Cray® XC40™ System Diagnosability: Functionality, Performance, and Lessons Learned

Jeffrey J. Schutkoske
Platform Services Group (PSG)
Cray, Inc.
Bloomington, MN, USA
jjs@cray.com

Abstract*— The Intel® Xeon Phi™ CPU 7250 processor presents new opportunities for diagnosing the nodes based on this processor in the Cray® XC™ system. This processor supports a new high-bandwidth on-package MCDRAM memory and related interfaces. It also provides the ability to support different Non-Uniform Memory Access (NUMA) configurations. The new Cray Processor Daughter Card (PDC) also supports an optional PCIe SSD card. This processor requires new BIOS, administrative commands, power and thermal limits, as well as new diagnostics to validate functionality and performance. This paper describes the diagnostic tool chain changes required to support this processor and PCIe SSD card. It describes the functionality, performance, and lessons learned from diagnosing problems at scale. It also provides detailed examples on how to diagnose node faults within the Cray® XC40™ system.

Keywords: Cray® XC40™ system, Cray Linux Environment, CLE, Intel® Xeon Phi™ CPU 7250 processor, MCDRAM, diagnostic, diagnosability

I. Introduction

The Cray® XC40™ system includes the new Cray Processor Daughter Card (PDC) which supports the new Intel® Xeon Phi™ CPU 7250 processor and an optional PCIe SSD card. Previous work [1] [2] [3] has outlined Cray system diagnosability for the Cray® XC™ Series. There are a number of enhancements required to support the new node processor and optional SSD card in the Cray® XC40™ system.

The Cray Hardware Supervisory System (HSS) supports the new high-bandwidth on-package MCDRAM memory and interfaces. It supports the On-Demand configuration of the MCDRAM and NUMA. The MCDRAM and NUMA configurations, as well as SSD enable/disable, are configurable from both the command line on the System Management Workstation (SMW) and by a workload manager (WLM) running under the Cray Linux Environment (CLE) utilizing the Cray Advanced Platform Monitoring and Control (CAPMC) interface.

The HSS commands, utilities, and diagnostics are used to diagnose a faulty node. Initially BIOS is used to verify that the node has powered on correctly and that the node is initialized and all interface ports are trained successfully. It also validates that any on-demand configuration settings are valid. Any errors or warnings are reported to HSS.

Once the node is successfully initialized by BIOS, the CLE is booted. Cray Linux Environment (CLE) utilizes the Resiliency Communication Agent (RCA) interface to communicate between HSS and nodes. The Hardware Error Log channel is used to send hardware errors to HSS.

HSS monitors a number of power and thermal sensors within the Cray® XC40™ system at the blade and cabinet level. These sensors and devices are managed via the out-of-band paths on the blade by HSS.

HSS also utilizes a number of Intel Reliability, Availability, and Service (RAS) features including Platform Environment Control Interface (PECI) and In-Target Probe (ITP).

New Intel® Xeon Phi™ CPU 7250 processor on-line diagnostics have been written to validate the node and MCDRAM functionality and performance. The diagnostics validate the node based on the MCDRAM and NUMA configurations. The Workload Test Suite (WTS) has also been updated to detect and diagnose Intel® Xeon Phi™ CPU 7250 processor problems under CLE.

There are new utilities and diagnostics to support the PCIe SSD card. There is a new diagnostic utility that executes in CLE on the Data Warp Node or the compute node to support the SSD. This diagnostic utility is periodically scheduled to check the health of the SSD. It reports the status to the SMW via RCA. The system administrator can query and display the current SSD health, as well as historical data. The results of the SSD diagnostic utility can also be viewed on the SMW.

This paper describes the tool chain changes required to support the new blade with the Intel® Xeon Phi™ CPU 7250 processor and optional PCIe SSD cards. It also provides detailed examples on how to diagnose Intel® Xeon Phi™ CPU 7250 processor faults within the Cray® XC40™ system. Section II describes the on-demand configuration options for the node and the attached SSD cards. BIOS initialization is covered in Section III. The Resiliency Communication Agent (RCA) interface and monitoring are cover in Section IV and Section V respectively. Section VI covers the HSS Out-Of-Band (OOB) debug, Section VII covers the online diagnostics, and Section VIII explains the Workload Test Suite (WTS). Section IX describes the PCIe SSD cards. Related work is described in Section X. Finally, Section XI provides a summary.

II. ON-DEMAND CONFIGURATION

The Intel® Xeon Phi™ CPU 7250 processor can be reconfigured on-demand using the Cray Advanced Platform Monitoring and Control (CAPMC) interface [11]. The new configuration is applied on the next cold boot of the node or nodes. This processor supports five different NUMA modes including All-to-All, Sub NUMA Cluster 2, Sub NUMA Cluster 4, Hemisphere, and Quadrant. It also supports 5 different MCDRAM modes including Cache, Flat, Hybrid Equal, and Hybrid Split. All the permutations and combinations of these NUMA modes and MCDRAM modes are supported. Note that not all modes and combinations have proven effective, but this discussion is outside the scope of this paper. [21] The PCIe SSD cards can be enabled or disabled via the CAPMC interface.

CAPMC provides a published interface [12], shown in Table I, that allows the WLM to make reconfiguration requests to HSS as shown in the table below. System administrators and operators can also use the CAPMC interface on the SMW. The changed settings are stored in the power management database (PMDB). The changes are applied on the next reboot. Typically, the WLM requests the reboot via the CAPMC *node_reinit* command.

TABLE I. ÇRAY CAPMC CONFIGURATION REQUESTS

Command	**Description**
get_ssd_enable	Return list of enabled SSD cards
set_ssd_enable	Enable/disable SSD
get_mcdram_cfg_capabilities	Return supported MCDRAM configurations
get_mcdram_cfg	Return current MCDRAM configuration
set_mcdram_cfg	Set new MCDRAM configuration
get_numa_cfg_capabilities	Return supported NUMA configurations
get_numa_cfg	Return current NUMA configuration
set_numa_cfg	Set new NUMA configuration
node_reinit	Reinitialize node
node_status	Returns current node status

Once HSS receives the CAPMC *node_reinit* request, the node is stepped through the standard states of ready, halt, on, standby, and returning to the ready state. Any node that fails while in a state remains in that state. For example, if a node fails to complete the CLE boot, the node persists in the standby state. The WLM can check the current state of the node using the CAPMC *node_status* command.

Failures that occur during a reboot initiated by CAPMC are logged in /var/opt/cray/log/xtremoted-YYYYMMDD. *xtremoted* logs the full *xtbounce* output when *xtbounce* returns non-zero. *xtremoted* logs the full *xtcli* boot output regardless of return code.

III. BIOS INITIALIZATION

The Cray BIOS is used to initialize and configure the Intel® Xeon Phi™ CPU 7250 processor node as requested by CAMPC. When HSS brings the node out of reset, BIOS is loaded and executed. HSS passes the new configuration settings to BIOS, which uses them to configure the NUMA and MCDRAM modes as requested.

BIOS is used to initialize the Intel® Xeon Phi™ CPU 7250 processor, initialize and train the attached memory and MCDRAM, and initialize and train any attached PCIe devices including the Aries network interface (NIC).

BIOS executes the Rank Margining Tool (RMT) as part of the normal BIOS process. The RMT generates, calculates, and reports out the system memory interface cold boot signal timing and voltage reference (VREF) margins. This ensures that the DDR4 and MCDRAM interfaces can support expected memory performance. If BIOS determines that RMT was successful on the previous cold boot, BIOS skips the RMT thereby enabling a cold fast boot. RMT is run by default after each BIOS flash.

The Cray BIOS reports any MCDRAM or DIMM failures during memory training including any Machine Check Architecture (MCA) errors that were detected. NUMA and MCDRAM configurations are verified and reported.

BIOS detected errors or failures are logged to the BIOS logs. All errors are logged to the BIOS log on the Blade Controller (BC). Controller log forwarding with the Lightweight Log Manager (LLM) can be configured to forward the BIOS logs to the SMW. The Blade and Cabinet Controllers are diskless Linux systems, so log forwarding is enabled by default.

```
CrayBdsHook.Entry(B9E00300)
AddLpcFaultEntry: PostCode 0x68
LPC_SCRATCH_FAULT_REPORT_ENTRY
FaultNum:  1
Type:     11
Flags:    0x00
CodeMajor: 0x68
CodeMinor: 0x00
ApicId:   0x00
CpuNum:   0
Timestamp: 04/05/2017  18:18:20
LogData:  0x00000020
FaultMsg:  BIST disabled via POC_RESET_STRAPS
LPC_SCRATCH_FAULT_INFO_ENTRY
LineNum:  1085
FileName: z:\knl\CrayPkg\Dxe\CrayBdsHook\CrayBdsHook.c
```

Figure 1. BIOS Hardware Failure Detected

In Figure 1. BIOS Hardware Failure Detected BIOS requested that the Intel® Xeon Phi™ CPU 7250 processor BIST execute and BIOS detected that it did not execute. In Figure 2. Figure 2. BIOS MCDRAM Failure Detected BIOS detected multiple MCDRAM failures where the MCDRAM did not train successfully and it reported the errors.

```
** EDC-0 Memory Init: cmdcrc_err = 1
** EDC-4 Memory Init: cmdcrc_err = 1

EDC Meminit Time Elapsed: 99ms
EDC-0: memory Init Status 0x0003

LPC_SCRATCH_FAULT_REPORT_ENTRY
    FaultNum:  1
    Type:      9
    Flags:     0x00
    CodeMajor: 0xA1
    CodeMinor: 0x06
    ApicId:    0x00
    CpuNum:    0
    Timestamp: 07/29/2015  22:29:22
    LogData:   0x0000FFFF
    FaultMsg:  CRAY_MCDRAM_WARNING

A warning has been logged! Warning Code = 0xA1, Minor Warning Code =
0x6, Data = 0xFFFF

S0 Ch0

EDC-3: memory Init Status 0x0001
EDC-4: memory Init Status 0x0003

LPC_SCRATCH_FAULT_REPORT_ENTRY
    FaultNum:  2
    Type:      9
    Flags:     0x00
    CodeMajor: 0xA1
    CodeMinor: 0x06
    ApicId:    0x00
    CpuNum:    0
    Timestamp: 07/29/2015  22:29:22
    LogData:   0x0004FFFF
    FaultMsg:  CRAY_MCDRAM_WARNING

A warning has been logged! Warning Code = 0xA1, Minor Warning Code =
0x6, Data = 0x4FFFF
```

Figure 2. BIOS MCDRAM Failure Detected

Any errors are logged and reported to the SMW command, *xtbounce*, as part of the system boot sequence.

The node is prevented from booting if any hardware failure is detected during BIOS execution. The Cray BIOS logs are copied to the Cray SMW on any BIOS detected error or failure for further analysis. This ensures that at the time of system boot all devices are properly discovered, initialized, and trained successfully or flagged as failed, without stopping the boot process.

IV. Resiliancy Communication Agent (RCA)

The CLE kernel captures node hardware errors. It reads MCA errors (correctable and uncorrectable) and logs them in the node console log, which is saved on the SMW. It also reports them via the Hardware Error Log Channel connected from the node to the BC, which forwards them to the SMW Hardware Error Logger Daemon (*xthwerrlogd*). The SMW command, *xthwerrlog*, displays all hardware errors logged via the Hardware Error Channel including MCA errors as shown in Figure 3. HSS provides the decoding for all Intel® Xeon Phi™ CPU 7250 processor machine check registers. This includes banks 7-14 where MCDRAM errors are reported.

```
HWERR[c1-0c2s14n1]:0xfd0b:
    Uncorrectable:MFG[0]:CPUID[50671]:SOCKET[0]:APIC[0]:
    BANK[11]:STATUS[0xf60000800040009e]:MISC[0x0]:
    ADDR[0x153fffc300]:CTL2[0x0]
HWERR[c1-0c1s6n0]:0xfd07:
    Uncorrectable:MFG[0]:CPUID[50671]:SOCKET[0]:APIC[0]:
    BANK[7]:STATUS[0xf60000800040009e]:MISC[0x0]:
    ADDR[0x176477c000]:CTL2[0x0]
HWERR[c1-0c1s3n2]:0xfd09:
    Uncorrectable:MFG[0]:CPUID[50671]:SOCKET[0]:APIC[0]:
    BANK[9]:STATUS[0xf60001000040009e]:MISC[0x0]:
    ADDR[0x17647f8000]:CTL2[0x0]
HWERR[c0-0c0s11n2]:0xfd0d:
    Uncorrectable:MFG[0]:CPUID[50671]:SOCKET[0]:APIC[0]:
    BANK[13]:STATUS[0xf60000400040009e]:MISC[0x0]:
    ADDR[0x17647a2000]:CTL2[0x0]
```

Figure 3. xthwerrlog Output

The SMW command, *xtmcadecode*, decodes and provides detailed explanation for Intel MCA errors as shown in Figure 4.

```
xtmcadecode -t knl 16 84000040000800C0

Bank 16: IMC1: Integrated Memory Controller 1:
MCA Status = 0x84000040000800c0:

MCACOD = 0x00c0, MSCOD = 0x0008
Other Info = 0x00
Corrected Error Count = 1
Threshold-based Error Status = 0, No Tracking

Common Status Info:
    VALID = 1 = Valid Error Detected
    OVER  = 0 = No overflow
    UC    = 0 = Error Corrected by HW
    EN    = 0 =
    MISCV = 0 =
    ADDRV = 1 = Error address in MCi_ADDR
    PCC   = 0 =
    S     = 0 =
    AR    = 0 =

Model-specific error: Correctable Patrol Scrub
    Channel: 0
    MCA: Undefined Error
```

Figure 4. xtmcadecode Output

Advanced Error Reporting (AER) is enabled in the CLE kernel by default. With AER enabled the PCIe SSD card errors are reported to the SMW. The Compute Node Linux (CNL) kernel reads these PCIe errors and logs them in the node console log, which is saved on the SMW. It also reports them via the Hardware Error Log Channel connected from the node to the BC, which forwards them to the SMW Hardware Error Logger Daemon (*xthwerrlogd*). The PCIe errors are viewable using the SMW command, *xtpcimon*.

V. MONITORING

The Cray Processor Daughter Card (PDC) has 26 sensors available for monitoring node power and memory power. [14] The sensors are connected to seven separate I^2C buses. This topology allows the node power readings to be scanned both at a higher rate and in greater detail than was possible in previous designs.

The HSS utility System Environment Data Collections (SEDC) [10] monitors the system health and records the environmental data coming from the system's hardware components. The telemetry data is logged to the PMDB.

To validate the HSS hardware and software, the SMW HSS debug utility, *xtcheckhss*, is used. The SMW command validates the Cray® XC40™ system HSS infrastructure by checking each blade and each cabinet. It can be used to get a quick validation that the HSS infrastructure is functioning normally and can also be used to troubleshoot a blade or a given cabinet. On a blade, it validates the basic blade functionality. It can also validate HSS voltages, Aries voltages and current, temperatures, PDC sensors, as well as the Intel processor and DIMM temperatures and voltages.

The SMW command SEDC command, *xtgetsedcvalues*, returns the available SEDC values.

To view the System Environment Data Collections (SEDC) temperature telemetry data collected for the node as stored in the PMDB, use the database query as follows:

SELECT value FROM pmdb.bc_sedc_data WHERE bc_sedc_data.id where (sensor_id >= 1300 and sensor_id <= 1306)

The results of the above query are shown in TABLE II.

TABLE II. NODE TEMPERATURE OUTPUT

SEDC PMDB Temperature Data			
Node	**Sensor ID**	**Sensor Name**	**Value**
c1-0c0s10	1306	BC_T_NODE3_CPU0_TEMP	36
c1-0c0s10	1304	BC_T_NODE2_CPU0_TEMP	34
c1-0c0s10	1302	BC_T_NODE1_CPU0_TEMP	36
c1-0c0s10	1300	BC_T_NODE0_CPU0_TEMP	38
c1-0c0s14	1306	BC_T_NODE3_CPU0_TEMP	27
c1-0c0s14	1304	BC_T_NODE2_CPU0_TEMP	25
c1-0c0s14	1302	BC_T_NODE1_CPU0_TEMP	29
c1-0c0s14	1300	BC_T_NODE0_CPU0_TEMP	27

To view the SEDC processor global power telemetry data collected for the node as stored in the PMDB, use the database query as follows:

SELECT value FROM pmdb.bc_sedc_data WHERE bc_sedc_data.id where (sensor_id >= 2776 and sensor_id <= 2779)

The results of the above query are shown in TABLE III. Node Global Power Usage.

TABLE III. NODE GLOBAL POWER USAGE

SEDC PMDB Temperature Data			
Node	**Sensor ID**	**Sensor Name**	**Value**
c1-0c0s10	2776	BC_P_NODE0_GLOBAL_PROC_POWER	23
c1-0c0s10	2777	BC_P_NODE1_GLOBAL_PROC_POWER	22
c1-0c0s10	2778	BC_P_NODE2_GLOBAL_PROC_POWER	23
c1-0c0s10	2779	BC_P_NODE3_GLOBAL_PROC_POWER	23
c1-0c0s14	2776	BC_P_NODE0_GLOBAL_PROC_POWER	25
c1-0c0s14	2777	BC_P_NODE1_GLOBAL_PROC_POWER	25
c1-0c0s14	2778	BC_P_NODE2_GLOBAL_PROC_POWER	25
c1-0c0s14	2779	BC_P_NODE3_GLOBAL_PROC_POWER	27

It is also possible to search for a specific window of time and for specific nodes. This data is invaluable when troubleshooting hardware or software failures.

VI. HSS OUT-OF-BAND (OOB) DEBUG

The HSS system management platform performs monitoring and management out-of-band to the compute node and high-speed network. HSS also implements interfaces to various vendors Out-Of-Band interfaces to enhance OOB debug of the Cray® XC40™ system. These interfaces are as follows:

- Intel Platform Environment Control Interface (PECI)
- Intel In-Target Probe (ITP)

A. *Intel Platform Environment Control Interface (PECI) Debug*

Intel processors provide a single wire signal for their Platform Environment Control Interface (PECI). During some early debug, Cray uses PECI to dump out various Intel® Xeon Phi™ CPU 7250 processor registers to aid in debugging when a node failed and no other dump mechanism is available.

Intel uses the Intelligent Platform Management Interface (IPMI) Specification [1] as the protocol to connect to the Intel Management Engine (ME) within the Intel Wellsburg Platform Controller Hub (PCH). This interface is primarily used to collect power, thermal, and status information by the HSS utility SEDC, which monitors the system health and records the environmental data coming from the system's hardware components. And finally, the data can be viewed real-time using the HSS debug utility, *xtcheckhss*.

To display the various HSS telemetry data for a select blade in the system use the *xtcheckhss* command line as follows:

xtcheckhss --cclist=none --bclist=c0-0c2s0 --detail=f

This command outputs the details of all the components that are part of the blade. The partial output shown in Figure 5. xtcheckhss Blade Output focuses on the DIMM temperature, node power, and node temperature readings for two of the nodes on the blade.

```
c0-0c2s0n0   kpdc1_n1_s0
      dimm0_temp            n/a      n/a   n/a   30        degc     n/a   n/a
c0-0c2s0n0   kpdc1_n1_s0
       dimm1_temp            n/a      n/a   n/a   29        degc     n/a   n/a
c0-0c2s0n0   kpdc1_n1_s0
      dimm2_temp            n/a      n/a   n/a   29        degc     n/a   n/a
c0-0c2s0n0   kpdc1_n1_s0
      dimm3_temp            n/a      n/a   n/a   30        degc     n/a   n/a
c0-0c2s0n0   kpdc1_n1_s0
      knl_power           n/a      n/a   n/a   83252     w*1000    n/a   n/a
c0-0c2s0n0   kpdc1_n1_s0
      knl_temp            n/a      n/a   n/a   38        n/a      n/a   n/a


c0-0c2s0n1   kpdc0_n1_s0
      dimm0_temp            n/a      n/a   n/a   25        degc     n/a   n/a
c0-0c2s0n1   kpdc0_n1_s0
      dimm1_temp            n/a      n/a   n/a   25        degc     n/a   n/a
c0-0c2s0n1   kpdc0_n1_s0
      dimm2_temp            n/a      n/a   n/a   25        degc     n/a   n/a
c0-0c2s0n1   kpdc0_n1_s0
      dimm3_temp            n/a      n/a   n/a   25        degc     n/a   n/a
c0-0c2s0n1   kpdc0_n1_s0
      knl_power           n/a      n/a   n/a   29000     w*1000    n/a   n/a
c0-0c2s0n1   kpdc0_n1_s0
      knl_temp            n/a      n/a   n/a   26        n/a      n/a   n/a
```

Figure 5.   xtcheckhss Blade Output

The detailed output definitions for *xtcheckhss* are described in TABLE IV.

TABLE IV.   XTCHEXHSS DETAIL DESCRIPTIONS

xtcheckhss Detail Column Descriptions	
Name	**Description**
Component	Cray component name (cname)
Sensor	Hardware sensor name
FRSH	Freshness Counter
HLMIN	Hardware Limit Minimum (cannot be modified)
SLMIN	Software Limit Minimum (software configurable)
DATA	Data value
UNIT	Data units (i.e. Degrees Celsius, Watts, etc.)
SLMAX	Software Limit Maximum (software configurable)
HLMAX	Hardware Limit Maximum (cannot be modified)

xtcheckhss reports the component, sensor, data, and unit for all detailed telemetry data. There are a number of HSS devices where the additional fields become important. In the case below where a Voltage Regulator Module (VRM) is reporting a low voltage reading, the VRM is below the SLMIN but higher than the HLMIN where a VRM fault would occur. When any device reading is outside of these limits, a message is sent to the SMW. *xtcheckhss* is used to verify that the device reading is out of the expected range as shown in Figure 6.

```
c0-0c0s7n2   qpdc0_n0_s0_mem_vrm          vdd_vdr01_s0_c_i
             1200        1350        1339        v*1000
             1650        1800
```

Figure 6.   HSS Low Voltage Output

B. *Intel In-Target Probe (ITP) Debug*

The Intel In-Target Probe (ITP) [2] is a JTAG bus with some Intel-specific signals and protocol added. Typically, this is done with a physical Intel ITP interface connected to the processor via the eXtended Debug Port (XDP). Cray has implemented an embedded ITP so that no external hardware needs to be connected to the Cray® XC40™ system. The embedded ITP is used as a processor hardware debug tool by launching and executing scripts to debug the node. These scripts reside on the SMW and are executed via the SMW command *xtitp*.

Many of the scripts provide useful hardware debug information about the PCIe configuration and status, processor information, MCA errors, and model specific register (MSR) data.

To query for any machine check errors in Node 0 use the *xtitp* command as follows:

```
xtitp -t c3-0c2s15 mca-error-check-all 0
```

Sometimes a node can crash due to an MCDRAM error. In this case *xtitp* can be used to pull the MCDRAM error as shown in Figure 7.

```
xtitp -t c0-0c0s13 mca-error-check-all 2

Check for MCA errors - Node 2, Socket 0
        MCA_ERR_SRC_LOG = 0x00000000

MCA found in bank 14, socket 0, core 0
          IA32_MC14_STATUS = 0xf40000400040009e
          IA32_MC14_ADDR = 0x18be8bcbc0

MCA found in bank 0, socket 0, core 46
          IA32_MC0_STATUS = 0xf600000092000810
          IA32_MC0_ADDR = 0x18be8bcd00

MCA found in bank 0, socket 0, core 47
          IA32_MC0_STATUS = 0xf600000092000810
          IA32_MC0_ADDR = 0x18be8bcd00
```

Figure 7.   xtitp MCDRAM Error Output

The *xtitp* command can be used to read the CPUID as shown in Figure 8.

```
xtitp -t c3-0c2s15 cpuid-brand-string 3

CPUID Processor Brand String (Functions 0x80000002, 0x80000003,
0x80000004)
Intel(R) Xeon Phi(TM) CPU 7210 @ 1.30GHz
```

Figure 8.   xtitp CPUID Error Output

The *xtitp* command can also be used to read the PPIN as shown in Figure 9.

```
xtitp -t c3-0c2s15 msr-read 0 0x4f

>>> MSR  0x0000004f (Socket 0, Core 0, Thread 0) <<<
0x5a8e4e432e06638e
```

Figure 9. xtitp PPIN Output

Executing any script via the *xtitp* command on the SMW temporarily pauses the node, until the data is read from the processor and resumes the processor once the read is complete. Therefore, running any script causes a temporary performance degradation and should be avoided under normal operations. It is also important to note that if the processor is paused for greater than 30 seconds, HSS would lose the node heartbeat and consider the node down.

VII. Xeon Phi Diagnostics

The Cray® XC40™ system provides four diagnostic tests available for the compute nodes that validate the nodes functionality and performance. These diagnostic tests execute under CLE and are installed in /opt/cray/diag/bin.

A. Xeon Phi Memory

The Cray Xeon Phi memory test, *xtphimemory*, targets all external DDR memory and internal MCDRAM memory. *xtphimemory* is an effective user space memory test for stress-testing the memory subsystem. The maximum amount of memory that *xtphimemory* can test is less than the total amount of memory installed in the system; the kernel, libraries, and other system usage limits the memory available for testing.

The *xtphimemory* diagnostic provides four primary test algorithms as follows:

- DDR4 Memory Validation: Tests access from each Core, L2 Cache, and Caching Agent (CHA) to DDR4 DIMM
- MCDRAM Memory Validation: Tests access from each Core, L2 Cache, and CHA to MCDRAM module
- DDR4 Memory Stress: Stresses all DDR4 memory channels. Optionally, the user can define a stride to use when performing memory accesses, to allow for flexibility in creating additional conflicts
- MCDRAM Memory Stress: Stresses all MCDRAM memory channels

The parent process allocates the shared memory region to test, and one child thread is created per division of the shared memory region. For example, a 68-core processor supports 272 threads with the Intel® Hyper-Threading Technology enabled. If the node is configured to support four NUMA nodes, then the test creates 68 threads per NUMA node. If the user selected two divisions on the command line (ddr_num_slices = 2), then each division would support 34 threads. The divisions are a way to group threads within a NUMA node to allow for better failure isolation within the node and memory.

It is extremely rare to have the *xtphimemory* test provide failure information. If the hardware indicates a single bit error (SBE) has occurred, it is corrected and the diagnostic will not see the SBE as it is corrected. If the diagnostic forces a double bit error (DBE), the diagnostic is terminated with an EC_NODE_FAILED status by the kernel. Therefore, it is important to review the SBE and DBE that are logged on the SMW. The SMW command, *xthwerrlog*, can be used to display the output on the SMW for the time period when the diagnostic executes on the node under CLE.

Single bit MCDRAM errors are silent on the node (no machine checks are generated). Double bit MCDRAM errors cause a correctable MCA to be generated. Any greater than two bit MCDRAM errors are uncorrectable. The MCDRAM error are logged to the SMW and are viewable using the *xthwerrlog* command.

The diagnostic reports a data miscompare in Figure 10.

```
c0-0c2s9n3, nid00167,
    testName: testRandomValue,
    loopNum: 2,
    CpuId [88] compare regions test FAILURE:
    0x28000 != 0x50000 at offset 0xba62f0.
```

Figure 10. Memory Diagnostic Data Miscompare Output

B. Xeon Phi Non-Uniform Memory Access (NUMA)

The Cray NUMA test, *xtphinuma*, validates the NUMA capabilities of node for any given memory model that is configured with more than one NUMA node. *xtphinuma* validates both local memory and remote memory. It also provides an interleave test where memory is allocated across all the NUMA nodes. The remote memory in the context of this test does not include memory connected to other nodes that may be accessible via the Aries network, but instead only refers to memory that is in a "remote" NUMA node within the physical node. To perform the standard NUMA verification, execute the command as follows:

```
xtphinuma -s 0x9f
```

The diagnostic allocates buffers of memory and validates the ability of the Intel® Xeon Phi™ CPU 7250 processor to properly read and write the buffers. The libnuma API is used to control the NUMA allocations to local or remote NUMA nodes. The diagnostic uses threads pinned to logical cores to validate that each logical core can access memory allocated under the desired NUMA memory policy.

xtphinuma provides a bandwidth test that performs writes and reads for local and remote NUMA nodes for each core. To perform the NUMA bandwidth verification execute the command as follows:

```
xtphinuma -s 0x60
```

xtphinuma also provides a stress test that exercises all cores simultaneously to stress the memory paths from each CPU to local and remote memory. To perform the NUMA stress verification execute the command as follows:

xtphinuma -s 0x100

The diagnostic reports all data miscompares that are detected Figure 11.

```
c0-0c2s15n1, nid00189, Fail:  Data Failure for word: 1245416
             Exp Data: 1245416 Act Data: 0
c0-0c2s15n1, nid00189, Fail:  Data Failure for word: 1245417
             Exp Data: 1245417 Act Data: 0
c0-0c2s15n1, nid00189, Fail:  Data Failure for word: 1245418
             Exp Data: 1245418 Act Data: 0
```

Figure 11. NUMA Diagnostric Data Miscompare Output

The diagnostic may also fail when allocating memory and will output an appropriate error message as follows:

- allocateNumaMemoryInterleave failed
- allocateNumaMemory failed
- allocateNumaMemory failed for expected data
- allocateNumaMemory failed for read buffer
- allocateNumaMemoryInterleave failed for write buffer
- allocateNumaMemory failed for write buffer

C. *Xeon Phi Processor Performance*

The Cray Xeon Phi performance test, *xtphiperf*, provides a computationally intensive processor test to validate the node. This test outputs the performance and the power for the processor during each pass of the diagnostic test. This test uses the standard CBLAS DGEMM. It validates the results of the DGEMM matrix multiply. The diagnostic supports a command line option to enable MPI. When MPI is enabled, the diagnostic compares the results across the other nodes under test using the same input data.

Three arrays are allocated to maximum memory usage as shown in Figure 12.

```
A x B = C

Array A:[k,m] B:[n,k] and C:[n,m]
```

Figure 12. DGEMM Maximum Memory Usage

An extra matrix is used to store the initial C matrix data as shown in Figure 13.

```
So {k x m} + {n x k} + {n x m} + {n x m} < max mem

Where default values are: m = 8192, n = 8192, k = 8192
```

Figure 13. DGEMM Matrix Multiply

xtphiperf can target the DDR4, MCDRAM, or both. This is controlled through the command line option -t. To effectively target both DDR4 and MCDRAM, the user must select a memory size of matrix size (m, n, k) greater than the available DDR4 memory or the -d option should be set to option 3, NUMA node.

The *xtphiperf* diagnostic provides significant control over testing each processor core. The diagnostic outputs the performance, power, and temperature for each iteration of the diagnostic as shown in Figure 14.

```
c0-0c0s14n3, nid00059, Iteration, GFLOPS, Power(W), Temp(C)
c0-0c0s14n3, nid00059,      0,    1964,     215,    38
c0-0c0s14n3, nid00059,      1,    1968,     224,    40
c0-0c0s14n3, nid00059,      2,    1968,     223,    42
c0-0c0s14n3, nid00059,      3,    1978,     222,    44
c0-0c0s14n3, nid00059,      4,    1978,     219,    44
```

Figure 14. Performance Diagnostic Standard Output

When the *xtphiperf* diagnostic detects a residual error a non-zero return code is output and the diagnostic prints the actual and expected values as shown in Figure 15.

```
13:52:20, c0-0c2s12n2, nid00178, 3, 2039.3, 197.806, 44
13:52:21, c0-0c2s12n2, nid00178, Failed:
             CPU actual: 502.630097504761,
             CPU expected: 502.63009941210
```

Figure 15. Performance Diagnostic Residual Output

When the *xtphiperf* diagnostic detects a performance issue, the diagnostic prints the actual GFLOPS and the expected GFLOPS as shown in Figure 16.

```
13:55:22, c0-0c2s12n2, nid00178, 1, 1999.9, 197.906, 46
13:55:22, c0-0c2s12n2, nid00178, Failed:
     actual: 1899.9 GFLOPS, expected greater than: 1900 GFLOPS
```

Figure 16. Performance Diagnostic Performance Failure Output

The user expected performance target is defined in the *xtphiperf.ini* file. It is important to note that *xtphiperf* assumes that the node is dedicated to performance testing. Any other jobs that are running on the node affect the performance of the node.

D. *Xeon Phi Processor Stress Test*

The diagnostic, *xtphinls*, is both validation test and a stress test for the node. This test is a collection of diagnostic tests that validate functionality of a specific node. The test supports concurrent execution of independent test programs exercising all or part of the node resources. There are two diagnostic tests that are executed concurrently on different cores: *xtphimemory* and *xtphiperf*. These are the same diagnostics that were previously discussed. However, in this case the diagnostics are run as a single threaded test with multiple copies of these tests executing in different threads on a core. Currently the odd threads run the memory test and the even threads run the performance test. In this way one core is executing a computationally intense test and the next core is executing a memory intensive test. The combination of these two tests running on alternating cores stresses the node. Each test is pinned to a thread which is pinned to a core so on failure the failing core is properly identified. Each thread also reports a heartbeat status to the parent thread to indicate the general thread health. The thread manager within the diagnostic watches for the heartbeat and reports any stalled or failed threads.

xtphinls only verifies the results. So, when a test fails the validation check of the diagnostic, it outputs a failed message as shown in Figure 17.

```
aprun -n 1 -N 1 -L 225 --cc=none ./xtphinls -v 1
xtphinls -v 1
c1-0c0s8n1, nid00225, Version 1.0
c1-0c0s8n1, nid00225,  Test, ThreadID, Status
c1-0c0s8n1, nid00225, xtphimemory,    17532,   Pass
c1-0c0s8n1, nid00225, xtphimemory,    17534,   Failed
                                             .
                                             .
c1-0c0s8n1, nid00225, xtphiperf,    17629,   Failed
c1-0c0s8n1, nid00225, xtphiperf,    17757,   Pass
c1-0c0s8n1, nid00225, Number of passing tests: 128
c1-0c0s8n1, nid00225, Number of failing tests: 128
c1-0c0s8n1, nid00225, Test Failed
```

Figure 17. Node Stress Diagnostic Failure Output

E. Xeon Phi Processor Check

The diagnostic utility, *xtphicheck*, provides a simple verification of the node within the system. It uses MPI to gather basic information about each node and returns the data to *stdout*. Any nodes that report values that are different from the root node (first node in the list), are reported with the cname, nid, and value. Sample outout is shown in Figure 18.

```
aprun -n 2 -N 1 -L 50-51 /opt/cray/diag/bin/knl/xtphicheck -v 3

/opt/cray/diag/bin/knl/xtphicheck -v 3
/opt/cray/diag/bin/knl/xtphicheck -v 3

c0-0c0s12n2 nid00050 2 node(s) Sockets              : 1
c0-0c0s12n2 nid00050 2 node(s) Processor            : KnightsLanding
c0-0c0s12n2 nid00050 2 node(s) Cores Per Socket  : 68
c0-0c0s12n2 nid00050 2 node(s) CPU MHz             : 1401.000
c0-0c0s12n2 nid00050 2 node(s) L1d cache           : 32K
c0-0c0s12n2 nid00050 2 node(s) L1i cache            : 32K
c0-0c0s12n2 nid00050 2 node(s) L2 cache             : 1024K
c0-0c0s12n2 nid00050 2 node(s) L3 cache             : N/A
c0-0c0s12n2 nid00050 2 node(s) Memory Total      : 98880724kB
c0-0c0s12n2 nid00050 2 node(s) Memory cluster config: a2a
c0-0c0s12n2 nid00050 2 node(s) Memory cache config: cache
c0-0c0s12n2 nid00050 2 node(s) Mcdram Total     : 0x400000000
c0-0c0s12n2 nid00050 2 node(s) Numa Nodes       : 1
```

Figure 18. Node Diagnostic Utility Output

This diagnostic utility proved extremely useful during initial bring up and testing of nodes within the system. It also provided a simple check to validate the nodes configuration is set as expected.

VIII. Workload Test Suite (WTS)

The Workload Test Suite (WTS) consists of a control script, *xtsystest*, and a number of benchmarks and diagnostics. The benchmarks and diagnostic tests are used to simulate a generic application workload to verify that the system is ready to execute user applications. In some cases, a customer may have one or two applications that are representative of the workload on-site. In other cases, standard benchmarks are used as follows:

1. **Intel MPI Benchmarks (IMB):** Performs a set of MPI performance measurements for point-to-point and global communication operations for a range of message sizes [3].
2. **High Performance Computing Challenge (HPCC):** Consists of seven tests including HPL, DGEMM, STREAM, PTRANS, Random Access, FFT, and Communication bandwidth and latency [4].
3. **High Performance Linpack (HPL):** Cray uses HPL as two different tests: one to test out the processor running HPL (DGEMM) and the second to use as large a memory foot print as possible [5].

These benchmarks have been updated to support the Intel® Xeon Phi™ CPU 7250 processor. The Workload Test Suite is comprised of a script, *xtsystest.py*, a default configuration file, *xtsystest.ini*, a set of standard component test modules located under the tests folder, and a set of utilities located under the *util* folder. When executed, the tool sequentially executes the defined set of benchmarks, diagnostics, and applications. The site can also define a custom configuration file. Upon completion, the script outputs a summary of all the tests that have been executed.

By default, *xtsystest.py* continuously runs the set of tests defined in *xtsystest.ini* on the maximum number of compute nodes available until the <Ctrl-C> signal is sent. However, it is possible to limit the set of tests that are executed, the number of times each test is executed, and the set of resources under test, using the available command-line options. There is also a command line option to show the results of a given session upon completion. Each test script has a corresponding check utility that provides additional information about a failure.

On larger systems, especially for systems with multiple rows, it is recommended that an instance of the workload script is targeted to each row within the system. Each instance of the WTS script should also be initiated from a separate login node. The *xtsystest.py* script launches a diagnostic or benchmark on each node within the system concurrently. It also logs each instance individually. Therefore, a 10k node system will start 10k copies of *xhpl* and will generate 10k output files. This has been shown to quickly consume the login node and WLM resources.

The Workload Test Suite executes on CLE and supports ALPS and SLURM. SLURM support is available as a patch for CLE 6.0 UP02, CLE 6.0 UP03, and CLE 6.0 UP04.

IX. SSD DIAGNOSTIC UTILITES

There are new utilities and diagnostics used to debug and verify the configured PCIe SSD cards within the Cray® XC™ Series. There is a new diagnostic utility, *xtcheckssd*, that executes in CLE on the Data Warp node or the compute node to verify the SSD card. [13] This diagnostic utility is periodically scheduled to check the health of the SSD cards on DataWarp nodes. On compute nodes, the SSD utility is run manually as required by the site. It reports the status to the SMW via RCA. The system administrator can query and display the current SSD health, as well as the historical data. They can also query and display the results of the SSD diagnostic utility on the SMW.

There are two tools available under CLE are described in TABLE V.

TABLE V. CLE SSD DIAGNOSTIC TOOLS AND UTILITES

SSD Diagnostic Tools and Utilities	
Name	**Description**
xtcheckssd	Report SSD health
xtiossdflash	Update the firmware on the SSD card

xtcheckssd reports the health of the attached PCIe SSD cards. The diagnostic utility supports both the DataWarp SSD cards and the optional SSD cards attached to the node on the Cray PDC. The *xtcheckssd* output is shown in Figure 19.

```
PCIe slot#:1,Name:INTEL     SSDPECME040T4,
SN:CVF8515300094P0DGN-1, Size: 4000GB, Remaining life:100%,
Temperature:22(c)

PCIe slot#:1,Name:INTEL SSDPECME040T4,
SN:CVF8515300094P0DGN-2, Size: 4000GB, Remaining life:100%,
Temperature:24(c)

PCIe slot#:0,Name: INTEL SSDPECME040T4,
SN:CVF85153001V4P0DGN-1, Size: 4000GB, Remaining life:100%,
Temperature:22(c)

PCIe slot#:0,Name: INTEL SSDPECME040T4,
SN:CVF85153001V4P0DGN-2, Size: 4000GB, Remaining life:100%,
Temperature:24(c)
```

Figure 19. xtcheckssd SSD Health Output

The SSD utility, *xtssdconfig*, is available on the SMW, which is used to display the SSD configuration information.

When a DataWarp SSD reaches 90% of its life expectancy, a message is written to the console log file. If enabled, the Simple Event Correlator (SEC) monitors system log files for significant events such as this and sends a notification (either by email, IRC, writing to a file, or some user-configurable combination of all three) that this has happened. [9] The *xtcheckssd* results are logged on the SMW in the *xtdiag* log.

Additionally, *xtcheckhss* reports the PCIe attached SSD cards. *xtcheckhss* reports the targeted and trained PCIe speed and width as shown in Figure 20.

```
==============================================
============= PCIe Card Info ==============
==============================================
Node       Slot Name
           Target Gen Trained Gen Target Width Trained Width

c0-0c2s0n0  0  Samsung_SM951_M.2_SSD
           Gen2      Gen2        x4        x4
c0-0c2s0n1  0  Samsung_SM951_M.2_SSD
           Gen2      Gen2        x4        x4
c0-0c2s0n2  0  Samsung_SM951_M.2_SSD
           Gen2      Gen2        x4        x4
c0-0c2s0n3  0  Samsung_SM951_M.2_SSD
           Gen2      Gen2        x4        x4
```

Figure 20. xtcheckhss SSD configuration

This ensures that the state of the PCIe SSD card is known at time of boot.

X. RELATED WORK

As mentioned previously, an earlier paper has described in detail the Cray® XC™ system level diagnosability toolset [1]. That paper focused on the Aries high speed network (HSN), compute processors, co-processors, GPUs, and Cray® XC™ cabinet power and cooling. The system diagnostics included boot, confidence, stress, performance, workload, and error and data reporting.

A subsequent paper then described the Cray® XC™ system node level diagnosability [2] toolset in greater detail. Various troubleshooting examples were presented with detailed examples. The Cray® XC™ system diagnosability roadmap [3] was also presented. The roadmap covered Intel® node, Nvidia® GPU, and I/O diagnosability enhancements. It also covered the workload test suite (WTS), HSS diagnostic utilities, telemetry data, HSS controller monitoring, notification, and Cray® Data Virtualization Service (Cray DVS).

D. Petesch and M. Swan [16] described mechanisms to properly instrument a user level application to quickly checkout the hardware and software components in a large external Lustre® file system [19]. That paper also provided insights into full scale performance issues and how IOR [18] can be used to troubleshoot them.

M. Swan [17] described the mechanisms and tool sets required to tune and analyze Cray® Sonexion 1600 performance issues in a Cray® XC™ system.

J. Fullop and R. Sisneros [19] have created a framework for identifying causes for observed differences in job performance from one run to another. Their approach is to analyze the various logs over a period of time looking for any noteworthy events. This work was done at the

University of Illinois at Urbana-Champaign on the Blue Waters system.

Finally, A. DeConnick, et.al. [20] have described a scalable monitoring system for Trinity. The authors work with the Baler log file analysis tool to extract a greatly reduced set of patterns from the voluminous number of logs on a Cray® XC™ system is particularly important as the number of nodes and controllers increases and the requirement for additional automated analysis grows.

XI. SUMMARY

Cray has provided a number of diagnostics, commands, and utilities that enhance the diagnosability of the Cray® XC40™ system. The focus of system diagnosability has been on ensuring that each component is functioning properly by ensuring that each component can be validated and that all data is captured at the time of failure.

Each aspect of the tool chain has been enhanced on the Cray® XC40™ system to better ensure that the Cray® XC40™ system is performing as expected. The changes to support the Intel® Xeon Phi™ CPU 7250 processor include BIOS, SMW commands, utilities, and diagnostics, as well as, power and thermal telemetry data, and event logs. Future enhancements are planned to continue to improve system diagnosability of the Cray® XC40™ system.

REFERENCES

[1] J. Schutkoske, "Cray XC System Level Diagnosability: Commands, Utilities and Diagnostic Tools for the Next Generation of HPC Systems ", *Proceedings of the Cray User Group (CUG)*, 2014, https://cug.org/proceedings/cug2014_proceedings/includes/files/pap120.pdf

[2] J. Schutkoske, "Cray XC System Node Level Diagnosability", *Proceedings of the Cray User Group (CUG)*, 2015, https://cug.org/proceedings/cug2015_proceedings/includes/files/pap130.pdf

[3] J. Schutkoske, "Cray XC System Diagnosability Roadmap", *Proceedings of the Cray User Group (CUG)*, 2015, https://cug.org/proceedings/cug2015_proceedings/includes/files/pap131.pdf

[4] Intelligent Platform Management Interface Specification, version 2.0, 2004, http://www.intel.com/design/servers/ipmi/spec.htm.

[5] ITP700 Debug Port, Design Guide, February 2004, http://download.intel.com/support/processors/xeon/sb/24967914.pdf

[6] High Performance Computing Challenge (HPCC) Benchmark, Version 1.4.2, [Online], http://icl.cs.utk.edu/hpcc/.

[7] High Performance Linpack (HPL) Benchmark, Version 2.1, [Online] http://www.netlib.org/benchmark/hpl/.

[8] Intel MPI Benchmark, Version 3.2.4, [Online], https://software.intel.com/en-us/articles/intel-mpi-benchmarks.

[9] "XC™ Series SEC Software Configuration Guide", (CLE 6.0.UP03), S-2542, http://docs.cray.com/PDF/XC_Series_SEC_Software_Configuration_Guide_CLE60UP03_S-2542.pdf

[10] "XC™ Series System Environment Data Collections (SEDC) Administration Guide", (CLE 6.0.UP03), S-2491, http://docs.cray.com/PDF/XC_Series_System_Environment_Data_Collections_SEDC_Administration_Guide_CLE60UP03_S-2491.pdf

[11] "XC™ Series Power Management Administration Guide", (CLE 6.0.UP03), S-0043, http://docs.cray.com/PDF/XC_Series_Power_Management_Administration_Guide_CLE60UP03_S-0043.pdf

[12] "CAPMC API Documentation," http://docs.cray.com/PDF/CAPMC_API_Documentation_1.2.pdf

[13] "XC™ Series DataWarp™ Installation and Administration Guide", (CLE 6.0.UP03), S-2564, http://docs.cray.com/PDF/XC_Series_DataWarp_Installation_and_Administration_Guide_CLE60UP03_S-2564.pdf

[14] S. Martin, D. Rush, M. Kappel, M. Sandstedt, and J. Williams, "Cray XC40 Power Monitoring and Control for Knights Landing," *Proceedings of the Cray User Group (CUG)*, 2016, https://cug.org/proceedings/cug2016_proceedings/includes/files/tut103.pdf

[15] M. Swan, "Tuning and Anlyzing Sonexion Performance", *Proceedings of the Cray User Group (CUG)*, 2014, https://cug.org/proceedings/cug2014_proceedings/includes/files/pap119.pdf

[16] D. Petesch and M Swan, "Instrumenting IOR to Diagnose Performance Issues on Lustre File Systems", *Proceedings of the Cray User Group (CUG)*, 2013, https://cug.org/proceedings/cug2013_proceedings/includes/files/pap152.pdf

[17] IOR HPC Benchmark, Version 2.10.3, [Online], https://github.com/LLNL/ior

[18] Lustre filesystem, [Online], http://lustre.org

[19] J. Fullop and R. Sisneros, "A Diagnostic Utility For Analyzing Periods of Degraded Job Performance", *Proceedings of the Cray User Group (CUG)*, 2014, https://cug.org/proceedings/cug2014_proceedings/includes/files/pap161.pdf

[20] A. DeConinck, A. Bonnie, K. Kelly, S. Sanchez, C. Martin, M. Mason, J. Brandt, A. Gentile, B. Allan, A. Agelastos, M. Davis, and M. Berry, "Design and implementation of a scalable monitoring system for Trinity", *Proceedings of the Cray User Group (CUG)*, 2016, https://cug.org/proceedings/cug2016_proceedings/includes/files/pap126.pdf

[21] P. Hill, C. Snyder, and J. Sygulla, "KNL System Software", *Proceedings of the Cray User Group (CUG)*, 2017