AI ATLAS FORGE

Guida GB10 — doppio nodo · Due sistemi collegati, 2 × 128 GB: schede di progetto, modelli e procedure documentate

Estratto dell'edizione Novembre 2026 (ED-2026-11) · Riferimento dei contenuti: 5 novembre 2026

a cura del Dipartimento di Informatica e IA — Electronic Solution

Classificazione dell'edizione: paper tecnico-scientifico destinato alla diffusione pubblica gratuita (candidata; approvazione del titolare aperta); revisione R02 (ED-2026-11). Procedure documentate, non eseguite: nessuna prova su hardware per questa edizione. Per questa edizione non è stata emessa alcuna attestazione interna: la revisione, svolta da sistemi AI, è documentata nel fascicolo dell'edizione.

Avvertenza

Questa guida riunisce le parti dell'edizione Novembre 2026 (ED-2026-11) di AI ATLAS FORGE, con riferimento dei contenuti al 5 novembre 2026, dedicate alla configurazione con **due sistemi GB10 collegati, 2 × 128 GB**. Il testo deriva da quello dell'edizione: cambiano l'impaginazione e il livello tipografico dei titoli delle sezioni 1.5.15 e 1.5.16, raccolte sotto due titoli di raggruppamento; inoltre alcune precisazioni proprie dell'estratto, introdotte dalla revisione R02 dell'edizione di ottobre 2026 e riesaminate nella revisione R02 di questa edizione, fanno sì che una decina di paragrafi della sezione 1.5.15 e delle Appendici B e C possa differire nella formulazione dal paper (differenze registrate nel fascicolo dell'edizione). Frontespizio, Avvertenza, titoli di raggruppamento e note «Nota della guida» sono apparato editoriale della guida.

La guida contiene:

- sezione 1.5.15 completa: introduzione e sottosezioni 1.5.15.1–1.5.15.5 (Schede A e B), con le Figure 1.4 e 1.5 e le Tabelle 1.40–1.44;
- sezione 1.5.16: introduzione, sottosezione 1.5.16.6 limitatamente all'introduzione, alle Tabelle 1.50 e 1.51 e alle conclusioni, e sottosezione 1.5.16.7;
- Appendice B, procedura documentata per un sistema GB10 a nodo singolo, con le Tabelle B.1 e B.2;
- Appendice C, procedura documentata per due sistemi GB10 collegati, con le Tabelle C.1 e C.2;
- la Bibliografia della guida, con le 123 voci citate e la numerazione della bibliografia dell'edizione.

Le parti a nodo singolo, Scheda A e Appendice B, sono incluse perché la Scheda B e l'Appendice C vi rinviano: convenzioni delle etichette (Sezione B.1), percorsi di servizio per le repliche (Sezioni B.6–B.8) e schemi di verifica (Blocchi B.7-2 e B.7-3). Due sistemi collegati restano due nodi con memorie distinte: nessuna procedura crea una memoria coerente da 256 GB.

Per la lettura:

- nessun comando è stato eseguito e nessuna prova, misura o benchmark è stata svolta su hardware GB10 per questa edizione. Le procedure riportano la documentazione ufficiale consultata il 25 ottobre 2026 e riverificata con lo stato al 5 novembre 2026; gli adattamenti editoriali sono motivati ma non provati; budget di memoria e limiti di prestazione sono stime editoriali, con formula e ipotesi; le prestazioni citate sono misure di terzi o dichiarazioni del produttore, con le condizioni indicate dalle fonti;
- versioni, immagini dei container e playbook cambiano nel tempo: prima dell'uso vanno rilette le pagine citate e lette sul nodo le versioni installate (Sezione B.2);
- i numeri di sezioni, tabelle, figure, appendici e voci bibliografiche sono quelli dell'edizione. I rimandi a parti non incluse, per esempio alle sezioni 1.5.14, 2.4 o 5.4.9, si riferiscono all'edizione completa, che tratta anche il dimensionamento della memoria da cui derivano le stime dei pesi e dei margini citate nelle Schede A e B e nelle Tabelle 1.42, 1.44, 1.50, 1.51 e B.1 (esempi svolti della sezione 1.5.14.4), il censimento dell'hardware (sezioni 1.5.12–1.5.13), la matrice modello × piattaforma × runtime (sezione 1.5.16.4) e l'architettura dell'AI privata (sezione 5.4.9); le schede complete dell'hardware e dei modelli sono negli Allegati 1 e 2 dell'edizione, e i calcoli di memoria richiamati nella sezione C.6 (ME08 e ME36) nell'Allegato 2;
- sigle usate in questa guida senza scioglimento nel testo, per la maggior parte definite nel Glossario dell'edizione completa: TTFT (Time To First Token), ITL (Inter-Token Latency), KV cache (key-value cache), MoE (Mixture of Experts), MTP (Multi-Token Prediction), UMA (Unified Memory Architecture), QSFP (Quad Small Form-factor Pluggable), RDMA (Remote Direct Memory Access), RoCE (RDMA over Converged Ethernet), HCA (Host Channel Adapter), NCCL (NVIDIA Collective Communications Library), TP e PP (tensor e pipeline parallelism), MTU (Maximum Transmission Unit), TDP (Thermal Design Power), OEM (Original Equipment Manufacturer) e OTA (over-the-air);
- la guida non costituisce certificazione, approvazione del produttore né garanzia di funzionamento, e per questa edizione non è stata emessa alcuna attestazione interna: la revisione R02 dell'edizione ED-2026-11, svolta da sistemi AI su incarico del Dipartimento e documentata nel fascicolo dell'edizione, è candidata alla diffusione con approvazione del titolare aperta; l'esecuzione delle procedure resta sotto la responsabilità di chi amministra i sistemi. La guida presuppone familiarità con l'amministrazione di sistemi Linux, con Docker, SSH e reti IP e con la lettura di script di shell; il percorso a doppio nodo documentato da NVIDIA usa componenti comunitari mantenuti fuori da NVIDIA, cioè l'immagine eugr/spark-vllm-b12x con build notturna e lo script setup-cluster.sh del progetto spark-vllm-docker (sezioni 1.5.15.4, C.3 e C.6), da esaminare prima dell'esecuzione [297], [302], [326].

Sommario

Avvertenza 2
Schede di progetto: due sistemi GB10 collegati, 2 × 128 GB 6
1.5.15 Configurazioni GB10: nodo singolo e doppio nodo 6
1.5.15.1 Superchip, sistema e memoria: cosa è condiviso e cosa no 6
1.5.15.2 Stack software verificato al 5 novembre 2026 8
1.5.15.3 Scheda A — un sistema GB10 da 128 GB 10
1.5.15.4 Scheda B — due sistemi GB10 collegati, 2 × 128 GB 13
1.5.15.5 Colli di bottiglia e piano di misura 17
Modelli e verifiche: raccomandazioni per due sistemi GB10 18
1.5.16 Modelli per l'AI privata: selezione per scenario, matrice di compatibilità e raccomandazioni 18
1.5.16.6 Raccomandazioni per configurazione 18
1.5.16.7 Verifiche prima dell'adozione 21
Appendice B — Procedura documentata: sistema GB10 a nodo singolo (128 GB) 22
B.1 Convenzioni, prerequisiti e perimetro 22
B.2 Inventario delle versioni 22
B.3 Aggiornamento di sistema operativo, driver e firmware 23
B.4 Runtime dei container e accesso alla GPU 24
B.5 Memoria disponibile, riserve e budget 24
B.6 Percorso 1 — Ollama con Open WebUI 26
B.7 Percorso 2 — vLLM con API compatibile OpenAI 27
B.8 Percorso 3 — llama.cpp con modelli GGUF 30
B.9 Percorso 4 — SGLang 32
B.10 Monitoraggio 32
B.11 Verifica finale 32
B.12 Ripristino 33
B.13 Limiti della procedura 34
Appendice C — Procedura documentata: due sistemi GB10 collegati (2 × 128 GB) 35
C.1 Topologia, ruoli e prerequisiti 35
C.2 Cablaggio 35
C.3 Configurazione della rete ConnectX-7 36
C.4 Verifica del collegamento, di RoCE e di SSH 37
C.5 Verifica della banda con NCCL 38
C.6 vLLM su due nodi in tensor parallelism 39
C.7 Repliche indipendenti e separazione dei servizi 42
C.8 Verifica finale 43
C.9 Ripristino 43
C.10 Limiti della procedura 44
Bibliografia della guida 45

Elenco delle figure

Figura 1.4 — Sistema GB10 a nodo singolo: memoria unificata e stack di servizio 8
Figura 1.5 — Due sistemi GB10 collegati: memorie distinte, collegamento diretto e ripartizione esplicita 16

Elenco delle tabelle

Tabella 1.40 — GB10: grandezze di riferimento a nodo singolo e a doppio nodo. 7
Tabella 1.41 — Stack software e runtime documentati per DGX Spark al 5/11/2026. 9
Tabella 1.42 — Scheda A, 1 × 128 GB: profili, configurazioni documentate e budget di memoria stimato. 11
Tabella 1.43 — Strategie su due nodi GB10: obiettivi, memoria, comunicazione e stato documentale. 14
Tabella 1.44 — Scheda B, 2 × 128 GB: profili e strategie consigliate. 15
Tabella 1.50 — Raccomandazioni per GB10 a nodo singolo (1×128 GB). 18
Tabella 1.51 — Raccomandazioni per due sistemi GB10 (2×128 GB). 19
Tabella B.1 — Foglio di calcolo del budget a nodo singolo (stima). 25
Tabella B.2 — Verifiche a nodo singolo: comando ed esito atteso. 32
Tabella C.1 — Interfacce ConnectX-7 per porta QSFP, identiche su ogni DGX Spark. 36
Tabella C.2 — Verifiche a doppio nodo: comando ed esito atteso. 43

Schede di progetto: due sistemi GB10 collegati, 2 × 128 GB

Estratto della sezione 1.5.15 dell'edizione.

1.5.15 Configurazioni GB10: nodo singolo e doppio nodo

GB10 è il superchip Grace Blackwell; NVIDIA DGX Spark e i prodotti OEM sono sistemi che lo integrano. Questa sottosezione traduce la documentazione ufficiale consultata il 25 ottobre 2026 e riverificata il 5 novembre 2026 — guida di porting, guida utente, note di rilascio e playbook NVIDIA — in due schede di progetto: la **Scheda A** per un sistema con 128 GB e la **Scheda B** per due sistemi collegati, 2 × 128 GB. Le procedure complete sono nelle Appendici B e C. Le versioni software indicate da NVIDIA si riferiscono ai sistemi DGX Spark Founders Edition; per i sistemi di altri produttori le procedure di aggiornamento possono differire. [227], [303] Nessuna prova è stata eseguita su hardware: i budget di memoria sono stime editoriali costruite con la formula della sezione 1.5.1 e del dimensionamento della sezione 1.5.14, mentre le prestazioni riportate sono misure di terzi o dichiarazioni del produttore, con le rispettive condizioni.

1.5.15.1 Superchip, sistema e memoria: cosa è condiviso e cosa no

La Figura 1.4 riassume memoria e stack di servizio di un nodo. La guida di porting descrive un SoC (system-on-chip) Arm con GPU Blackwell integrata che condivide con la CPU 128 GB di LPDDR5X, con interfaccia a 256 bit e 273 GB/s. L'architettura di memoria è unificata e dinamica: CPU e GPU accedono allo stesso spazio fisico «without a fixed carve-out». La CPU ha 20 core ARMv9.2 organizzati in due cluster, ciascuno con cinque Cortex-X925 e cinque Cortex-A725. [233] La scheda hardware della guida utente indica 6.144 CUDA core, fino a 1.000 TOPS per l'inferenza e fino a 1 PFLOP FP4 con sparsità [304]: sono picchi teorici del produttore, non prestazioni applicative. La stessa scheda riporta 4.266 MHz e, nella sezione sulle prestazioni, 16 canali LPDDR5X 8533: i due valori sono compatibili, perché la memoria trasferisce due dati per ciclo, e 8.533 MT/s × 256 bit / 8 = 273,06 GB/s, in accordo con la banda dichiarata (calcolo editoriale). [304]

I 128 GB nominali non sono integralmente disponibili per pesi e KV cache. Il firmware UEFI riserva memoria al display: 2 GB per default, mentre con l'impostazione Auto o 0,5 GB si rendono disponibili fino a 1,5 GB in più per i carichi headless. [305] Le note di rilascio documentano una riserva regolabile, di 2 o 4 GB, da luglio 2026 e, nella release di ottobre 2026 (DGX OS 7.6.0 nella tabella delle versioni correnti), un'opzione UEFI che adatta automaticamente la riserva all'uso del display, con la quale i sistemi senza display collegato possono scendere al minimo di 512 MB. [303] Sistema operativo, servizi e processi utente non hanno una riserva quantificata nella documentazione. Il sistema può recuperare memoria liberando la cache di pagina e spostando pagine nello swap, ma l'API `cudaMemGetInfo` non conteggia questa memoria recuperabile e può riportare meno memoria di quella effettivamente allocabile: NVIDIA indica di stimare la disponibilità da `/proc/meminfo` (MemAvailable, SwapFree e, se configurate, le huge page). [305], [306] Anche `nvidia-smi` mostra «Memory-Usage: Not Supported», comportamento atteso perché la GPU integrata non dispone di un framebuffer dedicato. [306] Le fonti scrivono «128 GB» senza distinguere fra unità decimali e binarie: la capacità visibile al sistema va letta sul nodo, non dedotta dalla targa.

Due sistemi collegati restano due nodi. Ciascuno conserva la propria memoria unificata e coerente, mentre il collegamento ConnectX-7 unisce due spazi di indirizzamento distinti: non nasce una memoria coerente da 256 GB, e un modello o un carico distribuito richiede software compatibile e una ripartizione esplicita. Ogni DGX Spark dispone di due porte QSFP, fino a 200 Gb/s per porta e configurabili soltanto in Ethernet; ogni porta espone due funzioni PCIe, quindi due interfacce Ethernet e due dispositivi RoCE. [307] Un collegamento a 200 Gb/s corrisponde a 25 GB/s lordi, circa un undicesimo dei 273 GB/s della

memoria locale (273/25 ≈ 10,9; calcolo editoriale), prima di overhead e latenza, che la documentazione consultata non quantifica. Il confronto con i cluster su Ethernet a 5 Gbit/s è nella sezione 1.5.3.

La presenza di RoCE non implica il supporto di GPUDirect RDMA. La pagina CUDA della guida di porting precisa che, in DGX Spark, la memoria restituita dagli allocatori di memoria pinned del dispositivo, come `cudaMalloc`, non è accessibile in modo coerente né dal complesso CPU né dai dispositivi di I/O PCI Express, come la NIC; ne conclude: «Hence the GPUDirect RDMA technology is not supported». I meccanismi di I/O diretto che ne dipendono — nvidia-peermem per DOCA-Host, dma-buf e GDRCopy — «do not work». Le applicazioni conformi devono interrogare a runtime gli attributi `CU_DEVICE_ATTRIBUTE_GPU_DIRECT_RDMA_SUPPORTED` e `CU_DEVICE_ATTRIBUTE_DMA_BUF_SUPPORT` e prevedere un percorso alternativo; per le applicazioni ib verbs la guida indica di allocare i buffer di comunicazione con `cudaHostAlloc` e di registrarli con `ibv_reg_mr`. [308] Nella tabella delle dipendenze DOCA risulta non supportato. [309] Il trasporto RDMA su RoCE resta disponibile ed è quello che la ricetta vLLM a due nodi seleziona per NCCL (`NCCL_IB_HCA` sui dispositivi RoCE, `NCCL_IB_DISABLE=0`). [297] La stessa ricetta imposta anche VLLM_ENABLE_ROCE_ALLREDUCE=1 e VLLM_ROCE_ALLREDUCE_MAX_SIZE=2MB, variabili che il playbook non descrive: quale implementazione esegua le all-reduce del decode, NCCL o un percorso specifico dell'immagine comunitaria, va verificato nei log di avvio. [297] Le pagine consultate non trattano GPUDirect Storage. Ne segue che le ricette per server con GPU discrete basate su peermem, dma-buf o GDRCopy non si trasferiscono a GB10 senza verifica. La Tabella 1.40 riepiloga le grandezze di riferimento a nodo singolo e a doppio nodo.

Tabella 1.40 — GB10: grandezze di riferimento a nodo singolo e a doppio nodo.

Grandezza	Nodo singolo	Doppio nodo	Fonti
Memoria	128 GB LPDDR5X unificata e coerente; UMA senza carve-out fisso	2 × 128 GB in due spazi distinti; nessuna memoria coerente comune	[172], [233]
Banda	273 GB/s, interfaccia a 256 bit	273 GB/s per nodo; collegamento fino a 200 Gb/s per porta (25 GB/s lordi)	[233], [307]
Riserve documentate	Display: 2 GB per default; dalla release di ottobre 2026 Auto, 0,5, 1, 2 o 4 GB, con Auto ridotto al minimo di 512 MB senza display; SO e servizi non quantificati	Le stesse riserve su ciascun nodo	[303], [305]
Rete	1 × RJ-45 10 GbE; ConnectX-7 con 2 porte QSFP, solo Ethernet; Wi-Fi 7	Cavo QSFP diretto (2–3 sistemi) o switch (fino a 4)	[304], [307]
GPUDirect RDMA	Non supportato; nvidia-peermem, dma-buf e GDRCopy non funzionano	RoCE disponibile; buffer con `cudaHostAlloc` e `ibv_reg_mr`	[297], [308]
Lettura della memoria	`nvidia-smi`: «Memory-Usage: Not Supported»; `cudaMemGetInfo` non conta la memoria recuperabile	Da leggere su ciascun nodo	[305], [306]

Fonti: [172], [233], [297], [303]–[308]. Dati consultati il 25/10/2026 e riverificati il 5/11/2026 (guida utente aggiornata il 4/11/2026 senza variazioni nei valori citati). Note: 25 GB/s = 200 Gb/s / 8, calcolo editoriale prima degli overhead; i picchi FP4 e TOPS sono dichiarazioni del produttore.

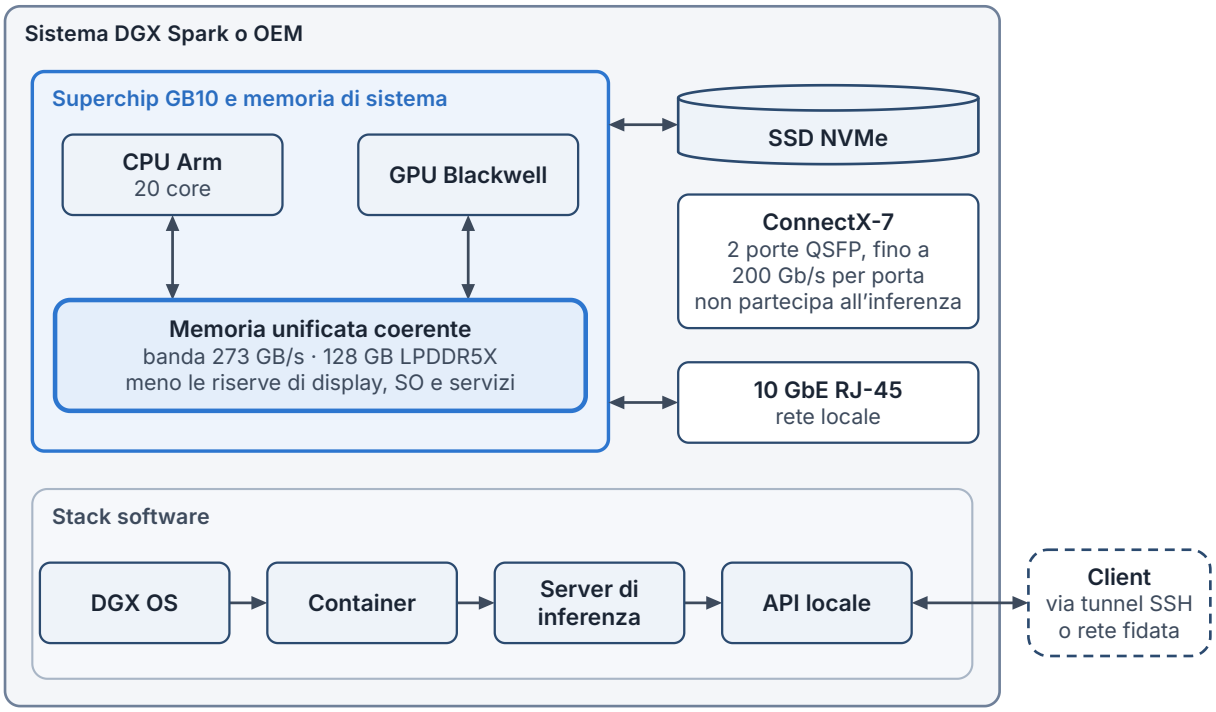


Figura 1.4 — Sistema GB10 a nodo singolo: memoria unificata e stack di servizio. CPU Arm e GPU Blackwell del superchip accedono alla stessa memoria LPDDR5X da 128 GB, dalla quale si sottraggono le riserve di display, sistema operativo e servizi; lo stack procede da DGX OS ai container, al server di inferenza e all'API locale, raggiunta dai client tramite tunnel SSH o rete fidata. A nodo singolo la ConnectX-7 non partecipa all'inferenza; le interfacce Wi-Fi e Bluetooth integrate, non rappresentate, vanno disattivate o escluse nelle installazioni isolate (sezione 1.5.11.1). Tipo: schema concettuale. Fonti: [233], [295], [303]–[305], [307], [310].

1.5.15.2 Stack software verificato al 5 novembre 2026

Le note di rilascio, aggiornate il 4 novembre 2026, riportano per i sistemi Founders Edition questi componenti correnti, invariati rispetto alla versione del 21 ottobre 2026: DGX OS 7.6.0, driver GPU 580.178.04, CUDA Toolkit 13.0.3, TensorRT 10.14.1, kernel Canonical 7.0.0-1019, UEFI 1.111.17, embedded controller 3.13.7, USB PD (USB Power Delivery) 0.5.25, TPM (Trusted Platform Module) 7.516.1 e firmware SoC 2.158.15. [303] La pagina sullo stack software della guida di porting descrive invece la configurazione di lancio: Ubuntu 24.04 in immagine server con pacchetti desktop, kernel Linux 6.11, driver «R580.GA» e CUDA Toolkit 13.0. [311] Le due fonti non sono in conflitto: la guida documenta l'architettura dello stack, le note di rilascio la versione corrente. La cronologia, indicata soltanto per mese, comprende fra l'altro il kernel HWE 6.14 a novembre 2025, la correzione di una regressione di prestazioni fra più sistemi collegati dopo l'aggiornamento a DGX OS 7.4.0 a febbraio 2026, NCCL 2.30u1 con il supporto all'anello di tre sistemi a giugno 2026 e, a ottobre 2026, il kernel Linux 7.0, la riserva dinamica del display, il supporto dei sistemi da 64 GB, cloud-init nativo, l'assistente Wi-Fi della DGX Dashboard e le «DGX Spark Skills», risorse per agenti distribuite tramite NVIDIA Sync (OpenCode) e indicate come disponibili da fine ottobre. Dalla release di giugno 2026 gli aggiornamenti OTA (over-the-air) non vengono installati per default durante la configurazione iniziale. [303] Un sistema appena consegnato può quindi non essere allineato alla tabella corrente: le versioni vanno lette sul nodo prima di applicare una ricetta.

NVIDIA Container Toolkit è preinstallato e configurato, ma la pagina non ne indica la versione, che si legge con `nvidia-ctk --version`; la tabella delle dipendenze fissa come minimi Container Toolkit 1.17.7, cuDNN 9.11, NCCL 2.28.9-1, TensorRT 10.14.1 e TensorRT-LLM 1.2, e indica come non supportati, fra gli altri, DOCA, HPC SDK e TensorRT for RTX. [309], [310] Per NCCL coesistono quindi tre riferimenti distinti: il minimo della guida, la versione 2.30u1 della release di giugno 2026 e la 2.31.2-1 che il playbook NCCL, aggiornato il 14 ottobre 2026, compila da sorgente. [303], [309], [312]

Il container NGC (NVIDIA GPU Cloud, il catalogo di software di NVIDIA) di vLLM ha come ultimo tag 26.10-py3, aggiornato il 29 ottobre 2026, che sostituisce il tag 26.10.pre1-py3 del 16 ottobre; l'ultima release documentata dalle note è ora la 26.10 e la pagina del catalogo dichiara il supporto multi-

architettura senza elencare le architetture; i metadati pubblici dell'API NGC indicano per il tag 26.10-py3 le varianti linux/amd64 e linux/arm64 [1071], che da sole non provano il supporto del GB10 (sm_121). [313] Le note della 26.10, aggiornate il 30 ottobre 2026, indicano vLLM 0.31.0, CUDA 13.4.3 e PyTorch 2.15.0a0, con cache precompilate di kernel, compilazione e autotuning per le configurazioni Qwen3.8 Flash-Next (A4/A16) e 27B su DGX Spark e RTX PRO 6000 [1127]; quelle della 26.09 indicavano vLLM 0.29.0, CUDA 13.4.1 e PyTorch 2.14.0a0 [314]; entrambe le pagine contengono però un collegamento alla documentazione di vLLM v0.11.0. È un rinvio documentale incoerente, che non dimostra una versione diversa del binario: la versione effettiva va letta nel container prima dell'uso. Fra i problemi noti figura che `vllm serve` usa di fatto `--gpu-memory-utilization` vicino a 1,0, con errori di memoria esaurita su memoria unificata come quella di DGX Spark: il rimedio indicato è 0,7. Su DGX Spark i modelli Nemotron Nano V3 e Super V3 in NVFP4 richiedono inoltre `--max-num-seqs 4`; i due avvisi sono ripetuti invariati nelle note della 26.10. [314], [1127] Il container segue a distanza il progetto upstream, che ha pubblicato vLLM v0.32.0 il 18 ottobre 2026 e la release v0.33.0 il 2 novembre 2026 (oggetto di release associato al tag v0.33.0rc3, non contrassegnato come pre-release), con wheel e immagini pubblicate solo per CUDA 13.0 e con una voce, riferita a DGX Spark, sugli embedding PLE di Qwen3.8-Flash-Next-NVFP4 impacchettati in NVFP4 («packed NVFP4 PLE embeddings») per far entrare il modello in un solo DGX Spark. [315], [1128] Le note del container rimandano alle regole generali CUDA senza indicare un minimo specifico per questa immagine. Le Release Notes CUDA associano il Toolkit 13.4 al ramo R615 e, nella versione corrente al 5 novembre 2026 (Toolkit 13.5), le nuove funzionalità al ramo R620, mentre la compatibilità fra versioni minori CUDA 13.x ammette driver almeno R580 con funzionalità limitate: nuove funzioni, codice PTX (Parallel Thread Execution, la rappresentazione intermedia dei kernel CUDA) e kernel possono richiedere un driver più recente. Questa regola non costituisce collaudo dei container 26.09 e 26.10 su GB10 con driver 580.178.04; la combinazione resta da provare su banco. [1026], [1027], [1129] La ricetta vLLM a due nodi usa invece l'immagine comunitaria `eugr/spark-vllm-b12x:nightly-20261015`, con backend B12X secondo la descrizione del playbook. [297] Il playbook vLLM a nodo singolo, nella scheda Intermediate riletta il 5/11/2026, abbina ancora i checkpoint ai container per taglio di memoria senza recepire il tag finale di NGC: nvcr.io/nvidia/vllm:26.10.pre1-py3 per nvidia/Qwen3.8-27B-NVFP4 sui sistemi da 128 GB, vllm/vllm-openai:v0.28.0 per lo stesso checkpoint sui sistemi da 64 GB e immagini comunitarie per Nemotron 3.5 Bolt (eugr/spark-vllm:nightly-20261016, 64 GB) e Qwen3.8 Flash-Next (eugr/spark-vllm-b12x:nightly-20261015, 128 GB) [1120].

Le pagine della guida consultate non trattano wheel aarch64 né immagini multi-architettura. La compatibilità ARM64 va quindi verificata componente per componente: manifesto multi-architettura delle immagini, disponibilità di wheel per aarch64 con CUDA 13, oppure compilazione da sorgente con destinazione esplicita. Il manifesto si legge prima del download, per esempio con docker manifest inspect <immagine>, che per le immagini multi-architettura elenca le piattaforme disponibili; la documentazione Docker classifica il comando come sperimentale [1119] (adattamento editoriale). NCCL si compila con `-gencode=arch=compute_121,code=sm_121` [316], llama.cpp con `CMAKE_CUDA_ARCHITECTURES=121a-real` [298]; per il codice CPU la guida indica `-mcpu=gb10` con LLVM 21 o GCC 15 e successivi. [305] La Tabella 1.41 riepiloga versioni e stato dei runtime.

Tabella 1.41 — Stack software e runtime documentati per DGX Spark al 5/11/2026.

Componente	Versione o stato	Data della fonte	Note	Fonti
DGX OS, kernel, driver, CUDA	7.6.0; kernel 7.0.0-1019; driver 580.178.04; CUDA 13.0.3	04/11/2026	Solo Founders Edition; al lancio Ubuntu 24.04, kernel 6.11, R580, CUDA 13.0	[303], [311]
Firmware	UEFI 1.111.17; EC 3.13.7; SoC 2.158.15; TPM 7.516.1; USB PD 0.5.25	04/11/2026 ([303]); 21/10/2026 ([311])	Firmware ConnectX-7 e MLNX_OFED senza versione nelle pagine lette	[303], [311]

Componente	Versione o stato	Data della fonte	Note	Fonti
Container Toolkit	Preinstallato; minimo 1.17.7	21/10/2026 ([309]); 04/11/2026 ([310])	Versione installata da leggere con `nvidia-ctk --version`	[309], [310]
NCCL	Minimo 2.28.9-1; 2.30u1 (release di giugno 2026); 2.31.2-1 nel playbook	14/10–04/11/2026	Il playbook compila per sm_121	[303], [309], [312]
vLLM	NGC 26.10-py3 (vLLM 0.31.0, CUDA 13.4.3), che sostituisce 26.10.pre1 nel catalogo; 26.09 (vLLM 0.29.0, CUDA 13.4.1); nel playbook a nodo singolo ancora 26.10.pre1-py3 (128 GB) e vllm/vllm-openai:v0.28.0 (64 GB); immagine comunitaria eugr/spark-vllm-b12x:nightly-20261015 per due nodi; eugr/spark-vllm:nightly-20261002 nel Model Launcher di NVIDIA Sync (beta)	19–30/10/2026	`--gpu-memory-utilization 0.7` su memoria unificata (avviso ripetuto nelle note 26.10)	[297], [302], [313], [314], [1120], [1127], [1130]
SGLang	Container «pre-built CUDA 13», tag non indicato in panoramica	31/07/2026	Multi-nodo non indicato; NVFP4 con `--quantization modelopt_fp4`	[317]
llama.cpp	Build da sorgente, nessun tag fissato	09/10/2026	`CMAKE_CUDA_ARCHITECTURES=121a-real`; build con RPC	[298], [318]
Ollama con Open WebUI	Immagine `ghcr.io/open-webui/open-webui:ollama`	14/10/2026	Porta 8080; tag senza numero di versione	[319], [320]
TensorRT-LLM	Minimo 1.2; nessun playbook dedicato al serving nel catalogo	05/11/2026	Serving a nodo singolo con trtllm-serve nel container nvcr.io/nvidia/tensorrt-llm/release:spark-single-gpu-dev del playbook di quantizzazione NVFP4 [1072]; nessun esempio a più nodi nel catalogo (il repository dei playbook ne contiene uno: sezione 1.5.15.4)	[309], [321]

Fonti: [297], [298], [302], [303], [309]–[314], [317]–[321], [1072], [1120], [1127], [1130], [1128]. Dati consultati il 25/10/2026 e riverificati il 5/11/2026. Note: le date sono quelle di aggiornamento delle pagine, salvo per TensorRT-LLM, per cui è indicata la data di consultazione del catalogo; la versione installata su un nodo può differire dalla tabella fino all'applicazione degli aggiornamenti.

1.5.15.3 Scheda A — un sistema GB10 da 128 GB

Prerequisiti. Sistema aggiornato e versioni lette sul nodo; alimentatore esterno da 240 W fornito con il sistema, il cui uso è richiesto per prestazioni ottimali: un alimentatore diverso o di potenza inferiore può ridurre le prestazioni, impedire l'avvio o causare spegnimenti imprevisti. [304], [306] Docker utilizzabile dall'utente, con l'appartenenza al gruppo docker facoltativa in alternativa a sudo (il playbook vLLM avverte che il gruppo concede di fatto accesso root senza password); Hugging Face CLI (command-line interface, interfaccia a riga di comando) per i checkpoint con accesso controllato; accesso SSH per l'uso remoto. [294], [1120], [310], [317]

Formati. I playbook usano checkpoint NVFP4 pubblicati da NVIDIA, come `nvidia/Qwen3.8-27B-NVFP4`, e per SGLang l'opzione `--quantization modelopt_fp4`; FP8 compare fra i formati validati con SGLang e come formato della KV cache nella ricetta vLLM a due nodi; MXFP4 è il formato nativo di gpt-oss; GGUF è il formato di llama.cpp e Ollama, per esempio la variante UD-Q4_K_XL di una conversione di terzi. [4], [5], [297], [302], [317], [318] Il costo per valore dei formati a blocchi è nella sezione 1.5.10.2: 4,5 + 32/N bit per NVFP4 e 4,25 bit per MXFP4. [221], [222] Provenienza e integrità delle conversioni vanno controllate come indicato nella sezione 1.5.9, licenze e condizioni d'uso dei pesi come nella sezione 1.3.4.

Budget di memoria. Si applica la scomposizione

$$M_{\text{tot}} \approx M_W + M_{\text{meta}} + M_{\text{KV}} + M_{\text{act}} + M_{\text{rt/SO}} + M_{\text{comm}} + M_{\text{margine}}, \qquad M_W \approx \frac{N_{\text{tot}}\, b_W}{8}.$$

Su memoria unificata tutti i termini attingono allo stesso insieme fisico, compresi i processi che girano sulla CPU. In vLLM `--gpu-memory-utilization` è la frazione della memoria della GPU assegnata all'executor, entro la quale pesi, lavoro del runtime e KV cache si contendono lo spazio. [295] Le ricette NVIDIA a due nodi usano 0,8 sui sistemi da 64 GB, insieme a `--kv-cache-memory-bytes 3G`, e 0,87 su quelli da 128 GB; il rimedio del container NGC è 0,7. [297], [314] Su 128 GB nominali, $u = 0{,}87$ assegna circa 111,4 GB al runtime e lascia 16,6 GB a sistema operativo, display e altri processi; $u = 0{,}7$ assegna 89,6 GB e ne lascia 38,4. Sono stime: la frazione si applica alla memoria totale riportata dal driver, che va letta sul nodo. Su questa base il dichiarato «fino a 200 miliardi di parametri» per unità [172] corrisponde a 100 GB di pesi a 4 bit esatti, con 11,4 GB residui entro $u = 0{,}87$, mentre in NVFP4 servono $200{\cdot}10^9 \times 4{,}5/8$ = 112,5 GB, oltre il budget: il valore è un limite di capienza, non un punto di esercizio con contesto utile (calcolo editoriale).

Contesto, KV cache e concorrenza. Il limite di contesto vale per singola richiesta. La guida vLLM del playbook avverte che un tetto di 262.144 token con `--max-num-seqs 8` non garantisce otto richieste di lunghezza piena contemporanee e rimanda alla stima della capacità della KV cache stampata all'avvio. [295] llama.cpp tenta per default di contenere l'intero contesto del modello servendo quattro richieste concorrenti e adatta i parametri se necessario; per coding e agenti il playbook indica almeno 32.768 token di contesto, preferibilmente 100.000 o più. [318] Su memoria unificata spostare layer o KV cache nella «memoria della CPU» non aggiunge capacità, perché la memoria fisica è la stessa (inferenza dalla UMA [233]); lo swap e lo scaricamento su NVMe, per esempio della KV cache con LMCache (sezione 1.5.5), scambiano capacità con latenza: le fonti consultate citano lo swap su GB10 solo come area verso cui il sistema può spostare pagine per liberare memoria [305], [306] e non documentano l'uso di swap o NVMe per estendere pesi o KV cache.

Profili. La Tabella 1.42 associa a ciascun profilo una configurazione documentata su DGX Spark e un budget stimato. Per l'**interazione rapida** contano il decode a basso batch e il TTFT: le configurazioni documentate sono Ollama con Open WebUI e gpt-oss:20b [320] e SGLang con Llama-3.1-8B-Instruct in FP8 o NVFP4 [317]. Per la **qualità elevata** si accettano modelli più grandi con decode più lento: SGLang valida su DGX Spark gpt-oss-120b in MXFP4 e Llama-3.3-70B in NVFP4 [317], ma un modello più grande non è automaticamente migliore sul compito, che va valutato come nella sezione 1.5.16. Per **coding e agenti** contano contesto lungo, tool calling e stabilità degli output strutturati (sezioni 3.2 e 3.4): la ricetta vLLM con Qwen3.8-27B in NVFP4 include un test di tool calling [295], quella llama.cpp usa Qwen3.6-35B-A3B in GGUF UD-Q4_K_XL, di 22,9 GB, pari a 21,3 GiB (circa 21 GB secondo il playbook) [288], [298], [318]. Per il **RAG documentale privato** generatore, embedding, reranker, indice vettoriale e ingestione condividono la stessa memoria: un database vettoriale o un OCR su CPU riducono lo spazio disponibile per la KV cache del generatore, ed è per questo che il profilo adotta $u = 0{,}7$. L'architettura del RAG è trattata nella sezione 2.4. Anche i modelli decisionali e i classificatori compatti delle sezioni 4.5.7–4.5.8, quando affiancano il generatore sullo stesso nodo, vanno conteggiati nella stessa memoria.

Tabella 1.42 — Scheda A, 1 × 128 GB: profili, configurazioni documentate e budget di memoria stimato.

Profilo	Configurazione documentata su DGX Spark	Pesi (stima)	KV, stato e servizi (stima)	Budget e verifica	Prestazioni documentate
Interazione rapida	Ollama + Open WebUI, gpt-oss:20b (MXFP4); SGLang, Llama-3.1-8B FP8/NVFP4	13,74 GB (gpt-oss-20b, checkpoint di 12,8 GiB); 8 GB FP8, 4,5 GB NVFP4 (8B)	Llama-3.1-8B: 16 GiB di KV FP16 per 4 × 32.768 token, 8 GiB in FP8	Ampiamente entro 111,4 GB (u = 0,87); capacità KV dai log	Decode a batch 1: 49,7 token/s gpt-oss-20b (Ollama); 20,5 token/s Llama 3.1 8B FP8 (SGLang); terzi, 2025
Qualità elevata	SGLang: gpt-oss-120b MXFP4; Llama-3.3-70B NVFP4	65,28 GB (gpt-oss-120b, checkpoint di 60,8 GiB); 39,4 GB (70B NVFP4)	Configurazione KV non verificata; 24–46 GB residui per KV e attivazioni (gpt-oss-120b)	Entra; contesto e concorrenza solo dopo i log di avvio	Llama 3.1 70B FP8: 2,7 token/s (terzi); limite ideale 70B NVFP4 ≈ 6,9 token/s (stima)

Profilo	Configurazione documentata su DGX Spark	Pesi (stima)	KV, stato e servizi (stima)	Budget e verifica	Prestazioni documentate
Coding e agenti	vLLM, Qwen3.8-27B NVFP4 con tool calling; llama.cpp, Qwen3.6-35B-A3B UD-Q4_K_XL	21,92 GB (file del checkpoint nvidia/Qwen3.8-27B-NVFP4, con NVFP4 per MLP e LM head e FP8 per attenzione e attenzione lineare; stima della sezione 1.5.14.4: circa 20,1 GB) [1067]; 22,9 GB (GGUF, pari a 21,3 GiB) [288]	Contesto ≥ 32.768 token, preferibilmente ≥ 100.000; layout KV da verificare	262.144 token × 8 sequenze non garantiti; verifica nei log	37,29 token/s di decodifica (valore medio) a concorrenza 1, dichiarati da NVIDIA per la ricetta vLLM a nodo singolo: misura del 19/10/2026 con decodifica speculativa DFlash2 e con un driver diverso da quello corrente, non riprodotta (Appendice B, sezione B.13) [1105], [1131]; nessuna misura indipendente e nessuna per Qwen3.6-35B-A3B con llama.cpp; limite ideale senza decodifica speculativa ≈ 15,5 token/s con circa 17,6 GB letti per token dal checkpoint, esclusi embedding, encoder visivo e modulo MTP (stima)
RAG documentale privato	Generatore della riga precedente + embedding da 0,56–0,74B + reranker + indice vettoriale (composizione editoriale)	21,92 GB + 1,1–1,5 GB (BF16) + reranker N/D	Indice FP32 per 10^6 chunk: 3,1 GB a 768 dimensioni, 12,3 GB a 3.072	u = 0,7: 89,6 GB al generatore, 38,4 GB a SO, indice, embedding e ingestione	Non documentate

Fonti: [4], [5], [56], [99], [158], [173], [205], [287], [288], [295], [297], [314], [317], [318], [320], [322], [1067], [1073], [1074], [1075], [1105], [1131]. Note: stime editoriali su 128 GB nominali, B = 10^9 parametri; pesi = parametri × bit/8, valore che esclude le parti lasciate a precisione superiore: i checkpoint NVIDIA validati nel playbook SGLang pesano 9,08 GB (Llama-3.1-8B FP8), 6,03 GB (Llama-3.1-8B NVFP4) e 42,71 GB (Llama-3.3-70B NVFP4), con embedding e LM head in BF16 (dimensioni dei file .safetensors lette il 25/10/2026 e riconfermate il 5/11/2026 alle stesse revisioni); per gpt-oss i pesi sono le dimensioni dichiarate dei checkpoint, 12,8 e 60,8 GiB (sezione 1.5.14.1); 16 GB e una GPU da 80 GB sono le memorie indicate dal produttore per l'esecuzione, non dimensioni dei pesi; KV di Llama-3.1-8B da 32 layer, 8 teste KV e dimensione 128, cioè 131.072 byte per token in FP16; le misure di terzi provengono da un'unità del programma di accesso anticipato NVIDIA e da autori del progetto SGLang, pubblicate il 13 ottobre 2025 senza indicare le versioni del software; per la misura a 70B il batch non è indicato esplicitamente (sezione 1.5.2); N/D = dato non attestato (non zero).

Storage. Per DGX Spark la guida utente indica un SSD NVMe M.2 da 1 o 4 TB con autocifratura; nei sistemi OEM taglio, tecnologia e autocifratura dipendono dalla configurazione ordinata (per esempio, unità predefinita da 2 TB QLC nel configuratore Dell; sezione 1.5.12.2, [234]). [304] I playbook indicano circa 7 GB per l'immagine Open WebUI, circa 15 GB per gpt-oss:20b, circa 25 GB per qwen3.6 e circa 40 GB liberi per l'esempio llama.cpp; le cache di Hugging Face, vLLM, FlashInfer e Triton risiedono nella home dell'utente e vengono montate nei container. [297], [298], [319] Corpus documentali, indici e backup del RAG si dimensionano a parte, con i criteri della sezione 5.4.9.

Alimentazione e raffreddamento. Il GB10 ha un TDP di 140 W e l'alimentatore esterno da 240 W lascia 100 W agli altri componenti, fra cui ConnectX-7, Wi-Fi, SSD e USB-C. L'intervallo operativo ideale è 5–30 °C con umidità 10–90% senza condensa; il raffreddamento è un sistema integrato e la pagina hardware non indica flussi d'aria, distanze o impilamento. [304] Le note di rilascio dichiarano fino a 18 W risparmiati grazie al collegamento a caldo della ConnectX-7 quando l'adattatore non è in uso e correggono, a ottobre 2026, il comportamento delle ventole in modalità headless. [303] Il consumo medio non è documentato: per il bilancio della sezione 5.4.6 va misurato sul carico reale.

Monitoraggio. La DGX Dashboard offre monitoraggio in tempo reale, raggiungibile in remoto tramite NVIDIA Sync o un tunnel SSH sulla porta 11000; la pagina non elenca le metriche, la frequenza di aggiornamento né il calcolo della memoria. [323] La guida utente di NVIDIA Sync (versione corrente 0.148.0, pagina aggiornata il 30/10/2026) documenta inoltre un Model Launcher, dichiarato in beta, che

scarica e avvia nvidia/Qwen3.8-27B-NVFP4 con vLLM nell'immagine comunitaria eugr/spark-vllm:nightly-20261002 su un DGX Spark singolo o su un cluster di due, tre o quattro sistemi da 64 o 128 GB, installa OpenCode sul nodo primario e, in modalità LAN, espone l'endpoint HTTP non cifrato su tutte le interfacce con una chiave API facoltativa; le note di rilascio non datano la versione e nessuna misura accompagna la procedura [1130]. Sulla memoria unificata `nvidia-smi` non riporta l'occupazione complessiva e `cudaMemGetInfo` la sottostima, perciò la disponibilità si legge da `/proc/meminfo` e la capacità della KV cache dai log del runtime. [295], [306], [318] Per la profilazione la guida indica Nsight Systems, Nsight Compute e `perf`, avvertendo che la registrazione di tracce `perf` ha implicazioni di sicurezza. [305]

Supporto. La pagina di supporto NVIDIA per DGX Spark (data di aggiornamento 24 marzo 2026 nella consultazione del 25/10/2026; nessuna data esposta nella rilettura del 5/11/2026, a contenuto invariato) rimanda a forum, ticket, chat e supporto NVIDIA AI Enterprise e mette a disposizione il software per aggiornare il sistema o ripristinarne le impostazioni predefinite; non indica versioni di DGX OS, durata della garanzia né un ciclo di vita del supporto software, e non descrive il supporto dei sistemi OEM. [324] Questi elementi vanno chiesti al fornitore del sistema prima dell'acquisto.

1.5.15.4 Scheda B — due sistemi GB10 collegati, 2 × 128 GB

Topologia e cablaggio. La Figura 1.5 rappresenta il collegamento diretto di due nodi e la ripartizione esplicita del carico. La pagina prodotto indica fino a quattro sistemi: due o tre collegabili direttamente con cavi QSFP, senza switch, e quattro tramite uno switch esterno da 200 GbE con RoCE v2. [172] La guida utente precisa che il Cluster Assistant di NVIDIA Sync supporta fino a tre sistemi collegati direttamente via cavo e fino a quattro tramite switch; il playbook richiede, con lo switch, un collegamento Ethernet da 200 Gbit/s per dispositivo, mentre la scheda di cablaggio corrente precisa collegamenti Ethernet a 200 Gbit/s, rete Layer 2 comune (bridge o VLAN, virtual LAN, secondo lo switch), MTU (maximum transmission unit) predefinito e controllo dello stato dei link; non fornisce una configurazione universale RoCE/PFC/ECN (priority flow control, explicit congestion notification) per ogni switch. Per quattro nodi documenta anche un anello diretto, utilizzabile soltanto con il percorso manuale. [307], [325] I cavi approvati sono Amphenol NJAAKK-N911, QSFP–QSFP112 da 400 mm (NJAAKK0006 nella versione da 0,5 m), e Luxshare LMTQF022-SD-R, DAC (direct attach copper) QSFP112 400G da 400 mm; servono cavi da almeno 200 Gb/s, mentre cavi più veloci non aumentano la velocità della porta. [307] Secondo il playbook NCCL un solo cavo basta per la banda piena; con due cavi tutte e quattro le interfacce devono ricevere un indirizzo IP. [316] Tutti i nodi devono avere la stessa memoria e lo stesso nome utente, e DGX OS deve essere aggiornato almeno alla release di aprile 2026. [302], [312], [325]

Rete di gestione e collegamento dei dati. I sistemi devono essere sulla stessa LAN di gestione, raggiungibili per indirizzo IP o nome mDNS (multicast DNS); il collegamento ConnectX-7 è una rete punto-punto distinta, sulla quale il Cluster Assistant configura anche l'SSH con chiavi fra i nodi, che non passa dalla LAN di gestione; il test NCCL avvia invece i processi con mpirun sugli indirizzi di gestione. [316], [325] La configurazione raccomandata usa il Cluster Assistant di NVIDIA Sync, con circa 10 minuti e rischio basso; il percorso manuale richiede circa un'ora, con rischio medio. [325] Il percorso manuale documentato usa lo script `setup-cluster.sh` del progetto comunitario spark-vllm-docker: prova a secco, applicazione, diagnosi e ripristino; sottoreti /24 da 10.20.0.0/16; MTU 9000 soltanto per due nodi in collegamento diretto; chiavi SSH Ed25519 dedicate; backup e manifest per il ripristino. [326] La panoramica del playbook nomina invece un file accessorio `spark_cluster_setup`: la documentazione è in transizione e va riletta prima dell'uso. [325] L'associazione fra dispositivi RoCE e interfacce Ethernet si verifica con `ibdev2netdev`, che con due cavi collegati elenca quattro righe in stato «(Up)». [307]

Verifica di banda. Il test documentato è nccl-tests: il playbook compila NCCL 2.31.2-1 per sm_121 e nccl-tests 2.19.7, poi esegue `all_gather_perf` con `mpirun`, usando l'interfaccia di gestione per l'avvio dei processi; la variante con buffer grandi usa `-b 16G -e 16G`. [312], [316] Né la panoramica né la scheda a due nodi indicano un valore atteso di banda: il confronto con il tetto nominale di 25 GB/s per collegamento è una stima. Le pagine consultate non documentano test con iperf o ib_write_bw.

Comunicazione fra processi e runtime distribuito. La procedura manuale documentata per vLLM non usa Ray: ogni nodo esegue un rank del tensor parallelism (`--tensor-parallel-size 2`, `--nnodes 2`), il nodo head è il rank 0, il worker il rank 1 con `--headless`; il rendezvous avviene sull'indirizzo ConnectX del nodo head, porta 29501, e l'API risponde sulla porta 8000 del nodo head. [297] La ricetta per 2 × 128 GB serve GLM-5.3-Flash nel checkpoint `nvidia/GLM-5.3-Flash-FP4-Local` con KV cache FP8, contesto massimo di 500.000 token per richiesta, al più quattro sequenze, speculative decoding MTP e `--gpu-memory-utilization 0.87`; quella per 2 × 64 GB usa Qwen3.8 Flash-Next con 0,8 e 3 G di KV cache. Il playbook avverte che i limiti di contesto sono tetti per richiesta, non capacità concorrente garantita, e che con tre o quattro nodi è disponibile solo il percorso NVIDIA Sync, senza esempio a riga di comando. [297], [302] GLM-5.3-Flash è un MoE da 320B parametri totali e 18B attivi; il modello base è rilasciato con licenza MIT [112], mentre l'uso del checkpoint quantizzato nvidia/GLM-5.3-Flash-FP4-Local impiegato dalla ricetta è regolato dall'NVIDIA Open Model Agreement, indicato dalla scheda come condizione del download. [1028] A 4,5 bit i pesi richiederebbero idealmente 180 GB ($320 \cdot 10^9 \times 4{,}5/8$), ma alla revisione consultata i file .safetensors del checkpoint della ricetta occupano 177,46 GB e la scheda dichiara per le scale NVFP4 un formato compresso senza perdita, un uso di memoria di circa 166 GiB e la validazione su due DGX Spark da 128 GB con 2 milioni di token di KV cache FP8 (dichiarazioni del produttore) [1028]. Con una ripartizione uniforme fra i due rank (ipotesi) ogni nodo conserva circa 89 GB di pesi, che entro il budget di 111,4 GB lasciano circa 22 GB per KV cache, attivazioni, componenti MTP e comunicazione; con le ipotesi di workspace e margine della sezione 1.5.14.3 il requisito, cache esclusa, vale circa 104–105 GB per nodo, al limite (stima). Il dichiarato «fino a 400 miliardi di parametri» per due unità [172] richiede 200 GB a 4 bit esatti, 100 GB per nodo, ma 112,5 GB per nodo in NVFP4, oltre il budget di 111,4 GB: è un limite di capienza, non un punto di esercizio (calcolo editoriale).

Il playbook SGLang non indica il supporto multi-nodo per DGX Spark [317]; llama.cpp viene compilato con il backend RPC, ma il suo uso fra due sistemi GB10 non è documentato nelle pagine consultate [318]; per TensorRT-LLM il catalogo non contiene un playbook dedicato: il playbook di quantizzazione NVFP4 lo usa soltanto per servire a nodo singolo il checkpoint quantizzato con trtllm-serve, senza esempi a più nodi. [321], [1072] Il repository dei playbook NVIDIA su GitHub contiene però «Serve LLMs with NVIDIA TensorRT-LLM», aggiornato al 12 agosto 2026, con una scheda «Multi-node serving» per DGX Spark (contenitore nvcr.io/nvidia/tensorrt-llm/release:1.3.0rc5, TP_SIZE=2 nell'esempio), non pubblicata nel catalogo e non verificata in questa edizione (Appendice C, sezione C.10) [1132]. La Tabella 1.43 distingue le strategie per obiettivo e costo.

Tabella 1.43 — Strategie su due nodi GB10: obiettivi, memoria, comunicazione e stato documentale.

Strategia	Obiettivo	Memoria per nodo	Comunicazione fra nodi	Effetto atteso	Stato su GB10
Tensor parallelism, TP = 2	Servire un modello che non entra in un nodo	Circa metà dei pesi più la KV cache di competenza	Collettive NCCL su RoCE a ogni passo di decodifica	Capienza maggiore; latenza per token dipendente dalla rete	Documentato: vLLM, ricette 2 × 64 e 2 × 128 GB
Pipeline parallelism, PP = 2	Ripartire i layer con meno collettive	Circa metà dei layer	Attivazioni al confine fra gli stadi	Capienza maggiore; a batch 1 gli stadi lavorano in sequenza	Opzione presente nel comando con valore 1; nessun esempio con 2
Expert parallelism	Distribuire gli esperti di un MoE	Esperti ripartiti	Scambi all-to-all secondo il routing	Dipende dal routing e dal carico	Non documentato nelle pagine consultate
Repliche indipendenti	Throughput multiutente e ridondanza	Pesi completi su ciascun nodo	Nessuna nel ciclo dei token	Throughput aggregato fino a circa 2× (limite ideale); nessuna capienza aggiuntiva per replica	Composizione di procedure a nodo singolo
Separazione dei servizi	Isolare carichi diversi, per esempio LLM e RAG	Indipendente per servizio	Chiamate HTTP o RPC applicative	Nessuna contesa di memoria sul nodo del generatore	Composizione editoriale

Fonti: [297], [302], [317], [318], [321]. Note: «documentato» indica una procedura nei playbook NVIDIA consultati il 25/10/2026 e riverificati il 5/11/2026; «composizione» indica un'architettura costruita con procedure documentate a nodo singolo, senza prova eseguita. Nessuna strategia promette prestazioni doppie.

Le strategie servono obiettivi diversi. Il tensor parallelism è l'unica documentata per servire su GB10 un modello più grande della memoria di un nodo e paga il costo di comunicazione a ogni token. Il pipeline parallelism riduce le collettive ma non accelera la singola richiesta a basso batch. Le repliche indipendenti aumentano il numero di richieste servite in parallelo, non la memoria disponibile a ciascuna replica: replicare lo stesso modello non raddoppia la capienza. La separazione dei servizi è la scelta naturale per il RAG privato, perché sposta ingestione, embedding, reranking e indice sul secondo nodo e lascia al generatore la memoria del primo nodo, al netto di sistema operativo e servizi (u fino a 0,87; Tabella 1.44).

Tabella 1.44 — Scheda B, 2 × 128 GB: profili e strategie consigliate.

Profilo	Strategia	Memoria per nodo (stima)	Traffico sul collegamento	Stato	Prestazioni
Interazione rapida multiutente	Repliche indipendenti, una per nodo, dietro un gateway	Come nella Scheda A, per replica	Nessuno nel ciclo dei token; solo richieste applicative	Composizione di procedure documentate	Throughput aggregato fino a circa 2× (limite ideale); latenza della singola richiesta invariata
Qualità elevata	TP = 2: GLM-5.3-Flash FP4-Local, 320B/18B	circa 89 GB di pesi per nodo (177,46 GB di file del checkpoint, ripartizione uniforme ipotizzata); circa 22 GB residui entro u = 0,87 (sezione 1.5.14.4) [1028]	Collettive NCCL su RoCE a ogni passo	Documentato, con immagine comunitaria	33,94 token/s di decodifica (media ponderata sul tempo attivo) a concorrenza 1, dichiarati da NVIDIA per la ricetta: misura del 16/10/2026 con decodifica speculativa MTP e con driver e revisione del checkpoint diversi da quelli correnti, non riprodotta (Appendice C, sezione C.10) [1133]
Coding e agenti	Separazione: modello di coding sul nodo 1; strumenti, sandbox ed embedding del repository sul nodo 2	Nodo 1 come nella Scheda A	Chiamate degli strumenti e del retrieval	Composizione editoriale	Non documentate
RAG documentale privato	Separazione: generatore sul nodo 1; ingestione, embedding, reranker e indice sul nodo 2	Nodo 1 fino a u = 0,87; nodo 2 per i servizi	Query e passaggi recuperati	Composizione editoriale	Non documentate

Fonti: [112], [172], [297], [302], [326], [1028], [1133]. Note: stime editoriali su 128 GB nominali per nodo; per GLM-5.3-Flash FP4-Local la base è la dimensione dei file del checkpoint alla revisione consultata, 177,46 GB, inferiore ai 180 GB = $320 \cdot 10^9 \times 4{,}5/8$, e la scheda dichiara per le scale NVFP4 un formato compresso senza perdita; memoria residente e capacità della KV cache vanno lette nei log di avvio di ciascun rank.

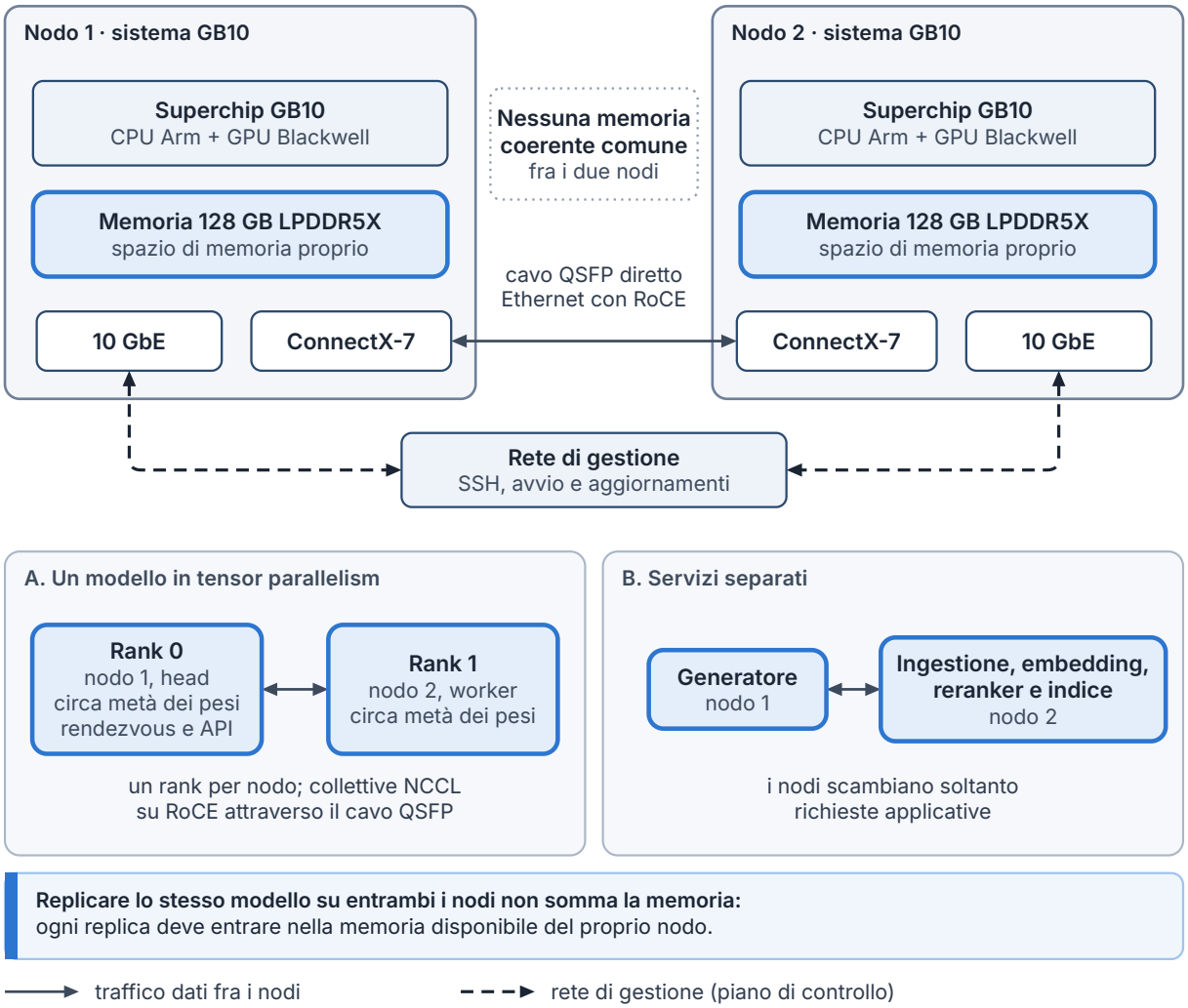


Figura 1.5 — Due sistemi GB10 collegati: memorie distinte, collegamento diretto e ripartizione esplicita. Nel pannello A un modello è ripartito in tensor parallelism con un rank per nodo, collettive NCCL su RoCE attraverso il cavo QSFP e rendezvous sul nodo head; nel pannello B i servizi sono separati e i nodi scambiano soltanto richieste applicative. La rete di gestione trasporta SSH, avvio e aggiornamenti; replicare il modello non somma la memoria. Tipo: schema concettuale. Fonti: [297], [302], [307], [308], [312], [316], [325], [326].

Storage, alimentazione, raffreddamento e monitoraggio. Ogni nodo monta la propria cache dei modelli (Appendice C, Blocco C.6-3): lo spazio per i file del checkpoint va previsto su entrambi i nodi, e per GLM-5.3-Flash FP4 l'API Hugging Face, consultata il 25/10/2026 e riletta il 5/11/2026 alla stessa revisione 76348ddb2be2e35e39ecfc00e75dc07703f4f3d5, elenca 42 file .safetensors per 177.463.841.024 byte (177,464 GB decimali; 165,276 GiB), 177,52 GB con gli altri file del repository. È lo spazio minimo da riservare al checkpoint su ciascun nodo, cui si aggiungono immagine e cache del runtime; non coincide con la memoria residente, che la scheda indica in circa 166 GiB complessivi (dichiarazione del produttore) e che va verificata, con i margini, nei log di ciascun rank [1028]. Ciascun sistema usa il proprio alimentatore da 240 W, con TDP del GB10 di 140 W [304]; distanze, flussi d'aria e impilamento di due unità non sono documentati e il consumo va misurato. Memoria disponibile, temperatura, ventole e log di avvio si controllano su entrambi i nodi, perché nel tensor parallelism un nodo rallentato rallenta ogni passo (inferenza dal funzionamento sincrono delle collettive).

Sicurezza operativa. Le ricette espongono l'API su tutte le interfacce: per vLLM il playbook avverte che il bind su 0.0.0.0 può esporre un'API non autenticata e indica una rete fidata o controlli d'accesso, mentre per i client locali e il tunnel SSH basta l'interfaccia di loopback. [295], [297] Il container a due nodi gira con `--privileged` e `--network host`, il test NCCL disattiva la verifica delle chiavi host SSH e le immagini e gli script comunitari introducono dipendenze di supply chain da valutare come nelle sezioni 1.5.9 e 5.4.3. [297], [316] La guida di sicurezza di vLLM [228], richiamata nella sezione 5.4.9.7, indica che le comunicazioni fra i nodi di un deployment multinodo non sono sicure per default e richiede una rete

dedicata e isolata, `VLLM_HOST_IP` fissato e porte limitate agli host fidati; la ricetta a due nodi non imposta `VLLM_HOST_IP` e, con `--network host`, i processi del runtime sono raggiungibili anche dalle interfacce di gestione (Appendice C). Esposizione degli endpoint, autenticazione, controllo dell'egress e strumenti degli agenti sono trattati nelle sezioni 1.5.8, 5.1 e 5.4.9; protezione dei dati e obblighi dell'AI Act non derivano dalla collocazione locale dell'hardware (sezione 5.5).

1.5.15.5 Colli di bottiglia e piano di misura

Il **prefill** elabora l'intero prompt in parallelo ed è limitato soprattutto dal calcolo (sezione 1.5.2): il TTFT cresce con la lunghezza del prompt e si governa con chunked prefill, budget di token per iterazione e prefix caching, opzioni che il playbook vLLM descrive. [295] Il picco di 1 PFLOP FP4 con sparsità non è una velocità di prefill.

Il **decode** a basso batch è limitato dalla banda di memoria. Con il modello semplificato della sezione 1.5.2, $r \leq BW/W_{\text{read}}$, un modello denso da 27B in NVFP4 ha un limite ideale di circa 273/17,6 ≈ 15,5 token/s per richiesta con il checkpoint nvidia/Qwen3.8-27B-NVFP4, che conserva in FP8 attenzione e attenzione lineare, se a ogni token si leggono tutti i pesi tranne embedding, encoder visivo e modulo MTP [1067] (273/15,19 ≈ 18,0 token/s se tutti i parametri fossero NVFP4), uno da 70B in NVFP4 di circa 6,9 token/s e un 8B in FP8 di circa 34,1 token/s, contro i 20,5 token/s misurati da terzi. [173] Nei MoE i byte letti per token dipendono dai parametri attivi, ma la capienza dai parametri totali: per gpt-oss-20b i 3,61B parametri attivi comprendono 0,64B di attenzione e circa 0,58B di unembedding, metà degli 1,16B di embedding e unembedding, entrambi in BF16 [287]: con il resto in MXFP4 a 4,25 bit si leggono circa 3,7 GB per token e il limite ideale è circa 74 token/s, di cui i 49,7 token/s misurati con Ollama rappresentano circa due terzi; con tutti i 3,6B attivi a 4,25 bit il limite sarebbe circa 143 token/s. [4], [5], [173] I limiti sono superiori, non previsioni, e valgono quando ogni token generato richiede un passaggio completo del modello: precisione dei parametri non quantizzati, KV cache, kernel e runtime li abbassano, mentre la decodifica speculativa, che verifica in un solo passaggio più token proposti da un modello o da un componente di bozza (sezione 1.5.5), può superarli. Per questo i 37,29 token/s di decodifica dichiarati da NVIDIA per la ricetta Qwen3.8-27B NVFP4 con il modello di bozza DFlash2 su un DGX Spark da 128 GB (Tabella 1.42), non riprodotti e pubblicati senza il tasso di accettazione dei token proposti, non contraddicono il limite di 15,5 token/s e non lo verificano (inferenza editoriale) [1105], [1131]. Con il batch il throughput aggregato cresce mentre la velocità della singola richiesta diminuisce: la stessa fonte riporta, per Llama 3.1 8B FP8 con SGLang, 368 token/s aggregati a batch 32 contro 20,5 a batch 1, circa 11,5 per sequenza. [173]

Fra due nodi il **collegamento** aggiunge un termine che a batch 1 dipende soprattutto dalla latenza. In prima approssimazione, per un passo di decodifica in tensor parallelism,

$$t_{\text{token}} \approx \frac{W_{\text{nodo}}}{BW} + n_{\text{coll}}\,\alpha + \frac{V}{B_{\text{link}}},$$

dove W_{nodo} sono i byte letti da ciascun nodo, n_{coll} il numero di collettive per passo, α la loro latenza, V il volume trasferito e B_{link} la banda del collegamento. La comunicazione domina quando $n_{\text{coll}}\,\alpha$ supera W_{nodo}/BW. Come esempio parametrico, con i 18B parametri attivi di GLM-5.3-Flash a 4,5 bit divisi fra due nodi, $W_{\text{nodo}} \approx 5{,}06$ GB e $W_{\text{nodo}}/BW \approx 18{,}5$ ms; con un valore ipotetico di 180 collettive per passo la soglia è $\alpha^{*} \approx 103$ µs. Il numero di collettive dipende dall'architettura e dall'implementazione e la latenza delle collettive su GB10 non è documentata: entrambi vanno misurati, per esempio ripetendo nccl-tests con messaggi piccoli oltre a quelli da 16 GB del playbook. [316] Il calcolo è un'ipotesi di lavoro editoriale, non una previsione.

Il piano di misura minimo comprende: TTFT e ITL p50/p95 alla concorrenza di progetto; token/s per richiesta e aggregati; capacità della KV cache riportata all'avvio; memoria disponibile da `/proc/meminfo` prima e durante il carico; banda di bus di nccl-tests sul collegamento; temperatura e comportamento delle ventole nel tempo. I criteri di riproducibilità sono quelli della sezione 1.5.10.4. Nessuna di queste misure è stata eseguita per questa edizione.

Modelli e verifiche: raccomandazioni per due sistemi GB10

Estratto della sezione 1.5.16 dell'edizione.

1.5.16 Modelli per l'AI privata: selezione per scenario, matrice di compatibilità e raccomandazioni

Non esiste un modello scaricabile migliore in assoluto. La scelta dipende dal compito, dalla lingua, dalla piattaforma, dal runtime disponibile, dalla licenza e dal livello di servizio richiesto. Questa sottosezione seleziona i candidati per sette scenari, ne riporta le caratteristiche verificate sulle schede ufficiali il 25 ottobre 2026 e riverificate il 5 novembre 2026, e li incrocia con sei piattaforme e sei runtime, distinguendo tre domande diverse: se il modello **entra in memoria**, stimato con il metodo del §1.5.14; se la combinazione **è supportata**, cioè documentata da produttori del modello, della piattaforma o del runtime; se esistono **prestazioni utili documentate**, cioè misure con condizioni dichiarate. Nessuna prova è stata eseguita per questa edizione: le misure riportate sono quelle delle fonti, le stime sono calcoli editoriali.

Nota della guida. Della sezione 1.5.16 sono riportate l'introduzione, la sottosezione 1.5.16.6, limitatamente all'introduzione, alle Tabelle 1.50 e 1.51 e alle conclusioni, e la sottosezione 1.5.16.7. Le sottosezioni 1.5.16.1–1.5.16.5, con le Tabelle 1.45–1.49, e la Tabella 1.52, dedicata alle altre piattaforme, sono nell'edizione completa; le «tabelle precedenti» su cui l'introduzione della 1.5.16.6 fonda le raccomandazioni sono le Tabelle 1.45–1.49, non le Tabelle 1.40–1.44 riportate in questa guida.

1.5.16.6 Raccomandazioni per configurazione

Le raccomandazioni seguono quattro obiettivi: equilibrio fra qualità, costo di memoria e supporto; priorità alla qualità; priorità alla latenza; uso multiutente. Sono motivate dalle evidenze delle tabelle precedenti; dove le evidenze non bastano, la tabella lo dichiara. In ogni caso la scelta finale richiede le verifiche del §1.5.16.7.

Tabella 1.50 — Raccomandazioni per GB10 a nodo singolo (1×128 GB).

Obiettivo	Scelta	Motivazione	Condizioni e limiti	Evidenza
Equilibrata	gpt-oss-120b MXFP4 con SGLang; vLLM come alternativa	Validato da NVIDIA su DGX Spark; decode misurato di 50,54 token/s a batch 1 e 117,06 aggregati a batch 8; 65,28 GB di pesi e 36 KiB per token di cache lasciano 24–46 GB per cache e attivazioni entro il budget del motore; Apache 2.0; unico dato per l'italiano (MMMLU 85,0)	Solo testo; contesto 131.072; misure del 2025 da ripetere con la versione adottata	Misura di terzi; dichiarazioni del produttore

Obiettivo	Scelta	Motivazione	Condizioni e limiti	Evidenza
Qualità	Evidenza insufficiente per una scelta univoca. Da confrontare sul proprio carico: Nemotron 3.5 Super VL NVFP4, Qwen3.8-27B (NVFP4 o FP8), Qwen3.5-122B-A10B GPTQ-Int4	Nemotron: configurazione del produttore per GB10 e confronto NVFP4–BF16 pubblicato; Qwen3.8-27B: LiveBench Overall 62,7 e Coding 86,8 sul checkpoint originale, contro 50,0 e 61,3 del checkpoint NVFP4 di Nemotron nella stessa valutazione indipendente a esecuzione singola (rilevazione del 25/10/2026 della release del 14/10/2026; nei file pubblicati dal valutatore il 30/10/2026 l'Overall di Nemotron NVFP4 è 52,9, §1.5.16.1); confronto NVFP4–BF16 pubblicato da NVIDIA; ricette vLLM per GB10; Qwen3.5-122B: benchmark multilingue e agentici più ampi del produttore	Nessuna misura indipendente di prestazioni su GB10 per i tre modelli: nelle proprie ricette NVIDIA dichiara, a concorrenza 1 e con decodifica speculativa, 51,33 token/s di decodifica per Nemotron 3.5 Super VL NVFP4, 37,29 per Qwen3.8-27B NVFP4 e 29,56 per un proprio checkpoint NVFP4 di Qwen3.5-122B-A10B, valori non riprodotti (Tabella 1.49); Qwen3.5-122B GPTQ-Int4 al limite di memoria e non documentato su GB10; Nemotron documentato dalla scheda con una sola sequenza e dalla ricetta NVIDIA con al più 8	Dichiarazioni dei produttori; valutazione indipendente LiveBench a esecuzione singola; stime
Latenza	gpt-oss-20b MXFP4 con SGLang o Ollama	Decode misurato di 69,56 token/s (SGLang) e, con Ollama, 60,91 token/s nel foglio o 49,7 nel testo dell'articolo, a batch 1; 13,74 GB di pesi	Qualità inferiore al 120b sulle suite del produttore; Qwen3.6-35B-A3B con llama.cpp è documentato dal playbook NVIDIA ma non misurato; per un proprio checkpoint NVFP4 dello stesso modello, con vLLM e MTP, NVIDIA dichiara 124,4 token/s di decodifica a concorrenza 1, non riprodotti (Tabella 1.49)	Misura di terzi
Multiutente	gpt-oss-120b o gpt-oss-20b con SGLang e batching continuo	Scalabilità misurata fino a 324,62 token/s aggregati (120b, batch 128) e 1.345,93 (20b, batch 256); bilancio di memoria del carico misurato al limite secondo la regola del §1.5.14.3 ma entro il budget massimo documentato (98,1 GB per 128 × 4.096 token)	Con una soglia di progetto di 10 token/s per utente (ipotesi) il 120b sostiene circa 8 sequenze attive (14,63 token/s a batch 8; 9,24 a batch 16); cache FP8 per contesti lunghi	Misura di terzi; stima

Fonti: [32], [99], [173], [223], [287], [289]–[291], [294]–[296], [298], [299], [317], [1077], [1067], [1105], [1131]. Note: memoria secondo il §1.5.14; «evidenza insufficiente» indica l'assenza di misure con condizioni o di una valutazione comune, non un giudizio negativo sui modelli.

Tabella 1.51 — Raccomandazioni per due sistemi GB10 (2×128 GB).

Obiettivo	Scelta	Motivazione	Condizioni e limiti	Evidenza
Equilibrata	Due repliche indipendenti di gpt-oss-120b, oppure separazione dei servizi: modello generativo su un nodo; embedding, reranker, indice vettoriale e un modello compatto sull'altro	Ogni nodo ripete la configurazione validata a nodo singolo; nessun traffico per token fra i nodi; la memoria disponibile per replica non cambia	Throughput aggregato non misurato; servono bilanciatore e gestione delle sessioni; la cache di prefisso non è condivisa fra le repliche	Proposta ingegneristica su evidenze per nodo

Obiettivo	Scelta	Motivazione	Condizioni e limiti	Evidenza
Qualità	GLM-5.3-Flash FP4 con vLLM e TP = 2; in alternativa Qwen3.5-122B-A10B FP8 con TP = 2	Ricetta del playbook NVIDIA a due nodi per i sistemi da 128 GB, con un modello che richiede due nodi per la capacità (u = 0,87; 4 sequenze; tetto di 500.000 token per richiesta); licenza MIT; LiveBench 2.0 Overall 70,6 e Coding 80,6 sul checkpoint originale nella release del 14/10/2026 (valutazione indipendente a esecuzione singola, aggiornamento del 23/10/2026), 62,1 e 80,6 nella release del 25/10/2026, che riporta punteggi diversi per due suite (Agentic da 28,4 a 19,0) e calcola l'Overall con pesi diversi (Tabella 5.9); Terminal-Bench 2.1 84,3 dichiarato dal produttore con harness Claude Code 2.1.207: la versione 2.1 è documentata dagli autori del benchmark; il risultato, dichiarato da Z.ai per il checkpoint originale e non replicato, non è direttamente confrontabile con Terminal-Bench 4.0, che usa compiti e condizioni differenti (§3.2.9) [1134]. Per Qwen3.5-122B FP8 la stima dà 80,4 GB per nodo con i file pubblicati (84,2 GB con la stima superiore dei pesi dell'Allegato 2) con quattro sequenze da 131.072 token e cache FP8 (Tabella 1.39)	Immagine comunitaria nightly; nessuna misura indipendente (per la ricetta GLM NVIDIA dichiara 33,94 token/s di decodifica a concorrenza 1, non riprodotti: Tabella 1.44); lingue dichiarate inglese e cinese; margine ridotto; Qwen3.5-122B FP8 non documentato su due GB10	Evidenza insufficiente sulle prestazioni
Latenza	Evidenza insufficiente per il TP su due nodi; preferire la scelta per la latenza a nodo singolo (Tabella 1.50)	Il TP aggiunge a ogni token due all-reduce per layer, circa 1,31 MB per token e sequenza per Qwen3.8-27B, con latenza delle operazioni collettive non documentata	Misurare banda e latenza con i test NCCL del §1.5.15 prima di usare il TP per ridurre la latenza; per Qwen3.8-27B NVFP4 NVIDIA dichiara 54,74 token/s di decodifica su due nodi con TP = 2 e 37,29 su un nodo, con immagini, frazioni di memoria e numero di token speculativi diversi: i due valori, non riprodotti, non isolano l'effetto del TP (Tabella 1.49)	Stima
Multiutente	Repliche indipendenti con bilanciatore	Raddoppia, al più, le sequenze servibili in aggregato quando il carico è distribuibile; per ciascun nodo valgono le misure della configurazione a nodo singolo	La dichiarazione NVIDIA sul doppio nodo (fino a 2,6 volte con NVFP4 e speculative decoding rispetto a FP8) non indica metrica, carico né concorrenza	Proposta; dichiarazione del produttore

Fonti: [32], [112], [291], [296], [297], [299], [300], [302], [317], [334], [1077], [1105], [1133], [1135], [1131]. Note: due nodi = due spazi di memoria distinti collegati da ConnectX-7 (§1.5.15); il vincolo è per nodo (§1.5.14.3); nessuna promessa di prestazioni doppie.

Tre conclusioni valgono per tutte le piattaforme. Un modello più grande non è automaticamente la scelta migliore: su GB10 un denso da $70 \cdot 10^9$ parametri in FP8 è circa 19 volte più lento nel decode di gpt-oss-120b con SGLang (2,66 contro 50,54 token/s, misure LMSYS), che ha più parametri totali ma $5{,}13 \cdot 10^9$ attivi. La memoria disponibile determina prima di tutto il contesto e la concorrenza, non soltanto quali pesi si possono caricare (§1.5.14). Per il servizio multiutente la grandezza da garantire è la velocità per richiesta alla concorrenza prevista, non il throughput aggregato.

1.5.16.7 Verifiche prima dell'adozione

1. Fissare modello, revisione, formato e provenienza del checkpoint o della conversione; leggere la licenza della revisione adottata (Tabella 1.13), in particolare per GLM-5.3, FunctionGemma e le conversioni di terzi.
2. Verificare l'integrità dei file: formati senza deserializzazione di oggetti, hash o firme, ispezione e fissaggio del chat template dei file GGUF, che può contenere istruzioni nascoste (§1.5.9) [218].
3. Misurare la qualità sulla quantizzazione e sul runtime di destinazione con un insieme di valutazione in italiano costruito sui compiti reali, incluse chiamate a strumenti con argomenti in italiano, usando lo stesso harness per tutti i candidati e registrando livello di ragionamento e parametri di generazione (§1.2.1).
4. Misurare TTFT, ITL e token/s per richiesta e aggregati, con p50 e p95, alla concorrenza e alle lunghezze previste; misurare il picco di memoria e l'avvio a freddo (§1.5.10.3–1.5.10.4).
5. Controllare che il runtime sfrutti finestre locali, cache FP8 e stato delle architetture ibride come assunto nelle stime (§1.5.14).
6. Su due nodi, misurare banda e latenza fra i sistemi e confrontare TP, repliche e separazione dei servizi sullo stesso carico (§1.5.15); preferire immagini e versioni con supporto dichiarato a quelle nightly per l'esercizio.
7. Disattivare le funzioni cloud dei runtime, come i modelli remoti di Ollama, per i quali la documentazione prevede una modalità solo locale [213], e verificare il traffico uscente; gli strumenti degli agenti restano canali di trasferimento dei dati anche con un modello locale (§1.5.8, §5.4.9).
8. Ripetere le misure a ogni aggiornamento di runtime, immagine, driver o checkpoint: i risultati di questa sezione valgono per le versioni e le date indicate.

Appendice B — Procedura documentata: sistema GB10 a nodo singolo (128 GB)

Questa appendice raccoglie, in forma eseguibile, le procedure ufficiali NVIDIA per un sistema GB10 a nodo singolo, come risultano dalle pagine consultate il 25 ottobre 2026 e riverificate al 5 novembre 2026. È il complemento operativo della Scheda A (sezione 1.5.15.3). Nessun comando è stato eseguito durante la preparazione di questa edizione: i blocchi riportano procedure documentate, adattamenti motivati e stime, distinti da un'etichetta. Le versioni e le procedure di aggiornamento citate valgono per i sistemi DGX Spark Founders Edition; per i sistemi di altri produttori possono differire. [227], [303]

B.1 Convenzioni, prerequisiti e perimetro

Ogni blocco è preceduto da un'etichetta:

- **Procedura documentata (fonte …)**: comandi o file riportati come nella pagina ufficiale indicata; i segnaposto tra parentesi angolari, come `<model-id>`, vanno sostituiti.
- **Adattamento editoriale**: modifica motivata di una procedura documentata, per esempio il bind sull'interfaccia di loopback, oppure comando standard del sistema operativo non citato dalle pagine NVIDIA consultate; va verificato prima dell'uso.
- **Stima**: calcolo editoriale, con formula e ipotesi.

Per ogni verifica si indica l'esito atteso quando la fonte lo documenta; in caso contrario l'esito è dichiarato non documentato.

Prerequisiti documentati:

- alimentatore esterno da 240 W fornito con il sistema; un alimentatore diverso o di potenza inferiore può ridurre le prestazioni, impedire l'avvio o causare spegnimenti imprevisti [304], [306];
- temperatura operativa ideale 5–30 °C e umidità 10–90% senza condensa [304];
- Docker disponibile e utilizzabile dall'utente; NVIDIA Container Toolkit preinstallato e configurato [294], [310];
- Hugging Face CLI installata e, per i modelli ad accesso controllato, un token di accesso [294], [317];
- accesso SSH al sistema per l'uso remoto; NVIDIA Sync facoltativo [294];
- spazio su disco per immagini e modelli: circa 7 GB per l'immagine Open WebUI, circa 15 GB per gpt-oss:20b, circa 25 GB per qwen3.6, circa 40 GB liberi per l'esempio llama.cpp [298], [319].

Perimetro: inventario e aggiornamento del sistema, runtime dei container, memoria disponibile, quattro percorsi di servizio (Ollama con Open WebUI, vLLM, llama.cpp, SGLang), monitoraggio, verifica finale e ripristino. Configurazione di rete, autenticazione degli utenti, controllo dell'egress, log e backup dell'architettura privata sono trattati nella sezione 5.4.9.

B.2 Inventario delle versioni

Prima di applicare una ricetta si leggono le versioni installate: dalla release di giugno 2026 gli aggiornamenti OTA non vengono installati per default durante la configurazione iniziale. [303]

Blocco B.2-1 — procedura documentata (fonti [310], [312]). Controlli dei prerequisiti indicati dal playbook NCCL e verifica della versione del Container Toolkit.

```
nvidia-smi
nvcc --version
sudo whoami
nvidia-ctk --version
```

Esito atteso: `nvidia-smi` mostra il driver e, sulla GPU integrata, «Memory-Usage: Not Supported», comportamento atteso [306]; `nvcc --version` mostra la versione del CUDA Toolkit; `sudo whoami` conferma l'accesso con privilegi amministrativi; `nvidia-ctk --version` mostra la versione del Toolkit, da confrontare con il minimo 1.17.7. [309] I valori correnti per la Founders Edition sono nella Tabella 1.41: driver 580.178.04 e CUDA Toolkit 13.0.3. [303]

Blocco B.2-2 — adattamento editoriale (comandi standard di Ubuntu, non citati dalle pagine NVIDIA consultate).

```
uname -r
cat /etc/os-release
```

Esito atteso (stima): kernel della serie 7.0 dopo la release di ottobre 2026 (versione corrente indicata: 7.0.0-1019) e Ubuntu 24.04. [303], [311] Il comando per leggere la versione di DGX OS non è indicato nelle pagine consultate; la DGX Dashboard mostra gli aggiornamenti disponibili. [227], [323]

B.3 Aggiornamento di sistema operativo, driver e firmware

Il percorso raccomandato è la DGX Dashboard, che mostra gli aggiornamenti disponibili, installa patch di sicurezza e aggiornamenti di sistema e gestisce driver e firmware. Prima dell'aggiornamento la guida raccomanda alimentazione stabile, chiusura delle applicazioni, un piano di ripristino, una finestra di manutenzione e il backup dei dati critici. [227]

Blocco B.3-1 — procedura documentata (fonte [227]). Aggiornamento da riga di comando, indicato per utenti esperti o quando la Dashboard non è disponibile.

```
sudo apt update
sudo apt dist-upgrade
sudo fwupdmgr refresh
sudo fwupdmgr upgrade
sudo reboot
```

Esito atteso: non documentato oltre al riavvio finale; dopo il riavvio si ripete il Blocco B.2-1. La pagina non descrive un rollback degli aggiornamenti; nella guida aggiornata il 4 novembre 2026 la segue la pagina System Repair, che per un sistema che non si avvia descrive l'avvio di un kernel installato in precedenza dal menu GRUB oppure di Ubuntu da una chiavetta USB in modalità live, per completare un aggiornamento dei pacchetti, ricostruire l'initramfs, installare i pacchetti firmati dei driver NVIDIA o, dall'ambiente live, riparare il file system, senza cancellare l'SSD interno [1136]; la pagina System Recovery descrive invece un ripristino da USB che riporta sistema operativo e firmware allo stato di fabbrica e cancella completamente l'SSD interno [1103]; la guida di porting indica un ripristino tramite USB con immagini dai repository di Canonical e NVIDIA. [227], [311] Le note annunciano aggiornamenti da repository locale o USB e installazioni air-gapped nella release di aprile 2026. La pagina Custom Installation with cloud-init, consultata il 25 ottobre 2026 e riverificata al 5 novembre 2026 (aggiornamento del 4 novembre), richiede però per Founders Edition il pacchetto nvidia-spark-cloud-init della release di ottobre 2026 o successiva; per gli OEM rimanda alla rispettiva documentazione. Non si deve applicare retroattivamente il flusso corrente a un sistema fermo alla release di aprile. I flussi OEMDATA, repository APT e mirror sono stati riesaminati documentalmente, senza creare supporti né eseguire reinstallazioni; il reflash descritto dalla fonte cancella lo storage locale. [226], [303], [972]

Blocco B.3-2 — procedura documentata (fonte [323]). Disattivazione dell'interfaccia di aggiornamento della Dashboard, per esempio per governare le modifiche con una procedura di change management. Contenuto del file `/opt/nvidia/dgx-dashboard/settings.json`, di proprietà di root:

```
{
  "update": {
    "enabled": false
  }
}
sudo systemctl restart dgx-dashboard.service
```

Esito atteso: la Dashboard mostra «Updates Disabled» anche agli utenti con `sudo`; senza riavvio del servizio la modifica si applica entro 10 minuti. Il file supporta anche `wifiAssistant.enabled`, con cui l'amministratore ammette o esclude il Wi-Fi Assistant [323], introdotto con la release di ottobre 2026 per collegare il sistema a una nuova rete Wi-Fi quando è usato senza tastiera, mouse e display [303]; se il file è gestito da Landscape la Dashboard disattiva gli aggiornamenti. [323] Per gli ambienti che richiedono controlli di sicurezza aggiuntivi, fra cui le installazioni isolate della sezione 1.5.11.1, le note di rilascio documentano dalla release di gennaio 2026 la disattivazione di Wi-Fi e Bluetooth anche

nell'UEFI [303], dove la voce «Wireless LAN & Bluetooth» del menu IO Port Access le disattiva insieme, mentre per disattivarne una sola la guida utente rinvia alle impostazioni del sistema operativo [1137].

Blocco B.3-3 — adattamento editoriale. La guida indica il file ma non il comando per crearlo. Poiché `tee` sovrascrive l'intero file, che può contenere anche il campo `wifiAssistant.enabled`, se il file esiste se ne conserva prima una copia (`sudo cp -a /opt/nvidia/dgx-dashboard/settings.json /opt/nvidia/dgx-dashboard/settings.json.bak`) e se ne integra il contenuto; per un file nuovo una possibile scrittura è:

```
printf '{\n  "update": {\n    "enabled": false\n  }\n}\n' | sudo tee /opt/nvidia/dgx-dashboard/settings.json
```

B.4 Runtime dei container e accesso alla GPU

Blocco B.4-1 — procedura documentata (fonte [310]). Passo facoltativo: senza di esso Docker si usa con `sudo`. L'appartenenza al gruppo `docker` consente di avviare container privilegiati e quindi equivale di fatto a privilegi di amministratore sull'host (la documentazione Docker avverte che il gruppo docker concede privilegi di livello root [1104]): su un nodo condiviso va concessa soltanto a chi amministra il sistema.

```
sudo usermod -aG docker $USER
newgrp docker
```

Blocco B.4-2 — procedura documentata (fonte [310]). Verifica dell'accesso alla GPU da un container.

```
docker run -it --gpus=all nvcr.io/nvidia/cuda:13.0.1-devel-ubuntu24.04 nvidia-smi
```

Esito atteso: informazioni sul dispositivo GPU, versione del driver e versione CUDA. [310] La pagina del runtime cita anche occupazione di memoria e temperatura, ma sulla GPU integrata il campo di occupazione della memoria risulta «Not Supported». [306]

Blocco B.4-3 — procedura documentata (fonte [310]). Diagnostica per runtime non trovato, discrepanze CUDA, permessi e avvio dei container.

```
nvidia-ctk --version
cat /etc/docker/daemon.json
sudo systemctl restart docker
groups $USER
ls -la /dev/nvidia*
docker run --rm --gpus=all nvcr.io/nvidia/cuda:13.0.1-devel-ubuntu24.04 echo "GPU test successful"
```

La pagina NGC della guida utente, aggiornata il 4 novembre 2026, richiede l'accesso autenticato al registro (chiave API personale NGC e docker login nvcr.io con utente $oauthtoken) per i container NIM e, nella procedura generale, premette allo scaricamento dei container la creazione della chiave API e il login, avvertendo che la chiave serve per scaricare i container e accedere alle risorse NGC, senza distinguere i container pubblici [1138]; la pagina del runtime non descrive alcun login. [310] La ricetta NVIDIA per vLLM indica che l'immagine nvcr.io/nvidia/vllm è pubblica e non richiede login e, se docker pull fallisce con unauthorized, suggerisce di rimuovere un login scaduto con docker logout nvcr.io. [1105]

B.5 Memoria disponibile, riserve e budget

Su memoria unificata `cudaMemGetInfo` non conteggia la memoria recuperabile da cache di pagina e swap e può riportare meno memoria di quella allocabile. La guida fornisce una funzione C d'esempio che legge `/proc/meminfo`: usa MemAvailable e SwapFree; se le huge page sono configurate, considera disponibili le huge page libere per la dimensione della pagina e azzera lo swap libero, perché le huge page non sono soggette a swap. [305], [306]

Blocco B.5-1 — adattamento editoriale (lettura in shell degli stessi campi usati dall'esempio C di [306]).

```
grep -E '^(MemTotal|MemAvailable|SwapFree|HugePages_Total|HugePages_Free|Hugepagesize):' /proc/meminfo
```

Esito atteso (stima): senza huge page, memoria allocabile stimata ≈ MemAvailable + SwapFree, secondo la logica dell'esempio; il ricorso allo swap ha però un costo di latenza non documentato per l'inferenza.

Blocco B.5-2 — procedura documentata (fonte [305]). Solo per il debug: svuotamento della buffer cache e riavvio dell'applicazione. La pagina del catalogo NGC del container vLLM suggerisce lo stesso svuotamento prima di avviare il server quando, su DGX Spark e Jetson Thor, l'avvio fallisce per esaurimento della memoria [313].

```
sudo sh -c 'sync; echo 3 > /proc/sys/vm/drop_caches'
```

Riserva del display. Il firmware UEFI riserva 2 GB al display per default; per carichi headless o ad alta memoria la guida indica l'impostazione `Auto` o `0.5 GB`, che rende disponibili fino a 1,5 GB in più. [305] Dalla release di ottobre 2026 i sistemi senza display collegato possono adeguarsi automaticamente al minimo di 512 MB [303]: la pagina UEFI Settings, aggiornata il 4 novembre 2026, descrive questo comportamento per l'impostazione Auto, che dimensiona la riserva sui display collegati all'avvio (0,5 GB su un sistema senza display), mentre il valore predefinito resta 2 GB [1137]. La guida di ottimizzazione documenta anche comandi Linux per impostare (`NvidiaDispFrmSizeRequest`) e leggere (`NvidiaDispFrmStatus`) la riserva, che non sono riportati in questa appendice: vanno ripresi dalla guida stessa [305]; la pagina UEFI Settings descrive invece la modifica dal menu di configurazione del firmware (Advanced → Advanced Menu → Display Reserved Memory; valori Auto, 0,5, 1, 2 e 4 GB), con il confronto di free -h prima e dopo il riavvio [1137].

Budget del runtime. In vLLM `--gpu-memory-utilization` è la frazione della memoria della GPU assegnata all'executor; pesi, lavoro del runtime e KV cache condividono quello spazio. [295] I container NGC 26.09 e 26.10 usano di fatto un valore vicino a 1,0 e su memoria unificata possono esaurire la memoria: in caso di esaurimento della memoria le note di entrambe le release indicano di avviare vllm serve con un valore più basso, per esempio 0,7. [314], [1127] La ricetta NVIDIA a nodo singolo per Qwen3.8-27B NVFP4, con l'immagine 26.10.pre1, usa invece 0,60 perché 0,70 ha causato arresti del nodo in alcune prove (sezione B.7) [1105]: la frazione va scelta per immagine e modello e controllata nei log di avvio. Con due istanze del runtime sullo stesso nodo le frazioni si sommano sulla stessa memoria fisica (inferenza dalla definizione).

Tabella B.1 — Foglio di calcolo del budget a nodo singolo (stima).

Voce	Formula o fonte	Esempio A1: coding, Qwen3.8-27B NVFP4, u = 0,87	Esempio A2: RAG privato, u = 0,7
Capacità nominale	Targa del sistema [233]	128 GB	128 GB
Budget del runtime	u × memoria totale riportata dal driver	111,4 GB	89,6 GB
Fuori budget: SO, display, servizi	128 GB − budget	16,6 GB	38,4 GB
Pesi del generatore	File del checkpoint nvidia/Qwen3.8-27B-NVFP4 [1067]; stima per componenti della sezione 1.5.14.4, Esempio 2 [99], [221]	21,92 GB di file (stima per componenti: circa 20,1 GB)	21,92 GB di file (stima per componenti: circa 20,1 GB)
KV cache, attivazioni, workspace	Budget − pesi	Fino a circa 89 GB, meno il workspace (N/D)	Fino a circa 68 GB, meno il workspace (N/D)
Servizi RAG su CPU	Indice + embedding + reranker	—	3,1–12,3 GB di indice FP32 per 10^6 chunk; 1,1–1,5 GB di embedding; reranker N/D

Voce	Formula o fonte	Esempio A1: coding, Qwen3.8-27B NVFP4, u = 0,87	Esempio A2: RAG privato, u = 0,7
Verifica	Log di avvio del runtime e `/proc/meminfo` [295], [306]	Capacità della KV cache e concorrenza sostenibile	MemAvailable sotto carico di ingestione

Fonti: [56], [99], [158], [221], [233], [295], [297], [306], [314], [322], [1067], [1105]. Note: calcoli editoriali su 128 GB nominali; pesi = somma dei file .safetensors del checkpoint alla revisione 482ca0f, invariata al 5/11/2026, come nella Tabella 1.42 e nell'Esempio 2 della sezione 1.5.14.4; u = 0,87 è il valore della ricetta NVIDIA a due nodi per i sistemi da 128 GB [297], riferito a un altro modello, non un'impostazione raccomandata a nodo singolo: la ricetta NVIDIA per Qwen3.8-27B NVFP4 su un DGX Spark da 128 GB usa u = 0,60, cioè 76,8 GB di budget, 51,2 GB fuori budget e fino a circa 54,9 GB per KV cache, attivazioni e workspace, da cui vanno sottratti i 3,85 GB del modello di bozza della ricetta (sezione 1.5.14.4), e segnala un blocco del nodo a 0,85 [1105]; l'indice è calcolato come $N\,d\,b$ con $N = 10^6$, d = 768–3.072 e 4 byte per coordinata (sezione 1.4.3); embedding da 0,56–0,74B parametri in BF16 a 2 byte per parametro; N/D = dato non attestato (non zero).

B.6 Percorso 1 — Ollama con Open WebUI

Percorso per l'interazione rapida da browser con modelli locali. Il playbook, aggiornato il 14 ottobre 2026, indica 15–20 minuti, rischio basso e la nota «The Open WebUI application operates offline, and your queries go to a model running locally». [319] Dalla v0.6.6 Open WebUI è distribuito con la Open WebUI License, che aggiunge alla BSD-3-Clause una clausola sul marchio: oltre 50 utenti finali in un periodo mobile di 30 giorni il marchio «Open WebUI» non può essere modificato, rimosso o sostituito senza permesso scritto o licenza enterprise; le condizioni vanno rilette sulla versione adottata (sezione 1.3.4). [1117]

Blocco B.6-1 — procedura documentata (fonte [320]). Verifica di Docker, scaricamento dell'immagine e avvio del container.

```
docker ps > /dev/null
docker pull ghcr.io/open-webui/open-webui:ollama
docker run -d -p 8080:8080 --gpus=all \
  -v open-webui:/app/backend/data \
  -v open-webui-ollama:/root/.ollama \
  --name open-webui ghcr.io/open-webui/open-webui:ollama
```

Passi successivi documentati: aprire `http://localhost:8080` in un browser locale; selezionare **Get Started**, compilare il modulo e registrare l'account amministratore locale; in **Select a model** digitare `gpt-oss:20b`, scaricarlo e selezionarlo; inviare un messaggio di prova. Esito atteso: l'interfaccia risponde su `http://localhost:8080` e il modello risponde nella chat. [320] Se `docker ps` restituisce un errore di permessi, si applica il Blocco B.4-1. Il percorso remoto tramite NVIDIA Sync usa la porta 12000. [319]

Blocco B.6-2 — adattamento editoriale (esposizione di rete e modalità solo locale). La pagina non tratta l'esposizione di rete; con `-p 8080:8080` la porta è pubblicata sull'host. Per limitarla all'interfaccia di loopback e raggiungerla da remoto con un tunnel SSH, e per chiedere a Ollama la modalità solo locale documentata nella sezione 1.5.8:

```
docker run -d -p 127.0.0.1:8080:8080 --gpus=all \
  -v open-webui:/app/backend/data \
  -v open-webui-ollama:/root/.ollama \
  -e OLLAMA_NO_CLOUD=1 \
  --name open-webui ghcr.io/open-webui/open-webui:ollama
```

La sintassi `-p 127.0.0.1:8080:8080` è documentata da Docker: con un indirizzo di loopback solo l'host raggiunge la porta pubblicata, mentre senza indirizzo la porta è pubblicata su tutti gli indirizzi dell'host; la stessa documentazione avverte che nelle release precedenti a Docker 28.0.0 host dello stesso segmento di rete locale potevano raggiungere anche le porte pubblicate su loopback, quindi la versione si controlla con docker version [1106]; se il container del Blocco B.6-1 esiste già, va prima arrestato e rimosso (`docker stop open-webui`, `docker rm open-webui`) senza rimuovere i volumi. Dopo il download dei modelli necessari si possono aggiungere le opzioni di Open WebUI della sezione 5.4.9.4 (`-e OFFLINE_MODE=true -e ENABLE_VERSION_UPDATE_CHECK=false -e ENABLE_COMMUNITY_SHARING=false`), che non bloccano le connessioni uscenti; ENABLE_COMMUNITY_SHARING è però una variabile ConfigVar:

dopo il primo avvio prevale il valore salvato nel database del volume open-webui, quindi va passata già al primo avvio, che non richiede download, oppure disattivata dal pannello di amministrazione [981]; l'account amministratore va registrato subito, perché la registrazione resta aperta finché il primo utente non la completa (sezione 5.4.9.6). Il bind su loopback non va sostituito dal solo firewall dell'host: secondo la documentazione Docker il traffico verso le porte pubblicate dai container viene instradato prima delle regole gestite con ufw, che quindi non lo filtrano [1107]; la variabile `OLLAMA_NO_CLOUD=1` è documentata da Ollama [213], ma il suo effetto nell'immagine combinata Open WebUI–Ollama non è verificato: secondo la FAQ, con le funzioni cloud disattivate il log del server Ollama riporta «Ollama cloud disabled: true» [213], riga da cercare con docker logs open-webui; che l'immagine combinata inoltri i log di Ollama all'output del container non è verificato. L'isolamento va comunque controllato sul traffico uscente (sezione 5.4.9).

Blocco B.6-3 — procedura documentata (fonte [320]). Aggiornamento successivo dell'immagine.

```
docker pull ghcr.io/open-webui/open-webui:ollama
```

La pagina non descrive la ricreazione del container dopo l'aggiornamento dell'immagine. Adattamento editoriale, non provato: prima dell'aggiornamento si arresta il container (docker stop open-webui) e si salva il volume open-webui, che contiene account e cronologia, per esempio con sudo tar -C "$(docker volume inspect --format '{{ .Mountpoint }}' open-webui)" -czf "$HOME/open-webui-backup.tar.gz" . (la copia contiene dati personali, va protetta come indicato nella sezione 5.4.9.7 e si reintegra a container arrestato se il ritorno alla versione precedente lo richiede), e si annota l'immagine in uso (`docker inspect --format '{{.Image}}' open-webui`); dopo `docker pull` il container si arresta e si rimuove (`docker stop open-webui`, `docker rm open-webui`, senza `docker volume rm`) e si ricrea con lo stesso comando del Blocco B.6-1 o B.6-2; il ritorno alla versione precedente usa l'immagine annotata. La guida di hardening di Open WebUI raccomanda release con tag anziché etichette mobili e una validazione prima dell'esercizio (sezione 5.4.9.8): il tag `ollama` è mobile. Il registro GHCR pubblica anche tag con versione nel formato vX.Y.Z-ollama, con varianti linux/amd64 e linux/arm64: il 25/10/2026 il più recente era v0.12.4-ollama, corrispondente alla release v0.12.4 e con un indice diverso da quello corrente di ollama [1108]; al 5/11/2026 è v0.13.0-ollama (digest sha256:1c3661c6b30d79232973eb4ae36be6db3c15ef58eb23ca61af70cb50922bd0ab, con varianti linux/amd64 e linux/arm64), della release v0.13.0 del 3 novembre 2026, preceduta dalla v0.12.5 del 29 ottobre: le note della v0.13.0 segnalano modifiche dello schema del database, raccomandano negli ambienti di produzione il backup del database e dei dati associati prima dell'aggiornamento e, nelle installazioni con più worker, più server o bilanciamento del carico, chiedono di aggiornare insieme tutte le istanze, senza aggiornamenti progressivi [1108], [1139]. Per l'esercizio si fissa un tag con versione, o il suo digest, dopo la validazione.

Blocco B.6-4 — procedura documentata (fonte [320]). Rimozione completa; cancella definitivamente dati di Open WebUI e modelli scaricati.

```
docker stop open-webui
docker rm open-webui
docker rmi ghcr.io/open-webui/open-webui:ollama
docker volume rm open-webui open-webui-ollama
```

B.7 Percorso 2 — vLLM con API compatibile OpenAI

Percorso per qualità elevata, coding e agenti con tool calling. Il playbook a nodo singolo, aggiornato il 19 ottobre 2026, prevede un percorso intermedio con script di avvio per NVIDIA Sync e uno avanzato con script di avvio manuali, verifiche dell'API e del tool calling; per DGX Spark la ricetta indicata, per 64 e 128 GB, è `qwen/qwen3.8-27b` in NVFP4. [294], [295] Il comando di avvio non compare nella pagina del playbook, che indica di comporre lo script launch-vllm.sh dai comandi «Start container» e «Launch» della pagina della ricetta: le impostazioni d'ambiente del server diventano opzioni -e, --entrypoint /bin/bash diventa --entrypoint vllm, dopo il nome dell'immagine si aggiungono serve, il checkpoint e gli argomenti provati, si conservano i passi di preparazione dell'host e si assegna al container il nome vllm-

local. [295] La ricetta NVIDIA collegata per 128 GB, validata dal produttore il 19/10/2026 con driver 580.126.20 e CUDA 13.4, avvia l'immagine nvcr.io/nvidia/vllm:26.10.pre1-py3 con --entrypoint /bin/bash e, nel container, vllm serve nvidia/Qwen3.8-27B-NVFP4 con --host 0.0.0.0, --kv-cache-dtype fp8, --max-model-len 262144, --max-num-seqs 8, --gpu-memory-utilization 0.60 e decodifica speculativa con il modello di bozza di terzi z-lab/Qwen3.8-27B-DFlash2. La ricetta motiva il valore 0,60 con un blocco del nodo a 0,85 e con arresti a 0,70 in alcune prove, si qualifica come configurazione di valutazione e chiede, per l'esercizio, di fissare immagine, modello e revisioni del codice; il repository avverte che le ricette vanno riesaminate e validate in modo indipendente prima dell'uso. [1105] Al 5 novembre 2026 la ricetta è invariata (release v0.66 del 24/10/2026, unico commit del repository) e la scheda Intermediate del playbook indica ancora nvcr.io/nvidia/vllm:26.10.pre1-py3 per lo stesso checkpoint sui sistemi da 128 GB [1120], mentre il catalogo NGC pubblica dal 29/10/2026 il tag finale 26.10-py3, con note del 30/10/2026 che indicano vLLM 0.31.0 e CUDA 13.4.3 [313], [1127]: sostituire nello script il tag della ricetta equivale a una nuova configurazione, da validare come tale. La ricetta avverte inoltre che il prefix caching, attivo per default, dipende dalla correzione vllm#60252 inclusa nell'immagine 26.10.pre1: senza di essa ogni riuso della cache dei prefissi con il modello di bozza DFlash2 restituisce testo corrotto (vllm#60174), e il comando non va eseguito su un'immagine priva della correzione [1105]; le note della 26.10 non la citano [1127].

Blocco B.7-1 — procedura documentata (fonte [295]). Avvio con lo script di ricetta e lettura dei log del container vllm-local.

```
chmod +x launch-vllm.sh
./launch-vllm.sh
docker logs --tail 50 --follow vllm-local
```

Nei log di avvio si legge la stima della capacità della KV cache: il playbook avverte che un limite di 262.144 token con `--max-num-seqs 8` non garantisce otto richieste di lunghezza piena contemporanee. [295]

Blocco B.7-2 — procedura documentata (fonte [295]). Verifiche dell'API.

```
curl --fail-with-body -i http://localhost:8000/health
curl --fail-with-body -sS http://localhost:8000/v1/models
curl --fail-with-body -sS http://localhost:8000/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -d '{"model":"<model-id>","messages":[{"role":"user","content":"Reply with a short
greeting."}],"max_tokens":4096}'
```

Esito atteso: HTTP 200 per `/health`; un array `data` non vuoto per `/v1/models`, da cui si copia l'`id` del modello, che può differire dal nome del checkpoint; per la chat, una risposta in `choices[0].message.content`. Se `finish_reason` è `length`, il ragionamento può avere consumato il budget di token: si aumenta `max_tokens` o si seguono le impostazioni della ricetta. [295]

Blocco B.7-3 — procedura documentata (fonte [295]). Verifica del tool calling.

```
curl --fail-with-body -sS http://localhost:8000/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -d '{"model":"<model-id>","messages":[{"role":"user","content":"Use get_weather to look up
the weather in Boston."}],
"tools":[{"type":"function","function":{"name":"get_weather",
"description":"Get current weather for a location.",
"parameters":{"type":"object","properties":{"location":{"type":"string"}},"required":["locatio
n"]}}}],
"tool_choice":"auto","max_tokens":4096}'
```

Esito atteso: una voce `tool_calls` per `get_weather` con Boston negli argomenti e `finish_reason` uguale a `tool_calls`; se il modello risponde in testo libero vanno controllate le impostazioni di tool calling della ricetta e il supporto del modello. [295] L'esecuzione dello strumento spetta al client; i controlli sugli strumenti sono trattati nelle sezioni 3.4 e 5.1.

Blocco B.7-4 — procedura documentata (fonte [295]). Accesso da un portatile con tunnel SSH e arresto del servizio; come indica la pagina, user@dgx-host va sostituito con l'utente SSH e l'indirizzo del sistema.

```
# sul portatile
ssh -N -L 8000:127.0.0.1:8000 user@dgx-host
# sul sistema GB10, per arrestare il servizio
docker stop vllm-local
```

Il playbook avverte che il bind su `0.0.0.0` della ricetta può esporre un'API non autenticata su tutte le interfacce del dispositivo: servono una rete fidata o controlli d'accesso, mentre per client locali e tunnel SSH basta l'interfaccia di loopback. [295] Il bind su loopback non limita però le richieste inviate dai browser: le versioni di vLLM precedenti alla 0.32.0 ammettono per default richieste cross-origin da qualunque origine (--allowed-origins con valore predefinito ["*"]) e, senza --api-key, non richiedono autenticazione: una pagina web aperta nel browser del sistema o del portatile collegato con il tunnel può quindi inviare richieste all'API e leggerne le risposte; dalla 0.32.0 il valore predefinito è un elenco vuoto (avviso GHSA-vvpg-rrww-wpmh, CVE-2026-115917) [1140]. Le immagini 26.10.pre1-py3 della ricetta e 26.10-py3 del Blocco B.7-5 contengono vLLM 0.31.0 in un build di NVIDIA, e né la ricetta né le note della 26.10 indicano se quel build includa la correzione [1105], [1127]: con queste immagini l'opzione va quindi impostata in modo esplicito con l'elenco JSON delle sole origini fidate, per esempio --allowed-origins '[]' se nessuna applicazione web deve usare l'API, a meno che il valore predefinito letto nel container con vllm serve --help=allowed-origins sia già un elenco vuoto (adattamento editoriale, non provato); i valori predefiniti degli altri server sono nella Tabella 5.15 (sezione 5.4.9.6).

Blocco B.7-5 — adattamento editoriale (avvio manuale con il container NGC, non provato). Composizione dei parametri documentati quando non si usa lo script di ricetta: opzioni Docker della ricetta a due nodi [297], tag NGC finale più recente al 5/11/2026 [313], frazione di memoria del problema noto [314], [1127] e bind su loopback [295]. La scheda Intermediate dello stesso playbook abbina invece alla ricetta Qwen3.8-27B NVFP4 per 128 GB l'immagine nvcr.io/nvidia/vllm:26.10.pre1-py3 e il modello di bozza z-lab/Qwen3.8-27B-DFlash2 [1120]: il tag qui usato, 26.10-py3, non è quello della ricetta, e la variante non usa la bozza.

```
export IMAGE="nvcr.io/nvidia/vllm:26.10-py3"
export HF_CACHE_DIR="$HOME/.cache/huggingface"
docker run --rm -it \
  --name vllm-local --entrypoint /bin/bash \
  --gpus all --ipc host --network host \
  --shm-size=16g \
  -e HF_TOKEN="${HF_TOKEN:-}" \
  -v "$HF_CACHE_DIR:/root/.cache/huggingface" \
  -v "$HOME/.cache/vllm:/root/.cache/vllm" \
  "$IMAGE"
```

La riga `-e HF_TOKEN=…` serve soltanto al primo download di modelli ad accesso controllato, da eseguire in staging (sezione 5.4.9.3): passato con `-e`, il token resta leggibile nell'ambiente del container e nella sua configurazione (docker inspect): conviene un token a sola lettura e con ambito limitato, come raccomanda la ricetta NVIDIA [1105]. Per un nodo in esercizio, con la cache già popolata, la riga si sostituisce con `-e HF_HUB_OFFLINE=1 -e VLLM_NO_USAGE_STATS=1 -e DO_NOT_TRACK=1`, perché le opzioni offline e anti-telemetria della sezione 5.4.9.4 devono trovarsi nell'ambiente del processo che le legge (adattamento editoriale, non provato).

All'interno del container:

```
vllm serve <model-id> \
  --host 127.0.0.1 \
  --port 8000 \
  --gpu-memory-utilization 0.7 \
  --max-model-len <token> \
  --max-num-seqs <sequenze> \
  --enable-prefix-caching \
  --enable-chunked-prefill
```

La pagina del catalogo riporta docker run --gpus all -it --rm nvcr.io/nvidia/vllm:xx.xx-py3 e l'avvio del server nel container con python3 -m vllm.entrypoints.openai.api_server [313]; le note 26.09 e 26.10 usano vllm serve nei loro esempi [314], [1127]. Il tag del blocco, 26.10-py3, è il più recente del catalogo al 5/11/2026: aggiornato il 29/10/2026, ha secondo l'API del catalogo l'indice multi-architettura

sha256:0a0de9362545e3e062750bec4ad4f722422a04a81182b5258a7deda4325e9313, con varianti linux/arm64 e linux/amd64 [1071]; le note della 26.10 indicano vLLM 0.31.0, CUDA 13.4.3 e PyTorch 2.15.0a0 [1127] e la configurazione arm64 di quell'indice dichiara il punto d'ingresso /opt/nvidia/nvidia_entrypoint.sh, senza comando predefinito, con VLLM_VERSION=0.31.0+88641915, CUDA_VERSION=13.4.3.001 e CUDA_DRIVER_VERSION=615.71.09 [1141]. Per la 26.09-py3, usata nell'edizione di ottobre, il registro letto il 25/10/2026 indicava le varianti linux/amd64 e linux/arm64 e, per arm64, il punto d'ingresso /opt/nvidia/nvidia_entrypoint.sh, senza comando predefinito, con VLLM_VERSION=0.29.0+5013de39 e CUDA_VERSION=13.4.1.012 [1109]; il comportamento dello script senza argomenti non è stato verificato. La ricetta a due nodi [297] e la ricetta NVIDIA a nodo singolo con l'immagine 26.10.pre1 [1105] avviano il container con --entrypoint /bin/bash, ripreso nel blocco, e aggiungono --ulimit memlock=-1 --ulimit stack=67108864; la seconda monta soltanto la cache Hugging Face e avverte che montare cartelle sopra l'intera /root/.cache nasconde le cache precompilate di kernel e autotuning dell'immagine; le note della 26.10 dichiarano cache precompilate di kernel, compilazione e autotuning per le configurazioni Qwen3.8 27B e Flash-Next su DGX Spark, senza indicarne il percorso [1127]: con quei modelli il montaggio di /root/.cache/vllm del blocco potrebbe nasconderne una parte e conviene montare soltanto la cache Hugging Face, come nella ricetta (inferenza, non verificata). Come in [297], le cartelle montate si creano con mkdir -p prima del docker run: in mancanza, con -v le crea il demone Docker [1110]. La compatibilità minore CUDA 13.x ammette in generale driver almeno R580 con limitazioni; CUDA 13.4 corrisponde al ramo R615 e nuove funzioni o PTX possono richiedere driver più recenti. Non è stato eseguito il collaudo dei container 26.09 e 26.10 su GB10 con il driver di sistema. [1026], [1027], [314] Per i modelli Nemotron Nano V3 e Super V3 in NVFP4 il problema noto, ripetuto nelle note della 26.10, richiede --max-num-seqs 4. [314], [1127]

B.8 Percorso 3 — llama.cpp con modelli GGUF

Il playbook, aggiornato il 9 ottobre 2026, compila llama.cpp da sorgente con CUDA; servono circa 30 GB di memoria libera e circa 40 GB di disco per il download d'esempio e gli artefatti di compilazione, una conversione GGUF di terzi da circa 21 GB. [298]

Aggiornamento documentale del 25 ottobre 2026, riverificato al 5 novembre 2026. La prerelease llama.cpp b11871, pubblicata il 25 ottobre alle 06:28 UTC, introduce un limite di 128 livelli per i JSON di richiesta prima della visita ricorsiva, per evitare arresti del server descritti dagli autori; la nota copre anche schemi e argomenti dei tool e il corpo di /v1/systemone. È una correzione dichiarata dal progetto, non collaudata qui su GB10: fissare una revisione che la includa e verificarla nell'ambiente di prova prima dell'uso operativo. Ollama v0.40.11 risulta pubblicato il 24 ottobre; la generazione vocale annunciata riguarda macOS e non prova disponibilità su GB10. [1029], [1030] Al 5 novembre 2026 il repository llama.cpp pubblica anche release versionate: la v0.9.0 del 4 novembre, contrassegnata come «Latest», corrisponde alla build b12110 e segue di 239 commit la b11871 sul ramo master, di cui include quindi la correzione, da collaudare come ogni revisione [1142]. Ollama è alla v0.40.15 del 3 novembre; le note delle versioni da v0.40.12 a v0.40.15 non citano GB10 né DGX Spark [1143].

Blocco B.8-1 — procedura documentata (fonte [318]). Prerequisiti, clonazione e compilazione.

```
sudo apt update
sudo apt install -y git clang cmake libcurl4-openssl-dev libssl-dev
git clone https://github.com/ggml-org/llama.cpp ~/llama.cpp
cd ~/llama.cpp
cmake -B build -DGGML_NATIVE=ON -DGGML_CUDA=ON -DGGML_CURL=ON -DGGML_RPC=ON -
DCMAKE_CUDA_ARCHITECTURES=121a-real
cmake --build build --config Release --target llama-server -j
```

Esito atteso: compilazione in circa 5–10 minuti, con `llama-server` in `build/bin/`. [318] La pagina non fissa tag né commit. L'opzione `-DGGML_CURL=ON` del playbook non compare nella guida di compilazione di llama.cpp (ramo master, consultata il 25/10/2026 e riverificata al 5/11/2026), che per le funzioni HTTPS indica le librerie di sviluppo OpenSSL (libssl-dev) e avverte che senza di esse il progetto si

compila e funziona senza supporto SSL [1118]: nel log di configurazione va verificato che il supporto SSL, necessario per i download con `-hf`, risulti attivo.

Blocco B.8-2 — adattamento editoriale (riproducibilità). Registrazione del commit compilato, perché la procedura non ne fissa uno.

```
git -C ~/llama.cpp rev-parse HEAD
```

Blocco B.8-3 — procedura documentata (fonte [318]). Avvio del server con scaricamento del modello; le ultime tre opzioni sono quelle indicate per la variante MTP.

```
./build/bin/llama-server \
  -hf unsloth/Qwen3.6-35B-A3B-MTP-GGUF:UD-Q4_K_XL \
  --host 0.0.0.0 \
  --port 30000 \
  --chat-template-kwargs '{"preserve_thinking": true}' \
  --spec-type draft-mtp \
  --spec-draft-n-max 3
```

Esito atteso: nei log, per ciascuno slot, `n_ctx = 262144`, un limite della cache dei prompt di 8192 MiB e le righe «model loaded», «server is listening on http://0.0.0.0:30000» e «all slots are idle». Per default `llama-server` tenta di contenere l'intero contesto del modello servendo quattro richieste concorrenti e adatta i parametri se necessario; per lavoro agentico o di coding servono almeno 32.768 token di contesto, preferibilmente 100.000 o più. [318] La pagina non ha una sezione sulla sicurezza: con `--host 0.0.0.0` il server ascolta su tutte le interfacce ed è raggiungibile da altri dispositivi su `http://<HARDWARE_IP>:30000/v1`, senza autenticazione descritta. [318]

Blocco B.8-4 — adattamento editoriale (bind su loopback). Per l'uso locale o tramite tunnel SSH si sostituisce soltanto l'indirizzo di ascolto, nel blocco seguente, che corrisponde all'avvio di base della pagina [318]; per la variante MTP vi si aggiungono, invariate, le opzioni --chat-template-kwargs, --spec-type draft-mtp e --spec-draft-n-max 3 del Blocco B.8-3; dopo il primo download si può aggiungere `--offline`, che impone l'uso della cache e impedisce l'accesso alla rete (sezione 5.4.9.4):

```
./build/bin/llama-server \
  -hf unsloth/Qwen3.6-35B-A3B-MTP-GGUF:UD-Q4_K_XL \
  --host 127.0.0.1 \
  --port 30000
```

Blocco B.8-5 — procedura documentata (fonte [318]). Attesa della disponibilità e richiesta di prova. Le righe sono quelle della pagina; le parentesi del corpo JSON sono ricomposte. La prima riga è pensata per uno script: incollata in una shell interattiva, `|| exit 1` chiude la sessione, anche SSH, se il server non risponde entro 900 s; per l'uso manuale conviene sostituire `exit 1` con un messaggio e con la lettura dei log (adattamento editoriale).

```
timeout 900 bash -c 'until curl -sf http://127.0.0.1:30000/health > /dev/null 2>&1; do sleep
5; done' || exit 1
curl -X POST http://127.0.0.1:30000/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "unsloth/Qwen3.6-35B-A3B-MTP-GGUF:UD-Q4_K_XL",
    "messages": [{"role": "user", "content": "New York is a great city because..."}],
    "max_tokens": 100
  }'
```

Esito atteso: con 100 token la risposta può avere `finish_reason: "length"` e `content` vuoto, perché il modello spende il budget nel `reasoning_content`; la prova successiva della pagina, con 2.048 token, è pensata per produrre una risposta finale in `content`. [318]

Blocco B.8-6 — procedura documentata (fonte [318]). Arresto e rimozione: il server si arresta con Ctrl+C, poi si eliminano clone e modello.

```
rm -rf ~/llama.cpp
rm -rf ~/.cache/huggingface/hub/models--unsloth--Qwen3.6-35B-A3B-MTP-GGUF
```

B.9 Percorso 4 — SGLang

Il playbook SGLang, aggiornato il 31 luglio 2026, usa un container «pre-built CUDA 13» con Docker e NVIDIA Container Toolkit e indica come validati su DGX Spark, fra gli altri, gpt-oss-20b e gpt-oss-120b in MXFP4, Llama-3.1-8B-Instruct in FP8 e NVFP4, Llama-3.3-70B-Instruct in NVFP4 e Qwen3-32B in NVFP4; per i checkpoint NVFP4 occorre aggiungere `--quantization modelopt_fp4` al comando di avvio. Il primo avvio richiede più tempo per il download e la cattura dei CUDA graph, fino a 45–60 minuti per i MoE più grandi; il ripristino consiste nell'arresto e nella rimozione del container. [317] La scheda Instructions [1017] usa l'immagine `lmsysorg/sglang:latest-cu130`, avvia il container `sglang-server` con `--gpus all --ipc host --cap-add SYS_NICE --ulimit memlock=-1 --ulimit stack=67108864 -p 30000:30000 -e HF_TOKEN` e la cache `$HOME/.cache/huggingface/hub` montata, e lancia `sglang serve --model-path "$MODEL_HANDLE" --host 0.0.0.0 --port 30000 --context-length $MAX_MODEL_LEN --mem-fraction-static 0.85 --attention-backend flashinfer --enable-cache-report --trust-remote-code`; per DGX Spark suggerisce `--mem-fraction-static 0.75` sotto pressione di memoria. La verifica attende `/health` fino a 900 s e interroga `/v1/chat/completions`; il ripristino usa `docker stop sglang-server`, `docker rm sglang-server`, `docker rmi "$SGLANG_IMAGE"` e la rimozione con `sudo` della cartella del modello, scaricata come root. Chi trascrive la procedura in comandi applica tre adattamenti editoriali: porta pubblicata su 127.0.0.1, --trust-remote-code solo se il checkpoint lo richiede e dopo la revisione del codice (sezione 5.4.9.3), tag dell'immagine fissato al posto di latest-cu130 (il 25/10/2026 v0.5.22-cu130 aveva lo stesso digest di latest-cu130; al 5/11/2026 lo ha v0.5.23-cu130, pubblicato il 30/10/2026, con indice sha256:f832b5c39559c9caa900b22ad37c76fcce66470676855c42052102c537812f1f e variante linux/arm64 [1111], ma nessuno dei due tag è indicato come validato dal playbook); la cartella da rimuovere va indicata per nome esatto (models--<organizzazione>--<modello>), mai con un segnaposto vuoto, che con sudo rm -rf cancellerebbe l'intera cache. Le misure di terzi con SGLang su DGX Spark sono nella sezione 1.5.2. [173] La porta 30000 è usata anche da llama-server nel Blocco B.8-3: i due percorsi non vanno avviati insieme senza cambiare la porta di uno dei due (inferenza dai comandi riportati).

B.10 Monitoraggio

Blocco B.10-1 — procedura documentata (fonte [323]). Accesso remoto alla DGX Dashboard con tunnel SSH, poi apertura di `http://localhost:11000`.

```
ssh -L 11000:localhost:11000 <username>@<IP or spark-abcd.local>
```

La Dashboard offre monitoraggio in tempo reale e JupyterLab integrato; la pagina non elenca le metriche né il metodo di calcolo della memoria. [323]

Blocco B.10-2 — procedura documentata (fonte [305]). Installazione di `perf` per la profilazione della CPU; la registrazione usa `sudo perf record` con gli eventi indicati dalla guida e la lettura `perf report`.

```
sudo apt install linux-tools-$(uname -r)
```

L'abilitazione dei contatori può richiedere `kernel.perf_event_paranoid=-1` e `kernel.kptr_restrict=0` in `/etc/sysctl.d`; la guida avverte che registrare tracce `perf` ha implicazioni di sicurezza. [305] Per la GPU la guida indica Nsight Systems e Nsight Compute.

Letture di memoria: `nvidia-smi` non riporta l'occupazione complessiva sulla GPU integrata, `cudaMemGetInfo` la sottostima, `/proc/meminfo` (Blocco B.5-1) e i log dei runtime (Blocchi B.7-1 e B.8-3) danno la disponibilità e la capacità effettiva della KV cache. [295], [306], [318]

B.11 Verifica finale

Tabella B.2 — Verifiche a nodo singolo: comando ed esito atteso.

Verifica	Comando o azione	Esito atteso	Etichetta e fonte
Driver e CUDA	`nvidia-smi`; `nvcc --version`	Versioni coerenti con la Tabella 1.41	Documentato [312]; confronto con [303]

Verifica	Comando o azione	Esito atteso	Etichetta e fonte
Container Toolkit	`nvidia-ctk --version`	Versione ≥ 1.17.7	Documentato [309], [310]
GPU nel container	Blocco B.4-2	Dispositivo, driver e versione CUDA	Documentato [310]
Memoria disponibile	Blocco B.5-1	MemAvailable compatibile con il budget della Tabella B.1	Adattamento di [306]
API vLLM	Blocco B.7-2	HTTP 200; `data` non vuoto; risposta in `content`	Documentato [295]
Tool calling	Blocco B.7-3	`tool_calls` e `finish_reason` = `tool_calls`	Documentato [295]
llama.cpp	Log del Blocco B.8-3	«server is listening…», «all slots are idle»	Documentato [318]
SGLang	Attesa di `/health` fino a 900 s, poi richiesta a `/v1/chat/completions` (sezione B.9)	Risposta del modello	Documentato [1017]
Open WebUI	Browser su `http://localhost:8080`	Account locale creato; risposta del modello	Documentato [320]
Indirizzi di ascolto	`sudo ss -ltnp`	Con gli adattamenti B.6-2, B.7-5, B.8-4 e B.9, servizi su 127.0.0.1	Adattamento editoriale
Dashboard	Blocco B.10-1	Pagina raggiungibile su `http://localhost:11000`	Documentato [323]

Fonti: [295], [303], [306], [309], [310], [312], [318], [320], [323], [1017]. Note: `ss -ltnp` è un comando standard di Linux non citato dalle pagine NVIDIA consultate; nessuna verifica è stata eseguita per questa edizione.

B.12 Ripristino

- **Open WebUI e Ollama:** Blocco B.6-4, documentato; elimina dati e modelli. [320]
- **vLLM:** `docker stop vllm-local`, documentato. [295] La rimozione dell'immagine non è descritta dalla pagina; per analogia con il Blocco B.6-4 si usa `docker rmi <immagine>` (adattamento editoriale).
- **llama.cpp:** Blocco B.8-6, documentato. [318]
- **Cache dei modelli Hugging Face:** per analogia con il Blocco B.8-6, la cartella del modello in `~/.cache/huggingface/hub/`, indicata per nome esatto (adattamento editoriale); i file scaricati da un container eseguito come root, come nel Blocco B.7-5 e in SGLang, appartengono a root e si rimuovono con sudo [1017]; le cache di vLLM, FlashInfer e Triton risiedono nella home dell'utente. [297]
- **Gruppo docker:** la rimozione dell'utente dal gruppo non è descritta dalle pagine consultate; il comando standard di Ubuntu è `sudo gpasswd -d $USER docker` (adattamento editoriale).
- **Politica della Dashboard:** ripristinare la copia settings.json.bak del Blocco B.3-3 oppure, se il file non esisteva, impostare "enabled": true, valore predefinito del campo; poi riavviare il servizio con il comando del Blocco B.3-2. [323]
- **Contatori di `perf`:** rimuovere il file creato in `/etc/sysctl.d` e riportare kernel.perf_event_paranoid e kernel.kptr_restrict ai valori letti prima della modifica (sysctl kernel.perf_event_paranoid kernel.kptr_restrict), con sudo sysctl -w oppure con un riavvio: ricaricare i file non ripristina un parametro che nessun altro file imposta (adattamento editoriale).
- **Riserva del display:** ripristinare il valore precedente dal menu di configurazione UEFI (Advanced → Advanced Menu → Display Reserved Memory, predefinito 2 GB), come descrive la pagina UEFI Settings, che per i problemi successivi a una modifica indica il ritorno ai valori predefiniti UEFI e il riavvio; se la modifica è stata fatta da Linux, valgono i comandi della guida di ottimizzazione. [305], [1137]
- **Aggiornamenti di sistema:** nessun rollback documentato; se dopo un aggiornamento il sistema non si avvia, la pagina System Repair descrive l'avvio di un kernel installato in precedenza dal menu GRUB o di Ubuntu da USB, senza cancellare l'SSD [1136]; in ultima istanza, ripristino tramite System Recovery, con immagini su USB, che riporta sistema operativo e firmware allo stato di fabbrica e cancella completamente l'SSD interno: dati, modelli, volumi dei container e configurazioni vanno salvati prima dell'aggiornamento; per i sistemi OEM immagini e procedure sono dei rispettivi produttori. [227], [311], [1103]

B.13 Limiti della procedura

- Nessun comando è stato eseguito e nessuna misura propria di throughput o latenza è disponibile per questa edizione: le procedure sono riportate dalle fonti, gli adattamenti e le stime sono dichiarati come tali. Per la ricetta NVIDIA a nodo singolo della sezione B.7 la raccolta di ricette di NVIDIA per l'hardware locale (versione v0.66 del 24 ottobre 2026) dichiara 37,29 token/s di decodifica, valore medio misurato il 19/10/2026 con aiperf 0.12.0 su SPEED-Bench (profilo qualitative, categoria coding; 10 richieste, 8.192 token in uscita) a concorrenza 1, con la decodifica speculativa della ricetta (modello di bozza DFlash2, 7 token speculativi), --gpu-memory-utilization 0.60, il driver 580.126.20 e la revisione 482ca0f3832238542f8f5295dde86b5f22711d80 del checkpoint, corrente al 5/11/2026: è una dichiarazione del produttore, non riprodotta, ottenuta con un driver diverso da quello corrente della Founders Edition (580.178.04) e pubblicata senza il tasso di accettazione dei token speculativi, senza il quale non è confrontabile con il limite ideale della sezione 1.5.15.5. [1105], [1131]
- Diverse schede dei playbook sono state lette solo in parte: per vLLM la ricetta NVIDIA a nodo singolo è riassunta nella sezione B.7 senza un blocco eseguibile e il contenuto di launch-vllm.sh dipende dalla ricetta scelta; la scheda Instructions di SGLang, letta il 25/10/2026 in sede di riesame e riverificata al 5/11/2026, è riassunta nella sezione B.9 senza blocchi eseguibili; le citazioni testuali dello strumento di consultazione sono limitate a brevi estratti, e alcune righe di comando sono state riportate riga per riga.
- Versioni e procedure valgono per la Founders Edition al 5/11/2026; tag come `ollama` e i build notturni comunitari cambiano nel tempo.
- GPUDirect Storage e le metriche della Dashboard non sono documentati nelle pagine consultate. La regola generale di compatibilità minore CUDA è stata riesaminata; il collaudo driver–container per CUDA 13.4.1 e 13.4.3 (container 26.09 e 26.10) su GB10 resta non eseguito. [1026], [1027]

Appendice C — Procedura documentata: due sistemi GB10 collegati (2 × 128 GB)

Questa appendice raccoglie le procedure ufficiali NVIDIA per collegare due sistemi GB10 con un cavo QSFP diretto, verificare il collegamento e servire un modello ripartito su entrambi i nodi, come risultano dalle pagine consultate il 25 ottobre 2026 e riverificate con lo stato al 5 novembre 2026. È il complemento operativo della Scheda B (sezione 1.5.15.4). Nessun comando è stato eseguito: le etichette **procedura documentata**, **adattamento editoriale** e **stima** hanno il significato definito nell'Appendice B.1. Due sistemi collegati restano due nodi con memorie distinte: nessuna procedura di questa appendice crea una memoria coerente da 256 GB, e nessuna promette prestazioni doppie.

C.1 Topologia, ruoli e prerequisiti

La topologia trattata è quella di due sistemi collegati direttamente, senza switch. [172] Il Cluster Assistant di NVIDIA Sync supporta fino a tre sistemi collegati direttamente e fino a quattro tramite switch; per tre e quattro nodi il playbook vLLM prevede soltanto il percorso NVIDIA Sync, senza esempio a riga di comando. [302], [307] Il blog NVIDIA del 21 ottobre 2026 dichiara cluster fino a otto unità in configurazione manuale [246]; al 5 novembre 2026 la guida di NVIDIA Sync, aggiornata il 30 ottobre, limita ancora il Cluster Assistant a 2–4 sistemi, annuncia le otto unità per una release successiva e per più di quattro sistemi rimanda alla scheda di configurazione manuale [1112], mentre la scheda «Cable the Devices» indica che NVIDIA Sync e lo script fornito supportano al più quattro sistemi tramite switch e che i cluster più grandi richiedono una configurazione manuale o un'automazione propria, senza fornirne la procedura [325]. Il nodo 1 è il nodo head, rank 0 del runtime distribuito; il nodo 2 è il worker, rank 1. [297]

Prerequisiti documentati:

- due sistemi con la stessa quantità di memoria, 64 o 128 GB [302];
- DGX OS aggiornato almeno alla release di aprile 2026 [325];
- i due sistemi sulla stessa LAN di gestione, raggiungibili per indirizzo IP o nome mDNS, con credenziali `sudo` su ciascuno [325];
- lo stesso nome utente su tutti i nodi, perché gli script lo presuppongono [312], [326];
- Docker installato e utilizzabile dall'utente e Hugging Face CLI installata e autenticata su ciascun nodo; il gestore di pacchetti uv sul nodo head [302];
- per il percorso manuale: Netplan con il modulo Python `yaml`, Python 3.10 o successivo, client e server OpenSSH, `ip`, `ping`, `ssh-keygen` e `runuser`; interfacce ConnectX-7 con i nomi standard, non rinominate, non in bonding, bridge o VLAN [326];
- un cavo QSFP approvato (Sezione C.2).

Lo script del percorso manuale non installa pacchetti e chiede la password quando serve. [326]

C.2 Cablaggio

Ogni sistema ha due porte QSFP ConnectX-7, fino a 200 Gb/s per porta e configurabili soltanto in Ethernet; il cavo determina la velocità in ingresso. Servono cavi da almeno 200 Gb/s, e un cavo più veloce non aumenta la velocità della porta. Cavi approvati: Amphenol NJAAKK-N911, QSFP–QSFP112, 32 AWG, 400 mm, LSZH, con NJAAKK0006 nella versione da 0,5 m; Luxshare LMTQF022-SD-R, DAC QSFP112 400G, 400 mm, 30 AWG. [307]

Le due porte sono fisicamente intercambiabili, ma i playbook presuppongono porte specifiche: la porta sinistra è quella più vicina alla porta Ethernet guardando il retro del sistema. Il cavo si inserisce con la linguetta rivolta verso l'alto, senza forzare, perché la forzatura può danneggiare la porta; si rimuove tirando la linguetta. [307] Per la banda piena basta un solo cavo; con due cavi tutte e quattro le interfacce devono ricevere un indirizzo IP. [316] La scheda «Cable the Devices», consultata il 25/10/2026 e invariata al 5/11/2026, prescrive per due nodi un solo cavo con porta 0 su entrambi oppure porta 1 su en-

trambi; due cavi non aumentano le prestazioni. Documenta inoltre un anello di tre nodi, un anello di quattro nodi esclusivamente con il percorso manuale e collegamenti tramite switch. [325] Queste prescrizioni sono documentali, non prove di cablaggio eseguite.

Tabella C.1 — Interfacce ConnectX-7 per porta QSFP, identiche su ogni DGX Spark.

Porta QSFP	Numero	Interfacce Ethernet	Dispositivi RoCE
Sinistra	0	`enp1s0f0np0`, `enP2p1s0f0np0`	`rocep1s0f0`, `roceP2p1s0f0`
Destra	1	`enp1s0f1np1`, `enP2p1s0f1np1`	`rocep1s0f1`, `roceP2p1s0f1`

Fonti: [307]. Dati consultati il 25/10/2026 e riverificati con lo stato al 5/11/2026 (pagina aggiornata il 4/11/2026, dati invariati). Note: ogni porta espone due indirizzi PCIe, quindi due interfacce Ethernet e due dispositivi RoCE; con entrambi i cavi collegati sono attive quattro interfacce Ethernet e quattro dispositivi RoCE.

C.3 Configurazione della rete ConnectX-7

La rete di gestione, sulla porta RJ-45 da 10 GbE o sul Wi-Fi, resta distinta dal collegamento ConnectX-7, che diventa una rete punto-punto con indirizzi statici. [316], [325], [326]

Percorso A — NVIDIA Sync Cluster Assistant. È il percorso raccomandato: circa 10 minuti, rischio basso; il ripristino consiste nell'eliminare il cluster in NVIDIA Sync. [325] La scheda «Configure with NVIDIA Sync», consultata il 25/10/2026 e invariata al 5/11/2026, prevede NVIDIA Sync sul portatile, l'aggiunta di ciascun sistema con nome o indirizzo, utente e password, e il Cluster Assistant, che controlla accesso SSH, hardware e software di sistema, privilegi sudo, nomi e identificativi utente, collegamenti e piano di rete esistente, poi configura SSH a chiave con alias sulla rete ConnectX-7, non sulla LAN di gestione; i dati del cluster mostrati alla fine vanno salvati in un file. [325] Secondo la guida di NVIDIA Sync il piano di rete è scritto in /etc/netplan/99-nvidia-sync-cluster.yaml [1113]. Nella versione del 2 settembre 2026, consultata il 25/10/2026 e conservata nella documentazione della release 0.117.3 di NVIDIA Sync, il test di velocità dei collegamenti usava una soglia inferiore di 184 Gbit/s, cioè 23 GB/s [1144]; la versione aggiornata il 30 ottobre 2026, corrente al 5 novembre, indica soltanto il valore atteso di 200 Gbit/s per collegamento e precisa che un avviso del test di velocità non significa necessariamente che il cluster non possa funzionare. [1112]

Il Model Launcher di NVIDIA Sync, in beta (sezione 1.5.15.3), si configura dopo il Cluster Assistant anche su un cluster di due, tre o quattro sistemi: la guida di NVIDIA Sync non indica come il modello sia ripartito fra i nodi e precisa che in modalità Localhost l'endpoint non è aperto sulle interfacce di rete del dispositivo e si raggiunge tramite il tunnel SSH di NVIDIA Sync, mentre in modalità LAN ascolta su tutte le reti del dispositivo, VPN comprese, in HTTP non cifrato anche quando è impostata una chiave API; il Model Launcher installa inoltre OpenCode sul nodo primario, con opzioni di aggiornamento da verificare sulla versione installata (sezione 5.4.9.9). [1130] Non fa parte delle procedure di questa appendice.

Percorso B — configurazione manuale. Richiede circa un'ora, con rischio medio. [325] Lo script documentato, `setup-cluster.sh`, appartiene al progetto comunitario spark-vllm-docker e non a un pacchetto NVIDIA; la panoramica del playbook nomina invece un file accessorio `spark_cluster_setup`, segno di una documentazione in transizione, e la scheda «Troubleshooting» rimanda, per il percorso manuale, a un file di configurazione config/my-cluster.json e all'opzione --pre-validate-only, propri dello strumento NVIDIA spark_cluster_setup e assenti dallo script setup-cluster.sh. [325], [326], [1145], [1146] Il codice dello script non è stato esaminato: va rivisto prima dell'esecuzione, come ogni dipendenza di terzi (sezioni 1.5.9 e 5.4.3). Il README del progetto lo presenta come aiuto sperimentale [1114], mentre il playbook lo colloca fra gli script forniti da NVIDIA [325]; il clone del Blocco C.3-1 segue il ramo predefinito, quindi conviene registrare il commit usato, per esempio con git rev-parse HEAD, come per gli script NCCL del Blocco C.5-1 (adattamento editoriale).

Blocco C.3-1 — procedura documentata (fonte [326]). Sul nodo head, come utente normale e non con `sudo`: clonazione e prova a secco con gli indirizzi di gestione, quello del nodo head per primo;

l'ordine dei cavi conta soltanto negli anelli. Gli indirizzi 192.168.1.11 e 192.168.1.12 sono gli esempi della pagina.

```
git clone https://github.com/eugr/spark-vllm-docker.git
cd spark-vllm-docker
./setup-cluster.sh 192.168.1.11 192.168.1.12 --topology direct --mtu 9000 --dry-run
```

Esito atteso: «Dry run complete; no persistent configuration was changed.» L'opzione --mtu 9000, che fissa a 9000 byte l'unità massima di trasmissione (MTU, maximum transmission unit), vale soltanto per due nodi in collegamento diretto; `--topology` va sempre indicata. [326]

Blocco C.3-2 — procedura documentata (fonte [326]). Applicazione: lo stesso comando senza `--dry-run`, confermando con `y`. Prima di accettare un'impronta di chiave host SSH sconosciuta occorre verificarla; `netplan apply` può interrompere brevemente la rete, quindi il cluster deve essere inattivo.

```
./setup-cluster.sh 192.168.1.11 192.168.1.12 --topology direct --mtu 9000
```

Esito atteso: «Cluster networking and mutual passwordless SSH are configured and verified.» [326]

Blocco C.3-3 — procedura documentata (fonte [326]). Dopo avere aperto nuove sessioni di login su ciascun nodo, perché l'appartenenza al gruppo `docker` abbia effetto, diagnosi senza modifiche:

```
./setup-cluster.sh --doctor --dry-run
```

Esito atteso: «Doctor: all saved configuration, CX7 links, SSH, and Docker checks passed.» Le riparazioni si applicano con `./setup-cluster.sh --doctor`, che elenca le riparazioni e chiede conferma prima di applicarle. [326]

Modifiche introdotte dallo script, secondo la pagina: file Netplan `/etc/netplan/98-spark-vllm-docker.yaml` con indirizzi IPv4 statici e MTU scelto, dopo il backup e lo spostamento delle voci ConnectX-7 preesistenti in `/etc/netplan`; sottoreti /24 assegnate da 10.20.0.0/16 con host `.11`, `.12` nell'ordine indicato; una chiave SSH Ed25519 per dispositivo in `~/.ssh/spark-vllm-cluster/`, un blocco in testa a `~/.ssh/config` e le chiavi pubbliche in `~/.ssh/authorized_keys`; l'appartenenza dell'utente al gruppo `docker`; backup in `/var/lib/spark-vllm-docker/setup-cluster/` su ciascun nodo; manifest di ripristino `~/.local/state/spark-vllm-docker/cluster-setup.json` e configurazione di avvio `.env` nella cartella del repository, sul nodo head. Se la configurazione fallisce o viene interrotta, lo script tenta il rollback. [326]

Blocco C.3-4 — adattamento editoriale. Ispezione dei file generati, ai percorsi documentati; il contenuto del file Netplan non è riportato dalla pagina.

```
sudo cat /etc/netplan/98-spark-vllm-docker.yaml
cat ~/.local/state/spark-vllm-docker/cluster-setup.json
```

Per una configurazione senza script il playbook rimanda alla documentazione di Netplan, non consultata; un file Netplan scritto a mano non è quindi riportato in questa appendice. [325] La guida utente rimanda per la configurazione manuale anche a un playbook «Connect Two Sparks», il cui indirizzo nel catalogo mostra però, nella lettura dell'8/11/2026 (stato al 5/11/2026 non provato da elementi datati), il playbook per più sistemi [307]; il repository dei playbook NVIDIA conserva la versione autonoma, «Connect Two DGX Sparks for Distributed Workloads», aggiornata al 12 agosto 2026, che prevede un file /etc/netplan/40-cx7.yaml, con permessi 600 e indirizzi IPv4 statici /24 sulle due interfacce della porta usata (nell'esempio enp1s0f1np1 ed enP2p1s0f1np1) e senza MTU, e le chiavi SSH con lo script discover-sparks o con ssh-copy-id; la cartella assets del playbook per più sistemi contiene inoltre un esempio non richiamato dalle schede, cx7-netplan-example.yaml, per le quattro interfacce di un solo nodo. Non fanno parte della procedura delle schede correnti e non sono riportati né verificati in questa appendice. [1147], [1148]

C.4 Verifica del collegamento, di RoCE e di SSH

Blocco C.4-1 — procedura documentata (fonti [307], [316]). Associazione fra dispositivi RoCE e interfacce e indirizzo dell'interfaccia di gestione usata dal test NCCL.

```
ibdev2netdev
ip addr show enP7s7
```

Esito atteso: con entrambi i cavi collegati `ibdev2netdev` elenca quattro righe, ciascuna terminante con «(Up)». [307] Con un solo cavo nella porta sinistra il playbook NCCL riporta quattro righe: `rocep1s0f0` e `roceP2p1s0f0` «(Up)», `rocep1s0f1` e `roceP2p1s0f1` «(Down)». [316]

Blocco C.4-2 — adattamento editoriale. Indirizzo della ConnectX-7 e raggiungibilità del nodo remoto sul collegamento diretto, con il nome d'interfaccia della Tabella C.1 per un cavo nella porta destra e l'indirizzo assegnato dallo script.

```
ip addr show enp1s0f1np1
ip addr show enP2p1s0f1np1
ping -c 3 <IP-ConnectX-nodo-2>
# MTU 9000 senza frammentazione: 9000 − 20 (IPv4) − 8 (ICMP) = 8972 byte di dati
ping -c 3 -M do -s 8972 <IP-ConnectX-nodo-2>
# ripetere i due ping con l'indirizzo del nodo 2 nella sottorete di enP2p1s0f1np1
```

Blocco C.4-3 — procedura documentata (fonte [1018]). SSH senza password e prova minima di `mpirun`, i controlli suggeriti quando `mpirun` si blocca.

```
ssh <remote_ip>
mpirun -np 2 -H <IP for Node 1>:1,<IP for Node 2>:1 hostname
```

Esito atteso: `ssh` non chiede la password [1018]; `mpirun` stampa il nome di entrambi i nodi (esito non esplicitato dalla pagina; inferenza dal comando).

C.5 Verifica della banda con NCCL

Il playbook NCCL, aggiornato il 14 ottobre 2026, compila NCCL 2.31.2-1 per Blackwell `sm_121` e nccl-tests 2.19.7, e dichiara un rischio medio. [312] Al 5 novembre 2026 le release più recenti sono NCCL 2.32.3-1, del 17 settembre 2026, e nccl-tests 2.21.1, del 5 ottobre 2026: il playbook non le adotta e motiva soltanto il blocco di nccl-tests alla 2.19.7 (si veda più avanti); l'uso di NCCL 2.32.3 su GB10 non è documentato nelle pagine consultate. [1149], [1150]

Blocco C.5-1 — procedura documentata (fonte [316]). Percorso rapido con gli script del repository NVIDIA, eseguito sul nodo 1 con gli indirizzi di gestione. Gli URL sono quelli riportati per intero dalla pagina; il contenuto degli script non è stato esaminato. Gli script provengono dal ramo `main`: per la riproducibilità conviene scaricarli da un commit fissato, registrarne l'hash e rileggerli prima dell'esecuzione (adattamento editoriale). Al 5 novembre 2026 l'ultimo commit che modifica il playbook NCCL è 18fd621b888e0722dfb28ef35bc4451eda042763, del 19 ottobre 2026, e setup.sh è stato modificato da ultimo il 15 ottobre: negli URL si può sostituire refs/heads/main con l'hash del commit scelto (adattamento editoriale). [1151]

```
curl -fsSL "https://raw.githubusercontent.com/NVIDIA/dgx-spark-
playbooks/refs/heads/main/nvidia/playbook-nccl/assets/setup.sh" -o setup.sh
curl -fsSL "https://raw.githubusercontent.com/NVIDIA/dgx-spark-
playbooks/refs/heads/main/nvidia/playbook-nccl/assets/launch.sh" -o launch.sh
bash setup.sh <NODE_2_IP>
bash launch.sh --topology direct <NODE_1_IP> <NODE_2_IP>
```

Blocco C.5-2 — procedura documentata (fonte [316]). Percorso manuale, su entrambi i nodi: compilazione di NCCL e di nccl-tests.

```
sudo apt-get update && sudo apt-get install -y libopenmpi-dev
git clone -b v2.31.2-1 https://github.com/NVIDIA/nccl.git ~/nccl/
cd ~/nccl/
make -j src.build NVCC_GENCODE="-gencode=arch=compute_121,code=sm_121"
export CUDA_HOME="/usr/local/cuda"
export MPI_HOME="/usr/lib/aarch64-linux-gnu/openmpi"
export NCCL_HOME="$HOME/nccl/build/"
export LD_LIBRARY_PATH="$NCCL_HOME/lib:$CUDA_HOME/lib64/:$MPI_HOME/lib:$LD_LIBRARY_PATH"
git clone -b v2.19.7 https://github.com/NVIDIA/nccl-tests.git ~/nccl-tests/
cd ~/nccl-tests/
make MPI=1
```

nccl-tests è fissato alla 2.19.7 perché dalla 2.20.0 i test GIN non si collegano con l'OpenMPI di sistema. [316]

Blocco C.5-3 — procedura documentata (fonte [316]). Esecuzione del test dal nodo 1, con gli indirizzi di gestione seguiti dal suffisso :1, come richiama la pagina; il comando va lanciato nella stessa shell in cui sono state esportate le variabili del Blocco C.5-2, perché -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH trasmette ai due rank il valore corrente (adattamento editoriale).

```
mpirun -np 2 -H <management IP for Node 1>:1,<management IP for Node 2>:1 \
  --mca plm_rsh_agent "ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no" \
  -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH \
  -x UCX_NET_DEVICES=enP7s7 \
  -x NCCL_SOCKET_IFNAME=enP7s7 \
  -x OMPI_MCA_btl_tcp_if_include=enP7s7 \
  $HOME/nccl-tests/build/all_gather_perf
```

La variante con buffer grandi sostituisce l'ultima riga con `$HOME/nccl-tests/build/all_gather_perf -b 16G -e 16G`. Va usata la stessa interfaccia su tutti i nodi: `enP7s7` ovunque, oppure ovunque l'interfaccia Wi-Fi, per esempio wlP9s9, da verificare con ip -o link show: nei Blocchi C.5-3 e C.5-4, e nel comando ip addr show del Blocco C.4-1, enP7s7 si sostituisce con l'interfaccia Wi-Fi e come indirizzi di gestione si usano gli indirizzi Wi-Fi dei nodi; nel percorso rapido del Blocco C.5-1 si antepone invece `MGMT_IFNAME=wlP9s9` al comando `launch.sh`. Il comando disattiva la verifica delle chiavi host SSH. [316]

Esito atteso: la pagina non indica un valore numerico, né per la banda di bus né per quella dell'algoritmo. Poiché `NCCL_SOCKET_IFNAME` e `UCX_NET_DEVICES` puntano all'interfaccia di gestione, va controllato il trasporto usato per i dati (adattamento editoriale): aggiungendo `-x NCCL_DEBUG=INFO` al comando, il log deve indicare il trasporto di rete `NET/IB` sui dispositivi RoCE della porta collegata e non `NET/Socket`; un ripiego su socket misurerebbe la rete di gestione a 10 GbE, 1,25 GB/s lordi, invece del collegamento ConnectX-7. [316] Come stima, il tetto nominale di un collegamento è 200 Gb/s / 8 = 25 GB/s, prima degli overhead.

Blocco C.5-4 — adattamento editoriale (latenza e host noti). Per stimare la latenza delle collettive, che governa il tensor parallelism a batch 1 (sezione 1.5.15.5), si ripete il test con messaggi piccoli; dopo il Blocco C.3-2 le chiavi host sono già distribuite e verificate, quindi le opzioni che disattivano la verifica possono essere omesse.

```
mpirun -np 2 -H <management IP for Node 1>:1,<management IP for Node 2>:1 \
  -x LD_LIBRARY_PATH=$LD_LIBRARY_PATH \
  -x UCX_NET_DEVICES=enP7s7 \
  -x NCCL_SOCKET_IFNAME=enP7s7 \
  -x OMPI_MCA_btl_tcp_if_include=enP7s7 \
  $HOME/nccl-tests/build/all_reduce_perf -b 8 -e 1M -f 2
```

Le pagine consultate non documentano test con iperf o ib_write_bw, né valori attesi di latenza.

C.6 vLLM su due nodi in tensor parallelism

Il playbook, aggiornato il 22 ottobre 2026, distingue percorsi con NVIDIA Sync e percorsi manuali per due nodi e fornisce ricette diverse per memoria: per 2 × 64 GB Qwen3.8 Flash-Next (`nvidia/Qwen3.8-Flash-Next-FP4-Local`), per 2 × 128 GB GLM-5.3-Flash (`nvidia/GLM-5.3-Flash-FP4-Local`). Immagine, checkpoint, ambiente e opzioni vanno presi dalla stessa ricetta, senza mescolarli fra modelli o tagli di memoria. [297], [302] La procedura manuale non usa Ray: ogni nodo esegue un rank del tensor parallelism, con rendezvous sull'indirizzo ConnectX del nodo head, e il playbook rimanda alla documentazione vLLM sull'esecuzione multi-nodo con multiprocessing. [297]

Blocco C.6-1 — procedura documentata (fonte [297]). Variabili d'ambiente, su ciascun nodo.

```
export HEAD_IP="<head-connectx-ip>"
export ETH_IF="<local-connectx-interface>"
export IB_IF="<local-roce-hca-1>,<local-roce-hca-2>"
export MASTER_PORT=29501
export IMAGE="eugr/spark-vllm-b12x:nightly-20261015"
export HF_CACHE_DIR="$HOME/.cache/huggingface"
```

Blocco C.6-2 — adattamento editoriale. Valori di esempio per un cavo nella porta destra, ricavati dalla Tabella C.1 e da verificare con `ibdev2netdev` (con un cavo nella porta sinistra, quella dell'esempio del playbook NCCL, `enp1s0f0np0` e `rocep1s0f0,roceP2p1s0f0`); HEAD_IP è l'indirizzo assegnato, dallo script o da NVIDIA Sync, all'interfaccia ETH_IF del nodo head: ogni porta espone due interfacce, ciascuna su una propria sottorete (/24 con lo script) [326], e la scheda del checkpoint colloca l'indirizzo del nodo head nella sottorete dell'interfaccia Ethernet scelta, con entrambi i dispositivi RoCE gemelli in IB_IF [1028].

```
export ETH_IF="enp1s0f1np1"
export IB_IF="rocep1s0f1,roceP2p1s0f1"
```

Blocco C.6-3 — procedura documentata (fonte [297]). Creazione delle cartelle di cache montate e avvio del container vllm_cluster, su ciascun nodo; prima di docker run la pagina esegue mkdir -p "$HF_CACHE_DIR" "$HOME/.cache/vllm" "$HOME/.cache/flashinfer" "$HOME/.triton" "$HOME/.cache/b12x".

```
docker run --rm -it \
  --name vllm_cluster \
  --privileged --gpus all --ipc host --network host \
  --shm-size=16g \
  --ulimit nofile=1048576:1048576 \
  --ulimit memlock=-1 --ulimit stack=67108864 \
  -e HF_TOKEN="${HF_TOKEN:-}" \
  -e MN_IF_NAME="$ETH_IF" -e UCX_NET_DEVICES="$ETH_IF" \
  -e NCCL_SOCKET_IFNAME="$ETH_IF" -e NCCL_IB_HCA="$IB_IF" \
  -e NCCL_IB_DISABLE=0 -e NCCL_IGNORE_CPU_AFFINITY=1 \
  -e OMPI_MCA_btl_tcp_if_include="$ETH_IF" \
  -e GLOO_SOCKET_IFNAME="$ETH_IF" -e TP_SOCKET_IFNAME="$ETH_IF" \
  -e HEAD_IP="$HEAD_IP" -e MASTER_PORT="$MASTER_PORT" \
  -v "$HF_CACHE_DIR:/root/.cache/huggingface" \
  -v "$HOME/.cache/vllm:/root/.cache/vllm" \
  -v "$HOME/.cache/flashinfer:/root/.cache/flashinfer" \
  -v "$HOME/.triton:/root/.triton" \
  -v "$HOME/.cache/b12x:/root/.cache/b12x" \
  --entrypoint /bin/bash "$IMAGE"
```

Per le variabili NCCL_* la documentazione di NCCL 2.32.3, consultata per questa edizione, indica che NCCL_SOCKET_IFNAME seleziona le interfacce IP usate da NCCL, NCCL_IB_HCA gli adattatori RDMA, NCCL_IB_DISABLE=0 lascia attivo il trasporto IB/RoCE e NCCL_IGNORE_CPU_AFFINITY=1 fa ignorare a NCCL l'affinità di CPU ereditata dal lanciatore [1152]; la versione di NCCL contenuta nell'immagine non è documentata, e il ruolo delle variabili UCX_* e GLOO_* resta quello della documentazione dei rispettivi progetti, non consultata; la ricetta assegna i socket all'interfaccia ConnectX-7 e il trasporto RDMA ai due dispositivi RoCE locali.

Blocco C.6-4 — procedura documentata (fonte [297]). Argomenti del nodo, da impostare nella shell del container aperta dal Blocco C.6-3: sul nodo head la prima riga, sul worker la seconda. NODE_ARGS è un array di shell e non passa al container dall'ambiente dell'host.

```
NODE_ARGS=(--node-rank 0)
NODE_ARGS=(--node-rank 1 --headless)
```

Blocco C.6-5 — procedura documentata (fonte [297]). Avvio del modello nella ricetta per 2 × 128 GB, con comando, variabili e opzioni identici nei due container; si avvia prima il nodo head, poi il worker, e l'avvio attende entrambi i rank.

```
export CUTE_DSL_ARCH="sm_121a"
export SAFETENSORS_FAST_GPU="1"
export VLLM_ENABLE_ROCE_ALLREDUCE="1"
export VLLM_ROCE_ALLREDUCE_MAX_SIZE="2MB"
export NCCL_BUFFSIZE=1048576
export NCCL_MAX_CTAS=8
export VLLM_WORKER_MULTIPROC_METHOD="spawn"
export VLLM_SSM_CONV_STATE_LAYOUT="DS"
export VLLM_USE_AOT_COMPILE="1"
export VLLM_USE_MEGA_AOT_ARTIFACT="1"
export VLLM_USE_V2_MODEL_RUNNER="1"
export VLLM_ENABLE_PCIE_ALLREDUCE="0"
```

```
export B12X_POLICY_MODE="auto"
export VLLM_B12X_MOE_FP4_FORCE_A16=1
export B12X_W4A16_A4_PREFILL=1

vllm serve nvidia/GLM-5.3-Flash-FP4-Local \
  --host 0.0.0.0 \
  --port 8000 \
  --tensor-parallel-size 2 \
  --pipeline-parallel-size 1 \
  --decode-context-parallel-size 1 \
  --mamba-cache-mode align \
  --enable-prefix-caching \
  --enable-chunked-prefill \
  --dtype bfloat16 \
  --kv-cache-dtype fp8 \
  --quantization modelopt_mixed \
  --attention-backend B12X \
  --block-size 256 \
  --moe-backend b12x \
  --linear-backend b12x \
  --no-enable-flashinfer-autotune \
  --load-format b12x \
  --max-model-len 500000 \
  --max-num-seqs 4 \
  --max-num-batched-tokens 4096 \
  --speculative-config
'{"method":"mtp","num_speculative_tokens":5,"moe_backend":"humming","attention_backend":"B12X"
}' \
  --reasoning-parser glm45 \
  --tool-call-parser glm47 \
  --enable-auto-tool-choice \
  --gpu-memory-utilization 0.87 \
  --nnodes 2 \
  --master-addr "$HEAD_IP" \
  --master-port "$MASTER_PORT" \
  "${NODE_ARGS[@]}"
```

Note documentate: il rendezvous usa l'indirizzo ConnectX del nodo head e la porta 29501, l'API la porta 8000; la ricetta per 2 × 64 GB usa invece `--gpu-memory-utilization 0.8` e `--kv-cache-memory-bytes 3G`; i limiti di contesto, 500.000 token per GLM-5.3-Flash e 262.144 per Qwen3.8 Flash-Next, sono tetti per richiesta e non capacità concorrente garantita. Se il worker fallisce, il nodo head può restare bloccato in avvio: vanno controllate entrambe le sessioni. [297]

Stima della memoria per nodo. GLM-5.3-Flash ha 320B parametri totali e 18B attivi. [112] Con 4,5 bit per parametro i pesi richiederebbero idealmente $320 \cdot 10^9 \times 4{,}5/8 = 180$ GB; con le parti a precisione maggiore della quantizzazione `modelopt_mixed`, ipotizzate in 5–20 GB, la stima per componenti dell'Allegato 2 (calcoli ME08 e ME36) è di 93–100 GB per nodo: entro $u = 0{,}87$, circa 111,4 GB su 128 GB nominali, resterebbero 11–18 GB per nodo per KV cache FP8, attivazioni, componenti MTP e comunicazione, e con workspace e margine il requisito, 108,9–116,6 GB per nodo, sarebbe al limite del budget o lo supererebbe; la dimensione dei file del checkpoint, riportata di seguito, è però inferiore alla stima dei pesi, che per questo checkpoint risulta cautelativa. L'API Hugging Face, consultata il 25/10/2026 alla revisione 76348ddb2be2e35e39ecfc00e75dc07703f4f3d5 e riletta con lo stato al 5/11/2026 (revisione ancora corrente, ultima modifica del repository il 16/10/2026), elenca 42 file .safetensors per 177.463.841.024 byte (177,464 GB decimali; 165,276 GiB). È la dimensione dichiarata dei file, non la memoria residente del runtime: le stime di memoria e i margini vanno verificati nei log di avvio di ciascun rank [1028]. Ripartiti in parti uguali fra i due rank, i file corrispondono a circa 88,7 GB per nodo, meno dell'estremo inferiore della stima: con le stesse ipotesi di workspace e margine il requisito scende a circa 104,2 GB per nodo, cache esclusa, entro il budget di 111,4 GB ma ancora nella fascia «al limite» della regola della sezione 1.5.14.3 (calcolo editoriale; la ripartizione effettiva dei tensori fra i rank non è documentata). La scheda del checkpoint dichiara un uso di circa 166 GiB e una convalida su due DGX Spark da 128 GB con 2 milioni di token di KV cache FP8 (dichiarato dal produttore) [1028]; la sezione 1.5.14.4 (Esempio 5) assume questa memoria dichiarata, circa 89,1 GB

per nodo, e ne ricava un requisito di circa 104,6 GB per nodo, cache esclusa, nella stessa fascia «al limite».

Blocco C.6-6 — procedura documentata (fonte [297]). Verifiche dell'API sul nodo head e accesso da un portatile.

```
# sul nodo head
curl --fail-with-body -i http://127.0.0.1:8000/health
curl --fail-with-body -sS http://127.0.0.1:8000/v1/models
# sul portatile
ssh -N -L 8000:127.0.0.1:8000 <user>@<head-host>
```

Esito atteso: HTTP 200 per `/health`; l'`id` del modello da `/v1/models`, da usare come nome del modello con base URL `http://localhost:8000/v1`; le verifiche di chat e tool calling seguono lo schema dei Blocchi B.7-2 e B.7-3, con `max_tokens` a 4.096 e la richiesta `get_weather` per Boston, che deve produrre `tool_calls` con `finish_reason` uguale a `tool_calls`. Prima di regolare le opzioni si misurano latenza, throughput e memoria su entrambi i nodi, cambiando un'impostazione alla volta su entrambi i rank. [297]

Blocco C.6-7 — adattamento editoriale (bind su loopback). Il playbook avverte che l'API si lega alle interfacce dell'host e va usata solo su una rete fidata; l'immagine è comunitaria, non ufficiale NVIDIA. [297] Se i client passano dal tunnel SSH del Blocco C.6-6, sul nodo head si può sostituire `--host 0.0.0.0` con `--host 127.0.0.1`, lasciando invariato il resto del comando; la variante non è stata provata in configurazione a due nodi. La guida di sicurezza di vLLM richiamata nella sezione 5.4.9.7 dichiara non sicure per default le comunicazioni fra i nodi e chiede una rete dedicata e isolata, `VLLM_HOST_IP` fissato e porte limitate agli host fidati [228]; la ricetta non imposta VLLM_HOST_IP e, con --network host, i processi del runtime ascoltano anche sulle interfacce di gestione e Wi-Fi, perché il TCPStore di PyTorch ascolta per default su tutte le interfacce [228]. Adattamento editoriale, non provato: aggiungere al Blocco C.6-3 `-e VLLM_HOST_IP=<indirizzo ConnectX del nodo>`, ammettere con il firewall dell'host la porta 29501 e le altre porte del runtime soltanto dall'indirizzo ConnectX dell'altro nodo, disattivare il Wi-Fi, anche nell'UEFI, dove dalla release di gennaio 2026 lo si disattiva soltanto insieme al Bluetooth [303], [1137], e usare `HF_TOKEN` solo per il primo download, poi rimuoverlo dall'ambiente e dalla configurazione della CLI: il Blocco C.6-3 monta l'intera cartella $HOME/.cache/huggingface, che per default contiene anche il token salvato dalla CLI [973]. Dopo un primo avvio riuscito su entrambi i nodi, con le cache già popolate, la riga -e HF_TOKEN=… del Blocco C.6-3 si sostituisce con -e HF_HUB_OFFLINE=1 -e VLLM_NO_USAGE_STATS=1 -e DO_NOT_TRACK=1, come nel Blocco B.7-5 e con le opzioni della sezione 5.4.9.4. [973], [974] Per l'API vale inoltre la cautela sulle richieste cross-origin della sezione B.7: la ricetta non imposta --allowed-origins e la build di vLLM dell'immagine notturna, 0.1.dev21830+g8244d1c44 secondo la raccolta di ricette NVIDIA [1133], parte da un commit che non contiene la correzione della release 0.32.0 e nel quale il valore predefinito è ["*"] [1140], [1153]: sul nodo head conviene aggiungere al comando del Blocco C.6-5 --allowed-origins '[]', o l'elenco JSON delle sole origini dei client ammessi, dopo averne letto il valore predefinito con vllm serve --help=allowed-origins (adattamento editoriale, non provato).

Blocco C.6-8 — procedura documentata (fonte [297]). Arresto, su ciascun nodo; il tunnel si chiude con Ctrl+C.

```
docker stop vllm_cluster
```

I container vengono rimossi all'arresto, mentre le cache dei modelli e del runtime restano sul disco di ciascun nodo. [297]

C.7 Repliche indipendenti e separazione dei servizi

Le due alternative al modello ripartito non hanno un playbook dedicato e si costruiscono con le procedure dell'Appendice B: sono **adattamenti editoriali**, non provati.

- **Repliche indipendenti.** Su ciascun nodo si avvia lo stesso servizio con uno dei percorsi B.6–B.8; un gateway con autenticazione, componente dell'architettura della sezione 5.4.9, distribuisce le richieste. Il collegamento ConnectX-7 e NCCL non servono nel ciclo dei token. Ogni replica dispone della memoria di un nodo: la replica non somma la memo-

ria, e il throughput aggregato può al massimo avvicinarsi al doppio con richieste indipendenti, senza ridurre la latenza della singola richiesta (stima).

- **Separazione dei servizi.** Il nodo 1 esegue il generatore, con `--gpu-memory-utilization` fino a 0,87 se non ospita altri servizi; il nodo 2 esegue ingestione, embedding, reranker e database vettoriale. I servizi comunicano con richieste HTTP o RPC sulla rete ConnectX-7 o sulla rete di gestione. Gli endpoint dipendono dal server di embedding scelto, la cui procedura su GB10 non è documentata nelle pagine consultate. La cifratura del traffico fra i nodi non è trattata dalle fonti: per dati sensibili si applicano TLS e autenticazione dei servizi come nella sezione 5.4.9.

C.8 Verifica finale

Tabella C.2 — Verifiche a doppio nodo: comando ed esito atteso.

Verifica	Comando o azione	Esito atteso	Etichetta e fonte
Prova a secco della rete	Blocco C.3-1	«Dry run complete; no persistent configuration was changed.»	Documentato [326]
Rete e SSH configurati	Blocco C.3-2	«Cluster networking and mutual passwordless SSH are configured and verified.»	Documentato [326]
Diagnosi	Blocco C.3-3	«Doctor: all saved configuration, CX7 links, SSH, and Docker checks passed.»	Documentato [326]
Collegamenti RoCE	`ibdev2netdev`	Con un cavo, «(Up)» le due righe della porta collegata e «(Down)» le altre due; con due cavi, quattro righe «(Up)»	Documentato [307], [316] (con un cavo, esempio della porta sinistra; porta destra per simmetria)
SSH e MPI	Blocco C.4-3	Accesso senza password; nomi dei due nodi	Documentato [1018]; esito di `mpirun` inferito
Banda NCCL	Blocco C.5-3	Valore non documentato; tetto nominale 25 GB/s	Documentato [316]; tetto = stima
API distribuita	Blocco C.6-6	HTTP 200; `id` del modello	Documentato [297]
Tool calling	Schema del Blocco B.7-3 sul nodo head	`tool_calls` e `finish_reason` = `tool_calls`	Documentato [295], [297]
Memoria per nodo	Log di avvio dei due rank; `/proc/meminfo`	Capacità KV coerente con la stima di C.6	Adattamento di [295], [306]

Fonti: [295], [297], [306], [307], [316], [326], [1018]. Note: nessuna verifica è stata eseguita per questa edizione.

C.9 Ripristino

- **Servizio distribuito:** `docker stop vllm_cluster` su ciascun nodo (Blocco C.6-8). [297] La rimozione dell'immagine non è descritta; per analogia con l'Appendice B si usa `docker rmi eugr/spark-vllm-b12x:nightly-20261015` (adattamento editoriale). Le cache dei modelli vanno rimosse su entrambi i nodi indicando la cartella per nome esatto, per esempio ~/.cache/huggingface/hub/models--nvidia--GLM-5.3-Flash-FP4-Local, senza cancellare l'intera cartella di Hugging Face, che contiene anche gli altri modelli e, per default, il token della CLI [973]; poiché il container vi scrive come root, la rimozione può richiedere sudo, con le cautele della sezione B.9 (adattamento editoriale).

Blocco C.9-1 — procedura documentata (fonti [312], [316]). Test NCCL: rimozione dei repository da ciascun nodo.

```
rm -rf ~/nccl/
rm -rf ~/nccl-tests/
```

Blocco C.9-2 — procedura documentata (fonte [326]). Rete e SSH, percorso manuale: dal nodo head, con lo stesso utente e con tutti i dispositivi raggiungibili.

```
./setup-cluster.sh --restore
```

Il ripristino riporta i file Netplan, SSH e `.env` originali con i loro permessi, rimuove chiavi e file creati dallo script, riapplica la configurazione di rete precedente e toglie l'appartenenza al gruppo `docker` solo se l'aveva aggiunta lo script. Se un file gestito è stato modificato dopo la configurazione, il ripristino si arresta e lo indica: la modifica va riconciliata prima di ripetere il comando. I backup vanno conservati

fino al termine del ripristino; per cambiare topologia, nodo head o indirizzi si ripristina prima e si riconfigura poi. [326]

- **Rete, percorso NVIDIA Sync:** dopo l'arresto del servizio su entrambi i nodi, eliminazione del cluster in NVIDIA Sync. [302], [325] Secondo la guida di NVIDIA Sync, nella versione del 2 settembre 2026, l'eliminazione rimuove la configurazione SSH fra i nodi e la relazione di cluster [1144]; la versione del 30 ottobre 2026 non ne elenca più gli effetti; la rimozione del piano di rete ConnectX-7 è descritta come procedura distinta per ciascun nodo: spostare /etc/netplan/99-nvidia-sync-cluster.yaml fuori da /etc/netplan, per esempio in /root/netplan-disabled/, eseguire sudo netplan generate e sudo netplan try, poi controllare con ip -br addr che gli indirizzi ConnectX-7 non compaiano più; la rimozione del file non elimina necessariamente chiavi, alias e altre impostazioni SSH fra i nodi, che si rimuovono con l'eliminazione del cluster. [1112], [1113] Poiché il playbook vLLM indica l'eliminazione del cluster anche per annullare la configurazione di rete [302], l'esito va comunque controllato con ip -br addr su ciascun nodo.
- **Cavi:** rimozione tirando la linguetta. [307]
- **Switch, se usato:** vanno ripristinate le impostazioni modificate su porte, bridge, VLAN, DHCP, velocità e MTU. [326]

C.10 Limiti della procedura

- Nessun comando è stato eseguito e nessuna misura propria di banda, latenza o throughput è disponibile per questa edizione. Per la ricetta GLM-5.3-Flash a due nodi la raccolta di ricette di NVIDIA per l'hardware locale (release v0.66 del 24 ottobre 2026) dichiara 33,94 token/s di decodifica, media ponderata sul tempo di decodifica attiva e non per richiesta, misurata il 16/10/2026 con aiperf 0.12.0 su SPEED-Bench (profilo qualitative, categoria coding; 10 richieste, 8.192 token in uscita) a concorrenza 1, con la decodifica speculativa della ricetta (MTP, 5 token speculativi), --gpu-memory-utilization 0.87, l'immagine nightly-20261015 (stesso digest riportato più avanti in questo elenco; vLLM 0.1.dev21830+g8244d1c44), driver 580.159.03, CUDA 13.0.88 e la revisione f5a0baba30120da613e27973ee3535a725d4ece3 del checkpoint, diversa da quella corrente: è una dichiarazione del produttore, non riprodotta, ottenuta con un driver diverso da quello corrente della Founders Edition (580.178.04) e pubblicata senza il tasso di accettazione dei token speculativi. [1133]
- Le schede «Cable the Devices» e «Configure with NVIDIA Sync», la guida del Cluster Assistant di NVIDIA Sync e la scheda «Troubleshooting» del playbook vLLM a due nodi sono state consultate il 25/10/2026 e riverificate con lo stato al 5/11/2026: le schede dei playbook sono invariate (ultimi commit del 20 e del 22 ottobre 2026), mentre la guida di NVIDIA Sync è stata aggiornata il 30 ottobre 2026 (sezioni C.1, C.3 e C.9). La scheda «Troubleshooting» del playbook vLLM tratta soltanto controlli del Cluster Assistant non completati, Docker con permesso negato, errori di autenticazione Hugging Face, da controllare con hf auth whoami su entrambi i nodi, e API del nodo head che non risponde, caso in cui va letto l'output del terminale di avvio. [325], [1112], [1115] Lo script setup-cluster.sh è un involucro di cinque righe che esegue setup-cluster.py, modificato da ultimo il 30 ottobre 2026, dopo l'ultimo aggiornamento del playbook nel repository NVIDIA (20 ottobre 2026), con l'introduzione di un'attesa fino a 60 secondi per nodo che precede le prove di connettività, descritta dalla documentazione del progetto aggiornata il 1° novembre 2026 [1114]: al 5 novembre 2026 il file Python contiene le opzioni e i messaggi attesi documentati da [326], controllati come testo; il codice di setup-cluster.py e degli script NCCL non è stato esaminato. [1146]
- La ricetta distribuita usa un'immagine comunitaria con build notturna: risultati e opzioni possono cambiare con l'immagine. Al 5/11/2026 Docker Hub riporta il tag nightly-20261015, pubblicato il 15/10/2026 e non più modificato, come immagine linux/arm64 con digest sha256:03a956544e9f9b4b7ba91690469fbe7848bdbef29a7d2e0f5788368467bb3595, da registrare insieme al tag [1116]; il tag latest, aggiornato da ultimo il 2/11/2026, punta invece alla build nightly-20261102, con digest sha256:19ab778dd57f019c44caf5df91a671cf688923d1ca9fa8ce92be0c9acb46d215, diversa da quella del playbook (metadati consultati l'8/11/2026). [1154] La scheda del checkpoint, invariata al 5/11/2026, avverte che i fattori di scala NVFP4 usano un formato compresso non ancora supportato da vLLM upstream e richiede i container comunitari (le note della release upstream v0.33.0, del 2/11/2026, non ne menzionano il supporto e dichiarano deprecata, e in quella versione senza effetto, l'opzione --mamba-cache-mode presente nella ricetta, che usa però un'immagine del 15/10/2026, anteriore alla release [1128]); indica inoltre l'immagine eugr/spark-vllm-b12x:latest e l'avvio dei worker prima del nodo head, mentre il playbook fissa il tag notturno e avvia prima il nodo head. [297], [1028]
- Pipeline parallelism con più di uno stadio, expert parallelism, SGLang su due nodi e l'uso di llama.cpp RPC fra due sistemi GB10 non sono documentati nelle pagine consultate, rilette con lo stato al 5 novembre 2026: nei playbook SGLang e llama.cpp la colonna dell'hardware idoneo al multi-nodo riporta un trattino per DGX Spark, e quello di llama.cpp compila il backend RPC senza descriverne l'uso [317], [318]. Per TensorRT-LLM il catalogo non contiene un playbook, ma il repository dei playbook NVIDIA include «Serve LLMs with NVIDIA TensorRT-LLM», aggiornato al 12 agosto 2026, con una scheda «Multi-node serving» per DGX Spark: OpenMPI con file degli host, contenitore nvcr.io/nvidia/tensorrt-llm/release:1.3.0rc5 su ogni nodo e trtllm-serve con TP_SIZE=2 nell'esempio, per nvidia/Qwen3-235B-A22B-FP4; la procedura non è riportata né verificata in questa appendice [1132].
- GPUDirect RDMA non è supportato su DGX Spark: le ricette che dipendono da nvidia-peermem, dma-buf o GDRCopy non si applicano. [308]

Bibliografia della guida

[4] OpenAI, “gpt-oss-120b” e “gpt-oss-20b,” *OpenAI API documentation*, s.d. [Online]. Disponibile: https://developers.openai.com/api/docs/models/gpt-oss-120b; https://developers.openai.com/api/docs/models/gpt-oss-20b. Accesso: 08/11/2026.

[5] OpenAI, “Introducing gpt-oss,” 5 ago. 2025. [Online]. Disponibile: https://openai.com/index/introducing-gpt-oss/. Accesso: 08/11/2026.

[32] LiveBench Team, “LiveBench 2.0 — Release 2026-10-14,” 14 ott. 2026, agg. 18 ott. 2026 (versione consultata; pagina nuovamente aggiornata il 23 ott. 2026; al 5 nov. 2026 la release elenca 55 modelli ed è seguita dalla release 2026-10-25). [Online]. Disponibile: https://livebench.ai/#/; https://livebench.ai/#/methodology; dati della release: https://livebench.ai/data/20261014/table.csv. Accesso: 08/11/2026.

[56] Google, “Gemma releases,” *Google AI for Developers*, s.d. [Online]. Disponibile: https://ai.google.dev/gemma/docs/releases (documentazione dichiarata deprecata; versione corrente: https://gemma.ai/models/releases/). Accesso: 08/11/2026.

[99] Qwen Team, “Qwen3.8-27B,” *Hugging Face*, s.d. [Online]. Disponibile: https://huggingface.co/Qwen/Qwen3.8-27B. Accesso: 08/11/2026.

[112] Z.ai, “GLM-5.3-Flash,” *Hugging Face*, s.d. [Online]. Disponibile: https://huggingface.co/zai-org/GLM-5.3-Flash. Accesso: 08/11/2026.

[158] Google, “Embeddings,” *Google AI Studio*, s.d. [Online]. Disponibile: https://aistudio.google.com/docs/embeddings. Accesso: 08/11/2026.

[172] NVIDIA, “Personal AI Supercomputer Powered by Blackwell — NVIDIA DGX Spark,” s.d. [Online]. Disponibile: https://www.nvidia.com/en-us/products/workstations/dgx-spark/. Accesso: 08/11/2026.

[173] J. Zhou e R. Chen, “NVIDIA DGX Spark In-Depth Review: A New Standard for Local AI Inference,” *LMSYS Org Blog*, 13 ott. 2025. [Online]. Disponibile: https://www.lmsys.org/blog/2025-10-13-nvidia-dgx-spark. Accesso: 10/11/2026.

[205] A. Zandieh, M. Daliri, M. Hadian e V. Mirrokni, “TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate,” arXiv:2504.19874, 28 apr. 2025, doi: 10.48550/arXiv.2504.19874; versione pubblicata in ICLR 2026 (poster). [Online]. Disponibile: https://arxiv.org/abs/2504.19874. Accesso: 08/11/2026.

[213] Ollama, “FAQ,” s.d. [Online]. Disponibile: https://docs.ollama.com/faq.md. Accesso: 08/11/2026.

[218] A. Fogel, O. Hofman, E. Cohen e R. Vainshtein, “Inference-Time Backdoors via Hidden Instructions in LLM Chat Templates,” arXiv:2602.04653v1, 4 feb. 2026 (titolo della v4 del 24 mag. 2026: “Inference-Time Backdoors via Chat Templates: From LLM Supply Chains to Agentic System Compromise”; secondo i commenti arXiv, la v3 del 9 mar. 2026 è accettata all’ICLR 2026 Trustworthy AI Workshop e la v4 è «Submitted to CCS 2026»). [Online]. Disponibile: https://arxiv.org/html/2602.04653v1. Accesso: 08/11/2026.

[221] E. Alvarez *et al.*, “Introducing NVFP4 for Efficient and Accurate Low-Precision Inference,” *NVIDIA Technical Blog*, 24 giu. 2025. [Online]. Disponibile: https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/. Accesso: 08/11/2026.

[222] Open Compute Project, “OCP Microscaling Formats (MX) Specification Version 1.0,” specifica di Microsoft, AMD, Arm, Intel, Meta, NVIDIA e Qualcomm, set. 2023. [Online]. Disponibile: https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf/. Accesso: 08/11/2026.

[223] OpenAI, “gpt-oss-120b,” *Hugging Face*, s.d. [Online]. Disponibile: https://huggingface.co/openai/gpt-oss-120b. Accesso: 08/11/2026.

[226] NVIDIA, “Custom Installation with cloud-init — DGX Spark User Guide,” *DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/enterprise-custom-install.html. Accesso: 10/11/2026.

[227] NVIDIA, “OS and Component Update Guide — DGX Spark User Guide,” NVIDIA DGX Spark User Guide, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/os-and-component-update.html. Accesso: 08/11/2026.

[228] vLLM Project, “Security,” s.d., agg. 30 ott. 2026. [Online]. Disponibile: https://docs.vllm.ai/en/latest/usage/security/. Accesso: 08/11/2026.

[233] NVIDIA, “System Overview — NVIDIA DGX Spark Porting Guide,” NVIDIA Docs, s.d., agg. 21 ott. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark-porting-guide/latest/overview.html. Accesso: 10/11/2026.

[234] Dell Technologies, “Dell Pro Precision with Nvidia Blackwell — Dell USA (Dell Pro Precision with GB10, Model FCM1253; configuratore),” s.d. [Online]. Disponibile: https://www.dell.com/en-us/shop/desktop-computers/spd/dellpromaxwithgb10fcm1253. Accesso: 08/11/2026.

[246] G. Delgado, “Bringing the Spark to Europe: Local Intelligence Powers Next Wave of Agents,” *NVIDIA Blog*, 21 ott. 2026. [Online]. Disponibile: https://blogs.nvidia.com/blog/local-ai-gtc-rtx-dgx-spark-station-sync. Accesso: 08/11/2026.

[287] OpenAI, “gpt-oss-120b & gpt-oss-20b Model Card,” arXiv:2508.10925v1, 8 ago. 2025. [Online]. Disponibile: https://arxiv.org/pdf/2508.10925v1; https://arxiv.org/abs/2508.10925. Accesso: 08/11/2026.

[288] Unsloth AI, “unsloth/Qwen3.6-35B-A3B-MTP-GGUF,” *Hugging Face*, s.d. [Online]. Disponibile: https://huggingface.co/unsloth/Qwen3.6-35B-A3B-MTP-GGUF. Accesso: 08/11/2026.

[289] Qwen Team, “Qwen/Qwen3.5-122B-A10B-GPTQ-Int4,” *Hugging Face*, s.d. [Online]. Disponibile: https://huggingface.co/Qwen/Qwen3.5-122B-A10B-GPTQ-Int4. Accesso: 08/11/2026.

[290] NVIDIA, “nvidia/NVIDIA-Nemotron-3.5-Super-VL-120B-A12B-NVFP4,” *Hugging Face*, s.d. [Online]. Disponibile: https://huggingface.co/nvidia/NVIDIA-Nemotron-3.5-Super-VL-120B-A12B-NVFP4. Accesso: 08/11/2026.

[291] Qwen Team, “Qwen/Qwen3.5-122B-A10B,” *Hugging Face*, s.d. [Online]. Disponibile: https://huggingface.co/Qwen/Qwen3.5-122B-A10B. Accesso: 08/11/2026.

[294] NVIDIA, “Serve LLMs with vLLM on One Device — Overview,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d., agg. 19 ott. 2026. [Online]. Disponibile: https://build.nvidia.com/spark/vllm. Accesso: 08/11/2026.

[295] NVIDIA, “Serve LLMs with vLLM on One Device — scheda Advanced,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/vllm/advanced. Accesso: 08/11/2026.

[296] eugr, “spark-vllm-docker — recipes/qwen3.8-27b-nvfp4-dflash2.yaml,” *GitHub*, s.d. [Online]. Disponibile: https://github.com/eugr/spark-vllm-docker/blob/main/recipes/qwen3.8-27b-nvfp4-dflash2.yaml. Accesso: 08/11/2026.

[297] NVIDIA, “Serve LLMs with vLLM Across DGX Sparks — scheda Advanced,” *NVIDIA Build*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/vllm-cluster/advanced. Accesso: 08/11/2026.

[298] NVIDIA, “Serve Models with llama.cpp,” *NVIDIA Build*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/llama-cpp. Accesso: 08/11/2026.

[299] J. Zhou e R. Chen, “NVIDIA DGX Spark Benchmarks,” *Google Sheets*, s.d. [Online]. Disponibile: https://docs.google.com/spreadsheets/d/1SF1u0J2vJ-ou-R_Ry1JZQ0iscOZL8UKHpdVFr85tNLU/edit?usp=sharing. Accesso: 08/11/2026.

[300] Qwen Team, “Qwen3.5,” *Hugging Face*, s.d. [Online]. Disponibile: https://huggingface.co/collections/Qwen/qwen35. Accesso: 08/11/2026.

[302] NVIDIA, “Serve LLMs with vLLM Across DGX Sparks,” *NVIDIA Build*, s.d., agg. 22 ott. 2026. [Online]. Disponibile: https://build.nvidia.com/spark/vllm-cluster. Accesso: 10/11/2026.

[303] NVIDIA, “DGX Spark User Guide — Release Notes (Current Software Versions e note delle release),” *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/release-notes.html. Accesso: 10/11/2026.

[304] NVIDIA, “DGX Spark User Guide — Hardware Overview,” *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/hardware.html. Accesso: 10/11/2026.

[305] NVIDIA, “DGX Spark Porting Guide — Optimization for DGX Spark,” *NVIDIA DGX Spark Porting Guide*, s.d., agg. 21 ott. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark-porting-guide/latest/optimization.html. Accesso: 08/11/2026.

[306] NVIDIA, “DGX Spark User Guide — Known Issues,” *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/known-issues.html. Accesso: 08/11/2026.

[307] NVIDIA, “DGX Spark User Guide — ConnectX-7 Networking,” *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/spark-clustering.html. Accesso: 10/11/2026.

[308] NVIDIA, “DGX Spark Porting Guide — CUDA (GPUDirect RDMA),” *NVIDIA DGX Spark Porting Guide*, s.d., agg. 21 ott. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark-porting-guide/latest/porting/cuda.html. Accesso: 08/11/2026.

[309] NVIDIA, “DGX Spark Porting Guide — Dependencies & Tools,” *NVIDIA DGX Spark Porting Guide*, s.d. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark-porting-guide/latest/porting/dependencies.html. Accesso: 08/11/2026.

[310] NVIDIA, “DGX Spark User Guide — NVIDIA Container Runtime for Docker,” *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/nvidia-container-runtime-for-docker.html. Accesso: 08/11/2026.

[311] NVIDIA, “DGX Spark Porting Guide — DGX Spark Software Stack,” *NVIDIA DGX Spark Porting Guide*, s.d., agg. 21 ott. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark-porting-guide/latest/porting/software-requirements.html. Accesso: 08/11/2026.

[312] NVIDIA, “Set Up NCCL for Multi-Node GPU Communication — Overview,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d., agg. 14 ott. 2026. [Online]. Disponibile: https://build.nvidia.com/spark/nccl. Accesso: 10/11/2026.

[313] NVIDIA, “vLLM,” *NVIDIA NGC Catalog*, s.d., agg. 29 ott. 2026 (ultimo tag 26.10-py3). [Online]. Disponibile: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/vllm. Accesso: 08/11/2026.

[314] NVIDIA, “vLLM Release 26.09,” *NVIDIA Deep Learning Frameworks Documentation*, s.d., agg. 30 ott. 2026. [Online]. Disponibile: https://docs.nvidia.com/deeplearning/frameworks/vllm-release-notes/rel-26-09.html. Accesso: 10/11/2026.

[315] vLLM Project, “vLLM v0.32.0,” *GitHub*, 18 ott. 2026. [Online]. Disponibile: https://github.com/vllm-project/vllm/releases/tag/v0.32.0; https://github.com/vllm-project/vllm/blob/v0.32.0/vllm/v1/structured_output/__init__.py. Accesso: 08/11/2026.

[316] NVIDIA, “Set Up NCCL — Two nodes,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/nccl/two-nodes. Accesso: 08/11/2026.

[317] NVIDIA, “Serve LLMs with SGLang,” *NVIDIA Build*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/sglang. Accesso: 08/11/2026.

[318] NVIDIA, “Serve Models with llama.cpp — Instructions,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/llama-cpp/instructions. Accesso: 08/11/2026.

[319] NVIDIA, “Chat with LLMs Using Open WebUI and Ollama — Overview,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d., agg. 14 ott. 2026. [Online]. Disponibile: https://build.nvidia.com/spark/open-webui. Accesso: 08/11/2026.

[320] NVIDIA, “Open WebUI on Desktop,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/open-webui/instructions. Accesso: 08/11/2026.

[321] NVIDIA, “DGX Spark — elenco dei playbook,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark. Accesso: 10/11/2026.

[322] K. Enevoldsen *et al.*, “MMTEB: Massive Multilingual Text Embedding Benchmark,” arXiv:2502.13595, 19 feb. 2025, rev. v4 13 nov. 2025; versione pubblicata in ICLR 2025. [Online]. Disponibile: https://arxiv.org/abs/2502.13595. Accesso: 08/11/2026.

[323] NVIDIA, “DGX Spark User Guide — DGX Dashboard,” *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/dgx-dashboard.html. Accesso: 10/11/2026.

[324] NVIDIA, “DGX Spark Support,” s.d., agg. 24 mar. 2026. [Online]. Disponibile: https://www.nvidia.com/en-us/support/dgx-spark/. Accesso: 08/11/2026.

[325] NVIDIA, “Connect Multiple DGX Sparks for Distributed Workloads — Overview,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d., agg. 24 set. 2026. [Online]. Disponibile: https://build.nvidia.com/spark/connect-multiple-sparks. Accesso: 10/11/2026. Sottopagine: «Cable the Devices», consultata il 25/10/2026: https://build.nvidia.com/spark/connect-multiple-sparks/cable-devices; «Configure with NVIDIA Sync»: https://build.nvidia.com/spark/connect-multiple-sparks/beginner; «Troubleshooting»: https://build.nvidia.com/spark/connect-multiple-sparks/troubleshooting (cartella del playbook nel repository NVIDIA dgx-spark-playbooks senza modifiche dal 20/10/2026).

[326] NVIDIA, “Connect Multiple DGX Sparks — Configure Manually,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/connect-multiple-sparks/advance. Accesso: 08/11/2026.

[334] A. Bourgoyne, “New Software and Model Optimizations Supercharge NVIDIA DGX Spark,” *NVIDIA Technical Blog*, 5 gen. 2026, agg. 28 gen. 2026. [Online]. Disponibile: https://developer.nvidia.com/blog/new-software-and-model-optimizations-supercharge-nvidia-dgx-spark/. Accesso: 08/11/2026.

[972] NVIDIA, “DGX Spark User Guide — Enterprise Manageability,” *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/enterprise-manageability.html. Accesso: 08/11/2026.

[973] Hugging Face, “Environment variables — huggingface_hub documentation,” s.d. [Online]. Disponibile: https://huggingface.co/docs/huggingface_hub/package_reference/environment_variables. Accesso: 08/11/2026.

[974] vLLM Project, “Usage Stats Collection,” 17 ott. 2025. [Online]. Disponibile: https://docs.vllm.ai/en/latest/usage/usage_stats/. Accesso: 08/11/2026.

[981] Open WebUI, “Environment Variable Configuration,” s.d. [Online]. Disponibile: https://docs.openwebui.com/reference/env-configuration. Accesso: 08/11/2026.

[1017] NVIDIA, “Serve LLMs with SGLang — Instructions,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/sglang/instructions. Accesso: 08/11/2026.

[1018] NVIDIA, “Set Up NCCL for Multi-Node GPU Communication — Troubleshooting,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/nccl/troubleshooting. Accesso: 08/11/2026.

[1026] NVIDIA, “CUDA Toolkit 13.4 Update 2 — Release Notes,” sez. 2.2, tabelle 2–3, agg. 12 ott. 2026. [Online]. Disponibile: https://docs.nvidia.com/cuda/archive/13.4.3/cuda-toolkit-release-notes/index.html; pagina corrente (CUDA Toolkit 13.5, agg. 5 nov. 2026): https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html. Accesso: 10/11/2026.

[1027] NVIDIA, “Minor Version Compatibility,” CUDA Compatibility, tab. 1 e limitazioni, s.d. [Online]. Disponibile: https://docs.nvidia.com/deploy/cuda-compatibility/minor-version-compatibility.html. Accesso: 08/11/2026.

[1028] NVIDIA, “GLM-5.3-Flash-FP4-Local,” *Hugging Face*, revisione 76348ddb2be2e35e39ecfc00e75dc07703f4f3d5, aggiornata il 16 ott. 2026; metadati API dei file. [Online]. Disponibile: https://huggingface.co/nvidia/GLM-5.3-Flash-FP4-Local/tree/76348ddb2be2e35e39ecfc00e75dc07703f4f3d5; https://huggingface.co/api/models/nvidia/GLM-5.3-Flash-FP4-Local?blobs=true. Accesso: 08/11/2026.

[1029] ggml-org, “llama.cpp prerelease b11871: chat JSON depth limit,” *GitHub Releases*, 25 ott. 2026. [Online]. Disponibile: https://github.com/ggml-org/llama.cpp/releases/tag/b11871. Accesso: 08/11/2026.

[1030] Ollama, “Ollama v0.40.11,” *GitHub Releases*, 24 ott. 2026. [Online]. Disponibile: https://github.com/ollama/ollama/releases/tag/v0.40.11. Accesso: 08/11/2026.

[1067] NVIDIA, “nvidia/Qwen3.8-27B-NVFP4,” *Hugging Face*, revisione 482ca0f3832238542f8f5295dde86b5f22711d80, agg. 17 set. 2026; scheda del modello, hf_quant_config.json e metadati API dei file. [Online]. Disponibile: https://huggingface.co/nvidia/Qwen3.8-27B-NVFP4; https://huggingface.co/api/models/nvidia/Qwen3.8-27B-NVFP4?blobs=true. Accesso: 08/11/2026.

[1071] NVIDIA, “vLLM — metadati del repository NGC (latestTag 26.10-py3, latestArchitectureVariants linux/arm64 e linux/amd64),” NGC Catalog API, agg. 29 ott. 2026 (al 25/10/2026: latestTag 26.10.pre1-py3, agg. 16 ott. 2026). [Online]. Disponibile: https://api.ngc.nvidia.com/v2/repos/nvidia/vllm. Accesso: 10/11/2026.

[1072] NVIDIA, “Quantize Models to NVFP4 with NVIDIA Model Optimizer,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d., agg. 21 ott. 2026. [Online]. Disponibile: https://build.nvidia.com/spark/nvfp4-quantization. Accesso: 08/11/2026.

[1073] NVIDIA, “nvidia/Llama-3.1-8B-Instruct-FP8,” *Hugging Face*, revisione 42d9515ebd69eea3a87351d079c671c3c5ff0a31, agg. 22 ago. 2025; metadati API dei file. [Online]. Disponibile: https://huggingface.co/nvidia/Llama-3.1-8B-Instruct-FP8; https://huggingface.co/api/models/nvidia/Llama-3.1-8B-Instruct-FP8?blobs=true. Accesso: 08/11/2026.

[1074] NVIDIA, “nvidia/Llama-3.1-8B-Instruct-NVFP4,” *Hugging Face*, revisione bdb54e24298451af785c0ac63c1b485e9b7400a2, agg. 15 set. 2025; metadati API dei file (l'handle `nvidia/Llama-3.1-8B-Instruct-FP4` del playbook SGLang reindirizza a questo repository). [Online]. Disponibile: https://huggingface.co/nvidia/Llama-3.1-8B-Instruct-NVFP4; https://huggingface.co/api/models/nvidia/Llama-3.1-8B-Instruct-NVFP4?blobs=true. Accesso: 08/11/2026.

[1075] NVIDIA, “nvidia/Llama-3.3-70B-Instruct-NVFP4,” *Hugging Face*, revisione f792b707002f2c613e0729c6b37bbcedc4efa94e, agg. 22 ago. 2025; `hf_quant_config.json` e metadati API dei file (l'handle `nvidia/Llama-3.3-70B-Instruct-FP4` del playbook SGLang reindirizza a questo repository). [Online]. Disponibile: https://huggingface.co/nvidia/Llama-3.3-70B-Instruct-NVFP4; https://huggingface.co/api/models/nvidia/Llama-3.3-70B-Instruct-NVFP4?blobs=true. Accesso: 08/11/2026.

[1077] LiveBench Team (Abacus.AI), “LiveBench 2.0 — Leaderboard,” release 14 ott. 2026, agg. 23 ott. 2026 (stessa pagina della voce [32], citata qui nella versione aggiornata al 23 ott. 2026). [Online]. Disponibile: https://livebench.ai/#/. Accesso: 08/11/2026.

[1103] NVIDIA, “DGX Spark User Guide — System Recovery,” *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/system-recovery.html. Accesso: 08/11/2026.

[1104] Docker Inc., “Linux post-installation steps for Docker Engine,” *Docker Docs*, s.d. [Online]. Disponibile: https://docs.docker.com/engine/install/linux-postinstall/. Accesso: 08/11/2026.

[1105] NVIDIA, “Qwen3.8-27B · DGX Spark · vLLM — ricetta 128 GB, NVFP4 (validata il 19/10/2026),” *Local AI Inference Recipes*, GitHub, tag v0.66 (commit 7b2ee45), 24 ott. 2026. [Online]. Disponibile: https://nvidia.github.io/local-ai-inference-recipes/hardware/dgx-spark/index.html?model=qwen%2Fqwen3.8-27b&os=linux&backend=vllm&memory=128&precision=nvfp4&variant=&memoryConfig=128; https://github.com/NVIDIA/local-ai-inference-recipes/blob/main/models/qwen/qwen3.8-27b/dgx-spark/linux/vllm/128gb/nvfp4/recipe.yaml. Accesso: 08/11/2026.

[1106] Docker Inc., “Port publishing and mapping,” *Docker Docs*, s.d. [Online]. Disponibile: https://docs.docker.com/engine/network/port-publishing/. Accesso: 08/11/2026.

[1107] Docker Inc., “Packet filtering and firewalls,” *Docker Docs*, s.d. [Online]. Disponibile: https://docs.docker.com/engine/network/packet-filtering-firewalls/. Accesso: 08/11/2026.

[1108] Open WebUI, “open-webui — tag del container (v0.12.4-ollama e ollama),” *GitHub Container Registry*, s.d.; v0.12.4-ollama creato il 25 ott. 2026; al 5 nov. 2026 il tag con versione più recente è v0.13.0-ollama, creato il 3 nov. 2026. [Online]. Disponibile: https://github.com/open-webui/open-webui/pkgs/container/open-webui; release https://github.com/open-webui/open-webui/releases/tag/v0.12.4. Accesso: 08/11/2026.

[1109] NVIDIA, “nvcr.io/nvidia/vllm:26.09-py3 — indice multi-architettura (sha256:557747337846eecf34486866f3d992383c6f8af869fdb0e365b5093fcaee02c2) e configurazione linux/arm64,” *NVIDIA NGC Container Registry*, s.d. [Online]. Disponibile: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/vllm/tags; API del registro https://nvcr.io/v2/nvidia/vllm/manifests/26.09-py3. Accesso: 08/11/2026.

[1110] Docker Inc., “Bind mounts,” *Docker Docs*, s.d. [Online]. Disponibile: https://docs.docker.com/engine/storage/bind-mounts/. Accesso: 08/11/2026.

[1111] LMSYS, “lmsysorg/sglang — tag v0.5.22-cu130 e latest-cu130,” *Docker Hub*, agg. 16 ott. 2026; dal 30 ott. 2026 latest-cu130 coincide con v0.5.23-cu130 (stato al 5 nov. 2026). [Online]. Disponibile: https://hub.docker.com/r/lmsysorg/sglang/tags. Accesso: 08/11/2026.

[1112] NVIDIA, “Configure a Multi-Node DGX Spark Cluster,” *NVIDIA Sync User Guide*, versione 0.148.0, s.d., agg. 30 ott. 2026 (copia versionata: https://docs.nvidia.com/sync/0.148.0/cluster-assistant.html); la versione del 2 set. 2026, intitolata “Cluster Assistant for Configuring a Multi-Node DGX Spark Cluster”, è conservata nella documentazione della versione 0.117.3 di NVIDIA Sync, voce [1144]. [Online]. Disponibile: https://docs.nvidia.com/sync/latest/cluster-assistant.html. Accesso: 10/11/2026.

[1113] NVIDIA, “Inspect and Verify a ConnectX-7 Cluster Network Plan,” *NVIDIA Sync User Guide*, s.d., agg. 30 ott. 2026. [Online]. Disponibile: https://docs.nvidia.com/sync/latest/cluster-network-inspection.html. Accesso: 10/11/2026.

[1114] eugr, “spark-vllm-docker — README e docs/NETWORKING.md (Automated cluster setup),” *GitHub*, ramo main, s.d.; revisioni in linea al 5 nov. 2026: README al commit bee724f del 4 nov. 2026, docs/NETWORKING.md al commit b7f6b4c del 1° nov. 2026. [Online]. Disponibile: https://github.com/eugr/spark-vllm-docker/blob/main/README.md; https://github.com/eugr/spark-vllm-docker/blob/main/docs/NETWORKING.md. Accesso: 08/11/2026.

[1115] NVIDIA, “Serve LLMs with vLLM Across DGX Sparks — scheda Troubleshooting,” *NVIDIA Build*, s.d. [Online]. Disponibile: https://build.nvidia.com/spark/vllm-cluster/troubleshooting. Accesso: 08/11/2026.

[1116] eugr, “eugr/spark-vllm-b12x — tag nightly-20261015,” *Docker Hub*, pubblicato il 15 ott. 2026; metadati API del tag (linux/arm64, digest sha256:03a956544e9f9b4b7ba91690469fbe7848bdbef29a7d2e0f5788368467bb3595). [Online]. Disponibile: https://hub.docker.com/v2/repositories/eugr/spark-vllm-b12x/tags/nightly-20261015. Accesso: 10/11/2026.

[1117] Open WebUI Inc., “Open WebUI License,” *Open WebUI Docs*, s.d.; testo nel repository: https://github.com/open-webui/open-webui/blob/main/LICENSE. [Online]. Disponibile: https://docs.openwebui.com/license/. Accesso: 08/11/2026.

[1118] ggml-org e contributori, “llama.cpp — Build llama.cpp locally (docs/build.md),” *GitHub*, ramo master, s.d. [Online]. Disponibile: https://github.com/ggml-org/llama.cpp/blob/master/docs/build.md. Accesso: 08/11/2026.

[1119] Docker Inc., “docker manifest inspect,” *Docker Docs — CLI reference*, s.d. [Online]. Disponibile: https://docs.docker.com/reference/cli/docker/manifest/inspect/. Accesso: 08/11/2026.

[1120] NVIDIA, “Serve LLMs with vLLM on One Device — scheda Intermediate,” *NVIDIA build.nvidia.com, DGX Spark Playbooks*, s.d., agg. 23 ott. 2026 (commit 3088315 del testo sorgente). [Online]. Disponibile: https://build.nvidia.com/spark/vllm/intermediate; testo sorgente: https://github.com/NVIDIA/dgx-spark-playbooks/blob/main/nvidia/playbook-vllm/README.md. Accesso: 08/11/2026.

[1127] NVIDIA, “vLLM Release 26.10,” *NVIDIA Deep Learning Frameworks Documentation*, s.d., agg. 30 ott. 2026 (vLLM 0.31.0, CUDA 13.4.3, Torch 2.15.0a0; Known Issues). [Online]. Disponibile: https://docs.nvidia.com/deeplearning/frameworks/vllm-release-notes/rel-26-10.html. Accesso: 08/11/2026.

[1128] vLLM Project, “vLLM v0.33.0 (tag v0.33.0rc3),” *GitHub Releases*, 2 nov. 2026. [Online]. Disponibile: https://github.com/vllm-project/vllm/releases/tag/v0.33.0rc3. Accesso: 05/11/2026.

[1129] NVIDIA, “CUDA Toolkit 13.4 — Release Notes (archivio 13.4.2; testo della versione 13.4 Update 1),” sez. 2.2, tab. 2 (ramo driver R615) e compatibilità fra versioni minori (driver ≥ 580), s.d. [Online]. Disponibile: https://docs.nvidia.com/cuda/archive/13.4.2/cuda-toolkit-release-notes/index.html. Accesso: 05/11/2026.

[1130] NVIDIA, “Model Launcher,” *NVIDIA Sync User Guide*, v. 0.148.0 (beta), agg. 30 ott. 2026 (DGX Spark singolo o cluster di 2–4 sistemi; nvidia/Qwen3.8-27B-NVFP4 con vLLM, immagine eugr/spark-vllm:nightly-20261002; OpenCode; modalità Localhost/LAN). [Online]. Disponibile: https://docs.nvidia.com/sync/latest/models.html; note di rilascio: https://docs.nvidia.com/sync/latest/release-notes.html. Accesso: 08/11/2026.

[1131] NVIDIA, “Local AI Inference Recipes — file recipe.yaml e performance.yaml delle ricette qwen/qwen3.8-27b (dgx-spark, dgx-spark-2gpu e rtx-pro), qwen/qwen3.5-122b-a10b, qwen/qwen3.6-35b-a3b (dgx-spark e geforce-rtx), qwen/qwen3.8-flash-next, nvidia/nemotron-3.5-super-vl-120b-a12b, google/gemma-4-31b e deepseek/deepseek-v4-flash,” *GitHub*, NVIDIA/local-ai-inference-recipes, tag v0.66 (commit 7b2ee45, «Release v0.66»), 24 ott. 2026; misure dal 12 al 23 ott. 2026. [Online]. Disponibile: https://github.com/NVIDIA/local-ai-inference-recipes/tree/7b2ee453234617f58e1e100c195ac3be7f79e233/models. Accesso: 10/11/2026.

[1132] NVIDIA, “Serve LLMs with NVIDIA TensorRT-LLM — scheda Multi-node serving,” *DGX Spark Playbooks*, GitHub, NVIDIA/dgx-spark-playbooks, nvidia/playbook-trt-llm/README.md, Last Updated 12 ago. 2026; cartella modificata da ultimo il 23 ott. 2026 (commit 3088315). [Online]. Disponibile: https://github.com/NVIDIA/dgx-spark-playbooks/blob/main/nvidia/playbook-trt-llm/README.md. Accesso: 08/11/2026.

[1133] NVIDIA, “Local AI Inference Recipes — z-ai/glm-5.3-flash, dgx-spark-2gpu/linux/vllm/fp4-local (recipe.yaml e performance.yaml),” *GitHub*, NVIDIA/local-ai-inference-recipes, release v0.66 del 24 ott. 2026 (commit 7b2ee45); misura del 16 ott. 2026. [Online]. Disponibile: https://github.com/NVIDIA/local-ai-inference-

recipes/blob/main/models/z-ai/glm-5.3-flash/dgx-spark-2gpu/linux/vllm/fp4-local/performance.yaml; https://github.com/NVIDIA/local-ai-inference-recipes/blob/main/models/z-ai/glm-5.3-flash/dgx-spark-2gpu/linux/vllm/fp4-local/recipe.yaml; https://nvidia.github.io/local-ai-inference-recipes/hardware/dgx-spark-2gpu/index.html. Accesso: 08/11/2026.

[1134] Terminal-Bench contributors, "Terminal-Bench 2.1," "Terminal-Bench 3.0" e "Terminal-Bench 4.0," *Terminal-Bench News*, 6 mag. 2026, 30 lug. 2026 e 28 ago. 2026. [Online]. Disponibile: https://www.tbench.ai/news/terminal-bench-2-1; https://www.tbench.ai/news/terminal-bench-3-0; https://www.tbench.ai/news/terminal-bench-4-0; indice: https://www.tbench.ai/news. Accesso: 11/11/2026.

[1135] LiveBench Team (Abacus.AI), "LiveBench 2.0 — Release 2026-10-14 e Release 2026-10-25: file di dati (table.csv, cost.csv, categories.json, breakdown.json, notes.json) e metodologia," distribuzione del sito del 30 ott. 2026 (Last-Modified 30/10/2026 22:21 GMT; bundle main.1e1ce12d.js); copie Internet Archive del 23 ott. 2026 (release 2026-10-14) e del 2 nov. 2026 (release 2026-10-25). [Online]. Disponibile: https://livebench.ai/data/20261025/table.csv; https://livebench.ai/data/20261014/table.csv; https://livebench.ai/#/methodology. Accesso: 08/11/2026.

[1136] NVIDIA, "System Repair — DGX Spark User Guide," *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/boot-repair.html. Accesso: 08/11/2026.

[1137] NVIDIA, "UEFI Settings — DGX Spark User Guide," *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/uefi-settings.html. Accesso: 08/11/2026.

[1138] NVIDIA, "NGC — DGX Spark User Guide," *NVIDIA DGX Spark User Guide*, s.d., agg. 4 nov. 2026. [Online]. Disponibile: https://docs.nvidia.com/dgx/dgx-spark/ngc.html. Accesso: 08/11/2026.

[1139] Open WebUI, "v0.13.0," *GitHub Releases*, 3 nov. 2026. [Online]. Disponibile: https://github.com/open-webui/open-webui/releases/tag/v0.13.0. Accesso: 08/11/2026.

[1140] vLLM Project, "Permissive default CORS origins allow cross-origin requests to the vLLM HTTP server," *GitHub Security Advisory* GHSA-vvpg-rrww-wpmh, 20 ott. 2026 (CVE-2026-115917, record pubblicato il 26 ott. 2026; versioni precedenti alla 0.32.0; corretto nella 0.32.0). [Online]. Disponibile: https://github.com/vllm-project/vllm/security/advisories/GHSA-vvpg-rrww-wpmh; https://cveawg.mitre.org/api/cve/CVE-2026-115917. Accesso: 08/11/2026.

[1141] NVIDIA, "nvcr.io/nvidia/vllm:26.10-py3 — indice multi-architettura (sha256:0a0de9362545e3e062750bec4ad4f722422a04a81182b5258a7deda4325e9313) e configurazione linux/arm64 (sha256:03d494c050f1497d2427f7327bcaaa59ba88ae8a8c5d6f657b018de249983c53)," *NVIDIA NGC Container Registry*, agg. 29 ott. 2026. [Online]. Disponibile: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/vllm/tags; API del registro https://nvcr.io/v2/nvidia/vllm/manifests/26.10-py3. Accesso: 08/11/2026.

[1142] ggml-org, "llama.cpp v0.9.0," *GitHub Releases*, 4 nov. 2026. [Online]. Disponibile: https://github.com/ggml-org/llama.cpp/releases/tag/v0.9.0. Accesso: 08/11/2026.

[1143] Ollama, "Ollama v0.40.15," *GitHub Releases*, 3 nov. 2026. [Online]. Disponibile: https://github.com/ollama/ollama/releases/tag/v0.40.15. Accesso: 08/11/2026.

[1144] NVIDIA, "Cluster Assistant for Configuring a Multi-Node DGX Spark Cluster," *NVIDIA Sync User Guide*, documentazione della versione 0.117.3, s.d., agg. 2 set. 2026. [Online]. Disponibile: https://docs.nvidia.com/sync/0.117.3/cluster-assistant.html. Accesso: 08/11/2026.

[1145] NVIDIA, "spark_cluster_setup — README," *DGX Spark Playbooks*, GitHub, NVIDIA/dgx-spark-playbooks, cartella nvidia/playbook-connect-multiple-sparks/assets/spark_cluster_setup, ramo main (ultimo commit della cartella del playbook: 9d3b4fc, 20 ott. 2026). [Online]. Disponibile: https://github.com/NVIDIA/dgx-spark-playbooks/tree/main/nvidia/playbook-connect-multiple-sparks/assets/spark_cluster_setup. Accesso: 08/11/2026.

[1146] eugr, "spark-vllm-docker — setup-cluster.sh e setup-cluster.py," *GitHub*, ramo main: setup-cluster.sh invariato dal 6 ott. 2026 (commit 2dd02bd), setup-cluster.py modificato da ultimo il 30 ott. 2026 (commit 33516bb). [Online]. Disponibile: https://github.com/eugr/spark-vllm-docker/blob/main/setup-cluster.sh; https://github.com/eugr/spark-vllm-docker/blob/main/setup-cluster.py; https://github.com/eugr/spark-vllm-docker/commits/main/setup-cluster.py. Accesso: 08/11/2026.

[1147] NVIDIA, "Connect Two DGX Sparks for Distributed Workloads," *DGX Spark Playbooks*, GitHub, NVIDIA/dgx-spark-playbooks, nvidia/playbook-connect-two-sparks/README.md, Last Updated 12 ago. 2026; cartella invariata dal commit 1949190 dell'8 ott. 2026. [Online]. Disponibile: https://github.com/NVIDIA/dgx-spark-playbooks/blob/main/nvidia/playbook-connect-two-sparks/README.md. Accesso: 08/11/2026.

[1148] NVIDIA, "cx7-netplan-example.yaml," *DGX Spark Playbooks*, GitHub, NVIDIA/dgx-spark-playbooks, nvidia/playbook-connect-multiple-sparks/assets, commit bc8a606 del 10 set. 2026. [Online]. Disponibile: https://github.com/NVIDIA/dgx-spark-playbooks/blob/main/nvidia/playbook-connect-multiple-sparks/assets/cx7-netplan-example.yaml. Accesso: 08/11/2026.

[1149] NVIDIA, "NCCL v2.32.3-1 Release," *GitHub*, NVIDIA/nccl, 17 set. 2026; NVIDIA, "NCCL Release Notes," *NVIDIA Deep Learning Documentation*, agg. 23 set. 2026. [Online]. Disponibile:

https://github.com/NVIDIA/nccl/releases/tag/v2.32.3-1; https://docs.nvidia.com/deeplearning/nccl/release-notes/index.html. Accesso: 08/11/2026.

[1150] NVIDIA, “nccl-tests — tag v2.21.1 (5 ott. 2026), v2.21.0 (2 ott. 2026), v2.20.0 (28 ago. 2026) e v2.19.7 (3 ago. 2026),” *GitHub*, NVIDIA/nccl-tests. [Online]. Disponibile: https://github.com/NVIDIA/nccl-tests/tags. Accesso: 08/11/2026.

[1151] NVIDIA, “dgx-spark-playbooks — cronologia dei commit di nvidia/playbook-nccl (ultimo commit 18fd621b888e0722dfb28ef35bc4451eda042763, 19 ott. 2026; assets/setup.sh modificato in 95157bc3dc56bf5180c282b3e391011b1953593c, 15 ott. 2026),” *GitHub*. [Online]. Disponibile: https://github.com/NVIDIA/dgx-spark-playbooks/commits/main/nvidia/playbook-nccl; https://github.com/NVIDIA/dgx-spark-playbooks/commit/18fd621b888e0722dfb28ef35bc4451eda042763. Accesso: 08/11/2026.

[1152] NVIDIA, “Environment Variables,” *NCCL User Guide*, NCCL 2.32.3, s.d. [Online]. Disponibile: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html. Accesso: 08/11/2026.

[1153] vLLM Project, “vllm/entrypoints/launchers/cli_args.py,” *GitHub*, commit 8244d1c44348ae8a5bbfc9f1b607021652283ed4 del 15 ott. 2026, appartenente al fork local-inference-lab e raggiungibile dalla rete di fork del repository vllm-project/vllm (valore predefinito di `allowed_origins`: ["*"]); correzione nel ramo principale: PR #61009, “[Security] Restrict default CORS origins,” integrata l’11 ott. 2026 (commit fcae5c98870ff2c20659731dce7ba3a3ff0d48aa), non compresa fra gli antenati del commit. [Online]. Disponibile: https://github.com/vllm-project/vllm/blob/8244d1c44348ae8a5bbfc9f1b607021652283ed4/vllm/entrypoints/launchers/cli_args.py; https://github.com/vllm-project/vllm/pull/61009. Accesso: 10/11/2026.

[1154] eugr, “eugr/spark-vllm-b12x — elenco dei tag (latest e nightly-20261102 aggiornati il 2 nov. 2026, digest sha256:19ab778dd57f019c44caf5df91a671cf688923d1ca9fa8ce92be0c9acb46d215; nightly-20261015 del 15 ott. 2026),” *Docker Hub*, metadati API. [Online]. Disponibile: https://hub.docker.com/v2/repositories/eugr/spark-vllm-b12x/tags?page_size=100&ordering=last_updated. Accesso: 08/11/2026.

Riferimenti in stile IEEE citati nella guida, con la numerazione della bibliografia dell'edizione Novembre 2026 (ED-2026-11): la sequenza dei numeri non è continua e i numeri da [1127] in poi identificano le fonti aggiunte nell'edizione di novembre. Le voci sono riportate senza modifiche; per le risorse web la data di accesso indica l'ultima consultazione documentata, che per le voci non rilette in questa edizione resta quella di ottobre 2026.