TASK SCHEDULING FOR EDGE COMPUTING IN CLOUD MANUFACTURING ENVIRONMENT

THESIS

Submitted as the requirements for the degree of Sarjana Teknik Industri
Industrial Engineering International Program
Department of Industrial Engineering
At Universitas Islam Indonesia

Name : Zikra Wahyudi
Student Number : 15522345

INTERNATIONAL PROGRAM

DEPARTMENT OF INDUSTRIAL ENGINEERING

UNIVERSITAS ISLAM INDONESIA YOGYAKARTA

JANUARY 2023

AUTHENCITY STATEMENT

For the sake of Allah, I confess that this research was conducted by me except for the summaries of the sources that have been cited and mentioned. If in the future my confession is proved to be wrong and dishonest resulting in the violence of legal regulation of the papers and intellectual property rights, then I am willing to return my degree I received to be withdrawn by Universitas Islam Indonesia.

Yogyakarta, January 2023

Zikra Wahyudi

THESIS APPROVAL OF SUPERVISOR

TASK SCHEDULING FOR EDGE COMPUTING IN CLOUD MANUFACTURING ENVIRONMENT

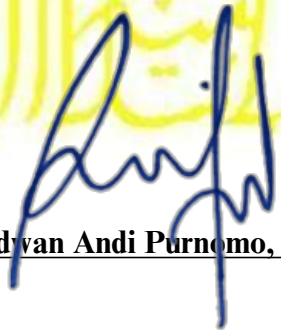
THESIS

Arranged by:

Name : **Zikra Wahyudi**
Student Number : **15522345**

Yogyakarta, January 2023

Supervisor,



Ir. Muhammad Ridwan Andi Purnomo, S.T,. M.Sc., Ph.D., IPM

THESIS APPROVAL OF EXAMINATION COMMITTEE

TASK SCHEDULING FOR EDGE COMPUTING IN CLOUD MANUFACTURING ENVIRONMENT

Arranged by:

Name : Zikra Wahyudi
Student Number : 15522345

Was defended before Examination Committee in Partial Fulfillment of the requirements for the degree of Idnustrial Engineering Departement

Universitas Islam Indonesia

Examination Committee

Ir. Muhammad Ridwan Andi Purnomo, S.T,. M.Sc., Ph.D., IPM
Examination Committee Chair

Agus Mansur, S.T., M.Eng.Sc.
Member I

Annisa Uswatun Khasanah, S.T., M.Sc.
Member II

Acknowledged by,

Head of Department
Industrial Engineering - Industrial Program
Universitas Islam Indonesia

Ir. Muhammad Ridwan Andi Purnomo, S.T,. M.Sc., Ph.D., IPM

Table of Contents

List Of Table

List Of Figure

Abstract

Cloud manufacturing (CMfg) is a service-centric approach that transforms manufacturing resources and capabilities into manufacturing services, using technologies such as network, cloud computing, and service computing. The main problem in CMfg environment is the network latency in cloud layer. This research proposes a generic system architecture for collaboration between edge computing and CMfg and consist of into three layer; cloud layer, edge layer, and manufacturing resource layer. Also in this research proposed Dynamic Based Prioritize Algorithm (DPBA) a dynamic algorithm for efficient task scheduling in edge computing that create scheduling prioritize based on task length. The proposed algorithm is evaluated through simulations and compared with other existing algorithm such as Shortest Job First (SJF) and Early Due Date (EDD), the results demonstrate its effectiveness and potential benefits of DPBA rather than SJF and EDD. Overall, this research contributes to the field of CMfg and edge computing, and provides a practical solution for improving manufacturing productivity.

Keywords: Cloud Manufacturing, Cloud Edge Collaboration, Dynamic Scheduling

Chapter I

Introduction

1.1 Background

The development of information technology has transformed the way production is organized and carried out in manufacturing. In the 1960s, the focus of manufacturing was on increasing production scale. In the 1970s, the emphasis shifted to reducing production costs. In the 1980s, the focus was on improving product quality, and in the 1990s, manufacturing sought to respond quickly to market demand. (Tao et al., 2011)

The emergence of industry 4.0 has recently transformed manufacturing through the integration of technologies from different disciplines, such as cyber-physical systems (CPS), the internet of things (IoT), and service-oriented architecture (SOA). These technologies enable the creation of new and innovative manufacturing systems that are more flexible, efficient, and responsive than traditional approaches. (Tao et al., 2011)

The development of information technology has led to the emergence of new manufacturing paradigms, such as cloud manufacturing (CMfg). CMfg is a service-centric approach that transforms manufacturing resources and capabilities into manufacturing services, using technologies such as network, cloud computing, and service computing (Zhang et al., 2014). The main goal of CMfg is to improve resource utilization and collaboration effectiveness. In CMfg, all resources and capabilities are provided as services (Wang & Wang, 2017).

A CMfg system is made up of manufacturing resources and capabilities, a manufacturing cloud, and the entire manufacturing life cycle (Zhang et al., 2012). According to Ren (2013), a manufacturing resource is any entity that can support an activity or function involved in the product life cycle. There are two main types of manufacturing resources: hard resources, such as manufacturing cells or IT hardware,

and soft resources, such as software, data, information, knowledge, or other intellectual elements. The cloud manufacturing platform is organized into five layers: the resource perception layer, the virtual pool layer, the middleware layer, the toolkit layer, and the user interface layer (Ren et al., 2013). Figure 1 shows the architecture of the cloud manufacturing platform.

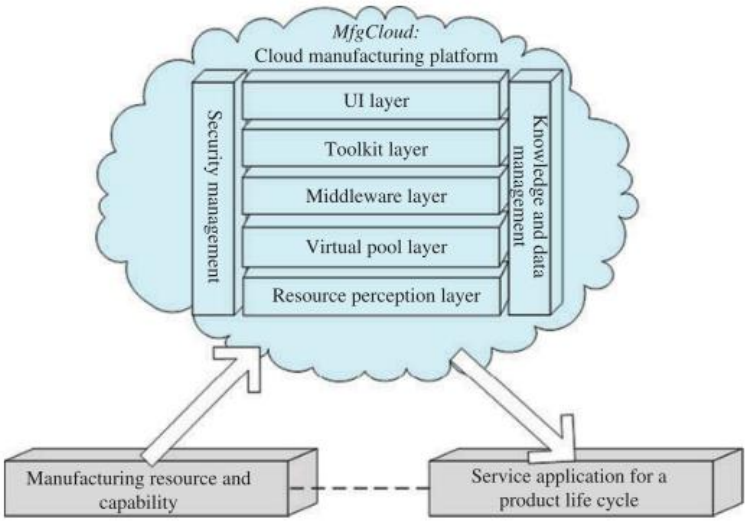


Figure 1.2 The architecture of cloud manufacturing systems (Ren et al. 2013)

In practice, cloud manufacturing can be seen as an intilligenet and collaborative manufacturing service model. Distributed manufacturing resources, such as machine tools, 3D printining, and CAx software, and manufacturing capabilities, such as design, fabrication, assembly, simulation, and testing, are interconnected and form a shared pool in the cloud manufacturing platform. Through cloud manufacturing, customers can easily access services such as design as a service (Wu et al., 2012).

Amongst these solutions, CMfg provides a promising solution and an even broader definition for the concept of "Design Anywhere and Make Anywhere" or called as DAMA (Wang & Xu, 2013). More broadly, DAMA can be described as completing designing and manufacturing procedures via clouds, while the user and business partners are loosely connected by a cloud-centralized network. Cloud manufacturing is also a smart networked manufacturing model that embraces cloud computing, aiming at meeting growing demands for higher product individualization, broader global cooperation, knowledge-intensive innovation, and increased agility in market response (Wang & Wang, 2017).

The aim of cloud manufacturing is to deliver on-demand manufacturing services to consumers over the internet (Liu et al. 2018). Scheduling is a critical point for achieving the aim of cloud manufacturing. In manufacturing or production, scheduling can be defined as a process of arranging, controlling, and optimizing works or workloads (Pinedo, 2016). In the context of cloud manufacturing (CMfg), scheduling can be defined narrowly as the process of allocation of resources or services to tasks, monitoring, controlling, and optimizing resources or service status and task execution so as satisfy consumers' individualized requirements (Liu et al. 2018). Whereas in a broad sense, scheduling encompasses not only the scheduling process in the narrow sense but also many other activities such as task processing (especially task decomposition), service discovery, matching, selection, and composition that are either involved in or indispensable for the scheduling of process (Tao et al. 2015; Cheng et al 2017).

The traditional scheduling system of cloud manufacturing is located in the cloud (Jian, Ping, & Meiyu, 2021). This condition creates high latencies in communication between each edge factory and the cloud center when processing multiple delay-sensitive complex manufacturing tasks, resulting in data transmission lag so that real-time performance could not be guaranteed (Li, Wan & Dai et al. 2019). In recent times, several research on the scheduling system of cloud manufacturing, and the subtask scheduling system can be deployed on the edge. The processing devices which dispatch the subtasks to the edge nodes at the network edge can reduce the cost of sub-task data transmission effectively (Ciobanu et al. 2019).

In recent years, the use of edge computing in combination with intelligent devices has gained significant attention (Ray, 2016). Moving computing, storage, and intelligence closer to the source of information has been shown to have numerous benefits in domains such as manufacturing. The advantages of edge computing are often measured in terms of improved response times (Caiazza et al., 2022). In a cloud manufacturing environment, data traffic scheduling on edge computing can help to improve communication efficiency and optimize resource utilization. Edge computing can provide real-time data processing and analytics, which can be used to schedule production tasks and optimize manufacturing processes.

The several challenge in cloud manufacturing with 3D printers is ensuring reliable and continuous operation. 3D printing requires a steady and uninterrupted

power supply and internet connectivity. If the 3D printer or the cloud infrastructure experiences downtime, it can cause significant disruption to production schedules and negatively impact customer satisfaction.

The implementation an edge layer to address the problems in cloud manufacturing with 3D printers refers to the computing infrastructure located at the edge of a network, near the end-users and devices, rather than in a centralized data center. By processing data locally, the edge layer can help to address some of the challenges associated with cloud manufacturing with 3D printers. The edge layer also can minimize the network latency and connectivity which is critical for real-time applications such as 3D printing.

Furthermore, the increase demand of the cloud manufacturing and also the collaboration with the edge computing make the research in this field more importan, in this research problem addressed in this research is task scheduling on edge computing in a cloud manufacturing environment. The objective is to design and system architecture for cloud edge collaboration and optimize the scheduling of tasks for edge computing to improve manufacturing productivity. The output from this research is still prototype and the system can be implemented in MSME-scale cloud manufacturing systems. This research aims to contribute to the field of cloud manufacturing and edge computing system.

1.2 Problem Formulation

Based on the background described, the problem formulation that can be highlighted for this research is:

1. What is the system architecture that effectively support the edge computing in a cloud manufacturing environment?
2. How to schedule task for edge computing in a cloud manufacturing environment?

1.3 Research Objective

The objectives of this research are as follows:

1. Identify the key features and design a system architecture that can effectively support edge computing in a cloud manufacturing environment

2. Design an dynamic algorithm to scheduling the task for edge computing cloud manufacturing environment.

1.4 Research Limitation

In order to derive the conclusion in a more directed way, a research limitation is determined. The scope of this research follows

1. This research only focuses on designing a generic system architecture for collaboration between edge computing and cloud manufacturing.
2. This research only focuses on designing a dynamic algorithm for efficient task scheduling in edge computing.

1.5 Research Benefits

The expected benefits of this research to some other involved parties are floors:

1. For the author, This study is aimed to fulfill the obligation of the author as an undergraduate student to accomplish and obtain Industrial Engineering Degree at the International Program, Department of Industrial Engineering, Universitas Islam Indonesia.

2. For manufacturers The result of this research can be used as a benchmarking to improve manufacturing capabilities by the manufacturers.

3. For other parties the result of this research can be developed in the future since this research has its own limitation and this research can be beneficial as an additional reference for further development in terms of integration between cloud manufacturing and edge computing.

1.6 Writing System

In order to systematically write the thesis in a structured order thus the thesis is arranged in the following order:

CHAPTER I INTRODUCTION

This chapter makes a brief summary of the background of the problem, problem formulation, problem boundaries, research objectives, the benefits of research, and systematic of the research.

CHAPTER II LITERATURE REVIEW

This chapter contains basic concepts and principles needed to resolve the issue in this research. Moreover, to summarize previous research that has been conducted before that is related to the research.

CHAPTER III RESEARCH METHODOLOGY

This chapter contains a description of the framework and flowchart of research, the techniques carried out, the model used, the development of models, materials or materials, tools, procedures for research, and the data to be studied, and the method of analysis used.

CHAPTER IV SYSTEM SIMULATION AND ANALYSIS

In this chapter will provide an system architecture fo Cloud Edge Collaboration Manufacturing system. Also in this chapter will describe about the designed algorithm for task scheduling in edge computing.

CHAPTER V RESULT AND DISCUSSION

This chapter provides a discussion of the results obtained in the study, and the suitability of the results with the objectives of the study subsequently to produce a recommendation.

CHAPTER VI CONCLUSION AND RECOMMENDATION

This chapter contains conclusions about the analysis made and suggested suggestions for the results obtained and debates found during the study so that evaluation needs to be done to be studied in further research.

CHAPTER II

Literature Review

2.1 Empirical Study

Empirical study refers to research with the topic pertaining to this research that has been conducted before. The empirical study can be used as a reference and benchmark for the concepts used. The empirical study overview is compiled in a table that encompasses the research objective, method, object, and result. Table 1 drafts the empirical study as shown as follow:

Table 2.1 Empirical Study

No.	Year	Author	Title	Objective	Method	Result
1.	2018	Raileanu S., Borangiu T., Morariu O., Iacob I.	Edge Computing in Industrial IoT Framework for Cloud. based Manufacturing Control	Develop a generic architecture for information and data collection, smart processing, and aggregation at the edge of large-scale manufacturing control systems; the edge is represented by the set of shop floor entities (things) – resources and intelligent products that are identified and communicate in multi-agent systems for decentralized MES tasks.	The two-layer agent system is designed to allow for flexibility in how the capabilities of the agents are configured. The lower-layer agents are responsible for data streaming and primary analysis, while the higher-layer agent's process time-ordered data and track the covariance of multiple monitored metrics. This system is designed to allow for better decision-making in areas such as optimizing global cost functions, reconfiguring resource teams, and predicting behaviors and trends.	The paper discusses a solution that would allow devices on a shop floor to communicate with each other, with resources and intelligent products within a production control system. This solution is based on aggregation nodes, would collect data from multiple directly connected devices and then forward this data to the cloud control platform hosting the high-level MES control system. This system would be responsible for operation optimization, execution, and monitoring.

No.	Year	Author	Title	Objective	Method	Result
2.	2020	Jian, C., Ping, J., Zang, M.	A cloud edge-based two-level hybrid scheduling learning model in cloud manufacturing	To minimize the completion time of the atomic tasks and the best load between edge factory nodes. The author proposed a cloud edge-based two-level hybrid scheduling learning model.	The author classified it into three layers, the first layer is the cloud scheduling center which deploys the improved LSTM prediction model; the second layer is the edge scheduling center, which directly interacts with edge nodes and industrial devices or cloud centers, and the third layer consists of industrial devices such as intelligent robots, which are the physical resources that ultimately perform manufacturing tasks in each edge factory node.	Experiments show that the proposed learning model can improve the performance of the cloud manufacturing platform efficiently, whereas the proposed two-level scheduling strategy can effectively balance the load between the edge factory nodes. The improved LSTM prediction model can predict the scheduling results quickly and efficiently.

No.	Year	Author	Title	Objective	Method	Result
3.	2021	Wang X., Wan, J.	Cloud-Edge Collaboratio Based Knowledge Sharing Mechanism for Manufacturing Resources	A cloud-side collaborative manufacturing resource knowledge-sharing mechanism provides a new solution for the mutual learning and knowledge-sharing of manufacturing resources.	This paper adopts the concept of a cloud-edge collaboration. It constructs data communication network architecture in a manufacturing environment to enable the reuse of multi-variety mixed-flow manufacturing and processing experience.	The experiment demonstrated the rationality and effectiveness of the proposed cloud-edge combined knowledge reuse system for manufacturing resource information. The results showed that the proposed knowledge reuse method had high accuracy, and the action reduction effect was 95.8%. The experiment confirmed the superiority of the proposed task decomposition strategy.
4.	2022	Yang, B., Pang, Z., Wang S.,	A coupling optimization method of	The purpose of this paper is to Production scheduling-Computation	Designed and enhanced multi-population multi-objective whale	The experimental results based on standard test functions and engineering

No.	Year	Author	Title	Objective	Method	Result
		Mo, F., Gao, Y.	production scheduling and computation offloading for intelligent workshops with cloud-edge terminal architecture.	offloading Coupling optimization (PCCO) model for the intelligent workshop with CET architecture. The maximum completion time of production jobs and the minimum total offloading delay time of computing tasks are used as the objectives to establish a bi-objective optimization model of the problem.	optimization algorithm (EMMOWOA) and arithmetic exploration mechanism to selection method and ht multi-population strategy is designed to enhance the exploration and exploitation capabilities of EMMOWOA.	examples show that EMMO- WOA has higher convergence and diversity than the commonly used multi-objective optimization algorithms and the proposed PCCO model can well balance production efficiency and computing delay, and better meets the actual needs of intelligent workshops.
5.	2022	Cui, J., Ren, L., Mai, J., Zheng, P., Zhang.	3D Printing in the Context of Cloud Manufacturing	The aims of this paper presents a comprehensive model, technology, platform, and application to	The author of this paper defined fourth layers, i.e. access adapter layer, virtual pool layer, 3D printing service management layer,	The research will provide a valuable theoretical and practical reference to the future development and deployment of 3D printing

No.	Year	Author	Title	Objective	Method	Result
				facilitate 3D printing in a cloud manufacturing environment.	and user toolkit layer. The overall architecture also contains the physical resource layer and 3D printing application layer.	clouds, and promote a novel 3D printing business model in a service-oriented manner to achieve mass customization in Industry 4.0.
6.	2022	Wang, H., Sun, M., Zhang, L., Dong, P., Wei, Y., Mei, J.	Scheduling optimization for upstream data flows in edge computing	This paper proposes an edge node control model in the network environment where the cloud edge is integrated and proposes a multilevel feedback queue scheduling for upstream data flow also proposes a logic structure of multilevel feedback queue scheduling is TSAS.	To solve the problem, this paper first constructs a mixed-integer programming (MIP) model, then proposes a constructive greedy heuristic (CGH) algorithm, and finally develops an improved particle swarm optimization with local optimization (IPSOL) algorithm by combining the characteristics of the studied problem.	The experimental results verified the feasibility of the TSAS algorithm in processing terminal dataflows of edge nodes. This multilevel feedback queue scheduling algorithm with an adaptive fusion time slice achieved the objective of reducing energy consumption, shortening delay, and providing high throughput.

No.	Year	Author	Title	Objective	Method	Result
7.	2022	Zhang, Tang, Luo, and Zhang	Deadline-Aware Dynamic Task Scheduling in Edge-Cloud Collaborative Computing	Propose a hierarchical architecture for edge-cloud collaborative environments in the Industrial Internet of Things (IoT) and	Present a dynamic method time-sensitive scheduling algorithm (DSOTS) and optimize with greedy strategy.	Aiming to choose a more suitable server to handle dynamically incoming tasks, our algorithm decreases the average processing time and cost by 30% and 45%, respectively, as well as the average SLA violation by 25%, when compared to existing state-of-the-art solutions.

8.	2022	Wahyudi, Z.	Task Scheduling For Edge Computing In Cloud Manufacturing Environtment	Propose a system architecture for Cloud Edge Collaboration Manufacturing and design an algorithm for task scheduling in edge computing.	Design a generic system architecture design with 3 layersand algorithm based prioritize called DPBA.	Implementing the system architecture for cloud edge collaboration manufacturing on cloud server and edge node and design DPBA that more efficiently to doing task scheduling in edge computing.

Based on the empirical study above, there are some conclusions that can be derived. A study by Raileanu, Borangiu, Morariu & Iacob (2018) aims to develop a generic architecture that is needed for collecting information and data, processing it smartly, and aggregating it at the edge of large-scale manufacturing control systems. The edge is represented by the set of shop floor entities (things) – resources and intelligent products that communicate in multi-agent systems for decentralized MES tasks. This system would be responsible for operation optimization, execution, and monitoring. This paper did not mention directly about edge computing in cloud manufacturing. From the experiments, the researcher found MQTT protocol is the best voice for sending data from the embedded systems to the cloud.

The study by Jian, Ping & Zhang (2021) aimed to minimize the completion time of the atomic tasks and the best load between edge factory nodes. The author proposed a cloud edge-based two-level hybrid scheduling learning model and Long Short-Term Memory (LSTM) to train the scheduling data using the deep learning method. The experiments show that the proposed learning model can improve the performance of the cloud manufacturing platform in real-life applications efficiently.

Wang & Wan (2021) proposed an understanding of the mechanism for knowledge sharing between manufacturing resources that are based on cloud-edge collaboration. As the object of their research, the researcher used Robot CNC Machine as the manufacturing physical resource. The result of these experiments showed the rationality and effectiveness of the proposed cloud-edge combined knowledge reuse system for manufacturing resource information. The results showed that the proposed knowledge reuse method had high accuracy, and the action reduction effect was 95.8%. The experiment confirmed the superiority of the proposed task decomposition strategy. The author also proposed combining the existing knowledge graph research results to transform them into a semantic model and a knowledge graph to enhance the universality of knowledge sharing.

Furthermore, a study by Sing et al. (2021) aimed to find the importance of the integration of 3d printers and Cloud Manufacturing in the construction field. This research wants to prove and propose the implementation of 3D printing in sustainable construction technologies. Additional important topics in this paper include automation with robotics, predictive analytics for 3D printers, research for

enhancement of integrating CMfg with 3D printers and also eco-friendly printing, and 5G technology for reliable and fast connectivity for IoT-based CMfg.

The recent research by Yang, Pang, Wang, Mo & Gao (2022) proposed a model of Production scheduling-Computation offloading Coupling optimization (PCCO) for the intelligent workshop with cloud-edge terminal (CET) architecture. The maximum completion time of production jobs and the minimum total offloading delay time of computing tasks are used as the objectives to establish a bi-objective optimization model of the problem. The experimental results based on standard test functions and engineering examples show that EMMOWOA has higher convergence and diversity than the commonly used multi-objective optimization algorithms and the proposed PCCO model can well balance production efficiency and computing delay, and better meets the actual needs of intelligent workshops.

Cui, Ren, Mai, Zheng & Zhang (2022) have a study that aimed to present a comprehensive model, technology, platform, and application to facilitate 3D printing in a cloud manufacturing environment. The researcher of this paper designed a system architecture of a #D printing cloud platform. A prototype of the 3D printing cloud platform is deployed with a case study of an aircraft engine part. The research will provide a valuable theoretical and practical reference to the future development and deployment of 3D printing clouds, and promote a novel 3D printing business model in a service-oriented manner to achieve mass customization in Industry 4.0.

The last recent research from Zhang, Li, and Jia (2022) provides an understanding of parallel batch processing in machine scheduling in cloud manufacturing. The purpose of this paper is to create an understanding of parallel batch processing machines in cloud manufacturing and study the effect of different algorithm parameters *n* the performance of the improved meta-heuristic algorithm and selected the best combination for the algorithm. this paper first constructs a mixed-integer programming (MIP) model, then proposes a constructive greedy heuristic (CGH) algorithm, and finally develops an improved particle swarm optimization with local optimization (IPSOL) algorithm by combining the characteristics of the studied problem. Experimental results show that the IPSOL algorithm can achieve more competitive results under these two distributions.

Therefore it can be concluded that data scheduling on edge computing in a cloud manufacturing environment is a prospective management system that

theoretically and practically offers many advantageous features that facilitates the resolution of issues and works dynamically. However, the challenge in the data scheduling on edge computing in a cloud manufacturing environment is in the implementation which requires profound improvement, especially in the scheduling optimization algorithms. While most of the research is performed directly on a cloud server, this research is carried out through edge computing as the bridge between the physical manufacturing resource and the cloud manufacturing server. Therefore, this research will conduct a case study in 3D printing. This research implements data traffic scheduling in edge computing using raspberry pi as the edge gateway. The implementation process follows four major aspects i.e. constructing the cloud manufacturing, edge computing, data traffic scheduling, and 3D printing.

2.2 Cloud Manufacturing

Cloud Manufacturing is the application of cyber-physical systems, the Internet of Things, and cloud computing to enable the rapid, on-demand, and distributed production of products and services. CMfg can provide a variety of manufacturing services that are safe, reliable, high-quality, and affordable. These services can be used throughout the entire life cycle of a product, from its initial design to its eventual disposal. According to Zhang et al. (2012), A CMfg system is primarily made up of the manufacturing resources and capabilities, the manufacturing cloud, and the applications that cover the entire manufacturing life cycle.

The initial research about cloud manufacturing was originally carried out by a group of researchers in China and funded by the National Science Foundation of China (NSFC). One of the researchers Wu et al. (2013) define cloud manufacturing as

> *Cloud-Based Design and Manufacturing (CBDM) refers to a service-oriented product development model where service consumers are able to configure products or services as well as reconfigure manufacturing systems through Infrastructure-as-a-Service, Platform-as-a-Service, Hardware-as-a-Service and Software-as-a-Service in response to rapidly changing customer needs.*

Based on this definition, Wu, Schaefer & Rosen (2013) further articulate two aspects of CBDM: cloud-based design (CBD) and cloud-based manufacturing (CBM).

Ren et al. (2013) in their research tried to define the fundamental concepts used in cloud manufacturing, such as:

a. Manufacturing Resource

A manufacturing resource is an entity that can support anything that is needed in order to manufacture a product. This resource layer (R-Layer) is physical resource and include the hard resource (e.g. machine tools, machining centers, simulation equipment, and test equipment) and soft resource (e.g. computational model, data, software and knowledge in manufacturing process) (Wu et al., 2014).

b. Manufacturing Capability

A manufacturing capability is the ability of an organization or individual to perform a specific task in the manufacturing process. This can include resource advantages, intellectual advantages, and credits. A manufacturing capability can help an organization or individual to improve competitiveness and achieve specific targets.

c. Cloud Manufacturing Platform

A cloud manufacturing platform is a type of cloud computing platform that helps manage manufacturing resources and capabilities over a network. It offers integrated IT-based infrastructure and tools for both suppliers and demanders to release and to utilise cloud services on demand, respectively. Cloud manufacturing platform increase the communication between customers and manufacturers (Jacob et al., 2022). In other words, it will help to customize users desired and products request with user's personalization.

d. Cloud Manufacturing Service

A Cloud Manufacturing Service or Cloud Service is a function based on a manufacturing capability that can achieve a goal in an activity of a product life cycle, such as design service, production service, testing service and management service. It is described in a machine-processable format, and its information is under centralised management of a cloud platform. From a technical perspective, there are two basic types of cloud services:

OnCloud Service, which is in full control of a cloud platform, and OffCloud Service, which needs additional operations by an operator of a cloud platform.

e. Cloud User

Cloud user is an actor that involved to share the manufacturing resource, manufacturing capability, and manufacturing resources. Cloud user divided to three types: Cloud Provider is an user that provides manufacturing resources and capabilities as cloud services via cloud platform, and Cloud User is an user that use cloud services on cloud platform. Wu et al. (2012) add the following actor as Cloud Operator or Cloud Broker that who can operates and manga a cloud platform.

f. Resource Virtualisation

In resource virtualization, a real manufacturing resource is mapped to a logical one. This process can help to improve efficiency and flexibility in manufacturing. Resource virtualization is necessary to encapsulated the manufacturing resource, capabilities and service to be shared in cloud platform (Liu & Li, 2012). Moreover, resource virtualization also the fundamental for edge computing in cloud manufacturing to separate hardware resources from software, making it possible to run multiple tenants on the same hardware (Mansouri & Babar, 2021)

g. Capability Servitisation

The Servitisation process involves taking an abstract description of a manufacturing capability and transforming it into a standard cloud service that meets a specific set of specifications. This allows manufacturers to make their capabilities available to a wider range of customers and allows for easier integration of these capabilities into new and existing systems.

h. Cloud Manufacturing System

Cloud manufacturing system is a whole system that consists of cloud users, cloud platform, manufacturing resources, and manufacturing capabilities, and also provides specific applications in a manufacturing domain.

The fundamental definition above will guide this research to understand the term cloud manufacturing. Besides that, the previous research has several abbreviations to describe cloud manufacturing initiative research, a research conducted by Beihang University used CMfg to describe cloud manufacturing (Ren et al, 2013, Wu et al., 2013, Zhang et al., 2014), a research group from the University of Auckland use Cmanufacturing and presented it as an Interoperable Cloud-based Manufacturing System (ICMS) (Wang & Xu, 2012), a research group from Georgia Institute of Technology use Cloud-based design and manufacturing (CBDM) as a term to following research about cloud manufacturing (Schaefer et al., 2012, Wu, Rosen, Wang, Schaefer, 2015)), and Cloud-based Manufacturing as-a-service environment used by group research that funded by European Commission that has the purpose to develop a service-oriented architecture of cloud manufacturing environment (Meier, Seidelmann, Mezgar, 2010, Wang, Zhou, Jing, 2012). This research will use CMfg more often to describe cloud manufacturing.

2.3 Edge Computing

Edge computing is a type of computing where data is processed at the edge of a network, near the source of the data. This is in contrast to traditional centralized cloud computing, where data is processed in a central location. Edge computing is becoming increasingly popular as it can improve the user experience of terminal applications. One of the key challenges for edge computing is efficient resource scheduling, including decisions about when, where, and which resources to use for deploying different applications (Sahni, Cao, Yang, Wang, 2022). Approaches to control the several resources are to use the centralized controller with global knowledge, but there are many issues including poor scalability, bandwidth congestion while sending all the data to a single controller, and single point failure. In the novelty research, the approach is distributed scheduling resource (DRS) where managing of resources is distributed among several edge devices that communicate with each other.

Edge devices can pre-process, aggregate, and analyze sensory data closer to the data-generating sources (Nain, Pattanaik, Sharma, 2022). The intelligence in smart edge devices is derived from Machine Learning (ML) techniques. It comprises two phases: model training and inference, where the inference is generated in terms of classification and regression. The ML models need to be designed with significant

accuracy and confidence to be reliable enough to be used in the industry. This reliability increases the resource and computation requirements of the model in training and deploying larger and deeper ML models. The plenty of edge nodes in the edge layer are resource and computation scarce compared to an enterprise data center or the public cloud infrastructure.

There are several current research that discussed data offloading on edge devices, Wang et al. (2022) propose an edge control model in the network environment where the cloud edge is integrated. The edge node plays the role of controlling the processing in business logic. The cloud and edge effectively improve the ability to process the data flow of an intelligent user terminal through task coordination. Jian, Ping, and Zhang (2020) put proposed a two-level hybrid scheduling learning model as a method for cloud edge integration. Where First-level scheduling sorts and decomposes multiple tasks into several first-level sub-tasks according to the FIFO principle. These sub-tasks are then completed in the cloud. Second-level scheduling involves breaking down level-one sub-tasks into smaller, more manageable tasks, and then scheduling those tasks to different devices in different edge factory nodes. This allows for more efficient use of resources and prevents any one node from becoming overloaded.

2.4 Task Scheduling

Scheduling plays an important role in the efficiency of the Edge Cloud collaboration. There must be an efficient scheduling mechanism in the Edge Cloud that consider challenging issues like heterogeneous resources, inter-task dependence on user mobility, and variable user requirements. Scheduling is the process of allocating resources (computations) to a particular user who requests service (Asghar & Jung, 2022). It is challenging to decide whether to schedule a particular job at the edge or in the cloud in order to maximize resource utilization.

Scheduling in Industry 4.0 and cloud manufacturing is still at the beginning of its investigation. According to Dolgui (2019) in Ivanov, Sokolov, and Dolgui (2020) The results from various types of scheduling (hybrid shop scheduling and sequencing with alternative parallel machines, resource-constrained project scheduling with alternative chains, design of reconfigurable manufacturing systems and scheduling with both terminal and logical constraints) can now be integrated into a unified framework

of Industry 4.0. However, this framework requires an extension to models with hybrid structural-terminal-logical constraints.

According to Wang et al. (2022), The scheduling of traffic dataflows at the edge nodes of a network is designed to reasonably schedule network dataflow, regulate intelligent terminals, and manage resources at the edge of the network. The cloud server has the highest priority control over the entire network architecture, but the cloud has a large delay in request processing. For real-time requirements, the edge server layer can provide data computing and control services.

Currently, the cloud edge collaboration real-time task scheduling in cloud manufacturing is still at an early stage. Asghar & Jung (2022) categorized the scheduling algorithm in edge cloud into broader categories; heuristic and meta-heuristic. In heuristic algorithm includes the adaptive and dynamic approach, greedy approach, QoS parameter-baked approach, Machine Learning based approach, incentive based approach, and prediction-based approach. Furthermore, for meta-heuristic algorithm includes PSO based approach, an artificial fish swarm-based approach, and a genetic algorithm-based approach.

CHAPTER III

RESEARCH METHODOLOGY

3.1 Research Framework

The framework of this research follows guidelines from the research conducted by Talhi, Huet, Fortineau, and Lamouri (2020) about the methodology for CMfg architecture (analysis-specification-design-implementation) as a framework for designing the service-oriented architecture (SOA) in cloud manufacturing. Moreover, this research also guided by the existing research by Wang, Guo, Yu, Lui, and Deng (2022) that conducted research about task offloading in cloud-edge collaboration-based cyber-physical machine tools.

3.2 Research Flowchart

On the figure 3 below will illustrate the flowchart of this research.

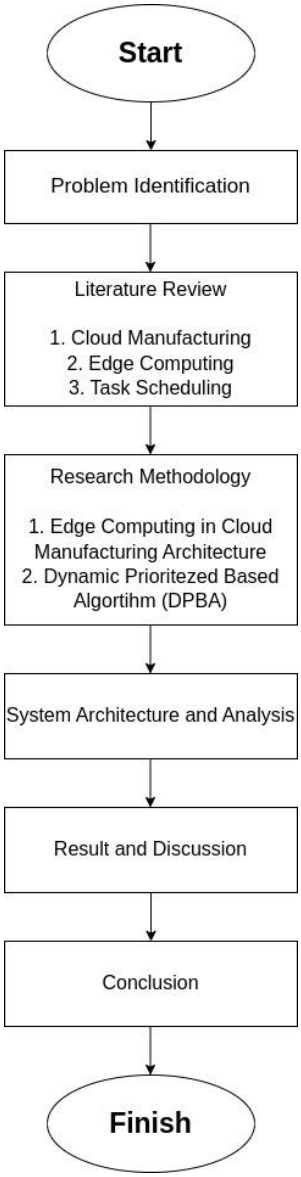


Figure 3.1 Flowchart Diagram

3.2.1 Problem Identification

This stage involves identifying the problem that will later become the focus of this research. In this study, the problem is identified in three stages:

1. Problem formulation

   This step outlines the extent of the problem that will be the focus of the investigation based on the identification of the problem in the form of questions.

2. Research Objective

   This step describes the objectives of the final output of the research.

3. Problem Limitation

   This step specifies the research limitations, which are designed to focus on the purpose. As a result, the research discussion will not stray too far from the research.

3.2.2 Literature Review

Literature references are used as the basis for thoughts in this research. Empirical references that use the same methods and techniques as this research are studied to gain a piece of new knowledge on the position of this research towards the application of these techniques. And other than that, a theoretical literature study is conducted to get an idea of the basis of the theories that are applied here.

3.2.2 Data Collection

In this research, dummy data is created in the form of order data to run a simulation. The goal is to study the behavior of the simulation using this artificially generated data, and the results are obtained based on the observations made during the simulation using 3d printer simulation software. The use of dummy data provides a controlled environment to test and evaluate the simulation, as well as eliminates any privacy or ethical concerns that may arise from using real-world data. The results obtained from the simulation using dummy data should be interpreted with caution, as they may not fully reflect real-world situations. However, the findings from this research can provide valuable insights and help to further validate the simulation before it is applied to real-world data.

3.2.3 Research Methodology

In this stage will present methodology of this research, for the analysis will conduct by designing the system architecture and determine the scheduling algorithm. These steps will describe in more detail below.

1. Edge Computing In Cloud Manufacturing Architecture

   A system architecture is a conceptual model that defines the structure, behavior, and views of a system (Jakkolla & Thalheim, 2011). An architecture description is a formal description and representation of a system that is organized in a way that supports reasoning about the system's structures and behaviors. This section provides an overview of the architecture of cloud-edge collaboration. The overall architecture consists of three layers: the manufacturing resource layer, the edge layer, and the cloud layer.

   In manufacturing, the resource layer covers all the physical items, including 3D printers. The edge layer includes constructing the processing actions, decomposing different task execution process actions, and scheduling the center, which directly interacts with the edge nodes and industrial devices or cloud centers, where each edge node can communicate with each other or with the edge scheduling system. Furthermore, the cloud layer is the application layer and interacts with the cloud user.

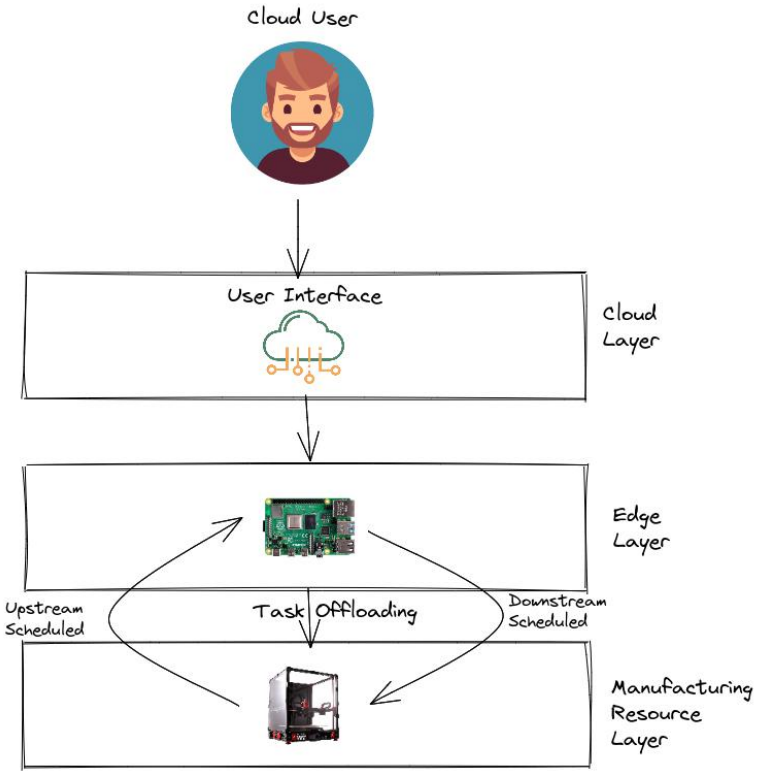


Figure 3.2 Task Scheduling in edge computing

2. Dynamic Prioritized Based Algorithm

The scheduling process involves making decisions about which jobs to execute on which computing nodes. task are submitted to a scheduler and then queued based on characteristics such as the amount and type of resources required, their execution time, and priority. One traditional scheduling strategy is to schedule task as they come in, known as first come, first served or in another word First In First Out (FIFO). The resource manager then takes queued jobs from the scheduler and allocates resources to them. While task are being executed, the resource manager constantly monitors them to know when to reclaim the allocated resources and move on to the next queued job.

Several factors are considered when scheduling jobs in an edge computing environment. Since computing nodes are physically distributed and connected over a wireless network, data transmission between nodes can be expensive and can slow down job processing if bandwidth is limited. To

reduce the network burden, latency-sensitive jobs are scheduled on the closest computing node that is able to run the job.

In this research proposed an algorithm called Dynamic Prioritized Based Algorithm (DPBA) a dynamic algorithm that prioritizes tasks based on their length and required manufacturing capabilities. In this algorithm, shorter tasks and tasks with higher manufacturing capabilities are given higher priority, as they are more important and need to be completed quickly and efficiently. This helps to ensure that the 3D printer is utilized efficiently and that important tasks are completed in a timely manner.

3.2.4 System Simulation and Analysis

In this section will present the simulation the system architecture and the DPBA as the scheduling algorithm. The simulation running in two scenarios, the first scenario is the system will be implement in cloud server for cloud layer and respberry as the edge layer. The second scenario the system will be implement in local computer to try simulation of DPBA. In this section will use several tech stack such as:

a. Rabbit MQ

RabbitMQ is an open-source message broker software that implements the Advanced Message Queuing Protocol (AMQP). It allows applications to communicate and exchange messages through a common platform, enabling decoupled and asynchronous communication between microservices. In this research RabbitMQ will help to become the message protocol between the cloud layer and the edge layer.

b. MongoDB

MongoDB is a NoSQL document-oriented database management system. It stores data in semi-structured format as documents, rather than in a tabular relational database. It is horizontally scalable, flexible and provides high performance for handling large amounts of unstructured data. MongoDB is commonly used for web, mobile, geospatial, and analytics applications. In

this simulation MongoDB will be the database to store the order data from the customers.

c. Octoprint

OctoPrint is a web-based control and monitoring software for 3D printers. It runs on a small computer connected to the printer and provides a user-friendly interface for controlling and monitoring the printing process, uploading and slicing files, and managing the printer's settings. OctoPrint also offers features such as time-lapse video creation, printer status monitoring, and a plugin system for adding additional functionality. In this research octoprint will act as the 3D printer to simulate the proposed architecture. The 3D printer will be deploy on Raspberry as the edge layer.

3.2.5 Results and Discussion

After the system is built, a discussion will be conducted to discuss the problem formulation and the result of data processing. The interpretation of outcome data is explained through discussion.

3.2.6 Conclusion

This section will briefly answer the question posed in the previous section on how best to deal with the problem of increasing traffic congestion. In addition, this section provides recommendations for further research on the subject.

CHAPTER IV

System Architecture and Analysis

4.1 Cloud Edge Architecture

System Architecture is a conceptual model that defines a system's structure, behavior, and view. An architectural description is a formal description and representation of a system, organized in a way that supports reasoning about the system's structure. System architecture may consist of system components, the externally visible properties of the components, and the relationships between them. It can provide a plan from which the product can be obtained, and the system developed, which will work together to implement the system as a whole. There have been attempts to formalize languages for describing system architecture, collectively called architecture description languages.

The fundamental idea of the cloud manufacturing system architecture concept is to encapsulate manufacturing resources and manufacturing capabilities as well as capacities in networks (manufacturing clouds) and make them available as services, according to the requirements and at the request of consumers, through relative service platforms. In the other words, manufacturing resources and capacities are transformed into production services that can be managed and operated in an intelligent and unified way. Cloud manufacturing reflects both the concept of "integration of distributed resources" as well as the concept of "distribution of integrated resources".

The rising number of data nodes in a manufacturing system leads to an increase in data volume and more detailed descriptions of equipment. This research uses the concept of cloud-edge collaboration to construct a data communication network architecture in a manufacturing environment that allows for the reuse of mixed-flow manufacturing and processing experiences. The edge functions include constructing the processing actions' knowledge ontology, decomposing different task execution processes' actions, orderly combining the existing knowledge action primitives to

obtain complex task-oriented processing procedures, storing the local resources and knowledge, and performing the matching for knowledge reuse. The cloud functions include deploying the overall manufacturing environment knowledge base, enabling the interaction between users and managers and the system, and supplementing and updating the edge-side knowledge in a timely manner.

For this research, the system architecture will be composed of three main layers: manufacturing resource layers, edge layer, and cloud layer. The manufacturing resource layer comprises a variety of 3D printing resources that are spread out geographically, e.g. 3D printers, 3D printing services, etc. Utilizing pre-established communication protocols, these 3D printers can be connected to the web and accessed by customers. The 3D printing services constitute the specialized services that 3D printing service providers can present. These will be systematized into a common data structure, which can be interpreted as various cloud services. The edge layer is the edge node that will offload tasks from the cloud and scheduled tasks that can be printed by the physical device. The cloud layer provides the ability to deploy a comprehensive understanding of the manufacturing environment, facilitate communication between users and executives and the system, and supplement and promptly update the knowledge stored at the edge. For more understanding, the architecture illustrates in Figure 5 below.

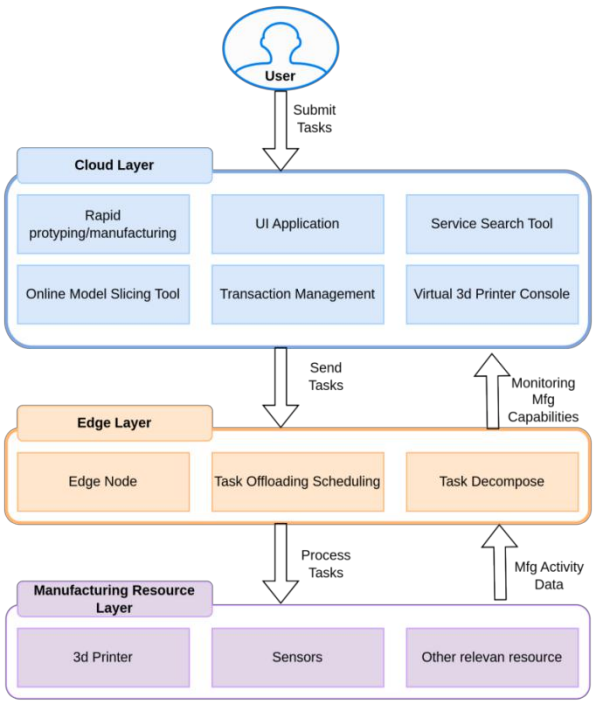


Figure 4.1 High-level Cloud Edge Architecture.

The manufacturing task requests submitted by users are first handled in the cloud. After that, these complex tasks are broken down into smaller subtask workflows, which are then transmitted to the edge scheduling center. The edge scheduling center then performs scheduling and further deconstructs the subtasks into atomic tasks, allocating them to the industrial equipment of each edge factory node in real-time. The overall processing flow of complete manufacturing is depicted in the following flowchart.

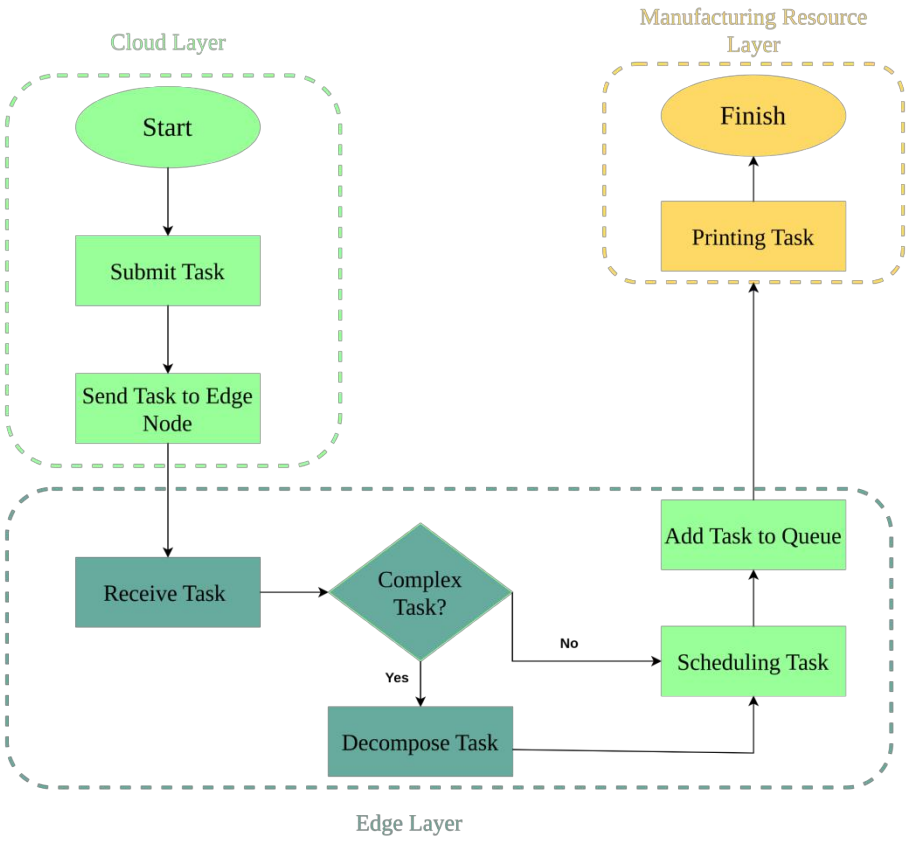


Figure 4.2 Flowchart Cloud Edge 3D Printing Process.

The processing flow of complete manufacturing tasks is, *firstly* the user submits the manufacturing task request to the cloud platform. For the complex manufacturing task, the cloud center scheduling system sorts tasks according to the FIFO principle. Secondly, the cloud center scheduling system transmits the decomposed first-level sub-tasks to the edge scheduling center in real time. Third, in the edge node, the task will decompose into atomic task workflow in real-time based on the FIFO principle. Then

the edge scheduling will use the FIFO, which calculates the scheduling solution in real-time based on the load information of each edge factory node. Finally, the scheduling model will offloads atomic tasks to a 3D printing controller.

4.1.1 Cloud Layer

The cloud layer is the first layer of the cloud edge manufacturing platform, and it is the point of interaction between the cloud user and the platform. This layer provides a range of services and tools that enable users to access and use the capabilities and resources of the manufacturing platform in a convenient and user-friendly way.

The cloud layer provides services and tools for a variety of manufacturing-related tasks, such as product design, prototyping, slicing, and service search. For example, the cloud layer may provide rapid prototyping tools that enable users to quickly create and test 3D models of their products, or online slicing tools that enable users to prepare digital models for 3D printing. The cloud layer may also provide service search tools that allow users to find and compare different manufacturing services, such as 3D printing. In addition, the cloud layer may provide transaction management tools that enable users to manage and track their orders and payments.

By providing a range of services and tools, the cloud layer enables users to access and use the capabilities and resources of the manufacturing platform in a seamless and integrated way. This can improve the efficiency and effectiveness of the manufacturing process, and it can enable users to take advantage of the benefits of the cloud edge manufacturing platform, such as access to a wide range of resources and capabilities, increased productivity and collaboration, and improved product quality.

In this research the edge layer will use in the Google Cloud Platform and deployed in Google Compute Engine with specification are:

a. Processor EC2 8 GB
b. Server Location in Southeast Asia 2 - Jakarta a
c. The operating system is Ubuntu 20.04

4.1.2 Edge Layer

The edge layer of the cloud edge manufacturing platform is made up of edge nodes, which are devices and resources located at the edge of the network. These edge nodes are connected to the cloud layer through the network, and they enable data and information to be exchanged between the cloud and the edge.

The main function of the edge nodes is to provide access to the manufacturing devices and resources at the edge of the network, such as 3D printers, CNC machines, and injection molders. These devices and resources are typically located close to the users and the products, and they enable the production of products in a decentralized and distributed way. The edge nodes enable the cloud user to remotely control and monitor these devices and resources, and they enable the collection and processing of data and information from these devices and resources.

In addition to providing access to manufacturing devices and resources, the edge nodes also provide capabilities for data analytics, real-time communication, and data storage. These capabilities enable the edge nodes to support the operations and functionality of the manufacturing platform, and they enable the integration and coordination of the cloud and the edge.

Overall, the edge nodes are an important part of the cloud edge manufacturing platform, as they enable the exchange of data and information between the cloud and the edge, and they support the decentralized and distributed production of products.

In this research the edge layer as edge node will use a Raspberry pi 3 model B, a small, low-cost computer that was developed for educational purposes. The specification of raspberry will describe below:

a. 1.2 GHz Quad COre ARM Cortex-A53 Processor
b. 1 GB RAM
c. 16 GB Micro SD card
d. 802.11 b/g/n/ac wireless LAN
e. Bluetooth 4.2
f. Operrating system is Raspbian.

4.1.3 Physical Resource Layer

In the context of the cloud edge manufacturing platform, the physical resource layer is the layer that provides access to the manufacturing devices and resources at the edge of the network. These devices and resources are the physical resources that enable the production of products, and they can include a variety of technologies and materials.

The edge devices are connected to the edge nodes through a local network, and they enable the production of products in a decentralized and distributed way. The edge devices can be controlled and monitored by the cloud user through the edge nodes, and they can provide data and information to the edge nodes for analysis and storage.

4.2 Design Cloud Edge Collaboration System

In this paper, we provide a generic architecture for edge-cloud collaborative computing. This section covers the structure of the edge-cloud mechanism, the task scheduling strategy for edge-cloud collaborative computing, and the details of the overall system architecture for scheduling optimization.

4.2.1 Overview of Cloud Edge Collaboration System

In the cloud manufacturing platform, the physical manufacturing equipment is virtualized into manufacturing services depending on their suitability to perform certain manufacturing operations. The capability and availability of the manufacturing equipment are the two main factors determining its suitability. The capability of the manufacturing equipment is generally fixed since its structure and functions remain the same.

There are three actors involved in a CMfg system: service providers, cloud operators, and service users. The system includes manufacturing resources and capabilities, the manufacturing cloud, and applications for the entire manufacturing life cycle. Additionally, it has knowledge-based management, and two processes (upstream and downstream), as illustrated in Figure 7.

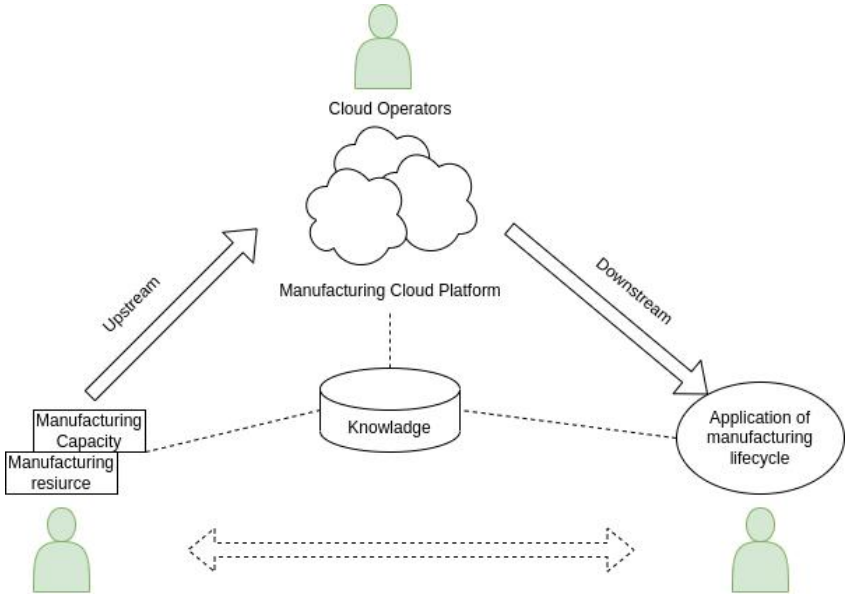


Figure 4.3 Abstract Operation of CMfg

This approach uses an effective task scheduling strategy to efficiently select a suitable server for task processing. This method can reduce the latency of cloud computing while ensuring that tasks are completed within the specified deadline. By choosing the right edge server to handle time-sensitive processes, it is possible to avoid dispatching tasks to the cloud and save on bandwidth and execution costs.

As shown in Figure 1, IoT applications serve users' processing needs according to their demands. Other devices such as machines in smart factories and automobiles generate service requests, with the necessary metadata provided by sensors and actuators. Edge servers manage the underlying middleware and process data. In the edge-cloud collaborative architecture, the cloud infrastructure is responsible for the long-term storage of processed data and application outcomes. To process data at the edge, the middleware must have the ability to handle real-time data. In this architecture, connections between devices are mostly maintained through industrial networks.

The edge servers act as the primary processing module. On one hand, they must receive input data from edge sensors and perform pre-processing. On the other hand, they are responsible for using available resources or retrieving necessary information from the cloud to process data and return results to users. Edge servers in close proximity process data from end devices, but their storage, RAM, bandwidth, and processing capacity are limited.

When an edge server receives a task processing request, it will first try to use its current available information for processing. If running applications fully utilize the available resources, the task may be forwarded to neighboring edge servers for processing. In this case, edge servers should periodically send a "signal" to the broker to update it on their status and available resources. In the three-tier architecture of edge-cloud collaborative computing, end devices are connected to the edge server and the edge servers are connected to the cloud.

4.2.2 Communication Protocol

In the context of the cloud edge manufacturing platform, the communication protocol between the cloud layer and the edge layer can be implemented using the Advanced Message Queuing Protocol (AMQP) and the RabbitMQ messaging broker.

The AMQP is a standard protocol for message-oriented middleware, and it defines a common format and semantics for the communication of messages between different systems and components. The AMQP enables the exchange of messages between the cloud layer and the edge layer in a reliable, secure, and interoperable way.

The RabbitMQ is a popular open-source implementation of the AMQP, and it provides a scalable and flexible messaging broker that can support the communication and coordination of the cloud layer and the edge layer. The RabbitMQ can be deployed and configured in the cloud and the edge, and it can enable the exchange of messages between these layers using the AMQP.

By using the AMQP and the RabbitMQ, the communication between the cloud layer and the edge layer of the manufacturing platform can be efficient, reliable, and scalable. The AMQP and the RabbitMQ can enable the integration and coordination of the services and tools provided by the cloud layer with the devices and resources provided by the edge layer, and they can support the overall operations and functionality of the manufacturing platform.

4.2.3 Task Execution Model

The edge server needs a data center as its primary container to run, and a host is established to run the server using virtualization software to provide the CPU and memory resources required for the edge server. Similarly, edge servers also need to access graphics cards, data stores, and network connectivity from their higher-level host. In the space-shared scenario, tasks are executed according to the first-in-first-out (FIFO) rule. A machine can only process one task at a time, and if a task is executed on the machine when other tasks arrive, the task must wait for the task on the machine to finish before it can start executing.

In the edge-cloud collaborative environment, tasks generated by mobile devices can be offloaded to cloud servers for local computing or leveraged for cloud resources. However, sending tasks to the cloud requires additional communication time and creates bandwidth pressure. The total task execution time (from the initiation of the request to receiving a response) can be formulated using Equation (1).

$$Ttot = Tedge + Tproc\_edge + Tcloud + Tproc\_cloud \quad \textbf{(1)}$$

Where the total time from when a task request is sent until the user receives the result of the calculation is represented by *Ttot*. Tasks need to spend on *Tedge* to establish a connection with the server. Once the task reaches the server through the connection, it will not start until all previous tasks are finished, if there are tasks being executed on the server. After the task starts execution, the actual execution time (*Tproc_edge*) varies depending on the size and difficulty of the task and the processing power of the server. If there is no nearby server that can complete the task within the user's desired deadline, the server will request resources from the cloud, which increases the time required to establish a connection with the cloud (*Tcloud*), and the task will take *Tproc_cloud* to execute the remaining tasks while still on the cloud.

In order to reduce network traffic on the core network and improve processing power and bandwidth, it is important to complete computations at the edge as much as possible. This is especially important for time-sensitive applications, where the task must be completed and the result returned before the deadline. When a task is to be

executed on an edge server, the execution time of the task can be calculated using the following equation.

$$Tproc\_edge = Twait + Transfer + Texec \quad \textbf{(2)}$$

When a task request is submitted, it must first spend time to establish a connection with the allocated edge server and then provide user data as execution parameters. Since separate thread transfer data in our model, the time to send data is almost negligible. After being offloaded, the task must be allocated to an appropriate server for execution. In our scheduling system, priority is given to disposing of it on the single machine for execution to reduce communication time. If the execution cannot be completed on the nearest server within the specified stringent deadline, the agent scheduling center must dispatch the application to a nearby server that can perform the task in time for immediate execution, utilizing server communication.

Furthermore, choosing a short and quick task-waiting queue can reduce task-waiting time, as it takes into account the time it takes to transfer data to the server before tasks begin executing. Although jobs must wait for the CPU, data transfer happens through multiple separate threads immediately after jobs are submitted.

In this research, the cloud server is responsible for resource allocation and task scheduling, allowing the edge server to focus solely on processing tasks without spending time on security and other concerns. This integration of the complete task-processing flow of edge computing improves the high availability and efficiency of computing scenarios through model optimization. Based on the optimized model, an effective scheduling strategy is developed to further enhance the efficiency of task processing.

4.3 Design the Scheduling Algorithm

In the cloud edge collaboration, the scheduling adopts the static scheduling model. It focuses on the present solution for the arrival application in the edge cloud collaborative environment. To meet the deadline objective while considering dynamic client requirements in the edge-cloud collaborative environment, scheduling optimization will be performed in the following steps:

(1) Submission Strategy.

The submission strategy is an important part of the edge-cloud collaborative environment. It involves submitting tasks to the cloud for execution, and the edge server requesting resources from the cloud. However, this can increase communication time and cost overhead, so it is important to maximize the use of the edge server for executing tasks and try to intercept tasks to complete execution on the edge servers in order to avoid submitting to the cloud.

For example, when a cloud user submits a task to the cloud server, the server should prioritize the tasks based on their length, with shorter tasks being given higher priority. This will help to ensure that shorter tasks are completed quickly and do not get stuck behind longer tasks in the queue.

The server should also prioritize tasks based on the required manufacturing capabilities of the 3D printer. Tasks that require higher manufacturing capabilities should be given higher priority to ensure that the 3D printer is utilized efficiently.

By implementing this submission strategy, we can improve the efficiency of task scheduling in the simulation and ensure that important tasks are completed quickly and efficiently.

(2) Servers Comprehensive execution capability

Servers have a comprehensive execution capability, which means they are capable of running a wide range of tasks and processing large amounts of data. This is an important characteristic for scheduling algorithms, as it allows the server to handle a large number of tasks and prioritize them based on their length and required manufacturing capabilities.

The comprehensive execution capability of servers allows for the decomposition of long tasks into atomic tasks with a length of no more than 1000. This helps to prevent the 3D printer from being overburdened with long tasks and ensures that they can be completed in a timely manner.

Overall, the comprehensive execution capability of servers is an important factor in the efficiency of task scheduling algorithms and can help to improve the overall efficiency of cloud manufacturing processes.

(3) Prioritize Based Task Scheduling

To implement a dynamic task scheduling algorithm in the proposed scheduling strategy, we can add additional functionality to the edge node class. This functionality would allow the edge node to monitor the progress of tasks being printed and adjust the scheduling as needed to ensure efficient utilization of the 3D printer.

For example, the edge node could implement a monitor_progress method that is called periodically to check the status of each task in the queue. If a task is taking longer than expected to complete, the edge node could adjust the priority of other tasks to ensure that the 3D printer is not idle. This would allow the edge node to optimize the scheduling of tasks in real-time and improve the efficiency of the 3D printing process.

Additionally, the edge node could use machine learning techniques, such as deep learning, to predict the completion time of tasks based on their length and required manufacturing capabilities. This would allow the edge node to more accurately predict the time required to complete tasks and adjust the scheduling accordingly.

To handle the task scheduling strategy proposed, the following steps can be taken:

1. When a cloud user submits a task to the cloud server, the server prioritizes the tasks based on their length, with shorter tasks being given higher priority.
2. The server also prioritizes tasks based on their required manufacturing capabilities, with tasks that require higher capabilities being given higher priority.
3. The server sends the prioritized tasks to the edge node, where they are placed in a queue.

4. When a task enters the queue, the edge node checks its length. If the task length is complex, it is decomposed into atomic tasks, with each atomic task.
5. The edge node schedules the tasks for 3D printing based on their priority, with higher priority tasks being scheduled first.

By implementing these steps, the proposed scheduling strategy can effectively handle the dynamic behavior of users and provide efficient and responsive task processing in cloud manufacturing processes.

4.3.1 Dynamic Prioritized Based Algorithm

The proposed algorithm is Dynamic Prioritized Based Algorithm (DPBA), this algorithm is a scheduling strategy designed to improve the efficiency of task processing in cloud manufacturing processes. It is a dynamic algorithm that is able to handle complex and changing environments, where the availability and characteristics of tasks and resources can vary over time.

DPBA prioritizes tasks based on their length and required manufacturing capabilities, with shorter tasks and tasks with higher manufacturing capabilities being given higher priority. This helps to ensure that important tasks are completed quickly and efficiently, and that the 3D printer is utilized efficiently.

When a task is sent from the cloud server to the edge node, it is placed in a queue, where it will be held until it is ready to be scheduled for 3D printing. If a task has a length greater than 1000, it is decomposed into atomic tasks, with each atomic task being given a length of no more than 1000. This helps to ensure that the 3D printer is not overburdened with long tasks and can complete them in a timely manner.

DPBA also monitors the progress of the tasks being printed and adjusts the scheduling as needed to ensure efficient utilization of the 3D printer. This helps to improve the performance and scalability of the system and provide better services to users. Overall, DPTS is an effective algorithm for scheduling tasks in cloud manufacturing processes.

The proposed algorithm uses a mathematical model to calculate the priority and completion time of tasks. The set of tasks submitted to the cloud server is represented by T, and the length of a task t is represented by L(t). The priority of a task is represented by P(t), which is calculated based on the length and required

manufacturing capabilities of the task. The queue of tasks at the edge node is represented by Q, and the completion time of a task t is represented by C(t). The start time of a task t is represented by S(t), which is calculated as the maximum of the completion time of the previous task and the current time.

The proposed algorithm uses a mathematical model to calculate the priority and completion time of tasks. The set of tasks submitted to the cloud server is represented by T, and the length of a task t is represented by L(t). The priority of a task is represented by P(t), which is calculated based on the length and required manufacturing capabilities of the task. The queue of tasks at the edge node is represented by Q, and the completion time of a task t is represented by C(t). The start time of a task t is represented by S(t), which is calculated as the maximum of the completion time of the previous task and the current time.

The priority of a task is calculated as where a and b are constants and MC(t) is the required manufacturing capabilities of task t, the priority task can be calculate following the Formula (3) below.

$$P(t) = aL(t) + bMC(t) \quad \textbf{(3)}$$

The completion time of a task is calculated following the Formula (4) where S(t) is the start time of the task and L(t) is the length of the task.

$$C(t) = S(t) + L(t) \quad \textbf{(4)}$$

Using these equations, we can calculate the priority and completion time of each task and schedule them efficiently to ensure efficient utilization of the 3D printer.

Table 4.1 Dynamic Prioritized Based Algorithm Pseudocode

Algorithm 1 Dynamic Prioritized Based Algorithm

Input: *Task_Q <task>: task waiting in queue; ServerSubmitList <server>: Cloud server that handle requests; EdgeNode <edge>: edge node receive the task from cloud server; ti: the i-th task*

```
Init EdgeQueue・ EdgeNode <edge>
TaskList <task>・ Task_Q <task>
PrioritazedTask
for each taski・ PrioritizedTask do
        sort Lt & MCt
end for
if Lt > 1000 then
        num_atomic_task・ round(Lt/1000)
        for i・ range(num_atomic_task) do
                A = T(Lt=min(Lt <1000))
                EdgeQueue・A
            end for
        sort EdgeQueue
        for T in EdgeQueue do
            if 3dP = idle then
                Printing T
            elif T(Pt) > ti(Pt) then
                stop ti
                printing T
             elif T(Pt) < ti(Pt) then
                 continue ti
             end if
        end for
        continue
```

Following the pseudocode above, the scheduling process happen after task decompose task in edge node. After that the task will sorting based on the priority level. Thera are three priority level; high, medium and low. High is the highest priority, this priority will come up to the first priority when scheduled the task to 3D printing. The second priority is Medium and the last priority is low. To create more understanding about pseudo code, the notions will attach in Table 3.

Table 4.2 Notations

Symbol	Meaning
T	Tasks submitted to Edge Node
Lt	Length of the task t
Pt	Priority of task t
Q	Queue of the task
Ct	Completation time of task t
St	Start time of task t
MCt	Manufacturing Capability task t
ti	The i-th task
A	Atomic Task

The scheduling algorithm will implement in cloud edge simulation. The simulation system is made with the Python programming language. To conduct the simulation, each entity in cloud edge scheduling simulation will be represented by class. There are five class and two function such as main function and decompose function. Each class will have several attributes, for example in edgeNode class will have attributes decompose_task, schedule_task, and monitor_progress. The class attributes perform as the method that will be called when performing the simulation. Classes in this simulation will have relationship to another class to conduct the simulation well.

CHAPTER V

RESULT AND DISCUSSION

This chapter will divided into two section. First section will discuss about system implementation Cloud Edge Collaboration and second section will discuss about implementation and simulation of task scheduling for edge computing in CMfg environment.

5.1 Cloud Edge Collaboration System

5.1.1 Task Submission In Cloud Layer

In this research, task orders will be made dummy with the help of the fake python library. The parameters that are created in an order are the user name, address, and the gcode file. For further explanation in table 4.

Table 5.1 Task Parameter Field

Task Parameter	Description
Name	The name of the customer who ordered the task
Adress	The address of the user who ordered the task.
Gcode File	The g code file to be printed
Size	Size of gcode file
Order Date	Date when the task was submitted by the user
Order Status	Status order task

A dummy task is a simplified or mock version of a real task that is used for testing or demonstration purposes. In the context of a field are “Name” as the name customer who ordered the task. "Address" for the customer address who ordered the task, “Gcode” file as the gcode file that to be printed, “Size” for file size of the gcode file, “Order Date” for the time when the customer place the order, and the last “Order Status” for production status of the task. This could be useful for testing a database or user interface without using real customer information. The code to create the dummy will present in Figure 5.1 and Figure 5.2.

```python
tasks = int(os.environ.get("tasks"))


def create_dummy():
    for i in range(tasks):
        start_time = time.time()
        order_id = randrange(1, 1000, 1)
        name = fake.name()
        address = fake.address()
        gcode = generate_gcode()
        data = {
            "order_id": order_id,
            "name": name,
            "address": address,
            "gcode": gcode[0],
            "file_size": gcode[1],
            "datetime": str(datetime.datetime.now()),
            "order_status": ""
        }
        save_data = SaveData()
        save_data.save(data)
        send_task(name)
        print(f"orders from {name} saved into db at {datetime.datetime.now()}")
```

Figure 5.1 Code to create dummy order

```python
def generate_gcode():

  file_sizes = randint(10000, 15000)

  # Inisialisasi string G-code

  gcode_str = "; generated by PrusaSlicer 2.3.1+win64\n"
  gcode_str += "; external perimeters extrusion width = 0.45mm\n"
  gcode_str += "; perimeters extrusion width = 0.45mm\n"
  gcode_str += "; infill extrusion width = 0.45mm\n"
  gcode_str += "; solid infill extrusion width = 0.45mm\n"
  gcode_str += "; top infill extrusion width = 0.40mm\n"
  gcode_str += "; first layer extrusion width = 0.42mm\n"

  for i in range(file_sizes):

    x = randint(1, 30)
    y = randint(1, 30)
    z = randint(1, 30)
    gcode_str += f"G00 X{x} Y{y} Z{z}\n"

  return gcode_str, file_sizes
```

Figure 5.2 Code to G-Code

In the Figure 5.1 it can be seen the code to create dummy order, the initial step is write the function name *create_dummy()*. The *create_dummy()* function that generates dummy data and saves it to a database. The function has a for loop that iterates over a range of *task*. Inside the loop, the function generates some random data using the *fake* and *randrange* modules.

This code in Figure 5.2 defines a *generate_gcode()* function that generates a G-code. G-code is a programming language used to instruct CNC (computer numerical control) machines (such as 3D printing) how to manufacture a part. It consists of a series of commands that tell the machine what actions to perform and in what order to perform them. The function begins by generating a random *file_sizes* value between 10,000 and 15,000. It then initializes a string *gcode_str* with some default values for various G-code parameters. The function has a for loop that iterates over a range of file_sizes. Inside the loop, the function generates random *x, y, and z* values and *appends* them to the *gcode_str* in the form of a G-code G00 command. This process continues until the loop has iterated *file_sizes* times, at which point the *gcode_str* is returned along with the *file_sizes* value. This *generate_gcode* function can be used to generate a G-code with a specified number of lines.

Furthermore, after create dummy order, the data will store in a dicioinary file type. Then saves this data to a database using an instance of the *SaveData* class. It also sends a task using the *send_task* function and calculates the time it took to complete the task. This calculated time is saved to a list called cloud_proc_time. Finally, the function writes the a CSV file that contains of the total processing time in Cloud Server.

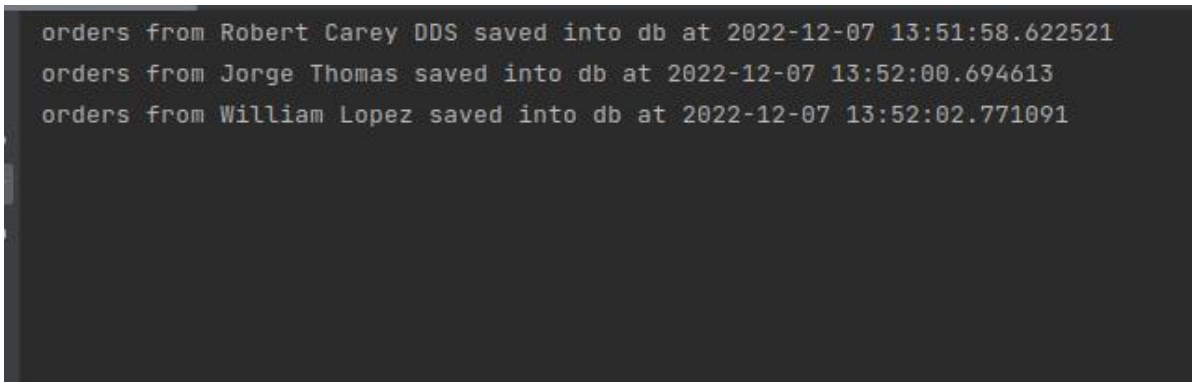


Figure 5.3 Dummy Task Submission process

The dummy order data saved into Cloud Server Database, in this case the database is MongoDB. MongoDB is one of the database management system that uses

a NoSQL database model. NoSQL databases are a type of database that does not use the traditional SQL (Structured Query Language) for defining and manipulating the data. Instead, NoSQL databases use a variety of data models, including key-value, document, columnar and graph formats, to store and retrieve data. This allows them to be more flexible and scalable than SQL databases, which can make them a good fit for certain types of applications. The example saved data in MongoDB illustrate in Figure 5.4.

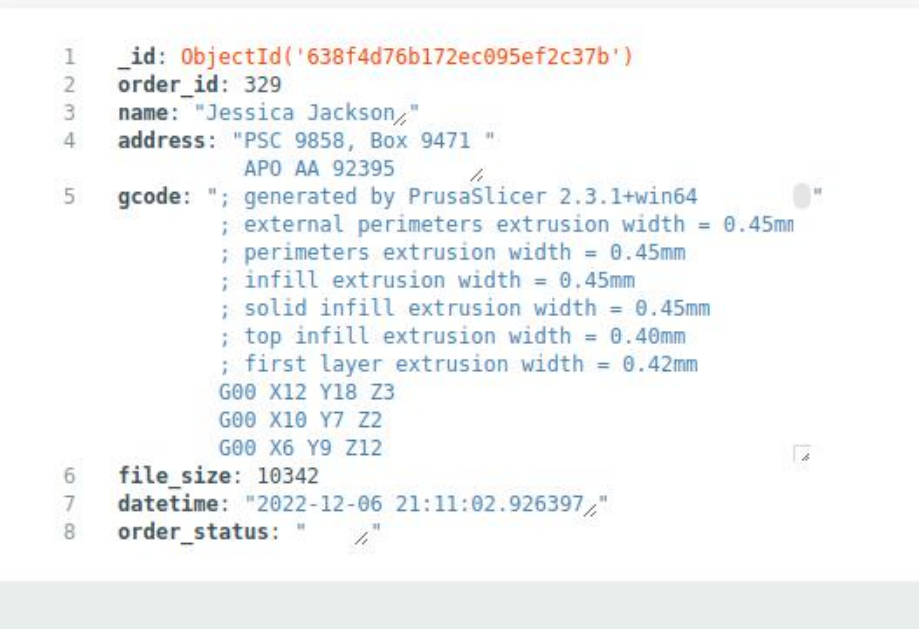


Figure 5.4 Dummy Task Submission process

The last step in the cloud server process involves sending data from the cloud server to a message broker. In this step, the cloud server acts as a producer, sending tasks to the message broker. The message broker is deployed at the local cloud server and is responsible for encoding the order task into a binary file. This is necessary because the AMQP protocol, which is used for communication between the cloud server and the message broker, can only send data in binary format. Before sending the message to the message broker, the cloud server queries the order data from the database. This ensures that the task being sent to the message broker contains the latest information about the order. The message broker then uses this information to encode the task into a binary file, which is then sent to the cloud server for further processing. To illustration for query order task from database and code to send order data to Message broker present in Figure below.

```python
def get_order_tas(db, coll, id):
    try:
        task = db[coll].find_one({"name": id})
        return task
    except Exception as e:
        print(e, "task not found")
```

Figure 5.5 Query Order Task from MongoDB

```python
def send_task(task_id):
    # Create a connection to the RabbitMQ server
    connection = BlockingConnection(
        parameters=pika.ConnectionParameters(
            host="localhost",
            port=5672,
            credentials=pika.PlainCredentials("guest", "guest"),
        )
    )

    # Create an exchange
    channel = connection.channel()
    channel.exchange_declare(exchange="myexchange", exchange_type="direct")
    channel.queue_declare("myqueue", durable=False)
    timestamp = int(time.time() * 1000)
    properties = pika.BasicProperties(
        delivery_mode=2,
        timestamp=timestamp
    )
```

Figure 5.6 Send Data to Message Broker

The Figure 5.6 show lines of code creates a connection to a message broker server and then creates an exchange on that server. The code specifies the host and port of the message broker server and provides credentials for authenticating the connection. It then creates an exchange with the name *myexchange* and specifies the exchange type as *direct*. The code also creates a queue called *myqueue* and sets its durability flag to *False,* it is mean the message will lose when the message broker server is down. This *BasicPropoerties* object is used to configure the properties of messages that are sent over the exchange. Besides sending messages, message brokers also timestamp to edge node that will be used to calculate total ffessing time as described in Formula (1). Also

the time stamp will be used to calculate the total delay (Dt) with diminish the time tasks are received by the Edge Node (Receive_t) by the time tasks are sent from the Cloud (Sende_t). The formula shown below.

$$Dt = Receive\_t - Send\_t \quad \textbf{(5)}$$

5.1.2 Communication Protocol With AMQP

Communication protocol is a set of rules and conventions that govern how two or more entities communicate with each other. AMQP (Advanced Message Queuing Protocol) is a communication protocol for message-oriented middleware. It is an open standard that defines a network protocol for messaging. AMQP provides a way for applications to communicate with each other by sending and receiving messages over a network. AMQP allows applications to exchange messages in a flexible and reliable way, without requiring them to be tightly coupled or to have a priori knowledge of each other. This makes it a useful protocol for building distributed systems that need to process and route large volumes of data in real-time.

In this research, to implement AMQP protocol then will be used a RabbitMQ as middleware between cloud server and edge node. RabbitMQ is an open-source message-oriented middleware that implements the AMQP (Advanced Message Queuing Protocol) communication protocol. It is a popular choice for building distributed systems that need to process and route large volumes of data in real-time. RabbitMQ will act as a middleware, cloud server will act as a producer, and node edge will act as subscriber or consumer.

After send order task to message broker that show at Figure 5.7, the data will save into Virtual Host in Rabbit MQ server. RabbitMQ has User Interface (UI), making it easier for the user to monitor the sending and receiving task. Besides of that, RabbitMQ also provide a queue In RabbitMQ, a queue is a buffer that stores messages until they can be delivered to their intended recipients. Queues are the primary mechanism for storing and routing messages in RabbitMQ. When a client sends a message to a RabbitMQ broker, the broker routes the message to one or more queues based on a set of rules defined by the user. Figure 14 below shows the chart of the queues in the RabbitMQ server.

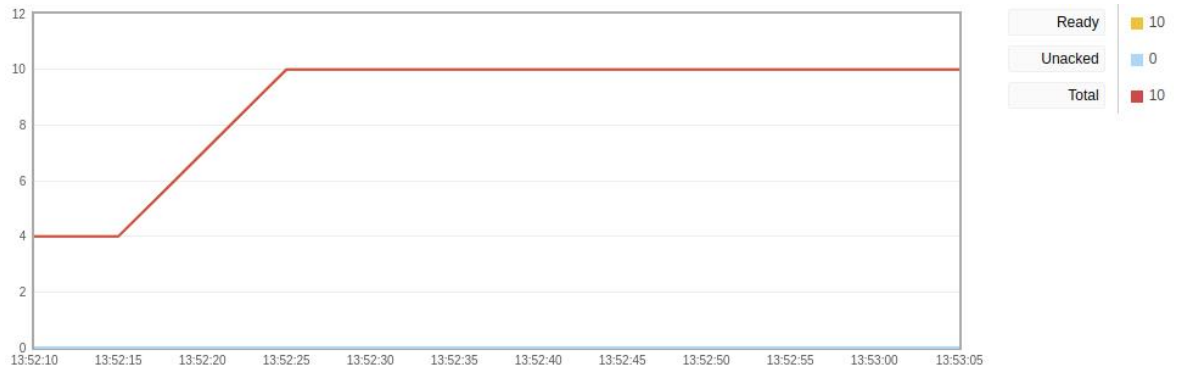


Figure 5.7 Queue Process on RabbitMQ Server

Figure 15 below shows the message rate that come from edge cloud to message broker. Based on the figure below that can see the average publish speed from cloud server to RabbitMQ server is 0,40/s, this means that on average, the cloud server is sending 0.40 messages to the RabbitMQ server every second. This is just an average, so the actual number of messages sent in any given second may be higher or lower than 0.40. The message rate can vary depending on a number of factors, such as the workload of the cloud server, the configuration of the RabbitMQ server, and the network conditions between the two. The ability message broker to received task from cloud server determine by the size of the message (task) and computer hardware where the RabbitMQ hosted.

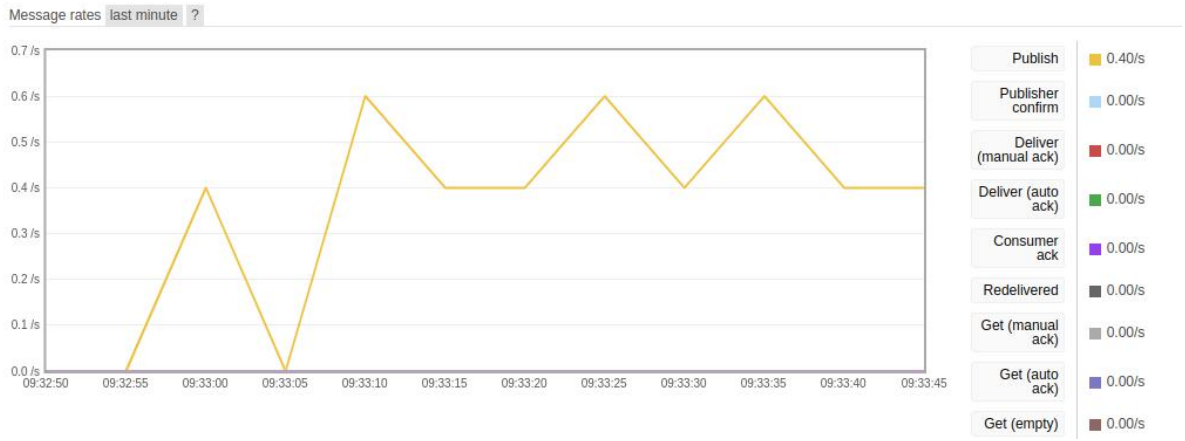


Figure 5.8 Message Rates Chart on RabbitMQ Server

5.1.3 Scheduling Task in Edge Node

Edge node known as an edge computing, the place where the task will processed to scheduling before send the task to physical manufacturing resource such as 3D printing. The edge node location is located at the edge of a network, close to the physical manufacturing research. This can improve the performance and efficiency of the system by reducing the amount of data that needs to be transmitted over the network, and by reducing the latency of the data processing.

For this research to build an edge node will used an a raspberry pi 3, a single board computer, meaning that all of its components, including the CPU, RAM and storage are integrated onto a single circuit board. Despite it is small size and low price, the raspberry Pi is a fully functional computer that can run of operating system linux based, in this case will used Raspbian OS as the operating system.

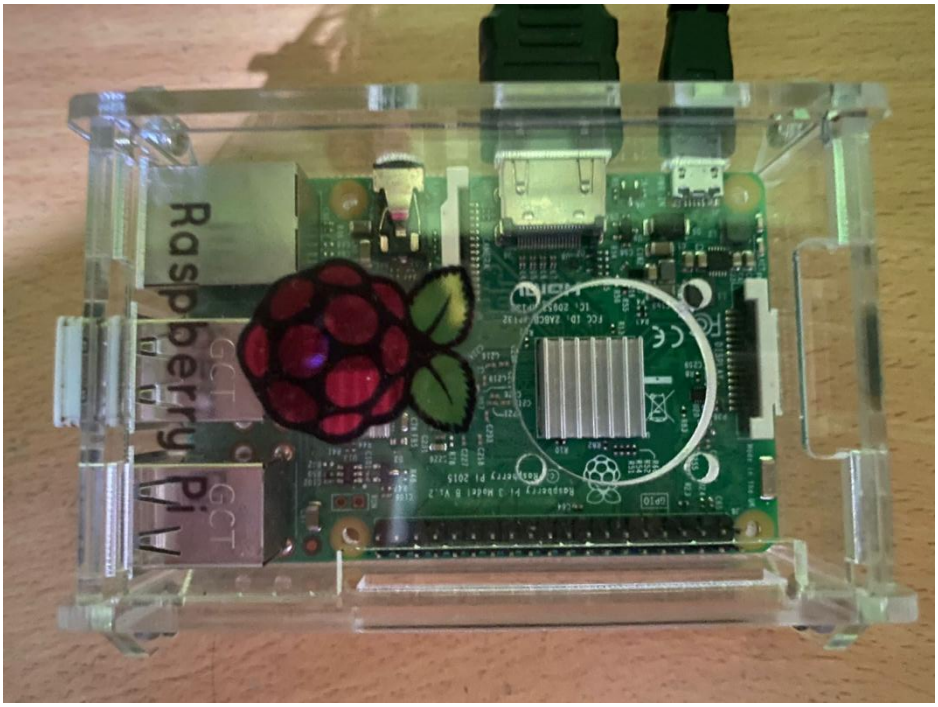


Figure 5.9 Raspberry Pi 3

In edge node will deployed a AMQP consumer that can consume the task from RabbitMQ queue. To consume the task the task, the edge node firstly should connect to RabbitMQ server same with the producer (Cloud server) connected. To connect the edge node with the RabbitMQ server, create a python code like the figure below.

```
# Create a connection to the RabbitMQ server
connection = BlockingConnection(
    parameters=pika.ConnectionParameters(
        host="192.168.8.49",
        port=5672,
        credentials=pika.PlainCredentials("guest", "guest")
    )
)

# Create a channel
channel = connection.channel()

# Declare a queue
channel.queue_declare("myqueue", durable=False)
```

Figure 5.10 Build connection between edge node and RabbitMQ

On the Figure 5.10 is the code to build connection between consumer at edge node and RabbitMQ server. The host of rabbitmq is 192.169.8.49 and the port listeners is 5672. After the connection is established, a channel object is created using the *channel()* method of the connection object. This channel object can be used to communicate with the RabbitMQ server and perform various operations such as publishing and consuming messages. The figure below shows logs activity when edge node received task from RabbitMQ server.

```
(.myenv) [11:01:22] [~/Documents/THESIS 2022/code] ❱❱❱ python consumer.py
Received printing order from Robert Carey DDS at 2022-12-07 13:51:58.540520
Received printing order from Jorge Thomas at 2022-12-07 13:52:00.658791
Received printing order from William Lopez at 2022-12-07 13:52:02.732942
Received printing order from Theresa Gross at 2022-12-07 13:52:04.867403
Received printing order from Jay Wilkins at 2022-12-07 13:52:06.970260
Received printing order from David Cisneros at 2022-12-07 13:52:09.060841
Received printing order from Brittany Lee at 2022-12-07 13:52:11.149227
Received printing order from John Martinez at 2022-12-07 13:52:13.231218
Received printing order from Anthony Sanders at 2022-12-07 13:52:15.318565
Received printing order from Jennifer Miller at 2022-12-07 13:52:17.396343
Received printing order from Dale Williams at 2022-12-07 13:55:06.828007
Received printing order from Jordan Rodriguez at 2022-12-07 13:55:09.014500
Received printing order from Andrew Morgan at 2022-12-07 13:55:11.141261
Received printing order from Karen Day at 2022-12-07 13:55:13.213601
Received printing order from Mr. Jeffrey Jenkins at 2022-12-07 13:55:15.290720
```

Figure 5.11 Example process when edge node received an order task

Furthermore, the edge node will receive the task from RabbitMQ server and load the task from binary format into json format. After that the task will saved into database in edge node, same with in Cloud Server, in edge node database also used MongoDB. The task including gcode string will save in database, while order id will send to the edge node queue with arrival time. The order Id will used to schedule based on FIFO algorithm. The early come task will early process to be printed.

```python
# Create a consumer
def consumer(channel, method, properties, body):
    # Process the message
    message = body.decode()
    task = json.loads(message)
    print(f"Received printing order from {task['name']} at {task['datetime']}")

    # Perform the necessary tasks using the data from the message
    del task['_id']
    save_data = SaveData()
    save_data.save(task)

    # send task to queue
    task_id = task['order_id']
    arrived_time = time.time()
    scheduler.queue([task_id, arrived_time])
```

Figure 5.12 Receive Order, Save to DB, and Send Order Id to Queue

The reason why did not send a whole task in queue because it more efficient to save the task in the database and only send the order ID to the queue. This can help reduce the amount of data that needs to be sent to the queue, which can help improve the performance of your system. Another potential reason to save data in a database and only send the order ID to the queue is to provide flexibility and scalability. By saving the task data in the database, it can easily retrieve it later if necessary, and can also update it if needed. This can be especially useful if the task data is complex or if it is likely to change over time.

```python
class Scheduler:
    # Initialize the FIFO instance
    def __init__(self):
        self.queue = []

    # Add a task to the queue
    def add_task(self, task):
        self.queue.append(task)
        self.sort_task()

    def sort_task(self):
        self.queue.sort(key=lambda x: x[2])

    # Get the next task from the queue
    def get_next_task(self):
        if len(self.queue) == 0:
            return None
        else:
            return self.queue.pop(0)
```

Figure 5.13 Scheduler Task on Edge Node

After sorting the data in the queue, the task will be sent to OctoPrint, an open-source software platform for controlling and monitoring 3D printers. OctoPrint is hosted on the edge node, which allows for efficient processing of the task data. The task data is automatically sent to the 3D printer using OctoPrint's RestAPI, enabling

seamless communication between the edge node and the printer. This ensures that the task is processed quickly and efficiently. By using OctoPrint, you can take advantage of its advanced features and capabilities to manage the task smoothly and effectively.

```python
# Send a task to OctoPrint
def send(self, task):

    # call query method
    task = self.query(task)

    # Create a temporary file to upload gcode file into octoprint
    temp_file = tempfile.NamedTemporaryFile(mode='w+t', prefix=task['name'], suffix='.gcode')
    gcode_string = task['gcode']
    temp_file.write(gcode_string)
    gcode_path = temp_file.name

    # Construct the URL, API key, and headers for the request
    url, headers = self._construct_request()

    # Sends one or multiple comma separated G-codes to the print
    response = requests.post(url, headers=headers, json=gcode_path)
    # Check if the upload failed
    if response.status_code != 201:
        print('Failed to upload G-code file:', response.text)
    else:
        print('Gcode file uploaded successfully')
```

Figure 5.14 Code upload task to Octoprint

The code appears to be query gcode string as a task from a database and writing it to a temporary file using the customer's name as the file name and the ".gcode" format. The path to the file is then sent as a JSON object and the file itself is sent using the "octet-stream" format, which is a binary format commonly used for sending files over the internet. If the upload success the request status code will show 201, its mean that the request to create the new resource (in this case, the temporary file containing the gcode string) was successful and the resource was successfully created on the server. The status code 201 indicates that the server understood the request and was able to create the resource as requested.

```
▼{files: [{date: 1670487835, display: "William Lopez.gcode",…}], free: 83940057088, total: 137491111936}
  ▼files: [{date: 1670487835, display: "William Lopez.gcode",…}]
    ▼0: {date: 1670487835, display: "William Lopez.gcode",…}
        date: 1670487835
        display: "William Lopez.gcode"
      ▼gcodeAnalysis: {dimensions: {depth: 197.039, height: 27.5, width: 209.088}, estimatedPrintTime: 20857.12076236081,…}
        ▶dimensions: {depth: 197.039, height: 27.5, width: 209.088}
          estimatedPrintTime: 20857.12076236081
        ▼filament: {tool0: {length: 22359.923040009046, volume: 53.781917623577385}}
          ▶tool0: {length: 22359.923040009046, volume: 53.781917623577385}
        ▶printingArea: {maxX: 209.088, maxY: 194.039, maxZ: 27.5, minX: 0, minY: -3, minZ: 0}
        hash: "9ec70574db298f12605b5c90e1d7590001939c03"
        name: "William Lopez.gcode"
        origin: "local"
        path: "William Lopez.gcode"
      ▼refs: {download: "http://localhost:5000/downloads/files/local/William%20Lopez.gcode",…}
          download: "http://localhost:5000/downloads/files/local/William%20Lopez.gcode"
          resource: "http://localhost:5000/api/files/local/William%20Lopez.gcode"
        size: 28387495
        type: "machinecode"
      ▼typePath: ["machinecode", "gcode"]
          0: "machinecode"
          1: "gcode"
  free: 83940057088
  total: 137491111936
```


Figure 5.15 Example of Gcode file Successed Upload to Octoprint

Based on the limitations mentioned in the beginning of the chapter, this research only focuses on the implementation of task scheduling for edge computing. This means that the research does not discuss scheduling in the context of 3D printing. Instead, the focus is on scheduling tasks in an edge computing environment, where computational tasks are performed at the edge of a network, closer to the data source, rather than in a centralized location. The research may discuss how scheduling algorithms can be used to optimize the performance of edge computing systems, but it does not cover scheduling in the context of 3D printing. In the next section this research will explain and discuss about Dynamic Prioritize Based Algorithm (DPBA) for improvement scheduling algorithm in edge computing.

Figure 5.16 below shows the average delay time when send data from cloud server to edge node. Based on the chart that can see number of task did not affected to delay time. The average delay time when sending data from a cloud server to an edge node can be affected by many different factors. One of the most significant factors is the network speed, which determines how quickly data can be transferred from one location to another. However, other factors such as the size of the data being sent and the distance between the cloud server and edge node may also play a role in determining the delay time. Additionally, the type of network being used (e.g. wired or wireless) and the number of intermediate steps or "hops" required to complete the data transfer can also influence the delay time.

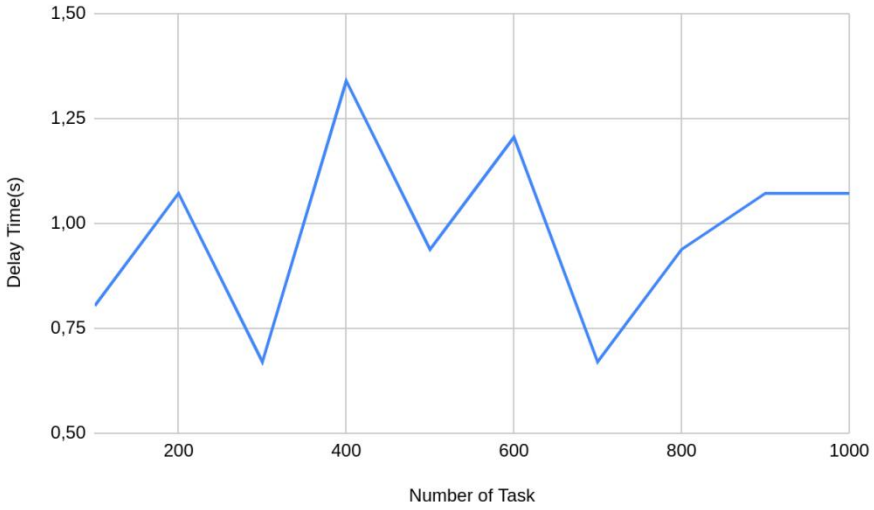


Figure 5.16 Delay Time Between Cloud Server to Edge Node

5.2 Simulation of Dynamic Scheduling

In the implementation of task scheduling in the section before, the task scheduler in the edge computing is using First In First Out (FIFO method. In this algorithm, tasks are executed in the order in which they are received, with the oldest task being executed first. This means that tasks are executed on a first-come, first-served basis, with the earliest tasks taking precedence over later tasks.

The FIFO method is simple to implement and can be effective in many situations, but it can also lead to inefficiencies if the order in which tasks are received does not match the optimal order for execution. For example, if tasks with different priorities are received at the same time, the FIFO method may not ensure that the most important tasks are executed first. In these cases, more complex scheduling algorithms may be necessary to ensure optimal task execution.

The proposed algorithm, called DPBA, is a dynamic algorithm that prioritizes tasks based on their length and required manufacturing capabilities. In this algorithm, shorter tasks and tasks with higher manufacturing capabilities are given higher priority, as they are more important and need to be completed quickly and efficiently. This helps to ensure that the 3D printer is utilized efficiently and that important tasks are completed in a timely manner.

The use of task length and required manufacturing capabilities as factors in the prioritization process allows the DPBA algorithm to take into account the specific requirements of each task and ensure that they are executed in the optimal order. This can help to improve the overall performance of the 3D printing system and ensure that tasks are completed efficiently and effectively. The DPBA algorithm is likely to be more complex than the FIFO method, but it has the potential to provide better results in cases where the order in which tasks are received does not match the optimal order for execution.

Due to the limitations of computer resources, the testing of the proposed DPBA algorithm was performed using a simulation built from scratch using python code. This allowed the researchers to test the algorithm without requiring access to a physical 3D printer or other hardware. The simulation was designed to mimic the behavior of a 3D printer and the task scheduling process, so that the researchers could evaluate the performance of the DPBA algorithm under different conditions.

5.2.1 Simulation Scenario and Configuration

The algorithm was explained in Algorithm 1 in the previous chapter. To simulate the algorithm the firstly should define the scenario follow the steps bellow:

1. A cloud user submits a task to a cloud server.
2. The cloud server sends the task data to an edge node for processing.
3. The edge node calculates the length of the task.
4. The task is decomposed into atomic tasks.
5. The atomic tasks are placed in a queue.
6. A 3D printer retrieves a task from the queue.
7. The 3D printer executes the task to create the specified object.

Before simulating the proposed DPBA algorithm, it is necessary to define the parameters that will be used in the simulation. In this particular simulation, the researchers have decided to use 1 cloud server, 1 edge node, and 1 3D printer as the hardware resources. With these resources, the researchers will test the task-processing capabilities of the system by simulating the submission of 1000 - 2750 tasks according to a specific task submission pattern.

The following table provides more information about the resource configuration and parameters for the simulation.

Table 5.2 Resource Configuration

Resource	Value
Cloud Server	1
Edge Node	1
3D Printer	1
RAM Edge Server (GB)	8
Ram Edge Node (GB)	1

Table 5.3 Task Parameter Setting

Resource	Value
Average Task length	500 - 3000
Data Size (MB)	10 - 20
Priority	1: High, 2: Medium 3: Low

The simulation will be run multiple times to evaluate the performance of the DPBA algorithm under different conditions. The researchers will analyze the results of the simulation to evaluate the performance of the algorithm and identify any areas for improvement. By carefully defining the parameters and resources for the simulation, the researchers can ensure that the simulation accurately reflects the behavior of a real-world system and provides reliable and meaningful results.

5.2.2 Simulation Results

This section will present the results of the simulation and explain them using a matrix to show the performance of the DPBA algorithm under different conditions. The matrix will include information about the task submission pattern, the number of tasks, the latency and average processing time of each task, and the overall completion rate of the tasks.

The visualization matrix in Figure 23 shows that when the number of tasks increases, the time and cost of various task scheduling strategies also increases. However, the DPBA algorithm remains relatively consistent, with only a slight increase in processing time and cost as the number of tasks increases up to 1500. After that, the processing time of the DPBA algorithm stabilizes, indicating that it is able to effectively handle a large number of tasks without incurring excessive costs.

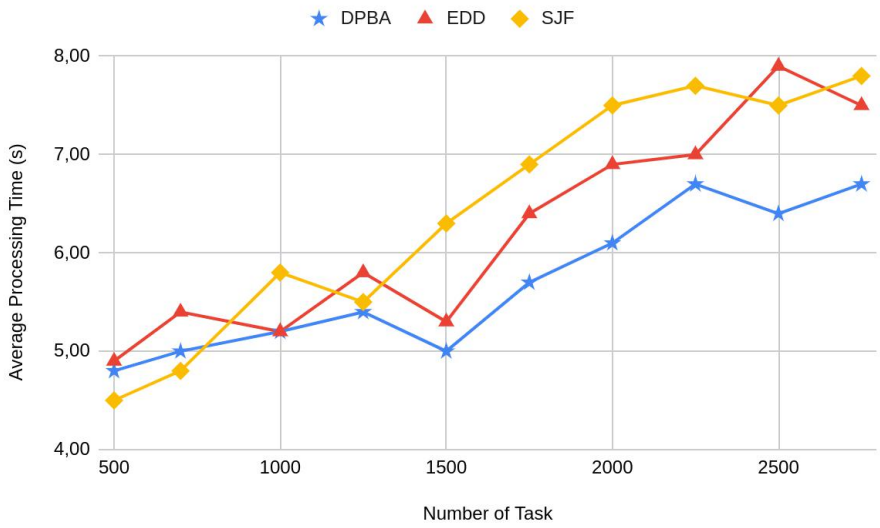


Figure 5.17 Average Processing Time

SJF algorithm may not be able to effectively utilize the 3D printer and may not provide optimal performance in all scenarios. For example, a task that requires high-precision manufacturing may take longer to process than a shorter task that can be printed using a lower-precision setting. In this case, the SJF algorithm may not be able to take into account the specific requirements of the tasks and may not provide optimal performance.

The visualization matrix in Figure 5.17 shows the relationship between the number of tasks and the average number of waiting tasks in the queue for the DPBA algorithm. In the early stages, the performance of DPBA is mediocre, with a moderate number of waiting tasks in the queue. However, as the number of tasks increases, the performance of DPBA becomes more stable, with a relatively low number of waiting tasks in the queue.

This indicates that the DPBA algorithm is able to effectively handle a large number of tasks without creating a backlog in the queue. This is likely due to the task decomposition feature of DPBA, which allows it to distribute the workload among multiple 3D printers and reduce the overall processing time of tasks. This helps to ensure that tasks are completed quickly and efficiently, without causing excessive delays in the queue.

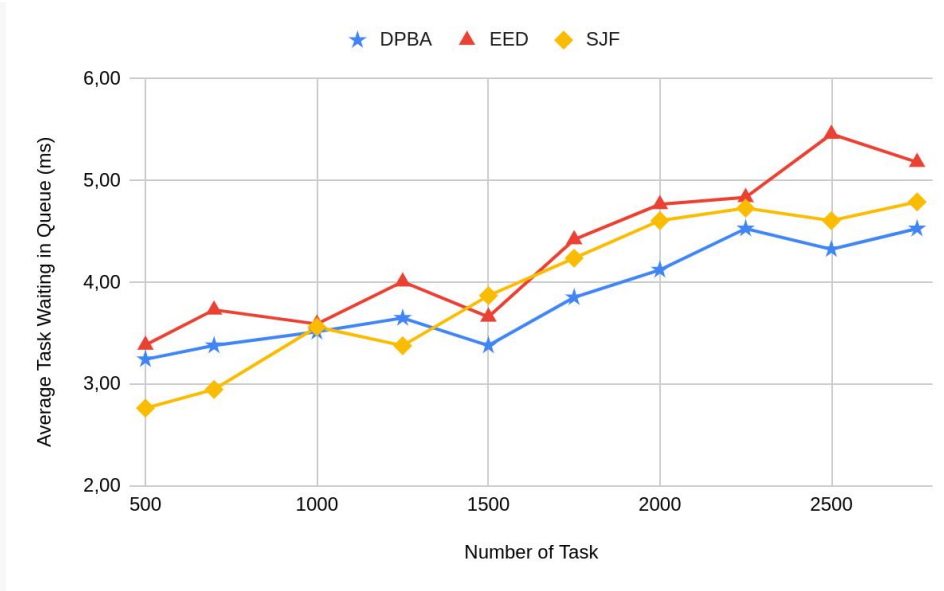


Figure 5.18 Average Waiting Task

Based on the simulation above can see DPBA can effectively to optimize the scheduling process in edge node. DPBA proved the task decomposition that make scheduling more efficient. It also prioritizes tasks based on their length and required manufacturing capabilities, with shorter tasks and tasks with higher manufacturing capabilities being given higher priority. This helps to ensure that important tasks are completed quickly and efficiently, and that the 3D printer is utilized efficiently.

CHAPTER VI

CONSLUSION

6.1 Conclusion

Based on the research and simulation in the previous chapter, there are several points that can be conducted as follows:

1. Cloud manufacturing (CMfg) is a new paradigm in manufacturing that uses cloud computing and Internet of Things (IoT) technologies to improve the efficiency and flexibility of manufacturing processes. In CMfg, manufacturing processes are distributed across multiple locations, and data and information are shared and accessed in real-time over the cloud. This enables manufacturers to optimize their production processes, reduce waste and downtime, and improve the responsiveness and agility of their operations. The CMfg system is typically organized into three layers: the cloud layer, the edge layer, and the physical resource layer.

2. The research provides a system implementation of the three-layer architecture, with a focus on the task scheduling process in the edge layer. The protocol used for communication between the cloud and edge layers is the Advanced Message Queuing Protocol (AMQP), and the software used is RabbitMQ. The AMQP protocol is used as a bridge to send tasks from the cloud to the edge nodes. The research demonstrates that the proposed three-layer architecture and task scheduling process can improve the efficiency and flexibility of manufacturing processes in a CMfg system. In this research also proposed an algorithm to task scheduling in Manufacturing Cloud Edge Collaboration, the algorithm is Dynamic Prioritize Based Algorithm (DPBA), The algorithm prioritizes tasks based on their length and required manufacturing capabilities, and decomposes long tasks into smaller ones to prevent the manufacturing equipment from being

overburdened. Based on the simulation this algorithm efficiently to optimize the scheduling in edge computing rather than other algorithms used in this research.

6.2 Suggestion

There are still many limitations in this research, because of that this is the suggestion for future works:

1. Improving the performance and scalability of cloud edge manufacturing systems, such as build a platform with User Interface (UI) that enables users to easily and effectively interact with the system and manage their tasks and resources.

2. Addressing potential challenges in current cloud edge manufacturing approaches, such as security as a critical concern in cloud edge manufacturing, as the system involves the sharing and access of sensitive data and information across multiple locations and stakeholders.

3. Improving the new algorithm in edge computing to create efficiently task scheduling such as using Machine Learning (ML) to create optimization algorithms that can adapt to the changing and complex environment of edge computing.

Bibliography

Asghar, H., & Jung, E.-S. (2022). *A Survey on Scheduling Techniques in the Edge Cloud: Issues, Challenges and Future Directions*. 1–19. http://arxiv.org/abs/2202.07799

Caiazza, C., Giordano, S., Luconi, V., & Vecchio, A. (2022). Edge computing vs centralized cloud: Impact of communication latency on the energy consumption of LTE terminal nodes. *Computer Communications*, *194*(March), 213–225. https://doi.org/10.1016/j.comcom.2022.07.026

Ciobanu, R., C.Negru, F.Pop, C.Dobre, C.X.Mavromoustakis, and G. Mastorakis. 2019. "Drop Computing: Ad- hoc Dynamic Collaborative Computing." Future Generation Computer Systems 92: 889–899.

Cheng, Y., F. Tao, D. Zhao, and L. Zhang. 2017. "Modeling of Manufacturing Service Supply–Demand Matching Hypernetwork in Service-oriented Manufacturing Systems." Robotics and Computer-Integrated Manufacturing 45: 59–72.

Ivanov, D., Sokolov, B., & Dolgui, A. (2020). *Introduction to Scheduling in Industry 4 . 0* (Issue June, pp. 0–9). https://doi.org/10.1007/978-3-030-43177-8

Jaakkola, H. & Thalheim, B. (2011) "Architecture-driven modelling methodologies." In: Proceedings of the 2011 conference on Information Modelling and Knowledge Bases XXII. Anneli Heimb• rger et al. (eds). IOS Press.

Jian, C., Ping, J., & Zhang, M. (2021). A cloud edge-based two-level hybrid scheduling learning model in cloud manufacturing. *International Journal of Production Research*, *59*(16), 4836–4850. https://doi.org/10.1080/00207543.2020.1779371

Li, X., J.Wan, H.Dai, M. Imran, M.Xia, and A.Celesti. 2019."A Hybrid Computing Solution and Resource Scheduling Strategy for Edge Computing in Smart Manufacturing." IEEE Transactions on Industrial Informatics 15 (7): 4225–4234.

Meier, M., Seidelmann, J., Mezg• r, I. "ManuCloud: the next-generation manufacturing as a service environment." ERCIM News, vol. 83, no. 33-34, 2010.

Nain, G., Pattanaik, K. K., & Sharma, G. K. (2022). Towards edge computing in intelligent manufacturing: Past, present and future. *Journal of Manufacturing Systems*, *62* (December 2021), 588–611. https://doi.org/10.1016/j.jmsy.2022.01.010

Pinedo, Michael L. 2016. Scheduling: Theory, Algorithms, and Systems. New York: Springer

P.P. Ray, A survey of IoT cloud platforms, Future Comput. Inf. J. 1 (1) (2016) 35–46, http://dx.doi.org/10.1016/j.fcij.2017.02.001, URL https://www. sciencedirect.com/science/article/pii/S2314728816300149.

Raileanu, S., Borangiu, T., Morariu, O., & Iacob, I. (2018). Edge computing in industrial iot framework for cloud-based manufacturing control. *2018 22nd International Conference on System Theory, Control and Computing, ICSTCC 2018 - Proceedings*, 261–266. https://doi.org/10.1109/ICSTCC.2018.8540725

Ren, L., Zhang, L., Tao, F., Zhao, C., Chai, X., & Zhao, X. (2015). Cloud manufacturing: From concept to practice. *Enterprise Information Systems*, *9*(2), 186–209.

Sahni, Y., Cao, J., Yang, L., & Wang, S. (2022). Distributed resource scheduling in edge computing : Problems , solutions , and opportunities. *Computer Networks*, *219*(November), 109430. https://doi.org/10.1016/j.comnet.2022.109430

Schaefer, D., Thames, J.L., Wellman Jr, R.D., Wu, D., Yim, S., Rosen, D.W., Distributed collaborative design and manufacture in the cloud–motivation, infrastructure, and education. in ASEE 2012 Annual Conference and Exposition, (University of Bath, Bath, 2012)

Tao, F., Zhang, L., & Nee, A. Y. C. (2011). A review of the application of grid technology in manufacturing. *International Journal of Production Research*, *49*(13), 4119–4155.

Tao, F., L. Zhang, Y. Liu, Y. Cheng, L. Wang, and X. Xu. 2015. "Manufacturing Service Management in Cloud Manufacturing: Overview and Future Research Directions." Journal of Manufacturing Science and Engineering 137 (4): 040912.

Wang, M., Zhou, J., Jing, S., Cloud manufacturing: Needs, concept and architecture. In Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2012 (2012).

Wang, L., & Wang, X. V. (2018). Cloud-based cyber-physical systems in manufacturing. In *Springer*.

Wang, X., & Wan, J. (2021). Cloud-edge collaboration-based knowledge sharing mechanism for manufacturing resources. *Applied Sciences (Switzerland)*, *11*(7). https://doi.org/10.3390/app11073188

Wang, H., Sun, M., Zhang, L., Dong, P., Wei, Y., & Mei, J. (2022). Scheduling optimization for upstream dataflows in edge computing. *Digital Communications and Networks*. https://doi.org/10.1016/j.dcan.2022.08.003

Wu, D., Rosen, D.W., Wang, L., Schaefer, D., Cloud-based design and manufacturing: a new paradigm in digital manufacturing and design innovation. CAD Comput. Aided Des. 59,1–14 (2015)

X.V. Wang, X.W. Xu, An interoperable solution for Cloud manufacturing. Robot. Comput. Integr. Manuf. 29(4), 232–247 (2013)

Zhang, L., Luo, Y., Tao, F., Li, B. H., Ren, L., Zhang, X., Guo, H., Cheng, Y., Hu, A., & Liu, Y. (2014). Cloud manufacturing: a new manufacturing paradigm. *Enterprise Information Systems*,*8*(2), 167–187. https://doi.org/10.1080/17517575.2012.683812