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.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
Agenda

- Distributed Performance with Intel® MPI Library
- Tuning MPI Application Performance with Intel® Trace Analyzer and Collector
- Summary and Resources
Intel® MPI Library
Intel® MPI Library Overview

- Optimized MPI application performance
  - Optimized collectives with topology and architecture awareness

- Lower-latency and multi-vendor interoperability
  - Industry leading latency
  - Performance optimized support for the fabric capabilities through OpenFabrics* (OFI) / libfabric

- Sustainable scalability up to 340K cores
  - Efficient path by relying on libfabric
  - New: Faster startup and finalization

- More robust MPI applications
  - Seamless interoperability with Intel® Trace Analyzer and Collector

- Extra Features
  - Conditional numerical reproducibility via I_MPI_CBWR
  - Automatic tuning via I_MPI_TUNING_MODE
Intel® MPI Library Overview

- Streamlined product setup
  - Part of Intel® oneAPI HPC Toolkit or standalone
  - Install as root or as standard user
  - Environment variable script setvars.sh or vars.sh sets paths

- Compilation scripts to handle details
  - One set to use Intel compilers, one set for user-specified compilers

- Environment variables for runtime control
  - I_MPI_* variables control many factors at runtime
    - Process pinning, collective algorithms, device protocols, and more
Compiling MPI Programs

- Compilation scripts automatically passes necessary libraries and options to underlying compiler
  - `mpiifort`, `mpiicpc`, and `mpiicc` use the Intel compiler by default
  - `mpif77`, `mpicxx`, `mpicc`, and others use GNU compiler by default
- Multiple ways to specify underlying compiler
  - `I_MPI_F77`, `I_MPI_CXX`, etc. environment variables
  - `-f77`, `-cc`, etc. command line options
  - Useful for makefiles portable between MPI implementations
- All compilers are found via PATH
MPI Launcher

- Robust launch command

```
mpirun <mpi args> executable <program args>
```

- Options available for:
  - Rank distribution and pinning
  - Fabric selection and control
  - Environment propagation
  - And more
Process Placement

• Layout Across Nodes
  - Default placement puts one rank per core on each node
  - Use –ppn to control processes per node
  - Use a machinefile to define ranks on each node individually
  - Use arguments sets or configuration files for precise control for complex jobs

• Pinning on Node
  - Can pin to single or multiple cores
  - Multiple options for automatic distribution based on resources such as socket, shared cache level, NUMA arrangement
  - See documentation for details:
GPU Pinning

- `I_MPI_OFFLOAD=1` enables GPU features
- `I_MPI_OFFLOAD_CELL` defines unit of division for offload
  - tile – Single tile/subdevice
  - device – Single device (GPU)
- `I_MPI_OFFLOAD_DOMAIN_SIZE` sets the number of cells per rank
- `I_MPI_OFFLOAD_DEVICES` can limit which device numbers to use
Fabric Control via libfabric

- `I_MPI_OFI_PROVIDER` chooses provider (select based on interconnect hardware):
  - Default is normally fine
  - `tcp` – Ethernet
  - `psm2` – Intel® Omni-Path Architecture
  - `mlx` – InfiniBand* (requires at least Intel® MPI Library 2019 Update 5 and UCX 1.4)
Conditional Numerical Reproducibility

- **I_MPI_CBWR**
  - 0 (default) – no reproducibility controls, utilize all optimizations
  - 1 (weak) – disable topology aware optimizations, reproducible across different rank placements/topologies
  - 2 (strict) – disables topology aware optimizations and hardware optimizations, reproducible across hardware and topology

- **MPI_Comm_dup_with_info**
  - “I_MPI_CBWR”=“yes”, sets strict mode for communicator
Automatic Tuning via Autotuner

- Tuning happens behind the scenes during application run
- Tuning is per communicator
- To tune:
  - I_MPI_TUNING_MODE=auto
  - I_MPI_TUNING_BIN_DUMP=<tuning file> (optional)
- To use tuning results:
  - I_MPI_TUNING_BIN=<tuning file>
Debugging MPI Applications

- **GDB**
  - mpirun <mpi options> -gdb <application and options>
  - mpirun –n <nranks> -gdba <mpirun pid>

  - Set via –gtool option, -gtoolfile option, or I_MPI_GTOOL
  - “<prepend>:<rank set>[=launch mode][@arch]"
Intel® Trace Analyzer and Collector
Event-based Tracing for Distributed Applications
Intel® Trace Analyzer and Collector Overview

- Intel® Trace Analyzer and Collector helps the developer:
  - Visualize and understand parallel application behavior
  - Evaluate profiling statistics and load balancing
  - Identify communication hotspots

- Features
  - Event-based approach
  - Low overhead
  - Excellent scalability
  - Powerful aggregation and filtering functions
  - Performance Assistance and Imbalance Tuning
Strengths of Event-based Tracing

**Predict**
- Detailed MPI program behavior

**Record**
- Exact sequence of program states – keep timing consistent

**Collect**
- Collect information about exchange of messages: at what times and in which order

An event-based approach is able to detect temporal dependencies!
Summary page shows computation vs. communication breakdown

Is your application MPI-bound?

Resource usage

Largest MPI consumers

Next Steps
Views and Charts

- Helps navigate the trace data
- A View can show several Charts
- All Charts in a View are linked to a single:
  - time-span
  - set of threads
  - set of functions
- All Charts follow changes to View (e.g. zooming)
Get detailed impression of program structure
Display functions, messages, and collective operations for each rank/thread along time-axis
Retrieval of detailed event information
Quantitative Timeline

Get impression on parallelism and load balance
Show for every function how many threads/ranks are currently executing it
Flat Function Profile

Statistics about functions
Call Tree and Call Graph

Function statistics including calling hierarchy

- **Call Tree** shows call stack
- **Call Graph** shows calling dependencies
Communication Profiles

Statistics about point-to-point or collective communication

Matrix supports grouping by attributes in each dimension

- Sender, Receiver, Data volume per msg, Tag, Communicator, Type

Available attributes

- Count, Bytes transferred, Time, Transfer rate
MPI Performance Assistant

- Automatic Performance Assistant
- Detect common MPI performance issues
- Automated tips on potential solutions

Automatically detect performance issues and their impact on runtime
Checking MPI Application Correctness

Runtime Correctness Checks
Integration with Debuggers
MPI Correctness Checking

Solves two problems:

- Finding programming mistakes which need to be fixed by the application developer
- Detecting errors in the execution environment

Two aspects:

- Error Detection – done automatically by the tool
- Error Analysis – manually by the user based on:
  - Information provided about an error
  - Knowledge of source code, system, ...
How Correctness Checking Works

- All checks are done at runtime in MPI wrappers
- Detected problems are reported on stderr immediately in textual format
- A debugger can be used to investigate the problem at the moment when it is found
Categories of Checks

- Local checks: isolated to single process
  - Unexpected process termination
  - Buffer handling
  - Request and data type management
  - Parameter errors found by MPI

- Global checks: all processes
  - Global checks for collectives and p2p ops
    - Data type mismatches
    - Corrupted data transmission
    - Pending messages
    - Deadlocks (hard & potential)
  - Global checks for collectives – one report per operation
    - Operation, size, reduction operation, root mismatch
    - Parameter error
    - Mismatched MPI_Comm_free()
Severity of Checks

Levels of severity:

- **Warnings**: application can continue
- **Error**: application can continue but almost certainly not as intended
- **Fatal error**: application must be aborted

Some checks may find both warnings and errors

- Example: CALL_FAILED check due to invalid parameter
- Invalid parameter in MPI_Send() => msg cannot be sent => error
- Invalid parameter in MPI_Request_free() => resource leak => warning
Correctness Checking on Command Line

Command line option via –check_mpi flag for Intel MPI Library:

```
$ mpirun --check_mpi -n 2 overlap
[...]
[0] WARNING: LOCAL:MEMORY:OVERLAP: warning
[0] WARNING: New send buffer overlaps with currently active send buffer at address 0x7fbffec10.
[0] WARNING: Control over active buffer was transferred to MPI at:
[0] WARNING: MPI_Isend(*buf=0x7fbffec10, count=4, datatype=MPI_INT, dest=0, tag=103, comm=COMM_SELF [0], *request=0x508980)
[0] WARNING: overlap.c:104
[0] WARNING: Control over new buffer is about to be transferred to MPI at:
[0] WARNING: MPI_Isend(*buf=0x7fbffec10, count=4, datatype=MPI_INT, dest=0, tag=104, comm=COMM_SELF [0], *request=0x508984)
[0] WARNING: overlap.c:105
```
Correctness Checking in GUI

Enable correctness checking info to be added to the trace file:

- Enable VT_CHECK_TRACING environment variable:

  ```bash
  $ mpirun --check_mpi --genv VT_CHECK_TRACING on --n 4 ./a.out
  ```
Warning indicate potential problems that could cause unexpected behavior (e.g., incomplete message requests, overwriting a send/receive buffer, potential deadlock, etc.).

Errors indicate problems that violate the MPI standard or definitely cause behavior not intended by the programmer (e.g., incomplete collectives, API errors, corrupting a send/receive buffer, deadlock, etc.).
Debugger Integration

Debugger must be in control of application before error is found

A breakpoint must be set in MessageCheckingBreakpoint()
Trace of a Simple MPI Program

Demo
Online Resources

Intel® MPI Library product page
  • www.intel.com/go/mpi

Intel® Trace Analyzer and Collector product page
  • www.intel.com/go/traceanalyzer

Intel® oneAPI HPC Toolkit Forum
  • https://community.intel.com/t5/Intel-oneAPI-HPC-Toolkit/bd-p/oneapi-hpc-toolkit

Intel® MPI Library Tuning Files