

## [PATCH] ppc64: fix memcpy\_to/from\_io

**Source:** <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-07/6111.html>

---

**From:** Benjamin Herrenschmidt ([benh\\_at\\_kernel.crashing.org](mailto:benh_at_kernel.crashing.org))

**Date:** 07/30/04

To: Andrew Morton <[akpm@osdl.org](mailto:akpm@osdl.org)>

Date: Fri, 30 Jul 2004 14:22:19 +1000

Hi !

The ppc64 implementation of memcpy\_to/from\_io was bogus (used memcpy which uses cache hints and thus is broken on non cacheable IO space).

This re-implements them with some simple/gross C code doing 32 bits accesses when aligned and bytes accesses when not.

Signed-off-by: Benjamin Herrenschmidt <[benh@kernel.crashing.org](mailto:benh@kernel.crashing.org)>

==== include/asm-ppc64/eeh.h 1.12 vs ? (writable without lock!) =====

--- 1.12/include/asm-ppc64/eeh.h 2004-07-02 15:23:45 +10:00

+++ ?/include/asm-ppc64/eeh.h 2004-07-30 14:19:48 +10:00

@@ -180,26 +180,95 @@

out\_be64(vaddr, val);

}

+#define EEH\_CHECK\_ALIGN(v,a) \

+ (((unsigned long)(v)) & ((a) - 1)) == 0)

+

static inline void eeh\_memset\_io(void \*addr, int c, unsigned long n) {

void \*vaddr = (void \*)IO\_TOKEN\_TO\_ADDR(addr);

- memset(vaddr, c, n);

+ u32 lc = c;

+ lc |= lc << 8;

+ lc |= lc << 16;

+

+ while(n && !EEH\_CHECK\_ALIGN(vaddr, 4)) {

+ \*((volatile u8 \*)vaddr) = c;

+ vaddr = (void \*)((unsigned long)vaddr + 1);

+ n--;

+ }

+ while(n >= 4) {

+ \*((volatile u32 \*)vaddr) = lc;

+ vaddr = (void \*)((unsigned long)vaddr + 4);

+ n -= 4;

+ }

```

+ while(n) {
+ *((volatile u8 *)vaddr) = c;
+ vaddr = (void *)((unsigned long)vaddr + 1);
+ n--;
+ }
+ __asm__ __volatile__ ("sync" ::: "memory");
+ }
static inline void eeh_memcpy_fromio(void *dest, void *src, unsigned long n) {
    void *vsrc = (void *)IO_TOKEN_TO_ADDR(src);
- memcpy(dest, vsrc, n);
+ void *vsrccsave = vsrc, *destdsave = dest, *srccsave = src;
+ unsigned long nsave = n;
+
+ while(n && (!EEH_CHECK_ALIGN(vsrc, 4) || !EEH_CHECK_ALIGN(dest, 4))) {
+ *((u8 *)dest) = *((volatile u8 *)vsrc);
+ __asm__ __volatile__ ("eieio" ::: "memory");
+ vsrc = (void *)((unsigned long)vsrc + 1);
+ dest = (void *)((unsigned long)dest + 1);
+ n--;
+ }
+ while(n > 4) {
+ *((u32 *)dest) = *((volatile u32 *)vsrc);
+ __asm__ __volatile__ ("eieio" ::: "memory");
+ vsrc = (void *)((unsigned long)vsrc + 4);
+ dest = (void *)((unsigned long)dest + 4);
+ n -= 4;
+ }
+ while(n) {
+ *((u8 *)dest) = *((volatile u8 *)vsrc);
+ __asm__ __volatile__ ("eieio" ::: "memory");
+ vsrc = (void *)((unsigned long)vsrc + 1);
+ dest = (void *)((unsigned long)dest + 1);
+ n--;
+ }
+ __asm__ __volatile__ ("sync" ::: "memory");
+
    /* Look for ffff's here at dest[n]. Assume that at least 4 bytes
    * were copied. Check all four bytes.
    */
- if ((n >= 4) &&
- (EEH_POSSIBLE_ERROR(src, vsrc, (*((u32 *) dest+n-4)), u32))) {
- eeh_check_failure(src, (*((u32 *) dest+n-4)));
+ if ((nsave >= 4) &&
+ (EEH_POSSIBLE_ERROR(srccsave, vsrccsave, (*((u32 *) destdsave+nsave-4)),
+ u32))) {
+ eeh_check_failure(srccsave, (*((u32 *) destdsave+nsave-4)));
    }
+ }

static inline void eeh_memcpy_toio(void *dest, void *src, unsigned long n) {
    void *vdest = (void *)IO_TOKEN_TO_ADDR(dest);

```

Linux-Kernel: [PATCH] ppc64: fix memcpy\_to/from\_io

```
- memcpy(vdest, src, n);
+
+ while(n && (!EEH_CHECK_ALIGN(vdest, 4) || !EEH_CHECK_ALIGN(src, 4))) {
+ *((volatile u8 *)vdest) = *((u8 *)src);
+ src = (void *)((unsigned long)src + 1);
+ vdest = (void *)((unsigned long)vdest + 1);
+ n--;
+ }
+ while(n > 4) {
+ *((volatile u32 *)vdest) = *((volatile u32 *)src);
+ src = (void *)((unsigned long)src + 4);
+ vdest = (void *)((unsigned long)vdest + 4);
+ n-=4;
+ }
+ while(n) {
+ *((volatile u8 *)vdest) = *((u8 *)src);
+ src = (void *)((unsigned long)src + 1);
+ vdest = (void *)((unsigned long)vdest + 1);
+ n--;
+ }
+ __asm__ __volatile__ ("sync" ::: "memory");
+ }
+
+#undef EEH_CHECK_ALIGN

#define MAX_ISA_PORT 0x10000
extern unsigned long io_page_mask;
```

-

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

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

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