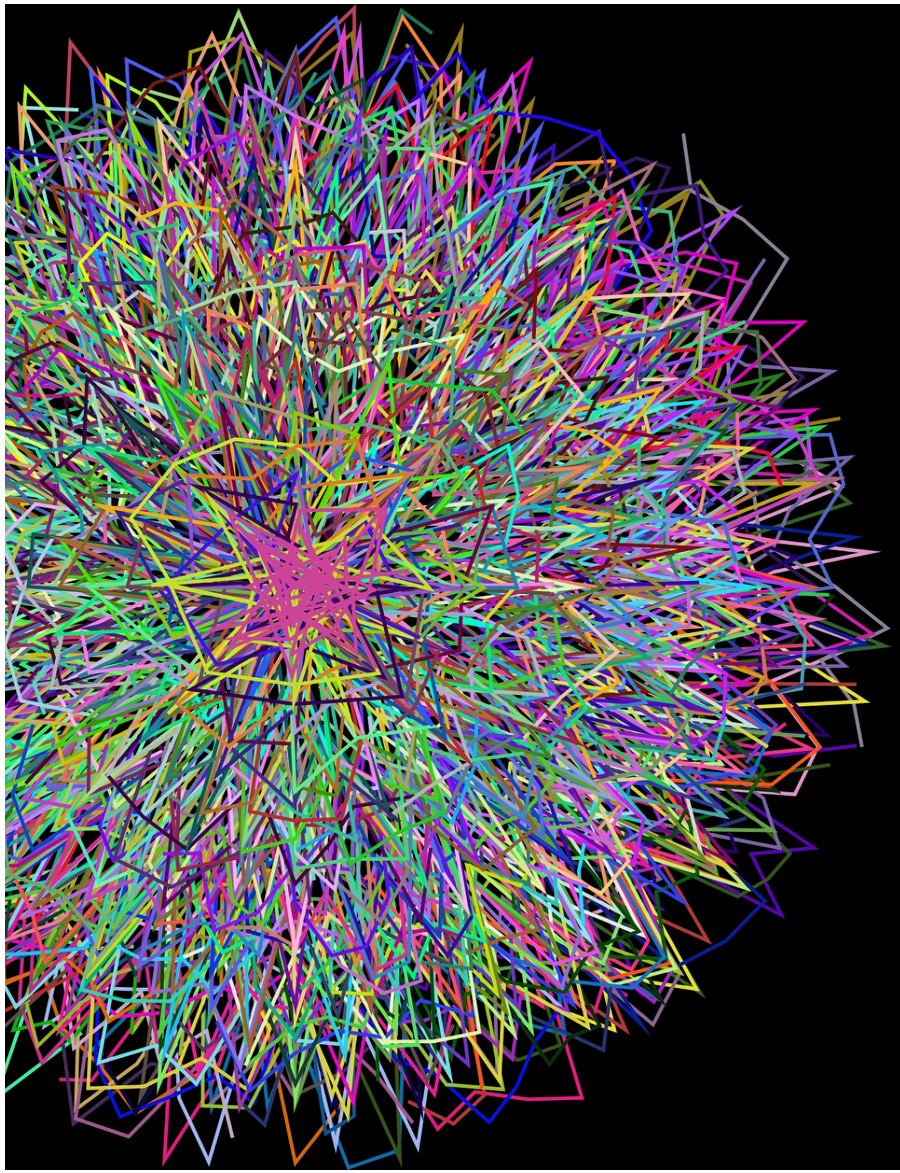


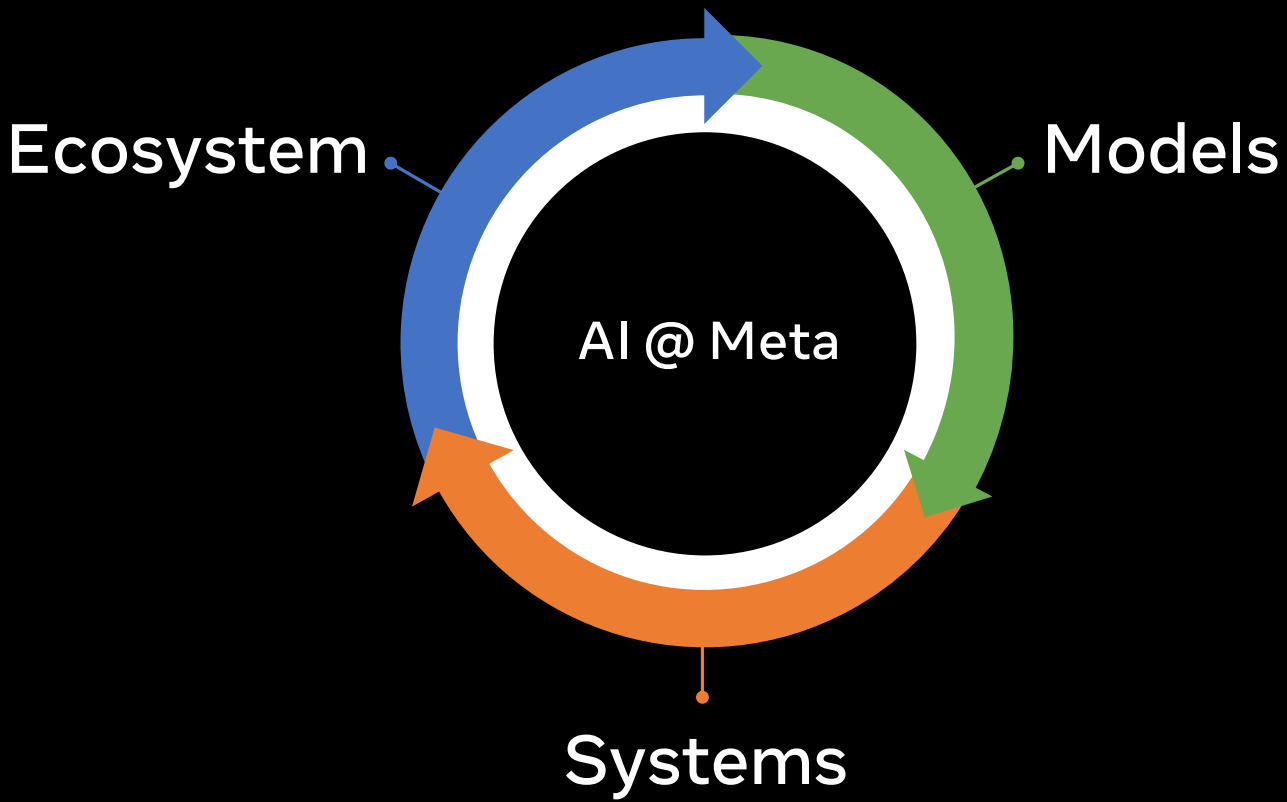
Leading with open

Models, frameworks, and systems

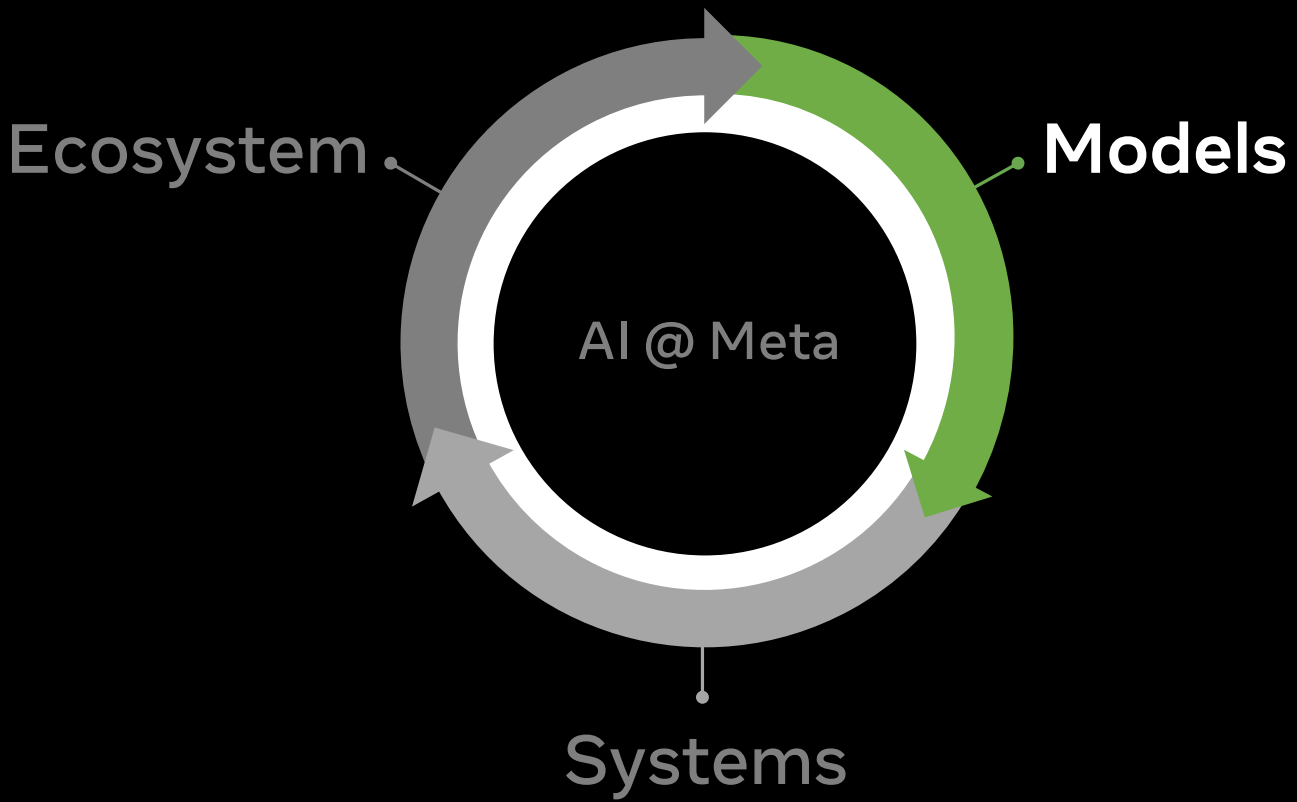
Alexis Black Björlin
VP, Infrastructure

Petr Lapukhov
Software Engineer

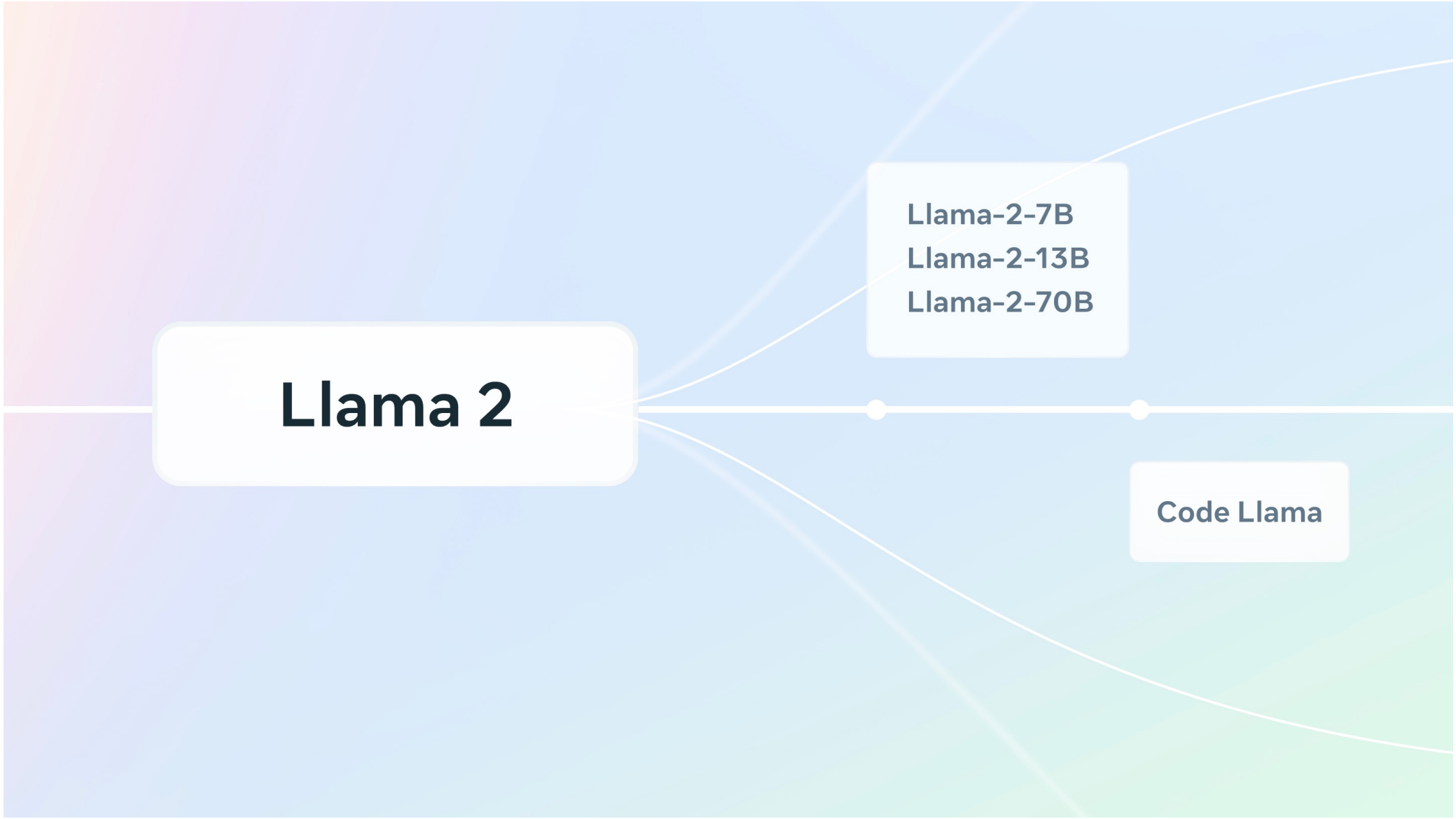
Meta

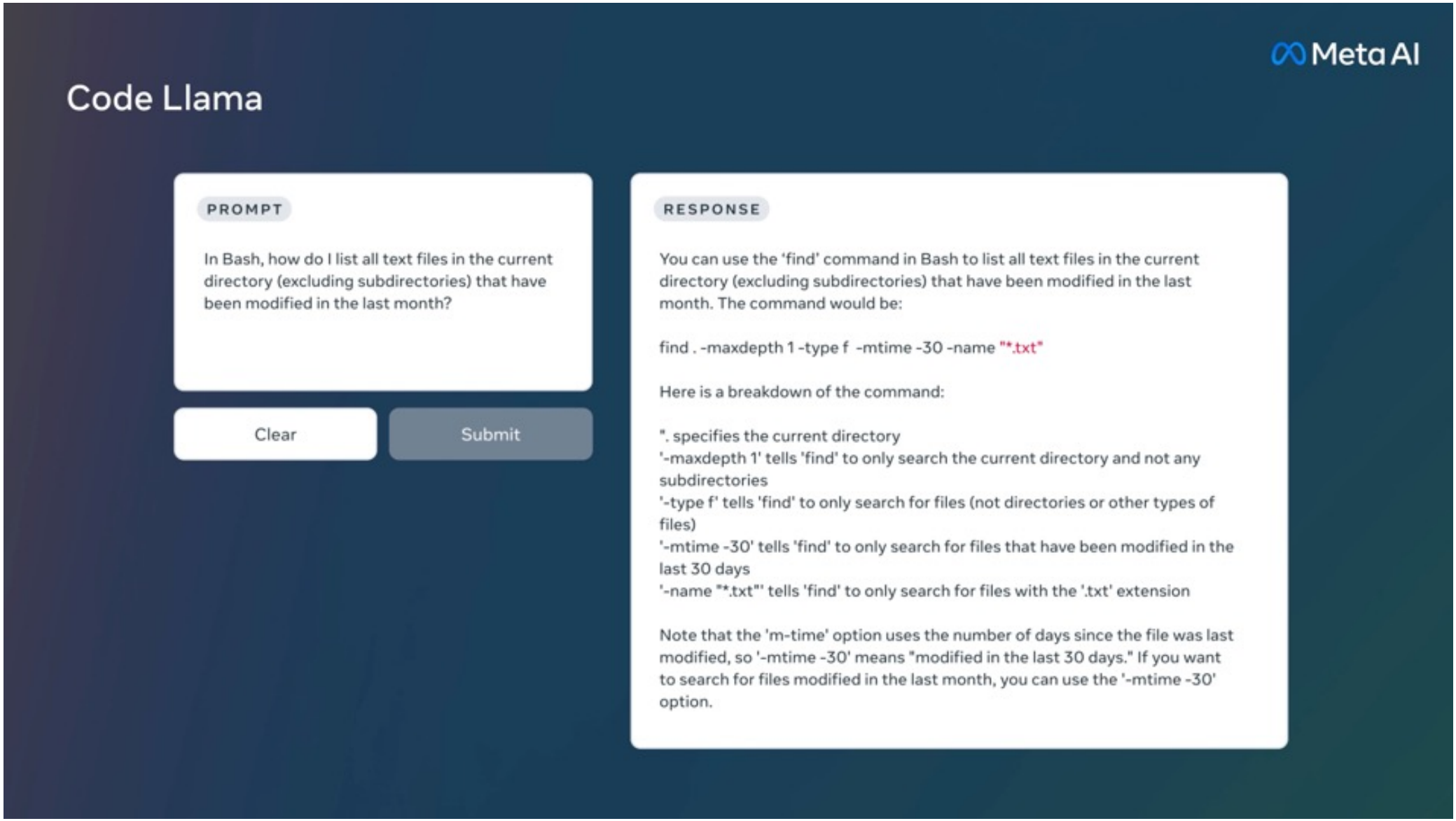


Meta



Meta


Llama 2
Llama-2-7B
Llama-2-13B
Llama-2-70B
Code Llama


Meta AI
Code Llama
PROMPT
In Bash, how do I list all text files in the current directory (excluding subdirectories) that have been modified in the last month?
Clear
Submit
RESPONSE
You can use the 'find' command in Bash to list all text files in the current directory (excluding subdirectories) that have been modified in the last month. The command would be:
find . -maxdepth 1 -type f -mtime -30 -name "*.txt"
Here is a breakdown of the command:
". specifies the current directory
'-maxdepth 1' tells 'find' to only search the current directory and not any subdirectories
'-type f' tells 'find' to only search for files (not directories or other types of files)
'-mtime -30' tells 'find' to only search for files that have been modified in the last 30 days
'-name "*.txt"' tells 'find' to only search for files with the '.txt' extension
Note that the 'm-time' option uses the number of days since the file was last modified, so '-mtime -30' means "modified in the last 30 days." If you want to search for files modified in the last month, you can use the '-mtime -30' option.

- Handles complex coding tasks
- Trained on 16K tokens
- Supports 100K input context tokens
- ***Code Llama – Python,*** a language-specialized variation of Code Llama, further fine-tuned on 100B tokens of Python code

Code Llama

Meta AI

PROMPT

I have a csv file with those headers: Model type, Model size, Checkpoint path, Python, C++, Java, PHP, TS, C#, Bash, Average. Write a code that reads the csv file and plot a nice seaborn visualization of the correlations between the Python, C++, Java, PHP, TS, C#, and Bash for the 7B model only. Please mask the upper triangle of the correlation heatmap

Clear | Submit

RESPONSE

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Read the CSV file
df = pd.read_csv('model_correlations.csv')

# Filter the data to include on the 7B model
df = df[df['Model size'] == '7B']

# Compute the correlations between Python, C++, Java, PHP, TS, C#, and
Bash
correlations = dff[['Python', 'C++', 'Java', 'PHP', 'TS', 'C#', 'Bash']] . corr()

#Create a mask to hide the upper triangle of the plot
mask = np.zeros_like(correlations)
mask[np.triu_indices_from(mask)] = True

# Plot the heatmap
sns.heatmap(correlations, mask=mask, annot=True,cmap='coolwarm')
plt.show()
```

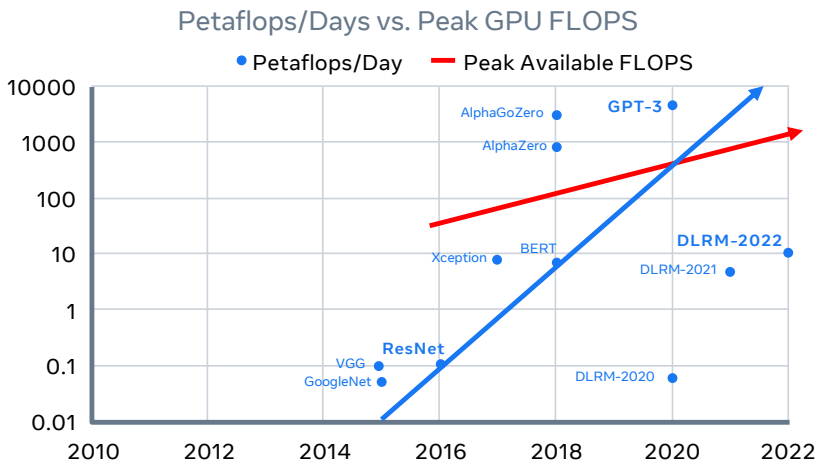
AI has become a system problem

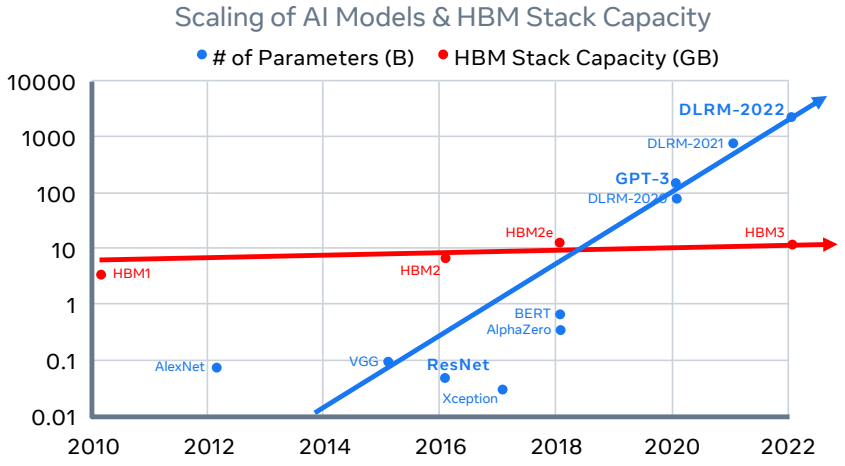
Model parameters have scaled faster than compute and memory over the last 5 years

- Compute growth: ~1/30th
- Memory growth: ~1/300th

Compute & memory needs vary by models

- Recommenders 10X the size but 1/1000th of the FLOPs of LLMs



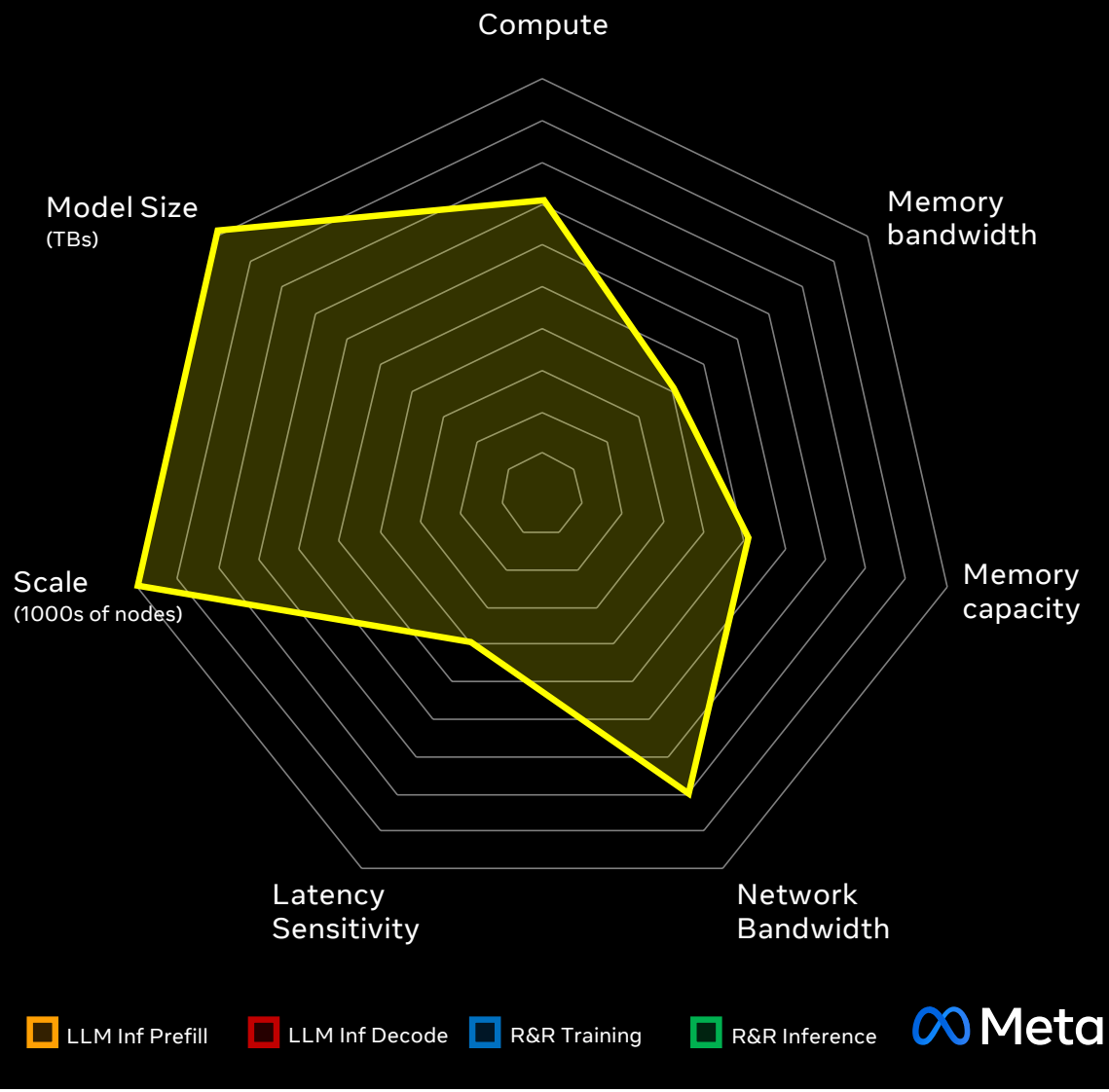


LLMs (Training)

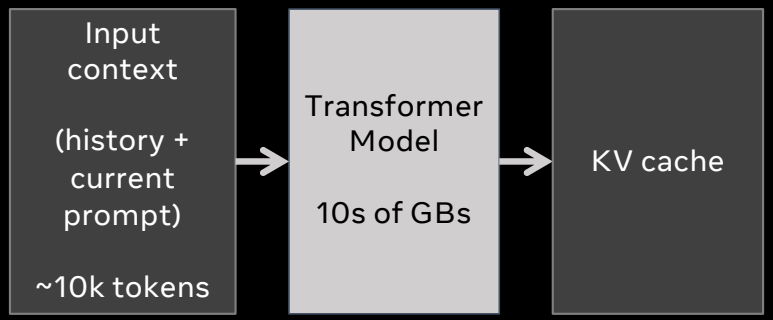
Large model sizes

Across thousands of GPUs to address the compute requirements

3D Parallelism exercises different communication primitives

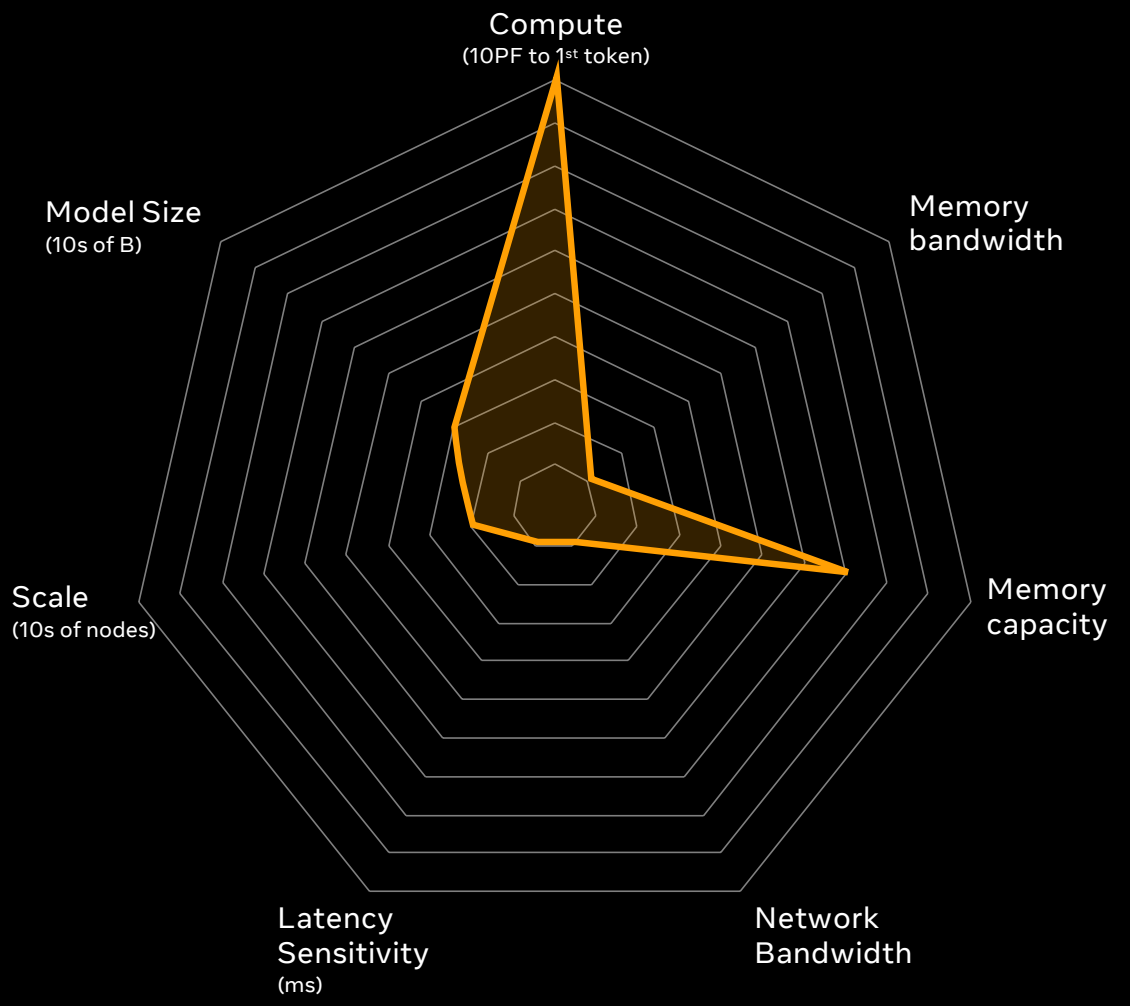


LLMs (Inference - Prefill)



Prefill processes many tokens in parallel → compute bound

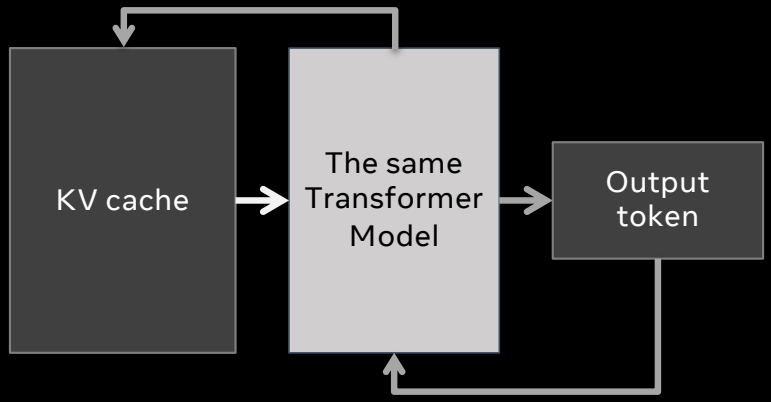
Time to first token <= few sec



LLM Training | LLM Inf Prefill | LLM Inf Decode | R&R Training | R&R Inference

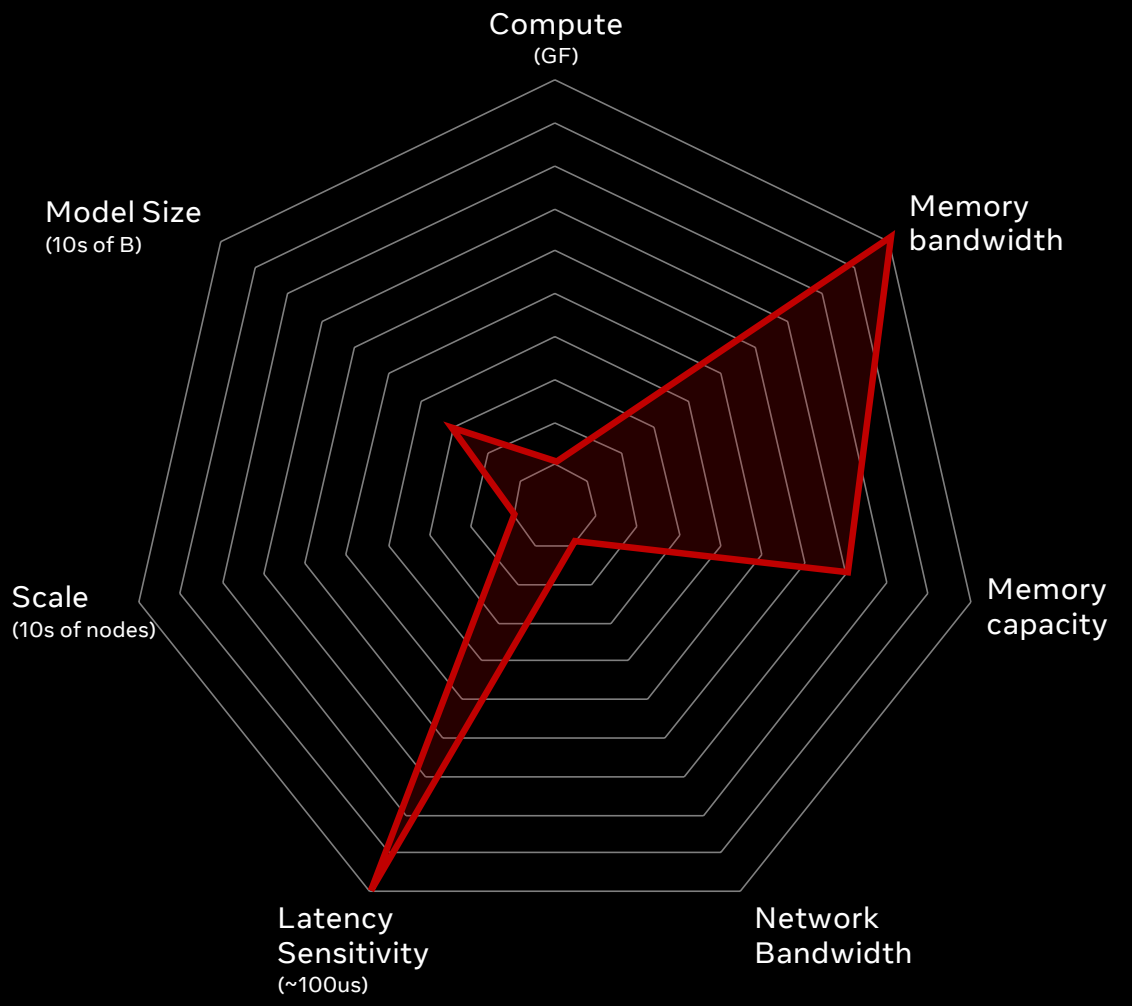
Meta

LLMs (Inference - Decode)



Decoding processes one output token at a time in autoregressive manner

Time per output token <= 100s of ms



LLM Training | LLM Inf Prefill | LLM Inf Decode | R&R Training | R&R Inference

Meta

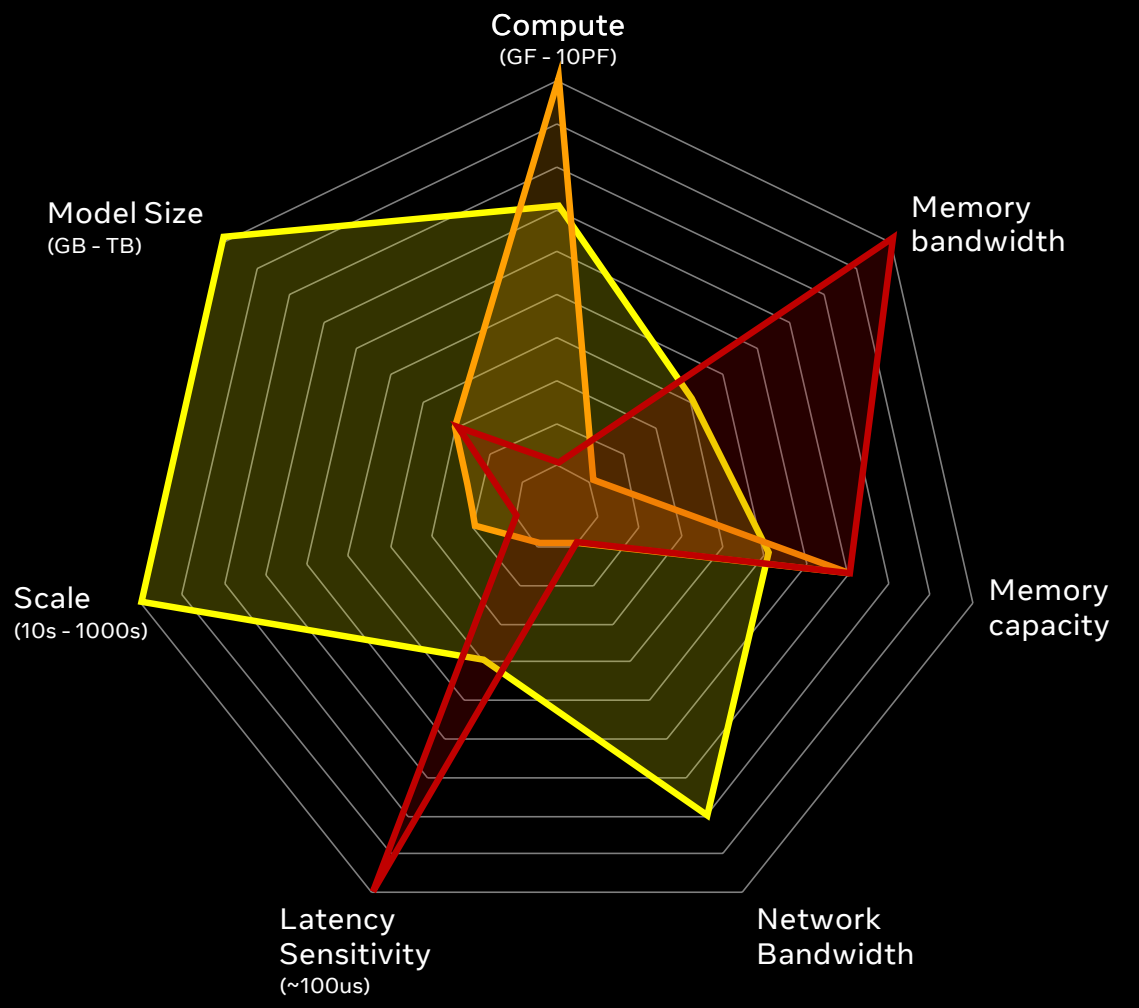
LLMs (Training & Inference)

Training:

- Large (and growing) model sizes
- Thousands of GPUs to address the compute requirements
- 3D Parallelism exercises different communication primitives

Inference:

- Diverse requirements between prefill (compute bound) and decode (memory bound)

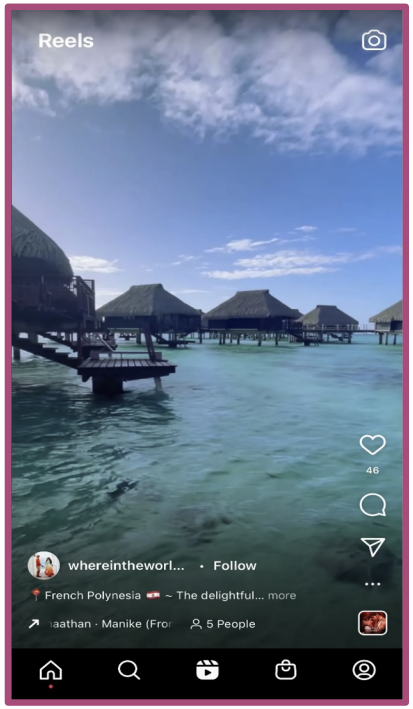


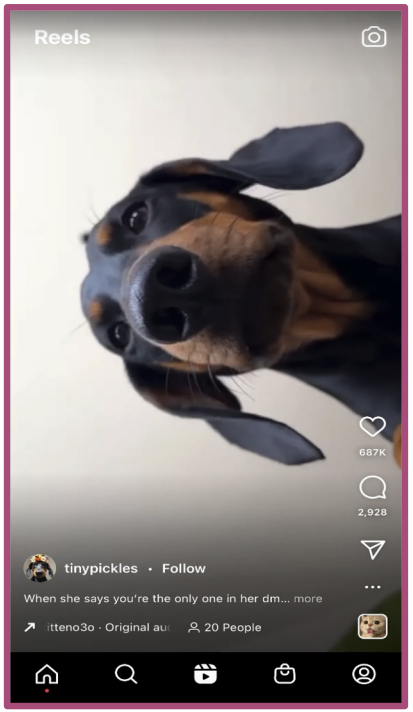
Meta

Ranking & Recommendation

Making personalized recommendations by ranking previous interactions to predict future interests

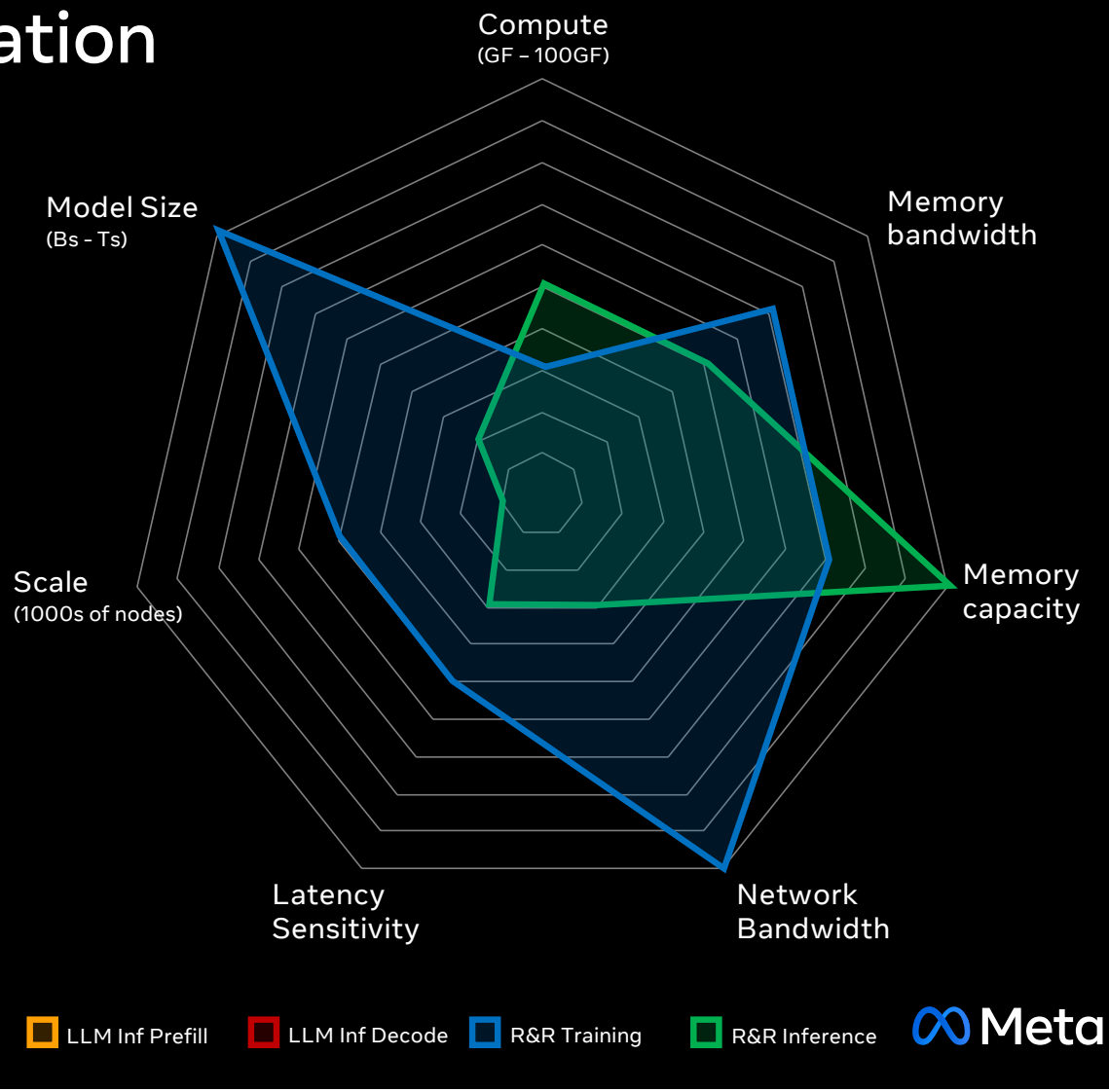
Leverages giant **Deep Learning Recommendation Models** (DLRMs)





Ranking & Recommendation (Combined)

- Model size often even bigger than LLMs (mostly from embeddings)
- High global bisection bandwidth for all-to-all communication
- Model size for inference can be reduced by up to ~10x (quantization, compression, pruning)



Compounding ops complexity

High Scale

x Extreme Freshness

x Multi-surface

x Multi-modality

On path to 10s of billions of rows/day

Signals in seconds
Streaming Features in < a minute
Batch Features in < 1 day
Models in minutes

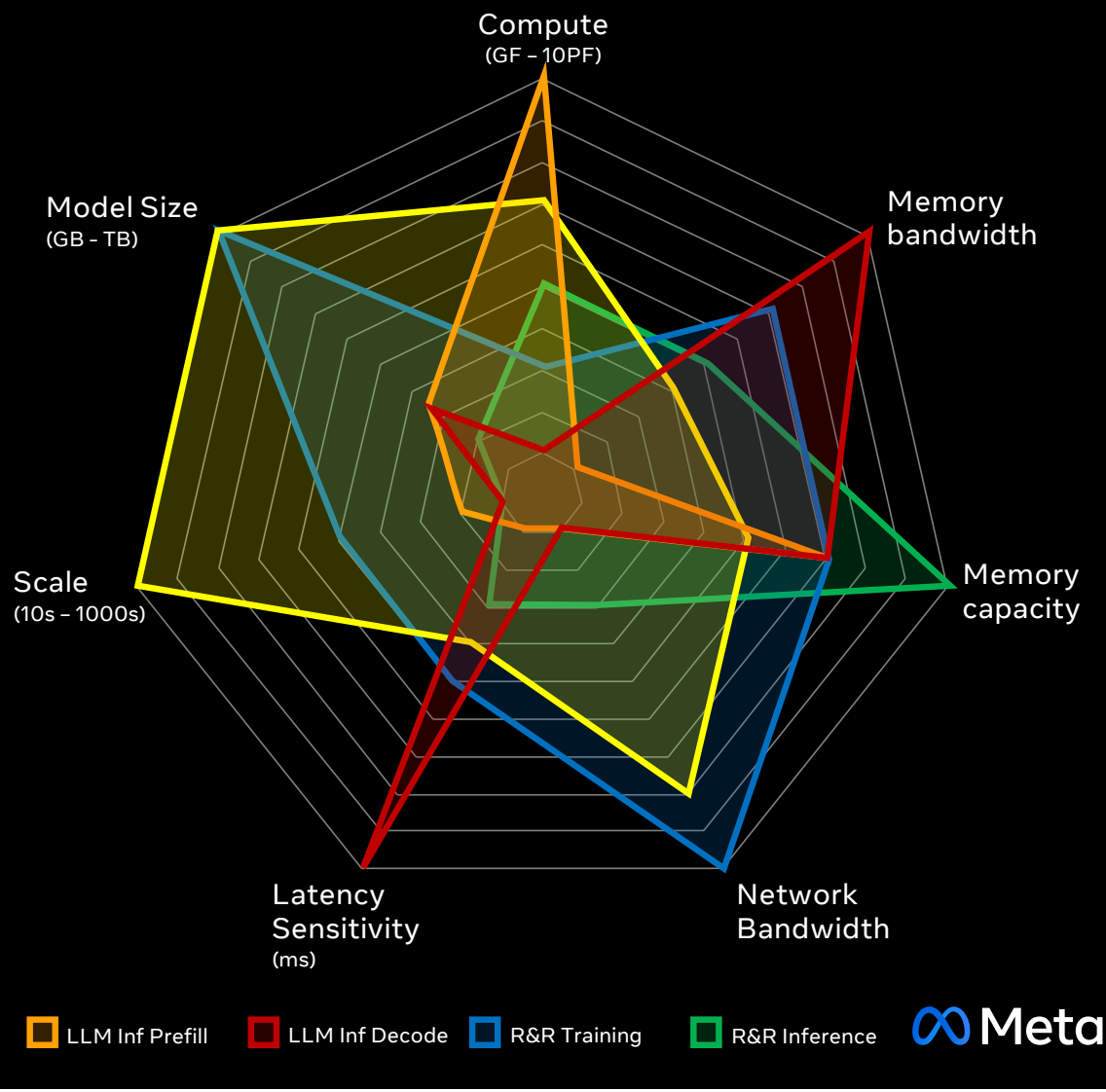
Reels + Stories + Feed + Shopping
Instagram + Facebook

Reactions + Captions + Comments
+ Photos + Videos + Audio + AR/VR

Meta

Diversity of AI system requirements

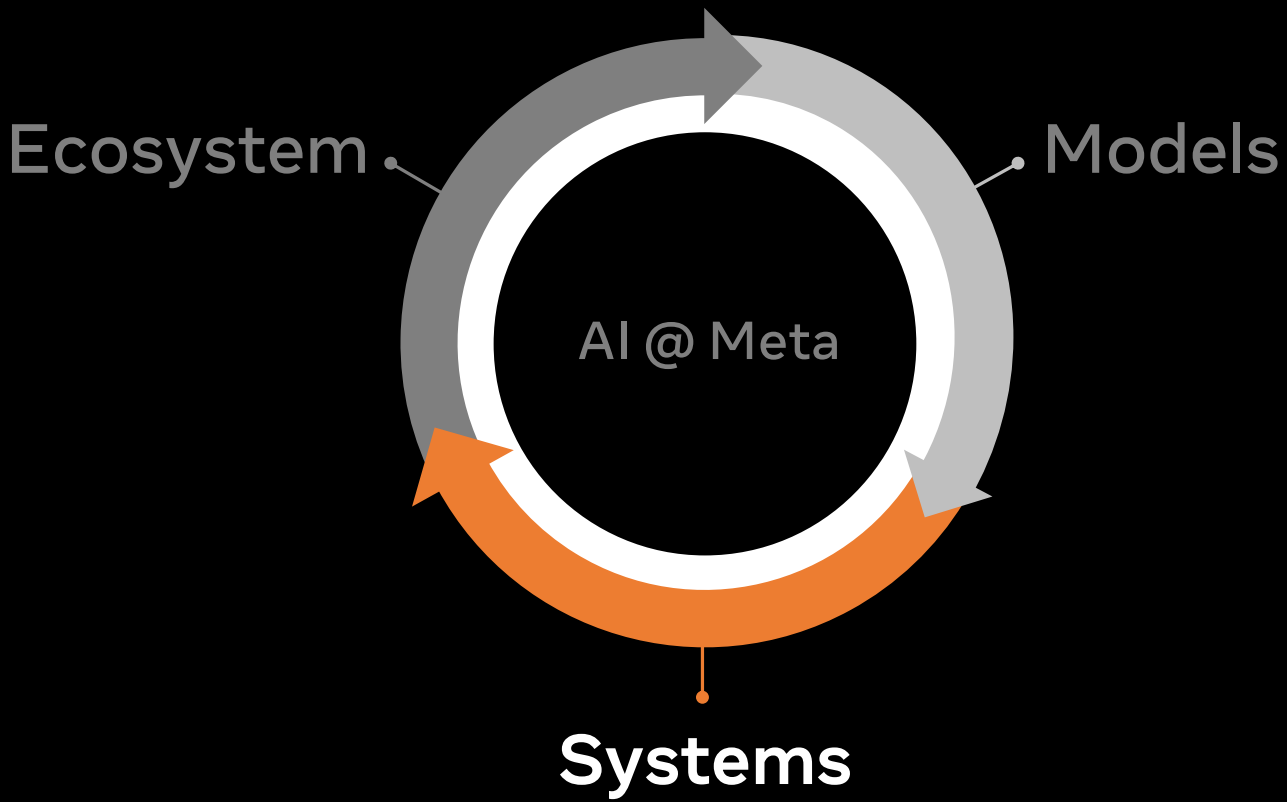
- Difficult to serve all classes of models with a single system design point
- New models & parallelism techniques put unexpected pressures on AI systems
- The next frontier of innovation is in software/hardware co-design



Meta

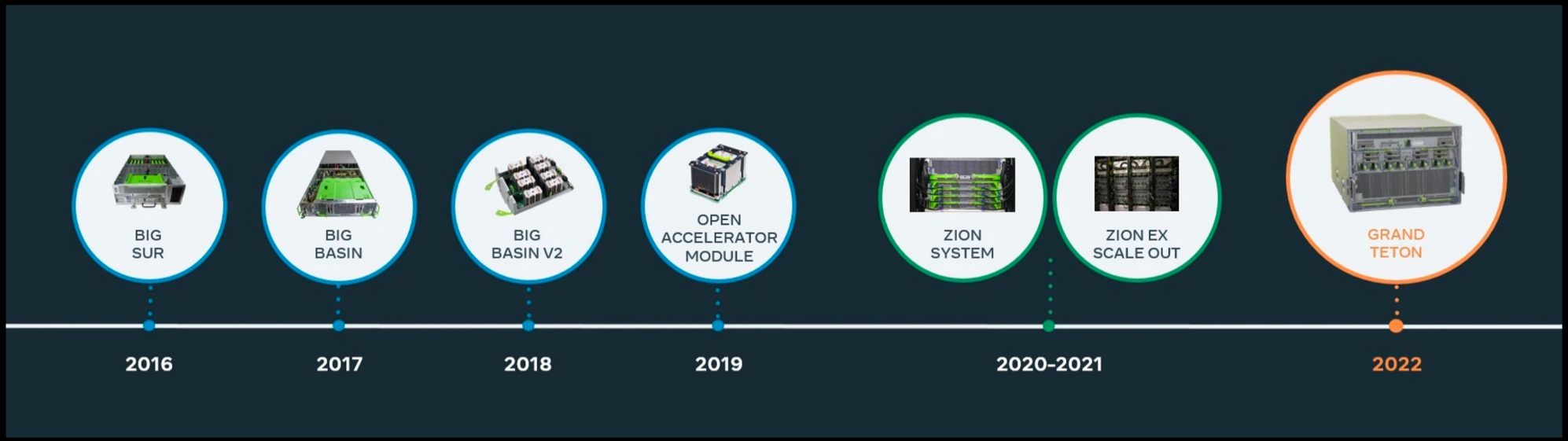
Models evolving quickly along divergent paths

Our challenge: deliver balanced systems that efficiently serve all of our models



Meta

Meta OCP contributions for AI systems



Meta

Recapping diverse model requirements

	Job Size (GPUs)	**Scale-up** Network BW	**Scale-out** Network BW	Major bottleneck
Ranking	10s-100s	0.5 TB/s		IO
LLM	8k-16k	1 TB/s	0.05 - 0.1 TB/s	Compute

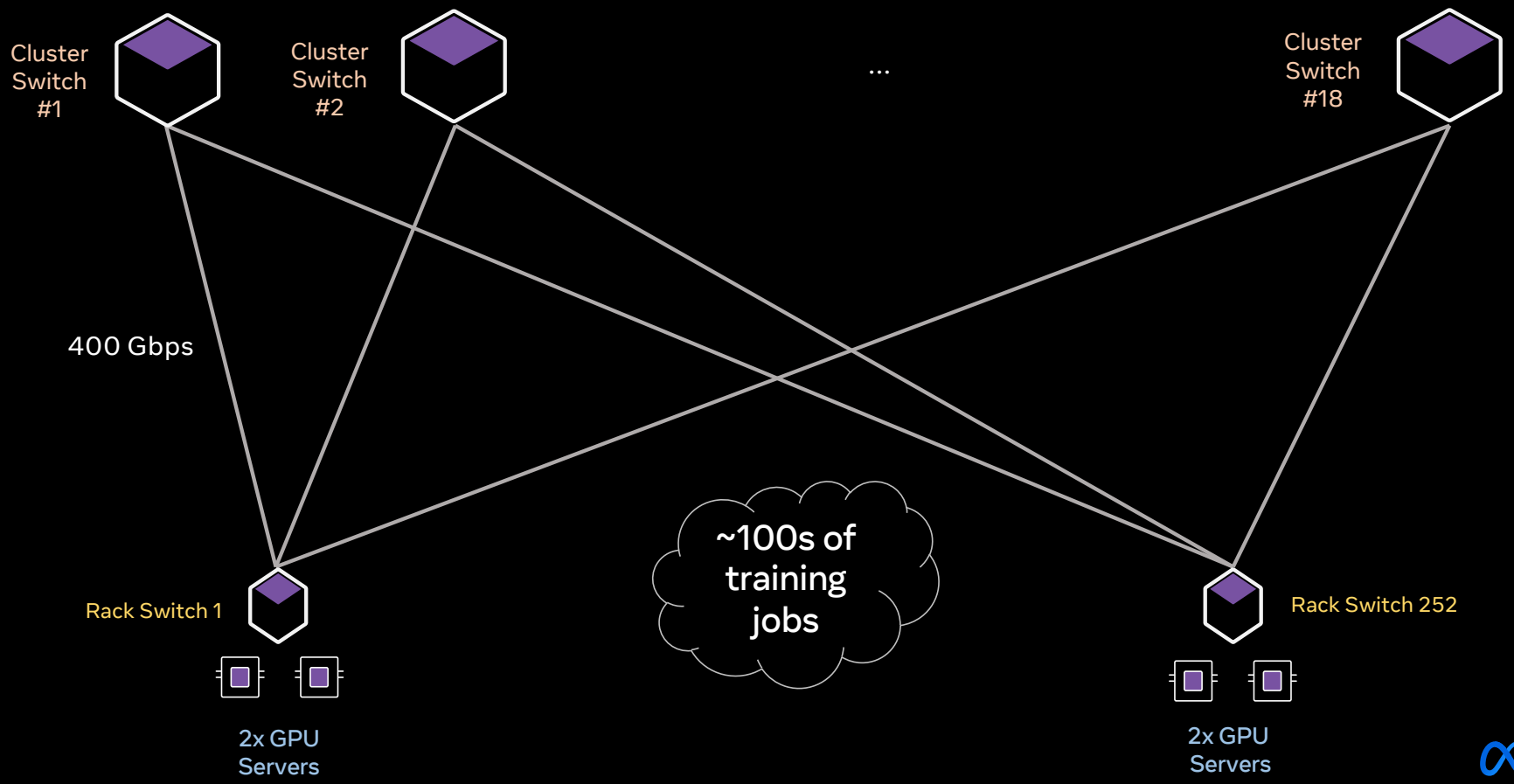
Meta

Ranking Training Systems:

Medium-size ”flat fabric” running multiple training jobs ("AI cloud")

Meta

Ranking training cluster: ~4K GPUs (2021+)



Meta

LLM training Systems:

Massive amount of compute for one big job

Target for 2023: 20+ ExaFLOPs FP8

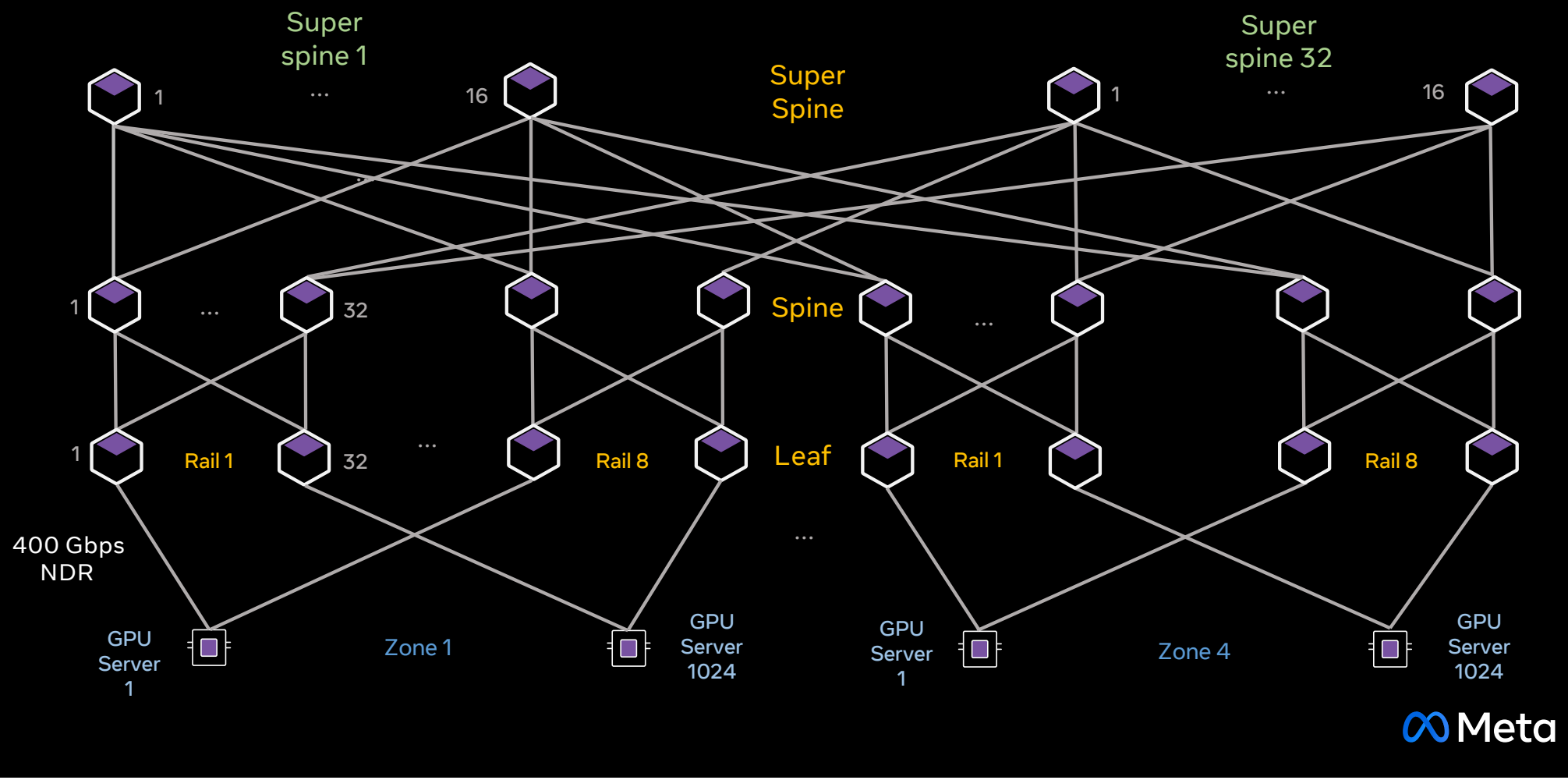
→

LLaMa-65B training in <1 day

Meta

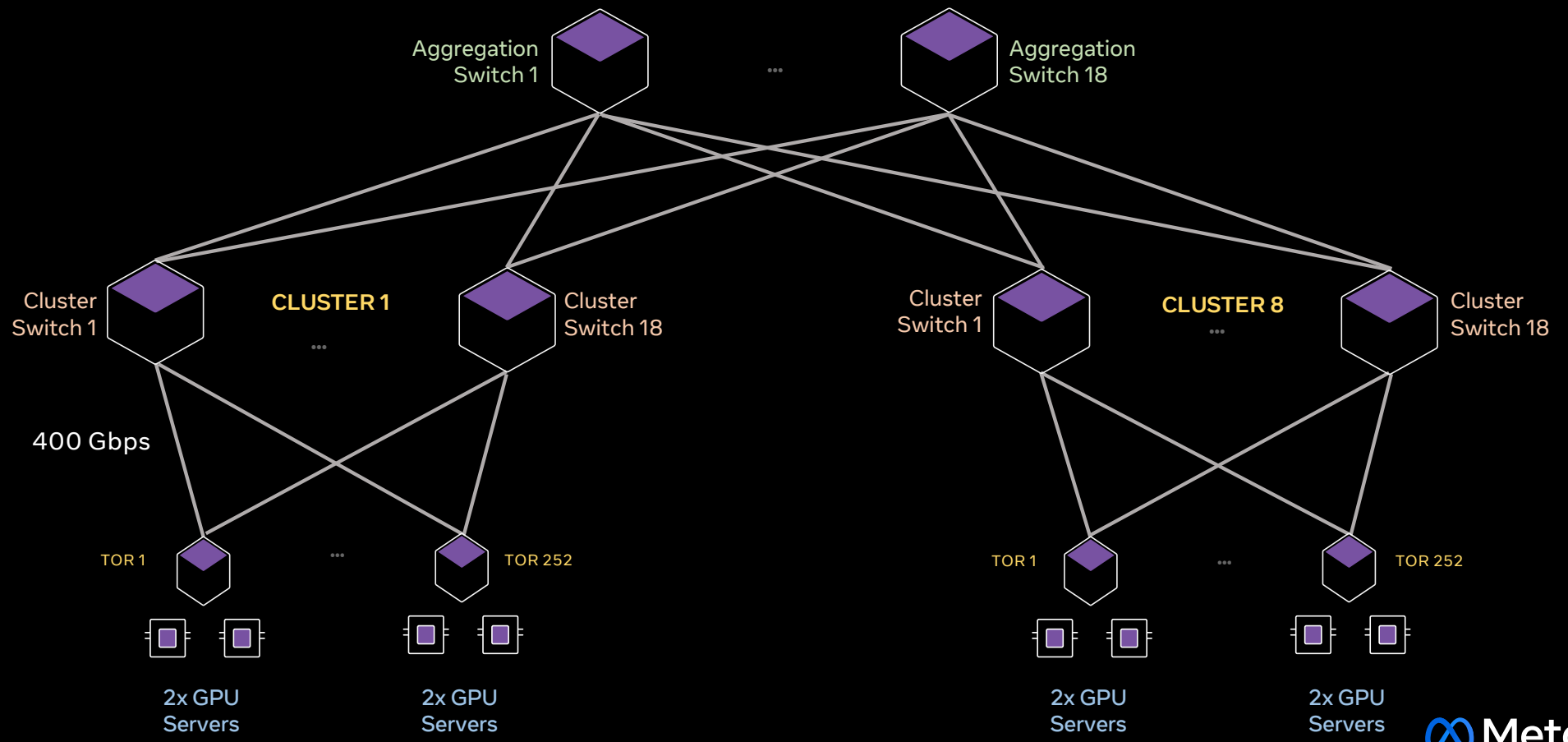

First LLMs were trained in
FAIR Research Super Cluster:
InfiniBand cluster of 16K GPU

LLM Training: InfiniBand cluster for 32K GPUs



Meta

LLM Training: Ethernet cluster for 32K GPUs



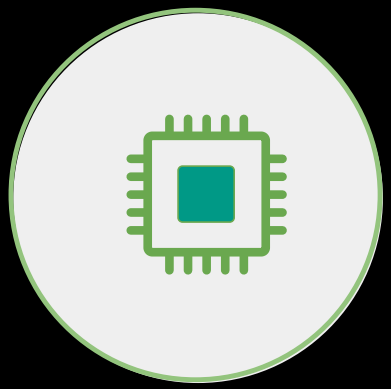
Meta

There are no reference designs in the industry, and we are looking forward to share our learnings in building large clusters – InfiniBand and Ethernet

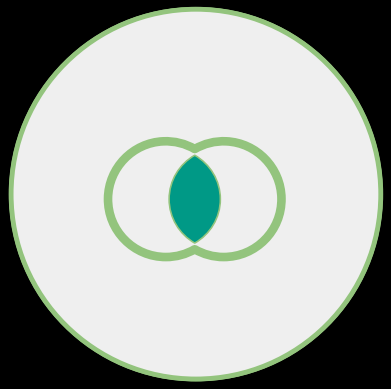
Meta

Scaling LLM training to 32K GPUs

Keep global batch size same up to 32K GPUs!



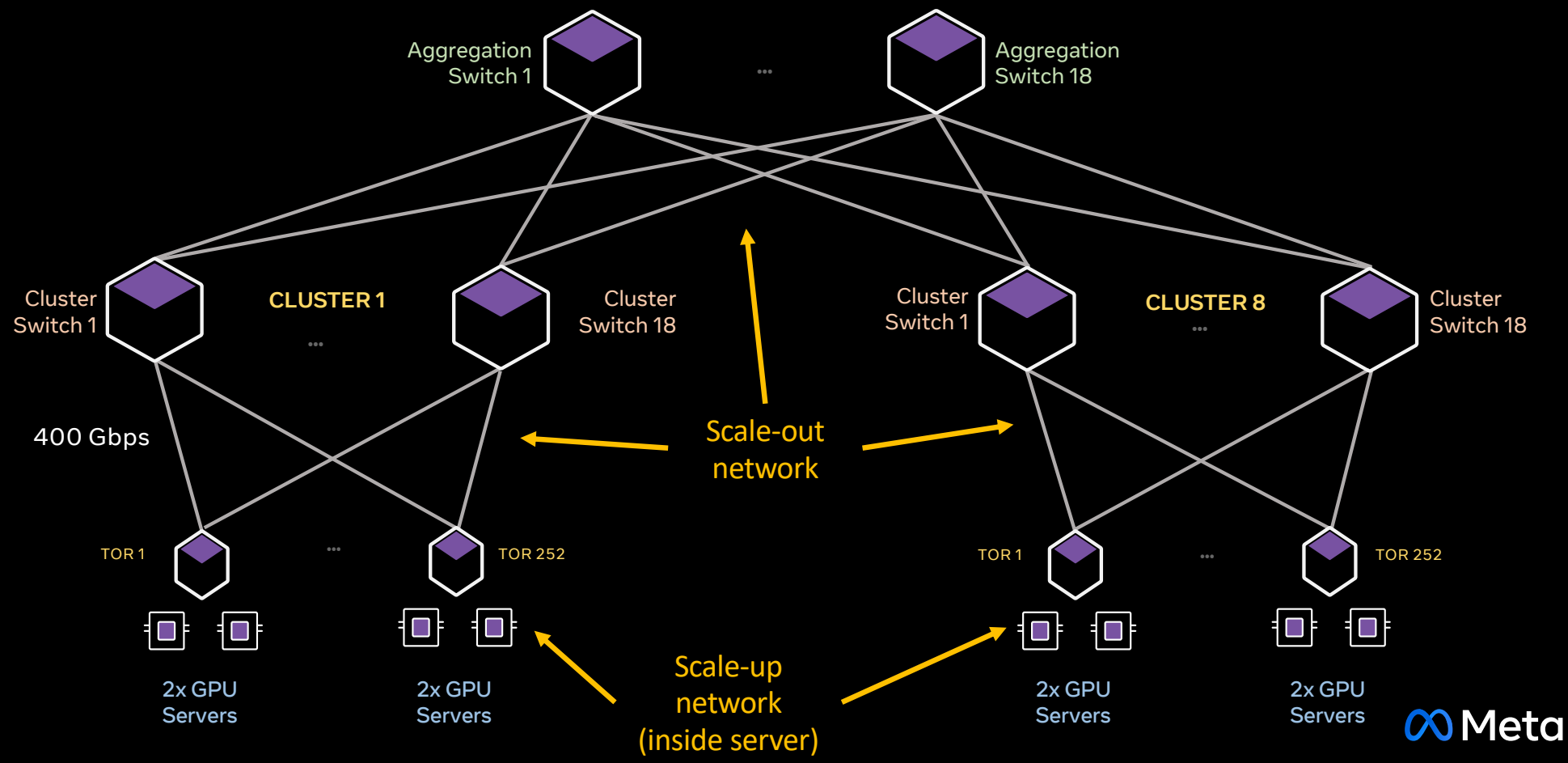
Today's batch size is 2k - runs out of samples per GPU



2D/3D parallelization: data + model-parallel (tensor + pipeline)

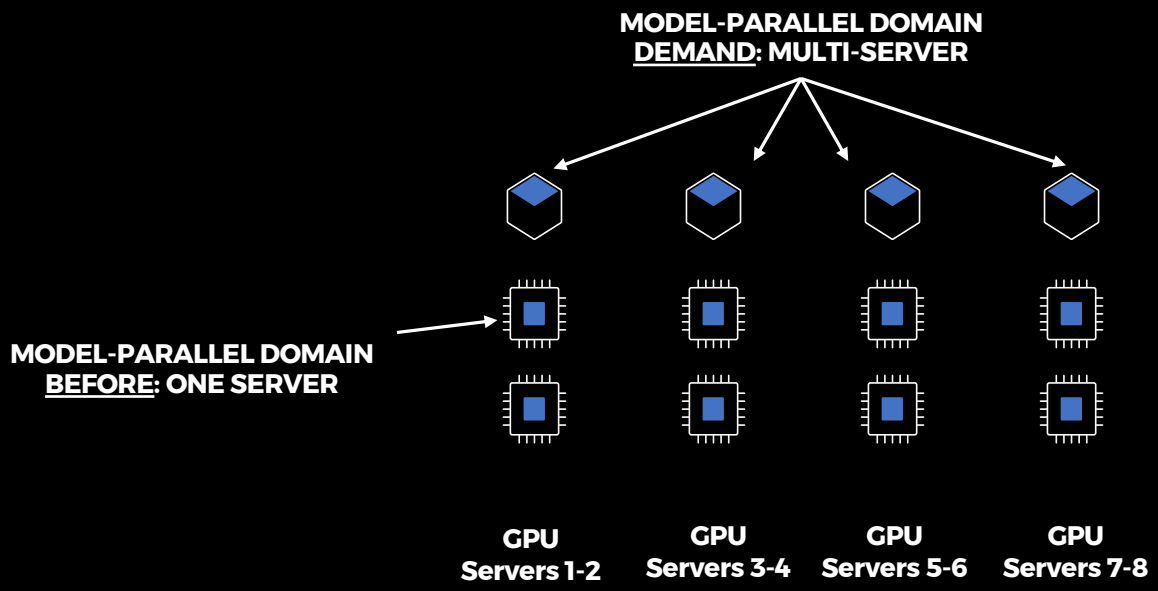
Meta

LLM Training: Scale-out vs. Scale-up domains



Model parallelism pushes scale-up networking multi-rack:

64-128 GPUs , ~1TB/s



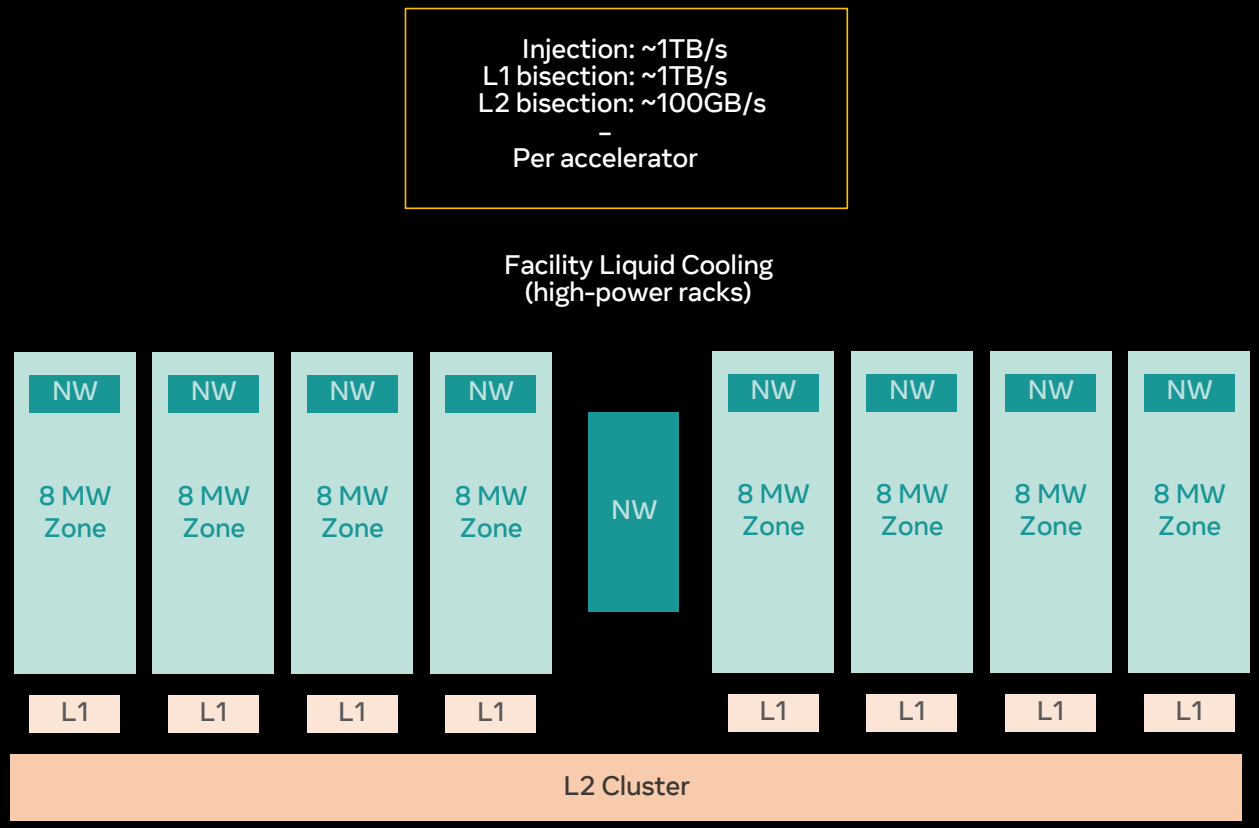
Meta

Network is getting "fused" with compute

High-speed, power efficient
interconnect & endpoints

Scale-up and scale-out becoming
one fabric in the long-term

Unified DC infrastructure (2025+): LLMs and Ranking



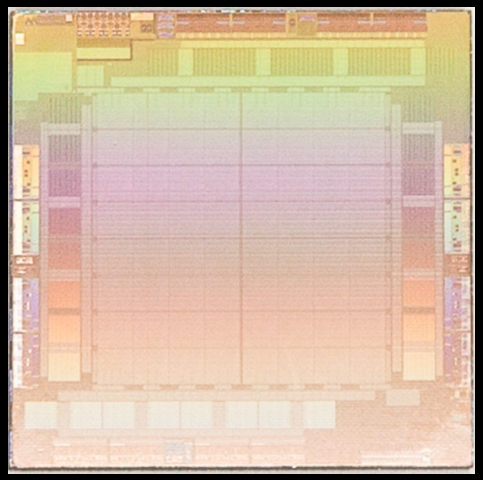
Meta

"Brute-force" compute is not enough: model innovation is necessary to keep infrastructure efficient

Likely sparse models: MoE, switch transformers ...

Meta

MTIA v1 – 1st gen “Meta” ranking inference accelerator



	Pre-MTIA	MTIA v1
INT8	30-45 TOPs	102.4 TOPs
FP16	4-6 TOPs	51.2 TOPs
DDR	16 GB	32 GB (LPDDR)
TDP	12 W	25 W

Meta

What's after MTIA v1?

Inference

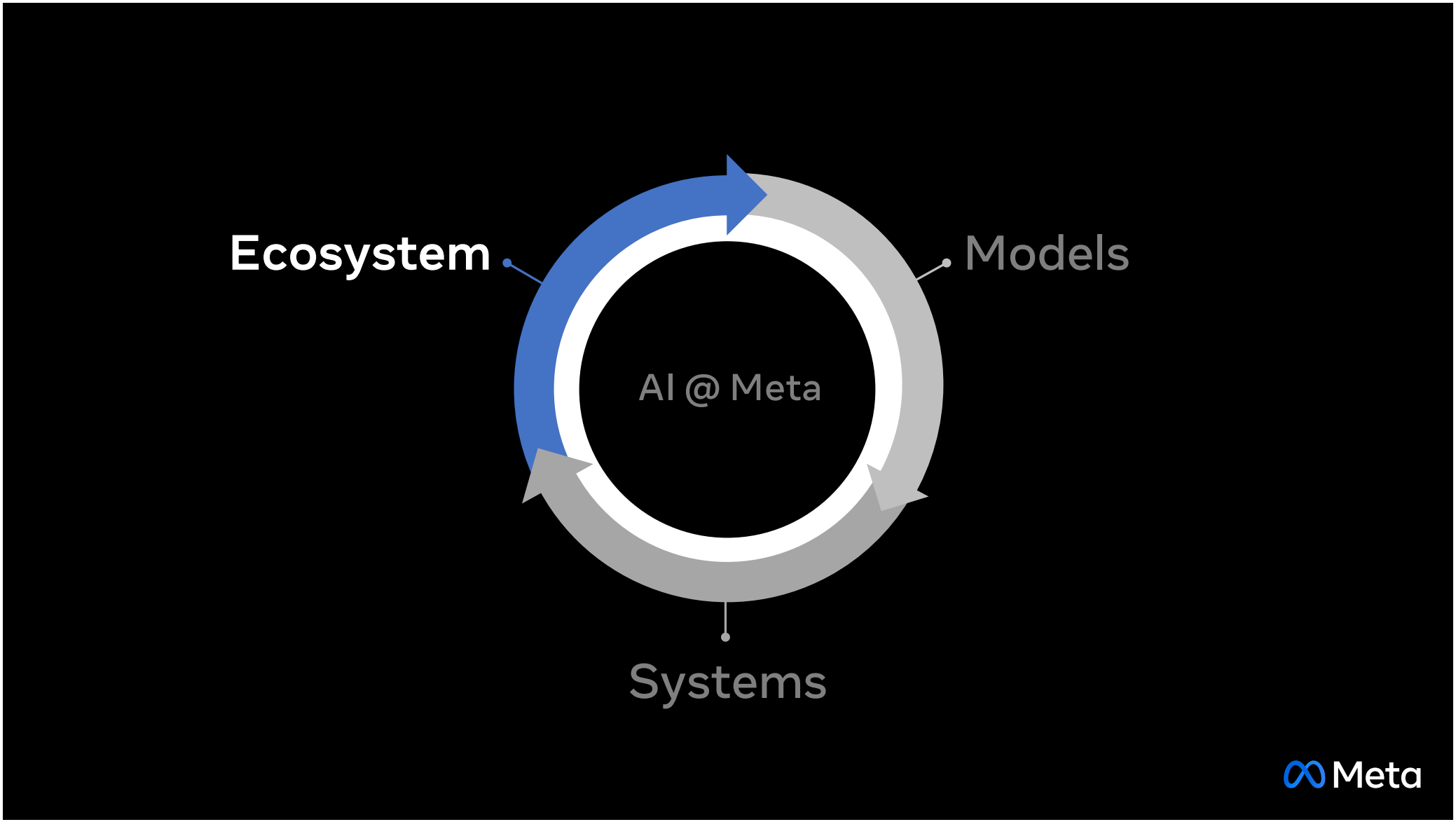
- ~2x INT8/FP16 in the next chip
- Optimized Perf/TCO at system level
- Native PyTorch eager mode

Training

- Build on inference learnings
- 1TB/s+ IO and non-blocking fabric
- Performant comms and collectives

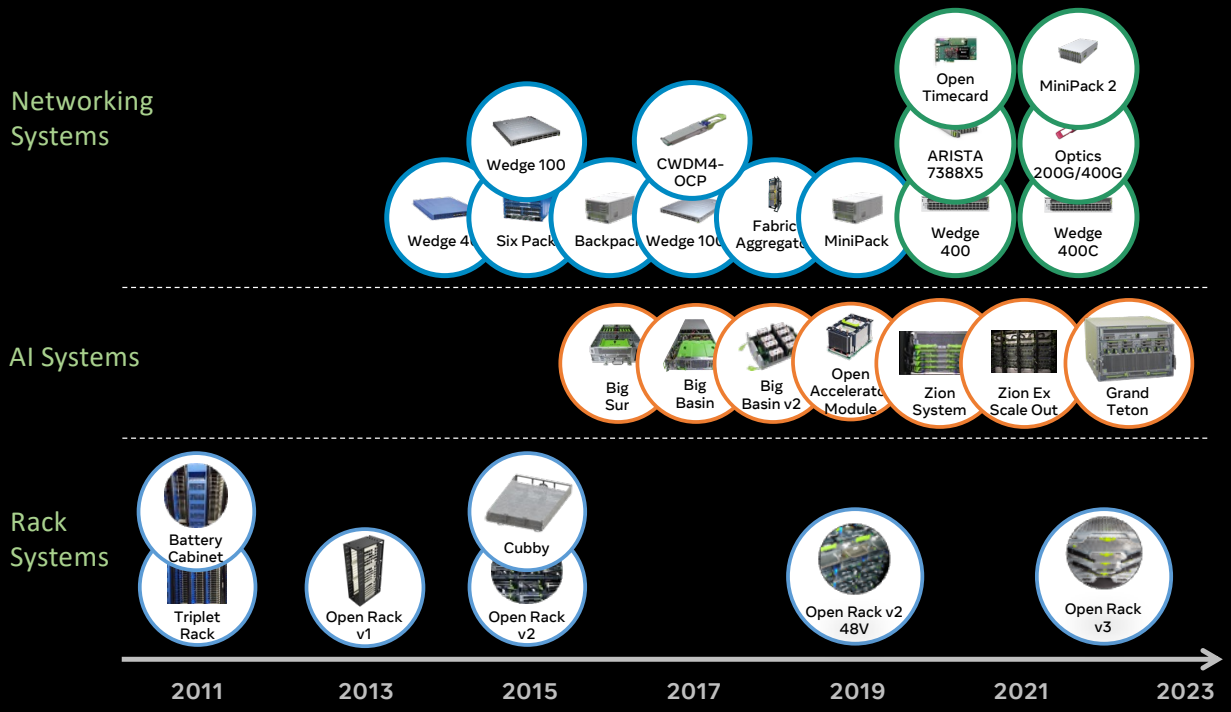
Optimized for ranking & recommendation

Meta


Ecosystem
Models
AI @ Meta
Systems
Meta

Our ongoing commitment to open

OCP Contributions



Other Initiatives

RISC-V®

UCIe Universal Chiplet Interconnect Express

CXL Compute Express Link

Ultra Ethernet Consortium

OIF

Data Formats (MXFP...)

Meta

Collaborating on tools & frameworks

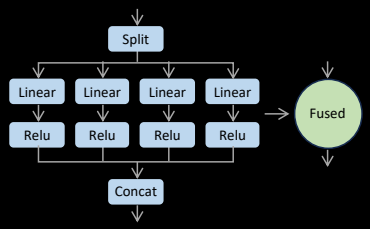
PyTorch

PyTorch: An open-source ML framework that offers flexible and efficient deep learning model development

OpenAI Triton: A high-performance programming language that enables researchers to write highly-efficient GPU code

TensorRT-LLM

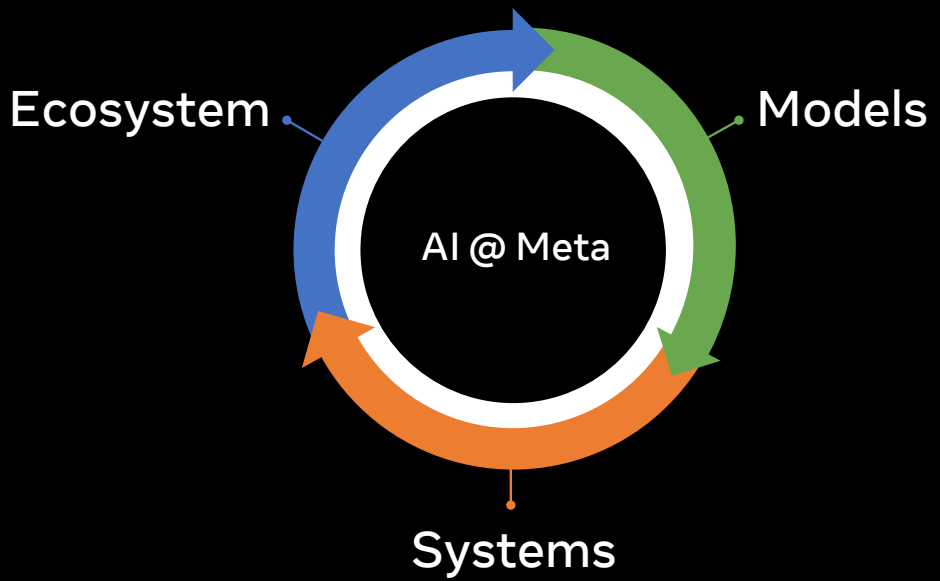
TensorRT-LLM: A collaboration with NVIDIA and others to accelerate and optimize LLM inference on H100 GPUs



AITemplate: A unified inference engine for GPUs that delivers near hardware-native performance on a variety of models

Meta

Summary



Rapidly-evolving models are creating increasingly divergent system requirements

Performant infrastructure requires a heterogeneous approach

Developer velocity and efficiency requires an abstraction to the underlying hardware

Opportunity for step change innovation in hardware capabilities and sustainability

Meta

Thank You!

Meta