CHAPTER 5: DISTRIBUTED PROCESS SCHEDULING

Chapter outline

- Three process models: precedence, communication, and disjoint
- A system performance model that illustrates the relationship among algorithm, scheduling, and architecture
- Static scheduling: precedence and communication models
- Dynamic scheduling: load sharing and balancing for disjoint and interacting process models
- Implementation: remote service and execution, and process migration
- Real-time scheduling and synchronization

Process models

(a) Precedence process model
(b) Communication process model
(c) Disjoint process model
A system performance model

Speed-up factor

\[ S = F(Algorithm, System, Schedule) \]

\[ S = \frac{OSPT}{CPT} = \frac{OSPT}{OCPT_{ideal}} \times \frac{OCPT_{ideal}}{CPT} = S_i \times S_d \]

Ideal speed-up

\[ S_i = \frac{RC}{RP} \times n \]

\[ RP = \frac{\sum_{i=1}^{m} P_i}{OSPT} \]

\[ RC = \frac{\sum_{i=1}^{m} P_i}{OCPT_{ideal} \times n} \]

System degradation

\[ S_d = \frac{1}{1 + \rho} \]

\[ \rho = \frac{CPT - OCPT_{ideal}}{OCPT_{ideal}} \]

Finally

\[ S = \frac{RC}{RP} \times \frac{1}{1 + \rho} \times n \]
Static scheduling

Precedence process model

(a) Precedence process model

(b) Communication system model

(a) LS

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/6</td>
<td>D/6</td>
<td>G/4</td>
<td></td>
</tr>
<tr>
<td>B/5</td>
<td>F/4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>C/4</td>
<td>2</td>
<td>E/6</td>
<td>4</td>
</tr>
</tbody>
</table>

Makespan = 16

(b) ELS

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/6</td>
<td>D/6</td>
<td>10</td>
<td>G/4</td>
</tr>
<tr>
<td>B/5</td>
<td>F/4</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>C/4</td>
<td>10</td>
<td>E/6</td>
<td>8</td>
</tr>
</tbody>
</table>

Makespan = 28

(c) ETF

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/6</td>
<td>E/6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>B/5</td>
<td>D/6</td>
<td>14</td>
<td>G/4</td>
</tr>
<tr>
<td>C/4</td>
<td>4</td>
<td>F/4</td>
<td>6</td>
</tr>
</tbody>
</table>

Makespan = 18
Communication process model

<table>
<thead>
<tr>
<th>Process</th>
<th>Cost on A</th>
<th>Cost on B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>infinity</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>infinity</td>
<td>4</td>
</tr>
</tbody>
</table>

(a) Computation cost

(b) Communication cost

Cut Set Cost = 38
Dynamic scheduling

Load sharing and balancing

Sender initiated algorithms

- **transfer policy**: When does a node become the sender?
- **selection policy**: How does the sender choose a process for transfer?
- **location policy**: Which node should be the target receiver?

![Flowchart for Sender Initiated Algorithms]

Receiver initiated algorithms

Similar policies but require preemption

Hybrid algorithms and their performance
Remote process implementation

Remote service

- As remote procedure calls at the language level
- As remote commands at the operating system level
- As interpretive messages at the application level

Remote execution

The remote operation initiated by a client is created by the client for resource or load sharing (processor-pool model).

- Load sharing algorithm
- Location independence
- System heterogeneity
- Protection and security
Process migration
Preemption and reconfiguration

link redirection and message forwarding

state and context transfer
freeze time and residual computation dependency
Real-time scheduling

Soft/hard deadlines, periodic/aperiodic, priority scheduling

Priority scheduling

- Rate monotonic priority assignment: task period
- Deadline monotonic priority assignment: deadline
- Earliest deadline first: dynamic deadline

Real-time synchronization

Priority inversion

\[ \tau_1 \]

\[ \tau_2 \]

\[ \tau_3 \]

Chain blocking

\[ \tau_1 \]

\[ \tau_2 \]

\[ \tau_3 \]

Protocols to reduce priority inversion and blocking: PIP and PCP