Performance Optimization for Intel Knights Landing

Silvio Stanzani , Raphael Cóbe , Rogério Iope, Jefferson Fialho, Júlio Amaral e José Vargas

UNESP - Núcleo de Computação Científica

[silvio, rmcobe, jfialho, rogerio, julioamaral, jruizvar]@ncc.unesp.br

UNESP Center for Scientific Computing

- Consolidates scientific computing resources for São Paulo State University (UNESP) researchers
  - It mainly uses Grid computing paradigm

- Main users
  - UNESP researchers, students, and software developers
  - SPRACE (São Paulo Research and Analysis Center)
  - physicists and students

- Caltech, Fermilab, CERN
- São Paulo CMS Tier-2 Facility

Open Positions

email to: cs-jobs[at]ncc unesp br

- High Performance Heterogeneous Computing
- Network Engineer/Architect
- Software-Defined Networking (SDN).
- Grid & Cloud Computing

Agenda

- High Performance Computing (HPC)
- Data Parallelism
- Task Parallelism
- Basic Linear Algebra Subroutines (Blas)
- Intel Xeon and Intel Xeon Phi
- Intel Knights Landing
- Performance Evaluation

High Performance Computing (HPC)

- High Performance Computing (HPC) [1]
  - Practice of aggregating computing power in a way that delivers much higher performance;
  - Solve large problems in science, engineering, or business.

- Motivations
  - ❑ My problem is big
  - ❑ My problem is complex
  - ❑ My computer is too small and too slow to solve my problem

- When I care how fast I get an answer, I need HPC!

[1]http://insidehpc.com/hpc-basic-training/what-is-hpc/

Parallel Processing

- A parallel computer is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem

- Parallel processing includes techniques and technologies that make it possible to compute in parallel

  - Hardware, networks, operating systems, parallel libraries, languages, compilers, algorithms, tools, …

- Parallel computing is an evolution of serial computing
  - Parallelism is natural
  - Computing problems differ in level / type of parallelism

Parallelism

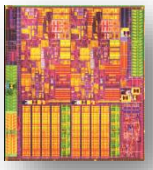
- Consider multiple tasks to be executed in a computer
  - Tasks are concurrent with they can execute at the same time (**concurrent** execution)

- If a task requires results produced by other tasks in order to execute correctly, the task's execution is dependent
  - Some form of synchronization must be used to enforce (satisfy) dependencies

- Parallelism = concurrency + "parallel" hardware
  - Find concurrent execution opportunities
  - Develop application to execute in parallel
  - Run application on parallel hardware

Multilevel Parallelism

- Node Level Parallelism
  - Cluster
- Multiple Devices
  - intra-node - offload
- Thread/Task Level Parallelism
  - Multi & Many Core
- Data Level Parallelism (Vectorization)
  - Intel® AVX & Intel® MIC
- Instruction Level Parallelism
  - Processor Architecture



Agenda

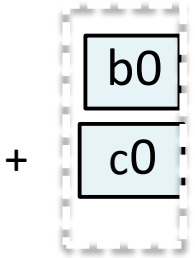
- High Performance Computing (HPC)
- Data Parallelism
- Task Parallelism
- Basic Linear Algebra Subroutines (Blas)
- Intel Xeon and Intel Xeon Phi
- Intel Knights Landing
- Performance Evaluation

Scalar and Vector Instructions

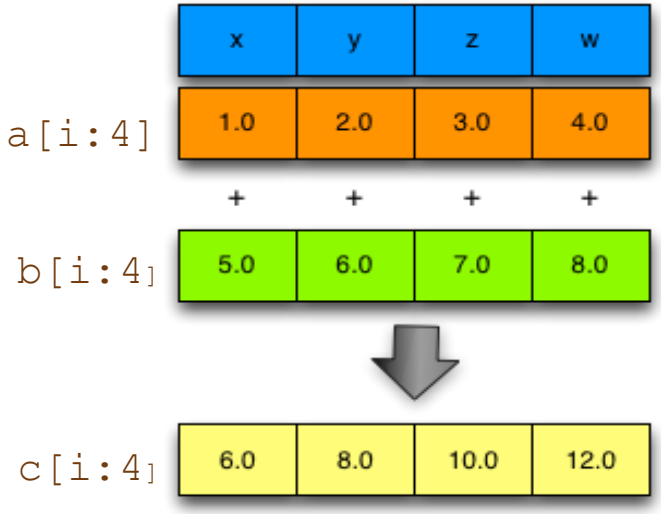
- **Scalar** Code computes this one-element at a time.
- **Vector (or SIMD)** Code computes more than one element at a time.
  - SIMD stands for **S**ingle **I**nstruction **M**ultiple **D**ata.
- Vectorization
  - Loading data into cache accordingly;
  - Store elements on SIMD registers or vectors;
  - Iterations need to be independent;
  - Usually on inner loops.

```
float *A, *B, *C;
for(i=0;i<n;i++){
  A[i] = B[i] + C[i];
}
```

- Scalar



- SIMD



Memory System

Fast

CPU Register: internal Processor Memory. Stores data or instruction to be executed

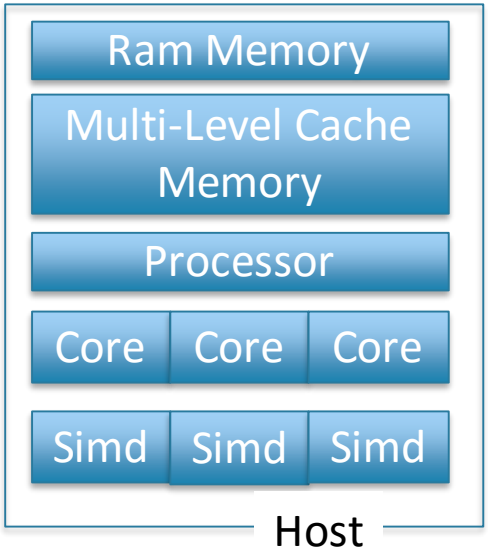
Cache: stores segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations

Main memory: only program and data currently needed by the processor resides in main memory

Auxiliary memory: devices that provides backup storage

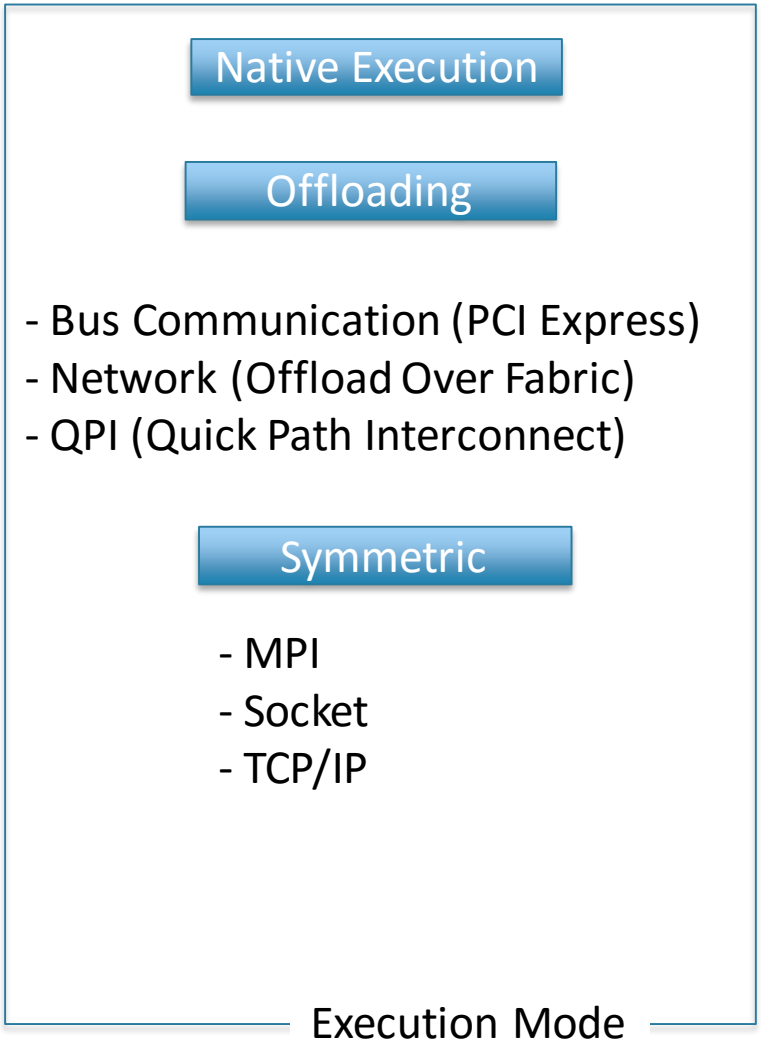
Larger in Size

Hybrid Parallel Architectures



- Heterogeneous Multilevel Memory System

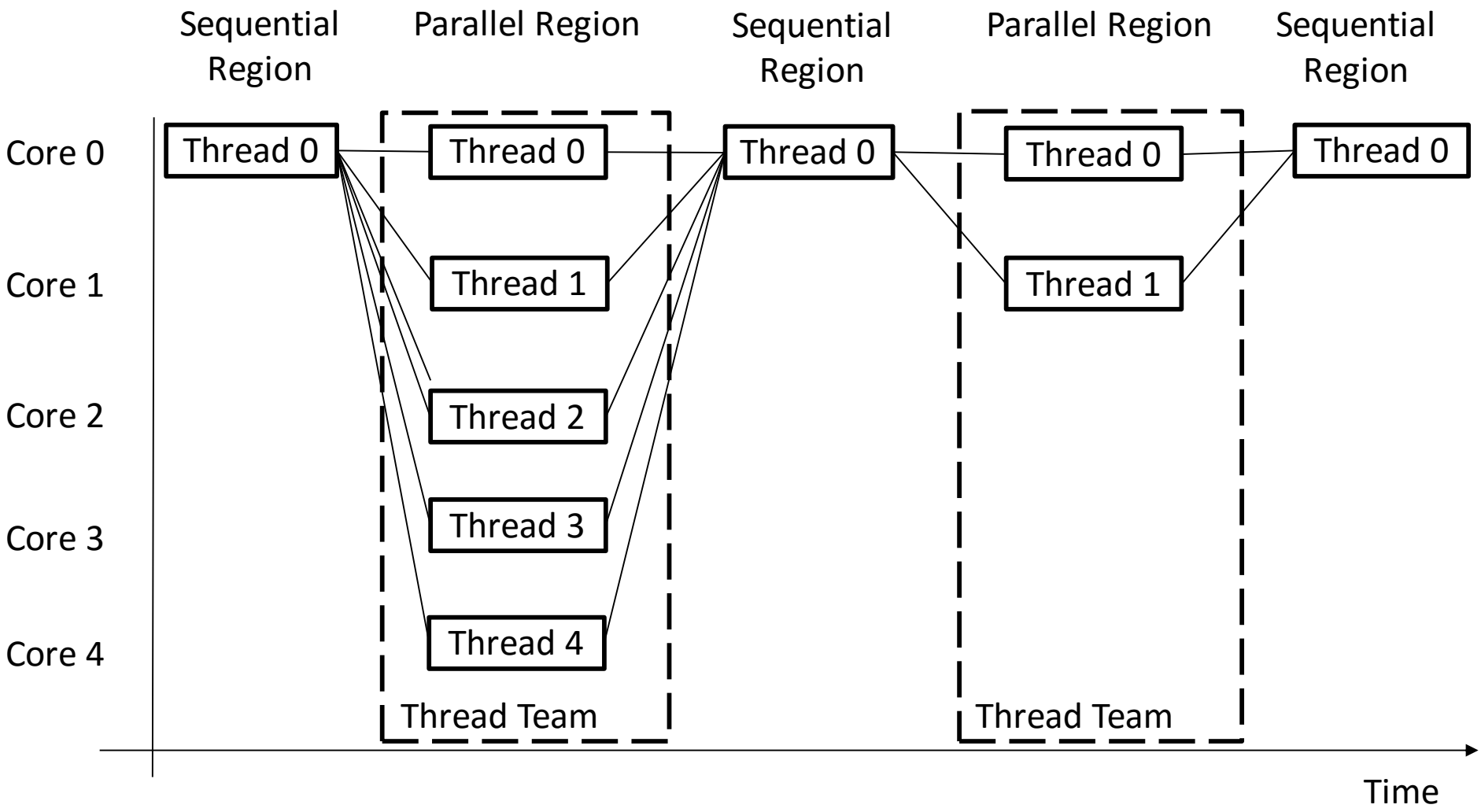
- Multilevel parallelism:
  - Instruction Level;
  - Vectorization;
  - Multithreading;
  - Multiprocessing;



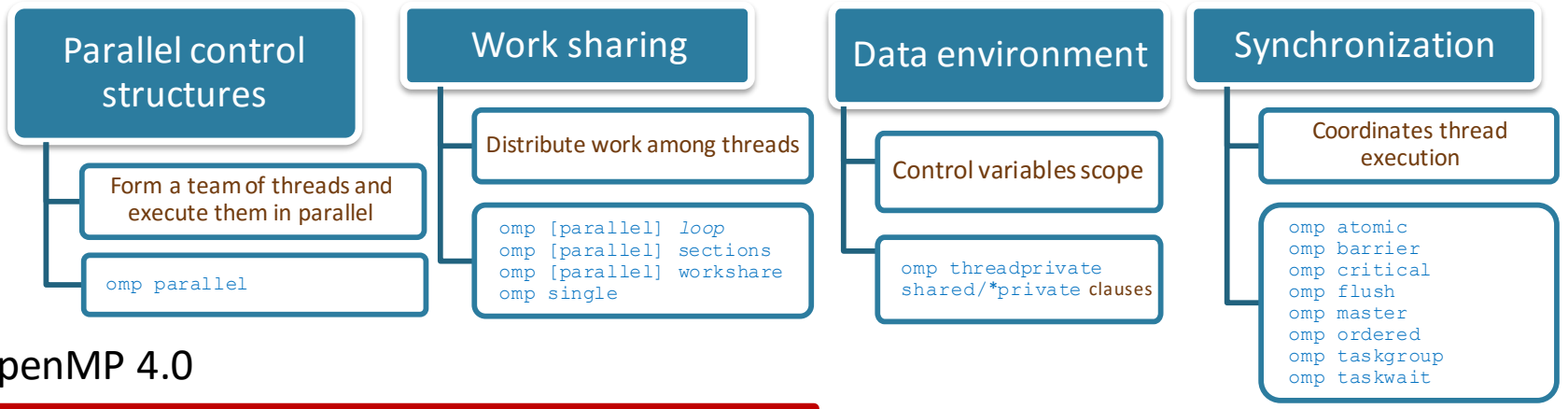
Agenda

- High Performance Computing (HPC)
- Data Parallelism
- Task Parallelism
- Basic Linear Algebra Subroutines (Blas)
- Intel Xeon and Intel Xeon Phi
- Intel Knights Landing
- Performance Evaluation

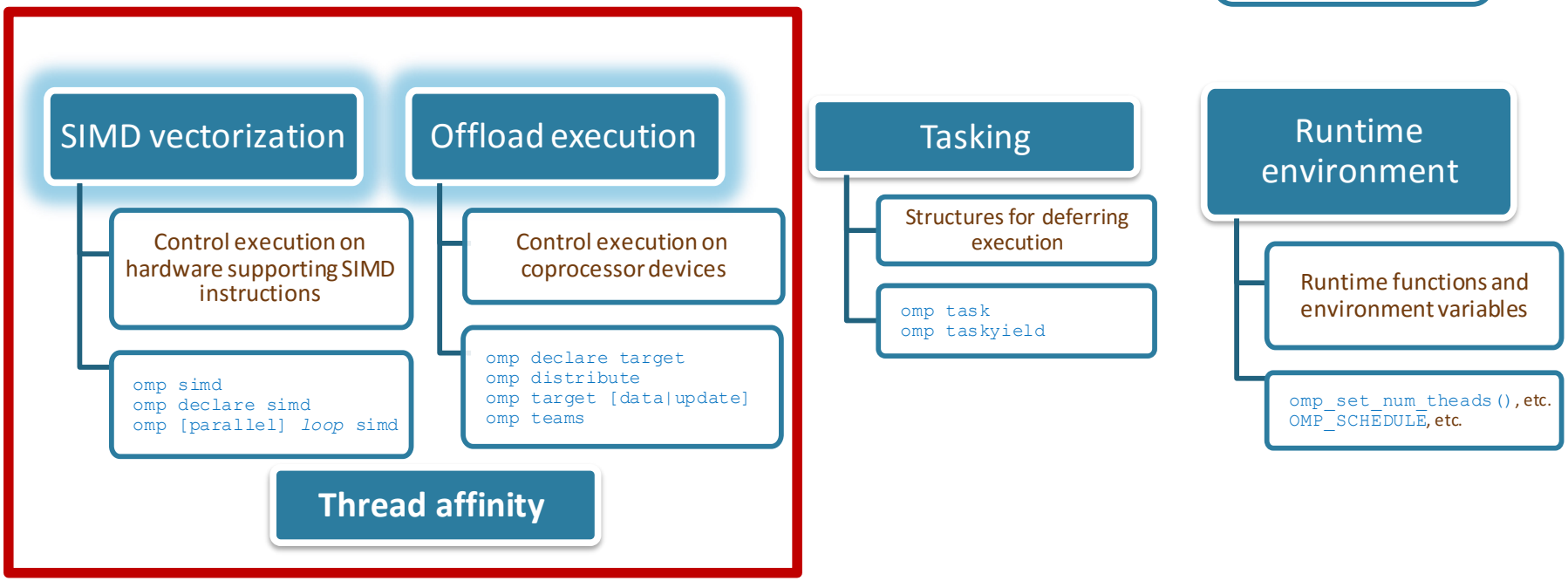
OpenMP


Sequential Region
Parallel Region
Sequential Region
Parallel Region
Sequential Region
Core 0
Core 1
Core 2
Core 3
Core 4
Thread 0
Thread 0
Thread 1
Thread 2
Thread 3
Thread 4
Thread 0
Thread 0
Thread 1
Thread 0
Thread Team
Thread Team
Time

OpenMP - Core elements

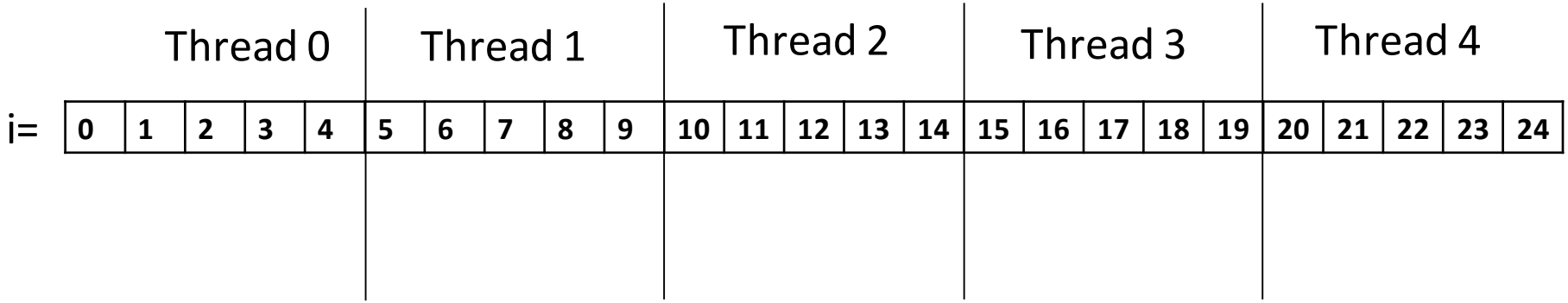


OpenMP 4.0



OpenMP Sample Program

```
N=25;
#pragma omp parallel for
for (i=0; i<N; i++)
    a[i] = a[i] + b;
```



OpenMP Sample Program

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <unistd.h>

int main() {
  int thid; char hn[600], i;
  double res, p[100];

  #pragma omp parallel
  {
    gethostname(hn,600);
    printf("hostname %s\n",hn);
  }
  res = 0;

  #pragma omp for
  for ( i = 0 ; i < 100 ; i++ ) {
    p[i] = i/0.855;
  }

  #pragma omp for
  for ( i = 0 ; i < 100 ; i++ ) {
    res = res + p[i];
  }

  printf("sum: %f", res);
}
```

Compiling and running an OpenMP application

#Build the application for Multicore Architecture (Xeon)

icc <*source-code*> -o <*omp_binary*> -fopenmp

#Launch the application on host

./*omp_binary*

Compiling and running an OpenMP application

```
export OMP_NUM_THREADS=10
./OMP-hello

hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
hello from hostname phi02.ncc.unesp.br
```

Launch the application on the Coprocessor from host

Agenda

- High Performance Computing (HPC)
- Data Parallelism
- Task Parallelism
- Basic Linear Algebra Subroutines (Blas)
- Intel Xeon and Intel Xeon Phi
- Intel Knights Landing
- Performance Evaluation

Blas

- BLAS (Basic Linear Algebra Subroutines)
  - is a specification that prescribes a set of low-level routines for performing common linear algebra operations such as:
    - vector addition,
    - scalar multiplication,
    - dot products,
    - linear combinations,
    - matrix multiplication.
  - They are the de facto standard low-level routines for linear algebra libraries;

MKL (Math Kernel Library)

- Intel's engineering, scientific, and financial math library

- Addresses:
  - Basic matrix-vector operations (BLAS)
  - Linear equation solvers (LAPACK, PARDISO, ISS)
  - Eigenvector/eigenvalue solvers (LAPACK)
  - Some quantum chemistry needs (GEMM from BLAS)
  - PDEs, signal processing, seismic, solid-state physics (FFTs)
  - General scientific, financial - vector transcendental functions (VML) and vector random number generators (VSL)

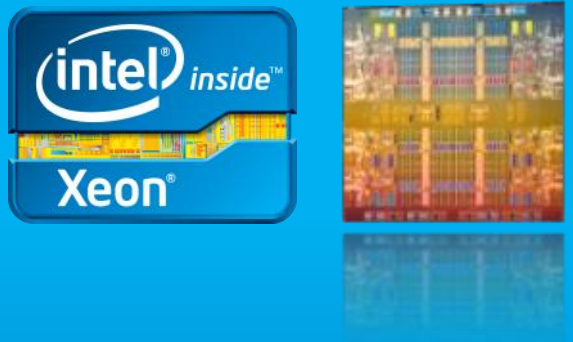
- MKL-DNN: Intended for accelerating deep learning frameworks
  - Building blocks for implementing convolutional neural networks with C and C++ interfaces.

Agenda

- High Performance Computing (HPC)
- Data Parallelism
- Task Parallelism
- Basic Linear Algebra Subroutines (Blas)
- Intel Xeon and Intel Xeon Phi
- Intel Knights Landing
- Performance Evaluation

Intel Xeon and Intel® Xeon Phi™ Overview

Intel® Multicore Architecture

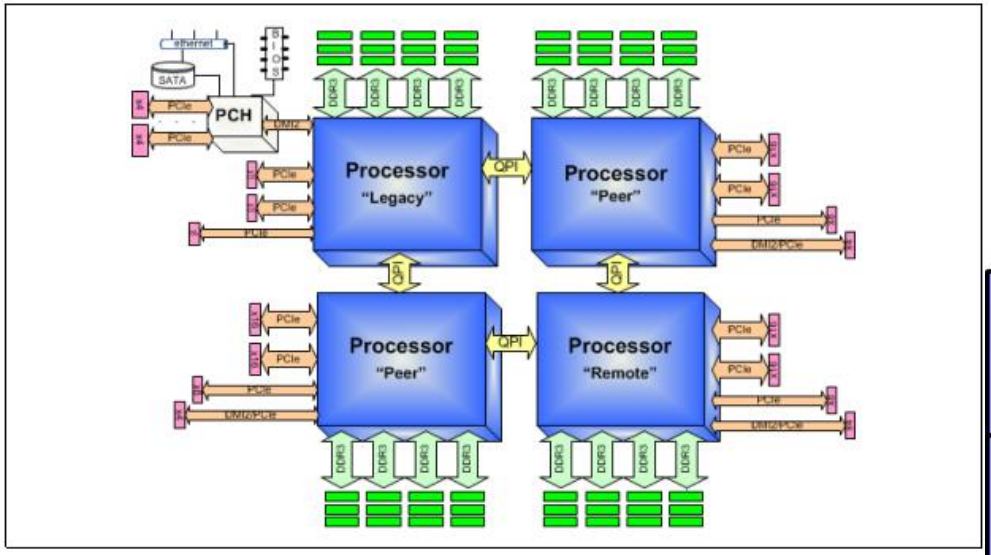


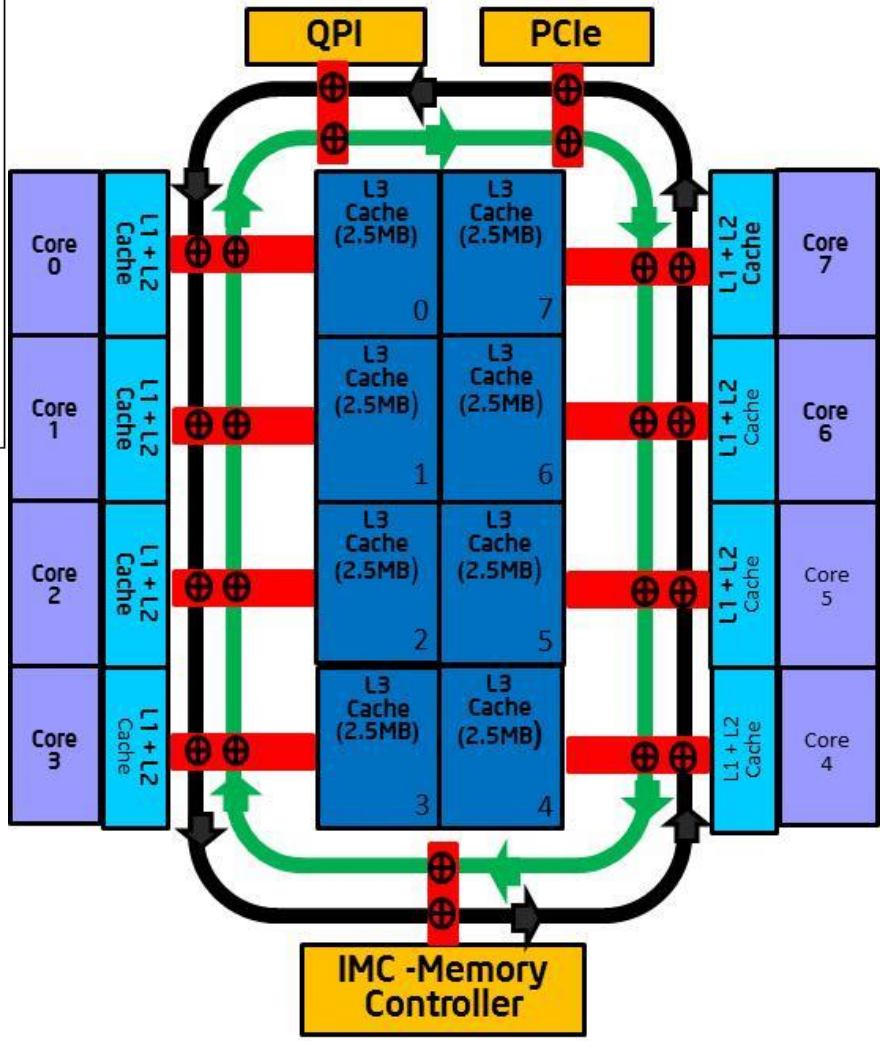
- Foundation of HPC Performance
- Suited for full scope of workloads
- Focus on fast single core/thread performance with “moderate” number of cores

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



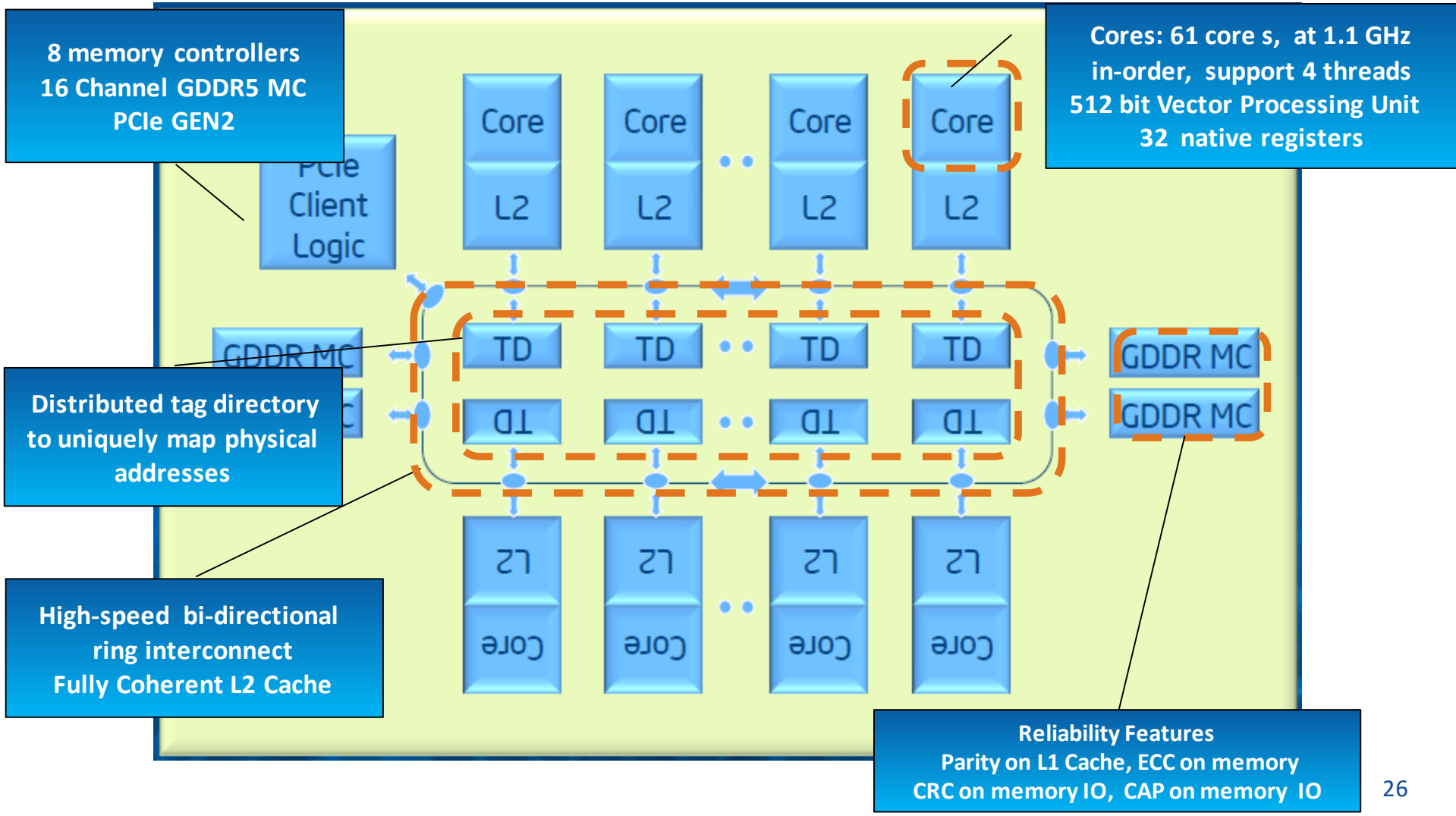


•Socket: mechanical component that provides mechanical and electrical connections between a microprocessor and a printed circuit board (PCB).

•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)

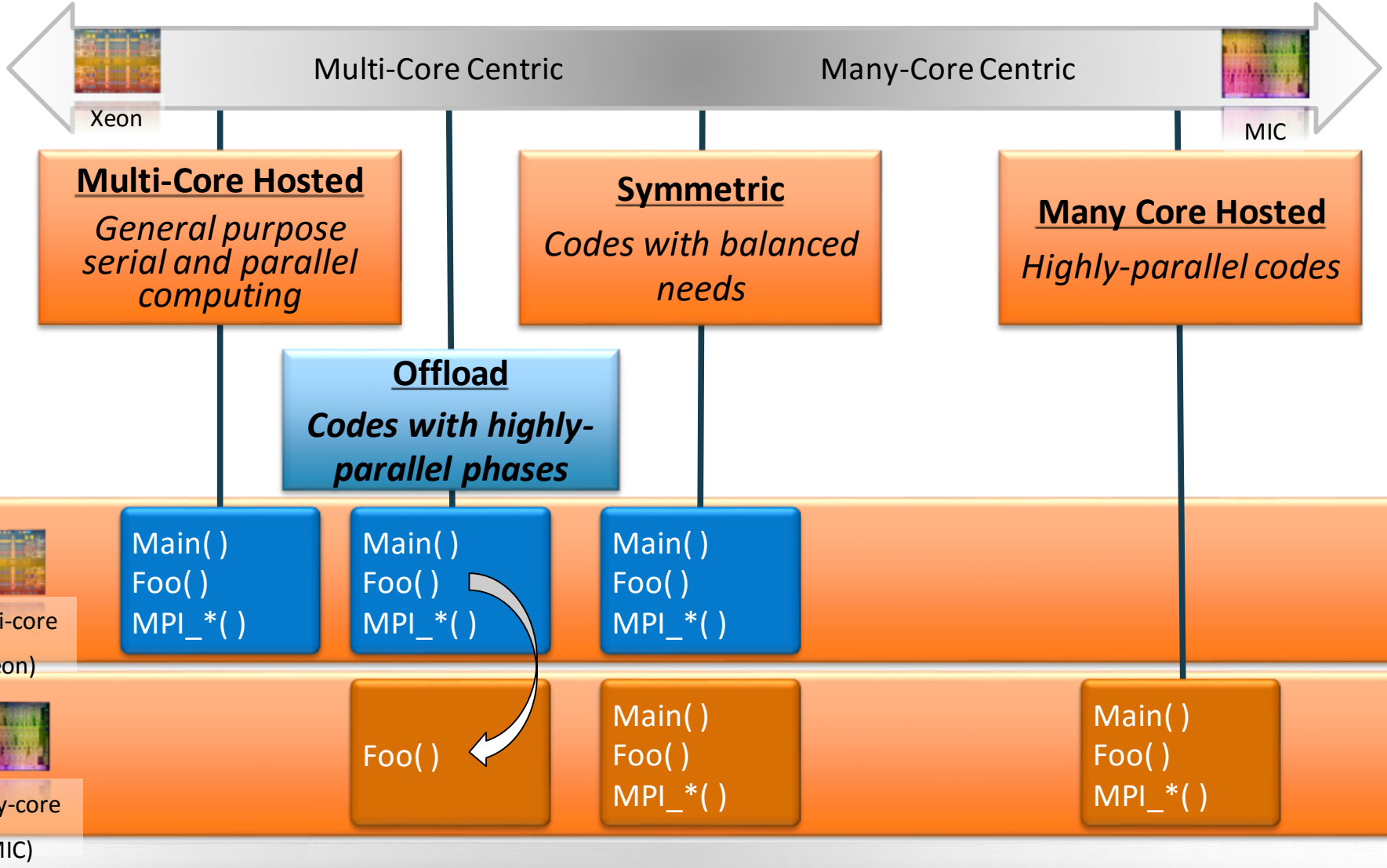


Intel® Xeon Phi™ Coprocessor Arch – System SW Perspective

- 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

Programming Models



Range of models to meet application needs

Agenda

- High Performance Computing (HPC)
- Data Parallelism
- Task Parallelism
- Basic Linear Algebra Subroutines (Blas)
- Intel Xeon and Intel Xeon Phi
- Intel Knights Landing
- Performance Evaluation

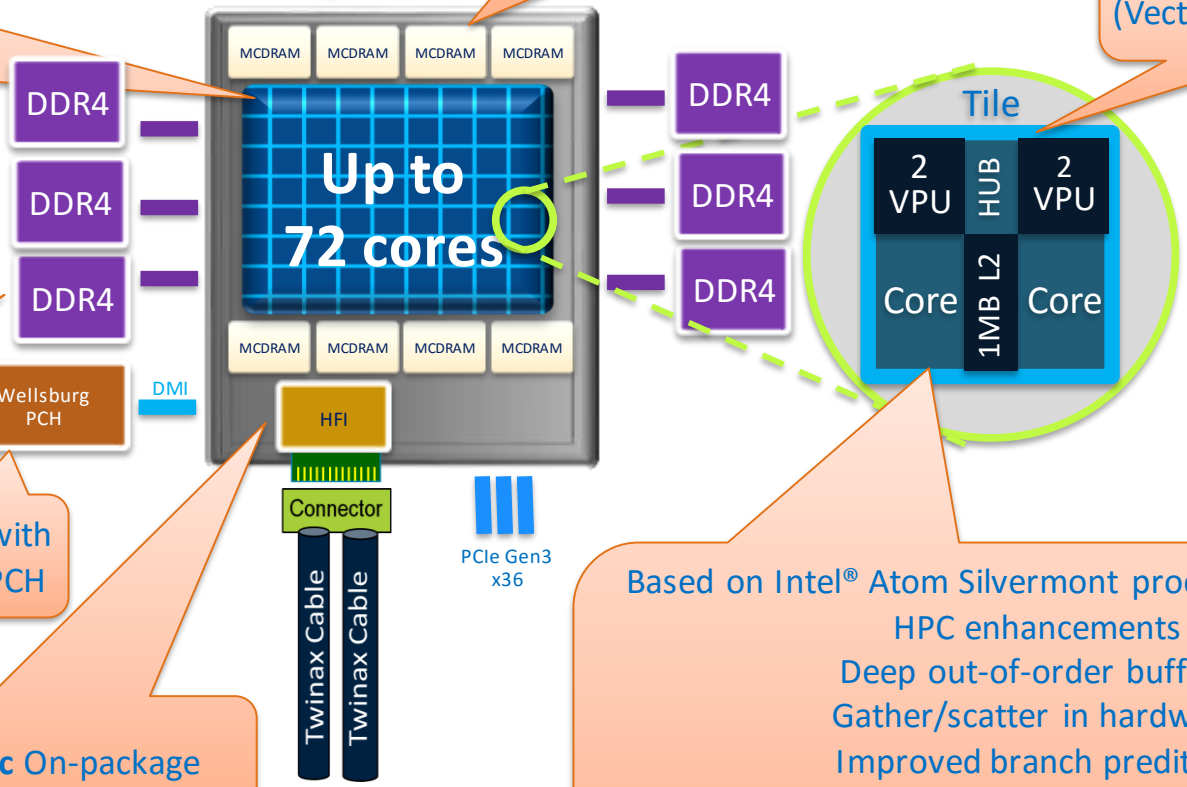
Knights Landing (KNL)

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

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

Up to 72 cores
2D mesh architecture



6 channels
DDR4
Up to
384GB

Common with
Grantley PCH

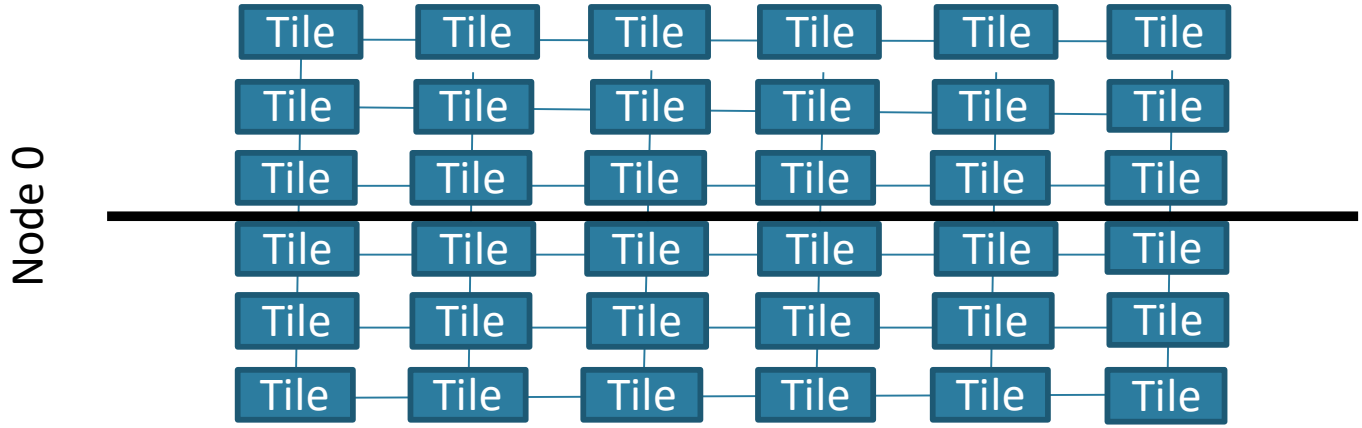
2 ports
Intel Omni-Path Fabric On-package
50 GB/s bi-directional

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

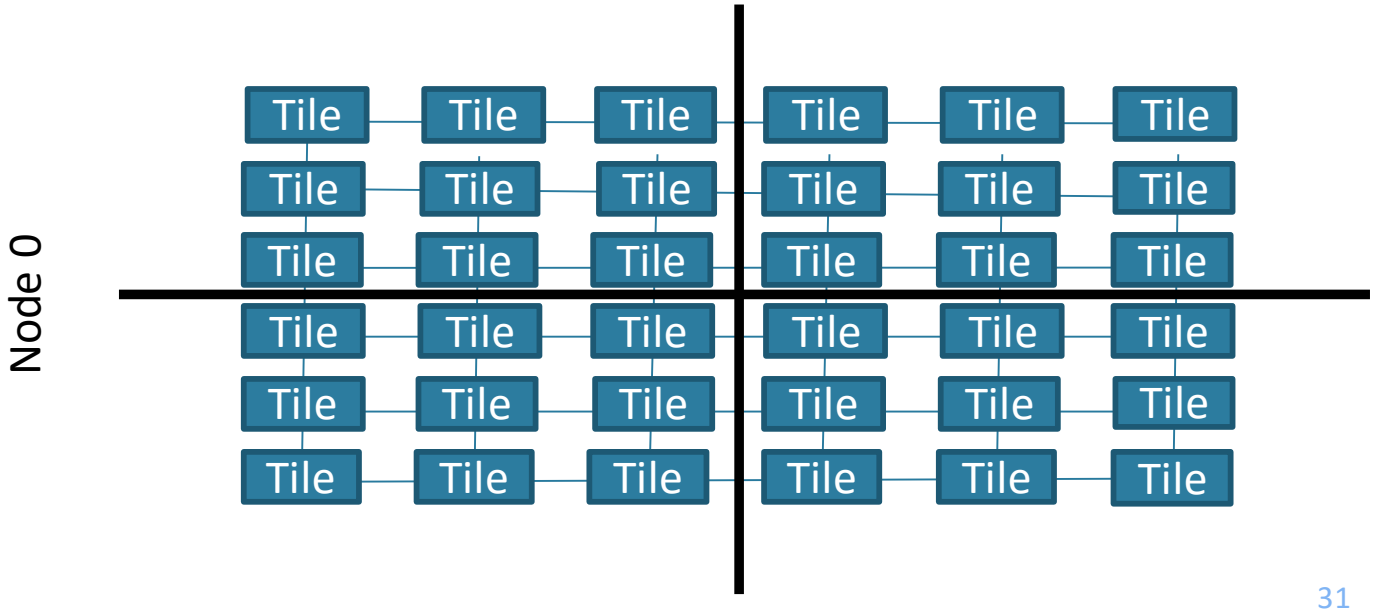
Cluster modes

One single space address

Hemisphere:
the tiles are divided into two parts called hemisphere



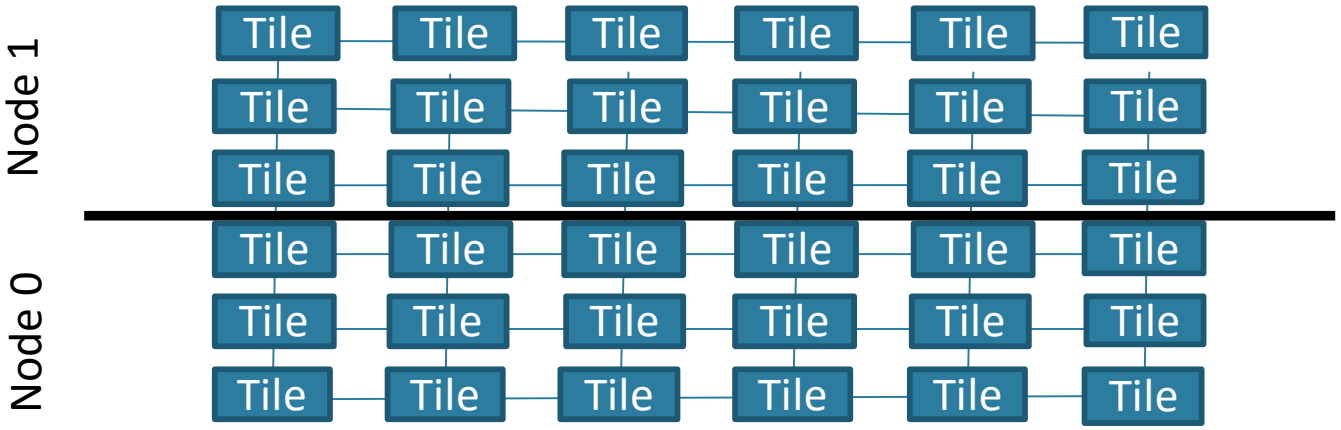
Quadrant:
tiles are divided into two parts called hemisphere or into four parts called qudrants



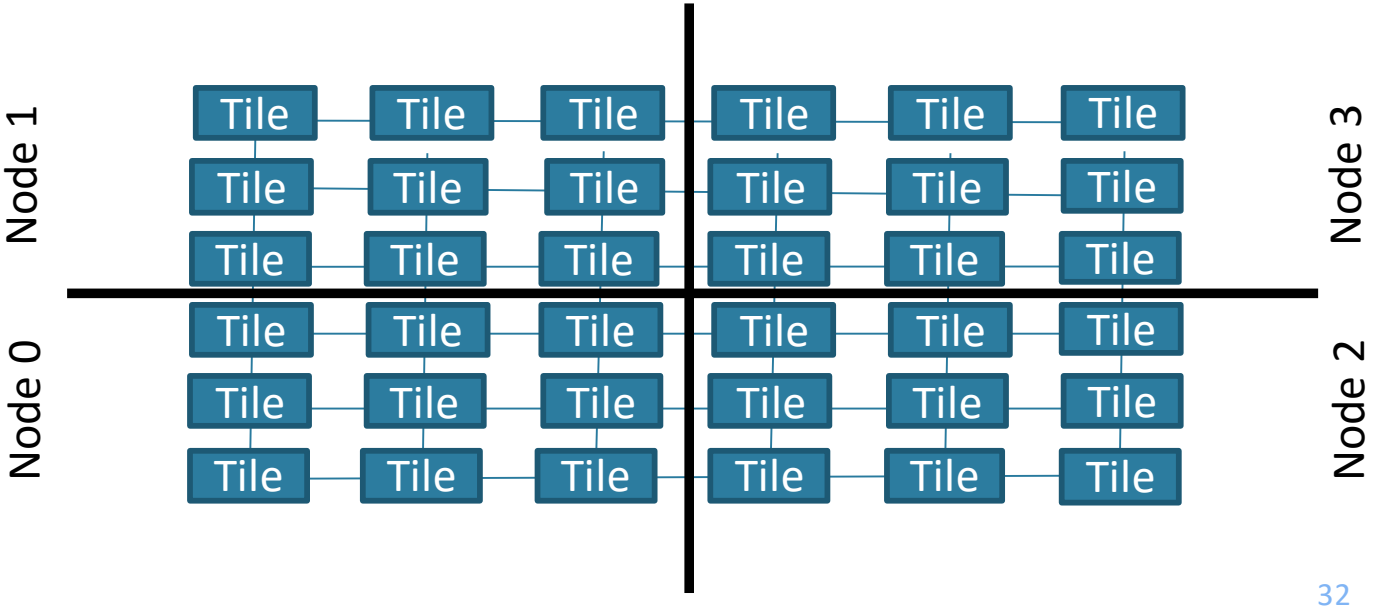
Cluster modes

Cache data are isolated in each sub numa domain

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

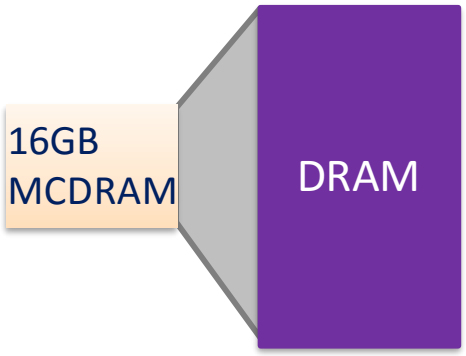


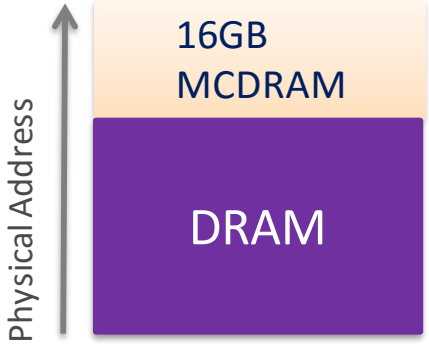
SNC-4:
the tiles are divided into two Numa Nodes

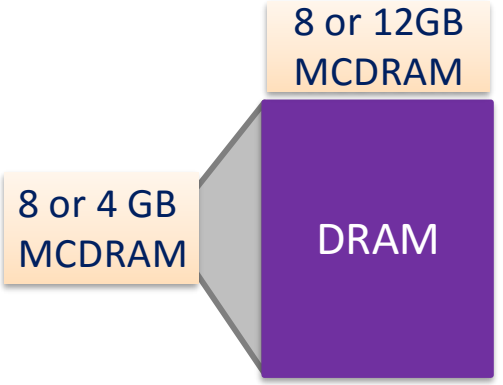


Integrated On-Package Memory Usage Models

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

Programming model

- Offload Over Fabric
  - Code region can be offloaded through the network to a KNL node
- OpenMP
- MPI/OpenMP
- Numa aware
  - Ex: numactl
- Vectorization (AVX512)
  - New instructions
  - More resources for automatic vectorization

Agenda

- High Performance Computing (HPC)
- Data Parallelism
- Task Parallelism
- Basic Linear Algebra Subroutines (Blas)
- Intel Xeon and Intel Xeon Phi
- Intel Knights Landing
- Performance Evaluation

Performance Evaluation of KNL

Matrix Transposition

- Tests scenarios:
  - Xeon;
  - KNL with MCDRAM;
  - KNL with DRAM;
- Resources:
  - Xeon
    - 2 sockets Intel Xeon 2699v3 @ 2.3GHz
    - 36 Cores - 72 threads
    - Main memory: 128 GB
  - Xeon Phi (KNL):
    - Intel Xeon Phi CPU 7250 @ 1.40GHz
    - 68 cores - 272 threads
    - Main memory: 192 GB
    - Configuration:
      - Cluster mode: SNC-4
      - MCDRAM: flat model.

"Multithreaded Transposition of Square Matrices with Common Code for Intel Xeon Processors and Intel Xeon Phi Coprocessors" - http://research.colfaxinternational.com/post/2013/08/12/Trans-7110.aspx

Performance Evaluation of KNL

- Tests Results:

Scenario	Execution Time (seconds)
Xeon	23.7
KNL - MCDram	16.1
KNL - Dram	36.7

- Vtune Analysis:
  - MCDRAM has Less L2 Miss Bound than Dram;
  - More SIMD Instructions can be executed;

	KNL with DRAM	KNL with MCDRAM
Memory Bandwidth:	41 GB/s	52 GB/s
Memory Latency		
L1 Hit Rate	97.9%	99.0%
L2 Hit Rate	28.5%	1.6%
L2 Miss Bound	16.9%	4.6%
SIMD Compute-to-L2 Access Ratio	1.767	164.000

Center of Excellence on Machine Learning @ NCC

- Period of project:
  - 2017 – 2019 (With the possibility of extension)

- Purpose:
  - Create a research center working on challenging projects related to Machine Learning;
  - Establish an international network of partners which are interested in collaborate and exchange knowledge in the area;
  - Train new motivated promising professionals in software development;

Open Positions

email to: cs-jobs[at]ncc unesp br

- High Performance Heterogeneous Computing
- Network Engineer/Architect
- Software-Defined Networking (SDN).
- Grid & Cloud Computing