Estudo Experimental do Linux como Plataforma para Aplicações de Tempo Real Brando

Cássia Yuri Tatibana Carlos Alexandre Piccioni Rômulo Silva de Oliveira
cytatiba@das.ufsc.br piccioni@das.ufsc.br romulo@das.ufsc.br

LCMI - DAS - Universidade Federal de Santa Catarina
Caixa Postal 476, Campus Universitário, Florianópolis-SC, 88040-900

Resumo

O trabalho apresentado descreve o processo de implementação e execução de aplicações hipotéticas de tempo real no Linux padrão. Tais atividades apresentaram dados que ajudam a delinear as limitações do Linux para tempo real e prover informações que ajudem na implementação de aplicações de tempo real brando.

Palavras-chave: tempo real, sistema operacional, Linux, avaliação.

1 Introdução

Na realização de operações de controle de tráfego aéreo ou jogos de vídeo game, é exigido do sistema computacional que os gerencia um comportamento temporal previsível. Para esses tipos específicos de aplicações, é importante que o sistema forneça respostas dentro de um prazo específico, sob pena de provocar danos até mesmo catastróficos ao ambiente com o qual iteragem. É identificado nestes sistemas, requisitos de natureza temporal. Essa é a classe de sistemas denominada de sistemas de tempo real [3].

Conforme a literatura, nos sistemas tempo real críticos (*hard real-time*) o não atendimento de um requisito temporal pode resultar em conseqüências catastróficas tanto no sentido econômico quanto em vidas humanas. Para sistemas deste tipo é necessária uma análise de escalonabilidade ainda em tempo de projeto (*off-line*). Quando os requisitos temporais são brandos (*soft real-time*) eles apenas descrevem o comportamento desejado. O não atendimento de tais requisitos reduz a utilidade da aplicação mas não a elimina completamente nem resulta em conseqüências catastróficas.

O Linux é um kernel convencional, monolítico, não preemptivo com suporte a Multiprocessamento Simétrico. Entre outras propriedades, estão inclusas multiprogramação, protocolo de rede TCP/IP e muitas outras características consistentes com um sistema tipo Unix [2].

A popularização do Linux como plataforma para aplicações diversas é ao mesmo tempo causa e conseqüência de sua constante evolução. A flexibilidade e o custo desse sistema tem atraído a atenção de projetistas de todas as áreas, e sua utilização como plataforma para aplicações de tempo real brando torna-se cada vez mais comum. É consenso que o Linux convencional não é apropriado para aplicações de tempo real crítico. Entretanto, não está claro na literatura até que ponto o Linux pode ser utilizado em aplicações de tempo real brando [6].

O objetivo deste trabalho é a observação do comportamento temporal de aplicações de tempo real brando sobre o Linux convencional. A observação e análise do comportamento foi feita através de tempos de resposta capturados durante a execução de aplicações hipotéticas especialmente projetadas para estas experiências. Espera-se com isto fornecer subsídios para os desenvolvedores de aplicações de tempo real brando julgarem se o Linux convencional apresenta comportamento satisfatório, capaz de atender os requisitos temporais da aplicação

em questão. A seção 2 descreve as condições das experiências. A seção 3 apresenta as medições. Na seção 4 os resultados são comentados e a seção 5 contém as conclusões.

2. Condições das experiências

No decorrer da etapa de preparação dos experimentos procurou-se basear as aplicações na teoria de tarefas periódicas descrita em [3] e no conhecimento existente na literatura a respeito de tarefas de tempo real. Por promover a concorrência e causarem o mínimo custo sobre o sistema operacional, *threads* foram adotadas na implementação das tarefas de tempo real das aplicações. Embora existam mecanismos que implementem esforços no sentido de atender as necessidades de aplicações de tempo real como as classes de escalonamento SCHED_FIFO e SCHED_RR, não existe no Linux facilidades para tornar *threads* periódicas, por exemplo. A construção dessas tarefas depende do esforço do programador, usando os mecanismos disponíveis para conseguir o melhor resultado possível.

De maneira bastante simplificada, as tarefas periódicas implementadas são constituídas por um laço de instruções contendo uma tarefa de tempo real hipotética, a captura de instantes de tempo, e uma chamada do tipo *sleep*. A precisão da periodicidade das tarefas implementadas estão intrinsecamente relacionadas com a precisão dos mecanismos de captura de instante de tempo e da função *sleep*. O uso da função *sleep* permite inserir a periodicidade na tarefa, que permanece inativa a partir do instante de término de uma ativação até o instante de início da próxima ativação. O intervalo utilizado pela chamada é calculado a cada ativação com base no tempo capturado no final da execução da tarefa de tempo real hipotética e no período escolhido para a mesma.

A grade de tempo da tarefa é definida pelo instante inicial e pelo valor do período. Ela independente de qualquer outro parâmetro relacionado à execução da tarefa e por isso pode facilmente ser deslocada no eixo do tempo. A partir do tempo de chegada de cada instância da tarefa, e da captura de instante de início e fim de tempo de computação, o cálculo das demais características temporais da mesma é realizado.

A função de inativação utilizada nas tarefas das aplicações é a chamada de sistema *nanosleep()*. Embora ela use como parâmetro um valor especificando o tempo com precisão de nanossegundos, assim como as demais tarefas que executam sobre o Linux, está sujeita ao efeito do temporizador. Isto implica que a tarefa pode demorar mais de 10ms, além do tempo solicitado, para retornar; seu funcionamento especifica inativação por pelo menos o tempo indicado pelo parâmetro, sem especificar limite máximo de inativação.

A partir da observação do comportamento das tarefas, verificou-se que as tarefas periódicas descrevem uma grade de tempo cuja precisão pode ser melhorada se a grade puder ser sincronizada com a grade descrita pelas interrupções do temporizador do Linux. Por isso, ao inserir uma chamada *sleep(1)* no início de cada *thread* de tempo real, o comportamento observado apresenta melhora significativa.

O estudo sobre o comportamento das funções do tipo *sleep* revelou que essas chamadas estão sujeitas a um atraso aproximadamente constante. Mesmo efetuando a sincronização das grades da tarefa e do sistema, o tempo fornecido como parâmetro da chamada e o tempo durante a qual a tarefa realmente permanece inativa apresentam uma diferença durante todas as ativações. Uma vez encontrado um valor compensatório, pode-se aproximar o *jitter* de liberação de zero. O valor compensatório é encontrado através de métodos empíricos, e varia de acordo com o ambiente de execução da tarefa. A descrição dos experimentos realizados na busca do valor compensatório são detalhadas em [5].

O valor compensatório utilizado nos experimentos relatados neste trabalho foi -18ms, ou seja, o tempo de *sleep* solicitado é 18ms do que o realmente necessário. Este ajuste antecipa o fato do Linux dormir, nas aplicações em questão, 18ms a mais do que o solicitado. Um efeito colateral negativo é que -18ms é um valor aproximado, e provoca eventualmente valores de R negativos, pois a tarefa acorda antes do instante de chegada. Tais valores indicam que o início da execução da tarefa ocorreu antes do início de seu período. Embora o valor compensatório varie de acordo com o ambiente de execução, não existe qualquer espécie de erro cumulativo.

Os tempos de chegada, determinados pela grade de tempo da tarefa são definidos em sua primeira ativação, e podem ser deslocadas sobre o eixo do tempo de acordo com a finalidade desejada pelo projetista da aplicação. Assim, caso se tenha interesse em que os valores coletados pela aplicação sejam sempre positivos, basta efetuar o deslocamento da grade de tempo da tarefa. Este deslocamento é conceituado em [1] e é denominado *offset*.

2.1 Aplicações

A construção das aplicações objetivou tarefas simples que pudessem prover informações válidas ao maior número possível de aplicações; e um ambiente que permitisse expor as tarefas de tempo real a fontes diversas de interferências. São descritos neste trabalho os resultados obtidos pelas aplicações denominadas TR7 e TR9. Ambas as aplicações são compostas por cinco *threads* de tempo real. TR7 possui ainda duas *threads* não tempo real implementando funções de uso de teclado e vídeo, e TR9 possui além das *threads* de TR7, mais duas *threads* que utilizam intensamente dispositivos de rede e disco.

A aplicação TR7 foi construída com o objetivo principal de observar o comportamento de várias tarefas de tempo real no Linux. Baseado em experimentos anteriores foi possível concluir que tarefas simples, como as tarefas comuns de TR7 possuem pouca influência sobre os tempos de resposta capturados. A aplicação TR9, por outro lado, tem como principal objetivo investigar o comportamento diante de tarefas comuns de uso intenso de dispositivos.

As prioridades das tarefas de tempo real foram definidas seguindo a política Taxa Monotônica [4]. A primeira thread (T0), sendo a mais prioritária, possui período igual a 50ms, as demais *threads* apresentam períodos de 150, 200, 250, e 300ms. Elas são denominadas T1, T2, T3 e T4, respectivamente. O corpo da tarefa de tempo real construída para os experimentos não apresenta nenhum propósito específico, mas deve simbolizar uma seqüência de operações que farão uso de tempo do processador. O corpo da tarefa é composto por um laço de operações de ponto flutuante, com duração de cerca de 5 microssegundos.

As tarefas de leitura e escrita em dispositivos são constituídas de uma leitura de teclado impressão de caracter na tela. A tarefa de rede permanece em repetição, requisitando pacotes de rede de um servidor local, enquanto a tarefa de disco realiza a leitura de arquivos comuns de tamanho similar ao tamanho da memória principal da máquina.

3. Resultados das experiências

O gráfico de histograma, construído para cada tarefa, mostra a distribuição dos tempos de resposta. As tarefas da aplicação executam, respectivamente 3000, 1000, 750, 600 e 500 ativações por sessão de experimento.

A tabela 1 expõe os resultados obtidos a partir da execução de TR7. Nela são ilustrados, os tempos de resposta mínimo e máximo de cada tarefa, a variação de acordo com estes tempos, o desvio padrão calculado para este intervalo de variação, e dados referentes a 90% do total de ativações com tempo de resposta mais próximos do valor médio.

tarefa	100% Rmin	100% Rmax	100% Variação	100% d.padrão	90% Rmin	90% Rmax	90% Variação	90% D.padrão
T0	-83	25	108	8.97	-77	-50	27	6.83
T1	-101	9	110	10.64	-93	-60	33	8.46
T2	-113	17	130	15.53	-105	-58	47	12.61
T3	-128	-36	92	17.95	-121	-63	58	14.55
T4	-126	3	129	17.34	-114	-56	58	13.32

Tabela 1 - gui-TR7 (valores em microssegundos).

A oscilação dos valores de tempo de resposta de TR7 é baixa, o que indica um comportamento estável da tarefa durante o tempo de duração da experiência. Os valores correspondentes a 90% das ativações apresentam uma pequena diminuição. As figuras 1 e 2 mostram os histogramas.

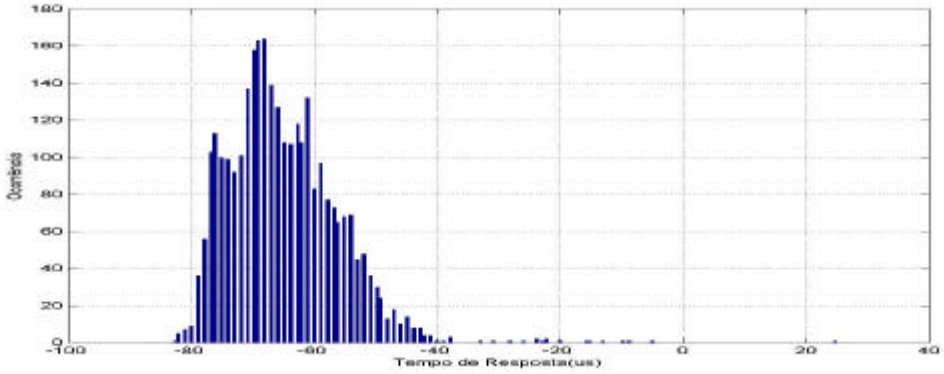

Figura 1 - Histograma de R - gui-TR7 T0.

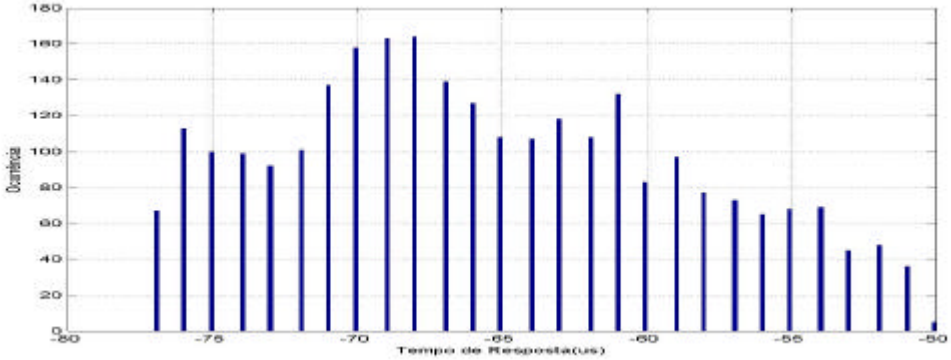

Figura 2 - Histograma parcial (90%) de R - gui-TR7 T0.

As tarefas subseqüentes de TR7 apresentam, apesar dos diferentes valores de período, um comportamento similar a tarefa T0. Como visto na tabela 1, os valores aumentam conforme o período da tarefa. Existe uma expansão da variação de cada tarefa em comparação com a tarefa anterior. Histogramas nas figuras 3 e 4 mostram a distribuição de R da tarefa T4.

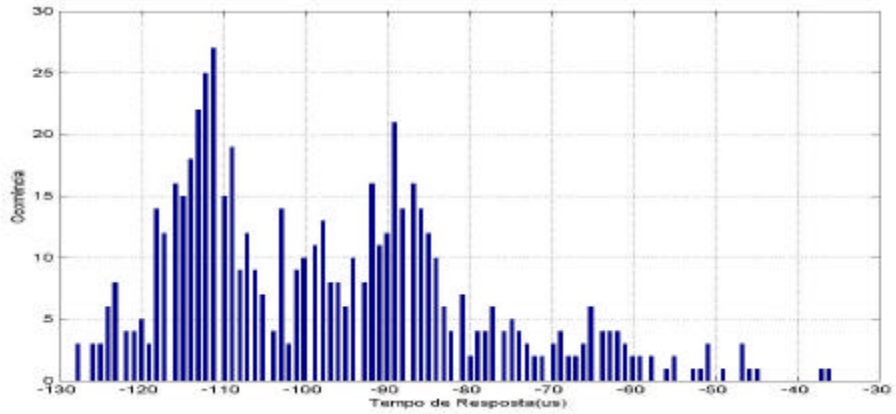

Figura 3 - Histograma de R - gui-TR7 T4.

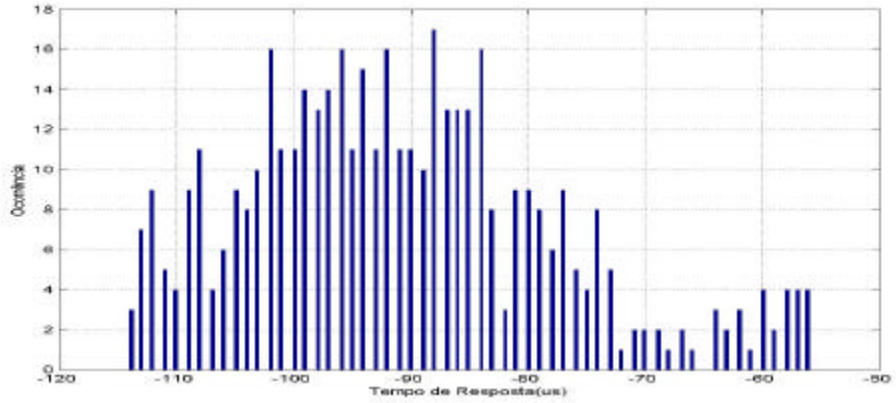

Figura 4 - Histograma parcial (90%) de R - gui-TR7 T4.

A execução de TR7 permite observar o comportamento de aplicações de tempo real no Linux executando no sistema supostamente dedicado a esta aplicação. Excetuando os processos do próprio sistema operacional, cada tarefa de tempo real estaria sujeita apenas a interferências provocadas por outras tarefas da mesma aplicação. Uma vez que as tarefas de tempo real no Linux possuem prioridades maiores que as tarefas não tempo real da aplicação, cada tarefa de tempo real poderia sofrer interferência direta apenas das demais tarefas com prioridade maior implementadas em TR7.

Experimentos apresentaram eventualmente picos correspondentes a um tempo de resposta muito maior que os demais valores de R apresentados pelo restante das instâncias da mesma tarefa. Este tempo de resposta é geralmente apresentado por uma única instância de cada tarefa. A este tempo de resposta estão associadas possíveis interferências provocadas por atividades no kernel relacionadas com a ativação da própria aplicação de tempo real. Mesmo assim, em experimentos em que existe a ocorrência deste valor de tempo de resposta maior, a propagação desta distorção não se estende por muitas instâncias, de forma que o restante do experimento apresenta comportamento uniforme. Esses picos não aparecem nos histogramas.

A aplicação TR9 objetivou delinear o comportamento de tarefas de tempo real no Linux na presença de tarefas comuns de uso intensivo de dispositivos de rede e disco além de tarefas simples de operação de entrada e saída. Como indicado pela tabela 2, os tempos de resposta desta aplicação são significativamente maiores que na aplicação TR7, mesmo existindo entre

elas apenas a diferença de duas *threads* não tempo real. Histogramas para T0 aparecem nas figuras 5 e 6.

tarefa	100% Rmin	100% Rmax	100% Variação	100% d.padrão	90% Rmin	90% Rmax	90% Variação	90% D.padrão
T0	-17993	42731	60724	10590	-9971	24166	34137	9152
T1	-9989	34762	44751	10755	-9950	25254	35204	9377
T2	-10021	32588	42608	10657	-9978	22405	32383	9353
T3	-10036	56246	66282	10467	-9992	22403	32395	8864
T4	-10051	34322	44373	11050	-9971	26217	36188	9640

Tabela 2 - gui-TR9 (Valores em microssegundos).

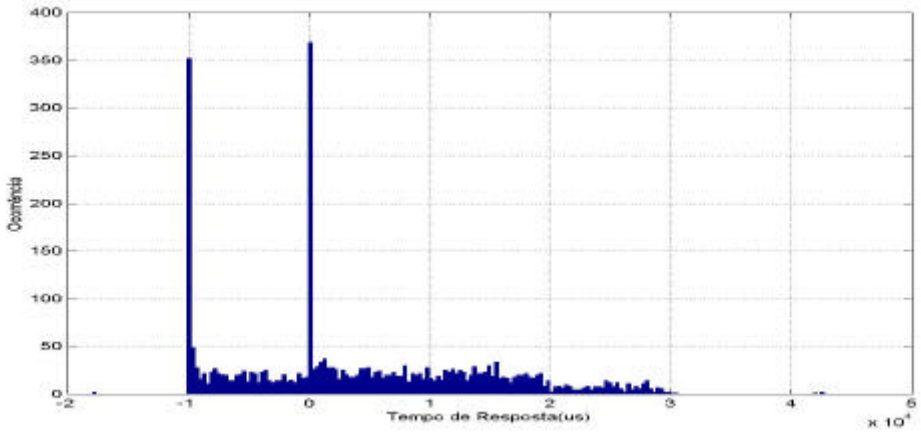

Figura 5 - Histograma de R - gui-TR9 T0.

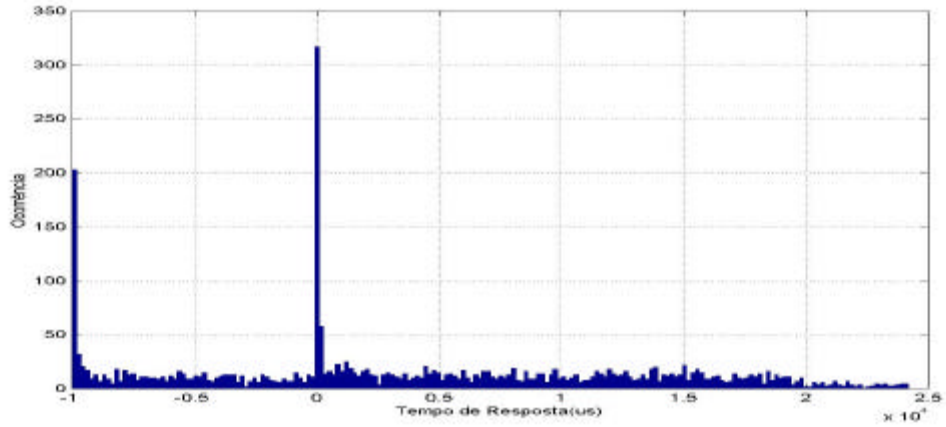

Figura 6 - Histograma parcial (90%) de R -gui-TR9 T0

De acordo com o histograma de T0 ilustrado na figura 5, a distribuição de tempos de resposta se concentra sobre dois conjuntos de valores, formando dois picos no gráfico de histograma. Os picos de ocorrências apresentam entre sí uma diferença de 10ms. Baseado no conhecimento sobre o funcionamento do Linux e na resolução do *timer* desse sistema, considera-se este comportamento, resultante do atraso do atendimento do *nanosleep()*.

O que pode ser observado pelo histograma é que T0 conclui sua execução em um tempo t ou t+10ms. Interrupções de tempo geradas com período de um *tick* determinam a execução de várias atividades do sistema. Tais atividades, tem prioridade sobre processos executados

em espaço de usuário, e portanto provocarão interferência significativa sobre eles, mesmo quando os processos de usuário são tarefas de tempo real. Histograma parcial é mostrado na figura 6. Esse comportamento ilustrado no histograma de T0 é freqüentemente repetido em experimentos que envolvem a execução de tarefas não tempo real complexas. A interferência provocada por estas tarefas nas tarefas de tempo real tem reflexo direto sobre os tempos de resposta gerados pelas mesmas. A distribuição de ocorrências das demais tarefas de TR9 descrevem a mesma curva de distribuição de T0.

4. Comentários

A aplicação TR7 provê tempo de resposta com variações de microssegundos. Embora a função *sleep*, forneça um bom desempenho na maior parte das ativações das tarefas, eventuais interferências provocadas por atividades internas ao kernel, aliadas ao efeito da resolução do timer do Linux (inapropriada para tempo real), provocam tempo de resposta bem maior que o tempo de resposta típico, para algumas instâncias da tarefa.

A análise de resultados obtidos a partir dos cenários com a aplicação TR9, mostram que as aplicações que envolvem tarefas não tempo real com uso intenso de dispositivos de rede e disco provocam muito mais interferências que aquelas de teclado e vídeo. O tempo de resposta das tarefas desta aplicação são medidos em milissegundos, três ordens de grandeza maior que os tempos medidos pela aplicação TR7. A execução das tarefas não tempo real que fazem o uso intensivo de dispositivos de rede e disco, provoca um grande número de chamadas a funções do kernel do Linux. Uma vez que se trata de um kernel não preemptivo com várias regiões de interrupção desabilitada, a interferência provocada pelo próprio sistema operacional sobre tarefas de tempo real são bastante grandes.

A atribuição de prioridades da classe de tempo real para as tarefas da aplicação estabelece uma ordem de preferência entre elas que é respeitada pelo escalonador. Entretanto, as maiores interferências sofridas pelas tarefas de tempo real são provocadas pelas tarefas convencionais e por atividades dentro do kernel. Tais interferências atingem igualmente todas as tarefas de tempo real sem distinção de prioridades.

Sendo o kernel do Linux, não preemptivo, a presença de tarefas mais prioritárias na fila de prontos não provoca a troca de contexto de tarefas. Tarefas de tempo real, mesmo prontas para executar, devem esperar a conclusão de atividades disparadas dentro do kernel até que seja encontrado neste, um ponto de preempção. Esta característica do sistema se torna evidentemente prejudicial a tarefas de natureza temporal quando tarefas convencionais de uso intenso de dispositivos de rede e disco estão presentes no ambiente de execução.

Considerando o que foi observado durante os experimentos, verifica-se que os atrasos em ativações das tarefas de tempo real podem ser provocadas por uma série de fatores decorrentes de aspectos inerentes ao funcionamento do próprio Linux. O fato de adotar uma resolução de interrupções de *timer* de 10ms, proporciona uma margem de atraso muito grande para tarefas de tempo real. Embora as chamadas a função de *sleep* sejam feitas baseadas no instante corrente, não há garantias de que esta chamada será processada conforme solicitado. Mesmo executando sem qualquer outra aplicação no sistema, tarefas auxiliares do kernel são processadas periodicamente, fazendo a manutenção do sistema. A interferência provocada por este tipo de tarefa é contabilizada nos tempos de resposta.

Se o sistema esta ocupado com tarefas mais prioritárias no momento em que a chamada de *sleep* foi feita, a chamada deve esperar, provocando a desatualização do intervalo de tempo usado como parâmetro da chamada *sleep*. Vários *ticks* de clock podem ser perdidos nesta

situação. Não existe um limite para este tempo de atraso, somente dados estatísticos que baseiam a probabilidade desse tempo. Situação semelhante é encontrada novamente pela tarefa de tempo real no momento em que a tarefa cumpre seu tempo em *sleep* e deve retornar a fila de prontos. O instante em que é terminado o prazo de *sleep* da tarefa pode não ser o instante em que esta será colocada na fila de prontos. Toda interferência presente no sistema durante estas atividades são contabilizadas no tempo de resposta da instância em questão e caracterizam o *jitter* de liberação da ativação.

Tendo em vista a magnitude da interferência provocada no cenários da aplicação TR9 em comparação com os cenários de TR7, verifica-se que tarefas de acesso a rede e disco provocam a execução de longos trechos dentro do kernel, resultando em tempos de resposta muito maiores por parte de tarefas de tempo real desta aplicação. Para este tipo de problema apresentado por sistemas de propósito geral, a troca da política de escalonamento aliada ao aumento de pontos de preempção no kernel poderiam significar o aumento da aplicabilidade do Linux padrão para sistemas de tempo real.

5. Conclusões

Considerando o fato do Linux ter sido construído para aplicações de propósito geral, ele apresenta um bom desempenho quando dedicado exclusivamente para o atendimento de aplicações de natureza temporal, sem acesso aos periféricos. Seu comportamento permanece aceitável quando inseridos no sistema tarefas não tempo real simples. Porém, ao fazer o uso de dispositivos como rede e disco de forma intensa, o comportamento do sistema sofre uma degradação com reflexo sobre os tempos de resposta na forma de milissegundos, três ordens de grandeza maior que em aplicações simples.

Os dados apresentados aqui foram obtidos em um PC com processador AMD K6-II 500 MHz. Experimentos em outro computador mais lento mostraram que, embora os números sejam diferentes, o comportamento geral das tarefas é o mesmo. Foi empregada a versão 2.4 do kernel.

Os dados coletados nos experimentos fornecem informações sobre o comportamento temporal do sistema operacional Linux e pode ajudar desenvolvedores na decisão sobre a aplicabilidade do sistema às suas necessidades específicas.

Referências

[1] Burns A., e Wellings A., Real Time Systems and Programming Languages, 2a Edição Addison-Wesley, 1997.

[2] Oliveira R. S. de, Carisimi A. e Toscani S. S. Sistemas Operacionais como Programas Concorrentes. ERAD 2002, 2a Escola Regional de Alto Desempenho (São Leopoldo, RS, Brasil, Janeiro2002), pp139-177.

[3] Farines J. M., Fraga J. S. e Oliveira R. S., Sistemas de tempo real, 1a Edição Escola de Computação 2000, Departamento de Ciência da Computação da Universidade de São Paulo, São Paulo-Brasil, Julho 2000.

[4] Liu C. A., e Layland J. W., Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. pp. 46-61.

[5] Piccioni C. A., Tatibana C. Y., e Oliveira R. S., Trabalhando com Tempo Real em Aplicações sobre o Linux. Relatório Técnico DAS-UFSC, 2001.

[6] Willshire P. Real Time Linux: Testing and Evaluation Second Real Time Linux Workshop, Florida, USA, 1751 Hotel Plaza Boulevard, Lake Buena Vista, Nov. 27-28, 2000. www.thinkingnerds.com