Scheduling Strategies for minimizing response time on Heterogeneous Master-Slave Platforms

Laboratoire de l'Informatique du Parallélisme
ENS LYON

Coordonators:  Yves Robert
               Frederic Vivien

Student:       Maria Vlad
Outline

- Master-Slave platform
  - Platform description
- Known Results
  - Throughput optimization
  - Bandwidth centric principle
- First Heuristic — Optimization of Bandwidth centric principle
- Second Heuristic — Analytical construction of periodic schedules
- Conclusions/Results
Outline

- Master-Slave platform
  - Platform description
- Known Results
  - Throughput optimization
  - Bandwidth centric principle
- First Heuristic — Optimization of Bandwidth centric principle
- Second Heuristic — Analytical construction of periodic schedules
- Conclusions/Results
Outline

● Master-Slave platform
  - Platform description

● Known Results
  - Throughput optimization
  - Bandwidth centric principle

● First Heuristic — Optimization of Bandwidth centric principle

● Second Heuristic — Analytical construction of periodic schedules

● Conclusions/Results
Outline

- Master-Slave platform
  - Platform description
- Known Results
  - Throughput optimization
  - Bandwidth centric principle
- **First Heuristic** — Optimization of Bandwidth centric principle
- **Second Heuristic** — Analytical construction of periodic schedules
- Conclusions/Results
Outline

- Master-Slave platform
  - Platform description
- Known Results
  - Throughput optimization
  - Bandwidth centric principle
- First Heuristic — Optimization of Bandwidth centric principle
- Second Heuristic — Analytical construction of periodic schedules
- Conclusions/Results
Master-Slave platform
-Platform description-

- Heterogeneous platform – computation time and communication times are different for each slave
- The tasks sent to the slaves are *independent* and *identical* - each represents the same amount of computation
- Tasks are sent regularly to the slaves every \( R \) units of time
Master-Slave platform

-Platform description-

- $w_i$ – time needed for slave Pi to compute a task
- $c_i, d_i$ – time needed to send/receive a task to/from slave Pi
- The communication model is *one-port* – a processor can send/receive only one task at a given time
- The communications and the computations are *overlapped*
- **Complexity**
  - the minimum time to process $n$ tasks having $p$ slaves is $O(n^2p^2)$ using *Greedy Algorithm*
  - the problem is *polynomial* for a linear chain or for a fork graph
  - *NP-complete* for tree-shaped platforms
Outline

- Master-Slave platform
  - Platform description
- Known Results
  - Throughput optimization
  - Bandwidth centric principle
- First Heuristic - Optimization of Bandwidth centric principle
- Second Heuristic — Analytical construction of periodic schedules
- Conclusions/Results
Known Results

-Throughput optimisation-

- Traditional objective of scheduling algorithms - *makespan minimisation*
  - NP-hard in most practical situations, long and error-prone

- Instead of absolute minimisation of the execution time -> asymptotic optimality – optimal steady state algorithm

- Optimal steady state
  - for each processor determine the fraction of time spent computing and the fraction of time spent sending or receiving
  - try to maximize the *throughput* (number of tasks processed per time-unit)
Known results
-Bandwidth-Centric principle-

- Used for tree-shaped platforms
- *Bandwidth* - the communication speed of the parent node
- If two children are in *concurrence* for obtaining a task -> task is given to the child with *fastest* communication time -> optimize communication for parent
- Using a bottom-up transversal of the tree => obtain steady state throughput of the tree
  - leaves are reduced with the parent into a single node of equivalent computing power: 
    
    \[ w = \frac{1}{n_{task}(F)} \]
Example
**Known results**

-Bandwidth - Centric principle-

**Algorithm**

1) Sort the children by increasing communication times

\[ c_1 \leq c_2 \leq \ldots \leq c_k \]

2) Let \( p \) be the largest index so that \( \sum_{i=1}^{p} \frac{c_i}{w_i} \leq 1 \). If \( p < k \) let

\[ \varepsilon = 1 - \sum_{i=1}^{p} \frac{c_i}{w_i} \]

otherwise let \( \varepsilon = 0 \).

3) Then,

\[ n_{task}(F) = \min \left( \frac{1}{c_0}, \frac{1}{w_0} + \sum_{i=1}^{p} \frac{1}{w_i} + \frac{\varepsilon}{c_{p+1}} \right) \]
Known results
-Bandwidth - Centric principle-

- First term: $\frac{1}{c_0}$, the proc can not consume more tasks than sent by $P_{-1}$
- Second term:
  - if $p=k$ then all the slaves are fed with tasks and they are computing steadily
  - if $p<k$ some children will partially starve

- A slow processor with a fast communication link is preferred to a fast processor with a slow communication link !!!

- After solving the linear program
  - characterize the schedule during one time-period
  - derive an actual schedule whose asymptotic efficiency will hopefully be optimal
Examples

- Without saturation of the communication bandwidth all children can be kept *fully active*
Examples

- With saturation of the communication bandwidth some children are *partially idle* due to low bandwidth between Po and its parent.
Examples

- With saturation of the communication bandwidth- P3 is partially idle because of its high computation speed
Known results
-Bandwidth - Centric principle-

- Periodic schedule – detailed list of actions of the processors during a time period
  - Starting at time-step t₀ the whole pattern of computations and communications repeats every T time-units at time step t₀+T, t₀+2T and so on
  - Linear problem doesn’t necessarily imply a polynomial T in the problem size
  - T might be exponential in the problem size

- Drawback - the period can be very large
- Solution: - restriction to fixed length periods
Known results
-Bandwidth - Centric principle-

- Because the period is too large the response time can be also very big
- **Response time** – time passed since the task is released in the system till the master receives the result of the computation for that task
- Build the schedule considering the **response time** while enforcing a maximum throughput
- **Goal** – to obtain the maximum response time for a certain number of tasks as small as possible
Outline

- Master-Slave platform
  - Platform description
- Known Results
  - Throughput optimization
  - Bandwidth centric principle
- First Heuristic — Optimization of Bandwidth centric principle
- Second Heuristic — Analytical construction of periodic schedules
- Conclusions/Results
First Heuristic – Optimisation of Bandwidth centric principle

- The fork-graph platform is considered
- Having the maximum throughput $r_i \in Q$ for each node the periodic schedule is built:
  - The rate $R$ at which the tasks are arriving in the system is chosen as $2/(\text{maximum throughput of the master})$
  - Heuristic will ensure 50% of optimal throughput
- **Sending order** of the tasks to slaves
  - for each slave build a heap
  - each time a task is sent increase the heap of the slave with the value of its maximum throughput
  - next task sent to the slave with the lowest heap value
- **Receiving order**: a slave sends the result to the master when it receives another task from the master
Example

- The slaves are ordered after the sending time \( ci \)
- 3 slaves with the next values of the throughput

\[
\begin{align*}
    r_1 &= 0,5 \\
    r_2 &= 0,2 \\
    r_3 &= 0,3
\end{align*}
\]

\[
\begin{align*}
    \frac{1}{r_1} &= 2 \\
    \frac{1}{r_2} &= 5 \\
    \frac{1}{r_3} &= 3,33
\end{align*}
\]

- The sending order will be: P1, P3, P1, P2, P1, P3, P1, P3, P1, P2
Outline

- Master-Slave platform
  - Platform description
- Known Results
  - Throughput optimization
  - Bandwidth centric principle
- First Heuristic — Optimization of Bandwidth centric principle
- Second Heuristic — Analytical construction of periodic schedules
- Conclusions/Results
Second Heuristic—Theoretically build of periodic schedules

- Period $T$ will be fixed at the beginning - multiple of the arrival rate $R$ for the tasks
- $R$ – computed like in the first period
- $\rho$ - the objective response time
- The schedule will be built by:
  - Ordering the slaves by increasing order of $ci$, $wi$ or $ci+wi+di$
  - Find the **maximum response time** by building a schedule for the arrival rate $R$ and a certain period $T$
  - Find the **minimum response time** $\min(ci+wi+di)$
  - **Binary search** between the minimum and the maximum value of the response time $\Rightarrow$ the objective response time $\rho$ for which the schedule can be built

$$R = \frac{2}{\max \text{throughput}}$$
Building the schedule

• Apply the binary search for different values of the period $T$ – $T=5*R$, $T=10*R$, $T=20*R$
• Consider the period for which the objective response time is minimum
• The number of tasks in one period is: $tasksNo=\frac{T}{R}$
• The total number of periods is $totalNumberOfTasks = \frac{totalNumberOfTasks}{taskNo}$ where the total number of tasks for which the program is executed should be a multiple of $tasksNo$
Building the schedule

- **About the period**
  - the period will be wrapped around
  - if a task can not finish its execution or its communication in one period it will be scheduled for the next period
  - each task will have an interval for sending, an interval for computing and one for receiving
  - each interval has an offset which shows if the interval belongs to the current period or to a period before it
Example for building the schedule

<table>
<thead>
<tr>
<th>M</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ c[1]=1; \quad w[1]=4; \quad d[1]=1; \]
\[ c[2]=2; \quad w[2]=5; \quad d[2]=1; \quad R=3; \quad T=7*R; \quad \text{tasksNo}=7; \]
\[ c[3]=3; \quad w[3]=7; \quad d[3]=1; \]
**Example for building the schedule**

<table>
<thead>
<tr>
<th>M</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P2 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P3 -1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

\[ c[1]=1; \quad w[1]=4; \quad d[1]=1; \]
\[ c[2]=2; \quad w[2]=5; \quad d[2]=1; \quad R=3; \quad T=7*R; \quad \text{tasksNo}=7; \]
\[ c[3]=3; \quad w[3]=7; \quad d[3]=1; \]
Second Heuristic

- The objective response time is also considered when sending a task to a processor
  - If end date – release date > objective response time for one task => the next processor is tried

- If a task can not be sent to a certain slave because there is no more space or the response time is too big the cleanup has to be done
  - All the intervals used for sending, computing and receiving that task are deleted from the schedule