Instituto Multidisciplinar
Ciência da Computação
Sistemas Operacionais – 02/2018
Juliana M. N. S. Zamith

Trabalho SO – Opção 1 – Implementação de um alocador de memória

Título: MyMalloc - Alocador eficiente de memória

Objetivo: Implementar um alocador de memória para processos de usuários que apresente melhor desempenho que o malloc.

Contexto: A alocação de memória possui duas funcionalidades básicas. A primeira delas é solicitar ao SO um espaço de memória correspondente ao *heap*. A segunda refere-se ao gerenciamento do espaço de memória livre, o que implica em encontrar um espaço contíguo de memória livre que atenda à requisição do usuário.

O alocador de memória mais conhecido (o malloc) faz parte de uma biblioteca do padrão C e não é uma função do kernel do SO. Isto significa que o malloc não conhece nada sobre as páginas alocadas aos outros processos e como estas páginas estão alocadas na memória física. Sendo assim, a forma como o malloc solicita e gerencia a memória pode ter impactos negativos significativos no tempo de execução da aplicação devido ao tempo gasto na alocação e a forma como estes espaços são utilizados. Este impacto pode ser pior em aplicação multithreadeds [1].

O objetivo do trabalho é implementar o MyMalloc. Você deverá implementar as funções dentro de uma biblioteca compartilhada (libmem.so). Para criá-la, você deve fazer o seguinte:

```
gcc -c -fpic mymalloc.c (mymalloc possui ocódigo da sua biblioteca)
gcc -shared -o libmem.so mymalloc.o
```

Para "linkar" esta biblioteca você deve especificar o nome "-lmymalloc" e o caminho para encontrar a biblioteca usando -L, veja:

```
gcc mymain.c -lmymalloc -L. -o myprogram (mymain um programa qualquer que usa sua
biblioteca para fazer alocação dinâmica, myprogram é o nome do executável que será gerado)
```

Antes de começar a implementar suas ideias é interessante que você conheça o funciomamento da função malloc e como a mesma gerencia a memória (recomento leitura de [2,3,4]). Assim, você conseguirá apontar quais as melhorias que poderiam ser realizadas considerando o tempo de alocação, fragmentação, coalescência de memória, etc. No trabalho será avaliado a melhoria de pelo menos uma destas caractarísticas em relação a função malloc.

Para inspirar (leia, é importante!): *A alocação de memória em arquiteturas multicore é uma desafio já que os processos usam de forma compartilhada os barramentos o DMA. Alguns funções já implementadas são: PTMalloc, TCMalloc, etc. Você pode se inspirar nelas. Para isto, leia mais em [5, 6].*

Outros alunos já fizeram este trabalho, se quiserem podem comparar com de algum colega!

Funcionalidades:

A biblioteca MyMalloc deverá possuir **pelo menos** as seguintes funções:

- **int MyMallocInit(int size): e**sta função deverá ser chamada uma única vez e deverá requisitar ao SO uma espaço *size* de memória (para fazer alocação use a função mmap ou bsk [4]). Este será o espaço utilizado para alocar ao processo do usuário a memória requisitada quando a função mymalloc for chamada. Este espaço **poderia ser divido** em páginas para facilitar a alocação (leia mais em [4,5,6]). Não esqueça que este espaço poderia conter também a estrautura de dados que você utilizará para gerenciá-lo. Retorna 1 se houve sucesso na alocação.
- **void *MyMalloc(int size):** similar a função malloc. Dado o tamanho da memória solicitada pelo processo do usuário, a função deverá retornar um ponterio para um espaço contíguo de memória de tamanho *size*.
- **int MyMallocFree(void *ptr):** libera o espaço de memória alocado previamente (equivalente ao free()). Retorna se houve sucesso ou não.
- **MyMallocGerency:** este útlimo trata de um conjunto de funções que poderão ser utilizadas para imprimir estado da memória, ganho em relação à fragmentação, etc, ou seja, funções necesárias para avaliar o desempenho do MyMalloc.

Testes e avaliação de desempenho: para avaliar a função desenvolvida o grupo deverá executar pelo menos duas aplicações diferentes que usam a função MyMalloc. Sugiro que as aplicações sejam escolhidas de modo que a funcionalidade na qual se deseja obter um melhor desempenho seja evidenciada. Por exemplo, se o objetivo do seu MyMalloc é diminuir a fragmentação, aplicações que alocam e "desalocam" (usam free) a memória várias vezes durante a execução poderiam ser utilizadas como teste (aplicações que fazem muita atualização em uma lista encadeada, por exemplo). Se deseja avaliar a coalescência de memória, uma aplicação com várias *threads* poderia ser utilizada.

A escolha destas aplicações também fazem parte da avaliação. Então não deixem para fazer os testes de última hora. Sugiro que os membros do grupo se dividam para que não ocorra atrasos nos trabalho.

Para ajudar na leitura:

[1] http://www.oracle.com/technetwork/articles/servers-storage-dev/mem-alloc-1557798.html

[2] Costa, D E D; Um Estudo Experimental para Caracterização de Alocações Dinâmicas de Memória. Dissertação de mestrado. Universidade Federal de Uberlância, 2014. (tem um artigo com o mesmo nome fácil de ler!)

[3] http://danluu.com/malloc-tutorial/

[4] http://www.inf.udec.cl/~leo/Malloc_tutorial.pdf

[5] http://cs.nyu.edu/~lerner/spring12/Preso05-MemAlloc.pdf

[6] http://ppedreiras.av.it.pt/resources/str1112/apresentacoes_pesquisa/mnobrega-allocators.pdf

[7] https://www.ieee.org/publications_standards/publications/authors/author_templates.html

Trabalho SO – Opção 2 – Monitoramento

Título: Monitoramento do escalonador

Objetivo: Desenvolver um monitorador de escalonamento de processos. O monitorador deve ficar analisando por um tempo aplicações de diferentes objetivos e produzir um gráfico que apresente o tempo em que o processo foi lançado (iniciou a execução), o tempo que o processo ficou na fila de espera e o tempo de término.

Contexto: Monitorar o escalonamento feito pelo SO é interessante para avaiar a vasão de execução dos processos/threads de uma aplicação. Esta avaliação pode auxiliar no desenvolvimento de escalonadores mais elaborados que apresentem melhores resultados de desempenho considerando as arquiteturas atuais. Ou ainda, pode ajudar a propor escalonamentos para uma aplicação específica com o intuito de acelerar sua execução.

Para entender melhor o contexto leia:

https://www.princeton.edu/~wentzlaf/documents/Wentzlaff.2009.OSR.fos.pdf;

https://dl.acm.org/doi/10.1145/2901318.2901326;

(Alexandra Ferdorova tem vários artigos de 2010 que discute escalonamento e usa benchmarks para tal fim, vale a pena ler sobre)

https://dl.acm.org/doi/10.1145/2379776.2379780

Este é um trabalho de pesquisa e muitos testes, portanto, deve ser feito durante todo o semestre, sendo o grupo responsável por avaliar como deve ocorrer o monitoranmento, quais aplicações poderiam ser usadas de testes etc. Use benchmarks já apropriados para tal situação

Este tipo de monitoramento é comumente chamado de tracing e é realizado pelo Kernel. Existem alguns tracers no linux:

1) perf_sched: https://www.brendangregg.com/perf.html#SchedulerAnalysis e para coletar informações monitoradas, veja: http://www.brendangregg.com/perf.html

3) Outras opções: trace-cmd, instalar e usar kernelshark;

4) https://stackoverflow.com/ :) Lá tem várias discussões sobre como fazer este monitoramento, inclusive usando o /proc

Para as duas opções :

Grupos: O trabalho poderá ser desenvolvido por grupos de no máximo três pessoas.

O que deve ser entregue:

O trabalho deverá ser apresentado no dia 09 ou 14 de dezembro (agende sua apresentação). **Prepare uma pequena apresentação** (com slides apresentando como foi feito o trabalho, quais estratégias usaram, monte gráficos que mostrem, por exemplo, onde houve ganho de desempenho em relação ao malloc, pseudocódigos para te ajudar a apresentar a implementação do MyMalloc. No caso da segnda opção monte o gráfico de Gant – veja os gráficos que são usados para mostrar escalonamento)

* **Neste mesmo dia** você deverá entregar um relatório (use o modelo disponível no email) contendo detalhes a respeito da sua implementação e a avaliação proposta acima. O seu trabalho deve conter pelo menos as seguintes seções: Título, Introdução, Trabalhos Relacionados, Desenvolvimento, Resultados, Conclusão, Referências. Quem quiser pegar o template para latex é só baixar na página da IEEE [7].

*** MAIS IMPORTANTE:** Veja que nos dois casos o trabalho inclui estudo e muita pesquisa, portanto não deixe para fazer o trabalho de última hora - não dá tempo!!!!