CS7NS1 Module Final Report

Name: Ashish Arvindrao Kannur

ID: 19300875

Date: 25th November 2019

SECTION 1:

Group Project 4 self-evaluation :

**Team name**: Group 3

**Team members**: -

1. Ashish Arvindrao Kannur
2. Basit Hamid Sofi
3. Tanvi Bagla
4. Yeshwanth RajaReddy

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Name1 | Name2 | Name3 | Name4 |
| Name1 Initials |  | Ashish | Basit | Tanvi | Yeshwanth |
| Effort |  | High | High | High | High |
| Effectiveness |  | High | High | High | High |

SECTION 2:

What I studied in this module:

**Scalability**:This module has taught me how distributed systems can be made scalable. Distributed systems are formed by integrating multiple systems or machines. This is achieved by the systems communicating in between them through system messages. Thus , the features of individual systems like memory, computing power and storage are also multiplied or scaled. Thus these integrated scaled systems can further be used solve real world problems which single systems cannot .

Scalability , redundancy, availability , reliability are the needs for the distributed systems. Scalability in terms of multiple systems being integrated together to form one system, redundancy in terms of system having the capability of withstanding failures, availability in terms of replications of system parts so that system is guaranteed to be active all the time, and all these properties described which final account for reliability.Scalable systems must be adaptive so theta they are reconfigurable and reusable. Distributed systems work by co-ordinating in between the small systems and distributing the compute intensive tasks amongst them. These compute intensive tasks are divided into several parts/ algorithms and then run on each single subsystem using multithreading or multiprocessing.This shows the dispersed property of the scalable computing. While scaling a system we often come across 2 scenarios: The system is constant but the latency of the system is not as per requirement and second is that as the data increases , the latency of the system increases. Now in a system, not all parts are scalable. Some parts which can be parallelized , can be scaled only. In general we would expect that if we just double the number of resources of a system, then the runtime would be reduced to half. But that is not the case in distributed systems. Amdahl’s law perfectly explains the relationship of speed up factor and the serialization factor.

**CPUs and GPUs:** The progress of the module was pretty smooth. It all started with properties of scalable systems like how range of their scalability, adaptiveness, dispersed behaviour of resources/nodes/tasks, accessibility of distributed system as a whole, its affordability and reliability. Then after understanding the distributed systems in nutshell, we went on to understanding first domain of Scalable computing systems: CPUs and GPUs. I learnt about the programmable GPU architecture called CUDA(Compute unified device architecture) which was designed by NVIDIA for general purpose parallel computing. CUDA had its own advantages of capability to handle high throughput rates, low latency in updating the entries over existing technologies of FPGAs and TCAMs. Host functions are basically handled by CPU and the device functions are handled by GPUs. Even though CPU and GPU can perform same functions, the execution time differs in both due to the hardware configuration difference in both. GPUs have many cores as compared to CPU and hence most complex calculations/computations are carried out by GPU as multicores in the GPU helps it execute the processes or computations faster and more efficiently through parallel processing. Thus GPUs are efficient in computing high resolution images too along with tasks which are not graphical. Both GPUs and CPUs have their own benefits and are versatile . Neither of them can replace one another. It is the user who has to decide which of the two should he use to execute a particular task efficiently. Or else he might just end up wasting his resources. When used inefficiently , GPUs and CPUs get heated and may damage the hardware.

**IOT:** Then we went on to understanding the most famous IOT which has revolutionised our perspective towards things. Now-a-days there is this trend of smart city booming up everywhere. Almost each and every device right from household appliances to your vehicle are interconnected via internet. Thus the term internet of things. IOT is nothing but a network of devices/machines that are interrelated and do not require hum intervention for the whole system to function. But it is not as easy as it sounds. It requires convergence of multiple technologies like machine learning, network programming, network protocols, sensors/devices capable of capturing and storing and transmitting data, embedded systems, etc. IOT has its applications in multiple domains like smart home, healthcare, transportation, industrial applications, environmental monitoring, etc. Growth of IOT has pushed the technology to make available sensors with low power and lost cost and has also necessitated the need for efficient communication protocols for sending the data over the network and finally to the cloud. Recent boom in cloud computing and infrastructure has opened up a horizon of options in IOT and increased its capacity to handle data and processing it remotely rather than locally. With the extensive research and advancement in analytics and machine learning , it has been possible to predict the sensor behaviour and hence future actions can be informed . Now-a-days voice recognition softwares are gaining a lot of attention and we can see a large scale implementation of voice recognition in IOT to communicate the systems in cloud network. Now when all the technologies converge , this leads to generation of vast amount of data . To handle this big data, appropriate devices and protocols are necessary. This is where IOT starts to face scalability issues. The range of sensors available and their sophistication level is very high, but also what is to be done with the data generated from them needs to also be strategized efficiently. Working on all the acquired data is currently based on cloud servers , but it wont stay that way. In near future, this will have to be taken up by the IOT nodes locally. Because in the former case, the intelligence basically resides in the cloud and every time the data needs to go from the network capturing node to the cloud, then the cloud computes and informs a decision , which has o again travel back to the node to carry out the desired action. This takes some time and introduces latency in the system when multiple nodes interact with the cloud. There are more considerations that need to be taken into account in IOT like power /energy considerations, memory , resources, optimal /robust system design, secured and trustable communication over the network, etc. IOT devices need power to capture data and to send them over the network. Power consumption varies based on the geographical locations of the nodes in the IOT network. If the nodes are far away , then they would need more power to communicate or transfer the data from one node to another. SO the duty cycle of the IOT sensors need to be carefully designed so as to function optimally in terms of energy consumption. Ideally the nodes should wake up only when it has to do a task like capturing/ sending/ receiving data. Other times it must be in sleep mode. Also for the IOT devices to be affordable , they cannot have large memory to store accumulating data. Hence the data needs to be disposed off to the network. Also , when the data travels in an IOT network, it must travel the shortest path to the cloud ,i.e, in minimum hops the data must reach the cloud from the sensor. This leads to waking up of minimum number of devices and hence efficient utilization of system resources of a network. To solve the problem of location of devices, recently research has started on implementing IOT comms through 5G. Next, the design of the IOT system must be robust ,i.e, it must be capable of withstanding failures. The system must not cease to work even if some devices or nodes in the network stop responding. This means that the system must be highly available always. Also the communication in between the nodes in the network must be trustable ,i.e, robust protocols which are unhackable must be used as mostly sensitive data is being transferred in IOT network and misuse of the data may happen. The recent major catastrophic hack of IOT devices in BOTNET attack which impacted the entertainment industry giants is one such example. In IOT network , generally the data from the sensors/nodes is transported to the a device called sink which is closest to the sensor. Sink usually keeps accumulating the data unless it makes connection with an Aggregator in the network to dispose the data collected.

Sinks usually have low memory as compared to the Aggregator. Thus data is usually deleted from sink once it is sent to the aggregator. The Aggregator then sends the data to the cloud for further processing . This type of architecture is usually followed in BAN(Body Area Networking) and VAN(Vehicular Area Network).Implementing this structure in VAN is a little bit difficult because the motion of the vehicles pose a major challenge in establishing communication between the vehicles and the edge devices.

**Types of Computing:** Cloud computing was already in picture due to IOT. But then the issues in IOT like latency and efficient use of network resources gave rise to the need of edge and fog computing. Major challenge was the time being taken for the computations as the intel of the network resided in the cloud. Thus in edge /fog computing the intel of the network was brought closer to the nodes. Due to this the end user enjoyed faster response to online requests, faster access to the web applications, etc which is the crucial need of today. This also eliminated the need of large data centers to transfer the data to the cloud continuously and thus their energy consumption. This also provided the users high availability of data as multiple edge devices could now be configured to perform similar functions as that of the cloud. Thus fog/edge computing does not have a central control system or we can say that there is no centralized intel in fog/edge computing. The intel of the network is spread throughout the network. Thus most of the computations are done at the edge itself rather than the cloud. Only very compute intensive tasks which cannot be done at the edge, are sent further to cloud for processing. However in fog/edge computing , there is a compromise on security as compared to cloud computing.

SECTION 3:

What I did during the module:

In parallel with the theoretical knowledge, I also gained practical knowledge by doing multiple projects and group project where I got the opportunity to execute a large scale project by combining my ideas with my team.

My introduction to this module was through the lecture followed by assignment to prepare report by reading the research papers on CPUs and GPUs. It then I got to know more about the research papers published online , how they need to be searched for and how to pick the right research paper from all the available papers on the topic of your interest. I had picked up following papers for my report: “Scalable pre-processor for Network Security Systems” and “IP address lookup device using GPU”. After going through these papers I got know that it only when we pick up the correct research paper , we get a clear picture of the concepts. Also going through the research papers in the next assignments on IOT, Fog computing, cloud computing, Edge computing throughout the module gave me an overview about how to write my research paper for my dissertation, plan about the pre-requisites, cautions need to be taken and plan of execution. From the CPU-GPU research I learnt about the architecture of GPU, different protocols used internally in GPU for data computation, data structure ,memory structure inside GPU, functions of CPU and GPU, etc.

Also I got the opportunity to work on few practical projects. First one was about

**Project 1:**

4 character Captcha Image Recognition: In this project I was introduced to CNN. We were basically asked to go through the code, modify it to train and make a model to recognize 4 character captcha images. I learnt some techniques about how to generate training samples, then train the sample over sufficient sample size to give more accuracy, then classify the given question images using the model prepared. So, in this I got an overview of CNN pipeline and how the machine learning works.

5 Character Captcha Image Recognition: This was a one step up as compared to 4 character captcha images. We were required now to modify the scripts used for the previous project ,i.e, generate 5 character images for training, increase the training sample size, train and prepare a model to get more and more accuracy . Then check if the model we prepared works correctly on the question images. Here I got to know that using GPU makes the training faster . For CPU it almost took me overnight to train my model over 100000 images. Whereas with GPU , it took me only about 3 hours to train my model to give 90% accuracy.

**Project 2:**

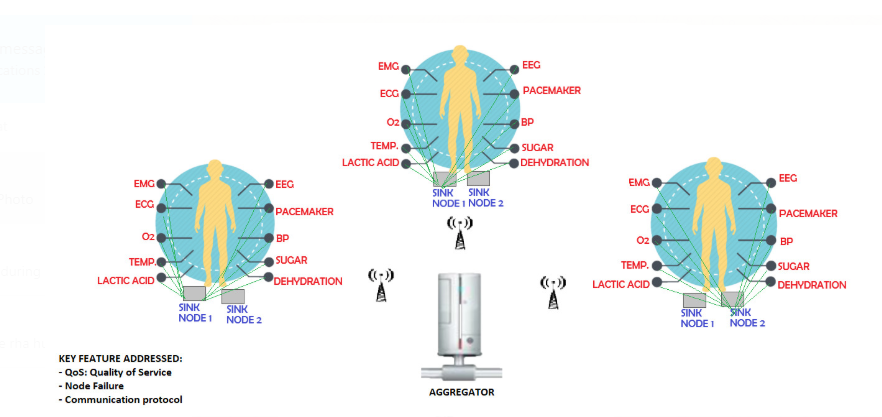
8 Character Captcha Image Recognition: Now the level of difficulty was increased one step again. Here, just the generating of 8 character images , increasing the sample size beyond 100000 and training a model did not work much. Accuracy was not that good. So after a discussion in lecture, we got to know that dimensionality reduction was what we were expected to do as the image content increased in size so that the machine learning or rather CNN is trained efficiently. Through this the model accuracy increased and helped us understand the importance of dimensionality reduction.

**Project 3:**

8 Character Audio captcha Recognition: Now this was more challenging. We had a hard time figuring out how could audio be incorporated in CNN. Again after discussion through lectures, we got to know that audio can actually be converted to spectrograms and spectrograms can be treated as images , which could further be used for training. First challenge was to generate audio captchas from some text to speech service. Google TTS was well known and so I prepared script for generating audio captchas by using Google TTS service, converting it into suitable format so that spectrogram can be generated, then converting the .wav file to spectrogram and then modified the training script used previously to reduce the dimensions of the spectrogram image and train a model. Also , using GPU the process was faster. However the whole process of generating thousands of images and training them took almost 4 days as this was a very compute intensive tasks. So here we were introduced to the cloud AWS instances which could do these tasks for us without using our systems at some cost. AWS instances are powerful and wide range of configurations are available for us to select according to our time needs.

**Project 4**: **(Github link:** <https://github.com/basithamid/Scalable-Project4>)

Here we were introduced to cloud/fog/edge computing concepts and had to simulate a model demonstrating Body Area Networking or Vehicular Area Networking. Following was my group project on Body Area Networking:

**Architecture:** 

**Body Area Network (BAN): (Github link:** <https://github.com/basithamid/Scalable-Project4>)

This was a group project where we had to simulate a Body Area Network either with the help of AWS Lambda or locally with the help of our own laptops and computers.

At first we tried our hand at trying to understand how this would work with AWS lambda. We were able to get some of the setup done for AWS but were not able get it to work. So, due to the time constraint for the project and also because we felt comfortable with using the local systems, we decided to stick creating our system from scratch on our machines.

**Our approach:**

* Our approach was to have 3 instances (machines) simulating three bodies with 10 python programs simulating ten different sensors inside a human body.
* Each instance (body) has two sinks for getting data from the sensors. We decided to stick with two sinks because so that in case the primary sink goes down, the secondary sink will start receiving data from the sensors till the primary sink is back on. The sink could go down because of one of the two reasons: 1) If the battery is low. 2) If there is something wrong with the connection.
* We decided to have a fourth machine which would work as a Data Aggregator. So, all the data that is coming from the three machines will be accumulated at the aggregator. If there is ever a problem with connecting to the aggregator, the data would keep accumulating at the sinks of the respective machines till the aggregator is back up.

**Challenges Selected:**

* Power considerations and power efficiency.
* Security of communication protocol used in exchanging the data.
* Availability of nodes incase of failures
* Prevention of data loss.
* Storage efficiency.
* Effective duty cycle implementation.

**Platform in use:**

Python and socket programming.

**Sensor Data:**

We decided to have 10 python programs simulating 10 devices on the machines. The sensors were:

1. Pacemaker
2. Body Temperature
3. Blood Pressure Sensor
4. Blood Oxygen Level
5. ECG Level Sensor
6. EEG Sensor
7. EMG Sensor
8. Lactic Acid Sensor
9. Insulin Level Sensor
10. Dehydration Sensor

We generated data for every single sensor after a fixed interval of time and that data was sent to the sink in the form of a JSON record. At the sink, the data was saved in separate files before sending the files to the data aggregator. This was done to make sure we don’t lose ever lose the data.

**Power considerations:**

We wanted to make sure that we measured the battery of the sensors at all points when they transmitted the data to the sink and also while they were in the idle state. This was because we did not want the sensor to send the data if it was low on battery. We wanted to make sure that the sensor battery was getting charged when it was low on battery. We were able to achieve this functionality.

The same considerations were taken for the sinks. Since the primary sink was mostly going to be used for receiving and transmitting data, it would lose battery faster than the secondary sink. Therefore, we implemented the functionality to disconnect the sensors from the primary sink if its battery was low. Once that happened, the sensors would connect to the secondary sink.

**Feedback of Professor on our project:**

Data: We decided to send the entire sensor data to the sink node, and onwards to the data aggregator. This would mean that more power would be used by the sensors when sending data as the size of the data would be large and more power would be used by the sinks while receiving the data.

Node discovery: We had static fixed connections between our sensors and their sinks and there was a fixed connection between the sinks and the data aggregator. There was no dynamic node discovery.

Alerts: When the certain values went above or below a threshold, such as above 100 in case of pacemaker, we just appended an alert message as part of our data record. There was no actionable item when something like this happened. Essentially, if this was part of an actual Body Area Network, the person being affected had a heart attack, there was nothing actionable that this alert would lead to and the person would most definitely end up dying.

Multihop: Sensor to sensor communication would have increased the power efficiency of the sensors so that they wont spend much power in sending the data directly to the sink, but to the nearest node.

**What we could have done better:**

Data: We could have multiple hops for the sensor data so that similar data at different points of the body would not differ and we could send only some of the data, the critical/important data to the sinks. This would also reduce the amount of power used while sending the data as we are only sending a part of the information generated by the sensors.

Node Discovery: We could have implemented dynamic node discovery to discovering the edge nodes/data aggregator since the human body is not stationary and tends to move around a lot. Therefore, the human might be in closer proximity to a different edge node at a different point in time and dynamic discovery of the aggregator node could reduce the battery consumption if we connected to a node which was closer to the sensors/sinks. This consideration can also be made for discovering the sensor nodes in the body so that in case any sensor goes down, data can be transmitted through other sensors.

Load Balancing:

Load balancing of the data can be a future implementation in the future scope of our project. This means that whenever the load on a particular sink/aggregator is more, the load should automatically get distributed or get diverted to the other available sink/aggregator. This would mean high availability of systems, i.e, they are capable of withstanding failures and system remains active all the time.

Routing:

Replication of sensors could have been done for example: Multiple blood pressure sensors located on different parts of the body would send the data to one final sensor(which has storage) and then the aggregated data of all similar types of sensors can then be forwarded to the sink. This would increase the power efficiency of each sensor as the sensors wont have to send the data directly to the sink. Also the power efficiency of sink would be increased as instead of receiving data multiple times from different sensors of similar type, it would receive the aggregated data of all those sensors from the main sensor in one go. But this has a caveat that number of sensors would increase in the body.

Alerts: We made simple alert messages as part of our data. What we could have done was implement some kind of email functionality where if the data goes above or below a threshold an email would go to the General Practitioner or the doctor of the person affected by the health problem.

Platform: We could have used AWS platform which already has inbuilt functions for network programming. This could have saved time and also helped us with node discovery.

SECTION 4: MODULE EVALUATION

**Likes about the module**: -

* Lectures we quite engaging and interactive. Lectures focussed more upon practical concepts rather than theoretical. ‘Scalability’ aspects made completely understandable throughout the module.
* The increasing difficulty of assignments/projects was quite a learning experience.
* Assignments and report writing introduced me to literature of research papers.
* Projects introduced me to machine learning and CNN, their pipeline and gave new perspective of seeing at things. They also introduced me to Amazon products , their importance and how to use them to make our tasks easier.

**Improvements about the module**: -

* Lab systems were not much helpful as most of the time was spent in configuring the systems.
* More time could be spent on explanation of coding along with the theoretical concepts which would save the time of students spent on initial research about the coding approach and hence they can utilise the time to do more indepth research about the concepts and programming.