OS Project 3 Team 02 Report

B04902027 陳昇 B04902023 鄭士驤

Part I Code Reading

備註：

在我們讀code的時候，會發現一些語句特別常出現，例如：

一、if(likely()), if(unlikely())：

經查詢資料了解，在邏輯上if(likely(val))及if(unlikely(val))的意思皆與if(val)相同，但是likely(val)是指val=1的情形比較常出現、unlikely(val)是指val=0較常出現，它可以提升CPU中branch predictor預測正確率而提升效能，但是我們就以if(val)來理解。

二、EXPORT_SYMBOL_GPL

壹、

How filemap_fault() is set as the page fault handler when mmap is called?

Handle_mm_fault -> __Handle_mm_fault -> Handle_pte_fault -> do_read_fault, do_cow_fault, do_shared_fault -> __do_fault (ret =vma->vm_ops->fault(vmf);)

```
do_read_fault(vmf);
        else if (!(vma->vm_flags & VM_SHARED))
                ret = do_cow_fault(vmf);
        else
                ret = do_shared_fault(vmf);
```

貳、How and when the readahead algorithm takes place when filemap_fault() is invoked?

(1) __do_page_cache_readahead()

Before doing the readahead , it will allocate pages first. When doing the readahead , it will check the cache to see if the desired page is already in it. If the page can't be found in the cache , the page is allocated and added into the page pool. When the index of a page is nr_to_read - lookahead_size , put on the PG_readahead mark in order to set the asynchronous readahead next time.

(2) ondemand_readahead()

If the page is read from the beginning , readahead algorithm will be initialized. If not , but it is a sequential read , it will expand the readahead amount. If it is a random read , readahead algorithm will not be used.

(3) page_cache_async_readahead()

It is implemented on the pages with PG_readahead mark. If the page does not need to readahead , or is in the state of writing back , then return. If not , clear the mark.

(4) do_async_mmap_readahead()

When we find the page with PG_readahead mark , asynchronous readahead will be implemented. So that we have to consider the situation of page to implement respectively.

(5) filemap_fault()

When a page fault is happened , filemap_fault() is called. After it make sure that whether the page is in the cache or not , it will check if the up-to-date is added to the page. If not , it will be added and re-check again ; if yes , then return.

Part II

壹、實作細節：

一、執行環境為 Virtualbox, **基礎記憶體**1024MB, Hard Disk

二、修改 include/linux/mm.h**中的**VM_MAX_READAHEAD**從預設的**128**改成**1024

三、編譯內核並執行測試程式

貳、實驗結果：

VM_MAX_READAHEAD/ VM_MIN_READAHEAD	# of major pagefault	# of minor pagefault	# of resident set size(KB)	execution time(real)
128/16 (default)	4203	2551	26680	0m39.606s
4/1	5552	1199	26680	1m0.781s

1024/16	188	6563	26680	0m4.159s

參、討論：

一、為什麼修改VM_MAX_READAHEAD可以改到readahead演算法

1.修改VM_MAX_READAHEAD會修改struct file_ra_state的ra_pages

2. include/linux/mm/readahead.c中的ondemand_readahead會用到unsigned long max = max_sane_readahead(ra->ra_pages);

3. ondemand_readahead被include/linux/mm/readahead.c中的page_cache_sync_readahead及page_cache_async_readahead呼叫

4.page_cache_sync_readahead及page_cache_async_readahead被mm/filemap.c中的do_async_mmap_readahead及do_sync_mmap_readahead呼叫

5.do_async_mmap_readahead及do_sync_mmap_readahead被mm/filemap.c中的filemap_fault()呼叫

二、實驗結果討論

將VM_MAXREADAHEAD調大可以增加預先讀進來的資料的大小，進而減少major pagefault，(自然地，minor pagefault增加)。由於Hard Disk是非常慢的裝置，這樣做可以減少Hard Disk讀取次數，縮短執行時間。