Informe tarea 2
Sitema Operativo y Redes

Gianfranco Traverso

Katherine Jara

jueves 17 de octubre

1 Revisión bibliográfica

La **paginación** es un esquema de administración de memoria que permite que el espacio de direcciones físicas de un proceso no sea contiguo, previniendo asi la fragmentación externa y la necesidad de compactación, por ende se utiliza paginación en la mayoría de los SS.OO. Modernos. El método básico para implementar paginación requiere romper la memoria física en bloques de tamaño fijo llamados marcos. La memoria lógica también se rompe en bloques de tamaño fijo, del mismo tamaño, llamados páginas. Cuando se ejecuta un proceso, se cargan sus páginas en los marcos de memoria disponibles.

La **memoria virtual** abstrae al programador de la cantidad de memoria física disponible. El proceso simplemente tiene acceso a un espacio nominal de memoria virtual, que puede ser más grande que la memoria física. Con memoria virtual podemos lograr que varios procesos puedan compartir páginas y así permitir que existan bibliotecas compartidas cargadas en memoria y utilizadas simultáneamente por varios procesos entre otras cosas.

Con memoria virtual y un esquema de paginación floja, puede ocurrir que un proceso intente acceder a una página que no está cargada en memoria y se conoce como **falta de página**. Esto puede ocurrir debido a que los esquemas de paginación usan bits de validez-invalidez para indicar en tabla de páginas las páginas que se encuentran disponibles en memoria física. Sin embargo puede ocurrir que la CPU genera una dirección lógica para referenciar un dato que está en una página inválida.

2 Uso del programa

Para ejecutar el programa, es necesario escribir los siguientes commandos en la terminal:

```
$ make
$ ./virtmem <pages> <frames> <algoritmo de reemplazo> <patron de acceso>
```

En **pages** y **frames** se tiene que especificar la cantidad de paginas y marcos que se desean usar respectivamente, mientras que en **algoritmo de reemplazo** se especifica el tipo de algoritmo que se desea usar para el remplazo de páginas a utilizar. En este caso se tienen solo dos algoritmos a elección, random y FIFO. Por último tenemos el **patron de acceso**, en donde se elige uno de los tres patrónes de acceso a memoria.

Patrones de accesos :
Existen tres tipos distintos de patrones a utilizar en este programa. El primero es el **a1**, accede a la memoria de manera aleatoria. El segundo algoritmo **a2**, utiliza la funcion qsort() de la libreria estandar de C, mientras que el último **a3**, es el algoritmo secuencial presentado en la guia de trabajo.

Ejemplo de uso:
Si se quiere ejecutar el programa con 100 páginas de memoria virtual y 50 marcos de memoria física usando reemplazo aleatorio de páginas y el patron de acceso a2 (qsort()), se usan los siguientes comando:

```
$ make
$ ./virtmem 100 50 rand a2
```

En caso que se quiera el mismo ejemplo anterior, pero esta vez usando el algoritmo FIFO, se escribe lo siguiente:

```
$ make
$ ./virtmem 100 50 fifo a2
```

.

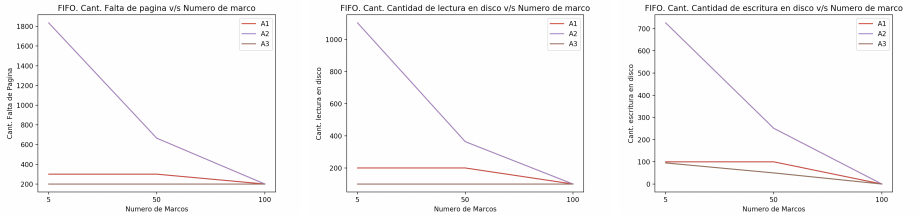

Figure 1: FIFO

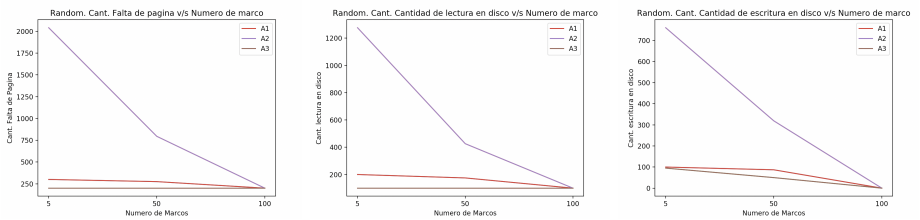

Figure 2: Random

3 Análisis de resultados

Cómo se puede observar en los gráficos, los distintos patrones de acceso de memoria tienen un comportamiento similar independiente del tipo de algoritmo que se use. Sin Embargo, en los gráficos sobre falta de páginas y lectura de disco, existe una pequeña diferencia entre ambos algoritmos, en donde FIFO llega a ser un poco mas eficiente debido a que hay menos lecturas en disco y una menor cantidad de páginas faltantes. No obstante estas diferencias van disminuyendo a medida que la cantidad de marcos aumenta, esto es debido a que la cantidad de páginas se acerca a la cantidad de marcos que se van usando.

Los patrones de acceso de memoria tienen un desempeño distinto entre si, donde el mas eficiente es el a3 (Secuencial), que se menciona en la guía, mientras que el mas lento es el a2 (Sort), teniendo una diferencia de mas de mil lecturas de disco y páginas faltantes con respecto a a3.

4 Referencias

`https://github.com/iheanyi/school_code/tree/master/cse30341/project5/source_pr5`