UiO : Department of Informatics
University of Oslo

Implementation of Intelligent Algorithms on Data Centers for Smart Energy Utilization

Muhammad Qammar Shehzad

Master's Thesis Autumn 2017

Implementation of Intelligent Algorithms on Data Centers for Smart Energy Utilization

Muhammad Qammar Shehzad

December 18, 2017

Acknowledgement

First of all, I thank to Allah o Rasool (Peace be upon him) who blessed me in every moment of my life. Then I would like to express my sincere gratitude to my thesis supervisor, Hårek Haugerud and Stefano Nichele for their expertise, guidance and inspiration, which motivate me to finish this project. Hårek and Stefano, I am very thankful for your humble and polite attitude towards me.

I am also very grateful to Kyrre Begnum, Paal Engelstad and Anis Yazidi at Oslo and Akershus University College of Applied Sciences for spending their precious time and sharing their knowledge with me.

Thanks to Oslo and Akershus University College of Applied Sciences and the University of Oslo for accepting International students and giving us the chance to take part in this master programme and for providing an interesting education of high quality under skilled, knowledgeable professors.

Finally, I would like to thank my family and friends for their support, encouragement, which helps me to complete this project.

Sincerely,
Muhammad Qammar Shehzad

Abstract

Cloud computing is an emerging technology that offers on-demand services. With the invention of sensors and actuators, more physical devices connects to internet which creates bulk amount of data. The data generated from these devices processed, analyzed, and transmitted on cloud which results massive load on servers.

With the help of hardware monitoring chips integrated on CPU(s) servers we can monitor temperature of data centers to reduce the electric energy consumption of servers. To optimize data centers workload many techniques have been explored, where load balancing technique is one of them which has been explored by many researchers from all over the world to evenly distribute the workload.

In this paper, we proposed two modified version of ant colony optimization algorithm based on static load balancing strategies to gives homogeneous temperature on all NUMA node(s) server. Experiment results were illustrated with the help of charts and statistical distribution tools. The results and experiments showed that proposed high to low mechanism performed better and gives homogeneous temperature on all NUMA node(s) compared to move to next algorithm technique.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Earth is warming day by day. One of the reason is global warming which caused by human activities who emits more Carbon Dioxide (CO2) and other greenhouse gases in the atmosphere which results trapping of extra heat in Earth's climate system [54]. According to National Centers for Environmental Information 90 percent of extra heat trapped in Earth's climate system ends up in oceans which makes ocean heating and sea level rise. Natural disasters like Hurricanes, Drought, Inland floods, Severe local storms, Wildfires, Crop freeze events and Winter storms are all results of ocean heating and sea level rise. [9]

With the rise of physical devices connected to the Internet, and growing billions of internet users, tons of data are collected from devices all over the world. [32] The data generated by billions of internet users, requires huge storage space called data centers which includes dedicated servers. These data centers needs huge amount of energy to process data which heats up servers and become cause of pollution, change of climate, and resource extraction. [55]

Data centers of big companies like Facebook, Google, Microsoft, and Amazon consumes huge amount of electricity. According to the U.S. Department of Energy, data centers in U.S consumes approximately "one-fiftieth of total U.S. electricity use in 2014, equivalent to the energy consumption of 6.4 million average U.S. homes." [39] As the computer servers heats up, there is a need of cooling system at data centers which consumes roughly 40 percent of energy. To reduce data center energy consumption Google and Microsoft are applying different techniques like trapping hot air and cooling with water and to build underwater data centers.[39]

Small, medium, and large organizations are moving their services to the cloud because of high flexibility, cost efficiency, robustness and scalability. [4] Cloud computing is a way to provide users remote access to virtualized computing storage and resources to build their IT infrastructures. To make cloud services up and running all the time, live migration techniques used which al-

lows virtual machines to move from one host to another physical host without considering of electricity consumption by servers.

Connectivity, networks, things, applications, big data and cloud are the main components of IoT. With the invention of sensors and actuators, more physical devices connects to internet which creates bulk amount of data.[36] This data needs to process, analyze, and transmit to cloud for storage. One of the domain of IoT is smart energy and utilities which focus on energy efficiency, reducing cost, and automation of services. With the setup of temperature sensors at data centers, we can optimize the cooling system to reduce energy costs. [27]

According to Gartner, the number of IoT devices will reach upto 20 billion by year 2020. [15] As data and communication networks increase, centralized control management of whole network is challenging in this scenario. Traditional centralized management systems may not be the optimal solution because of single point of failure, and dynamic requirements of cloud services. So we need some kind of optimization algorithms for self managed systems.

Bio inspired algorithms which includes artificial immune systems, genetic algorithms, and neural networks are optimal solutions for self managed systems. Ant Colony Optimization (ACO) is one of the optimization algorithm which finds the optimal path from source to destination. [5] With implementation of Ant Colony Optimization on data centers, we can reduce energy consumption, balance load work between servers, and optimize the temperature at data centers using IoT sensors.

1.1 Problem Statement

Implementation of Intelligent Algorithms on Data Centers for Smart Energy Utilization

This paper addresses the following issues:

- How we can smartly utilize energy in Data Centers using Ant colony optimization Algorithm ?
- How we can minimize temperature of data center using server sensors ?
- How an Ant colony optimization algorithm can used to balance servers workload ?
- How server sensors are useful to prevent data centers overheating with the implementation of ACO on workload ?

So to summarize with the help of monitoring hardware temperature sensors on servers we will optimize the efficient usage of energy at data centers using static load balancing mechanism based on Ant Colony Optimization algorithm.

Chapter 2

Background

2.1 Internet of Things

2.1.1 Definition

We are living in a connecting world where physical devices are connecting to the internet to improve our lives. The term Internet of Things comes up when Kevin Ashton [35] was working on advanced technology called Radio Frequency Identification in 1999.

Several components involve in the Internet of Things like: people, things, infrastructure, processes, and data. People who utilize the things like thermostats, wearable devices, cameras, home automation. Infrastructure which is the way of communicating to Internet, data which collects from devices and transfer to other devices, from human to machine (H2M), machine to machine (M2M), or machine to human (M2H). Processes which make sure that people, things, infrastructure and data all work together in a manageable way.

The Institution of Electrical and Electronic Engineering, or IEEE, which is the world's largest technical organization, defines the Internet of Things as "An IoT is a network that connects uniquely identifiable "Things" to the internet. The "Things" have a sensing/actuating and potential programmability capabilities. Through the exploitation of unique indentification and sensing, information about the "Things" can be collected and the state of "Things" can be changed from anywhere, anytime, by anything." [20]

2.2 Physical Devices

One of the main component of IoT is hardware. The main objective of hardware products is to capture data, sometime process the data and then transfer the data over the network to the application. They also have the ability to convert

analog to digital signals. In hardware products we usually have sensors, actuators and embedded system.

2.2.1 Sensors

Sensors are playing important role in IoT hardware which enable communication between devices over the internet. Sensors collects real time data from physical devices which are available in real environment.

Many applications are using sensors such as bio-sensors, nanosensors and implantable sensors are implemented in health care industry for patient care and monitoring patient health. Smartphones, tablets, and gaming consoles are using motion sensors. Refrigerators, ACs, washing machines, and other appliances in homes and kitchen are using pressure, temperature, and proximity sensors.

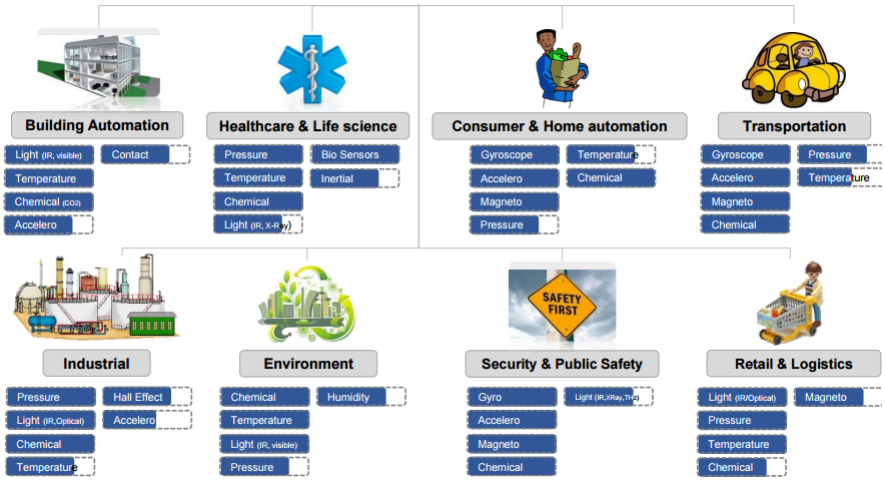

Figure 2.1: Sensors used in IoT Applications Area (Courtesy: Yole Development, as presented at Sensors Expo 2015.)

2.2.2 Actuators

As sensors collects data from the real environment and that data sometimes process locally or transferred to internet for analysis. Now after processing and analysis of data, it has to trigger some action where the actuator takes part. [3] In IoT, an actuator is a physical device which do some operations based on the results of the data from sensors, it modifies the physical state of the device. [20] For example, if room temperature gets hot, it triggers alarms and if there is a fire in room it switched on water shower.

2.2.3 Embedded Systems

The backbone of IoT are the sensors and actuators which are embedded in real environment to work autonomously without the interaction of human. They collect huge amount of data from various sensors like temperature sensors to balance the room temperature and light sensors to optimize energy consumption that are implemented in real environment. [36]

2.3 Applications of IoT

2.3.1 Buildings and Homes Automation

With the introduction of sensors and actuators, homes and buildings are taking advantages to connect to the internet. With the help of IoT devices at homes and buildings, energy consumption and budgets can be reduced. The implementation of IoT at homes and buildings improved the overall quality of life. [34] For example, in terms of IoT within Smart Home automation devices are being used for HVAC (heat ventilation and air conditioning) system to balance the temperature and humidity inside rooms and buildings.

Smart locks, surveillance cameras, and security alarms are some of the examples in Smart Home automation. Within Smart buildings, light sensors are being used for smart lighting to reduce energy consumption and ultrasonic sensor for smart parking. [36]

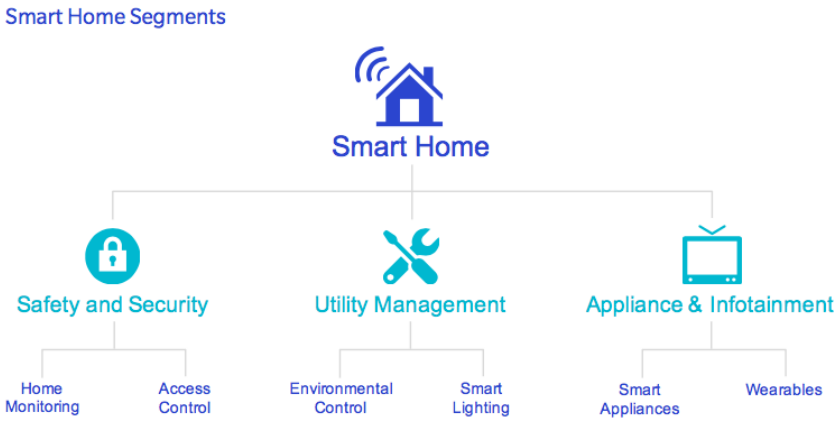

Figure 2.2: Smart Home Segments [52]

2.3.2 Medical and Healthcare Systems

E health management systems is one of the important domain in IoT applications. Medical monitoring of elderly and disabled patients while treatment allowed them to feel comfortable in their zones. [10] Intelligent sensors applied inside or outside of patient bodies depending on their conditions, intelligent sensors collects patients physiological information and via gateways to process autonomously or if there is a need of further analysis transferred to cloud and transmitted to respective medical staff if there is a need of further treatment or any emergency. [36]

It lowered the cost of healthcare and improve patients health with remote monitoring. Huge amount of medical data stored on cloud assists in smart decision making by analyzing patients individual health. With smartphone devices patients trust more on health apps rather than doctor.[40] Below figure shows some of the Smart health cases:

Figure 2.3: Smart Health Use Cases [30]

2.3.3 Infrastructure Management

Another important domain in IoT applications is industrial automation. Wireless connectivity, innovative hardware, advanced sensors networks and machine to machine communication is changing the automation process in industries. For data communication with Cloud, many API libraries and industrial protocol modules are developing to integrate with industrial devices. Some of industrial protocols includes CAN Bus, RS-232 and RS-485.[21]

To reduce the emission of carbon dioxide and other green house gases, organizations and governments all over the world are using wireless technologies to reconstruct traditional energy infrastructure into interconnected Smart Energy Grids.

Smart metering is plays a pivot role in this regard which has vision to give consumers having control over energy usage to reduce carbon emission and save money. With cellular connectivity for smart metering infrastructure provides low power cost effective solution which can be accessible from anywhere. [8]

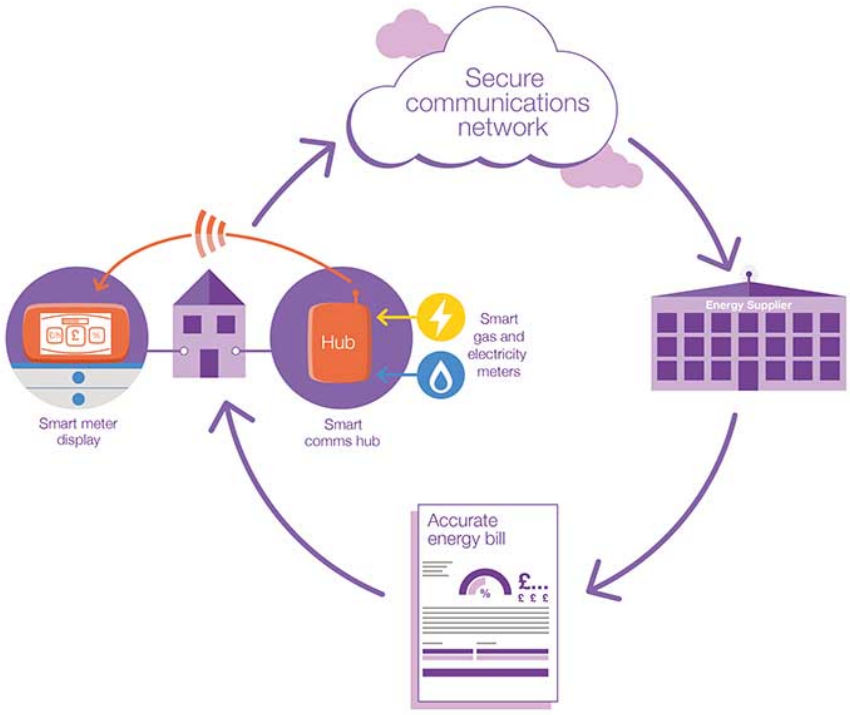

Figure 2.4: Smart meters: paving the way for our future energy system [58]

2.3.4 Energy Resource Management

Electricity plays a vital role in our daily life activities to run smoothly in office's, buildings, homes, institutions, industry and transportation. With the manufacturing of embedded devices containing distinct electronic circuits and other components such as sensors and actuators, the utilization of appliances and everyday objects becoming digitize and more smart day by day.

Rapid growing of smart devices usage for personal purposes, gives more attention to IoT based applications for smart homes, smart buildings, smart mobility, and smart healthcare etc. [24]

Smart home applications which focuses on consuming energy efficiently, with implementation of smart lights, heat controlling system alarms and security systems. IoT based devices connects remotely via wireless technologies such as ZigBee, 6LoWPAN or Bluetooth etc to create smart energy ecosystems which integrate smart meters, smart grids distribution facilities and power generation.[13]

The IoT enables new energy management with Smart metering and Smart grid to build greener energy solutions such as:

- Smart meters to use energy efficiently, and managing of home electric appliances remotely.
- Dynamic load balancing and decreasing peak energy consumption.
- Self managing grids in case of electrical failure.
- Optimization of generated power with forecasting and streamlining production.
- Integration of renewable power sources.
- Monitoring of alarms and events and avoid energy leakages [53]

2.4 Connectivity

IoT based smart devices must have connectivity and exchange information between them and transferred to centralized servers. For that purpose, they need some communication protocols which connects devices like sensors, actuators, databases and cloud platforms, where information is stored. Machine to Machine (M2M) term is used where devices talk to each other. Based on data collected from IoT devices, they make decisions locally or remotely which trigger events such as turning lights on and off.[2]

Communication technologies used between IoT devices and servers are Wireless Wide Area Networks (WWAN) and Wireless Local Area Neworks (WLAN). Some of medium range wireless technologies used to communicate between IoT based devices and IoT gateways are WiFi, Zigbee and Bluetooth etc. [25]

2.4.1 ZigBee

ZigBee is a self healing, robust, and mesh capability protocol which defines the network security and application layers TCP/IP using IEEE 802.11b network

specification. ZigBee protocols are used where a cost effective solutions are required, which supports low data rate, low power, security and reliability. IoT domain where ZigBee protocols are being used are Smart homes, Smart buildings automation and Smart health care. ZigBee protocol range lies between 10 to 100 meters, and supports data rate up to 250 kilobytes per second, and the frequency band is around 2.4 GHz range to 915 MHz.[22]

2.4.2 6LoWPAN

Another low power wireless technology is 6LoWPAN which stands for Low power Wireless Personal Area Networks where 6 represents IPv6. In this communication protocol, data is communicated over IEEE 802.15.4 networks specification. IPv6 over low-power wireless standard is approved by Internet Engineering TaskForce (IETF) organization to enable IP communication over any low-power wireless or wired media. [41]

2.5 Data Centers

Data centers are spread all over the world which consists of multiple servers. Data centers grows exponentially to store big data. It provides IT services for infrastructure, cloud computing and virtualization. Big Data centers have hundreds of servers which stores huge amount of information which serves their respective users 24/7.[11]

Big businesses like banking, online stores, transactions and online services are all dependent on data centers. As more devices are connecting to internet, data centers are increasing as well.Due to extra workload, servers heats up, which causes cooling system to run that consumes roughly 40 percent of total energy at data centers.[44]

Without consideration of energy consumption, service providers keeps their services online. One of the important challenge for service providers is to create infrastructure of data centers environmentally friendly and fossil fuel free economy.

Big names in IT market Amazon, Google, Microsoft, and IBM providing cloud services are experimenting to develop "Green Cloud". For example Google Data Centers trap the hot air produced by servers and cooling down with water.[57] Also Microsoft are building underwater data centers for energy efficiency and to protect the environment. [56]

2.5.1 Big Data

Massive amount of data generated by IoT devices needs to store for some useful events or sometimes automated workflows. Big data comes handy which make it possible to analyze the data and turns into meaningful actionable information. Big Data is that everything we do is leaving a digital trail and is useful for making self decision and making autonomous system.

Mainly big data is categorized into three V's, Volume which defines as huge data size from Terabytes (TBs) to Petabytes (PBs), Velocity means processing, analyzing and modifying of real-time incoming data with high speed and output the result without any delay, Variety includes different kinds of structured and unstructured data.[17] Related to IoT the most prominent features of IoT is its real-time or near real-time communication of data with the IoT connected devices.

2.5.2 Cloud Computing

Cloud computing is a prominent technologies adopted by small, medium and large organizations. Cloud computing provides resources which are shared to computers and can access remotely. This means users paid for on-demand services, instead of having physical data centers to match their enterprise computing requirements.

Access is given to users to their cloud-based computing resources, which they can scaled up or down according to their own needs. With internet connection, users can have access to their resources at anytime and from anywhere. [37]

In cloud computing system, two components are involved the front end which displays interfaces and applications to manage cloud computing resources to users and the back end which involves servers integration with resources to store data. Some features of cloud computing environments are: on-demand self service, scalability, availability, resource pooling and elasticity. [37] [38]
Following are three common service models of cloud computing:

2.5.2.1 Software as a Service (SaaS)

In this service, consumer have access to use applications running on infrastructure of cloud. Users don't have control to modify the applications background resources on which operating system, database or network the applications are running. Office 365 and Google Docs are examples of SaaS model. [38] [51]

2.5.2.2 Platform as a Service (PaaS)

In this service, users can deploy their applications on cloud infrastructure which are compatible with PaaS provider set of tools and programming languages. Same as SaaS model, users doesn't have control on underlying technologies on which their application will deploy. Google App Engine and Micrsoft Azure are famous examples of PaaS model. [51]

2.5.2.3 Infrastructure as a Service (IaaS)

In this service, users have access to the infrastructure of computing resources where they can manage servers, networks, operating systems, storage and other resources. Clients can run any software regardless of operating systems, services and applications compatibility as compared to Platform as a Service (PaaS) model. Amazon Web Services EC2 and S3 are prominent example of IaaS model. [38]

2.5.3 Virtualization

Vritaulization is the IT infrastructure which provides resources including operating systems, storage, servers, and networks. Small medium sized organizations can build their own infrastructure virtually. Some features of virtualization are dynamic application development and software testing which can be tested under development servers before moving to production servers.

Dedicated servers for small organization who can run multiple virtual environments with small amount of processing power. System security which introduced a layer of security policies. [33] Below are Some of the benefits of virtualization

- Dedicated Servers
- Green Computing
- Availability, Scalability, Reliability
- Optimize Energy Utlization Solutions

2.5.4 Load balancing

With rapid growing of internet users and connected devices, number of users also increasing in cloud technology. It is a challenging task for cloud providers to balance the workload of their shared resources at data centers infrastructure. Many algorithms have been suggested by researchers for load balancing in cloud

environment for example distribution of task among nodes based on virtual machines performance, task scheduling and nodes temperature etc.[26]

The main objective of load balancing is to distribute the workloads across servers without affecting resources in Cloud. Some features of load balancing are:

- Performance: which ensure the execution of task and response in less interval of time.
- System Stability: which ensure that their will be no data or packet loss and communication delay.
- Security, reliability and customer satisfaction are important aspects in cloud environment.
- Backup nodes: in case of system failure there should be backup servers.

2.6 Related work

With the increase of mobile dynamic applications, online gaming, live streamed contents, social networking websites and other online services, the burden on cloud services provider are getting high. Therefore cloud service providers are more concern about the management of cloud computing shared resources efficiently to enhance performances and quality of service (QoS).

As mentioned in load balancing section, the main objective of load balancing is to distribute the workload among resources to avoid resource utilization overloaded or underloaded in cloud environment. Load balancing has been categorized into two types, dynamic load balancing which migrates the workload while processes are running and static load balancing which scheduling the processes on system start up.

Besides traditional algorithms like First Come First Serve (FCFS), Round Robin (RR), Random Allocation (RA) and Shortest Job First (SJF) etc, many other intelligent algorithms such as Genetic Algorithms (GA), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO) have been proposed for load balancing by researchers from all over the world.

This section contains review of previously conducted research on load balancing of nodes on cloud environment using Genetic Algorithms (GA), Swarm intelligent algorithms, Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO).

2.6.1 Genetic Algorithms

Genetic algorithms have been proposed by researcher for its simple implementation of solving optimization problems such as load balancing. The main objective of this algorithm is to generate optimal solutions by filtration and search problems by depending on bio-inspired operators such as mutation for strong individuals, crossing over and selection.[16] In this [12] paper authors proposes load balancing strategy to search overloaded node and under loaded node and finally simulate the results using CloudAnalyst simulation tool.

In this paper[23] authors focuses on energy consumption of cloud environment. Based on genetic algorithm, the authors performed a scheduling strategy for efficient energy usage in Cloud computing systems. [23] In this paper [43] genetic algorithm (GA) based task scheduling in cloud is proposed where population is generated by enhanced Max Min technique for individual tasks. [43]

2.6.2 Particle Swarm Optimization

Another bio inspired algorithm is Particle Swarm Optimization, which simulates the birds foraging process. PSO is a methodology for optimization whose objective is to find a global optimal solution. The PSO algorithm iterates from a set of local solutions and find the targeted value which obtained from the fitness function.[46]

In PSO, the analogy of bird flocks referred to as "swarm" and the birds referred to as "particles". And if we consider population-based approach then analogy would be "swarm" considered as population and the candidate solutions referred to as "particles".[14]

In low computational applications, PSO is useful due to its simple implementation. But in large scale optimization problems, it takes a lot of computational costs therefore researchers considered other bio inspired algorithms like Ant Colony Optimization (ACO). In this paper[49] authors, proposed a self adaptability of ant colony optimization parameters, where the parameters of ACO such as selection and pheromone's update are taken from the implementation of PSO.

With the help of CloudSim simulation tools, they compared the period ACO based scheduling algorithm (PACO) and self adaptive ant colony optimization (SAACO) algorithm and results shows that SAACO has better performance regarding makespan and load balancing as compared to PACO.

2.6.3 Ant Colony Optimization

Ant Colony Optimization (ACO) is one of the optimization algorithm which finds the shortest optimal path from source to destination. [31] This algorithm inspired from Ant's natural behavior. First, they explored the area randomly through the path. As they found the food source they measure quantity and quality of food. And take some of the food back to home. On their way back, they use pheromones to guide other ant's and set the shortest path between destination and food source. Pheromones are temporary information and they evaporate quickly from the path.[18][7]

Ant colony optimization algorithm have been explored by many researchers, which they implemented for various tasks such as minimizing power consumption, finding optimal routing path, load balancing of nodes in data centers, task scheduling in cloud environment, balancing workload among virtual machines etc.

In this paper[6], authors concerns is to enable green computing environment for task scheduling where they proposed an ACO-based algorithm to reduce the makspan time and ensure load balancing among resources.

Chapter 3

Approach

This chapter will focus on methodology and procedure in order to address the main problem stated in thesis title i.e "Implementation of Intelligent Algorithms on Data Centers for Smart Energy Utilization". Furthermore, the proposed technique of "Smart Energy Utilization" on data centers is based on "Intelligent Algorithm" which is Ant Colony Optimization which will detect the overloaded node and underloaded nodes on server and balance the temperature of nodes. Following are the main objectives of our project which will be discussed in future sections:

- Exploring the basic Ant Colony Optimization algorithm flow
- Making analogy with our model based on basic ACO model
- Generating workload on servers CPUs having sensors temperature
- Analyzing nodes temperature without modified algorithm
- Running our modified ACO algorithm on servers
- Analyzing nodes temperature with modified algorithm
- Comparing results

3.1 Objectives

As described in the problem statement, the main objective is to enable green data centers based on the implementation of Ant Colony Optimization (ACO) algorithm in order to have balanced temperature on nodes of servers. The model presented in this thesis will be modified version of ACO algorithm which will demonstrate self optimization of homogeneous temperature on server nodes.

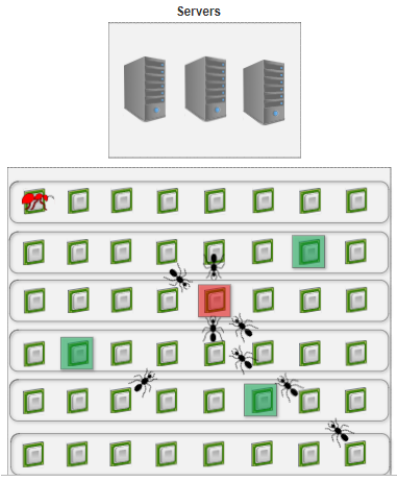

Figure 3.1: Expected scenario on servers CPU(s)

ACO algorithm benefits

To enable green cloud computing, is one of our main goal in this project. Besides traditional algorithms implementation in cloud environment which have centralized system, where one node works as master node who make decisions. There are many drawbacks of implementation of traditional algorithms such as single point of failure cause systems down, dynamic applications requirements cannot be handled by this algorithm. Therefore, decentralized algorithms such as swarm intelligence methods are useful for current era dynamic online applications. In our approach, the modified version of ant colony optimization algorithm which is member of swarm intelligence methods will be implemented which is self organized decentralized system. IBM[42] defines self-managed systems into following four areas:

- Self-Configuration which automatically configure system components
- Self-Healing which detect the failure and correct automatically
- Self-Optimization which monitor and having control over system resources so the defined functions will be self-optimized
- Self-Protection which make sure the system security from attacks

Basic ACO algorithm

First, we will summarize the basic algorithms of Ant Colony Optimization proposed in cloud environment by many researchers. ACO is inspired from

natural behaviour of ants which works together in foraging process. In section 2.6.3, the natural behaviour of ants have been described briefly. Followings are the main tasks of ACO algorithm for load balancing in Cloud:

- Ants started from head node and move over all the nodes in a network
- Ants lay down pheromone while travelling from source to destination and vice versa.
- The strength of the pheromone trails defines the shortest optimal path from source to destination.
- Pheromone evaporate quickly,therefore time and original strength matters.
- While traverse ants update pheromone table .
- Ants moves in two ways direction i.e forward and backward movement.
- In forward movement ants move in forward direction and if finds overloaded node, it will be marked current node and move the workload to underloaded node after the current node. In backward movement if underloaded node comes before overloaded node ants will move in backward direction, and move the workload there.

The main goal of this thesis is to have balanced temperature on server NUMA node(s) through the implementation of ant colony optimization algorithm (ACO). To achieve that goal, this thesis has been structured into following phases:

- Implementation and Testing phase
- Measurement and Analysis phase
- Comparison and Discussion phase

3.1.1 Implementation and Testing phase

In this phase, the modified version of basic ant colony optimization will be implemented, where NUMA node(s) temperature will be stabilized. Following tasks will be included in this phase:

1. Proposing use case for experimenting
2. Creating modified algorithm according to use case
3. Showing algorithm in form of flowchart
4. Creating bash script according to proposed algorithm
5. Performing multiple tests to achieve the required results

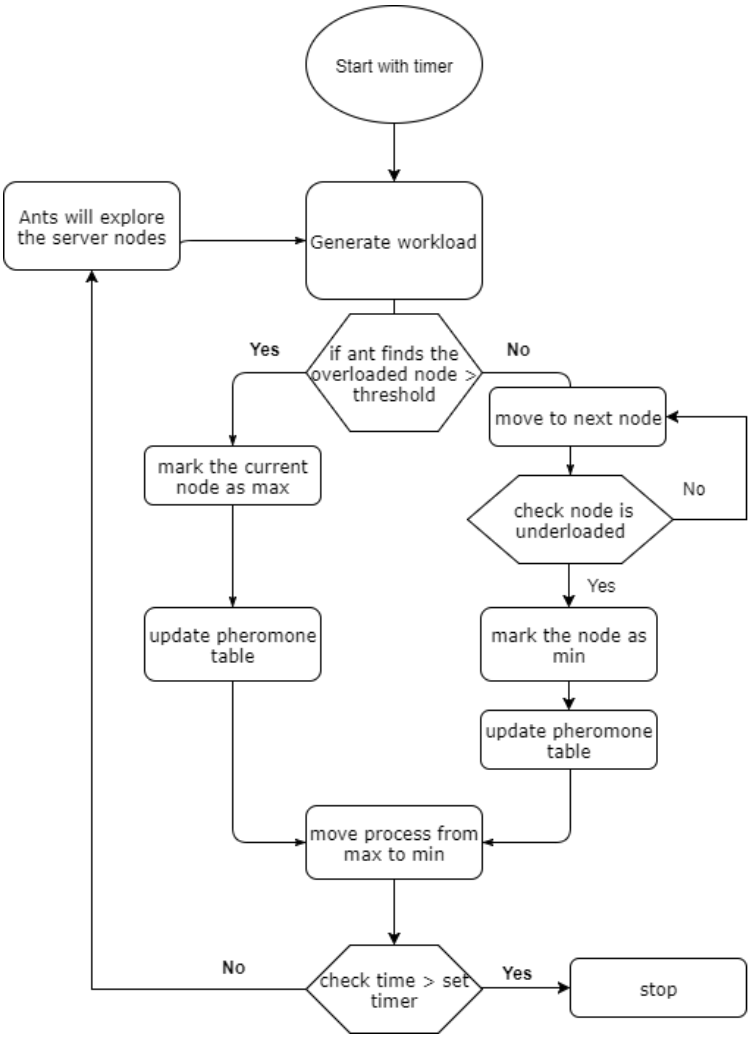

Figure 3.2: Basic ACO algorithm flowchart for load balancing in Cloud

Generating workload on testing environment

In testing environment, there will be no workload by default so in order to generate some workload multiple methods can be applied. In our case we will generate the workload using stress tool. *"Stress is a deliberately simple workload generator for POSIX systems. It imposes a configurable amount of CPU, memory, I/O, and disk stress on the system. It is written in C, and is free software licensed under the GPLv2."*[47]

It is very simple method to generate workload and quickly stress out on server resources which in our case is very useful to move the processes while managing nodes temperature. Stress tool can be installed using command: *"apt-get install stress"*. The updated version of stress workload generator tool is stress-ng which stress out not just CPU compute but also other server components like I/O syncs, drive stress, pipe and UNIX socket stressors, shared memory stressor and virtual memory stressor etc. To get CPU hot matrix size option is used in stress-ng tool. [48]

Testing Environment

The testing environment will be one server at HiOA. The purpose of our project is to have balanced temperature, and the workload will be generated on server CPU(s), therefore no virtual environment will be used.Table 3.1 shows the specifications of server used in our testing environment:

Table 3.1: Testing server specifications

Architecture	x86_64
Model name	AMD Opteron(TM) Processor 6234
Operating System	Ubuntu 16.04.3 LTS
CPUs	48
CPU op-mode(s)	32-bit, 64-bit
On-line CPU(s) list	0-47
NUMA node(s)	8

AMD Opteron(TM) Processor are optimal for cloud infrastructure and hosting environment for delivering excellent performance. Server based on AMD Opteron processor have two types of temperatures CPU temperatures and Core temperatures. Core temperature isn't a real temperature from CPU(s) socket, it's inconsistent which shows temperature in Celsius degrees. Whereas CPU temperature will be our focus in this project which gives sensor temperature in Celsius degrees.[19] In testing server 8 NUMA nodes and 48 CPU(s) will be used to generate workload and temperature reading from sensors. Table 3.2 shows the list of NUMA node(s) and their corresponding CPU(s) in testing server.

Table 3.2: NUMA node(s) and CPU(s)

node 0 cpus	0 4 8 12 16 20
node 1 cpus	24 28 32 36 40 44
node 2 cpus	2 6 10 14 18 22
node 3 cpus	26 30 34 38 42 46
node 4 cpus	3 7 11 15 19 23
node 5 cpus	27 31 35 39 43 47
node 6 cpus	1 5 9 13 17 21
node 7 cpus	25 29 33 37 41 45

3.1.2 Measurement and Analysis phase

In this phase, the expected results of Implementation and testing phase will be presented in form of graphing tools, charts and statistical data. List of tasks performed in this section are following:

- Create a technique to capture NUMA node(s) temperature data consistently based on time interval
- Multiple experiments results
- Comparison of different use cases with statistical data

Chapter 4

Implementation

An algorithm for load balancing suggested that ants pheromone should evaporate after some interval of time. And spread the ants again when needed. To make analogy and some modification in ant colony optimization algorithm (ACO), we will generate work load at initial level after that no new work load will be generated until all NUMA node(s) are stabilized with respect to temperature. In other way, it will be implementation of proposed algorithm on static load balancing instead of dynamic load balancing.

In basic ACO ants decide the path based on strength of pheromones guide by other ants and they update the pheromone table while traversing from one node to another. In our proposed ACO, the path to source or destination will be based on level of NUMA node(s) temperature. Two use cases will be proposed in our experiment to achieve the targeted goal. In following section, detailed structure of proposed algorithm will be presented.

4.0.1 Use cases I: High to Low

In this use case, the generated workload from stress tool will stress out randomly on all CPU(s) of NUMA node(s). After the workload is being generated using stress tool, ants spread all over the network. Ants move to next node and check whether it is high temperature node or low temperature node. If the current node is having high temperature, it will be marked as max node otherwise it will be move to next node until it finds the highest temperature node from all over the node. Same way it will find the lowest temperature and mark it min node.

After both nodes found out, processes running on highest temperature NUMA node(s) CPU(s) which is max node will be moved to lowest temperature NUMA node which is min node. The traversing of ants happens in order to search for food sources in basic ACO algorithm. Same way after some interval of time

max node and min node will be marked to change the affinity of processes and have balanced temperature. Some by default useful commands of linux will be used in this project.

lm-sensors

This linux project main objective is to monitor hardware health. Monitoring thermal sensors which are integrated in CPU of servers is one of the feature of this linux project besides other feature such as monitoring voltage, fan speeds, and hardware sensors integrated in I/O chips, memory modules etc.[29] For this thesis, thermal sensors command *"sensors"* will be used in order to monitor CPU temperature which will output temperature information in the Celsius temperature format.

sort head tail commands

With the help of some simple linux command, the targeted result can be achieved easily. For example sort command which sorts the data numerically and alphabetically from a any output, one of the option in sort command is *"sort -n"* which sorts all the values in a string numerically.[28] Another useful commands in linux are head and tail, head which output the first lines from a given data whereas tail which prints out the last lines from a given data. the default value is 10, if there is no specific number is defined in head and tail commands.[1] For example command *"tail anyfile.txt"* will print last 10 lines from anyfile.txt whereas *"tail anyfile.txt -3"* will print last 3 lines of anyfile.txt.

Therefore the combination of these commands can help in finding the highest and lowest value from a given data. For example the command *"sort -n | head -1"* can print out the first line from a numerically sorted data. The command to print the minimum value of NUMA node(s) CPU temperature is following:

```
#!/bin/bash
sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 | sort -n |
    head -1
```

Listing 4.1: command to get first lowest value

While tail command can print out the maximum value of NUMA node(s) CPU temperature:

```
#!/bin/bash
sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 | sort -n |
    tail -1
```

Listing 4.2: command to get last highest value

When both values found out from given list of NUMA node(s) CPU(s) temperature, the next task is to figure out which NUMA node have highest and lowest temperature. In order to achieve that iteration from all the values is

required and compare each value with our marked highest and lowest values. Following is the pseudocode of *for-loop* which is implemented in this case:

Algorithm 1: ForEach loop to get corresponding NUMA node

Result: Set max and min NUMA node based on CPU temperature
Input: x is the NUMA node(s) starting from 0

```
1 x ← 0
2 foreach no. of NUMA node(s) temperature values do
      /* NUMA node(s) started from 0                    */
3     if If NUMA node value = marked min or max value then
4         set NUMA node to max or min:
5     end
6     increase x by 1
7 end
```

When highest and lowest value NUMA node(s) are identified. The next step is to find the running process IDs (PIDs) of stress inside CPU(s) from highest and lowest temperature NUMA node(s). And change the CPU affinity of running processes from highest NUMA node to lowest value NUMA node based on PIDs. There are multiple ways to figure out the PIDs of running processes. The simple command `"pidof stress"` is used to find out the PIDs. As the workload is generated randomly, processes of stress spread out all over the CPU(s) of NUMA node(s).

CPU Affinity and Taskset

Binding a process to only a specific CPU is CPU affinity so that process will run only from specified CPU. With *taskset* command a process can be set to CPU affinity by providing process ID (PID) of running process. There are multiple options in taskset command such as "-p" which work on an existing PID instead of launching new one and "-c" which gets the list of CPU in its parameters. The list can have more than one CPU separated by comma, and ranges. For example, 0,1,4,7-12 etc.[50]

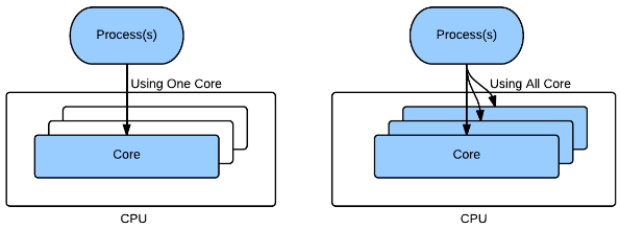

Figure 4.1: CPU and Process architecture[59]

Iterate through all the PIDs of stress and compare it with PIDs running inside highest NUMA node and change the affinity of NUMA node has been shown in following bash code:

```
#!/bin/bash

c=0
pids=$(pidof stress)
for item in ${pidsArray[@]}
do
        cpuNo=$(cat /proc/$item/stat | cut -d' ' -f39)
        echo Running process[$item] on: $cpuNo

        # Getting list of CPU(s) of current NUMA node
        getListofCPUofNode=$(numactl --hardware | grep cpus | head
    -$maxTemNode | tail -1 )

        # Put the list into array
        for i in ${nodeCPUArray[@]}
        do
            #echo Node $node CPU is: $citem
            if [[ $cpuNo -eq $i ]]; then

                if [ $(echo "$c < "4""| bc) -eq 1 ];  then
                    echo Counter: $c
                    (( c = $c + 1 ))
                    echo "Running PID[$item] on CPU $cpuNo lies in
    Node: $maxTemNode"
                    # Moving running process to low temperature
    node

                    listOFlowNodeCPU=$(numactl --hardware | grep
    cpus | head -$minTemNode | tail -1 )

                    # Put the list into array
                    rand=$[ $RANDOM % 6 ]
                    # RANDOM is default function in bash, as we
    have 6 CPU(s) in each NUMA node, so it will randomly pick the
    number upto 6

                    #
                    taskset -p -c ${lowNodeCPUArray[$rand]} $item
                fi
            fi
        done
done
```

Listing 4.3: Bash scripting to change affinity from highest NUMA node to lowest NUMA node

The modified ACO algorithm can further be described in form of following flowchart:

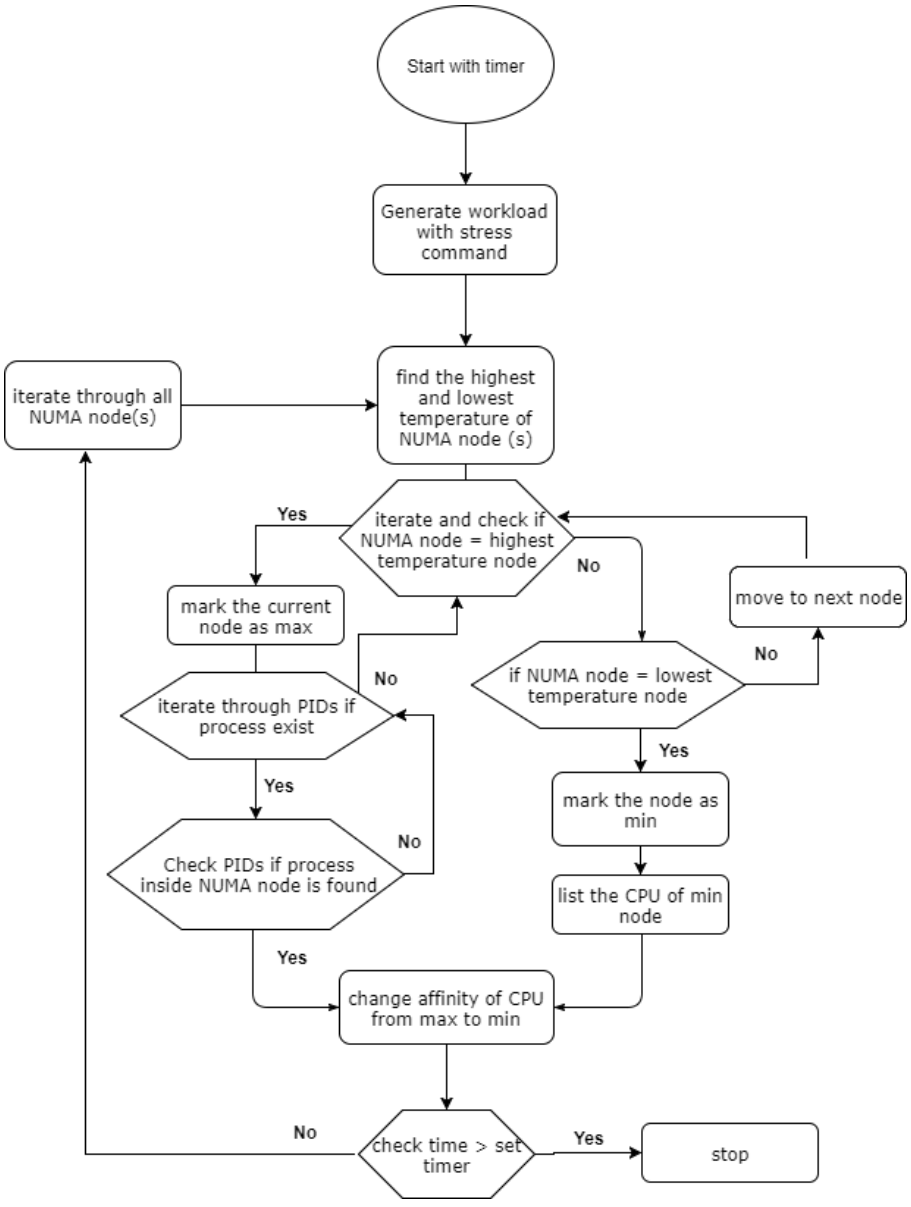

Figure 4.2: Flowchart: high to low algorithm

4.0.2 Use cases II: Move to next

In this use case, the generated workload from stress tool will stress out randomly on all CPU(s) of NUMA node(s). After the workload is being generated using stress tool, ants spread all over the CPU(s) of NUMA node(s). Starting from first node ants check whether it is high temperature node or low temperature node. If the current node is having high temperature than next node, processes of high temperature NUMA node CPU(s) will change affinity to next node CPU(s). Below Figure 4.3 shows the changing affinity of CPU(s) while experimenting.

```
Running PID[1367] on CPU 3 lies in Node: 4
Node 7 contains CPU:  25 29 33 37 41 45
pid 1367's current affinity list: 3
pid 1367's new affinity list: 45
```

Figure 4.3: CPU affinity change

The function will run until it reach to last node. Now the last node will be compared to first node and change affinity with first node CPU(s) if it has high temperature. If current node and next node same temperature, it will do nothing and move to next node. The whole methodology will work in forward direction. The methodology presented in use case I for finding process ID (PIDs) of stress and list corresponding CPU(s) of NUMA node(s) will be same in this case.

In this use case, current node and next node will be compared, so next node temperature and changing of CPU affinity of NUMA node has been shown in following bash code:

```
#!/bin/bash

nextNodeTemp=${nodeTempArray[$(($currentNode+1))]}
lastNodeTemp=${nodeTempArray[${#nodeTempArray[@]}-1]}
if [ -z "${lastNodeTemp}" ]; then

    # Change affinity to first node CPU(s)
    taskset -p -c ${lowNodeCPUArray[$rand]} $item
elif (( $(echo "$currentNodeTemp <= $nextNodeTemp" | bc -l) )); then
  echo "Do nothing move to next node"
else
  # Change affinity to next NUMA node
  #echo Next Node is: $s
  nextNodeCPU=$(numactl --hardware | grep cpus | head -$s | tail -1)
  rand=$[ $RANDOM % 6 ]
  taskset -p -c ${lowNodeCPUArray[$rand]} $item
fi
```

Listing 4.4: Bash scripting to change affinity to next NUMA node

The Use Case II algorithm can further be described in form of following flowchart:

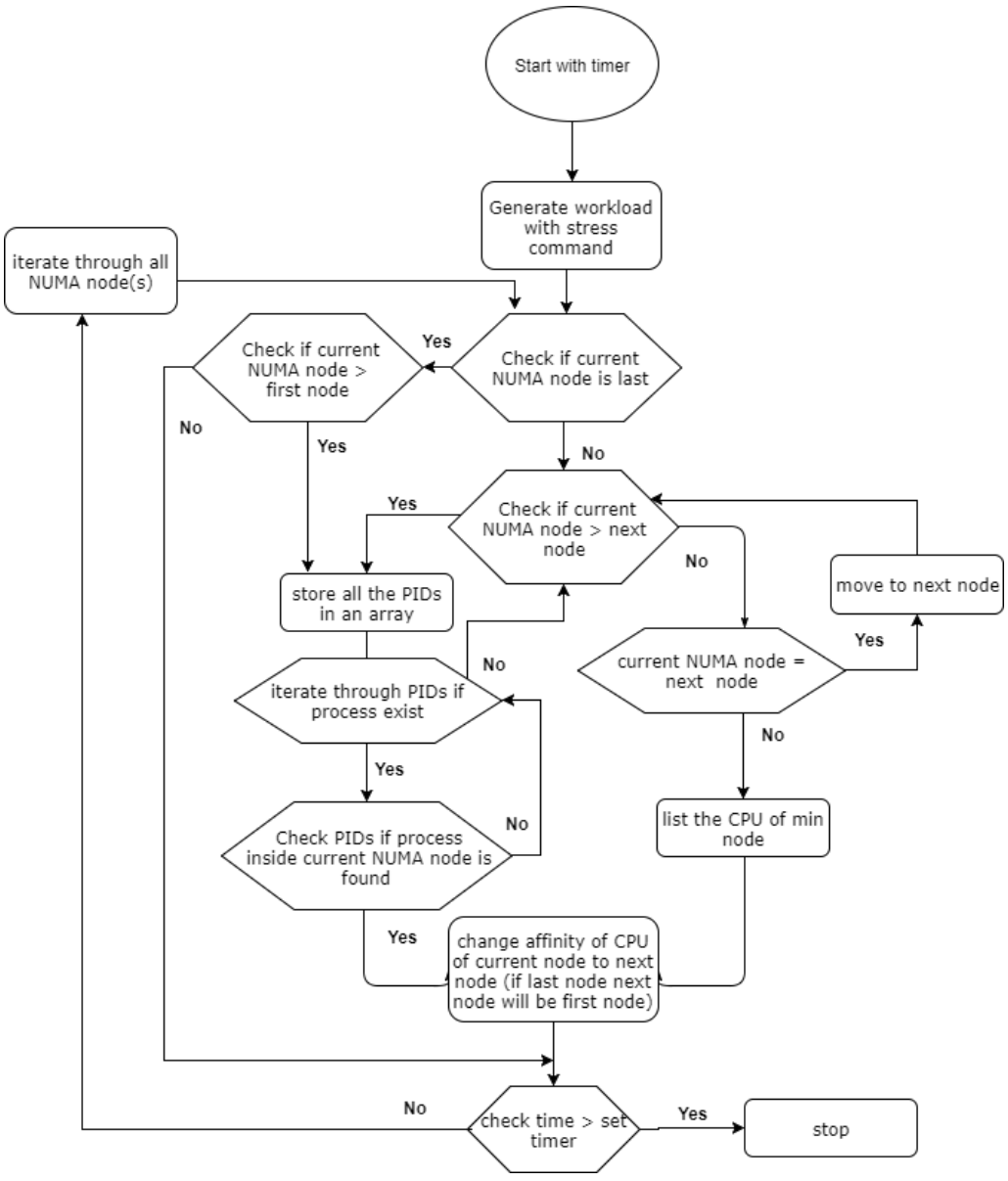

Figure 4.4: Flowchart: Move to next algorithm

Chapter 5

Results, Analysis and Comparison

The experiment performed to enable green computing and having homogeneous temperature on all NUMA node(s) will described in this chapter. The obtained results from proposed modified algorithms will be analyzed using charts and tables. Multiple experiments have been conducted to get required result. The obtained data will be further discussed in discussion section to make a link with proposed problem statement about the smart utilization of energy to enable green cloud computing.

5.1 Preliminary Experiments

Multiple experiments have been conducted in order to have balanced average temperature between lowest and highest temperature values. In testing server, there are 48 NUMA node(s) CPU(s) therefore experiment is grouped in different parameters. Workload generator tool *"stress"* will stress out CPU(s) with -c options such as command: ***"stress -c 34"*** will generate workload randomly on 34 CPU(s).

For tuning and further analyzing of the parameters for our experiment, random workload has been generated on 48 CPU(s) with **"stress -c 48"** command and record the temperature with an interval of 1 minute. These experiments done to have an idea of CPU(s) maximum temperature level and the time to stressed on CPU(s), so that limitations are known before implementation of modified algorithms.

Figure 5.1 shows the NUMA node(s) temperature of 48 CPU(s) stressed without implementation of algorithm. The difference in node(s) temperature is due to baseline of node(s) value i.e. node(s) 2,3,4 and 5 have baseline temperature around 18°C whereas node(s) 0,1,6 and 7 baseline temperature around 22°C. All

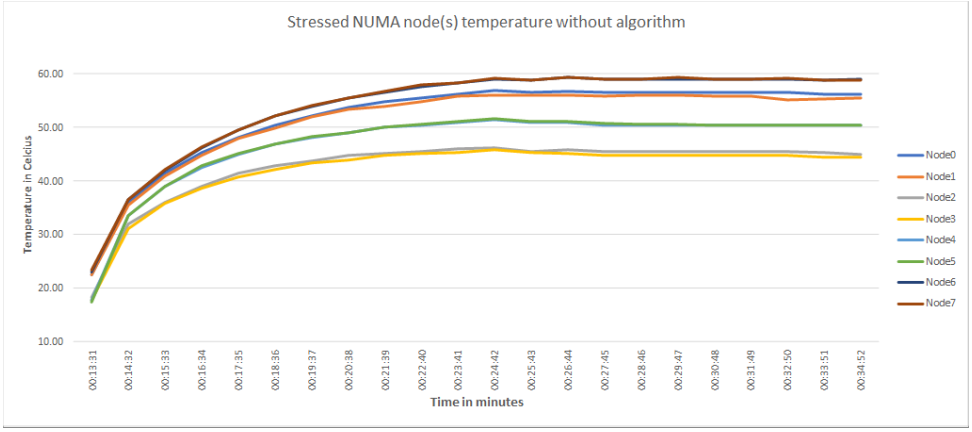

Figure 5.1: This figure shows the temperature of stressed NUMA node(s) on 48 CPU(s) without implementation of algorithm

the node(s) are stabilized after some time where highest value reach upto 60°C and lowest value is around 48°C and average temperature is around 52°C.

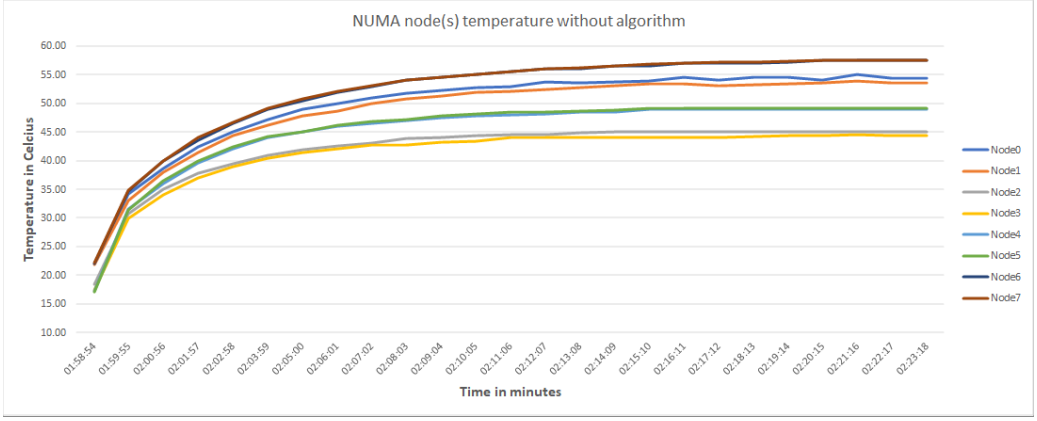

Figure 5.2: This figure shows the temperature of stressed NUMA node(s) on 36 CPU(s) without implementation of algorithm

The tuning of CPU(s) by generating workload using different parameters of stress was done to get an idea about the individual NUMA node(s) temperature level. Further results using different parameters of **stress -c Options** can be found in Appendices section. From above results, the parameters which are identified in experiment are number of stress CPU(s) and time interval for recording NUMA node(s) temperature. Therefore above experiments conclude Table 5.1 which depict the average, lowest and highest temperature recorded

Table 5.1: NUMA node(s) temperature with distinct no. of stress CPU parameters

in Celsius degree	24 CPU(s) stress approx.	36 CPU(s) stress approx.	48 CPU(s) stress approx.
highest temp	52	58	60
lowest temp	40	45	48
average temp	43	46	52

after an interval of 1 minute.

Therefore specific parameters will be used in order to compare the results. In testing server, there are 48 CPU(s), so stressing half CPU(s) 24 will work out for changing affinity among them. For this project, workload will be generated using ***stress -c 24*** command and the server will choose 24 CPU(s) randomly from NUMA node(s).

5.2 Experiment 1: High to Low

In this experiment, ant will search for high temperature node and low temperature node and will change the affinity of processes between them. When the workload will be generated from start, there is a possibility that corresponding CPU(s) of one node has more processes inside so if all processes will move to low temperature node that node will become overloaded.

To avoid this scenario there is decision made check which identifies the number of processes from high temperature node and move only few processes based on threshold i.e. difference between highest temperature node and lowest temperature node.

After generating the proposed workload, the stressed temperature of NUMA node(s) has been recorded with interval of 1 minute to check the high,low and average level.

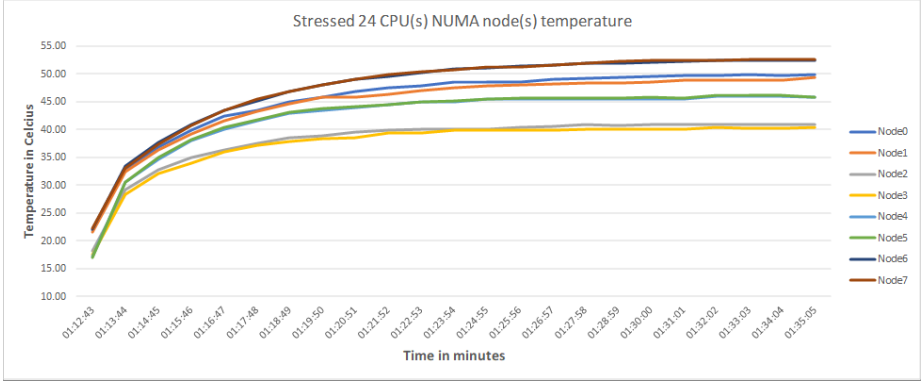

Figure 5.3: NUMA node(s) 24 CPU(s) stressed

Sample Standard Deviation, s	6.594674309548
Variance (Sample Standard), s^2	43.489729249012
Population Standard Deviation, σ	6.449718711353
Variance (Population Standard), σ^2	41.598871455577
Total Numbers, N	23
Sum:	990.44
Mean (Average):	43.062608695652
Standard Error of the Mean ($SE_{\bar{x}}$):	1.3750846495526

Figure 5.4: Online Standard Deviation Calculator [45] is used to show the Standard deviation properties of average temperatures of NUMA node(s) without implementation of algorithm

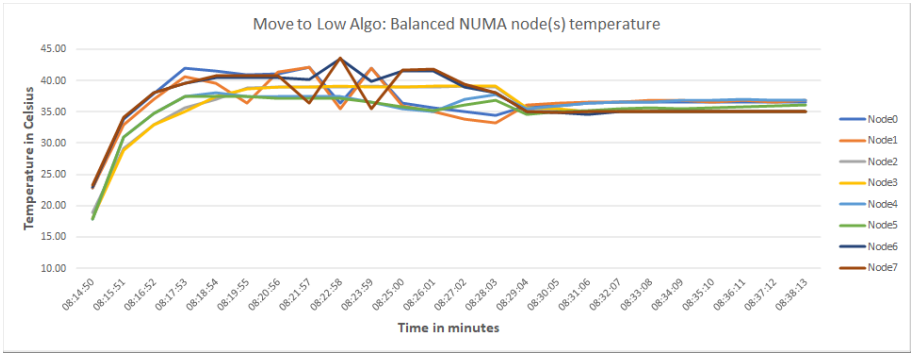

Figure 5.5: The figure shows the line chart of NUMA node(s) temperature with implementation of modified ACO algorithm

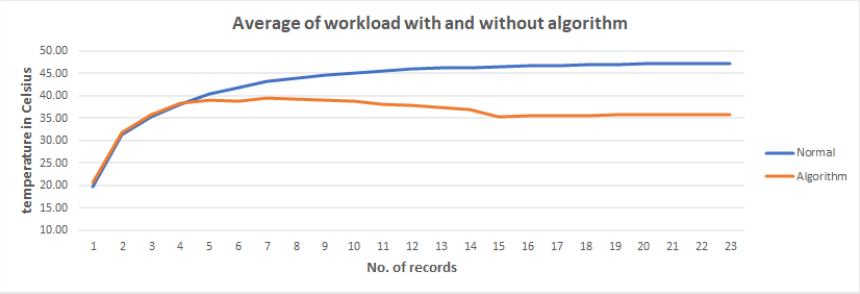

Figure 5.6: The figure shows the line chart of average temperatures of NUMA node(s) with and without implementation of modified high to low ACO algorithm

Sample Standard Deviation, s	3.8706094340949
Variance (Sample Standard), s^2	14.981617391304
Population Standard Deviation, σ	3.7855307054781
Variance (Population Standard), σ^2	14.330242722117
Total Numbers, N	23
Sum:	831.93
Mean (Average):	36.170869565217
Standard Error of the Mean ($SE_{\bar{x}}$):	0.80707785819408

Figure 5.7: Online Standard Deviation Calculator [45] is used to show the Standard deviation properties of average temperatures of NUMA node(s) based on modified high to low algorithm

5.3 Analysis

Figure 5.3 shows the NUMA node(s) temperature of 24 CPU(s) stressed without implementation of algorithm. The difference in node(s) temperature is due to baseline of node(s) value i.e. node(s) 2,3,4 and 5 have baseline temperature around 18°C whereas node(s) 0,1,6 and 7 have baseline temperature around 22°C. All the node(s) are stabilized after some time where highest temperature reach around 52°C and lowest temperature is approximately 40°C so the average temperature will be around 43°C.

The distributional characteristics of both records as well as the level of temperature can be seen in Figure 5.8. From Figure 5.8, we can extract distributional data which is represented in Table 5.2.

This table 5.2 represent that interquartile range (IQR) which is the distance between the 1st and 3rd quartiles (Q1 and Q3) of normal data i.e. stressed 24 CPU(s) of NUMA node(s) **without** implementation of algorithm is **4.59**. whereas interquartile range (IQR) of algorithm data i.e. stressed 24 CPU(s) of NUMA node(s) **with** implementation of algorithm data is **3.04**.

Min represents the lower outer fence whereas max represents the upper inner fence. The shape of distribution of Normal data is skewed left means that data is concentrated towards upper end of values whereas shape of distribution of algorithm data is skewed right means that data more is concentrated towards lower end of values.

Table 5.2: Distributional analysis of Figure 5.8

	Q1	median (Q2)	Q3	lower fence	min	max
Normal	42.24	45.96	46.83	38.23	19.64	47.19
Algorithm	35.66	35.80	38.70	31.84	20.54	39.54

From Figure 5.9, we can analyze how much data i.e. Normal and Algorithm are spread out around the mean. From Figure 5.9, we can extract distributional data which is represented in Table 5.3.

This table 5.3 represent that interquartile range (IQR) of average normal data i.e. stressed 24 CPU(s) of NUMA node(s) **without** implementation of algorithm is **1.46**. whereas interquartile range (IQR) of average algorithm data i.e. stressed 24 CPU(s) of NUMA node(s) **with** implementation of algorithm data is **1.57**.

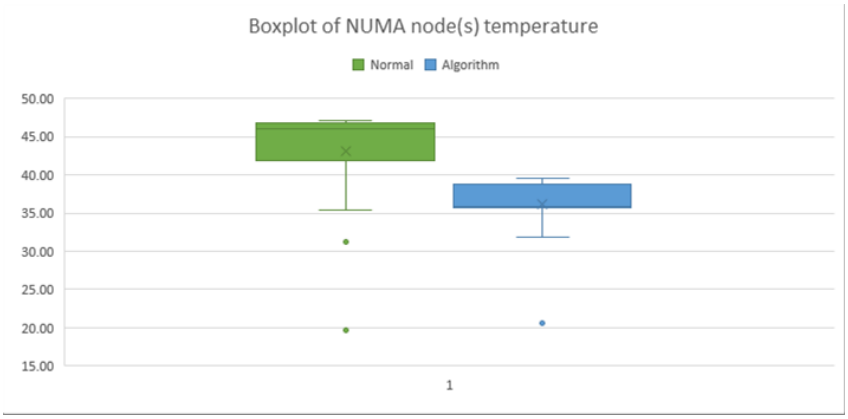

Figure 5.8: The figure shows the boxplot of average temperatures of NUMA node(s) Normal data represents without implementation of algorithm and Algortihm represents the implementation of modified high to low ACO algorithm

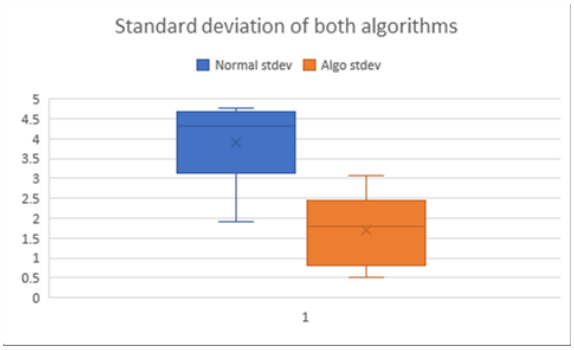

Figure 5.9: The figure shows the boxplot standard deviation of average temperatures of NUMA node(s).

Table 5.3: Distributional analysis of Figure 5.9

	Q1	median (Q2)	Q3	min	max
Normal	3.21	4.31	4.67	1.92	4.76
Algorithm	0.80	1.78	2.37	0.51	3.07

5.4 Comparison: High to Low

Other statistical data such as Population Standard Deviation, Sample Standard Deviation, Variance and Confidence Interval (CI) are explained in Figures 5.11, 5.10. Figures 5.11 shows that Confidence interval of 68% of temperatures lies within 1 standard deviation of the mean i.e. within approximately 41 to 44 degree Celsius. Whereas in modified ACO algorithm Figures 5.10 shows that 68% of temperature lies within 1 standard deviation of the mean i.e. within approximately 35 to 36 degree Celsius. Therefore the modified ACO algorithm decreases the temperature and balanced it to provide efficient usage of energy.

Confidence Level	Range
68.3%, $SE_{\bar{x}}$	35.363791707023 - 36.977947423411
90%, 1.645$SE_{\bar{x}}$	34.843226488488 - 37.498512641947
95%, 1.960$SE_{\bar{x}}$	34.588996963157 - 37.752742167278
99%, 2.576$SE_{\bar{x}}$	34.091837002509 - 38.249902127925
99.9%, 3.291$SE_{\bar{x}}$	33.514776333901 - 38.826962796534
99.99%, 3.891$SE_{\bar{x}}$	33.030529618984 - 39.311209511451
99.999%, 4.417$SE_{\bar{x}}$	32.606006665574 - 39.735732464861
99.9999%, 4.892$SE_{\bar{x}}$	32.222644682932 - 40.119094447503

Figure 5.10: Online Standard Deviation Calculator [45] is used to show the Confidence Interval and Range of average temperatures of NUMA node(s) based on modified high to low algorithm

Figures 5.11 shows that Confidence interval of 99.7% of temperature lies within 3 standard deviation of the mean i.e. within approximately 38 to 47 degree Celsius. Whereas in modified ACO algorithm Figures 5.10 shows that 99.7% of temperature lies within 3 standard deviation of the mean i.e. within approximately 33 to 38 degree Celsius.

Confidence Level	Range
68.3%, $SE_{\bar{x}}$	41.6875240461 - 44.437693345205
90%, 1.645$SE_{\bar{x}}$	40.800594447138 - 45.324622944166
95%, 1.960$SE_{\bar{x}}$	40.367442782529 - 45.757774608775
99%, 2.576$SE_{\bar{x}}$	39.520390638405 - 46.6048267529
99.9%, 3.291$SE_{\bar{x}}$	38.537205113974 - 47.58801227733
99.99%, 3.891$SE_{\bar{x}}$	37.712154324243 - 48.413063067061
99.999%, 4.417$SE_{\bar{x}}$	36.988859798578 - 49.136357592726
99.9999%, 4.892$SE_{\bar{x}}$	36.335694590041 - 49.789522801264

Figure 5.11: Online Standard Deviation Calculator [45] is used to show the Confidence Interval and Range of average temperatures of NUMA node(s) without implementation of algorithm

5.5 Experiment 2: Move to Next Node

In this experiment, after the workload is being generated using stress tool, ant will change the affinity of current node to next node without comparing to the next node temperature. All of processes from current node will move to next node whether temperature is high, low or equal on next node. When the processes moved to last node, it will change the affinity to first node. In this scenario there is no threshold or no self decision made check which identifies the number of processes from high temperature node and move only few processes.

The generated workload uses 24 CPU(s) with ***stress -c 24*** command. The stressed NUMA node(s) temperature has been recorded with interval of 1 minute to check the high,low and average level.

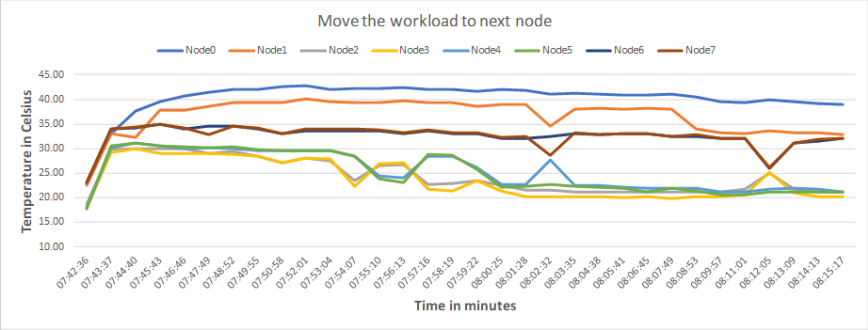

Figure 5.12: This figure shows the line chart of NUMA node(s) temperature after implementation of move to next algorithm.

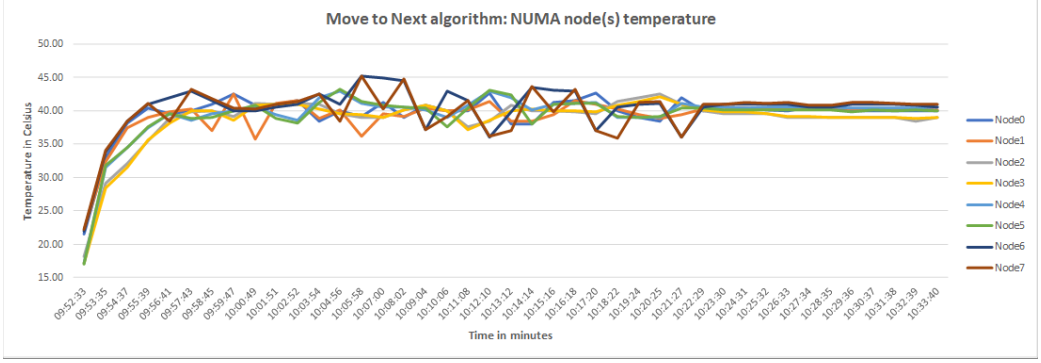

Figure 5.13: This figure shows the line chart of NUMA node(s) temperature while moving processes to next node after comparing the next node temperature.

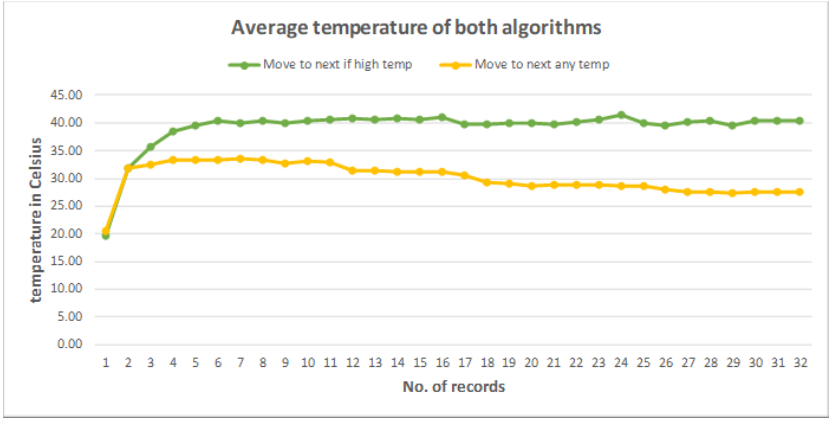

Figure 5.14: The figure shows the line chart of average temperatures of NUMA node(s) of both algorithms i.e. move to next node if current node is high and move to next without comparing to next node.

5.6 Analysis

Figure 5.12 shows the NUMA node(s) temperature of 24 CPU(s) stressed based on implementation of algorithm which move the processes from current node to next node whether temperature is equal, high or low, . The difference in node(s) temperature is due to baseline of node(s) value i.e. node(s) 2,3,4 and 5 have baseline temperature around 18°C whereas node(s) 0,1,6 and 7 have baseline temperature around 22°C. All the node(s) are stabilized after some time where highest temperature reach around 40°C and lowest temperature is approximately 21°C.

Figure 5.16 shows the balanced workload on 24 CPU(s) of NUMA node(s) based on implementation of algorithm which move the processes from current node to next node if and only if temperature is low on next node. If temperature is high on last node, it will compare with first node and move the processes if first node is on low temperature. The difference in node(s) temperature is due to baseline of node(s) value i.e. node(s) 2,3,4 and 5 have baseline temperature around 18°C whereas node(s) 0,1,6 and 7 have baseline temperature around 22°C. All the node(s) are balanced after some time where homogeneous temperature is around 40°C.

The distributional characteristics of both records as well as the level of temperature can be seen in Figure 5.17. From Figure 5.17, we can extract distributional data which is represented in Figure 5.15, 5.16 and Table 5.4.

This table 5.4 represent that interquartile range (IQR) which is the distance between the 1st and 3rd quartiles (Q1 and Q3) of move to next algorithm with-

out comparing next node temperature is **4.3** whereas interquartile range (IQR) of move to next node if current node is high algorithm is **0.73**.

Table 5.4: Distributional analysis of Figure 5.17

	Q1	median (Q2)	Q3	lower fence	min	max
Move2Next Any temp	28.30	29.91	32.60	27.31	20.39	33.64
Move2Next if High	39.78	40.24	40.51	39.46	19.66	41.40

Sample Standard Deviation, s	2.8162331675542
Variance (Sample Standard), s^2	7.9311692540323
Population Standard Deviation, σ	2.7718802670469
Variance (Population Standard), σ^2	7.6833202148438
Total Numbers, N	32
Sum:	960.55
Mean (Average):	30.0171875
Standard Error of the Mean ($SE_{\bar{x}}$):	0.49784439254501

Figure 5.15: Online Standard Deviation Calculator [45] is used to show the Standard deviation characteristics of average temperatures of NUMA node(s) based on move to next algorithm without comparing next node temperature

Sample Standard Deviation, s	3.9713042770972
Variance (Sample Standard), s^2	15.77125766129
Population Standard Deviation, σ	3.908760143495
Variance (Population Standard), σ^2	15.278405859375
Total Numbers, N	32
Sum:	1253.3
Mean (Average):	39.165625
Standard Error of the Mean ($SE_{\bar{x}}$):	0.70203404612264

Figure 5.16: Online Standard Deviation Calculator [45] is used to show the Standard deviation characteristics of average temperatures of NUMA node(s) based on move to next node algorithm if current node is on high temperature

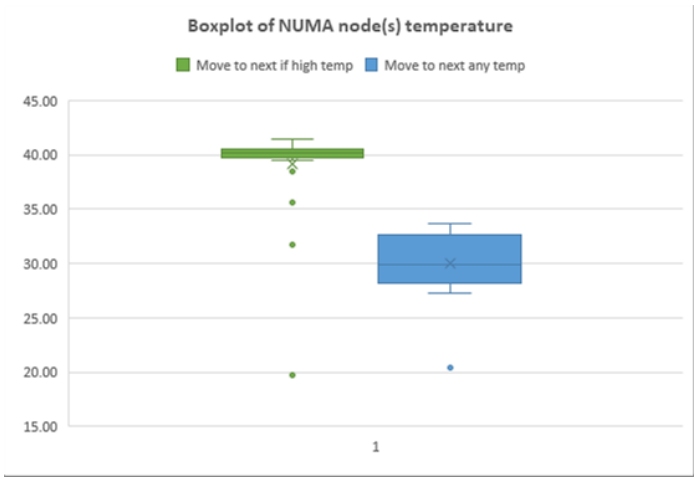

Figure 5.17: The figure shows the box chart of standard deviation of average temperatures of NUMA node(s) of both algorithms i.e. move to next node if current node is high and move to next without comparing to next node.

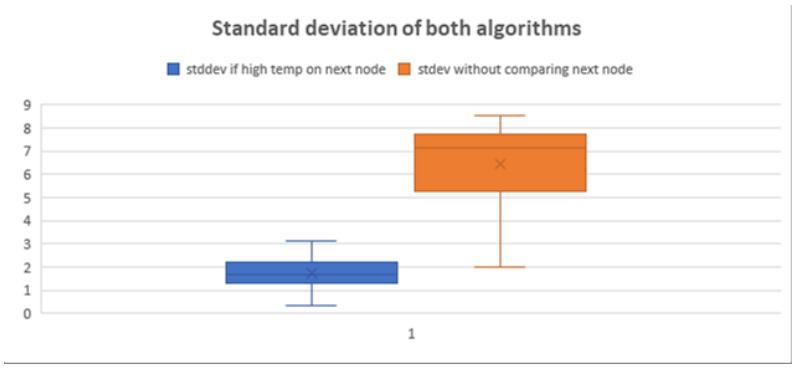

Figure 5.18: The figure shows the box chart of standard deviation of average temperatures of NUMA node(s)of both algorithms i.e. move to next node if current node is high temperature and move to next without comparing next node temperature.

5.7 Comparison: Move to Next

Other statistical data such as Population Standard Deviation, Sample Standard Deviation, Variance and Confidence Interval (CI) are explained in Figures 5.15, 5.16, 5.19 and 5.20.

Figures 5.19 shows that Confidence interval of 68% of temperature lies within 1 standard deviation of the mean i.e. within approximately 29 to 30 degree Celsius temperature. Whereas move to next node algorithm if current node is on high temperature Figures 5.20 shows that 68% of temperature lies within 1 standard deviation of the mean i.e. within approximately 38 to 39 degree Celsius temperature.

Figures 5.19 shows that Confidence interval of 99.7% of temperature lies within 3 standard deviation of the mean i.e. within approximately 28 to 31 degree Celsius. Whereas in move to next node algorithm if current node is on high temperature Figures 5.20 shows that 99.7% of temperature lies within 3 standard deviation of the mean i.e. within approximately 36 to 41 degree Celsius.

Confidence Level	Range
68.3%, $SE_{\bar{x}}$	29.519343107455 - 30.515031892545
90%, 1.645$SE_{\bar{x}}$	29.198233474263 - 30.836141525737
95%, 1.960$SE_{\bar{x}}$	29.041412490612 - 30.992962509388
99%, 2.576$SE_{\bar{x}}$	28.734740344804 - 31.299634655196
99.9%, 3.291$SE_{\bar{x}}$	28.378781604134 - 31.655593395866
99.99%, 3.891$SE_{\bar{x}}$	28.080074968607 - 31.954300031393
99.999%, 4.417$SE_{\bar{x}}$	27.818208818129 - 32.216166181871
99.9999%, 4.892$SE_{\bar{x}}$	27.58173273167 - 32.45264226833

Figure 5.19: Online Standard Deviation Calculator [45] is used to show the Confidence Interval and Range of average temperatures of NUMA node(s) based on move to next algorithm without comparing next node temperature

Confidence Level	Range
68.3%, $SE_{\bar{x}}$	38.463590953877 - 39.867659046123
90%, 1.645$SE_{\bar{x}}$	38.010778994128 - 40.320471005872
95%, 1.960$SE_{\bar{x}}$	37.7896382696 - 40.5416117304
99%, 2.576$SE_{\bar{x}}$	37.357185297188 - 40.974064702812
99.9%, 3.291$SE_{\bar{x}}$	36.85523095421 - 41.47601904579
99.99%, 3.891$SE_{\bar{x}}$	36.434010526537 - 41.897239473463
99.999%, 4.417$SE_{\bar{x}}$	36.064740618276 - 42.266509381724
99.9999%, 4.892$SE_{\bar{x}}$	35.731274446368 - 42.599975553632

Figure 5.20: Online Standard Deviation Calculator [45] is used to show the Confidence Interval and Range of average temperatures of NUMA node(s) based on move to next node algorithm if current node is on high temperature

Chapter 6

Discussion

The objective of this chapter is to discuss the obtained results and related to the problem statement.

Initial experiments start for the tuning of parameters and check the maximum, minimum and average temperature of NUMA node(s). Instead of NUMA node(s) temperatures starting from zero, the temperature baseline is different on every node which is useful information in order to subtract or add temperature into final result if required. Knowing the limitations of testing server CPU sensors is very much important regarding final results analysis and comparisons.

By giving different parameters of stress on CPU(s), we found out that the highest and lowest temperature of node(s) varies along with number of processes. If number of processes generated by stress tool increases the temperature of node(s) will increase, so we can say that NUMA node(s) temperature is directly proportional to number of processes generated by stress command.

After performing preliminary experiments and choosing right parameters for our proposed algorithm, we record the node(s) temperature with specific *-c options* of stress workload generator. The specific parameters plays a vital role so that the final results will be compared with same set of records and same time interval.

Usually the servers NUMA node(s) CPU(s) have some virtual CPU(s) which means every CPU have possibility of its sibling CPU, so if one CPU gets hot its sibling CPU also gets hot. The NUMA node(s) CPU(s) siblings and corresponding node has been described in earlier section. While giving stress the CPU siblings also considered to avoid all processes moved to low temperature node.

Another important aspect is when to move the processes to low temperature NUMA node. So we define some threshold in our algorithms which checks every interval of time if current temperature is higher than threshold then the pro-

cesses will change its affinity otherwise it will go for another interval.

6.1 Use Case I: High to Low

In this test, we can see that initially the workload generated randomly at start which is considered as static load balancing. After running algorithm, it takes some time to have equal processes on all NUMA node(s). The time it takes to have homogeneous temperature on all NUMA node(s) depends on the number of processes.

In comparing section of results chapter, we look at the results after generating workload at initial level of both experiments the one which just record the temperature and the second which runs the algorithm. Th 99.7% confidence interval (CI) of just record experiment is lies within 3 standard deviation of the mean i.e. within approximately 38 to 47 degree Celsius. Whereas in modified ACO algorithm results shows that 99.7% confidence interval (CI) of temperature lies within 3 standard deviation of the mean i.e. within approximately 33 to 38 degree Celsius.

So the modified ACO algorithm gives lower temperature to avoid over heating of servers CPU(s). From line charts it indicates that all NUMA node(s) have same level of temperature. In case of algorithm it takes approximately 15 minutes to have homogeneous temperature with 24 number of processes. The average mean of just record experiment is around 43 degree Celsius whereas the average mean of modified ACO algorithm experiment is around 36 degree Celsius. The difference between average means has huge gap which indicates that modified algorithm efficiently self managed to have balanced temperature lower than normal behaviour of servers CPU sensors temperature.

6.2 Use Case II: Move to Next

In this experiment, same scenario of time interval and workload and has been generated using stress workload generator tool. Two different algorithm have been implemented in this test instead of just recording normal behaviour as we did in High to Low use case. First algorithm move all the processes to next node regardless of temperature on next node in forward direction. Second algorithm move the processes if and only if next node temperature is lower than current node.

Based on specified threshold, self made decision has been taken to avoid moving all processes. In comparing section of results chapter, we look at the results after generating workload at initial level of both experiments. The 99.7% confidence interval (CI) of first experiment lies within 3 standard deviation of

the mean which is between 28 to 31 degree Celsius approximately. Whereas the second experiment which move the processes to next node if and only if current node is on high temperature shows that 99.7% confidence interval (CI) of temperature lies within 3 standard deviation of the mean which is between 37 to 41 degree approximately.

From line charts and statistical distribution tool, we can see that even the second experiment levels the NUMA node(s) temperature its average mean 39 degree Celsius approximately. And in case of first experiment, the average mean is 30 degree Celsius approximately which is much lower than proposed Move to Next algorithm. Thus the lower balanced temperature criteria doesn't fulfill our requirement, there is further modification in Move to Next algorithm is needed regarding time interval, number of processes move and when to move the processes.

The difference between Move to Next and High to Low experiments of balanced temperature indicates that modified algorithm of High to Low has better performance which efficiently self managed to have balanced temperature lower than balanced algorithm of Move to Next.

Relation to Problem statement phrase "Energy Utilization at Data Centers"

As proposed earlier in the problem statement the modified High to Low ACO algorithm efficiently utilize energy of servers at data centers which enables green cloud computing environment. Internet of Things (IoT) based CPU sensors plays vital role to have an overlook of environmental conditions of servers hardware at data centers.

Relation to Problem statement phrase "Intelligent Algorithms"

Nowadays where dynamic applications and 24/7 online services are everywhere, therefore consumers avoid traditional algorithms because these algorithms uses centralized systems which causes whole systems down if they encounter single point of failure. Therefore, decentralized algorithms such as swarm intelligence methods are useful for current era dynamic online applications.

In this project, the modified version of ant colony optimization algorithm which is member of swarm intelligence methods implemented where Self-Optimization features has been used which monitors CPU sensor temperature and shared the resources based on threshold and time interval parameters of our algorithm.

Chapter 7

Conclusion and Future Work

In this paper, two algorithms have been proposed based on ant colony optimization for efficient energy utilization at data centers. The generated workload was based on static load balancing strategies which was applied on both algorithms with high to low and move to next mechanism. Experiment results were illustrated with the help of charts and statistical distribution tools. The results showed that proposed high to low mechanism performed better and gives homogeneous NUMA node(s) temperature than move to next algorithm technique.

Although in load balancing static algorithms are more suitable for homogeneous and stable environments but with rapid growth of dynamic services at cloud infrastructures there is a need of dynamic algorithms. So future work can be done by generating workload on run-time environment to improve the efficient usage of energy at data centers. In this project, response time was fix this parameter can further increase or decrease for dynamic applications scenario.

After workload has been generated system decides the CPU(s) for setting infinity, so one can bind the workload on specific CPU(s) of NUMA node(s) before generating workload. Furthermore, the move to next algorithm was proposed only in forward direction, so it can be further explored in backward direction and neighbour nodes based on threshold and number of processes. After high temperature node or low temperature node have been marked, the random movement of processes can be further explored based on high to low algorithm.

Furthermore, live migration of virtual machines (VMs) can be looked after the discovery of overloaded or underload NUMA node(s). CPU sensors temperature have been explored in this project, there are also other system's hardware sensors such as Core sensors, thermal sensors and power sensors which can be implemented in future work with our proposed algorithm . In future, we will investigate how to implement other intelligent algorithms using our approach to enable green cloud computing environment and efficient utilization of energy at data centers.

Chapter 8

Appendices

```
#!/bin/bash

stemp=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5)
echo String is: $stemp

arr+=(${stemp// / })
time=$(date '+%H:%M:%S');
echo "$time"

sum=0
arr=($time)
arr+=(${stemp// / })
echo "Array length: "${#arr[@]}

for item in ${arr[@]}
do
        #echo Node $sum temp is: $item
        printf "%s" "${arr[$sum]}" $'\t' >> f2high2low.txt
        let sum=sum+1
done

printf "%s" $'\n' >> f2high2low.txt

{ printf 'Time\tNode0\tNode1\tNode2\tNode3\tNode4\tNode5\tNode6\
    tNode7\n'; cat f2high2low.txt; } | tr "\\t" "," > /home/sufi/
    f2highTolow.csv

# Find low temp Node number
node=0
minTemNode=0
minTemVal=0.0
lowVal=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 |
    sort -n | head -1)
for i in ${arr[@]:1}
do
        #nodeTem=${arr[$node+1]}
        #echo NodeTem: $i
        if [ "$(echo "$i == $lowVal" | bc)" -eq 1 ]; then
```

```
36                  echo Min Matches: $i = $lowVal Running on Node:
    $node
37                  minTemNode=$node
38                  minTemVal=$i
39          break
40          fi
41          let node=node+1
42  done
43
44  # Find high temp node number
45  node=0
46  maxTemNode=0
47  maxTemVal=0.0
48  highVal=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 |
    sort -n | tail -1)
49  last3hVal=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 |
     sort -n | tail -1)
50
51  maxValarr=(${last3hVal// / })
52
53  diff="$(echo  "$highVal - $lowVal" | bc)"
54  echo $diff
55
56  if [ $(echo "$diff < "2.7"" | bc) -eq 1 ]; then
57     echo "$(tput setaf 3)Not Much difference$(tput sgr0)"
58  else
59  for i in ${arr[@]:1}
60  do
61     maxTemVal=$i
62     c=1
63     for max in ${maxValarr[@]}
64     do
65        #echo "$i >= $max"
66        if [ "$(echo "$i == $max" | bc)" -eq 1 ]; then
67          echo Max Matches: $i = $max Running on Node: $node
68          maxTemNode=$node
69          maxTemVal=$i
70          #echo MaxTemNode: $maxTemNode
71          pids=$(pidof stress)
72          set -f
73          pidsArr=($pids)
74          set +f
75          #c=1
76          #echo "PIDs Array length: "${#pidsArr[@]}
77          for item in ${pidsArr[@]}
78          do
79                  cpuNo=$(cat /proc/$item/stat | cut -d' ' -f39)
80                  echo Running process[$item] on: $cpuNo
81                  loop=1
82                  (( sum = $loop + $node ))
83                  # echo Sum is: $sum
84                  sen=$(numactl --hardware | grep cpus | head -$sum |
     tail -1 )
85                  var=${sen#*:}
86                  sinCpuArr=($var)
87                  #echo Node $node contains CPU: $var
88                  #c=1
```

```
                for citem in ${sinCpuArr[@]}
                do

                        #taskset -p -c ${nCpuArr[$rand]} $item
                        #echo Node $node CPU is: $citem
                        if [[ $cpuNo -eq $citem ]]; then

                            if [ $(echo "$diff < "4.7"" | bc) -eq 1 ]; then
                                echo "$(tput setaf 3)lt :$c$(tput sgr0)"
                                (( c = $c + 1 ))
                            fi
                            if [ $(echo "$c < "7""| bc) -eq 1 ]; then
                                echo Counter: $c
                                (( c = $c + 1 ))
                                echo "$(tput setaf 3)Running PID[$item] on CPU $cpuNo lies in Node: $maxTemNode$(tput sgr0)"
                                # Moving running process to low temperature node
                                (( s = $minTemNode + $loop ))
                                lowNodeCpu=$(numactl --hardware | grep cpus | head -$s | tail -1 )
                                varl=${lowNodeCpu#*:}
                                nCpuArr=($varl)
                                #for fitem in ${nCpuArr[@]}
                                #do
                                        echo "$(tput setaf 5)Node $minTemNode contains CPU: $varl$(tput sgr0)"
                                        rand=$[ $RANDOM % 6 ]
                                        taskset -p -c ${nCpuArr[$rand]} $item
                                #done
                            fi
                        fi
                done
            done
            break
        fi
    done
    let node=node+1
done
fi
```

Listing 8.1: Bash scripting to change affinity from highest NUMA node to lowest NUMA node

```

#!/bin/bash

stemp=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5)

echo String is: $stemp

arr+=(${stemp// / })

```

```
time=$(date '+%H:%M:%S');

echo "$time"

sum=0

arr=($time)

arr+=(${stemp// / })

echo "Array length: "${#arr[@]}

for item in ${arr[@]}
do
        #echo Node $sum temp is: $item
         printf "%s" "${arr[$sum]}" $'\t' >> justrecord.txt
         let sum=sum+1
done

printf "%s" $'\n' >> justrecord.txt

{ printf 'Time\tNode0\tNode1\tNode2\tNode3\tNode4\tNode5\tNode6\
    tNode7\n'; cat justrecord.txt; } | tr "\\t" "," > /home/sufi/
    justrecord.csv
```

Listing 8.2: Bash Scripting to just record temperature after workload generation

```
#!/bin/bash

stemp=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5)

arr+=(${stemp// / })
time=$(date '+%H:%M:%S');
echo "$time"

sum=0
arr=($time)
arr+=(${stemp// / })
echo "Array length: "${#arr[@]}

for item in ${arr[@]}
do
        #echo Node $sum temp is: $item
         printf "%s" "${arr[$sum]}" $'\t' >> lmove2next.txt
         let sum=sum+1
done

printf "%s" $'\n' >> lmove2next.txt

{ printf 'Time\tNode0\tNode1\tNode2\tNode3\tNode4\tNode5\tNode6\
    tNode7\n'; cat lmove2next.txt; } | tr "\\t" "," > /home/sufi/
    lmoveToNext.csv

# Find low temp Node number
node=0
lowVal=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 |
    sort -n | head -1)
```

```
highVal=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 |
    sort -n | tail -1)

diff="$(echo  "$highVal - $lowVal" | bc)"

if [ $(echo "$diff < "3.3"" | bc) -eq 1 ]; then
   echo "$(tput setaf 3)Not Much difference$(tput sgr0)"
else
for i in ${arr[@]:1}
do
        #pids=$(pidof "qemu-system-x86_64")
        pids=$(pidof stress)
        set -f
        pidsArr=($pids)
        set +f
        c=1
        echo "PIDs Array length: "${#pidsArr[@]}
        for item in ${pidsArr[@]}
        do
                cpuNo=$(cat /proc/$item/stat | cut -d' ' -f39)
                #echo Running process[$item] on: $cpuNo
                loop=1
                (( sum = $loop + $node ))
                # echo Sum is: $sum
                sen=$(numactl --hardware | grep cpus | head -$sum |
     tail -1 )
                var=${sen#*:}
                sinCpuArr=($var)
                #echo Node $node contains CPU: $var
                for citem in ${sinCpuArr[@]}
                do

                        #taskset -p -c ${nCpuArr[$rand]} $item
                        #echo Node $node CPU is: $citem
                    if [[ $cpuNo -eq $citem ]]; then

                        if [ $(echo "$c < "10""| bc) -eq 1 ];  then
                           echo Counter: $c
                           (( c = $c + 1 ))
                           nextVal=${arr[$(($node+2))]}
                           last=${arr[${#arr[@]}-1]}
                           (( s = $node + 2 ))
                           if [ -z "${nextVal}" ]; then
                             echo "$(tput setaf 2)Last Val$(tput
    sgr0)"
                             nextVal=${arr[1]}
                             s=1
                             #echo 1st val:$nextVal
                           fi
                           #if (( $(echo "$i == $nextVal" | bc -l)
    )); then
                           #   echo "$(tput setaf 3)Current Val is
    Equal$(tput sgr0)"
                #Do nothing
                           if (( $(echo "$i <= $nextVal" | bc -l) )
    ); then
                              echo "$(tput setaf 4)Current Val is
```

```
    low$(tput sgr0)"

                        else
                echo "$(tput setaf 3)Running PID[$item] on CPU
    $cpuNo lies in Node: $node$(tput sgr0)"
                # Moving running process to low temperature node

                #echo Next Node is: $s
                lowNodeCpu=$(numactl --hardware | grep cpus | head
    -$s | tail -1 )
                varl=${lowNodeCpu#*:}
                nCpuArr=($varl)
                #echo "$(tput setaf 5)Node $s contains CPU: $varl$(
    tput sgr0)"
                rand=$[ $RANDOM % 6 ]
                taskset -p -c ${nCpuArr[$rand]} $item
                        fi
                    fi
                fi
                done
        done
   let node=node+1
   done
fi
```

Listing 8.3: Move to next node if current node is high temperature

```
#!/bin/bash

node=0
minTemNode=0
minTemVal=0.0
lowVal=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 |
    sort -n | head -1)
for i in ${arr[@]:1}
do
        #nodeTem=${arr[$node+1]}
        #echo NodeTem: $i
        if [ "$(echo "$i == $lowVal" | bc)" -eq 1 ]; then
                echo Min Matches: $i = $lowVal Running on Node:
    $node
                minTemNode=$node
                minTemVal=$i
        break
        fi
        let node=node+1
done

# Find high temp node number
node=0
maxTemNode=0
maxTemVal=0.0
highVal=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 |
    sort -n | tail -1)
last3hVal=$(sensors | grep 'temp1' | awk '{print $2}' | cut -c2-5 |
     sort -n | tail -2)

maxValarr=(${last3hVal// / })
```

```
28 echo "Max Val Array length: "${#maxValarr[@]}
29
30 echo "$(tput setaf 5)High Temp Node is: $maxTemNode and Value is:
       $highVal$(tput sgr0)"
31 echo "$(tput setaf 4)Low Temp Node is: $minTemNode and Value is:
       $minTemVal$(tput sgr0)"
32 diff="$(echo  "$highVal - $lowVal" | bc)"
33 echo $diff
34
35 if [ $(echo "$diff < 3.2" | bc) -eq 1 ]; then
36    echo "$(tput setaf 3)Not Much difference$(tput sgr0)"
37 else
38 for i in ${arr[@]:1}
39 do
40    maxTemVal=$i
41    for max in ${maxValarr[@]}
42    do
43       #echo "$i >= $max"
44       if [ "$(echo "$i == $max" | bc)" -eq 1 ]; then
45         echo Max Matches: $i = $max Running on Node: $node
46         maxTemNode=$node
47         maxTemVal=$i
48         #echo MaxTemNode: $maxTemNode
49         #pids=$(pidof "qemu-system-x86_64")
50         pids=$(pidof stress)
51         set -f
52         pidsArr=($pids)
53         set +f
54         c=1
55         echo "PIDs Array length: "${#pidsArr[@]}
56         for item in ${pidsArr[@]}
57         do
58                 cpuNo=$(cat /proc/$item/stat | cut -d' ' -f39)
59                 echo Running process[$item] on: $cpuNo
60                 loop=1
61                 (( sum = $loop + $node ))
62                 # echo Sum is: $sum
63                 sen=$(numactl --hardware | grep cpus | head -$sum |
       tail -1 )
64                 var=${sen#*:}
65                 sinCpuArr=($var)
66                 #echo Node $node contains CPU: $var
67
68                 for citem in ${sinCpuArr[@]}
69                 do
70
71                         #taskset -p -c ${nCpuArr[$rand]} $item
72                         #echo Node $node CPU is: $citem
73                         if [[ $cpuNo -eq $citem ]]; then
74
75                            if [ $(echo "$c < 4"| bc) -eq 1 ];  then
76                              echo Counter: $c
77                              (( c = $c + 1 ))
78                              #echo C Check: $c
79                              echo "$(tput setaf 3)Running PID[
       $item] on CPU $cpuNo lies in Node: $maxTemNode$(tput sgr0)"
80                              # Moving running process to next
```

```
    node
                                    (( s = $minTemNode + $loop ))
                                    lowNodeCpu=$(numactl --hardware |
    grep cpus | head -$s | tail -1 )
                                    varl=${lowNodeCpu#*:}
                                    nCpuArr=($varl)
                                    echo "$(tput setaf 5)Node
    $minTemNode contains CPU: $varl$(tput sgr0)"
                                    rand=$[ $RANDOM % 6 ]
                                    taskset -p -c ${nCpuArr[$rand]}
    $item
                                fi
                            fi
                    done
            done
            break
        fi
    done
    let node=node+1
done
fi
```

Listing 8.4: Bash Scripting: Move to next node regardless of next node temperature

```
#! /bin/bash

echo "Below are the Nodes and corresponding CPU's architecture"
numactl --hardware | grep cpu
echo $'\n'
echo -e  "Press 0 to run VM's randomly\nPress 1 to run VM's on
    Nodes Self Increasing\nPress 2 to run VM's on Specific Node"
read input
case "$input" in
  "0")
    echo "Randomly"
    echo -e  "Enter no. of VM's randomly"
    read vm
    #killall -9 qemu-system-x86_64 2> /dev/null
    node=0
    COUNTER=1
    while [ Your != "47" ]
    do
        for i in $(seq 1 $vm)
        do
                #qemu-system-x86_64 --enable-kvm -kernel
    chainloader -initrd calcDouble.img -m 16 -nographic &> /dev/
    null&
                stress -c 24 &
        done
        for i in $(seq 1 "120")
        do

            echo "Round: $COUNTER"
            # Random number generator code

            seconds=60; date1=$((`date +%s` + $seconds));
```

```
30
31              while [ "$date1" -ge `date +%s` ]; do
32                  timerun="$(date -u --date @$(($date1 - `date +%s` )
    ) +%H:%M:%S)\r"
33                  echo -ne $timerun;
34              done
35
36              #echo "********* Running Run3 file **********"
37              #/bin/bash ./run3
38              #/bin/bash ./high2low
39              #/bin/bash ./backuph2lFile
40              #/bin/bash ./storeTofile
41              /bin/bash ./lmove2next
42              #killall -9 qemu-system-x86_64 2> /dev/null
43              COUNTER=$[$COUNTER +1]
44          done
45      node=0
46      COUNTER=$[$COUNTER +1]
47      done
48      ;;
49
50  "1")
51      echo "Self Increasing Node"
52
53      node=0
54      killall -9 qemu-system-x86_64 2> /dev/null
55      COUNTER=1
56      while [ Your != "47" ]
57
58      do
59      echo "Round: $COUNTER"
60
61      for i in $(seq 1 "8")
62      do
63                  echo "VM's Started run on Node: $node"
64                  for i in $(seq 1 "6")
65                  do
66                  numactl --cpunodebind=$node qemu-system-x86_64 --
    enable-kvm -kernel chainloader -initrd calcDouble.img -m 16 -
    nographic &> /dev/null&
67                  done
68                  seconds=90; date1=$((`date +%s` + $seconds));
69                  while [ "$date1" -ge `date +%s` ]; do
70                  echo -ne "$(date -u --date @$(($date1 - `date +%s`
     )) +%H:%M:%S)\r";
71                  done
72                  /bin/bash ./run3
73                  echo "VM's killed on Node: $node"
74                  let node=node+1
75                  killall -9 qemu-system-x86_64 2> /dev/null
76                  sleep 5
77                  printf "\033c"
78      done
79      node=0
80      COUNTER=$[$COUNTER +1]
81      done
82      ;;
```

```
83
84  "2")
85    echo "Specific"
86    echo -e  "Enter Specific Node No."
87    read nm
88    killall -9 qemu-system-x86_64 2> /dev/null
89    for i in $(seq 1 "6")
90    do
91       numactl --cpunodebind=$nm qemu-system-x86_64 --enable-kvm -
      kernel chainloader -initrd calcDouble.img -m 16 -nographic &> /
      dev/null&
92    done
93    seconds=40; date1=$((`date +%s` + $seconds));
94        while [ "$date1" -ge `date +%s` ]; do
95        echo -ne "$(date -u --date @$(($date1 - `date +%s` )) +%H:%
      M:%S)\r";
96    done
97    killall -9 qemu-system-x86_64 2> /dev/null
98    ;;
99  *)
100   echo "You have failed to specify what to do correctly."
101   exit 1
102   ;;
103 esac
```

Listing 8.5: Bash Scripting: To generate workload

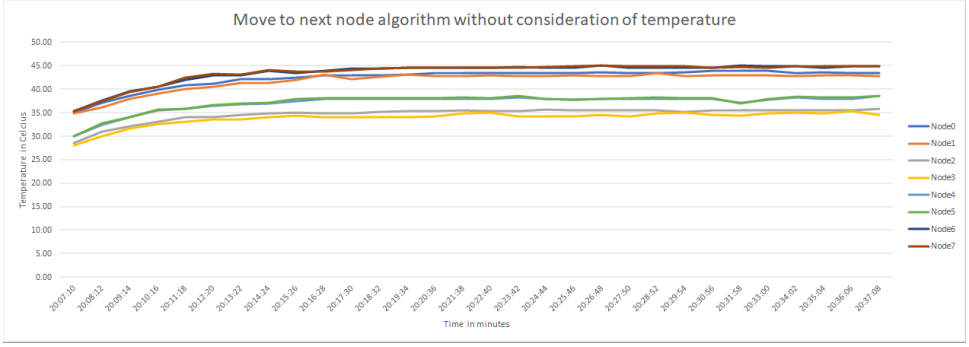

Figure 8.1: Preliminary Experiments: Stressed 12 CPU(s) record

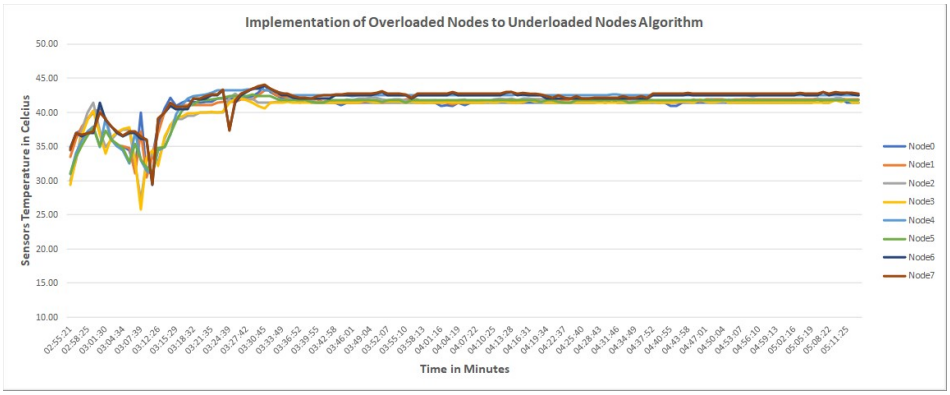

Figure 8.2: Preliminary Experiments: Nodes Temperature with implementation of High to Low Algorithm without setting threshold

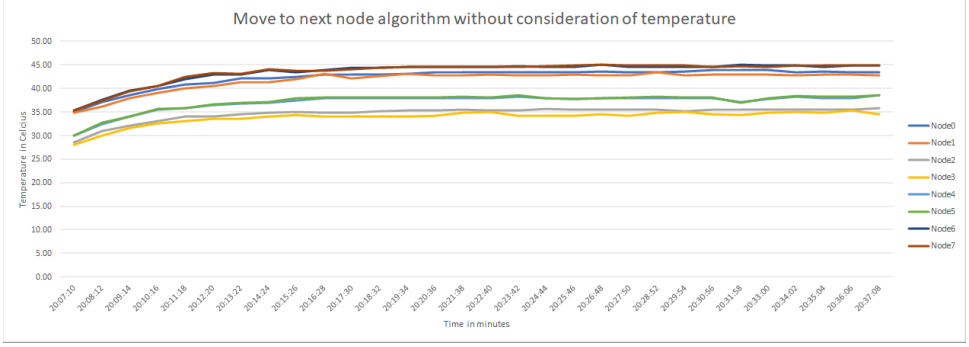

Figure 8.3: Preliminary Experiments: Move to next node regardless of temperature

Bibliography

[1] *14 tail and head commands in Linux/Unix*. URL: https://www.linux.com/blog/14-tail-and-head-commands-linuxunix.

[2] Orsino A et al. "Energy Efficient IoT Data Collection in Smart Cities Exploiting D2D Communications." In: *Sensors Basel, Switzerland* 16.6 (2016). DOI: http://doi.org/10.3390/s16060836.

[3] *Actuators in Internet of Things (IoT)*. URL: https://learniot.wordpress.com/2016/03/29/actuators-in-iot/.

[4] Weiwei Fang et al. "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers." In: *Computer Networks* 57.1 (2013), pp. 179–196.

[5] Yongqiang Gao et al. "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing." In: *Journal of Computer and System Sciences* 79.8 (2013), pp. 1220–1245.

[6] A. A. A. Ari et al. "Efficient and Scalable ACO-Based Task Scheduling for Green Cloud Computing Environment." In: *2017 IEEE International Conference on Smart Cloud (SmartCloud)*. 2017, pp. 66–71. DOI: 10.1109/SmartCloud.2017.17.

[7] N. C. Brintha, J. T. W. Jappes, and S. Benedict. "A Modified Ant Colony based optimization for managing Cloud resources in manufacturing sector." In: *2016 2nd International Conference on Green High Performance Computing (ICGHPC)*. 2016, pp. 1–6. DOI: 10.1109/ICGHPC.2016.7508068.

[8] *Build Greener Energy Solutions with Smart Meter Technology*. URL: https://www.sierrawireless.com/applications/energy-and-industrial/smart-metering/.

[9] *Calculating the Cost of Weather and Climate Disasters*. URL: https://www.ncei.noaa.gov/news/calculating-cost-weather-and-climate-disasters.

[10] Parag Chatterjeeh. *IoT Week Geneva, Workshop, IoT in Medical Sciences and Healthcare*. 2017. URL: http://presentations2017.iotweek.org/.

[11] Brunelli D et al. "Self-Powered WSN for Distributed Data Center Monitoring." In: *Sensors Basel, Switzerland* 16.1 (2016). DOI: http://doi.org/10.3390/s16010057.

[12] S. Dam et al. "Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing." In: *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*. 2015, pp. 1–7. DOI: 10.1109/C3IT.2015.7060176.

[13] W. Ejaz et al. "Efficient Energy Management for the Internet of Things in Smart Cities." In: *IEEE Communications Magazine* 55.1 (2017), pp. 84–91. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600218CM.

[14] A. A. S. Farrag, S. A. Mahmoud, and E. S. M. El-Horbaty. "Intelligent cloud algorithms for load balancing problems: A survey." In: *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*. 2015, pp. 210–216. DOI: 10.1109/IntelCIS.2015.7397223.

[15] *Gartner, Inc. (NYSE: IT)*. URL: http://www.gartner.com/newsroom/id/3165317.

[16] *Genetic algorithm*. URL: https://en.wikipedia.org/wiki/Genetic_algorithm.

[17] P. Grover and R. Johari. "BCD: BigData, cloud computing and distributed computing." In: *2015 Global Conference on Communication Technologies (GCCT)*. 2015, pp. 772–776. DOI: 10.1109/GCCT.2015.7342768.

[18] E. Gupta and V. Deshpande. "A Technique Based on Ant Colony Optimization for Load Balancing in Cloud Data Center." In: *2014 International Conference on Information Technology*. 2014, pp. 12–17. DOI: 10.1109/ICIT.2014.54.

[19] *How to Monitor Your Computer's CPU Temperature*. URL: https://www.howtogeek.com/howto/windows-vista/ever-wonder-what-temperature-your-cpu-is-running-at/.

[20] *IEEE Towards a Definition of the Internet of Things (IoT)*. URL: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf.

[21] *IoT Applications Spanning across Industries*. URL: https://www.ibm.com/blogs/internet-of-things/iot-applications-industries/.

[22] *Iot Systems and Medium Range Radio Solutions*. URL: https://bridgera.com/iot-systems-medium-range-radio-signals/.

[23] Huang Zhen Jin, Lu Yang, and Ouyang Hao. "Scheduling strategy based on genetic algorithm for Cloud computer energy optimization." In: *2015 IEEE International Conference on Communication Problem-Solving (ICCP)*. 2015, pp. 516–519. DOI: 10.1109/ICCPS.2015.7454218.

[24] M. Khan, B. N. Silva, and K. Han. "Internet of Things Based Energy Aware Smart Home Control System." In: *IEEE Access* 4 (2016), pp. 7556–7566. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2016.2621752.

[25] Z. Khan et al. "IoT Connectivity in Radar Bands: A Shared Access Model Based on Spectrum Measurements." In: *IEEE Communications Magazine* 55.2 (2017), pp. 88–96. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600444CM.

[26] Mohit Kumar and S.C. Sharma. "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing." In: *Procedia Computer Science* 115.Supplement C (2017). 7th International Conference on Advances in Computing and Communications, ICACC-2017, 22-24 August 2017, Cochin, India, pp. 322–329. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.09.141. URL: http://www.sciencedirect.com/science/article/pii/S1877050917319695.

[27] Paul Lin. "Optimize Data Center Cooling with Effective Control Systems." In: *White Paper 225* (2017).

[28] *Linux sort command*. URL: https://www.computerhope.com/unix/usort.htm.

[29] *lm-sensors*. URL: https://web.archive.org/web/20120428045441/http://www.lm-sensors.org:80/wiki/ProjectInformation.

[30] Giuseppe Fico ACTIVAGE Technical Manager Universidad Politécnica de Madrid. *IoT Week Geneva, IoT & SDG Sessions, IoT for Good Health and Well Being*. 2017. URL: http://presentations2017.iotweek.org/.

[31] M. Malekloo and N. Kara. "Multi-objective ACO virtual machine placement in cloud computing environments." In: *2014 IEEE Globecom Workshops (GC Wkshps)*. 2014, pp. 112–116. DOI: 10.1109/GLOCOMW.2014.7063415.

[32] Shiwen Mao Min Chen and Yunhao Liu. "Big Data: A Survey: Mobile Networks and Applications." In: 19.2 (2014), pp. 169–212.

[33] *Oracle Corporation. Introduction to Virtualization*. 2011. URL: https://docs.oracle.com/cd/E20065_01/doc.30/e18549/intro.htm.

[34] D. Osada and E. Gambart de Lignières. *Home automation and connected cars: a source of services and tools for insurers - Blog Sopra Steria*. 2017. URL: http://blog.soprasteria.com/insurance-home-automation/.

[35] SINTEF Ovidiu Vermesan & Peter Friess. *Internet of Things Applications From Research and Innovation to Market Deployment*. International series of monographs on physics. Clarendon Press, 1981. ISBN: 9780198520115.

[36] SINTEF Ovidiu Vermesan & Peter Friess. *Internet of Things Applications From Research and Innovation to Market Deployment*. River Publishers Series in Communications, 2014. ISBN: 9788793102941.

[37] R.K. Pateriya Prachi Verma Sonika Shrivastava. "Enhancing Load Balancing in Cloud Computing by Ant Colony Optimization Method." In: *International Journal of Computer Engineering In Research Trends* 4.6 (2017), pp. 269–276. DOI: http://ijcert.org/ems/ijcert_papers/V4I6012.pdf.

[38] D. Puthal et al. "Cloud Computing Features, Issues, and Challenges: A Big Picture." In: *2015 International Conference on Computational Intelligence and Networks*. 2015, pp. 116–123. DOI: 10.1109/CINE.2015.31.

[39] *Reducing the Massive Energy Appetite of Data Centers*. URL: https://www.insidescience.org/news/reducing-massive-energy-appetite-data-centers.

[40] Parag Chatterjee & G Chandra Deka Ricardo Armentano Robin S Bhadoria. *The Internet of Things: Foundation for Smart Cities, eHealth and Ubiquitous Computing*. CRC Press, Taylor & Francis, 2017. ISBN: 9781498789028.

[41] S. Al-Sarawi et al. "Internet of Things (IoT) communication protocols: Review." In: *2017 8th International Conference on Information Technology (ICIT)*. 2017, pp. 685–690. DOI: 10.1109/ICITECH.2017.8079928.

[42] *Self-management*. URL: https://en.wikipedia.org/wiki/Self-management_(computer_science).

[43] Mala Kalra Shekhar Singh. "TASK SCHEDULING OPTIMIZATION OF INDEPENDENT TASKS IN CLOUD COMPUTING USING ENHANCED GENETIC ALGORITHM." In: *International Journal of Application or Innovation in Engineering and Management (IJAIEM)* 3 (2014), pp. 286–291. ISSN: 2319 - 4847.

[44] Z. Song, X. Zhang, and C. Eriksson. "Data Center Energy and Cost Saving Evaluation." In: *Energy Procedia* 75.Supplement C (2015). Clean, Efficient and Affordable Energy for a Sustainable Future: The 7th International Conference on Applied Energy (ICAE2015), pp. 1255–1260. ISSN: 1876-6102. DOI: https://doi.org/10.1016/j.egypro.2015.07.178. URL: http://www.sciencedirect.com/science/article/pii/S1876610215009467.

[45] *Standard Deviation Calculator*. URL: http://www.calculator.net/standard-deviation-calculator.html.

[46] G. Stoykov and A. Yazidi. "A Biomorphic Model for Automated Cloud Adaptation." In: *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. 2015, pp. 179–185. DOI: 10.1109/UCC.2015.34.

[47] *Stress*. URL: https://people.seas.harvard.edu/~apw/stress/.

[48] *Stress-ng*. URL: https://wiki.ubuntu.com/Kernel/Reference/stress-ng.

[49] W. Sun et al. "SAACO: A Self Adaptive Ant Colony Optimization in Cloud Computing." In: *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*. 2015, pp. 148–153. DOI: 10.1109/BDCloud.2015.53.

[50] *taskset(1) - Linux man page*. URL: https://linux.die.net/man/1/taskset.

[51] S Sharon Amulya Joshi T.Deepa. "A Survey on Load Balancing Algorithms in Cloud." In: *International Journal of Computer Engineering In Research Trends* 3.7 (2016), pp. 371–374.

[52] *The challenge of securing smart homes*. URL: https://www.rambus.com/blogs/the-challenge-of-securing-smart-homes.

[53] *The Internet of Things (IoT) is Accelerating the Transformation to Smart Energy*. URL: https://www.sierrawireless.com/applications/energy-and-industrial/energy/.

[54] *To Study Earth's Climate, Look to the Ocean*. URL: https://www.ncei.noaa.gov/news/study-earth%E2%80%99s-climate-look-ocean.

[55] *Virtual Waste: Flowing from a data center near you*. URL: https://discardstudies.com/2017/09/25/virtual-waste-flowing-from-a-data-center-near-you.

[56] *Want an Energy-Efficient Data Center? Build It Underwater*. URL: https://spectrum.ieee.org/computing/hardware/want-an-energyefficient-data-center-build-it-underwater/.

[57] *Water and Cooling*. URL: https://www.google.com/about/datacenters/efficiency/internal/#water-and-cooling.

[58] *What is a smart meter?* URL: https://www.confused.com/home-and-lifestyle/gas-and-electricity/what-is-a-smart-meter.

[59] *What is CPU Affinity?* URL: https://community.mellanox.com/docs/DOC-1924.