MODERNIZE YOUR DEVELOPMENT APPROACH FOR HETEROGENEOUS COMPUTE WITH STANDARDS BASED PROGRAMMING METHODS ON INTEL® XEON PHI™ PROCESSOR

Jan Zielinski
Intel Corporation
November 2016
Intel technologies features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

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.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps. 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.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit [http://www.intel.com/performance].

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

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

Intel, the Intel logo, Intel. Experience What's Inside, the Intel. Experience What's Inside logo, Intel Xeon Phi, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

*Other names and brands may be claimed as the property of others.

© 2016 Intel Corporation. All rights reserved.
Agenda

- Enriching the standards: Offload over Fabric (OOF)
- OOF software architecture, installation and configuration
- Code modernization with Intel® Xeon Phi™ processor and Offload over Fabric
- Resource management in heterogeneous clusters
SOFTWARE MODERNIZATION

The path to fully leverage potential of the modern computing hardware
Modernize

To make (something) modern and more suited to present styles or needs

Modernization of Computing Software

When modernizing software to fully leverage potential of computing hardware, five stages of code modernization should be repeated in an iterative process:

- Leverage optimization tools and libraries
- Scalar, serial optimization
- Vectorization
- Thread Parallelization
- Scale your application from multicore to manycore

The cost of modernization can be shared between the application developers and hardware/middleware vendors by using programming standards.
OFFLOAD OVER FABRIC

Resource management in heterogeneous environments
Heterogeneous Computing Environments

Computing nodes within single group of machines (cluster) have more and more specific capabilities – increasing heterogeneity

- Number of cores (degree of optimal parallelism)
- Different kinds of memory (fast, large, persistent)
- FPGA accelerators

Heterogeneous applications easily divided into blocks using different node types take most benefits from this model

- Using widely adopted standards make heterogeneous programming easier and helps saving investments
Cluster Application Topology

Hardware defined topology

Fixed assignment of accelerators to computing nodes

Software defined topology

Heterogeneous computing nodes can be joined for particular application
Resource Management for Heterogeneous Computing

Effective resource management in heterogeneous environment requires:

- Attributes-based node selection
  - Instruction Set Architecture (ISA), available memory kinds, memory size and mode, cluster mode

- Support for software-defined job topology
  - Execute binaries on correct nodes, configuration of application topology

Call to action: enable heterogeneity features in computing infrastructure
OpenMP* Code Structuring and Data Movement

```c
double *host_data = (double *)malloc(SIZE)
double result;

#pragma omp target map(tofrom: host_data) map(from: result)
{
    transform_data(host_data);
    result = compute(host_data);
}
compute_even_more(host_data, result);
free(host_data);
```
Offload over Fabric: Extending the Offload Model

Significant investments have been made into creating applications using OpenMP* target directives or offloading using offload directives.

OpenMP* standard allow for explicit partitioning of the code to be executed on different targets.

Network environment enables high flexibility in system administration and programming.

New approach: allow target directives to be executed over the fast network.

- Offload over Fabric
OFFLOAD OVER FABRIC

Basic concept and plumbing
Offload over Fabric
Extending the Offload Model to the Network
Offload over Fabric: Software Stack

OpenFabrics Interfaces (OFI) provides standardized interface to access networking hardware using plugins based architecture.

- User application using OpenMP* target directives
- User application using offload directives
- Compiler runtime
- Intel® Coprocessor Offload Infrastructure (Intel® COI)
- OpenFabrics Interface (OFI)
- Fabric Interconnection
Code Example & Compilation
OpenMP* Code Structuring

```c
int main()
{
    printf("Running on host: ");
    what_cpu();
    #pragma omp target
    {
        printf("Running on target: ");
        what_cpu();
    }
    return 0;
}
```

Compile for offloading: `icc -qoffload-arch=mic-avx512`

Use Intel® Parallel Studio XE version 2017 or newer for compilation
Application Topology – Option 1

Use OFFLOAD_NODES and OFFLOAD_DEVICES environmental variables

- Fast and easy, but does not scale well; for testing and small applications

**host1**

- OFFLOAD_NODES = target01, target02, target03, target04, target05, target06
- OFFLOAD_DEVICES = 0, 1

**host2**

- OFFLOAD_NODES = target1, target2, target3, target4, target5, target6
- OFFLOAD_DEVICES = 2, 3

**host3**

- OFFLOAD_NODES = target1, target2, target3, target4, target5, target6
- OFFLOAD_DEVICES = 4, 5
Application Topology – Option 2**

Use topology file: OFFLOAD_NODES_FILE=<path_to_topology_file>

- Designed for cluster environments

**- available in Intel Xeon® Phi™ Processor Software release 1.4.3
Run Simple Application

Tell offloading runtime which target to offload to:

```bash
OFFLOAD_NODES=target1
```

Run your application:

```
./my_app
```
## User Authentication

Available authentication modes:

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Description</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH</td>
<td>Use SSH keyless authentication</td>
<td>Mainly for fast prototyping and workstation environments, but can be used in clusters. No manual service startup required, but initialization times can be longer than for other cases.</td>
</tr>
<tr>
<td>Munge</td>
<td>Use MUNGE infrastructure</td>
<td>Solution for cluster environments where no SSH between the nodes is available. Faster application initialization than SSH mode, but manual service startup is required before starting the application.</td>
</tr>
<tr>
<td>No auth</td>
<td>No user authentication</td>
<td>Intended for experiments or isolated environment (or when SSH and Munge are not available). Performance is expected to be on pair with Munge mechanism (or better).</td>
</tr>
</tbody>
</table>

Configuration: set environmental variable on the host: `OFFLOAD_AUTH_MODE` to `munge` or `noauth` (ssh is the default)
User Authentication – Process Startup Time

Startup and teardown time of simple offload processes (Lower is Better) [s]

- SSH
- MUNGE
- noauth

2S Intel® Xeon® CPU E5-2699v3
Intel® Xeon Phi™ processor 7250
Intel® Omni-Path Architecture fabric

See Benchmarking and Benchmarks Details slide for more information on benchmarking and configuration
Using High Bandwidth Memory (HBW)**

Offloading runtime binds user processes on target to selected memory kind

- User can select memory kind (default is HBW) and fallback mechanism:
  
  \[ \text{OFFLOAD\_MEM\_KIND=}<\text{memory\_kind}>,<\text{fallback}> \]

- Memory kind is \textit{hbw} or \textit{ddr}; fallback is \textit{hbw}, \textit{ddr} or \textit{abort}

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbw,ddr</td>
<td>Use MCDRAM first, then fallback to DDR</td>
</tr>
<tr>
<td>hbw,abort</td>
<td>Use MCDRAM only (fail if MCDRAM is out of free space)</td>
</tr>
<tr>
<td>ddr,hbw</td>
<td>Use DDR first, fallback to MCDRAM</td>
</tr>
<tr>
<td>ddr,abort</td>
<td>Use DDR only (fail if DDR is out of free space)</td>
</tr>
</tbody>
</table>

** - available in Intel Xeon® Phi™ Processor Software release 1.4.3
Using High Bandwidth Memory (HBW)

Approach 1: put all allocations into HBW

Select HBW kind and one of two fallback mechanisms: DDR or abort
- Easy way to experiment and explore performance benefits of HBW
- No control over individual allocations

Approach 2: put selected allocations into HBW

Select DDR memory kind \((ddr,abort)\) and use \(hbwmalloc/hbwfree\) functions from memkind library and \(target_ptr\) OpenMP features
- Requires code changes but allow for fine-grained control of allocations

** - available in Intel Xeon® Phi™ Processor Software release 1.4.3
Using High Bandwidth Memory (HBW)

LAMMPS Benchmarks Speedup
(Higher is Better)

- 2S Intel® Xeon® CPU E5-2699v3
- Intel® Xeon Phi™ processor 7250
- 14 MPI ranks
- Naïve port of LAMMPS* to
  Offloading over Fabric (no optimizations)
- Intel® Omni-Path Architecture fabric

See Benchmarking and Benchmarks Details slide for more information on benchmarking and configuration
Move Selected Allocations in HBW

Code example

```c
#pragma omp target map(from: target_data) is_device_ptr(target_data)
{
    target_data = (double *)hbw_malloc(SIZE);
}

omp_target_memcpy(target_data, host_data, SIZE, 0, 0, 0, omp_get_initial_device());

#pragma omp target is_device_ptr(target_data) map(from: result)
{
    result = compute(target_data);
}

#pragma omp target is_device_ptr(target_data)
{
    hbw_free(target_data);
}
```

Allocate memory from HBW using memkind

Transfer data to the device

Use allocated data on the device

Free HBW memory
LAMMPS* 30Jul16 with naïve port to offloading to Intel® Xeon Phi™ processor (no optimizations)

Intel® Omni-Path Architecture fabric

See Benchmarking and Benchmarks Details slide for more information on benchmarking and configuration
OFFLOAD OVER FABRIC
Modernize your application for Intel® Xeon Phi™ processor
Code Modernization with Intel® Xeon Phi™ Processor and Offload over Fabric

Heterogeneous application with offload

Homogeneous application

Prototype heterogeneity with OpenMP* directives

(Re)compile for OOF & execute

Experiment with HBW usage

Optimize for many cores, AVX-512, better HBW utilization

Modernize to MPI/OpenMP* without offload model (optimal performance)

Exit at any step if performance goals are met
Summary

Save investments made for the coprocessor/offloading model with simple recompilation and execution in network environment

Jump start modernization of your application based on Intel compiler or OpenMP* offloading towards MPI model by just recompiling your code

Easily experiment in heterogeneous hardware environments to explore diversified capabilities of available network nodes
Benchmarking and Benchmarks Details

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. For more complete information visit www.intel.com/benchmarks.

Intel measured results as of November 2016. Measured using:

1. Offloading micro-benchmark looping #pragma target mic directive with 1 integer or a 4MB array of integer numbers
2. Intel® Coprocessor Offload Infrastructure coi_simple tutorial
3. LAMMPS* benchmarks from USER-INTEL package (src/USER-INTEL/TEST) with naïve port to Offload over Fabric (compilation for offloading to Intel® Xeon Phi™ x200 processor).

OFFLOAD HOST AND BASELINE CONFIGURATION: Dual Socket Intel® Xeon® processor E5-2699 v3 (45 M Cache, 2.3 GHz, 18 Cores) with Intel® Hyper-Threading and Turbo Boost Technologies enabled, 64 GB DDR4-2133 MHz memory, Red Hat Enterprise Linux* 7.2 (Maipo), Intel® Omni-Path Host Fabric Interface Adapter 100 Series 1 Port PCIe® x16, Intel® Server Board S2600WT2, 500GB SATA drive ST500DM002 and 1TB ST1000NM0033 Disks.

14 MPI ranks and 2 threads on the Intel® Xeon® processor based host used in the benchmark, Intel package, 1 offload target, automatic load balancing.

OFFLOAD TARGET CONFIGURATION: One node Intel® Xeon Phi™ processor 7250 (16 GB, 1.4 GHz, 68 Cores) in Intel® Server System LADMP2312KXXX41, 64GB DDR4, quad cluster mode, MCDRAM flat memory mode, Red Hat Enterprise Linux* 7.2 (Maipo), Intel® Omni-Path Fabric Interface, 250MB SATA WD2502ABYS-0 System Disk.

Intel® Compiler 17.0.0, Intel® MPI Library 2017, LAMMPS* code base: 30Jul16
THANK YOU FOR YOUR TIME

Jan Zielinski

jan.zielinski@intel.com

www.intel.com/hpcdevcon