How Large Language Models Work: A Practical Guide for OCI Engineers

Large Language Models (LLMs) have moved from research curiosity to production infrastructure seemingly overnight. For those of us building and maintaining OCI's AI infrastructure, understanding how these models actually work isn't just academic—it's essential for making smart decisions about GPU allocation, networking architecture, and system design. Let's break down the fundamentals in terms that connect to our daily work.

The Core Mechanism: Predicting the Next Token

At their heart, LLMs do something deceptively simple: they predict the next word (or "token") in a sequence. When you ask Claude or GPT a question, the model doesn't "understand" your query the way humans do. Instead, it calculates probabilities across its entire vocabulary—typically 50,000 to 100,000 tokens—to determine what should come next.

Think of it like autocomplete on steroids. When you type in a search bar and see suggestions, that's a primitive version of next-token prediction. LLMs do this at a vastly larger scale, considering billions of parameters that encode patterns learned from enormous text corpora. The magic happens because these patterns are rich enough to capture grammar, facts, reasoning styles, and even nuanced understanding of context.

Here's a concrete example relevant to our work. If an LLM sees the prompt "The NVL72 rack contains 72 " and is trained on technical documentation, the next token probabilities would heavily favor words like "GPUs," "Blackwell," or "processors." The model has learned these associations from the text it was trained on—not because it "knows" what an NVL72 is, but because it has seen these words co-occur frequently in similar contexts.

Transformers: The Architecture That Changed Everything

The transformer architecture, introduced in 2017, is why we're having this conversation. Before transformers, language models processed text sequentially—one word at a time, left to right. This was slow and made it hard to capture long-range dependencies in text.

Transformers introduced "attention mechanisms" that let the model consider all parts of the input simultaneously. When processing a sentence, a transformer can directly connect a pronoun to the noun it references, even if they're separated by dozens of words. This parallel processing is also why GPUs are so critical to our infrastructure: the matrix multiplications that power attention are embarrassingly parallel and map beautifully to GPU compute.

Consider how this relates to our RDMA networking work. When we optimize NVLink and InfiniBand connections between GPUs in a GB200 cluster, we're directly enabling the distributed matrix operations that make large transformers feasible. A single H100 or B200 GPU might handle billions of floating-point operations per second, but training models with hundreds of billions of parameters requires spreading those operations across hundreds or thousands of GPUs—all communicating their results in near real-time.

Training vs. Inference: Two Different Beasts

Understanding the distinction between training and inference helps explain why different workloads stress our infrastructure differently.

Training is the computationally intense process of teaching a model. The model sees billions of text examples, makes predictions, compares them to actual outcomes, and adjusts its billions of parameters to improve. This requires massive compute—think thousands of GPUs running for weeks or months. The gradient calculations during training demand high memory bandwidth and tight GPU-to-GPU communication, which is why our NVLink topology and RDMA fabric matter so much. A poorly optimized network can turn a training run that should take two weeks into one that takes two months.

Inference is using a trained model to generate outputs. When you chat with an LLM, you're doing inference. It's less compute-intensive per operation than training, but it has different challenges: latency matters enormously (users expect fast responses), and throughput needs to scale with demand. Our inference clusters

need to balance cost efficiency with responsiveness, which drives decisions about GPU selection, batch sizes, and model serving architectures.

Why Scale Matters—And What It Costs

One of the most surprising discoveries in LLM development has been scaling laws: larger models trained on more data consistently perform better, often in predictable ways. This has created an arms race toward ever-larger models and clusters.

For our infrastructure teams, this translates directly into design requirements. When customers are planning zettascale AI workloads, they're not being hyperbolic—the largest training runs genuinely require this kind of compute density. Each doubling of model size roughly doubles the compute requirement for training, and the communication overhead grows even faster because more GPUs need to synchronize more frequently.

This is why the work we do on GPU diagnostics and hardware validation matters. A single faulty GPU in a large training cluster doesn't just slow things down proportionally—it can corrupt gradient updates and ruin days of training progress. The CPV Rules Engine and our diagnostic automation exist because at scale, human-driven troubleshooting simply can't keep pace with the failure modes that emerge.

Bringing It Home

Understanding LLMs at this level helps us make better infrastructure decisions. When we debate whether to prioritize memory bandwidth versus raw compute in the next GPU architecture, we're really asking: will future models be more limited by parameter size or by attention computation? When we design our networking fabric, we're betting on how distributed the next generation of training will be.

The models themselves will continue to evolve—new architectures, new training techniques, new applications. But the fundamental pattern of massive parallel computation on specialized hardware, connected by high-speed fabrics, is here to stay. And that means the infrastructure we're building at OCI isn't just keeping pace with AI—it's enabling what comes next.