```
**********************************************************
   94820 Mon May  5 11:11:12 2014
new/usr/src/uts/common/io/comstar/port/iscsit/iscsit.c
4780 comstar iSCSI target shouldn't abuse ddi_get_time(9f)
Reviewed by: Eric Diven <eric.diven@delphix.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 *
 * Copyright 2014 Nexenta Systems, Inc. All rights reserved.
 * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
 */

#include <sys/cpuvar.h>
#include <sys/types.h>
#include <sys/conf.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/modctl.h>
#include <sys/sysmacros.h>
#include <sys/socket.h>
#include <sys/strsubr.h>
#include <sys/nvpair.h>

#include <sys/stmf.h>
#include <sys/stmf_ioctl.h>
#include <sys/portif.h>
#include <sys/idm/idm.h>
#include <sys/idm/idm_conn_sm.h>

#include "iscsit_isns.h"
#include "iscsit.h"

#define ISCSIT_VERSION          BUILD_DATE "-1.18dev"
#define ISCSIT_NAME_VERSION     "COMSTAR ISCSIT v" ISCSIT_VERSION

/*
 * DDI entry points.
 */
static int iscsit_drv_attach(dev_info_t *, ddi_attach_cmd_t);
static int iscsit_drv_detach(dev_info_t *, ddi_detach_cmd_t);
static int iscsit_drv_getinfo(dev_info_t *, ddi_info_cmd_t, void *, void **);
static int iscsit_drv_open(dev_t *, int, int, cred_t *);
```



```
static int iscsit_drv_close(dev_t, int, int, cred_t *);
static int iscsit_drv_busy(void);
static int iscsit_drv_ioctl(dev_t, int, intptr_t, int, cred_t *, int *);

extern struct mod_ops mod_miscops;


static struct cb_ops iscsit_cb_ops = {
        iscsit_drv_open,        /* cb_open */
        iscsit_drv_close,       /* cb_close */
        nodev,                  /* cb_strategy */
        nodev,                  /* cb_print */
        nodev,                  /* cb_dump */
        nodev,                  /* cb_read */
        nodev,                  /* cb_write */
        iscsit_drv_ioctl,       /* cb_ioctl */
        nodev,                  /* cb_devmap */
        nodev,                  /* cb_mmap */
        nodev,                  /* cb_segmap */
        nochpoll,               /* cb_chpoll */
        ddi_prop_op,            /* cb_prop_op */
        NULL,                   /* cb_streamtab */
        D_MP,                   /* cb_flag */
        CB_REV,                 /* cb_rev */
        nodev,                  /* cb_aread */
        nodev,                  /* cb_awrite */
};
```

*unchanged_portion_omitted_*

```
/*
 * iscsit_check_cmdsn_and_queue
 *
 * Independent of the order in which the iSCSI target receives non-immediate
 * command PDU across the entire session and any multiple connections within
 * the session, the target must deliver the commands to the SCSI layer in
 * CmdSN order. So out-of-order non-immediate commands are queued up on a
 * session-wide wait queue. Duplicate commands are ignored.
 *
 */
static int
iscsit_check_cmdsn_and_queue(idm_pdu_t *rx_pdu)
{
        idm_conn_t              *ic = rx_pdu->isp_ic;
        iscsit_conn_t           *ict = ic->ic_handle;
        iscsit_sess_t           *ist = ict->ict_sess;
        iscsi_scsi_cmd_hdr_t    *hdr = (iscsi_scsi_cmd_hdr_t *)rx_pdu->isp_hdr;

        mutex_enter(&ist->ist_sn_mutex);
        if (hdr->opcode & ISCSI_OP_IMMEDIATE) {
                /* do not queue, handle it immediately */
                DTRACE_PROBE2(immediate__cmd, iscsit_sess_t *, ist,
                    idm_pdu_t *, rx_pdu);
                mutex_exit(&ist->ist_sn_mutex);
                return (ISCSIT_CMDSN_EQ_EXPCMDSN);
        }
        if (iscsit_sna_lt(ist->ist_expcmdsn, ntohl(hdr->cmdsn))) {
                /*
                 * Out-of-order commands (cmdSN higher than ExpCmdSN)
                 * are staged on a fixed-size circular buffer until
                 * the missing command is delivered to the SCSI layer.
                 * Irrespective of the order of insertion into the
                 * staging queue, the commands are processed out of the
                 * queue in cmdSN order only.
                 */
                rx_pdu->isp_queue_time = gethrtime();
                rx_pdu->isp_queue_time = ddi_get_time();
```

```
3129                 iscsit_add_pdu_to_queue(ist, rx_pdu);
3130                 mutex_exit(&ist->ist_sn_mutex);
3131                 return (ISCSIT_CMDSN_GT_EXPCMDSN);
3132         } else if (iscsit_sna_lt(ntohl(hdr->cmdsn), ist->ist_expcmdsn)) {
3133                 DTRACE_PROBE3(cmdsn__lt__expcmdsn, iscsit_sess_t *, ist,
3134                     iscsit_conn_t *, ict, idm_pdu_t *, rx_pdu);
3135                 mutex_exit(&ist->ist_sn_mutex);
3136                 return (ISCSIT_CMDSN_LT_EXPCMDSN);
3137         } else {
3138                 mutex_exit(&ist->ist_sn_mutex);
3139                 return (ISCSIT_CMDSN_EQ_EXPCMDSN);
3140         }
3141 }
______unchanged_portion_omitted_

3356 static void
3357 iscsit_rxpdu_queue_monitor_session(iscsit_sess_t *ist)
3358 {
3359         iscsit_cbuf_t   *cbuf   = ist->ist_rxpdu_queue;
3360         idm_pdu_t       *next_pdu = NULL;
3361         uint32_t        index, next_cmdsn, i;

3363         /*
3364          * Assume that all PDUs in the staging queue have a cmdsn >= expcmdsn.
3365          * Starting with the expcmdsn, iterate over the staged PDUs to find
3366          * the next PDU with a wait time greater than the threshold. If found
3367          * advance the staged PDU to the SCSI layer, skipping over the missing
3368          * PDU(s) to get past the hole in the command sequence. It is up to
3369          * the initiator to note that the target has not acknowledged a cmdsn
3370          * and take appropriate action.
3371          *
3372          * Since the PDU(s) arrive in any random order, it is possible that
3373          * that the actual wait time for a particular PDU is much longer than
3374          * the defined threshold. e.g. Consider a case where commands are sent
3375          * over 4 different connections, and cmdsn = 1004 arrives first, then
3376          * 1003, and 1002 and 1001 are lost due to a connection failure.
3377          * So now 1003 is waiting for 1002 to be delivered, and although the
3378          * wait time of 1004 > wait time of 1003, only 1003 will be considered
3379          * by the monitor thread. 1004 will be automatically processed by
3380          * iscsit_process_pdu_in_queue() once the scan is complete and the
3381          * expcmdsn becomes current.
3382          */
3383         mutex_enter(&ist->ist_sn_mutex);
3384         cbuf = ist->ist_rxpdu_queue;
3385         if (cbuf->cb_num_elems == 0) {
3386                 mutex_exit(&ist->ist_sn_mutex);
3387                 return;
3388         }
3389         for (next_pdu = NULL, i = 0; ; i++) {
3390                 next_cmdsn = ist->ist_expcmdsn + i; /* start at expcmdsn */
3391                 index = next_cmdsn % ISCSIT_RXPDU_QUEUE_LEN;
3392                 if ((next_pdu = cbuf->cb_buffer[index]) != NULL) {
3393                         /*
3394                          * If the PDU wait time has not exceeded threshold
3395                          * stop scanning the staging queue until the timer
3396                          * fires again
3397                          */
3398                         if ((gethrtime() - next_pdu->isp_queue_time)
3399                             < (rxpdu_queue_threshold * NANOSEC)) {
3398                         if ((ddi_get_time() - next_pdu->isp_queue_time)
3399                             < rxpdu_queue_threshold) {
3400                                 mutex_exit(&ist->ist_sn_mutex);
3401                                 return;
3402                         }
3403                         /*
3404                          * Remove the next PDU from the queue and post it
```



```
3405                          * to the SCSI layer, skipping over the missing
3406                          * PDU. Stop scanning the staging queue until
3407                          * the monitor timer fires again
3408                          */
3409                         (void) iscsit_remove_pdu_from_queue(ist, next_cmdsn);
3410                         mutex_exit(&ist->ist_sn_mutex);
3411                         DTRACE_PROBE3(advanced__to__blocked__cmdsn,
3412                             iscsit_sess_t *, ist, idm_pdu_t *, next_pdu,
3413                             uint32_t, next_cmdsn);
3414                         iscsit_post_staged_pdu(next_pdu);
3415                         /* Deliver any subsequent PDUs immediately */
3416                         iscsit_process_pdu_in_queue(ist);
3417                         return;
3418                 }
3419                 /*
3420                  * Skipping over i PDUs, e.g. a case where commands 1001 and
3421                  * 1002 are lost in the network, skip over both and post 1003
3422                  * expcmdsn then becomes 1004 at the end of the scan.
3423                  */
3424                 DTRACE_PROBE2(skipping__over__cmdsn, iscsit_sess_t *, ist,
3425                     uint32_t, next_cmdsn);
3426         }
3427         /*
3428          * following the assumption, staged cmdsn >= expcmdsn, this statement
3429          * is never reached.
3430          */
3431 }
______unchanged_portion_omitted_
```

```
**********************************************************
   14701 Mon May  5 11:11:12 2014
new/usr/src/uts/common/sys/idm/idm_impl.h
4780 comstar iSCSI target shouldn't abuse ddi_get_time(9f)
Reviewed by: Eric Diven <eric.diven@delphix.com>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright 2014 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */

#ifndef _IDM_IMPL_H_
#define _IDM_IMPL_H_

#ifdef  __cplusplus
extern "C" {
#endif

#include <sys/avl.h>
#include <sys/socket_impl.h>
#include <sys/taskq_impl.h>

/*
 * IDM lock order:
 *
 * idm_taskid_table_lock, idm_task_t.idt_mutex
 */

#define CF_LOGIN_READY          0x00000001
#define CF_INITIAL_LOGIN        0x00000002
#define CF_ERROR                0x80000000

typedef enum {
        CONN_TYPE_INI = 1,
        CONN_TYPE_TGT
} idm_conn_type_t;
_____unchanged_portion_omitted_

#define PDU_MAX_IOVLEN  12
#define IDM_PDU_MAGIC   0x49504455      /* "IPDU" */

typedef struct idm_pdu_s {
```



```
        uint32_t        isp_magic;      /* "IPDU" */

        /*
         * Internal - Order is vital.  idm_tx_link *must* be the second
         * element in this structure for proper TX PDU ordering.
         */
        list_node_t     idm_tx_link;

        list_node_t     isp_client_lnd;

        idm_conn_t      *isp_ic;        /* Must be set */
        iscsi_hdr_t     *isp_hdr;
        uint_t          isp_hdrlen;
        uint8_t         *isp_data;
        uint_t          isp_datalen;

        /* Transport header */
        void            *isp_transport_hdr;
        uint32_t        isp_transport_hdrlen;
        void            *isp_transport_private;

        /*
         * isp_data is used for sending SCSI status, NOP, text, scsi and
         * non-scsi data. Data is received using isp_iov and isp_iovlen
         * to support data over multiple buffers.
         */
        void            *isp_private;
        idm_pdu_cb_t    *isp_callback;
        idm_status_t    isp_status;

        /*
         * The following four elements are only used in
         * idm_sorecv_scsidata() currently.
         */
        struct iovec    isp_iov[PDU_MAX_IOVLEN];
        int             isp_iovlen;
        idm_buf_t       *isp_sorx_buf;

        /* Implementation data for idm_pdu_alloc and sorx PDU cache */
        uint32_t        isp_flags;
        uint_t          isp_hdrbuflen;
        uint_t          isp_databuflen;
        hrtime_t        isp_queue_time;
        time_t          isp_queue_time;

        /* Taskq dispatching state for deferred PDU */
        taskq_ent_t     isp_tqent;
} idm_pdu_t;
_____unchanged_portion_omitted_
```