Machine Learning for Resource Management in the Datacenter and the Cloud

Neeraja J. Yadwadkar
PhD Student, UC Berkeley
November 28th, 2017
...have caused two major changes in computing:

1. Parallel data-intensive computational frameworks
2. Cloud environments

Physical Servers  Virtual Servers  Private Cloud  Public Cloud

Growth of Data and Emerging Applications
Traditional Resource Management Techniques

Task

Resource Requirements?

Physical Servers

Available Resources?

Expert knowledge

Heuristics
Simple modeling

Physical Servers

Virtual Servers

Private Cloud

Public Cloud

Azure

AWS

Google
Traditional Resource Management Techniques

Task

Resource Requirements?

Virtual Servers
Physical Servers
Private Cloud
Public Cloud

Expert knowledge
Heuristics
Simple modeling

Resource Management Challenges:

Large Scale
Lack of complete control
Commodity hardware
Shared resources
Heterogeneous nodes and workloads
Traditional Resource Management Techniques

Estimating **performance** and **resource requirements** of workloads has become more challenging.

…but the data these systems collect, contain performance and utilization indicators that can help us!

Resource Management Challenges:
- Large Scale
- Lack of complete control
- Commodity hardware
- Shared resources
- Heterogeneous nodes and workloads
Vision: Data-Driven Models for Resource Management

We need to extract insights from this data to derive effective actions: Data Driven Models.
Challenges: Data-Driven Models

Uncertainty
- Wrangler [SoCC 2014]
- Modeling Uncertainty

Cost of Training
- Multi-Task Learning
  - For Efficient Training
    - [SDM 2015]
    - [JMLR 2016]

Research Goal
Achieve faster and predictable performance while reducing cost, by building data-driven models

Cloud-Hosted Systems
- PARIS [SoCC 2017]
  - Learning to Generalize from Benchmarks

Predictive Scheduling
- Resource Allocation

Distributed Systems

This talk
Research Goal

Achieve *faster and predictable performance while reducing cost*, by building *data-driven models*
• Hard-to-predict slow running tasks, called **stragglers**, impede job completion

• **Classify** machine state to be healthy or straggler-prone based on resource usage statistics,
  • and build a predictive scheduler

• Job completions improve by up to **60%** while reducing the resources consumed by up to **55%**

• **Machine Learning models used in a closed loop enable proactive mechanisms**
Parallel Data Intensive Computational Frameworks

Job queue

Master

Slaves

Job completed
Despite addressing data-skew, and blacklisting faulty hardware or slow nodes, stragglers continue to exist…
Impact of Stragglers:

We measure the potential in speeding up jobs in the trace using the following crude analysis: replace the progress rate of every task of a phase that is slower than the median task with the median task’s rate. If this were to happen, the average completion time of jobs improves by 47%, 29% and 36% in the Facebook, Bing and Yahoo! traces, respectively; small jobs ($\leq 10$ tasks) improve by 49%, 38% and 41%.
Stragglers

Tasks of a job

Task Execution Time

Median

Threshold

As defined in literature [Mantri OSDI’10]
Speculative Execution

Job queue

- In progress
- Replicating
- Job completed

Wasted Resources

Wasted Time in detecting stragglers
Existing Approaches: Design Space

Wasted Resources

Wasted Time in detecting stragglers

Replicate

Speculative Execution

OSDI'04

LATE

OSDI'08

Mantri

OSDI'10

Dolly

NSDI'13

Wrangler

Wait
Design Goals

I. Identify stragglers as early as possible

II. Schedule tasks for improved job finishing times
   
   1. To avoiding creation of stragglers
   2. To avoid replication

Avoid Wasting Time in detecting stragglers

Avoid Wasting Resources
1. Identify stragglers as early as possible

Classify machine state to be healthy or straggler prone
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
Job Scheduling in Data Intensive Computational Frameworks

- Master
- Job Scheduler
- Worker
- Workers
- Heartbeats
- Scheduling Decisions
Our proposal: Wrangler

Master

Straggler Predictor → Predictive Scheduler

Heartbeats

Utilization Counters

Workers

Scheduling Decisions
Our proposal: Wrangler

Master

Straggler Predictor → Predictive Scheduler

Utilization Counters

Heartbeats

Scheduling Decisions

Workers

Worker 1
Selecting **Input Features**

**CPU**
- cpu_idle
- cpu_system
- cpu_user
- ...

**Memory**
- mem_buffers
- memCached
- mem_free
- mem_shared
- mem_total
- ...

**Disk**
- swap_free
- swap_total
- disk_free
- disk_total
- ...

**Network**
- bytes_in
- bytes_out
- ...

**Other**
- Jvm. memHeapUsedM
- Jvm. threadsBlocked
- Jvm. gcTimeMillis
- ...
- datanode.blocks_written
- ...
- rpc.callQueueLen
- rpc.OpenConnections
- ...

---

Selecting input features that slow down performance from 30 MB/s to 1 MB/s. The cluster scheduling system may have scheduled other tasks on the machine, causing the MapReduce code to execute more slowly due to competition for CPU, memory, local disk, or network bandwidth. A recent problem we experienced was a bug in machine initialization code that caused process.

**MapReduce**: Dean, et al, OSDI’04

**LATE**: Zaharia, et al, OSDI’08

**Mantri**: Ananthanarayanan, et al, OSDI’10

Stragglers can arise for many reasons, including faulty hardware and misconfiguration. Google has noted that speculative execution can improve job response times by 44%. [1]
Our proposal: Wrangler

Master

- Straggler Predictor
- Predictive Scheduler

Utilization Counters

Scheduling Decisions

Workers

Workers
Classification for Predicting Stragglers

For every node in a cluster,

Learn:

{Utilization Counters, Straggler/Non-Straggler} → Learning → Classifier

Predict:

{Utilization Counters} (~100) → Classifier → Straggler → Non-Straggler
Classification Technique: SVM

Non-Stragglers

Stragglers

Separating Hyper-plane

feature_1

feature_2
Classification Accuracy using SVM

Is this accuracy good enough?
Our proposal: Wrangler

- Master
  - Straggler Predictor
  - Predictive Scheduler

- Workers
  - Workers Utilization Counters
  - Scheduling Decisions
Predictive Scheduler (Naïve)

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

Stage₂: Task₄
Stage₁: Task₃
Stage₁: Task₂
Stage₁: Task₁

Deferred Assignment

Net Gain

Job Submitted
Job finished
Task₂ finished
Task₂ finished
Job finished
Does the Naïve approach improve job completion?

Workload: CC_b

Baseline: Speculative Execution
Our proposal: Wrangler

Only confident predictions influence scheduling decisions
Confidence Measure

- Non-Stragglers
- Stragglers

Separating Hyper-plane

Corresponds to $P$: confidence threshold
$p$: Confidence Threshold

Value calculated via cross-validation

Verified via empirical sensitivity analysis

(detailed in the SoCC’14 paper)
Aim 1: Does Wrangler Improve Job Completion Times?

Workload: CC_b

Confidence measure is the key!

Baseline: Speculative Execution
Aim II: Does Wrangler Reduce Resources Consumed?

Baseline: Speculative Execution

<table>
<thead>
<tr>
<th>Task</th>
<th>Percentage Reduction in Total Task-Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB2009</td>
<td>55.09</td>
</tr>
<tr>
<td>FB2010</td>
<td>24.77</td>
</tr>
<tr>
<td>CC_b</td>
<td>40.15</td>
</tr>
<tr>
<td>CC_e</td>
<td>8.24</td>
</tr>
</tbody>
</table>
Load-Balancing with Wrangler

- Without Wrangler: Few highly loaded nodes
- With Wrangler (p=0.7): Load distribution is more balanced
Wrangler: proactively avoids stragglers to enable faster and predictable jobs while using fewer resources
Uncertainty

Challenges: Data-Driven Models

Cost of Training

Research Goal

Achieve *faster and predictable performance* while *reducing cost*, by building *data-driven models*
Training overhead?

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

In our 20 node set up, typically the training data collection phase took 2-4 hours…
Global model?

Straggler-causing situations vary across nodes and workloads...

A global model results in poor overall accuracy
How do we reduce this time?

Observations:

• Underlying modeling task remains the same
• Learning from other similar tasks should help
  ➢ Reduce training data capture time
  ➢ Improve accuracy by generalizing better

Idea

Share data across nodes and workloads: Multi Task Learning
Regularized Multi-Task Learning

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

\[ w_t = w_0 + v_t \]

Common across all the learning tasks

Specific for a learning task, $t$

\[
\min_{w_0, v_t, b} \lambda_0 \|w_0\|^2 + \frac{\lambda_1}{T} \sum_{t=1}^{T} \|v_t\|^2 + \text{Loss function}
\]

*Evgeniou, et al., KDD 2004*
Regularized Multi-Task Learning

\[ \mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t \]
Proposed Formulation

\[ w_t = w_0 + v_t \]

\( w_{gpu} \)

\( w_{ssd} \)

\( w_{hm} \)
Proposed Formulation

\[ w_t = w_0 + v_t + w_g \]

Common across the tasks in a group, denoted by \( g \)

\[ w_t = w_0 + v_t + w_{gpu} + w_{ssd} + \ldots \]
Proposed Formulation

\[ w_t = w_0 + v_t + w_g \]

\[ w_t = w_0 + v_t + \sum_{p=1}^{P} w_{p,g_p}(t) \]

- All tasks belong to the same group
- Each task is its own group

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

\[ w_t = \sum_{p=1}^{P} w_{p,g_p}(t) \]
Proposed Formulation

$$\min_{w_{p,g,b}} \sum_{p=1}^{P} \sum_{g=1}^{G_p} \lambda_{p,g} \|w_{p,g}\|^2 + \text{Loss function}$$
Reduction to a Standard SVM

With an appropriate change of variable,

$$\min_{\tilde{w}, b} \lambda \|\tilde{w}\|^2 + \sum_{t=1}^{T} \sum_{i=1}^{m_t} \xi_{i,t}$$

s.t.

$$y_{i,t}(\tilde{w}^T \phi(x_{it}) + b) \geq 1 - \xi_{it} \ \forall i, t$$

$$\xi_{it} \geq 0 \ \forall i, t$$

Can use off-the-shelf solvers
Application to straggler avoidance:

Node 1

FB2009: \( v_1 \sum \) 

FB2010: \( v_4 \sum \) 

CC_e: \( v_7 \sum \)

Node 2

\( v_2 \sum \)

\( v_5 \sum \)

\( v_8 \sum \)

Node 3

\( v_3 \sum \)

\( v_6 \sum \)

\( v_9 \sum \)
Application to straggler avoidance:

\[ w_1 = w_0 + \]

\( \sum \)

NODE 1

FB2009

FB2010

CC_e
Application to straggler avoidance:

\[ w_1 = w_0 + w_{node_1} + \]
Application to straggler avoidance:

\[ w_1 = w_0 + w_{node_1} + w_{fb09} + \]
Application to straggler avoidance:

\[ w_1 = w_0 + w_{node_1} + w_{fb09} + v_1 \]
Application to straggler avoidance:

\[ w_1 = w_0 + w_{node_1} + w_{fb09} + v_1 \]
Proposed Formulation: Predicting Stragglers

The corresponding training problem is then,

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

Workload: FB2009

- Wrangler
- Our formulation

Note: Our dataset is balanced.
Evaluation I: Prediction Accuracy

Workload: FB2009

- Wrangler
- Our formulation

Note: Our dataset is balanced.
Evaluation II: Faster Jobs

Workload: FB2009

We need only a sixth of training data!
A new MTL formulation that captures structure of learning tasks to enable:
- More accurate models with lesser data
- Improved job completions

Showed Benefits of MTL on a real-world problem
Challenges: Data-Driven Models

Research Goal

Achieve faster and predictable performance while reducing cost, by building data-driven models

Uncertainty

This talk

Modeling Uncertainty

Wrangler [SoCC 2014]

Cost of Training

This talk

Machine Learning

Multi-Task Learning

For Efficient Training

[SDM 2015]

[JMLR 2016]

Predictive Scheduling

Resource Allocation

Distributed Systems

Cloud-Hosted Systems

PARIS [SoCC 2017]

Learning to Generalize from Benchmarks

Cloud-Hosted Systems

Distributed Systems

Machine Learning

Multi-Task Learning

For Efficient Training

[SDM 2015]

[JMLR 2016]
PARIS
Selecting the Best VM across Multiple Public Clouds:
A Data-Driven Performance Modeling Approach
What VM type should I use for my workload?

Answer is **workload specific** and depends on **Cost** and **performance** goals.
How do we choose the best VM?

Rules of thumb? (Read “Myths”!)

#1: Smaller is cheaper

#2: Bigger is better

#3: Similar configurations imply similar performance
Specify cost/performance goals

Objective: Enable informed cost-perf trade-off decisions

Run on all VM types?
Run user-workload task
Trivial! but expensive!

VM Types
VM\textsubscript{i}  \quad VM\textsubscript{II}  \quad \ldots  \quad VM\textsubscript{k}

Accurate
Cost Efficient

Key Ingredient: Cost-Perf Trade-off Map
However, learning them simultaneously makes it expensive…

Attempting to learn:
- VM type behavior, and
- Workload behavior

Our Proposal: PARIS

Run on all VM types?
Run user-workload task

VM Types
VM_1 VM_2 ... VM_k

Trivial! but expensive!

Accuracy
	- Accurate

Cost Efficiency
	- Cost Efficient
Our Proposal: PARIS

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 the relationship between VM types

Cost Efficient  Accurate

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

\[ g: \{ \text{Benchmark Data, Fingerprint} \} \rightarrow \text{Performance and variability} \]

Key Insight: De-couple learning of VM types and workloads
Challenges: Data-Driven Models

Uncertainty

Cost of Training

Research Goal

Achieve faster and predictable performance while reducing cost, by building data-driven models

What’s next?
Industry Interaction

Predictive Scheduling

- Cloudera
- YouTube
- Apple

Resource Allocation: PARIS

- AWS
- Capital One
- Alibaba.com
- Splunk
Looking Forward: Dealing with Change

Why not simple feedback loop?

Models become key components of the eco-system

How do we update the models efficiently and automatically?
I have worked with incredibly awesome collaborators!

Randy Katz
Burton Smith
Joseph Gonzalez
Bharath Hariharan
Ganesh Ananthanarayanan
Chiranjib Bhattacharyya
Sarah Bird
Wontae Choi
Giulio Zhou
Eugene Huang
Machine Learning for Resource Management in the Datacenter and the Cloud

Actions

Model

Modeling Uncertainty

Distributed Systems

Wrangler [SoCC 2014]

Machine Learning

Multi-Task Learning
For Efficient Training
[SDM 2015]
[JMLR 2016]

Data

Cloud-Hosted Systems

PARIS [SoCC 2017]
Learning to Generalize from Benchmarks

Virtual Servers
Physical Servers

Private Cloud

Public Cloud

Virtual Servers
Physical Servers

Physical Servers
Private Cloud

Public Cloud

Private Cloud

Public Cloud

Virtual Servers
Physical Servers

Private Cloud

Public Cloud

Data

Actions

Thank you!
neerajay@eecs.berkeley.edu
https://people.eecs.berkeley.edu/~neerajay