

1. Purpose

This document helps **platform teams and project owners** understand how to run AI workloads on BC Government's sovereign compute infrastructure. It covers:

- **How to request resources** — what your application needs, why, and how to make a namespace request that the platform team can act on
- **What tools are available** — as the OpenShift AI Hub is upgraded with tools like vLLM, KServe, LLM Compressor, InstructLab, lm-eval, and Feast, this document helps teams navigate these tools without being overwhelmed
- **How to utilize all available compute** — GPUs, CPUs with AMX acceleration, memory, and network — for the best utilization across all workloads
- **How platform teams can support allocation** — from oversubscription to underutilization, guiding resource allocation based on actual workload needs

Why this document exists: Our current AI Hub has evolved from dual-socket 2nd-generation Xeon, to 5th-generation Xeon, and is now upgrading to dual-socket Xeon 6 with GPU acceleration. Through each upgrade, not all teams and projects fully utilized the resources available. The question is: if we provision these new resources, can teams actually use them? If there is underutilization, can teams that need more compute get more? Can we do distributed computing to combine more memory, more GPUs, more CPUs — and what would that look like? This report helps guide those decisions.

Practical output: This report is intended to help platform teams and application teams translate AI use cases into schedulable workload shapes. Before a project requests "AI compute" or "a GPU," the workload should be described by model size, precision, context length, concurrency, latency target, data movement, storage growth, CPU/RAM needs, GPU/MIG needs, and operational ownership. Platform teams can then map the request to a CPU-only profile, MIG slice, full GPU, full node, or future multi-node pattern. Application teams can use the same framework to ask better questions before designing an architecture around a tool or model.

Candidate Sovereign AI Workload Patterns for CoE Investigation

The following areas are candidate workload patterns for the AI Team of Excellence to investigate as the OpenShift AI Hub and sovereign compute environment mature. These are not commitments to build specific applications. The purpose is to identify reusable patterns that can help BC Government teams decide when to use local compute, cloud services, managed AI services, or a hybrid approach.

The investigation should include the **ML/AI algorithms being used**, the **data pipeline shape**, data quality and readiness, model evaluation methods, training or fine-tuning requirements, inference cost, latency, governance, and operational ownership. In many cases, the most important question is not which tool to use, but whether the data, algorithm, compute, and workflow actually fit the problem.

Core question: not only *can this workload run on the platform?* The stronger question is: *what reusable AI pattern does this teach us that other BC Government teams can adopt?*

AREA	LIKELY AI PATTERN	ALGORITHMS, DATA PIPELINE, AND COE INVESTIGATION	WHY SOVEREIGN / LOCAL COMPUTE MATTERS
BC Wildfire	Custom models, geospatial prediction, data science workflows, model training, simulation, forecasting.	Forecasting, geospatial ML, computer vision, simulation, sensor/weather data pipelines, training and evaluation workflows, GPU/CPU scheduling, experiment tracking, model monitoring, and emergency-response latency.	Sensitive operational data, emergency response timelines, large geospatial datasets, and local compute during high-demand periods.
BC Stats	Foundational model evaluation, statistical analysis, data science, synthetic data review, and possible inference workflows.	Statistical models, forecasting, clustering, privacy-preserving analysis, synthetic-data methods, notebook workflows, local inference, model evaluation, data lineage, reproducibility, and governance.	Public-sector data stewardship, reproducible analysis, controlled model evaluation, and transparent methodology.
Finance	Fraud detection, anomaly detection, audit/compliance support, risk scoring, and local LLM-assisted analysis.	Anomaly detection, graph analytics, supervised classification, retrieval, LLM-assisted review, transaction/event pipelines, explainability, auditability, false positives, and governance.	Financial sensitivity, audit trails, controlled data access, and lower tolerance for external data exposure.
HR / Policy	Local LLM inference over HR policy, employment rules, internal guidance, case support, and semantic search.	Retrieval-augmented generation, semantic search, document pipelines, policy grounding, access controls, user feedback, citation quality, and escalation boundaries.	Sensitive employee information, policy interpretation risk, and controlled access to internal guidance.
Legal / BC Laws	Retrieval, grounding, legal/policy reference, citation support, local inference, and shared reference APIs.	Retrieval algorithms, embeddings, graph retrieval, chunking/versioning pipelines, citation quality, legal-advice boundaries, and service-to-service retrieval patterns.	Legal/policy sensitivity, citation quality, and the need for shared reference infrastructure across programs.
Health	Local inference over health policy, operational guidance, protected data workflows, and clinical/admin support patterns.	Privacy-preserving retrieval, document and policy pipelines, role-based access, model boundaries, audit logging, red-teaming, evaluation, and fallback paths.	High sensitivity, privacy obligations, Protected B/C concerns, and strict operational accountability.
SDPR / OCR	Document intelligence, OCR, form extraction, evidence comparison, human-in-the-loop review, and workflow triage.	OCR/VLM extraction accuracy, field-level confidence, document pipelines, validation logic, evidence matching, benchmark datasets, worker review impact, exception handling, and handoff risk.	Income/support documents may contain sensitive personal information, and high-stakes service workflows require strong accuracy evidence before operational adoption.
MOTT / App Remediation	AI-assisted software remediation, code analysis, migration planning, test generation, documentation, and developer productivity patterns.	Static analysis, LLM-assisted code review, repo/document analysis, dependency mapping, modernization pipelines, automated test generation, quality gates, token/cost tracking, and measurable developer throughput.	Legacy systems and remediation work often involve sensitive code, internal architecture, and operational dependencies; local or governed AI patterns help manage risk while measuring real productivity gains.
Local LLM / Inference Services	Shared local model serving for finance, HR, legal, health, document processing, and internal knowledge workflows.	Serving algorithms and runtimes, vLLM, KServe, llama.cpp, SGLang, routing, context sizing, KV-cache, quantization, batching, concurrency, cost, latency, and observability.	Keeps sensitive prompts and data in government-controlled infrastructure while testing viability before wider adoption.

Patterns to extract: ML/AI algorithm selection, data pipeline readiness, OCR/VLM extraction, software remediation, training and fine-tuning, inference serving, retrieval and grounding, evaluation, resource sizing, governance, observability, cost modelling, and operational ownership.

Procurement Status

ORDER	PLATFORM	DESCRIPTION	MODEL	ETA
CR-CAP-134	Network	CITZ – WAN Refresh – Kamloops	2× Arista DCS-7280SR3AM-48YC8-R	June 17, 2026
CR-CAP-134	Network	CITZ – WAN Refresh – Calgary	2× Arista DCS-7280SR3AM-48YC8-F	June 17, 2026
CR-CAP-142	Managed Container Service	CITZ – OpenShift AI capability	3× Dell R770 – 2 TB – 2× Xeon 6527P – 2× RTX Pro 6000	November 23, 2026

Timeline: When the hardware is installed (estimated December 2026), the CITZ AI Centre of Excellence is preparing this report alongside the new upgrades in the OpenShift AI Hub so that projects and products can understand how to utilize this new compute for diverse AI applications — sovereign inference, fine-tuning, evaluation, embedding, and preprocessing.

A Practical Example: What Does a GPU Partition Actually Support?

To illustrate why this guide matters, consider the GPU resources alone:

We have **6× RTX Pro 6000 GPUs**, each with **96 GB VRAM**. With NVIDIA MIG (Multi-Instance GPU), each GPU can be partitioned into up to 4 isolated slices — that is **4 different projects running on 1 GPU**. Across the cluster, that means a maximum of **24 GPU slots** (6 GPUs × 4 partitions). But each partition only gets **96 GB ÷ 4 = 24 GB VRAM**.

Is 24 GB enough? That depends entirely on your **application context**:

QUESTION	WHY IT MATTERS	EXAMPLE IMPACT ON 24 GB MIG SLICE
What is your application?	A single-user chat demo has different needs than a batch document processor or an agentic workflow	A demo fits easily. A production service serving a ministry does not — it needs concurrency headroom.
How many concurrent users?	Each concurrent user grows the KV-cache. Government services do not serve 1 user.	A 7B Q4 model uses ~5 GB for weights. At 4 concurrent users with 8K context: +2 GB KV-cache (7 GB total, fits). At 16 users: +8 GB (13 GB, fits). At 32 users with 16K context: +32 GB (exceeds 24 GB — OOM).
What context length do you need?	Short Q&A (2K–4K tokens) vs. long document processing (16K–32K) vs. agentic workflows (32K–128K) have very different memory profiles	At 32K context, KV-cache per user is 4× larger than at 8K. An agentic workflow requiring 64K+ context may not fit even a single user alongside model weights in a 24 GB slice.
Is this for real-time serving or batch?	Batch processing can queue requests sequentially (low concurrency, high throughput over time). Real-time serving must handle bursts.	Batch: 1–2 concurrent requests, KV-cache stays small — 24 GB works well. Real-time burst of 20 users: KV-cache spike, potential OOM.
Will your workload grow?	A model serving 4 users today may need to serve 40 after onboarding more teams. KV-cache grows linearly.	What fits today at 4 users may OOM at 20. The namespace request should account for projected growth, not just day-one load.
Is this inference, fine-tuning, or evaluation?	Training requires optimizer states + gradients (2–4× model weight size). Evaluation may not need a local GPU at all.	Fine-tuning a 7B model with LoRA needs ~15–20 GB (weights + adapters + gradients) — a 24 GB MIG slice is tight. QLoRA of a larger model does not fit.

The point: "Can my model fit?" is the wrong first question. The right questions are: What is my application? How many users? What context length? Real-time or batch? Will it grow? Those answers determine whether a MIG slice, a full GPU, multiple GPUs, or CPU-only compute is the right fit. This document helps you work through those questions.

Tools Available on the OpenShift AI Hub

As the platform is upgraded, these open source tools become available for teams to use. Each serves a specific purpose in the AI workflow:

TOOL	PURPOSE	WHEN YOU NEED IT
vLLM	LLM inference server	Serving a language model as an API endpoint (chat, RAG, document processing)
KServe	Model serving on Kubernetes	Standardized model deployment with autoscaling, canary rollout, multi-framework support
LLM Compressor	Model quantization and pruning	Reducing model size to fit in less VRAM (e.g., 70B model from 140 GB → 35 GB)
InstructLab	Model fine-tuning	Adapting a model to BC Gov domain knowledge (BC Laws, policy, ministry-specific)
lm-eval	LLM benchmarking	Measuring model accuracy before and after quantization or fine-tuning
Feast	Feature store	Managing feature data for ML models with a unified API
TruSTyAI	Bias, fairness, explainability	Drift detection and fairness monitoring for production models
Kubeflow Pipelines	ML workflow orchestration	Automating training, evaluation, and deployment pipelines

Living document: We are still learning. Tools may change, new capabilities will emerge, and research into using CPUs and GPUs together for more powerful AI inference and training is evolving rapidly. This document is grounded in what we know today and will be updated as the AI and compute landscape develops and as we observe how teams actually use these resources on the platform.

2. Key Terms

This document uses terms from distributed systems, GPU computing, and AI infrastructure. Here is what they mean in plain language:

TERM	WHAT IT MEANS	WHY IT MATTERS HERE
Heterogeneous Computing	Using different types of processors together — CPUs, GPUs, and specialized accelerators — each handling the work they do best, rather than relying on one type alone.	Our platform has both Xeon 6 CPUs (with AMX matrix acceleration) and RTX Pro 6000 GPUs. Using both together serves more projects at lower cost than GPUs alone.
Distributed Computing	Spreading a workload across multiple servers, CPUs, or GPUs so they work together on the same task — combining memory, compute, and bandwidth from multiple machines.	A model too large for 1 GPU can be split across 2. A training job can offload optimizer states to CPU RAM across the network. Distributed computing lets teams access more resources than a single server provides.
MIG (Multi-Instance GPU)	NVIDIA technology that partitions a single physical GPU into multiple isolated slices. Each slice gets its own dedicated memory and compute — like separate smaller GPUs inside one card.	A 96 GB GPU can be split into 4× 24 GB slices, letting 4 different projects run on 1 GPU without interfering with each other. Useful when many projects need a small amount of GPU.
Namespace	An isolated workspace on OpenShift where a project runs. Each namespace has its own resource limits (CPU, RAM, GPU, storage) set by the platform team.	When you request resources for your AI application, you are requesting a namespace with specific resource quotas. This guide helps you determine what to request.
VRAM	Video RAM — the memory on a GPU. This is where model weights, KV-cache, and intermediate computations live during GPU inference or training.	VRAM is the primary sizing constraint for GPU workloads. A 70B model in Q4 needs ~35–45 GB of VRAM just for weights, before any users connect.
KV-Cache	Key-Value cache — temporary memory that stores the conversation context during inference. Grows with each concurrent user and with longer conversations or documents.	KV-cache is what makes the same model work for 1 user but run out of memory with 20 users. It is the variable cost that determines how many people your model can serve simultaneously.
Quantization (Q4, Q5, Q8)	Compressing a model's weights from high precision (FP16 = 2 bytes per parameter) to lower precision (Q4 = 0.5 bytes per parameter). Reduces memory by 2–4× with small accuracy trade-off.	A 70B model goes from ~140 GB (FP16) to ~35 GB (Q4) — the difference between needing 2 GPUs and fitting on 1. LLM Compressor does this.
AMX (Advanced Matrix Extensions)	Hardware acceleration built into Intel Xeon 6 CPUs for matrix math (BF16/INT8). Similar in purpose to GPU tensor cores, but on the CPU.	AMX lets CPUs handle AI workloads like embedding, encoding, and small-model inference — tasks that would otherwise require a GPU.
NUMA (Non-Uniform Memory Access)	In a dual-socket server, each CPU socket has its own local memory. Accessing memory on the other socket is slower. NUMA-aware scheduling places workloads close to their memory.	Scheduling an AI workload on the wrong socket can reduce throughput. Platform teams use NUMA-aware placement to ensure workloads run efficiently.
Tensor Parallelism	Splitting a single model across multiple GPUs so each GPU holds part of the model and they compute together in parallel.	When a model is too large for 1 GPU, tensor parallelism lets it run on 2 or more. Requires high-bandwidth connection between GPUs.
KV-Cache Offload	Moving KV-cache data from GPU VRAM to CPU RAM when GPU memory runs low. Trades some latency for the ability to serve more concurrent users.	With 2 TB of CPU RAM per server, KV-cache offload can dramatically increase the number of concurrent users a GPU model can serve.
Sovereign Compute	Computing infrastructure owned and operated by the government, where data stays within government-controlled systems and never leaves to third-party cloud providers.	All AI workloads described in this guide run on BC Gov's own hardware in government data centers — no data leaves sovereign infrastructure.

3. Heterogeneous Compute Budget

This is a **heterogeneous compute platform**, not a GPU cluster with CPUs attached. Every compute tier — GPU, CPU+AMX, DRAM, cache, network — has finite capacity, specific bandwidth characteristics, and multi-tenant contention properties. Namespace sizing must respect all of them.

2.1 Compute Tiers — Every Tier Is Constrained

COMPUTE TIER	PER SERVER	CLUSTER TOTAL	BINDING CONSTRAINT
GPU (RTX Pro 6000)	2	6	VRAM capacity, PCIe host ↔ device bandwidth
VRAM	~192 GB	~576 GB	Shared by model weights + KV-cache + activations per GPU
CPU cores (Xeon 6527P)	48 (2× 24c)	144	Per-core LLC (6 MB/core), AMX throughput, NUMA locality
LLC (Last-Level Cache)	288 MB (2× 144 MB)	864 MB	Cache pollution under multi-tenant load; 6 MB/core is finite
DRAM	2 TB (8 channels/socket, DDR5-6400)	6 TB	~400 GB/s per socket, ~800 GB/s per server — capacity and bandwidth to support multiple AI workloads
PCIe 5.0 lanes	176 (88/socket)	528	GPU ↔ CPU data movement, NIC, storage — shared bus
Network (Arista 25G/100G)	25G–100G ports	~200 Gbps aggregate	Cross-node model movement, distributed inference, storage I/O
MIG slices (if enabled)	Up to 8	Up to 24	Isolated GPU partitions — dedicated memory/compute per slice

Heterogeneous compute: This platform has GPU, CPU+AMX, and large memory resources at every tier. The Xeon 6527P CPUs with AMX are not just "the CPU that hosts the GPU" — they are compute-capable accelerators in their own right. The goal of namespace sizing is to make sure teams utilize the right tier for their workload: GPUs for high-throughput model serving, CPUs for embedding/encoding/preprocessing, and memory for offloaded training — so that all resources stay productive.

2.2 What This Compute Can Do — The Utilization Challenge

This is a powerful platform. With 144 cores, 288 MB LLC, ~800 GB/s aggregate memory bandwidth (DDR5-6400 × 8 channels × 2 sockets per server × 3 servers), 6 TB RAM, and 6 GPUs with 576 GB VRAM — the question is not whether we have enough compute. **The question is whether teams are actually utilizing it.**

Historically, our AI Hub evolved from dual-socket 2nd-gen Xeon to 5th-gen Xeon, and not all projects fully utilized the resources available at each stage. The challenge with this upgrade is the same:

- **Are teams aware of what is available?** CPU+AMX acceleration, GPU inference, large memory pools, and high-bandwidth networking are all available — but only useful if teams know how to request and configure them.
- **Are allocated resources being used?** A namespace allocated 32 cores and 64 GB RAM but only using 4 cores and 8 GB wastes capacity that other projects could use.
- **Can underutilized capacity be shared?** If a GPU sits idle 80% of the day, can another project burst onto it? If training runs overnight, can inference reclaim those resources during the day?
- **Can teams that need more get more?** Distributed computing can combine GPUs, CPUs, and memory across servers — but the namespace and scheduling must be configured to enable it.

The compute tiers in Section 2.1 show what is available per server. The profiles in Section 4 help teams request what they actually need. The scheduling policies in Section 6 help the platform team ensure that idle resources flow to teams that can use them.

2.3 Multi-Tenant Platform Overhead

COMPONENT	ESTIMATED OVERHEAD	RESOURCES CONSUMED
OpenShift platform (API, etcd, ingress, SDN)	~8–12 cores, ~32–48 GB RAM per server	CPU, RAM, storage IOPS
Monitoring (Prometheus, Grafana stack)	~4–8 cores, ~16–32 GB RAM cluster-wide	CPU, RAM, storage
NVIDIA GPU Operator + device plugin	~2–4 cores, ~4–8 GB RAM per server	CPU, RAM, PCIe management
Storage operators (ODF, CSI drivers)	~4–8 cores, ~16–32 GB RAM cluster-wide	CPU, RAM, storage bandwidth
Effective AI workload budget		~110–120 CPU cores, ~5.0–5.4 TB RAM, 6 GPUs

3. How Workload Math Determines Resource Requests

When a project requests resources, the platform team needs to understand **what the application actually requires** — not just "give me a GPU" or "give me 32 cores." The math below helps both sides speak the same language.

3.1 GPU Memory: Model Weights + KV-Cache + Concurrency

GPU memory is shared between the model weights (fixed cost) and the KV-cache (variable cost that grows with users and context length):

GPU memory used = Model weights + KV-cache (per user × concurrent users) + Framework overhead

Example: Llama-3.1-70B, 8K context, FP16 KV-cache

KV-cache per request = 2 × 80 layers × 8 kv_heads × 128 head_dim × 8192 context × 2 bytes ≈ 2.6 GB

Scenario A – Low concurrency (batch processing, 4 users):

Weights (Q4): ~35 GB + KV-cache: ~10 GB = ~45 GB → fits on 1 GPU (96 GB)

Scenario B – Medium concurrency (production service, 16 users):

Weights (Q4): ~35 GB + KV-cache: ~42 GB = ~77 GB → fits on 1 GPU, but limited headroom

Scenario C – High concurrency (ministry-wide service, 32 users):

Weights (Q4): ~35 GB + KV-cache: ~84 GB = ~119 GB → exceeds 1 GPU, needs 2 GPUs or KV-cache offload to CPU RAM

This is why the same model can work for a pilot (4 users) but needs a different profile for production (32 users). The profiles in Section 4 account for this.

3.2 CPU Compute: Active AI Participants, Not Just GPU Hosts

Modern CPUs are not bystanders in AI inference. The Xeon 6527P with AMX (BF16/INT8 matrix acceleration), AVX-512, and ~400 GB/s memory bandwidth per socket is a **compute-capable AI accelerator in its own right**. This is not theoretical — recent research and industry direction confirm it:

Heterogeneous EPD Disaggregation (Intel + SGLang/LMSYS, June 2026): Offloading vision encoding to Xeon 6 CPUs (AMX) while GPUs handle language model decode achieved **~1.2–1.3× request throughput improvement** and **~1.3–30× P99 TPOT (time per output token) reduction** over pure GPU EPD. The CPU resources are effectively "free" — they exist in every server regardless. Using them for appropriate inference stages improves platform utilization without additional hardware cost.

Arm AGI CPU (Arm, 2026): Arm has released its first production silicon designed specifically for agentic AI workloads — 136 Neoverse V3 cores, 2 MB L2 cache per core, DDR5-8800 support, and PCIe Gen6. This signals that the industry recognizes CPUs as first-class AI processors, not just GPU hosts. The compute direction is heterogeneous: CPUs and GPUs working together, each handling the stages where they are most efficient.

What CPU compute enables on this platform:

CPU WORKLOAD	HOW IT WORKS	WHY IT MATTERS FOR UTILIZATION
Vision/audio encoding (EPD)	AMX handles ViT/CNN encoder stages; GPU handles language decode	1.2–1.3× throughput over GPU-only; frees GPU decode capacity
CPU inference (7B–13B models)	Full model inference on CPU with AMX; ~800 GB/s bandwidth per server supports weight streaming	No GPU needed; at realistic 10–30% utilization, CPUs are 3–5× more power-efficient than idle GPUs
Embedding generation	Small models (~0.5–2 GB) with high throughput on CPU	Multiple embedding services per server; GPU is overkill for this workload
Preprocessing / tokenization / OCR	Data pipeline, batch document processing	Runs alongside GPU workloads with minimal resource contention
Training offload (ZeRO-Offload)	Optimizer states in 2 TB CPU RAM instead of VRAM	Makes 70B QLoRA feasible on 2 GPUs instead of 4+
Agentic orchestration	Tool calling, routing, planning stages that do not need GPU	Agentic workloads spend most time on orchestration, not matrix math — CPU handles this naturally

The key insight from the LMSYS research: the **device-aware weighted router** dynamically balances work between CPU and GPU pools based on actual load. When CPUs are underutilized relative to their capacity, the router sends encoding work to CPUs. When GPUs are under pressure, CPUs absorb overflow. This is the operational model that maximizes utilization of all compute — and it is directly applicable to our Xeon 6527P + RTX Pro 6000 configuration.

Utilization, not hierarchy: The question is not "GPU for real work, CPU for leftovers." The question is: which stage of your AI pipeline benefits most from which compute tier? Encoding → CPU (AMX). Decode → GPU. Embedding → CPU. Training gradients → GPU. Optimizer states → CPU RAM. Orchestration → CPU. When all tiers are utilized for the stages they handle best, the platform serves more projects, more efficiently, at lower power cost.

3.3 Utilization: From Oversubscription to Underutilization

In a shared platform, the challenge is not just "do we have enough" — it is making sure resources flow to where they are needed:

SITUATION	WHAT HAPPENS	HOW THIS GUIDE HELPS
Oversubscription	Too many projects requesting GPUs; queuing, preemption, or OOM	Profiles help right-size requests; MIG partitioning shares GPUs; CPU-first defaults reduce GPU pressure
Underutilization	Resources allocated but idle — GPUs reserved but not running inference, CPU cores allocated but not used	Scheduling policies (burst, preemption) let idle resources flow to other projects; monitoring identifies waste
Unaware teams	Projects do not know CPU inference, AMX acceleration, or quantization are options — default to "I need a GPU"	Profiles E and D show what can run without GPU; tools table shows what is available; intake form captures actual needs
Growing workloads	A project starts with 4 users, scales to 40; KV-cache grows, memory pressure increases	Profiles account for growth; distributed inference and KV-cache offload are available when a single GPU is no longer enough

4. Reference Namespace Profiles

Five profiles cover the spectrum of AI workloads. Each profile helps project owners and platform teams agree on what resources an application needs — and why — based on the workload type, not guesswork.

A Lightweight Inference (7B–8B Model)

Chat, RAG, classification, routing, embedding backend

RESOURCE	REQUEST	LIMIT	RATIONALE
GPU	1 (or 1 MIG slice)	1	7B Q4 fits in ~5 GB VRAM; KV-cache at 8K ≈ 0.5 GB/request
VRAM footprint	~8–16 GB	~24 GB	Weights + KV-cache for 8–16 concurrent users
CPU	4–8 cores	16 cores	Tokenization, API serving — low bandwidth pressure
RAM	16 GB	32 GB	Scheduler state, request buffers
Storage	20 GB	50 GB	Model weights cache, logs
NUMA affinity	Same socket as GPU	—	GPU PCIe path must be NUMA-local; cross-socket adds latency
Bandwidth pressure	Low	—	Weights on GPU; CPU does tokenization only — minimal DRAM streaming

Cluster capacity: Up to 6 instances (1 per GPU), or up to 24 with MIG. **Co-tenancy:** Multiple Profile A workloads can share a NUMA domain without significant cache/bandwidth contention because CPU work is light.

B Medium Inference (30B–70B Model, Quantized)

Higher-accuracy sovereign inference, policy analysis, complex extraction

RESOURCE	REQUEST	LIMIT	RATIONALE
GPU	1 (Q4) or 2 (FP16)	2	70B Q4 ≈ 35–45 GB; FP16 needs tensor parallelism across 2 GPUs
VRAM footprint	~45–96 GB	~192 GB	Weights + KV-cache at 2.6 GB per 8K concurrent request
CPU	8–16 cores	32 cores	Heavier tokenization, longer sequences, potential KV-cache offload scheduling
RAM	32 GB	64 GB	Scheduler state, KV-cache overflow to CPU RAM if configured
Storage	50 GB	100 GB	Larger model weights, evaluation artifacts
NUMA affinity	Same socket as GPU(s)	—	Tensor-parallel across 2 GPUs: both GPUs should be on the same server, ideally same PCIe root complex
Bandwidth pressure	Moderate	—	PCIe traffic for KV-cache offload; DRAM bandwidth if CPU-offload paths are enabled

Cluster capacity: 2–3 quantized 70B instances (1 GPU each), or 1–2 FP16 instances (2 GPUs each). **Co-tenancy:** Profile B with KV-cache offload can pressure DRAM bandwidth — avoid co-locating two Profile B workloads on the same NUMA domain.

Quantization impact: Without quantization, a 70B model requires 2+ GPUs (FP16). With Q4, it fits on 1 GPU by weight alone. But this shifts pressure to KV-cache — at high concurrency, KV-cache can exceed the freed VRAM, requiring offload to CPU DRAM and consuming memory bandwidth. Quantization changes the bottleneck, it does not eliminate it.

c Fine-Tuning / Training (LoRA / QLoRA)

Domain adaptation, custom classifiers, InstructLab-based fine-tuning

RESOURCE	REQUEST	LIMIT	RATIONALE
GPU	1 (7B LoRA) or 2 (70B QLoRA)	2	Base model + optimizer state + gradients
VRAM footprint	~20–60 GB	~192 GB	Weights + adapters + gradient checkpoints
CPU	16–32 cores	48 cores	Data loading, synthetic data generation, optimizer compute (ZeRO-Offload)
RAM	64–256 GB	512 GB	CPU offload for optimizer states (ZeRO-Offload: optimizer states = 2–4× model weights)
Storage	100 GB	500 GB	Training data, checkpoints, model registry
NUMA affinity	Dedicated NUMA domain preferred	—	Optimizer offload saturates DRAM bandwidth; co-tenants will be starved
Bandwidth pressure	High	—	~100–200 GB/s DRAM streaming for optimizer states; PCIe saturated for GPU ↔ CPU gradient sync

Why 2 TB RAM matters: The 2 TB RAM per server enables CPU offloading of optimizer states, making 70B QLoRA feasible on 2 GPUs instead of 4+. But this consumes significant memory bandwidth — a training namespace should own its NUMA domain, not share it.

d Batch Evaluation / Benchmarking

lm-eval benchmarks, model comparison, quantization quality validation

RESOURCE	REQUEST	LIMIT	RATIONALE
GPU	0 (API-targeted) or 1	1	lm-eval can target a vLLM endpoint — no local GPU needed
CPU	8–16 cores	32 cores	Evaluation framework, scoring, data loading
RAM	16 GB	64 GB	Task datasets, results accumulation
Storage	20 GB	100 GB	Evaluation results, comparison reports
NUMA affinity	Any (low sensitivity)	—	API-targeted eval is network-bound, not compute-bound
Bandwidth pressure	Low	—	Scoring is compute-light; data loading is bursty not sustained

Recommended pattern: Evaluation consumes the inference API, not raw GPU. Good co-tenant — low bandwidth pressure, tolerant of NUMA placement.

E CPU-Only Inference / Preprocessing

Embedding generation, OCR, small model classification, batch summarization, AMX-accelerated inference

RESOURCE	REQUEST	LIMIT	RATIONALE
GPU	0	0	No GPU required
CPU	8–24 cores	48 cores	AMX-accelerated inference, embedding, preprocessing
RAM	16–64 GB	128 GB	Model weights in RAM (streaming through LLC), batch buffers
Storage	20 GB	50 GB	Model cache, intermediate outputs
NUMA affinity	Same socket — critical	—	CPU inference streams weights from DRAM through LLC. Cross-socket access halves bandwidth.
Bandwidth pressure	High per socket	—	7B Q4 at ~307 GB/s socket bandwidth ≈ 88 tok/s. Two on same socket: ~44 tok/s each.

Critical: CPU inference is **memory-bandwidth-bound**, not core-count-bound. Scheduling 2 CPU inference workloads on the same socket halves throughput for both. The platform should limit Profile E density per NUMA domain based on bandwidth, not cores.

5. Capacity Scenarios (Multi-Tenant Aware)

These scenarios show workload placement across the 3-server, 6-NUMA-domain cluster. Each scenario accounts for co-tenancy constraints — not just resource sums.

Scenario 1: Conservative (Shared Inference + Evaluation)

NAMESPACE	PROFILE	SERVER	NUMA	GPUS	VRAM	CPU	RAM	BW PRESSURE
sovereign-chat (8B Q4)	A	S1	Sock 0	1	~12 GB	8	32 GB	Low
bc-laws-service (8B Q4)	A	S1	Sock 1	1	~12 GB	8	32 GB	Low
doc-extraction (8B Q4)	A	S2	Sock 0	1	~12 GB	8	32 GB	Low
embedding-service	E	S2	Sock 1	0	0	16	32 GB	Moderate
evaluation-pipeline	D	S3	Any	0	0	8	16 GB	Low
experimentation (8B)	A	S3	Sock 0	1	~16 GB	8	32 GB	Low
Total used				4	~52 GB	56	176 GB	
Remaining				2	~524 GB	~74	~5.2 TB	

Co-tenancy analysis: Profile A workloads have low bandwidth pressure — safe to co-locate. S2 pairs a GPU inference (Sock 0) with CPU embedding (Sock 1) on separate NUMA domains — no contention. S3 has headroom for additional workloads on Sock 1.

Scenario 2: Moderate (Mixed Inference + Fine-Tuning)

NAMESPACE	PROFILE	SERVER	NUMA	GPUS	VRAM	CPU	RAM	BW PRESSURE
sovereign-chat (70B Q4)	B	S1	Sock 0	1	~55 GB	16	64 GB	Moderate
bc-laws-service (8B Q4)	A	S1	Sock 1	1	~12 GB	8	32 GB	Low
doc-extraction (8B Q4)	A	S2	Sock 0	1	~12 GB	8	32 GB	Low
embedding-service	E	S2	Sock 1	0	0	16	32 GB	Moderate
bc-laws-finetuning (7B LoRA)	C	S3	Sock 0+1	1	~20 GB	16	128 GB	High
evaluation-pipeline	D	S3	Sock 1	0	0	8	16 GB	Low
Total used				4	~99 GB	72	304 GB	
Remaining				2	~477 GB	~58	~5.1 TB	

Co-tenancy analysis: Training (S3) gets its own server with high bandwidth pressure from optimizer offload. Eval co-locates on S3 Sock 1 — tolerable because eval is API-targeted (network-bound, not bandwidth-bound). The 70B inference on S1 Sock 0 is paired with lightweight 8B on Sock 1 — safe because Profile B's CPU-side work is moderate.

Scenario 3: Aggressive (Maximum Utilization)

NAMESPACE	PROFILE	SERVER	NUMA	GPUS	VRAM	CPU	RAM	BW PRESSURE
sovereign-chat (70B Q4)	B	S1	Sock 0	1	~55 GB	16	64 GB	Moderate
bc-laws-service (70B Q4)	B	S1	Sock 1	1	~55 GB	16	64 GB	Moderate
doc-extraction (8B Q4)	A	S2	Sock 0	1	~12 GB	8	32 GB	Low
connected-services (8B Q4)	A	S2	Sock 1	1	~12 GB	8	32 GB	Low
bc-laws-finetuning (70B QLoRA)	C	S3	Sock 0+1	2	~120 GB	32	256 GB	High
embedding-service	E	S2	Sock 1	0	0	16	32 GB	Moderate
evaluation-pipeline	D	S1	Any	0	0	8	16 GB	Low
Total used				6	~254 GB	104	496 GB	
Remaining				0	~322 GB	~26	~4.9 TB	

Contention risk: S1 runs two 70B models on separate sockets — each GPU's PCIe path is NUMA-local, but both compete for cross-socket interconnect if any CPU work touches the other socket's memory. S2 co-locates embedding (Sock 1) with lightweight GPU inference (Sock 1) — monitor for bandwidth pressure. S3 is fully consumed by training — no co-tenants possible during active training. **At this utilization, scheduling policy is not optional.**

6. Scheduling and Placement Policy

In a heterogeneous multi-tenant cluster, scheduling is not "give me N cores" — it is workload placement across a topology of servers, sockets, NUMA domains, GPUs, and bandwidth envelopes.

POLICY	PURPOSE	MECHANISM
NUMA-aware scheduling	Ensure workloads run on the socket closest to their GPU/memory	Topology Manager (single-numa-node policy), CPU Manager (static)
Bandwidth-aware co-tenancy	Prevent two bandwidth-heavy workloads on the same NUMA domain	Anti-affinity rules; Profile C/E density limits per socket
GPU partitioning (MIG)	Isolated GPU slices for lightweight workloads	NVIDIA MIG with GPU Operator; dedicated memory/compute per slice
Preemption	Training yields GPUs to inference on demand	PriorityClass: inference=high, training=low, eval=lowest
Burst capacity	Training uses all 6 GPUs overnight when inference is idle	ResourceQuota with burst limits, cron-triggered scaling
CPU-first default	New projects start on CPU-only, graduate to GPU with evidence	Default profile = E; GPU allocation requires workload math
LLC partitioning (RDT/CAT)	Prevent cache pollution between co-tenants on same socket	Intel RDT Cache Allocation Technology — assign LLC ways per workload

7. AI Workload Boundary for GPU-Enabled Hosts

GPU-enabled hosts should be reserved for AI workload execution and tightly coupled AI platform services. This keeps expensive GPU, CPU+AMX, memory, PCIe, and network capacity available for workloads that actually need AI compute locality. General application hosting should continue to use standard OpenShift or application hosting patterns unless there is a documented workload-sizing reason to co-locate with AI compute.

Boundary principle: An AI workload is compute, data, model, evaluation, or orchestration work that directly supports AI/ML training, fine-tuning, inference, evaluation, retrieval, simulation, model serving, or model-quality operations. "AI-adjacent" application code should not automatically qualify for GPU-enabled host placement.

ALLOWED BY DEFAULT	WHY IT BELONGS ON AI COMPUTE
Model inference and model serving	Directly consumes GPU, CPU+AMX, VRAM, RAM, or model-runtime resources.
VLM, OCR, ASR, document intelligence, and batch scoring	AI processing workloads where throughput, latency, model size, or data locality affect compute placement.
Embeddings, retrieval indexing, and vectorization jobs	Often CPU/GPU/memory intensive and directly supports RAG, search, or grounding workloads.
Training, fine-tuning, LoRA/QLoRA, simulation, forecasting, and geospatial ML	Consumes accelerator, CPU, memory, storage, and scheduling capacity as part of model development or AI analysis.
Model evaluation, benchmarking, red-team tests, drift checks, and quality monitoring	Produces the evidence needed to decide whether a model or platform path should be adopted, scaled, changed, or stopped.
AI pipeline workers and orchestration jobs	Allowed when they directly execute or coordinate AI workload stages such as data preparation, model calls, evaluation, or batch inference.
AI notebooks and workbenches	Allowed for AI/ML experimentation, benchmark development, workload profiling, and platform validation.

EXCLUDED BY DEFAULT	WHY IT SHOULD RUN ELSEWHERE
General React frontends, static websites, dashboards, and UI shells	These normally do not require GPU, CPU+AMX, NUMA-local memory, or AI compute locality.
Generic Node/Nest/Java/.NET backends and CRUD APIs	Application logic should run on standard hosting unless it is tightly coupled to AI runtime execution.
Ordinary relational databases or application state stores	Databases should use normal managed database or application hosting patterns unless there is a specific AI data-locality requirement.
Project management tools, admin portals, and non-AI business applications	These consume capacity without improving AI workload execution.

Practical rule: An API that directly serves a model, launches AI jobs, manages model/runtime execution, or performs AI pipeline stages may belong near AI compute. An API that only manages users, forms, settings, dashboards, or general workflow state should run on standard application infrastructure. A lightweight UI for an AI demo may be useful, but the UI itself should not consume GPU-enabled host capacity unless a documented technical reason exists.

This boundary matters for licensing and operations. If GPU-enabled hosts are licensed or operated as AI-only hosts, the platform needs a defensible definition of what counts as AI work. Without that definition, any application can be described as AI-adjacent and the cluster can slowly become general-purpose hosting with expensive accelerators attached.

8. Dynamic Workload Sizing, Queues, and Burst Utilization

AI workloads are often bursty, experimental, batch-oriented, or time-boxed. A static namespace allocation model can leave expensive GPU, CPU, memory, and network capacity idle. A better operating model is to define guaranteed baseline allocations, then allow approved AI workloads to temporarily use more capacity when the cluster is underutilized.

Dynamic sizing principle: if the cluster is quiet, approved workloads should be able to burst into unused CPU, GPU, MIG, memory, or distributed capacity to finish faster. If the cluster is busy, workloads should fall back to their guaranteed quota, queue, or follow priority and pre-emption rules. This requires licensing, host eligibility, scheduling policy, quota rules, and monitoring to agree with each other.

OPERATING MODE	WHAT IT MEANS	WHY IT MATTERS	REQUIRED CONTROLS
Guaranteed allocation	Each approved workload or namespace receives a baseline CPU/RAM/GPU/MIG allocation.	Protects production and priority workloads from starvation.	ResourceQuota, LimitRange, namespace profile, support owner, monitoring baseline.
Burst utilization	Workloads temporarily consume idle cluster capacity beyond their baseline allocation.	Improves utilization and lets batch, evaluation, training, or simulation jobs finish faster when resources are idle.	Burst limits, priority classes, scheduler policy, approval rules, utilization monitoring.
Priority queues	Workloads wait or run based on priority: production inference above training, training above evaluation, evaluation below live service traffic.	Prevents low-priority experiments from blocking high-priority operational workloads.	Queue policy, pre-emption policy, job controller, user/project priority mapping.
Pre-emption	Lower-priority jobs yield resources when higher-priority workloads need them.	Keeps production inference available while still allowing training/evaluation to use idle capacity.	PriorityClass, checkpointing, restart policy, job tolerance for interruption.
Oversubscription	The platform accepts more requested capacity than is physically available, assuming not all workloads peak at once.	Can increase utilization, but unmanaged oversubscription creates queues, OOM events, contention, and missed service targets.	Measured utilization, admission control, quotas, alerts, workload profiles, executive/service priority rules.
Availability-aware placement	Workloads are placed based on resilience needs: best-effort, recoverable batch, production inference, or future DR-capable service.	Not every AI workload needs disaster recovery. Critical AI services may need a different architecture and budget.	Service tier, restart policy, backup/data recovery plan, DR decision, support model.
Dynamic distributed execution	Approved jobs temporarily use multiple GPUs, full nodes, CPU pools, or cross-node resources when topology allows.	Allows large batch jobs, model evaluation, simulations, or training to finish faster without permanently reserving the full cluster.	Confirmed topology, NIC/switch bandwidth, storage path, distributed runtime support, queueing, monitoring.

Dynamic sizing is different from traditional web application hosting. Web workloads are often sized for steady-state service capacity and horizontal scaling. AI workloads may need temporary access to large compute windows: a training run overnight, a document batch over a weekend, a benchmark sweep, a simulation burst, or an evaluation job that can wait behind production inference.

LAYER	ROLE IN DYNAMIC SIZING	QUESTIONS TO RESOLVE
OpenShift / Kubernetes	Provides namespaces, quotas, scheduling, node labels, taints/tolerations, priority classes, jobs, and resource requests/limits.	Can namespaces burst beyond baseline quota? How are priority and pre-emption configured? Can jobs target NUMA/GPU/MIG topology correctly?
NVIDIA stack / RunAI / NeMo / Triton-Dynamo	May provide GPU orchestration, model serving, workload queues, model/runtime management, and accelerator-aware scheduling.	Does it manage GPU sharing, queues, model serving, and utilization better than base OpenShift? What licensing is required? What does it solve versus what still depends on topology?
VMware / NSX / VMware Private AI	May affect host placement, isolation, virtual networking, microsegmentation, GPU exposure, and operational management.	Are GPU nodes bare metal, virtualized, passed through, or otherwise constrained? Does VMware add useful control, or does it limit low-level AI placement and bandwidth assumptions?
Licensing model	Determines which hosts can run which workloads and whether mixed or AI-only host modes are cost-effective.	Are GPU-enabled hosts licensed as AI-only hosts or mixed-use hosts? Are licenses per physical host, per core, per GPU, per user, or per platform component? Does bursting across hosts increase cost?
Monitoring and chargeback/showback	Provides evidence for utilization, contention, cost, and expansion decisions.	Can we measure GPU utilization, VRAM, CPU+AMX, memory bandwidth, PCIe, storage, network, queue time, and namespace-level usage?

Cost-control point: dynamic utilization only works if the licensing model supports the desired placement pattern. A cheaper license may reduce flexibility. A broader license may improve utilization but increase cost. The platform decision should therefore compare not only product features, but also the cost of flexibility: which workloads can run where, who can burst, what must queue, and what evidence is needed before expanding capacity.

8.1 Configuration, Licensing, Cost, and Workload Justification Matrix

As Platform, ADSL, EA, Red Hat, NVIDIA, VMware, and procurement details become clearer, the roadmap should maintain a simple decision matrix. The purpose is to connect each possible platform configuration to the software stack, licensing model, cost implication, operational constraint, and workload justification. This prevents the platform decision from becoming a generic tool preference exercise.

CONFIGURATION OPTION	SOFTWARE / PLATFORM STACK	LICENSING MODEL	COST IMPLICATION	WORKLOADS IT SUPPORTS	CONSTRAINTS / RISKS	DOES THE COST JUSTIFY THE WORKLOAD?	STATUS EVIDENCE NEEDED
GPU-enabled hosts reserved for AI-only workloads	OpenShift AI / NVIDIA stack / VMware Private AI or selected combination	To confirm: per physical host, per core, per GPU, per user, or platform component.	Potentially cleaner scope, but may require dedicated licensing for each AI host.	Inference, model serving, VLM/OCR/ ASR, training, fine-tuning, evaluation, embedding, simulation, and tightly coupled AI pipeline workers.	Requires clear AI workload boundary. Non- AI application hosting should be excluded by default.	Unknown until workload inventory and licensing facts are confirmed.	Need confirmed licensing terms, ho eligibility rules, and workload profiles.
GPU-enabled hosts allow mixed AI and non-AI workloads	OpenShift + AI platform components with general application co-location	To confirm: whether mixed-use mode expands license scope or cost.	May improve utilization, but could increase licensing cost and operational complexity.	AI workloads plus general APIs, databases, frontends, and supporting services.	Risk of expensive AI nodes becoming general app hosting. More difficult to defend GPU/ CPU/RAM allocation and isolate contention.	Likely only justified if licensing allows it and monitoring proves unused capacity can safely support non-AI workloads.	Need licensing comparis scheduler constraint and policy excluding ordinary & workload
MIG-enabled shared GPU slices	NVIDIA GPU Operator / MIG strategy / OpenShift scheduling / optional RunAI	To confirm: whether MIG usage changes license or support model.	Improves multi-tenant GPU access if supported; reduces need for full-GPU allocation.	Lightweight inference, small VLM/ OCR workloads, evaluation, small model serving, demos, low- concurrency services.	Each slice has limited VRAM. A 24 GB slice may not support larger models, long context, or high concurrency.	Likely justified for many small workloads if MIG support and isolation are validated.	Need confirmed MIG supp partition sizes, scheduling behaviour monitorin and represent tests.
Full GPU or multi- GPU allocation	vLLM, SGLang, Triton- Dynamo, KServe, PyTorch, NVIDIA stack, OpenShift AI	To confirm: software stack and host licensing required for full GPU / multi-GPU jobs.	High-value but scarce allocation. Should require workload math and evidence.	70B+ inference, high- throughput serving, large VLM workloads, fine-tuning, training, simulations, large batch jobs.	Can starve other teams if over-allocated. Multi-GPU performance depends on PCIe, topology, storage, and network.	Justified when model size, precision, context, concurrency, latency, or training profile requires it.	Need benchmark evidence, topology details, ai queue/pre emption policy.
CPU-first / CPU+AMX workloads	OpenShift CPU namespaces, llama.cpp,	To confirm: whether CPU-only AI	Can reduce GPU pressure and improve utilization of	Embeddings, preprocessing, OCR pipelines,	CPU inference is memory- bandwidth and NUMA	Often justified when workload does not require GPU,	Need Inte Dell/Platf support p AMX

CONFIGURATION OPTION	SOFTWARE / PLATFORM STACK	LICENSING MODEL	COST IMPLICATION	WORKLOADS IT SUPPORTS	CONSTRAINTS / RISKS	DOES THE COST JUSTIFY THE WORKLOAD?	STATUS EVIDENCE NEEDED
	oneDNN/ MKL, PyTorch CPU, embedding services, batch workers	workloads on GPU- enabled hosts affect licensing, or whether they should run on separate CPU pools.	high-memory Xeon capacity.	small model inference, batch classification, evaluation, orchestration, CPU offload.	sensitive. Not all CPU jobs belong on GPU- enabled hosts.	or when CPU stages improve overall heterogeneous utilization.	validation NUMA pc and workl benchma
Dynamic burst / queue-based utilization	OpenShift/ Kubernetes queues, PriorityClass, optional RunAI, job scheduling, monitoring	To confirm: whether bursting across hosts changes licensing or support cost.	Can increase utilization of expensive hardware, but may require broader licensing or platform tooling.	Batch evaluation, training windows, simulations, benchmark sweeps, document processing bursts, off- hours jobs.	Requires strong monitoring, priority rules, pre-emption, and clear fallback to guaranteed quotas.	Justified if idle capacity is common and burst workloads produce measurable time/cost benefit.	Need utilization data, que policy, licensing facts, and platform schedule decision.
DR-enabled AI / GPU capability	Emerald DR with AI/GPU capacity, if funded and designed	To confirm: additional host/GPU/ platform licensing required for DR.	Likely significant additional cost. May be future-state only unless critical workloads require it.	Critical AI services that must survive primary-site or cluster failure.	If GPUs exist only in the primary site, GPU workloads may be unavailable during DR activation.	Only justified when an AI workload becomes operationally critical and has approved resilience requirements.	Need ser criticality assessment DR architectu decision, funding p

Decision question: for each configuration, the team should ask: what software stack enables it, what license permits it, what does it cost, what workloads does it support, and does that cost make sense for the workloads BC Government actually plans to run?

9. Validating Large Resource Requests

When a project requests large allocations, there should be concrete justification:

JUSTIFIED	EXAMPLE	TYPICAL NEED
Training with CPU offload	70B QLoRA, ZeRO-Offload	128–512 GB RAM
Large batch embedding	Millions of documents for vector index	64–256 GB RAM
Multiple model instances	3–4 models for A/B testing	32–128 GB RAM

SHOULD BE QUESTIONED	WHY
"We might need it"	Over-reservation starves other projects. RAM is shared.
"Framework defaults say so"	Defaults are for single-tenant machines, not shared clusters.
"We want headroom"	Headroom should be at cluster level (platform team), not namespace level.

10. Information Needed for Precise Calculations

DETAIL	IMPACT
Exact GPU VRAM (48 GB vs 96 GB)	Changes all VRAM budgets and MIG partition options
MIG support and partition options	Determines whether lightweight profiles can share a GPU
NIC bandwidth per server	Determines feasibility of distributed training / multi-node inference
Initial project list from leadership	Converts abstract scenarios into real namespace allocations

11. Recommendation

1. **Adopt the 5-profile system (A–E)** with NUMA affinity and bandwidth pressure as first-class sizing dimensions — not just GPU/CPU/RAM counts.
2. **Start with Scenario 1** (conservative) to validate the platform topology, then graduate to Scenario 2 once contention characteristics are measured.
3. **Default new projects to Profile E** (CPU-only). GPU allocation requires workload math: model size × precision × concurrency × context length, not "we need GPUs."
4. **Schedule by NUMA domain, not by node.** The scheduling unit is the socket+memory+PCIe complex, not the server. Two bandwidth-heavy workloads on the same socket degrade each other regardless of core count.
5. **Quantize aggressively.** LLM Compressor (Q4/Q5) changes the bottleneck from VRAM capacity to KV-cache bandwidth — understand both before approving a namespace.
6. **Measure before expanding.** Before adding workloads beyond Scenario 1, measure actual memory bandwidth utilization, LLC miss rates, and PCIe throughput per namespace. The numbers in this document are estimated from hardware specs — operational data will refine them.

Bottom line: This is a heterogeneous compute platform — 6 GPUs, 6 NUMA domains (144 MB LLC each), 6 TB DRAM (~1.8 TB/s aggregate bandwidth), and 200 Gbps fabric. Every tier is finite. Namespace sizing must respect cache locality, memory bandwidth, NUMA topology, and PCIe placement — not just GPU count. The bottleneck shifts by workload stage; the profiles and placement policies in this document are designed to keep every tier productive without starving co-tenants.