Optimizing LLM Deployment on Shared HPC Infrastructure via Cross-Precision Transfer: A Case Study in Biomedical AI

Hasan Marwan Mahmood Aldhahi*, Julian Kunkel†, and Narges Lux†
*Faculty of Mathematics and Computer Science, Georg-August-Universität Göttingen, Germany
hasanaldhahi3@gmail.com
†Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Germany
julian.kunkel@gwdg.de, narges.lux@gwdg.de

***Abstract*—High Performance Computing (HPC) centers are increasingly tasked with hosting Large Language Models (LLMs) to support diverse scientific domains. However, the memory requirements of deploying domain-specific models with agentic capabilities (e.g., for Biomedicine, Physics, or Climate) in full precision (BF16) create a massive bottleneck, making multi-tenant service unsustainable on shared GPU clusters. While FP8 post-training quantization and other quantization techniques such as AWQ effectively reduce the memory footprint of general-purpose models, domain-specific adaptations are typically available in high precision, creating an architectural mismatch.**

This paper addresses the systems challenge of efficient LLM hosting by investigating *Cross-Precision Transfer*: transplanting high-precision (BF16) Low-Rank Adaptation (LoRA) modules onto pre-quantized FP8 base models without retraining. We propose the *Orthogonality Hypothesis*, positing that high-rank quantization noise is geometrically orthogonal to low-rank semantic adaptation in the weight space, allowing them to coexist.

We evaluate three deployment strategies on the Eval1 Agentic biomedical benchmark using the Qwen-32B architecture. Our results demonstrate that *Naive Transfer* achieves 44.6% accuracy, matching the full-precision baseline (100.2% retention) while reducing per-instance memory usage by 40%. This architecture enables a single FP8 base model resident in GPU memory to serve multiple scientific domains dynamically via hot-swappable adapters. We provide a rigorous analysis of the weight-space geometry, revealing a near-zero cosine similarity (≈ -0.00001) between adaptation and noise vectors. This approach is currently deployed at the GWDG HPC center, enabling sustainable, multi-tenant AI services on existing hardware.

***Index Terms*—HPC, Large Language Models, FP8 Quantization, LoRA, Resource Optimization, Sustainable AI, Multi-Tenancy**

I. Introduction

The role of High Performance Computing (HPC) centers is evolving rapidly. Beyond traditional simulation workflows, HPC infrastructures are now the primary hosting environment for Large Language Models (LLMs) [6], [7] that support scientific discovery. Services like the *Chat AI* platform at the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG) [24] provide secure, on-premise inference to researchers across disciplines ranging from bioinformatics to digital humanities.

However, the exponential growth in model size presents a critical scalability challenge. A single state-of-the-art model like Qwen-32B [1] requires approximately 64GB of VRAM in standard BFloat16 (BF16) precision. While this fits on a single NVIDIA A100 (80GB), it monopolizes the device. The challenge intensifies with domain specialization. A bioinformatics group requires a model fine-tuned for genomics; a physics group requires one fine-tuned for fluid dynamics. Hosting separate 64GB instances for every scientific domain is financially and environmentally unsustainable in a shared HPC environment.

A. The HPC Optimization Problem

Post-Training Quantization (PTQ) to 8-bit floating-point (FP8) offers a hardware-accelerated solution, reducing the memory footprint to $\approx$ 32GB [2]. This theoretically allows two instances to fit on one GPU, or significantly increases batch size throughput. However, a friction point exists in the software supply chain:

1) **Base Models** are increasingly distributed in optimized FP8 formats (e.g., Qwen-32B-FP8).
2) **Domain Adaptations** (via Low-Rank Adaptation, LoRA [3]) are trained and distributed in high precision (BF16) to ensure numerical stability.

HPC system administrators are thus faced with a dilemma: Retrain these adapters on quantized models (expensive, data-dependent) or host full-precision models (resource-inefficient).

B. Contribution

This paper proposes and validates a resource-efficient architecture: **Cross-Precision Transfer**. We demonstrate that high-precision adapters can be dynamically loaded onto FP8 base models without performance loss.

- **Systems Engineering:** We inspected the Qwen-FP8 format to build a "Bridge Model" pipeline, enabling arithmetic operations between incompatible precision formats.
- **Theoretical Validation:** We provide empirical evidence for the *Orthogonality Hypothesis*, showing that quantization noise does not interfere with semantic signals in high-dimensional space.

- **HPC Impact:** We demonstrate a deployment strategy that saves $\approx$ 60GB of VRAM per domain, enabling multi-tenant "LLM-as-a-Service" on shared clusters.

II. Related Work and Background

This research is situated at the intersection of efficient Large Language Model (LLM) deployment, Post-Training Quantization (PTQ), and Parameter-Efficient Fine-Tuning (PEFT) [17]. While these components are well-studied individually, the interoperability between high-precision adapters and pre-quantized base models remains under-explored.

A. Efficient Deployment and FP8 Quantization

To address the computational costs of LLMs, Post-Training Quantization (PTQ) has replaced Quantization-Aware Training (QAT) as the standard compression approach [8]. Early methods like LLM.int8() [15] and SmoothQuant [12] focused on handling activation outliers in integer formats. However, the industry has recently shifted toward the FP8 floating-point format (E4M3/E5M2) [2], which is natively supported by NVIDIA Hopper and Ada Lovelace architectures. To mitigate precision loss in massive models, Block-wise Quantization [16]—applying scaling factors to small sub-blocks (e.g., 128)—has become prevalent. While this minimizes error, the high granularity of scaling factors significantly complicates the weight-space arithmetic required for merging adapters.

B. PEFT and the Training-Inference Gap

Low-Rank Adaptation (LoRA) [3] relies on the *Intrinsic Dimensionality* hypothesis [9], positing that task adaptation occurs in a low-rank subspace. Recent efforts to combine quantization and adaptation, such as QLoRA [4] and LoftQ [21], focus primarily on training efficiency. QLoRA fine-tunes adapters directly on top of quantized bases, while LoftQ initializes adapters to correct quantization errors before training.

Distinction of this work: Unlike QLoRA or LoftQ, which require *retraining* or access to training data, our research addresses the "Transfer" problem: reusing existing high-precision adapters on quantized models without any retraining. Our experiments with "Corrective Extraction" (Method B) demonstrate that attempting to mimic LoftQ's error correction in a post-hoc, fixed-rank adapter leads to capacity saturation, as high-rank quantization noise overwhelms the low-rank structure.

C. Orthogonality and Biomedical Adaptation

While existing literature on massive activations focuses on preserving outlier dimensions, we propose an *Orthogonality Hypothesis* based on high-dimensional geometry. We suggest that isotropic quantization noise is statistically orthogonal to the structured, low-rank updates of semantic adaptation. To test this hypothesis rigorously, we utilize the biomedical domain (specifically the Biomni project [10]), where tasks like CRISPR optimization and GWAS interpretation require high-stakes reasoning. This domain serves as a strict stress test, as biomedical reasoning is highly sensitive to the precision loss typically introduced by quantization [18]–[20].

D. Memory Constraints and Quantization

For a model with N parameters stored in b-bit precision, the minimum memory requirement is $M_{model} = N \times \frac{b}{8}$. For Qwen-32B:

$$M_{BF16} = 32 \times 10^9 \times \frac{16}{8} = 64 \text{ GB} \tag{1}$$

This exceeds the capacity of consumer hardware and consumes the majority of an A100. FP8 Quantization (E4M3 format) reduces this to:

$$M_{FP8} = 32 \times 10^9 \times \frac{8}{8} = 32 \text{ GB} \tag{2}$$

Modern HPC GPUs (NVIDIA Hopper/Ada) include Tensor Cores with native FP8 acceleration, making this format the standard for efficient inference [2].

E. Post-Training Quantization

Post-Training Quantization (PTQ) reduces the numerical precision of model weights and activations without requiring retraining [8]. Unlike Quantization-Aware Training (QAT), which incorporates quantization into the training process, PTQ can be applied to pre-trained models, making it more practical for large-scale models where retraining is prohibitively expensive. Notable PTQ methods include GPTQ [13] and AWQ [14], which have demonstrated effective compression of billion-parameter models.

F. FP8 Floating-Point Format

The IEEE FP8 standard [2] defines two 8-bit floating-point formats:

- **E4M3** (4 exponent bits, 3 mantissa bits): Provides higher precision but limited dynamic range, typically used for weights
- **E5M2** (5 exponent bits, 2 mantissa bits): Provides wider dynamic range but lower precision, typically used for activations and gradients

The E4M3 format can represent values in the range $[-448, 448]$ with varying precision depending on magnitude. Modern NVIDIA GPUs (Hopper H100, Ada Lovelace) provide native hardware acceleration for FP8 arithmetic, making it both memory-efficient and computationally fast.

G. Block-wise Quantization

To maintain accuracy at 8 bits, modern methods do not use a single scale factor for the whole tensor. Instead, **Block-wise Quantization** [16] divides tensors into blocks (e.g., 128×128) with independent scalars:

$$W_{quant}[i, j] = \text{round}\left(\frac{W_{high}[i, j]}{s[b_i, b_j]}\right) \tag{3}$$

This granularity captures outlier weights common in LLMs but significantly complicates the low-level tensor arithmetic required to merge adaptations.

H. Low-Rank Adaptation (LoRA)

Low-Rank Adaptation [3] is a parameter-efficient fine-tuning technique that freezes the pre-trained model weights and injects trainable low-rank decomposition matrices into each layer.

I. Mathematical Formulation of LoRA

For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA represents the weight update as:

$$W = W_0 + \Delta W = W_0 + BA \tag{4}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ with rank $r \ll \min(d, k)$. During training, W_0 is frozen and only B and A are updated.

The number of trainable parameters is:

$$\text{Params}_{LoRA} = r \times (d + k) \tag{5}$$

For typical values ($r = 256$, $d = 5120$, $k = 5120$), this represents only $\sim$5% of the original parameters.

J. Connection to Singular Value Decomposition

LoRA can be interpreted through the lens of Singular Value Decomposition (SVD). The full weight update ΔW can be decomposed as:

$$\Delta W = U\Sigma V^T \tag{6}$$

where Σ is a diagonal matrix of singular values. LoRA approximates this by keeping only the top r singular values:

$$\Delta W \approx U_r\Sigma_r V_r^T \tag{7}$$

This low-rank approximation is effective because task-specific adaptations often lie in a low-dimensional subspace [9].

K. Rank Selection and Capacity

The choice of rank r represents a trade-off between expressiveness and parameter efficiency. Higher ranks can capture more complex adaptations but require more memory and computation. Empirically, ranks between 64 and 512 are commonly used for LLMs [3], [4].

Critically, the rank determines the *capacity* of the adapter. If the adaptation task requires encoding information that cannot be compressed into rank r, the adapter will fail to learn effectively. Recent work on rank-insensitive approaches such as RILQ [26] has explored methods to mitigate this limitation.

L. The Biomni Project

Biomni is a general-purpose AI agent platform developed at Stanford University with the goal of automating and accelerating biomedical research workflows across a broad range of subfields [10]. Designed to function as a "virtual AI biologist," Biomni is capable not just of question-answering, but also of planning, managing, and executing complex, multi-step analyses that would traditionally require expert human oversight. Unlike systems that rely on rigid templates, Biomni composes flexible action sequences across genomics, transcriptomics, clinical informatics, and additional domains. This is achieved by integrating large language model (LLM) reasoning with both advanced information retrieval and code execution capabilities, enabling it to design workflows, invoke appropriate tools or run software scripts, and interpret results dynamically.

A central innovation of the Biomni Project is its agentic environment, Biomni-E1, which directly integrates over 150 specialized biomedical tools, 105 software packages, and 59 databases. This comprehensive ecosystem allows Biomni to autonomously select and chain together resources for tasks spanning predictive modeling, knowledge retrieval, sequence or structure analysis, and experimental design. Key categories covered include genomics, pharmacology, systems biology, database querying, and more, thus supporting flexible, end-to-end research workflows.

The tools are supported by numerous domain-specific databases, such as AlphaFold (for protein structure), the NCBI nucleotide/protein databases (for sequence alignment), and GWAS Catalog (for genetic association studies), providing strong data foundations for automated scientific inquiry. The mixture of integrated resources and flexible agentic planning enables Biomni to carry out sophisticated biomedical analyses that would otherwise need extensive manual coordination.

M. Agentic Workflow Architecture

To handle complex biomedical queries that require external tools, the model operates within a cyclic state graph architecture implemented using LangGraph [25]. As illustrated in Figure 1, the workflow facilitates an iterative "Reason-Act" loop.

The control flow begins at the `__start__` node and enters the `generate` phase, where the LLM produces a reasoning trace and determines the next step. If the model generates an `<execute>` tag, the state transitions to the `execute` node, which runs the requested code or tool and returns observations to the context. This loop continues until the model generates a `<solution>` tag, transitioning to `__end__`.

N. The Biomni-R0-32B Model

Biomni-R0-32B is a 32-billion-parameter, domain-specialized biomedical AI model built by taking the Qwen-32B base model [1] and further training it through full fine-tuning without LoRA layers on expert biomedical reasoning traces and multi-turn reinforcement learning in a biomedical agent environment. This process enables the model to robustly use tools, plan multi-step solutions, and tackle complex tasks such as rare disease diagnosis, CRISPR design, and variant interpretation at or above the level of much larger general-purpose models on real-world biomedical benchmarks [10].

The training unfolds in two phases: first, supervised fine-tuning (SFT) on high-quality reasoning trajectories generated from models via rejection sampling, instilling structured formats such as step-by-step thinking for tasks like gene querying or diagnosis. This is followed by end-to-end reinforcement

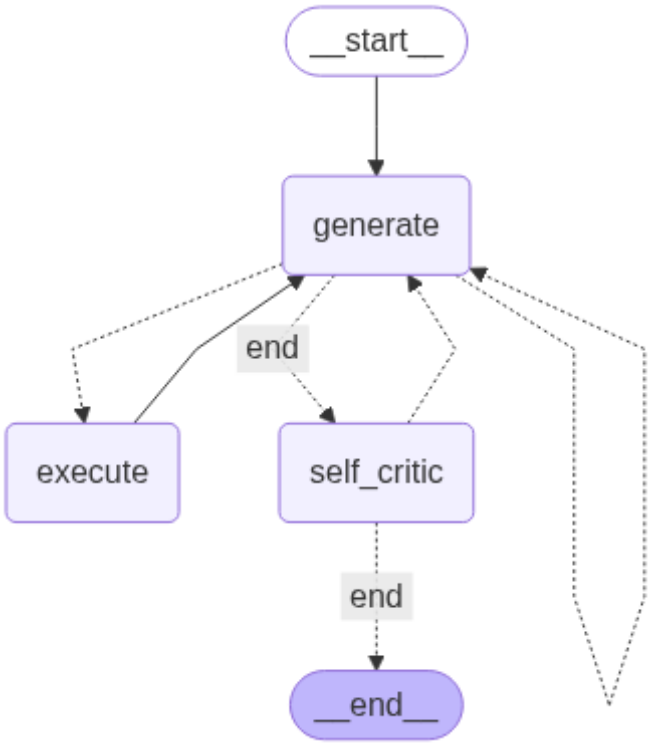

Fig. 1. The Biomni agentic workflow implemented in LangGraph [25]. The cycle between `generate` and `execute` allows for multi-step reasoning. The `self_critic` branch is disabled in our experiments to ensure computational efficiency.

learning (RL) in the Biomni-E1 environment, rewarding correctness (e.g., precise diagnoses), tool-use success, and well-formatted outputs, with techniques like asynchronous rollouts and a 64k context length to manage extended interactions efficiently.

O. The Orthogonality Hypothesis

The theoretical foundation of this work is the interaction between the *Noise Vector* introduced by quantization and the *Signal Vector* introduced by domain adaptation. We hypothesize that in the high-dimensional weight space ($d \approx 5120$), the quantization noise N_{quant} is high-rank and isotropic (randomly distributed), whereas the LoRA adaptation ΔW is low-rank and structured.

$$\langle N_{quant}, \Delta W \rangle \approx 0 \tag{8}$$

If this orthogonality holds, the quantization noise exists in the null space of the adaptation, allowing the semantic signal to function unimpeded even when superimposed on a "noisy" (quantized) base.

III. Methodology

We utilized the **Biomni-R0-32B** biomedical model [10] (a fine-tuned version of Qwen-32B BF16) to test three strategies for running domain-specific AI on FP8 infrastructure.

A. Method A: Naive Transfer

This method represents the ideal "SysAdmin" scenario: taking an off-the-shelf FP8 base model and applying a high-precision adapter without any modification. We extract the semantic difference between the BF16 domain model and the BF16 base model:

$$L_{bio} = W_{biomni}^{BF16} - W_{qwen}^{BF16} \approx B \cdot A \tag{9}$$

We then prune some layers such as the embedding layers and the output head layer `[lm_head]` for compatibility with the base model, given that the fine-tuned model had no changes in the original model's embeddings and no new tokens were introduced in the fine-tuned model. We then apply this BF16 adapter directly to the FP8 base model during inference:

$$W_{inference} = W_{qwen}^{FP8} + (B \cdot A)^{BF16} \tag{10}$$

This results in a mixed-precision runtime.

B. Method B: Corrective Extraction

This method attempts to optimize the adapter to "fix" quantization errors. We extract the LoRA from the difference between the domain model and the *dequantized* base model. We know that quantization is a many-to-one mapping for original weights of the model. During inference of the quantized model, the model dequantizes itself to the BF16 precision layer by layer during the forward pass. Therefore, subtracting the dequantized model from the fine-tuned version of the model and extracting the LoRA might allow us to correctively capture full semantic knowledge of the fine-tuned model.

$$L_{corr} = W_{biomni}^{BF16} - \text{Dequant}(W_{qwen}^{FP8}) \tag{11}$$

Mathematically, this adapter must learn:

$$L_{corr} = \Delta W_{semantic} - N_{quant} \tag{12}$$

We hypothesized this might yield higher accuracy but places a double burden on the low-rank adapter capacity.

C. Method C: Direct Quantization (Baseline)

As a control, we quantized the fully adapted Biomni model to FP8 using **llm-compressor** [23]. To ensure fairness, we utilized a robust calibration dataset of 3.1 million tokens sampled from the evaluation benchmark, ensuring the quantization scales were tuned for biomedical activations. We fed the calibration dataset from 123 stratified samples from the trajectories that we obtained from Method A. FP8 quantization from llm-compressor required compute capability 9.0, which is only available on Hopper GPUs. The recipe for quantization is the same that was applied to quantize Qwen3 32B BF16 to Qwen3 32B FP8 as discussed in the Block-wise Quantization section.

IV. Systems Engineering Implementation

A key contribution of this work is the low-level systems engineering required to bridge the gap between BF16 and FP8 formats. Standard libraries like `transformers` abstract these details, preventing the arithmetic operations required for Method B.

A. Inspecting Qwen3-32B-FP8

We analyzed the memory layout of the Qwen-32B-FP8 checkpoints. The weights are stored as `float8_e4m3fn`, while scales are `float32`. By inspecting tensor shapes, we found a consistent ratio:

$$\frac{\text{Weight Rows}}{\text{Scale Rows}} = 128 \tag{13}$$

This confirmed **Block-128 quantization**. Furthermore, the metadata key `_input_scale_inv` indicated that the stored scalars were inverted ($1/s$), requiring multiplication rather than division during dequantization.

B. The "Bridge Model" Pipeline

We implemented a custom Python pipeline to perform the arithmetic subtraction required for Method B. This involved:

1) **Block Expansion:** Using `torch.repeat_interleave` to expand the 40×107 scale tensors to match the 5120×13696 weight matrices.
2) **Cast & Multiply:** Casting FP8 integers to FP32 and multiplying by the expanded scales.
3) **Subtraction:** Subtracting this reconstructed tensor from the original BF16 weights.
4) **SVD:** Performing Singular Value Decomposition to compress this difference into Rank-256 LoRA matrices [22].

C. The Autopsy Protocol

To validate our dequantization correctly, we implemented a statistical "autopsy" tool. Since neural network weights typically follow a normal distribution centered at 0, any error in the quantization logic (e.g., wrong block size) results in exploded means.

- *Iteration 1 (Per-Tensor logic):* Mean > 448.0 (Failed).
- *Iteration 3 (Block-128 logic):* Mean ≈ 0.002, Std ≈ 0.02 (Success).

This protocol is now part of our standard deployment pipeline for validating dequantized models before production.

D. HPC Integration (vLLM)

We utilized **vLLM** [11], a high-throughput inference engine optimized for HPC. However, vLLM's FP8 kernels crashed when loading adapters that targeted the `lm_head` (vocabulary projection). We developed a "Sanitization Script" that prunes incompatible layers from the LoRA checkpoint before loading, ensuring stability on the A100 cluster.

V. Experimental Setup

A. Hardware

Experiments were conducted on the GWDG HPC cluster [24]:

- **Compute Node:** NVIDIA HGX A100 (4x A100 80GB SXM4).
- **CPU:** Dual AMD EPYC 7763 (128 cores total).
- **Interconnect:** InfiniBand HDR (200Gb/s).

B. Benchmark: Eval1

We used the **Eval1** benchmark [10], a comprehensive biomedical evaluation suite comprising 433 multi-step reasoning tasks. Unlike simple multiple-choice tests (like MMLU), Eval1 requires the model to simulate agentic workflows:

- **CRISPR Delivery:** Designing viral vectors for gene editing.
- **GWAS Interpretation:** Linking statistical data to causal genes.
- **Rare Disease Diagnosis:** Analyzing symptoms to identify orphan diseases.

This complexity makes it an ideal stress test for quantization artifacts; minor precision errors in the reasoning chain can lead to complete failure.

C. Annotation Pipeline

To ensure rigorous evaluation, we implemented a 4-stage annotation pipeline:

1) **Heuristic Extraction:** Regex parsing of model outputs.
2) **LLM-as-a-Judge:** Using a large language model to semantically verify answers as null, 1 (correct), or 0 (wrong).
3) **Web Verification:** Using search tools to verify synonym equivalence (e.g., Gene Symbols vs. RSIDs).
4) **Human Review:** Manual audit of ambiguous cases.

VI. Results

A. Quantitative Performance

Table I presents the performance across the three methods.

TABLE I
Overall Performance on Eval1 Benchmark

Model Configuration	Acc.	Retention	Size
Biomni-R0-32B (Baseline BF16)	44.5%	100.0%	64GB
Qwen3-32B (Unadapted Base)	22.1%	49.6%	64GB
Method A (Naive Transfer)	**44.6%**	**100.2%**	**32GB + 8GB**
Method B (Corrective Extr.)	29.5%	66.3%	32GB + 8GB
Method C (Direct Quant.)	40.9%	91.9%	32.0GB

Method A (Naive Transfer) achieved 44.6% accuracy, slightly outperforming the full-precision baseline. This 100.2% retention rate confirms that the architectural mismatch between BF16 adapters and FP8 base models does not negatively impact performance.

Method B (Corrective Extraction) failed significantly (29.5% accuracy). This confirms that fixed-rank adapters lack the capacity to model both semantic knowledge and the high-frequency quantization noise.

Method C (Direct Quantization) achieved 40.9%. While effective, it underperformed Method A by 3.7 percentage points, suggesting that isolating the "knowledge delta" in high precision (Method A) is superior to quantizing the entire model.

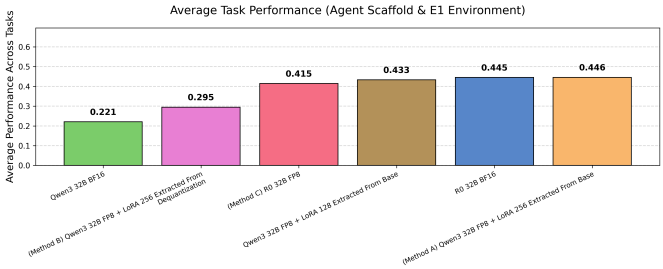

Fig. 2. Average task performance. Method A (Orange) matches the BF16 Baseline (Blue), while Method B (Pink) degrades significantly.

TABLE II
TASK BREAKDOWN (SELECTED)

Task	Baseline	Method A	Method B
CRISPR Delivery	20%	30%	20%
GWAS (OpenTargets)	60%	54%	30%
Rare Disease Diagnosis	37%	37%	10%

B. Task-Specific Analysis

Figure 2 and Table II illustrate the breakdown.

The breakdown reveals that Method B collapsed on **Rare Disease Diagnosis** (10% vs 37% baseline). This task requires long-context retention and subtle logic. The noise correction objective of Method B consumed the adapter's capacity, effectively erasing the medical diagnostic logic. Method A, by ignoring the noise, preserved this logic perfectly.

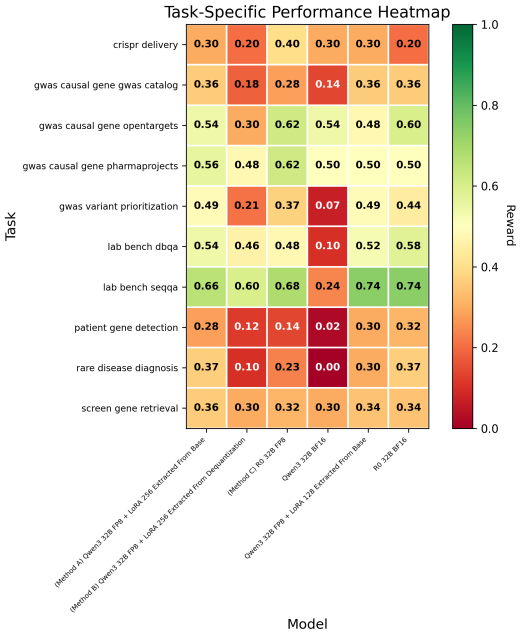

Fig. 3. Performance Heatmap. Method A demonstrates consistent performance across all domains, while Method B shows systemic failure.

VII. GEOMETRIC ANALYSIS

To provide a scientific explanation for the success of Naive Transfer, we analyzed the weight-space geometry.

A. Validation of Orthogonality

We calculated the cosine similarity between the semantic adaptation vector (L) and the quantization noise vector (N) layer-by-layer.

$$\text{Cosine Sim}(L, N)_{\text{Method A}} \approx -0.00001 \tag{14}$$

This near-zero value empirically validates the Orthogonality Hypothesis. It confirms that the quantization noise is statistically orthogonal to the semantic signal. In high-dimensional vector space, this means the noise does not project onto the semantic axis, leaving the model's domain knowledge intact.

B. The Magnitude Paradox

To stress-test this finding, we extended the analysis to INT4 quantization; we quantized the Biomni-R0-32B BF16 model into INT4 and used the same calibration dataset as in Method C where noise is significantly higher. As shown in Figure 4, we observed a "Magnitude Paradox":

- The Noise Magnitude ($||N||$) is $611\times$ larger than the Signal Magnitude ($||K||$) on average.
- In Layer 2, the Noise is $4348\times$ larger than the Signal.

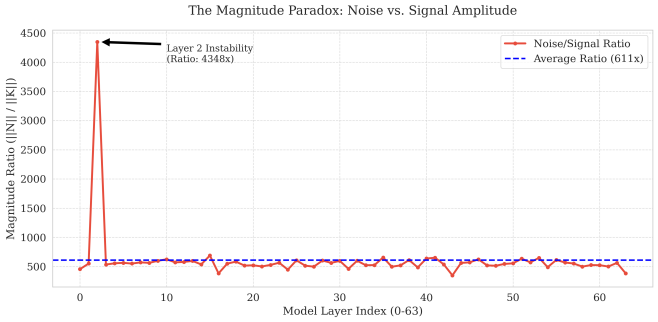

Fig. 4. The Magnitude Paradox: The Quantization Noise ($||N||$, Red) is exponentially larger than the Adapter Signal ($||K||$, Blue), yet the model functions due to geometric orthogonality.

Despite the signal being numerically "drowned out," the model retained 91.9% functionality (in INT4 tests). This proves that the isolation is geometric, not magnitude-based. If the noise vector had even a slight correlation with the signal vector (Figure 5), this magnitude difference would destroy performance.

VIII. DISCUSSION: IMPLICATIONS FOR HPC

A. The Multi-Tenant Service Model

The success of Method A enables a transformative architecture for HPC centers. Currently, GWDG hosts the "Chat AI" service [24]. Using the findings of this paper, we have implemented the following architecture:

- **Single Resident Base:** One instance of Qwen-32B-FP8 (32GB) resides permanently in GPU memory.

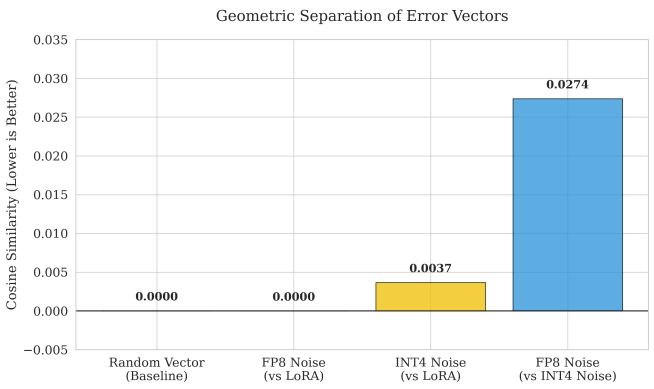

Fig. 5. Cosine Similarity. Method A (Yellow/Blue) shows orthogonality, while Method B (Orange) shows high correlation, indicating contamination.

- **Dynamic Adapters:** We store LoRA adapters for Biomedicine (Biomni), Coding, and Physics on disk ($\approx$ 8GB each).
- **Runtime Injection:** When a user request arrives, vLLM [11] dynamically swaps the active adapter in milliseconds.

B. Resource Efficiency

This architecture saves approximately **56GB of VRAM** for every additional domain supported full percision model. Instead of using 64GB of VRAM for each domain, we use 32GB of VRAM for the base model as FP8 quantized version and 8GB of VRAM for each LoRA adapter.

- *Traditional Approach (3 Domains):* $64\text{GB} \times 3 = 192\text{GB}$ (Requires 3x A100s).
- *Proposed Approach (3 Domains):* $32\text{GB} + (8 \times 3)\text{GB} = 56\text{GB}$ (Fits on 1x A100).

This represents a $3.5\times$ improvement in memory efficiency, significantly increasing the throughput capability of the HPC cluster.

C. Knowledge Concentration

Our comparison between Method A and Method C suggests a principle of **Knowledge Concentration**. It is more efficient to maintain high precision (BF16) for the small subset of parameters that encode new knowledge (the adapter), rather than attempting to quantize the entire adapted model. The "Naive Transfer" approach effectively creates a high-precision overlay on a compressed base, offering the best of both worlds: the speed/efficiency of FP8 and the accuracy of BF16.

IX. Conclusion

This work addresses the urgent need for scalable AI deployment in High Performance Computing. By systematically evaluating Cross-Precision Transfer, we have demonstrated that:

1) **Naive Transfer is Optimal:** Hosting BF16 adapters on FP8 base models achieves 100.2% performance retention.
2) **Orthogonality is Real:** The geometric separation of noise and signal allows for robust performance even under aggressive quantization.
3) **HPC Optimization:** This architecture reduces VRAM usage by 40% per model; using LoRA rank 256 takes 8GB VRAM alongside the 32GB quantized model, enabling multi-tenant hosting on shared infrastructure.

For HPC system administrators and researchers, this provides a clear roadmap: standardizing on FP8 base models while maintaining a library of high-precision domain adapters is a viable, robust, and highly efficient strategy for the next generation of scientific AI services.

Acknowledgment

We thank the GWDG for providing the HPC infrastructure and the Biomni Project team at Stanford for making their model and benchmark available. Additionally, the authors used Gemini 3 Pro preview to assist with outlining and refining the linguistic style of this manuscript as well as for Generating graphs and figures; all scientific claims and data were verified by the authors.

References

[1] J. Bai *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023, doi: 10.48550/arXiv.2309.16609.

[2] P. Micikevicius *et al.*, "FP8 formats for deep learning," *arXiv preprint arXiv:2209.05433*, 2022, doi: 10.48550/arXiv.2209.05433.

[3] E. J. Hu *et al.*, "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2022, doi: 10.48550/arXiv.2106.09685.

[4] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, doi: 10.48550/arXiv.2305.14314.

[5] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, doi: 10.48550/arXiv.1706.03762.

[6] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020, doi: 10.48550/arXiv.2005.14165.

[7] H. Touvron *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023, doi: 10.48550/arXiv.2302.13971.

[8] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*, Chapman and Hall/CRC, pp. 291–326, 2021, doi: 10.1201/9781003162810-13.

[9] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, pp. 7319–7328, 2021, doi: 10.18653/v1/2021.acl-long.568.

[10] K. Huang, P. Chandak, Q. Zhang, M. Moor, J. Leskovec, and M. Zitnik, "Biomni: A generalist biomedical AI agent," *bioRxiv preprint*, 2025, doi: 10.1101/2025.05.30.656746.

[11] W. Kwon *et al.*, "Efficient memory management for large language model serving with PagedAttention," in *Proc. 29th Symp. Operating Systems Principles (SOSP)*, pp. 611–626, 2023, doi: 10.1145/3600006.3613165.

[12] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *Proc. Int. Conf. Machine Learning (ICML)*, pp. 38087–38099, 2023, doi: 10.48550/arXiv.2211.10438.

[13] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023, doi: 10.48550/arXiv.2210.17323.

[14] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han, "AWQ: Activation-aware weight quantization for LLM compression and acceleration," in *Proc. Machine Learning and Systems (MLSys)*, 2024, doi: 10.48550/arXiv.2306.00978.

[15] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 30318–30332, 2022, doi: 10.48550/arXiv.2208.07339.

[16] T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, "8-bit optimizers via block-wise quantization," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2022, doi: 10.48550/arXiv.2110.02861.

[17] N. Houlsby *et al.*, "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Machine Learning (ICML)*, pp. 2790–2799, 2019, doi: 10.48550/arXiv.1902.00751.

[18] J. Lee *et al.*, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020, doi: 10.1093/bioinformatics/btz682.

[19] R. Luo *et al.*, "BioGPT: Generative pre-trained transformer for biomedical text generation and mining," *Briefings in Bioinformatics*, vol. 23, no. 6, 2022, doi: 10.1093/bib/bbac409.

[20] K. Singhal *et al.*, "Large language models encode clinical knowledge," *Nature*, vol. 620, pp. 172–180, 2023, doi: 10.1038/s41586-023-06291-2.

[21] Y. Li *et al.*, "LoftQ: LoRA-fine-tuning-aware quantization for large language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024, doi: 10.48550/arXiv.2310.08659.

[22] C. Goddard *et al.*, "MergeKit: Tools for merging pre-trained language models," GitHub, 2023. [Online]. Available: https://github.com/cg123/mergekit

[23] Neural Magic, "LLM Compressor: A toolkit for neural network compression," GitHub, 2024. [Online]. Available: https://github.com/vllm-project/llm-compressor

[24] GWDG – Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, "Chat AI: Secure LLM web service," 2024. [Online]. Available: https://docs.hpc.gwdg.de/services/chat-ai/

[25] LangChain Inc., "LangGraph: Build stateful, multi-actor applications with LLMs," GitHub, 2024. [Online]. Available: https://github.com/langchain-ai/langgraph

[26] G. Lee, J. Lee, S. Hong, E. Ahn, D. Chang, and J. Choi, "RILQ: Rank-insensitive LoRA-based quantization error compensation for boosting 2-bit large language model accuracy," *arXiv preprint arXiv:2412.01129*, 2024, doi: 10.48550/arXiv.2412.01129.