MArk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving

Chengliang Zhang, Minchen Yu, Wei Wang — HKUST
Feng Yan — University of Nevada, Reno

Background and Motivation

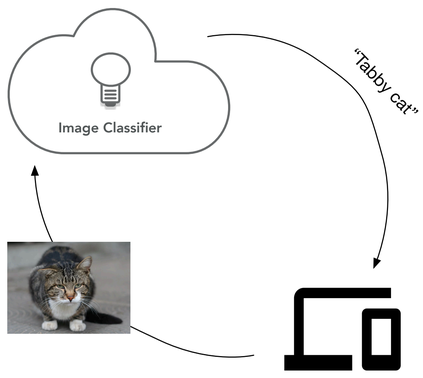
1. Why a dedicated system for ML Inference?

ML serving has many distinct properties:

- *compute intensive*;
- *deterministic* processing time;
- can benefit from hardware *accelerators*, e.g, GPUs, TPUs;
- *stateless*.

2. Design objectives

- serve ML inference on public cloud;
- scale quickly to dynamic queries;
- cost-effective;
- SLO-aware.



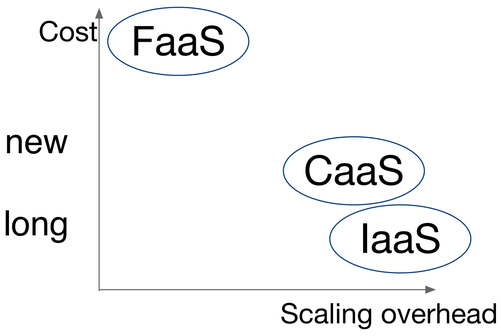
Challenges:

- Which cloud services to use, Infrastructure as a Service (IaaS), Container as a Service (CaaS), or Function as a Service (FaaS)?
- How to navigate through the large configuration space?
- How to leverage cost-performance tradeoffs such as preemptable instances and burstable instances?

Characterization Highlights

4. IaaS vs CaaS vs FaaS

- IaaS has the lowest cost and latency, but new instances takes minutes to launch.
- FaaS scales well, but it is expensive with long latency.
- CaaS offers the middle ground.



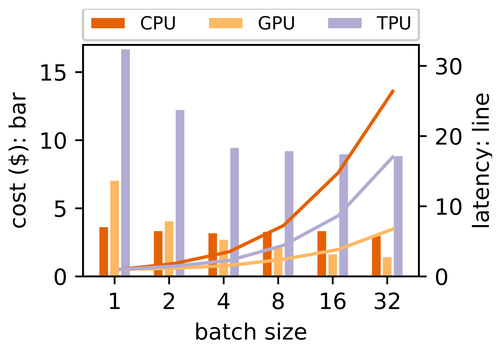
★ *Combine the cost-effective IaaS and the scalable FaaS.*

5. CPU vs GPU vs TPU

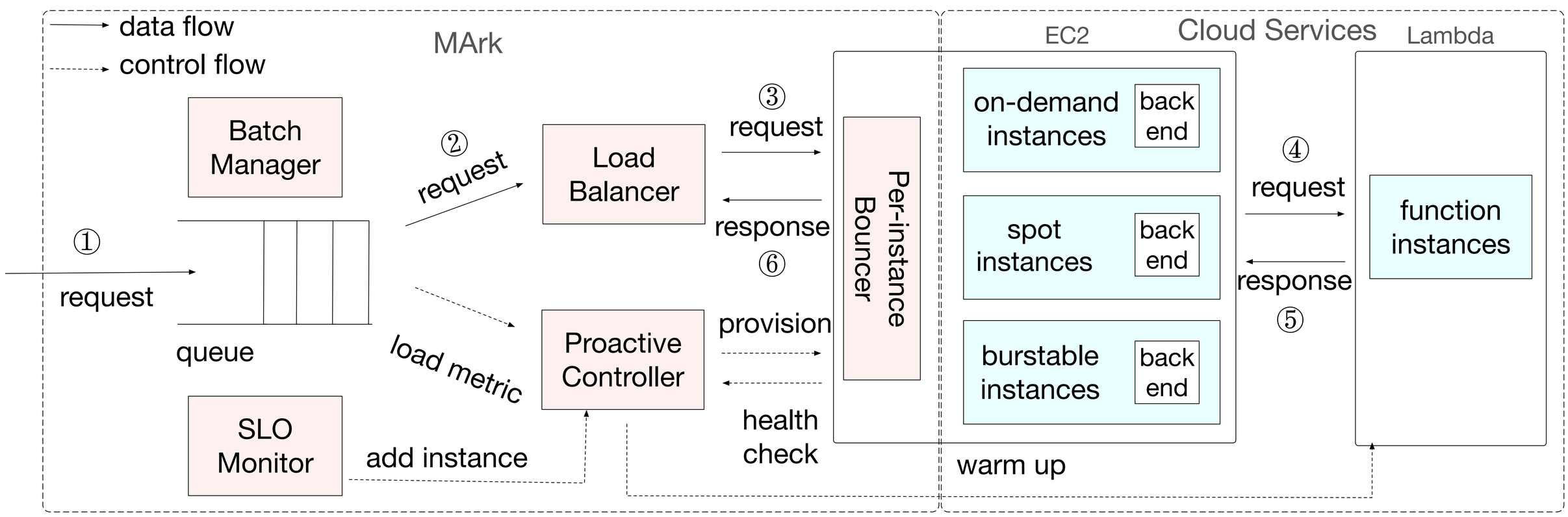
- High utilization is required to justify the high cost of GPUs and TPUs
- Judicious batching is needed
- GPUs can be cheaper than CPUs
- TPUs are not suitable for inference

★ *Predict workload to maintain high utilization*

★ *Judicious batching to navigate the tradeoff between cost and latency*



The cost and batch latency of serving 1 million inception-v3 inference requests with various batch sizes.



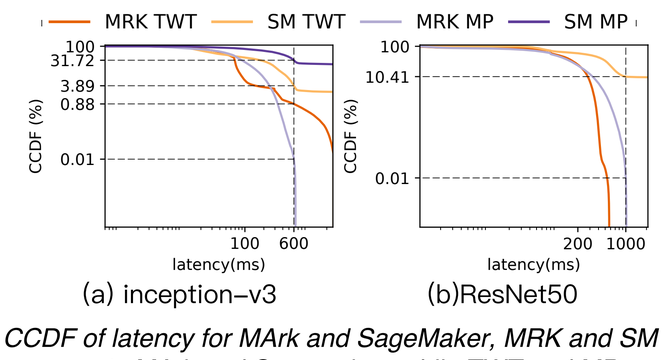
System Design

6. MArk (Model Ark) system design

- uses IaaS (EC2) as the primary means of service;
- employs FaaS (Lambda) to cover load spikes;
- uses proactive provisioning, planing instances based on prediction;
- dynamically batches requests according to instance types and request arrivals;
- tracks SLO compliance, and launches burstable instances when needed.

SLO compliance:

As shown, MArk retained SLO compliance, with both real life and synthetic workloads.



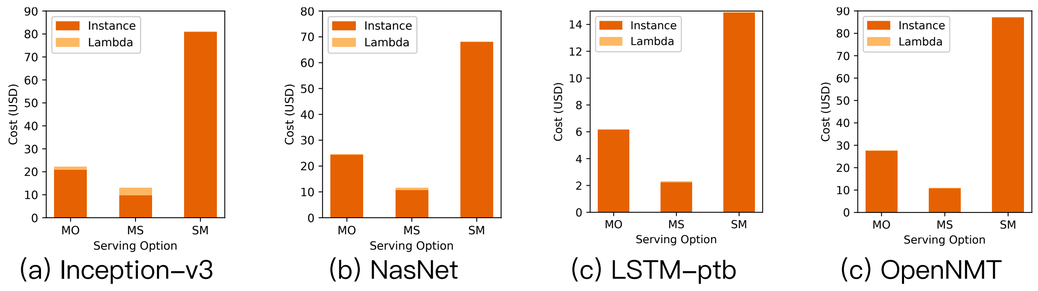
(a) inception–v3 (b)ResNet50

CCDF of latency for MArk and SageMaker, MRK and SM represents MArk and Sagemeker, while TWT and MP represents Twitter and MMPP workload respectively

Evaluation Highlights

7. What about performance?

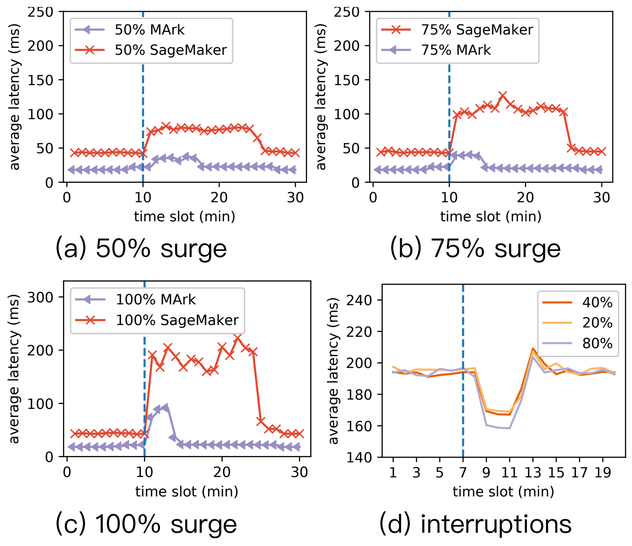
Cost savings:



(a) Inception–v3 (b) NasNet (c) LSTM–ptb (c) OpenNMT

Cost ($) comparison of MArk-ondemand (MO), MArk-spot (MS), and SageMaker (SM)

- Up to 3.6x cost reduction with only on-demand instances.
- Up to 7.8x cost reduction if spot instances are considered.



(a) 50% surge (b) 75% surge (c) 100% surge (d) interruptions

Microbenchmarks:

We applied sudden demand surges on MArk and depicted the results in (a) - (c).

We interrupted up to 80% of the instances in a 20 machine cluster, and depicted the results in (d).

MArk is open sourced at: https://github.com/marcoszh/MArk-Project