Abstract

The scheduling problem is specified by a set of workers, a set of tasks, an optimality criterion, environmental specifications, and by other constraints. The goal of a scheduling policy is to find an optimal schedule in the environment and to satisfy all constraints. The two main scheduling techniques are: centralized and decentralized. The main drawback of centralized scheduling is the central failure, which makes it non-practical for implementation in P2P and non-dependable environments. Decentralized scheduling is to locally schedule jobs to suitable workers from the client node. In this work, we propose a client side scheduler and Grid client for UNICORE based on HiLA API. The scheduling process is composed of two main steps: 1) gathering the resource information about TSs from the associated registry, and 2) assign submitted jobs to suitable TSs through adaptive matchmaking.
Scheduling Mechanism

Decentralized Scheduling scenario

Get Worker Specifications

Workers

Monitor

Initial List

1 2 3 4

W1
W2
W4
W3

Requester

Attribute Manager

1 6

List Manager

W2
W4
W3
W1

Matchmaker

W1
W2
W4
W3
W1

Subtask Pool

Modified List

Client

Scheduler

Distributor

Subtask Manager

Task Manager

1 2 3 4 5 6 7
Scheduling Mechanism

Matchmaking

Client

Matchmaker

Task requirements

Worker Capabilities

Degree of Matching

Worker Matchmaker

Task requirements

Worker Capabilities

Degree Of Matching

>=0 \rightarrow \text{True (Suitable)}

<0 \rightarrow \text{False (Not Suitable)}
Scheduling Mechanism

Matchmaking

Worker Capabilities:

Worker capabilities are calculated within a collection of time units \( \{TU(i) \rightarrow TU(i + NTU(T)) \} \), as follows:

Available CPU (VC) [MHz]

\[ \mu_{ov}(w,VC,i,d,NTU(T)), \sigma_{ov}(w,VC,i,d,NTU(T)) \]

Available Memory (VM) [MBytes]

\[ \mu_{ov}(w,VM,i,d,NTU(T)), \sigma_{ov}(w,VM,i,d,NTU(T)) \]

Number of Failures (NF)

\[ NF_{ov}(w,i,d,NTU(T)) = \text{Max}(NF(w,j,d)) \]
Task Requirements:

Task resource requirements are calculated from the resource usage of previous executions.

**CPU Cycles (CC (T)) [MCycles]**

\[ UC (T, w, e) = \% \text{ Processor time of (w) used by (T) at execution (e)} \]

**CPU Time of (T,e)**

\[ CPU \text{ Time of (T,e)} = UC(T,w,e) \times \text{Total execution time of (T)} \text{ [sec]} \]

**CC(T, e)**

\[ CC(T, e) = CPU \text{ Time of (T,e) [sec]} \times CPU \text{ Speed of (w) [MHz]} \]

\[ m(T, CC) = \text{Median}(CC(T, e)) \quad \{e = 1,2,3,\ldots, E_f\} \]

**Used Memory (UM (T)) [MBytes]**

\[ UM(T) = (\text{Input data of (T) [MBytes]} + \text{Output data of (T) [MBytes]} + \text{Intermediate data during execution of (T) [MBytes]}) \text{ [MBytes]} \]
The fuzzy matchmaking approach is implemented based on Takagi-Sugeno fuzzy model.

The following steps will construct the fuzzy inference process:

1. Fuzzification of Inputs.
3. Applying Implication Method.
4. Defuzzification.
Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)

1. Fuzzification of Inputs

Each available worker \((w)\) within the collection of TUs \(\{\text{TU}(i) \rightarrow \text{TU}(i + \text{NTU}(T))\} \) will have a separate fuzzy set represented by two membership functions. An input membership function is included for each parameter concerning worker capabilities.

The input membership functions can be described as follows:

\[
\text{Free}_\text{CPU}(x,w,i,d,\text{NTU}(T)) = \exp\left(-\frac{x - \mu_{ov}(w,\text{VC},i,d,\text{NTU}(T))}{\sigma_{ov}(w,\text{VC},i,d,\text{NTU}(T))}\right)
\]

\[
\text{Free}_\text{Memory}(x,w,i,d,\text{NTU}(T)) = \exp\left(-\frac{x - \mu_{ov}(w,\text{VM},i,d,\text{NTU}(T))}{\sigma_{ov}(w,\text{VM},i,d,\text{NTU}(T))}\right)
\]
Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)

1. Fuzzification of Inputs

The input values to the fuzzification process can be described as follows:

\[ \text{Required } _{CPU} (T) = \frac{m(T, CC)}{NTU (T) \times TU} \]

\[ \text{Required } _{Memory} (T) = UM (T) \]
Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)

2. Applying Fuzzy Operator

A separate rule will be created for each worker included in the matchmaking

\[
\text{If} \quad \text{Required\_CPU}(T) \text{ Is Free\_CPU}(w_j) \\
\text{AND} \quad \text{Required\_Memory}(T) \text{ Is Free\_Memory}(w_j) \\
\text{THEN} \quad \text{Suitable\_Worker}(T) = \text{ID}(w_j)
\]

\[ j = 1, 2, \ldots, N \]

\[
N: \text{number of workers}
\]

The rule weight is specified as a function of the number of failures:

\[
RW(w_i, d, NTU(T)) = \frac{1}{NF_{ov}(w_i, d, NTU(T))}
\]


**Scheduling Mechanism**

**Fuzzy Matchmaking approach (FMA)**

3. Applying Implication Method

The consequent of a rule is an output fuzzy set represented by an output membership function.

The output membership function associated with each fuzzy set will be in the form of a unique identifier of the associated worker $[\text{ID}(w_j) \ j = 1, 2, 3, \ldots, N]$.

![Diagram showing output membership functions and their values](image)

4. Defuzzification

Output = $\text{MAX} \ (\text{value}[\text{ID}(w_1)], \text{value}[\text{ID}(w_2)], \ldots, \text{value}[\text{ID}(w_n)])$
Scheduling Mechanism

Fuzzy Matchmaking approach (FMA)

**Rule1** (w1)
If Required_CPU(T) Is Free_CPU(w1) 
AND Required_Memory(T) Is Free_Memory(w1) 
Then Suitable_Worker(T) = ID(w1)

**Rule2** (w2)
If Required_CPU(T) Is Free_CPU(w2) 
AND Required_Memory(T) Is Free_Memory(w2) 
Then Suitable_Worker(T) = ID(w2)

**Rule6** (w6)
If Required_CPU(T) Is Free_CPU(w6) 
AND Required_Memory(T) Is Free_Memory(w6) 
Then Suitable_Worker(T) = ID(w6)

Input 1
CPU [MHz]

Input 2
Memory [Mbytes]

Output
ID(Worker)
Scheduling Mechanism

Simplified Fuzzy Matchmaking approach (SFMA)

Useful for use on the Internet where workers are PCs and ruffling consumption level of worker machines is expected.

Efficient for scheduling short running tasks.

Input membership functions:

\[
Free_{-}\text{CPU}(x, w, i, d) = \begin{cases} 
\frac{x}{\mu_{\text{cur}}(w, VC, i, d)} & x \leq \mu_{\text{cur}}(w, VC, i) \\
0 & x > \mu_{\text{cur}}(w, VC, i)
\end{cases}
\]

\[
Free_{-}\text{Memory}(x, w, i, d) = \begin{cases} 
\frac{x}{\mu_{\text{cur}}(w, VM, i, d)} & x \leq \mu_{\text{cur}}(w, VM, i) \\
0 & x > \mu_{\text{cur}}(w, VM, i)
\end{cases}
\]

\[
\mu_{\text{cur}}(w, VC, i, d) = \mu(w, VC, i - 1, d)
\]

\[
\mu_{\text{cur}}(w, VM, i, d) = \mu(w, VM, i - 1, d)
\]
$$\text{Free}_CPU(x, w, i) = \begin{cases} 
\frac{x}{2500} & x \leq 2500 \\
0 & x > 2500 
\end{cases}$$

\text{Required}_CPU(T) = 1400
## Performance Evaluation

**Performance evaluation of the proposed decentralized scheduling mechanism based on (SFMA)**

### 1. Parallel Execution Scheduling performance for 2400x2400 matrix size

<table>
<thead>
<tr>
<th>Number Of subtasks</th>
<th>Scheduling Mechanism</th>
<th>Subtask index</th>
<th>%CPU utilization</th>
<th>%Memory utilization</th>
<th>Execution time (Seconds)</th>
<th>NTU(T) (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>SFMA</td>
<td>1</td>
<td>99.07</td>
<td>86.3</td>
<td>631</td>
<td>900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>94.4</td>
<td>60.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Traditional</td>
<td>1</td>
<td>52.1</td>
<td>58.4</td>
<td>794</td>
<td>900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>96.3</td>
<td>81.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SFMA</td>
<td>1</td>
<td>98.6</td>
<td>54.3</td>
<td>440</td>
<td>600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>99.3</td>
<td>54.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>99.1</td>
<td>54.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Traditional</td>
<td>1</td>
<td>99.7</td>
<td>88.7</td>
<td>778</td>
<td>600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>99.2</td>
<td>88.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>52.4</td>
<td>58.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance Evaluation

Performance evaluation of the proposed decentralized scheduling mechanism based on (SFMA)

1. Parallel Execution Scheduling performance for 2400x2400 matrix size

<table>
<thead>
<tr>
<th>Number Of subtasks</th>
<th>Scheduling Mechanism</th>
<th>Subtask index</th>
<th>%CPU utilization</th>
<th>%Memory utilization</th>
<th>Execution time (Seconds)</th>
<th>NTU(T) (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>SFMA</td>
<td>1</td>
<td>99.2</td>
<td>87.55</td>
<td>377.8</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>98.4</td>
<td>87.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>97</td>
<td>88.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>60.8</td>
<td>98.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Traditional</td>
<td>1</td>
<td>96.5</td>
<td>86.54</td>
<td>575</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>99.5</td>
<td>87.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>98</td>
<td>87.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>59.3</td>
<td>71.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Performance Evaluation

**Performance evaluation of the proposed decentralized scheduling mechanism based on (SFMA)**

1. **Parallel Execution Scheduling performance for 2400x2400 matrix size**

<table>
<thead>
<tr>
<th>Number Of subtasks</th>
<th>Scheduling Mechanism</th>
<th>Subtask index</th>
<th>%CPU utilization</th>
<th>%Memory utilization</th>
<th>Execution time (Seconds)</th>
<th>NTU(T) (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>SFMA</td>
<td>1</td>
<td>99.33</td>
<td>55.5</td>
<td>354</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>98.9</td>
<td>88.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>79.5</td>
<td>87.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>64.3</td>
<td>76.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>50.9</td>
<td>56.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Traditional</td>
<td>1</td>
<td>58.6</td>
<td>72.3</td>
<td>502</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>99.8</td>
<td>85.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>51.5</td>
<td>55.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>98.5</td>
<td>87</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>92.7</td>
<td>87.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Performance Evaluation

Performance evaluation of the proposed decentralized scheduling mechanism based on (SFMA)

1. Parallel Execution Scheduling performance for 2400x2400 matrix size

<table>
<thead>
<tr>
<th>Number Of subtasks</th>
<th>Scheduling Mechanism</th>
<th>Subtask index</th>
<th>%CPU utilization</th>
<th>%Memory utilization</th>
<th>Execution time (Seconds)</th>
<th>NTU(T) (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>SFMA</td>
<td>1</td>
<td>99.5</td>
<td>87.3</td>
<td>319</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>98.7</td>
<td>89.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>98.8</td>
<td>88</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>72.4</td>
<td>79.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>51.3</td>
<td>57.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>50.6</td>
<td>84.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Traditional</td>
<td>1</td>
<td>99.8</td>
<td>89.6</td>
<td>439</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>99.7</td>
<td>87.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>99</td>
<td>73.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>99.2</td>
<td>88.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>51.3</td>
<td>57.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>98.4</td>
<td>87.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>