What Have We Done So Far?

- Scheduling a set of tasks with various constraints on a *single* processor.

- What should we do if schedulability condition for the given task set can’t be met?

- Question: which tasks should be assigned to which processors and why?

- Ideally, *combined* task assignment and scheduling is desirable, but this is *very hard*.

- *Common approach*: assign tasks and then schedule them on each processor.
Task Assignment

• What should we consider for task assignment?

• NP-Complete ⇒ Use heuristics

Examples:

• Utilization-balancing algorithm: assign tasks one-by-one selecting the least utilized processor

\[
\frac{\sum_{i=1}^{p} (u_i^B)^2}{\sum_{i=1}^{p} (u_i^*)^2} \leq \frac{9}{8}
\]

where \(u_i^*\) = \(P_i\)’s utilization under an optimal alg. that minimizes \(\sum\) utilization\(^2\)

\(u_i^B\) = \(P_i\)’s utilization under best-fit alg.
Next-fit alg for RM scheduling

- Homogeneous multiprocessor systems

- There are $m$ classes of tasks such that
  - $T_i$ belongs to class $j < m$ if $\frac{1}{2^{j+1}} - 1 < \frac{e_i}{p_i} \leq 2^{\frac{1}{j}} - 1$.
  - $T_i$ belongs to class $m$ otherwise.

- Each class of tasks are assigned to a corresponding set of processors.
Example

There are $m = 4$ task classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Untilization bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>$(0.41, 1.00]$</td>
</tr>
<tr>
<td>$C_2$</td>
<td>$(0.26, 0.41]$</td>
</tr>
<tr>
<td>$C_3$</td>
<td>$(0.19, 0.26]$</td>
</tr>
<tr>
<td>$C_4$</td>
<td>$(0.00, 0.19]$</td>
</tr>
</tbody>
</table>

Task set

<table>
<thead>
<tr>
<th>Task set</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_i$</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>$P_i$</td>
<td>10</td>
<td>21</td>
<td>22</td>
<td>24</td>
<td>30</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>$u(i)$</td>
<td>0.50</td>
<td>0.33</td>
<td>0.14</td>
<td>0.04</td>
<td>0.33</td>
<td>0.40</td>
<td>0.02</td>
</tr>
<tr>
<td>Class</td>
<td>$C_1$</td>
<td>$C_2$</td>
<td>$C_4$</td>
<td>$C_4$</td>
<td>$C_2$</td>
<td>$C_2$</td>
<td>$C_4$</td>
</tr>
<tr>
<td>Processor</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task set</th>
<th>$T_8$</th>
<th>$T_9$</th>
<th>$T_{10}$</th>
<th>$T_{11}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_i$</td>
<td>3</td>
<td>9</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>$P_i$</td>
<td>55</td>
<td>70</td>
<td>90</td>
<td>95</td>
</tr>
<tr>
<td>$u(i)$</td>
<td>0.05</td>
<td>0.13</td>
<td>0.19</td>
<td>0.22</td>
</tr>
<tr>
<td>Class</td>
<td>$C_4$</td>
<td>$C_4$</td>
<td>$C_4$</td>
<td>$C_3$</td>
</tr>
<tr>
<td>Processor</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Kang Shin (kgshin@eecs.umich.edu)
Bin-packing assignment for EDF

• Same assumptions on tasks and processors as Next-fit alg.

• Task set is EDF-schedulable if $U \leq 1$.

• Assign tasks such that $U \leq 1$ for all processors.
Myopic offline scheduling alg

- Can consider resources other than CPU

- Given: set of tasks, their arrival times, execution times, deadlines.

- Allocation tree:
  - Root: null allocation
  - Node: an assignment and scheduling of a subset of tasks.
  - Child node: parent’s allocation + a task
  - Leaf: “complete” allocation.
  - How many levels for an $n$-task system?
  - A level-$i$ node means?
  - Very expensive to generate a complete allocation tree $\Rightarrow$ heuristics.
Combined Assignment and Scheduling

- *Static* (offline) assignment of periodic and/or critical tasks: myopic scheduling, B&B alg.

- *Dynamic* (online) load sharing of aperiodics and/or non-criticals
  - Bidding
  - Focused addressing
  - Drafting
  - Buddy

Kang Shin (kgshin@eecs.umich.edu)
Offline Allocation of Periodics


- **Task allocation**: combined task assignment and scheduling of periodics
  - Derive an “optimal” assignment that yields feasible schedules for all processors. How?

- Main features:
  - Inter-task communications $\Rightarrow$ precedence constraints hence task structure.
  - Tasks are periodic and time-critical $\Rightarrow$ allocation objective function.

- Want allocation $x$ of communicating periodic tasks in a *heterogeneous* distributed system that minimizes *system hazard*, $\Theta(x)$, or maximum *normalized task response time*.

Kang Shin (kgshin@eecs.umich.edu)
System Model

- Tasks $T = \{T_i : i = 1, 2, \ldots, m\}$;

Heterogeneous PNs
$N = \{N_k : k = 1, 2, \ldots, n\}$.

- Allocation constraints:
  - Co-location of $T_i$ and $T_j$ on same PN.
  - Location of $T_i$ and $T_j$ on different PNs.
  - Location of $T_i$ on a special PN.

- Task invocations and release times, precedence constraints, planning cycle.

- Execution times of computation and communication modules.
Example Task Graph

L=40
T1 (P1 = 40)

T2 (P2 = 40)

T3 (P3 = 20)

M1
M2
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M13
M14
M15
M16
M17
M18
M19
M20
M21
M22
M23
M24

Kang Shin (kgshin@eecs.umich.edu)
Problem Formulation

- Normalized task response time of the $v$-th invocation of $T_i$:
  \[
  \bar{c}_{iv} = \frac{c_{iv} - r_{iv}}{d_{iv} - r_{iv}}
  \]

- System hazard under allocation $x$:
  \[
  \Theta^x = \max_{T_i \in T} \bar{c}_{iv}
  \]

- **Problem**: find an optimal $x^*$ that minimizes the system hazard.

- Both $\bar{c}_{iv}$ and $\Theta^x$ depend on:
  - how tasks are assigned under $x$ and
  - how assigned tasks are scheduled on each PN.

Kang Shin (kgshin@eecs.umich.edu)
Task Allocation Algorithm

- Consists of two branch-and-bound algorithms, one for assignment (B&BA) and the other for scheduling (B&BS).

- Same as traversing a tree

- Vertex = allocation

- Complete allocation = leaf node; B&BS alg

- Partial allocation = intermediate vertex; B&BS is too expensive, so compute and use a lower bound of optimal system hazard
Branch and Bound Algorithm

1 let active set \( A = \{ \text{Root} \} \)
2 let vertex cost \( \Theta(\text{Root}) = 0 \)
3 let best solution cost, \( \Theta_{\text{min}} = \infty \)

4 while true do
5 let \( V_{\text{best}} = \) minimum cost vertex in \( A \)
6 if \( V_{\text{best}} \) is a leaf vertex then
7 prune all vertices \( V \in A \) except \( V_{\text{best}} \)
8 return \( V_{\text{best}} \) as optimal solution
9 else
10 generate (task assignments of) all children of \( V_{\text{best}} \)
11 remove \( V_{\text{best}} \) from active set \( A \)
12 for each child \( x \) of \( V_{\text{best}} \) do
13 if assignment constraints in set \( AC \) are not satisfied then prune \( x \)
14 else
15 compute vertex cost \( \Theta(x) \)
16 add \( x \) to active set \( A \)
17 if \( x \) is a leaf vertex then
18 if \( \Theta(x) < \Theta_{\text{min}} \) then
19 \( \Theta_{\text{min}} = \Theta(x) \)
20 prune all vertices \( V \in A \) for which \( V \neq x \) and \( \Theta(V) \geq \Theta_{\text{min}} \)
21 else prune \( x \)
22 end if
23 end if
24 end for
25 end if
26 end while
Search Tree

Root of the Search Tree

Allocating T1

Allocating T2

Allocating T3
B&BA Algorithm

- A terminal vertex or complete assignment: B&BS alg based on dominance properties.

- For each non-terminal vertex or partial assignment $x$:
  - B&BS is too expensive
  - As long as a lower-bound, $\Theta^x_{lb}$ of the optimal cost for $x$ is used, B&BA will find an optimal assignment.
  - $\Theta^x_{lb}$ is obtained by relaxing task invocation times, precedence constraints, etc.
Computing Lower-Bound Vertex Cost

1. Compute the minimum computational load imposed on each processor by tasks already assigned to PNs at search vertex $x$.

2. Estimate the minimum additional load to be imposed on each PN due to those tasks not yet assigned at $x$.

3. Schedule the combined load at each PN and compute the system hazard. We ensure that the system hazard of the resulting schedule is a lower bound on the system hazard of any leaf vertex descending from $x$, i.e., it represents $\Theta(x) = \Theta_{lb}(x)$. 

Kang Shin (kgshin@eecs.umich.edu)
B&BS Algorithms

Scheduling tasks w.r.t. $\Theta$ for a given complete assignment is NP-Hard $\Rightarrow$ Dominance properties are derived to guide search for an optimal schedule.

- Preemptions which do not reduce $\Theta$ must be disallowed.

- A PN is not allowed to idle when there are ready (uncompleted) modules on the PN.

- Always advantageous to reduce the completion time of a task without increasing others’.
Example

The same as before: 3 tasks and 2 PNs
Module execution times on $N_1$

<table>
<thead>
<tr>
<th>$M_j$</th>
<th>$e_{j1}$</th>
<th>$M_j$</th>
<th>$e_{j1}$</th>
<th>$M_j$</th>
<th>$e_{j1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>4</td>
<td>$M_9$</td>
<td>1:3</td>
<td>$M_{17}$</td>
<td>1:3</td>
</tr>
<tr>
<td>$M_2$</td>
<td>1:4</td>
<td>$M_{10}$</td>
<td>1:4</td>
<td>$M_{18}$</td>
<td>1</td>
</tr>
<tr>
<td>$M_3$</td>
<td>2</td>
<td>$M_{11}$</td>
<td>1</td>
<td>$M_{19}$</td>
<td>2</td>
</tr>
<tr>
<td>$M_4$</td>
<td>2</td>
<td>$M_{12}$</td>
<td>2:4</td>
<td>$M_{20}$</td>
<td>0:1</td>
</tr>
<tr>
<td>$M_5$</td>
<td>2:6</td>
<td>$M_{13}$</td>
<td>2</td>
<td>$M_{21}$</td>
<td>1:3</td>
</tr>
<tr>
<td>$M_6$</td>
<td>2</td>
<td>$M_{14}$</td>
<td>0:2</td>
<td>$M_{22}$</td>
<td>1</td>
</tr>
<tr>
<td>$M_7$</td>
<td>1</td>
<td>$M_{15}$</td>
<td>2:3</td>
<td>$M_{23}$</td>
<td>2</td>
</tr>
<tr>
<td>$M_8$</td>
<td>1:2</td>
<td>$M_{16}$</td>
<td>3</td>
<td>$M_{24}$</td>
<td>1:2</td>
</tr>
</tbody>
</table>
Search Tree Generated and Optimal Schedule

Kang Shin (kgshin@eecs.umich.edu)
Task Graph at Vertex 5

L = 40

T_1 (P_1 = 40)  

\[ d_{2,10} = 8 \]

T_2 (P_2 = 40)  

\[ d_{8,20} \]

T_3 (P_3 = 20)  

\[ \rho_1 = 0 \]

\[ \rho_2 = \min \{4, 0.5\} = 0.5 \]

\[ 
\begin{align*}
e_1 &= 4 \\
e_2 &= 4 \\
e_3 &= 2 \\
e_4 &= 2 \\
e_5 &= 6 \\
e_6 &= 1.5 \\
e_7 &= 0.5 \\
e_8 &= 0.5 \\
e_9 &= 0.5 \\
e_{10} &= 2 \\
e_{11} &= 0.5 \\
e_{12} &= 2 \\
e_{13} &= 1 \\
e_{14} &= 0 \\
e_{15} &= 1 \\end{align*}
\]

\[ 
\begin{align*}
d_{17,9} \\
d_{12,5} = 10 \\
d_{14,24} \\
d_{21,15} \\
\end{align*}
\]

Kang Shin (kgshin@eecs.umich.edu)
## Computational Experiences

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Expanded Vertices</th>
<th>Total Space</th>
<th>% Expanded</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>18</td>
<td>4096</td>
<td>0.43</td>
</tr>
<tr>
<td>8</td>
<td>65</td>
<td>65536</td>
<td>0.1</td>
</tr>
<tr>
<td>10</td>
<td>95</td>
<td>1048576</td>
<td>0.01</td>
</tr>
<tr>
<td>12</td>
<td>133</td>
<td>16777216</td>
<td>0.0008</td>
</tr>
<tr>
<td>14</td>
<td>274</td>
<td>268435456</td>
<td>0.0000004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Expanded Vertices</th>
<th>Coef of Variation</th>
<th>Total Space</th>
<th>% Expanded</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>16</td>
<td>0.35</td>
<td>256</td>
<td>6.17</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>0.8</td>
<td>65536</td>
<td>0.06</td>
</tr>
<tr>
<td>6</td>
<td>38</td>
<td>0.8</td>
<td>1679616</td>
<td>0.002</td>
</tr>
</tbody>
</table>

Kang Shin (kgshin@eecs.umich.edu)