1 Setup

Applications: 10 jobs, each job contains 50 tasks. Each task runs 30 – 40 seconds. Submit 1 job every 10 seconds. Application heartbeat interval is 1 second.

Nodes: 50 nodes. The node heartbeat interval is 1 second.

Environment: The Yarn Scheduler Load Simulator (YARN-1021). The *yarn.scheduler.fair.assignmultiple* is closed for better analyzing continuous scheduling (minimizing affect from node heartbeat-triggered assignment).

2 Results

Here the *Unsorted* refer to the old mechanism implemented in YARN-1010, and the *Sorted* refer to new approach that sorts all nodes by the capacity.

Record the number of running containers for each node every 5 seconds. Assume C_i is the number of running containers for node N_i, and we have n nodes. We calculate the workload imbalance ratio as:

$$\phi = \frac{n \times \max_{i=1}^{n} C_i}{\sum_{i=1}^{n} C_i}$$

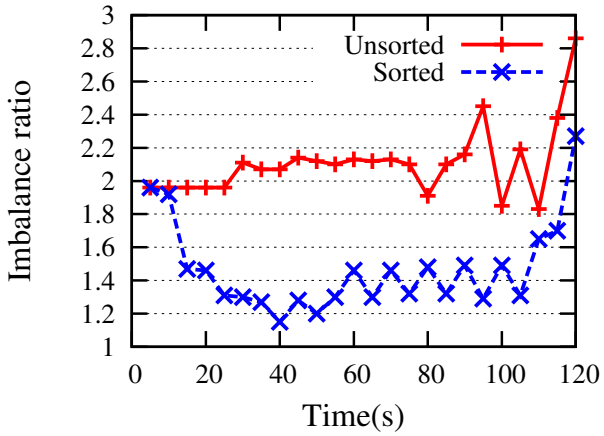

Figure 1: Workload imbalance ratio of Unsorted and Sorted mechanisms

Figure 1 illustrates the results. The *Sorted* approach can balance the workload among various nodes, as it always assigns the new container to the node with maximum capacity. The *Unsorted* approach always assigns container to the first node, as shown in attached logs.

At the end, the imbalance ratios for both approaches are very high. This is because many containers finish and only few containers are running.

The following two figures show the number of running containers for each node.

Time(s)		node1		node2		node3		node4		node5		node6		node7		node8		node9		node10	
5	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
15	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
20	6	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
25	6	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
30	8	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
35	8	4	4	3	4	3	4	3	4	4	4	3	4	3	4	4	4	3	4	4	4
40	9	4	4	4	4	4	4	4	4	4	5	4	5	4	4	4	5	4	5	5	5
45	8	4	4	4	3	3	3	4	3	4	4	4	3	4	4	3	4	4	4	4	3
50	9	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
55	8	4	4	4	3	3	4	4	3	3	4	4	3	4	3	4	4	4	4	3	4
60	9	4	4	4	4	4	5	4	4	4	4	4	4	4	4	5	4	4	5	4	4
65	8	3	3	4	3	4	3	4	4	4	4	3	4	4	4	3	4	4	4	4	4
70	9	4	4	4	4	4	4	4	4	5	5	4	4	4	4	4	4	4	4	4	4
75	8	4	4	4	4	4	4	4	4	4	3	3	4	4	4	4	4	4	3	4	3
80	9	4	4	4	4	5	4	4	4	5	4	4	5	4	5	4	4	5	4	5	4
85	7	3	4	4	3	4	3	4	4	3	4	4	3	3	4	4	4	3	3	3	4
90	9	4	4	4	4	4	4	4	5	4	5	4	4	4	4	4	4	4	4	4	4
95	8	4	3	4	4	3	3	4	4	4	3	4	4	4	4	4	4	4	4	4	3
100	8	3	3	3	3	3	3	4	3	3	3	3	3	4	3	3	4	3	3	4	3
105	5	3	3	2	2	3	2	2	3	3	2	2	2	3	3	2	2	3	2	2	3
110	5	3	3	2	2	2	2	2	2	2	2	2	2	3	3	2	2	2	2	2	3
115	3	2	2	2	2	2	2	1	2	1	2	2	2	2	2	1	2	1	1	1	2
120	3	2	1	2	1	1	1	1	1	1	1	2	1	2	1	1	1	1	1	1	2
125	2	0	1	1	1	1	1	0	1	0	0	1	1	1	0	1	1	1	0	0	0
130	2	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2: Number of running containers (the *Unsorted* mechanism)

Time(s)		node1		node2		node3		node4		node5		node6		node7		node8		node9		node10	
5	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
20	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
25	4	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
30	4	4	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
35	5	5	5	5	3	4	3	4	3	4	4	4	4	4	4	4	4	3	4	4	
40	4	4	4	4	4	3	3	3	3	3	3	4	3	4	3	3	3	4	3	4	4
45	5	4	5	5	5	4	4	3	3	4	3	4	4	4	3	4	4	3	4	4	4
50	4	4	4	4	4	3	3	3	3	3	3	4	3	3	3	3	3	3	3	4	3
55	4	4	5	5	5	4	4	3	3	4	4	4	3	4	4	3	4	3	4	4	4
60	4	3	5	4	4	3	3	4	3	4	4	3	3	3	3	3	3	3	4	3	3
65	5	4	4	5	5	3	3	5	3	4	4	4	4	4	4	3	4	4	4	4	4
70	4	4	3	4	4	3	3	5	3	3	3	3	3	4	4	3	4	4	3	3	3
75	5	4	3	4	5	4	4	5	4	4	4	3	4	4	4	4	4	3	4	3	3
80	4	4	3	3	5	4	3	4	4	3	4	3	4	3	3	3	4	3	3	3	3
85	5	5	4	4	4	3	3	5	4	3	4	4	5	4	3	4	4	3	3	4	4
90	4	5	4	3	3	3	3	4	3	3	4	3	4	4	3	4	4	3	3	3	3
95	5	5	4	4	4	4	4	5	4	4	4	4	4	3	4	4	4	3	4	4	4
100	3	5	4	3	3	3	3	4	4	4	3	4	4	3	4	3	3	3	4	3	3
105	2	4	3	3	3	3	3	3	2	3	3	3	4	3	2	3	3	3	2	2	3
110	2	4	3	3	2	3	3	3	2	2	3	2	3	3	2	2	2	3	2	2	2
115	1	2	3	1	2	1	2	1	1	2	2	2	3	2	2	2	2	1	2	2	1
120	1	1	2	1	1	1	2	1	1	1	2	1	3	1	1	1	1	1	1	2	1
125	1	0	2	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	0	1	1
130	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	1
135	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3: Number of running containers (the *Sorted* mechanism)