ECP SOLLVE – OpenMP Users Monthly Teleconferences, April 28th, 2023

Solving Linear Systems with OpenMP Target Offloading and oneMKL

Presenter: Shiquan Su, PhD, Intel Senior Technical Consulting Advisor

intel®

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation. Learn more at intel.com or from the OEM or retailer.

Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

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

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. See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Outline:

- Code Snippet
- How OpenMP Dispatch a MKL Call
- Data Movement
- Kernel Debug Info
- Performance

- The content is from the following paper published in the latest issue of “The Parallel Universe Magazine” by Henry A. Gabb and Nawal Copty
- https://www.intel.com/content/www/us/en/developer/articles/technical/solve-linear-systems-onemkl-openmp-target-offload.html
- Full code example available on GitHub.

Code snippet of solving a linear system aX=b

```
!$ include "mkl_omp_offload.f90"
program solve_batched_linear_systems
!$  use onemkl_lapack_omp_offload_ilp64   ! 64-bit
…
real (kind=8), allocatable :: a(:,:), b(:,:), a_orig(:,:), b_orig(:,:)
(skip other data preparation part)
…
 !$omp target data map(to:a) map(tofrom: b) map(from:info_rf, info_rs) map(alloc:ipiv(1:stride_ipiv, 1:batch_size))
        !$omp dispatch
          call dgetrf_batch_strided(n, n, a, lda, stride_a, ipiv, stride_ipiv, batch_size, info_rf)
        !$omp dispatch
           call dgetrs_batch_strided('N', n, nrhs, a, lda, stride_a, ipiv, stride_ipiv, b, ldb, stride_b, batch_size, info_rs)
!$omp end target data
…
end program solve_batched_linear_systems
```

How to dispatch external function call with OpenMP

!$omp dispatch [clause[[,] clause] ...] new-line

call target-call ([arguments]); !! or: expression = target-call ([arguments]);

where clause is one of the following:

- device(scalar-integer-expression)
- depend([depend-modifier,] dependence-type : locator-list)
- nowait
- novariants(scalar-logical-expression)
- nocontext(scalar-logical-expression)
- is_device_ptr(list)

How to move data between host and target

`target data` Construct Syntax

- Create scoped data environment and transfer data from the host to the device and back

- Syntax (C/C++)

```
#pragma omp target data [clause[[,] clause],…]
structured-block
```

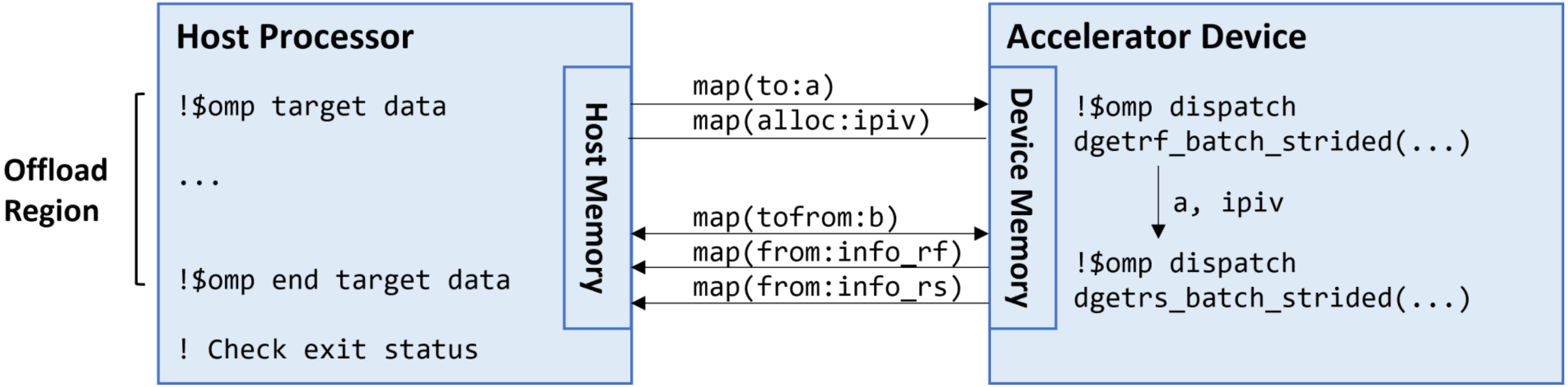
- Syntax (Fortran)

```
!$omp target data [clause[[,] clause],…]
structured-block
!$omp end target data
```

- Clauses

```
device(scalar-integer-expression)
map([{alloc | to | from | tofrom | release | delete}:] list)
if(scalar-expr)
```

Data Movement between Host and Target Device



- The large matrix a (4 Gb) only moves from host to target once.
- Matrix a resides on device memory within the Offload region, all the dispatch calls on target device “inherits” the data in the OMP target data region.
- 2 MKL dispatch calls compute on it.

```
$ ifx -i8 -DMKL_ILP64 -qopenmp -fopenmp-targets=spir64 -fsycl -free lu_solve.F90 -o lu_solve -L${MKLROOT}/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -ldl
```

```
$ OMP_TARGET_OFFLOAD=MANDATORY ZE_AFFINITY_MASK=0.0 LIBOMPTARGET_DEBUG=1 ./lu_solve 8000 8 1 1 >& lu_solve.out
```

```
$ grep Moving lu_solve.out
```

```
Libomptarget --> Moving 88 bytes (hst:0x00007ffe443ba9c8) -> (tgt:0x000000001e55008)
Libomptarget --> Moving 64 bytes (hst:0x00007ffe443bad68) -> (tgt:0x000000001e55088)
Libomptarget --> Moving 64 bytes (hst:0x00007ffe443bad18) -> (tgt:0x000000001e55108)
Libomptarget --> Moving 512000 bytes (hst:0x00007f39ada86240) -> (tgt:0x00007f399a06d000)
Libomptarget --> Moving 88 bytes (hst:0x00007ffe443bae38) -> (tgt:0x000000001e55188)
Libomptarget --> Moving 4096000000 bytes (hst:0x00007f3913fff200) -> (tgt:0x00007f3819edd000)
Libomptarget --> Moving 88 bytes (hst:0x00007ffe443bae98) -> (tgt:0x000000001e55208)
Libomptarget --> Moving 512000 bytes (tgt:0x00007f399a06d000) -> (hst:0x00007f39ada86240)
Libomptarget --> Moving 64 bytes (tgt:0x000000001e67040) -> (hst:0x00007f39add5fd80)
Libomptarget --> Moving 64 bytes (tgt:0x000000001e67000) -> (hst:0x00007f39add5fdc0)
```

- real (kind=8) , allocatable :: a(:,:) , !! 64 b per element
- 64 x 8000 x 8000 = 4,096 M, size of a, transfer from host to target memory during setup omp target data region.

Other useful info from LIBOMPTARGET_DEBUG=1

- Get GPU work distribute info: number of teams (grid size in CUDA), team size (block size in CUDA), SIMD width (warp in CUDA), loop bound, device number, kernel identity

```
Libomptarget --> Launching target execution __omp_offloading_3e_487d8356_addvec__l17 with pointer 0x00000000035e7900 (index=0).
Target LEVEL0 RTL --> Executing a kernel 0x00000000035e7900...
Target LEVEL0 RTL --> Assumed kernel SIMD width is 32
Target LEVEL0 RTL --> Preferred team size is multiple of 64
Target LEVEL0 RTL --> Loop 0: lower bound = 0, upper bound = 99999, Stride = 1
Target LEVEL0 RTL --> Team sizes = {64, 1, 1}
Target LEVEL0 RTL --> Number of teams = {1563, 1, 1}
Target LEVEL0 RTL --> Kernel Pointer argument 0 (value: 0xff002aaaaa400000) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Pointer argument 1 (value: 0xff002aaaaa480000) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Pointer argument 2 (value: 0xff002aaaaa500000) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Scalar argument 3 (value: 0x00000000000186a0) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Scalar argument 4 (value: 0x00000000000186a0) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Scalar argument 5 (value: 0x00000000000186a0) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Scalar argument 6 (value: 0x00000000000186a0) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Scalar argument 7 (value: 0x000000000001869f) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Scalar argument 8 (value: 0x0000000000000000) was set successfully for device 0.
Target LEVEL0 RTL --> Kernel Scalar argument 9 (value: 0x00000000000186a0) was set successfully for device 0.
Target LEVEL0 RTL --> Setting indirect access flags 0x0000000000000002
Target LEVEL0 RTL --> Created a command list 0x0000000004451840 (Ordinal: 0) for device 0.
Target LEVEL0 RTL --> Submitted kernel 0x0000000004399dd0 to device 0
Target LEVEL0 RTL --> Executed kernel entry 0x00000000035e7900 on device 0
```

Performance on Intel GPU vs CPU

Matrix Size	CPU Time (in second)	GPU Time (in second)
1000x1000	0.13	1.70
4000x4000	3.90	2.10
1,6000x1,6000	139.45	15.71

Timing the solution of 3 batched linear systems of varying matrix sizes on a Linux* (Ubuntu* 20.04 x64, 5.15.47 kernel) system with two 2.0 GHz 4th Gen Intel® Xeon® Platinum 8480+ processors (CPU), an Intel® Data Center GPU Max 1550 (GPU), and 528 GB memory.

All times are in seconds. GPU tests used only one tile. Each linear system had one RHS. Each experiment was run five times. The first run was discarded because it includes the just-in-time compilation overhead for the oneMKL functions. The reported time is the sum of the remaining four runs.

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation. Learn more at intel.com or from the OEM or retailer.

Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

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

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. See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.