
12th ANNUAL WORKSHOP 2016

# NEW FEATURES AND CAPABILITIES OF PSM2

Ravi Murty
Intel

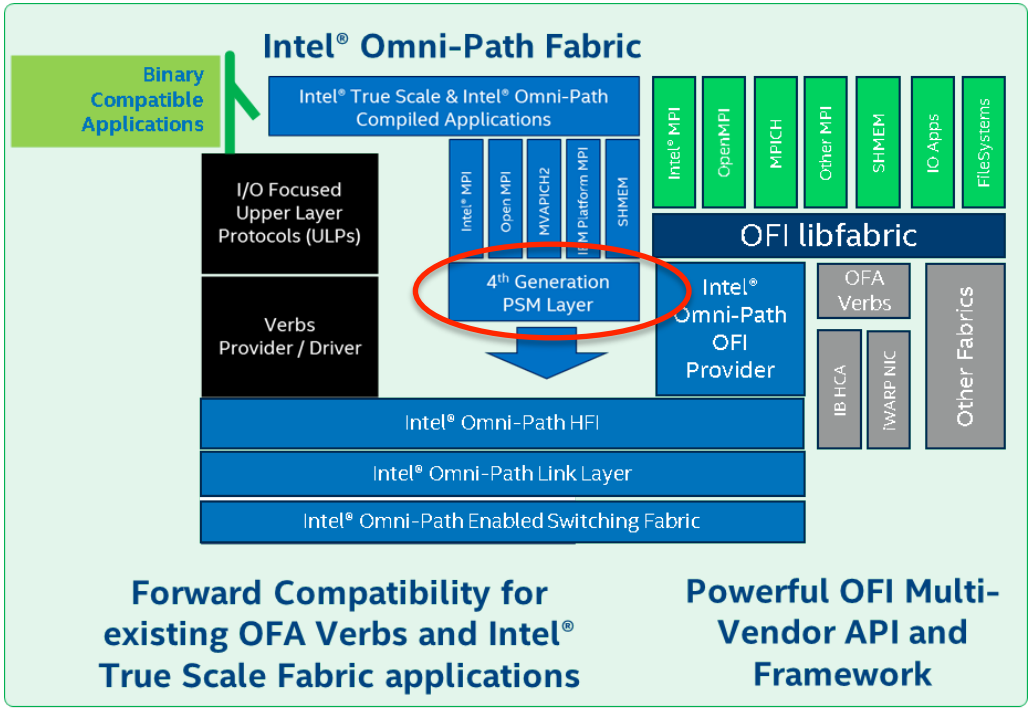
April 7th, 2016

# OUTLINE

- What is Performance Scaled Messaging (PSM)?
- New Features in PSM2
- Compatibility with PSM1
- PSM Architecture
- PSM and OFI
- PSM Open Source Strategy

# INTEL® OMNI-PATH ARCHITECTURE

## OPTIMIZED HOST IMPLEMENTATION




- **Host Strategy: Leverage OpenFabrics Alliance* (OFA)**
  - **Host Strategy: Leverage OpenFabrics Alliance* (OFA)**
  - OpenFabrics Alliance compliant: Off-the-shelf application compatibility
  - Provides an extensive set of mature upper layer protocols
  - Integrates 4th generation proven, scalable PSM capability for HPC
- OpenFabrics Interface (OFI) API aligned with application requirements
  - **Access: Open Source Key Elements**
    - Intel® Omni-Path Fast Fabric Tools, Fabric Manager, and GUI
    - Host software stack via OFA
- **Channels: Integrate into Linux* Distributions**
  - Intel® Omni-Path Architecture support included in standard distributions


# PSM – A QUICK OVERVIEW

# WHAT IS PSM?

- **A scalable messaging library for large clusters and millions of ranks**
  - PSM is carefully designed to match the semantics needed by compute middlewares such as MPI
- **End-point communication model**
  - Manage flow state for reliability with support for dynamic addition of end-points
- **Matched Queue (MQ) component**
  - Semantically matched to the needs of MPI using tag matching
  - Provides calls for communication progress guarantees
  - MQ completion semantics (standard vs. synchronized)
- **Data transfer strategies optimized for latency and bandwidth**
  - Adaptively takes advantage of on-load and off-load mechanisms in HFI, transparent to library user
- **Optimized inter and intra node communications encapsulated in the API**
- **Error handling scheme**
  - Per Endpoint and Global error handler and user-defined error handlers.

# PSM API

- **Global tag matching API with 64-bit tags**
- **Scale up to 64K processes per job**
- **MQ APIs provide point-to-point message passing between endpoints**
  - e.g. psm_mq_send, psm_mq_irecv
- **No “recvfrom” functionality – needed by some applications**

# PSM2 API

- **4th generation mature API with forward compatible extensions to existing PSM API**
- **Tag matching improvement**
  - Increased tag size to 96 bits
  - Fundamentally ((stag ^ rtag) & rtagsel) == 0
  - Supports wildcards such as MPI_ANY_SOURCE or MPI_ANY_TAG using zero bits in rtagsel
- **Allows for practically unlimited scalability**
  - Up to 64M processes per job
- **Added "recvfrom" API**
  - Allows caller to specify the *source* from which to receive message
  - e.g. difference between

**PSM2 API**
- **4th generation mature API with forward compatible extensions to existing**
- **Tag matching improvement**
  - Increased tag size to 96 bits

# PSM2 TAG MATCHING

```
#define PSM_MQ_TAG_ELEMENTS 3
typedef struct psm2_mq_tag {
    union {
        uint32_t tag[PSM_MQ_TAG_ELEMENTS] __attribute__((aligned(16)));
        struct {
            uint32_t tag0;
            uint32_t tag1;
            uint32_t tag2;
        };
    };
} psm2_mq_tag_t;
```

- **Application fills ‘tag’ array or ‘tag0/tag1/tag2’ and passes to PSM**
- **Both tag and tag *mask* use the same 96 bit tag type**

# PSM2 SEND SIDE FUNCTIONS

```
psm_error_t
psm2_mq_isend2(psm2_mq_t mq, psm2_epaddr_t dest,
        uint32_t flags,  psm2_mq_tag_t *stag,
        const void *buf, uint32_t len, void *context,
        psm2_mq_req_t *req);

psm_error_t
psm2_mq_send2(psm2_mq_t mq, psm2_epaddr_t dest,
        uint32_t flags, psm2_mq_tag_t *stag,
        const void *buf, uint32_t len);
```

- **Non-blocking and blocking send PSM API extended to support 96-bit tag**

# PSM2 RECEIVE SIDE STATUS

```
typedef struct psm2_mq_status2 {
    psm2_epaddr_t  msg_peer;
    psm2_mq_tag_t  msg_tag;
    uint32_t  msg_length;
    uint32_t  nbytes;
    psm_error_t  error_code;
    void   *context;
} psm2_mq_status2_t;
```

- **psm2_mq_status2_t is derived from psm2_mq_status_t**
  - msg_tag field changed from 64-bits to the new 96-bit tag
- **Additionally, a new field msg_peer added**
  - Provide the source of the message
  - Eliminates application burden to maintain relationship between message and source of message

# PSM2 RECEIVE SIDE FUNCTIONS

```
psm_error_t
psm2_mq_irecv2(psm2_mq_t  mq, psm2_epaddr_t  src, psm2_mq_tag_t  *rtag,
      psm2_mq_tag_t  *rtagsel, uint32_t  flags, void  *buf, uint32_t len,
      void *context, psm2_mq_req_t  *req);

psm2_error_iprobe2
psm_mq_iprobe2(psm2_mq_t  mq, psm2_epaddr_t src, psm2_mq_tag_t *rtag,
      psm2_mq_tag_t *rtagsel, psm2_mq_status2_t *status);

psm2_error_t
psm_mq_improbe2(psm2_mq_t mq, psm2_epaddr_t src, psm2_mq_tag_t *rtag,
      psm2_mq_tag_t *rtagsel, psm_mq_req_t *req, psm2_mq_status2_t *status);
```

- **Functions psm2_mq_irecv2 and psm2_mq_iprobe2**
  - Tag and tag mask are 96-bit wide. Also src argument allows receiver to specify the source from which to receive the message
- **psm2_mq_improbe2 is a new function for MPI3 support**

# PSM2 PROGRESSION API

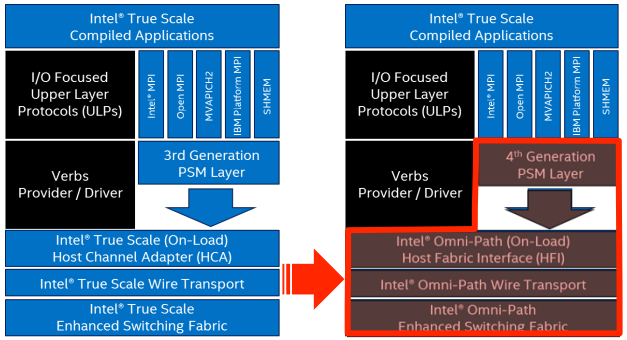
```
psm_error_t psm2_mq_ipeek2(psm2_mq_t mq, psm_mq_req_t *req,
psm2_mq_status2_t *status);

psm_error_t psm2_mq_wait2(psm2_mq_req_t *request, psm2_mq_status2_t
*status);

psm_error_t psm2_mq_test2(psm2_mq_req_t *request, psm2_mq_status2_t
*status);
```

- **Change in status argument from previous PSM API**


# PSM1/PSM2 FORWARD COMPATIBILITY AND CO-EXISTENCE

# PSM1/PSM2 FORWARD COMPATIBILITY

- **Both PSM and PSM2 are supported on Intel® OPA**
  - On Intel® OPA PSM1 64-bit tag is zero-extended to 96-bits
  - A PSM function only returns the first 64 bits of tag
  - A PSM2 function returns all 96 bits of tag
  - For a PSM2 receive function, if ‘src’ argument is NULL, PSM uses only the 96b *tag* to match a message from any source
  - status2 will always return the message source epaddr ‘msg_src’, independent of the ‘src’ argument in receiving function




No software application changes

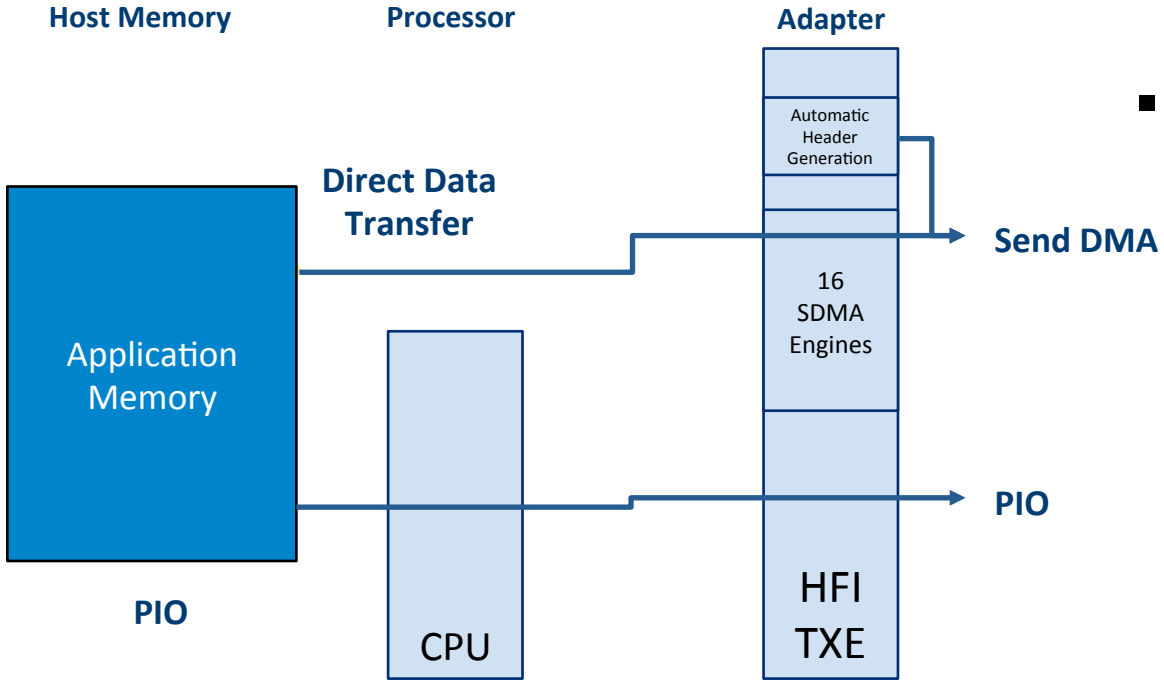
# PSM1/PSM2 CO-EXISTENCE

- **Co-existence between Intel® True Scale and Intel® Omni-Path Architecture**
  - PSM2 renamed the symbol names but …
- **Forward compatibility is provided by psm2-compat RPM:**
  - Provides binary compatibility for PSM1 based
  - Set LD_LIBRARY_PATH=/usr/lib64/psm2-compat/ prior to application launch.
  - /usr/lib64/psm2-compat/libpsm_infinipath.so.1, is a very thin wrapper exporting PSM APIs, but uses PSM2 library. Minimal performance impact from native PSM2


# PSM2 OPTIMIZED ON INTEL® OMNI-PATH ARCHITECTURE

# INTEL® OPA: SEND OPTIONS

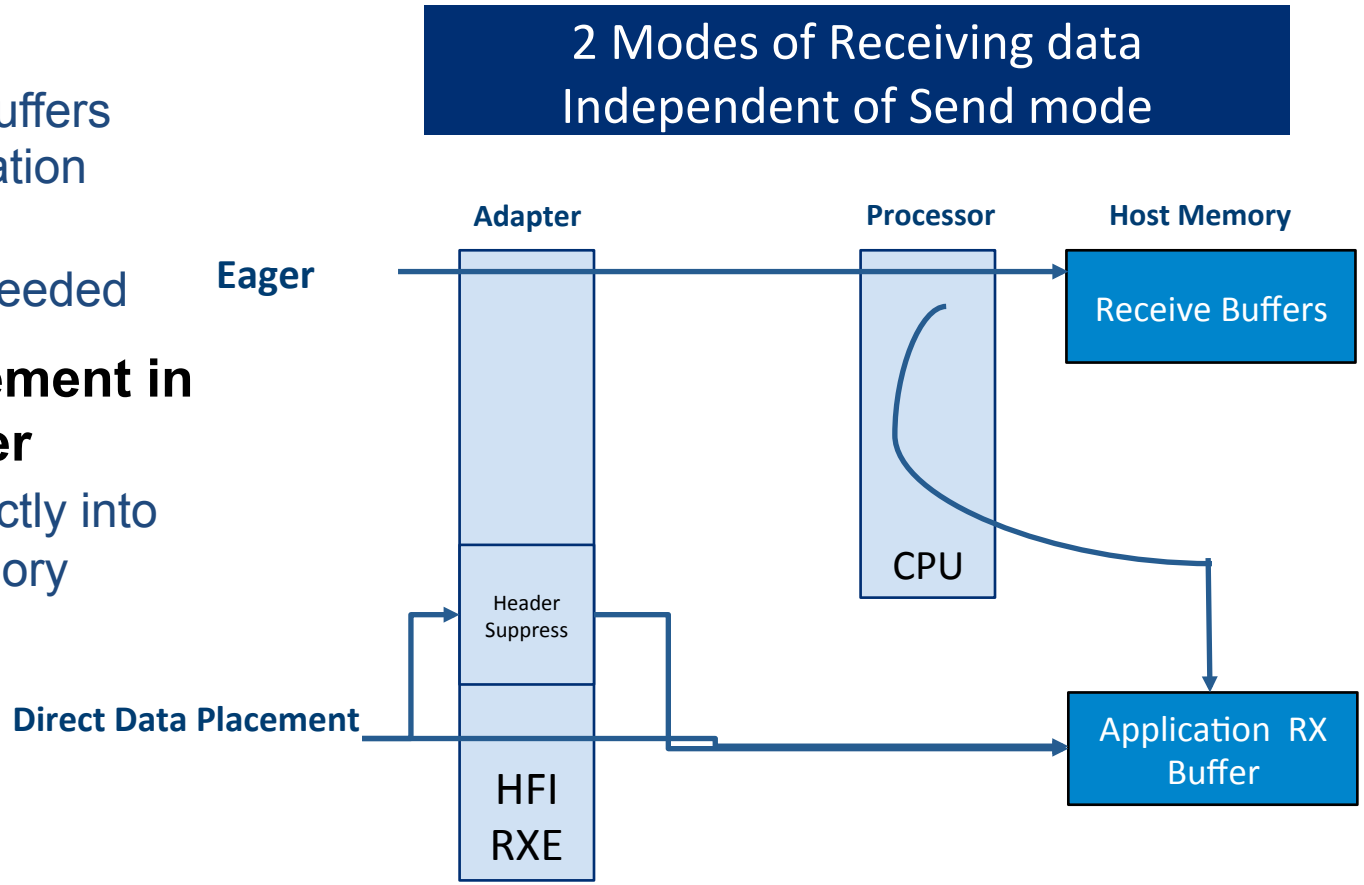
2 Modes of Sending data Independent of Receive mode




- **Send DMA (SDMA)**
  - Optimizes Bandwidth for Large messages
  - 16 SDMA Engines for CPU Offload
- **Programmed I/O (PIO)**
  - Optimizes Latency and Message Rate for small messages

# INTEL® OPA: RECEIVE OPTIONS

- **Eager-Receive**
  - Received data buffers copied to Application Buffer
  - No handshake needed
- **Direct Data placement in Application Buffer**
  - Data placed directly into Application Memory



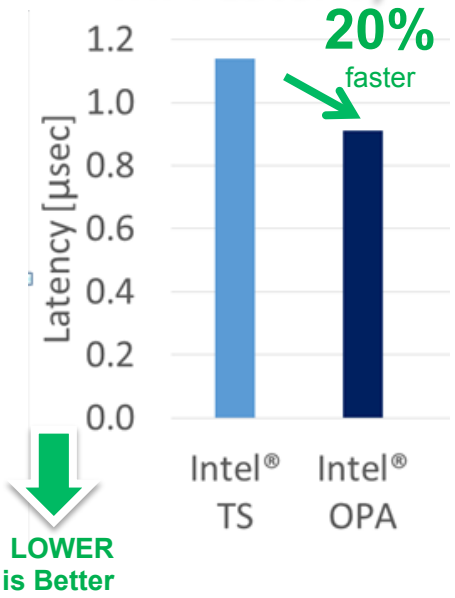
# INTEL® OPA PROVIDES EFFICIENT DATA TRANSFER MECHANISMS

| Message Size | Send Side | Receive Side |
|---|---|---|
| Up to 8KB | PIO Send | Eager Receive |
| > 8KB and < 64KB | SDMA | Eager Receive |
| 64KB or more | SDMA | Expected Receive |

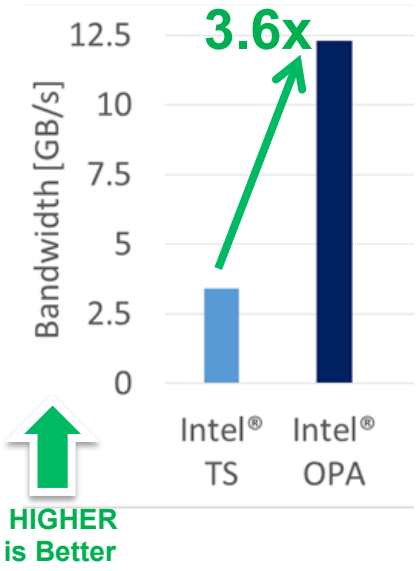
- **Most efficient data movement method automatically chosen based on message size**

# MPI PERFORMANCE ON INTEL® OPA

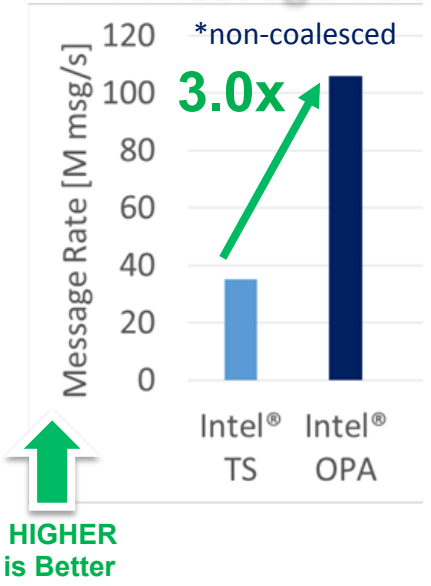
## MPI Latency[1]




## MPI Bandwidth[2]




## MPI Message Rate[3]




Tests performed on Intel® Xeon® Processor E5-2697v3 dual-socket servers with 2133 MHz DDR4 memory. Turbo mode enabled and hyper-threading disabled. Ohio State Micro Benchmarks v. 4.4.1. Intel OPA: Open MPI 1.10.0 with PSM2. Intel Corporation Device 24f0 – Series 100 HFI ASIC. OPA Switch: Series 100 Edge Switch – 48 port. IOU Non-posted Prefetch disabled in BIOS. Intel® True Scale: Open MPI. QLG-QLE-7342(A), 288 port True Scale switch. 1. osu_latency 8 B message. 2. osu_bw 1 MB message. 3. osu_mbw_mr, 8 B message (uni-directional), 28 MPI rank pairs


# PSM AND OFI

# PSM PROVIDERS FOR OFI

- **In addition to a *standalone* library, OFI supports providers for both versions of PSM**
- **OFI provider for PSM and PSM2: https://github.com/ofiwg/libfabric**
  - The psm provider runs over PSM1.x interface supported by Intel® True Scale
  - The psm2 provider runs over PSM2.x interface supported by Intel® OPA
  - The psm2 provider supports a richer set of capabilities due to the additional functionality provided by the PSM2 library described earlier.
- **The PSM2 provider allows creation of multiple OFI endpoints without limitations**
  - PSM provider has limitations on endpoint capability settings when multiple endpoints are opened.
- **The fi_psm and fi_psm2 man pages provide additional information including capabilities supported by these providers**
  - **https://github.com/ofiwg/libfabric/wiki/Provider-Feature-Matrix-v1.3.0**
- **Status: Both providers are in stable state. On-going work will be mainly bug fixes/code refactoring**

# PSM IS OPEN SOURCE

- **PSM1 has been part of OFED for a very long time.**
  - PSM2 is still in the works
- **PSM2 sources: https://github.com/01org/opa-psm2**
- **Distros: PSM2 is accepted to go into RHEL 7.2 and SLES SP2**
- **Documentation: Additional details on PSM2 and its API are available: http://www.intel.com/content/dam/support/us/en/documents/network/omni-adptr/sb/Intel_PSM2_PG_H76473_v1_0.pdf**

# CONCLUSIONS

- PSM2 is an optimized user-space library designed to meet the requirements of MPI
- MQ component of PSM2 supports tag matching with 96-bit tags and “recvfrom” functionality, among other improvements
- Existing applications written to the PSM1 API should run on Intel® OPA via the compat library
- PSM support two send mechanism and two receive mechanisms that allow for very efficient messaging from a latency and BW perspective.
- OFI has providers for PSM1 and PSM2
- Finally, PSM2 is open source

# LEGAL DISCLAIMERS

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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

Any forecasts of goods and services needed for Intel's operations are provided for discussion purposes only. Intel will have no liability to make any purchase in connection with forecasts published in this document.

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

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.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families: Go to: Learn About Intel® Processor Numbers

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Intel, Intel Xeon, Intel Xeon Phi™ are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

# OPTIMIZATION NOTICE

**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 SSE3 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

OPENFABRICS
ALLIANCE

12th ANNUAL WORKSHOP 2016

# THANK YOU

Ravi Murty

Intel