Continuum

A Platform for Cost-Aware, Low-Latency Continual Learning

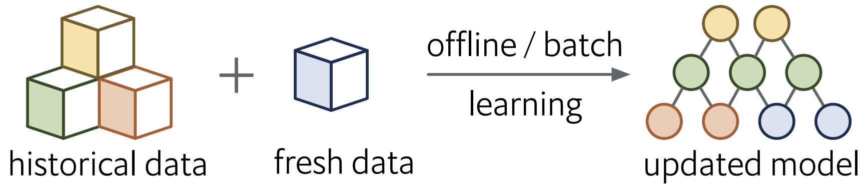
Huangshi Tian, Minchen Yu, Wei Wang @ HKUST

Oct 11, 2018

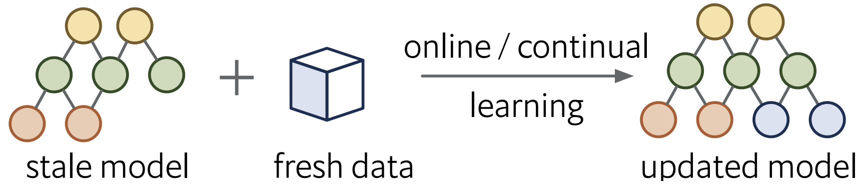
Continual/Online *vs.* Batch/Offline Learning

When fresh data arrive,

- offline learning **trains model from scratch** with all historical data;

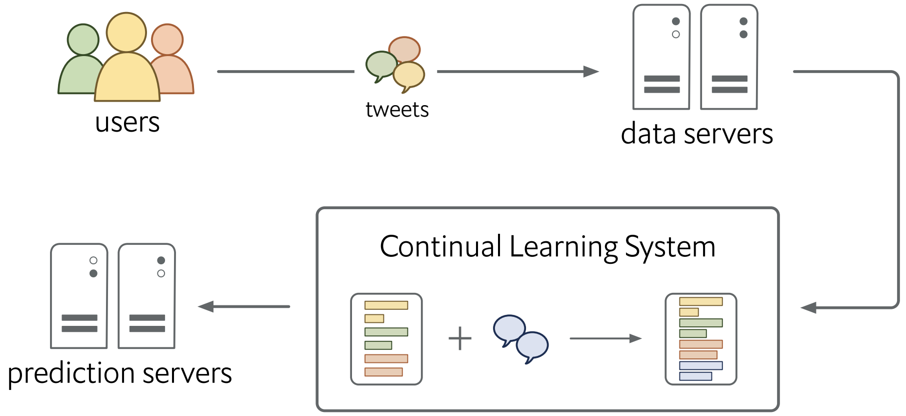


- online learning **updates model** with fresh data.



Case Study: Topic Monitoring

- **Scenario**
  - Users continuously generate tweets;
  - We deploy *topic models* to detect new topics;
  - Topic models are continually updated with new data.

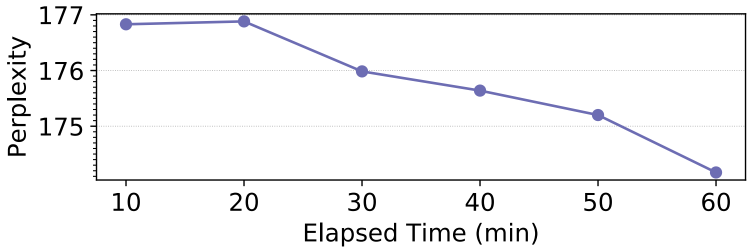


- **Setting**
  - AWS EC2 (c5.4xlarge instance)
  - Latent Dirichlet Allocation (LDA) and a dataset of real-world tweets

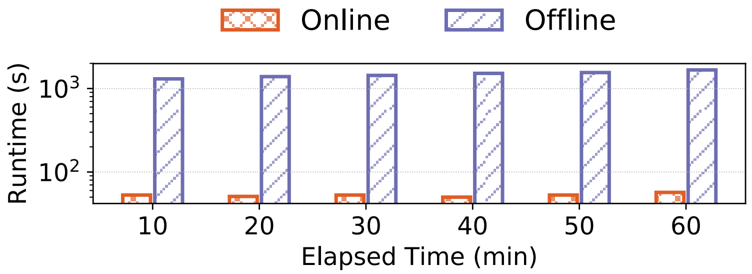
Case Study: Topic Monitoring

- Results
    - *Perplexity* measures the model quality (lower means better).

      Incorporating fresh data *improves* model quality.



    - Online updating takes *much less* time than offline retraining.



Advantage of Online Learning

- **better performance**
  - quickly exploit data recency to improve model quality
  - consume less hardware resources

- **wide application** in industry
  - Microsoft : recommendation, contextual decision makin, click-through rate prediction
  - Google, facebook., twitter : online advertising

Why do we need a platform?

- **no support** from mainstream learning systems
  - scikit learn
  - VOWPAL WABBIT
  - TensorFlow
  - mxnet
- **ad-hoc scripts** bacome status quo

> This becomes particularly challenging when data changes over time and fresh models need to be produced continuously. Unfortunately, such orchestration is often **done ad hoc using glue code and custom scripts** developed by individual teams for specific use cases, leading to **duplicated effort** and **fragile systems** with high technical debt.
>
> —Google

Why do we need a platform?

- **wasted effort** in (re)implementing training loop

Lines of Code in Case Studies

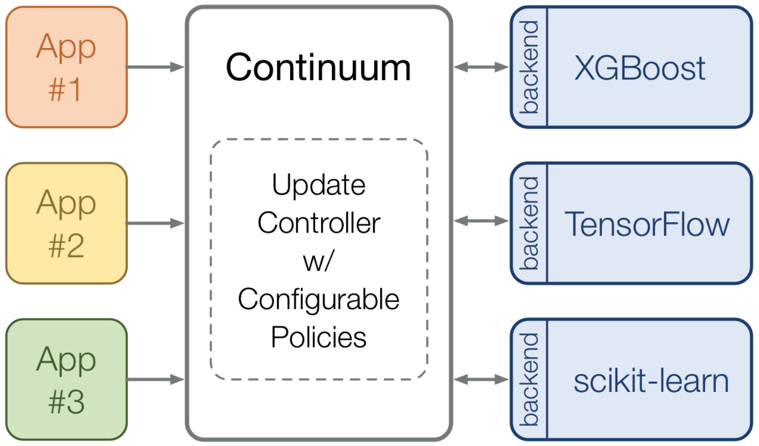
Application	Training Loop	Model Updating
Topic Monitoring	377	56
Friend Suggestion	211	41
Click Prediction	558	44

In need of a general-purpose, automated solution
for continual learning, we present

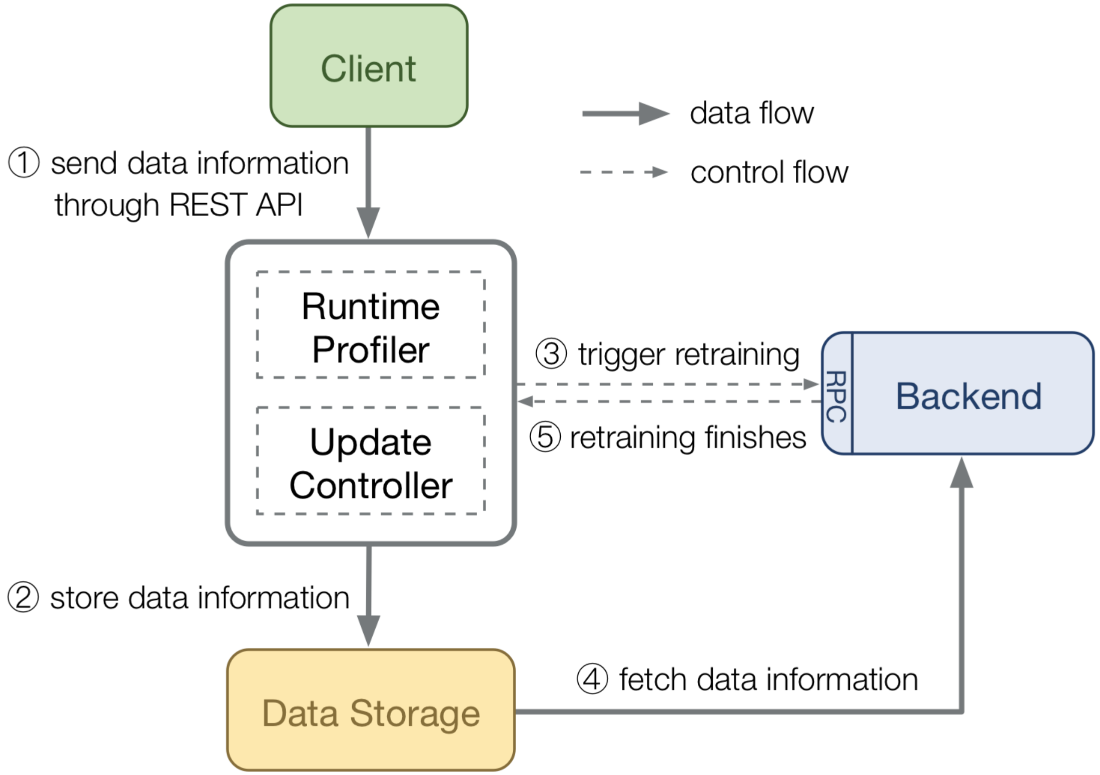
Continuum

System Overview

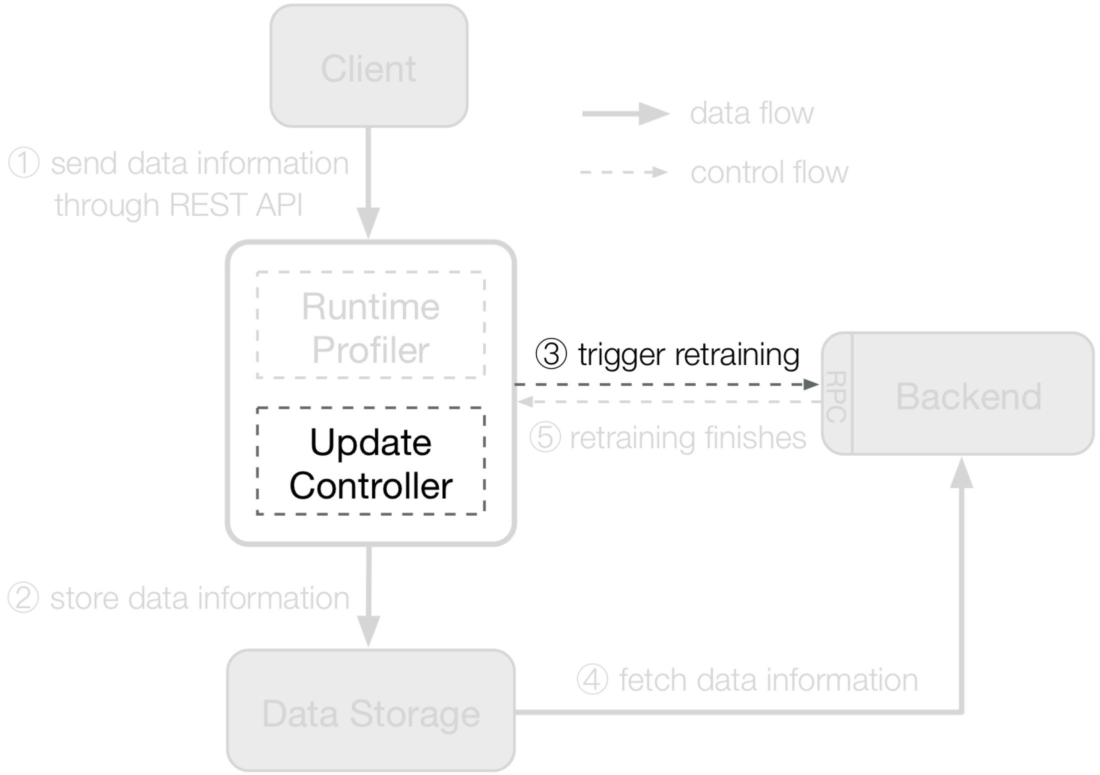
- **automated**: streamlines the process of online learning
- **general-purpose**: applicable to heterogeneous ML frameworks and systems
- **lightweight**: a thin layer on existing systems



Overall Workflow

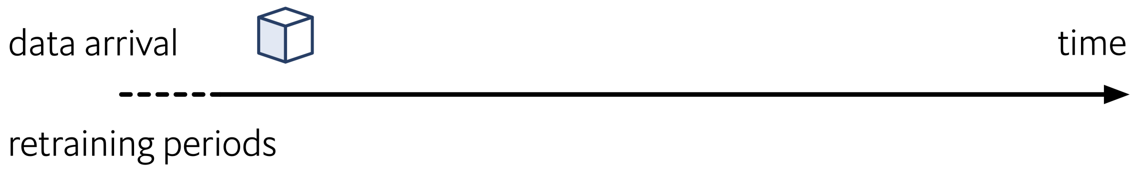


Overall Workflow



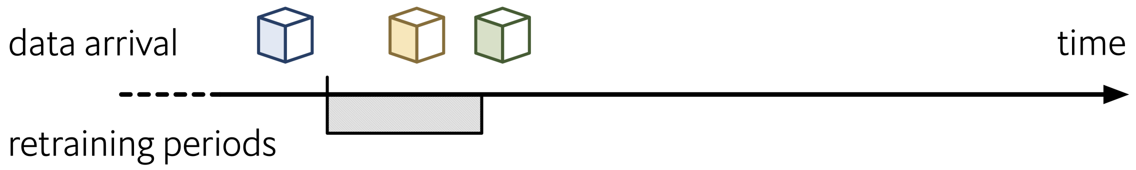
When to Retrain Models?

- **Setting**: As data keep arriving, Continuum determines when to retrain models.



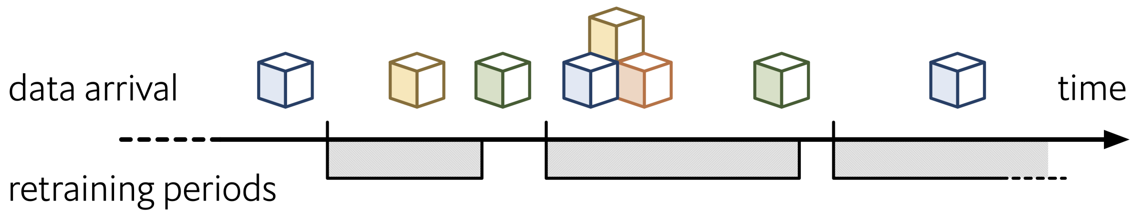
When to Retrain Models?

- **Setting**: As data keep arriving, Continuum determines when to retrain models.



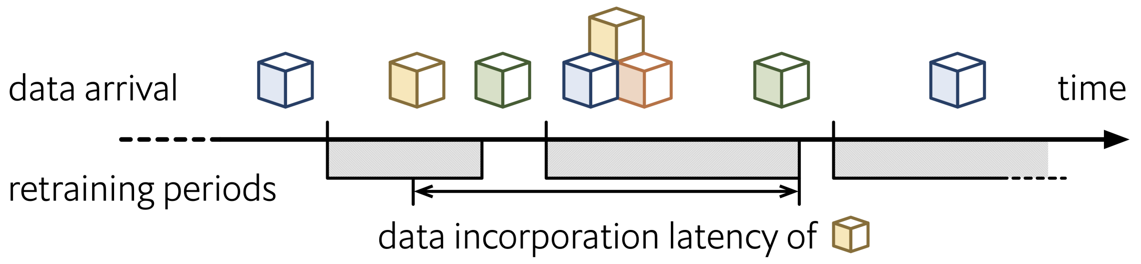
When to Retrain Models?

- **Setting**: As data keep arriving, Continuum determines when to retrain models.



When to Retrain Models?

- **Setting**: As data keep arriving, Continuum determines when to retrain models.



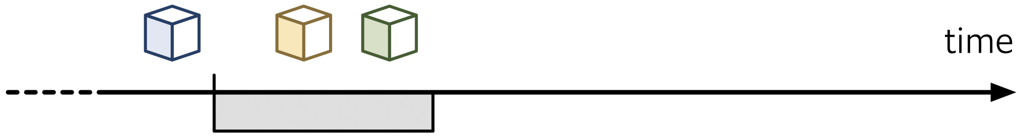
- **Objectives**
  - better model quality → minimize data incorporation latency
  - less hardware cost → minimize training cost (i.e., machine time)

Scenario I: Seeking Fast Data Incoporation

- Naive Approach: **Continuous Update**

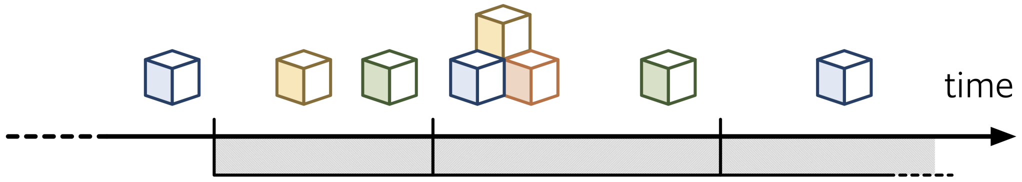
Scenario I: Seeking Fast Data Incoporation

- Naive Approach: **Continuous Update**



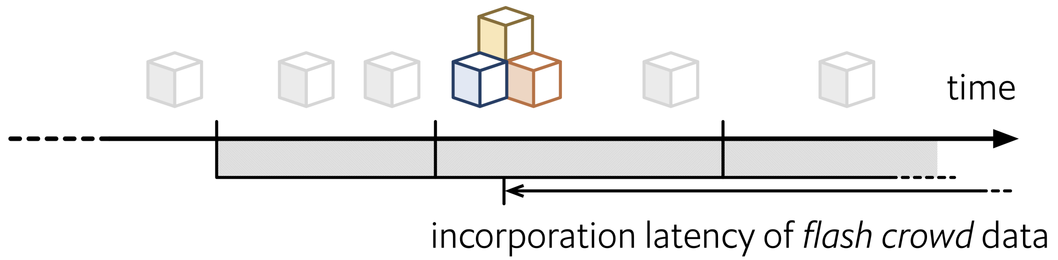
Scenario I: Seeking Fast Data Incoporation

- Naive Approach: **Continuous Update**



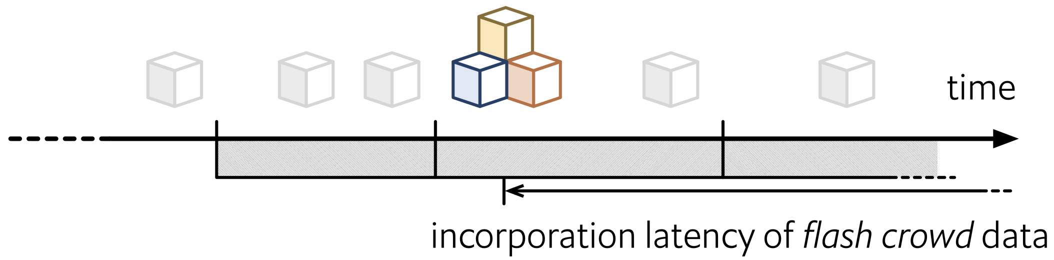
Scenario I: Seeking Fast Data Incoporation

- Naive Approach: **Continuous Update**

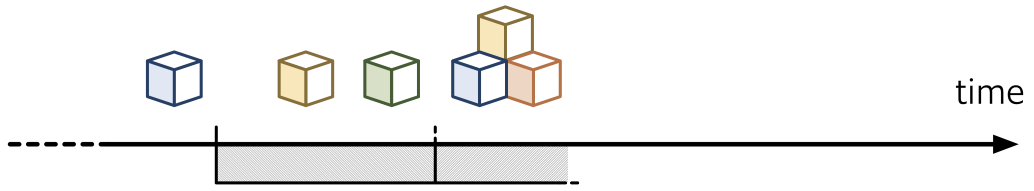


Scenario I: Seeking Fast Data Incoporation

- Naive Approach: **Continuous Update**

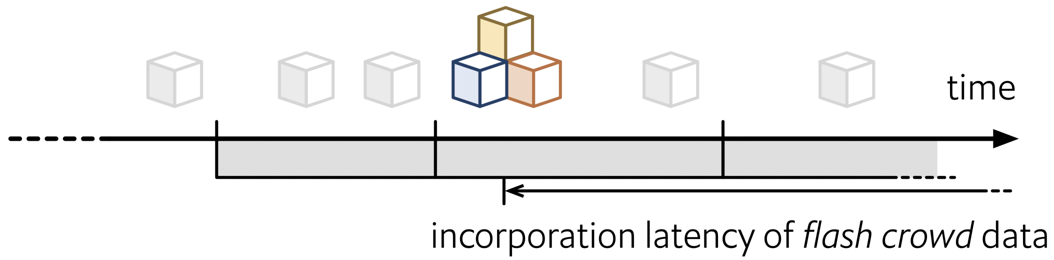


- Proposed Approach: **Best-Effort Policy**

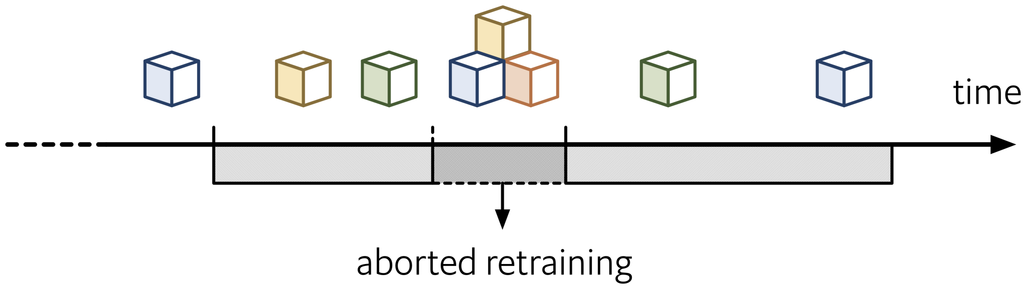


Scenario I: Seeking Fast Data Incoporation

- Naive Approach: **Continuous Update**

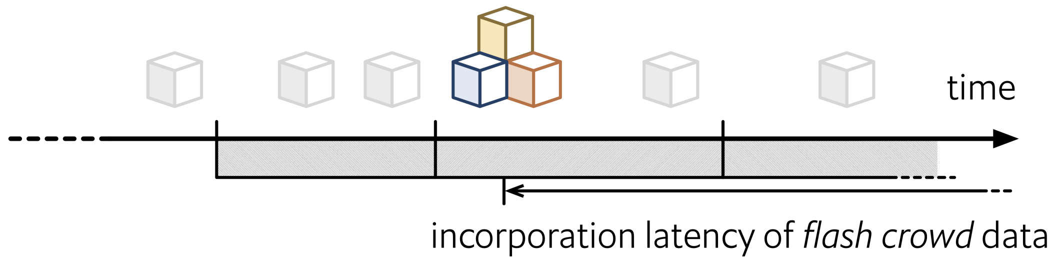


- Proposed Approach: **Best-Effort Policy**

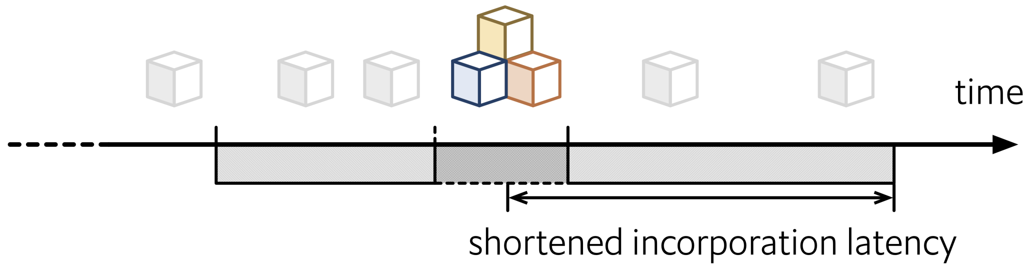


Scenario I: Seeking Fast Data Incoporation

- Naive Approach: **Continuous Update**

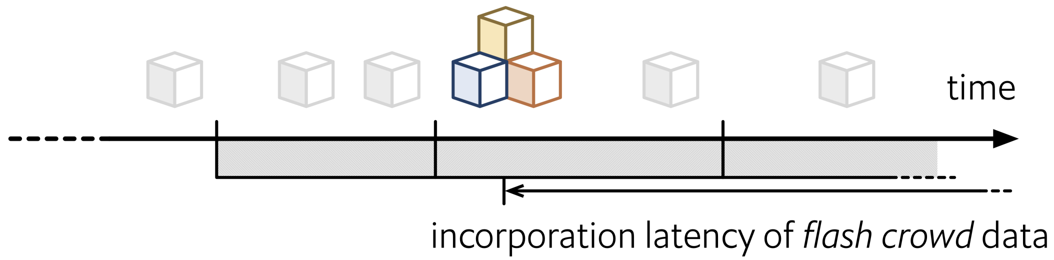


- Proposed Approach: **Best-Effort Policy**

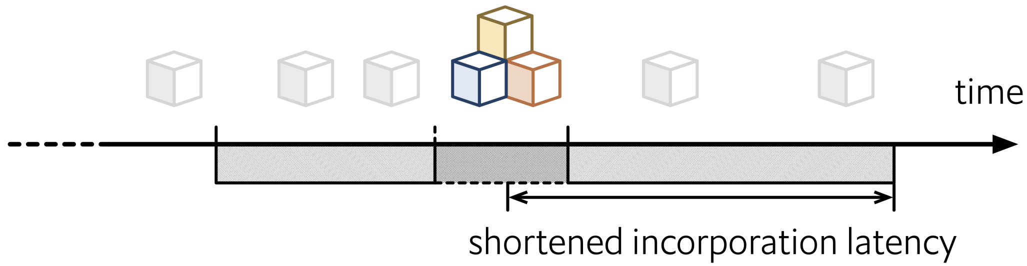


Scenario I: Seeking Fast Data Incoporation

- Naive Approach: **Continuous Update**



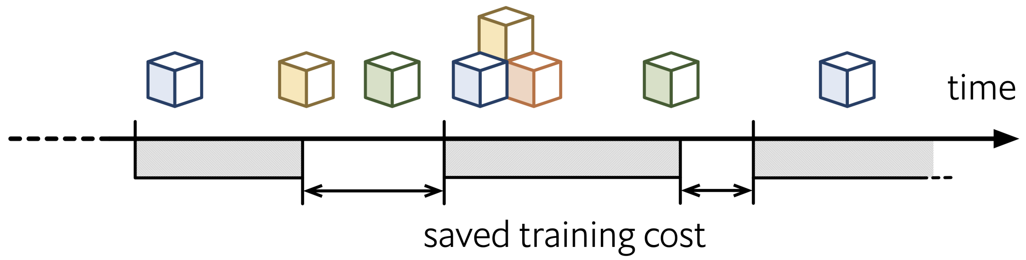
- Proposed Approach: **Best-Effort Policy**



- Potential Problem: high training cost because the machine is always occupied

Scenario II: Saving Cost of Training

- Naive Approach: **Periodic Update**



- Proposed Approach: **Cost-Aware Policy**
  - a regret-based online algorithm
  - jointly optimize the weighted sum of latency and training cost
  - proven to be **2-competitive** (never worse than twice the offline optimum)

Experimental Setting

- Testbed
  - AWS EC2 (c5.4xlarge instance)
- Applications
  - Latent Dirichlet Allocation (LDA) from Mallet + twitter dataset
  - Gradient-Boost Decision Tree (GBDT) from XGBoost + Criteo click dataset
  - Personalized PageRank (PPR) + twitter user dataset
- Methodology
  - Replay data generation and update models under different policies.
- Metrics
  - *incorporation latency* of all data samples
  - *training cost* measured by machine time

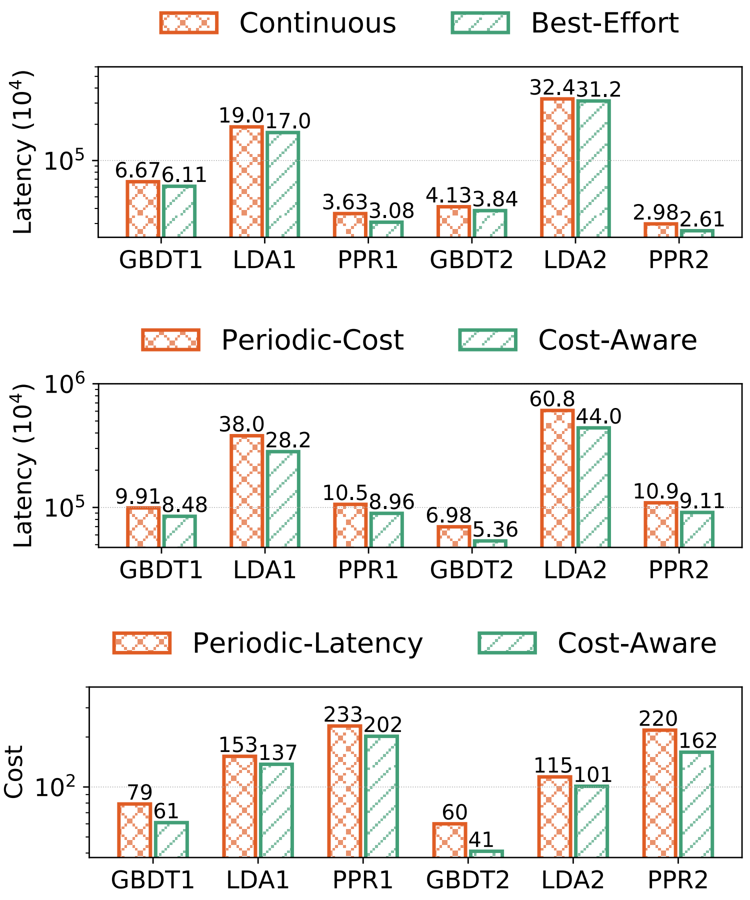
Evaluation of Proposed Policies

Compared with *Continuous Update*, **Best-Effort Policy** can

- reduce the latency by up to 15.2%.

Compared with *Periodic Update*, **Cost-Aware Policy** can

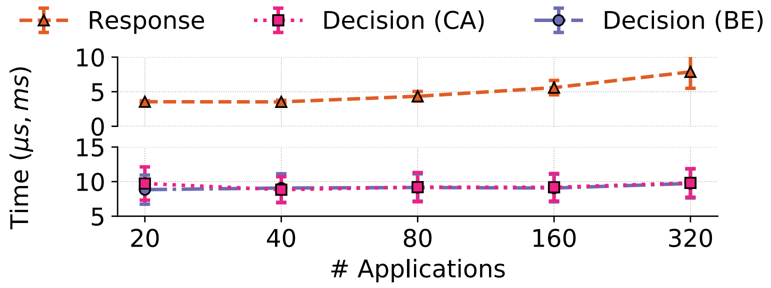
- reduce the latency by up to 28%,
- saves hardware cost by up to 32%.



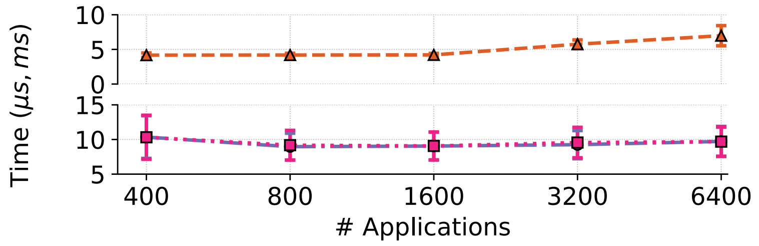
Evaluation of Implemented System

Continuum achieves

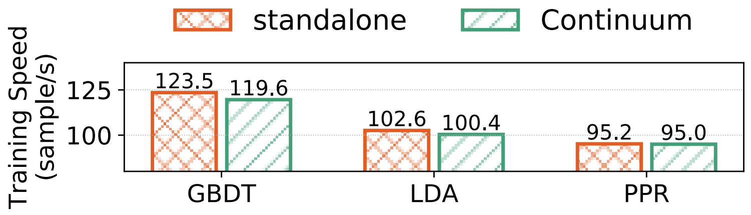
- **high efficiency** in responding to requests and deciding to update models,



- **linear scalability** to a 20-node cluster,



- **low overhead** imposed on backend.



Conclusion

- motivate the need of an online learning platform
- design and implement Continuum
- propose two policies for fast data incorporation and low cost

Source code available at

Thanks for your attention!

Customized Policy

For users who want to decide when to retrain on their own, we provide two mechanisms.

- **REST API** to trigger retraining
    - Users can leverage external information (cluster usage, model monitor).
    - Example: When model quality drops below a threshold, retrain the model.
- **abstract policy class** for extension
    - Users can access internal information (data amount, estimated training time).
    - Users can implement their own decision logic.

Backend Abstraction

- Continuum communicates with backends through an RPC layer.

- The following interface abstracts away the heterogeneity of learning frameworks and systems.

Listing 1 Common interface for backends.

```
interface Backend<X, Y> {
    X retrain(X prev_model, List<Y> data)
}
```