Paper summary

Scheduling can be a massive overhead in a multithreaded environment. Conventional schedulers use priority-based scheduling and fair share scheduling, but such systems are inefficient for interactive systems such as databases and multimedia. Lottery scheduling helps to randomized resource allocation by assigning the resources based on the number of tickets. The priority of the process decides the number of tickets. The resources are given based on the winning lottery, which is selected randomly. Since the expected allocation of resources is proportional to the number of the client's tickets, it can be considered priority scheduling in a non-deterministic fashion.

Strengths

1. Aside from the CPU, lottery systems can be used in memories and networks.

2. The novel concept of randomness.

Weaknesses

1. The high-priority task "currently important" may not get allocation at the required time due to the randomness, especially in real-time systems.

2. Only comparison with priority-based scheduling. Some benchmark comparisons with other scheduling systems would give a better idea of the performance.

3. The performance gain shown in the microbenchmarks does not favor the adoption of this system in the real world.

4. Priority Scheduling can avoid starvation by aging—no justification against such a scenario.

Comments for author

1. Since the pseudo-random numbers are not truly random, will the resource distribution be genuinely fair? Is there an occurrence of starvation in the system with the current implementation? If hardware random number generators (HRNG) are used instead of pseudo randoms, will there be a performance improvement and a lesser probability of starvation?

2. Why is tree base data structure not used for small values of n lotteries and used only for more significant values?