Implementação de um escalonador de processos baseado em passos largos

Ivair Puerari[1], **Jeferson Schein**[2]

[1]Universidade Federal Fronteira Sul(UFFS)
Chapecó – SC – Brasil

[2]Ciência da Computação
Universidade Federal Fronteira Sul (UFFS) – Chapecó, SC – Brasil

ivaair@hotmail.com, schein.jefer@gmail.com

Abstract. *This article refers to the implementation of a process-based Stride Scheduler, the scheduler has the function perform the distribution of resources, so that all processes will have the opportunity to use, more or less times the resource. Given this it is proposed a new approach, using the XV6 operating system, developed by Massachusetts Institute of Technology. The scheduler based on stride, deterministically, using of features between the processes, makes this distribution. Is demonstrated the planning and implementation of this prototype, as well as its peculiarities during development and the finding of influences between the choice of algorithms for the scheduling of processes in an operating system.*

Resumo. *Este artigo artigo refere-se a implementação de um escalonador de processos baseado em passos largos, o escalonador tem a função de realizar a distribuição de recursos, de forma que todos os processos tenham a oportunidade de usar, mais ou menos vezes o recurso. Diante disso é proposto uma nova abordagem, utilizando do sistema operacional XV6,desenvolvido pelo Instituto Tecnologia de Massachusetts. O escalonador baseado em passos largos, deterministicamente, utilizando de características entre os processos, faz esta distribuição. É demonstrado o planejamento e implementação deste protótipo, bem como suas particularidades durante o desenvolvimento, e a constatação da influencia entre a escolha de algoritmos para o escalonamento de processos em um sistema operacional.*

1. Introdução

Em um sistema operacional, por traz de um gráfico visual que na maioria dos sistemas operacionais apresentam para os usuários, há diversos mecanismos para que o sistema operacional funcione como os usuários desejam, partir disso, para suprir essa necessidade, a utilização de processos é usado em todo sistema operacional, para as movimentações e chamadas, afim de resolver tarefas.

Com isso, logicamente há um acumulo de processos, bem como uma quantidade grande de novos processos a todo momento. E também existem processos onde se tem alguma relevância na questão de tempo, ou seja, processos que precisam ser resolvidos quase que instantaneamente, como a de inicialização de um sistema operacional ou processos que dependem de um outro processo, para terminar a sua tarefa, que sejam dependentes.

Com estas informações pode se retirar, que o mecanismo de escalonamento de processos é de grande valia dentro de um sistema operacional, pois um escalonador de processo tem o papel, por meio de um algoritmo, equilibrar a utilização de um recurso de forma eficaz, decidindo quem terá a oportunidade de utilizar o recurso, assim decidindo quem deve ser o atendido e melhorando o sistema operacional.
O escalonamento baseado em passos largos, busca isso, uma melhor divisão de tempo e recursos, onde que por meio da utilização de formas de sinalizações de prioridades dentro de um processo, é feito de forma deterministicamente o escalonamento dos processos. Nas próximas seções será exposto o problema, e critérios contidos no escalonamento, que fazem parte no escalonamento baseado em passos largos, bem como, o planejamento de uma solução para o problema, e a implementação de um escalonador que valide a solução, utilizando de testes aplicados no sistema operacional XV6.

2. Problema

O intuito deste artigo é relatar a forma de planejamento e implementação de um escalonador de processos baseado em passos largos, (stride scheduler), utilizando como sistema operacional tipo Unix XV6, criado pelo instituto de Tecnologia de Massachusetts, nos Estados Unidos da América.
O escalonamento por passos largos, assume o papel de um escalonamento, que deterministicamente aloca espaço de tempo proporcional, para execução da tarefa proposta do processo, variando conforme a quantidade de bilhetes que o processo possui.
Assim cada processo deve conter uma quantidade fixa de bilhetes na instanciação do processo, assim calcula-se o passo (stride) de cada processo. Logo é necessário adicionar a sua passada, o valor anteriormente contido mais a sua passada, passada essa encontrada sendo o quociente entre a divisão de um numero constante pelo seu numero de bilhetes contidos.
Inicialmente cada processo tem passada igual a zero, ”stride.pass = 0”é um exemplo, com isso o escalonador escolhe o processo com a passada menor, para que seja o processo executado, portanto, qualquer processo entre todos os processos podem inicialmente ser o escolhido, e ficando de acordo a abordagem implementada, a prioridade por ser qual processo o primeiro escolhido, a partir disso é feito a operação de soma da passada do processo e assim sucessivamente.

3. Planejamento

Para uma solução ao problema de implementação de um escalonador de processos, primeiramente, foi realizado um estudo sobre o método de escalonamento passos largos, quais suas especificações e requisitos para o desenvolvimento.
Como anteriormente, em um primeiro momento, foi realizado um estudo em cima do manual do XV6, para que fosse redigido o artigo ”Implementação de um escalonador de processos baseado em loteria”. Foi possível que pulássemos esta etapa, e que houvesse uma maior concentração na implementação. Após buscas, com o objetivo de obter o entendimento do método de escalonamento por passos largos, foi verificado quais blocos de códigos deveria ser mudado ou deixados como eram, que não implicavam na solução.
A proposta para planejamento foi seguir as características do escalonamentos em passos largos. Logo, iniciando os processos com uma quantidade de bilhetes, onde o usuário ou qualquer outro criador de processo, passe como informação a quantidade bilhetes que

aquele processo ira conter.
Após isso, a construção de um algoritmo ou mecanismo para que se calcule o passo de cada processo, bem como, inicie os mesmos. Em outro momento, haverá a escolha entre como proceder na escolha de qual processo ser escolhido quando todos tem sua passada iguais, e posteriormente realizar a atribuição da sua nova passada, que seria a soma de sua passada mais o passo, originado pelo numero de bilhetes atribuído ao processo.

4. implementação

Na implementação, foi constatado que os blocos de códigos que seriam necessários mudar para que a solução fosse eficiente, se encontram nos arquivos ”proc.c”e ”proc.h”, que também utiliza outros arquivos que dão suportes as funções contidas.
Como anteriormente proposto, iniciamos a implementação para que os processos sejam possíveis conter os bilhetes, bem como, atribuir os o bilhetes ao processo, nas próximas seções deste artigo sera relatado de como houve essas mudanças, informando suas particularidades, e disponibilizando os trechos de códigos para que ilustre melhor a explicação de como procedeu a implementação do escalonador de processos baseado em passos largos.

4.1. Bilhetes para processos

Houve a necessidade de mudanças na função fork no arquivo ”proc.c”, a função fork é a responsável pela criação de processos, e tem como seu retorno o id do processo pai, criador do novo processo, aquele processo que originou o chamado da função.
Para a solução funcionar, não foi necessário criar um parâmetro dentro da estrutura processo, mas apenas repassar a informação quando realizado uma chamada para a função fork, como parâmetro da função, um numero inteiro, representando os bilhetes do novo processo.
Cada processo pode conter no máximo 1000 bilhetes,e no minimo 50 bilhetes, caso houver algum erro, e não haver o numero de bilhetes de um processo, como default, sera usado um numero médio de bilhetes a quantidade de 500 bilhetes para o processo.
Dentro da função fork existe uma chamada para outra função, função allocproc , nela realmente é criado um novo processo, onde tem os parâmetros do processo, instanciados. Sabendo disto, logicamente deve ser passado a quantidade de bilhetes como parâmetro para a função allocproc também, pois é de suma importância que a quantidade de bilhete dada ao processo, esteja no momento de criação do processo, já que os bilhetes significam o quanto um processo é prioritário ou não.
Prioridade seria ter o privilegio de executar cedo, e ter fatias maiores de tempo para que o processo possa executar suas tarefas, ou seja, obtendo o direito sobre o recurso requerido para que o processo termine suas tarefas rapidamente.

4.2. Passo e passada de um processo

Passo em um processo, é o quociente de uma divisão entre um numero constante e a quantidade de bilhetes que o processo possui, neste caso foi utilizado um numero constante 50000, escolhido arbitrariamente. O passo é o resultado desta operação, do quociente desta divisão. Para exemplificar, e que a ideia seja abstraída para um melhor entendimento, anteriormente a demonstração da implementação da solução, considere:
Há três processos, A, B, C, processo A com 100 bilhetes, processo B com 250 bilhetes e

Figura 1. Paramêtro da função fork.

processo C com 50 bilhetes. O processo A ira ter um passo de 500, processo B um passo de 200 e o processo C com passo 1000. Como dito anteriormente um escalonamento em passos largos, os processos tem sua passada todos iniciados com zero, para uma forma didática iremos elaborar a ordem de forma que foi exposto os processos, em ordem léxica. Então o processo A seria o primeiro, assim sua passada que é igual a 0, mudaria para 0 + passo, que seria 500, logo passaríamos para o menor, como o escalonamento por passos largos pede, com isso e também por ordem léxica, o processo B seria atualizado sua passada que antes era zero para 200 e após isto, é escolhido o processo C que tem sua passada igual a 0 ainda, que atualizada para 1000. Logo em uma próxima rodada, o processo B com a menor passada no momento, seria o escolhido, e no minimo por duas rodadas sendo o escolhido.
Então quanto maior o numero de bilhetes, menor é sua passada,ou seja, mais tempo utilizando do recurso para a tarefa proposta pelo processo.

4.2.1. Passo

Para implementação, que um processo possa conter um passo, foi necessário que o fosse atribuído um novo parâmetro dentro da estrutura processo, que esta localizado no arquivo ”proc.h’, lá foi criado dentro da estrutura a variável step.
Como cada processo tem seu próprio passo, partindo da quantidade de bilhetes pertencentes, intuitivamente foi achado melhor que dentro da função allocproc, já que instancia os parâmetros da estrutura processo, que ali seja feito a atribuição do valor de passo do processo. O atributo bilhetes já é passado como parâmetro da função, logo a partir da utilização de algumas condições de fluxos, foi implementada a operação para descobrir o passo do processo e atribuído ao step o seu valor de passo.

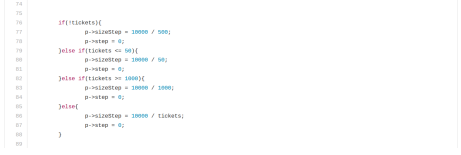

Figura 2. Função allocproc, operação implementada para o passo.

4.2.2. Passada

Para implementação da passada, foi feito um estudo aonde melhor se encaixaria, foi visto que propriamente na função scheduler é realizado a escolha do processo entre a tabela de processos do sistema operacional. Mas, anteriormente foi realizado a atribuição de uma nova variável a estrutura processo chamada sizeStep, representando a passada, no arquivo ”proc.h”. Após isso, e como especifidade do problema toda passada deve ser igual a 0 para todos os processos, assim, logo quando é atribuído o valor de passo do processo, também é atribuído a sua passada o valor 0 para a variável sizeStep que pode ser verificada na **Figura 2**.

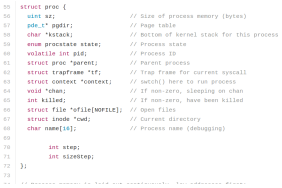

Figura 3. Varivel step (Passo) e sizeStep(Passada) na estrutura processo.

Na função scheduler, Primeiramente deve se levar em conta que todo processo inicialmente terá 0 como sua passada, logo deve haver um critério de desempate. Para que não haja influencia ou que seja algo natural, foi decidido que a primeira passada 0, de qualquer processo seria o escolhido, ou seja, o primeiro processo encontrado com passada 0 é o escolhido. A implementação para esta logica ocorreu de forma simples, comparamos todos os processos com a utilização de condições de fluxos, dentro de um for que passa pela tabela de processos do sistema operacional.
Com o processo escolhido, é necessário ser atualizado a passada do processo,então implementamos algo como se fosse uma flag, para que quando fosse encontrado o processo com a menor passada, ele entre neste bloco de código para que seja feita a operação. Assim, a passada do processo é somado com o passo do processo. Na possibilidade de haver um overflow de inteiros, pela causa de um acúmulo grande de somas, maior que a quantidade de memoria que um inteiro tem, tratamos de forma simples, pois achamos que seja algo difícil de acontecer, mas possível, para que caso acontece o processo possa ser reiniciado a contagem de sua passada.

Figura 4. Função scheduler, operação de soma de passada.

5. Testes aplicados no xv6

Os testes realizados aconteceram em dois momentos, primeiro teoricamente um teste de mesa, para encaminhar as implementações, bem como, dar garantia que se tem uma logica. Após a implementação foi criado um arquivo ”test.c”. No ”test.c”, foi criado com o intuito de simular a interação de um usuário com o sistema operacional, nele foi criado diversos processos com diferentes quantidades de bilhetes, imprimindo sua identificação e quantidade de bilhetes, afim de poder medir se conforme a quantidade de bilhetes do processo, ele terminaria de forma mais rápida , devida a prioridade dada pelo numero de bilhetes que contém.
Também foi feito um teste onde utilizamos o sistema operacional XV6 original, sem mudanças, e o sistema operacional XV6 que foi feita a implementação do escalonador de processos baseados em passo largos, houve a criação de três processos, iguais nos dois sistemas operacionais, com a utilização de um timer, pode ser notado que houve uma diferença de 0,5 segundos, do termino do segundo sistema operacional aquele que tinha mudanças no seu escalonador para o primeiro sistema operacional original.
Assim possível visualmente tirar alguma conclusão referente a implementação bem sucedida, e uma solução eficiente ao problema do desenvolvimento de um escalonador de processos baseados em passos largos.

6. Conclusões

O escalonamento de processos, tem um grande impacto dentro de um sistema operacional, na sua eficiência. O escalonamento de processos baseados em passos largos, traz de uma forma determinística essa alocação de recursos, seja eficaz. Em testes foi verificado que houve um ganho em performance, o escalonador influenciando diretamente no tempo de resposta dos processos, e quanto mais a quantidade de processos crescem, melhor se torna a resposta. Assim a utilização da estratégia de um escalonador baseado em passos largos, pode melhorar a performance de um sistema operacional, demonstrando que a utilização de uma estratégia de escalonamento, deve ser escolhida de forma sensata, para que o sistema operacional seja eficiente em suas tarefas.

7. Refenrencias

Manual xv6 - xv6a simple, Unix-like teaching operating system.
TANENBAUM, A. S. Sistemas Operacionais Modernos. 2. ed. São Paulo: Prentice-Hall do Brasil, 2003.
Waldspurger, Carl A.; Weihl William E.; Stride Scheduling:Deterministic Proportional-Share Resource Management. June 22, 1995.