

솔루션 개발지원 보고서
일사천리 커널 에러 작업
((주)카이시스)

한국소프트웨어진흥원
공개SW기술지원센터

<Revision 정보>

일자	VERSION	변경내역	작성자
2007. 02. 22	0.1	초기 작성	박준완

목 차

1. 문서 개요	4
가. 문서의 목적	4
나. 본 문서의 사용방법	4
다. 참고 사항	4
2. 개발지원 요청내역	5
가. 대상 기관	5
나. 요청 사항	5
다. 지원인력구성	5
3. 개발지원 내용	6
가. 에러 사항	6
나. 에러 발생 이유	6
다. 에러 해결 방안	6
4. 참조자료	9

1. 문서 개요

본 문서는 KIPA 공개S/W 기술지원 센터에서의 리눅스 데스크탑 솔루션 개발지원 결과를 보고하기 위해 제작되었다.

가. 문서의 목적

다음과 같은 세부적인 목적을 달성하기 위하여 작성되었다.

0 개발 지원 요청 사항

0 개발 지원 결과 내역

나. 본 문서의 사용방법

다음과 같은 방법으로 사용할 수 있다.

0 __find_get_block_slow() failed 에러에 대한 대처 방안

0 개발 지원 결과 내역을 참조한다.

다. 참고사항

0 개발지원시 시스템 관련사항, 기술적 배경 등 참고사항들을 기술한다.

2. 개발지원 요청내역

가. 대상 기관

구 분		비 고
업체명	(주) 카이시스	
담당자	안효정	
연락처	031-786-1888(내선222)	
E메일	pearl@kaisys.net	
회사 위치	경기도 성남시 분당구 수내1동	

나. 요청 사항

항목	요청내용	지원 내역	비 고
	한소프트리눅스 Open Edition 3.0 beta3 버전에 일사처리 솔루션 포팅 작업을 완료한 후, 클라이언트 PC를 네트워크 부팅 시키는 과정에서 일사처리 서버에서 부팅 커널 이미지를 로딩하여 진행되는 중에 [__find_get_block_slow() failed] 같은 에러메시지가 뜨면서 부팅이 실패되었습니다. 해당 오류에 대한 일반적인 발생원인과 해결 방안에 대해 요청합니다.	에러 메시지 발생 원인에 대해 파악하고 에러 해결 방안을 모색하여 패치 작업 및 솔루션 개발에 필요한 여러 가지 사항 제공	

다. 지원인력구성

담당	직급	성명	소속사	인력구분	기간	지원내용	비 고
솔루션 개발팀	과장	박준완	리눅스원(주)	상주	2007.02.12~ 2007.02.23	발생 에러에 대한 문제해결 방안 및 원인 분석	

3. 개발지원 내용

가. 에러사항

일사처리에서 발생한 [__find_get_block_slow() failed] 에러 메시지 내용

```
Suspended screenshot for D:\WMy Document\WMy Virtual Machines\Wilsa_client\Wother26xlinu...
__find_get_block_slow() failed. block=18446744071562067968, b_blocknr=2147483648
b_state=0x00000020, b_size=4096
device blocksize: 4096
__find_get_block_slow() failed. block=18446744071562067968, b_blocknr=2147483648
b_state=0x00000020, b_size=4096
device blocksize: 4096
__find_get_block_slow() failed. block=18446744071562067968, b_blocknr=2147483648
b_state=0x00000020, b_size=4096
device blocksize: 4096
__find_get_block_slow() failed. block=18446744071562067968, b_blocknr=2147483648
b_state=0x00000020, b_size=4096
device blocksize: 4096
__find_get_block_slow() failed. block=18446744071562067968, b_blocknr=2147483648
b_state=0x00000020, b_size=4096
device blocksize: 4096
__find_get_block_slow() failed. block=18446744071562067968, b_blocknr=2147483648
b_state=0x00000020, b_size=4096
device blocksize: 4096
__find_get_block_slow() failed. block=18446744071562067968, b_blocknr=2147483648
b_state=0x00000020, b_size=4096
device blocksize: 4096
__find_get_block_slow() failed. block=18446744071562067968, b_blocknr=2147483648
b_state=0x00000020, b_size=4096
device blocksize: 4096
```

나. 에러 발생 이유

- Denial of Services 상태로 전이될 수 있는 오염된 데이터 구조체에 대해 제어하지 못했을 경우 이 특별한 취약점은 race condition(경쟁상태)와 signedness 이슈에 기인하여 발생한다.
- 커널에서 기본 설정되어 있는 파티션당 마운트 최대 개수 및 파일시스템 체크 주기에 관련 하여 발생한다.
- 커널 2.6 버전이상에서 발생하고 있는 에러 메시지 이다.
- 현재 이 버그에 대한 패치 파일이 웹 사이트에 나와 있다.

다. 에러 해결 방안

- 간단한 해결방안(임시적 해결 방안)
tune2fs -c 0 -i 0 \$part
이 명령은 \$part 부분에 대해서 마운트 최대 개수를 무시하고, 파일 시스템의 체크 주기를 없애 파티션 마운트시 커널에서 발생하는 block_slow()에 대한 에러를 억제 할 수 있다.

- 커널 컴파일을 통한 해결 방안 (kernel ver. 2.6.18 기준)

- ㄱ. Kernel Source를 다운로드 한다.
- ㄴ. (kernel ver.)/fs/buffer.c 파일을 수정한다.

※ 아래 내용은 원본 파일과 수정한 파일을 diff 명령으로 비교해 놓은 내용이다.

===== buffer.c 파일 diff 내용 =====

```
diff --git a/fs/buffer.c b/fs/buffer.c
```

```
index 23f1f3a..71649ef 100644
```

```
--- a/fs/buffer.c
```

```
+++ b/fs/buffer.c
```

```
@@ -18,7 +18,6 @@
```

```
 * async buffer flushing, 1999 Andrea Arcangeli <andrea@suse.de>
 */
```

```
-#include <linux/config.h>
```

```
#include <linux/kernel.h>
```

```
#include <linux/syscalls.h>
```

```
#include <linux/fs.h>
```

```
@@ -331,7 +330,6 @@ long do_fsync(struct file *file, int dat
```

```
    goto out;
```

```
}
```

```
- current->flags |= PF_SYNCWRITE;
```

```
ret = filemap_fdatawrite(mapping);
```

```
/*
```

```
@@ -346,7 +344,6 @@ long do_fsync(struct file *file, int dat
```

```
err = filemap_fdatawait(mapping);
```

```
if (!ret)
```

```
    ret = err;
```

```
- current->flags &= ~PF_SYNCWRITE;
```

```
out:
```

```
    return ret;
```

```
}
```

```
@@ -473,13 +470,18 @@ out:
```

```
    pass does the actual I/O. */
```

```
void invalidate_bdev(struct block_device *bdev, int destroy_dirty_buffers)
```

```
{
```

```
+ struct address_space *mapping = bdev->bd_inode->i_mapping;
```

```
+
```

```
+ if (mapping->npages == 0)
```

```
    return;
```

```
+
```

```
invalidate_bh_lrus();
```

```
/*
```

```
 * FIXME: what about destroy_dirty_buffers?
```

```
 * We really want to use invalidate_inode_pages2() for
```

```
 * that, but not until that's cleaned up.
```

```
*/
```

```
- invalidate_inode_pages(bdev->bd_inode->i_mapping);
```

```
+ invalidate_inode_pages(mapping);
```

```
}
```

```
/*
```

```
@@ -566,7 +568,7 @@ still_busy:
```

```
 * Completion handler for block_write_full_page() - pages which are unlocked
```

```
 * during I/O, and which have PageWriteback cleared upon I/O completion.
```

```
*/
```

```
-void end_buffer_async_write(struct buffer_head *bh, int uptodate)
```

```
+static void end_buffer_async_write(struct buffer_head *bh, int uptodate)
```

```
{
```

```
    char b[BDEVNAME_SIZE];
```

```
    unsigned long flags;
```

```
@@ -854,7 +856,7 @@ int __set_page_dirty_buffers(struct page
```

```
    write_lock_irq(&mapping->tree_lock);
```

```
    if (page->mapping) { /* Race with truncate? */
```

```
        if (mapping_cap_account_dirty(mapping))
```

```
-        inc_page_state(nr_dirty);
```

```
+        __inc_zone_page_state(page, NR_FILE_DIRTY);
```

```
        radix_tree_tag_set(&mapping->page_tree,
```

```
            page_index(page),
```

```
            PAGECACHE_TAG_DIRTY);
```

```
@@ -2600,7 +2602,7 @@ int nobh_truncate_page(struct address_sp
```

```
    unsigned offset = from & (PAGE_CACHE_SIZE-1);
```

```
    unsigned to;
```

```
    struct page *page;
```

```
- struct address_space_operations *a_ops = mapping->a_ops;
```

```
+ const struct address_space_operations *a_ops = mapping->a_ops;
```

```
char *kaddr;
int ret = 0;

@@ -3168,7 +3170,6 @@ EXPORT_SYMBOL(block_sync_page);
EXPORT_SYMBOL(block_truncate_page);
EXPORT_SYMBOL(block_write_full_page);
EXPORT_SYMBOL(cont_prepare_write);
-EXPORT_SYMBOL(end_buffer_async_write);
EXPORT_SYMBOL(end_buffer_read_sync);
EXPORT_SYMBOL(end_buffer_write_sync);
EXPORT_SYMBOL(file_fsync);

===== buffer.c 파일 diff 내용 =====
ㄷ. 수정된 내용을 토대로 커널 컴파일을 통해 커널에 적용한다.
ㄹ. 새로운 커널을 이용하여 시스템을 부팅한다.
```

4. 참조자료

- `__find_get_block_slow` 함수
kernel source내의 fs/buffer.c 파일
(line 386, 424, 1434, 1698 참조)
- www.kernel.org
- <http://www.redhat.com/archives/dm-devel/2005-February/msg00088.html>
- <http://www.gelato.unsw.edu.au/lxr/source/fs/buffer.c?v=2.6.16>
- <http://www.linuxhq.com/kernel/v2.6/18/fs/buffer.c>
(위 주소는 2.6.18 커널의 buffer 패치 파일을 받을 수 있다)