Real-Time Scheduling on Two-type Heterogeneous Multiprocessors

Gurulingesh Raravi, Björn Andersson, Konstantinos Bletsas, Vincent Nelis
1. Assign the tasks to processors
2. Schedule the tasks

Uni-processor scheduling
1. Assign the **tasks to groups** of processors
2. Schedule the tasks

Global scheduling

Global scheduling
System Model

- **Implicit-deadline** sporadic tasks
- **Two-type** heterogeneous multiprocessors
- **Task Assignment Problem**: Main Challenge

![Diagram](image)

- $\tau_1$
- $\tau_2$
- $P_1$
- $P_2$

Type-1

Type-2
Three Migration Models

- Fully migrative
- Intra migrative
- Non migrative

Uni-processor scheduling
Global scheduling

\(\tau\) \(\tau\) \(\tau\) \(\tau\) 
\(\tau\) \(\tau\) \(\tau\) 

Intra migrative

Non migrative
Scope of This Work

Adversary (OPTIMAL)

Algorithm

<table>
<thead>
<tr>
<th>Fully migrative</th>
<th>Intra migrative</th>
<th>Non migrative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fully migrative</strong></td>
<td><strong>Intra migrative</strong></td>
<td><strong>Non migrative</strong></td>
</tr>
<tr>
<td><strong>Algorithm</strong></td>
<td><strong>Adversary (OPTIMAL)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>LST90, SKB04, SKB04 ARB10</strong></td>
<td><strong>t-type; time = O(P), bound = 2</strong></td>
<td><strong>2-type; time = O(n * max(log n, m)), bound = 2</strong></td>
</tr>
<tr>
<td></td>
<td><strong>SA</strong></td>
<td><strong>t-type; time = O(P), bound = 4</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>2-type; time = O(P), bound = 4</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>2-type; time = O(n * max(log n, m)), bound = 2</strong></td>
</tr>
</tbody>
</table>
Problem Definition

• Problem P1: Intra-migrative assignment
  • Assign tasks to *processor types* so that each processor type is utilized to at most 100%.
Our Approach

• Step 1: Partition the tasks as **heavy** and **light**
  
  – Heavy
    
    • $H_1 = \{\text{cannot be assigned to type-2 processors}\}$
    
    • $H_2 = \{\text{cannot be assigned to type-1 processors}\}$

  – Light
    
    • $L = \{\text{can be assigned to both the processor types}\}$
Our Approach

• Step1: Partition the tasks as **heavy** and **light**
  - H1 = {cannot be assigned to type-2 processors}
  - H2 = {cannot be assigned to type-1 processors}
  - L = {can be assigned to both the processor types}

• Step2: Assign the **heavy** tasks
  - Assign H1 to type-1 and H2 to type-2
Our Approach

• Step 3: Assign the *light* tasks
  – **Sort the tasks in** $L$ **in non-increasing order of**
    
    - utilization of the task on type-2
    - utilization of the task on type-1
Our Approach

• Step 3: Assign the *light* tasks
  – Sort the tasks in \( L \) in non-increasing order of
    \[
    \frac{\text{utilization of the task on type-2}}{\text{utilization of the task on type-1}}
    \]

• Intuition: Left-hand side tasks are more preferable to be assigned to type-1
Sort and Assign (SA)

- **Problem P1**: Intra-migrative assignment
- **Property of SA**: At most one task split between type-1 and type-2
• **Problem P1**: Intra-migrative assignment

• **Theorem 1**:  
  – The approximation ratio of SA is: $1 + \alpha/2 \leq 1.5$
First Result – SA

Adversary (OPTIMAL)

- Fully migrative
- Intra migrative
- Non migrative

Algorithm

- Fully migrative
- Intra migrative
- Non migrative

---

CSV11

- t-type; time = \(O(P)\), bound = 4

SA

- 2-type; time = \(O(n \log n)\), bound = 1.5

---

LST90, SKB04, SKB04 ARB10

- t-type; time = \(O(P)\), bound = 2
- 2-type; time = \(O(n \times \max(\log n, m))\), bound = 2
**Problem Definition**

- **Problem P2**: Non-migrative assignment
  - Assign tasks to *processors* so that each processor is utilized to at most 100%.
Our Approach – SA-P

• Take the solution of SA

• Do wrap-around assignment
Our Approach – SA-P

- **Wrap-around assignment**

- **Properties** of wrap-around assignment
  - At most $|\text{type-1}| - 1$ tasks split between type-1
  - At most $|\text{type-2}| - 1$ tasks split between type-2
  - At most 1 task split between type-1 and type-2
• **Problem P2**: Non-migrative task assignment

• **Theorem 2**:  
  - The “approximation ratio” of SA-P is $1 + \alpha \leq 2$
Second Result – SA-P

**Adversary (OPTIMAL)**

- **Fully migrative**
- **Intra migrative**
- **Non migrative**

**Algorithm**

- **Fully migrative**
- **Intra migrative**
- **Non migrative**

Arrow Connections:

- **CSV11**
  - t-type; time = $O(P)$, bound = 4

- **SA-P**
  - 2-type; time = $O(n \log n)$, bound = 2

- **LST90, SKB04, SKB04 ARB10**
  - t-type; time = $O(P)$, bound = 2
  - 2-type; time = $O(n \times \max(\log n, m))$, bound = 2
Simulation Results – SA

- **Random task sets**
  - 100000 *critically feasible* task sets

- **Results**

![](image)
Simulation Results – SA-P

- Random task sets
  - 100000 critically feasible task sets

- Results

Histogram for SA-P

Number of task sets (in %)

Required processor speedup

- Random task sets
- 100000 critically feasible task sets

Results
Summary

• **Contributions**
  
  - **Adversary (OPTIMAL)**
  - **Algorithm**

  - **Fully migrative**
  - **Intra migrative**
  - **Non migrative**

  ![Diagram showing Adversary (OPTIMAL) and Algorithm with migration types]

  - **SA**
    - 2-type: time = \(O(n \log n)\), bound = 1.5
    - **SA-P**
      - 2-type: time = \(O(n \log n)\), bound = 2

• **Significance**
  
  - **Intra-migrative** assignment
    - SA: First solution
  
  - **Non-migrative** assignment
    - SA-P: better performance compared to SOTA