

MOPS: A Modified Priority Scheduler for Improving Performance in Hadoop Cluster

Abhishek Agrawal, Dr. Bina Ramamurthy

Department of Computer Science and Engineering,
University at Buffalo, The State University of New York



Objectives

- Increase the throughput of the high priority queues and ensure that throughput of low priority queues is not adversely affected.
- It should be possible to use any scheduling algorithm within any job pool.

Our Strategy

- MOPS- A modified priority scheduler which ensures that the total waiting time for a job pool is inversely proportional to its priority.
- Make the scheduler for each job pool an independent component so that any scheduling algorithm can be applied within any job pool.

Current Implementation[1]

- **Moab is currently the most popular batch scheduler** for clusters .
- Jobs are put into various pools depending upon user name, group, etc. Job pools are allotted resources according to their priority.
- Each pool has a hard limit and a soft limit.
- The soft limit is the guaranteed amount of resources that each pool will receive.
- If after satisfying the soft limits of each pool there are still some resources left over then they are allotted according to the hard limit .

Issues With Current Implementation

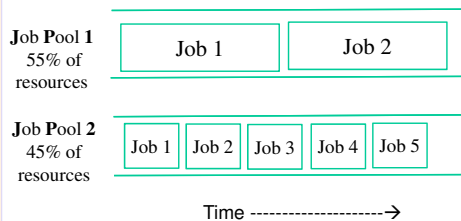


Figure 1: JP1 has higher priority than JP2 but the throughput of JP2 is much higher than JP1.

- The main problem is that after the scheduling algorithm has been applied, the job simply has to wait for its turn. Nothing can be done to reduce the waiting time of the job. As we can see from Fig 1, 3 jobs have finished in JP2 before job 2 in JP1 starts.
- Such a situation is very unfair for JP1. In spite of being at a higher priority level the throughput of JP1 is lesser than that of JP2.
- Any commercial model will charge more for queuing the jobs in JP1 as opposed to JP2. No customer would like to pay more and yet wait longer for his jobs to finish executing.

Our Solution: MOPS

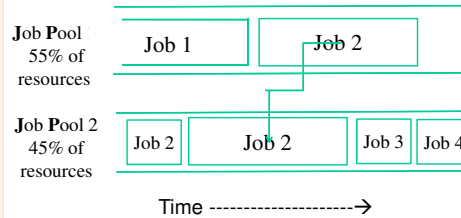


Figure 2

- Jobs are placed into job pools based on user name, UNIX group, etc. Resource allotment to job pools is according to the priority of the jobs being sent to the pool.
- Each pool has an upper priority level and a lower priority level.
- **Priority of the job is reduced till it is lesser than the lower priority level of the pool. A copy of the job is then added into the pool one level below. Only 1 instance of the job is executed. Priority Reduction of a job depends upon**

Time Passed After Job Submission

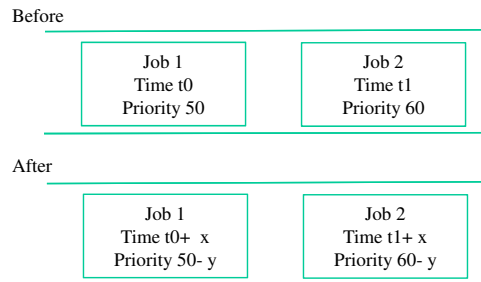


Figure 3: This shows that as the time passed after the job submission increases, the priority of the job decreases

Number of jobs completed in the lower priority pools

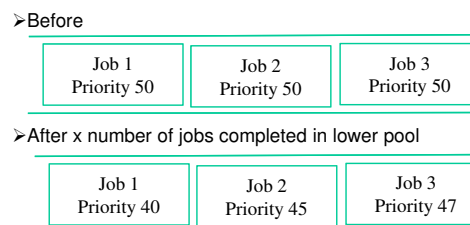


Figure 4: The modified priority ensures that all the high priority jobs do not go to the low priority queue at the same time. This prevents flooding of low priority queues

Results

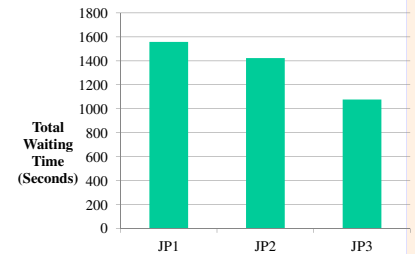


Figure 5

Job Pool v/s Total Waiting Time using Moab

Highest Priority pool has the maximum waiting time

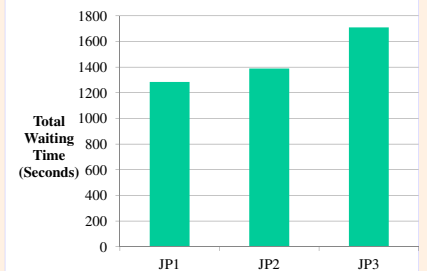


Figure 6

Job Pool v/s Total Waiting Time using MOPS

Highest Priority pool has the least waiting time

Expected Benefits

- Faster job completion will increase the usage of clusters.

Future Directions

- Adding "Backfilling" to our scheme
- Developing new algorithms for backfilling

Conclusion

- MOPS is a new attempt at job scheduling. It makes the waiting time of the job pools inversely proportional to priority.
- We do that by maintaining an instance of the job in multiple job pools at the appropriate time as determined by MOPS
- Such a system is necessary for wide spread use and better utilization of clusters.

References

- [1] Jackson, David and Snell, Quinn and Clement, Mark (2001) *Core algorithms of the maui scheduler*. In Proceedings of the 7th Workshop on Job Scheduling Strategies for Parallel Processing