INTEL® PARALLEL STUDIO XE 2016 WHAT'S NEW

September 2015

Intel Software and Services Group

Intel® Parallel Studio XE 2016 Suites

Vectorization – Boost Performance By Utilizing Vector Instructions / Units

- Intel® Advisor XE - ***Vectorization Advisor*** identifies new vectorization opportunities as well as improvements to existing vectorization and highlights them in your code. It makes actionable coding recommendations to boost performance and estimates the speedup.

Scalable MPI Analysis– Fast & Lightweight Analysis for 32K+ Ranks

- Intel® Trace Analyzer and Collector add ***MPI Performance Snapshot*** feature for easy to use, scalable MPI statistics collection and analysis of large MPI jobs to identify areas for improvement

Big Data Analytics – Easily Build IA Optimized Data Analytics Application

- Intel® Data Analytics Acceleration Library (Intel® DAAL) will help data scientists speed through big data challenges with optimized IA functions

Standards – Scaling Development Efforts Forward

- Supporting the evolution of industry standards of ***OpenMP*, MPI, Fortran and C++*** Intel® Compilers & performance libraries

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

DETAIL SLIDES

Intel® C/C++ and Fortran Compilers 16.0
Get best performance with latest standards

- More of C++14, generic lambdas, member initializers and aggregates
- More of C11, _Static_assert, _Generic, _Noreturn, and more
- BLOCK_LOOP Pragma/Directive Syntax
- Intel® Cilk™ Plus Combined Parallel & SIMD loops
- OpenMP 4.0 C++ User Defined Reductions, Fortran Array Reductions
- Vectorization enhancements
- OpenMP 4.1 asynchronous offloading, simdlen, simd ordered
- Offload Feature Enhancements for Intel® Xeon Phi™
- F2008 Submodules, Impure Elemental Functions
- F2015 TYPE(*), DIMENSION(..), RANK intrinsic, attributes for args with BIND

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

C++14 Standard Support

Generic Lambdas

```
auto glambda = [ ] (auto a) { return a; };
```

Generalized lambda captures

```
int x = 4;
int z = [&r = x, y = x+1]
    {
      r += 2;       // set x to 6; "R is for Renamed Ref"
      return y+2;   // return 7 to initialize z
    }();            // invoke lambda
```

Digit Separators

[[deprecated]] attribute

Function return type deduction

Member initializers and aggregates

Feature Test macros

```
#if __has_include("shared_mutex") // use standard header
#elif __has_include("boost/shared_mutex.h") // use from BOOST
#endif
…
#ifndef __cpp_constexpr           // no constexpr available
#elif __cpp_constexpr == 200704   // c++11 constexpr available
#else                             // c++14 constexpr available
#endif
```

- Test for header files existence
- Test for compiler features

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

C11 Standard Support

New Feature Support

- Unicode strings
- C11 anonymous unions

New Keyword Support

_Alignas	_Alignof
_Static_assert	_Thread_local
_Noreturn	_Generic

_Generic : template like functionality in C w/o template support in the language defines different actions based on different arguments types

```
#define  pow(X) _Generic((X), long double: powl, \
                              default: pow, \
                                float: powf)(X)
```

when pow() is called with a long double argument it calls powl()
when pow() is called with a float, it calls powf(),
when pow() is called with other types it calls pow()

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

BLOCK_LOOP Pragma/Directive Syntax

C++:

```
#pragma block_loop [clause[,clause]...]
#pragma noblock_loop
```

Fortran:

```
!DIR$ BLOCK_LOOP [clause[[,] clause]...]
!DIR$ NOBLOCK_LOOP
```

clause → factor(expr), level(levels)

- Enables or disables loop blocking for the immediately following nested loops
- Works seamlessly with other directives including SIMD directive

```
#pragma block_loop factor(256) level(1:2)
 for( j=1; j<n ; j++){
    f = 0;
    for( i=1; i<n; i++){
        f = f + a[i] * b[i];
   }
   c[j] = c[j] + f;
 }
```

```
for( jj=1 ; jj<n/256+1; jj+){
  for( ii=1; ii<n/256+1; ii++){
    for( j=(jj-1)*256+1; min(jj*256, n); j++){
      f = 0;
      for ( i=(ii-1)*256+1; i<min(ii*256,n); i++){
        f = f + a[i] * b[i];
      }
      c[j]  = c[j] + f;
    }
  }
}
```

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® Cilk™ Plus
Combined Parallel & SIMD loops

- New combined Parallel + Vector loop

```
_Cilk_for _Simd (int i = 0; i < N; ++i)
    // Do something
```

- or

```
#pragma simd
_Cilk_for (int i = 0; i < N; ++i)
    // Do something
```

- Combined loop gives both thread-parallelism and vectorization
- Behaves approximately like this pair of nested loops

```
_Cilk_for (int i_1 = 0; i_1 < N; i_1 += M)
    for _Simd (int i = i_1; i < i_1 + M; ++i)
        // Do something
```

- The chunk size, M, is determined by the compiler and runtime

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Improvements in Vectorization Intel® Cilk™ Plus and OpenMP* 4.0

simdlen (i.e. vectorlength) and safelen for loops

- Usable with #pragma simd (Intel Cilk™ Plus) and omp simd (OpenMP*)

```
#pragma simd safelen(9) simdlen(4)
for (i = 0; i < N; i++) {
   a[i] = a[i] + a[i-9];
}
```

```
#pragma omp simd simdlen(4,8)
for (i = 0; i < N; i++) {
 foo(a[i]); //versions for VL=4,8
}
```

Array reductions

- Fortran only

User-defined reductions

- Supported for parallel in C/C++ for POD types. No support for Fortran, SIMD, or non-POD types (C++)

omp-simd collapse(N) clause

```
#pragma omp simd collapse(2)
for (i = 0; i < N; i++)
  for (j = 0; j < 4; j++) {
    a[5*i+j] += da[i]*cf[j];
  }
```

FP-model honoring for simd loops

- No more undesirable speculations and transformations

Adjacent gathers optimization

- Replaces series of gathers with series of vector loads & sequence of permutes

Ordered blocks in SIMD context

- ordered with simd specifies structured block simd loop or SIMD function

```
#pragma omp ordered [simd]
    structured code block
#pragma simdoff
    structured code block
```

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

OpenMP* 4.1 Extensions

Support for Features in OpenMP* 4.1 Technical Report 3

- Non-structured data allocation
  - omp target [enter | exit] data
  - Begins/ends data region for the target
  - map specifies data to be mapped/unmapped with motion
- Asynchronous offload
  - nowait clause on omp task and target
  - Enables asynchronous offload using existing tasking model with nowait clause
- Dependence (signal)
  - depend clause on omp task and target
  - Enables offload with dependence
- Map clause extensions
  - Modifiers always and delete
- Available for C/C++ and Fortran

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Offload Feature Enhancements for Intel® Xeon Phi™

New Features available for C/C++

- Offload of structures with pointer members
  - Enabling offload inside member function
  - ‘pointer->field’ is now allowed in offload constructs
  - Structured objects are copied without pointer fields. This makes them bitwise copyable
  - Pointer fields transfer must be pointed out in the pragma explicitly
- Offload with MIC-only memory allocation using new modifiers
  - targetptr for MIC-only memory allocation by offload runtime
  - preallocated targetptr for informing offload runtime about user allocated memory on MIC
- Offload using Streams - new stream clause and associated APIs
  - Offload multiple concurrent computations to a coprocessor from a single CPU thread
  - stream is a logical queue of offloads
  - Offloads in any one stream complete in the order they were issued to the stream
- Performance Improvements for C/C++ and Fortran
- Asynchronous offload
- Memory Allocation and Data Transfers

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Submodules (F2008) – The Problem

```
module bigmod
…
contains
subroutine sub1
…<implementation of sub1>

function func2
…<implementation of func2>

subroutine sub47
…<implementation of sub47>

…
end module bigmod
```

```
! Source source1.f90
use bigmod
…
Call sub1
```

```
! Source source2.f90
use bigmod
…
x = func2(…)
```

```
! Source source47.f90
use bigmod
…
call sub47
```

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Submodules (F2008) – The Solution

```
module bigmod
…
interface
module subroutine sub1
…

module function func2
…

module subroutine sub47
…
end interface

end module bigmod
```

```
submodule (bigmod) bigmod_submod

contains
module subroutine sub1
… <implementation of sub1>

module function func2
… <implementation of func2>

module subroutine sub3
… <implementation of sub3>


end submodule bigmod_submod
```

Changes in the submodule do not force recompilation of uses of the module – **as long as the interface does not change**

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

IMPURE ELEMENTAL (F2008)

In Fortran 2003, ELEMENTAL procedures are PURE

- No I/O, no side-effects, can call only other PURE procedures

New IMPURE prefix allows non-PURE elemental procedures

- Can do I/O, call RANDOM NUMBER, etc.

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Further C Interoperability (F2015)

TS29113 on “Further Interoperability of Fortran with C” to be part of Fortran 2015. Motivations include:

- Support needs of MPI3
- Provide Fortran equivalent of C’s “void*” – assumed type and rank
- Enable C code to manipulate array descriptors
- Extend interoperable interfaces to ALLOCATABLE, POINTER, OPTIONAL, assumed shape, character assumed length
- Extend ASYNCHRONOUS beyond I/O
- Relaxed restrictions

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Uninitialized Variable Run-time Detection

Uninitialized variable checking using `[Q]init` option is extended to local, automatic, and allocated variables of intrinsic numeric type

Example:

```
 4  real, allocatable, dimension(:) :: A
 ...
20  ALLOCATE(A(N))
 ...
49  do i = 1, N
50     Total = Total + A(I)
51  enddo
```

```
$ ifort -init=arrays,snan -g -traceback sample.F90 -o sample.exe
$ sample.exe
forrtl: error (182): floating invalid - possible uninitialized real/complex variable.
Image              PC                Routine            Line        Source
...
sample.exe         0000000000402E12  MAIN__                   50   sample.F90
...
Aborted (core dumped)
```

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® Data Analytics Acceleration Library 2016

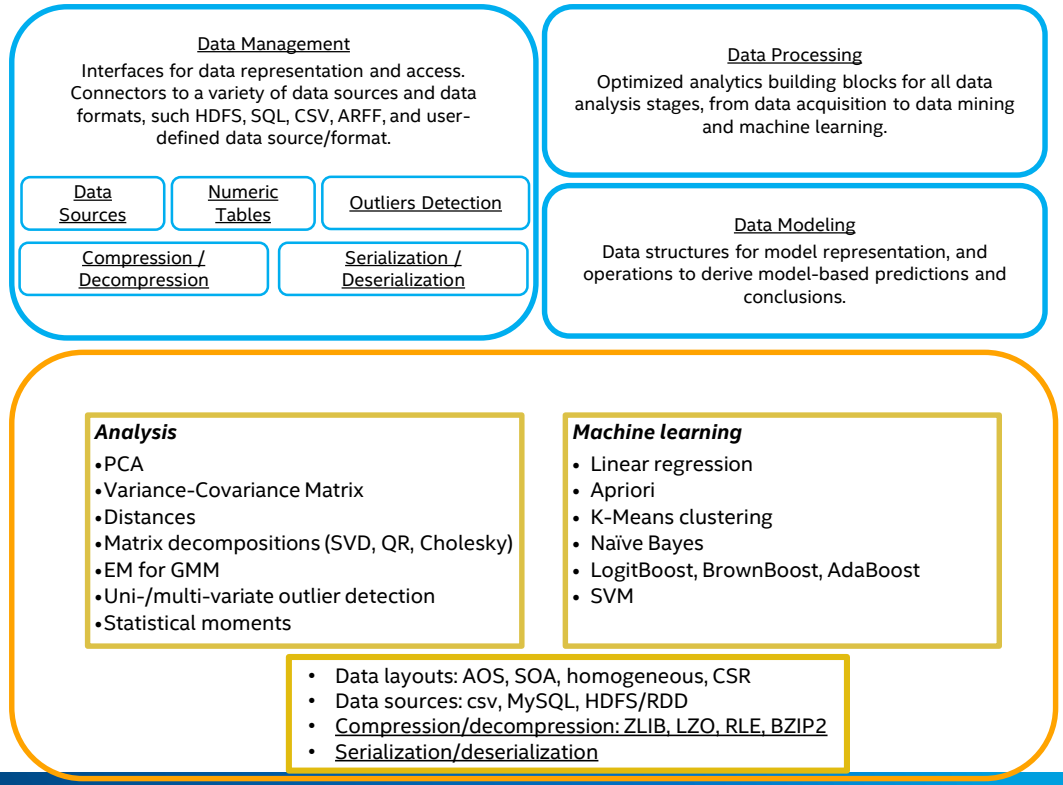
A C++ and Java API library of optimized analytics building blocks for all data analysis stages, from data acquisition to data mining and machine learning. Essential for engineering high performance Big Data applications.

New library targeting data analytics market

- **Customers:** analytics solution providers, system integrators, and application developers (FSI, Telco, Retail, Grid, etc.)
- **Key benefits:** improved time-to-value, forward-scaling performance and parallelism on IA, advanced analytics building blocks

Key features

- Building blocks highly optimized for IA to support all data analysis stages
- Support batch, streaming, and distributed processing with easy connectors to popular platforms (Hadoop, Spark) and tools (R, Python, Matlab)
- Flexible interfaces for handling different data sources (CSV, MySQL, HDFS, RDD (Spark))
- Rich set of operations to handle sparse and noisy data
- C++ and Java APIs



Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® Math Kernel Library (Intel® MKL) 11.3

Better performance with new two-stage API for Sparse BLAS routines

Additional Sparse Matrix Vector Multiplication API

- new two-stage API for Sparse BLAS level 2 and 3 routines

MKL MPI wrappers

- all MPI implementations are API-compatible but MPI implementations are **not ABI-compatible**
- MKL MPI wrapper solves this problem by providing an MPI-independent ABI to MKL

Support For Batched Small Matrix multiplication

- a single call executes multiple independent ?GEMM operation simultaneously

Support for Philox4x35 and ARS5 RNG

- two new pseudorandom number generators with a period of 2^128 are highly optimized for multithreaded environment

Sparse Solver SMP improvements

- significantly improved overall scalability for Intel Xeon Phi coprocessors and Intel Xeon processors

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® Integrated Performance Primitives 9.0

Improved Threading Support

Additional optimization for Intel® Quark™, Intel® Atom™, and the processors with Intel® AVX2 instructions support

- Intel® Quark™: data compression, cryptography optimization
- Intel® Atom™: computation vision, image processing optimization
- Intel® AVX2: computer vision, image processing optimization

New APIs to support external threading

Improved CPU dispatcher

- Auto-initialization. No need for the CPU initialization call in static libraries.
- Code dispatching based on CPU features

Optimized cryptography functions to support SM2/SM3/SM4 algorithm

Custom dynamic library building tool

New APIs to support external memory allocation

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® Threading Building Blocks 4.3
Improved cross-platform support

Fully supported tbb::task_arena

- Task arenas provide improved control over workload isolation and the degree of concurrency

Dynamic replacement of standard memory allocation routines for OS X*

- Utilize the powerful TBB scalable allocator easily on OS X

Binary files for 64-bit Android* applications were added as part of the Linux* OS package

- Improvements to the Flow Graph features

Don't forget to check out Flow Graph Designer

- Several Improvements to examples and documentation

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® Advisor XE – New! Vectorization Advisor

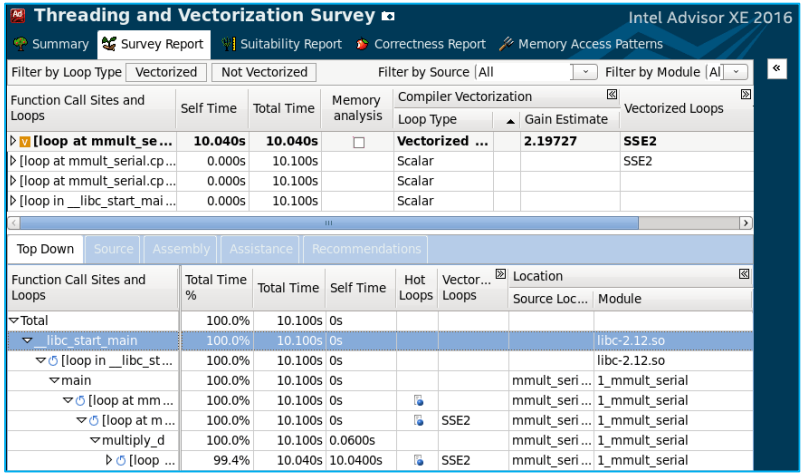
Data Driven Vectorization Design

Have you:

- Recompiled with AVX2, but seen little benefit?
- Wondered where to start adding vectorization?
- Recoded intrinsics for each new architecture?
- Struggled with cryptic compiler vectorization messages?

Breakthrough for vectorization design

- What vectorization will pay off the most?
- What is blocking vectorization and why?
- Are my loops vector friendly?
- Will reorganizing data increase performance?
- Is it safe to just use pragma simd?



More Performance
Fewer Machine Dependencies

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® Advisor XE – Vectorization Advisor

Provides the data you need for high impact vectorization

Compiler diagnostics + Performance Data = All the data you need in one place

- Find “hot” un-vectorized or “under vectorized” loops.
- Trip counts

Recommendations – How do I fix it?

Correctness via dependency analysis

- Is it safe to vectorize?

Memory Access Patterns analysis

- Unit stride vs Non-unit stride access, Unaligned memory access, etc.

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® VTune™ Amplifier XE 2016
Enhanced GPU and Microarchitecture Profiling

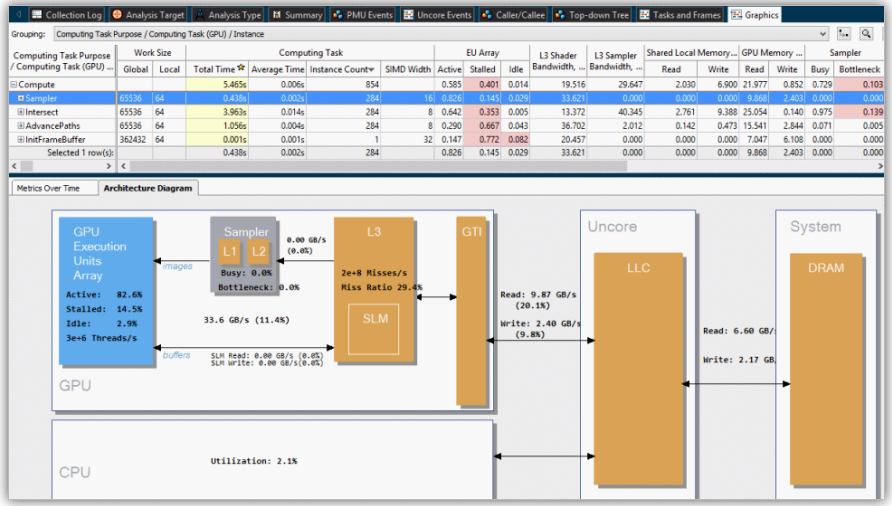
New OS and IDE support: Visual Studio* 2015 & Windows* 10 Threshold

GPU profiling

- GPU Architecture Annotation Diagram
- GPU profiling on Linux (OpenCL, Media SDK)

Microarchitecture tuning

- General Exploration analysis with confidence indication
- Driverless ‘perf’ EBS with stacks

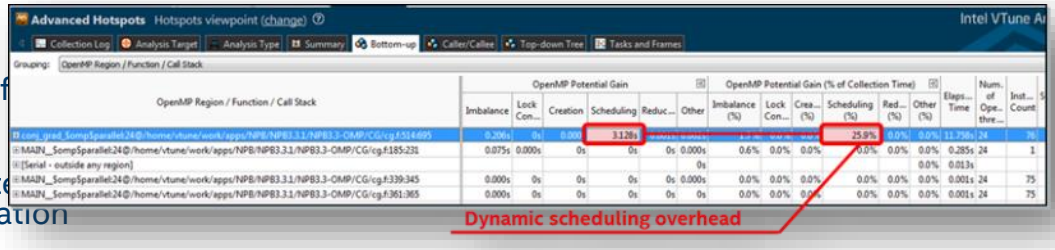


Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® VTune™ Amplifier XE 2016
Improved OpenMP and Hybrid Support

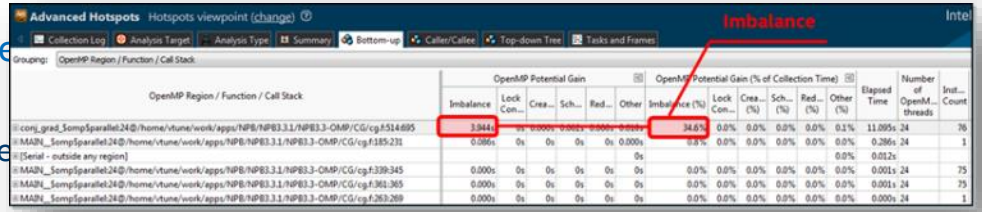
Intel OpenMP analysis enhancements

- Precise trace-based imbalance calculation that is especially useful for prof small region instances
- Classification and issue highlighting of pote gains, e.g., imbalance, lock contention, creation overhead, etc.
- Detailed analysis of barrier-to-barrier region segments



MPI+OpenMP: multi-rank analysis on a compute node

- Per-rank OpenMP potential gain and serial time metrics
- Per-rank Intel MPI communication busy wait time detection

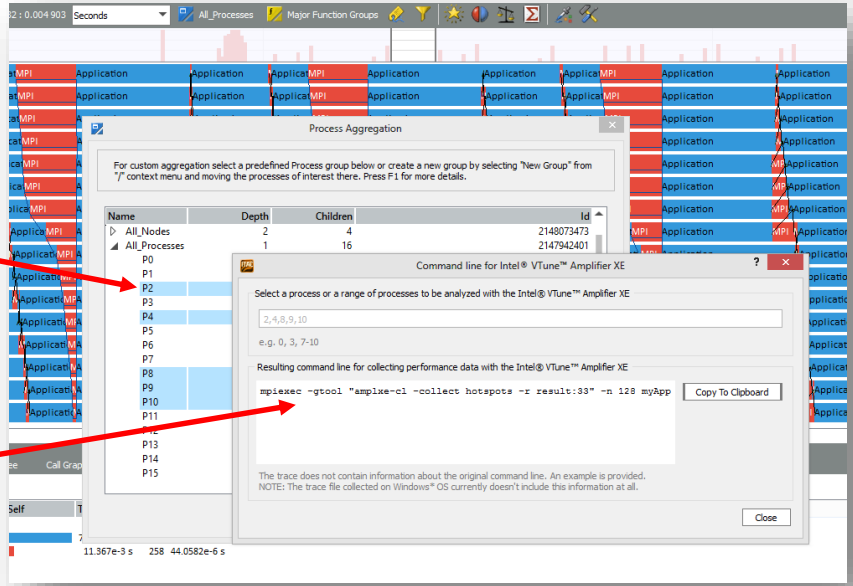


Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Improved Hybrid Profiling Support

Rank Selection/Multi-selection in ITAC for Profiling in VTune

- Automatically generate MPI command line script
- User will run command line script and automatically launch profiling of selected ranks by VTune



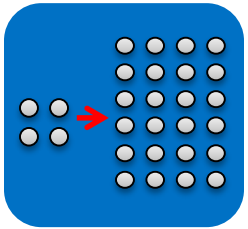
Run tool on specified ranks with gtool

```
$ mpirun –gtool “amplxe-cl –c advanced-hotspots \
     –r my_dir:all=node-wide” –n 4 –ppn 2 my_mpi_app
```

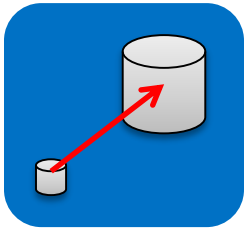
Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

MPI Performance Snapshot

Scalable profiling for MPI and Hybrid

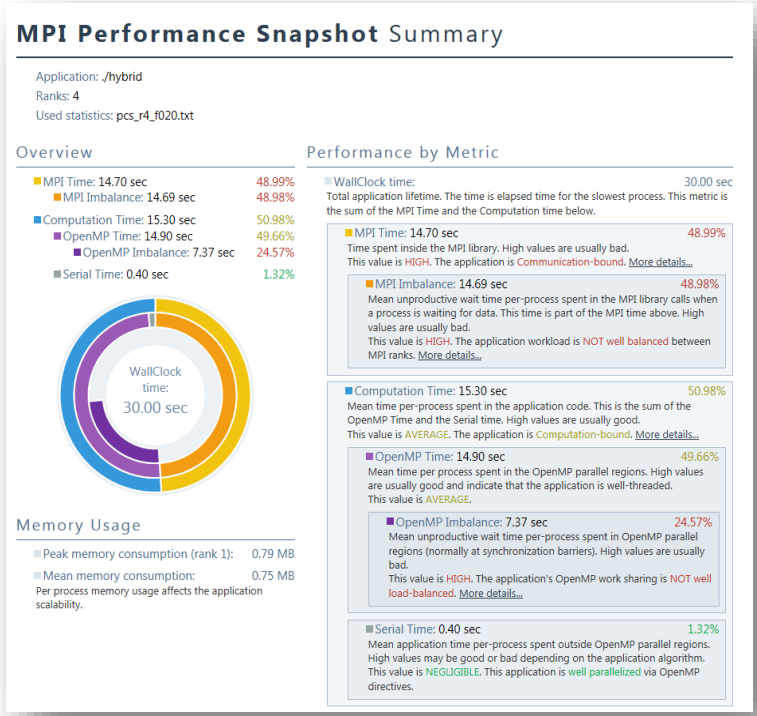


Lightweight – Low overhead profiling for 32K+ Ranks



Scalability- Performance variation at scale can be detected sooner

Identifying Key Metrics – Shows PAPI counters and MPI/OpenMP imbalances



Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

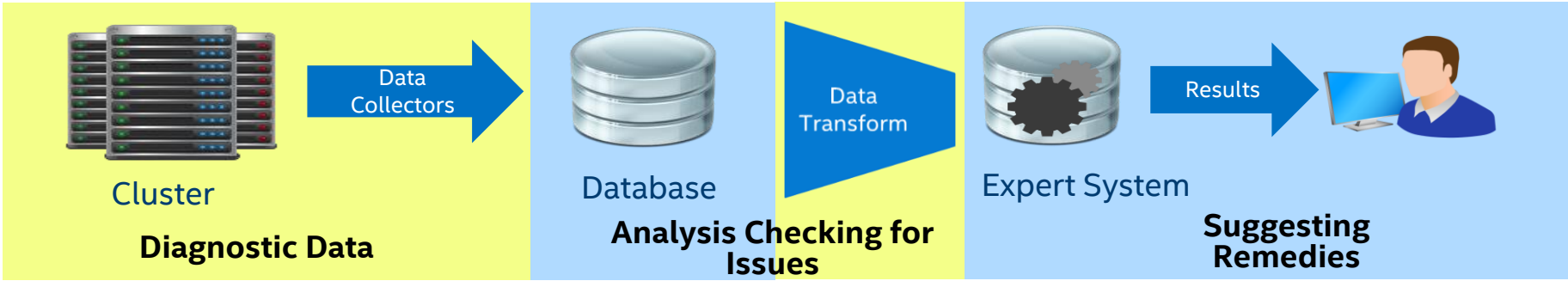
Intel® Cluster Checker 3.0
Cluster system expertise packaged into a utility

Provides Assistance

- Cluster Health Checks (on-demand, background)
- Diagnoses and remedies for common issues
- Compliance with Intel® Cluster Ready spec

Simplifies Cluster Computing Platforms

- Reduces need for specialized expertise
- Enables cluster health checks by applications
- Extensible and customizable, API



Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

HPC Webinars

Date	Title
Sep 1, 9:00 AM Pacific	What's New in Intel® Parallel Studio XE 2016?
Sep 15, 9:00 AM Pacific	Build Fast and Cross-Platform applications with Intel® Performance Libraries
Sep 22, 9:00 AM Pacific	Strategies for Tuning Multi-Level Parallelism
Sep 29, 9:00 AM Pacific	Faster Big data Analytics Using New Intel® Data Analytics Acceleration Library
Oct 6, 9:00 AM Pacific	Using Nested Parallelism in OpenMP* with James Reinders
Oct 13, 9:00 AM Pacific	From Serial to Awesome: Transforming Serial Code to Parallel Using Threading and Vectorization
Oct 20, 9:00 AM Pacific	From Serial to Awesome: Advanced Code Vectorization and Optimization
Oct 27, 9:00 AM Pacific	Performance Validation on Intel® Xeon® Processor and Xeon Phi™ Coprocessor Clusters
Nov 10, 9:00 AM Pacific	Nurture Happy Clusters With Intel® Cluster Checker

https://software.intel.com/en-us/events/hpc-webinars

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Summary/Call to Action

Current Users:

- Go to https://registrationcenter.intel.com and get the latest version

New Users:

- Evaluation available at https://software.intel.com/intel-parallel-studio-xe

All Users:

- Product forums at https://software.intel.com/forum

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.


intel®
experience
what's inside™

Legal Disclaimer & Optimization Notice

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

Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.