A Lookup-table Driven Approach to Partitioned Scheduling

Bipasa Chattopadhyay
Sanjoy Baruah

Department of Computer Science
The University of North Carolina at Chapel Hill
Partitioned Scheduling

- Given - a task system $\tau$ and $m$ processors
- Objective - partition the tasks onto the processors
- Advantages - Good cache affinity, no migration delay
Implicit Deadline Sporadic Task System

- Each task has a -
  - Worst Case Execution Time, $C_i$
  - Time Period, $T_i$
  - Deadline, $D_i = T_i$
  - Utilization, $u_i = \frac{C_i}{T_i}$
Partitioned Scheduling

- Implicit Deadline Sporadic Task System
  - Utilization, $u_i = \frac{C_i}{T_i}$

- Each processor is identical and has computing capacity $= 1$

- Scheduling algorithm on each processor is pre-emptive EDF
  - optimal uniprocessor scheduling algorithm if sum of utilizations of tasks $\leq 1$
Partitioning tasks is equivalent to bin-packing
  - Bin-packing is \textbf{NP-hard} in the strong sense

Hochbaum and Shmoys designed a polynomial time approximation scheme (PTAS)
Partitioning tasks is equivalent to bin-packing
  - Bin-packing is **NP-hard** in the strong sense

Hochbaum and Shmoys designed a polynomial time approximation scheme (PTAS)

If an **optimal algorithm** can partition a task system onto $m$ processors each of computing capacity 1 then a **PTAS algorithm** can partition the task system onto $m$ processors each of computing capacity $1 + \epsilon$ ($0 < \epsilon < 1$) \textit{in polynomial time}. 
Proposed PTAS partitioning algorithm has large run time

We suggest a lookup-table driven approach

Pre-compute the lookup-table
  - The results of the expensive computation are stored in a lookup-table
  - The table is consulted for task partitioning
Computing the Lookup-table

- The lookup-table is computed only once for a given value of $m$ and $\epsilon$
Computing the Lookup-table

- Choosing $\epsilon$
- Determining Utilization Values
- Single Processor Configurations
- Multiprocessor Configurations
Computing the Lookup-table

- Choosing $\epsilon$
- Determining Utilization Values
- Single Processor Configurations
- Multiprocessor Configurations
Choosing $\epsilon$

If an optimal algorithm can partition a task system onto $m$ processors each of computing capacity 1 then a PTAS algorithm can partition the task system onto $m$ processors each of computing capacity $1 + \epsilon$ in polynomial time ($0 < \epsilon < 1$).
Choosing $\epsilon$

If an optimal algorithm can partition a task system onto $m$ processors each of computing capacity $1 \left( \frac{1}{1+\epsilon} \right)$ then a PTAS algorithm can partition the task system onto $m$ processors each of computing capacity $1 + \epsilon$ \quad (1)

in polynomial time \quad (0 < \epsilon < 1).
For example -

system utilization loss = 10%

⇒ \[ \frac{1}{1+\epsilon} \geq 0.9 \]

⇒ \[ \epsilon \leq \frac{1}{9} \approx 0.1 \]
For example -

system utilization loss = 10%

\[ \Rightarrow \frac{1}{1+\epsilon} \geq 0.9 \]
\[ \Rightarrow \epsilon \leq \frac{1}{9} \approx 0.1 \]

\[ \Rightarrow \epsilon = 0.1 \]
For example -

- system utilization loss = 10\%

\[ \Rightarrow \frac{1}{1+\epsilon} \geq 0.9 \]
\[ \Rightarrow \epsilon \leq \frac{1}{9} \approx 0.1 \]

- \( \epsilon = 0.1 \)

- The size of the lookup-table is a function on \( m \) and \( \epsilon \)
  - If \( \epsilon \) decreases, the size of the lookup-table and the time needed to compute it increases
Computing the Lookup-table

- Choosing $\epsilon$
- Determining Utilization Values
- Single Processor Configurations
- Multiprocessor Configurations
Determining Utilization Values

Let tasks have fixed utilization values

- Fixed utilization values
Determining Utilization Values

- Let tasks have fixed utilization values

\[ V(\varepsilon) \text{ - set of fixed utilization values} \]
Determining Utilization Values

- Let tasks have fixed utilization values

\[ \mathcal{V}(\epsilon) = \{\epsilon, \epsilon(1 + \epsilon), \epsilon(1 + \epsilon)^2, \ldots, \epsilon(1 + \epsilon)^j\} \]

- \( j \) is a non-negative integer
- \( \epsilon(1 + \epsilon)^j \leq 1 \)
Let $\epsilon = 0.3$ (running example)

<table>
<thead>
<tr>
<th>$j$</th>
<th>util. value $\times (1.3)^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.3000</td>
</tr>
<tr>
<td>1</td>
<td>0.3900</td>
</tr>
<tr>
<td>2</td>
<td>0.5070</td>
</tr>
<tr>
<td>3</td>
<td>0.6591</td>
</tr>
<tr>
<td>4</td>
<td>0.8568</td>
</tr>
<tr>
<td>5</td>
<td>1.114</td>
</tr>
</tbody>
</table>

$V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$
Let $\epsilon = 0.3$ (running example)

<table>
<thead>
<tr>
<th>j</th>
<th>util. value $(0.3) \times (1.3)^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.3000</td>
</tr>
<tr>
<td>1</td>
<td>0.3900</td>
</tr>
<tr>
<td>2</td>
<td>0.5070</td>
</tr>
<tr>
<td>3</td>
<td>0.6591</td>
</tr>
<tr>
<td>4</td>
<td>0.8568</td>
</tr>
<tr>
<td>5</td>
<td>1.114</td>
</tr>
</tbody>
</table>

$V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$

As $\epsilon$ decreases the number of utilization values in $V(\epsilon)$ increases

If $\epsilon = 0.1$, size of $V(\epsilon) = 25$
As $\epsilon$ decreases -

- The solution is closer to optimal - Less inflation, less utilization loss
As $\epsilon$ decreases -
- The solution is **closer to optimal** - Less inflation, less utilization loss

The size of the lookup-table and the **time** needed to compute it increases
Computing the Lookup-table

- Choosing $\epsilon$
- Determining Utilization Values
- **Single Processor Configurations**
- Multiprocessor Configurations
How many different tasks with utilization in $V(\epsilon)$ can be assigned to a single processor?
How many different tasks with utilization in $V(\epsilon)$ can be assigned to a single processor?

$V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$

Configurations determine the number of tasks of each utilization

- 3, 0, 0, 0, 0, 0 - 3 tasks of $u_i = 0.3$ are assigned to a processor
- 1, 1, 0, 0, 0 - 1 task of $u_i = 0.3$ and 1 task of $u_j = 0.39$ are assigned to a processor
$V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$

- **Valid configuration** - sum of utilizations of tasks $\leq 1$
  - $3, 0, 0, 0, 0 (0.3 + 0.3 + 0.3 \leq 1)$
  - $1, 1, 0, 0, 0 (0.3 + 0.39 \leq 1)$
$V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$

- **Valid configuration** - sum of utilizations of tasks $\leq 1$
  - $3, 0, 0, 0, 0$ ($0.3 + 0.3 + 0.3 \leq 1$)
  - $1, 1, 0, 0, 0$ ($0.3 + 0.39 \leq 1$)

- **Maximal Configuration** - no other configuration can fit more tasks
  - $3, 0, 0, 0, 0$
  - $1, 1, 0, 0, 0$ is not a maximal configuration ($0.3 + 0.39 + 0.3 = 0.99$)
  - $2, 1, 0, 0, 0$ is a maximal configuration. We say that $2, 1, 0, 0, 0$ *dominates* $1, 1, 0, 0, 0$
The lookup-table for $\epsilon = 0.3$

Maximal single processor configurations

<table>
<thead>
<tr>
<th>Config ID</th>
<th>0.3000</th>
<th>0.3900</th>
<th>0.5070</th>
<th>0.6591</th>
<th>0.8568</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
The lookup-table for $\epsilon = 0.3$

Maximal single processor configurations

<table>
<thead>
<tr>
<th>Config ID</th>
<th>0.3000</th>
<th>0.3900</th>
<th>0.5070</th>
<th>0.6591</th>
<th>0.8568</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The lookup-table for $\epsilon = 0.1$ has 9604 maximal single processor configurations
Computing the Lookup-table

- Choosing $\epsilon$
- Determining Utilization Values
- Single Processor Configurations
- Multiprocessor Configurations
Multiprocessor Configurations

- Maximal single processor configurations are used to determine maximal $m$ processor configurations
Maximal single processor configurations are used to determine maximal $m$ processor configurations

If $m = 2$, combine any pair of maximal single processor configurations

<table>
<thead>
<tr>
<th>Single-proc.</th>
<th>Config ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt; A &gt;$</td>
<td>3 0 0 0 0 0</td>
</tr>
<tr>
<td>$&lt; B &gt;$</td>
<td>2 1 0 0 0 0</td>
</tr>
<tr>
<td>$&lt; C &gt;$</td>
<td>1 0 1 0 0 0</td>
</tr>
<tr>
<td>$&lt; D &gt;$</td>
<td>1 0 0 1 0 0</td>
</tr>
<tr>
<td>$&lt; E &gt;$</td>
<td>0 2 0 0 0 0</td>
</tr>
<tr>
<td>$&lt; F &gt;$</td>
<td>0 1 1 0 0 0</td>
</tr>
<tr>
<td>$&lt; G &gt;$</td>
<td>0 0 0 0 0 1</td>
</tr>
</tbody>
</table>
Maximal single processor configurations are used to determine maximal $m$ processor configurations.

If $m = 2$, combine any pair of maximal single processor configurations.

```
<table>
<thead>
<tr>
<th>Single-proc. Config ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; A &gt; 3 0 0 0 0 0</td>
</tr>
<tr>
<td>&lt; B &gt; 2 1 0 0 0 0</td>
</tr>
<tr>
<td>&lt; C &gt; 1 0 1 0 0 0</td>
</tr>
<tr>
<td>&lt; D &gt; 1 0 0 1 0 0</td>
</tr>
<tr>
<td>&lt; E &gt; 0 2 0 0 0 0</td>
</tr>
<tr>
<td>&lt; F &gt; 0 1 1 0 0 0</td>
</tr>
<tr>
<td>&lt; G &gt; 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>
```

Example -

1, 2, 1, 0, 0 $<$ C, E $>$
Multiprocessor Configurations

- Maximal single processor configurations are used to determine maximal \( m \) processor configurations
- If \( m = 2 \), combine any pair of maximal single processor configurations

<table>
<thead>
<tr>
<th>Single-proc. Config ID</th>
<th>3 0 0 0 0 0</th>
<th>2 1 0 0 0 0</th>
<th>1 0 1 0 0 0</th>
<th>1 0 0 1 0 0</th>
<th>0 2 0 0 0 0</th>
<th>0 1 1 0 0 0</th>
<th>0 0 0 0 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; A &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; B &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; C &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; D &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; E &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; F &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; G &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Example -
  - 1, 2, 1, 0, 0 < C, E >
  - 2, 2, 1, 0, 0 < B, F >
- $V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$

- **Valid Configuration** - sum of utilization of tasks on each processor $\leq 1$
  - $1, 2, 1, 0, 0 < C, E >$
  - $2, 2, 1, 0, 0 < B, F >$

- Combining 2 maximal single processor configurations results in a valid $m = 2$ processor configuration.
\[ V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\} \]

- **Valid Configuration** - sum of utilization of tasks on each processor \(\leq 1\)
  - \(1, 2, 1, 0, 0 < C, E >\)
  - \(2, 2, 1, 0, 0 < B, F >\)

- Combining 2 maximal single processor configurations results in a valid \(m = 2\) processor configuration.

- **Maximal Configuration** - A configuration is maximal if no other configuration dominates it
  - Configuration \(< B, F >\) dominates \(< C, E >\)
  - Remove configurations that are not maximal
The lookup-table consists of only maximal $m$ processor configurations
- The lookup-table consists of only maximal $m$ processor configurations

- We compute maximal $m$ processor configurations from maximal $k$ processor configurations, $k \leq m$

- If $m = 4$, combine any pair of maximal $m = 2$ processor configurations

<table>
<thead>
<tr>
<th>Single-Proc. Config ID</th>
<th>0.3000</th>
<th>0.3900</th>
<th>0.5070</th>
<th>0.6591</th>
<th>0.8568</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;D, D, E, C&gt;</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>&lt;F, F, E, A&gt;</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;F, F, F, G&gt;</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>&lt;G, D, C, B&gt;</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&lt;D, D, D, C&gt;</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Partitioning Tasks in Polynomial Time

Task system $\tau$ – $n$ tasks

$f$ = Utilization Inflation

Input to LUT

Task Assignment

Yes. (tasks assigned to processors)

No. (task system is not schedulable on $m$ processors each of utilization $1/1+\varepsilon$)
Partitioning Tasks in Polynomial Time

Task system $\tau - n$ tasks

$\mathbf{f}$

Input to LUT

Task Assignment

Yes. (tasks assigned to processors)

No. (task system is not schedulable on $m$ processors each of utilization $1/1+\varepsilon$)

$f = $ Utilization Inflation
Utilization Inflation

- Inflate the utilization values of the task to equal the nearest utilization value \( \in V(\epsilon) \)

- Tasks with \( u_i \geq \frac{\epsilon}{1+\epsilon} \) (large tasks) are inflated by at most \( (1 + \epsilon) \)
  - Inflation factor = \( \frac{\epsilon(1+\epsilon)^{j+1}}{u_i} < \frac{\epsilon(1+\epsilon)^{j+1}}{\epsilon(1+\epsilon)^j} = (1 + \epsilon) \)
Utilization Inflation

- Inflating the utilization values of the task to equal the nearest utilization value $\in V(\varepsilon)$

- Tasks with $u_i \geq \frac{\varepsilon}{1+\varepsilon}$ (large tasks) are inflated by at most $(1 + \varepsilon)$
  - Inflation factor $= \frac{\varepsilon(1+\varepsilon)^{j+1}}{u_i} < \frac{\varepsilon(1+\varepsilon)^{j+1}}{\varepsilon(1+\varepsilon)^j} = (1 + \varepsilon)$

- Tasks with $u_i < \frac{\varepsilon}{1+\varepsilon}$ (small tasks) are not inflated
Example

- $V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$

- Given task system $\tau$ -

\[
\begin{align*}
\frac{1}{5}, & \quad \frac{1}{5}, \quad \frac{1}{3}, \quad \frac{7}{20}, \quad \frac{9}{25}, \quad \frac{2}{5}, \quad \frac{1}{2}, \quad \frac{1}{2}, \quad \frac{3}{4}
\end{align*}
\]
Example

- $V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$

- Given task system $\tau$ -

- Large tasks are inflated -

\[
\frac{1}{5}, \frac{1}{5}, \frac{1}{3}, \frac{7}{20}, \frac{9}{25}, \frac{2}{5}, \frac{1}{2}, \frac{1}{2}, 3, \frac{4}{4}
\]
Example

- $V(0.3) = \{0.3000, 0.3900, 0.5070, 0.6591, 0.8568\}$

- Given task system $\tau$ -

- Large tasks are inflated -

- Input to the lookup-table (small tasks are ignored) -
Partitioning Tasks in Polynomial Time

Task system $\tau - n$ tasks

$\mathbf{f}$

$\mathbf{f} = \text{Utilization Inflation}$

Input to LUT

Task Assignment

Lookup table

Yes. (tasks assigned to processors)

No. (task system is not schedulable on $m$ processors each of utilization $1/(1+\varepsilon)$)
Task Assignment

- Let $m = 4$, Input = 0, 3, 3, 0, 1

<table>
<thead>
<tr>
<th>0.3000</th>
<th>0.3900</th>
<th>0.5070</th>
<th>0.6591</th>
<th>0.8568</th>
<th>Single-Proc. Config ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>&lt;D, D, E, C&gt;</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>&lt;F, F, E, A&gt;</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>&lt;F, F, F, G&gt;</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;G, D, C, B&gt;</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>&lt;D, D, D, C&gt;</td>
</tr>
</tbody>
</table>

- Output = <F, F, F, G>
Given task system \( \tau = \frac{1}{5}, \frac{1}{5}, \frac{1}{3}, \frac{7}{20}, \frac{9}{25}, \frac{2}{5}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4} \)

Inflate large tasks-
\( \frac{1}{5}, \frac{1}{5}, 0.3900, 0.3900, 0.3900, 0.5070, 0.5070, 0.5070, 0.8568 \)

0, 3, 3, 0, 1 (Input) \( \implies \) Lookup-table \( \implies \) \( < F, F, F, G > \) (Output)
Given task system $\tau - \frac{1}{5}, \frac{1}{5}, \frac{1}{3}, \frac{7}{20}, \frac{9}{25}, \frac{2}{5}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}$

Inflate large tasks-
$\frac{1}{5}, \frac{1}{5}, 0.3900, 0.3900, 0.3900, 0.5070, 0.5070, 0.5070, 0.8568$

$0, 3, 3, 0, 1$(Input) $\implies$ Lookup-table $\implies < F, F, F, G >$(Output)

- Phase 1 - Large tasks are assigned as specified by configuration $< F, F, F, G >$
  - $0, 1, 1, 0, 0 < F >; 0, 0, 0, 0, 1 < G >$

<table>
<thead>
<tr>
<th>Single-Proc. Config ID</th>
<th>Task Assignment</th>
<th>Remaining Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>1/3, 2/5</td>
<td>0.2667</td>
</tr>
<tr>
<td>F</td>
<td>7/20, 1/2</td>
<td>0.15</td>
</tr>
<tr>
<td>F</td>
<td>9/25, 1/2</td>
<td>0.14</td>
</tr>
<tr>
<td>G</td>
<td>3/4</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Given task system \( \tau = \frac{1}{5}, \frac{1}{5}, \frac{1}{3}, \frac{7}{20}, \frac{9}{25}, \frac{2}{5}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4} \)

Inflate large tasks-
\( \frac{1}{5}, \frac{1}{5}, 0.3900, 0.3900, 0.3900, 0.5070, 0.5070, 0.5070, 0.8568 \)

0, 3, 3, 0, 1 (Input) \( \implies \) Lookup-table \( \implies \) \( < F, F, F, G > \) (Output)

- **Phase 1** - Large tasks are assigned as specified by configuration \( < F, F, F, G > \)
  - 0, 1, 1, 0, 0 \( < F > \); 0, 0, 0, 0, 1 \( < G > \)

<table>
<thead>
<tr>
<th>Single-Proc.</th>
<th>Config ID</th>
<th>Task Assignment</th>
<th>Remaining Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>1/3, 2/5</td>
<td></td>
<td>0.2667</td>
</tr>
<tr>
<td>F</td>
<td>7/20, 1/2</td>
<td></td>
<td>0.15</td>
</tr>
<tr>
<td>F</td>
<td>9/25, 1/2</td>
<td></td>
<td>0.14</td>
</tr>
<tr>
<td>G</td>
<td>3/4</td>
<td></td>
<td>0.25</td>
</tr>
</tbody>
</table>

- **Phase 2** - Small tasks are accommodated in the remaining capacity
Given task system $\tau = \frac{1}{5}, \frac{1}{5}, \frac{1}{3}, \frac{7}{20}, \frac{9}{25}, \frac{2}{5}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}$

Inflate large tasks-
$\frac{1}{5}, \frac{1}{5}, 0.3900, 0.3900, 0.3900, 0.5070, 0.5070, 0.5070, 0.8568$

$0, 3, 3, 0, 1$ (Input) $\implies$ Lookup-table $\implies$ $< F, F, F, G >$ (Output)

- **Phase 1 - Large tasks are assigned as specified by configuration $< F, F, F, G >$**
  - $0, 1, 1, 0, 0 < F >; 0, 0, 0, 0, 1 < G >$

<table>
<thead>
<tr>
<th>Single-proc.</th>
<th>Config ID</th>
<th>Task Assignment</th>
<th>Remaining Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>1/3, 2/5, 1/5</td>
<td></td>
<td>0.0667</td>
</tr>
<tr>
<td>F</td>
<td>7/20, 1/2</td>
<td></td>
<td>0.15</td>
</tr>
<tr>
<td>F</td>
<td>9/25, 1/2</td>
<td></td>
<td>0.14</td>
</tr>
<tr>
<td>G</td>
<td>3/4, 1/5</td>
<td></td>
<td>0.05</td>
</tr>
</tbody>
</table>

- **Phase 2 - Small tasks are accommodated in the remaining capacity**
Performance Guarantee

*If the partitioning algorithm fails to partition the tasks in $\tau$, then no algorithm can partition $\tau$ on a platform of $m$ processors each of computing capacity $\frac{1}{1+\epsilon}$.*
Performance Guarantee

If the partitioning algorithm fails to partition the tasks in $\tau$, then no algorithm can partition $\tau$ on a platform of $m$ processors each of computing capacity $\frac{1}{1+\epsilon}$.

- Phase 1 fails - tasks with utilization $\geq \frac{\epsilon}{1+\epsilon}$ are inflated by atmost $(1 + \epsilon)$
Performance Guarantee

*If the partitioning algorithm fails to partition the tasks in $\tau$, then no algorithm can partition $\tau$ on a platform of $m$ processors each of computing capacity $\frac{1}{1+\epsilon}$.*

- **Phase 1 fails** - tasks with utilization $\geq \frac{\epsilon}{1+\epsilon}$ are inflated by at most $(1 + \epsilon)$

  if inflated tasks can not be partitioned onto $m$ processors each of utilization 1 then *unmodified tasks* can not be partitioned onto $m$ processors each of utilization $\frac{1}{1+\epsilon}$
If the partitioning algorithm fails to partition the tasks in $\tau$, then no algorithm can partition $\tau$ on a platform of $m$ processors each of computing capacity $\frac{1}{1+\epsilon}$.

- **Phase 1 fails** - tasks with utilization $\geq \frac{\epsilon}{1+\epsilon}$ are inflated by at most $(1 + \epsilon)$
  
  if inflated tasks can not be partitioned onto $m$ processors each of utilization 1 then unmodified tasks can not be partitioned onto $m$ processors each of utilization $\frac{1}{1+\epsilon}$

- **Phase 2 fails** - tasks with utilization $< \frac{\epsilon}{1+\epsilon}$ can not be accommodated
Performance Guarantee

If the partitioning algorithm fails to partition the tasks in $\tau$, then no algorithm can partition $\tau$ on a platform of $m$ processors each of computing capacity $\frac{1}{1+\epsilon}$.

- **Phase 1 fails** - tasks with utilization $\geq \frac{\epsilon}{1+\epsilon}$ are inflated by atmost $(1 + \epsilon)$

  if inflated tasks can not be partitioned onto $m$ processors each of utilization $1$ then unmodified tasks can not be partitioned onto $m$ processors each of utilization $\frac{1}{1+\epsilon}$

- **Phase 2 fails** - tasks with utilization $< \frac{\epsilon}{1+\epsilon}$ can not be accommodated

  sum of utilizations of tasks assigned to each processor must be $> (1 - \frac{\epsilon}{1+\epsilon})$

  $\implies$ total utilization of $\tau > m \times (1 - \frac{\epsilon}{1+\epsilon})$

  $> m(\frac{1}{1+\epsilon})$
Context and Summary

- Build the lookup-table during platform synthesis
Context and Summary

- Build the lookup-table during platform synthesis
  - Ship the platform with the lookup-table
Context and Summary

- Store the lookup-table on a central server
Future Work

- What happens if the exact configuration does not exist in the table?
  - Configuration 0, 3, 3, 0, 1 was easy to find, what if configuration was 0, 2, 3, 0, 0
  - Need good heuristic for lookup function.
Future Work

- What happens if the exact configuration does not exist in the table?
  - Configuration 0, 3, 3, 0, 1 was easy to find, what if configuration was 0, 2, 3, 0, 0
  - Need good heuristic for lookup function.

- Partition the tasks onto processors and memory.
Thank You
Does better than First Fit - $0.25, 0.25 + \phi, 0.50, 0.50 + \phi$
(utilization $1.5 + 2 \times \phi$)

For $m = 2$ processors first fit can not partition this task system. Our partitioning algorithm can.

$\epsilon = 0.1, \phi = 0.01 \rightarrow 0.25, 0.26, 0.50, 0.51 \rightarrow 0.259, 0.285, 0.505, 0.556$
Size of lookup-table for $\epsilon = 0.15$ is -
- $m = 1$, 14KB (252)
- $m = 2$, 240KB (8523)
- $m = 3$, 3MB (122599)

Size of lookup-table for $\epsilon = 0.1$ is -
- $m = 1$, 250KB (9604)
- $m = 2$, $\approx$ tens of MB
- $m = 3$, $\approx$ hundreds of MB