Intel Xeon®, Intel Xeon Phi™ Coprocessor and KNL Architecture

Silvio Stanzani , Raphael Cóbe , Rogério Iope

UNESP - Núcleo de Computação Científica

silvio@ncc.unesp.br , rmcobe@ncc.unesp.br , rogerio@ncc.unesp.br

Module Outline

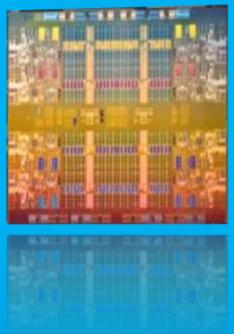
- Intel Xeon and Intel® Xeon Phi™
- System Software and OS
- Intel KNL
- Programming Model

Module Outline

- **Intel Xeon and Intel® Xeon Phi™**
- System Software and OS
- Intel KNL
- Programming Model

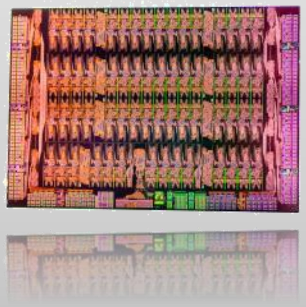
Intel Xeon and Intel® Xeon Phi™ Overview

Intel® Multicore Architecture



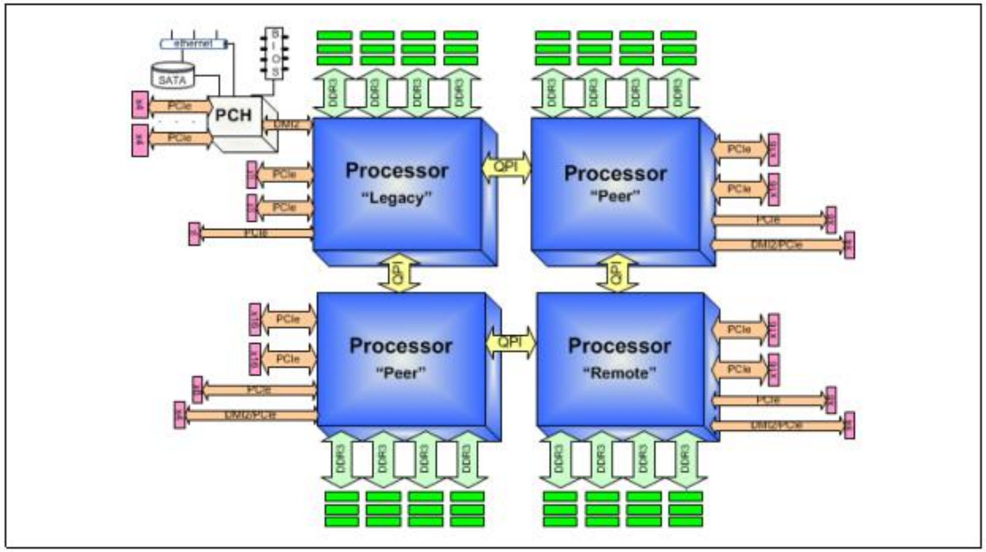
- ❖ Foundation of HPC Performance
- ❖ Suited for full scope of workloads
- ❖ Focus on fast single core/thread performance with “moderate” number of cores

Intel® Many Integrated Core Architecture



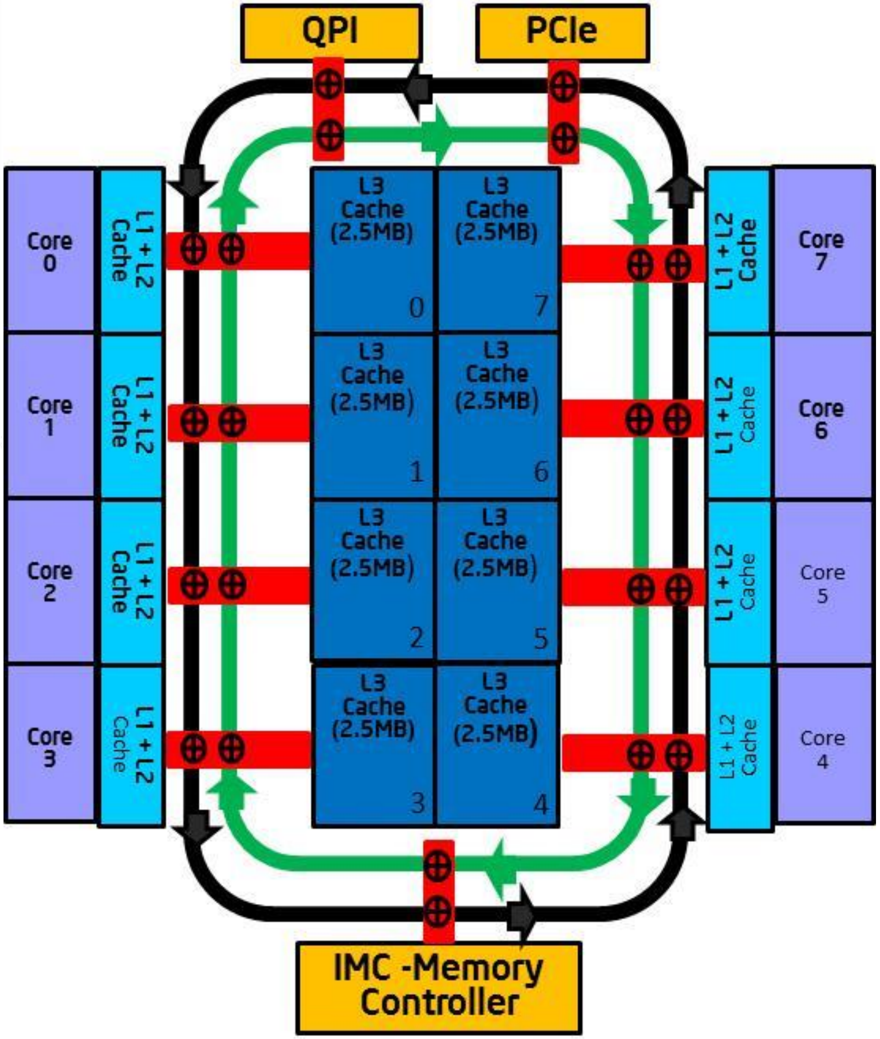
- ❖ Performance and performance/watt optimized for highly parallelized compute workloads
- ❖ IA extension to Manycore
- ❖ Many cores/threads with wide SIMD

Intel Xeon Architecture Overview



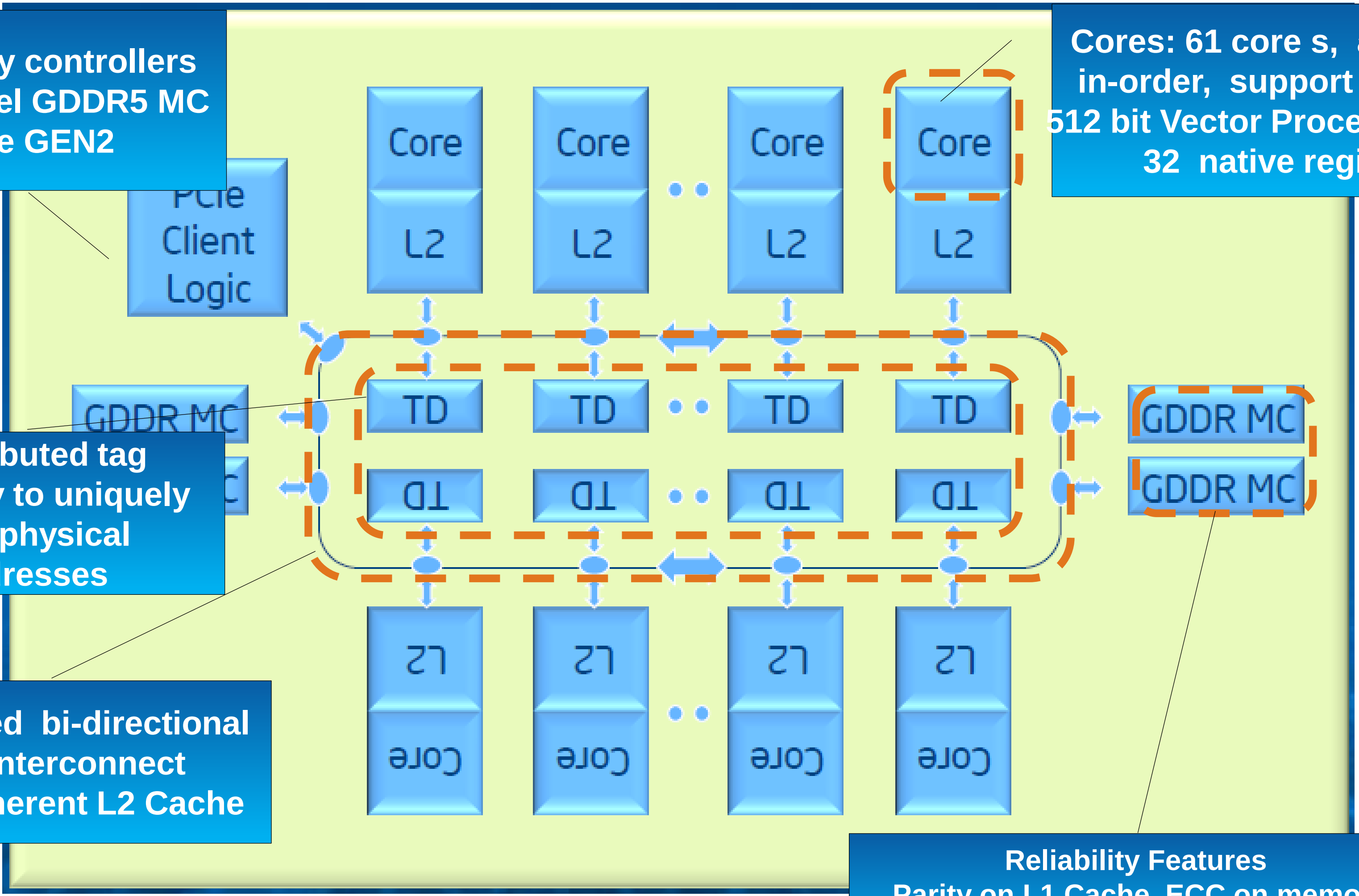
•Socket: mechanical component that provides mechanical and electrical connections between a microprocessor and a printed circuit board (PCB).

•QPI (Intel QuickPath Interconnect): high speed, packetized, point-to-point interconnection, that stitch together processors in distributed shared memory and integrated I/O platform architecture.



Intel® Xeon Phi™ Architecture Overview

- Knights Core (KNC)



8 memory controllers
16 Channel GDDR5 MC
PCIe GEN2

Cores: 61 core s, at 1.1 GHz
in-order, support 4 threads
512 bit Vector Processing Unit
32 native registers

Distributed tag directory to uniquely map physical addresses

High-speed bi-directional ring interconnect
Fully Coherent L2 Cache

Reliability Features
Parity on L1 Cache, ECC on memory
CRC on memory IO, CAP on memory IO

Module Outline

- Intel Xeon and Intel® Xeon Phi™
- **System Software and OS**
- Intel KNL
- Programming Model

Intel® Xeon Phi™ Coprocessor Arch – System SW Perspective

- Large SMP UMA machine – a set of x86 cores
  - 4 threads
    - ❑ 32 KB L1 I/D
    - ❑ 512 KB L2 per core
  - Supports loadable kernel modules
  - VM subsystem, File I/O

- Virtual Ethernet driver
  - supports NFS mounts from Intel® Xeon Phi™ Coprocessor
  - Support bridged network

Intel® MIC Programming Considerations

- Getting full performance from the Intel® MIC architecture requires both a high degree of parallelism and vectorization
  - Not all code can be written this way
  - Not all programs make sense on this architecture

- KNC comes with 8GB or 16GB of memory
  - Only ~7GB or ~15GB is available to your program
    - ❑ The other ~1GB is used for data transfer and is accessible to your Intel® MIC Architecture code as buffers.

- Very short (low-latency) tasks not optimal for offload to the coprocessor
  - Costs that you need to amortize to make it worthwhile:
    - ❑ Cost of code and data transfer
    - ❑ Cost of process/thread creation
  - Fastest data transfers currently require careful data alignment

Module Outline

- Intel Xeon and Intel® Xeon Phi™
- System Software and OS
- **Intel KNL**
- Programming Model

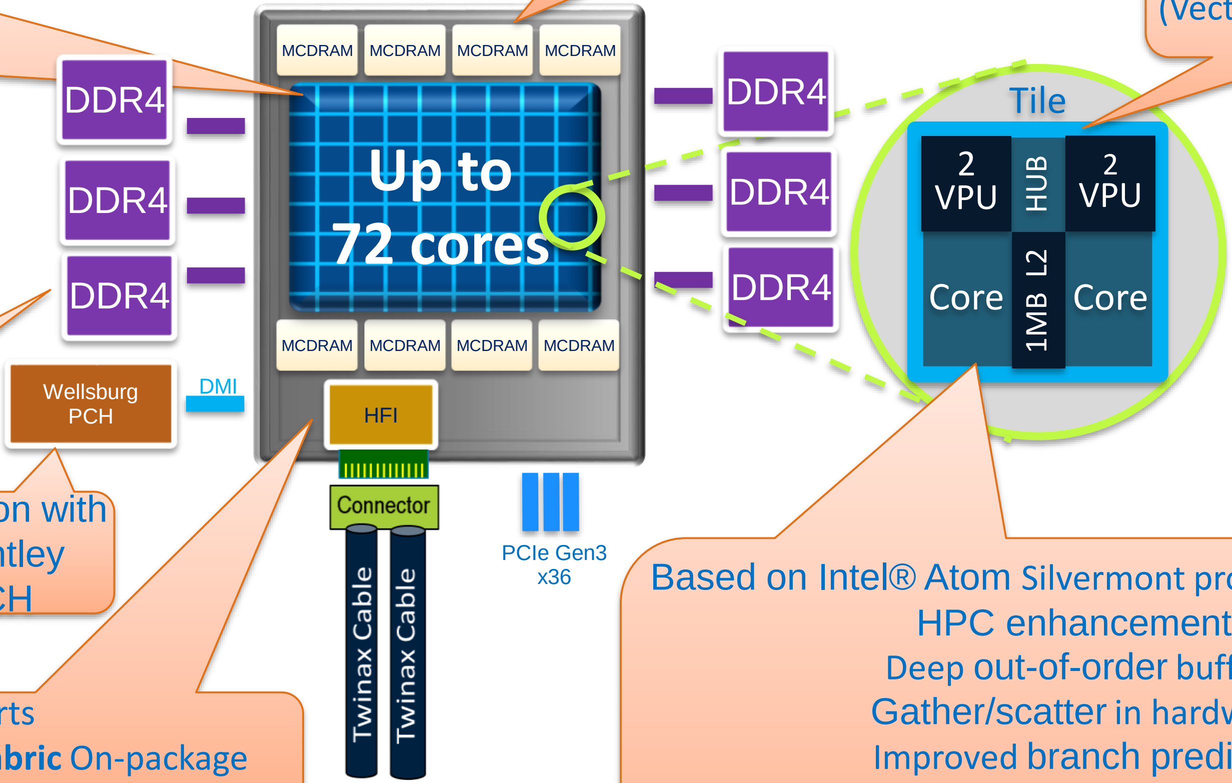
Knights Landing (KNL)

Over 3 TF DP peak
Full Xeon ISA compatibility through AVX-512
~3x single-thread vs. compared to Knights Corner

Up to 16GB high-bandwidth on-package memory (MCDRAM)
Exposed as NUMA node
~500 GB/s sustained BW

Up to 72 cores
2D mesh architecture

2x 512b VPU per core
(Vector Processing Units)



6 channels DDR4
Up to 384GB

Common with Grantley PCH

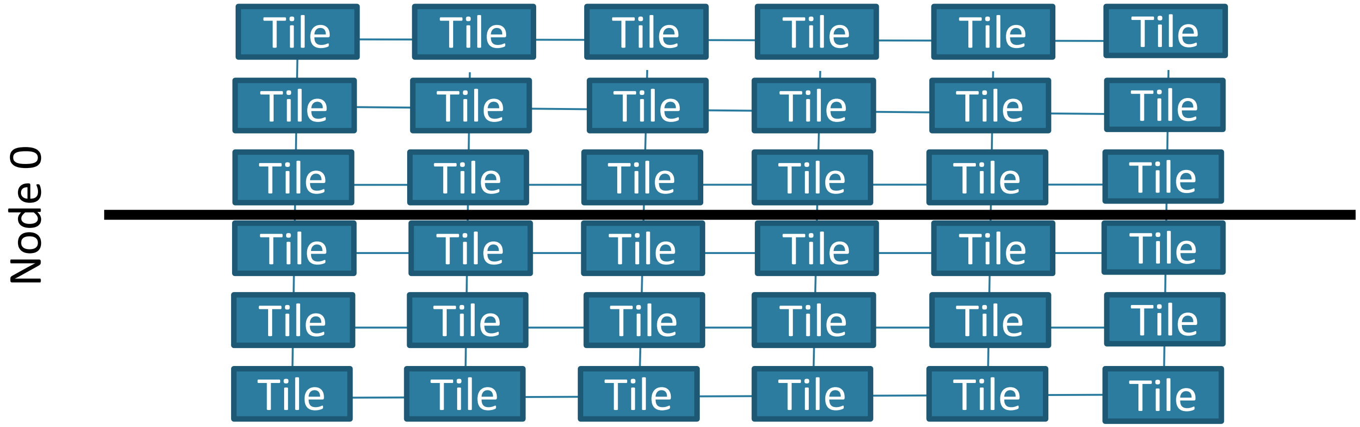
2 ports
Intel Omni-Path Fabric On-package
50 GB/s bi-directional

Based on Intel® Atom Silvermont processor with many HPC enhancements
Deep out-of-order buffers
Gather/scatter in hardware
Improved branch predition
4 threads/core
High cache bandwidth
& more

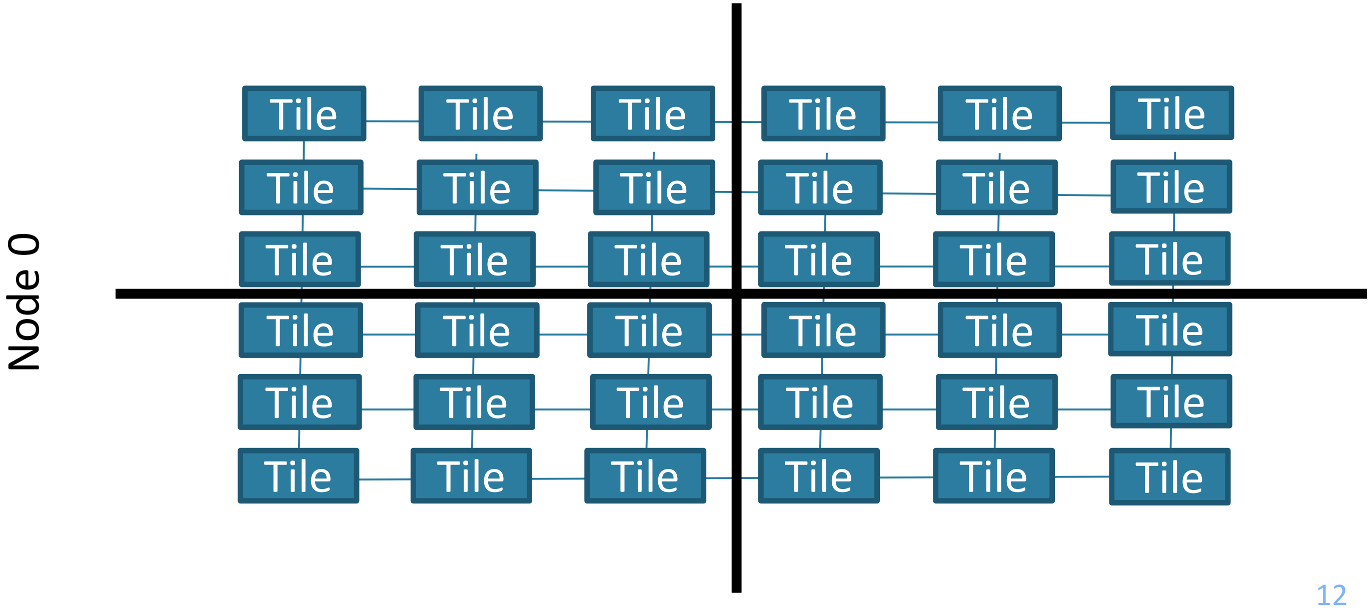
Cluster modes

One single space address

Hemisphere:
the tiles are divided
into two parts
called hemisphere


Node 0
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile

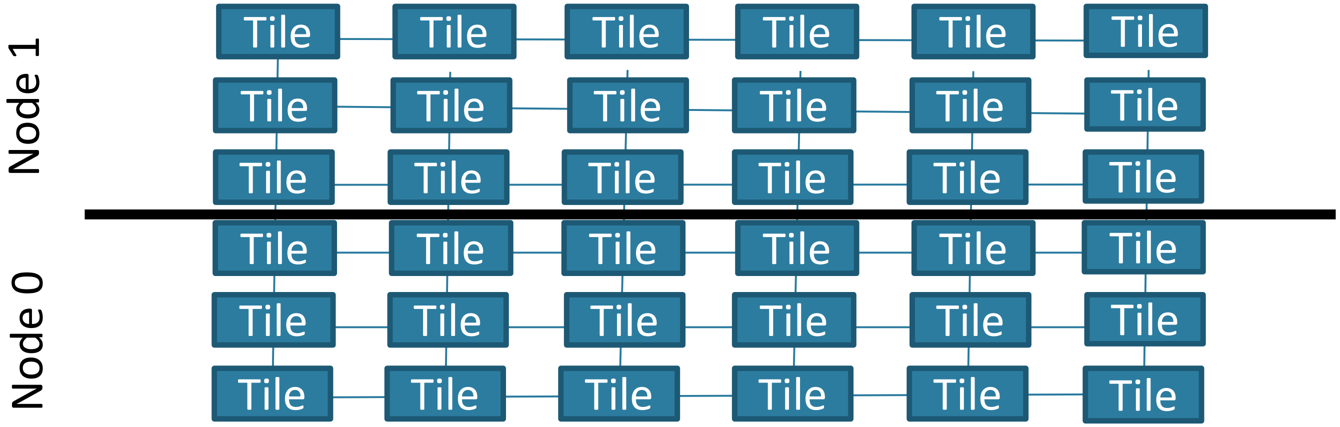
Quadrant:
tiles are divided
into two parts
called hemisphere
or into four parts
called qudrants


Node 0
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile
Tile Tile Tile Tile Tile Tile

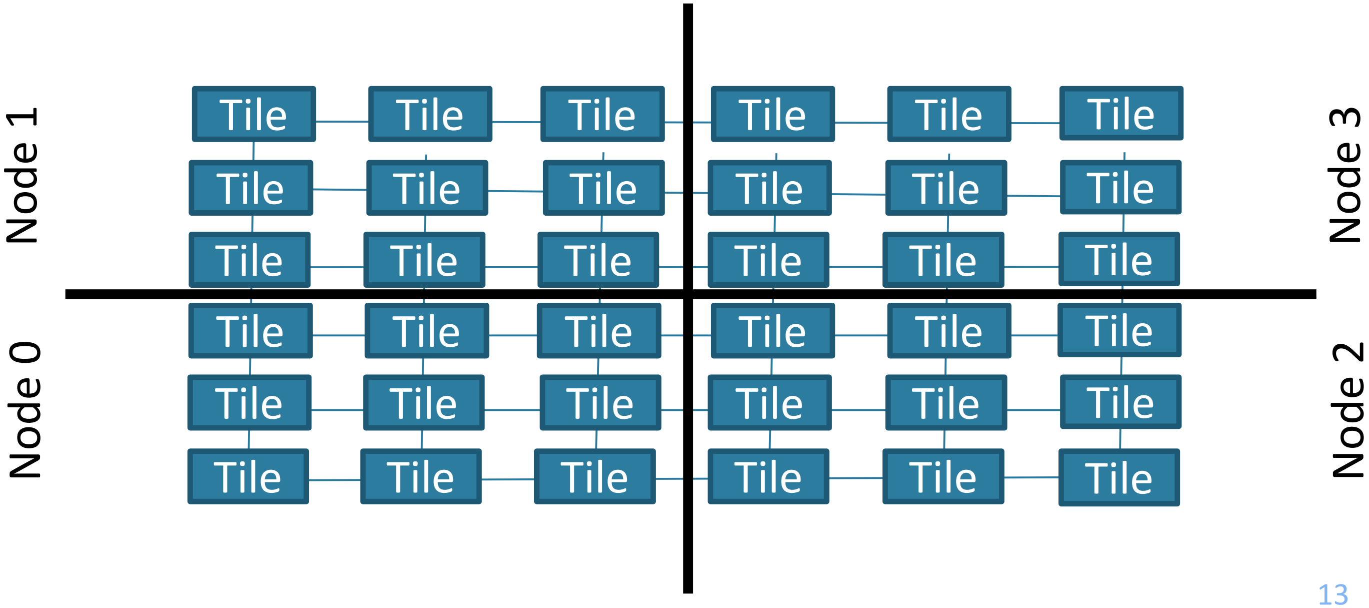
Cluster modes

Cache data are isolated in each sub numa domain

SNC-2:
the tiles are
divided into two
Numa Nodes

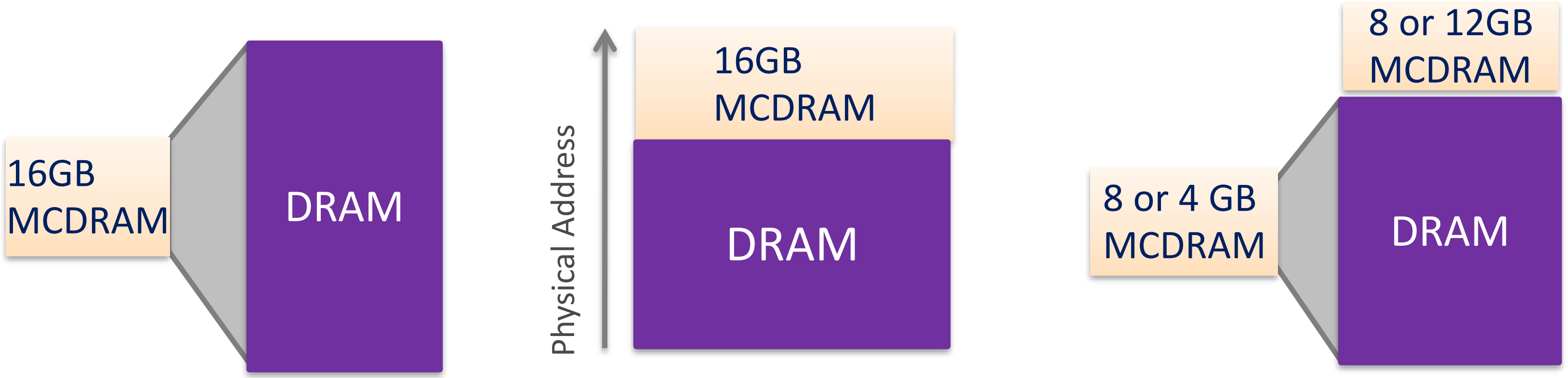


SNC-4:
the tiles are
divided into two
Numa Nodes



Integrated On-Package Memory Usage Models

Integrated On-Package Memory Usage Models



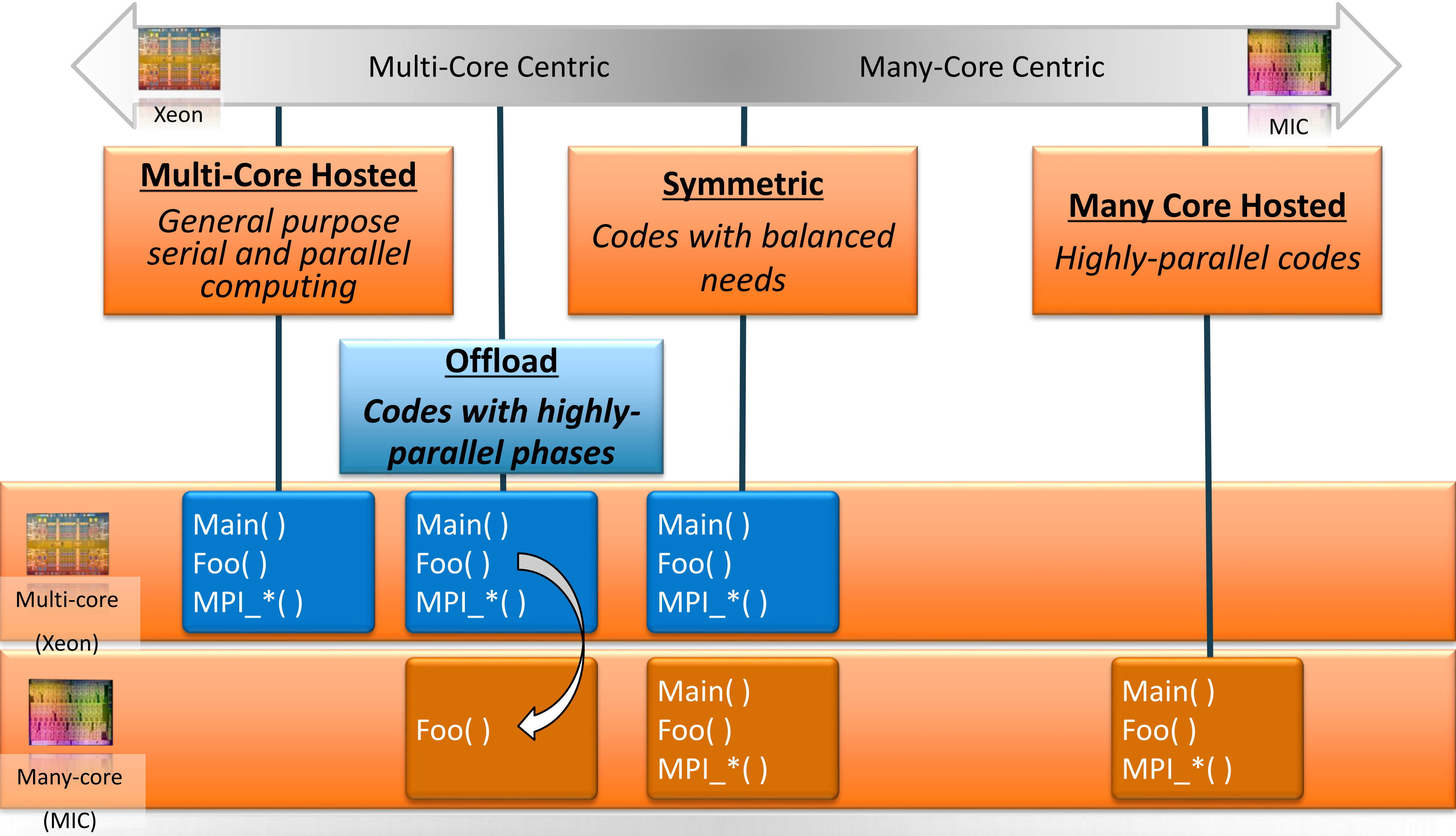
Split Options:
25/75% or 50/50%

Cache Model	Flat Model	Hybrid Model
Hardware automatically manages the MCDRAM as a “L3 cache” between CPU and ext DDR memory	Manually manage how the app uses the integrated on-package memory and external DDR for peak perf	Harness the benefits of both Cache and Flat models by segmenting the integrated on-package memory
▪ App and/or data set is very large and will not fit into MCDRAM ▪ Unknown or unstructured memory access behavior	▪ App or portion of an app or data set that can be, or is needed to be “locked” into MCDRAM so it doesn’t get flushed out	▪ Need to “lock” in a relatively small portion of an app or data set via the Flat model ▪ Remaining MCDRAM can then be configured as Cache

Module Outline

- Intel Xeon and Intel® Xeon Phi™
- System Software and OS
- Intel KNL
- **Programming Model**

Programming Models



Xeon

Multi-Core Centric

Many-Core Centric

MIC

Multi-Core Hosted

General purpose serial and parallel computing

Offload

Codes with highly-parallel phases

Symmetric

Codes with balanced needs

Many Core Hosted

Highly-parallel codes

Multi-core (Xeon)

Main( )
Foo( )
MPI_*( )

Main( )
Foo( )
MPI_*( )

Main( )
Foo( )
MPI_*( )

Many-core (MIC)

Foo( )

Main( )
Foo( )
MPI_*( )

Main( )
Foo( )
MPI_*( )

Range of models to meet application needs

Offload Programming Model Data Transfer

- Programmer designates variables that need to be copied between host and card in the offload directive using Pragma/directive;

- Variables and functions Allocation on both the host and device (C/C++):
  - __attribute__ ((target(mic))) [variable or function definition]
  - __declspec(target(mic)) [variable or function definition] (windows only)

  - For entire files or large blocks of code
    - ❑ #pragma offload_attribute (push, target(mic))
    - ❑ #pragma offload_attribute (pop)

Offload Programming Model Data Transfer

- Target
  - #pragma offload target(mic[:dev-id]) [clauses]
- dev-id: number of device to perform the offload
- clauses for explicit copy:
  - nocopy : allocate memory on device;
  - in : transfer a variable from host to device;
  - out : transfer a variable from device to host;
  - inout :
    - ❑ transfer a variable from host to device before start execution;
    - ❑ transfer a variable from device to host after finish execution;

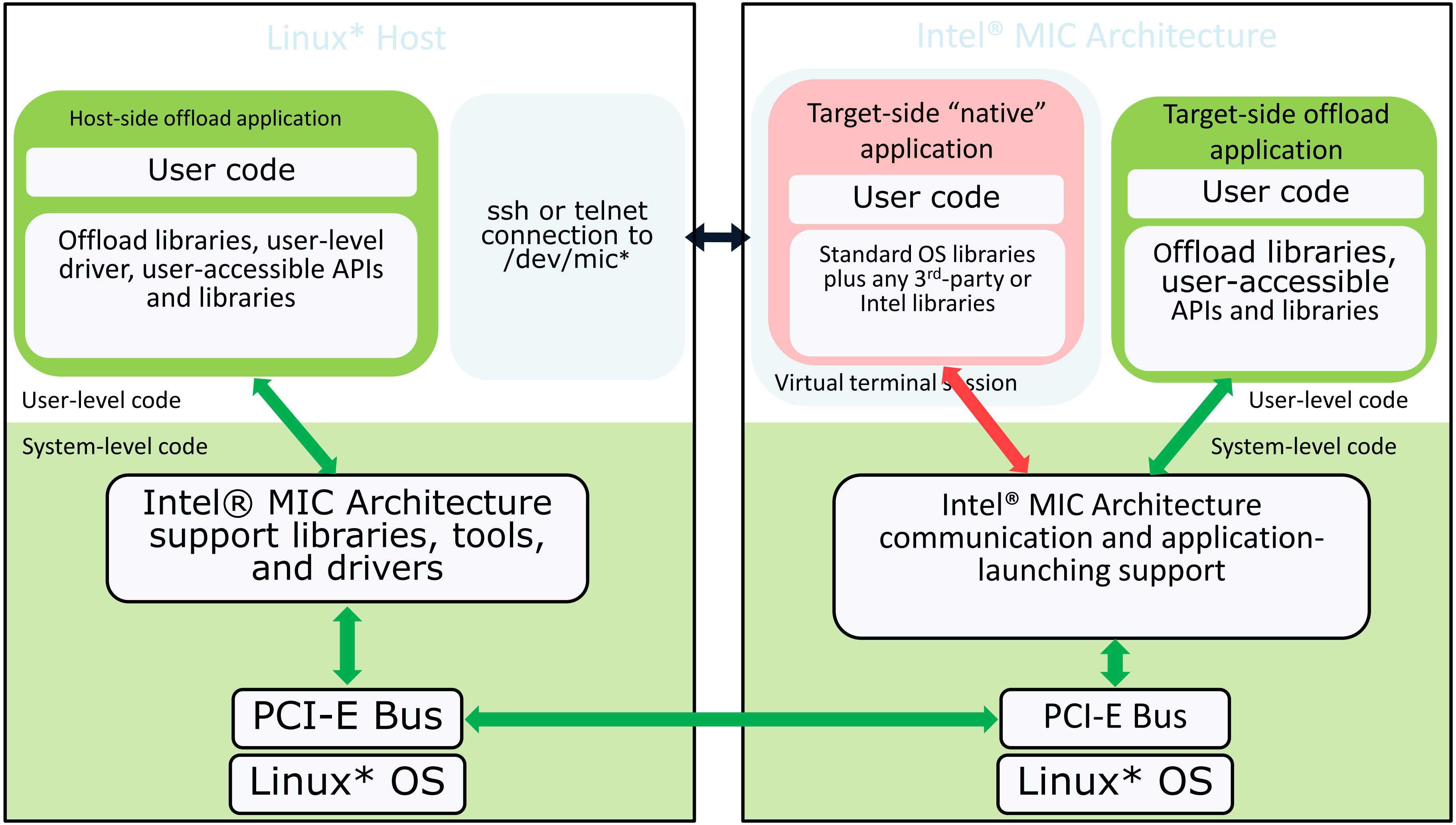
Offload Report

- OFFLOAD REPORT:
  - Measures the amount of time it takes to execute an offload region of code;
  - Measures the amount of data transferred during the execution of the offload region;
  - Turn on the report: export OFFLOAD_REPORT=2

- **[Var]** The name of a variable transferred and the direction(s) of transfer.
- **[CPU Time]** The total time measured for that offload directive on the host.
- **[MIC Time]** The total time measured for executing the offload on the target.
- **[CPU->MIC Data]** The number of bytes of data transferred from the host to the target.
- **[MIC->CPU Data]** The number of bytes of data transferred from the target to the host.

Two Application Execution Environments

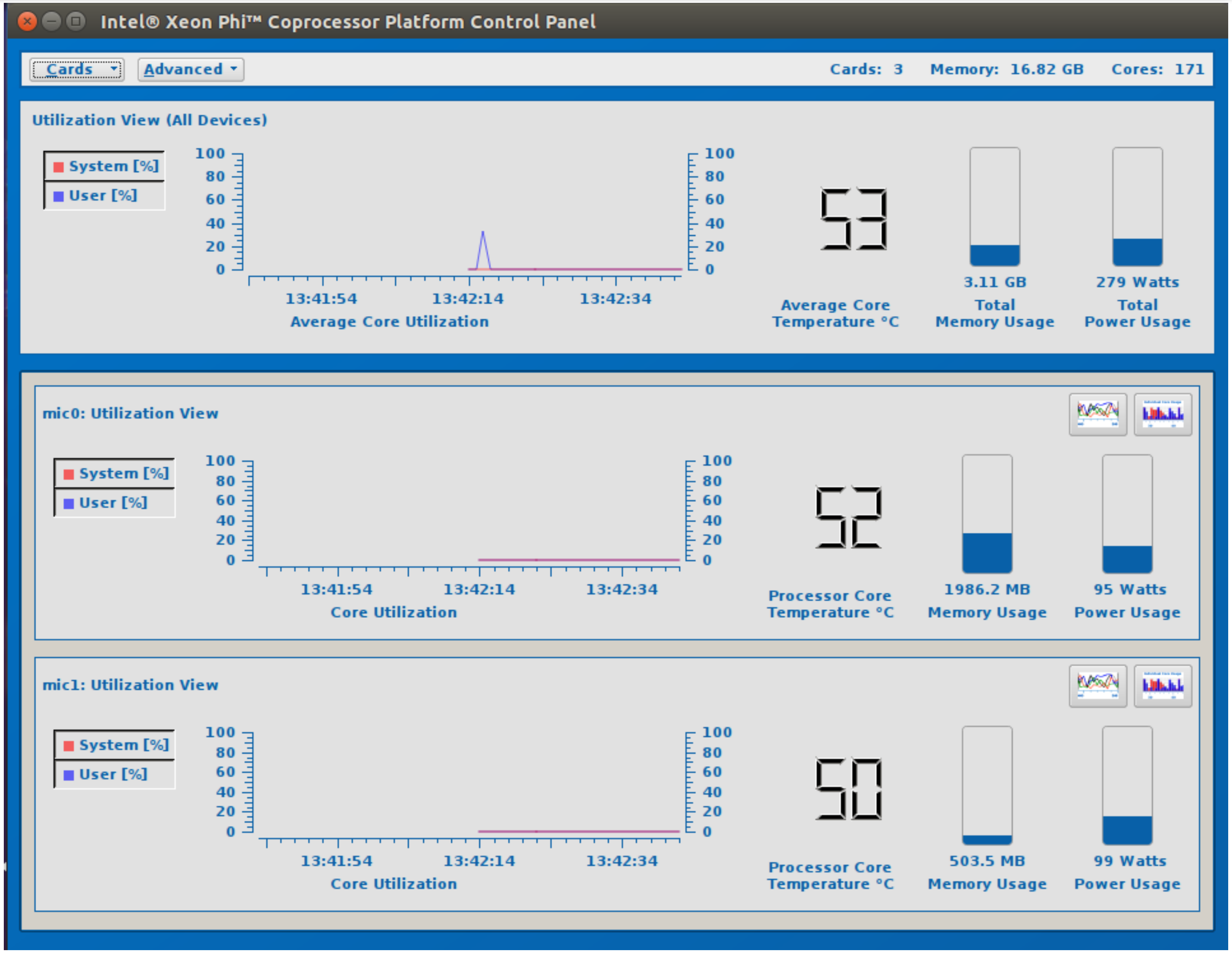
+Hybrid



Linux* Host
Host-side offload application
User code
Offload libraries, user-level driver, user-accessible APIs and libraries
ssh or telnet connection to /dev/mic*
User-level code
System-level code
Intel® MIC Architecture support libraries, tools, and drivers
PCI-E Bus
Linux* OS
Intel® MIC Architecture
Target-side "native" application
User code
Standard OS libraries plus any 3rd-party or Intel libraries
Virtual terminal session
Target-side offload application
User code
Offload libraries, user-accessible APIs and libraries
User-level code
System-level code
Intel® MIC Architecture communication and application-launching support
PCI-E Bus
Linux* OS

Examples

- Compiling
- Executing
- Offload
  - Code region
  - function
- Utilities
  - Micsmc
  - miccheck



Intel® Xeon Phi™ Coprocessor Platform Control Panel

Cards: 3 Memory: 16.82 GB Cores: 171

Utilization View (All Devices)

53 Average Core Temperature °C

3.11 GB Total Memory Usage

279 Watts Total Power Usage

mic0: Utilization View

52 Processor Core Temperature °C

1986.2 MB Memory Usage

95 Watts Power Usage

mic1: Utilization View

50 Processor Core Temperature °C

503.5 MB Memory Usage

99 Watts Power Usage