Lottery Scheduling

Flexible Proportional-Share Resource Management

(OSDI 1994)

Presented by Brevan Chun

Scheduling

- Scheduling computations in multithreaded systems is challenging.
- The consumption of shared resources must be regulated.
- Existing schemes are:
  - ↳ Not responsive, flexible
  - ↳ Not modular, no encapsulation
  - ↳ Poorly understood service rates (decay-usage scheme)
  - ↳ Inefficient (fair share scheme)

Lottery Scheduling

- *“a novel randomized mechanism that provides responsive control over the relative execution rates of computations”*

+ Responsiveness
+ Modularity
+ Control
+ Efficiency

Lotteries

- Generate random numbers
- Search a data structure for the client with the winning number/ticket
  - ↳ List → O(n)
  - ↳ Tree → O(lg n)
- Allocation of resources is proportional to the number of tickets a client has.
  - ↳ e.g., a client with 75 of 100 tickets is entitled to 75% of the resource usage

Fairness Through Probability

$$p = t/T$$ (single win probability)

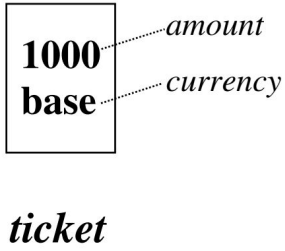
$$E[w] = np$$ (expected wins)

$$E[n] = 1/p$$ (expected lotteries)

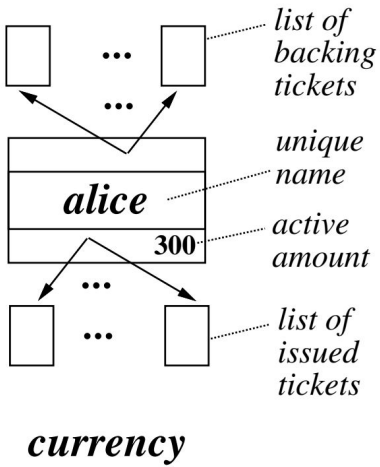
- These probabilities are well understood
- Any client with a ticket will eventually win.

Lottery Tickets

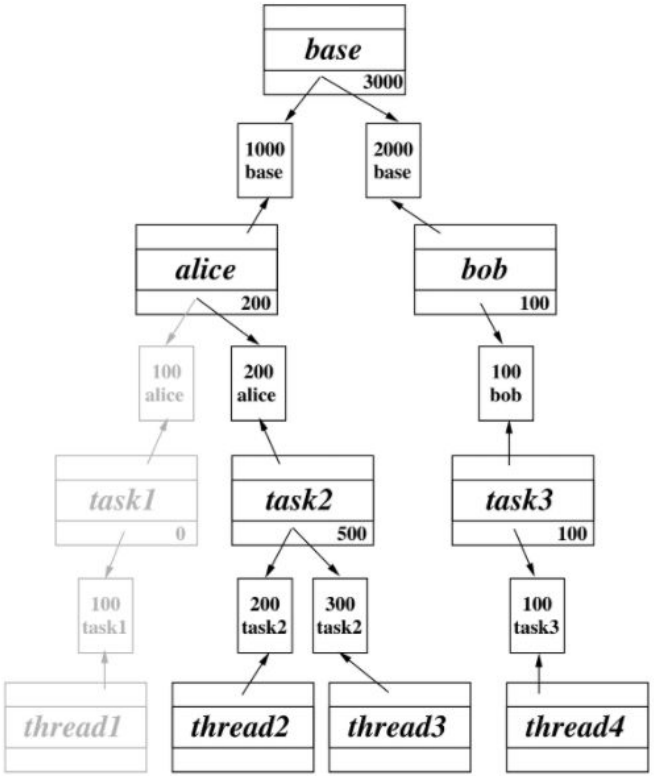
1. Ticket transfers
   - explicit transfer of tickets from a client to another
2. Ticket inflation
   - escalate resource rights by creating more lottery tickets
3. Ticket currencies
   - express resource rights in local groups
4. Compensation tickets
   - a client is given more tickets if it uses less than it is allocated



Currencies



(currency object)



(currency graph)

Implementation

- DECstation 5000 Model 125
  - ↳ 25MHz CPU
  - ↳ Modified Mach 3.0 microkernel
- 100ms lotteries (10 per second)
- Operations to create, destroy, fund, and compute values of currency and tickets
- Ticket transfers, currencies, compensation
- User interface
  - ↳ Currency and ticket manipulation via command-line interface

DECstation 5000 computer

Tests

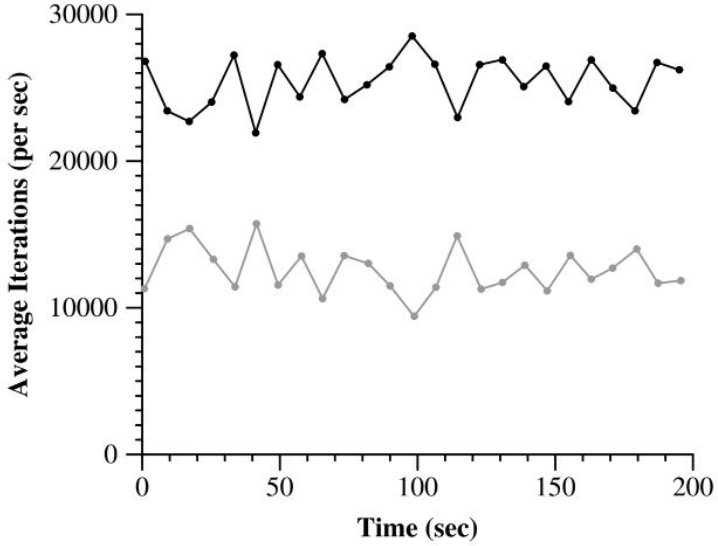


Figure 5: **Fairness Over Time.** Two tasks executing the Dhrystone benchmark with a 2 : 1 ticket allocation. Averaged over the entire run, the two tasks executed 25378 and 12619 iterations/sec., for an actual ratio of 2.01 : 1.

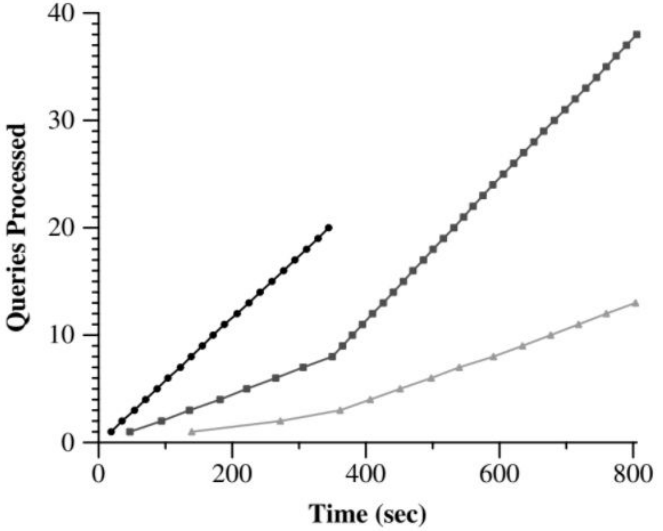


Figure 7: **Query Processing Rates.** Three clients with an 8 : 3 : 1 ticket allocation compete for service from a multithreaded database server. The observed throughput and response time ratios closely match this allocation.

(Client-Server)

More Tests

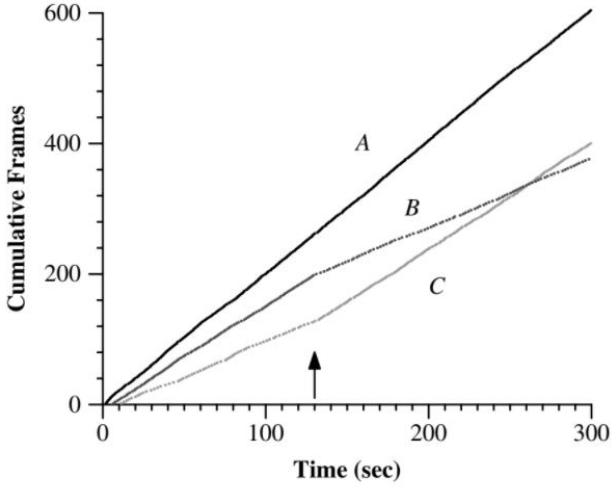


Figure 8: **Controlling Video Rates.** Three MPEG viewers are given an initial $A:B:C = 3:2:1$ allocation, which is changed to $3:1:2$ at the time indicated by the arrow. The total number of frames displayed is plotted for each viewer. The actual frame rate ratios were $1.92:1.50:1$ and $1.92:1:1.53$, respectively, due to distortions caused by the X server.

(Multimedia App)

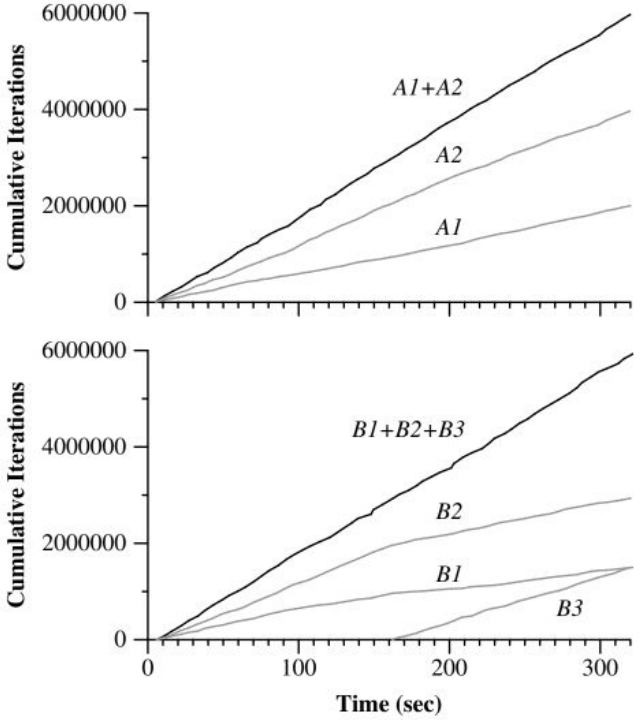


Figure 9: **Currencies Insulate Loads.** Currencies A and B are identically funded. Tasks $A1$ and $A2$ are respectively allocated tickets worth $100.A$ and $200.A$. Tasks $B1$ and $B2$ are respectively allocated tickets worth $100.B$ and $200.B$. Halfway through the experiment, task $B3$ is started with an allocation of $300.B$. The resulting inflation is locally contained within currency B, and affects neither the progress of tasks in currency A, nor the aggregate $A:B$ progress ratio.

Evaluation

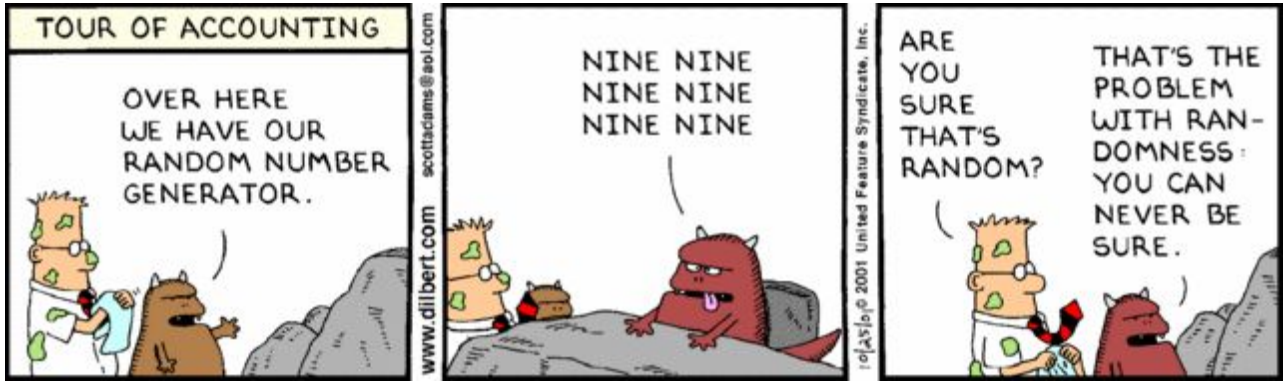
- Fairness ✓
- Responsiveness ✓
- Modularity ✓
- Control ✓
- Efficiency ✓
  - ↳ Random numbers generated with 10 instructions
  - ↳ A tree-based lottery is O(lg n) to traverse
  - ↳ Prototype implementation is **NOT** optimized
  - ↳ On par with standard Mach 3.0

Other Resources

1. I/O bandwidth
   - Pretty much the same
2. Synchronized resources
   - Blocked threads give tickets to thread with the mutex
3. Space-shared resources
   - *Inverse lottery*: a loser is chosen to relinquish a unit of resource
4. Multiple resources
   - Implement a *manager* thread?

Discussion

1. Can compensation tickets be abused?
2. How should tickets be assigned?
3. Is pure probability a good design?



Presented in CSE 231, UCSC Fall 2021

Paper:
https://www.usenix.org/legacy/publications/library/proceedings/osdi/full_papers/waldspurger.pdf