

Tópico Especial: Arquitetura de Sistemas para Inferência de LLMs Locais

Modelo Pedagógico: Aprendizado Baseado em Problemas (PBL)

Pré-requisitos desejáveis: Sistemas Distribuídos ou Arquitetura de Computadores (nível de graduação).

Objetivo Geral

Capacitar os alunos a projetar, dimensionar, otimizar e orquestrar ambientes híbridos (hardware e software) para a execução de modelos de linguagem de grande porte (LLMs) locais e sistemas multi-agentes, resolvendo problemas reais de restrição de memória, latência, concorrência e custo de infraestrutura.

Ciclo de Problemas

O curso é dividido em **4 Grandes Problemas**. Cada equipe de alunos receberá um cenário de falha ou uma meta de infraestrutura que precisa ser alcançada com os recursos disponíveis.

Problema 1: "O Modelo não Cabe na Memória"

- **O Cenário:** Uma empresa comprou um servidor com uma GPU de 12GB de VRAM e precisa rodar localmente o modelo *Qwen 2.5 de 14B ou 32B parâmetros* para um pipeline interno. No estado padrão (FP16), o modelo sofre *Out of Memory* (OOM) instantaneamente.
- **O Desafio da Equipe:** Fazer o modelo rodar no hardware estipulado sem degradar drasticamente a qualidade das respostas.
- **Conceitos Computacionais Emergentistas:**
 - Cálculo matemático de pegada de memória (VRAM) por bilhão de parâmetros.
 - Técnicas de quantização de baixo nível (GGUF, EXL2, AWQ) de 8-bit, 4-bit e menos.
 - Métricas de degradação de modelo (Perplexidade vs. Compressão).

Problema 2: "A Latência Inaceitável e o Gargalo de I/O"

- **O Cenário:** O modelo agora cabe na VRAM (ou na memória RAM do sistema), mas o tempo para começar a responder (*Time to First Token*) é de 15 segundos, e a velocidade de geração é de apenas 2 tokens por segundo. O usuário final está abandonando o sistema.
- **O Desafio da Equipe:** Identificar onde está o gargalo físico do sistema (CPU, barramento PCIe, velocidade da memória ou paginação) e reconfigurar o runtime para atingir uma meta de tokens/segundo aceitável para produção.
- **Conceitos Computacionais Emergentistas:**
 - Gargalos de *Memory-Bound* vs. *Compute-Bound* em inferência de LLMs.
 - Arquitetura de runtimes de alta performance (llama.cpp, vLLM, Ollama).
 - Otimização de Contexto (KV Cache) e paginação de memória em runtimes de IA.

Problema 3: "O Ataque dos Agentes Concorrentes"

- **O Cenário:** A equipe implementou um sistema multi-agente. Quando todos os agentes disparam requisições simultâneas para ler e resumir documentos, o runtime local trava, corrompe o estado ou entra em colapso por falta de fila.
- **O Desafio da Equipe:** Criar uma arquitetura de software intermediária (Middleware) que gerencie a concorrência e a distribuição de carga no hardware local.
- **Conceitos Computacionais Emergentistas:**
 - Filas de mensagens e buffers concorrentes (Redis, RabbitMQ) aplicados a runtimes de IA.
 - Structured Outputs (forçar o modelo local a responder estritamente em JSON válido via *Outlines/Instructor* para não quebrar o código dos agentes).
 - Estratégias de *Continuous Batching* e paralelismo de requisições.

Problema 4: "Construindo o Cluster Híbrido e Distribuído"

- **O Cenário Final:** O modelo de 70B parâmetros é necessário para uma tarefa crítica, mas nenhuma máquina individual do laboratório (ou dos alunos) possui VRAM suficiente para ele, mesmo quantizado.
- **O Desafio da Equipe:** Unir os recursos computacionais heterogêneos das máquinas da equipe (ou do laboratório) através da rede local para rodar o modelo de forma distribuída.
- **Conceitos Computacionais Emergentistas:**
 - Paralelismo de Tensores (*Tensor Parallelism*) e Paralelismo de Pipeline (*Pipeline Parallelism*).
 - Latência de rede local vs. latência de interconexão de chips (NVLink vs. PCIe vs. Gigabit Ethernet).
 - Arquitetura de sistemas distribuídos aplicados ao particionamento de grafos de redes neurais.

Dinâmica de Aulas e Avaliação (Metodologia PBL)

1. **Apresentação do Problema (Início do Ciclo):** O professor apresenta o cenário técnico e as restrições de hardware/software.
2. **Fase de Brainstorming e Pesquisa:** As equipes identificam o que já sabem e o que *precisam aprender* para resolver o problema. O professor atua como um facilitador/consultor, indicando artigos.
3. **Desenvolvimento do Protótipo:** Os alunos colocam a mão na massa nas próprias máquinas ou servidores da UFF para implementar a solução técnica.
4. **A Entrega e a Defesa Técnica:** Em vez de uma prova, cada ciclo termina com um "Show & Tell" e uma sabatina:
 - A equipe precisa rodar o sistema ao vivo.
 - Eles devem apresentar os dados empíricos (gráficos de consumo de VRAM, latência, tokens/segundo).
 - Devem defender as decisões de design de software e arquitetura escolhidas.

Nota Final

A composição da nota será baseada no portfólio de soluções dos 4 problemas (60%) + um Artigo Técnico Final/Relatório no formato de conferência (SBC/ACM), documentando a arquitetura do ecossistema local construído pela equipe ao longo do semestre (40%).