Lottery Scheduling

Robert Grimm
New York University
Altogether Now: The Three Questions

- What is the problem?
- What is new or different?
- What are the contributions and limitations?
Motivation

- Scheduling of scarce computer resources (e.g., CPU)
  - Has major impact on throughput and response time
  - Should be fair (scientific applications)
  - But also needs to adjust rapidly (interactive applications)

- Priority-based schemes
  - Do not provide encapsulation, modularity
  - Rely on ad-hoc assignment of priorities
  - Are poorly understood
Enter Lottery Scheduling

- Provides a randomized mechanism
  - Not suitable for real-time systems
- Provides control over relative execution rates
- Can be implemented efficiently
- Supports modular resource management
The Basic Ingredients

- Tickets
  - Abstract, relative, and uniform resource rights

- Lotteries
  - Probabilistically fair selection of next resource holder
  - Throughput proportional to client’s ticket allocation
    - Binomial distribution, accuracy improves with $\sqrt{n}$
  - Average response time inversely proportional to client’s ticket allocation
Modular Resource Management

- **Ticket transfers**
  - Useful for RPC-based systems
  - Avoid priority inversion problem

- **Ticket inflation**
  - Provides alternative to transfers (no communication!)
  - Needs to be avoided/contained in general

- **Ticket currencies**
  - Support flexible naming, sharing, protecting of access rights

- **Compensation tickets**
  - Make up for underutilization
Implementation

- Integrated into Mach 3.0
- Supports ticket transfers, inflation, currencies, compensation tickets
- Relies on fast pseudo-random number generator
  - Generator of choice is subject to discussion
    - Communications of the ACM, 36(7):105-110, July 1993
- Selects winning thread from list of tickets: $O(n)$
  - Ordered by relative amount
  - Possible optimization: Tree-based $O(\log n)$
Ticket Currencies

- Tickets only active when thread is ready: Active amount/currency
- (De)activation propagates up/down the tree on change to/from zero
- Currency’s value equals to sum of backing tickets
  - Ticket’s value equals to fraction of currency’s value
Experimental Evaluation
Fairness

- Two tasks executing the Dhrystone benchmark
- Varying ticket allocations

Observed ratio over 60 seconds

8 sec time windows, 2:1 alloc
Flexible Control

- Three Monte-Carlo tasks
  - Dynamic ticket inflation proportional to \((relative\ error)^2\)
Client-Server Computation

- Three clients querying text-search server
  - 8:3:1 ticket allocation for clients
  - *None* for server
Multimedia Applications

- Three video viewers
  - 3:2:1 initial allocation, changed to 3:1:2 at arrow
  - Real ratios: 1.92:1.50:1 before, 1.92:1:1.53 after → Why?
Load Insulation

- 5 tasks executing Dhrystone
  - Two currencies A and B
    - Funded equally
  - Task group A
    - 1:2
  - Task group B
    - 1:2, then 1:2:3
Lock Scheduling

- Lottery-scheduled mutex has
  - Mutex currency
  - Inheritance ticket
- Waiting threads fund mutex currency
- When done, mutex holder conducts lottery to determine next holder
  - Passes on inheritance ticket
- \( n = 8 \) threads competing for single mutex
  - Each thread
    - Acquires mutex
    - Holds it for \( h = 50 \) milliseconds
    - Releases mutex
    - Computes for \( t = 50 \) milliseconds
  - Threads divided into two groups
    - 2:1 ticket allocation
    - 1:2.11 waiting times
    - 1.80:1 mutex acquisition rates
What Do You Think?