



```

kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/buffer.c
--- linux-2.6.18-FRESH-OPENVZ/fs/buffer.c 2006-11-26 04:07:28.000000000 -0500
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/buffer.c 2006-11-26 05:52:40.000000000
-0500
@@ -41,7 +41,7 @@
#include <linux/bitops.h>
#include <linux/mpage.h>
#include <linux/bit_spinlock.h>
-
+#include <ub/ub_misc.h>
static int fsync_buffers_list(spinlock_t *lock, struct list_head *list);
static void invalidate_bh_lrus(void);

@@ -858,8 +858,10 @@
if (!TestSetPageDirty(page)) {
write_lock_irq(&mapping->tree_lock);
if (page->mapping) { /* Race with truncate? */
- if (mapping_cap_account_dirty(mapping))
- __inc_zone_page_state(page, NR_FILE_DIRTY);
+ if (mapping_cap_account_dirty(mapping)) {
+ ub_byteswritten_charge(PAGE_CACHE_SIZE);
+ __inc_zone_page_state(page, NR_FILE_DIRTY);
+ }
radix_tree_tag_set(&mapping->page_tree,
page_index(page),
PAGECACHE_TAG_DIRTY);
@@ -2862,8 +2864,11 @@
void ll_rw_block(int rw, int nr, struct buffer_head *bhs[])
{
int i;
+ if (likely(nr) && !(rw & WRITE))
+ ub_bytesread_charge(nr * bhs[0]->b_size);

for (i = 0; i < nr; i++) {
+
struct buffer_head *bh = bhs[i];

if (rw == SWRITE)
@@ -2999,7 +3004,9 @@
* This only applies in the rare case where try_to_free_buffers
* succeeds but the page is not freed.
*/
- clear_page_dirty(page);
+ if (test_clear_page_dirty(page))
+ ub_byteswritten_uncharge(PAGE_CACHE_SIZE);
+
}
spin_unlock(&mapping->private_lock);

```

```

out:
diff -ur linux-2.6.18-FRESH-OPENVZ/fs/cifs/file.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/cifs/file.c
--- linux-2.6.18-FRESH-OPENVZ/fs/cifs/file.c 2006-09-19 23:42:06.000000000 -0400
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/cifs/file.c 2006-11-26
04:36:24.000000000 -0500
@@ -39,6 +39,7 @@
#include "cifs_unicode.h"
#include "cifs_debug.h"
#include "cifs_fs_sb.h"
+#include <ub/ub_misc.h>

```

```

static inline struct cifsFileInfo *cifs_init_private(
    struct cifsFileInfo *private_data, struct inode *inode,
@@ -1815,6 +1816,7 @@
    }
    break;
} else if (bytes_read > 0) {

```

```

+ ub_bytesread_charge(bytes_read);
  pSMBr = (struct smb_com_read_rsp *)smb_read_data;
  cifs_copy_cache_pages(mapping, page_list, bytes_read,
    smb_read_data + 4 /* RFC1001 hdr */ +

```

```

diff -ur linux-2.6.18-FRESH-OPENVZ/fs/direct-io.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/direct-io.c
--- linux-2.6.18-FRESH-OPENVZ/fs/direct-io.c 2006-09-19 23:42:06.000000000 -0400
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/direct-io.c 2006-11-26
04:36:01.000000000 -0500
@@ -35,6 +35,7 @@
#include <linux/rwsem.h>
#include <linux/uio.h>
#include <asm/atomic.h>
+#include <ub/ub_misc.h>

```

```

/*
 * How many user pages to map in one call to get_user_pages(). This determines
@@ -675,6 +676,11 @@
{
    int ret = 0;

```

```

+    if (dio->rw & WRITE) {
+ ub_byteswritten_charge(len);
+    }
+
+

```

```

/*
 * Can we just grow the current page's presence in the dio?
*/

```

```

diff -ur linux-2.6.18-FRESH-OPENVZ/include/ub/beancounter.h

```

```

kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/include/ub/ beancounter.h
--- linux-2.6.18-FRESH-OPENVZ/include/ub/beancounter.h 2006-11-26 04:07:28.000000000
-0500
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/include/ub/ beancounter.h 2006-11-26
06:57:14.000000000 -0500
@@ -71,13 +71,21 @@
#define UB_NUMOTHERSOCK 17 /* Number of other sockets. */
#define UB_DCACHESIZE 18 /* Size of busy dentry/inode cache. */
#define UB_NUMFILE 19 /* Number of open files. */
+#define UB_NUMREADS 20
+#define UB_NUMWRITES 21
+#define UB_KBYTESREAD 23
+#define UB_KBYTESWRITTEN 24
+
+
+#define UB_RESOURCES 25

-#define UB_RESOURCES 24

#define UB_UNUSEDPRIVVM (UB_RESOURCES + 0)
#define UB_TMPFSPAGES (UB_RESOURCES + 1)
#define UB_SWAPPAGES (UB_RESOURCES + 2)
#define UB_HELDPAGES (UB_RESOURCES + 3)
+#define UB_BYTESREAD (UB_RESOURCES + 4)
+#define UB_BYTESWRITTEN (UB_RESOURCES + 5)

struct ubparm {
/*
diff -ur linux-2.6.18-FRESH-OPENVZ/include/ub/ub_misc.h
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/include/ub/ ub_misc.h
--- linux-2.6.18-FRESH-OPENVZ/include/ub/ub_misc.h 2006-11-26 04:07:28.000000000 -0500
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/include/ub/ ub_misc.h 2006-11-26
06:25:02.000000000 -0500
@@ -18,6 +18,12 @@
struct file_lock;
struct sigqueue;

+UB_DECLARE_FUNC(int, ub_numreads_charge())
+UB_DECLARE_FUNC(int, ub_numwrites_charge())
+UB_DECLARE_FUNC(int, ub_numwrites_uncharge())
+UB_DECLARE_FUNC(int, ub_byteswritten_charge(int bytes))
+UB_DECLARE_FUNC(int, ub_byteswritten_uncharge(int bytes))
+UB_DECLARE_FUNC(int, ub_bytesread_charge(int bytes))
UB_DECLARE_FUNC(int, ub_file_charge(struct file *f))
UB_DECLARE_VOID_FUNC(ub_file_uncharge(struct file *f))
UB_DECLARE_FUNC(int, ub_flock_charge(struct file_lock *fl, int hard))
diff -ur linux-2.6.18-FRESH-OPENVZ/kernel/ub/beancounter.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/kernel/ub/b eancounter.c

```

```

--- linux-2.6.18-FRESH-OPENVZ/kernel/ub/beancounter.c 2006-11-26 04:07:28.000000000 -0500
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/kernel/ub/beancounter.c 2006-11-26
08:22:41.000000000 -0500
@@ -60,14 +60,18 @@
    "numothersock",
    "dcachesize",
    "numfile",
- "dummy", /* 20 */
- "dummy",
- "dummy",
- "numiptent",
- "unused_privvmpages", /* UB_RESOURCES */
+   "numreads", /* 20 */
+   "numwrites",
+   "numiptent",
+"kbytesread",
+"kbyteswritten",
+   "unused_privvmpages", /* UB_RESOURCES */
    "tmpfs_respages",
    "swap_pages",
    "held_pages",
+   "bytesread",
+   "byteswritten",
+
};

static void init_beancounter_struct(struct user_beancounter *ub);
@@ -623,6 +627,13 @@
    ub->ub_parms[UB_NUMSIGINFO].limit = 1024;
    ub->ub_parms[UB_DCACHESIZE].limit = 1024*1024;
    ub->ub_parms[UB_NUMFILE].limit = 1024;
+   ub->ub_parms[UB_NUMREADS].limit = UB_MAXVALUE;
+   ub->ub_parms[UB_NUMWRITES].limit = UB_MAXVALUE;
+   ub->ub_parms[UB_BYTESREAD].limit = 1024;
+   ub->ub_parms[UB_BYTESWRITTEN].limit = 1024;
+   ub->ub_parms[UB_KBYTESREAD].limit = UB_MAXVALUE;
+   ub->ub_parms[UB_KBYTESWRITTEN].limit = UB_MAXVALUE;
+

    for (k = 0; k < UB_RESOURCES; k++)
        ub->ub_parms[k].barrier = ub->ub_parms[k].limit;
diff -ur linux-2.6.18-FRESH-OPENVZ/kernel/ub/ub_misc.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/kernel/ub/ub_misc.c
--- linux-2.6.18-FRESH-OPENVZ/kernel/ub/ub_misc.c 2006-11-26 04:07:28.000000000 -0500
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/kernel/ub/ub_misc.c 2006-11-28
15:53:18.000000000 -0500
@@ -53,6 +53,8 @@
    new_bc->pgfault_handle = 0;

```

```

    new_bc->pgfault_allot = 0;
}
+int ub_numreads_charge(){
+    struct user_beancounter *ub;
+    int retval;
+    ub=get_exec_ub();
+    retval=charge_beancounter(ub, UB_NUMREADS, 1, UB_FORCE);
+    return retval;
+}
+int ub_numwrites_charge(){
+    struct user_beancounter *ub;
+    int retval;
+    ub=get_exec_ub();
+    retval=charge_beancounter(ub, UB_NUMWRITES, 1, UB_FORCE);
+    return retval;
+}
+ub_numwrites_uncharge(){
+    struct user_beancounter *ub;
+    ub=get_exec_ub();
+    if(ub->ub_parms[UB_NUMWRITES].held < 1) return;
+    uncharge_beancounter(ub, UB_NUMWRITES, 1);
+    return;
+}
+int ub_bytesread_charge(int bytes){
+    ub_numreads_charge();
+    struct user_beancounter *ub;
+    int retval;
+    ub=get_exec_ub();
+    int kbytes_charge=0, uncharge_val=0;
+    if((bytes + ub->ub_parms[UB_BYTESREAD].held)>1024){
+    kbytes_charge=(int)((bytes + ub->ub_parms[UB_BYTESREAD].held)/1024);
+    bytes=(bytes + ub->ub_parms[UB_BYTESREAD].held)-kbytes_charge*1024;
+    uncharge_beancounter(ub,UB_BYTESREAD,ub->ub_parms[UB_BYTESREAD].held);
+    charge_beancounter(ub,UB_BYTESREAD, bytes, UB_FORCE);
+    retval=charge_beancounter(ub,UB_KBYTESREAD,kbytes_charge,UB_FORCE);
+    }else{
+    retval=charge_beancounter(ub,UB_BYTESREAD, bytes, UB_FORCE);
+    }
+    return retval;
+}
+int ub_byteswritten_uncharge(int bytes){
+    struct user_beancounter *ub;
+    int retval;
+    ub=get_exec_ub();
+    int kbytes_uncharge=0;
+    int held=ub->ub_parms[UB_BYTESWRITTEN].held;
+    if((held - bytes)<0 || held==0){
+    kbytes_uncharge=(int)((bytes+held)/1024);

```

```

+         bytes=(kbytes_uncharge*1024-(bytes - held));
+
+         uncharge_beancounter(ub,UB_BYTESWRITTEN,held);
+ charge_beancounter(ub,UB_BYTESWRITTEN, bytes, UB_FORCE);
+ if(ub->ub_parms[UB_KBYTESWRITTEN].held > kbytes_uncharge){
+     uncharge_beancounter(ub,UB_KBYTESWRITTEN,kbytes_uncharge);
+ }
+ }else{
+ uncharge_beancounter(ub,UB_BYTESWRITTEN, bytes);
+
+ }
+ }
+ }
+ int ub_byteswritten_charge(int bytes){
+ ub_numwrites_charge();
+     struct user_beancounter *ub;
+     int retval;
+     ub=get_exec_ub();
+     int kbytes_charge=0;
+     if((bytes + ub->ub_parms[UB_BYTESWRITTEN].held)>1024){
+
+         kbytes_charge=((bytes + ub->ub_parms[UB_BYTESWRITTEN].held)/1024);
+         bytes=(bytes + ub->ub_parms[UB_BYTESWRITTEN].held)-kbytes_charge*1024;
+ if( ub->ub_parms[UB_BYTESWRITTEN].held < 1024 )
+ ub_byteswritten_uncharge(ub->ub_parms[UB_BYTESWRITTEN].held);
+         charge_beancounter(ub,UB_BYTESWRITTEN, bytes, UB_FORCE);
+         retval=charge_beancounter(ub,UB_KBYTESWRITTEN,kbytes_charge, UB_FORCE);
+     }else{
+         retval=charge_beancounter(ub,UB_BYTESWRITTEN, bytes, UB_FORCE);
+     }
+     return retval;
+
+
+ }
+
+
+

```

```

void ub_init_task_bc(struct task_beancounter *tbc)

```

```

{
diff -ur linux-2.6.18-FRESH-OPENVZ/mm/page-writeback.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/page-wri teback.c
--- linux-2.6.18-FRESH-OPENVZ/mm/page-writeback.c 2006-09-19 23:42:06.000000000 -0400
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/page-wri teback.c 2006-11-26
05:49:54.000000000 -0500
@@ -29,6 +29,7 @@
#include <linux/sysctl.h>
#include <linux/cpu.h>
#include <linux/syscalls.h>
+#include <ub/ub_misc.h>

```

```

/*
 * The maximum number of pages to writeout in a single bdflush/kupdate
@@ -623,9 +624,11 @@
    mapping2 = page_mapping(page);
    if (mapping2) { /* Race with truncate? */
        BUG_ON(mapping2 != mapping);
-   if (mapping_cap_account_dirty(mapping))
+   if (mapping_cap_account_dirty(mapping)) {
        __inc_zone_page_state(page,
            NR_FILE_DIRTY);
+   ub_byteswritten_charge(PAGE_CACHE_SIZE);
+   }
    radix_tree_tag_set(&mapping->page_tree,
        page_index(page), PAGECACHE_TAG_DIRTY);
    }
diff -ur linux-2.6.18-FRESH-OPENVZ/mm/readahead.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/readahea d.c
--- linux-2.6.18-FRESH-OPENVZ/mm/readahead.c 2006-09-19 23:42:06.000000000 -0400
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/readahea d.c 2006-11-26
04:36:12.000000000 -0500
@@ -14,6 +14,7 @@
#include <linux/blkdev.h>
#include <linux/backing-dev.h>
#include <linux/pagevec.h>
+#include <ub/ub_misc.h>

void default_unplug_io_fn(struct backing_dev_info *bdi, struct page *page)
{
@@ -143,6 +144,7 @@
    page_cache_release(page);
    continue;
}
+ ub_bytesread_charge(PAGE_CACHE_SIZE);
ret = filler(data, page);
if (!pagevec_add(&lru_pvec, page))
    __pagevec_lru_add(&lru_pvec);
diff -ur linux-2.6.18-FRESH-OPENVZ/mm/truncate.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/truncate .c
--- linux-2.6.18-FRESH-OPENVZ/mm/truncate.c 2006-11-26 04:07:28.000000000 -0500
+++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/truncate .c 2006-11-26
10:22:52.000000000 -0500
@@ -41,8 +41,9 @@

if (PagePrivate(page))
do_invalidatepage(page, 0);
+   if (test_clear_page_dirty(page))
+   ub_byteswritten_uncharge(PAGE_CACHE_SIZE);

```



```
- clear_page_dirty(page);
  ClearPageUptodate(page);
  ClearPageMappedToDisk(page);
  remove_from_page_cache(page);
```

---

---

Subject: Re: OpenVZ patch for VE disk i/o accounting :: 2.6.18  
Posted by [dev](#) on Wed, 29 Nov 2006 13:04:38 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Rick,

looks like Pavel wrote a bit different points, but I want to stress this one:  
writes can happen in arbitrary context (e.g. when memory is being reclaimed  
it is written from arbitrary process context looking for the memory),  
so `get_exec_ub()` imho is not always a valid context to be charged to.

Thanks,  
Kirill

> I fixed a small bug and attached a revised patch.

>

> I tested this patch by running 10 VEs on all night bonnie++ sessions and  
> 10 VEs doing nothing. In the morning, the HN and 10 non bonnie VEs were  
> charged for very little disk access (only maintaining syslog). The 10  
> bonnie VEs were charged for the right amount of disk I/O.

>

> Rick Blundell

>

>

> -----

>

> diff -ur linux-2.6.18-FRESH-OPENVZ/block/ll\_rw\_blk.c  
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/block/ll\_rw\_blk.c  
> --- linux-2.6.18-FRESH-OPENVZ/block/ll\_rw\_blk.c 2006-11-26 04:07:27.000000000 -0500  
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/block/ll\_rw\_blk.c 2006-11-26  
04:36:44.000000000 -0500

> @@ -28,6 +28,7 @@

> #include <linux/interrupt.h>

> #include <linux/cpu.h>

> #include <linux/blktrace\_api.h>

> +#include <ub/ub\_misc.h>

>

> /\*

> \* for max sense size

> @@ -3131,10 +3132,16 @@

> BIO\_BUG\_ON(!bio->bi\_size);

> BIO\_BUG\_ON(!bio->bi\_io\_vec);

```

> bio->bi_rw |= rw;
> - if (rw & WRITE)
> - count_vm_events(PGPGOUT, count);
> - else
> - count_vm_events(PGPGIN, count);
> +
> +   if (rw & WRITE) {
> +       count_vm_events(PGPGOUT, count);
> +   } else {
> + ub_bytesread_charge(bio->bi_size);
> +       count_vm_events(PGPGIN, count);
> +   }
> +
> +
> +
>
> if (unlikely(block_dump)) {
> char b[BDEVNAME_SIZE];
> diff -ur linux-2.6.18-FRESH-OPENVZ/fs/buffer.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/buffer.c
> --- linux-2.6.18-FRESH-OPENVZ/fs/buffer.c 2006-11-26 04:07:28.000000000 -0500
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/buffer.c 2006-11-26
05:52:40.000000000 -0500
> @@ -41,7 +41,7 @@
> #include <linux/bitops.h>
> #include <linux/mpage.h>
> #include <linux/bit_spinlock.h>
> -
> +#include <ub/ub_misc.h>
> static int fsync_buffers_list(spinlock_t *lock, struct list_head *list);
> static void invalidate_bh_lrus(void);
>
> @@ -858,8 +858,10 @@
> if (!TestSetPageDirty(page)) {
> write_lock_irq(&mapping->tree_lock);
> if (page->mapping) { /* Race with truncate? */
> - if (mapping_cap_account_dirty(mapping))
> - __inc_zone_page_state(page, NR_FILE_DIRTY);
> + if (mapping_cap_account_dirty(mapping)) {
> + ub_byteswritten_charge(PAGE_CACHE_SIZE);
> + __inc_zone_page_state(page, NR_FILE_DIRTY);
> + }
> radix_tree_tag_set(&mapping->page_tree,
> page_index(page),
> PAGECACHE_TAG_DIRTY);
> @@ -2862,8 +2864,11 @@
> void ll_rw_block(int rw, int nr, struct buffer_head *bhs[])
> {

```

```

> int i;
> + if (likely(nr) && !(rw & WRITE))
> + ub_bytesread_charge(nr * bhs[0]->b_size);
>
> for (i = 0; i < nr; i++) {
> +
> struct buffer_head *bh = bhs[i];
>
> if (rw == SWRITE)
> @@ -2999,7 +3004,9 @@
> * This only applies in the rare case where try_to_free_buffers
> * succeeds but the page is not freed.
> */
> - clear_page_dirty(page);
> + if (test_clear_page_dirty(page))
> + ub_byteswritten_uncharge(PAGE_CACHE_SIZE);
> +
> }
> spin_unlock(&mapping->private_lock);
> out:
> diff -ur linux-2.6.18-FRESH-OPENVZ/fs/cifs/file.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/cifs/file.c
> --- linux-2.6.18-FRESH-OPENVZ/fs/cifs/file.c 2006-09-19 23:42:06.000000000 -0400
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/cifs/file.c 2006-11-26
04:36:24.000000000 -0500
> @@ -39,6 +39,7 @@
> #include "cifs_unicode.h"
> #include "cifs_debug.h"
> #include "cifs_fs_sb.h"
> +#include <ub/ub_misc.h>
>
> static inline struct cifsFileInfo *cifs_init_private(
> struct cifsFileInfo *private_data, struct inode *inode,
> @@ -1815,6 +1816,7 @@
> }
> break;
> } else if (bytes_read > 0) {
> + ub_bytesread_charge(bytes_read);
> pSMBr = (struct smb_com_read_rsp *)smb_read_data;
> cifs_copy_cache_pages(mapping, page_list, bytes_read,
> smb_read_data + 4 /* RFC1001 hdr */ +
> diff -ur linux-2.6.18-FRESH-OPENVZ/fs/direct-io.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/direct-io.c
> --- linux-2.6.18-FRESH-OPENVZ/fs/direct-io.c 2006-09-19 23:42:06.000000000 -0400
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/fs/direct-io.c 2006-11-26
04:36:01.000000000 -0500
> @@ -35,6 +35,7 @@
> #include <linux/rwsem.h>

```

```

> #include <linux/uio.h>
> #include <asm/atomic.h>
> +#include <ub/ub_misc.h>
>
> /*
>  * How many user pages to map in one call to get_user_pages(). This determines
> @@ -675,6 +676,11 @@
> {
> int ret = 0;
>
> +    if (dio->rw & WRITE) {
> + ub_byteswritten_charge(len);
> +    }
> +
> +
> /*
>  * Can we just grow the current page's presence in the dio?
> */
> diff -ur linux-2.6.18-FRESH-OPENVZ/include/ub/beancounter.h
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/include/ub/ beancounter.h
> --- linux-2.6.18-FRESH-OPENVZ/include/ub/beancounter.h 2006-11-26 04:07:28.000000000
-0500
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/include/ub/ beancounter.h 2006-11-26
06:57:14.000000000 -0500
> @@ -71,13 +71,21 @@
> #define UB_NUMOTHERSOCK 17 /* Number of other sockets. */
> #define UB_DCACHESIZE 18 /* Size of busy dentry/inode cache. */
> #define UB_NUMFILE 19 /* Number of open files. */
> +#define UB_NUMREADS    20
> +#define UB_NUMWRITES  21
> +#define UB_KBYTESREAD 23
> +#define UB_KBYTESWRITTEN 24
> +
> +
> +#define UB_RESOURCES  25
>
> -#define UB_RESOURCES 24
>
> #define UB_UNUSEDPRIVVM (UB_RESOURCES + 0)
> #define UB_TMPFSPAGES (UB_RESOURCES + 1)
> #define UB_SWAPPAGES (UB_RESOURCES + 2)
> #define UB_HELDPAGES (UB_RESOURCES + 3)
> +#define UB_BYTESREAD (UB_RESOURCES + 4)
> +#define UB_BYTESWRITTEN (UB_RESOURCES + 5)
>
> struct ubparm {
> /*
> diff -ur linux-2.6.18-FRESH-OPENVZ/include/ub/ub_misc.h

```

```

kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/include/ub/ ub_misc.h
> --- linux-2.6.18-FRESH-OPENVZ/include/ub/ub_misc.h 2006-11-26 04:07:28.000000000 -0500
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/include/ub/ ub_misc.h 2006-11-26
06:25:02.000000000 -0500
> @@ -18,6 +18,12 @@
> struct file_lock;
> struct sigqueue;
>
> +UB_DECLARE_FUNC(int, ub_numreads_charge())
> +UB_DECLARE_FUNC(int, ub_numwrites_charge())
> +UB_DECLARE_FUNC(int, ub_numwrites_uncharge())
> +UB_DECLARE_FUNC(int, ub_byteswritten_charge(int bytes))
> +UB_DECLARE_FUNC(int, ub_byteswritten_uncharge(int bytes))
> +UB_DECLARE_FUNC(int, ub_bytesread_charge(int bytes))
> UB_DECLARE_FUNC(int, ub_file_charge(struct file *f))
> UB_DECLARE_VOID_FUNC(ub_file_uncharge(struct file *f))
> UB_DECLARE_FUNC(int, ub_flock_charge(struct file_lock *fl, int hard))
> diff -ur linux-2.6.18-FRESH-OPENVZ/kernel/ub/beancounter.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/kernel/ub/b eancounter.c
> --- linux-2.6.18-FRESH-OPENVZ/kernel/ub/beancounter.c 2006-11-26 04:07:28.000000000
-0500
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/kernel/ub/b eancounter.c 2006-11-26
08:22:41.000000000 -0500
> @@ -60,14 +60,18 @@
> "numothersock",
> "dcachesize",
> "numfile",
> - "dummy", /* 20 */
> - "dummy",
> - "dummy",
> - "numiptent",
> - "unused_privvmpages", /* UB_RESOURCES */
> + "numreads", /* 20 */
> + "numwrites",
> + "numiptent",
> +"kbytesread",
> +"kbyteswritten",
> + "unused_privvmpages", /* UB_RESOURCES */
> "tmpfs_respages",
> "swap_pages",
> "held_pages",
> + "bytesread",
> + "byteswritten",
> +
> };
>
> static void init_beancounter_struct(struct user_beancounter *ub);
> @@ -623,6 +627,13 @@

```

```

> ub->ub_parms[UB_NUMSIGINFO].limit = 1024;
> ub->ub_parms[UB_DCACHESIZE].limit = 1024*1024;
> ub->ub_parms[UB_NUMFILE].limit = 1024;
> +   ub->ub_parms[UB_NUMREADS].limit = UB_MAXVALUE;
> +   ub->ub_parms[UB_NUMWRITES].limit = UB_MAXVALUE;
> +   ub->ub_parms[UB_BYTESREAD].limit = 1024;
> +   ub->ub_parms[UB_BYTESWRITTEN].limit = 1024;
> +   ub->ub_parms[UB_KBYTESREAD].limit = UB_MAXVALUE;
> +   ub->ub_parms[UB_KBYTESWRITTEN].limit = UB_MAXVALUE;
> +
>
> for (k = 0; k < UB_RESOURCES; k++)
>   ub->ub_parms[k].barrier = ub->ub_parms[k].limit;
> diff -ur linux-2.6.18-FRESH-OPENVZ/kernel/ub/ub_misc.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/kernel/ub/u b_misc.c
> --- linux-2.6.18-FRESH-OPENVZ/kernel/ub/ub_misc.c 2006-11-26 04:07:28.000000000 -0500
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/kernel/ub/u b_misc.c 2006-11-28
15:53:18.000000000 -0500
> @@ -53,6 +53,86 @@
>   new_bc->pgfault_handle = 0;
>   new_bc->pgfault_allot = 0;
> }
> +int ub_numreads_charge(){
> +   struct user_beancounter *ub;
> +   int retval;
> +   ub=get_exec_ub();
> +   retval=charge_beancounter(ub, UB_NUMREADS, 1, UB_FORCE);
> +   return retval;
> +}
> +int ub_numwrites_charge(){
> +   struct user_beancounter *ub;
> +   int retval;
> +   ub=get_exec_ub();
> +   retval=charge_beancounter(ub, UB_NUMWRITES, 1, UB_FORCE);
> +   return retval;
> +}
> +ub_numwrites_uncharge(){
> +   struct user_beancounter *ub;
> +   ub=get_exec_ub();
> +   if(ub->ub_parms[UB_NUMWRITES].held < 1) return;
> +   uncharge_beancounter(ub, UB_NUMWRITES, 1);
> +   return;
> +}
> +int ub_bytesread_charge(int bytes){
> +   ub_numreads_charge();
> +   struct user_beancounter *ub;
> +   int retval;
> +   ub=get_exec_ub();

```

```

> + int kbytes_charge=0, uncharge_val=0;
> + if((bytes + ub->ub_parms[UB_BYTESREAD].held)>1024){
> + kbytes_charge=(int)((bytes + ub->ub_parms[UB_BYTESREAD].held)/1024);
> + bytes=(bytes + ub->ub_parms[UB_BYTESREAD].held)-kbytes_charge*1024;
> + uncharge_beancounter(ub,UB_BYTESREAD,ub->ub_parms[UB_BYTESREAD].held);
> + charge_beancounter(ub,UB_BYTESREAD, bytes, UB_FORCE);
> + retval=charge_beancounter(ub,UB_KBYTESREAD,kbytes_charge,UB_FORCE);
> + }else{
> + retval=charge_beancounter(ub,UB_BYTESREAD, bytes, UB_FORCE);
> + }
> + return retval;
> + }
> +int ub_byteswritten_uncharge(int bytes){
> + struct user_beancounter *ub;
> + int retval;
> + ub=get_exec_ub();
> + int kbytes_uncharge=0;
> + int held=ub->ub_parms[UB_BYTESWRITTEN].held;
> + if((held - bytes)<0 || held==0){
> + kbytes_uncharge=(int)((bytes+held)/1024);
> + bytes=(kbytes_uncharge*1024-(bytes - held));
> +
> + uncharge_beancounter(ub,UB_BYTESWRITTEN,held);
> + charge_beancounter(ub,UB_BYTESWRITTEN, bytes, UB_FORCE);
> + if(ub->ub_parms[UB_KBYTESWRITTEN].held > kbytes_uncharge){
> + uncharge_beancounter(ub,UB_KBYTESWRITTEN,kbytes_uncharge);
> + }
> + }else{
> + uncharge_beancounter(ub,UB_BYTESWRITTEN, bytes);
> +
> + }
> + }
> +int ub_byteswritten_charge(int bytes){
> + ub_numwrites_charge();
> + struct user_beancounter *ub;
> + int retval;
> + ub=get_exec_ub();
> + int kbytes_charge=0;
> + if((bytes + ub->ub_parms[UB_BYTESWRITTEN].held)>1024){
> +
> + kbytes_charge=((bytes + ub->ub_parms[UB_BYTESWRITTEN].held)/1024);
> + bytes=(bytes + ub->ub_parms[UB_BYTESWRITTEN].held)-kbytes_charge*1024;
> + if( ub->ub_parms[UB_BYTESWRITTEN].held < 1024 )
> + ub_byteswritten_uncharge(ub->ub_parms[UB_BYTESWRITTEN].held);
> + charge_beancounter(ub,UB_BYTESWRITTEN, bytes, UB_FORCE);
> + retval=charge_beancounter(ub,UB_KBYTESWRITTEN,kbytes_charge,
UB_FORCE);
> + }else{

```

```

> +         retval=charge_beancounter(ub,UB_BYTESWRITTEN, bytes, UB_FORCE);
> +     }
> +     return retval;
> +
> +}
> +
> +
>
> void ub_init_task_bc(struct task_beancounter *tbc)
> {
> diff -ur linux-2.6.18-FRESH-OPENVZ/mm/page-writeback.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/page-wri teback.c
> --- linux-2.6.18-FRESH-OPENVZ/mm/page-writeback.c 2006-09-19 23:42:06.000000000 -0400
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/page-wri teback.c 2006-11-26
05:49:54.000000000 -0500
> @@ -29,6 +29,7 @@
> #include <linux/sysctl.h>
> #include <linux/cpu.h>
> #include <linux/syscalls.h>
> +#include <ub/ub_misc.h>
>
> /*
> * The maximum number of pages to writeout in a single bdflush/kupdate
> @@ -623,9 +624,11 @@
>     mapping2 = page_mapping(page);
>     if (mapping2) { /* Race with truncate? */
>         BUG_ON(mapping2 != mapping);
> -     if (mapping_cap_account_dirty(mapping))
> +     if (mapping_cap_account_dirty(mapping)) {
>         __inc_zone_page_state(page,
>             NR_FILE_DIRTY);
> +     ub_byteswritten_charge(PAGE_CACHE_SIZE);
> +     }
>     radix_tree_tag_set(&mapping->page_tree,
>         page_index(page), PAGECACHE_TAG_DIRTY);
>     }
> diff -ur linux-2.6.18-FRESH-OPENVZ/mm/readahead.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/readahea d.c
> --- linux-2.6.18-FRESH-OPENVZ/mm/readahead.c 2006-09-19 23:42:06.000000000 -0400
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/readahea d.c 2006-11-26
04:36:12.000000000 -0500
> @@ -14,6 +14,7 @@
> #include <linux/blkdev.h>
> #include <linux/backing-dev.h>
> #include <linux/pagevec.h>
> +#include <ub/ub_misc.h>
>
> void default_unplug_io_fn(struct backing_dev_info *bdi, struct page *page)

```



```
> {
> @@ -143,6 +144,7 @@
>   page_cache_release(page);
>   continue;
> }
> + ub_bytesread_charge(PAGE_CACHE_SIZE);
>   ret = filler(data, page);
>   if (!pagevec_add(&lru_pvec, page))
>       __pagevec_lru_add(&lru_pvec);
> diff -ur linux-2.6.18-FRESH-OPENVZ/mm/truncate.c
kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/truncate.c
> --- linux-2.6.18-FRESH-OPENVZ/mm/truncate.c 2006-11-26 04:07:28.000000000 -0500
> +++ kernel-2.6.18-rjb-004-rev-028-005.1/linux-2.6.18/mm/truncate.c 2006-11-26
10:22:52.000000000 -0500
> @@ -41,8 +41,9 @@
>
>   if (PagePrivate(page))
>       do_invalidatepage(page, 0);
> +   if (test_clear_page_dirty(page))
> +       ub_byteswritten_uncharge(PAGE_CACHE_SIZE);
>
> - clear_page_dirty(page);
>   ClearPageUptodate(page);
>   ClearPageMappedToDisk(page);
>   remove_from_page_cache(page);
>
>
> -----
>
```

---