Agenda

New User to Intel Compilers

What is Xeon Phi

Compiling for Xeon phi

Vectorize your code

Tips and tricks

Code the Future
What is Xeon Phi

The Intel® Xeon Phi™ Coprocessor has up to 61 inorder Intel® MIC Architecture processor cores running at 1GHz (up to 1.3GHz).

The Intel® MIC Architecture is based on the x86 ISA, extended with 64-bit addressing and new 512-bit wide SIMD vector instructions and registers.

Each core supports 4 hardware threads
Why Use Intel Compilers?

Compatibility

Platforms: Source and binary compatible with
- Visual C++ 2008/2010/2012/2013 on Windows*
- gcc 4.1 ~ gcc 4.8 on Linux*
- Xcode* 4.6 or 5.0 on OS X*

ANSI C/C++ and OpenMP* compliance:
- ISO/IEC 9899:1990 for C language
- C++ ISO/IEC 14882:2011 for C++11
- Partial support of OpenMP* 4.0

Performance

Industry leading optimization technologies: auto-vectorization, PGO, IPO, processor targeting optimization

Outstanding performance on Intel® architecture processors

Performance libraries: Intel® IPP, Intel® MKL and Intel® TBB

Support

World class support with secure, web-based, engineer-to-engineer support through Intel Premier Support

Community based forum support from technical experts around the world

Code the Future
Why Use Intel® Compilers?

Parallelism

Numerous tools to enable parallelism

Vector Parallelism

- Automatic Vectorization
- Vector statements (Intel® Cilk Plus)
- Lower level SIMD (pragmas, intrinsic functions)

Task Parallelism

- Language extensions (Intel® Cilk Plus)
- C++ Task Libraries (Intel® TBB)
- Automatic Parallelism
- GAP – let the compiler help you restructure code for more parallelism opportunities

Multi-threaded Performance Libraries – Intel® MKL, Intel® IPP

Code the Future
Why Use Intel® Compilers?

Performance

Our goal is performance

Performance to be gained in a variety of ways:

- The future is Multi-core (and the future is now!)
- New instructions enable new opportunities (SSE, AVX, AVX2)
- Micro architectural improvements

Intel Compilers Support the latest Features

- Be on the cutting edge of new performance features
  - Latest Instructions
    - Code generation tuned for latest microarchitecture

Highly Optimized libraries

- MKL – Math functions (BLAS, FFT, LAPACK, &c)
- IPP – (compression, video encoding, image processing &c)

Code the Future
Linux: Basic Compiler Usage

source <installdir>/bin/[compilervars.sh | compilervars.csh] [intel64 | ia32]

Sets environment vars for compiler, libraries, headers, etc.

Compiler drivers are ‘ifort’, ‘icc’ for C, and ‘icpc’ for C++

“-O” switches compatible, but not identical to gcc

- O2 default optimization level (gcc default is –O0)
- O* doesn’t imply the same set of opts for gcc and Intel, but similar concepts, -O0 for debugging, -O2 for default, -O3 for more advanced optimizations

icc –help, icpc -help or ifort -help provides extensive list

Intel debugger IDB or Intel-provided GDB (extended)

Linux: ‘idbc’ command line, ‘idb’ X11 GUI
Compiling for Xeon Phi

• **Native Mode**
  Easiest to start
  Add the –mmic flag to the compiler

• **Offload**
  No need for any extra flags
  Specify sections of code to be run on Xeon Phi through pragmas
Compiling for Xeon Phi

• Native Mode
  icpc -mmic -vec-report3 -openmp
  omp_native.cpp

```
omp_native.cpp(126): (col. 3) remark: loop was not vectorized: not inner loop
omp_native.cpp(126): (col. 3) remark: loop was not vectorized: not inner loop
omp_native.cpp(52): (col. 4) remark: LOOP WAS VECTORIZED
omp_native.cpp(52): (col. 4) remark: PEEL LOOP WAS VECTORIZED
omp_native.cpp(52): (col. 4) remark: REMAINDER LOOP WAS VECTORIZED
omp_native.cpp(51): (col. 3) remark: loop was not vectorized: not inner loop
omp_native.cpp(50): (col. 2) remark: loop was not vectorized: not inner loop
```
Compiling for Xeon Phi

- **Native Mode**
  
  Copy compiled binary over to Xeon Phi
  
  ```
  sudo scp a.out mic0:/tmp
  ```
  
  Copy over *libiomp5.so* as well
  
  ```
  sudo scp /opt/intel/lib/mic/libiomp5.so mic0:/tmp
  ```
  
  ```
  sudo ssh mic0
  ```
Compiling for Xeon Phi

• Native Mode
  Run a.out
  Fails because it can’t find the openmp library

  export LD_LIBRARY_PATH=/tmp

  Depending on what libraries you use you will need to copy those into your path as well.
Compiling for Xeon Phi

- **Offload mode**
  Useful for complex applications where you only need heavy compute during certain times.

  Can control what gets run on the coprocessor.
Compiling for Xeon Phi

- Offload mode
  icpc -openmp -O3 -xhost offload.cpp

  No need copy over files to the coprocessor.

  Runs on host.
Compiling for Xeon Phi

• Offload mode

#pragma offload target(mic){
/* code in here runs on xeon phi
It will run on the host if there is no coprocessor available
*/
}

Code the Future
Profiling
Function level and/or Loop-level
Function and Loop Profiler
Identify Time Consuming Functions and Loops

Compiler switch:
-`profile-functions`  /Qprofile-functions, Insert instrumentation calls on function entry and exit points to collect the cycles spent within the function.

Compiler switch:
-`profile-loops= <inner|outer|all>`  /Qprofile-loops=<inner|outer|all>
  Insert instrumentation calls for function entry and exit points as well as the instrumentation before and after instrument able loops of the type listed as the option’s argument.

Loop Profiler switches trigger generation of text (.dump) and XML (.xml) output files

Invocation of XML on command line:
```
java -jar loopprofviewer.jar <xml datafile>
```
<table>
<thead>
<tr>
<th>time(abs)</th>
<th>time(%)</th>
<th>self(abs)</th>
<th>self(%)</th>
<th>call_count</th>
<th>exit_count</th>
<th>loop_ticks(%)</th>
<th>file:line</th>
</tr>
</thead>
<tbody>
<tr>
<td>4378070322</td>
<td>99.83</td>
<td>1810826157</td>
<td>41.29</td>
<td>1</td>
<td>1</td>
<td>41.29</td>
<td>deflate.c:623</td>
</tr>
<tr>
<td>6474995316</td>
<td>14.76</td>
<td>6428298878</td>
<td>14.66</td>
<td>16768796</td>
<td>16768796</td>
<td>0.01</td>
<td>trees.c:961</td>
</tr>
<tr>
<td>1462208444</td>
<td>33.34</td>
<td>4877549666</td>
<td>11.12</td>
<td>10546669</td>
<td>10546669</td>
<td>7.07</td>
<td>deflate.c:360</td>
</tr>
<tr>
<td>1197744296</td>
<td>2.73</td>
<td>1196868960</td>
<td>2.73</td>
<td>513</td>
<td>513</td>
<td>2.73</td>
<td>util.c:63</td>
</tr>
<tr>
<td>1376532400</td>
<td>3.13</td>
<td>89563374</td>
<td>2.04</td>
<td>512</td>
<td>512</td>
<td>2.04</td>
<td>bits.c:185</td>
</tr>
<tr>
<td>1982539400</td>
<td>4.52</td>
<td>66623288</td>
<td>1.52</td>
<td>512</td>
<td>512</td>
<td>1.46</td>
<td>deflate.c:478</td>
</tr>
<tr>
<td>471568228</td>
<td>1.08</td>
<td>47102278</td>
<td>1.07</td>
<td>1025</td>
<td>1025</td>
<td>0.00</td>
<td>util.c:153</td>
</tr>
<tr>
<td>65947871</td>
<td>1.59</td>
<td>32157819</td>
<td>0.73</td>
<td>344059</td>
<td>344059</td>
<td>0.66</td>
<td>trees.c:454</td>
</tr>
<tr>
<td>115928920</td>
<td>2.73</td>
<td>24484703</td>
<td>0.56</td>
<td>1536</td>
<td>1536</td>
<td>0.12</td>
<td>trees.c:611</td>
</tr>
<tr>
<td>132122614</td>
<td>3.01</td>
<td>12346758</td>
<td>0.28</td>
<td>513</td>
<td>513</td>
<td>0.00</td>
<td>zip.c:106</td>
</tr>
<tr>
<td>229042224</td>
<td>0.52</td>
<td>6738036</td>
<td>0.15</td>
<td>1537</td>
<td>1537</td>
<td>0.15</td>
<td>trees.c:571</td>
</tr>
<tr>
<td>6675927</td>
<td>0.15</td>
<td>6672487</td>
<td>0.15</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:1511</td>
</tr>
<tr>
<td>15997356</td>
<td>0.36</td>
<td>6029850</td>
<td>0.14</td>
<td>139288</td>
<td>139288</td>
<td>0.12</td>
<td>bits.c:151</td>
</tr>
<tr>
<td>2792164</td>
<td>0.06</td>
<td>2620132</td>
<td>0.06</td>
<td>1536</td>
<td>1536</td>
<td>0.06</td>
<td>trees.c:485</td>
</tr>
<tr>
<td>1290621</td>
<td>0.03</td>
<td>1175933</td>
<td>0.03</td>
<td>1024</td>
<td>1024</td>
<td>0.03</td>
<td>trees.c:699</td>
</tr>
<tr>
<td>495976</td>
<td>0.01</td>
<td>387226</td>
<td>0.01</td>
<td>513</td>
<td>513</td>
<td>0.01</td>
<td>trees.c:408</td>
</tr>
<tr>
<td>259246700</td>
<td>5.91</td>
<td>201552</td>
<td>0.01</td>
<td>512</td>
<td>512</td>
<td>0.00</td>
<td>trees.c:857</td>
</tr>
<tr>
<td>47392247</td>
<td>1.08</td>
<td>162869</td>
<td>0.00</td>
<td>1025</td>
<td>1025</td>
<td>0.00</td>
<td>util.c:121</td>
</tr>
<tr>
<td>52254086</td>
<td>0.12</td>
<td>113633</td>
<td>0.00</td>
<td>512</td>
<td>512</td>
<td>0.00</td>
<td>trees.c:791</td>
</tr>
<tr>
<td>4385684262</td>
<td>100.00</td>
<td>66840</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:424</td>
</tr>
<tr>
<td>630948</td>
<td>0.01</td>
<td>56083</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>deflate.c:289</td>
</tr>
<tr>
<td>4385610543</td>
<td>100.00</td>
<td>35086</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:704</td>
</tr>
<tr>
<td>6711753</td>
<td>0.15</td>
<td>32771</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:853</td>
</tr>
<tr>
<td>82583</td>
<td>0.00</td>
<td>23661</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>trees.c:335</td>
</tr>
<tr>
<td>53760</td>
<td>0.00</td>
<td>22140</td>
<td>0.00</td>
<td>513</td>
<td>513</td>
<td>0.00</td>
<td>bits.c:164</td>
</tr>
<tr>
<td>4378817661</td>
<td>99.84</td>
<td>19622</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>zip.c:35</td>
</tr>
<tr>
<td>26175</td>
<td>0.00</td>
<td>13585</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:1605</td>
</tr>
<tr>
<td>12528</td>
<td>0.00</td>
<td>12466</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:1583</td>
</tr>
<tr>
<td>30798</td>
<td>0.00</td>
<td>11268</td>
<td>0.00</td>
<td>512</td>
<td>512</td>
<td>0.00</td>
<td>bits.c:122</td>
</tr>
<tr>
<td>9294</td>
<td>0.00</td>
<td>9232</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:915</td>
</tr>
<tr>
<td>7575</td>
<td>0.00</td>
<td>7463</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>util.c:170</td>
</tr>
<tr>
<td>6237</td>
<td>0.00</td>
<td>6113</td>
<td>0.00</td>
<td>2</td>
<td>2</td>
<td>0.00</td>
<td>gzip.c:1421</td>
</tr>
<tr>
<td>4761</td>
<td>0.00</td>
<td>4699</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>util.c:283</td>
</tr>
<tr>
<td>2358</td>
<td>0.00</td>
<td>2234</td>
<td>0.00</td>
<td>2</td>
<td>2</td>
<td>0.00</td>
<td>util.c:183</td>
</tr>
<tr>
<td>9588</td>
<td>0.00</td>
<td>1153</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:1062</td>
</tr>
<tr>
<td>10218</td>
<td>0.00</td>
<td>862</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:989</td>
</tr>
<tr>
<td>8373</td>
<td>0.00</td>
<td>686</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:939</td>
</tr>
<tr>
<td>216</td>
<td>0.00</td>
<td>154</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:1703</td>
</tr>
<tr>
<td>87</td>
<td>0.00</td>
<td>25</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>bits.c:99</td>
</tr>
<tr>
<td>84</td>
<td>0.00</td>
<td>22</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>gzip.c:1398</td>
</tr>
</tbody>
</table>
## Loop Profiler Data Viewer GUI

### Function Profile View
- Column headers allow selection to control sort criteria independently for function and loop table.

### Loop Profile View
- Menu to allow user to enable filtering or displaying the source code.
Vector Report

Want to know if the compiler vectorized your code the vector report will tell you

Adding –vec-report[n] will give you output from compiler.

```
-vec-report[n]
control amount of vectorizer diagnostic information
  n=0  no diagnostic information
  n=1  indicate vectorized loops (DEFAULT when enabled)
  n=2  indicate vectorized/non-vectorized loops
  n=3  indicate vectorized/non-vectorized loops and prohibiting data dependence information
  n=4  indicate non-vectorized loops
  n=5  indicate non-vectorized loops and prohibiting data dependence information
  n=6  indicate vectorized/non-vectorized loops with greater details and prohibiting data dependence information
```
Compiler will not vectorize a loop if it can’t be certain at compile time that it is safe to do so.

• `#pragma simd` is a way to assert to the compiler that everything in the loop is safe to vectorize. (DANGEROUS)
Vector Report

Let’s run the vector report on the offload code and see if we can make it run faster.

Let’s change line 118 to #pragma simd
You will see that the compiler vectorized the loop now

Code the Future
Memory Alignment

– Static memory
  o Allocated by compiler/linker
  o Add `__attribute__((aligned(n)))` in front of variable declaration
  o Applies to global/local static variables as well as stack/auto variables
– Dynamic memory
  o Allocated by language runtime
  o Use `__mm_aligned_malloc(size, alignment_bytes)`
  o Example: `buf = (char*) __mm_malloc(bufsize, 4096);`
  o Pair it with `__mm_aligned_free()`
Other hints from the compiler

• **-guide**
  
  This flag combined with –parallel will give the user feedback from the compiler which can also help you parallelize your code.
Linux: Documentation, Samples, and Tutorials

HTML- and PDF-based documentation:

```
<install dir>/composerxe/Documentation/en_US/
  Release_Notes[F | C].pdf
  documentation_[f | c].htm
```

Samples:

```
<install dir>/Samples/en_US/[C++ | Fortran]/sample.htm
```

Vectorization, openmp, PGO, IPO, GAP, Coarray Fortran, Cilk Plus

Tutorials: 
```
<install dir>/composerxe/Documentation/en_US/
  /tutorials_[c | f]/index.htm
```

Tutorials are based on the Samples included
Steps user through new technologies

Code the Future
Code the Future
OpenMP 4.0* support

- TEAMS pragmas, directives and clauses
- DISTRIBUTE pragmas, directive and clauses
- SIMD pragmas, directives, and clauses
- TARGET pragmas, directives and clauses for attached coprocessors (or devices)
- #pragma omp taskgroup construct
- Atomic clause seq_cst
- Six new forms of atomic capture and update:
  - Atomic swap: \{v = x; x = expr;\}
  - Atomic update: \( x = \text{expr} \text{binop} x; \)
  - Atomic capture 1: \( v = x = \text{expr} \text{binop} x; \)
  - Atomic capture 2: \( v = \text{expr} \text{binop} x; \)
  - Atomic capture 3: \( \{x = \text{expr} \text{binop} x; v = x\} \)
  - Atomic capture 4: \( \{v = x = \text{expr} \text{binop} x;\} \)
- proc_bind(<type>) clause where <type> is “spread”, “close”, or “master”
- OMP_PLACES environment variable
- OMP_PROC_BIND environment variable
- omp_get_proc_bind() API