Page Replacement

Pedro H. Penna, Henrique Freitas,
Márcio Castro and Jean-François Méhaut

Pontifical Catholic University of Minas Gerais
Federal University of Santa Catarina
University of Grenoble Aples

Abstract

The memory of a computer is hierarchically organized, with the operating system acting as a manager. In this approach, a core functionality is to maintain in memory the most used pages, and offload to the disk those that are not used. In nanvix, this job is accomplished by the page replacement component, and in this assignment you should propose enhancements in it.

Background

To enable the illusion of a computer with a infinitely large, fast and non-volatile memory, the memory of computer is hierarchically organized. In this approach, fast and small memories are placed near the processor, whereas large and slow memories are farther.

In the memory hierarchy, the job of the operating system is to act of a manager, in a efficient and transparent way. To do so, modern operating systems rely on a hybrid hardware/software technique called virtual memory.

In this technique, the address space of a process is divided in equal-size regions, named pages. In turn, pages are mapped to regions of the same size in the physical memory, which are referred as page frames. This way, a set of process can share the physical memory, even though all of them may not fit in at the same time.

To maintain such illusion based on pages and page frames, all that the operating system does is to maintain in memory those pages that are more frequently used, and offload to disk those that are no logger. This bookkeeping has a great influence in the overall performance of the system, and it is carried out by the page replacement component, a module of the memory management subsystem.

Assignment Description

The page replacement component in Nanvix relies on a first-in-first-out local page replacement policy. That is, whenever a page fault occurs, the operating system chooses, among the in-core pages of the faulting process, the oldest page to be evicted. This policy has is easy to implement and yields to minimum overhead. However, it leads to a poor performance, specially those process that present a well-behaved data access pattern – which is usually true.

For instance, suppose program that performs a matrix multiplication of two matrices that do not fit in memory. In this situation, the wisest choice is not to keep in memory the most recently used pages, but to maintain in-core those that represent the working set of the program, which are the pages that store current lines and columns that are being multiplied.

To address this problem, you have to hack the page replacement algorithm in Nanvix and implement any other policy that yields to a better performance. You have to present a quantitative performance analysis of your solution. For doing so, you may use the matrix multiplication program that we have provided along with the system. Keep in mind that robust algorithms like the Working Set and Aging may yield to good results.

Start Point

The implementation of the memory management subsystem is in the `kernel/mm` directory, and it is split in several files:

- `mm.c`: inicialização do gerenciador de memória.
- `paging.c`: sistema de paginação e *swapping*.
- `region.c`: gerenciamento de regiões.

In this assignment, you should focus on the `paging.c` source file. More precisely, in the `allocf()` kernel function, which implements the paging replacement algorithm. Besides this, the source file for the matrix multiplication algorithm is located in `src/sbin/test/test.c`