Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters

Yanghua Peng
The University of Hong Kong
yhpeng@cs.hku.hk

Yixin Bao
The University of Hong Kong
yxbao@cs.hku.hk

Yangrui Chen
The University of Hong Kong
yrchen@cs.hku.hk

Chuan Wu
The University of Hong Kong
cwu@cs.hku.hk

Chuanxiong Guo
Bytedance Inc.
guochuanxiong@bytedance.com

Presenter: Silvery Fu

What is the problem?

- What is the optimal resource allocation strategy for deep learning workloads?

What is the problem?

- What is the optimal resource allocation strategy for deep learning workloads?

- Why? Improve training completion **time** and resource **efficiency.**
  - specifically, avg. job completion time and makespan - across multiple jobs

What is the problem?

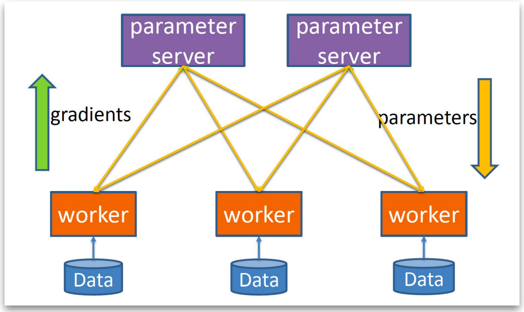
- What is the optimal resource allocation strategy for deep learning workloads?

- Why? Improve training completion **time** and resource **efficiency.**
  - specifically, avg. job completion time and makespan - across multiple jobs

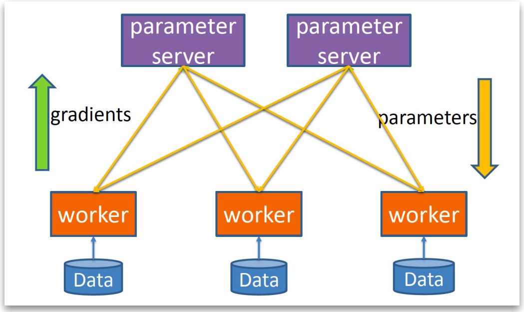
- How? Leverage more application-level semantics.

Application semantics: DL

- What semantics?
  - DL's job-task model: parameter server tasks and worker tasks
  - JCT correlates w/ convergence, iterative, ...



Application semantics: DL

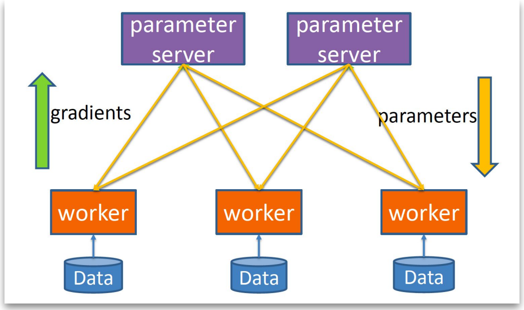


- What semantics?
  - DL's job-task model: parameter server tasks and worker tasks
  - JCT correlates w/ convergence, iterative, ...

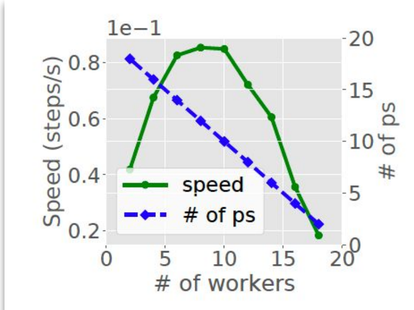
- How to capture these semantics?
  - Performance modeling

Application semantics: DL

- What semantics?
  - DL's job-task model: parameter server tasks and worker tasks
  - JCT correlates w/ convergence, iterative, ...



- How to capture these semantics?
  - Performance modeling



(a) 20 ps and workers

- How to use these semantics?
  - Tune knobs: `num_worker and num_parameter_server`
  - ..minimize makespan and average job completion time

Performance modeling

- Goal: predict the JCT for each DL training job
  - ..given the knobs: `num_worker` and `num_parameter_server`
  - JCT = **F**(`num_worker, num_parameter_server`)

Performance modeling

- Goal: predict the JCT for each DL training job
  - ..given the knobs: `num_worker and num_parameter_server`
  - JCT = **F**(`num_worker, num_parameter_server`)

- Hypothesis about **F?** Use application semantics, e.g.,
  - JCT is the sum of the time to complete each training step (process one minibatch)
  - How many training steps remaining? Estimate how far away from convergence.
  - The duration of each training step breaks down to:
    - forward/back propagation time
    - data transfer time
    - ...
  - Express these JCT breakdowns in terms of `num_worker and num_parameter_server`

Performance modeling

- Goal: predict the JCT for each DL training job
  - ..given the knobs: `num_worker and num_parameter_server`
  - JCT = **F**(`num_worker, num_parameter_server`)

graybox vs.
blackbox?

- Hypothesis about **F?** Use application semantics, e.g.,
  - JCT is the sum of the time to complete each training step (process one minibatch)
  - How many training steps remaining? Estimate how far away from convergence.
  - The duration of each training step breaks down to:
    - forward/back propagation time
    - data transfer time
    - ...
  - Express these JCT breakdowns in terms of `num_worker and num_parameter_server`

Performance modeling

- Curve 1: convergence curve
  - loss-based training convergence
  - f: completed epochs → training loss

$$l = \frac{1}{\beta_0 \cdot k + \beta_1} + \beta_2$$

learned
online

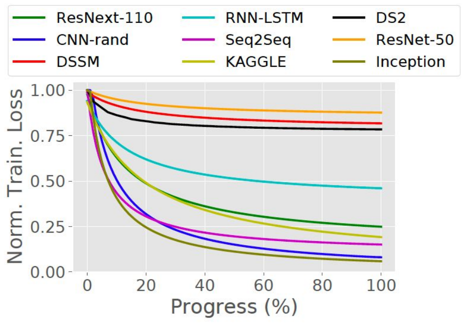


Figure 5: Training loss curves for different DL jobs

Performance modeling

- Curve 1: convergence curve
  - loss-based training convergence
  - f: completed epochs → training loss

$$l = \frac{1}{\beta_0 \cdot k + \beta_1} + \beta_2$$

learned online

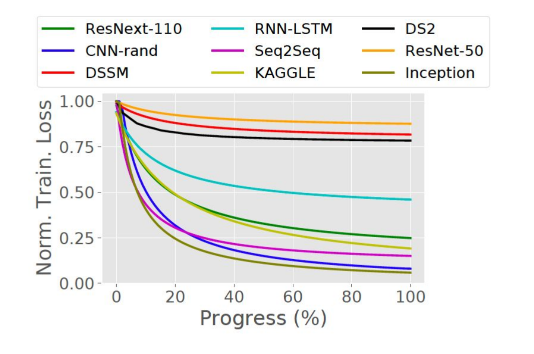
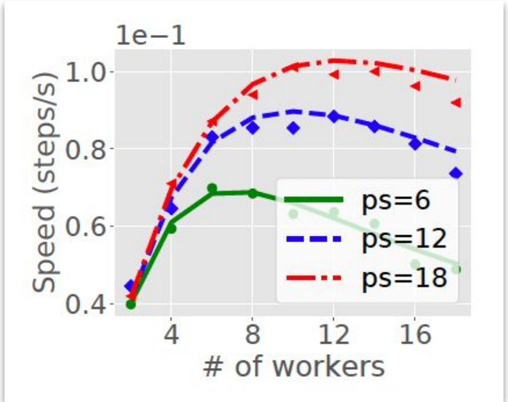


Figure 5: Training loss curves for different DL jobs

- Curve 2: resource-learning speed curve
  - f: resource configuration → learning speed

$$f(p, w) = \left(\theta_0 \cdot \frac{M}{w} + \theta_1 + \theta_2 \cdot \frac{w}{p} + \theta_3 \cdot w + \theta_4 \cdot p\right)^{-1}$$

learned "offline", adapted online



Performance modeling

- Curve 1: convergence curve
  - loss-based training convergence
  - f: completed epochs → training loss

$$l = \frac{1}{\beta_0 \cdot k + \beta_1} + \beta_2$$

learned online

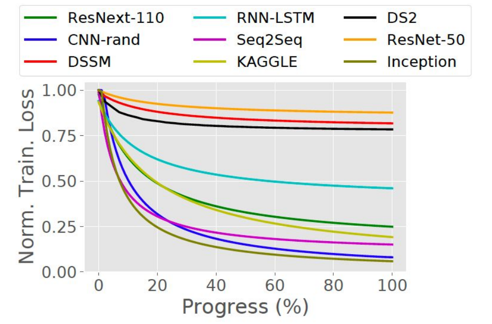
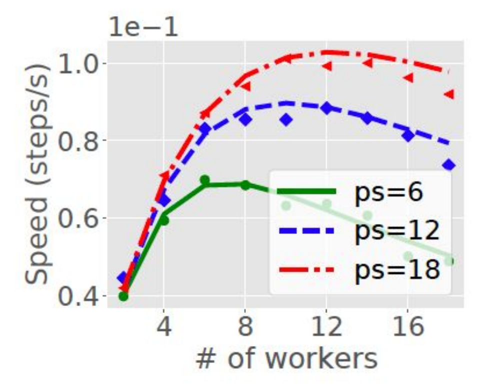


Figure 5: Training loss curves for different DL jobs

- Curve 2: resource-learning speed curve
  - f: resource configuration → learning speed

$$f(p, w) = (\theta_0 \cdot \frac{M}{w} + \theta_1 + \theta_2 \cdot \frac{w}{p} + \theta_3 \cdot w + \theta_4 \cdot p)^{-1}$$

learned "offline", adapted online

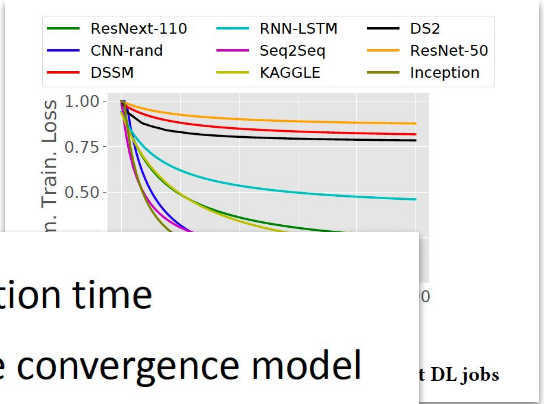


- Given the two curves:
  - Optimus decides the numbers of parameter servers and workers for *each* job

Performance modeling

- Curve 1: convergence curve

learned



t DL jobs

minimize $\sum_{j \in J} t_j$ → Minimize job completion time

remaining steps predicted by the convergence model

subject to: $t_j = \frac{\boxed{Q_j}}{\boxed{f(p_j, w_j)}} \quad \forall j \in J$ → job remaining time

training speed estimated by the speed function

$\sum_{j \in J} (w_j \cdot O_j^r + p_j \cdot N_j^r) \le C_r \quad \forall r \in R$ → capacity constraint

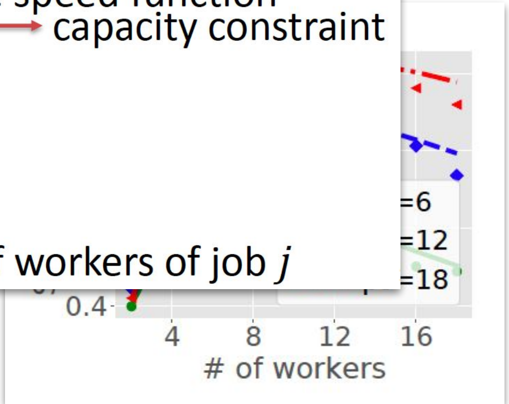
$p_j \in Z^+, w_j \in Z^+ \quad \forall j \in J$

p_j: number of parameter servers of job j w_j: number of workers of job j

- C

$f($

- G
  - Optimus decides the numbers of parameter servers and workers for *each* job



Task placement

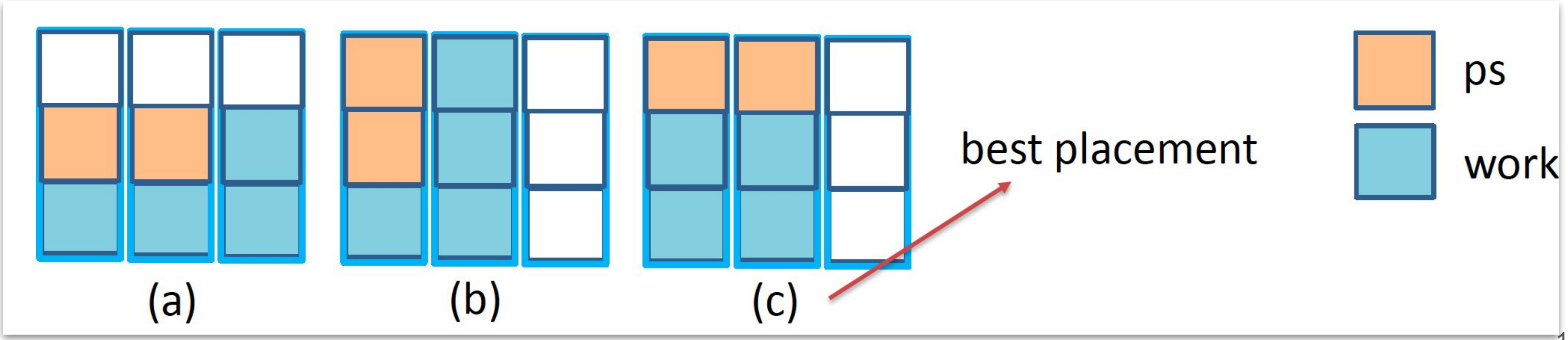
- Given the number of workers and the number parameter servers, decide where to place them at each scheduling iteration.

Task placement

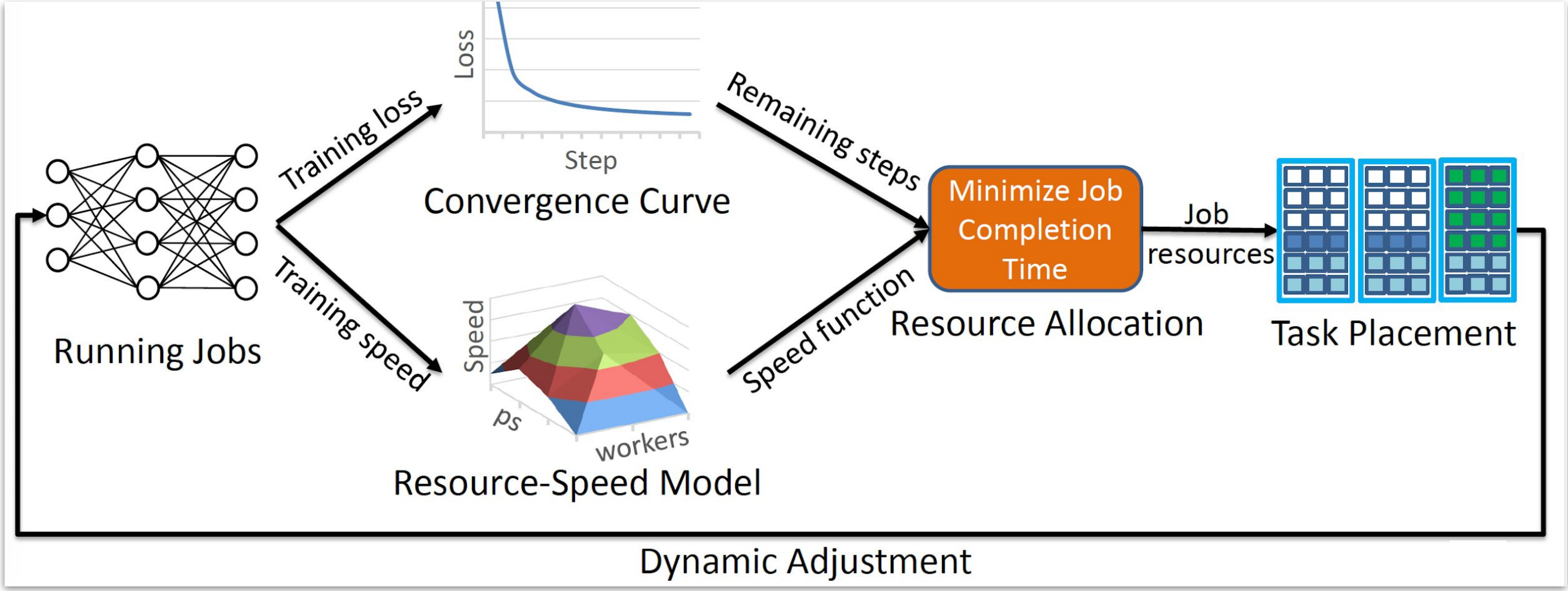
- Given the number of workers and the number parameter servers, decide where to place them at each scheduling iteration.

- Optimal placement should minimize the data transfer

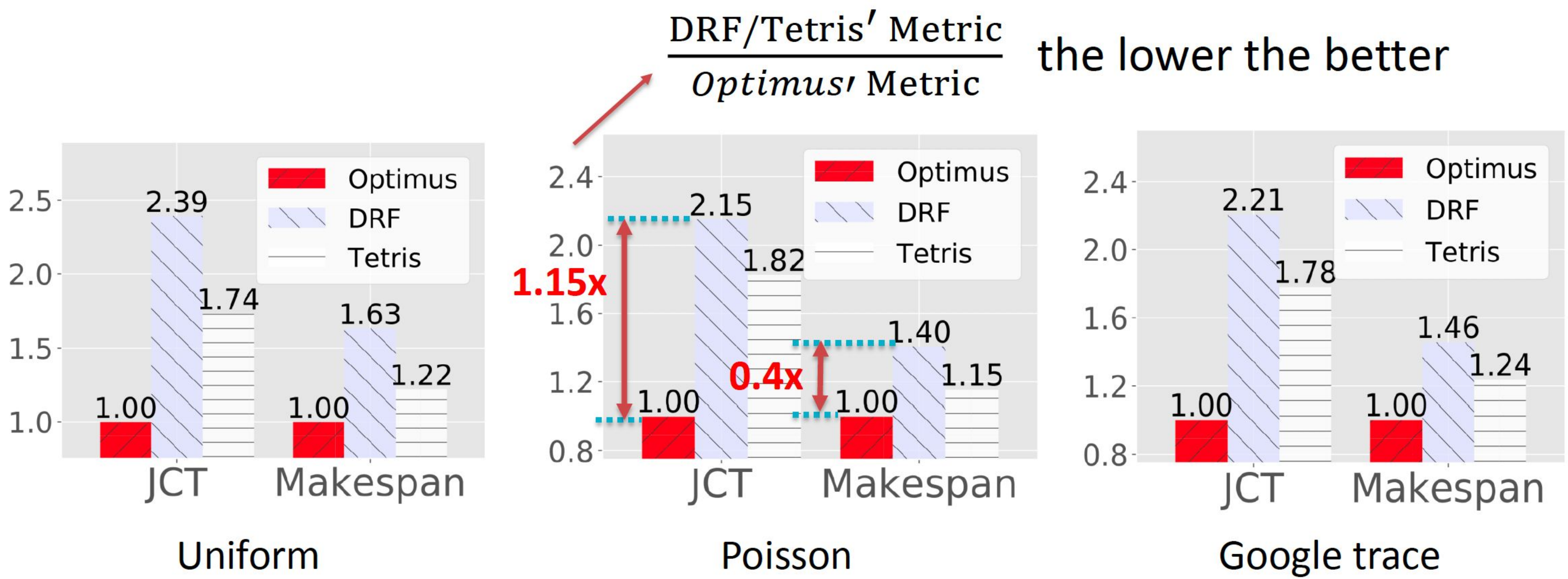
The paper has a proof for this



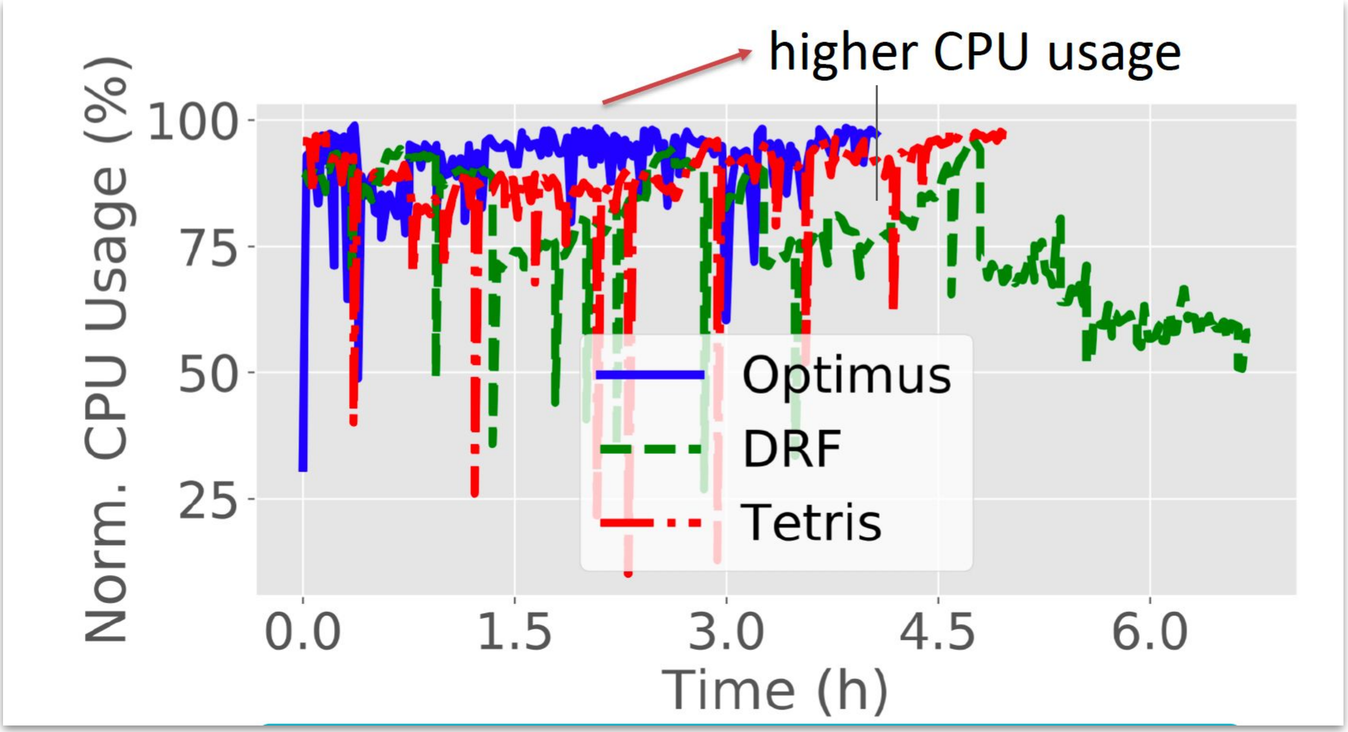
Put together:



Key results: avg. JCT and Makespan



Key results: cpu utilization



Prediction Accuracy vs. Performance

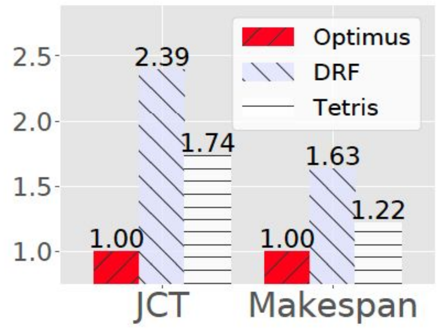
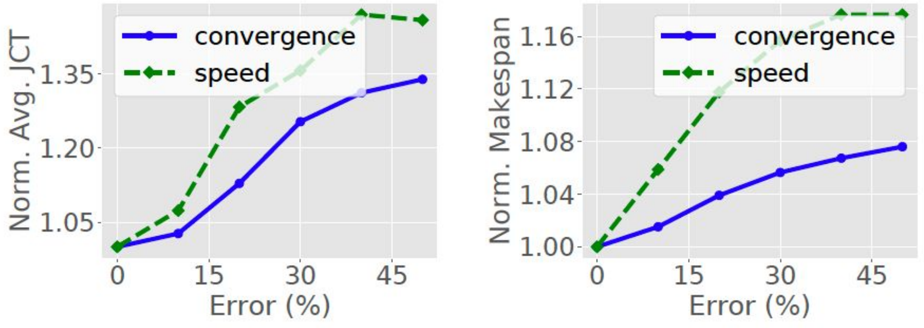


Figure 11: Performance comparison



(a) Average completion time **(b) Makespan**

Figure 15: Sensitivity to prediction errors

Prediction Accuracy vs. Performance

how much accuracy do we need?

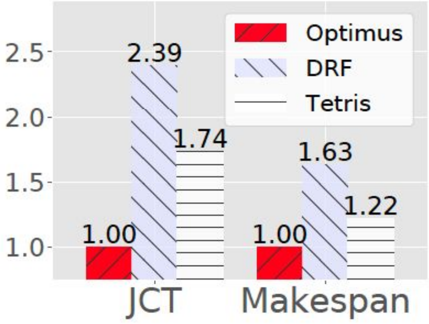
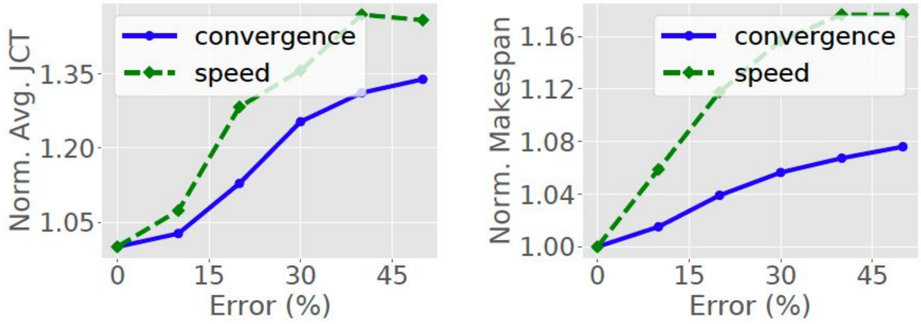


Figure 11: Performance comparison



(a) Average completion time **(b) Makespan**

Figure 15: Sensitivity to prediction errors

Prediction Accuracy vs. Performance

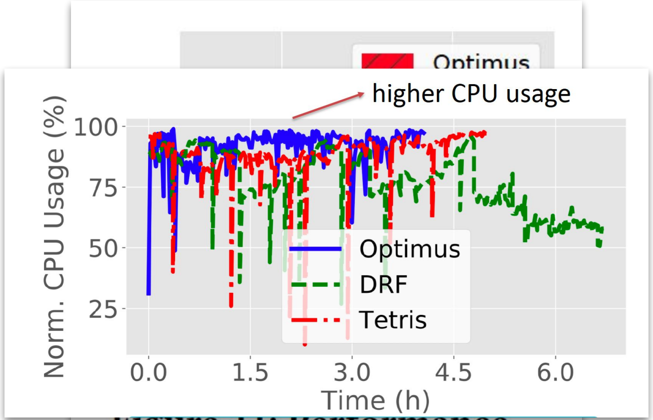
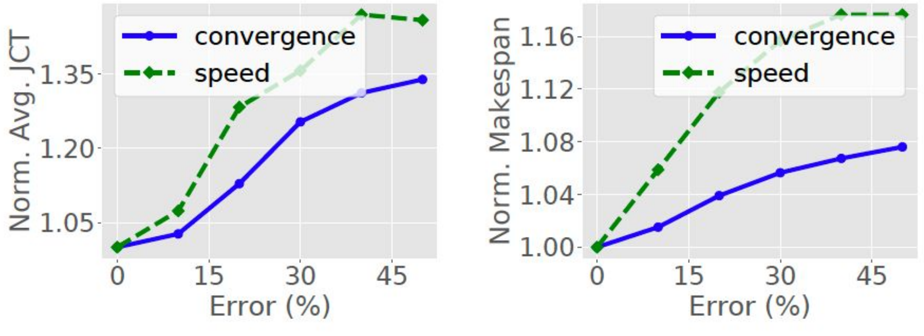


Figure 11: Performance comparison

how much accuracy do we need?



(a) Average completion time

(b) Makespan

Figure 15: Sensitivity to prediction errors

Prediction Accuracy vs. Performance

how "cheap" is accuracy?

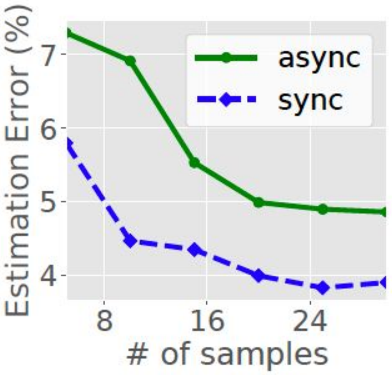
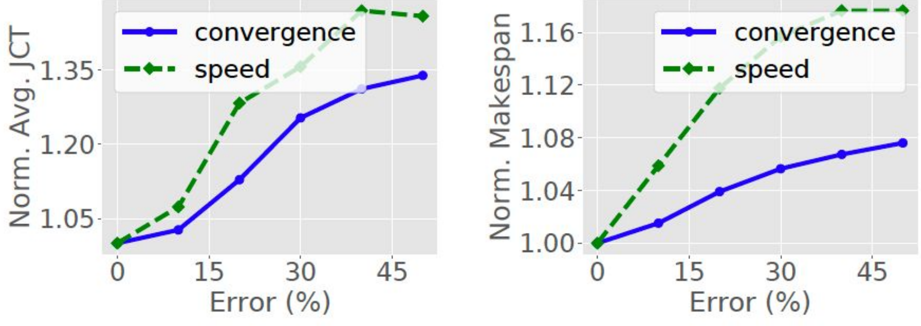


Figure 8: Estimation errors of training speeds

how much accuracy do we need?



(a) Average completion time **(b) Makespan**

Figure 15: Sensitivity to prediction errors

What are the metrics of success?

- Resource efficiency
  - makespan
- Training time
  - average job completion time

- Others:
  - scalability: scheduling overhead, scaling overhead
  - easy adoption

Long-term impact

System ⇄ ML

Deep Learning

- Increasing deep learning workloads in production clusters
  - Speech recognition
  - Object classification
  - Machine translation

Google Translate

- Many machine learning frameworks
  - TensorFlow
  - MXNet
  - PaddlePaddle

TensorFlow mxnet PaddlePaddle

Discussion

- "..Optimus has a relatively **small configuration space** (i.e., the number of tasks) and **5-10 sample** runs are enough for fitting the performance model quite accurately."
  - is this a useful assumption in practice? (each container fixed resources, horizontal scaling)

Discussion

- "..Optimus has a relatively **small configuration space** (i.e., the number of tasks) and **5-10 sample** runs are enough for fitting the performance model quite accurately."
  - is this a useful assumption in practice? (each container fixed resources, horizontal scaling)

- Who will use this? Who will run this?
  - Application user? Cluster operators?

Discussion

- "..Optimus has a relatively **small configuration space** (i.e., the number of tasks) and **5-10 sample** runs are enough for fitting the performance model quite accurately."
  - is this a useful assumption in practice? (each container fixed resources, horizontal scaling)

- Who will use this? Who will run this?
  - Application user? Cluster operators?

- Graybox vs. blackbox
  - can we extract the "formula" from offline profiling for a given type of application?

$$f(p, w) = (\theta_0 \cdot \frac{M}{w} + \theta_1 + \theta_2 \cdot \frac{w}{p} + \theta_3 \cdot w + \theta_4 \cdot p)^{-1}$$