Slide 2: Conceitos de Tempo Real

Sistemas de Tempo Real

Um sistema pode ser dito de tempo real quando sua correção lógica está relacionada tanto a correção das saídas produzidas quanto pela sua pontualidade (correção temporal). [Laplante, P. A.; Ovaska, S. J. Real-Time Systems Design and Analysis. 4. ed. Estados Unidos da América: IEEE Press, Wiley, 2012]

Duas das principais características de um sistema de tempo real que estão ligadas diretamente com sua corretude temporal são a previsibilidade e o determinismo.

Os sistemas de tempo real podem ser classificados, quanto ao grau de exigência que fazem da pontualidade do sistema, como sendo do tipo Soft (brandos) ou Hard (rígidos).

Um sistema de tempo real é brando quando o não cumprimento de suas restrições temporais provoca apenas alguma degradação no desempenho do sistema mais sem provocar falhas graves.

Um sistema de tempo real é rígido quando a falta no cumprimento de uma única restrição temporal pode levar o sistema a uma falha catastrófica.

Tarefas de Tempo Real

Uma tarefa pode se classificada como de tempo real se obedece as mesmas definições estabelecidas para sistemas de tempo real, e assim como eles, também são classificadas em brandas e rígidas

As tarefas de tempo real, também são classificadas quanto a periodicidade de sua ativação podendo ser periódicas, quando sua ativação ocorre em intervalos regulares predefinidos, ou como aperiódicas quando sua ativação ocorre de modo aleatório, normalmente em resposta a algum evento externo ou interno ao sistema.

Do ponto de vista do escalonamento, as tarefas de tempo real podem ser definidas por quatro parâmetros temporais: tempo de computação, frequência de ativação, deadline e o tempo que decorre do momento da ativação de uma tarefa ao momento em que ela realmente inicia sua execução, esse intervalo de tempo também é conhecido como jitter ou latência de ativação.

Em alguns algoritmos de escalonamento existe um parâmetro correspondente a prioridade da tarefa, que é usado para estabelecer uma ordem de precedência entre as várias tarefas executadas pelo sistema.

Slide 3: Sistemas Operacionais de Tempo Real

Um Sistema Operacional de Tempo Real (SOTR), além de incorporar os principais objetivos comuns aos SO, deve criar um ambiente de desenvolvimento previsível e determinístico qualquer que seja a condição do sistema. Um SOTR deve fornecer um ambiente no qual aplicações de tempo real (ATR) tenham seus requisitos temporais respeitados e executar de modo que suas ações possam ser previstas de forma determinística.

Ao contrário do que diz o senso comum, um SOTR não tem como principais objetivos a redução dos tempos de resposta a estímulos ou ter uma performance superior quando comparado a um Sistemas Operacionais de Propósito Geral (SOPG).

Slide 4: Latência

Segundo o dicionário Priberam da Língua Portuguesa, Latência é: “Tempo decorrido entre o estímulo e a resposta correspondente.”

Assim como todos os sistemas reais, SOTR, estão sujeitos a latências que surgem como consequência do seu próprio funcionamento e do hardware sobre os quais executam. O conhecimento dos valores de latência e principalmente sua constância, mesmo que em um cenário de sobrecarga do sistema, são primordiais na garantia da previsibilidade e determinismo dos SOTR. O conhecimento dos valores de latência também são de vital importância na seleção de um SOTR que seja capaz de atender aos requisitos temporais de uma aplicação. São causas comuns de latência em SO:

- Latência de interrupção: É o tempo decorrido entre a ocorrência de uma interrupção e o momento em que é atendida. Há também o tempo entre o atendimento da interrupção e a rotina que realmente processará o sinal recebido. O tempo decorrido entre o despertar de um processo de alta prioridade e a sua execução as vezes podem ser consideradas latência de interrupção, uma vez que o seu despertar muitas vezes ocorre devido a algum evento externo.
- Latência de escalonamento: É o tempo entre o instante em que uma tarefa de alta prioridade acorda e o momento em que ela inicia a execução.
- Latência por inversão de prioridade: É o tempo que uma thread com prioridade alta espera para utilizar algum recurso em uso por uma thread de prioridade baixa. Em muitos casos a inversão de prioridade não pode ser evitada, porém é desejável que um SOTR de qualidade evite sua ocorrência de forma ilimitada.
- Latência por inversão de interrupção: É o tempo que uma thread de prioridade alta espera pelo manipulador de interrupções que executa uma tarefa de prioridade baixa. Diferente da

inversão de prioridade e inversão de interrupção não pode ser antecipada visto que a ocorrência da maior parte das interrupções não pode ser prevista.

Slide 6: Kernels de tempo real

Para proporcionar uma camada mínima de abstração para o programador, um kernel com suporte a multitarefa, devem fornecer três recursos básicos: escalonamento, comunicação e sincronia entre tarefas e despachante. O escalonador é responsável pela alocação temporal das tarefas, é ele quem determina em que momento cada tarefa concorrente poderá utilizar a CPU. O despachante gerencia a troca de contexto entre as tarefas, salvando o contexto da tarefa interrompida e carregando o contexto da nova tarefa, escolhida pelo escalonador, que entrará em execução. A comunicação e sincronia entre tarefas permite que um problema seja solucionado por várias delas de forma colaborativa. Os sistemas que não implementam muito mais que estas funcionalidades básicas também são chamados de: núcleo de tempo real ou microkernel.

No mercado existem diversas soluções que implementam estas funcionalidades mínimas, como FREERTOS, que tem por objetivo ser um sistema pequeno, rápido e com garantias de tempo bastante rígidas, direcionado ao mercado de sistemas embarcados, em sua maior parte, microcontrolados, e empregados em sistemas de tempo real rígidos que controlam aplicações críticas.

Os grandes sistemas operacionais de propósito geral como: Linux e Windows, oferecem outros recursos mais avançados: serviços de entrada e saída de dados, proteção de memória, sistemas de arquivos, políticas de segurança, interface com o usuário etc. Estes recursos tornam possível ao desenvolvedor construir e executar aplicações mais complexas que proporcionam um maior nível de segurança e interação com outros sistemas e usuários. Vale lembrar que muitos dos sistemas em que se baseiam alguns SOTR, como Linux, já são utilizados em várias aplicações de missão crítica além de que sistemas de controle baseados em PC são uma realidade na indústria como complemento e até substitutos aos tradicionais CLPs [Siemens PC-based Automation]. Estes sistemas permitem uma maior flexibilidade na programação, expansão e configuração de software e hardware.

Slide 7 e 8: Objetivos

Esse trabalho tem como objetivo avaliar a capacidade do patch Preempt_RT, oficialmente suportado pelos desenvolvedores do kernel, de transformar um PC monoprocessado em um

computador capaz atender aos requisitos de uma aplicação de tempo real rígida e comparar os resultados com o RTAI, um sistema maduro, testado e consolidado.

Para esta avaliação foram escritas duas aplicações, baseadas nos testes realizados por Anderson(2007) e no benchmark Cyclictest, e portadas para o RTAI para que fosse possível realizar a comparação. Nesse processo também foi observada a facilidade de instalação dos respectivos patch e desenvolvimento de aplicações para ambas as soluções.

Slide 8: Trabalhos Relacionados

Durante a pesquisa bibliográfica foram identificados alguns trabalhos semelhantes e que nos forneceram diversos recursos para a elaboração deste trabalho, dentre eles se destacam Moreira(2007), que nos forneceu o modelo de testes por nós utilizados, assim como medições de performance do RTAI, utilizados como valores de referência.

Também podem ser destacados os trabalhos de Litayem(2011) e Hallberg(2017) que nos forneceram valores de referência e demonstram a real qualidade dos resultados oferecidos pelo benchmark Cyclictest na avaliação e comparação de sistemas de tempo real baseados em Linux.

Slide 9: Patch Preemp_RT

Consiste da solução de tempo real para linux que não utilizam uma arquitetura dual kernel, mais bem sucedida e é oficialmente suportada pelos desenvolvedores do kernel Linux. Sua abordagem proporciona um maior aproveitamento dos recursos já suportados pelo sistema, como: utilização da API de tempo real definida pelo padrão Posix, utilização do conjunto das chamadas de sistema e a criação de aplicações que executem em modo usuário ou modo kernel.

Slide 10: RTAI

É o acronimo de Real-Time Application Interface, surgido de uma variação do projeto RTLinux e desenvolvido por um grupo de pesquisadores da Universidade politécnica de Milão. Sua abordagem utiliza uma solução com um microkernel, responsável pelo gerenciamento e execução das tarefas de tempo real, em paralelo ao kernel principal do Linux, este microkernel se assemelha aos núcleos de tempo real visto anteriormente.

Embora utilize um kernel separado para as aplicações de tempo real, é possível executar estas tarefas denttro programas executados no modo usuário.

Para que seja possível utilizar dois kernels simultaneamente sobre um mesmo hardware o RTAI utiliza os recursos de uma camada de abstração de hardware chamada ADEOS.

Slide 11 e 12: Hardware utilizado

Para a execução dos testes propostos foi utilizado como hardware um netbook Acer Aspire One com processador Intel Aton, que também é utilizado em alguns projeto de sistemas embarcados.

Porém, na escolha de um hardware para ser utilizado como sistemas de tempo real é necessário observar algumas restrições: incompatibilidade entre os serviços de gerenciamento de energia e as configurações do kernel, principalmente em sistemas com bateria, a exigência de certos recursos por parte do processador (AMD MCE) ou o não provimento de recursos por parte dele as aplicações (486 + RTAI).

Infelizmente não existe um banco de dados sobre hardwares suportados, por isso a execução de testes é bastante importante.

Slide 13: Distribuição Utilizada

Foi utilizada a distribuição debian em sua versão 8.8, foi instalado um sistema mínimo sem recursos de servidor ou interfaces gráficas. O kernel utilizado na instalação do Preempt_RT foi o 4.4.17 e para o RTAI o 4.4.43 ambos Vanilla. É importante notar que não existe uma versão do kernel que seja suportada por ambas as soluções simultaneamente.

Slide 14 a 19: Produzindo um kernel de tempo real

Utiliza-se o método convencional para aplicação de patch, configuração e compilação do kernel, com detalhe para alguns recursos do kernel que devem ser habilitados e desabilitados de acordo com as necessidades de cada uma das soluções de tempo real utilizadas (Ipipe e full preemptible kernel), outro cuidado tomado foi com alguns recursos do kernel que provocam aumento da latência e imprevisibilidade no sistema, estes recursos foram identificados utilizando o programa Cyclictest e sua função de rastreio de processos vilões da latência. Alguns recursos foram desabilitados na tentativa de que a arquitetura do sistema interferisse da menor forma possível no comportamento do sistema, como as configurações referentes ao processador.

Slide 20: Produzindo um Kernel de Tempo Real - RTAI

Além da aplicação de um patch e compilação do kernel, o RTAI precisa ser configurado e instalado no sistema, para configuração usa-se o make menuconfig, compilação e instalação por meio de make e make install

Slide 21 e 25: Programa de tempo real básico

Para produzir programas de tempo real alguns passos precisam ser seguidos:

- Alocação de memória e travamento para evitar paginação
- Configuração das tarefas como tarefas de tempo real
    - Prioridade
    - Escalonamento
    - Periódica ou aperiódica (RTAI)
- Execução de alguma computação
    - Ler valores de tempo
    - Dormir por um determinado intervalo

Com exceção do travamento de memória Preempt_RT e RTAI utilizam sintaxes diferentes para lidar com tarefas de tempo real, cada uma com sua API.

Slide 26: Programa de tempo real básico - RTAI

Para que um programa possa ser compilado e executado utilizando recursos do RTAI é necessário o carregamento de módulos e a definição de algumas variáveis. Para automatizar o processo nós criamos um script de inicialização do RTAI.

Slide 27 ao 32: Os testes

Foram desenvolvidos testes para avaliar o sistema em duas situações, conforme o benchmark desenvolvido por Moreira(2007), durante a execução de cinco tarefas periódicas e de cinco tarefas periódicas junto a duas aperiódicas.

Todas as tarefas executam alguma computação. As periódicas nas frequências de 1Hz, 2Hz, 4Hz, 8Hz e 16Hz. Todas com a mesma prioridade e deadlines iguais os períodos. As tarefas aperiódicas executam com prioridade maior que as periódicas e dentro de um instante aleatório entre 20ms e 40ms e deadline de 20ms.

Ambos os programas seguem o mesmo fluxo básico: Configurar as tarefas, inicia-las todas ao mesmo tempo por meio de uma thread “criadora”, dormir por um intervalo de tempo, calcular a latência, verificar se atingiu o deadline, quando chega ao fim, imprimir o histograma de cada tarefa.

O algoritmo de medição é bastante simples e pode ser utilizado para medir latências dentro da aplicação final.

Slide 33 ao 39: Resultados

Os testes utilizando apenas tarefas periódicas nos mostram intervalos de latência bastante consistentes e estreitos para a maior parte das tarefas executadas, com exceção da thread 4 do teste executado no RTAI, que pelo gráfico fica evidente a existência de uma anomalia, e que ainda não teve sua causa avaliada. Algo negativo foi o fato de que os valores de latência, em alguns casos, superam 100% do tempo de computação das tarefas, porém a soma dos valores de Latência e Tempo de Computação foram bem inferior aos deadlines das tarefas.

Quando adicionamos duas tarefas aperiódicas aos testes tivemos alguns comportamentos interessantes. A execução do patch Preempt_RT a primeira vista se mostrou inalterada, mas uma análise detalhada dos valores de latência mostram alguns pontos fora da curva e o registro de latências máximas bem superior a média das medições, embora os valores não tenham comprometido a aplicação, esse tipo de comportamento não é adequado para um sistema de tempo real.

Já o RTAI teve um comportamento que parece adiar a execução das tarefas aperiódicas ao longo do tempo embora os valores tenha estado dentro de um intervalo mais bem definido. Podemos nos questionar se a adição de novas tarefas aperiódicas provocaria o aumento das latências destas tarefas.

Mais uma vez os valores somados da Latência com o Tempo de Computação estejam bem abaixo dos valores de deadline, o tempo de computação das tarefas executadas pelo Preempt_RT foram bastante elevados, enquanto no RTAI foram praticamente os mesmos.