

[PATCH 2.6.21-rc2] ehca: fix mismatched sync between completion handler and destroy cq

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg10105.html>

- From: Hoang-Nam Nguyen <hnguyen@xxxxxxxxxxxxxxxxxxxx>
- Date: Wed, 28 Feb 2007 18:01:02 +0100

This patch fixes two issues reported by Roland and Christoph H.:

- Mismatched sync/locking between completion handler and destroy cq

We introduced a counter nr_events per cq to track number of irq events seen. This counter is incremented when an event queue entry is seen and decremented after completion handler has been called regardless if scaling code is active or not. Note that nr_callbacks tracks number of events assigned to a cpu and both counters can potentially diverge.

The sync between running completion handler and destroy cq is done by using the global spin lock ehca_cq_idr_lock.

- Replace yield by wait_event on the counter above to become zero

Signed-off-by: Hoang-Nam Nguyen <hnguyen@xxxxxxxxxx>

```
ehca_classes.h | 6 ++++
ehca_cq.c | 16 ++++++++
ehca_irq.c | 59 ++++++++++++++++++++++++++++++++++++++-----
ehca_main.c | 4 +--
4 files changed, 60 insertions(+), 25 deletions(-)
```

```
diff --git a/drivers/infiniband/hw/ehca/ehca_classes.h b/drivers/infiniband/hw/ehca/ehca_classes.h
index 40404c9..85fe741 100644
```

```
--- a/drivers/infiniband/hw/ehca/ehca_classes.h
+++ b/drivers/infiniband/hw/ehca/ehca_classes.h
@@ -52,6 +52,8 @@ struct ehca_mw;
struct ehca_pd;
struct ehca_av;
```

```
+#include <linux/completion.h>
+
#include <rdma/ib_verbs.h>
#include <rdma/ib_user_verbs.h>
```

```
@@ -153,7 +155,9 @@ struct ehca_cq {
```

[PATCH 2.6.21-rc2] ehca: fix mismatched sync between completion handler and destroy cq

```
spinlock_t cb_lock;
struct hlist_head qp_hashtab[QP_HASHTAB_LEN];
struct list_head entry;
- u32 nr_callbacks;
+ u32 nr_callbacks; /* #events assigned to cpu by scaling code */
+ u32 nr_events; /* #events seen */
+ wait_queue_head_t wait_completion;
spinlock_t task_lock;
u32 ownpid;
/* mmap counter for resources mapped into user space */
diff --git a/drivers/infiniband/hw/ehca/ehca_cq.c b/drivers/infiniband/hw/ehca/ehca_cq.c
index 6ebfa27..e2cdc1a 100644
--- a/drivers/infiniband/hw/ehca/ehca_cq.c
+++ b/drivers/infiniband/hw/ehca/ehca_cq.c
@@ -146,6 +146,7 @@ struct ib_cq *ehca_create_cq(struct ib_d
spin_lock_init(&my_cq->spinlock);
spin_lock_init(&my_cq->cb_lock);
spin_lock_init(&my_cq->task_lock);
+ init_waitqueue_head(&my_cq->wait_completion);
my_cq->ownpid = current->tgid;

cq = &my_cq->ib_cq;
@@ -302,6 +303,16 @@ create_cq_exit1:
return cq;
}

+static int get_cq_nr_events(struct ehca_cq *my_cq)
+{
+ int ret;
+ unsigned long flags;
+ spin_lock_irqsave(&ehca_cq_idr_lock, flags);
+ ret = my_cq->nr_events;
+ spin_unlock_irqrestore(&ehca_cq_idr_lock, flags);
+ return ret;
+}
+
int ehca_destroy_cq(struct ib_cq *cq)
{
u64 h_ret;
@@ -329,10 +340,11 @@ int ehca_destroy_cq(struct ib_cq *cq)
}

spin_lock_irqsave(&ehca_cq_idr_lock, flags);
- while (my_cq->nr_callbacks) {
+ while (my_cq->nr_events) {
spin_unlock_irqrestore(&ehca_cq_idr_lock, flags);
- yield();
+ wait_event(my_cq->wait_completion, !get_cq_nr_events(my_cq));
spin_lock_irqsave(&ehca_cq_idr_lock, flags);
+ /* recheck nr_events to assure no cq has just arrived */
}
}
```

```

idr_remove(&ehca_cq_idr, my_cq->token);
diff --git a/drivers/infiniband/hw/ehca/ehca_irq.c b/drivers/infiniband/hw/ehca/ehca_irq.c
index 3ec53c6..7d8b795 100644
--- a/drivers/infiniband/hw/ehca/ehca_irq.c
+++ b/drivers/infiniband/hw/ehca/ehca_irq.c
@@ -404,10 +403,11 @@ static inline void process_eqe(struct eh
u32 token;
unsigned long flags;
struct ehca_cq *cq;
+
eqe_value = eqe->entry;
ehca_dbg(&shca->ib_device, "eqe_value=%lx", eqe_value);
if (EHCA_BMASK_GET(EQE_COMPLETION_EVENT, eqe_value)) {
- ehca_dbg(&shca->ib_device, "... completion event");
+ ehca_dbg(&shca->ib_device, "Got completion event");
token = EHCA_BMASK_GET(EQE_CQ_TOKEN, eqe_value);
spin_lock_irqsave(&ehca_cq_idr_lock, flags);
cq = idr_find(&ehca_cq_idr, token);
@@ -419,16 +419,20 @@ static inline void process_eqe(struct eh
return;
}
reset_eq_pending(cq);
- if (ehca_scaling_code) {
+ cq->nr_events++;
+ spin_unlock_irqrestore(&ehca_cq_idr_lock, flags);
+ if (ehca_scaling_code)
queue_comp_task(cq);
- spin_unlock_irqrestore(&ehca_cq_idr_lock, flags);
- } else {
- spin_unlock_irqrestore(&ehca_cq_idr_lock, flags);
+ else {
comp_event_callback(cq);
+ spin_lock_irqsave(&ehca_cq_idr_lock, flags);
+ cq->nr_events--;
+ if (!cq->nr_events)
+ wake_up(&cq->wait_completion);
+ spin_unlock_irqrestore(&ehca_cq_idr_lock, flags);
}
} else {
- ehca_dbg(&shca->ib_device,
- "Got non completion event");
+ ehca_dbg(&shca->ib_device, "Got non completion event");
parse_identifier(shca, eqe_value);
}
}
@@ -478,6 +482,7 @@ void ehca_process_eq(struct ehca_shca *s
"token=%x", token);
continue;
}
+ eqe_cache[eqe_cnt].cq->nr_events++;

```

[PATCH 2.6.21-rc2] ehca: fix mismatched sync between completion handler and destroy cq

```
spin_unlock(&ehca_cq_idr_lock);
} else
eqe_cache[eqe_cnt].cq = NULL;
@@ -504,12 +509,18 @@ void ehca_process_eq(struct ehca_shca *s
/* call completion handler for cached eqes */
for (i = 0; i < eqe_cnt; i++)
if (eq->eqe_cache[i].cq) {
- if (ehca_scaling_code) {
- spin_lock(&ehca_cq_idr_lock);
+ if (ehca_scaling_code)
queue_comp_task(eq->eqe_cache[i].cq);
- spin_unlock(&ehca_cq_idr_lock);
- } else
- comp_event_callback(eq->eqe_cache[i].cq);
+ else {
+ struct ehca_cq *cq = eq->eqe_cache[i].cq;
+ comp_event_callback(cq);
+ spin_lock_irqsave(&ehca_cq_idr_lock, flags);
+ cq->nr_events--;
+ if (!cq->nr_events)
+ wake_up(&cq->wait_completion);
+ spin_unlock_irqrestore(&ehca_cq_idr_lock,
+ flags);
+ }
} else {
ehca_dbg(&shca->ib_device, "Got non completion event");
parse_identfier(shca, eq->eqe_cache[i].eqe->entry);
@@ -523,7 +534,6 @@ void ehca_process_eq(struct ehca_shca *s
if (!eqe)
break;
process_eqe(shca, eqe);
- eqe_cnt++;
} while (1);

unlock_irq_spinlock:
@@ -567,8 +577,7 @@ static void __queue_comp_task(struct ehc
list_add_tail(&__cq->entry, &cct->cq_list);
cct->cq_jobs++;
wake_up(&cct->wait_queue);
- }
- else
+ } else
__cq->nr_callbacks++;

spin_unlock(&__cq->task_lock);
@@ -577,18 +586,21 @@ static void __queue_comp_task(struct ehc

static void queue_comp_task(struct ehca_cq *__cq)
{
- int cpu;
int cpu_id;
```

[PATCH 2.6.21-rc2] ehca: fix mismatched sync between completion handler and destroy cq

```
struct ehca_cpu_comp_task *cct;
+ int cq_jobs;
+ unsigned long flags;

- cpu = get_cpu();
cpu_id = find_next_online_cpu(pool);
BUG_ON(!cpu_online(cpu_id));

cct = per_cpu_ptr(pool->cpu_comp_tasks, cpu_id);
BUG_ON(!cct);

- if (cct->cq_jobs > 0) {
+ spin_lock_irqsave(&cct->task_lock, flags);
+ cq_jobs = cct->cq_jobs;
+ spin_unlock_irqrestore(&cct->task_lock, flags);
+ if (cq_jobs > 0) {
cpu_id = find_next_online_cpu(pool);
cct = per_cpu_ptr(pool->cpu_comp_tasks, cpu_id);
BUG_ON(!cct);
@@ -608,11 +620,17 @@ static void run_comp_task(struct ehca_cp
cq = list_entry(cct->cq_list.next, struct ehca_cq, entry);
spin_unlock_irqrestore(&cct->task_lock, flags);
comp_event_callback(cq);
- spin_lock_irqsave(&cct->task_lock, flags);

+ spin_lock_irqsave(&ehca_cq_idr_lock, flags);
+ cq->nr_events--;
+ if (!cq->nr_events)
+ wake_up(&cq->wait_completion);
+ spin_unlock_irqrestore(&ehca_cq_idr_lock, flags);
+
+ spin_lock_irqsave(&cct->task_lock, flags);
spin_lock(&cq->task_lock);
cq->nr_callbacks--;
- if (cq->nr_callbacks == 0) {
+ if (!cq->nr_callbacks) {
list_del_init(cct->cq_list.next);
cct->cq_jobs--;
}
diff --git a/drivers/infiniband/hw/ehca/ehca_main.c b/drivers/infiniband/hw/ehca/ehca_main.c
index c183512..a5e564a 100644
--- a/drivers/infiniband/hw/ehca/ehca_main.c
+++ b/drivers/infiniband/hw/ehca/ehca_main.c
@@ -52,7 +52,7 @@ #include "hca_if.h"
MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("Christoph Raisch <raisch@xxxxxxxxxx>");
MODULE_DESCRIPTION("IBM eServer HCA InfiniBand Device Driver");
-MODULE_VERSION("SVNEHCA_0021");
+MODULE_VERSION("SVNEHCA_0022");

int ehca_open_aqp1 = 0;
```

[PATCH 2.6.21-rc2] ehca: fix mismatched sync between completion handler and destroy cq

```
int ehca_debug_level = 0;
@@ -810,7 +809,7 @@ int __init ehca_module_init(void)
int ret;
```

```
printk(KERN_INFO "eHCA Infiniband Device Driver "
- "(Rel.: SVNEHCA_0021)\n");
+ "(Rel.: SVNEHCA_0022)\n");
idr_init(&ehca_qp_idr);
idr_init(&ehca_cq_idr);
spin_lock_init(&ehca_qp_idr_lock);
```

—
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>