Systems Meets AI

COS 316: Principles of Computer System Design
Lecture 17

Nicolaas Kaashoek

AI is a Big Deal

Recent advances have seen massive growth in AI space:

- ChatGPT
- Google/Microsoft/Amazon/etc all building around AI
- Non-tech companies trying to use AI
- $ Billions, Millions of users

Built on Large Language Models (LLMs)

The Bitter Lesson

Rich Sutton, 2019

"The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin"

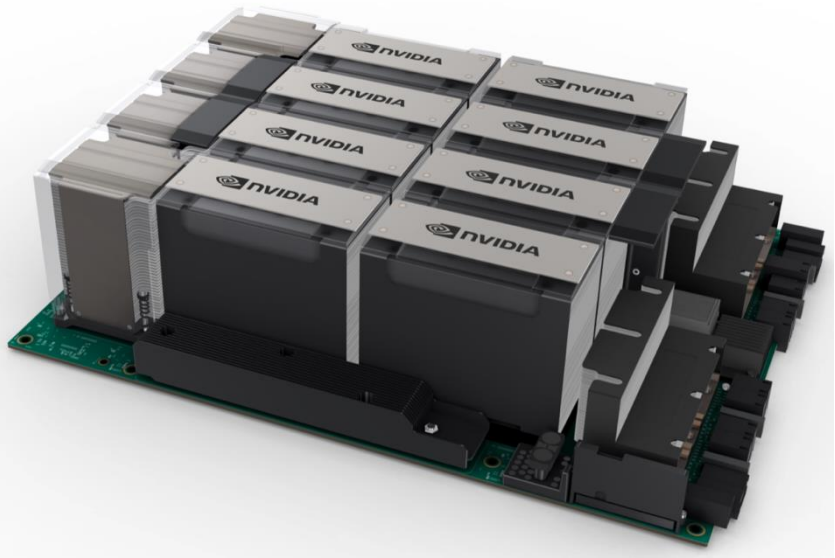
- DeepBlue (1996)
- AlphaGo (2015)
- ChatGPT (2022)!

It’s “all” big data + matrix multiplication + parallelism

GPUs Power Modern Machine Learning

GPUs are the foundation of all modern machine learning

Consumer GPUs: Primarily used for graphics



Datacenter GPUs: Primarily used for ML

Key feature: GPUs accelerate linear algebra

GPUs and Matrices

Turns out lots of things are matrix-based!

Examples:

- Rotating an image is a transform on a matrix
- Applying shaders in a video game is matrix multiplication
- Most machine learning boils down to repeated matrix multiplication

GPU Parallelism

Unlike CPUs, which have a few, beefy cores, GPUs have many, many wimpy ones

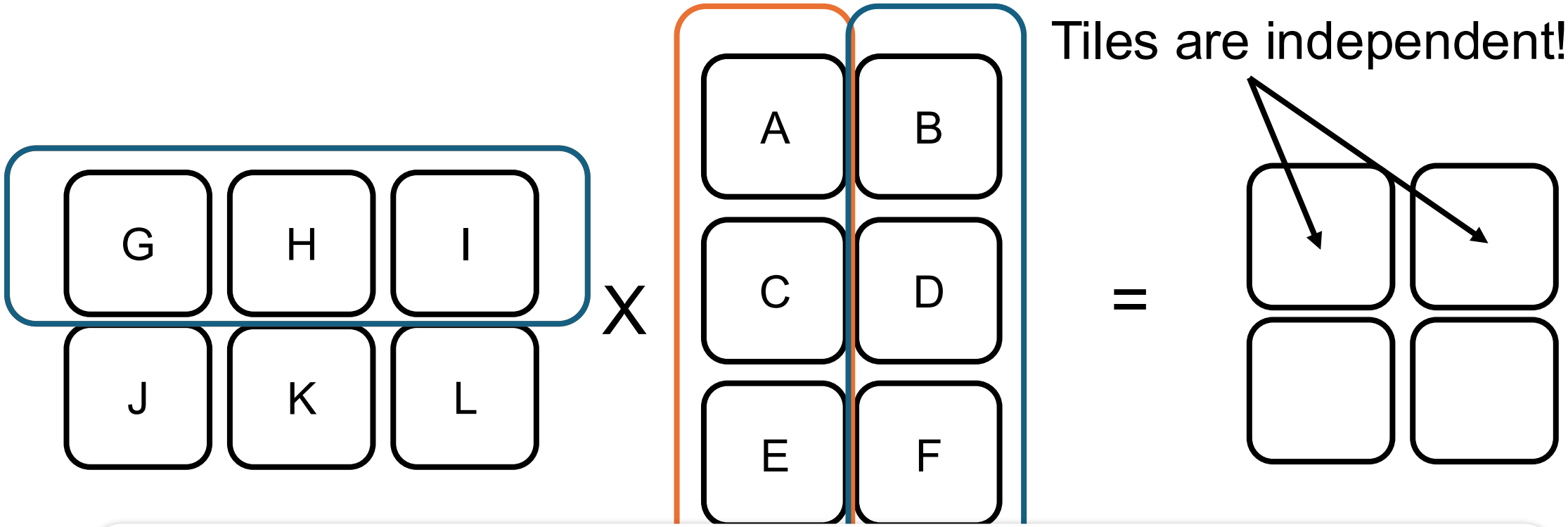
- 8/16/32/etc vs. 1000s
- H100: 18,432 CUDA/640 Tensor cores!

Each of the GPU cores can act independently

- Programming to take advantage of this is not trivial

Concrete example: See demo

GPU Parallelism: Independent Ops



GPUs are awesome when you can split jobs into independent *tiles!*

General Machine Learning Pipeline

Low-level: lots of matrix multiplication

High-level: Create a *model* that learns a task.

- Models are effectively doing very complex pattern matching
- Two parts: Training and inference

Everything is Matrix Multiplication

Both training and inference consist of mostly operations on vectors and matrices

Training: Repeated matrix multiplication to learn *weights*

- Learning the pattern

Inference: Passing input data through a pipeline of matrix transformations and queries to get output

- Matching the pattern

Task: Prediction

LLMs are prediction models

- Task: Predict next item based on the input

Example: “Today is Tuesday, tomorrow will be ___”

ChatGPT predicts the right answer based on the question/prompt

Prediction Challenges

Models find and encode patterns in training data, need some way to represent this

Internally a high dimensional space, determined by two things

- Vocabulary (keys)
- Features (values)

If space gets too big, then the model becomes unusable!

If space is too small, then the vocabulary is not expressive enough, or there’s not enough meaning

Tokenization

How can LLMs encode the whole language without state space exploding?

- Words would result in massive state space
- Characters are missing meaning /context

Solution: Tokens

Tokenization 2

the quick brown fox jumps over the lazy dog

01110100 01101000 01100101 00100000
01110001 01110101 01101001 01100011
01101011 00100000 01100010 01110010
01101111 01110111 01101110 00100000
01100110 01101111 01111000 00100000
01101010 01110101 01101101 01110000
01110011 00100000 01101111 01110110
01100101 01110010 00100000 01110100
01101000 01100101 00100000 01101100
01100001 01111010 01111001 00100000
01100100 01101111 01100111

Tokenization establishes a common vocabulary without exploding in size

290, 29082, 6446

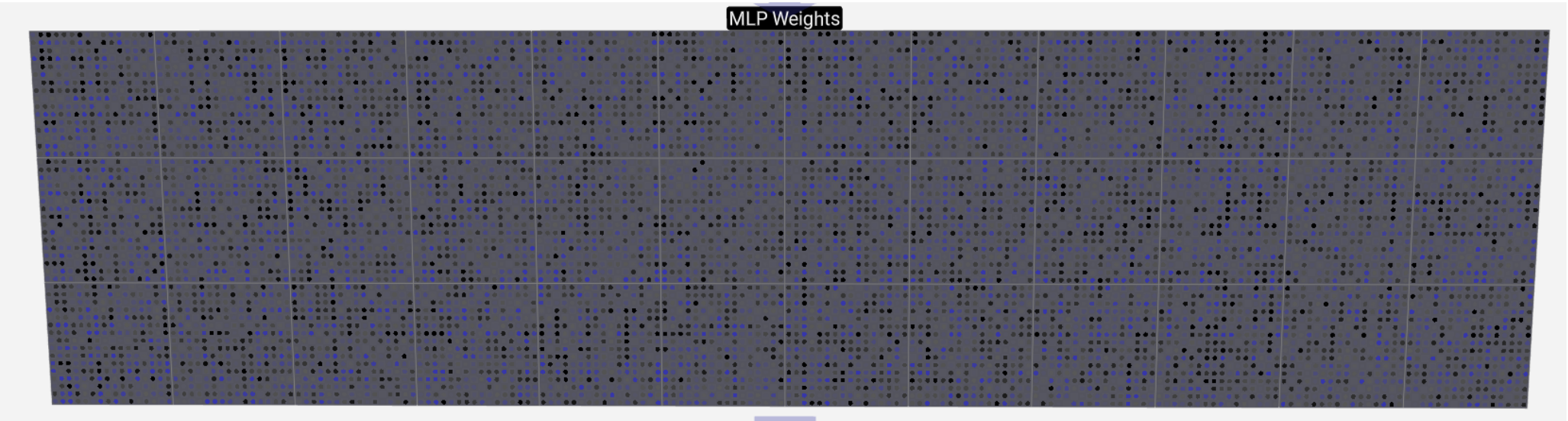
Tokens Become Vectors

Tokens are our keys in the internals of the model

Our values are *parameters*

- Each token maps to a vector, each element in the vector is a parameter

Modern LLMs: Multiple models each with 10^{11} parameters



Vectors: Space = Semantics

What do the parameters mean?

- Random values at start of training, learned over time

Parameters assign point in space. Proximity to other points captures the pattern in the training data

Meaning manifests spatially, and enables vector math:

- King – Man + Woman ≈ Queen

Output is Probabilistic/Non-det.

Models are inherently probabilistic

- King – Man + Woman is *close to* Queen
- Probably also close to Princess, Empress, etc.

The output is selected based on a probability

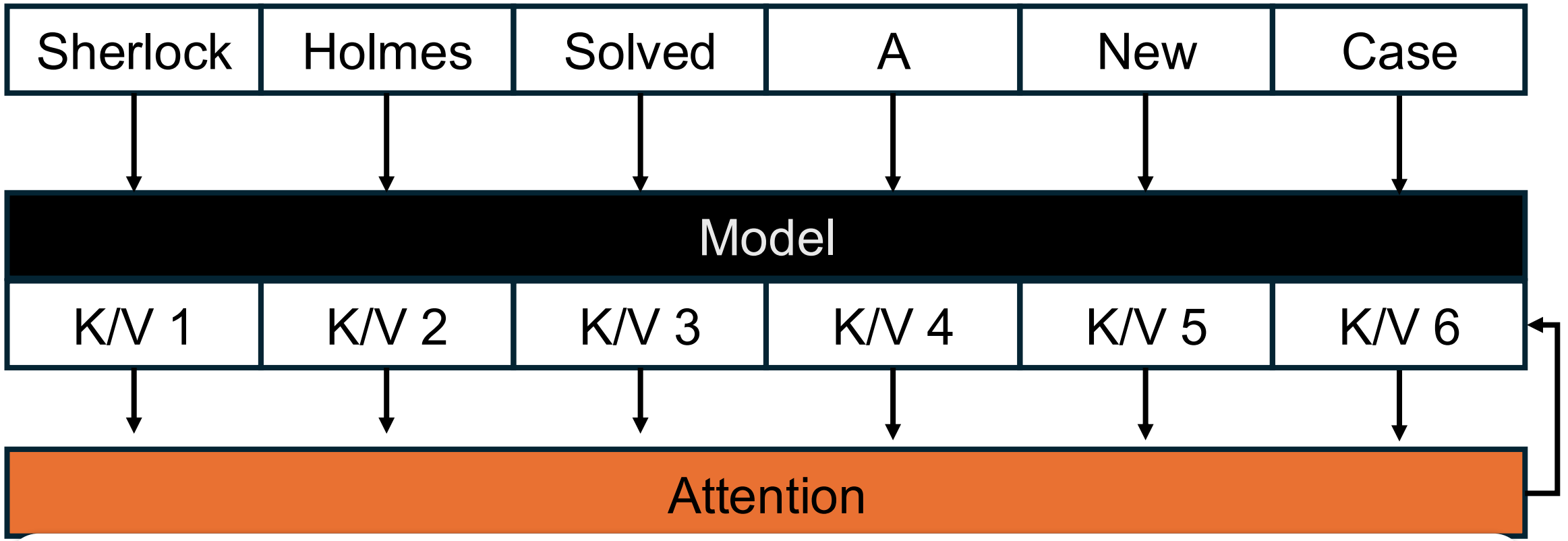
This process is *non-deterministic*: asking ChatGPT the same thing twice will get different results

How does the Model Learn?

Training Data

9111	1234	9081
9111	1234	8291
9081	8291	1234
9111	8291	1234
1234	1234	9081

LLMs Are Highly Parallel



Prior models were *recursive*. Transformer parallelism reinforces the Bitter Lesson: Allows them to process more data!

More Data = Better Results

How much data?

- As much data as can possibly be fed into the model

Estimates:

- Total training data: Petabytes (Internet++)
- Training time: Months
- Training cost: $ Millions
- Final model parameter count: Multiple 100s of billions

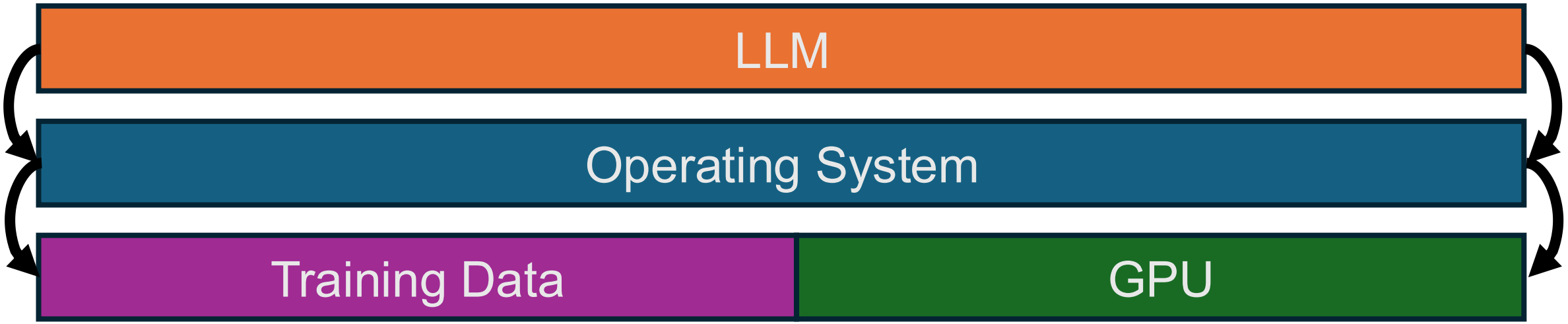
LLMs are a New Application

LLMs and Machine Learning at scale are totally new apps

New applications create new systems problems:
- Computer graphics accelerated by GPUs
- Smartphones: Battery power, touchscreens
- Cloud computing: Storage, Sharding, RPC, Fault Tolerance

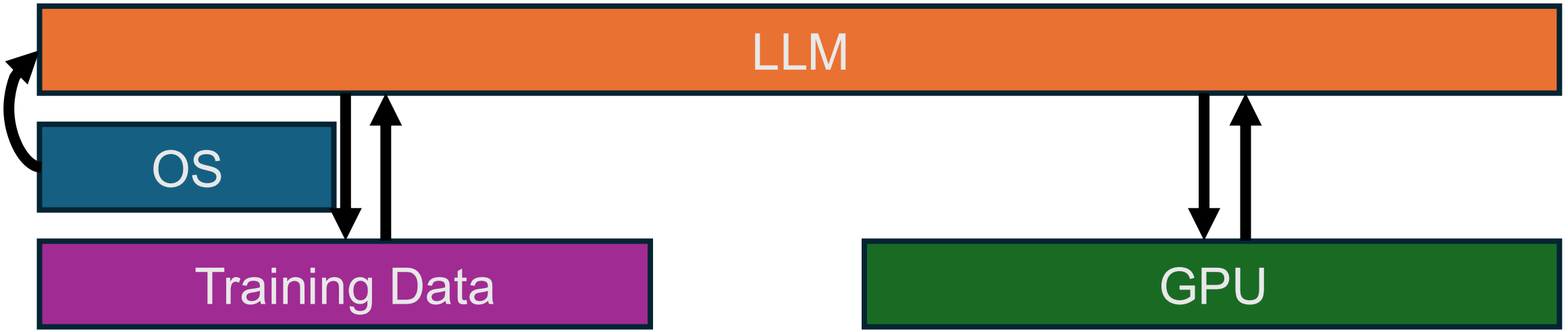
LLMs are no different

Bypassing the OS for Speed



Having to go through the OS every time is inefficient

Bypassing the OS for Speed



Better to use the OS as a control plane, and keep data plane separate for speed

New Networks to Move Data Quickly

During the training process, huge amounts of data moved

- Not just training data: intermediate steps as well

Petabytes of data being moved *often*!

Workload closer to High-Performance computing than cloud:

- Driven by single provider, no need for TCP-style congestion control

Result: Infiniband

Scalable Storage for Training Data

Need distributed storage systems to manage and move all the training data

Existing systems aren’t fast enough

Example: DeepSeek Fire-Flyer File System (3FS) - https://github.com/deepseek-ai/3fs

LLMs Create New Security Problems

LLMs also introduce new security concerns

Access control: LLMs are trained on massive amount of data, some sensitive. Users shouldn’t have access to it!

- Example: Backing out training data
- Example: Exposing API keys

Solution: Guardrails

LLMs Improve Systems Security

LLMs ingest and analyze data at a rate that’s never existed before, enabling them to find subtle and complex bugs in existing code

Recent news: Claude Mythos Preview

Anthropic reports 1000s of existing security flaws in the Linux kernel and other widely used OSS

LLMs Make it Easier to Build Systems

Systems are complex and hard to implement

Coding agents, at the very least, streamline this process

Developers can build entire systems, end-to-end, quickly

LLMs Accelerate Research

Barbarians at the Gate: Paper from UC Berkeley

Key result: Give a coding agent a simulator and evaluation tool, let it iterate on a problem ad nauseum and it’ll produce interesting results

Concrete example: Congestion control algorithms

Conclusion

The Bitter Lesson

Modern machine learning tasks are dominated by matrix multiplication and vector operations, GPUs rock at these

Transformer models further exploit parallelism to ingest data at super high scales

LLMs are a new application, and new applications need new systems