INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE P
DEPARTAMENTO ACADÊMICO DE SISTEMA, PROCESSOS E CONTROLES EI
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE

Daniel Barlavento Gomes

COLOQUE SEU TÍTULO AQUI

Recife – Pernambuco

2017

Daniel Barlavento Gomes

COLOQUE SEU TÍTULO AQUI

Trabalho de conclusão apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas da IFPE, como requisito parcial para a obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Mestre Paulo Abadie Guedes

Recife – Pernambuco

2017

Daniel Barlavento Gomes

COLOQUE SEU TÍTULO AQUI

Trabalho de conclusão apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas da IFPE, como requisito parcial para a obtenção do grau de bacharel em Ciência da Computação.

Recife, DD/MM/YYYY.

BANCA EXAMINADORA

Paulo Abadie Guedes

Prof. Mestre - IFPE

(Professor Orientador)

Examinador Interno 1

Prof. Doutor - IFPE

(Professor do Instituto Federal de Pernambuco)

Examinador Externo 2

Prof. Doutor - IFPE

(Professor do Curso de Sistemas de Informações da Universidade)

Dedico a minha família, por todo o apoio e confiança.

AGRADECIMENTOS

"Prefiro não fazer".

Herman Melville (Bartleby, o escriturário)

RESUMO

Morbi efficitur molestie pellentesque. Fusce tincidunt vitae dolor ac ornare. Mauris nibh mi, condimentum nec ex a, semper posuere augue. Ut sagittis condimentum lacus, et lacinia sem ornare nec. Praesent cursus sagittis lacus ut iaculis. Nunc faucibus, elit ac imperdiet malesuada, velit est faucibus diam, vitae ullamcorper sapien augue a nisi. Morbi consectetur pulvinar felis vel feugiat. Phasellus tempor magna eget purus placerat luctus. Vestibulum bibendum dapibus arcu in semper. Phasellus vel porta mauris. Ut nec mauris vel ante auctor vulputate a sed nisi. Nam ullamcorper purus vel dolor interdum efficitur. Aenean rhoncus mollis porta. Vivamus est urna, finibus vel leo at, porta tempus sem.

Palavras-chave:

ABSTRACT

Curabitur malesuada ante lorem, a auctor urna euismod et. Nam viverra, dolor eu feugiat euismod, justo velit tincidunt purus, faucibus interdum mauris metus in turpis. Maecenas hendrerit, felis quis condimentum convallis, metus turpis porttitor ex, non iaculis nisi ex id ligula. Vivamus sed consectetur felis. Maecenas non ligula eu nulla iaculis dictum. Phasellus accumsan tempus purus et consectetur. Praesent dapibus, arcu ut porta dictum, velit lacus ultricies nisl, vitae congue purus mi id ipsum. Pellentesque ac tempus enim, at egestas nulla. Quisque vitae ultrices odio. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed vitae purus ultricies, maximus magna a, aliquet mauris. Aliquam ornare odio sit amet urna placerat vestibulum. Aenean a cursus mauris, quis vulputate erat. Nullam convallis scelerisque ligula, at finibus lectus laoreet at.

Keywords:

Sumário

1 INTRODUÇÃO

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed imperdiet lacus consectetur vestibulum scelerisque. Integer accumsan odio nisi, sed aliquet quam consequat tincidunt. Ut at sollicitudin felis. Duis tempor condimentum velit ac molestie. Donec vitae luctus velit, vitae faucibus mi. Suspendisse potenti. Mauris accumsan mi quis neque aliquet ultricies.

Suspendisse porta ultricies turpis, id porta risus. Sed nec bibendum ligula. Praesent sapien tortor, condimentum ut interdum quis, ornare ac nisi. Phasellus blandit ipsum ac mollis porta. Nunc egestas elementum est, sit amet hendrerit leo finibus placerat. Etiam finibus lorem quis dolor pellentesque, eu hendrerit nisl rhoncus. Aenean a mi consectetur, efficitur justo non, aliquam risus. Sed feugiat nisl vitae venenatis pulvinar. Suspendisse molestie quis tellus eget fringilla. Nunc at posuere tortor. Sed venenatis dui risus, non congue magna vehicula eu.

Morbi efficitur molestie pellentesque. Fusce tincidunt vitae dolor ac ornare. Mauris nibh mi, condimentum nec ex a, semper posuere augue. Ut sagittis condimentum lacus, et lacinia sem ornare nec. Praesent cursus sagittis lacus ut iaculis. Nunc faucibus, elit ac imperdiet malesuada, velit est faucibus diam, vitae ullamcorper sapien augue a nisi. Morbi consectetur pulvinar felis vel feugiat. Phasellus tempor magna eget purus placerat luctus. Vestibulum bibendum dapibus arcu in semper. Phasellus vel porta mauris. Ut nec mauris vel ante auctor vulputate a sed nisi. Nam ullamcorper purus vel dolor interdum efficitur. Aenean rhoncus mollis porta. Vivamus est urna, finibus vel leo at, porta tempus sem.

1.1 Justificativa

1.2 Objetivos

1.2.1 Gerais

1.2.2 Específicos

- Item 1
- Item 2
- Item 3

2 ESTADO DA ARTE

2.1 Sistemas de Tempo Real

2.1.1 Conceitos

2.1.2 Classificação

2.1.3 Algoritmos de Escalonamento

2.2 Sistemas Operacionais de Tempo Real

Segundo (ROSTEDT, 2007) um Sistema Operacional de Tempo Real (SOTR) tem como propósito principal, além daqueles inerentes a todo sistema operacional, proporcionar um ambiente de desenvolvimento e execução previsível e determinístico de forma que as tarefas definidas por um suário sejam executadas corretamente dentro dos prazos definidos nas especificações do sistema. Não são seus objetivos primeiros, embora sejam desejáveis, a redução do tempo entre uma ação e uma resposta a esta ação, redução da latência de execução de tarefas ou o aumento da velocidade do sistema. Um SOTR deve fornecer a seus usuários um ambiente em que as ações de uma determinada aplicação possam ser previstas de forma determinística, qualquer que seja a carga sobre o sistema. O senso comum nos faz pensar que um SOTR possui um desempenho geral melhor que um Sistema Operacional de Propósito Geral (SOPG). Embora um SOTR de qualidade possa ter um desempenho geral equivalente a um SOPG, o primeiro sacrificará esta performance em detrimento da previsibilidade. SOPG podem, em 99,9% dos casos, executar tarefas com um desempenho melhor que um SOTR, todavia nos 0,01% dos casos restantes o tempo de execução de uma tarefa pode ser até 100 vezes maior do que o limite estabelecido por um sistema de tempo real crítico, mais que suficiente para reprovar o sistema. Embora um SOTR execute um conjunto de tarefas num tempo maior que um SOPG, o usuário tem a garantia de que as tarefas serão executadas dentro dos prazos estabelecidos.

Além das garantias de tempo relacionadas a execução de tarefas, um SOTR

deve atender a requisitos referentes a responsividade. Quando um SOTR é submetido a um estímulo, seja interno ou externo, deve garantir que sua resposta é rápida o suficiente para atender aos requisitos temporais do sistema.

Como em qualquer sistema que lida com estímulos, os SOTR estão sujeitos a ocorrência de latências, um intervalo de tempo entre a percepção de um evento e a produção de uma resposta correspondente a este evento. Um evento pode ser um estímulo externo, como uma interrupção de hardware indicando o alarme de um relógio do sistema, ou um estímulo interno como uma *thread* que foi colocada no estado pronto a espera para ser executada. O conhecimento dos valores de latências de um sistema são de grande importância para a escolha de um SOTR e para a determinação precisa dos tempos de execução de um conjunto de tarefas. Sistemas computacionais, conforme (ROSTEDT, 2007), estão sujeitos a algumas causas comuns de latência: latência provocada por interrupção, latência provocada pelo algoritmo de escalonamento, latência provocada por inversão de prioridade e latência provocada por inversão de interrupção.

A latência provocada por interrupções corresponde ao intervalo de tempo entre a ocorrência do sinal de interrupção e o momento em que a rotina de tratamento é executada. A latência provocada pelo algoritmo de escalonamento é o tempo decorrido entre o instante em que uma tarefa é colocado no estado de pronto e o início da execução da tarefa. A latência por inversão de prioridade o corre em SOTR que suportam algoritmos de escalonamento baseados em prioridades e representa o tempo que uma *thread* de prioridade alta espera pera utilizar um recurso que está sendo usado por uma *thread* de menor prioridade. A latência por inversão de interrupção corresponde ao tempo que uma *thread* de alta prioridade espera pelo termino da execução de uma tarefa de tratamento de interrupção de baixa prioridade termine sua execução. (Explicar melhor!!!)

- Mais características dos SOTR!!!

2.2.1 Linux

2.2.2 Implementações

2.3 Patch PREEMPT-RT

2.4 RTAI

3 ANÁLISE E TESTE DOS SISTEMAS

Este capítulo apresenta as principais características utilizadas na comparação dos sistemas testados e as premissas que orientam os testes executados.

3.1 Avaliação de Sistemas Operacionais de Tempo Real

3.1.1 O Ambiente de teste

Na comparação entre diferentes sistemas operacionais, e mais especificamente de *kernels* de tempo real, é importante que a configuração do *hardware* utilizado seja igual o no mínimo equivalente, isso garante que os resultados obtidos são consistentes e que não foram profundamente influenciados pelo *hardware*.

O *hardware* utilizado para testar as duas soluções de tempo real escolhidas foi um *netbook Acer*, modelo *Aspire One* D250-1023, processador com arquitetura x86, *Intel Aton* N270, *clock* de 1,60GHz, memória *cache* L2 de 512KB, 1GB de memória DDR2-533, disco rígido de 320GB SATA.

Ambas as soluções de tempo real testadas usam como base o sistema operacional *GNU/Linux*, foi escolhida a distribuição *Debian* 8.8 (*Jessie*) para processadores de 32 bits. A distribuição *Debian* foi escolhida, dada a facilidade de se produzir um sistema com funcionalidades reduzidas, sua ampla documentação, sua grande coleção de pacotes contendo programas e bibliotecas pré compilados e por ser a base de inúmeras outras distribuições que se aplicam de servidores a sistemas embarcados.

Foi considerado de grande importância produzir *kernels* com configurações idênticas, com exceção das opções específicas exigidas por cada uma das soluções, para que recursos específicos não alterassem o desempenho dos sistemas. As configurações utilizadas tiveram como ponto de partida a versão *vanilla* de cada *kernel*. A versão utilizada do *patch PREEMPT-RT* foi a 4.4.17-rt25 publicada em 25 de agosto de 2016, aplicado sobre um *kernel*, *vanilla*, versão 4.4.17. A versão testada do RTAI foi a 5.0.1 publicada em 15 de maio de 2017, o *patch HAL* foi aplicado em um *kernel*, *vanilla*, versão

4.4.43.

Certas funcionalidades do *kernel*, como *debug*, gerenciamento de energia, paginação, e consequentemente acessos a disco, podem comprometer a previsibilidade das aplicações de tempo real, para evitar estes problemas, as funcionalidades de *debug* e gerenciamento e economia de energia do *kernel* foram desabilitadas, os problemas relacionados a paginação e acessos a disco foram resolvidos nas próprias aplicações como veremos mais adiante. Embora esta configuração não tenha apresentado problemas no *hardware* de teste, a ausência de recursos de gerenciamento de energia inviabilizou o carregamento do *kernel* em outras configurações de *hardware*. Como este trabalho trata do teste de soluções de tempo real que executam sobre sistemas mono processados a funcionalidade do *kernel* que da suporte a *SMP* foi desabilitada.

Para que um sistema operacional possa executar aplicações de tempo real é necessário que o sistema possua suporte a relógios com uma boa granularidade e precisão, assim as opções do *kernel* relacionadas aos relógios de alta precisão foram habilitadas.

3.1.2 Parâmetros Utilizados na Avaliação

A avaliação de um SOTR é definida principalmente pela capacidade de suas características atenderem aos requisitos de um determinado projeto, o que pode envolver diversas variáveis que alteram o desempenho do sistema em diversas circunstâncias diferentes. Outros parâmetros relacionados a requisitos não funcionais de uma aplicação como: suporte e documentação, custo, tamanho do código, etc, corroboram com o número de possibilidades. Isso torna a comparação entre SOTR algo, no mínimo, confuso.

As avaliações de desempenho mais completas de SOTR normalmente são baseadas na avaliação do sistema como aplicações destinadas a fins específicos. Estas avaliações são difíceis de generalizar e portar para outras soluções e arquiteturas de destino diferentes da proposta original dos testes. A escolha de parâmetros quantitativos que sejam comuns a maioria dos sistemas de tempo real, e que estejam diretamente relacionados a execução dos principais casos em que estes sistemas se aplicam, facilita a comparação entre as diversas soluções existentes e proporcionam uma excelente forma de avaliação dos SOTR.

- (Sugestão) O sistema deve apresentar latência inferior a menos de 10% do

deadline das tarefas em execução.

3.2 Sobre os Testes Executados

3.2.1 Testes Preliminares

As soluções estudadas foram submetidos a testes preliminares, que foram utilizados para identificar possíveis falhas nos processo de instalação dos sistemas como para identificar funcionalidades do *kernel* que pudessem alterar o desempenho e a preempção do sistema. O principal parâmetro testado foi a latência do sistema. Os resultados obtidos com estes testes também serviram para avaliar a qualidade dos resultados obtidos com os testes desenvolvidos neste trabalho.

Estes testes foram executados por meio de ferramentas recomendadas e fornecidas pelos próprios desenvolvedores dos sistemas abordados. Foram utilizadas principalmente os programas: *Latency*, para testes no *RTAI* e *Cyclictest*, para testes no *PREEMPT-RT*. O algoritmo de medição do programa *Cyclictest* foi utilizado como base para os testes desenvolvidos neste trabalho.

—-Falar sobre o programa Latency —-

O programa *Cyclictest* (FOUNDATION, 2017a) é fornecido junto a suite *rt-tests*, um conjunto de ferramentas para teste de sistemas de tempo real, desenvolvidas e mantidas pelos desenvolvedores do *kernel linux* e hospedada no próprio *site* do projeto.

O programa *Cyclictest* mede com alto grau de precisão, os resultados são fornecidos em microssegundos, a latência do sistema para um número definido de tarefas. Mostrou-se de extrema utilidade seu recurso que possibilita o rastreio de funcionalidades do *kernel* que provocam aumento da latência do sistema, por meio da função *FTRACER*. Este recurso foi bastante utilizado para produzir uma configuração adequada do *kernel linux*.

Para que os valores das medições, obtidos com os testes, sejam válidos, é preciso que os testes sejam executados diversas vezes por um período de tempo suficiente longo e que os recursos do sistema (entradas, saídas, CPU, etc) estejam sobrecarregados, com a finalidade de reproduzir um cenário com a pior situação possível para a execução de uma aplicação de tempo real. Como os programas *Cyclictest* e *Latency* medem a latência

do sistema, um cenário adequado de sobrecarga é o uso intensivo do processador, que no pior caso deve estar com valores próximos de 100% de utilização com poucas variações durante o período de execução dos testes.

A solução adotada para este problema foi a proposta por (LIM, 2017). Esta abordagem, além de proporcionar o uso de 100% do processador, possui uma construção simples, e utiliza um conjunto de programas que já se encontram pré instalados na maioria das distribuições *Linux*.

3.2.2 *Benchmarks*

- Definir o que é um benchmark.

3.2.3 Métodos de Análise Utilizados

4 RESULTADOS

5 CONCLUSÕES

5.1 Trabalhos Futuros

Bibliografia

FARINES, Jean-Marie. **Sistemas de Tempo Real**. [S.l.]: Departamento de Automação e Sistemas Universidade Federal de Santa Catarina, 2000.

FOUNDATION, The Linux. **Cyclitest**. 2017. Disponível em: <https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cyclictest>. Acesso em: 17 jun. 2017.

_____. **HOWTO setup Linux with PREEMPT$_R$*Tproperly***. 2017. Disponível em: <https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/preemptrt_setup>. Acesso em: 14 nov. 2017.

_____. **Preemption Models**. 2017. Disponível em: <https://wiki.linuxfoundation.org/realtime/documentation/technical_basics/preemption_models>. Acesso em: 19 abr. 2017.

_____. **RT-Tests**. 2017. Disponível em: <https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/rt-tests>. Acesso em: 14 nov. 2017.

GITE, Vivek. **How to Compile and Install Linux Kernel v4.5 Source On a Debian / Ubuntu Linux**. 2017. Disponível em: <https://www.cyberciti.biz/faq/debian-ubuntu-building-installing-a-custom-linux-kernel/>. Acesso em: 24 jan. 2017.

LIM, Geunsik. **Worst Case Latency Test Scenarios**. 2017. Disponível em: <https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/worstcaselatency>. Acesso em: 14 nov. 2017.

MOREIRA, Anderson Luiz Souza. **Análise de Sistmas Operacionais de Tempo Real**. 2007. Diss. (Mestrado) – Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco.

ROSTEDT, Steven. Internals of the RT Patch. **Proceedings of the Linux Symposium**, 2007.

A APÊNDICE

Para adicionar outros apêndices ou anexos basta usar adicionar capítulos a este arquivo.

A.1 Apêndice 1

1. Item 1