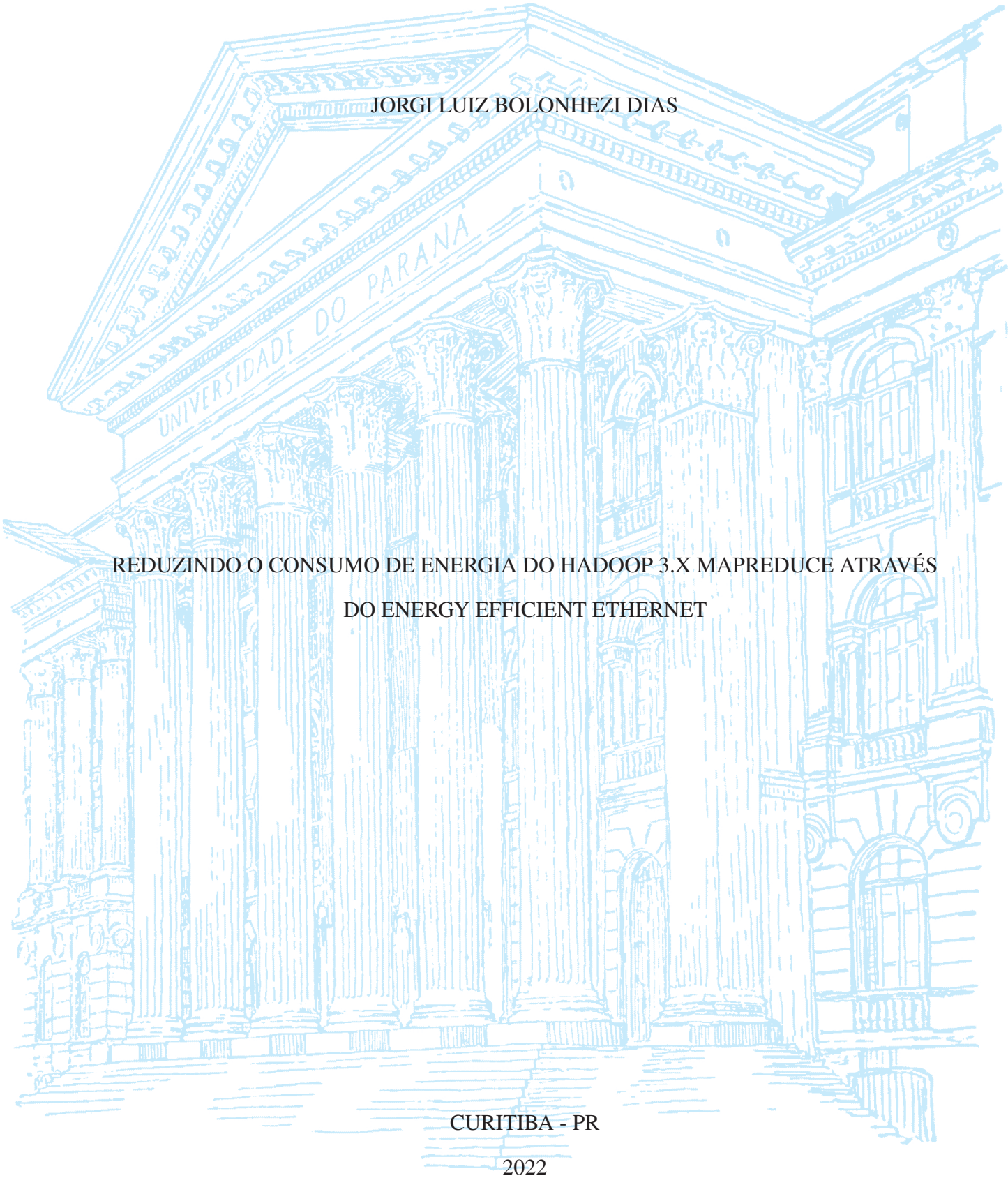
UNIVERSIDADE FEDERAL DO PARANÁ

JORGI LUIZ BOLONHEZI DIAS




# REDUZINDO O CONSUMO DE ENERGIA DO HADOOP 3.X MAPREDUCE ATRAVÉS DO ENERGY EFFICIENT ETHERNET

CURITIBA - PR

2022

JORGI LUIZ BOLONHEZI DIAS

# REDUZINDO O CONSUMO DE ENERGIA DO HADOOP 3.X MAPREDUCE ATRAVÉS DO ENERGY EFFICIENT ETHERNET

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Luiz Carlos Pessoa Albini.

Coorientador: Leandro Batista De Almeida.

CURITIBA - PR

2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Dias, Jorgi Luiz Bolonhezi
Reduzindo o consumo de energia do Hadoop 3.x MapReduce através do energy efficient ethernet / Jorgi Luiz Bolonhezi Dias – Curitiba, 2022.
1 recurso on-line : PDF.

Dissertação (Mestrado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.
Orientador: Luiz Carlos Pessoa Albini
Coorientador: Leandro Batista de Almeida

1. Big data. 2. Eficiência energética (Energy Efficient Ethernet). I. Albini, Luiz Carlos Pessoa. II. Almeida, Leandro Batista de. III. Universidade Federal do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

# TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **JORGI LUIZ BOLONHEZI DIAS** intitulada: **REDUZINDO O CONSUMO DE ENERGIA DO HADOOP 3.X MAPREDUCE ATRAVÉS DO ENERGY EFFICIENT ETHERNET**, sob orientação do Prof. Dr. LUIZ CARLOS PESSOA ALBINI, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 25 de Fevereiro de 2022.

Assinatura Eletrônica
04/03/2022 14:35:14.0
LUIZ CARLOS PESSOA ALBINI
Presidente da Banca Examinadora

Assinatura Eletrônica
04/03/2022 14:26:14.0
EDUARDO CUNHA DE ALMEIDA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
07/03/2022 08:29:51.0
HERMES IRINEU DEL MONEGO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

*A toda equipe da UFPR pela oportunidade de estudo.*

## AGRADECIMENTOS

Agradeço a minha família, pelo apoio e incentivo nas horas difíceis. Aos professores, pelos ensinamentos e correções que me permitiram apresentar um melhor desempenho na minha formação profissional. Aos orientadores Luiz e Leandro, pelo seu grande desprendimento em ajudar e amizade sincera. Por fim, agradeço a todos que contribuíram diretamente ou indiretamente durante a formação.

## RESUMO

O consumo de energia é um dos maiores desafios na infraestrutura de processamento de *Big Data*. Atualmente os gastos com energia são ainda maiores do que na aquisição do *hardware*, representando 75% do custo total dos *data centers*. Aproximadamente 30% de toda a energia do *data center* é consumida por *switches* de rede. O *Energy Efficient Ethernet* é um padrão recente que visa reduzir o consumo de energia, embora a prática atual na indústria seja desativá-lo em produção, pois pode causar sobrecargas na rede e perda de desempenho. Esta dissertação fornece uma visão geral de como a atual versão do Apache Hadoop, a 3.x, se comporta com o *Energy Efficient Ethernet* habilitado para links de 1GbE até 400GbE. Os resultados apresentados mostram que há economia de energia significativa com pouca ou nenhuma perda de desempenho para conexões de até 40GbE. No entanto, conexões de 100GbE e 400GbE apresentam perdas significativas de desempenho devido ao despertar do link para transmissões de um único *frame*.

Palavras-chave: Green Networking, Big Data, Energy Efficient Ethernet, MapReduce, Hadoop.

## ABSTRACT

Energy consumption is one of the major challenges on the big data processing infrastructure. The energy expenses are even higher than hardware, accounting for 75% of the total cost of nowadays data centers. Narrowing, approximately 30% of all data center energy is consumed by the network switches. Energy Efficient Ethernet is a recent standard aiming at reduce network power consumption, notwithstanding the current practice in industry is to disable it in production use, since it can cause network overloads and performance loss. This thesis provides an overview on how Apache Hadoop 3.x, the current version, behaves with Energy Efficient Ethernet enabled for links from 1GbE up to 400GbE links. Presented results show that there is significant energy savings with little or no performance loss for connections up to 40GbE. Nevertheless, connections of 100GbE and 400GbE present significant performance losses due to link wake up to single transmissions.



Keywords: Green Networking, Big Data, Energy Efficient Ethernet, MapReduce, Hadoop.

## LISTA DE FIGURAS


1.1 Tráfego em Data Centers estimado pela Cisco . . . . . 15
1.2 Tráfego em Data Centers por Categoria estimado pela Cisco . . . . . 16
1.3 Previsões de Tráfegos para 2021 de acordo com a Cisco . . . . . 16

2.1 Overview of EEE LPI Operation . . . . . 19
2.2 Overview of EEE fast wake operation . . . . . 21
2.3 EEE with Packet Coalescing . . . . . 22
2.4 The Random Early Detection (RED) queue management scheme . . . . . 23
2.5 ECN in Action . . . . . 24
2.6 Etapas de uma Aplicação MapReduce WordCount . . . . . 26
2.7 Aplicações YARN . . . . . 29
2.8 Arquitetura do SLS . . . . . 32
2.9 Gráficos gerados em tempo real pelo YARN Scheduler Load Simulator . . . . . 32

3.1 Esquema do NetSLS sendo controlado por um OpenFlow externo . . . . . 36
3.2 Topologia de Rede Leaf-Spine . . . . . 37

4.1 Comparação do consumo de energia entre *Ethernet* Padrão e EEE no modo *Deep Sleep* em conexões de 1GbE, 10GbE e 25GbE . . . . . 44
4.2 Comparação do consumo de energia entre *Ethernet* Padrão e EEE no modo *Deep Sleep* em conexões de 40GbE, 100GbE e 400GbE . . . . . 45
4.3 Comparação do consumo de energia entre *Ethernet* Padrão e EEE no modo *Fast Wake* em conexões de 40GbE e 100GbE . . . . . 45
4.4 Comparação de desempenho entre *Ethernet* Padrão e EEE no modo *Deep Sleep* em conexões de 1GbE, 10GbE e 25GbE . . . . . 46
4.5 Comparação de desempenho entre *Ethernet* Padrão e EEE no modo *Deep Sleep* em conexões de 40GbE, 100GbE e 400GbE . . . . . 47
4.6 Comparação de desempenho entre *Ethernet* Padrão e EEE no modo *Fast Wake* em conexões de 40GbE e 100GbE . . . . . 47

5.1 Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing* . . . . 50
5.2 Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing* . . . . 51
5.3 Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing* . . . . . 52
5.4 Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing* . . . . . 53
5.5 Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDeL e ECN . . . . . 54

5.6 Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*, CoDeL e ECN . . . . . 55
5.7 Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDel e ECN 56
5.8 Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*, CoDel e ECN 57
5.9 Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN . . . . . 58
5.10 Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*, RED e ECN . . . . . 59
5.11 Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN. 60
5.12 Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*, RED e ECN . 60
5.13 Comparação dos resultados de consumo de energia entre Nossa Pesquisa e o E-EON . . . . . 61
5.14 Comparação dos resultados de performance entre Nossa Pesquisa e o E-EON . . 62
5.15 Comparação dos resultados de consumo de energia com as melhores configurações entre Nossa Pesquisa e o E-EON . . . . . 62
5.16 Comparação dos resultados de performance com as melhores configurações entre Nossa Pesquisa e o E-EON . . . . . 63

## LISTA DE TABELAS


2.1 EEE 150-byte frame Efficiency . . . . . 20
2.2 EEE 1500-byte frame Efficiency . . . . . 20
2.3 A comparison of MapReduce 1 and YARN components . . . . . 29
2.4 Comparação entre Hadoop 2.x e Hadoop 3.x . . . . . 31
2.5 Tempo de execução das tarefas no Hadoop 2.x e 3.x . . . . . 31

3.1 Topologia Simulada . . . . . 38
3.2 Configuração dos Switches . . . . . 39
3.3 Configurações do Cluster Hadoop . . . . . 39
3.4 EEE - Tempo das operações *wake* e *sleep* no modo *Deep Sleep* . . . . . 40
3.5 EEE - Tempos assumidos para as operações *wake* e *sleep* no modo *Deep Sleep* . . 40
3.6 EEE - Tempo das operações *wake* e *sleep* no modo *Fast Wake* . . . . . 40
3.7 Configurações do *Packet Coalescing* . . . . . 40
3.8 Valores definidos $Min_{th}$ e $Max_{th}$ para o *Random Early Detection* . . . . . 41
3.9 Valores definidos para o *Controlled Delay* . . . . . 41
3.10 Cargas de Trabalho Simuladas . . . . . 42

## LISTA DE ACRÔNIMOS

| | |
|---|---|
| ACK | *Acknowledge* |
| API | *Application Programming Interface* |
| AM | *Application Master* |
| AQM | *Active Queue Management* |
| CDN | *Content Delivery Networks* |
| CE | *Congestion Experienced* |
| CoDel | *Controlled Delay* |
| CPU | *Central Processing Unit* |
| CWR | *Congestion Window Reduced* |
| EC | *Erasure Coding* |
| ECN | *Explicit Congestion Notification* |
| ECE | *ECN-Echo* |
| ECT | *ECN-Capable Transport* |
| EEE | *Energy Efficient Ethernet* |
| E-EON | *Energy-Efficient and Optimized Networks* |
| FIFO | *First In First Out* |
| GC | *Grid Computing* |
| GB | *Gigabytes* |
| HDFS | *Hadoop Distributed File System* |
| HPC | *High-performance Computing* |
| IEEE | *Institute of Electrical and Electronics Engineers* |
| IP | *Internet Protocol* |
| I/O | *Imput and Ouput* |
| JVM | *Java Virtual Machine* |
| KWh | Quilowatt-hora |
| LAN | *Local Area Network* |
| LPI | *Low Power Idle* |
| MADLFS | *Microsoft Azure Data Lake File System* |
| MII | *Media Independent Interface* |
| MRJ | *MapReduce Job* |
| NM | *Node Manager* |
| NIC | *Network Interface Controller* |
| RAM | *Random Access Memory* |
| RED | *Random Early Detection* |
| RM | *Resource Manager* |

| | |
|---|---|
| SAN | *Storage Area Network* |
| SYN | *Synchronize* |
| TCP | *Transmission Control Protocol* |
| TWh | Terawatt-hora |
| WAN | *Wide Área Network* |
| Wh | Watt-hora |
| YARN | *Yet Another Resource Negotiator* |

**SUMÁRIO**


**1 INTRODUÇÃO . . . . . 14**
1.1 OBJETIVOS E CONTRIBUIÇÕES . . . . . 17
1.2 ESTRUTURA DA DISSERTAÇÃO . . . . . 18
**2 FUNDAMENTAÇÃO TEÓRICA . . . . . 19**
2.1 ENERGY EFFICIENT ETHERNET . . . . . 19
2.1.1 Deep Sleep Mode . . . . . 19
2.1.2 Fast Wake Mode. . . . . 21
2.2 PACKET COALESCING . . . . . 21
2.3 ACTIVE QUEUE MANAGEMENT . . . . . 22
2.3.1 Random Early Detection . . . . . 22
2.3.2 Controlled Delay . . . . . 23
2.4 EXPLICIT CONGESTION NOTIFICATION . . . . . 24
2.5 MAPREDUCE . . . . . 25
2.6 APACHE HADOOP . . . . . 26
2.6.1 Módulos do Ecossistema Hadoop. . . . . 27
2.6.2 Processos do Hadoop . . . . . 27
2.6.3 Aplicações YARN . . . . . 28
2.6.4 Hadoop 1.x, 2.x e 3.x . . . . . 29
2.6.5 YARN Scheduler Load Simulator . . . . . 31
2.7 CONSIDERAÇÕES FINAIS . . . . . 33
**3 METODOLOGIA . . . . . 35**
3.1 FERRAMENTA DE SIMULAÇÃO . . . . . 35
3.2 TOPOLOGIA . . . . . 37
3.3 CONSUMO DE ENERGIA DOS SWITCHES. . . . . 38
3.4 CONFIGURAÇÕES DO CLUSTER HADOOP . . . . . 39
3.5 CONFIGURAÇÕES DO ENERGY EFFICIENT ETHERNET . . . . . 39
3.6 CONFIGURAÇÕES DO PACKET COALESCING . . . . . 40
3.7 CONFIGURAÇÕES DO ACTIVE QUEUE MANAGEMENT E EXPLICIT CONGESTION NOTIFICATION . . . . . 41
3.8 CARGAS DE TRABALHO. . . . . 41
3.9 CICLO DE TESTES . . . . . 42
3.10 CONSIDERAÇÕES FINAIS . . . . . 42

**4 ENERGY EFFICIENT ETHERNET EM CLUSTERS HADOOP MAPREDUCE . . . . . 44**
4.1 ECONOMIA DE ENERGIA . . . . . 44
4.2 DESEMPENHO . . . . . 46
4.3 CONSIDERAÇÕES FINAIS . . . . . 48
**5 ECONOMIA DE ENERGIA, CONTROLE DA LATÊNCIA E OTIMIZAÇÃO DE DESEMPENHO. . . . . 49**
5.1 ENERGY EFFICIENT ETHERNET COM PACKET COALESCING. . . . . 49
5.1.1 Economia de Energia . . . . . 49
5.1.2 Desempenho . . . . . 51
5.2 ENERGY EFFICIENT ETHERNET COM PACKET COALESCING, CONTROLLED DELAY E EXPLICIT CONGESTION NOTIFICATION . . . . . 53
5.2.1 Economia de Energia . . . . . 53
5.2.2 Desempenho . . . . . 55
5.3 ENERGY EFFICIENT ETHERNET COM PACKET COALESCING, RANDOM EARLY DETECTION E EXPLICIT CONGESTION NOTIFICATION. . . . . 57
5.3.1 Economia de Energia . . . . . 57
5.3.2 Desempenho . . . . . 59
5.4 COMPARAÇÃO COM O E-EON . . . . . 61
5.5 CONSIDERAÇÕES FINAIS . . . . . 63
**6 CONCLUSÃO . . . . . 65**
6.1 TRABALHOS FUTUROS . . . . . 66
6.2 PUBLICAÇÃO DOS RESULTADOS . . . . . 66
**REFERÊNCIAS . . . . . 67**

# 1 INTRODUÇÃO

Estima-se que em 2020 já existiam cerca de 40.000 *exabytes* de dados digitalizados no mundo, e uma fração de 33% destes dados podem ser valiosos depois de analisados (Gantz e Reinsel, 2012). Construir *clusters* com capacidade de armazenar e processar essa enorme quantidade de dados para extrair informações úteis às empresas tem-se tornado um enorme desafio, principalmente devido ao ritmo de crescimento exponencial deste volume de dados.

Nas últimas duas décadas o Hadoop, uma plataforma de código fonte aberto mantida pela Apache Software Foundation para criar e gerenciar Infraestruturas de *clusters* baseados em *MapReduce* tem se destacado. A principal motivação para uso do Hadoop é sua escalabilidade, que permite unificar várias máquinas convencionais gerando grandes capacidades de armazenamento e processamento em massa de dados estruturados e, principalmente, não estruturados (Goldman et al., 2012).

Atualmente servidores com Hadoop tem sido oferecido no mercado por multinacionais como Amazon, Microsoft, Google, entre outras. Estes serviços fazem parte da computação em nuvem e utilizam conexões Ethernet para trocas de informações (tráfego) entre *clusters*, e consequentemente, aumentam o consumo de energia nos *Data Centers*. Tráfegos de dados gerados nas comunicações entre *clusters* em empresas privadas também são contabilizados como tráfego da computação em nuvem pela Cisco (Cisco, 2020).

Embora seu conceito tenha surgido em 1960, a computação em nuvem só ganhou forma no século XXI devido à indisponibilidade de recursos para implementá-la de fato. Computação em nuvem é um termo utilizado para descrever o ambiente de computação baseado em uma imensa rede de servidores virtuais e físicos, que geram um conjunto de recursos com capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na Internet (Taurion, 2009). Dentre suas principais características pode-se notar:

- A sensação de possuir recursos infinitos disponibilizados sob demanda do usuário;
- Elimina a necessidade de adquirir recursos antecipadamente, como por exemplo, servidores para implementar negócios;
- Disponibilidade elástica de recursos, permitindo que os usuários usem os recursos na quantidade que forem necessários, aumentando e diminuindo a capacidade computacional de forma dinâmica;
- O pagamento final se dá somente pela quantidade de recursos computacionais utilizados (*pay-per-use*).

No início dos anos 2000, multinacionais como Google e Amazon começaram a criar imensos parques computacionais baseados no conceito de computação em nuvem para operarem seus negócios, e logo adiante outras seguiram seus exemplos, como a IBM, Intel, Apple, Microsoft, Yahoo, Apache Software Foundation, Ask, entre outras (Taurion, 2009).

A Cisco vem publicando artigos nos últimos anos que demonstram o impacto deste investimento e desenvolvimento de tecnologias no tráfego de dados em nuvem. Em 2013 o tráfego de *Cloud Data Centers* já representava cerca de 54% de todo o tráfego em *Data Centers* globais (Cisco, 2014). A partir daí só continuou a aumentar, em 2018 já eram 76%, em 2020 cerca de 92%, e em 2021 foi de aproximadamente 95% (Cisco, 2018).

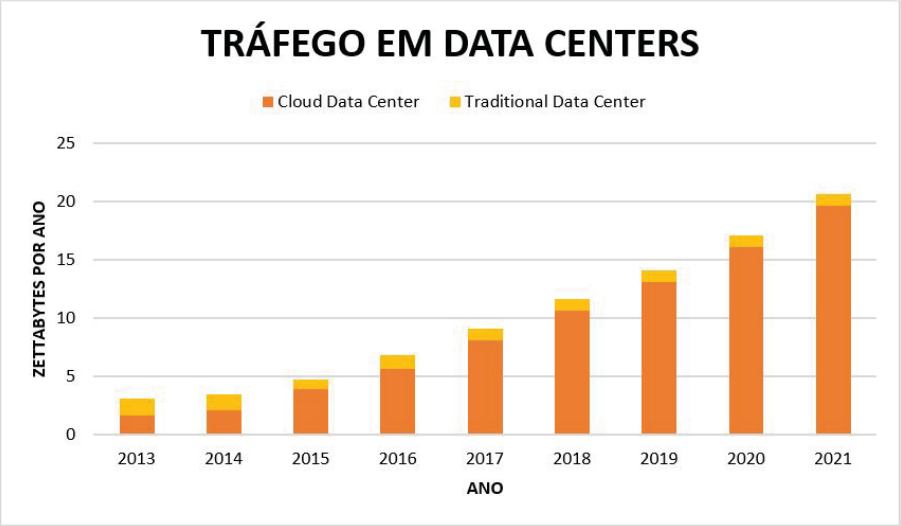



Figura 1.1: Tráfego em Data Centers estimado pela Cisco
(Cisco, 2018)

Como esperado, a previsão de 2018 (Cisco, 2018) superou a anterior de 2014 (Cisco, 2014), isso se deve ao fato de a *Cloud Computing* estar sendo implantada no cotidiano das organizações mais rápido do que se esperava. Na figura 1.1 pode-se observar o tráfego de dados em *Data Centers*, incluindo 2021.

Além disso, outras informações são notáveis, como o detalhamento das formas de tráfego de dados que acontecem nas *Cloud Data Centers*. Pode-se classificá-las de três maneiras:

- Tráfego de *Data Center* para Usuário: Este tráfego é gerado quando usuários utilizam serviços de E-mail e *Video-on-demand* (VOD).

- Tráfego de *Data Center* para *Data Center*: O tráfego entre *data centers* é gerado por tarefas como replicação de arquivos (comuns no *Hadoop Distributed File System*), *Content Delivery Networks* (CDN) e enlaces entre nuvens que enviam e recebem informações sobre o armazenamento e processamento de dados em cada *Data Center*, entre outros.

- Tráfego Interno no *Data Center*: Também conhecido como Tráfego Leste-Oeste, consiste em dados de armazenamento, processamento (principalmente os massivos de *Big Data*), desenvolvimento, entres outras.

Pode-se observar na figura 1.2 a quantia de tráfego em cada uma das categorias nos últimos seis anos em *Zettabytes*. Com essas informações conclui-se que o tráfego gerado pelo processamento de dados dentro dos *Data Centers* tem sido o principal motivo deste aumento gigantesco.

O tráfego que permanece nos *data centers* diminuiu ligeiramente até 2021, de 75,4% para 71,5%. Entretanto, estes dados não incluem as métricas de tráfegos locais dos racks (intra-racks). Caso fossem incluídas, a previsão é que o tráfego residente nos *data centers* atingiria mais de 90%. A *Big Data* possui uma porcentagem extremamente significativa no tráfego dentro dos *data centers*. Embora grande parte dos processamentos desta área seja intra-rack, ainda assim, o tráfego de um rack para outro foi o suficiente para ser responsável por pelo menos 20% de todo o tráfego global entre *data centers* em 2021 (Cisco, 2018).

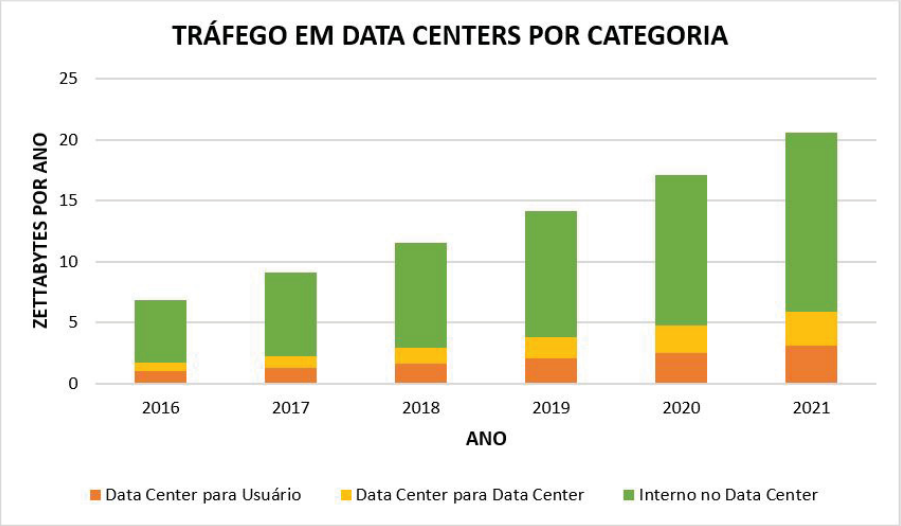



Figura 1.2: Tráfego em Data Centers por Categoria estimado pela Cisco (Cisco, 2018)

No geral, o tráfego leste-oeste (tráfego interno nos *data centers* e entre os *data centers*) representou 85,10% do total em 2021, e o tráfego norte-sul (tráfego saindo dos data centers para a Internet ou WAN) foi de apenas 14,90% do tráfego associado a *data centers*. Pode-se observar estas informações detalhadamente na figura 1.3, que demonstram o crescimento anual dos tráfegos em *data centers* até 2021 (Cisco, 2018).




(a) Taxa de Crescimento Anual Composta

(b) Tráfego Estimado em 2021

Figura 1.3: Previsões de Tráfegos para 2021 de acordo com a Cisco (Cisco, 2018)

Consequentemente este enorme tráfego de dados gera um dos principais problemas nos *data centers* atuais, o consumo de energia, que chega facilmente na casa de bilhões de kWh consumidos por ano (Raizada et al., 2020). Este é o caso dos *Data Centers* do Estados Unidos que em 2021 consumiram aproximadamente 73 bilhões de kWh (Shehabi et al., 2016). Um relatório da EPA aponta que de toda a energia consumida no mundo em 2010, cerca de 1,3% foram em *Data Centers* (Koomey et al., 2011). Outra pesquisa ainda indica que esse consumo chegue a 3000 bilhões de kWh em 2030 (Raizada et al., 2020).

De acordo com (Belady, 2007) os custos energéticos dos *Data Centers* seriam maiores que o valor gasto com as infraestruturas (*hardware*) em 2008, e em 2014 este valor já seria responsável por mais de 75% de todo o valor investido, que envolve gastos com energia, infraestrutura, funcionários e manutenção.

Em média 10% a 50% do consumo de energia dos *Data Centers* são por DCNs - *Data Center Networks* (Abts et al., 2010), e este número tende a tornar-se maior quando os recursos não são utilizados totalmente por uma execução. De toda a energia utilizada nos *clusters*, cerca de 30% são consumidas por *switches* (Kliazovich et al., 2012).

Conforme relatado em (Mahadevan et al., 2009) é possível diminuir este consumo de energia, uma vez que os *links* de interconexão que são responsáveis por 65% do consumo de energia total dos *switches*, não cessam o consumo total mesmo quando não estão sendo utilizados pelo *data center* (Christensen et al., 2010).

*Clusters* Hadoop possuem comunicações complexas que utilizam *links* de interconexão com frequência, principalmente nas fases *Shuffle* e *Reduce* de uma aplicação *MapReduce*, em que todos os nodos do aglomerado se comunicam entre si (Dean e Ghemawat, 2004).

O padrão Energy Efficient Ethernet (EEE) visa reduzir o consumo de energia das conexões Ethernet, definindo um modo de hibernação de link conhecido como *Low Power Idle*. Inicialmente o EEE foi desenvolvido para pequenos ambientes, como pequenas empresas e escritórios domésticos, e posteriormente para aplicações de *big data* de *data centers*, incluindo streaming de vídeo e *data science*. Como o EEE pode causar sobrecarga de rede e perda de desempenho, muitos fornecedores de equipamentos aconselham seus clientes a desativá-lo em uso de produção. No entanto, estudos recentes apontam que o EEE pode ser habilitado em ambientes de produção *MapReduce* se configurado corretamente (Silva e Carpenter, 2015) (Silva e Carpenter, 2017a).

O estudo mais importante que demonstra a eficácia do EEE em *clusters* Hadoop pode ser encontrado em E-EON - *Energy-Efficient and Optimized Networks for Hadoop* (Silva, 2018). Recentemente, foi demonstrado como obter alto desempenho em *clusters* Hadoop com o EEE habilitado para links de 1GbE e 10GbE. Nesta pesquisa é apresentado um conjunto de técnicas que podem ser utilizadas para reduzir o consumo de energia da rede e diminuir ainda mais a latência, enquanto melhora a taxa de transferência do cluster Hadoop. Além disso, essas técnicas não são exclusivas do Hadoop e pode-se obter benefícios semelhantes se aplicadas a qualquer outra carga de trabalho que seja semelhante em funcionamento ao *MapReduce*.

## 1.1 OBJETIVOS E CONTRIBUIÇÕES

Conforme a demanda de energia por *Data Centers* Hadoop aumentam torna-se necessária a otimização e redução do consumo. O principal objetivo da nossa pesquisa é avaliar o consumo de energia da atual versão do Hadoop *MapReduce* (3.x) através da implementação do EEE nas conexões, para isso, os seguintes objetivos específicos foram considerados:

1. Analisar de forma geral como a versão atual do Apache Hadoop (3.x) se comporta quando o EEE está habilitado para conexões de 1GbE e 10GbE no *cluster MapReduce*;

2. Realizar um estudo experimental para descobrir o potencial de economia de energia nos clusters Hadoop *MapReduce* se o EEE for implementado para conexões 25GbE, 40GbE, 100GbE e 400GbE.

Ainda como objetivos secundários, pretendemos responder algumas questões que permanecem sem respostas exatas até o presente momento:

- Como estas configurações de fato afetam o funcionamento de uma execução completa de um processamento no Hadoop *MapReduce*?

- Estas configurações afetam a versão 3.x da mesma forma que a versão 2.x e 1.x, ou há diferenças?

## 1.2 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está dividida em 6 capítulos, incluindo a Introdução como o primeiro Capítulo. No Capítulo 2 está a Fundamentação Teórica. Neste capítulo são explorados os conceitos abordados nesta pesquisa, como *Ethernet Energy Efficient*, *Packet Coalescing*, *Random Early Detection*, *Controlled Delay*, *Explicit Congestion Notification*, *MapReduce* e Hadoop.

O Capítulo 3 contém a Metodologia, em que é apresentada de forma completa o NetSLS, uma extensão do *YARN Scheduler Load Simulator*, que nos permite realizar simulações de *clusters* Hadoop, observar o consumo de recursos em tempo real através de gráficos, além de uma visão ampla da execução com rastreamentos em *logs*, e assim calcular a economia de energia e desempenho. No mesmo capítulo é definida as configurações das tecnologias *Ethernet Energy Efficient*, *Packet Coalescing*, *Random Early Detection*, *Controlled Delay*, *Explicit Congestion Notification*, as cargas de trabalhos *small tasks* e *batch*, e um ciclo de testes.

Em seguida, no Capítulo 4, é apresentada uma visão geral de como a versão atual do Apache Hadoop (3.x) se comporta quando o EEE está habilitado para links de 1GbE e 10GbE no cluster, além de uma demonstração do potencial de economia de energia em *clusters Hadoop MapReduce* se o EEE estiver habilitado para links de 25GbE, 40GbE, 100GbE e 400GbE.

No Capítulo 5 são tratados os problemas das conexões acima de 40GbE, as quais possuem uma grande perda de desempenho quando o EEE está habilitado. Para isso, foram combinados *Ethernet Energy Efficient* com *Packet Coalescing*, *Random Early Detection*, *Controlled Delay* e *Explicit Congestion Notification*, chegando assim à uma ótima economia de energia com perda de desempenho relativamente baixa. Ainda no Capítulo 5, comparamos nosso trabalho com o E-EON - uma série de configurações para se obter alto desempenho em *clusters* Hadoop com o EEE habilitado para links de 1 GbE e 10 GbE. Por fim, o Capítulo 6 apresenta as Considerações Finais e os Trabalhos Futuros desta pesquisa.

# 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo encontram-se os conceitos e fundamentos teóricos das tecnologias que foram utilizadas nesta dissertação, como o *Ethernet Energy Efficient*, *Packet Coalescing*, *Random Early Detection*, *Controlled Delay*, *Explicit Congestion Notification*, *MapReduce* e Hadoop. Além disso, é exposta a necessidade de estudos adicionais devido à constante evolução da área e ferramentas contidas nela.

## 2.1 ENERGY EFFICIENT ETHERNET

O *Energy Efficient Ethernet* também conhecido como IEEE802.3az, ou ainda *Green Ethernet*, é um protocolo de rede aprovado em 2010 pelo Instituto de Engenheiros Eletricistas e Eletrônicos (*Institute of Electrical and Electronics Engineers* - IEEE). Como a Ethernet é a tecnologia dominante para LANs com fio, espera-se que os mecanismos de economia de energia do EEE tragam economias de energia consideráveis (De La Oliva et al., 2013). O EEE já foi implantado, mas, os fornecedores de sistemas e máquinas aconselham seus clientes a desabilitá-lo em seu uso diário (Dell, 2020); (Ethernet, 2011), uma vez que seu impacto é mal compreendido em aplicações reais, não deixando claro o benefício em relação à economia de energia e desempenho.

### 2.1.1 Deep Sleep Mode

O EEE802.3az no modo *Deep Sleep* funciona da seguinte maneira: Quando uma estação transmissora de um *link* com esse protocolo detecta uma baixa utilização do *link*, ela pode solicitar que o transmissor PHY local entre no modo *Low Power Idle* (LPI) e envie símbolos apropriados pelo *link*. Ao receber e decodificar esses símbolos, o receptor do parceiro de *link* pode entrar no modo LPI, gerando assim economia de energia. Os caminhos de transmissão e recepção podem entrar e sair dos estados de baixa energia de forma independente. A energia é conservada desativando os blocos funcionais correspondentes do caminho individual (IEEE, 2010).

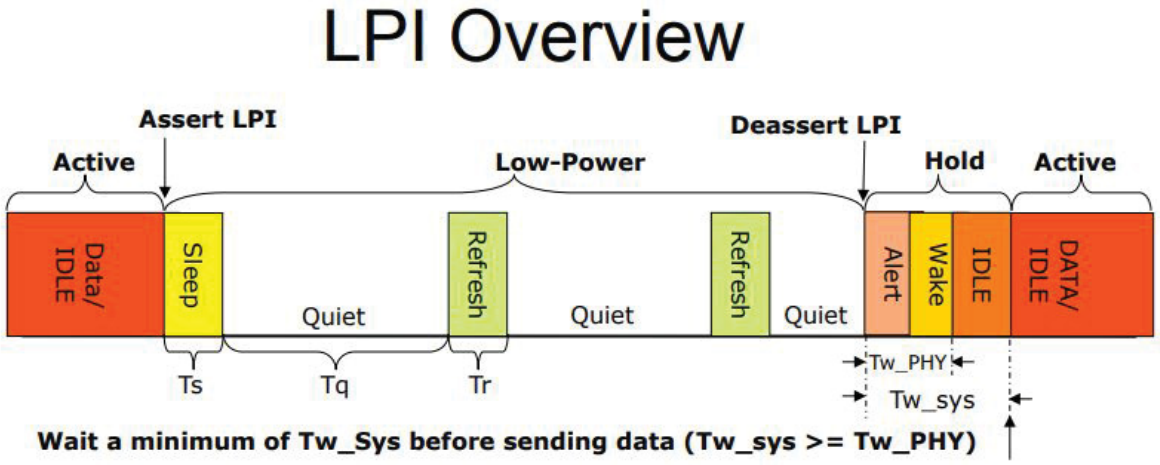



Figura 2.1: Overview of EEE LPI Operation
(Bennett e Diab, 2012)

Quando o início da codificação '*Assert LPI*' no xMII (*Media Independent Interface*) é detectado, os sinais PHY (camada física que provê uma interface entre a subcamada MAC e o canal do rádio) hibernam para seu parceiro de *link*, para indicar que o transmissor local está entrando no modo LPI como observamos na figura 2.1.

A capacidade EEE na maioria dos PHYs (por exemplo, 100BASE-TX, 10GBASE-T, 1000BASE-KX, 10GBASEKR e 10GBASE-KX4) requer que o transmissor PHY local fique no modo *Quiet* após o modo *Low-Power* ser sinalizado. No modo 1000BASE-T LPI, o transmissor PHY local fica *Quiet* somente depois que o PHY local sinaliza para o modo *Sleep* e recebe um sinal de *Sleep* do PHY remoto. Se o PHY remoto optar por não sinalizar o modo LPI, nenhum dos PHY pode entrar no modo de baixa energia; no entanto, as solicitações LPI são passadas de uma extremidade do *link* para a outra independentemente, e a economia de energia do sistema pode ser alcançada mesmo se o *link* PHY não entrar em um modo de baixa energia (IEEE, 2010).

A função de transmissão do PHY local é habilitada periodicamente para transmitir sinais de atualização que são usados pelo parceiro de *link* para atualizar filtros adaptativos e circuitos de temporização a fim de manter a integridade do *link*. Este ciclo de atualização silenciosa continua até a recepção da codificação normal entre quadros no xMII. A função de transmissão no PHY comunica isso ao parceiro de *link*, enviando um sinal de despertar por um período de tempo predefinido. O PHY então entra no estado operacional normal novamente.

De acordo com (Reviriego et al., 2010), a eficiência de energia do EEE é impactada por dois fatores: o consumo de energia ocioso - cerca de 10%; e as despesas gerais de *Wake*/*Sleep*. O consumo de energia ocioso possuí um custo fixo, que não é afetado pela entrada ou saída do modo LPI, e reflete apenas nas despesas gerais de *Wake*/*Sleep*. Quando a carga é moderada à alta, a sobrecarga de energia da transição para dentro e fora do modo LPI podem ser amortizadas em vários pacotes. Entretanto, quando a carga é moderada à baixa, pode ser necessário despertar e hibernar o *link* de conexão para transmitir um único *frame*, resultando em penalidades de gastos altos de energia e latência em termos relativos.

Tabela 2.1: EEE 150-byte frame Efficiency

| | Min. *Tw* (µs) | Min. *Ts* (µs) | Min. *Tframe* (µs) | Efficiency (%) |
|---|---|---|---|---|
| 100Base-TX | 30,5 | 200 | 12 | 4,9 |
| 1000Base-T | 16,5 | 182 | 1,2 | 0,6 |
| 10GBase-T | 4,48 | 2,88 | 0,12 | 1,6 |

Pode-se observar na tabela 2.1, em alguns momentos é necessário sair do Modo LPI para transmitir um único ou poucos *frames*, o que resulta em um acumulo de latência na rede como mencionado em (Reviriego et al., 2010), além da perda de eficiência, que em alguns casos pode chegar a apenas 0,6% comparado a conexões comuns *Ethernet*.

Tabela 2.2: EEE 1500-byte frame Efficiency

| | Min. *Tw* (µs) | Min. *Ts* (µs) | Min. *Tframe* (µs) | Efficiency (%) |
|---|---|---|---|---|
| 100Base-TX | 30,5 | 200 | 120 | 34,2 |
| 1000Base-T | 16,5 | 182 | 12 | 5,7 |
| 10GBase-T | 4,48 | 2,88 | 1,2 | 14 |

Como esperado, os índices de alta economia de energia com EEE são atingidos quando os pacotes que devem ser transmitidos são maiores e em grande quantidade como demonstrado na tabela 2.2, em que podemos atingir até 34,2% de eficiência energética em relação ao padrão *Ethernet* comum.

Felizmente, o problema descrito na tabela 2.1 pode ser corrigido adicionando *Active Queue Management* a rede, como demonstra o estudo do impacto da utilização de EEE com *Packet Coalescing* e *Active Queue Management* no Hadoop 1.x (Silva, 2018). Entretanto, se faz necessário estudos destas mesmas configurações na linha do Hadoop 3.x que executa o *MapReduce* através do YARN diferentemente da versão 1.x, e utiliza *Erasure Coding* em seu armazenamento diferentemente das versões 1.x e 2.x. Estas informações estão detalhadas de uma forma abrangente no subtópico do Apache Hadoop neste mesmo capítulo.

### 2.1.2 Fast Wake Mode

*Fast wake* refere-se ao modo pelo qual o transmissor continua a transmitir sinais durante o estado LPI para que o receptor possa retomar a operação com um tempo *wake* mais curto (como exposto na figura 2.2). Para o processo de transmissão, além das codificações do LPI, não há diferença entre o *Deep Sleep Mode* e o *Fast Wake Mode* (IEEE, 2014). O suporte ao modo *Fast Wake* é obrigatório para PHYs com velocidade operacional de 40 Gb/s ou superior que implementam o EEE. Utilizar este modo resulta em um consumo energia do link em cerca de 60% (Georgakoudis et al., 2019); (IEEE, 2014), um valor elevado se comparado aos 10% do modo *Deep Sleep*.

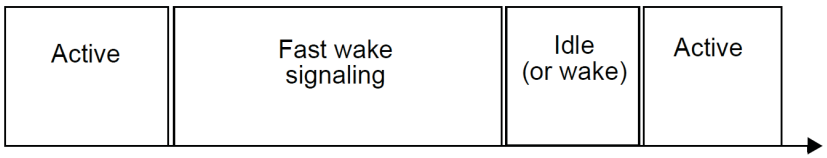



Figura 2.2: Overview of EEE fast wake operation
(IEEE, 2014)

## 2.2 PACKET COALESCING

De uma forma simples, o *Packet Coalescing* é o agrupamento de pacotes como forma de limitar o número de interrupções de recebimento e, como resultado, diminuir a quantidade de processamento necessária. Isso pode aumentar a latência da rede, mas em contrapartida, aumenta as chances de que a rede consuma menos energia, principalmente quando aliada ao EEE.

O *Packet Coalescing*, do português Coalescência de Pacotes, é um método que inclui o recebimento de múltiplos pacotes de IP (*Internet Protocol*). Cada um destes múltiplos pacotes de IP possuí um cabeçalho, além de um segmento de TCP (*Transmission Control Protocol*) com cabeçalho, e um *payload*, em que vários pacotes pertencentes a um mesmo fluxo TCP/IP são transmitidos. O método também inclui a preparação de um pacote de IP com um único cabeçalho de IP e um único segmento de TCP com um único cabeçalho de TCP e uma única carga formada por uma combinação das cargas de segmento de TCP dos vários pacotes de IP. O método inclui ainda a geração de um sinal que causa o processamento de recepção do pacote de IP (Makineni et al., 2020).

Um estudo feito por (Mostowfi e Christensen, 2011) estima que uma economia de energia potencial de 3,5 TWh/ano pode ser obtida pela implantação do *Packet Coalescing* em todos os futuros *switches* domésticos ou de pequenas empresas nos Estados Unidos. Outro artigo publicado por (Silva e Carpenter, 2017a) mostra que quando utilizado em redes de 10GbE nos *clusters MapReduce*, o *Packet Coalescing* juntamente ao EEE proporciona uma economia de energética de 20% a 60%.

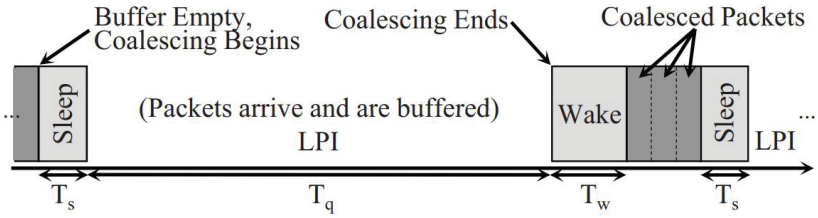



Figura 2.3: EEE with Packet Coalescing
(Mostowfi e Christensen, 2011)

Pode-se observar na figura 2.3 o resultado da junção do *Packet Coalescing* com EEE. Conforme mostrado nesta figura, quando todos os pacotes na fila de transmissão (ou *buffer*) são transmitidos e o *buffer* fica vazio, o link é colocado no modo LPI com uma transição para o modo *Sleep* que leva um tempo T*s*. Os pacotes que chegam depois não são transmitidos imediatamente, mas são armazenados dentro de um *buffer* de coalescência. Quando um tempo máximo passa da chegada do primeiro pacote ao *buffer* de coalescência, ou o número de pacotes armazenados no *buffer* atinge um número máximo predefinido, o link sai do modo LPI, o que leva um tempo T*w*, e então, todos os pacotes coalescidos são transmitidos em uma única rajada (Mostowfi e Christensen, 2011).

## 2.3 ACTIVE QUEUE MANAGEMENT

*Active Queue Management* é uma política implementada em roteadores e *switches* para descartar pacotes dentro de um *buffer* associado a um *Network Interface Controller* (NIC) antes que o *buffer* fique cheio, geralmente com o objetivo de reduzir o congestionamento da rede ou melhorar a latência de ponta a ponta (Adams, 2012). Esta tarefa é realizada pelo escalonador da rede, que para isso pode utilizar vários algoritmos como o *Random Early Detection* (RED) (Floyd e Jacobson, 1993) ou *Controlled Delay* (CoDel) (Nichols e Jacobson, 2012).

Devido à simplicidade do mecanismo de enfileiramento *First In First Out* (FIFO), as filas *drop-tail* que descartam os pacotes de entrada quando a fila está cheia são as mais amplamente utilizadas na Internet atualmente. Infelizmente, quando confrontado com congestionamento persistente, as filas *drop-tail* se enchem, resultando em atrasos maiores. Além disso, as filas *drop-tail* também podem descartar vários pacotes, agravando a instabilidade do sistema e a imparcialidade da largura de banda. O *Active Queue Management* (AQM) se propõe a substituir o gerenciamento de fila *drop-tail* para melhorar o desempenho da rede em termos de atraso, utilização do *link*, taxa de perda de pacotes e integridade do sistema. O AQM aprimora os roteadores para detectar e notificar os sistemas finais de congestionamento iminente com antecedência, permitindo que as fontes de tráfego reduzam suas taxas de transmissão antes que as filas de roteadores e comutadores congestionadas estourem (Chung e Claypool, 2003).

### 2.3.1 Random Early Detection

Podemos observar o esquema do *Random Early Detection* (RED) na figura 2.4. Este é um conhecido AQM leve que usa o tamanho médio da fila e o limite mínimo ($min_{th}$) e o limite máximo ($max_{th}$) para detectar o congestionamento iminente e determinar a probabilidade de notificação de congestionamento. Quando o tamanho médio da fila está entre ($min_{th}$) e ($max_{th}$), o RED descarta aleatoriamente (ou marca no modo ECN) os pacotes de entrada com uma probabilidade que aumenta linearmente de zero para a probabilidade máxima ($max_p$) conforme o tamanho médio da fila aumenta de ($min_{th}$) a ($max_{th}$). No entanto, quando o tamanho médio da

fila é maior que ($max_{th}$), o RED descarta todos os pacotes de entrada até que o tamanho médio da fila caia abaixo de ($max_{th}$) (Floyd e Jacobson, 1993); (Lochin e Talavera, 2011).

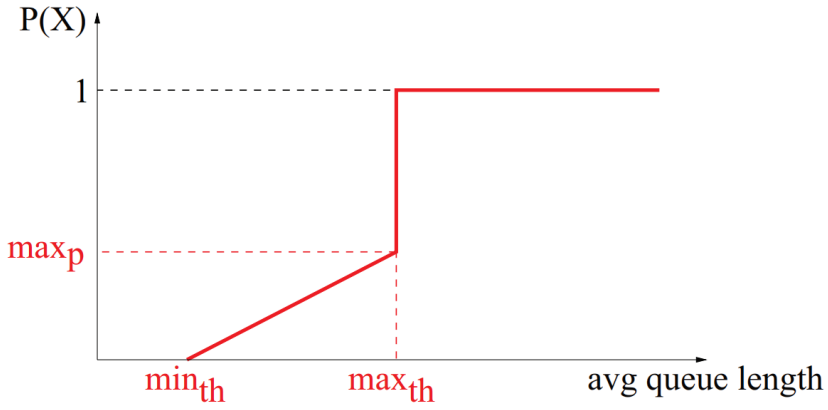



Figura 2.4: The Random Early Detection (RED) queue management scheme
(Lochin e Talavera, 2011)

Estudos realizados por (Floyd e Jacobson, 1993) e (Rosolen et al., 1999) mostram que o RED pode melhorar o rendimento e a distribuição dos recursos de rede, enquanto mantém um atraso médio de enfileiramento baixo. No entanto, esse benefício só é alcançado quando o RED está adequadamente configurado sob algumas cargas de tráfego, especificamente quando o comprimento médio da fila não oscila significativamente e permanece abaixo de $max_{th}$.

O artigo publicado por (Chung e Claypool, 2002) ainda revela que ao utilizar RED combinado com ECN adequadamente configurado, pode resultar em uma taxa de perda de pacotes e atraso de fila muito baixos, o que torna o RED muitas vezes superior ao gerenciamento de fila *drop-tail*, o que também foi confirmado por (Silva e Carpenter, 2017b) ao testar estas configurações em um *cluster* Hadoop.

### 2.3.2 Controlled Delay

*Controlled Delay* (CoDel) é um algoritmo de escalonamento para o escalonador de rede. Ele é projetado para superar o *bufferbloat* (alta latência em redes com comutação causada pelo excesso de carregamento de pacotes) em *hardware* de rede, como roteadores, definindo limites na experiência de pacotes de rede de atraso à medida que passam pelos *buffers* neste equipamento. Segundo (Nichols e Jacobson, 2012), CoDel difere dos demais AQMs por:

- É sem parâmetros - não possui parâmetros para os operadores, usuários ou implementadores ajustarem;
- Trata as filas de maneiras diferentes mantendo os atrasos baixos enquanto permite picos de tráfego;
- Controla o atraso e ao mesmo tempo ser insensível a atrasos de ida e volta, taxas de *link* e cargas de tráfego;
- Se adapta dinamicamente trocando as taxas de *link* sem nenhum impacto negativo na utilização da rede;
- É simples e eficiente - pode facilmente abranger todo o espectro de baixo custo, pontos de acesso baseados em Linux, roteadores domésticos, e até mesmo roteadores comerciais de silício.

O algoritmo do CoDel calcula cada salto de forma independente (Nichols e Jacobson, 2012). Primeiramente, inicia em um intervalo de 100 milissegundos. O atraso de enfileiramento por pacote é monitorado por meio do salto. À medida que cada pacote é retirado da fila para encaminhamento, o atraso de enfileiramento (quantidade de tempo que o pacote passou esperando na fila) é calculado. O menor atraso de fila para o intervalo é armazenado.

$$SequenciaDeIntervalos = (100, \frac{100}{\sqrt{2}}, \frac{100}{\sqrt{3}}, \frac{100}{\sqrt{4}}, \frac{100}{\sqrt{5}}, \frac{100}{\sqrt{6}}, \frac{100}{\sqrt{7}}, ...) \quad (2.1)$$

No momento em que o último pacote do intervalo é retirado da fila, se o menor atraso de enfileiramento para o intervalo for maior que 5 milissegundos, esse único pacote é descartado e o intervalo usado para o próximo grupo de pacotes é reduzido. Se o menor atraso de enfileiramento para o intervalo for inferior a 5 milissegundos, o pacote será encaminhado e o intervalo será redefinido para 100 milissegundos. Quando o intervalo é reduzido, isso é feito de acordo com a raiz quadrada inversa do número de intervalos sucessivos nos quais os pacotes foram descartados devido ao atraso excessivo de enfileiramento, conforme demonstrado na expressão 2.1.

## 2.4 EXPLICIT CONGESTION NOTIFICATION

*Explicit Congestion Notification* é uma extensão do protocolo TCP/IP definida em 2001. Com o ECN habilitado a conexão tem alguns aprimoramentos, como por exemplo, realizar notificações de congestionamento de ponta a ponta na rede sem que haja perda de pacotes. Além disso, as fontes podem ser informadas sobre o congestionamento de forma rápida e inequívoca, não havendo necessidade de esperar por um temporizador de retransmissão ou três confirmações duplicadas para inferir um pacote descartado. O ECN pode ser habilitado em qualquer rede LAN/WAN que tenham roteadores ou *switches* nas pontas que ofereçam suporte a esta tecnologia (Floyd e Ramakrishnan, 2000).

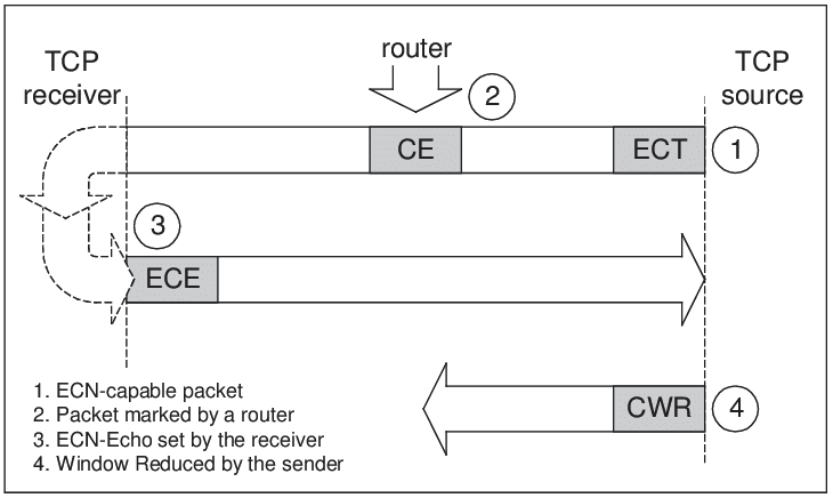



Figura 2.5: ECN in Action
(Małowidzki, 2003)

O ECN permite roteadores que usam AQMs (por exemplo, RED ou CoDel) marquem os pacotes em caso de congestionamento, em vez de descartá-los. Todo este processo pode ser observado na figura 2.5, em que dois *bits* no cabeçalho IP fornecem quatro marcas possíveis: No-ECN (00), *Congestion Experienced* (CE, 11) e dois pontos de código para *ECN-Capable*

*Transport* (ECT (0), 01; e ECT (1), 10) . Um remetente com capacidade para ECN define ECT (0) ou ECT (1), que pode ser alterado para CE por um roteador para sinalizar o congestionamento. ECN usa dois sinalizadores adicionais no cabeçalho TCP: *ECN-Echo* (ECE) é definido em todos os pacotes do receptor para o remetente para sinalizar a chegada de um pacote com marcação CE até que o remetente defina a janela de *Congestion Window Reduced* (CWR) para reconhecer a ECE. Esses sinalizadores também são usados para negociar o uso de ECN: um iniciador de conexão solicita o ECN configurando ECE e CWR no SYN inicial, e o respondente reconhece configurando ECE no SYN/ACK. Depois de concluir a negociação com êxito, os remetentes podem definir um ponto de código ECT em todos os pacotes subsequentes na conexão (Kühlewind et al., 2013).

Atualmente o ECN está implementado na maioria dos sistemas operacionais. No entanto, mesmo habilitado por padrão, geralmente está apenas no 'modo de servidor': O ECN será negociado se solicitado por um nodo remoto iniciando uma conexão, mas as conexões abertas pelo nodo não tentarão negociar o uso de ECN (Kühlewind et al., 2013).

## 2.5 MAPREDUCE

O *MapReduce* surgiu em fevereiro de 2003, quando dois funcionários do Google, Jeffrey Dean e Sanjay Ghemawat buscavam aperfeiçoar o sistema de busca do Google (Dean e Ghemawat, 2004). Considerado um modelo de programação altamente avançado para processar grandes conjuntos de dados estruturados e principalmente não estruturados, ele consiste em duas funções:

- *Map* (Mapeamento): A função *Map* recebe uma lista ordenada <chave, valor> gerando um conjunto intermediário de dados com o mesmo formato, lista ordenada <chave, valor>.
- *Reduce* (Redução): A função *Reduce* é executada para cada chave intermediaria gerada, combinando as chaves associadas.

```
1  map(String key, String value):
2        // key: nome do documento
3        // value: conteúdo do documento
4        for each word w in value:
5              EmitIntermediate(w, "1");
6  reduce(String key, Iterator values):
7        // key: uma palavra
8        // values: uma lista de contagens
9        int result = 0;
10       for each v in values:
11             result += ParseInt(v);
12       Emit(AsString(result));
```

Durante a execução do *MapReduce*, as funções *Map* e *Reduce* recebem e emitem dados no formato de Listas ordenadas <chave, valor>. Em outras palavras, o algoritmo *Map* é usado para encontrar algo ou resolver pequenos problemas e o *Reduce* para reunir os resultados, gerando assim uma solução para o problema maior. Para ajudar a ilustrar o modelo de programação *MapReduce*, considere o problema de contar quantas vezes cada palavra aparece em um mesmo documento. Pode-se observar acima os pseudocódigos das funções *Map* e *Reduce* de um contador de palavras, em que na fase de mapeamento (*map*) a variável *key* recebe o nome do documento, e a value seu conteúdo, então para cada palavra é emitido no arquivo virtual uma lista <palavra,1>, em seguida, na fase de redução (*reduce*) o *key* recebe novamente uma palavra, enquanto o *iterador*

*values* recebe a todas as listas com a palavra, e então, a variável *result* soma todas, chegando assim na quantidade de vezes em que a palavra aparece no arquivo de entrada.

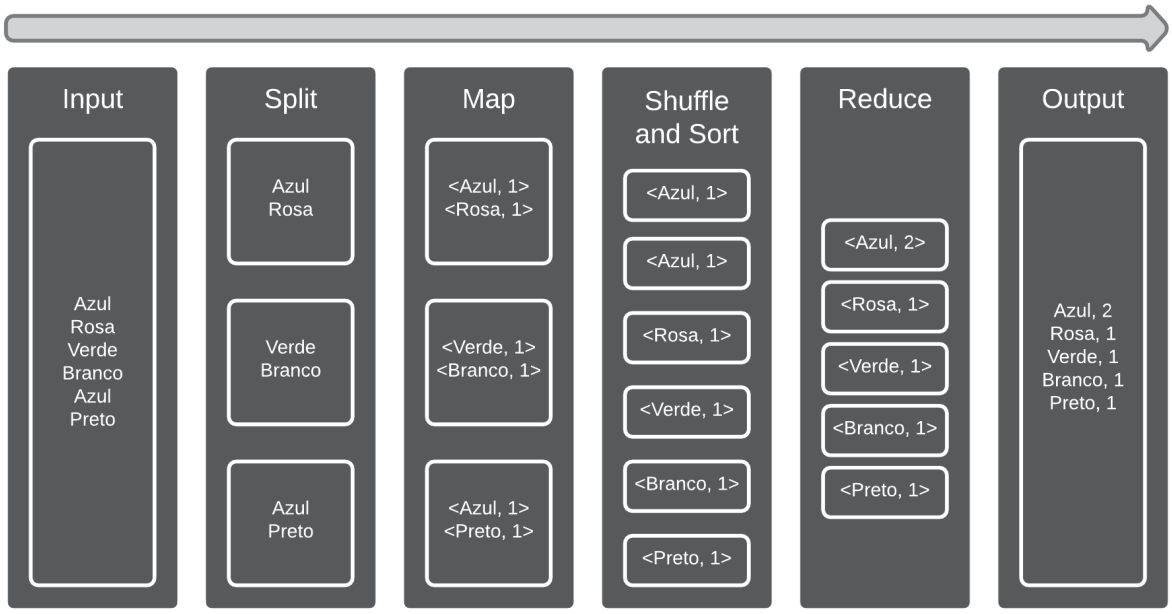



Figura 2.6: Etapas de uma Aplicação MapReduce WordCount
Adaptada de (White, 2015)

Para ilustrar observe a figura 2.6. Ao iniciar uma execução *MapReduce* no *cluster*, o arquivo de entrada é repartido em blocos *(split phase)*, estes blocos são enviados aos nodos do *cluster* responsáveis por realizar um mapeamento dos dados *(map phase)* e emitir várias listas, cada uma contendo uma palavra e uma chave *<word, key>;* nas fases de embaralhamento e ordenação *(shuffle and sort phase)* as listas são ordenadas, e, em seguida, transferidas para a fase de redução *(reduce phase)* que efetua as somas dos valores das chaves e escreve os resultados no arquivo de saída *(output phase)*.

Embora o pseudocódigo do *MapReduce* permaneça o mesmo desde sua criação, a implementação do *Hadoop MapReduce* continua a ser atualizada constantemente (White, 2009); (White, 2010); (White, 2012); (White, 2015).

## 2.6 APACHE HADOOP

Desenvolvido por Doug Cutting e Mike Cafarella e mantido atualmente pela Apache Software Foundation, o ecossistema Hadoop é uma plataforma de código-fonte aberto em Java para armazenamento e processamento de dados em larga escala. A configuração dos *clusters* Hadoop é extremamente fácil, enquanto integridade dos dados, disponibilidade dos nodos, escalabilidade da aplicação e recuperação de falhas são tratadas automaticamente de forma transparente ao usuário, além de possuir um processamento superior as demais tecnologias da área (Goldman et al., 2012).

A comunidade de *BigData* vem utilizando a Computação de Alto Desempenho (*High-performance Computing* - HPC) e Computação em Grade (*Grid Computing* - GC) por anos para o processamento de dados em larga escala. O problema que surge ao utilizar uma dessas abordagens é de os nodos acessarem a informação através de uma Rede de Área de Armazenamento (*Storage Area Network* - SAN), o que funciona bem para problemas que necessitam de computação

extrema, mas, se torna uma fraqueza para aqueles que necessitam acessar grandes volumes de dados constantemente (White, 2015).

Diferentemente do HPC e GC, o Hadoop leva a computação até os dados, ou seja, ele co-localiza os arquivos com os nodos de computação, tornando o acesso mais rápido por ser local (Gray, 2004). Este recurso do Hadoop é conhecido como '*Data Locality*', sendo o coração do processamento de dados neste Ecossistema e a razão para o seu desempenho superior. Reconhecendo que a largura de banda é o recurso mais precioso em um ambiente de *Data Center* (é fácil saturar os *links* de uma rede com a cópia constante de dados), o Hadoop tenta ao máximo preservar a saúde da mesma modelando explicitamente a topologia da rede (White, 2015).

### 2.6.1 Módulos do Ecossistema Hadoop

Conforme a documentação do Apache Hadoop (Hadoop, 2022), a atual versão do ecossistema 3.x é composta por quatro módulos principais, cada um com sua finalidade específica:

- *Hadoop Common*: Módulo que contém as bibliotecas e arquivos necessários para o funcionamento total das estruturas do Hadoop.
- *Hadoop Distributed File System* (HDFS™): Módulo que age como um sistema de arquivos distribuídos, controlando e armazenando os dados nos nodos, além de permitir largura de banda gigantesca em todo o *cluster*. É altamente tolerante a falhas e pode se recuperar automaticamente.
- *Hadoop YARN*: *Framework* que funciona como um *middleware*, gerenciando os recursos disponíveis e fazendo agendamentos dos recursos que serão usados no processamento de dados.
- *Hadoop MapReduce*: Contém um sistema baseado em YARN. Quando este módulo está operando, todo o poder computacional do *cluster* é direcionado para as funções *Map* e *Reduce*, permitindo assim o processamento paralelo de grandes conjuntos de dados.

### 2.6.2 Processos do Hadoop

Uma aplicação típica do Hadoop executa cinco processos principais para gerenciar a utilização dos recursos do *Cluster*.

***NameNode*:** Na versão 1.x do Hadoop, o *NameNode* era um ponto único de falha (SPOF) em um *cluster HDFS*. Cada *cluster* tinha um único *NameNode* e, se essa máquina ou processo ficasse indisponível, o *cluster* como um todo ficaria indisponível até que o *NameNode* fosse reiniciado ou criado em uma máquina nova. Isso afetava a disponibilidade total do cluster HDFS de duas maneiras principais:

- No caso de um evento não planejado, como uma falha de nodo, o *cluster* ficava indisponível até que um administrador reiniciasse o *NameNode*.
- Eventos de manutenção planejada, como atualizações de *software* ou *hardware* na máquina que continha o *NameNode*, resultavam em janelas de tempo de inatividade do *cluster*.

O recurso *HDFS High Availability* aborda os problemas acima fornecendo a opção de executar dois (e a partir da versão 3.x mais de dois) *NameNodes* redundantes no mesmo *cluster* em uma configuração ativa/passiva com um *hot standby*. Isso permite um *failover* (método de proteção de sistemas contra falhas, no qual o equipamento de reserva assume automaticamente quando o sistema principal falha) rápido para um novo *NameNode* no caso de falha de uma máquina ou um *failover* iniciado pelo administrador para fins de manutenção e/ou atualização planejada.

***DataNode*:** Os *DataNodes* são responsáveis por atender as solicitações de leitura e escrita do sistema de arquivos. Os *DataNodes* também executam a criação, exclusão e replicação de blocos mediante instruções do *NameNode*.

***ResourceManager*:** O *ResourceManager* (RM) é responsável por rastrear os recursos disponíveis dentro do *cluster* e agendar aplicações (por exemplo, trabalhos *MapReduce*). Antes do Hadoop 2.4, o *ResourceManager* era o único ponto de falha em um *cluster YARN*. O recurso *High Availability* adiciona redundância na forma de um par *Active/Standby ResourceManager*, substituindo assim o RM principal por um secundário caso haja falha.

***NodeManager*:** O *NodeManager* é responsável por iniciar e gerenciar contêineres em um nodo. Os contêineres executam tarefas conforme especificado pelo AppMaster. O *NodeManager* executa serviços para determinar a integridade do nodo em que está sendo executado. Os serviços realizam verificações no disco, bem como quaisquer testes especificados pelo usuário. Se qualquer verificação de integridade falhar, o *NodeManager* marcará o nodo como não íntegro e comunicará isso ao *ResourceManager*, que interromperá a atribuição de contêineres ao nodo.

***SecondaryNameNode*:** O *SecondaryNameNode* mescla o *fsimage* e os arquivos de *log* de edições periodicamente e mantém o tamanho do *log* de edições dentro de um limite. Geralmente é executado em uma máquina diferente do *NameNode* principal, pois seus requisitos de memória são iguais.

O início do processo de *checkpoint* no *NameNode* secundário é controlado por dois parâmetros de configuração:

- *dfs.namenode.checkpoint.period*, definido como 1 hora por padrão, especifica o atraso máximo entre dois pontos de verificação consecutivos;
- *dfs.namenode.checkpoint.txns*, definido como 1 milhão por padrão, define o número de transações sem *checkpoint* no *NameNode* que forçarão um *checkpoint* urgente, mesmo que o período do *checkpoint* não tenha sido atingido.

O *SecondaryNameNode* armazena o último ponto de verificação em um diretório que é estruturado da mesma forma que o diretório do *NameNode* principal, para que a imagem de verificação esteja sempre pronta para ser lida pelo *NameNode* principal, se necessário.

### 2.6.3 Aplicações YARN

Apache YARN (*Yeat Another Resource Negotiator*) é o sistema de gestão dos recursos dos *clusters* Hadoop, introduzido no ecossistema a partir da versão 2.x para melhorar o desempenho do *MapReduce*, mas também pode ser utilizado em outros paradigmas de computação paralela.

O YARN fornece APIs para fazer requisições de recursos do *cluster*, mas estas APIs não são utilizadas diretamente no código-fonte das aplicações. Em vez disso, os usuários implementam algoritmos para computação distribuída em linguagem de programação de alto nível a partir de classes abstratas do YARN, que por sua vez, oculta os detalhes de gerenciamento

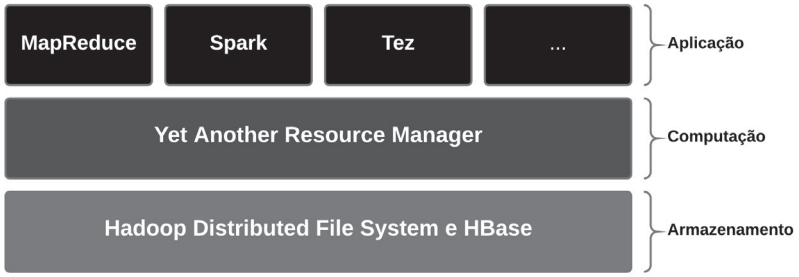



Figura 2.7: Aplicações YARN
Adaptada de (White, 2015)

de recursos dos programadores. Na figura 2.7 pode-se observar os *frameworks* de computação distribuída do ecossistema Hadoop como *MapReduce*, Spark, Tez, e assim por diante que são executados como Aplicativos YARN na camada de computação utilizando como sistema de armazenamento o HDFS ou HBase (White, 2015).

Ainda há outra camada de aplicativos acima que podem ser executados nos *frameworks MapReduce*, Tez, Spark ou nos três, como por exemplo o Pig, Hive e Crunch, mas não interagem diretamente com o YARN.

O YARN fornece seus principais serviços por meio de dois tipos de *daemon* de longa duração: um *Resource Manager* (um por *cluster*) para gerenciar o uso de recursos no *cluster*, e o *Node Managers* em execução em todos os nós do *cluster* para iniciar e monitorar *Containers*. Um *Container* executa um processo específico da aplicação com um conjunto restrito de recursos (memória, CPU e assim por diante). Dependendo de como o YARN está configurado, um *Container* pode ser um processador Unix ou um cgroup Linux.

### 2.6.4 Hadoop 1.x, 2.x e 3.x

A seguir apresentamos as principais diferenças entre as versões do Hadoop divididas em dois subsubtópicos para melhor entendimento de novos módulos e as alterações entre os mesmos. Estas inovações impactam diretamente em sua performance e gasto energético, e por isso, foram levadas em conta na definição da metodologia.

#### *2.6.4.1 Diferenças entre Hadoop 1.x e 2.x*

Ao longo das atualizações, diversas modificações foram realizadas na estrutura do Ecossistema Hadoop, entre as séries 1.x e 2.x a mais notável é a introdução do YARN entre as camadas do HDFS e *MapReduce*, resultando em uma nova forma (*YARN MapReduce*) de implementar e executar MRJs (White, 2015).

Tabela 2.3: A comparison of MapReduce 1 and YARN components

| **Hadoop 1.x** | **Hadoop 2.x** |
|---|---|
| **MapReduce 1** | **YARN (MapReduce V2 - NextGen)** |
| Jobtracker | Resource Manager, Application Master, Timeline Server |
| Tasktracker | Node Manager |
| Slot | Container |

Na segunda versão do Hadoop, o módulo *Yet Another Resource Negotiator* foi agregado e mudou a arquitetura do arcabouço significativamente como demonstrado na tabela 2.3. O

YARN é responsável por gerenciar recursos do *cluster* e agendamento de trabalhos. Nas versões anteriores do Hadoop, essa funcionalidade era integrada ao módulo *MapReduce*, onde era realizada pelo componente *JobTracker*. O *JobTracker* era responsável pelo agendamento, gerenciamento de recursos, monitoramento e reexecução de tarefas com falhas, relatório do status dos trabalhos aos usuários, registro de logs de auditoria, agregação de estatísticas, autenticação de usuários e várias outras funções. A grande quantidade de responsabilidades gerava limitações de escalabilidade. A ideia fundamental do YARN é dividir as duas principais funcionalidades do *JobTracker*, gerenciamento de recursos e agendamento/monitoramento de tarefas para ter um *ResourceManager* global e um *ApplicationMaster* específico de cada aplicação. Ao separar as funções de gerenciamento de recursos do modelo de programação, o YARN delega muitas tarefas relacionadas ao agendamento para componentes por trabalho e se afasta completamente do particionamento estático de recursos para *Mappers* e *Reducers*, considerando os recursos do *cluster* como contínuos, o que traz melhorias significativas para a utilização do mesmo (Glushkova et al., 2019).

Como analisado por (Glushkova et al., 2019), para o Hadoop melhorar o desempenho geral, bem como a utilidade e compatibilidade com outros aplicativos de processamento de dados distribuídos, alguns requisitos foram adicionados, como alta utilização dos recursos do *cluster*, alto nível de confiabilidade e disponibilidade, suporte para diversidade de modelos de programação e modelos de recursos flexíveis. Assim, a arquitetura da segunda versão do Hadoop passou por melhorias significativas, introduzindo o YARN, um módulo de gerenciamento de recursos separado que altera visivelmente a arquitetura geral, em que desacopla o modelo de programação da infraestrutura de gerenciamento de recursos e delega muitas funções de agendamento para os componentes da aplicação. Os recursos do *cluster* agora estão sendo considerados contínuos, portanto, não há particionamento estático de recursos (ou seja, uma divisão entre *slots* de *mappers* e *reducers*). Então, as tarefas de mapeamento e redução competem agora pelos mesmos recursos. Estas diferenças causam em média de 11% a 13.5% de redução no custo de MRJs entre as versões 1.x e 2.x do Hadoop.

#### *2.6.4.2 Diferenças entre Hadoop 2.x e 3.x*

O Apache Hadoop 3.x incorpora uma série de melhorias significativas em relação à sua linha de lançamentos anteriores Hadoop 2.x. Uma das principais modificações é o *Erasure Coding* (EC), um método para armazenar dados de forma duradoura com economia de espaço significativa em comparação a forma antiga (replicação). O EC fornece o mesmo nível de tolerância a falhas com um menor espaço de armazenamento consumido, em suas configurações típicas a sobrecarga de armazenamento não é superior a 50%. A integração do EC ao HDFS pode melhorar a eficiência do armazenamento, ao mesmo tempo em que oferece durabilidade de dados semelhante às implantações HDFS tradicionais baseadas em replicação. Por exemplo, um arquivo replicado na versão 2.x do Hadoop com 6 blocos consumiria (6x3=18) 18 blocos de espaço em disco, mas, na versão 3.x com a implantação EC (6 de dados, 3 de paridade), ele consumirá apenas 9 blocos de espaço em disco (Hadoop, 2017).

Como pode-se observar na tabela 2.4 as principais mudanças são a implementação do *Erasure Coding* que reduz a sobrecarga de armazenamento em 150% em relação a replicação, aumento da escalabilidade para mais de 10.000 nodos e o balanceamento de dados que agora é feito de forma interna pelos nodos.

Estudos comparativos realizados por (Masur e Mcintosh, 2017) mostram que o Hadoop 3.x leva vantagem em média de 40% em operações *TeraSort* sobre a versão anterior, e esta porcentagem tende a aumentar para mais de 50% quando envolve operações de leitura e escrita,

Tabela 2.4: Comparação entre Hadoop 2.x e Hadoop 3.x

| **Atributos** | **Hadoop 2.x** | **Hadoop 3.x** |
|---|---|---|
| Lida com Falhas | Através da Replicação | Através do *Erasure Coding* |
| Sobrecarga de Armazenamento | 200% no HDFS | 50% no HDFS |
| Sistema de Arquivos | DFS, FTP e Amazon S3 | Anteriores + MADLFS |
| Intervenção Manual | Não é necessária | Não é necessária |
| Escalabilidade | Limitado a 10,000 nodos | Mais de 10,000 nodos |
| Gerenciamento de Recursos | YARN | YARN |
| Balanceamento dos Dados | Usa balanceador HDFS | Usa balanceador Intra-dados |

o que resulta em MRJs mais rápidos na versão atual. Os resultados destas pesquisas podem ser observados na tabela 2.5.

Tabela 2.5: Tempo de execução das tarefas no Hadoop 2.x e 3.x

| **Operação** | **Hadoop 2.x** | **Hadoop 3.x** |
|---|---|---|
| Escrita 500 MB | 65 Segundos | 27 Segundos |
| Escrita 1 GB | 103 Segundos | 27 Segundos |
| Escrita 10 GB | 214 Segundos | 118 Segundos |
| Leitura 500 MB | 62 Segundos | 19 Segundos |
| Leitura 1 GB | 125 Segundos | 16 Segundos |
| Leitura 10 GB | 148 Segundos | 48 Segundos |
| TeraSort 1 GB | 176 Segundos | 83 Segundos |
| TeraSort 10 GB | 423 Segundos | 275 Segundos |

Neste estudo feito por (Masur e Mcintosh, 2017), o *benchmark MRBench* de (Noll, 2011) foi utilizado para testar a camada *MapReduce* de ambas as versões do Hadoop, e o *AverageTime* levado para completar duas tarefas de mapeamento e uma de redução foi contabilizado para cada uma delas. Várias operações de escrita, leitura e classificação foram realizadas em ambas as versões do Hadoop para comparação; e os resultados indicam que o desempenho do Hadoop 3.x excede o do Hadoop 2.x por uma grande margem, especialmente com relação ao tempo médio gasto para a conclusão das tarefas de Mapeamento (*Mappers*) e Redução (*Reducers*).

### 2.6.5 YARN Scheduler Load Simulator

O Agendador de tarefas do YARN é uma área fértil em interesses com diferentes implementações, por exemplo, agendamentos *First In First Out*, *Capacity* e *Fair*, ao mesmo tempo em que várias otimizações também são feitas para melhorar o desempenho dos agendadores para diferentes cenários e cargas de trabalhos. Cada algoritmo do agendador tem seu próprio conjunto de recursos e orienta as decisões de agendamento por muitos fatores, como equidade, garantia de capacidade, disponibilidade de recursos, entre outras. É muito importante avaliar um algoritmo do agendador antes de implantá-lo em um *cluster* de produção. Infelizmente, atualmente não é trivial avaliar um algoritmo do escalonador, pois, avaliar em um *cluster* real é sempre demorado e custoso, além de ser muito difícil encontrar um *cluster* grande o suficiente. Consequentemente, seria bastante útil um simulador que possa prever quão bem será um algoritmo de agendamento para alguma carga de trabalho específica.

O *YARN Scheduler Load Simulator* (SLS) é uma ferramenta que pode simular *clusters* YARN em grande escala e cargas de aplicativos em uma única máquina. Este simulador é

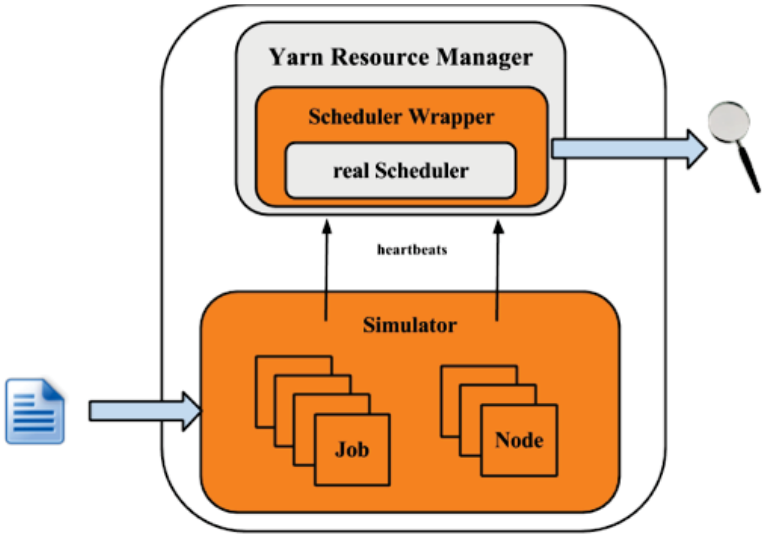
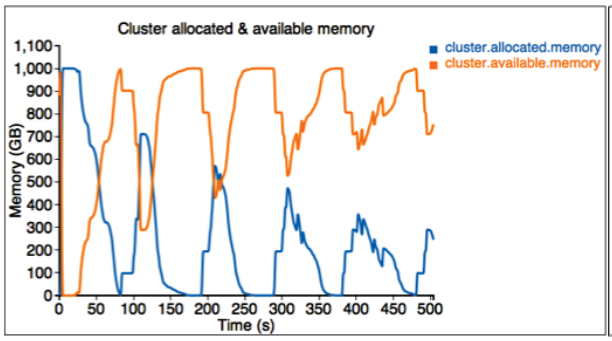
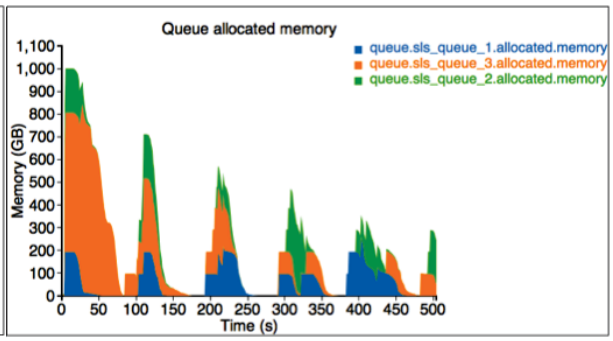


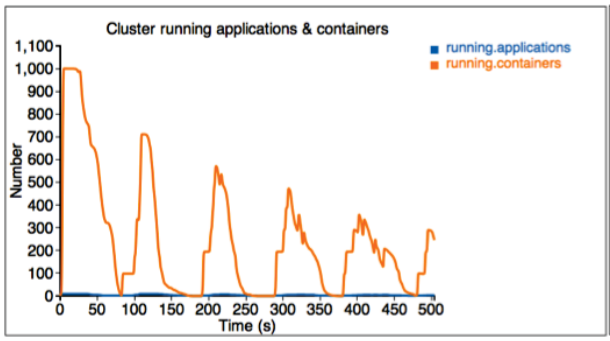

Figura 2.8: Arquitetura do SLS
(Hadoop, 2020)



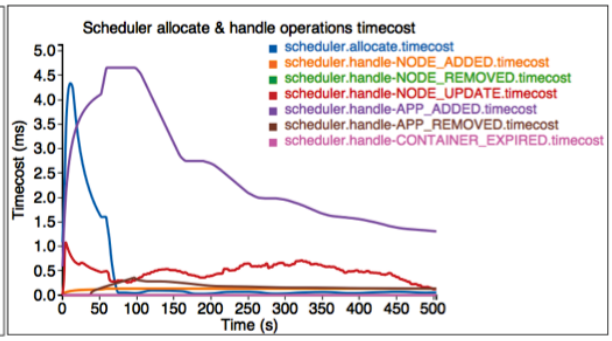

(a) SLS - Cluster Memory



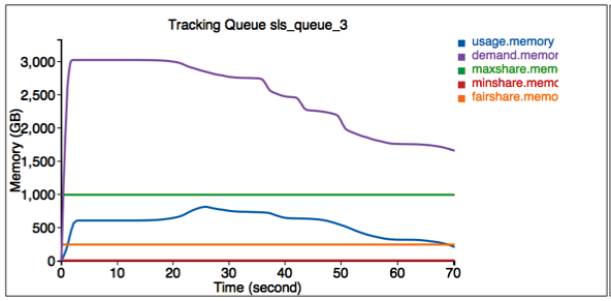

(b) SLS - Queue Allocated Memory




(c) SLS - Running Apps Containers




(d) SLS - Scheduler Operation Timecost




(e) SLS - Track Queue




(f) SLS - Track Job

Figura 2.9: Gráficos gerados em tempo real pelo YARN Scheduler Load Simulator
(Hadoop, 2020)

uma ferramenta para pesquisadores e desenvolvedores criarem protótipos de novos recursos de agendamento e prever seu comportamento e desempenho com uma quantidade razoável de confiança, ajudando assim a inovação. O simulador exercitará o *YARN ResourceManager* real removendo o fator de rede, simulando *NodeManagers* e *ApplicationMasters* por meio do tratamento e envio de eventos de pulsação NM/AMs de dentro da mesma JVM. Para manter o controle do comportamento e desempenho do planejador, um embrulhador do agendador envolverá o agendamento real (Hadoop, 2020).

Como pode-se observar na figura 2.8, o SLS obtém dados de rastreios de cargas de trabalhos ou distribuições de carga sintética e gera as informações de *cluster* e aplicações. Para cada NM e AM, o simulador constrói uma instância de simulador para simular sua execução. Todos os simuladores NM/AM são executados em um *Pool of Threads*. O simulador reutiliza o *YARN Resource Manager* e cria um *Wrapper* a partir do agendador. O *Scheduler Wrapper* pode rastrear os comportamentos do planejador e gerar vários *logs*, que são as saídas do simulador e podem ser analisados posteriormente.

Ao observar a figura 2.9 é possível notar que o SLS produz diversas métricas em tempo real durante a execução, incluindo:

- Usos de recursos para todo o *cluster* e cada fila, que podem ser utilizados para configurar o *cluster* e a capacidade da fila;
- O rastreamento detalhado da execução do aplicativo (registrado em relação ao tempo simulado), que pode ser analisado para entender e validar o comportamento do planejador (tempo de giro de tarefas individuais, rendimento, *fairness*, garantia de capacidade, etc.);
- Várias métricas importantes do algoritmo do planejador de tarefas, como custo de tempo de cada operação do planejador (alocar, manipular, etc.), que podem ser utilizadas pelos desenvolvedores do Hadoop para encontrar os pontos de código e os limites de escalabilidade.

## 2.7 CONSIDERAÇÕES FINAIS

Neste capítulo foram descritos os conceitos explorados nesta dissertação. Primeiramente, na seção 2.1 foi abordada a definição do *Energy Efficient Ethernet* e seus dois modos de configuração *Fast Wake Mode* e *Deep Sleep Mode* os quais possuem um diferente consumo de energia, 60% e 10% respectivamente. Já na seção 2.2, foi descrito o *Packet Coalescing*, um agrupamento de pacotes como forma de limitar o número de interrupções de recebimento de pacotes e, como resultado, diminuir a quantidade de processamento necessária. Isso pode aumentar a latência da rede, mas em contrapartida, aumenta as chances de que a rede consuma uma quantidade menor de energia, principalmente quando aliada ao EEE.

A seção 2.3 descreve o *Active Queue Management*, uma política implementada em roteadores e comutadores para descartar pacotes dentro de um *buffer* associado a um *Network Interface Controller* (NIC) antes que o *buffer* fique cheio, geralmente com o objetivo de reduzir o congestionamento da rede ou melhorar a latência de ponta a ponta. Em seguida, foram descritos dois tipos de AQM, o *Random Early Detection* (RED) que descarta pacotes preventivamente antes que o *buffer* fique completamente cheio; e o *Controlled Delay* (CoDel) que defini limites no atraso dos pacotes à medida que passam pelos *buffers* no equipamento. A seção 2.4 aborda o *Explicit Congestion Notification*, uma extensão do protocolo TCP/IP que pode ser utilizada juntamente ao RED e CoDel. Com o ECN habilitado a conexão tem alguns aprimoramentos,

como por exemplo, realizar notificações de congestionamento de ponta a ponta na rede sem que haja a perda de pacotes.

No item 2.5 é descrito o *MapReduce*, um modelo de programação avançado para processar grandes conjuntos de dados estruturados e principalmente não estruturados que consiste em duas funções, *Map* (usado para encontrar algo ou resolver pequenos problemas) e *Reduce* (usado para reunir os resultados, gerando assim uma solução para o problema maior).

Por fim, a seção 2.6 abrange os conceitos do Apache Hadoop (uma plataforma de código-fonte aberto para armazenamento e processamento de dados em larga escala), como seus principais módulos que compõem o ecossistema central; processos executados na JVM durante uma aplicação; o YARN - sistema de gestão dos recursos dos *Clusters Hadoop*; as diferenças entre as versões 1.x, 2.x e 3.x do Hadoop; além do *YARN Scheduler Load Simulator* (SLS), uma ferramenta que pode simular *clusters YARN* em grande escala e cargas de aplicativos em uma única máquina.

# 3 METODOLOGIA

Este capítulo apresenta a metodologia que foi utilizada ao longo do desenvolvimento de nossa pesquisa, e as justificativas para estas escolhas.

## 3.1 FERRAMENTA DE SIMULAÇÃO

Durante nossa pesquisa bibliográfica, duas ferramentas de simulação compatíveis com o *MapReduce* se destacaram, o MRPerf (Wang et al., 2009) e o SLS (Hadoop, 2020). Avaliado a alguns anos em (Liu et al., 2013), o MRPerf possuí limitações para medir com precisão algumas etapas da estrutura do *MapReduce*, especialmente nas fases de competição por recursos, tendo eficácia apenas na fase *Shuffle*. Além disso, os benchmarks de TeraSort implementados pelos desenvolvedores do Hadoop e utilizados na validação do MRPerf em (Wang et al., 2009) correspondem a versão do Hadoop 1.x, e, há claras diferenças entre as versões do Hadoop, que já foram apresentadas no tópico 2.5.4 do capítulo anterior.

Por outro lado, o SLS vem sendo utilizado constantemente em pesquisas de gerenciamento de filas no Hadoop, como em (Sharma e Ganpati, 2015) e (Shao et al., 2016). Apesar de não ter sido desenvolvido originalmente para levar a rede em conta em suas simulações, recentemente o SLS foi estendido para NetSLS (Wette et al., 2015), passando a integrar um módulo de rede (Wette et al., 2014), que permite avaliar traços em tempo real da simulação.

O NetSLS (Wette et al., 2015) é uma ferramenta de simulação para avaliar ideias que, em conjunto, resolvem o problema de programação de trabalho e fluxo para aplicações de *big data*. Esta ferramenta combina o *Yarn Scheduler Load Simulator* (Hadoop, 2020) com o emulador de rede distribuída MaxiNet (Wette et al., 2014).

Com esta ferramenta, a interdependência entre a rede e as tarefas em execução nela podem ser incluídas na avaliação de novas ideias, aproveitando a pesquisa em aplicativos de *big data* com programação conjunta de trabalho e fluxo de dados como demonstrado na figura 3.1.

Como ilustrado na figura 3.1, a extensão do SLS para NetSLS adiciona transmissões de dados reais às tarefas simuladas. No Hadoop, o processamento de uma tarefa consiste nas três fases sequenciais de busca de dados, processamento de dados e armazenamento de dados. Buscar e armazenar requer acesso aos dados. Dependendo da localização dos dados, isso envolve a rede. Para modelar a rede que interconecta os nós do Hadoop, o NetSLS usa o MaxiNet, um emulador de rede distribuída altamente realista para Redes Definidas por *Software* (SDN). O MaxiNet distribui uma rede virtual em várias máquinas físicas (chamadas de *workes*) tornando possível emular redes muito grandes (como as de *datacenters*) em apenas alguns recursos físicos. Para distribuir um $G_V$ de rede virtual que deve ser emulado em *n workers*, o MaxiNet primeiro divide o $G_V$ em *n* partes. Em seguida, cada partição é emulada em cada um dos *workers*. Todo o tráfego entre as diferentes partições é canalizado por meio da rede física que interconecta os *workes* na partição de destino. Na inicialização, o NetSLS busca a topologia do *cluster* de um arquivo complementar ao rastreamento do trabalho e configura o MaxiNet para emular a rede. Para cada nó $N_i$ simulado do Hadoop, o NetSLS emula um nó correspondente, $\tilde{N}_i$ usando MaxiNet. Sempre que o planejador decide executar uma tarefa no $N_i$, o NetSLS procura informações sobre as transferências de dados do rastreamento do trabalho e inicia as transferências correspondentes em $\tilde{N}_i$. A conclusão da transferência é subsequentemente relatada de $\tilde{N}_i$ para $N_i$ que, por sua vez, relata uma conclusão bem-sucedida ao planejador de trabalho quando as três fases do trabalho terminam (Wette et al., 2015).

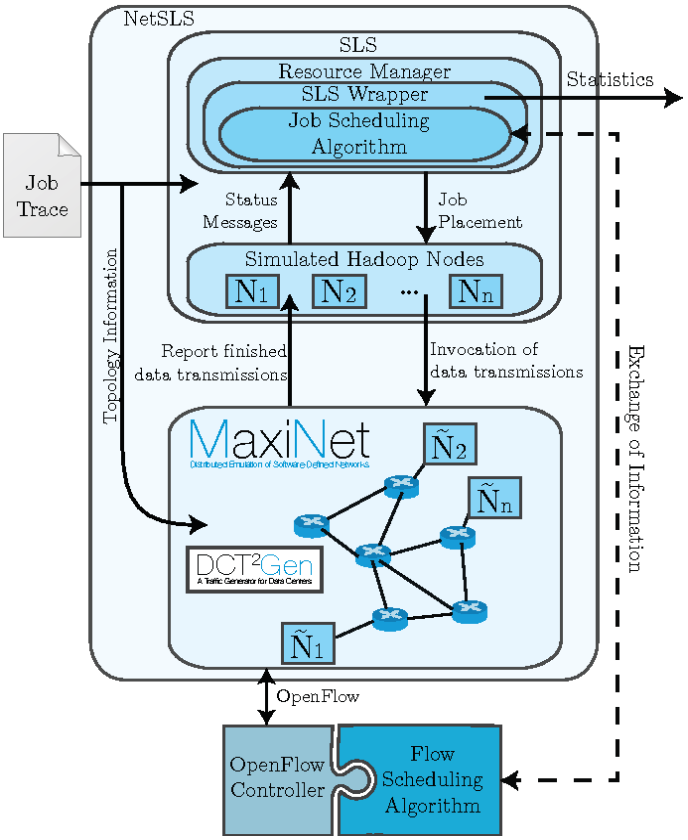



Figura 3.1: Esquema do NetSLS sendo controlado por um OpenFlow externo (Wette et al., 2015)

Como o MaxiNet emula um SDN compatível com *OpenFlow*, um controlador *OpenFlow* gerencia a rede. O *OpenFlow* é um protocolo de comunicação baseado em um *switch Ethernet*, com uma tabela de fluxo interna e uma interface padronizada para adicionar e remover entradas de fluxo (McKeown et al., 2008). Um administrador de rede pode particionar o tráfego em fluxos de produção e pesquisa. Os pesquisadores podem controlar seus próprios fluxos - escolhendo as rotas que seus pacotes seguem e o processamento que recebem. Dessa forma, é possível experimentar novos protocolos de roteamento, modelos de segurança, esquemas de endereçamento e até alternativas ao IP (McKeown et al., 2008).

O controlador é externo ao NetSLS, portanto, qualquer controlador *OpenFlow* pode ser usado para controlar o comportamento da rede. O planejador de fluxo reside no controlador *OpenFlow*. Por fazer parte do controlador, ele tem acesso a todas as informações presentes no controlador. Para trocar informações com o planejador de trabalho, uma interface personalizada entre o planejador de trabalho e fluxo pode ser estabelecida. Para avaliar novas ideias sob diferentes cargas de rede, ao NetSLS também foi integrado o DCT²Gen (Wette e Karl, 2014), um gerador de tráfego para tráfego de *data center* altamente realista. No NetSLS, o DCT²Gen cria tráfego de segundo plano, imitando o tráfego cruzado de outras aplicações.

Ao transformar o SLS em NetSLS, tem-se a oportunidade de testar novos algoritmos de agendamento de trabalho e fluxo. Ao longo do rastreamento de trabalho e uma descrição de topologia, as únicas duas entradas para NetSLS são um algoritmo de agendamento de trabalho e um controlador *OpenFlow* implementando o algoritmo de agendamento de fluxo desejado. O NetSLS irá então simular a interação entre o fluxo e o planejador de trabalho no rastreamento do trabalho, levando a uma avaliação rápida e realista de novos algoritmos de planejamento sem exigir um teste caro ou uma simulação personalizada (Wette et al., 2015).

Diante dessas avalições, optamos para nossa pesquisa a utilização do NetSLS para avaliar o impacto do EEE no Hadoop em relação à economia de energia e desempenho. O NetSLS é o SLS estendido com um modelo realista de rede e tráfego, o MaxiNet. Para poder simular conexões EEE, o NetSLS foi amplificado com um modelo realista (Reviriego et al., 2011). Essa metodologia nos permitiu ter uma visão ampla da execução com traços em *logs*, e assim calcular a economia de energia e o desempenho.

## 3.2 TOPOLOGIA

A topologia do *cluster* escolhida para este trabalho foi a *Leaf-Spine* (Cisco, 2016), como pode ser observado na Figura 3.1. A escolha desta topologia se dá ao fato da arquitetura *Leaf-Spine* ser recomendada para computadores de escala *Warehouse* (Cisco, 2012) quando concentrados em tráfego leste-oeste entre servidores do mesmo *cluster* (Nelson, 2017).

A topologia *Leaf-Spine* traz *Switches* de *Spine* e *Leafs* que são semelhantes aos *Switches* de acesso e agregação encontrados na clássica *K-ary Fat Tree* (Vahdat et al., 2013). A vantagem da topologia *Leaf-Spine* quando comparada com a *K-ary Fat Tree* é sua conectividade de malha completa entre *Switches* de *Leafs* e *Spine*, mas segue de fato uma arquitetura *Two-Tier Clos*. Cada *Switch* em *Leaf*, também conhecido como *Switch Top of Rack* (ToR), é conectado ao *Switch* de *Spine*, denominado *Switch* de Agregação regularmente no modelo de *Fat Tree*, usando um único link. A *Leaf-Spine* pode 'expandir-se' para um número razoavelmente grande de servidores simplesmente adicionando mais *Switches*, embora isso também aumente a quantidade de cabos e os custos dos equipamentos de rede necessários, uma vez que cada *Leaf Switch* deve ser conectado a cada *Spine Switch* (Nelson, 2017).

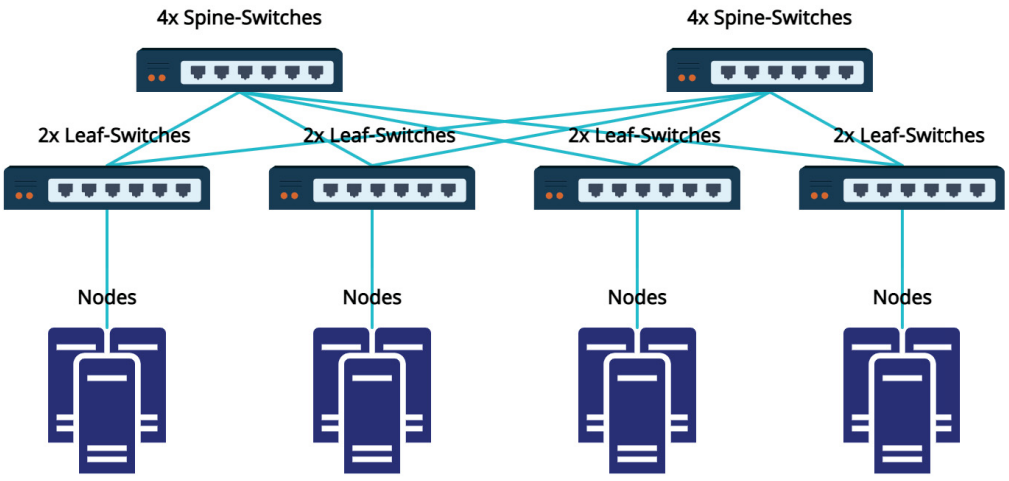



Figura 3.2: Topologia de Rede Leaf-Spine
(Cisco, 2016)

É importante mencionar que a topologia da *Leaf-Spine* pode ser categorizada como uma topologia de *Fat Tree* de dois níveis, ou seja, sem a camada de agregação entre a borda e a camada central, e não está organizada em *Pods* (Solnushkin, 2013); (Testa e Pavesi, 2017). Como o foco de nossa pesquisa foi analisar as configurações do E-EON (Silva, 2018) sobre a atual versão do Hadoop *MapReduce*, decidimos considerar a mesma topologia *Leaf-Spine* para esta pesquisa, que também é recomendada para o Hadoop, como visto em várias referências de *design* de *clusters* (Bechtolsheim et al., 2011); (Cisco, 2012); (Networks, 2014).

Tabela 3.1: Topologia Simulada

| Categoria | Parâmetro | Valor |
|---|---|---|
| Cluster | Nodes | 60, 96, 100, 150, 160, 240, 250, 256, 400 ou 640 |
| | Number of racks | 4 |
| Network | Each node | 1GbE/10GbE/25GbE 40GbE : 1 |
| | | — |
| | Each leaf switch | 1GbE/10GbE/25GbE 40GbE : 15, 24, 25, 37, 38, 40, 60, 62, 63 64, 100 ou 160 10GbE/25GbE/40GbE 100GbE/400GbE: 2 |
| | Each spine switch | — 10GbE/25GbE/40GbE 100GbE/400GbE: 4 |

A Tabela 3.1 detalha a topologia de rede simulada. Simulamos vários *clusters* de 60, 96, 100, 150, 160, 240, 250, 256, 400 e 640 nodos divididos em 4 *racks*, cada nodo com um *Intel Xeon 2.13GHz E5506 quad-core* e um único link no *top-of-rack* (ToR), que varia entre 1GbE, 10GbE, 25GbE ou 40GbE. Cada *switch top-of-rack* é conectado a um *switch* de agregação usando um único link de 10GbE, 25GbE, 40GbE, 100GbE ou 400GbE, dependendo da simulação. As taxas de excesso de assinatura (*oversubscription ratio*) em nossos testes foram de 1.5:1, 2.5:1 e 4:1, o que corresponde à recomendação da Cisco para implantação do *MapReduce* com uma taxa de excesso de assinatura de 4:1 ou menor na camada de acesso (Cisco, 2011). Segundo (Hortonworks, 2021), ter um baixo índice de excesso de assinatura é uma forma de melhorar o desempenho da rede com investimentos em equipamentos.

## 3.3 CONSUMO DE ENERGIA DOS SWITCHES

Um dos aspectos mais importantes desta pesquisa foi a definição do consumo de energia das redes simuladas, dando assim uma base de cálculo correta. Para isso, foi realizada uma pesquisa à procura dos switches que são optados pelas indústrias em produção. Por ter um guia completo disponível com todas as informações necessárias para nossa pesquisa e ser um produto padrão do mercado, optamos pela escolha da Série Juniper QFX5220 (Networks, 2021).

Na tabela 3.2 é possível observar as configurações padrões dos *Switches* desta linha, em que temos 64 *megabytes* de *buffers* distribuídos entre as portas, além de um consumo de 0.5w/h para links ethernet de 1Gb; 2.68w/h para 10Gb; 5.41w/h para 25Gb; 10.89w/h para 100Gb e 22.21/wh para 400Gb.

Tabela 3.2: Configuração dos Switches

| Categoria | Parâmetro | Valor |
|---|---|---|
| Buffers | Juniper QFX5220 | Total packet 64Mb |
| Link Power | 1GbE | 0.5w/h |
| | 10GbE | 2.68w/h |
| | 25GbE | 5.41w/h |
| | 40GbE | 7.49w/h |
| | 100GbE | 10.89w/h |
| | 400GbE | 22.21w/h |

## 3.4 CONFIGURAÇÕES DO CLUSTER HADOOP

Reservamos um nodo de um dos *racks* do *cluster* para execução do *JobTracker*, enquanto 56, 92, 96, 146, 156, 236, 246, 252, 396 ou 636 nodos são reservados para execução das tarefas *Map* e *Reduce*. Além disso, cada um dos 3 *racks* restantes executa 1 *Namenode*, totalizando 3, que é a recomendação da Apache, oferecendo assim um equilíbrio entre escalabilidade e sobrecarga de comunicação (Hadoop, 2021).

Tabela 3.3: Configurações do Cluster Hadoop

| Categoria | Parâmetro | Valor |
|---|---|---|
| Sistema | Nodes | 60, 96, 100, 150, 160, 240, 250, 256, 400 ou 640 |
| | Racks | 4 |
| Config. | Jobtracker | 1 |
| | Namenodes | 3 |
| | Workers | 56, 92, 96, 146, 156, 236, 246, 252, 396 ou 636 |
| | Scheduler Scheme | Fair |
| TCP buffer | Default size | 128 KB por conexão |
| TCP packet | Default size | 1500 bytes |

## 3.5 CONFIGURAÇÕES DO ENERGY EFFICIENT ETHERNET

Para as operações de *wake* e *sleep* do EEE assumimos os valores disponíveis na Tabela 3.4 fornecida por (Reviriego et al., 2009) e (IEEE, 2014), e na Tabela 3.5. Para links de 40GbE e 100GbE, usamos os valores dos modos *Deep Sleep* e Fast Wake (IEEE, 2014), disponíveis nas tabelas 3.4 e 3.6 respectivamente.

Como não há projetos ou pesquisas para implementações do Energy Efficient Ethernet em links de 25GbE e 400GbE, os valores do Min.$T_w$ e Min.$T_s$ não estão disponíveis em nenhuma fonte confiável conhecida. Assim, calculamos os valores médios para 25GbE dos links de 10GbE

Tabela 3.4: EEE - Tempo das operações *wake* e *sleep* no modo *Deep Sleep*

| PHY | Min. *Tw* (µs) | Min. *Ts* (µs) |
|---|---|---|
| 1000Base-T | 16,5 | 182 |
| 10GBase-T | 4,48 | 2,88 |
| 40GBase | 5,5 | 0,9 |
| 100GBase | 5,5 | 0,9 |

e 40GbE; por outro lado, para o link de 400GbE, assumimos os mesmos valores dos links de 100GbE fornecidos por (IEEE, 2014), uma vez que o projeto se assemelha.

Tabela 3.5: EEE - Tempos assumidos para as operações *wake* e *sleep* no modo *Deep Sleep*

| PHY | Min. *Tw* (µs) | Min. *Ts* (µs) |
|---|---|---|
| 25GBase | 4,99 | 1,89 |
| 400GBase | 5,5 | 0,9 |

Tabela 3.6: EEE - Tempo das operações *wake* e *sleep* no modo *Fast Wake*

| PHY | Min. *Tw* (µs) | Min. *Ts* (µs) |
|---|---|---|
| 40GBase | 0,34 | 0,25 |
| 100GBase | 0,34 | 0,25 |

## 3.6 CONFIGURAÇÕES DO PACKET COALESCING

De acordo com (Herrerı et al., 2012), configurar o *Packet Coalescing* da forma correta é o ponto chave crítico para economia de energia e performance adequadas. Nossas configurações vem de artigos publicados ao longo de pesquisas que envolvem *Packet Coalescing* e *Energy Efficient Ethernet*, como 12$\mu$s10 e 120$\mu$s100 de (Christensen et al., 2010), 500$\mu$s500 de (Silva, 2018) e 1ms1000 de (Reviriego et al., 2010). Configurações com tempo de espera acima de 10ms não foram consideradas, pois, degradam drasticamente o desempenho do *MapReduce* (Silva, 2018).

Tabela 3.7: Configurações do *Packet Coalescing*

| Label | Tempo de Espera | Gatilho |
|---|---|---|
| 12µs10 | 12µs | 10 pacotes |
| 120µs100 | 120µs | 100 pacotes |
| 500µs500 | 500µs | 500 pacotes |
| 1ms1000 | 1ms | 1000 pacotes |

## 3.7 CONFIGURAÇÕES DO ACTIVE QUEUE MANAGEMENT E EXPLICIT CONGESTION NOTIFICATION

Para a utilização da *Active Queue Management* escolhemos dois algoritmos, o *Random Early Detection* e o *Controlled Delay*.

Tabela 3.8: Valores definidos $Min_{th}$ e $Max_{th}$ para o *Random Early Detection*

| *Label* | $Min_{th}$ | $Max_{th}$ |
|---|---|---|
| Min70Max70 | 70 | 70 |
| Min125Max375 | 125 | 375 |
| Min25Max75 | 25 | 75 |
| Min50Max150 | 50 | 150 |

O primeiro, RED, é normalmente implementado usando o AQL (*Average Queue Length*) para decisões de marcação ou descarte de pacotes, entretanto, usamos o IQL (*Instant Queue Length*), pois trabalhos anteriores já demonstraram que se encaixa melhor em redes de alta velocidade como em *data centers* (Alizadeh et al., 2010); (Silva, 2018). Por fim, definimos os valores demonstrados na tabela 3.8 baseados em trabalhos anteriores, como 70 $Min_{th}$ e 70 $Max_{th}$ em (Alizadeh et al., 2010); 125 $Min_{th}$ e 375 $Max_{th}$, 25 $Min_{th}$ e 75 $Max_{th}$, 50 $Min_{th}$ e 150 $Max_{th}$ em (Silva, 2018).

O segundo, CoDel, usamos a implementação *default* sem nenhuma modificação. O CoDel é considerado uma fila sem parâmetros, mas o administrador da rede ainda precisa fornecer dois valores, atraso de destino (*target delay*) e intervalo (*interval*).

Tabela 3.9: Valores definidos para o *Controlled Delay*

| *Label* | *Target delay* | *Interval* |
|---|---|---|
| 500µs20ms | 500µs | 20ms |
| 300µs0.75µs | 300µs | 0.75µs |
| 400µs1ms | 400µs | 1ms |
| 800µs1.5ms | 800µs | 1.5ms |

Os valores assumidos para o CoDel foram de: 500$\mu$s para *target delay* e 20ms para *interval* fornecidos por (Getty, 2021); 300$\mu$s para *target delay* e 0.75$\mu$s para *interval* fornecidos por (Ismail et al., 2010); 400$\mu$s e 800$\mu$s para *target delay*, 1ms e 1.5ms para *interval* fornecidos por (Silva, 2018); e estão expostos na tabela 3.9.

Por fim, utilizamos a configuração *deafult* do *Explicit Congestion Notification* considerando uma implementação real (nos *switches*) - nesta configuração pacotes ECT-capable, SYN, SYN-ACK e ACKs que possuam o ECE-bit não são descartados antecipadamente.

## 3.8 CARGAS DE TRABALHO

Para os testes de nossa pesquisa foram definidas duas cargas de trabalho, *small tasks* e batch. A *small tasks* consiste em uma sequência de pequenos trabalhos, cada um com dez tarefas de mapeamento (*map*) e uma tarefa de redução (*reduce*). Forçamos uma utilização média do CPU em torno de 40%, o que é compatível com um estudo de traços obtido no Facebook (Chen et al., 2011).

Tabela 3.10: Cargas de Trabalho Simuladas

| Categoria | Parâmetro | Valor |
|---|---|---|
| Jobs | Maps | Small jobs: 10 |
| | | Batch jobs: 2 |
| | Reduces | Small jobs: 1 |
| | | Batch jobs: 2 |

Quanto ao *batch*, a carga de trabalho foi semelhante ao processamento em lote para aplicativos de *big data* reais (Chen et al., 2012), e envolvemos todo o sistema usando um único *job* grande, com o número de tarefas de mapeamento (*map*) e redução (*reduce*) igual ao dobro do número de nodos de trabalho.

## 3.9 CICLO DE TESTES

Para nosso ciclo de testes foi definido uma metodologia baseada na média aritmética 3.1 (Plackett, 1958) e desvio padrão 3.2 (Brenner e Subrahmanyan, 1988).

$$A = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + \cdots + a_n}{n} \tag{3.1}$$

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{3.2}$$

Primeiro é realizado cinco testes e calculado o desvio padrão dos mesmos. Os novos testes são interrompidos quando um novo teste está dentro da média dos testes anteriores acrescida ou subtraída do desvio padrão. Então é calculado a média dos testes anteriores juntamente ao novo teste, e por fim anotado o resultado final da simulação.

$$nT \leq A + s \quad \text{ou} \quad nT \geq A - s \tag{3.3}$$

Pode-se observar de forma clara na fórmula 3.3, em que **'nT'** significa novo teste, **'A'** corresponde a média aritmética e **'s'** ao desvio padrão. Desta forma atingimos uma confiabilidade dos resultados de 95% (Peck et al., 2015).

## 3.10 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados as escolhas de configurações para o desenvolvimento da pesquisa. Primeiramente nossa ferramenta de simulação, o NetSLS (Wette et al., 2015), uma extenção do *YARN Scheduler Load Simulator* (Hadoop, 2020), em que torna-se possível a simulação de *clusters* Hadoop *MapReduce* em máquinas convencionais, facilitando assim o teste de protocolos de rede e esquemas de gerenciamento de fila.

Em seguida, definimos a topologia *Leaf-Spine* (Cisco, 2016) baseada em trabalhos anteriores sobre máquinas de escala *Warehouse* concentradas em tráfego leste-oeste entre servidores do mesmo *cluster* (Cisco, 2012); (Nelson, 2017). Quanto ao consumo de energia dos *switches*, por ser um produto padrão do mercado, optamos pela escolha da Série Juniper QFX5220 (Networks, 2021). O *cluster* Hadoop foi configurado de acordo com as recomendações

da Apache (Hadoop, 2021), oferecendo assim um equilíbrio entre escalabilidade e sobrecarga de comunicação.

Para as conexões Ethernet escolhemos utililzar *Active Queue Managements* como *Controlled Delay* e *Random Early Detection* com *Explicit Congestion Notification* juntamente ao *Energy Efficient Ethernet*, com o objetivo de procurar as melhoreres combinações e configurações dentre as disponíveis (Alizadeh et al., 2010); (Christensen et al., 2010); (Getty, 2021); (Ismail et al., 2010); (Reviriego et al., 2010); (Silva, 2018) para as conexões de 1GbE, 10GbE, 25GbE, 40GbE, 100GbE e 100GbE. Por fim, aplicamos cargas de trabalhos *small tasks* e *batch* semelhantes à aplicativos de *big data* reais (Chen et al., 2011); (Chen et al., 2012), e definimos um ciclo de testes baseado na média aritmética 3.1 (Plackett, 1958) e desvio padrão 3.2 (Brenner e Subrahmanyan, 1988) que proporciona uma confiabiliade dos resultados de 95% (Peck et al., 2015).

# 4 ENERGY EFFICIENT ETHERNET EM CLUSTERS HADOOP MAPREDUCE

Neste capítulo são apresentados os resultados das simulações de *clusters Hadoop 3.x MapReduce* utilizando a ferramenta NetSLS e o *Energy Efficient Ethernet* em modo *Fast Wake* e *Deep Sleep*. Através destes dados mostramos que uma boa economia de energia pode ser obtida em conexões de até 40GbE, enquanto conexões acima de 100GbE possuem problemas de performance quando o EEE está habilitado no modo *Deep Sleep*.

## 4.1 ECONOMIA DE ENERGIA

Esta seção contém os dados de economia de energia obtidos das simulações com topologia *leaf-spine* e conexões de 1GbE e 10GbE; 1GbE e 25GbE; 10GbE e 100GbE; 10GbE e 400GbE; 25GbE e 400GbE; e, finalmente, 40GbE e 400GbE. O restante da metodologia empregada pode ser visualizada no Capítulo 3 de forma detalhada.

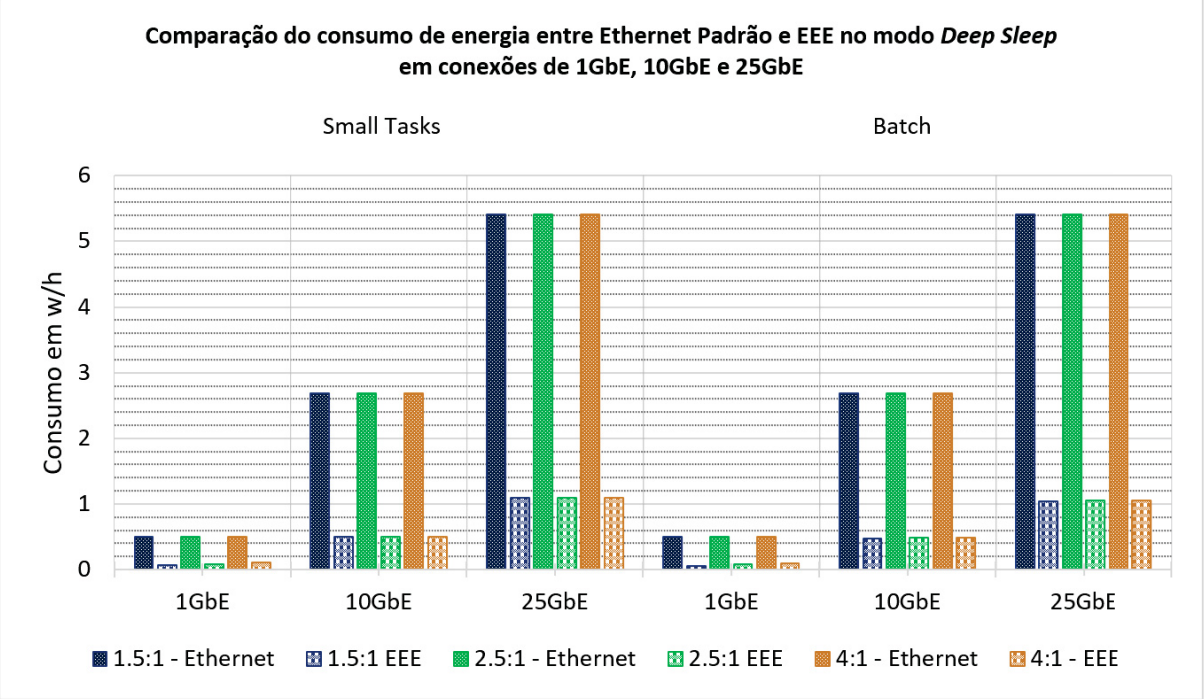



Figura 4.1: Comparação do consumo de energia entre *Ethernet* Padrão e EEE no modo *Deep Sleep* em conexões de 1GbE, 10GbE e 25GbE

Através das simulações com o modo ***Deep Sleep*** do EEE encontramos os dados mostrados nas figuras 4.1 e 4.2, que demonstram a economia de energia em *Jobs MapReduce* em *clusters* Hadoop 3.x, com reduções médias de 0,5w/h para 0,080w/h em links de 1GbE - redução de 83,89%; 2,68w/h para 0,490w/h em 10GbE - redução de 81,68%; de 5,41w/h para 1,072w/h em 25GbE - redução de 80,12%; 7,49w/h para 1,487w/h em 40GbE - redução de 80,14%; de 10,89w/h para 2,646w/h em 100GbE - redução de 75,69%; e de 22,21w/h para 10,685w/h em 400GbE - redução de 51,89.

Usando a configuração de topologia de rede *leaf-spine* recomendada pela Cisco, com uma taxa de assinatura excessiva igual a 4:1, encontramos resultados que, como em (Silva, 2018); (Silva e Carpenter, 2015); (Silva e Carpenter, 2017a), apontam para economias de energia

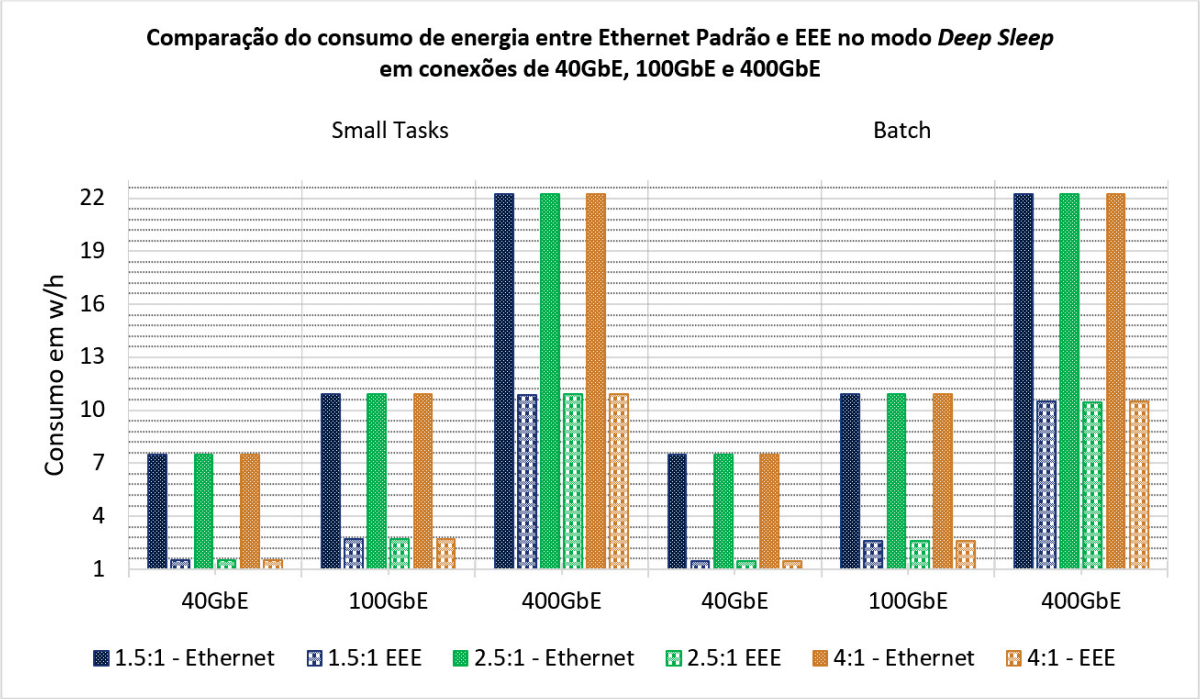



Figura 4.2: Comparação do consumo de energia entre *Ethernet* Padrão e EEE no modo *Deep Sleep* em conexões de 40GbE, 100GbE e 400GbE

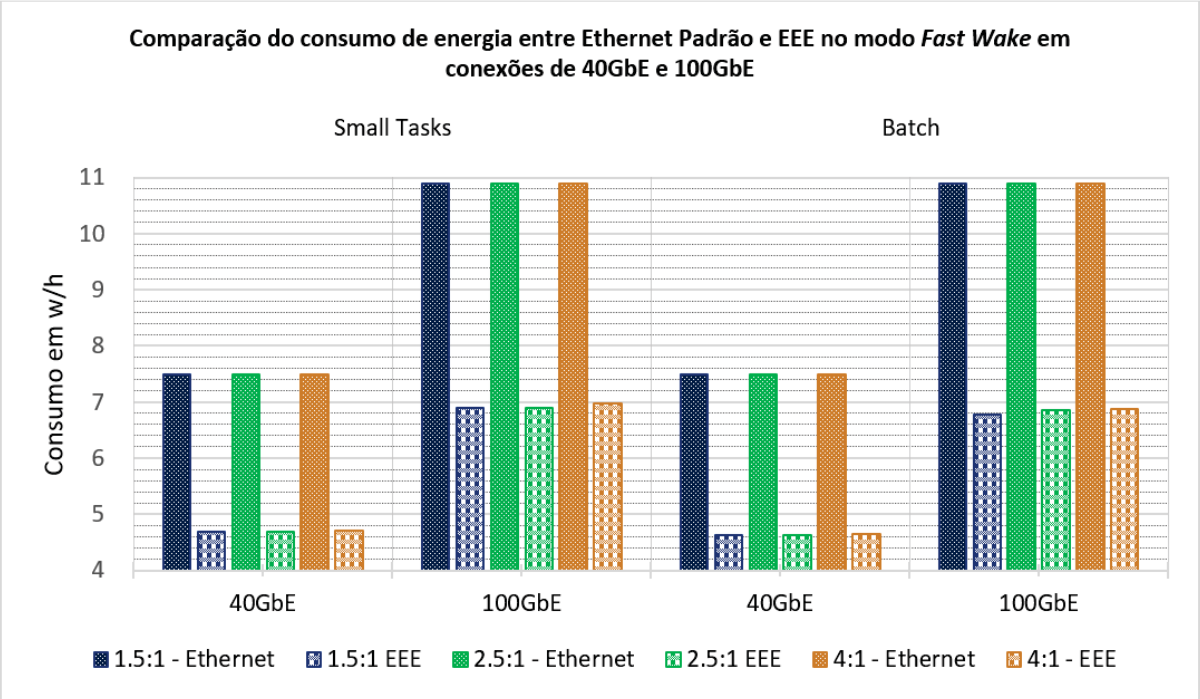



Figura 4.3: Comparação do consumo de energia entre *Ethernet* Padrão e EEE no modo *Fast Wake* em conexões de 40GbE e 100GbE

semelhantes habilitando o EEE no modo ***Deep Sleep*** ao executar *Small Tasks* e *Batch Jobs*, no entanto, os *Batch Jobs* tiveram um desempenho ligeiramente melhor.

Como pode-se observar na figura 4.3, ao habitarmos o EEE no modo ***Fast Wake*** para as conexões de 40GbE e 100GbE, percebe-se uma economia de energia relativamente menor do que comparado ao modo ***Deep Sleep***, de 7,49w/h para 4,634w/h em conexões de 40GbE - redução de 38,13%, e de 10,89w/h para 6,916w/h - redução de 36,49% em conexões de 100GbE respectivamente. Em contrapartida, o desempenho foi melhor como detalhado na próxima seção.

## 4.2 DESEMPENHO

Esta seção contém os dados de performance obtidos das simulações com topologia *leaf-spine* e conexões de 1GbE e 10GbE; 1GbE e 25GbE; 10GbE e 100GbE; 10GbE e 400GbE; 25GbE e 400GbE; e, finalmente, 40GbE e 400GbE. O restante da metodologia empregada pode ser visualizada no Capítulo 3 de forma detalhada.

Ao contrário do que afirmam os fornecedores de equipamentos, nossos testes demonstram que é possível obter uma boa economia de energia habilitando o *Energy Efficient Ethernet* no modo ***Deep Sleep***, com uma perda média de desempenho de praticamente zero para links de 1GbE - 0,21%; 10GbE - 0,94%; e 25GbE - 1,52%; ou razoável no caso de 40GbE - 2,85% conforme mostrado nas figuras 4.4 e 4.5. Para links de 100GbE e 400GbE há uma boa economia de energia, mas com uma perda de desempenho que não é considerada ideal, em torno de 4,55% e 8,58% respectivamente.

Em geral, ao habilitar o EEE em modo ***Deep Sleep*** no *cluster*, percebe-se que os *Batch Jobs* obtêm melhor economia de energia e desempenho em relação aos *Small Tasks*. Ainda é possível observar um cenário específico, em que ao diminuir a taxa de assinatura excessiva da rede para 1.5:1 em links 1GbE, os *Batch Jobs* obtiveram uma melhora significativamente maior na economia de energia e no desempenho em comparação com outras conexões.

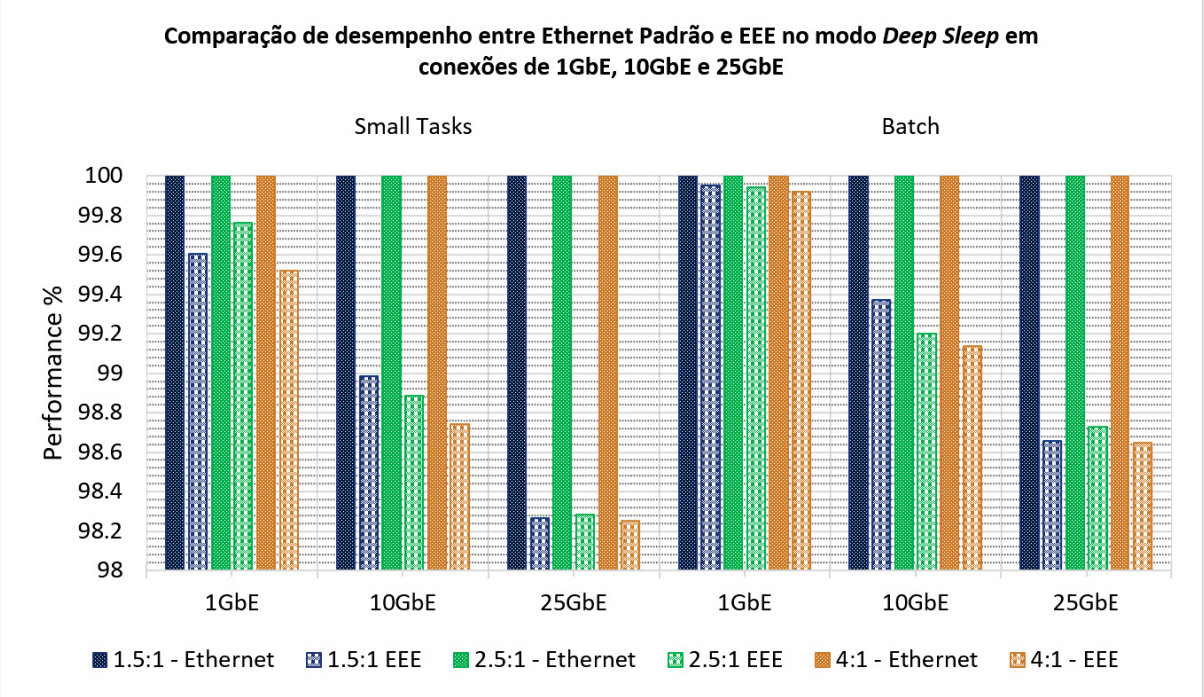



Figura 4.4: Comparação de desempenho entre *Ethernet* Padrão e EEE no modo *Deep Sleep* em conexões de 1GbE, 10GbE e 25GbE

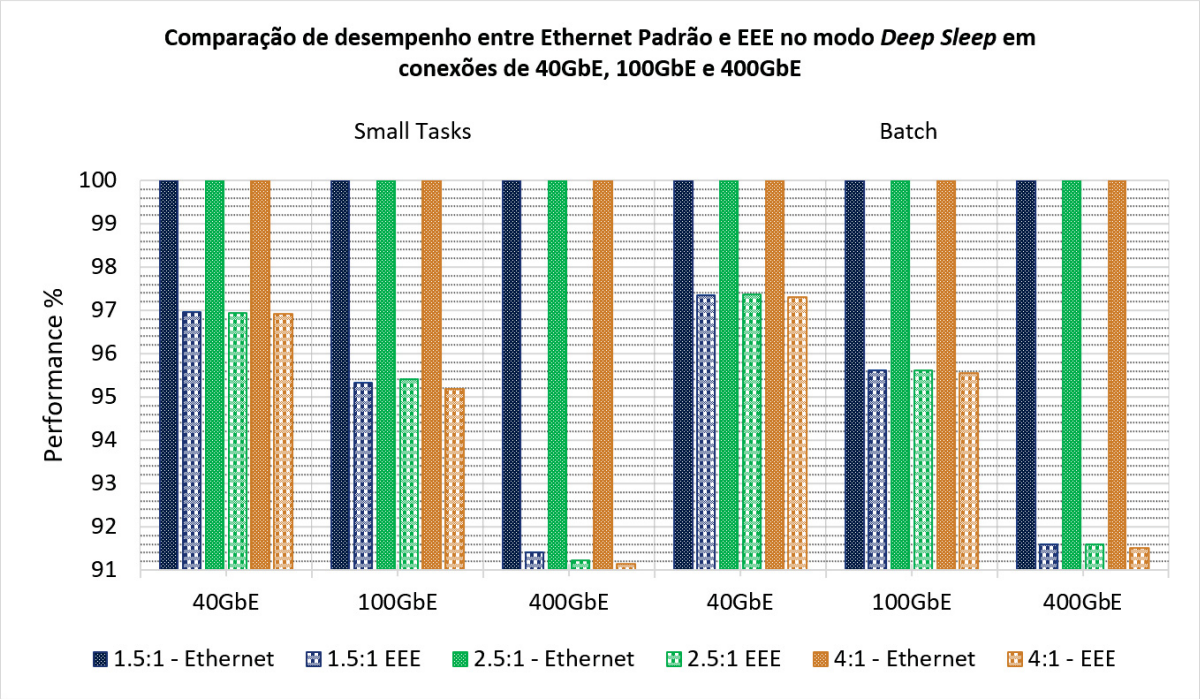



Figura 4.5: Comparação de desempenho entre *Ethernet* Padrão e EEE no modo *Deep Sleep* em conexões de 40GbE, 100GbE e 400GbE

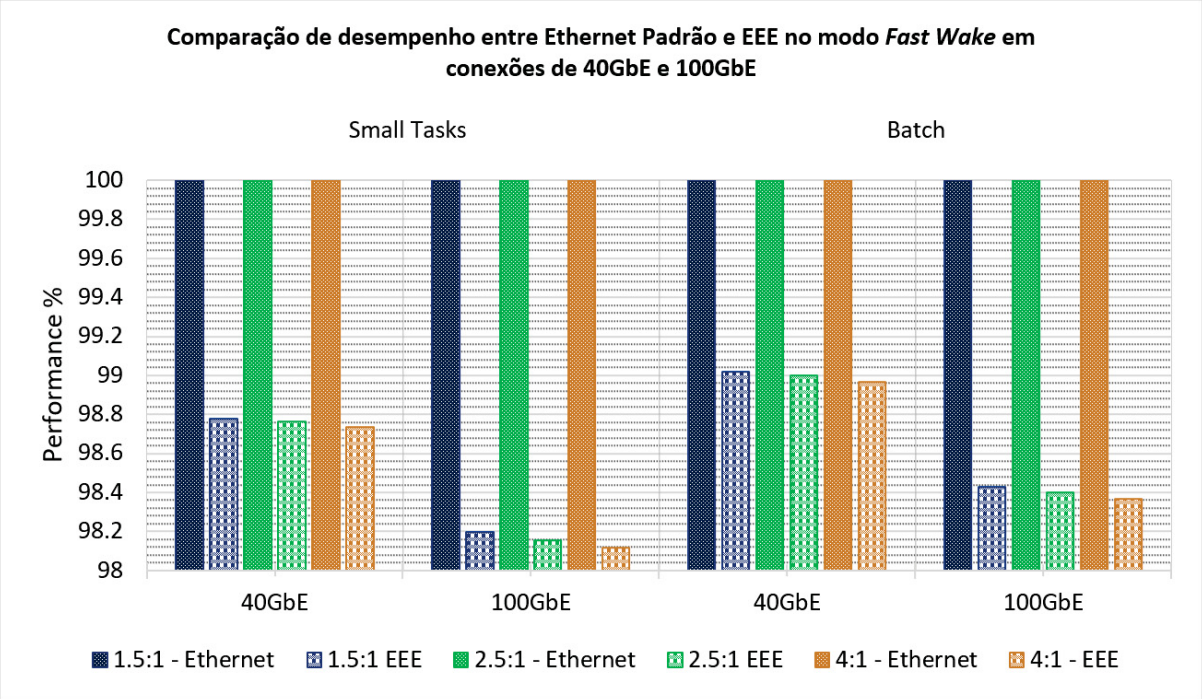



Figura 4.6: Comparação de desempenho entre *Ethernet* Padrão e EEE no modo *Fast Wake* em conexões de 40GbE e 100GbE

Ao utilizarmos o modo ***Fast Wake*** do EEE em nossas simulações, foi possível notar que embora este modo não forneça uma economia de energia como o modo ***Deep Sleep***, ele não afeta o desempenho de forma que seja considerável, com uma perda de 1,2% no caso das conexões de 40GbE e 1,8% em conexões de 100GbE, tornando-se assim uma opção interessante para as indústrias que buscam um balanceamento entre economia de energia e desempenho.

As perdas de desempenho mais significativas encontradas para os links de 100GbE e 400GbE com o modo ***Deep Sleep*** ocorreram porque em determinados momentos é necessário acordar um link para transmitir um único *frame*, o que causa penalidades de latência e consumo de energia em termos relativos que são agravadas de acordo com a largura de banda (Jiang et al., 2021); (Reviriego et al., 2009); (Reviriego et al., 2010); (Silva e Carpenter, 2017a). Se os momentos de *wake* do EEE puderem ser controlados para evitar transmissões de um *frame* único, pode ser possível obter economia de energia com perdas de desempenho baixas ou nulas para estas conexões.

## 4.3 CONSIDERAÇÕES FINAIS

Neste capítulo, apresentamos uma análise do impacto ao habilitar ao *Energy Efficient Ethernet* nos modos ***Deep Sleep*** e ***Fast Wake*** em *clusters MapReduce* que utilizam o *framework* Apache Hadoop 3.x. Avaliamos execuções de *Small Tasks* e *Batch Jobs* com EEE em diversos *clusters* simulados, e no modo ***Deep Sleep*** encontramos economia de energia entre 78% e 82% para links de até 40GbE sem perda considerável de desempenho, entre 0,21% e 2,85 %. Para links de 100GbE e 400GbE houve uma economia de energia significativa de 75,69% e 51,88% respectivamente, mas com uma taxa de perda de desempenho considerável de 4,55% e 8,58%, o que não é particularmente interessante em *Batch Jobs*. Quando optamos pelo uso do modo ***Fast Wake*** do EEE, obtivemos uma redução do consumo de 7,49w/h para 4,634w/h em conexões de 40GbE, e de 10,89w/h para 6,916w/h em conexões de 100GbE, com uma perda de performance de 1.2% e 1.8%.

Em termos de custo-benefício para *data centers*, o recomendado para conexões de até 40GbE é habilitar o EEE no modo ***Deep Sleep***, o que resulta em uma boa redução do consumo de energia e uma taxa de perda de desempenho baixa ou nula. Para conexões de 100GbE o recomendado é habilitar o modo ***Fast Wake*** do EEE, obtendo assim um balanceamento entre consumo de energia e performance. Para conexões de 400GbE habilitar o EEE não é interessante, pois há uma perda de 8,58% do desempenho.

O problema de latência e economia de energia encontrado em conexões acima de 100GbE com ***Deep Sleep*** são causados por ter que acordar o link de tempos em tempos para transmitir um único *frame*. Para resolver este problema, no próximo capítulo combinamos o *Energy Efficient Ethernet* com *Packet Coalescing*, *Random Early Detection*, *Controlled Delay* e *Explicit Congestion Notification*, podendo assim configurar de forma manual e inteligente quando os links devem ser acordados do modo LPI para transmissões de pacotes.

# 5 ECONOMIA DE ENERGIA, CONTROLE DA LATÊNCIA E OTIMIZAÇÃO DE DESEMPENHO

Este capítulo apresenta os resultados quantitativos obtidos através de nossas simulações em busca de uma economia de energia ainda maior, e do aumento de performance em relação ao *Energy Efficient Ethernet* padrão. Para isso, combinamos primeiramente o EEE com *Packet Coalescing*, maximizando assim a economia de energia, em seguida, utilizamos dois diferentes AQMs, o *Random Early Detection* e o *Controlled Delay*, juntamente com *Explicit Congestion Notification* para diminuir a latência e aumentar a performance nas cargas de trabalho do *Hadoop 3.x MapReduce*.

Redes de *data centers* estão começando a empregar uma nova geração de *switches* com *buffers* maiores, para acomodar comunicações em rajadas (*bursty*) e oferecer melhor desempenho nas aplicações. Uma grande parte dos aplicativos atualmente usa o TCP como protocolo de transmissão, e sofrem com grande latência de pacotes, devido à tendência do TCP de utilizar totalmente o buffer disponível. *Switches* recentes que implementam AQM e suportam ECN permanecem com estas tecnologias desativadas, deixando desnecessariamente um recurso que poderia reduzir significativamente diminuir a latência melhorando assim a performance (Silva, 2018).

Com este estudo, encontramos economias de energia ideais para as conexões acima de 10GbE, mas somente quando o *Packet Coalescing* está habilitado. Com o *Packet Coalescing*, os *switches* atrasam intencionalmente os pacotes de saída enquanto o link está no modo LPI, para que possam ser transmitidos consecutivamente com os pacotes subsequentes (Silva, 2018). As configurações do *Packet Coalescing*, no entanto, devem ser cuidadosamente escolhidas e testadas para evitar uma perda excessiva de desempenho.

## 5.1 ENERGY EFFICIENT ETHERNET COM PACKET COALESCING

A metodologia de forma detalhada está presente no Capítulo 3. Para o EEE, primeiramente testamos o modo *Deep Sleep*, e posteriormente o *Fast Wake*. Quanto ao *Packet Coalescing*, assumimos valores de configurações de 12$\mu$s10 e 120$\mu$s100 (Christensen et al., 2010), 500$\mu$s500 (Silva, 2018) e 1ms1000 (Reviriego et al., 2010).

### 5.1.1 Economia de Energia

Durante nossos experimentos não encontramos economia de energia suficiente para conexões de 1GbE que justificasse a implementação do *Packet Coalescing*, pelo contrário, em alguns casos ainda houve um aumento do gasto energético em 0.17% (de 0.0805 w/h para 0.0810 w/h), como no caso dos *Batch Jobs* em topologia 4:1.

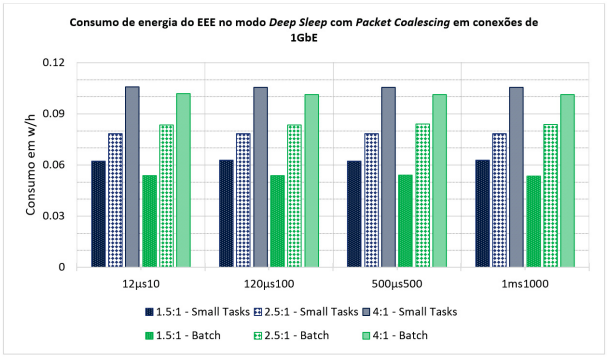
Para conexões de 10GbE, o gasto energético foi reduzido de 18.32% (0.4909 w/h) para 11.82% (0.3168 w/h) com a configuração 120$\mu$s100, reduzindo assim quase pela metade do consumo de energia em relação ao EEE padrão. Para este link a pior configuração foi de 12$\mu$s10, o que resultou em uma redução baixa do consumo de 18.32% para 15.08% (0.4042 w/h), seguida de e 1ms1000 com redução de 18.32% para 15.06% (0.4035 w/h).

Em links de 25GbE, a redução foi ainda mais significativa, caindo de 19.82% (1.0725 w/h) para 13.37% (0.7235 w/h) na configuração 120$\mu$s100. Neste link, a pior economia foi obtida pela configuração 1ms1000, que reduziu o consumo de 19.82% para 16.71% (0.904 w/h).

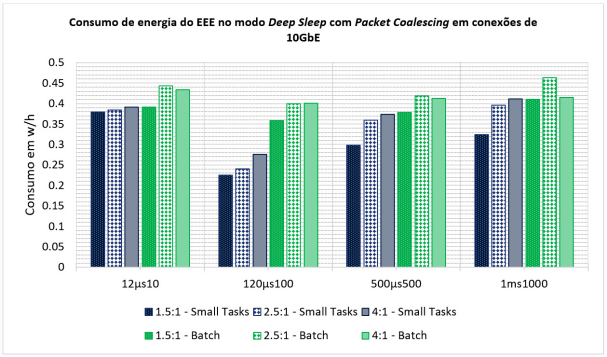
Uma redução do consumo de 19.86% (1.4877 w/h) para 13.24% (0.9917 w/h) foi encontrada em conexões de 40GbE com o EEE no modo *Deep Sleep* ao aplicar a configuração 120$\mu$s100, fazendo com que este link obtivesse um gasto energético similar ao de 25GbE.

Para as conexões de 100GbE com o EEE no modo *Deep Sleep*, a redução do consumo foi de 24.31% (2.6469 w/h) para 15.72% (1.7115 w/h) em nossa melhor configuração 120$\mu$s100, e 19.29% (2.1010 w/h) em nossa pior 12$\mu$s10.

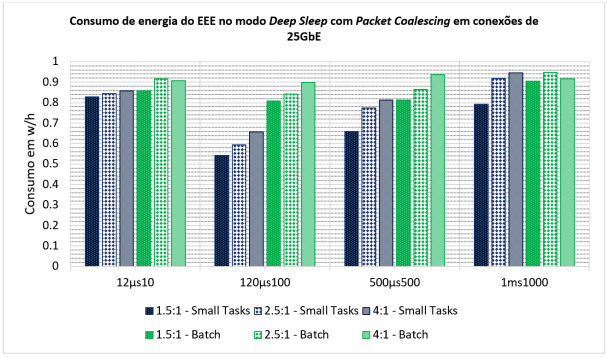
A maior redução do consumo de energia foi encontrada em conexões de 400GbE ao aplicarmos o *Packet Coalescing* com a configuração 120$\mu$s100, reduzindo assim o consumo do EEE em relação ao Ethernet Padrão de 48.11% (10.6853 w/h) para 25.88% (5.7483 w/h). Neste link, a pior configuração foi de 12$\mu$s10, o que resultou em uma redução de 48.11% para 30.67% (6.8107 w/h).



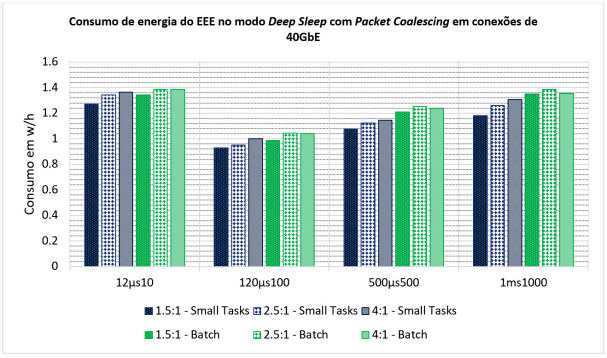

(a) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing* em conexões de 1GbE



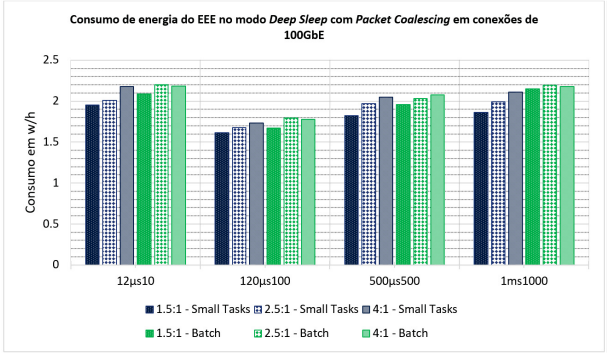

(b) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing* em conexões de 10GbE



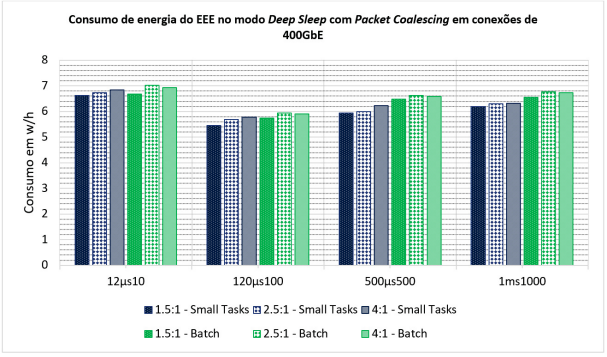

(c) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing* em conexões de 25GbE




(d) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing* em conexões de 40GbE




(e) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing* em conexões de 100GbE

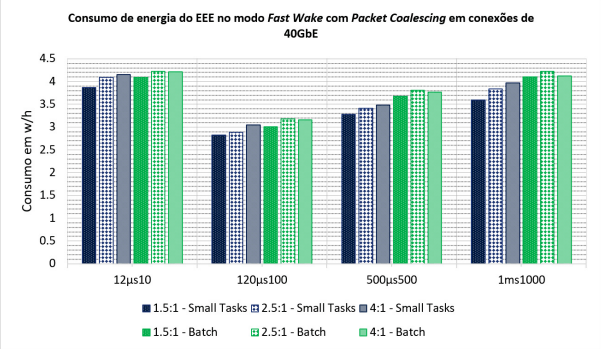



(f) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing* em conexões de 400GbE

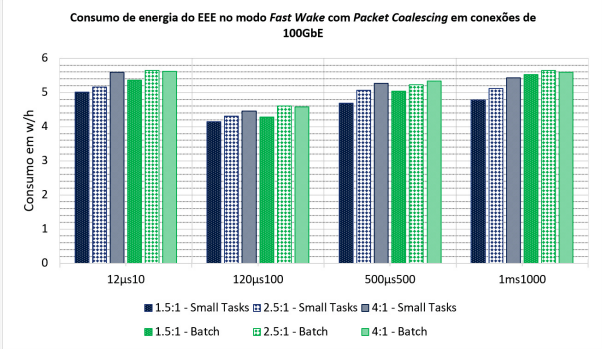
Figura 5.1: Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*

O modo *Fast Wake* do EEE tende a consumir mais energia que o modo padrão, *Deep Sleep*. Para este modo também encontramos reduções no consumo de energia consideráveis nos links que abrigam esta tecnologia no momento, 40GbE e 100GbE. No caso dos links de 40GbE o gasto energético em relação ao Ethernet foi reduzido de 61.87% (4.6340 w/h) para 40,3% (3.0182 w/h) em nossa melhor configuração 120$\mu$s100, e de 61.87% para 54.83% (4.1067 w/h) na pior 12$\mu$s10. Para os links de 400GbE nossa melhor configuração do *Packet Coalescing* 120$\mu$s100, reduziu o consumo de 63.51% (6.9160 w/h) para 40.37% (4.3966 w/h).

As figuras 5.1 e 5.2 ilustram todos os resultados expostos nesta subseção, e revelam ainda que *Batch Jobs* tendem a gastar em média 1% de energia a mais que os *Small Tasks*, e em ambos, pode-se reduzir o consumo de energia através de uma taxa de assinatura excessiva baixa (próxima à 1:1).




(a) Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing* em conexões de 40GbE




(b) Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing* em conexões de 100GbE

Figura 5.2: Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*

### 5.1.2 Desempenho

Como observado por (Mostowfi e Christensen, 2011) e (Silva, 2018), aplicar o *Packet Coalescing* em conexões com *Energy Efficient Ethernet* sem estar configurado corretamente pode acarretar em uma degradação do desempenho do *Cluster*, além de não obter uma economia de energia vantajosa. Nesta subseção estão expostos os resultados de performance das nossas simulações, em que identificamos configurações que à mantém relativamente estável, enquanto outras a degradam exponencialmente.

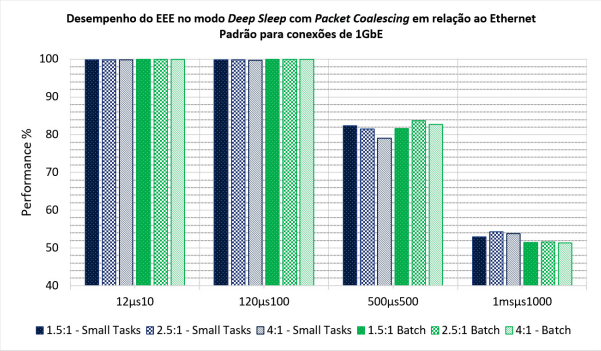
As figuras 5.3 e 5.4 ilustram os resultados para o *Packet Coalescing* combinado com EEE no modo *Deep Sleep* e *Fast Wake* respectivamente. Em conexões de 1GbE a melhor configuração 12$\mu$s10 trouxe uma ligeira melhora no desempenho, de 99.783% para 99.92% em relação ao Ethernet padrão. Por outro lado, uma configuração ruim como é o caso de 1ms1000, degradou o desempenho para 52.566%.

Os links de 10GbE demonstraram performance relativamente iguais quando configurados com 12$\mu$s10 e 120$\mu$s100, cera de 99.175% e 99.141%. Por outro lado, a configuração de 1ms1000 degradou o desempenho para 52.352%. Nos links de 25GbE e 40GbE, as configurações 12$\mu$s10 e 120$\mu$s100 também surgiram efeitos parecidos, melhorando levemente o desempenho de 98.472% para 98.53% e de 97.147% para 98.26%. Assim como nas conexões anteriores, a configuração 1ms1000 degradou o desempenho para 51.577% e 50.189%.

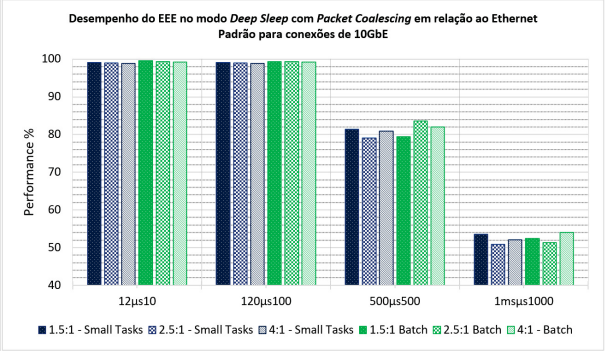
A partir das conexões de 100GbE, o desempenho da configuração 12$\mu$s10 degradou-se em relação à 120$\mu$s100. No caso dos links de 100GbE, obtivemos uma estabilidade do desempenho ao utilizarmos 120$\mu$s100, passando de 95.44% para 95.53%, enquanto 12$\mu$s10

reduziu o desempenho para 93.13%. Para 400GbE, 120$\mu$s100 alterou o desempenho de 91.41% para 91.51%, enquanto 12$\mu$s10 o reduziu para 87.476%.

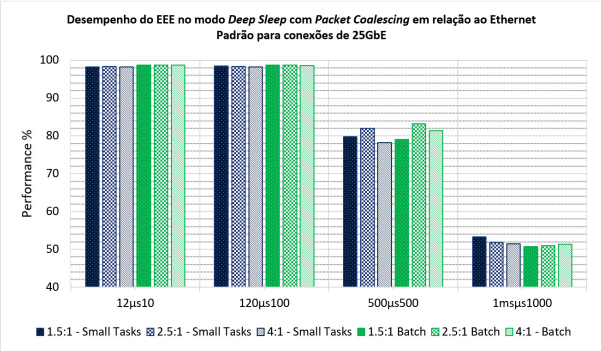
Em todos os casos a configuração 1ms1000 degradou o desempenho em mais de 40%. Isso se deve ao fato de que configurar o *Packet Coalescing* com tempos de espera e gatilhos maiores fazem com que a rede tenha transmissões mais intensas (*burstiness*), o que pode causar perda de pacotes Ethernet (Mostowfi e Christensen, 2011) e (Silva, 2018).



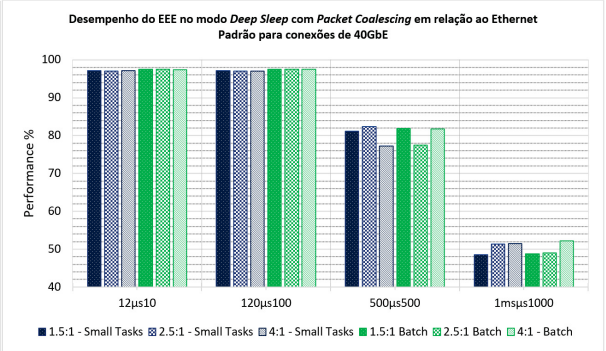

(a) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing* em relação ao Ethernet Padrão para conexões de 1GbE



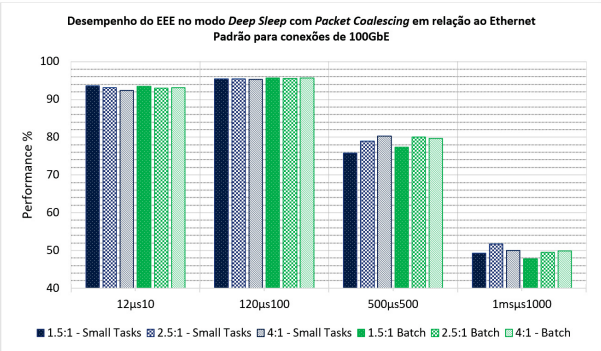

(b) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing* em relação ao Ethernet Padrão para conexões de 10GbE



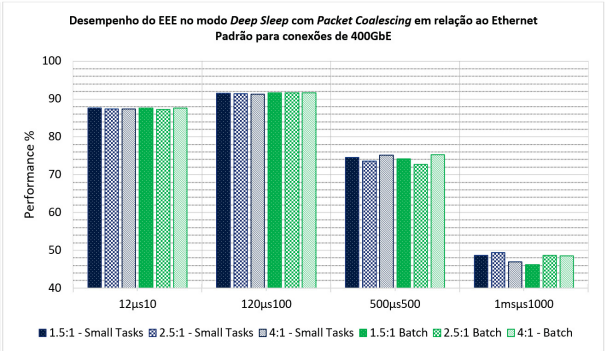

(c) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing* em relação ao Ethernet Padrão para conexões de 25GbE




(d) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing* em relação ao Ethernet Padrão para conexões de 40GbE



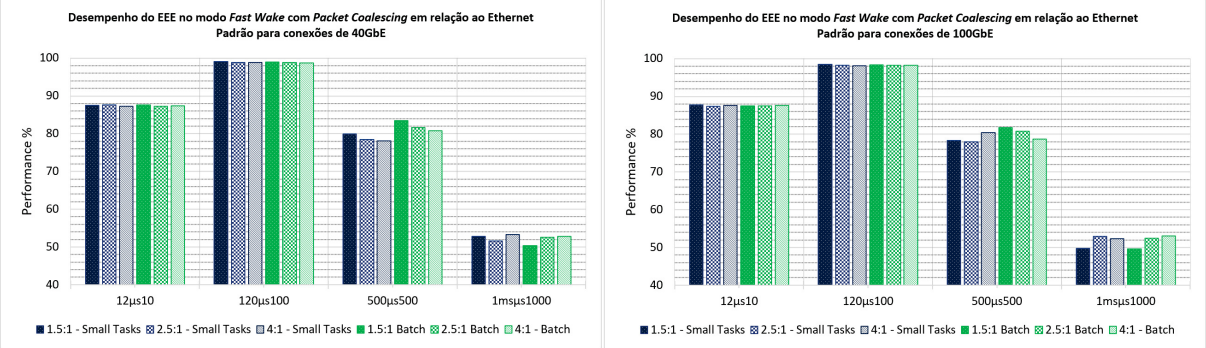

(e) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing* em relação ao Ethernet Padrão para conexões de 100GbE




(f) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing* em relação ao Ethernet Padrão para conexões de 400GbE

Figura 5.3: Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*

Para as conexões que suportam o EEE no modo *Fast Wake* a configuração 120$\mu$s100 tornou-se a mais crucial, sendo disparadamente a com maior desempenho. No caso das conexões de 40GbE, o desempenho manteve-se estável de 98.8% para 98.89%, enquanto em links de 400GbE foi de 98.2% para 98.28%. Ao utilizarmos configurações como 12$\mu$s10, o desempenho degradou-se para 87.466% e 87.52% respectivamente, só não sendo pior que o desempenho de




(a) Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing* em relação ao Ethernet Padrão para conexões de 40GbE

(b) Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing* em relação ao Ethernet Padrão para conexões de 100GbE

Figura 5.4: Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*

1ms1000, que atingiu valores de 52.228% e 51.649%. Como esperado, a performance dos *Batch Jobs* foram vagamente melhores.

Durante nossas simulações quatro diferentes configurações para o *Packet Coalescing* foram testadas, 12$\mu$s10, 120$\mu$s100, 500$\mu$s500 e 1ms1000. A primeira demonstrou-se ser de certa forma eficaz em manter a performance em conexões até 40GbE, entretanto ao irmos para conexões maiores como 100GbE e 400GbE tornou-se necessário a utilização da segunda configuração para manter o desempenho. Em contrapartida, a terceira e quarta configurações demonstraram uma degradação grande da performance, sendo assim descartadas. Ao avaliarmos o contexto de **economia de energia x performance**, optamos pela escolha da configuração 120$\mu$s100 para ser a padrão nos próximos testes que envolvem *Packet Coalescing* combinado com CoDel, RED e ECN.

## 5.2 ENERGY EFFICIENT ETHERNET COM PACKET COALESCING, CONTROLLED DELAY E EXPLICIT CONGESTION NOTIFICATION

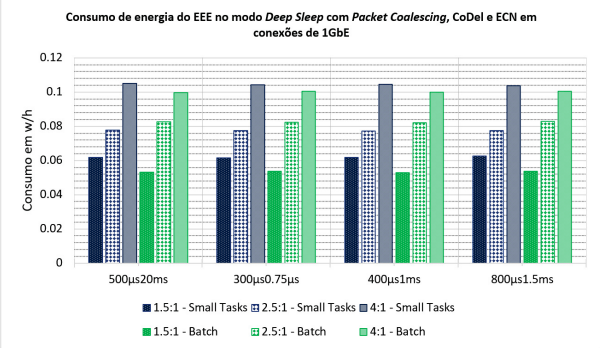
Na seção anterior demonstramos através dos resultados como a utilização de *Packet Coalescing* configurado corretamente e combinado com *Energy Efficient Ethernet* pode reduzir ainda mais o consumo de energia sem degradação agressiva da performance do *cluster*. Nesta seção apresentamos os resultados de nossas tentativas de combinar *Energy Efficient Ethernet*, *Packet Coalescing*, *Controlled Delay* e *Explicit Congestion Notification* para melhorar a performance. A metodologia detalhada das configurações pode ser observada no Capítulo 3.

### 5.2.1 Economia de Energia

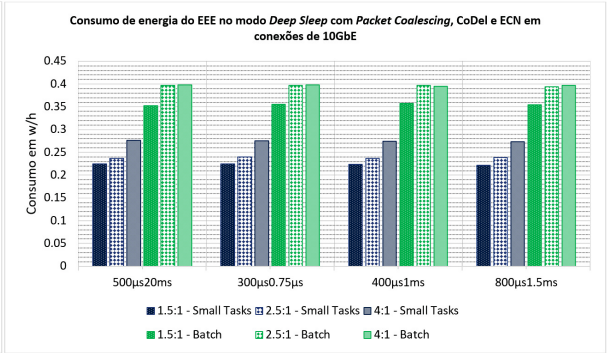
As figuras 5.5 e 5.6 ilustram os resultados das simulações. De forma geral a agregação do *Controlled Delay* não alterou o consumo de energia significativamente. Em conexões de 1GbE demonstrou-se mais uma vez ser desnecessário a implementação de recursos como *Packet Coalescing*, tendo uma redução de 16.11% (0.0805 w/h) para 15.95% (0.0798 w/h) em relação ao Ethernet padrão na melhor das configurações, 400$\mu$s1ms.

Nas conexões de 10GbE o melhor valor foi obtido na configuração 800$\mu$s1.5ms, com um consumo reduzido de 18.32% (0.4909 w/h) para 11.69% (0.3134 w/h) em relação ao Ethernet padrão. Para os links de 25GbE, o melhor consumo foi anotado ao usarmos 300$\mu$s0.75$\mu$s, reduzindo de 19.82% (1.0725 w/h) para 13.22% (0.715 w/h).

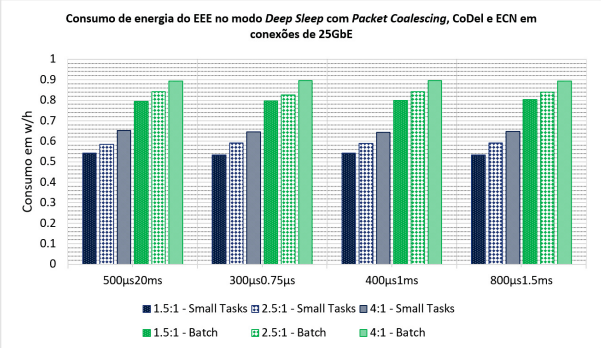
Em links de 40GbE todas as configurações se mantiveram com um mesmo padrão de consumo de energia, cerca de 13.1%, o que representa 0.98 w/h. Para os links de 100GbE



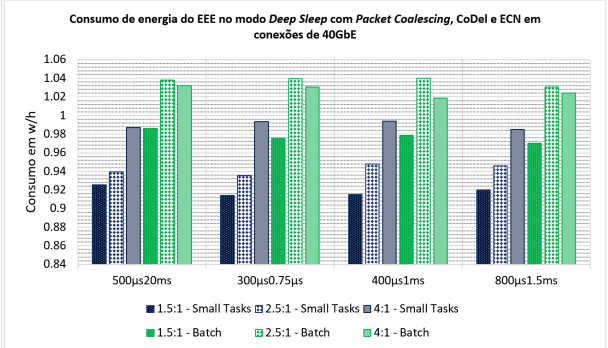

(a) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDeL e ECN em conexões de 1GbE



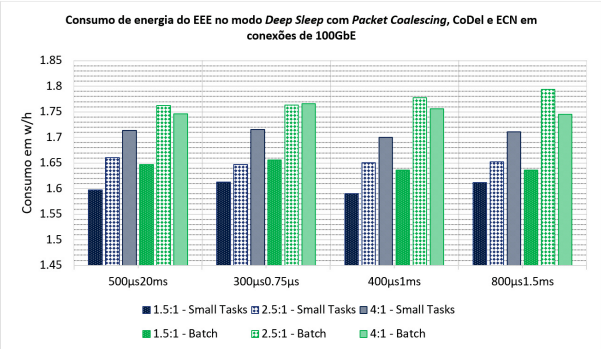

(b) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDeL e ECN em conexões de 10GbE



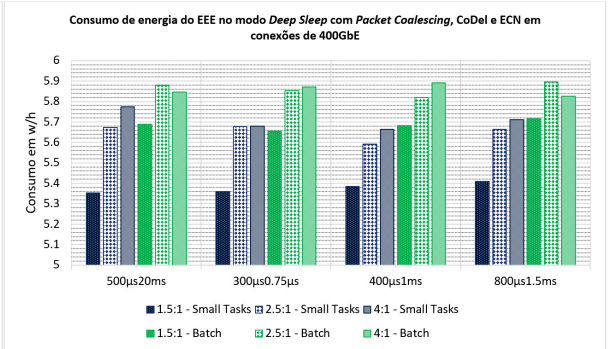

(c) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDeL e ECN em conexões de 25GbE




(d) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDeL e ECN em conexões de 40GbE




(e) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDeL e ECN em conexões de 100GbE



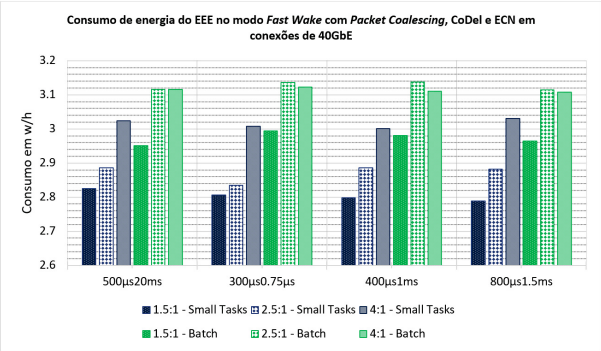

(f) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDeL e ECN em conexões de 400GbE

Figura 5.5: Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDeL e ECN

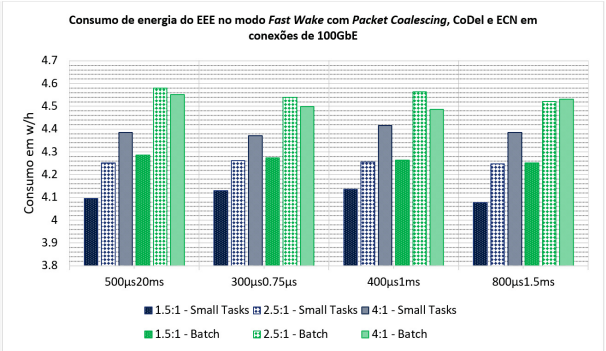
o melhor consumo foi obtido ao utilizar a configuração 400$\mu$s1ms, reduzindo o consumo de 24.31% (2.6469 w/h) para 15.58% (1.6853 w/h).

Por fim, para conexões de 400GbE a melhor configuração foi 400$\mu$s1ms, reduzindo o consumo de energia de 48.11% (10.6853 w/h) para 25.54% (5.6724 w/h). Em contrapartida, a pior configuração para esta configuração foi de 800$\mu$s1.5ms, atingindo um consumo de energia de 25.68% (5.7036 w/h).

Ao habilitarmos a tecnologia *Fast Wake* no EEE para conexões de 40GbE e 100GbE com *Packet Coalescing*, CoDel e ECN também foi possível notar leves diferenças no consumo de energia, como a redução de 61.87% (4.6340 w/h) e 63.51% (9.9160) para 39.8% (2.9814 w/h) e 39.81% (4.3357 w/h) em configurações de 800$\mu$s1.5ms.




(a) Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*, CoDeL e ECN em conexões de 40GbE




(b) Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*, CoDeL e ECN em conexões de 100GbE

Figura 5.6: Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*, CoDeL e ECN

Uma paridade entre os consumos de energia do EEE com *Packet Coalescing* e EEE com *Packet Coalescing*, CoDel e ECN já era esperada, uma vez que o foco principal deste algoritmo é melhorar o desempenho do *cluster*, e não a economia de energia em si.

### 5.2.2 Desempenho

Esta subseção contém os resultados da performance das simulações - fator principal para aplicar o *Controlled Delay* juntamente ao *Packet Coalescing* e *Explicit Congestion Notification*. Em conexões de 1GbE a melhor performance de 99.88% foi obtida ao utilizarmos a configuração 800$\mu$s1.5ms. Comparado ao Ethernet padrão, a performance deste link era de 99.78% com apenas o EEE. Já a pior performance foi encontrada ao usarmos a configuração 400$\mu$s1ms, chegando à 99.86%.

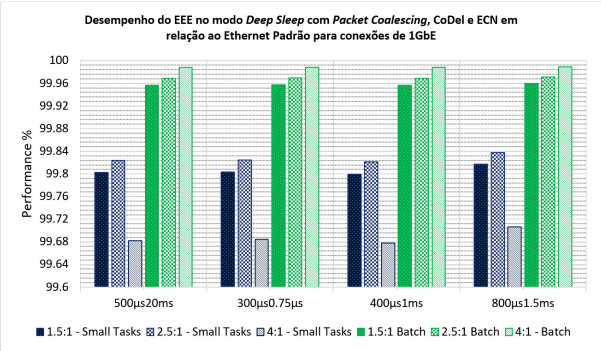
Para os links de 10GbE obtivemos uma média de performance melhor com a configuração 800$\mu$s1.5ms, aumentando a performance de 99.054% para 99.1535%. Por outro lado, o posto de pior configuração ficou com as demais configurações 500$\mu$s20ms, 300$\mu$s0.75$\mu$s e 400$\mu$s1m, que atingiram uma média de performance de 99.14%.

Conforme avançamos nos testes para conexões com uma maior banda ficou claro como uma boa configuração do CoDel com ECN impactava no desempenho do *cluster*. Para os links de 25GbE, o melhor desempenho de 98.673% foi alcançando ao usarmos a configuração 800$\mu$s1.5ms, e a pior de 98.549% ao usar como parâmetro 400$\mu$s1ms.

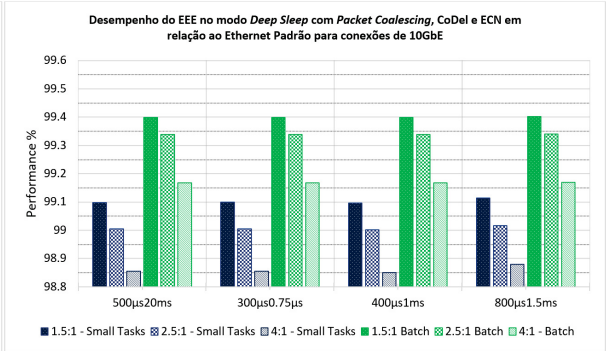
Nas simulações à partir de 40GbE já ficavam claras a melhor e pior configuração dentre as opções adotadas dentro da bibliografia disponível, 800$\mu$s1.5ms e 400$\mu$s1ms. Na conexão de 40GbE, o melhor parâmetro 800$\mu$s1.5ms foi responsável por elevar a performance de 97.147% para 97.521%, enquanto o pior parâmetro 400$\mu$s1ms atingiu 97.284%.

Ao considerarmos conexões maiores como 100GbE e 400GbE a diferença de performance entre os parâmetros se tornou ainda maior. No caso dos links de 100GbE a melhor configuração 800$\mu$s1.5ms obteve uma performance de 95.97%, contra 95.57% da pior 400$\mu$s1ms. A performance com EEE padrão para este link é de 95.44%. Em conexões de 400GbE a performance foi ligeiramente aumentada de 91.413% para 92.32% com o melhor parâmetro 800$\mu$s1.5ms, e manteve-se em 91.58% com 400$\mu$s1ms.

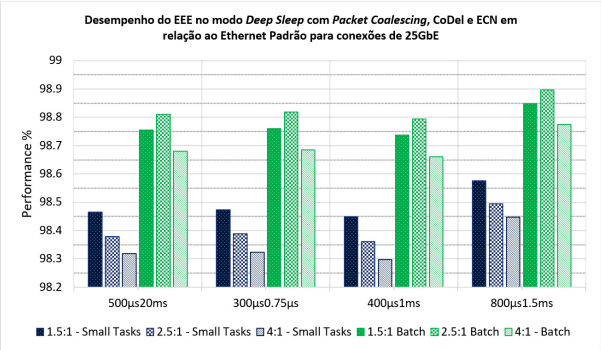
Com a tecnologia *Fast Wake* do EEE também foram realizadas simulações. Entretanto, notamos que ao adicionar o *Packet Coalescing* juntamente ao CoDel e ECN houve uma degradação da performance. Para conexões de 40GbE, mesmo com a nossa melhor configuração 800$\mu$s1.5ms a performance caiu para 95.962%, e ainda mais, para 95.577% quando utilizamos o pior parâmetro dentro das opções 400$\mu$s1ms.



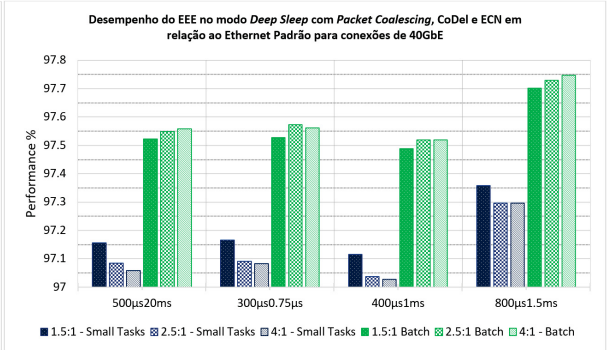

(a) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDel e ECN em relação ao Ethernet Padrão para conexões de 1GbE



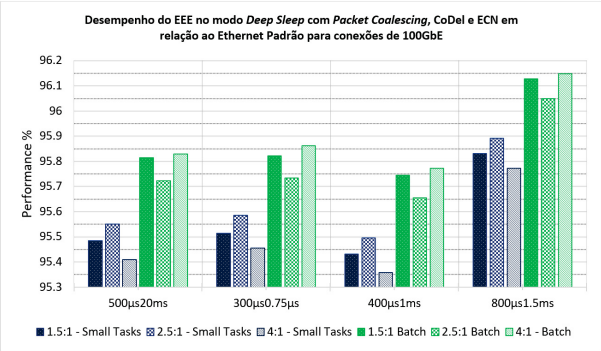

(b) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDel e ECN em relação ao Ethernet Padrão para conexões de 10GbE



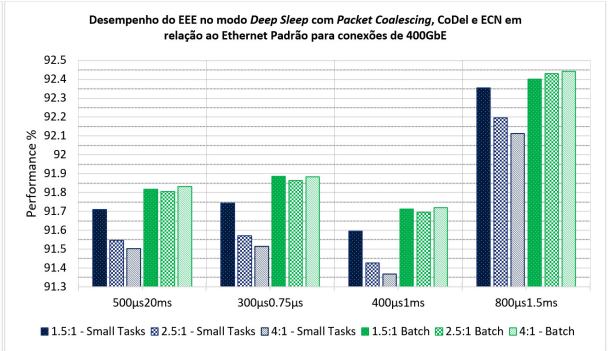

(c) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDel e ECN em relação ao Ethernet Padrão para conexões de 25GbE




(d) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDel e ECN em relação ao Ethernet Padrão para conexões de 40GbE




(e) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDel e ECN em relação ao Ethernet Padrão para conexões de 100GbE

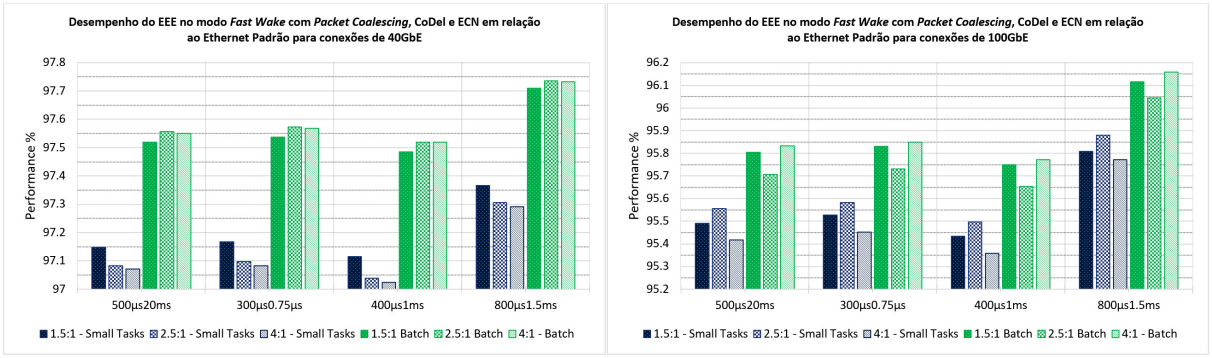



(f) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDel e ECN em relação ao Ethernet Padrão para conexões de 400GbE

Figura 5.7: Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, CoDel e ECN

Dentre os resultados obtidos, consideramos interessante a aplicação do *Packet Coalescing* com *Controlled Delay* e *Explicit Congestion Notification* em conexões EEE no modo *Deep Sleep* superiores à 40GbE, principalmente quando os parâmetros adequados para tais conexões são encontrados como foi o caso de 800$\mu$s1.5ms.

Ao utilizarmos o modo *Fast Wake* do EEE com *Packet Coalescing*, *Controlled Delay* e *Explicit Congestion Notification* houve uma degradação considerável da performance em relação ao EEE no modo *Fast Wake* com apenas *Packet Coalescing*. Isso se deve ao fato do *Controlled Delay* adicionar uma latência extra superior a própria latência do EEE no modo *Fast Wake* agravando a congestão do tráfego (Silva, 2018); (Silva e Carpenter, 2016); (Silva e Carpenter, 2019).




(a) Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*, CoDel e ECN em relação ao Ethernet Padrão para conexões de 40GbE

(b) Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*, CoDel e ECN em relação ao Ethernet Padrão para conexões de 100GbE

Figura 5.8: Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*, CoDel e ECN

## 5.3 ENERGY EFFICIENT ETHERNET COM PACKET COALESCING, RANDOM EARLY DETECTION E EXPLICIT CONGESTION NOTIFICATION

Esta seção apresenta os resultados das simulações em que combinamos *Packet Coalescing* com *Random Early Detection* e *Congestion Notification*, e investigamos como projetar e configurar conexões EEE para fornecer o máximo de economia de energia sem degradar o desempenho do *cluster*. Como mencionado anteriormente durante nossas simulações quatro diferentes configurações para o *Packet Coalescing* foram testadas, 12$\mu$s10, 120$\mu$s100, 500$\mu$s500 e 1ms1000, e optamos pela escolha da configuração 120$\mu$s100 para ser a padrão nos testes que envolvem *Packet Coalescing* combinado, RED e ECN por ser a que manteve a performance e forneceu uma melhor economia de energia para as conexões.

A metodologia detalhada das configurações pode ser observada no Capítulo 3. Consideramos *Random Early Detection* como o *Active Queue Management* selecionado para marcar pacotes com recurso *Explicit Congestion Notification* configurado em remetentes TCP, e definimos os valores baseados em trabalhos anteriores, como 70 $Min_{th}$ e 70 $Max_{th}$ em (Alizadeh et al., 2010); 125 $Min_{th}$ e 375 $Max_{th}$, 25 $Min_{th}$ e 75 $Max_{th}$, 50 $Min_{th}$ e 150 $Max_{th}$ em (Silva, 2018).

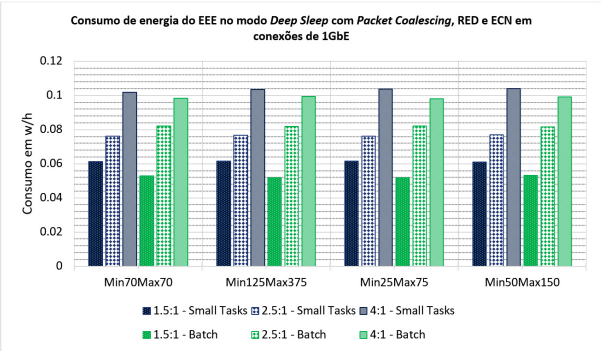
### 5.3.1 Economia de Energia

Os resultados das simulações de economia de energia com EEE nos modos *Deep Sleep* e *Fast Wake* combinados com RED e ECN podem ser observados nas figuras 5.9 e 5.10. Uma paridade envolvendo os resultados somente com *Packet Coalescing* e *Packet Coalescing* com RED era esperada, isso porque o RED é um algoritmo focado em ganho de desempenho.

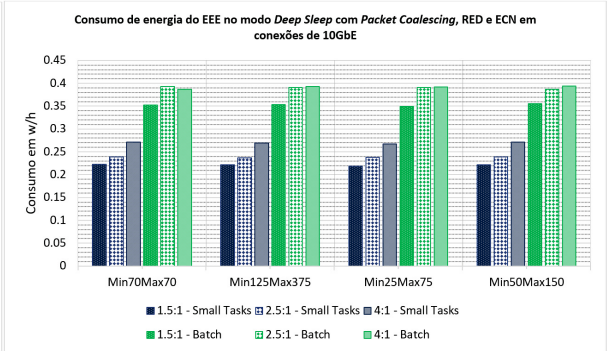
Como pode-se observar na figura 5.9, em links de 1GbE acontece uma pequena redução do consumo de energia em relação ao EEE padrão, de 16.11% (0.0805 w/h) para 15.76% (0.0788 w/h) ao usarmos a configuração do RED em 70$Min_{th}$70$Max_{th}$.

Para as conexões de 10GbE encontramos uma redução no consumo de energia de 18.32% (0.4909 w/h) para 11.53% (0.3091 w/h) ao aplicar a configuração 25$Min_{th}$75$Max_{th}$. A pior configuração para esta conexão 125$Min_{th}$375$Max_{th}$ forneceu um consumo de 11.60% (0.3110 w/h).

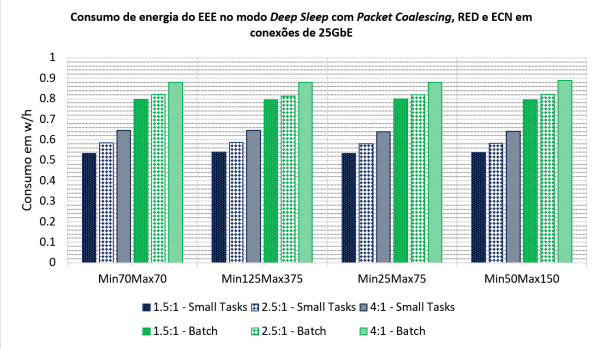
Em links de 25GbE ao aplicarmos o parâmetro do RED em 25$Min_{th}$75$Max_{th}$, o consumo de energia foi reduzido de 19.82% (1.0725 w/h) para 13.10% (0.7084 w/h). Em contrapartida o pior parâmetro 50$Min_{th}$150$Max_{th}$ consumiu 13.14% (0.7109 w/h).



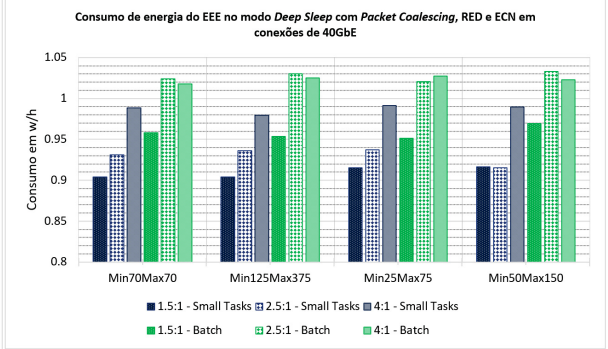

(a) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em conexões de 1GbE



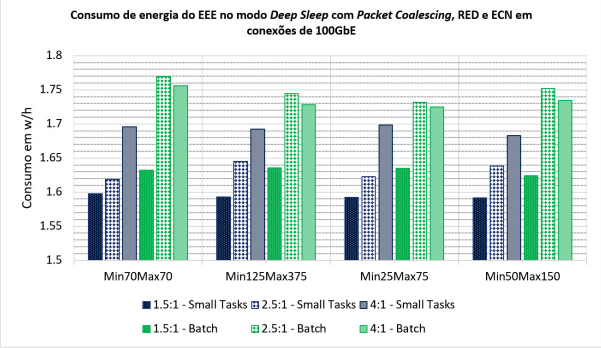

(b) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em conexões de 10GbE



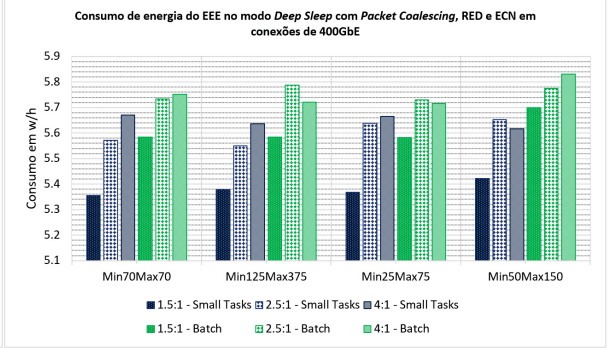

(c) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em conexões de 25GbE




(d) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em conexões de 40GbE




(e) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em conexões de 100GbE



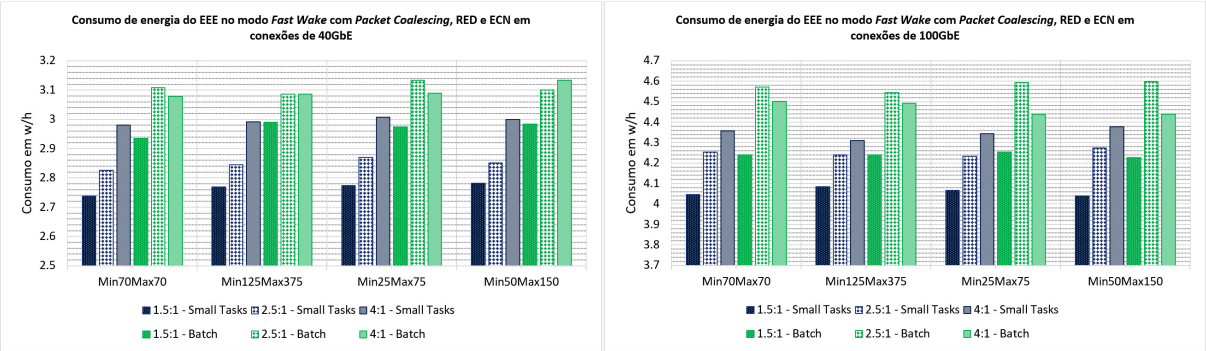

(f) Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em conexões de 400GbE

Figura 5.9: Consumo de energia do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN

Nas conexões de 40GbE o melhor consumo foi alcançado ao usarmos como parâmetro $70Min_{th}70Max_{th}$, o que representou uma redução de 19.86% (1.4877 w/h) para 12.96% (0.9706 w/h). O pior parâmetro neste caso, $50Min_{th}150Max_{th}$, consumiu 13.01% (0.9743 w/h).

Por fim, os links de 100GbE e 400GbE obtiveram uma melhor redução no consumo de energia ao utilizarmos as configurações $25Min_{th}75Max_{th}$ e $125Min_{th}375Max_{th}$, reduzindo o consumo de 24.31% (2.6469 w/h) para 15.31% (1.6675 w/h), e de 48.11% (10.6853 w/h) para 25.26% (5.6101 w/h) respectivamente. Em sua pior configuração $70Min_{th}70Max_{th}$ o link de 100GbE consumiu 15.41% (1.6782 w/h), quanto ao link de 400GbE, consumiu 25.51% (5.6663 w/h) ao aplicarmos o parâmetro $50Min_{th}150Max_{th}$.

Ao habilitarmos a tecnologia *Fast Wake* do EEE com *Packet Coalescing* e RED, assim como com *Packet Coalescing* e CoDel, houve leves diferenças no consumo de energia. Em sua melhor configuração $70Min_{th}70Max_{th}$ os links de 40GbE reduziram seu consumo de energia de




(a) Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*, RED e ECN em conexões de 40GbE

(b) Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*, RED e ECN em conexões de 100GbE

Figura 5.10: Consumo de energia do EEE no modo *Fast Wake* com *Packet Coalescing*, RED e ECN

61.87% (4.6340 w/h) para 39.31% (2.9442 w/h). Quanto a sua pior configuração $25Min_{th}75Max_{th}$, o consumo ainda foi reduzido significativamente para 39.72% (2.9751 w/h).

O link de 100GbE com *Fast Wake* por sua vez, reduziu o consumo de 63.51% (6.9160 w/h) para 39.65% (4.3182 w/h) em sua melhor configuração $125Min_{th}375Max_{th}$. Em seu pior parâmetro $70Min_{th}70Max_{th}$, o consumo foi reduzido para 39.75% (4.3284 w/h).

### 5.3.2 Desempenho

Esta subseção contém os resultados da performance das simulações - fator principal para aplicar o AQM *Random Early Detection* juntamente com *Packet Coalescing* e *Explicit Congestion Notification* ao EEE nas conexões Ethernet.

Através dos testes encontramos ganhos de performance consideráveis em todos os links, exceto 1GbE, em que o EEE padrão é o suficiente para uma ótima economia de energia com boa performance. Para os links de 1GbE, o RED foi capaz de aumentar a performance de 99.783% para 99.927% em sua melhor configuração $125Min_{th}375Max_{th}$. Por outro lado, sua pior configuração $70Min_{th}70Max_{th}$ degradou o desempenho para 99.163%.

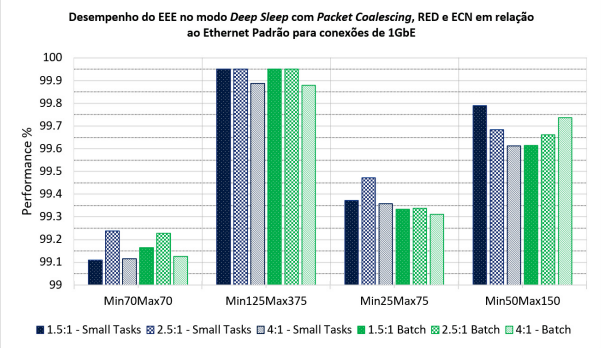
Em links de 10GbE, o melhor parâmetro do RED $125Min_{th}375Max_{th}$, aumentou a performance de 99.054% para 99.432%. Neste link o pior parâmetro $70Min_{th}70Max_{th}$ degradou o desempenho para 99.035%.

Quando aplicamos a melhor configuração do RED $125Min_{th}375Max_{th}$ nos links de 25GbE, houve um ganho de performance de 98.47% para 99.35%. Por outro lado, a pior configuração $70Min_{th}70Max_{th}$, elevou o desempenho para apenas 98.981%

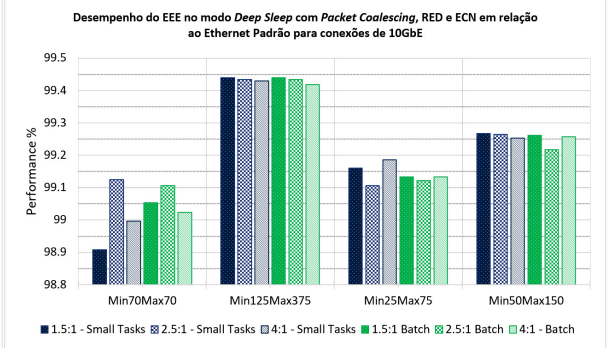
Em conexões de 40GbE houve um aumento da performance de 97.147% para 98.851% ao aplicarmos a melhor configuração do RED encontrada, $125Min_{th}375Max_{th}$. Por outro lado, a configuração $70Min_{th}70Max_{th}$ degradou o desempenho para 96.27%.

Nas conexões de 100GbE e 400GbE ficaram claros os impactos que são causados pelo RED quando bem ou mal configurado. Nos links de 100GbE, nossa melhor configuração $125Min_{th}375Max_{th}$ aumentou o desempenho de 95.447% para 98.659%, enquanto a pior $25Min_{th}75Max_{th}$, atingiu apenas 95.611% de desempenho. Em links de 400GbE por sua vez, a melhor configuração $125Min_{th}375Max_{th}$ elevou o desempenho para 98.239%, enquanto a pior $70Min_{th}70Max_{th}$, o degradou para 87.02%.

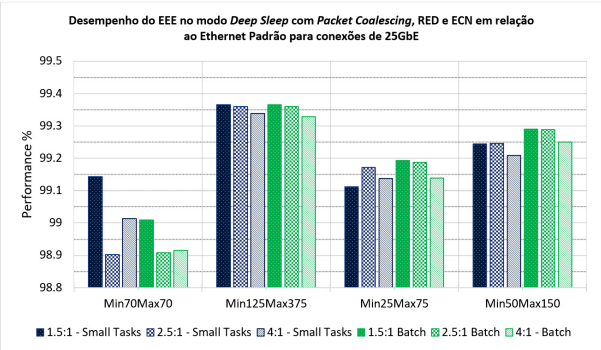
Com a tecnologia *Fast Wake* do EEE habilitada não houve ganhos de performance significativos para que justificasse a implementação do RED. Em sua melhor configuração nos links de 40 GbE $125Min_{th}375Max_{th}$, o desempenho foi elevado de 98.8% para 98.94%, e, em sua pior $25Min_{th}75Max_{th}$, degradou para 95.204%.



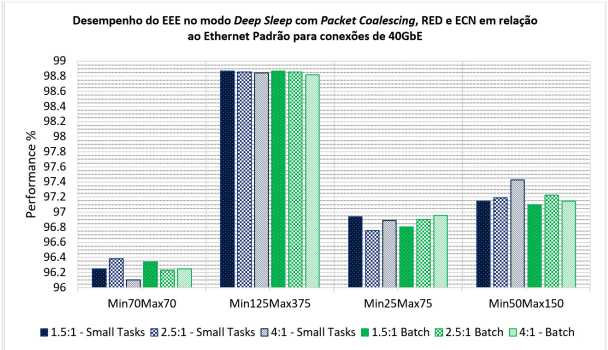

(a) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em relação ao Ethernet Padrão para conexões de 1GbE



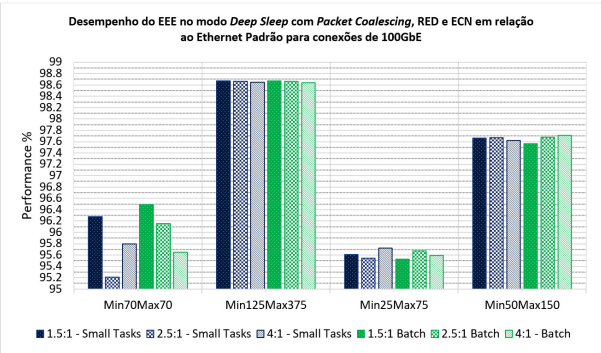

(b) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em relação ao Ethernet Padrão para conexões de 10GbE



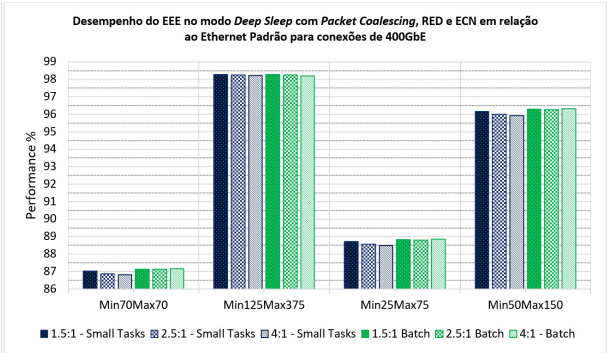

(c) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em relação ao Ethernet Padrão para conexões de 25GbE




(d) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em relação ao Ethernet Padrão para conexões de 40GbE

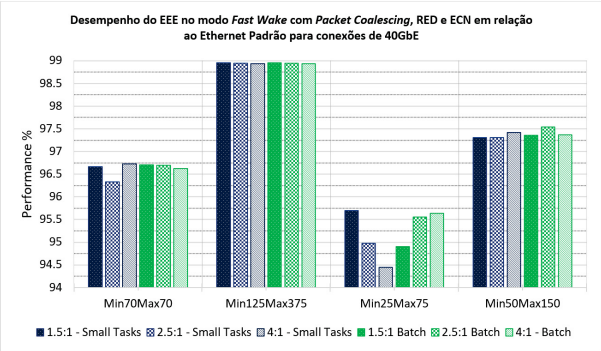



(e) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em relação ao Ethernet Padrão para conexões de 100GbE

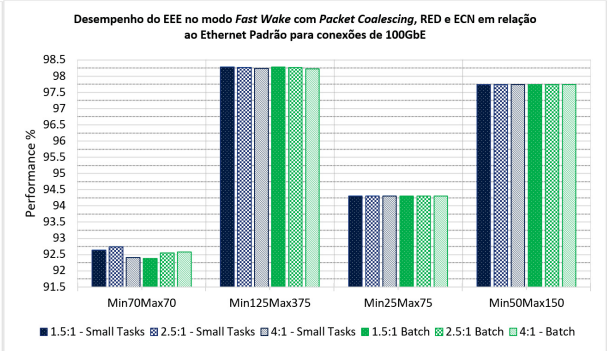



(f) Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN em relação ao Ethernet Padrão para conexões de 400GbE

Figura 5.11: Desempenho do EEE no modo *Deep Sleep* com *Packet Coalescing*, RED e ECN




(a) Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*, RED e ECN em relação ao Ethernet Padrão para conexões de 40GbE




(b) Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*, RED e ECN em relação ao Ethernet Padrão para conexões de 100GbE

Figura 5.12: Desempenho do EEE no modo *Fast Wake* com *Packet Coalescing*, RED e ECN

Quanto aos links de 100GbE com *Fast Wake*, o melhor parâmetro $125Min_{th}375Max_{th}$ aumentou o desempenho de 98.2% para 98.26%. Já em seu pior parâmetro $70Min_{th}70Max_{th}$, degradou para 92.55%.

Todos os resultados estão ilustrados de forma detalhada nas figuras 5.11 e 5.12. De forma geral o melhor parâmetro encontrado para o RED foi $125Min_{th}375Max_{th}$, enquanto $70Min_{th}70Max_{th}$ e $25Min_{th}75Max_{th}$ degradaram consideravelmente a performance dos links.

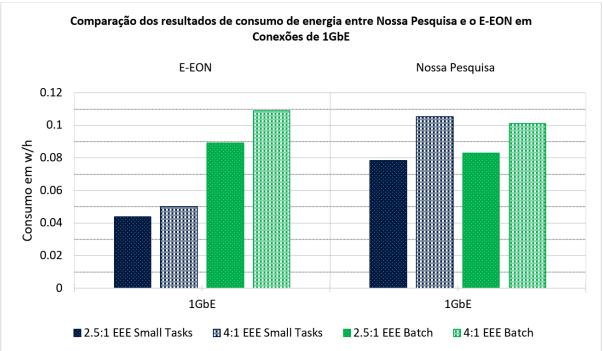
## 5.4 COMPARAÇÃO COM O E-EON

Esta seção abrange os conceitos do principal trabalho relacionado, o *Energy-Efficient and Optimized Networks for Hadoop*, e uma comparação com os resultados obtidos durante nossa pesquisa. O E-EON (Silva, 2018) é um conjunto de técnicas que podem ser utilizadas para reduzir o consumo de energia e latência da rede, e melhorar a taxa de transferência do *cluster* Hadoop. Além disso, tais técnicas não são exclusivas do Hadoop e podem ser aplicadas com benefícios semelhantes à qualquer outra carga de trabalho que se enquadre nas especificações do *MapReduce*.

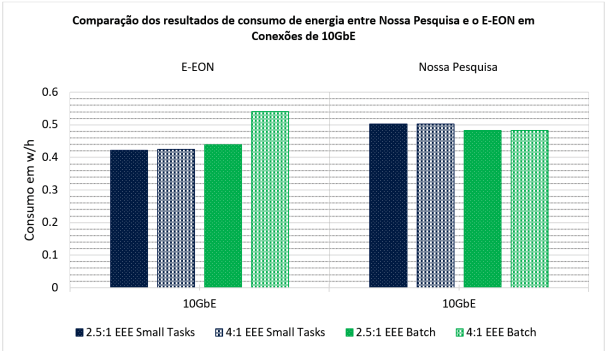
Este estudo foi o primeiro a avaliar o protocolo 802.2az *Energy Efficient Ethernet* em *clusters* Hadoop, e seus resultados demonstram que uma boa economia de energia pode ser obtida nas interconexões de folha (*Leaf Interconnections*), mas no nível de agregação (*Spines*) a coalescência de pacotes (*Packet Coalescing*) é necessária para reduzir o consumo de energia a um nível próximo do ideal. Na mesma pesquisa, aconteceu a primeira análise do *Active Queue Management* (AQM) e extensões de protocolo TCP que permitem o uso de ECN, como ECN autônomo e DCTCP. Os resultados apresentam uma redução da latência nas redes do Hadoop de 85%, mantendo a taxa de transferência do *cluster* em 95% da linha de base. No entanto, não é uma tarefa fácil configurar tais filas.

Outra contribuição foi a compreensão do por que AQMs são tão difíceis de configurar em *clusters* do Hadoop. Durante os testes foram identificados os motivos e caracterizado o padrão de tráfego que aumenta a latência e diminui a taxa de transferência de rede do *cluster*, e, posteriormente, dadas sugestões de modificações de como as filas de comutação tratam os pacotes marcados com ECN. Uma redução da latência de rede e, em alguns casos, uma melhora do rendimento em 10% foram alcançadas, e definido padrões de configurações.

Por fim, outros testes com conclusões residem na identificação das condições certas para combinar *Energy Efficient Ethernet* e *Packet Coalescing* com AQMs. A partir destes resultados é possível atingir um consumo de energia reduzido e redes de baixa latência.



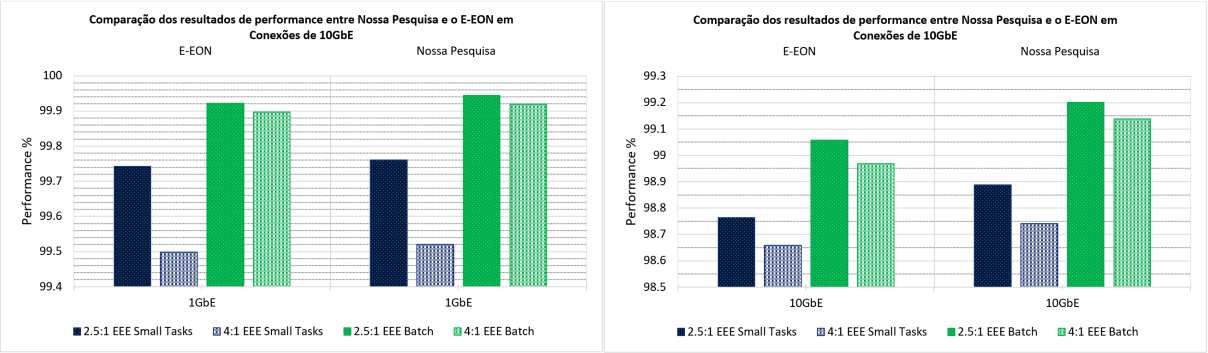

(a) Comparação dos resultados de consumo de energia entre Nossa Pesquisa e o E-EON em Conexões de 1GbE




(b) Comparação dos resultados de consumo de energia entre Nossa Pesquisa e o E-EON em Conexões de 10GbE

Figura 5.13: Comparação dos resultados de consumo de energia entre Nossa Pesquisa e o E-EON

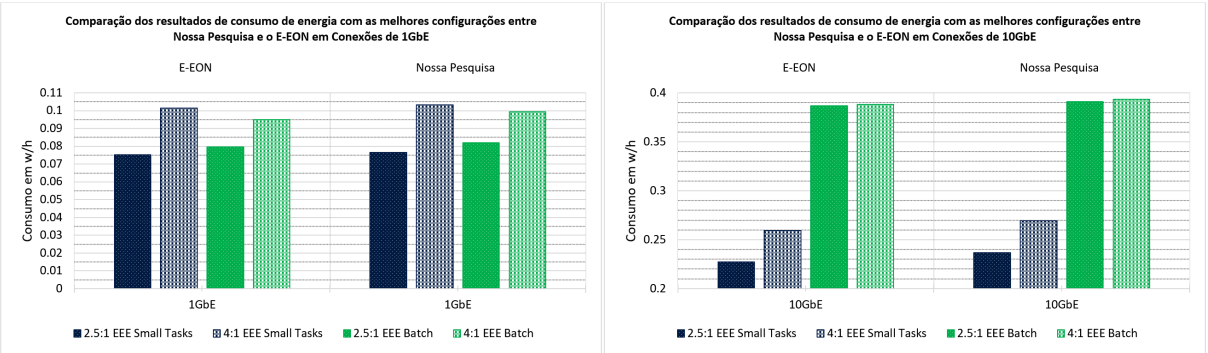
Os gráficos 5.13 e 5.14 expõem as leves diferenças entre os resultados de nossa pesquisa e o E-EON. Levando em consideração o desvio padrão, os resultados são tecnicamente iguais. Entretanto, a bibliografia disponível aponta que essas diferenças na verdade se devem a fatores como a arquitetura do Hadoop 1.x ser diferenciada do Hadoop 3.x com as introduções do YARN e EC (White, 2009); (White, 2015). Outra condição que pode contribuir é o fato de que o MRPerf (Wang et al., 2009) utilizado nas simulações do E-EON possuí limitações para medir com precisão algumas etapas da estrutura do *MapReduce*, especialmente nas fases de competição por recursos, tendo eficácia apenas na fase *Shuffle* (Liu et al., 2013), enquanto o NetSLS (Wette et al., 2015) mede com precisão todas as etapas de uma aplicação, incluindo a alocação de recursos e inicialização dos *Jobs* (Hadoop, 2020); (Wette et al., 2015).




(a) Comparação dos resultados de performance entre Nossa Pesquisa e o E-EON em Conexões de 1GbE

(b) Comparação dos resultados de performance entre Nossa Pesquisa e o E-EON em Conexões de 10GbE

Figura 5.14: Comparação dos resultados de performance entre Nossa Pesquisa e o E-EON



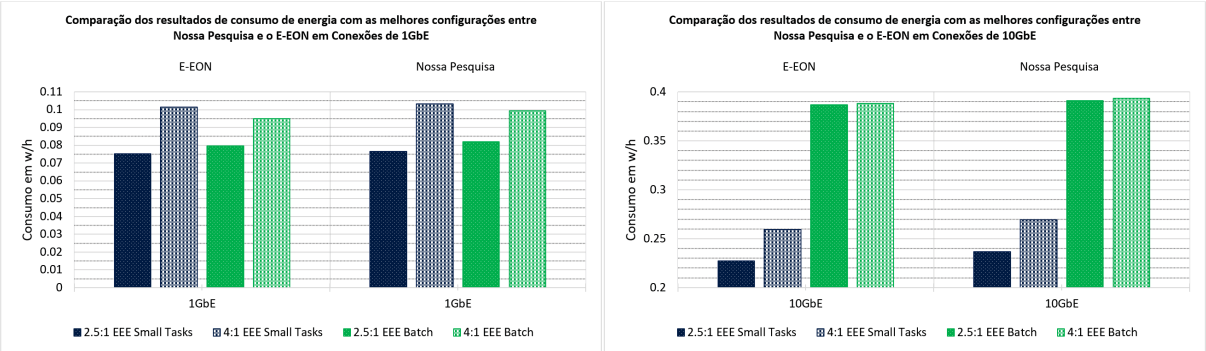

(a) Comparação dos resultados de consumo de energia com as melhores configurações entre Nossa Pesquisa e o E-EON em Conexões de 1GbE

(b) Comparação dos resultados de consumo de energia com as melhores configurações entre Nossa Pesquisa e o E-EON em Conexões de 10GbE

Figura 5.15: Comparação dos resultados de consumo de energia com as melhores configurações entre Nossa Pesquisa e o E-EON

As figuras 5.15 e 5.16 mostram os resultados de economia de energia e performance para o E-EON e também para nossa pesquisa. De modo geral, em ambos os trabalhos, as configurações do *Packet Coalescing* usando o parâmetro 120$\mu$s100, AQM *Random Early Detection* no modo *Instant Queue Length* com parâmetro 125$Min_{th}$375$Max_{th}$ e *Explicit Congestion Notification* no modo *Default* performaram melhor, oferecendo uma ótima economia de energia com uma perda mínima de desempenho.

Durante nossos testes não levamos em conta o DCTCP, uma vez que não houve resultados satisfatórios em (Silva, 2018). Por outro lado, para o EEE testamos os dois modos disponíveis, o




(a) Comparação dos resultados de performance com as melhores configurações entre Nossa Pesquisa e o E-EON em Conexões de 1GbE

(b) Comparação dos resultados de performance com as melhores configurações entre Nossa Pesquisa e o E-EON em Conexões de 10GbE

Figura 5.16: Comparação dos resultados de performance com as melhores configurações entre Nossa Pesquisa e o E-EON

*Deep Sleep* e o *Fast Wake*, enquanto no E-EON foi levado em conta apenas o modo *Deep Sleep*. Além da pesquisa feita no E-EON para conexões de 1GbE e 10GbE fomos adiante realizando simulações e oferecendo resultados de economia de energia e performance com o EEE habilitado para conexões de 25GbE, 40GbE, 100GbE e 400GbE. Em contrapartida, o E-EON procurou aumentar a largura de banda do *cluster* e reduzir a latência em conexões Ethernet comuns (sem o EEE), mas este não era o foco de nossa pesquisa.

## 5.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados estudos experimentais (simulações) com o objetivo de reduzir ainda mais o consumo de energia aliando o *Energy Efficient Ethernet* ao *Packet Coalescing*. Posteriormente aplicamos *Active Queue Managements* como *Controlled Delay* e *Random Early Detection* com *Explicit Congestion Notification* para recuperar a performance degradada pelo EEE e *Packet Coalescing*, obtendo assim redes com baixo consumo de energia e desempenho próximo ao Ethernet Padrão.

Durante nossos experimentos encontramos resultados que demonstram que quando configurado corretamente, com o melhor parâmetro 120$\mu$s100, o *Packet Coalescing* juntamente ao EEE proporciona uma redução do consumo de energia até 45% maior em relação ao EEE padrão.

Ao procurarmos por soluções de AQMs para reduzir a perda de desempenho apresentada pelo EEE e EEE com *Packet Coalescing*, o RED com a configuração $125Min_{th}375Max_{th}$ demonstrou ser 65% mais eficiente que o CoDel com todas as suas configurações 500$\mu$s20ms, 300$\mu$s0.75$\mu$s, 400$\mu$s1ms e 800$\mu$s1.5ms. Uma explicação para tal diferença é que o RED usando o modo IQL (*Instant Queue Length*) acaba sendo mais responsivo do que o cálculo dinâmico realizado pelo CoDel (Silva, 2018). Sendo assim, segue um sumário da melhor combinação que proporciona um **balanceamento entre economia de energia e desempenho** encontrada para cada conexão:

- **1GbE**: *Energy Efficient Ethernet* no modo *Deep Sleep*. Esta configuração proporciona uma economia de energia de 83.89% com uma perda de desempenho de apenas 0.217% em relação ao Ethernet Padrão;
- **10GbE**: *Energy Efficient Ethernet* no modo *Deep Sleep*, com *Packet Coalescing* usando o parâmetro 120$\mu$s100, AQM *Random Early Detection* no modo *Instant Queue Length*

com parâmetro 125$Min_{th}$375$Max_{th}$ e *Explicit Congestion Notification* no modo *Default*. Esta configuração proporciona uma economia de energia de 88.4% com uma perda de desempenho de 0.57% em relação ao Ethernet Padrão;

- **25GbE**: *Energy Efficient Ethernet* no modo *Deep Sleep*, com *Packet Coalescing* usando o parâmetro 120$\mu$s100, AQM *Random Early Detection* no modo *Instant Queue Length* com parâmetro 125$Min_{th}$375$Max_{th}$ e *Explicit Congestion Notification* no modo *Default*. Esta configuração proporciona uma economia de energia de 86.88% com uma perda de desempenho de 0.65% em relação ao Ethernet Padrão;

- **40GbE**: *Energy Efficient Ethernet* no modo *Deep Sleep*, com *Packet Coalescing* usando o parâmetro 120$\mu$s100, AQM *Random Early Detection* no modo *Instant Queue Length* com parâmetro 125$Min_{th}$375$Max_{th}$ e *Explicit Congestion Notification* no modo *Default*. Esta configuração proporciona uma economia de energia de 87.03% com uma perda de desempenho de 1.15% em relação ao Ethernet Padrão;

- **100GbE**: *Energy Efficient Ethernet* no modo *Deep Sleep*, com *Packet Coalescing* usando o parâmetro 120$\mu$s100, AQM *Random Early Detection* no modo *Instant Queue Length* com parâmetro 125$Min_{th}$375$Max_{th}$ e *Explicit Congestion Notification* no modo *Default*. Esta configuração proporciona uma economia de energia de 84.64% com uma perda de desempenho de 1.342% em relação ao Ethernet Padrão;

- **400GbE**: *Energy Efficient Ethernet* no modo *Deep Sleep*, com *Packet Coalescing* usando o parâmetro 120$\mu$s100, AQM *Random Early Detection* no modo *Instant Queue Length* com parâmetro 125$Min_{th}$375$Max_{th}$ e *Explicit Congestion Notification* no modo *Default*. Esta configuração proporciona uma economia de energia de 74.73% com uma perda de desempenho de 1.765% em relação ao Ethernet Padrão;

No modo *Fast Wake* do *Energy Efficient Ethernet* os melhores resultados que fornecem balanceamento entre economia de energia e desempenho foram encontrados ao utilizar apenas a tecnologia do *Packet Coalescing* com o parâmetro 120$\mu$s100. Esta configuração proporciona uma economia de energia nos links de 40GbE e 100GbE de 59.7% e 59.63% com uma perda de desempenho de 1.11% e 1.72% em relação ao Ethernet Padrão respectivamente.

Por fim, avaliamos assim como (Herreria-Alonso et al., 2017) que o uso do modo *Fast Wake* do EEE se torna útil para administradores com baixo conhecimento em infraestrutura, pois de modo geral, proporciona um desempenho similar ao EEE no modo *Deep Sleep* configurado com *Packet Coalescing*, RED e ECN, porém, com um consumo de energia maior.

# 6 CONCLUSÃO

Um dos principais problemas da infraestrutura de processamento de *big data* é o consumo de energia, que facilmente atinge a cifra de bilhões de kWh consumidos por ano (Raizada et al., 2020). Em 2006, os *data centers* dos EUA consumiram 3 bilhões de kWh (Mahadevan et al., 2011), enquanto em 2019 os *data centers* globais atingiram 360 bilhões de kWh consumidos (Raizada et al., 2020). As previsões indicam que o consumo de energia pode chegar a 3.000 bilhões de kWh em 2030 (Raizada et al., 2020).

Estudos mostram que os custos com energia tornaram-se maiores do que os com *hardware*, respondendo por 75% do custo total dos *data centers* (Belady, 2007). Em média 10% a 50% do consumo de energia do *data center* é por DCNs (Abts et al., 2010), e esse número tende a ser maior quando recursos como CPU e memória não são totalmente utilizados. Os principais dispositivos consumidores de energia das redes são os *switches*, consumindo cerca de 30% de toda a energia do *data center* (Kliazovich et al., 2012).

Os *switches* que interligam os links são responsáveis por 65% do consumo total de energia da rede, e não cessam o consumo mesmo quando não há tráfego na rede (Kliazovich et al., 2012). Conforme relatado em (Mahadevan et al., 2009), é possível reduzir esse "desperdício" de energia.

O padrão *Energy Efficient Ethernet* (EEE) visa reduzir o consumo de energia das conexões Ethernet configurando um modo de hibernação de link conhecido como *Low Power Idle*. Esta dissertação foi a primeira a testar e analisar o impacto do EEE em links de 1GbE, 10GbE, 25GbE, 40GbE, 100GbE e 400GbE na versão atual do *Hadoop MapReduce*, 3.x. Atualmente o Apache Hadoop (Hadoop, 2022) - uma implementação de código aberto do *MapReduce* (Dean e Ghemawat, 2004) - é uma das ferramentas de *big data* mais utilizadas para armazenamento e processamento de dados no mundo.

Ao aplicarmos o EEE nos links, encontramos perdas significativas de desempenho para conexões de 100GbE e 400GbE. Estas perdas ocorreram porque em determinados momentos é necessário acordar o link para transmitir um único *frame*, o que causa penalidades de consumo de energia e desempenho em termos relativos que são agravadas de acordo com a largura de banda (Jiang et al., 2021); (Reviriego et al., 2009); (Reviriego et al., 2010); (Silva e Carpenter, 2017a). Para contornar tal problema, aplicamos a técnica conhecida como *Packet Coalescing*, um agrupamento de pacotes como forma de limitar o número de interrupções de recebimento, juntamente a dois algoritmos de *Active Queue Management*, o *Controlled Delay* e o *Random Early Detection* para evitar o congestionamento da conexão, e por fim, adicionamos o *Explicit Congestion Notification* para evitar o descarte de pacotes TCP.

Para tanto, avaliamos a execução de *Small Tasks* e *Batch Jobs* com todas essas configurações em diversos *clusters* simulados e encontramos as maiores economias de energia ao utilizarmos *Energy Efficient Ethernet*, com *Packet Coalescing*, *Random Early Detection* e *Explicit Congestion Notification* entre 74% e 84% para links de até 400GbE sem perda considerável de desempenho, entre 0,21% e 1.77%. Com esses resultados[1], é possível argumentar que há uma economia de energia significativa com pouca ou nenhuma perda de desempenho para conexões de até 400GbE, o que é interessante se analisarmos em termos de custo-benefício.

[1]Todos os resultados estão disponíveis em https://github.com/holyblade/MestradoUFPR-Jorgi.

## 6.1 TRABALHOS FUTUROS

Esta dissertação apresenta contribuições para profissionais que realizam pesquisas com *Green Networking* e trabalham na área de *Big Data*. Como trabalhos futuros, pretende-se continuar testando diferentes tamanhos de *cluster*, com outras taxas de assinatura excessiva e topologias, como por exemplo a tradicional hierárquica *fat-tree* com rede de três camadas. Por fim, aplicar todas as configurações recomendadas para balanceamento entre economia de energia e desempenho em outros *frameworks* que seguem o modelo *MapReduce*.

Uma vez que as conexões de 50GbE e 200GbE ainda não foram lançadas pelas Juniper (Networks, 2021), também há possibilidades de considerar essas larguras de banda em pesquisas futuras.

## 6.2 PUBLICAÇÃO DOS RESULTADOS

Os resultados do Capítulo 4 foram pulicados na ***2022 IEEE International Conference on Big Data and Smart Computing (BigComp)***[2] na modalidade *regular paper* (Dias et al., 2022). Para eventos futuros pretendemos submeter um novo artigo apresentando os resultados do Capítulo 5.

[2]http://bigcomputing.org/.

# REFERÊNCIAS


Abts, D., Marty, M. R., Wells, P. M., Klausler, P. e Liu, H. (2010). Energy proportional datacenter networks. Em *Proceedings of the 37th annual international symposium on Computer architecture*, páginas 338–347.

Adams, R. (2012). Active queue management: A survey. *IEEE communications surveys & tutorials*, 15(3):1425–1476.

Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S. e Sridharan, M. (2010). Data center tcp (dctcp). Em *Proceedings of the ACM SIGCOMM 2010 Conference*, páginas 63–74.

Bechtolsheim, A., Dale, L., Holbrook, H. e Li, A. (2011). Why big data needs big buffer switches. arista white paper. *Tech. Rep.*

Belady, C. L. (2007). In the data center, power and cooling costs more than the it equipment it supports. *Electronics cooling*, 13(1):24.

Bennett, M. J. e Diab, W. W. (2012). An overview of energy-efficient ethernet. `https://grouper.ieee.org/groups/802/3/RTPGE/public/may12/bennett_diab_01_0512.pdf`. Acessado em 15/12/2020.

Brenner, M. e Subrahmanyan, M. G. (1988). A simple formula to compute the implied standard deviation. *Financial Analysts Journal*, 44(5):80–83.

Chen, Y., Alspaugh, S. e Katz, R. (2012). Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *arXiv preprint arXiv:1208.4174*.

Chen, Y., Ganapathi, A., Griffith, R. e Katz, R. (2011). The case for evaluating mapreduce performance using workload suites. Em *2011 IEEE 19th annual international symposium on modelling, analysis, and simulation of computer and telecommunication systems*, páginas 390–399. IEEE.

Christensen, K., Reviriego, P., Nordman, B., Bennett, M., Mostowfi, M. e Maestro, J. A. (2010). Ieee 802.3 az: the road to energy efficient ethernet. *IEEE Communications Magazine*, 48(11):50–56.

Chung, J. e Claypool, M. (2002). Analysis of red-family active queue management over a wide-range of tcp loads. *Worcester Polytechnic Institute, Technical Report WPI-CS-TR-02-05*.

Chung, J. e Claypool, M. (2003). Analysis of active queue management. Em *Second IEEE International Symposium on Network Computing and Applications, 2003. NCA 2003.*, páginas 359–366. IEEE.

Cisco, C. I. N. (2011). Big data in the enterprise: Network design considerations. *Cisco Systems*.

Cisco, C. I. N. (2012). Cisco's massively scalable data center. *Cisco Systems*.

Cisco, C. I. N. (2014). Forecast and methodology, 2013–2018. *Cisco Systems*.

Cisco, C. I. N. (2016). Cisco data center spine-and-leaf architecture: Design overview white paper. *Cisco Systems*.

Cisco, C. I. N. (2018). Forecast and methodology, 2016–2021 white paper. *Cisco Systems*, 1.

Cisco, C. I. N. (2020). Cisco annual internet report (2018–2023) white paper.

De La Oliva, A., Hernández, T. R. V., Guerri, J. C., Hernández, J. A. e Reviriego, P. (2013). Performance analysis of energy efficient ethernet on video streaming servers. *Computer Networks*, 57(3):599–608.

Dean, J. e Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. *Google, Inc.*

Dell, T. (2020). Solução de problemas com eficiência energética ethernet (eee) ou green ethernet. `https://www.dell.com/support/kbdoc/pt-br/000134483/solu%C3%A7%C3%A3o-de-problemas-com-efici%C3%AAncia-energ%C3%A9tica-ethernet-eee-ou-green-ethernet`. Acessado em 21/02/2021.

Dias, J. L. B., De Almeida, L. B. e Albini, L. C. P. (2022). Reducing hadoop 3. x energy consumption through energy efficient ethernet. Em *2022 IEEE International Conference on Big Data and Smart Computing (BigComp)*, páginas 9–14. IEEE.

Ethernet, E. E. (2011). Ieee 802.3 az energy efficient ethernet: Build greener networks. *Tech. Rep.*

Floyd, S. e Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*, 1(4):397–413.

Floyd, S. e Ramakrishnan, K. (2000). The addition of explicit congestion notification (ecn) to ip. *Transport*.

Gantz, J. e Reinsel, D. (2012). The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16.

Georgakoudis, G., Jain, N., Ono, T., Inoue, K., Miwa, S. e Bhatele, A. (2019). Evaluating the impact of energy efficient networks on hpc workloads. Em *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, páginas 301–310. IEEE.

Getty, J. (2021). Technical introduction to bufferbloat. `https://www.bufferbloat.net/projects/bloat/wiki/`. Acessado em 15/12/2021.

Glushkova, D., Jovanovic, P. e Abelló, A. (2019). Mapreduce performance model for hadoop 2. x. *Information Systems*, 79:32–43.

Goldman, A., Kon, F., Junior, F. P., Polato, I. e de Fátima Pereira, R. (2012). Apache hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. *XXXI Jornadas de atualizaçoes em informatica*, páginas 88–136.

Gray, J. (2004). Distributed computing economics. Em *Computer Systems*, páginas 93–101. Springer.

Hadoop, A. (2017). Hadoop: Release 3.0.0 generally available. https://hadoop.apache.org/release/3.0.0.html. Acessado em 09/12/2020.

Hadoop, A. (2020). Yarn scheduler load simulator (sls). https://hadoop.apache.org/docs/current/hadoop-sls/SchedulerLoadSimulator.html. Acessado em 17/08/2020.

Hadoop, A. (2021). Hdfs high availability. https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithNFS.html. Acessado em 24/07/2021.

Hadoop, A. (2022). The apache™ hadoop® project develops open-source software for reliable, scalable, distributed computing. https://hadoop.apache.org/. Acessado em 01/02/2022.

Herrerı, S., Rodrı, M., Fernández-Veiga, M., López-Garcı, C. et al. (2012). Optimal configuration of energy-efficient ethernet. *Computer Networks*, 56(10):2456–2467.

Herreria-Alonso, S., Rodriguez-Perez, M., Fernandez-Veiga, M. e Lopez-Garcia, C. (2017). Optimizing dual-mode eee interfaces: Deep-sleep is healthy. *IEEE Transactions on Communications*, 65(8):3374–3385.

Hortonworks (2021). Cluster planning guide. http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.4/bk_cluster-planning/content/hardware-for-slave.1.html. Acessado em 22/07/2021.

IEEE (2010). Ieee standard for information technology– local and metropolitan area networks–specific requirements– part 3: Csma/cd access method and physical layer specifications amendment 5: Media access control parameters, physical layers, and management parameters for energy-efficient ethernet. *IEEE Std 802.3az-2010 (Amendment to IEEE Std 802.3-2008)*, páginas 1–302.

IEEE (2014). Ieee standard for ethernet amendment 2: Physical layer specifications and management parameters for 100 gb/s operation over backplanes and copper cables. *IEEE Std 802.3bj-2014 (Amendment to IEEE Std 802.3-2012 as amended by IEEE Std 802.3bk-2013)*, páginas 1–368.

Ismail, S., Pirzada, H. e Qazi, I. (2010). On the effectiveness of codel in data centers. *Tech. Rep.*, páginas 3–21.

Jiang, W., Liao, K., Yan, Y., Cheng, W., Zhang, L. e Wang, J. (2021). Modeling and analysis of latency distribution in the 40-100gbps dual-mode energy efficient ethernet. *IEEE Transactions on Green Communications and Networking*.

Kliazovich, D., Bouvry, P. e Khan, S. U. (2012). Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283.

Koomey, J. et al. (2011). Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*, 9(2011):161.

Kühlewind, M., Neuner, S. e Trammell, B. (2013). On the state of ecn and tcp options on the internet. Em *International Conference on Passive and Active Network Measurement*, páginas 135–144. Springer.

Liu, Y., Li, M., Alham, N. K. e Hammoud, S. (2013). Hsim: a mapreduce simulator in enabling cloud computing. *Future Generation Computer Systems*, 29(1):300–308.

Lochin, E. e Talavera, B. (2011). Managing internet routers congested links with a kohonen-red queue. *Engineering Applications of Artificial Intelligence*, 24(1):77–86.

Mahadevan, P., Banerjee, S., Sharma, P., Shah, A. e Ranganathan, P. (2011). On energy efficiency for enterprise and data center networks. *IEEE Communications Magazine*, 49(8):94–100.

Mahadevan, P., Sharma, P., Banerjee, S. e Ranganathan, P. (2009). Energy aware network operations. Em *IEEE INFOCOM Workshops 2009*, páginas 1–6. IEEE.

Makineni, S., Iyer, R., Minturn, D., Sen, S., Newell, D. e Zhao, L. (2020). Packet coalescing. US Patent 10,652,147.

Małowidzki, M. (2003). Simulation-based study of ecn performance in red networks.

Masur, R. G. e Mcintosh, S. K. (2017). Preliminary performance analysis of hadoop 3.0. 0-alpha3. Em *2017 New York Scientific Data Summit (NYSDS)*, páginas 1–3. IEEE.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. e Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2):69–74.

Mostowfi, M. e Christensen, K. (2011). Saving energy in lan switches: New methods of packet coalescing for energy efficient ethernet. Em *2011 International Green Computing Conference and Workshops*, páginas 1–8. IEEE.

Nelson, W. (2017). Introduction to spine-leaf networking designs. `https://lenovopress.com/lp0573.pdf`. Acessado em 01/01/2021.

Networks, E. (2014). Extreme networks: Big data - a solutions guide. `https://docplayer.net/1630876-Extreme-networks-big-data-a-solutions-guide.html`. Acessado em 01/01/2021.

Networks, J. (2021). Qfx5220 switch hardware guide. `https://www.juniper.net/documentation/us/en/hardware/qfx5220/qfx5220.pdf`. Acessado em 24/04/2021.

Nichols, K. e Jacobson, V. (2012). Controlling queue delay. *Communications of the ACM*, 55(7):42–50.

Noll, M. (2011). Benchmarking and stress testing an hadoop cluster with terasort, testdfsio & co. *Online: http://www. michael-noll. com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoopcluster-with-terasort-testdfsio-nnbench-mrbench*.

Peck, R., Olsen, C. e Devore, J. L. (2015). *Introduction to statistics and data analysis*. Cengage Learning.

Plackett, R. L. (1958). Studies in the history of probability and statistics: Vii. the principle of the arithmetic mean. *Biometrika*, 45(1-2):130–135.

Raizada, A., Pal Singh, K. e Sajid, M. (2020). Worldwide energy consumption of hyperscale data centers: A survey. *International Research Journal on Advanced Science Hub*, 2:8–15.

Reviriego, P., Christensen, K., Sánchez-Macián, A. e Maestro, J. A. (2011). Using coordinated transmission with energy efficient ethernet. Em *International Conference on Research in Networking*, páginas 160–171. Springer.

Reviriego, P., Hernadez, J.-A., Larrabeiti, D. e Maestro, J. A. (2010). Burst transmission for energy-efficient ethernet. *IEEE Internet Computing*, 14(4):50–57.

Reviriego, P., Hernández, J. A., Larrabeiti, D. e Maestro, J. A. (2009). Performance evaluation of energy efficient ethernet. *IEEE Communications Letters*, 13(9):697–699.

Rosolen, V., Bonaventure, O. e Leduc, G. (1999). A red discard strategy for atm networks and its performance evaluation with tcp/ip traffic. *ACM SIGCOMM Computer Communication Review*, 29(3):23–43.

Shao, Y., Li, C., Dong, W. e Liu, Y. (2016). Energy-aware dynamic resource allocation on hadoop yarn cluster. Em *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, páginas 364–371. IEEE.

Sharma, G. e Ganpati, A. (2015). Performance evaluation of fair and capacity scheduling in hadoop yarn. Em *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, páginas 904–906. IEEE.

Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I. e Lintner, W. (2016). United states data center energy usage report. Relatório técnico, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States).

Silva, R. F. (2018). *E-eon: energy-efficient and optimized networks for hadoop*. Tese de doutorado, Universitat Politècnica de Catalunya (UPC).

Silva, R. F. e Carpenter, P. M. (2015). Exploring interconnect energy savings under east-west traffic pattern of mapreduce clusters. Em *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, páginas 10–18. IEEE.

Silva, R. F. e Carpenter, P. M. (2016). Controlling network latency in mixed hadoop clusters: Do we need active queue management? Em *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, páginas 415–423. IEEE.

Silva, R. F. e Carpenter, P. M. (2017a). Energy efficient ethernet on mapreduce clusters: Packet coalescing to improve 10gbe links. *IEEE/ACM Transactions on Networking*, 25(5):2731–2742.

Silva, R. F. e Carpenter, P. M. (2019). Tcp proactive congestion control for east-west traffic: The marking threshold. *Computer Networks*, 151:1–11.

Silva, R. F. E. e Carpenter, P. M. (2017b). Interconnect energy savings and lower latency networks in hadoop clusters: The missing link. Em *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, páginas 514–517. IEEE.

Solnushkin, K. (2013). Fat-tree design. `https://clusterdesign.org/fat-trees/`. Acessado em 01/01/2021.

Taurion, C. (2009). *Cloud computing-computação em nuvem*. Brasport.

Testa, F. e Pavesi, L. (2017). *Optical switching in next generation data centers*. Springer.

Vahdat, A., Al-Fares, M. e Loukissas, A. (2013). Scalable commodity data center network architecture. US Patent 8,483,096.

Wang, G., Butt, A. R., Pandey, P. e Gupta, K. (2009). Using realistic simulation for performance analysis of mapreduce setups. Em *Proceedings of the 1st ACM workshop on Large-Scale system and application performance*, páginas 19–26.

Wette, P., Dräxler, M., Schwabe, A., Wallaschek, F., Zahraee, M. H. e Karl, H. (2014). Maxinet: Distributed emulation of software-defined networks. Em *2014 IFIP Networking Conference*, páginas 1–9. IEEE.

Wette, P. e Karl, H. (2014). Dct2gen: A versatile tcp traffic generator for data centers. *arXiv preprint arXiv:1409.2246*.

Wette, P., Schwabe, A., Splietker, M. e Karl, H. (2015). Extending hadoop's yarn scheduler load simulator with a highly realistic network & traffic model. Em *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, páginas 1–2. IEEE.

White, T. (2009). *Hadoop The Definitive Guide*. O'Reilly Media, Inc., 1st edition.

White, T. (2010). *Hadoop The Definitive Guide*. O'Reilly Media, Inc., 2nd edition.

White, T. (2012). *Hadoop The Definitive Guide*. O'Reilly Media, Inc., 3rd edition.

White, T. (2015). *Hadoop The Definitive Guide*. O'Reilly Media, Inc., 4th edition.