Escalonamento por loteria no XV6

Rômulo Rossi, *Student, UFFS,*

***Abstract*—At present it may seem difficult to imagine a computer that only allows one application at a time, but in fact there has been a remarkable historical path to the current level of process parallelization. In a multi-tasking operating system, one of the key performance maintenance challenges is the policy of distributing processor usage across processes. The XV6 is a didactic UNIX operating system developed by the Massachusetts Institute of Technology in the summer of 2006. The idea behind this scholarly work is to introduce a process scheduler based on the lottery scheduling algorithm proposed by Waldspurger in 1994. Scheduling by lottery is a probabilistic allorit, its idea is to assign tickets to the processes, to draw a ticket and to give the CPU to the winning ticket process.**

***Keywords*—Escalonamento, Loteria, XV6.**

I. Introdução

Atualmente pode parecer difícil imaginar um computador que apenas permita utilizar uma aplicação por vez mas, de fato, houve um percurso histórico notável até se atingir o nível atual de paralelização de processos.

Em um sistema operacional multi-tarefa, um dos principais desafios para manutenção da performance é a política de distribuição do uso do processador entre os processos.

O XV6 é um sistema operacional UNIX didático desenvolvido pelo Instituto de Tecnologia de Massachusetts no verão de 2006. A ideia deste trabalho acadêmico é introduzir no XV6 um escalonador de processos baseado no algoritmo de escalonamento por loteria, proposto por Waldspurger em 1994. O escalonamento por loteria é um aloritmo probabilistico, sua ideia consiste em atribuir bilhetes aos processos, sortear um bilhete e dar a CPU ao processo portador do bilhete vencedor.

II. Definições Iniciais

A. Processo

Em sistemas operacionais um processo nada mais é do que uma tarefa a ser executada. Toda solicitação feita pelo usuário está associado é visto pelo sistema operacional como uma tarefa.

B. Sistemas operacionais multi-tarefa

Multitarefa é a característica do sistema operacional que permite repartir a utilização do processador entre várias tarefas aparentemente simultâneas. Pode parecer difícil de imaginar um computador monotarefa, que apenas permita utilizar uma aplicação mas, de fato, houve um percurso histórico notável até se atingir o nível atual de paralelização de processos. Um dos principais fatores a serem considerados para garantir a eficiência de um sistema operacional multi-tarefa é a forma como o mesmo distribui o uso de suas CPUs entre os processos que precisa executar.

C. Escalonamento de Processos

O Escalonador (Scheduler) é a parte do sistema operacional que determina qual processo deve ser executado em determinado momento. Pode parcer algo bastante simples, porém esta tarefa que precisa ser executada de maneira extremamente rápida pois interfere na performance do sistema operacional como um todo.

D. Problema da Inanição

Em sistemas multi-tarefa ocorre inanição quando um processo nunca é executado, ou seja, nunca recebe a CPU. Pode ocorrer em algoritmos de escalonamento por prioridade, quando processos de prioridade alta estão sempre a frente, deixando os de menor prioridade para sempre aguardando.

E. Escalonamento por Loteria

O escalonamento por loteria é um algoritmo probabilístico proposto por Waldspurger em 1994. A ideia é bastante simples, cada processo existente recebe um número x de bilhetes. Ao final de cada quantum é sorteado um número e o processo portador do bilhete com este número será o próximo a ter posse da CPU.

III. O escalonador do XV6

Originalmente, o algoritmo de escalonamento do XV6 é extremamente simples. Ao final de cada quantum é chamada a função scheduler(), que percorre a lista de processos existentes concedendo a CPU ao primeiro que estiver pronto para execução. A implementação da mesma é ilustrada na figura 1.

```
for(;;){
// Enable interrupts on this processor.
sti();

// Loop over process table looking for process to run.
acquire(&ptable.lock);
for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
    if(p->state != RUNNABLE)
    continue;
    // Switch to chosen process.  It is the process's job
    // to release ptable.lock and then reacquire it
    // before jumping back to us.
    proc = p;
    switchuvm(p);
    p->state = RUNNING;
    swtch(&cpu->scheduler, p->context);
    switchkvm();

    // Process is done running for now.
    // It should have changed its p->state before coming back.
    proc = 0;
}
release(&ptable.lock);
}
```

Fig. 1. Função scheduler() do XV6.

A função é um loop infinito sem nenhum retorno, quando um processo pronto é encontrado, a execução altera-se para o contexto do mesmo e ao final do quantum a ele concedido, de volta para o anterior e o loop continua a ser executado.

IV. Escalonamento por Loteria no XV6

Para implantação do método proposto por Waldspurger no XV6 foram necessárias algumas mudanças, porém nada de muito complexo.

A. Struct proc

Foi necessário adicionar ao processo um atributo que indicasse a quantidade de bilhetes a ele atribuídos, como mostra a figura 2.

```
// Per-process state
struct proc {
  uint sz;                     // Size of process memory (bytes)
  pde_t* pgdir;                // Page table
  char *kstack;                // Bottom of kernel stack for this
  enum procstate state;        // Process state
  int pid;                     // Process ID
  struct proc *parent;         // Parent process
  struct trapframe *tf;        // Trap frame for current syscall
  struct context *context;     // swtch() here to run process
  void *chan;                  // If non-zero, sleeping on chan
  int killed;                  // If non-zero, have been killed
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
  int tickets;                 // Procces tickets to sheduler
};
```

Fig. 2. Struct proc, atributos do processo.

Para atribuir um valor a este atributo foi adicionado à função fork() um parâmetro inteiro, bem como criada uma chamada de sistema com a assinatura cht(int pid, int tickets), através da qual pode-se atribuir o alterar o número de bilhetes atribuídos a um processo.

B. Sorteio

Para sortear o bilhete vencedor é utilizada a função random_number, que se baseia no algoritmo de Park-Muller para geração de números pseudo-aleatórios através do overflow de uma estrutura de dados numéricos. A implementação de random_number é ilustrada pela figura 3

```
int
random_number(int seed)
{
    return (unsigned long)(seed * 48271l10398420394UL) % 2147483647UL;
}
```

Fig. 3. Sorteio do bilhete vencedor

C. A função Scheduler

Na função scheduler foi necessária a criação de duas novas variáveis, sendo elas: winner o número do bilhete vencedor, obtido através do método random_number e sum, utilizada na busca pelo processo a ser executado.

```
for(;;){
  sti(); // Enable interrupts on this processor.
  acquire(&ptable.lock);
  sum = 0;
  winner = random_number(ticks*87329823458)%(get_tickets_number()+1);

  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
    //skip if procces is not runnable
    if(p->state != RUNNABLE)
      continue;

    sum += p->tickets;
    if(sum < winner)
      continue;

    proc = p;
    switchuvm(p);
    p->state = RUNNING;
    swtch(&cpu->scheduler, p->context);
    switchkvm();

    // Process is done running for now.
    // It should have changed its p->state before coming back.
    proc = 0;
  }
  release(&ptable.lock);
}
```

Fig. 4. Função scheduler adaptada para o método de loteria.

A figura 4 mostra a implementação da nova função scheduler().

A variável winner é definida pelo resto da divisão de um número pseudo-aleatório pelo número total de bilhetes no sorteio, o que garante que seja sorteado um bilhete válido.

A variável sum é iniciada com 0, e no início de cada repetição do laço é adicionado a ela o número de bilhetes atribuídos ao processo atualmente apontado pelo iterador p. Se o número de bilhetes atribuídos a p for maior do que sum, então p é o vencedor e recebe a cpu.

D. Chamada de sistema ps

Para auxiliar na visualização do método foi adicionada uma chamada de sistema ps, que chama a função cps(), implementada no arquivo proc.c, a qual imprime nome, pid, número de tickets e status de todos os processos com status RUNNABLE, RUNNING ou SLEEPING.

V. Conclusão

O algoritmo de escalonamento por loteria tem como principal vantagem a resolução do problema da inanição, já que um processo, mesmo que com um número pequeno de bilhetes, terá chances de ser sorteado. É ainda um algoritmo relativamente rápido, por sua simplicidade, embora a performance pudesse ser melhorada com a implementaçao de uma lista apenas com os processos prontos para execução, reduzindo o tempo de busca pelo processo portador do bilhete vencedor.

Link para o projeto: https://www.dropbox.com/s/qjzipmhogrua1jm/xv6 public.tar.gz?dl=0