Load Test Results

Enrol Online

For

Electoral Commission

By

Catalyst.Net Limited

Version 1.0

2/09/2025

Commercial in Confidence

open source technologists

Level 6, Catalyst House, 150 Willis Street, Wellington 6011
PO Box 11053, Manners Street, Wellington 6142, New Zealand
+64 4 499 2267 // enquiries@catalyst.net.nz // www.catalyst.net.nz

Table of contents

1 Overview

This document details the load testing of the Enrol Online AJAX endpoints. The load tests were run against the Auckland DR site and the newly configured instance of the Mike system in the Catalyst Cloud Hamilton region.

The Electoral Commission's goal for the 2020 General Election is to be able to be able to meet peak loads of 70,000 transactions* per hour, with a ratio of 3 search only transactions (where a user finds their record and does not need to proceed to an upload or enrolment) to 2 enrolment or update transactions (roughly 42,000 search-only and 28,000 enrolments). This usage pattern and target is based on predictions drawn from the peak usage at the 2017 General Election and the changes in usage trends observed since the 2017 General Election.

* For the purpose of this test and report, a "**transaction**" is the complete multi-step user journey that a user completes in their website visit (see section 7 for full details of the transactions used).

2 Executive summary

Although previous tests have been run on Catalyst Cloud, those results are not directly comparable with the test results discussed in this document due to changes in the predicted peak load, the definition of a transaction, and the construction of those transactions in the load test framework.

The maximum throughput achieved was:

- 101,022 transactions per hour on the Hamilton cloud instance
- 51,860 transactions per hour on Auckland DR

The Cloud instance exceeded the peak usage goal. However, there is some risk associated with migrating the production site to the Hamilton cloud instance before the Election and some tasks to complete before that could happen.

The existing Auckland DR site performed significantly less well. This result is indicative of the expected peak usage that could be met by the – largely identical – Wellington site. The risks associated with leaving the production system in its current location (Wellington) relate mostly to the risk of failure due to the age of the hardware, lack of warranty support, and the out of support operating system.

2.1 Considerations and risks

- The impact of 6(c), 9(2)(k), or any similar DDOS protection, on the application has not been explored and no load test of the application has been performed in that context. This implementation may require application changes and affect performance.
- Network and bandwidth considerations of accessing the site from public networks are out of scope of this test. This load testing only accessed the application from hosts on the Catalyst Cloud Hamilton region.
- In the last few weeks some critical systems have had disk failure and have needed replacements to be purchased and installed.

- The production database servers in the hardware (Wellington and Auckland) clusters are nearing their disk capacity – this is exacerbated by the increase in forms around election time and by hosting UAT databases on production hardware.
- There is a significant amount of application time spent on application analytics reports. As a risk mitigation, development has been completed to give EC the option of switching off the analytics requests. This would stop the enrolment dashboard from working, but increase application performance for users. More investigation would need to be carried out to identify what difference this change would make to the user experience and peak load handling.
- The increase in other applications connecting to the Mike sytem, including the eRoll application and the new Overseas application, may have an impact on performance during peak periods. Some of this extra load may be able to be mitigated if necessary.

3 Test strategy

The load tests were performed using the existing EOL load test framework which emulates a web browser making requests. This approach ensures that the full application path is tested and that the response times reflect the end user experience.

The test load was a mixture of the following transactions (the percentage varies slightly between the two load tests due to the nature of writing the test scripts so figures are provided in Sections 4 and 5):

- Change details with drivers licence
- Change details with passport
- Change details without ID
- Enrol with drivers licence
- Enrol with passport
- Enrol without ID
- Search only

Changes from previous tests:

- The original test mix did not contain any search only transactions.
- The number of address completion calls has been reduced to match that seen in production.

Multiple test runners were created in the Hamilton cloud region, and the processing scripts were updated to handle multiple runners.

4 Test environment

The load test tool was run from multiple servers in the Hamilton cloud region and connected to:

- The Mike system running in the Hamilton cloud region.
- The Mike system on the Auckland DR environment.

The Auckland DR environment's configuration and hardware is identical to the current Wellington Production environment and is a suitable indication of production capability. The Staging environment – where previous tests have been performed - has significantly less available hardware resources.

Prior to testing against the Auckland DR site, the risk of changes was discussed by EC and Catalyst technical staff. EC accepted the risk that if a failure that required a DR switch occurred during the testing period, then the time to restore production on the DR environment would be longer than usual due to the extra configuration changes required.

5 Hamilton Cloud environment results

A run of 160,000 user transactions was made against the Enrol Online application running in the Hamilton cloud region.

The maximum rate achieved was 101,022[1] transactions per hour:

- 37,378 enrolments
- 63,644 search

The following transactions were tested:

- Change details with drivers licence (24,640)
- Change details with passport (8,960)
- Change details without ID (7,840)
- Enrol with drivers licence (7,680)
- Enrol with passport (4,480)
- Enrol without ID (5,600)
- Search (100,800)

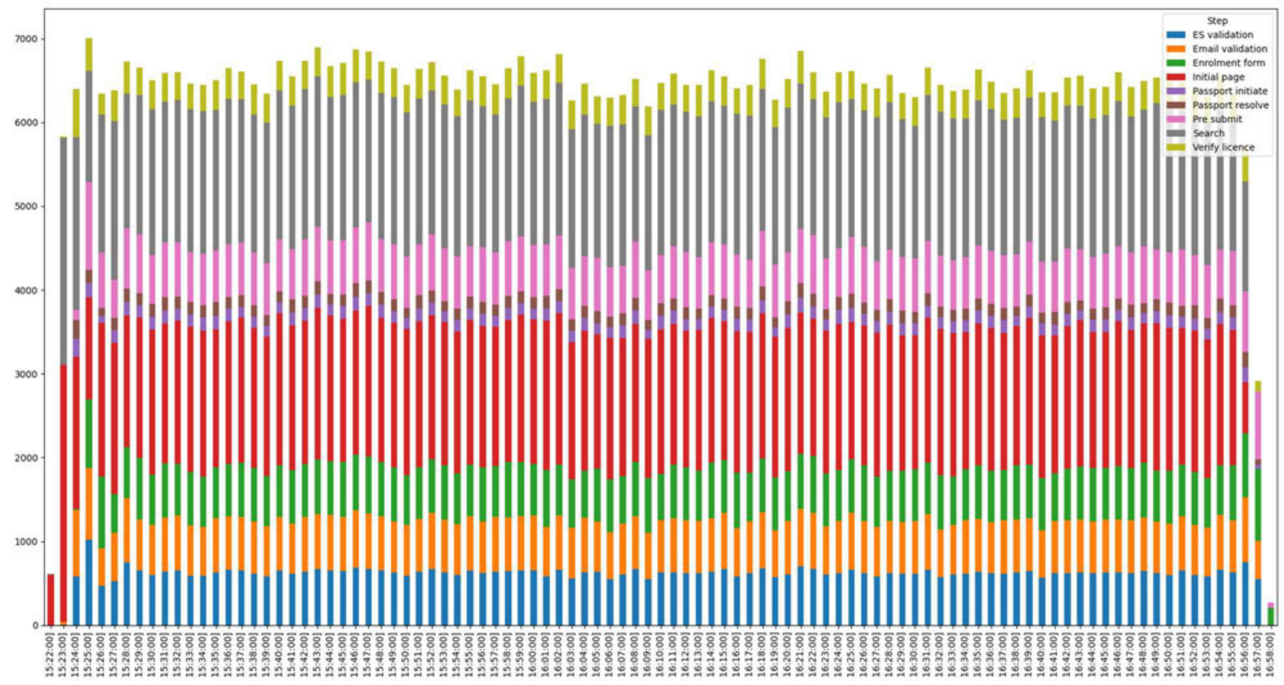
Summary statistics

AJAX request	Requests	Min	Max	Median	95th %	Mean	Failed[2]	>10 sec
Address completion	875,416	0.02	3.13	0.09	0.16	0.10	0	0
Analytics	772,098	0.06	4.40	2.96	3.44	2.92	7	0
ES validation	59,181	0.03	3.70	2.71	3.18	2.66	0	0
Email validation	59,181	0.05	4.98	0.15	4.13	0.43	0	0
Enrolment form	59,178	0.21	6.07	4.13	4.71	4.05	0	0
Initial page	160,000	0.12	5.95	3.42	3.92	3.40	1	0
Passport initiate	13,434	0.11	4.27	3.37	3.86	3.32	0	0
Passport resolve	13,434	0.13	4.40	3.43	3.92	3.38	0	0
Pre submit	59,181	0.14	7.93	3.48	6.44	3.60	0	0
Search	159,995	0.17	18.55	3.60	4.11	3.58	4	2
Verify licence	32,307	0.12	5.00	3.48	3.99	3.44	0	0

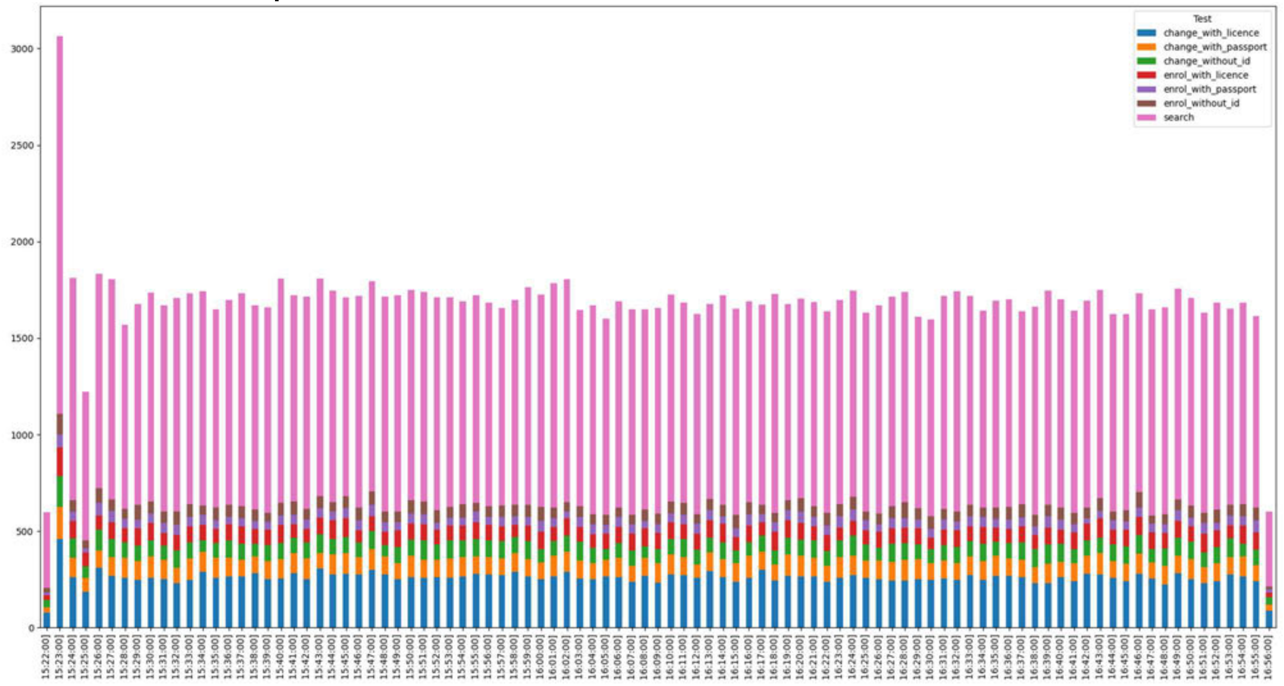
[1]This rate was calculated for the period from 15:26 to 116:55, to avoid test runner start up and completion effects.

[2]All the failures occurred during the test runner start up phase.

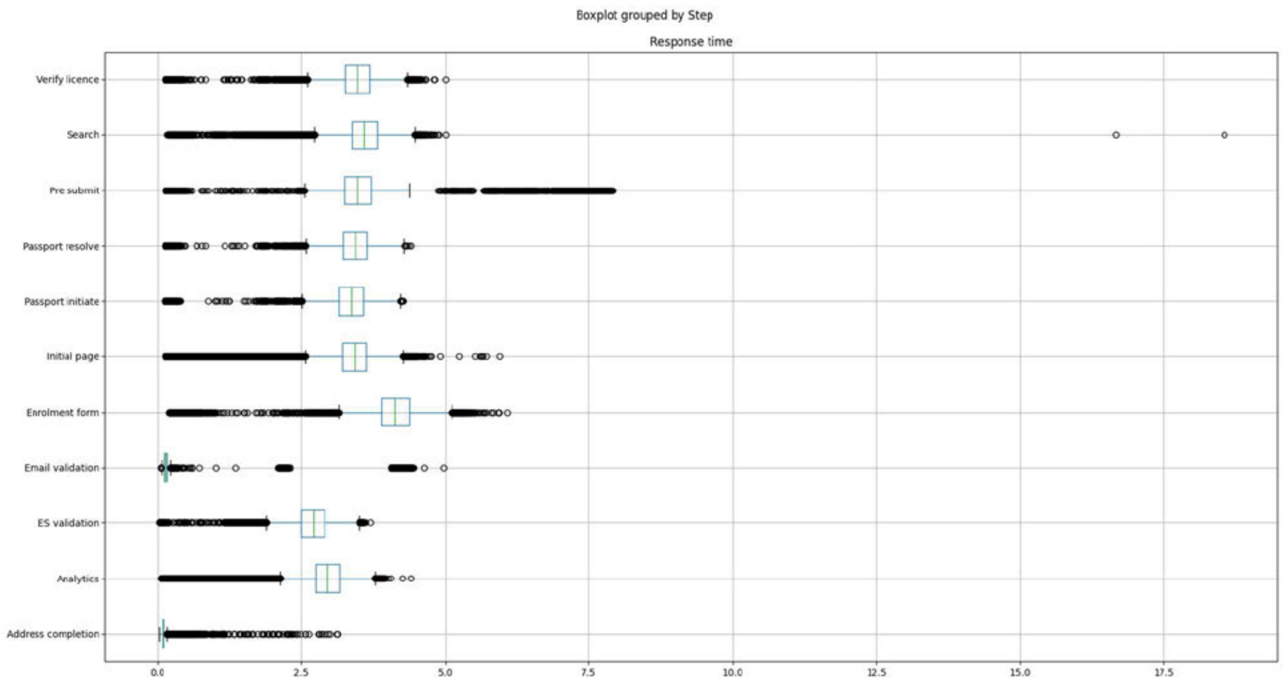
AJAX requests per minute



Transactions started per minute



AJAX response time box plot



5.1 Considerations

At various stages in the load testing Catalyst was met with barriers/bottlenecks. We attempted to change as little as possible to ensure any improvements to performance could be made without development or application changes required.

Barrier	Mitigation
Lack of available CPU and RAM resources on the front-end web server	Allocated extra CPU and RAM resources to the affected server. The cloud system makes it easy to vary the resources available to the system to meet demand
Lack of available processes on the front-end web server	The application configuration was change to tune the number of worker processes, allowing more requests to be serviced simultaneously
Not enough test runners to push capacity on the system	Increased the number of test runners

6 Auckland DR environment results

A run of 12,000 user transactions was made against the Auckland DR environment

The maximum rate achieved correlated to 51,860 requests per hour.

- 20,744 enrolments
- 31,116 searches

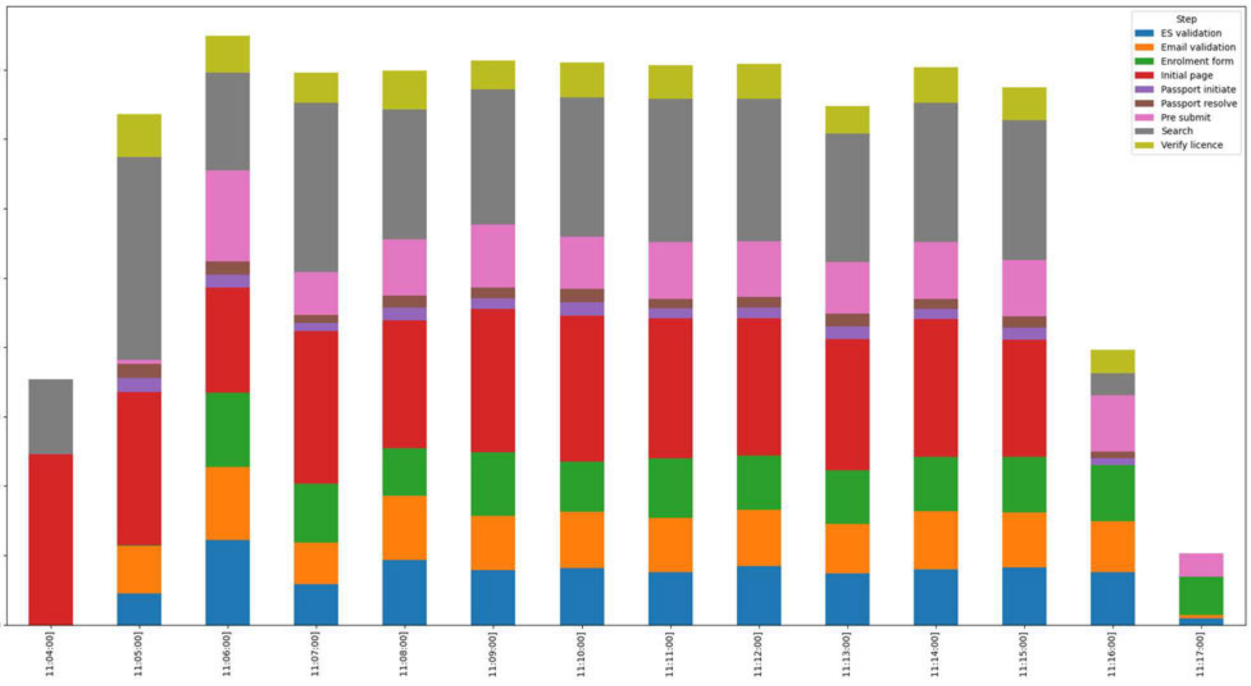
The following transactions were tested:

- Change details with drivers licence (1,760)
- Change details with passport (640)
- Change details without ID (560)
- Enrol with drivers licence (1,120)
- Enrol with passport (320)
- Enrol without ID (400)
- Search (7,200)

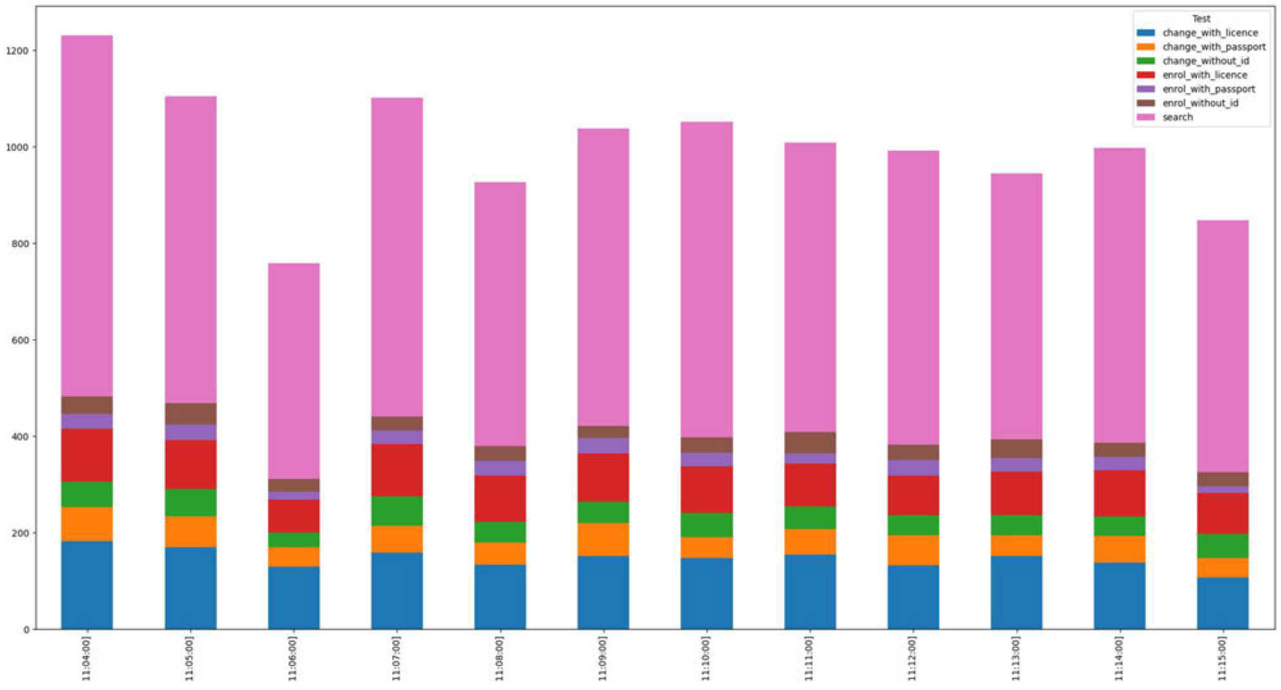
Summary statistics

AJAX request	Requests	Min	Max	Median	95th %	Mean	Failed	>10 sec
Address completion	67,087	0.07	7.81	0.14	0.53	0.20	0	0
Analytics	60,866	0.08	15.47	0.14	0.44	0.18	0	1
ES validation	4,799	0.07	3.75	0.12	0.21	0.15	0	0
Email validation	4,799	0.07	4.02	0.13	0.23	0.16	0	0
Enrolment form	4,798	0.14	3.82	0.24	0.45	0.28	0	0
Initial page	12,000	0.11	16.04	0.17	0.58	0.25	0	1
Passport initiate	960	0.10	3.73	0.16	0.32	0.20	0	0
Passport resolve	960	0.10	3.74	0.16	0.36	0.21	0	0
Pre submit	4,799	0.10	7.85	0.15	0.24	0.18	0	0
Search	12,000	0.11	7.64	0.18	0.55	0.24	0	0
Verify licence	2,879	0.10	16.11	0.18	0.36	0.36	0	1

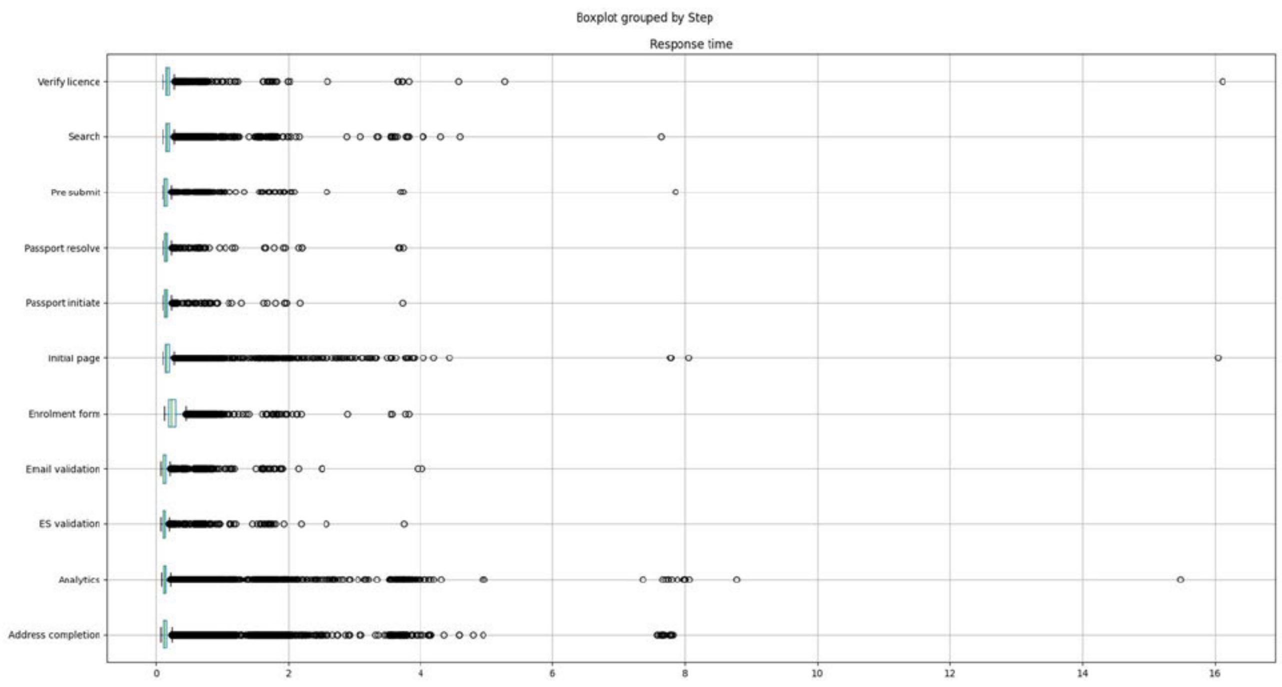
AJAX requests per minute



Transactions started per minute



AJAX response time box plot



6.1 Considerations

At various stages in the load testing Catalyst was met with barriers/bottlenecks. We attempted to change as little as possible to ensure any improvements to performance could be made without development or application changes required.

Barrier	Mitigation
Lack of available processes on the front-end web server	Changes made to service configuration to increase the number of worker processes, allowing more requests to be serviced simultaneously
Lack of available CPU and RAM resources on the front-end web server	It was not possible to modify the hardware to address this limitation. Changes would require new hardware procurement

7 Transaction details

This section details the AJAX endpoints called by each user journey transaction. This simulates a user moving through the pages of the site in a normal manner. All transactions make multiple analytics calls to record the user's site interactions for processing by the Enrolment Dashboard (these are not shown in the transaction breakdowns below).

Change details with drivers licence

- Initial page
- Address completion
- Search
- Licence verification

- Email validation
- ES validation
- Address completion
- Pre submit
- Enrolment form

Change details with passport

- Initial page
- Address completion
- Search
- Passport initiate
- Passport resolve
- Email validation
- ES validation
- Address completion
- Pre submit
- Enrolment form

7.1

Change details without ID

- Initial page
- Address completion
- Search
- Email validation
- ES validation
- Address completion
- Pre submit
- Enrolment form

Enrol with licence

- Initial page
- Address completion
- Search
- Licence verification
- Address completion
- Email validation
- ES validation
- Pre submit
- Enrolment form

Enrol with passport

- Initial page
- Address completion
- Search
- Passport initiate
- Passport resolve
- Address completion
- Email validation
- ES validation
- Pre submit
- Enrolment form

Enrol without ID

- Initial page
- Address completion
- Search
- Address completion
- Email validation
- ES validation
- Pre submit
- Enrolment form

Search

- Initial page
- Address completion
- Search

7.2 AJAX endpoint URLs

Endpoint	URL
Initial page	/app/enrol
Address completion	/js-ac/anon/residential-address
Search	/app/enrol/ajax/search
Email validation	/js-ac/anon/email-valid
ES validation	/app/enrol/ajax/esvalidator
Enrolment form	/app/enrol/ajax/enrolment_form
Licence verification	/app/enrol/ajax/nzta_dvl_verify
Passport initiate	/app/enrol/ajax/pp_assertion_initiate
Passport resolve	/app/enrol/ajax/pp_assertion_resolve
Pre submit	/app/enrol/ajax/pre_submit_processing
Analytics	/app/enrol/ajax/enrolment_analytics

Vote.nz – Load Testing Report

Version 0.1

September 2023

Version Information

Version	Date	Author	Changes
0.1	08/09/2023	Jonathan Pot	Initial Draft
0.2	12/09/2023	Jonathan Pot	Update to Draft

Table of Contents

1 Executive Summary

1.1 Background

The purpose of this Test Exit Report is to document the results and findings of the load testing undertaken by Electoral Commission for the vote.nz website. In preparation for the upcoming 2023 elections, it is crucial for the website to effectively handle the projected workloads and accommodate future growth. EC Systems Analyst have been engaged to conduct performance testing on the vote.nz website and ensure its capacity to manage the anticipated volumes of visits.

1.2 Test Approach

Smoke tests were conducted on the system to ensure both the environment and test is working correctly, before performing any big performance tests on the system.

To replicate this traffic, the website Dotcom monitor was used to simulate the API throughput. The scripts were created JMeter and then uploaded in Dotcom Monitor. All tests were executed when the traffic on vote.nz is the lowest.

Dotcom monitor is a web-based application that allows you to perform different kinds of load tests. The following tests were conducted:

Test Scenario	Description
Concurrent test scenario **500 Concurrent users**	Simulate the behaviour of 500 public users visiting the website and navigation on different pages of the website. Assess the results and run the next test of 1000 concurrent users
Concurrent test scenario **1000 Concurrent users**	Simulate the behaviour of 1000 public users visiting the website and navigation on different pages of the website. Assess the results.

1.3 Key Findings and Observation

Concurrent test:

- Vote.nz can is capable of handling 1000 concurrent users accessible the website and navigating through different pages. Both the 500 & 1000 concurrent users test executed successfully with no major errors. When looking at the reports the main errors were validation errors and a few runtime errors on the big load.
- All tests were done on the production website outside of business hours.
- The 1000 concurrent users test was done twice. The first test concluded that the load was pushing the response time to be longer, approximately 10 seconds to load page at the pick. After the scaling up the server we can see that the response time was cut by half averaging at 5 seconds.
- The 500 concurrent test was done before changes were done on the server. Rerunning it would show lower response time on the pages

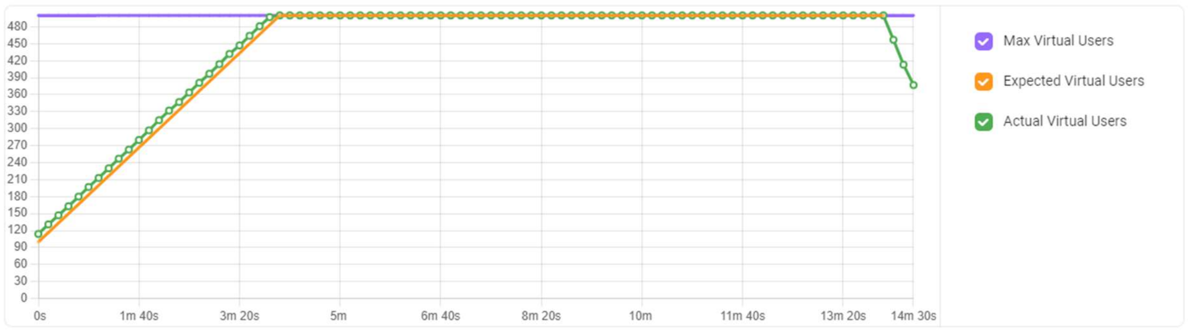
1.4 Load Test Setup

The load tests were to simulate a set of concurrent users accessing vote.nz. In the JMeter script the users would navigate through out multiple pages on the website, checking how to enrol online, visiting FAQs, checking information about the Māori electoral option and looking at key dates for the 2023 General Election.

Both tests had a delay set to a minimum of 3 seconds and a maximum of 6 seconds. The delay function adds programmed pause between steps in the monitoring script execution process which simulates the behaviour of a real user reading through pages..

1.4.1 500 Concurrent users

2 different devices hosted in Datacentres in Australia were used to simulate the pick of 500 users connected on our website.

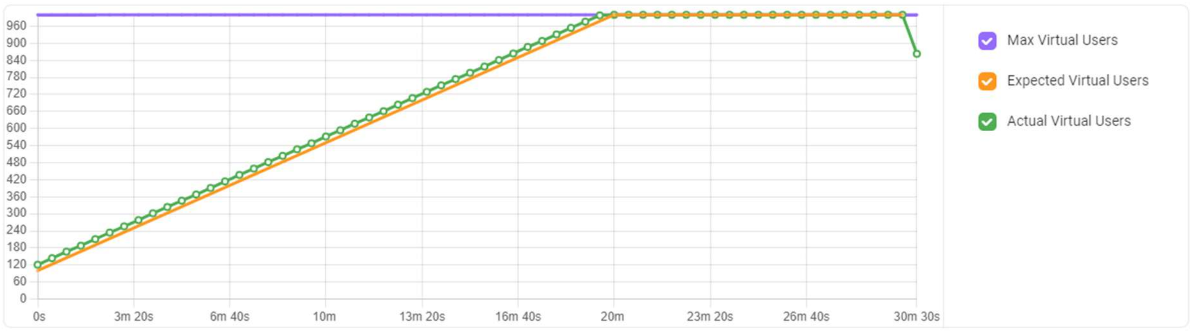


The test planned to start with 100 concurrent user and scale by 100 users every minute for 4 minutes. 500 concurrent users to hold for approximately 10 minutes.

Load coming from Asia Pacific, Sydney – AWS, Australia East, New South Wales – Azure with a 50% split between them two.

1.4.2 1000 Concurrent users

2 different devices hosted in Datacentres in Australia were used to simulate the pick of 1000 users connected on our website.



The test planned to start with 100 concurrent user and raised by 45 users every minute for 20 minutes. 1000 concurrent users to hold for approximately 10 minutes.
Load coming from Asia Pacific, Sydney – AWS, Australia East, New South Wales – Azure with a 50% split between them two.

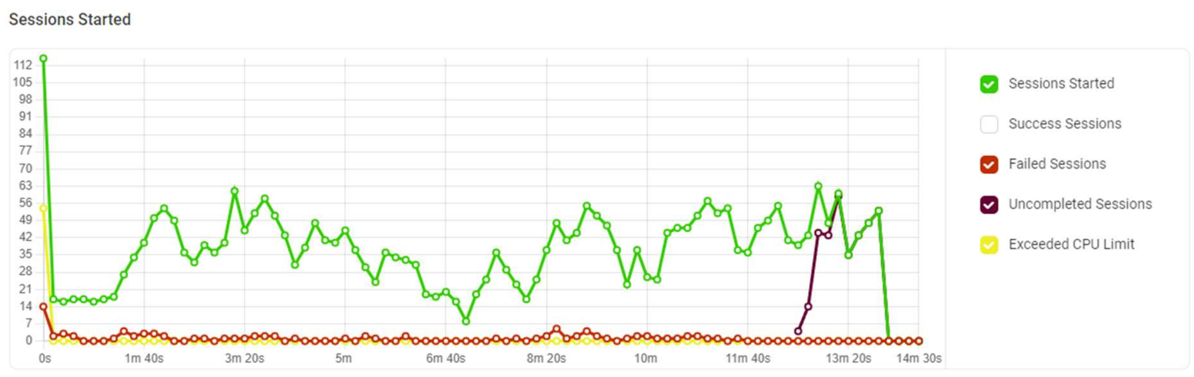
2 Load test results

2.1 500 concurrent users

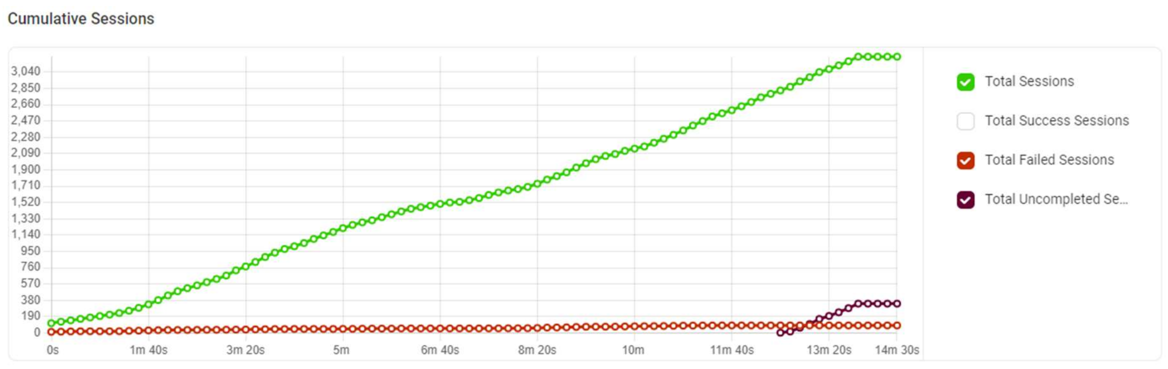
The test was a scheduled test that started on the 31st of August 2023 at 03:30:00AM for a duration of 14 minutes.

A total of 3220 session were created with 86% that were succeeded and 3% that failed. 11% were uncompleted due to validation errors. An average of 220 sessions per minute were created.
The average time a simulated user would have spent on the website was 168 seconds.

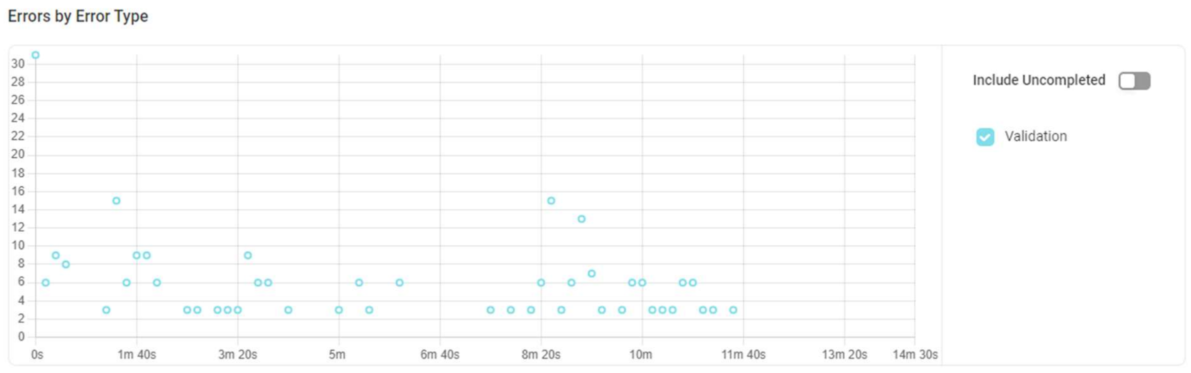
The blow graph shows the number of sessions that were started throughout the test. We can see that 12 minute in the script a few sessions were uncompleted due to the length of the test.



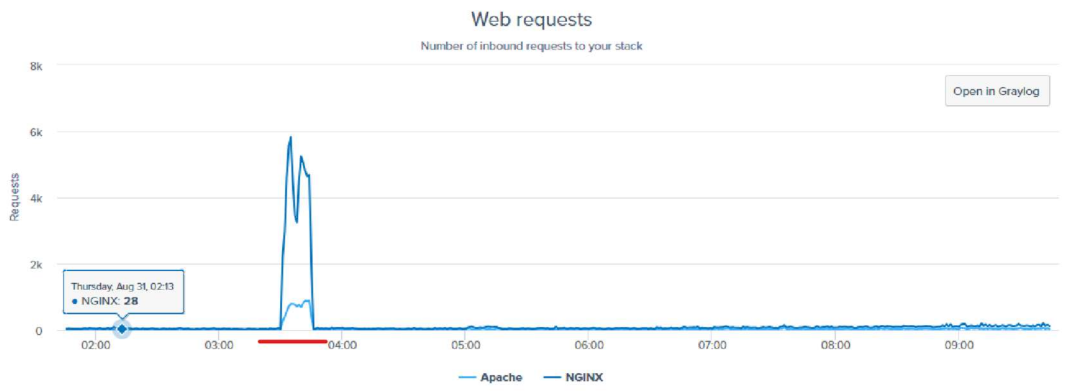
The graph below shows the number of cumulative sessions slowly ramping up throughout the test. 12 minutes in the test we can see those uncompleted sessions due to length of the test (unable to finish script).



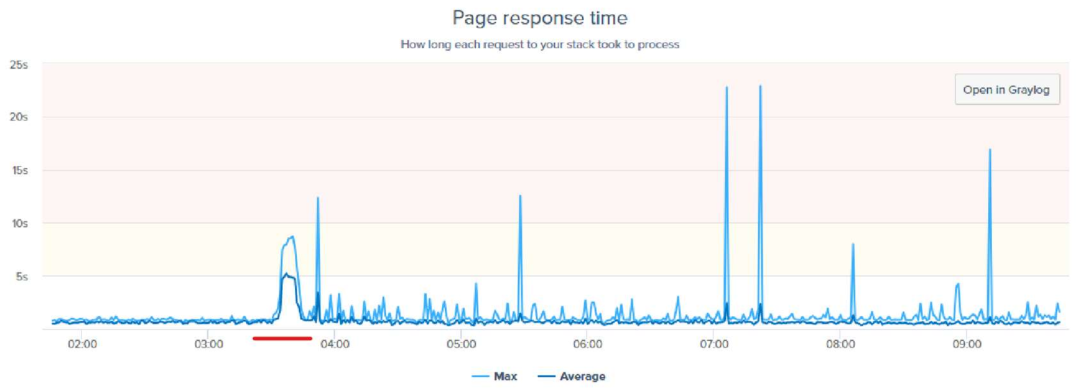
The following show the number of errors and errors by type during the test. Only validation errors happened during this test. This can be due to a variety of factors and specific use cases, such as page processing logic changes, web server behaviour, browser behaviour on a particular request.



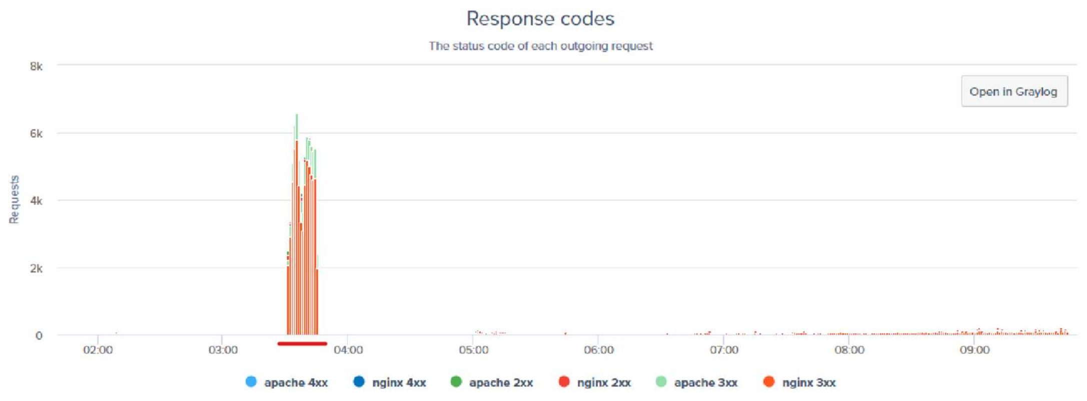
The below shows the number of web request that were done on the server of the website peaking at 6000 requests. Highlighted in red the timeline of the test.



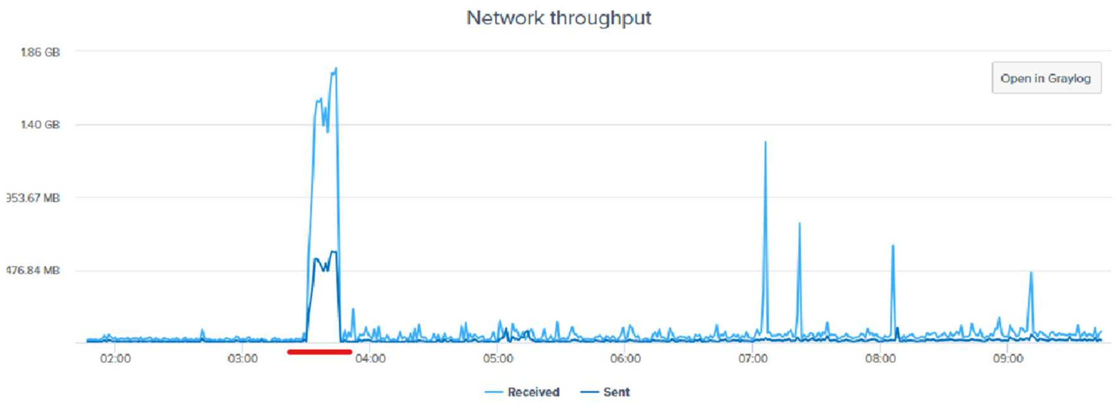
Page response time was in average 5 seconds to open a page with a maximum of 9 seconds for some users. Highlighted in red the timeline of the test.



We can see the response codes are what we would expect during a load test. The response codes received are what you would expect.



The network throughput sent an average of 590~ MB and received nearly 1.9GB.



2.2 1000 concurrent users

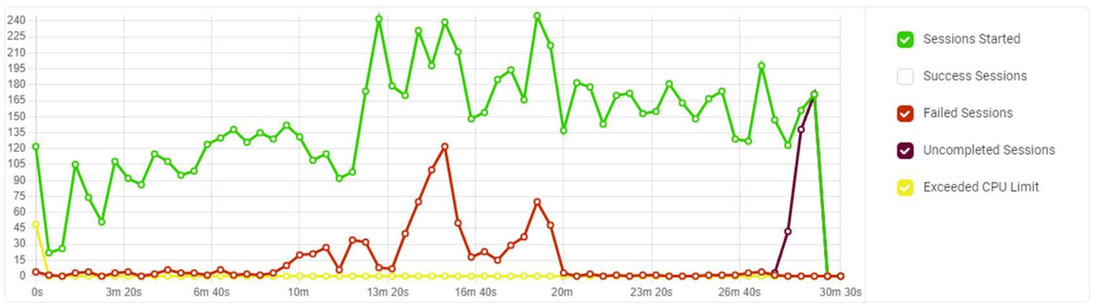
A first test was executed on the 1st of September, but the page response time was too high. A request was done to silverstripe to scale up our Virtual Machine to run a second test.

The second test was a scheduled test that started on the 8st of September 2023 at 03:00:00AM for a duration of 30 minutes.

A total of 8699 sessions were created with 86% that were succeeded and 10% that failed and 4% were uncompleted due to validation errors. An average of 280 sessions per minute were created. The average time a user would spend on the website was 157 seconds.

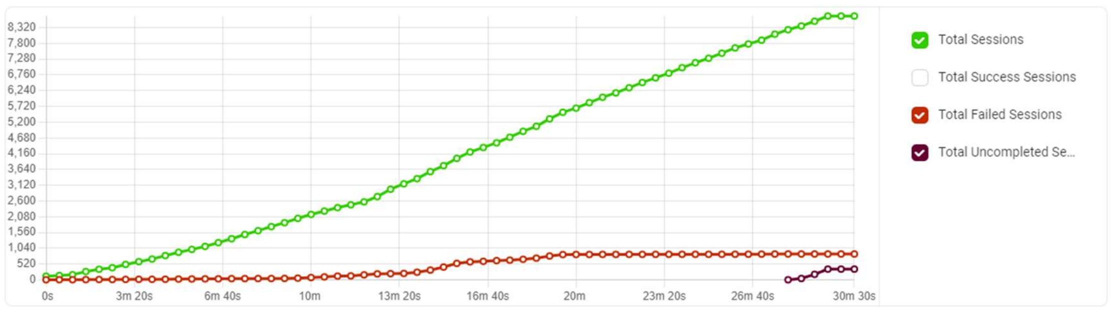
The blow graph shows the number of sessions that were started throughout the test. We can see that 27 minute in the script a few sessions were uncompleted due to the length of the test not allowing them to finish.

Sessions Started



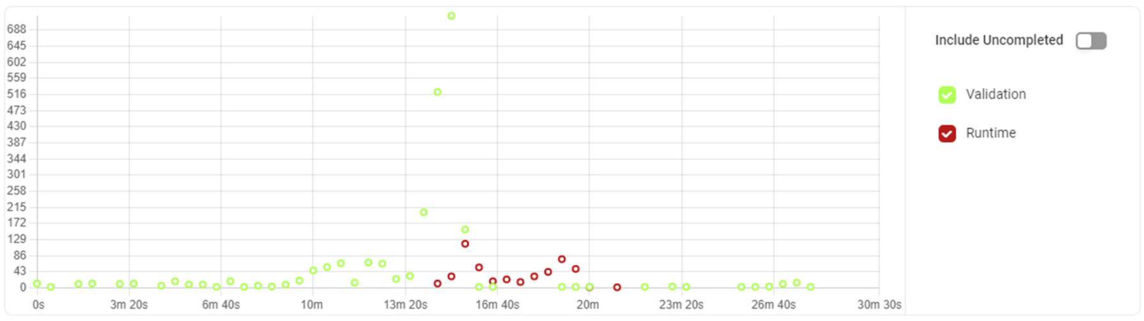
The graph below shows the number of cumulative sessions slowly ramping up throughout the test and stagnating for a few minutes at the end. 27 minutes in the test we can see those uncompleted sessions due to length of the test (unable to finish script). The errors are stable and low.

Cumulative Sessions

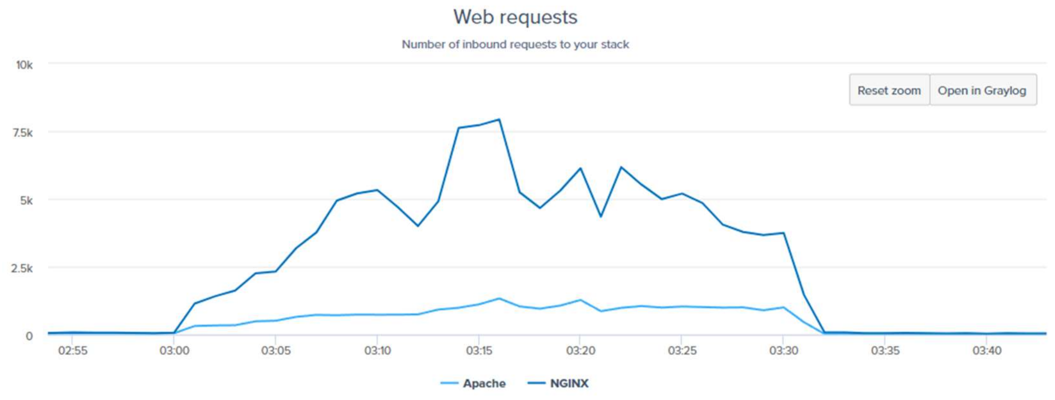


The following show the number of errors and errors by type during the test. We had mainly validation errors during the script and a few runtime errors. The runtime error indicates the issues in the script (some script variables are invalid or cannot be found).

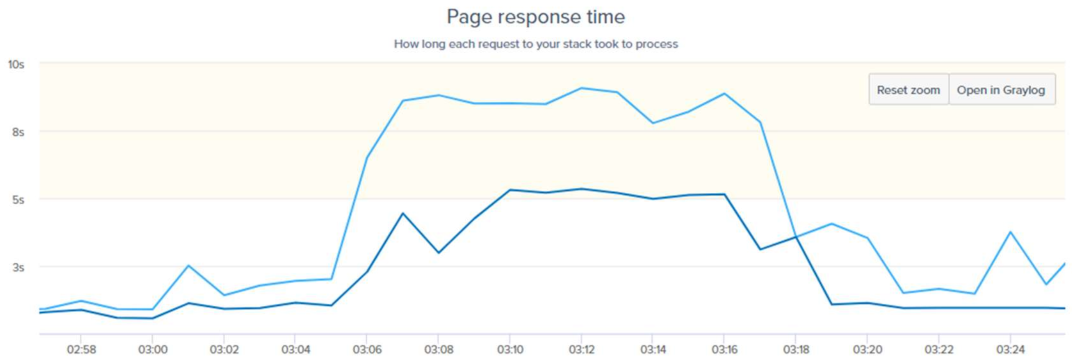
Errors by Error Type



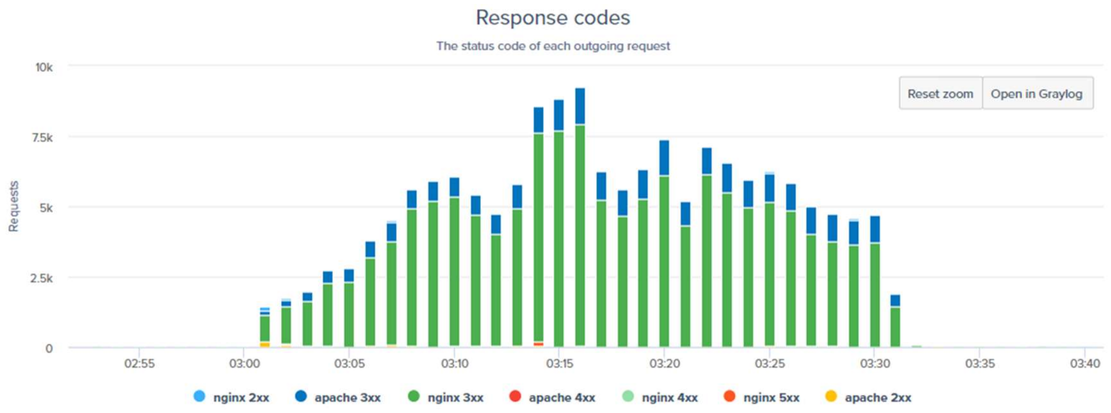
The below shows the number of web request that were done on the server of the website peaking at 7500 requests at 3:15 (middle of the test).



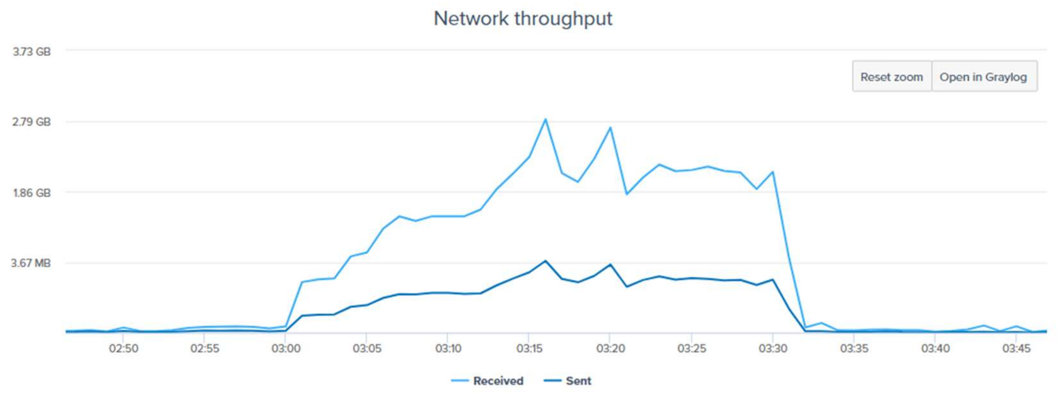
Page response time was in average 5 seconds to open a page with a maximum of 9 seconds for some users. The first test ran for 1000 concurrent users were picking at 10 seconds or more in average.



We can see the response codes are what we would expect during a load test. The response codes received are what you would expect.



The network throughput on the server sent an average of 3.67~ MB and received nearly 2.7GB.



Load Test Report

MIKE Application Load Test Report

Catalyst.Net Ltd
a Catalyst IT group company

Version 1.2

September 2023

Commercial in Confidence

expert open source solutions

Level 6, Catalyst House, 150 Willis Street, Wellington 6011
PO Box 11053, Manners Street, Wellington 6142, New Zealand
+64 4 499 2267 // enquiries@catalyst.net.nz // www.catalyst.net.nz

Table of contents

1 Document Control

1.1 Version

Version	Date	Change Reason	Author	Reviewed By
1.0	14/09/2023	Original Version	[redacted]	[redacted]
1.1	18/09/2023	Avoid repeated aparagraphs	[redacted] 9(2)(a)	[redacted]
1.2	18/09/2023	Typos and minor edits	[redacted]	[redacted]

2 Overview

The main goal of the load testing exercise was to ensure that Key All processing in the DET queue was not excessively slow. During the last general election, the new DET queue Key All processing required an emergency fix due to excessive load being put by the Key All session handling. For this election, the Key All processing has been rewritten. The load tests focused on three queues that use the Key All functionality: Data Entry Team, Special Vote Declaration & EROE5 Name Change.

3 Summary

For the Data Entry Team queue, the first load test runs showed that the Key All request took on average 30 seconds. This is obviously an unacceptable wait time. The SQL Database query involved in determining available SVD forms was subsequently analysed. A modified version of the SQL query was tested and this resulted in an average response time under heavy load of 3 seconds.

In the Special Vote Declaration queue, the Key All request took on average under 6 seconds. For general user expectations this is within a acceptable wait time. During the load test run, the queue was also accessed interactively and found to have a good response time too. The SQL Database query involved in determining available SVD forms was analysed. Various attempts to make this specific query faster did not result in any conclusive change.

The new Key All handling in the EROE5 Name Change queue worked well.

4 Test Environment

4.1 Test hosts

The secondary cluster in Porirua was temporarily reconfigured to act as an independent environment for performing load testing against. This ensured that the load test results would match actual the production environment since the Porirua and Hamilton clusters are identically configured.

A single load test runner host was used to execute the test scenarios in parallel against the load test environment.

4.2 Test scripts

The load test scripts are the same as used in previous MIKE Load Test exercises.

These scripts have the following features:

- The scenario creation script extracts random data from the test database and writes out a set of test scenarios using this data.
- The load test run script performs each test scenario in parallel using a configured number of workers on the load test runner host.
- The load test script uses the WWW::Mechanize Perl library to drive the application interface.
- The output files containing data about each request of every scenario from every worker are analysed by both a statistics script to generate overall statistics and a script to generate statistical graphs.

5 Test Scenarios

The scenarios were all designed to stress the backend DB locking functionality. The data was created to have a large number of forms from which to select just one. This results in more data having to be checked by the DB to determine a free and available form that is not already taken by another user.

5.1 DET queue

This test scenario comprised of a user with appropriate permissions navigating the following actions in MIKE:

- LOGIN PAGE
- INTERNAL APPLICATION HOME PAGE
- DET FORMS QUEUE PAGE
- KEY ALL BUTTON

Each test scenario was performed by a distinct user in order to ensure every scenario would obtain a session to key the forms in the queue.

The DET queue was populated with 100,000 forms split across 5 scanner batches.

5.2 SVD queue

This test scenario comprised of a user with appropriate permissions navigating the following actions in MIKE:

- LOGIN PAGE
- INTERNAL APPLICATION HOME PAGE
- OPEN SVD SUBMENU
- SVD CHECK QUEUE PAGE
- SELECT SCANNER BATCH
- R - RESCAN REQUIRED - *repeated 9 times, so each scenario performed a total of 10 queue locks.*

Each test scenario was performed by a distinct user in order to ensure every scenario would obtain a session to key the forms in the queue.

The SVD Check queue was populated with 100,000 forms split across 5 scanner batches.

5.3 EROE5 queue

This test scenario comprised of a user with appropriate permissions navigating the following actions in MIKE:

- LOGIN PAGE
- INTERNAL APPLICATION HOME PAGE
- EROE5 QUEUE PAGE
- KEY ALL BUTTON

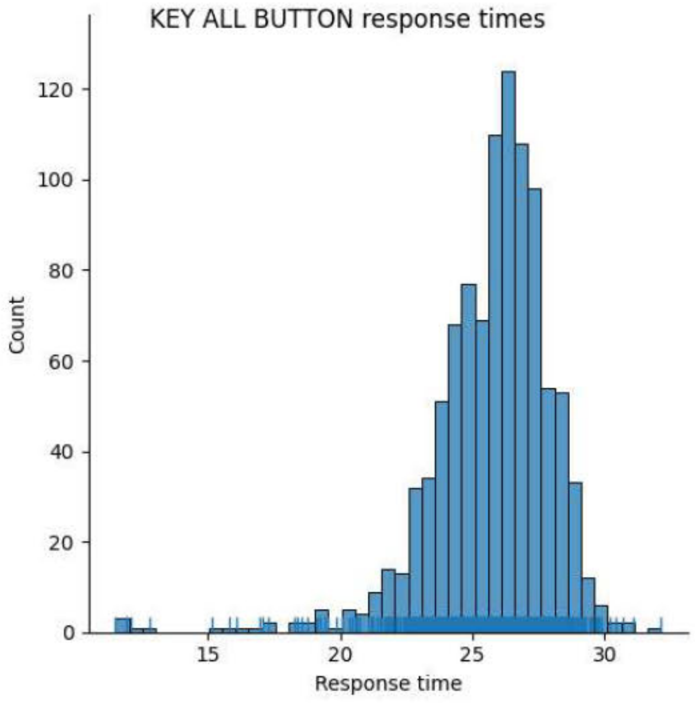
The EROE5 queue was populated with 100,000 forms.

6 Results

6.1 DET queue

Original Run

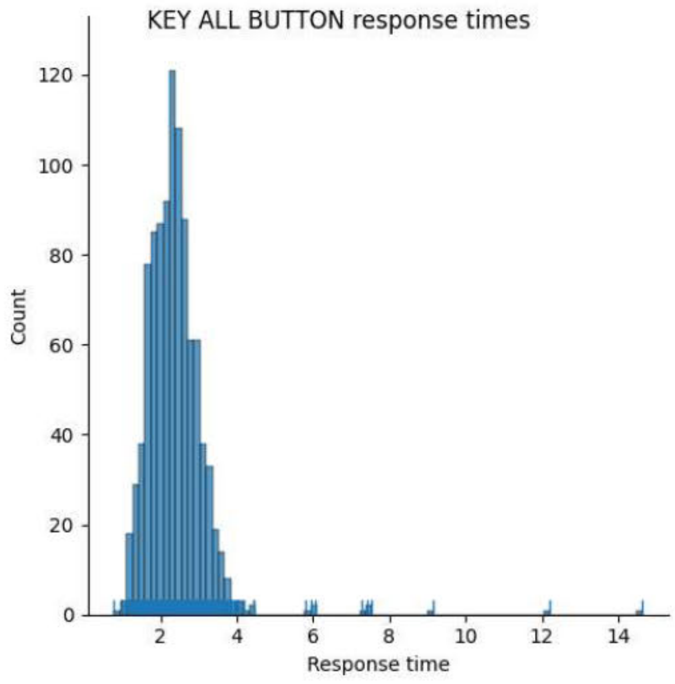
This run was with 1,000 DET users. The average time to perform the Key All button request was over 25 seconds. The following graph shows the kernel density estimate (KDE) plot of the response times for the KEY ALL button request.



The high average request time was unexpected and investigation of the query involved was performed. A modified query was trialled in the next run.

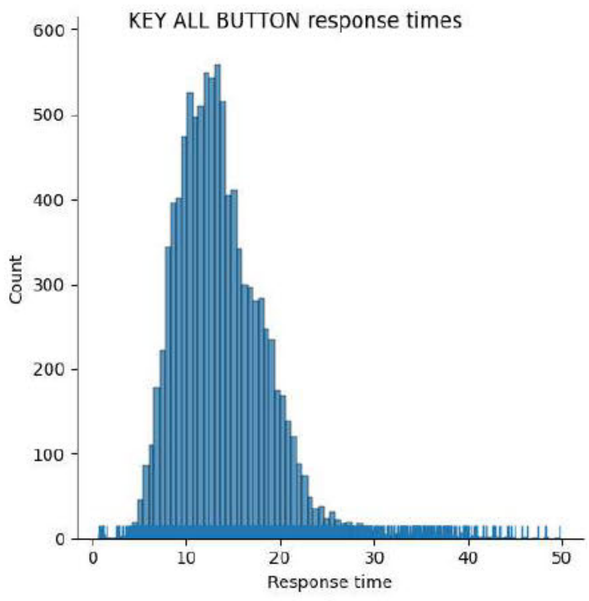
Modified Query Run

This run was the same as the original with 1,000 DET users. The average time to perform the Key All button request was reduced to an average below 2.5 seconds i.e. 10 times faster. The following graph shows the KDE plot of the response times for the KEY ALL button request.



Extreme Load Run

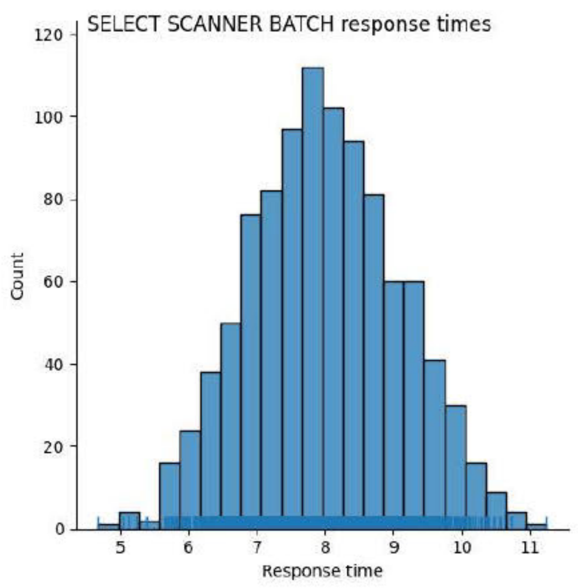
This run used 10,000 unique DET users. The average time to perform the Key All button request was below 14 seconds. The following graph shows the KDE plot of the response times for the KEY ALL button request.

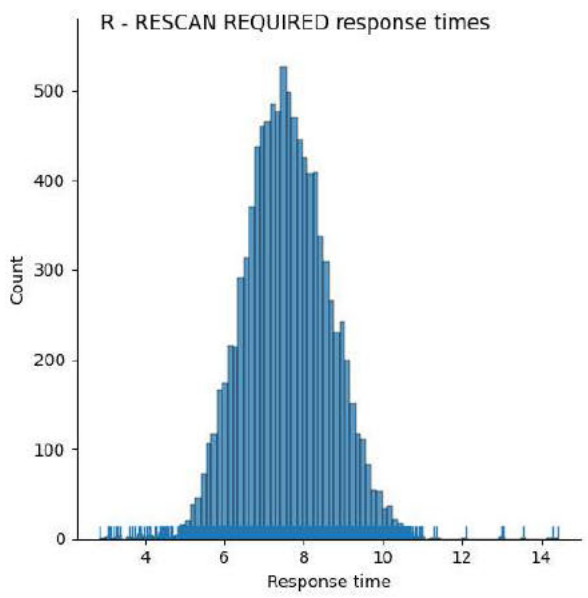


6.2 SVD queue

This run was with 1,000 SVD users each performing 10 queue lock operations, for a total of 10,000. The average time to perform the queue locking request was on average just over 7.5 seconds. The following graphs show the kernel density estimate (KDE) plot of the response times for the initial SELECT SCANNER BATCH request and the following nine R - RESCAN REQUIRED action response times.

The initial selection of the scanner batch takes a fraction longer on average 0.5 seconds but it is doing a little more work on the backend than the subsequent action request. However, they are still very similar in wait time.





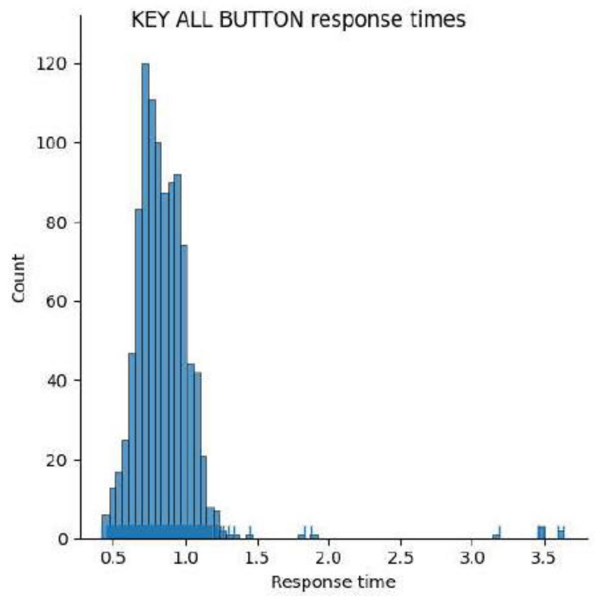
Whilst the load test was running, a manual access of the same queue was performed and the interface performed without noticeable lag in performance.

However, some investigation of the query involved was performed to see if any meaningful improvement could be found. The SVD queue handling is much more complex and several attempts were made to adjust the query and the database. But nothing was found that consistently provided any significant improvement.

6.3 EROE5 queue

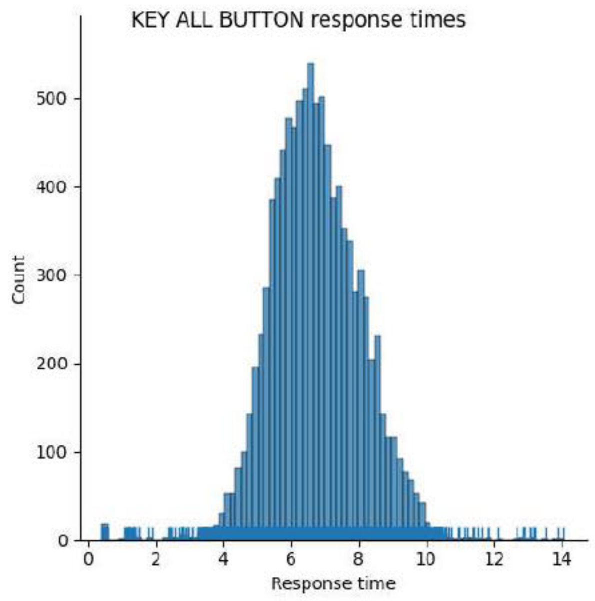
Small Run

This run was with 1,000 EROE5 users. The average time to perform the Key All button request was under 0.9 seconds. The following graph shows the kernel density estimate (KDE) plot of the response times for the KEY ALL button request.



Large Run

This run was with 10,000 EROE5 users. The average time to perform the Key All button request was under 7 seconds. The following graph shows the kernel density estimate (KDE) plot of the response times for the KEY ALL button request.



7 Recommendations

The DET Key All handling was found to require further development to reduce the unacceptable wait time for obtaining the next item in that queue. A Redmine ticket was raised to implement this change prior to the general election: 6(c), 9(2)(k) This was applied to production in the code migration on 6th September 2023,

Performance testing recommendations and results

Review of the ecvote stack for the Electoral Commission

Presented by [redacted]

11 September 2020

Silverstripe

Level 5, 101 Courtenay Place, Wellington, 6011, New Zealand
+64 4 978 7330
www.silverstripe.com

Table of Contents

Overview of results

Silverstripe has been requested by the Electoral Commission to review the performance characteristics of the https://vote.nz and https://elections.nz websites, both of which are hosted on the same Silverstripe Cloud (AWS) stack.

The performance review has been conducted from two different angles:

1. Load test of common user journeys across the websites
2. Code review of the Silverstripe CMS application to identify targets for further optimisation

The Electoral Commission has provided statistics based on previous analytics which provide the following expectations that we have based our performance testing around:

- **Peak average page views per second:** 34 - 42
- **Maximum page views per second:** 42 - 67

All of the above numbers are based on highest expectations for 'peak hours' e.g. the highest expected traffic to the websites during the busiest hour of the busiest day of the year for the websites.

Our testing methodology and detailed results are provided below for review. Additionally, we have included some recommendations for changes that we suggest could be made to the websites to further increase capacity and stability in high-load scenarios.

In conclusion, we believe that both websites are well situated for any legitimate traffic spike that we expect to see. As you will see in the detailed results below we achieved sustained traffic rates of 500 requests per second, which is well in excess of the maximum page views per second noted above (67/sec).

Recommendations

The below recommendations are suggested by Silverstripe to help increase the performance of the websites - some of these can have immediate user-facing impacts (e.g. faster webpage loading times), and others will help to increase the 'headroom' that the websites have (e.g. the capability to serve more users at the same time). These are based on our in-depth review of the performance while the websites were under load.

Each recommendation is prioritised in two different ways, both based on a High, Medium, and Low ranking scale:

1. **Importance:** Silverstripe's opinion as to the importance of making this change - balancing the expected difficulty of making the change, any risks associated with the change, and the benefit that the change may make to the websites.
2. **Impact:** Slverstripe's opinion as to the positive impact that the change may have on the websites. Note that some recommendations are not expected to help with normal day-to-day performance, but may only be noticed under certain circumstances, these are noted within each recommendation.

R01: Lower database utilisation by adding template partial caching

Importance: High
Impact: Medium

As noted during an early video conference call prior to the release of this document, we recommend that template partial caching is implemented by the development vendor. During the performance testing work on UAT, it was identified that the database utilisation was the primary issue that was causing the websites to slow down. Implementing template partial caching can help to alleviate a lot of unnecessary database queries, lowering the CPU utilisation of the database servers.

This will also help ensure that page response times are lowered when HTTP-level caching is not available (e.g. because a website visitor has visited a page not currently being cached by 6(c), 9(2)(k)

More technical information on template partial caching is available on the Silverstripe CMS developer documentation site[1]. We strongly recommend that this is enabled for the following areas:

1. Main website navigation (HeaderElectionsMainNav.ss and HeaderVote.ss)
2. Breadcrumb trails on pages (BreadcrumbsTemplate.ss), if it lowers the number of database queries required to render the page
3. Anywhere else that the <% loop %> template structure is used, especially where ORM objects are iterated over, if it lowers the number of database queries required to render the page

When developing template partial cache functionality, it can be very helpful to use the lekoala/silverstripe-debugbar module[2] to ensure that the template cache is being hit, and also to ensure that the number of database queries is lowered.

[1] https://docs.silverstripe.org/en/4/developer_guides/performance/partial_caching/
[2] https://github.com/lekoala/silverstripe-debugbar

R02: Consider caching errors and ignoring irrelevant unnecessary GET parameters

Importance: Medium
Impact: Medium

By default, 6(c), 9(2)(k) does not cache errors (such as 'Page not found' errors), meaning that all of these are passed through to Silverstripe CMS to determine whether or not they are still errors. This provides a potential [redacted] method for attackers, as Silverstripe CMS is much slower than [redacted] to determine whether a request should result in an error or not.

[redacted]

Making both of these changes require coordination between yourselves, your development partner and Silverstripe - in particular to ensure that these changes do not introduce any defects, and that any necessary GET parameters are not ignored by [redacted] (meaning that incorrect or invalid content is returned). If you are interested in applying this configuration change, please get in touch with Silverstripe to confirm potential impacts.

R03: Consider increasing the cache age for dynamic content

Importance: Medium
Impact: Medium

The websites both have a hard-coded 'Cache control' maximum age of 5 minutes for the majority of dynamic content. While this is a good introductory measure, we recommend allowing this value to be increased, especially for content that does not often change, and where the change does not need to be reflected everywhere immediately. For example, a cache age of 15 - 30 minutes would help ensure that [redacted] can provide more cached content without needing to contact the web servers, with the caveat that unless [redacted] cache is purged, old content may be provided to visitors for up to 30 minutes.

Consider allowing content authors or administrators to determine the maximum allowable cache age, or alternatively decide on a cache age based on for example page type (e.g. the homepage has a 10 minute cache age, but all other content pages have a 1 hour cache age).

R04: Investigate how YouTube videos are embedded

Importance: Medium
Impact: Low

By default, Silverstripe CMS requires API calls to the YouTube API for every YouTube video that is embedded via the WYSIWYG editor field in the CMS. This isn't necessary for the vote.nz website, and this can add multiple seconds to page rendering times while we wait for multiple API responses from the YouTube API. As discussed during an earlier video conference, we recommend removing these and replacing them with a direct integration via an HTML <iframe> tag (e.g. not using the WYSIWYG editor, but instead create a specific 'Video' content block).

We have noted that the impact of this is Low, but that is only because there are a small number of pages on the websites that include videos. If a high-traffic page (such as the homepage) includes YouTube videos, the impact on this could be significantly higher.

Detailed results

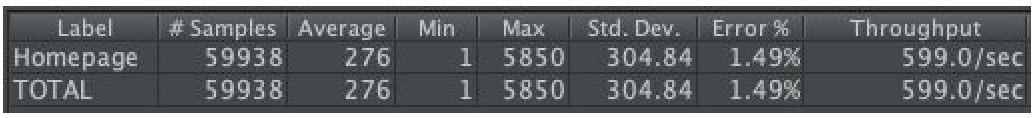
There are two major tests that we have performed. The first test ensures that the most often-requested page (the homepage) is well configured within 6(c), 9(2)(k) and the second ensures that a wider range of different types of content (e.g. images, CSS and JS files) and web pages are also well configured.

The results of this testing is sufficient to have us agree that the websites are well-cached by [redacted] and should handle any legitimate traffic sufficiently well during the election period.

We have performed additional testing prior to these tests over longer periods of time and have not noted any degradation in performance during these longer 'soak' tests, however these are not replicated below to avoid confusion with the different test methods. These soak tests did not highlight any other issues (the key confirmation with soak tests is that performance does not degrade over time, and this was not noted).

T01: Homepage load test

In this first case, we only request the homepage HTML and no other content. This ensures that [redacted] is correctly caching dynamic content that would otherwise be rendered by Silverstripe CMS. As can be seen from the results below, this is successful, with an average of 599 requests per second. At this point it is clear that [redacted] is capable of caching the homepage successfully, and this is significantly higher than any of the expected metrics as noted in the Overview section.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Homepage	59938	276	1	5850	304.84	1.49%	599.0/sec
TOTAL	59938	276	1	5850	304.84	1.49%	599.0/sec

Figure 1: Test showing the average request took just 276ms, with a throughput of 599 requests per second.

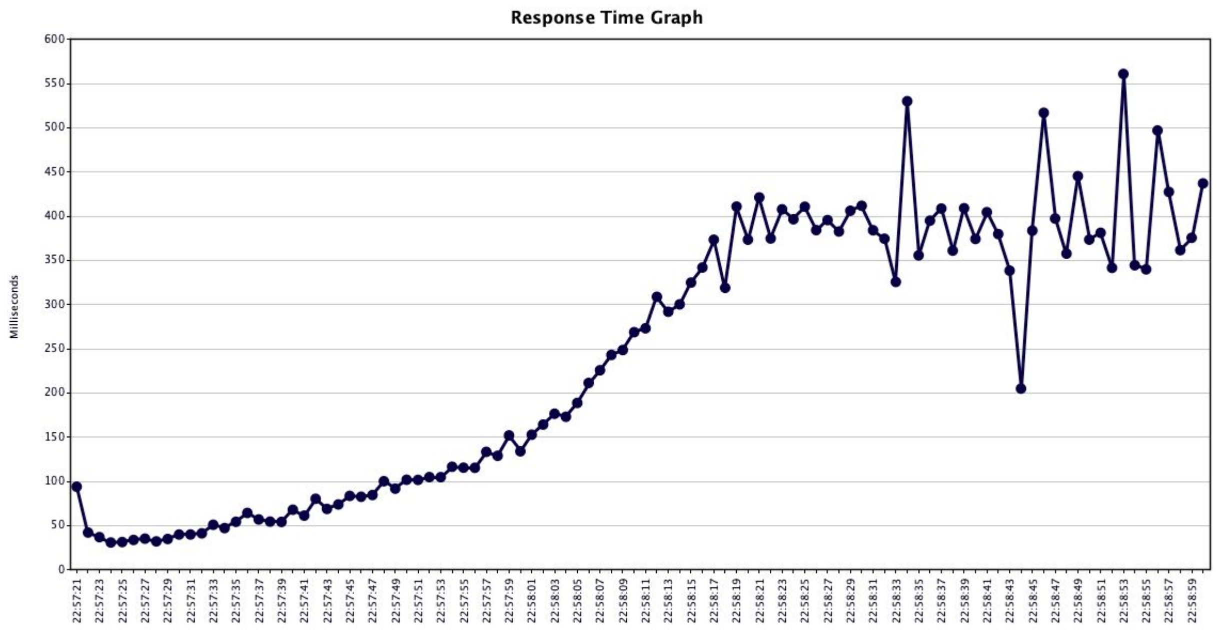


Figure 2: Showing the average response time for all requests within a given second. Note that the slow increase here indicates that the testing infrastructure was under pressure, rather than 6(c), 9(2)(k) *(in other words, the increase is not a problem that needs to be resolved).*

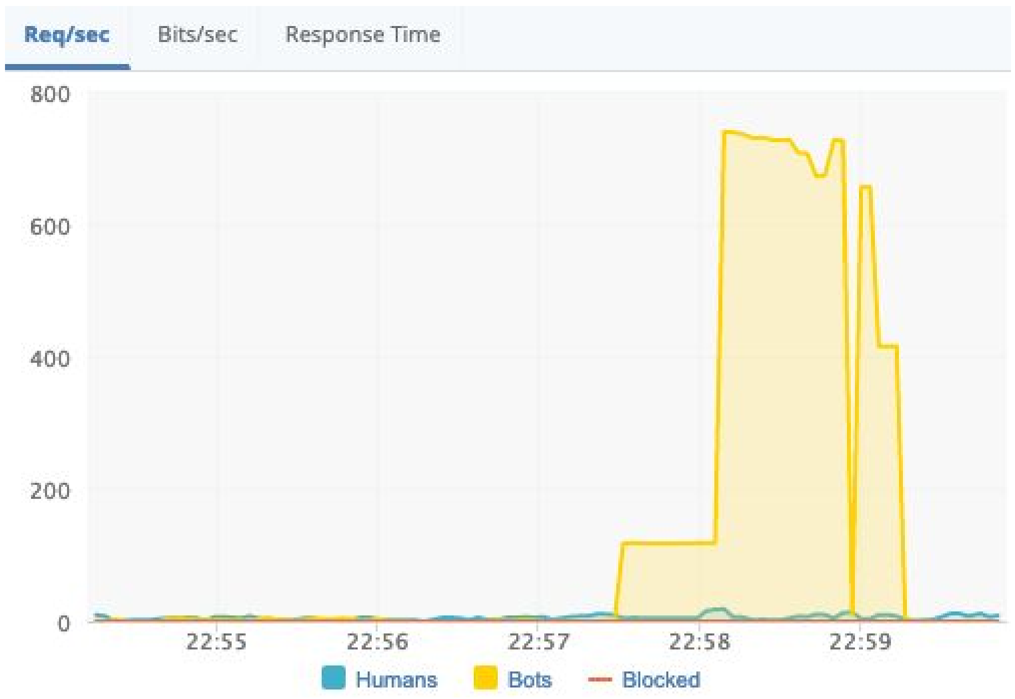


Figure 3: [redacted] *graph showing the significant increase in traffic to the websites during testing, showing requests per second as high as ~750req/sec. Typical website traffic is seen before and after the spikes, well down in the 20-30 requests per second range.*

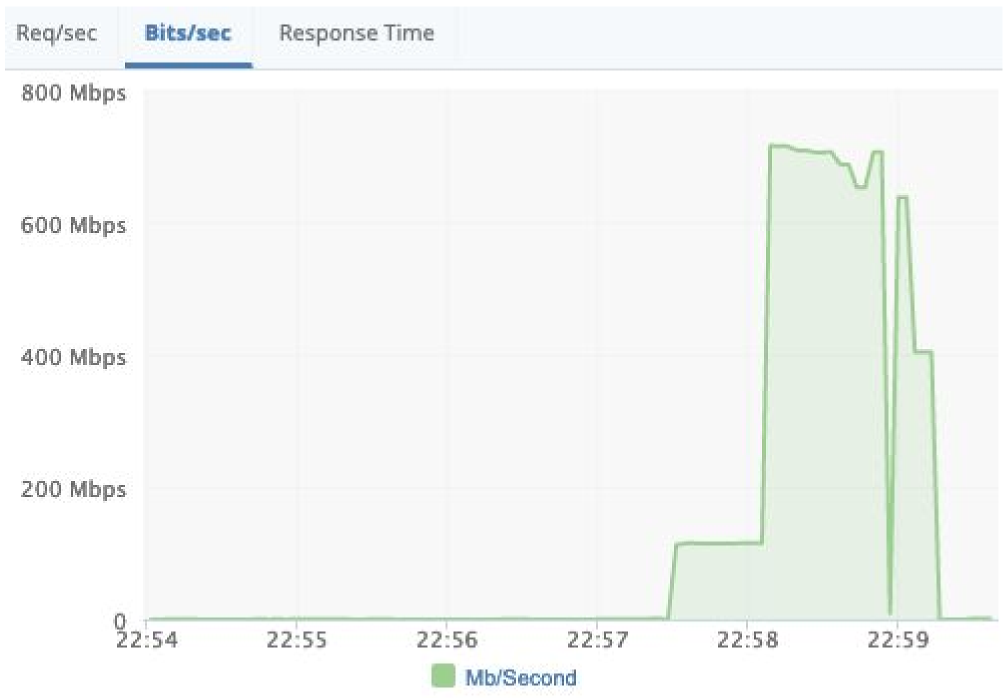


Figure 4: 6(c), 9(2)(k) *graph showing the tests impact on Mbit per second (aka throughput).*

T02: Comprehensive load test ([redacted]

The second test we performed was a more comprehensive test that included loading the homepage and all assets necessary to render the homepage (e.g. images, CSS and JS files, favicons etc) to ensure that all files are suitably cached by [redacted] We also included a small number of other website pages to ensure [redacted] is also caching those.

Unfortunately during this test our load testing infrastructure had some issues compiling results, and so we don't have the same level of detail that we have for the first test (T01 above). However, while running this test we noted a similar level of traffic within [redacted] with peaks around 600 requests per second and approximately 250Mbit/sec at peak. You may notice that this is significantly lower throughput than T01 above, however that is expected because the larger responses (HTML pages) were performed less frequently compared to the other smaller requests (images and the favicon in particular).

If the Electoral Commission prefers, we are happy to re-perform this testing to gather the graphs and statistics, however we do not believe this is necessary as we did not identify any issues with the websites during this testing.

Load Test Report

Public Applications Load Test Report

Catalyst.Net Ltd

a Catalyst IT group company

Version 1.1

September 2023

Commercial in Confidence

expert open source solutions

Level 6, Catalyst House, 150 Willis Street, Wellington 6011
PO Box 11053, Manners Street, Wellington 6142, New Zealand
+64 4 499 2267 // enquiries@catalyst.net.nz // www.catalyst.net.nz

Table of contents

1 Document Control

1.1 Version

Version	Date	Change Reason	Author	Reviewed By
1.0	14/09/2023	Original Version	9(2)(a) [redacted]	[redacted]
1.1	18/09/2023	Typos and minor edits	[redacted]	[redacted]

2 Overview

The main goal of the load testing exercise was to provide a baseline of performance of functionality enhancements for the Public applications. A secondary objective was to find any quick wins that would provide increased throughput or response times.

The Download Voting Papers application has been recently changed to perform additional checks when the Remote Grounds option is chosen by an elector. This new scenario was incorporated in the DVP load tests.

The eRoll application has been recently extended to allow for limited updates to be sent back to MIKE for processing. An additional address search feature has also been added. These new scenarios were incorporated in the eRoll load tests.

The Enrol OnLine application has had many enhancements made to existing functionality, including the recent legislation change to the Māori Electoral Option. No updates to the load test scenarios were needed as the existing scenarios proved to be sufficient.

3 Summary

For the baseline load test run, the overall time taken to process all the scenarios was 29 minutes. A number of simple parameter changes were made to the NGINX, Apache and Starman webserver configurations. The same set of scenarios were then run again and the overall time taken to process them. It was found that a single parameter change each to NGINX, Apache and a Starman webserver reduced the overall time to 21 minutes. This provided an increase in throughput of over 25%.

4 Test Environment

4.1 Test hosts

The secondary cluster in Porirua was temporarily reconfigured to act as an independent environment for performing load testing against. This ensured that the load test results would match actual the production environment since the Porirua and Hamilton clusters are identically configured.

Three load test runner hosts were used to execute the test scenarios in parallel against the load test environment.

4.2 Test scripts

These are the same as used in previous Public Application Load Test exercises. Although the eRoll and DVP load test scripts were enhanced to incorporate new scenarios:

- DVP scripts had a new remote grounds scenario added
- eRoll scripts had a new Address search API call added
- eRoll scripts had a person update scenario added

All these load test scripts have the following features:

- The scenario creation script extracts random data from the test database and writes out a set of test scenarios using this data.
- The load test run script performs each test scenario in parallel using a configured number of workers on the load test runner host.
- The load test script uses the WWW::Mechanize Perl library to drive the application interface.
- The output files containing data about each request of every scenario from every worker are analysed by both a statistics script to generate overall statistics and a script to generate statistical graphs.

4.3 Host Statistics

Each host has data collected from it to read various aspects of the state of the host. This data is visualised through a tool called Totempole. These graphs show the following states: CPU %, Network Requests, Load, Memory, Processes.

Note that this data collection is performed as standard on all Production and UAT hosts as part of standard monitoring by Catalyst.

5 Test Scenarios

5.1 Enrol OnLine application

The test scenarios for Enrol OnLine comprise of a user navigating through the web pages using one of six different paths:

- New Enrolment with Passport assertion
- New Enrolment with Licence assertion
- New Enrolment without ID check
- Update Enrolment with Passport assertion
- Update Enrolment with Licence assertion
- Update Enrolment without ID check

Each test path includes calling the analytics API several times and multiple address completion API calls as appropriate.

The Load test runs consist of 1,000 scenarios of each type above for each runner host.

The Passport and Licence assertions are mocked in the Load test environment to return success directly.

5.2 Download Voting Papers application

The test scenarios for DVP comprise of a user navigating through the web pages following one of three different situations:

- NZ elector with no overseas address, but not downloading voting papers
- Overseas NZ elector, with outside NZ reason & downloading voting papers
- Overseas NZ elector, with approved remote grounds reason & downloading voting papers

The Load test runs consist of 1,000 scenarios of each of the above types for each runner host. Each scenario includes multiple address completion API calls as appropriate.

The number of scenarios that request a download (second and third above) is set to be 53.4%, of which the third scenario accounts for 5%.

5.3 eRoll application

The test scenarios for Enrol OnLine comprise of a user performing a selection of the following application API calls:

- Get the current set of Electorates with an election
- Perform a Lookup Address operation
- Perform a Person Search (variations include using wildcards, no midname)
- Perform some Address Completion calls
- Submit an Update to generate an MDA2 correspondence
- Perform a user Logout

The percentage of each of the above API calls between July 2022 and August 2023 was obtained from the database.

API route	%
electorates	11
enrol	2
persons	40
residential_addresses	43
lookup_address	1
logout	3
Total	*100*

The Load test runs consist of 1,000 scenarios for each runner host, with the overall number of API calls for each route matching the above percentage.

5.4 Methodology

Load Test Scenarios

A load test run is in two parts. The first part prepares a mix of all 6,000 EOL scenarios and 1,000 DVP scenarios for executing against the pubweb server. The second part is the 1,000 eRoll scenarios prepared for executing against the eRoll web server. These are created for all three load test runner hosts. Overall there are 24,000 scenarios executed in each full test run.

Load Test Execution

On each runner host, the 7000 (EOL & DVP) scenarios were started running against the pubweb server. Then, after a few minutes, on each runner host, the 1,000 eRoll scenarios were started running against the eroll web server.

6 Results

The following sections presents details from two Load test runs. The first being the baseline run using existing production configuration settings. The second run evaluates a set of configuration changes made to the public web and application services.

A number of other configuration runs were performed. The data from those are not presented here, but they helped with the choice of configuration changes presented in the second load test run.

6.1 Baseline Run

The baseline run used the existing NGINX, Apache and Starman configurations used in production.

The total overall time to execute all 24,000 scenarios, using the three load test runners was 29.5 minutes.

Request Statistics

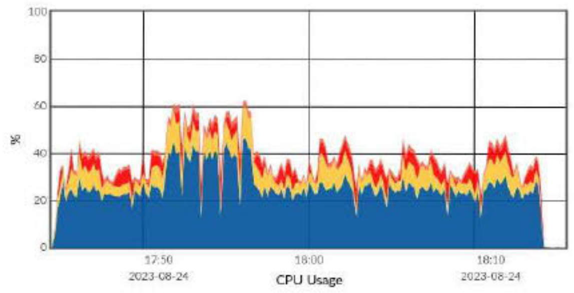
All responses occur under 10 seconds and generally under 1s for EOL & DVP , and under 3s for eRoll.

Step	requests	min	median	mean	95th	max	>10s	Error
DVP - Address Completion	16,033	0.03	0.45	0.50	0.55	7.75	0	0
DVP - Confirm Details	1,601	0.13	0.55	0.59	0.67	8.00	0	0
DVP - Download PDF	1,601	0.32	0.84	0.88	1.14	4.77	0	221†
DVP - Enrolment Details	3,000	0.20	0.62	0.68	0.82	8.74	0	1†
DVP - Initial Page	3,000	0.13	0.57	0.61	0.71	7.70	0	0
DVP - Reason - Outside Selection	2,922	0.11	0.56	0.59	0.69	6.76	0	0
DVP - Reason - Remote	78	0.40	0.56	0.56	0.64	0.69	0	0
EOL - Address completion	193,396	0.02	0.45	0.50	0.56	7.83	0	0
EOL - Analytics	170,934	0.05	0.49	0.51	0.60	8.08	0	0
EOL - Email validation	17,994	0.03	0.45	0.45	0.54	7.54	0	0
EOL - Enrolment form	17,986	0.21	0.85	0.91	1.23	8.63	0	0
EOL - ES validation	17,994	0.03	0.45	0.47	0.53	7.84	0	0
EOL - Initial page	18,000	0.13	0.57	0.61	0.71	8.18	0	0
EOL - Passport initiate	5,999	0.13	0.55	0.60	0.70	8.01	0	0
EOL - Passport resolve	5,999	0.13	0.56	0.67	0.81	8.08	0	0
EOL - Pre submit	17,994	0.11	0.56	0.59	0.70	8.02	0	8†
EOL - Search	18,000	0.14	0.59	0.64	0.77	8.07	0	6†
EOL - Verify licence	5,997	0.14	0.59	0.66	0.90	8.08	0	0
eRoll - Address completion	3,227	0.18	1.98	2.20	2.86	9.31	0	0
eRoll - Electorates	2,837	0.21	1.98	2.11	2.52	9.27	0	0
eRoll - Enrol Update MDA2	623	0.70	2.71	2.89	3.75	9.96	0	0
eRoll - Logout	947	0.21	1.97	2.08	2.50	9.22	0	0
eRoll - Lookup Address	279	0.21	1.96	2.17	2.76	9.25	0	0
eRoll - Search	7,561	0.23	2.02	2.19	2.61	9.43	0	0
eRoll - Search - Mononym	277	0.26	2.04	2.24	2.63	9.21	0	0
eRoll - Search - No Midnames	7,486	0.24	2.01	2.17	2.58	9.29	0	0
eRoll - Search - Wild	7,354	0.23	2.03	2.18	2.60	9.33	0	0
eRoll - Search - Wild - No Midnames	7,322	0.25	2.02	2.17	2.58	9.35	0	0

† These errors appeared consistently across the load test runs - and are dependent upon the test data

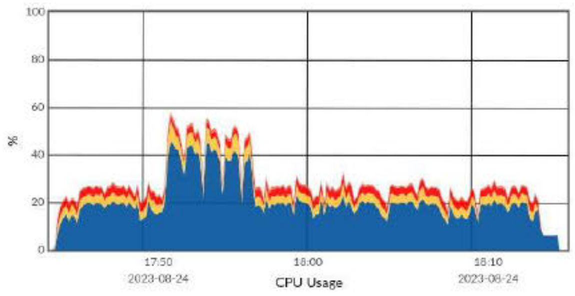
CPU utilisation

The following graphs show the CPU utilisation on the pubweb, pubapp and db servers. The extra processing load of eRoll can be easily seen in these server graphs.



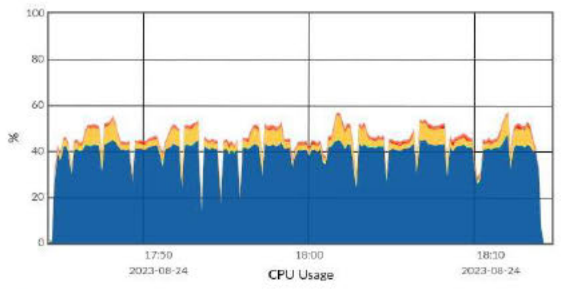
```
🟦 User     : Min: 0.00 Max: 46.72 Current: 0.00
🟩 Nice     : Min: 0.00 Max: 0.61 Current: 0.00
🟨 System   : Min: 0.00 Max: 13.26 Current: 0.00
🟥 IOWait   : Min: 0.00 Max: 12.24 Current: 0.00
🟫 Hard IRQ: Min: 0.00 Max: 0.00 Current: 0.00
🟧 Soft IRQ: Min: 0.00 Max: 2.79 Current: 0.00
🟫 Steal    : Min: 0.00 Max: 0.49 Current: 0.00
```

Database



```
🟦 User     : Min: 0.00 Max: 45.96 Current: 0.00
🟩 Nice     : Min: 0.00 Max: 0.00 Current: 0.00
🟨 System   : Min: 0.00 Max: 8.57 Current: 0.00
🟥 IOWait   : Min: 0.00 Max: 5.76 Current: 0.00
🟫 Hard IRQ: Min: 0.00 Max: 0.00 Current: 0.00
🟧 Soft IRQ: Min: 0.00 Max: 1.31 Current: 0.00
🟫 Steal    : Min: 0.00 Max: 0.15 Current: 0.00
```

Public Application Server

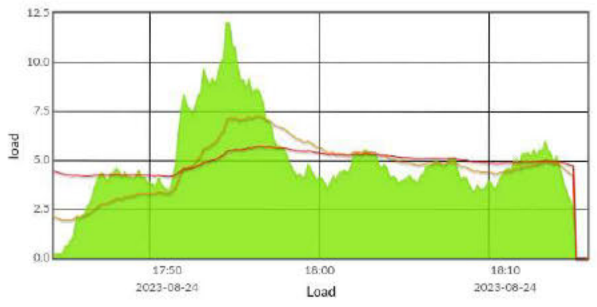


```
🟦 User     : Min: 0.00 Max: 46.90 Current: 0.00
🟩 Nice     : Min: 0.00 Max: 0.56 Current: 0.00
🟨 System   : Min: 0.00 Max: 10.67 Current: 0.00
🟥 IOWait   : Min: 0.00 Max: 1.66 Current: 0.00
🟫 Hard IRQ: Min: 0.00 Max: 0.00 Current: 0.00
🟧 Soft IRQ: Min: 0.00 Max: 1.22 Current: 0.00
🟫 Steal    : Min: 0.00 Max: 0.46 Current: 0.00
```

Public Webserver

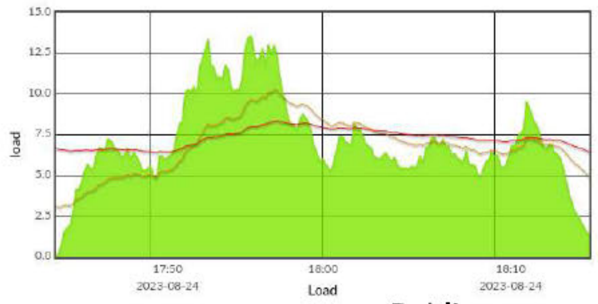
Host Load

The following graphs show the load on the pubweb, pubapp and db servers. The load is well within the capacity of the hosts.



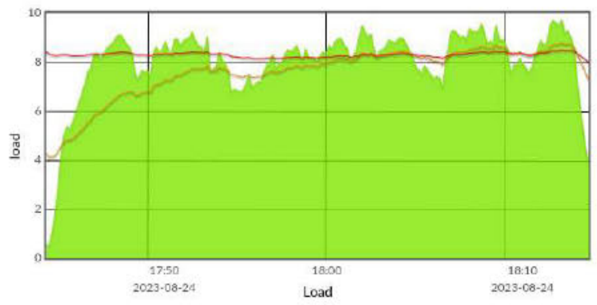
```
🟩 1 minute : Min: 0.00 Max: 12.03 Current: 0.00
🟧 5 minute : Min: 0.00 Max: 7.27 Current: 0.00
🟥 15 minute: Min: 0.00 Max: 5.72 Current: 0.00
```

Database



```
🟩 1 minute : Min: 0.17 Max: 13.57 Current: 1.35
🟧 5 minute : Min: 3.05 Max: 10.31 Current: 4.94
🟥 15 minute: Min: 6.40 Max: 8.37 Current: 6.45
```

Public Application Server



```
🟩 1 minute : Min: 0.49 Max: 9.72 Current: 3.81
🟧 5 minute : Min: 4.14 Max: 8.80 Current: 7.33
🟥 15 minute: Min: 8.02 Max: 8.50 Current: 8.02
```

Public Webserver

6.2 Reconfiguration run

This run included the following configuration changes:

- doubled the number of NGINX worker processes from 4 to 8 on the public web server
- Increased Apache MaxRequestWorkers (MaxClients) from 30 to 40 on the public application server
- Increased Starman public backend service workers from 30 to 40 on the application server

The total overall time to execute all 24,000 scenarios, using the three load test runners was 21.5 minutes.

Request Statistics

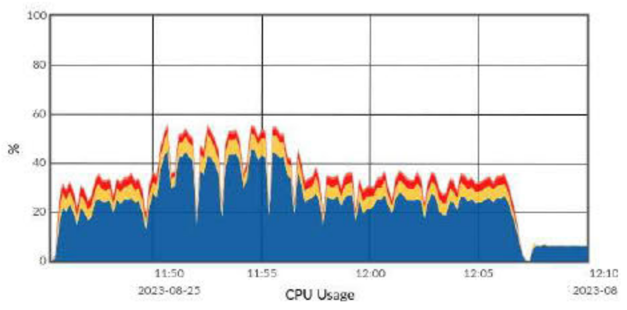
Almost all responses occur under 10s and still under 1s for EOL & DVP , and generally under 4s for eRoll.

Step	requests	min	median	mean	95th	max	>10s	Error
DVP - Address Completion	16,033	0.02	0.13	0.19	0.27	6.53	0	0
DVP - Confirm Details	1,601	0.11	0.32	0.38	0.56	6.60	0	0
DVP - Download PDF	1,601	0.31	0.69	0.73	1.06	7.24	0	221[†]
DVP - Enrolment Details	3,000	0.17	0.40	0.46	0.67	6.98	0	1[†]
DVP - Initial Page	3,000	0.12	0.34	0.39	0.59	6.96	0	0
DVP - Reason - Outside Selection	2,922	0.13	0.34	0.39	0.57	6.96	0	0
DVP - Reason - Remote	78	0.18	0.34	0.35	0.55	0.78	0	0
EOL - Address completion	193,392	0.02	0.13	0.20	0.28	6.68	0	1
EOL - Analytics	170,931	0.05	0.23	0.27	0.42	6.94	0	0
EOL - Email validation	17,994	0.02	0.11	0.13	0.23	6.61	0	0
EOL - Enrolment form	17,985	0.21	0.68	0.75	1.14	8.57	0	0
EOL - ES validation	17,994	0.03	0.17	0.20	0.33	6.76	0	0
EOL - Initial page	18,000	0.12	0.34	0.38	0.57	7.01	0	0
EOL - Passport initiate	5,999	0.12	0.32	0.36	0.54	6.96	0	0
EOL - Passport resolve	5,999	0.12	0.33	0.46	0.67	7.04	0	0
EOL - Pre submit	17,993	0.12	0.32	0.37	0.55	7.00	0	8[†]
EOL - Search	18,000	0.12	0.37	0.43	0.64	7.07	0	6[†]
EOL - Verify licence	5,997	0.14	0.38	0.44	0.71	7.02	0	0
eRoll - Address completion	3,276	0.20	2.52	2.85	8.30	9.35	0	0
eRoll - Electorates	2,967	0.25	2.50	2.74	3.62	9.33	0	30
eRoll - Enrol Update MDA2	631	0.80	3.35	3.84	9.53	10.23	3	0
eRoll - Logout	621	0.22	2.49	2.72	3.65	9.25	0	0
eRoll - Lookup Address	327	0.24	2.51	2.71	3.76	9.26	0	4
eRoll - Search	7,473	0.26	2.53	2.81	7.36	9.37	0	82
eRoll - Search - Mononym	249	0.32	2.56	2.84	6.64	9.44	0	3
eRoll - Search - No Midnames	7,648	0.27	2.52	2.76	3.55	9.40	0	65
eRoll - Search - Wild	7,313	0.28	2.55	2.80	3.67	9.45	0	62
eRoll - Search - Wild - No Midnames	7,283	0.27	2.55	2.77	3.41	9.48	0	76

[†] These errors appeared consistently across the load test runs - and are dependent upon the test data

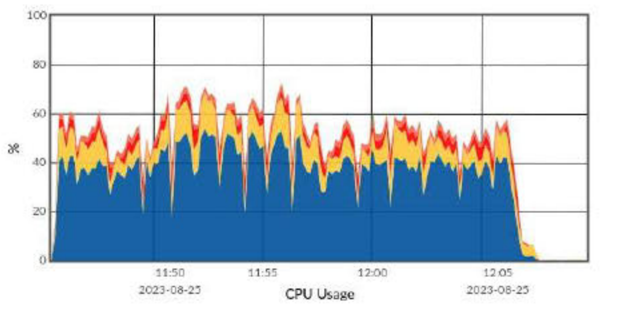
CPU utilisation

The following graphs show the CPU utilisation on the pubweb, pubapp and db servers. The extra processing load of eRoll can be easily seen in these server graphs.



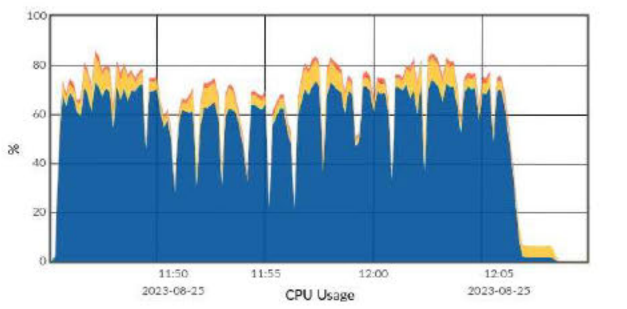
```
■ User    : Min: 0.04 Max: 45.81 Current: 6.25
■ Nice    : Min: 0.00 Max: 0.00 Current: 0.00
■ System  : Min: 0.07 Max: 9.00 Current: 0.13
■ IOWait  : Min: 0.01 Max: 3.88 Current: 0.01
■ Hard IRQ: Min: 0.00 Max: 0.00 Current: 0.00
■ Soft IRQ: Min: 0.00 Max: 1.17 Current: 0.00
■ Steal   : Min: 0.00 Max: 0.28 Current: 0.00
```

Database



```
■ User    : Min: 0.00 Max: 53.99 Current: 0.01
■ Nice    : Min: 0.00 Max: 0.69 Current: 0.18
■ System  : Min: 0.01 Max: 14.35 Current: 0.12
■ IOWait  : Min: 0.00 Max: 9.50 Current: 0.00
■ Hard IRQ: Min: 0.00 Max: 0.00 Current: 0.00
■ Soft IRQ: Min: 0.00 Max: 2.61 Current: 0.00
■ Steal   : Min: 0.00 Max: 3.04 Current: 0.00
```

Public Application Server

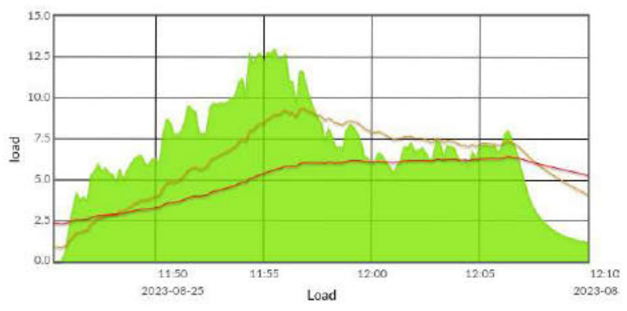


```
■ User    : Min: 0.01 Max: 74.24 Current: 0.03
■ Nice    : Min: 0.00 Max: 0.54 Current: 0.26
■ System  : Min: 0.02 Max: 11.23 Current: 0.29
■ IOWait  : Min: 0.00 Max: 1.04 Current: 0.02
■ Hard IRQ: Min: 0.00 Max: 0.00 Current: 0.00
■ Soft IRQ: Min: 0.00 Max: 1.55 Current: 0.00
■ Steal   : Min: 0.00 Max: 0.17 Current: 0.00
```

Public Webserver

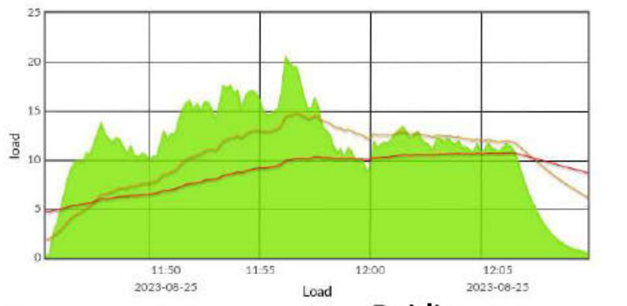
Host Load

The following graphs show the load on the pubweb, pubapp and db servers. The load is also well within the capacity of the hosts.



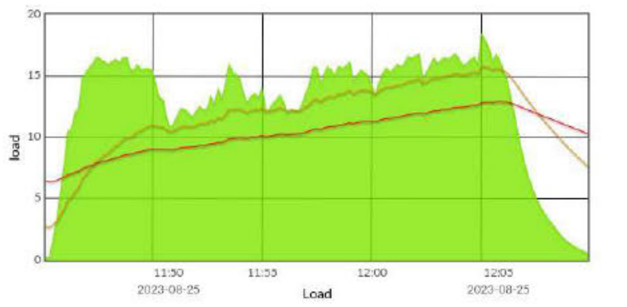
```
■ 1 minute : Min: 0.00 Max: 13.03 Current: 1.18
■ 5 minute : Min: 0.05 Max: 9.41 Current: 4.11
■ 15 minute: Min: 2.30 Max: 6.42 Current: 5.30
```

Database



```
■ 1 minute : Min: 0.21 Max: 20.60 Current: 0.52
■ 5 minute : Min: 1.89 Max: 14.85 Current: 6.24
■ 15 minute: Min: 4.70 Max: 10.74 Current: 8.72
```

Public Application Server



```
■ 1 minute : Min: 0.21 Max: 18.40 Current: 0.59
■ 5 minute : Min: 2.65 Max: 15.79 Current: 7.67
■ 15 minute: Min: 6.39 Max: 12.90 Current: 10.31
```

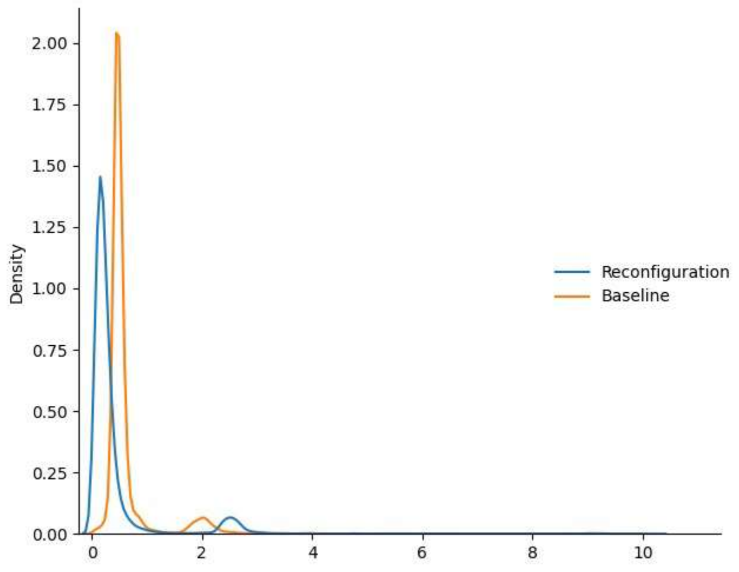
Public Webserver

6.3 Comparison

The kernel density estimate graph shows time distribution of all the AJAX response times. The following graphs show the results from the two load test runs.

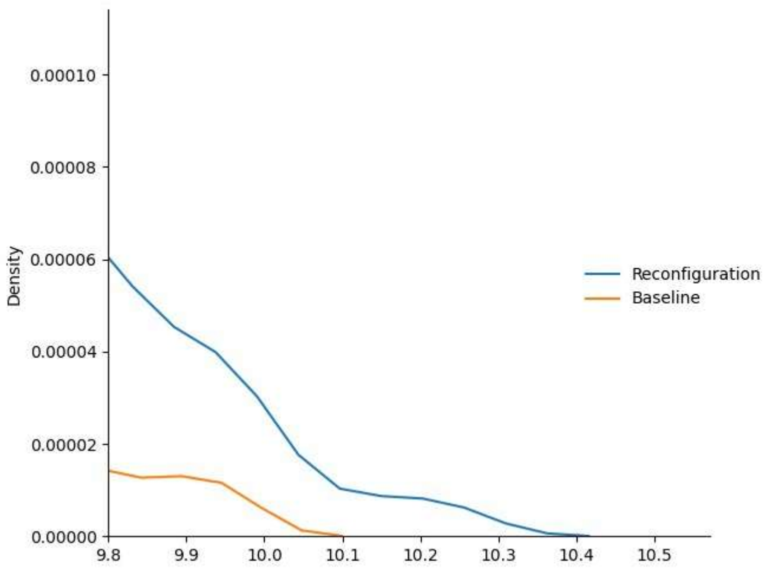
The large peak shows the majority of the EOL related requests and the small peak shows the majority of eRoll related requests.

The Reconfiguration run shows the reduction in response times for the EOL related requests with a marginal increase for the eRoll released requests. This shows why the reconfiguration reduces the overall time for all the requests from 29.5 minutes to 21.5 minutes. This is an over 25% improvement in overall response times.



The tail of the density graph (magnified) shows the times of the very small number of longer running requests.

The Reconfiguration run shows just a marginal increase in maximum request time from 10.1 to 10.4 seconds.



7 Recommendations

An over 25% reduction in overall time has been shown to be achieved with a few simple configuration changes on the Public Web and Public Application servers. A Redmine ticket was raised to implement these changes prior to the general election: 6(c), 9(2)(k) These were applied to production in the code migration on 6th September 2023, [redacted]

A quick investigation into some errors in the eRoll NGINX logs from the Reconfiguration were due to SSL handshaking between the eRoll webserver and the backend pubapp webserver. There are some NGINX options that may help reduce the SSL processing overhead incurred and minimise these types of errors. A Redmine story has been raised to investigate these options: [redacted]