Software

Communication Overlap using Open Fabric Interfaces

Sayantan Sur, Intel

SEA Symposium on Overlapping Computation and Communication

April 4th, 2018

Legal Disclaimer & Optimization Notice

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

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.

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.

Copyright © 2018, 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

Presentation Outline

- Introduction to OpenFabrics Interface (OFI)
- Progress modes exposed by OFI
- OFI enables hardware to progress many communication constructs
- Different hardware under OFI has different levels of offload
- *How can we develop portable applications that deliver excellent overlap on a variety of OFI providers?*

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Open Fabric Interfaces

User-centric interfaces lead to innovation and adoption

Open Source

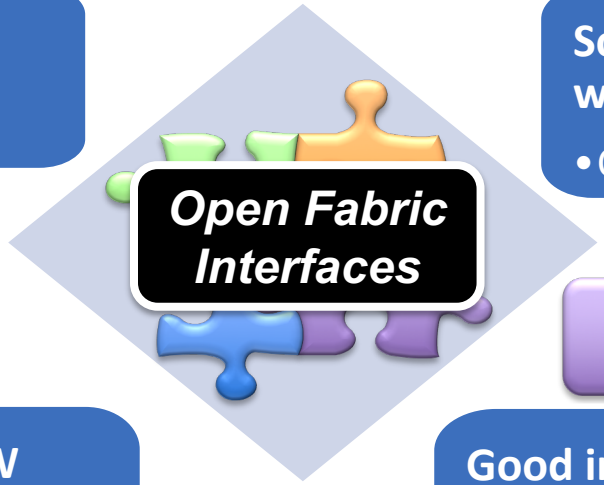
Inclusive development

- App and HW developers

User-Centric

Software interfaces aligned with user requirements

- Careful requirement analysis



Scalable

Optimized SW path to HW

- Minimize cache and memory footprint, memory accesses
- Reduce instruction count

Implementation Agnostic

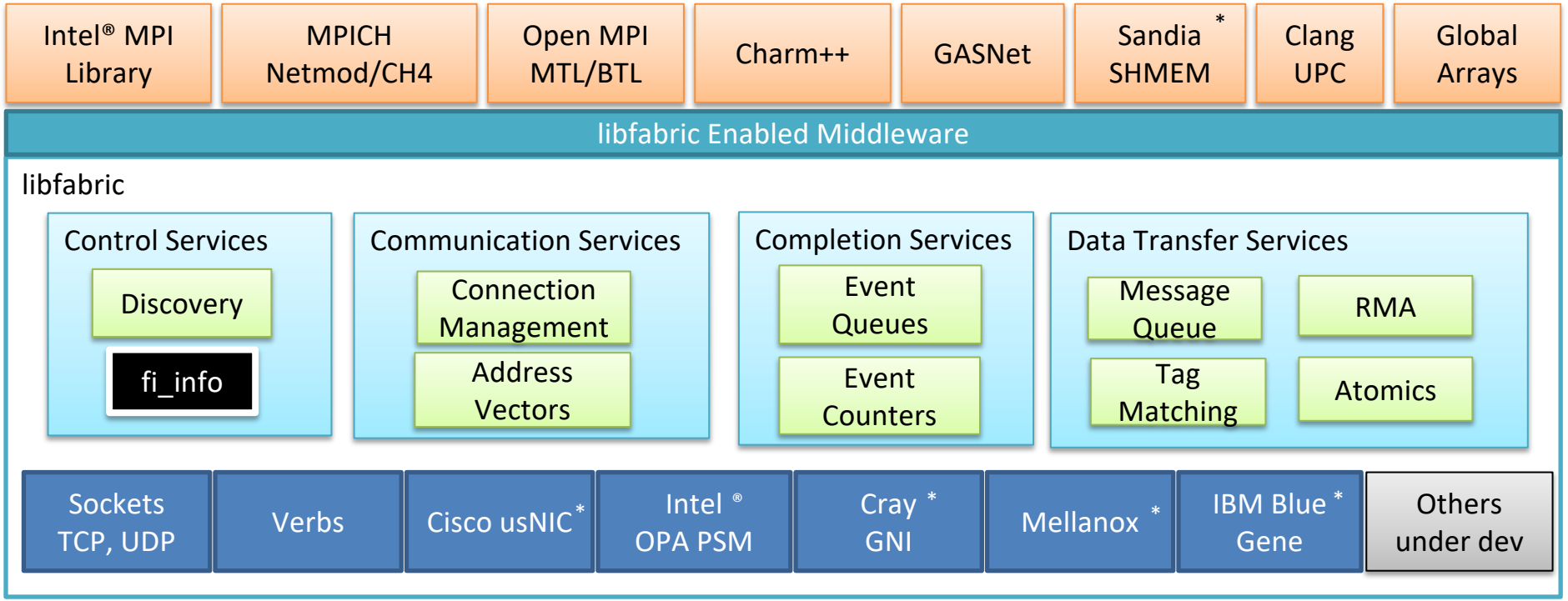
Good impedance match with multiple fabric hardware

- InfiniBand, iWarp, RoCE, raw Ethernet, UDP offload, Omni-Path, GNI, BGQ, …

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

OFI HPC Community

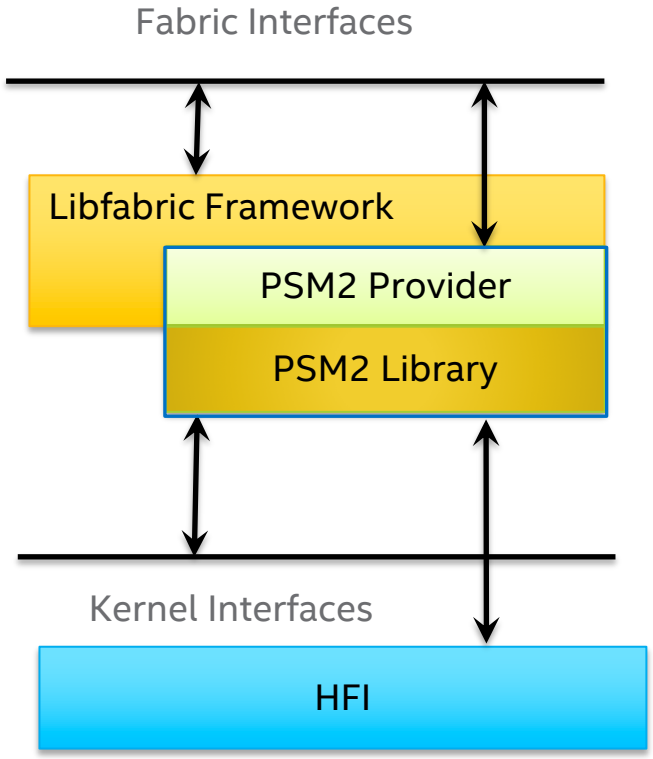
Sampling of HPC Middleware already targeting OFI



Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

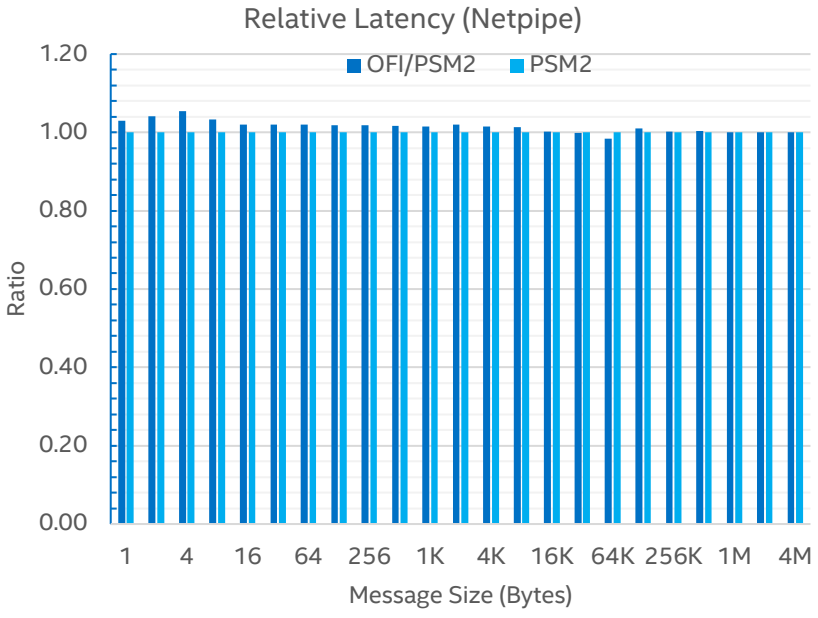
Libfabric on Intel® OPA

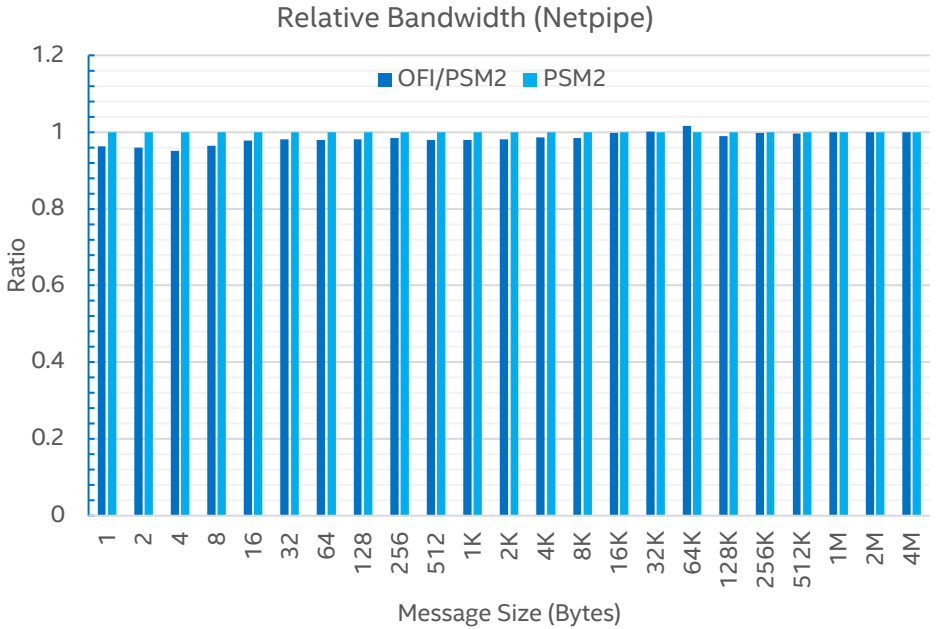
- Libfabric PSM2 provider uses the public PSM2 API
- Semantics matches with MPI
- PSM2 library uses PIO/Eager for small message and DMA/Expected flow for high bandwidth for large messages
- Current focus is full OFI functionality
- Provider for libfabric-1.6.0 requires PSM2 library version 10.2.235 (IFS 10.5) or later
- Performance optimizations are possible to reduce call overheads, and provide better semantic mapping of OFI directly to HFI



Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® OPA/PSM Provider Performance





Performance is comparable for most messages (optimizations are ongoing)

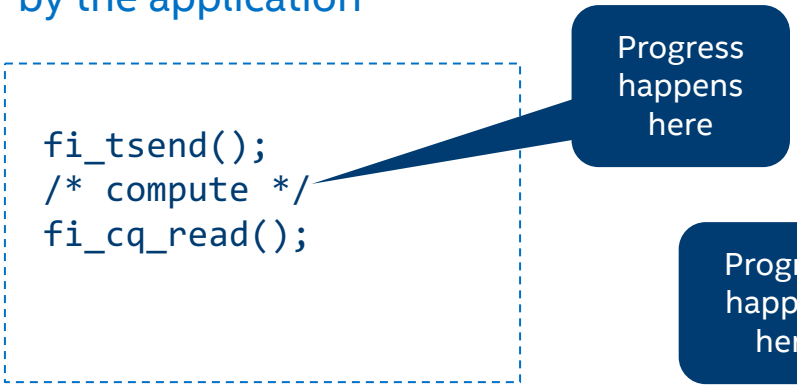
Intel Xeon E5-2697 (Ivy Bridge) 2.6GHz; Intel Omni-Path; RHEL 7.3; libfabric 1.6.0; IFS 10.8.0.0.98

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Progress Modes Exposed by OFI

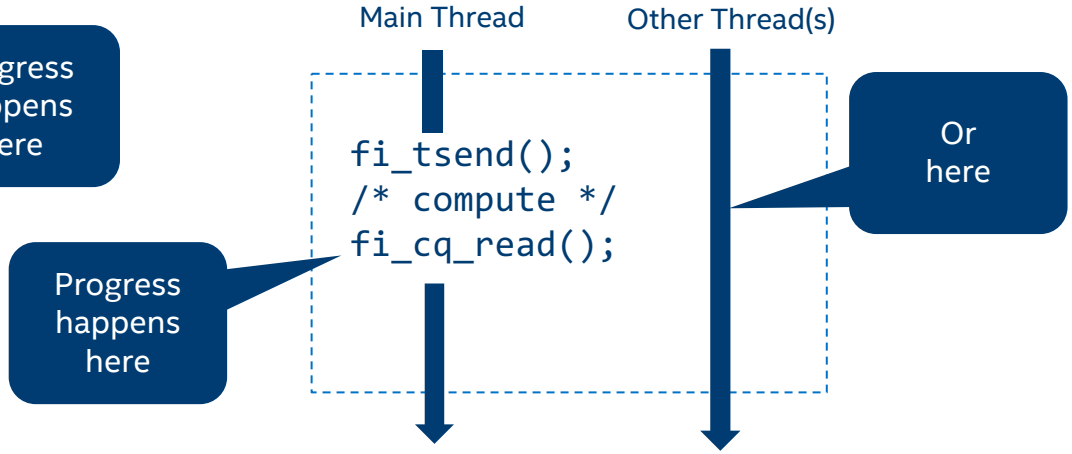
Automatic Progress

- Indicates that the provider will make forward progress on an asynchronous operation without further intervention by the application



Manual Progress

- Indicates that the provider requires the use of an application thread to complete an asynchronous request



Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Automatic Progress in OFI

Hardware Offload ✓

- Depends on provider implementation

Software Driven

- Can use provider internal threads
- Internal threads must be scheduled, affinitized carefully such as to avoid performance penalties
- Generally OK for apps to rely on provider threads for progress
- *Can we do better?*

PSM Provider Implementation

- One progress thread / rank
- Invoked if not called in a while
- Experimental: when urgent packets arrive (such as RTS, CTS)

Scheduling of progress threads

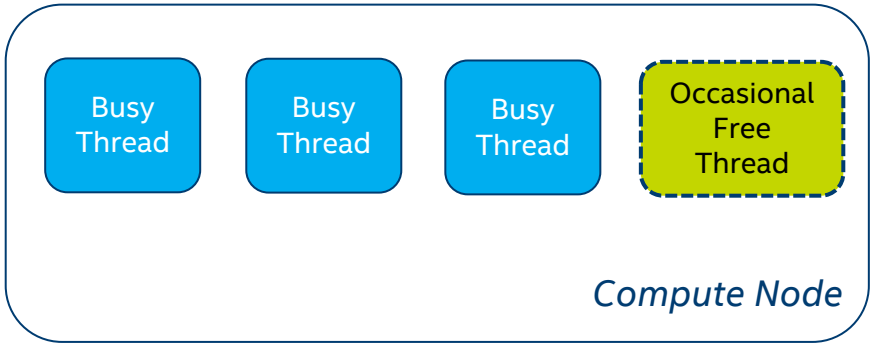
- Do not contend with main thread
- Avoid context switch overheads by using Hyper-Threads

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

When should we assist software Auto Progress?

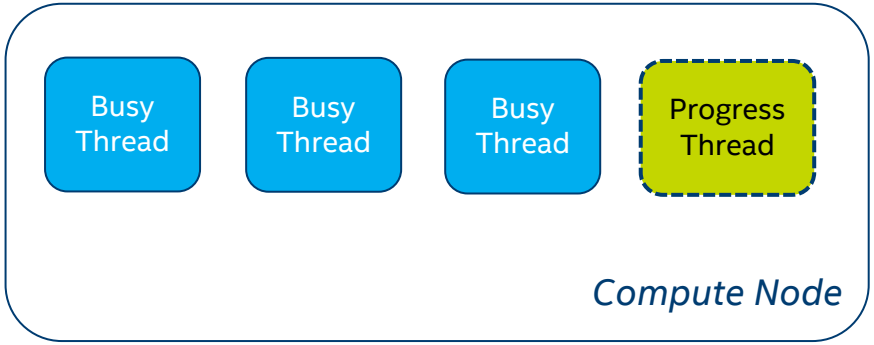
Opportunistically

- Not all compute resources are used by the application due to various factors like load imbalance



Deliberately – when progress is critical to performance

- Heavy reliance on asynchronous progress semantics, such as non-blocking collectives

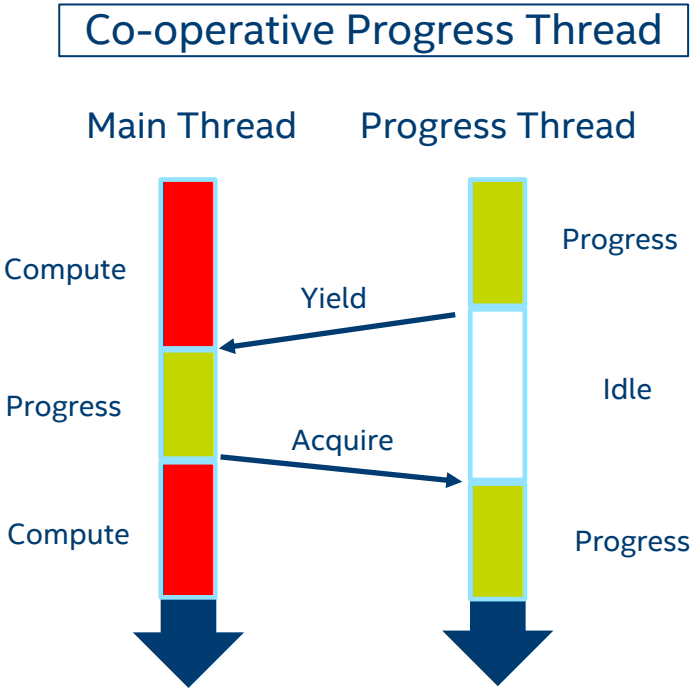


Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

How can we assist Software Auto Progress?

Basic Idea:

- Make threads available to library that uses OFI, such as MPI, SHMEM, …
- Programming model knows best how to drive the OFI library
- *Opportunistic*: main thread drives progress
- *Deliberate*: co-operate with main thread to drive progress



Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

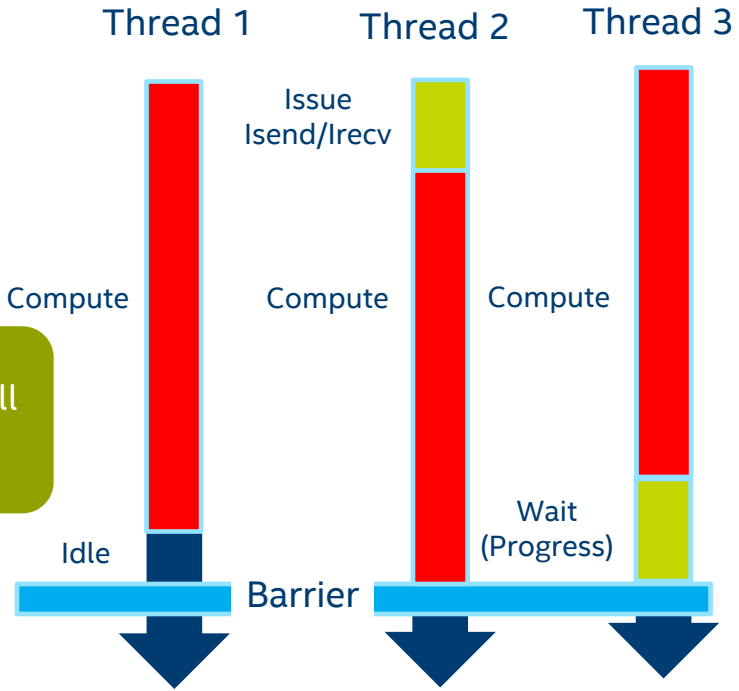
Opportunistic - an MPI/OpenMP* example

Sample from stencil code

OpenMP* task semantic allows first available thread to do work

```
#pragma omp parallel
{
     #pragma omp single nowait
     {
          MPI_Irecv();
          MPI_Isend();
     #pragma omp task
          MPI_Waitall();
     }
     /* imbalanced compute */
}
```

Typical OpenMP implementation will defer task until barrier



Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Deliberate Progress – Helper Threads

```
#pragma omp parallel
{
  // Do some things …

  if (omp_get_thread_num() > 0)
    MPI_Recv(…, MPI_COMM_SELF);
  else {
    MPI_Allreduce(boatload_of_data,
                  …, MPI_COMM_WORLD);
    for (int i = 0; i < omp_get_num_threads()-1; i++)
      MPI_Send(…, MPI_COMM_SELF);
  }
}
```

MPI Library can interpret a receive from self as a "progress thread"

Donate any extra OpenMP threads to the MPI library

Application can claim the threads back when it needs them

MPI Forum could augment the standard to codify this behavior, for example via info keys

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Overlapping in domains beyond traditional HPC

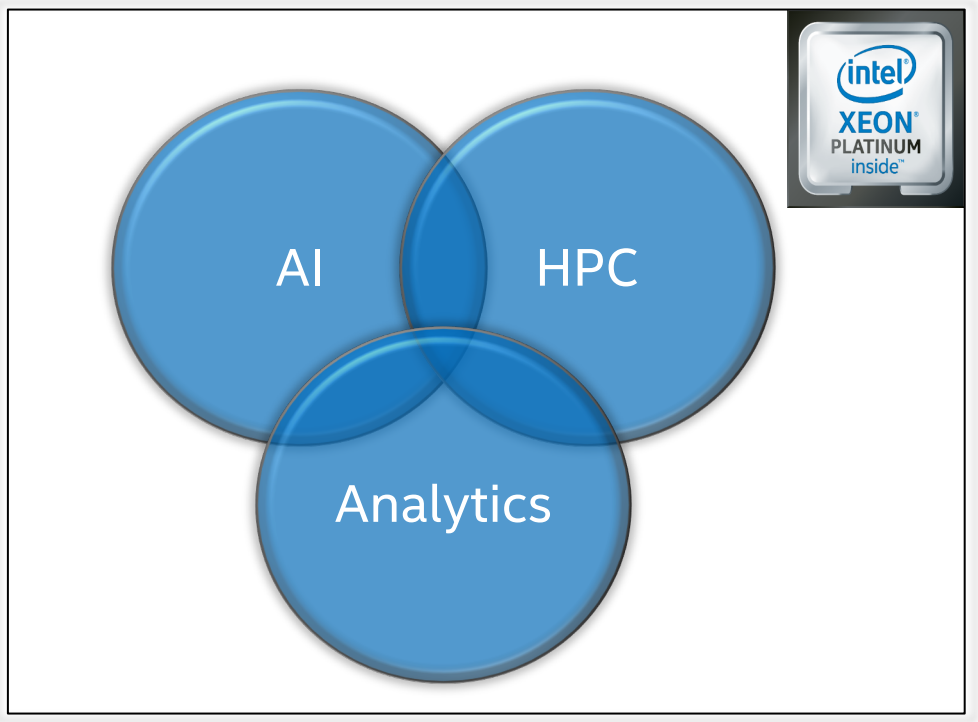
Rapid rise of ML/DL workloads

Emerging fused ML/DL and HPC workloads

Deep Learning is adapting to use HPC like techniques

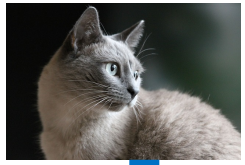
Recently, TensorFlow has been modified to use Ring like Allreduce algorithms

Are there overlap sensitive communication patterns?



Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Image Processing Operation



Forward Propagation (computation)

Layer 0
Layer 1
Layer N-1
Layer N

Communicate Revised Weights

Layer 0
Layer 1
Layer N-1
Layer N

Back Propagation

output

expected

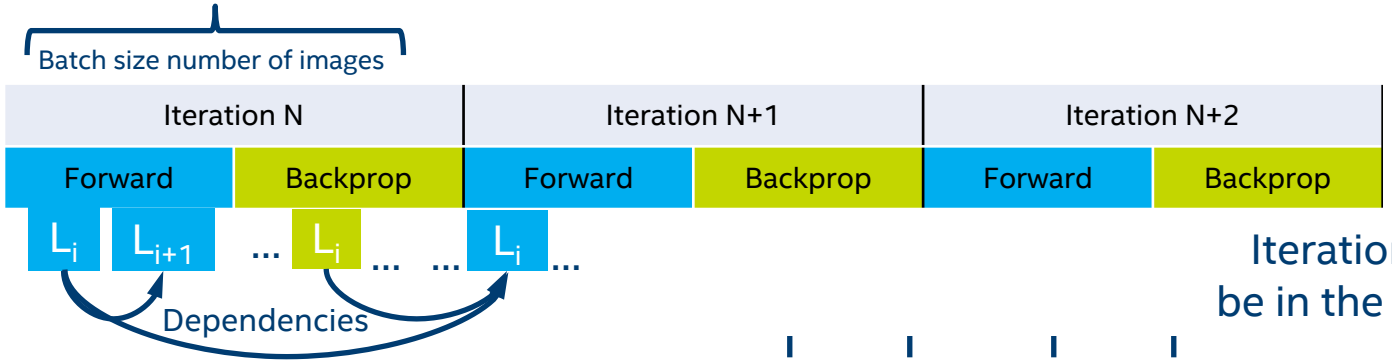
penalty (error or cost)

0.10	0.15	0.20	···	0.05
person	cat	dog		bike

0	1	0	···	0
person	cat	dog		bike

Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Data Parallel Image Recognition Training



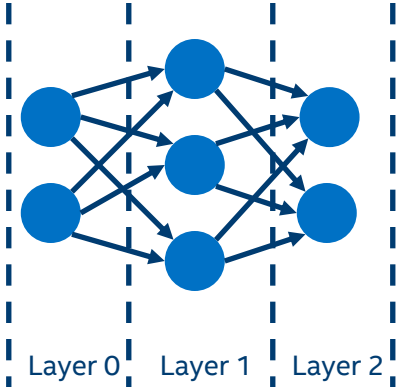
Iterations can be in the millions

Data parallel mode, the model is replicated on each node

Model is further split into individual layer of neurons

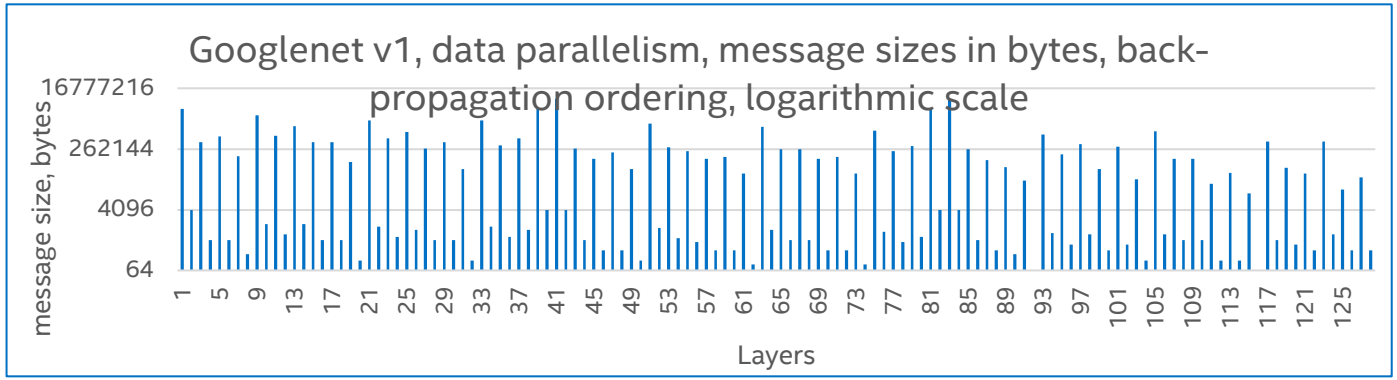
Computation for each layer is fixed

Communication for each layer is fixed



Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Message Sizes for Sample Topology



There can be many layers in the topology

Message sizes vary per layer

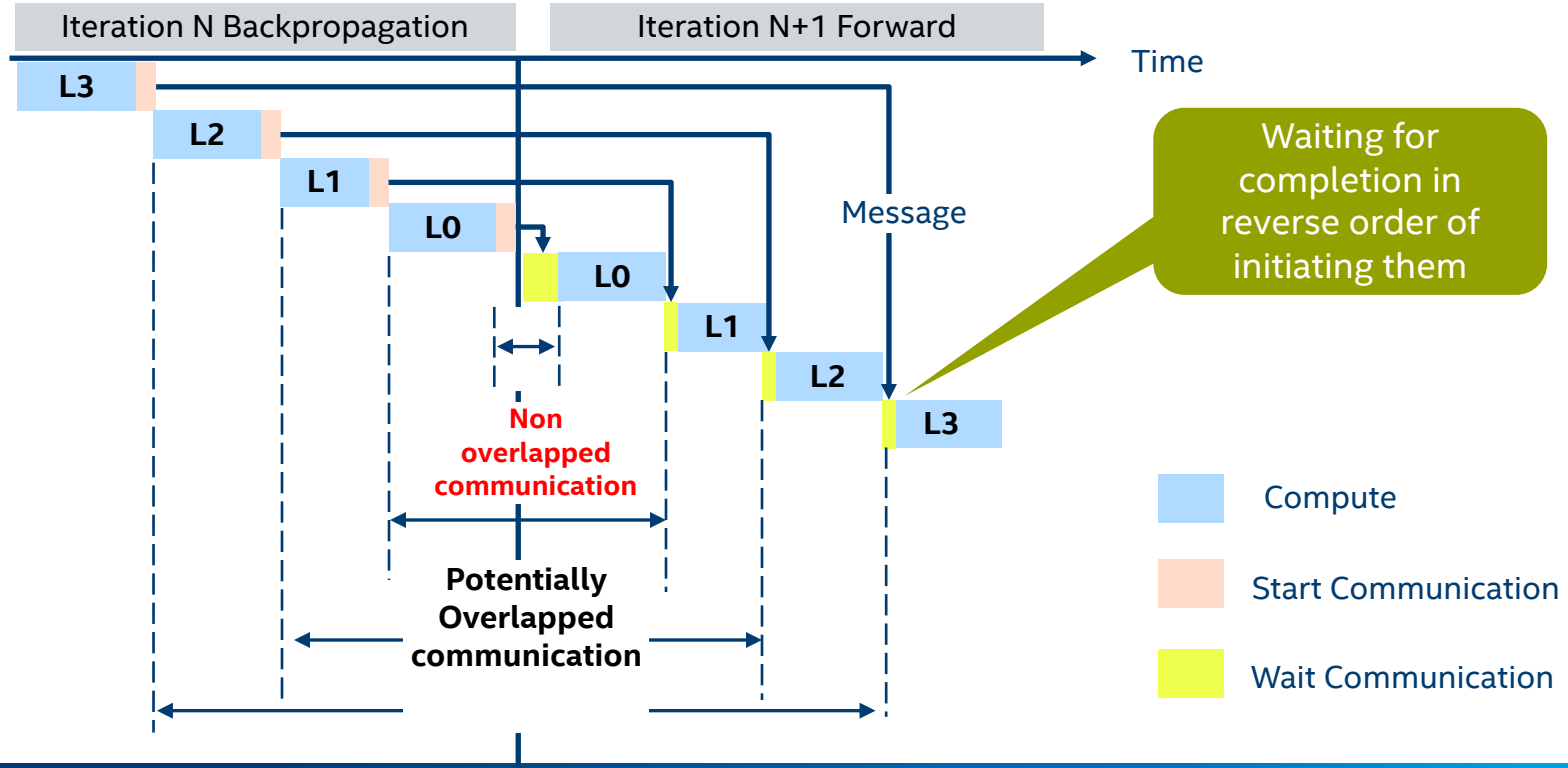
Multiple Allreduce algorithms are required for optimal performance

$$Num\ Iterations = \frac{Num\ Images\ in\ Dataset\ * Epochs}{Batch\_size}$$

- Example: Imagenet 1K
  - 1.2M images, Batch Size 64
  - 19K iterations / epoch, 64 epochs
  - Total ~1.24M iterations

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Performance does depend on Overlap

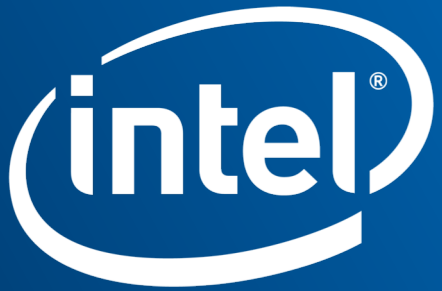


Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Summary and Future Work

- OFI exposes various progress modes
- MPI / middleware can choose optimizations based on provider capabilities
- Software progress can be aided with application help
- Opportunistic progress can be achieved using OpenMP and other task models
- Deliberate progress model can be aided with minimal MPI modifications
- Emerging workloads like AI might rely on overlap beyond those in traditional HPC use models
- Advances in MPI Standards will enable overlap performance on a wide variety of OFI providers

Optimization Notice
Copyright © 2018, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Software