BATCH: Machine Learning Inference Serving on Serverless Platforms with Adaptive Batching

Samuel Adams
Richard Brun
David Melanson
Outline

- Batching workloads in Machine Learning (ML)
- Bursty workloads in ML
- Utilization of Function as a Service (FaaS)
- Lightweight BATCH framework
- Results
- Criticisms and concerns
Understanding the problem

In Machine learning (ML) we train models to infer new instances of data.

It is ideal to classify data in batches, not one at a time.

Given the burstiness of data inference needs in real life applications, this can be difficult to achieve.

![Real-world traces from [29] and [30].](image)
Previous work

Sagemaker: An industry standard solution which uses an IaaS platform to infer data [1]

*Problem: Strategy does not lend itself to bursty ML workloads.*
Previous work

MArk: Creates AWS EC2 instance and creates serverless functions to deal with bursty workloads [2]

Method far outperforms SageMaker in latency, but can still struggle with bursty workloads
Use Functions as a Service (FaaS) to horizontally scale to workloads

Challenges

- Given the stateless nature of FaaS, batching is not supported
- Inference needs low latency
- Dynamic tuning of FaaS parameter

Authors introduce their framework, BATCH, to solve these issues
BATCH Framework

Arrival Process: Observes distribution of incoming jobs (1a)

Service Times: How long it takes to infer data (1b)
BATCH Framework

Profiler: Transforms incoming into a stochastic process (2a)

Uses regression analysis to capture the relationship between system configuration and request service times (2b)
BATCH Framework

Performance Optimizer: Takes multiple inputs and tries to determine optimal batch size/timeout

Budget: How much we are willing to spend

SLO: How much latency we are willing to endure
BATCH Framework

Performance Optimizer:
Recommends ideal batch size/timeout (4a)

Memory Size: storage for FaaS (4b)

Workload: Data to infer (5)

Batch Dispatching Buffer: array of data to infer (6)

Figure was inspired by Fig. 5 of the paper
The Profiler

Uses an analytical model to predict runtimes and cost for batches on different hardware

No machine learning involved, interpolates based on data for low computational cost and low latency

Runs intermittently, takes 10s in tests (on t2.nano)

Fig. 13: Request latency distribution with arrivals driven from real workload traces.
BATCH overhead

When compared to AWS SageMaker, cost is only half ($0.14/day)

All components of BATCH run on a single, low cost VM (in the authors’ test a t2.nano)

BATCH is itself bursty, lending to cost savings
Testing Distribution

Decently models bursty workloads!

Fig. 10: Intensity of arrival processes used to evaluate BATCH.
Results: Latency

BATCH generally keeps latency within acceptable bounds, but is significantly higher in general
Results: Cost

At the cost of latency, cost is dramatically reduced

Batch is vastly better than SageMaker in all workloads tested

Batch is even better than optimally configured lambda
CRITIQUE OF RESEARCH PAPER

- Title
- Keywords
- Section I: Introduction
- Section II: Motivation and Challenges
- Section III: BATCH Design
- Section IV: Problem Formulation and Solution
- Section V: Prototype Implementation
- Section VI: Results
- Section VII: Related Work
- Section VIII: Concluding Remarks
STRENGTHS

- Clear and concise
  - “The Performance Optimizer is the core component of BATCH”
  - Incorporating “Observation #” boxes within paper
- Numerous and relevant references
- Documentation
WEAKNESSES

- Figure 11
- Optimization
  - Is this the best we can do?
- Assumptions

Fig. 11: Coefficient of variation (CV) of service time for various memory/batch size configurations.
“To the best of our knowledge this is the first analytical model that can capture accurately the shape of the latency distribution in the presence of bursty arrivals and deterministic service times.”
Future working includes extending BATCH to support different service time distributions and adopting optimization algorithms that are faster than the exhaustive search used here to support co-optimization of latency and cost.
Citations

[1] “Amazon. Build, train, and deploy machine learning models at scale.”