# Intro

Language Models (LMs) are AI systems processing and generating human language. While Large Language Models (LLMs) like GPT-3 contain billions of parameters, making them powerful but resource-intensive, Small Language Models (SLMs) offer a cost-effective alternative through distillation, maintaining core capabilities with fewer parameters.

Fine-tuning of LLMs allows it to excel in specific domains. This adaptation enhances their understanding of industry-specific terminology and requirements, making them more reliable for specialized tasks like medical diagnosis, legal analysis, or code generation, while preserving their broad language capabilities.

When working with models like GPT, LLaMA, or any other LLMs, understanding how to estimate the required GPU memory is essential. Whether you're dealing with a 7B parameter model or something significantly larger, correctly sizing the hardware to serve these models is critical. Let's dive into the math that will help you estimate the GPU memory needed for deploying these models effectively.

## Computing Requirements by Scale:

Small (1-10B params):

- Memory: 32GB RAM
- GPU: 16 TFLOPS (1x A5000)
- CPU: 16 cores
- Storage: 500GB NVMe

Medium (10-70B params):

- Memory: 128GB RAM
- GPU: 312 TFLOPS (2x A100)
- CPU: 32 cores
- Storage: 2TB NVMe

Large (70B+ params):

- Memory: 256GB+ RAM
- GPU: 1000+ TFLOPS (4+ A100)
- CPU: 64+ cores
- Storage: 4TB+ NVMe

Scaling Strategy:

1. Horizontal: Add nodes
2. Model Parallel: DeepSpeed ZeRO-3
3. Pipeline Parallel: Megatron-LM
4. Data Parallel: PyTorch DDP
5. Cache: Redis/Memcached

Performance targets:

- Latency: <100ms (inference)

- Throughput: 100+ req/s/node
- Availability: 99.9%

## Essential Infrastructure Stack for LLM Deployment:

1. Model Serving Platform

- FastAPI as your main API server - it's fast, modern, and handles async operations well
- Either host models directly or use Triton Inference Server if you need optimized inference

2. Core Database Setup

- PostgreSQL for storing structured data like user requests and model responses
- Redis for quick caching and managing rate limits - critical for high-performance serving

3. Basic Monitoring

- Prometheus to collect key metrics around GPU usage and request patterns
- Grafana to visualize these metrics in simple dashboards
- Focus on tracking: GPU memory usage, inference times, and request volumes

4. Model Management

- MLflow for basic experiment tracking and model versioning
- Keep track of which model versions are deployed and their performance

5. LangChain Integration

- Use basic LangChain components for prompt management and chain orchestration
- Start with simple chains and expand as needed

vLLM

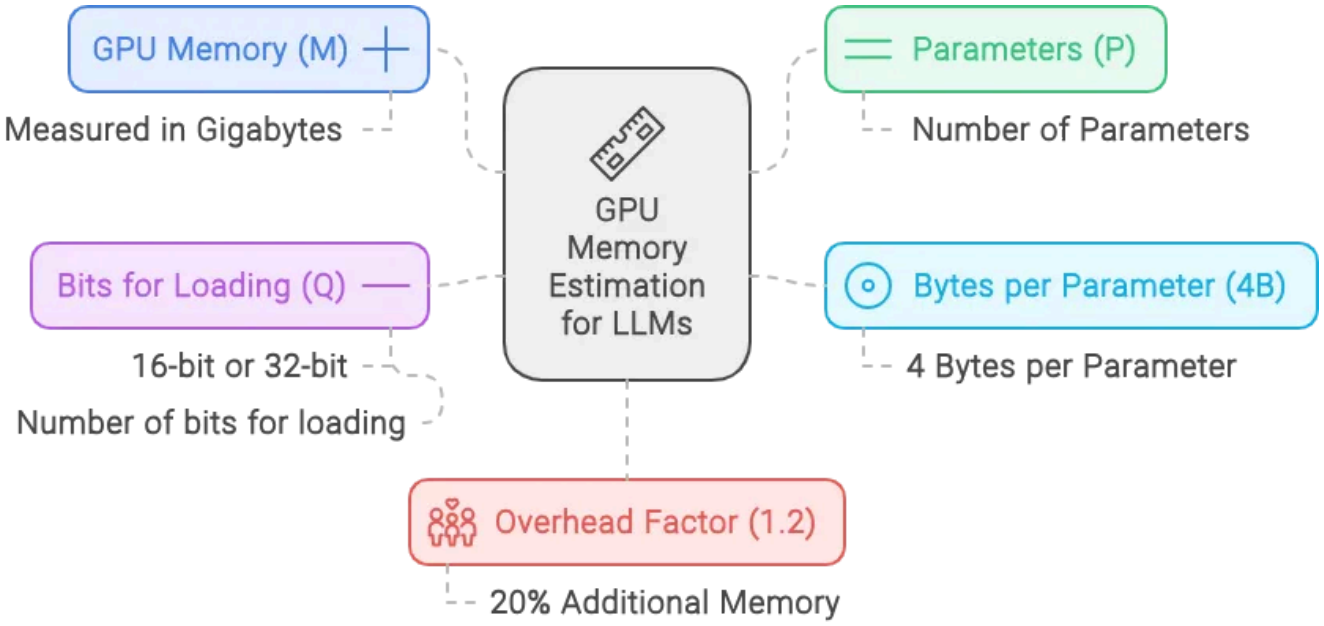
# The Formula to Estimate GPU Memory

To estimate the GPU memory required for serving a Large Language Model, you can use the following formula:

$$M = \left( \frac{P \times 4B}{\frac{32}{Q}} \right) \times 1.2$$

The Formula to Estimate GPU Memory

- **M** is the GPU memory in Gigabytes.
- **P** is the number of parameters in the model.
- **4B** represents the 4 bytes used per parameter.
- **Q** is the number of bits for loading the model (e.g., 16-bit or 32-bit).
- **1.2** accounts for a 20% overhead.




# Example Calculation

Let's consider you want to estimate the memory required to serve a LLaMA model with 70 billion parameters, loaded in 16-bit precision:

$$M = \left( \frac{70 \times 4 \text{ Bytes}}{\frac{32}{16}} \right) \times 1.2$$

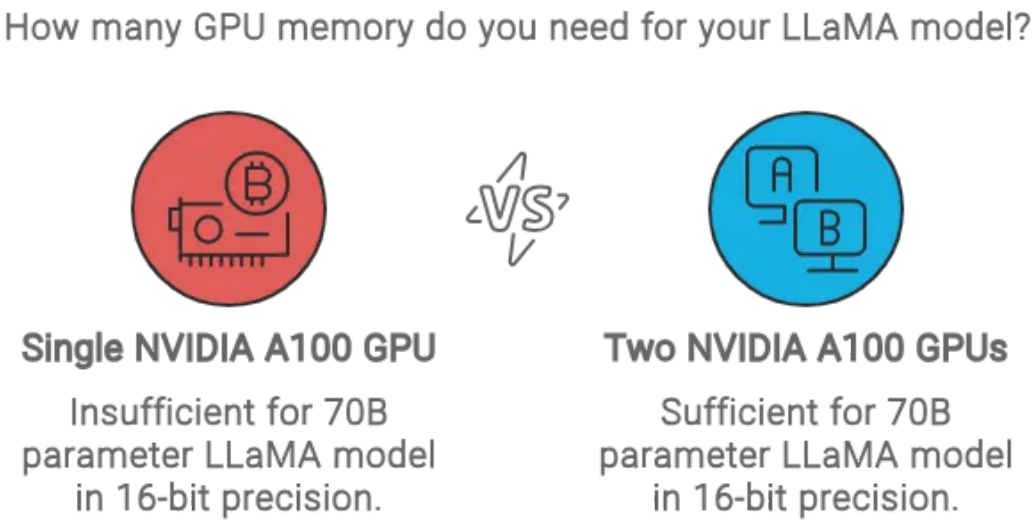
This simplifies to:

$$M = 168GBytes$$

This calculation tells you that you would need approximately **168 GB of GPU memory** to serve the LLaMA model with 70 billion parameters in 16-bit mode.

# Practical Implications

Understanding and applying this formula is not just theoretical; it has real-world implications. For instance, a single NVIDIA A100 GPU with 80 GB of memory wouldn't be sufficient to serve this model. You would need at least two A100 GPUs with 80 GB each to handle the memory load efficiently.




By mastering this calculation, you'll be equipped to answer this essential question in interviews, and more importantly, avoid costly hardware bottlenecks in your deployments. Next time you're sizing up a

deployment, you’ll know exactly how to estimate the GPU memory needed to serve your LLMs effectively.