

Ume LLM Gateway compared with eneo

Both platforms ran on the same hardware against the same simulated AI model, which answers instantly, and were called the way an external application calls them: each with its own API key. Everything measured is the platforms' own work.

On the same hardware the gateway handles about **350 times as many messages**, adds **43 times less delay** to each one and keeps working when overloaded. It also **checks every message for personal data** and **switches to another AI provider** when one fails; eneo does neither.

● Ume LLM Gateway ● eneo v2.2.3

Four numbers that sum it up

350×

more messages per second

17 772 vs 51 per second,
both on 4 vCPUs

43×

less delay added per message

1.6 ms vs 68 ms, one user

370×

less computing per message

0.21 ms vs 78 ms CPU time

0 %

errors with 256 users at once

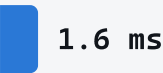
eneo: 53 % of requests
failed

ONE USER

Delay the platform adds to each answer

Measured with a single user. Both are small next to the AI model's own answer time, which is usually one to ten seconds. The difference shows up as soon as many people use the service at once and in how much hardware it needs.

Ume gateway

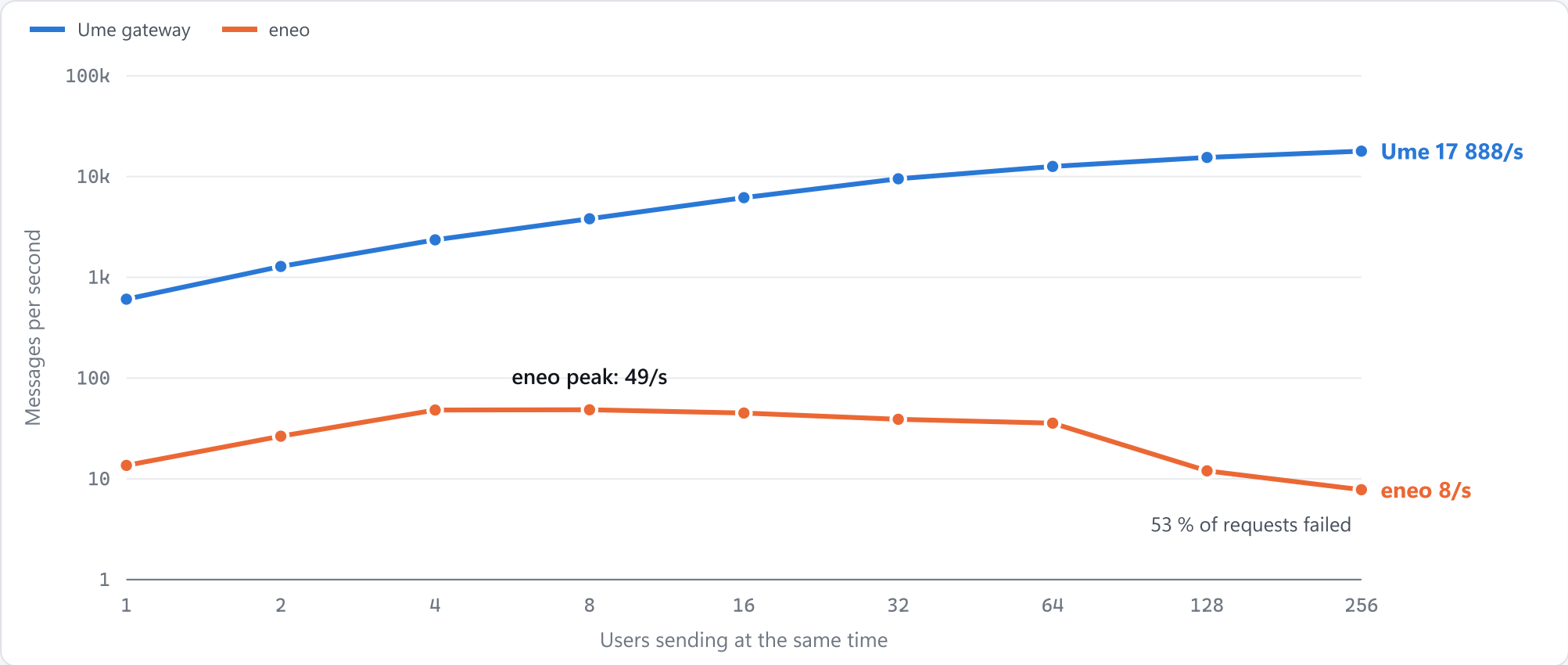


eneo



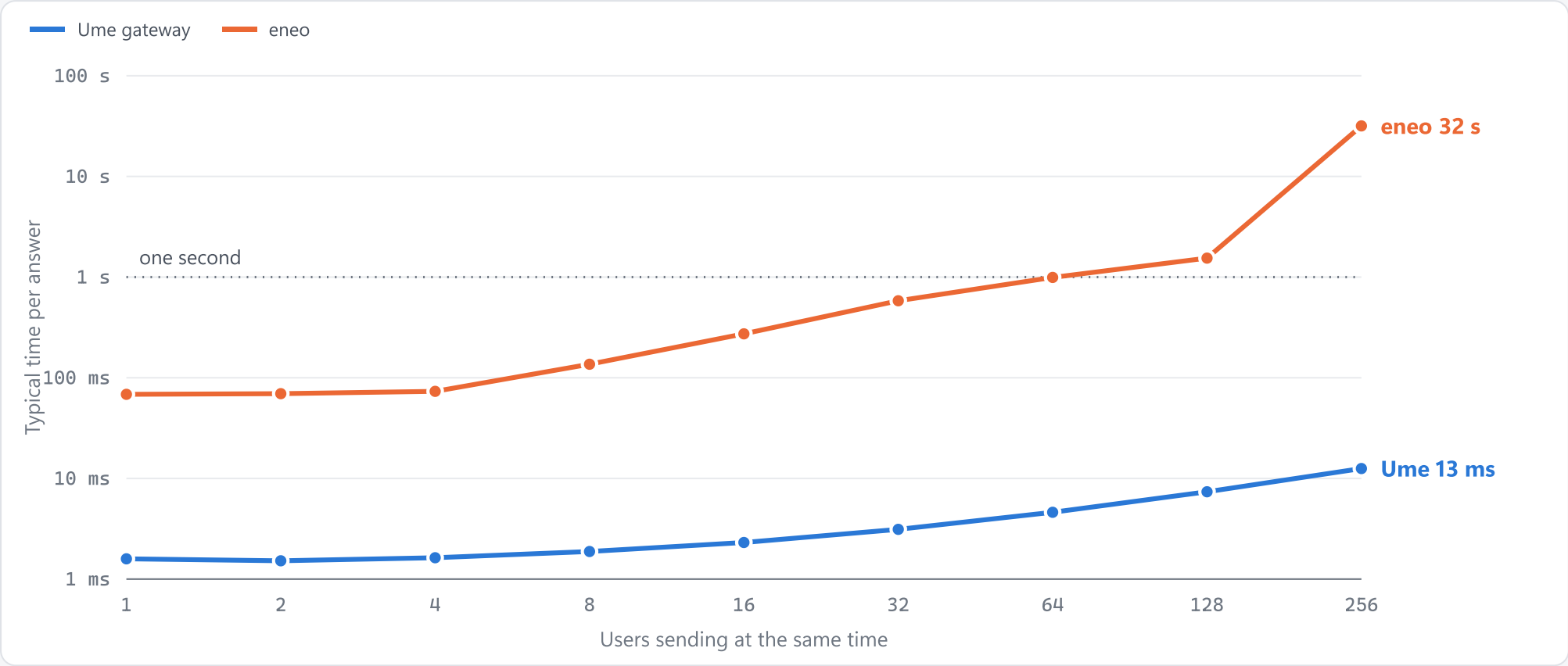
eneo peaks at 49 messages per second; the gateway keeps climbing to 17 900

Messages handled per second as more users send at the same time, both on 4 vCPUs. Note the scale: each step up is ten times more.
Past eight users, eneo gets slower in total instead of levelling off.



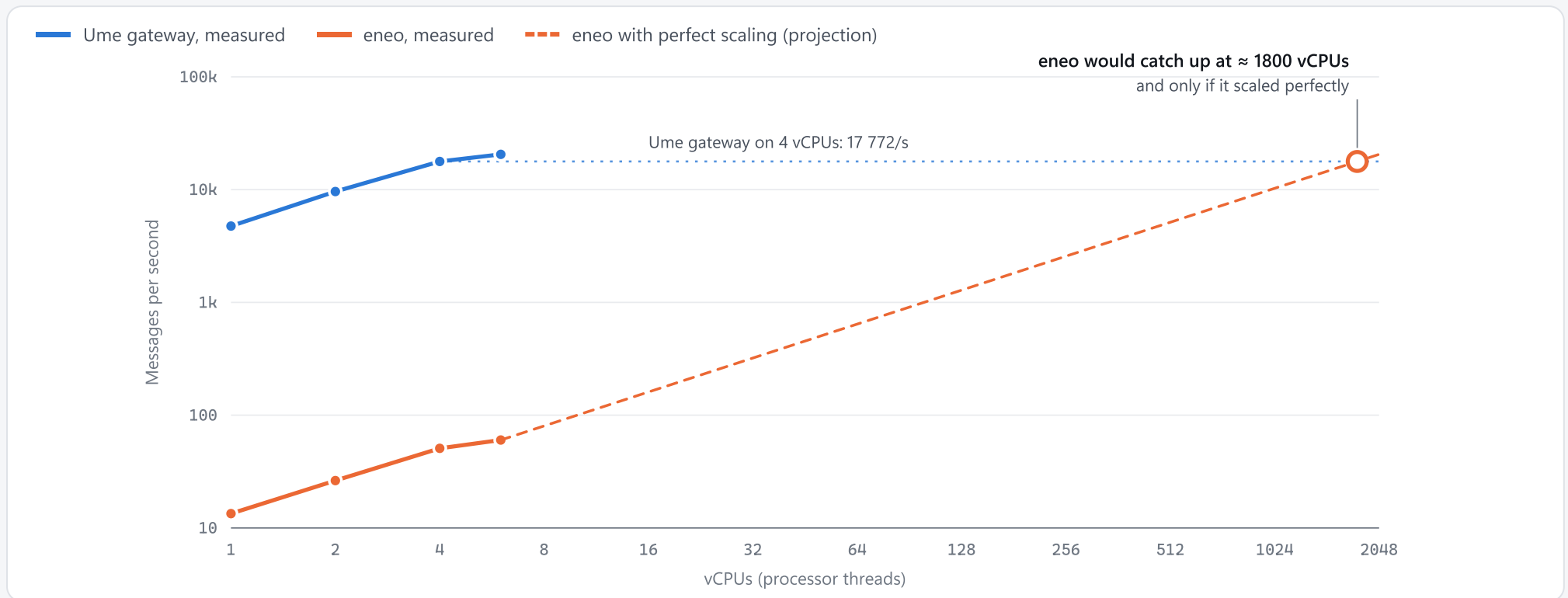
With 256 users the gateway still answers in 13 ms; eneo needs 32 seconds

Typical (median) time for the platform to deliver a whole answer, again on a ten-times scale. The dotted line marks one second.



eneo scales well, but from so far behind that it never catches up

Measured with 1 to 6 vCPUs. eneo gains almost proportionally from each extra vCPU (4.5 times faster on 6), but each vCPU only gives it about 10–13 messages per second. The dashed line continues that trend as if eneo scaled perfectly forever, which is better than it does in practice.



≈ 1800

vCPUs eneo would need

to match the gateway on 4 vCPUs, even with perfect scaling

$350\times$

more work per vCPU

4 751 vs 13 messages/s on a single vCPU

COST PER MESSAGE

Less hardware and far less database work per message

CPU time and database statements needed for one message at full load on 4 vCPUs (for eneo including its background worker, which writes an audit entry for every API-key request; the gateway writes its usage records in batches of about ten). Less of both means smaller servers, a smaller database and lower running costs for the same number of users.

CPU time per message

Ume gateway



eneo



Database statements per message

Ume gateway



eneo



The gateway slows down gracefully; eneo starts failing

Ume gateway, 256 users

Typical wait **13 ms**

Failed requests **0 %**

✓ **Every request answered**

When it is busy, requests wait a little longer in line. Nothing is lost.

eneo, 256 users

Typical wait **32 s**

Failed requests **53 %**

✗ **Errors after 30 s waits**

Each request needs several database connections at once. Under load they run out, and requests fail with an error after waiting 30 seconds.

The gateway stops personal data before it reaches the AI model; eneo passes it on

We sent the same message to both, with a personal identity number (Skatteverket's public test number), an e-mail address and a phone number. This is exactly what the AI model received.

eneo v2.2.1 sent unchanged, and saved in its chat history

Kan du hjälpa mig med ett ärende för **19121212-1212**, epost **anna.andersson@example.com**, tel **070-123 45 67**?

Ume gateway policy "redact": the values are replaced before sending; nothing of the message is stored

Kan du hjälpa mig med ett ärende för **[PERSONNUMMER]**, epost **[E-POST]**, tel **[TELEFON]**?

Redact

Replace the values, as above, and send the rest.

Block

Refuse the message and tell the user why.

Keep on-premises

Send it only to an AI model in our own data centre.

Log only

Send it, but record which kinds of data were found.

The policy is chosen per key. Detection looks for identity and co-ordination numbers (with checksum), e-mail addresses, phone numbers and bank account numbers (IBAN); it does not recognise names or free-text health information.

What protects users who send data to the wrong place

Safeguard	Ume gateway	eneo v2.2.1
Checks each message for personal data	✓ Yes identity and co-ordination numbers, e-mail, phone, IBAN	✗ No the content is not inspected tested
Acts on what it finds	✓ Yes block, redact or keep on-premises, per key	✗ No
Limits which AI models a user can reach	✓ Yes per key: allowed models, providers and data location (own data centre, EU, external)	🔴 Partly per space: an administrator sets a security level and only models with at least that level can be chosen
Switches provider when one fails	✓ Yes tries the next provider in a set order, never outside the allowed data locations; failing providers are tried last	✗ No the user gets “AI service is temporarily unavailable” tested
Stores what users write	✓ Never only usage data: tokens, cost, kinds of personal data found	🔴 Always every question and answer is kept as chat history (by design)

eneo's security levels decide which models a space may use, set in advance by an administrator. They cannot react to what a user actually types into a space.

NOT EITHER OR

eneo can run on top of the gateway and get these safeguards without changes

eneo talks to AI models through a standard interface, which the gateway provides. We pointed eneo at the gateway and tested it.

Personal data

✓ Redacted before the AI model

The same message sent through eneo reached the model as “[PERSONNUMMER], [E-POST], [TELEFON]”.

Provider failure

✓ Answered by the backup provider

With the main AI provider unreachable, eneo's user still got an answer; the gateway switched automatically.

eneo still saves the original message in its own chat history. The gateway protects what is sent to the AI model, not eneo's database.

Different kinds of product, measured on the same task

What eneo does extra

eneo is a full chat application: it stores every conversation, resolves assistants, spaces and permissions, and counts tokens for each message. The gateway passes requests on to the AI model and records usage, which is its whole job. Both checked an API key, a rate limit and recorded the use of the key for every message.

How it was measured

Same machine (8-core AMD Ryzen 7 9800X3D), same simulated AI model, each platform given the same processor cores; eneo v2.2.3 (its latest release) built from the official release tag, with its background worker running as in its own production set-up. Both were called like an external application: the gateway with a virtual key, eneo with a service API key scoped to one assistant (a new conversation per message, streamed). Measured 27 October 2026; each figure is 20 seconds of steady load, and every answer was checked. Scripts: [benchmarks/compare](#).

What it means in practice

For one user the difference is hidden behind the AI model's own answer time. For many users it decides how much hardware is needed and whether the service stays up at peak load.

The gateway's own limit

One gateway instance handled about 18 000 messages per second on 4 vCPUs and 20 500 on 6. That is far above expected load, and more instances can be added.