# 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 An Overview – MKL

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

# An Overview of Xeon Phi Coprocessor

## Lecture Outline

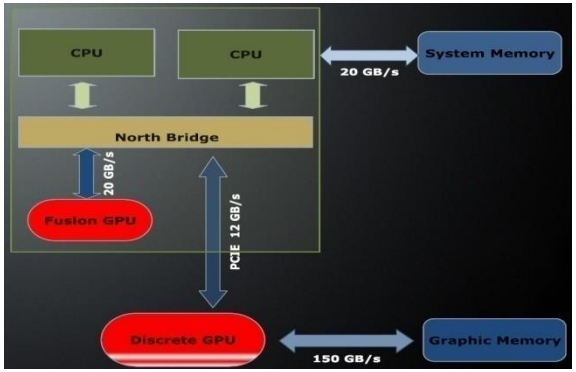
Following topics will be discussed

- Understanding of Xeon –Phi Architectures
- Programming on Xeon-Phi Architectures –MKL
- Tuning & Performance – Software Threading

# MIC Architecture, System Overview

# 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**

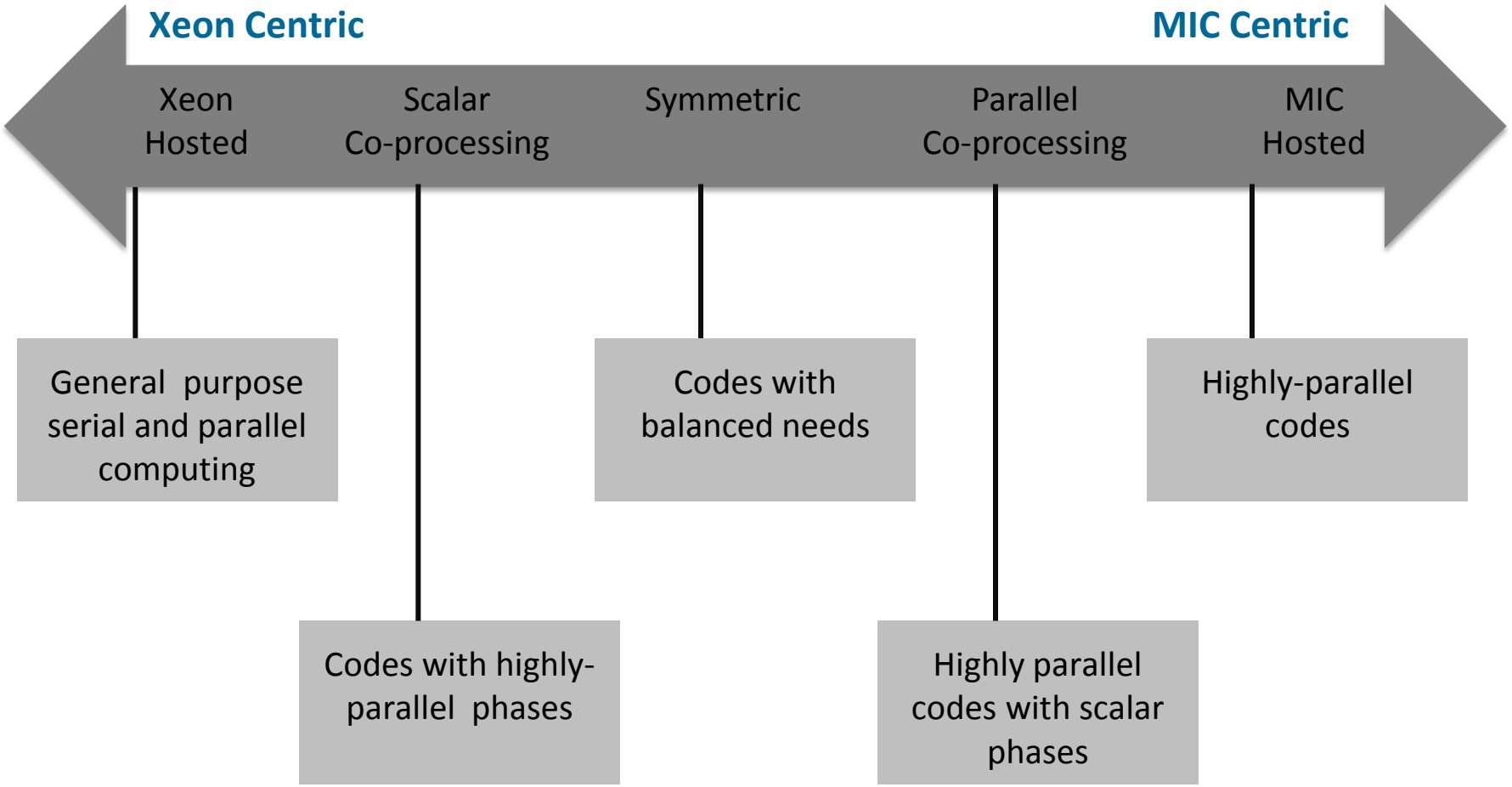
CPU & ACC

**Architectural integration**

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

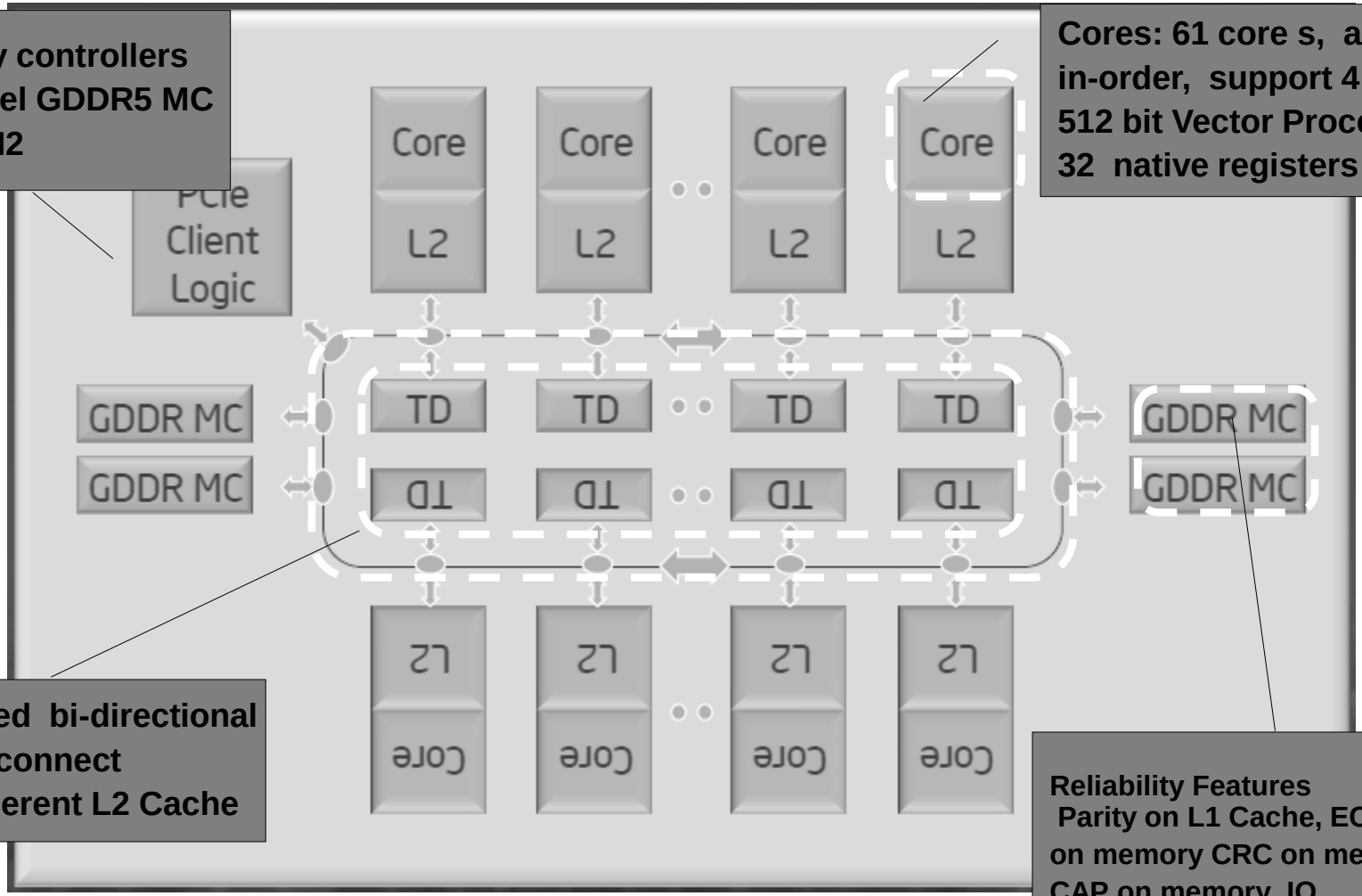
**Source :** NVIDIA, AMD, SGI, Intel, IBM Alter, Xilinux References

# Compute modes vision



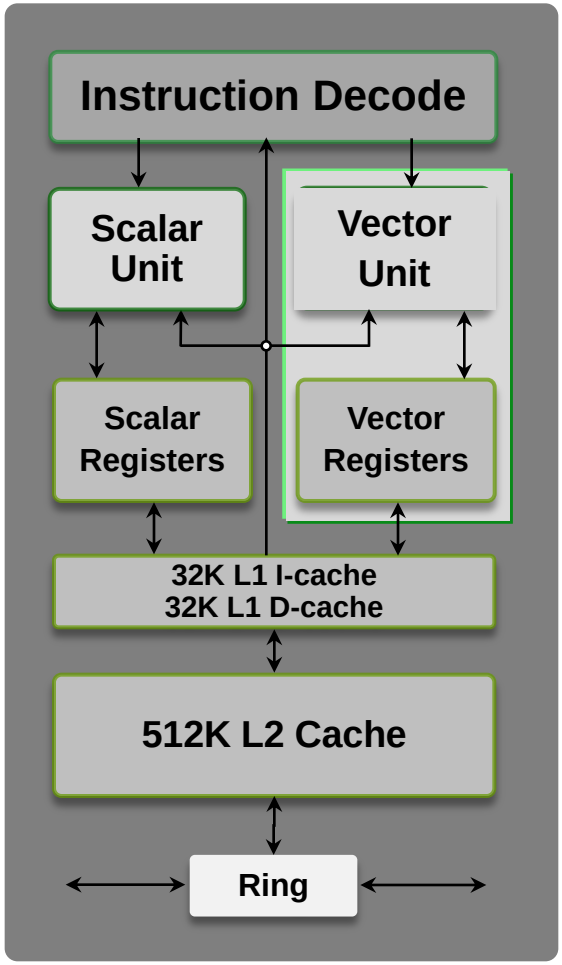

Source : References & Intel Xeon-Phi; http://www.intel.com/

# Intel® Xeon Phi™ Architecture Overview




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

# 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

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

# Intel Xeon-Phi Coprocessor architecture Overview

## Quick Glance*

- The Intel Xeon Phi coprocessor Architecture Overview (Core, VPU, CRI, Ring, SBOX, GBOX, PMU)
- The Cache hierarchy (Details of L1 & L2 Cache)
- Network Configuration (MPSS) : (Obtain the information can be obtained by running the micinfo program on the host. )
- System Access

**Remark : Root** privileges are necessary for the destination directories (Required for availability of some library usage for codes such MKL)

*(* = Useful for tuning and Performance)*

# Intel Xeon-Phi Coprocessor architecture Overview

- The Intel Xeon Phi coprocessor consists of up to 61 cores connected by a high performance on-die bidirectional interconnect.
- The coprocessor runs a full service Linux operating system
- The coprocessor supports all important Intel development tools, like C/C++ and Fortran compiler, MPI and OpenMP
- To Coprocessor support s high performance libraries like MKL, debugger and tracing tools like Intel VTune Amplifier XE.

# Intel Xeon-Phi Coprocessor architecture Overview

- The Intel Xeon Phi coprocessor The coprocessor is connected to an Intel Xeon processor - the "host" - via the PCI Express (PICe) bus.
- The implementation of a virtualized TCP/IP stack allows to access the coprocessor like a network node.

**Remark :** Summarized information can be found In the following MIC architecture from the System Software Developers Guide and other references

# Intel Xeon-Phi Coprocessor System Access

## Quick Glance:

- Details about the system startup and the network configuration can be found in Intel Xeon-Phi documentation coming with MPSS
- To start the Intel Manycore Platform Software Stack (Intel MPSS) and initialize the Xeon Phi coprocessor the following command has to be executed as root or during host system start-up:

```
hypack-root@mic-0:~> sudo service mpss start
```

**Remark :** The above command has to be executed as a root

# Intel Xeon-Phi Coprocessor System Access

## Quick Glance:

- To start the Intel Manycore Platform Software Stack (Intel MPSS) and initialize the Xeon Phi coprocessor the following command has to be executed as root or during host system start-up:

```
hypack-root@mic-0:~> sudo service mpss start
```

**Remark :** The above command has to be executed as a root.

Details about the system startup and the network configuration can be found in Intel Xeon-Phi documentation coming with MPSS. For other necessary commands, refer Intel Xeon Phi documentation

# 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:

```
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 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.X
```

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

```
[hypack01@mic-0]$ tail -n26 /proc/cpuinfo
```

.....

# Intel Xeon-Phi
# Shared Address Space Programming
# OpenMP, Intel TBB
# Explicit Message Passing – MPI
# MKL Math Kernel Library

# MKL
## *Math Kernel Library*

# Intel Xeon-Phi Coprocessors (Intel MKL)

**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 Intel Xeon Phi coprocessor:

- BLAS Level 3, and much of Level 1 & 2
- Sparse BLAS:
- Some important LAPACK routines (LU, QR, Cholesky)
- Fast Fourier Transformations
- Vector Math Library
- Random number generators in the Vector Statistical Library

**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

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Automatic Offload (AO) :**

- In the case of automatic offload the user does not have to change the code at all.
- For automatic offload enabled functions the runtime may automatically download data to the Xeon Phi coprocessor and execute (all or part of) the computations there.
- The data transfer and the execution management is completely automatic and transparent

**Remark :** The matrix sizes for which MKL decides to offload the computation should be **indicated in function statement.** Refer Intel MKL documents

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Automatic Offload (AO) :**

- **Approach 1 :** call the function **`mkl_mic_enable()`** within the source code
- **Approach 2 :** Set the environment variable **`MKL_MIC_ENABLE =1`**

The data transfer and the execution management is completely automatic and transparent

**Remark :** If **no** Xeon Phi coprocessor is detected the application runs on the host without penalty.

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Automatic Offload (AO) :** To build a program for automatic offload, the same way of building code as on the **Xeon host** is used:

```
icc -O3 -mkl file.c -o file
```

By default, the MKL library decides when to offload and also tries to determine the optimal work division between the host and the targets . In case of the BLAS routines the user can specify the work division between the host and the coprocessor by calling the routine

```
mkl_mic_set_Workdivision(MKL_TARGET_MIC,0,0.5)
```

or by setting the environment variable

```
MKL_MIC_0_WORKDIVISION=0.5
```

Both examples specify to offload 50% of computation only to the 1st card (card #0).

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Compiler Assisted Offload (CAO) :** In this mode of MKL the offloading is explicitly controlled by compiler pragmas or directives.

**Advantage :**

1. A big advantage of this mode is that it allows for data persistence on the device.
2. All MKL function can be offloaded in CAO-mode. (In contrast to the automatic offload mode.)

**Remarks :**

- For Intel compilers it is possible to use AO and CAO in the same program, however the work division must be explicitly set for AO in this case. Otherwise, all MKL AO calls are executed on the host.
- MKL functions are offloaded in the same way as any other offloaded function.

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Compiler Assisted Offload (CAO) :** To build a program for compiler assisted offload, the following command is recommended by Intel:

```
#pragma offload target(mic) \
      in(transa, transb, N, alpha, beta) \
      in(A:length(N*N)) in(B:length(N*N)) \
      in(C:length(N*N)) \
      out(C:length(N*N) alloc_if(0))
{
   sgemm(&transa, &transb, &N, &N, &N, \
         &alpha, A, &N, B, &N, &beta, C, &N);
}
```

**Remarks :**. Refer Intel MKL documents

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Compiler Assisted Offload (CAO) :** To build a program for compiler assisted offload, the following command is recommended by Intel:

```
icc -O3 -openmp -mkl \
-offload-option,mic,ld, \
 "-L$MKLROOT/lib/mic -Wl,\
 --start-group -lmkl_intel_lp64 \
 -lmkl_intel_thread \
 -lmkl_core -Wl,--end-group" \
  hello.c -o file
```

**Remarks :** Setting larger pages by the environment setting `MIC_USE_2MB_BUFFERS=16K` usually increases performance. It is also recommended to exploit data persistence with CAO. Refer Intel MKL documents

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Native Execution :** In this mode of MKL the Intel Xeon Phi coprocessor is used as an independent compute node.

To build a program for native mode, the following compiler settings should be used:

```
icc -O3 -mkl -mmic file.c -o file
```

**Example code :** Example code can be found under

`$MKLROOT/examples/mic_ao` and

`$MKLROOT/examples/mic_offload`

**Remarks :** The binary must then be manually copied to the coprocessor via **ssh** and directly started on the coprocessor or Cluster environment automatically copy the data

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Native Execution :** In this mode of MKL the Intel Xeon Phi coprocessor is used as an independent compute node.

To build a program for native mode, the following compiler settings should be used:

```
icc -O3 -mkl -mmic file.c -o file
```

**Example code :** Example code can be found under

`$MKLROOT/examples/mic_ao` and

`$MKLROOT/examples/mic_offload`

**Remarks :** The binary must then be manually copied to the coprocessor via **ssh** and directly started on the coprocessor or Cluster environment automatically copy the data

# Intel Xeon-Phi Coprocessors (Intel MKL)

**Summary: Tricks for Performance**

- Use asynchronous data transfer and double buffering offloads to overlap the communication with the computation
- Optimizing memory use on Intel MIC architecture target relies on understanding access patterns
- Many old tricks still apply: peeling, collapsing, unrolling, vectorization can all benefit performance

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

# An Overview of Intel Xeon-Phi Coprocessors

## Conclusions

- An Overview of Intel Xeon-Phi Architecture; Tuning & Performance of Software threading- using MKL

# Thank You

*Any questions ?*

# 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