Lottery Scheduling

Robert Grimm
New York University
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
  - Rely on ad-hoc assignment of priorities
  - Are poorly understood
  - Do not provide encapsulation, modularity
Enter Lottery Scheduling

- Provides a randomized mechanism
  - Not suitable for (hard) 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
Fun with Lottery Tickets

- 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, and protecting of resource 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
  * [Park & Miller CACM '88] is likely not the best choice
* Selects winning thread from list of tickets in \( O(n) \) time
  * Ordered by relative amount
  * Possible optimization: Tree-based \( \rightarrow O(\log n) \) time
Ticket Currencies

- Tickets only active when thread is ready
- Deactivation propagates up the tree if currency's active amount becomes 0
- Down tree on activation
- Currency's value equals sum of backing tickets
- Ticket's value equals to fraction of currency's value
Evaluation
Two tasks executing the Dhrystone benchmark

- Varying ticket allocations

![Graph showing observed iteration ratio vs allocated ratio](image1)

![Graph showing average iterations per second over time](image2)

- Observed ratio over 60s
- 8s windows, 2:1 alloc
**Flexible Control**

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

---

**Figure 5: Fairness Over Time.** Two tasks were scheduled on different processors. The string used for this experiment was `lottery`, which incidentally occurs a total of 8 times in Shakespeare's plays.
Three clients querying text-search server
- 8:3:1 ticket allocation for clients
- None for server

Figure 7: Query Processing Rates. Three clients with...
Multimedia Applications

* Three video viewers
  * 3:2:1 initial allocation, changed to 3:1:2 at arrow
  * Real ratios: 1.92:1.50:1 and 1.92:1:1.53 — why?

![Graph](image)
• 5 times 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
More Lock Scheduling

- \( n = 8 \) threads competing for single mutex
  - Each thread
    - Acquires mutex
    - Holds it for \( h = 50 \) ms
    - Releases mutex
    - Computes for \( t = 50 \) ms
  - Threads divided into two groups
    - 2:1 ticket allocation
    - 1:2.11 waiting times
    - 1.80:1 mutex acquisition rates
What Do You Think?