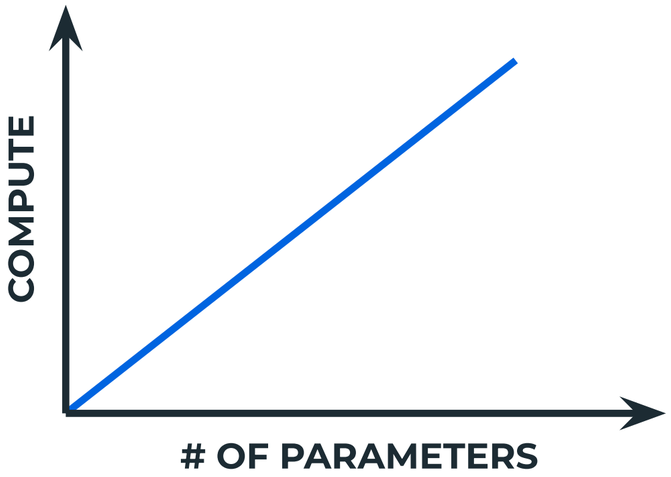
Scaling Large Language Model Serving Infrastructure at Meta

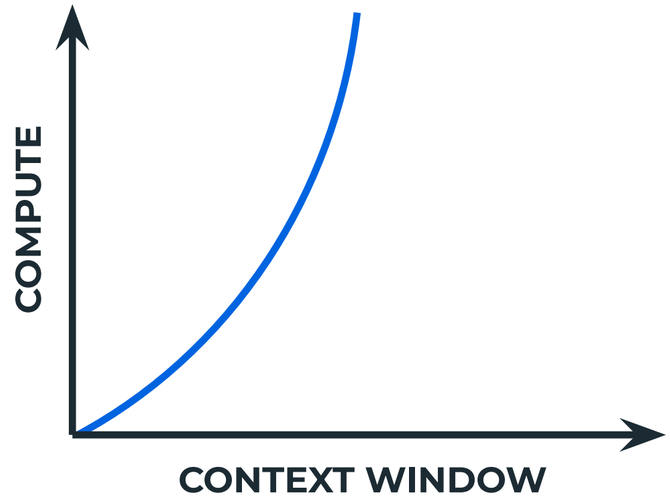
A comprehensive recipe to turn LLMs into LLM serving infrastructure

Ye (Charlotte) Qi
AI Inference @ Meta

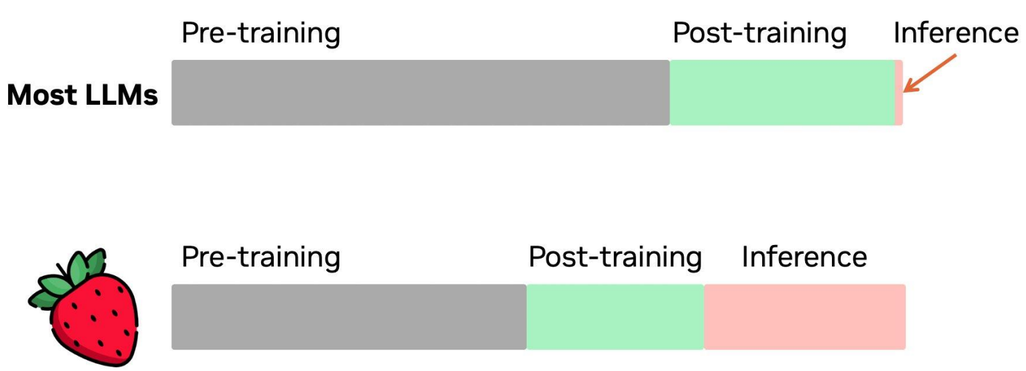
Meta

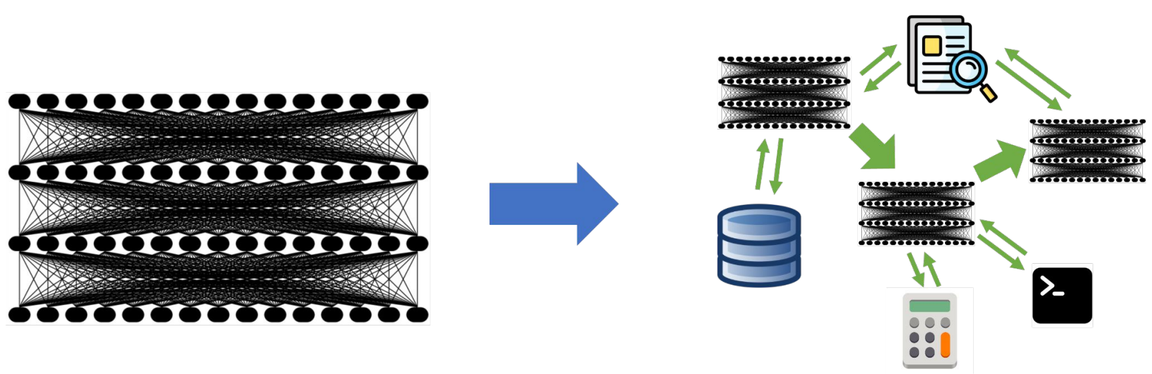
The AI Gold Rush





Inference Scaling and Compound Systems Are Coming





credit:
https://x.com/DrJimFan/status/1834279865933332752
https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/

Background about Myself

Ye (Charlotte) Qi

AI Infra

Had been running model services for 6.5 years

- Ads model serving
- LLaMa serving

Machine translation research before Meta

We Support Product Backend for Meta AI

500M

Monthly active users

Behind the making of LLaMa

Meet Llama 3.1

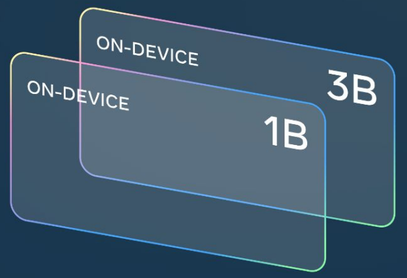
405B

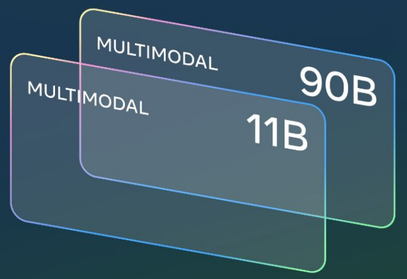
70B

8B

INTRODUCING

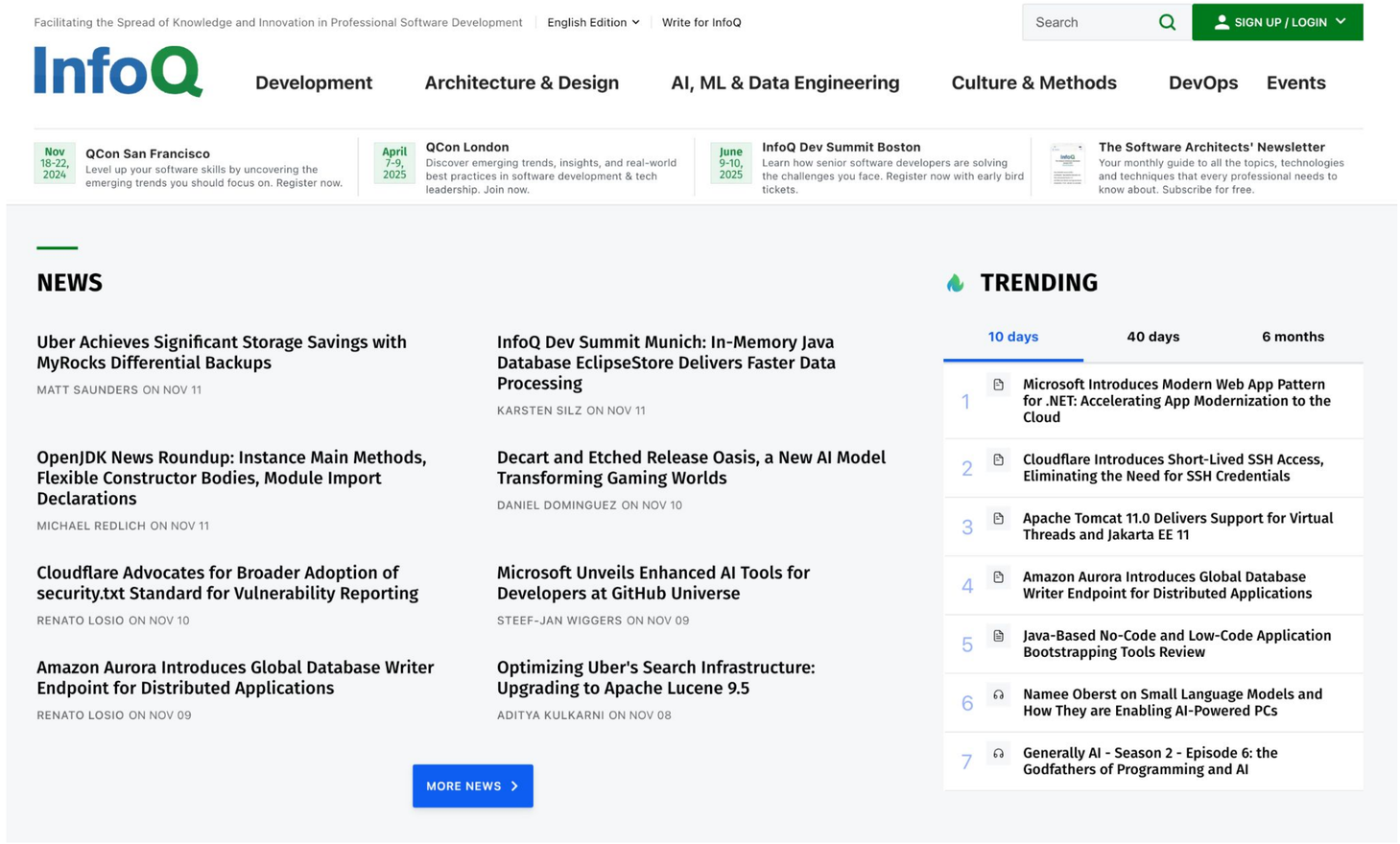
Lightweight and multimodal Llama models





“Should I run my own LLM services?”

Let’s Build This Step By Step



Challenge 1
Fitting

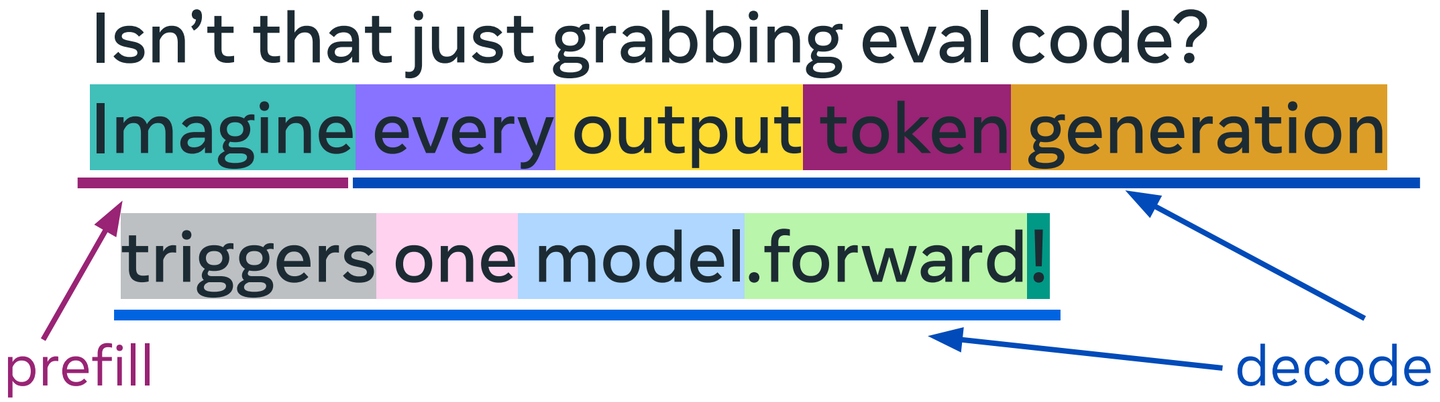
Challenge 2

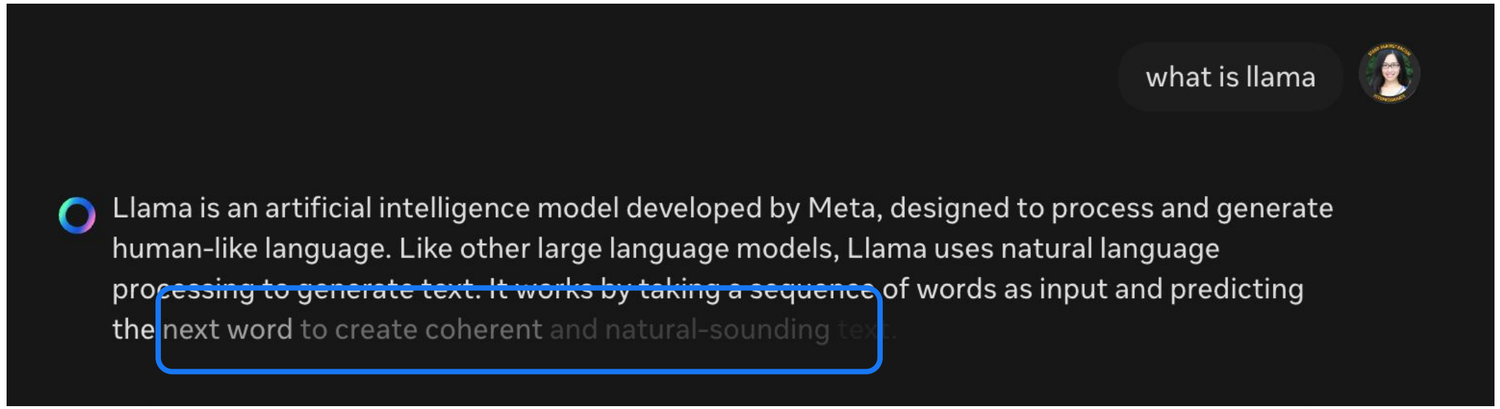
Challenge 3

Challenge 4

STEP 1

Find a good runtime

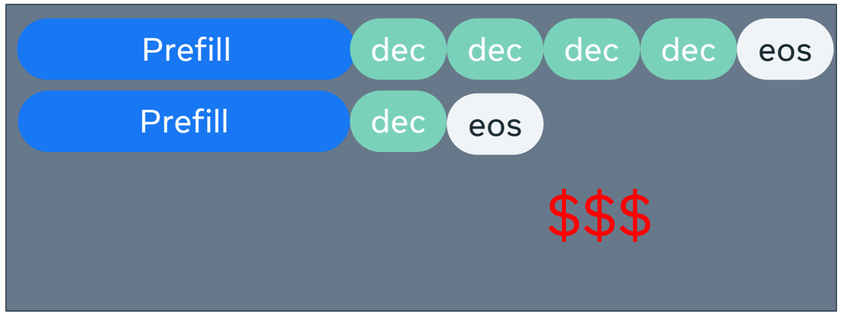




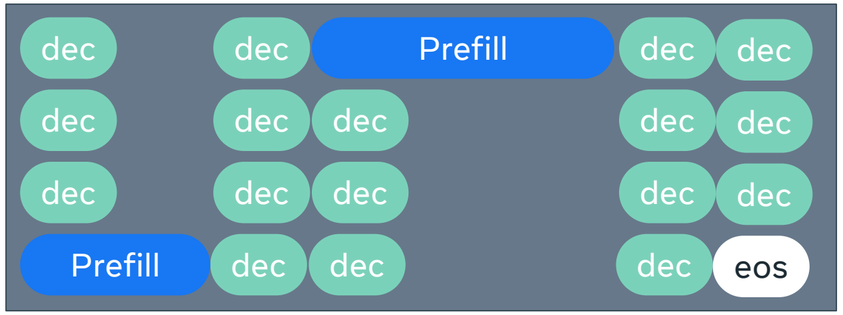
working on it!!!

Continuous Batching

Not this



Use this



KV Cache

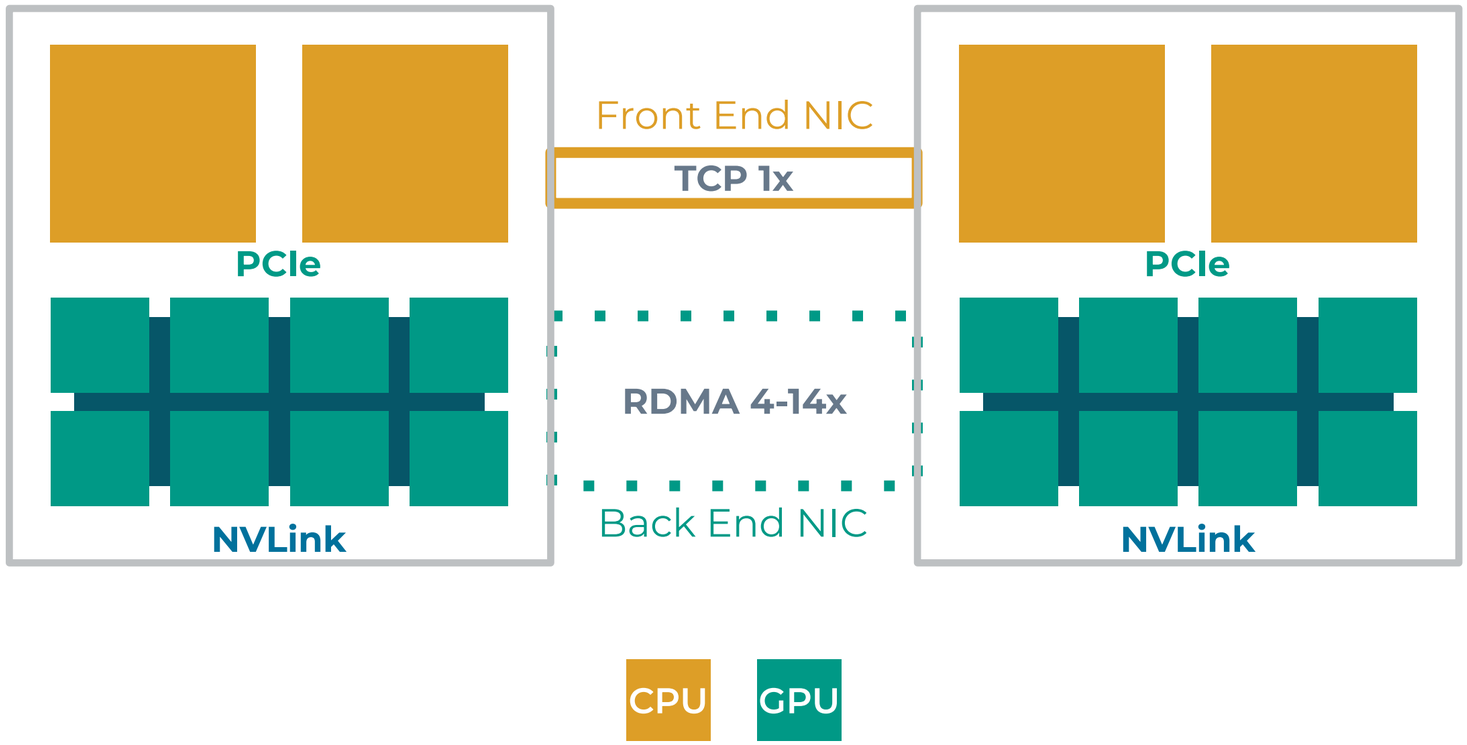
> How does KV cache work?

Imagine this sentence being generated by an LLM. KV tensors for yellow parts are cached in GPU memory at 320KiB/tok (LLaMa3-70B), 128KiB/tok (LLaMa3-8B) under bf16.

vLLM SGL TGI NVIDIA TensorRT-LLM deepspeed

STEP 2

Understand hardware resources



Let’s Only Worry About Model Loading

40/80GB

NVIDIA A100

80/96GB

NVIDIA H100

192GB

AMD Mi300x

STEP 3: Start fitting some models with 80GB H100x8

STEP 3: Single-Card Inference

LLaMa3-8B

bf16: 16GB **< 80GB**

STEP 3: Distributed Inference: Tensor Parallelism

LLaMa3-70B

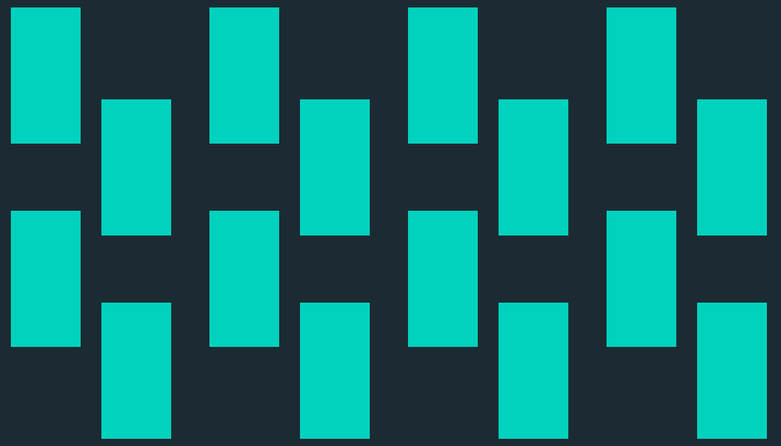
bf16: 140GB **< 80GB x 2**

Partitioning Weights

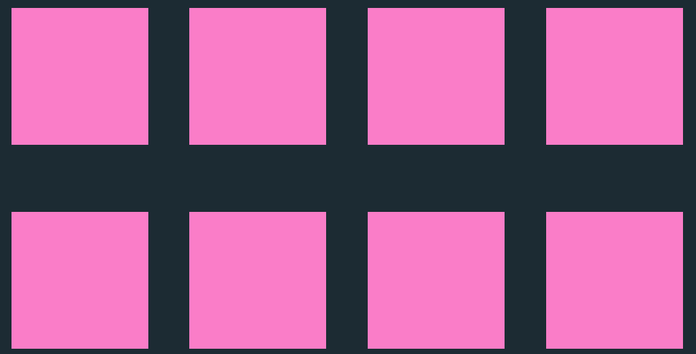
STEP 3: Distributed Inference: Pipeline Parallelism

LLaMa3-405B

bf16: 810GB **< 80GB x 16**
bf16: 810GB **< 192GB x 8**



Partitioning Weights More



Or Find GPUs With Bigger HBM

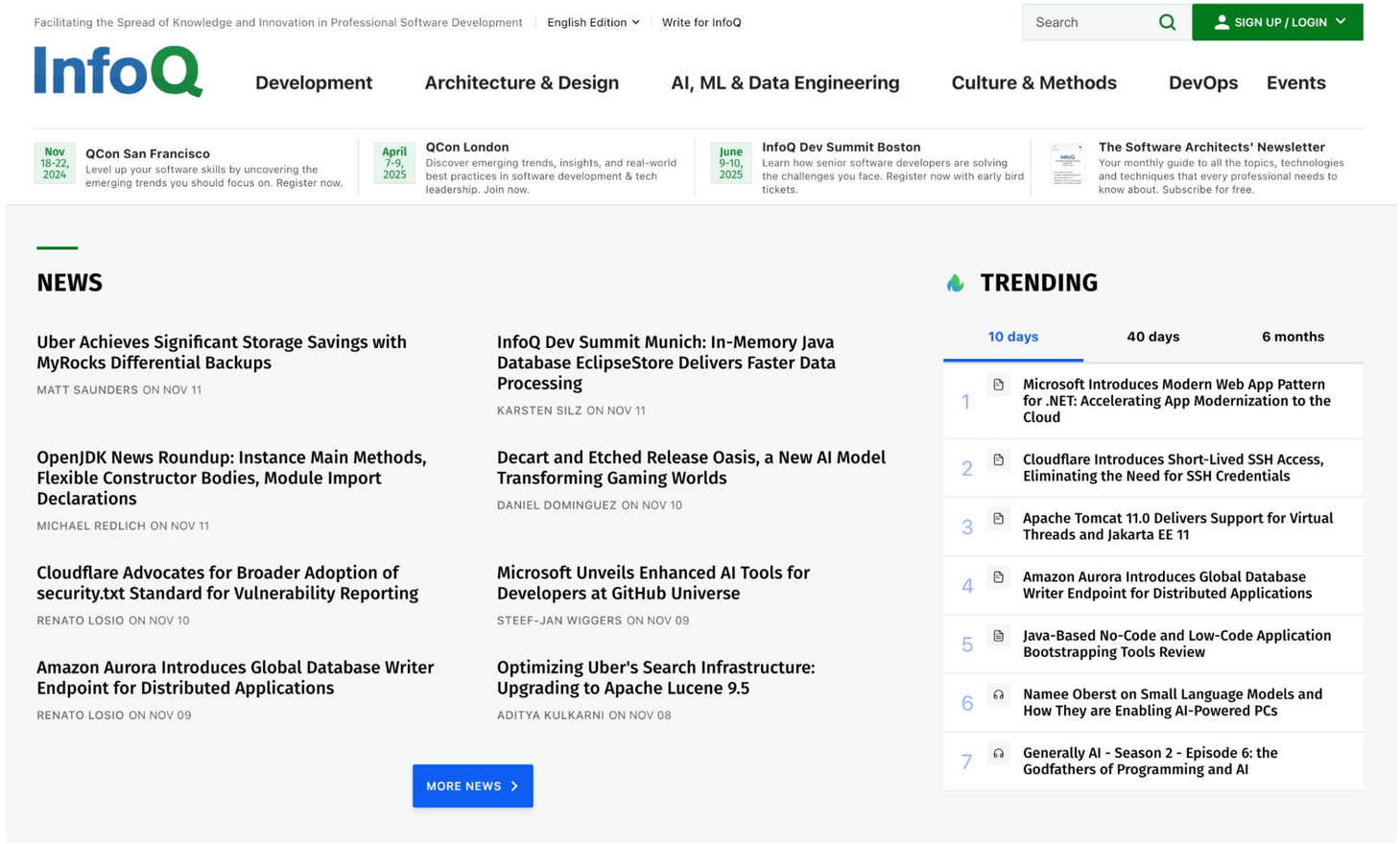
Takeaways

Find a specialized LLM runtime as your starting point

Understand system resources available on AI hardware

Use tensor/pipeline parallelism to fit your models

Let’s Build This Step By Step (Part 2!)



Challenge 1
Fitting

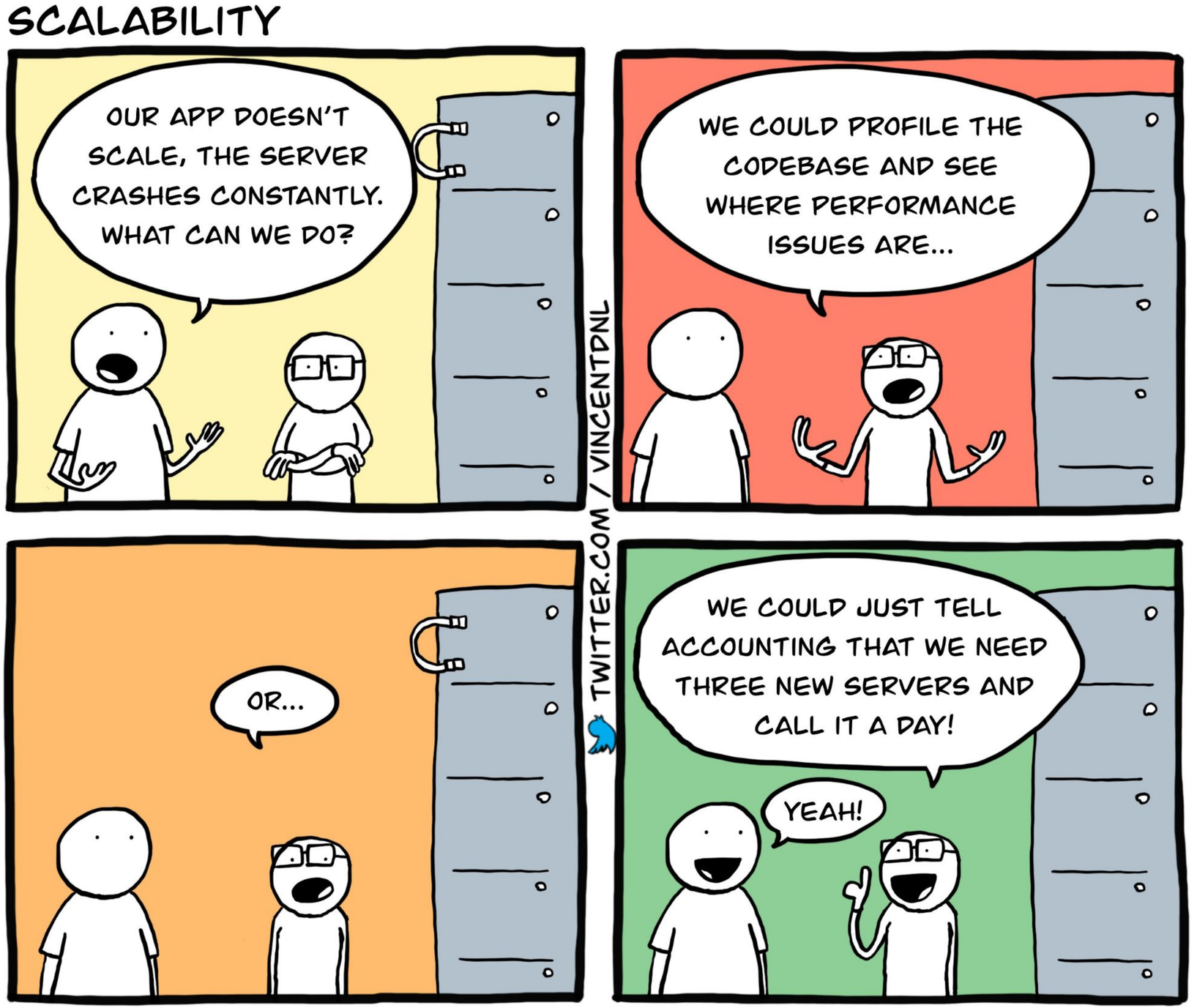
Challenge 2
It’s Slow

Challenge 3

Challenge 4

Making it faster ...?

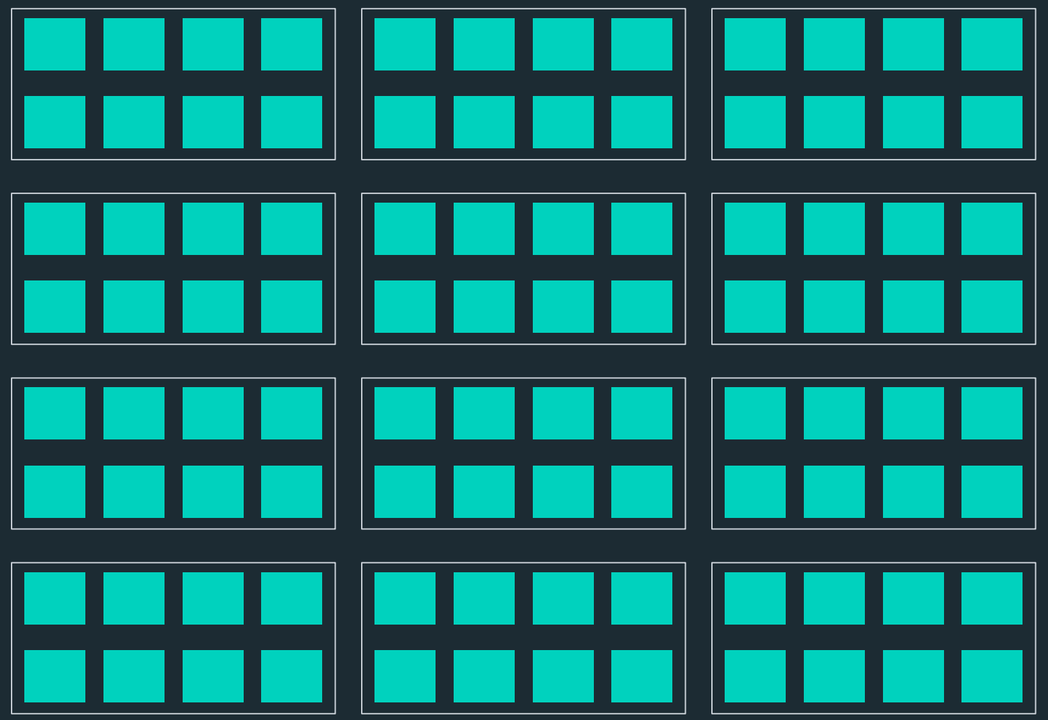
Throw capacity at the problem?



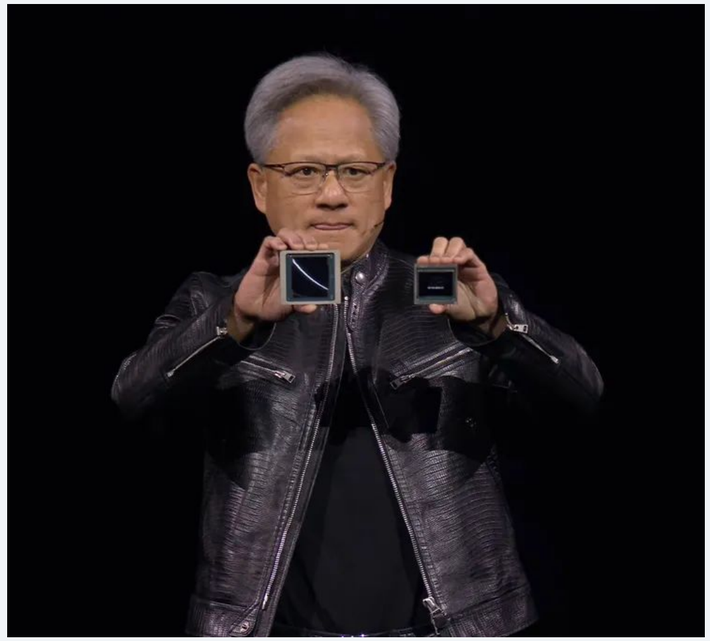
credit:
reddit.com/r/ProgrammerHumor/comments/hnlge5/why_solve_problems_when_you_can_just_throw_money/

STEP 1 Throw More Capacity

Add more replicas



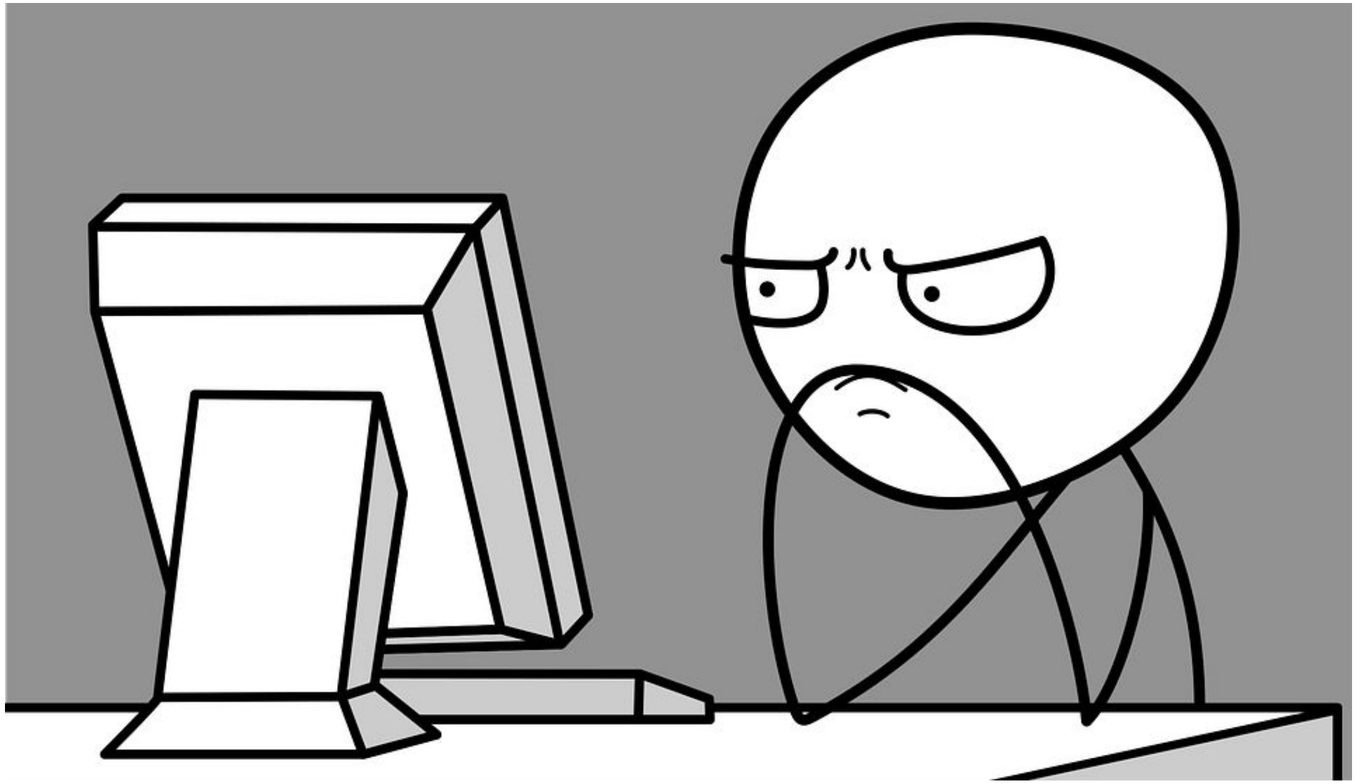
Buy faster hardware



(Image credit: nvidia)

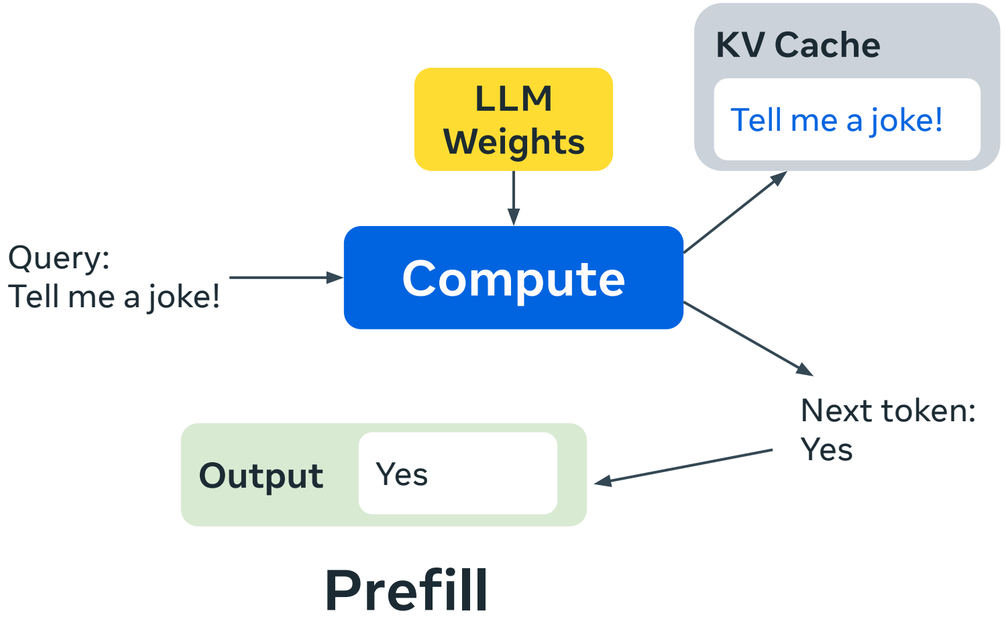
```
usage: vllm serve [-h] [--host HOST] [--port PORT]
                  [--uvicorn-log-level {debug,info,warning,error,critical,trace}]
                  [--allow-credentials] [--allowed-origins ALLOWED_ORIGINS]
                  [--allowed-methods ALLOWED_METHODS]
                  [--allowed-headers ALLOWED_HEADERS] [--api-key API_KEY]
                  [--lora-modules LORA_MODULES [LORA_MODULES ...]]
                  [--prompt-adapters PROMPT_ADAPTERS [PROMPT_ADAPTERS ...]]
                  [--chat-template CHAT_TEMPLATE]
                  [--response-role RESPONSE_ROLE] [--ssl-keyfile SSL_KEYFILE]
                  [--ssl-certfile SSL_CERTFILE] [--ssl-ca-certs SSL_CA_CERTS]
                  [--ssl-cert-reqs SSL_CERT_REQS] [--root-path ROOT_PATH]
                  [--middleware MIDDLEWARE] [--return-tokens-as-token-ids]
                  [--disable-frontend-multiprocessing]
                  [--enable-auto-tool-choice]
                  [--tool-call-parser {granite-20b-fc,granite,hermes,internlm,jamba,llama3_j
                  [--tool-parser-plugin TOOL_PARSER_PLUGIN] [--model MODEL]
                  [--task {auto,generate,embedding}] [--tokenizer TOKENIZER]
                  [--skip-tokenizer-init] [--revision REVISION]
                  [--code-revision CODE_REVISION]
                  [--tokenizer-revision TOKENIZER_REVISION]
                  [--tokenizer-mode {auto,slow,mistral}]
                  [--chat-template-text-format {string,openai}]
                  [--trust-remote-code]
                  [--allowed-local-media-path ALLOWED_LOCAL_MEDIA_PATH]
                  [--download-dir DOWNLOAD_DIR]
                  [--load-format {auto,pt,safetensors,npcache,dummy,tensorizer,sharded_state
                  [--config-format {auto,hf,mistral}]
                  [--dtype {auto,half,float16,bfloat16,float,float32}]
                  [--kv-cache-dtype {auto,fp8,fp8_e5m2,fp8_e4m3}]
                  [--quantization-param-path QUANTIZATION_PARAM_PATH]
                  [--max-model-len MAX_MODEL_LEN]
                  [--guided-decoding-backend {outlines,lm-format-enforcer}]
                  [--distributed-executor-backend {ray,mp}] [--worker-use-ray]
                  [--pipeline-parallel-size PIPELINE_PARALLEL_SIZE]
                  [--tensor-parallel-size TENSOR_PARALLEL_SIZE]
                  [--max-parallel-loading-workers MAX_PARALLEL_LOADING_WORKERS]
                  [--ray-workers-use-nsight] [--block-size {8,16,32,64,128}]
                  [--enable-prefix-caching] [--disable-sliding-window]
                  [--use-v2-block-manager]
                  [--num-lookahead-slots NUM_LOOKAHEAD_SLOTS] [--seed SEED]
                  [--swap-space SWAP_SPACE] [--cpu-offload-gb CPU_OFFLOAD_GB]
                  [--gpu-memory-utilization GPU_MEMORY_UTILIZATION]
                  [--num-gpu-blocks-override NUM_GPU_BLOCKS_OVERRIDE]
                  [--max-num-batched-tokens MAX_NUM_BATCHED_TOKENS]
                  [--max-num-seqs MAX_NUM_SEQS] [--max-logprobs MAX_LOGPROBS]
                  [--disable-log-stats]
                  [--quantization {aqlm,awq,deepspeedfp,tpu_int8,fp8,fbgemm_fp8,modelopt,mar
                  [--rope-scaling ROPE_SCALING] [--rope-theta ROPE_THETA]
                  [--hf-overrides HF_OVERRIDES] [--enforce-eager]
                  [--max-seq-len-to-capture MAX_SEQ_LEN_TO_CAPTURE]
                  [--disable-custom-all-reduce]
                  [--tokenizer-pool-size TOKENIZER_POOL_SIZE]
                  [--tokenizer-pool-type TOKENIZER_POOL_TYPE]
                  [--tokenizer-pool-extra-config TOKENIZER_POOL_EXTRA_CONFIG]
                  [--limit-mm-per-prompt LIMIT_MM_PER_PROMPT]
                  [--mm-processor-kwargs MM_PROCESSOR_KWARGS] [--enable-lora]
                  [--max-loras MAX_LORAS] [--max-lora-rank MAX_LORA_RANK]
                  [--lora-extra-vocab-size LORA_EXTRA_VOCAB_SIZE]
                  [--lora-dtype {auto,float16,bfloat16,float32}]
                  [--long-lora-scaling-factors LONG_LORA_SCALING_FACTORS]
                  [--max-cpu-loras MAX_CPU_LORAS] [--fully-sharded-loras]
                  [--enable-prompt-adapter]
                  [--max-prompt-adapters MAX_PROMPT_ADAPTERS]
                  [--max-prompt-adapter-token MAX_PROMPT_ADAPTER_TOKEN]
                  [--device {auto,cuda,neuron,cpu,openvino,tpu,xpu,hpu}]
                  [--num-scheduler-steps NUM_SCHEDULER_STEPS]
                  [--multi-step-stream-outputs [MULTI_STEP_STREAM_OUTPUTS]]
                  [--scheduler-delay-factor SCHEDULER_DELAY_FACTOR]
                  [--enable-chunked-prefill [ENABLE_CHUNKED_PREFILL]]
                  [--speculative-model SPECULATIVE_MODEL]
                  [--speculative-model-quantization {aqlm,awq,deepspeedfp,tpu_int8,fp8,fbgem
                  [--num-speculative-tokens NUM_SPECULATIVE_TOKENS]
                  [--speculative-disable-mqa-scorer]
                  [--speculative-draft-tensor-parallel-size SPECULATIVE_DRAFT_TENSOR_PARALLE
                  [--speculative-max-model-len SPECULATIVE_MAX_MODEL_LEN]
                  [--speculative-disable-by-batch-size SPECULATIVE_DISABLE_BY_BATCH_SIZE]
                  [--ngram-prompt-lookup-max NGRAM_PROMPT_LOOKUP_MAX]
                  [--ngram-prompt-lookup-min NGRAM_PROMPT_LOOKUP_MIN]
                  [--spec-decoding-acceptance-method {rejection_sampler,typical_acceptance_s
                  [--typical-acceptance-sampler-posterior-threshold TYPICAL_ACCEPTANCE_SAMPL
                  [--typical-acceptance-sampler-posterior-alpha TYPICAL_ACCEPTANCE_SAMPLER_P
                  [--disable-logprobs-during-spec-decoding [DISABLE_LOGPROBS_DURING_SPEC_DEC
                  [--model-loader-extra-config MODEL_LOADER_EXTRA_CONFIG]
                  [--ignore-patterns IGNORE_PATTERNS]
                  [--preemption-mode PREEMPTION_MODE]
                  [--served-model-name SERVED_MODEL_NAME [SERVED_MODEL_NAME ...]]
                  [--qlora-adapter-name-or-path QLORA_ADAPTER_NAME_OR_PATH]
                  [--otlp-traces-endpoint OTLP_TRACES_ENDPOINT]
                  [--collect-detailed-traces COLLECT_DETAILED_TRACES]
                  [--disable-async-output-proc]
                  [--override-neuron-config OVERRIDE_NEURON_CONFIG]
                  [--scheduling-policy {fcfs,priority}]
                  [--pooling-type {LAST,ALL,CLS,STEP,MEAN}] [--pooling-norm]
                  [--no-pooling-norm] [--pooling-softmax]
                  [--no-pooling-softmax]
                  [--pooling-step-tag-id POOLING_STEP_TAG_ID]
                  [--pooling-returned-token-ids POOLING_RETURNED_TOKEN_IDS [POOLING_RETURNED
                  [--disable-log-requests] [--max-log-len MAX_LOG_LEN]
                  [--disable-fastapi-docs]
```

But it doesn't seem to help ...

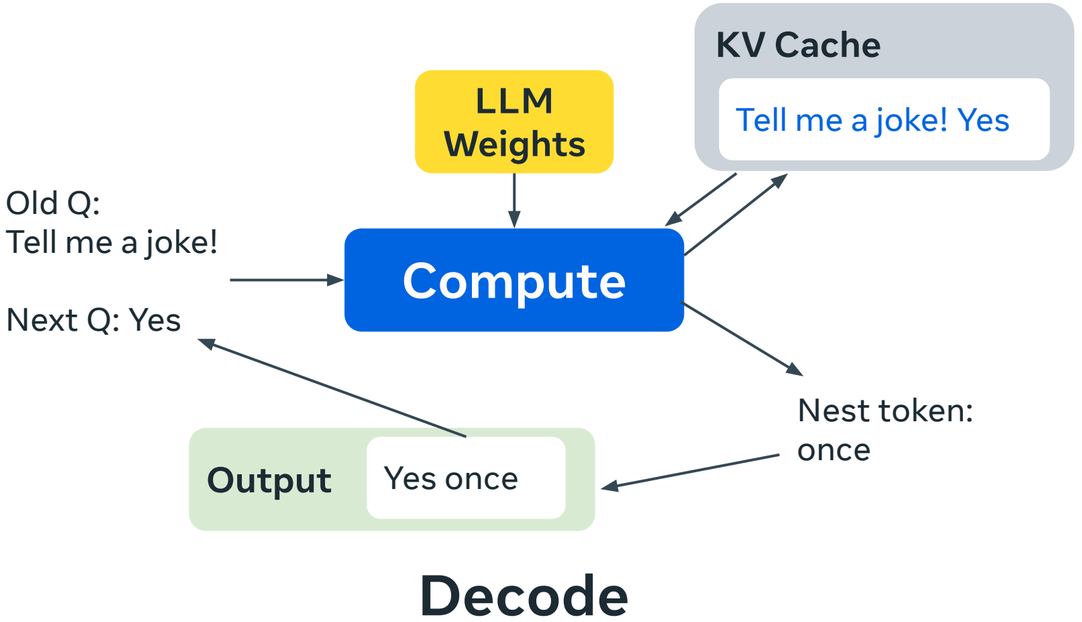


credit: https://docs.vllm.ai/en/latest/serving/openai_compatible_server.html#command-line-arguments-for-the-server

Why is That?... Let's Understand System Bottleneck a Bit Deeper..



Prefill



Decode

Prefill = read O(10GB) weight, compute O(1) - O(100) TFLOPs!

Decode = read O(10GB) weight, compute O(0.01) - O(0.1) TFLOPs for O(10)-O(100) times! And generate O(100KB) KV cache per token.

GPU COMPUTE

MEMORY BANDWIDTH

MEMORY CAPACITY

Make LLM faster

= Fit LLM operations within system resources

= Fit GPU compute + fit memory bandwidth + fit memory capacity

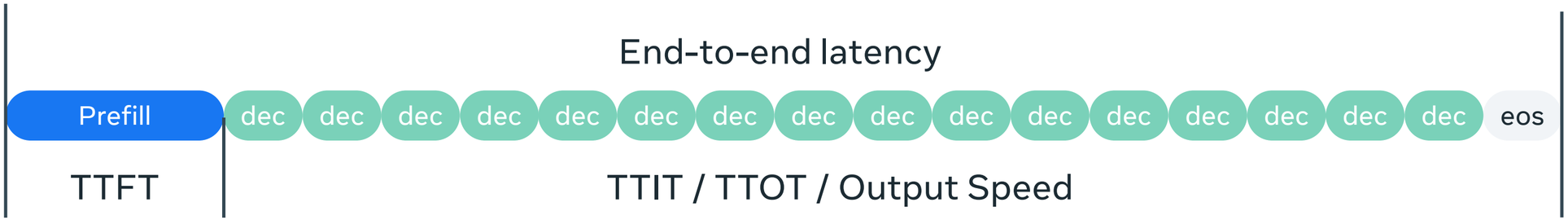
Prefill | Decode | KV Cache

Which scales ~ model sizes, sequence length, batch size

But each GPU has fixed configuration for system resource ratio

🥹

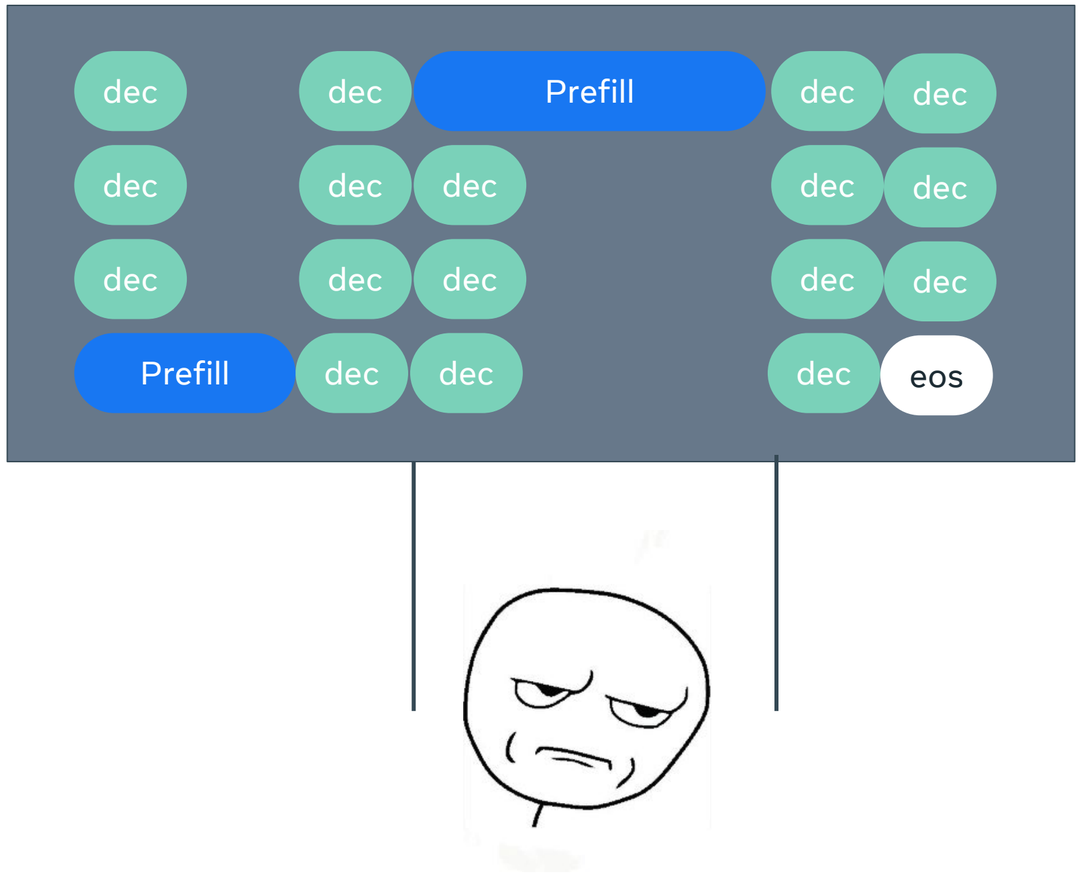
What Type of Slowness are You Talking About...



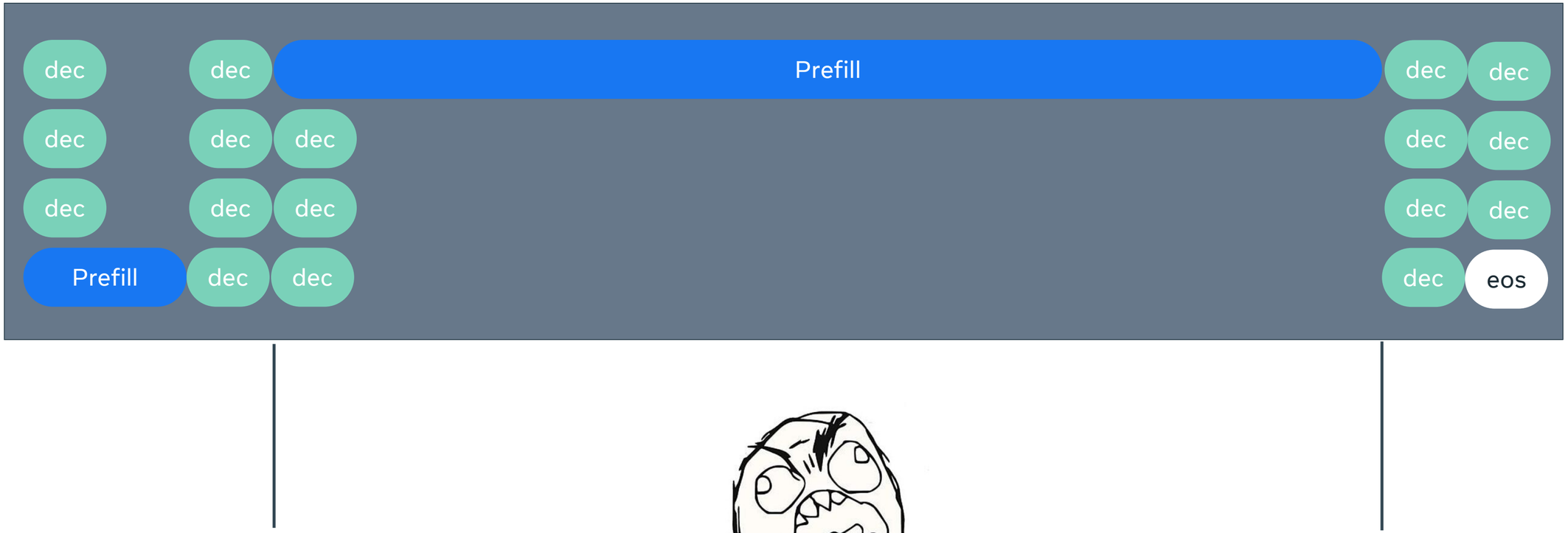
Understand what's reasonable expectation. Look at public benchmark data by

- Model Size
- Input Length
- Output Length
- Hardware Choice
- Cost
- Quality Trade-off

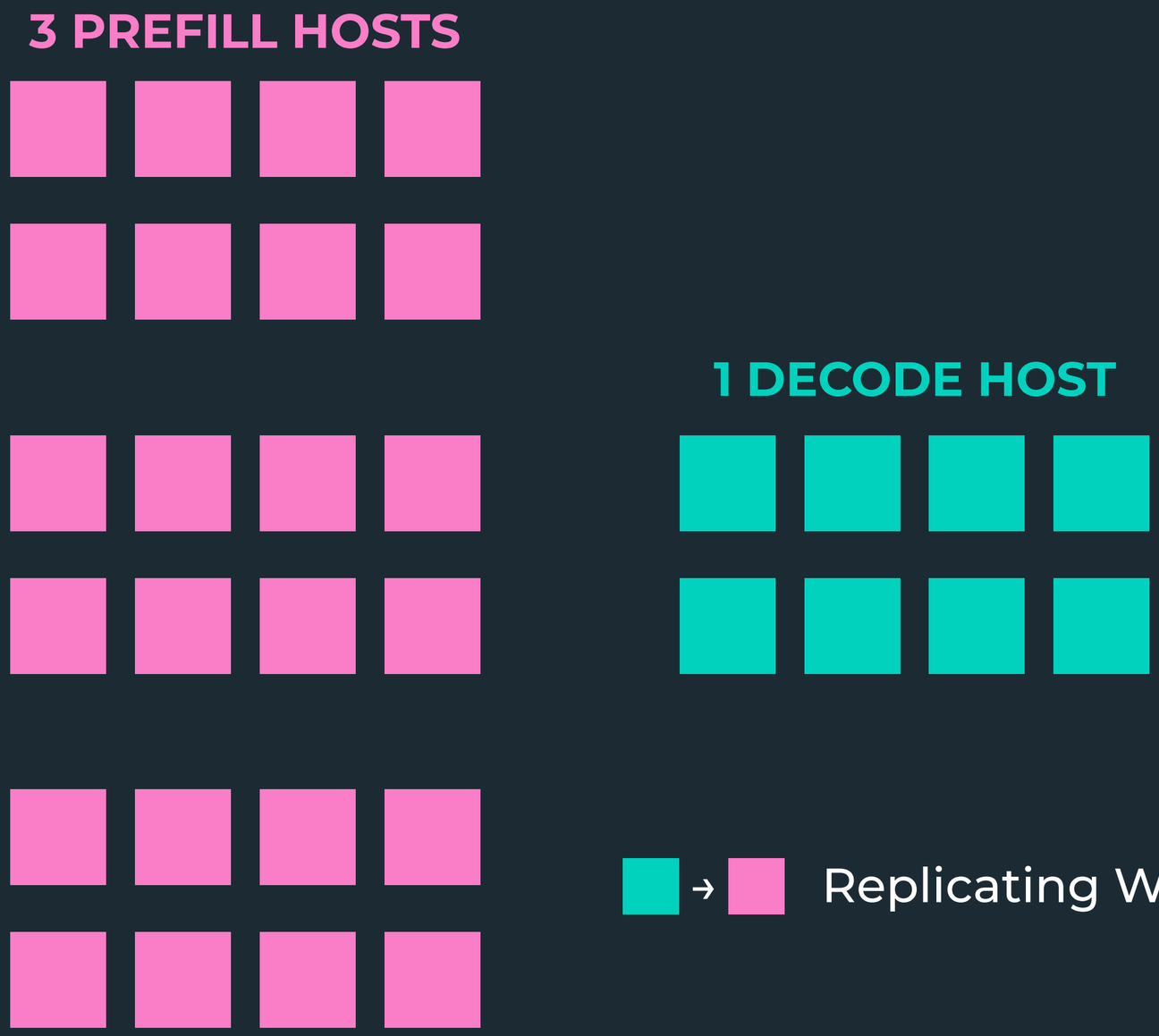
You are unhappy with average generation speed



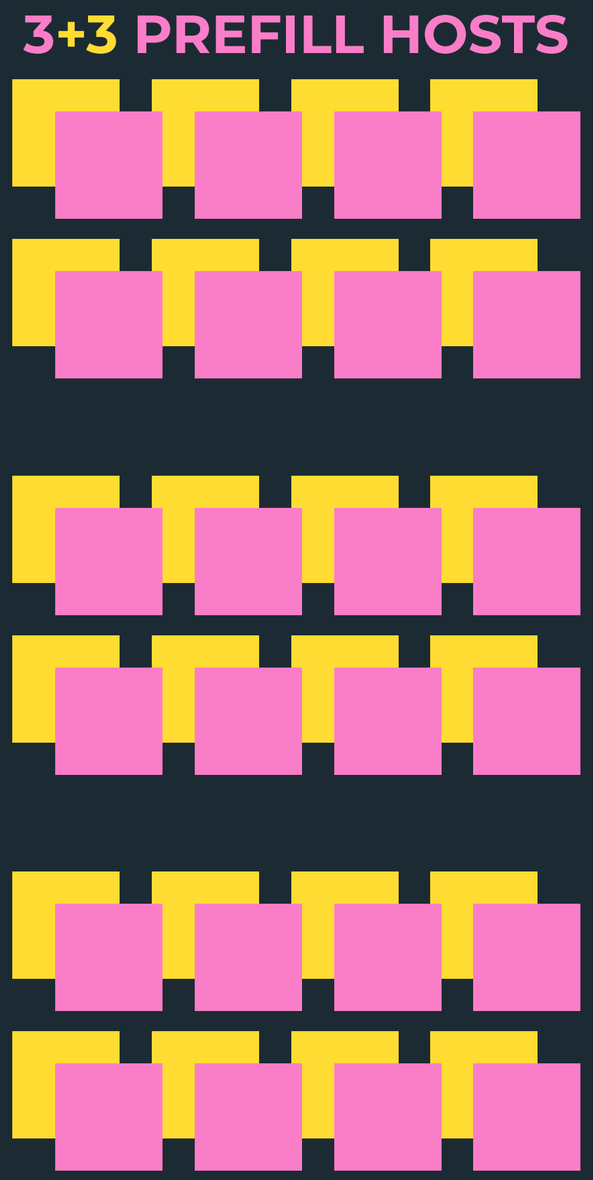
As context window gets longer

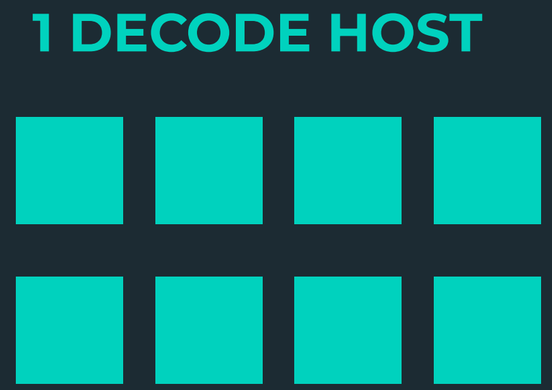


Disaggregated prefill/decode improve latency-bound throughput and reduce tail decode latency



1 min for 128K input
without parallelism






Replication


PARTITION

STEP 2 Make Your Problems Smaller

$$\text{Problem Size} \sim \frac{\text{Tokens}}{\text{Query}} \times \frac{\text{FLOPs}}{\text{Token}} \times \frac{\text{FLOPS}}{\text{Hardware}}$$

Shorter Prompts

Be a good and concise communicator to LLM

Smaller Models

Fine-tuning small generalist

Distillation / pruning

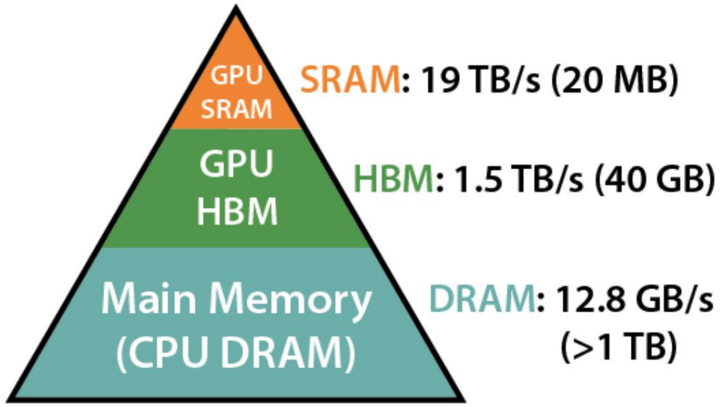
Quantization

Many components to choose (FFN / KV / ATTN)

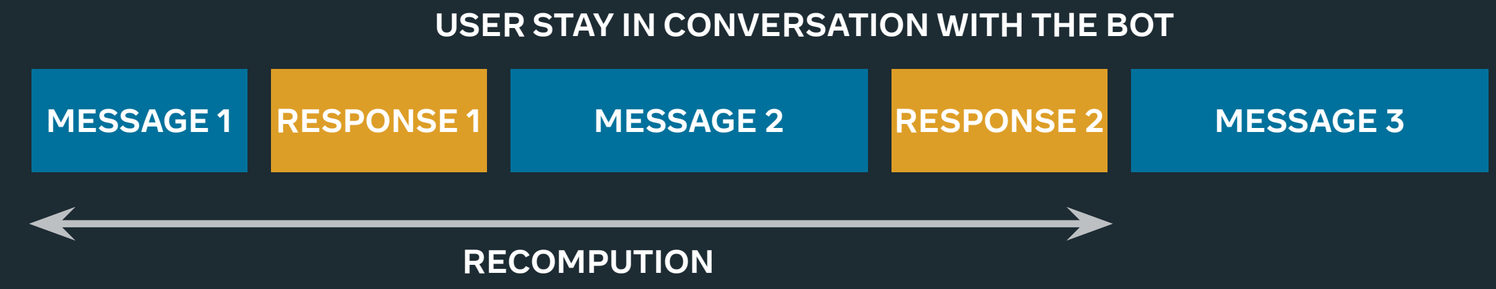
Different data types (INT8 / FP8 / INT4 / NF4)

Different policies to choose (W8A8 / W4A16 / tensor-wise / row-wise / group-wise)

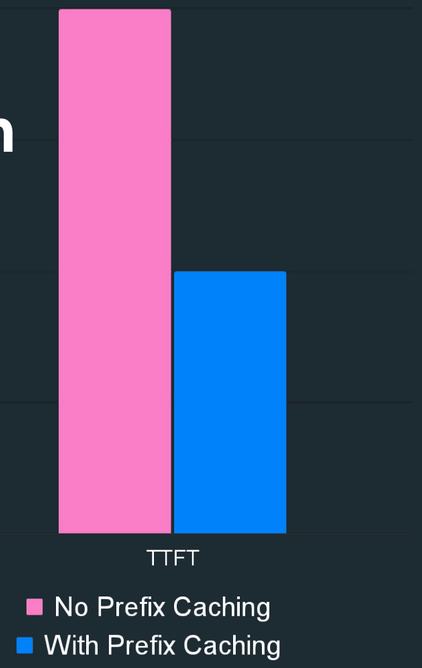
STEP 3 Prefix Caching



Memory Hierarchy with Bandwidth & Memory Size



-50% is common



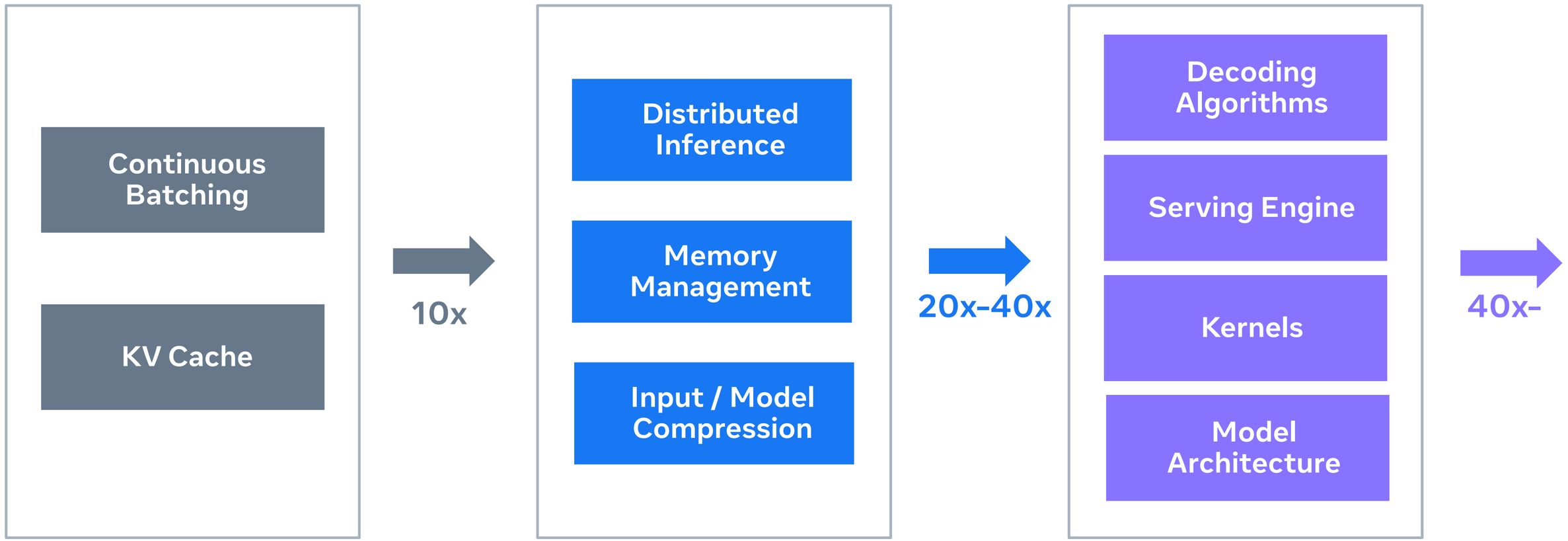
Credit: https://arxiv.org/abs/2205.14135

Typically ...

Technique	TTFT	TTIT	$	Quality
Speculative Decoding	😶	🤗	😨–😶	😶
Chunked Prefill	😨	🙂	😶	😶
Attention Kernels	🙂–🤗	😶–🙂	🙂	😶
Token Sparsity	🤗	🤗	🤗	😨

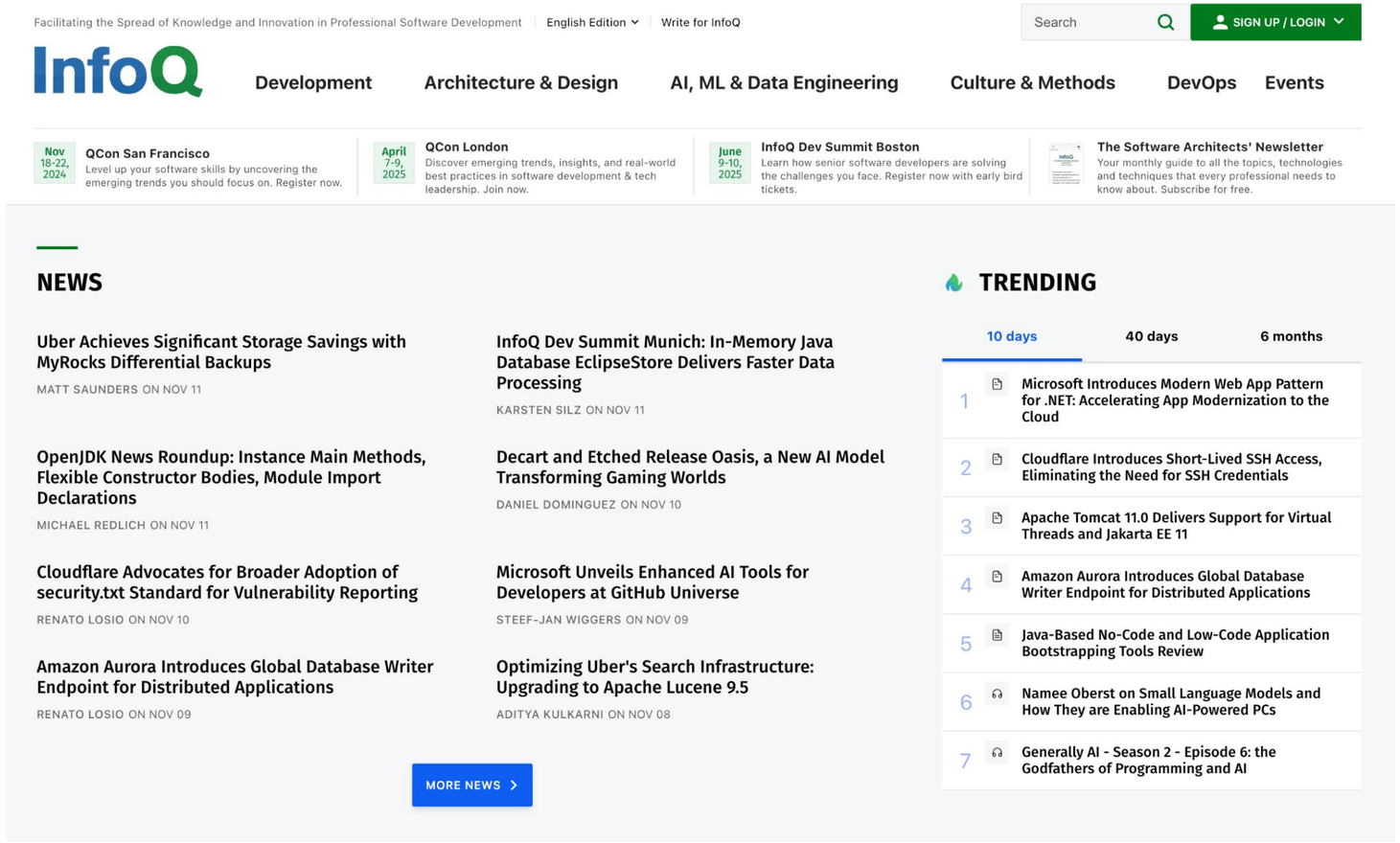
for your unique workload requirement,
every 🤗🙂😨😶 may be flipped

What We've Learned So Far



“Did you do all of these at Meta?”

Let’s Build This Step By Step (Part 3!)



Challenge 1
Fitting

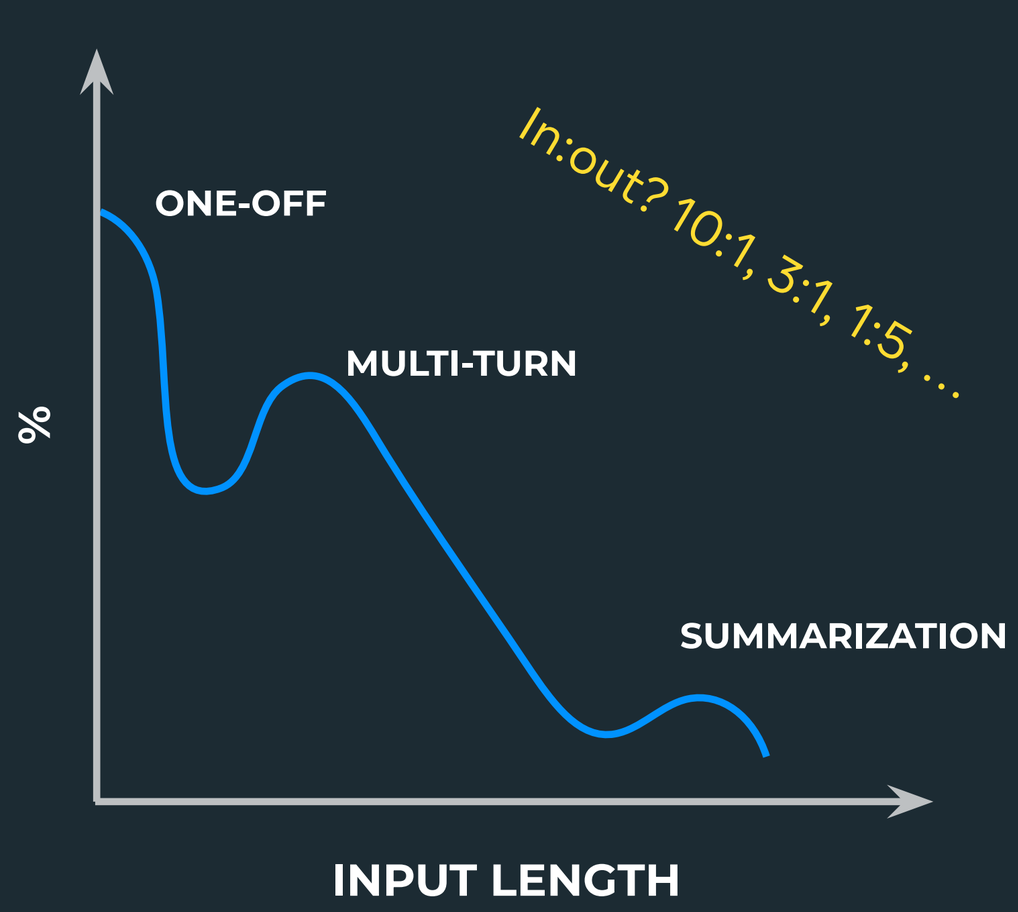
Challenge 2
It’s Slow

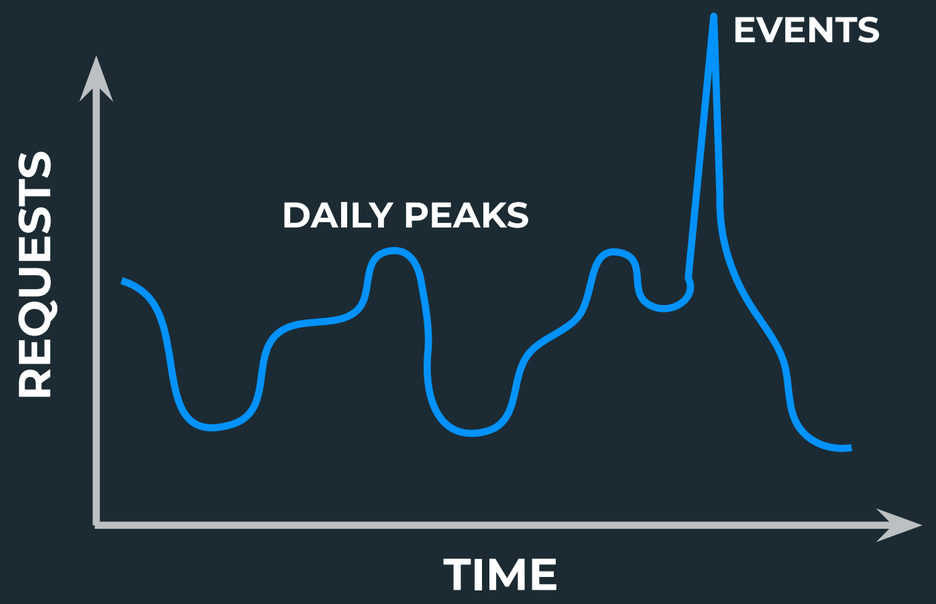
Challenge 3
Production

Challenge 4


THIS IS FINE.

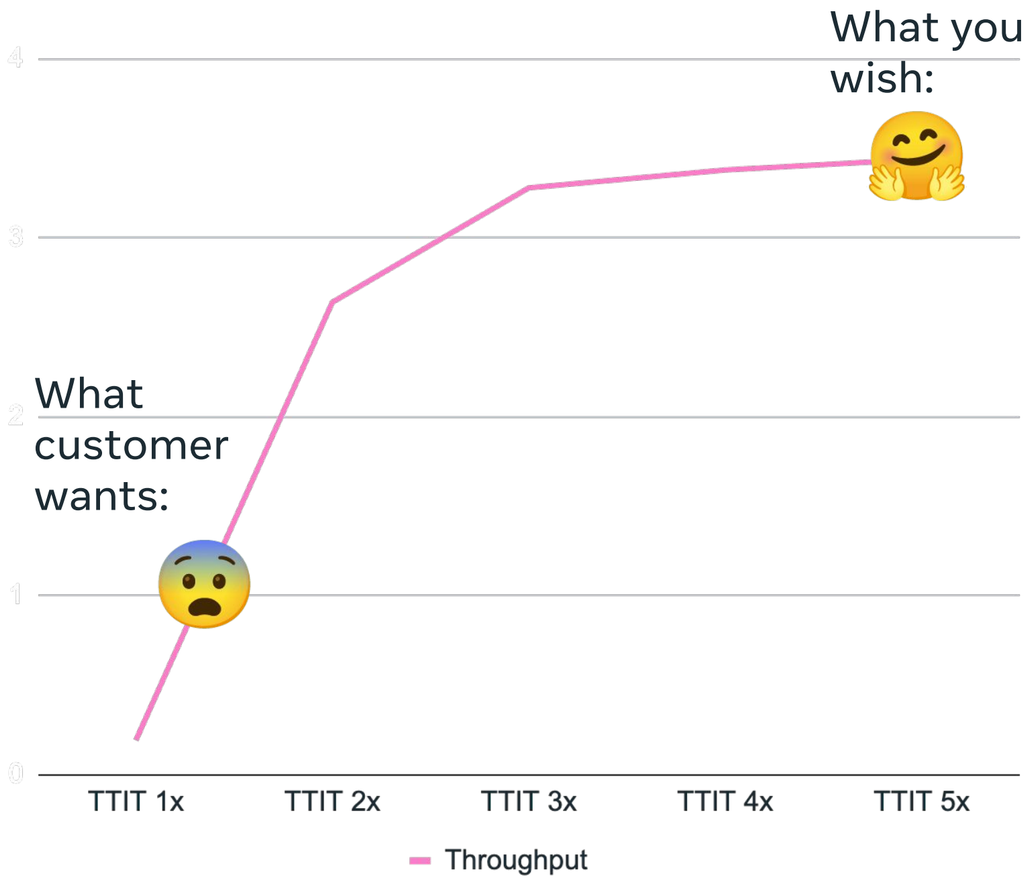
Input length and traffic change over time



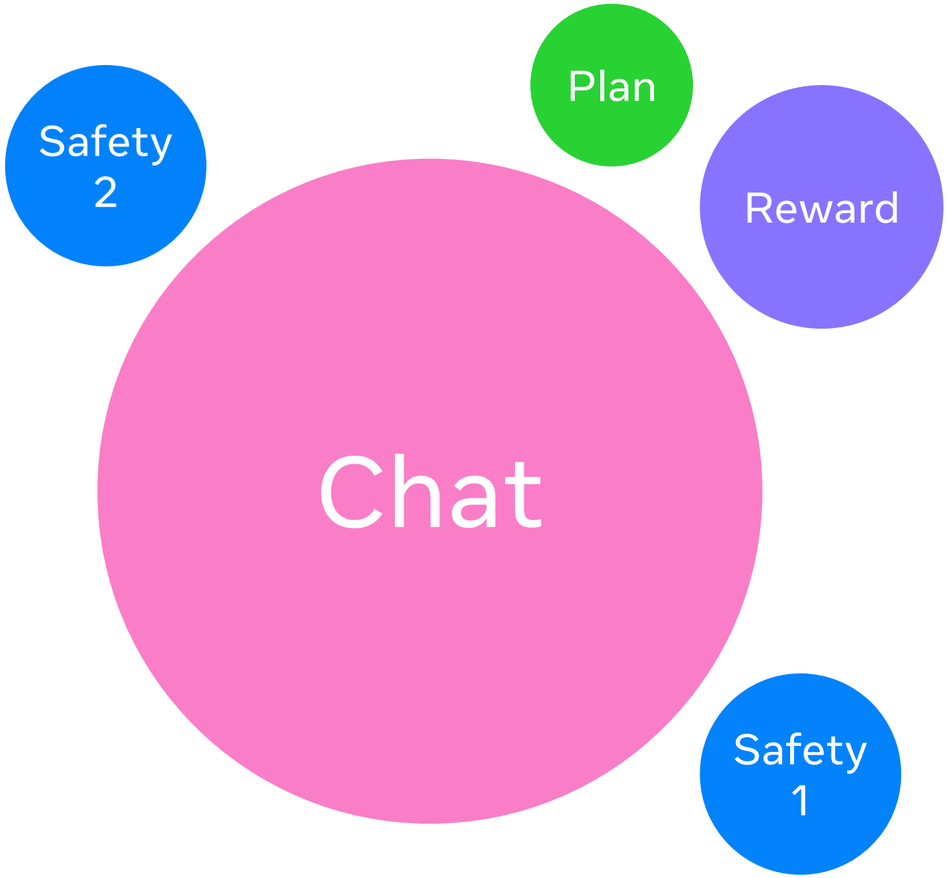


What about annotation, summarization, feature generation?

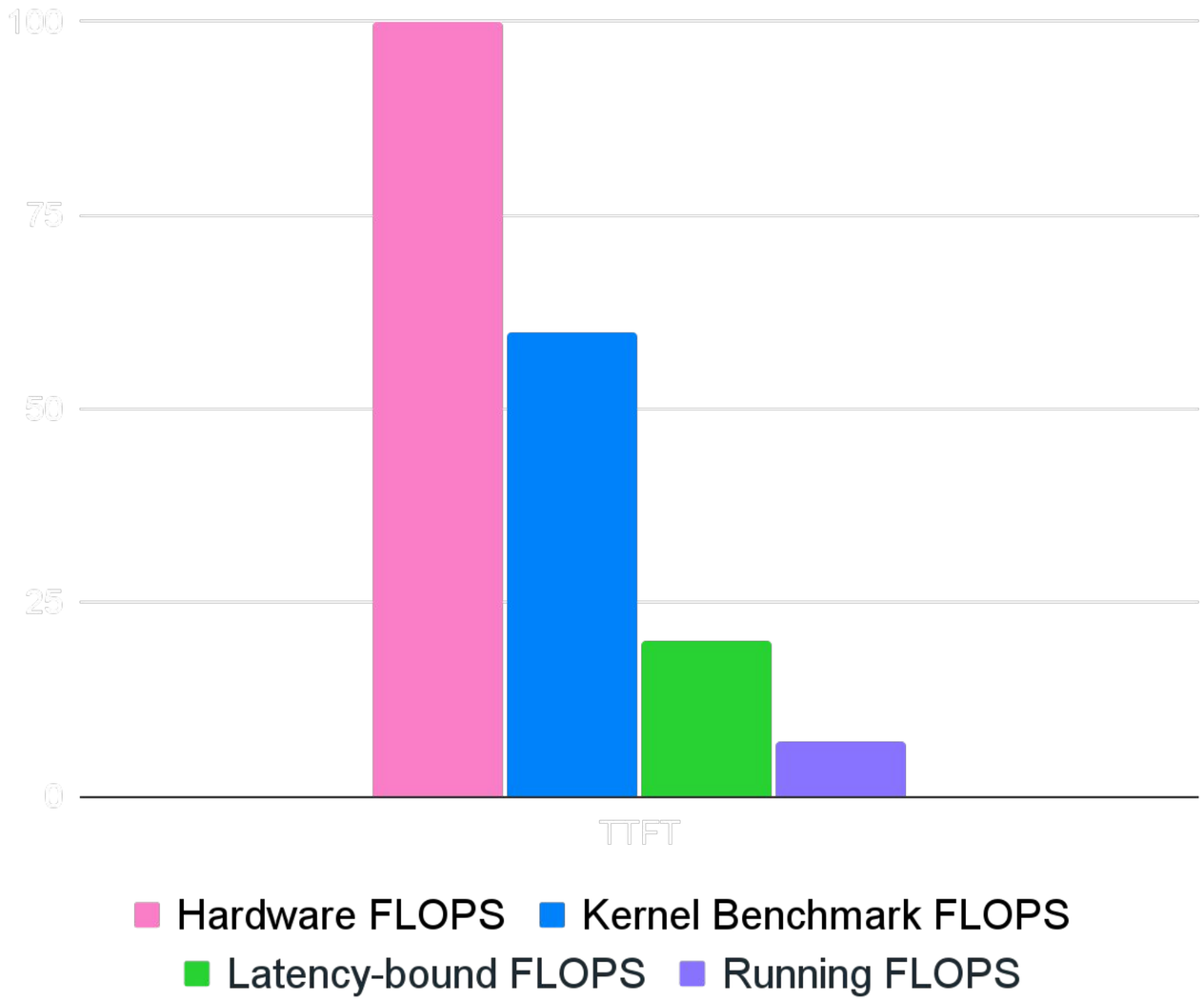
Customer SLOs



More Than 1 Model

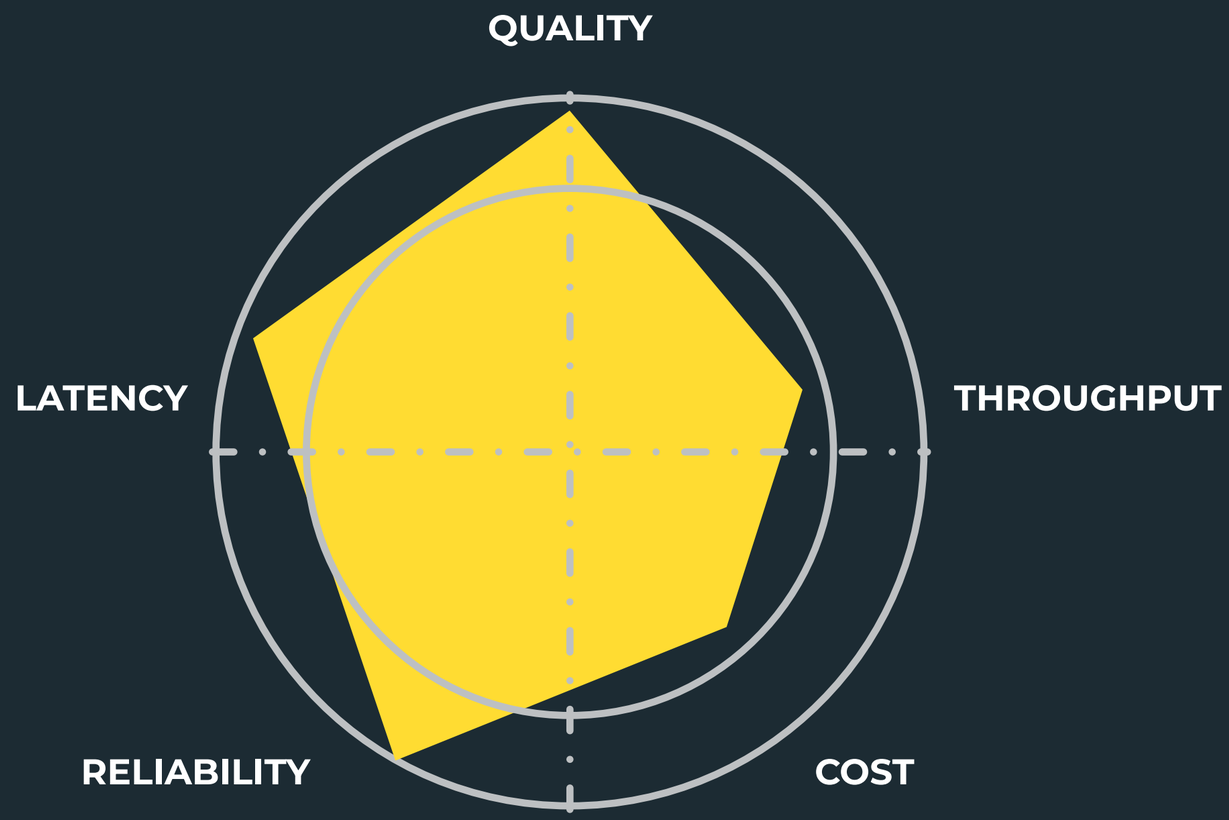


The Hard Lessons from Reality



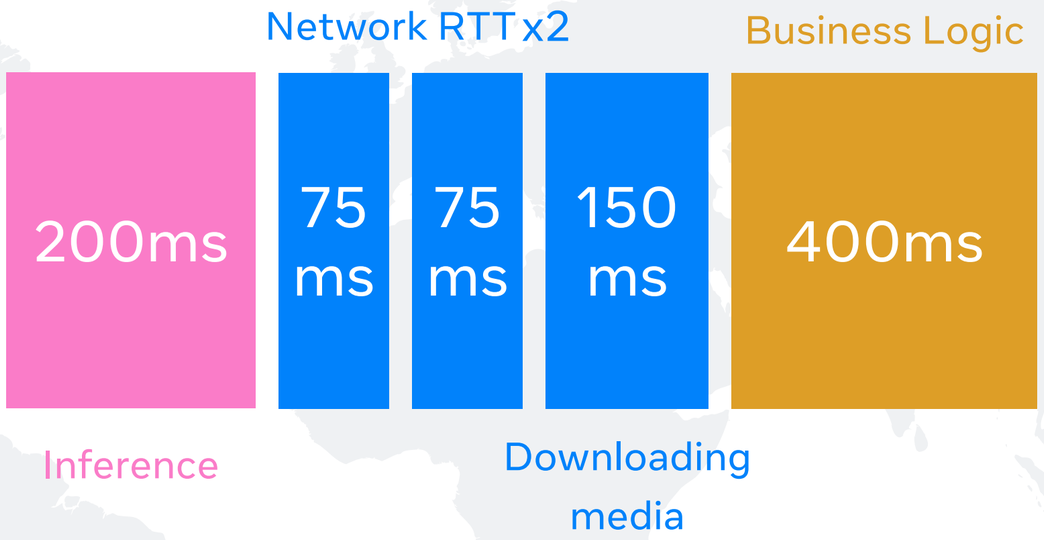
Trade-offs!

- Reliability:
  - Understand the cost of different 9s
- Latency: average adult
  - Reads English @ **200-300** words per minute
  - Speaks English @ **150-200** words per minute



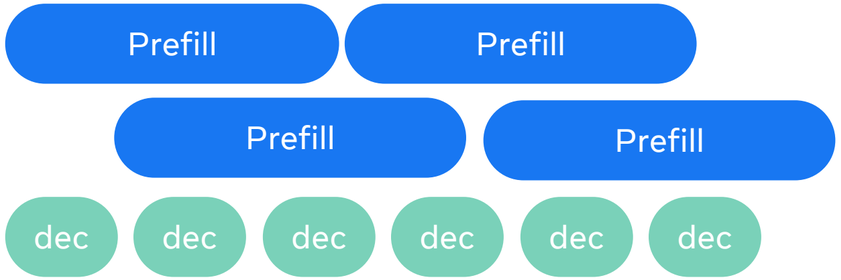
credit:
https://www.sciencedirect.com/science/article/pii/S0749596X19300786
https://virtualspeech.com/blog/average-speaking-rate-words-per-minute

STEP 2: Look at E2E Latency - Network

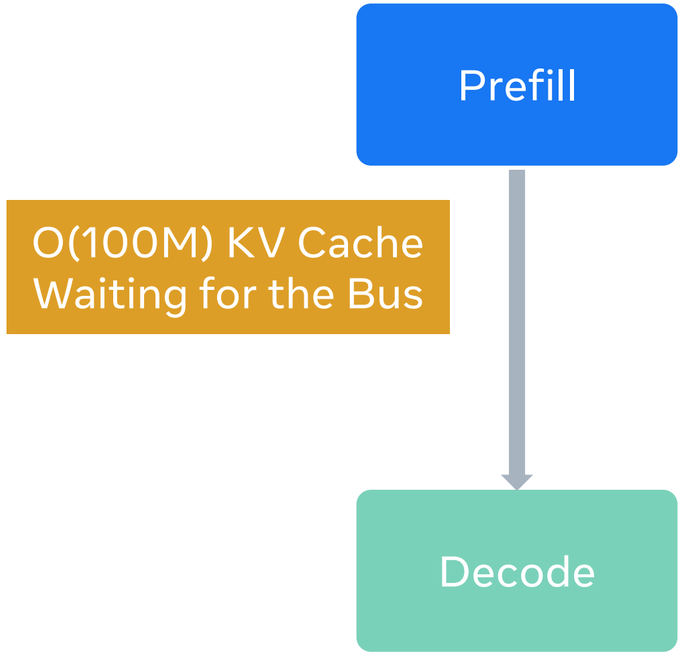


Note: This data is made up and does not represent real latency at Meta.

YES,



BUT,

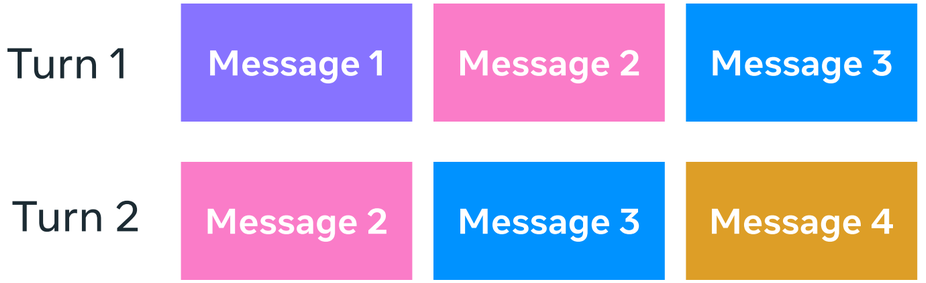


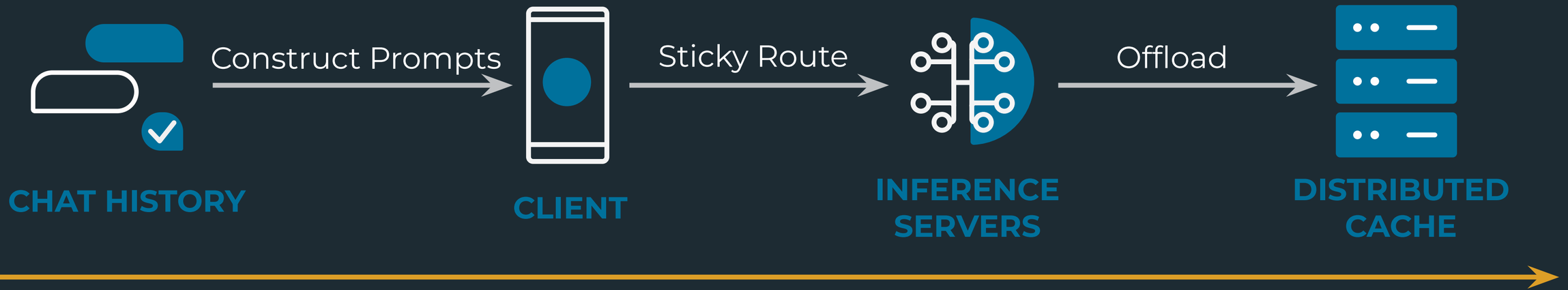
Compute & transfer overlap, system scheduling optimization

STEP 3: Look at E2E Caching – Prefix Caching

YES,

BUT,





STEP 3: Look at E2E Quality - Quantization

YES,

MMLU bf16

MMLU fp8

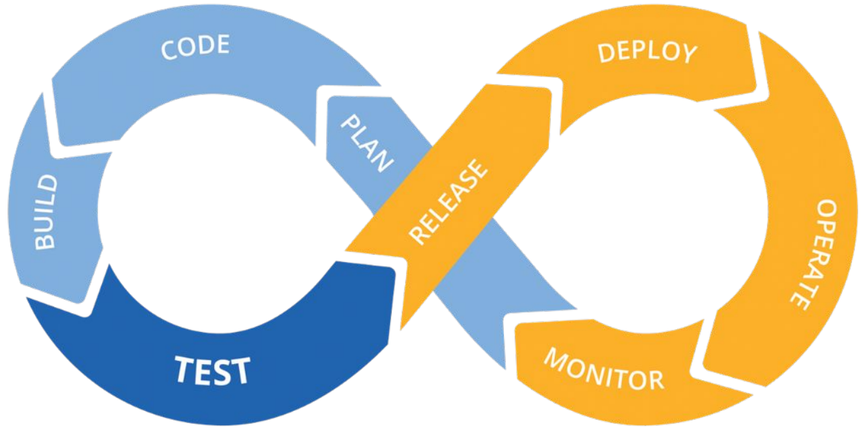
BUT,

Build you own product eval

STEP 3: Look at E2E Quality - CI/CD

Category Benchmark	Llama 3.1 405B	Llama 3.1 70B	Llama 3.1 8B
General MMLU (0-shot, CoT)	88.6	86.0	73.0
MMLU PRO (5-shot, CoT)	73.3	66.4	48.3
IFEval	88.6	87.5	80.4
Code HumanEval (0-shot)	89.0	80.5	72.6
MBPP EvalPlus (base) (0-shot)	88.6	86.0	72.8
Math GSM8K (8-shot, CoT)	96.8	95.1	84.5
MATH (0-shot, CoT)	73.8	68.0	51.9
Reasoning ARC Challenge (0-shot)	96.9	94.8	83.4
GPQA (0-shot, CoT)	51.1	46.7	32.8
Tool use BFCL	88.5	84.8	76.1
Nexus	58.7	56.7	38.5
Long context ZeroSCROLLS/QuALITY	95.2	90.5	81.0
InfiniteBench/En.MC	83.4	78.2	65.1
NIH/Multi-needle	98.1	97.5	98.8
Multilingual Multilingual MGSM (0-shot)	91.6	86.9	68.9

credit: https://ai.meta.com/blog/meta-llama-3-1/



Credit: https://www.mabl.com/blog/what-is-cicd

Takeaways

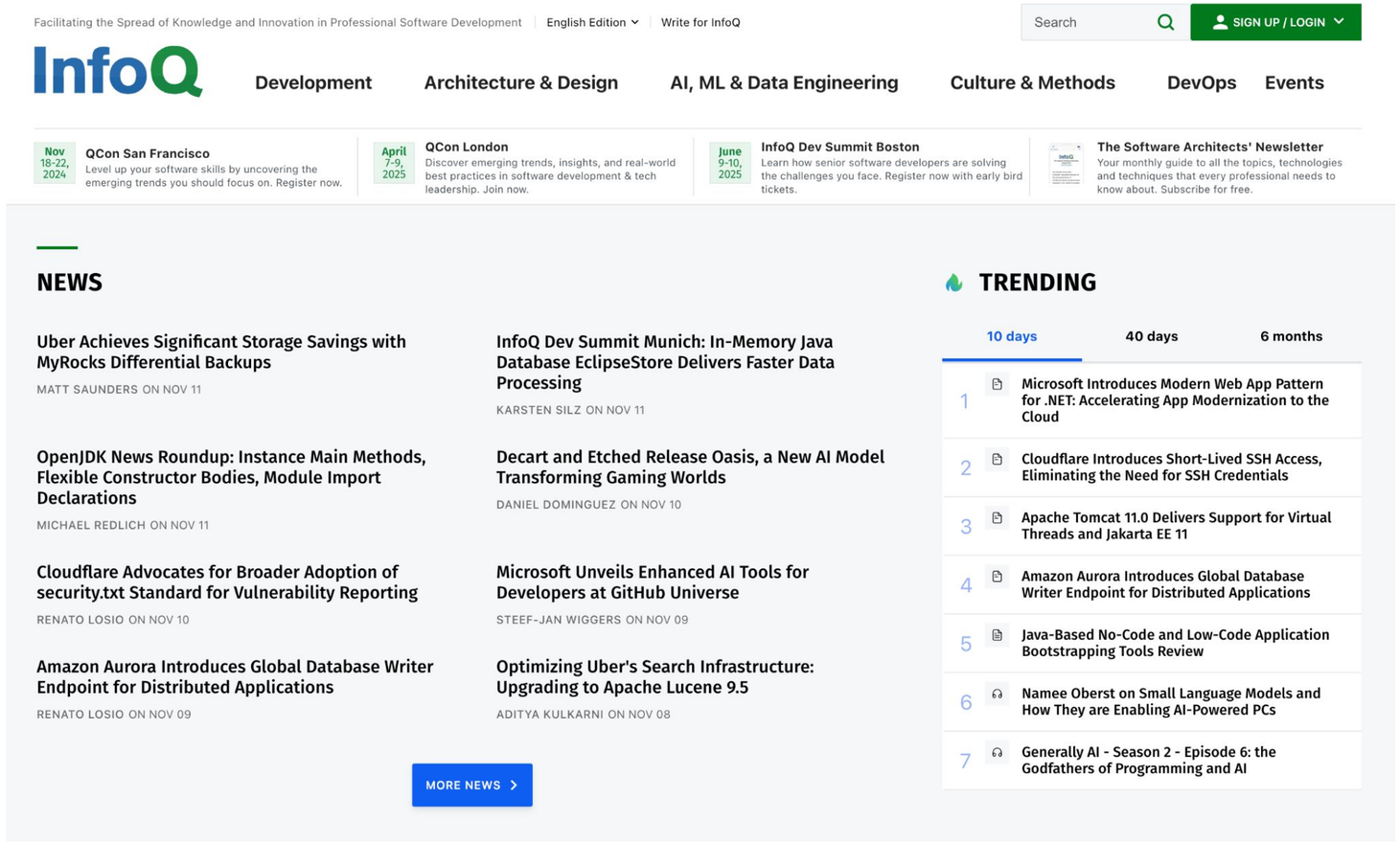
We lose theoretical performance in production environment A LOT

Non-inference latency is substantial

Infra x product co-design to optimize effective cache hit rate

Continuously test acceleration techniques using product signals

Let’s Build This Step By Step (Part 4!)



Challenge 1
Fitting

Challenge 2
It’s Slow

Challenge 3
Production

Challenge 4
Scaling

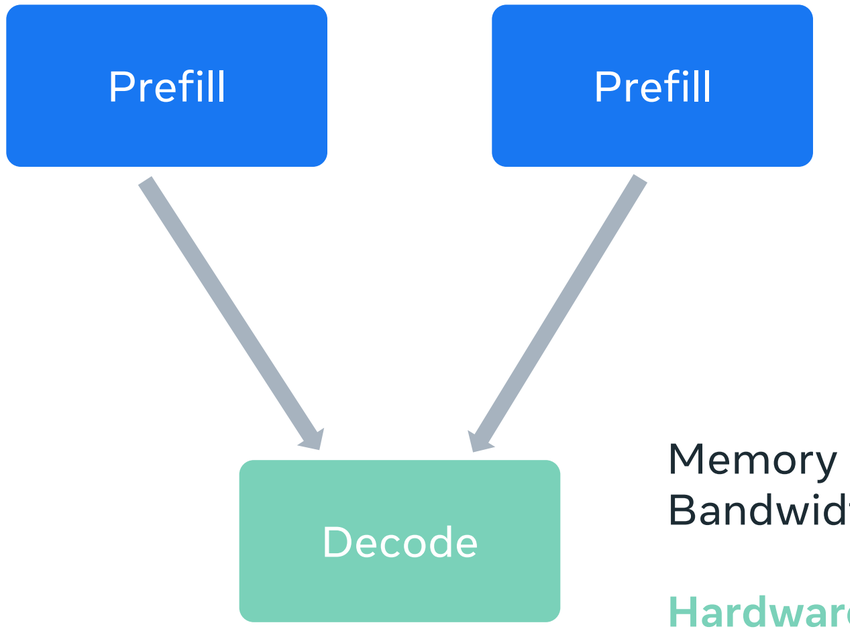
Let’s build a rocket and make it fly!

What Will Scale?

- Number of Deployments
- Number of GPUs
- Number of Developers
- Number of Models

STEP 1: Scale Number of Deployments – Allocation at Scale (Physical Availability)

YES,



Compute Heavy?

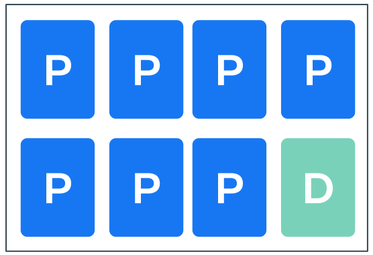
Hardware P with
Higher Peak FLOPS

Memory
Bandwidth Heavy?

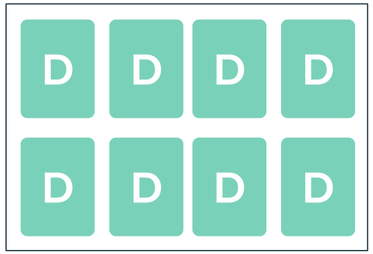
Hardware D w/
Higher Peak GB/s

BUT,

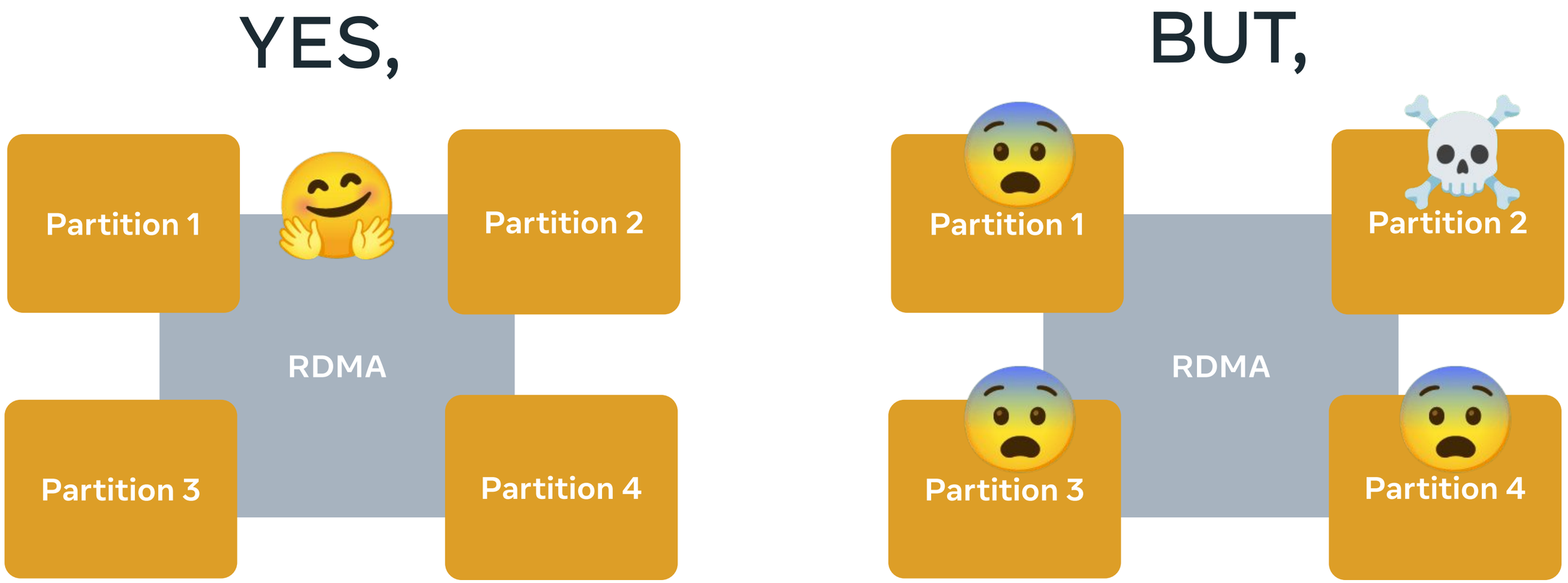
Region 1



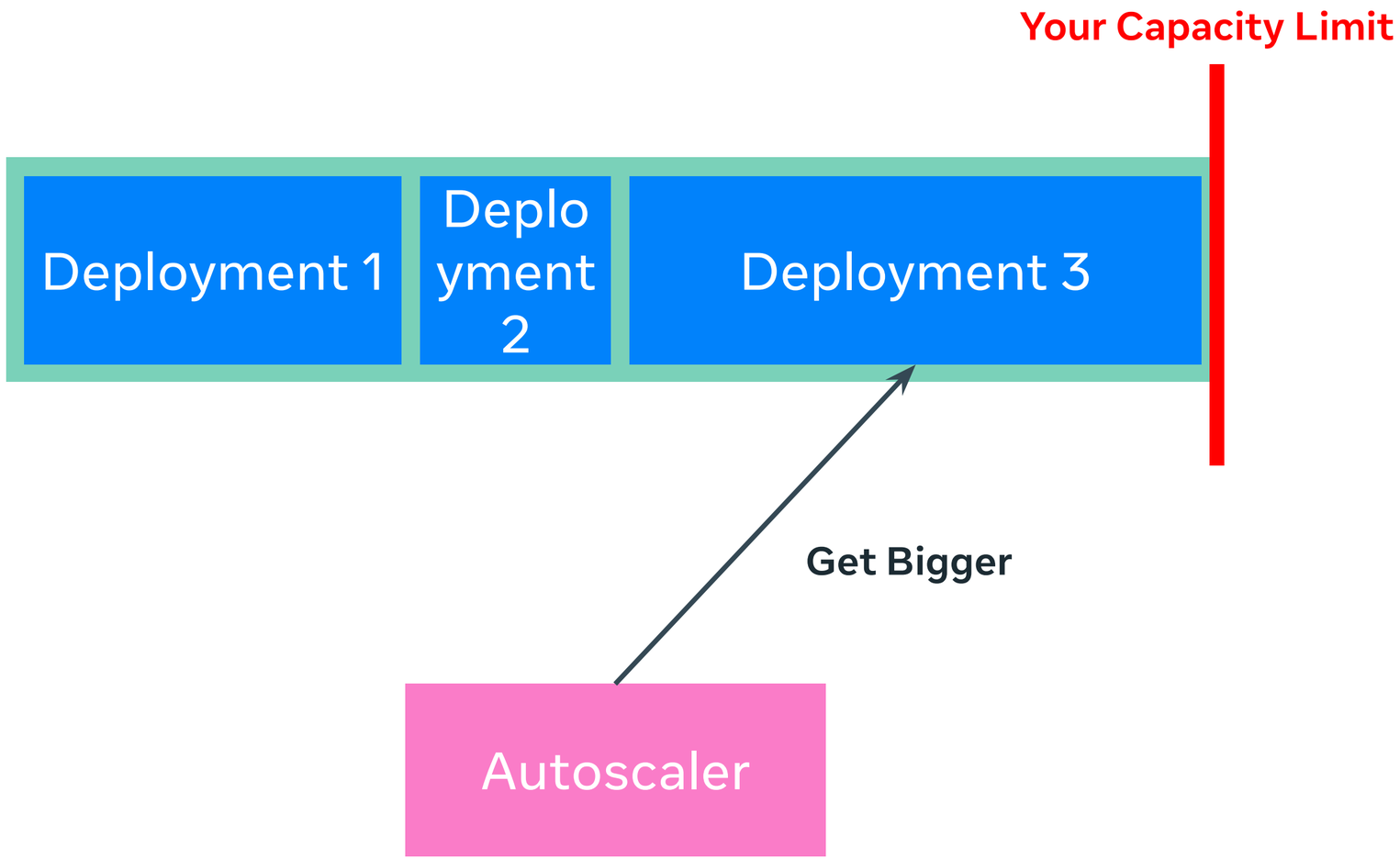
Region 2



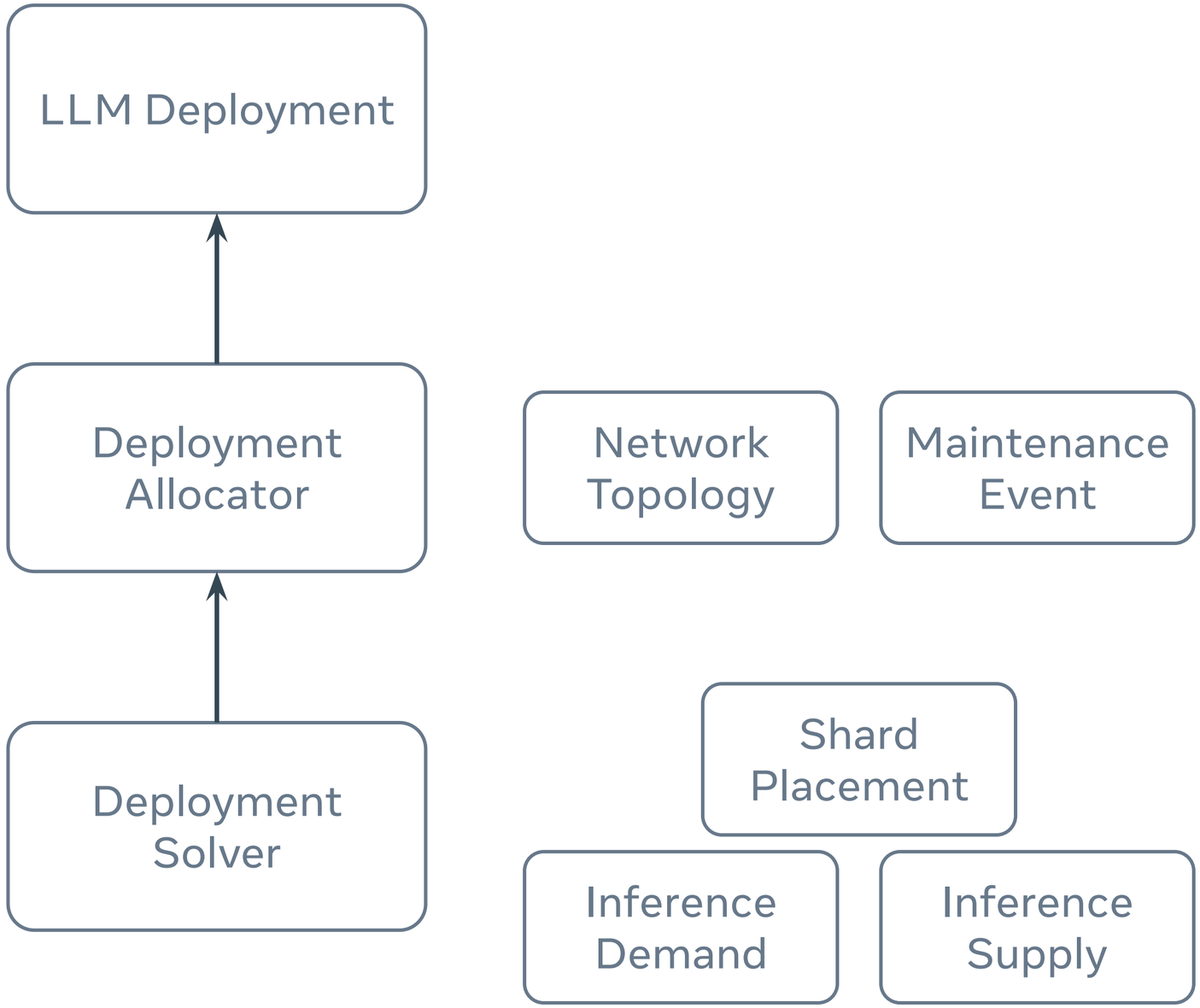
STEP 1: Scale Number of Deployments – Allocation at Scale (Shared Fate)

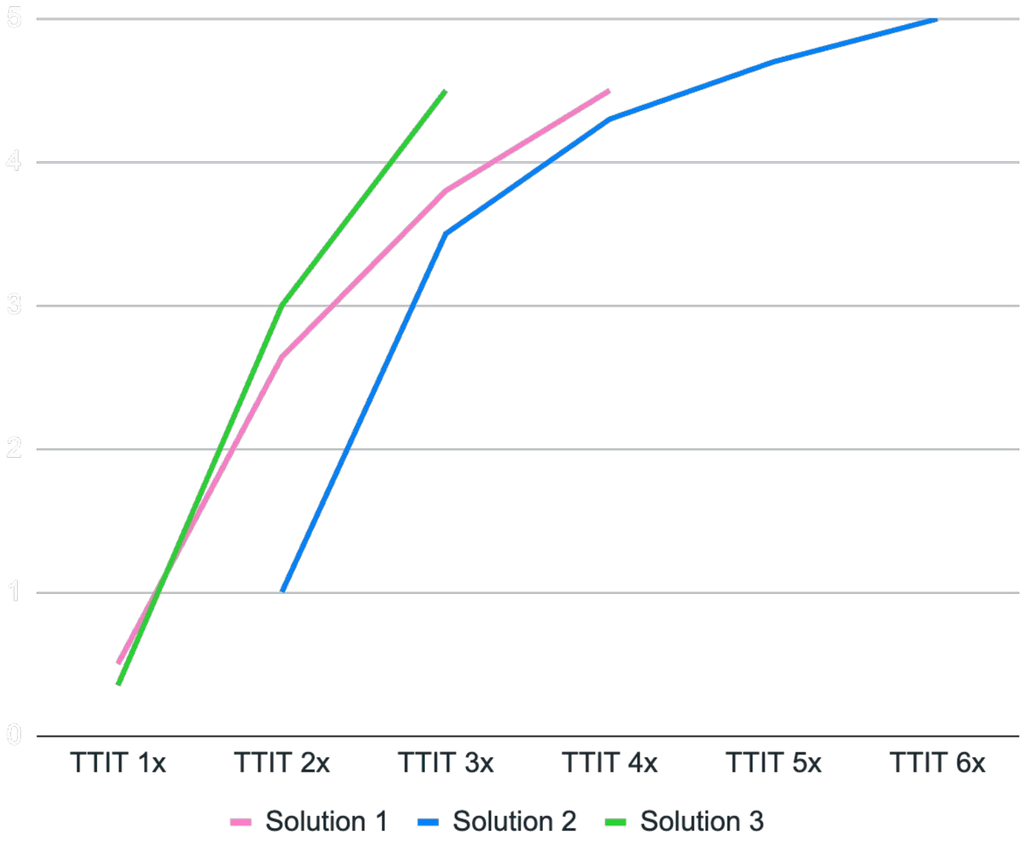


STEP 1: Scale Number of Deployments – Allocation at Scale (Capacity Constraints)



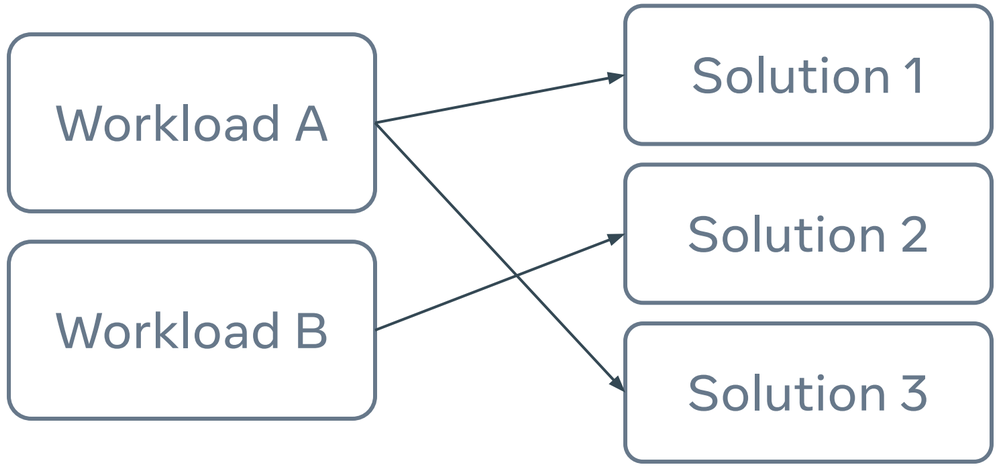
STEP 1: Scale Number of Deployments – Allocation at Scale



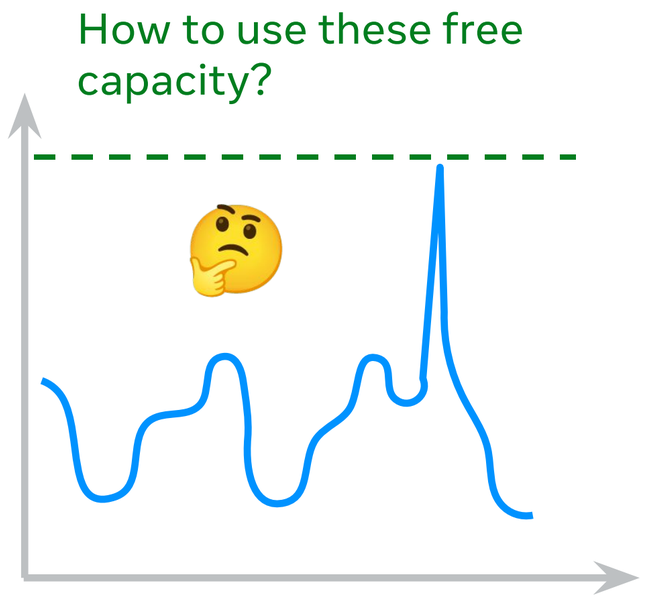


Inference Menu

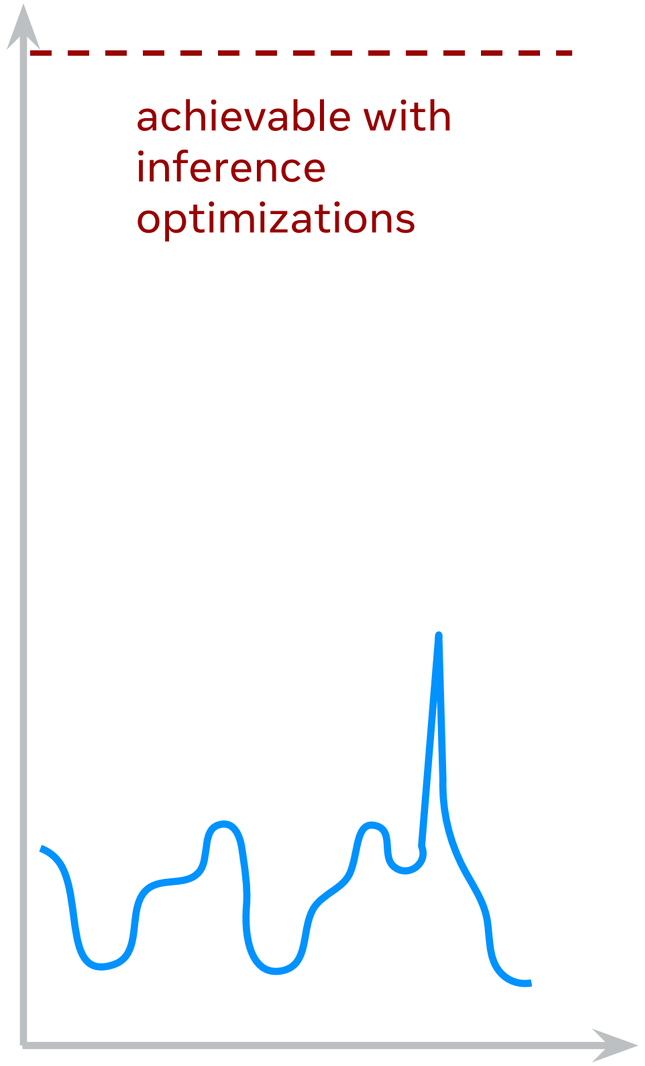
Solution	Latency	Quality	Complexity	Price
1	⭕	✅	⭕	$$
2	✅	⭕	⭕	$
3	✅	✅	✅	$$$



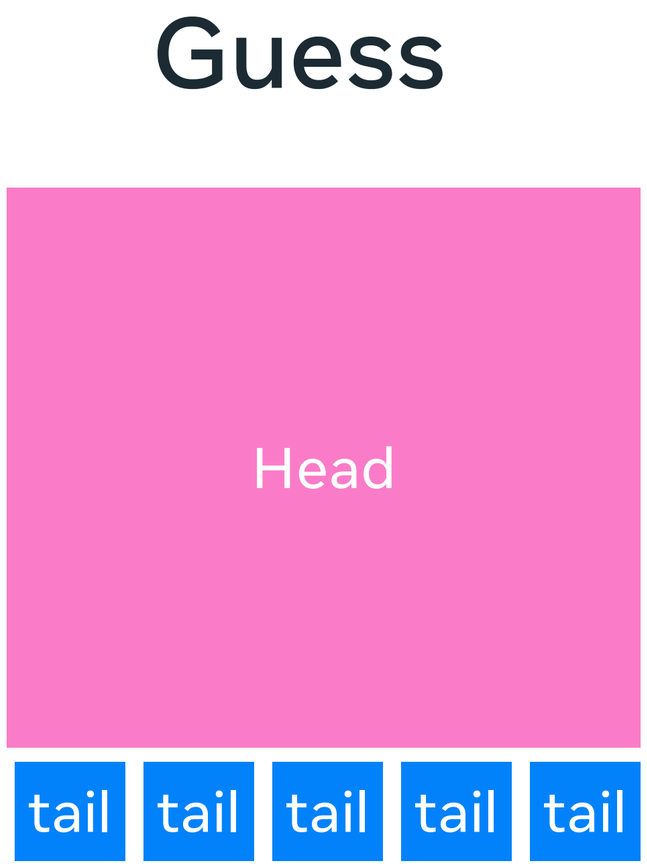
YES,

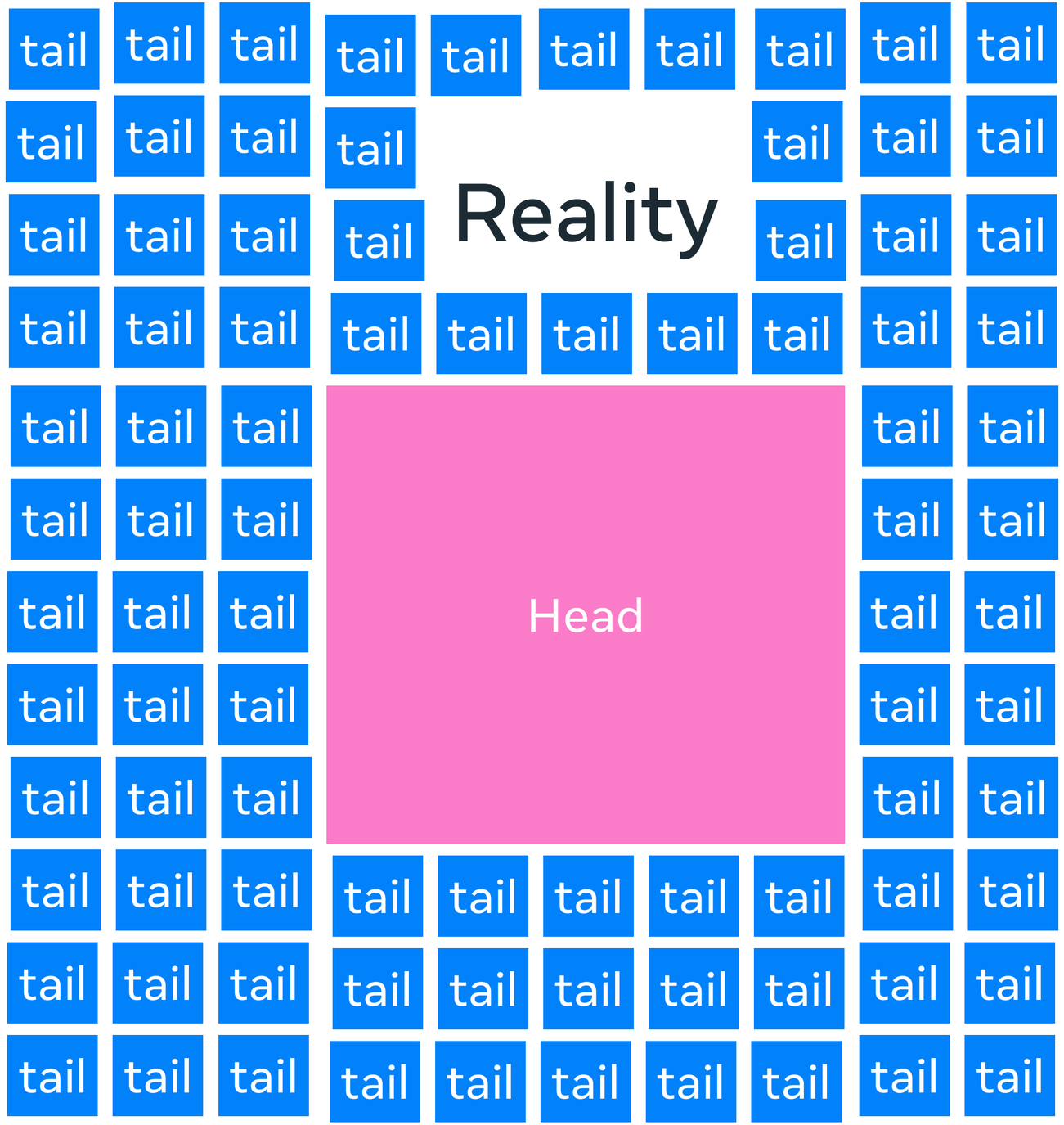

How to use these free capacity?
🤔

BUT,

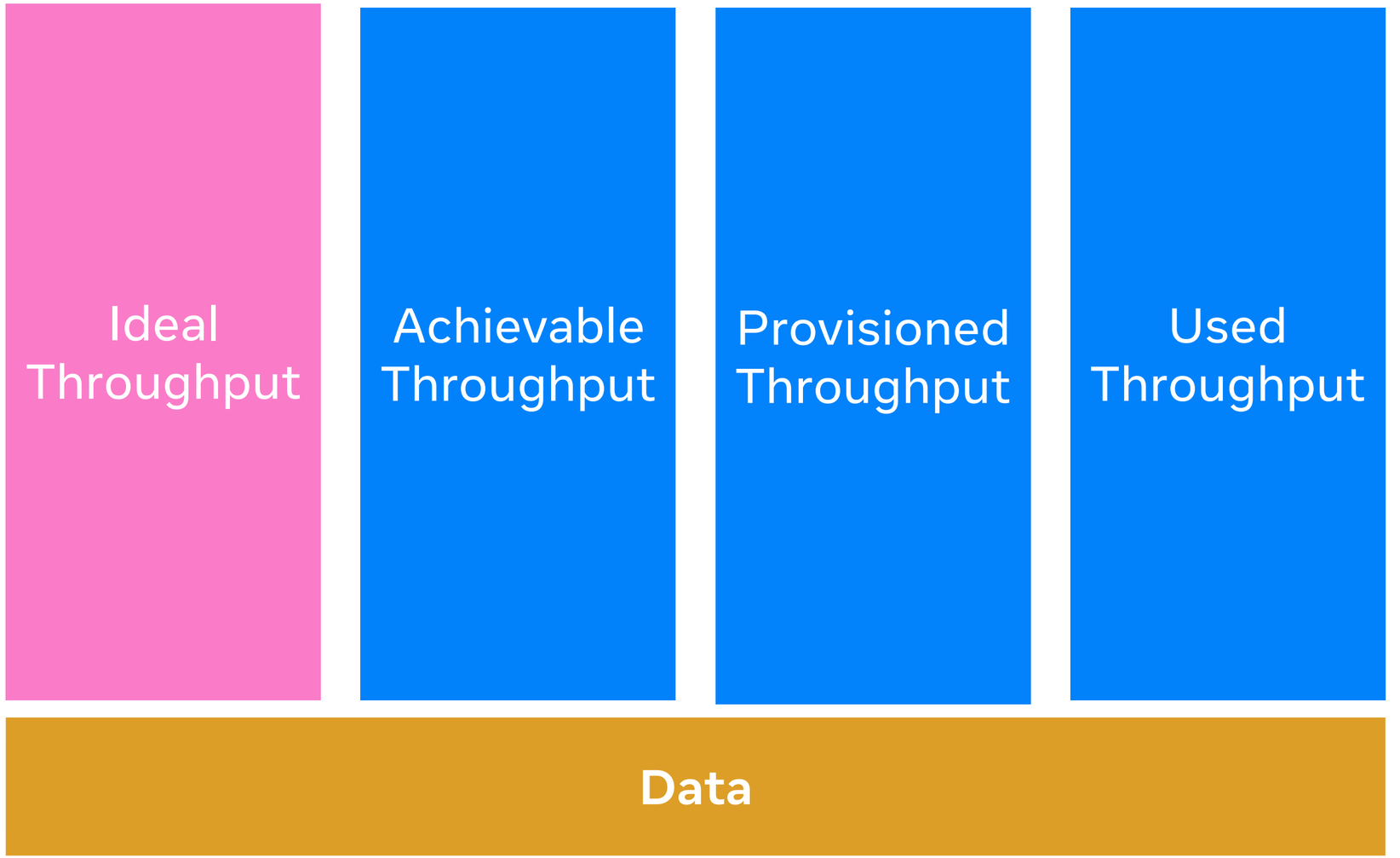

achievable with inference optimizations

STEP 2: Inference Optimization At Scale (Where to Focus)

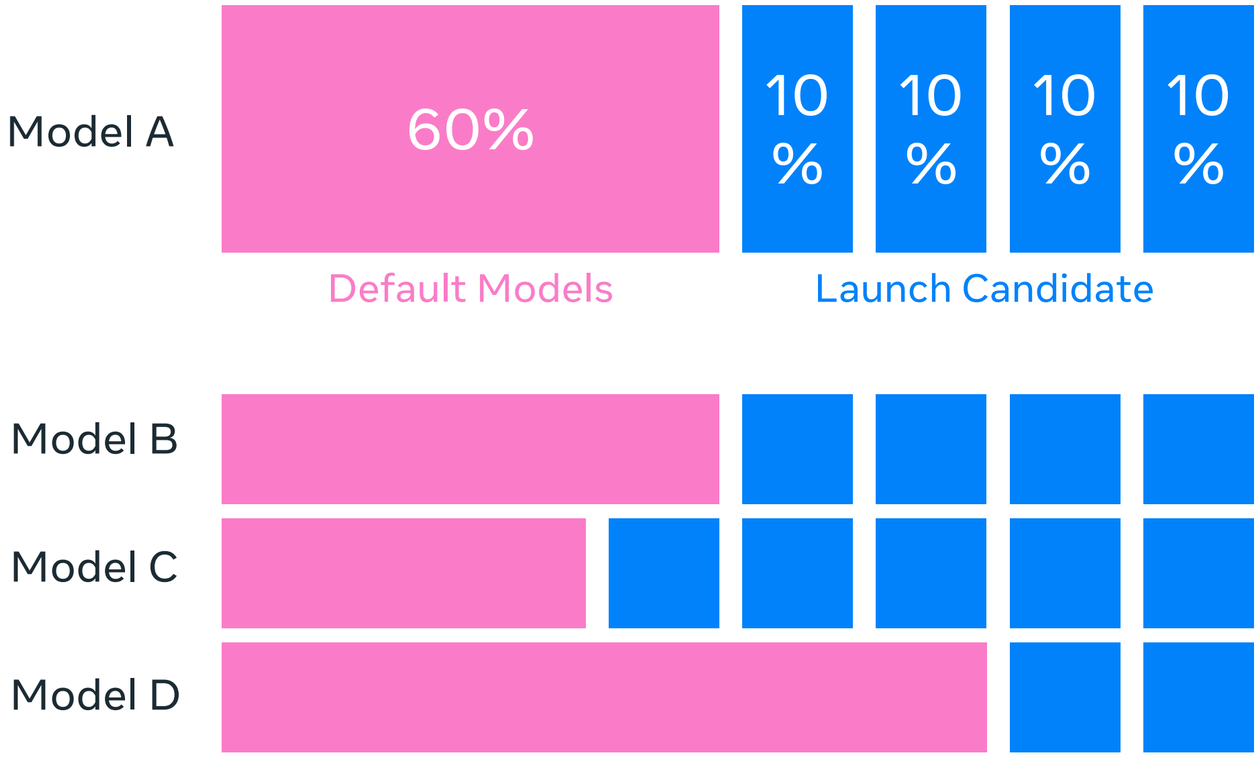


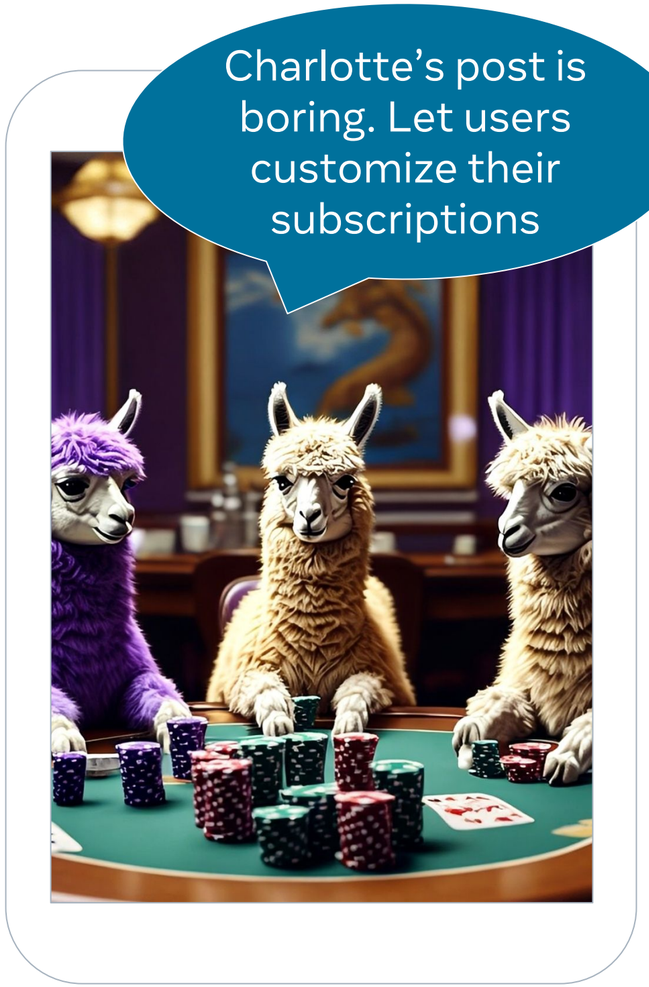


STEP 2: Inference Optimization At Scale (Data Driven Cost Reduction)



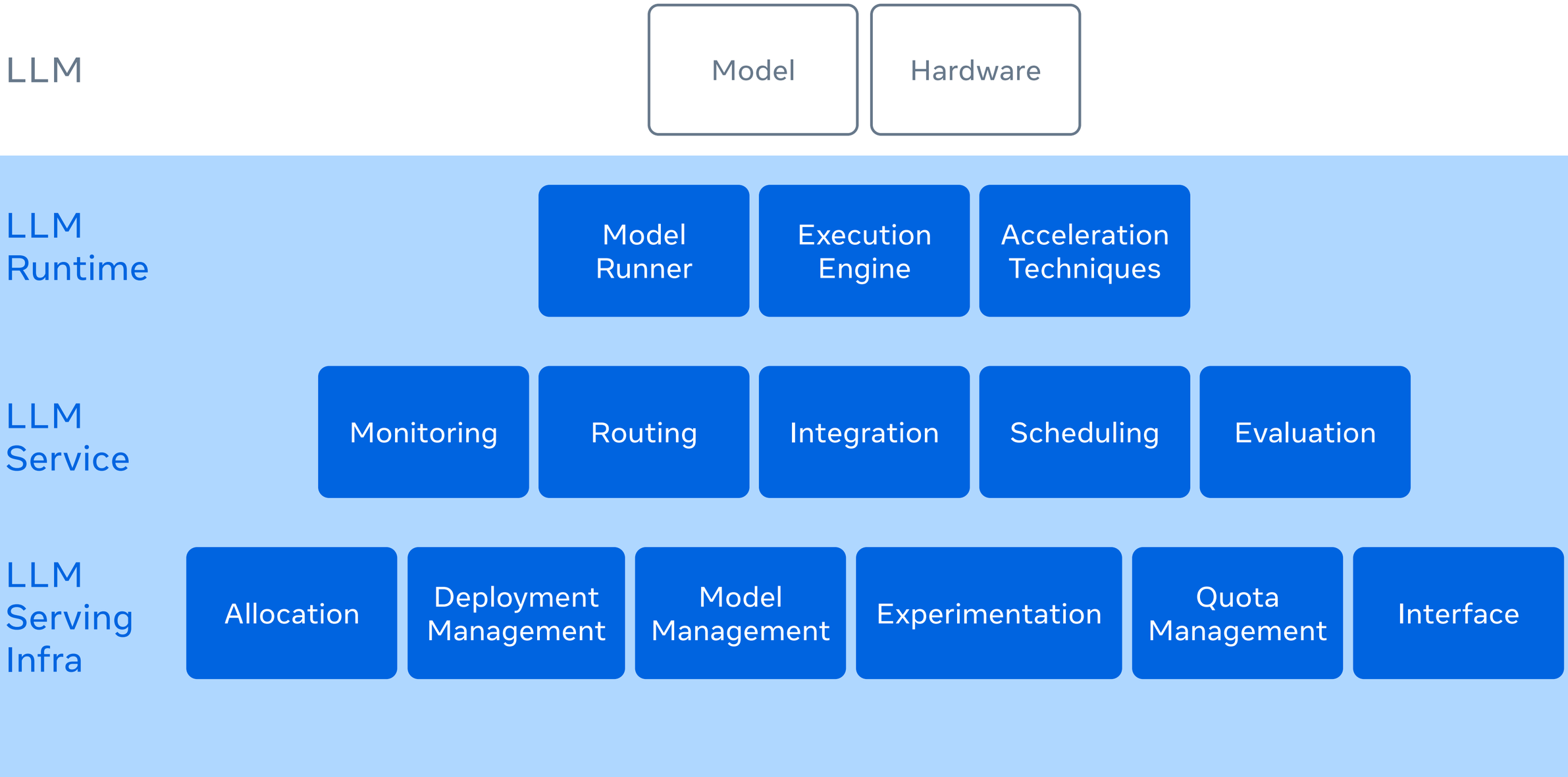
Traffic Behind the Scene





Putting together everything we talked so far … You’ll get a scalable LLM serving infrastructure!

THE LLM SERVING ICEBERG




Meta