MVAPICH2 on Intel® Omni-Path Architecture

Sayantant Sur
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.

Copyright © 2015, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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
The Intel® OPA 100 Series is an end-to-end Fabric solution
Scales to 10,000 nodes or more
Future Omni-Path Fabric to be deployed on Argonne Aurora that has greater than 50,000 nodes
Unique integration of CPU and Fabric
• Density ↑, Reliability ↑, Power ↓
Massively scaled up of the Host Fabric Interface (HFI) capabilities
• 100 Gbps HFI with PCIe v3.0 x16 host interface
• Host ASIC contains two separate full performance HFI instances
• Scaling and optimization of the internal HFI micro-architecture
Fabric features: Adaptive Routing, Dispersive Routing, Traffic Flow optimizations, and many others
Software on Intel® Omni-Path 100

**PSM:**
PSM version 2: [https://github.com/01org/opa-psm2](https://github.com/01org/opa-psm2)

Fully backward compliant with PSM

*New feature:* tag size increased to 96-bits from 64-bits

**Open Fabrics Interface (OFI):**

Next-generation Fabric interface being defined and developed by the Open Fabrics Interfaces Working Group under the Open Fabrics Alliance (OFA)

libfabric version 1.1 was released recently

OFI libfabric PSM provider: [https://github.com/ofiwg/libfabric](https://github.com/ofiwg/libfabric)

The PSM 2 provider will be available soon
Tag-bits usage in MVAPICH2

**PSM:**

<table>
<thead>
<tr>
<th>Communicator</th>
<th>Source</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bits</td>
<td>16-bits</td>
<td>32-bits</td>
</tr>
</tbody>
</table>

Under this scheme, the sender rank, thereby number of ranks in communicator, is limited to 64K

**PSM2:**

<table>
<thead>
<tr>
<th>Communicator</th>
<th>Source</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bits</td>
<td>32-bits</td>
<td>32-bits</td>
</tr>
</tbody>
</table>

There is more space for both communicator and source bits with adequate space up to Exascale limits
Packed structure instead of flat uint64_t bitfield

Tag passed in as a pointer instead of by value

Fully working patch to MVAPICH2 available from Intel, working with the team to integrate into release

(thanks to Henry Estela who developed the patch)
Open Fabrics Interfaces

Open Source
Leverage existing open source community
- Inclusive development effort
- App and HW developers

Application-Centric
Software interfaces aligned with application requirements
- 168 requirements from MPI, PGAS, SHMEM, DBMS, sockets, NVM, ...

Scalable
Optimized SW path to HW
- Minimize cache and memory footprint
- Reduce instruction count
- Minimize memory accesses

Implementation Agnostic
Good impedance match with multiple fabric hardware
- InfiniBand, iWarp, RoCE, raw Ethernet, UDP offload, Omni-Path, GNI, others

Next-Generation OFA Interfaces, and future direction for OpenFabrics
Open Fabrics Interface Architecture

Libfabric Enabled Applications

MPICH (Netmod) | Open MPI (MTL / BTL) | Clang UPC | GASNet | ES-API rsockets | MVAPICH??

Libfabric

Control Services
- Discovery

Communication Services
- Connection Management
- Address Vectors

Completion Services
- Event Queues
- Counters

Data Transfer Services
- Message Queues
- Tag Matching
- Atomics

Sockets
- Verbs: MLNX, iWarp

Cisco usNIC
- Intel Omni-Path

Cray GNI

MUG '15
### libfabric API Analysis: Critical path send

**Issues apply to many APIs: Verbs, AIO, DAPL, Portals, NetworkDirect, ...**

Table from Libfabric paper at HOTI 2015

<table>
<thead>
<tr>
<th>Structure</th>
<th>Field</th>
<th>Write Size</th>
<th>Branch?</th>
<th>Type</th>
<th>Parameter</th>
<th>Write Size</th>
<th>Branch?</th>
</tr>
</thead>
<tbody>
<tr>
<td>sge</td>
<td></td>
<td>16</td>
<td></td>
<td>void *</td>
<td>buf</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>send_wr</td>
<td></td>
<td>60</td>
<td></td>
<td>size_t</td>
<td>len</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>next</td>
<td></td>
<td>Yes</td>
<td></td>
<td>void *</td>
<td>desc</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>num_sge</td>
<td></td>
<td>Yes</td>
<td></td>
<td>fi_addr_t</td>
<td>dest_addr</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>opcode</td>
<td></td>
<td>Yes</td>
<td></td>
<td>void *</td>
<td>context</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>flags</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td></td>
<td><strong>76+8 = 84</strong></td>
<td><strong>4+1 = 5</strong></td>
<td></td>
<td></td>
<td><strong>40</strong></td>
<td><strong>0</strong></td>
</tr>
</tbody>
</table>

- **Generic entry points** result in additional memory reads/writes
- **Interface parameters** can force branches in the provider code
- **Move operation flags** into initialization code path for optimal SW paths

**Move operation flags into initialization code path for optimal SW paths**
libfabric API Analysis: Memory Footprint

**Per peer addressing data**

Table from Libfabric paper at HOTI 2015

<table>
<thead>
<tr>
<th>Type</th>
<th>Data</th>
<th>Size</th>
<th>Type</th>
<th>Data</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct *</td>
<td>ibv_ah</td>
<td>8</td>
<td>uint64</td>
<td>fi_addr_t</td>
<td>8</td>
</tr>
<tr>
<td>uint32</td>
<td>QPN</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uint32</td>
<td>QKey</td>
<td>4 [0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ibv_ah</td>
<td></td>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>36</strong></td>
<td></td>
<td></td>
<td><strong>8</strong></td>
</tr>
</tbody>
</table>

**Map Address Vector**:  
- encodes peer address
- direct mapping to HW command data

**Index Address Vector**:  
- minimal footprint
- requires lookup/calculation for peer address

**IB Data**:  
- DLID: 2  
- SL: 1  
- QPN: 3

**Size**:  
- DLID: 2  
- SL: 1  
- QPN: 3

**Shared Address Table**: easily shareable for all processes on the node!!
Growing OFI Ecosystem and adoption

Officially sanctioned by OFA
Developed by a broad set of stakeholders
Adoption, contributions from industry and lab partners
Positive feedback from users and implementers alike
Collaborative publications and tutorial material being developed
31 members on GitHub, five different fabric providers
Intel views libfabric as the best way to foster and support fabric innovation, including Omni-Path
Summary

MVAPICH2 over Omni-Path works out of the box (no changes)
Can be further enhanced for scalability by using 96-bit tag
Patch to MVAPICH2 available as open-source

OFI libfabric has made progress
- Two releases
- Middleware support available: MPICH, Open MPI, UPC, GASNet, Sockets

Offers fundamental performance and scaling benefits compared to libibverbs

MVAPICH2 on OFI would benefit users by enabling multiple fabrics and encouraging fabric innovation by vendors!