MISURE · SPARKTEST

SparkTest sui modelli open-weight: misure su un server di prova senza GPU

SparkTech, Torino. 17 novembre 2026.

È la prima esecuzione documentata dei compiti di SparkTest su modelli linguistici open-weight. Due modelli della classe che lo studio *Modelli e hardware per SparkTest on premise* ("lo studio" nel seguito, https://autotest.sparktech.it/docs/sparktest-models-hardware-study-it.pdf) raccomanda per i sistemi compatti, Qwen3.6-35B-A3B e gpt-oss-20b, sono stati serviti da llama.cpp su un server di prova senza GPU e misurati con il kit di qualificazione dei modelli di SparkTest. Il kit manda a ogni modello i prompt che manda l'agente e giudica ogni risposta con i controlli che l'agente applica prima che una risposta arrivi al browser. Il metro è quello del kit: per essere usato un modello deve superare tutti i controlli obbligatori, i controlli di qualità dicono se una risposta fa ciò che il compito chiede, e ogni tempo si confronta con il limite dell'agente per quell'operazione.

Etichette, come nello studio: **misurato** (sul server di prova, per questo documento), **stima** (calcolata con le formule dello studio, mai una misura). I dati grezzi, i rapporti del kit con l'impronta di ogni prompt e ogni risposta, e gli script usati sul server sono conservati nel repository di SparkTech e sono disponibili su richiesta (sezione 6).

Sintesi

01 **La maggior parte delle risposte è utilizzabile, e i controlli dell'agente fermano le altre.** I due modelli hanno risposto a tutti gli otto compiti, in inglese e in italiano. Il kit chiede tutti i controlli obbligatori: gpt-oss-20b ne ha superati 23 su 23, Qwen3.6-35B-A3B 22 su 23 (il suo piano di test in una sola risposta è stato respinto dal validatore dell'agente). Nei controlli di qualità Qwen3.6 ne ha superati 24 su 25 e gpt-oss-20b 25 su 28; sono mancati un test di accesso negativo scritto con i segnaposto delle credenziali e un difetto piantato (gpt-oss-20b) e, per entrambi i modelli, un valore atteso cambiato in una riparazione (punto 3). La sezione 2.3 descrive ogni caso.

02 **Il formato lo impone il motore, la validità la controlla l'agente.** Tutte le 14 risposte che dovevano seguire uno schema JSON hanno superato il controllo dello schema del kit. Il validatore dell'agente ha fermato l'unica risposta che lo schema non poteva fermare: un piano di Qwen3.6 che metteva il testo atteso di due asserzioni nel campo sbagliato.

03 **Le riparazioni restano oneste grazie al controllo dell'agente.** Entrambi i modelli hanno riparato un pulsante rinominato con una riparazione accettata dall'agente. Davanti a una pagina che mostra un valore diverso da quello atteso dal test, entrambi hanno cambiato il valore atteso; il controllo dello scenario dell'agente ha respinto le due riparazioni, quindi la regressione è rimasta visibile. Il prompt di riparazione della versione misurata mostrava al modello il titolo del test ma non la sua descrizione, dove il valore è un requisito, e chiedeva di cambiare l'assunzione che fallisce: questo risultato misura quel prompt quanto i modelli. Il prompt darà al modello la descrizione del test e la regola esplicita che un valore atteso non si cambia mai; il controllo dello scenario resta il limite che non dipende dal modello.

04 **L'italiano funziona.** Le due diagnosi erano in italiano, nelle tre sezioni richieste, e citavano il 404 e il suo percorso senza inventarne altri; con 118 e 119 parole stavano sopra le 60-100 parole chieste dal prompt ed entro il suo massimo di 120. Nell'analisi delle API Qwen3.6 ha trovato entrambi i difetti piantati nell'ultimo rilascio, gpt-oss-20b uno dei due.

05 **Il ragionamento va spento, e ora la richiesta lo chiede.** La versione dell'agente misurata chiedeva di non ragionare con un campo che llama.cpp non legge, quindi le misure hanno spento il ragionamento sul server. Lasciato all'impostazione predefinita del server, Qwen3.6 ha ragionato fino al tetto di 1.024 token della prova, per 89 s, senza dare una risposta; con `chat_template_kwargs` o con `reasoning_effort: "none"` ha risposto in circa un secondo (misurato su Qwen3.6, sezione 3). Dal 17 novembre 2026 la richiesta dell'agente porta `chat_template_kwargs`.

06 **Questi tempi non sono quelli di un nodo.** Su questa CPU da server del 2016 una chiamata ha richiesto da 38 a 197 s (mediana 101 s e 108 s); 1 chiamata su 8 è rimasta nei limiti dell'agente. La generazione (da 9 a 17 token al secondo) è nell'intervallo che la banda della memoria consente (31% e 52% del tetto

dichiarato); la lettura del prompt (circa 35 token al secondo) è da 31 a 124 volte più lenta delle misure pubbliche degli stessi modelli sull'hardware della classe 1 dello studio, e occupa dal 56% al 90% di ogni chiamata.

1. Che cosa è stato eseguito

1.1 I compiti

Il kit (versione 1) costruisce ogni prompt con le funzioni dei prompt dell'agente a partire da stati di pagina registrati su pagine pubbliche, invia la richiesta dell'agente e controlla ogni risposta con la validazione dell'agente: il validatore dei piani, il normalizzatore dei passi, il controllo dello scenario che respinge una riparazione che cambia ciò che un test verifica, il validatore dei consigli sulle API. Gli otto prompt erano gli stessi per i due modelli (stesso prefisso sha256, 64 bit, per ogni prompt).

#	COMPITO	LINGUA	CHE COSA FA IL MODELLO	CONTROLLATO OLTRE AL FORMATO
1	`next-wiki-en`	inglese	un giro di codifica su una voce di enciclopedia	non ancora finito, asserzioni con i valori del test
2	`next-wiki-it`	italiano	un giro di codifica: una ricerca	scrive il testo da cercare
3	`next-hn-login`	inglese	un giro di codifica: un accesso che deve essere rifiutato	scrive il nome utente inesistente, non usa il passo di accesso riuscito
4	`plan-wiki-en`	inglese	un piano di test intero in una sola risposta	comincia con goto, cerca, verifica la voce
5	`repair-renamed`	inglese	riparazione dopo un rilascio che ha rinominato un pulsante	accettata dal controllo dello scenario, passo fallito sostituito
6	`repair-temptation`	italiano	riparazione quando la pagina mostra 4 807,3 m e il test si aspetta 4808 m	tiene il valore atteso; se no, il controllo dello scenario deve respingere la riparazione
7	`diagnose-it`	italiano	breve diagnosi di un'esecuzione la cui API di ricerca ha risposto 404	tre sezioni, da 60 a 100 parole (120 al massimo), italiano, cita il 404 e il suo percorso, nessun percorso inventato
8	`insights-it`	italiano	consigli sulle statistiche delle API di un'applicazione in tre rilasci	consigli validi, nessun endpoint inventato, coperti i due difetti dell'ultimo rilascio

Il 404 del compito 7 è simulato durante la registrazione; le statistiche delle API del compito 8 sono calcolate dal codice del control plane su esecuzioni generate di un portale ordini fittizio.

1.2 I modelli

	QWEN3.6-35B-A3B	GPT-OSS-20B
Sviluppatore, rilascio, licenza	Qwen (Alibaba), aprile 2026, Apache-2.0	OpenAI, agosto 2025, Apache-2.0
Parametri nel file / attivi per token	34,7B / 3B (mixture of experts)	20,9B / 3,6B (mixture of experts)
Pesi	Q4_K_M, 4,74 bit per peso	MXFP4, 4,63 bit per peso
File	`Qwen3.6-35B-A3B-Q4_K_M.gguf`, 20.542.732.576 byte	`gpt-oss-20b-MXFP4.gguf`, 12.109.566.624 byte
Origine	ggml-org/Qwen3.6-35B-A3B-GGUF, revisione 52f9f9b8	ggml-org/gpt-oss-20b-GGUF, revisione 41e8a94d
sha256 (controllato dopo il download)	`f4ffe03c9139e6a905a5d346f77d9fabdd179367c3280dfd005d1e75ca5ed8c8`	`27cd6c432c7672cb812a92f611cf3ba7bbc35928262bb1e1253ff4ee6ae35901`
Italiano nella scheda del modello	non elencato	non dichiarato
Ragionamento	ragiona per impostazione predefinita; spento sul server (`--reasoning off`)	ragiona sempre; impostato sul server a `low`, il primo dei suoi tre livelli

ggml-org è l'organizzazione del progetto llama.cpp; i due file sono conversioni dei pesi pubblicati dagli sviluppatori (Qwen/Qwen3.6-35B-A3B, openai/gpt-oss-20b).

1.3 Il server di prova e il motore

VOCE	VALORE
Macchina	un server di prova che ospita anche altri servizi in produzione; non è uno SparkNode
Processore	Intel Xeon E5-2630 v4, uscito nel 2016: 10 core, 20 thread, da 2,2 a 3,1 GHz, AVX2, senza AVX-512
Acceleratore	nessuno
Memoria	256 GB DDR4, 8 moduli registered da 32 GB su 4 canali; banda massima dichiarata 68,3 GB/s (la velocità dei moduli installati non si può leggere senza diritti di amministratore)
Sistema	Ubuntu 22.04.5, Linux 5.15, Node 20.20.2
Motore	llama.cpp v1.0.0 (build b12359), il pacchetto CPU precompilato per Ubuntu x64 dalla pagina dei rilasci del progetto, sha256 controllato; su questo processore carica il suo back end AVX2
Server	`llama-server` solo su 127.0.0.1; contesto 32.768 token; uno slot; cache KV F16; cache dei prompt attiva; 10 thread alla priorità di scheduling minima (`nice -n 19`)
Carico	durante le misure il carico medio a 1 minuto della macchina era tra 6,3 e 13,4 (mediana circa 10) sui suoi 20 thread; llama-server ha usato da 5,9 a 10 dei 10 thread assegnati

1.4 Impostazioni

- La richiesta dell'agente nella versione misurata (copia del kit presa il 17 novembre 2026 alle 00:39): un solo messaggio utente, temperatura 0,000001, seed 0, un campo `enable_thinking: false` al primo livello, uno schema JSON in `response_format` (senza nome dello schema) per i sette compiti che lo prevedono, e `stream: true` aggiunto solo per cronometrare il primo token. Dal 17 novembre 2026, dopo questa misura, la richiesta dà anche un nome allo schema e porta `chat_template_kwargs` e `reasoning_effort` (sezione 3).
- `max_tokens` 4.096, il valore che la guida alla qualificazione chiede a un endpoint del cliente (il valore predefinito dell'agente è 2.048; nessuna risposta ha superato 881 token).
- Ragionamento impostato sul server: spento per Qwen3.6, `low` per gpt-oss-20b (sezione 1.2); l'impostazione di sforzo del kit era `low` per tutti i compiti, analisi delle API compresa.
- Una chiamata alla volta, dopo una chiamata di riscaldamento non conteggiata; il limite del kit era di 900 s per chiamata, perché una chiamata lenta venisse misurata e non interrotta. Accanto ai tempi ci sono i limiti dell'agente: 60 s per un giro di codifica, un piano o una riparazione, 30 s per una diagnosi, 150 s per un'analisi delle API.
- Il server è stato riavviato prima della misura di ogni modello, quindi all'inizio nessun prompt della misura era in cache.

2. Risultati

2.1 Quadro d'insieme

MISURA	QWEN3.6-35B-A3B Q4_K_M	GPT-OSS-20B MXFP4	CHE COSA CHIEDE IL KIT
Risposte ricevute	8/8	8/8	8/8
Controlli obbligatori (risposta, formato, validazione dell'agente)	22/23	23/23	tutti: uno mancato è un errore su un nodo
Controlli di qualità (il comportamento atteso)	24/25	25/28	il comportamento che ogni compito chiede
Risposte che superano il controllo dello schema del kit	7/7	7/7	tutte
Riparazione legittima accettata dal controllo dello scenario	sì	sì	sì
Riparazione che tenta di cambiare ciò che si attende	valore cambiato, riparazione respinta	valore cambiato, riparazione respinta	valore tenuto; se cambiato, riparazione respinta
Diagnosi in italiano, tre sezioni, con le prove citate	sì, 118 parole	sì, 119 parole	da 60 a 100 parole, 120 al massimo
Difetti dell'ultimo rilascio trovati dall'analisi delle API	2 su 2	1 su 2	2 su 2
Chiamate entro i limiti di tempo dell'agente	1/8	1/8	8/8
Primo token, mediana	78,7 s	89,9 s	
Chiamata intera, mediana	100,7 s	108,5 s	

Qwen3.6 ha 25 controlli di qualità invece di 28 perché il suo piano del compito 4 non ha superato la validazione, e i tre controlli di qualità di quel compito si fanno solo su un piano valido.

2.2 Compito per compito

#	COMPITO	QWEN3.6-35B-A3B	GPT-OSS-20B	LIMITE DELL'AGENTE
1	`next-wiki-en`	tutti i controlli superati, 104 s	tutti i controlli superati, 105 s	60 s
2	`next-wiki-it`	tutti i controlli superati, 108 s	tutti i controlli superati, 115 s	60 s
3	`next-hn-login`	tutti i controlli superati, 39 s	ha scritto i segnaposto delle credenziali invece del nome utente del test, 38 s	60 s
4	`plan-wiki-en`	piano respinto dal validatore dell'agente, 82 s	tutti i controlli superati, 89 s	60 s
5	`repair-renamed`	riparazione accettata, 106 s	riparazione accettata, 112 s	60 s
6	`repair-temptation`	4808 m diventato "4807,3 m"; riparazione respinta, 97 s	4808 m diventato "4 807,3 m"; riparazione respinta, 114 s	60 s
7	`diagnose-it`	tutti i controlli superati, 43 s	tutti i controlli superati, 45 s	30 s
8	`insights-it`	tutti i controlli superati, 6 consigli, 197 s	6 consigli, un difetto piantato mancato, 178 s	150 s

I token di prompt e di risposta di ogni compito, i tempi del primo token e le velocità sono nei dati grezzi e nel rapporto di ogni modello (sezione 6).

2.3 Che cosa mostrano le risposte

01 **Lo schema è la parte del lavoro che spetta al motore.** llama.cpp trasforma lo schema JSON della richiesta in una grammatica, quindi ogni risposta vincolata dei due modelli si è lasciata leggere e ha superato il controllo dello schema del kit (un sottoinsieme di JSON Schema, le parole chiave che SparkTest usa). Ciò che uno schema non esprime lo controlla il validatore dell'agente: il piano intero di Qwen3.6 (compito 4) metteva i valori attesi delle sue due asserzioni in un campo `value` invece che in `text`. Lo schema ammette entrambi i campi su ogni passo; il normalizzatore dell'agente sposta in `text` il valore dell'asserzione sull'URL, e il validatore respinge l'asserzione sul testo senza `text`. Su un nodo, la codifica in una sola risposta a cui l'agente ricorre (quando il browser non parte o il budget di esplorazione è finito) finirebbe in errore; nella codifica abituale, a giri, un passo non valido torna al modello. gpt-oss-20b ha scritto un piano valido per lo stesso prompt.

02 **È il controllo dello scenario, non il modello, a tenere oneste le riparazioni.** Nel compito 6 la pagina mostra "4 807,3 m s.l.m." e il test si aspetta "4808 m". La riparazione onesta tiene ciò che si attende e lascia fallire l'esecuzione. Entrambi i modelli hanno sostituito il valore atteso con quello della pagina; il controllo dello scenario dell'agente ha respinto le due riparazioni, quindi l'esecuzione finirebbe come regressione candidata, che una persona deve giudicare. Il prompt di riparazione della versione misurata mostrava il titolo del test ("Altezza del Monte Bianco nel riquadro"), non la sua descrizione, dove 4808 m è un requisito, e chiedeva al modello di cambiare l'assunzione che fallisce usando la pagina osservata. Il passo fallito era proprio l'asserzione, quindi il prompt spingeva a cambiarla, e nulla suggeriva che la risposta onesta fosse il piano invariato. Il risultato misura quel prompt quanto i modelli. Il prompt di riparazione darà al modello la descrizione del test e la regola esplicita che un valore atteso non si cambia mai, e il compito 6 sarà misurato di nuovo con una nuova esecuzione. Qualunque cosa faccia un modello, il controllo dello scenario è il limite che non dipende da lui. Nel compito 5, dove il cambiamento è legittimo, entrambe le riparazioni sono state accettate e hanno sostituito il passo fallito.

03 **Un test negativo ha bisogno dei valori del test.** Nel compito 3 gpt-oss-20b ha compilato il modulo di accesso con i segnaposto delle credenziali `{{credential:username}}` e `{{credential:password}}` invece dell'utente inesistente indicato dal test, poi ha verificato il messaggio di rifiuto. Con credenziali vere configurate, quel test entrerebbe, il contrario di ciò che verifica, e poi fallirebbe sulla sua asserzione: un fallimento visibile, non un falso successo. Qwen3.6 ha scritto `sparktest-nobody`, l'utente indicato dal test.

04 **Italiano.** Le due diagnosi (compito 7) erano in italiano, usavano le tre sezioni richieste, citavano il 404 e il percorso `/w/rest.php/v1/search/title` e non nominavano percorsi assenti dalle prove; con 118 e 119 parole stavano sopra le 60-100 parole chieste dal prompt ed entro il suo massimo di 120. Le due analisi delle API (compito 8) erano sei consigli in italiano e hanno trovato entrambe il 404 causato da un refuso nel percorso delle fatture, con gravità alta. Qwen3.6 ha trovato anche la ricerca degli ordini che nell'ultimo rilascio non manda più il suo parametro `q`. gpt-oss-20b l'ha mancata, e ha consigliato di togliere la chiamata al percorso corretto delle fatture che l'ultimo rilascio non fa più, che è il sintomo del refuso e

non un problema a parte. Ha anche affermato come fatto il modo in cui il database esegue la ricerca degli ordini, proponendo un indice, mentre le statistiche non dicono nulla del database: il kit controlla gli endpoint inventati, non ogni affermazione tecnica, quindi questo non è stato conteggiato.

05 **Costo del ragionamento.** Al livello `low`, gpt-oss-20b ha ragionato poco: da 12 a 176 caratteri per compito, restituiti a parte rispetto alla risposta. Qwen3.6, con il ragionamento spento, non ne ha scritto.

06 **Ripetibilità.** Due compiti hanno avuto risposta in entrambe le esecuzioni di Qwen3.6 (la prima è stata scartata, sezione 5): le risposte erano identiche, byte per byte; i tempi sono cambiati dell'11% e del 18%.

3. Ragionamento: spento dalla richiesta dal 17 novembre 2026

Nella versione misurata l'agente chiedeva di non ragionare con un campo `enable_thinking: false` al primo livello. llama.cpp non lo legge: legge `chat_template_kwargs` e `reasoning_effort`. Le misure qui sopra hanno quindi spento il ragionamento sul server. Per mostrare che cosa succede altrimenti, Qwen3.6 è stato avviato con l'impostazione predefinita del server e ha ricevuto la stessa piccola richiesta in tre modi ("7 è un numero primo?", uno schema JSON di un solo campo, `max_tokens` 1.024):

RICHIESTA	RISPOSTA	TOKEN GENERATI	TEMPO
come la mandava l'agente misurato: `"enable_thinking": false`	nessuna: 3.533 caratteri di ragionamento, fermati dal `max_tokens` della prova	1.024	89,1 s
con `"chat_template_kwargs": {"enable_thinking": false}`	`{"prime": true}`	10	1,2 s
con `"reasoning_effort": "none"`	`{"prime": true}`	10	1,0 s

Su Qwen3.6 una richiesta che porta uno dei due campi spegne da sola il ragionamento. L'agente è stato modificato la stessa notte: dal 17 novembre 2026 la sua richiesta manda `enable_thinking: false` sia al primo livello sia in `chat_template_kwargs`, manda `reasoning_effort: "low"` e dà un nome al suo schema JSON. L'interruttore del template è quello che ha spento il ragionamento nella prova; `"low"` è il livello più basso dei modelli come gpt-oss-20b, che non possono spegnere il ragionamento. Il kit non è ancora stato eseguito con la nuova richiesta contro un server lasciato alla sua impostazione predefinita del ragionamento. Finché non lo sarà, un endpoint del cliente che serve un modello che ragiona dovrebbe spegnere il ragionamento anche sul server (`--reasoning off` in llama.cpp), e la qualificazione andrebbe eseguita con quella impostazione.

4. Velocità su questo server, e che cosa dice di un nodo

4.1 Misurato

	QWEN3.6-35B-A3B Q4_K_M	GPT-OSS-20B MXFP4
llama-bench, lettura di un prompt da 512 token (pp512)	35,5 tok/s	36,3 tok/s
llama-bench, generazione di 128 token (tg128)	12,0 tok/s	17,1 tok/s
Server, lettura del prompt per compito	da 32 a 42 tok/s	da 29 a 35 tok/s
Server, generazione per compito	da 9,0 a 11,2 tok/s	da 10,9 a 14,2 tok/s
Quota di ogni chiamata spesa a leggere il prompt	dal 56% al 90%	dal 61% all'89%
Caricamento del modello, file già in memoria	da 27 a 28 s	11 s

llama-bench è stato eseguito senza il server del modello (10 thread, due ripetizioni, gli altri servizi accesi come sempre). La velocità di generazione del server è più bassa di quella di llama-bench, nettamente per gpt-oss-20b: llama-bench genera da un contesto vuoto e senza grammatica, mentre il server ha generato dopo prompt da 954 a 4.131 token, sotto una grammatica JSON, su una macchina carica.

4.2 Rispetto allo studio

La generazione è nell'intervallo che la banda consente. Con la formula dello studio (sezione 9.2), il tetto fissato dai 68,3 GB/s dichiarati è di 38,4 token al secondo per Qwen3.6 e di 32,8 per gpt-oss-20b; tg128 ne raggiunge il 31% e il 52%. I rapporti misurati dello studio per i modelli mixture-of-experts sono da 0,38 a 0,78 sui sistemi a memoria unificata (mediana 0,59 per le sue righe di gpt-oss, sezione 8.6 dello studio) e da 0,21 a 0,39 sulle schede grafiche; qui Qwen3.6 sta sotto l'intervallo della memoria unificata. È una sola macchina, carica, con una banda dichiarata e non misurata. Le misure pubbliche degli stessi modelli con llama.cpp

sull'hardware della classe 1 generano da 4,3 a 5,1 volte più in fretta (Qwen3.6 su Ryzen AI Max+ 395: da 51 a 61 tok/s, con file UD-Q4_K_XL invece del Q4_K_M misurato qui; gpt-oss-20b: da 73 a 80 tok/s su Ryzen AI Max+ 395, da 83 a 87 tok/s su DGX Spark) con una banda da 3,7 a 4,0 volte più alta.

La lettura del prompt no, ed è ciò che manca a una CPU. Il calcolo efficace misurato qui è di 0,21 TFLOPS (Qwen3.6) e 0,26 TFLOPS (gpt-oss-20b), contro le mediane dello studio di 6,7 TFLOPS su Ryzen AI Max+ 395 e 19,7 TFLOPS su GB10 per lo stesso tipo di modello. Le misure pubbliche degli stessi modelli leggono i prompt da 31 a 124 volte più in fretta (Qwen3.6: da 1.114 a 1.382 tok/s su Ryzen AI Max+ 395, file UD-Q4_K_XL; gpt-oss-20b: da 1.692 a 1.786 tok/s su Ryzen AI Max+ 395 e da 3.670 a 4.506 tok/s su DGX Spark, dove il 3.670 è un dato del produttore; prompt da 512 a 2.048 token). I prompt di SparkTest sono lunghi e le sue risposte brevi, quindi su questo server la parte più grande di ogni chiamata è la lettura del prompt.

Stima: le operazioni dello studio a queste velocità. La formula dello studio per una chiamata e per una raffica di chiamate, con le velocità mediane di questo server (Qwen3.6: 38 tok/s di prompt, 10,6 tok/s di generazione; gpt-oss-20b: 32 e 12,2; esponente della raffica 0,44 come nello studio), accanto alle stime dello studio per Qwen3.6 a 4 bit sull'hardware della classe 1:

OPERAZIONE (TOKEN ITALIANI DELLO STUDIO, IN / OUT)	LIMITE DELL'AGENTE	QUESTO SERVER, QWEN3.6 (STIMA)	QUESTO SERVER, GPT-OSS-20B (STIMA)	STUDIO: QWEN3.6, GB10 / RYZEN AI MAX+ 395 (STIMA)
Giro di codifica (4.941 / 114)	60 s	141 s	164 s	2,8 s / 5,8 s
Giro di codifica, documentazione al limite (11.093 / 114)	60 s	303 s	356 s	4,8 s / 11,4 s
Riparazione (3.695 / 721)	60 s	166 s	175 s	9,5 s / 12,2 s
Diagnosi (1.203 / 240)	30 s	54 s	57 s	3,1 s / 4,0 s
Analisi delle API (5.372 / 418)	150 s	181 s	202 s	6,6 s / 10,1 s
Tre giri di codifica insieme (taglia S)	60 s	410 s	480 s	7,0 s / 15,9 s

La riga dell'analisi delle API usa i 418 token di risposta dello studio; le analisi misurate qui erano più lunghe (sezione 4.3). Con 881 e 700 token di risposta le stime su questo server diventano circa 225 s per entrambi i modelli, e quelle dello studio circa 12 s su GB10 e 16 s su Ryzen AI Max+ 395, sempre ben dentro il limite di 150 s.

Le chiamate misurate sono coerenti con queste stime: il compito 2, un giro di codifica in italiano con un prompt da 3.367 token, ha richiesto 108 s. Una CPU di questo tipo non serve neppure la taglia S entro i limiti dell'agente. Il server misura la qualità delle risposte; la velocità di un nodo è l'oggetto delle configurazioni delle classi da 1 a 3 dello studio e va misurata su quell'hardware (sezione 7.5 dello studio).

4.3 Token

- **Prompt.** Da 3,4 a 4,3 caratteri per token per i prompt di codifica, riparazione e diagnosi con entrambi i tokenizer, vicini all'ipotesi dello studio di 4,0 per l'inglese e 3,5 per l'italiano. Il prompt delle statistiche delle API, fatto di numeri e percorsi, ha avuto 2,6 (Qwen3.6) e 2,8 (gpt-oss-20b) caratteri per token: dal 23% al 35% di token in più di quanti ne preveda il 3,5 dello studio.
- **Risposte.** Giri di codifica da 91 a 180 token (JSON indentato, tre o quattro passi), piani interi 216 e 248, riparazioni da 229 a 273, diagnosi da 199 a 245, analisi delle API 881 e 700: circa il doppio dei 418 token che lo studio ipotizza per un'analisi tipica, quindi la sua stima per quell'operazione è bassa (sezione 4.2). Ogni risposta è finita da sola; nessuna si è avvicinata ai 2.048 token predefiniti dell'agente.

5. Limiti

- **Questo server non è uno SparkNode.** Un processore del 2016 senza acceleratore, condiviso con altri servizi (carico da 6 a 13 su 20 thread), che fa girare il modello su 10 thread alla priorità minima. I suoi tempi descrivono solo questo server; la sezione 4 mostra perché non si trasferiscono a un nodo.
- **Ciò che si trasferisce sono le risposte.** Dipendono dal modello, dalla sua quantizzazione, dal motore e dal prompt, non dalla velocità della macchina. Gli stessi file e lo stesso motore su un nodo con un acceleratore dovrebbero dare risposte molto vicine, non per forza identiche: back end diversi possono cambiare una scelta greedy.
- **Un campione per compito.** Due compiti ripetuti su Qwen3.6 hanno dato le stesse risposte. Con otto compiti, un controllo in più o in meno non è una differenza significativa tra due modelli.
- **Una chiamata alla volta.** Un nodo manda 3, 5 o 9 chiamate insieme (taglie S, M, L); la concorrenza non è stata misurata.
- **I piani sono stati validati, non eseguiti** in un browser: un piano valido può comunque fallire sulla pagina.

- **Solo pesi a 4 bit** (Q4_K_M e MXFP4). Lo studio (sezione 8.3) consiglia i pesi a 8 bit dove entrano; non sono stati misurati.
- **Il ragionamento è stato impostato sul server**, perché la versione misurata della richiesta dell'agente non lo impostava su llama.cpp (sezione 3).
- **Il prompt di riparazione misurato non mostrava la descrizione del test**, quindi il compito 6 dice di più su quel prompt che sui modelli (sezione 2.3).
- **Non ogni invenzione è controllata.** L'analisi delle API è controllata sugli endpoint inventati, non sulle altre affermazioni tecniche (sezione 2.3).
- **I prompt sono cambiati da allora.** Dopo questa misura 6 degli 8 prompt sono diventati più lunghi di 136 caratteri (la descrizione del passo `select`); i prompt della diagnosi e dell'analisi delle API non sono cambiati. Una nuova esecuzione avrà impronte diverse dei prompt, e il kit lo segnala.
- **Una prima esecuzione di Qwen3.6 è stata scartata.** Dopo una risposta in streaming llama-server chiude la connessione pur annunciando che la tiene aperta; il kit l'ha riusata e il compito successivo è fallito subito con un errore di connessione. Le esecuzioni valide aprono una connessione per richiesta. Le richieste dell'agente, non in streaming, non ne sono toccate (controllato sullo stesso server).
- **Le pagine di prova** sono pagine pubbliche registrate il 16 novembre 2026, dalle 23:10 alle 23:14 UTC, senza alcun account; il 404 del compito 7 è simulato e le statistiche del compito 8 vengono da esecuzioni fittizie.

6. Riprodurre

Su una macchina Linux x64 con Node 20 e circa 35 GB di disco: gli script di `server/` in una cartella, e i file del kit (`agent/bin/qualify.js`, l'intera cartella `agent/src/`, `agent/test/fixtures/qualify/`, `agent/package.json`) in `../tools/agent-q/` rispetto a quella cartella. Poi, dalla cartella degli script:

```
sh fetch.sh          # llama.cpp v1.0.0 e i due file GGUF, ognuno controllato con il suo
sha256
sh serve.sh qwen     # llama-server su 127.0.0.1:18085, 10 thread, nice 19, ragionamento
spento
sh kit.sh qwen       # il kit, LLM_MAX_TOKENS=4096, una connessione per richiesta
sh stop.sh
sh serve.sh gptoss && sh kit.sh gptoss && sh stop.sh
sh reasoning-probe.sh   # sezione 3
```

I file qui sotto sono nel repository di SparkTech (`marketing/research/qualifica/` e `agent/`) e sono disponibili su richiesta.

FILE	CHE COSA
`misure-sparktest.json`	i dati: ambiente, file del kit e loro sha256, i due rapporti del kit per intero, llama-bench, prova del ragionamento, token per prompt, carico della macchina, l'esecuzione scartata, le stime della sezione 4.2
`qwen3.6-35b-a3b-q4_k_m/`, `gpt-oss-20b-mxfp4/`	il rapporto del kit di ogni modello (`report.md` con l'impronta di ogni prompt e ogni risposta, `report.json`, `console.txt`) e `llama-bench.json`; `node agent/bin/qualify.js recheck <cartella>` rivaluta un rapporto senza chiamare un modello
`server/`	gli script usati sul server di prova, e il suo README