YARN-4090 Test Result

Test Case

Test Method:

Hadoop-sls

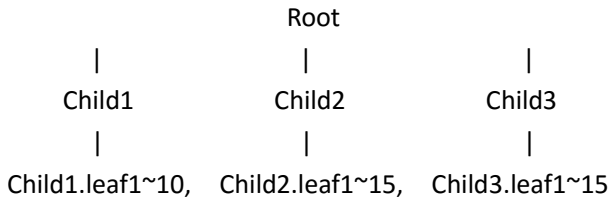
Hardware env:

40 cores, Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz; about 396 GB mem;

Software env:

Red Hat Enterprise Linux Server release 7.0;

Queues：



There’re a total of 40 leaf queues, and 2k apps runs randomly in these queues. For each app, there’re 100 maps and 20 reduces with resource needs <1 vcore, 1GB>, and each task runs 1 min. These apps needs 2000 * 120 = 240000 containers.

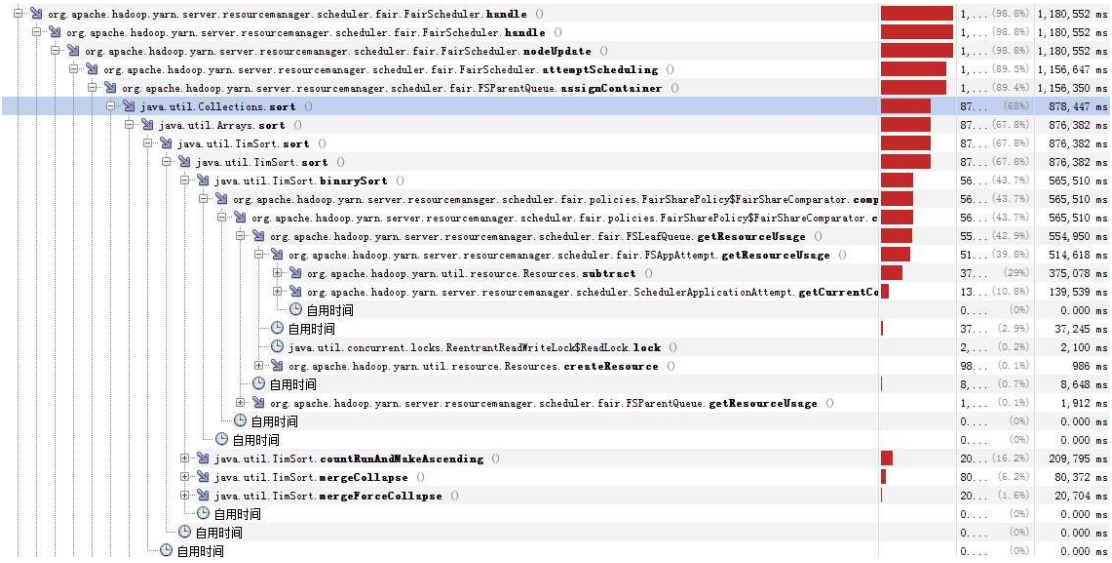
Virtual cluster:

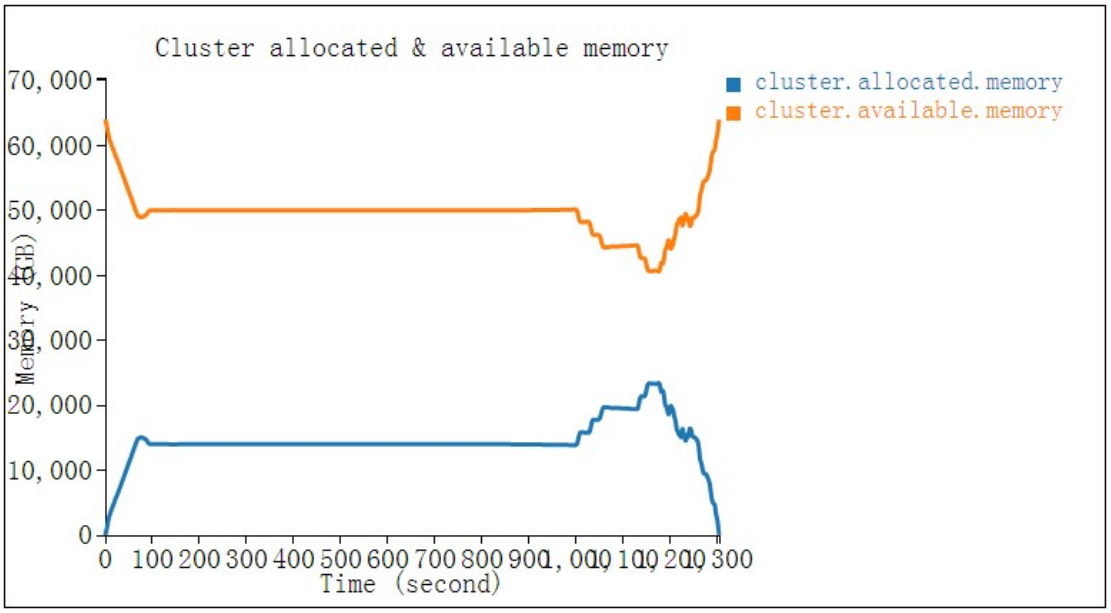
200 racks, and 10 nodes on each. For each node, there’re 32 vcores and 32 GB mem. If each container takes <1 vcore, 1GB>, the total number of container is 2000 * 32 = 64000.

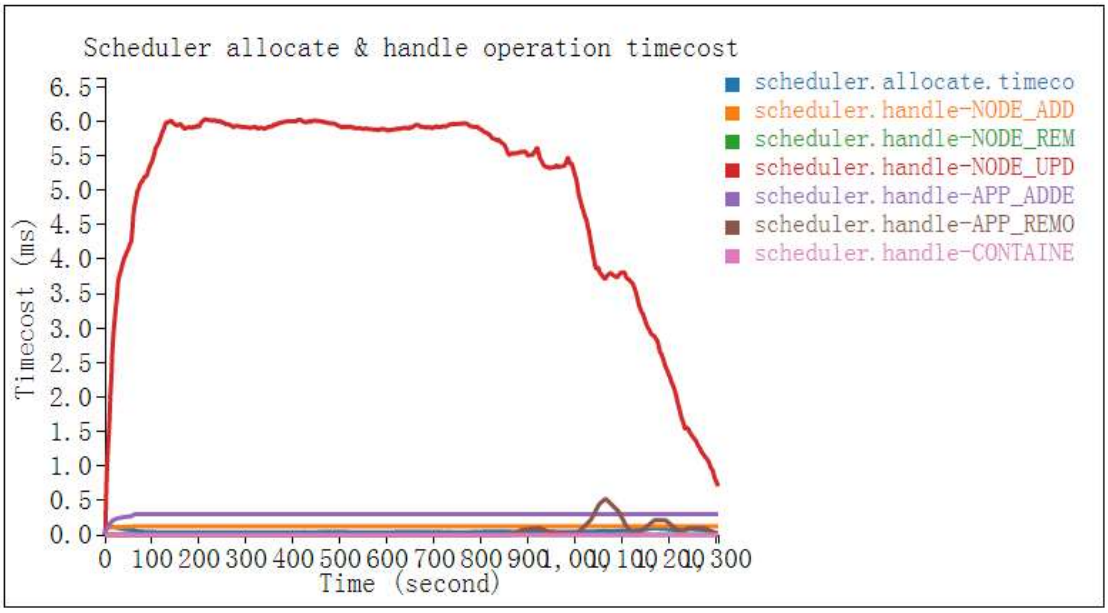
To test the concurrency, I submit all of the job at the start time.

Results

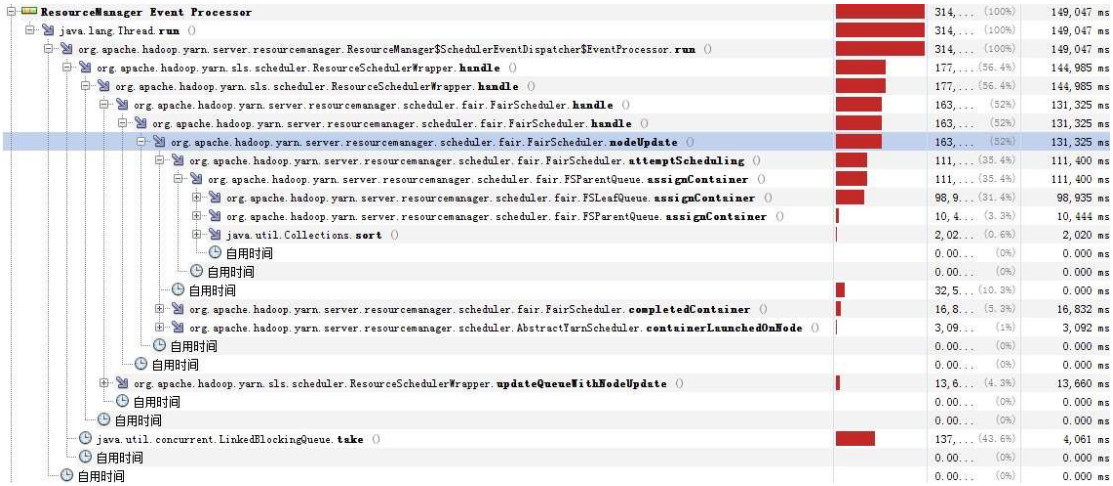
BEFORE:

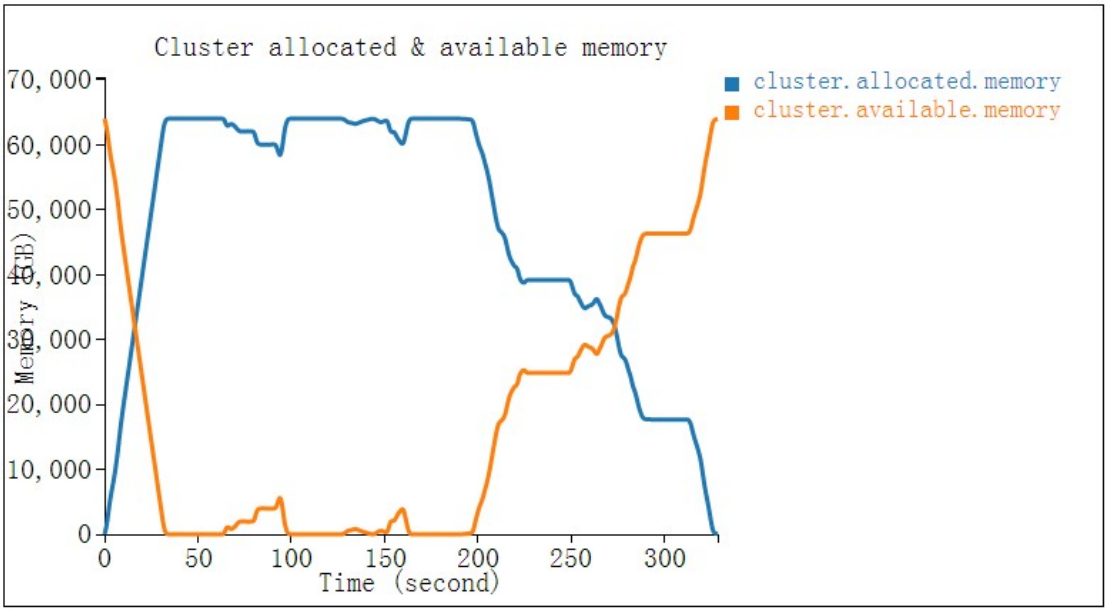

org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.handle ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.handle ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.nodeUpdate ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.attemptScheduling ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FSParentQueue.assignContainer ()
java.util.Collections.sort ()
java.util.Arrays.sort ()
java.util.TimSort.sort ()
java.util.TimSort.sort ()
java.util.TimSort.binarySort ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.policies.FairSharePolicy$FairShareComparator.comp
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.policies.FairSharePolicy$FairShareComparator.c
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FSLeafQueue.getResourceUsage ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FSAppAttempt.getResourceUsage ()
org.apache.hadoop.yarn.util.resource.Resources.subtract ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.SchedulerApplicationAttempt.getCurrentCo
自用时间
自用时间
java.util.concurrent.locks.ReentrantReadWriteLock$ReadLock.lock ()
org.apache.hadoop.yarn.util.resource.Resources.createResource ()
自用时间
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FSParentQueue.getResourceUsage ()
自用时间
自用时间
java.util.TimSort.countRunAndMakeAscending ()
java.util.TimSort.mergeCollapse ()
java.util.TimSort.mergeForceCollapse ()
自用时间
自用时间
自用时间
1,... (98.8%) 1,180,552 ms
1,... (98.8%) 1,180,552 ms
1,... (98.8%) 1,180,552 ms
1,... (89.5%) 1,156,647 ms
1,... (89.4%) 1,156,350 ms
87... (68%) 878,447 ms
87... (67.8%) 876,382 ms
87... (67.8%) 876,382 ms
87... (67.8%) 876,382 ms
56... (43.7%) 565,510 ms
56... (43.7%) 565,510 ms
56... (43.7%) 565,510 ms
55... (42.9%) 554,950 ms
51... (39.8%) 514,618 ms
37... (29%) 375,078 ms
13... (10.8%) 139,539 ms
0.... (0%) 0.000 ms
37... (2.9%) 37,245 ms
2,... (0.2%) 2,100 ms
98... (0.1%) 986 ms
8,... (0.7%) 8,648 ms
1,... (0.1%) 1,912 ms
0.... (0%) 0.000 ms
0.... (0%) 0.000 ms
20... (16.2%) 209,795 ms
80... (6.2%) 80,372 ms
20... (1.6%) 20,704 ms
0.... (0%) 0.000 ms
0.... (0%) 0.000 ms
0.... (0%) 0.000 ms

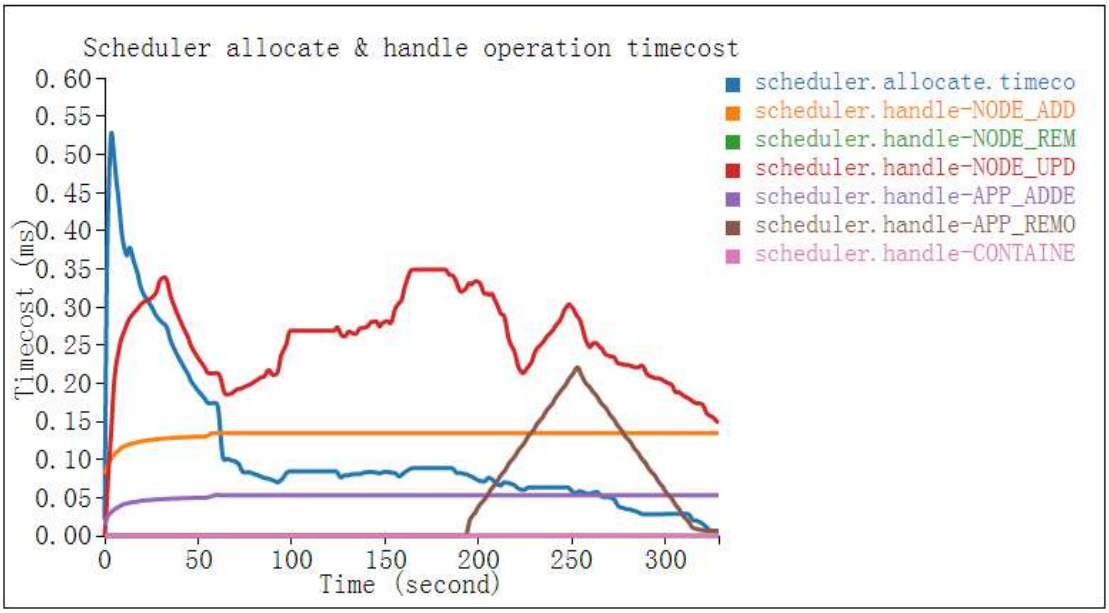

Cluster allocated & available memory
cluster.allocated.memory
cluster.available.memory
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
Memory (MiB)
0 100 200 300 400 500 600 700 800 900 1,000 1,100 1,200 1,300
Time (second)


Scheduler allocate & handle operation timecost
scheduler.allocate.timeco
scheduler.handle-NODE_ADD
scheduler.handle-NODE_REM
scheduler.handle-NODE_UPD
scheduler.handle-APP_ADDE
scheduler.handle-APP_REMO
scheduler.handle-CONTAINE
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Timecost (ms)
0 100 200 300 400 500 600 700 800 900 1,000 1,100 1,200 1,300
Time (second)

AFTER:


ResourceManager Event Processor
java.lang.Thread.run ()
org.apache.hadoop.yarn.server.resourcemanager.ResourceManager$SchedulerEventDispatcher$EventProcessor.run ()
org.apache.hadoop.yarn.sls.scheduler.ResourceSchedulerWrapper.handle ()
org.apache.hadoop.yarn.sls.scheduler.ResourceSchedulerWrapper.handle ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.handle ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.handle ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.nodeUpdate ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.attemptScheduling ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FSParentQueue.assignContainer ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FSLeafQueue.assignContainer ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FSParentQueue.assignContainer ()
java.util.Collections.sort ()
自用时间
自用时间
自用时间
org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.completedContainer ()
org.apache.hadoop.yarn.server.resourcemanager.scheduler.AbstractYarnScheduler.containerLaunchedOnNode ()
自用时间
自用时间
org.apache.hadoop.yarn.sls.scheduler.ResourceSchedulerWrapper.updateQueueWithNodeUpdate ()
自用时间
自用时间
java.util.concurrent.LinkedBlockingQueue.take ()
自用时间
自用时间
314,... (100%) 149,047 ms
314,... (100%) 149,047 ms
314,... (100%) 149,047 ms
177,... (56.4%) 144,985 ms
177,... (56.4%) 144,985 ms
163,... (52%) 131,325 ms
163,... (52%) 131,325 ms
163,... (52%) 131,325 ms
111,... (35.4%) 111,400 ms
111,... (35.4%) 111,400 ms
98,9... (31.4%) 98,935 ms
10,4... (3.3%) 10,444 ms
2,02... (0.6%) 2,020 ms
0.00... (0%) 0.000 ms
0.00... (0%) 0.000 ms
32,5... (10.3%) 0.000 ms
16,8... (5.3%) 16,832 ms
3,09... (1%) 3,092 ms
0.00... (0%) 0.000 ms
0.00... (0%) 0.000 ms
13,6... (4.3%) 13,660 ms
0.00... (0%) 0.000 ms
0.00... (0%) 0.000 ms
137,... (43.6%) 4,061 ms
0.00... (0%) 0.000 ms
0.00... (0%) 0.000 ms


Cluster allocated & available memory
cluster.allocated.memory
cluster.available.memory
Memory (MB)
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
0
50
100
150
200
250
300
Time (second)


Scheduler allocate & handle operation timecost
scheduler.allocate.timeco
scheduler.handle-NODE_ADD
scheduler.handle-NODE_REM
scheduler.handle-NODE_UPD
scheduler.handle-APP_ADDE
scheduler.handle-APP_REMO
scheduler.handle-CONTAINE
Timecost (ms)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
50
100
150
200
250
300
Time (second)

SUMMARY

indexes	before	after	improvement
Scheduling Throughput	200 containers/s	~2100containers/s	~10x
Resource utilization(when stable)	~20%	~100%	~5x
NODE_UPDATE duration	~6ms	~0.25ms	~24x
Time costs for all jobs	1300s	330s	~75%