

# [PATCH 0/3] ia64: Migrate data off physical pages with correctable errors v6

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-06/msg03562.html>

---

- *From:* Russ Anderson <[rja@xxxxxxx](mailto:rja@xxxxxxx)>
  - *Date:* Mon, 9 Jun 2008 11:18:06 -0500
- 

Migrate data off physical pages with corrected memory errors (Version 6)

Version 6 changes:

page.cleanup.v6:

- Fix cut–paste comment (per Linus Torvalds's request).

page.discard.v6:

- Fixed a problem where a page failed to migrate would end up with an extra reference count (per Christoph Lameter's request).
- Fixed comments (per Christoph Lameter's request).
- Moved totalbad\_pages definition from mm/migrate.c to arch/ia64/kernel/mca.c (per Christoph Lameter's request).

cpe.migrate.v6:

- Move totalbad\_pages from mm/migrate.c to ia64/kernel/mca.c.
- Replace tab with space (per Christoph Lameter's request).

Version 5 changes:

page.cleanup.v5:

- Change names to reflect the use and add comments to explain the meaning. (per Linus Torvalds's request).

Version 4 changes:

page.discard.v4:

- Remove the hot path checks in lru\_cache\_add() and lru\_cache\_add\_active(). Avoid moving the bad page to the LRU in unmap\_and\_move() and putback\_lru\_pages(). (per Linus Torvalds's request).

cpe.migrate.v4

- More code style cleanup (per Andrew Morton's request).
- Removed locking when calling the migration code (per Christoph Lameter's request).
- If the page fails to migrate, clear the PG\_memerror flag. This avoids a page with PG\_memerror on the free list.

## [PATCH 0/3] ia64: Migrate data off physical pages with correctable errors v6

Version 3 changes:

page.cleanup.v3:

- Put PAGE\_FLAGS definitions back in page-flags.h (per Christoph Lameter's request).

cpe.migrate.v3

- Use putback\_lru\_pages() when returning an individual page (per Christoph Lameter's request).
- Code style cleanup (per Pekka Enberg's request).
- Use strict\_strtoul() (per Pekka Enberg's request).
- Added locking (per Pekka Enberg's request).
- Use /sys/kernel/ instead of /proc (per Pekka Enberg's request).

Version 2 changes:

Broke the page.discard patch into two patches, per request by Christoph Lameter.

page.cleanup.v2:

- minor clean-up of page flags in page\_alloc.c.

page.discard.v2:

- Updated for recent page flag clean-up.
- Removed the change to the sysinfo struct.

cpe.migrate.v2

- Added /proc/badram interface to print page discard information and to free bad pages.

Purpose:

Physical memory with corrected errors may decay over time into uncorrectable errors. The purpose of this patch is to move the data off pages with correctable memory errors before the memory goes bad.

The patches:

[1/3] page.cleanup.v6: Minor clean-up of page flags in mm/page\_alloc.c

Minor source code clean-up of page flags in mm/page\_alloc.c.

The cleanup makes it easier for the next patch to add PG\_memerror.

[2/3] page.discard.v6: Avoid putting a bad page back on the LRU.

page.discard.v6 are the arch independent changes. It adds a new page flag (PG\_memerror) to mark the page as bad and avoids putting

## [PATCH 0/3] ia64: Migrate data off physical pages with correctable errors v6

the page back on the LRU after migrating the data to a new page. The reference count on the bad page is not decremented to zero to avoid it being reallocated. PG\_memerror is only defined if CONFIG\_PAGEFLAGS\_EXTENDED is defined.

[3/3] cpe.migrate.v6: Call migration code on correctable errors

cpe.migrate.v6 are the IA64 specific changes. It connects the CPE handler to the page migration code. It is implemented as a kernel loadable module, similar to the mca recovery code (mca\_recovery.ko), so that it can be removed to turn off the feature. Create /sys/kernel/badram to print page discard information and to free bad pages.

Comments:

There is always an issue of how aggressive the code should be on migrating pages. Should it migrate on the first correctable error, or wait for some threshold? Reasonable people may disagree on the threshold and the "right" answer may be hardware specific. The decision making is confined to the cpe\_migrate.c code and can be built as a kernel loadable module. It is currently set to migrate on the first correctable error.

Only pages that can be isolated on the LRU are migrated. Other pages, such as compound pages, are not migrated. That functionality could be a future enhancement.

/sys/kernel/badram is a way of displaying information about the bad memory and freeing the bad pages. A userspace program (or sysadmin) could determine if a discarded page needs to be freed.

Sample output:

```
linux> insmod cpe_migrate.ko
linux> cat /sys/kernel/badram // This shows no discarded memory
Bad RAM: 0 kB, 0 pages marked bad
List of bad physical pages

linux> ./errsingle -c 6 -s 1 // Inject correctable errors on
// six pages.
linux> cat /sys/kernel/badram
Bad RAM: 384 kB, 6 pages marked bad
List of bad physical pages
0x06048e10000 0x06870c40000 0x06870c20000 0x06870c10000 0x06007f00000
0x06042070000

linux> echo 0x06870c20000 > /sys/kernel/badram // Free one of the pages

linux> cat /sys/kernel/badram // Five pages remain on the list
Bad RAM: 320 kB, 5 pages marked bad
```

## [PATCH 0/3] ia64: Migrate data off physical pages with correctable errors v6

List of bad physical pages

0x06048e10000 0x06870c40000 0x06870c10000 0x06007f00000 0x06042070000

```
linux> echo 0 > /sys/kernel/badram // Free all the bad pages
```

```
linux> cat /sys/kernel/badram // All the pages are freed
```

Bad RAM: 0 kB, 0 pages marked bad

List of bad physical pages

Flow of the code description (while testing on IA64):

- 1) A user level application test program allocates memory and passes the virtual address of the memory to the error injection driver.
- 2) The error injection driver converts the virtual address to physical, functions the Altix hardware to modify the ECC for the physical page, creating a correctable error, and returns to the user application.
- 3) The user application reads the memory.
- 4) The Altix hardware detects the correctable error and interrupts prom. SAL builds a CPU error record, then sends a CPE interrupt to linux.
- 5) The linux CPE handler calls the `cpe_migrate` module (if installed).
- 6) `cpe_migrate` parses the physical address from the CPE record and adds the address to the migrate list (if not already on the list) and schedules the worker thread (`cpe_enable_work`).
- 7) `cpe_enable_work` calls `ia64_mca_cpe_move_page`.
- 8) `ia64_mca_cpe_move_page` validates the physical address, converts to page, sets `PG_memerror` flag and calls the migration code (`migrate_prep()`, `isolate_lru_page()`, and `migrate_pages()`). If the page migrates successfully, the bad page is added to `badpagelist`.
- 9) Because `PG_memerror` is set, the bad page is not added back on the LRU by avoiding calls to `move_to_lru()`. Avoiding `move_to_lru()` prevents the page count from being decremented to zero.
- 10) If the page fails to migrate, `PG_memerror` is cleared and the page returned to the LRU. If another correctable error occurs on the page another attempt will be made to migrate it.

--

Russ Anderson, OS RAS/Partitioning Project Lead  
SGI – Silicon Graphics Inc rja@xxxxxxx

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>