C-DAC Four Days Technology Workshop

ON

Hybrid Computing – Coprocessors/Accelerators
Power-Aware Computing – Performance of Applications Kernels

hyPACK-2013
Mode 3 : Intel Xeon Phi Coprocessors

Lecture Topic :
Intel Xeon-Phi Coprocessor : Prog. Env

***Venue** : CMSD, UoHYD ; **Date** : October 15-18, 2013*

An Overview of Prog. Env on Intel Xeon-Phi

Lecture Outline

Following topics will be discussed

- Understanding of Intel Xeon-Phi Coprocessor Architecture
- Programming on Intel Xeon-Phi Coprocessor
- Performance Issues on Intel Xeon-Phi Coprocessor

Intel Xeon Phi - Coprocessor :

Multi-to-May-Core Processors : Background

How to run Programs faster ?

You require Super Computer

Era of Single - Multi-*to*-Many Core - Heterogeneous Computing

Sequential Computing

- Fetch/Store
- Compute

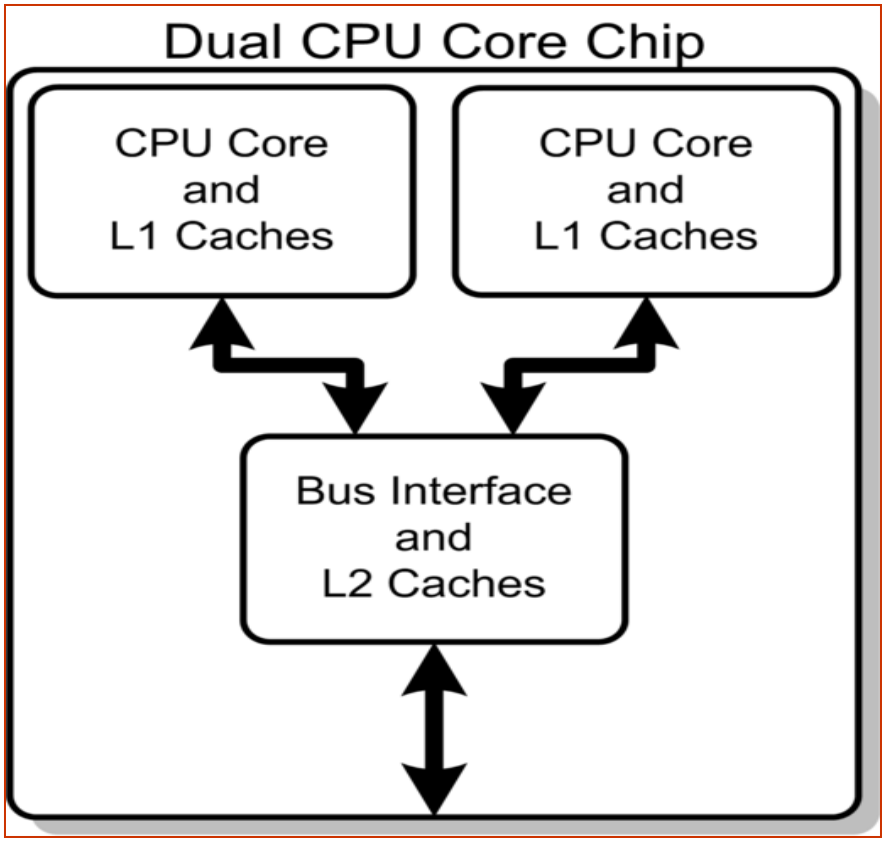
How to run Programs faster ?

- Fast Access of data
- Fast Processor
- More Memory to Manage data

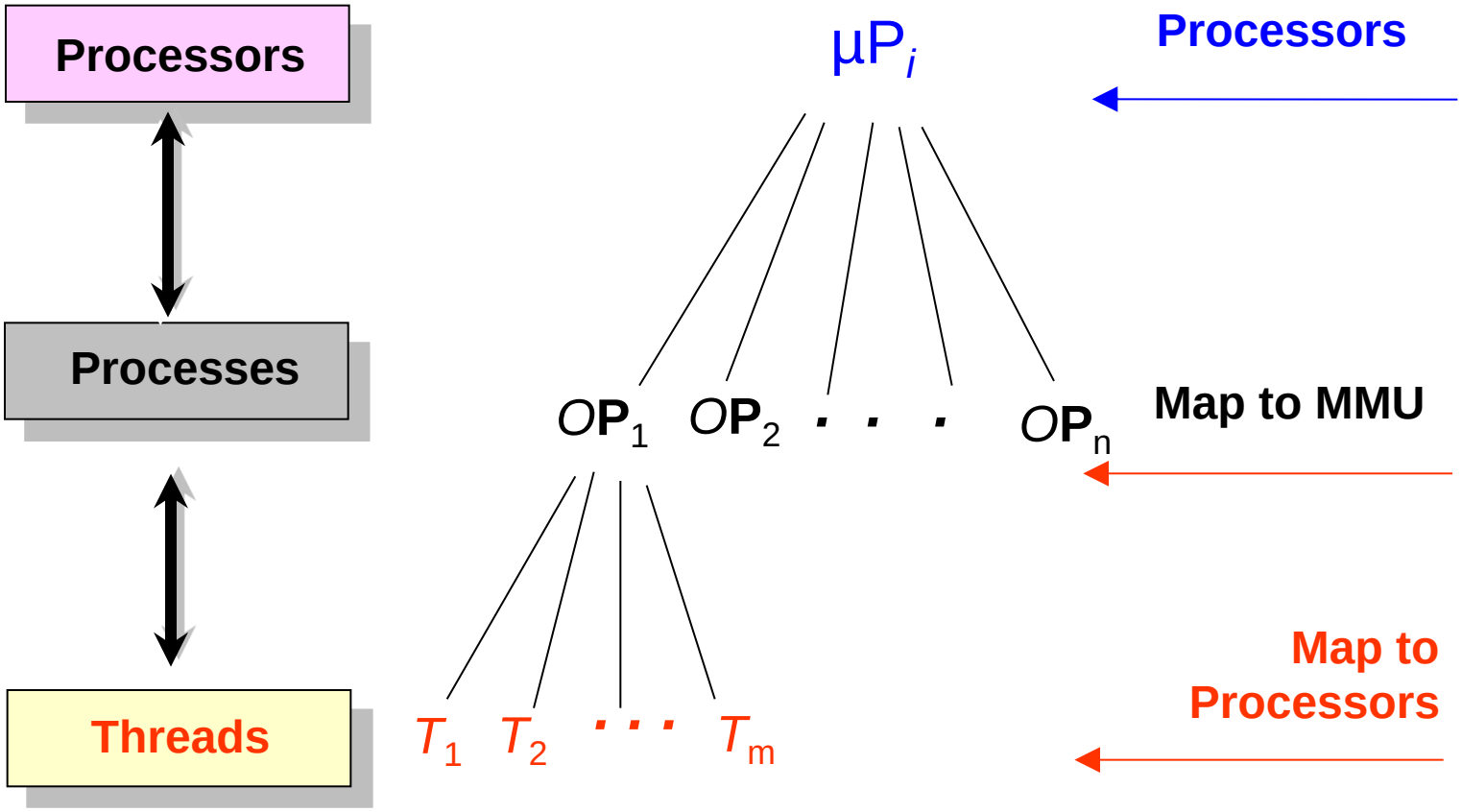
Dual Core Processor

Conceptual diagram of a dual-core CPU, with

- CPU-local Level 1 caches, and
- Shared, on-chip Level 2 caches



Relationship among Processors, Processes, &Threads

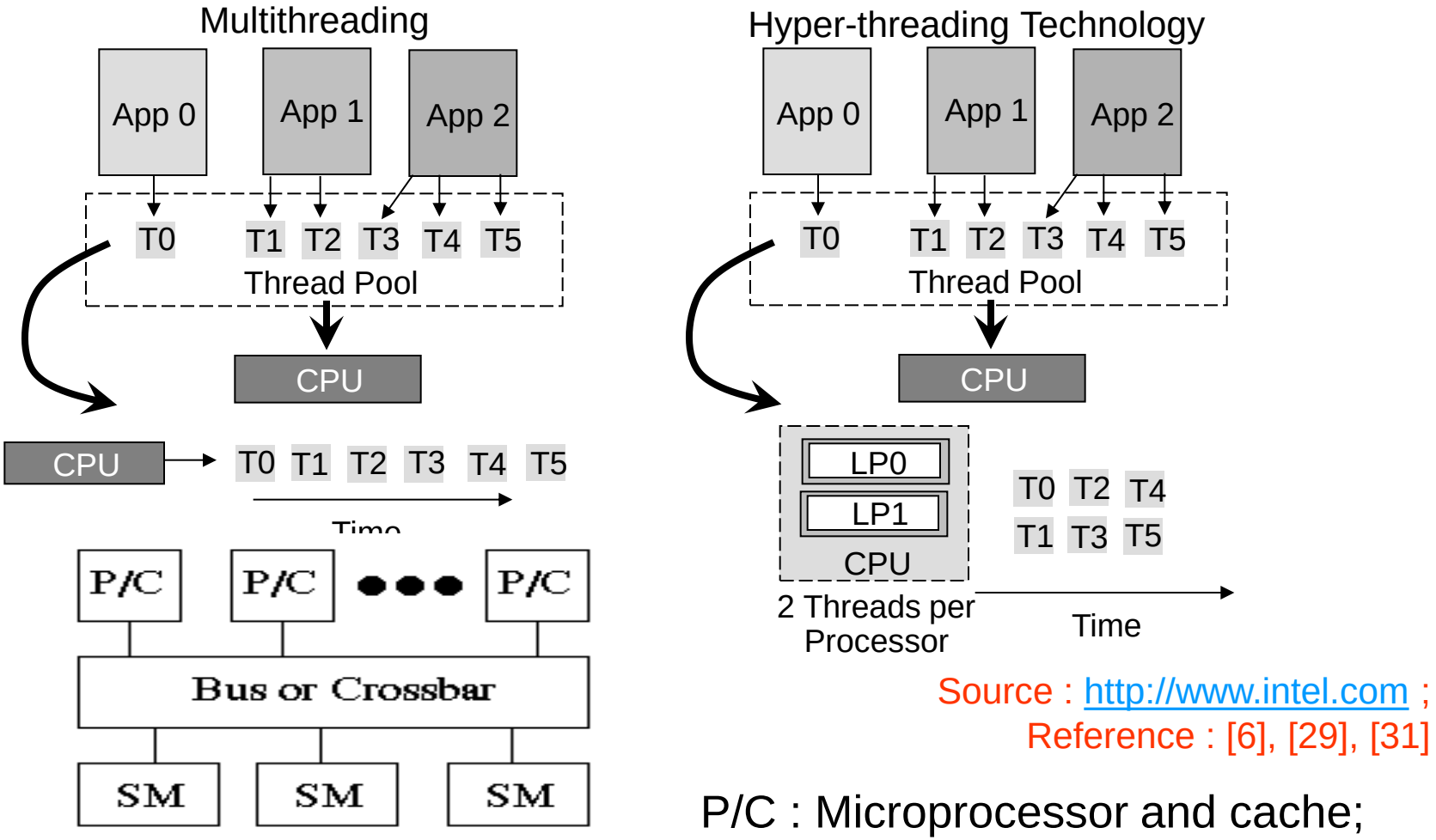


μP$_i$: Processor **OP**$_1$: Process T_1 : Thread **MMU** : Main Memory Unit

Source : Reference [4],[6], [7]

Multi-threaded Processing using Hyper-Threading Technology

- Time taken to process *n* threads on a single processor is significantly more than a single processor system with HT technology enabled.



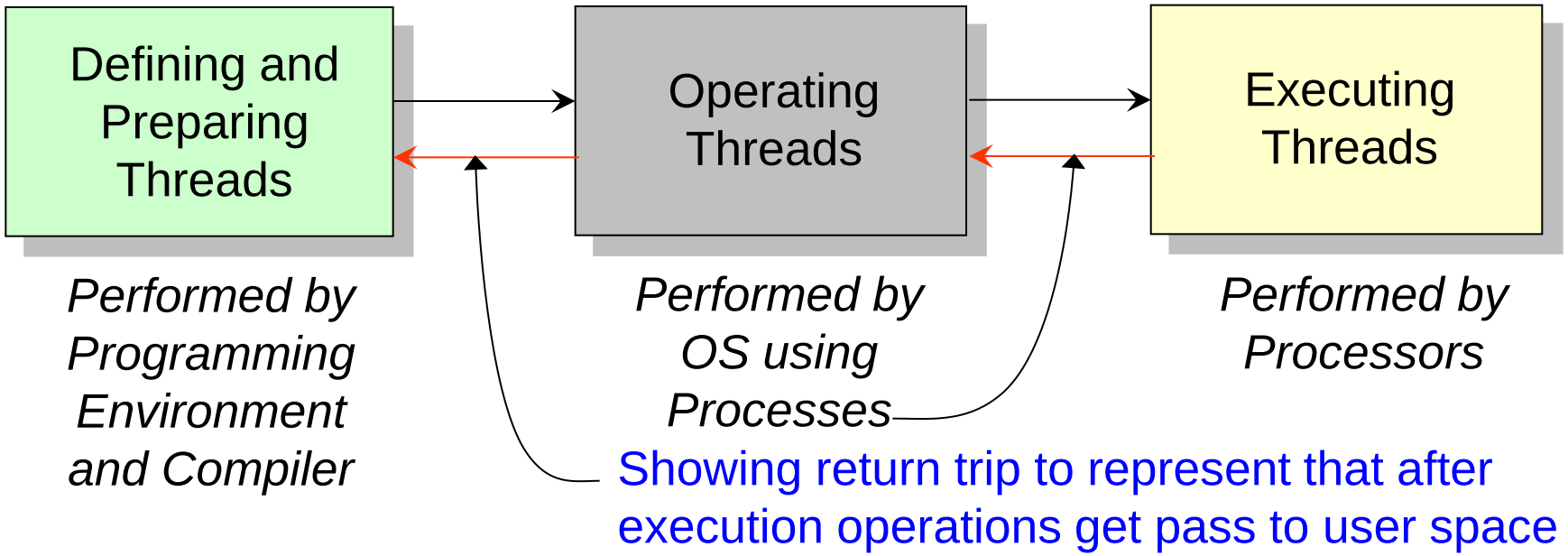
Source : http://www.intel.com ;
Reference : [6], [29], [31]

P/C : Microprocessor and cache;
SM : Shared memory

System View of Threads

Threads Above the Operating System

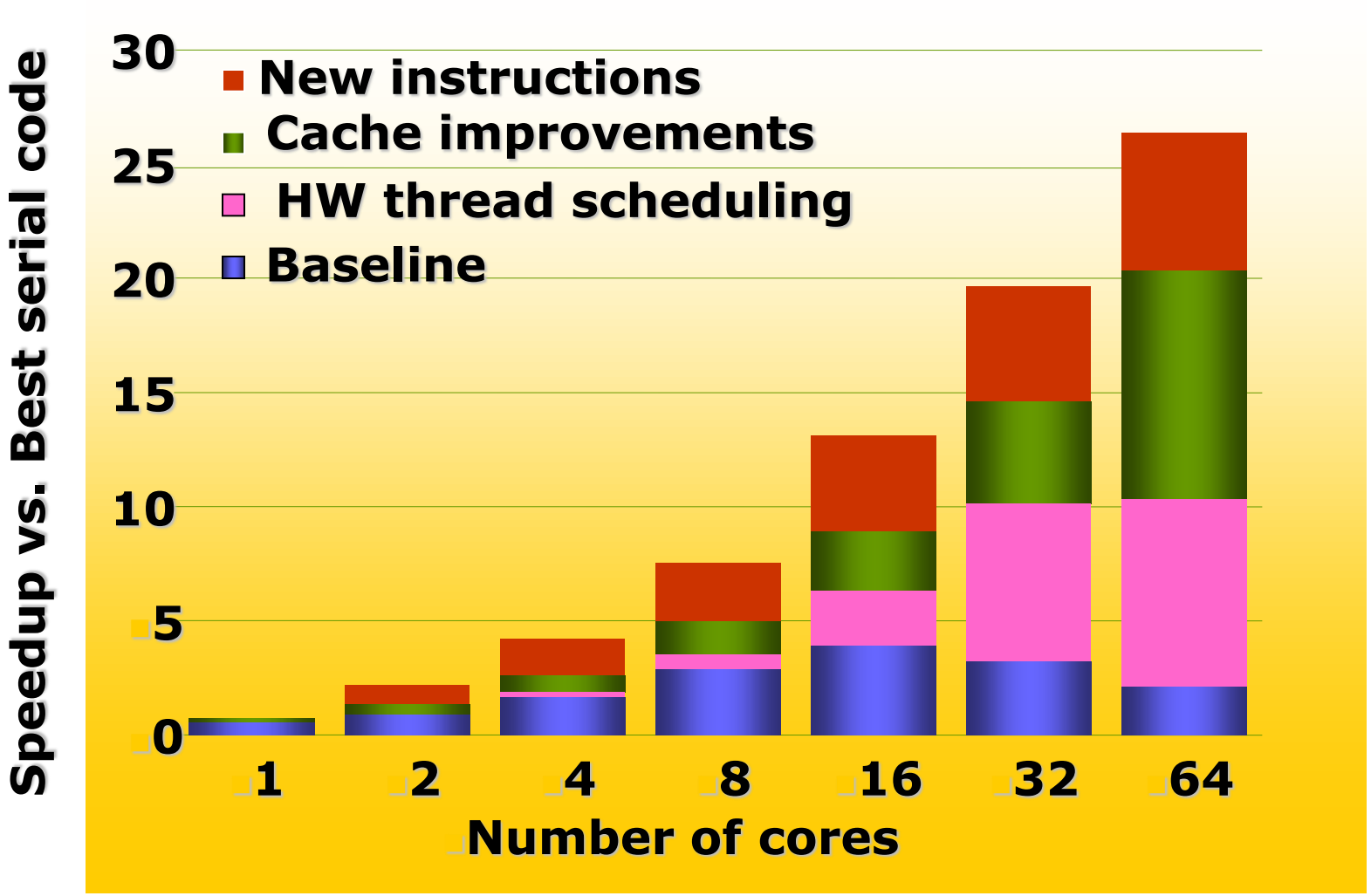
- Understand the problems - Face using the threads – Runtime Environment



Flow of Threads in an Execution Environment

Source : Reference [4],[6], [7]

Architecture-Algorithm Co-Design



Source : http://www.intel.com

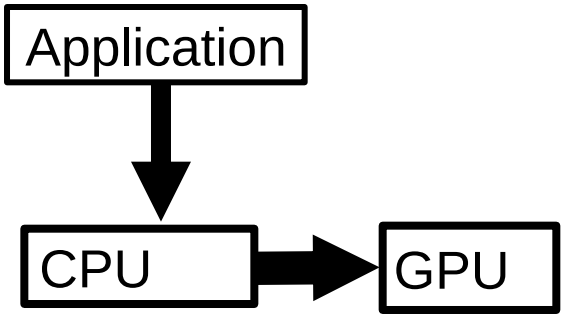
GPU Computing : Think in Parallel

- **Performance = parallel hardware**

+

scalable parallel program

- GPU Computing drives new applications
  - Reducing "Time to Discovery"
  - 100 x Speedup changes science & research methods
- New applications drive the future of GPUs
  - Drives new GPU capabilities
  - Drives hunger for more performance

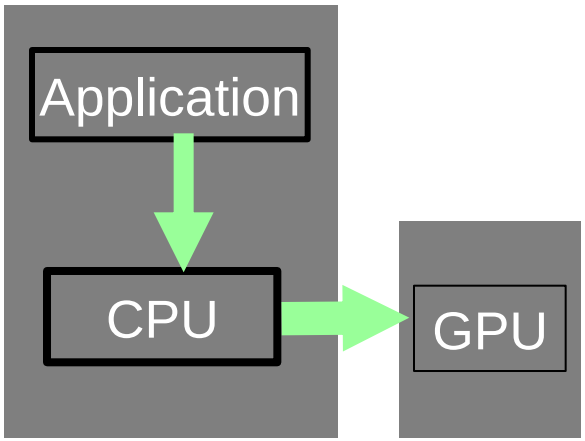


Source : NVIDIA, AMD,References

Programming paradigms-Challenges
Large scale data Computing – Current trends

Accelerators – GPU : Think in Parallel

- Speedups of 8 x to 30x are quite common for certain class of applications
- Best results when you "**Think Data Parallel**"
  - Design your algorithm for data-parallelism
  - Understand parallel algorithmic complexity and efficiency
  - Use data-parallel algorithmic primitives as building blocks

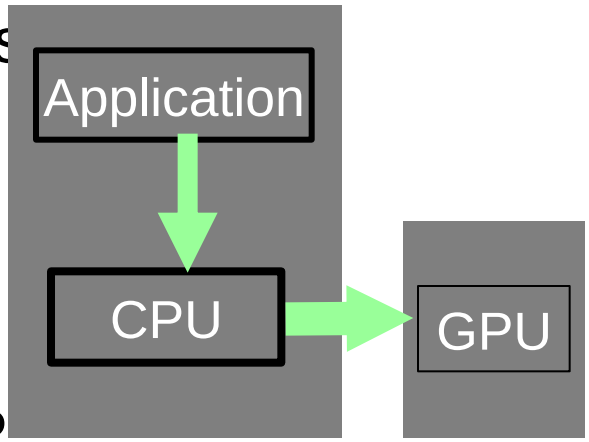


Source : NVIDIA, AMD, References

Accelerator Computing : Think in Parallel

Accelerators –GPU : Think in Parallel

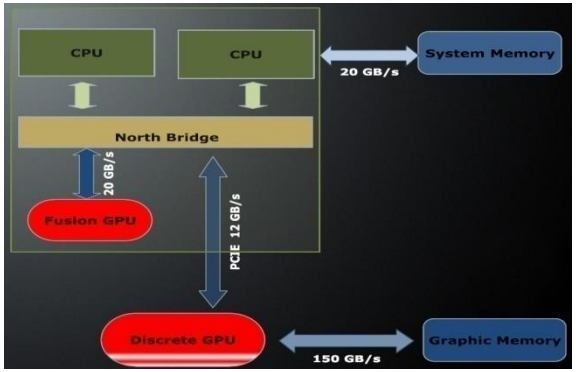
- Speedups of 8 x to 30x are quite common for certain class of applications
- The GPU is a data-parallel processor
  - Thousands of parallel threads
  - Thousands of data elements to proces
  - All data processed by the same program
  - SPMD computation model
  - Contrast with task parallelism and ILP



Source : NVIDIA, AMD, References

Systems with Accelerators

A set (one or more) of very simple execution units that can perform few operations (with respect to standard CPU) with very high efficiency. When combined with full featured CPU (CISC or RISC) can accelerate the “nominal” speed of a system.



CPU

ACC

Single thread perf.

throughput

CPU ACC.

Physical integration

CPU & ACC

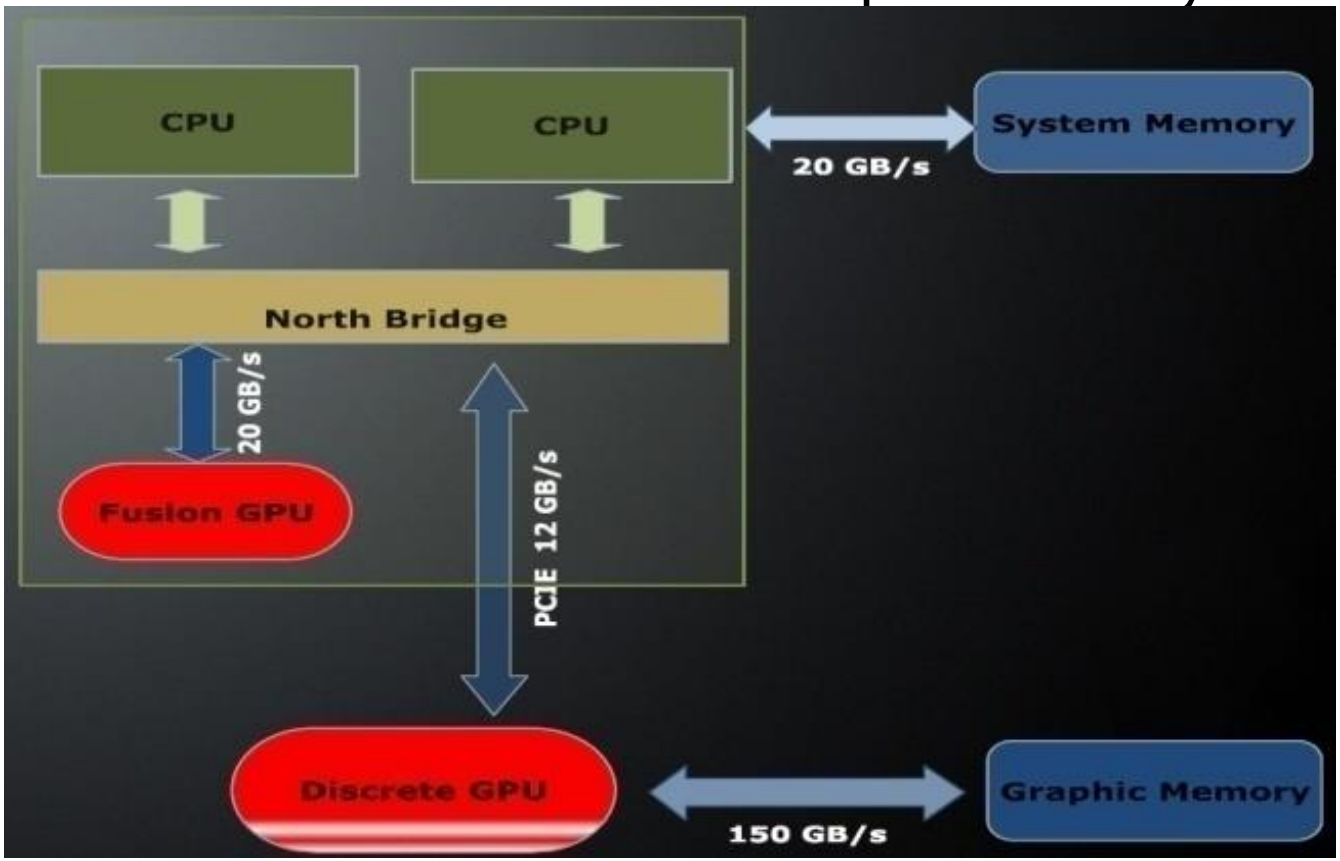
Architectural integration

CPU : Control Path & Data Path
ALU : Arithmetic and logic units
ACC: Accumulator
PC : Program counter
Micro-Instructions (Register transfer)

Source : NVIDIA, AMD, SGI, Intel, IBM Alter, Xilinux References

Systems with Accelerators

A set (one or more) of very simple execution units that can perform few operations (with respect to standard CPU) with very high efficiency. When combined with full featured CPU (CISC or RISC) can accelerate the “nominal” speed of a system.



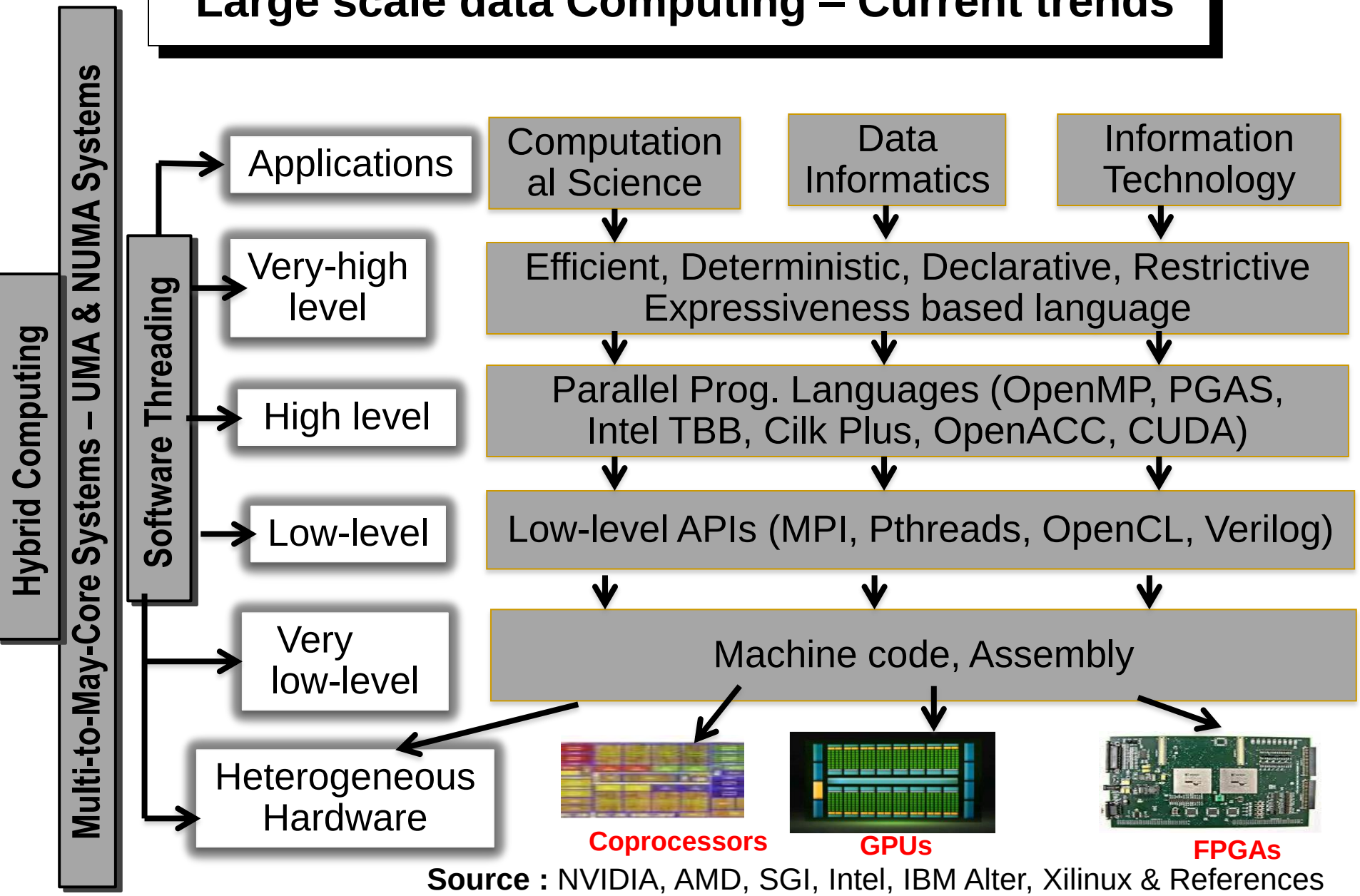
Source : NVIDIA, AMD, SGI, Intel, IBM Alter, Xilinux References

Multi-Core Systems with Accelerator Types (partial set)

- **FPGA**
  - Xilinx, Alter
- **GPU**
  - Nvidia (Kepler),
  - AMD Trinity APU
- **MIC (Intel Xeon-Phi)**
  - Intel Xeon-Phi (MIC)

Source : NVIDIA, AMD, SGI, Intel, IBM Alter, Xilinux References

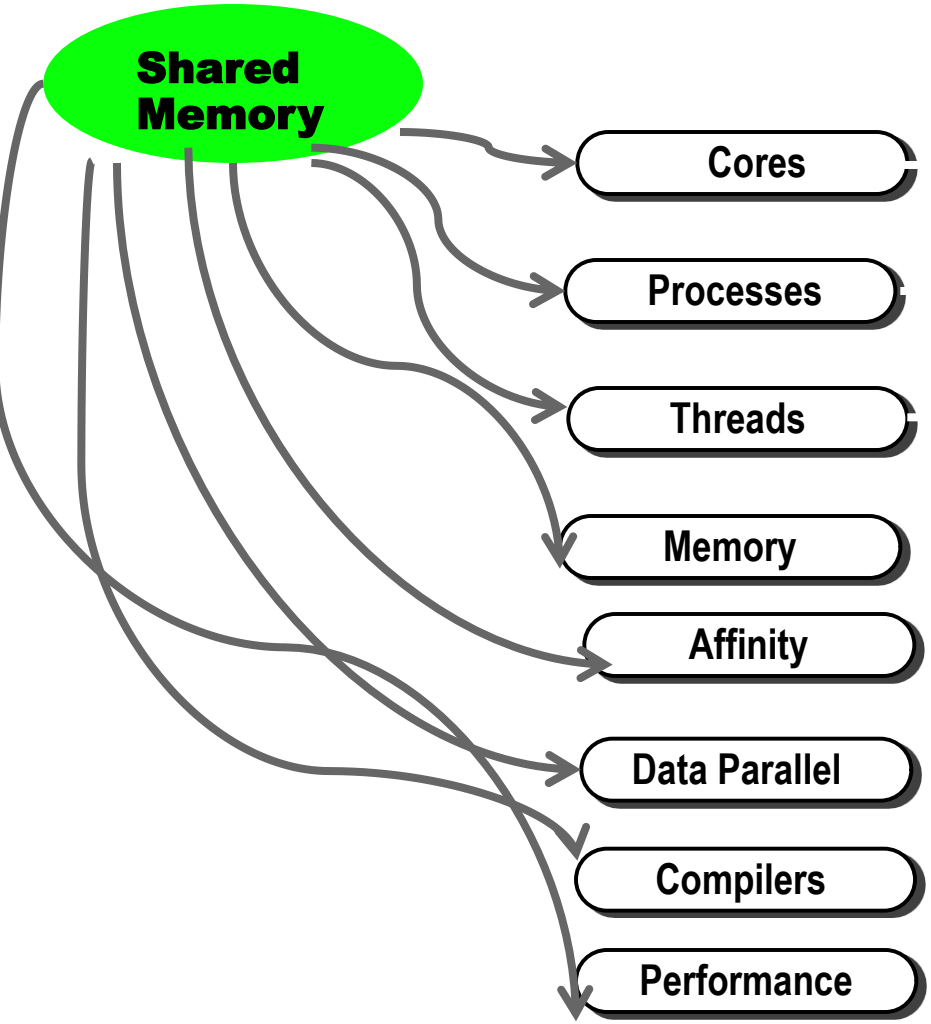
Programming paradigms-Challenges
Large scale data Computing – Current trends



Source : NVIDIA, AMD, SGI, Intel, IBM Alter, Xilinux & References

Programming paradigms-Challenges
Large scale data Computing – Current trends

Typical UMA /NUMA Computing Systems



System updates & Performance

Improvements
Quantify impacts prior to implementation

Is the machine working ?
Performance be expected

Small prototype available
What will be the performance of system ?

System available for measurement
What will be performance for App

Resources Availability
Application Scaling – Resources Avilable

.....

.....

Programming API - Multi-Core Systems with Accelerator Types (partial set)

- DO Parallel
- DO Synchronize
- Get Maximum of all values
- Give to Compiler

- DO TRANSFER from Host to Device
- Perform C omp. On Device

- Use as Linux OS
- DO TRANSFER from Host to Device

Parallel Prog. Languages (OpenMP, PGAS, Intel TBB, Cilk Plus, OpenACC, CUDA)

Programming with High Level APIs

Host : Multi-Core Systems OR ARM Multi-core Systems

CPU- Multi-Core Sys With

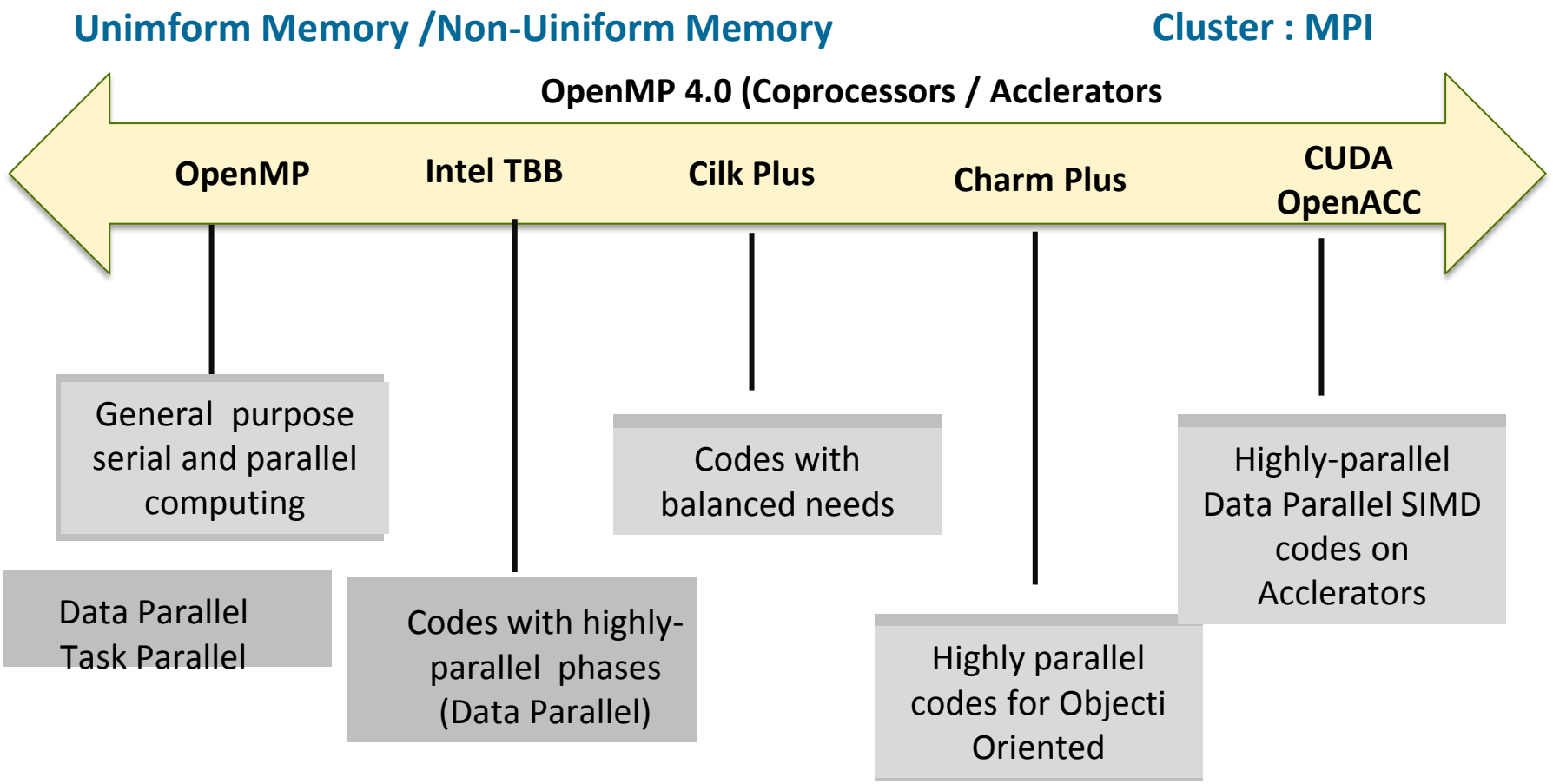
FPGA (Xilinx, Alter)

GPU - Nvidia (Kepler), AMD Trinity APU

MIC (Intel Xeon-Phi)

Source : NVIDIA, AMD, SGI, Intel, IBM Alter, Xilinux References

Prog. On Hybrid Computing Platforms

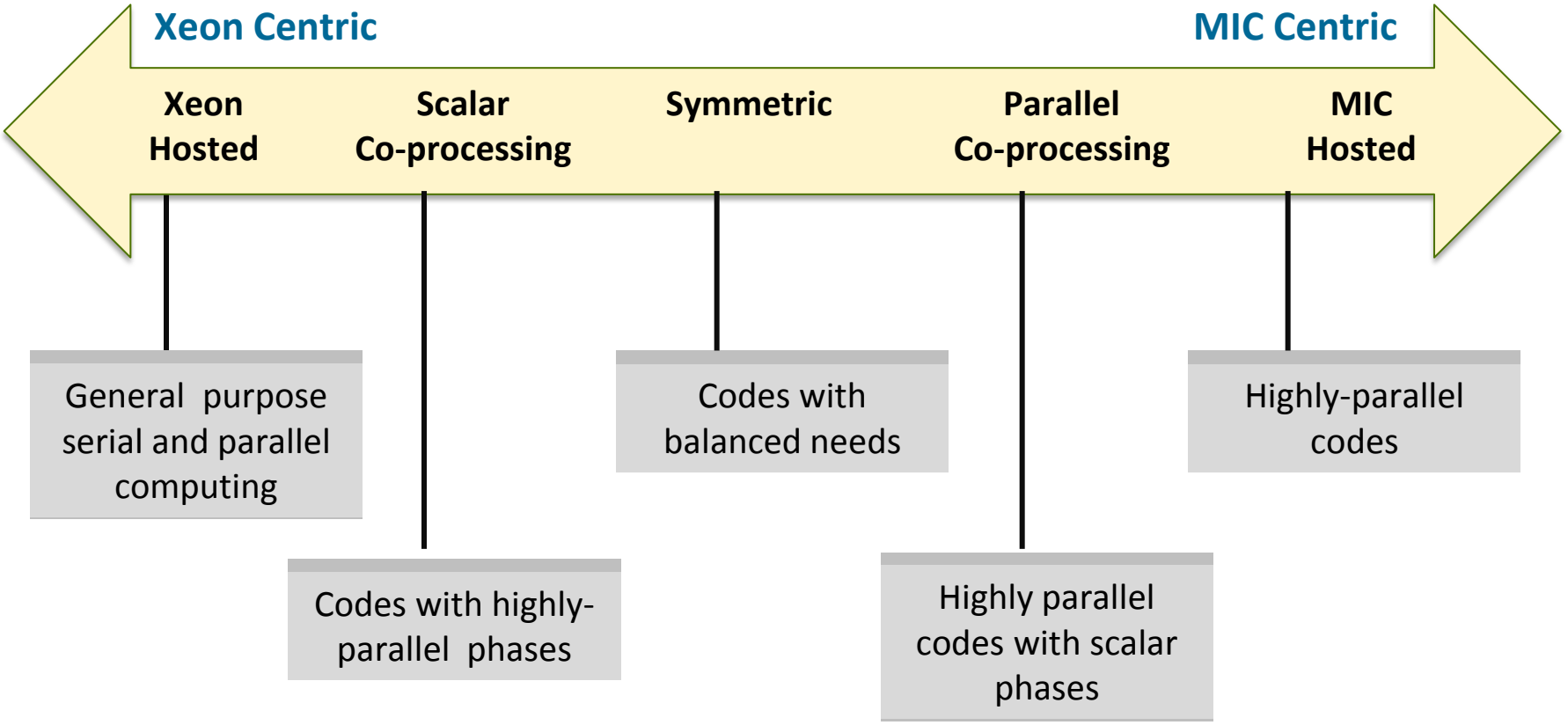


Source : References & NVIDIA, ntel Xeon-Phi; http://www.intel.com/

Intel Xeon Phi - Coprocessor :

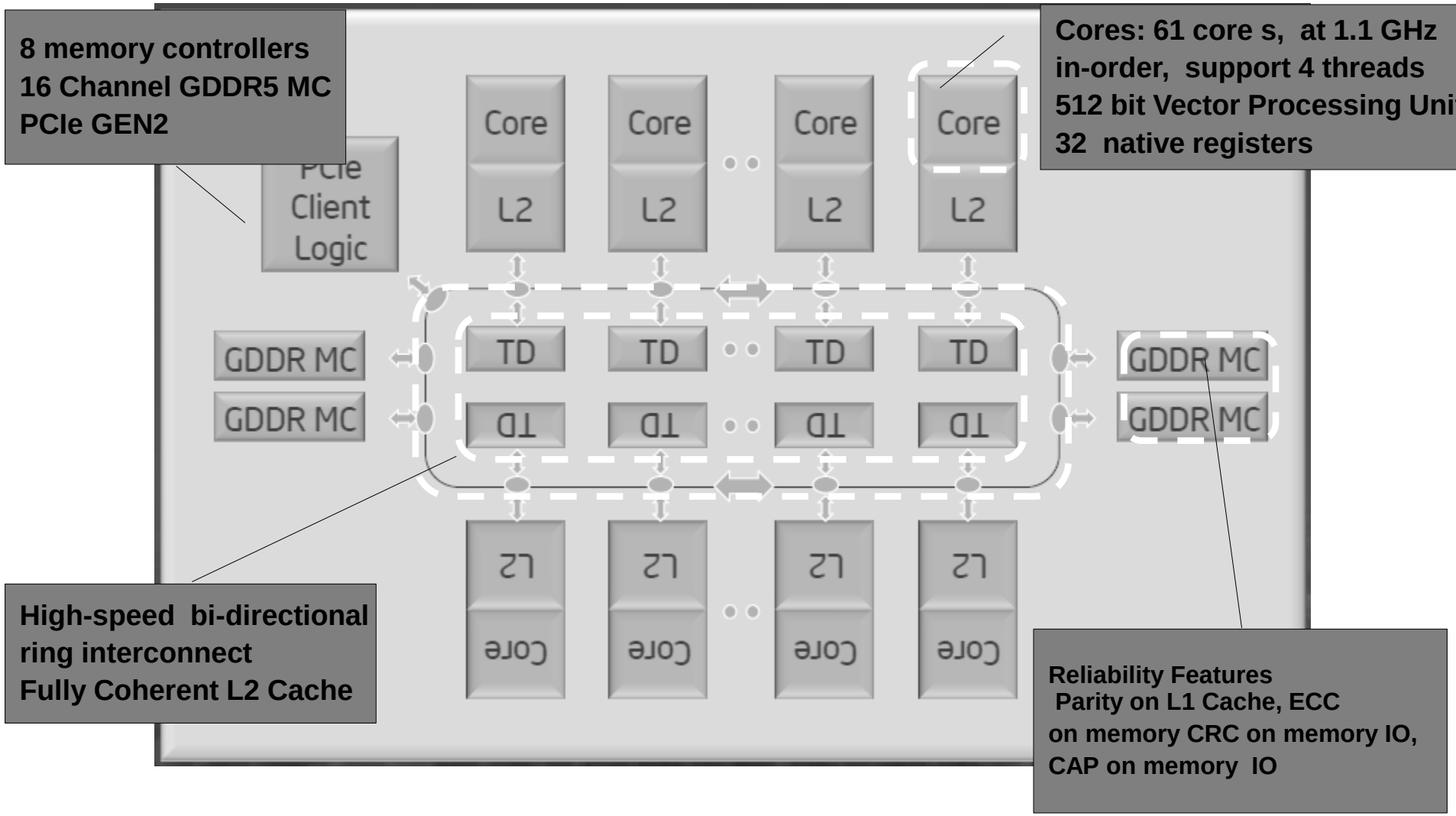
Architecture Overview

An Overview of Computing Scenrario

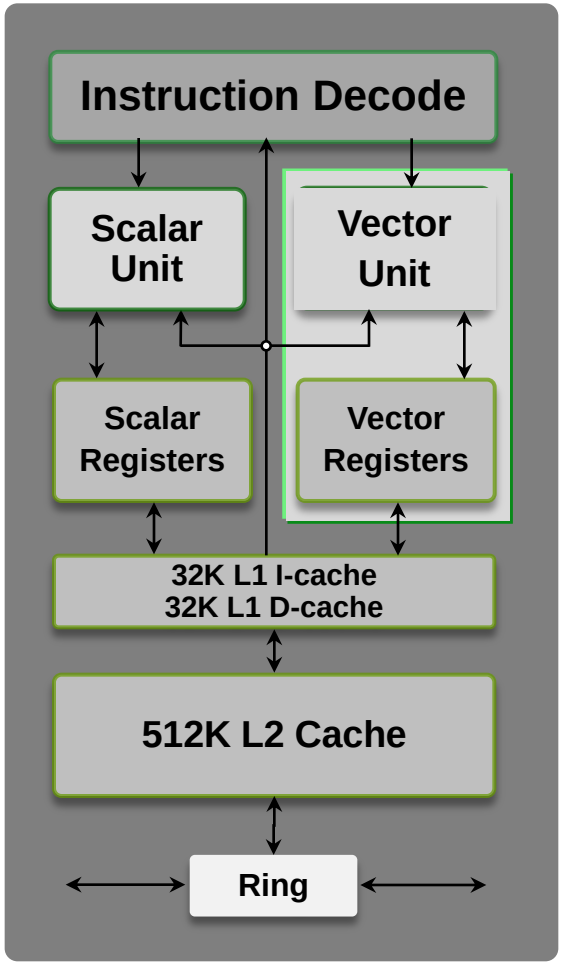


Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel® Xeon Phi™ Architecture Overview



Intel Xeon Phi Core Architecture Overview



- 60+ in-order, low power IA cores in a ring interconnect
- Two pipelines
  - Scalar Unit based on Pentium® processors
  - Dual issue with scalar instructions
  - Pipelined one-per-clock scalar throughput
- SIMD Vector Processing Engine
- 4 hardware threads per core
  - 4 clock latency, hidden by round-robin scheduling of threads
  - Cannot issue back to back inst in same thread
- Coherent 512KB L2 Cache per core

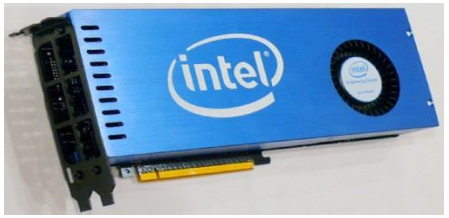
MIC Architecture

- MIC : Many Integrated Core
- Knight Corner co-processor
- Intel Xeon Phi co-processor
  - 22 nm technology
  - > 50 Intel Architecture cores
  - connected by a high performance on-die bi-directional interconnect.
  - I/O Bus: PCIe
  - Memory Type: GDDR5 and >2x bandwidth of KNF
  - Memory size: 8 GB GDDR5 memory technology
  - Peak performance: >1 TFLOP (DP)
  - Single Linux image per chip

Source : References & Intel Xeon-Phi; http://www.intel.com/

(Xeon Phi Hardware)

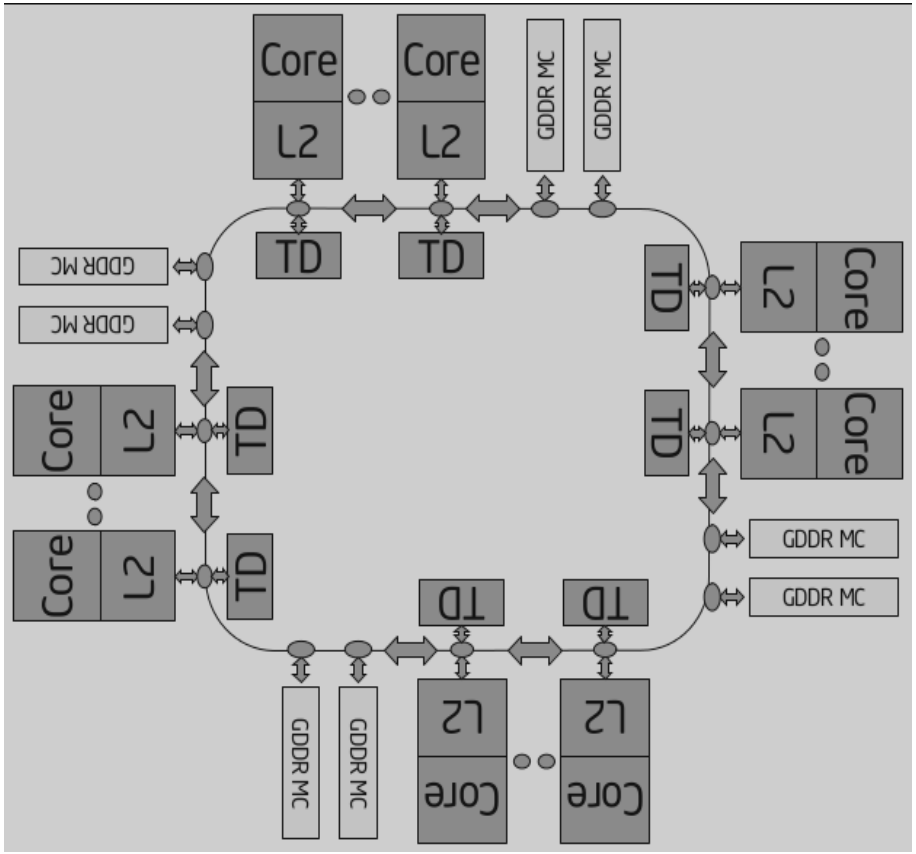
- X16 PCIe 2.0 card in Xeon host system
  - Up to 60 cores, bi-directional ring bus
  - 1-2GB GDDR5 main memory
- CPU cores
  - 1.2GHz, 4-way threading
  - 512-bit SIMD vector unit
  - 32KB L1, 256KB L2



Source : References & Intel Xeon-Phi; http://www.intel.com/

- Xeon-Phi coprocessor capacity 8GB;
  - processor :Xeon Phi 5110P; memory channel interface speed: 5.0 Giga Transfer/ Sec (GT/s); 8 memory controllers, each accessing two memory channels, used on co-processor

MIC Intel Xeon Phi Ring



- Each microprocessor core is a fully functional, in-order core capable of running IA instructions independently of the other cores.
- Hardware multi-threaded cores
- Each core can concurrently run instructions from four processes or threads.
- The Ring Interconnect connecting all the components together on the chip

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel® Xeon Phi™ Coprocessor Arch – System SW Perspective

- Large SMP UMA machine – a set of x86 cores to manage
  - 4 threads and 32KB L1I/D, 512KB L2 per core
  - Supports loadable kernel modules – we'll talk about one today
- Standard Linux kernel from kernel.org
  - 2.6.38 in the most recent release
  - Completely Fair Scheduler (CFS), VM subsystem, File I/O
- Virtual Ethernet driver– supports NFS mounts from Intel® Xeon Phi™ Coprocessor
- New vector register state per thread for Intel® IMCI
  - Supports "Device Not Available" for Lazy save/restore
- Different ABI – uses vector registers for passing floats
  - Still uses the x86_64 ABI for non-float parameter passing (rdi, rsi, rdx ..)

Intel Xeon Phi - Coprocessor :

Usage

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

- Deafault IP addresses ???•?? •?•??? , ???•?? •?•???, etc. are assigned to the attached Intel Xeon Phi coprocessors. The IP addresses of the attached coprocessors can be listed via the traditional **ifconfig** Linux program.

```
hypack-root@mic-0:~> /sbin/ifconfig
```

Further information can be obtained by running the **micinfo** program on the host.

```
hypack-root@mic-0:~>/sudo/opt/intel/mic/bin/micinfo
```

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

```
hypack-root@mic-0:~>/sudo/opt/intel/mic/bin/micinfo
System Info
 Host OS : Linux
 OS Version : 3.0.13-0.27-default
 Driver Version : 4346-16
 MPSS Version : 2.1.4346-16
 Host Physical Memory : 66056 MB
 ……
 Device No: 0, Device Name: Intel(R) Xeon Phi(TM) coprocessor
 ……
 Version
 ……..…
 Board
 …….
```

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

```
hypack-root@mic-0:~>/sudo/opt/intel/mic/bin/micinfo
 Device No: 0, Device Name: Intel(R) Xeon Phi(TM) coprocessor
 ……..
 Core
 ……..…
 Thermal
 ……..
 GGDR
 ……..
 Device No: 1, Device Name: Intel(R) Xeon Phi(TM) coprocessor
 ……..
 ……..
 ……..
```

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

Users can log in directly onto the Xeon Phi coprocessor via ssh. User can get basic information abbot Xeon-Phi by executing the following commands.

```
[hypack01@mic-0]$ ssh mic-0
[hypack01@mic-0]$ hostname
.....
[hypack01@mic-0]$ cat /etc/issue
 Intel MIC Platform Software Stack release 2.1
```

To get further information about the cores, memory etc. can be obtained from the virtual Linux /proc or /sys filesystems:

```
[weinberg@knf1-mic0 weinberg]$
[hypack01@mic-0]$ tail -n26 /proc/cpuinfo
.....
```

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

```
hypack-root@mic-0:~>/sudo/opt/intel/mic/bin/micinfo
 Device No: 0, Device Name: Intel(R) Xeon Phi(TM) coprocessor
 ……..
 Core
 ……..…
```

Intel Xeon Phi Coprocessor : Prog. Env & Performance Issues (In brief)

Intel Xeon-Phi : Shared Address Prog.

- **Rule of thumb :** An application must scale well past one hundred threads on Intel Xeon processors to profit from the possible higher parallel performance offered with e.g. the Intel Xeon Phi coprocessor.

- The scaling would profit from utilising the highly parallel capabilities of the MIC architecture, you should start to create a simple performance graph with a varying number of threads (from one up to the number of cores)

Xeon Phi : Programming Environment

- **Shared Address Space Programming (Offload, Native, Host)**

  OpenMP, Inetl TBB, Cilk Plus, Pthreads

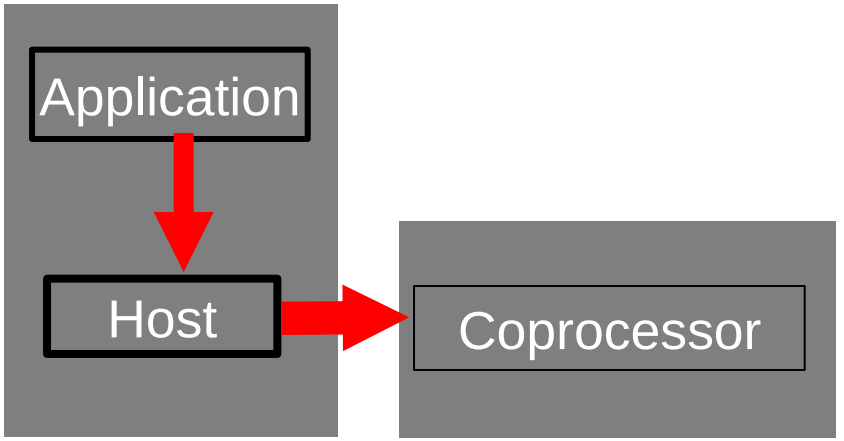
- **Message Passing Programming**

  (Offload – MIC Offload /Host Offload)

  (Symmetric & Coprocessor /Host)

- **Hybrid Programming**

  (MPI – OpenMP, MPI Cilk Plus
  MPI-Intel TBB)



Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : Shared Address Prog.

- **What we should know from programming point of view :** We treat the coprocessor as a 64-bit x86 **SMP-on-a-chip** with an high-speed bi-directional **ring** interconnect, (up to) **four** hardware threads per core and **512-bit SIMD** instructions.

- With the available number of cores, we have easily 200 hardware threads at hand on a single coprocessor.

Keys to Productive Performance on Intel ® MIC Architecture

- Choose the right Multi-core centric or Many-core centric model for your application
- Vectorize your application (today)
  - Use the Intel vectorizing compiler
- Parallelize your application (today)
  - with MPI (or other multi-process model)
  - With threads (via Intel ® Cilk TM Plus, OpenMP*, Intel ® Threading Building Blocks, Pthreads, etc.)
- Go asynchronous to overlap computation and communication

Source : References & Intel Xeon-Phi; http://www.intel.com/

Programming Challenges

- Alignment with the needs of the business / user / non-computer specialists / community and society
- Need to address the scalability issue: large scale data, high performance computing, automation, response time, rapid prototyping, and rapid time to production
- Need to effectively address (i) ever shortening cycle of obsolescence, (ii) heterogeneity and (iii) rapid changes in requirements
- Transform data from diverse sources into intelligence and deliver intelligence to right people/user/systems
- What about providing all this in a cost-effective manner?

Performance: Intel Xeon-Phi Coprocessor

- Vectorization is key for performance
  - Sandybridge, MIC, etc.
  - Compiler hints
  - Code restructuring
- Many-core nodes present scalability challenges
  - Memory contention
  - Memory size limitations

Source : References & Intel Xeon-Phi; http://www.intel.com/

Optimization Framework

A collection of methodology and tools that enable the developers to express parallelism for Multicore and Manycore Computing

Objective: Turning unoptimized program into a scalable, highly parallel application on multicore and manycore architecture

Step 1: Leverage Optimized Tools, Library

Step 2: Scalar, Serial Optimization /Memory Access

Step 3: Vectorization & Compiler

Step 4: Parallelization

Step 5: Scale from Multicore to Manycore

Source : References & Intel Xeon-Phi; http://www.intel.com/

Porting on MIC

Pros:

- Compilation with an additional Intel compiler flag (`-mmic`);
- Scalability tests: fast and smooth;
- Quick analysis with Intel tools (VtuneT, Itac Intel Trace Analyzer and Collector;
- Porting time: one day with validation of the numerical result;
- expert developer of FARM, with good knowledge of the Intel Compiler, But with only a basic knowledge of MIC.
- Best scalability with OpenMP and Hybrid.

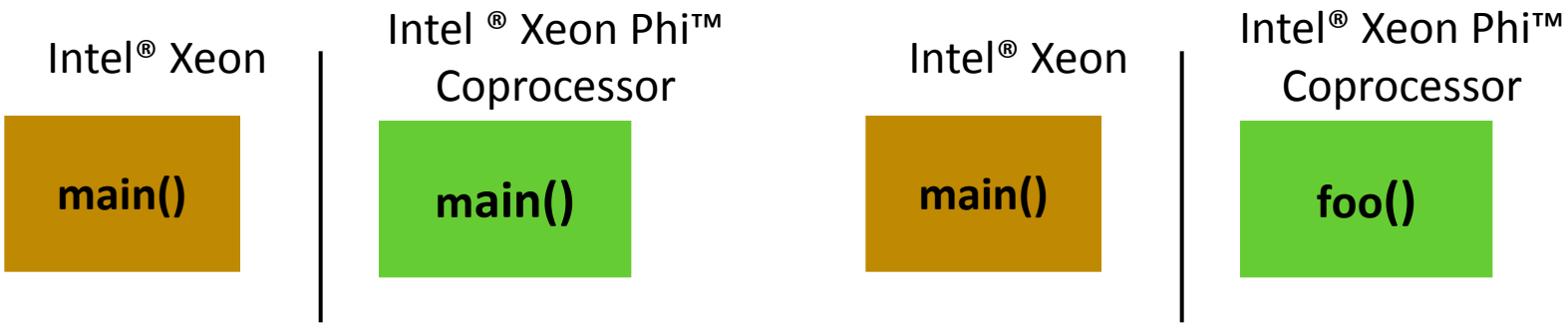
Source : References & Intel Xeon-Phi; http://www.intel.com/

Issues to be addressed

- `MPI Init` routine problem: increasing CPU time for increasing number of processes; Same problem when using two MICs together;
- Detailed analysis of OpenMP threads & thread affinity and Memory available per thread
- Execution time depends strongly from code vectorization, so compiler vectorization for data parallel and task parallel constructs
- code re-structure and memory access pattern are a key point to have a vectorizable satisfactory overall Performances.

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon & Xeon Phi : Execution Modes



Native

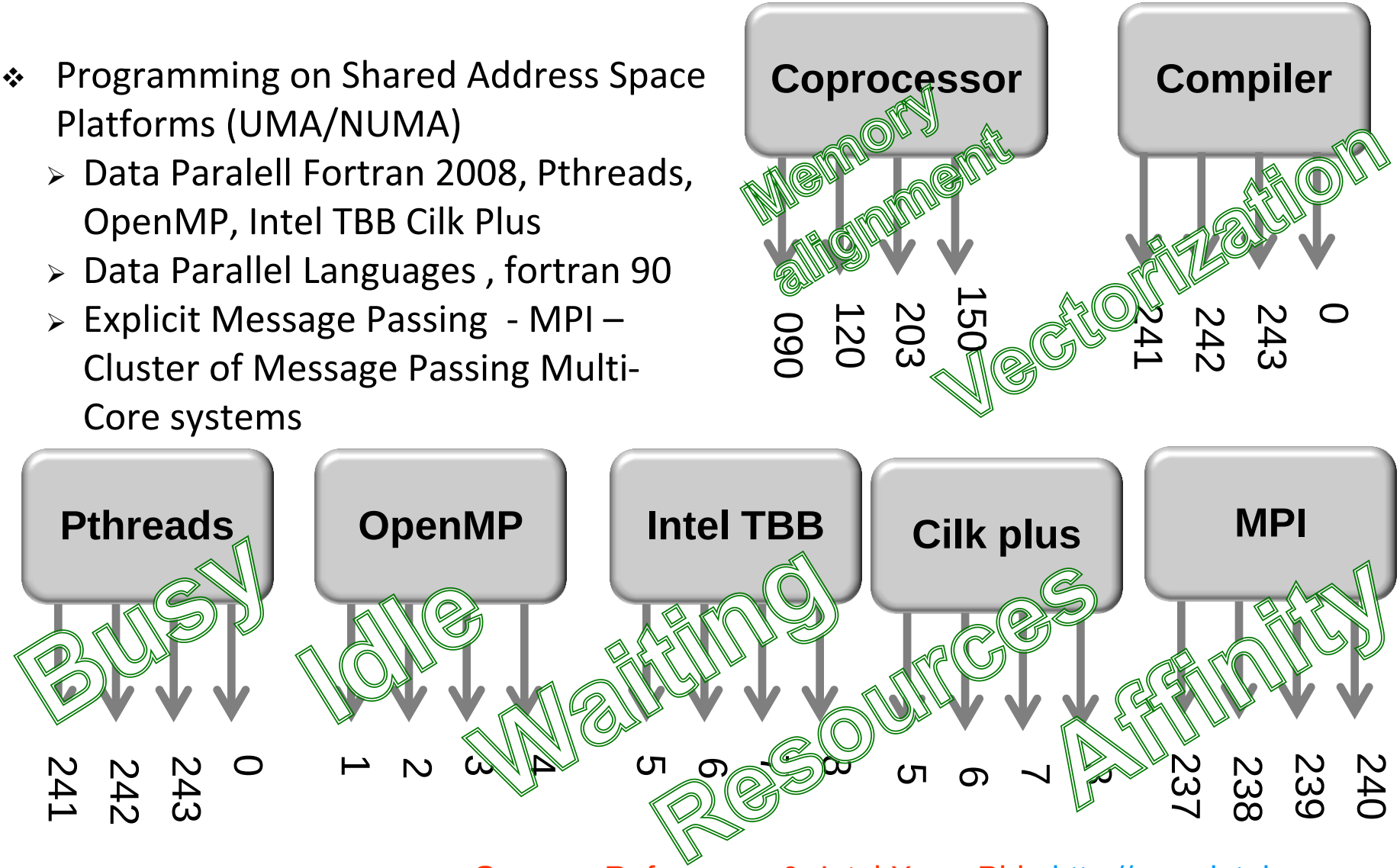
- Card is an SMP machine running Linux
- Separate executables run on both MIC and Xeon
  - e.g. Standalone MPI applications
- No source code modifications most of the time
  - Recompile code for Xeon Phi™ Coprocessor
- Autonomous Compute Node (ACN)

Offload

- "main" runs on Xeon
- Parts of code are offloaded to MIC
- Code that can be
  - Multi-threaded, highly parallel
  - Vectorizable
  - Benefit from large memory BW
- Compiler Assisted vs. Automatic
  - #pragma offload (…)

Intel Xeon –Phi Programming Paradigms

- Programming on Shared Address Space Platforms (UMA/NUMA)
  - Data Paralell Fortran 2008, Pthreads, OpenMP, Intel TBB Cilk Plus
  - Data Parallel Languages , fortran 90
  - Explicit Message Passing - MPI – Cluster of Message Passing Multi-Core systems

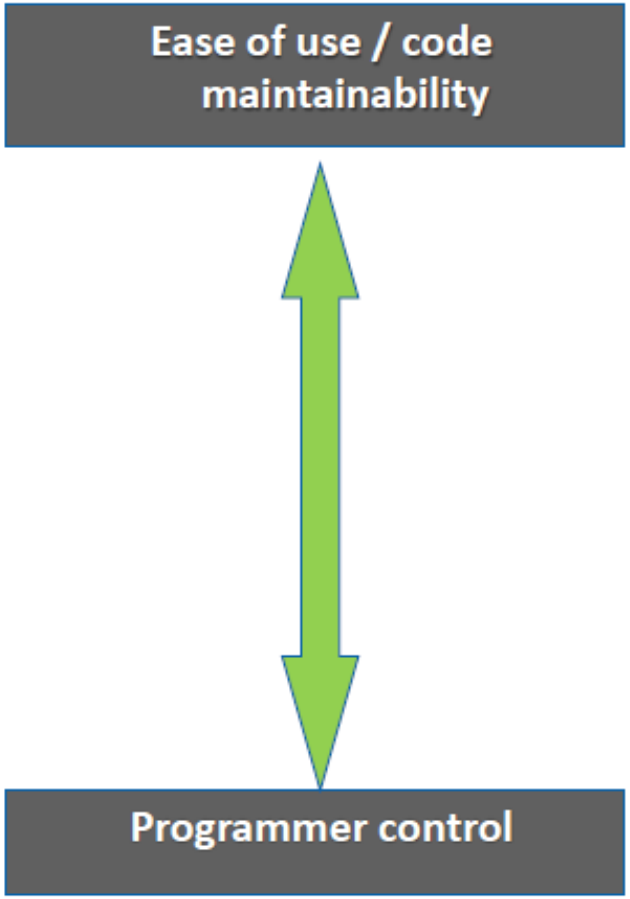


Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon Phi Coprocessor : Prog. Env & Compiler & Vectorization

Options for Thread Parallelism

- Intel® Math Kernel Library
- Intel® Threading Building Blocks
  Intel® Cilk™ Plus
- OpenMP*
- Pthreads* and other threading libraries



Source : References & Intel Xeon-Phi; http://www.intel.com/

Use Compiler Optimization Switches

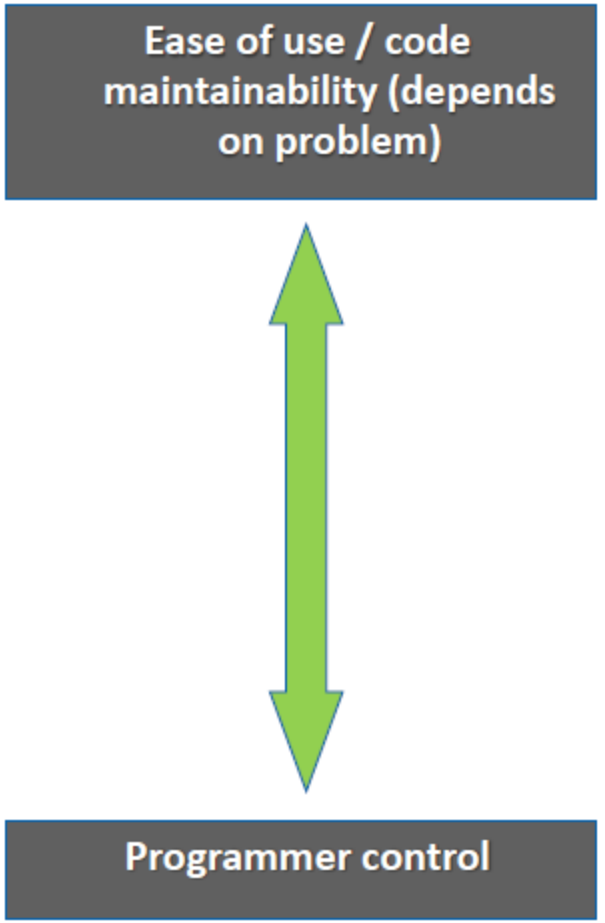
Compiler & Vectorization

Optimization Done	Linux*
Disable optimization	-O0
Optimize for speed (no code size increase)	-O1
Optimize for speed (default)	-O2
High-level loop optimization	-O3
Create symbols for debugging	-g
Multi-file inter-procedural optimization	-ipo
Profile guided optimization (multi-step build)	-prof-gen -prof-use
Optimize for speed across the entire program	-fast (same as: -ipo –O3 -no-prec-div -static -xHost)
OpenMP 3.0 support	-openmp
Automatic parallelization	-parallel

Source : References & Intel Xeon-Phi; http://www.intel.com/

Options for Vectorization

- Intel® Math Kernel Library
- Array Notation: Intel® Cilk™ Plus
- Automatic vectorization
- Semiautomatic vectorization with annotation: `#pragma vector`, `#pragma ivdep`, and `#pragma simd`
- C/C++ Vector Classes (`F32vec16`, `F64vec8`)
- Vector intrinsics (`_mm_add_ps`, `addps`)



Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon Phi : Coprocessors – Intel Compiler's Offload Programs

❖ Work done – Compiler's Offload

1. When the Intelcompiler encounters an offload pragma, it generates code for both the coprocessor and the host.
2. Code to transfer the data to the coprocessor is automatically created by the compiler,
3. The programmer can influence the data transfer by adding data clauses to the offload pragma.

Details can be found under "**Offload Using a Pragma**" in the Intel compiler documentation.

Offload Code Examples

C/C+ Offload Pragma

```
#pragma offload target (mic)
#pragma omp parallel for reduction(+:pi)
 for (i = 0; i<count; i++) {
      float t = (float) (i+0.5/count);
      pi += 4.0/(1.0t*t);
 }
pi/ = count;
```

C/C++ Offload Pragma

```
#pragma offload target(mic)
      in(transa, transb, N, alpha, beta) \
      in(A:length(matrix_elements)) \
      in(B:length(matrix_elements)) \
      inout(C:length(matrix_elements))
         sgemm(&transa, &transb, &N, &N, &N,
& alpha, A, &N, B, & N, &beta, C &N);
```

Fortran Offload Directives

```
!dir$ omp offload target(mic)
!$omp parallel do
     do i = 1, 10
                A(i) = B(i) * C(i)
     enddo
```

C/C++ Language Extension

```
class_Cilk_Shated common {
    int data1;
    int *data2;
    class common *next;
    void process();
}
_Cilk_Shared class common obj1, obj2;
_Cilk_spawn _offload obj1.process();
_Cilk_spawn  _offload  obj2.process();
```

Source : References & Intel Xeon-Phi; http://www.intel.com/

Options for Parallelism – pthreads*

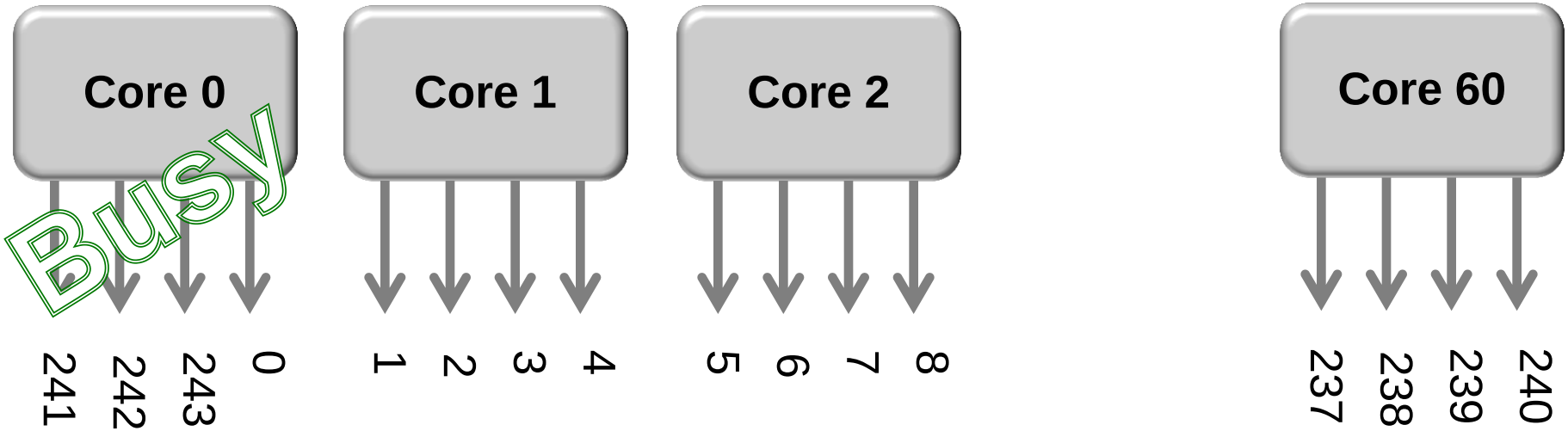
POSIX Threads

- POSIX* Standard for thread API with 20 years history
- Foundation for other high level threading libraries
- Independently exist on the host and Intel® MIC
- No extension to go from the host to Intel® MIC
- Advantage: Programmer has explicit control
  - From workload partition to thread creation, synchronization, load balance, affinity settings, etc.
- Disadvantage: Programmer has too much control
  - Code longevity
  - Maintainability
  - Scalability

Source : References & Intel Xeon-Phi; http://www.intel.com/

Thread Affinity using pthreads*

- Partition the workload to avoid load imbalance
  - Understand static vs. dynamic workload partition
- Use pthread API, define, initialize, set, destroy
  - Set CPU affinity with `pthead_setaffinity_np()`
  - Know the thread enumeration and avoid core 0
  - Core 0 boots the coprocessor, job scheduler, service interrupts



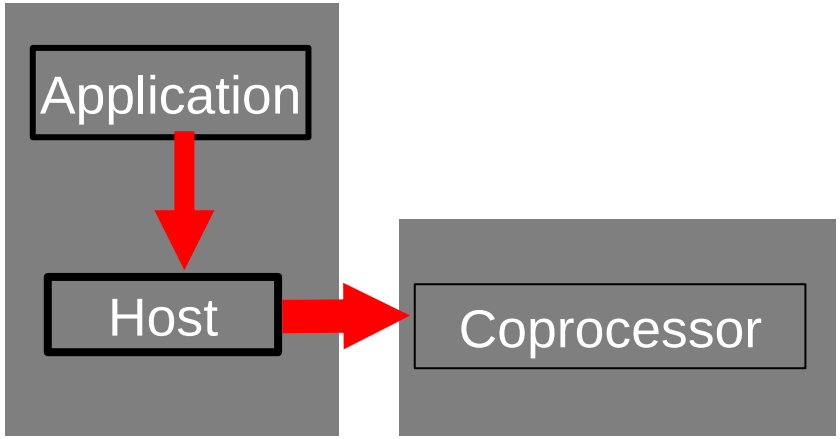
Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : OpenMP I Prog. Model

OpenMP

- OpenMP parallelization on an "**Intel Xeon + Xeon Phi coprocessor machine**" can be applied in **four** different programming models.

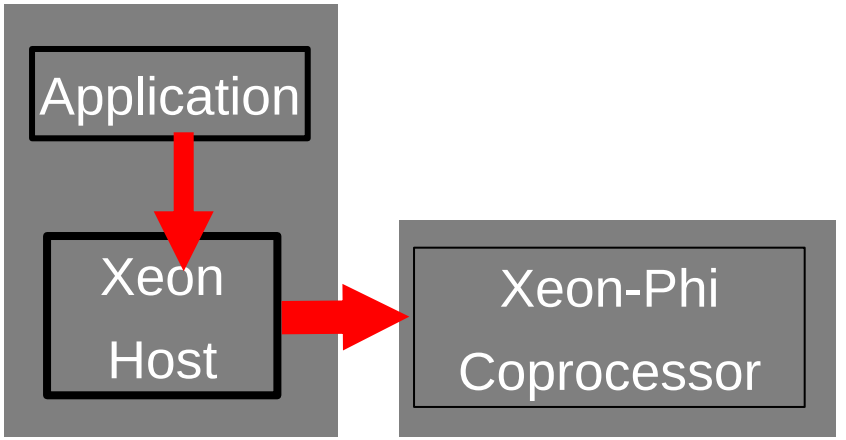
  - **Realized with Complier Options**



Intel Xeon-Phi : OpenMP I Prog. Model

❖ **Four** Models with different programming models

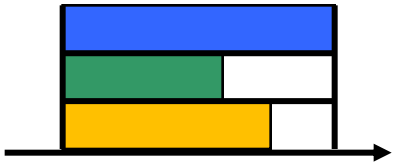
- Native OpenMP on the Xeon host
- Serial Xeon host with OpenMP offload
- Native OpenMP on the Xeon Phi coprocessor
- OpenMP on the Xeon Host with OpenMP offload



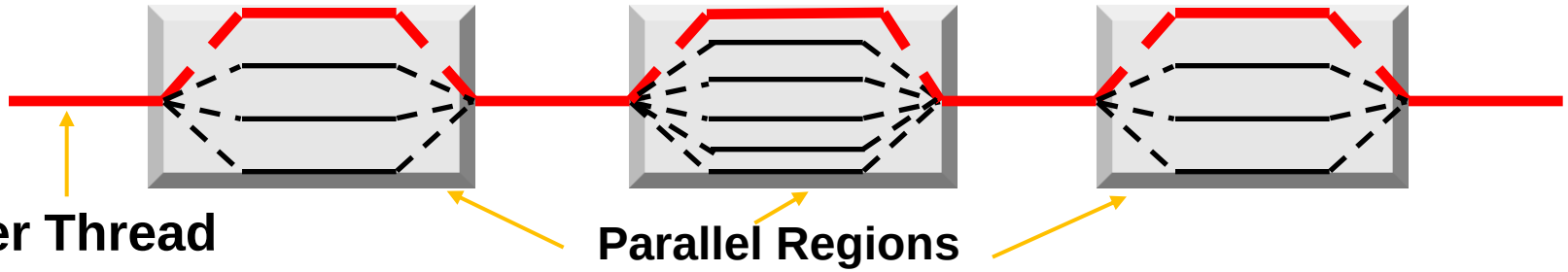
Options for Parallelism – OpenMP*

- Compiler directives/pragmas based threading constructs
  - Utility library functions and Environment variables
- Specify blocks of code executing in parallel

```
#pragma omp parallel sections
{
    #pragma omp section
    task1();
    #pragma omp section
    task2();
    #pragma omp section
    task3();
```



- Fork-Join Parallelism:
  - Master thread spawns a team of worker threads as needed
  - Parallelism grow incrementally

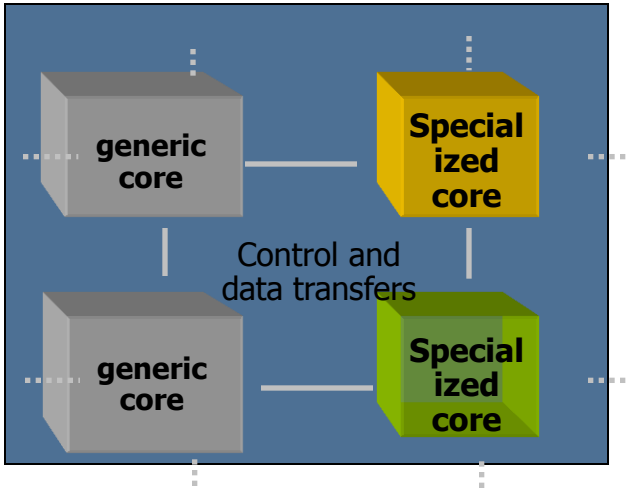


Source : References & Intel Xeon-Phi; http://www.intel.com/

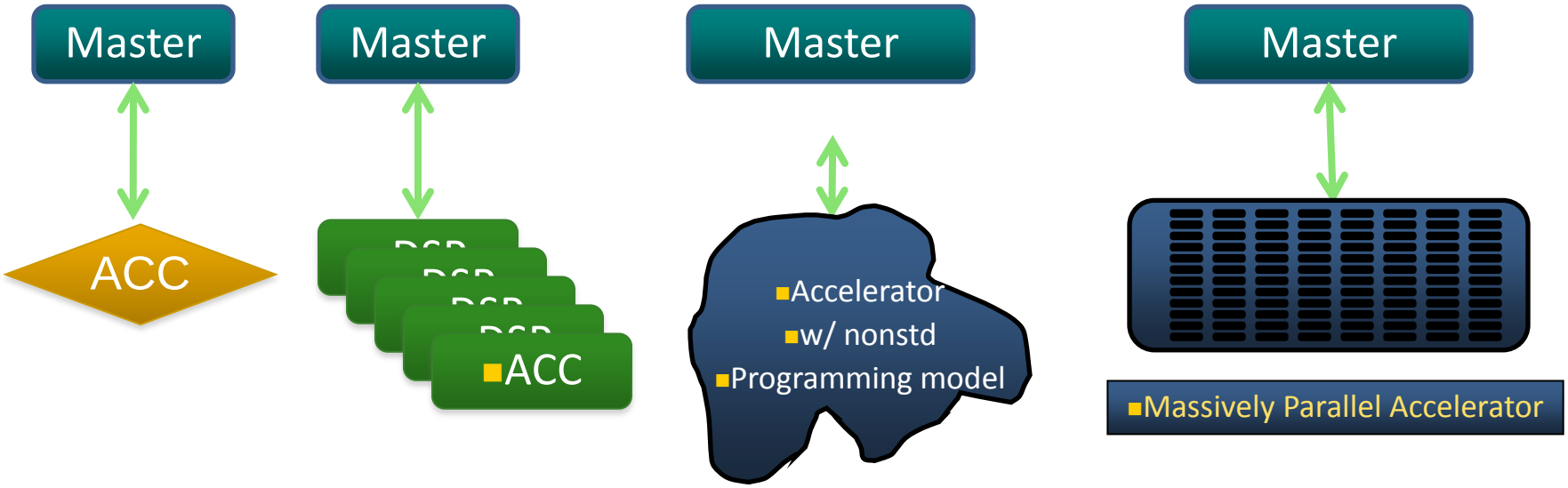
OpenMP Evolution Beyond 1,00,000 cores

- OpenMP language committee is actively working toward the expression of locality and heterogeneity
  - And to improve task model to enhance asynchrony

- How to identify code that should run on a certain kind of core?
- How to share data between host cores and other devices?
- How to minimize data motion?
- How to support diversity of cores?



OpenMP 4.0 Attempts To Target Range of Acceleration Configurations



- Dedicated hardware for specific function(s)
  - ❑ Attached to a master processor
  - ❑ Multiple types or levels of parallelism
    - Process level, thread level, ILP/SIMD
- May not support a full C/C++ or Fortran compiler
  - May lack stack or interrupts, may limit control flow, types

Intel Xeon-Phi Coprocessor : MPI on Cluster

Threading and affinity : Settings :

Settings	Description
OpenMP on host without HT	1 x ncore-host
OpenMP on host with HT	2 x ncore-host
OpenMP on Xeon Phi in native mode	4 x ncore-phi
OpenMP on Xeon Phi in offload mode	1 x ncore-phi-1

- If OpenMP regions exist on the **host** and on the part of the code **offloaded** to the Xeon Phi, **two** separate OpenMP runtimes exist.

Intel® Cilk™ Plus Technology: Elemental Function

Cilk Plus

- Allow you to define data operations using scalar syntax
- Compiler apply the operation to data arrays in parallel, utilizing both SIMD parallelism and core parallelism

Programmer	Build with Intel Cilk Plus Compiler
1. Writes a standard C/C++ scalar syntax 2. Annotate it with **__declspec**(vector) 3. Use one of the parallel syntax choices to invoke the function	1. Generates vector code with SIMD Instr. 2. Invokes the function iteratively, until all elements are processed 3. Execute on a single core, or use the task scheduler, execute on multicores

```
__declspec (vector)
double BlackScholesCall(double S,
                        double K,
                        double T)
                    {
 double d1, d2, sqrtT = sqrt(T);
d1 = (log(S/K)+R*T)/(V*sqrtT)+0.5*V*sqrtT;
            d2 = d1-(V*sqrtT);
 return S*CND(d1) - K*exp(-R*T)*CND(d2);
                    }
```

```
Cilk_for (int i=0; i < NUM_OPTIONS; i++)
     call[i] = BlackScholesCall(SList[i],
                                KList[i],
                                TList[i]);
```

Source : References & Intel Xeon-Phi; http://www.intel.com/

MPI Prog. Models for Xeon systems with MIC

Message Passing : MPI

Offload

- Intel ® MIC Architecture or host CPU as an accelerator

MIC Offload (direct acceleration)	Host Offload (reverse acceleration)
❖ MPI ranks on the host CPU only ❖ Messages into/out of the host CPU ❖ Intel ® MIC Architecture as an accelerator	❖ MPI ranks on the MIC CPU only ❖ Messages into/out of the MIC CPU ❖ Host CPU as an accelerator

MPI

- MPI ranks on several co-processors and/or host nodes
- Messages to/from any core

Co-processor-only	Symmetric
❖ MPI ranks on the MIC CPU only ❖ Messages into/out of the MIC CPU c/o host CPUs ❖ Threading possible	❖ MPI ranks on the MIC and host CPUs ❖ Messages into/ out of the MIC and host CPUs ❖ Threading possible

Source : References & Intel Xeon-Phi; http://www.intel.com/

MKL (Math Kernel Lib.

Simple way to Jobs using Intel MKL (Math Kernel Library)

Details on using MKL (11.0) with Intel Xeon Phi co-processors can be found in references. Also the MKL developer zone contains useful information. Intel MKL 11.0 Update 2 the following functions are highly optimized for the Xeon Phi

- BLAS Level 3, and much of Level 1 & 2
- Sparse BLAS:
- Some important LAPACK routines (LU, QR, Cholesky)
- Fast Fourier Transformations
- Vector Math Library & Other Lib.

Remark : All functions can be used on the Xeon Phi, however the optimization level for wider 512-bit SIMD instructions differs.

Intel Xeon-Phi Coprocessors (Intel MKL)

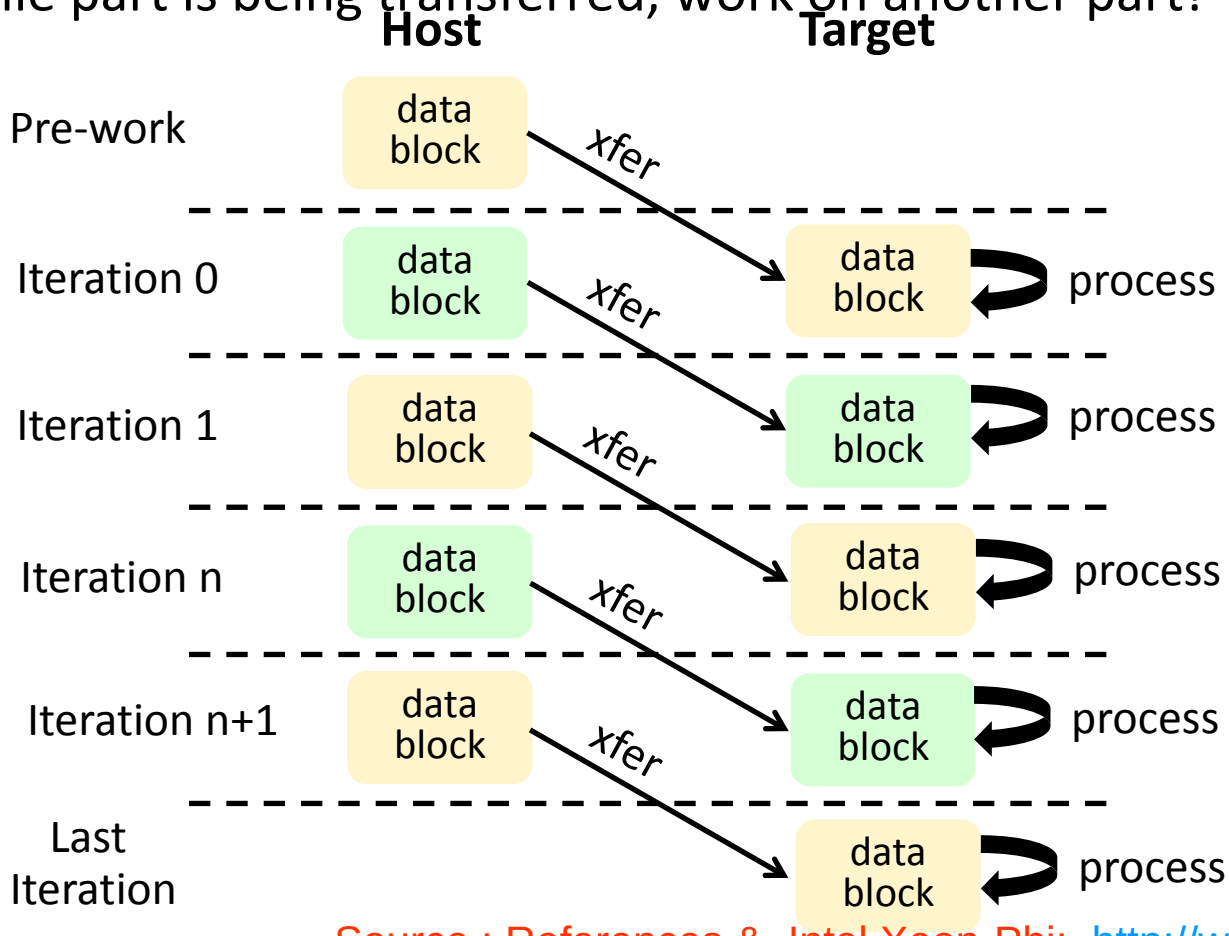
- On Xeon Phi coprocessor, the following usage models of MKL are available :
  - **Automatic Offload**
  - **Compiler Assisted Offload**
  - **Native Execution**

To know more about the availability of various functions for above usage models, Please refer MKL documents

Double Buffering Example

Asynchronous Comp.

- Transfer and work on a dataset in small pieces
- While part is being transferred, work on another part!



Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon Phi Coprocessor : Example Demonstration

Vector-Vector & Matrix-Matrix Addition

Vector-Vector & Matrix-Matrix Multiplication

References & Acknowledgements

References :

1. Theron Voran, Jose Garcia, Henry Tufo, University Of Colorado at Boulder National Center or Atmospheric Research, TACC-Intel Hihgly Parallel Computing Symposium, Austin TX, April 2012
2. Robert Harkness, Experiences with ENZO on the Intel R Many Integrated Core (Intel MIC) Architecture, National Institute for Computational Sciences, Oak Ridge National Laboratory
3. Ryan C Hulguin, National Institute for Computational Sciences, Early Experiences Developing CFD Solvers for the Intel Many Integrated Core (Intel MIC) Architecture, TACC-Intel Highly Parallel Computing Symposium April, 2012
4. Scott McMillan, Intel Programming Models for Intel Xeon Processors and Intel Many Integrated Core (Intel MIC) Architecture, TACC-Highly Parallel Comp. Symposium April 2012
5. Sreeram Potluri, Karen Tomko, Devendar Bureddy , Dhabaleswar K. Panda, Intra-MIC MPI Communication using MVAPICH2: Early Experience, Network-Based Computing Laboratory, Department of Computer Science and Engineering The Ohio State University, Ohio Supercomputer Center, TACC-Highly Parallel Computing Symposium April 2012
6. Karl W. Schulz, Rhys Ulerich, Nicholas Malaya ,Paul T. Bauman, Roy Stogner, Chris Simmons, Early Experiences Porting Scientific Applications to the Many Integrated Core (MIC) Platform ,Texas Advanced Computing Center (TACC) and Predictive Engineering and Computational Sciences (PECOS) Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin ,Highly Parallel Computing Symposium ,Austin, Texas, April 2012
7. Kevin Stock, Louis-Noel, Pouchet, P. Sadayappan ,Automatic Transformations for Effective Parallel Execution on Intel Many Integrated, The Ohio State University, April 2012
8. http://www.tacc.utexas.edu/
9. Intel MIC Workshop at C-DAC, Pune April 2013
10. First Intel Xeon Phi Coprocessor Technology Conference iXPTC 2013 New York, March 2013
11. Shuo Li, Vectorization, Financial Services Engineering, software and Services Group, Intel ctel Corporation;
12. Intel® Xeon Phi™ (MIC) Parallelization & Vectorization, Intel Many Integrated Core Architecture, Software & Services Group, Developers Relations Division

References & Acknowledgements

References :

13. Intel® Xeon Phi™ (MIC) Programming, Rama Malladi, Senior Application Engineer, Intel Corporation, Bengaluru India April 2013
14. Intel® Xeon Phi™ (MIC) Performance Tuning, Rama Malladi, Senior Application Engineer, Intel Corporation, Bengaluru India April 2013
15. Intel® Xeon Phi™ Coprocessor Architecture Overview, Dhiraj Kalamkar, Parallel Computing Lab, Intel Labs, Bangalore
16. Changkyu Kim,Nadathur Satish ,Jatin Chhugani ,Hideki Saito,Rakesh Krishnaiyer ,Mikhail Smelyanskiy ,Milind Girkar, Pradeep Dubey, Closing the Ninja Performance Gap through Traditional Programming and Compiler Technology , Technical Report Intel Labs , Parallel Computing Laboratory , Intel Compiler Lab, 2010
17. Colfax International Announces Developer Training for Intel® Xeon Phi™ Coprocessor, Industry First Training Program Developed in Consultation with Intel SUNNYVALE, CA, Nov, 2012
18. Andrey Vladimirov Stanford University and Vadim Karpusenko , Test-driving Intel® Xeon Phi™ coprocessors with a basic N-body simulation Colfax International January 7, 2013 Colfax International, 2013 http://research.colfaxinternational.com/
19. Jim Jeffers and James Reinders,Intel® Xeon Phi™ Coprocessor High-Performance Programming by Morgann Kauffman Publishers Inc, Elsevier, USA. 2013
20. Michael McCool, Arch Robison, James Reinders, Structured Parallel Programming: Patterns for Efficient Computation, Morgan Kaufman Publishers Inc, 2013.
21. Dan Stanzione, Lars Koesterke, Bill Barth, Kent Milfeld by Preparing for Stampede: Programming Heterogeneous Many-Core Supercomputers. TACC, XSEDE 12 July 2012
22. John Michalakes, Computational Sciences Center, NREL, & Andrew Porter, Opportunities for WRF Model Acceleration, WRF Users workshop, June 2012
23. Jim Rosinski , Experiences Porting NOAA Weather Model FIM to Intel MIC, ECMWF workshop On High Performance Computing in Meteorology, October 2012
24. Michaela Barth, KTH Sweden , Mikko Byckling, CSC Finland, Nevena Ilieva, NCSA Bulgaria, Sami Saarinen, CSC Finland, Michael Schliephake, KTH Sweden, Best Practice Guide Intel Xeon Phi v0.1, Volker Weinberg (Editor), LRZ Germany March 31 ,2013

References & Acknowledgements

References :

25. Barbara Chapman, Gabriele Jost and Ruud van der Pas, Using OpenMP, MIT Press Cambridge, 2008
26. Peter S Pacheco, An Introduction Parallel Programming, Morgann Kauffman Publishers Inc, Elsevier, USA. 2011
27. Intel Developer Zone: Intel Xeon Phi Coprocessor,
28. http://software.intel.com/en-us/mic-developer
29. Intel Many Integrated Core Architecture User Forum,
30. http://software.intel.com/en-us/forums/intel-many-integrated-core
31. Intel Developer Zone: Intel Math Kernel Library, http://software.intel.com/en-us
32. Intel Xeon Processors & Intel Xeon Phi Coprocessors – Introduction to High Performance Applications Development for Multicore and Manycore – Live Webinar, 26.-27, February .2013,
33. recorded http://software.intel.com/en-us/articles/intel-xeon-phi-training-m-core
34. Intel Cilk Plus Home Page, http://cilkplus.org/
35. James Reinders, Intel Threading Building Blocks (Intel TBB), O'REILLY, 2007
36. Intel Xeon Phi Coprocessor Developer's Quick Start Guide,
37. http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-developers-quick-start-guide
38. Using the Intel MPI Library on Intel Xeon Phi Coprocessor Systems,
39. http://software.intel.com/en-us/articles/using-the-intel-mpi-library-on-intel-xeon-phi-coprocessor-systems
40. An Overview of Programming for Intel Xeon processors and Intel Xeon Phi coprocessors,
41. http://software.intel.com/sites/default/files/article/330164/an-overview-of-programming-for-intel-xeon-processors-and-intel-xeon-phi-coprocessors_1.pdf
42. Programming and Compiling for Intel Many Integrated Core Architecture,
43. http://software.intel.com/en-us/articles/programming-and-compiling-for-intel-many-integrated-core-architecture
44. Building a Native Application for Intel Xeon Phi Coprocessors,
45. http://software.intel.com/en-us/articles/

References & Acknowledgements

References :

46. Advanced Optimizations for Intel MIC Architecture, http://software.intel.com/en-us/articles/advanced-optimizations-for-intel-mic-architecture
47. Optimization and Performance Tuning for Intel Xeon Phi Coprocessors - Part 1: Optimization Essentials, http://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeonphi-coprocessors-part-1-optimization
48. Optimization and Performance Tuning for Intel Xeon Phi Coprocessors, Part 2: Understanding and Using Hardware Events, http://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-2-understanding
49. Requirements for Vectorizable Loops,
50. http://software.intel.com/en-us/articles/requirements-for-vectorizable-
51. R. Glenn Brook, Bilel Hadri, Vincent C. Betro, Ryan C. Hulguin, Ryan Braby. Early Application Experiences with the Intel MIC Architecture in a Cray CX1, National Institute for Computational Sciences. University of Tennessee. Oak Ridge National Laboratory. Oak Ridge, TN USA
52. http://software.intel.com/mic-developer
53. Loc Q Nguyen , Intel Corporation's Software and Services Group , Using the Intel® MPI Library on Intel® Xeon Phi™ Coprocessor System,
54. Frances Roth, System Administration for the Intel® Xeon Phi™ Coprocessor, Intel white Paper
55. Intel® Xeon Phi™ Coprocessor, James Reinders, Supercomputing 2012 Presentation
56. Intel® Xeon Phi™ Coprocessor Offload Compilation, Intel software

References

1. Andrews, Grogory R. (2000), Foundations of Multithreaded, Parallel, and Distributed Programming, Boston, MA : Addison-Wesley
2. Butenhof, David R (1997), Programming with POSIX Threads , Boston, MA : Addison Wesley Professional
3. Culler, David E., Jaswinder Pal Singh (1999), Parallel Computer Architecture - A Hardware/Software Approach , San Francsico, CA : Morgan Kaufmann
4. Grama Ananth, Anshul Gupts, George Karypis and Vipin Kumar (2003), Introduction to Parallel computing, Boston, MA : Addison-Wesley
5. Intel Corporation, (2003), Intel Hyper-Threading Technology, Technical User's Guide, Santa Clara CA : Intel Corporation Available at : http://www.intel.com
6. Shameem Akhter, Jason Roberts (April 2006), Multi-Core Programming - Increasing Performance through Software Multi-threading , Intel PRESS, Intel Corporation,
7. Bradford Nichols, Dick Buttlar and Jacqueline Proulx Farrell (1996), Pthread Programming O'Reilly and Associates, Newton, MA 02164,
8. James Reinders, Intel Threading Building Blocks – (2007) , O'REILLY series
9. Laurence T Yang & Minyi Guo (Editors), (2006) High Performance Computing - Paradigm and Infrastructure Wiley Series on Parallel and Distributed computing, Albert Y. Zomaya, Series Editor
10. Intel Threading Methodology ; Principles and Practices Version 2.0 copy right (March 2003), Intel Corporation
11. William Gropp, Ewing Lusk, Rajeev Thakur **(1999),** Using MPI-2, Advanced Features of the Message-Passing Interface, The MIT Press..
12. Pacheco S. Peter, **(1992),** Parallel Programming with MPI, , University of Sanfrancisco, Morgan Kaufman Publishers, Inc., Sanfrancisco, California
13. Kai Hwang, Zhiwei Xu, (**1998**), Scalable Parallel Computing (Technology Architecture Programming), McGraw Hill New York.
14. Michael J. Quinn (**2004**), Parallel Programming in C with MPI and OpenMP McGraw-Hill International Editions, Computer Science Series, McGraw-Hill, Inc. Newyork
15. Andrews, Grogory R. **(2000),** Foundations of Multithreaded, Parallel, and Distributed Progrmaming, Boston, MA : Addison-Wesley

References

16. SunSoft. Solaris multithreaded programming guide. SunSoft Press, Mountainview, CA, **(1996),** Zomaya, editor. Parallel and Distributed Computing Handbook. McGraw-Hill,
17. Chandra, Rohit, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, and Ramesh Menon, **(2001)**,Parallel Programming in OpenMP San Fracncisco Moraan Kaufmann
18. S.Kieriman, D.Shah, and B.Smaalders **(1995),** Programming with Threads, SunSoft Press, Mountainview, CA. 1995
19. Mattson Tim, **(2002),** Nuts and Bolts of multi-threaded Programming Santa Clara, CA : Intel Corporation, Available at : http://www.intel.com
20. I. Foster **(1995,** Designing and Building Parallel Programs ; Concepts and tools for Parallel Software Engineering, Addison-Wesley (1995
21. J.Dongarra, I.S. Duff, D. Sorensen, and H.V.Vorst **(1999),** Numerical Linear Algebra for High Performance Computers (Software, Environments, Tools) SIAM, 1999
22. OpenMP C and C++ Application Program Interface, Version 1.0". **(1998),** OpenMP Architecture Review Board. October 1998
23. D. A. Lewine. *Posix Programmer's Guide:* **(1991),** Writing Portable Unix Programs with the Posix. 1 Standard. O'Reilly & Associates, 1991
24. Emery D. Berger, Kathryn S McKinley, Robert D Blumofe, Paul R.Wilson, *Hoard : A Scalable Memory Allocator for Multi-threaded Applications* ; The Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX). Cambridge, MA, November (**2000**). Web site URL : http://www.hoard.org/
25. Marc Snir, Steve Otto, Steyen Huss-Lederman, David Walker and Jack Dongarra, (**1998**) *MPI-The Complete Reference: Volume 1, The MPI Core, second edition* [MCMPI-07].
26. William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, and Marc Snir (**1998**) *MPI-The Complete Reference: Volume 2, The MPI-2 Extensions*
27. A. Zomaya, editor. Parallel and Distributed Computing Handbook. McGraw-Hill, **(1996)**
28. OpenMP C and C++ Application Program Interface, Version 2.5 (**May 2005**)", From the OpenMP web site, URL **: http://www.openmp.org/**
29. Stokes, Jon 2002 Introduction to Multithreading, Super-threading and Hyper threading *Ars Technica*, October **(2002)**

References

30. Andrews Gregory R. 2000, Foundations of Multi-threaded, Parallel and Distributed Programming, Boston MA : Addison – Wesley (**2000)**
31. Deborah T. Marr , Frank Binns, David L. Hill, Glenn Hinton, David A Koufaty, J . Alan Miller, Michael Upton, "Hyperthreading, Technology Architecture and Microarchitecture", Intel (**2000-01)**
32. http://www.erc.msstate.edu/mpi
33. http://www.arc.unm.edu/workshop/mpi/mpi.html
34. http://www.mcs.anl.gov/mpi/mpich
35. The MPI home page, with links to specifications for MPI-1 and MPI-2 standards : http://www.mpi–forum.org
36. Hybrid Programming Working Group Proposals, Argonne National Laboratory, Chiacago (2007-2008)
37. TRAC Link : https://svn.mpi-forum.org/trac/mpi-form-web/wiki/MPI3Hybrid
38. Threads and MPI Software, Intel Software Products and Services 2008 - 2009
39. Sun MPI 3.0 Guide November 2007
40. Treating threads as MPI processes thru Registration/deregistration –Intel Software Products and Services 2008 – 2009
41. Intel MPI library 3.2 - http://www.hearne.com.au/products/Intelcluster/edition/mpi/663/
42. http://www.cdac.in/opecg2009/
43. PGI Compilers http://www.pgi.com

Thank You

Any questions ?