Part A
FORMULA FUNCTION FOR REMAINING EXECUTION TIME ESTIMATION

ReLu activation function $f(x)$ is improved as:

$$f(x) = \begin{cases} \frac{x+|x|}{2}, \alpha = 0 \\ \frac{1+\alpha}{2}x + \frac{1-\alpha}{2}|x|, \alpha \neq 0 \end{cases} \quad (1)$$

where x is symbolic tensor to compute the activation function, α is scalar of tensor, which is optional, aiming to slope for negative input, usually between 0 and 1. The default value of 0 will lead to the standard rectifier, 1 will lead to a linear activation function, and any value in between will give a leaky rectifier.

Part B
MORE EXPERIMENTS RESULTS

We evaluated our DLAforBT vs YARN capacity scheduler performance using 4 real world applications. The first case has been presented in the main paper. Here we describe cases 2, 3 and 4 and their experimental results.

A. Real world applications

Case 2: Number plate image recognition: This License Plate Recognition System (LPRS) recognises a vehicle plate license from images with edge detection used to identify points in digital images with discontinuities. A 40GB image data file is loaded and read once from disk. This application is predominantly ***CPU-intensive***.

Case 3: Hadoop log file text search: This application tracks Hadoop's logs to search for error information using a simple lambda expression based on the "error" string, identifying a cluster's health status and weakness. Its complexity is low and it consumes little CPU resource. A 40GB log file is buffered in memory, read and searched. Hence, this application is ***memory-intensive***.

Case 4: Hadoop data migration: In a Hadoop cluster, the input file is split into one or more blocks stored in a set of DataNodes (running on commodity machines). When data volume is huge, tasks split from jobs are deployed on one node, however the needed data may be stored on different nodes and even different racks. Thus the system needs to copy other nodes' data to this destination node. A 40GB telecommunications data file is copied and transmitted among nodes. It is ***I/O-intensive***.

B. Real application evaluation results

Locality rate, throughput, and completion time when running use cases 2, 3 and 4 together:

There are 3 different group combinations. Group 1 has 12,000 jobs including 4,000 use case 2 jobs, 4,000 use case 3 jobs and 4,000 use case 4 jobs and other group combinations are presented in horizontal axis in Fig. 1. Fig. 1 presents locality rate. DLAforBT's average locality rate is 48.85% and capacity scheduler's is 33.72%, which improves by 15.13%. the job combination changes, the throughput is steady. In addition, compared with example benchmark applications, the throughput improves by at least 11%.

In Fig. 2, DLAforBT's average throughput is 664.84 job units/s and capacity scheduler's is only 504.61 job units/s. DLAforBT improves by 71.75% for throughput. Whatever

Fig. 3 shows completion times of different job combinations. The average completion time is 2,451,812ms using DLAforBT and average completion time is 2,954,790ms. DLAforBT can reduce completion time by 17.00%, compared with YARN capacity scheduler.

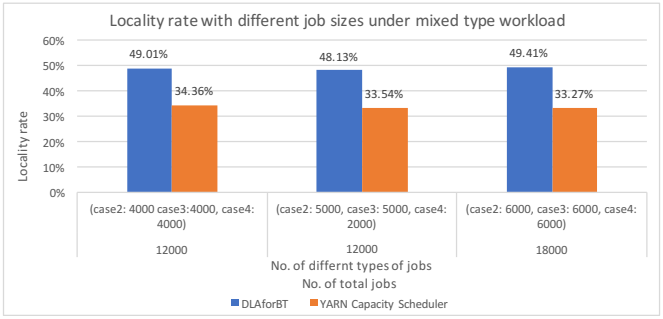

Fig. 1. Locality rate on a mixed use cases workload

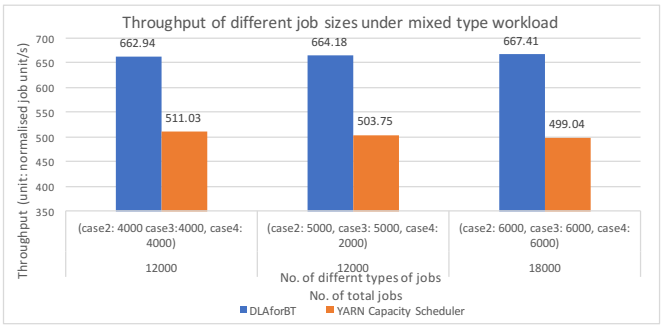

Fig. 2. Throughput on a mixed use cases workload

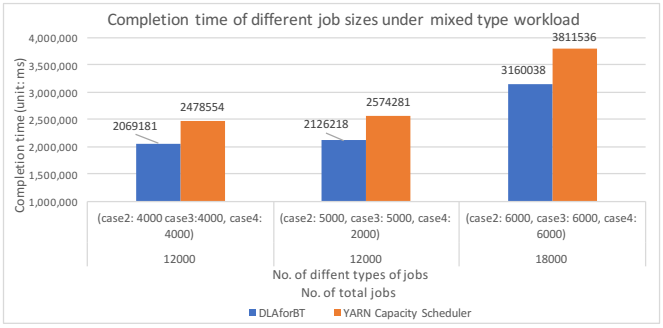

Fig. 3. Completion time on a mixed use cases workload