Trabalho Prático de Sistemas Operacionais

Implementando uma melhoria no Kernel do GNU/Linux

Arthur Lopes

Pedro Pires

Relatório I: Proposta de Trabalho

Introdução

O Sistema Operacional é o software responsável pela definição e execução das diretrizes básicas de uso e segurança de um computador. Ele age na forma de um intermediário entre o usuário e o hardware, abstraindo completamente o hardware (ou seja, tornando-o "invisível" para o usuário), de tal forma que permite que usuários de todos os tipos utilizem de forma idêntica diferentes organizações de computadores e até mesmo diferentes arquiteturas de computadores. O Sistema Operacional é também o responsável pela gerência direta dos recursos computacionais do hardware, tais como tempo de processamento e ocupação de memória.

O Kernel de um Sistema Operacional é o seu núcleo, sua camada mais próxima ao hardware e, portanto, é o responsável pela administração direta de recursos do hardware. O Kernel de um sistema operacional moderno geralmente é dividido em vários módulos, cada um responsável pela gerência de um recurso específico.

O GNU/Linux é um dos principais Sistemas Operacionais atualmente, não apenas por sua reconhecida estabilidade e segurança, mas também por ser o mais difundido Sistema Operacional de código aberto atualmente. O fato de ter seu código totalmente aberto permite que ele possa ser estudado e até mesmo modificado, o que o torna extremamente atraente para trabalhos acadêmicos ou situações onde é preciso garantir alguma funcionalidade ou otimização específica.

Proposta

A proposta do trabalho é modificar o Kernel do GNU/Linux para tentar melhorar algum aspecto seu, seja tentando aprimorar algum algoritmo, ou seja modificando-o para se adequar melhor a alguma situação específica.

Uma das principais tarefas realizadas por um Sistema Operacional moderno é o gerenciamento de memória primária, mais especificamente o gerenciamento da Memória Virtual. Como todos os demais recursos computacionais, a memória primária possui um limite, de forma que é importante conseguir aproveitá-la da melhor maneira possível. Através da abstração conseguida com o uso de Memória Virtual, é possível utilizar memória secundária como se fosse memória primária. Essa abstração extende a quantidade de memória disponível no computador, mas possui um alto custo de tempo sempre que houver um acesso à memória secundária. Para minimizar este problema é preciso minimizar os acessos à memória secundária, tentando garantir que todos os dados necessários em um determinado momento estejam presentes na memória primária. Para obter um resultado perfeito sera preciso poder prever com exatidão quais dados serão necessários em um próximo instante, e isto é impossível. Entretando, existem várias técnicas que procuram prever de forma razoável quais dados têm mais chance de serem requisitados no futuro próximo.

Simplificando bastante o algoritmo, podemos dizer que o GNU/Linux utiliza o algoritmo LRU (Least Recently Used) para tentar determinar quais dados são menos prováveis de serem utilizados e, portanto, bons candidatos a serem removidos da memória primária caso haja necessidade de liberar espaço para a entrada de novos dados. Este algoritmo procura explorar o princípio de localidade temporal, segundo o qual um dado utilizado mais recentemente tem maiores chances de voltar a ser utilizado em um futuro próximo do que um dado que já não é utilizado a mais tempo. Buscaremos aprimorar este algoritmo ao explorar melhor a previsibilidade do uso da memória. O Kernel possui um módulo chamado escalonador, que é o responsável por determinar como será a execução dos processos, ou seja, entre outras coisas ele determina a ordem na qual os processos serão executados. Assim, um dado associado a um processo que está no final da fila do escalonador terá grande chance de demorar mais para ser utilizado do que um dado associado a um processo que está no início da fila. Mesclando esta informação com o princípio de localidade temporal é possível construir um algoritmo LRU modificado de forma a obter melhores resultados do que no LRU tradicional.

Implementação e testes

Para a implementação, será utilizada a mais recente versão do kernel do GNU/Linux (3.0.4). Os testes serão executados em máquinas virtuais, pois dessa forma podemos definir a quantidade de memória disponível para o sistema operacional, além de a recuperação ser mais fácil, em caso de erro.

Para os testes serão utilizados programas que demandam muitas trocas de páginas da memória, para que a eficiência do algoritmo de swap possa ser medida. O algoritmo de swap é o responsável pela escolha das páginas que serão retiradas da memória primária. Inicialmente serão feitos testes com o kernel sem modificações, e posteriormente os mesmos testes serão feitos no kernel modificado.

Como o algoritmo LRU utilizado atualmente leva em conta somente a localidade de referência temporal dos acessos à memória, esperamos que a modificação produza um ganho de desempenho, melhorando assim o uso da memória.