UTILISING EDGE COMPUTING IN A 3-TIER SYSTEM BETWEEN CLIENT DEVICES AND A DATA CENTRE.

Project Description

With the growing rise in popularity of Client Devices and Cloud Computing, it follows that there would be an increase in the strain placed on central servers to process a multitude of requests. The purpose of this project is to see how the implementation of Edge Computing [2] can reduce the stress placed on these Data Centres and improve the performance of Client Devices.

BI Intelligence says *"Edge computing provides a means to collect and process data at local computing devices rather than in the cloud or a remote data center*" [1] This means that the stress of data centres should be reduced and the response times to clients should be improved.

A 3-tier network will be set up, this will consist of a "Client Device", "Edge Node" and a "Data Centre". The Client Device will send data to an Edge Node that will process it and then send the processed data on to the Data Centre or return some required information to the client directly.

The Edge Node will be used as a form of Gateway [19] to handle traffic from a certain subset of computers. This will allow for some requests to be processed or pre-processed on the Edge Node directly to ease stress on a Data Centre.

An advantage [22] of Edge Computing is that there is a reduction in response time from the Client's perspective achieved by Edge Nodes assisting in heavy application processing, which in turn improves their user experience.

The Client Device will be physically closer to the Edge Node than it is to the Data Centre. This is to allow for reduced latency and faster processing of requests. The Edge Node could be located at a Primary Connection Point [20], which is between the Client Device and the Data Centre. The purpose of it being located here is that it is in a centralised location so that the beneficial effects of the added resources can be shared among several hundred customers but not so many as to have a detrimental effect caused by delayed response times. However, it could be configured to run in a home network as well.

It will also be possible to host multiple Edge Computing solutions in one environment as to ease the stress on multiple Data Centres while utilising a single Edge Node. This could mean dynamically adding solutions to pre-existing Edge Nodes as necessary and controlling access to resources.

A few likely candidates of applications for this technology are;

- Voice Recognition. Aspects of this application include; (1). Receiving a voice request from a Client Device (2). Performing some voice to text processing on an Edge Node. (3). Infer information from the processed request on an Edge Node. (4) Sending the text and the inference

to the Data Centre for storage. (5). A benefit of this application could be reduced latency in getting a response to the Client Device's request.

- Deep Learning. Aspects of this application include; (1). Receiving information from a Client Device. (2). Processing requests based on learning models and data sets on the Edge Node. (3). Sending the processed request on to the Data Centre. (4). A benefit of this application could be reduced CPU load on the Data Centre.

- Processing many requests from Internet of Things (IoT) devices. Aspects of this project include; (1). Receive requests from Client Devices. (2). Process request data to infer information on the Edge Node. (3). It will then be sent for storage in a Data Centre. (4). It can be queried at a later stage. (5). A benefit of this application could be that network use between the IoT devices and Data Centre would be reduced.

- Caching. Aspects of this application include; (1). Receive requests from Client Devices for web pages. (2). Handle this request on the Edge Device and store it for later retrieval. (3). Process requests quickly for web pages that have been already stored on the Edge Node and return the information directly to the Client Device. (4). A benefit of this application could be that the network traffic between Client Devices and the Data Centre is reduced.

There are four main ideas to be investigated within a 3-tier system;

- Improved Latency for the Client to obtain a response achieved by being physically closer
- Reduced strain on data centres by pre-processing and handling requests on the Edge Node
- Resource sharing in the Edge Node, utilising Edge Node resources for faster processing for the client
- Dynamic off-loading of workloads to the Edge Node, such as processing data in a way that it would have been processed in the Data Centre

Goals and Requirements

Goals

The goal of this project is to design an Edge Computing infrastructure capable of servicing multiple Edge Computing applications aimed at reducing strain of Data Centres and improving performance of Client Nodes.

The way that resources are allocated at the Edge Node currently will also have to be investigated as this will allow for better control over the applications running on the Edge Node. The default load balancing policy (such as round robin [21]) will need to be investigated to ensure that this is the best method available or to find a better alternative.

Requirements

The most critical aspect of this project is the applications that run on the infrastructure but it is necessary to have the infrastructure in place to service these applications.

Requirements are as follows;

1. A user interface for the Client Node to allow interaction with the Edge Node and Data Centre and to display data to the user
2. A low latency connection between the Client and Edge Node
3. The requests from the Client to be pre-processed quickly in the Edge Node before being passed on to the Data Centre or returned to the Client
4. The implementation of a Data Centre application
5. The computational strain placed on the Data Centre should be lessened by the introduction of Edge Computing
6. The development of applications capable of performing Edge Computing tasks
7. The ability to run multiple applications on the same Edge Node to allow for extensibility
8. Applications can be easily deployed, controlled, and updated

Acceptance Criteria

There are several research questions that can be addressed with this project;

- Demonstrate on at least one of the applications that latency (response time to user requests) is reduced in the setup
- Demonstrate on at least one of the applications that network utilisation between Client Device and Data Centre is optimised
- Demonstrate on at least one of the applications that CPU load of the Data Centre has been reduced
- Demonstrate that the developed load balancing policy can improve utilisation of CPU, memory and network and improve the metrics above compared to the default policy available in the setup.

Hardware and Software

Hardware

The hardware needed will be;

- The Client
    - A low powered client in the form of a Raspberry Pi 3 [5]
- The Edge Node
    - A cluster of small devices so they can be stored easily, this Edge Node will use 3 Raspberry Pi 3's
    - The decision to use the Raspberry Pi for this cluster was because they are affordable, portable and have a small impact in both size and power consumption. [3, 4, 5]
- The Data Centre
    - A laptop or desktop computer as these will have more CPU and memory available to process requests when compared to the Client or Edge Node, which is what should be expected

Software

The software needed will be;

- An operating system for the Raspberry Pi's
  - An option would be Raspbian [9] as it is optimised for the Raspberry Pi hardware
- A method of handling the Edge Computing applications
  - It could be possible to develop applications and run them directly on the host
  - It could also be possible to run the applications in virtual machines
  - Another option would be Docker [7] as it was developed as an easy solution to software development and deployment
- A method of developing the client's user interface
  - It could be possible to develop a native Linux application to run on the Raspberry Pi's
  - A second option is to develop a website that can be run on the client
  - Another option would be Electron [6] as it allows the creation of native desktop applications using JavaScript and HTML while also having access to Node.js API's
- An application that is best to receive requests on the Edge Node
  - There is a multitude of server software available to use, some options to consider are;
    - Nginx [10]
    - Apache [11]
    - Lighttpd [12]
    - Node.js [13]
- A way to develop applications for the Edge Node
  - It will be necessary to process requests for different applications, it is possible to use Python or JavaScript or other appropriate languages and libraries
- An implementation of a web server to act as the Data Centre
  - It will be possible to consider the web server applications described as options for the Edge Node
  - It will also be possible to consider .NET/C# solutions to see if these are better fitted if running on a windows environment

Solution Approach

A few of the candidates stated in the problem description will be utilised in this project to find out if Edge Computing is viable and useful to reduce latency, ease stress on central Data Centres, share the resources of the Edge Node and offload workloads to the Edge Node.

A way to efficiently utilise these resources for an application that requires more computational demand than Client Devices can or should offer will be investigated. These metrics can be recorded by measuring latency of requests between these devices, the computational stress on each node and the total cost of these systems including their potential re-usability and energy consumption.

Client

Electron [6] will be used for the Client UI and for controlling requests to the Edge node and Data Centre;

- Electron is a framework that allows development of applications that can run natively on Windows, Linux and OSX
- It is beneficial because the application can be used across a multitude of devices
- There is also native access to the host hardware as the framework runs on Node.js

Electron was chosen over a native Linux application as it not dependant on architecture or operating system.

Electron was chosen over the use of a website as it gives more control and allows access to the host hardware to make requests to see available resources.

Edge Node

Docker [7] will be used for hosting Edge Node applications and controlling access to hardware resources. It will also manage requests from the Client and to the Data Centre

- Docker allows for easy and fast application development and deployment
- Docker has a swarm mode that allows for easy distribution of computing between multiple computers which will be of use when running on multiple Raspberry Pi's
- It has API's for controlling and dynamically scaling applications resources as they are hosted in containers

A benefit to running the applications directly on the host would be that there is less overhead with containerisation or virtualisation. However, Docker was chosen as it keeps the host from needing many pre-requisites, such as a web server and can easily be used to see applications running and for use of controlling access to resources.

A benefit to running the applications in virtual machines is that the applications are all kept separate and can be removed or updated individually without the need to interfere with other hosted applications. However, Docker was chosen over virtual machines as containers require less resources than virtual machines [18].

There are a few possible options for the caching application

- Redis [14]
- Memcached [15]

These cache engines are comparable so it is unclear which one will be best for this application. Memcached can be more efficient with RAM but Redis has more scope to be configured to meet requirements [16].

There are a few possible options for the voice recognition application

- CMU Sphinx [23]
- Kaldi [25]
- Julius [26]

- Jasper [27]

The setup time for preparing, optimising, and obtaining results was least for Kaldi as well as it obtaining outstanding results. The Sphinx family does not seem to be as accurate as Kaldi [24]. Within the documentation for Jasper mentions the use of Raspberry Pi's [27] which could indicate that this is a good solution with the restricted hardware of a Raspberry Pi and we do not have a lot of computing power available, such that the Kaldi engine seems to require [25]. Jasper along with the CMU Sphinx Speech-To-Text (STT) engine ensures that the processing occurs offline. There will need to be some experimentation done in this area to see which STT engine is best suited to this experiment. The processed text can be passed to the Data Centre for processing to occur. This will mean that the Client received spoken work, send this to the Edge Node that processed it into text and forwarded it to the Data Centre.

These are a few of the possible options for the Machine Learning technologies that can be used

- Weka [28]
- TensorFlow [29]

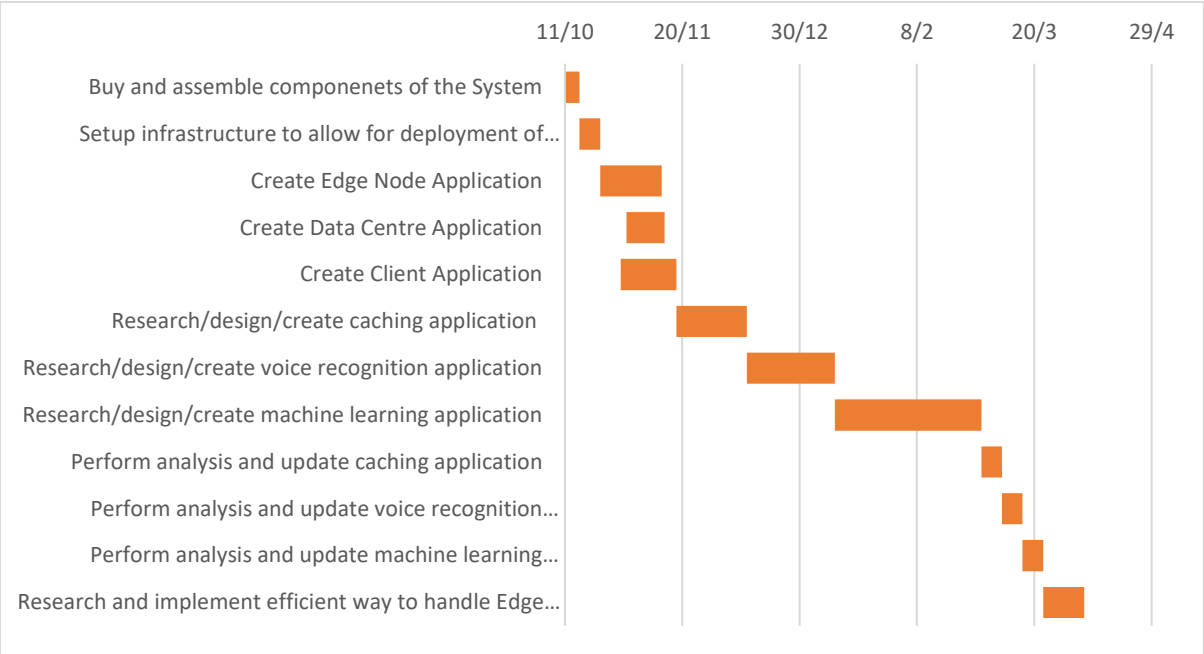
It will be hard to know exactly what the best machine learning technology or library is until it can be tested for speed and compatibility within the Docker environment. Machine learning will be investigated to see if it is feasible in a smaller Edge Computing environment. A possible use case for this application is to take user input from the Client Device and predict or categorise it on the Edge Node so the Data Centre can be queried for information pre-emptively.

For the web server, Node.js [13] will be used as there will be some crossover between the client Electron application which uses Node.js also.

Data Centre

In the Data Centre, ASP.NET Web API [8] will be used as there is no UI overhead needed for hosting web pages and it is possible to use HTTP put and get methods. This can be hosted and configured in IIS [17] as it is readily available on windows. It will also be possible to utilise existing websites as the Data Centre in this project.

Work Plan



Major Milestones

- Getting Edge Computing infrastructure setup (16/10/2016)
- Getting Client user interface implemented (18/11/2016)
- Creating and deploying Edge Computing caching application (12/12/2016)
- Creating and deploying Edge Computing voice recognition application (11/01/2017)
- Creating and deploying Edge Computing machine learning application (02/03/2017)
- Analysing Edge Computing applications (23/03/2017)
- Researching efficient way to handle Edge Node Resources (06/04/2017)

Deliverables

- Infrastructure to manage all the applications, this will be delivered in the form of the Raspberry Pi cluster utilising Docker Swarm
- Client Application, this will take the form of an electron application running on a Raspberry Pi that acts as a Client. The electron application will consist of HTML and JavaScript
- Edge Node Applications, these will be delivered as source code that contains a Dockerfile so it can be built and a script to deploy the built image as a Docker service on the Edge Devices.
- Data Centre Application that will take the form of an executable to be installed in IIS

References

[1]B. Intelligence. (2016,10,18). EDGE COMPUTING IN THE IoT: Forecasts, key benefits, and top industries adopting an analytics model that improves processing and cuts costs [Online]. Available: http://uk.businessinsider.com/edge-computing-in-the-iot-forecasts-

key-benefits-and-top-industries-adopting-an-analytics-model-that-improves-processing-and-cuts-costs-2016-7
[2]D. LeClair. (2014,7,22). The edge of computing: It's not all about the cloud [Online]. Available: http://insights.wired.com/profiles/blogs/the-edge-of-computing-it-s-not-all-about-the-cloud
[3] W. Hajji and F. Po Tso. (2016,6,6). Understanding the performance of low power raspberry pi cloud for big data [Online]. Available: http://www.mdpi.com/2079-9292/5/2/29/htm
[4] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. O'Brien, "Iridis-pi: A low-cost, compact demonstration cluster," Cluster Computing, vol. 17, no. 2, Jun. 2013. [Online]. Available: http://link.springer.com/article/10.1007/s10586-013-0282-7
[5]"Raspberry pi 3 model B" Raspberry Pi Foundation [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b
[6]"Electron" Electron [Online]. Available: http://electron.atom.io
[7]"Docker" Docker [Online]. Available: https://www.docker.com
[8] "Web API" The Official Microsoft ASP.NET Site, 2015. [Online]. Available: https://www.asp.net/web-api
[9] "Raspbian" [Online]. Available: https://www.raspbian.org
[10] "NGINX" NGINX [Online]. Available: https://www.nginx.com
[11] "Apache HTTP server project" [Online]. Available: https://httpd.apache.org
[12] "Home - Lighttpd - fly light" [Online]. Available: https://www.lighttpd.net
[13] "Node.js" [Online]. Available: https://nodejs.org/en
[14] "Redis" [Online]. Available: http://redis.io
[15] "Memcached" [Online]. Available: https://memcached.org
[16] I. Haber. "Why Redis beats Memcached for caching" InfoWorld, 2016. [Online]. Available: http://www.infoworld.com/article/3063161/application-development/why-redis-beats-memcached-for-caching.html
[17] "Home: The official Microsoft IIS site" 2016. [Online]. Available: https://www.iis.net
[18] S. Seshachala "Webinars & hangouts" in DevOps Toolbox, DevOps.com, 2014. [Online]. Available: https://devops.com/docker-vs-vms
[19] "What is a gateway?" 2000. [Online]. Available: http://whatismyipaddress.com/gateway
[20] "BT glossary". [Online]. Available: http://www.btglossary.co.uk/index.php?title=PCP
[21] lherrera and L. H. Benítez. "Poor man's load balancing with Docker" Medium, 2016. [Online]. Available: https://medium.com/@lherrera/poor-mans-load-balancing-with-docker-2be014983e5
[22] N. Takahashi, H. Tanaka, and R. Kawamura. "Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing - IEEE Xplore document" 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7130892
[23] "CMU Sphinx" [Online]. Available: http://cmusphinx.sourceforge.net
[24] C. Gaida, P. Lange, R. Petrick, P. Proba, A. Malatawy, and D. Suendermann-Oeft. "Comparing open-source speech recognition Toolkits" 2014. [Online]. Available: http://suendermann.com/su/pdf/oasis2014.pdf
[25] "Kaldi" [Online]. Available: http://kaldi-asr.org
[26] julius-speech, "Julius-speech/julius" GitHub, 2016. [Online]. Available: https://github.com/julius-speech/julius
[27] "Control everything with your voice". [Online]. Available: http://jasperproject.github.io
[28] "Weka 3 - data mining with open source machine learning software in java". [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka
[29] "TensorFlow — an open source software library for machine intelligence" 2016. [Online]. Available: https://www.tensorflow.org