OpenMP* 4.0 for Intel® C++ Compiler (Linux*)

Lab

Vectorization

Student Workbook

Developer Product Division
Software and Service Group

Disclaimer

The information contained in this document is provided for informational purposes only and represents the current view of Intel Corporation ("Intel") and its contributors ("Contributors") on, as of the date of publication. Intel and the Contributors make no commitment to update the information contained in this document, and Intel reserves the right to make changes at any time, without notice.

DISCLAIMER. THIS DOCUMENT, IS PROVIDED "AS IS." NEITHER INTEL, NOR THE CONTRIBUTORS MAKE ANY REPRESENTATIONS OF ANY KIND WITH RESPECT TO PRODUCTS REFERENCED HEREIN, WHETHER SUCH PRODUCTS ARE THOSE OF INTEL, THE CONTRIBUTORS, OR THIRD PARTIES. INTEL, AND ITS CONTRIBUTORS EXPRESSLY DISCLAIM ANY AND ALL WARRANTIES, IMPLIED OR EXPRESS, INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, NON-INFRINGEMENT, AND ANY WARRANTY ARISING OUT OF THE INFORMATION CONTAINED HEREIN, INCLUDING WITHOUT LIMITATION, ANY PRODUCTS, SPECIFICATIONS, OR OTHER MATERIALS REFERENCED HEREIN. INTEL, AND ITS CONTRIBUTORS DO NOT WARRANT THAT THIS DOCUMENT IS FREE FROM ERRORS, OR THAT ANY PRODUCTS OR OTHER TECHNOLOGY DEVELOPED IN CONFORMANCE WITH THIS DOCUMENT WILL PERFORM IN THE INTENDED MANNER, OR WILL BE FREE FROM INFRINGEMENT OF THIRD PARTY PROPRIETARY RIGHTS, AND INTEL, AND ITS CONTRIBUTORS DISCLAIM ALL LIABILITY THEREFOR. INTEL, AND ITS CONTRIBUTORS DO NOT WARRANT THAT ANY PRODUCT REFERENCED HEREIN OR ANY PRODUCT OR TECHNOLOGY DEVELOPED IN RELIANCE UPON THIS DOCUMENT, IN WHOLE OR IN PART, WILL BE SUFFICIENT, ACCURATE, RELIABLE, COMPLETE, FREE FROM DEFECTS OR SAFE FOR ITS INTENDED PURPOSE, AND HEREBY DISCLAIM ALL LIABILITIES THEREFOR. ANY PERSON MAKING, USING OR SELLING SUCH PRODUCT OR TECHNOLOGY DOES SO AT HIS OR HER OWN RISK.

Licenses may be required. Intel, its contributors and others may have patents or pending patent applications, trademarks, copyrights or other intellectual proprietary rights covering subject matter contained or described in this document. No license, express, implied, by estoppels or otherwise, to any intellectual property rights of Intel or any other party is granted herein. It is your responsibility to seek licenses for such intellectual property rights from Intel and others where appropriate. Limited License Grant. Intel hereby grants you a limited copyright license to copy this document for your use and internal distribution only. You may not distribute this document externally, in whole or in part, to any other person or entity. LIMITED LIABILITY. IN NO EVENT SHALL INTEL, OR ITS CONTRIBUTORS HAVE ANY LIABILITY TO YOU OR TO ANY OTHER THIRD PARTY, FOR ANY LOST PROFITS, LOST DATA, LOSS OF USE OR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OF THIS DOCUMENT OR RELIANCE UPON THE INFORMATION CONTAINED HEREIN, UNDER ANY CAUSE OF ACTION OR THEORY OF LIABILITY, AND IRRESPECTIVE OF WHETHER INTEL, OR ANY CONTRIBUTOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING THE FAILURE OF THE ESSENTIAL PURPOSE OF ANY LIMITED REMEDY.

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

See: http://software.intel.com/en-us/articles/optimization-notice/

C/C++

Activity 1 – Cancellation

To demonstrate the new OpenMP* 4.0 cancellation points, we use a simple application. This application initializes an array "val" with "0". In a parallel section it finally sets the values of the array to "1". However, assume that within the parallel section there is an interims result "255" written to it. This should never show up in the final result – even if the parallel section is cancelled!

1. Take a look at the source code of the application cancel.c. Compile and execute it:

```
$ icc –openmp cancel.c -o cancel
$ ./cancel
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
```

2. Introduce the cancellation point where it is safe to interrupt the parallel section. To actually trigger the cancellation, issue a cancel at the iteration "i == 50".

Note:
In order to enable cancelation points, set **`$OMP_CANCELLATION=1`**.

Solution: `solutions/cancel.c`

Activity 2 – Taskgroup

The **`taskgroup`** directive allows to wait for all tasks created underneath the current scope. Let's use a simple application that creates a binary tree. In the end we calculate the nodes of that tree in a recursively traversing way, spawning a separate task for every node.

1. Take a look at the source code of the application taskgroup.c. Compile and execute it:

```
$ icc –openmp taskgroup.c -o taskgroup
$ ./taskgroup
40
Total count: 1
```

   Apparently we did not wait for all tasks to complete. OpenMP* 3.x had a construct **`taskwait`** already. However, this does not work here, since tasks are recursively created and the **`taskwait`** construct only waits for tasks directly created by the current scope (and not its child tasks).

2. Confirm that **`taskwait`** is not sufficient here.
3. Use **`taskgroup`** instead and confirm that the result is correct.

   Solution: `solutions/taskgroup.c`

Activity 3 – Task Dependencies

The ability to create multiple tasks and define their dependences during runtime is another new feature of OpenMP* 4.0. Again, we're using a simple example to demonstrate this. The example creates the sum of all elements of array "val" where each is decremented before. The decrement and the summing could be considered as two different tasks (just for this simple example – not meaningful for a real world application!). Since both will then operate on the same element "val[i]", their dependency needs to be defined.

1. Take a look at the source code of the application **`taskdep.c`**. Compile and execute it:

```
$ icc -openmp taskdep.c -o taskdep
$ ./taskdep
40
6
```

2. In the original example no tasks have been created. Create the tasks now in a way to have "val[i]--;" and the block with "res += val[i];" executed as separate tasks.
3. When just creating the tasks, the result is incorrect, because the dependencies of "val[i]" are not defined. Define them now and verify the result.

Solution: solutions/taskdep.c

Activity 4 – User Defined Reductions

Note:
This new feature requires Intel® C++ Compiler 16.0!

Prior to OpenMP* 4.0, reductions were only possible for pre-defined operations (add, min, max, …). The current version of OpenMP* does allow users to create their own reductions.

Let's take a small example with an array of structures (2D points). To find the minimum and maximum coordinates of all the elements in the array a parallel for loop is used.

1. Take a look at the source code of the application **`udr.c`**. Compile and execute it:

```
$ icc -openmp udr.c -o udr
$ ./udr
min: {-3,-3}
max: {1,9}
```

2. The finding of the min. and max. values was done in a parallel for loop. The results are not always correct since the tests and storing of min. and max. values inside the parallel for loop are reductions. The reductions, however, have not been used. Define them now, rerun the application and verify the correctness.

   Solution: **`solutions/udr.c`**