Lecture 8: Lottery Scheduling
Class Plan

• Lottery scheduling:
  – Motivation
  – Solution proposed

• Group work and answers on:
  – Implementation issues: challenges? Solutions? Surprises?
  – Experiments: objective; experimental setup (methodology); results; interpretation of results.
Figure 1: Example Lottery. Five clients compete in a list-based lottery with a total of 20 tickets. The fifteenth ticket is randomly selected, and the client list is searched for the winner. A running ticket sum is accumulated until the winning ticket value is reached. In this example, the third client is the winner.
Figure 2: Kernel Objects. A ticket object contains an amount denominated in some currency. A currency object contains a name, a list of tickets that back the currency, a list of all tickets issued in the currency, and an active amount sum for all issued tickets.
Implementation: Ticket Currency

Figure 3: Example Currency Graph. Two users compete for computing resources. Alice is executing two tasks: task1 is currently inactive, and task2 has two runnable threads. Bob is executing one single-threaded task, task3. The current values in base units for the runnable threads are thread2 = 400, thread3 = 600, and thread4 = 2000. In general, currencies can also be used for groups of users or applications, and currency relationships may form an acyclic graph instead of a strict hierarchy.
Experiments: Fairness

Figure 4: Relative Rate Accuracy. For each allocated ratio, the observed ratio is plotted for each of three 60 second runs. The gray line indicates the ideal where the two ratios are identical.
Experiments: Fairness Over Time

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.
Experiments: Flexible Control

Figure 6: Monte-Carlo Execution Rates. Three identical Monte-Carlo integrations are started two minutes apart. Each task periodically sets its ticket value to be proportional to the square of its relative error, resulting in the convergent behavior. The “bumps” in the curves mirror the decreasing slopes of new tasks that quickly reduce their error.
Experiments: Client-Server Computations

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.
Experiments: Flexible Control 2

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.
Experiments: Load Insulation

![Graph showing cumulative iterations over time for different tasks and currencies]

Figure 9: **Currencies Insulate Loads.** Currencies \(A\) and \(B\) are identically funded. Tasks \(A_1\) and \(A_2\) are respectively allocated tickets worth 100.\(A\) and 200.\(A\). Tasks \(B_1\) and \(B_2\) are respectively allocated tickets worth 100.\(B\) and 200.\(B\). Halfway through the experiment, task \(B_3\) 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.
Applicable not only to CPU sharing

Figure 10: Lock Funding. Threads $t_3$, $t_7$, and $t_8$ are waiting to acquire a lottery-scheduled lock, and have transferred their funding to the lock currency. Thread $t_2$ currently holds the lock, and inherits the aggregate waiter funding through the backing ticket denominated in the lock currency. Instead of showing the backing tickets associated with each thread, shading is used to indicate relative funding levels.

Figure 11: Mutex Waiting Times. Eight threads compete to acquire a lottery-scheduled mutex. The threads are divided into two groups ($A$, $B$) of four threads each, with the ticket allocation $A:B = 2:1$. For each histogram, the solid line indicates the mean ($\mu$); the dashed lines indicate one standard deviation about the mean ($\mu \pm \sigma$). The ratio of average waiting times is $A:B = 1:2.11$; the mutex acquisition ratio is 1.80:1.