```
**********************************************************
    8567 Mon Jan 14 13:12:18 2019
new/usr/src/uts/intel/io/intel_nhm/dimm_topo.c
10082 intel_nhm is checking for NULL arrays
**********************************************************
/*
 * 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 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

#include <sys/types.h>
#include <sys/time.h>
#include <sys/nvpair.h>
#include <sys/cmn_err.h>
#include <sys/cred.h>
#include <sys/open.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/conf.h>
#include <sys/modctl.h>
#include <sys/cyclic.h>
#include <sys/errorq.h>
#include <sys/stat.h>
#include <sys/cpuvar.h>
#include <sys/mc_intel.h>
#include <sys/mc.h>
#include <sys/fm/protocol.h>
#include "nhm_log.h"
#include "intel_nhm.h"

extern nvlist_t *inhm_mc_nvl[MAX_CPU_NODES];
extern char closed_page;
extern char ecc_enabled;
extern char lockstep[MAX_CPU_NODES];
extern char mirror_mode[MAX_CPU_NODES];
extern char spare_channel[MAX_CPU_NODES];

static void
inhm_vrank(nvlist_t *vrank, int num, uint64_t dimm_base, uint64_t limit,
    uint32_t sinterleave, uint32_t cinterleave, uint32_t rinterleave,
    uint32_t sway, uint32_t cway, uint32_t rway)
```

```
{
        char buf[128];

        (void) snprintf(buf, sizeof (buf), "dimm-rank-base-%d", num);
        (void) nvlist_add_uint64(vrank, buf, dimm_base);
        (void) snprintf(buf, sizeof (buf), "dimm-rank-limit-%d", num);
        (void) nvlist_add_uint64(vrank, buf, dimm_base + limit);
        if (sinterleave > 1) {
                (void) snprintf(buf, sizeof (buf), "dimm-socket-interleave-%d",
                    num);
                (void) nvlist_add_uint32(vrank, buf, sinterleave);
                (void) snprintf(buf, sizeof (buf),
                    "dimm-socket-interleave-way-%d", num);
                (void) nvlist_add_uint32(vrank, buf, sway);
        }
        if (cinterleave > 1) {
                (void) snprintf(buf, sizeof (buf), "dimm-channel-interleave-%d",
                    num);
                (void) nvlist_add_uint32(vrank, buf, cinterleave);
                (void) snprintf(buf, sizeof (buf),
                    "dimm-channel-interleave-way-%d", num);
                (void) nvlist_add_uint32(vrank, buf, cway);
        }
        if (rinterleave > 1) {
                (void) snprintf(buf, sizeof (buf), "dimm-rank-interleave-%d",
                    num);
                (void) nvlist_add_uint32(vrank, buf, rinterleave);
                (void) snprintf(buf, sizeof (buf),
                    "dimm-rank-interleave-way-%d", num);
                (void) nvlist_add_uint32(vrank, buf, rway);
        }
}
______unchanged_portion_omitted_

static nvlist_t *
inhm_dimm(nhm_dimm_t *nhm_dimm, uint32_t node, uint8_t channel, uint32_t dimm)
{
        nvlist_t *newdimm;
        uint8_t t;
        char sbuf[65];

        (void) nvlist_alloc(&newdimm, NV_UNIQUE_NAME, KM_SLEEP);
        (void) nvlist_add_uint32(newdimm, "dimm-number", dimm);

        if (nhm_dimm->dimm_size >= 1024*1024*1024) {
                (void) snprintf(sbuf, sizeof (sbuf), "%dG",
                    (int)(nhm_dimm->dimm_size / (1024*1024*1024)));
        } else {
                (void) snprintf(sbuf, sizeof (sbuf), "%dM",
                    (int)(nhm_dimm->dimm_size / (1024*1024)));
        }
        (void) nvlist_add_string(newdimm, "dimm-size", sbuf);
        (void) nvlist_add_uint64(newdimm, "size", nhm_dimm->dimm_size);
        (void) nvlist_add_uint32(newdimm, "nbanks", (uint32_t)nhm_dimm->nbanks);
        (void) nvlist_add_uint32(newdimm, "ncolumn",
            (uint32_t)nhm_dimm->ncolumn);
        (void) nvlist_add_uint32(newdimm, "nrow", (uint32_t)nhm_dimm->nrow);
        (void) nvlist_add_uint32(newdimm, "width", (uint32_t)nhm_dimm->width);
        (void) nvlist_add_uint32(newdimm, "ranks", (uint32_t)nhm_dimm->nranks);
        inhm_rank(newdimm, nhm_dimm, node, channel, dimm,
            nhm_dimm->dimm_size / nhm_dimm->nranks);
        if (nhm_dimm->manufacturer[0]) {
        if (nhm_dimm->manufacturer && nhm_dimm->manufacturer[0]) {
                t = sizeof (nhm_dimm->manufacturer);
                (void) strncpy(sbuf, nhm_dimm->manufacturer, t);
                sbuf[t] = 0;
                (void) nvlist_add_string(newdimm, "manufacturer", sbuf);
        }
        if (nhm_dimm->serial_number[0]) {
        if (nhm_dimm->serial_number && nhm_dimm->serial_number[0]) {
                t = sizeof (nhm_dimm->serial_number);
                (void) strncpy(sbuf, nhm_dimm->serial_number, t);
                sbuf[t] = 0;
                (void) nvlist_add_string(newdimm, FM_FMRI_HC_SERIAL_ID, sbuf);
        }
        if (nhm_dimm->part_number[0]) {
        if (nhm_dimm->part_number && nhm_dimm->part_number[0]) {
                t = sizeof (nhm_dimm->part_number);
                (void) strncpy(sbuf, nhm_dimm->part_number, t);
                sbuf[t] = 0;
                (void) nvlist_add_string(newdimm, FM_FMRI_HC_PART, sbuf);
        }
        if (nhm_dimm->revision[0]) {
        if (nhm_dimm->revision && nhm_dimm->revision[0]) {
                t = sizeof (nhm_dimm->revision);
                (void) strncpy(sbuf, nhm_dimm->revision, t);
                sbuf[t] = 0;
                (void) nvlist_add_string(newdimm, FM_FMRI_HC_REVISION, sbuf);
        }
        t = sizeof (nhm_dimm->label);
        (void) strncpy(sbuf, nhm_dimm->label, t);
        sbuf[t] = 0;
        (void) nvlist_add_string(newdimm, FM_FAULT_FRU_LABEL, sbuf);
        return (newdimm);
}
```

______***unchanged_portion_omitted***_