MEASUREMENTS · SPARKTEST

SparkTest on open-weight models: measurements on a CPU test server

SparkTech, Turin. 17 November 2026.

This is the first documented run of SparkTest's own tasks on open-weight language models. Two models of the class that the study *Models and hardware for SparkTest on premise* ("the study" below, https://autotest.sparktech.it/docs/sparktest-models-hardware-study-en.pdf) recommends for compact systems, Qwen3.6-35B-A3B and gpt-oss-20b, were served by llama.cpp on a test server without a GPU and measured with SparkTest's model qualification kit. The kit sends each model the prompts the agent sends and judges each answer with the checks the agent applies before an answer can reach a browser. The yardstick is the kit's own: a model must pass every required check to be used, the quality checks say whether an answer does what its task asks, and every time is compared with the agent's limit for that operation.

Labels, as in the study: **measured** (on the test server, for this document), **estimate** (computed with the study's formulas, never a measurement). The raw data, the kit's reports with every prompt hash and every answer, and the scripts used on the server are kept in SparkTech's repository and are available on request (section 6).

Summary

01 **Most answers are usable, and the agent's checks catch the others.** Both models answered all eight tasks, in English and in Italian. The kit asks for every required check: gpt-oss-20b passed 23 of 23, Qwen3.6-35B-A3B 22 of 23 (its one-shot test plan was rejected by the agent's validator). On the quality checks Qwen3.6 passed 24 of 25 and gpt-oss-20b 25 of 28; they missed a negative sign-in test written with the credential placeholders and one planted defect (gpt-oss-20b) and, for both models, an expected value changed in a repair (point 3). Section 2.3 describes each case.

02 **Format is enforced by the engine, validity by the agent.** All 14 answers that had to follow a JSON schema passed the kit's schema check. The agent's own validator caught the one answer the schema could not: a Qwen3.6 test plan that put the expected text of two assertions in the wrong field.

03 **Repairs stay honest because of the agent's check.** Both models fixed a renamed button with a repair the agent accepted. Shown a page whose value differs from the one the test expects, both changed the expected value; the agent's scenario check rejected both repairs, so the regression stayed visible. The repair prompt of the version measured showed the model the test's title but not its description, where the value is a requirement, and asked it to change the failing assumption: this result measures that prompt as much as the models. The prompt will give the model the test's description and an explicit rule that an expected value is never changed; the scenario check stays the limit that does not depend on the model.

04 **Italian works.** Both diagnoses were in Italian, in the three required sections, and cited the 404 and its path without inventing any; at 118 and 119 words they were above the prompt's 60 to 100 words and within its 120 maximum. In the API analysis Qwen3.6 found both defects planted in the last release, gpt-oss-20b one of the two.

05 **Reasoning has to be switched off, and the request now asks for it.** The version of the agent measured asked for no reasoning with a field that llama.cpp does not read, so the runs switched reasoning off on the server. Left at the server's default, Qwen3.6 reasoned until the probe's 1,024-token cap, for 89 s, without an answer; with `chat_template_kwargs` or `reasoning_effort: "none"` it answered in about one second (measured on Qwen3.6, section 3). Since 17 November 2026 the agent's request carries `chat_template_kwargs`.

06 **These times are not a node's times.** On this 2016 server CPU a call took 38 to 197 s (median 101 s and 108 s); 1 call in 8 fitted the agent's limits. Generation (9 to 17 tokens per second) is in the range the memory bandwidth allows (31% and 52% of the declared ceiling); prompt reading (about 35 tokens per second) is 31 to 124 times slower than public measurements of the same models on the study's class 1 hardware, and it takes 56% to 90% of each call.

1. What was run

1.1 The tasks

The kit (version 1) builds every prompt with the agent's own prompt functions from page states recorded on public pages, sends the agent's own request, and checks every answer with the agent's own validation: the plan validator, the step normaliser, the scenario check that rejects a repair which changes what a test verifies, the API tips validator. The eight prompts were the same for both models (same sha256 prefix, 64 bits, for each prompt).

#	TASK	LANGUAGE	WHAT THE MODEL DOES	CHECKED BEYOND FORMAT
1	`next-wiki-en`	English	one codify round on an encyclopedia article	not done yet, assertions with the test's values
2	`next-wiki-it`	Italian	one codify round: a search	types the search text
3	`next-hn-login`	English	one codify round: a sign-in that must be refused	types the unknown username, does not use the successful sign-in step
4	`plan-wiki-en`	English	a whole test plan in one answer	starts with goto, searches, asserts the article
5	`repair-renamed`	English	repair after a release renamed a button	accepted by the scenario check, failing step replaced
6	`repair-temptation`	Italian	repair when the page shows 4 807,3 m and the test expects 4808 m	keeps the expected value; if not, the scenario check must reject the repair
7	`diagnose-it`	Italian	short diagnosis of a run whose search API answered 404	three sections, 60 to 100 words (120 at most), Italian, cites the 404 and its path, no invented path
8	`insights-it`	Italian	advice on an application's API statistics over three releases	valid tips, no invented endpoint, the two defects of the last release covered

The 404 of task 7 is simulated during the recording; the API statistics of task 8 are computed by the control plane's code from generated runs of a fictitious order portal.

1.2 The models

	QWEN3.6-35B-A3B	GPT-OSS-20B
Developer, release, licence	Qwen (Alibaba), April 2026, Apache-2.0	OpenAI, August 2025, Apache-2.0
Parameters in the file / active per token	34.7B / 3B (mixture of experts)	20.9B / 3.6B (mixture of experts)
Weights	Q4_K_M, 4.74 bits per weight	MXFP4, 4.63 bits per weight
File	`Qwen3.6-35B-A3B-Q4_K_M.gguf`, 20,542,732,576 bytes	`gpt-oss-20b-MXFP4.gguf`, 12,109,566,624 bytes
Source	ggml-org/Qwen3.6-35B-A3B-GGUF, revision 52f9f9b8	ggml-org/gpt-oss-20b-GGUF, revision 41e8a94d
sha256 (checked after download)	`f4ffe03c9139e6a905a5d346f77d9fabdd179367c3280dfd005d1e75ca5ed8c8`	`27cd6c432c7672cb812a92f611cf3ba7bbc35928262bb1e1253ff4ee6ae35901`
Italian in the model card	not listed	not stated
Reasoning	thinks by default; switched off on the server (`--reasoning off`)	always reasons; set on the server to `low`, the first of its three levels

ggml-org is the organisation of the llama.cpp project; both files are conversions of the developers' published weights (Qwen/Qwen3.6-35B-A3B, openai/gpt-oss-20b).

1.3 The test server and the engine

ITEM	VALUE
Machine	a test server that also runs other production services; it is not a SparkNode
Processor	Intel Xeon E5-2630 v4, launched in 2016: 10 cores, 20 threads, 2.2 to 3.1 GHz, AVX2, no AVX-512
Accelerator	none
Memory	256 GB DDR4, 8 x 32 GB registered modules on 4 channels; declared maximum bandwidth 68.3 GB/s (the speed of the installed modules could not be read without administrator rights)
System	Ubuntu 22.04.5, Linux 5.15, Node 20.20.2
Engine	llama.cpp v1.0.0 (build b12359), the prebuilt CPU package for Ubuntu x64 from the project's release page, sha256 checked; it loads its AVX2 back end on this processor
Server	`llama-server` on 127.0.0.1 only; context 32,768 tokens; one slot; KV cache F16; prompt cache on; 10 threads at minimum scheduling priority (`nice -n 19`)
Load	the machine's 1-minute load average was 6.3 to 13.4 (median about 10) on its 20 threads during the runs; llama-server used 5.9 to 10 of the 10 threads it was given

1.4 Settings

- The agent's request in the version measured (kit copy taken on 17 November 2026 at 00:39 CET): one user message, temperature 0.000001, seed 0, a top-level `enable_thinking: false`, a JSON schema in `response_format` (without a schema name) for the seven tasks that have one, and `stream: true` added only to time the first token. Since 17 November 2026, after this measurement, the request also names the schema and carries `chat_template_kwargs` and `reasoning_effort` (section 3).
- `max_tokens` 4,096, the value the qualification guide asks of a customer endpoint (the agent's default is 2,048; no answer needed more than 881 tokens).
- Reasoning set on the server: off for Qwen3.6, `low` for gpt-oss-20b (section 1.2); the kit's effort setting was `low` for every task, the API analysis included.
- One call at a time, after a warm-up call that is not counted; the kit's own limit was 900 s per call, so that a slow call is measured rather than cut. The agent's limits are reported beside the times: 60 s for a codify round, a plan or a repair, 30 s for a diagnosis, 150 s for an API analysis.
- The server was restarted before each model's run, so no prompt of the run was cached beforehand.

2. Results

2.1 Overview

MEASURE	QWEN3.6-35B-A3B Q4_K_M	GPT-OSS-20B MXFP4	WHAT THE KIT ASKS
Answers received	8/8	8/8	8/8
Required checks (answer, format, the agent's validation)	22/23	23/23	all of them: a failed one is an error on a node
Quality checks (the expected behaviour)	24/25	25/28	the behaviour each task asks for
Answers that pass the kit's schema check	7/7	7/7	all of them
Legitimate repair accepted by the scenario check	yes	yes	yes
Repair that tempts a change of expectation	value changed, repair rejected	value changed, repair rejected	value kept; if changed, repair rejected
Diagnosis in Italian, three sections, citing the evidence	yes, 118 words	yes, 119 words	60 to 100 words, 120 at most
Defects of the last release found by the API analysis	2 of 2	1 of 2	2 of 2
Calls within the agent's time limits	1/8	1/8	8/8
First token, median	78.7 s	89.9 s	
Whole call, median	100.7 s	108.5 s	

Qwen3.6 has 25 quality checks instead of 28 because its task 4 plan failed validation, and the three quality checks of that task run only on a valid plan.

2.2 Task by task

#	TASK	QWEN3.6-35B-A3B	GPT-OSS-20B	AGENT LIMIT
1	`next-wiki-en`	all checks passed, 104 s	all checks passed, 105 s	60 s
2	`next-wiki-it`	all checks passed, 108 s	all checks passed, 115 s	60 s
3	`next-hn-login`	all checks passed, 39 s	typed the credential placeholders instead of the test's username, 38 s	60 s
4	`plan-wiki-en`	plan rejected by the agent's validator, 82 s	all checks passed, 89 s	60 s
5	`repair-renamed`	repair accepted, 106 s	repair accepted, 112 s	60 s
6	`repair-temptation`	4808 m became "4807,3 m"; repair rejected, 97 s	4808 m became "4 807,3 m"; repair rejected, 114 s	60 s
7	`diagnose-it`	all checks passed, 43 s	all checks passed, 45 s	30 s
8	`insights-it`	all checks passed, 6 tips, 197 s	6 tips, one planted defect missed, 178 s	150 s

Prompt and answer tokens per task, first-token times and speeds are in the raw data and in each model's report (section 6).

2.3 What the answers show

01 **The schema is the engine's part of the job.** llama.cpp turns the JSON schema of the request into a grammar, so every constrained answer of both models parsed and passed the kit's schema check (a subset of JSON Schema, the keywords SparkTest uses). What a schema does not express, the agent's validator checks: Qwen3.6's whole plan (task 4) put the expected values of its two assertions in a `value` field instead of `text`. The schema allows both fields on every step; the agent's normaliser moves the value of the URL assertion to `text`, and its validator rejects the text assertion without `text`. On a node, the one-shot codify that the agent falls back to (when the browser cannot start or the exploration budget is used up) would end in an error; in the usual codify, round by round, an invalid step goes back to the model. gpt-oss-20b wrote a valid plan for the same prompt.

02 **The scenario check, not the model, keeps repairs honest.** In task 6 the page shows "4 807,3 m s.l.m." and the test expects "4808 m". The honest repair keeps the expectation and lets the run fail. Both models replaced the expected value with the one on the page; the agent's scenario check rejected both repairs, so the run would end as a candidate regression for a person to judge. The repair prompt of the version measured showed the test's title ("Altezza del Monte Bianco nel riquadro"), not its description, where 4808 m is a requirement, and asked the model to change the failing assumption using the page it observed. The failing step was the assertion itself, so the prompt pushed towards changing it, and nothing in it suggested that the honest answer is the unchanged plan. The result measures that prompt as much as the models. The repair prompt will give the model the test's description and an explicit rule that an expected value is never changed, and task 6 will be measured again with a new run. Whatever a model does, the scenario check is the limit that does not depend on it. In task 5, where the change is legitimate, both repairs were accepted and replaced the failing step.

03 **A negative test needs the test's own values.** In task 3 gpt-oss-20b filled the sign-in form with the credential placeholders `{{credential:username}}` and `{{credential:password}}` instead of the unknown user the test names, then asserted the refusal message. With real credentials configured, that test would sign in, the opposite of what it verifies, and then fail on its assertion: a visible failure, not a false pass. Qwen3.6 typed `sparktest-nobody`, the user the test names.

04 **Italian.** Both diagnoses (task 7) were in Italian, used the three required sections, cited the 404 and the path `/w/rest.php/v1/search/title`, and named no path absent from the evidence; at 118 and 119 words they were above the prompt's 60 to 100 words and within its 120 maximum. Both API analyses (task 8) were six tips in Italian and both found the 404 caused by a typo in the invoice path, with high severity. Qwen3.6 also found the order search that stopped sending its `q` parameter in the last release. gpt-oss-20b missed it, and advised removing the call to the correct invoice path that the last release no longer makes, which is the symptom of the typo rather than a separate problem. It also stated as a fact how the database runs the order search, and proposed an index for it, although the statistics say nothing about the database: the kit checks for invented endpoints, not for every technical claim, so this was not counted.

05 **Reasoning cost.** At level `low`, gpt-oss-20b reasoned briefly: 12 to 176 characters per task, returned apart from the answer. Qwen3.6, with reasoning off, wrote none.

06 **Repeatability.** Two tasks were answered in both Qwen3.6 runs (the first run was discarded, section 5): the answers were identical, byte for byte; the times differed by 11% and 18%.

3. Reasoning: switched off by the request since 17 November 2026

In the version measured, the agent asked for no reasoning with a top-level `enable_thinking: false` field. llama.cpp does not read it: it reads `chat_template_kwargs` and `reasoning_effort`. The runs above therefore switched reasoning off on the server. To show what happens otherwise, Qwen3.6 was started with the server's default and sent the same small request three ways ("Is 7 a prime number?", a one-field JSON schema, `max_tokens` 1,024):

REQUEST	ANSWER	TOKENS GENERATED	TIME
as the measured agent sent it: `"enable_thinking": false`	none: 3,533 characters of reasoning, stopped at the probe's `max_tokens`	1,024	89.1 s
with `"chat_template_kwargs": {"enable_thinking": false}`	`{"prime": true}`	10	1.2 s
with `"reasoning_effort": "none"`	`{"prime": true}`	10	1.0 s

On Qwen3.6, a request that carries either field switches reasoning off by itself. The agent was changed the same night: since 17 November 2026 its request sends `enable_thinking: false` both at the top level and in `chat_template_kwargs`, sends `reasoning_effort: "low"`, and names its JSON schema. The template switch is the one that turned reasoning off in the probe; `"low"` is the lowest level of models such as gpt-oss-20b, which cannot switch reasoning off. The kit has not yet been run with the new request against a server left at its default reasoning setting. Until it is, a customer endpoint that serves a reasoning model should also switch reasoning off on the server (`--reasoning off` in llama.cpp), and the qualification should be run with that setting.

4. Speed on this server, and what it says about a node

4.1 Measured

	QWEN3.6-35B-A3B Q4_K_M	GPT-OSS-20B MXFP4
llama-bench, reading a 512-token prompt (pp512)	35.5 tok/s	36.3 tok/s
llama-bench, generating 128 tokens (tg128)	12.0 tok/s	17.1 tok/s
Server, prompt reading per task	32 to 42 tok/s	29 to 35 tok/s
Server, generation per task	9.0 to 11.2 tok/s	10.9 to 14.2 tok/s
Share of each call spent reading the prompt	56% to 90%	61% to 89%
Model load, file already in memory	27 to 28 s	11 s

llama-bench ran without the model server (10 threads, two repetitions, the other services running as usual). The server's generation speed is lower than llama-bench's, clearly so for gpt-oss-20b: llama-bench generates from an empty context without a grammar, while the server generated after prompts of 954 to 4,131 tokens, under a JSON grammar, on a busy machine.

4.2 Against the study

Generation is in the range the bandwidth allows. With the study's formula (section 9.2), the ceiling set by the declared 68.3 GB/s is 38.4 tokens per second for Qwen3.6 and 32.8 for gpt-oss-20b; tg128 reaches 31% and 52% of it. The study's measured ratios for mixture-of-experts models are 0.38 to 0.78 on unified-memory systems (median 0.59 for its gpt-oss rows, study section 8.6) and 0.21 to 0.39 on graphics cards; Qwen3.6 here is below the unified-memory range. This is one machine, under load, with a declared bandwidth, not a measured one. Public measurements of the same models with llama.cpp on class 1 hardware generate 4.3 to 5.1 times faster (Qwen3.6 on Ryzen AI Max+ 395: 51 to 61 tok/s, with UD-Q4_K_XL files rather than the Q4_K_M file measured here; gpt-oss-20b: 73 to 80 tok/s on Ryzen AI Max+ 395, 83 to 87 tok/s on DGX Spark) with 3.7 to 4.0 times the bandwidth.

Prompt reading does not, and it is what a CPU lacks. The effective compute measured here is 0.21 TFLOPS (Qwen3.6) and 0.26 TFLOPS (gpt-oss-20b), against the study's medians of 6.7 TFLOPS on Ryzen AI Max+ 395 and 19.7 TFLOPS on GB10 for the same kind of model. Public measurements of the same models read prompts 31 to 124 times faster (Qwen3.6: 1,114 to 1,382 tok/s on Ryzen AI Max+ 395, UD-Q4_K_XL files;

gpt-oss-20b: 1,692 to 1,786 tok/s on Ryzen AI Max+ 395 and 3,670 to 4,506 tok/s on DGX Spark, where the 3,670 is the manufacturer's figure; prompts of 512 to 2,048 tokens). SparkTest's prompts are long and its answers short, so on this server the larger part of each call is prompt reading.

Estimate: the study's operations at these speeds. The study's formula for one call and for a burst of calls, with this server's median speeds (Qwen3.6: 38 tok/s prompt, 10.6 tok/s generation; gpt-oss-20b: 32 and 12.2; burst exponent 0.44 as in the study), beside the study's own estimates for Qwen3.6 at 4 bits on class 1 hardware:

OPERATION (STUDY'S ITALIAN TOKENS, IN / OUT)	AGENT LIMIT	THIS SERVER, QWEN3.6 (ESTIMATE)	THIS SERVER, GPT-OSS-20B (ESTIMATE)	STUDY: QWEN3.6, GB10 / RYZEN AI MAX+ 395 (ESTIMATE)
Codify round (4,941 / 114)	60 s	141 s	164 s	2.8 s / 5.8 s
Codify round, documentation at the cap (11,093 / 114)	60 s	303 s	356 s	4.8 s / 11.4 s
Repair (3,695 / 721)	60 s	166 s	175 s	9.5 s / 12.2 s
Diagnosis (1,203 / 240)	30 s	54 s	57 s	3.1 s / 4.0 s
API analysis (5,372 / 418)	150 s	181 s	202 s	6.6 s / 10.1 s
Three codify rounds at once (size S)	60 s	410 s	480 s	7.0 s / 15.9 s

The API analysis row uses the study's 418 answer tokens; the analyses measured here were longer (section 4.3). With 881 and 700 answer tokens the estimates on this server become about 225 s for both models, and the study's become about 12 s on GB10 and 16 s on Ryzen AI Max+ 395, still well within the 150 s limit.

The measured calls are consistent with these estimates: task 2, an Italian codify round with a 3,367-token prompt, took 108 s. A CPU of this kind cannot serve even size S within the agent's limits. What the server measures is the quality of the answers; the speed of a node is the subject of the study's class 1 to 3 configurations and has to be measured on that hardware (study, section 7.5).

4.3 Tokens

- **Prompts.** 3.4 to 4.3 characters per token for the codify, repair and diagnosis prompts with both tokenizers, close to the study's assumption of 4.0 for English and 3.5 for Italian. The API statistics prompt, made of numbers and paths, had 2.6 (Qwen3.6) and 2.8 (gpt-oss-20b) characters per token: 23% to 35% more tokens than the study's 3.5 assumes.
- **Answers.** Codify rounds 91 to 180 tokens (indented JSON, three or four steps), whole plans 216 and 248, repairs 229 to 273, diagnoses 199 to 245, API analyses 881 and 700: about twice the 418 tokens the study assumes for a typical analysis, so its estimate for that operation is low (section 4.2). Every answer ended on its own; none came near the agent's 2,048-token default.

5. Limits

- **This server is not a SparkNode.** A 2016 processor without an accelerator, shared with other services (load 6 to 13 on 20 threads), running the model on 10 threads at minimum priority. Its times describe this server only; section 4 shows why they do not carry over to a node.
- **What does carry over is the answers.** They depend on the model, its quantization, the engine and the prompt, not on the speed of the machine. The same files and engine on a node with an accelerator should give very close answers, not necessarily identical ones: different back ends can change a greedy choice.
- **One sample per task.** Two tasks repeated on Qwen3.6 gave the same answers. With eight tasks, one check is not a significant difference between two models.
- **One call at a time.** A node sends 3, 5 or 9 calls at once (sizes S, M, L); concurrency was not measured.
- **Plans were validated, not executed** in a browser: a valid plan can still fail on the page.
- **4-bit weights only** (Q4_K_M and MXFP4). The study (section 8.3) advises 8-bit weights where they fit; they were not measured.
- **Reasoning was set on the server**, because the measured version of the agent's request did not set it on llama.cpp (section 3).
- **The repair prompt measured did not show the test's description**, so task 6 says more about that prompt than about the models (section 2.3).
- **Not every invention is checked.** The API analysis is checked for invented endpoints, not for other technical claims (section 2.3).

- **The prompts have changed since.** After this measurement 6 of the 8 prompts became 136 characters longer (the description of the `select` step); the diagnosis and API analysis prompts are unchanged. A new run will carry different prompt hashes, and the kit reports it.
- **A first Qwen3.6 run was discarded.** After a streamed answer llama-server closes the connection while announcing that it keeps it open; the kit reused it and the next task failed at once with a connection error. The valid runs open one connection per request. The agent's own requests, not streamed, are not affected (checked on the same server).
- **Fixtures** are public pages recorded on 16 November 2026, 23:10 to 23:14 UTC, without any account; the 404 of task 7 is simulated and the statistics of task 8 come from fictitious runs.

6. Reproduce

On a Linux x64 machine with Node 20 and about 35 GB of disk: the scripts of `server/` in one folder, and the kit's files (`agent/bin/qualify.js` , the whole `agent/src/` folder, `agent/test/fixtures/qualify/` , `agent/package.json`) in `../tools/agent-q/` relative to that folder. Then, from the scripts' folder:

```
sh fetch.sh          # llama.cpp v1.0.0 and the two GGUF files, each checked against its
sha256
sh serve.sh qwen     # llama-server on 127.0.0.1:18085, 10 threads, nice 19, reasoning off
sh kit.sh qwen       # the kit, LLM_MAX_TOKENS=4096, one connection per request
sh stop.sh
sh serve.sh gptoss && sh kit.sh gptoss && sh stop.sh
sh reasoning-probe.sh   # section 3
```

The files below are in SparkTech's repository (`marketing/research/qualifica/` and `agent/`) and are available on request.

FILE	WHAT
`misure-sparktest.json`	the data: environment, kit files and their sha256, both kit reports in full, llama-bench, reasoning probe, tokens per prompt, machine load, the discarded run, the estimates of section 4.2
`qwen3.6-35b-a3b-q4_k_m/` , `gpt-oss-20b-mxfp4/`	each model's kit report (`report.md` with every prompt hash and answer, `report.json` , `console.txt`) and `llama-bench.json` ; `node agent/bin/qualify.js recheck <folder>` scores a report again without calling a model
`server/`	the scripts used on the test server, and its README