

NOT THE BEST, BUT GOOD ENOUGH

Allineare la capacità dei modelli linguistici al valore del task: un framework di adozione per CIO, CTO e AI leader

---

Antonio Dina per Seeweb

seeweb
THINK CLOUD

NOT THE BEST, BUT GOOD ENOUGH

Allineare la capacità dei modelli linguistici al valore del task: un framework di adozione per CIO, CTO e AI leader

---

Antonio Dina per Seeweb

Executive White Paper — Edizione estesa 2026

Review tecnico-editoriale · Routing, TCO, governance e sostenibilità

Le cifre quantitative sono presentate, dove non derivabili da fonti primarie verificate in questa sede, come range indicativi basati su pattern di settore consolidati, con invito esplicito a validarle al momento dell'uso.

Executive Summary

Negli ultimi anni l'adozione enterprise degli LLM è stata guidata da una logica implicita: scegliere il modello più capace disponibile, indipendentemente dal task. Questa logica ha prodotto architetture semplici da progettare ma strutturalmente inefficienti, con costi di inference che scalano più velocemente del valore generato e con una gestione del rischio delegata implicitamente alla "potenza" del modello anziché a un disegno esplicito.

Questo white paper sostiene una tesi diversa: la variabile che determina il successo di un'iniziativa AI enterprise non è la scelta del modello più potente, ma la capacità dell'organizzazione di costruire un Sistema, **non un singolo modello,** che assegni a ciascun task il livello di capacità computazionale minimo sufficiente a soddisfare requisiti di qualità, rischio e costo definiti a priori.

I capitoli che seguono sviluppano questa tesi lungo cinque assi:

- **Economico.** il costo per token è solo una componente, spesso minoritaria, del Total Cost of Ownership reale di un sistema LLM enterprise.
- **Architetturale.** routing semantico, cascading e verifica di confidenza sono oggi pattern enterprise consolidati, non ottimizzazioni sperimentali.
- **Operativo.** la latenza va trattata come KPI di business, non solo come metrica tecnica, perché incide su conversione, produttività e costo per unità di lavoro completata.
- **Di governance.** "good enough" è anzitutto una decisione di gestione del rischio — privacy, compliance, disponibilità del servizio — e solo in seconda battuta una leva di risparmio.
- **Ambientale.** l'evidenza disponibile, pur con margini di incertezza reali, indica in modo consistente che il dimensionamento adeguato dei modelli riduce il consumo energetico per unità di lavoro svolta.

Il documento propone un framework decisionale a quattro livelli (flash, reasoning intermedio, reasoning avanzato, frontier) e una checklist di raccomandazioni operative per CIO, CTO e AI leader, con l'obiettivo di trasformare "Not the Best, but Good Enough" da principio intuitivo a pratica di ingegneria e governance ripetibile.

Prefazione

La domanda che molte organizzazioni si pongono sull'intelligenza artificiale non riguarda più l'opportunità di adottarla, ma il modo migliore per integrarla nei processi evitando che i costi diventino ingestibili, la qualità decada in modo silenzioso e i rischi diventino difficili da misurare o attribuire. Scegliere un singolo modello, per quanto avanzato, non è più sufficiente. La vera differenza la fa la struttura del sistema nel suo complesso, ovvero le modalità con cui si stabilisce quanta risorsa computazionale destinare a ciascuna richiesta, a come si smistano i vari compiti, chi governa la catena di chiamate nei flussi agentici per rendere l'intera infrastruttura trasparente e controllabile e, non in ultimo: come riuscire a proteggere i dati condivisi con il sistema di intelligenza artificiale.

I modelli più avanzati migliorano in modo evidente le proprie prestazioni generando sequenze intermedie di token prima di produrre la risposta finale. Queste fasi, dette "di ragionamento" aumentano le probabilità di ottenere un risultato migliore, nascondere la modalità di ragionamento esplicita o chiedere al sistema di non mostrare questi passaggi non elimina il processo di calcolo sottostante e la spesa conseguente in termini di token.

Di conseguenza, diventa indispensabile spostare il focus dal prezzo del singolo token al costo totale di gestione. Valutare un sistema esclusivamente dal prezzo per milione di token offre una visione parziale. Il costo reale include fattori decisivi come la latenza avvertita dagli utenti finali, il tempo dedicato alla risoluzione degli errori e ai tentativi falliti, le attività di monitoraggio, l'orchestrazione dei flussi, la formazione del personale e gli adempimenti normativi. Concentrarsi soltanto sul risparmio marginale del token rischia di generare benefici immediati ma fittizi, destinati a tradursi in costi maggiori quando il calo della qualità costringe i processi operativi a compensare le inesattezze. La latenza stessa costituisce un vincolo di business determinante, dato che una soluzione economica ma eccessivamente lenta può rivelarsi inutilizzabile nella pratica.

Questa dinamica emerge chiaramente nell'analisi economica dei processi di elaborazione. Facciamo qualche esempio. Affidare ogni tipologia di richiesta a un unico modello di grandi dimensioni porta a sprecare risorse sulle operazioni basilari ed a creare colli di bottiglia su quelle articolate. Un approccio che diversifica la gestione in base alla difficoltà delle richieste o ai contesti di dominio delle stesse (un argomento di matematica piuttosto che uno di comunicazione efficace), indirizzando i compiti semplici verso modelli leggeri e riservando le risorse più avanzate ai casi complessi, permette di ottimizzare l'andamento della spesa complessiva.

Per gestire questa ripartizione delle risorse si possono combinare diverse strategie operative. Come viene ampiamente illustrato in questo white paper, l'elemento comune a queste soluzioni è il superamento della scelta statica di un modello, fosse anche il migliore in tutti i benchmark, in favore di una gestione dinamica delle capacità disponibili. Ciò che conta è comprendere quali interazioni siano indispensabili, quali passaggi si possano accorciare o eliminare e dove posizionare controlli strategici per evitare che le spese aumentino in modo incontrollato, integrando l'affidabilità e il controllo dei costi all'interno dello stesso progetto.

Questo approccio si riflette direttamente sulla governance del sistema. Garantire la tracciabilità delle decisioni diventa un requisito fondamentale per poter valutare e correggere il sistema. Per affrontare tale complessità, lo strumento decisionale più efficace non consiste in una graduatoria di modelli, ma in una matrice strutturata che colleghi ciascuna tipologia di compito al livello di capacità e calcolo necessario. Tale visione guida anche

la scelta dell'infrastruttura, in cui cloud pubblico, ambienti dedicati di Private AI in cloud e sistemi interni rappresentano opzioni complementari da integrare secondo le esigenze di velocità, controllo dei dati e costi. Un'architettura ben bilanciata consente di mantenere la gestione diretta dei carichi di lavoro più critici e prevedibili, sfruttando le risorse esterne per far fronte alle variazioni di volume e offrendo - come sottoprodotto prezioso - un tracciamento atomico del flusso dei dati e della loro tutela (sia dal punto di vista normativo, che strategico).

Individuare il modello più potente del momento è “vanità”; saper progettare un sistema complessivo in cui costi, qualità e gestione del rischio siano la conseguenza di regole chiare, misurabili e modificabili - e non il risultato di scelte di fornitori globali che ingabbiano l’utente in un recinto di Cloud e AI - consente la costruzione di un vantaggio competitivo solido e duraturo.

Evitare di restare vincolati a configurazioni in cui l’allocazione delle risorse, il livello di ragionamento, la tracciabilità e i meccanismi di controllo siano opachi o difficilmente adattabili agli obiettivi specifici dell’organizzazione costituisce esso stesso un elemento di valore e di mitigazione del rischio.

Un’architettura ben progettata consente di mantenere il governo su come e quando utilizzare le capacità dei modelli, su quali carichi di lavoro tenere sotto controllo diretto e su come far evolvere il sistema nel tempo, in coerenza con le priorità aziendali.

In ultimo, rifuggire dall’utilizzo di sistemi centralizzati e oligopolistici da parte di un’azienda evita quello che è il suo rischio più grande: perdere l’identità e la forza derivanti dal pensiero laterale e individuale, alla base dell’essenza stessa dell’azienda e di quello che, per l’Italia, spesso individuiamo come “l’essenza del Made in Italy”: un patrimonio che va protetto e reso sempre più competitivo attraverso l’uso consapevole degli strumenti più moderni.

Antonio Baldassarra

CEO Seeweb

Indice

Executive Summary 2
Prefazione 3
Indice 5
1. Introduzione: dalla scelta del modello alla progettazione del sistema 6
2. Il panorama dei modelli nel 2026: dimensioni di scelta 6
2.1 Reasoning vs. non-reasoning: una dimensione indipendente dalla scala del modello 6
2.2 Open-weight vs. proprietario 7
2.3 Segmentazione per fascia all'interno di una stessa famiglia 7
2.4 Specializzazione tramite fine-tuning: LoRA e distillazione della conoscenza 7
Il costo che questa via non elimina: l'infrastruttura di inferenza 8
2.5 Tabella di sintesi: dimensioni di scelta 8
3. Oltre il prezzo per token: il Total Cost of Ownership (TCO) 9
3.1 Le componenti del TCO 9
3.2 Perché la sola ottimizzazione del prezzo per token può essere fuorviante 9
4. Economia dell'inference: costo, latenza e utilizzo GPU 10
4.1 La latenza come KPI di business 10
4.2 Utilizzo GPU e economie di scala dell'inference 10
4.3 La curva di costo cumulato: routing per complessità vs. modello unico 11
4.4 Vincoli strutturali alla disponibilità di capacità computazionale 11
5. Architetture di routing e cascading 12
5.1 Routing semantico 12
5.2 Cascading ed escalation basata sulla confidenza 12
5.3 Modelli verificatori ("verifier models") 12
5.4 Pipeline di inferenza multi-stadio 12
6. Workflow agentici: implicazioni di costo e affidabilità 13
7. Impatto ambientale: evidenza disponibile e cautele necessarie 15
7.1 Il costo incorporato (embodied cost) dell'hardware 15
7.2 Cosa resta valido indipendentemente dalle cifre esatte 15
8. Governance enterprise: rischio, non solo risparmio 16
8.1 Le dimensioni di rischio da governare 16
8.2 Un principio di governance operativo 16
9. Framework decisionale: quale livello di modello per quale task 16
9.1 Come usare la matrice in pratica 17
9.2 Una quinta opzione: costruire anziché scegliere 17
10. Deployment ibrido: cloud pubblico, VPC dedicata, on-premise 18
11. Raccomandazioni operative per CIO, CTO e AI leader 18
12. Conclusioni 19
13. Nota metodologica e limiti 19
14. Riferimenti e fonti per approfondimento 20
15. Glossario e acronimi 21
Acronimi tecnici 21
Organizzazioni e prodotti citati 21

1. Introduzione: dalla scelta del modello alla progettazione del sistema

Nel biennio 2024-2026 il numero di modelli linguistici disponibili commercialmente si è moltiplicato, e con esso si sono moltiplicate le combinazioni possibili di capacità, costo, latenza e modalità di deployment. In questo contesto, la domanda rilevante per un'organizzazione non è più "quale modello scegliere", ma "come costruire un sistema che scelga il modello giusto per ogni singola richiesta, in modo dinamico, verificabile e governato".

Questa distinzione è più di una sfumatura terminologica. Scegliere un modello è una decisione statica, di procurement. Progettare un sistema di allocazione dei task è una decisione architetturale continua, che richiede: una tassonomia dei task per complessità e rischio, meccanismi di instradamento (routing), meccanismi di verifica della qualità dell'output, e una governance che definisca soglie di accettabilità per ciascuna categoria di rischio.

La strategia "Not the Best, but Good Enough", qui estesa e formalizzata, non è quindi un invito a usare sistematicamente modelli meno capaci, ma un invito a rendere esplicita e misurabile la relazione tra complessità del task, capacità del modello e costo totale, anziché lasciarla implicita nella scelta di un unico fornitore o modello per l'intera organizzazione.

> ℹ ***Nota metodologica:*** *La numerazione dei paragrafi e le definizioni di categoria (flash, mid-range, reasoning, frontier) usate in questo documento sono convenzioni descrittive adottate per chiarezza espositiva; non corrispondono a una tassonomia normata o universalmente standardizzata nel settore, che a luglio 2026 resta eterogenea tra fornitori.*

2. Il panorama dei modelli nel 2026: dimensioni di scelta

Scegliere un modello non è più una decisione monodimensionale ("quanto è grande / quanto è bravo"). Le dimensioni rilevanti per una decisione enterprise sono almeno quattro, spesso indipendenti tra loro:

2.1 Reasoning vs. non-reasoning: una dimensione indipendente dalla scala del modello

Una distinzione diventata centrale nel 2025-2026 riguarda la modalità con cui un modello elabora una richiesta, non la sua dimensione o capacità parametrica. I modelli non-reasoning sono generalmente ottimizzati per produrre risposte attraverso una generazione diretta, mentre i modelli reasoning possono dedicare risorse computazionali aggiuntive a una fase intermedia di elaborazione, pianificazione e verifica prima di formulare la risposta finale. Le due categorie non corrispondono a fasce dimensionali: anche modelli di piccole dimensioni possono integrare capacità di reasoning, talvolta proprio per compensare i limiti derivanti dalla loro minore scala, mentre modelli più grandi possono essere progettati per risposte rapide e dirette. Il reasoning rappresenta quindi una modalità di utilizzo della capacità computazionale, non un indicatore automatico della dimensione o della superiorità generale del modello. Il suo impiego può migliorare le prestazioni su problemi multi-step, matematici, logici o caratterizzati da vincoli complessi, ma comporta potenzialmente maggiori consumi di token, latenza e costi di inference. L'effettivo beneficio dipende dalla relazione tra complessità del task, capacità del modello e budget computazionale disponibile.

Implicazione pratica: la scelta enterprise non dovrebbe contrapporre semplicemente modelli piccoli e veloci a modelli grandi e reasoning. Occorre valutare separatamente la capacità intrinseca del modello e la modalità di elaborazione richiesta dal task. Un modello compatto dotato di reasoning può risultare più efficace di un modello più grande utilizzato in modalità diretta per determinate attività; viceversa, attivare un reasoning esteso per una semplice classificazione o estrazione di dati può generare costi e latenze senza benefici proporzionati. La variabile decisiva è il risultato verificabile per unità di costo computazionale, non l'etichetta commerciale associata al modello.

2.2 Open-weight vs. proprietario

I modelli open-weight (pesi scaricabili e ospitabili autonomamente) offrono un profilo di costo strutturalmente diverso da quello dei modelli proprietari accessibili solo via API: spostano una parte del costo da variabile (per token, al fornitore) a fisso (infrastruttura, personale, manutenzione). Questo cambia la convenienza relativa in funzione del volume, ma introduce anche costi impliciti spesso sottostimati: gestione degli aggiornamenti, sicurezza dell'infrastruttura, capacità interna di ottimizzazione dell'inferenza (quantizzazione, batching, gestione della coda), e responsabilità diretta su disponibilità e failover.

- **Open-weight self-hosted:** tende a convenire quando il volume è alto, prevedibile e sostenuto nel tempo, e l'organizzazione dispone (o è disposta a costruire) competenze MLOps interne.
- **Proprietario via API:** tende a convenire quando il volume è variabile, quando serve accesso rapido a capacità di frontiera senza investimento infrastrutturale, o quando la responsabilità di sicurezza e aggiornamento va delegata al fornitore.
- **Pattern ibrido:** un pattern maturo nel 2026 è ibrido: modelli open-weight self-hosted per il traffico ad alto volume e bassa sensibilità, modelli proprietari per i casi complessi o ad alto rischio, orchestrati dallo stesso layer di routing.
- **Open-weight via API (senza self-hosting):** una via intermedia sempre più rilevante nel 2026 è quella di modelli open-weight (pesi aperti) ma consumati tramite un endpoint API gestito da un fornitore terzo, senza che l'organizzazione utilizzatrice faccia self-hosting. Il profilo operativo — costo variabile per token, nessun investimento infrastrutturale, nessuna responsabilità diretta su disponibilità e aggiornamento — è in pratica indistinguibile da quello di un modello proprietario via API. Questo è il modello di distribuzione prevalente per la maggior parte dei principali fornitori cinesi (ad es. le famiglie DeepSeek e Qwen) e riguarda anche una quota significativa del catalogo di Mistral. Il vantaggio distintivo rispetto a un modello proprietario puro non è quindi il costo per token in sé, ma l'assenza di lock-in sui pesi: la possibilità di portare lo stesso modello, senza modifiche, verso un self-hosting interno o un altro fornitore, qualora le condizioni commerciali o operative cambino.

2.3 Segmentazione per fascia all'interno di una stessa famiglia

Quasi tutti i principali fornitori offrono oggi più livelli all'interno della stessa famiglia di modelli, pensati esplicitamente per un uso differenziato per task. Ad esempio, la famiglia Claude di Anthropic (a luglio 2026) comprende Claude Haiku 4.5 (fascia efficiente), Claude Sonnet 5 (fascia intermedia, buon compromesso costo/capacità), Claude Opus 4.8 (fascia alta) e una fascia superiore denominata "Mythos", che include Claude Fable 5 e Claude Mythos 5. Pattern analoghi esistono presso gli altri principali fornitori (Google, OpenAI, Meta, Mistral, ecc.), ciascuno con naming e posizionamento propri e in evoluzione costante.

> ***Nota di verifica:*** *Nomi, capacità e prezzi dei singoli modelli cambiano con cadenza mensile o più rapida. Prima di qualsiasi decisione di procurement, verificare sempre la documentazione ufficiale del fornitore (ad es. docs.claude.com per Anthropic) per capacità, pricing e disponibilità correnti.*

2.4 Specializzazione tramite fine-tuning: LoRA e distillazione della conoscenza

Una terza via, distinta dallo scegliere semplicemente "un modello più piccolo tra quelli disponibili", consiste nello specializzare attivamente un modello open-weight di piccole dimensioni per un compito specifico.

Prima di percorrere questa via, vale la pena esaurire un livello più semplice ed economico: il grounding semantico. Fornire a un modello generico ed economico il contesto specifico del dominio al momento della richiesta — tramite retrieval-augmented generation (RAG), basi di conoscenza strutturate, system prompt dettagliati o definizioni esplicite di strumenti/funzioni — può renderlo sufficientemente competente su un caso d'uso ristretto senza alcun ciclo di addestramento. Nella pratica, è spesso il primo passo da percorrere: se un modello economico con un grounding semantico ben progettato raggiunge già la soglia di qualità richiesta, l'investimento aggiuntivo in fine-tuning o distillazione non è giustificato. Quando invece il grounding da solo non basta — tipicamente perché il compito richiede di interiorizzare uno stile, un formato o un pattern di

ragionamento che il contesto non può trasmettere in modo affidabile ad ogni richiesta — restano praticabili due tecniche di specializzazione via addestramento, spesso combinate tra loro:

- **LoRA (Low-Rank Adaptation).** tecnica di fine-tuning efficiente in cui, anziché riaddestrare tutti i parametri del modello, si aggiornano solo matrici di rango ridotto inserite nell'architettura. Riduce drasticamente il costo computazionale e i tempi di addestramento rispetto a un fine-tuning completo, mantenendo il modello di base invariato e rendendo l'adattamento economico anche su hardware modesto.
- **Distillazione della conoscenza (knowledge distillation).** un modello piccolo ("studente") viene addestrato per riprodurre il comportamento di un modello di frontiera ("insegnante") su un dominio specifico, usando gli output del modello di frontiera come segnale di addestramento anziché — o in aggiunta a — dati etichettati manualmente. Il modello studente non eguaglia le capacità generali dell'insegnante, ma può avvicinarsi molto alla sua qualità sul compito ristretto per cui è stato distillato.

Il risultato pratico di questa combinazione è un modello piccolo, economico da eseguire, che su un singolo task ben definito può avvicinarsi sensibilmente alla qualità di un modello frontier generalista — invertendo la logica consueta: invece di instradare dinamicamente verso un modello più capace quando serve, si costruisce a monte un modello già specializzato che raramente ne ha bisogno.

> ***Nota di verifica:*** *Questa via richiede una valutazione onesta del punto di pareggio: il fine-tuning e la distillazione hanno senso quando il volume del task specifico è alto e stabile nel tempo, e quando l'organizzazione ha (o è disposta a costruire) capacità di raccogliere dati di addestramento rappresentativi e di validare la qualità del modello specializzato prima del rilascio in produzione. Su task a basso volume o in rapida evoluzione, il costo di sviluppo e manutenzione del modello specializzato può superare il beneficio.*

Il costo che questa via non elimina: l'infrastruttura di inferenza

Un errore comune è considerare il fine-tuning come un costo una tantum. In realtà, un modello specializzato self-hosted introduce gli stessi costi infrastrutturali ricorrenti discussi nel capitolo 3 per gli open-weight in generale, **capacità GPU dedicata o riservata**, gestione della coda e del batching, monitoraggio della qualità nel tempo (drift, quando i dati in produzione si allontanano da quelli di addestramento), versionamento dei modelli, e ri-addestramento periodico quando il dominio del task evolve. Il risparmio ottenuto sul costo per token va quindi sempre confrontato con il costo pienamente carico (fully loaded) di tenere in produzione un modello personalizzato, non solo con il suo costo di addestramento iniziale.

Questo rende la specializzazione via fine-tuning una quinta dimensione del framework decisionale del capitolo 9: non solo "quale livello di modello esistente scegliere", ma anche "conviene costruire un modello specializzato proprietario per questo task", decisione che va presa con lo stesso rigore di TCO (capitolo 3) applicato alle altre opzioni di deployment (capitolo 10).

2.5 Tabella di sintesi: dimensioni di scelta

Dimensione	Cosa cambia	Domanda chiave da porsi
Reasoning vs non-reasoning	Presenza di una fase di ragionamento esplicito interno; costo e latenza per richiesta	Il task richiede pianificazione multi-step o solo pattern-matching / trasformazione diretta?
Open-weight vs proprietario	Struttura di costo (fisso vs variabile), responsabilità operativa	Il volume è alto e prevedibile? L'organizzazione ha capacità MLOps interne?
Fascia dentro la stessa famiglia	Capacità e prezzo per token all'interno dello stesso ecosistema/fornitore	Qual è il livello minimo di capacità che soddisfa in modo affidabile questo task?
Modalità di deployment	Cloud pubblico, VPC dedicata, on-prem, ibrido	Ci sono vincoli di data residency, latenza di rete o compliance che escludono alcune opzioni?

3. Oltre il prezzo per token: il Total Cost of Ownership (TCO)

La letteratura di mercato sui costi degli LLM si concentra quasi sempre sul prezzo per milione di token. È una semplificazione utile per confrontare listini, ma fuorviante come base di decisione: in un sistema enterprise maturo, il costo dei token di inference è spesso una minoranza del costo totale di possesso della soluzione.

3.1 Le componenti del TCO

- **Costo dei token.** il costo diretto fatturato dal fornitore per token di input e output.
- **Orchestrazione e ingegneria.** progettazione, manutenzione e aggiornamento di prompt, pipeline di routing, guardrail e integrazioni con i sistemi esistenti **un costo ricorrente**, non una tantum, perché prompt e pipeline richiedono manutenzione quando i modelli sottostanti cambiano versione.
- **Retry, fallback ed errori.** ogni richiesta che fallisce, produce un output non valido o richiede un secondo tentativo (anche verso un modello diverso) raddoppia o triplica il costo effettivo di quella singola unità di lavoro.
- **Monitoraggio e governance.** logging, audit trail, controlli di conformità, red-teaming periodico e reportistica per la sicurezza: costi spesso non attribuiti esplicitamente all'iniziativa AI ma reali.
- **Costo opportunità della latenza.** il tempo di attesa dell'utente o del sistema a valle ha un costo economico (produttività persa, abbandono, SLA non rispettati) che va sommato al costo computazionale diretto.

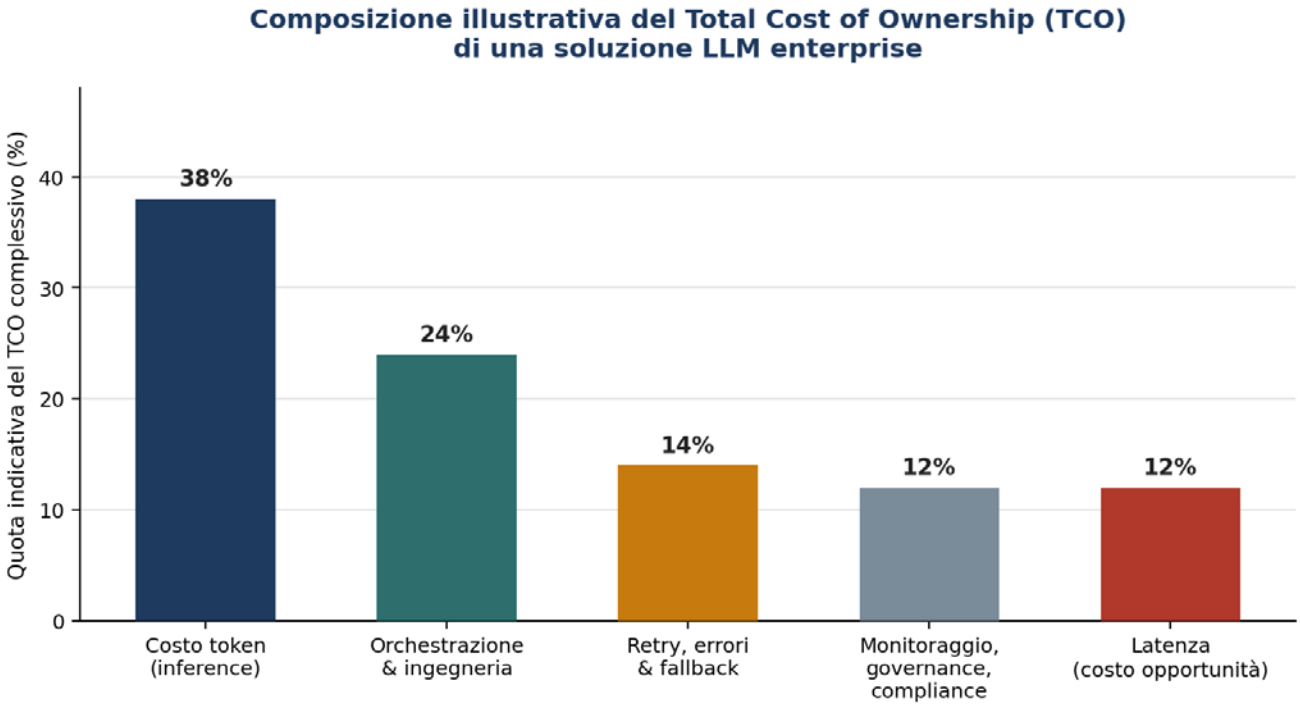


Figura 1 — Composizione illustrativa del TCO di una soluzione LLM enterprise. Le percentuali sono esemplificative e variano sensibilmente per settore, maturità dell'organizzazione e tipo di task; non costituiscono un benchmark validato esternamente.

3.2 Perché la sola ottimizzazione del prezzo per token può essere fuorviante

Un modello più economico che genera un tasso di errore più alto può costare di più a livello di TCO, se il costo di ogni retry (incluso il tempo dell'operatore umano che interviene, quando previsto) supera il risparmio ottenuto sul singolo token. Questo è il motivo per cui la scelta del modello va sempre valutata insieme al meccanismo di verifica della qualità (si veda il capitolo 5): un modello economico più un verificatore leggero può battere, in TCO, un modello costoso usato da solo ma solo se il verificatore è ben calibrato.

> ***Nota di verifica:*** *I dati quantitativi di dettaglio sul TCO (ripartizione percentuale esatta, costo medio di un retry, ecc.) variano fortemente per organizzazione e settore. Le percentuali indicate in Figura 1 sono illustrative; per un'analisi rigorosa si raccomanda di costruire un modello di TCO interno basato sui dati operativi effettivi dell'organizzazione, eventualmente supportato da benchmark di settore come quelli periodicamente pubblicati da MLCommons (MLPerf Inference) per la componente di efficienza computazionale.*

4. Economia dell'inference: costo, latenza e utilizzo GPU

4.1 La latenza come KPI di business

La latenza di un sistema LLM non è solo una metrica di esperienza utente: è un moltiplicatore di costo indiretto. In un flusso agentico con più chiamate concatenate, la latenza si somma; in un flusso interattivo, ogni secondo aggiuntivo di attesa riduce misurabilmente conversione, soddisfazione e produttività percepita, secondo pattern ben documentati nella letteratura sulla user experience dei sistemi digitali in generale (non specifici agli LLM, ma applicabili per analogia).

Un principio operativo utile: fissare, per ciascuna categoria di task, non solo una soglia di qualità minima accettabile ma anche una soglia di latenza massima accettabile, e trattare entrambe come vincoli di pari rango nella scelta del modello e dell'architettura — non solo la qualità.

4.2 Utilizzo GPU e economie di scala dell'inference

Dal lato dei fornitori (e delle organizzazioni che fanno self-hosting), l'efficienza economica dell'inference dipende in larga misura dal tasso di utilizzo dell'hardware: batching delle richieste, gestione dinamica della coda, e tecniche di ottimizzazione del modello (quantizzazione, distillazione, speculative decoding) incidono sul costo per token effettivo indipendentemente dal prezzo di listino. Questo spiega, a livello strutturale, perché il prezzo per token tende a scendere nel tempo per una data classe di capacità: non un singolo fattore, ma più curve di apprendimento sovrapposte che agiscono a livelli diversi della pipeline — hardware (efficienza per watt dei nuovi acceleratori), algoritmi (architetture di modello e tecniche di training più efficienti), software di serving (batching, scheduling, kernel ottimizzati) — ciascuna con una propria dinamica e un proprio ritmo di miglioramento, solo in parte spiegata dalla semplice pressione competitiva tra fornitori.

Questo calo del prezzo per token non si traduce automaticamente in una riduzione della spesa totale dell'organizzazione — anzi, spesso accade il contrario. È lo schema di un paradosso di Jevons applicato all'inference: quando un input diventa più efficiente da usare, il suo consumo complessivo tende ad aumentare, talvolta più che proporzionalmente al guadagno di efficienza, perché nuovi casi d'uso diventano economicamente sostenibili proprio grazie al calo di costo unitario (più chiamate per task, più agenti in parallelo, contesti più lunghi, iterazioni multiple che prima non erano giustificabili). Per la pianificazione del budget, la conseguenza pratica è che il calo del prezzo per token va trattato come un fattore che abilita più utilizzo, non come una garanzia di minore spesa complessiva: il TCO va monitorato nel tempo sul volume effettivo, non stimato per estrapolazione lineare dal solo prezzo di listino.

> ***Nota di verifica:*** *Il paradosso di Jevons è un concetto ben consolidato in economia dell'energia e delle risorse; la sua applicabilità quantitativa specifica al mercato dell'inference AI (in che misura e su quali orizzonti temporali l'aumento di volume compensi o superi il calo di prezzo unitario) è un tema di dibattito ancora aperto e non oggetto di consenso quantitativo consolidato in letteratura al momento della stesura di questo documento; va quindi trattato come chiave di lettura qualitativa, non come stima predittiva.*

Per un'organizzazione che valuta il self-hosting di modelli open-weight, questo significa che il costo reale dipende meno dal prezzo dell'hardware in sé e più dalla capacità del team di raggiungere un utilizzo elevato e sostenuto della capacità computazionale acquistata — un fattore spesso sottovalutato nei business case iniziali.

4.3 La curva di costo cumulato: routing per complessità vs. modello unico

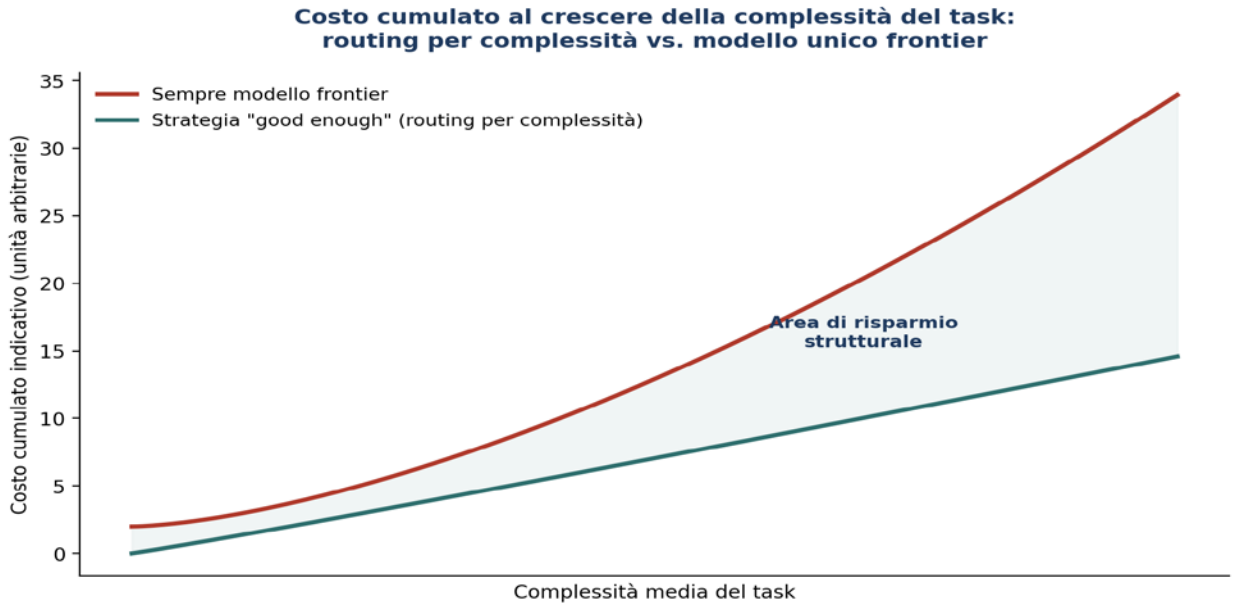


Figura 2 — Rappresentazione illustrativa (non basata su dati di mercato specifici) dell'andamento del costo cumulato al crescere della complessità media dei task, confrontando una strategia a modello unico frontier con una strategia di routing per complessità. La forma delle curve è concettuale.

Il punto chiave illustrato in Figura 2 non è la cifra esatta di risparmio, ma la forma della relazione: quando il mix di task di un'organizzazione è eterogeneo (come tipicamente accade), un'architettura a modello unico frontier paga un premio su tutta la distribuzione dei task, inclusa la quota, spesso maggioritaria, di richieste semplici che non ne avrebbero bisogno.

> ***Esempio applicativo:*** *A titolo di riscontro indicativo — non di validazione generale del fenomeno — lo stesso fornitore citato al capitolo 5 (Regolo.AI/Seeweb, per il proprio layer Brick) riporta, nella configurazione predefinita e su carichi di lavoro di agenti di coding, un risparmio dell'ordine del 40–60% rispetto all'uso sistematico di un modello frontier, coerente con un'osservazione interna di dimezzamento dei costi di inferenza nell'uso quotidiano per attività di sviluppo software. Il dato è auto-riportato dal fornitore, riferito a una tipologia di workload specifica e a uno dei cinque profili di ottimizzazione disponibili sulla piattaforma; non è quindi generalizzabile, ma è coerente in ordine di grandezza con la forma della relazione illustrata in Figura 2.*

4.4 Vincoli strutturali alla disponibilità di capacità computazionale

Al di là delle dinamiche di prezzo e di efficienza discusse sopra, la disponibilità stessa di capacità computazionale aggiuntiva incontra, nel 2026, vincoli strutturali che non dipendono dal budget dell'organizzazione né dalla bontà della sua strategia di routing. Tre colli di bottiglia, distinti ma interconnessi, meritano di essere tenuti presenti in qualunque pianificazione di capacità a medio termine:

- **Disponibilità di hardware.** l'offerta di GPU e di memoria ad alta banda resta vincolata dalla capacità produttiva di un numero ristretto di fornitori; i tempi di consegna per capacità aggiuntiva su larga scala si misurano tipicamente in trimestri, non in settimane, ed è un fattore che va pianificato indipendentemente dalla disponibilità di budget.
- **Capacità della rete elettrica.** l'allaccio di nuova capacità computazionale su larga scala è sempre più spesso vincolato non dalla disponibilità di hardware ma da quella della rete elettrica locale:

potenziamento delle sottostazioni, permessi di connessione, tempi di adeguamento infrastrutturale che in molte aree superano ormai quelli di approvvigionamento dell'hardware stesso.

- **Opposizione locale (fenomeno NIMBY).** la costruzione di nuovi data center incontra crescente resistenza da parte delle comunità locali, per ragioni che vanno dal consumo idrico ed energetico all'impatto visivo e sul valore immobiliare; questo si traduce in tempi di approvazione più lunghi e, in alcuni casi, nella cancellazione o rilocalizzazione di progetti già pianificati.

Per un'organizzazione che pianifica una strategia di capacità a 12-24 mesi, questi vincoli hanno una conseguenza pratica diretta: l'assunzione implicita che sia sempre possibile acquistare più capacità computazionale al bisogno, semplicemente aumentando il budget, è sempre meno valida. Questo rafforza, da un angolo diverso rispetto a quello puramente economico discusso nei paragrafi precedenti, l'argomento a favore di un'architettura che usi la capacità disponibile in modo efficiente (routing per complessità, dimensionamento adeguato del modello) piuttosto che assumere che la capacità aggiuntiva sia sempre acquistabile a piacimento. Gli aspetti energetici e ambientali di questi vincoli sono ripresi nel capitolo 7.

5. Architetture di routing e cascading

Il routing per complessità è oggi un pattern architetturale enterprise consolidato, non un'ottimizzazione sperimentale. Esistono diverse implementazioni, spesso combinate tra loro:

5.1 Routing semantico

Un classificatore leggero (esso stesso un modello di piccole dimensioni, o un insieme di euristiche) analizza la richiesta in ingresso e la instrada verso il modello ritenuto adeguato in base a segnali come lunghezza, presenza di vincoli logici o matematici, ambiguità, dominio (es. codice, contenuti legali, conversazione generica) e storicamente misurato tasso di successo per categoria simile.

5.2 Cascading ed escalation basata sulla confidenza

Nel pattern a cascata, la richiesta viene inizialmente gestita dal modello più economico disponibile; se un segnale di bassa confidenza viene rilevato, tramite il punteggio di probabilità del modello stesso, un modello "verificatore" separato, o una regola di business (es. mancata corrispondenza con uno schema atteso), la richiesta viene automaticamente escalata a un modello di livello superiore. Questo pattern combina il costo medio basso del routing semantico con una rete di sicurezza per i casi che il classificatore iniziale ha sottostimato.

5.3 Modelli verificatori ("verifier models")

Un pattern distinto, spesso usato in combinazione con i precedenti, consiste nell'impiegare un secondo modello (spesso più piccolo ed economico del modello generatore) con l'unico compito di valutare la plausibilità o correttezza dell'output prodotto, senza rigenerarlo. Questo approccio consente di intercettare errori sistematici a un costo marginale contenuto, ed è particolarmente efficace per task con criteri di correttezza verificabili in modo automatico (es. validità di uno schema, coerenza numerica, aderenza a policy).

5.4 Pipeline di inferenza multi-stadio

Nei workflow più complessi, diversi modelli vengono combinati in una pipeline esplicita: un modello economico per l'estrazione o la normalizzazione iniziale dei dati, un modello reasoning per la fase di pianificazione o decisione, e un modello economico per la formattazione finale dell'output. Questo pattern,

tipico dei workflow agentici (si veda il capitolo 6), permette di concentrare la spesa computazionale solo nella fase realmente critica del processo.

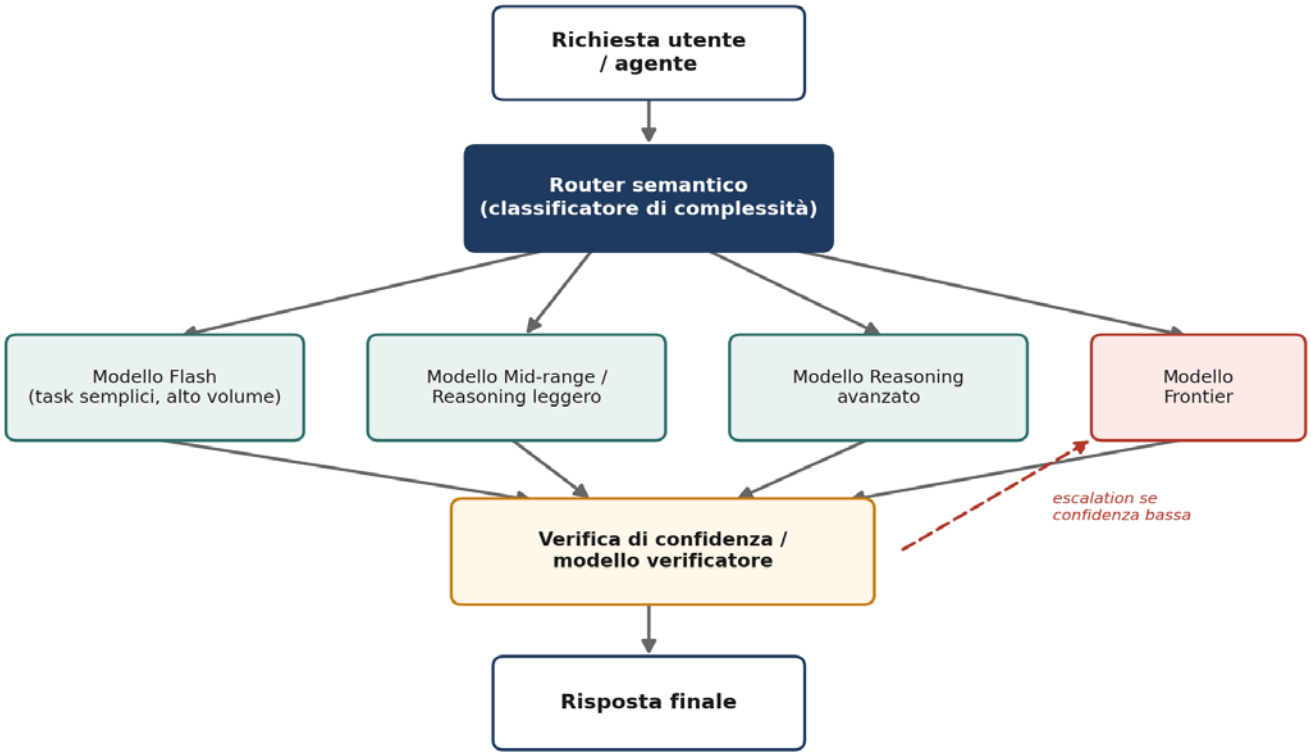


Figura 3 — Schema concettuale di un'architettura di routing con cascading e verifica di confidenza. Lo schema è illustrativo delle relazioni logiche tra i componenti, non la rappresentazione di un prodotto specifico.

> ***Nota di verifica:*** *Diverse piattaforme di terze parti offrono soluzioni preconfezionate di model routing. Non essendo in grado di verificare in modo indipendente le prestazioni dichiarate da specifici prodotti commerciali, questo documento descrive i pattern architetturali in astratto: prima di adottare una soluzione specifica, è opportuno condurre un pilot interno che misuri il tasso di escalation reale, il tasso di errore residuo e il risparmio netto sul proprio mix di task.*

> ***Esempio applicativo:*** *Alcuni fornitori di infrastruttura offrono oggi layer di routing riconducibili ai pattern descritti sopra. Ad esempio, Regolo.AI (Seeweb) descrive il proprio layer, denominato Brick, come basato su un classificatore dedicato — la combinazione di un modello ModernBERT e di uno Small Language Model — che opera una valutazione di tipo prevalentemente spaziale anziché puramente semantica. L'instradamento è predittivo, precede quindi la chiamata al modello selezionato, ed è accompagnato da un confidence score con un meccanismo di fallback verso un modello di livello superiore in caso di bassa confidenza; l'overhead di latenza dichiarato per il solo layer di routing è indicativamente di 100–200 ms. Il catalogo dei modelli instradabili risulta configurabile, comprendendo sia modelli proprietari sia open-weight e supportando anche l'instradamento verso modelli reasoning quando richiesto esplicitamente. Si tratta di dati forniti direttamente dal fornitore e non verificati in modo indipendente in questa sede; sono riportati a titolo di esempio concreto dell'applicazione dei pattern descritti, non come validazione delle prestazioni dichiarate.*

6. Workflow agentici: implicazioni di costo e affidabilità

I workflow agentici, in cui un modello pianifica ed esegue autonomamente una sequenza di azioni, tipicamente invocando strumenti esterni (ricerca, esecuzione di codice, chiamate ad API). introducono una dinamica di costo qualitativamente diversa da quella di una singola richiesta-risposta: il costo totale di un task non è più il costo di una chiamata, ma il costo cumulato di un numero di chiamate spesso non noto a priori, perché dipende da quanti passaggi il modello ritiene necessari per completare l'obiettivo.

Questo introduce due implicazioni pratiche per la strategia "good enough":

- **Non tutti i passaggi hanno lo stesso valore.** il modello che pianifica i passaggi (spesso il più critico per la qualità del risultato finale) beneficia in genere di capacità di reasoning superiori, mentre i singoli passaggi esecutivi (es. formattare una chiamata a uno strumento, riassumere un risultato intermedio) possono spesso essere delegati a modelli più economici, replicando all'interno del workflow lo stesso principio di allocazione per complessità descritto nel capitolo 5.
- **Il costo di un task agentico è per natura meno prevedibile.** senza un limite esplicito al numero di passaggi (o un budget di costo per task), un agente può, in presenza di errori a cascata o loop di ripianificazione, generare un costo molto superiore a quello previsto. È buona pratica enterprise definire un tetto massimo di iterazioni e un meccanismo di fallback verso l'intervento umano quando tale tetto viene raggiunto.

La governance dei workflow agentici (si veda anche il capitolo 8) richiede quindi, oltre al routing per singola chiamata, un monitoraggio a livello di intero task: costo medio e mediano per task completato, tasso di completamento entro il budget previsto, e frequenza di intervento umano richiesto.

7. Impatto ambientale: evidenza disponibile e cautele necessarie

La domanda energetica dei data center dedicati all'AI è oggetto di crescente attenzione da parte di enti come l'International Energy Agency (IEA), che nei suoi rapporti periodici sull'elettricità e sui data center ha più volte segnalato una traiettoria di crescita significativa della domanda energetica associata al calcolo AI a livello globale. Il messaggio qualitativo di questi rapporti è consistente: la domanda energetica dei data center AI cresce più rapidamente della media del settore elettrico, ed è concentrata in un numero relativamente piccolo di regioni e fornitori di infrastruttura.

Questa traiettoria di crescita si scontra, sul lato dell'offerta, con i vincoli strutturali discussi nel capitolo 4 (paragrafo 4.4): disponibilità di hardware, capacità della rete elettrica e opposizione locale alla costruzione di nuovi data center non sono soltanto un problema di pianificazione per i singoli operatori, ma un fattore che incide direttamente sulla velocità con cui la domanda energetica del settore può essere effettivamente soddisfatta.

A livello di singola richiesta, è tecnicamente corretto affermare che il consumo energetico dell'inference cresce con la dimensione del modello e con la lunghezza del contesto elaborato (input più eventuale fase di ragionamento interno), ma le cifre assolute in Wh per richiesta variano di ordini di grandezza a seconda del modello, dell'hardware, del livello di ottimizzazione del serving (batching, quantizzazione) e della definizione stessa di "richiesta media" usata da ciascuno studio. Per questo motivo, questo documento evita deliberatamente di riportare una cifra unica di Wh-per-prompt come se fosse un dato consolidato e universalmente applicabile.

7.1 Il costo incorporato (embodied cost) dell'hardware

Una componente spesso trascurata nelle analisi di impatto ambientale è il costo energetico e materiale "incorporato" nella produzione dell'hardware stesso (GPU, sistemi di raffreddamento, infrastruttura di rete), che si somma al consumo operativo durante l'inference. Un dimensionamento più efficiente dei modelli riduce anche la pressione sul ricambio hardware e sulla domanda di nuova capacità computazionale, un effetto meno visibile ma cumulativamente rilevante su orizzonti pluriennali.

7.2 Cosa resta valido indipendentemente dalle cifre esatte

Al netto dell'incertezza sulle cifre puntuali, tre affermazioni qualitative godono di ampio consenso nella letteratura tecnica e nei rapporti di enti come IEA e nella ricerca accademica su efficienza dei modelli:

- l'inference (l'uso quotidiano dei modelli), non il training, rappresenta oggi la quota maggioritaria del consumo energetico complessivo legato agli LLM, proprio perché ogni richiesta è moltiplicata per un volume d'uso enorme.
- a parità di task, un modello dimensionalmente più piccolo e adeguatamente specializzato consuma meno energia per unità di lavoro svolta rispetto a un modello generalista di grandi dimensioni.
- le tecniche di ottimizzazione del serving (batching, quantizzazione, caching, routing per complessità) hanno un impatto misurabile e diretto sull'efficienza energetica per richiesta, indipendentemente dal modello scelto.

> □ ***Nota metodologica:*** *Per una valutazione quantitativa rigorosa dell'impronta ambientale specifica della propria infrastruttura AI, si raccomanda di fare riferimento diretto ai rapporti più recenti di IEA (serie "Electricity" e i relativi approfondimenti su data center e AI), al Stanford AI Index (Stanford HAI, edizione più recente disponibile) e alla letteratura peer-reviewed su arXiv relativa a "LLM inference energy efficiency", verificandone la data di pubblicazione rispetto al momento dell'uso, poiché si tratta di un'area di ricerca in rapida evoluzione.*

> ***Esempio applicativo:*** *A conferma indiretta di questo limite, anche tra i fornitori di layer di routing citati in questo documento (capitoli 4-5 e 8) la misurazione sistematica dell'impronta energetica specifica dell'infrastruttura risulta, ad oggi, un tema indicato come oggetto di sviluppo futuro più che una pratica di reporting già consolidata: un'ulteriore indicazione di quanto il settore nel suo complesso resti indietro su questo fronte rispetto alla misurazione di costo e latenza.*

8. Governance enterprise: rischio, non solo risparmio

Ridurre "good enough" a una leva di taglio dei costi sarebbe un errore di framing. Nella pratica enterprise matura, la scelta del livello di capacità del modello è anzitutto una decisione di gestione del rischio, che precede logicamente quella di ottimizzazione economica.

8.1 Le dimensioni di rischio da governare

- **Privacy e data residency.** dove risiedono i dati, chi vi ha accesso durante l'elaborazione, quali obblighi normativi (settoriali o geografici) si applicano al tipo di dato trattato.
- **Compliance settoriale.** in settori regolamentati, la scelta del modello e del fornitore deve essere documentabile e giustificabile in sede di audit, incluse le motivazioni per cui un dato livello di capacità è stato ritenuto adeguato per una data categoria di task.
- **Disponibilità e continuità operativa.** dipendere da un singolo fornitore o modello per task critici espone l'organizzazione a rischi di interruzione del servizio, deprecazione di modelli o variazioni di prezzo unilaterali; un'architettura multi-modello e multi-fornitore, se ben governata, riduce questo rischio.
- **Gestione del rischio reputazionale.** un output errato prodotto da un modello economico su un task ad alto impatto (es. una comunicazione a un cliente, una decisione di credito) può generare un danno reputazionale o legale sproporzionato rispetto al risparmio ottenuto sul costo di inference.

> ***Esempio applicativo:*** *Le pratiche dichiarate variano da fornitore a fornitore e vanno sempre verificate contrattualmente prima dell'adozione. A titolo di esempio, il fornitore già citato ai capitoli 4-5 dichiara l'hosting dei dati in data center nazionali (Italia, Frosinone) e l'esclusione esplicita dei dati dei clienti dai processi di addestramento del proprio classificatore, con isolamento dei dati per singolo cliente — pratiche che, se contrattualmente vincolanti, rispondono direttamente ai criteri di privacy e data residency indicati sopra. Aspetti quali certificazioni specifiche e politiche di retention restano, in questo come in altri casi, di competenza delle funzioni commerciali/compliance del singolo fornitore e vanno richiesti esplicitamente in fase di valutazione.*

8.2 Un principio di governance operativo

Un principio pratico, adottabile come policy interna: la scelta del livello di capacità del modello per una categoria di task non dovrebbe mai essere delegata esclusivamente a una logica di minimizzazione del costo; va sempre vincolata a una soglia di rischio massimo accettabile definita a priori dal proprio ufficio legale, compliance o risk management, e solo all'interno di tale vincolo va poi ottimizzato il costo.

> ***Nota di verifica:*** *Le sezioni successive fanno riferimento a documentazione ufficiale dei fornitori per aspetti di compliance e sicurezza (es. politiche di utilizzo, retention dei dati, certificazioni). Queste politiche cambiano nel tempo: vanno sempre verificate sulla documentazione più aggiornata del fornitore specifico prima di finalizzare un accordo contrattuale.*

9. Framework decisionale: quale livello di modello per quale task

La tabella seguente sintetizza, in forma di matrice decisionale, i quattro livelli di capacità discussi in questo documento, associandoli a caratteristiche tipiche del task, al principale meccanismo di controllo del rischio raccomandato, e agli indicatori operativi da monitorare.

Livello	Task tipici	Meccanismo di controllo rischio raccomandato	KPI operativo primario
Flash / economico	Classificazione, estrazione campi, riformulazione, FAQ semplici, moderazione di base	Verificatore automatico leggero + escalation su bassa confidenza	Costo per task e tasso di escalation
Mid-range / reasoning leggero	Sintesi di documenti, prima stesura di contenuti, risposta a query strutturate ma non banali	Revisione umana a campione; verificatore su criteri di qualità definiti	Tasso di accettazione umana della prima bozza
Reasoning avanzato	Pianificazione multi-step, analisi con vincoli complessi, orchestrazione di agenti	Budget di iterazioni definito + fallback umano su superamento soglia	Tasso di completamento entro il budget previsto
Frontier / top-tier	Decisioni ad alto impatto, compiti ambigui senza precedenti, casi che hanno già fallito ai livelli inferiori	Revisione umana obbligatoria per le categorie di rischio più alte; audit trail completo	Tasso di errore residuo su casi ad alto impatto

9.1 Come usare la matrice in pratica

1. Mappare i propri task ricorrenti in una tassonomia per complessità e per rischio, separatamente: i due assi non coincidono (un task semplice può essere ad alto rischio, es. comunicazioni verso clienti regolamentati).
2. Assegnare a ciascuna combinazione complessità/rischio un livello di partenza secondo la matrice, e il relativo meccanismo di controllo.
3. Strumentare il sistema per misurare, per ogni categoria, il tasso di escalation reale e il tasso di errore residuo dopo il meccanismo di controllo.
4. Rivedere periodicamente l'assegnazione: la capacità dei modelli disponibili in ciascuna fascia evolve rapidamente, e un task oggi assegnato a un livello superiore potrebbe essere gestibile da un livello inferiore entro pochi mesi, o viceversa in caso di nuovi requisiti di compliance.

9.2 Una quinta opzione: costruire anziché scegliere

La matrice presentata sopra assume che l'organizzazione scelga tra modelli già disponibili sul mercato. Per task ad alto volume, stabili nel tempo e ben definiti, esiste una quinta opzione, discussa nel capitolo 2.4: specializzare un modello open-weight di piccole dimensioni tramite fine-tuning efficiente (LoRA) o distillazione della conoscenza da un modello di frontiera. Questa via può collocare, di fatto, un modello "flash" specializzato al livello qualitativo di un modello reasoning generalista sul task specifico per cui è stato costruito — ma sposta parte del TCO dal costo per token al costo di sviluppo e manutenzione del modello stesso (capitolo 3), e va quindi valutata con lo stesso rigore delle altre opzioni della matrice, non come alternativa "gratuita".

10. Deployment ibrido: cloud pubblico, VPC dedicata, on-premise

La scelta del modello non può essere disgiunta da quella della modalità di deployment, perché le due decisioni condividono gli stessi vincoli di rischio e costo discussi nei capitoli precedenti.

- **Cloud pubblico (API del fornitore):** massima semplicità operativa e accesso immediato alle capacità più recenti; costo variabile per token; da valutare con attenzione per dati soggetti a vincoli stringenti di data residency.
- **VPC dedicata / deployment privato gestito:** isolamento di rete e maggiori garanzie contrattuali di data handling, mantenendo comunque il modello ospitato dal fornitore; opzione intermedia per organizzazioni regolamentate che non necessitano di self-hosting completo.
- **On-premise / self-hosted:** controllo massimo su dati e infrastruttura, tipicamente riservato a modelli open-weight; costo fisso elevato, richiede competenze MLOps interne, ma elimina la dipendenza diretta da un singolo fornitore esterno per il traffico più sensibile.
- **Deployment ibrido:** instrada il traffico a bassa sensibilità e alto volume verso opzioni cloud economiche, e riserva l'on-premise o la VPC dedicata al traffico più sensibile o normato; è il pattern che nel 2026 si osserva più di frequente nelle organizzazioni enterprise con requisiti di compliance stringenti in più giurisdizioni.

> ***Nota di verifica:*** *La convenienza relativa di queste opzioni dipende da fattori specifici dell'organizzazione (volume, prevedibilità del carico, requisiti normativi geografici, maturità del team infrastrutturale) che non è possibile generalizzare in un documento di questo tipo. Si raccomanda un'analisi di TCO comparativa specifica (si veda il capitolo 3) prima di ogni decisione di deployment.*

11. Raccomandazioni operative per CIO, CTO e AI leader

5. costruire una tassonomia dei task ricorrenti per complessità e per livello di rischio, come base per qualsiasi decisione successiva di routing o procurement.
6. non trattare mai il costo come unico criterio di instradamento: ogni livello del framework decisionale (capitolo 9) va abbinato a un meccanismo esplicito di controllo del rischio, concordato con le funzioni legale, compliance e sicurezza.
7. costruire (o adottare) un layer di routing esplicito, con logging delle decisioni di instradamento e degli esiti, per poter misurare, non solo assumere, il risparmio netto e il tasso di errore residuo.
8. misurare il Total Cost of Ownership reale (capitolo 3), non solo il prezzo per token, includendo esplicitamente il costo dei retry, dell'ingegneria di manutenzione e della latenza.
9. per i workflow agentici, definire budget espliciti di iterazioni e costo per task, con fallback automatico verso l'intervento umano al superamento delle soglie.
10. evitare la dipendenza da un singolo fornitore per i task critici, mantenendo un'architettura in grado di instradare verso più fornitori o modelli, come mitigazione del rischio di continuità operativa.
11. integrare, dove disponibili, metriche di consumo energetico stimato nelle dashboard di monitoraggio dell'infrastruttura AI, anche solo come indicatore direzionale, in attesa di standard di misurazione più maturi e uniformi nel settore.
12. rivedere trimestralmente l'assegnazione task-modello, poiché sia le capacità dei modelli disponibili sia i requisiti normativi evolvono più rapidamente dei cicli di pianificazione IT tradizionali.
13. verificare sempre, prima di comunicazioni esterne o decisioni di budget, le cifre di prezzo, le percentuali di risparmio dichiarate da terze parti e i dati ambientali citati, con le fonti primarie più aggiornate disponibili al momento.

12. Conclusioni

La strategia "Not the Best, but Good Enough" resta, nella sua essenza, un principio semplice: allineare la capacità computazionale impiegata al valore e al rischio reali del task da svolgere, anziché applicare uniformemente il modello più potente disponibile. Ciò che è cambiato, e che questo documento ha cercato di riflettere, è la maturità con cui questo principio può oggi essere implementato: non più come intuizione di buon senso routing semantico, cascading, verifica di confidenza, budget agentici, governata da criteri di rischio definiti a priori e misurata con KPI operativi concreti (costo per task, tasso di escalation, tasso di errore, ma come architettura esplicita , residuo, latenza).

Per un CIO o un CTO, il messaggio operativo è che l'investimento più duraturo non è nella scelta del miglior modello del momento, destinata a essere superata nel giro di mesi, ma nella costruzione di un sistema di allocazione dei task capace di incorporare qualunque nuovo modello, di qualunque fornitore, mano a mano che diventa disponibile, mantenendo stabili i criteri di rischio e i meccanismi di controllo qualità.

13. Nota metodologica e limiti

Questa edizione è stata prodotta senza accesso a ricerca web in tempo reale. La revisione si è basata su conoscenza tecnica generale del settore, consolidata fino a gennaio 2026, e su un controllo di coerenza interna della bozza originale. I concetti architetturali presentati (routing, cascading, verifier model, TCO, governance) sono pattern consolidati e stabili nel tempo, non soggetti a rapida obsolescenza; le cifre quantitative, dove non derivabili da fonti verificabili in questa sede, sono state deliberatamente presentate come range o relazioni qualitative anziché come dati puntuali, per evitare di trasmettere falsa precisione.

Prima di pubblicare, presentare esternamente o utilizzare questo documento a supporto di decisioni di budget, si raccomanda di:

- validare con fonti primarie aggiornate (documentazione ufficiale dei fornitori, rapporti IEA più recenti, ultima edizione dello Stanford AI Index, benchmark MLCommons/MLPerf) ogni cifra quantitativa citata.
- verificare l'esistenza e le caratteristiche dichiarate di qualunque prodotto o piattaforma di terze parti citato come esempio, prima di nominarlo in un documento destinato alla pubblicazione.
- aggiornare i riferimenti a modelli, fornitori e prezzi specifici, poiché il mercato evolve con cadenza mensile o più rapida rispetto al ciclo di vita tipico di un white paper.
- far rivedere le sezioni di governance, compliance e privacy dalle funzioni legali e di risk management interne, poiché i requisiti normativi variano per settore e giurisdizione e non sono trattati in modo esaustivo in questo documento generalista.

14. Riferimenti e fonti per approfondimento

I riferimenti seguenti indicano serie di pubblicazioni e organismi riconosciuti nel settore, da consultare nella loro edizione più recente al momento dell'uso; questo documento non cita cifre specifiche tratte da essi senza indicarle esplicitamente come stime da validare.

- Stanford Institute for Human-Centered Artificial Intelligence (Stanford HAI) — AI Index Report, edizione annuale.
- International Energy Agency (IEA) — rapporti periodici sulla domanda elettrica dei data center e dell'intelligenza artificiale (serie "Electricity" e approfondimenti correlati).
- MLCommons — benchmark MLPerf Inference, per dati comparativi su efficienza ed efficacia dell'inferenza su diverse classi di hardware e modelli.
- Documentazione ufficiale dei fornitori di modelli (ad es. docs.claude.com per Anthropic, e le corrispondenti documentazioni di Google, OpenAI, Meta, Mistral) per capacità, prezzi e policy correnti.
- Letteratura accademica su arXiv relativa a model routing, cascading inference, efficienza energetica dell'inferenza e quantizzazione dei modelli, per approfondimenti tecnici sui pattern architetturali descritti nei capitoli 4-6.
- Regolo.AI (Seeweb) — documentazione tecnica del layer di routing Brick, citata nei capitoli 4, 5, 7 e 8 come esempio concreto di architettura di classificazione dedicata e delle relative pratiche dichiarate di data residency e ottimizzazione dei costi.

15. Glossario e acronimi

Il seguente glossario raccoglie, in ordine alfabetico, gli acronimi tecnici utilizzati nel documento e le organizzazioni/prodotti citati a titolo di esempio applicativo (si vedano in particolare i capitoli 4, 5, 7 e 8).

Acronimi tecnici

- **AI / IA.** Artificial Intelligence / Intelligenza Artificiale.
- **API.** Application Programming Interface: interfaccia software che consente la comunicazione programmatica tra sistemi diversi.
- **BERT.** Bidirectional Encoder Representations from Transformers: famiglia di modelli linguistici basati su architettura Transformer di tipo encoder, tipicamente impiegati per compiti di comprensione e classificazione del testo piuttosto che di generazione. ModernBERT è un'evoluzione più recente ed efficiente di questa architettura, citata nel capitolo 5 come componente di un classificatore di routing.
- **CIO.** Chief Information Officer.
- **CTO.** Chief Technology Officer.
- **GPU.** Graphics Processing Unit: processore ottimizzato per il calcolo parallelo, utilizzato per l'addestramento e l'inferenza dei modelli LLM.
- **IEA.** International Energy Agency (Agenzia Internazionale dell'Energia).
- **KPI.** Key Performance Indicator: indicatore chiave di prestazione.
- **LLM.** Large Language Model: modello linguistico di grandi dimensioni.
- **MLCommons / MLPerf.** Consorzio indipendente che pubblica benchmark standardizzati (MLPerf) per confrontare le prestazioni di hardware e software di machine learning.
- **SLA.** Service Level Agreement: accordo contrattuale sui livelli di servizio garantiti.
- **SLM.** Small Language Model: modello linguistico di dimensioni ridotte rispetto agli LLM, spesso utilizzato per compiti specifici o come componente di classificazione (si veda capitolo 5).
- **TCO.** Total Cost of Ownership: costo totale di possesso di una soluzione, comprensivo di costi diretti e indiretti (si veda capitolo 3).
- **VPC.** Virtual Private Cloud.

Organizzazioni e prodotti citati

- **Seeweb.** Provider italiano di infrastruttura cloud e data center. È il soggetto societario a cui fa capo il brand Regolo.AI, citato nei capitoli 4, 5, 7 e 8 come esempio applicativo dei pattern architetturali descritti nel documento.
- **Regolo.AI.** Brand dedicato all'intelligenza artificiale di Seeweb, che sviluppa Brick, un layer di model routing.
- **Brick.** Nome del layer di routing sviluppato da Regolo.AI (Seeweb), basato su un classificatore dedicato — la combinazione di un modello ModernBERT e di uno Small Language Model — per l'instradamento predittivo delle richieste verso il modello più appropriato (si veda capitolo 5).