Real Time Scheduling for Multi-Processor and Distributed Systems

Corso di Sistemi RT
Prof. Davide Brugali
Università degli Studi di Bergamo

Based on: Singhoff, Real-Time Scheduling analysis, University of Brest
Definitions

- **Distributed systems:** "A distributed system is a set of autonomous processors that are connected by a network and which have software to coordinate them self or share resources."
  - No shared memory.
  - Message passing communication.
  - A clock for each processor.

- **Multiprocessors systems:** a multiprocessors system is a set of autonomous processors which have software to coordinate them self and share resources but share the same clock and the same main memory.

- **Two scheduling approaches:**
  - Global scheduling.
  - Partitioning.
Scheduling approaches

- **Global scheduling**: choose a task, and assign it to one of the idle processors (otherwise, preempt one of the running task). On-line processor assignment. With migration.
- **Partitioning**: choose a processor for all tasks, and then run local scheduler on each processor. No migration. Off-line processor assignment.
Global scheduling

• Few theoretical results (feasibility tests) compared to mono-processor real-time scheduling.
• We can expect optimal processor usage: busy processors, less preemptions … but task migrations.
• Difficult to apply to heterogeneous systems: task migration, hardware, operating systems.
• Well suited for multiprocessors architectures.
Partitioning

• Theoretical foundations of mono-processor scheduling compliant with current system implementation.
• Non optimal resource sharing: a processor may stay idle, even if a task waits for a processor elsewhere in the system.
• Better reliability, deterministic behavior in case of failure: failures of a task may only imply failure on its processor.
• Partitioning is a NP-hard problem: Bin-packing.
• Well suited for synchronous distributed systems (e.g. aircraft, ARINC 653).
A global scheduler has to solve two problems:
• When and how to choose the processor to run tasks or jobs.
• When and how to assign priority to tasks or jobs.

When migrations occur:

• **No migration**: a task must always run on a given processor => partitioning.

• **Job level migration**: each job of a task can be run on any processor. But a job started on a given processor can not be moved on another.

• **Task level migration**: a task can run at any time on any processor.
Global scheduling

Two types of scheduling algorithms:
- Adapt mono-processor scheduling algorithms:
  - Global RM, global EDF, global DM, global LLF, …
  - Choose the level of migration.
  - Apply the scheduling algorithm on all the set of the processors.
  - At each time, assign m highest priority tasks to the m processors.
  - Preemptions occur when a job/task has to be run since its priority is high and when all processors are busy.

- New algorithms: PFair, LLREF, EDF(k), SA, EDZL, …
**Global scheduling**

- **Example:** global Deadline Monotonic

<table>
<thead>
<tr>
<th></th>
<th>Ci</th>
<th>Pi</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>T2</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>T3</td>
<td>7</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Priority assignment: $T_1 > T_2 > T_3$.

Job level migration.
Global scheduling

- Example: global Deadline Monotonic

<table>
<thead>
<tr>
<th></th>
<th>Ci</th>
<th>Pi</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>T2</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>T3</td>
<td>7</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

- Priority assignment: $T_1 > T_2 > T_3$.
- Task level migration.
**Partitioned systems**

How to statically assign a set of tasks to a set of processors?

• "bin-packing" problem: find how to put a set of object in identical boxes. We expect to minimize the required number of boxes.
• NP-hard problem => heuristic, usually based on feasibility tests.

**Many parameters:**

• Processors (identical or not), tasks (priority, period, capacity).
• Take into account task communications, shared resources,…

**Examples of objective functions:**

• Minimize the number of processors, the number of communications, latencies, …
Skeleton of a partitioning heuristic:

• Order tasks according to a given parameter (e.g., period, priority, processor utilization).
• Do task assignment according to their order.
• For each task, look for an available processor, according to a policy (e.g., Best Fit, Next Fit, ...).
• An available processor is a processor that is able to run a task with no missed deadline: processors are selected according to a feasibility test.
• Stop when all tasks are assigned.
Example of the Rate Monotonic Next Fit:

- Tasks are increasingly ordered by their periods.
- We start with task $i = 0$ and processor $j = 0$.
- Assign task $i$ on processor $j$ if the feasibility tests is true (e.g. $U \geq 0.69$).
- Otherwise, put task $i$ on processor $j + 1$.
- Assign the next task $i + 1$.
- Stop when all tasks have been assigned. $j =$ the number of required processors.