Lottery Scheduling: Flexible Proportional-Share Resource Management

Carl A. Waldspurger  William E. Weihl
MIT Laboratory for Computer Science

OSDI ’94
Contents

• Background
• Overview
• Design & Implementation
• Evaluation
• Contribution
• Flaws
Background (1)

- Scheduler needs
  - Flexibility
  - High responsiveness
  - Modular
  - Robust
  - Fine grained
  - Low-overhead
  - Conceptually-simple
Background (2)

• Priority based scheduling is often ad-hoc[1]
  • Pretty good for considering all the factors
• Several problems of priority scheduling
  • Absolute priority schemes
  • Fairness
  • Too many parameters make scheduler complicated
• Proportional-share scheduling in network packet scheduling gets famous

Approach to Solve Fairness\textsuperscript{[1,2]}

- Introduce dynamically controlled priority
- Solve absolute priority problem
- Limitations
  - Large overhead
  - Response time is bad

\textsuperscript{[1]} G. J. Henry. “The Fair Share Scheduler”, AT&T Bell Lab, 1984
Proportional-Share Scheduling

- At that time, main issue in packet scheduling in network is proportional-share scheduling
  - GPS is ideal model
    - Importance of fair queueing
  - Problem of WFQ\[1\]
    - Monitoring overhead was high

Overview

• Lottery scheduling
  • Randomized resource allocation mechanism
  • Proportional-share resource management

• Guarantee probabilistic fairness
  • Starvation-free
  • Low overhead than others

• Simple and elegant with not that much overhead
Design: Lottery

- In each round, one winner is picked
- Thread with winning ticket gets CPU resource
- Several implementations can exist
  - List based structure
  - Tree based structure
Design: Lottery Tickets

- Each thread receives variable numbers of lottery tickets
- Amount
  - The number of tickets that received
- Currency
  - Who published lottery tickets?
Design: Ticket Currency

- Looks like local lottery at each level
- Makes trusty boundaries
- Global load balancing
Design: Balancing Issue (1)

• Ticket transfers
  • One thread can send a bunch of tickets to another
  • Useful in interactive systems
  • Eliminate priority inversion problem

• Ticket inflation & deflation
  • If more tickets are added, the value of one ticket decreases
  • With trusting threads, it can be the key to dynamic balancing between threads
Design: Balancing Issue (2)

- **Ticket compensation**
  - Threads which consumes a fraction $f$ of time quantum receives $1/f$ times scheduling until they get lottery

- **CPU Bound Thread**
  - Use whole time quantum
  - Tickets : 400

- **I/O Bound Thread**
  - Use $1/5$ time quantum
  - $f = 1/5$
  - Tickets : 400 $\rightarrow$ 2000 ($*1/f$)
Experiment Environment

- Mach 3.0 microkernel (MK82)
- 25MHz MIPS-based DEC station 5000/125
- Apps
  - Dhrystone benchmark
  - Monte-Carlo integration
  - Multithreaded client-server
  - MPEG Video
Evaluation

Fairness Over Time
- 2 : 1 ticket allocation for 2 threads

Controlling Video Rates
- 3 video threads with different ticket
- At arrow point, change ticket ratio
Contributions

• Simple, flexible, but strong scheduler
  • Solve priority inversion and starvation
  • Fairness without much overhead
  • Dynamically flexible scheduling

• Well support for modular management
  • Can adjust resource allocation without explicit communication by currencies, ticket inflation and deflation
Flaws

• Bad response time
  • Do not provide responsiveness for interactive systems

• Non-deterministic
  • Unsuitable when the programmers may control better

• Probabilistic can not guarantee when the universe of ticket is small or too big

• Is it really ‘random’ number generator?
  • If skewed, it’s critical to entire system
After Lottery Scheduling (1)

- Stride scheduling\[1\]
  - Select the smallest value
  - Pass is advanced by its stride value
  - Ties are broken arbitrarily

\[1\] C. A. Waldspurger, “Lottery and stride scheduling”, MIT Lab of Computer Science, 1995
After Lottery Scheduling (2)

• Stride scheduling\textsuperscript{[1]}
  
  • Task $\tau_1$: tickets = 3, stride = 2
  
  • Task $\tau_2$: tickets = 2, stride = 3
  
  • Task $\tau_3$: tickets = 1, stride = 6

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$ pass value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$ pass value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$ pass value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\textsuperscript{[1]} C. A. Waldspurger, “Lottery and stride scheduling”, MIT Lab of Computer Science, 1995

\textsuperscript{[2]} Source: “Intro to CPU Scheduling” Provided by Minsoo Ryu, Hanyang Univ.
After Lottery Scheduling (2)

- Stride scheduling\[1\]
  - Task $\tau_1$ : tickets = 3, stride = 2
  - Task $\tau_2$ : tickets = 2, stride = 3
  - Task $\tau_3$ : tickets = 1, stride = 6

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$ pass value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\tau_2$ pass value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$\tau_3$ pass value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

After Lottery Scheduling (2)

- Stride scheduling[1]
  - Task $\tau_1$: tickets = 3, stride = 2
  - Task $\tau_2$: tickets = 2, stride = 3
  - Task $\tau_3$: tickets = 1, stride = 6

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$ pass value</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$ pass value</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$ pass value</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After Lottery Scheduling (2)

• Stride scheduling\textsuperscript{[1]}
  • Task $\tau_1$: tickets = 3, stride = 2
  • Task $\tau_2$: tickets = 2, stride = 3
  • Task $\tau_3$: tickets = 1, stride = 6

\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
$\tau_1$ pass value & \cellcolor{gray!30}2 & 4 & & & & & & \\
\hline
$\tau_2$ pass value & \cellcolor{gray!30}3 & 3 & & & & & & \\
\hline
$\tau_3$ pass value & 6 & 6 & & & & & & \\
\hline
\end{tabular}

\textsuperscript{[1]} C. A. Waldspurger, “Lottery and stride scheduling”, MIT Lab of Computer Science, 1995
\textsuperscript{[2]} Source: “Intro to CPU Scheduling” Provided by Minsoo Ryu, Hanyang Univ.
After Lottery Scheduling (2)

• Stride scheduling[1]
  • Task $\tau_1$ : tickets = 3, stride = 2
  • Task $\tau_2$ : tickets = 2, stride = 3
  • Task $\tau_3$ : tickets = 1, stride = 6

<table>
<thead>
<tr>
<th>Task</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$ pass value</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$ pass value</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$ pass value</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After Lottery Scheduling (2)

- Stride scheduling[1]
  - Task $\tau_1$: tickets = 3, stride = 2
  - Task $\tau_2$: tickets = 2, stride = 3
  - Task $\tau_3$: tickets = 1, stride = 6

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$ pass value</td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$ pass value</td>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$ pass value</td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After Lottery Scheduling (2)

- Stride scheduling\(^1\)
  - Task \(\tau_1\) : tickets = 3, stride = 2
  - Task \(\tau_2\) : tickets = 2, stride = 3
  - Task \(\tau_3\) : tickets = 1, stride = 6

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1) pass value</td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\tau_2) pass value</td>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\tau_3) pass value</td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) C. A. Waldspurger, “Lottery and stride scheduling”, MIT Lab of Computer Science, 1995

\(^2\) Source : “Intro to CPU Scheduling” Provided by Minsoo Ryu, Hanyang Univ.
After Lottery Scheduling (2)

• Stride scheduling[1]
  • Task $\tau_1$: tickets = 3, stride = 2
  • Task $\tau_2$: tickets = 2, stride = 3
  • Task $\tau_3$: tickets = 1, stride = 6

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$ pass value</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_2$ pass value</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau_3$ pass value</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After Lottery Scheduling (2)

• Stride scheduling[1]
  • Task $\tau_1$: tickets = 3, stride = 2
  • Task $\tau_2$: tickets = 2, stride = 3
  • Task $\tau_3$: tickets = 1, stride = 6

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$ pass value</td>
<td></td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>$\tau_2$ pass value</td>
<td></td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>$\tau_3$ pass value</td>
<td></td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>12</td>
</tr>
</tbody>
</table>

Question?