PRACE Summer School, CINECA 8-11 July 2013

Vectorization and Code Optimization

Hans Pabst, July 2013

Software and Services Group
Intel Corporation

Activity 1: InterProcedural Optimization

1. Generate a vectorization report for `pi_kernel`.
   - Add the `-vec-reportn` option.
   - What is the message?
2. Repeat the previous step
   - Add the `-ipo` options.
   - What is the message?
3. At which level of the code is the vectorization applied? Is it at source code or at code level?

Example Background

This example uses a reduction pattern that helps to implement an algorithm in a lock-free manner. Thread-local storage introduces multiple buffers rather than locking a single common buffer for concurrent accesses.

The command line program instantiates multiple buffers (usually according to the number of sockets or cores), and summarizes all buffers into a final result buffer which is usually identical to one of the thread-local buffers.

Activity 2: Aliasing

1. Compile and compare (“time1”):

```
$  scripts/make.sh; ./reduce
$  scripts/make.sh –fstrict-aliasing; ./reduce
```

2. Implement one of the following fixes
   - pragma ivdep
   - pragma simd
   - restrict (RESTRICT)

   Note: `scripts/make.sh; ./reduce` should now report the better time as if strict aliasing had been applied.

Activity 3: Unrolling

1. Use pragma unroll and specify an unroll factor. What strategy would you suggest:
   1. Use only power-of-two unroll factors
   2. Use 1, 2, 4, 8, etc. (until no additional benefit is found)
   3. Use the smallest unroll factor that still gives a measurable benefit.

2. Tune the unroll factor accordingly and compare the before/after timing ("time1")

```
$  scripts/make.sh; ./reduce
```

Activity 4: Prefetching

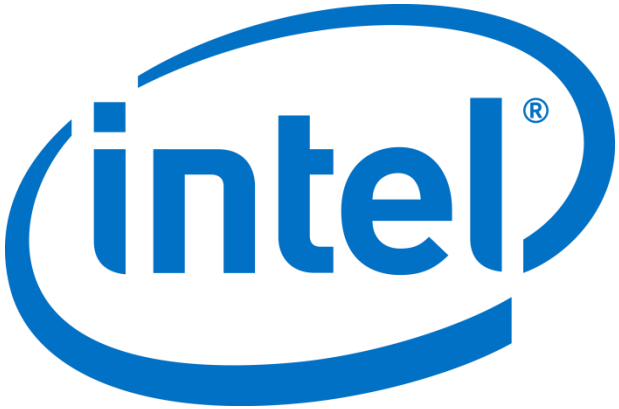
1. What is the correct preprocessor symbol to target an optimization only to the coprocessor?
   1. `__INTEL_OFFLOAD`
   2. `__MIC__`

   Write down what each of the macros achieve.

2. Tune the program and try the following:
   - Use `-opt-prefetch=`*n* option
   - Use pragma prefetch

Bonus Questions

- Have a look at the offload specification in `reduce.cpp` (just a single pragma is capable of offloading an entire call chain). Do you think that the second `pragma offload` located within a loop is able to overcome the limitation of only transferring contiguous data?
- Which of the following statements is true: (1) alignment of a buffer on the host is propagated into an offloaded code section (unless an align modifier is used within the pragma offload clause, (2) aligned memory may not only speeding up execution but also accelerating data transfers.


intel

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.

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

Backup

Bonus: Non-temporal Streaming Stores

1. Compile the program (fixed aliasing) and write the execution time down ("time1")

   ```
   $  scripts/make.sh; ./reduce
   ```

2. Use non-temporal streaming stores. Mark the variable where the stores apply ("result")
   - pragma nontemporal(*variable*)

   Note: the directive is usually applied to a loop and lists the variable(s) that should skip the cache hierarchy.

3. Compile the program and compare the execution time with the time written down in step 1.