pode usar uma das seguintes semânticas dinâmicas:

- *Servidor único*: N disparos sequenciais;
- *Servidor infinito*: N disparos paralelos;
- *Servidor múltiplo (k-servidores)*: a transição é habilitada até no máximo k vezes em paralelo; marcações que habilitam transições para um grau maior que k são manipuladas depois dos primeiros k disparos.

2.1.1 EXEMPLO DO USO DO GSPN

Para ilustrar o uso prático de GSPN, considere a seguir um exemplo que expressa a chegada de clientes na fila de um banco. Suponha que cada cliente consome um certo tempo na fila até que seja atendido por servidores do banco. A fila, portanto, varia de tamanho em função da taxa de chegada de clientes, da quantidade de atendentes, e do tempo médio que os atendentes, em conjunto, dão vazão ao fluxo.

Um dos objetivos poderia ser então estimar, para um dado fluxo de chegada de clientes, o tempo médio que cada um espera na fila, por exemplo. Por meio da mera variação da taxa de chegada de clientes ao banco, seria possível representar inúmeros cenários operacionais e, para cada um, obter estimativas úteis ao planejamento de capacidade de atendimento do banco. Para representar esse cenário, foi construída uma GSPN, como mostrado na Figura 1.

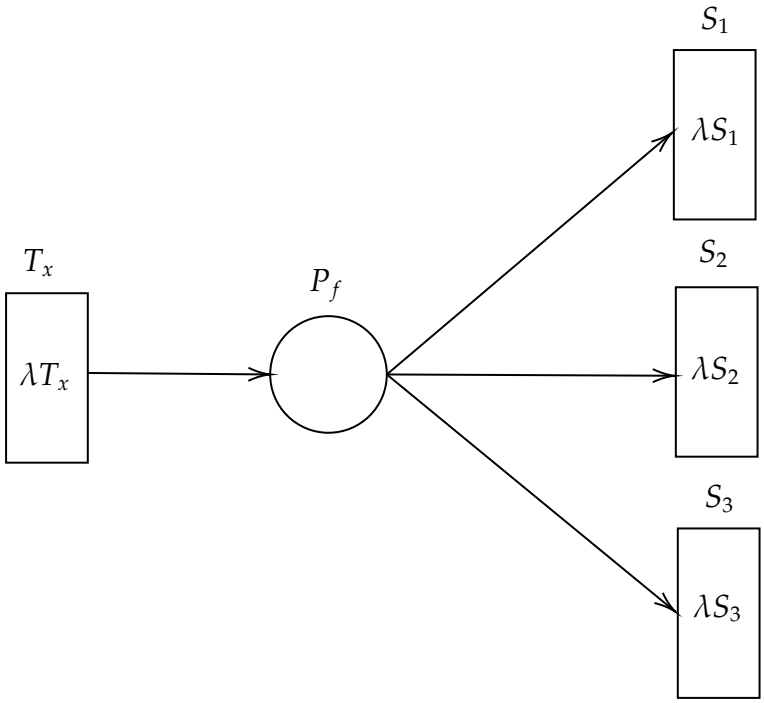

Figura 1: Comportamento de uma fila modelado em Redes de Petri

Nesse modelo, a transição T_x representa o fluxo de chegada de clientes. Essa transição é temporizada e dispara a uma taxa inversamente proporcional ao seu delay, λT_x (um número real atribuído ao delay da transição). Ao disparar, T_x libera tokens que são armazenados no lugar P_f. Esse lugar modela a fila do banco. Cada token inserido em P_f significa um cliente que ingressou na fila.

A partir de P_f, os clientes são atendidos por um dos servidores S_i, $i = 1, 2, 3$ (caixas do banco). Cada servidor demora um tempo específico para cada atendimento. Em GSPNs, esse tempo em geral corresponde ao tempo médio tomado por cada serviço e é modelado pelo delay λS_i da respectiva transição S_i. Quando o tempo médio não é interessante, como por exemplo quando o tempo de atendimento dos caixas apresenta um desvio padrão significativo, a GSPN pode ser adaptada a diferentes distribuições de probabilidade (e.g. *Erlang*, *Hipo* ou *Hiper Exponencial*, etc.) (DESROCHERS, 1994) a fim de absorver e melhor modelar essas características.

2.1.1.1 ANÁLISES

Com a GSPN construída, pode-se proceder com algumas simulações a fim de coletar estimativas de interesse. Assuma, por exemplo, que seja de interesse estimar o tempo que cada cliente aguarda na fila, para diferentes fluxos de chegada à fila.

Suponhamos que se queira avaliar, por exemplo, o tempo de fila para a chegada de um cliente a cada 5, 4, 3, 2 e 1 minuto. Suponhamos, ainda, que o tempo médio de atendimento de cada atendente seja de 2 minutos por cliente em um cenário, e de 3 minutos em outro cenário.

Então, para modelar o tempo de atendimento de 2 e 3 minutos, basta atribuir incrementalmente 2 e 3 aos delays λS_i, ou seja, $\lambda S_1 = \lambda S_2 = \lambda S_3 = 2$. Ainda, para modelar o fluxo de chegada, atribui-se incrementalmente os parâmetros 5, 4, 3, 2 e 1 ao delay λT_x da transição T_x, para cada cenário de chegada. Para cada variação é coletado o tempo de fila (TF), por meio da fórmula $TF = E\{\#\, P_f\} * \lambda T_x$, em que $E\{\#\, P_f\}$ captura a expectativa de o cliente estar na fila e a multiplica pela taxa de chegada, resultando no tempo médio que cada clientes gasta na fila. O resultado das simulações são apresentados na Tabela 1.

Tabela 1: Simulação do TF em min, para diferentes fluxos de chegada e diferentes tempos de atendimentos dos caixas

	Tempo atendimento	
Fluxo (min - cliente)	2 min	3 min
5	0,70	1,70
4	0,78	1,73
3	1,00	2,02
2	1,01	3,18
1	2,08	6,30

Por exemplo, a primeira linha dos resultados (linha 3) ilustra o caso da chegada de 1 cliente a cada 5 minutos. Nesse caso, o tempo de fila é menor, sendo de menos de 1 minuto (0,7 min) para a primeira configuração dos caixas (2 min) e de 1,7 min para a segunda configuração (3 min).

Na medida em que mais clientes ingressam na fila por minuto (incrementalmente nas linhas 4, 5, 6 e 7), o tempo médio de fila aumenta. O gráfico da Figura 2 explicita esse comportamento.

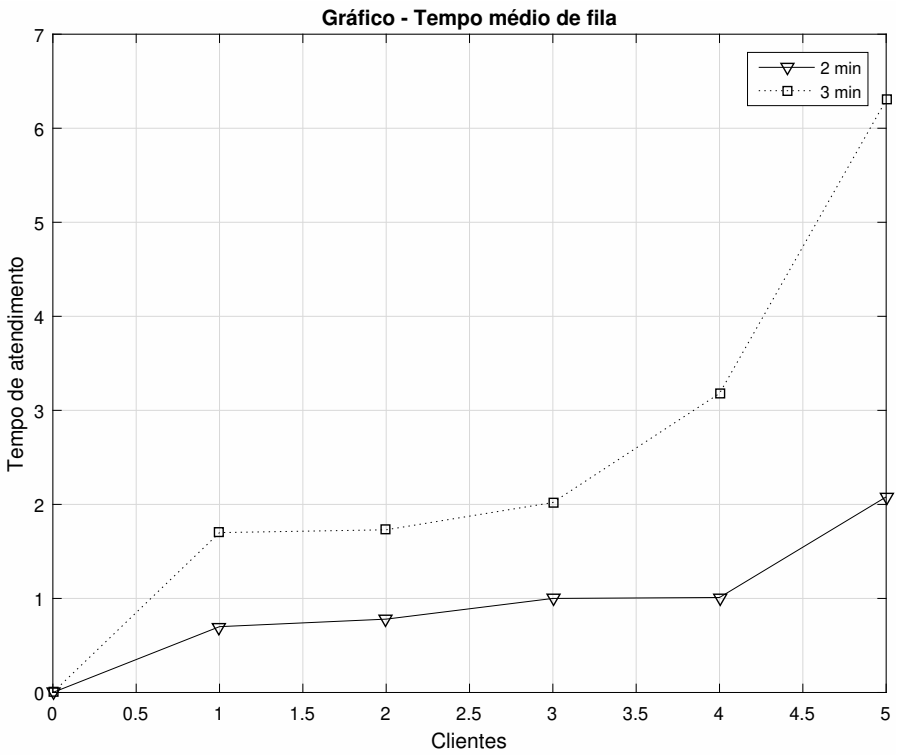

Figura 2: Aumento do tempo de fila para cada cenário avaliado

Embora esse exemplo não seja explicitamente relacionado a um sistema computacional real, ele pode ser contextualizado como tal. Para isso, pode-se interpretar a chegada de clientes como sendo requisições em um servidor, ou em um buffer de uma interface de rede, etc. Assim, se torna possível explorar inúmeras estimativas de cunho computacional, úteis ao planejamento de capacidade de sistemas.

Como pode ser visto em (RISTA et al., 2018), essa mesma analogia é usada para a construção do modelo, em que o modelo é usado para estimar a utilização da largura de banda da rede em diferentes configurações do sistema. Com base nas estimativas do modelo, é apresentado como o sistema pode ser modificado para criar melhorias de desempenho e planejamento de infraestrutura realista.

2.2 ELASTICIDADE

Ainda não se tem uma definição estabelecida por toda a comunidade, todavia o conceito mais aceito foi proposto pelo NIST (*National Institute of Standards and Technology*), sendo a capacidade de provisionar e desprovisionar recursos virtuais de forma rápida, sem restrição e a qualquer momento, dando ao usuário a sensação de recursos infinitos (MELL; GRANCE, 2009).

Embora elasticidade e escalabilidade sejam sinônimos, seus conceitos são bem diferentes, (ISLAM et al., 2012) apresenta a escalabilidade como sendo a capacidade de um sistema para adicionar mais recursos para atender a uma demanda maior de carga de trabalho. Além disso, a escalabilidade é livre da noção de tempo, e não captura o tempo que o sistema leva para atingir o nível de desempenho desejado. A elasticidade consiste no crescimento e redução de recursos de acordo com a carga de trabalho, enquanto a escalabilidade considera apenas o crescimento. Nesse caso, o tempo é um elemento fundamental para a elasticidade, que depende da velocidade de resposta para lidar com a variação da carga de trabalho (RISTA, 2018).

Com base na Figura 3, a qual apresenta uma comparação entre escalabilidade e elasticidade, as áreas em preto representam a falta de recursos, enquanto que as áreas em cinza representam os recursos alocados ociosos, ficando evidente que a escalabilidade se mostra menos vantajosa em relação a elasticidade, devido a uma grande ociosidade de recursos, enquanto a elasticidade se readequa a necessidade.

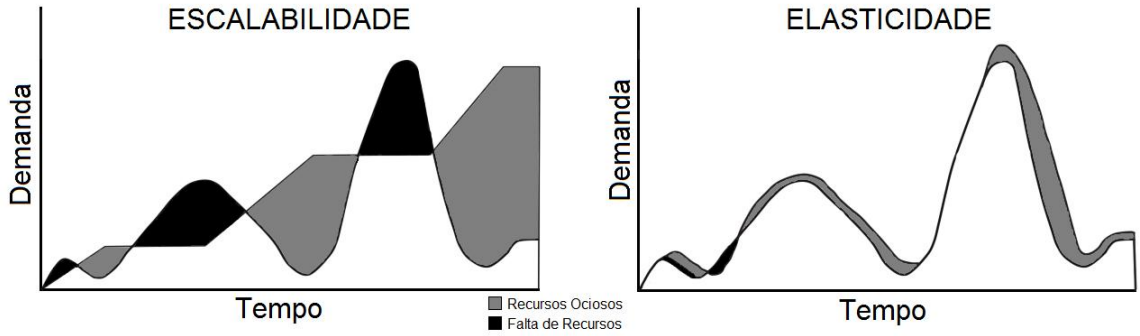

Figura 3: Escalabilidade X Elasticidade

Fonte: (NASSER; BREITMAN, 2012)

Para (COUTINHO et al., 2014), existem diferentes soluções que implementam elasticidade, tais soluções adicionam e removem recursos de acordo com políticas baseadas em carga de trabalho, por exemplo, quando o uso da CPU é maior que um valor definido ou quando o Acordo de Nível de Serviço (ANS) é violado.

Uma das formas de utilizar a elasticidade no MPTCP, é utilizando elasticidade horizontal, adicionando ou removendo instâncias do ambiente virtual conforme a necessidade da demanda, havendo um aumento do rendimento, deixando mais dinâmico,

entre outros benefícios. Entretanto esse trabalho tem por objetivo, propor uma melhoria na rede através do MPTCP e da escalabilidade.

Porém para se projetar um modelo, é necessário algumas informações de comportamento do sistema, seu funcionamento e de que forma pode ser usado. Então, antes de tudo, é fundamental o conhecimento da tecnologia que será utilizada nesse trabalho, o MPTCP.

2.3 MULTIPATCH TCP (MPTCP)

Segundo (OLTEANU; RAICIU, 2016) o MPTCP é uma extensão recente do TCP (*Transmission Control Protocol* / Protocolo de Controle de Transmissão) que permite que os terminais utilizem múltiplos caminhos através da rede na mesma conexão de transporte e é um substituto do TCP. A escolha do MPTCP está ganhando espaço entre grandes empresas como a Apple IOS (e OSX) implementam MPTCP, Samsung e LG oferecem versões para Android com implementações MPTCP e várias operadoras coreanas usam MPTCP para unir a conexão LTE (*Long Term Evolution*) e a 802.11ac para atingir transmissões a gigabit em telefones celulares.

O MPTCP permite o uso simultâneo das interfaces de rede disponíveis por meio da criação de múltiplos sub-fluxos. Com o MPTCP a aplicação continua acessando a rede como se fosse uma única interface de rede, porém, na verdade os dados são distribuídos pelos sub-fluxos existentes, conforme ilustrado na Figura 4. Os benefícios disso incluem melhor utilização de recursos, melhor rendimento e reação mais suave a falhas.

Para (BONAVENTURE et al., 2012) o projeto do MPTCP foi influenciado por muitos requisitos, mas há dois que se destacam: compatibilidade de aplicativos e compatibilidade de rede. A compatibilidade de aplicativos implica que os aplicativos que hoje são executados sobre TCP devem funcionar sem qualquer alteração em relação ao MPTCP. A compatibilidade de rede refere-se que o uso do MPTCP deve operar em qualquer caminho da Internet em que o TCP opera.

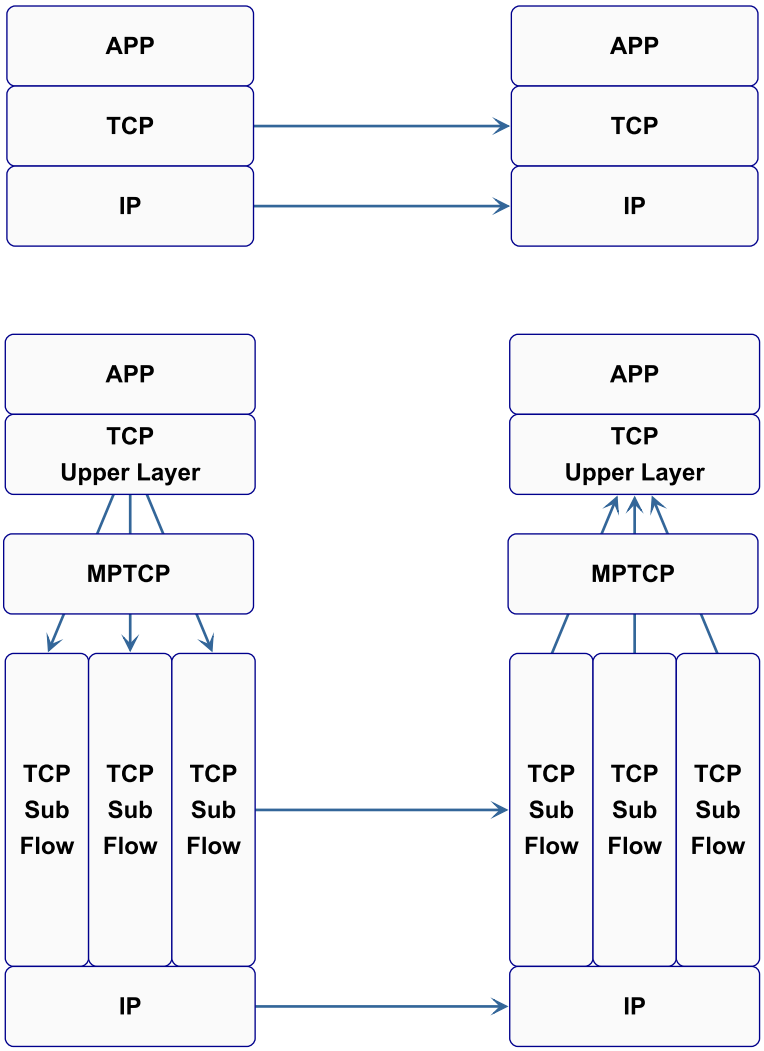

Figura 4: Funcionamento TCP e MPTCP

Fonte: (HUSTON, 2015)

2.3.1 TCP (TRANSMISSION CONTROL PROTOCOL)

Para uma melhor compreensão do MPTCP, (BONAVENTURE et al., 2012) propõe que entenda-se primeiro o funcionamento básico do TCP, podendo ele ser dividido em três partes:

- Estabelecimento de conexão;
- Transferência de dados;
- Liberação de conexão.

Uma conexão TCP começa com um *handshake* de três vias, como pode ser visto na Figura 5. Para abrir uma conexão TCP, o cliente envia um pacote SYN (*Synchronize* / Sincronizar) para a porta na qual o servidor está escutando (item 1 da Figura 5). O pacote SYN contém a porta de origem, o número de sequência inicial escolhido pelo cliente da transação e pode conter opções de TCP usadas para negociar o uso de extensões TCP. O servidor responde com um pacote SYN + ACK (*Acknowledgement* / Confirmação), confirmando o SYN e fornecendo o número de sequência inicial do servidor e as opções que ele suporta (item 2 da Figura 5). O cliente reconhece o SYN + ACK e a conexão agora está totalmente estabelecida (item 3 da Figura 5). Todos os pacotes subsequentes na conexão usam os endereços IP e as portas usadas para o *handshake* inicial. Eles compõem a tupla que identifica exclusivamente a conexão.

Após o *handshake*, o cliente e o servidor podem enviar pacotes de dados (segmentos, na terminologia TCP). O número sequencial é usado para delinear os dados nos diferentes segmentos, reordená-los e detectar perdas. O cabeçalho TCP também contém uma confirmação cumulativa, essencialmente um número que reconhece os dados recebidos, informando ao remetente qual é o próximo byte esperado pelo receptor. Várias técnicas são usadas pelo TCP para retransmitir os segmentos perdidos. Após a transferência de dados, a conexão TCP deve ser fechada. Uma conexão TCP pode ser fechada abruptamente se um dos hosts envia um pacote de redefinição, mas a maneira usual de encerrar uma conexão é usando pacotes FIN. Estes pacotes FIN indicam o número sequencial do último byte enviado. A conexão é finalizada após os segmentos FIN terem sido reconhecidos em ambas as direções.

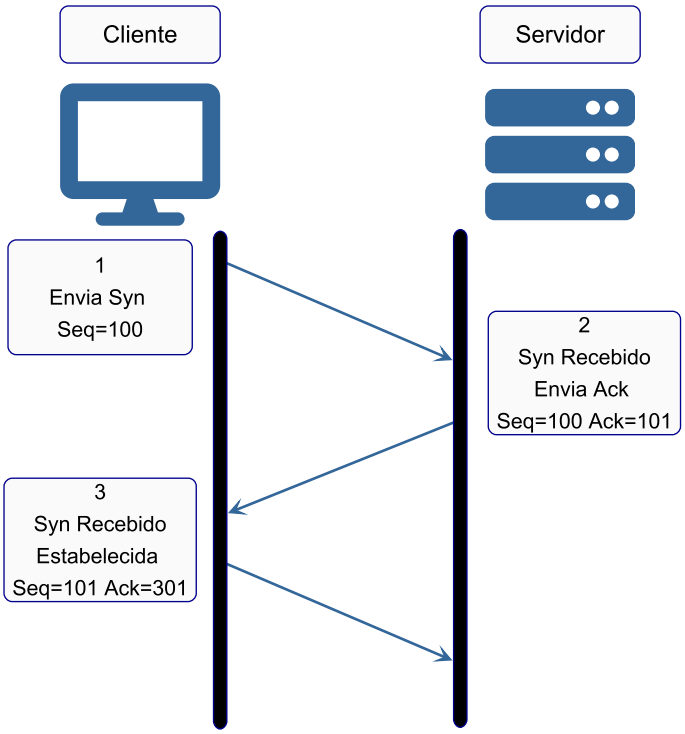

Figura 5: Conexão TCP

Fonte: Adaptado de (CELESTINO, 2017)

Diferentemente, o MPTCP permite que vários subfluxos sejam configurados para uma única sessão MPTCP. Uma sessão MPTCP começa com um subfluxo inicial, que é semelhante a uma conexão TCP regular. Após o primeiro subfluxo MPTCP ser configurado, subfluxos adicionais podem ser estabelecidos. Cada subfluxo adicional também é semelhante a uma conexão TCP regular, completa com *handshake* SYN e desmontagem FIN, mas em vez de ser uma conexão separada, o subfluxo é ligado a uma sessão MPTCP existente. Os dados para a conexão podem, então, ser enviados através de qualquer um dos subfluxos ativos com capacidade para recebê-lo (BONAVENTURE et al., 2012).

2.3.2 EXEMPLO DO USO DO MPTCP

Considere um cenário simples, apresentado por (BONAVENTURE et al., 2012), apresentado na Figura 6, em que há um cliente de smartphone e um servidor de hospedagem. O smartphone possui duas interfaces de rede: uma interface WiFi e

uma interface 3G/LTE. Cada interface tem seu próprio endereço IP. O servidor, tem um único endereço IP. Nesse ambiente, o MPTCP permitiria que um aplicativo no smartphone usasse uma única conexão TCP que pudesse usar as interfaces WiFi e 3G para se comunicar com o servidor. O aplicativo não precisa levar em consideração qual interface está funcionando melhor em qualquer instante, pois o MPTCP trata disso para o aplicativo. Suponha que o smartphone escolha sua interface 3G para abrir a conexão. Primeiro, o smartphone envia um segmento SYN ao servidor. Este segmento contém a opção TCP *MP_CAPABLE*, indicando que o smartphone suporta MPTCP. Esta opção também contém uma chave escolhida pelo smartphone. O servidor responde com um segmento SYN + ACK contendo a opção *MP_CAPABLE* (item 01 na Figura 6) e a chave escolhida pelo servidor. O smartphone conclui o *handshake* enviando um segmento ACK.

Nesse ponto, a conexão MPTCP é estabelecida, o cliente e o servidor podem trocar segmentos TCP por meio do caminho 3G. Porém também é possível enviar dados pela interface WiFi simultaneamente com o 3G. Para que isso ocorra é necessário executar um *handshake* SYN completo no caminho WiFi antes de enviar qualquer pacote dessa maneira. Então esse *handshake* SYN carrega a opção *MP_JOIN* TCP, fornecendo informações suficientes ao servidor para poder identificar com segurança a conexão correta com a qual associar esse subfluxo adicional. O servidor responde com *MP_JOIN* (item 02 na Figura 6) no SYN + ACK e o novo subfluxo é estabelecido.

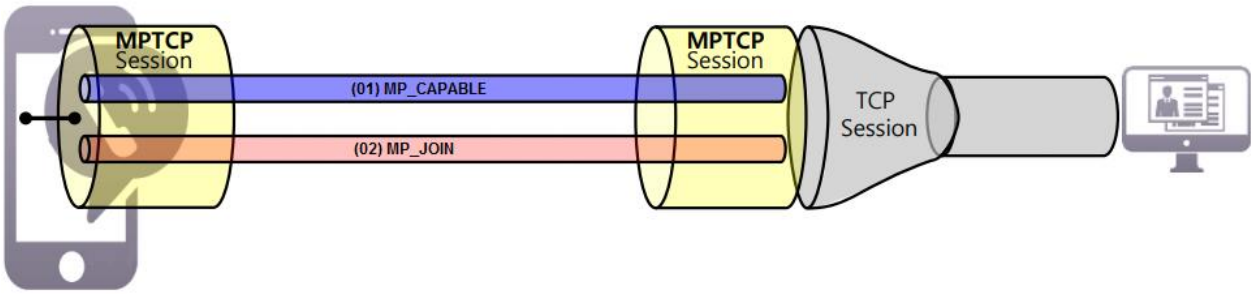

Figura 6: Opções do MPTCP

Fonte: (SEO, 2015)

Um ponto importante sobre o MPTCP, é que o conjunto de subfluxos associados a uma conexão TCP do Multipath não é fixo. Os subfluxos podem ser adicionados e removidos dinamicamente de uma conexão MPTCP durante todo o seu tempo de vida, sem afetar o fluxo de *bytes* transportados em nome do aplicativo.

Contudo, para ser possível comparar os resultados encontrados por (RISTA et al., 2017), é preciso analisar o IEEE 802.3ad, entender seu funcionamento e quais as diferenças quando em comparação com o MPTCP. Para então ser possível reproduzir os experimentos em uso do IEEE 802.3ad e fazer novos experimentos com uso do MPTCP, podendo assim chegar a uma conclusão das vantagens de cada um.

2.4 AGREGAÇÃO DE LINKS

Segundo (CHRIST et al., 2018), a agregação de *links* implementa vários métodos para combinar múltiplas conexões de rede. Pode ser usado tanto com a finalidade de aumentar o rendimento, quanto para fornecer redundância em caso de falha em algum *link* ou ambos. Um exemplo é o método aplicado pelo protocolo *bonding*, que busca a combinação de uma série de portas físicas para criar um único *link* lógico de dados. Dependendo da quantidade e capacidade dos *links* físicos, este protocolo consegue oferecer um *link* de alta largura de banda, compartilhando a carga de tráfego entre as portas delegadas.

Como pode ser visto na Figura 7, os dois canais físicos (eth2 e eth3) são combinados criando assim um único canal lógico de dados.

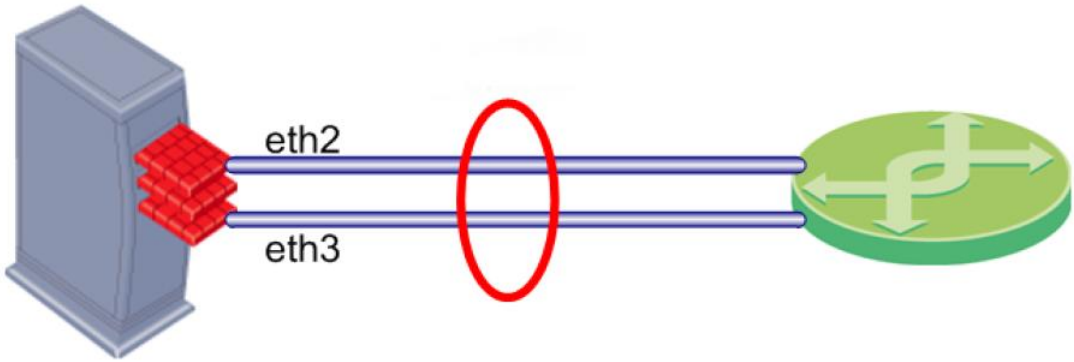

Figura 7: Agregação de *links*

Fonte: (HINTERSTEINER, 2016)

Conforme (DAVIS et al., 2000), essas ligações podem ser configuradas segundo os modos da Tabela 2.

Neste trabalho iremos focar apenas no modo de ligação Balance-RR utilizado nos experimentos apresentados adiante.

Algoritmo *Bonding*	**Modo**	**Descrição**
Balance-RR	0	Transmite pacotes em ordem sequencial. Provê balanceamento de carga e tolerância a falhas.
Active-Backup	1	Apenas uma interface da ligação está ativa. Uma interface diferente se torna ativa se, e somente se, a interface ativa falhar. Provê tolerância a falhas.
Balance-Xor	2	Associa o endereço MAC de destino do pacote com o endereço MAC de uma das interfaces escravas. Provê tolerância a falhas e balanceamento de carga XOR.
Broadcast	3	Todos os pacotes serão enviados em todas as interfaces escravas. Provê tolerância a falhas.
802.3ad (LACP)	4	IEEE 802.3ad Agregação de link dinâmico (*Link Aggregation Control Protocol*).
Balance-tlb	5	Balanceamento de carga de transmissão adaptável: associação de canal que não requer nenhum suporte de switch especial.
Balance-alb	6	Balanceamento de carga adaptativo incluindo Balance-tlb plus, balanceamento de carga de recebimento e não requer nenhum suporte de switch especial.

Tabela 2: Modos de Ligação

2.4.1 IEEE 802.3AD (LACP)

A definição de LACP (*Link Aggregation Protocol*) consiste em um algoritmo constituído por máquinas de estados e funções, que torna possível através de trocas de pacotes, combinar diversos enlaces físicos em um único enlace lógico. Vários benefícios compõem esta técnica, tais como aumento na largura de banda, redundância e balanceamento de carga nos enlaces físicos (BONDAN et al., 2012). O LACP faz parte da especificação IEEE 802.3ad que permite o agrupamento de várias portas físicas para formar um único canal lógico (SARABANDO, 2008).

O LACP pode ser inserido em dois tipos de equipamentos de redes distintos: *switchs* ou servidores. Seu funcionamento correto depende da garantia de que o protocolo esteja disponível em ambos os lados da conexão. O LACP também aceita diferentes topologias. Pode ser configurado, por exemplo, entre dois *switchs* Ethernet, ou entre um *switch* Ethernet e um servidor (BONDAN et al., 2012).

Entretanto por uma falta dos equipamentos necessários para reprodução dos testes realizados por (RISTA et al., 2017), foi-se optado pelo uso do modo Balance-RR, pois desse modo não se faz necessário o uso de *switchs*, mas apenas a ligação de um cliente ao outro é suficiente.

2.4.2 BALANCE-RR

O algoritmo Balance-RR(Round-Robin) usa uma política de rodízio, ou seja, transmite pacotes em ordem sequencial do primeiro escravo disponível até o último. Este modo oferece balanceamento de carga e tolerância a falhas (DAVIS et al., 2000).

Contudo, o objetivo é analisar o desempenho da rede em um ambiente de nuvem de alto desempenho, para isso, usamos instâncias baseadas em *container*, e aplicativos como Hadoop para uma análise de dados em larga escala. O uso de *containers* permite que sistemas isolados de Linux possam ser executados em um único *host*. Foi escolhido então a tecnologia LXC (*Linux Container*), pois se trata de um software gratuito que oferece ferramentas e utilitários para gerenciar *containers*.

2.5 LINUX CONTAINERS (LXC)

Segundo (MALISZEWSKI et al., 2018), o LXC é uma virtualização em nível de sistema operacional, que abstrai recursos computacionais por meio de grupos de controle (*cgroups*) para controlar recursos do sistema por meio de *namespaces*, criar e isolar os objetos dentro do *container* chamado. Esses *containers* compartilham o *kernel* com o sistema operacional *host* e, portanto, seus processos e sistema de arquivos são acessíveis a partir do *host*. Portanto, ele é definido como uma virtualização leve, que não requer emulação de hardware físico. O principal objetivo de usar o LXC é fornecer recursos ou o mesmo ambiente, como uma Máquina Virtual (*Virtual Machine* - VM). Finalmente, através de uma combinação de recursos de segurança do *kernel*, é possível criar um ambiente virtual na mesma máquina sem um *hypervisor*.

Os *containers* podem dividir recursos gerenciados por um sistema operacional em grupos isolados para equilibrar as demandas no uso de recursos. Além disso, o LXC pode executar instruções nativas no núcleo da CPU sem qualquer mecanismo de interpretação especial. Assim, através do uso de *containers*, o sistema operacional dá aos aplicativos a ilusão de serem executados em máquinas separadas enquanto compartilham os recursos subjacentes. Um exemplo pode ser visto na Figura 8, que mostra uma comparação entre o LXC (virtualização em nível de sistema operacional) e o KVM (*Kernel-based Virtual Machine*) (virtualização completa). Ambas as tecnologias usam a API do *Libvirt*. Como pode ser visto, o LXC requer menos abstração de software, usando o mesmo *kernel* do sistema operacional nativo e usando *Linux Bridges* ou interfaces nativas para conectividade de rede, enquanto o KVM usa *drivers* paravirtualizados

para operações de E/S e conectividade de rede oferecida a VMs (MALISZEWSKI et al., 2018).

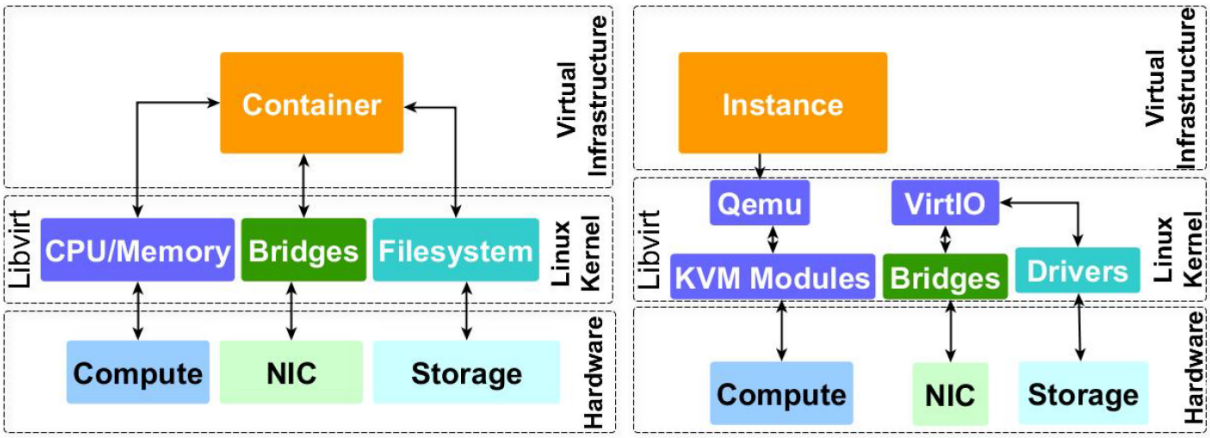

Figura 8: Comparação entre LXC e KVM

Fonte: (MALISZEWSKI et al., 2018)

Contudo, um dos objetivos deste trabalho é a comparação entre MPTCP e o Balance-RR para ambientes de computação em nuvem. Como um cenário de nuvem se trata de grandes quantidades de dados, apenas alguns *containers* não conseguem obter um bom desempenho, para isso foi feito a escolha do uso do Hadoop. Com o Hadoop é possível criar *clusters* de *containers* o que permite aproveitar o gerenciamento de recursos, simulando assim um ambiente mais real como utilizado por grandes empresas.

2.6 APACHE HADOOP

O projeto Apache Hadoop desenvolve um software de código aberto para computação distribuída confiável e escalável. A biblioteca de software Apache Hadoop é uma estrutura que permite o processamento distribuído de grandes conjuntos de dados em *clusters* de computadores usando modelos de programação simples. Ele é projetado para escalar de servidores únicos para milhares de máquinas, cada uma oferecendo computação e armazenamento locais. Levando em consideração prováveis falhas de *hardware* para fornecer a disponibilidade necessária, a biblioteca é projetada para detectar e lidar com falhas na camada do aplicativo, entregando um serviço altamente disponível em um *cluster* de computadores (FOUNDATION, 2018).

A estrutura do Hadoop permite o processamento distribuído em *clusters* de computador. Ele foi projetado para funcionar com um único servidor ou conjunto de mil máquinas, oferecendo processamento e armazenamento local. A estrutura em si é projetada para detectar e resolver falhas na camada de aplicativo para fornecer serviços de alta disponibilidade (RISTA et al., 2017).

2.6.1 HADOOP DISTRIBUTED FILE SYSTEM

O HDFS (*Hadoop Distributed File System*) foi projetado para armazenar conjuntos de dados muito grandes de forma confiável e para transmitir esses conjuntos de dados em alta largura de banda para os aplicativos do usuário. Em um grande cluster, milhares de servidores hospedam o armazenamento diretamente conectado e executam tarefas de aplicativos do usuário. Ao distribuir armazenamento e computação em muitos servidores, o recurso pode crescer com a demanda, ao mesmo tempo em que permanece econômico em todos os tamanhos (CHANSLER et al., 2010).

O HDFS armazena metadados do sistema de arquivos e dados do aplicativo separadamente. O HDFS armazena metadados em um servidor dedicado, chamado *NameNode*. Os dados do aplicativo são armazenados em outros servidores chamados *DataNodes*. Todos os servidores estão totalmente conectados e se comunicam entre si usando protocolos baseados em TCP. Os *DataNodes* no HDFS não contam com mecanismos de proteção de dados, para tornar os dados duráveis, em vez disso, o conteúdo do arquivo é replicado em vários *DataNodes* para confiabilidade. Ao mesmo tempo em que garante a durabilidade dos dados, essa estratégia tem a vantagem adicional de que a largura de banda de transferência de dados é multiplicada e há mais oportunidades para localizar computação perto dos dados necessários (CHANSLER et al., 2010).

2.6.2 MAPREDUCE

O MapReduce é um modelo de programação e uma implementação associada para processar e gerar grandes conjuntos de dados. Os usuários especificam uma função map que processa um par chave/valor para gerar um conjunto de pares chave/-valor intermediários e uma função reduce que mescla todos os valores intermediários associados à mesma chave intermediária (DEAN; GHEMAWAT, 2004).

Os programas escritos neste estilo funcional são paralelizados automaticamente e executados em um grande *cluster* de máquinas comuns. O sistema de tempo

de execução cuida dos detalhes de particionamento dos dados de entrada, agendamento da execução do programa em um conjunto de máquinas, tratamento de falhas de máquina e gerenciamento da comunicação necessária entre máquinas. Isso permite que os programadores sem qualquer experiência com sistemas paralelos e distribuídos utilizem facilmente os recursos de um grande sistema distribuído (DEAN; GHEMAWAT, 2004).

2.6.3 SHUFFLE

Segundo (RISTA et al., 2017) o conceito básico de MapReduce é processar *big data* em paralelo, realizando operações Map e Reduce em estágios. No entanto, existe um estágio intermediário chamado Shuffle. Ele coleta dados mapeados da operação Map para mesclar e entregar cálculos para a operação Reduce. Como há muitas comunicações que ocorrem entre muitos processos e nós no estágio Shuffle, uma grande quantidade de dados está sendo transferida pela rede. A Figura 9 representa como a comunicação ocorre de maneira de alto nível ao usar MapReduce no Hadoop.

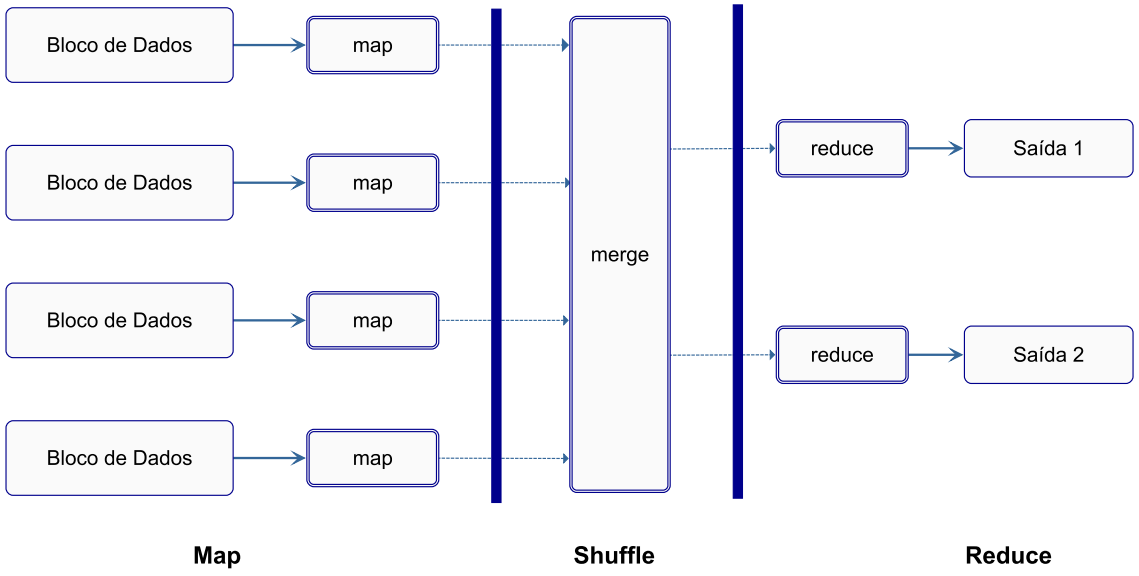

Figura 9: Representação de alto nível dos estágios do MapReduce

Fonte: (RISTA et al., 2017)

Conforme (DAVIDSON; OR, 2013), em estruturas MapReduce tradicionais, a fase de Shuffle geralmente é ofuscada pelas fases Map e Reduce. Na verdade, o Shuffle

é comumente integrado como parte da fase Reduce, embora realmente tenha pouco a ver com a semântica dos dados. Todo o conjunto de trabalho, que geralmente é uma grande fração dos dados de entrada, deve ser transferido pela rede. Isso sobrecarrega significativamente o sistema operacional, tanto na origem quanto no destino, exigindo muitos arquivos e E/S de rede.

3 TRABALHOS RELACIONADOS

Esta seção descreve os principais trabalhos relacionados com Redes de Petri e o objetivo de melhorar a rede em aplicativos de *big data*.

O trabalho de (YAZDANOV et al., 2015) propõem um escalonador de E/S sensível à rede chamado EHadoop para um cluster MapReduce elástico. A análise apresentou que, durante os picos de carga, mais nós de computação são adicionados para manter o desempenho do cluster. No entanto, a taxa de transferência não melhora devido à saturação da rede. Assim, o EHadoop realiza o agendamento de tarefas com base na largura de banda disponível. Os resultados da avaliação mostram que o EHadoop evita a contenção da rede, mas não aumenta o tempo de conclusão das tarefas MapReduce e até mesmo causa a degradação da largura de banda da rede.

A abordagem apresentada por (CHRIST et al., 2018), mostra o método da otimização da largura de banda através da implementação da agregação de links. O objetivo deste trabalho é avaliar o desempenho de rede usando a agregação de link com o protocolo bonding em máquinas virtuais LXC e KVM. Os resultados apresentaram que o protocolo bonding tem comportamento similar com ambos os tipos de virtualização.

Um estudo comparativo entre aplicativos com diferentes necessidades e complexidades de comunicação foi realizado por (EKANAYAKE; FOX, 2010). Este trabalho concluiu que os aplicativos sensíveis à latência tem maior degradação do desempenho do que os aplicativos sensíveis à largura de banda. Ademais, este estudo também levou em consideração diferentes implementações de MapReduce, apresentando uma análise de desempenho de aplicativos paralelos de alto desempenho em ambientes virtualizados.

O trabalho de (HE et al., 2016), apresenta uma Rede Dinâmica Estocástica de Petri (DSSPN) para modelar e analisar o desempenho de sistemas de computação em nuvem. O DSSPN pode não apenas representar claramente os comportamentos

dinâmicos do sistema de uma maneira intuitiva e eficiente, mas também descobrir facilmente as deficiências de desempenho e os gargalos dos sistemas. Para validar o algoritmo aprimorado e a aplicabilidade do DSSPN, foi realizado experimentos extensivos por meio do Stochastic Petri Net Package (SPNP). Os resultados de desempenho mostraram que o algoritmo melhorado é melhor do que o agendamento justo em alguns indicadores-chave de desempenho, como rendimento médio, tempo de resposta e tempo médio de conclusão.

O modelo proposto por (PENGSONG, 2015), de avaliação de sistema em nuvem baseado em rede de Petri de filas (QPNC), melhorou a análise quantitativa e o sistema de avaliação do sistema em nuvem e simulou um sistema em nuvem com serviço dinâmico em um ambiente maciçamente paralelo. Os resultados experimentais mostraram que o QPNC é capaz de refletir efetivamente as características da arquitetura de vários sistemas em nuvem nas perspectivas de desempenho, serviço, etc., e simular vários tipos de comportamentos de serviço dinâmico de sistemas em nuvem.

É possível notar que a literatura não apresenta nenhum trabalho focado em demonstrar as vantagens do MPTCP para latência, taxa de transferência e tempo de execução em aplicativos de *big data*, nem mesmo um modelo capaz de avaliar o uso das interfaces de rede com uso do MPTCP como nosso trabalho faz. Em contraste com trabalhos anteriores, nossa pesquisa apresenta um modelo de implantação com reconhecimento de desempenho de rede de baixo nível para aplicativos Hadoop.

4 ABORDAGEM DE APRIMORAMENTO DA REDE

No decorrer deste capítulo são apresentados o modelo GSPN e o ambiente de testes montado neste trabalho.

4.1 MODELO GSPN

No artigo (RISTA et al., 2018), é apresentado dois modelos GSPN, o primeiro trata-se do modelo sem agregação de links, enquanto que o segundo modelo apresenta a utilização da agregação de links, sendo essa uma extensão do primeiro modelo. Com base nesses modelos foi introduzido um novo modelo, com finalidade de estimar a utilização das interfaces através do MPTCP. Com base nos resultados obtidos pelo modelo, é possível realizar melhorias e realizar um melhor planejamento em casos realistas.

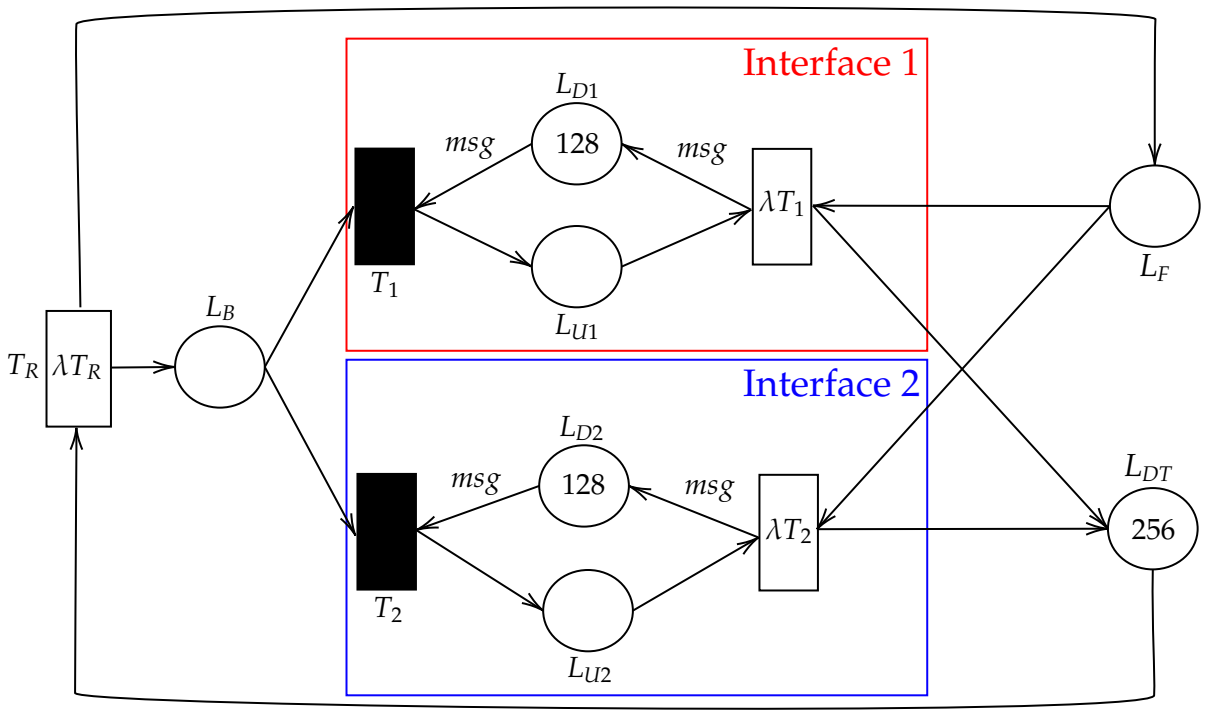

Figura 10: Modelo GSPN com uso do MPTCP

Como é apresentado na Figura 10, o modelo começa quando a transição T_R aciona tokens para o lugar L_B com um delay λT_R da transição T_R, em que os tokens representam solicitações de redes que chegam para o processamento das interfaces. O lugar L_B é encarregado de receber as solicitações e encaminhá-las para as interfaces através das transações T_1 e T_2 de maneira arbitrária e com a mesma probabilidade.

O lugar L_{Di}, sendo i = 1, 2, representa a disponibilidade da interface, ou seja, a solicitação só será aceita pela interface caso msg (mensagem) $\leq L_{Di}$ atual, visto que msg equivale ao tamanho da mensagem, caso contrário a solicitação terá que esperar até a condição ser aceita. O lugar L_{Ui} equivale ao uso da interface, isto é, quanto mais a interface for usada para o processamento das mensagens mais tokens passará pelo lugar L_{Ui}.

A msg, representa o tamanho médio das mensagens que são transferidas em um sistema real. Como em um sistema real o tamanho das mensagens não são sempre fixos, optamos por escolher uma média desses valores para utilizar no modelo.

As transições T_1 e T_2 não são temporizadas, ou seja, são imediatas. Conforme os tokens chegam no lugar L_B, elas são capturadas instantaneamente para as transições T_1 e T_2 de forma arbitrária, sendo que cada transição tem a mesma probabilidade de capturar os tokens.

Após o processamento da mensagem, o token sai do modelo, e a disponibilidade da interface é restituída com um delay de λT_i, sendo esse o tempo de transmissão da mensagem, podendo assim aceitar novas solicitações. O cálculo de λT_i é apresentado na equação (10).

$$\lambda T_i = \frac{\lambda T_R}{\#L_{Di} * msg * 0.00015} \tag{10}$$

Conforme a fórmula acima, a variável λT_R é a taxa de chegada das solicitações, a variável $\#L_{Di}$ representa a capacidade máxima da interface, ou seja, nesse caso de 128 MB (Mega Bytes), a variável msg expressa a média do tamanho das mensagens sendo o parâmetro variável. Em relação a constante 0.00015, foi encontrada de forma empírica, através de vários testes.

O lugar L_F apresenta todas as solicitações que estão em espera ou processamento. O lugar L_{DT} expressa a soma das disponibilidades das interfaces, desse modo o modelo opera apenas com o máximo de solicitações suportadas pelas interfaces, que

nesse caso seria de 256 MB, ou seja, a quantidade de tokens para o lugar L_{DT} é expressa a partir da expressão (12).

$$\#L_{DT} = \#L_{D1} + \#L_{D2} \tag{12}$$

4.1.1 CONFIGURAÇÕES DO MODELO

Para a simulação do modelo, é necessário um conjunto de parâmetros, que são apresentados na Tabela 3.

Tabela 3: Parâmetros do modelo

λT_R	$\#L_{Di}$	$\#L_{DT}$	**msg**
600 ms	128 MB	256 MB	5 a 8 MB

Conforme a Tabela 3, o *delay* λT_R da transição T_R representa a taxa de chegada da solicitação desejada, pode ser alterada para simular diferentes cargas de trabalho. Para esse experimento foi atribuído um *delay* fixo de 600 ms para o λT_R, próximo de duas solicitações por segundo, como o modelo possui dois módulos, a taxa de cada módulo é de aproximadamente uma solicitação por segundo, equivalendo ao *Benchmark* utilizado para o experimento.

Para o L_{Di} foi determinado o uso de 128 MB que corresponde à largura de banda de 1Gb (Giga Bits) das placas de rede. Para as mensagens (msg), foi definido um intervalo entre 5 MB e 8 MB, sendo o parâmetro variável do modelo, a escolha desses valores se dá ao fato de ser valores próximos aos usados pelo *Benchmark*. Com os parâmetros em mãos é possível atribuir o *delay* λT_i, encontrando assim a latência da rede para a transmissão das mensagens.

Para o inicio da simulação é necessário algumas marcações iniciais, os lugares L_{Di} recebem a marcação inicial de 128, representando a largura de banda das placas de rede, enquanto que o lugar L_{DT} recebe a marcação inicial de 256, representando a soma das interfaces ou seja a soma dos lugares L_{Di}.

4.2 AMBIENTE DE TESTES

No ambiente do experimento, foram usadas duas máquinas com configurações iguais como apresentado na Tabela 4. Para comunicação entre as interfaces, foi utilizado

os cabos de rede CAT6, aonde cada interface esta ligada diretamente com a outra interface na segunda máquina. O sistema operacional adotado nas máquinas foi o Linux Debian 10 e Kernel 4.19.

Tabela 4: Hardware das Máquinas

Hosts	Processador	Placas de Rede	Memória RAM
Host 1	AMD Phenom(tm) II X2 B53 Processor, 2.8GHz	3 Interfaces Gigabit	8GB
Host 2	AMD Phenom(tm) II X2 B53 Processor, 2.8GHz	3 Interfaces Gigabit	8GB

4.2.1 MPTCP

Para o ambiente de testes, utilizou-se duas máquinas, dispondo de três interfaces de rede cada. Sendo que uma das interfaces está configurada para acesso à Internet, e as outras duas para a comunicação entre as máquinas. A interface de acesso à Internet é desativada para os experimentos.

4.2.1.1 CONFIGURAÇÃO MPTCP

Primeiramente foi instalado o MPTCP na versão 0.95, conforme (MPTCP, 2007), e após isso reinicia-se a máquina. Configura-se então as interfaces em cada host, conforme as informações descritas nas Tabelas 5 e 6. A configuração das interfaces dos *host* 1 e *host* 2 esta descrito nos Apêndices A.1 e A.2 respectivamente.

Tabela 5: Interfaces *Host* 1

Interface Escrava	Bridge	IP	Netmask	Gateway
enp47s0	breth0	10.1.1.2	255.255.255.0	10.1.1.1
enp31s0	breth1	10.1.2.2	255.255.255.0	10.1.2.1
enp63s0	-	DHCP	DHCP	DHCP

Tabela 6: Interfaces *Host* 2

Interface Escrava	Bridge	IP	Netmask	Gateway
enp47s0	breth0	10.1.1.4	255.255.255.0	10.1.1.1
enp31s0	breth1	10.1.2.4	255.255.255.0	10.1.2.1
enp63s0	-	DHCP	DHCP	DHCP

Como pode ser observado, atribuiu-se IPs as interfaces do *host* 1, a interface enp47s0 é definida como escrava pela *bridge* breth0 com IP 10.1.1.2, e a interface enp31s0

é escrava pela *bridge* breth1 com IP 10.1.2.2. Já a interface enp63s0 é configurada para acesso à Internet. Mais adiante será explicado o porque do uso das *bridges*.

Assim como no *host* 1, o modo de configuração do *host* 2 é o mesmo, a interface enp47s0 é definida como escrava pela *bridge* breth0 com IP 10.1.1.4, e a interface enp31s0 é escrava pela *bridge* breth1 com IP 10.1.2.4. Já a interface enp63s0 é configurada para acesso à Internet.

Segundo (PAASCH, 2007), com múltiplos endereços definidos em várias interfaces, pretende-se dizer ao kernel "Se selecionar tal endereço de origem, use aquela interface específica + *gateway*, não os padrões" . Consegue-se isso configurando uma tabela de roteamento por interface de saída, cada tabela de roteamento sendo identificada por um número. O processo de seleção de rotas acontece em duas fases. Primeiro, o kernel faz uma pesquisa na tabela de políticas (que precisa configurar com as regras de IP). As políticas, para esse caso, são "Para esse prefixo de origem , vá para o número da tabela de roteamento x". Em seguida, a tabela de roteamento correspondente é examinada para selecionar o *gateway* com base no endereço de destino. Assim, é necessário configurar as regras de roteamento do *host* 1 para que os pacotes com IP de origem 10.1.1.2 sejam roteados sobre breth0 e aqueles com 10.1.2.2 sejam roteados sobre breth1. A configuração necesária para o funcionamento adequado do roteamento é encontrada em scripts em (MPTCP, 2007).

Depois de configurado, a interface enp63s0, usada para acesso à Internet é desativada, e as outras duas interfaces que sobraram de cada *host*, ficam para fazer os experimentos com MPTCP. Após realizado todos os testes necessários e registrado todos os resultados obtidos, presseguiu-se para os experimentos relacionados ao *Balance-RR*, sendo necessário alterar as configurações de cada interface e as regras de roteamento.

4.2.2 BALANCE-RR

A intenção inicial era ser feito a reprodução dos experimentos realizados por (RISTA et al., 2017), entretanto por falta dos equipamentos necessários para a reprodução, não foi possível, pois para a realização dos experimentos com LACP é preciso de ao menos um *switch* com suporte para o LACP com portas Gigabits. Optamos então por realizar os mesmos experimentos, mas ao invés de usarmos LACP, usamos o Balance-RR, sem a necessidade de um *switch*, ligando um *host* ao outro diretamente. Foi usado os mesmos *hosts* da Tabela 3, e da mesma maneira que os experimentos com MPTCP, duas interfaces são usadas para a comunicação entre um *host* e outro, e a

terceira interface usada para acesso à Internet é desativada.

4.2.2.1 CONFIGURAÇÃO BALANCE-RR

Diferentemente da configuração do MPTCP, no Balance-RR é usado a técnica *Ethernet bonding*, na qual duas interfaces são configuradas como escravas de um *bond*, no modo agregação de *links*. Uma *bridge* é então criada para acesso a *bond*.

Primeiramente, é necessário configurar cada interface com as informações descritas nas Tabelas 7 e 8. A configuração das interfaces do *host* 1 e *host* 2 está descrito nos Apêndices B.1 e B.2 respectivamente.

Tabela 7: Interfaces *Host* 1

Interface Escrava	**Bridge**	**IP**	**Netmask**	**Gateway**
enp47s0/enp31s0	br0	10.3.3.2	255.255.255.0	10.3.3.1

Tabela 8: Interfaces *Host* 2

Interface Escrava	**Bridge**	**IP**	**Netmask**	**Gateway**
enp47s0/enp31s0	br0	10.3.3.3	255.255.255.0	10.3.3.1

Dessa forma, as interfaces enp47s0 e enp31s0 são configuradas como escravas de bond0, no modo agregação de *links*. A *bridge* br0, tem acesso então a bond0, com IP 10.3.3.2. Do mesmo modo que no *host* 1, a configuração do *host* 2 é igual, alterando somente o IP para 10.3.3.3.

Após os experimentos realizados e registrado os resultados obtidos, configura-se a parte do LXC, aonde cada máquina virtual possui duas interfaces de rede.

4.2.3 LXC

A instalação e configuração do LXC é feita conforme os passos fornecidos por (LXC, 2016), para a instalação propriamente dita é usado o comando da Listagem 4.1. Para os experimentos foi usado a versÃ£o LXC 4.0.

Listing 4.1: Instalação do LXC

```
$ sudo apt-get install lxc
$ sudo apt-get install bridge-utils
$ sudo apt-get install vlan
$ sudo apt-get instal ifenslave
```

Criou-se então um *container* em cada *host* com o sistema operacional Debian. Cada *container* possui quatro interfaces virtuais, duas para o uso do MPTCP, uma para acesso à Internet e uma para uso do Balance-RR. Quando realizado o experimento de MPTCP, desativa as outras duas, quando realizado o experimento de Balance-RR, desativa as outras três. Conforme a Tabela 9, cada interface tem acesso a uma *bridge* criada anteriormente como pode ser visto nos apêndices apresentados até o momento.

Aqui se faz necessário a justificação do porque do uso das *bridges*, pois a máquina virtual não consegue ter acesso direto a interface física. Cada interface virtual é ligada com uma interface física, ou seja, cada interface virtual é como se fosse uma interface física.

Contudo para que o *container* tenha acesso à Internet, é preciso criar uma *bridge* utilizando a interface física(enp63s0) usada para acesso à Internet. A configuração utilizada para isso pode ser encontrada no Apêndice C.2. Já a configuração das interfaces do *container* pode ser vista no Apêndice C.1.

Tabela 9: Interfaces *Container*

Interface Virtual	**Bridge**
eth0	breth0
eth1	breth1
ethnet	brnet
eth0bonding	br0

Foi chamado o *container* do *host* 1 de *container* 1, e de *container* 2 o *container* do *host* 2. Para realizar os experimentos do MPTCP com os *containers*, é preciso configurá-los, para isso é necessário acessar o *container* desejado, e configurar as interfaces como foi feito anteriormente em cada *host*. A configuração do *container* 1 e do *container* 2 são encontradas nos Apêndices C.3 e C.4 respectivamente.

Entretanto para fazer o uso do MPTCP, como nos *hosts*, é preciso das regras de roteamento, que da mesma forma cria-se duas tabelas de roteamento com base nos IPs de cada interface. Assim como anteriormente, os scripts para essa configuração são encontradas em (MPTCP, 2007).

Após realizados todas as configurações, é realizado o experimento e colhido os resultados da largura de banda, taxa de transferência e latência, para então partir para a próxima etapa, que é o experimento com o Balance-RR nos *containers*. Para isso se faz necessário algumas mudanças na configuração das interfaces, essas mudanças podem serem vistas nos Apêndice C.5 e C.6.

Para o acesso à Internet nos *containers*, também é preciso adicionar à configuração das interfaces algumas configurações referente a isso, esse adicional pode ser encontrado no Apêndice C.7.

5 RESULTADOS EXPERIMENTAIS

Neste capítulo são apresentados os resultados dos experimentos realizados com o modelo e com cada uma das configurações citadas no Capítulo 4.

5.1 MODELO GSPN

Para estimarmos a utilização das interfaces, variamos o tamanho das mensagens. Escolhemos a faixa de 5 MB a 8 MB, conforme estabelecido na Tabela 3. Para encontrar os pontos de saturação da rede, é preciso submeter o sistema a testes de estresse. Como a faixa do tamanho das mensagens foi escolhida com base na usada pelo *benchmark* Netpipe (NETPIPE, 1998), é possível comparar os resultados obtidos na prática com os resultados obtidos no modelo.

Os valores exibidos na Tabela 10, referem-se ao uso das interfaces com base no tamanho da mensagem. Os resultados obtidos através dos experimentos feitos na prática são apresentados como ”Medido”, enquanto que os resultados obtidos através do modelo são apresentados como ”Estimado”, nos dois casos foi realizada a variação no tamanho das mensagens. A coluna ”Média”, mostra a média do uso das interfaces durante todo o experimento.

Tabela 10: Resultados da Prática e do Modelo

msg	**5 MB**	**6 MB**	**7 MB**	**8 MB**	**Média**
Medido	83.5%	84.3%	84.75%	85.5%	84.5%
Estimado	82.5%	84%	85.5%	87%	84.75%

Em contra partida, os pontos de saturação encontrados no experimento prático poderiam ser previstos com a utilização do modelo GSPN apresentado na Figura 10. Dentre os benefícios, podemos citar o fato da viabilidade de simular, com uma precisão aceitável, uma faixa maior de pontos, com um período muito menor de tempo e sem os custos de implementação física do cenário de teste.

É importante ressaltar que com o uso da ferramenta TimeNET (ZIMMERMANN, 2017), podemos estimar a utilização das interfaces, através da fórmula P{#L_{Ui} > 0}, calculando assim a probabilidade de tokens em L_{Ui}. Em geral, as estimativas mostram um nível de precisão de 98,95%, sendo razoável do ponto de vista estocástico.

5.2 TAXA DE TRANSFERÊNCIA

Para os experimentos de taxa de transferência e de latência, foi utilizado o *Benchmark* NetPipe, que segundo (NETPIPE, 1998), utiliza uma série simples de testes de pingue-pongue em uma variedade de tamanhos de mensagens para fornecer uma medida completa do desempenho de uma rede. Ele envia mensagens de tamanho crescente entre dois processos através de uma rede. Os tamanhos das mensagens são escolhidos em intervalos regulares, e com pequenas perturbações, para fornecer uma avaliação completa do sistema de comunicação. Cada ponto de dados envolve muitos testes de pingue-pongue para fornecer um tempo preciso.

O *Benchmark NetPipe* registrou a taxa de transferência em Mbps, o tamanho da mensagem em *Bytes* e a latência em ms, no qual o servidor (*host* 1) recebe os pacotes enviados pelo cliente (*host* 2). Os comandos utilizados no servidor e no cliente estão apresentados nas Listagens 5.1 e 5.2 respectivamente.

Listing 5.1: Comando do *NetPipe* utilizado no servidor

```
$ NPtcp
```

Listing 5.2: Comando do *NetPipe* utilizado no cliente

```
$ NPtcp -h ip_destino
```

Utiliza-se o IP do servidor no lugar do "ip_destino". Posteriormente, com tudo configurado e testes para verificar o devido funcionamento, foi registrado os seguintes resultados, apresentados na Figura 11.

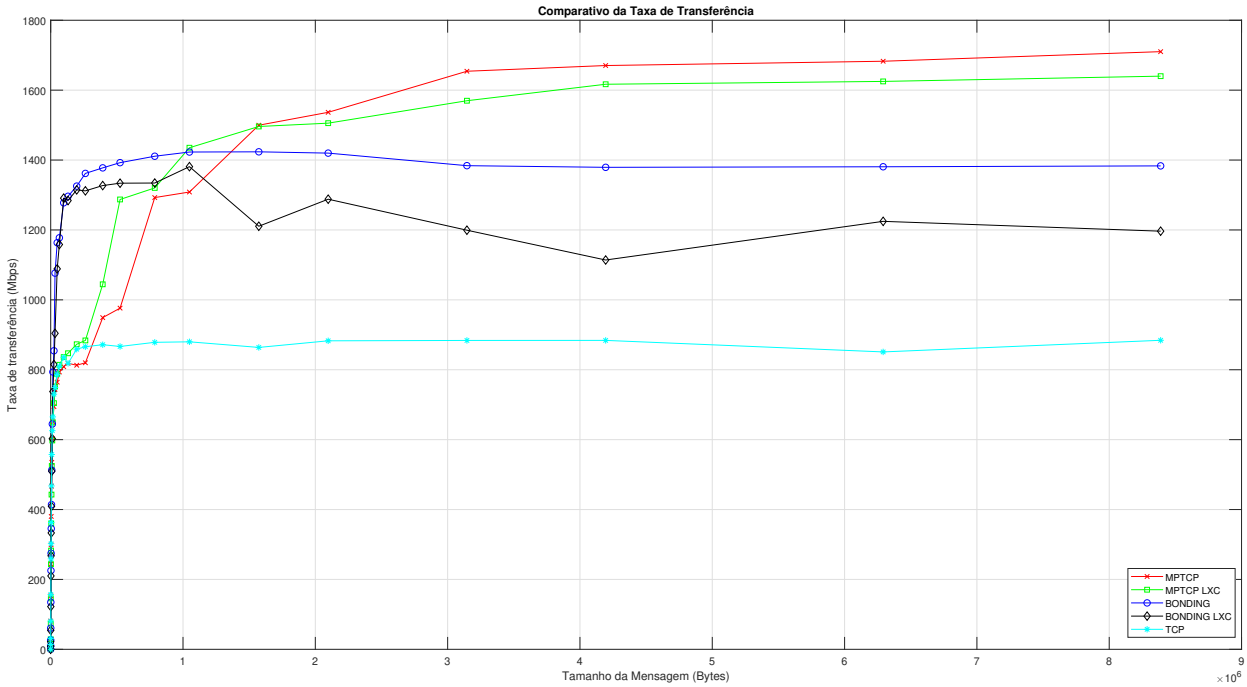

Figura 11: Comparativo Taxa de Transferência (NetPipe)

Conforme(COSTA, 2008) a taxa de transferência ou vazão dos dados demonstra o máximo de dados transportado pela rede desde sua origem até ao destino de chegada. Para as redes Ethernet a banda passante máxima absoluta será igual a taxa de dados, por exemplo, 10 Mbit/s, 100 Mbit/s ou 1000 Mbit/s.

Como pode ser visto na Figura 11, são apresentados os resultados obtidos nos experimentos de MPTCP, *Bonding* Balance-RR, MPTCP nos *containers* LXC, *Bonding* Balance-RR nos *containers* LXC e o TCP regular. Os resultados obtidos são admissíveis, não foi alcançado a taxa máxima permitida fisicamente que seria em torno de 1850 Mbit/s. Entretanto ficaram bem acima do TCP regular, o que já era de se esperar, pois usando apenas uma interface espera-se que o desempenho seja menor do que usando duas.

Os resultados em relação aos *containers* são mais baixos pelo fato de usarem mais o CPU dos *hosts*, resultando em um desempenho menor e em oscilações. Se os experimentos fossem realizados em *hosts* com maior desempenho computacional, os valores seriam mais próximos dos resultados obtidos nos *hosts* diretamente.

Os resultados obtidos nesse experimento, MPTCP no *host* nativo, foram usados também para comparação com os resultados obtidos pelo modelo da Figura 10. E através da Tabela 10, é possível ver mais claramente a porcentagem de uso das

interfaces, durante os experimento do MPTCP.

5.3 LATÊNCIA

De acordo com (COSTA, 2008), a latência define o tempo total que é gasto por um pacote no caminho entre a origem e o destino. Para calcular este parâmetro, são somados todos os atrasos de processamento de todos os elementos que consistem a rede. Para coletar os parâmetros, um pacote de teste é enviado contendo uma marca de tempo (*timestamp*). Quando o pacote é recebido no destino é enviado de volta ao receptor para que ocorra a análise do atraso de ida e volta.

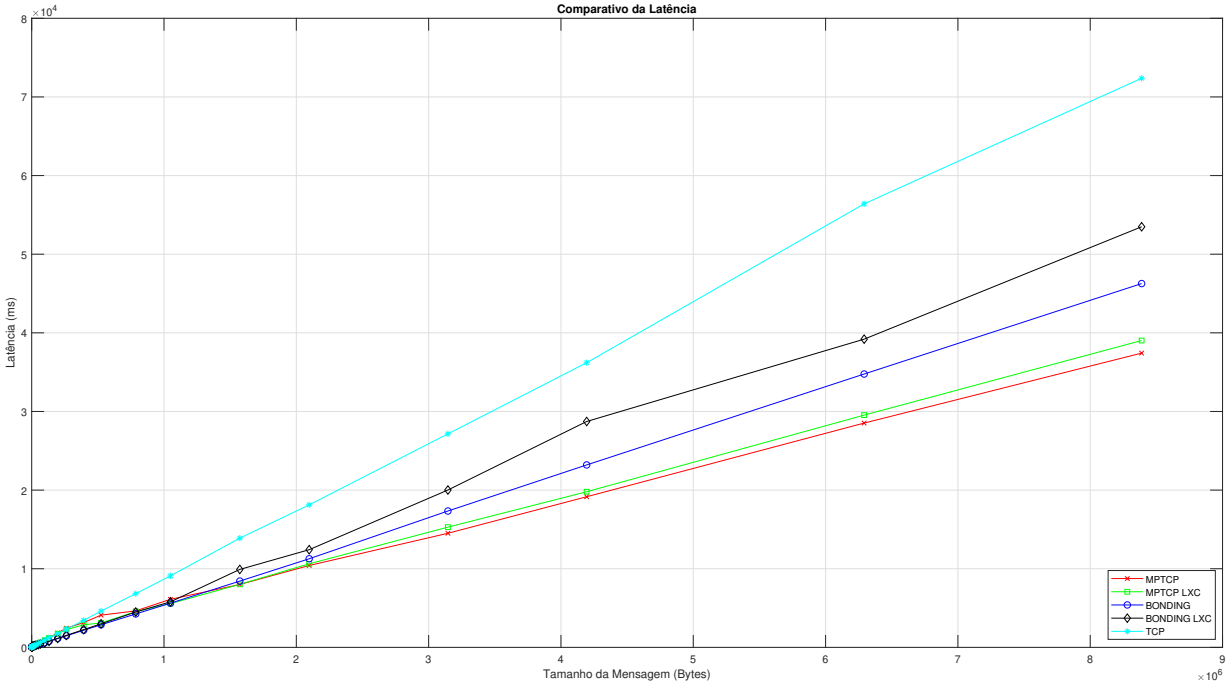

Figura 12: Comparativo Latência (NetPipe)

Para obter os resultados apresentados na Figura 12, foram usados o mesmo cenário que no teste de taxa de transferência. Como pode ser notado os resultados são próximos aos da taxa de transferência, ou seja, a latência é mais baixa quando em uso apenas do *host*, e quando em uso com os *containers* apresenta uma latência mais alta, com exceção do TCP regular.

Contudo, para uma real análise da rede é necessário testá-la em um ambiente de larga escala. Foi então utilizado o Hadoop, que fornece uma aplicação MapReduce, chamado TestDFSIO.

5.4 HADOOP

Para os testes com o Hadoop, foi utilizado a versão 3.3.0. O TestDFSIO testa o desempenho de E/S do HDFS. Ele faz isso usando um trabalho MapReduce como uma maneira conveniente de ler ou gravar arquivos em paralelo (WHITE, 2015).

TestDFSIO é um *benchmark* que enfatiza o *Hadoop Distributed File System* (HDFS) e é dividido em testes de leitura e gravação. É útil testar o HDFS, pois mede a taxa de transferência e o desempenho de armazenamento do *cluster* em termos de E/S de disco e rede. O *benchmark* TestDFSIO é projetado para usar uma tarefa de mapa por arquivo. As divisões são definidas de forma que cada operação de mapa obtenha apenas um nome de arquivo para leitura ou gravação (RADHAKRISHNAN et al., 2016).

Como encontrado em (IFSTAT, 2004), o Ifstat é uma pequena ferramenta para relatar a atividade da interface. Foi usada com o objetivo de monitorar o tráfego das interfaces.

Executamos o TestDFSIO no modo escrita com 30 arquivos com tamanho de 2Gb cada enquanto mediamos a taxa de transferência das interfaces com o Ifstat. A Figura 13 mostra um gráfico do tempo de execução versus a taxa de transferência para um *cluster* de dois nós.

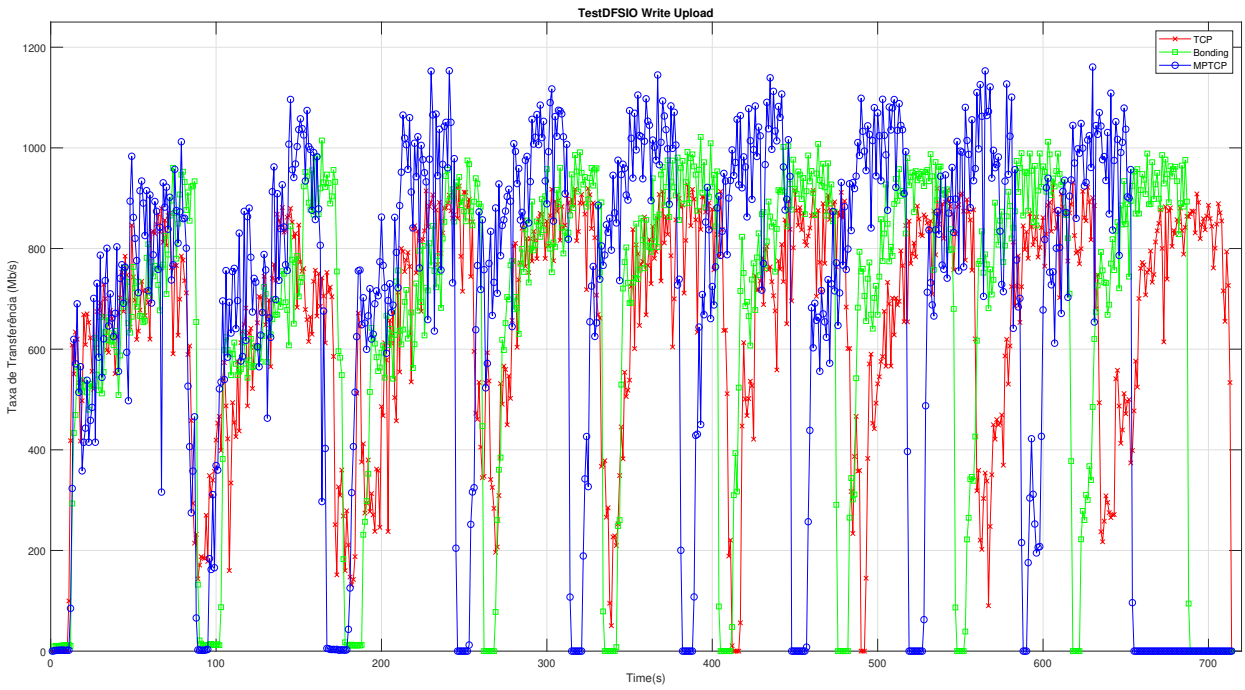

Figura 13: *TestDFSIO Write Upload*

Por limitações de hardware dos *hosts*, os testes nos *containers* LXC não puderam ser feitos. Pelo fato de que o LXC, MPTCP e o Bonding usarem muito da CPU, o *host* não foi capaz de suportar todo o processamento necessário para os testes.

Os resultados nos mostram que o MPTCP apresentou um desempenho melhor, alcançando maior taxa de transferência, terminando assim o teste em menor tempo do que o Bonding, e consequentemente que o TCP.

A taxa de transferência do TestDFSIO não chegou perto dos valores coletados no teste realizado com o *Benchmark NetPipe*. A razão para isso é novamente, a limitação do hardware dos *hosts*, pois durante os testes foi monitorado o uso da CPU, e durante praticamente todo o teste o uso fica em 100%, impedindo assim de se alcançar o melhor resultado da tecnologia, mas apenas conseguir o melhor resultado possível com as condições do hardware disponível.

Através do modelo da Figura 10, é possível avaliar o uso das interfaces de rede com MPTCP. Entretanto, devido a baixa capacidade de processamento dos *hosts*, os resultados obtidos com o experimento da Figura 13, não equivalem aos resultados encontrados pelo modelo, ou seja, os resultados encontrados pelo modelo apresentam uma média de 84% do uso das interfaces, muito próximo ao obtido na Figura 11, enquanto que a média do uso das interfaces obtidas na Figura 13 é abaixo de 50%.

A explicação por essa diferença de resultado da Figura 11 com a Figura 13, é o fato dos *hots* não conseguirem suportar toda a carga de processamento do Hadoop, impedindo assim de se chegar à valores mais próximos da Figura 11.

6 CONCLUSÃO

A pesquisa realizada por esse trabalho buscou um modelo do comportamento da rede com o uso do MPTCP através de GSPN, em conjunto de uma análise de desempenho da rede do MPTCP, em relação ao LACP no uso de aplicação de larga escala como o Hadoop, apresentado no trabalho de (RISTA et al., 2017).

Entretanto durante o desenvolvimento do trabalho, foi constatado limitações tanto em relação aos equipamentos fornecidos, quanto na falta de alguns equipamentos necessários para a reprodução dos testes realizados por (RISTA et al., 2017). Os *switches* fornecidos eram de 100 Mbps (*Fast Ethernet*), ou seja, não possuem largura de banda suficiente para aplicações Hadoop do mundo real, gerando assim gargalos de desempenho, impedindo uma análise mais precisa do experimento.

Para contornar tais problemas optou-se por fazer algumas alterações, como era imprescindível o uso de um *switch* para os testes do LACP, optou-se por trocarmos o LACP por *Bonding Balance-RR*, assim seria possível fazer os testes sem o uso de qualquer *switch*. Outro obstáculo encontrado foi a limitação do hardware dos *hosts*, pois as CPUs, durante os experimentos do Hadoop, usavam 100% de sua capacidade, gerando assim um gargalo de desempenho. Por esse motivo não foi possível realizar os teste de larga escala nos *containers* LXC, e afetando também alguns resultados obtidos como o experimento em larga escala dos *hosts*.

Apesar das restrições encontradas, foi viável constatar algumas conclusões. A primeira é que o modelo mostrado na Figura 10, mostrou-se condizente com os valores encontradas nos testes feitos em ambiente nativo, como é apresentado na Figura 11. Provando assim ser útil na simulação de um cenário em que se faça necessário o uso do MPTCP. Desse modo, é possível prever o comportamento de rede sem a necessidade de montar toda a rede necessária para tal. Economizando tanto em tempo como em custo.

O modelo pode ser adaptado para a situação desejada, neste trabalho foi usado

um modelo com duas interfaces, todavia as interfaces do modelo podem ser aumentadas sem comprometer a simulação.

Outro ponto foi que o uso do MPTCP apresentou resultados melhores do que os encontrados com o uso do *Bonding Balance-RR*.

Como não foi adquirido o *switch* para o experimento do LACP, todo o desempenho dos experimentos foi realizado pela CPU , diferentemente do LACP que faz o uso do *switch* para o seu funcionamento. Com isso foi possível perceber que o MPTCP é mais otimizado quando comparado com o *Bonding Balance-RR*, pois foi possível alcançar valores mais elevados na taxa de tranferência e valores mais baixos na latência, mesmo quando comparamos o uso do MPTCP em relação ao uso do *Bonding Balance-RR*.

Apesar de não ser possível validar o modelo através do uso do Hadoop, pois os mesmos apresentaram resultados muito abaixo do esperado, e a justificativa para isso é a limitação do hardware, incapaz de suportar a carga de trabalho exigida pelo Hadoop. Ainda assim, foi possível a validação do modelo através dos resultados obtidos com o experimento de taxa de transferência (Figura 11) realizado no ambiente nativo.

APÊNDICE A – MPTCP

A.1 CONFIGURAÇÃO DAS INTERFACES DO *HOST* 1

```
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

#Interface para acesso a internet
auto enp63s0
iface enp63s0 inet manual

auto brnet
iface brnet inet dhcp
bridge_ports enp63s0

#Interfaces de Comunicacao MPTCP
auto enp47s0
iface enp47s0 inet manual

auto breth0
iface breth0 inet static
        bridge_ports enp47s0
        bridge_stp off
        bridge_fd 0
        address 10.1.1.2
        network 10.1.1.0
        netmask 255.255.255.0
        gateway 10.1.1.1
```

```
auto enp31s0
        iface enp31s0 inet manual

auto breth1
iface breth1 inet static
        bridge_ports enp31s0
        bridge_stp off
        bridge_fd 0
        address 10.1.2.2
        network 10.1.2.0
        netmask 255.255.255.0
        gateway 10.1.2.1
```

A.2 CONFIGURAÇÃO DAS INTERFACES DO *HOST* 02

```
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

#Interface para acesso a internet
auto enp63s0
iface enp63s0 inet manual

auto brnet
iface brnet inet dhcp
        bridge_ports enp63s0

#Interfaces de Comunicacao MPTCP
auto enp47s0
iface enp47s0 inet manual

auto breth0
iface breth0 inet static
```

```
        bridge_ports enp47s0
        bridge_stp off
        bridge_fd 0
        address 10.1.1.4
        network 10.1.1.0
        netmask 255.255.255.0
        gateway 10.1.1.1
        dns-nameservers 8.8.8.8

auto enp31s0
iface enp31s0 inet manual

auto breth1
iface breth1 inet static
        bridge_ports enp31s0
        bridge_stp off
        bridge_fd 0
        address 10.1.2.4
        network 10.1.2.0
        netmask 255.255.255.0
        gateway 10.1.2.1
        dns-nameservers 8.8.8.8
```

APÊNDICE B – BALANCE-RR

B.1 CONFIGURAÇÃO DAS INTERFACES DO *HOST* 1

```
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

#Interface para acesso a internet
auto enp63s0
iface enp63s0 inet dhcp

#Configuracao Bond
auto enp47s0
iface enp47s0 inet manual

auto enp31s0
iface enp31s0 inet manual

auto bond0
iface bond0 inet manual
        slaves enp47s0 enp31s0
        bond-mode 0

#Configuracao Bridge
auto br0
iface br0 inet static
        address 10.3.3.2
        netmask 255.255.255.0
```

```
        gateway 10.3.3.1
        network 10.3.3.0
        bridge_ports bond0
```

B.2 CONFIGURAÇÃO DAS INTERFACES DO *HOST* 2

```
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

#Interface para acesso a internet
auto enp63s0
iface enp63s0 inet dhcp

#Configuracao Bond
auto enp47s0
iface enp47s0 inet manual

auto enp31s0
iface enp31s0 inet manual

auto bond0
iface bond0 inet manual
        slaves enp47s0 enp31s0
        bond-mode 0

#Configuracao Bridge
auto br0
iface br0 inet static
        address 10.3.3.3
        netmask 255.255.255.0
        gateway 10.3.3.1
        network 10.3.3.0
        bridge_ports bond0
```

APÊNDICE C – LXC

C.1 CRIAÇÃO DAS INTERFACES DO CONTAINER DO *HOST* 1

```
#Internet
lxc.net.2.type = veth
lxc.net.2.name = ethnet
lxc.net.2.flags = up
lxc.net.2.link = brnet

#Bonding
#Configuracao eth0bond
lxc.net.0.type = veth
lxc.net.0.name = eth0bond
lxc.net.0.flags = up
lxc.net.0.link = br0

#MPTCP
#Configuracao eth0
lxc.net.0.type = veth
lxc.net.0.name = eth0
lxc.net.0.flags = up
lxc.net.0.link = breth0

#Configuracao eth1
lxc.net.1.type = veth
lxc.net.1.name = eth1
lxc.net.1.flags = up
lxc.net.1.link = breth1
```

C.2 CONFIGURAÇÃO DAS INTERFACES DO *CONTAINER* 1 PARA USO DO MPTCP

```
auto lo
iface lo inet loopback

#Internet
auto ethnet
iface ethnet inet dhcp

auto eth0
iface eth0 inet static
        address 10.1.1.3
        network 10.1.1.0
        netmask 255.255.255.0
        gateway 10.1.1.1

auto eth1
iface eth1 inet static
        address 10.1.2.3
        network 10.1.2.0
        netmask 255.255.255.0
        gateway 10.1.2.1
```

C.3 CONFIGURAÇÃO DAS INTERFACES DO *CONTAINER* 2 PARA USO DO MPTCP

```
auto lo
iface lo inet loopback

#Internet
auto ethnet
iface ethnet inet dhcp

auto eth0
iface eth0 inet static
        address 10.1.1.5
```

```
        network 10.1.1.0
        netmask 255.255.255.0
        gateway 10.1.1.1

auto eth1
iface eth1 inet static
        address 10.1.2.5
        network 10.1.2.0
        netmask 255.255.255.0
        gateway 10.1.2.1
```

C.4 CONFIGURAÇÃO DAS INTERFACES DO *CONTAINER* 1 PARA USO DO BONDING

```
auto lo
iface lo inet loopback

#Internet
auto ethnet
iface ethnet inet dhcp

auto eth0bond
iface eth0bond inet static
        address 10.3.3.4
        network 10.3.3.0
        netmask 255.255.255.0
        gateway 10.3.3.1
```

C.5 CONFIGURAÇÃO DAS INTERFACES DO *CONTAINER* 2 PARA USO DO BONDING

```
auto lo
iface lo inet loopback

#Internet
```

```
auto ethnet
iface ethnet inet dhcp

auto eth0bond
iface eth0bond inet static
        address 10.3.3.5
        netmask 255.255.255.0
        gateway 10.3.3.1
        network 10.3.3.0
```

REFERÊNCIAS

BONAVENTURE, O.; HANDLEY, M.; RAICIU, C. *An Overview of Multipath TCP*. ***The Usenix Magazine***, p. 17–23, 2012.

BONDAN, L.; GOBBI, R. C.; FOCHI, V. M. O protocolo de agregacão de enlaces lacp. **Trabalho de Conclusão de Curso, Pontifícia Universidade Católica do Rio Grande do Sul**, 2012.

CARDOSO, J.; VALETTE, R. **Redes de Petri**. Florianópolis: Editora da UFSC, 1997.

CELESTINO, E. **Conexão TCP-Hand Shake Triplo**. 2017. Disponível em: <https://bloghandsonlabs.wordpress.com/2017/02/19/conexao-tcp-hand-shake-triplo/>.

CHANSLER, R. et al. **The Hadoop Distributed File System**. 2010. Disponível em: <http://www.aosabook.org/en/hdfs.html>.

CHRIST, M. J. K.; MALISZEWSKI, A. M.; GRIEBLER, D. Avaliação do desempenho do protocolo bonding em máquinasvirtuais lxc e kvm. **ERRC**, 2018.

COSTA, G. H. d. Métricas para avaliação de desempenho em redes qos sobre ip. **Trabalho de Conclusão**, 2008.

COUTINHO, E. F. et al. *Elasticity in cloud computing: a survey*. ***annals of telecommunications - annales des télécommunications***, p. 289–309, 2014.

DAVIDSON, A.; OR, A. Optimizing shuffle performance in spark. **University of California, Berkeley-Department of Electrical Engineering and Computer Sciences, Tech. Rep**, 2013.

DAVIS, T. et al. **Bonding**. 2000. Disponível em: <https://wiki.linuxfoundation.org/networking/bonding>.

DEAN, J.; GHEMAWAT, S. **MapReduce: Simplified Data Processing on Large Clusters**. 2004. Disponível em: <https://www.usenix.org/legacy/event/osdi04/tech/full_papers/dean/dean_html>.

DESROCHERS, A. A. *Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis*. **IEEE Press**, 1994.

EKANAYAKE, J.; FOX, G. High performance parallel computing with clouds and cloud technologies. **Berlin, Heidelberg: Springer Berlin Heidelberg**, 2010.

FOUNDATION, T. A. S. ***Apache Hadoop***. 2018. Disponível em: <https://hadoop.apache.org/>.

HE, H.; PANG, S.; ZHAO, Z. Dynamic scalable stochastic petri net:a novel model for designing and analysis ofresource scheduling in cloud computing. **Hindawi Publishing Corporation**, 2016.

HINTERSTEINER, J. Melhores práticas para switches de rede gerenciados. **Anais**, 2016.

HUSTON, G. **Multipath TCP**. 2015. Disponível em: <https://labs.apnic.net/?p=635>.

IFSTAT. 2004. https://linux.die.net/man/1/ifstat. Acessado em junho de 2019.

ISLAM, S. et al. *How a consumer can measure elasticity for cloud platforms*. ***In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering***, p. 85–96, 2012.

KARTSON, D. et al. *Modelling with Generalized Stochastic Petri Nets*. ***John Wiley and Sons***, p. –, 2005.

LXC. ***Linux containers project (lxc)***. 2016. Disponível em: <https://linuxcontainers.org>.

MALISZEWSKI, A. M. et al. *The NAS Benchmark Kernels for Single and Multi-Tenant Cloud Instances with LXC/KVM*. **2018 HPCS**, 2018.

MELL, P.; GRANCE, T. *The nist definition of cloud computing*. ***National Institute of Standards and Technology***, p. 2, 2009.

MPTCP. **MultiPath TCP - Linux Kernel implementation**. 2007. Disponível em: <https://multipath-tcp.org/pmwiki.php/Users/AptRepository>.

NASSER, R. B.; BREITMAN, K. K. *McCloud Service Framework:* arcabouço para desenvolvimento de serviços baseados na simulação de monte carlo na *Cloud*. **Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro**, p. 21, 2012.

NETPIPE. 1998. https://linux.die.net/man/1/netpipe. Acessado em maio de 2019.

OLTEANU, V.; RAICIU, C. *Datacenter Scale Load Balancing for Multipath Transport*. ***University Politehnica of Bucharest***, p. 20, 2016.

PAASCH, S. B. C. ***Manual configuration***. 2007. Disponível em: <https://multipath-tcp.org/pmwiki.php/Users/ConfigureRouting>.

PENGSONG, D. Evaluation model of the cloud systems based on queuing petri net. **International Conference on Algorithms and Architectures for Parallel Processing**, 2015.

PETERSON, J. L. **Petri nets**. [S.l.]: ACM Computing Surveys (CSUR), 1977.

RADHAKRISHNAN, S.; MUSCEDERE, B. J.; DAUDJEE, K. V-hadoop: Virtualized hadoop using containers. **2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)**, 2016.

RISTA, C. et al. *Improving the network performance of a container-based cloud environment for hadoop systems*. ***International Conference on High Performance Computing Simulation (HPCS)***, p. 619–626, 2017.

RISTA, C. et al. *Evaluating, estimating, and improving network performance in container-based clouds*. ***IEEE Symposium on Computers and Communications***, 2018.

RISTA, L. C. G. Provisionamento horizontal autônomo e elástico de recursos em cloud para paralelismo de stream. **Pontificia Universidade Católica do Rio Grande do Sul**, p. 28, 2018.

SARABANDO, N. G. *Validation of "triple-play"services in the access node*. **Trabalho de Mestrado, Universidade de Aveiro**, 2008.

SCHAD, J.; DITTRICH, J.; QUIANé-RUIZ, J. *Runtime measurements in the cloud: Observing, analyzing, and reducing variance*. ***Proceedings of the VLDB Endowment***, p. 460–471, 2010.

SEO, S. **KT's GiGA LTE**. 2015. Disponível em: <https://www.ietf.org/proceedings/93/slides/slides-93-mptcp-3.pdf>.

SOUSA, F. R. C. et al. Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. ***Brazilian Symposium on Databases***, p. 101–130, 2010.

VIANA, N. P. Um serviço para flexibilização da tarifação em nuvens de infraestrutura. **Dissertação(Pós-Graduação em Ciência da Computação)**, p. –, 2013.

WHITE, T. **Hadoop The Definitive Guide Storage and Analysis at Internet Scale**. [S.l.]: O'Reilly Media, 2015.

YAZDANOV, L.; GORBUNOV, M.; FETZER, C. Ehadoop: Network i/o aware scheduler for elastic mapreduce cluster. **8th Inter. Conference on Cloud Computing (CLOUD)**, 2015.

ZIMMERMANN, A. ***TimeNET***. 2017. Disponível em: <https://timenet.tu-ilmenau.de/>.