KADİR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES
PROGRAM OF ELECTRONICS ENGINEERING

CACHING ALGORITHM IMPLEMENTATION FOR EDGE COMPUTING IN IoT NETWORK

Mohammed Abduljabbar

MASTER'S THESIS

İSTANBUL, JUNE, 2020

CACHING ALGORITHM IMPLEMENTATION FOR EDGE COMPUTING IN IoT NETWORK

Mohammed Abduljabbar

MASTER'S THESIS

Submitted to the School of Graduate Studies of Kadir Has University in partial fulfillment of the requirements for the degree of Master's in the Program of Electronics Engineering

İSTANBUL, JUNE, 2020

DECLARATION OF RESEARCH ETHICS /
METHODS OF DISSEMINATION

I, Mohammed Abduljabbar, hereby declare that;

- This Master's Thesis is my own original work and that due references have been appropriately provided on all supporting literature and resources;
- This Master's Thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- I have followed "Kadir Has University Academic Ethics Principles" prepared in accordance with the "The Council of Higher Education's Ethical Conduct Principles"

In addition, I understand that any false claim in respect of this work will result in disciplinary action in accordance with University regulations.

Furthermore, both printed and electronic copies of my work will be kept in Kadir Has Information Center under the following condition as indicated below:

- The full content of my thesis will be accessible only within the campus of Kadir Has University.

Mohammed Abduljabbar

_____

30/6/2020

KADİR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES

ACCEPTANCE AND APPROVAL

This work entitled CACHING ALGORITHM IMPLEMENTATION FOR EDGE COMPUTING IN IoT NETWORK prepared by Mohammed Abduljabbar has been judged to be successful at the defense exam held on 30.06.2020 and accepted by our jury as MASTER'S THESIS.

APPROVED BY:

Assoc. Prof. Dr. Atilla Özmen (Advisor)	Kadir Has University	_____
Assist. Prof. Dr. Arif Selçuk Öğrenci (Co-Advisor)	Kadir Has University	_____
Assoc. Prof. Dr. Habib Şenol	Kadir Has University	_____
Assist. Prof. Dr. Figen Özen	Haliç University	_____

I certify that the above signatures belong to the faculty members named above.

_____
Prof. Dr. Sinem Akgül Açıkmeşe
Dean of School of Graduate Studies
DATE OF APPROVAL: 30/06/2020

TABLE OF CONTENTS

CACHING ALGORITHM IMPLEMENTATION FOR EDGE COMPUTING IN IoT NETWORK

ABSTRACT

The developing IoT concept brings new challenges to the service providers. The architecture of the networks changes to satisfy the needs arising by the large number of connected devices. Edge computing is the new architectural solution that will be used in the IoT networks. This architecture is more dynamic than the cloud computing network where the data can be quickly processed in the different layers of the network without going to the cloud. This will remove the problems faced by cloud computing: increase in data traffic and increase in latency of provided services. Research on edge computing in IoT networks encompass information-centric networks, use of 5G, and improving the hardware devices however a suitable solution for all the IoT use cases is not available yet. In this thesis, use of caching among IoT nodes is proposed as a solution to increase the efficiency of edge computing. Caching is an old but effective solution for dealing with data because it improves the real-time response of the system and can be used in IoT use cases. It will also not cause an extra hardware cost. In this research, two commonly used caching algorithms, LRU (Least Recently Used) and FIFO (First in First Out), are investigated and compared for their performance in sample IoT scenarios. Reductions in data processing time are observed where CPU and RAM utilizations are enhanced**.**

Keywords: IoT, caching, utilization performance

ÖZET

Gelişen IoT kavramı bu alandaki hizmet sağlayıcılarına başetmeleri gereken yeni sorunlar ortaya çıkarmaktadır. Ağ mimarileri, bağlı bulunan yoğun cihazların değişen ihtiyaçlarını karşılamak için değişmektedir ve çözüm olarak da "kenarda hesaplama" IoT ağlarında ortaya çıkan yeni mimari yaklaşımdır. Bu mimari bulutta hesaplamaya göre daha dinamiktir çünkü ağın her bir katmanında veri işlemeye olanak sağlamaktadır. Bu sayede bulutta hesaplamanın yarattığı iki soruna çare olmaktadır: veri trafiğinde artış ve sağlanan hizmetlerdeki gecikme. IoT ağlarında kenarda hesaplama konusunda yapılan araştırmalar enformasyon merkezli ağları, 5G kullanımını ve donanım cihazlarında iyileştirmeler gibi konuları da kapsamaktadır. Ancak hala tüm IoT kullanım alanları için uygun çözümler ortaya çıkmamıştır. Bu tezde, kenarda hesaplamada verimi artırmak için IoT düğümlerinde önbellekleme kullanımı önerilmektedir. Önbellekleme eski ama etkin bir veri işleme yöntemidir, sistemlerin gerçek zamanlı cevap süresini iyileştirmektetir ve IoT kullanım alanlarında uygulanabilir bir yöntemdir. Ayrı bir donanım maliyeti getirmemesi de bir avantajdır. Bu araştırmada, sık kullanılan iki önbellekleme algoritması (LRU ve FIFO) incelenmiş ve örnek IoT senaryolarında başarımları kıyaslanmıştır. İşlemci ve hafıza kullanımı iyileşirken, işlem sürelerinin azaldığı gözlenmiştir.

Anahtar Sözcükler: IoT, önbellekleme, kullanım başarımı

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Selcuk Ogrenci of the graduate school of science Kadir Has. The door to Prof. Selcuk Ogrenci, the office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my work but steered me in the right direction whenever he thought I needed it. special thanks to Dr. Atila for his support and guide and for the positive energy that he was given to me all the time

Deep gratitude to IALD that give me the chance to study in Turkey by gives me a scholarship for the master and for my colleagues for providing me with unfailing support during my years of study and through the process of researching and writing this thesis.

Finally, I must express my very profound gratitude to my parents and my brother and sister for the support and continuous encouragement throughout my scientific and life career also a special thanks for my big life coach and guide Mr. Abdulrahman Al-Ahmed who help me to be the best version of myself I will remain grateful to him all my life. This accomplishment would not have been possible without them.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 History

In this era, the concept of the Internet is changing, the Internet starts on the desktop and laptop that push the technology in the next level as we know it the sensor network and the RFID, Bluetooth and the related wireless technique which the data transmitting is hidden in the surrounding environment this continuous development make us push the technology into a new challenge, that produces new data that need processing and storage and Manage to get a beneficial result from it in real-time and that pushes the technology a step further to a new challenge which is how to deal with this produced data and information. The CLOUD computing was the solution for this case by providing a virtual structure for monitoring and processing the data also store the data in a cloud servers work base on an end to end system, this evolution in communication transceiver like (Wi-Fi, Bluetooth. . . .etc.) the different devices start to connect directly or with unique design bringing the IoT era. The main demands of any IoT System understand the user needs and the types of their gadget that he uses and his behavior; therefore each IoT operating system based on an architecture that has its own internal network and an analytical tool work on making the system more automated with no need for any human inter fearing is the backbone of the IOTs System. Kevin Ashton in 1999 was the first person how to quote the phrase of the Internet of things; however, the meaning of this term has been changed from that time until now, for example, the items terms now may refer to a Health devices or home devices..etc. But the central concept of the IoT remains the same, which is the data processing and transmitted computationally without any human interference to do an action, and this data is collected by a sensor and the effort done by an actuator, to achieve the above concept (transmitting or sensing the surrounding environment and processing the information for a goal related to the people demands) IOT uses an existing protocol to share this information by using devisees that have a wireless technology such as Wi-Fi, Bluetooth Xbee . . . . etc.[1]. IoT is a harmonic system connecting the smart devices that it have direct In touch with the Humans life, However the IoT system devises can exchange data without any human interfering this process happen by the network that connect these devises as a further advancement of this technology is to develop a system

in these devises enable these devises to communicate with each other and process the data and do an action without any human interferences called Machine-to-Machine system[2]. That will lead us to a conclusion that The IOT system will not connect the people with each other but it also connect virtual an physical things with each other based on operating systems and smart intelligence technology [3]. The IoT in the next decade will change the shape of our social life and will be connected to ever thing around us and it is an ample opportunity for business investment and competition to develop these technology because at the future the people who use this technology will choose the system that they like based on the quality of service that offered to them ,but this technology put the developer in an open challenges like the privies policy, the quality of service, the efficiency and capacity of the system, the data analytic, the Energy consumption, the security of the information, data Traffic produced by the new connected devices, system architecture also let us do not forget that Frequency bands width standers and Protocols and Some challenges related to the Wireless System Network (WSN) is a spirited field pertaining to the IoT challenges these challenges are the cost of this new technology to reach to a specific goal at the end which is make the IoT System a part from our daily life like the social media now [2].

1.1.1 IoT Architecture

The architecture of any IoT system is different from system to another according to the method used in (health, factory's, homes... .etc.) but it will still have the main three or four layers with some differences in the internal devices like a sensor or actuator types or operating system or even the driving application, but we can refer to the layers as [3][4][5].

The cloud layer: it is the layer that it represents a platform for all data and information ever saved and processed in it which it's the big server for the system. [5][6][7].

The application layer: this layer is different from the IoT system to another it depends on the system some so for some system it may use for mortaring and alarm notification

- so the user will not inter fear with any of the action in the system there like the M2M (machine to machine) and some PHD (Personal HealthCare Devises) but some of them the user can then use act with them to control the system according to his needs like the Smart Home System or Smart Vehicles System.[5][6][7].

- **The gateway and the network layer:** this layer where represent the Carrier medium for the data and signal from the physical layer to the cloud this. layer is consisting from set of nods that carry the data through a transferring Media which could be any type of signal by any type of transmitting protocol like (Wi-Fi, Bluetooth, GSM. . . etc.) [8].

- **Physical layer:** it is the layer that contends the sensor and actuator that that use to make the action even by collecting the data or handling with the case [9][10][11].

1.2 Subject and Scope

The spreading of the IoT system leads to connect new and different types of devices producing massive data that need to be optimized in real-time to keep the QoS quality of service of the application in the demand target. Therefore, the system must be optimized at the Node edges. [12].

1.2.2 Definitions

- **IoT**

  Era a broad group of computing and connected devices with each other directly or by its own by any networking methods to facilitate human life to achieve an action related to series of data processing and analyzing based on the collected data by an operating system from the surrounding physical area to achieve a specific gold. [13][14][15].

- **IoT applications**

  The IoT future will make a massive change in our social life every place will be work on IoT devices based on a Use case of the place like the smart market the smart hospital . . . Etc. That will produce data traffic and increases data processing [7].

Edge computing

Is an advanced cloud computing architecture methods base on putting external nods near to the physical layer of the IoT system .to provide real-time data process for the

end-user data with law latency as could as possible to achieve an improved QoS quality of service for the User [11] [12].

- **Offloading**

Processing and analyzing the data is very important for at any system but it will take a massive energy and capacity to give an accurate result there for you have to transmit the data to the cloud to get the accurate result but it will take a large number of resources for this process also you have to make sure about the quality of signal and the however this prosses may have some risk for transmitting the data to the main cloud from the quality of service point of view there for offloading the data to an edge servers to analytic proses near to the customer will provide a low latency and a real-time prosses for the data Offloading data from core to edgy is important to improve the response time of data proses and neglecting the unneeded data from being transmitted[11].

- **End to end system**

The end to end system is an estimation architecture model used to reduce the energy conception in the module The mine concept is to divide the IoT element into a three-part the data collecting devices part and the network part and the cloud part this methodology will compress the elements to reduce the energy consumption because energy consumption is directly proportional with the number of the devices connected on the IoT system [11].

- **Analytic Process**

Data lose its value when it did not analyses quick enough [11]. Some collected data could have an outlier data that need to be neglected in real-time or processed, Therefore the need to a massive analytic computational algorithm to achieve an improve data process time is a prior task in the [3].

- **Caching**

It is a temporary storage process concept that handles the data traffic to improve the processing time with low latency in all computational systems. The development of the IoT concept generates a new networking concept, which is edge computing to deal with the massive data in the local area in real-time and To manage the data exchange between the edge and the cloud so the use full (most

accessed) data will be kept at the edge. The unused-full data will go to the cloud to process and store [14]; we must refer to an important point here which is There are two types of data some of them need to a large and complicated process with no need for real-time. The prosses happens in the main cloud, such as stuff related to video streaming and serve lance cameras...Etc. and some need small or uncomplex prose but with a real-time result such as the health system or M2M or smart home. [15].

1.3 Motivation

Everything now can be sent through the wireless or sensor devices and can connect to internet [12]. Casio and Ericsson declared that the devices that it will be connected to the internet is going to be 50 billion devices in 2020[11], as we can see the iot now has become and will become a wide spread and there is a lot of devices is connected every day and this action will produce a new mount of data need to be handle with , the developer will be under new challenge which is , How to process this data and give the result in real time,[11]it is a big challenge especially when you dealing with data from deferent styles when every different data represent different event[4], the Edge computing was one of the solution because of the spreading of the IoT and the need of real time possess and feedback make the Inventors and developer make the local node could analyses and process the collected data from the physical layers [13],in these time we have to move from the classical big cloud servers towers the spreading the data between or a new architecture base on computing nodes near from the local areas or the physical layer or sensors there for we need to edge computing, still the capacity of the big data is greater than the edge for this reason the big data architecture are suitable for massive heavy computational system while edge is used for application need for real time processing without latency[12].

1.4 Problem Statement

Data quantity rapidly increasing because of the evolution of IoT system and the effect of the huge content exchange in social media which cause huge data traffic these data need to be processed in real-time with low latency this put the service provider to the challenge or improve the quality of service and make the processing executed in real-time as possible as it could, therefore, the cloud computing is not

useful anymore to deal with the new data stream there for the edge computing was the solution to deal with this new generation of data however the new edge computing architecture does not have the massive capability such as the storage capacity and the high processing ability that the cloud computing has and that brings the challenge of improving time process in the table again therefore caching was and still one of the solution of time process problems that face the data in many application, therefore, we will discuss the effect of cache Algorithm on-time process.

1.5 Objective and targets

IoTs Generally, will be affected directly on our life there for every IoT system is developed by new architectures to improve the QoS Quality of Service one of the architecture Solution is Edge computing essential Edge computing architecture found to improve QoS and reduce data traffic caused by the connected devices but still this architecture need for more advancement because of the widespread of the data transaction that Couse by data track of the new connected IoT devices and the competition between the company for best service with real-time For this reason [12], therefore we will implement caching methods based on caching strategy with improved algorithms help to improve the Quality Of Service and time processing at the edge so the outcome will be:

1. study the effect of caching algorithm on the time process.
2. reduce the time process for the data and make it executed in real-time as passable as could.
3. cache program can be excited at the edge to reduce the time process.
4. research that describing the effect of caching on improving the edge computing performance and the contribution of cache in the different computational application.

edge computing system with improved process time could matches the real time with low latency.

2. METHODOLOGY

2.1 Cache Overview

Caching is a permanent storage concept that can be achieved through a hardware or software component caching concept is how to provide a temporary storage for the data that user mostly use it in the future or frequently without need for any new processing action inside or from the main data storage area, Reducing the delay in processing time and achieving low latency is the main properties that the cache mechanism achieve in the grid, it becomes the commonly used methodology in the most technical application, however, the cache concept will not be effective unless the used application has a data traffic [9][16], The principle of cache is started as a computer concept use to improve computer processing [22]. The performance of the computer process become faster and much easier and adequate with the cost of the computer that time, the caching mechanism built on the Replacement Policy Algorithm (page replacement algorithm) applying this algorithm concept in the computer processor achieve this technical jump in time processing quality in computer performance in the past decade. The performance that the replacement policy algorithm did through improving the processing ability and performance make the researcher focus on this kind of algorithm and leave other algorithmic procedure that could improve computer processing, Replacement algorithm is not a random why of programing but it should be based on a writing policy and cache algorithm. the cache methodology this technology is old but still a serious topic with every new technology or application deals with data, Therefore Scholar still use this kind of algorithm to achieve the better data processing performance in that application, To achieve ether latency reduction or time proceeding improving and a real-time response Random replacement algorithm replaces the cache line at randomly by forming a random number in each cache access and set the memory access block on that lines' number. The major disadvantage of this methodology the priority does not take into consideration [16][20] [22][27][28] [30].

2.1.1 Cache methods

Each cache strategy has a different criteria and different task according to the use case that deals with and it is also different from application to application, therefor the cache strategy priorities in data storage are different from communication applications to the

computational application to the servers [22][24][28][31]. The flexibility of using the replacement algorithm is wide because we can combine several cache strategies to solve the case [9]. The caching strategy cannot be affected unless there is data traffic or data density in the system otherwise the cache is useless. Therefor the cache replacement algorithm is classified into the following types:

First in Frist Out (FIFO)

The first in first out algorithm work according to the queuing mechanism which stated that the data set will form a single queue in a specific size of data processed as shown in the fig 2.1 the data will shift registry move in the cache and whenever and when the cache size is full the cache will erase the first data (task) interred the queue, and this sequence will continue until the data is finished there for if the same task is repeated, we will call this a cache (hit) and when and when the same task arrived we call this a cache (fault).[17]

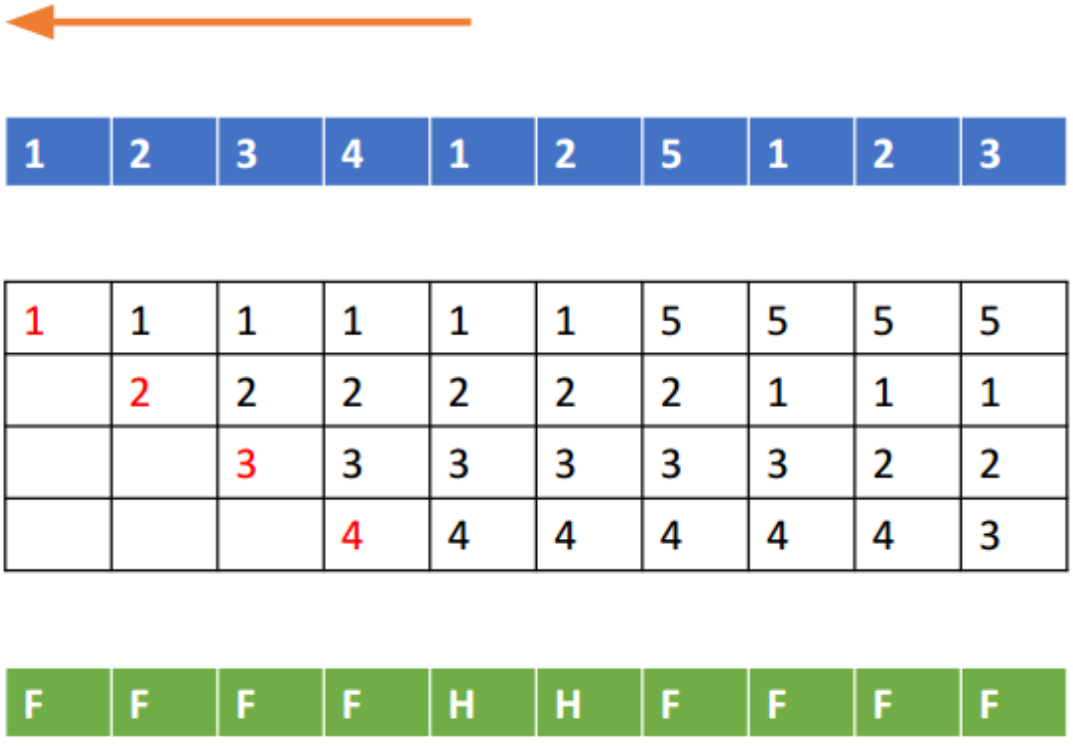

Figure 2.1: FIFO data processing

Last in First Out (LIFO)

LIFO is the queue strategy that is work in the opposite way of the FIFO the evacuation will be for the latest data that has been cached in the memory [18].

Least Recently Used (LRU)

The Least Recently Used Algorithm is similar to the FIFO but it is a bit complicated cause the FIFO is erased the first data inter to the queue if a new data arrived but the LRU will check the repeated data in each page and then erase the lest repeated data in the queue according to the old page, not the first finishing data therefore according to the figure2.2 the (M2, M) is the newest used data in the queue, therefore, it won't be neglected while (1) is the least recently used data in the queue their fore it will be erased the main difference between the LRU and FIFO is [19].

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2
7	7	7	2	2M2	2	2	4	4	4	0	0	0	1	1
	0	0	0	0M	0	0	0	0	3	3	3	3	3	3
		1	1	1	3	3	3	2	2	2	2	2	2	2
F	F	F	F	H	F	H	F	F	F	F	H	H	F	H

Figure 2.2: LRU data processing

LRU:

• Keep tracking the pages when the pages have a new Fault

• Difficult to implement

FIFO:

• We do not need to track the new page when the Fault is occurs

• Easy to implement.

Time-Aware - Lest Recently Used (TLRU)

This type of replacement algorithm is an advanced LRU method of cache strategy it is commonly used with the ICN information-centric network and the data evacuation take in the consideration the frequent time that this labeled data is common demanded and the possibility and the local places that need this data [21].

Most Recently Used (MRU)

The algorithm action in MRU is most the most popular or the most significant accessed data because it is possible that the users do not need this data. This strategy works mainly in the PC s [20].

2.2 Cache contribution

2.2.1 Cache contribution in computers Technology

2.2.1.1 CPU Cache

The CPU (Centric Process unit) is the primary unit of any computer, and the board contains the I/O ports and the MMU memory management unit clock ethernet caching memory In all of the computer architecture design now time the CPU the supported by a cache memory this memory store the most used data so the user could have the data from the cache memory not from the main memory in the CPU; therefore this hardware cache improve the CPU performance by reducing the data processing time [22]. Figure 2.3 shows the cache of the simple cache mechanism in the CPU. Moreover, the continuous advancement of the processor leads to improve data process performance of the CPU but in the same time leads to increase energy consumption, therefore implementing cashing algorithm at the processor of the computer to achieve low power consumption with high process speed the cashing not only reduce the power conception but advanced the data processing inside the CPU [23].

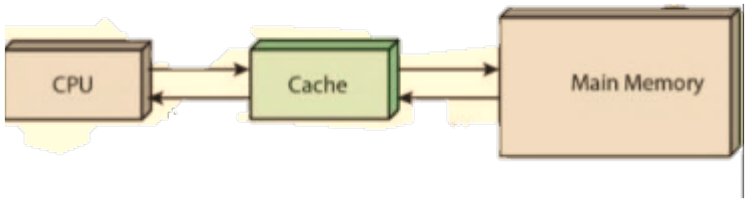

Figure 2.3: CPU Cache

2.2.1.2 GPU Cache

Graphic Process Unit is a computer hardware work side by side with it contains a very complicated mathematical matrixes algorithm that presenting the data to graphics on the main screen [24], The contribution of the GPU in the last several years has been increased in the different applications because of the high data processing performance it is used in forming the different digital currency (Bitcoin, Lit cons..etc.) and get involved in the other technological application because of its properties [25], One of the things that s makes the GPUs different from the CPU is the GPU is processing the data in parallel while the CPU process it in serial, it also contains a Management Memory unit and a caching memory [24], The block diagram in Figure 2.4 shows the architecture of the GPU.

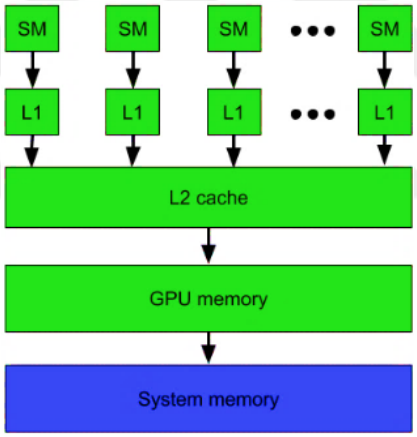

Figure 2.4: GPU simplified Architecture

2.2.1.3 CPU&GPU Cache

Because the two processors have the same architect the developer starts to develop a Hairdo core meet the need for the continuous data revolution especially when because of the heading toward the big data in the next several years, to prosses the tremendous amount of data now time it is become a trend to use the dual hybrid architecture which mean combining the two chip (Graphical processor unit and the centric process unit) to get the benefit of the two prosses in the same especially from the properties of the GPU in the industrial and applied technology because of the massive data possess that the GPU

gives. Therefore, and based on common memory cache computer companies start to develop the cache replacement algorithm to meet the criteria of the new CPU-GPU hybrid architecture [26][24]. Figure 2.5 shows the new GPU and CPU Heterogeneous Architecture.

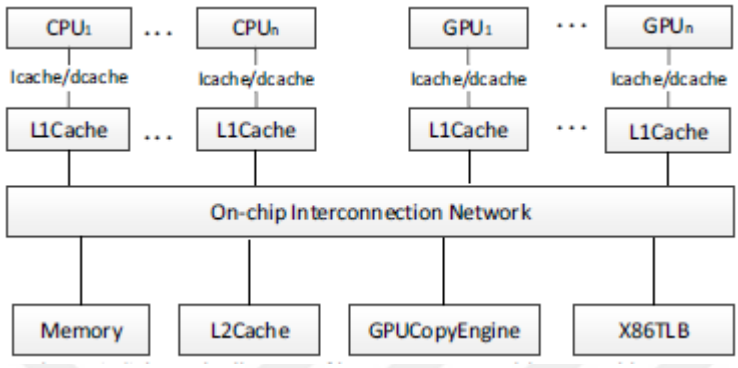

Figure 2.5 Heterogeneous Architecture

2.2.1.4. Disk Cache

Disk drive is the hardware derives in the computer that all the programs data are store inside it also there are two types of hard disk even it is a built-in or out disk, and old generation and generation the new generation of disk is the SSD which it uses transistor for the storage and the old generation is using the magnetic media to store data and transmitting and resaving bits and because of the massive data storage that we need for the new programs the disk should be able to meet the storage criteria for this programs and should have a sufficient storage space to satisfied the needs of the program [28][29] [30].And the high prosses speed proportional to CPU input-output functions of the PC. the internal caching it is implemented inside the hard disk by allocating a space for the cache inside the disk and there is a lot of research dealing with this method and this research focus on compressing and simplified the caching algorithm the other contribution of cache found to improve the magnetic delay on the hard disk that lead to use the caching, the mechanism by adding external RAM with the disk drive [29]. The time delay caused by the magnetic gap delay in the disk so scholars start to discover new methods to improve that delay[29], therefore moving to the new SSD drive is an advanced solution for this delay, moreover Caching is an open research topic in SSDs to find new heterogeneous architect for the disk drive to improve the performance[30].

2.2.2. Cache Contribution in Web

Data traffic now is huge challenge in the cloud computing the and this challenge is related to the widespread of the smartphones that bring new challenge to the developer and web service company to provide fast service with low latency and high time process therefor caching methods took place in this knowledge area to _x the problem of the data traffic for less using of the bandwidth example in [31][32][33]. We can classify caching into two layers caching at web browser layer and caching at the servers or proxy server.

2.2.2.1 Web Browser

This type of cache is based on saving the web page data temporary on the hard disk of the computer, so if the user needs this web page again the browser will not open the web page from the server it will open from the user computer hard disk [33], figure 2.6 shows a simple mechanism of cache algorithm in browser.

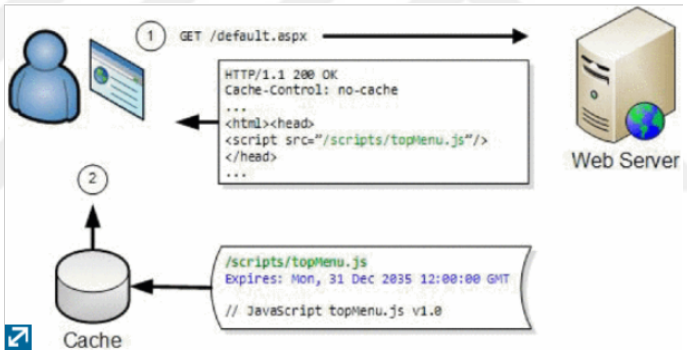

Fig 2.6: Cache Browser

2.2.2.2. Proxy Server

It is a sharable device in the internet network grid placed in the half distance between the client and the server and as any caching methodology. It is used to store the prior data of the web pages that user mostly access to proxy server is used to reduce the presser on the bandwidth[32] also reduce the data traffic in the internet network by storing data that mostly used that took a massive prosses operation like photos and videos with high resolution this kind of content that mostly used especially after the smart phone evolutions [31]. figure2.7 shows the proxy server concept.

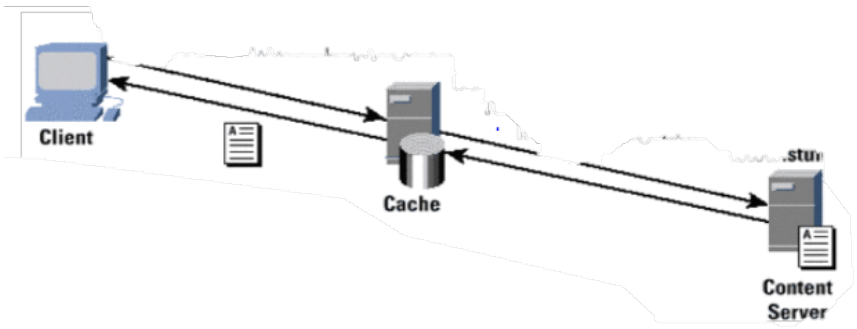

Figure 2.7: proxy server

2.2.2.3. Server Layer

The proxy cache is part of the internet network that used by some internet provider company but it is not a part of the client or the server there for some internet web site service provider use caching algorithm inside the web pages itself to make the web site much easy to access and this property more related to web pages programmer than the service provider[32].

2.2.3 Cache Contribution in wireless and Communication Network

The wireless network started with the internet evolution area, therefore we can define it as a computational network use to transmit and receive data between two different places it is using in large building and institute to reduce cabling usage instead of the wired networking and also use in the long-distance transceiver the wireless network. We will focus on several advanced wireless application dealing with cache [35][36][37][39].

2.2.3.1. D2D Cache

The D2D (Device to Device) is a communication concept based on the connecting of two mobile devices without any transceiver point and those devices can share content. The evolution of smart devices such as smartphone tablet.... etc., this concept is expanded to include connecting several devices in the same time, however, D2D also start to face different problems like the power consumption because the battery power of the smartphone devises is limited, also the data traffic this problem caused by the high data

owing by the devices [35], therefore the caching Replacement Algorithm took place to Minimize data traffic By Off-loading the data traffic that caused by High data rate transceiver content[36], The proposed solution was using the cache algorithm in the cluster to reduce the data traffic[35], and this action leads also to reduce battery consumption [36].

2.2.3.2. Ad Hoc

Mobile Ad Hoc Network MANET or Wireless Ad Hoc WANET is a network it is a name labeled on an exclusive type of network that it has no constant info structure simulated to any wireless network like router or constant node this kind of network is designed to do special tasks like the Wireless sensor network or Navy or industrial robots network or streets smart light network . . . etc. This network is very useful in transactivating the information in the surrounding and do not need for a high-cost info structure devices however this kind of network is facing great challenges because of the high data that the smart devises produce this data traffic increase the latency and reduce the process time The caching methodology was one of the solution to solve this issues [37] by follow caching strategy based on setting a cluster named as a (cluster cooperative) this cluster grouped in not overlapping clusters inside each claustral there is caching node this strategy achieve reducing in the latency comparing with the similar ad hoc network that does not use this strategy [38].

2.2.3.3 5G

The 5G technology is the next generation of the of cellular communication the scholars developing the 5G to be much prepared to overcome technology market challenges in the next several years the need for the 5G is prior to overcoming the large production of data that caused by smartphone application that needs for real-time response with low latency for the data process especially, that in the next several years (78 percent) of the data content will be video and image with high resolution [34]. However caching hot topic for reducing time processing by permanent saving for the data this concept is start as a computational concept to reduce the time process in the computers but still need it uses in other operating system technology for the same purpose There for the scholar still working on new caching methodology in the different level of 5G system (the user, the service provider, and the operating system level) [39].

2.3 ICN

Information-Centric network is a new concept based on developing the internet network architecture by improving the static protocols between the cloud and the end-user that most of the internet network use the data will be shareable based on information, not data the data will be labeled according to most accessed information from the end-user side[40] this structure Begin from the need to a new architecture focus on the shared information trend between the users, e.g., the expanding of social media such as Facebook YouTube . . . etc. and another platform that shares videos with high resolution from amazon and Netflix ..etc. Therefore, the network criteria have become based on what meets the requirements of the user and facilitates his experience in the information he shares, whether it is a video or high-resolution image and not an unknown data type. A Point out between data and information; the data is the lowest structure that any system is based on and after the data is processed and simplified and classified to be a piece of information [41]. Therefore, the characteristic of the ICN founded to Meet content needs, not the data and there are several types of ICN which is

- Content-Centric Networking (CCN).
- Publish-Subscribe Internet Routing Paradigm (PSIRP).
- Network of Information (NetInf) · Data-Oriented Network Architecture (DONA). [41]

2.3.1. Caching in ICN

Using cache mechanism makes us get the full efficient from Information-centric network most cache mechanism use in FIFO (First IN FIRST OUT) and LRU (Least Recently Used) [42] the cache is either be at the edge of the ICN network or in the network [40] or caching the specified information content at the node to [41].

2.3.2. IoT ICN

The Internet of Things is the new era of technology that everything around us will be connected to the internet like your home your cite and also the big machines and the small

cost electronic instrument for example (home sensors, refrigerators, microwave... etc.) these devices need to get the access in the system smoothly to give the best user experience to the user more over the IoT technology will lead us to the big data IoT cloud era where combining between the virtual content (Facebook, YouTube . . . etc) with the data of physical worlds .many of the connected devises have a limited design like the energy consumption memory size . . . etc. This brings new challenges to the scholars to provide a system that fits these devices, therefore the IP address versions or cloud computing are not appropriate for the IoT needs for the user data traffic to data security There for building IoT system based on ICN structure is appropriate more for the IoT system [43].

2.4 Edge Computing

Transmitting and resaving data in the next several years will grow more than ever because all the surrounding devises and sensor will be connected to the internet this huge data growing need to be handled in different whys there for developer and scholars are discussing a new method to handle this amount of data, The upcoming revolution of smart devices will generate data added to the system, for example, the face ID detection technique and the high-resolution video stream apps the user content production on social network (Facebook, YouTube. . . .)and their data production means these data will go through the networks by the IoT system which it will reach more than 1.6 Z-byte by 2020, that's will cause a pressure on the network, Therefore the available cloud computing and Consecutive versions of IP address Networking method are not suitable for the new data , because in the IP address even the new version application still facing the un-solved mobility problem challenge which is the moving devices will disconnect or the data will interrupt temporarily until the user reached to the next access point device and this situation does not fit for the new mobile devices that will connect to the internet such as robots health care accessories . . . etc. [44], and not appropriate for the IoT system, edge computing can be defined according to the continuous development in the IoT system in the next several years will produce a huge amount of data that needs to be handled. Edge computing is an advanced networking info structure consisting of placing a sub server nearby the user device areas; this server can store and process data and can achieve the seeking random data access for the mobile IoT devices without any temporary

interruption. The edge computing will be about 80B of the network industry BY 2021 [48]; the edge device is part of the network layer, not from the clouding layer [45].

2.4.1. Architecture

The edge computing is a networking model used to improve the data way of processing to reduce latency and get rid of the temporary disconnection that happens in the mobile device while moving from access point to another and this does not suit able for the IoT systems however according to the previse research [44][45][46][47]. the design of the IoT system can be executed according to our needs and the system purpose and classified the IoT system to a different layer inside the layer edge server is placed inside the layer. The classification will start from bottom to top; therefore, we can propose generally:

- **Layer 1**

  Hardwar is the basic info structure that contact with the physical world that any IoT system based on which it includes the sensors that collect data from surrounding environment like (humidity, temperature. . . etc.) the actuators which represent the data output hardware after the data is handled like (smart home lighting, speakers. . . .etc.) mobile devices like (phone, Smartwatches, cars. . . etc.).

Layer 2

The network layer is mid wear between the cloud and the hardware layer that transceiver data to the cloud to store or process and distribute the process data to the hardware layer to achieve action such as (Router, Node, Gateway, Microprocessor . . . .etc.), and as we mention before the IoT Architecture system that based on edge computing network placing an Edge Server at the network Layer, the task of that the server is to Prosses data to achieve real-time response with low latency and to neglect the cut-off or the disturbance that mobile devises facing.

- **Layer 3**

This layer is the clouding layer that all the data from all edges are transfer to be stored or to be processed moreover in the next several years we are heading to the big data clouding where the social network data like (Twitter, Facebook, YouTube . . . etc.) and the physical world data devices such as (vehicles, sensors, home devises .. etc.)

going to be mix the edge computing architect will be a good enhancement in data stream because it will reduce data traffic between network level and cloud level.

2.4.2. Different between Edge Computing and Cloud Properties

We can list a group of the main different properties between the edge computing and cloud computing:

1. Latency

The latency is the phenomenon that Couse a time difference or lag between the order and the response time during data processing action the latency in the communication networks depends on the system capacity and the broadcasting distance and the data rate, the broadcasting distance in edge computing is extend for several meters for the small transceiver similar to the device to device networking and maximum could reach to 1 Km between the edge server, and the user, the cloud computing broadcasting distance range from the user to the cloud server is extended form several Kilometers to distance could be a cross country because of the service provider server in another country. And this will cause a broadcasting delay for Cloud computing, which is the reversing of the Edge [46].

2. System capacity

Generally the cloud computing servers have a high computational capacity enable to process data in real time and that what cloud computing Precedes edge computing in, even though edge computing is dealing with a local data that does not require a high capacity in contrast with cloud and this will reduce the gap between the edge computing and cloud computing process speed More over the available server that will use in edge computing is suitable for that edge task it has a high processing speed to meet the obligated. Therefor this issue will not be a big challenge for edge computing,

Data Rate in the Cloud computing the data have to flow from the nods and radio access transvers and other networking part and this will Couse time delay related to the data traffic and other challenges before on the other hand in the edge computing the data will not go through this processing procedure because the servers will be at local places[46].

2. Energy Consumption

IoT devices are different size devices with low memory storage programmed to achieve specific tasks the spreading of IoT devices will bring us to a new open challenge which is providing sustainable power to operate this device, using a portable battery and change the battery is not a practical solution because we will be dealing with plenty of IoT Devices the Edge computing is providing solution for this challenge by offloading the intensive computational operation from the IoT devises to the Edge that will be a practical solution for this problem the researcher how were working on this area of research achieve a reduction in battery conception and increase the battery lifetime to 40-50 %[46].

3. Context

Awareness the edge computing server is placed in a local area nearby the Customer (IoT User) it gives an additional feature to the system by giving an easy and fast access to the user attitude, location . . . etc. That will give fast information source to the end-user behaviors and that will It is an opportunity to provide the user with his needs of services or products based on analyzing the trends or his location. For example, if you were in a specific place in the market the edge will analyze your location or attitude to provide suitable information about the stuff you want to buy and the best offers that you can get from the product[46].

4. Privacy and security improvement

In the cloud computing the user data will be collected in one place which is the service provider (Amazon, MICROSOFT. . . .etc.) server and its persistent target for the hackers because it contains the users (customers and companies) therefore it is an information treasure for them also the ownership and the management of data is separated (if I own my data I cannot manage it if I can manage my data, I am not the owner of my the data the service provider control my data) and this will Couse an issues of loss leakage of user information because the companies control my data not me, The edge computing provide an improve security properties for the edge services users, firstly the fact that the edge servers extend on a small scale region the probability of hacking a valuable information is difficult because the information not gathered in one place[46].

Secondly, generally, the ownership of the edge server will under the privet sector companies how to provide IoT service (health, money, transaction . . .. etc.) or the end-user himself and this will reduce the probability of hacking the privet information that

transmitted between the end-user and the service provider, more on that (IoT) companies will be able to control the access level to the information access without needing to external units[46], we can list of most different criteria with the following system structure as showed in table 2.1.

	Cloud Computing	**Edge Computing**
Server Size	Large, complicated servers	Small server with improved accessory devises
Server Location	The servers located in a remote data center each data center size equal football field size in several places around the world	The server located in the local area in the network layer nearby the gateways, router and the end-user themselves
Deployed	Cloud computing adopted by Internet Service provider companies such as Amazon and Microsoft required a complicated configuration and design.	The small companies or smart homes adopt edge computing, and it required a soft configuration and design
System management	Centric control	Hierarchical/ the network is even centric network or distributed network
Latency	More than 100 milliseconds	Less than several tens of milliseconds
Applications	The application criteria should have tolerance latency (social Media, learning … etc.)	The application criteria are deal with a critical latency smart vehicle, automated factory … etc.

Table 2.1: Different between cloud computing and edge computing [46]

2.4.3 5G and Edge

The talking about the next generation of mobile communication is become a hot topic in this several year the past generation the of communication methods like (3G, LTE, 4G...etc.) in the past several years was discussing the why of improve some of the communication criteria but the 5G will be a big jump for communication industries because of the massive data 5G will deal with therefore the [48] 5G new criteria will focus on improving the bandwidth by using the mm-Wave stream and improving this spectrum efficiency by using the MIMO methods [49].The spreading of smart devices like smartphones and tablets which it have a high processing energy equal to the computers processing ability will produce a high data density Cather on the edge and that's will bring us to a new challenge which is the grantee of the quality of service (QoS) [50] and the transmit media between the IoT devises and IoT system will be 5G, Moreover the caching strategy also took place in improving 5G data traffic to achieve good performance for the network [51].

2.4.4. Cache and edge

Edge network deals with devices and systems have Critical latency [46] the smart devices will be equals to billions in 2021 [38], therefore the continuous improvement in the edge computing service is essential in all layers. The rapid data follow need for a methods to reduce latency and provide real-time process as much as possible Each caching study has to focus on what to cache and where to cache therefore the cache must be implemented in two places even at the core edge (edge server) network or edge network because the ICN is important in the new data management system labeling the data based on the content information according to the end-user attitude and using LRU or FIFO is a promising cache strategy to improve data time process in the core edge and reduce latency by reducing the data traffic[52].

2.5. Cache contribution in IoT Applications

IoT Applications distinguished due to the Sustainable advancement of wireless networks the devices started to connect with each other via Wi-Fi, Bluetooth, GSM, therefor the IoT starts to rise up bringing a new life standard for the developing cities, In the next

several years the smartphones and laptop will reach to billions devises this will bring us to new era of communication network, the network will be heterogeneously connected and the virtual data for social media and the smart devise data will be grouped in one big cloud there for face a huge data challenges that will be related to data traffic and processing time . . . etc. The useful IoT Application should provide a perfect QoS (Quality of service) [53], Cisco announced that at 2021, 78 percent of internet traffic content will be videos [38], focusing on developing methodology to achieve an improved user experiment is prior task, To achieve a good QoS in IoT Applications scholars started to focus on the caching methodology and internet-centric network ICN mechanism to achieve effective transmission with real-time response they started to develop new technology based on caching strategy [54], in the different IoT applications [56][58][60], assuming Edge computing considered as business solution for IoT application services [54] Caching mechanism is a technology that trades time for space, studying how to minimize the required time to gain information and familiar network traffic and achieve efficient information transfer.to provide effective User Experience[54].

Generally, any IoT Architecture is made from several layers the sensor and actuator that collect data from the surrounding area and the network layer which contain a heterogeneously random access network and the cloud layer and the combination of this structure forming IoT application, IoT applications are the services that IoT could provide to the customer like smart homes, smart health system and smart vocals and smart grids . . . . etc., and each IoT application the possibility of implementing a cache strategy is available [54].

2.5.1 M2M

The Machine to machine technology is one of the IoT most advanced application the IoT system is designed and implemented to achieve a special task (industrial or civil) without any physical or human interfere the basic architecture as any IoT system in consist of the sensor layer and the cloud layer and the network layer which the data will flow according to (end-to-end) concept, the data will be collected from the end-user (physical world) and flow through the sensor network through the gateway to the end server and analyze and processes to achieve the application task of IoT/M2M. This system is widely used such as smart industry smart traffic . . . ..etc., where the device in communicate and process

data without any human interfere, however, this methodology is facing several challenges because each and every M2M system are unique system and have a different task and each solution will produce more data which it is huge by itself and the solutions subjected under the business and customer need [55]. therefore the cache is now taking place to produce a single improved cache strategy for the Differentiated and complicated M2M applications network to achieve high process time[56].

2.5.2 Smart cities

In the next several years the objects around us will be connecting to the Internet throw a Microcontroller have a digital transceivers device this transmitting and resaving method could be RFID, Wife, Xbee. . . etc., the smart cities implementation occur in the short term because in 2020 the investment in smart city will equal billions of dollars because it is a new open market and it going to be a part of the lifestyle, the design of the smart city will be an ecosystem have a smart hospitals and smart traffic and smart building and lighting . . . . Etc. or anything facilitates human's life [57]. this will produce a massive data need to be handled data caching algorithm is took place in advancing the Smart cities system network side by side with the ICN (information-centric network) and the MEC (Mobile Edge Computing)[58].

2.5.3 Smart Vehicular

The IoT in vehicles or the IoV (internet of vehicular) is a new IoT concept, and now the developed country start to adopt this concept in the transportation system like the European country and Japan this concept aims to achieve new automobile driving experiment and reduce the accidents and traffic and reduce the petrol consumption the object that will connect on the system will be even V2V vehicle to vehicle or V2I vehicle to internet or V2S vehicle to the sensor or V2R vehicle to road the ad hoc network is the used networking in the IoV network although the future of IoT system is facing a several challenges from the speed and range of the transceivers (RFID, WIFI . . . etc.) to the data traffic [59] there for the research took the cache algorithm as a strategy in improving the ad hoc network of the IoV, and it considers a promising solution to achieve the excellent IoV performance [60][61][62].

2.5.4 Smart health

The IoT is entering the health industry from the early moment of IoT, outset the enhancement started in the E-health industry by the IoT's contribution. E-HEALTH now cover more, and new health sector and that caused by the IoT Health system application like remote health treatment e-health wearable devise and smart hospital. etc., the new advancement of the IoT system Is took place in the new IoT health system according to the massive Data and the critical time that the health treatment need there for the Edge computing become an essential part in implementing any IoT Health application, for example, each hospital will have a local Edge server and in that server will contain a cache computing algorithm [63][64][65].

3. SYSTEM DESIGN

The problem of in-network caching can be divided into three defined sub-problems:

- **Content placement and content-to-cache distribution,** this is about the issue of which items of contents to place and how to distribute them to those nodes in which caching node.
- **Request-to-cache routing,** which addresses how the requester routes the content requests to an appropriate caching node containing a copy of the contents requested.
- **Cache allocation,** that discusses optimization of caching node positioning and size.

3.1 Content Placement

In general, content items can be proactively or reactively placed in in-network caches. Caches are pre-populated during off-peak traffic cycles with constructive information positioning. Using historical data and/or future forecasts, the position is usually calculated by an off-line optimization algorithm and replicated daily, approximately 24 hours a week. Several algorithms are proposed for automated placing of content under several objective functions and restrictions[23],In the case that an application moves beyond a buffer before reaching the data, a copy of the query in each node crossed defined as Left Copy Everywhere (LCE) should be left behind. These techniques, therefore, leads to a high degree of consistency as all caches use cache storage to hold similar artifacts along the delivery path, placing constructive material allows the caching nodes to be better deployed and improves performance. Nonetheless, since the caches community takes place at peak times with constructive positioning, working hours will be read-only. It enables multi-core caches to work lucklessly as no writing occurs. This also ensures better reading from storage technologies like SSD and HDD, which would give a less read output if concomitant writings were performed. However, two major disadvantages result from the simpler node implementation offered by proactive placement, first of all, it makes traffic demand changes more rigid, as any unpredictable variation in demand patterns would cause cache hit ratios to be reduced until proactive new content is positioned. Second, optimal content placement requires both data from cache operators–

cache topology, processing capabilities, cache sizes–and the contents providers, which can be very complicated to collect if the cache operator and content provider are different entities. This can also lead to data collecting by cache operators and content providers. As regards cache effectiveness, there is agreement that proactive placement is preferable to reactive placement, only in the case of certain workloads, such as Video on Demand (VoD)[25],[26], and adult video content[29], characterized by a small catalog of content and predictable variations of requests. In fact, Netflix, the world's largest provider of VoD content, uses its video caching infrastructure with proactive content placement [30], Other traffic types generally are characterized by rapid variations in the popularity for data that eliminate the benefits of a proactive optimized positioning. They have shown that even placing content items proactively with accurate knowledge of future demand would generate performance gains of only 1–18 percent compared to adaptive placements. All commercial site traffic CDN's refer to our knowledge, whether specifically optimized for static or adaptive web, fill their caches reactively. The specialized caching facilities of large-scale content providers such as Facebook photo storages and Google Web Cache.[21] also use reactive content placement. This strategy also includes the placement of the packet caches in network routers, selected by all ICN architectures [25].

3.2 Request routing

Routing approaches for applications can be primarily classified to two categories: opportunistic on and off the lane, the information requests are first routed from the requester into the nearest cache with on-path database routing. These are then routed over the caching network to the origin of data using the shortest path routing and served from a cache only if the information element on the query path on the specified node is accessible. The routing strategy is highly scalable because communication between caching nodes is not necessary and can be used with the proactive or reactive positioning of data. Nonetheless, reduced cache hits, especially in heavy-duty cache deployments, may occur because data cached next to the requestor is never reached on the shortest route to its source, it is worth noting that edge caching is also (simpler) an opportunity case. In edge cache applications are redirected to the nearest cache but are sent directly to the source of the data in the case of a cache error. This can be achieved in instance by Google Global Cache [23], which dynamically maps any cache installed on an ISP network to a

subset of requests and routes requests outside an ISP network if it fails the cache. In fact, queries can be managed by a neighboring node with off-pattern structured routing, even if not on the shortest source path. Nevertheless, it costs increased cooperation for the exchange of information on content access between caches, Off-path routing may be carried out by a mechanism of hierarchical or distributed data-to-cache resolution. A (logically) hierarchical object with a global view of cached contents is queried before routing an information query and returns the address of the closest node which stores the requested data element. In a centralized resolution process. Though, this method is only suitable for processes working under aggressive product placing or even responsive positioning so long as the content position is not adjustable. Several scalable off-path request routing algorithms have been proposed for reactive caching systems with a high content replacement rate (which also includes ICN architectures where items are cached with chunks of granularity. The main objective is to allow caching nodes to exchange states and route requests with each other in a lightweight way. The design of the application routing represents a clear compromise between scalability and efficiency. The limited scalability of off-road routing schemes particularly limits the availability of reactive cache and ICN architecture design choices, which are of our interest.

3.3 Caching Algorithm

Throughout previous work on cache replacement algorithm for other computer system implementations such as servers and storage systems, cache replacement algorithm was strongly rooted in the architecture for the delivery of information. Although these algorithms have been established for different purposes, their development is also suitable for distribution of information, while more regulations have also been expressly introduced for distribution of content.

3.3.1 Least Recently Used (LRU)

Least Recently Used (LRU), which substitutes the least requested element, is the most used for caching policy. This technique usually uses a double-linked server and functions as follows. Often moved to the top of the list when the item currently stored in the cache is submitted. Likewise, the requested item is placed at the top of the list on a query for a product not already in the cache, and the item is discarded at the bottom. It is made

popular by LRU with two key advantages. Furthermore, this reacts strongly to non-stationary events because its alternatives are only based on regeneration. The proportion between the optimal cache-hit ratio and the LRU-cache-hit ratio is not substantially worse than much-caching algorithms, LRU is not well-suited for simultaneous entry, given its simplicity and ease of use. Each quest culminating in a hit, and each substitution needs an object at the top of the double-linked list to be added. The serialized access to the list header, particularly in very parallel environments, will result in contention and detriment of results, Additional solutions to the problem of simultaneous application of LRU were proposed. One is CLOCK, which approximates LRU operation without shifting a cache struck element. In a rotating queue (that is, the term CLOCK) organizes objects explicitly. Each component is attached to a flag originally unset when applied and set to a cache hit. CLOCK holds a reference each iteration of the rotating queue to choose an object to be substituted. If the CLOCK finds an object whose flag is set, it sets the flag and moves to the next item, until an item is found and replaced with an unset flag. The hunt for the item to substitute starts from the spot where the last item has been substituted at the next substitution operation. In action, CLOCK is identical to FIFO, but with the exception that when an item is hit before the bottom of the list is reached, it is not deleted and provided a "second opportunity" [18].

In addition to the competition, LRU is not scanning resistance, as any scanning operation over many unpopular items would thrash out the cached content. In databases and disk based I / O, several legitimate workloads scan and read large sequences. This is a significant concern. In connected caching systems, this can also be a concern because adversarial workloads could scan thrash caches precisely. Furthermore, the distribution of contents is known to be affected by the one-time problem, i.e., many items only once requested. In addition to regency allocation decisions, the weak scan strength of LRU can be addressed with changes in the LRU design that incorporate frequency considerations. In fig 3.1 we can notice the flow chart for the LRU Program that we build our program based on.

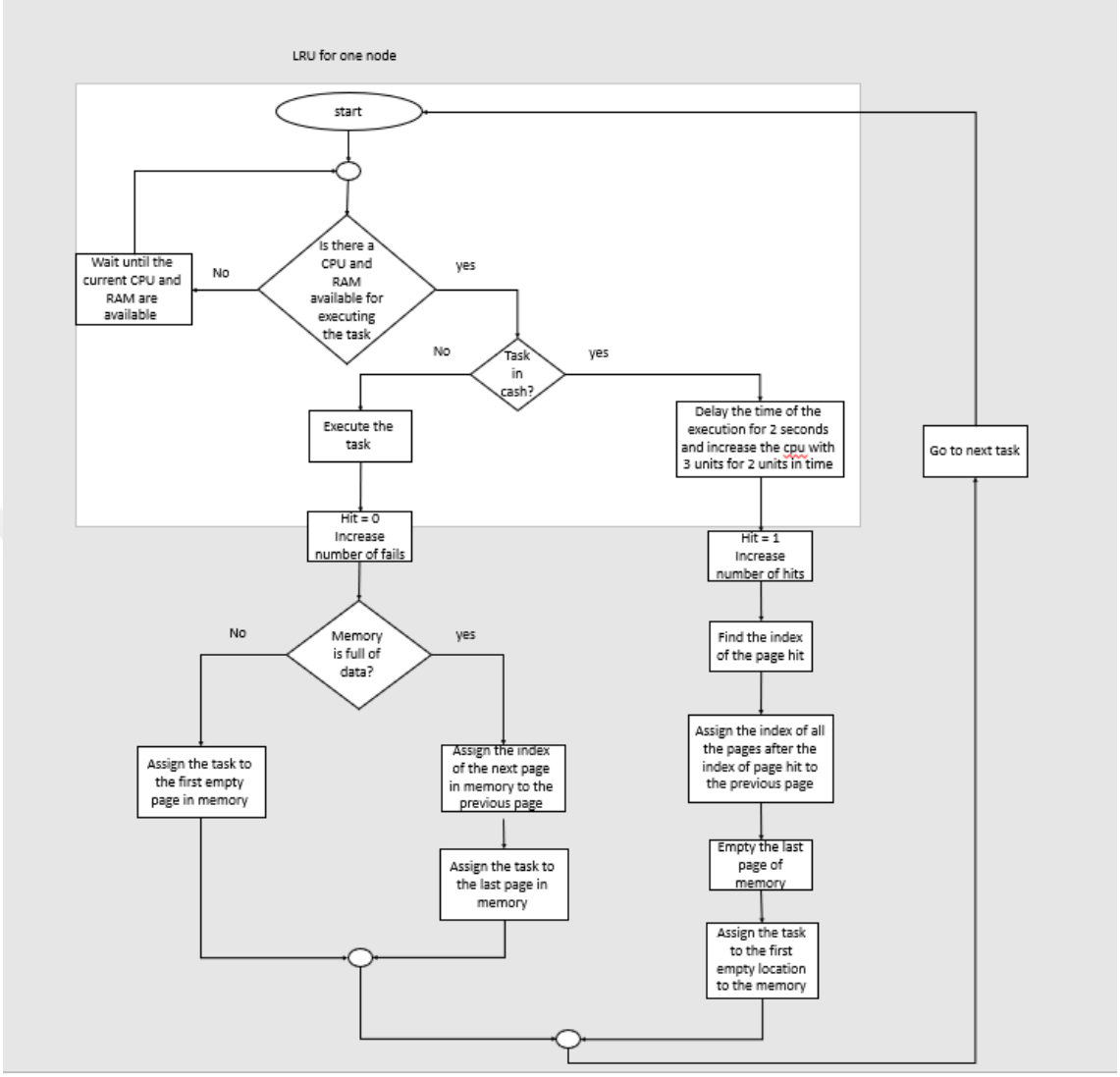

Figure 3.1: LRU Flowchart

LRU caching methodology depends on cache size and data size. If we consider that the LRU cache is equal to 3-page frames, then we can take size of the queue as 3 which considered as empty initially. The first three inputs will fill the queue makes it have no more space for another item. At this point, the oldest item in the queue (which happened to be the first in this example) is going to be sent to the client when a new item arrives to the cache. This methodology keeps going till data size become 0 (basically process all the data).

3.3.1.2 Doubly Linked List

A connected information structure composed of a series of sequencing records called nodes is a double-related chart in computer science. Every node has three areas: two connections (the previous and next branch ties in the node sequence) and one information region. The prior and next connections of the starting and finishing nodes signify a certain form of terminator, typically a sentinel node or node, to make it easier to cross the list. If only one sentinel node remains, the list is connected circularly via the sentinel node. It can be structured as two independently linked lists consisting of the same data objects in specific sequential orders, The first and last nodes on a double-related category are immediately reachable, which means they are accessible from beginning to end or from end to end, so they can cross the list from start to finish and thus the search for a node with specific value for data. In this scenario, the row is usually called head and tail. The list is accessed at the beginning or end of the list.

3.3.2 First in First Out (FIFO)

One rule is more suitable than LRU for concurrent implementations, though it transfers the First in First Out (FIFO) frequency to a lower cache level. The evicted object is the one first loaded into the inventory in compliance with the FIFO rule. Only when an item in the cache is necessary, the action of this policy varies from LRU. However, while this object is placed up at the top of the set in LRU, we upgrade the flowchart of the FIFO flow chart in three-step the first one as shown in figure 3.2 is working fancily but there is some major operation problem the first one is this flow chart does not check the load available, and the tasks are stored in the cache before the operation, and the correct thing is to the cache should be store in the cache after the task execution is finish and there

should be a two-unit step off time for the restoring the data from the cache if it repeated.

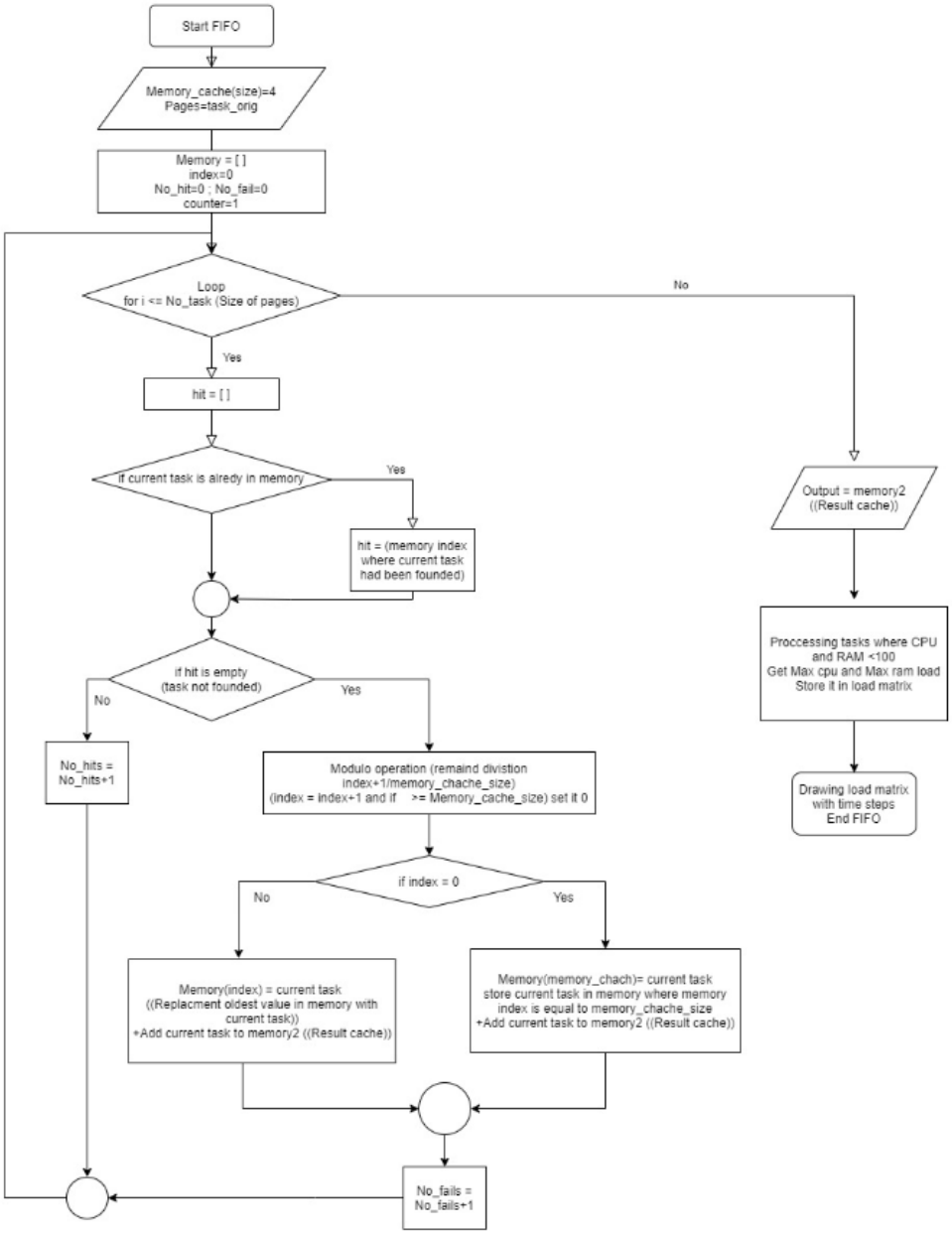

Figure 3.2: FIFO Flowchart step one

The next step was to improve the data load checking and the time step calculation unit; therefore, we upgrade the flow chart as shown in figure 3.3

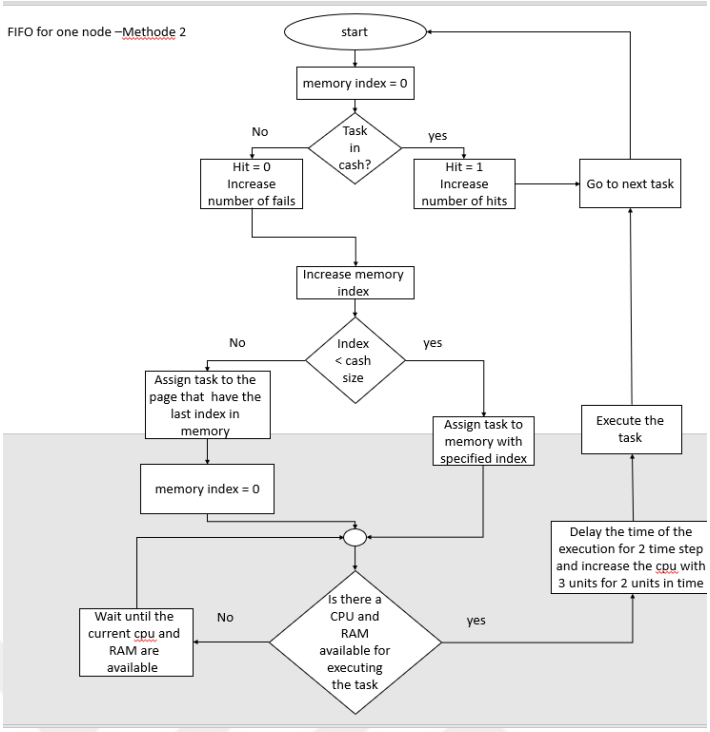

Figure 3.3: FIFO flow chart load addition

The third step as shown in figure 3.4 making the task store at the cache after the task calculation is finish and this move was made in FIFO because in LRU flow chart will be made the same step with adding the LRU algorithm strategy only the third step is representing the cache in a single node only, figure 3.4 representing the main FIFO algorithm code structure for a single node.

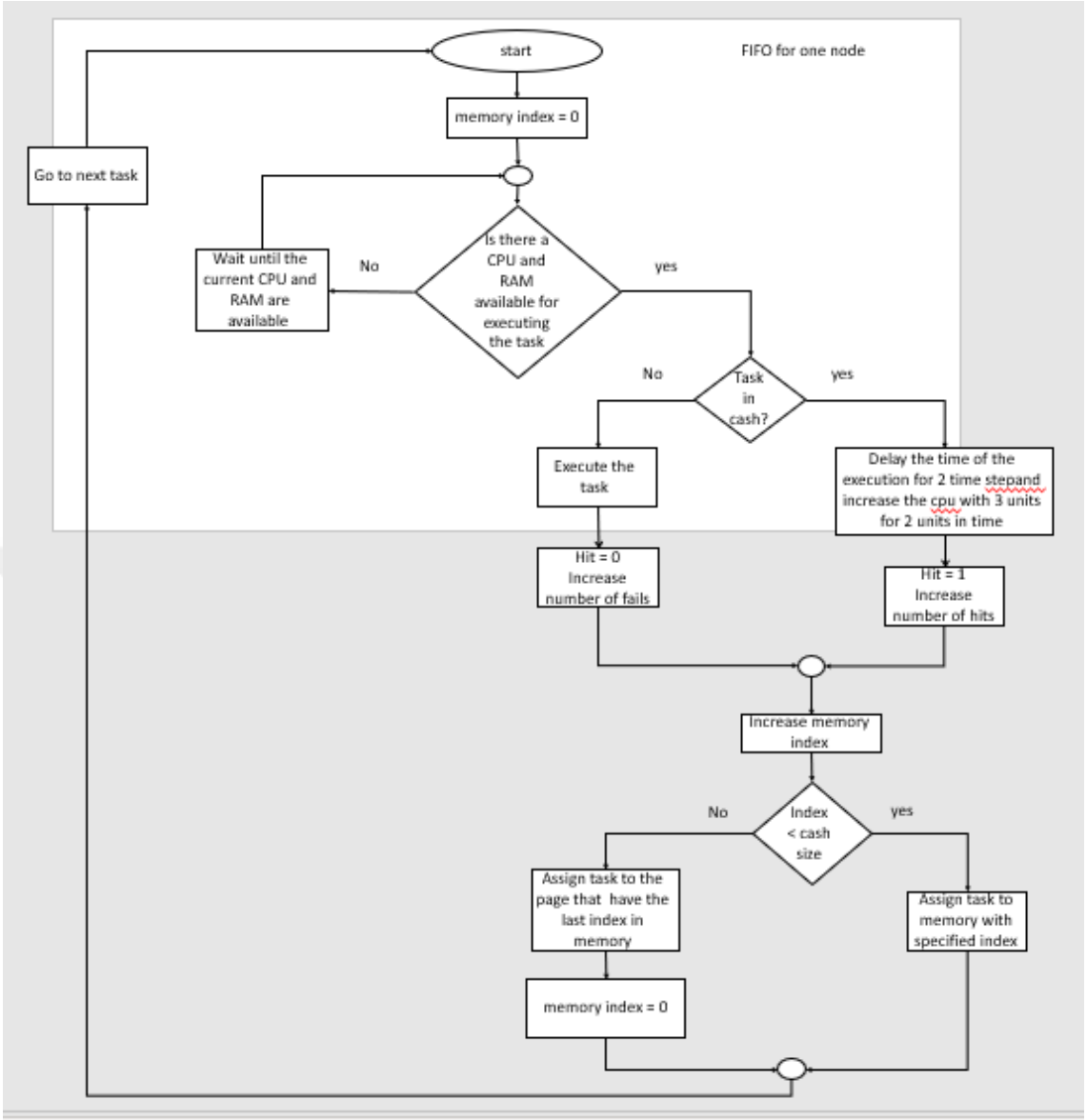

Figure 3.4: FIFO flow chart for a single node

3.4 Modeling

We now concentrate on theoretical models, which define the operations of such algorithms, after analyzing the layout space and algorithm for management of a cache network. This work in modeling has been given considerable attention, mainly because modeling cache performance is very difficult, even for simple isolated substitution

policies such as LRU or FIFO. When such caches are part of a network, the difficulty increases more.

3.4.1 Common assumptions

Market decisions are usually made in relation to three features: I the association (or failure) of subsequent applications, (ii) distribution of the importance of information and (iii) distribution of the volume of demand.

Most cache operations, whether within the CPU, database, I / O disk or the distribution of contents, assume that demand is consistent with the Independent Reference Model (IRM). This model assumes that requests are issued through a catalog of N items of finite size. The likelihood for each item I I= 1; N is stationary and unrelated to previous requests, denoting pi.

Although there are typically temporary local properties on operational workloads, IRM assumptions usually apply if both conditions apply.

- The cache serves applications from multiple applicants to minimize the temporal position inherent in a single applicant's application cumulatively.
- In contrast with the time frame for page updates, the cache churn duration is low.

3.4.1.1 Popularity distribution

While hypothesizing the distribution of product popularity, a heavy-duty distribution is usually expected, as past measurements of several heterogeneous user demand workloads have repeatedly demonstrated that conduct. The key hypothesis is that traffic follows a Zipf structure (in some publications referred to as zipf-like or general law on power) is especially and specifically for content distribution traffic. This is the normal presumption.

To be full, several works have proposed that material importance may be best based on implementations other than Zipf under certain circumstances. The traffic calculated in HTTP on an Orange connection in France is most suitable to remodeling, with a distinct Weibull for the face, a zipf for the wheel and another discrete Weibull for the tail (but with different settings). Nevertheless, given that a Zipf delivery throughout the collection typically models the popularity of content very accurately.

3.4.1.2 Size distribution

Several studies examined the distribution of product in volume for different workloads. In addition, the distribution of content volume is ideal for a heavy-duty distribution like a Pareto or a lognormal distribution. No previous work, however, has established substantial ties between prominence of product and data size. The common assumption is thus that all content is equal in size and the caches are measured in numerous items with the intent of cache design.

3.4.2 Single cache

There have been extensive attempts to create reliable and tractable structures with earliest results from the 1970s. As far as we learn, the first study has been focused on the exact formulation of the constant-state impact ratio of an IRM topic collection. He constructed an LRU cache as a series of Markov and States. Although this model provides the same cache impact ratio efficiency, it is not feasible for real caches because of the extremely large number of states, Thanks to LRU model's sophistication, a variety of theoretical versions with lower system magnitude have been suggested. O(NC) iterative approach for approximating the LRU replacement policy's steady-state strike ratio on IRM usage. It is also relevant to FIFO replacement schemes, but its complexity cannot be calculated because of the iterative nature, T is a random variable and is specific to any content in an LRU cache that is subject to the IRM request. If the cache is large enough, each item's characteristics fluctuate very little around its mean and can therefore be approximated with its mean value precisely. Moreover, it has been observed that the characteristic times values of every item are similar, so that all items in the catalog can be approximated by a single constant value.

3.4.3 Cache networks

The main source of complexity in cache modeling is that simplifying demand assumptions normally used to model a single cache (i.e. IRM requirements) usually do not apply when cache demand includes missing streams of other caches, as is common in cache networks. The missing stream of an IRM-subject LRU cache shows strong correlations between requests. Nevertheless, they have shown that, if the number of caches is high, the aggregated miss flow from a community of caches every target of

independent query appears to be IRM. As a result, the IRM principle could still be implemented on missed streams in certain particular cases like the caching trees with a large branching factor, rendering modeling easier to track, It is important to separate two types of topologies, feed-forward and random, before addressing models for database-networks. The topology of a cache network will be retrievable if the demands flow in one direction on every path only and the requested objects only flow in the other, and the graph with cache nodes and query flows are a Data Acyclic Graph (DAG) as guided edges (see for example Figures 3.3a and 3.3b). As is shown in Figure, a tree with leaf requestors and the original content of the root is a simple example of feed-forward topology. 3.3b. the second version. Database networks that are not routed are random (see Figure 3.3c, for example).

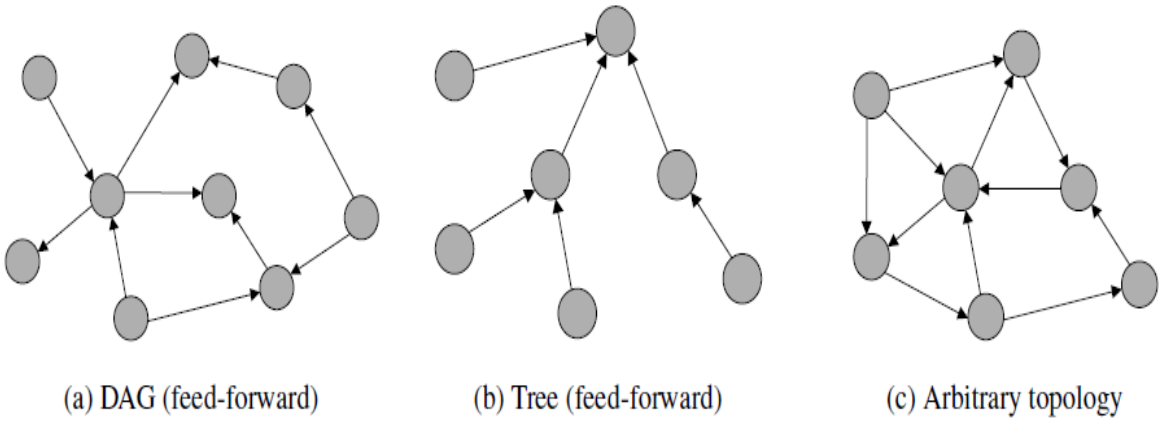

Fig.: 3.5 Examples of topologies for feed and conditional cache network

For determining the hit rate of cache, the distinction between feed forwards and random cache networks is significant because in feed forwards, by default, a cache hit ratio in a leaf node is not conditional on other caches (i.e. a node subject solely to exogenous cache requests from users with no misses from other caches). It makes an increasingly cache-by-cache estimation of the impact ratio of the entire network from leaf nodes. In addition, the mis flow of a cache will affect the demand on the same cache later in the arbitrary network because of loops. This complicates considerably the analysis of the cache tests all the task will plotted in the graph.

3.5 Data Generation

For the data Generation in the program and according to the flow chart in fig 3.4

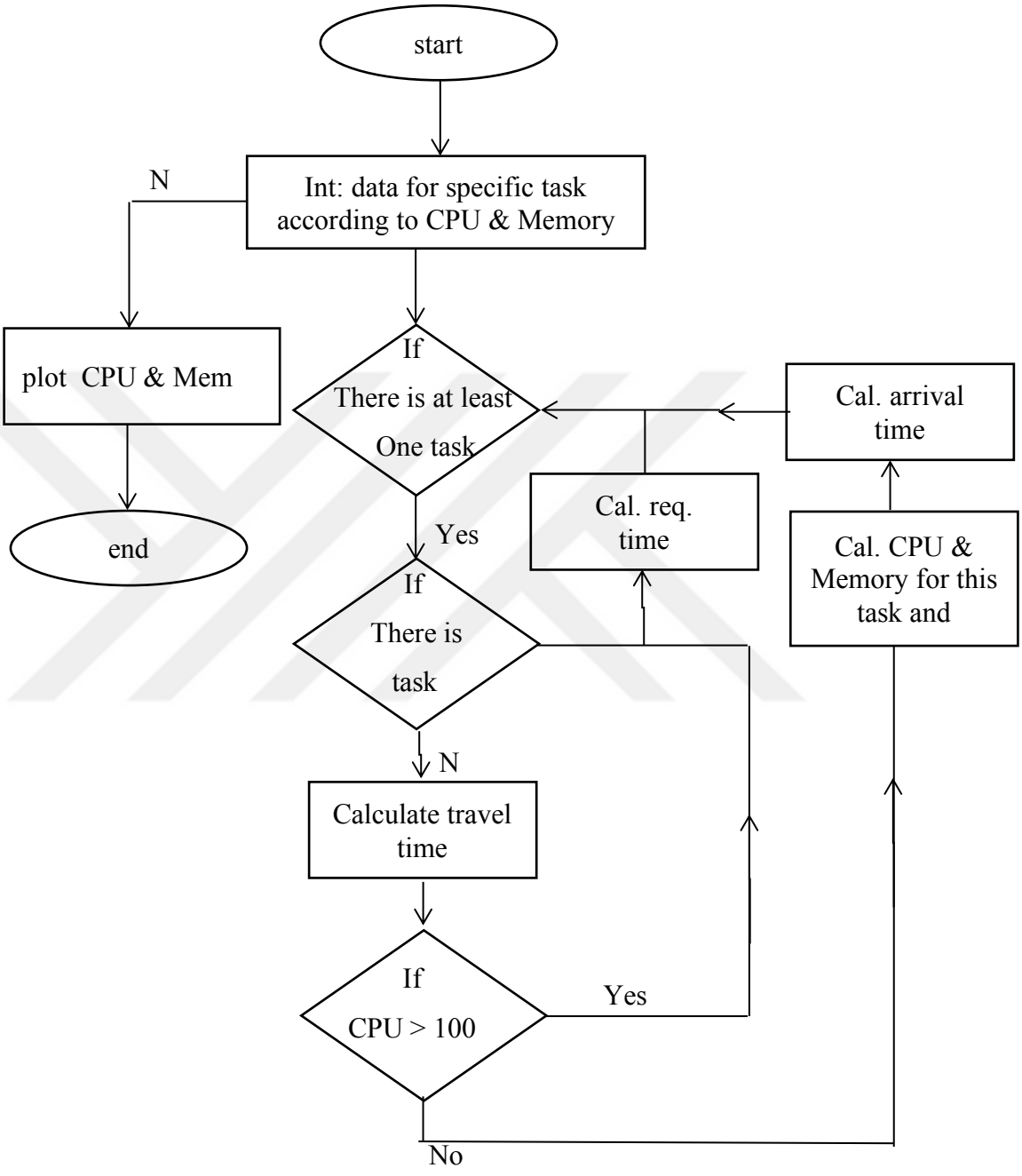

Figure 3.6: Data generation flowchart

the data is randomly generated by a pdf function that represents a random number of tasks repeated frequently this data contains a random number from CPU and Memory utilization, the data is generated sequence and uniformly as shown in fig3.5

no	arr	task	CPU	RAM	Dur	start	end	hit
1	16	2	20	35	31	16	47	0
2	31	1	35	50	76	31	107	0
3	37	1	35	50	0	47	47	1
4	50	1	35	50	0	50	50	1
5	50	1	35	50	0	50	50	1
6	57	3	27	42	68	57	125	0
7	68	9	33	47	68	107	175	0
8	82	1	35	50	0	125	125	1
9	92	1	35	50	0	125	125	1
10	108	1	35	50	0	125	125	1
11	123	6	39	27	55	125	180	0

Figure: 3.7 task generation for the CPU and Memory data

Then This data is forming the plot chart of the CPU and Memory and to change this data to that chart it should be run under several loop and programing process described in the above flow chart, Therefore after the data Generation initializing the if state gives two options if there is on new task generated (arrived) the program can plot the CPU and Memory and if there is a new task arrive the program will go under group of continuous data operation it will start to calculate the new arrival time for the task having sure that the utilization will not go above 100 percent and will continue to calculate the arrival time and the process and execution time for the task until all the task will be plotted in the graph. When the plot is generated each task is arrive in a specific time and execute in a different time line according to the load capacity and calculated in a specific duration time the cache hit column is checking the repeatable tasks in the data stream if the task is occur so it is a hit (1) if it not it will be a fault (0)

3.6 Remote Cache structure

According to the theoretical remote cache there is two use cases for the remote cache the first on the network will share the task to reduce the load between the nodes or the task will distributed between the node until each node is loaded to maximum value as shown in the flow chart in figure 3.6 for FIFO remote cache

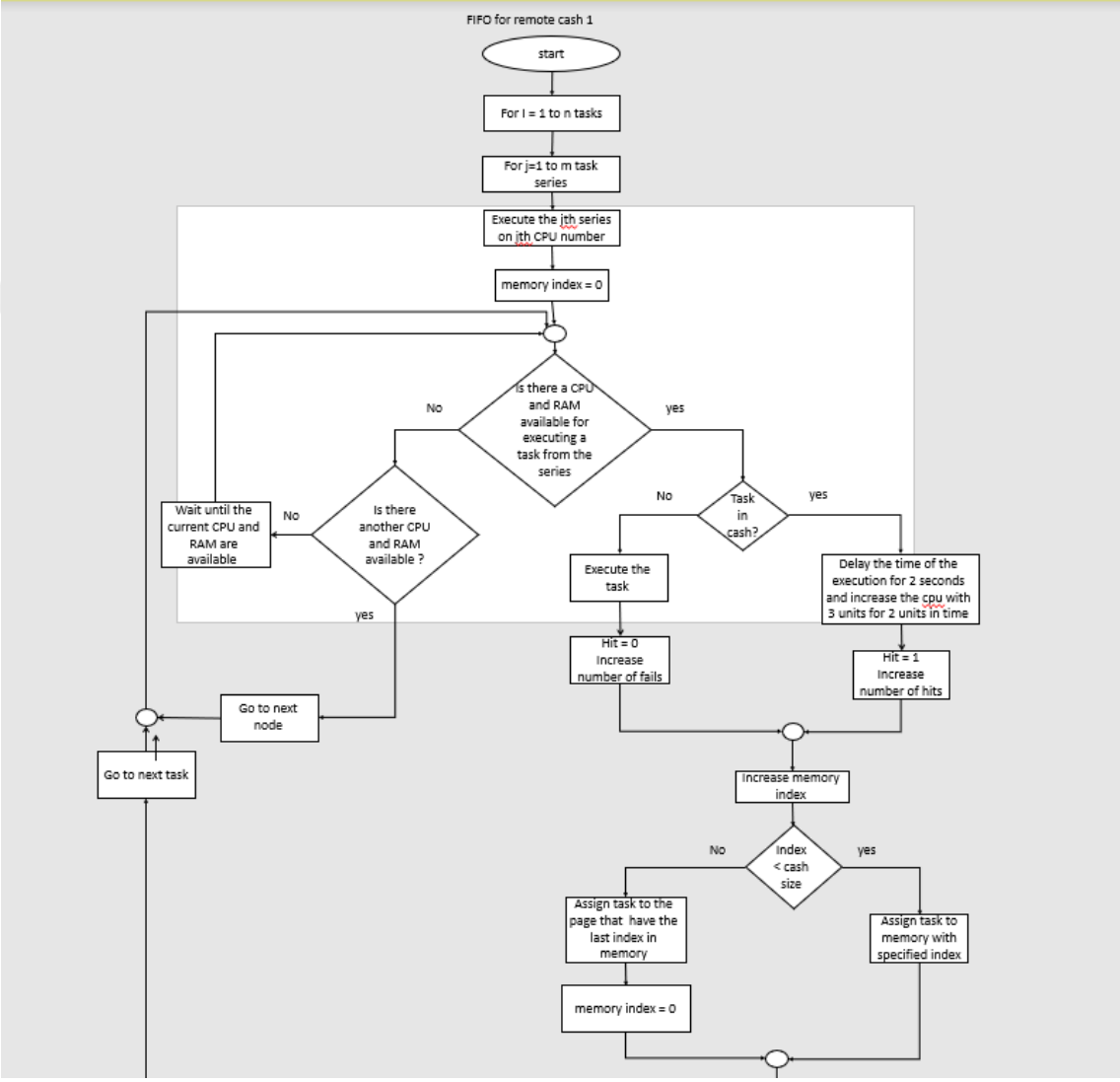

Figure3.8: FIFO remote cache CASE1

The same FIFO algorithm will be implemented but we will add multi-number of nodes

For LRU remote cache the flowchart will be as shown in fig (3.7)

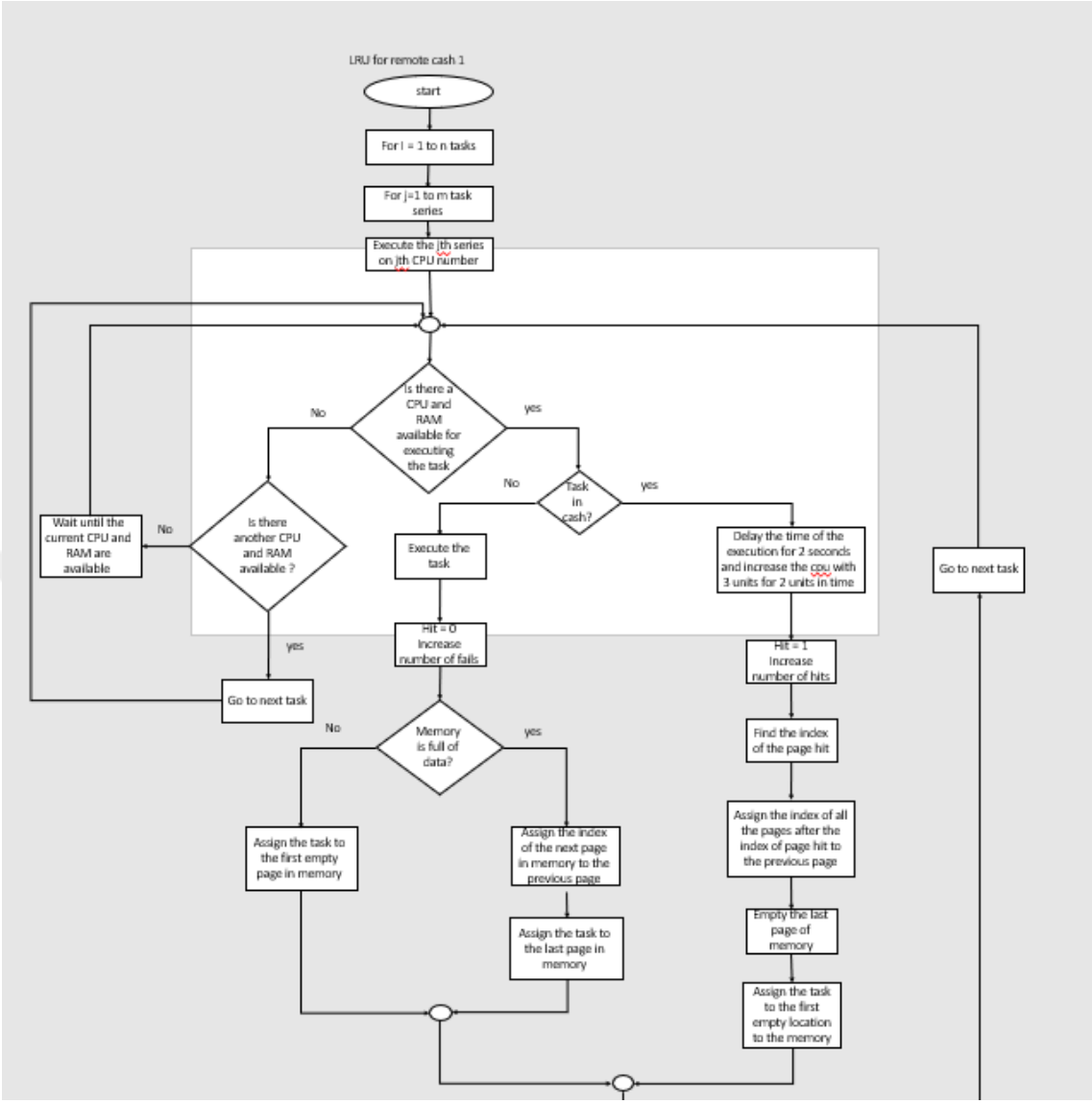

Figure 3.9: LRU Remote cache CASE1

We will implement the result for the same n number of tasks

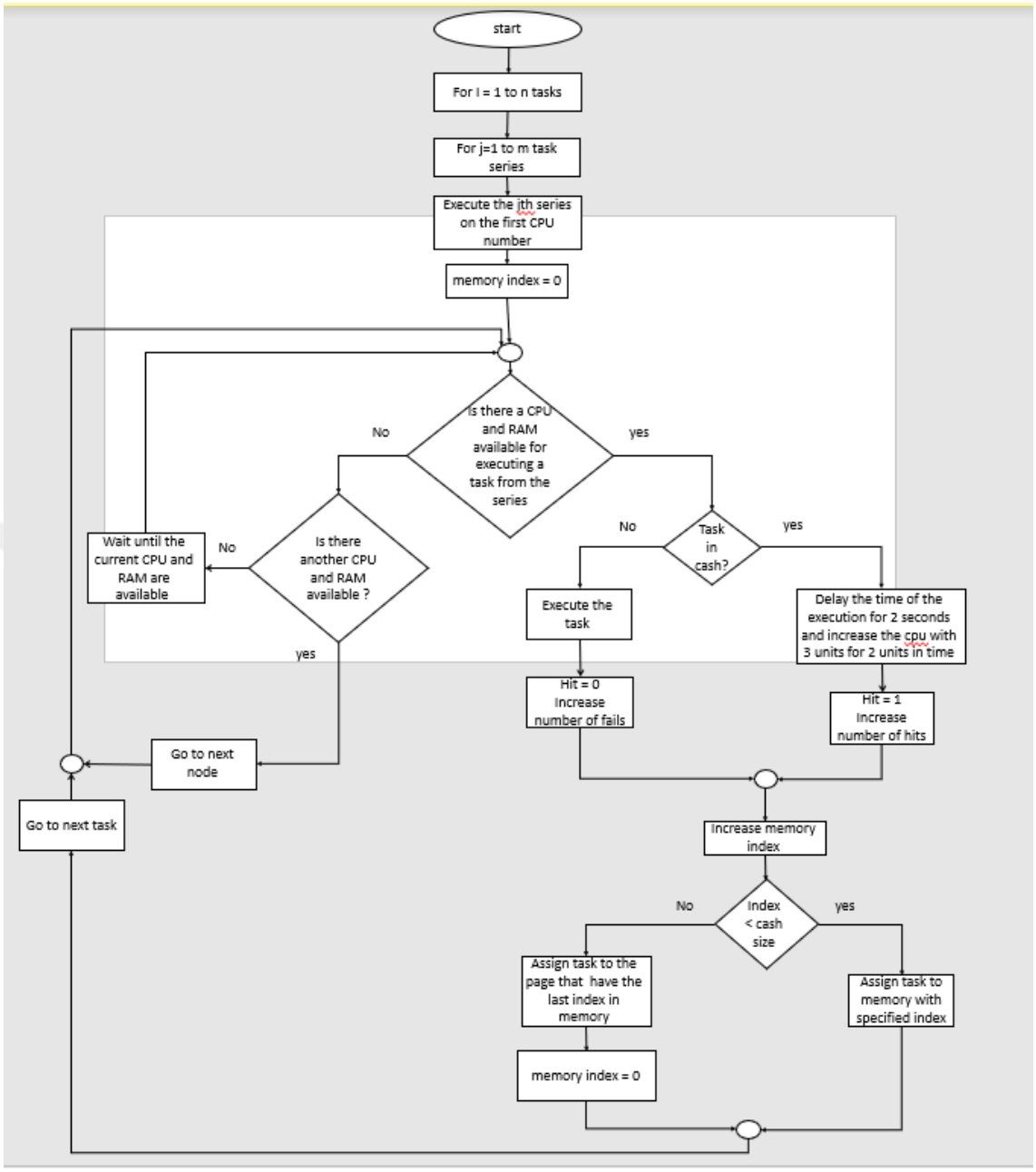

Figure 3.10: FIFO Remote cache CASE2

The different between the two use cases is the first case the task is distributed on the processer one by one until each CPU is full the other case the task is enter each node separately and the will start to share the task between each other in case of the load is full

3.7 System Work for a Single Node and Remote Cache

In the single node mode, the programs have five sub script

- FIFO.m
- Genetate_Task.m
- Sim_nocache
- Sim_fifo.m
- Sim_LRU.m
- MAINE.m

For this program structure the data is generated in the in generate task file have the following data (assigned number of tasks represent random number of CPU & RAM load value). Each task has it is own data label and this generated task is assigned in one queue to forum the CPU and Ram Graph have the load axis and the load axis then the Fifo sim and the FIFO will check the cache hit and cache the result (same goes for the LRU) the figure (3.9) show the single node flow the MAINE program is combining tree network case and the topology mode as shown in the figure 3.10 below

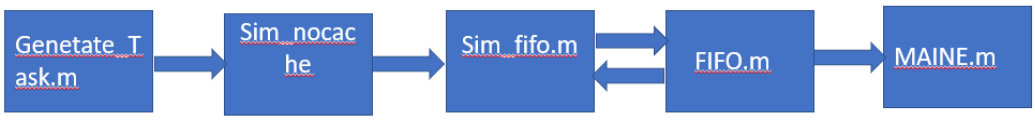

Figure 3.11: The single node flow

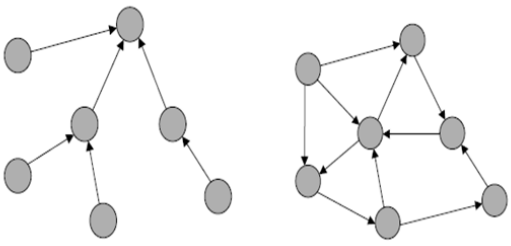

Figure 3.12: tree network case and the topology mode

4. RESULTS

This work was done using MATLAB for LRU and FIFO which they were compared to the same process without caching algorithm.

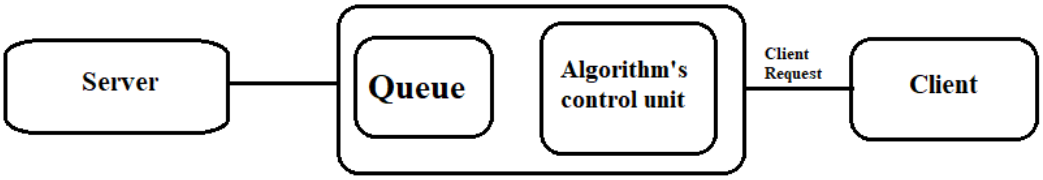

Figure 4.1: System architecture

According to figure 4.1 the single node system structure is based on three main components, sever (our cloud), caching system, and client (our node/sensor). The caching system is based of queue which is the local cache storage place for the caching system, and the algorithm's control unit which contains the algorithms caching method which in this study the methods used were LRU and FIFO, In our approach, cloud's data is generated randomly in form of tasks that has task processing requirement, storage requirements, time duration, and time arrival which the data processing is optimized by either LRU or FIFO, or, in no cache strategy, the local cache storage data will be used as queue. The optimized data will be sent to the client (node) for processing.

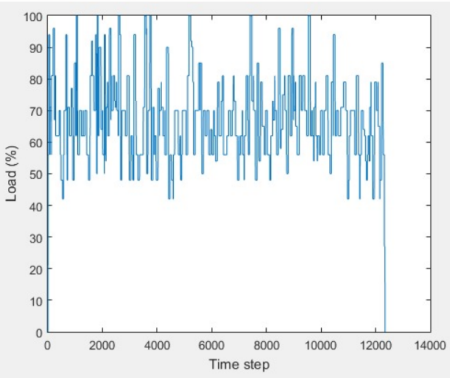

Figure 4.2: No cache CPU 1

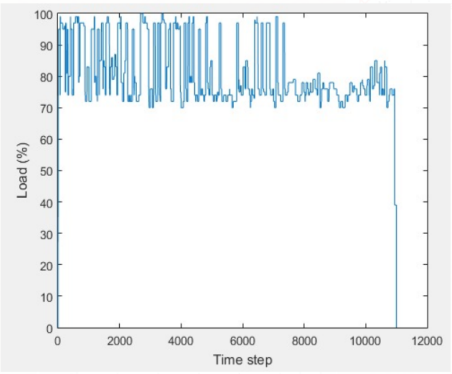

Figure 4.3: No cache CPU 2

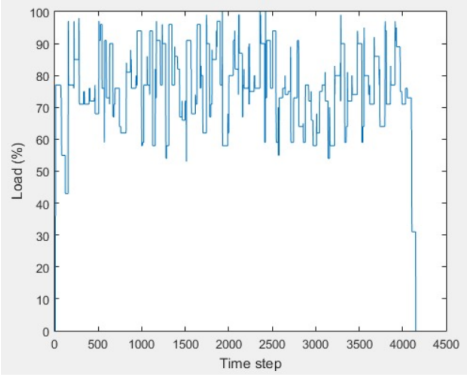

Figure 4.4: FIFO CPU

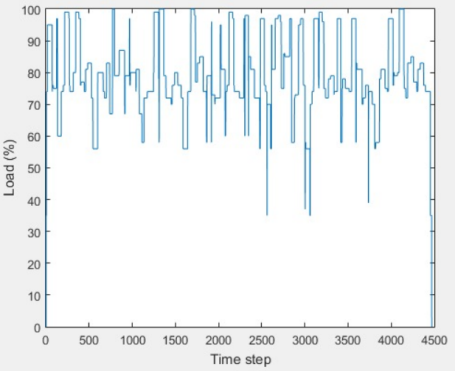

Figure 4.5: FIFO CPU2

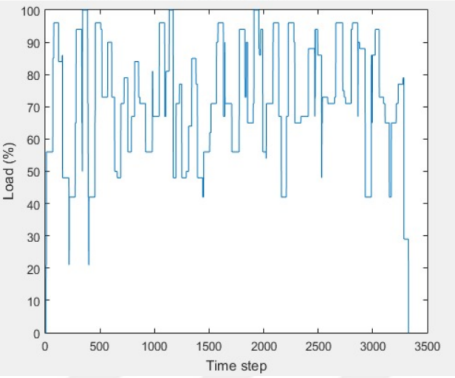

Figure 4.6: LRU Result CPU1

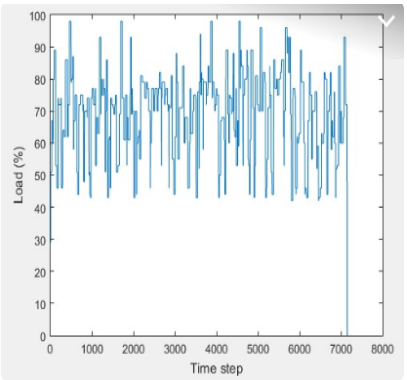

Figure 4.7: No cache CPU 2

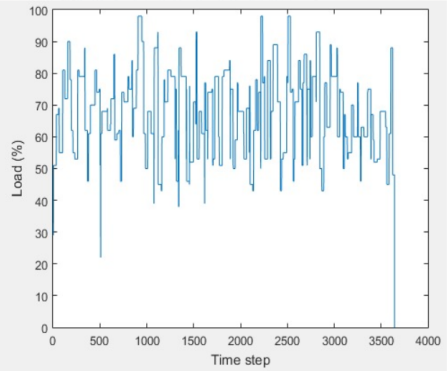

Figure 4.8: LRU CPU 2

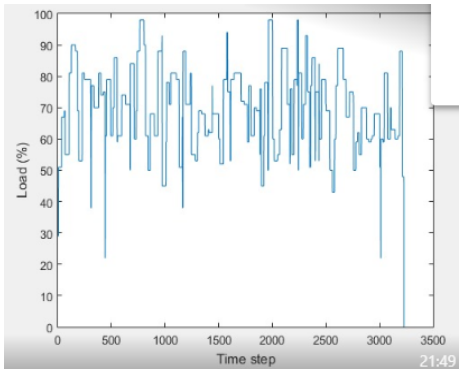

Figure 4.9: LRU CPU 2

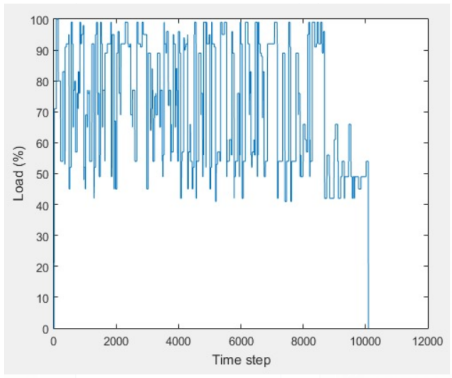

Figure 4.10: LRU CPU2

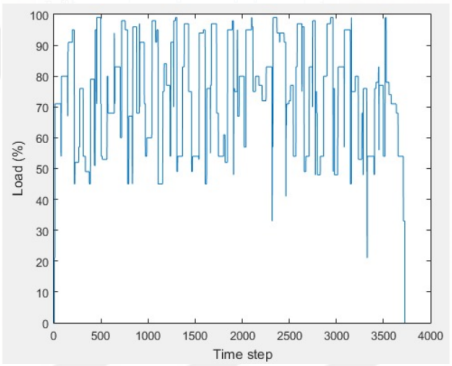

Fig 4.11: LRU CPU

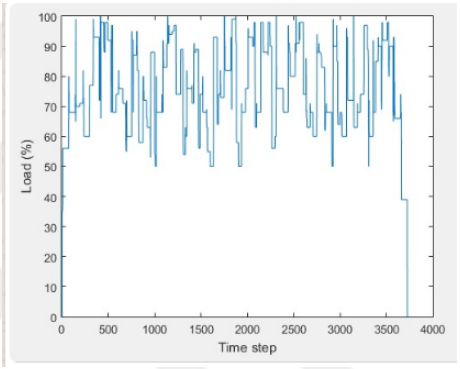

Fig 4.12: scenario 4 LRU FIFO no cache

The figure 4.2, 4.3, 4.4, 4.5, 4.6, to 4.11, represent the memory and CPU consumption for the system architecture (figure 4.1) when LRU and FIFO compared to the same system without any caching method. In this example the data size of total was 1000 task and the cache size are 10 tasks. As we see in this example LRU and FIFO were faster and used less time for processing and memory consumption compared to the system with no caching method. The figures show that LRU is significantly faster than FIFO in this example.

Table 4.1: The comparison between LRU, FIFO and No Caching in our study

Task 400	Time Requirement	Time Req.	Time Req.	Time Req.	Time Req.
LRU	3328	3886	3230	3727	3932
FIFO	4151	4470	3644	3727	4319
No cache	12341	11000	7152	10093	9074

Table 4.2: The Maximum comparison between LRU, FIFO and No Caching in our study

	Load	Load	Load	Load	Load	Load	Load
LRU	60%	60%	60%	60%	60%	60%	60%
FIFO	70%	70%	70%	70%	70%	70%	70%
NO CACHE	100%	100%	100%	100%	100%	100%	100%

Table 4.1 shows the comparison between LRU, FIFO, and no caching algorithm in terms of CPU usage, Memory consumption and Total time of the whole process. The results show that LRU was significantly faster compared to FIFO and when the system has no caching method and also the maximum CPU usage was less compare to the other methods, for the fifth scenario the FIFO result and the LRU is close to each other because it is depend on the changing of the arrival time so if the arrival time for the task is close the number of hit is increasing there for the LRU is effect much more if the arrival time much closer IF THE arrival time is far the fifo will be much efferent but still the LRU much better.

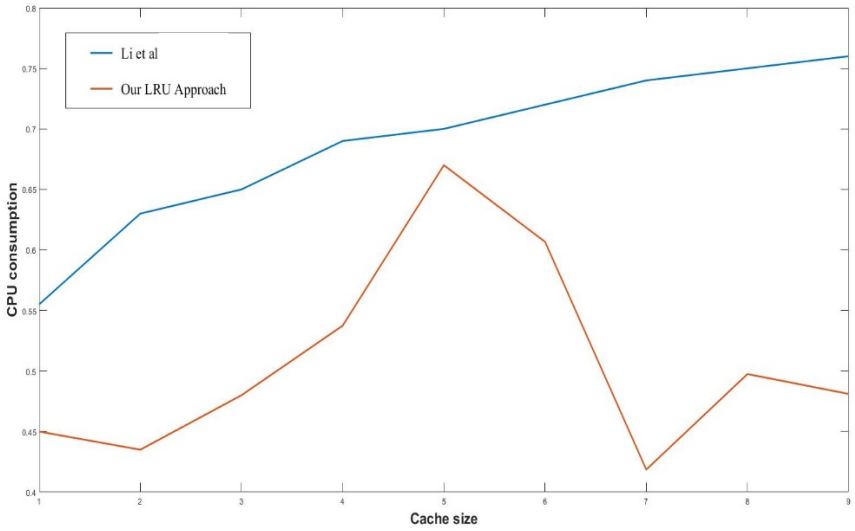

Figure 4.13 Comparison between our LRU strategy and Li et al [30]

Figure 4.8 shows the comparison between our LRU strategy and Li et al [30], which they used strategy called All-Edge-Caching (AEC) which also used LRU algorithm for replacement. The results showed that our strategy is great in terms of processing consumption compared to the other study that done by Li et al [35].

For the remote cache result the data result task=200, No.CPU=2

Table 4.3 CASE1:

Scenario	**N-nodes**	**Utilization**	**Unit of Time**
1	2	100%	6075
2	2	100%	6267
3	2	100%	5465
FIFO Result			
1	2	99%	1915
2	2	100%	2746
3	2	100%	2531

LRU Result			
1	2	99%	1716
2	2	99%	2413
3	2	99%	2265

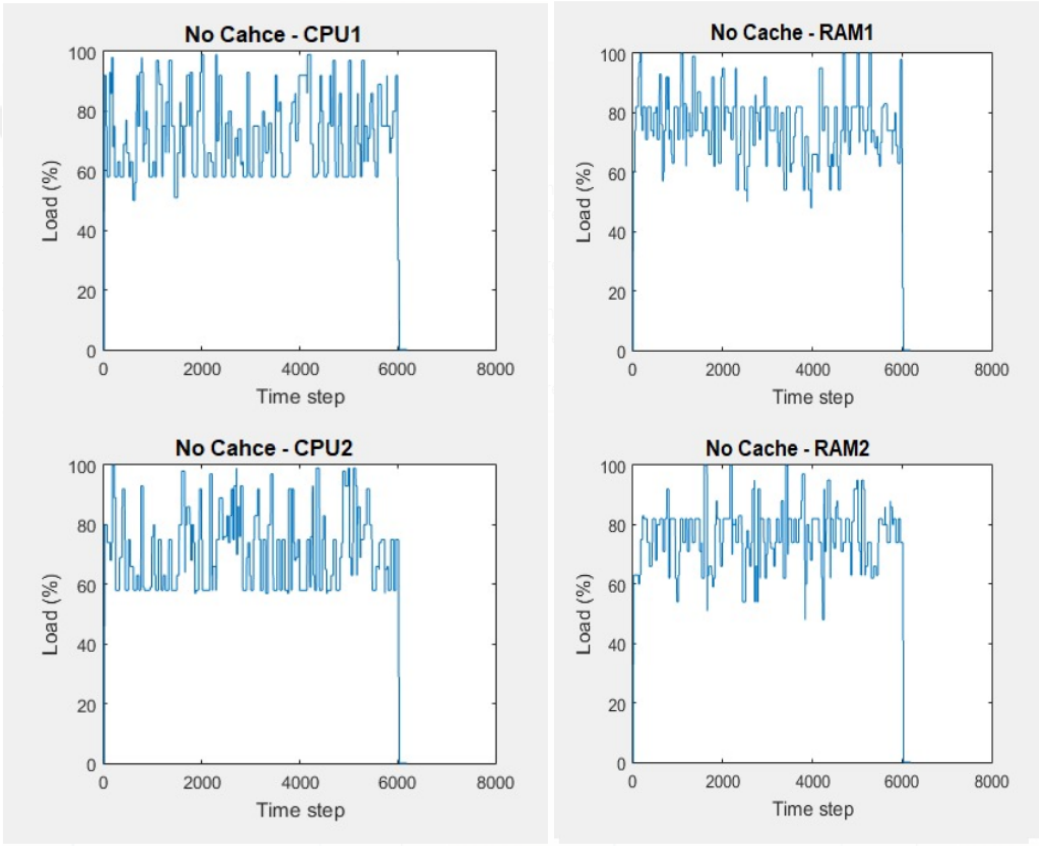

Figure: 4.14 node no cache result

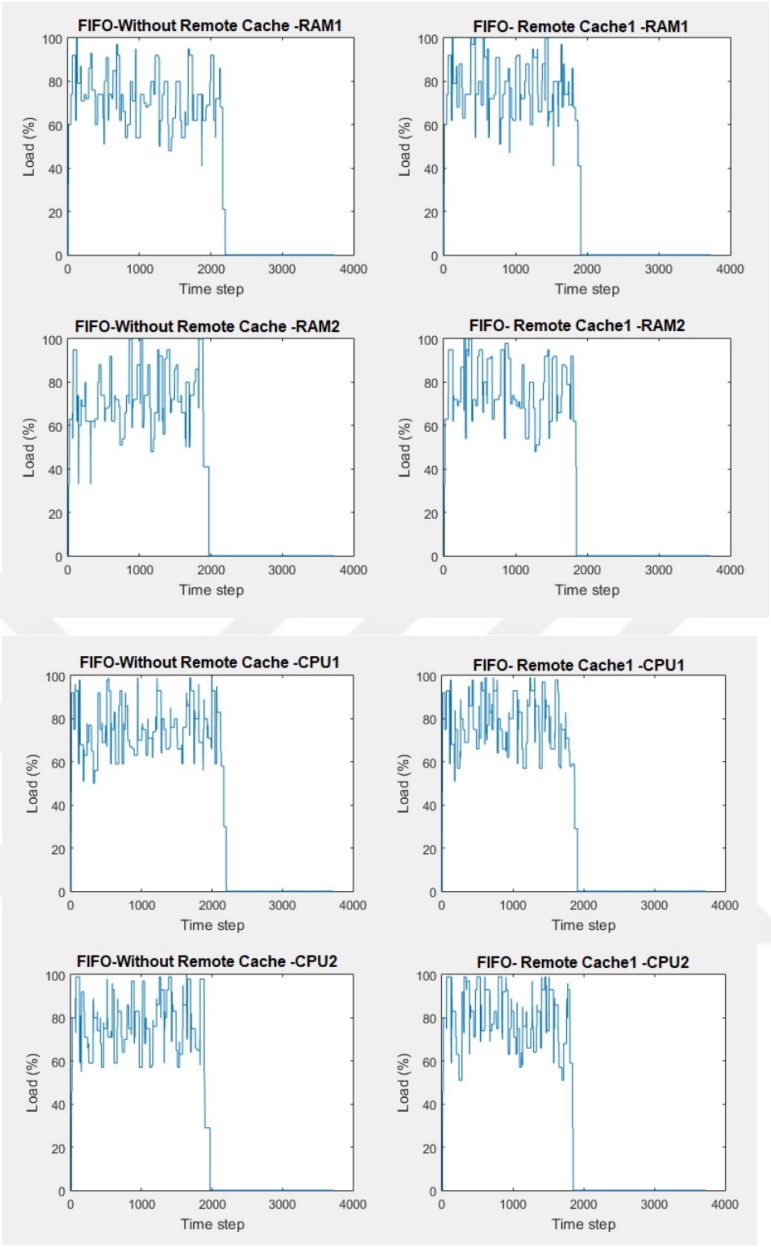

Fig: 4.15 Remote cache 2-node Case1 CPU

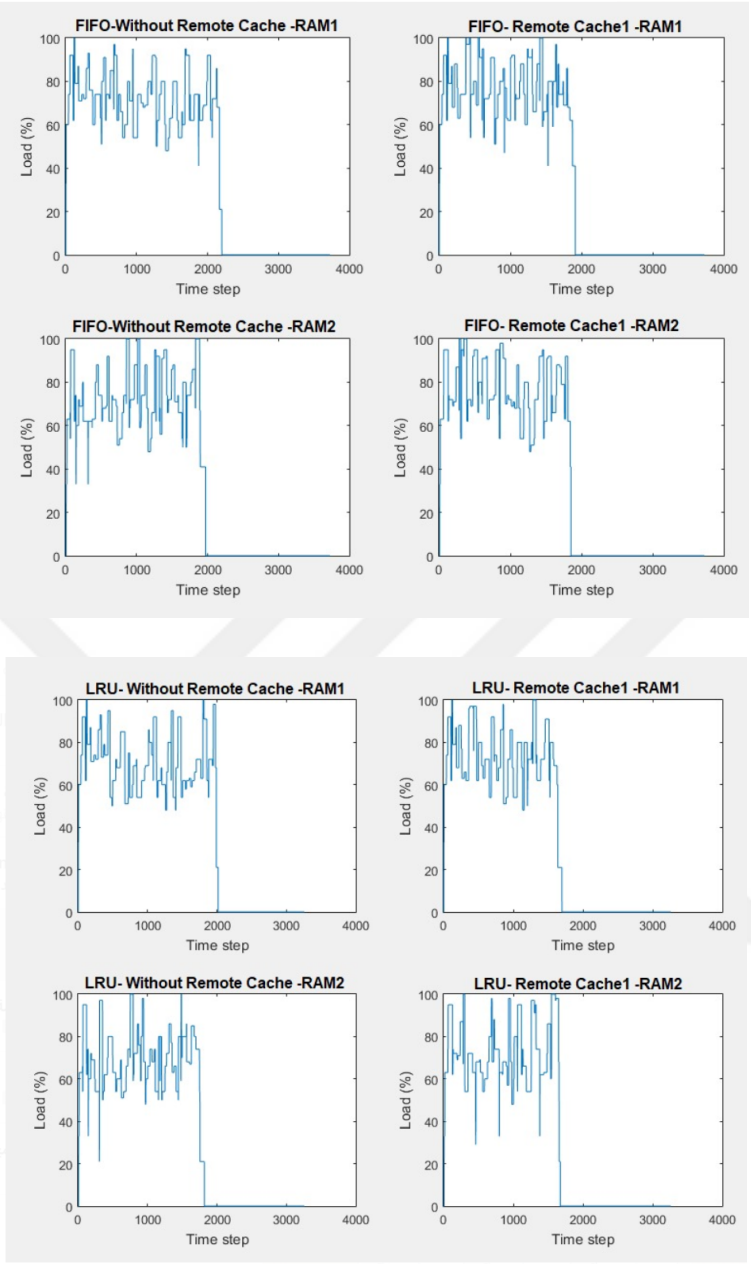

Fig: 4.16 Remote cache 2-node Case1 RAM

In the upper figures 4.14,4.15,4.16 For the single node result we can notice that the LRU have a significant better time process than FIFO because of the paging algorithm in the LRU and for all the caching result the load will still reach to 100 % but the time process will be much faster also in the remote Cache Case 1 the data will set as a one series entering the nodes one by one and the cache will hit the repeated data until the data set is finish , And for the case2 we did not use it result because of some technical issues in the implementation of the code and for the increasing of number of node we

could not get the result because of the huge time it took to get the result because of the PC that we operate the program with also the better task number for our PC for the remote cache result was 200 task because the PC ability was limited but for the single node the number was open there for we could not check the ability of the program and the result if we increase the number of the node.

5. CONCLUSIONS

The IoT is the most vulnerable Internet infrastructure to mass transport and the complexity of the communication. This is because the amounts of clever things are growing, and their implementations have strict requirements. ICN has therefore attracted the attention of the IoT group, encouraging the use of the definition based on information in IoT networks. The ability of ICN to help IoT applications was discussed in this dissertation. The NDN architecture has proven to be the best information focused architecture for IoT environments among several proposals from ICN architectures.

A recipient-driven, pull-based, reliable connection-free framework for interaction with simple and scalable data access, energy efficiency, and flexibility supports is described in Named Network Networking. While the use of NDNs in IoT networks has acquired numerous benefits, the demands made by IoT implementations delay the introduction of the information centric IoT model. This latter ecosystem is first heavily restricted by capital. On the other hand, the data availability and low response latency are very high. The most important feature that can meet the needs of IoT users is the in-network caching. The caching solution will, therefore, allow material for the users still accessible without growing data size and network resilience. In fact, because IoT information is dynamic and often modified, material residing at cache nodes is potentially outdated. In addition, certain critical IoT applications impose a strict data freshness requirement. In addition, IoT devices are generally mobile. It is worth noting As a consequence, after a move, farmers are no longer accessible, In this work, we used our approach for IoT edge caching using LRU and FIFO for single node and N-number of nodes (Remote Cache) which both of them were compared to the same system architecture without caching algorithm. The results showed that our LRU strategy was significantly better in terms of CPU usage, memory consumption and total time. Our approach showed that it is capable on reducing time for data and make it executable and also executed in much less time which provide the network with higher up the data processing limitation using less processing and less memory possible, the Remote cache for the two use cases we will notice a further improvement for the performance of the node network ad that will mitigate this era needs

.

APPENDIX A: TASK GENERATION

```
function [load, t_end,NCTask] = sim_nocache(tasks)
%define max capacity for cpu and ram
c_cpu=100;
c_ram=100;

%number of tasks finished/executed
task_finished=0;

%generate tasks according to parameters in function

% load array contains actual load of cpu and ram
% load matrix has 3 columns: time, cpu_load, ram_load
% max of time: timesteps should be increased if necessary
timesteps=sum(tasks(:,6));
load=zeros(timesteps,3);
% initialize first dimension of the load array
for i=1:timesteps
    load(i,1)=i-1;
end;



% total number of taks
no_task=size(tasks,1);

%generate 3 columns for actual start time end time and flag for cached
newcol=zeros(no_task,3);
task=[tasks newcol];

% execution of tasks starting with the first
for i=1:no_task
    % assign current task values to scalar variables
    tt=task(i,2);     % tt: task arrival time
    t_type=task(i,3); % task type
    t_dura=task(i,6); % task execution time
    t_cpu=task(i,4);  % task cpu load
    t_ram=task(i,5);  % task ram load
    flag=0;           % flag for task being assigned to a free time
slot
    %search for available cpu and ram capacity for the duration of
    %execution time for task
    while flag==0
        c_max=max(load(tt:tt+t_dura,2)); % max cpu load during the
period
        % tt and tt+execution time
        r_max=max(load(tt:tt+t_dura,3)); % max ram load during the
period
        % tt and tt+execution time
        if c_max+t_cpu<=c_cpu && r_max+t_ram<=c_ram

            % if the cpu and ram have available resource for task
            for j=1:t_dura       % increase cpu and ram load for
                % the period between tt and
                % tt+execution time
```

```
                load(tt+j-1,2)=load(tt+j-1,2)+t_cpu;
                load(tt+j-1,3)=load(tt+j-1,3)+t_ram;
            end
            flag=1;              % indication that task is assigned
            task(i,7)=tt;        % set task actual start time
            task(i,8)=tt+t_dura; % set task actual finish time

        else
            % cpu or ram does not have available resource for task
            % increment the time
            tt=tt+1;
        end   %if
    end  %while
end
% set the time to finish all tasks
t_end=max(task(:,8));
NCTask=task;
end
```

APPENDIX B: LOAD NO CACHE

```
function [No_hits, No_fails, load, t_end] = sim(tasks,
cpu_count, type)
%define max capacity for cpu and ram
c_cpu = 100;
c_ram = 100;


%generate tasks according to parameters in function

% load array contains actual load of cpu and ram
% load matrix has 3 columns: time, cpu_load, ram_load
% max of time: timesteps should be increased if necessary
load=zeros(1,3,1);

if type == 1

    memory_cash = 4;

    for i=1:cpu_count
        [No_hits, No_fails, memory] =
FIFO(tasks,memory_cash);
        size(memory)
        size(tasks)
        tasks(:,:,i) = memory;
    end;

end;

if type == 2
    memory_cash = 4;
    [No_hits, No_fails, memory] = LRU(tasks,memory_cash);

    tasks = memory;
end;


% total number of taks
no_task=size(tasks,1);

%generate 3 columns for actual start time end time and flag
for cached

newcol=zeros(no_task,4);


task=[tasks newcol];
for t=1:task(1,2)
```

```
    load(t,1,1:cpu_count) = t;
end;



% execution of tasks starting with the first
for i=1:no_task
    tt=task(i,2,j);     % tt: task arrival time
    flag = 0;          % flag for task being assigned to a
free time slot
    %search for available cpu and ram capacity for the
duration of
    %execution time for task

    if tt > size(load,1)
        for t=task(i-1,2,j):task(i,2,j)
            load(t,1,1:cpu_count) = t;
        end;
    end;
    while flag==0

        cpu_cnt = 1;
        [cpu_no, load, task, flag, tt] =
check_CPU_RAM(cpu_cnt, j, load, i, task, c_cpu, c_ram,
flag, tt);

    end  %while
end
% set the time to finish all tasks
t_end=max(task(:,8,j));


end
```

APPENDIX C: FIFO LOAD (simFIFO)

```
function [load, t_end, No_hits, No_fails] = sim_FIFO(tasks,
cpu_count)
%define max capacity for cpu and ram
c_cpu = 100;
c_ram = 100;


%generate tasks according to parameters in function

% load array contains actual load of cpu and ram
% load matrix has 3 columns: time, cpu_load, ram_load
% max of time: timesteps should be increased if necessary
load=zeros(1,3,1);

memory_cash = 4;
[No_hits, No_fails, memory] = FIFO(tasks(:,:,1),memory_cash);

tasks = memory;


% total number of taks
no_task=size(tasks,1);

%generate 3 columns for actual start time end time and flag for
cached
newcol=zeros(no_task,4);
task=[tasks newcol];

for t=1:task(1,2)
    load(t,1,1:cpu_count) = t;
end;

% execution of tasks starting with the first
for i=1:no_task
    tt=task(i,2);     % tt: task arrival time
    flag = 0;         % flag for task being assigned to a free
time slot
    %search for available cpu and ram capacity for the duration
of
    %execution time for task


    if tt > size(load,1)
        for t=task(i-1,2):task(i,2)
            load(t,1,1:cpu_count) = t;
        end;
    end;

    while flag==0

        cpu_no = 0;
```

```
        [cpu_no, load, task, flag, tt] =
check_CPU_RAM(cpu_count, cpu_no, load, i, task, c_cpu, c_ram,
flag, tt);

    end  %while
end
% set the time to finish all tasks
t_end=max(task(:,8));

//check fifo
function [memory, index] = check_FIFO(hit, index, memory, task,
cpu_no)
index(cpu_no) = mod(index(cpu_no)+1,size(memory,1));

if index(cpu_no) == 0
    memory(size(memory,1), cpu_no) = task;
else
    memory(index(cpu_no), cpu_no) = task;
end;
end
```

APPENDIX D: LRU LOAD

```
function [load, t_end,LRUTasko, No_hits, No_fails] = sim_LRU(tasks)
%define max capacity for cpu and ram
c_cpu=100;
c_ram=100;

%number of tasks finished/executed
task_finished=0;

%generate tasks according to parameters in function


% load array contains actual load of cpu and ram
% load matrix has 3 columns: time, cpu_load, ram_load
% max of time: timesteps should be increased if necessary
timesteps=sum(tasks(:,6));
load=zeros(timesteps,3);
% initialize first dimension of the load array
for i=1:timesteps
    load(i,1)=i-1;
end;

memory_cash = 4;
[No_hits, No_fails, memory] = LRU(tasks,memory_cash);

tasks = memory;

% total number of taks
no_task=size(tasks,1);

%generate 3 columns for actual start time end time and flag for cached
newcol  = zeros(no_task,4);
task=[tasks newcol];


% execution of tasks starting with the first
for i=1:no_task
    % assign current task values to scalar variables
    tt=task(i,2);     % tt: task arrival time
    t_type=task(i,3); % task type
    t_dura=task(i,6); % task execution time
    t_cpu=task(i,4);  % task cpu load
    t_ram=task(i,5);  % task ram load
    flag=0;           % flag for task being assigned to a free time
slot
    %search for available cpu and ram capacity for the duration of
    %execution time for task


    while flag==0

        c_max=max(load(tt:tt+t_dura,2)); % max cpu load during the
period
        % tt and tt+execution time
```

```
        r_max=max(load(tt:tt+t_dura,3)); % max ram load during the
period
        % tt and tt+execution time

        if c_max+t_cpu<=c_cpu && r_max+t_ram<=c_ram

            % if the cpu and ram have available resource for task
            for j=1:t_dura       % increase cpu and ram load for
                % the period between tt and
                % tt+execution time
                load(tt+j-1,2)=load(tt+j-1,2)+t_cpu;
                load(tt+j-1,3)=load(tt+j-1,3)+t_ram;
                load(tt+j-1,4)=t_type;


            end
            flag=1;              % indication that task is assigned
            task(i,7)=tt;        % set task actual start time
            task(i,8)=tt+t_dura; % set task actual finish time




        else
            % cpu or ram does not have available resource for task
            % increment the time

            tt=tt+1;

        end   %if

    end  %while


end
% set the time to finish all tasks
t_end=max(task(:,8));
LRUTasko=task;
end
```

APPENDIX E: FIFO CALCULATION

```
function [No_hits, No_fails, memory2] = FIFO(pages,memory_cash)

memory = [];

index = 0;
No_hits = 0;
No_fails = 0;
counter = 1;
for i=1:size(pages,1)
    hit = [];

    for j=1:length(memory)
        if pages(i,3) == memory(j)
            hit = j;
        end;
    end;

    if isempty(hit) == 1
        index = mod(index+1,memory_cash);
        if index == 0
            memory(memory_cash) = pages(i,3);
            memory2(counter,:) = pages(i,:);
            counter = counter+1;
        else
            memory(index) = pages(i,3);
            memory2(counter,:) = pages(i,:);
            counter = counter+1;
        end;
        No_fails = No_fails + 1;
    else
        No_hits = No_hits + 1;
    end;
end;

end
```

APPENDIX F: LRU CALCULATION

```
function [No_hits, No_fails, memory2] = LRU(pages,memory_cash)
memory = [];
index = 0;
cash = [];
No_hits = 0;
No_fails = 0;
counter = 1;
for i=1:size(pages,1)
    hit = [];

    for j=1:length(memory)
        if pages(i,3) == memory(j)
            hit = j;
        end;
    end;

    if isempty(hit) == 1
        if length(memory) < memory_cash
            index = index+1;
            memory(index) = pages(i,3);
            cash(index) = pages(i,3);

            memory2(counter,:) = pages(i,:);
            counter = counter+1;
        else
            LR = cash(1);
            for j=1:length(cash)-1
                cash(j)=cash(j+1);
            end;
            cash(length(cash))=pages(i,3);
            memory(find(memory==LR)) =pages(i,3);

            memory2(counter,:) = pages(i,:);
            counter = counter+1;
        end;
        No_fails = No_fails + 1;
    else
        k = find(cash==pages(i,3));
        for j=k:length(cash)-1
            cash(j)=cash(j+1);
        end;
        cash(length(cash))=pages(i);
        No_hits = No_hits + 1;
    end;
end;
end
```

APPENDIX G: REMOTE CACHE

```
 % clear

% clc
% Test basic cache functions
LRUSize=FIFOSize;
% DATASize=100;
ProcessedData=1;
% Create a cache with a maximum of 3 items
lru = LRU(LRUSize);
% lsize=lru.size();
t=0;

mem=[];
CPU=[];
TempCPU=[];
TimeLRU=[];
TimeLRUTemp=Timenocache;

% AllData=[];
i=1;

% DATASize=100;
% AllData=[];
% AllDataCPU=[];
% AllDataMem=[];
%
% MaxTask=9;
% r = rand(1, MaxTask);
% for i=1:10
%     while r>=1
%         r=rand;
%     end
% end
% r = r / sum(r);
%
% T=0:MaxTask;
%
% for i=1:MaxTask
%     nub=DATASize*r(i);
%     if nub-fix(nub)<0.5
%         fnub=fix(nub);
%     else
%         fnub=fix(nub)+1;
%     end
%     for j=1:fnub
%         AllData=[AllData i-1];
%     end
% end
```

```
%
% AllDataTemp=AllData;
% AllData=[];
% [p c]=size(AllDataTemp);
% if c>100
%     for i=1:100
%        AllData(i) = AllDataTemp(i);
%     end
% elseif size(AllDataTemp)<100
%     AllData(100) = MaxTask;
% else
%     AllData=AllDataTemp;
% end
% % AllDataTemp=AllData;
% AllData=AllData(randperm(length(AllData)));
% %     lru.size()
%
% % num2str(fix(ranber));
% AllDataFinal=AllData;
% AllData=[];
% for i=1:100
%     sranber = num2str(fix(AllDataFinal(i)));
%     AllData=[AllData;sranber];
% end
% %     lru.size()
% r = rand(1, DATASize);
% for i=1:DATASize
%     r(i)=rand;
%     while r(i)<0.1
%         r(i)=rand;
%     end
%     r(i)=r(i)*10;
%     r(i)=fix(r(i))*10;
%     AllDataCPU=[AllDataCPU;r(i)];
% end
%
% r = rand(1, DATASize);
% for i=1:DATASize
%     r(i)=rand;
%     while r(i)<0.1
%         r(i)=rand;
%     end
%     r(i)=r(i)*10;
%     r(i)=fix(r(i))*10;
%     AllDataMem=[AllDataMem;r(i)];
% end
%
while ProcessedData <= DATASize

i=1;
if i<LRUSize+1

    if ProcessedData <= DATASize
```

```
        if lru.size() < LRUSize
            lru.put(AllData(ProcessedData), i);
            mem=[mem AllDataMem(ProcessedData)];
            CPU=[CPU AllDataCPU(ProcessedData)];
            TimeLRU=[TimeLRU TimeLRUTemp(ProcessedData)];
        else

            assert(lru.size() == LRUSize);

            ranber=rand*10;
            if ranber<1
                ranber=rand*10;
            end
            sranber = num2str(fix(ranber));

            lru.put(sranber, LRUSize+1);
            assert(lru.size() == LRUSize);
            lru.remove(AllData(LRUSize));
            lru.put(AllData(ProcessedData), i);
            if DATASize<100
                for lp=1:TimeLRUTemp(ProcessedData)
                    CPU=[CPU AllDataCPU(ProcessedData)];
                    mem=[mem AllDataMem(ProcessedData)];
                end
                TimeLRU=[TimeLRU TimeLRUTemp(ProcessedData)];
            else
                CPU=[CPU AllDataCPU(ProcessedData)];
                mem=[mem AllDataMem(ProcessedData)];
                TimeLRU=[TimeLRU TimeLRUTemp(ProcessedData)];
            end

            i=i-1;
        end
    end
    i=i+1;

    ProcessedData=ProcessedData+1;

end

% mem=[mem lru.memusage];
t=t+1;

end

oldCPU=[];
TimeEndLRU=[];TimeLRU=[];temp=[];
for i=1:DATASize
%     temp1=[];
    if i==1
        TimeLRU=[TimeLRU AllDatanocache(5,1)];
        for lp=1:TimeLRUTemp(i)
            oldCPU=[oldCPU AllDataCPU(i)];
```

```
%              temp1=[temp1 AllDataCPU(i)];
        end
    else
        [ro col]=size(oldCPU);
        TimeLRU=[TimeLRU col];
        if i<DATASize
            if AllDataCPU(i-1)+AllDataCPU(i)<100
                oldCPU=[oldCPU oldCPU(i-1)+AllDataCPU(i)];
%                 temp1=[temp1 CPU(i-1)+AllDataCPU(i)];
            else
                for lp=1:TimeLRUTemp(i)
                    oldCPU=[oldCPU AllDataCPU(i)];
%                     temp1=[temp1 AllDataCPU(i)];
                end
            end
        else
            for lp=1:TimeLRUTemp(i)
                oldCPU=[oldCPU AllDataCPU(i)];
%                 temp1=[temp1 AllDataCPU(i)];
            end
        end
    end
%     temp=temp1;
    [ro col]=size(oldCPU);
    TimeEndLRU=[TimeEndLRU col];
end


% [ro is]=size(TimeLRU);
% for i=1:is
%     if i==1
%         sum=AllDatanocache(5,i);
%     else
%         sum=sum+AllDatanocache(4,i);
%     end
%     TimeLRU(i)=sum;
% end
%
% [ro col]=size(TimeLRU);
% TimeEndLRU=zeros(1,col);
% for i=1:DATASize
%     TimeEndLRU(i)=TimeLRU(i)+AllDatanocache(4,i)-LRUSize;
% end


f=figure(3);
movegui('northwest');
stairs(mem);
title("LRU Memory");
xlabel("Time");
ylabel("MEM");
LRU_Memory=mem;
```

```
f=figure(4);
movegui('southwest');
stairs(CPU)
title("LRU CPU usage");
xlabel("Time");
ylabel("CPU");
LRU_CPU=CPU;

AllDataLRU=[AllDatanocache(1,:);AllDatanocache(2,:);AllDatanocac
he(3,:);TimeLRU;TimeEndLRU];
```

APPENDIX H: MAIN

```
clc;clear;
type = 1;
cpu_count = 1;
series_count = 1;
a = [];
task_orig = [];
cpu = [20 50];
ram = [20 50];
exe = [30 80];
arr_time = [1 10];

no_types = 10;
no_task  = 100;

memory_cash = 4;

a = generate_task(cpu, ram, exe, no_types);
for i=1:series_count
    task_orig(:,:,i)=generate_task_time(a, arr_time, no_task);
end;

for type = 0:1
[ No_hits, No_fails, load1, t_end1 ] = sim_multi(task_orig, cpu_count,
series_count, memory_cash, type );
%[ No_hits, No_fails, load2, t_end2 ] = sim_multi_remote(task_orig,
cpu_count, series_count, memory_cash, type );
%[ No_hits, No_fails, load3, t_end3 ] = sim_multi_remote2(task_orig,
cpu_count, series_count, memory_cash, type );
load2=[0];load3=[0];t_end2=0;t_end3=0;


for i=1:cpu_count
    load1(1:max(t_end),1,i) = 1:max(t_end);
    load2(1:max(t_end),1,i) = 1:max(t_end);
    load3(1:max(t_end),1,i) = 1:max(t_end);
end;
t_end = [t_end1 t_end2 t_end3]
f1=figure('Name','CPU Load');

% for j=1:max(t_end)
%     for i=0:cpu_count-1
%
%         subplot(cpu_count,3,i*3+1)
%         plot (load1(1:max(t_end),1,i+1),load1(1:max(t_end),2,i+1));
%
%         subplot(cpu_count,3,i*3+2)
%         plot (load2(1:max(t_end),1,i+1),load2(1:max(t_end),2,i+1));
%
%         subplot(cpu_count,3,i*3+3)
%         plot (load3(1:max(t_end),1,i+1),load3(1:max(t_end),2,i+1));
%
%         T = timer('TimerFcn',@(~,~)disp(''),'StartDelay',0.001);
%         start(T)
%         wait(T)
%
%     end;
```

```
%
% end
load1(max(t_end)+1,1,1:cpu_count)=max(t_end)+1;
load2(max(t_end)+1,1,1:cpu_count)=max(t_end)+1;
load3(max(t_end)+1,1,1:cpu_count)=max(t_end)+1;

load1(max(t_end)+1,2,1:cpu_count)=0;
load2(max(t_end)+1,2,1:cpu_count)=0;
load3(max(t_end)+1,2,1:cpu_count)=0;

load1(max(t_end)+1,3,1:cpu_count)=0;
load2(max(t_end)+1,3,1:cpu_count)=0;
load3(max(t_end)+1,3,1:cpu_count)=0;

for i=0:cpu_count-1

    subplot(cpu_count,3,i*3+1)
    plot (load1(1:max(t_end),1,i+1),load1(1:max(t_end),2,i+1));
    subplot(cpu_count,3,i*3+2)
    plot (load2(1:max(t_end),1,i+1),load2(1:max(t_end),2,i+1));
    subplot(cpu_count,3,i*3+3)
    plot (load3(1:max(t_end),1,i+1),load3(1:max(t_end),2,i+1));

%     T = timer('TimerFcn',@(~,~)disp(''),'StartDelay',0.001);
%     start(T)
%     wait(T)

end;


for i=0:cpu_count-1
    subplot(cpu_count,3,i*3+1)
    title = strcat('CPU Load without cach 1 C ',string(i+1));
    xlabel('Time step')
    ylabel('Load (%)')

    subplot(cpu_count,3,i*3+2)
    title = strcat('CPU Load without cach 2 C ',string(i+1));
    xlabel('Time step')
    ylabel('Load (%)')

    subplot(cpu_count,3,i*3+3)
    title = strcat('CPU Load without cach 3 C ',string(i+1));
    xlabel('Time step')
    ylabel('Load (%)')
end;

f2=figure('Name','RAM Load');

for i=0:cpu_count-1

    subplot(cpu_count,3,i*3+1)
    plot (load1(1:max(t_end),1,i+1),load1(1:max(t_end),3,i+1));

    subplot(cpu_count,3,i*3+2)
    plot (load2(1:max(t_end),1,i+1),load2(1:max(t_end),3,i+1));

    subplot(cpu_count,3,i*3+3)
    plot (load3(1:max(t_end),1,i+1),load3(1:max(t_end),3,i+1));
```

```
%     T = timer('TimerFcn',@(~,~)disp(''),'StartDelay',0.001);
%     start(T)
%     wait(T)

end;

for i=0:cpu_count-1
    subplot(cpu_count,3,i*3+1)
    title = strcat('RAM Load without cach 1 C ',string(i+1));
    xlabel('Time step')
    ylabel('Load (%)')

    subplot(cpu_count,3,i*3+2)
    title = strcat('RAM Load without cach 2 C ',string(i+1));
    xlabel('Time step')
    ylabel('Load (%)')
    subplot(cpu_count,3,i*3+3)
    title = strcat('RAM Load without cach 3 C ',string(i+1));
    xlabel('Time step')
    ylabel('Load (%)')
end;
end;
```

APPENDIX I: Generate Task Time

```
function [task] = generate_task_time(a, arr_time, no_task)

% generation of task queue
% task is a matrix no_task x 6
% columns: index, arrival time, task type, cpu,ram,exec

t_arr=randi(arr_time,no_task,1);
t_arr2(1)=t_arr(1);
for i=2:no_task
    t_arr2(i)=t_arr2(i-1)+t_arr(i);
end

cdf2=[0 transpose(a(:,3))];
d=rand(no_task,1);
Y = discretize(d,cdf2);
for i=1:no_task
    task(i,1)=i;
    task(i,2)=t_arr2(i);
    task(i,3)=Y(i);
    task(i,4)=a(Y(i),4);
    task(i,5)=a(Y(i),5);
    task(i,6)=a(Y(i),6);
end
end
```

APPENDIX J:MEMORY AVAILABLE

```
unction [hit, No_hits, No_fails] = memory_avilable(task,cash, No_hits, No_fails,
cpu_no)


for j=1:size(cash,1)
   if task == cash(j,cpu_no)

      hit = 1;
      No_hits(cpu_no) = No_hits(cpu_no) +1;
      break;
   else
      hit = 0;

   end;
end;
if hit == 0
No_fails(cpu_no) = No_fails(cpu_no) +1;

end;
```

APPENDIX K: SIMULATION

```
function [No_hits, No_fails, load, t_end] = simulate(tasks, type, memory_cash, cpu_no)
%define max capacity for cpu and ram
c_cpu = 100;
c_ram = 100;
memory = [];

%generate tasks according to parameters in function

% load array contains actual load of cpu and ram
% load matrix has 3 columns: time, cpu_load, ram_load
% max of time: timesteps should be increased if necessary
load=zeros(1,3,1);

if type == 0
   No_hits = 0;
   No_fails = 0;
end;

index = 0;
% total number of taks
no_task=size(tasks,1);

%generate 3 columns for actual start time end time and flag for cached

newcol=zeros(no_task,4);

task=[tasks newcol];

for t=1:task(1,2)
   load(t,1,1:1) = t;
end;

cash = zeros(memory_cash,1);
No_hits = zeros(1);
No_fails = zeros(1);
% execution of tasks starting with the first
for i=1:no_task
   tt=task(i,2);     % tt: task arrival time
   flag = 0;         % flag for task being assigned to a free time slot
   %search for available cpu and ram capacity for the duration of
   %execution time for task

   if tt > size(load,1)
      for t=task(i-1,2):task(i,2)
         load(t,1,1) = t;
      end;
```

```
    end;
    while flag==0
        cpu_cnt = 1;

        [cpu_no, load, task, flag, tt, cash, No_hits, No_fails, index] = ...
            check_CPU_RAM_one(cpu_cnt,  load, i, task, c_cpu, c_ram, flag, tt, cpu_no,
cash, No_hits, No_fails, index, type);

    end  %while

end
% set the time to finish all tasks

t_end=max(task(:,8));


end
```

APPENDIX L: SIM REMOTE

```
function [No_hits, No_fails, load, t_end] = sim_remote(tasks, cpu_count, series_count,
type, memory_cash)
%define max capacity for cpu and ram
c_cpu = 100;
c_ram = 100;
memory = [];
load = [];

%generate tasks according to parameters in function

% load array contains actual load of cpu and ram
% load matrix has 3 columns: time, cpu_load, ram_load
% max of time: timesteps should be increased if necessary
load=zeros(1,3,1,1);
No_hits = zeros(cpu_count);
No_fails = zeros(cpu_count);
index = zeros(cpu_count,1);
% total number of taks
no_task=size(tasks,1);
%generate 3 columns for actual start time end time and flag for cached
newcol=zeros(no_task,4,series_count);
task=[tasks newcol];
for t=1:task(1,2)
   load(t,1,1:cpu_count) = t;
end;

cash = zeros(memory_cash,cpu_count);
No_hits = zeros(cpu_count);
No_fails = zeros(cpu_count);

% execution of tasks starting with the first
for i=1:no_task
   for j=1:series_count
      tt=task(i,2,j);     % tt: task arrival time
      flag = 0;         % flag for task being assigned to a free time slot
      %search for available cpu and ram capacity for the duration of
      %execution time for task
      if tt > size(load,1)
         for t=task(i-1,2,j):task(i,2,j)
            load(t,1,1:cpu_count) = t;
         end;
      end;
      while flag==0

         cpu_no = 0;
         [cpu_no, load, task, flag, tt, cash, No_hits, No_fails, index] ...
            = check_CPU_RAM_remote(cpu_count,  load, i, task, c_cpu, c_ram, flag, tt, cpu_no,
j, cash, No_hits, No_fails, index,type);

      end  %while
```

```
    end
end
for i=1:cpu_count
    t_end(i)=max(task(:,8,i));
end;
end;
```

APPENDEX M: TASK ASSIGNING

```
function [cpu_no, load, task, flag, tt, cash, No_hits, No_fails, index] ...
   = check_CPU_RAM_remote2(cpu_count, load, i, task, c_cpu, c_ram, flag, tt,
cpu_no, series_no, cash, No_hits, No_fails, index,type)

cpu_no = cpu_no+1;
% assign current task values to scalar variables

t_type=task(i,3, series_no); % task type
t_dura=task(i,6, series_no); % task execution time
t_cpu=task(i,4, series_no);  % task cpu load
t_ram=task(i,5, series_no);  % task ram load

if tt > 0
   for t=tt:tt+t_dura
      load(t,1,:) = t;
   end;
end;

c_max(cpu_no)=max(load(tt:tt+t_dura,2,cpu_no)); % max cpu load during the period
% tt and tt+execution time
r_max(cpu_no)=max(load(tt:tt+t_dura,3,cpu_no)); % max ram load during the period
% tt and tt+execution time


flag = 0;
if c_max(cpu_no)+t_cpu<=c_cpu && r_max(cpu_no)+t_ram<=c_ram
   if type > 0
      [hit, No_hits, No_fails] = memory_avilable(t_type,cash, No_hits, No_fails,
cpu_no);

      if hit == 0

         for t=1:tt+t_dura
            load(t,1,cpu_no) = t;
         end;
         % if the cpu and ram have available resource for task
         for j=1:t_dura      % increase cpu and ram load for
            % the period between tt and
            % tt+execution time
            load(tt+j-1,2,cpu_no)=load(tt+j-1,2,cpu_no)+t_cpu;
            load(tt+j-1,3,cpu_no)=load(tt+j-1,3,cpu_no)+t_ram;
         end;
```

APPENDEX N: CHECK CPU REMOTE

```
function [cpu_no, load, task, flag, tt, cash, No_hits, No_fails, index] ...
   = check_CPU_RAM_remote2(cpu_count, load, i, task, c_cpu, c_ram, flag, tt,
cpu_no, series_no, cash, No_hits, No_fails, index,type)

cpu_no = cpu_no+1;
% assign current task values to scalar variables

t_type=task(i,3, series_no); % task type
t_dura=task(i,6, series_no); % task execution time
t_cpu=task(i,4, series_no);  % task cpu load
t_ram=task(i,5, series_no);  % task ram load

if tt > 0
   for t=tt:tt+t_dura
      load(t,1,:) = t;
   end;
end;

c_max(cpu_no)=max(load(tt:tt+t_dura,2,cpu_no)); % max cpu load during the period
% tt and tt+execution time
r_max(cpu_no)=max(load(tt:tt+t_dura,3,cpu_no)); % max ram load during the period
% tt and tt+execution time


flag = 0;
if c_max(cpu_no)+t_cpu<=c_cpu && r_max(cpu_no)+t_ram<=c_ram
   if type > 0
      [hit, No_hits, No_fails] = memory_avilable(t_type,cash, No_hits, No_fails,
cpu_no);

      if hit == 0

         for t=1:tt+t_dura
            load(t,1,cpu_no) = t;
         end;
         % if the cpu and ram have available resource for task
         for j=1:t_dura      % increase cpu and ram load for
            % the period between tt and
            % tt+execution time
            load(tt+j-1,2,cpu_no)=load(tt+j-1,2,cpu_no)+t_cpu;
            load(tt+j-1,3,cpu_no)=load(tt+j-1,3,cpu_no)+t_ram;
         end;
```

REFERENCES

[1] J. Gubbia, R. Buyyab, S. Marusic , M. Palaniswami, Internet of Things (IoT): "A vision, architectural elements, and future directions, Future Generation Computer Systems", vol 29, no. 7, pp. 1645-1660, Sep 2013.

[2] M. A. Burhanuddin, Ali A. Mohammed, R. Ismail and H. Basiron, Internet of Things Architecture: "Current Challenges and Future Direction of Research", International Journal of Applied Engineering Research, vol. 12, no. 21 pp. 11055-11061, 2017.

[3] T. Yu, X. Wang and A. Shami, "Recursive Principal Component Analysis-Based Data Outlier Detection and Sensor Data Aggregation in IoT Systems", in IEEE Internet of Things Journal, vol. 4, no. 6, pp. 2207-2216, Dec. 2017.

[4] A.Akbar, A.Khan, F.Carrez, and K. Moessner , "Predictive Analytics for Complex IoT Data Streams", IEEE Internet of Things Journal, vol.4 , no. 5 pp. 1571 – 1582, Oct. 2017.

[5] B. Farahani, F.Firouzi, V.Chang, M. Badarogl and N. Constant, K. Mankodiya,Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare, Future Generation Computer Systems, vol. 78, no.2 , pp.659-676, Jan 2018.

[6] Y.Cao, T. Jiang, S.Member, and Z.Han, "A Survey of Emerging M2M Systems:Context, Task, and Objective", IEEE Internet of Things Journal, vol.6 no. 3 pp. 1246-1258, Dec 2016.

[7] B.Afzal, M.Umair, G.A.Shah, E.Ahmed , "Enabling IoT platforms for social IoT applications: Vision, feature mapping, and challenges, Future Generation Computer Systems", vol. 92, pp. 718-731, March 2019.

[8] V. G. Cerf and P. T. Kirstein, "Gateways for the Internet of Things: An old problem revisited", 2013 IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, pp. 2641-2647, Dec. 2013.

[9] Tang, Yayuan & Guo, Kehua & Ma, Jian"hua & Shen, Yutong & Chi, Tao, "A smart caching mechanism for mobile multimedia in information centric networking with edge computing", Future Generation Computer Systems,2018.

[10] F.Al-Turjman, “5G-enabled devices and smart-spaces in social-IoT: An overview”, vol. 92, pp. 732-744, March 2019.

[11] Y. Li, A.Orgerie, I. Rodero , B.L. Amersho, M. Parashar, J. Menaud,End-to-end energy models for Edge Cloud-based IoT platforms: “Application to data stream analysis in IoT”, Future Generation Computer Systems, vol. 87, pp. 667-678 , Oct 2018.

[12] S. K. Sharma and X. Wang, "Live Data Analytics With Collaborative Edge and Cloud Processing in Wireless IoT Networks," in IEEE Access, vol. 5, pp. 4621-4635, 2017.

[13] X. Xu, S. Huang, L. Feagan, Y. Chen, Y. Qiu and Y. Wang, “EAaaS: Edge Analytics as a Service,” IEEE International Conference on Web Services (ICWS), Honolulu, HI, pp. 349-356, 2017.

[14] Y.Tang, K.Guo, J.Mac, Y.Shen, T.Chi, “Future Generation Computer Systems” Vol. 91, Pages 590-600, Feb 2019.

[15] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, “Internet of Things for Smart Cities,” in IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22-32, Feb. 2014.

[16] A. Balamash and M. Krunz, “An overview of web caching replacement algorithms,” in IEEE Communications Surveys & Tutorials, vol. 6, no. 2, pp. 44-56, Second Quarter 2004.

[17] A. Muhammed, K. M. Saleh and A. Abdullah, “Optimum packet size of voice packet in the FIFO adversarial queuing model,” Asia-Pacific Conference on Applied Electromagnetics, Melaka, pp. 1-6, 2007.

[18] Cholvi, Vicent & Echagüe, Juan & LeBoudec, “On the Feasible Scenarios at the Output of a FIFO Server”, IEEE Communications Letters, vol .9, no.5, may 2005.

[19] Tanwir, G. Hendrantoro and A. Affandi, “Early result from adaptive combination of LRU, LFU and FIFO to improve cache server performance in telecommunication network,” International Seminar on Intelligent Technology and Its Applications (ISITIA), 2015, pp. 429-432, 2015.

[20] G. Kesidis, “Stationary Distribution of a Generalized LRU-MRU Content Cache,” 2018 International Conference on Computing, Networking and Communications (ICNC), pp. 676-681, 2018.

[21] M.Bilal, S.Kang, “Time Aware Least Recent Used (TLRU) Cache Management Policy in ICN”, Korea Electronics and Telecommunications Research Institute (ETRI), Feb. 2014.

[22] M. M. Hsieh, T. C. Wei and W. V. Loo, “A cached system architecture dedicated for the system IO activity on a CPU board,” IEEE International Conference on Computer Design: VLSI in Computers and Processors, Cambridge, pp. 518-522, 1989.

[23] A. K. Datta and R. Patel, "CPU Scheduling for Power/Energy Management on Multicore Processors Using Cache Miss and Context Switch Data," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 5, pp. 1190-1199, May 2014.

[24] J. Nickolls and W. J. Dally, "The GPU Computing Era," in IEEE Micro, vol. 30, no. 2, pp. 56-69, March-April 2010.

[25] P. Blinzer, "The Heterogeneous System Architecture: It's beyond the GPU," 2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), Agios Konstantinos, 2014, pp. iii-iii

[26] J. Fang, Q. Fan, X. Hao, Y. Cheng and L. Sun, "Performance Optimization by Dynamically Altering Cache Replacement Algorithm in CPU-GPU Heterogeneous Multi-core Architecture," IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, 2017, pp. 723-726, 2017.

[27] M. A. Maddah-Ali and U. Niesen, "Fundamental Limits of Caching," in IEEE Transactions on Information Theory, vol. 60, no. 5, pp. 2856-2867, May 2014.

[28] R. Rajamoni, R. Bhagavathula and R. Pendse, "Timing analysis of block replacement algorithms on disk caches," Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems, Lansing, pp. 408-411 vol.1,2000.

[29] W. Jiang, Y. Deng, X. Meng, C. Hu and Y. Zhou, "Boosting Disk Performance by Compressing On-board Disk Cache," 2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Bangkok, pp. 98-105, 2017.

[30] Y. Liu, J. Huang, C. Xie and Q. Cao, "RAF: A Random Access First Cache Management to Improve SSD-Based Disk Cache," 2010 IEEE Fifth International Conference on Networking, Architecture, and Storage, Macau, , pp. 492-500, 2010.

[31] Y. Niranjan, S. Tiwari and R. Gupta, "Average memory access time reduction in multilevel cache of proxy server," 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad pp. 44-47, 2013.

[32] A. Imtiaz and M. J. Hossain, "Distributed cache management architecture: To reduce the internet traffic by integrating browser and proxy caches," 2014 International Conference on Electrical Engineering and Information & Communication Technology, Dhaka, pp. 1-4, 2014.

[33] hangyao Chen, Xiufen Fu, Shaohua Teng, Peijiang Liu and Baixing Chen, "A cooperative browser-level web caching system based on chord," International Conference on Computer Supported Cooperative Work in Design, Xi'an, 2008, pp. 93-97,2008.

[34] “Cisco visual networking index: Global mobile data traffic forecast update, fg2016–2021,” Cisco, San Jose, CA, USA, White Paper, Feb. 2017.

[35] A.C. Kazez, T.Girici,” Clustering-based device-to-device cache placement, Ad Hoc Networks”, Vol. 84, pp. 170-177, March 2019.

[36] N. Deng and M. Haenggi, "The Benefits of Hybrid Caching in Gauss–Poisson D2D Networks," in IEEE Journal on Selected Areas in Communications, vol. 36, no. 6, pp. 1217-1230, June 2018.

[37] A. Kushwaha and H. R. Sharma, "Designing an Enhanced Selective Encryption Method for Securing Mobile Ad Hoc Network," 2014 International Conference on Computational Intelligence and Communication Networks, Bhopal, pp. 793-798, 2014.

[38] N. Chand, R. C. Joshi and M. Misra, "An efficient caching strategy in mobile ad hoc networks based on clusters," 2006 IFIP International Conference on Wireless and Optical Communications Networks, Bangalore, pp. 5, 2006.

[39] G. Paschos, E. Bastug, I. Land, G. Caire and M. Debbah, "Wireless caching: technical misconceptions and business barriers," in IEEE Communications Magazine, vol. 54, no. 8, pp. 16-22, August 2016.

[40] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher and B. Ohlman, "A survey of information-centric networking," in IEEE Communications Magazine, vol. 50, no. 7, pp. 26-36, July 2012.

[41] N. Deng and M. Haenggi, "The Benefits of Hybrid Caching in Gauss–Poisson D2D Networks," in IEEE Journal on Selected Areas in Communications, vol. 36, no. 6, pp. 1217-1230, June 2018.

[42] C. Ho, C. Ho and C. Tseng, "A case study of cache performance in ICN — Various combinations of transmission behavior and cache replacement mechanism," 2015 17th International Conference on Advanced Communication Technology (ICACT), pp. 323-328,2015.

[43] S. Arshad, M. A. Azam, M. H. Rehmani and J. Loo, "Recent Advances in Information-Centric Networking-Based Internet of Things (ICN-IoT)," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2128-2158, April 2019.

[44] I.Psaras, O. Ascigil, and S. Rene, G. Pavlou, L. Zhang," Mobile Data Repositories at the Edge", Workshop on Hot Topics in Edge Computing (HotEdge 18), jul 2018.

[45] S. Li, Y. Tao, X. Qin, L. Liu, Z. Zhang and P. Zhang, "Energy-Aware Mobile Edge Computation Offloading for IoT Over Heterogenous Networks," in IEEE Access, vol. 7, pp. 13092-13105, 2019.

[46] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," in IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322-2358, Fourthquarter 2017.

[47] R.A. Cherrueau, A. Lebre, and D. Pertin, and F. Wuhib and J. M. Soares, "Edge Computing Resource Management System: A Critical Building Block! Initiating the debate via Open Stack", Workshop on Hot Topics in Edge Computing (Hot Edge 18), jug 2018.

[48] S. Zang, W. Bao, and P.L. Yeoh, "Paying Less for More? Combo Plans for Edge-Computing Services", Workshop on Hot Topics in Edge Computing, jug 2018.

[49] C. Mehteroglu, Y. Durmus and E. Onur, "Semantic edge caching and prefetching in 5G," 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2017, pp. 692-695.

[50] J.G. Andrews, S. Buzzi, and W. Choi, and S. Hanly, A. Lozano, A.C.K. Soong, Zhang, "What Will 5G Be?", Selected Areas in Communications, IEEE Journal on, Vol. 32, no. 6, pp. 1065-1082, July 2014

[51] X. Wang, M. Chen, T. Taleb, A. Ksentini and V. C. M. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," in IEEE Communications Magazine, vol. 52, no. 2, pp. 131-139, Feb 2014.

[52] B. Li, L. Rui, X. Qiu and H. Huang, "Content Caching Strategy for Edge and Cloud Cooperation Computing," 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 2019, pp. 260-265.

[53] I. Yaqoob et al., "Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges," in IEEE Wireless Communications, vol. 24, no. 3, pp. 10-16, June 2017.

[54] Y. Tang, K. Guo, J. Mac, and Y. Shen, T. Chi, “A smart caching mechanism for mobile multimedia in information centric networking with edge computing”, Future Generation Computer Systems, pp. 590-600 vol. 91, Feb.2019.

[55] S. Jalali, "M2M solutions — Design challenges and considerations," 2013 IEEE Recent Advances in Intelligent Computational Systems (RAICS), Trivandrum, 2013, pp. 210-214.

[56] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil and K. Drira, "Cache coherence in Machine-to-Machine Information Centric Networks," 2015 IEEE 40th Conference on Local Computer Networks (LCN), Clearwater Beach, FL, 2015, pp. 430-433.

[57] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of Things for Smart Cities," in IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22-32, Feb. 2014.

[58] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung and H. Yin, "Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach," in IEEE Communications Magazine, vol. 55, no. 12, pp. 31-37, Dec. 2017.

[59] Y. U. Devi and M. S. S. Rukmini, "IoT in connected vehicles: Challenges and issues — A review," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, 2016, pp. 1864-1867.

[60] B. Ram, N. Chauhan, N. Chand and L. K. Awasthi, "A new mechanism to query latency minimization for cache invalidation in vehicular ad hoc networks," 2011 3rd International Conference on Electronics Computer Technology, Kanyakumari, 2011, pp. 250-253.

[61] F. Wu, T. Wu and M. R. Yuce, "Design and Implementation of a Wearable Sensor Network System for IoT-Connected Safety and Health Applications," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 87-90.

[62] K. Monteiro, É. Rocha, É. Silva, G. L. Santos, W. Santos and P. T. Endo, "Developing an e-Health System Based on IoT, Fog and Cloud Computing," 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), Zurich, 2018, pp. 17-18.

[63] R. K. Pathinarupothi, P. Durga and E. S. Rangan, "IoT-Based Smart Edge for Global Health: Remote Monitoring with Severity Detection and Alerts Transmission," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2449-2462, April 2019.