A Problem of Time vs. Density Tradeoff in Multicore Fluid Scheduling

Kang-Wook Kim, Jeongyoon Eo, and Chang-Gun Lee
Seoul National University, Korea

2017. 6. 27
Fluid Scheduling

- Based on P-Fair (Proportionate Fairness)
- Schedule tasks proportional to their density

\[ \text{density} = \frac{\text{WCET}}{\text{deadline}} \]

(WCET = Worst Case Execution Time)
Schedulability of Fluid Schedule

- Tasks are **schedulable**, if **sum of densities** of all active tasks is **smaller or equal to the number of processors** at any time instant.

**Time-Density Curves**

- Tasks
  - WCET = 6
  - density = 0.6
  - deadline = 10

- Tasks
  - WCET = 3
  - density = 0.3
  - deadline = 10

**System Density**

- density = 0.9
- time

**Task Density**

- density = 0.6
- time

**Task Density**

- density = 0.3
- time
Research Goal

Maximize schedulability of real-time multicore fluid scheduling

Minimize system peak density of fluid scheduling

by controlling

1) parallelization option (number of threads)
2) artificial deadline (shorter than original deadline)
3) artificial period (shorter than original period)
4) task offset
Open Problem 1: Time vs. Density Tradeoff

Artificial deadline control

1 thread

Parallelization option control

2 threads

Time-Density Curve

parallelization option ↑
Open Problem 1: Time vs. Density Tradeoff

Artificial deadline control

Parallelization option control

Time-Density Curve

1 thread

Parallelization option ↑

3 threads

1.2

0.7

0.4

0.3
Open Problem 1: Time vs. Density Tradeoff

Artificial deadline and parallelization option control

1 thread

Time-Density Curve

3 threads

Time-Density Curve
Open Problem 2: System-wide Tradeoff?

artificial deadline and parallelization option
+ artificial period and task offset control

System peak density
Problem Formulation (1 / 2)

System with $M$ homogeneous CPU cores, $N$ parallelizable periodic tasks

$$\tau_i = (P_i, D_i, C_i(O_i))$$

$P_i$: original period
$D_i$: original deadline
$O_i$: parallelization option
$e_i^k(O_i)$: WCET of $k$-th thread when parallelized into $O_i$ threads
$C_i(O_i)$: total computation amount

<table>
<thead>
<tr>
<th>$O_i$</th>
<th>$C_i(O_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_1 = 1$</td>
<td>$e_1^1(1) = C_1(1)$</td>
</tr>
<tr>
<td>$O_1 = 2$</td>
<td>${ e_1^1(2), e_1^2(2) } = C_1(2)$</td>
</tr>
<tr>
<td></td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$O_1 = O_{\max}$</td>
<td>${ e_1^1(O_{\max}), e_1^2(O_{\max}), \ldots } = C_1(O_{\max})$</td>
</tr>
</tbody>
</table>

$\tau_1$

\[ 0 \]

\[ D_1 \]

\[ P_1 \]

\[ \vdots \]

\[ 0 \]

\[ D_N \]

\[ P_N \]

$\delta_N = \frac{C_N(O_N = 3)}{D_N}$

$0$

$D_N$

$P_N$
Problem Formulation (2 / 2)

Four control parameters

\[ \tau_i \]

- \( \text{deadline } D_i \)
- \( \text{period } P_i \)

Parallelization option \( O_i \)

- \( \text{offset } \Phi_i \)
- \( \text{artificial deadline } d_i \)
- \( \text{artificial period } p_i \)

\[ \delta_i((\Phi_i, O_i, d_i, p_i), t) = \begin{cases} 
C_i(O_i)/d_i & (t - \Phi_i) \mod p_i \leq d_i \\
0 & (t - \Phi_i) \mod p_i > d_i 
\end{cases} \]

minimize \[ \max_{0 \leq t \leq HP} \sum_{i=1}^{N} \delta_i((\Phi_i, O_i, d_i, p_i), t) \]

\( d_i: \) artificial deadline \( (d_i < D_i) \)
\( O_i: \) parallelization option
\( p_i: \) artificial period \( (p_i < P_i) \)
\( \Phi_i: \) task offset
Preliminary Solution

3-step approach

**Step 1. per-task tradeoff**
- For given $d_i$, $O_i$ is decided

**Step 2. tradeoff of task group with same period**
- $P_1 = P_2 = P$, $d_i, D_i$ are decided

**Step 3. system-wide tradeoff**
- Back to original periods

System peak density
- $\tau_1, \tau_3$

Task group peak density
- $\tau_2, \tau_4$

Task group tradeoff
- $\tau_1, \tau_3$

Stacking
Simulation Result

Simulation Environment

1000 task sets
N = [3, 15]
M = 8
deadline looseness factor = deadline / period

![Graph showing schedulability vs deadline looseness factor for different thread setups.](image-url)
Remaining Issues

• Considering Inter-core Memory Interference

• Complicated Task Model
  • Multi-segment task model
  • Fork-join task model

• Tradeoff in G-EDF

https://github.com/rubis-lab/RealTimeTaskSimulator
Thank you