Intel® Xeon Phi™ Programming

James Tullos

james.a.tullos@intel.com
Technical Consulting Engineer
Intel® Cluster Tools

Learning Objectives

- Recognize the differences of Intel® Xeon™ vs. Intel® Xeon Phi™
- Recognize Intel® Software Development Product support levels for Intel® Xeon Phi™
- Differentiate execution models
- Optimize for Intel® Xeon Phi™ Architecture
- Access additional resources for further information

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

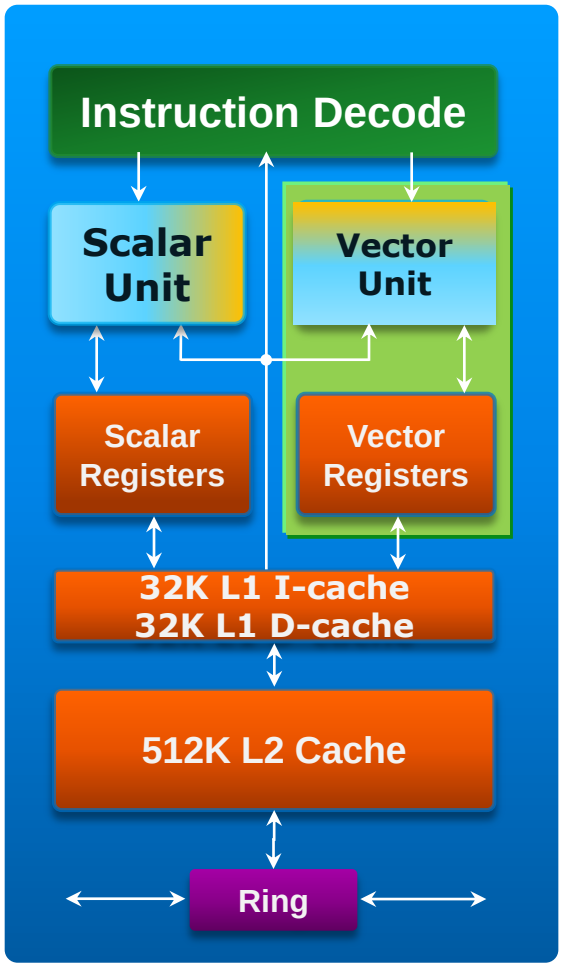
Agenda

- Intel® Many Integrated Core (MIC) Architecture
  - Compiling/Optimizing
- Execution models
  - Offload
  - Native
  - Symmetric
- Tuning

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Intel® Many Integrated Core (MIC) Architecture

Each Intel® Xeon Phi™ Coprocessor core is a fully functional multi-thread execution unit



>50 in-order cores

- Ring interconnect

64-bit addressing

Scalar unit based on Intel® Pentium® processor family

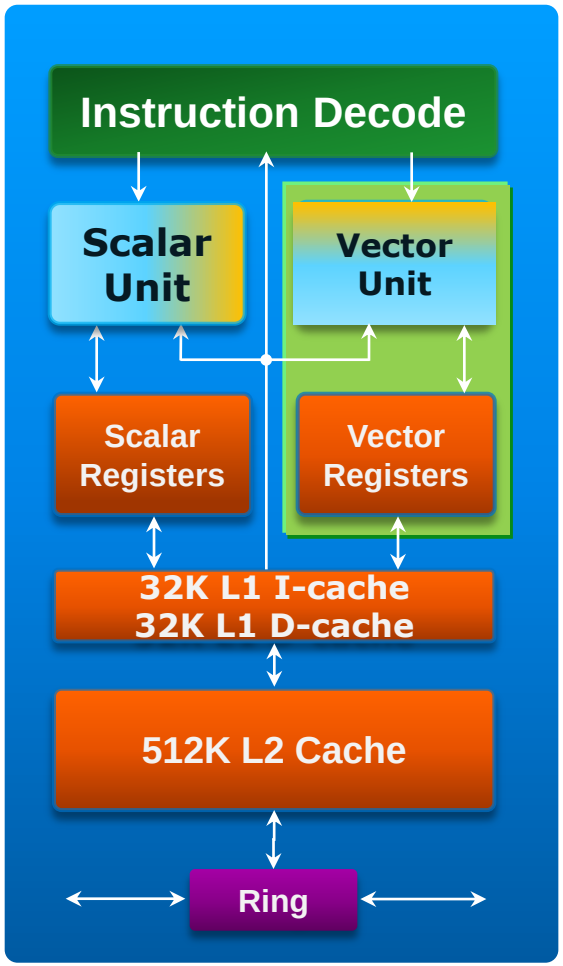
- Two pipelines
  - Dual issue with scalar instructions
- One-per-clock scalar pipeline throughput
  - 4 clock latency from issue to resolution

4 hardware threads per core

- Each thread issues instructions in turn
- Round-robin execution hides scalar unit latency

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Each Intel® Xeon Phi™ Coprocessor core is a fully functional multi-thread execution unit



Optimized

- Single and Double Precision

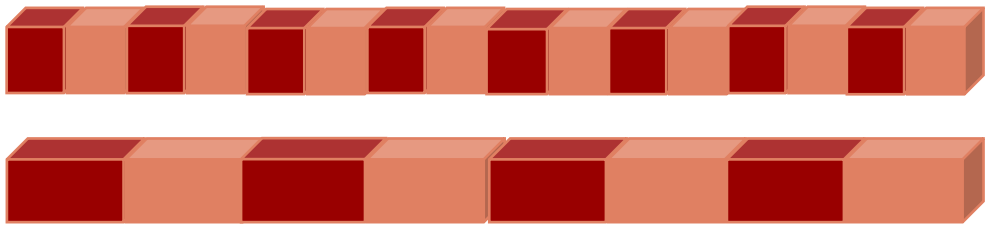
Vector Processing Unit

- 512-bit SIMD Instructions (Not Intel® SSE, MMX™, or Intel® AVX)
- 32 512-bit wide vector registers

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Data Types for Intel® MIC Architecture

now



16x floats

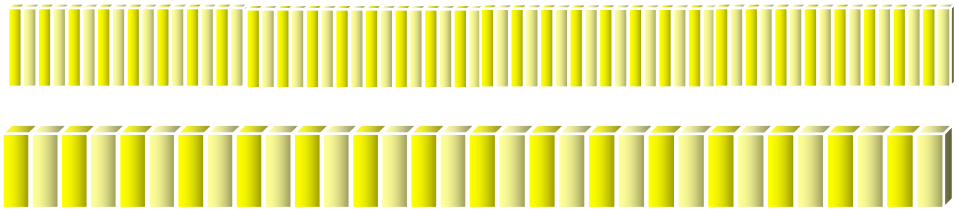
8x doubles

now

16x 32-bit integers

8x 64-bit integers

Not imp-
lemented



64x 8-bit bytes

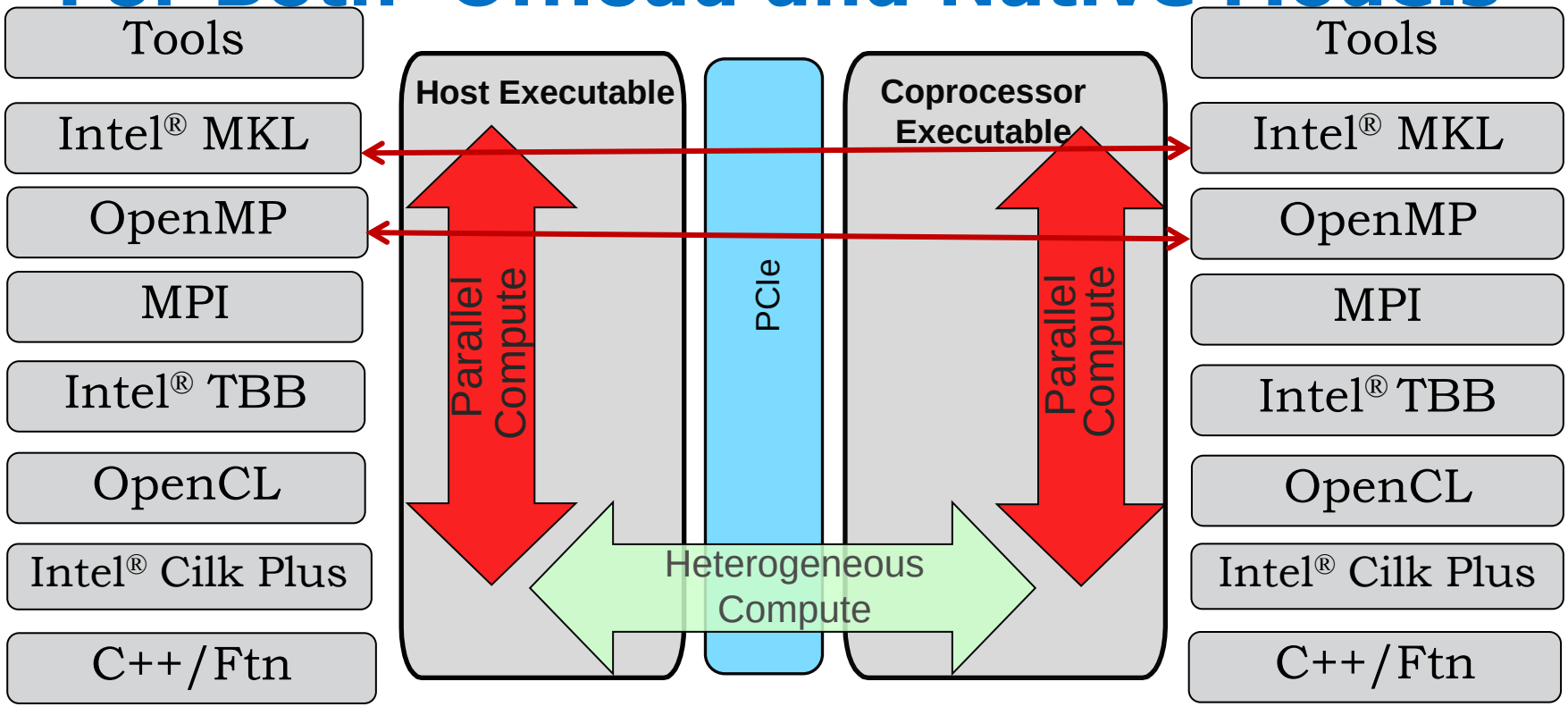
32x 16-bit shorts

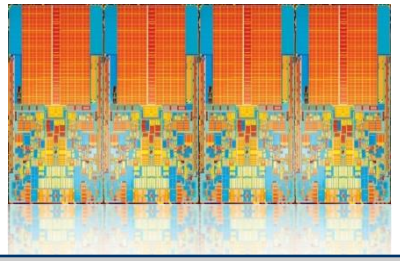
http://software.intel.com/mic-developer

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.

Optimization Notice

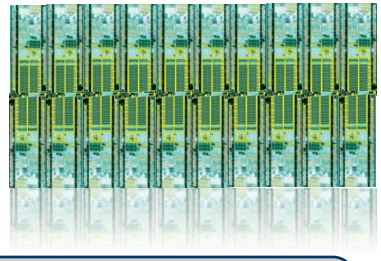
For Both Offload and Native Models





Offload Directives (Data Marshaling)

Offload Keywords (Shared Virtual Memory)



Parallel programming is the same on coprocessor and host

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Taming Intel® Xeon Phi™ with the Intel® Compilers

Key to Peak Intel® Xeon Phi™ Performance

Parallelism

- Vectorization
- Scaling

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

SIMD Support

Vector Classes

- See mic/micvec.h and zmmintrin.h
- #include <immintrin.h>

Auto-Vectorization (align to 512 bits)

Explicit Vector Programming

- Intel® Cilk™ Plus extensions
- OpenMP 4.0 RC1 SIMD constructs

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Auto-Vectorization

Similar on host and coprocessor

Enabled at –O2

Align data to 64 bytes

Many loops can be vectorized

- Masked vector instructions
- Gather/scatter instructions
- Fused multiply-add

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

How to Recognize Vectorization

Vectorization and Optimization Reports

-vec-report2 or –opt-report-phase hpo

Unmasked vector instructions

- No scalar instructions, uses masked vector inst.

Gather/Scatter instructions

Math library calls to libsvml

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Vectorization Reports

By default, both host and target compilations may generate messages for the same loop, e.g.

```
icc -vec-report2 test_vec.c

test_vec.c(10): (col. 1) remark: LOOP WAS VECTORIZED.
test_vec.c(10): (col. 1) remark: *MIC* LOOP WAS VECTORIZED.
```

To get a vectorization report for the offload target compilation, but not for the host compilation:

```
icc -vec-report0 -offload-option,mic,compiler,"-vec-report2
test_vec.c"

test_vec.c(10): (col. 1) remark: *MIC* LOOP WAS VECTORIZED.

test_vec.c(20): (col. 1) remark: *MIC* loop was not vectorized:
existence of vector dependence.

test_vec.c(20): (col. 1) remark: *MIC* PARTIAL LOOP WAS VECTORIZED.
```


Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Vectorization Requirements

See http://software.intel.com/en-us/articles/requirements-for-vectorizable-loops/ for full list

Innermost loop only (a few exceptions)

No function/subroutine calls (unless inline or vector)

No data-dependent exit conditions

Data should be contiguous

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Vectorization Requirements (cont.)

See http://software.intel.com/en-us/articles/requirements-for-vectorizable-loops/ for full list

Pointers can be difficult

Use Directives/Pragmas to help

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Useful Loop Optimization Pragmas/Directives

Pragma/Directive	Purpose
IVDEP	Ignore vector dependency
LOOP COUNT	Advise typical iteration count(s)
UNROLL	Suggest loop unroll factor
DISTRIBUTE POINT	Advise where to split loop
VECTOR …	Vectorization hints
Aligned	Assume data is aligned
Always	Override cost model
Nontemporal	Advise use of streaming stores
NOVECTOR	Do not vectorize
NOFUSION	Do not fuse loops
INLINE/FORCEINLINE	Invite/require function inlining
SIMD ASSERT	"Vectorize or die"

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Data Alignment

Needed for vectorization

Should align on 64 byte boundaries for Intel® MIC Architecture (32 for AVX, 16 for SSE)

Different between C/C++ and Fortran

But the pragmas/directives are similar

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Data Alignment in C/C++

Heap allocation aligned to n-byte boundary

```
void* _mm_malloc(int size, int n);

int posix_memaligned(void **p, size_t n, size_t size);
```

Alignment for variable declarations

```
__attribute__((aligned(n)))  var_name;     //or

__declspec(align(n))  var_name;
```


Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Data Alignment in Fortran

Align array on n-byte boundary (n is power of 2)

- works for dynamic, automatic, and static arrays
- not in common blocks

```
!dir$ attributes align:n :: array
```

For a 2D array, choose column length to be a multiple of n, so consecutive columns have the same alignment (pad if necessary)

-align array64byte compiler tries to align all array types

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Alignment Pragmas/Directives

Ask compiler to vectorize, overriding cost model and assuming all array data access in loop is aligned for target processor. (Could cause fault if data not aligned.)

```
#pragma vector aligned
!dir$ vector aligned
```

Compiler may assume array is aligned to n-byte boundary

```
__asume_aligned(array,n);
!dir$ assume_aligned array:n [,array2:n2,...]
```


Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Explicit Vector Programming:
Intel® Cilk™ Plus Array notation

```
void addit(double* a, double* b,
int m, int n, int x)
{
  for (int i = m; i < m+n; i++)   {
        a[i] = b[i] + a[i-x];
  }
}
```

loop was not vectorized:
existence of vector dependence.

```
void addit(double* a, double * b,
int m, int n, int x)
{
// I know x<0
   a[m:n]  = b[m:n] + a[m-x:n];
}
```

LOOP WAS VECTORIZED.

- Array notation asks the compiler to vectorize
  - asserts this is safe (for example, x<0)
  - Improves readability

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Explicit Vector Programming:
Pragma example

Using #pragma simd (C/C++) or !dir$ SIMD (Fortran) or #pragma omp simd (OpenMP* 4.0 RC1)

```
void addit(double* a, double* b,
int m, int n, int x)
{
  for (int i = m; i < m+n; i++)   {
        a[i] = b[i] + a[i-x];
  }
}
```

loop was not vectorized:
existence of vector dependence.

```
void addit(double* a, double * b,
int m, int n, int x)
{
#pragma simd      // I know x<0
  for (int i = m; i < m+n; i++)   {
        a[i] = b[i] + a[i-x];
  }
}
```

SIMD LOOP WAS VECTORIZED.

- Use when you KNOW that a given loop is safe to vectorize

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

SIMD Summary

The importance of SIMD parallelism is increasing

- Moore's law leads to wider vectors as well as more cores
- Don't leave performance "on the table"
- Be ready to help the compiler to vectorize, if necessary
  - With compiler directives and hints
  - With explicit vector programming
  - Use Intel® VTune™ Amplifier XE to find the best places (hotspots) to focus your efforts
- No need to re-optimize vectorizable code for new processors
  - Typically a simple recompilation

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Adding the Intel® Math Kernel Library to the Mix

Using Intel® Math Kernel Library on Intel® Xeon Phi™ Coprocessors

High level parallelism is critical in maximizing performance.

- BLAS (Level 3) and LAPACK with large problem size get the most benefit.
- Scaling beyond 100's threads, vectorized, good data locality

Minimize data transfer overhead when offload.

- Offset data transfer overhead with enough computation.
- Exploit data persistence: CAO to help!

You can always run on the host if offloading does not offer better performance.

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Usage Models on Intel® Xeon Phi™ Coprocessors

- Automatic Offload
  - No code changes required
  - Automatically uses both host and target
  - Transparent data transfer and execution management
- Compiler Assisted Offload
  - Explicit controls of data transfer and remote execution using compiler offload pragmas/directives
  - Can be used together with Automatic Offload
- Native Execution
  - Uses the coprocessors as independent nodes
  - Input data and binaries are copied to targets in advance

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Automatic Offload (AO)

- Offloading is automatic and transparent.
- Can take advantage of multiple coprocessors.
- By default, Intel® MKL decides:
  - When to offload
  - Work division between host and targets
- Users enjoy host and target parallelism automatically.
- Users can still specify work division between host and target. (for BLAS only)

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Compiler Assisted Offload (CAO)

- Offloading is explicitly controlled by compiler pragmas or directives.
- All Intel® MKL functions can be offloaded in CAO.
  - In comparison, only a subset of Intel® MKL is subject to AO.
- Can leverage the full potential of compiler's offloading facility.
- More flexibility in data transfer and remote execution management.
  - A big advantage is data persistence: Reusing transferred data for multiple operations.

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Native Execution

- Use the coprocessor as an independent compute node.
  - Programs can be built to run only on the coprocessor by using the `-mmic` build option.
- Intel® MKL function calls inside an offloaded code region executes natively.
  - Better performance if input data is already available on the coprocessor, and output is not immediately needed on the host side.

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Clustering, with the Intel® MPI Library

Using the Intel® MPI Library with Intel® Xeon Phi™

Three execution models supported

- Offload
- Native
- Symmetric

Mostly similar to standard MPI programming

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Job Schedulers

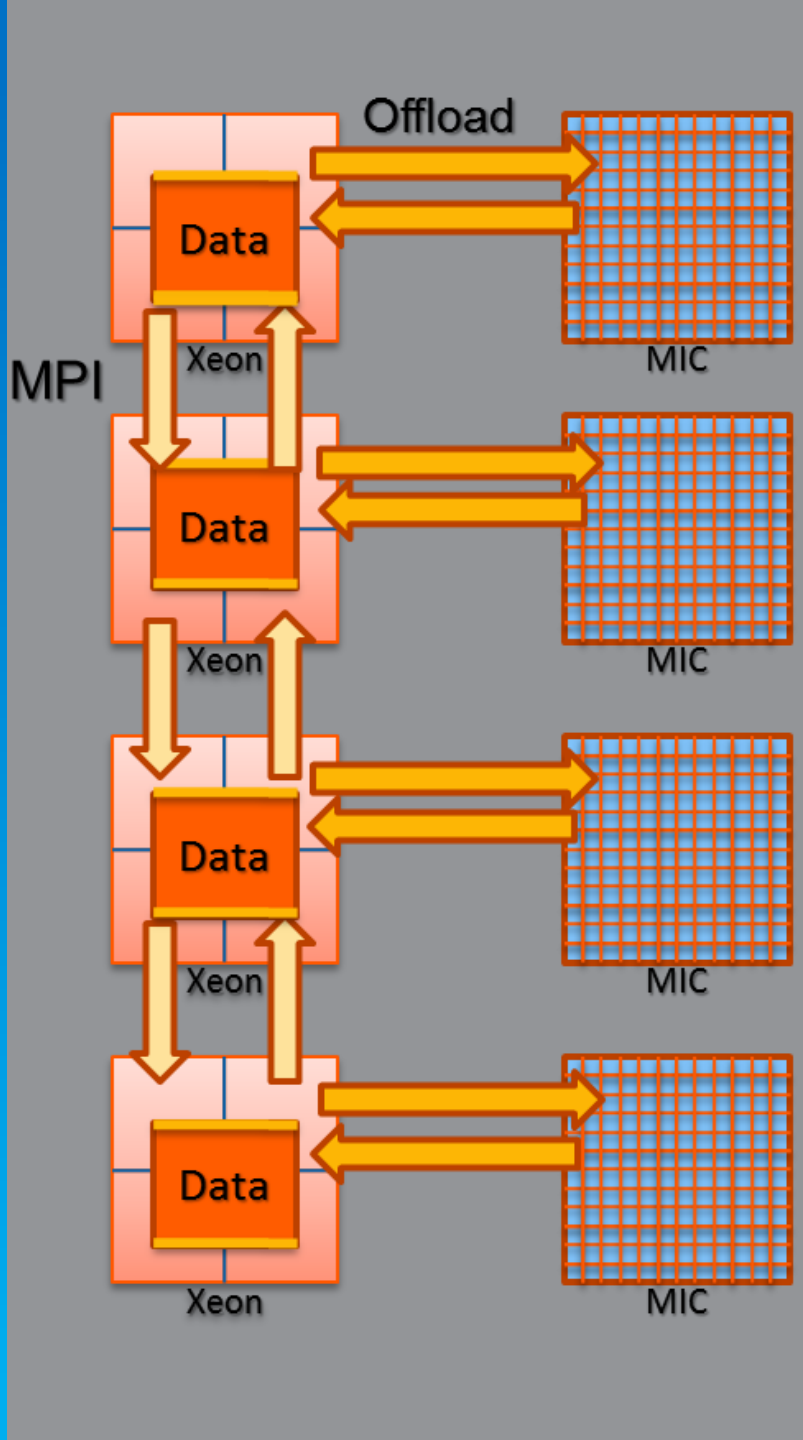
Job schedulers do not currently support Intel® Xeon Phi™

Best approach depends on chosen execution model

Usually involves additional scripts to manipulate hostfile and/or environment variables

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

The Execution Models - Offload



Offload Programming

Copies a portion of data and instructions to coprocessor

Executes on coprocessor

Copies data back to host

Can execute highly parallelized/vectorized code sections on coprocessor, serial sections on host

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

How to Offload?

Pragmas

Explicit Offloading

Automatic Offloading (MKL)

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Managing Offload

Target selection

- Offload target specified at compile-time or at run-time

Thread pinning

- Environment variables used to control thread pinning
- Can get complex, use –env option to help

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Offload and Schedulers

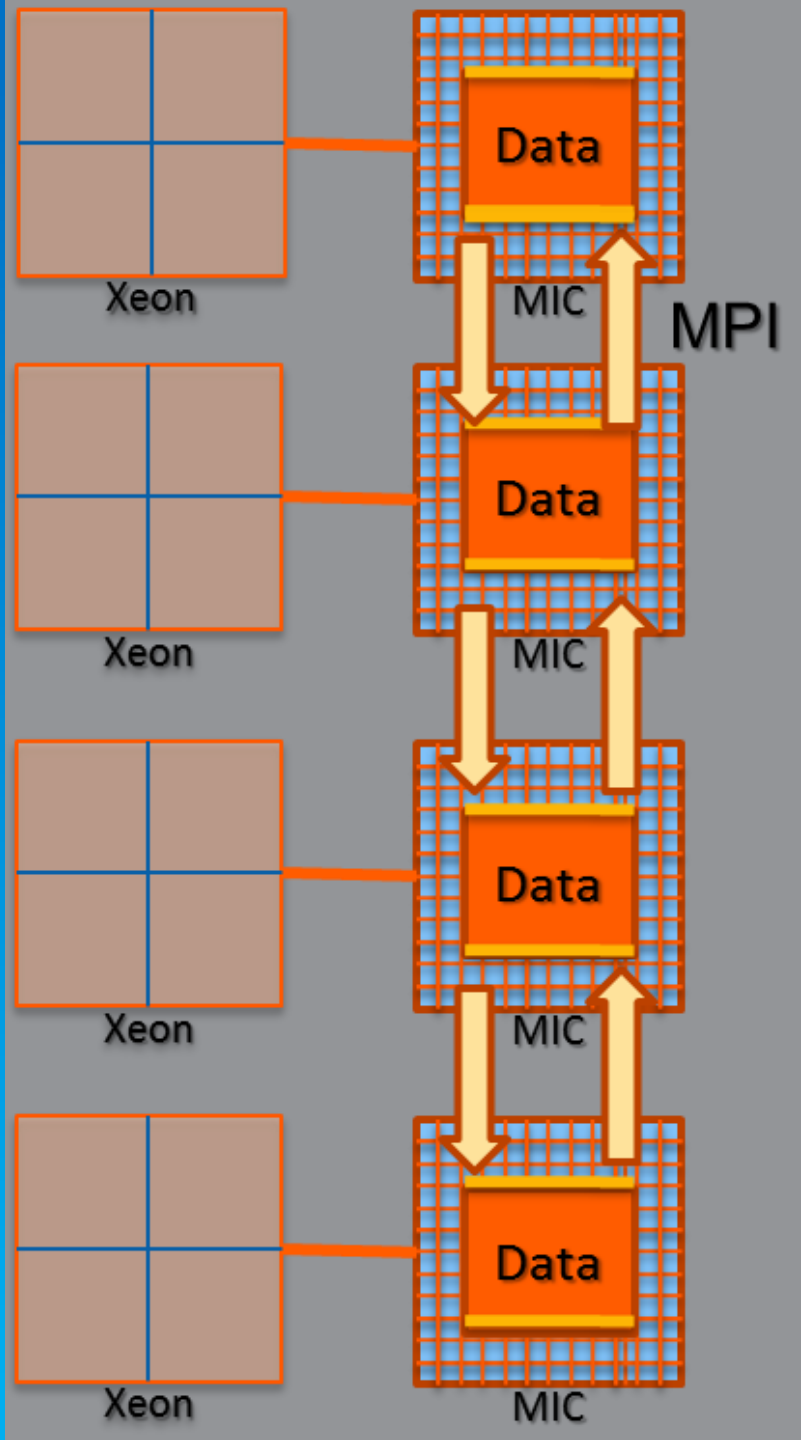
If running one rank per host, no changes

If running multiple ranks per host, pass/set appropriate environment variables to control offload targets/pinning

Remember rank placement!

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

The Execution Models - Native



Native Programming

Cross-compile to Intel® MIC Architecture

Make binary and any dynamic libraries available on the coprocessor

Treat coprocessor as a unique system

No need to modify code to run on coprocessor

Good if code is already threaded

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Considerations

Memory limitations

Recommended to run at most one rank per physical core, let threading utilize the rest

Avoid cores reserved for operating system

- 0, n-2, n-1, n

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

How to Use Native Model

Build for Intel® Xeon Phi™

```
mpiicc -mmic test.c -o hello.mic
```

Ensure binary is accessible on coprocessor

```
scp hello.mic node0-mic0:/home/user/test/hello
```

Launch application from host or coprocessor

```
$export I_MPI_MIC=enable
$cat mpi_hosts
node0-mic0
node1-mic0
$mpirun -n 2 -ppn 1 -hostfile mpi_hosts ./hello
Hello from rank 0 of 2 running on node0-mic0
Hello from rank 1 of 2 running on node1-mic0
```


Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Job Schedulers and Native Execution

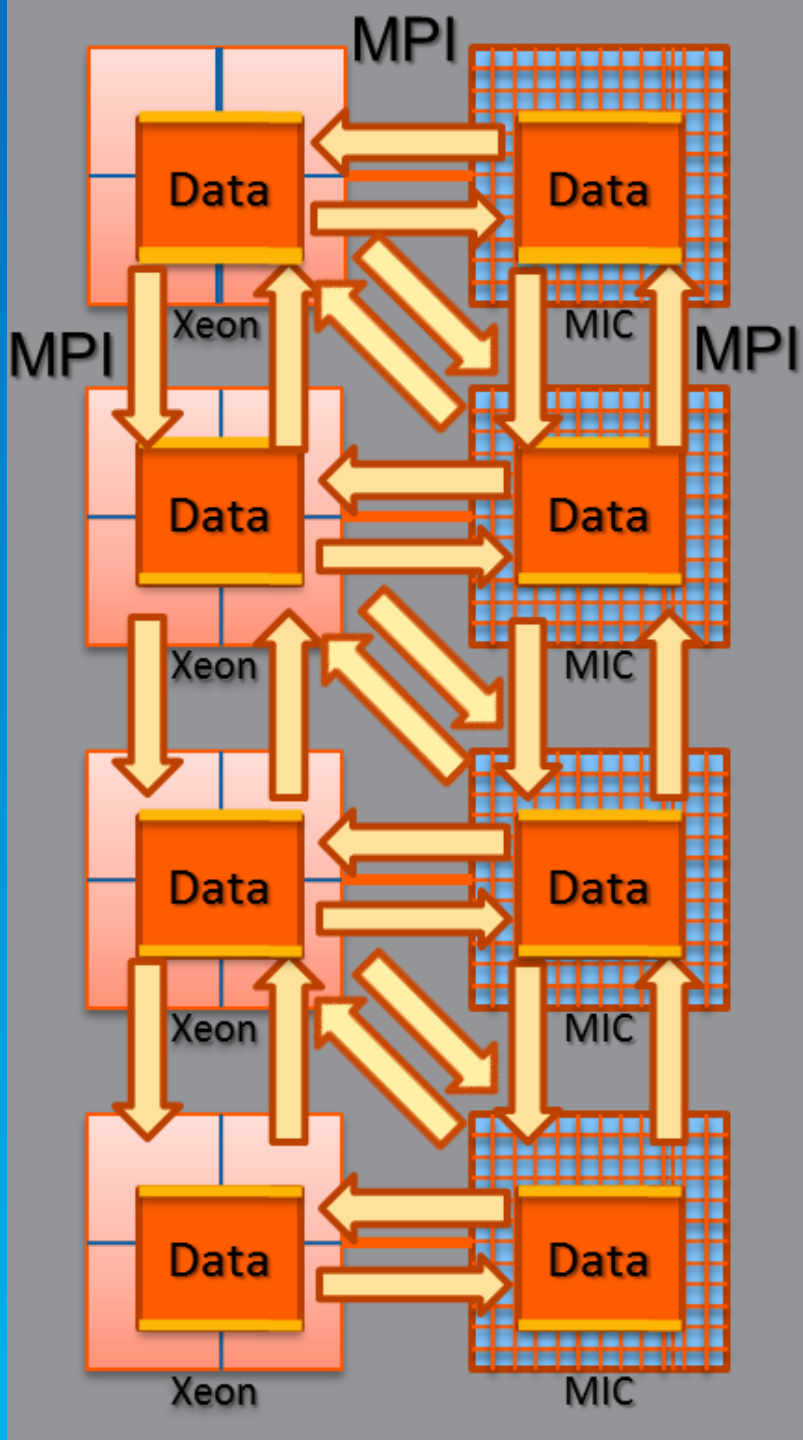
Job schedulers will likely generate host file for hosts, not coprocessors

Need to modify host file for coprocessors

If not using NFS, might need to copy binaries and/or data files as part of job

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

The Execution Models - Symmetric



Symmetric Model

Ranks on both host and coprocessor

Heterogeneous runs

Best if serial code runs on host and parallel code runs on coprocessor

Can be run with no code modification

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.

Optimization Notice

Symmetric Programming Model

Compile for both host and coprocessor

```
mpiicc test.c -o hello

mpiicc -mmic test.c -o hello.mic
```

Ensure binary is accessible on execution host

```
scp hello.mic node0-mic0:/home/user/test/hello
```

Launch the application on both

```
export I_MPI_MIC=enable

mpirun -hosts node0,node0-mic0 -n 2 -ppn 1 ./hello
```


Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Additional support for NFS shares

I_MPI_MIC_POSTFIX – Adds the text to the end of the executable if execution host is a coprocessor

I_MPI_MIC_PREFIX – Add the text to the end of the executable if execution host is a coprocessor

These only work if using –hostfile or -machinefile

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Example of NFS support

Begin in a shared folder with same path on host and coprocessor

```
$mpiicc test.c -o hello
$mpiicc -mmic test.c -o hello.mic
$export I_MPI_MIC=1
$export I_MPI_MIC_POSTFIX=.mic
$cat hosts
node0
node0-mic0
$mpirun -n 2 -ppn 1 -hostfile hosts ./hello
Hello from rank 0 of 2 running on node0
Hello from rank 1 of 2 running on node0-mic0
```


Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Argument Sets and Configuration Files

Use to run heterogeneous jobs

Global options applied to all sets

Local options only applied to current set

Pass different environment variables to each set

Run different executables in each set

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Configuration Files

Store sets in a file, one per line

```
cat <configfile>

-genv I_MPI_DEBUG 2

-n 1 -host node000 ./hello.host

-n 1 -host node000-mic0 ./hello.mic

mpirun -configfile <configfile>
```


Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Argument sets

Separate with :

```
mpirun -n 1 -host node000 /home/user/test/printenv.host : -n 1 -host
node000-mic0 -env TESTVAR 25 /home/user/test/printenv.mic

Rank 0 of 1 running on node000, TESTVAR is not defined

Rank 1 of 1 running on node000-mic0, TESTVAR=25
```


Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Job Schedulers and Symmetric Execution

Similar to native execution, host file will likely only include hosts, no coprocessors

Again, need to modify host file, but consider order of hosts

Consider building a configuration file from the host file

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Tuning – Intel® Xeon Phi™ Style

Where to Begin?

Start on the host

- Inspector XE currently only supports Intel® Xeon™
- Some VTune™ Amplifier XE reports are not supported on Intel® Xeon Phi™

Errors (races, deadlocks, etc.) will be the same on both architectures

Disable offloading in initial collections

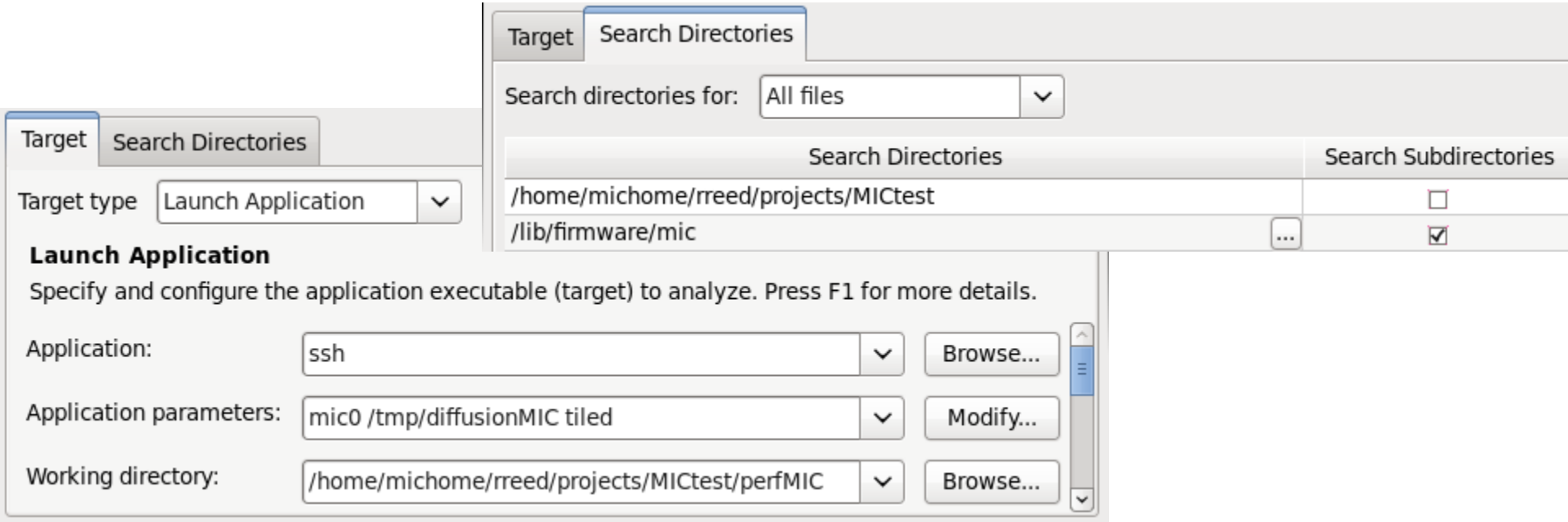
Remember load balancing

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Collect Xeon Phi™ Events

VTune™ Amplifier XE settings:

- Application: ssh
- Parameters: node0-mic0 <application and args>
- Working Directory: Usually won't matter (ssh)
- Remember to set search directories



Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Coprocessor Collections Have Separate Analysis Types

r000lh | r001lh | r002lh | r003lh | New Amplifier XE Result

Choose Analysis Type

Intel VTune Amplifier XE 2013

Analysis Type

- Access Contention
- Branch Analysis
- Client Analysis
- Core Port Saturation
- Cycles and uOps
- Loop Analysis
- Memory Access
- Port Saturation
- Intel Atom Processor Analysis
  - General Exploration
- Knights Corner Platform Analysis
  - Lightweight Hotspots
- Power Analysis
  - CPU Sleep States
  - CPU Frequency
- Custom Analysis
  - Branch Mispredicts
  - Cache Misses
  - Execution Stalls
  - False Sharing

Knights Corner Platform - Lightweight Hotspots

Copy

Identify your most time-consuming source code. Unlike Hotspots, Lightweight Hotspots has lower overhead because it does not collect stack information. It can also be used to sample all processes on a system. This analysis type uses hardware event-based sampling collection. Press F1 for more details.

List of MIC cards (e.g. 0,1,2,3): 0

Details

Events configured for CPU: Intel(R) Xeon(R) / Core i7 980X Processor

NOTE: For analysis purposes, Intel VTune Amplifier XE 2013 may adjust the Sample After values in the table below by a multiplier. The multiplier depends on the value of the Duration time estimate option specified in the

Event Name	Sample After	Event De
CPU_CLK_UNHALTED	10000000	Number of cycles during whicl
INSTRUCTIONS_EXECUTED	10000000	Number of instructions execut

Start

Start Paused

Project Properties

Command Line...

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.

Optimization Notice

Performance Analysis

Only "Lightweight Hotspots" for Intel® Xeon Phi™

- Uses Event-Based Sampling
- Uses the Performance Monitoring Unit
- No instrumentation ("Locks & Waits", "Concurrency", etc.)
- More to come!

Other analysis types need to be configured

- Create a copy of Lightweight Hotspots
- Add desired counters
- Add useful name and documentation
- Multi-event collections (over 2) can multiplex or use multiple runs

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Configuring a User-Defined Analysis

Analysis name: My Shiny New KNC Analysis

Description: This is the greatest analysis I ever created!

Properties

Events configured for CPU: Intel(R) Xeon(R) / Core i7 980X Processor

NOTE: For analysis purposes, Intel VTune Amplifier XE 2013 may adjust the Sample After values in the tab by a multiplier. The multiplier depends on the value of the Duration time estimate option specified in the Pr Properties dialog.

Event Name	Sample After	Event Description
CPU_CLK_UNHALTED	10000000	Number of cycles during which the processor is not halte
INSTRUCTIONS_EXECUTED	10000000	Number of instructions executed (up to two per clock)

Add Event | Remo

List of MIC cards (e.g. 0,1,2,3): 0

Command line name: shiny-new-analysis

Analysis identifier: sna

OK

Add Events <@mklemm-Inc>

Available events:

Filter (use '*' and '?'. e.g. br*retired) :

Clear Filter

- [ ] BANK_CONFLICTS
- [ ] BRANCHES
- [ ] BRANCHES_MISPREDICTED
- [ ] CODE_CACHE_MISS
- [ ] CODE_READ
- [ ] DATA_CACHE_LINES_WRITTEN_BACK
- [x] DATA_PAGE_WALK
- [ ] DATA_READ
- [x] DATA_READ_MISS
- [ ] DATA_READ_MISS_OR_WRITE_MISS
- [ ] DATA_READ_OR_WRITE
- [ ] DATA_WRITE
- [ ] DATA_WRITE_MISS
- [ ] EXEC_STAGE_CYCLES

Number of memory read accesses that miss the internal

Explain...

Check All | Check None

OK | Cancel

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.

Optimization Notice

Hardware Counters and Events

2 counters in core, most are thread specific

4 outside the core (uncore) that get no thread or core details

See PMU documentation for full list

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Collecting Hardware Performance Data

Invoke from VTune™ Amplifier XE

If collecting more than 2 core events:

- Select multi-run for more precise results OR
- Default multiplexed collection, all in one run

Uncore events are limited to 4 at a time in a single run

Uncore event sampling needs a source of PMU interrupts (programming cores to CPU_CLK_UNHALTED)

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Efficiency Metrics

CPI – Cycles Per Instruction

- Lower is better
- Minimum depends on number of hardware threads used
- Investigate if >4.0 per thread or >1.0 per core (or if increasing)

Compute to Data Access Ratio

- Two measures of computational density
- Investigate if
  - L1 Compute to Data Access Ratio < Vectorization Intensity
  - L2 Compute to Data Access Ratio < 100x L1 Compute to Data Access Ratio

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Problem Areas

Vectorization Intensity

- Investigate if <8 for double precision, <16 for single

L1 Hit Rate

- Investigate if <95%

Estimated Latency Impact

- Investigate if >145

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Problem Areas

TLB Usage

- Investigate if
  - L1 TLB Miss Ratio > 1%
  - L2 TLB Miss Ratio > 0.1%
  - L1 TLB Misses Per L2 TLB Miss > 100x

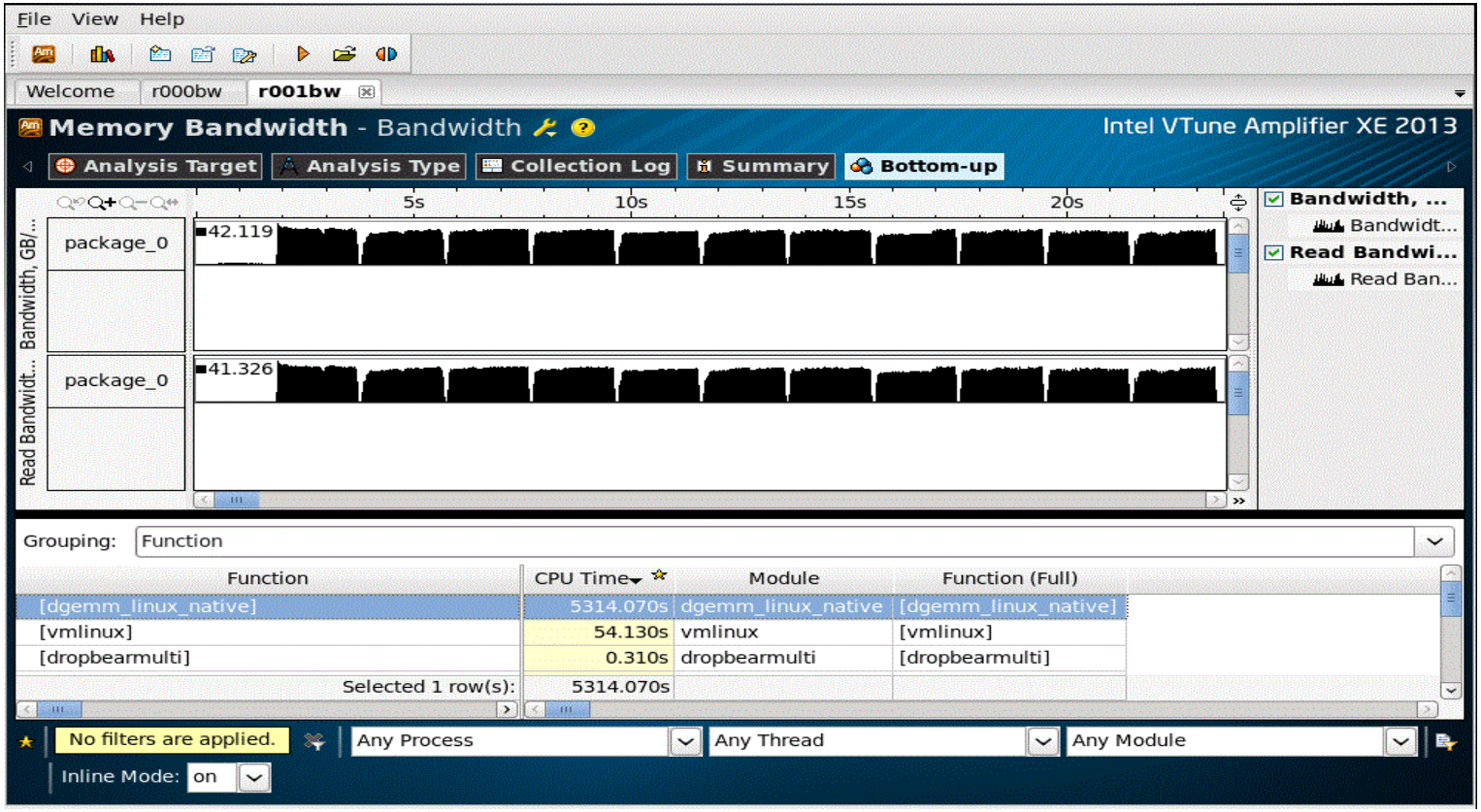
Memory Bandwidth

- Investigate if < 80 GB/s

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Event collections on the coprocessor can generate volumes of data

dgemm: on 60+ cores



Tip: Use cpu-mask to reduce data set, while maintaining the same accuracy.

http://software.intel.com/mic-developer
Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Resources

http://software.intel.com/mic-developer

http://software.intel.com/en-us/articles/programming-and-compiling-for-intel-many-integrated-core-architecture

http://software.intel.com/en-us/articles/intel-mpi-library-and-intel-xeon-phi-coprocessors

http://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-1-optimization

http://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-2-understanding

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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.

Copyright © , Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Xeon Phi, Core, VTune, and Cilk are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers 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 SSE2, SSE3, and 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. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Copyright© 2012, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.

intel®

Backup Slides

Vector Elemental Function

Compiler generates vector version of a scalar function that can be called from a vectorized loop:

```
__attribute__((vector(uniform(y, xp, yp))))
                    float func(float x, float y, float xp, float yp)  {
  float denom = (x-xp)*(x-xp) + (y-yp)*(y-yp);
  denom= 1./sqrtf(denom);
  return denom;
}
```

y, xp and yp are constant, x can be a vector

func_vec.f(1): (col. 21) remark: FUNCTION WAS VECTORIZED.

```
#pragma simd  private(x)  reduction(+:sumx)
      for (i=1; i<nx; i++)  {
        x = x0 + (float)i * h;
        sumx = sumx + func(x, y ,xp, yp);
      enddo
```

These clauses are required for correctness, just like for OpenMP*

SIMD LOOP WAS VECTORIZED.

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Clauses for Vector Functions

__attributes__((vector)) (Intel)

#pragma omp declare simd (OpenMP* 4.0 RC1)

Available clauses (both OpenMP and Intel versions)

- LINEAR (additional induction variables)
- UNIFORM (arguments that are loop constants)
- PROCESSOR (Intel)
- VECTORLENGTH (Intel)
- MASK / NOMASK (Intel)
- INBRANCH / NOTINBRANCH (OpenMP 4.0 RC1)
- SIMDLEN (OpenMP 4.0 RC1)
- ALIGNED (OpenMP 4.0 RC1)

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Efficiency Metric: CPI

- Cycles per Instruction
- Can be calculated per hardware thread or per core
- Is a ratio! So varies widely across applications and should be used carefully.

Number of Hardware Threads / Core	Minimum (Best) Theoretical CPI per Core	Minimum (Best) Theoretical CPI per Thread
1	1.0	1.0
2	0.5	1.0
3	0.5	1.5
4	0.5	2.0

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Efficiency Metric: CPI

- Measures how latency affects your application's execution

Metric	Formula	Investigate if
CPI Per Thread	CPU_CLK_UNHALTED/ INSTRUCTIONS_EXECUTED	> 4.0, or increasing
CPI Per Core	(CPI per Thread) / Number of hardware threads used	> 1.0, or increasing

- Look at how optimizations applied to your application affect CPI
- Address high CPIs through any optimizations that aim to reduce latency

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Efficiency Metric: Compute to Data Access Ratio

- Measures an application's computational density, and its suitability for Intel® MIC Architecture

Metric	Formula	Investigate if
L1 Compute to Data Access Ratio	VPU_ELEMENTS_ACTIVE / DATA_READ_OR_WRITE	< Vectorization Intensity
L2 Compute to Data Access Ratio	VPU_ELEMENTS_ACTIVE / DATA_READ_MISS_OR_ WRITE_MISS	< 100x L1 Compute to Data Access Ratio

- Increase computational density through vectorization and reducing data access (see cache issues, also, DATA ALIGNMENT!)

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Problem Area: VPU Usage

- Indicates whether an application is vectorized successfully and efficiently

Metric	Formula	Investigate if
Vectorization Intensity	VPU_ELEMENTS_ACTIVE / VPU_INSTRUCTIONS_EXECUTED	<8 (DP), <16(SP)

- Tuning Suggestions:
  - Use the Compiler vectorization report!
  - For data dependencies preventing vectorization, try using Intel® Cilk™ Plus #pragma SIMD (if safe!)
  - Align data and tell the Compiler!
  - Re-structure code if possible: Array notations, AOS->SOA

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Problem Area: L1 Cache Usage

- Significantly affects data access latency and therefore application performance

Metric	Formula	Investigate if
L1 Misses	DATA_READ_MISS_OR_WRITE_MISS + L1_DATA_HIT_INFLIGHT_PF1	
L1 Hit Rate	(DATA_READ_OR_WRITE – L1 Misses) / DATA_READ_OR_WRITE	< 95%

- Tuning Suggestions:
  - Software prefetching
  - Tile/block data access for cache size
  - Use streaming stores
  - If using 4K access stride, may be experiencing conflict misses
  - Examine Compiler prefetching (Compiler-generated L1 prefetches should not miss)

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Problem Area: Data Access Latency

- Significantly affects application performance

Metric	Formula	Investigate if
Estimated Latency Impact	(CPU_CLK_UNHALTED – EXEC_STAGE_CYCLES – DATA_READ_OR_WRITE) / DATA_READ_OR_WRITE_MISS	>145

- Tuning Suggestions:
  - Software prefetching
  - Tile/block data access for cache size
  - Use streaming stores
- Check cache locality – turn off prefetching and use CACHE_FILL events - reduce sharing if needed/possible
- If using 64K access stride, may be experiencing conflict misses

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Problem Area: TLB Usage

- Also affects data access latency and therefore application performance

Metric	Formula	Invest-igate if
L1 TLB miss ratio	DATA_PAGE_WALK/DATA_READ_OR_WRITE	> 1%
L2 TLB miss ratio	LONG_DATA_PAGE_WALK / DATA_READ_OR_WRITE	> .1%
L1 TLB misses per L2 TLB miss	DATA_PAGE_WALK / LONG_DATA_PAGE_WALK	> 100x

- Tuning Suggestions:
  - Improve cache usage & data access latency
  - If L1 TLB miss/L2 TLB miss is high, try using large pages
  - For loops with multiple streams, try splitting into multiple loops
  - If data access stride is a large power of 2, consider padding between arrays by one 4 KB page

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice

Problem Area: Memory Bandwidth

- Can increase data latency in the system or become a performance bottleneck

Metric	Formula	Investigate if
Memory Bandwidth	(UNC_F_CH0_NORMAL_READ + UNC_F_CH0_NORMAL_WRITE+ UNC_F_CH1_NORMAL_READ + UNC_F_CH1_NORMAL_WRITE) X 64/time	< 80GB/sec (practical peak 140GB/sec) (with 8 memory controllers)

- Tuning Suggestions:
  - Improve locality in caches
  - Use streaming stores
  - Improve software prefetching

Copyright© 2013, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Optimization Notice