YARN-5139 Parallel Scheduling Performance Test Report - V1

Wangda Tan

Background

YARN-5139 include changes to split scheduler logic into two pieces:

1) According to a given snapshot of scheduler state, make new allocation proposal
2) After allocation proposal is created, make decision to accept/reject it.

More details please refer to: YARN-5139 design doc.

After this change is done, #1 could be parallelized, this performance test report is majorly focused on how much performance gain we can get from parallel scheduling.

Test Environment

Use SLS (Scheduler Load Simulator) to run following tests:

- Mocked 20K nodes, each node has 128G memory, so in total the cluster manages 2.4+PB memory resources.
- Submitted 47K applications to the cluster, size of application varies, the cluster supports 4k-12k applications running in parallel.
- Resource of containers is ranged from 1G to 8G, subject to uniform distribution.
- Lifespan of containers is ranged from 5 sec to 4 mins, subject to uniform distribution
- Node heartbeat interval is 1 sec.
- Run the SLS test for 10 mins, and get average #containers allocated per second.

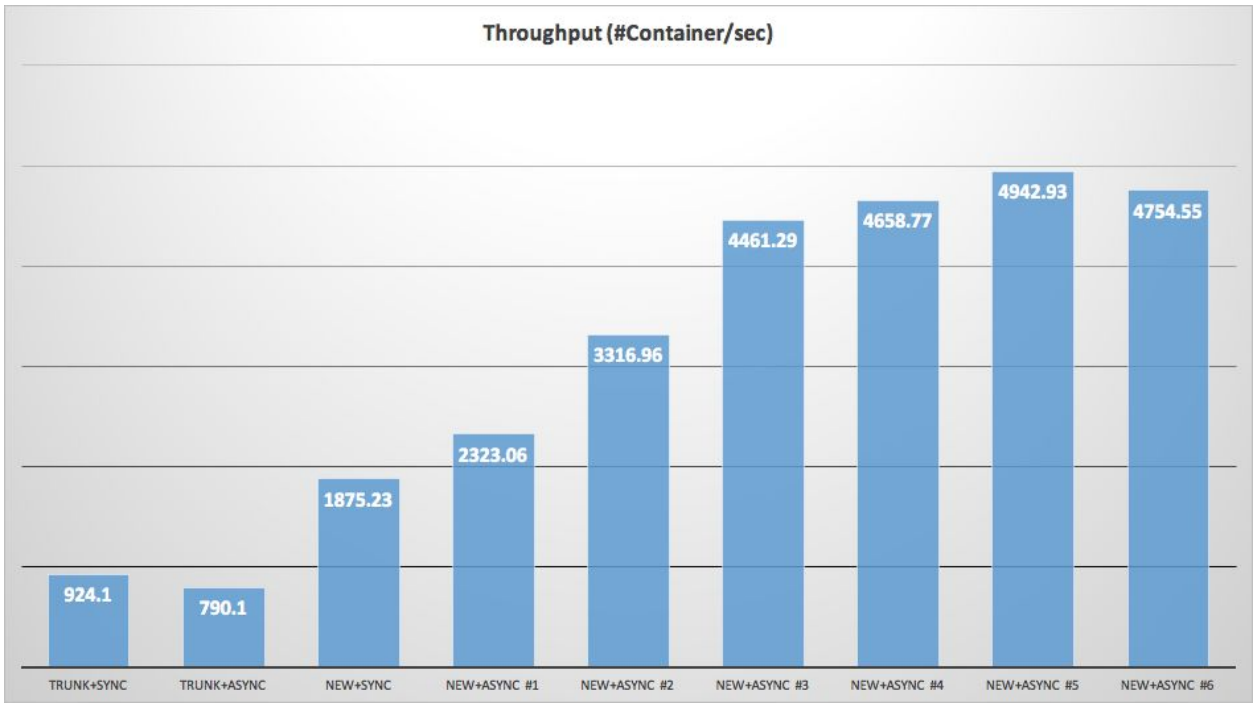
Test machine configuration:

- Centos 6.8
- 32 Virtual Cores
- 256G memory (16GB memory allocated to SLS process, which includes RM)

Test configuration (All use Capacity Scheduler)

- Conf 1: Latest trunk + synchronized scheduling
- Conf 2: Latest trunk + asynchronized scheduling
- Conf 3: With YARN-5139 + synchronized scheduling
- Conf 4-9: With YARN-5139 + async scheduling (#Allocation-Reader-Threads ranged from 1-6)

Test Result



Notes:

1) From the test report, you may noticed that YARN-5139 + synchronized scheduling runs much faster comparing to original synchronized scheduling in trunk. This performance gain comes from following improvements:
    a) Too many ResourcePBImpl object created along with scheduling, Instancing such -PBImpl object is very expensive, created a simple Resource object that has final fields of mem / cpu in some hot areas.
    b) Lock contention happens on ResourceUsage object's R/W lock. Instead of using R/W lock, added a AtomicResource object to support atomically increase/decrease/update resource.
    c) Optimized queue/user limit calculation: Originally queue and user limit calculation happens on EVERY application allocation. Now updated it to calculate queue limit once for every leaf queue, and calculate user limit once for every user inside leaf queue.
    d) Will file tickets of these improvements separately

Summary

According to the test report, parallel scheduling can significantly increase scheduler throughput.

Comparing to original async scheduling, we can get up to **6.25X** throughput
Comparing to original sync scheduling, we can get up to **5.34X** throughput

Future plan

- Doing more tests to make sure it doesn't break scheduler's functionalities. (Especially for correctness).
- Identify more hotspots and eliminate them.
- Get changes reviewed and committed