Intel® Xeon Phi™ Coprocessor Software Ecosystem
Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

- Intel may make changes to specifications and product descriptions at any time, without notice.

- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

- Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user

- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance

- Intel, Core, Xeon, VTune, Cilk, Intel and Intel Sponsors of Tomorrow. and Intel Sponsors of Tomorrow. logo, and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

- OpenCL and the OpenCL logo are trademarks of Apple Inc and are used by permission by Khronos.

- *Other names and brands may be claimed as the property of others.

- Copyright ©2011 Intel Corporation.

- Hyper-Threading Technology: Requires an Intel® HT Technology enabled system, check with your PC manufacturer. Performance will vary depending on the specific hardware and software used. Not available on all Intel® Core™ processors. For more information including details on which processors support HT Technology, visit http://www.intel.com/info/hyperthreading

- Intel® 64 architecture: Requires a system with a 64-bit enabled processor, chipset, BIOS and software. Performance will vary depending on the specific hardware and software you use. Consult your PC manufacturer for more information. For more information, visit http://www.intel.com/info/em64t

- Intel® Turbo Boost Technology: Requires a system with Intel® Turbo Boost Technology capability. Consult your PC manufacturer. Performance varies depending on hardware, software and system configuration. For more information, visit http://www.intel.com/technology/turboboost
SW Ecosystem Topics

• System Software and OS
• Application Environment
  – Performance Considerations
  – System Topology
  – Memory
  – Clocks and Timing
Module Outline

• System Software and OS
• Application Environment
Not much to say

It’s just Linux*
Card OS

• Bootstrap
  – Memory training
  – Build MP Table, e820 map, load image in memory (bzImage)
  – Jump to 32-bit protected mode entry point in the kernel

• It’s just Linux* with some minor modifications:
  ▪ IPI format, local APIC calibration, no support for compatibility mode (CSTAR)
  ▪ No global bit in PTE/PDE entries.
  ▪ IOAPIC programmed via MMIO
  ▪ Instructions not supported: cmov, in/out, monitor/mwait, fence
  ▪ No support for MMX/SSE registers
  ▪ Save/restore of vector state via DNA (CR0.TS=1)
• 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
  – 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® Initial Many Core Instructions
  – 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,...)
System SW Environment

Host
- Intel® TBB
- Intel® MKL
- Intel® CILK™ Plus
- Intel® C/C++ and Intel® Fortran Compilers
- OpenMP®
- Tools & Apps
- Debuggers
- Board Tools
- Control Panel
- Sockets
- COI
- MYO
- OFED* Verbs
- HCA Library
- OFED*/SCIF Library
- User SCIF Library
- IB Proxy Daemon
- HCA Proxy
- TCP/IP
- UDP/IP
- Host/SCIF Driver
- NetDev
- Linux* Kernel
- Management Middleware
- SMC
- BMC
- ME
- SMC Update Path
- /sys/proc
- Linux* Micro-OS
- PCI Express*
- InfiniBand* HCA

Intel® Xeon Phi™ coprocessor
- OpenMP®
- Intel® MKL
- Intel® CILK™ Plus
- Intel® C/C++ and Intel® Fortran Compilers
- Intel® Vtune Amplifier XE
- Intel® CILK™ Plus
- Intel® MKL
- Intel® C/C++ and Intel® Fortran Compilers
- Intel® Vtune Amplifier XE
- Debuggers
- Tools & Apps
- Intel® MPI
- NetDev
- Card OS
- Driver
- OFED* Core SW
- OFED*/SCIF Driver
- HCA Proxy
- NetDev
- Linux* Micro-OS
- SMC
- SMC Update Path
- /sys/proc
- Linux* Micro-OS
- PCI Express*
- InfiniBand* HCA

Legend
- MPSS Install
- Std. SW
- Mod. Linux*
- Intel® SW
- Std. OFED*
- Intel® HW
- Other HW
SW Ecosystem Topics

- System Software and OS
- Application Environment
  - Performance Considerations
  - System Topology
  - Memory
  - Clocks and Timing
Spectrum of Programming Models and Mindsets

Multi-Core Centric

- Multi-Core Hosted
  - General purpose serial and parallel computing

Symmetric

- Codes with balanced needs

Many-Core Centric

- Many Core Hosted
  - Highly-parallel codes

Offload

- Codes with highly-parallel phases

Range of models to meet application needs

Intel® Xeon® processor

Multi-core
(Xeon processor)

Many-core
(Intel® Xeon Phi™ coprocessor)
Offload Programming Model
Data Transfer

• Explicit distributed memory programming via MPI (or sockets)
• Two offload data transfer models are available:
  1. Explicit Copy
     o Programmer designates variables that need to be copied between host and card in the offload directive
     o Syntax: Pragma/directive-based
     o C/C++ Example: #pragma offload target(mic) in(data:length(size))
     o Fortran Example: !dir$ offload target(mic) in(a1:length(size))
  2. Implicit Copy
     o Programmer marks variables that need to be shared between host and card
     o The same variable can then be used in both host and coprocessor code
     o Runtime automatically maintains coherence at the beginning and end of offload statements
     o Syntax: keyword extensions based
     o Example: _Cilk_shared double foo; _Offload func(y);
Development Options: Just the Usual

IA Benefit: Wide Range of Development Options

Parallelization Options

- Intel® Math Kernel Library
- OpenMP*
- Intel® Threading Building Blocks
- Pthreads*

Vector Options

- Intel Math Kernel Library
- Auto vectorization
- Semi-auto vectorization: #pragma (vector, ivdep, simd)
- Array Notation: Intel Cilk Plus
- C/C++ Vector Classes (F32vec16, F64vec8)
- OpenCL*
- Intrinsics

Ease of use

Fine control
Development Environment

The familiar Intel development environment is available:

- Intel® C, C++ and Fortran Compilers
- OpenMP*
- Intel® MPI Library support for the Intel® MIC Architecture as an MPI node
- Intel® Parallel Building Blocks
  - Intel® Threading Building Blocks (Intel® TBB)
  - Intel® Cilk™ Plus
- GDB (enhanced)
- Intel® Performance Libraries (e.g. Intel® MKL)
  - Three versions: host-only, coprocessor-only, heterogeneous
- Intel® VTune™ Amplifier XE
- Standard runtime libraries, even pthreads*
Module Outline

• System Software and OS
• Application Environment
Two Application Execution Environments

**Linux** Host

- Host-side offload application
  - User code
- Offload libraries, user-level driver, user-accessible APIs and libraries

**Intel® MIC Architecture**

- Target-side “native” application
  - User code
  - Standard OS libraries plus any 3rd-party or Intel® libraries
- Target-side offload application
  - User code
  - Offload libraries, user-accessible APIs and libraries

Virtual terminal session

- ssh or telnet connection to /dev/mic*

Intel MIC Architecture support libraries, tools, and drivers

**PCI-E** Bus

- Linux OS

Intel MIC Architecture communication and application-launching support

**PCI-E** Bus

- Linux OS

Other brands and names are the property of their respective owners.
Execution Modes

Native

• Card is an SMP machine running Linux*
• Separate executables run on both coprocessor and Intel Xeon processor
  – e.g. Standalone MPI applications
• No source code modifications most of the time
  – Recompile code for coprocessor
• Autonomous Compute Node (ACN)

Offload

• “main” runs on Intel Xeon processor
• 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 (…)

* Other brands and names are the property of their respective owners.
Native is Easy

• Cross compile your application for \textit{k1om} arch
  – Intel® C/C++ and Fortran compiler, \textit{k1om} aware GCC port.
  – Binutils for \textit{k1om} e.g. \texttt{objdump}
  – LSB – glibc, libm, librt, libcurses, etc.
  – Busybox – minimal shell environment

• Virtual Ethernet driver allows:
  – ssh, scp
  – NFS mounts

You still have to spend time parallelizing and vectorizing your application for performance on Intel® Xeon Phi™ coprocessor
"Hello World"

```c
#include <stdio.h>
#include <string.h>

#define TWO_MB 2 * 1024 * 1024

int main()
{
    char *p;

    p = malloc(TWO_MB);
    if (p == NULL) {
        printf("malloc failed!\n");
        return (-1);
    }

    printf("malloc returned (%p)\n", p);

    memset(p, 'a', TWO_MB);
    return (0);
}

$ icc -mmic test.c -o test
$ scp test mic0:
$ ssh mic0
$ sshmic0 mic0: $ uname -a
Linux mic0.local 2.6.34.11-g65c0cd9 #2 SMP Tue Dec 11 14:34:04 PST 2012 klom klom0
$ sshmic0 mic0: $ ./test
malloc returned (0x7fa6ad144010)
```
Performance Considerations for Native

- Single vs. Multi-threaded applications
  - Scalability is important
- Scalar vs. Vector code
- Explicit cache management
  - SW prefetching and evicting

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th>cores</th>
<th>vector width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xeon</td>
<td>2.6</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Intel® Xeon Phi™ Coprocessor</td>
<td>1.09</td>
<td>61</td>
<td>16</td>
</tr>
</tbody>
</table>

Intel Xeon Phi coprocessor has a Fused Multiply Add (FMA) for 2x flops/cycle
General Programming Considerations
System from a App’s 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
  – 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,...)
System Topology and Thread Affinity

- Why – threads sharing L1 and L2 cache
- sched_affinity/pthread_setaffinity_np or KMP_AFFINITY=proclist=[...]
  - But is your affinity correct or expected?
  - KMP_AFFINITY=explicit, proclist=[0-243]

OMP internal thread 0 -> CPU # 0
OMP internal thread 1 -> CPU # 1
OMP internal thread 2 -> CPU # 2
...
OMP internal thread 243 -> CPU # 243

- KMP_AFFINITY=explicit, proclist[1-243, 0]?
Clocksource and gettimeofday()

- A “clocksource” is a monotonically increasing counter
- Intel® Xeon Phi™ coprocessor has three clocksources
  - jiffies, tsc, micetc
- The Local APIC in each HW thread has a timer (HZ)
  - jiffies is a good clocksource, but very low resolution
- TSC is a good clocksource
  - But TSC is not *frequency invariant* and *non-stop*
  - Future release will use another clocksource to fix this.
- Elapsed Time Counter (ETC) that is frequency invariant
  - Expensive when multiple gettimeofday() calls – involves an MMIO read
- Recommend using “clocksource = tsc” on kernel command line

```
> cat sys/devices/system/clocksource/clocksource0/current_clocksource
  tsc
> cat sys/devices/system/clocksource/clocksource0/available_clocksource
  micetc tsc
```
SW Ecosystem Summary

- System Software and OS
- Application Environment
  - Performance Considerations
  - System Topology
  - Memory
  - Clocks and Timing
Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101
Backup