Peer review - Lottery scheduling: flexible proportional-share resource management

María Fernanda Mora-Alba - MSc in Computer Science, ITAM

I. SUMMARY

Flexible response control is desirable in systems that serve requests of varying importance, such as databases, media-based applications and networks. *Lottery scheduling*, a probabilistic model, is the first resource allocation scheme that provides this type of control together with modular resource management. To allocate resource rights, tickets and currency abstractions are introduced. A prototype is successfully implemented for the Mach 3.0 microkernel with an overhead comparable to that of the standard Mach time-sharing policy. It can be generalized to other resources such as I/O bandwidth, memory and access to locks.

II. PROBLEM DEFINITION

Priority schedulers (absolute control), fair-share schedulers (dynamically manipulate priorities) and microeconomic schedulers (actions and bidding for resources) suffer from the following disadvantages in a greater or lesser degree:

- **Highly tuning-dependent:** depend on parameter tuning and its performance can be severely affected if the assumptions are wrong. The tuning does not consider several types of computations occurring simultaneously.
- **Non-responsive:** assumptions and overheads imply that they only serve for long-running applications where the approximate workload is known in advance. Not suitable for interactive systems which require rapid, flexible and dynamic control (databases or media-based applications).
- **Non-modular:** rely onto a simple notion of priority that does not provide encapsulation priorities required for the engineering of large end-to-end software systems.

Lottery scheduling is proposed to overcome these shortcomings.

III. CONTRIBUTION

A. Model

Resource rights are represented by *lottery tickets*, which are assigned dynamically to clients in proportion to the contention for that resource. Allocation is determined by holding a lottery and the resource is granted to the winning ticket. The number of winning lotteries for each client is modelled by a binomial distribution and the number of tries till the first win by a geometric one. Overall, this means we expect a "fair" scheduling on the long-run, no starvation and responsiveness.

Ticket abstraction can be used to insulate the resource management policies in independent modules using: *ticket transfers* (when a client is waiting a response), *ticket inflation* (to increase the number of tickets), *ticket currencies* (flexibly naming, sharing, and protecting resource rights) and *compensation tickets* (to compensate a client for not consuming its expected share in a quantum).

B. Implementation

LS was implemented in Mach 3.0 microkernel. Lotteries should be run very frequently to guarantee high responsiveness. The unoptimized implementation performs $O(n)$ comparisons to find the winning ticket. A tree-based approach using partial ticket sums reduces the number of operations to $O(log(n))$.

C. Experiments

LS's ability to flexibly, responsively, and efficiently control the relative execution rates of computations was successfully tested (fairness, accuracy and flexible control is achieved) in the compute-bound Dhrystone benchmark, in a Monte-Carlo numerical integration program, a multithreaded client-server application and competing MPEG video viewers.

IV. CRITIC

LS also relies in parameter tuning (in a lesser degree, but it does). How to determine individual number of tickets? This still implies assumptions on what process are more important.

LS assumes statistical independence, which is strong and implausible. Workloads do have memory and interdependence.

LS relies in long-term stability properties but we expect high short-term variance. Compensation tickets create a system imbalance: what happens to clients receiving more than its fare share? they are not removed any ticket fraction. What if these clients are many? A deterministic version can be used in the short term (see Stride scheduling).

LS seems not that fair: mutex and multimedia apps come out differently than expected. Hypothesis testing is recommended to see if this differences are statistically significant.

LS assumes p can be recalculated in each allocation decision, but very frequent recalculation will cause overhead. It will be simpler to assume that p is a random variable itself.

LS can be cheated to be unfair by quitting early to get compensation tickets and then run full-time.

Despite of exchange rate and base currency mechanisms to contain inflation it is still possible that monopolies within single groups exist as there is no cost to create more tickets.

REFERENCES

[1] Waldspurger, Carl A., and William E. Weihl. "Lottery scheduling: Flexible proportional-share resource management". Proceedings of the 1st USENIX conference on OS Design. USENIX Association, 1994.