Low-latency Job Scheduling with Preemption for the Development of Deep Learning

Hidehito Yabuuchi* (The University of Tokyo)
Daisuke Taniwaki, Shingo Omura (Preferred Networks, Inc.)

2019-05-20 OpML ‘19 @ Santa Clara

*Work done during an internship at Preferred Networks, Inc.

Outline

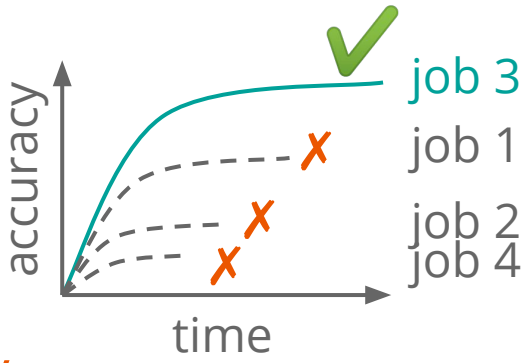
- Introduction
  - Deep Learning Development on Clusters
  - DL Training Jobs Characteristics
  - Related Work
  - Goal & Result Summary
- System Model
- Proposed Scheduling Algorithm: FitGpp
- Evaluation
- Conclusion

Deep Learning Development on Clusters

Efficient resource management of
clusters for deep learning development is essential

- DL dev conducted explosively in many fields
  - CV, NLP, robotics, bio, …
- Prompting usage of computing clusters
  - DL dev requires large computing resource including GPUs

DL Training Jobs Characteristics



- Many: *Trial-and-Error (TE)* jobs
  - for debugging & testing prototypes
  - supposed to be started immediately
    - Developers want to monitor progress soon
    - to save time for exploring numerous options
- Others: can be executed in *Best-Effort (BE)* manner
  - e.g. Large-scale evaluation after tuning

Related Work

Handle TE jobs & BE jobs mixture in limited situations

- *Big-C* [W. Chen+, ATC '17]
  - Container-based preemptive job scheduler
  - **GPUs not efficiently shared**
- *Optimus* [Y. Peng+, EuroSys '18], *Gandiva* [W. Xiao+, OSDI '18]
  - Job schedulers specialized for DL training jobs
  - **Only compatible with select DL frameworks**
- *Hawk* [P. Delgado+, ATC '15] , *Eagle* [P. Delgado+, SoCC '16]
  - Reserve part of cluster for immediate scheduling of short jobs
  - **Finding optimal reservation factor is not trivial**

Goals & Result Summary

- Low-latency scheduling of TE jobs
  - TE slowdown 95th: -99.6% cmp. with FIFO
- w/o greatly prolonging slowdown of BE jobs
  - BE slowdown 95th: -16.4% cmp. with other preemptive sched
- in versatile situations
  - Only requires that BE jobs can be suspended

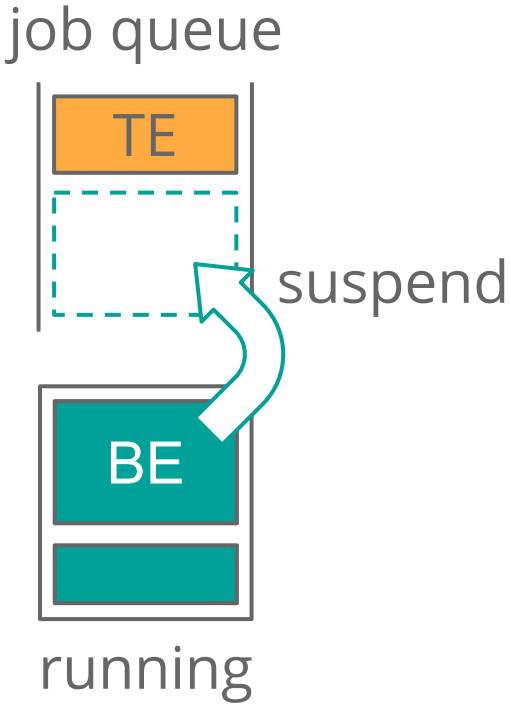
Outline

- Introduction
- System Model
  - System Model
  - Grace Period
- Proposed Scheduling Algorithm: FitGpp
- Evaluation
- Conclusion

System Model

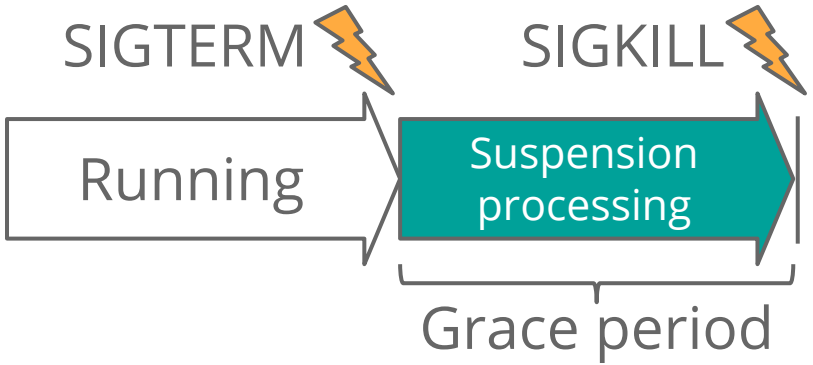
(Supposing systems similar to k8s)

- User-specified job spec
  - TE or BE
  - Resource demand
  - *Grace period (GP)*
- Schedulers may suspend BE jobs
  - placed back to job queue
  - Many DL frameworks support checkpointing



Grace Period

- Jobs do suspension processing before suspended
  - e.g. saving partially-trained models to storage
  - holding resources during GP
- Typical DL jobs require long GPs for checkpointing

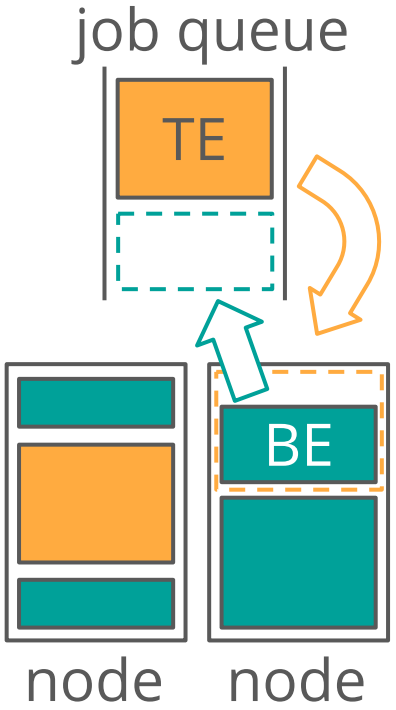


Outline

- Introduction
- System Model
- Proposed Scheduling Algorithm: FitGpp
  - Overview
  - Design
  - Formalization
- Evaluation
- Conclusion

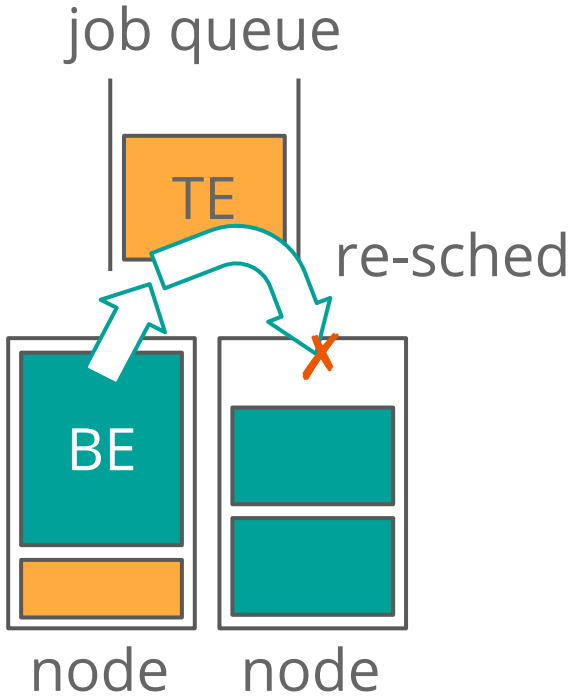
Fitting Grace Period Preemption (FitGpp)

- Built on FIFO principle
  - Suspended jobs placed back to *front of the queue*
- Prioritize TE jobs over BEs
  - to reduce latency
- Preempt BE jobs if resource is insufficient
  - *Which BE jobs?*
  - not to greatly delay BE jobs



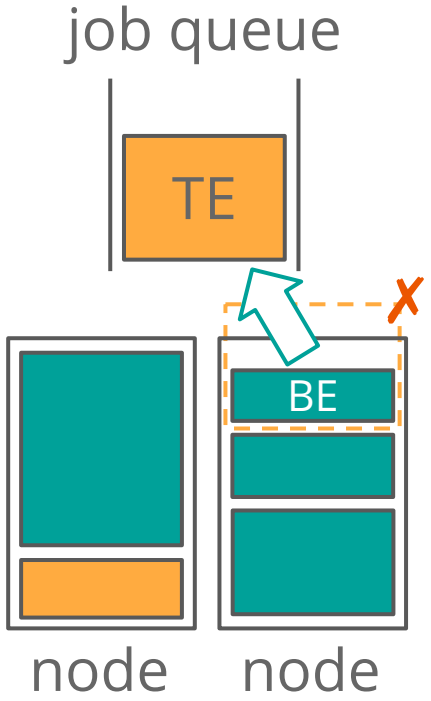
Minimizing Re-scheduling Intervals

- Preempting large BE jobs will likely cause head-of-line blocking
  - placed back to front of the queue
- Prefer BE jobs with small resource demand



Minimizing #Preemption

- Preempting too small BE jobs will likely require another preemption
  - increasing total re-scheduling time loss
- Select BE jobs that can offer enough resources for TE job



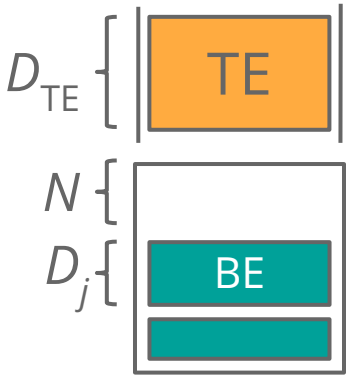
Minimizing Preemption-incurred Time Loss

- GP length determines
  time until incoming TE job starts running
- Prefer BE jobs with short GPs

Avoiding Starvation

- Limit max #preemption of each BE job
- making sure each BE job makes progress

Formalization



1. Evaluate *score* for each running BE job

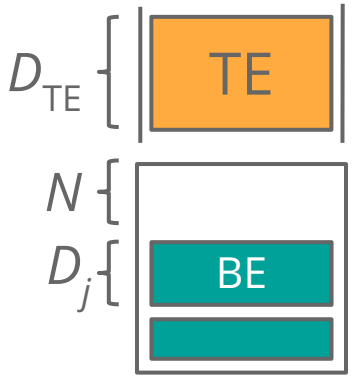
$$\text{Score}(j) := \frac{\|D_j\|}{\max_{j \in \mathcal{J}} \|D_j\|} + s \times \frac{\text{GP}_j}{\max_{j \in \mathcal{J}} \text{GP}_j}$$

Prefer small resource demand

Prefer short GP

D_j = normalized resource demand vector

Formalization



2. Preempt BE job that solves

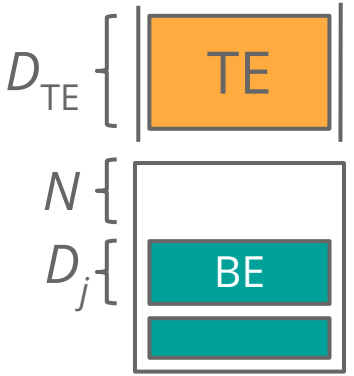
$$\arg\min \{\text{Score}(j) \mid D_{\text{TE}} \leq D_j + N \ \wedge \ \text{PreemptionCount}_j < P\}$$

Offer enough resource

Starvation-free

N = free resource of node on which *j* is running

Formalization



1. Evaluate *score* for each running BE job

$$\text{Score}(j) := \frac{\|D_j\|}{\max_{j \in \mathcal{J}} \|D_j\|} + s \times \frac{\text{GP}_j}{\max_{j \in \mathcal{J}} \text{GP}_j}$$

   D_j = normalized resource demand vector

2. Preempt BE job that solves

$$\arg\min \left\{ \text{Score}(j) \mid D_{\text{TE}} \leq D_j + N \ \wedge \ \text{PreemptionCount}_j < P \right\}$$

   N = free resource of node on which j is running

Not rely on exec time estimation

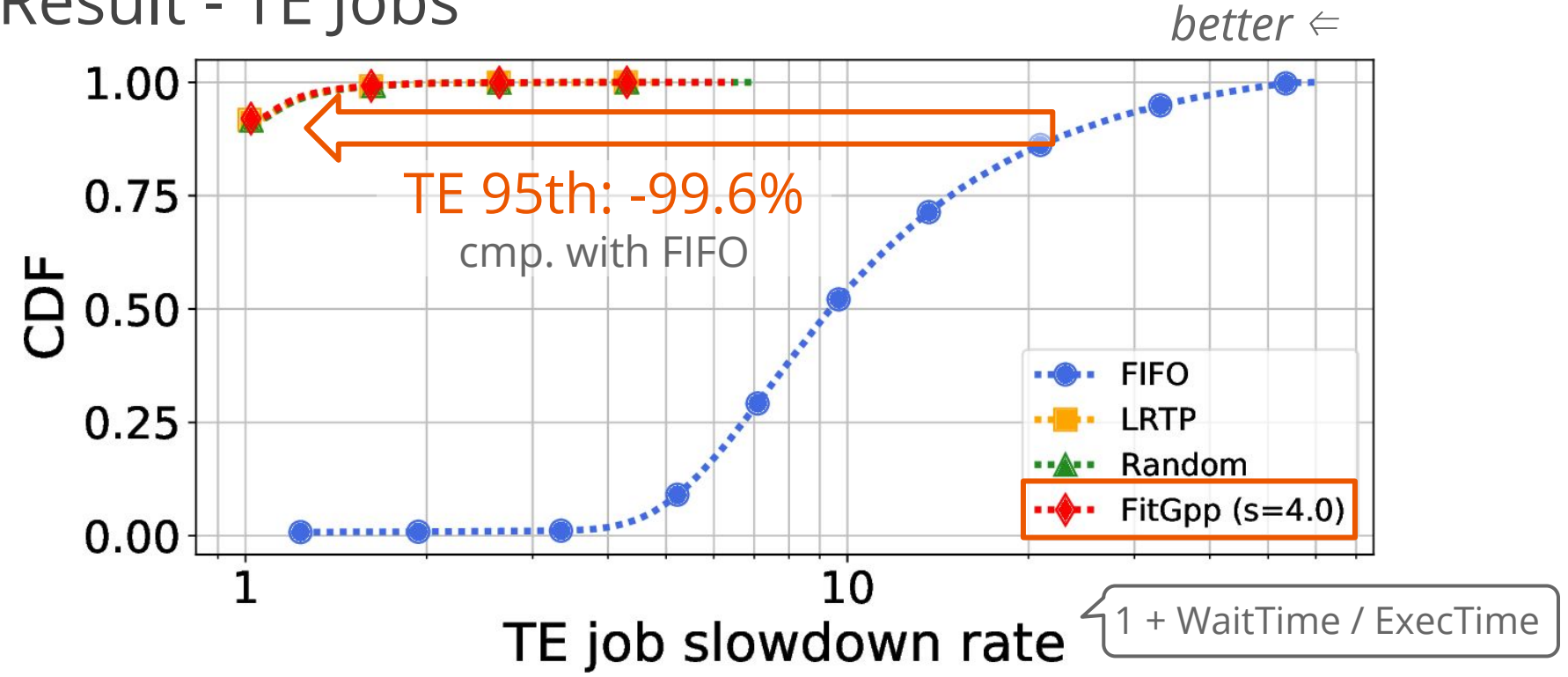
Outline

- Introduction
- System Model
- Proposed Scheduling Algorithm: FitGpp
- Evaluation
  - Setup
  - Result
  - Source of Improvement
- Conclusion

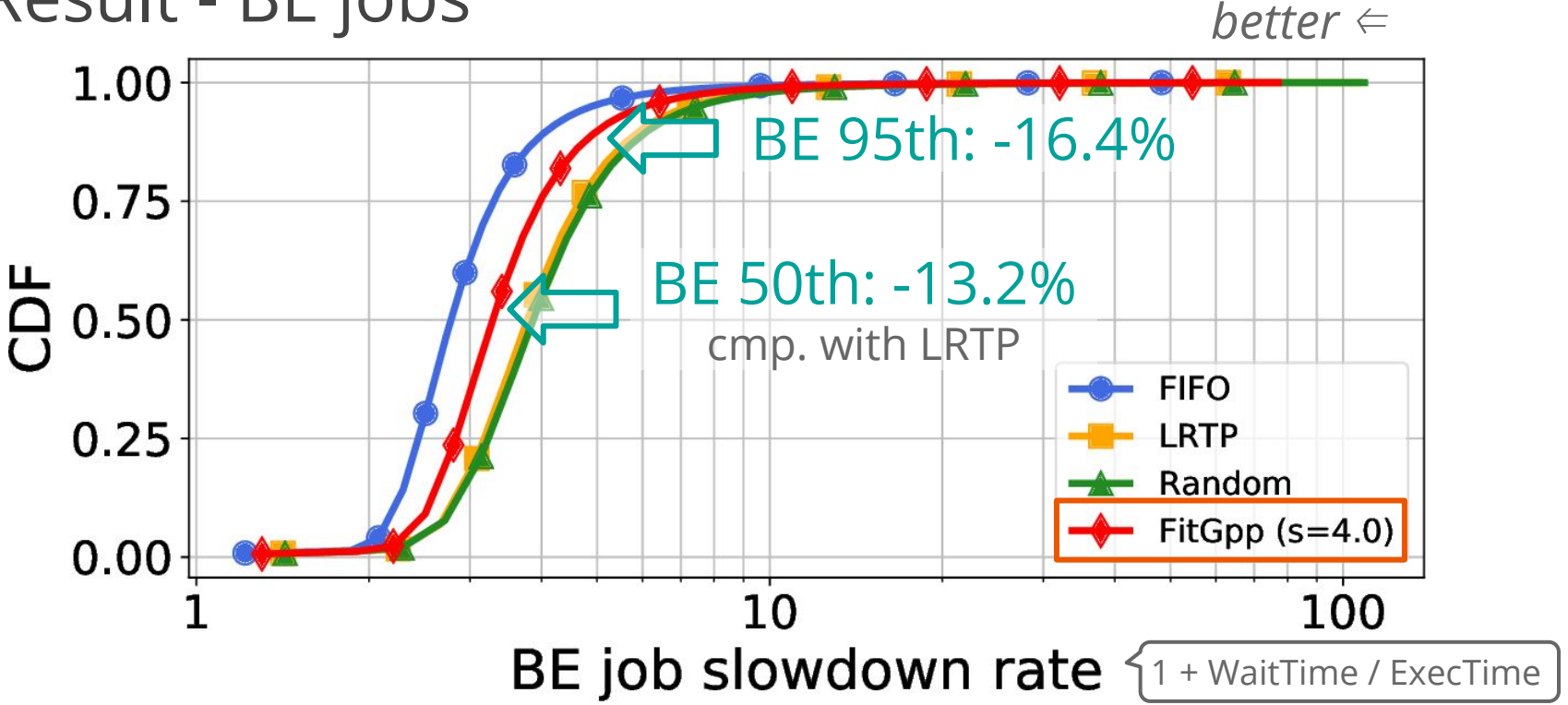
Setup

- Synthesized typical workload based on real jobs
  - 30% TE
- Compared with:
  - FIFO
  - *Longest Remaining Time Preemption* (LRTP; used in Big-C)
  - Random
- In simulated environment
- Max #preemption *P* = 1

Result - TE Jobs



Result - BE jobs



Sources of Improvement

- Re-scheduling intervals reduced to ~half

[min]

	50th	75th	95th
LRTP	4.0	4.0	5.0
Random	4.0	4.0	6.0
FitGpp	**2.0**	**2.0**	**4.0**

- Proportion of preempted jobs reduced to < 7.0%

LRTP	9.6%
Random	9.7%
FitGpp	**0.63%**

Outline

- Introduction
- System Model
- Proposed Scheduling Algorithm: FitGpp
- Evaluation
- Conclusion

Conclusion

- *FitGpp*, preemptive scheduling algorithm
  - reduce latency of TE jobs
    - TE slowdown 95th: -99.6% compared with FIFO
  - control slowdown of BE jobs
    - BE slowdown 95th: -16.4% compared with LRTP
  - applicable to versatile situations
- Future directions
  - Multi-node jobs in distributed DL
  - Apply to other fields than DL dev