Utilising Edge Computing in a 3-Tier system between Client Devices and a Data Centre

A dissertation submitted in partial fulfilment of the requirements for the degree of BACHELOR OF ENGINEERING in Computer Science

in

The Queens University of Belfast

by

Connor Dickson

9th May 2017

# Acknowledgements

* Hans
* Automated Intelligence
* Cassio

# Abstract

* 100 word outline of subject matter/findings

Table of Contents

[Acknowledgements 2](#_Toc476047612)

[Abstract 2](#_Toc476047613)

[Introduction and Problem Specification 3](#_Toc476047614)

[System Requirements Specification 3](#_Toc476047615)

[Design 4](#_Toc476047616)

[Architectural Description 4](#_Toc476047617)

[User Interface Design 4](#_Toc476047618)

[Software System Design 5](#_Toc476047619)

[Implementation and Testing 5](#_Toc476047620)

[Choice of implementation languages/dev environments 5](#_Toc476047621)

[Use of software libraries 5](#_Toc476047622)

[Key implementation details 5](#_Toc476047623)

[How each component was implemented 5](#_Toc476047624)

[Discussion of test approach 5](#_Toc476047625)

[Caching System Tests 5](#_Toc476047626)

[Client Application 5](#_Toc476047627)

[Edge Node Application 6](#_Toc476047628)

[Voice Recognition System Tests 7](#_Toc476047629)

[Client Application 7](#_Toc476047630)

[Edge Node Application 8](#_Toc476047631)

[Data Centre Application 9](#_Toc476047632)

[Machine Learning System Tests 9](#_Toc476047633)

[Client Application 9](#_Toc476047634)

[Edge Node Application 9](#_Toc476047635)

[Data Centre Application 9](#_Toc476047636)

[System Evaluation 9](#_Toc476047637)

[Experimentation 10](#_Toc476047638)

[Caching Experimentation 10](#_Toc476047639)

[Does utilising Edge Computing reduce the latency of requests for the Client Device? 10](#_Toc476047640)

[Voice Recognition Experimentation 11](#_Toc476047641)

[Does utilising Edge Computing reduce the computational load on the Client Device? 11](#_Toc476047642)

[Does utilising Edge Computing increase the latency of the request for the Client Device? 12](#_Toc476047643)

[Machine Learning Experimentation 13](#_Toc476047644)

[Load Balancing Experimentation 13](#_Toc476047645)

[Do the benefits of utilising Edge Computing deteriorate when multiple requests are made concurrently? Can this be improved with a custom load balancing aspect? 13](#_Toc476047646)

[Conclusion 14](#_Toc476047647)

[References 14](#_Toc476047648)

[Copyright 14](#_Toc476047649)

[Appendices 14](#_Toc476047650)

[User manual 14](#_Toc476047651)

[Test Results 14](#_Toc476047652)

[Minutes of meetings 14](#_Toc476047653)

# Introduction and Problem Specification

Background material which introduces the problem area, context and background.

Identify my problem

Systematically researched and fully analysed the problem

# System Requirements Specification

Precise description of the system developed. Should be updated for the final system delivered. List assumptions made about the problem and any system constraints(RAM?)

Functional and non-functional requirements should be complete, clear, accurate, feasible and objectively verifiable.

Read req’s for list of what can be included in this section

# Design

## Architectural Description



The entry point for the system is the Electron Application which runs on the Client Raspberry Pi

The Edge is comprised of 3 Raspberry Pi’s which form a Docker swarm. The applications run in containers that can be running on any of the Pi’s but the entry point is always the master who directs traffic. Traffic being directed is colour coded.

There are multiple data centres depending on what application is making the request. The WebAPI project is hosted in IIS

## User Interface Design

I tried to keep a consistent theme throughout my UI design



(Other pages might need to redo the homepage one also)

## Software System Design

Consistent code style.

* Role of each component and the interface between the components
* Clear correlation between design and specs

Design should be linked to requirements and give a critical discussion of key design decisions/styles/patterns. Read more in notes

# Implementation and Testing

## Choice of implementation languages/dev environments

* Electron that utilises nodejs for Client
* Docker/Nodejs for edge node
* WebAPI for Windows Server

## Use of software libraries

* Pocketsphinx for voice recognition
* Nodejs packages
* Testing suite

## Key implementation details

* Once I got the setup done.
* Deciding on redis because of docker image
* Deciding on node/webapi
* Read notebook
* Proxy instead of regular web server. This allows for extraction of URL’s on edge node and makes it seem more like real life

## How each component was implemented

* Client
* Edge
* Data Centre

## Discussion of test approach

* Because of the nature of the code it could prove difficult to test as it has to run in a nodejs server and made actual requests so there was a lot of manual testing
* Manual Testing
* Unit tests in WebAPI proved difficult also but I wrote some with the code that doesn’t deal with actual requests

## Caching System Tests

### Client Application

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Method | Expected Results | Passed |
| Can reach Caching Application page | Click on the Client Application button on the home page | The Caching Application page is now launched |  |
| Home button | Click home button | Should be returned to the home page |  |
| URL bar is present | On the caching page ensure there is a URL bar | The URL bar is present |  |
| Can type into URL bar | Click into the URL bar and type an address | The address types will show up in the URL bar |  |
| Go button without URL typed | Click the Go button when there is no URL typed out | Nothing, there will be no error |  |
| Go button with an invalid URL typed | Click the Go button when an invalid URL is typed into the URL bar | A message will appear warning the user that the request failed |  |
| Go button with a valid URL typed | Click the Go button when a valid URL is typed | The requested webpage will be fetched and displayed to the user |  |
| The timer | Click the go button with a valid URL | The timer will start and when the request is finished will record the total time for the request to finish |  |
| Clear cache button | Click the clear cache button | Information will appear to indicate that the cache on the edge node was cleared successfully |  |
| Can connect to the proxy | Make a web request | Ensure that the proxy is utilised |  |

### Edge Node Application

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Method | Expected Results | Passed |
| Can connect to the Redis instance | Deploy the web server | The server will automatically connect to Redis |  |
| Can make request to Caching Web Server | Make a curl request to the Edge Node using the host name and port number | Receive Data back |  |
| Receive clear cache request | Call the Edge Node with a ClearCache request | The clear cache command is executed using the redis-cli and the cache is cleared. This information is then returned to the user |  |
| Receive new request | Call the Edge Node with a new URL | The content is retrieved, stored in Redis and returned to the client |  |
| Receive a request for the second time | Call the Edge Node with a URL that has already been requested | The content is retrieved from Redis and returned to the user |  |

## Voice Recognition System Tests

### Client Application

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Method | Expected Results | Passed |
| Can reach the Voice Recognition page | Click on the Voice Recognition button on the home page | The Voice Recognition Application page is now launched |  |
| Home button | Click home button | Should be returned to the home page |  |
| Recording status after page launch | Launch the Voice Recognition application | The recording status should be “Not Recording” |  |
| Recording status after clicking record | Click the record button | The recording status should change to “Recording” |  |
| Recording status after stopping recording | Click the stop recording button after starting a recording | The recording status should change to “Finished recording” |  |
| Recording button when not recording | Stop a recording or don’t start a recording | The recording button should read to “Start Recording” |  |
| Recording button when recording | Start a recording or don’t stop a recording | The recording button should read “Stop Recording” |  |
| Execute local recognition button without recording | Press the Execute Local Recognition button without recording a phrase | Nothing should happen, no error should be thrown |  |
| Execute local recognition button after recording | Press the Execute Local Recognition button after recording a phrase | The execution of the recording should be performed locally |  |
| Execute remote recognition button with no recording | Press the Execute Remote Recognition button without recording a phrase | Nothing should happen, no error should be thrown |  |
| Execute remote recognition button after recording without edge processing | Press the Execute Remote Recognition button after recording a phrase and don’t tick the pre-process checkbox | The Execution of the recording should be performed remotely on the Data Centre |  |
| Execute remote recognition button after recording with edge processing | Press the Execute Remote Recognition button after recording a phrase and tick the pre-process checkbox | The Execution of the recording should be performed remotely on the Edge Node and the results sent to the Data Centre |  |
| Local Results section when no local evaluation has occurred | Launch the Voice Recognition Application and don’t perform local evaluation | The results section is empty |  |
| Local Results section when the local evaluation has occurred | Perform local evaluation | The local results section should show statistics |  |
| Remote Results section when no remote evaluation has occurred | Launch the Voice Recognition Application and don’t perform remote evaluation | The results section is empty |  |
| Remote Results section when no remote evaluation has occurred | Perform remote evaluation | The remote results section should show statisticss |  |

### Edge Node Application

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Method | Expected Results | Passed |
| Can make request to Voice Recognition Web Server | Make a curl request to the Edge Node using the hostname and port number | Receive Data back |  |
| Receive Get Request | Make a get request to the Edge Node | No error but no data returned |  |
| Receive Put Request | Make a put request to the Edge Node | No error but no data returned |  |
| Receive post request with invalid data | Make a post request with an invalid audio file or incorrect data | No error but no data is returned |  |
| Receive post request with valid recording | Make a request with a valid voice recording | A JSON object is returned with information about the request and the audio spoken |  |
| Receive valid request with the pre-process header | Make request with the pre-process request header set | The processing of the voice file happens on the Edge Node |  |
| Receive valid request without the pre-process header | Make request without the pre-process request header set | The voice recording is sent to the data centre for processing and the results returned to the user |  |
| Request information from the Data Centre when the WebAPI is not published | Remove the WebAPI and make a request | The service should record an error |  |
| Request information from the Data Centre when the WebAPI is published | Publish the WebAPI and make a request | The information should be returned correctly |  |
| Measure CPU use | Make a regular request | The CPU use on the edge node should be returned as part of the request |  |
| Measure Time of request | Make a regular request | The Time taken to process the request should be recorded |  |

### Data Centre Application

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Method | Expected Results | Passed |
| Receive invalid request to process voice | Make a valid POST request with binary voice data | The voice is processed correctly and a valid response is produced |  |
| Receive valid request to process voice | Make an invalid POST request with invalid voice data | The voice request should be handled correctly and gracefully respond |  |
| Receive invalid request to record information | Make a valid POST request with textual data from a pre-processed request | The request should be processed correctly and a response generated |  |
| Receive valid request to record information | Make an invalid POST request without valid textual data | The request should be handled gracefully and a response sent |  |

## Machine Learning System Tests

### Client Application

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Method | Expected Results | Passed |
|  |  |  |  |

### Edge Node Application

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Method | Expected Results | Passed |
|  |  |  |  |

### Data Centre Application

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Method | Expected Results | Passed |
|  |  |  |  |

# System Evaluation

Not sure what’s included here

# Experimentation

A lot of the emphasis in this project will be here.

Provide empirical results and draw conclusions.

Describe methodology (use experiment methodology like goals and hypothesis, what’s measured/controlled)

When is it good to use edge node. When is it good to use DC? Not always good to use both

Read more in notes

## Caching Experimentation

### Does utilising Edge Computing reduce the latency of requests for the Client Device?

Setup;

* Switch connecting all Pi’s to router through 100mbit powerline adapter
* Run the Caching Application on the Client Device
* Deploy the Caching Service and the Redis Instance to the Edge Device
* Publish the Data Centre WebAPI

Isolate Variables;

Constants:

* Internet speed
* The web page being loaded
* The background processing being used on the Pi

Variables

* Whether a webpage is cached or not

The variable in this experiment is whether the webpage is cached when I request the data.

Method;

Measurements recorded will be the time between when the WebView element in the Client Application starts a request and ends a request.

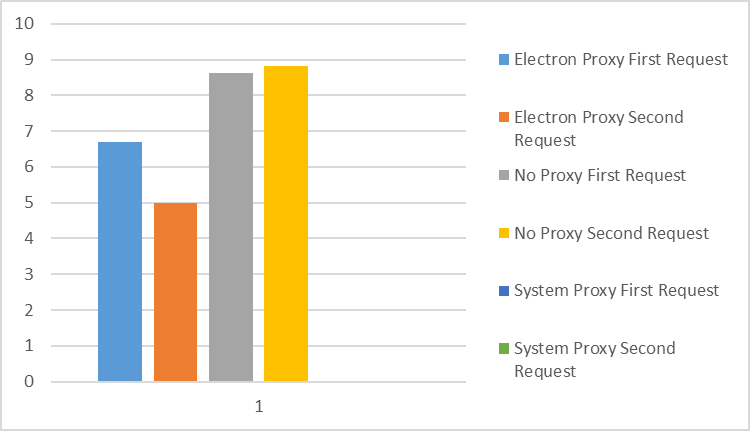
After the single Caching Application and single Redis server has been deployed 10 warmup requests will performed without wiping the cache. The experiment will be performed with 10 iterations and an average will be taken.

An iteration will consist of an initial request to “http://www.bbc.co.uk” and then a subsequent request when the information is already cached on the edge node. The two times will then be recorded in seconds and the cache will be cleared ready for the next request

Hypothesis;

The hypothesis is that the cached request should take less time to execute than the initial request.

Results;



Analyse;

Conclusion;

## Voice Recognition Experimentation

### Does utilising Edge Computing reduce the computational load on the Client Device?

Setup;

* Run the voice recognition application on the Client Device
* Save a voice recording using the Client Application
* Deploy the Voice Recognition Service to the Edge Device
* Publish the Data Centre WebAPI

Isolate Variables;

Constants:

* Internet speed
* The voice recording
* The software package being used
* The Language Model and Dictionary being used for the processing
* The CPU usage from applications that aren’t the voice recording

Variables

* Where the voice processing is occurring

The variable in this experiment is whether I process the file locally on the Client Device, remotely pre-process it on the Edge Device or remotely process it on the Data Centre.

Method;

Measurements recorded will be:

* CPU load in a percentage of the total CPU
* Processing time in seconds
* File size
* Size of the voice model being used
* Length of voice recording

After the Voice Recognition applications have been deployed 10 warmup requests will be performed on Client Device, a further 10 will be performed on the Edge Node and a final 10 on the Data Centre.

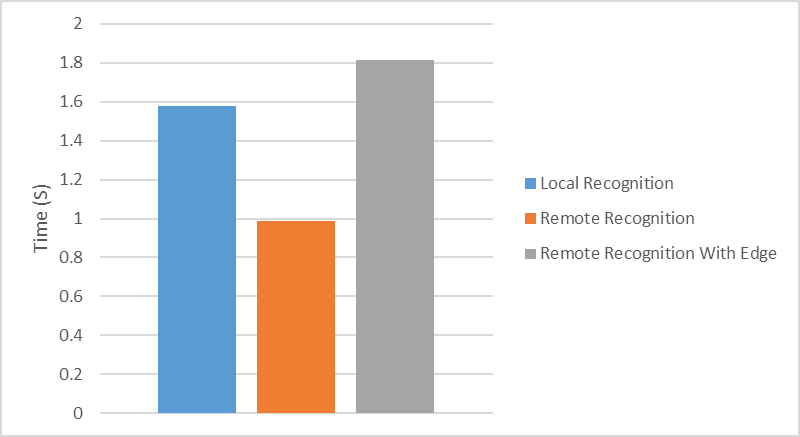
The measure of time will be recorded on the Client Device from as soon as the request starts to when the request finishes. The experiment will be repeated 10 times and an average will be taken.

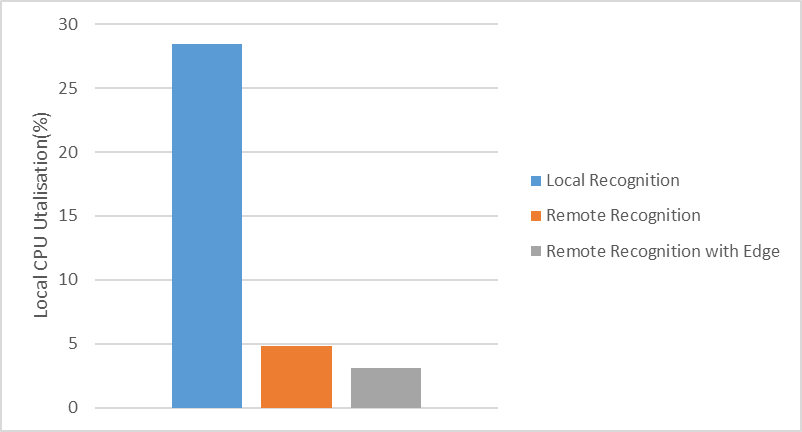
The experiment will consist of a voice being recorded. The same recording will be used throughout the experiment to allow for a fair comparison of computational load.

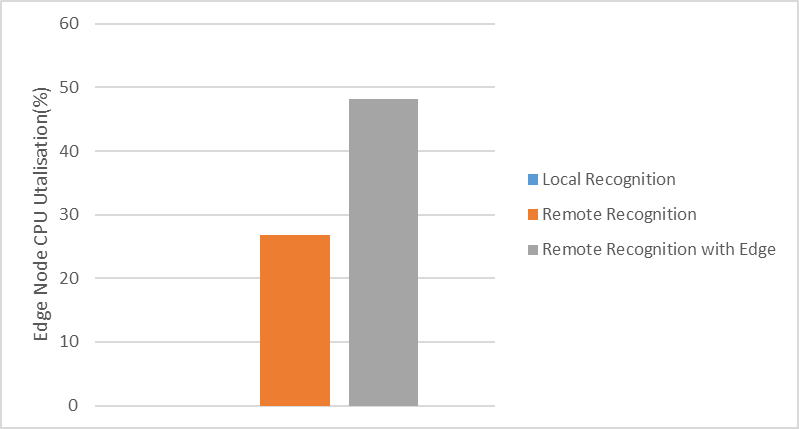
Hypothesis;

The hypothesis is that the Data Centre should process the request the fastest as it has the most powerful CPU but the CPU strain on the Client Device should be reduced regardless of the request being pre-processed on the Edge Device or the Data Centre.

Results;







Analyse;

Conclusion;

### Does utilising Edge Computing increase the latency of the request for the Client Device?

Setup;

Isolate Variables;

Method;

Hypothesis;

The latency of the request should not increase much if the request is pre-processed on the Edge Device as both devices are Raspberry Pi’s however the request should be processed much quicker on the Data Centre as it has a faster processor.

Results;

Analyse;

Conclusion;

## Machine Learning Experimentation

## Load Balancing Experimentation

### Do the benefits of utilising Edge Computing deteriorate when multiple requests are made concurrently? Can this be improved with a custom load balancing aspect?

Setup;

* Run the Stress Testing application on the Client Device
* Deploy the Voice Recognition service to the Edge Device
* Publish the Data Centre WebAPI

Isolate Variables;

Constants;

* Number of requests
* The Voice Recording
* The software package being used
* The Language Model and Dictionary being used for Processing
* The CPU use from other applications

Variables;

* Whether the deployed Edge Node service is utilising custom Load Balancing

Method;

Record CPU use and Time to finish request over a set number of request

Perform a set number of requests and record the CPU use.

Perform the same number of requests but with the Load Balancing application deployed

Hypothesis;

Adding load balancing will improve response times

Results;

Analyse;

Conclusion

# Conclusion

General summary of the success of the project with respect to criteria identified in the intro

Discussion of significance of experimental results. Agree with others work?

Strengths and weaknesses

Evaluation of hardware/software

Looking for critical appraisal and significance of contribution in the context of wider work.

# References

* First ref should be URL to code gitlab

# Copyright

* Electron
* Nodejs/ Nodejs libraries
  + Node JS proxy
* Pocketsphinx
* Testing Suites
* Redis/Docker src images

# Appendices

## User manual

## Test Results

Excel results?

## Minutes of meetings