Machine Learning for Resource Management
in the Datacenter and the Cloud

Neeraja J. Yadwadkar
Post Doctoral Researcher
Stanford University
November 19th, 2019

What this talk is about…

- ✓ What matters when setting up a learning problem in real-world distributed systems
- ✓ Challenges to anticipate
- ✓ Insights and lessons learned while addressing them.

A “Simple” Recipe!

- ❑ **Step I – Learning Problem Formulation**
  - ❑ Domain Knowledge –
    - ❑ Understand key limitations of existing mechanisms
    - ❑ Understand metrics of success
  - ❑ Gather a dataset
  - ❑ Learn a model

- ❑ **Step II – Close the loop**
  - ❑ Implement and Deploy learned models in existing system architecture

- ❑ **Step III – Evaluate using metrics of success**

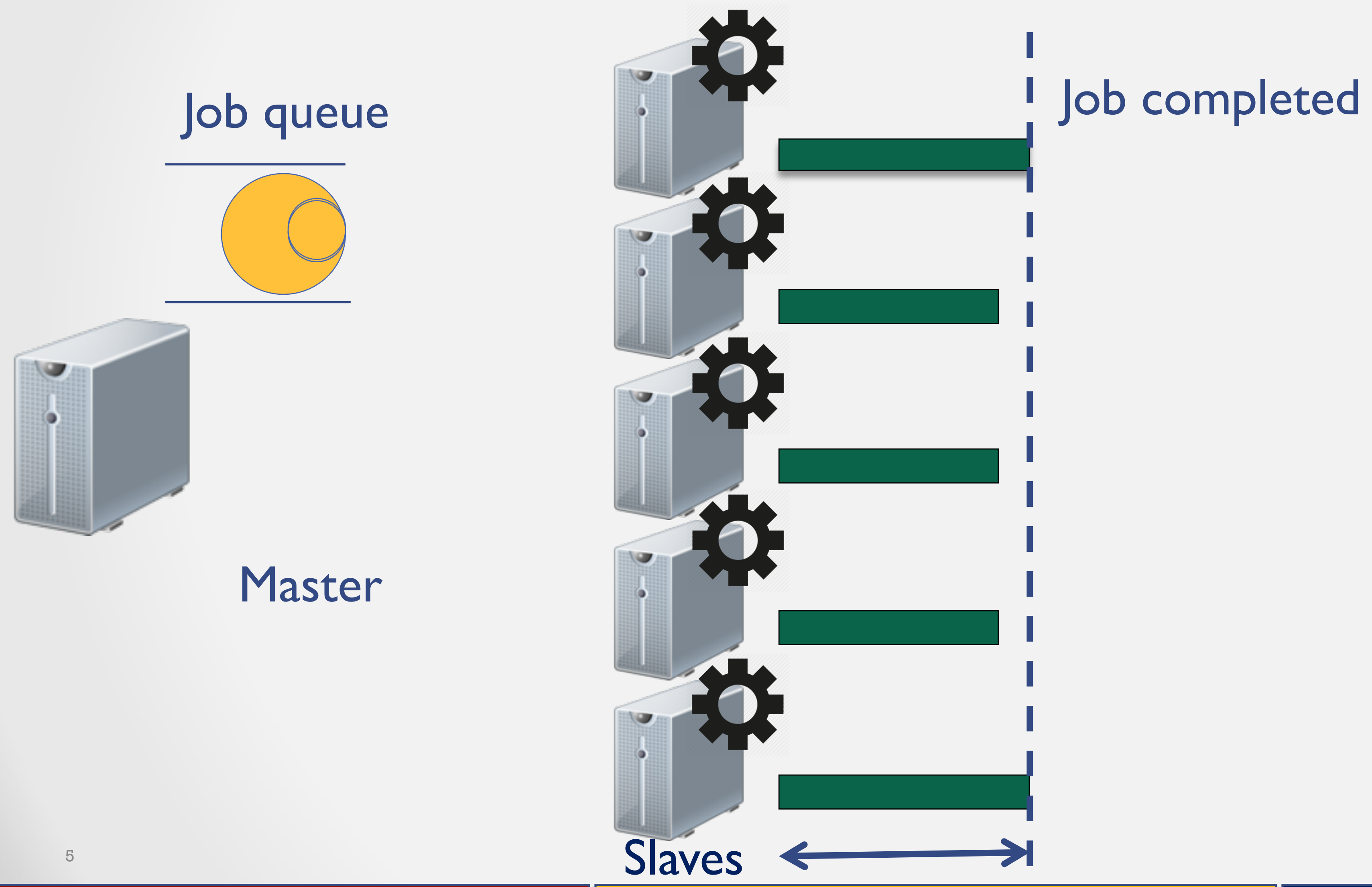
In reality, we face challenges that add many twists and turns in to this recipe.

Two Problem Instances

- Job scheduling in datacenter environments
  - Problem - Long tail of job completions

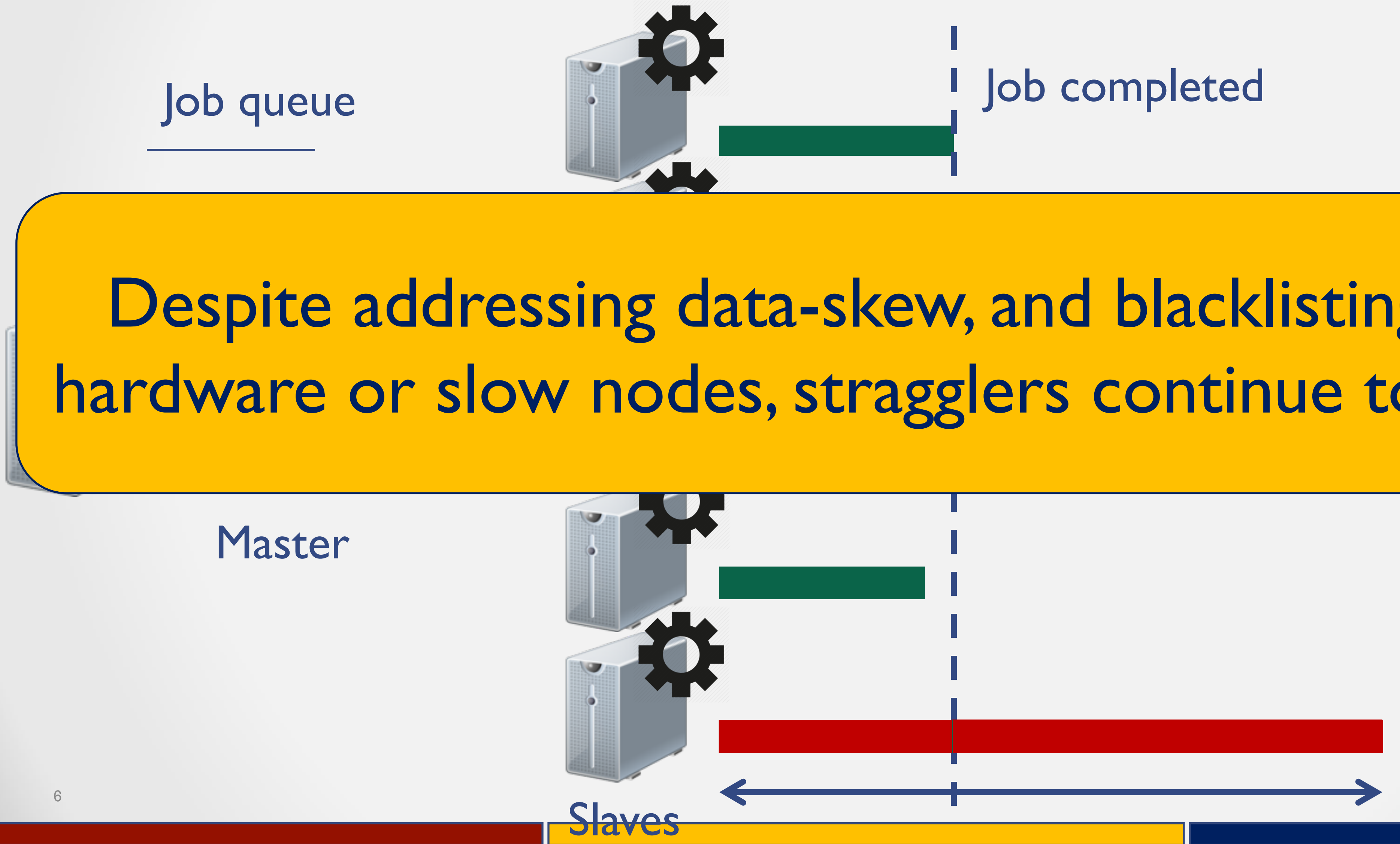
- Resource allocation in public cloud environments
  - Problem - Right-sizing resources for workloads

Parallel Data Intensive Computational Frameworks



Wrangler, SoCC'14

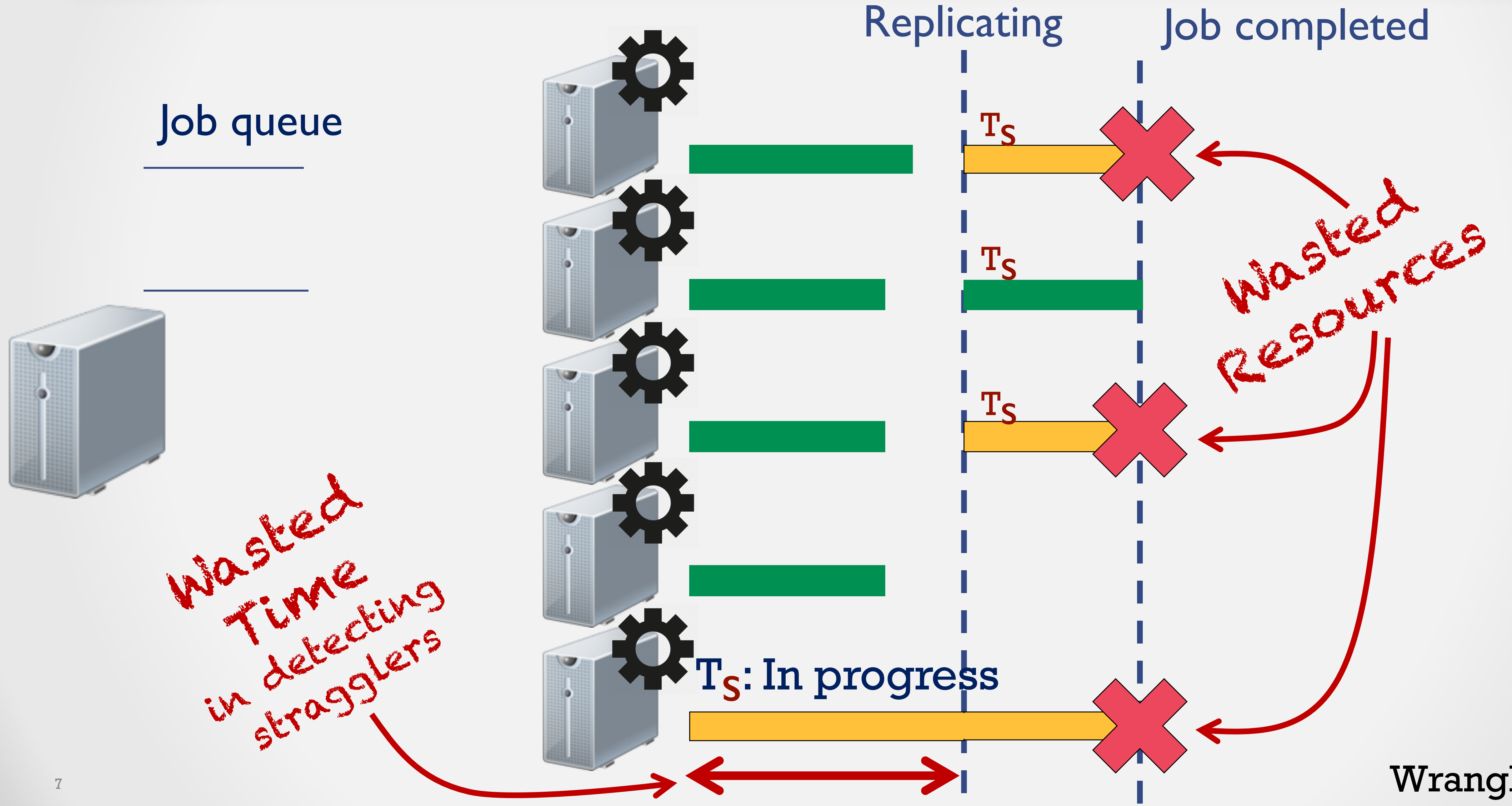
Stragglers



Despite addressing data-skew, and blacklisting faulty hardware or slow nodes, stragglers continue to exist…

Wrangler, SoCC'14

Existing Mechanism - Speculative Execution



Wrangler, SoCC'14

Design Goals

I. Identify stragglers as early as possible

Avoid Wasting Time
in detecting stragglers

II. Schedule tasks for improved job finishing times

1. To avoiding creation of stragglers
2. To avoid replication

Avoid Wasting Resources

Wrangler, SoCC'14

Design Goals: ML formulation

I. Identify stragglers as early as possible

Classify machine state to be healthy or straggler prone

State of a machine

Utilization statistics

Classifier

Straggler

Non-Straggler

Straggler Predictor

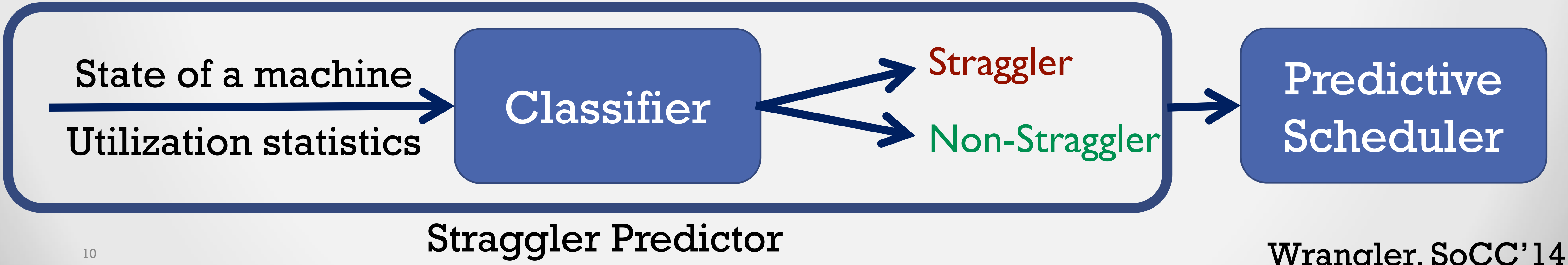
Wrangler, SoCC'14

Design Goals: ML formulation

I. Identify stragglers as early as possible

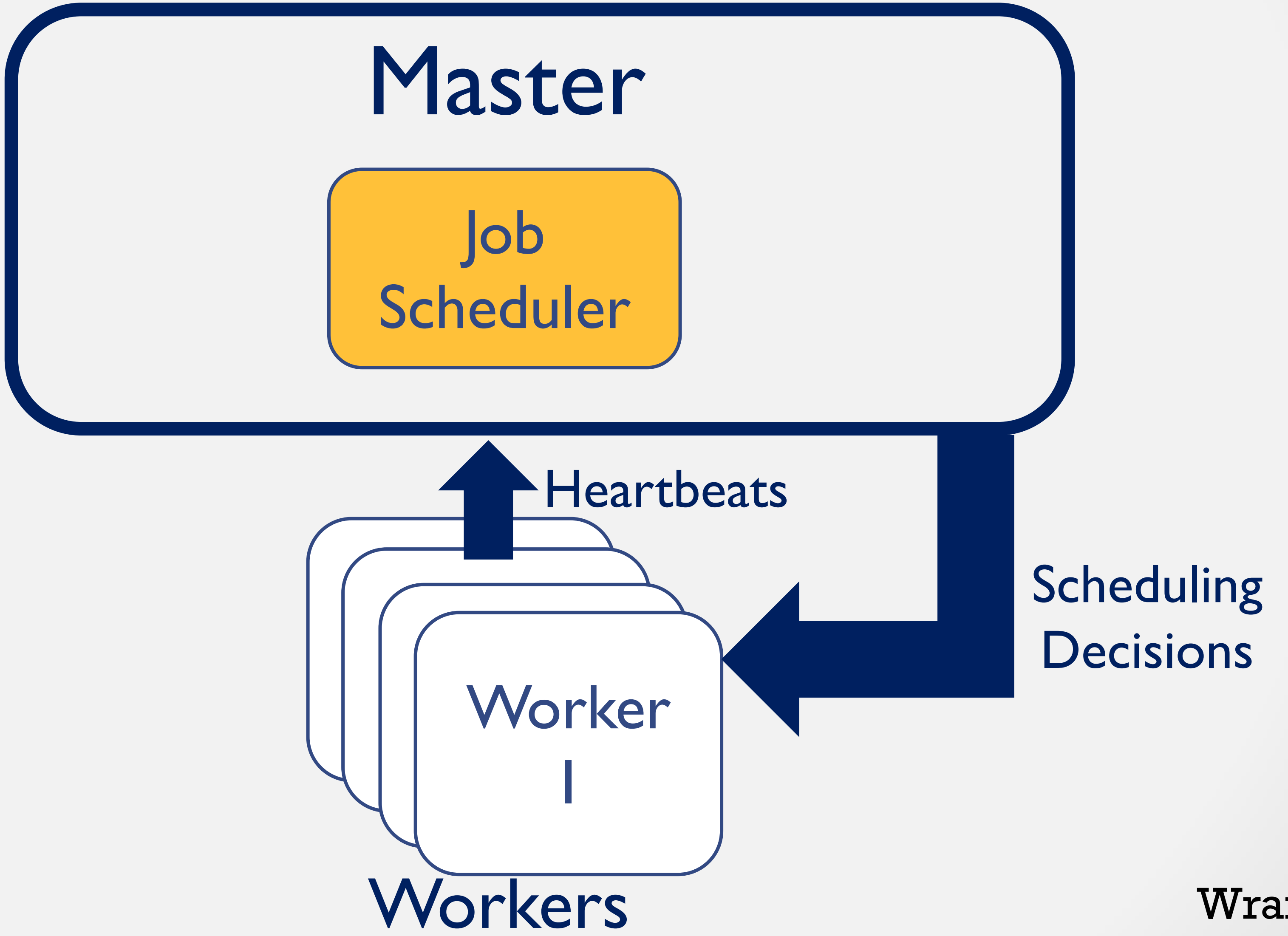
II. Schedule tasks for improved job finishing times

Use predictions as hints to the scheduler



Wrangler, SoCC'14

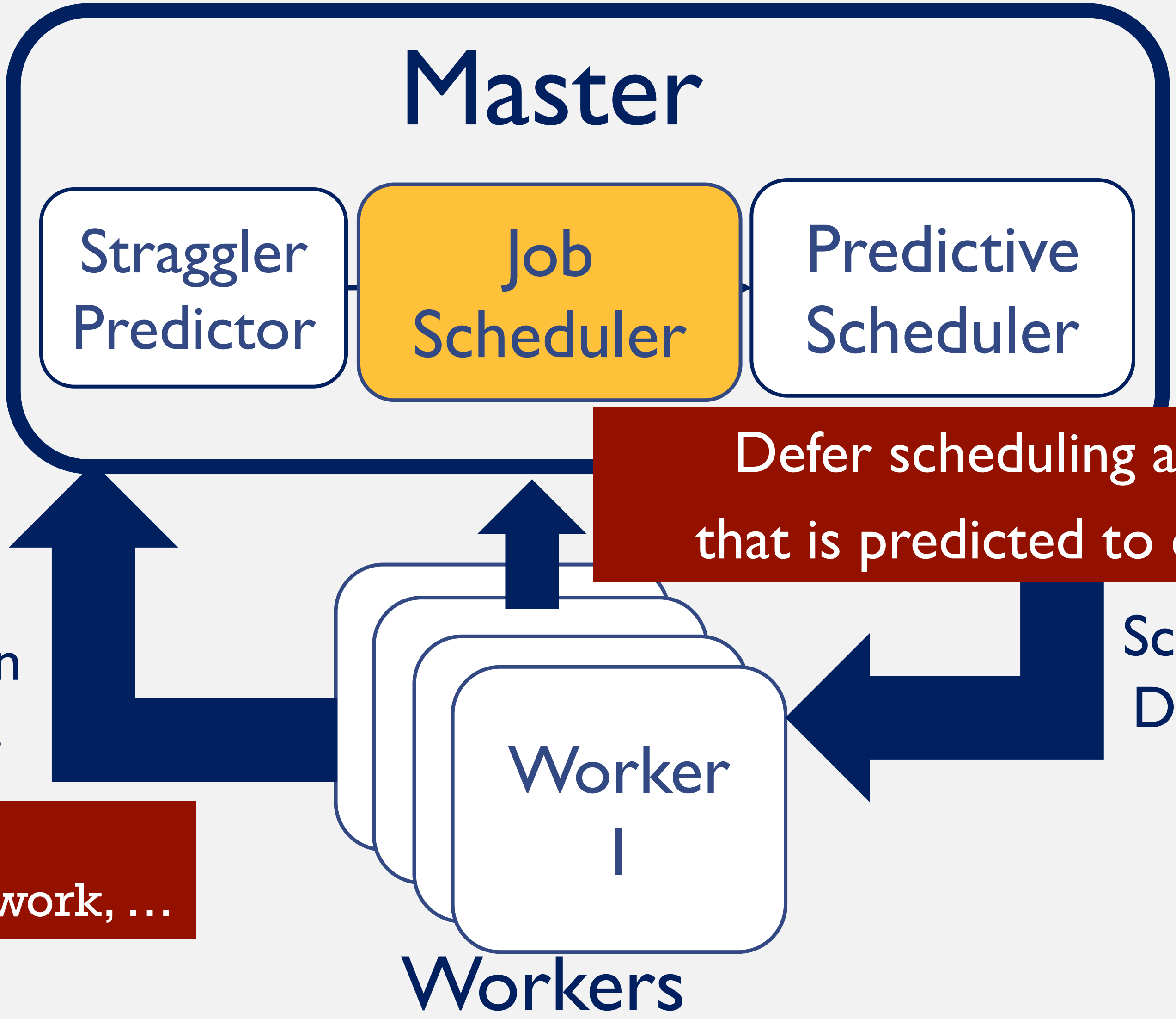
Job Scheduling in Data Intensive Computational Frameworks



Wrangler, SoCC'14

Our proposal: Wrangler

SVM
~70-80% accuracy



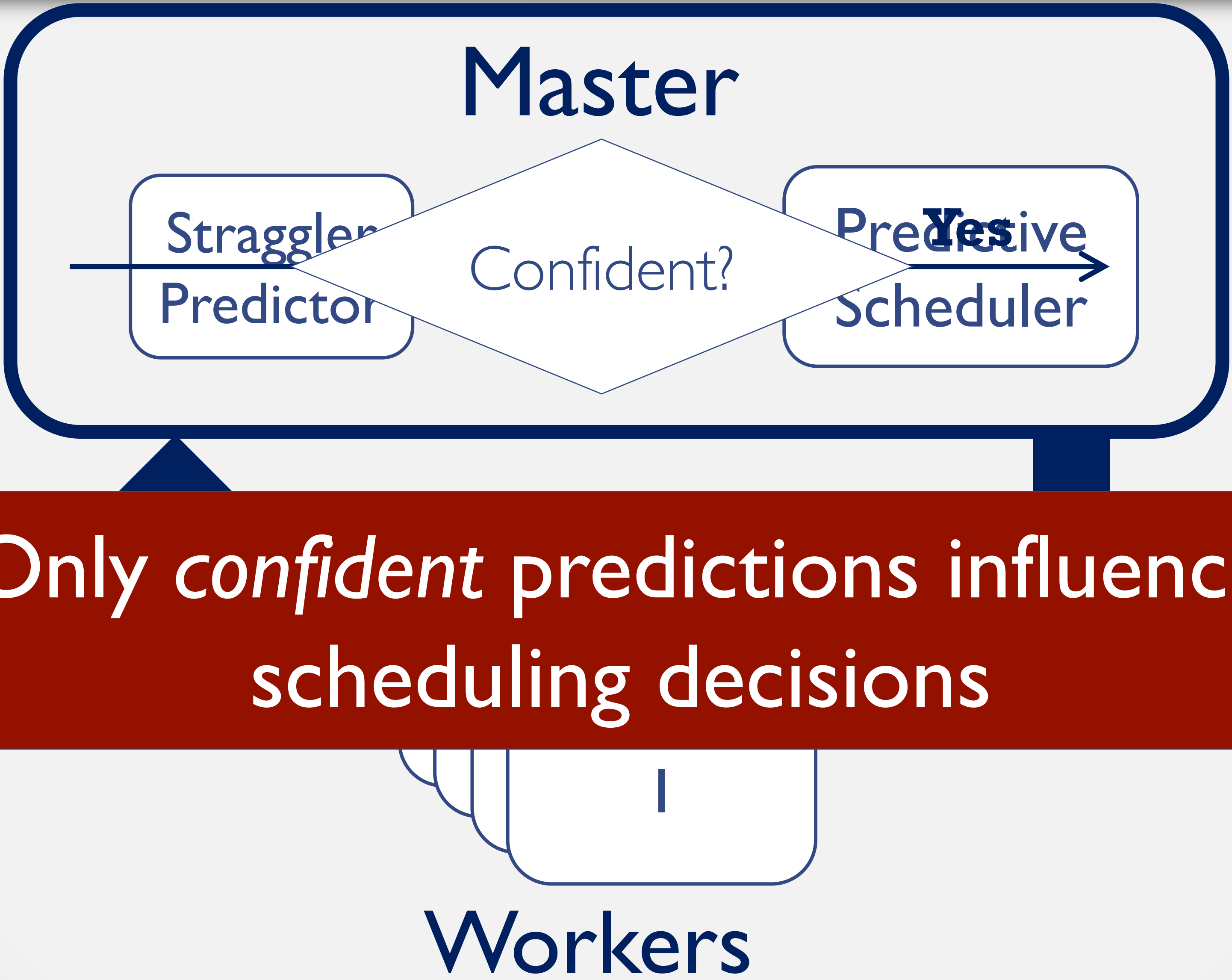
But, this managed to elongate jobs

Defer scheduling a task on a node that is predicted to create a straggler

~100 counters
CPU, Memory, Disk, Network, …

Wrangler, SoCC'14

Our proposal: Wrangler



Only *confident* predictions influence scheduling decisions

Wrangler, SoCC'14

Evaluation

- Does Wrangler Improve Job Completion Times?
  - With confidence measure, by up to 60%
- Does Wrangler Reduce Resources Consumed?
  - By up to 55%
- Load-Balanced clusters with Wrangler

ML for Systems - Guidelines

#1 Explore multiple domain-specific ways to formulate a problem

#2 Develop mechanisms to guard the system state from modeling errors

- Predicting straggler tasks
- Predicting straggler-causing situations on nodes

Training overhead? – No curated datasets

Too Many Models - We built per-node and per-workload models to be robust to heterogeneity…

Long Data-Collection time - In our 20 node set up, typically the training data collection phase took 2-4 hours…

Idea

Share data across nodes and workloads: Multi Task Learning

SDM'15, JMLR'16

Regularized Multi-Task Learning*

- T learning tasks
- Instead of one w, we need to learn a w for each of the T tasks

$$\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t$$

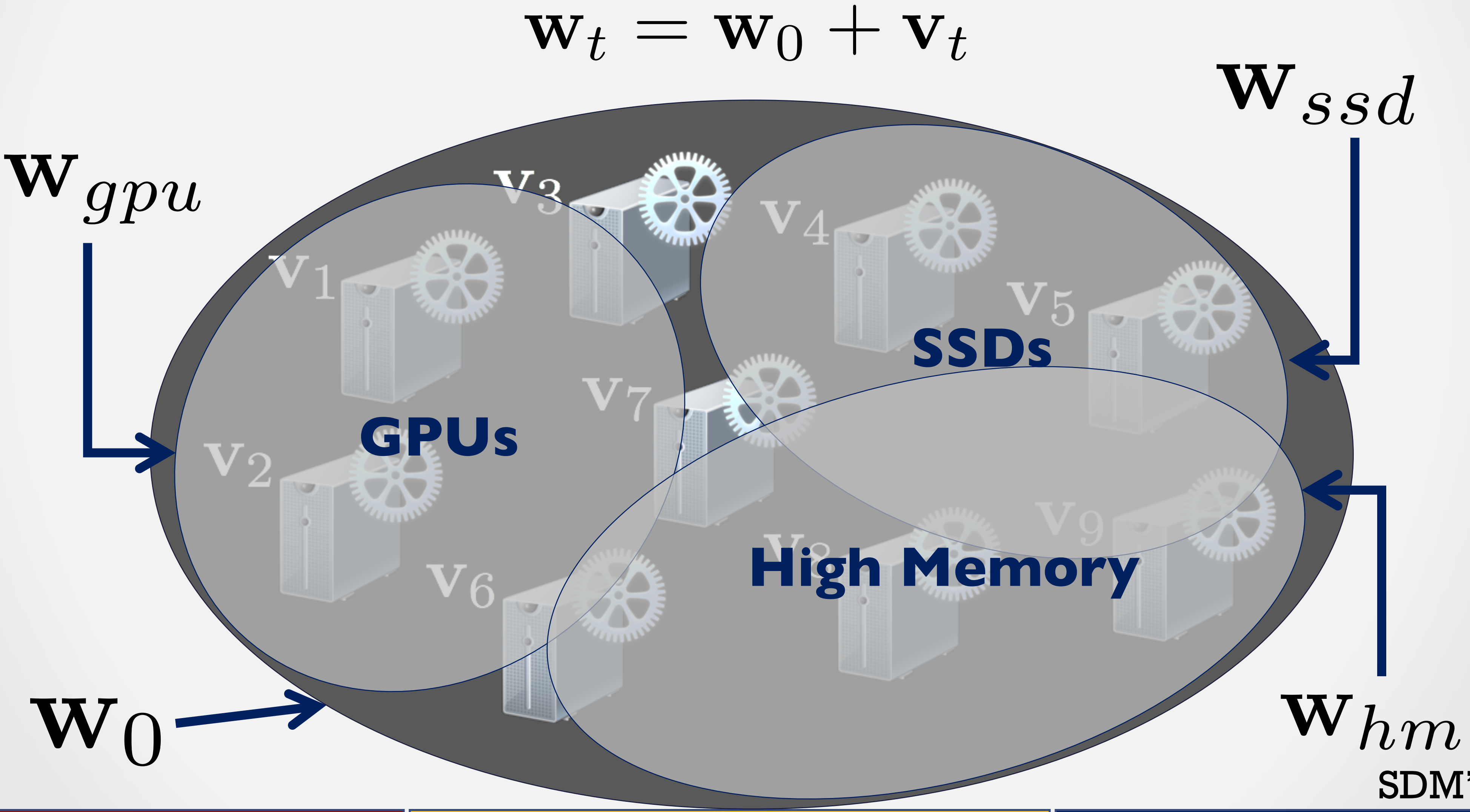
Common across all the learning tasks

Specific for a learning task, *t*

$$\min_{\mathbf{w}_0, \mathbf{v}_t, b} \lambda_0 \|\mathbf{w}_0\|^2 + \frac{\lambda_1}{T} \sum_{t=1}^{T} \|\mathbf{v_t}\|^2 + \text{Loss function}$$

*Evgeniou, et al., KDD 2004

Regularized Multi-Task Learning*



SDM'15, JMLR'16

Proposed Formulation

$$\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t + \mathbf{w}_g$$

Common across the tasks in a group, denoted by g

$$\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t + \mathbf{w}_{gpu} + \mathbf{w}_{ssd} + \ldots$$

SDM'15, JMLR'16

Proposed Formulation

$$\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t + \mathbf{w}_g$$

$$\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t + \sum_{p=1}^{P} \underbrace{\mathbf{w}_{p,g_p(t)}}$$

Weight vector of the g-th group of the p-th partition

All tasks belong to the same group

Each task is its own group

$$\mathbf{w}_t = \sum_{p=1}^{P} \mathbf{w}_{p,g_p(t)}$$

SDM'15, JMLR'16

Proposed Formulation

$$\min_{\mathbf{w}_{p,g,b}} \sum_{p=1}^{P} \sum_{g=1}^{G_p} \lambda_{p,g} ||\mathbf{w}_{p,g}||^2 + \text{Loss function}$$

SDM'15, JMLR'16

Proposed Formulation: Predicting Stragglers

The corresponding training problem is then,

$$\min_{\mathbf{w},b} \lambda_0 ||\mathbf{w}_0||^2 + \frac{\nu}{N}\sum_{n=1}^{N} ||\mathbf{w}_n||^2 + \frac{\omega}{L}\sum_{l=1}^{L} ||\mathbf{w}_l||^2 + \cancel{\frac{\tau}{T}\sum_{t=1}^{T} ||\mathbf{v}_t||^2} + \text{Loss function}$$

SDM'15, JMLR'16

Evaluation: MTL used in Real-world setting

- ✓ Reduced data collection time by 6x
- ✓ Better Generalization - Improved prediction accuracy by up to 7%
- ✓ Improved job completions – 99th percentile improved by 57.8%

SDM'15, JMLR'16

ML for Systems - Guidelines

#1 Explore multiple domain-specific ways to formulate a problem

#2 Develop mechanisms to guard the system state from modeling errors

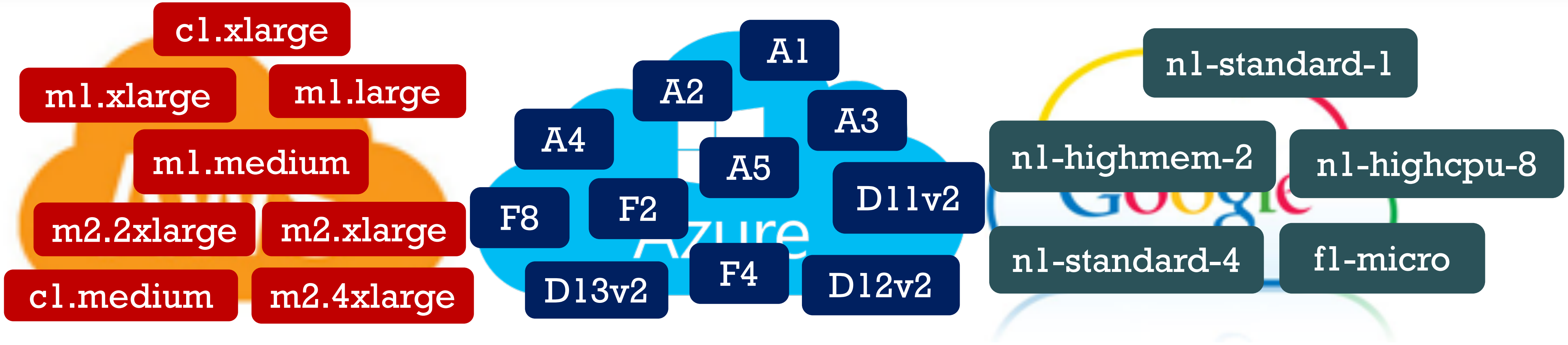
#3 Beware of the differences between similar-looking learning tasks

#4 When obtaining data is expensive, utilize existing data by exploring domain-specific correlation structures between learning tasks

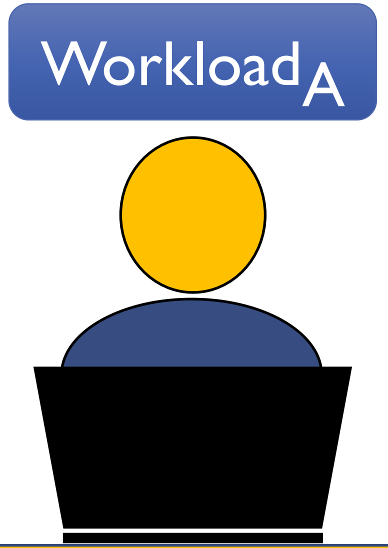
Two Problem Instances

- Job scheduling in datacenter environments
  - Problem - Long tail of job completions
- Resource allocation in public cloud environments
  - Problem - Right-sizing resources for workloads

Deploying a workload to the Cloud…



What VM type should I use for my workload?



Answer is **workload specific** and depends on **Cost** and **performance** goals

Objective: Enable informed cost-perf trade-off decisions

Run on all VM types?

Run user-workload task

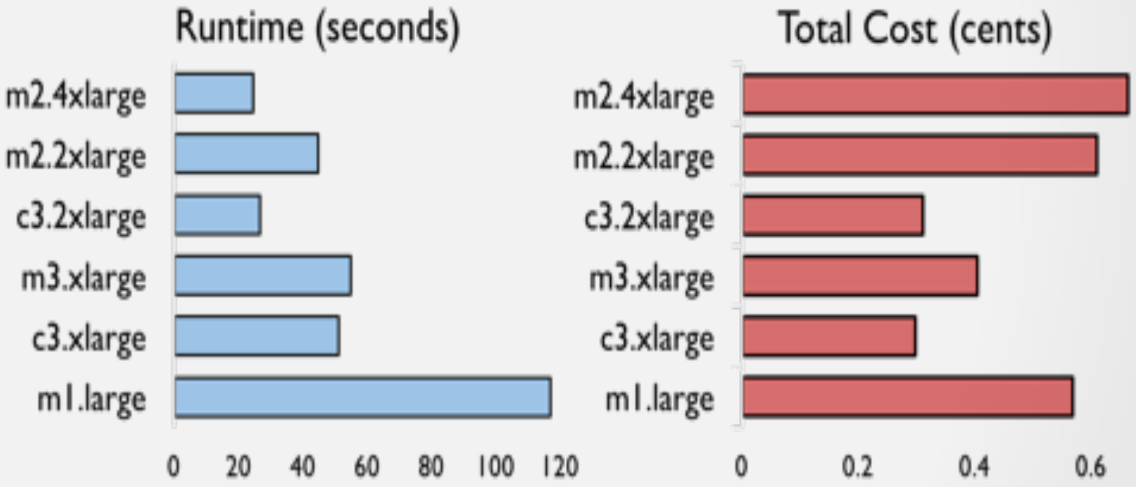
VM Types VM$_{I}$ VM$_{II}$ … VM$_{k}$

Trivial! but expensive!

Accurate

Cost Efficient

Specify cost/performance goals



Key Ingredient: Cost-Perf Trade-off Map

Our Proposal: PARIS

Run on all VM types?

Run user-workload task

VM Types VM_I VM_{II} … VM_k

Trivial! but expensive!

Accurate

Cost Efficient

Attempting to learn:

- VM type behavior, and
- Workload behavior

However, learning them simultaneously makes it expensive…

Our Proposal: PARIS

Learn
VM Type
behaviour

Learn
Workload
behaviour

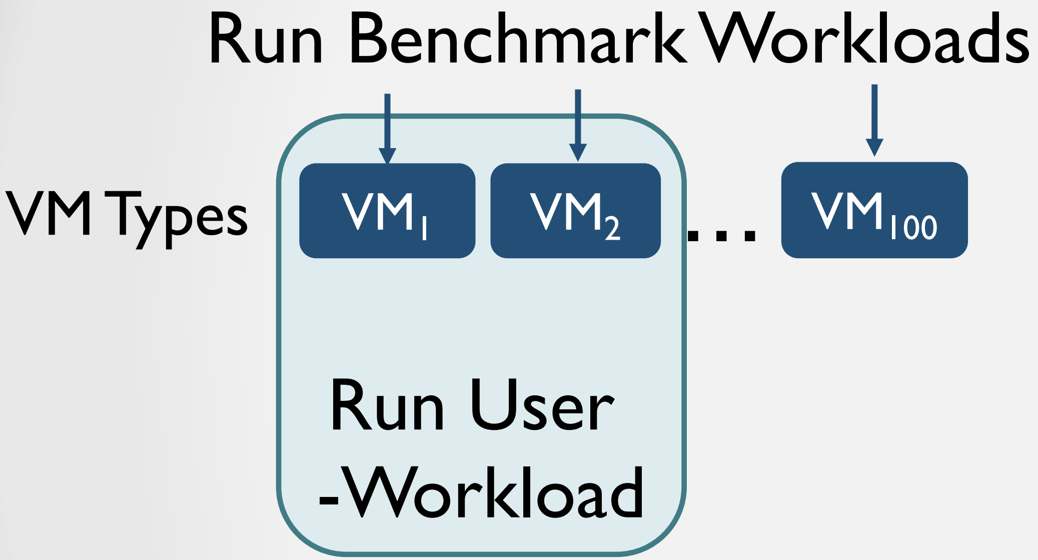
Attempting to learn:

- VM type behavior, and
- Workload behavior

However, learning them simultaneously
makes it expensive…

Key Insight: De-couple learning of VM types and workloads

Our Proposal: **PARIS**



Extensive benchmarking to model relationship between VM types

Cost Efficient

Accurate

Light-weight fingerprinting to model the relationship between user workloads and benchmark workloads

g: {**Benchmark Data, Fingerprint**} → **Performance and variability**

Key Insight: De-couple learning of VM types and workloads

ML for Systems - Guidelines

#1 Explore multiple domain-specific ways to formulate a problem

#2 Develop mechanisms to guard the system state from modeling errors

#3 Beware of the differences between similar-looking learning tasks

#4 When obtaining data is expensive, utilize existing data by exploring domain-specific correlation structures between learning tasks

#5 Develop triggers for re-learning to avoid biased predictions

#6 For cost-efficiency and generalizability, decouple learning of different systemic aspects

ML for Systems - Guidelines

#1 Explore multiple domain-specific ways to formulate a problem

#2 Develop mechanisms to guard the system state from modeling errors

#3 Beware of the differences between similar-looking learning tasks

#4 When obtaining data is expensive, utilize existing data by exploring domain-specific correlation structures between learning tasks

#5 Develop triggers for re-learning to avoid biased predictions

#6 For cost-efficiency and generalizability, decouple learning of different systemic aspects

Neeraja J. Yadwadkar
neeraja@cs.stanford.edu