avassa

Scalability Test Report

The Avassa Edge Platform

Avassa – Scalability Test Report

Introduction

Edge computing differs from the cloud in several fundamental ways:

- Limited footprint at each site
- Offline and intermittently connected environments
- A very large number of sites to manage (the focus of this article)
- ...and more

Choosing a platform that is *designed and validated* for these conditions makes all the difference.

If you take a cloud-centric solution and try to force it into an edge environment, you'll quickly run into obstacles.

One of the most defining requirements at the edge is the scale of the number of sites. This article describes the scale tests we perform at Avassa to ensure that our platform can manage deployments of up to 10,000 edge sites efficiently.

What this article covers

We'll walk through our scalability testing in the following way:

- **Test setup** – how the tests were performed and the environment configuration.
- **Test application** – description of the sample application deployed across edge sites.
- **Test results: resource usage** – results from the central orchestrator (Control Tower) in terms of memory, CPU, disk I/O, and network utilization.
- **Test results: performance** – time measurements for large-scale operations such as deploying an application to all sites.
- **Design principles** – background on how the Avassa architecture enables scalability.
- **Appendix** – detailed measurements and data points.

Key takeaways

We regularly perform scalability tests for environments of 100, 1,000, 5,000, and 10,000 sites. For each setup, we execute a set of representative operations — creating sites, simulating site registration ("call home"), and deploying applications. All tests use a simulated edge network that accurately implements the protocol of the Avassa Edge Enforcer agent.

The main results are:

- A 3-node Control Tower cluster (each node with 4 vCPU and 8 GiB RAM) efficiently manages 10,000 sites.
- When sites boot, they securely call home to the Avassa Control Tower to establish a trusted connection. This process — including certificate exchange and security handshakes — completes for all 10,000 sites in about 40 minutes, or roughly 0.2 seconds per site.
- Deploying an application to 10,000 sites takes approximately 18 minutes, or roughly 0.1 seconds per site.
- Client API calls summarizing the state of all 10,000 applications and sites complete within 20 seconds.
- Across all measurements, performance scales linearly with the number of sites.

In addition to automated tests, we also perform interactive scalability testing using the Control Tower Web UI.

Even with thousands of connected sites, the interface remains responsive and fluid for operations such as browsing site lists, viewing application states, and inspecting logs. A short recorded video demonstrates this responsiveness in practice.

These results confirm that the Avassa platform can manage very large-scale edge deployments with predictable performance, excellent responsiveness, and a minimal infrastructure footprint for the central orchestrator.

Test setup

For these tests, we use a hosted Avassa Control Tower configured in the same way as for production customer deployments. The Control Tower runs as a three-node cluster, each node being an AWS c6i.large instance equipped with:

- 4 vCPU
- 8 GiB RAM

The purpose of this test is to stress the Control Tower with the load generated by a large number of connected edge sites. To achieve this, we use an edge site simulator that fully implements the Avassa Edge Enforcer protocol. (These tests focus on central orchestration scalability — not on local edge resource use, which we evaluate separately in our *Edge Site Footprint* tests.)

To observe how performance and resource usage evolve with scale, we repeat the same set of tests across the following site counts:

- 100 sites
- 1,000 sites
- 5,000 sites
- 10,000 sites

In Avassa terminology, a *site* is a cluster of one or more hosts. For these tests, each site consists of a single host.

Test scenarios

For each of the site counts above, we measure the following scenarios:

1. **Creating sites in the Control Tower**

   Simulates batch registration of new edge sites — for example, pre-provisioning from an inventory file before hosts are shipped or installed.

2. **Edge Enforcer "call home"**

   When Edge Enforcer agents boot at the sites, they initiate secure connections to the Control Tower.

   In real deployments, this process would typically be distributed over time. To stress-test the system, we simulate all sites calling home almost simultaneously.

   The Control Tower includes built-in throttling to handle such high-load scenarios gracefully.

3. **Deploying an application across all sites**

   Represents a typical large-scale rollout of a new or upgraded application.

   In this test, no images or secrets are pre-cached at the sites, ensuring a true "cold-start" deployment.

   (In a normal upgrade scenario, only updated image layers are transferred, resulting in lower load.)

4. **Distributed pub/sub query (Volga)**

   The Avassa platform includes a built-in publish/subscribe bus for telemetry and logs — one instance per site and one in the Control Tower.

   This test performs distributed queries from the Control Tower to all sites and collects the aggregated results.

5. **Client API requests to the Control Tower**

   Measures API responsiveness in large-scale systems.

   These results also reflect the expected response times for the CLI and UI, both of which use the same API.

6. **Application deletion across all sites**

   Simulates retiring or undeploying an application globally.

7. **Site deletion from the Control Tower**

   Returns the system to an "empty" baseline state, ready for the next test cycle.

Each of these scenarios triggers a series of coordinated actions within the Avassa system.

Understanding these actions provides valuable context for interpreting the results, which we outline in the following sections.

Test case	Actions performed
Create sites in Control Tower	• A distribution certificate is generated for each host, used to authenticate host-to-host communication within a site. • A site-specific API certificate is created for the local site API. • Site-unique encryption keys are generated. - The Control Tower assigns controller nodes to the site.
Edge Enforcer calling home to Control Tower	• A pub/sub bus connection is established to the site. • Site configuration is transmitted to the site hosts. • Certificates and secret keys are distributed. • The site seal token is stored in the Control Tower. • If needed, the site automatically upgrades to the latest Edge Enforcer image (for these tests, we simulated the worst-case scenario that all sites were out of date).
Deploy an application across all edge sites	• The Control Tower's built-in container registry serves all application images; sites pull images directly from this registry. • Images are replicated locally if a site has multiple hosts. • Application secrets are distributed to the relevant sites. • The container runtime at the site starts the application. • The site reports deployment results back to the Control Tower. Rationale: Sites may not have access to arbitrary external registries, so the Control Tower must handle image distribution efficiently.
Volga query across all sites	• The Control Tower forms and distributes a query to all sites via the pub/sub bus. • Each site executes the query locally and returns 100 entries in this test scenario. • The Control Tower collects and merges all results.
Perform Control Tower API requests	• Get site summary status. • Get application deployment status. These requests measure Control Tower API responsiveness and reflect the expected performance of both the CLI and UI.
Delete the application from all edge sites	• A delete application deployment instruction is published on the pub/sub bus to all sites. • Each site stops the application locally. • The result of the stop operation is reported back to the Control Tower.
Delete all sites	• A site delete message is published on the pub/sub bus to all sites. • Each site cleans up local data and removes its configuration. • The Control Tower removes all state information related to the deleted sites.

Metrics collected

For each test configuration, we measure the following metrics across all three Control Tower instances:

- **Task completion time** – the total time required for each test scenario to finish.
- **Memory usage** – peak and average RAM consumption.
- **CPU utilization** – overall load across the cluster nodes.
- **Disk I/O** – read and write operations during each test.
- **Network traffic** – data received and transmitted by the Control Tower.

These metrics provide a complete view of how Control Tower resources scale with an increasing number of connected edge sites.

Test application

For these tests, we use the Movie Theaters demo application as the workload. It consists of three multi-architecture container images. Each site pulls only the image layers relevant to its architecture. In this test setup, all sites run on x86 architecture.

The compressed image sizes (as reported by docker manifest inspect -v <container>) are approximately:

- projector-operations – 69 MB
- digital-assets-manager – 69 MB
- curtain-controller – 64 MB

Each site also receives two secrets, distributed along with the application at deployment time.

The compressed image size is the relevant metric for these tests, as it represents the data actually transferred from the Control Tower to the edge sites during deployment — not the uncompressed size used at runtime.

Test results: resource usage

The graphs below illustrate resource usage when managing 10,000 sites. For comparison, the appendix includes corresponding graphs for the 1,000, and 5,000 site setups.

Each test case described earlier — such as *create sites* and *call home* — is executed sequentially. In the diagrams, each blue bar marks the start of a specific test phase. The plots display aggregate resource utilization across all three nodes in the Control Tower cluster, including CPU, memory, disk I/O, and network activity.

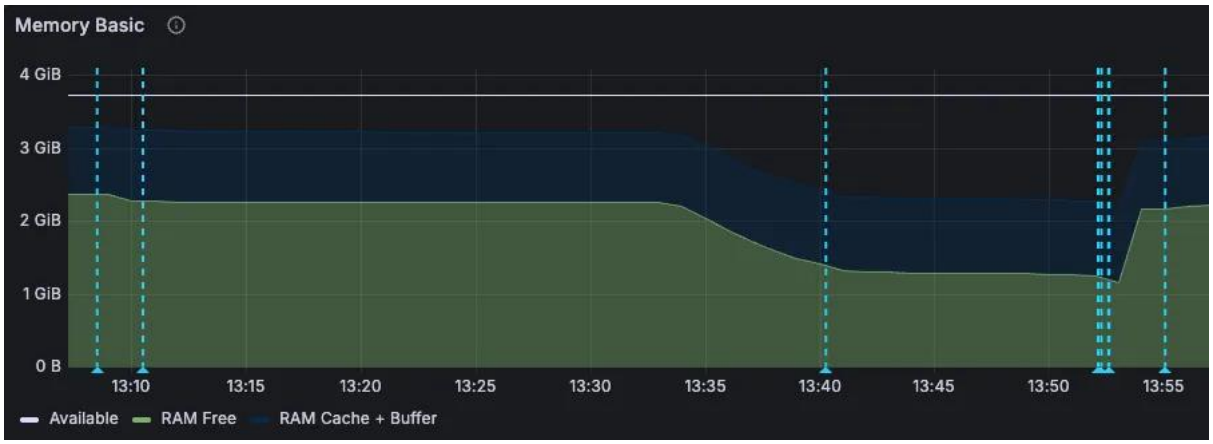
Observations on resource utilization

When updating the Edge Enforcer during *call home* operations or deploying applications, the image download process is the most network-intensive activity in the system.

Memory

Across all tests, the Control Tower uses approximately 2 GiB of RAM or less in total across the three cluster nodes, even when managing 10,000 sites.

This demonstrates a low and stable memory footprint that scales predictably with system size.



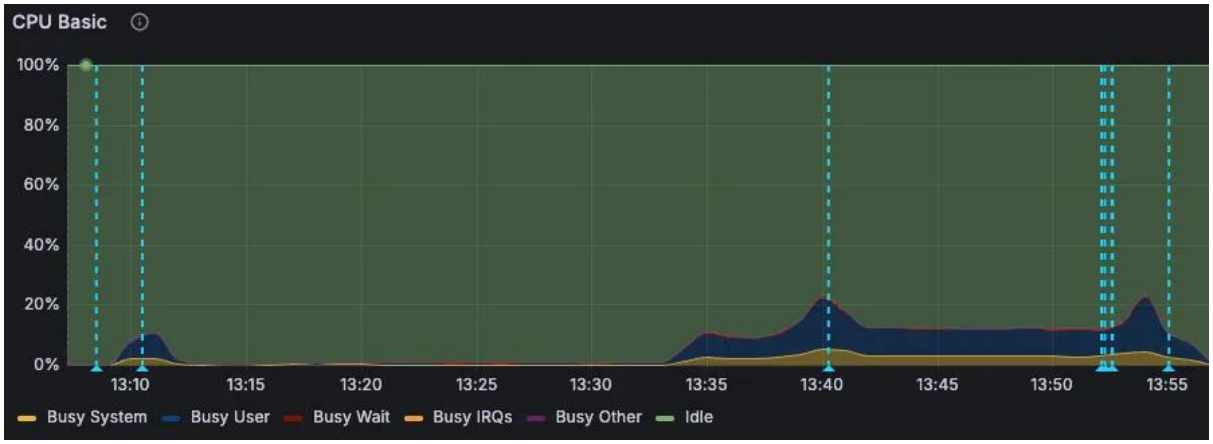
CPU

CPU load is evenly distributed across the three hosts.

The most CPU-intensive operations are:

- Creating and deleting sites
- Deploying applications

Even under these workloads, the Control Tower remains well within capacity — the cluster of three nodes with four vCPUs each comfortably handles all test cases at the 10,000-site scale.



Network

During application deployment, the Control Tower serves container images to all connected sites.

This is reflected in the network transmit peaks observed during the *deploy application* test case, which is consistent with the expected transfer of compressed image layers to 10,000 sites.

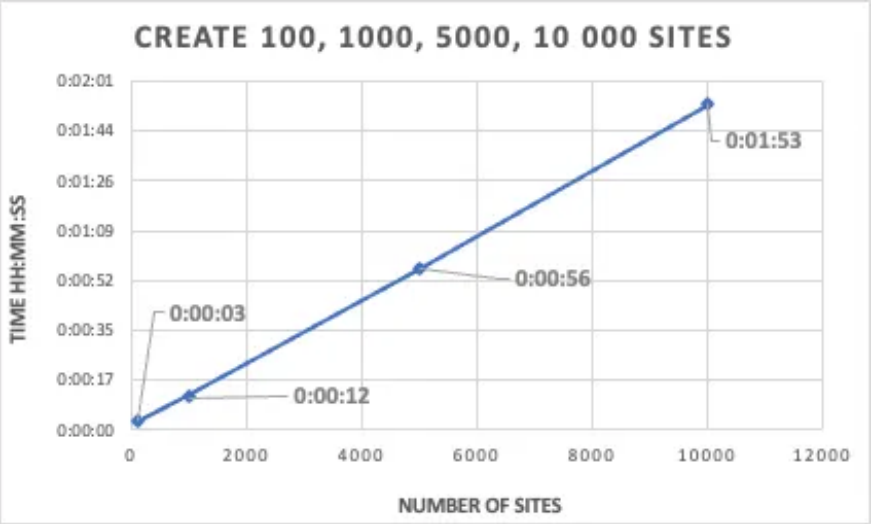
Test results: performance

The following performance results summarize the time-to-complete measurements for the different test setups — 100, 1,000, 5,000, and 10,000 sites.

In all cases, the results demonstrate linear scalability, confirming that the Avassa Control Tower maintains predictable behavior as the number of managed sites grows.

Create sites

The diagram below shows how creation time increases linearly from approximately 3 seconds for 100 sites to 1 minute and 50 seconds for 10,000 sites**.**



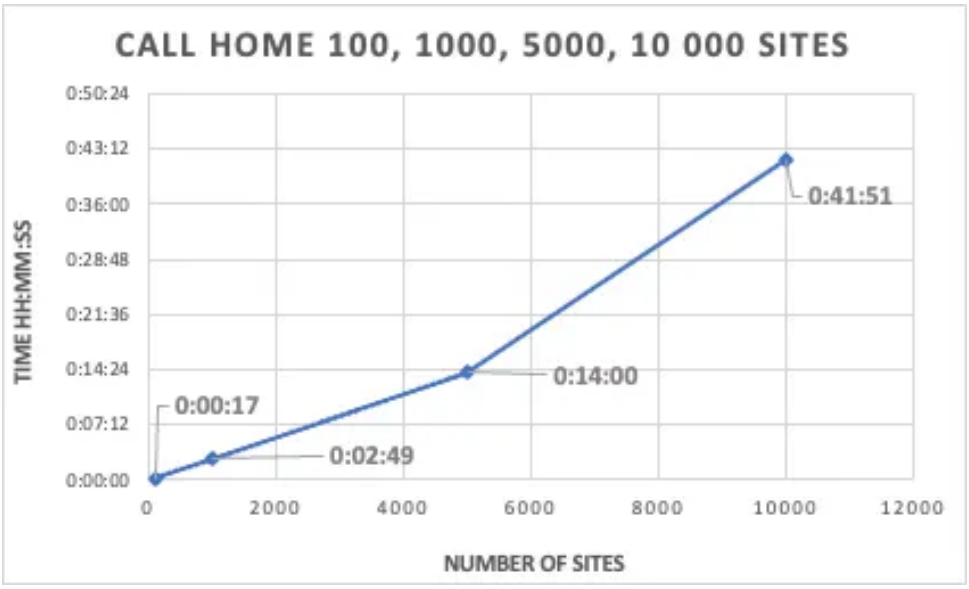
Creating sites in the Control Tower exhibits a linear time dependency.

The most compute-intensive operations in this phase are cryptographic functions, such as certificate generation and trust establishment.

As a rule of thumb, it takes about 10 seconds per thousand sites to create sites in bulk.

Edge Enforcer call-home

When Edge Enforcer agents call home to the Control Tower, they perform several coordinated steps: upgrading the Enforcer if needed, distributing encryption keys, and establishing secure communication.

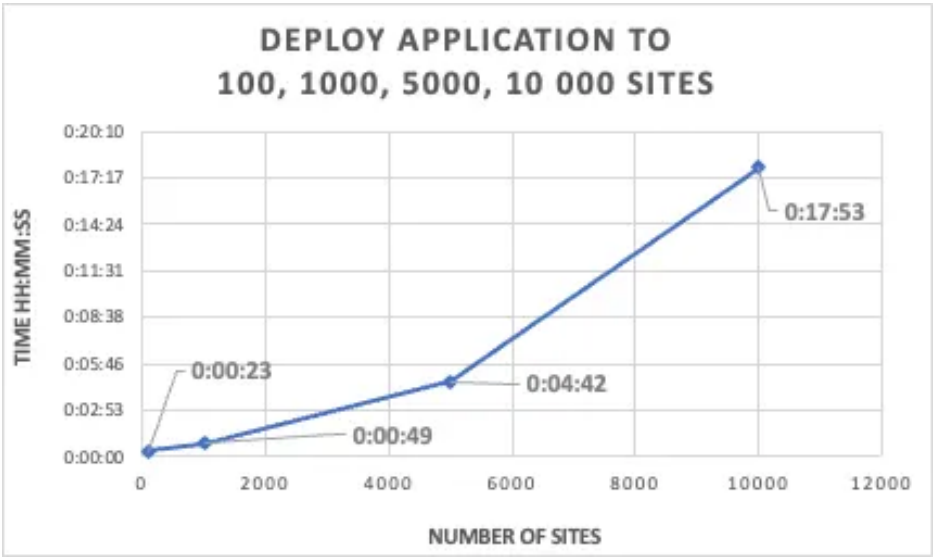


This operation scales linearly as well.

Across all site counts, it takes approximately 0.2–0.3 seconds per site, corresponding to about 5 minutes per thousand sites for the complete call-home process.

Deploy an application across all edge sites

A line graph titled *"Deploy application"* shows application deployment time for 100 to 10,000 sites (using a rolling batch of 100 sites).



Time increases linearly with the number of sites. This measurement addresses the key question:

How fast can you deploy or upgrade an application across all edge sites?

The process involves distributing three container images (each roughly 65 MB compressed).

In upgrade scenarios, only modified image layers are transferred, resulting in faster completion times.

On average, full application deployment takes 0.1 seconds per site, again demonstrating linear scaling across all test sizes.

Distributed Volga query (pub/sub bus)

The Avassa platform includes a built-in publish/subscribe bus, known as Volga, with one instance running at each site and one in the central Control Tower.

Producers publish data to topics, while consumers subscribe to selected topics of interest.

Common topics include host metrics, audit logs, and container logs — the latter meaning that Avassa continuously collects all container log entries at each site.

From the Control Tower, operators can perform distributed queries to search for specific log entries or metrics across a chosen set of sites.

In this test, we executed a log search across all sites and merged the results centrally.

Response times scale predictably — from sub-second for 100 sites to less than 10 seconds for 10,000 sites.

Client API requests

Client requests that summarize data across all sites and applications demonstrate similarly responsive behavior.

For the largest test setup (10,000 sites), API response times range from below one second to less than 20 seconds.

This confirms that the CLI, Avassa UI, and REST API remain fast and interactive even at a large scale.

The API calls measured were:

- Application status summary – retrieves the aggregated state of all deployed applications.
- Site status summary – retrieves the aggregated state of all managed sites.

Avassa scaling design principles

With ambitions to support tens of thousands of sites, Avassa's architecture has been shaped by a few deliberate design choices.

The primary principle is simple: push as much work as possible to the Edge Enforcer at the edge, while keeping the Control Tower lightweight.

This approach not only enables scalability and even distribution of processing load but also ensures that each site remains autonomous — continuing to operate even during network outages.

Transport layer design

Each Edge Enforcer needs to maintain a secure communication channel with the Control Tower.

Traditional TCP-based TLS can become a scalability bottleneck for large deployments. Without multiplexing, multiple control channels per host would be required, resulting in high overhead —

both in repeated TLS handshakes and in maintaining thousands of TLS sessions on the Control Tower. To address this, we chose the QUIC protocol — essentially *TLS over UDP with built-in multiplexing*. QUIC allows multiple logical streams to share a single connection, eliminating the need for separate tunnels and drastically reducing connection management overhead.

It's a *double win*: fewer connections and simpler architecture.

Efficient publish and subscribe

Over the QUIC connection, Avassa runs its built-in pub/sub protocol, called Volga. When the Control Tower needs to communicate with one or more sites, it publishes messages locally tagged with their target sites. Each site maintains a subscriber (over QUIC) that listens for relevant messages. The Control Tower's role remains lightweight — it simply publishes, while the sites handle message tracking. If a site has been disconnected, it automatically requests a replay of any missed messages once reconnected.

Volga also supports distributed queries: when a query is issued from the Control Tower, it is simply forwarded to the selected sites for local execution.

This keeps the Control Tower's processing load minimal, regardless of query scope.

Pushing work to the edge

When deploying an application to a group of sites, the Control Tower publishes a message on Volga identifying which sites should start, upgrade, or remove that application. From there, each site handles the heavy lifting — local scheduling, dependency resolution, and execution. The Control Tower remains a coordination layer, not a bottleneck.

In short, the guiding design idea is clear:

Do as little as possible in the central Control Tower, and push as much intelligence and work as possible toward the edge.

Scaling to 50,000 sites

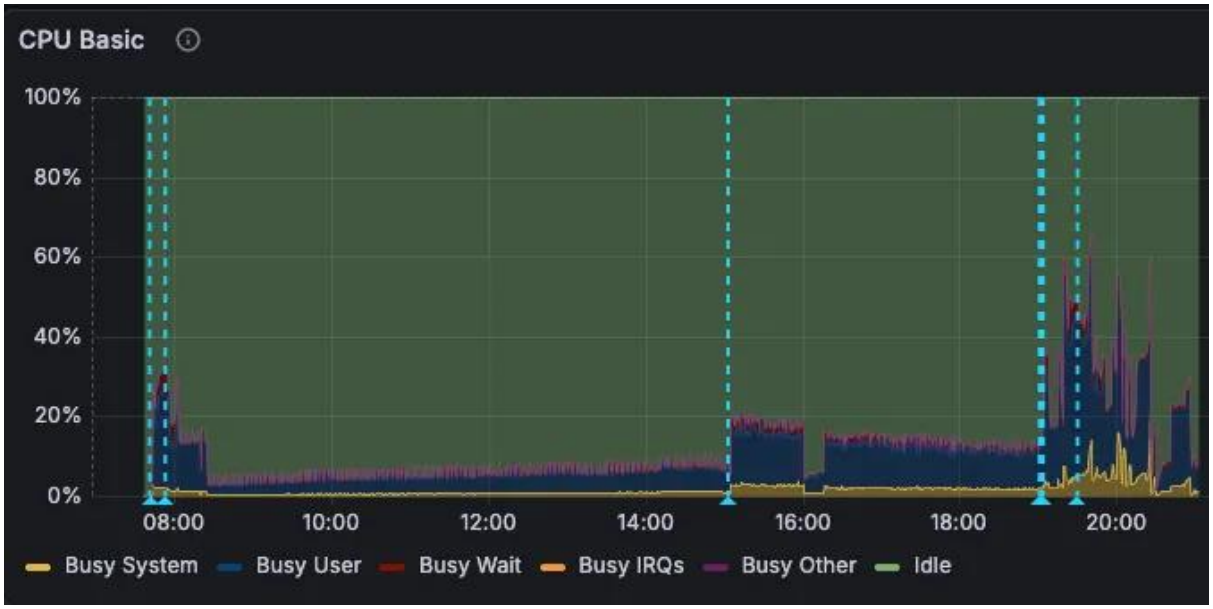
We have also conducted scalability tests with

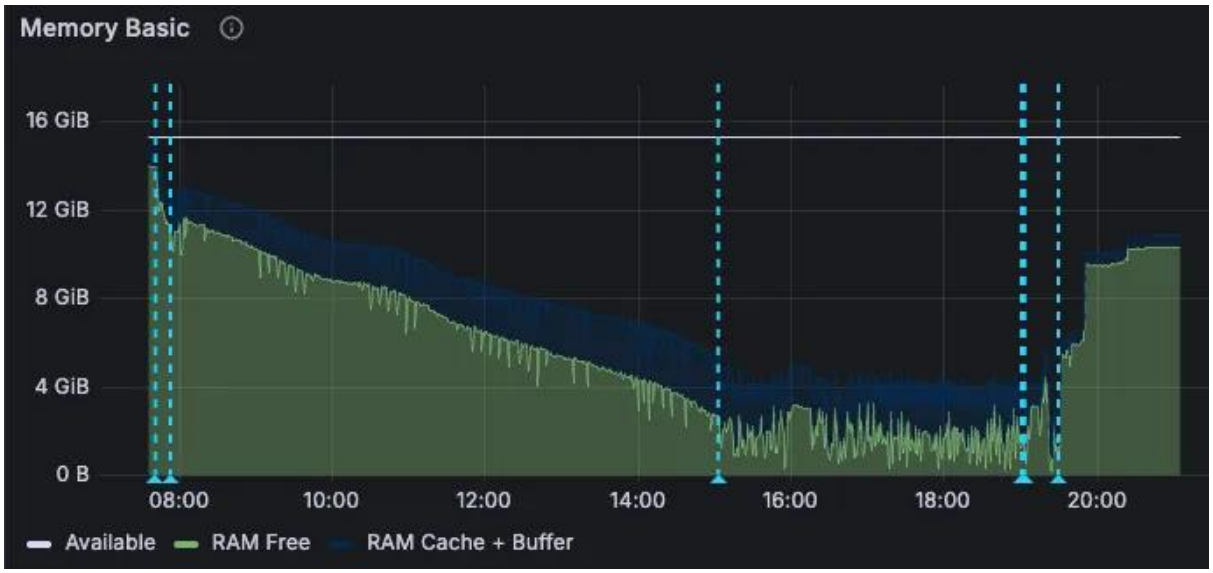
- 50,000 sites
- 8 vCPUs and 16 GiB RAM

The results below are rounded and compared with the corresponding 10,000-site results (in parentheses)

- Create 50,000 sites: 11 min (7 min)
- Call home (Edge Enforcer): 7 h (1 h)
- Deploy application: 4 h (20 min)
- Volga query: 1.5 min (10 s)
- Client commands: 1 min (20 s)
- Delete sites: 26 min (3 min)

Resource consumption graphs for the 50k test runs are available below for internal reference.





Conclusion

At Avassa, we focus on simplifying and scaling application lifecycle management at the edge.

Real-world edge deployments often span tens of thousands of sites, making scalability not just a performance goal but a design requirement.

This report has outlined how we validate and measure scalability through systematic testing.

As experience shows, you can't assume that a system performing well at a few hundred nodes will behave the same at ten thousand — you have to test it.

Our results confirm that 10,000 edge sites can be managed efficiently with a small Control Tower footprint and fast, predictable deployment times across all sites.

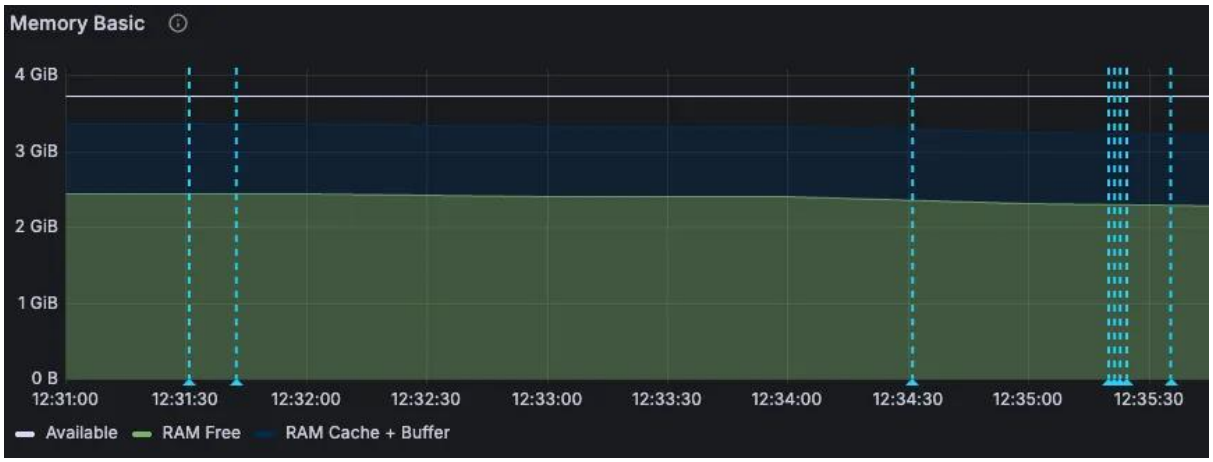
That combination — small central infrastructure, fast operations, and genuine autonomy at the edge — is at the heart of the Avassa platform design.

Appendix

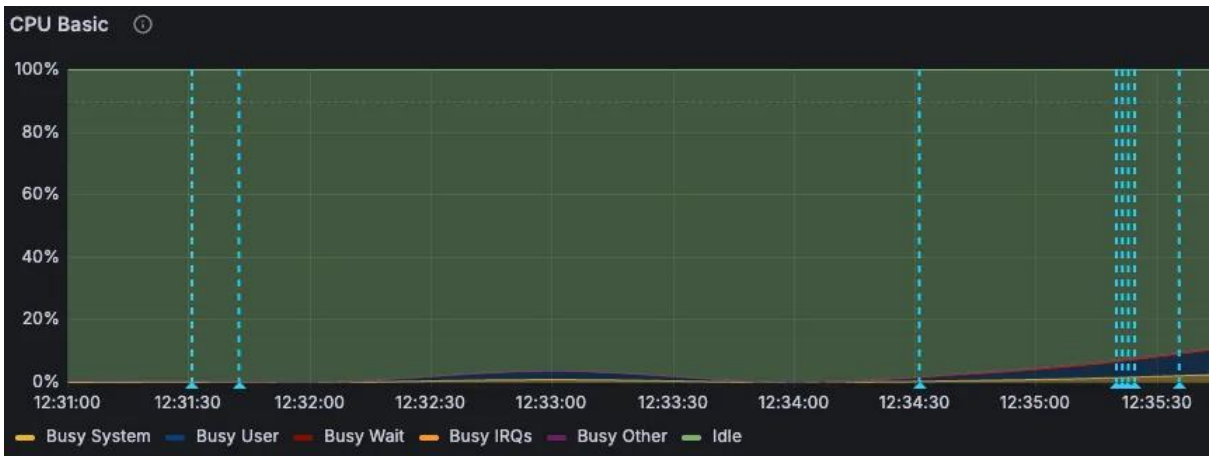
Below are test results for test batches: 1,000; 5,000; and 10,000 sites/clusters.

1000 Sites

Memory



CPU

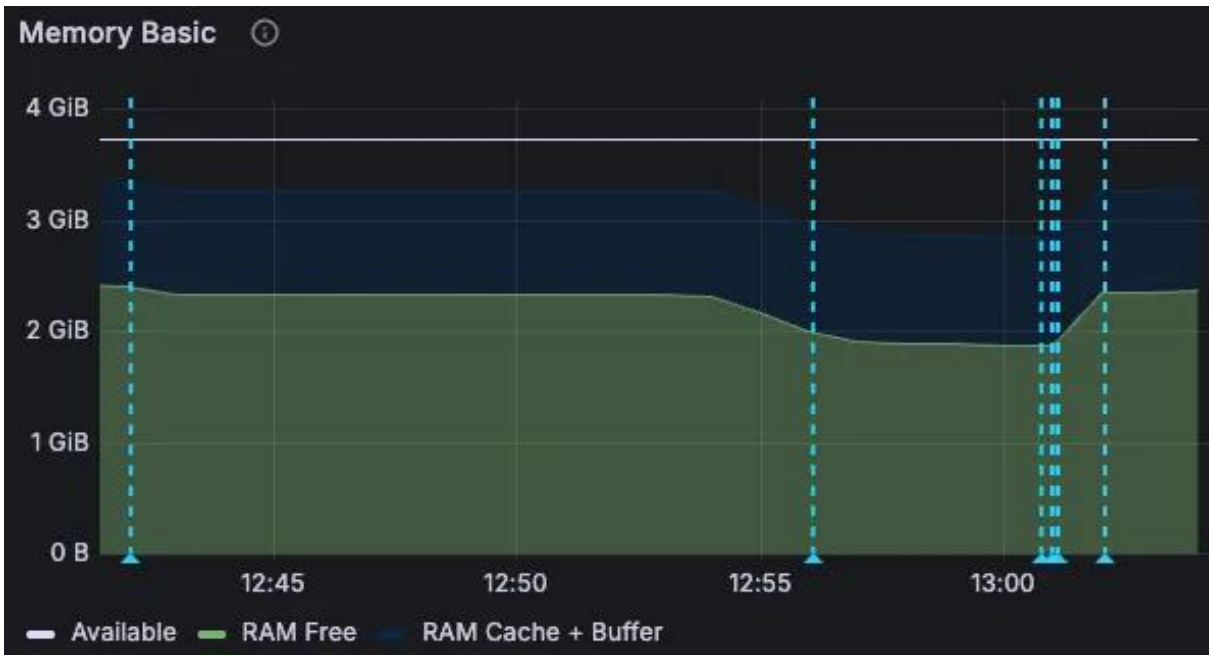


Disk IO

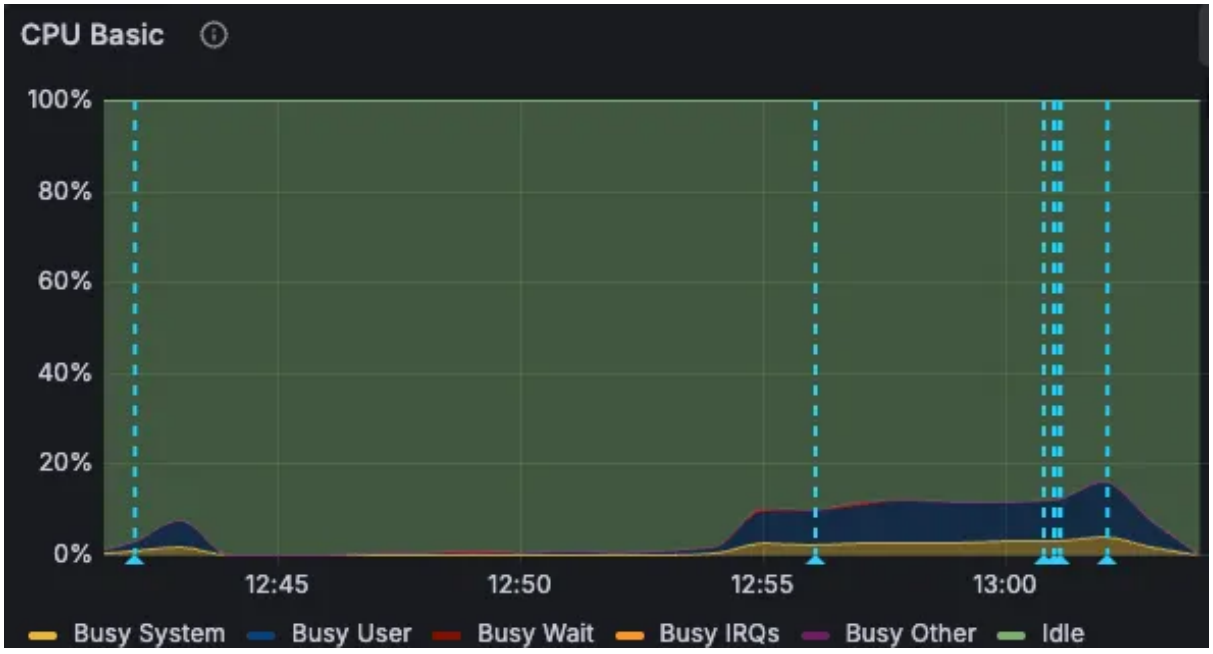
Network

5 000 Sites

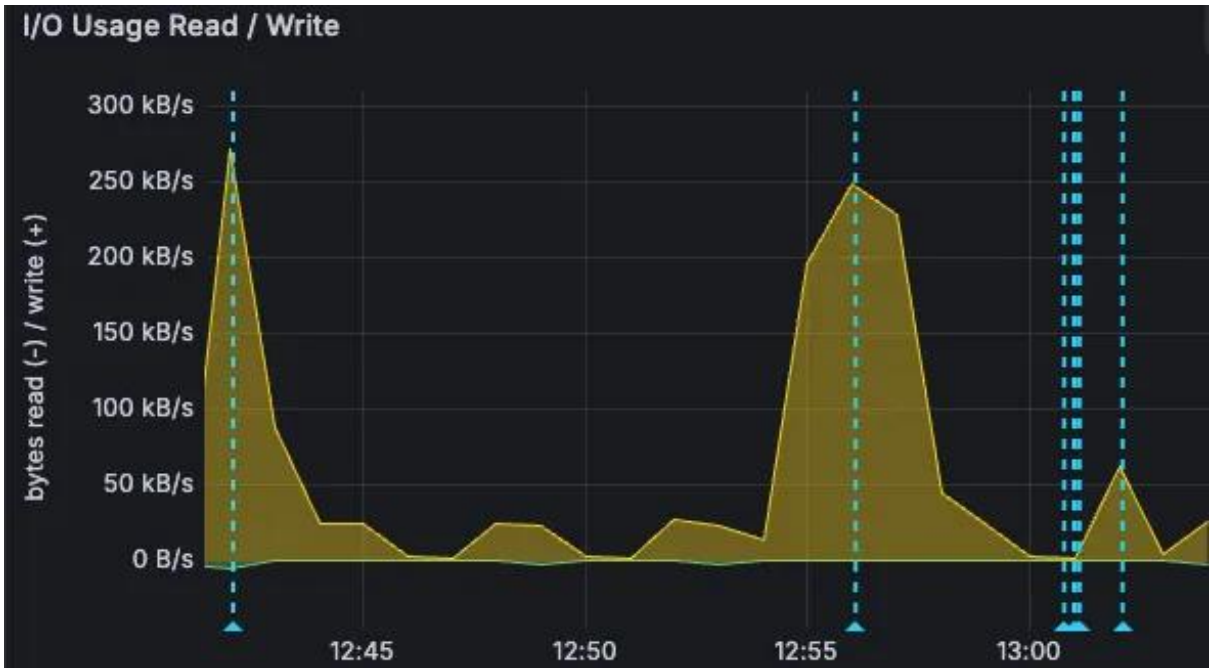
Memory



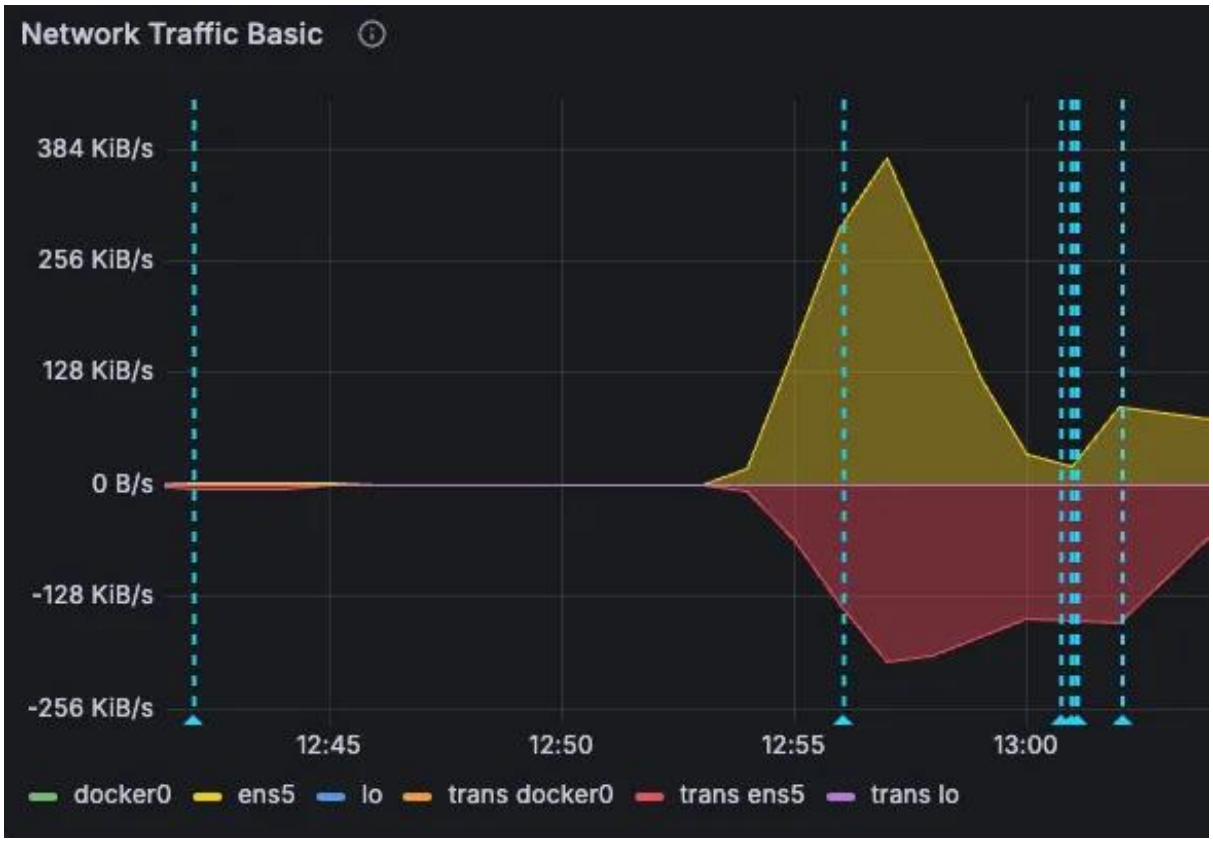
CPU



Disk IO

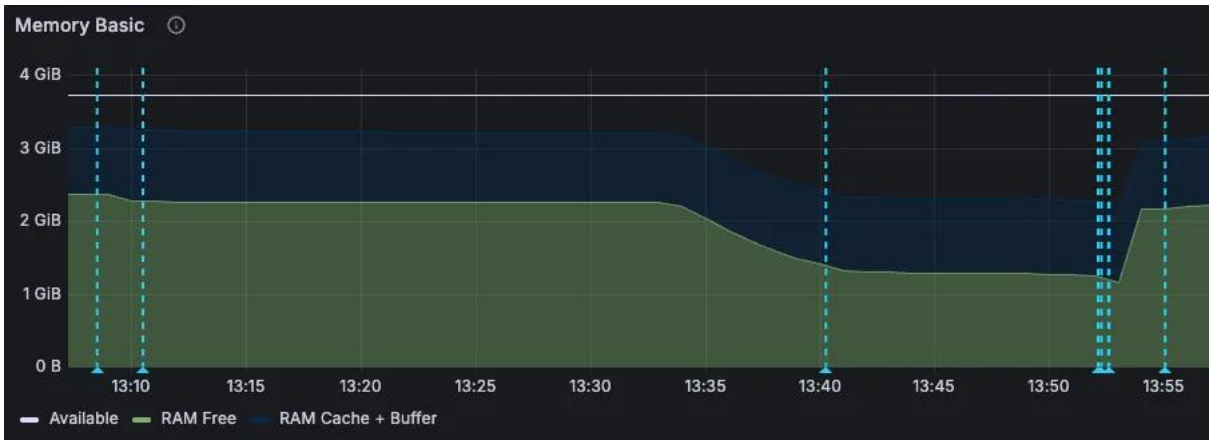


Network

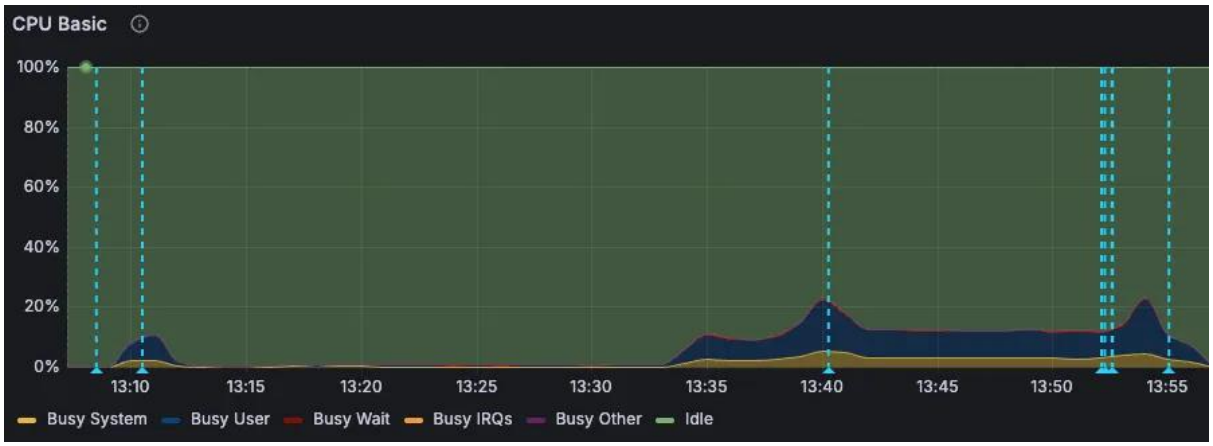


10 000 Sites

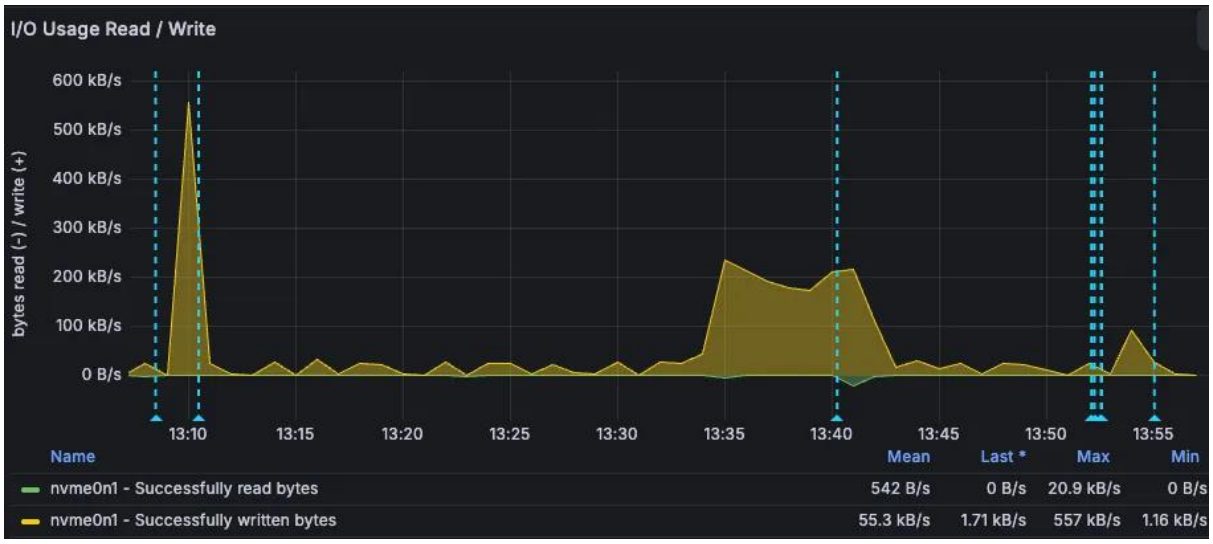
Memory



CPU



Disk IO



Network