2020

A picture containing object

Description automatically generated

FOLIO performance testing report

# LOAD MODEL

The Workload Distribution percentages presented in workload profile that is based on multiple endpoints.

A close up of text on a white background

Description automatically generated

## Load Model Coverage

|  |  |  |
| --- | --- | --- |
| **ID** | **Endpoint** | **Request** |
| 1 | /authn/login | POST\_bl-users/login |
| 2 | /configurations/entries?${query} | OPTIONS\_configurations/entries GET\_configurations/entries |
| 3 | /groups?${query} | OPTINOS\_groups  GET\_groups |
| 4 | /users?${query} | OPTIONS\_users  GET\_users |
| 5 | /circulation/loans?${query} | OPTIONS\_circulation/loans  GET\_circulation/loans |
| 6 | /accounts?${query} | OPTIONS\_accounts  GET\_accounts |
| 7 | /circulation/requests?${query} | OPTIONS\_circulation/requests  GET\_circulation/requests |
| 8 | /proxiesfor?${query} | OPTIONS\_proxiesfor  GET\_proxiesfor |
| 9 | /item-storage/items?${query} | OPTIONS\_item-storage/items  GET\_item-storage/items |
| 10 | /circulation/check-out-by-barcode | OPTIONS\_check-out-by-barcode  POST\_circulation/check-out-by-barcode |
| 11 | /loan-policy-storage/loan-policies?${query} | OPTIONS\_loan-policy-storage/loan-policies  GET\_loan-policy-storage/loan-policies |
| 12 | /service-points?${query} | OPTIONS\_service-points  GET\_service-points |
| 13 | /staff-slips-storage/staff-slips | OPTIONS\_staff-slips-storage/staff-slips  GET\_staff-slips-storage/staff-slips |
| 14 | /inventory/items?${query} | OPTIONS\_inventory/items  GET\_inventory/items |
| 15 | /circulation/check-in-by-barcode | OPTIONS\_circulation/check-in-by-barcode  POST\_circulation/check-in-by-barcode |

Acceptable response time with 3.5 requests per second:

|  |  |  |  |
| --- | --- | --- | --- |
| # | Dimension | Action | Requirements |
| **1** | Backend response time | get/post/options request | up to 100 ms |

1, 5, 8 and 20 vUsers load was chosen.

Based on the [capacity](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578937352708&to=1578938329179&var-percentile=99&var-test_type=capacity&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All) performance test results we can say that saturation point is around **10** parallel vUsers, which is around **44 req/sec** according to defined workload distribution.

A screenshot of a computer

Description automatically generated

# PERFORMANCE TESTING SUMMARY

The overall execution was successful according to the defined requirements. Most of the requests were performant enough. For the audit of the final results, we use **95th pct**. Table and chart below preset average responses for every percentile:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| vUsers Load | Total Test Time | Errors% | Total Requests | Median, sec | 75th pct, sec | 90th pct, sec | 95th pct, sec | 99th pct, sec | Max,  sec |
| [1 User](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578788060826&to=1578791971109&var-percentile=99&var-test_type=baseline&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All) | 1h | 0% | 18803 | 0.007 | 0.03 | 0.11 | 0.25 | 0.40 | 1.40 |
| [5 Users](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&var-percentile=95&var-test_type=baseline&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All&from=1578658869601&to=1578662605006) | 1h | 0% | 89120 | 0.01 | 0.04 | 0.13 | 0.34 | 0.55 | 3.76 |
| [8 Users](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578593709481&to=1578597514871&var-percentile=95&var-test_type=fix_load&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All) | 1h | 0% | 131097 | 0.01 | 0.05 | 0.17 | 0.43 | 0.78 | 3.23 |
| [20 Users](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578593709481&to=1578597514871&var-percentile=95&var-test_type=fix_load&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All) | 1h | 0% | 146487 | 0.04 | 0.20 | 0.84 | 2.35 | 3.75 | 5.65 |

# RESOURCES CONSUMPTION

Folio build was deployed with 30 services installed randomly across 3 **m5.large** instances in the cap1-pvt cluster and the database was created on the **db.r5.xlarge** AWS RDS instance. Logging level was set to default **INFO**.

According to the capacity performance test results, we can say that the saturation point was caused by high CPU utilization on one of three nodes in the cap1-pvt cluster.

A screen shot of a computer

Description automatically generated

Based on the CPU usage per service we can make a conclusion that the most consuming service was **okapi**

A screenshot of a cell phone

Description automatically generated

Memory usage was normal

A close up of a screen

Description automatically generated

Disk usage disk was normal

A screen shot of a computer

Description automatically generated

The database server was not loaded more than 5% with the default sample dataset in the peak which is normal usage.

A screenshot of a social media post

Description automatically generated

Derived results could have a slight infelicity depends on the cluster configuration, as build scripts deploy services randomly okapi could be installed on the machine with several resource-demanding services which would have an impact on the okapi performance and vice versa, okapi could be installed on the machine with lightweight services which will allow okapi use more resources. As a result, it could show a slightly higher or lower saturation point, however, actual bottlenecks will remain the same.

# RESPONSE TIMES PER EACH REQUEST

## GET\_circulation/loans

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.218 | 0.29 | 0.40 |
| 5 Users | 0% | 1.077 | 0.51 | 3.76 |
| 8 Users | 0% | 1.550 | 0.74 | 1.26 |
| 20 Users | 0% | 1.715 | 3.03 | 4.05 |

## GET\_circulation/requests

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.415 | 0.06 | 0.19 |
| 5 Users | 0% | 1.984 | 0.09 | 1.05 |
| 8 Users | 0% | 2.9 | 0.12 | 0.3 |
| 20 Users | 0% | 3.239 | 0.52 | 1.31 |

## GET\_configurations/entries

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.219 | 0.03 | 0.07 |
| 5 Users | 0% | 1.078 | 0.04 | 0.08 |
| 8 Users | 0% | 1.552 | 0.06 | 0.25 |
| 20 Users | 0% | 1.721 | 0.25 | 0.86 |

## GET\_groups

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.306 | 0.03 | 0.23 |
| 5 Users | 0% | 1.448 | 0.04 | 0.13 |
| 8 Users | 0% | 2.133 | 0.06 | 3.07 |
| 20 Users | 0% | 2.385 | 0.26 | 0.89 |

## 

## GET\_inventory/items

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.197 | 0.14 | 0.32 |
| 5 Users | 0% | 0.911 | 0.22 | 0.34 |
| 8 Users | 0% | 1.357 | 0.31 | 3.23 |
| 20 Users | 0% | 1.526 | 1.33 | 2.03 |

## GET\_loan-policy-storage/loan-policies

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.108 | 0.05 | 0.12 |
| 5 Users | 0% | 0.538 | 0.07 | 0.18 |
| 8 Users | 0% | 0.775 | 0.09 | 0.24 |
| 20 Users | 0% | 0.857 | 0.31 | 1.08 |

## GET\_proxiesfor

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.329 | 0.02 | 0.04 |
| 5 Users | 0% | 1.617 | 0.04 | 0.45 |
| 8 Users | 0% | 2.329 | 0.06 | 0.22 |
| 20 Users | 0% | 2.578 | 0.26 | 1.46 |

## GET\_service-points

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.197 | 0.03 | 0.06 |
| 5 Users | 0% | 0.912 | 0.05 | 0.10 |
| 8 Users | 0% | 1.358 | 0.06 | 0.21 |
| 20 Users | 0% | 1.527 | 0.29 | 1.20 |

## 

## GET\_staff-slips-storage/staff-slips

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.088 | 0.04 | 0.11 |
| 5 Users | 0% | 0.375 | 0.05 | 0.23 |
| 8 Users | 0% | 0.583 | 0.07 | 0.24 |
| 20 Users | 0% | 0.668 | 0.28 | 0.94 |

## GET\_users

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.197 | 0.06 | 0.12 |
| 5 Users | 0% | 0.911 | 0.07 | 0.17 |
| 8 Users | 0% | 1.357 | 0.09 | 0.21 |
| 20 Users | 0% | 1.526 | 0.29 | 0.96 |

## OPTINOS\_groups

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.11 | 0.004 | 0.009 |
| 5 Users | 0% | 0.539 | 0.01 | 0.03 |
| 8 Users | 0% | 0.776 | 0.01 | 0.03 |
| 20 Users | 0% | 0.86 | 0.03 | 0.14 |

## OPTIONS\_check-out-by-barcode

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.11 | 0.004 | 0.01 |
| 5 Users | 0% | 0.539 | 0.01 | 0.03 |
| 8 Users | 0% | 0.776 | 0.01 | 0.06 |
| 20 Users | 0% | 0.86 | 0.03 | 0.25 |

## 

## OPTIONS\_circulation/check-in-by-barcode

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.088 | 0.008 | 0.02 |
| 5 Users | 0% | 0.375 | 0.01 | 0.03 |
| 8 Users | 0% | 0.583 | 0.01 | 0.07 |
| 20 Users | 0% | 0.668 | 0.03 | 0.13 |

## OPTIONS\_circulation/loans

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.219 | 0.004 | 0.01 |
| 5 Users | 0% | 1.078 | 0.01 | 0.03 |
| 8 Users | 0% | 1.552 | 0.01 | 0.09 |
| 20 Users | 0% | 1.719 | 0.03 | 0.16 |

## OPTIONS\_circulation/requests

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.416 | 0.007 | 0.02 |
| 5 Users | 0% | 1.985 | 0.01 | 1.01 |
| 8 Users | 0% | 2.908 | 0.01 | 0.09 |
| 20 Users | 0% | 3.244 | 0.03 | 1.02 |

## OPTIONS\_configurations/entries

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.219 | 0.005 | 0.02 |
| 5 Users | 0% | 1.078 | 0.01 | 0.07 |
| 8 Users | 0% | 1.552 | 0.01 | 0.09 |
| 20 Users | 0% | 1.721 | 0.03 | 1.04 |

## 

## OPTIONS\_groups

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.197 | 0.005 | 0.02 |
| 5 Users | 0% | 0.911 | 0.01 | 0.05 |
| 8 Users | 0% | 1.357 | 0.01 | 0.14 |
| 20 Users | 0% | 1.526 | 0.03 | 1.03 |

## OPTIONS\_inventory/items

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.197 | 0.004 | 0.04 |
| 5 Users | 0% | 0.911 | 0.01 | 0.21 |
| 8 Users | 0% | 1.357 | 0.01 | 0.21 |
| 20 Users | 0% | 1.526 | 0.03 | 1.1 |

## OPTIONS\_loan-policy-storage/loan-policies

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.11 | 0.004 | 0.01 |
| 5 Users | 0% | 0.539 | 0.01 | 0.04 |
| 8 Users | 0% | 0.776 | 0.01 | 0.03 |
| 20 Users | 0% | 0.86 | 0.03 | 0.15 |

## OPTIONS\_proxiesfor

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.329 | 0.004 | 0.02 |
| 5 Users | 0% | 1.617 | 0.01 | 0.04 |
| 8 Users | 0% | 2.329 | 0.01 | 0.09 |
| 20 Users | 0% | 2.58 | 0.03 | 0.24 |

## 

## OPTIONS\_service-points

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.088 | 0.03 | 0.04 |
| 5 Users | 0% | 0.375 | 0.04 | 0.10 |
| 8 Users | 0% | 0.583 | 0.06 | 0.35 |
| 20 Users | 0% | 0.668 | 0.26 | 0.98 |

## OPTIONS\_staff-slips-storage/staff-slips

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.088 | 0.004 | 0.01 |
| 5 Users | 0% | 0.375 | 0.01 | 0.05 |
| 8 Users | 0% | 0.583 | 0.01 | 0.10 |
| 20 Users | 0% | 0.668 | 0.03 | 0.19 |

## OPTIONS\_users

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.197 | 0.01 | 0.02 |
| 5 Users | 0% | 0.912 | 0.01 | 0.08 |
| 8 Users | 0% | 1.357 | 0.01 | 0.10 |
| 20 Users | 0% | 1.527 | 0.04 | 1.05 |

## POST\_login

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.014 | 0.32 | 0.32 |
| 5 Users | 0% | 0.086 | 0.30 | 0.30 |
| 8 Users | 0% | 0.109 | 0.36 | 0.36 |
| 20 Users | 0% | 0.131 | 0.51 | 0.64 |

## 

## POST\_circulation/check-in-by-barcode

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 0% | 0.088 | 0.38 | 0.45 |
| 5 Users | 0% | 0.374 | 0.60 | 0.90 |
| 8 Users | 0% | 0.583 | 0.90 | 1.40 |
| 20 Users | 0% | 0.667 | 3.81 | 5.32 |

## POST\_circulation/check-out-by-barcode

Combined tests execution data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Errors | Req/sec | 95th pct, sec | Max, sec |
| 1 User | 1.02% | 0.09 | 0.11 | 0.45 |
| 5 Users | 0.2% | 0.539 | 0.74 | 1.62 |
| 8 Users | 0.1% | 0.776 | 1.14 | 1.67 |
| 20 Users | 0% | 0.859 | 4.62 | 5.65 |

# CPU PROFILING RESULTS

CPU profiling of the most resources consuming **okapi** service showed 2 methods which had a high CPU usage and impact on the overall service performance.

A screenshot of a cell phone

Description automatically generated

1. **com.fasterxml.jackson.databind.ObjectMapper.readValue** method uses most of CPU capacity which leads to performance degradation

A screenshot of a cell phone

Description automatically generated

1. Logs **org.slf4j.impl.Log4jLoggerAdapter.log** method has high CPU usage with default INFO logging level

A screenshot of a cell phone

Description automatically generated

Changing log level from **INFO** to **ERROR** showed significant performance enhancement and increased saturation point in two times to ~**88** rps and **20** vUsers so we can make a conclusion that logging implementation needs to be reviewed and improved.

A screenshot of a computer

Description automatically generated

# VOLUME TESTING RESULTS

Volume testing is testing of an application with large number of data in database is possible, it verifies if the system responds as expected for a certain volume of data.

The default sample dataset was provided to performance team and was chosen as an example for data replication. Provided scripts for applying this dataset included a bunch of indexes which have a solid impact on the overall application performance.

Default sample dataset consists of the following files with records count:

* feefines-accounts: 234
* items: 277172
* instances: 227786
* servicePointsUsers: 158
* recordholdings: 227568
* circulate: 562
* request: 122
* logins: 24
* names: 174
* notes: 171
* notifications: 585
* perms: 174

Items, instances and recordholdings files have been chosen for replicating with following numbers of records for each file:

* [227 000](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578593709481&to=1578597514871&var-percentile=95&var-test_type=fix_load&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All)
* [500 000](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578048787312&to=1578052621268&var-percentile=95&var-test_type=baseline&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All)
* [1 000 000](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578053106815&to=1578056997664&var-percentile=95&var-test_type=baseline&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All)
* [3 000 000](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578058345209&to=1578062441330&var-percentile=95&var-test_type=baseline&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All)
* [6 000 000](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1578479145984&to=1578481108287&var-percentile=99&var-test_type=baseline&var-test=folio&var-env=int&var-grouping=10s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All)

Existing indexes for fields of Items, instances and recordholdings files:

1. Items

* item\_materialtypeid\_idx\_gin
* item\_status\_name\_idx\_gin
* item\_barcode\_idx\_gin
* item\_title\_idx\_gin
* item\_id\_idx
* item\_status\_name\_idx
* item\_barcode\_idx
* item\_materialtypeid\_idx
* item\_pkey
* item\_holdingsrecordid\_idx

1. Instances

* instance\_title\_idx\_gin
* instance\_id\_idx
* instance\_title\_idx
* instance\_pkey
* instance\_subjects\_idx\_gin
* instance\_classifications\_idx\_gin
* instance\_languages\_idx\_gin
* instance\_instancetypeid\_idx\_gin
* instance\_identifiers\_idx\_gin
* instance\_contributors\_idx\_gin
* instance\_publication\_idx
* instance\_contributors\_idx

1. Recordholdings

* holdings\_record\_permanentlocationid\_idx\_gin
* holdings\_record\_instanceid\_idx\_gin
* holdings\_record\_permanentlocationid\_idx
* holdings\_record\_instanceid\_idx
* holdings\_record\_pkey
* holdings\_record\_id\_idx

As the test parameters, 80% of capacity was chosen which corresponds to **8** vUsers for a duration in **3600** seconds. Test results showed that there is no performance degradation related to increasing data in these files as there are required indexes for all frequently used fields that reduce the huge load on the database instance resources.

The image below presents the database CPU usage with 6 000 000 items, instances and holding records in the database:

A screenshot of a cell phone

Description automatically generated

Slowest queries which took the most of execution time were initiated by the **mod-circulation-storage** service presented in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| Total time | Average Time | Calls | Query |
| 60% | 360 ms | 380458 | SELECT COUNT(\*) FROM fs08000010\_mod\_circulation\_storage.loan WHERE (to\_tsvector($1, f\_unaccent(loan.jsonb->>$2)) @@ to\_tsquery($3, f\_unaccent($4))) AND (lower(f\_unaccent(loan.jsonb->$5->>$6)) NOT LIKE lower(f\_unaccent($7))) |
| 10% | 70 ms | 339443 | SELECT jsonb,id FROM fs08000010\_mod\_circulation\_storage.loan WHERE (to\_tsvector($1, f\_unaccent(loan.jsonb->>$2)) @@ to\_tsquery($3, f\_unaccent($4))) AND (lower(f\_unaccent(loan.jsonb->$5->>$6)) NOT LIKE lower(f\_unaccent($7))) LIMIT $8 OFFSET $9 |

Based on this information it was decided to increase circulate.tsv file to populate more data in **mod\_circulation\_storage.loan** table. 10%, 30% and 50% of the total items record count were chosen for the records of the loan.

## 27700 loan records (10% of default items count)

A screenshot of a cell phone

Description automatically generated

Database CPU was loaded around 20%

## 83100 loan records (30% of default items count)

A screenshot of a social media post

Description automatically generated

Database CPU was loaded around 50%

## 138500 loan records (50% of default items count)

A screenshot of a social media post

Description automatically generated

Database CPU was loaded around 80%

# LONGEVITY TESTING RESULTS

The [longevity](http://ec2-3-83-89-118.compute-1.amazonaws.com/grafana/d/q69rYQlik/jmeter-performance?orgId=1&from=1577315465206&to=1577345125436&var-percentile=95&var-test_type=longevity&var-test=folio&var-env=demo&var-grouping=30s&var-low_limit=250&var-high_limit=700&var-db_name=jmeter&var-sampler_type=All) performance test is a long high load test to check if the system can work with no issues for a long time. This is the most important stability test because it assumes no extreme cases, but rather a normal operation over time.

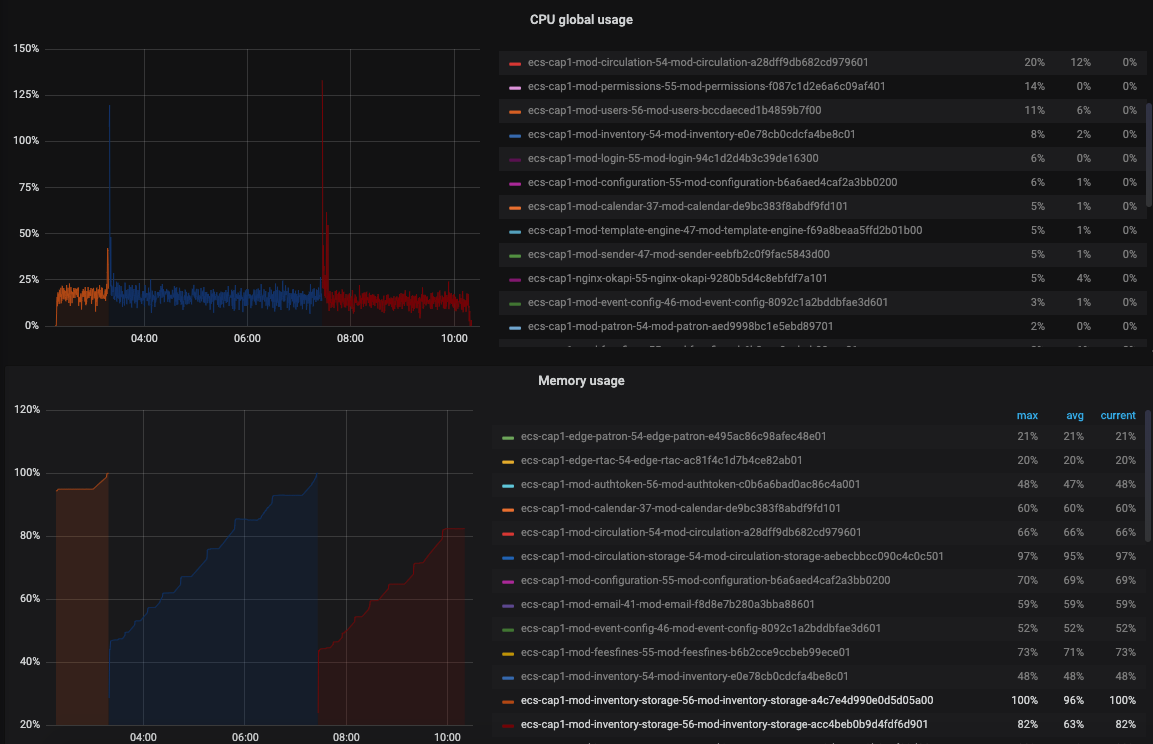
**28800** seconds (8 hours) test duration was chosen for longevity performance test which approximately corresponds to one working day.

A circuit board

Description automatically generated

Most of 1806 failed requests were related to **/service-points** requests that use **mod-inventory-storage** service

Resources utilization investigation revealed that there were several restarting of the **mod-inventory-storage** service during the test execution when service exceeds 100% of allocated RAM memory.



A black and silver text on a screen

Description automatically generated

With increasing loans count in the database over the time database CPU load was increasing accordingly so that also had an impact on the response time of the **/circulation/loans** request

A screen shot of a computer

Description automatically generated

A screenshot of a map

Description automatically generated

# RECOMMENDATIONS

1. From the configuration perspective, it is recommended to install okapi on the separate instance as it is the most resources consuming service and could affect efficient work of the other services
2. **com.fasterxml.jackson.databind.ObjectMapper.readValue** method of okapi serviceoveruses CPU resources as there are a lot of JSON decoding, this implementation could be reviewed and improved to reduce operations with JSON
3. **org.slf4j.impl.Log4jLoggerAdapter.log** method has high CPU usage which has an impact on the overall application performance. This method is related to logging functionality so logging implementation could be reviewed and improved
4. The **mod-inventory-storage** service’s RAM memory leak needs to be investigated and fixed as it leads to service restarting and causes errors
5. Indexes for the most used fields in loans table should be added because according to the longevity and volume testing increasing loans in the database leads to high database CPU usage and performance degradation as the result
6. Slow **mod\_circulation\_storage.loan** queries need to be improved or indexes should be added to speed up query execution time