

**Collaboard Sizing**

PRIVATE-CLOUD sizing recommendations

|  |  |
| --- | --- |
| Title: | Sizing Recommendations |
| Abstract: | This document our recommendation for te correct sizing of the environment |
| Status\* | Complete |
| Document Owner: | IBV |
| Document Author: | Gian Paolo Santopaolo |
| \*Draft, In progress, Reviewed, Authorized, Approved, Completed | |

|  |  |  |
| --- | --- | --- |
| **Document Control** | | |
| Document Distribution: | Internal: | All dev Teams |
|  | External: | Whom of interest |
| Document Authorization: | Authorizer(s): | Gian Paolo Santopaolo |

|  |  |  |  |
| --- | --- | --- | --- |
| **Change History** | | | |
| Date | Ver | Changed By | Details |
| 14/02/2022 | 1.0 | G.P Santopaolo | Document Release |

Index

[Goal 3](#_Toc127289021)

[Load test for Collaboard 3](#_Toc127289022)

[Comparing performance tests with real-world usage 3](#_Toc127289023)

[Our recommendation for a production environment 7](#_Toc127289024)

[Our recommendation for a staging environment 7](#_Toc127289025)

[Next steps 8](#_Toc127289026)

# Goal

This document aims to explain the concept of Collaboard's load test and to analyze the load test data coming from a cluster running on Azure Confidential Computing with Constellation and another running on Azure Kubernetes Service (AKS).

# Load test for Collaboard

Collaboard's load tests are designed to understand the limit of an environment and how many concurrent users it can tolerate by providing each connected user with a seamless user experience.

Usually, an environment with resource usage around 85-95% (CPUs, Ram, SQL Server, etc.) can still satisfy the above requirement.

To reach this result, any autoscaling possibility given by any provider is not taken into consideration.

For our load test, there are two main factors to keep in consideration:

* Number of concurrent active users
* Total requests generated by the active user

The first is a known parameter; the second can only be measured and varies from environment to environment.

To measure the total requests generated by an active user, we developed a custom telemetry, collecting all the data needed anonymously.

With such an approach, we can understand how many requests a user can generate per second.

We noticed that the RPUPS (requests per user per second) might vary significantly by the user's actions. For example, imagine a user following a workshop, passively watching the presenter.

On the other side, a user prepares a new project by adding content (images, documents, videos, links) and using all the functionalities available on the platform, which will generate way more RPUPS that the first use case.

The platform's usage, as said before, may also significantly vary by environment: in our public production environment, we have users using the platform seven days a week, 24/h a day.

At the customer's site one, users use the application five days a week, mostly eight hours a day, using Microsoft Office documents.

At customer's site two, we have multiple locations all around the world, users use the application five days a week, mostly eight hours a day (but the usage, in this case, is spread in different time zones), and they use PDFs and videos.

The number of use cases is endless.

As you can imagine, each use case generates a different load for the platform.

## Comparing performance tests with real-world usage

Our test consists of two phases:

* In the first phase, we collect anonymous usage data at the specific site
* In the second phase, we run a series of standard load tests until we reach the environment's limit

Once we collect all the needed data, we can import everything into our business intelligence (BI) and extract the estimates.

The estimates compare the total number of requests per second (RPS) the cluster can handle versus the average request per second a real user can generate using the specific environment (RPUPS).

We are able to understand the total number of the active user by time interval (usually 15 minutes)

Thanks to our telemetry, we can also understand the total HTTP, Websocket, and file operations requests in the same timeframe.

We end up with a massive table like the one below where we are reporting only a few rows to give the idea (usually, our sample is about 3000 rows, about one month's worth of samples)

Table

Description automatically generated

Sample table of the real RPUPS

This way, we know precisely the real RPUPS the environment was dealing with in any given time frame.

We are then able to aggregate the above data by the RPUPS ranges

Graphical user interface, table

Description automatically generated

Aggregated real RPUPS

This way, we know that for 1690 time frames (1690 rows of the table above, Sample table for real RPUPS) the average RPUPS of the cluster was between 0.00 and 0.10 RPUPS, average 0,0663 RPUPS.

For 876 times, we had a real RPUPS of 0,1326, and so on.

We then use K6 to perform several load tests against the environment until we reach a total cluster capacity of around 85%-95%. At the same time, the application still provides a seamless user experience (with no lag) to the user.

Once we are happy with the result, we choose one of the load tests as a reference and import the data collected inside our BI.

A K6 test has a maximum length of 30 minutes, to which, for the purpose of our tests, we have to remove the ramp-up and cool-down time. So usually, we have about 20 minutes of usable data.

With all the data collected by K6, we are primarily interested in the Web APIs logs, web sockets, and file logs that we can resume after all the aggregation and manipulation in the table below.

Graphical user interface, text, application

Description automatically generated

Data collected from K6

From the table above, we understand that during the test, the cluster was tested with 472.136 API calls, 294.295 web sockets calls, and 386.611 file operations requests, which makes the cluster capable of 382 RPS (request per second).

Now for the last part, we have only to match the RPS (request per second) with the RPUPS (request per user per second). This way, we will be able to understand how many users the cluster will be able to handle.

As said before, the number of RPUPS (request per user per second) varies and, based on it, also changes the number of concurrent users a cluster can sustain before becoming less responsive, which for a real-time application like Collaboard is not good.

Based on the anonymous data sampled on the environment, we already know how the users of the cluster behave, so we can make some predictions that are resumed in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| RPUPS range | Samples count | RPUPS avg | Estimated max online users |
| 0.00 – 0.10 | 912 | 0.0663 | 5.759 |
| 0.10 – 0.20 | 135 | 0.1236 | 3.091 |
| 0.20 – 0.30 | 7 | 0.2318 | 1647 |
| 0.30 - 0.40 | 2 | 0.3140 | 1216 |
| 0.40 – 0.50 | 4 | 0.4405 | 867 |

Max online users' estimation

In most sample cases, the cluster's users produced between 0.00 and 0.10 requests per user per second.

In this scenario, the cluster will be able to handle, with no problem, up to 5.759 concurrent active users.

In the second case, observed in 135 samples, the cluster will be able to handle, with no lag, up to 3.091 users.

Between 0.30 and 0.40 RPUPS, the cluster will be able to handle 1216 users.

And as the last sample, between 0.40 and 0.50 RPUPS, the cluster will be able to handle 867 simultaneous users.

These are the facto data results made with math and statistic calculations.

The engineering team shall discuss them with the customer to understand its needs and how to make benefit from them.

# Our recommendation for a production environment

To have a production environment able to reach the performances mentioned above, it shall be designed as following

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Server | Role | CPU | Memory | Runs |
| Bastion | Bastion | 2 | 4 | Tools |
| Server Node 1 | Master | 2/4 | 8/16 | Rancher |
| Server Node 2 | Master | 2/4 | 8/16 | Rancher |
| Server Node 3 | Master | 2/4 | 8/16 | Rancher |
| Agent Node 1 | Worker | 8 | 32 | Rancher |
| Agent Node 2 | Worker | 8 | 32 | Rancher |
| Agent Node 3 | Worker | 8 | 32 | Rancher |
| Agent Node 4 | Worker | 8 | 32 | Rancher |
| Agent Node 5\* | Worker | 8 | 32 | Rancher |
| TURN Server | Coturn | 2 | 8 | Coturn |

SQL Server

As an external service. At least 6 vCPU, 32 GB Ram, storage type local ssd. I/O latency 1-2 ms writes, 1-2 ms read. Max Data IOPS 32.000

File Share

Random Read/Write IOPS: 26.8k/20.1k.

Sequential Read/Write: 1227MiB/s / 1416MiB/s

Mixed Random Read/Write IOPS: 19.6k/6531

Network

As described in the chapter Network

With the sizing above, we will also run a Redis cluster. If the customer decides to have high availability for the components mentioned in the chapter Reliability, the Collaboard engineering team shall be informed and dive into the matter.

\*This extra node is needed only if the customer decides to run the logs with EFK + APM

## Our recommendation for a staging environment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Server | Role | CPU | Memory | Runs |
| Server Node 1 | Master | 2 | 8 | Rancher |
| Agent Node 1 | Worker | 8 | 32 | Rancher |
| TURN Server | Coturn | 2 | 4 | Coturn |

# Next steps

As mentioned above, we based the recommendation on the usage of our production environment web.collaboard.app.

The way your users may use the application may be similar or not to the one we got as a comparsion.

To have an exact overview of how many users your custom production environment can handle, we can offer, as a separate service, an analysis of your environment.

The analysis will consist:

* Collecting anonymous data about the usage your users are having of the application
* Perform a load test of your environment
* Analyze the collected data and came out with a precise forecast