Paper Critique

Title: Lottery Scheduling: Flexible Proportional-Share Resource Management
Name: Yash Aggarwal | yagga004 | 862333037

Summary

The Paper proposes lottery scheduling, a new scheduling algorithm that is randomized but fair and can prevent starvation. It also provides efficient and flexible control of resources, can escalate priority in case required, and stays fair in resource allocation. The key idea is to assign all the jobs a ticket or multiple tickets depending on the job and then randomly pick out the ticket, and the job with the ticket is given resource access till the next CPU cycle. The value of a ticket can be increased or decreased by inflation or suspension in case it is required. The chance for a job to run is proportional to the number of tickets. The authors implemented the algorithm in mac 3.0 and compared lottery scheduling with various other scheduling algorithms, and it outperforms other algorithms regarding response time and resource utilization.

Strength:

- The algorithm uses the concept of economics well and has influenced many scheduling algorithms.
- The algorithm is fair and easy to implement with less overhead.
- The algorithm can dynamically be scaled and can be modular if required.
- It provides better control over job execution and resource allocation.

Weaknesses:

- Lottery scheduling was created for CPU-intensive jobs, and little focus was given to I/O jobs. For example, if a job has to wait for an I/O task, it is not in the pool, and when it receives the Input, it then has to compete again for CPU access. If multiple such jobs compete with one another, they would first compete for I/O, then for CPU, and while competing for one, they are not considered for another.

Comments:

- When talking about scheduling algorithms, we should also consider what all algorithms are available. For resource-sharing algorithms, round-robin seems reasonable as it is fair and keeps resources busy. However, it is too simple, and there is no way to escalate priority. We could add some prioritization or use different priority scheduling algorithms, but it seems like that would increase the overhead. Lottery scheduling seems fair and considerate, can have priority escalation, and still have less overhead than other algorithms.