Capítulo 6

Escalonamento de tempo-real em ambientes multiprocessados

Os algoritmos apresentados até agora consideram que todas as tarefas serão executadas em apenas um processador. Isto é verdade em muitos sistemas, mas com o avanço de redes e microeletrônica temos hoje muitos ambientes que permitem a execução em vários processadores.

Para sistemas do tipo *multicore* ou *manycore* é possível usar os mesmos escalonadores vistos. Isso porque tais sistemas fazem uso de memória compartilhada, com todos os núcleos de processamento acessando os mesmos dados, exceto caches locais.

Para sistemas distribuídos é necessário definir outros modelos de escalonamento, em que se considere o impacto da localização das CPUs, recursos e tarefas.

No tratamento de sistemas de tempo-real executados em ambientes distribuídos é preciso definir:

i) Como as tarefas chegam ao sistema;

ii) Que mecanismos de comunicação e migração estão disponíveis;

iii) Que tipo de paralelismo é possível entre as tarefas.

Assim, podemos ter sistemas em que cada máquina tem sua própria fila de chegada ou aqueles em que a fila de chegada é centralizada. Podemos também ter sistemas com ou sem migração de tarefas entre máquinas. Por fim, podemos ter sistemas em que as tarefas podem ser particionadas e outros em que são monolíticas. Isso implica em dezenas de tratamentos possíveis para o escalonamento de tarefas. Examinaremos aqui apenas algumas dessas políticas, procurando tratar várias das diferenças apontadas neste parágrafo.

6.1 Algoritmo Míope

É um algoritmo simples, que considera tarefas monolíticas, sem preempção ou migração, e que cada máquina tem sua própria fila de tarefas. O seu funcionamento é baseado no escalonamento de subconjuntos de tarefas da fila de entrada, usando uma função pré-definida, servindo-se de *backtracking* quando o atendimento de *deadlines* não for obtido. Seu procedimento básico, para uma dada máquina, é:

1. Máquina i tem N_i tarefas em sua fila;
2. As N_i tarefas são ordenadas segundo seu *deadline*;
3. As primeiras K tarefas dessa lista são selecionadas;
4. Essas K tarefas são reordenadas segundo uma função H, usando-se *backtracking* se necessário;
5. A primeira tarefa dessa lista é alocada a CPU;
6. Volta-se ao passo 3 enquanto existirem tarefas para alocar.

6.2 Algoritmos Leilão e Leilão Focado

Usam política semelhante ao algoritmo Míope, porém permitindo migração no lugar de aplicar o *backtracking*. A migração é feita por leilão, ou seja, se uma máquina identifica que uma tarefa em sua fila perderá o *deadline*, então requisita das demais máquinas a folga que teriam para atender essa tarefa. A máquina com maior folga (lance) ganha o leilão, recebendo a tarefa para executar.

A diferença entre Leilão e Leilão Focado é que neste último o leilão ocorre apenas entre um conjunto pré-definido de máquinas.

6.3 Algoritmo de Particionamento

Apresentado por Manimaran e Murthy, o algoritmo de Particionamento tem como objetivo o escalonamento de tarefas de tempo-real com apenas uma lista de tarefas, chamada lista global de tarefas. As tarefas dessa lista devem ser distribuídas entre as N máquinas do sistema, de modo a atender todos os *deadlines* e manter a carga das máquinas balanceadas.

A principal característica deste algoritmo, comparado com o Míope, é que quando uma tarefa não é escalonável inteira, ocorre o seu particionamento entre duas os mais máquinas, ao invés de se fazer o *backtracking*. Esse particionamento

significa que o tempo para a execução da tarefa agora será dividido entre as máquinas que serão usadas. A nova carga da tarefa i, particionada em j máquinas, será dada por:

$$C_i^j = \lfloor C_i^{j-1} * \frac{(j-1)}{j} \rfloor + 1 \tag{6.1}$$

Ainda com relação às tarefas particionadas, é importante ressaltar que elas devem ser iniciadas no mesmo instante nas diferentes máquinas, devido à necessidade de sincronismo entre elas. Esse instante de início precisa ser considerado no momento de calcular o tempo de finalização da tarefa. Deve ficar claro que vai existir um compromisso entre o número de partições e esse instante de início, uma vez que quanto mais partições será também necessário esperar mais tempo para iniciar a execução.