UNIVERSITY OF TARTU

Institute of Computer Science

Computer Science Curriculum

Herman Meier

Simulating energy efficient fog computing

Bachelor's Thesis (9 ECTS)

Supervisor: Satish Narayana Srirama

Tartu 2019

Simulating energy efficient fog computing

Abstract:

With increasing demand on computing resources, there is a need to reduce energy consumption in order to keep computer systems sustainable. Current cloud and fog computing architectures need to be improved by designing energy efficient scheduling and placement algorithms. This thesis describes power efficiency in fog computing and cloud computing. It shows a way to minimize power usage by designing scheduling and placement algorithms that maximize the number of idle hosts. Algorithms are designed to archive that goal in cloud and fog systems. The algorithms are tested in different simulation scenarios. The results are compared and analysed. The thesis also contains a brief overview of similar research that has been done on this topic.

Keywords:

Cloud computing, algorithms, distributed systems

CERCS: P175 Informatics, systems theory

Energiasäästliku uduandmetöötluse simuleerimine

Lühikokkuvõte:

Nõudlus arvuti ressursside järele üha suureneb ning seega on vajadus vähendada energia kulu, et tagada arvutisüsteemide jätkusuutlikus. Praegused pilve- ja uduandmetöötlus arhitektuuride edasiarendamiseks on vaja ajajaotus- ja asetusalgoritme, mis arvestavad energia kuluga. Selles töös kirjeldatakse energiasäästlikust pilve- ja uduandmetöötluses. Töös luuakse ajajaotus- ja asetusalgoritmid, mis maksimeerivad vabade seadmete arvu ning vähendavad seeläbi süsteemi energiakulu. Algoritme katsetatakse erinevates simulatsioonides. Simulatsioonide tulemusi analüüsitakse ja võrreldakse ning tehakse järeldused algoritmide kasulikkusest. Töö sisaldab ka lühikest ülevaadet sarnastest uurimustest.

Võtmesõnad:

Pilveandmetöötlus, algoritmid, hajustöötlus

CERCS: P175 Informaatika, süsteemiteooria

Table of Contents

1. Introduction

Cloud computing has proven to be a widely useful approach to utilize large data centres over the internet. It is scalable and can be beneficial in different applications. Cloud computing utilization is expected to grow as organizations move even more of their services to the cloud, as was concluded in a survey [1] by IDG. One of the downsides of using the cloud is latency. Since the cloud is accessible using the internet, there is a delay between generating the data and processing it. This is especially critical in real time applications. The latency problem is even worse when the network is congested due to a large amount data being transferred.

Fog computing is an emerging companion to cloud computing which promises to eliminate some of the shortcomings of the cloud. It is an architecture, designed by CISCO, which connects the cloud with edge devices [2]. Edge devices are the computing devices that are closest to the source or consumer of data. Fog computing thus reduces network latency and makes real time solutions more viable. The main use cases and the reason fog computing was proposed is the rise of Internet of Thing (IoT). IoT applications can have many devices constantly producing data, which is not what cloud computing was designed for [3]. Fog computing has been shown to be better at handling a large number of connected devices that require real time services [4].

However, there are problems with fog computing that have to be researched more. One of the most critical of these is sustainability. Information and communications technologies (ICT) are currently consuming close to 10% of global electricity and growing number of internet-connected devices might increase it further according to an article [5] on energy consumption in ICT. Current cloud data centres are consuming a major amount of energy due to a large demand for high performance cloud networks. According to a 2008 report [6] an average cloud data centres consumes as much power as 25000 households. The high power cost of data centres is caused by many factors and one of them is idle power usage. The idle power consumption of modern server hardware is a significant percentage of the maximum power consumption under load [7, 8]. So one way to make data centres more energy efficient is to turn off idle hosts or put them to sleep mode. This can be archived by offloading some of the computation from the cloud, which is what fog computing is designed to do.

A survey [9] published by cloud computing researchers calls sustainability “the greatest challenge of our century”. Therefore, with the emergence of fog computing there is a need for more research into energy efficient algorithms to make fog computing networks more sustainable.

1.1 Purpose

The purpose of this thesis is to create an algorithm that helps reduce energy consumption in fog computing, while maintaining its advantages over the cloud. The algorithms proposed in this thesis aim to free up more hosts in order to save energy. These scheduling algorithms try to maximise the number of idle hosts, so that the energy savings from turning off idle hosts is maximised. The proposed algorithms will be tested with simulations and compared to cloud-only solutions.

1.2 Outline

The thesis begins with a brief background on cloud and fog computing as well as IoT. The second chapter also discusses related works. A mathematical model of energy consumption and fog computing is presented in the third chapter. The fourth chapter describes the proposed algorithms and simulations. The final chapter contains an analysis of the results.

2. State of the Art

2.1 Background

There has been an increase in the number of internet connected devices and CISCO estimates that this number will reach 50 billion by 2020 [10]. These smart devices also known as edge devices are collectively called IoT. IoT is an idea that different devices have embedded computers in them and they communicate over the internet. This can enable more automation and improve quality of life. The devices can be sensors or actuator and so can both produce and consume data. In order to create applications that utilise many devices over the internet there was a need for a central platform that facilitates processing, storage and communication for these applications. The most common solution is to use the cloud.

Cloud computing is a technology that helps to provide computing services over the internet. National Institute of Standards and Technology defines the cloud as "a pool of configurable computing resources" [11]. Cloud computing is currently widely used and is critically important for many businesses. A report [1] released in 2016 by IDG found that 70% of organisations have at least one application deployed in the cloud and 56% are looking for more ways to use cloud hosting. The cloud has several characteristics that make it so usable. For example, it offers on demand services, scalability, lower infrastructure costs and different service models. Although cloud computing has proven to be useful for many applications, it is not a solution to every problem. One issue related to cloud computing is data management. The global data creation rate is growing, IDG reported [12] in 2011 that there was over 1ZB of data generated mostly by devices at the edge of the network and this number is growing as more devices are connected to the internet.

The amount of data created by IoT puts more demand on data processing and communication. Cloud computing can offer the processing power needed, but it can cause network related issues, especially when real time solutions are needed. A survey [13] on IoT states that cloud computing is not the best choice for IoT applications and claims that fog computing can be the optimal choice for IoT. The survey brings out that fog computing could reduce network latency and traffic, provide better scalability and mobility as well as improving the performance of real time applications.

Fog computing is a computer network architecture created by CISCO in 2012 [2]. It unites the edge devices and the network with the cloud and creates a broad network of devices. The main idea behind fog computing is to address the shortcomings of the cloud namely

latency and response time. These issues are caused by the long distance between the data source or consumer and the cloud, which leads to more time spent on communicating the data. The fog architecture solves this by utilizing devices near the data source or consumer called edge devices. In addition to using edge devices, the fog can also take advantage of devices between the edge and the cloud. These can be gateways, switches, access points and base stations. In the fog architecture devices like these are referred to as fog nodes. An example of a three layer fog network can be seen on figure 1. Some of the computation can be done on fog nodes so that there is less need to send data to the cloud. Using all of the devices in the network makes fog architecture more flexible and dense, which in turn increases the quality of service (QoS) and accessibility of the system.

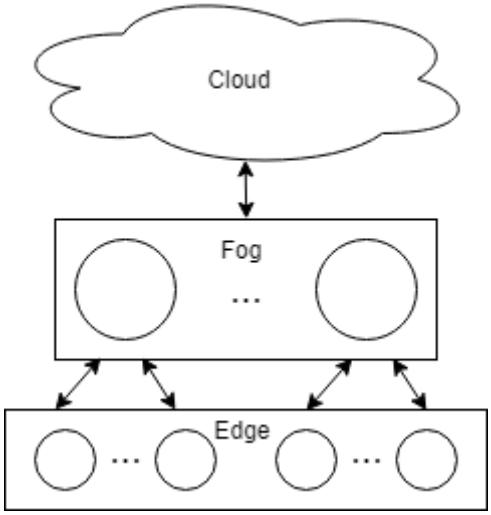

Figure 1. Fog computing network example

Fog computing architecture is advocated and standardised by OpenFog Consortium [14] founded in 2015. They state that their goal is “to create an open reference architecture for fog computing, build operational models and testbeds, define and advance technology, educate the market and promote business development through a thriving OpenFog ecosystem”. The consortium currently has 57 members including industry leaders such as Intel, ARM, CISCO, Microsoft and Dell.

The use cases of fog computing are applications where real time responses are important. Specific scenarios where fog computing will improve the performance of the system have been suggested by different researches [15, 16, 17, 18]. These include big data analytics, smart grids, content delivery networks, emergency response and autonomous vehicles. Most of the use cases for fog computing are IoT applications. In fact fog computing was originally

proposed as a solution to new set of problems arising from IoT applications [17]. The amount of data created by sensors connected to the internet is overwhelming for cloud platforms when real time response is needed. Network congestion would increase latency and response time too much. Utilizing devices near the sensors producing the data is the solution that fog computing provides for IoT applications. Offloading some of the computation to edge or fog devices reduces the need to send data to the cloud for processing and thus reducing the network latency and response time. However, there is a limit on how much can be done in the network edge since the devices there are much less powerful than servers in the cloud are. Having too much load on these devices could lead to processing latency and resource shortage. Therefore, there is a need for new scheduling, placement and migration algorithms in order to effectively utilise all the different layers of fog computing architecture

A survey [15] published in 2018 looks into the architecture and algorithms that make up the current fog systems. They look at both application agnostic and application specific architectures and evaluate them based on heterogeneity, QoS management, scalability, mobility, federation and interoperability. They also analyse different algorithms used in fog systems for scheduling resources. They conclude that fog systems will reduce latency compared to cloud systems if the application is deployed correctly and appropriate scheduling algorithms are used. The survey shows that energy consumption is generally better in fog systems, unless the network energy consumption is very large. These results show the importance of scheduling and placement algorithms in the fog. New scheduling algorithms are needed in order to reduce energy consumption while also maintaining the advantage of low latency in the fog.

In both cloud and fog computing architecture the division of resources is done by a scheduler. Resources can be CPU time, memory, storage or network bandwidth. The algorithm that decides how the resources will be divided is called a scheduling algorithm. Different scheduling algorithms are good for different goals. The choice of an algorithm thus depends on the requirements imposed on the system and they can change in time.

The performance of cloud and fog computing networks depend heavily on the scheduling algorithm used. The parameters of the network can be optimised by using different scheduling algorithms. A 2016 survey [19] of papers about scheduling algorithms in the cloud mapped out 13 different types of resource scheduling algorithms that have been researched

in the analysed 110 papers. The paper found that the most researched algorithms are based on energy consumption and QoS.

There are many different software tools for simulating cloud computing. A survey [20] compares them and concludes that there are differences and a choice should be made based on the user requirements. There are fewer options for fog computing simulations. One of the tools is iFogSim [21] wich is an extensions of cloud computing simulation tool CloudSim [22]. Since it is an extension to CloudSim it is also written in java. It allows the creation of custom applications, topologies and algorithms. There are also examples provided for all of these.

2.2 Related works

The authors of article [23] created an algorithm to efficiently schedule and migrate virtual machines (VMs) in a cloud data centre. The solution they proposed uses a model that has homogenous hosts and three tiers of VMs: small, medium and large. Their algorithm consists of two modules. The first one is a scheduler that divides the requested VMs between hosts in a way that maximizes the number of idle hosts. The other module is responsible for migrating VMs if some host can be made idle by the migration. They also allow to choose the maximum power draw of the hosts and guarantee that all of the VMs that the client request will be hosted.

The authors of [24] show that the dynamic migration of VM can lead to energy savings in a cloud data centre. They created an algorithm that selects VMs to migrate and then finds the hosts that can receive the migrating VMs. The selection of VMs is based on the resource utilization of the host. The other part of the algorithm is a heuristic solution to the bin-packing problem. They compare their algorithm with other migration policies and found that the algorithm they used was more energy efficient than the others.

The article [25] describes an energy-aware heuristic algorithm that dynamically migrates VMs between hosts. The algorithm tries to maximize hosts utilization so that more hosts can be turned off to save energy. The proposed solution is event based and reallocates VMs each time a new workload arrives or ends. Both this and the work by Anton Beloglazov and Rajkumar Buyya [24] show that migration of VMs is an effective measure to reduce energy consumption by a significant percentage.

In the work [26] Mahmud et al create a quality of experience aware placement algorithm for fog computing applications. They create a fog network that has four layers. The first layer

are IoT devices that only generate data. The second layer are gateway devices that connect the IoT devices. The next layer devices are fog nodes that perform computation and the last layer is the cloud. This is similar to the model created in this thesis with the exception of the middle two layer being treated as one in this model.

In article [27] an energy efficient scheduling algorithm for fog architecture is proposed that tries to approximate an optimal solution based on their model. The model they use consists of two types of hosts, one with faster hardware as in a cloud host and one with less computational power as in an edge host. They do not allow migration between hosts. The algorithm optimizes the total execution time and the number of working fast hosts. It then chooses which ever gave the most efficient result. Their model of energy consumption was an inspiration for the models described in this thesis.

In their paper [28], Deng et al discuss power consumption and delay trade off in fog computing. They create a model with a fog and a cloud layer, which are connected over a WAN. It is different from the fog computing model that is used in this thesis, where the model contains three layers instead of two. The article models the flow of requests from the users through the system and defines the power consumption and delay of the whole system. It then describes how to optimize the energy consumption and delay under the constraints proposed in the model. The optimization is done in three steps. First they optimize power consumption and delay in fog computing then the energy consumption in cloud computing and finally the WAN delay. After finding optimal parameters for these sub-problems, they analyse the effect of them on the overall system. The results they get show that fog computing helps to reduce latency which is the same conclusion that this thesis arrives to. However opposite to the results in this thesis, their scenarios show an increase in power consumption when offloading work to the fog layer.

Fog computing architecture is modelled in article [29]. The work describes the layers of fog computing architecture and how they interact with each other. The layered model of the fog computing architecture is very similar to the one used in this thesis as it was used as an example. Both models have three tiers of devices: cloud, gateway and edge. The energy consumption of a traditional cloud is compared with the proposed fog architecture. The papers show that using a fog architecture will decrease energy used in transmitting data over the network when the number of edge devices is sufficiently large. They also show that the energy used for processing in the fog architecture is significantly smaller than in the cloud

due to the fact that most of the processing is done at below the cloud layer. These results align with the results gathered from the simulations done in this thesis.

In conclusion, cloud computing has its weaknesses that are a problem when hosting IoT applications. Fog computing was proposed to ensure better performance when dealing with those applications. Energy efficiency in fog computing has been identified as an issue and there is an interest for research aimed at it. Scheduling and placement algorithms that prioritise energy consumption are therefore needed in order to make fog computing more sustainable.

There have been some studies on energy efficient scheduling and placement in fog computing. These works focus on modelling the fog architecture which is done differently by different authors. The results show that energy and latency savings can be archived by using fog computing with appropriate scheduling and placement algorithms.

3. Modelling

First, a mathematical model is created to describe energy consumption. It will be expressed as a function that depends on time and power usage. Secondly the network of the fog computing architecture will be described and modelled.

3.1 Energy consumption

The total energy consumption of a data centre can be expressed as a sum of the energy consumption of all of the hosts in the data centre.

$$E = \sum_{\forall h \in H} E_h$$

Where E is the total energy consumption, H is a set of all hosts, and E_h the energy consumption of host h.

The energy consumption of a single host is the power usage of the host times time and can be represented as

$$E_h = E_i + E_u = P_i * T_i + P_u * T_u$$

Where $E_i$is the energy consumption while idle, $E_u$is the energy consumption while utilized, P_i the power consumption while idle, T_i is the total idle time, P_u is the power consumption while utilized and T_u is the total time utilized. The whole period that the host is active is

$$T = T_u + T_i$$

The power usage of a host is dependent on the utilization of its resources. The power consumption is thus a function of utilization and can be expressed as

$$P(u)$$

Where P(u) is the power consumption at utilization u. The power consumption function can be different for hosts. A higher resource utilization will result in a higher power consumption. The power consumption is lowest when the host is idle and this value is constant for a host. This means that

$$\forall u_1, u_2 (u_1 > u_2 \rightarrow P(u_1) > P(u_2))$$

$$P(u) > P_i \; \forall u : u > \varepsilon$$

where ε is the utilization while idle.

Utilization is a percent value that is always greater than 0, because even when the host is idle there is still some resource utilization . The utilization of a host changes over the course of a timeframe so the utilization is a function of time and can be expressed as

$$U(t)$$

$$\varepsilon < u < 1 \; \forall u$$

Where U(t) is the utilization at time t. An additional constraint to utilization is added as an upper bound to utilization. This value can be set by the data centre administrator and is expressed here as α It can be useful to better ensure SLA requirements and to avoid very high utilization of hosts. So now the above statement becomes

$$\varepsilon < u < \alpha \; \forall u, \alpha \in (0, 1)$$

The energy consumption of a host over its lifetime can then be modelled as

$$E_h = \int_T P(U(t))dt$$

The power consumption while idle is a constant and can be viewed separately. Since the power consumption while utilized is always greater that the power consumption while idle, the above equation becomes

$$E_h = \int_{T_u} P(U(t))dt + P_i * T_i$$

In order to minimize the energy consumption of a network the idle time T_i can be maximized. This will mean that the host will stay in its lowest power consumption state for the longest time possible.

3.2 Fog network

The fog network described in this thesis will consist of three layers: edge layer, fog layer and cloud layer. The edge devices are grouped into sub networks that are connected with a single fog layer device. All of the fog layer devices will be connected to the cloud layer.

Edge layer devices represent devices that are the least powerful and energy consuming of all the devices in the network. These are the devices that are connected to or contain the sensors that are the source of the data that the network will have to process. The actuators that consume data are also connected or a part of the edge devices. The fog layer represents

intermediary network devices that route data between edge devices and the cloud layer. In this fog architecture, the fog devices can also do some processing. The cloud layer is a homogenous network of powerful hosts. It has considerably more computation and storage capabilities than the other layers, but also consumes the most energy.

The network has latency between different hosts. An assumption that the network latency between edge devices, the fog and the cloud are constant in time. Each fog device can have a different latency to the cloud and each edge device can have a different latency to its fog device.

The set of fog layer devices is denoted by F. Each device from that set has a set of edge layer devices connected to it. For a device f from the set F the connected edge devices are represented by D^f.

The latency from a fog device f to the cloud is denoted as γ_f. The latency from an edge device d to its fog device f is denoted as δ_d^f. So, the latency from an edge device to the cloud is the sum of these two values and can be expressed as

$$\theta_d = \delta_d^f + \gamma_f$$

The whole network is a undirected tree where the devices are the nodes and the edges are the connections between them. Each edge has a weight that is the latency between the connected nodes. The graph is denoted as G.

3.3 Application

An application is a set of modules that send data to each other. Modules that only create data are sensors and modules that only consume data are actuators. Other modules are called intermediary modules. The data that modules exchange can be thought of as a job, there is some computation that has to be performed with the data to get a result. Not all modules change data and the direction of the data flow can be unidirectional or bidirectional.

Based on this the application can be modelled as a directed weighted graph. The nodes of the graph are the modules. The edge direction shows the data flow and the weight shows the amount of data sent. The applications described here will have one sensor module and one actuator module. Let R represent the application graph, S the sensor module and A the actuator module. Intermediary nodes form a subgraph R' that is the graph R without nodes S and A.

3.4 Idle time

The number of idle hosts and the idle time of utilized host must be maximised to decrease energy consumption. The utilized time T_u can be expressed as

$$T_u = T_n - T_p$$

where T_n is the time spent on network communication and T_p is the processing time. The processing time depends on the computational capabilities of a host and it thus decreases from leaves to root in the tree G. The network time depends on the location of hosts and the amount of hops the data does in the application graph.

The idle time can be written as

$$T_i = T - T_u$$

and T_u can be replaced to get

$$T_i = T - T_n - T_p$$

In order to maximise the value of T_i, the values of T_n and T_p can be minimised. This means that minimising latency and processing time will increase the idle time and thus decrease the energy consumption.

This chapter showed how energy consumption can be modelled in cloud or fog computing. The energy consumption is based on the utilisation over time. In order to maximise idle time, processing and network delay have to be minimised, as is shown in the model. The application and fog architecture models described are used to create the algorithms for energy efficient placement.

4. Algorithms and simulations

In this chapter, the algorithms used in this thesis are described. The algorithms for cloud computing include the scheduling and migration. The algorithm for fog computing deal with only placement.

4.1 Scheduling and migration algorithms in the cloud

One way to reduce the energy consumption of a data centre is to maximize the number of idle hosts so that they can be but into sleep mode. This will reduce the hosts power usage and thus the total energy consumption of the data centre. A scheduling algorithm is needed to archive a maximum number of idle hosts. This algorithm prioritizes already utilized hosts over idle hosts when a new job arrives.

To further reduce power consumption a dynamic migration algorithm is used. It migrates existing VM to other host if that makes the original host idle. This will be done each time an existing VM is not needed anymore. If a VM is removed then there will be a chance that one host can be made idle. This can occur when the VM was the only one running on a host or when there exists a host that can be freed if all of its VMs are migrated to the freed up host. In the first case, there is no migration necessary since the host will become idle by itself. The second case means that all of the working hosts will be checked to find a host that can fit its VMs in the new free space. If such a host is found then the migration will be done.

Both the scheduling and migration algorithms will have to compare hosts to make an optimal choice. Choosing between hosts is based on the current utilization of the hosts. If the addition of a new VM would mean that the utilization of the hosts exceeds a set upper bound then the host is not suitable. Already idle hosts should be used only when the requested VM cannot fit any of the already utilized hosts.

This scheduling will have an effect on quality of service. If a host is already utilized then adding another job will mean that the resources of the host will have to be shared between all of the running jobs. This can lead to a quality of service drop when the combined resources needed for the jobs are greater than the ones available to the host. So when designing a scheduler this has to be taken into account. The algorithm should be able to evaluate if the extra utilization from a new VM on a host would risk breaking SLAs.

The scheduler will deal with the initial request from the client. It will divide the requested VMs between the hosts while leaving as many hosts idle as possible. The algorithm will take a list of all host and for each of them find the current utilization. Then the list is sorted in decreasing utilization order. When a new job is scheduled for execution, the list of hosts will be iterated from the beginning. The first host that has utilization less than a set upper bound will be chosen.

1) Sort hosts in decreasing utilization order
2) For each host
    a) If utilization < α
        i) Add VM to host and finish

The algorithm for power efficient migration will be triggered when a VM can be deleted. If a VM is not needed anymore and the host will not be idle then a list of all hosts except the freed host is created and sorted based on utilization. The sorting will be in ascending order. The list will be iterated from the start. If the VMs in the current host could fit the freed host then the migration is done and the iteration is stopped. If no migration can be done then the algorithm just finishes.

1) Sort hosts in ascending utilization order
2) For each host
    a) For VM in host
        i) If VM -> host => utilization < α
            (1) Migrate VM to host and finish

The proposed scheduler relies on keeping a sorted set of hosts in memory. When a new VM is requested the most utilized host that has enough resources left will be chosen. This ensures that the number of idle hosts is maximised. The migration algorithm provides additional benefits to energy consumption. It checks if a host can be freed up when an existing VM is no longer needed. A host can be freed up is all of its VMs can be migrated to the host that the VM was removed from. This is done starting from the least utilised host.

4.2 Scheduling in fog computing

To create the scheduler for a fog network an additional constraint will have to be introduced. The scheduler has to keep the processing as low on the network as possible in order to reduce latency. Therefore, the choice of a host will consist of the utilization of the host and the hosts location in the network.

The utilization time T_u of a host consists of network latency T_n and processing time T_p. In order to maximize the idle time T_i the utilization time should be minimal. This means that when choosing a host both the network latency and the processing time will have to be taken into account. The best choice of a host will have the lowest utilization time.

The application subgraph R' can be separated into tiers based on the distance from the sensor or the actuator. Let the set of tiers be K. The set K contains tiers that are based on the distance from the data source or consumer. All modules that are two connection away from the sensor or actuator belong in the second tier and so on. The tiers in K^s will be denoted with an index, for example, K_2^s is the second tier containing all modules that are two connections away from the sensor. The modules that are one connection away from the sensor or actuator will belong in K_1^s and K_1^a.

All of the hosts in the fog network will be added to set H. The set will be sorted by layers so that the edge devices will be first and the cloud hosts last.

First all modules in K_1^s and K_1^a are placed on the edge devices. If all of the edge devices are full then the next layer devices will be used. This is a bin packing problem with a constraint that the buckets have to be filled in order. This ensures that the modules, which communicate with sensors or actuators, are placed on a device that is directly connected to them.

After all first tier modules are placed as low as possible the next tier modules will be placed in the network starting from the lowest until all of the modules are placed. For each list of modules in current tier K_i a combination of all possible ways to place them in the network is collected. A module can be placed on a host if it has enough resources available. This will result in a set of module to host mappings. It would be better to find all possible combinations to place all modules in $K_i$but it would take too long as the number of modules to place grows. The set of combinations will be sorted by ascending unique hosts count. This will reduce the amount of steps necessary later in the algorithm.

The best of these combinations will be the actual placement used. The combinations will be compared based on how tightly they packs the modules and how much will they increase the total latency. A combination K_j is more tightly packed if K_j has less elements than K_i. That means it has fewer unique hosts. This results in more hosts that can be turned off if this placement is used. Since the set of combinations is sorted the algorithms can stop if it has found a valid placement and the next one is more tightly packed. By definition of K_i, every

module in that set has to be connected to at least one module from K_{i-1}. For each module in K_{i-1} minimum latency to a connected module in K_i is calculated. The current combination is discarded if the minimum latency is over a set limit for any module in K_{i-1}. The total latency for the current placement is the sum of all these minimum latencies. This leads to lower average T_n. The best placement will be the one that packs the modules tightly while keeping the network latency low. This will free up hosts to put into sleep mode and increase the idle time of utilized hosts by reducing the network latency.

The proposed placement algorithm for fog computing will reduce energy consumption by utilising the edge and gateway nodes. Offloading work to these devices will free up hosts in the cloud while also reducing latency. The algorithms places modules based on the utilisation and location of the hosts. The lower tier modules are placed closer to the edge and other modules are placed in order of distance from the edge.

4.3 Simulations

For fog computing simulations iFogSim [21] was used. It is an extension of CloudSim [22] written in Java by other researchers to enable easier fog modelling. The software supports creation of a network of different computing devices. These can be edge devices, gateways or the cloud. It also enables to simulate workloads that have discrete modules with data dependencies between them. In order to implement the proposed scheduling algorithm for energy efficient fog computing a new module placement class was written that controls the placement of the modules of the simulated application. The physical topology used in these simulations is similar to the theoretical model described in chapter 3. It is made up of three homogenous layers.

Two applications were simulated to test the algorithm. The first scenario is a smart camera system. The model of the application is shown in figure 2, it consists of motion detection, object detection and motion tracking modules. The input to the system is a camera and the output are actuators that move it to enable object tracking. The system also has a UI module. The camera system is connected so that there is no cycles in the application graph. This simulation scenario is one of the examples provided in iFogSim.

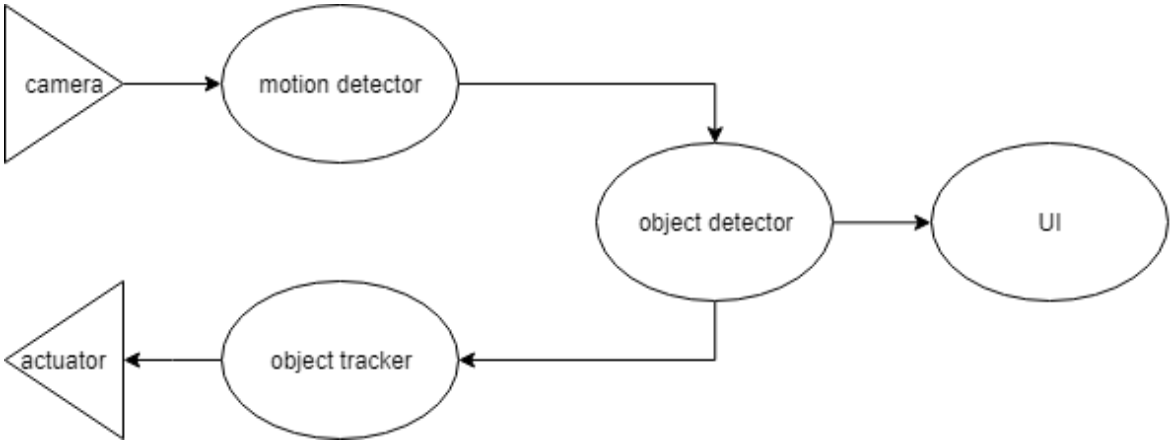

Figure 2. Camera system application model

The second application is a health monitoring system. The application is described in a book about fog computing [30]. The application has four modules: client module, data filtering module, data processing module and event handler module. The model is shown in figure 3. The application graph forms a cycle which makes it different from the camera system scenario. This scenario was implemented in iFogSim since it is not provided in the source code. The simulation was written based on the guidelines in the book where it was taken from.

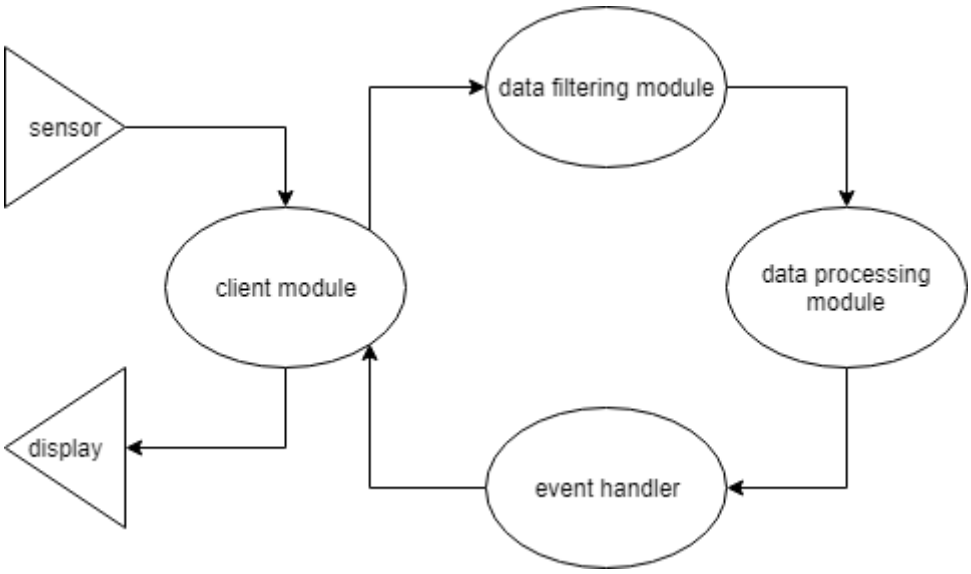

Figure 3. Health monitoring system model

Both of the simulations were run using the proposed algorithm and a scheduling that places every module in the cloud. The simulations were tested with different number of application modules and different number of devices in the fog network. The total energy consumption

as well as the individual energy consumption of each device was gathered. The results of the simulations are described in chapter 5.

The algorithms for cloud and fog computing proposed here aim to lower the energy consumption by maximising idle time. The scheduling algorithm for the cloud tries to keep a maximum number of hosts idle. The proposed fog placement algorithm tries to place modules so that the number of utilised devices is low while also keeping the latency between the modules at minimum. The algorithm is tested in two scenarios using iFogSim.

5. Results and analysis

This chapter describes each scenario that was tested and displays its results. Scenarios of the same application are compared and analysed. Conclusions about the proposed algorithms are presented at the end based on all the scenarios.

5.1 Results of the simulations

Both the camera system and the health monitoring system simulations were done with two scenarios. The differences were in the number of devices and number of modules. The details of all scenarios can be seen in table1.

Table 1. Application and topology details

Application	Scenario	Gateway devices	Edge devices	Motion detector modules	Object detector modules	Object tracker modules	UI modules
Camera system	1	2	8	8	2	1	1
	2	4	12	12	3	3	1
Application	Scenario	Gateway devices	Edge devices	Client modules	Filtering modules	Processing modules	Event handler modules
Health monitoring system	3	4	4	4	4	4	4
	4	4	11	11	3	3	3

The resulted energy consumption for each layer and for the whole system are shown in figure 4 and figure 5. All scenarios show that the total energy consumption is lower with the proposed algorithm. The edge layer energy consumption is higher in scenarios 2 and 4 because there were more edge devices in use. The number of gateway devices was equal in both health monitoring system simulations, but different in the camera system simulations. This is also visible on the chart.

Both the edge and gateway layer had higher energy consumption in all scenarios when using the proposed algorithm. This is expected since when all the modules are deployed in the cloud, the other devices will not be utilized. The cloud layer however shows a decrease in energy consumption when using the proposed algorithm.

The average latency between different modules is shown in table 2. The latency was measured for three connections: sensor to module, module to module, module to actuator. The measurements show that the proposed algorithm reduces latency for the devices that connect to sensors and actuators. At the same time the latency between modules is higher than it is for cloud only placement. The scenarios with the same application and algorithm have similar latencies. The latency between devices is defined in iFogSim.

Table 2. Latency in simulation scenarios

Application	Scenario	Algorithm	sensor to module (ms)	module to module (ms)	module to actuator (ms)
Camera system	1	cloud only	5.24	0.31	5.12
		proposed	1.00	2.78	3.11
	2	cloud only	5.29	0.34	5.16
		proposed	1.00	2.77	3.11
Health monitoring system	3	cloud only	10.04	0.84	5.14
		proposed	6.00	5.00	1.10
	4	cloud only	10.06	0.99	5.20
		proposed	6.00	5.37	1.09

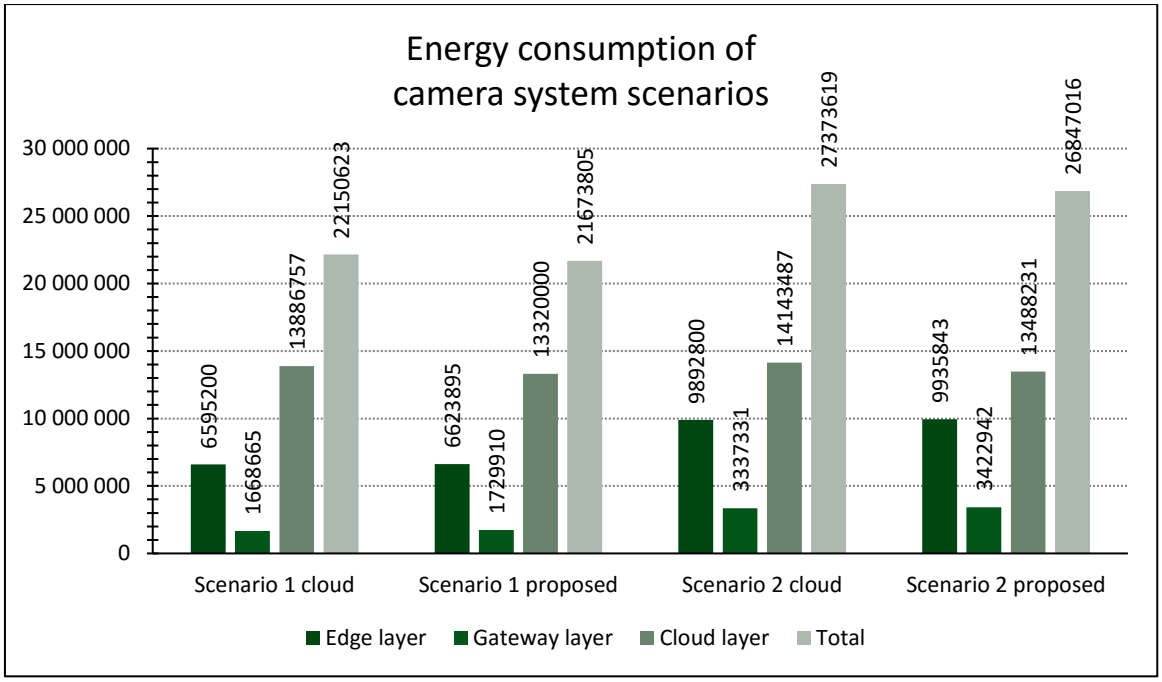

Figure 4. Energy consumption of camera system simulations

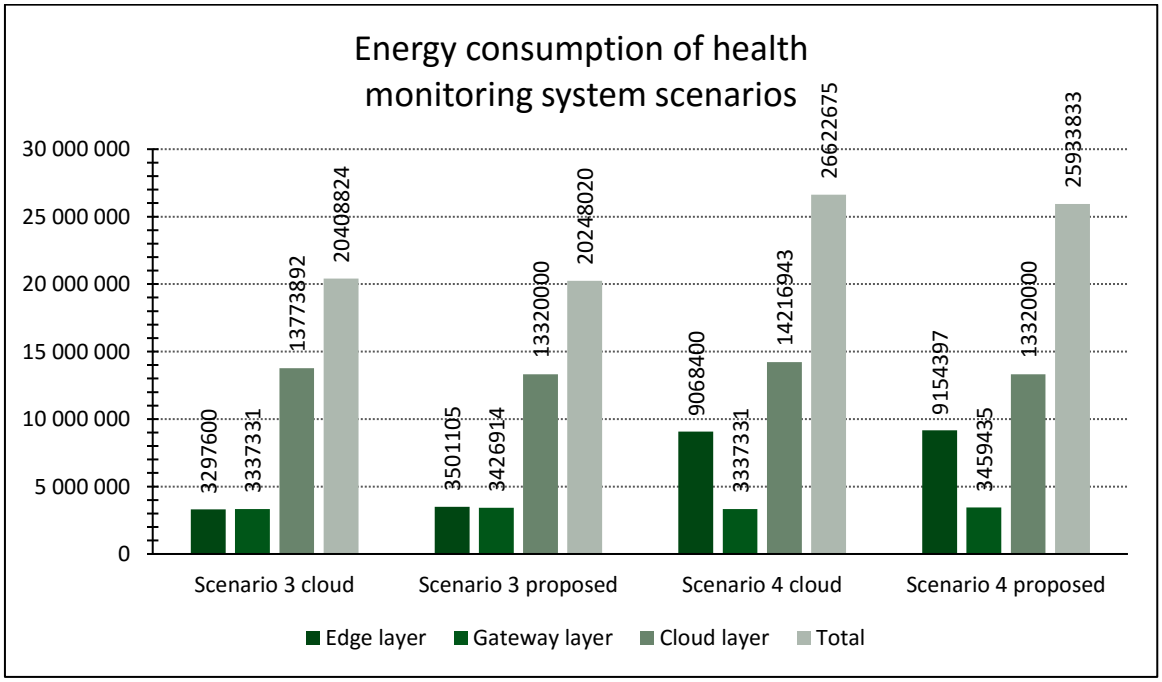

Figure 5. Energy consumption of health monitoring system simulations

5.2 Analysis of the results

The proposed algorithm tries to place modules as low as possible and group the up, so it leaves more free hosts in the cloud. While the power consumption in the cloud went down, the power consumption of gateway and edge devices went up. The power profile of these devices were taken from the example simulations in iFogSim. The profiles of cloud and

gateway devices are similar, idle power usage is about 20W lower than power consumption under load. In the edge devices however, the difference between idle and busy power usage is much smaller, only 5W. This means that offloading computation from the cloud to the edge devices in this case is energy efficient, since the power consumption increase is quite low. This together with the fact, that the maximum energy consumption of the edge devices is lower than the one of the cloud devices, explains why the total energy consumption is lower using the proposed algorithm. It should be noted that placing modules in gateway devices rather than the cloud results in little to no gain in power efficiency, because of the similar power profiles.

The difference in total energy usage between cloud only placement and proposed placement is different in the scenarios. The second and third scenarios show a difference of 526 603W and 688 842W while the first and second scenarios show 476 818W and 160 804W. The difference is larger in the scenarios which had more devices and more modules. The increased difference in energy efficiency gain is again caused by the fact that moving modules to edge devices causes energy efficiency to rise. The second scenario has more edge devices, which means that there is more potential energy savings to gain. Based on these results there is reason to believe that this algorithm will save more energy if the power consumption while utilized is more similar to the power consumption while idle in lower level devices. In addition, if the cloud hosts could be shut down or put into low energy sleep mode, then there should be a bigger reduction in energy consumption. These simulations were not able to simulate such behaviour.

The cloud only placement results in practically no latency between the modules. It does however take a long time for the data to travel between modules and sensors or actuators. The proposed algorithm reduces this latency by placing modules closer to the data source or consumer. This also means that the ideal latency between modules suffers because some of the modules are no longer in the cloud. The total latency in average is still better for the proposed algorithm. This result supports the claim that fog computing will reduce latency compared to cloud computing.

To test the proposed algorithm two applications were created. The applications simulated on two different topologies. The energy consumption of different layers and the delay between modules were recorded and presented. The results show a decrease in total energy

consumption when using the proposed algorithm. The total latency was also improved by using the fog architecture instead of the cloud. This is consistent with existing research.

6. Conclusions and future research directions

This thesis described energy efficient fog computing. It gave a brief overview of the state-of-the-art in fog computing. A short summary of different scheduling algorithms used in cloud and fog computing was written. An algorithm was proposed and tested with simulations. Finally the results were analysed and conclusions drawn.

Firstly a model of fog computing network was created. It consisted of three layers of devices: edge, gateway and cloud. In addition to that, the modelling also described how energy savings can be archived by maximising the idle time of the devices in the network. The thesis also shows a way to model applications as modules that communicate with each other.

The thesis proposed an algorithm that tries to maximize the number of idle hosts by grouping the modules on as few devices as possible. The algorithm was implemented in iFogSim software. Two applications were simulated with different number of devices and modules. The simulations were also done using iFogSim. Both the proposed algorithm and cloud only algorithm were tested on each scenario. The results were compared and analysed.

The results of the thesis show that the use of fog computing architecture can benefit some applications in terms of energy efficiency. The comparison with cloud only placement shows that, while the energy consumption of edge and fog devices went up, the total energy consumption of the system went down because of more idle hosts in the cloud. In addition to that, the thesis found that using the proposed algorithm resulted in lower latency. That is in line with other works on fog computing.

To improve on these results the algorithm should be optimized to have better time complexity. Then it could be tested out on larger systems. It should also be compared to other algorithms proposed for fog computing. The algorithm should be tested on more simulation scenarios, where the devices have different energy consumption rates when idle and under load.

7. References

[1] IDG Enterprise “2016 IDG Cloud Computing Survey”, 2016 https://www.idg.com/tools-for-marketers/2016-idg-enterprise-cloud-computing-survey/ (05.05.2019)

[2] CISCO. “Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are.” White paper, CISCO, 2015.

[3] Dastjerdi, Amir Vahid, and Rajkumar Buyya. "Fog computing: Helping the Internet of Things realize its potential." *Computer*49.8 (2016): 112-116.

[4] Sarkar, Subhadeep, Subarna Chatterjee, and Sudip Misra. "Assessment of the Suitability of Fog Computing in the Context of Internet of Things." *IEEE Transactions on Cloud Computing* 6.1 (2018): 46-59.

[5] Gelenbe, Erol, and Yves Caseau. "The impact of information technology on energy consumption and carbon emissions." *Ubiquity* 2015.June (2015): 1.

[6] Kaplan, James M., William Forrest, and Noah Kindler. *Revolutionizing data center energy efficiency*. Technical report, McKinsey & Company, 2008.

[7] Barroso, Luiz André, and Urs Hölzle. "The case for energy-proportional computing." (2007).

[8] Fan, Xiaobo, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power provisioning for a warehouse-sized computer." *ACM SIGARCH computer architecture news*. Vol. 35. No. 2. ACM, 2007.

[9] Buyya, Rajkumar, et al. "A manifesto for future generation cloud computing: research directions for the next decade." *ACM computing surveys (CSUR)* 51.5 (2018): 105.

[10] CISCO “Cisco Delivers Vision of Fog Computing to Accelerate Value from Billions of Connected Devices”, 2014 https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1334100 (09.05.2019)

[11] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

[12] Villars, Richard L., Carl W. Olofson, and Matthew Eastwood. "Big data: What it is and why you should care." White Paper, IDC 14 (2011): 1-14.

[13] Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." Computer networks 54.15 (2010): 2787-2805.

[14] https://www.openfogconsortium.org/

[15] Mouradian, Carla, et al. "A comprehensive survey on fog computing: State-of-the-art and research challenges." *IEEE Communications Surveys & Tutorials* 20.1 (2017): 416-464.

[16] Yi, Shanhe, Cheng Li, and Qun Li. "A survey of fog computing: concepts, applications and issues." *Proceedings of the 2015 workshop on mobile big data*. ACM, 2015.

[17] Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012.

[18] Stojmenovic, Ivan, and Sheng Wen. "The fog computing paradigm: Scenarios and security issues." *2014 Federated Conference on Computer Science and Information Systems*. IEEE, 2014.

[19] Singh, Sukhpal, and Inderveer Chana. "A survey on resource scheduling in cloud computing: Issues and challenges." *Journal of grid computing* 14.2 (2016): 217-264.

[20] Sinha, Utkal, and Mayank Shekhar. "Comparison of various cloud simulation tools available in cloud computing." International Journal of Advanced Research in Computer and Communication Engineering 4.3 (2015): 171-176.
[21] Gupta, Harshit, et al. "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments." *Software: Practice and Experience* 47.9 (2017): 1275-1296.
[22] Buyya, Rajkumar, Rajiv Ranjan, and Rodrigo N. Calheiros. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities." 2009 international conference on high performance computing & simulation. IEEE, 2009.
[23] Ghribi, Chaima, Makhlouf Hadji, and Djamal Zeghlache. "Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms." *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. IEEE, 2013.
[24] Beloglazov, Anton, and Rajkumar Buyya. "Energy efficient resource management in virtualized cloud data centers." *Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing*. IEEE Computer Society, 2010.
[25] Li, Bo, et al. "Enacloud: An energy-saving application live placement approach for cloud computing environments." *2009 IEEE International Conference on Cloud Computing*. IEEE, 2009.
[26] Mahmud, Redowan, et al. "Quality of Experience (QoE)-aware placement of applications in Fog computing environments." Journal of Parallel and Distributed Computing (2018).
[27] Wu, Hsiang-Yi, and Che-Rung Lee. "Energy efficient scheduling for heterogeneous fog computing architectures." *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. IEEE, 2018.
[28] Deng, Ruilong, et al. "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing." *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015.
[29] Sarkar, Subhadeep, and Sudip Misra. "Theoretical modelling of fog computing: A green computing paradigm to support IoT applications." *Iet Networks* 5.2 (2016): 23-29.
[30] Buyya, Rajkumar, and Satish Narayana Srirama, eds. *Fog and edge computing: principles and paradigms*. Wiley, 2019.

Appendix

I. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Herman Meier**,

(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Simulating energy efficient fog computing,

(title of thesis)

supervised by Satish Narayana Srirama,

(supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Herman Meier

09.05.2019