Scheduling CPU for GPU-based Deep Learning jobs

Wencong Xiao, Zhenhua Han, Hanyu Zhao, Xuan Peng, Quanlu Zhang, Fan Yang, Lidong Zhou
- Beihang University, Microsoft Research, The University of Hong Kong, Huazhong University of Science and Technology, Peking University

Existing Approaches of Scheduling CPU

**Isolated GPU, shared CPU**
- Allocate deep learning jobs only considering GPU
- No CPU isolation, potential interference
- Hard to diagnose job performance

**Based on task role**
- Heuristic number of CPUs to each role of tasks
- E.g.: 3-core for parameter-server, 4-core and 1-GPU for worker

**Bind CPU and GPU**
- Fairly divide CPUs to the GPUs
- Each GPU will get a certain number of CPU regardless of job type

CPU is Undervalued

(1) **CPU number affects performance**
- Incorrect CPU allocation affects GPU-based deep learning jobs up to 15x
- Given insufficient CPU number, different jobs show different slowdown

(2) **Heterogeneous CPU demand across jobs**
- Some applications (e.g., Overfeat) require almost all the CPUs to extract 1-GPU’s maximal performance
- RNN language model needs only 1-CPU to fully extract 1-GPU’s performance

(3) **Better GPU, more CPU**
- DL jobs are mixed with Ops in GPU (e.g., Convolution, Matrix Multiplication) and Ops in CPU (e.g., data augmentation)
- With better GPU, Ops in GPU is faster, making the Ops in CPU become bottleneck
- Different jobs shows different sensitivity moving around different types of GPU

(4) **Waving CPU demand over time**
- In training, GPU is highly utilized; in validation, CPU is dominating
- With more CPU allocated for validation, the validation time reduces a lot

Design and Preliminary Result

How to **automatically** decide **appropriate** CPU cores in **characteristic-aware** manner for effective GPU performance in a **heterogeneous** cluster?
- Light-weighted profiling for optimal experiment design based performance predictor
- Coarse-grained rescheduling
- Continual monitoring architecture

Performance predictor:

\[
S = \begin{cases} 
  P_0 \times \log C + P_1 \times C & C < \theta \\
  P_2 & C \geq \theta 
\end{cases}
\]

- C donates # of CPU cores. \( \theta \) is the sweet point. \( P_i \) is parameters

Preliminary result:
- Improve utilization by 19%
- Reduces the job completion time by 34%

CPU number required for best 1-GPU performance on P100 Azure VM (24 cores in total)