Resource Management for Advanced Data Analytics at Large Scale

Final Public Oral

Haoyu Zhang

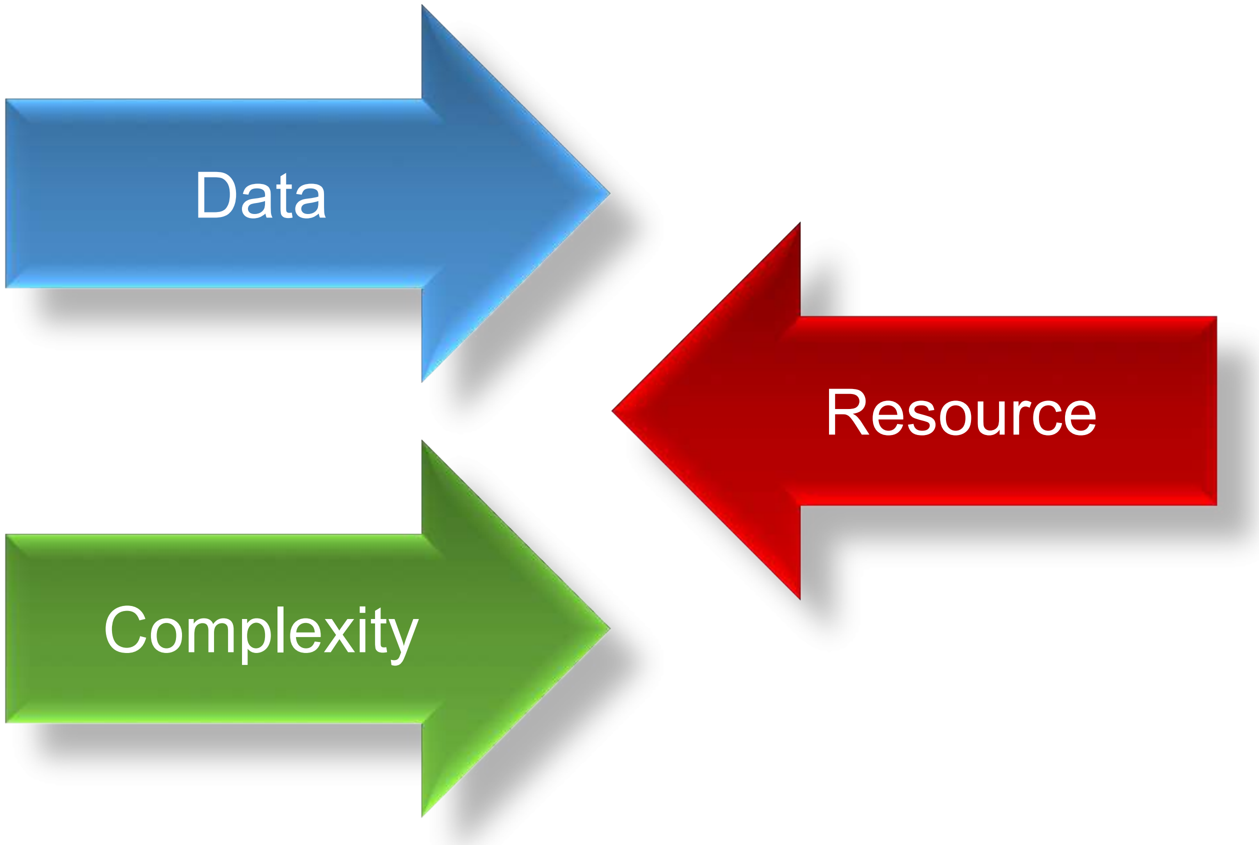
Committee: Mike Freedman (advisor),
Kyle Jamieson, Kai Li, Wyatt Lloyd, Jennifer Rexford

PRINCETON UNIVERSITY

Advanced data analytics: making sense of complex data

- Unstructured, multimodal
  numerical, text, images, videos, …
- High-dimensional, interconnected
  medical, linked social graphs, …
- Growing very fast in volume

- Discover interpretable patterns
- Understand causal relationships
- Make informed predictions and decisions


Data
Resource
Complexity

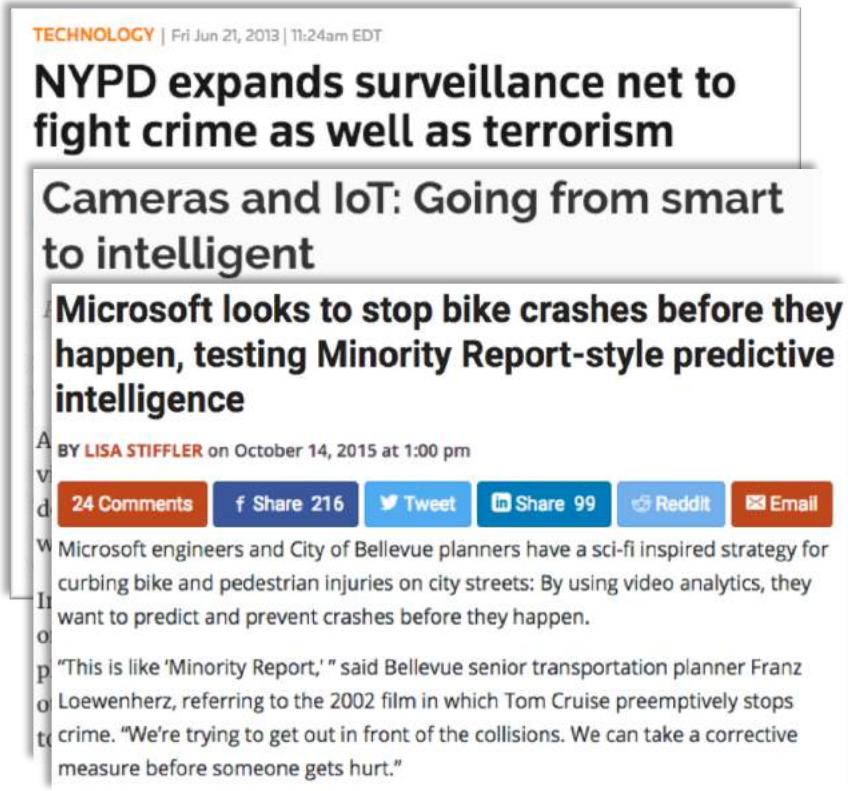
Challenge 1: the growth of data volume

Batch processing

10s PB new data per day for Spark jobs

100s TB new data per day for a single job

Video stream analytics


TECHNOLOGY | Fri Jun 21, 2013 | 11:24am EDT

NYPD expands surveillance net to fight crime as well as terrorism

Cameras and IoT: Going from smart to intelligent

Microsoft looks to stop bike crashes before they happen, testing Minority Report-style predictive intelligence

BY LISA STIFFLER on October 14, 2015 at 1:00 pm

24 Comments | f Share 216 | Tweet | Share 99 | Reddit | Email

Microsoft engineers and City of Bellevue planners have a sci-fi inspired strategy for curbing bike and pedestrian injuries on city streets: By using video analytics, they want to predict and prevent crashes before they happen.

"This is like 'Minority Report,'" said Bellevue senior transportation planner Franz Loewenherz, referring to the 2002 film in which Tom Cruise preemptively stops crime. "We're trying to get out in front of the collisions. We can take a corrective measure before someone gets hurt."

Machine learning

NETFLIX

100^{+}M user ratings of 17,770 movies

IMAGENET

14^{+}M images of 1,000 categories

Challenge 2: the complexity of analytics

Batch processing

>50% batch jobs have multiple stages

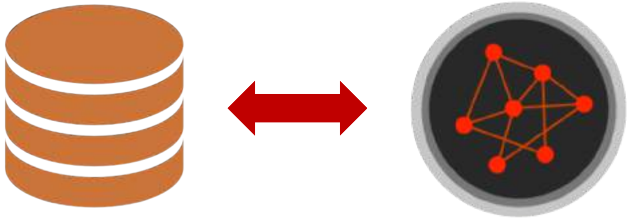
10x larger than available memory

Video stream analytics

1Fps object tracking on 8-core node [1]

30GFlops to recognize objects in image [2]

Machine learning



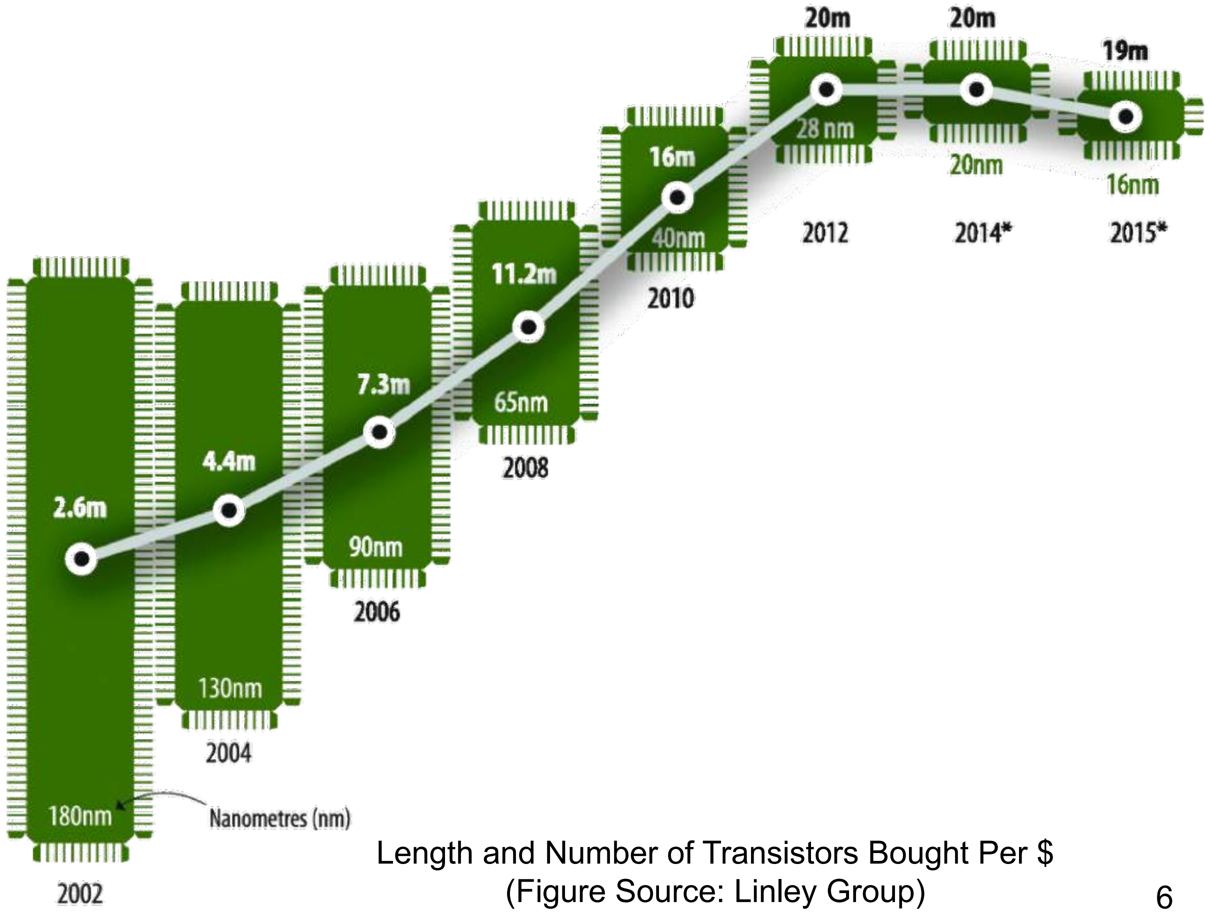
600K training steps to converge [3]

10K hyperparameter combinations to explore [4]

[1] VOT Challenge 2015 Results [2] Simonyan et al. 2014 [3] He et al. 2015 [4] Maclaurin et al. 2015

Challenge 3: limited cluster resources

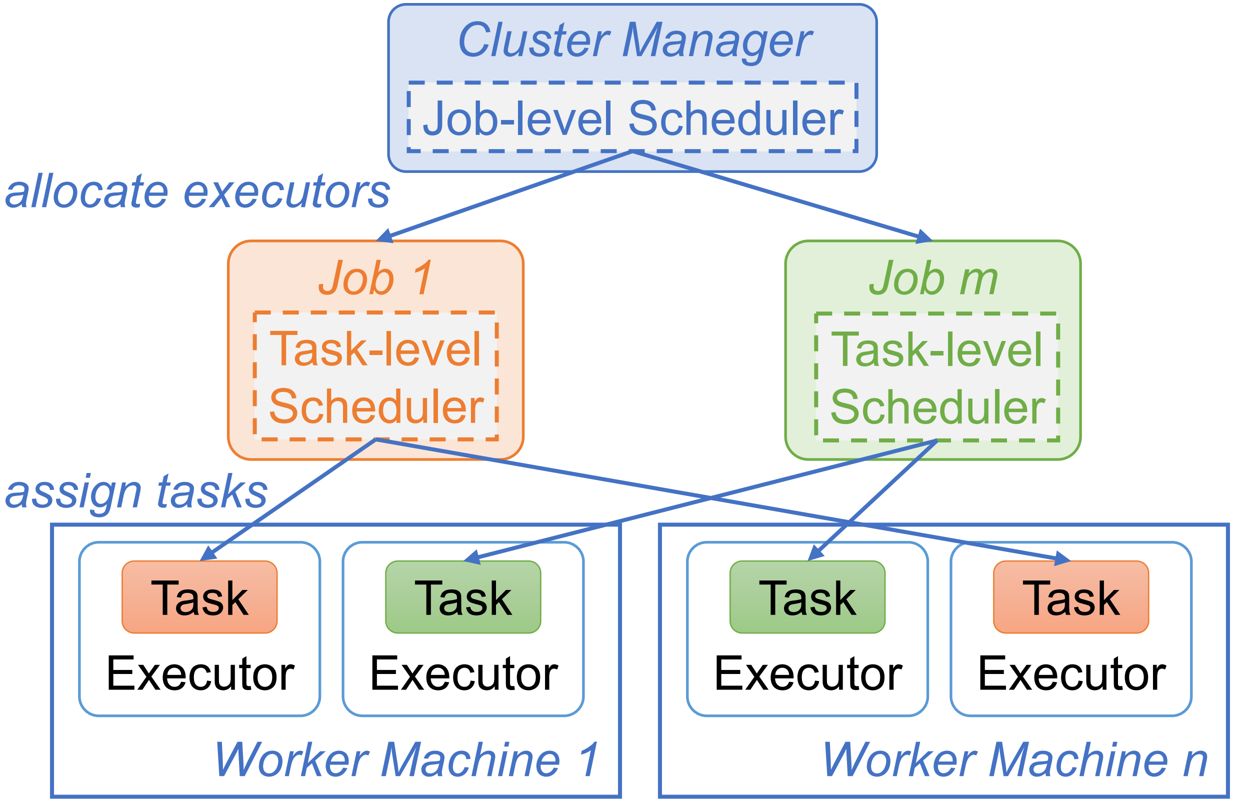
- Our rapidly improving hardware technology is coming to a “grinding halt” [1]
  - DRAM and disk capacity: double once in next decade [2]
  - CPU performance: double in two decades [2]
  - Moore’s Law is ending…



Length and Number of Transistors Bought Per $
(Figure Source: Linley Group)

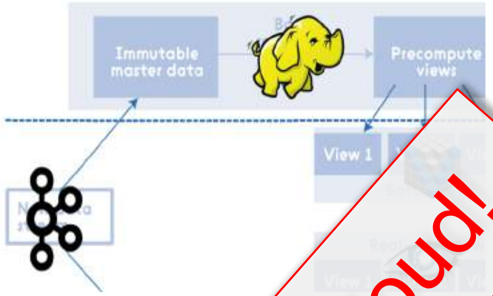
[1] Stoica et al. 2017 [2] Hennessy & Patterson. 6th Edition. 2017

Datacenter resource scheduling



- Treat tasks as black boxes
- Based on general principles
    - fairness, locality, load balancing, …

New opportunities to optimize scheduling



Occupying the cloud!

- Batch processing
  - large amount of fragmented I/O in multi-stage jobs

largest Apache Spark deployment known has 8,000 nodes

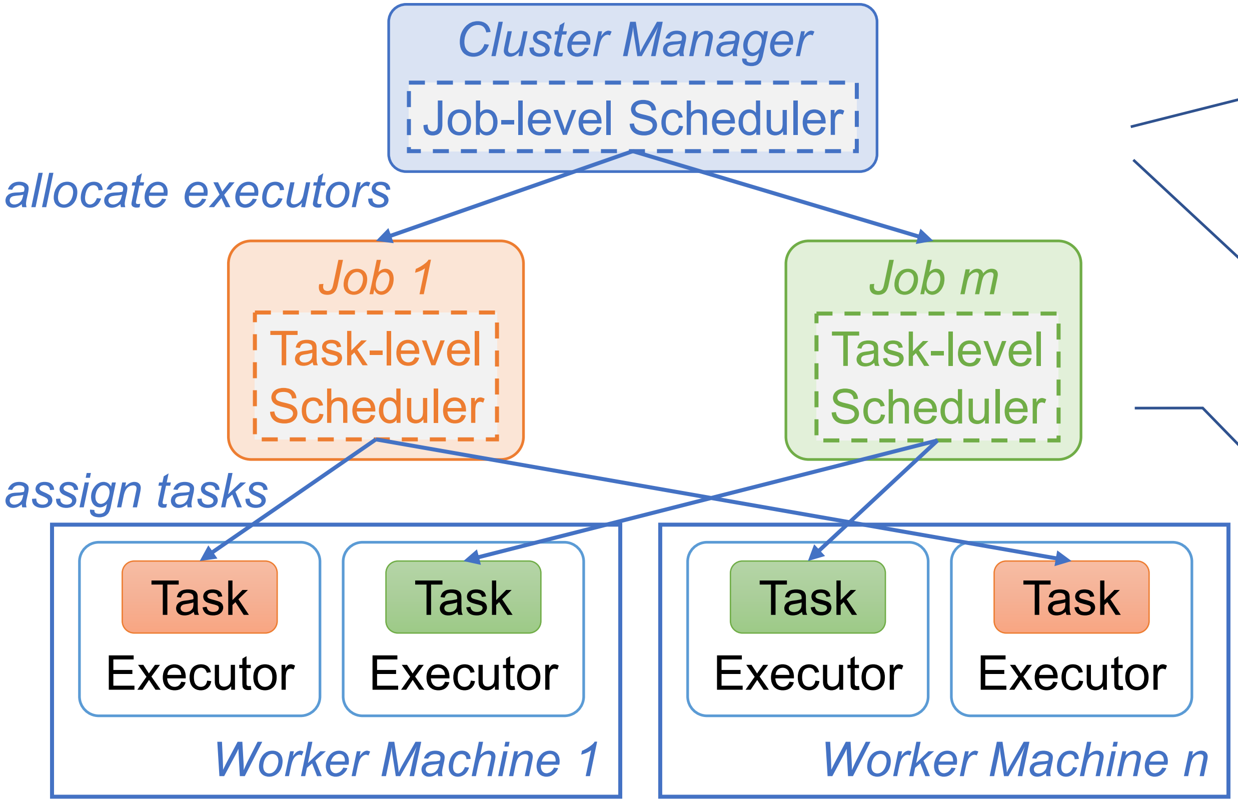
- Video stream analytics
  - quality-resource-delay tradeoffs between queries

live analytics deployed on public & private cloud

- Machine learning
  - iterative training process with diminishing returns

TPU, Big Basin in datacenters for ML jobs

In this talk



VideoStorm: Live Video Analytics [NSDI ’17]

SLAQ: Quality-Driven ML Scheduling [SoCC ’17 🏆]

Riffle: Optimized Shuffle Service [EuroSys ’18]

Riffle: Optimized Shuffle Service for Large-Scale Data Analytics

Haoyu Zhang, Brian Cho, Ergin Seyfe, Avery Ching, Michael J. Freedman

European Conference on Computer Systems (EuroSys ’18)

PRINCETON UNIVERSITY

facebook

Batch analytics systems are widely used

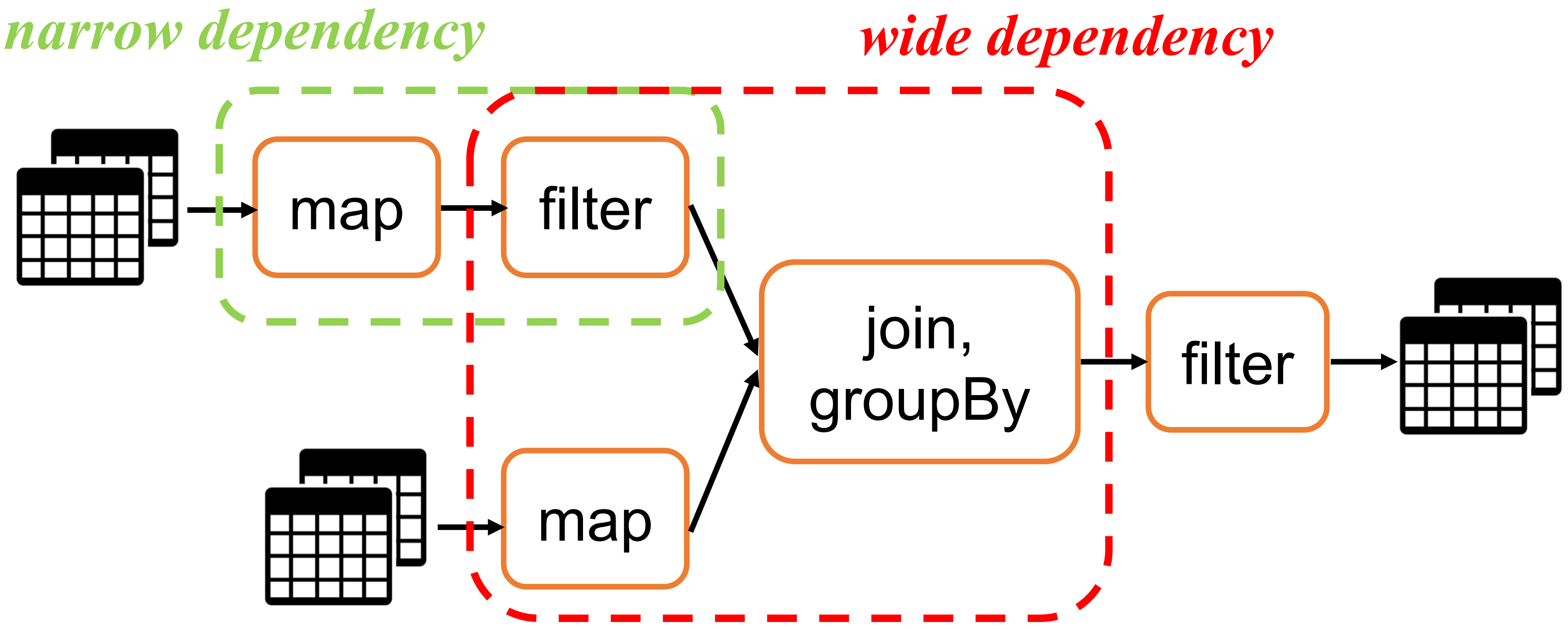
- Large-scale SQL queries
- Custom batch jobs
- Pre-/Post-processing for ML

At facebook

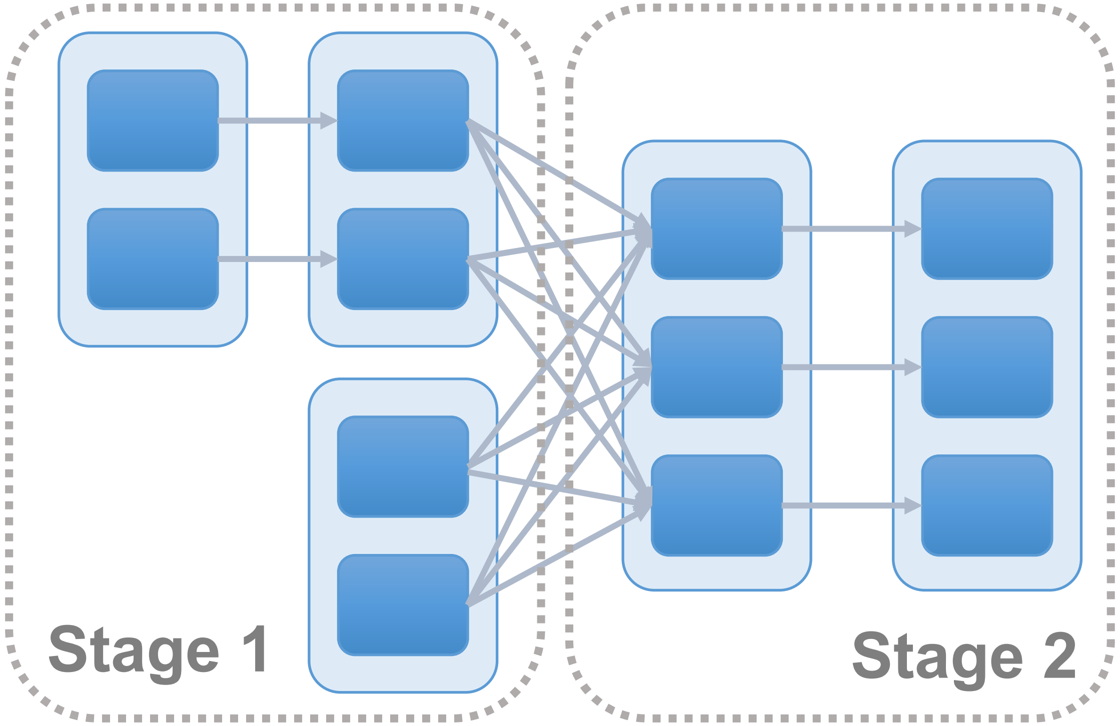
10s of PB new data is generated every day for batch processing

100s of TB data is added to be processed by a single job

Batch analytics jobs: logical graph

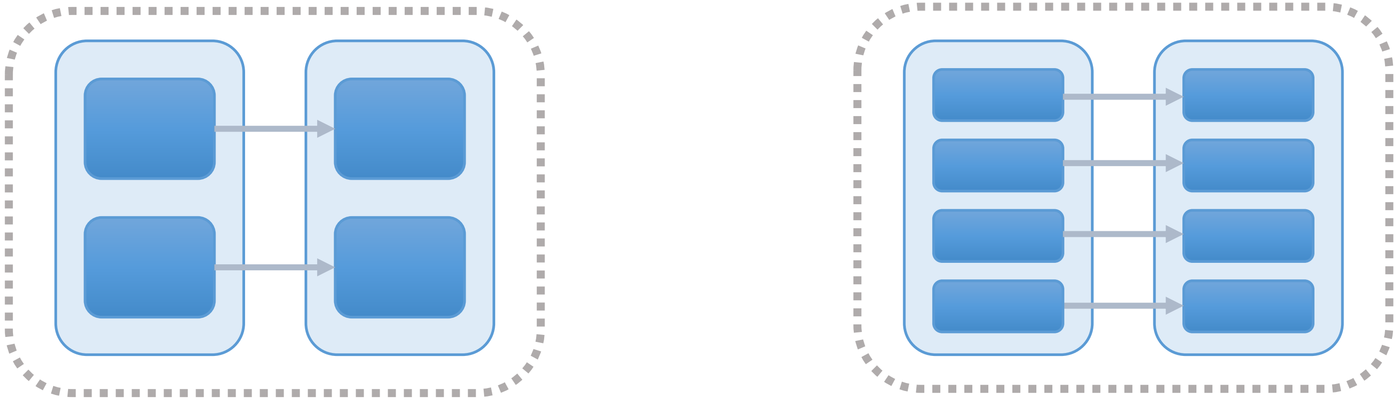


Batch analytics jobs: DAG execution plan



- Shuffle: all-to-all communication between stages
- >10x larger than available memory, strong fault tolerance requirements
  → on-disk shuffle files

The case for tiny tasks



- Benefits of slicing jobs into small tasks
  - Improve parallelism [Tinytasks HotOS 13] [Subsampling IC2E 14] [Monotask SOSP 17]
  - Improve load balancing [Sparrow SOSP 13]
  - Reduce straggler effect [Dolly NSDI 13] [SparkPerf NSDI 15]

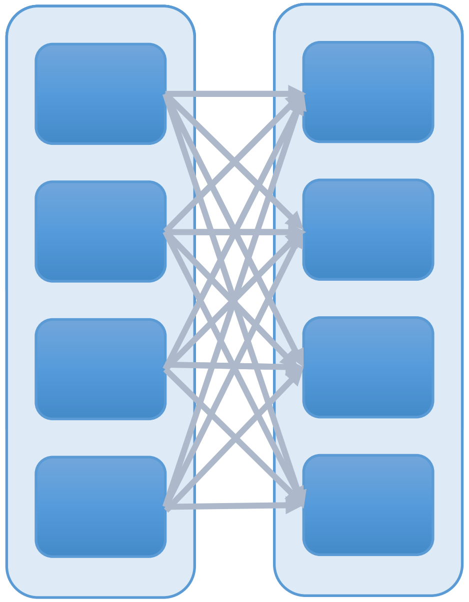
The case against tiny tasks

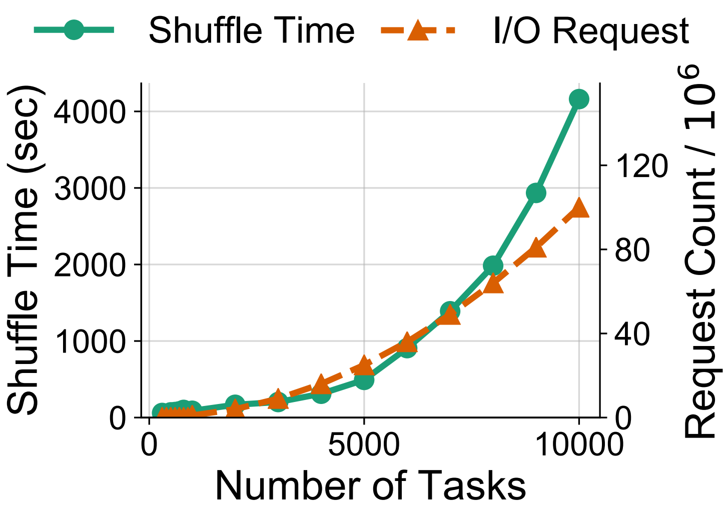
> *Although we were able to run the Spark job with such a high number of tasks, we found that there is significant performance degradation when the number of tasks is too high.*

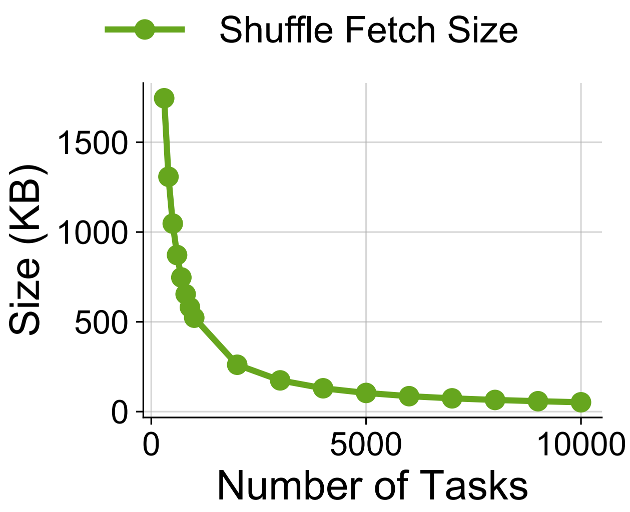
- Engineering experience often argues against running too many tasks
  - Medium scale → very large scale (10x larger than memory space)
  - Single-stage jobs → multi-stage jobs (> 50%)

[*] Apache Spark @Scale: A 60 TB+ Production Use Case. https://tinyurl.com/yadx29gl

Shuffle I/O grows *quadratically* with data

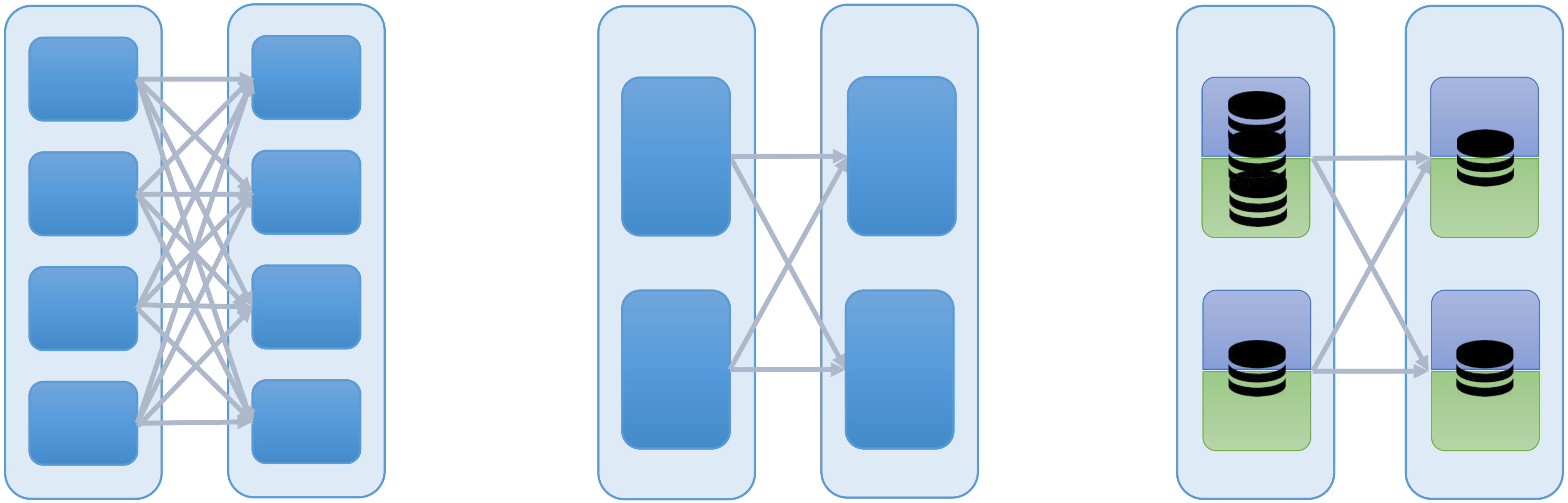






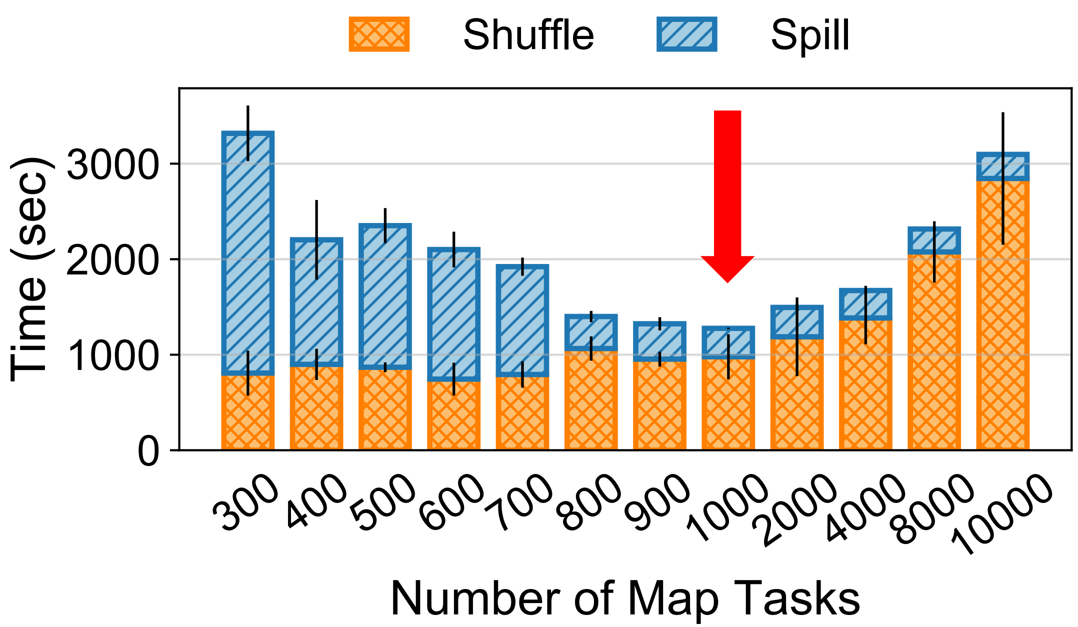
- Large amount of fragmented I/O requests
  - Adversarial workload for hard drives!

Strawman: fix number of tasks in a job

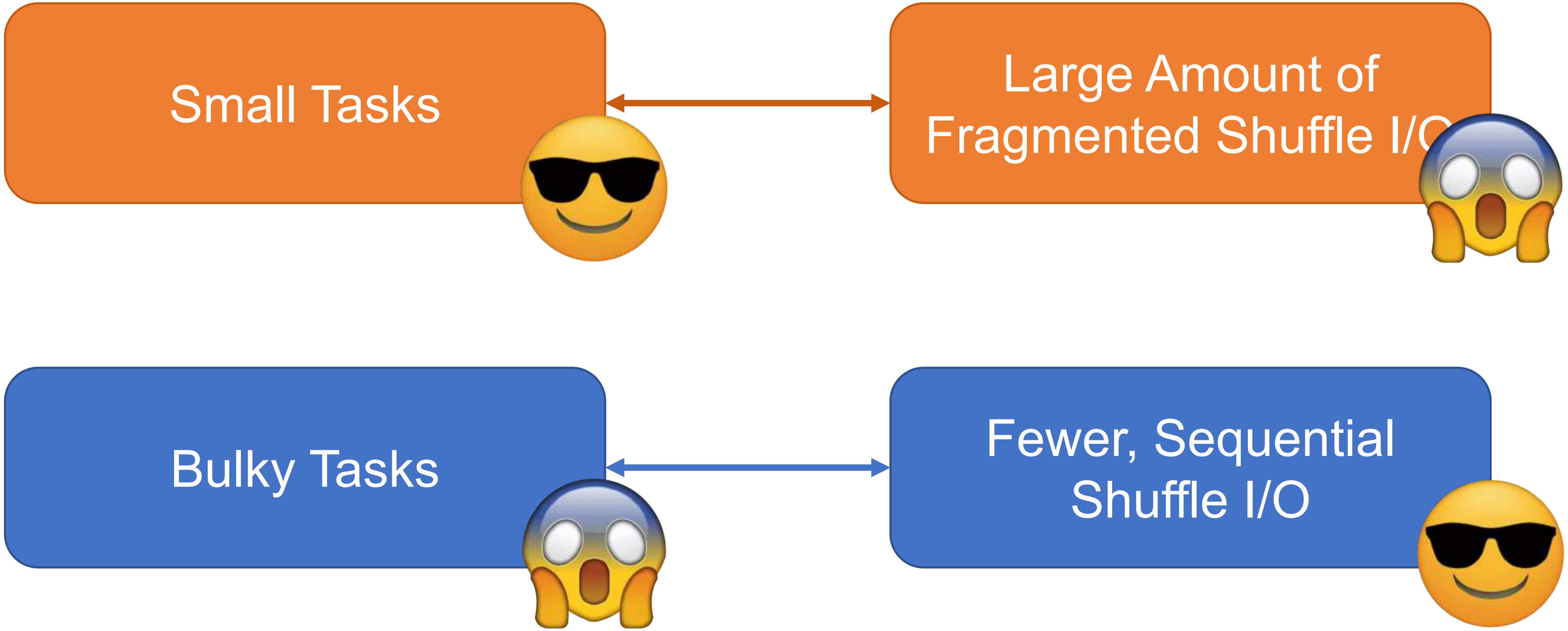


- Tasks spill intermediate data to disk if data splits exceed memory capacity
- Larger task execution reduces shuffle I/O, but increases spill I/O

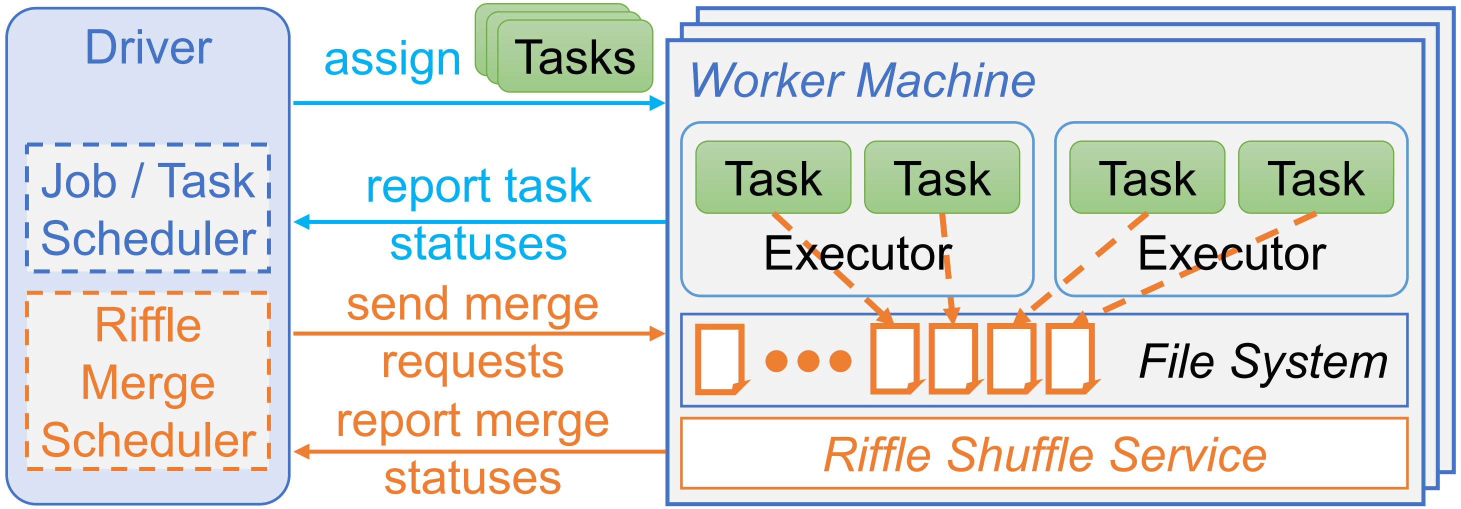
Strawman: tune number of tasks in a job



- Need to retune when input data volume changes for each individual job
- Bulky tasks can be detrimental [Dolly NSDI 13] [SparkPerf NSDI 15] [Monotask SOSP 17]
  - straggler problems, imbalanced workload, garbage collection overhead


Small Tasks
Large Amount of Fragmented Shuffle I/O
Bulky Tasks
Fewer, Sequential Shuffle I/O

Riffle: optimized shuffle service



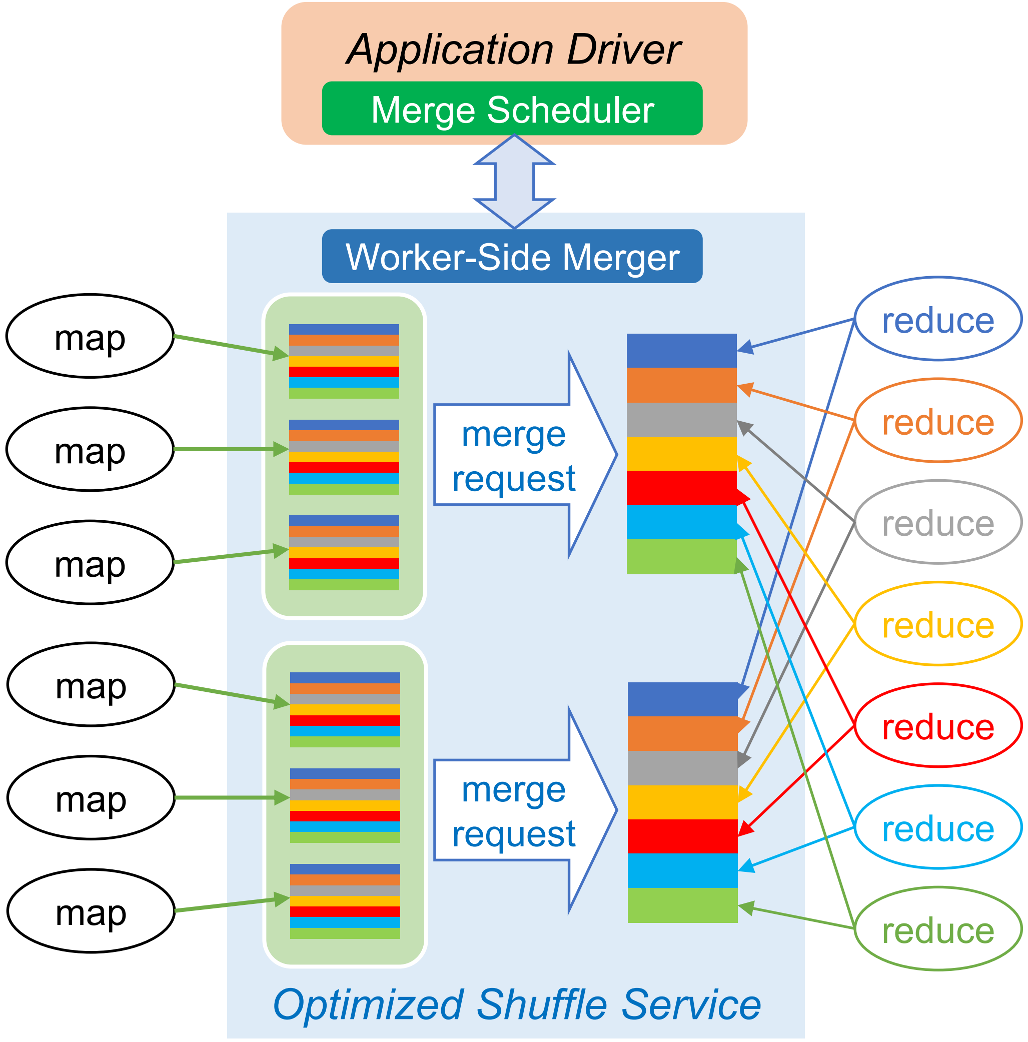
- Riffle shuffle service: a long running instance on each physical node
- Riffle scheduler: keeps track of shuffle files and issues merge requests

Riffle: optimized shuffle service

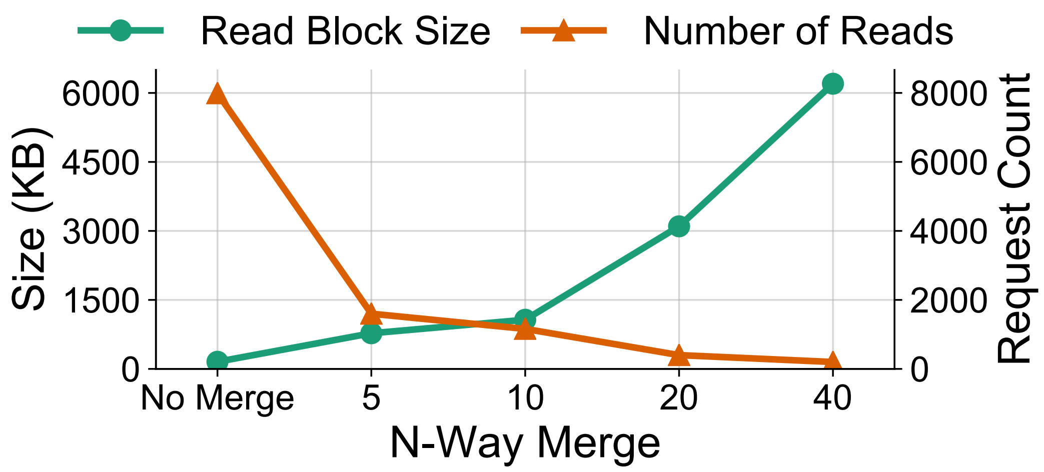
- When receiving a merge request

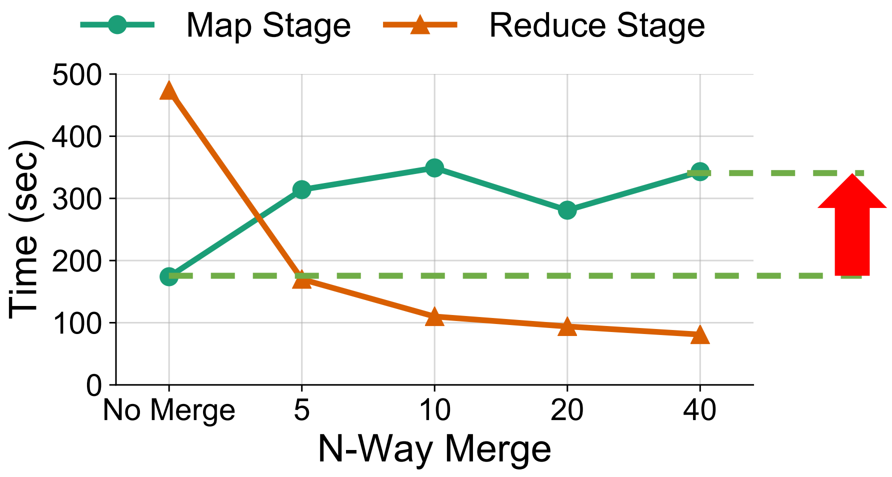
1. Combines small shuffle files into larger ones
2. Keeps original file layout

- Reducers fetch fewer, large blocks instead of many, small blocks



Results with merge operations on synthetic workload

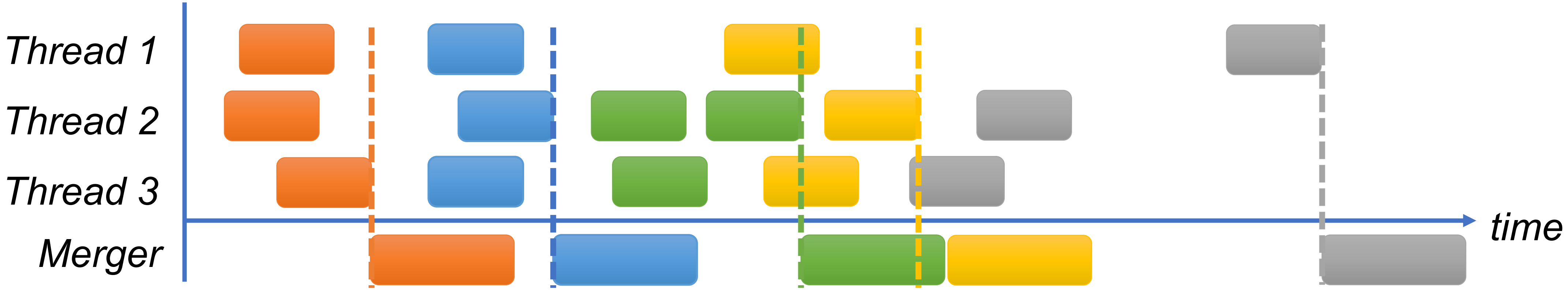




- Riffle reduces number of fetch requests by 10x
- Reduce stage -393s, map stage +169s → job completes 35% faster

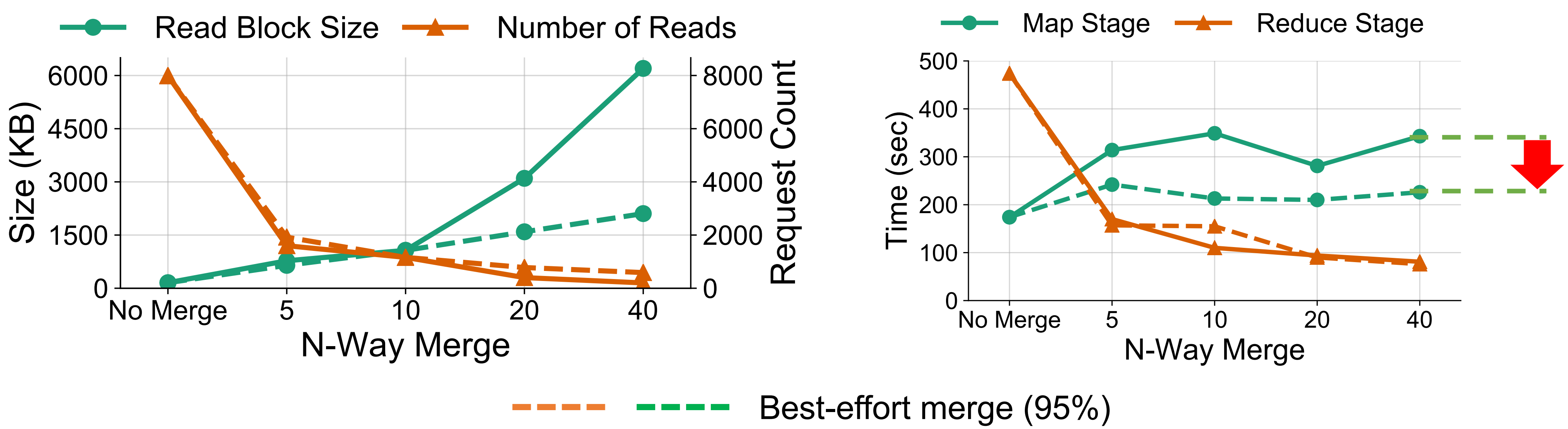
Best-effort merge

- Observation: slowdown in map stage is mostly due to stragglers



- Best-effort merge: mixing merged and unmerged shuffle files
  - When number of finished merge requests is larger than a user specified percentage threshold, stop waiting for more merge results

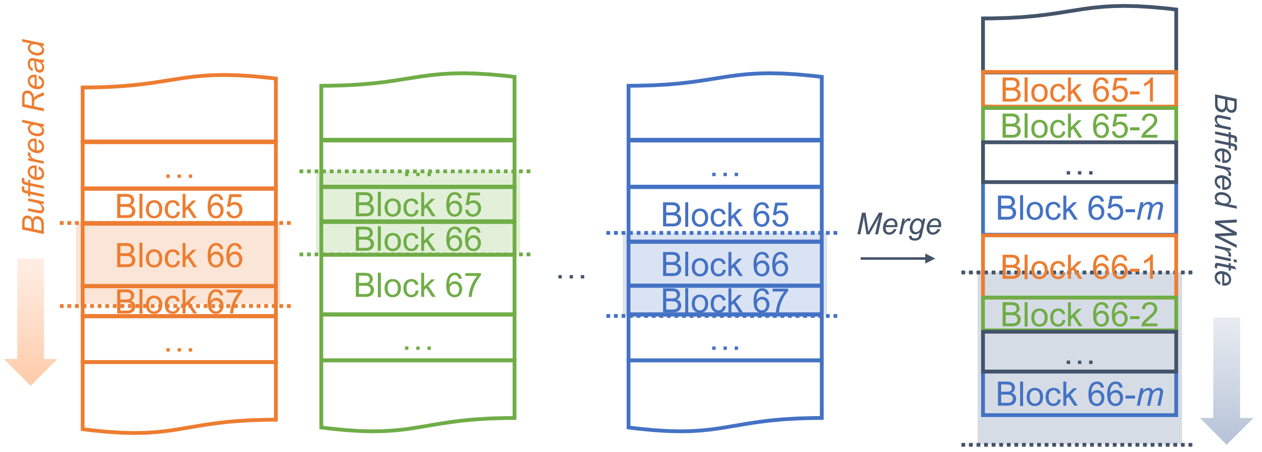
Results with best-effort merge

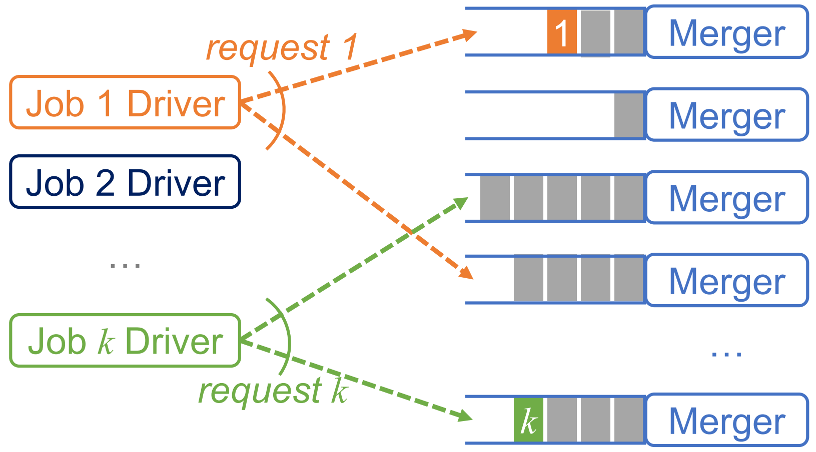


- Reduce stage -393s, map stage +52s → job completes 53% faster
  - Riffle finishes job with only ~50% of cluster resources!

Additional enhancements

- Handling merge operation failures
- Efficient memory management
- Balance merge requests in clusters





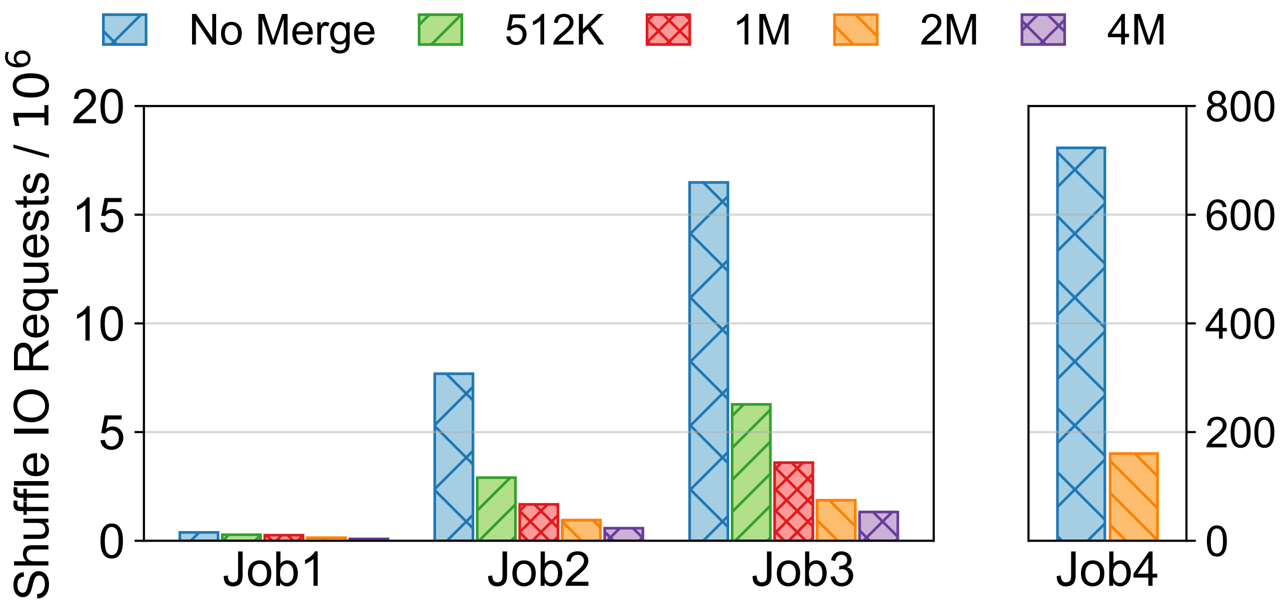
Experiment setup

- Testbed: Spark on a 100-node cluster
  - Each node has 56 CPU cores, 256GB RAM, 10Gbps Ethernet links
  - Each node runs 14 executors, each with 4 cores, 14GB RAM

- Workload: 4 representative production jobs at Facebook

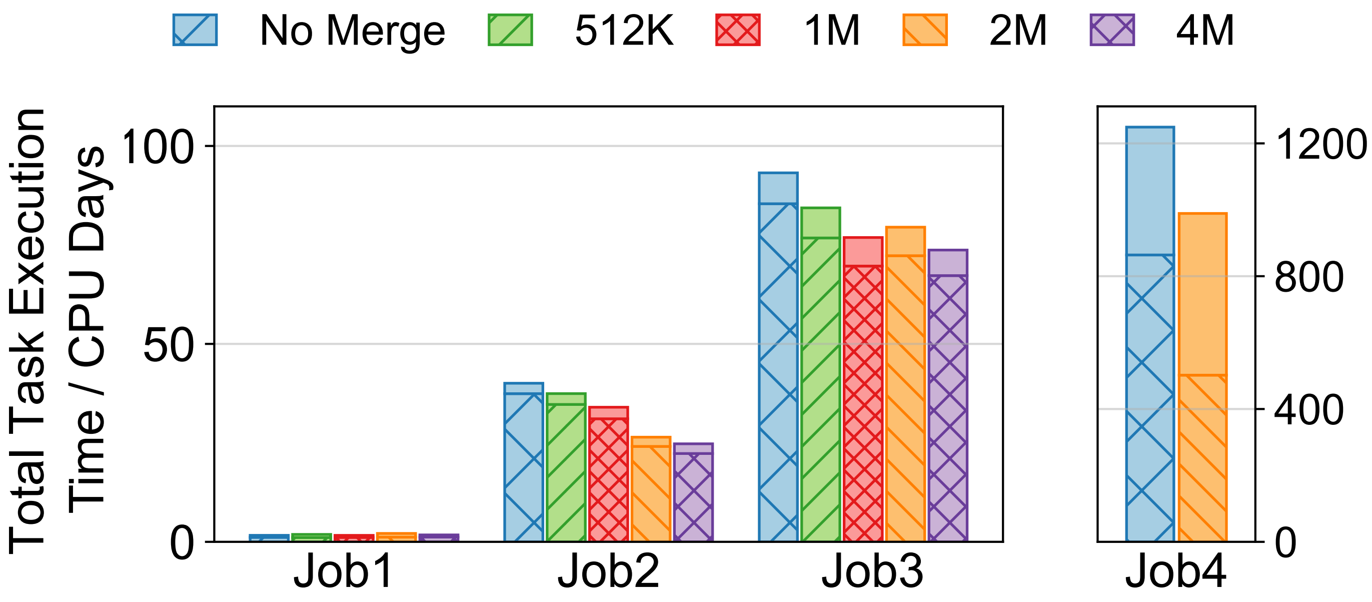
	Data	Map	Reduce	Block
1	167.6 GB	915	200	983 K
2	1.15 TB	7,040	1,438	120 K
3	2.7 TB	8,064	2,500	147 K
4	267 TB	36,145	20,011	360 K

Reduction in shuffle I/O requests



- Riffle reduces # of I/O requests by 5--10x for medium / large scale jobs

Savings in end-to-end job completion time

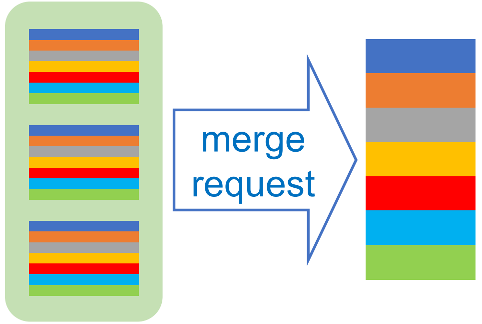


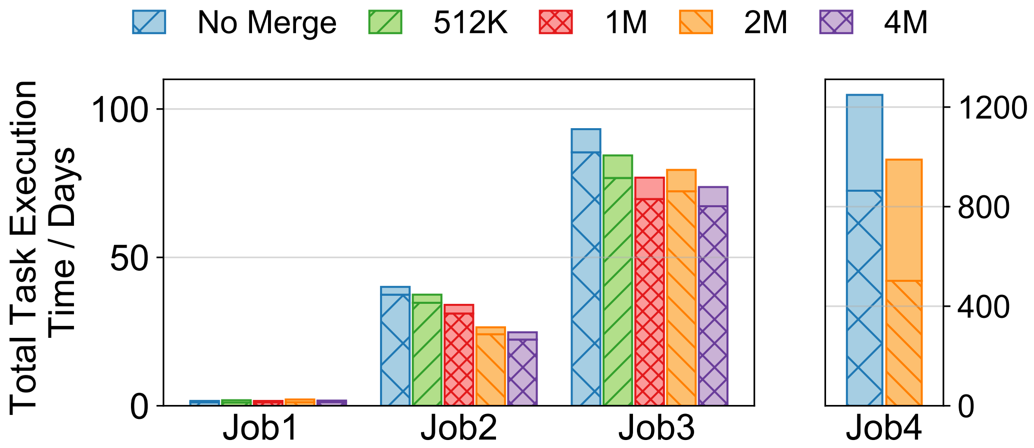
- Map stage time is almost not affected (with best-effort merge)
- Reduces job completion time by 20--40% for medium / large jobs

Part I Conclusion

- Shuffle I/O becomes scaling bottleneck for multi-stage jobs

- Efficiently schedule merge operations, mitigate merge stragglers





- Riffle is deployed for Facebook’s production jobs processing PBs of data

Live Video Analytics at Scale with Approximation and Delay-Tolerance

Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik,
Matthai Philipose, Paramvir Bahl, Michael J. Freedman

USENIX Symposium on Networked Systems Design and Implementation (NSDI ’17)

PRINCETON UNIVERSITY

Microsoft

Video analytics queries

Intelligent Traffic System

AMBER Alert

TOLL-BY-PLATE

Electronic Toll Collection

Video Doorbell

Video query: a pipeline of *transforms*

- Example: traffic counter pipeline

→ *transform* decode → *transform* detect object → *transform* track object → *transform* count object

Video queries are expensive in resource usage

- Example: traffic counter pipeline

- When processing *thousands* of video streams in multi-tenant clusters
  - How to reduce processing cost of a query?
  - How to manage resources efficiently across queries?

Vision algorithms are intrinsically *approximate*

- Knobs: parameters / implementation choices for transforms

Frame Rate

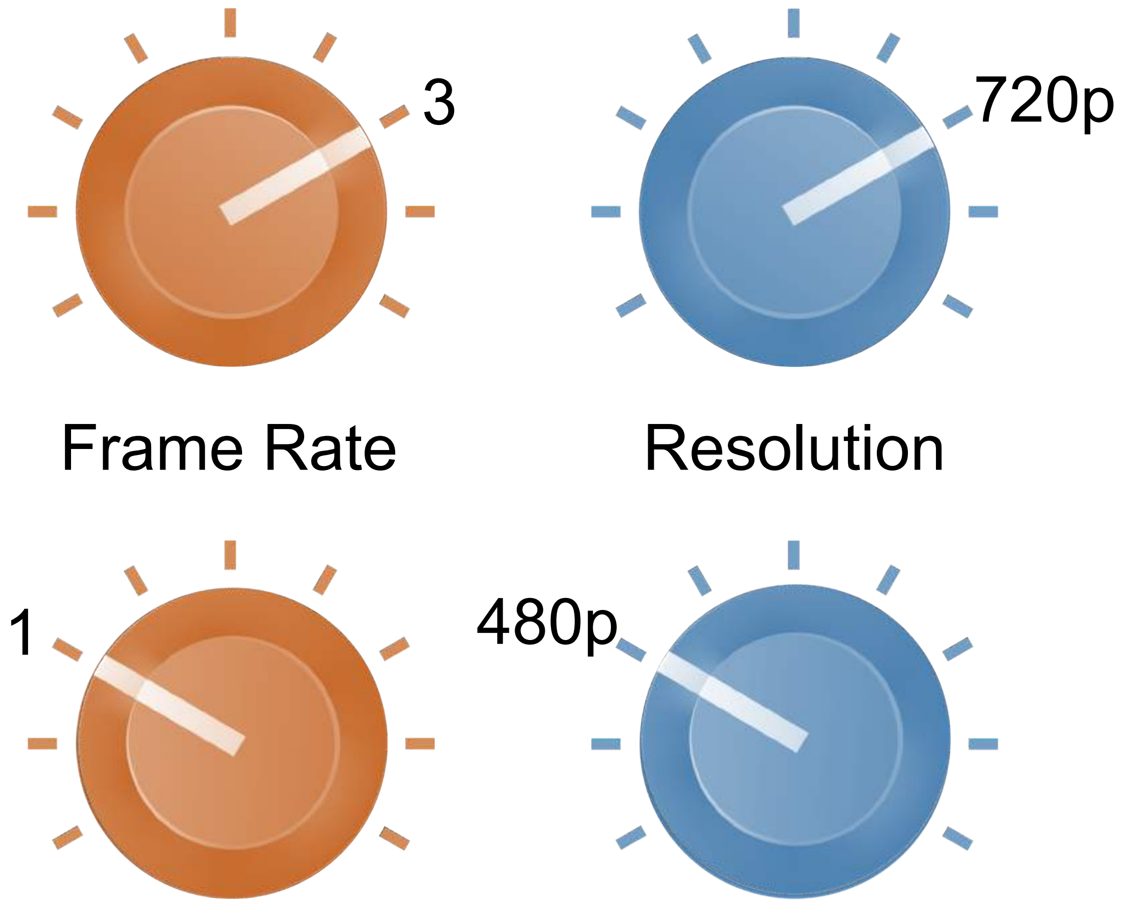
Resolution

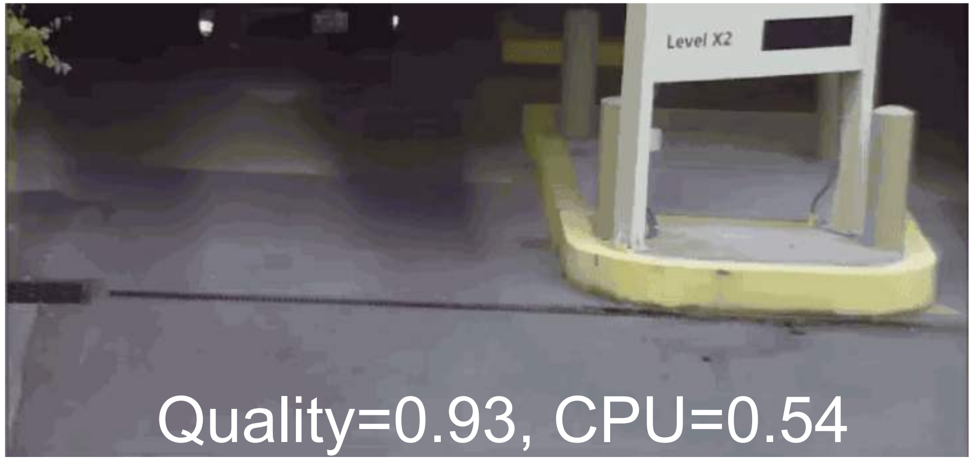
Window Size

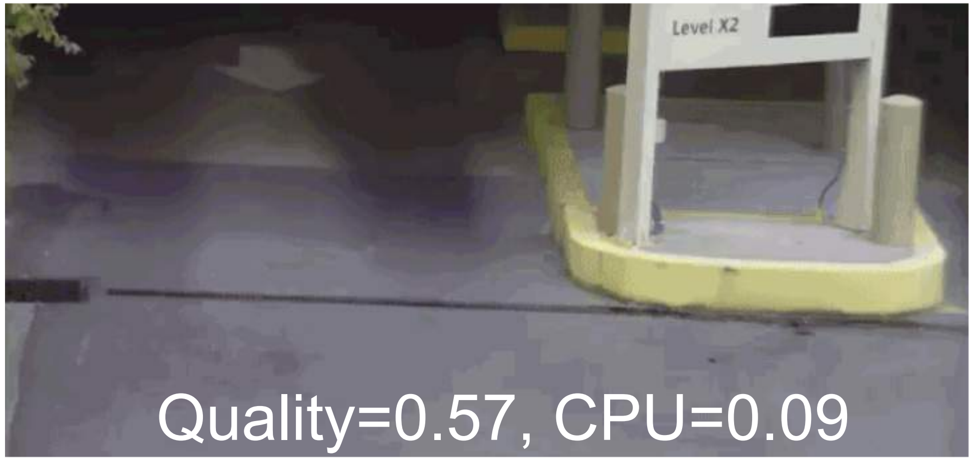
Mapping Metric

  - License plate reader → window size
  - Car tracker → mapping metric
  - Object classifier → DNN model
- Query configuration: a combination of knob values

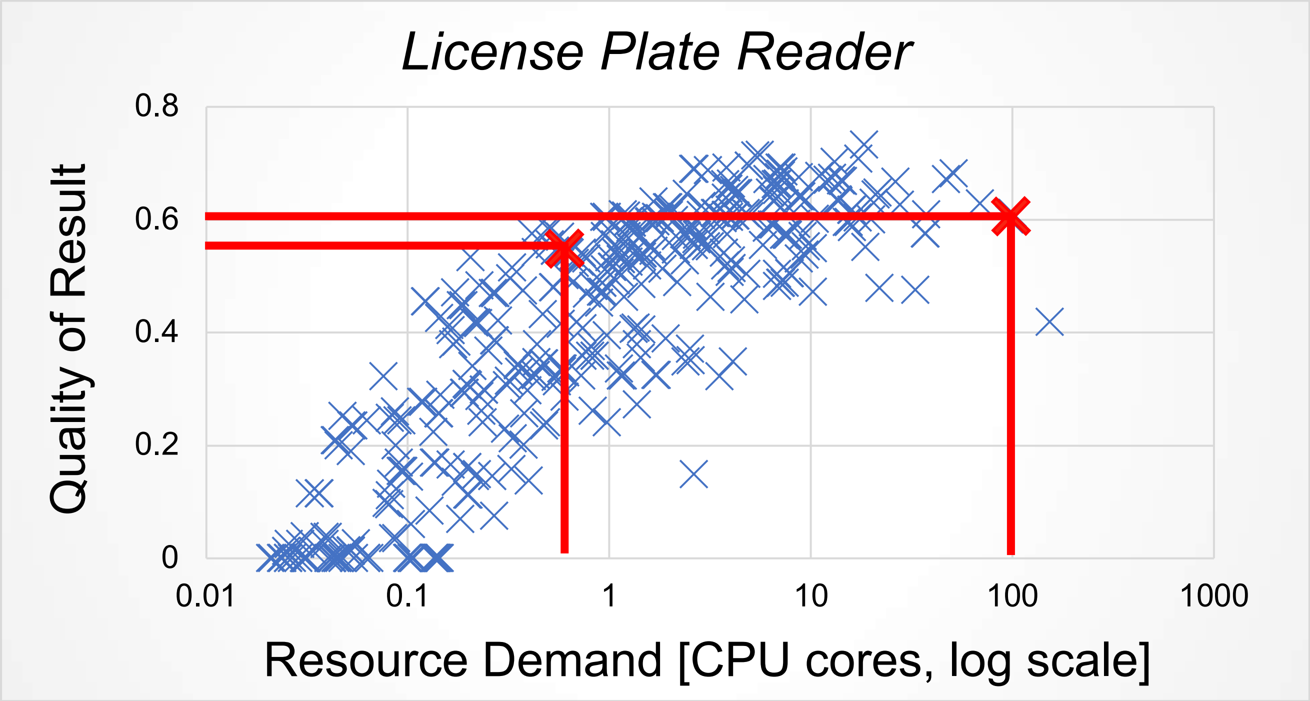
Knobs impact quality and resource usage







Tuning the knobs all together



- Orders of magnitude cheaper resource demand for little quality drop
- No analytical models to predict resource-quality tradeoff
  - Different from approximate SQL queries

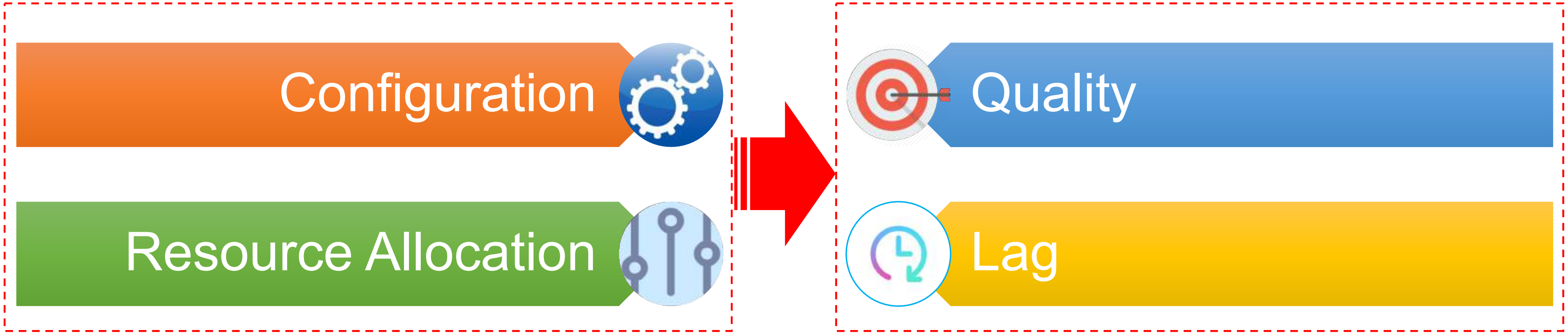
Diverse quality and lag requirements

Lag: time difference between frame arrival and frame processing

	Toll Collection	Intelligent Traffic	AMBER Alert
Quality?	High	Moderate	High
Lag?	Hours	Few Seconds	Few Seconds

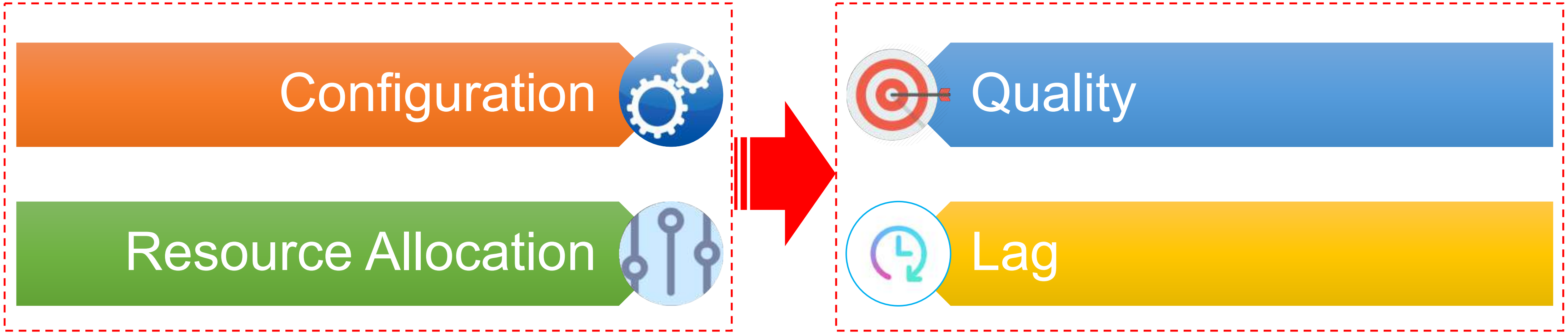
Goal

Decide configuration and resource allocation to maximize quality and minimize lag within the resource capacity

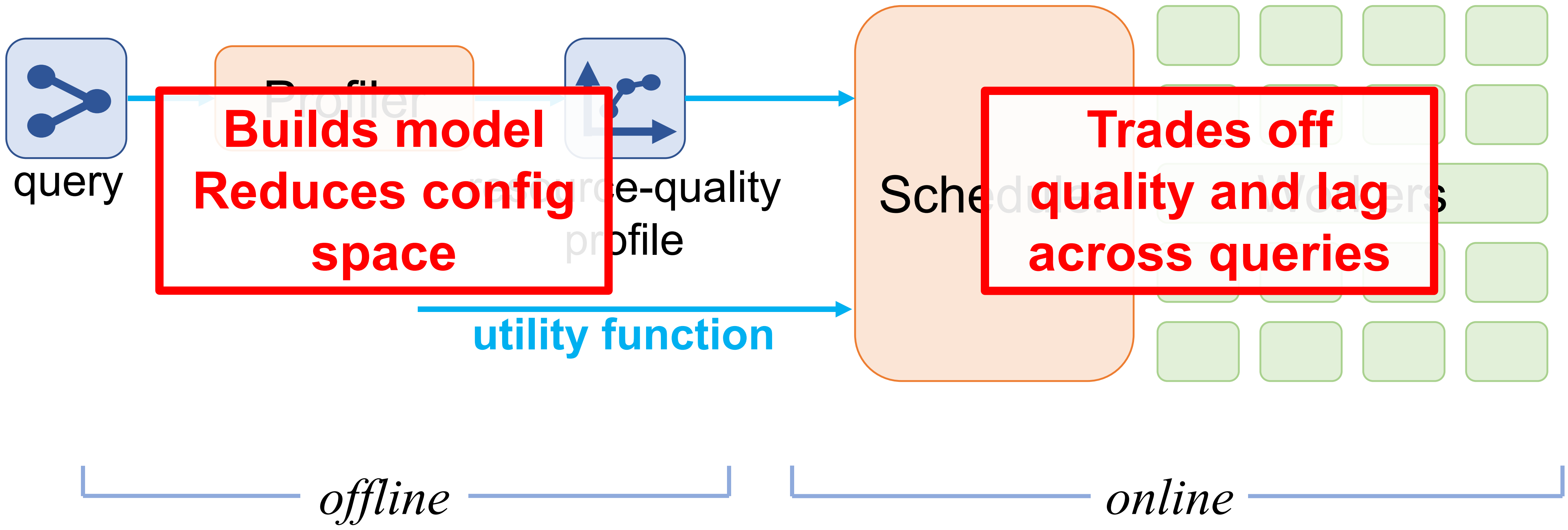


Video analytics framework: Challenges

1. Many knobs → large configuration space
   - No known analytical models to *predict* quality and resource impact
2. Diverse requirements on quality and lag
   - Hard to configure and allocate resources jointly across queries

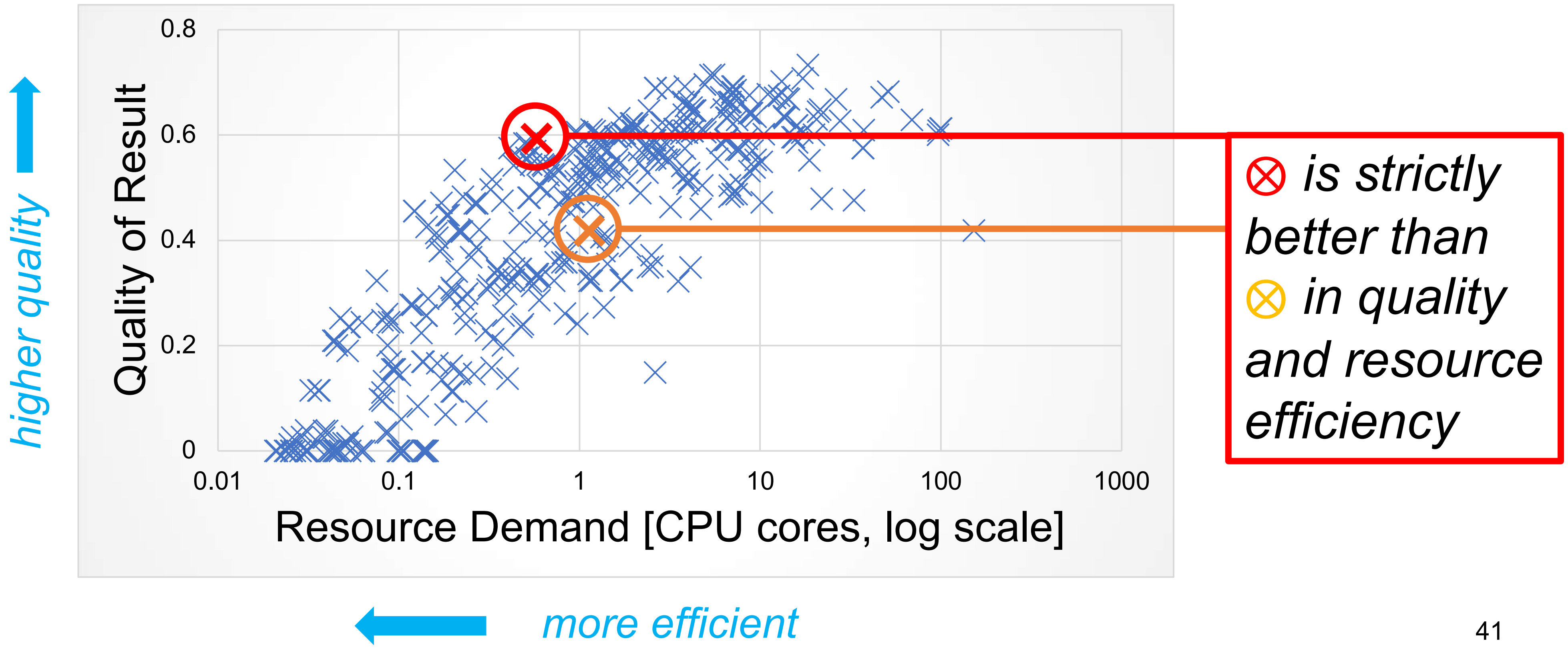


VideoStorm: Solution Overview



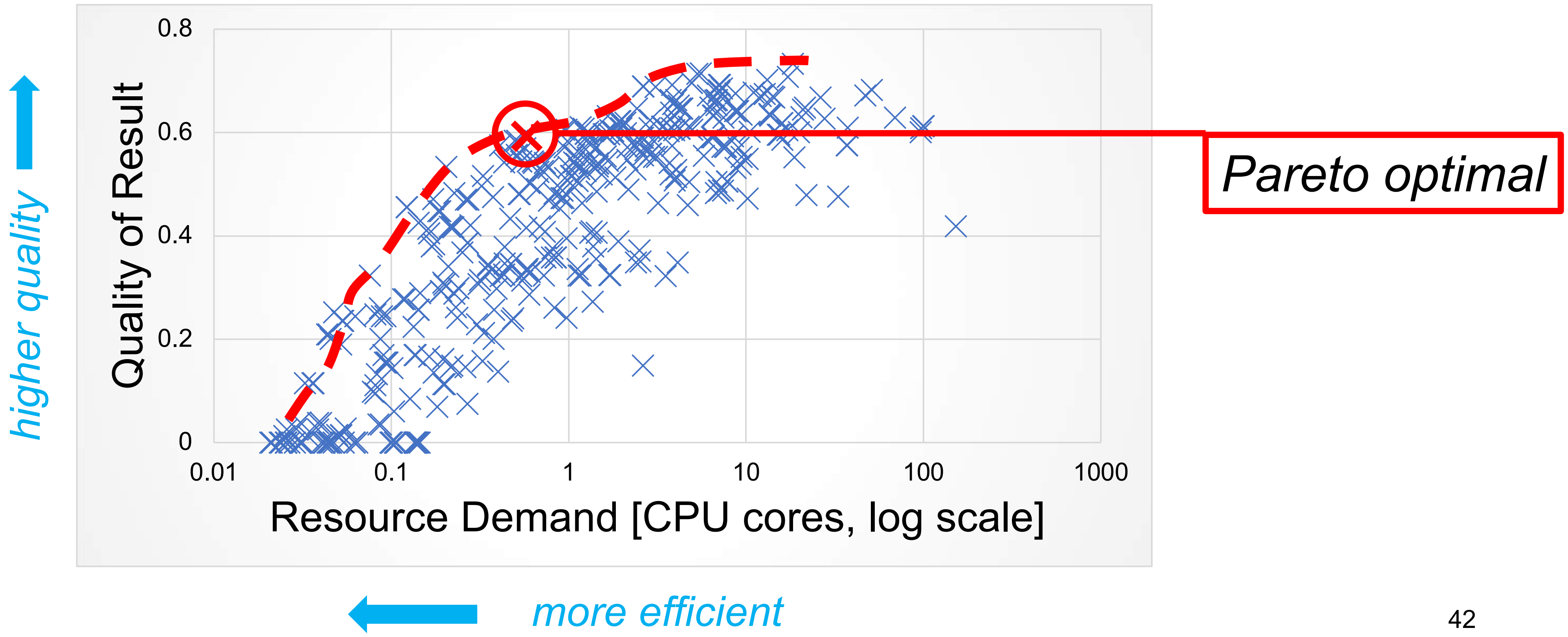
Offline: query profiling

- Profile: configuration ⟹ resource, quality
  - Ground-truth: labeled dataset or results from *golden* configuration
  - Explore configuration space, compute average resource and quality

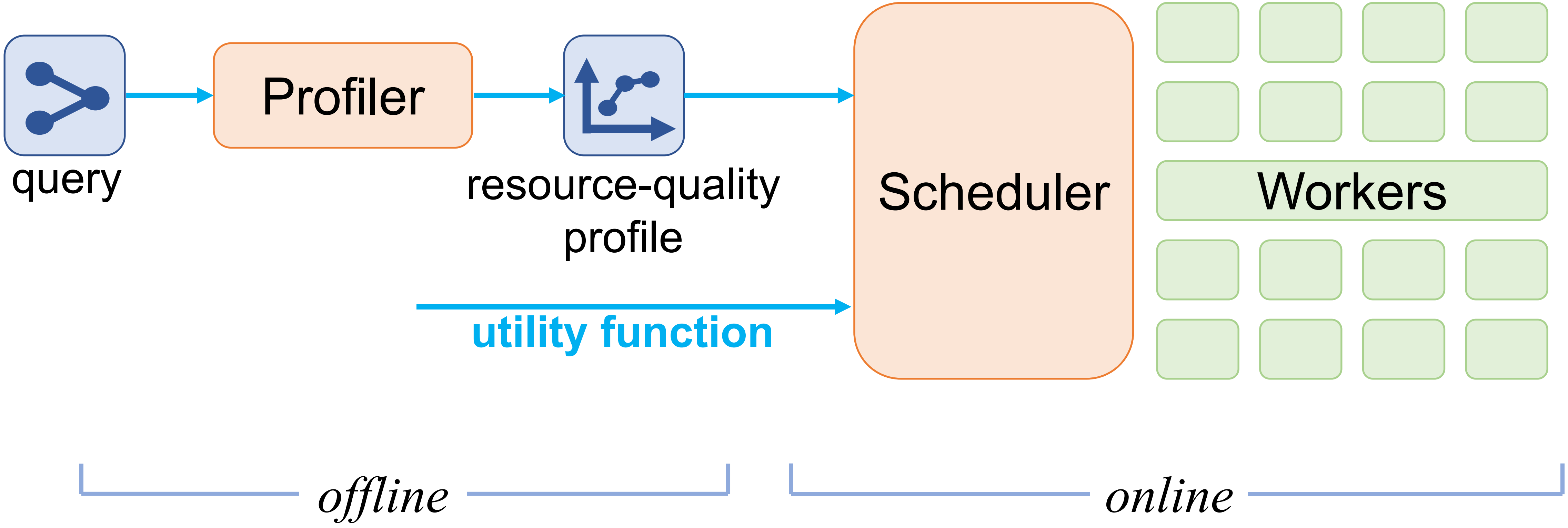


Offline: Pareto boundary of configuration space

- Pareto boundary: optimal configurations in resource efficiency and quality
  - Cannot further increase one without reducing the other
  - Orders of magnitude reduction in config. search space for scheduling

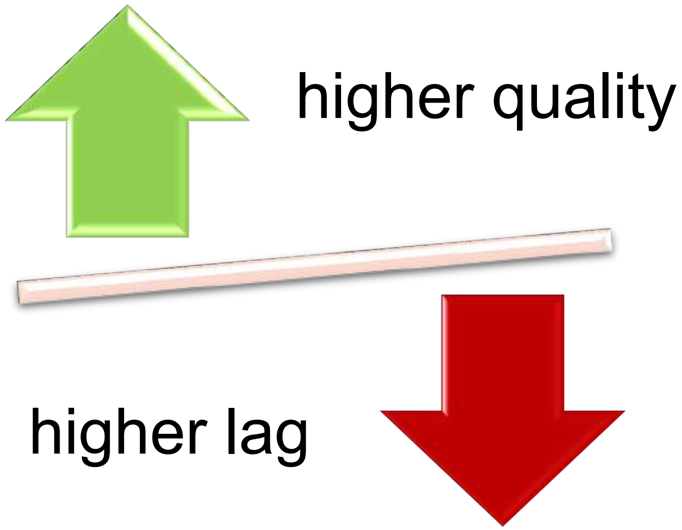


VideoStorm: Solution Overview



Online: utility function and scheduling

- Utility function: encode goals and sensitivities of quality and lag
  - Users set required quality and tolerable lag
  - Reward additional quality, penalize higher lag

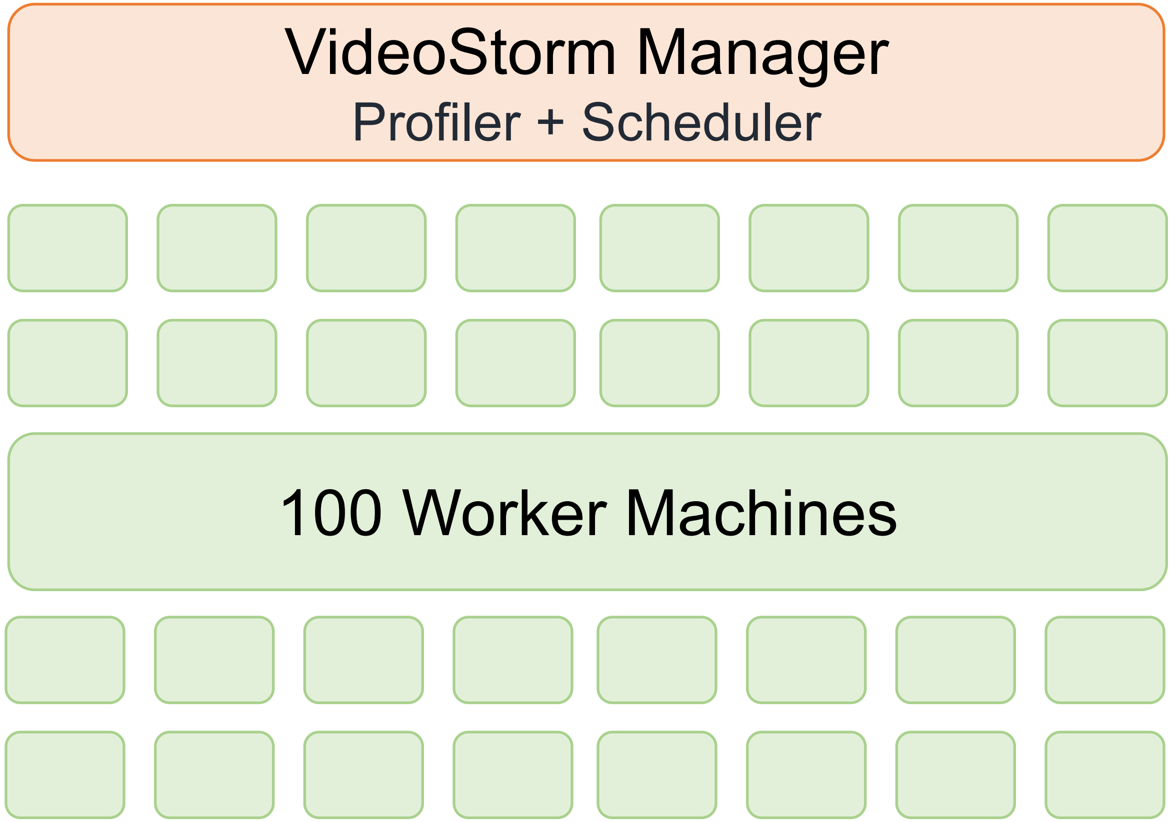


- Schedule for two natural goals
  - Maximize the minimum utility – (max-min) fairness
  - Maximize the total utility – overall performance

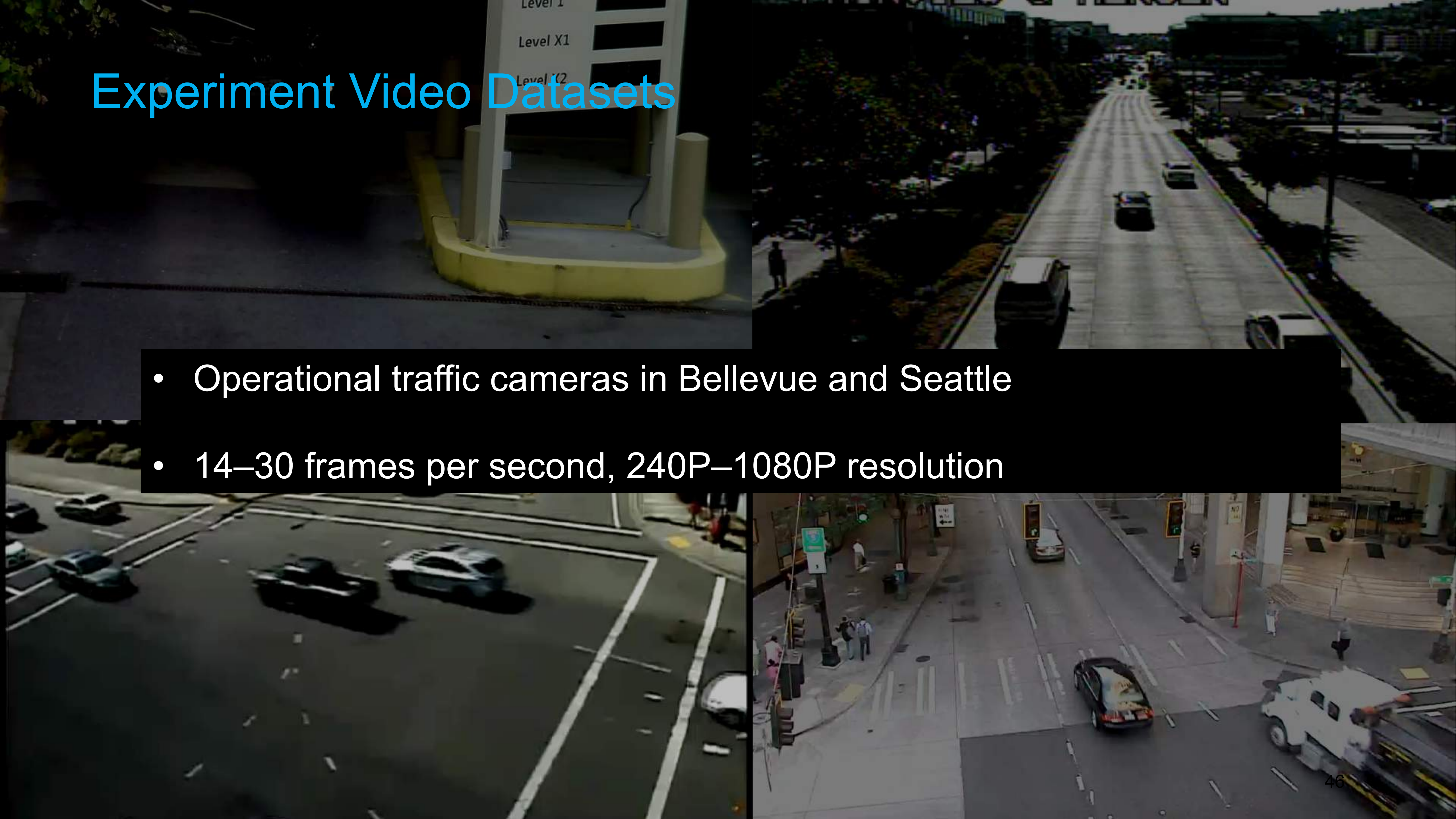
- Allow lag accumulation during resource shortage, then catch up

VideoStorm Evaluation Setup

- Platform:
  - Microsoft Azure cluster
  - Each worker contains 4 cores of the 2.4GHz Intel Xeon processor and 14GB RAM
- Four types of vision queries:
  - license plate reader
  - car counter
  - DNN classifier
  - object tracker



Experiment Video Datasets



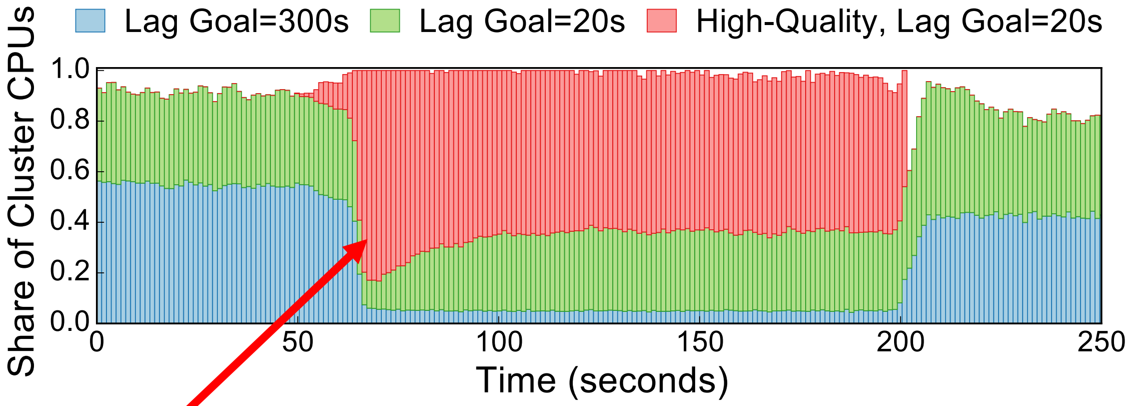
- Operational traffic cameras in Bellevue and Seattle
- 14–30 frames per second, 240P–1080P resolution

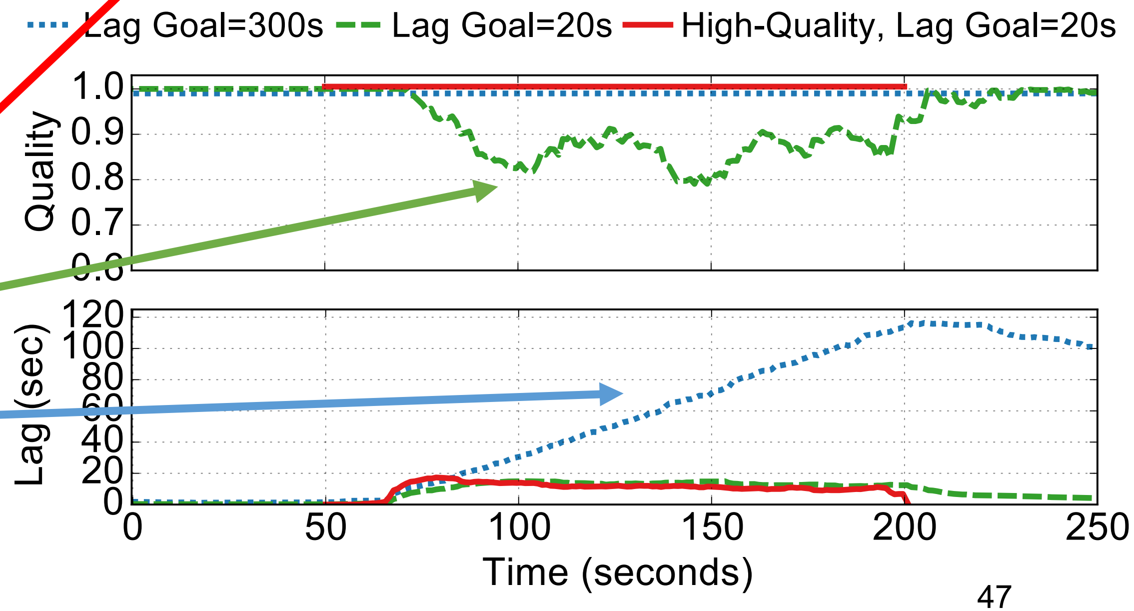
Resource allocation during burst of queries

- Start with 300 queries:
  1 Lag Goal=300s, Low-Quality 60%
  2 Lag Goal=20s, Low-Quality 40%

- Burst of 150 seconds (50 – 200):
  3 200 LPR queries (AMBER Alert)
  Lag Goal=20s, High-Quality

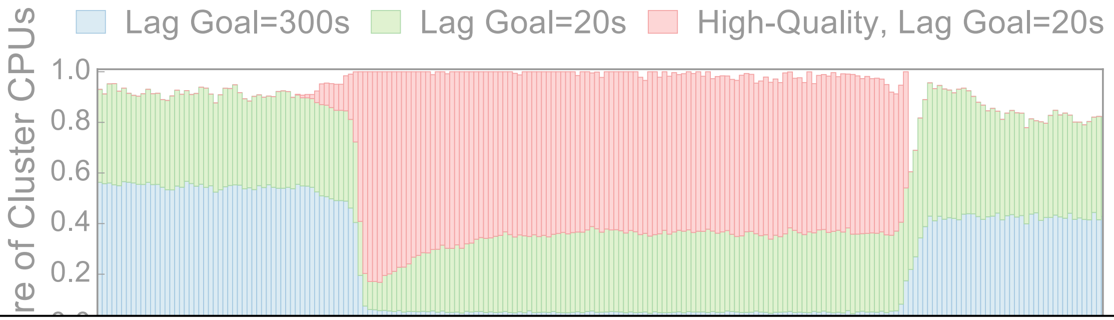
- VideoStorm scheduler:
  3 dominate resource allocation
  run 2 with lower quality
  significantly delay 1
  All meet quality and lag goals





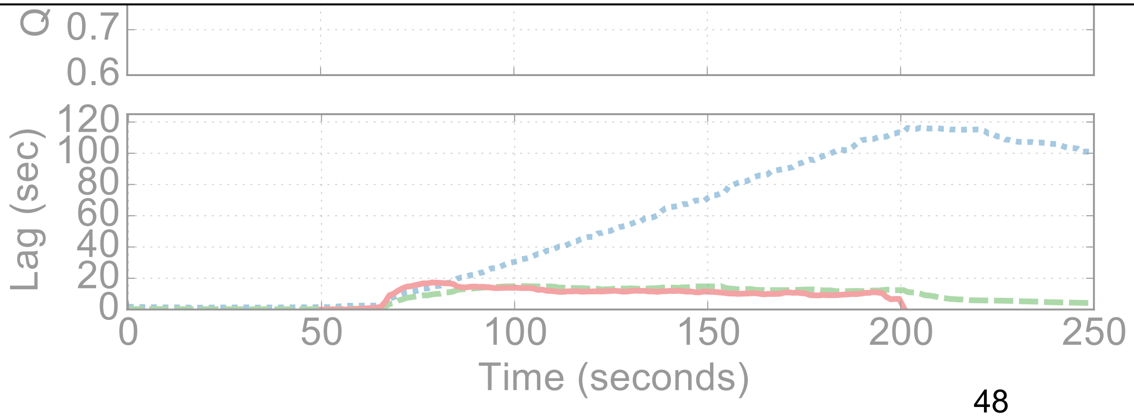
Resource allocation during burst of queries

- Start with 300 queries:
  - 1 Lag Goal=300s, low-quality ~60%
  - 2 Lag Goal=20s, low-quality ~40%



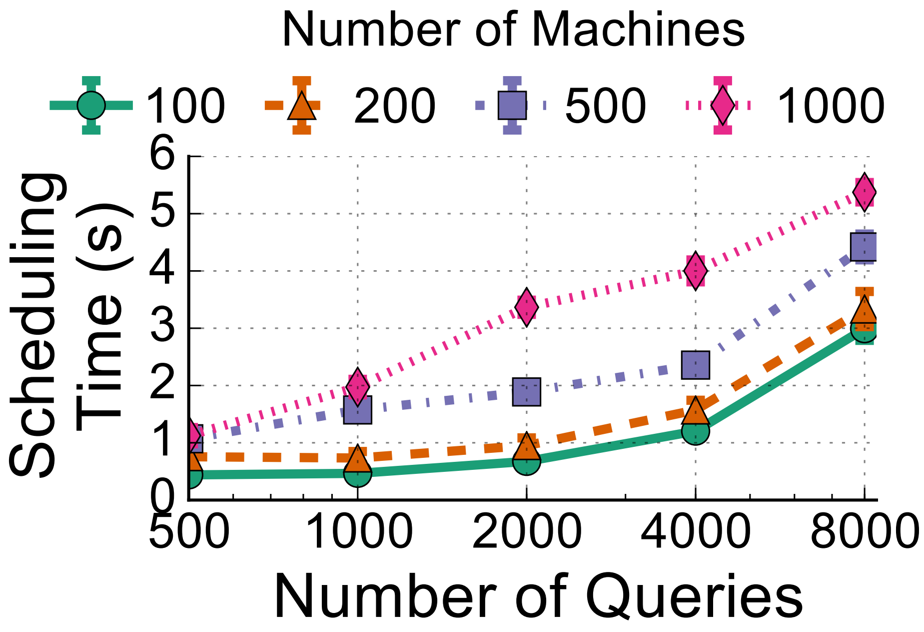
- Compare to a fair scheduler with varying burst duration:
  - Quality improvement: up to 80%
  - Lag reduction: up to 7x

- VideoStorm scheduler:
  - 3 dominate resource allocation
  - significantly delay 1
  - run 2 with lower quality
  - All meet quality and lag goals



VideoStorm Scalability

- Frequently reschedule and reconfigure in reaction to changes of queries
- Even with thousands of queries, VideoStorm makes rescheduling decisions in just a few seconds



Related Work

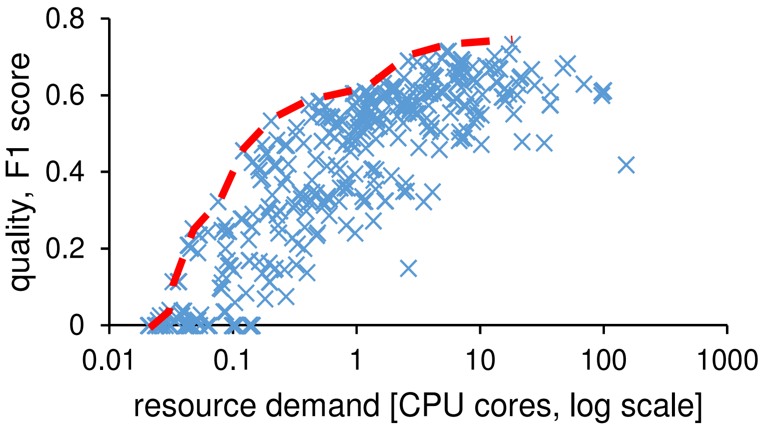
- Video query optimization
  - Optasia [SoCC ’16], NoScope [VLDB ’17], EVA [SysML ’18]
  - Share common operators and reuse results from different queries

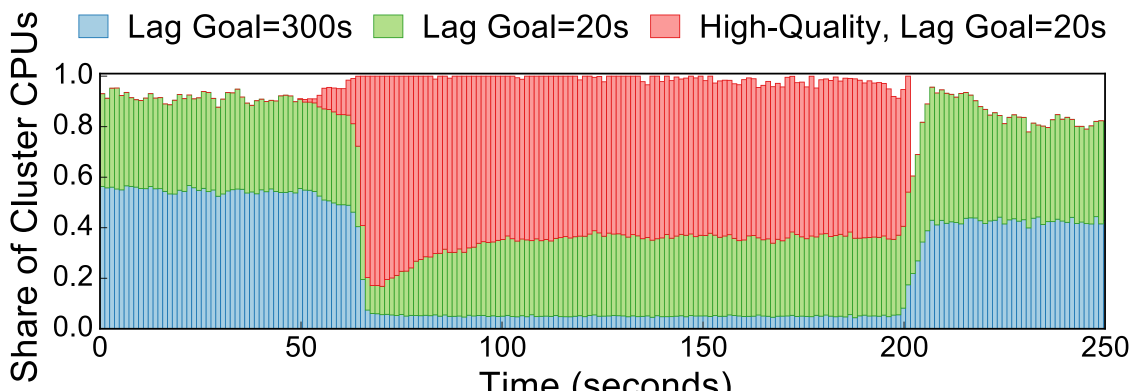
- Video systems on cloud-edge architecture
  - Vigil [MobiCom ’15], Firework [TPDS ’18], Chameleon [SIGCOMM ’18]
  - Placing tasks / operators of a processing pipeline to different locations

Part II Conclusion

- VideoStorm explores quality-resource-lag tradeoff in video queries

- Offline profiler: efficient estimates resource-quality profiles
- Online scheduler: optimizes jointly for quality and lag of queries

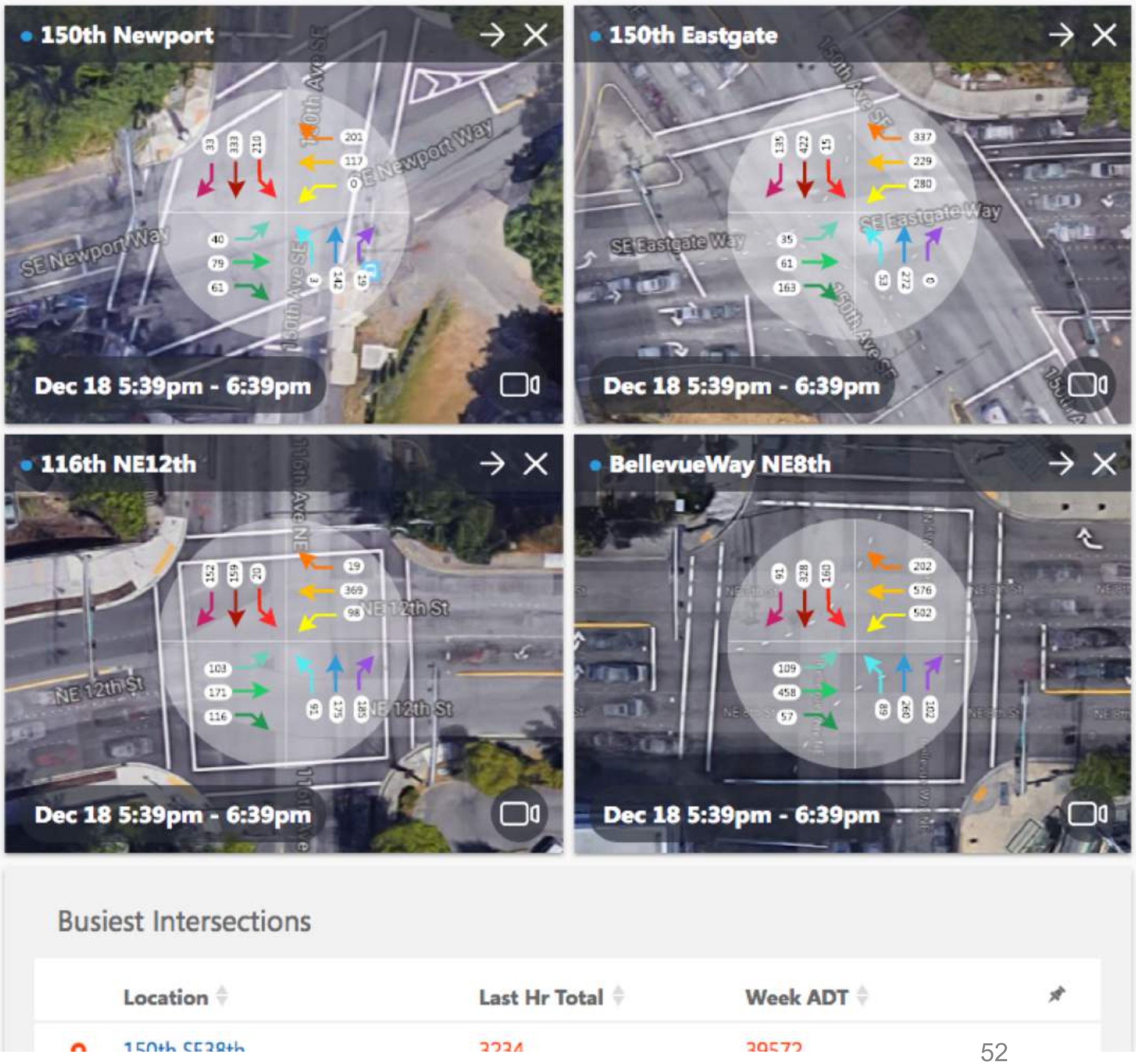




- Significant improvement in achieved quality and lag

Deployment at Bellevue Traffic Department

https://vavz.azurewebsites.net



SLAQ: Quality-Driven Scheduling for Distributed Machine Learning

Haoyu Zhang*, Logan Stafman*, Andrew Or, Michael J. Freedman

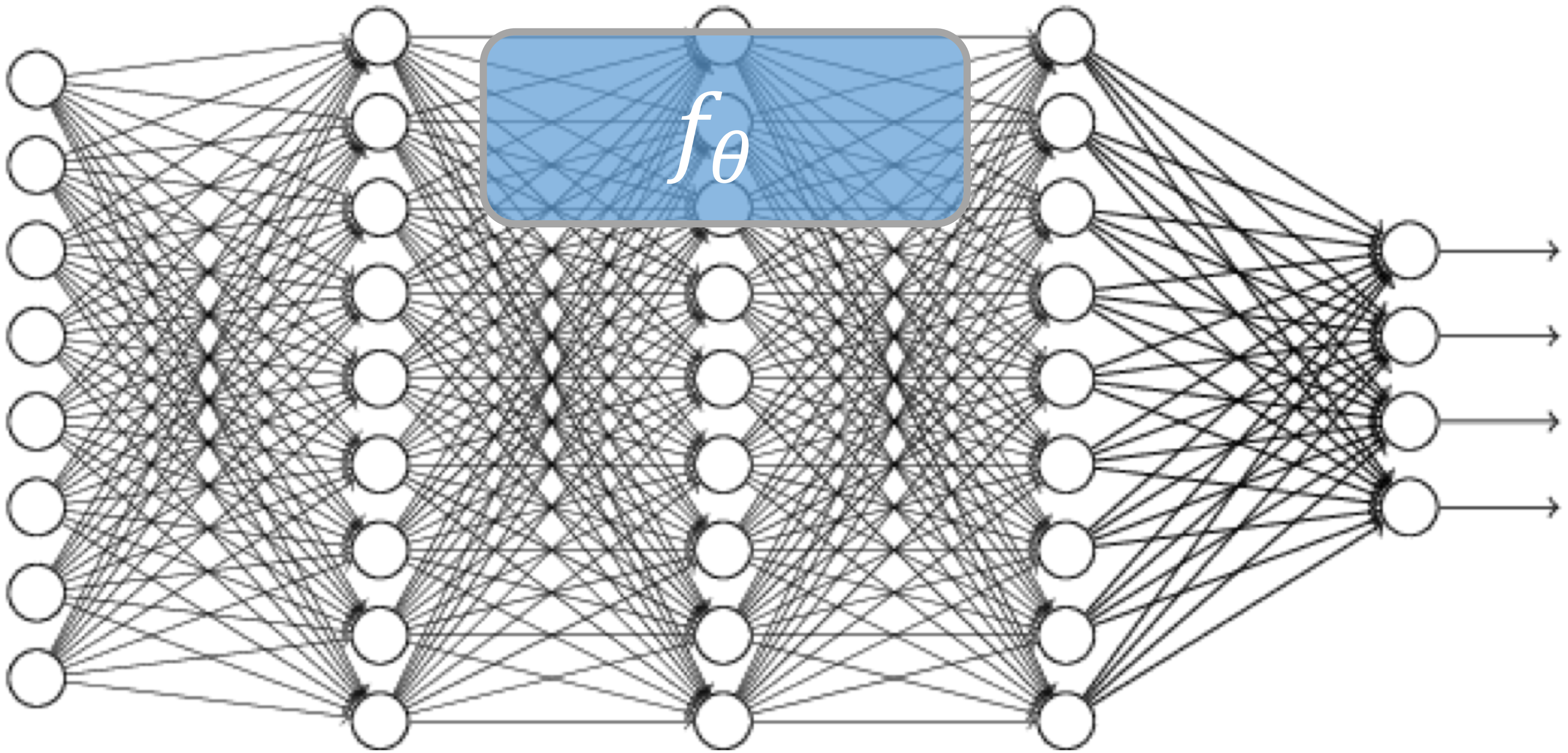
ACM Symposium on Cloud Computing (SoCC '17)

🏆 Best Paper Award

PRINCETON UNIVERSITY

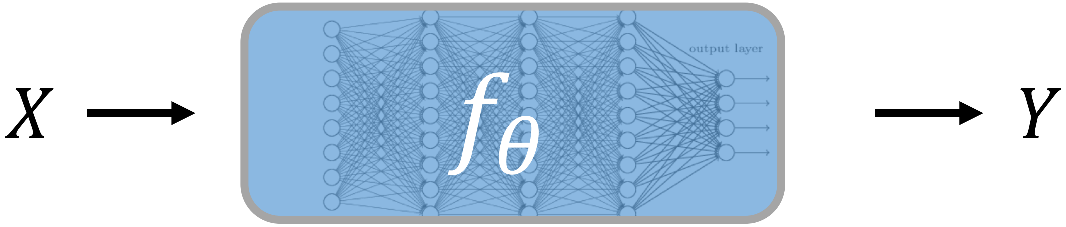
ML algorithms are *approximate*

- ML model: a parametric transformation



ML algorithms are *approximate*

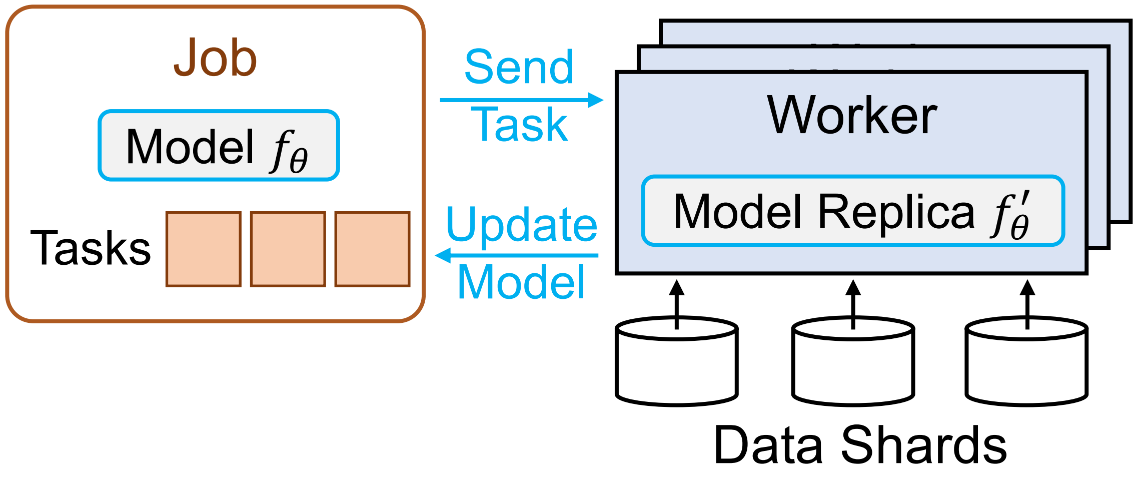
- ML model: a parametric transformation



  - maps input variables X to output variables Y
  - typically contains a set of parameters θ
  - Loss function: discrepancy of model output and ground truth

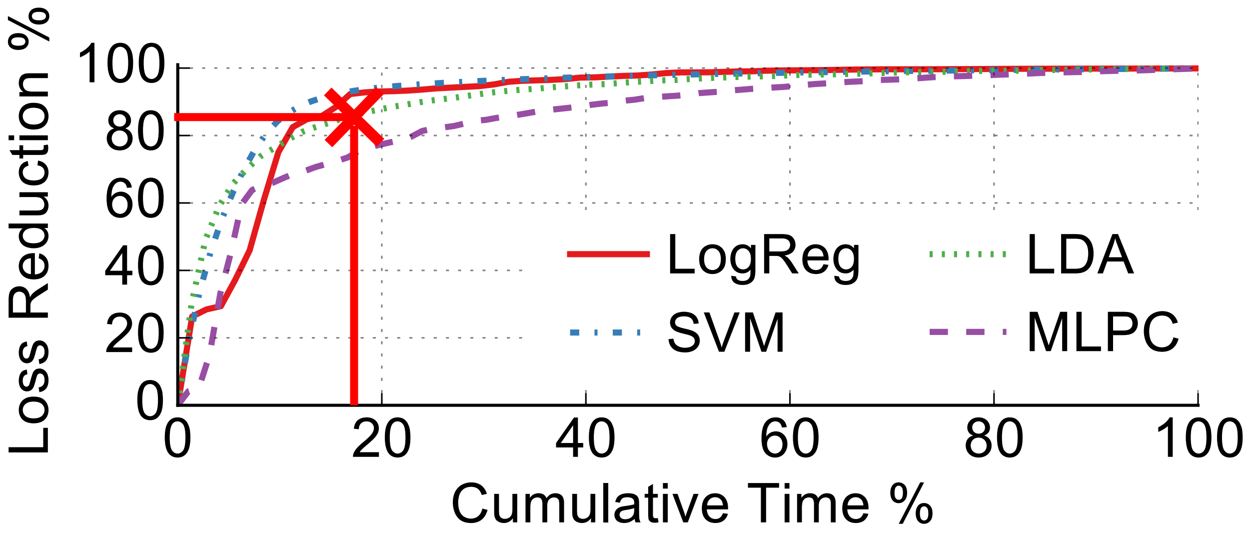
- Quality: how well model maps input to the correct output

Training ML models: an *iterative* process



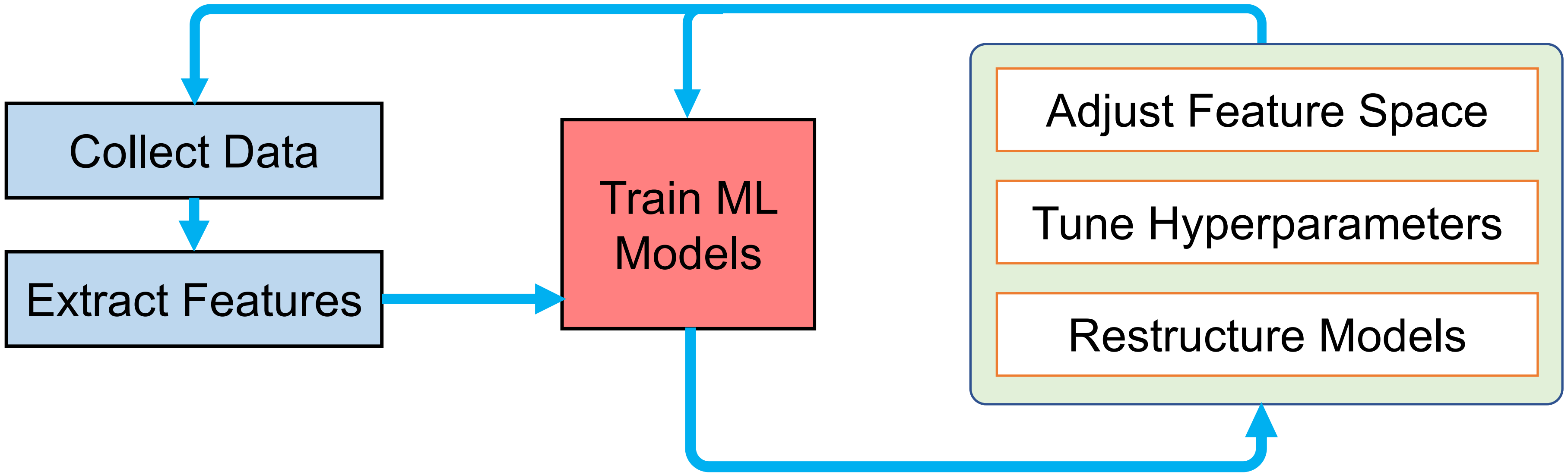
- Training algorithms iteratively minimize a loss function
  - E.g., stochastic gradient descent (SGD), L-BFGS

Training ML models: an *iterative* process



- Quality improvement is subject to diminishing returns
  - More than 80% of work done in 20% of time

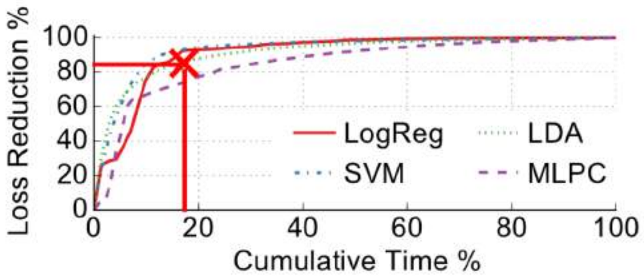
Exploratory ML training: not a one-time effort



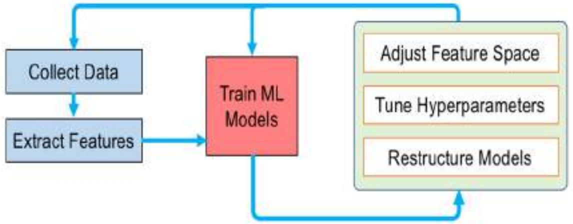
- Train model multiple times for exploratory purposes
- Provide early feedback, direct model search to high quality models

How to schedule multiple training jobs on shared cluster?

Approximate



Diminishing Returns

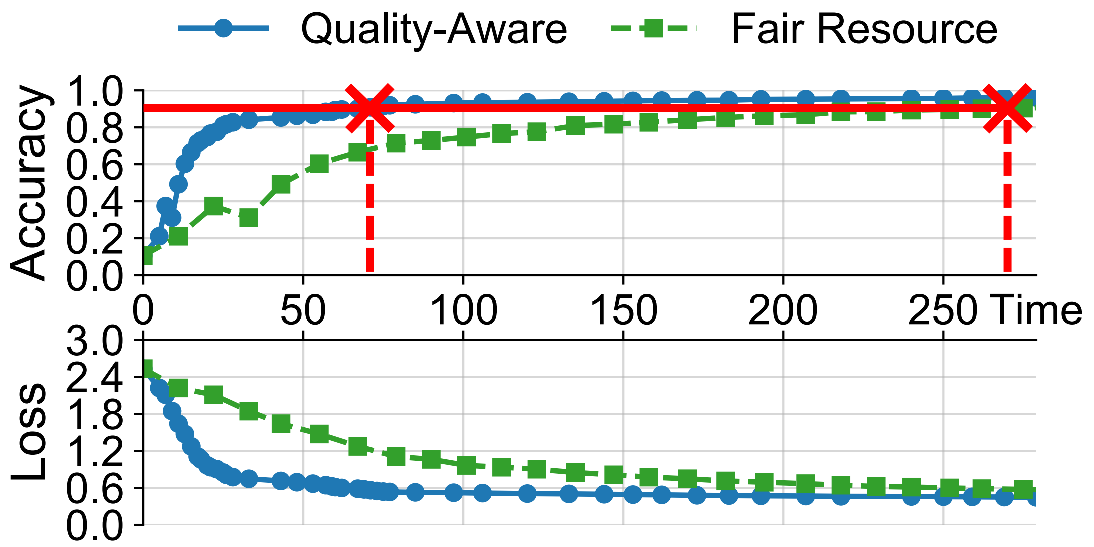


Exploratory

- Problems with resource fairness scheduling
  - Jobs in early stage: could benefit a lot from additional resources
  - Jobs almost converged: make only marginal improvement

SLAQ: quality-aware scheduling

- Intuition: in exploratory ML training, more resources should be allocated to jobs that have the most potential for quality improvement



Solution Overview

Universal quality measurement metric

- Accuracy?
  - *Precision, F1 Score, Area Under Curve, …*

  ✗ Not applicable to non-classification models

- Loss function values?
  - *Square loss, smoothed hinge loss, logistic loss, cross entropy loss, …*

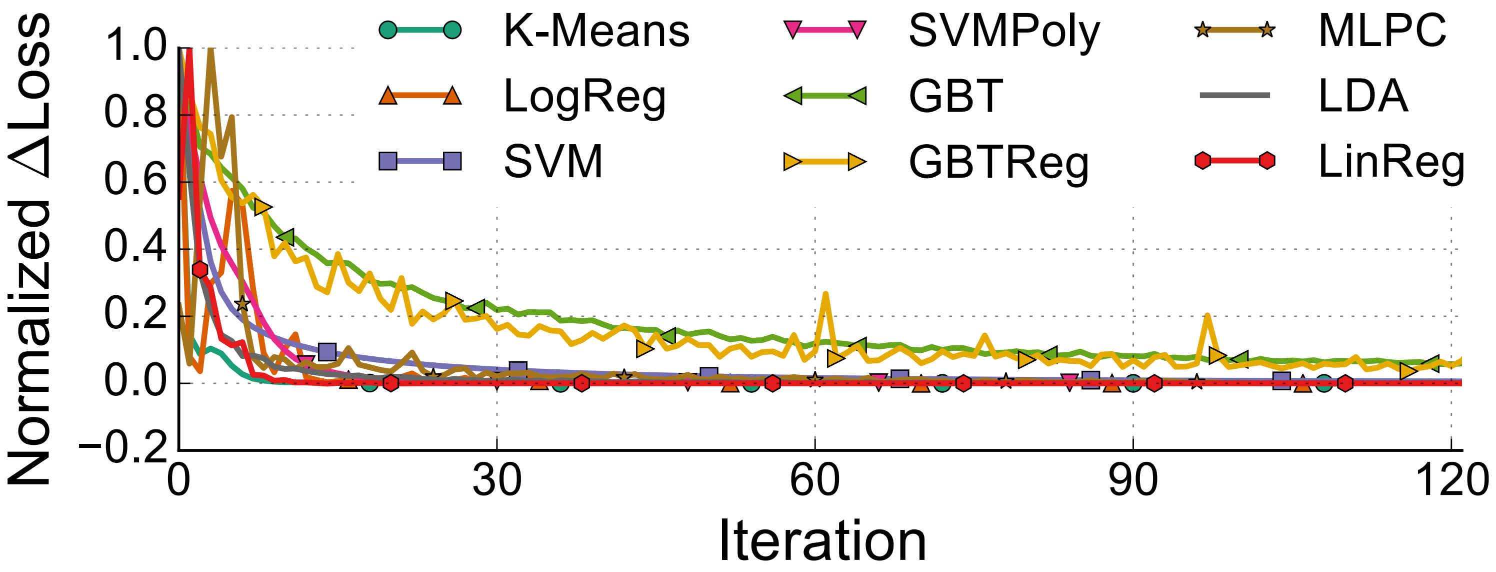
  ✗ Do not have comparable magnitudes or known ranges

- Reduction of loss values (ΔLoss)

  ✓ Always decrease to 0 as the loss function value converges

Normalizing quality metrics

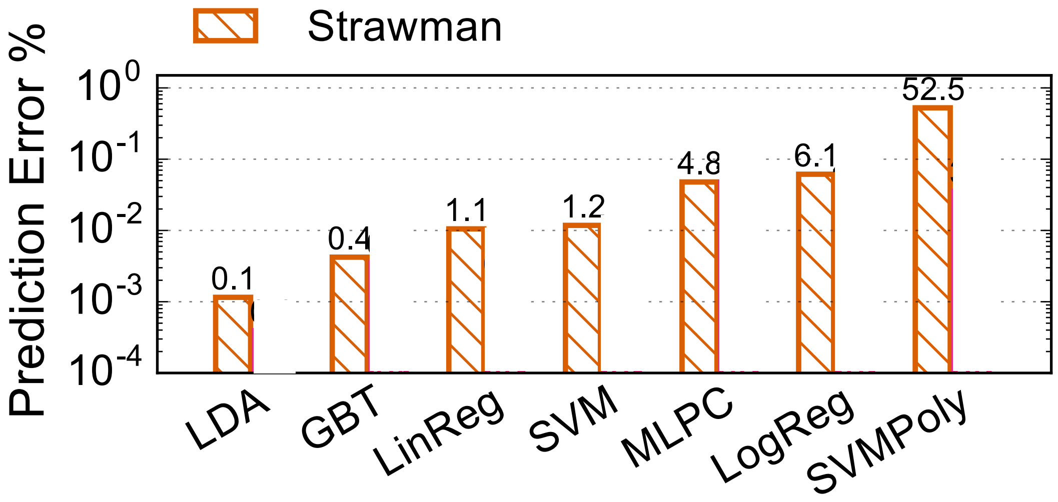
- Quality: normalized change of loss values *w.r.t.* largest change so far



- Currently does not support some non-convex optimization algorithms

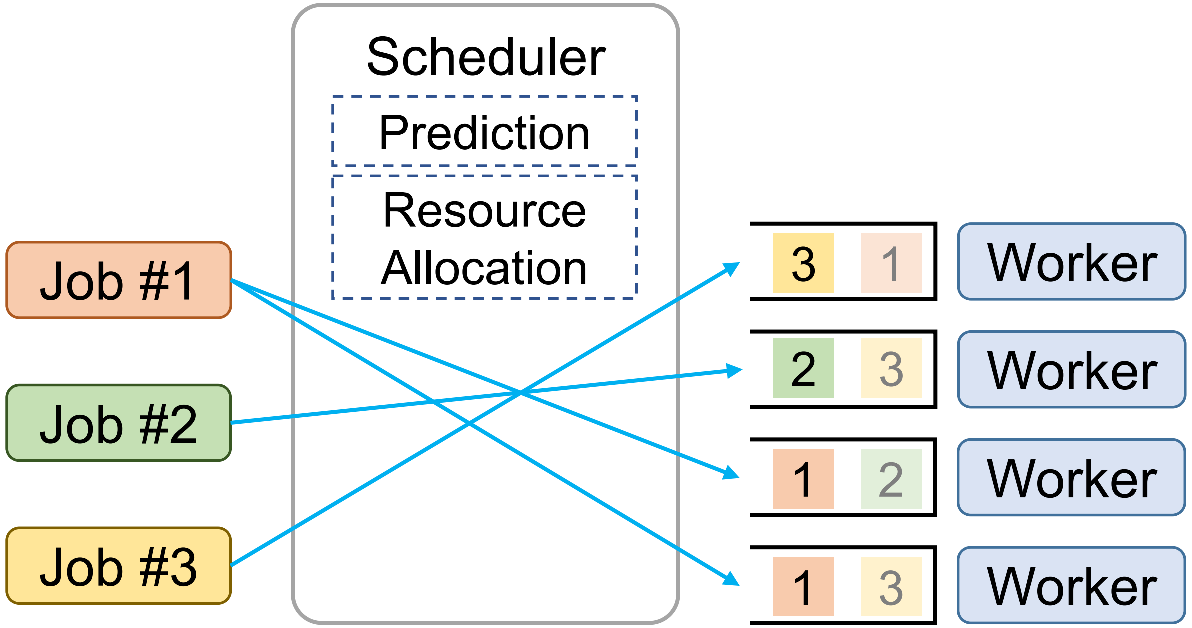
Training iterations: loss prediction

- Previous work: offline profiling / analysis [Ernest NSDI 16] [CherryPick NSDI 17]
  - Overhead for frequent offline analysis is huge
- Strawman: use last ΔLoss as prediction for future ΔLoss
- SLAQ: online prediction using weighted curve fitting



Scheduling approximate ML training jobs

- Predict how much quality can be improved when assign X workers to jobs
- Reallocate workers to maximize quality improvement



Experiment setup

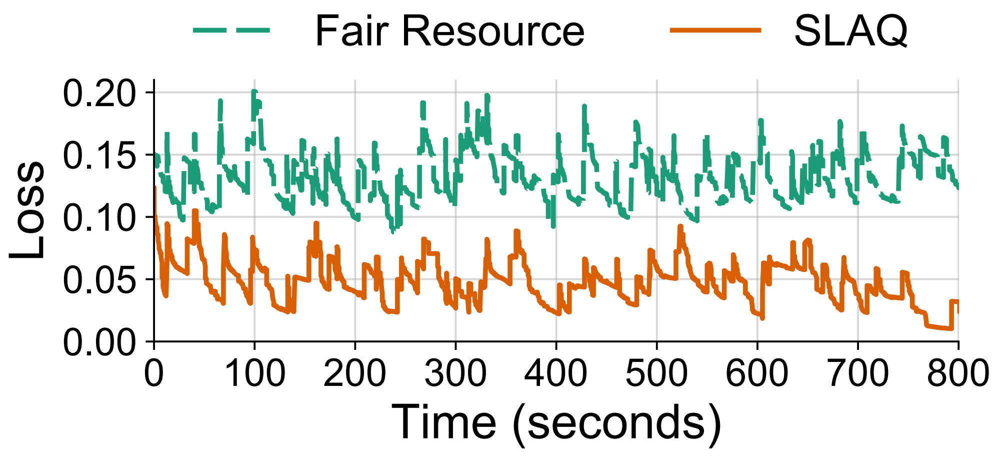
- Representative mix of training jobs with Spark MLlib
- Compare against a work-conserving fair scheduler

Algorithm	Acronym	Type	Optimization Algorithm	Dataset
K-Means	K-Means	Clustering	Lloyd Algorithm	Synthetic
Logistic Regression	LogReg	Classification	Gradient Descent	Epsilon [33]
Support Vector Machine	SVM	Classification	Gradient Descent	Epsilon
SVM (polynomial kernel)	SVMPoly	Classification	Gradient Descent	MNIST [34]
Gradient Boosted Tree	GBT	Classification	Gradient Boosting	Epsilon
GBT Regression	GBTReg	Regression	Gradient Boosting	YearPredictionMSD [35]
Multi-Layer Perceptron Classifier	MLPC	Classification	L-BFGS	Epsilon
Latent Dirichlet Allocation	LDA	Clustering	EM / Online Algorithm	Associated Press Corpus [36]
Linear Regression	LinReg	Regression	L-BFGS	YearPredictionMSD

Evaluation: cluster-wide quality and time

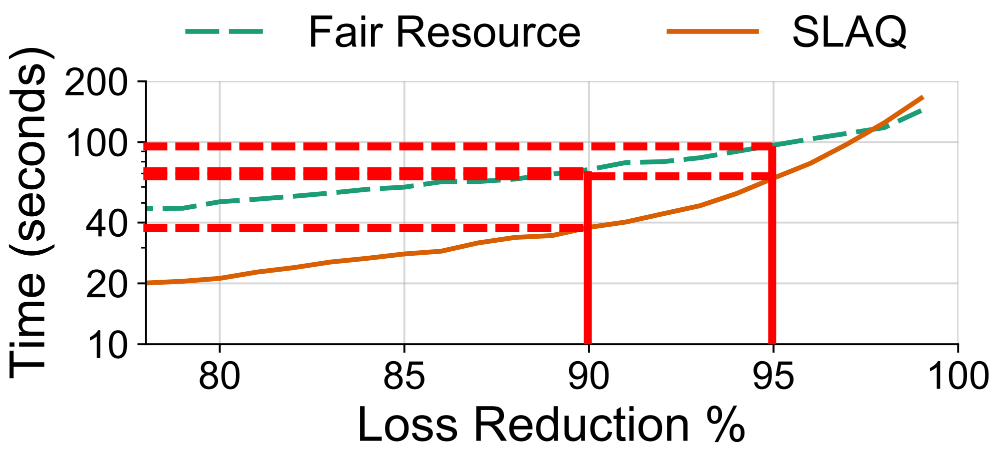
Quality

- SLAQ's average loss is 73% lower than that of the fair scheduler



Time

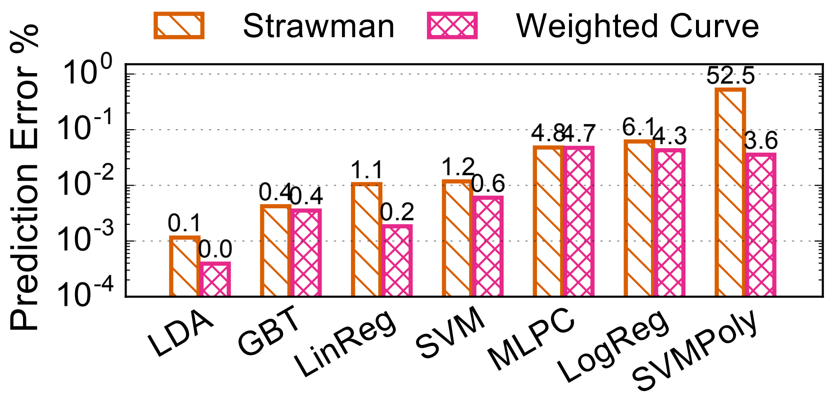
- SLAQ reduces time to reach 90% (95%) loss reduction by 45% (30%)

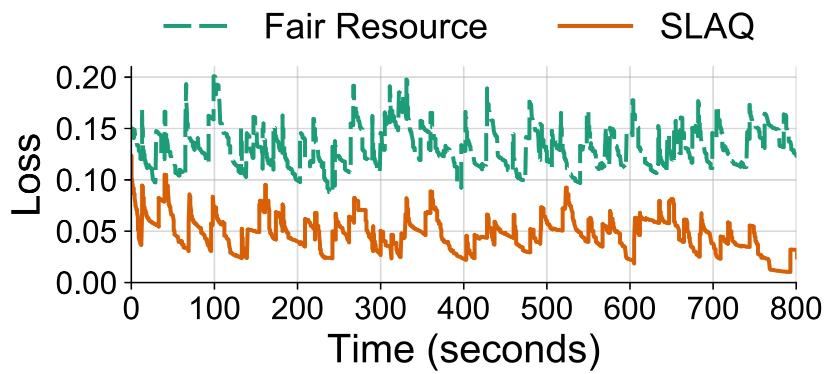


Part III Conclusion

- SLAQ leverages the approximate and iterative ML training process

- Highly tailored prediction for iterative job quality
- Allocate resources to maximize quality improvement



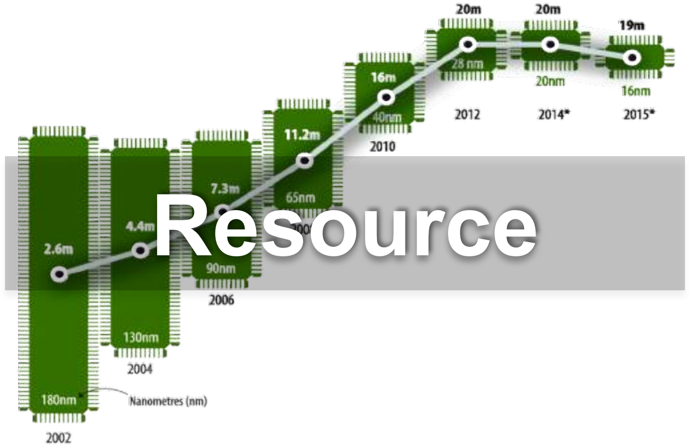


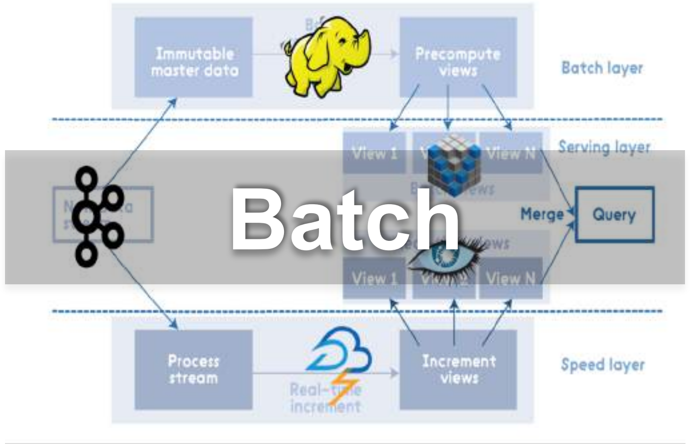
- SLAQ achieves better overall quality and end-to-end training time

Conclusion

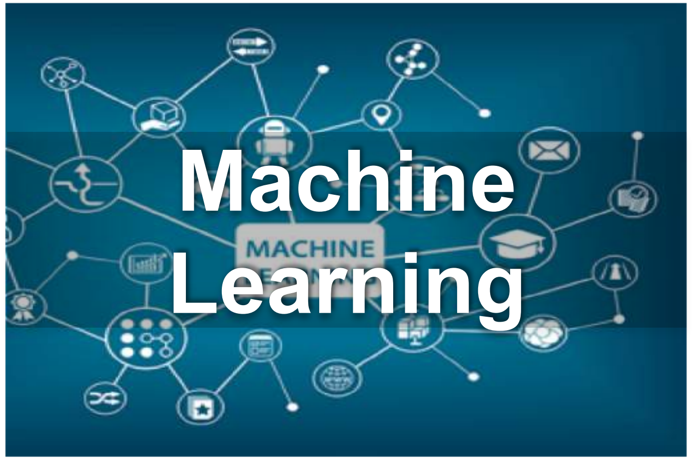

Data


Complexity


20m
20m
19m
16m
2012
2014*
2015*
2010
11.2m
7.3m
65nm
2.6m
4.4m
Resource
90nm
2006
130nm
2004
180nm
Nanometres (nm)
2002


Immutable master data
Precompute views
Batch layer
Serving layer
Batch
Merge
Query
Process stream
Increment views
Speed layer


Traffic Mix
Car
Bus
Trucks
Object detected
status:CONFIRMED
Video
325
per hr


Machine
MACHINE
LEARNING
Learning

Research Summary

- Resource management for advanced data analytics
  - *Live Video Analytics at Scale with Approximation and Delay-Tolerance* [NSDI '17]
  - *SLAQ: Quality-Driven Scheduling in Distributed Machine Learning* [SoCC '17 🏆][SysML '18]
  - *Riffle: Optimized Shuffle Service for Large-Scale Data Analytics* [EuroSys '18]
- Network-assisted system acceleration
  - *NetCache: Balancing Key-Value Stores with Fast In-Network Caching* [SOSP '17]
  - *NetChain: Scale-Free Sub-RTT Coordination* [NSDI '18 🏆]
- SDN fault tolerance
  - *Ravana: Controller Fault-Tolerance in Software-Defined Networks* [SOSR '15]


Thanks!
Haoyu Zhang
http://www.haoyuzhang.org