osie

Benchmarking OpenStack Horizon vs OSIE

MEDIAN RESPONSE TIME

34% FASTER

Osie outperformed OpenStack Horizon under identical test conditions

SAME TESTING CONDITIONS

Same instances (1). Same load (100 users).
Same tool (Locust).

The only variable was the dashboard.

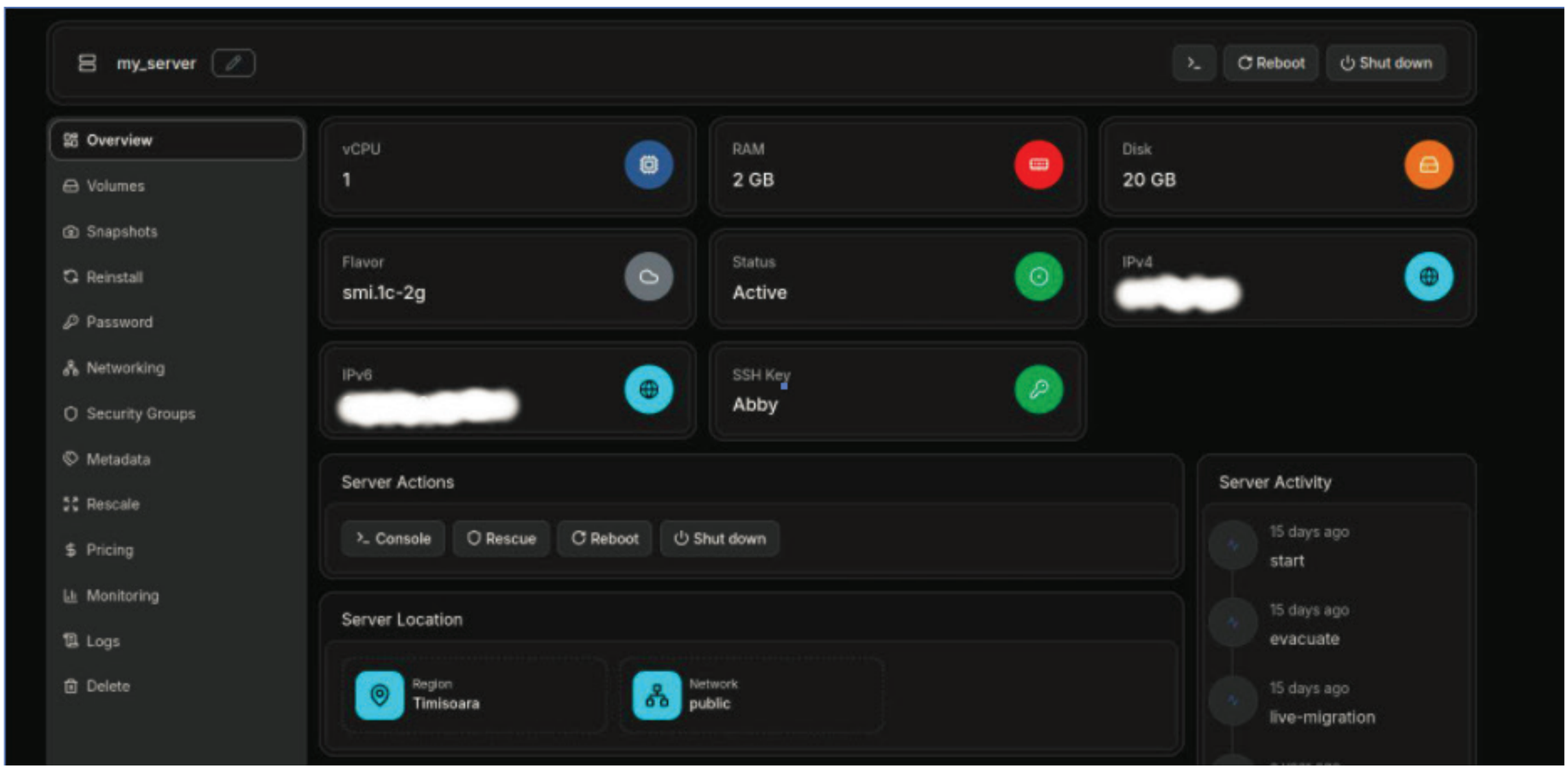
Instance Comparison

	Horizon	Osie
vCPUs	1	1
RAM	512MB	2GB
Disk	4GB	20GB
Status	Active	Active

Key Insight: A dashboard's speed isn't determined by the instance's RAM or CPU. It is determined by how the dashboard queries the OpenStack API, sequentially or in parallel

Osie Test Environment

my_server: 1 vCPU, 2GB RAM, 20GB disk
Running on Osie, Region Timisoara
Active. Ready for load testing.



Horizon Test Environment

sunbeam: 1 vCPU, 512MB RAM, 4GB disk (Active)
Running on Canonical OpenStack (Sunbeam deployment).

Specs

Flavor Name	m1.tiny
Flavor ID	
RAM	512MB
VCPUs	1 VCPU
Disk	4GB

Project / Compute / Instances

Instances

Instance ID = | Filter | Launch Instance | Delete Instances | More Actions

Displaying 1 item

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
sunbeam	ubuntu		m1.tiny	-	Active	nova	None	Running	5 days, 4 hours	Create Snapshot

Displaying 1 item

The Results

1. Osie was 34% faster at median response time.
2. Osie was 42% faster at the 95th percentile, meaning 95% of users got responses significantly faster.
3. The 99th percentile favours Horizon slightly, OSIE has occasional outlier spikes, likely cache misses or token refresh.
4. For 95% of requests, it was 42% faster than Horizon.
5. Both handled the same load with zero failures.

0% failures on both. The gap only widens from here.

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	**95%ile (ms)**	99%ile (ms)	100%ile (ms)
GET	/auth/login/	410	410	420	430	460	**500**	630	630
POST	/auth/login/	230	240	240	250	270	**300**	490	490
GET	/project/instances/	410	420	420	430	450	**480**	590	780
AGGREGATED		**410**	**420**	**420**	**430**	**450**	**480**	**590**	**780**

CPU Utilization

CPU utilization on the OpenStack controller during the 100-user Horizon load test. The server stayed consistently low throughout. Horizon's response times aren't a hardware problem, they're an architecture problem.

OPENSTACK CONTROLLER CPU (100-USER HORIZON TEST)

CONSISTENTLY LOW

Verified via Prometheus monitoring

- **Horizon: Sequential API Calls**

instances → images → volumes → networks → instances

Each call waits for the previous to finish before starting the next

- **Osie: Parallel API Calls**

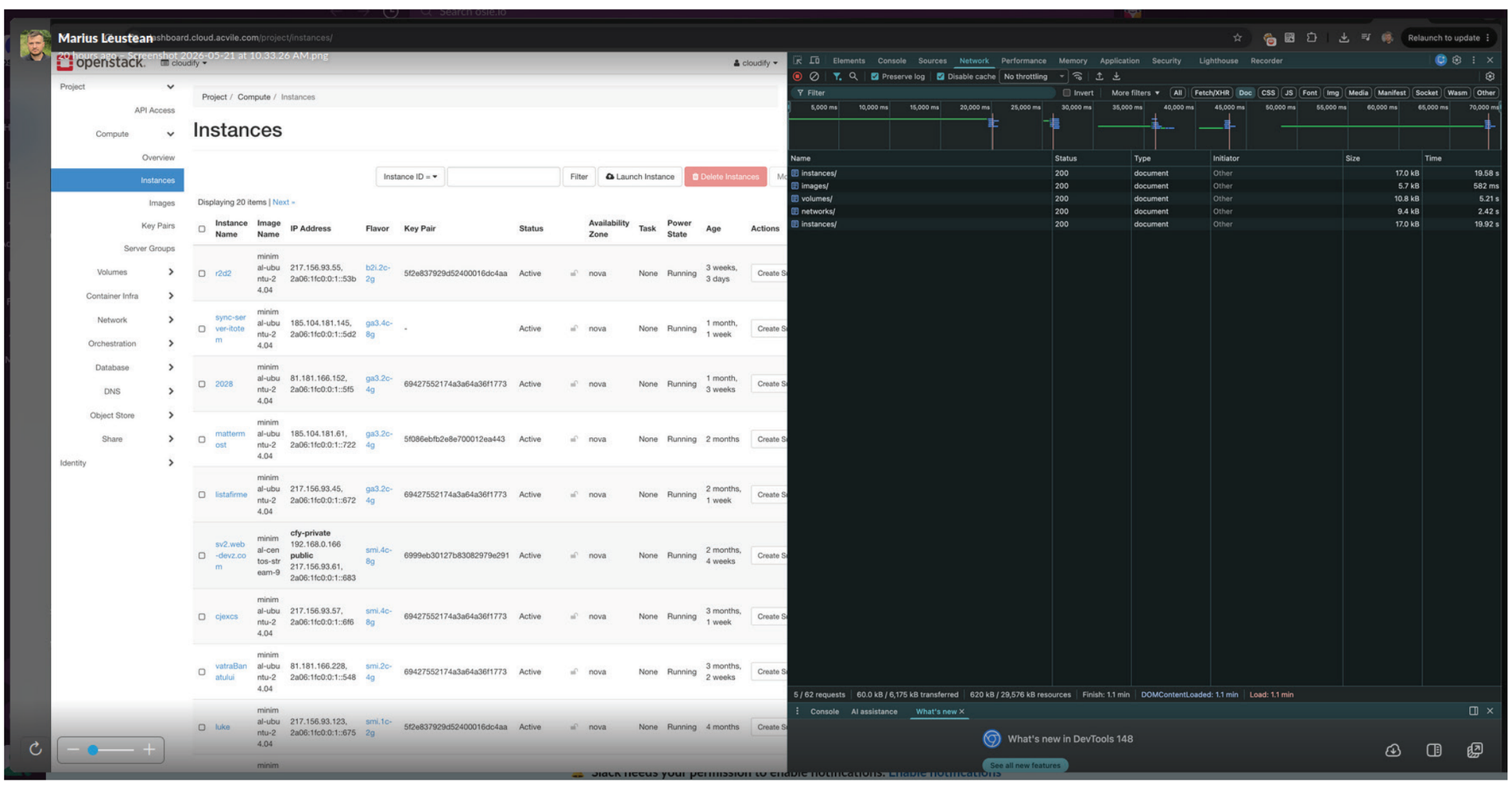
instances images volumes networks

Everything is fetched simultaneously, no waiting

Horizon at Scale (Production)

This is what happens when Horizon meets real scale. 40 instances.
10-20+ networks, volumes, security groups, and images.

The Instances page took 19.58 seconds to load. Total page load: 70 seconds.
Not because the server was slow, because Horizon makes sequential API calls.
One call to instances. Then images. Then volumes. Then networks. Then instances again. Each waiting for the previous to finish.



Osie at Scale (Production)

Same OpenStack backend. Same Timisoara region.

Different dashboard. 14 servers. Parallel API calls.
DOMContentLoaded: 845ms.
.
OSIE didn't wait for one call to finish before starting the next.
It fetched everything simultaneously.

THE BOTTOM LINE

SINGLE INSTANCE BENCHMARK

Osie vs Horizon.
Median Response Time

34%
Faster

95TH PERCENTILE (100 USERS)

95% of all requests
280ms vs 480ms

42%
Faster

AT PRODUCTION SCALE

4.91s total finish vs
70s total page load

14x
Faster

FAILURE RATE ON BOTH PLATFORMS

100 concurrent users.
Full load test

0%
failures

That’s not hardware. That’s design

Sequential vs Parallel API calls, that is an architecture choice, not a hardware limit