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, 04/12/2017.

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

2.2.1 O Padrão POSIX

2.2.2 Modelos de Implementação

2.2.3 Soluções para GNU-LINUX

2.3 PREEMPT_RT

2.4 RTAI

3 AVALIAÇÃO E COMPARAÇÃO DE SOTR

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 várias circunstâncias diferentes. Outros parâmetros relacionados a requisitos não funcionais de uma aplicação podem ter um peso maior ou menor na avaliação de um SOTR, como: suporte e documentação, reputação dos desenvolvedores, custo, integração com sistemas legados, etc, estes corroboram com o número de fatores que tornam a comparação entre SOTR algo, no mínimo, confuso.

Avaliações de desempenho mais completas de SOTR normalmente são baseadas na observação do sistema como aplicação destinada 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.

Um SO tem como principal finalidade fornecer um ambiente em que certas funcionalidades do sistema estejam ocultas ao desenvolvedor, proporcionando-lhe uma camada de abstração sobre a qual possa maximizar seu trabalho utilizando uma interface de programação mais amigável. Além desta finalidade comum a todos os SO, um SOTR, deve criar um ambiente de desenvolvimento previsível e determinístico qualquer que seja a carga do sistema a fim de que aplicações de tempo real possam ser executadas e seus requisitos temporais sejam respeitados.

Um SOTR é dito previsível quando fornece garantias de que tarefas de tempo real sejam executadas respeitando os prazos definidos pelo pior caso de tempo de execução (*Worst Case Execution Time* ou WCET, e determinístico quando torna possível conhecer os valores de tempo máximo de execução de cada uma das suas chamadas de sistema, os valores de tempo refente as latências que surgem devido as operações do sistema. É importante notar que estes valores de tempo podem variar de acordo com a arquitetura do

processador em que o sistema é executado, com o código gerado pelo compilador utilizado para gerar o sistema, com os algoritmos utilizados e com a qualidade das suas respectivas implementações.

Como geralmente são sistemas reativos, é comum a SOTR lidar com interrupções, sejam provocadas por estímulos externos, sejam provocadas por atividades concorrentes. A latência de interrupção corresponde ao intervalo de tempo decorrido entre a chegada de um sinal de interrupção ao processador e o início da rotina de tratamento correspondente. É característica desejável de um SOTR que este intervalo de tempo seja o menor possível.

A latência por troca de contexto é uma característica intrínseca ao escalonador do sistema e surge quando ele alterna o uso da CPU entre os diferentes processos em atividade. Pode ser definida como o intervalo de tempo decorrido entre o momento em que um processo é interrompido e o momento em que um novo processo executa sua primeira linha de código. Bons algoritmos de escalonamento, e consequentemente suas implementações, devem comutar a execução de tarefas no menor tempo possível evitando assim a sobrecarga da CPU com esta tarefa.

Embora o senso comum nos diga que um SOTR deva reduzir a latência entre um estímulos e suas respectivas respostas e aumentar a velocidade do sistema como um todo, provê estas características não são seu principal objetivo, embora sejam bastante desejáveis, e seus valores são de grande importância no que diz respeito ao atendimento de requisitos relacionados com a responsividade do sistema ou seja ao ser submetido a um estímulo, interno ou externo, um SOTR, deve garantir que uma resposta deve ser rápida o suficiente para atender a estes requisitos.

3.1 Parâmetros e premissas utilizados na Avaliação

3.2 O Ambiente de Testes

Na comparação entre diferentes SO, e mais especificamente de SOTR, é importante que a configuração do hardware utilizado nos testes propostos seja igual ou no mínimo equivalente, isso garante que os resultados obtidos sejam consistentes e que não tenham sido influenciados por funcionalidades específicas de uma determinada configuração

de 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 Linux e a distribuição escolhida foi 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 de forma a favorecer uma das soluções testadas. 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. Vale mencionar que não existem versões do kernel que sejam suportadas por ambas as soluções simultaneamente.

3.2.1 Testes Preliminares

As soluções estudadas foram submetidos a testes preliminares, utilizados tanto para identificar possíveis falhas nos processos de instalação, como para identificar funcionalidades do *kernel* que pudessem alterar o desempenho e o determinismo do sistema. Nestes testes o principal parâmetro observado foi a latência (QUAL LATÊNCIA!!!) do sistema. Embora a redução de latência (QUAL!!!), como dito anteriormente, não seja um dos principais objetivos de um SOTR, é de vital importância, junto com os outros parâmetros vistos anteriormente, que seus valores sejam conhecidos para que o sistema possa ser classificado como determinístico. Os valores de latência obtidos como resultado dos testes preliminares também serviram como referência para avaliar a qualidade e a uniformidade dos resultados obtidos com os benchmarks desenvolvidos neste trabalho.

Os testes preliminares foram executados por meio de ferramentas recomendadas e fornecidas pelos próprios desenvolvedores dos sistemas avaliados. Foram utilizados os programas: *Latency*, para testes executados no *RTAI* e *Cyclictest*, para testes executados no *kernel* com o patch *PREEMPT-RT* aplicado. 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* é fornecido junto a suíte *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 repositório do *kernel*.

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 o aumento da latência do sistema, por meio da função *FTRACER*. Este recurso foi 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, reproduzindo 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 produzir esse cenário foi a proposta por Geusik Lin. Esta abordagem, além de proporcionar o uso de 100% do processador, possui uma construção simples que utiliza um conjunto de instruções e programas que já se encontram pré instalados na maioria das distribuições *Linux*.

3.2.2 Configuração do *Kernel*

Rever funcionalidades apontadas como vilãs da latência!

Algumas 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 monoprocessados a funcionalidade do *kernel* que concede 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 (High Resolution Timer Support) foram habilitadas. Como sistemas de tempo real normalmente são sistemas reativos, seguindo as recomendações da configuração do kernel, a opção Clock Frequency foi configurada para 1000 Hz.

3.3 *Benchmarks*

4 RESULTADOS

5 CONCLUSÕES

5.1 Trabalhos Futuros

Bibliografia

CAICEDO, Angela. **Managing Multiple Screens in JavaFX**. 2013. Disponível em: <https://blogs.oracle.com/acaicedo/entry/managing_multiple_screens_in_javafx1>. Acesso em: 3 out. 2016.

ECKSTEIN, Robert. Java SE Application Design With MVC. **Oracle Technology Network**, 2007.

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

FOUNDATION, Free Software. **O que é um software livre?** 2016. Disponível em: <https://www.gnu.org/philosophy/free-sw.html>. Acesso em: 16 fev. 2017.

GROUP, Statistic Brain. **Attention Span Statistics**. 2016. Disponível em: <http://www.statisticbrain.com/attention-span-statistics>. Acesso em: 17 fev. 2017.

HOMMEL, Scott. **Implementing JavaFX Best Practices**. 2014. Disponível em: <http://docs.oracle.com/javafx/2/best_practices/jfxpub-best_practices.htm>. Acesso em: 21 set. 2016.

MOREIRA, Andeson Luiz Souza. **Análise de Sistemas Operacionais de Tempo Real**. 2007. Mestrado – Universidade Federal De Pernambuco.

PRESSMAN, Roger S. **Engenharia de Software: Uma abordagem profissional**. [S.l.]: McGraw Hill, 2011.

A APÊNDICE

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

A.1 Apêndice 1

1. Item 1