B.C. Gov AI Compute Modernization – Integration & Validation Technical Report

Prepared by: Connected Service B.C. – A.I. Technical Solutions Team
Date: November 2025

---

1. Purpose of This Report

This document expands on the “integration and validation” work referenced in the AI Compute Modernization Recommendation Package.

The purpose is to provide a clear, technically grounded summary of:

1. **How the team validated the current compute stack**
2. **Why Xeon 6 P-core systems require specific integration pathways**
3. **Where bottlenecks were identified**
4. **Why high-bandwidth networking (200 Gb/s per NIC) is necessary**
5. **How to integrate real-time inference and heavy AI workloads into government architecture**

This report ensures that hardware investments align with platform realities and that AI workloads receive predictable, sustained performance.

---

2. Overview of the Validation Methodology

To understand the real performance characteristics of AI workloads on the B.C. Gov compute platform, the team developed an **in-house GPT-2 runtime in C**, implementing both inference and full backpropagation. This removes the abstraction layers found in PyTorch or TensorFlow and exposes behaviour at the hardware and OS-scheduling level.

The runtime uses explicit AVX-512 vectorization, hugepage-backed contiguous memory buffers, to minimize memory fragmentation and improve TLB behaviour. Each phase of the transformer block (QK^T, Softmax, V, MLP) is instrumented with microsecond-resolution timers and GFLOPS measurements.

While explicit core pinning was not applied in this iteration, the observed performance patterns; particularly the periodic 50 ms scheduling stalls; strongly indicate that Kubernetes/OpenShift CPU management policies, and not compute limitations, are the primary source of performance degradation.

In addition, when the CPU was driven into memory-bound computation; such as during full backpropagation and various algorithmic validation runs; the timing behaviour strongly suggested L3 cache pressure and hyper-threading (H.T) contention, consistent with the limited L3-per-core ratio of the current Xeon 6244 platform.

This combined approach of low-level kernel benchmarking and full-model execution provides a highly transparent view of how the current Xeon 6244 dual-socket architecture behaves, and establishes a baseline for predicting the performance gains expected from next-generation Xeon 6 hardware.

Note on Additional Workload Validation

In addition to transformer kernel testing, we also validated CPU behaviour using real document-processing workloads relevant to government use cases. This included parsing the full point-in-time Acts and Regulations dataset (~10 GB), implementing a custom high-performance WordPiece tokenizer in C for fast token generation, and generating embeddings using Intel OpenVINO and oneAPI libraries. We also performed parallel-processing experiments using OpenMP and MPI to understand scaling limits for batch workloads. A more detailed analysis of this document-processing pipeline can be provided upon request.

---

3. What the Validation Revealed

While the document-processing tests (Sec. 2) were crucial for validating real-world use cases, the most critical platform bottlenecks were revealed by two primary workloads:

1. A custom GPT-2 inference engine written in C
2. A full GPT-2 training loop (forward + backward pass) implemented from scratch

This allowed us to observe *true hardware and platform behaviour* at a level not possible with high-level frameworks.

Understanding The Workload

To interpret the performance data below, it helps to understand how modern large language models (LLMs) process text:

Tokens: LLMs generate text one word-piece (token) at a time. Generating "The capital of France is Paris" means generating 6-7 tokens sequentially.

Layers: Modern transformer models like GPT-2, Llama, and similar architectures use multiple sequential processing layers (GPT-2 has 12, larger models have 24-96). Each token must pass through all layers before the next token can be generated.

Layer Operations: Each layer performs two main computational blocks:

- **Attention block** (measured below):
  - Phase 1 ($Q \cdot K^T$): Compute attention scores (~3ms, smaller matrix multiply)
  - Phase 2 (Softmax): Normalize attention weights (~2.5ms, element-wise operations)
  - Phase 3 (Softmax·V): Apply attention to values (~2.5ms normally, **largest matrix multiply, most compute-intensive**)
- **MLP block** (feed-forward network): Additional matrix operations (~similar duration to attention)

Why we focus on attention: The attention block has clearly separated phases, making it ideal for identifying exactly where scheduler throttling occurs. Phase 3 is the longest single operation and therefore most likely to be interrupted when CPU quota expires.

3.1 Compute Capability (Baseline Performance)

When the model executed without interruption, the dual-socket Xeon Gold 6244 delivered:

- 8–9 ms per full attention block
- 200–215 GFLOPS sustained
- Stable runtimes across layers
- Strong AVX-512 utilization
- Predictable cache behaviour when phases stayed on-socket

This confirms a critical point: The CPU hardware itself is not the bottleneck for small–medium transformer workloads; it is with platform behaviour.

3.2 Platform Bottleneck (Periodic 50–60ms Stalls)

When running the same workload inside the shared OpenShift worker node, the model exhibited a perfectly periodic performance collapse.

Observed Behaviour

- **Phase 3 (Softmax·V GEMM)** suddenly jumped from ~2.5 ms → **50–60 ms**
- GFLOPS dropped from **~200** → **~28**
- Spikes occurred at fixed intervals (not random)
- **Phase 1 (QK^T)** and **Phase 2 (Softmax)** remained normal

This pattern was visible both numerically and visually (plot of attention time vs tokens).
The graph clearly shows a **sawtooth pattern** of predictable throttling.

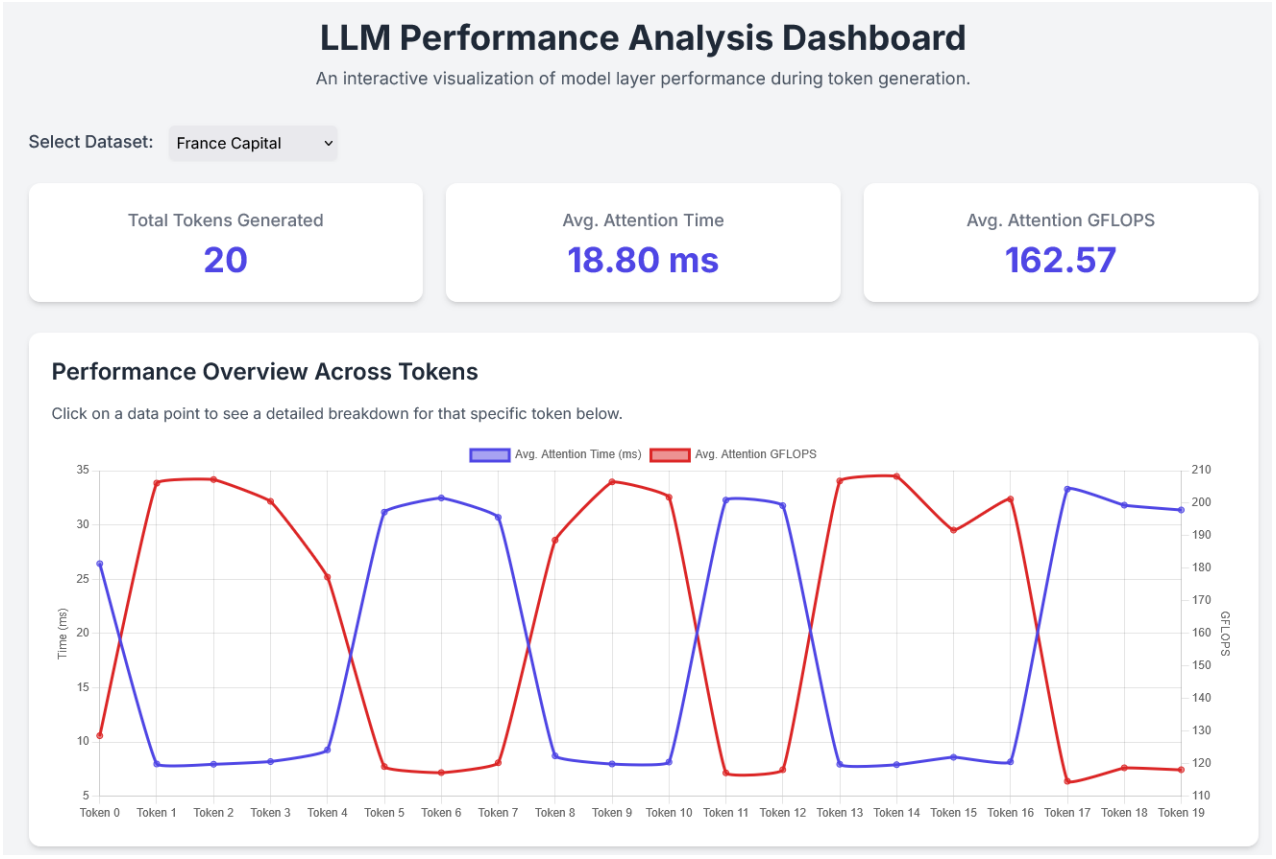

Figure 1 shows per-token attention performance during GPT-2 inference on the OpenShift platform. The graph reveals:

- Blue line (Avg. Attention Time): Periodic spikes from ~8ms (baseline) to ~33ms (throttled)
- Red line (Avg. Attention GFLOPS): Corresponding drops from ~200 GFLOPS to ~120 GFLOPS

The pattern is perfectly periodic and repeats every 3-5 tokens. This regularity rules out random causes (cache misses, thermal throttling, memory pressure) and confirms systematic scheduler interference from Kubernetes CPU quota enforcement.

Key observation: The oscillation between high and low performance is not gradual; it's abrupt and cyclic. This is the signature of container scheduling, not hardware limitations.

Table 1: Phase-Level Timing Analysis - Scheduler Throttling Evidence

Token	Layer	Phase 1 ($Q \cdot K^T$)	Phase 2 (Softmax)	Phase 3 (Softmax·V)	Total	Attention GFLOPS	Status
15	5	3.0 ms	2.6 ms	2.3 ms	7.9 ms	206	✅ Normal

16	1	3.7 ms	2.7 ms	**50.1 ms**	56.5 ms	29	❌ Throttled
16	2	3.4 ms	2.6 ms	2.3 ms	8.3 ms	197	✅ Normal
18	3	3.3 ms	2.8 ms	**54.1 ms**	60.2 ms	27	❌ Throttled

Observation: During throttled operations, Phases 1 and 2 execute normally (2-4 ms each), but Phase 3 absorbs a 50-55 ms stall. Note that during Token 16 generation, Layer 1 was throttled while Layer 2 executed normally; proving that throttling is time-based (CPU quota expiration) rather than layer-specific. The container is being descheduled mid-operation and waiting for CPU quota restoration.

Note on measurements: Figure 1 shows per-token averages across all 12 transformer layers. Table 1 shows individual layer measurements. During throttled tokens, typically 4-5 layers are affected by CPU quota expiration, while the remaining layers execute normally. This produces the ~30-33ms average shown in the graph (vs ~57ms for a fully throttled individual layer). Different runs may show different throttling patterns depending on system load and timing.

Empirical Conclusion: Source of the 50–60 ms Stall

The performance spikes we observed strongly align with known behaviour of Linux's CFS (Completely Fair Scheduler) when CPU quotas are applied. Under Kubernetes/OpenShift, CPU limits translate into CFS quotas that reset on a fixed ~100 ms period. When quota is exhausted, the container is descheduled until the next interval. Short transformer phases (QK^T and Softmax) finish well within the quota window, while the longer Softmax·V GEMM sometimes happens to be executing when quota expires and therefore absorbs the full 50–60 ms pause. This produces the appearance of a layer-level slowdown, but the pattern is consistent with scheduler-based throttling rather than a model or hardware constraint.

How This Relates to AI Workloads on OpenShift

The evidence indicates that these stalls originate from container CPU quota behaviour, not from the transformer kernels themselves. Outside of a quota-controlled environment, the same workload runs consistently at ~200 GFLOPS with no cyclic slowdown. In OpenShift, the presence of CFS throttling interrupts longer matrix operations and reduces throughput to ~30 GFLOPS during the throttled window. Based on these findings, allocating dedicated inference nodes—using Guaranteed QoS, static CPUManager policy, whole-core pinning, and NUMA locality—is likely necessary for predictable real-time AI inference.

A bare-metal validation run, without Kubernetes cgroups or CPU quotas, would provide final confirmation by verifying whether the stall disappears when the scheduler can no longer throttle the workload.

GFLOPS Baseline and Interpretation

A dual-socket Intel Xeon Gold 6244 system has a theoretical single-precision AVX-512 FMA peak of approximately **1.8 TFLOP/s**, based on:

3.6 GHz × 8 cores × 16 Floats × 2 OPs × 2 sockets ≈ 1,843 GFLOP/s

This represents an idealized ceiling assuming one FMA per cycle on all lanes.
In practice, Cascade Lake AVX-512 pipelines have **3–4 cycle effective throughput** due to port pressure, memory access stalls, and dependency chains.
A more realistic sustained peak for real workloads is in the range of **450–600 GFLOP/s**, depending on memory locality and kernel characteristics.

The GFLOPS reported in our benchmarks are **empirical**, computed as:

GFLOPSmeasured = total floating-point operations (e.g., 2MNK) / elapsed seconds × 10^9

GFLOPS measurements were taken on the largest GEMM kernel in the attention block (Softmax·V), which best reflects overall transformer throughput. This kernel dominates total compute cost and cleanly exposes platform-level throttling behavior.

Measured performance:

- **Steady state:** 200–215 GFLOPS (≈33–36% of realistic peak)
- **Throttled:** 27–30 GFLOPS (≈4–5% of realistic peak)

The issue is not that we fall short of theoretical peak (no real workload achieves this), but rather that **platform-level throttling causes a 7× collapse** in performance; from normal transformer efficiency (~200 GFLOPS) to degraded performance (~30 GFLOPS). This collapse is entirely attributable to container-scheduler interference, not hardware limits.

Conclusion: **Sustained AI workloads cannot run reliably on shared OpenShift worker nodes.**

---

4. Additional Backprop Findings (Cache- and Memory-Limited Behavior)

Our backpropagation testing is ongoing, but early results already reveal the key limitation of the current platform.

Even though we tested two different gradient-accumulation strategies
— **feature-major** and **index-major** — both produced nearly identical wall-clock performance. This confirms that:

Backprop on this hardware is dominated by memory bandwidth, not compute.

Backpropagation inherently requires:

- **2× the compute** of the forward pass
- **4× the memory movement** (activations, gradients, intermediate buffers)
- Frequent reads/writes of large tensors at every layer and every epoch
- Repeated refetching of data that does not fit in cache

Because the L3 cache (~50 MB across two sockets) cannot hold the required working set, backprop continuously evicts activations and gradients, forcing the CPU to:

- Re-read from DDR4 due to cache eviction
- Stall on memory rather than compute

The exact accumulation pattern mattered far less than expected because:

> The bottleneck is the platform's DDR4 memory bandwidth and small L3 cache, not the accumulation algorithm.

On 2nd-Gen Xeon, backprop speed is capped by:

- limited L3 capacity
- DDR4 bandwidth
- HT-related contention
- cache thrash from large activation tensors

Why This Supports the Xeon 6 Recommendation

Xeon 6 directly resolves the constraints shown in our empirical tests:

- Massively larger L3 cache
- DDR5/MRDIMM bandwidth approaching 800GB/s – 1.7 TB/s per node
- HT-free AMX cores optimized for BF16/FP16 training
- Dramatically improved memory-compute balance

The additional FLOPS only matter when the memory system can feed the cores; and our validation shows that this is exactly where the current platform fails.The following table summarizes how its key features map directly to the bottlenecks we identified:

Architectural Feature	Improvement vs Gold 6244	Impact
DDR5 + MRDIMM	~5–6× more bandwidth	Eliminates DDR bottlenecks in backprop
Much larger L3 cache	up to ~960 MB across nodes	Reduces eviction of activations/gradients
AMX (Advanced Matrix Extensions)	8–16× GEMM throughput	Makes CPU inference and training viable for medium models

DSA (Data Streaming Accelerator)	Hardware memory copy/zero	Removes memcpy pressure from CPU cores

5. Integration Requirements for Xeon 6 Deployment

The validation work makes clear that *how* Xeon 6 hardware is integrated into the platform is just as important as the hardware itself. The new CPUs are extremely capable; but their benefits are only realized when they operate in an environment that avoids the scheduling, memory, and networking constraints observed during GPT-2 validation.

As a result, Xeon 6 deployment naturally splits into **two integration pathways**, each optimized for a different class of AI workloads.

5.1 Integration Path A — OpenShift Model Serving (Real-Time Inference)

OpenShift is the recommended platform for **real-time, application-facing AI inference**, where requests are short-lived, latency-sensitive, and integrated directly into government services.

Examples of workloads that fit this pattern include:

- **Basic chat-style assistants** for websites or portals
- **Vector embeddings** for search and retrieval
- **Small agentic workflows** tied to specific tasks
- **Classification, PII or scoring APIs**
- **Lightweight document/OCR processing**
- **Simple recommendations or lookups**

These workloads map naturally to OpenShift because they:

- operate through APIs
- require predictable latency
- scale elastically with traffic
  integrate cleanly with routing, identity, RBAC, and existing DevOps processes

Why Tuning Is Required

For real-time inference using the OpenShift AI Operator's model-serving components, Xeon 6 nodes will require targeted testing and tuning by Platform Services to ensure predictable CPU and memory access and to avoid the scheduling-related stalls described earlier.

Teams that instead run AI models inside their own general application pods can expect the default OpenShift scheduling behaviour (including CPU quotas and fair-share policies), and will see similar performance characteristics to those observed in our validation unless their workloads are given dedicated, tuned resources.

Immediate Benefits on OpenShift Without Scheduling Changes

Even without adjusting OpenShift scheduling parameters, Xeon 6 hardware will deliver immediate improvements for real-time inference workloads. Many government use cases rely on small and medium models — such as classification, small agentic A.I, embeddings, PII detection, summarization, and vector search — all of which benefit directly from Xeon 6's architectural strengths:

- significantly higher single-core throughput
- substantially larger L3 cache
- 5–6× greater memory bandwidth via DDR5/MRDIMM
- AMX acceleration for BF16/FP16 compute
- improved sustained clock frequencies under load

These advantages reduce the duration of key transformer operations, making workloads naturally more resilient to CFS CPU quota throttling. Smaller kernels often complete before a throttling window occurs, and memory-bound operations benefit from the higher bandwidth and larger caches regardless of scheduling policy.

In practical terms, upgrading OpenShift model-serving nodes to Xeon 6 improves performance immediately, even prior to platform-level tuning.
Scheduler tuning (Guaranteed QoS, CPUManager static policy, NUMA alignment) can then be layered on to eliminate the periodic stalls entirely and enable predictable performance for latency-sensitive inference workloads.

5.2 Integration Path B — Dedicated HPC/AI Cluster (Training, Batch, Heavy Inference)

Integration Path B — Dedicated HPC/AI Cluster

Some workloads cannot run reliably inside a shared OpenShift worker environment. These include multi-hour training, large document processing, batch inference, embeddings at scale, and any workload that needs continuous 100% CPU use or high-bandwidth communication across nodes.

For these jobs, the key requirement is simple: **direct, exclusive access to the CPUs**.
A dedicated HPC/AI cluster (e.g., Slurm or similar scheduler) provides:

- Guaranteed control over physical CPU cores

- Deterministic NUMA locality and thread affinity
- Large contiguous memory allocations
- High-bandwidth networking for distributed jobs
- The ability to run training/batch workloads without container preemption

This ensures that long-running and high-intensity AI/HPC workloads can run predictably, at full performance, and without interference from other tenants.

5.3 How This Cluster Fits Into the Enterprise

The dedicated HPC/AI cluster does **not** replace OpenShift — it complements it.

- **OpenShift** → real-time inference and application-facing AI (APIs, microservices, citizen-facing workloads)
- **HPC/AI Cluster** → training, batch inference, research workloads, and high-throughput internal processing

Together, these two layers form a modern AI platform that provides:

- **Predictable latency** for public- and application-serving workloads
- **High throughput** for internal analytics, document processing, and training
- **Full hardware utilization** (AVX-512/AMX/DSA) for transformer models
- **A clear separation** between operational workloads and HPC workloads

This architecture allows teams to deploy AI models safely in production while still supporting intensive experimentation, training, and large-scale processing.

Note:
Security hardening, network segmentation, scheduling configuration (Slurm, MPI, etc.), and detailed integration with OpenShift are outside the scope of this validation report. They will require additional platform and infrastructure expertise to implement. However, based on the empirical evidence collected, this two-tier integration model is the most suitable foundation for reliable, scalable CPU-based AI for B.C. Government.

---

6. Handling Distributed Workloads Across Multiple Nodes

Once CPU throttling and NUMA constraints are removed, the next engineering challenge is scaling workloads beyond a single node. Distributed AI and HPC workloads; such as multi-node training, large-scale batch inference, and parallel document processing; depend heavily on **fast, predictable inter-node communication**.

6.1 Why Distributed Workloads Are Sensitive to Network Bandwidth

Inside a modern Xeon 6 server, memory bandwidth is extremely high:

- **~800GB/s – 1.7 TB/s per node** (dual socket, 8 - 12 DDR5 channels)

However, the moment a workload spans beyond one node, tensors, gradients, activations, and batch results must move **across the network instead of DRAM**. This creates a dramatic bandwidth drop:

Memory vs Network Bandwidth (Why This Matters)

To interpret distributed performance correctly, it's important to distinguish between **gigabytes per second (GB/s)** and **gigabits per second (Gb/s)**. Memory bandwidth is always measured in **GB/s**, while networking equipment is almost always labeled in **Gb/s**. Because **1 byte = 8 bits**, this means:

1 GB/s = 8 Gb/s

So a "100 Gb" network link actually provides only **12.5 GB/s** of usable throughput; dramatically lower than in-node DRAM bandwidth.

Component	Bandwidth	Notes
DRAM (Xeon 6)	**800GB/s – 1.7 TB/s**	In-node compute speed
100 Gb Ethernet	**12.5 GB/s**	64 × slower than memory
200 Gb Ethernet	**25 GB/s**	32 times slower, Still a bottleneck, but improved
4 × 200 Gb (per node)	**100 GB/s**	Only 8x slower than memory

Why these ratios matter

A node can compute internally at **800–1,200 GB/s**,
but can only exchange data between nodes at:

- **12.5 GB/s on 100Gb**
- **up to 100 GB/s with 4×200Gb**

This creates two totally different operating regimes:

Network Config	Effective Bandwidth	Slowdown vs DRAM	Distributed Training
100 Gb	12.5 GB/s	**64× slower**	❌ Not viable
2× 200 Gb	50 GB/s	**16× slower**	⚠️ Limited
4× 200 Gb per node	100 GB/s	**8× slower**	✅ Practical

A **64× gap** cannot be hidden by software optimization.Where as a **16× gap** can be mitigated using:

- gradient compression
- communication/computation overlap
- pipeline or tensor parallelism
- optimized AllReduce

An **8 × gap** (with 4 × 200Gb) becomes workable. This is why the infrastructure needs **200 Gb/s per NIC and the ability to populate multiple NICs per node**.

6.2 Requirements for Multi-Node AI / HPC Workloads

Requirement	Why It Matters
High-Bandwidth Interconnects (≥200 Gb/s per node)	Reduces the gap between DRAM speed and inter-node communication; prevents distributed jobs from becoming network-bound.
Low-Latency, Lossless Networking (RDMA)	Enables fast exchange of gradients, activations, and intermediate tensors for training, batch inference, and parallel processing.
Non-Oversubscribed Rack Fabric	Ensures the spine switch can handle the rack's aggregate throughput under load; avoids hidden network bottlenecks.
Deterministic HPC Scheduling (Slurm, MPI, Ray)	Provides predictable execution for distributed and batch workloads; avoids fair-share container jitter.
Exclusive Compute Resources	Prevents interruptions from CPU quotas, noisy neighbours, and container preemption; ensures consistent performance.

Summary

Distributed AI, ML, data, and HPC workloads scale according to how efficiently nodes can communicate. A modern cluster with ≥200 Gb/s inter-node bandwidth, a non-blocking rack fabric, deterministic HPC schedulers, predictable isolated compute resources, and RDMA-capable networking becomes fully application-agnostic and able to support a wide range of government workloads—from real-time or batch document processing to large-scale data transformations, multi-node AI training, parallel analytics, and scientific/HPC tasks. This architecture also provides a future-safe foundation that can easily expand to GPUs, accelerators, and other heterogeneous compute nodes without redesigning the underlying infrastructure.