

# Scalability report

E-Heza Application

2022.09.14.

Aron Novak , Senior Engineer, Gizra

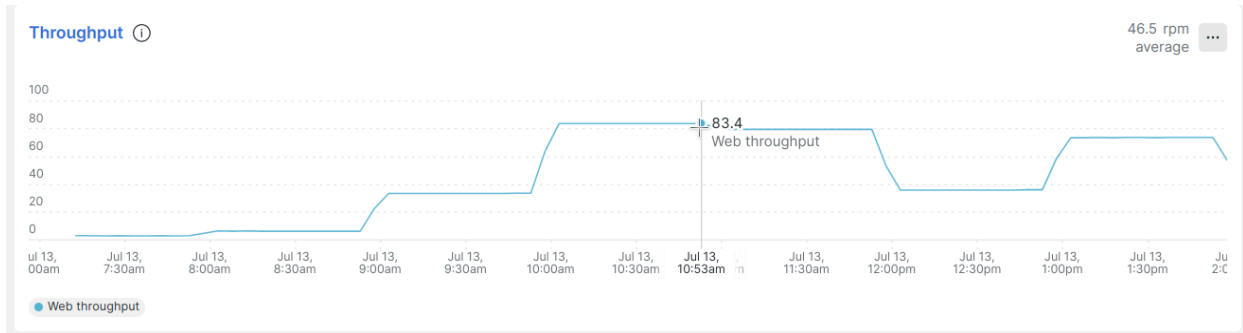
## Executive Summary

The E-Heza Application with Drupal backend and Elm frontend is architected to scale well; the frontend application is written with low-bandwidth conditions considered, so it communicates with the backend in batches rather than constantly generating requests on every activity, and it relies on the local device's persistent memory so as to not lose data. This architectural technique is helpful to serve many health centers and many healthcare workers in parallel as the load is distributed widely and in very small batches. In its current use case, E-Heza does not experience bandwidth related issues or slowdown, but for the purposes of understanding scalability, this report sought to determine how tests with actual data, and realistic stress testing, might reveal scalability issues, if any, at the software or in the infrastructure level. After two phases of tests, the first with a realistic increase in requests to demonstrate reasonable national expansion, and a second with an increase in requests well beyond any known implementation needs.

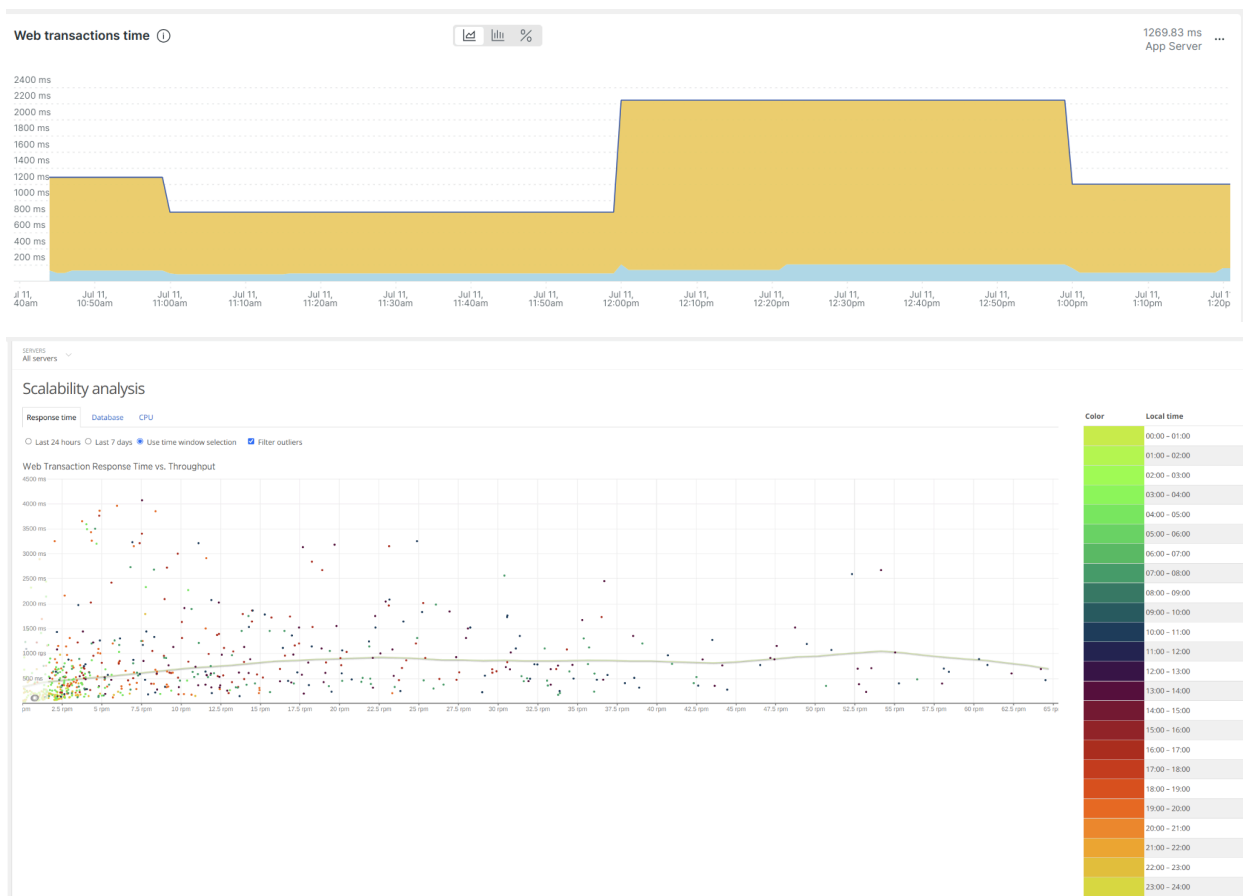
## Baseline

Currently E-Heza is in use in Rwanda among five districts with a patient population of over 115,000 and a health worker population of over 750, divided among nurses who work with entire health center catchment areas and CHWs who work mainly with individual villages. We carefully studied NewRelic reports in peak periods, how many requests hit the system and what's the average response time in those busy periods.

On average, there are ~10 requests per minute (RPM), in peak periods, and at some points it can be as high as ~85 RPM.



During peak hours, the web transaction time can be as high as two seconds, but due to the above mentioned technique - silently and opportunistically syncing in the background — the user experience of the client devices does not deteriorate.



The scalability analysis from NewRelic shows that as the RPM increases, the average response time does not increase. There are some slow operations in the system that take a significant amount of time (due to the volume of the content in the system), but when there are many requests, the whole application scales well, and there is no slowdown as the traffic grows - at least for this amount of traffic that currently hits the application.

## Current Infrastructure

The tested hosting environment has a container-based architecture, in which there are two application servers that are responsible for PHP execution, each of them has 2GB of RAM available for the application (allowing 8 PHP workers), and has an SSD-backed shared filesystem for the application servers, limited to 30 GB of storage. In addition, there is a Varnish edge cache, which is useful mostly for static assets like images.

## Testing Technique

To make a realistic test, we implemented a scalability test suite, available as open source code at <https://github.com/TIP-Global-Health/eheza-app/pull/442>, that takes the following steps:

- Creates a virtual machine (VM) in the cloud - the location of the VMs was Linode Germany datacenter
- Installs Chrome inside that VM
- Executes a set of actions (fake nurse session) inside the browser, including:
  - Pair a new device
  - Login as a new nurse
  - Sync Nyange Health Center
  - Record a new patient with a photo
  - Sync Nyange Health Center
  - Repeats the blue loop 50 times

In addition, we execute a script in the background that repeatedly requests the latest logs from the system, executes periodic tasks (cron), and so on.

## Some Limitations of the Test Suite

- It only uses one Health Center, as it fits easier in the cache.
- It only deals with patient creation, not with other types of action

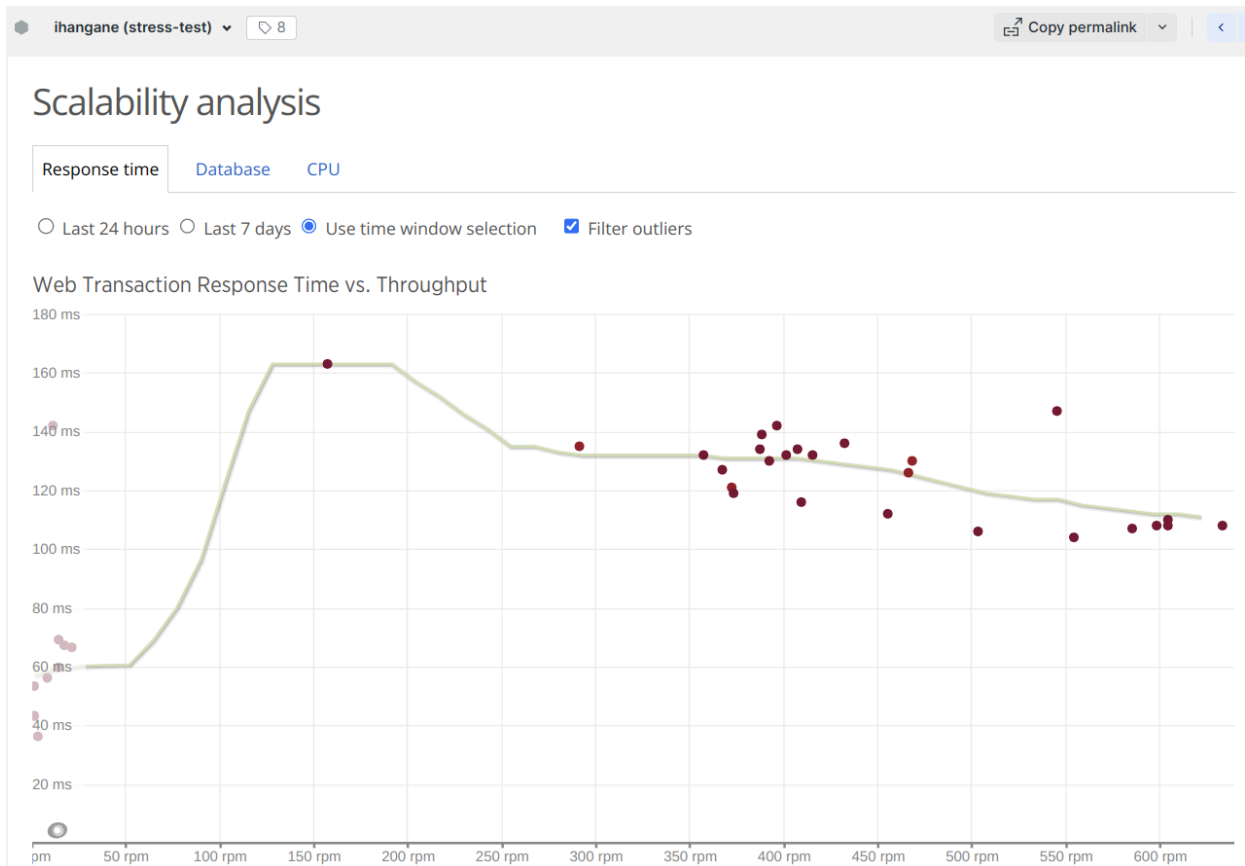
However the fake nurse session in some ways use much more resources than the real one:

- It executes the actions much faster than a human would
- Invokes the data sync operations often, which is the part which demands the most of the infrastructure and challenges scalability the most.

We evaluate the results with this in mind, knowing that real-world scenarios might be different, but not likely to challenge scalability in more significant ways.

## Phase one

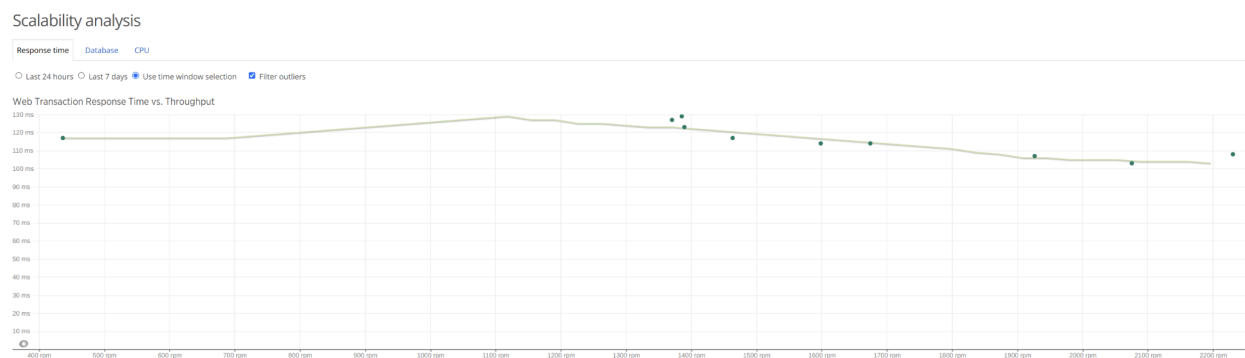
Initially we started the examination with 10 fake devices and 10 fake nurses, as we wanted to see how much RPM we can generate on the system.



It is 7X times higher than the current peak hour, and the system behaved well in the same way, the response times remained consistently low.

## Phase two

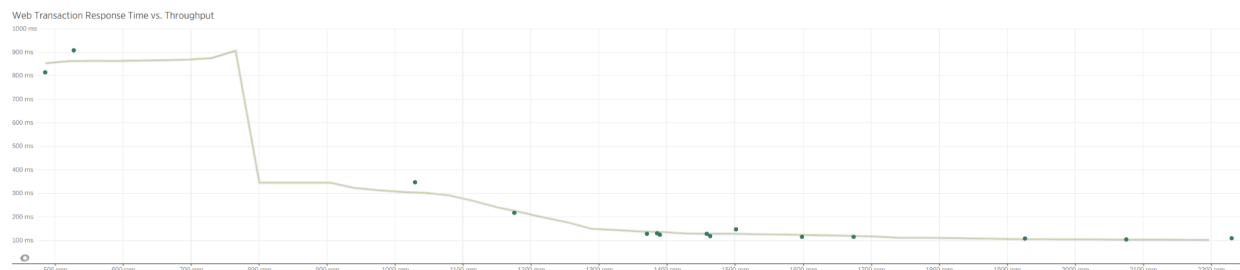
As the results were very promising from the phase one tests, we then conducted tests of one order of magnitude larger in phase two. We provisioned 100 virtual machines and repeated the test.



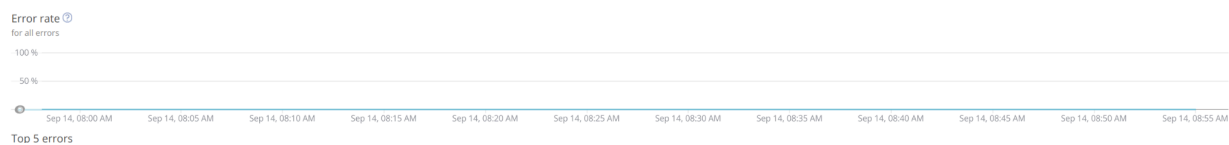
As seen above, we could generate ~2200 RPM this way, which is more than 20X of the current peak periods. During this test, thousands of new nodes were created (new person nodes). The

Web Transaction Response Times are in a narrow range (100-120 ms), which means that the increase in the amount of traffic did not exhaust the capacity of the infrastructure.

In the middle of the phase two tests, I initiated a full cache clear to see the effect of that, as a full cache clear can easily happen in a live environment, due to a deployment or other complicated action.



The New Relic scalability analysis chart changed immediately after the response time was increased to 1 second from 100-200 ms, which is still adequate for performance. We checked the error rate chart too, if the good performance is only because of failures, but it is not the case:



Also we kept an eye on the number of nodes, to make sure that indeed creation of new nodes happens accordingly:

```
MariaDB > select count(*) from node;
```

```
+-----+
| count(*) |
+-----+
| 1770802 |
+-----+
```

**1 row in set (0.619 sec)**

```
MariaDB > select count(*) from node;
```

```
+-----+
| count(*) |
+-----+
| 1770858 |
+-----+
```

**1 row in set (0.626 sec)**

```
MariaDB > select count(*) from node;
```






```
+-----+  
| count(*) |  
+-----+  
|  1770930 |  
+-----+
```

**1 row in set (0.588 sec)**

```
MariaDB > select count(*) from node;
```

```
+-----+  
| count(*) |  
+-----+  
|  1771168 |  
+-----+
```

**1 row in set (0.589 sec)**

Containers 		Instances		
1	18			
Table Charts				
Container 	Response time 	Throughput 	Error rate 	CPU utilization 
appserver-4cd71002-php-6568caa2c033468daf1779db9e95b...	173 ms	616 rpm	0%	35.48%

Also NewRelic reports that the second test resulted in ~35% average CPU utilization.

## Conclusions

Based on the stress test outcome, we can confidently say that E-Heza can be upscaled to a national system and likely to multiple countries in a single instance (although that is not a likely scenario). First of all, the architecture of the application supports scaling and the infrastructure provides a fine-tuned environment which gives great performance. Due to the test method limitations we are aware of the possibility that adding multiple larger countries to the system may require additional fine-tuning of the infrastructure performance-wise, but on both software and infrastructure level, E-Heza is ready to scale.