Utilization is the key to efficiency

Aleksei “Alex” Semiglazov
Senior Systems Engineer

About me

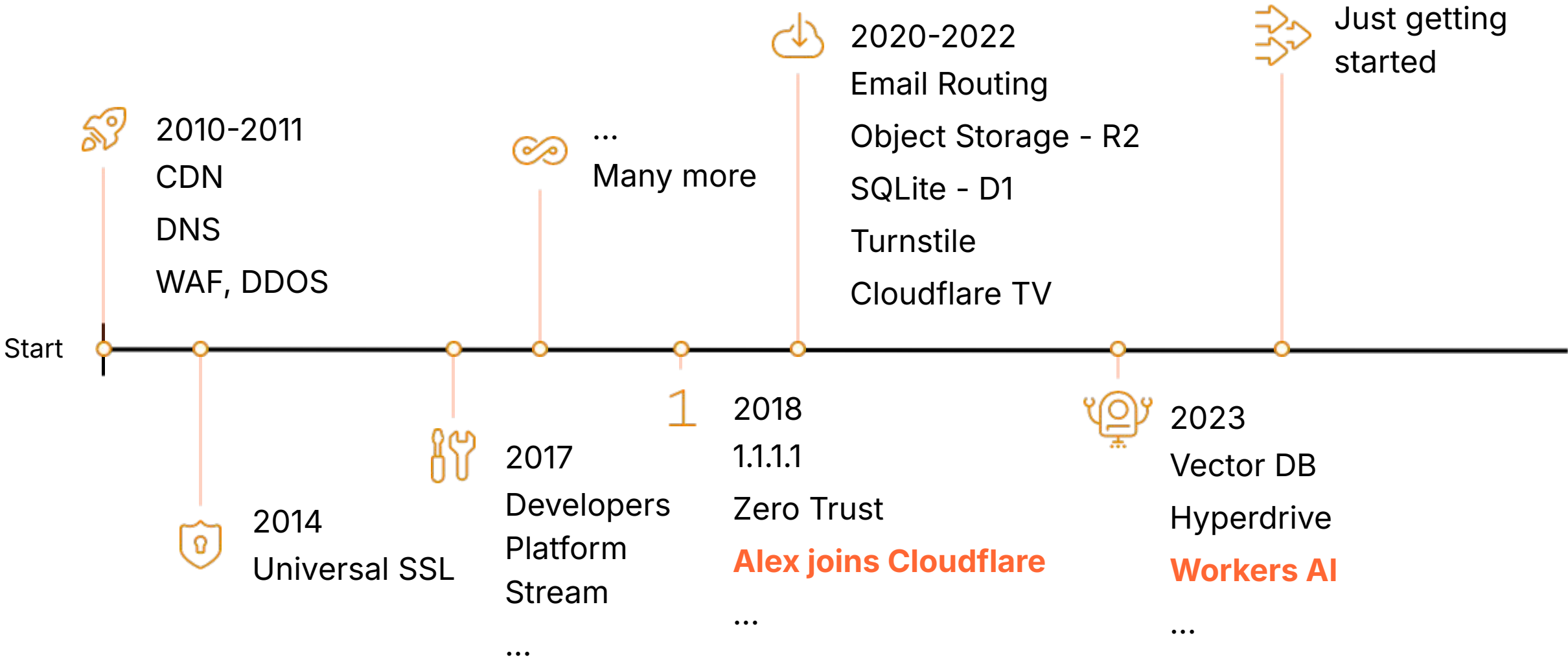
- Systems engineering
  - SRE and platform teams
    - observability
    - orchestration
    - analytics platform
- Product
  - Images
  - Workers AI
- Infrastructure focused
  - System design
  - Rust

Agenda

- ML evolution: our path to inference platform
- Container challenges: impact and optimizations
- Oversubscribing GPUs: ways to pack tighter
- Load balancing: permit management and load shedding
- Models scheduling, autoscaling and routing

Cloudflare evolution

https://developers.cloudflare.com/directory/



What is "The Edge"

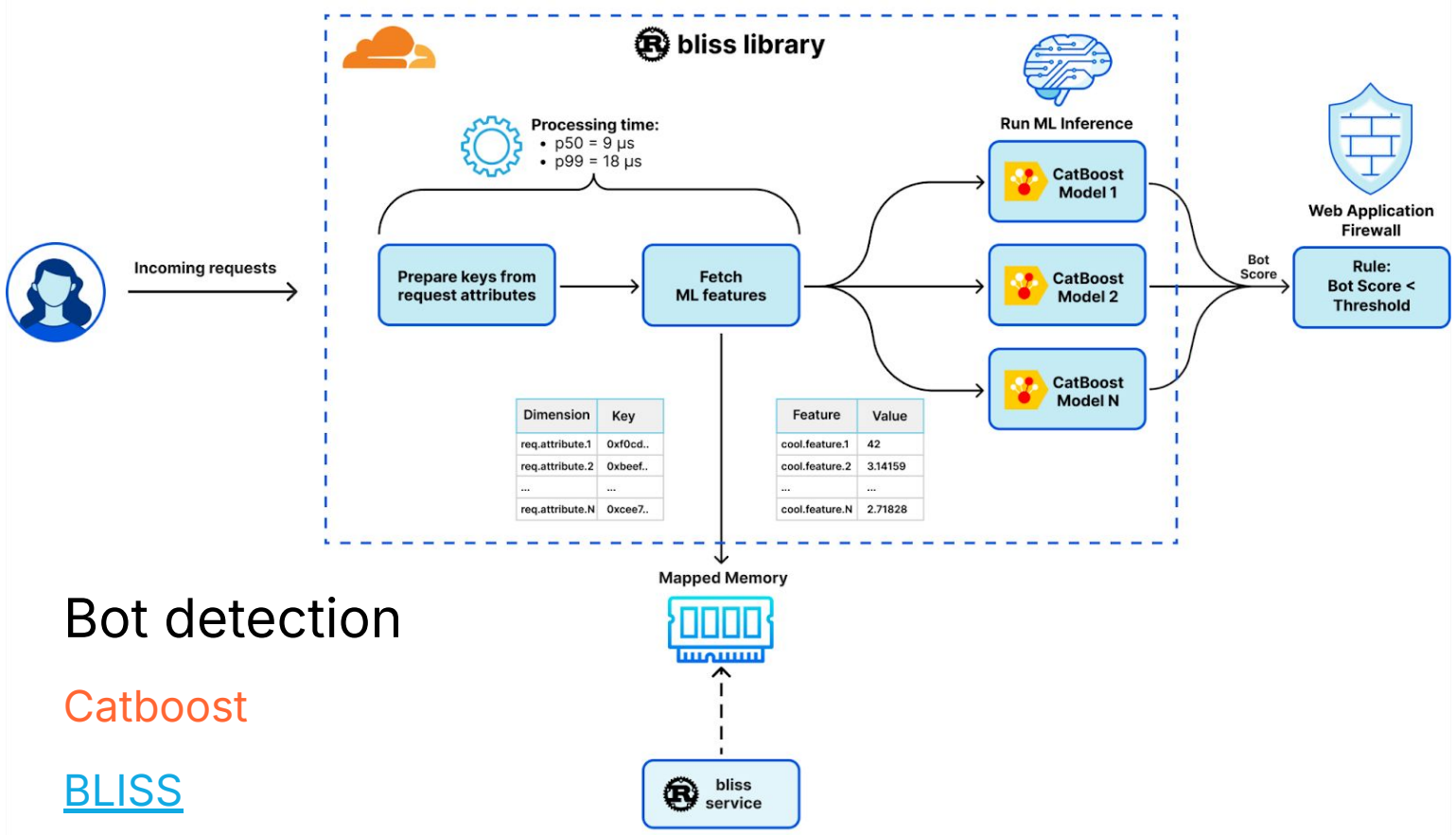
~~Not user devices~~

Edge Network:

- 330+ cities worldwide
- 200+ with GPUs
- 13000 peering networks
- 405 Tbps capacity
- ~50ms away from 95% population[1]

ML needs

CPU only, small models



Bot detection

Catboost

BLISS

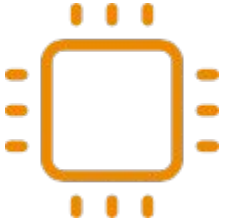
Deployed everywhere

Image: face cropping

Tensorflow library

Deployed everywhere

The Evolution Path



Pre-GPU era

CPU only

Small size models

Can run on every machine

Workers AI Beta

Small fraction of L4 GPUs

A few OSS models

Today

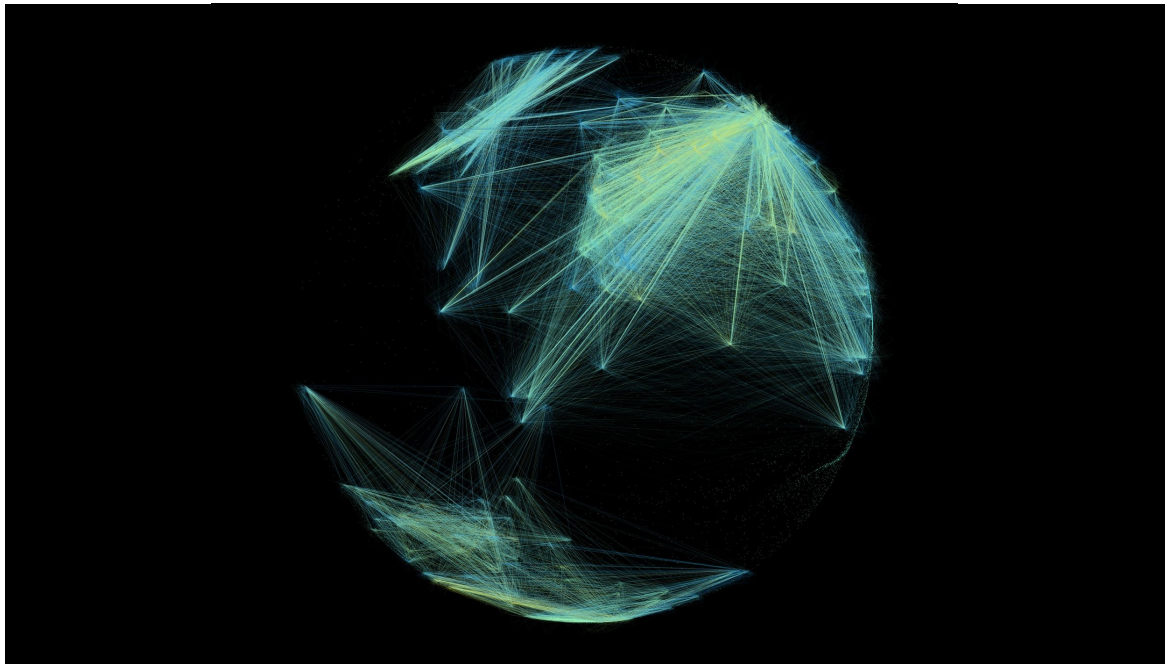
Large models

Heterogenous placements

Focus on GPU efficiency

Centralised vs Edge

- Resource pooling
- Easy Autoscaling
- Ultrafast network



- Laws of physics
- Comprehensive routing
- Different hardware profiles
- Shared resources

The Cloudflare paradox

Our Network is power and a challenge

Power of The Network

- Proximity to user
  - Can server users traffic from any site
- Anycast addresses
  - Can serve any* requests from any locations
- 13000 peering networks
  - Interconnects with IPX, cloud providers, private networks

Challenges with Inference

- Scarce placements
  - Low GPU density
- Shared resources and virtualization
  - Add overhead
- Distance
  - Model placements can very in distance from each other
  - Latency

Framing The Problem

Placement

- Security and isolation
  - Number one priority running 3rd party code
- Hardware profiles
  - Heterogeneous node profiles
  - Only subset of all nodes have GPUs installed
- Collocation
  - Resource constraints
  - Shared environment
- Routing

Models catalog

- Big and growing in size
  - Multiple models a month
- Variety of dependencies
  - OSS models come with plethora of different dependencies
- Different throughput and latency requirements
  - All models are different
- Varied demand
  - Traffic patterns vary per model/per day

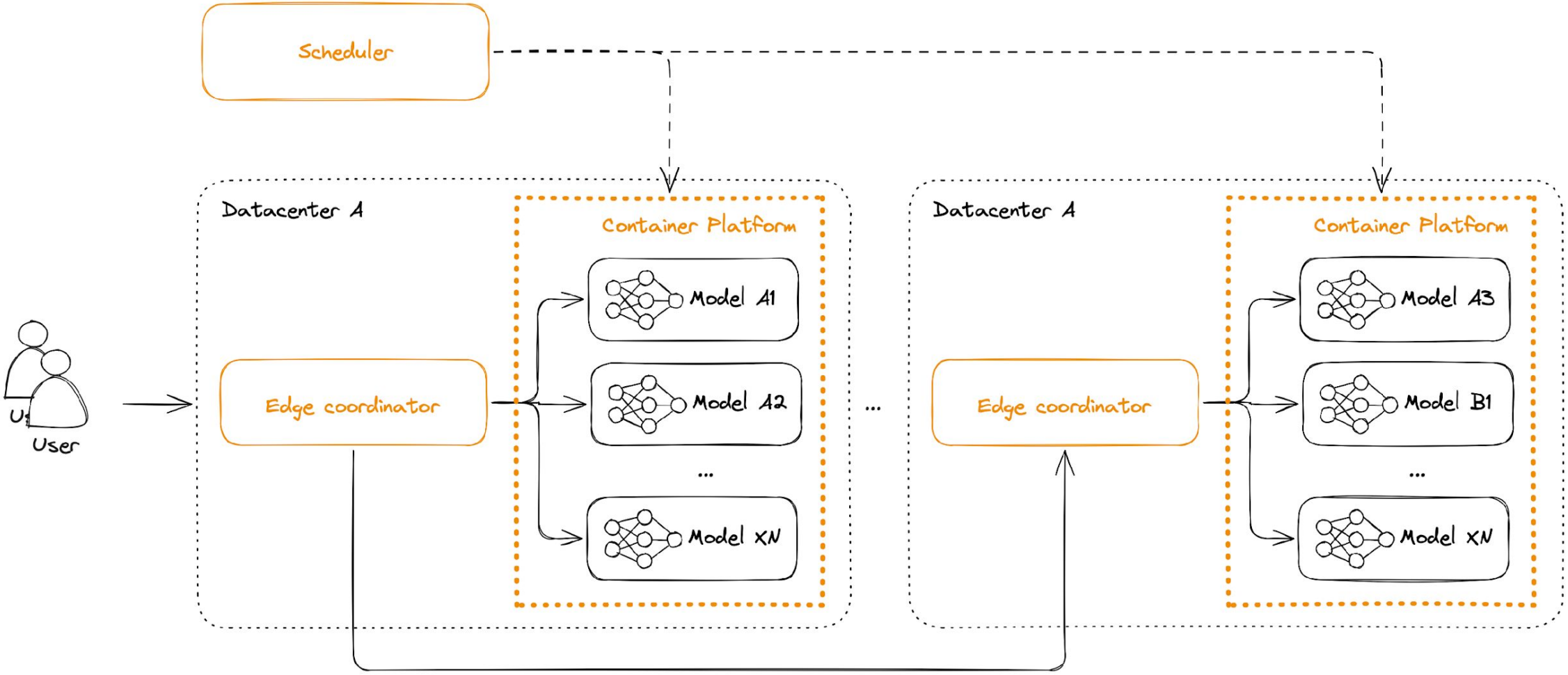
Scaling up

Objectives

- High utilisation
- No performance loss
- Low cost management

How to achieve high utilization without sacrificing user experience or reliability?

System Architecture



Model Catalog

- Text Generation
- Text Embeddings
- Text Classification
- Text Summarization
- Translation
- Text-to-Image
- Image-to-Text
- Speech Recognition
- Object Detection
- Image Classification

Container Challenges

Security requirements

- Untrusted code isolation
- Runtime agnostic support

Resource Constraints:

- Different GPU memory needs
- Driver compatibility
- Large Image Sizes (15GB+)
- Model weights can be huge

Model placement strategy

Container platform responsibilities

- GPU availability
  - GPU profiles,
    available resources
- Local scheduling delegation
  - Launches inference engine on a node
- Bin packing
  - Large models go first
- Affinity and location constraints
  - Instances clustering
  - Location prioritization and restrictions

Sandboxed execution

Multi-Runtime Support

- gVisor
  - Strong isolation
  - GPU pass-through
  - Noticeable performance overhead
- Firecracker
  - Doesn't support GPU passthrough
  - https://github.com/firecracker-microvm/firecracker/discussions/4845
- cloud-hypervisor
  - Evaluation phase

Resource constraints and limitations

- Cold start are long
  - Docker images are big
  - Filesystem is slow
- No nvidia capabilities support
  - No CPU sharing technics (MIG)

vLLM limitations with our setup

- No multi-model support
- Python overhead
- Sandbox performance penalty
- Long startup times

Infire: custom inference Engine

- JIT model-specific compilation
- cuBLASlt matrix operations
- Dedicated CUDA graphs
- Lower kernel overhead

up 50% improvement in throughput

https://blog.cloudflare.com/cloudflares-most-efficient-ai-inference-engine/

How we built the most efficient inference engine for Cloudflare's network

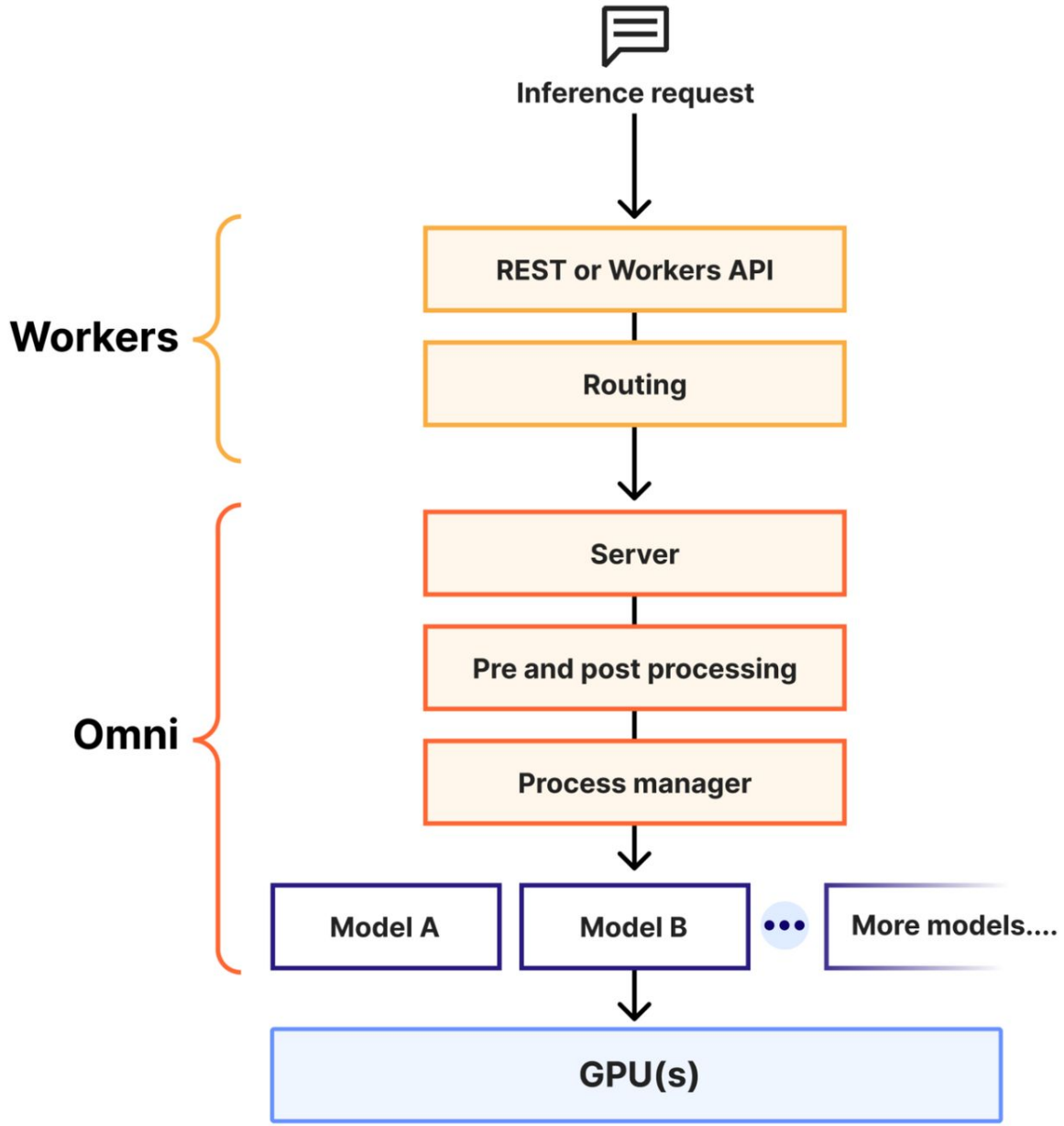
2025-08-27

AI Week | LLM | Workers AI

Infire is an LLM inference engine that employs a range of techniques to maximize resource utilization, allowing us to serve AI models more efficiently with better performance for Cloudflare workloads....

OMNI: multi-model platform

- Lightweight process isolation
  - Uses cgroups
  - Better for scaling/downscaling
- Filesystem isolation
  - Streamline model dependencies
- Single control plane
  - Spawn model instances by itself
- CPU memory management
  - Python ignores groups and uses plutil
- GPU overcommitment



OMNI: GPUs overcommitment

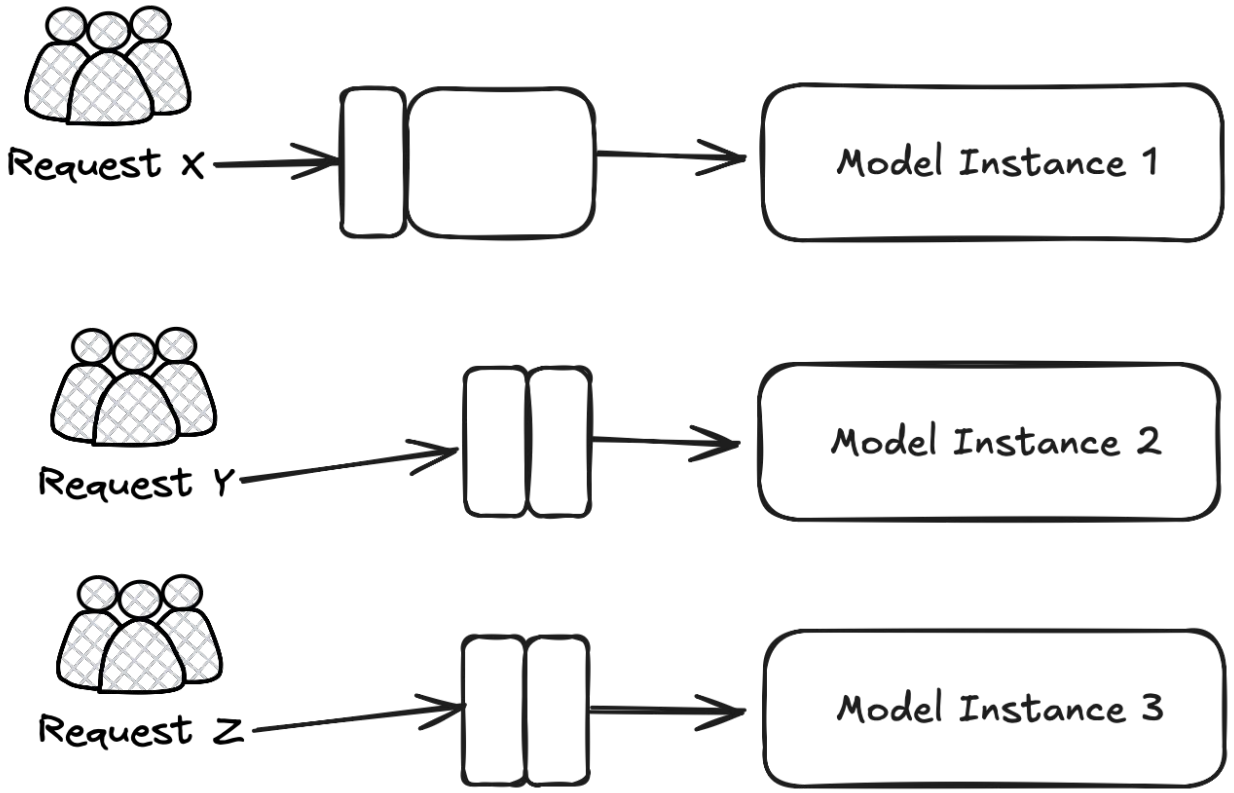
- Intercept CUDA Allocations
  - Multiple models, single GPU
- Force unified memory mode
  - Shares same memory address space with both GPU and CPU
- GPU memory management
  - Only fraction of memory is available to model
- Automatic swapping
  - Swap idle models to CPU memory
  - Works great for small models
- Saving up to 4 GPUs
  - 13 small models on 1 GPU, overcommitting 400% memory

Edge Coordinator

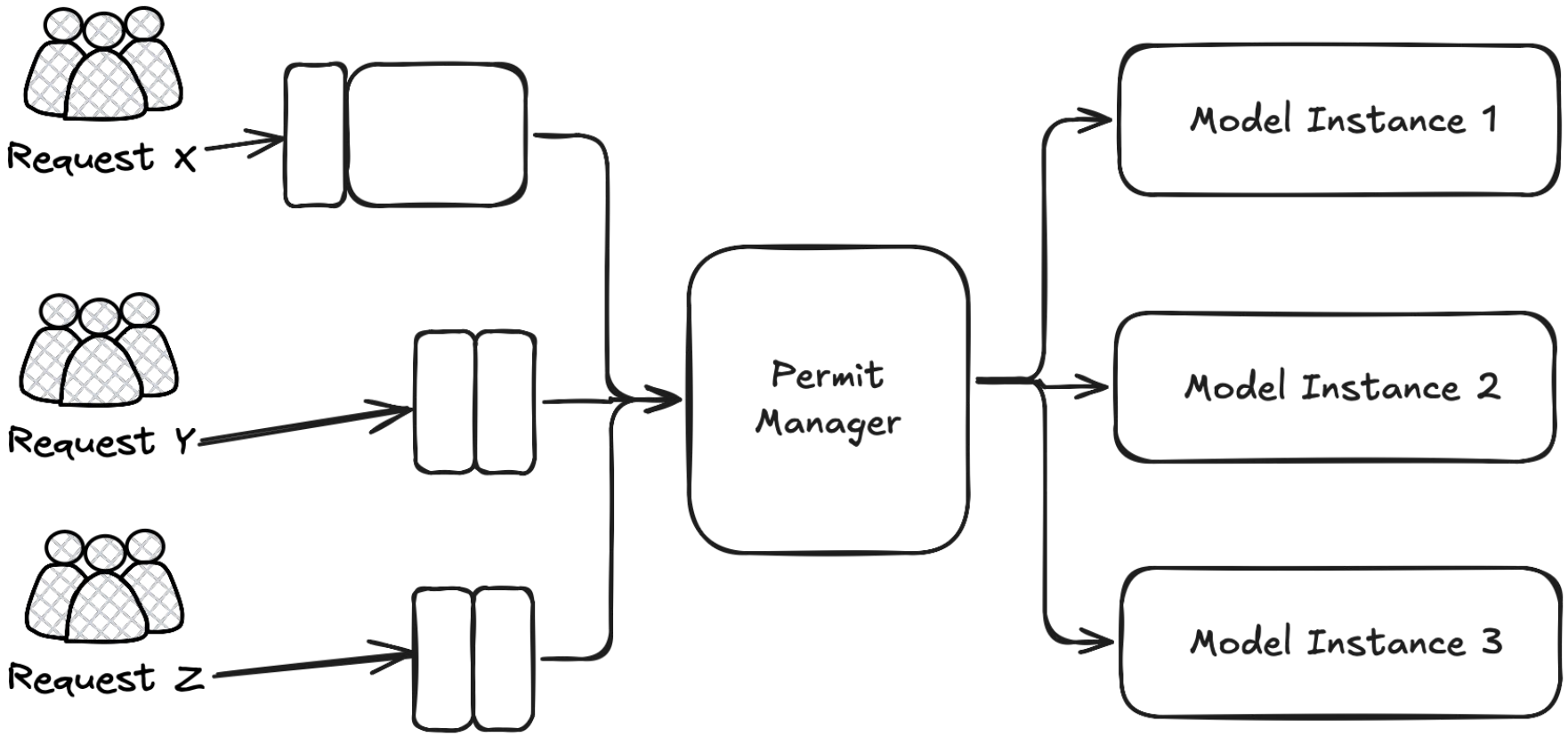
- routing
  - send requests through optimal paths
  - error discovery and backoff mechanics
- requests handling
  - sync/asynchronous workloads
  - request preparation
  - protocol negotiation: http, grpc
- permit management
  - in-memory state
  - local queues
- observability
  - tracing, metrics and logs collection

Stateless request distribution

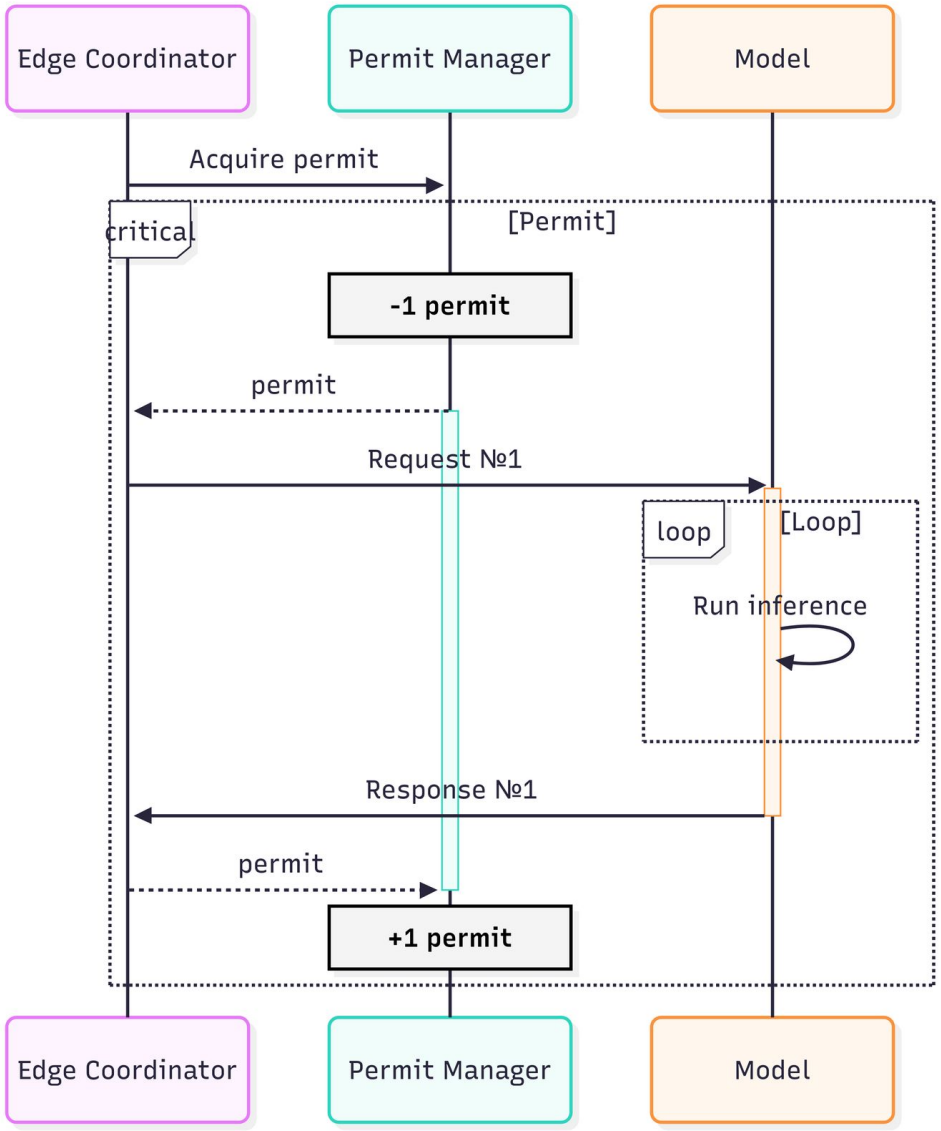
- Tail latencies
- Underutilization



Permit Manager



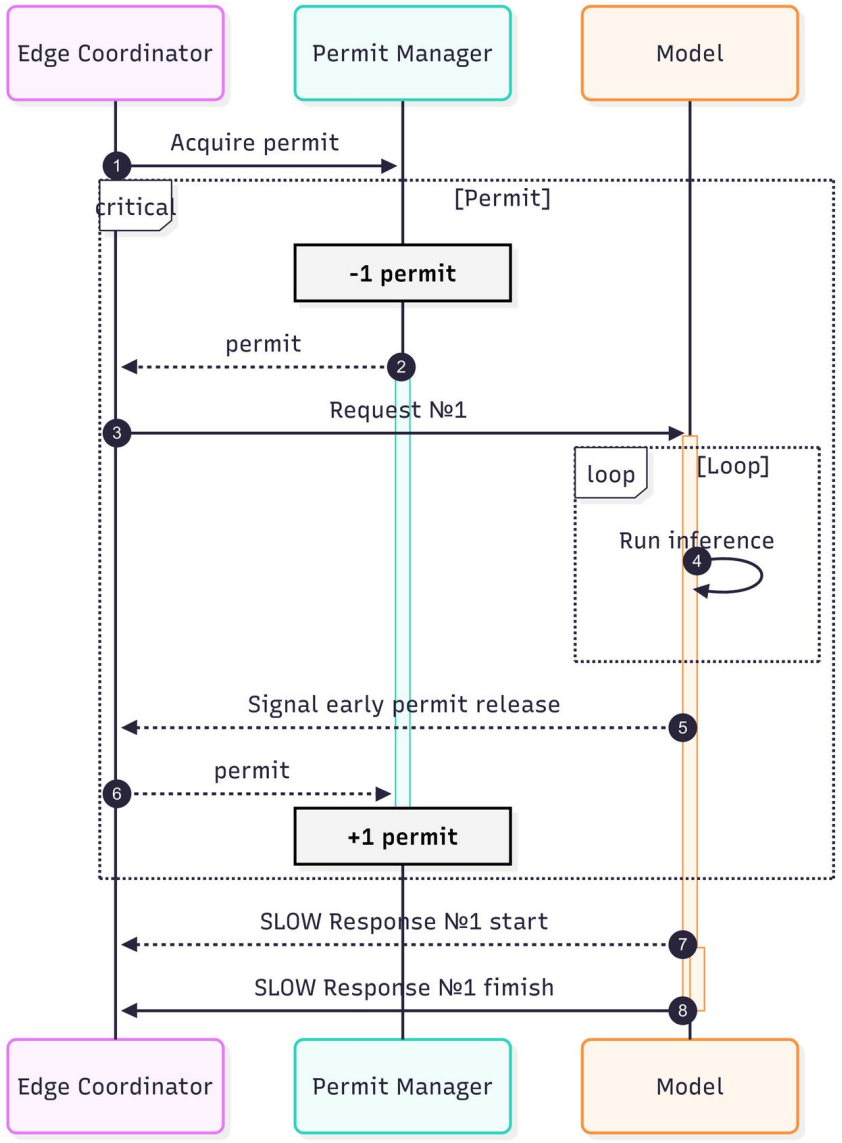
Inference permit



- permit acquisition

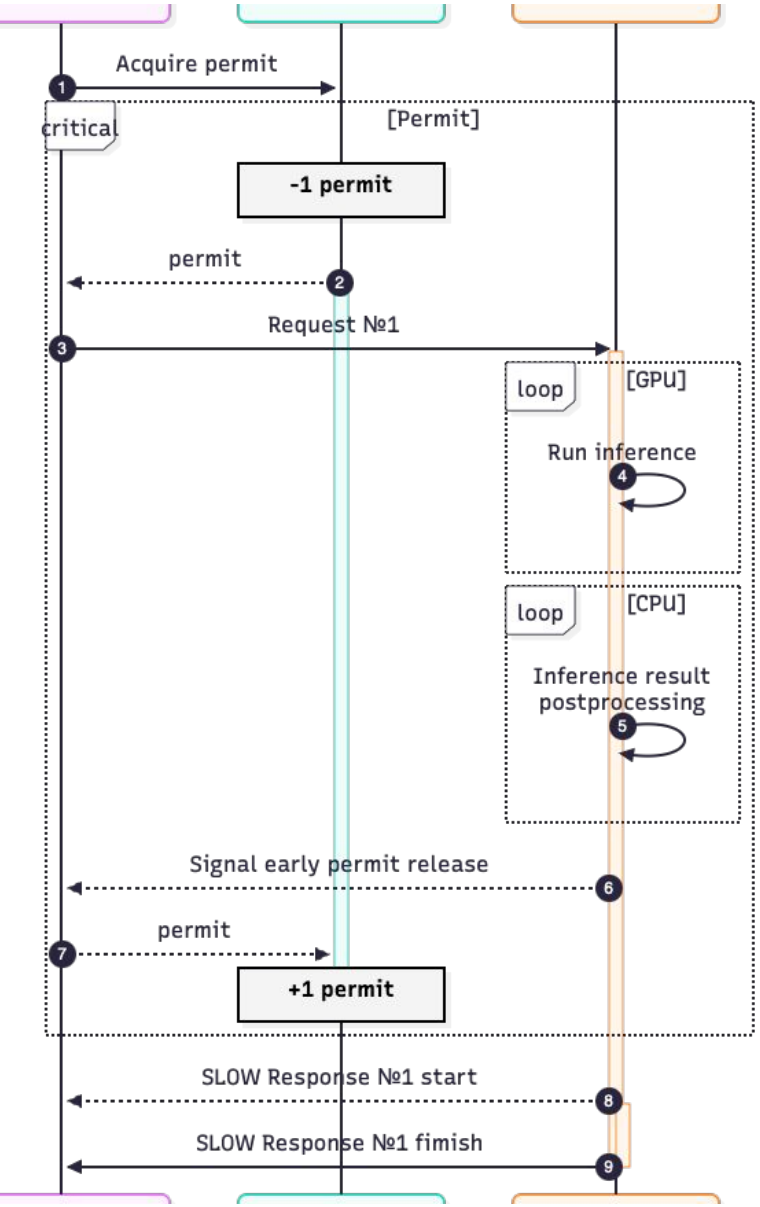
  acquires permit for request
- capacity management
- fast

  no network calls, in-memory state



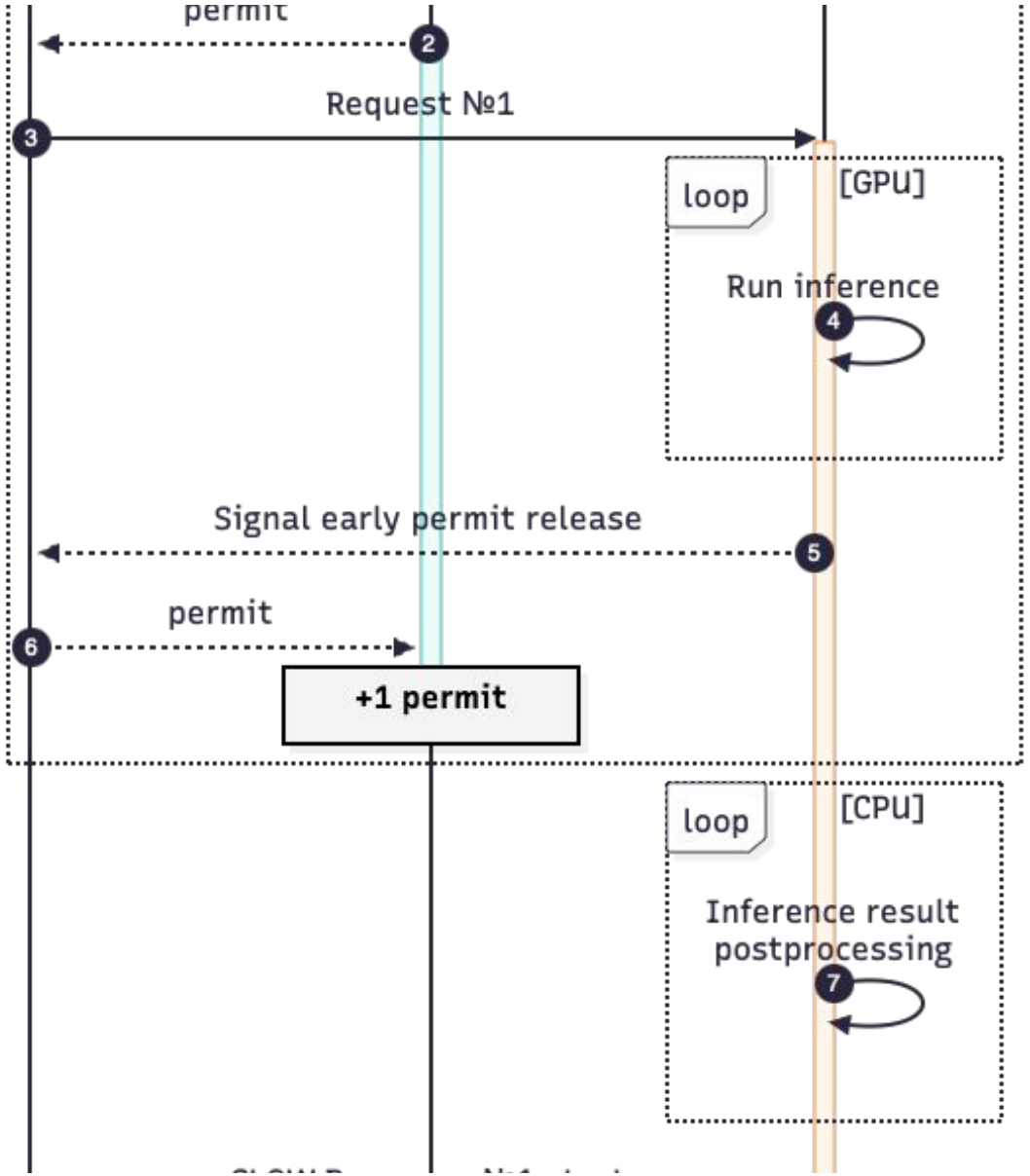
Early Release

- response phase doesn't affect
  slow readers, buffering
- early release signal (5th point)
  permit is held for inference
- permit is available for next request
  increase throughput



CPU-GPU split

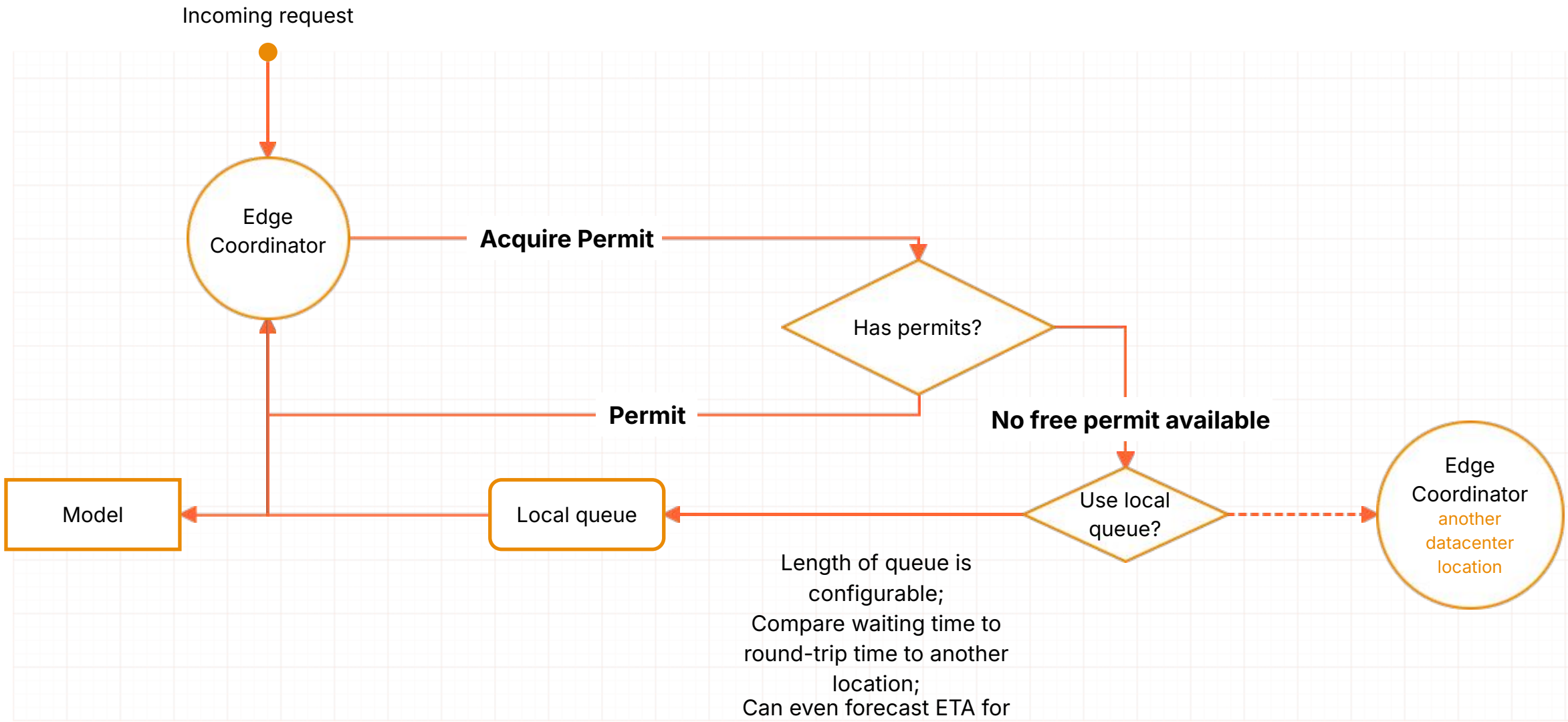
- No hardware encoder support
  - H100 doesn't have hardware encoder
- Diffusion models
  - CPU intensive for image/video encoding



CPU-GPU split

- diffusion models
  - CPU intensive for image/video encoding
- release signal when GPU is idle
  - can save 200/300 ms
- predictive ETA
  - increase throughput

Predictive Queuing



Prefix caching trade-offs

session level KV-Cache reuse

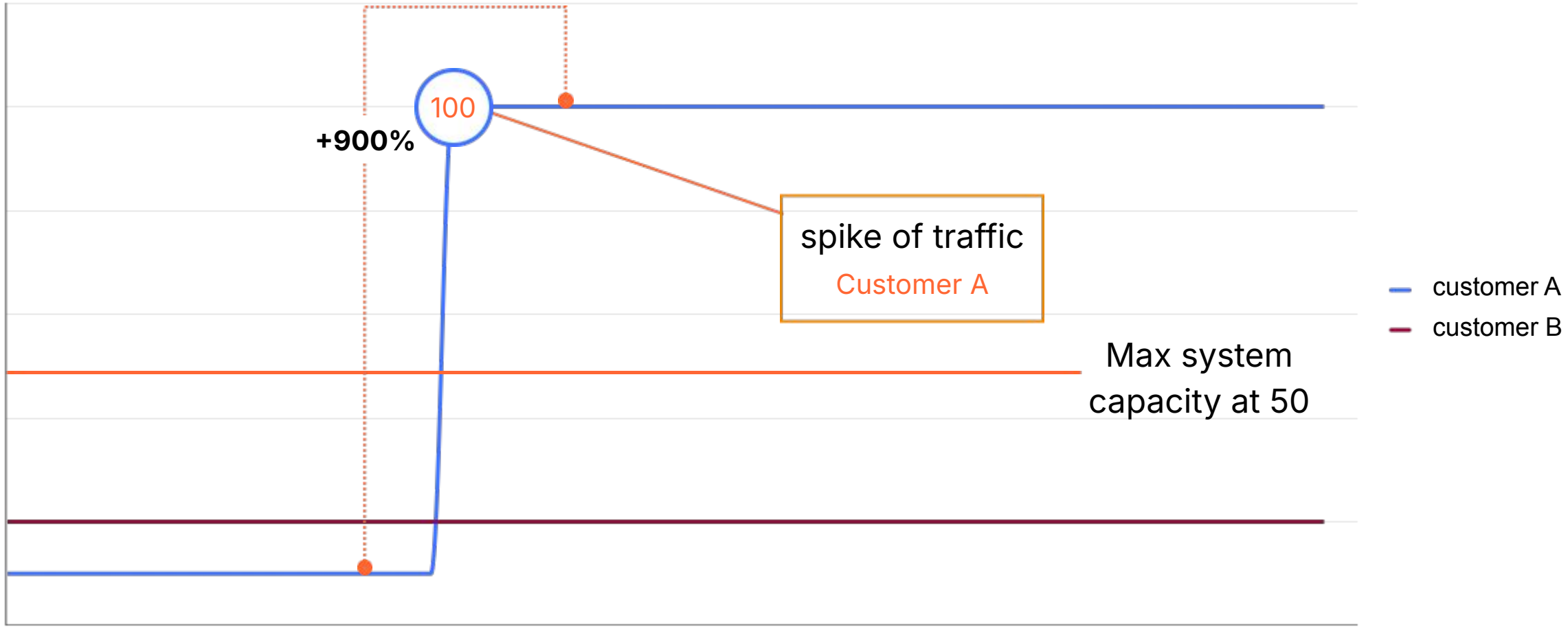
Pros

- Reuse calculations
  - chat sessions are good candidate
- Improves utilization
  - faster calculations, less time spent in GPU
- Faster TTFT

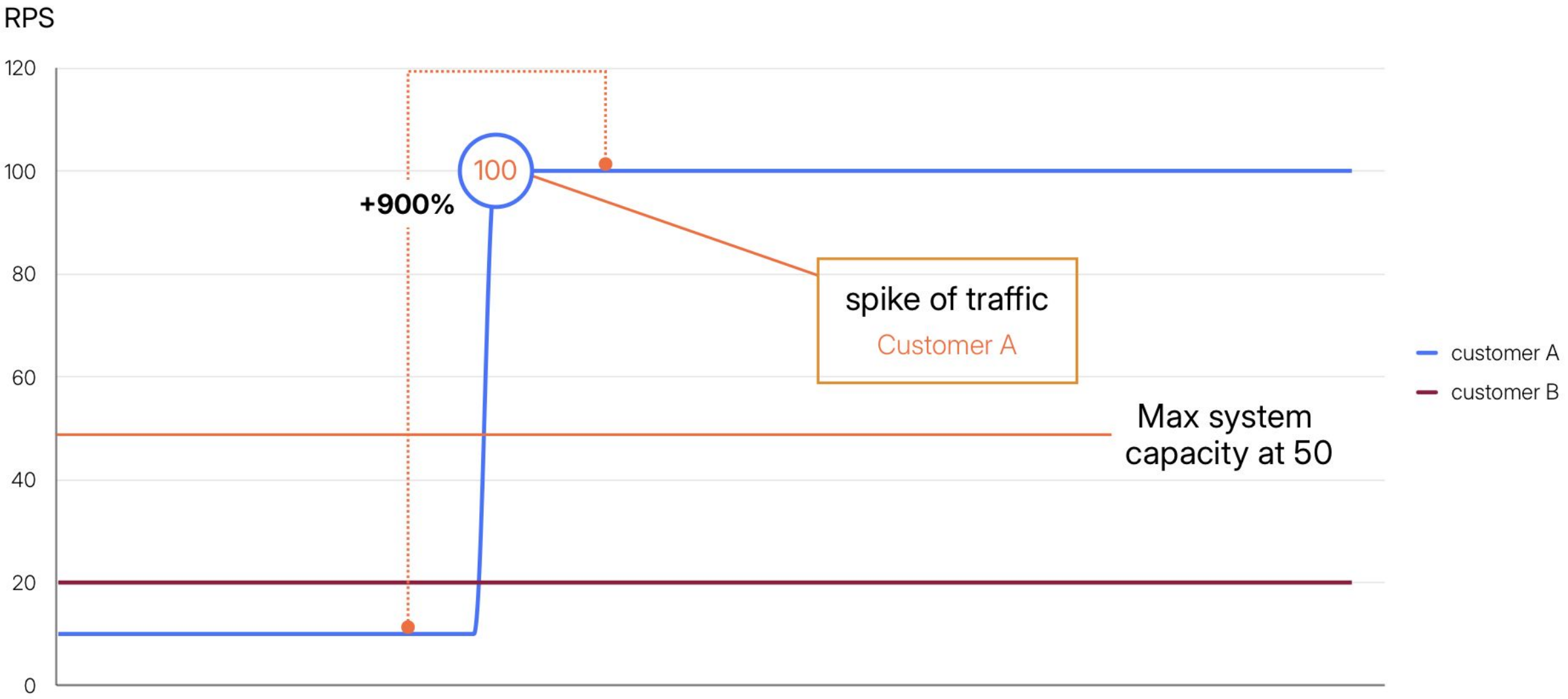
Cons

- Sticky routing
  - can involve extra time due to queuing
- Session leakage
  - don't share cache across sessions: no cross-principal reuse
- Memory demand
  - KV-cache needs extra GPU memory
  - paged cache management
  - large contexts can be an issue
  - session isolation multiplies usage

Traffic spikes

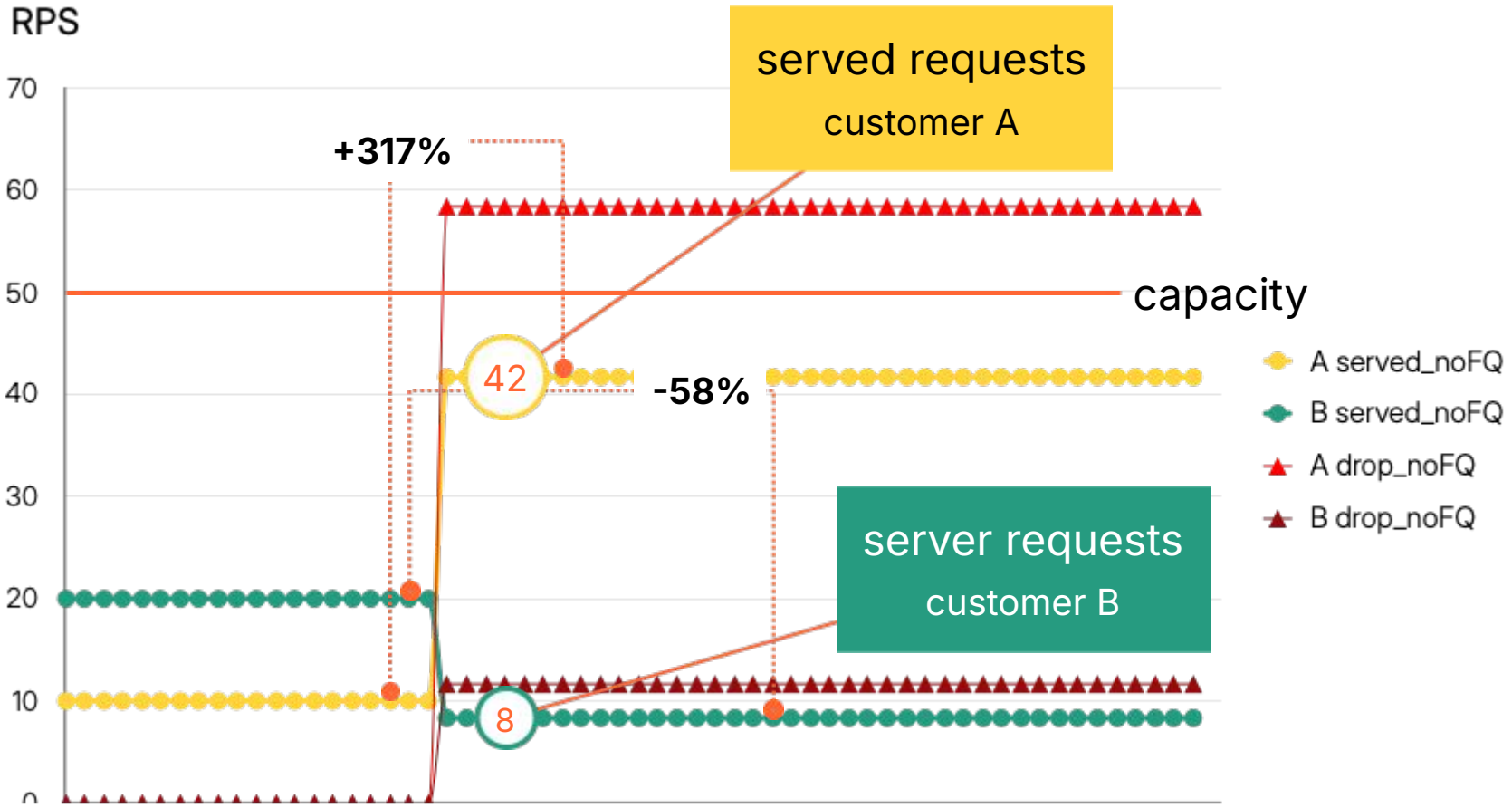


Traffic spikes



Proportional load distribution

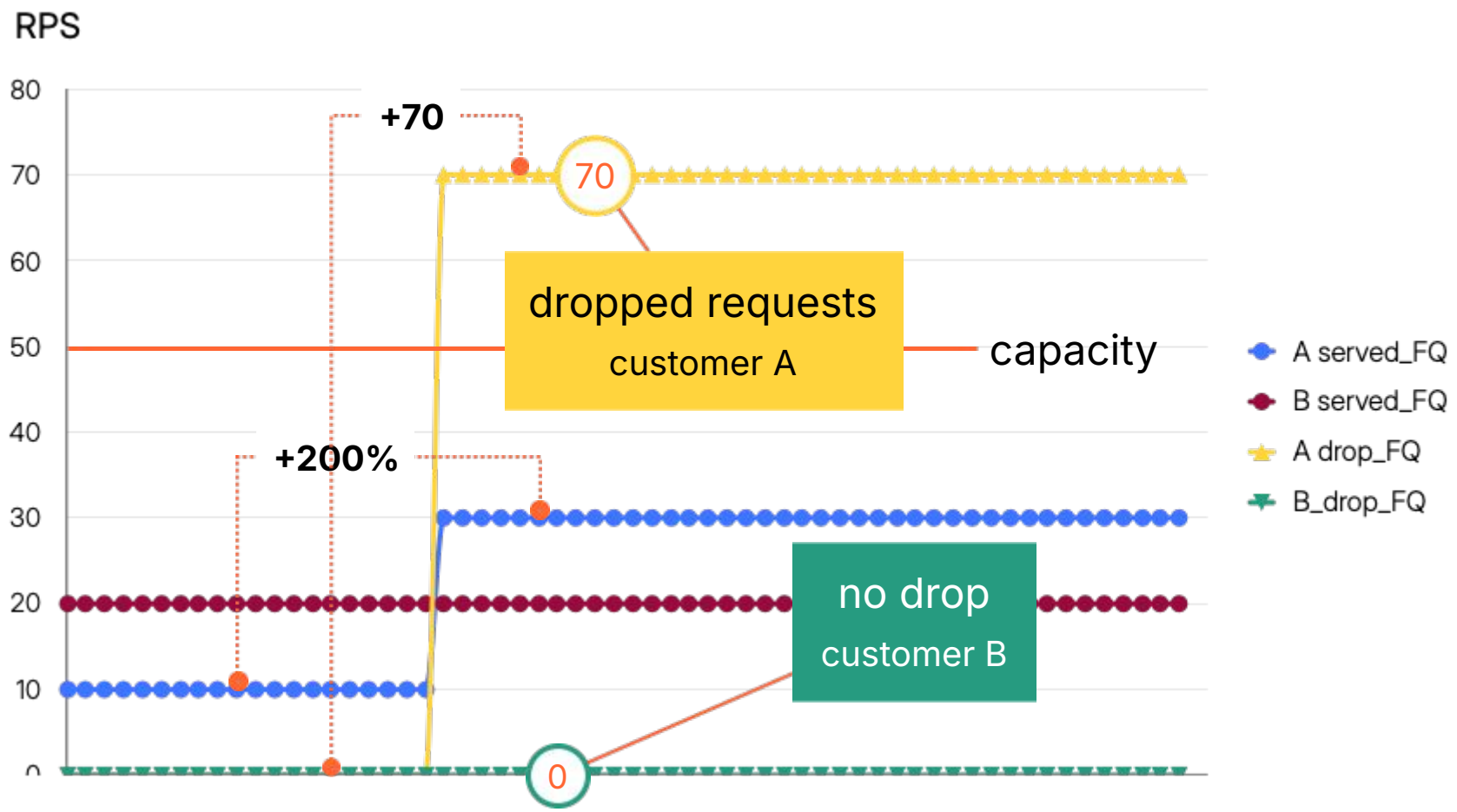
Saturation: requests rejected proportionally



- System is at capacity
  - all 50 permits are in use
- Unfair load shedding
  - both customers see the drop

Load shedding with Fair Queues

Saturation: request rejection - FAIR QUEUE enabled



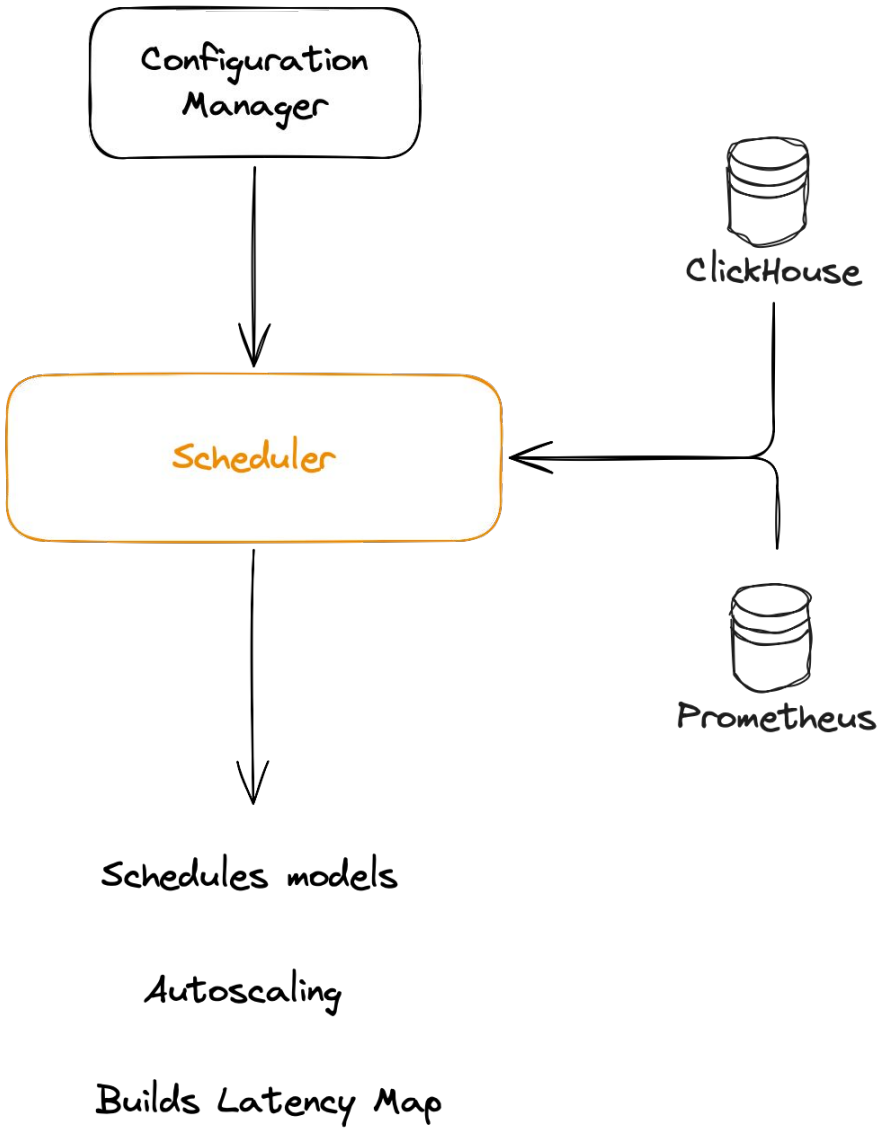
- Fair load shedding
  - capacity distributed equally
- No artificial limits
  - rate-limit for abuse
- weighted fair-queues
  - adjust resource fractions per customers tiers

Scheduler Overview

Automatic NOC

Key responsibilities

- Resource allocation across the globe
  - Optimal model allocation across location
- Traffic routing optimizations
  - Constructing a DC-to-DC map with RTT time
- Utilization prediction
  - Calculation and forecast footprint
- Configuration and release management
  - Gradual rollout
  - Configuration changes



Autoscaling

Metrics to target utilization

- 20 minutes window
  - ClickHouse and Prometheus
- Inference wall time
  - Works well for models with concurrency 1
  - Better for e2e than DCGM metrics
- Free permits
  - Concurrency > 1
  - Gauge is less precise, works ok summing over time
- Queuing time
  - Time requests spend in queues
- Out-of-Capacity errors

Issues

- Reactive and inert
  - Looking back, not now
- Slow scale up
  - Cold start takes time
- Spiky traffic
  - Out-of-capacity errors
- Oscillation
  - Scaling up and down

Forecast-Based solution

- Similar data points
  - ClickHouse
  - Prometheus
- Predict an hour ahead
  - Multiple intervals to account for seasonality
- Pattern recognition
  - Amortize for temporary traffic spikes

Global Routing challenge

- Can’t place models everywhere
  - Not feasible to have every copy in every location
  - Demand is different across locations
- Can’t serve from one location
  - Not enough capacity in one datacenter
  - Long RTT creates dissatisfaction

Latency map

- Calculate latency probes
  - Many-to-many probes
- Distribute map
- Intersect with model placement
  - Edge coordinator reads the map and calculates
- Route to optimal location

Routing algorithms

Different goals - different algorithms

- Latency
- Throughput

Tested approaches

- Sorted by distance
- Traveling Salesman
- Secretary Problem
- Random choice
  - no thundering herd
  - no cascading overload
  - no predictable hotspots

Lesson 1: Observability

- Request flow bottlenecks
- Cross-datacenter latency
- Model loading issues
- Resource contention

Lesson 2: Performance vs Virtualization

Performance requirements drove custom solutions

- gVisor adds significant overhead
- Python doesn’t serve well us in production
- Custom solutions - worth investment
- Oversubscription - reduces CapEx

Lesson 3: Prediction is a challenge

- Error margins are high
- User diversity helps
  - More customers = better predictions
  - Internal products are more predictable
- Spiky traffic remains hard
  - Async requests for smoothing
  - Cold start for large models is long

Lesson 4: Smart geography

Being closer to our client is in our DNA

- Regional vs global models
  - not all models need at every part of the world
  - Smart placement > global coverage
- Endpoint clustering
  - Model allocation in large groups
  - Trade-off availability vs utilization
- Adaptive routing
  - Adjust strategy with traffic shifts
- Deadline-aware queues
  - Choose waiting over roundtrip when it's faster

History lesson

Post-WWII: abundance

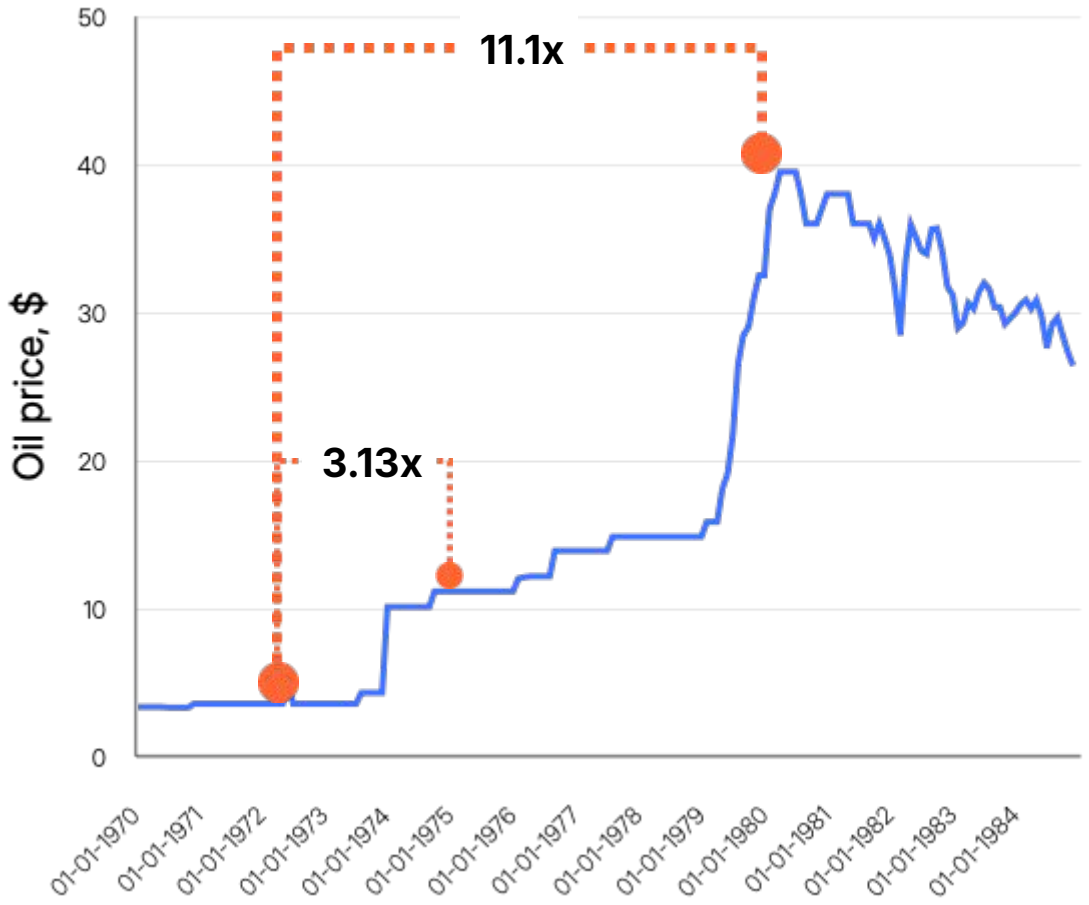
- Cheap oil
- Little renewable incentives
- Inefficient engines

  A 1966 Pontiac GTO with a 389" - 9 mpg (26L/100km)

  Beautiful cars, but very inefficient

Constraint

- Oil price 10x in a decade
- Overhaul in many industries
  - Power efficient engines
  - Preservable construction materials
  - Renewables and nuclear energy



Today's reality

- Bigger models
  - Increased footprint
- Growing catalog
  - Longer tail
- Deep reasoning
  - Compute demanding
  - Chain of thought increases context
- Hardware is expensive
  - Backlog of orders is shrinking

The Innovation Constraint

“Where nothing resists us, nothing sharpens us. Without hunger, there is no invention; without limits, no discovery. Abundance lulls us into complacency, dissolves discipline. Constraints are the forge where greatness is made. Remove the forge, and all that remains is decay masked as comfort.”

Thank you