PROJECT NO. 50

NKR: ON TOP SCHEDULER FOR APACHE MESOS

MS.PASINEE SANTIVORRANANT
MR.SUPAPAT SRI-ON
MS.PARATTHA WEERAPONG

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI
2020

Project No. 50

NKR: On top scheduler for Apache Mesos

Ms.Pasinee Santivorranant

Mr.Supapat Sri-on

Ms.Parattha Weerapong

A Project Submitted in Partial Fulfillment

of the Requirements for

the Degree of Bachelor of Engineering (Computer Engineering)

Faculty of Engineering

King Mongkut's University of Technology Thonburi

2020

Project Committee

..... (Asst Prof.Rajchawit Sarochawikasit)	Project Advisor
..... (Asst Prof. Dr. Khajonpong Akkarajitsakul)	Committee Member
..... (Asst Prof. Dr. Phond Phunchongharn)	Committee Member
..... (Asst Prof. Sanan Srakaew)	Committee Member

Project Title	Project No. 50 NKR: On top scheduler for Apache Mesos
Credits	3
Member(s)	Ms.Pasinee Santivorranant Mr.Supapat Sri-on Ms.Parattha Weerapong
Project Advisor	Asst Prof.Rajchawit Sarochawikasit
Program	Bachelor of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2020

Abstract

In a multihop ad hoc network, the interference among nodes is reduced to maximize the throughput by using a smallest transmission range that still preserve the network connectivity. However, most existing works on transmission range control focus on the connectivity but lack of results on the throughput performance. This paper analyzes the per-node saturated throughput of an IEEE 802.11b multihop ad hoc network with a uniform transmission range. Compared to simulation, our model can accurately predict the per-node throughput. The results show that the maximum achievable per-node throughput can be as low as 11% of the channel capacity in a normal set of α operating parameters independent of node density. However, if the network connectivity is considered, the obtainable throughput will reduce by as many as 43% of the maximum throughput.

Keywords: Multihop ad hoc networks / Topology control / Single-Hop Throughput

หัวข้อปริญญานิพนธ์	หัวข้อปริญญานิพนธ์บรรทัดแรก หัวข้อปริญญานิพนธ์บรรทัดสอง
หน่วยกิต	3
ผู้เขียน	นางสาวภาสินี สันติวรนันท์ นายศุภพัฒน์ ศรีอ่อน นางสาวปรัษฐา วีระพงษ์
อาจารย์ที่ปรึกษา	ผศ.ดร.ราชวิชช์ สโรชวิกสิต
หลักสูตร	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
ปีการศึกษา	2563

บทคัดย่อ

การวิจัยครั้งนี้มีวัตถุประสงค์ เพื่อศึกษาความพึงพอใจในการให้บริการงานทั่วไปของสานักวิชา พื้นฐานและภาษา เพื่อเปรียบเทียบระดับความพึงพอใจต่อการให้บริการงาน ทั่วไปของสานักวิชาพื้นฐานและภาษา ของนักศึกษาที่มาใช้บริการสานักวิชาพื้นฐานและภาษา สถาบัน เทคโนโลยีไทย-ญี่ปุ่น จาแนกตามเพศ คณะ และชั้นปีที่ศึกษา เพื่อศึกษาปัญหาและข้อเสนอแนะของ นักศึกษามาเป็นแนวทางในการพัฒนาและปรับปรุงการให้บริการของสานักวิชาพื้นฐานและภาษา

คำสำคัญ: การชุบเคลือบด้วยไฟฟ้า / การชุบเคลือบผิวเหล็ก / เคลือบผิวรังสี

ACKNOWLEDGMENTS

Acknowledge your advisors and thanks your friends here..

CONTENTS

PAGE

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

LIST OF SYMBOLS

SYMBOL		UNIT
α	Test variable	m^2
λ	Interarival rate	jobs/ second
μ	Service rate	jobs/ second

LIST OF TECHNICAL VOCABULARY AND ABBREVATIONS

ABC	=	Adaptive Bandwidth Control
MANET	=	Mobile Ad Hoc Network

CHAPTER 1 INTRODUCTION

1.1 Problem Statement and Approach

Nowadays, several different types of applications, which are short or long-lived jobs, container orchestration, or MPI jobs, are executed in clouds or large computer clusters. Multiple users can demand difference resources to execute their tasks. Apache Mesos is a Middleware for the data center by introducing an abstraction layer that provides an entire data centers as a single large server. Instead of focusing on one application that running on a specific server. Mesos resource-isolation allows multi-tenant — the ability to run multiple applications on a single machine. Default sharing for multiple resources in this multi-tenant environment is defined by the Dominant Resource Fairness (DRF). Mesos receives the resources based on their current usage, which are responsible for scheduling their tasks within the allocation. In multiple schedulers can cause the fairness-imbalance in a multi-user environment, liked a greedy scheduler. It consumes more than its share of resources. Running multiple small tasks is better than launching large ones in terms of time spent waiting for enough resources.

Therefore, this project aims to improve the fairness of the scheduler by reducing the unfair waiting time due to higher resource demand in a pending task list and use log data to improve the whole cluster.

1.2 Objectives

- To study about job scheduling in Apache Mesos
- To study how to develop an algorithm to improve performance of scheduler in large-scale clustered environments.
- · To evaluate result and compare with Apache Mesos scheduler by using difference job types in the list (short job, long job, MPI)

1.3 Scope

- This project focuses on the reduction of job failed.
- Design and develop an add-on architecture on top of the Apache Mesos scheduler, to track and distribute the incoming tasks.
- What are the limitations of existing approaches?

1.4 Tasks and Schedule

Table 1.1 Semester 1's Gantt chart

Task/Week	August				September				October				November			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.Idea Document																
1.1 Find interesting problems																
1.2 Brainstorm ideas and choose topic																
1.3 Project discussion with advisor																
1.4 Write idea document report																
2.Proposal																
2.1 Explore related work and technologies																
2.2 Task breakdown																
2.3 Gantt chart																
2.4 Write proposal																
2.5 Present proposal																
3.Semester Report																
3.1 Literature review																
3.2 Design architecture diagram																
3.3 Design sequence diagram																
3.4 Write semester diagram																
3.5 Present Semester report																
4.Setup project & preparation																
4.1 Setup cluster & framework application																
4.2 Observe sharing and waiting time in queue for each framework																
4.3 Gathering server logs																

Table 1.2 Semester 2's Gantt chart

Task/Week	January				February				March				April				May			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
5.Implementation																				
5.1 Implement new on top scheduler for the whole cluster scheduling																				
5.2 Train model for scheduler prediction																				
6.Evaluation																				
6.1 Evaluate new on top scheduler with many situations																				
6.2 Evaluate model for scheduler prediction																				
6.3 Improve way to distributed framework																				
6.4 Tune model for scheduler prediction																				
7.Final Report																				
7.1 Write final report																				
7.2 Present final report																				

CHAPTER 2 BACKGROUND KNOWLEDGE AND LITERATURE REVIEW

2.1 Knowledge Background

In 2009, Apache Mesos [6] was a research project at the University of California in Berkeley. Benjamin, et al. wanted to improve datacenter efficiency by allowing multiple applications to share a single computing cluster across the many servers that make up a modern datacenter. So, multiple applications can share the processor, memory, and hard drive with any laptop or workstation. In 2010, the Mesos project entered the Apache Incubator, an arm of the Apache Software Foundation, so this project can gain the full support of the ASF's efforts. In 2013, The Apache Mesos project graduated from the incubator and founded Mesosphere. Mesosphere's flagship product, the Datacenter Operating System (DCOS), commercializes the open-source project by providing a turnkey solution to enterprises looking to deploy applications and scale infrastructure as effortlessly as other companies using Mesos, such as Airbnb, Apple, and Netflix.

Meanwhile, most such operating systems only fairly divide and account for CPU cycles. So, performance isolation is essential to operating systems shared by dependable services. These dependable services require specifying and enforcing policies for all resources, and that current metrics for evaluating fair sharing are insufficient. In 2006, Aage Kvalnes et al. researched new policy specifications and metrics, and illustrated these with the help of a new operating system that supports holistic resource sharing. [18]

In data centers and clouds, where applications could be co-scheduled on the same physical nodes, resource fairness needs to extend to multiple resource types such as memory, disk I/O, and network bandwidth. Ali, et al. considered the problem of fair resource allocation in a system containing different resource types, where each user may have different demands for each resource and researched about a new generalization of max-min fairness to multiple resource types called Dominant Resource Fairness (DRF). [5]

2.2 Theoretical and Core Concepts

2.2.1 Tasks Failures Detection

Hadoop usually uses JobTracker to detect failures of the TaskTracker nodes. It detects with heartbeat-based failure detection. The TaskTracker will send heartbeat message to JobTracker and JobTracker will declare a TaskTracker as dead only when it does not receive heartbeat for a limited time. It cannot quickly detect the failures and it may assign task to dead nodes. This can increase the number of failure tasks in Hadoop. [16]

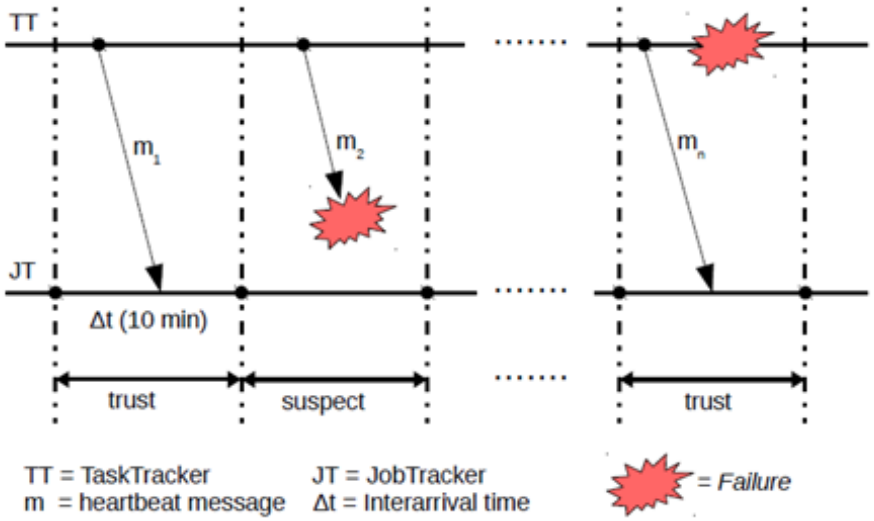

Figure 2.1 TaskTracker Failure Detection Model in Hadoop Framework [16]

For example, active TaskTracker send heartbeat messages to JobTracker every 3 seconds. While JobTracker check the timeout condition every 200 seconds. And there are network delays or messages losses, so some heartbeat may arrive late or loss. The JobTracker may consider that TaskTracker as dead node even it is available as shown in Figure 2.1. that heartbeat message m2 does not arrive and the JobTracker consider this TaskTracker as dead.

2.2.2 Container Technology

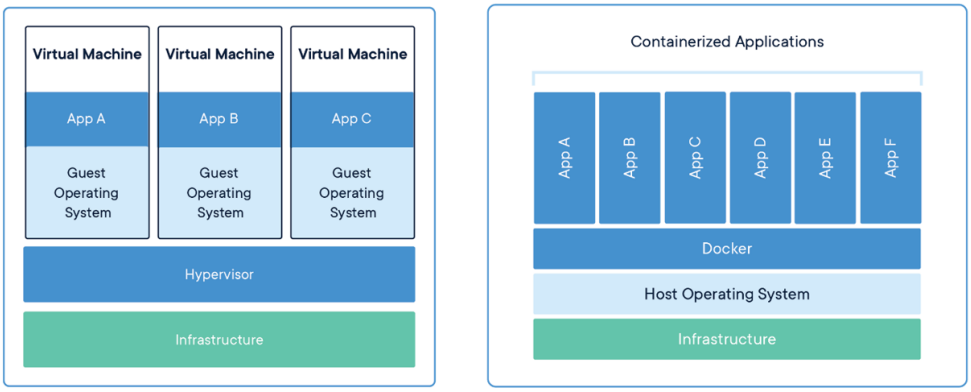

Figure 2.2 Virtual machine vs Container [2]

Container Technology is a method to package up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another, with container software having names including the popular choices of Docker, Apache Mesos, RKT, and Kubernetes. The virtual machine contains the entire operating system. Therefore, the physical server that runs several virtual machines is running several operating systems' simultaneously as shown in Figure 2.2. [2]

There is a lot of overhead on virtual machine. In contrast, with container technology, the server runs a single operating system. Each container can share this single operating system with other containers on server. Containers require less resource of server with less overhead and more efficient than virtual machines. [1]

Containers are set up to accomplish work in a multiple container architecture (container cluster). They also enable a program to be broken down into smaller pieces, which are known as microservices. So, the program can work on each of the containers separately.

2.2.3 Overview of machine learning

Machine learning is a branch of artificial intelligence (AI). It is the machine's ability to learn from data provided without human intervention and able to improve decision from experience. Without human directed programming instructions, the machine accesses data, observes and finds data pattern. The more data input the better data pattern learning and better decision making.[3]

2.2.4 Overview of machine learning

An artificial neural network (ANN) is a computational model imitates the natural human brain. The network consists of hundreds or thousands small neuron nodes. Those numerous neuron nodes communicate to each other in the web form. The neuron node is called processing unit. Each of processing unit is interconnected by nodes. Each processing unit comprises of input unit and output unit. Input unit receives various type of data format. It also has an internal weighting system. The neural network learns from the input and produce output result.

ANN use rules and guidelines to generate result/ output. The set of these learning rules is called backpropagation because it uses backward propagation of error to learn or improve the better result. ANN learns data patterns in training phase. It compares actual output with the desired output that is expected result in supervised phase. The difference between actual and expected result are worked backward to adjust the weight of its connections between the units. The purpose is to make the lowest possible error. [4]

2.2.5 Deep learning

Nowadays, there are collections of vast unlabeled and unstructured data gathering from various sources that is difficult to analyses useful information by traditional programs in a linear way. The hierarchical level of artificial neural network that work in web form to process data with a nonlinear approach is called Deep Learning. The first layer of the neural network processes a raw data and pass output on to the next layer. The second layer processes first layer output plus additional information and pass output to next layer again. This continues across all levels of the neuron network to make information more meaningful information. [12]

Deep learning models can achieve high accuracy, sometimes exceeding human-level performance. "Deep" refers to the powerful number of hidden layers in the neural network. However, it needs lots of labeled data and high-performance machines to analyze. The organized layers of interconnected nodes can be tens or hundreds of hidden layers. [17] as shown in Figure 2.3.

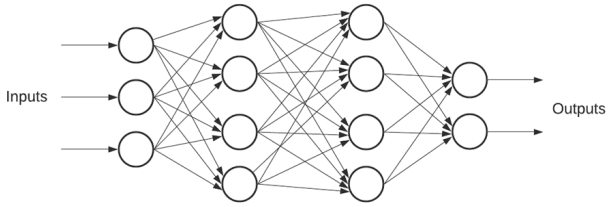

Figure 2.3 Neural networks.

2.3 Technologies survey

2.3.1 Apache Mesos

Mesos consists of a master, agent daemons running on each cluster node, and Mesos frameworks that run task on these agents as shown in Figure 2.4. Architecture consist of three components: masters, slaves, and the frameworks that run on them. Mesos relies on Apache ZooKeeper, a distributed database used specifically for coordinating leader election within the cluster, and for leader detection by other Mesos masters, slaves, and frameworks. [8]

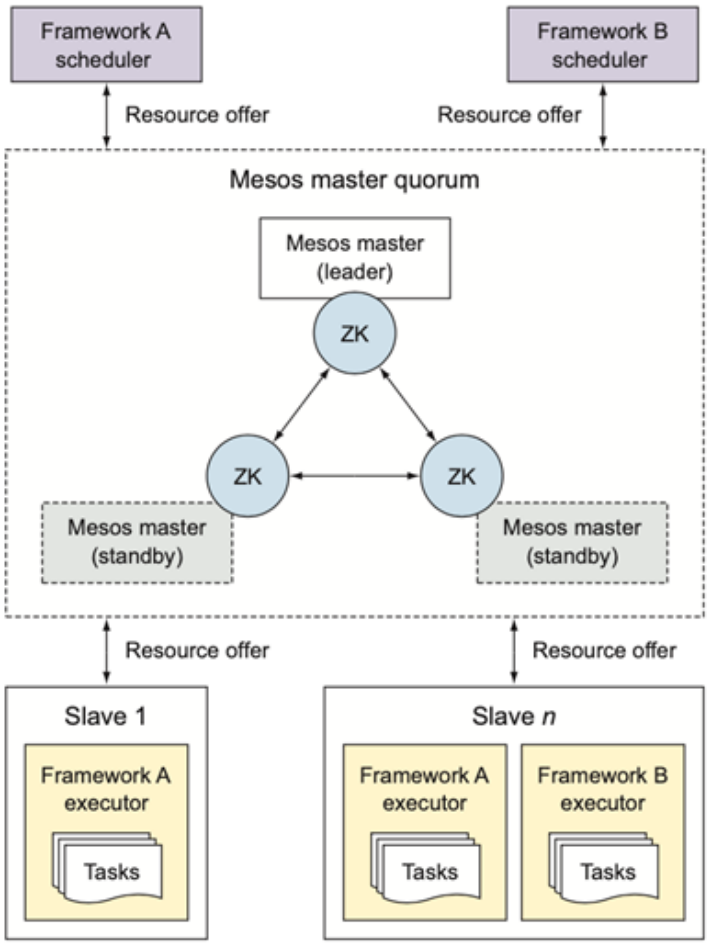

Figure 2.4 The Mesos architecture consists of one or more masters, slaves, and frameworks. [8]

1. **Masters:** Mesos masters are responsible for managing the Mesos slave daemons running on each machine in the cluster. Using Zookeeper, they coordinate which node will be the leading master, and which masters will be on standby, and ready to take over if the leading master goes offline. A Mesos cluster requires minimum one master, and three or more are recommended for production deployments. Zookeeper can run on the same machines as the Mesos masters themselves or use a standalone Zookeeper cluster.
2. **Slaves:** The machines in a cluster responsible for executing a framework's tasks.
3. **Frameworks:** Mesos application that's responsible for scheduling and executing tasks on a cluster. A framework is made up of two components: a scheduler and an executor.

- **Schedulers** A scheduler is a long-running service responsible for connecting to a Mesos master and accepting or rejecting resource offers. Mesos delegates the responsibility of scheduling to the framework, instead of attempting to schedule all the work for a cluster itself. The scheduler can then accept or reject a resource offer based on whether it has any tasks to run at the time of the offer.
- **Executor** An executor is a process launched on a Mesos slave that runs a framework's tasks on a slave.

Dominant resource is a resource of specific type (CPU, memory, disk, ports) which is most demanded by given framework among other resources it needs. DRF computes the share of resource allocated to a framework (dominant share) and tries to maximize the smallest dominant share in the system. for next round offers the resources first to the one with smallest dominant share, then to the second smallest one and so on. [9] Example with 9 CPUs and 18 GB RAM to two frameworks running task that require <1 CPU, 4GB> and <3CPUs, 1GB> shown in Table 2.1.

Table 2.1 Example of running 2 frameworks.

Schedule	Framework A		Framework B		total	
	Resource Share	Dominant Share	Resource Share	Dominant Share	CPU	RAM
B	<0, 0>	0	<3/9, 1/18>	1/3	3/9	1/18
A	<1/9>, <4/18>	**2/9**	<3/9, 1/18>	1/3	4/9	5/18
A	<2/9>, <8/18>	4/9	<3/9>, <1/18>	**1/3**	5/9	8/18
B	<2/9>, <8/18>	**4/9**	<6/9>, 2/18>	2/3	8/9	10/18
A	<3/9>, <12/18>	2/3	<6/9>, 2/18>	2/3	1	14/18

2.4 Technologies survey

Table 2.2 test table method1

Center	Center	left aligned	Right	Right aligned
Center	Center	left aligned	Right	Right aligned
Center	Center	left aligned	Right	Right aligned
Center	Center	left aligned	Right	Right aligned
Center	Center	left aligned	Right	Right aligned

2.5 Text Processing Algorithms

2.5.1 Algorithm I

You can place the figure and refer to it as Figure 2.5. The figure and table numbering will be run and updated automatically when you add/remove tables/figures from the document.

2.5.2 Algorithm II

Add more subsections as you want.

2.6 Development Tools

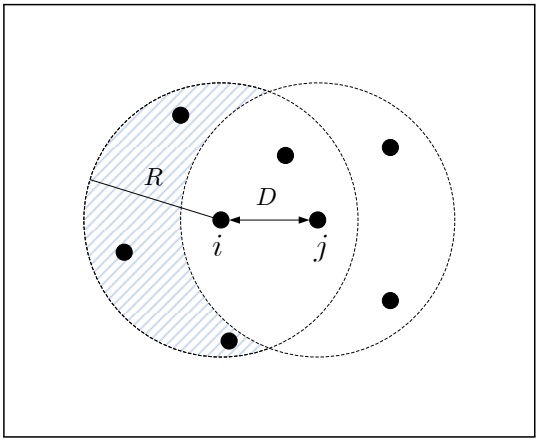

Figure 2.5 The network model

CHAPTER 3 PROPOSED WORK

Explain the design (how you plan to implement your work) of your project. Adjust the section titles below to suit the types of your work. Detailed physical design like circuits and source codes should be placed in the appendix.

3.1 System Architecture

Table 3.1 test table x1

SYMBOL		UNIT
α	Test variable	m^2
λ	Interarrival rate	jobs/ second
μ	Service rate	jobs/ second

3.2 System Specifications and Requirements

3.3 Hardware Module 1

3.3.1 Component 1

3.3.2 Logical Circuit Diagram

3.4 Hardware Module 2

3.4.1 Component 1

3.4.2 Component 2

3.5 Path Finding Algorithm

3.6 Database Design

3.7 GUI Design

CHAPTER 4 IMPLEMENTATION RESULTS

You can title this chapter as **Preliminary Results** or **Work Progress** for the progress reports. Present implementation or experimental results here and discuss them.

CHAPTER 5 CONCLUSIONS

This chapter is optional for proposal and progress reports but is required for the final report.

5.1 Problems and Solutions

State your problems and how you fixed them.

5.2 Future Works

What could be done in the future to make your projects better.

REFERENCES

1. Jonas DeMuro, 2019, "What is container technology?," .
2. docker, 2020, "What is a Container?," .
3. expertsystem, 2020, "What is Machine Learning? A definition," .
4. Jake Frankenfield, 2020, "Artificial Neural Network (ANN)," .
5. Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica, 2011, "Dominant resource fairness: fair allocation of multiple resource types," **8th USENIX conference on Networked systems design and implementation**, pp. 323–336, march 2011.
6. Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica, 2011, "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center," **8th USENIX conference on Networked systems design and implementation**, April 2011.
7. IBM, 2020, "Apache ZooKeeper," .
8. Roger Ignazio, 2016, **Mesos in Action**, Manning Publications Co.
9. Anton Kirillov, 2016, "Resource Allocation in Mesos: Dominant Resource Fairness," .
10. Qing Liu, Tomohiro Odaka, Jousuke Kuroiwa, and Hisakazu Ogura, 2013, "A Fair Scheduling Algorithm for Adaptive Heterogeneous Resources in Data Centers," **IEICE**, pp. 872–885, April 2013.
11. Wenbin Liu, Ningjiang Chen, Hua Li, Yusi Tang, and Birui Liang, 2018, "A Fair Scheduling Algorithm for Adaptive Heterogeneous Resources in Data Centers," **the Tenth Asia-Pacific Symposium**, September 2018.
12. Hargrave Marshall, 2019, "Deep Learning," .
13. Abueg Ralf, 2020, "Elasticsearch: What It Is, How It Works, And What It's Used For," .
14. Pankaj Saha, Angel Beltre, , and Madhusudhan Govindaraju, 2019, "Tromino: Demand and DRF Aware Multi-Tenant Queue Manager for Apache Mesos Cluster," **2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC) 63-72**, May 2019.
15. Yegulalp Serdar, 2019, "What is Docker? The spark for the container revolution," .
16. Mbarka Soualhia, Foutse Khomh, and Sofiène Tahar, 2018, "Adaptive Failure-Aware Scheduling for Hadoop," **2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)**, February 2018.
17. TheMathWorks, 2020, "What Is Deep Learning?," .
18. Robbert van Renesse, Aage Kvalnes, Dmitrii Zagorodnov, and Dag Johansen, 2006, "Policies and Metrics for Fair Resource Sharing," **University of Tromsø**, January 2006.
19. Wikipedia, 2020, "Apache Kafka," .