Laboratório de Sistemas Operacionais (CC)

Trabalho Prático 2

Escalonadores de Disco

1. Objetivos

1. Expandir o conhecimento sobre os mecanismos internos do kernel Linux.
2. Implementar um módulo de kernel para a infraestrutura de IO Scheduler.
3. Estudar algoritmos de escalonamento de disco.
4. Avaliação de desempenho da solução adotada.

2. Descrição

Este trabalho consiste na criação de um módulo escalonador de disco que implemente a política SSTF (*Shortest Seek Time First*). O módulo deve funcionar com o Linux kernel *4.13.9* utilizado nas aulas.

A execução do trabalho consiste em 5 etapas:

1. Compreensão das técnicas de escalonamento de disco, incluindo as implementações atuais do kernel Linux.
2. Implementação do "*Tutorial 2.4: Noções sobre IO-Scheduler e acesso a disco*" para adquirir noções básicas do subsistema de I/O de disco do Linux.
3. Implementação do módulo de kernel conforme a política SSTF.
4. Implementação de aplicação para testes.
5. Avaliação de desempenho da implementação.

Na etapa 1, deve-se usar material fornecido pelo professor ou encontrado na Internet sobre o assunto. Para a etapa 2, um tutorial estará disponível no Moodle. Na etapa 3 e 4, o professor deixará um esqueleto de código, conforme instruções da seção 2.1. A etapa 5 será explicada na seção 2.2.

2.1 Implementação do módulo

Para ajudar na implementação do módulo, está disponível um esqueleto de código em *https://github.com/miguelxvr/sstf-iosched-skeleton.git*. Ainda, pode-se consultar como apoio as outras implementações de escalonadores de disco do Linux. Por exemplo, os escalonadores *noop* e *CFQ* estão disponíveis, respectivamente, em:

linux-4.13.9/block/noop-iosched.c
linux-4.13.9/block/cfq-iosched.c

Junto com o módulo, existe uma aplicação para acesso direto (*raw*) ao disco. Tal aplicação deverá ser modificada para gerar múltiplas requisições concorrentes para o disco, com o objetivo de simular um ambiente multiprogramado real. A versão disponível gera apenas uma requisição. **Dica:** use *fork()* para criar um grande número de processos que geram requisições de leitura em regiões aleatórias do disco.

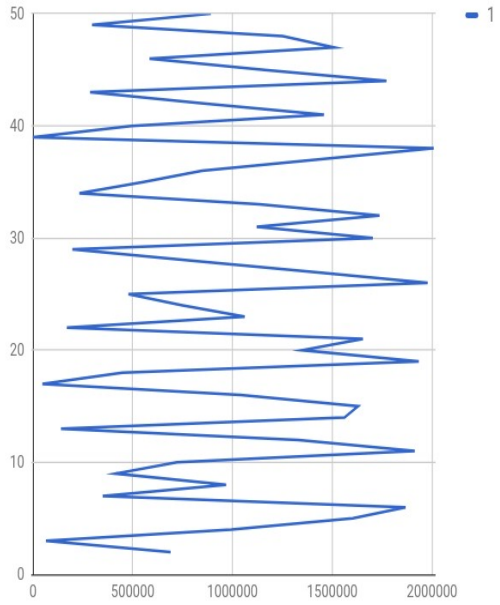
2.2 Avaliação de desempenho

Como estamos usando um simulador "*instruction accurate*", isto é, que simula o sistema a nível de instruções, não é possível tirarmos medidas efetivas de tempo para comparação entre políticas de escalonamento. Outro agravante, é que as requisições de acesso a disco do Linux emulado estão sujeitas às políticas de escalonamento do SO host. Contudo, podemos ter uma ótima ideia da efetividade da nossa implementação quando compararmos a ordem das requisições de acesso a disco recebidas pelo módulo e a ordem em que são atendidas. Por exemplo, considere um conjunto de 50 requisições de acesso a disco geradas na ordem abaixo.

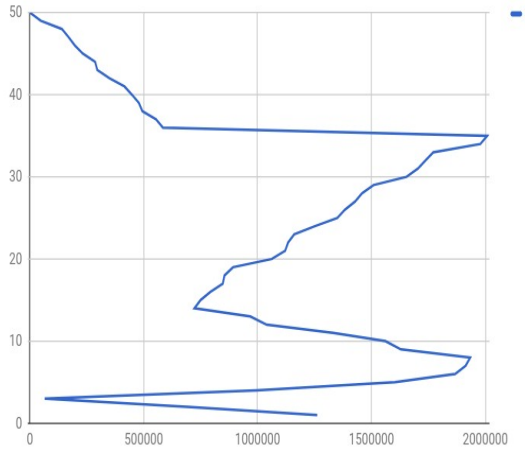
Ordem de Chegada	Sector
1	1261960
2	689984
3	64968
4	994448
5	1603288
6	1867480
7	349528
8	968120
9	415624
10	722632
11	1913536
12	1331304
13	140544
14	1561288
15	1629256
16	1040136
17	47272

18	448320
19	1933696
20	1350016
21	1653904
22	169880
23	1061512
24	749616
25	478432
26	1978192
27	1383664
28	792632
29	197520
30	1703504
31	1121120
32	1736912
33	1133936
34	232664
35	553800
36	847248
37	1428072
38	2008512
39	0
40	494056
41	1458928
42	854608
43	286384
44	1771728
45	1160664
46	584152
47	1509152
48	1251800
49	296120
50	891896

Isso resulta no gráfico de acesso a disco abaixo, que percorre um total de 37.074.272 setores.



Se as requisições forem ordenadas segundo o algoritmo SSTF, teremos o gráfico abaixo. O algoritmo SSTF requer percorrer apenas 7.571.176 setores para efetuar a mesma tarefa.



Desta forma, a implementação do módulo deve exibir os setores na ordem de chegada e depois na ordem de atendimento. Por isso, é importante gerar um grande número de requisições de disco com a aplicação de teste.

Obs: O comando **dmesg** mostra as mensagens do printk(). Pode ser necessário para obter os logs do escalonador e tabelar os resultados.

Entrega

Os alunos podem realizar esta atividade em duplas ou individualmente. Como entrega, é solicitado:

- Código fonte;
- Relatório com gráficos de desempenho.

Além disso, todo o desenvolvimento deve ser adicionado ao repositório de fontes e passar a fazer parte da distribuição. Na data da entrega, cada equipe deve apresentar o trabalho ao professor que fará questionamentos (presença obrigatória).