

# [PATCH 4/6] iSER RDMA CM (CMA) and IB verbs interaction

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-05/msg02652.html>

---

- *From:* Or Gerlitz <[ogerlitz@xxxxxxxxxxxxx](mailto:ogerlitz@xxxxxxxxxxxxx)>
  - *Date:* Thu, 11 May 2006 10:02:46 +0300 (IDT)
- 

This file contains the low level interaction with the RDMA CM and the IB verbs, where iSER is consumer of both.

Signed-off-by: Or Gerlitz <[ogerlitz@xxxxxxxxxxxxx](mailto:ogerlitz@xxxxxxxxxxxxx)>

```
--- /usr/src/linux-2.6.17-rc3/drivers/infiniband/ulp/iser-x/iser_verbs.c 1970-01-01 02:00:00.000000000
+0200
+++ /usr/src/linux-2.6.17-rc3/drivers/infiniband/ulp/iser/iser_verbs.c 2006-05-10 15:32:01.000000000
+0300
@@ -0,0 +1,827 @@
+/*
+ * Copyright (c) 2004, 2005, 2006 Voltaire, Inc. All rights reserved.
+ * Copyright (c) 2005, 2006 Cisco Systems. All rights reserved.
+ *
+ * This software is available to you under a choice of one of two
+ * licenses. You may choose to be licensed under the terms of the GNU
+ * General Public License (GPL) Version 2, available from the file
+ * COPYING in the main directory of this source tree, or the
+ * OpenIB.org BSD license below:
+ *
+ * Redistribution and use in source and binary forms, with or
+ * without modification, are permitted provided that the following
+ * conditions are met:
+ *
+ * - Redistributions of source code must retain the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer.
+ *
+ * - Redistributions in binary form must reproduce the above
+ * copyright notice, this list of conditions and the following
+ * disclaimer in the documentation and/or other materials
+ * provided with the distribution.
+ *
+ * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
+ * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
+ * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
+ * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
+ * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
```

## [PATCH 4/6] iSER RDMA CM (CMA) and IB verbs interaction

```
+ * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
+ * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
+ * SOFTWARE.
+ *
+ * $Id: iser_verbs.c 7051 2006-05-10 12:29:11Z ogerlitz $
+ */
+#include <asm/io.h>
+#include <linux/kernel.h>
+#include <linux/module.h>
+#include <linux/smp_lock.h>
+#include <linux/delay.h>
+#include <linux/version.h>
+
+#include "iscsi_iser.h"
+
+#define ISCSI_ISER_MAX_CONN 8
+#define ISER_MAX_CQ_LEN ((ISER_QP_MAX_RECV_DTOS + \
+ ISER_QP_MAX_REQ_DTOS) * \
+ ISCSI_ISER_MAX_CONN)
+
+static void iser_cq_tasklet_fn(unsigned long data);
+static void iser_cq+ enum iser_ib_conn_state exch)
+{
+ int ret;
+
+ spin_lock_bh(&ib_conn->lock);
+ if ((ret = (ib_conn->state == comp)))
+ ib_conn->state = exch;
+ spin_unlock_bh(&ib_conn->lock);
+ return ret;
+}
+
+/**
+ * triggers start of the disconnect procedures and wait for them to be done
+ */
+void iser_conn_terminate(struct iser_conn *ib_conn)
+{
+ int err = 0;
+
+ /* change the ib conn state only if the conn is UP, however always call
+ * rdma_disconnect since this is the only way to cause the CMA to change
+ * the QP state to ERROR
+ */
+
+ iser_conn_state_comp_exch(ib_conn, ISER_CONN_UP, ISER_CONN_TERMINATING);
+ err = rdma_disconnect(ib_conn->cma_id);
+ if (err)
+ iser_err("Failed to disconnect, conn: 0x%p err %d\n",
+ ib_conn, err);
+
+ wait_event_interruptible(ib_conn->wait,
```

## [PATCH 4/6] iSER RDMA CM (CMA) and IB verbs interaction

```
+ ib_conn->state == ISER_CONN_DOWN);
+
+ iser_conn_release(ib_conn);
+}
+
+static void iser_connect_error(struct rdma_cm_id *cma_id)
+{
+ struct iser_conn *ib_conn;
+ ib_conn = (struct iser_conn *)cma_id->context;
+
+ ib_conn->state = ISER_CONN_DOWN;
+ wake_up_interruptible(&ib_conn->wait);
+}
+
+static void iser_addr_handler(struct rdma_cm_id *cma_id)
+{
+ struct iser_device *device;
+ struct iser_conn *ib_conn;
+ int ret;
+
+ device = iser_device_find_by_ib_device(cma_id);
+ ib_conn = (struct iser_conn *)cma_id->context;
+ ib_conn->device = device;
+
+ ret = rdma_resolve_route(cma_id, 1000);
+ if (ret) {
+ iser_err("resolve route failed: %d\n", ret);
+ iser_connect_error(cma_id);
+ }
+ return;
+}
+
+static void iser_route_handler(struct rdma_cm_id *cma_id)
+{
+ struct rdma_conn_param conn_param;
+ int ret;
+
+ ret = iser_create_ib_conn_res((struct iser_conn *)cma_id->context);
+ if (ret)
+ goto failure;
+
+ iser_dbg("path.mtu is %d setting it to %d\n",
+ cma_id->route.path_rec->mtu, IB_MTU_1024);
+
+ /* we must set the MTU to 1024 as this is what the target is assuming */
+ if (cma_id->route.path_rec->mtu > IB_MTU_1024)
+ cma_id->route.path_rec->mtu = IB_MTU_1024;
+
+ memset(&conn_param, 0, sizeof conn_param);
+ conn_param.responder_resources = 4;
+ conn_param.initiator_depth = 1;
```

## [PATCH 4/6] iSER RDMA CM (CMA) and IB verbs interaction

```
+ conn_param.retry_count = 7;
+ conn_param.rnr_retry_count = 6;
+
+ ret = rdma_connect(cma_id, &conn_param);
+ if (ret) {
+ iser_err("failure connecting: %d\n", ret);
+ goto failure;
+ }
+
+ return;
+failure:
+ iser_connect_error(cma_id);
+}
+
+static void iser_connected_handler(struct rdma_cm_id *cma_id)
+{
+ struct iser_conn *ib_conn;
+
+ ib_conn = (struct iser_conn *)cma_id->context;
+ ib_conn->state = ISER_CONN_UP;
+ wake_up_interruptible(&ib_conn->wait);
+}
+
+static void iser_disconnected_handler(struct rdma_cm_id *cma_id)
+{
+ struct iser_conn *ib_conn;
+
+ ib_conn = (struct iser_conn *)cma_id->context;
+ ib_conn->disc_evt_flag = 1;
+
+ /* getting here when the state is UP means that the conn is being *
+ * terminated asynchronously from the iSCSI layer's perspective. */
+ if (iser_conn_state_comp_exch(ib_conn, ISER_CONN_UP,
+ ISER_CONN_TERMINATING))
+ iscsi_conn_failure(ib_conn->iser_conn->iscsi_conn,
+ ISCSI_ERR_CONN_FAILED);
+
+ /* Complete the termination process if no posts are pending */
+ if ((atomic_read(&ib_conn->post_rcv_buf_count) == 0) &&
+ (atomic_read(&ib_conn->post_send_buf_count) == 0)) {
+ ib_conn->state = ISER_CONN_DOWN;
+ wake_up_interruptible(&ib_conn->wait);
+ }
+}
+
+static int iser_cma_handler(struct rd
```