Relatório Intermediário

Kernel Acadêmico Version 0.5

Universidade Federal de Minas Gerais (UFMG)
Instituto de Ciências Exatas (ICEx)
Departamento de Ciência Da Computação (DCC)

- Alexandre Pretti
- Carolina Coimbra
- Francisco Malaguth
- Luis Felipe Cabral

Conteúdo

1. Introdução

“So, you know C and you know some UNIX basics, but do you know LINUX?”

Linux Kernel Internals

São tantas as tarefas com as quais um estudante precisa lidar dentro da faculdade que, às vezes, parece impossível resolver tudo de uma vez. No entanto, a tecnologia pode ser muito útil para ajudar a conciliar todas essas, bem como fornecer materiais de estudo e ferramentas úteis para ajudar em trabalhos. Essa ferramenta deve ser eficiente e estar sempre disponível.

O Kernel Acadêmico seria então uma modificação na sua estrutura do kernel atual visando criar um melhor desempenho do computador durante o uso desses programas, fazendo com que eles abram e fechem mais rapidamente, sejam processados com alta prioridade e encontrem-se ao alcance do usuário a qualquer momento. Ou seja, a proposta seria melhorar o desempenho dos programas amplamente utilizados pelos estudantes permitindo que os mesmos reduzam o tempo perdido com sistema lentos.

Dessa forma, o presente relatório visa apresentar o andamento do trabalho baseado no que já foi proposto no Relatório Inicial. Nessa etapa, o foco voltou-se para a compilação do kernel bem como o estudo mais aprofundado dos possíveis módulos afetados pela proposta de melhorar o desempenho dos programas mais utilizados por estudantes universitários.

2. Desenvolvimento

"...you might as well skip the Xmas celebration completely, and instead sit in front of your linux computer playing with the all-new-and-improved linux kernel version."

Linus Torvalds

Com os objetivos claros e uma meta bem traçada, começamos nosso trabalho de desenvolver o Kernel Acadêmico. Para um melhor planejamento do trabalho, um fluxograma englobando as principais atividades necessárias para se concluir o Kernel Acadêmico foi criado e está representado na figura abaixo.

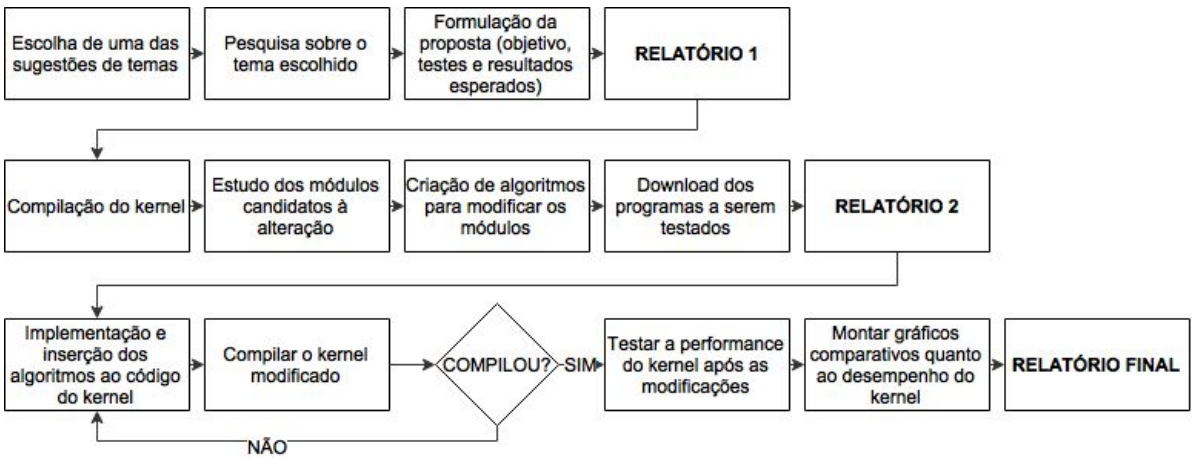

Imagem 2.1 Fluxograma do trabalho

Conforme mostrado no fluxograma acima, o Primeiro Relatório consistiu de uma proposta de modificação do kernel do Linux. Dessa forma, o que será apresentado a seguir envolve o desenvolvimento inicial dessa proposta que, ao final, espera-se atingir resultados satisfatórios que serão apresentados no Relatório Final.

3. Compilação do kernel

“Eu amo Máquina Virtual!”
Luis Felipe Cabral

Antes de ser possível compilar várias iterações do kernel utilizando uma máquina virtual, foi necessário configurar a BIOs do computador no qual seria realizado o projeto, ativando sua “virtualization technology” (VTx e VTd).

Após tal configuração, a máquina virtual foi ativada para iniciar a etapa de compilação do kernel propriamente dita. Como forma de exemplificar todas as dificuldades enfrentadas durante as tentativas de compilação de kernel, uma tabela foi criada e nela todos os kernels criados foram registrados com suas respectivas datas de “nascimento” e “falecimento” seguido do que foi aprendido com cada um deles (conquistas) e a causa do óbito de cada um.

Nome	Data de Nascimento	Data de falecimento	Conquistas	Causa do Óbito
Kernel 1	05/10/2016	07/10/2016	Compreendemos o uso da máquina virtual e configurações do kernel.	Corrompimento do OpenSSL durante a fase de compilação.
Kernel 2	07/10/2016	07/10/2016	Aprendemos a importância de alocar memória corretamente.	Falta de memória para a instalação da máquina virtual.
Kernel 3	07/10/2016	13/10/2016	Primeiro kernel a ser compilado com sucesso!	Ausência de mouse.
Kernel 4	13/10/2016		Primeiro kernel a ser parcialmente modificado e compilado com sucesso (mouse funcionando)!	
Kernel 5	19/10/2016		Primeiro kernel a sofrer modificações ainda não finalizadas!	

De fato, cada kernel contribuiu um pouco na construção do conhecimento acerca da compilação. Nesse caso, entende-se que a configuração inicial do kernel na máquina virtual foi um problema que demandou esforço e aprendizado de conteúdo extra. Além disso, o uso de uma máquina virtual foi fundamental nessa etapa tendo em vista que sem a mesma seria extremamente difícil recuperar a máquina após as tentativas frustradas de se compilar o kernel.

Logo após esses primeiros passos, surgiram novos desafios, envolvendo a etapa de identificação e estruturação das modificações dos módulos envolvidos na proposta de criação do Kernel Acadêmico. Essa fase será melhor apresentada na próxima seção onde serão discutidos os módulos que, até o momento, foram identificados como potenciais alvos de modificações. Entretanto, é válido mencionar um novo problema já encontrado referente à como encontrar partes específicas dentro do código do kernel, pois a documentação é confusa e muito fragmentada.

Sendo assim, os desafios envolvidos nesse trabalho não se restringem às configurações da BIOS, máquina virtual e a própria compilação do kernel mencionadas anteriormente, uma vez que a fase da implementação, já iniciada, demonstrou que possivelmente demandará ainda mais esforço do grupo tendo como base o que já foi definido e será mostrado na seção seguinte.

4. Implementação

“Talk is cheap. Show me the code.”
Linus Torvalds

A modificação do kernel com objetivo de melhorar o desempenho de programas acadêmicos é abordado de duas formas distintas, através de módulos que buscam otimizar suas dependências e uma implementação baseada em um sistema de tempo real.

A otimização das dependências ocorre a partir da melhor interação entre o kernel e os aplicativos em questão, uma vez que, busca-se trazer as dependências comuns entre os programas para dentro do kernel de forma a tornar seu acesso mais rápido e eficiente.

Outra forma de melhorar o desempenho desses programas específicos é alterar o kernel para tratá-los de maneira semelhante a um sistema de tempo real, priorizando assim sua execução em detrimento dos demais processos.

4.1. Módulos do Kernel

O kernel do Linux que se divide em módulos, conforme representado na imagem abaixo, sendo assim o primeiro passo para se implementar as mudanças descritas acima envolve o estudo dos mesmos.

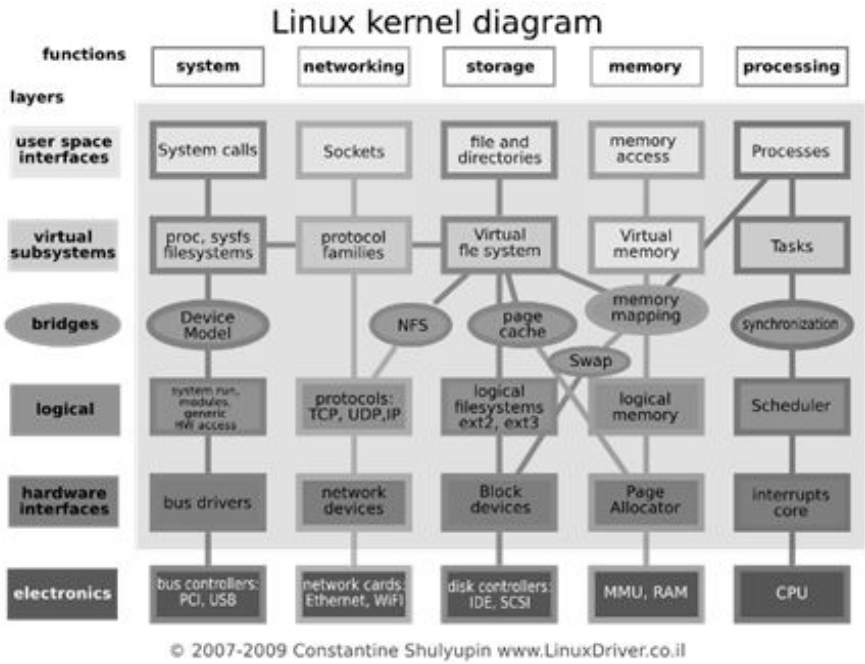

Imagem 3.1 Diagrama do kernel

O primeiro módulo a ser estudado foi o RTC (Real Time Clock) cujas funções serviram de base para a criação de algoritmos para tratamento de tempo real que favoreçam os processos acadêmicos.

Em sequência a pesquisa se voltou para como criar e modificar os módulos do kernel de forma a integrá-los ao sistema sem prejudicá-lo, o que, de certo modo pode não ser possível devido a necessidade de priorizar alguns processos.

Para decidir quais módulos seriam criados com objetivo de facilitar o acesso dos programas à suas dependências foi pesquisado quais delas são de maior importância e comuns entre esses programas para implementação. Como forma simples de visualizar essas dependências foram utilizados como depmod e afins.

Em adicional, vale citar que o módulo ZRAM nativo do kernel foi ativado e integrado ao sistema com objetivo de tornar mais eficaz o uso da partição de memória swap para os programas em questão. Isso ocorre devido ao fato dele criar um bloco na memória RAM que funciona com uma unidade de disco virtual, porém é comprimido e armazenado na memória RAM.

Os módulos em desenvolvimento no Kernel Académico serão compilados e então carregados para o sistema para teste. Os módulos são simples e seguem umas estrutura básica:

Estrutura de um módulo do Kernel

```
1   #define __KERNEL__
2   #define MODULE
3   #include <linux/module.h>
4
5   /* carga do módulo */
6   int init_module (void)
7       // Início do módulo...
8       return 0;
9   }
10
11    /* descarga do módulo */
12    void cleanup_module (void) {
13        // Fim do módulo...
14    }
```

Imagem 3.2 - Estrutura de um módulo do Kernel

A simplicidade dos módulos deve-se ao fato deles utilizarem dependências existentes no sistema, porém a principal dificuldade se encontra em integrá-las ao kernel.

5. Ferramentas de teste

“The Linux philosophy is "laugh in the face of danger". Oops. Wrong one. "Do it yourself". That's it. ”
Linus Torvalds

O estudo para se desenvolver o Kernel Acadêmico envolveu também as ferramentas que serão utilizadas ao longo dos testes.

Levando em conta o pouco tempo para a realização do projeto, optou-se pela utilização de ferramentas de benchmark, mesmo não sendo muito aconselhadas para medir o desempenho de programas específicos. Apesar de não serem as ferramentas mais adequadas, espera-se observar uma alteração na eficiência do kernel de acordo com os resultados indicados pelas ferramentas.

Após um breve pesquisa, foram selecionadas quatro ferramentas de benchmarking com as quais seria possível comparar o desempenho do sistema antes e depois das modificações realizadas.

- **Hardinfo:** O Hardinfo traz diversas funcionalidades para diagnósticos de hardware e software, além de centenas de informações a respeito da sua máquina. O programa divide suas funcionalidades em quatro categorias principais: Computer, Devices, Network e Benchmark. Sendo que no que diz respeito à testes, o mesmo fornece seis ferramentas distintas que rodam testes diferentes e estressam a CPU de forma distinta das demais. Além disso, os testes consistem na execução de cálculos e tarefas consideradas complexas.
- **GtkPerf:** O GtkPerf foi criado para testar o desempenho da parte gráfica de um computador. A ideia do GtkPerf é executar uma sequência de testes comuns, como abrir caixas de diálogo, simular cliques do mouse, navegar por textos e desenhar uma série de imagens de diversas cores e formatos na tela. Dessa forma, é possível mensurar a velocidade dos dispositivos gráficos instalados na máquina.
- **Phoronix Test Suite:** O Phoronix Test Suite realiza muito mais testes no hardware. É possível visualizar diversas informações a respeito da máquina, tais como: processador, placa-mãe, memória RAM, disco rígido, placa gráfica, áudio e dados de rede.
- **UnixBench:** O UnixBench mede o desempenho geral do Unix. Realiza testes com Input e Output e performance multitarefas do kernel.

Apesar dessas ferramentas não serem as únicas desenvolvidas para Linux, são apontadas como as mais utilizadas pela comunidade e pelos sites especialistas em análises de hardware. Sendo assim, optou-se por utilizá-las para avaliar o desempenho sob as diversas ópticas proporcionadas pelas mesmas.

---

6. Conclusão

“If Microsoft ever does applications for Linux, it means I’ve won”
Linus Torvalds

Esse trabalho foi proposto com o objetivo de melhorar a experiência dos usuários, principalmente os estudantes, ao utilizar softwares úteis à vida acadêmica.

Até o momento foi feita uma abrangente pesquisa envolvendo os módulos do kernel bem como as ferramentas de teste que serão úteis nesse contexto. Além disso, antecipando possíveis problemas que poderiam ser encontrados durante a compilação do kernel, uma pré-compilação foi realizada a fim de se obter uma familiarização com essa tarefa que, apesar de ser a mais simples ao longo do trabalho, demandou esforço e proporcionou alguns desafios que agregaram como experiência envolvendo sistemas operacionais.

Os próximos passos envolver de fato a codificação baseado nos módulos estudados e a compilação do kernel modificado. Caso a fase de compilação do kernel modificado resulte em sucesso, será possível testar e comparar o desempenho dos sistemas antes e após as mudanças utilizando as ferramentas de benchmark estudadas.

Sendo assim, de forma geral, o trabalho ainda que em fase de desenvolvimento, contribuiu muito com o aprendizado daqueles que não tinham tanta intimidade com o sistema operacional Linux e acrescentou mais experiência para aqueles que são mais familiarizados. Ou seja, está sendo, de fato, um trabalho desafiador mas enriquecedor.

7. Referências Bibliográficas

BOVET, Daniel P.; CESATI, Marco. Understanding the Linux Kernel. 3rd ed. O'Reilly Media, Inc. 2006.[1]

CAMPOS, Augusto. O que é o Linux. BR-Linux. Florianópolis, março de 2006.[2]

LOVE, Robert. Linux Kernel Development. 3rd ed. Pearson Education, Inc.2011.[3]

MARTINS, Ronnie P. B. Geekbench - Testando o Desempenho do Linux. 2009.[4]

PRACIANO, Elias. Como melhorar o desempenho do Linux ativando o zRam.[5]

BRITO, Edivaldo B. ZRam, no Linux para melhorar o desempenho do sistema.[6]

SILVA, Gleydson M. Guia Foca GNU/Linux. Capítulo 16 - Kernel e Módulos.[7]

MARINHO, Fábio. Como melhorar a performance do sistema Ubuntu Linux com preload.[8]

BALSA, Andre D. Linux Benchmarking HOWTO.[9]

[1] Disponível em: <https://goo.gl/3aCFOP>.
[2] Disponível em: <goo.gl/POy5LW>.
[3] Disponível em: <https://goo.gl/mNukvd>.
[4] Disponível em: <goo.gl/HNnJ1n>.
[5] Disponível em: <goo.gl/XLfnxG>.
[6] Disponível em: <goo.gl/KXq0iz>.
[7] Disponível em: <goo.gl/idKsN7>.
[8] Disponível em: <goo.gl/A5gLpu>.
[9] Disponível em: <goo.gl/9aGBND>.