CS 377

OS Design Project Theme A- Virtual Memory for Pranali

Team A3

Progress report submitted March 28th 2014

Table of Contents

1 Swap space management

1.1 Functions Analysed:

1.1.1 ld_load_prog_from_ctxconfig:

-This function reads the configfile and sets up the context for each process to be executed -It then calls the ld_load_exe function to load the program into memory

1.1.2 ld_load_exe

- This function calls elf_open which reads the executable to find out details of its sections and program headers
- then ld_load_section, ld_load_stack and others are called which will actually create pages for these things and map them to the main memory(of both pranali and current process)

1.1.3 ld_load_section/ld_load_stack:

- this function will map this memory to physical memory using memory_map function

1.1.4 memory_map:

- this function will map memory on the heap(of pranali) to the pages using mem_page_create function if it is not already assosciated to some page(checked using mem_get_page function)

1.1.5 mem_get_page:

- this function will take in a physical memory location and return the corresponding physical page(physical in perspective of this process on pranali)

1.1.6 mem_page_create:

- this function will create a new page in the heap allocated to pranali

1.2 Modifications needed:

1.2.1 mem_page_create:

- new page should now instead be created in the swap space of the process, which will be on the disk, that is simdisk in this case

1.2.2 mem_page_get:

- this should return the page in main memory if available.
- and invoke the fetching of the page to main memory otherwise.

1.3 new Functions needed:

1.3.1 disk_page_create:

- this function should create a new page on the simdisk in the swap space of the process

1.3.2 page_get:

- this function should fetch the page from swap space (simdisk) to main memory
- if found in main memory(main for both the process and pranali), we return it
- else it is a page fault and will be handled as:
- we will maintain a count for how many pages of a particular process are in memory
- and we will allocate new space for the page being fetched on the heap if count < max_allowed_count_for_a_process
- otherwise, we invoke the page replacement function which will free space of a page as per page replacement policy and allocate space for this page being fetched

2 Page Fault Handling

2.1 Functions Analysed:

2.1.1 mem_page_get

returns mem page corresponding to an address If the page is present, it returns it, otherwise returns NULL places the found page at the list head

2.1.2 mem_page_get_next

Returns the memory page following addr in the current memory map. This function is useful to reconstruct consecutive ranges of mapped pages. If the page following addr is not found, checks all

memory pages to find the one with the lowest tag following addr.

2.1.3 mem_page_create

Creates new mem page

2.1.4 mem_page_free

Frees mem pages

2.1.5 mmu_init

- checks if the page size is a power of 2
- allocates memory for mmu and its page_list

2.1.6 mmu_done

frees mmu and its page_list

2.1.7 mmu_get_page

calculates hash, tag etc. of the page and looks for the page in the hash. If the page is not found, then a new page is created, and in any case, the page is put to the head of the list having the same hash.

2.1.8 mmu_translate

translate virtual address to physical address and returns it

2.2 Modifications needed:

- The functions used by the memory management unit are not used at all. So basically, the OS does not use the virtual memory implementations at all. So, they need to be called at appropriate places.

2.2.1 mmu_get_page

- When a page is not found in the memory, it should be fetched from the swap space instead of creating a new page.
- When it is found that a page has to be replaced (due to space constraints), it should call the page replacement algorithm to choose the page that should be removed from memory.

2.3 New Functions needed

2.3.1 getPageToBeRemove:

It calls the appropriate page replacement policy and returns that page that should be removed from memory.

2.3.2 PageIn:

It will load the page from the swap space into memory

2.3.3 PageOut:

It will save the page on the swap space, if changes have been done.

Date: 2014-03-28T18:51+0530

Author: Sagar Jha

Org version 7.9.3f with Emacs version 24

Validate XHTML 1.0