Escalonamento Adaptativo (p+i,k)-firm

Carlos Montez montez@lcmi.ufsc.br
Joni da Silva Fraga fraga@lcmi.ufsc.br
Rômulo de Oliveira romulo@inf.ufrgs.br

LCMI - Depto de Automação Sistemas - Univ. Fed. de Santa Catarina
Caixa Postal 476 - 88040-900 - Florianópolis - SC - Brasil

Resumo

Recentemente, existe um interesse crescente em modelos de sistemas de tempo real e abordagens flexíveis de escalonamento para lidar com situações de sobrecarga transientes, que podem ocorrer em ambientes não deterministas. Este artigo apresenta a abordagem de escalonamento (p+i,k)-firm, voltada sistemas de tempo real. A abordagem lida com situações de sobrecarga através da combinação da técnica da computação imprecisa com a garantia (m,k)-firm.

Abstract

Recently, there is an interest in establishing flexible real-time models and approaches to deal with overload in environments that do not offer predictability. This article presents the (p+i,k)-firm adaptive scheduling approach for distributed real-time applications. The approach deals with overload conditions, aiming a graceful degradation, through the combination of the imprecise computation technique with the (m, k)-firm guarantee.

1. Introdução

Apesar da quantidade significativa de resultados obtidos nas pesquisas sobre escalonamento nesse último quarto de século, boa parte dos problemas de tempo real não são facilmente suportados. Alguns exemplos citados na literatura [9] incluem sistemas de banco de dados de tempo real, manufatura flexível, robótica, aplicações militares de comando e controle, os quais precisam lidar com eventos dinâmicos e desconhecidos.

O ambiente que um sistema computacional de tempo real opera é de fundamental relevância na previsibilidade obtida e no desempenho do próprio sistema. Ambientes não deterministas, ao contrário dos ambientes deterministas, dão origem a sistemas dinâmicos, onde não é possível um levantamento prévio do comportamento da carga de trabalho. Esses dois ambientes de sistema de tempo real, fundamentalmente distintos, demandam diferentes modelos de sistema e estratégias de implementação.

De uma forma geral, os novos modelos e estratégias para lidar com esses ambientes não deterministas precisam também possuir características dinâmicas, adaptativas e flexíveis. No sentido de melhor tratar as situações de sobrecarga, precisam possuir um ciclo fechado, isto é, monitorar as condições do ambiente (mudanças no comportamento da carga, perdas de deadlines, *etc.*) e o resultado deve ser usado para realimentar e ajustar continuamente a abordagem de escalonamento, objetivando obter um funcionamento mais estável e robusto. Técnicas de escalonamento que seguem esse princípio fazem parte do que é chamado escalonamento adaptativo.

Este artigo apresenta a abordagem de escalonamento adaptativo (p+i,k)-firm [8], cujas decisões são baseadas nos estados das últimas ativações (precisas, imprecisas, ou perdas de deadline) de cada tarefa. A próxima seção traz um levantamento dos principais trabalhos relacionados à técnica do escalonamento adaptativo. A abordagem de escalonamento descrita neste artigo é fundamentada no conceito da garantia (p+i,k)-firm, cujos principais aspectos são descritos na seção 3. A seção 4 apresenta alguns resultados da validação feita na abordagem de escalonamento. Finalmente, a seção 5 apresenta as principais conclusões.

2. Trabalhos relacionados

Diversas técnicas adaptativas têm sido propostas recentemente.

2.1. Alteração nas freqüências das tarefas

Tarefas periódicas de sistemas de tempo real geralmente possuem faixas de valores para

períodos dentro das quais operam de maneira satisfatória. Exemplos são encontrados em sistemas de controle com malha fechada, onde um melhor desempenho é fornecido quando maiores taxas de amostragem são usadas nesses sistemas, e em sistemas distribuídos multimídia que também fornecem melhor qualidade quanto maior for a freqüência de amostragem ou apresentação de quadros de áudio ou vídeo. As especificações de QoS de tais sistemas usualmente são colocadas na forma de intervalos de freqüência aceitáveis. Diversas abordagens adaptativas [6] propõem modelos de tarefas periódicas, nos quais os períodos das tarefas podem ser alterados dinamicamente, com objetivo de se adaptarem às condições do ambiente e evitar sobrecargas transientes do sistema.

2.2 Computação imprecisa

A técnica da computação imprecisa [7] surgiu como uma forma de lidar com situações dinâmicas, onde os recursos disponíveis possam ser temporariamente insuficientes para que todas as tarefas tenham seus deadlines satisfeitos. A idéia básica é a da produção de resultados parciais, usando menos tempo e recursos. Conceitualmente, cada tarefa é decomposta em duas partes: uma parte obrigatória e uma parte opcional. A parte obrigatória de uma tarefa precisa ser completada antes do deadline da tarefa para se produzir um resultado aproximado com uma qualidade aceitável. A parte opcional refina o resultado produzido pela parte obrigatória. Essa técnica permite a combinação de garantia determinista com degradação suave. Durante uma sobrecarga, um nível "mínimo" de operação do sistema pode ser garantido, através da execução apenas das partes obrigatórias.

Quando são executadas as duas partes (obrigatória e opcional), diz-se que ocorreu uma *execução precisa*; e quando é executada apenas a parte obrigatória, diz-se que ocorreu uma *execução imprecisa*. Execuções imprecisas consecutivas por uma tarefa podem produzir erros que se acumulam a cada execução. Esse tipo de erro foi estudado por Chung [3] com o nome de *erro cumulativo*. Assim, uma heurística de escalonamento foi apresentada para manter o erro cumulativo de uma tarefa periódica abaixo de um certo valor limite. Essa heurística garantia a ocorrência de, pelo menos, uma execução *precisa* a cada janela de Q ativações sucessivas da tarefa.

2.3 Garantia (m,k)-firm

Alguns trabalhos [1,2,4,5] estabeleceram modelos de tarefas que permitiam a tarefas periódicas perderem seus deadlines sem falhar. Todos esses trabalhos partem do princípio que a falta temporal, resultante da perda do deadline de uma ativação, pode ser recuperada se a tarefa tiver seu deadline satisfeito em uma das suas próximas ativações. Caso isso não ocorra, uma falha temporal é assumida no sistema.

Hamdaoui [4] propôs o conceito de *deadline (m,k)-firm*, definindo que uma tarefa periódica deva ter pelo menos m deadlines atendidos a cada janela de k ativações. O limite superior tolerado de perdas de deadlines é dado por k-m. Uma *falha dinâmica* é assumida no sistema quando esse limite é excedido. Um deadline especificado na forma (m,k)-firm é capaz de representar vários tipos de deadlines. Deadlines *hard* podem ser representados na forma (1,1)-firm, enquanto deadlines *soft* podem ser representados com valores k e m escolhidos de forma que $(k-m)/k$ seja próximo de 1. Apesar da capacidade de representar deadlines *hard*, a proposta original em [4] é a de empregar uma abordagem de escalonamento para ambientes dinâmicos, onde seja possível a ocorrência de sobrecargas transientes. A heurística de escalonamento DBP (*Distance Based Priority Assignment*) é empregada, objetivando minimizar o número de falhas dinâmicas, atribuindo as mais altas prioridades para as tarefas que estão próximas de exceder o limite superior especificado para as perdas de deadlines.

2.4 Descartes de ativações de tarefas periódicas

Koren [5] também estudou um modelo de tarefas periódicas, denominado *skip-over*, que

tolera perdas de deadlines ou descartes de ativações. Cada tarefa do modelo é caracterizada por um parâmetro de descarte *s*, que representa sua tolerância em perder deadlines, especificando uma distância mínima entre descartes (ou perdas de deadline) de ativações periódicas de uma tarefa. Uma tarefa com parâmetro de descarte *s*, após perder um deadline, precisa ter seus deadlines satisfeitos nas próximas *s - 1* ativações. Mais recentemente duas outras abordagens semelhantes foram propostas por [1] e [2]. Esses trabalhos têm em comum o fato de aproveitarem a flexibilidade de tarefas periódicas poderem ter eventualmente suas ativações descartadas, para aumentar a escalonabilidade do sistema.

2.5 Comentários sobre técnicas adaptativas

Cada técnica adaptativa apresenta características próprias. Na técnica de alteração de freqüência das tarefas, ao se variar a freqüência de uma tarefa pode ser necessário mudar as freqüências das tarefas relacionadas (predecessoras e sucessoras) no mesmo subsistema. Essa é uma característica indesejável em sistemas distribuídos por implicar em trocas de mensagens de controle extras entre as tarefas.

As técnicas que envolvem descartes de ativações, alterações nas freqüências e execuções imprecisas de tarefas, apresentam como característica a redução na carga do sistema, o que é desejável em situações de sobrecarga. Entretanto, em determinados momentos podem reduzir a qualidade até de tarefas que poderiam ser escalonadas. Esse problema não ocorre na abordagem de escalonamento (m,k)-firm. Sua heurística de escalonamento (DBP) não reduz a qualidade das tarefas, apenas distribui a capacidade disponível de processador de forma mais inteligente, priorizando as tarefas que estão mais próximas de exceder um limite especificado de perdas de deadlines.

Por outro lado, o fato da heurística DBP não reduzir a carga pode ser visto como uma desvantagem em determinadas situações onde um sistema tenha que conviver constantemente com situações de sobrecargas. Nesse caso, pode ser desejável a redução imediata da carga, e evitar perdas de deadlines, mesmo que de alguma forma a qualidade do sistema seja degradada.

A próxima seção descreve a abordagem de escalonamento (p+i,k)-firm que integra as técnicas da computação imprecisa com a garantia (m,k)-firm. A idéia básica é, em situações de sobrecarga, procurar distribuir as perdas de deadline entre as tarefas, e, simultaneamente, reduzir a carga através de execuções imprecisas das tarefas.

3. Garantia (p+i,k)-firm

A abordagem de escalonamento descrita neste artigo baseia-se em um modelo de tarefas, onde cada tarefa pode ter associado um *deadline (p+i,k)-firm*. Uma tarefa possui um requisito de *garantia (p+i,k)–firm* quando, para oferecer uma qualidade satisfatória, a cada *k* ativações consecutivas, pelo menos *(p+i)* ativações precisam ter seus deadlines satisfeitos; e dentre essas últimas, pelo menos *p* precisam ocorrer na forma precisa. O deadline (p+i,k)-firm é uma extensão do deadline (m,k)-firm [4] que inclui algumas propriedades da computação imprecisa [3].

A abordagem de escalonamento é materializada por uma heurística de escalonamento que implementa e integra duas políticas: de escolha de versão (precisa ou imprecisa) e de atribuição de prioridades. A seguir, o modelo de tarefas será visto com mais detalhes.

3.1 Modelo de tarefas e o deadline (p+i,k)-firm

O escalonamento adaptativo proposto na abordagem *(p+i,k)-firm* usa um modelo de tarefas que permite execuções precisas e imprecisas de tarefas. Nesse modelo de tarefas, é definido o conceito do *deadline (p+i,k)-firm*, onde em qualquer janela de *k* ativações consecutivas de uma tarefa periódica com deadline (p+i,k)-firm, as seguintes propriedades precisam ser verificadas:

(i) no máximo $k-(p+i)$ deadlines podem ser perdidos; e
(ii) pelo menos p execuções devem ser precisas, dentre as ativações da tarefa que possuem seus deadlines satisfeitos.

O limite inferior tolerado para o número de ativações com deadlines satisfeitos é dado por $(p+i)$. O valor i, por sua vez, representa o limite inferior para o número de execuções imprecisas com deadlines satisfeitos em uma janela de k invocações, quando apenas p execuções precisas são completadas.

A condição (i) acima limita o número máximo de deadlines perdidos, e a condição (ii) especifica o número mínimo aceitável de execuções precisas. Ambas as condições não podem ser garantidas porque é assumido um ambiente dinâmico e não determinista. Entretanto, da mesma forma que ocorre na heurística DBP do (m,k)-firm, a política de atribuição de prioridades irá garantir as maiores prioridades para tarefas próximas de violar a condição (i). Além disso, a heurística de escalonamento, através de sua política de aceitação de precisão, nunca irá liberar menos que p execuções precisas em quaisquer janelas de k invocações.

O conceito de falha dinâmica descrito em [4] é relacionado somente com a condição (i). Na abordagem de escalonamento (p+i,k)-firm esse conceito é estendido considerando também a condição (ii). Portanto:

uma tarefa experimenta uma *falha dinâmica* quando, em uma janela de k ativações, o número de deadlines não atendidos superar a $k-(p+i)$, ou quando ocorrerem menos que p execuções precisas.

O modelo de tarefas do deadline (p+i,k)-firm representa uma extensão do deadline (m,k)-firm com execuções precisas e imprecisas. Permite a uma tarefa executar na forma imprecisa, produzindo resultados aproximados e tendo um erro acumulado menor que uma perda de deadline. Além disso, se apenas p execuções precisas de uma tarefa em uma janela de k invocações oferecem uma qualidade aceitável, então o escalonador passa a ter um outro nível de flexibilidade para tratar sobrecargas: execuções imprecisas representam um tempo de computação menor, e o escalonador pode usá-las para reduzir a carga e controlar o número de perdas de deadlines de um conjunto de tarefas.

3.2 Políticas de escalonamento

No modelo de escalonamento proposto existe um grupo de tarefas competindo por uma CPU. Cada tarefa tem um deadline (p+i,k)-firm, e cada ativação (invocação) de tarefa possui um valor de deadline absoluto. O objetivo principal do escalonador do sistema é ordenar a execução das tarefas tentando evitar falhas (temporais) dinâmicas.

Conceitualmente, filas FIFO são definidas para armazenar invocações de tarefas (Figura 1). A cada tarefa j do sistema é atribuída uma fila de chegada, assegurando que invocações de uma mesma tarefa serão servidas pelo escalonador na ordem de suas chegadas. Somente invocações de tarefas no topo de suas filas são candidatas a serem servidas, sendo inseridas em uma fila de pronto de acordo com uma *política de atribuição de prioridades*. Além dessa atribuição de prioridades, uma *política de aceitação de precisão* também é adotada. Essa última política selecionará qual versão (precisa ou imprecisa) da tarefa será executada.

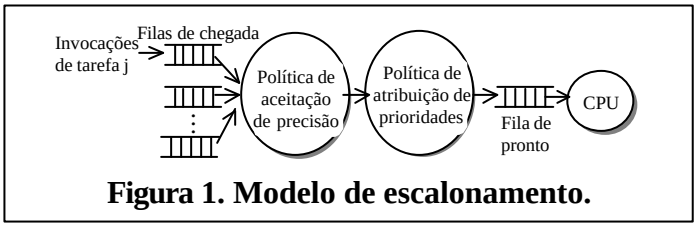

Figura 1. Modelo de escalonamento.

Nesse modelo de escalonamento, as políticas de aceitação de precisão e de atribuição de prioridades trabalham de uma forma integrada, selecionando a invocação na versão precisa ou imprecisa, e atribuindo sua prioridade.

3.3 Histórico de execução e autonomias para execução

A abordagem de escalonamento fundamenta-se em um histórico de execuções precisas, imprecisas e de perdas de deadlines. Um *histórico de execução* de uma tarefa é uma k-tupla que armazena o estado das últimas *k* invocações da tarefa.

Seja a seguinte representação para cada estado de invocação: *P* para execução precisa, *I* para execução imprecisa, e *X* para perda de deadline. Para cada novo estado produzido, o histórico é deslocado (da direita para esquerda) e o novo estado adicionado na posição mais à direita. Por exemplo, uma tarefa com um histórico *"PIPP"*, ao perder seu deadline em sua última invocação, passaria a ter um histórico *"IPPX"*.

Dois conceitos importantes na abordagem de escalonamento (p+i,k)-firm são: *autonomia para perda de deadlines* e *autonomia para execução imprecisa*. O primeiro especifica o número de deadlines consecutivos que uma tarefa pode perder, em uma janela de ativações da tarefa, sem que ocorra uma falha dinâmica; e o segundo especifica o número de invocações imprecisas consecutivas que uma tarefa tolera na mesma janela.

A heurística de escalonamento tenta reduzir o erro acumulado através de suas políticas que são guiadas pelos cálculos dinâmicos das autonomias para execução. A política de atribuição de prioridades é dirigida pela *autonomia para perda de deadline*; e a política de aceitação de precisão é dirigida pela *autonomia para execução imprecisa*. Em outras palavras, durante uma condição de sobrecarga transiente:

a política de atribuição de prioridades redefine prioridades, tentando evitar perdas de deadlines das tarefas que têm menor autonomia para perder deadlines; e a política de aceitação de precisão seleciona invocações de tarefas na forma imprecisa, tentando reduzir a carga, em tarefas que têm maior autonomia para executar suas versões imprecisas.

3.4 Política de atribuição de prioridades

Seja $d_j(k)$ a função que retorna o valor da *autonomia para perda de deadlines* de uma tarefa *j*. A interpretação desse valor é apresentada na Tabela 1.

Tabela 1. Autonomia para perda de deadline.

$d_j(k)$	Interpretação
0	Estado de falha
1	Se perder próximo deadline então falha
2	Se perder os próximos 2 deadlines então falha
⋮	
n	Se perder os próximos n deadlines então falha

É possível obter-se $d_j(k)$ através de um cálculo similar ao introduzido na heurística DBP. Seja $met_j(n,h)$ a função que denota para uma tarefa *j*, a posição (da direita para esquerda) da $n^{ésima}$ execução com deadline atendido (na forma precisa ou imprecisa) no histórico *h*. Se existirem menos que *n* deadlines atendidos em *h*, então essa função retorna *k+1*. Por exemplo, $met_j(1,\text{“XPP"}) = 1$, $met_j(1,\text{“XPX"}) = 2$, $met_j(2,\text{“IXP"}) = 3$, $met_j(2,\text{“XXP"}) = 4$. Usando essa função, o valor da autonomia para perda de deadline pode ser calculado por: $d_j(k) := k - met_j(p+i,h) + 1$

Por exemplo, uma tarefa *j* declarada com um deadline (2+0,4)-firm e um histórico *"PPXX"* teria: $d_j(k) = 4 - met_j(2, \text{“PPXX"}) + 1 = 4 - 4 + 1 = 1$, indicando que se essa tarefa perder o próximo deadline irá falhar. Entretanto, se o histórico dessa tarefa fosse *"XPXP"*: $d_j(k) = 4 - met_j(2, \text{“XPXP"}) + 1 = 4 - 3 + 1 = 2$, indicaria que essa tarefa poderia ainda perder o próximo deadline sem falhar.

Na heurística de escalonamento, o valor de $d_j(k)$ é aplicado diretamente, através do seu mapeamento nas prioridades nativas do sistema operacional. Por exemplo, supondo que a prioridade mais alta é 0, para qualquer ativação de uma tarefa *j* é suficiente fazer: $prioridade_j := d_j(k)$

3.5 Política de aceitação de precisão

Seja $v_j(k)$ a função que representa o valor da *autonomia para execução imprecisa* da tarefa j. A interpretação desse valor é apresentada na Tabela 2. O cálculo de $v_j(k)$ usa uma função $metp_j(n,h)$ que denota, para uma tarefa j, a posição do $n^{ésimo}$ deadline atendido com uma execução precisa no histórico h. Se existirem menos que n execuções precisas em h, então essa função retorna $k+1$. Por exemplo, $metp_j(1,"XIP") = 1$, $metp_j(1,"XPI") = 2$, $metp_j(2,"PXP") = 3$, $metp_j(2,"IXP") = 4$. Usando essa função, o valor da autonomia para execução imprecisa pode ser calculado como: $v_j(k) := k - metp_j(p,h) + 1$

Por exemplo, uma tarefa j declarada com um deadline (2+2,4)-firm e com um histórico *"PPII"* teria: $v_j(k) = 4 - metp_j(2, "PPII") + 1 = 4 - 4 + 1 = 1$, indicando que a próxima execução não pode ser na forma imprecisa. Entretanto, se o histórico dessa tarefa fosse *"IPIP"*: $v_j(k) = 4 - metp_j(2, "IPIP") + 1 = 4 - 3 + 1 = 2$, indicando que a tarefa poderia ainda ter uma execução imprecisa.

Tabela 2. Autonomia para execução imprecisa.

$v_j(k)$	Interpretação
0	Estado de falha
1	Se executar uma invocação na forma imprecisa então falha
2	Se executar as próximas 2 invocações na forma imprecisa então falha
⋮	
n	Se executar as próximas n invocações na forma imprecisa então falha

Na heurística de escalonamento, a principal função da política de aceitação de precisão é decidir, em cada tarefa, a versão precisa ou imprecisa para a próxima invocação. Portanto, para cada tarefa do sistema, existe uma variável ($next\_exec_j$) que armazena essa informação.

Como a abordagem de escalonamento não pode prever futuras perdas de deadline, a política de aceitação de precisão é dirigida pelas ocorrências de sobrecarga. A cada indicação de uma perda de deadline no conjunto de tarefas do sistema, uma invocação de tarefa é selecionada para executar sua versão imprecisa (reduzindo sua qualidade), tomando como base a sua autonomia para execução imprecisa.

A política de aceitação de precisão seleciona a invocação de tarefa, na cabeça de uma das filas de chegada, que possua o maior valor de autonomia para execução imprecisa ($v_j(k)$) e cujo $next\_exec_j$ especifica versão precisa (*i.e.*, $next\_exec_j$ = "precisa" e $\max(v_j(k))$). A versão imprecisa dessa tarefa é então selecionada para executar na próxima e nas invocações seguintes ($next\_exec_j$:= "imprecisa"), até que a situação crítica indicada pela autonomia para execução imprecisa é alcançada ($v_j(k)=1$; isto é, a tarefa não pode executar sua versão imprecisa na próxima ativação, de outra forma uma falha dinâmica ocorrerá). Quando essa última situação é alcançada, a tarefa é assinalada para executar sua versão precisa novamente ($next\_exec_j$:= "precisa"). Uma descrição mais detalhada da heurística de escalonamento pode ser encontrada em [8].

4. Validação da abordagem de escalonamento

A abordagem de escalonamento (p+i,k)-firm foi avaliada através de simulações. Sempre que possível, a abordagem de escalonamento foi comparada com as abordagens (m,k)-firm e EDF. Com esse objetivo, foi desenvolvido um simulador que permite a configuração do número de tarefas do sistema, os tempos de chegadas de suas ativações e suas restrições temporais. Além dos deadlines (p+i,k)-firm, as outras restrições temporais necessárias para a simulação do sistema são: os tempos de execução das versões precisa e imprecisa, os períodos e os valores de deadlines relativos. Todas essas restrições temporais podem ser configuradas para o simulador, antes de cada conjunto de experimentos. (Deve-se notar que os tempos máximos de execução são necessários apenas para conduzir as simulações, pois são desconsiderados pelas abordagens, que lidam somente com valores de deadlines.)

Durante as simulações, diversos dados foram contabilizados, tais como o total de ativações de

tarefas, de perdas de deadlines, e de falhas dinâmicas. Também foram contabilizados os totais de ativações de tarefas que têm seus deadlines satisfeitos nas formas precisas e imprecisas. Todos esses dados são utilizados para avaliar a abordagem de escalonamento, considerando as duas principais métricas aferidas nos experimentos: (i) a probabilidade de falhas dinâmicas, e (ii) o valor cumulativo obtidos.

Uma abordagem de escalonamento para o modelo de tarefas adotado, além de reduzir a probabilidade de falhas dinâmicas, deveria também, simultaneamente, maximizar a quantidade de ativações de tarefas com deadlines satisfeitos e, dentre essas, maximizar o número de execuções precisas. Com esse objetivo a métrica *valor cumulativo* [10] foi introduzida. Considera-se que uma tarefa quando perde seu deadline, ou executa na forma imprecisa, reduz sua qualidade (ou seu benefício) para o sistema. A forma encontrada de mensurar isso nas simulações foi estabelecendo-se que a qualidade de cada tarefa é proporcional ao tempo despendido na sua execução. A menos que a tarefa perca seu deadline, já que nesse caso considera-se que a qualidade obtida é zero.

Portanto, considerou-se nas simulações que, para cada término de ativação de uma tarefa T_j do sistema, um *valor cumulativo* Q é adicionado ao sistema tal que: Q = 0, quando T_j perde deadline; ou $Q = C_j$ quando T_j executa completamente (onde C_j representa o tempo gasto na execução (da versão precisa e imprecisa) da tarefa T_j.

4.1 Valores obtidos

A Figura 2 mostra alguns resultados obtidos, variando-se o número de tarefas do sistema. Cada tarefa recebeu requisições, cujos tempos de chegadas foram distribuídos exponencialmente, segundo uma distribuição de Poisson, com uma taxa média igual aos valores dos períodos (tempos médios entre ativações) especificados para as tarefas. A título de comparação com EDF, na abordagem (p+i,k)-firm os valores de deadlines absolutos foram usados para desempatar requisições que possuíam a mesma prioridade. Em todos os experimentos foi utilizada uma taxa média de chegada de 100% (portanto, com sobrecargas transientes constantes).

Observa-se pelos resultados que o desempenho da abordagem (p+i,k)-firm melhora consideravelmente quando o número de tarefas do sistema cresce de 2 para 4. Essa melhoria no desempenho tende a se estabilizar quando o número de tarefas se aproxima a 10. As outras abordagens também têm um melhor comportamento quando o número de tarefas cresce. Entretanto apresentam melhoria no desempenho bem modesta quando o número de tarefas cresce sob a mesma carga.

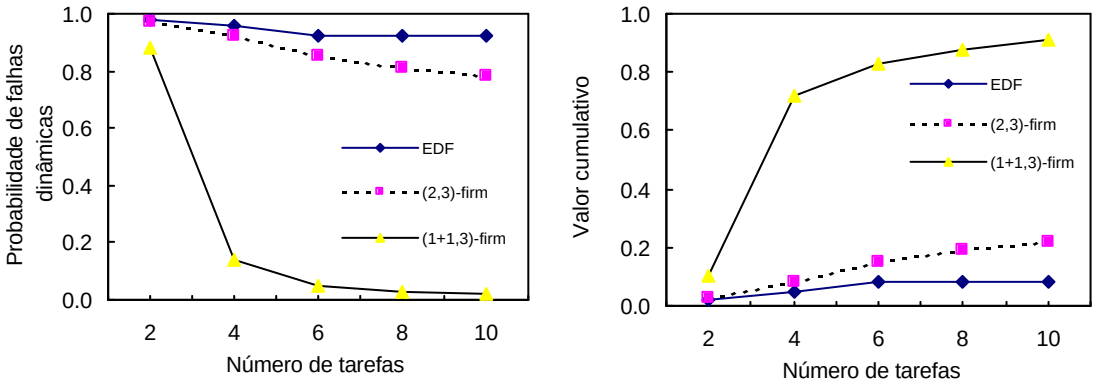

Figura 2. Falhas dinâmicas e valores cumulativos: variando o número de tarefas.

Em situações de sobrecarga, onde nem todas as tarefas podem ter seus deadlines satisfeitos, um escalonador precisa escolher quais tarefas serão escalonadas, e evitar o

denominado *efeito dominó*. Nessas situações, um escalonador não pode se basear somente nas restrições temporais. A abordagem de escalonamento (p+i,k)-firm procura estabelecer um compromisso entre evitar falhas temporais e maximizar a qualidade obtida. As tarefas recebem prioridades de acordo com suas autonomias para perdas de deadline. Durante uma sobrecarga, as duas políticas de sua heurística agem procurando reduzir a carga (selecionando algumas execuções na forma imprecisa) e tentando evitar falhas dinâmicas (priorizando determinadas tarefas). Essa foi, primordialmente, a razão dos excelentes resultados obtidos em comparação com as duas heurísticas testadas.

5. Conclusões

Futuros sistemas de tempo real serão complexos, podendo ser integrados com tecnologias de inteligência artificial e outras áreas de pesquisas, criando novas demandas e dificuldades na obtenção de suas correções temporais. A tendência encontrada atualmente de crescimento, distribuição e sofisticação desses sistemas implica na necessidade de encontrar abordagens que permitam suas execuções, ainda que em plataformas distribuídas e bastante heterogêneas. Abordagens tradicionais de escalonamento de tempo real mostram-se pouco flexíveis e, portanto, difíceis de serem aplicadas. Nessas situações, abordagens adaptativas são propostas, objetivando: (i) utilização de recursos mais eficientemente; (ii) suportar sistemas que podem ser modificados dinamicamente (*ex*. mudanças na missão); (iii) aumentar a robustez dos sistemas que executam em ambientes não deterministas.

Este artigo apresentou a abordagem de escalonamento (p+i,k)-firm. Uma abordagem adaptativa que combina as técnicas da computação imprecisa com a garantia (m,k)-firm. Simulações conduzidas no sentido de avaliar a abordagem em situações de sobrecarga mostrou que, para o modelo de tarefas proposto, a abordagem apresenta um excelente desempenho.

Uma característica importante da abordagem (p+i,k)-firm é o fato desta desconsiderar os tempos de computação das tarefas. Não existe a necessidade de se estabelecer o pior caso de execução de cada tarefa. Isso é bastante adequado, se considerarmos que essa abordagem é direcionada para ser usada em sistemas dinâmicos, em ambientes não deterministas.

6. Referências

[1] G. Bernat, A. Burns, "Combining (n m)-Hard Deadlines e Dual Priority Scheduling", *In Proc. of the 18th IEEE RTSS*, Dec. 1997.

[2] M. Caccamo, G. Butazzo, "Exploiting Skips in Periodic Tasks for Enhancing Aperiodic Responsiveness", *In Proc. of the 18th IEEE RTSS*, Dec. 1997.

[3] J. -Y. Chung, J. W. S. Liu, K. -J. Lin, "Scheduling Periodic Jobs that Allow Imprecise Results", *IEEE Transactions on Computer*, 39(9), 1990, pp.1156-1174.

[4] M. Hamdaoui, P. Ramanathan, "A Dynamic Priority Assignment Technique for Streams with Deadline (m,k)-firm", *In IEEE Trans. on Computer*, Apr. 1995.

[5] G. Koren, D. Shasha, "Skip-Over: Algorithms e Complexity for Overloaded Systems that Allow Skips", *In Proc. of the 16th IEEE RTSS*, Pisa, Italy, Dec. 1995.

[6] T. Kuo, A. K. Mok, "Incremental Reconfiguration e Load Adjustment in Adaptive Real-Time Systems", *IEEE Trans. on Computers*, Vol. 46, No. 12, Dec. 1997.

[7] J. W. S. Liu, W. Shih, K. -J. Lin, R. Bettati, J. -Y. Chung, "Imprecise Computations", *Proceedings of IEEE*, Vol. 82, No. 1, Jan. 1994, pp. 83-94.

[8] C. Montez, J. Fraga, R. S. Oliveira, J-M. Farines, "An Adaptive Scheduling Approach in Real-Time CORBA", 2nd IEEE International Symposium on Object-oriented Real-time Distributed Computing - ISORC'99, Saint-Malo, France, May 1999.

[9] J. A. Stankovic, C. Lu, S. H. Son, "The Case for Feedback Control Real-Time Scheduling" *Proc. ECRTS'99*, York, England, Jun. 1999.

[10] Baruah *et al*., "On the Competitiveness of On-Line Real-Time Task Scheduling", *Proc. of the 12th IEEE RTSS*, pp. 106-115, 1991.