Lottery Scheduling for Resource Management

Lottery Scheduling: Flexible Proportinal-Share Resource Management [SOSP'94]

Carl Waldspurger and William Wheil
MIT Laboratory for Computer Science

Presented by Stefan Birrer

CS-443 Advanced Operating Systems – Spring 2005
Motivation

• Scheduling computations
  – Complex & challenging in multithreaded systems
  – Rapid, dynamic control

• Existing priority-based schemes
  – Poorly understood

• Existing fair share/microeconomic schedulers
  – Assumptions
  – Overheads

Need for more an efficient scheduling algorithm
Lottery Scheduling

- Randomized mechanism
- Proportional-share resource management
- Flexible control over relative execution rates
- Generalizable
  - I/O bandwidth
  - Memory
  - Access to locks
Lottery Scheduling

- Lottery tickets
- Allocation is determined by holding a *lottery*
- Resource is granted to *winning ticket*
- Effective allocation is proportional to number of tickets held
- Probabilistic fair
- No starvation
  - Every client with tickets eventually wins a lottery
- Tickets may be transferred
  - Solves priority inversion problem
Lottery Scheduling

• Ticket inflations
  – Trusted environments
  – Boost performance of client by generating more tickets
• Ticket currency
  – Across logical boundaries
• Compensation tickets
  – If client consume only a fraction of the allocated resource quantum
Implementation

- Mach 3.0
- 25 Mhz MIPS-based DECStation 5000/125
- Scheduling quantum - 100 milliseconds
- Pseudo-random number generator
  - Park-Miller algorithm
  - 10 RISC instructions
- Lottery
  - “move-to-front” heuristic
  - Tree of partial ticket sums
Implementation

- Kernel Interface

- Ticket Currency
Evaluation

- Quantify
  - Flexibility
  - Responsivness
  - Control efficiency
- Workload
  - Compute-bound benchmark
  - Monte-carlo numerical integration
  - Multithreaded client-server application
  - Competing MPEG video viewers
Fairness

Computation rate is accurately controlled
Fairness

Good fairness with 100ms quantum
Flexible Control

Adjust ticket value to speed up new runs
Flexible Control

Catch up with older processes
Client-Server Application

- Client temporarily transfer tickets to server
Mutlimedia Application

- Change allocation ratio
Load Insulation

- Two untrusted domains
Synchronization Resources

- Mutex

![Diagram of Mutex and Waiting Threads]

- Chart showing mutex acquisitions over waiting time for Group A and Group B.
Conclusions

- Rapid, dynamic resource allocation
- Lottery scheduling
  - Proportional share
  - Probabilistically fair
  - Fast
  - Transferable
  - Adjustable
  - General purpose