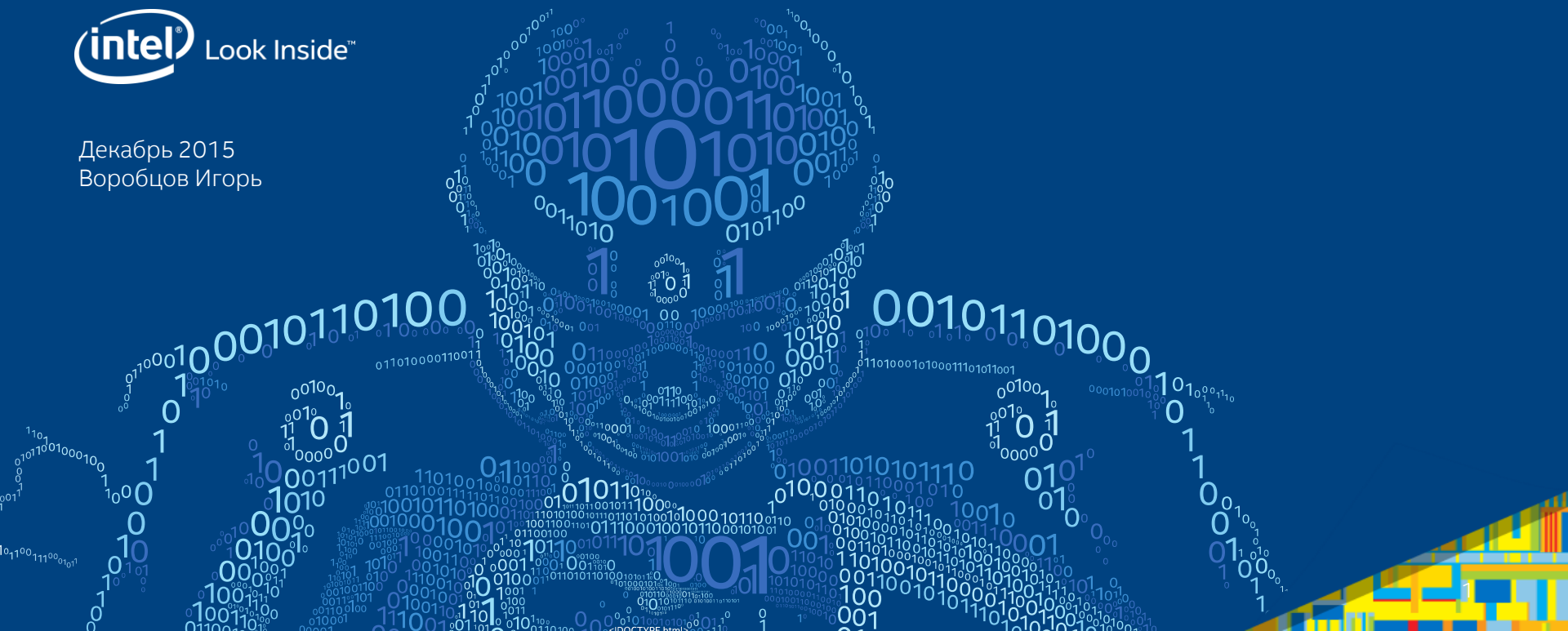


# НОВЫЕ ВОЗМОЖНОСТИ

# Intel® Parallel Studio XE 2016



Декабрь 2015  
Воробцов Игорь



# Intel® Parallel Studio XE 2016

- 1 Стандарты
- 2 Аналитика данных
- 3 Векторизация
- 4 MPI анализ



# Intel® Parallel Studio XE 2016

Много версий хороших и разных

	Composer Edition	Professional Edition	Cluster Edition
Что делает:	<i>Компилируем производительный код с помощью лучших компиляторов и библиотек</i>	<i>Добавляем средства анализа</i>	<i>Добавляем средства по работе с MPI</i>
Что включает:	<ul style="list-style-type: none"><li>• Компиляторы C++ и/или Fortran</li><li>• Библиотеки</li><li>• Параллельные модели</li></ul>	Composer edition + <ul style="list-style-type: none"><li>• Профилировка</li><li>• Моделирование параллелизма и работа с векторизацией</li><li>• Поиск ошибок с памятью и потоками</li></ul>	Professional edition + <ul style="list-style-type: none"><li>• Библиотека MPI</li><li>• Поиск ошибок и оптимизация производительности с MPI</li></ul>

# Intel® Parallel Studio XE 2016

Professional Edition

## Intel® VTune™ Amplifier XE

Профилировка  
производительности

## Intel® Advisor XE

Моделирование  
параллелизма и *работа  
с векторизацией (new)*



## Intel® Inspector XE

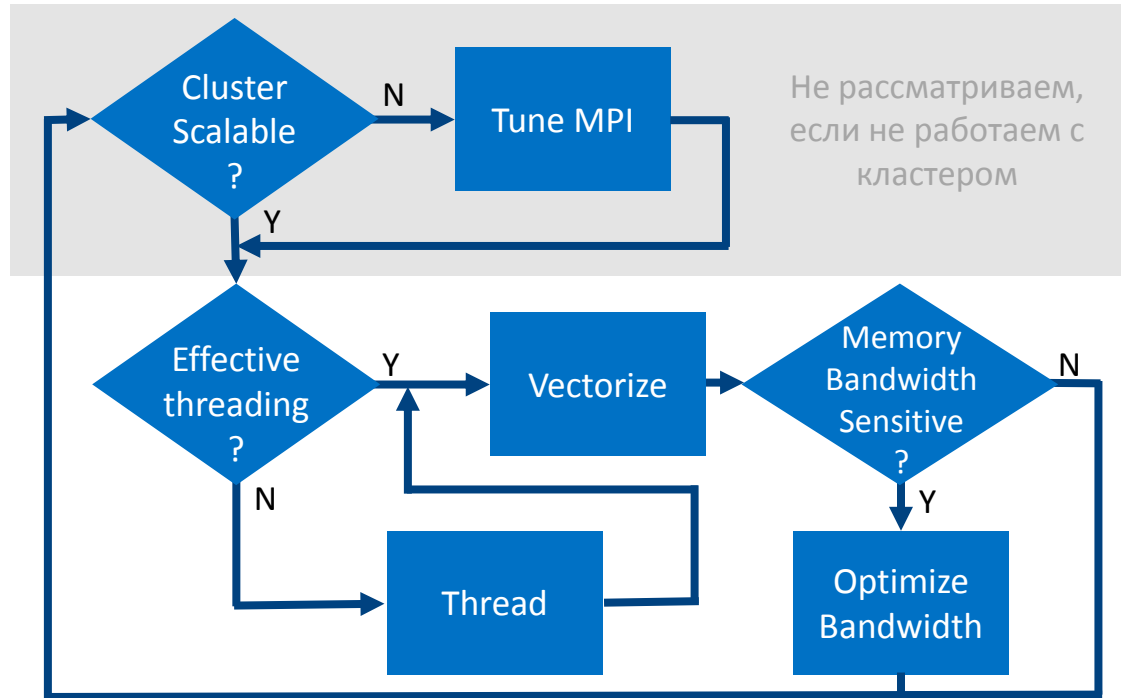
Поиск ошибок

## Intel® Parallel Studio XE Composer Edition

Компиляторы  
и библиотеки

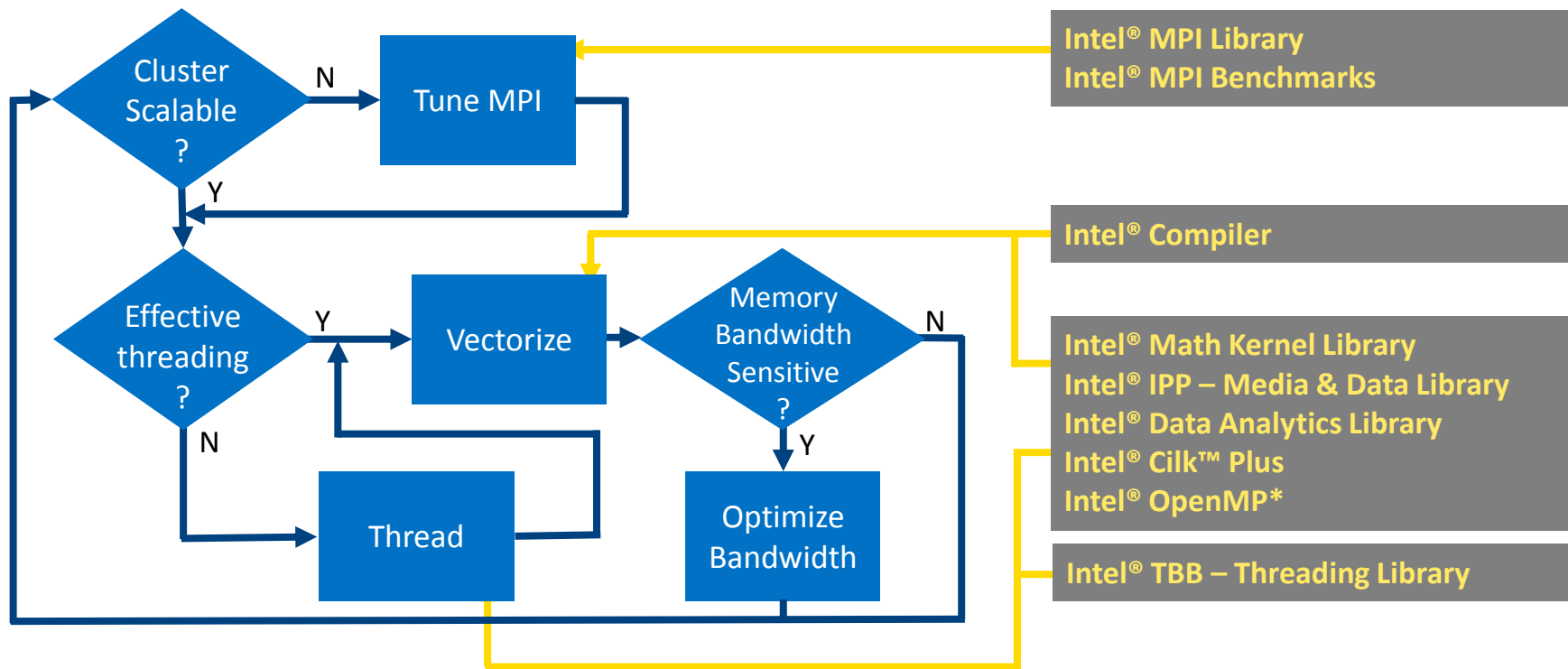
# Оптимизация производительности

Итеративный процесс...



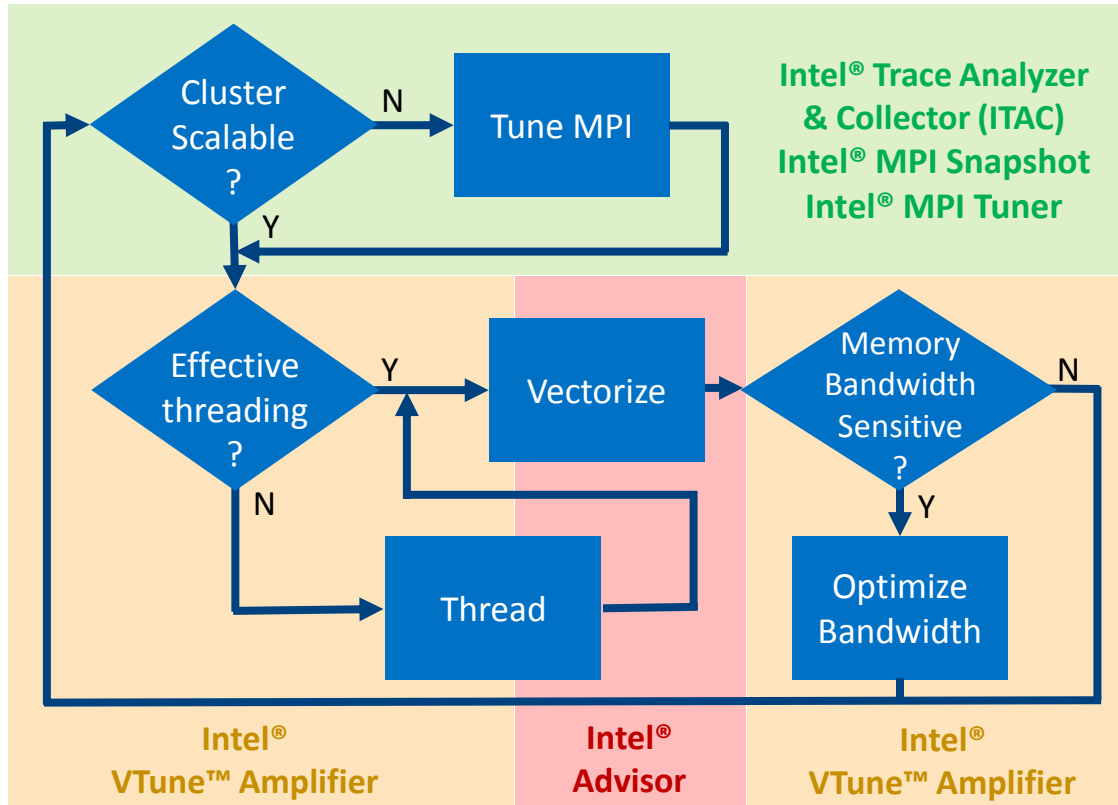
# Инструменты для реализации

Intel® Parallel Studio XE



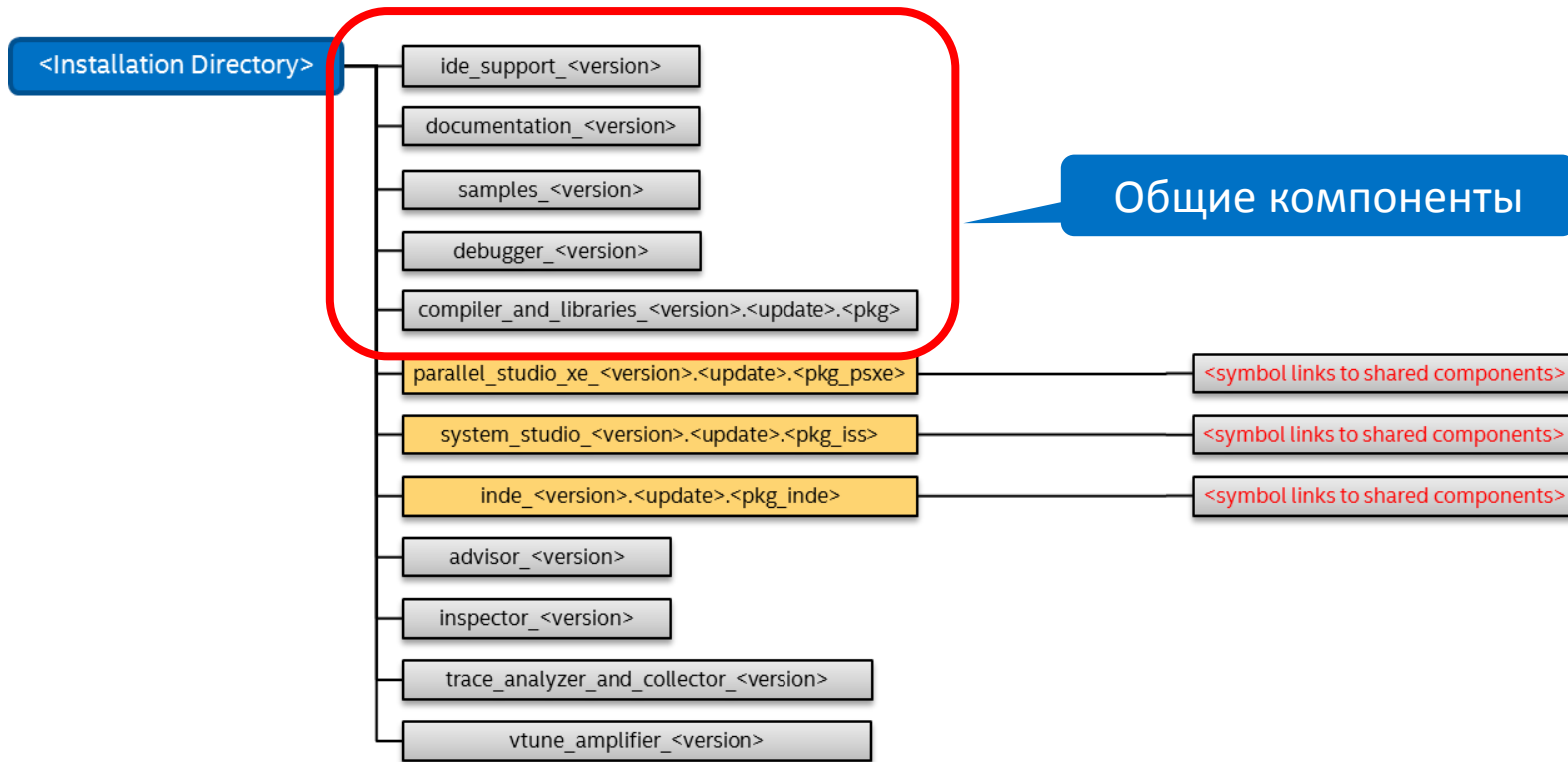
# Средства анализа производительности

Intel® Parallel Studio XE



# Новая структура директорий

Учим продукты «дружить»





# Компиляторы 16.0

Производительность с последними стандартами

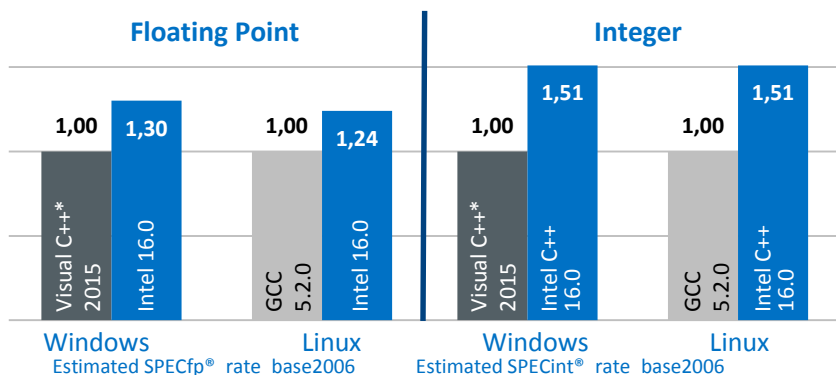


- Больше поддержки C++14
  - C++11 поддерживался полностью ещё в 15.0
- Почти полная поддержка C11
- Расширяем F2008
- Первые шаги в F2015
- OpenMP 4.1 TR3
- Новые возможности оффлоада на Intel® Xeon Phi™

# Высокая производительность Компиляторы Intel® C++ и Fortran



## Boost C++ application performance on Windows\* & Linux\* using Intel® C++ Compiler (higher is better)



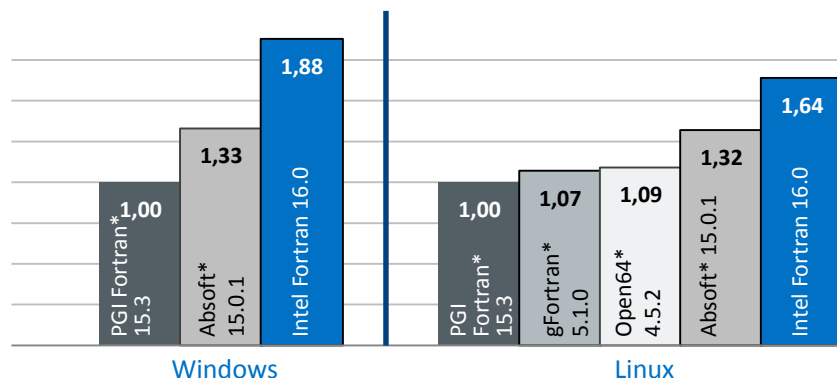
Relative geomean performance, SPEC\* benchmark - higher is better

Configuration: Windows hardware: HP DL320e Gen8 v2 (single-socket server) with Intel(R) Xeon(R) CPU E3-1280 v3 @ 3.60GHz, 32 GB RAM, HyperThreading is off; Linux hardware: HP BL460c Gen7 with Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.50GHz, 256 GB RAM, HyperThreading is on; Software: Intel C++ compiler 16.0, Microsoft (R) C/C++ Optimizing Compiler Version 19.00.23026 for x86/x64, GCC 5.2.0, Linux OS: Red Hat Enterprise Linux Server release 7.1 (Maipo), kernel 3.10.0-229.el7.x86\_64, Windows OS: Windows 8.1, SPEC\* Benchmark ( ).

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 these 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. \* Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation

**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.

## Boost Fortran application performance on Windows\* & Linux\* using Intel® Fortran Compiler (higher is better)



Relative geomean performance, Polyhedron\* benchmark - higher is better

Configuration: Hardware: Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz, HyperThreading is off, 16 GB RAM, Software: Intel Fortran compiler 16.0, Absoft\*15.0.1, PGI Fortran\* 15.3, Open64\* 4.5.2, gFortran\* 5.1.0, Linux OS: Red Hat Enterprise Linux Server release 7.0 (Maipo), kernel 3.10.0-122.el7.x86\_64, Windows OS: Windows 7, Service pack 1, Polyhedron Fortran Benchmark ( ). Windows compiler switches: Absoft: -m64 -O5 -speed\_math=10 -fast\_math -march=core -xINTEGER -stack:0x80000000, Intel\* Fortran compiler: /fast /Qparallel /link /stack:64000000, PGI Fortran: -fastsse -Munroll=n=4 -Mipa-fast\_inline -Mconcur=numa, Linux compiler switches: Absoft: -m64 -mavx -O5 -speed\_math=10 -march=core -xINTEGER, gFortran: -Ofast -mpmathsse -fno -marchnative -funroll-loops -fsee -parallelize-loops-4, Intel Fortran compiler: -fast -parallel, PGI Fortran: -fast -Mipa-fast\_inline -Msmartalloc -Mfprelaxed -Mstack\_arrays -Mconcur=bind, Open64: -march=bdver1 -mavx -mno-fma4 -Ofast -rso -qno.

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 these 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. \* Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation

**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.



# C++14

## Производительность с последними стандартами



- `/Qstd=c++14` на Windows и `-std=c++14` на Linux и Mac OS X
- <https://software.intel.com/en-us/articles/c14-features-supported-by-intel-c-compiler>
  - Обобщённые лямбда-функции
  - Захват выражений для лямбда-функций
  - Разделители разрядов
  - Атрибут `[[deprecated]]`
  - Вывод типа возвращаемого значения для функций
  - Агрегатная инициализация классов с инициализаторами полей

# C++14 TR

## Feature Test макрос



```
#if __cpp_binary_literals
int const packed_zero_to_three = 0b00011011;
#else
int const packed_zero_to_three = 0x1B;
#endif
```

```
#if __has_include("shared_mutex") // use standard header
#elif __has_include("boost/shared_mutex.h") // use BOOST header
#endif
```

# C11

## Поддержка стандарта C



- `/Qstd=c11` на Windows и `-std=c11` на Linux и Mac OS X
- <https://software.intel.com/en-us/articles/c11-support-in-intel-c-compiler>

<code>_Alignas</code>	<code>_Alignof</code>
<code>_Static_assert</code>	<code>_Thread_local</code>
<code>_Noreturn</code>	<code>_Generic</code>

# C11

## Выравнивание



- Раньше решения были зависимы от компилятора

```
__declspec(align(base)) <var>
```

```
<var> __attribute__((aligned(base)))
```

```
#ifdef __GNUC__
#define _ALIGN(N) __attribute__((aligned(N)))
#else
#define _ALIGN(N) __declspec(align(N))
#endif

_ALIGN(16) int foo[4];
```

# C11

## Выравнивание



- Раньше решения были зависимы от компилятора

```
__declspec(align(base)) <var>
```

```
<var> __attribute__((aligned(base)))
```

```
// массив cacheline выравнен по 64 байта
_Alignas(64) char cacheline[64];
printf("Alignment of char = %d\n", _Alignof(char));
```

# C11

## «Шаблоны для C»



- Выражения, не зависящие от типа, с использованием ключевого слова `_Generic`

```
#define sqrt(x) _Generic((x), long double: sqrtl, \  
                        default: sqrt, \  
                        float: sqrtf)(x)
```



# C11

## Анонимные структуры и объединения

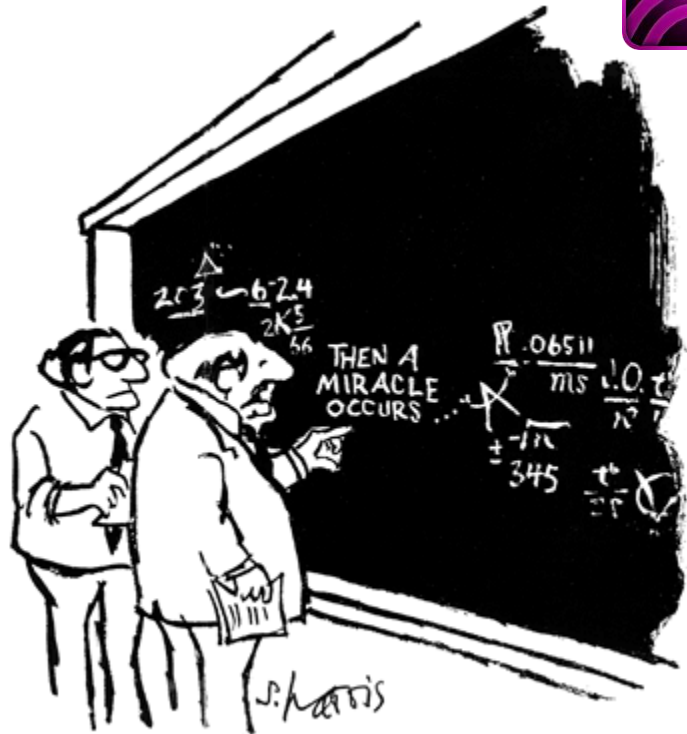


- Для вложения структур и объединений

```
struct T // C11
{
    int m;
    union //анонимное объединение
    {
        char * index;
        int key;
    };
};
struct T t;
t.key = 1300; //прямой доступ к члену объединения key
```

# Приоритет скобок

- $A+(B+C)$ 
  - компилятор не обязан соблюдать приоритет для вычислений
- `-fpprotect-parens` (Linux\* OS и OS X\*) или `/Qprotect-parens` (Windows\*)
- `fp-model`



# Оптимизация циклов

## Loop blocking



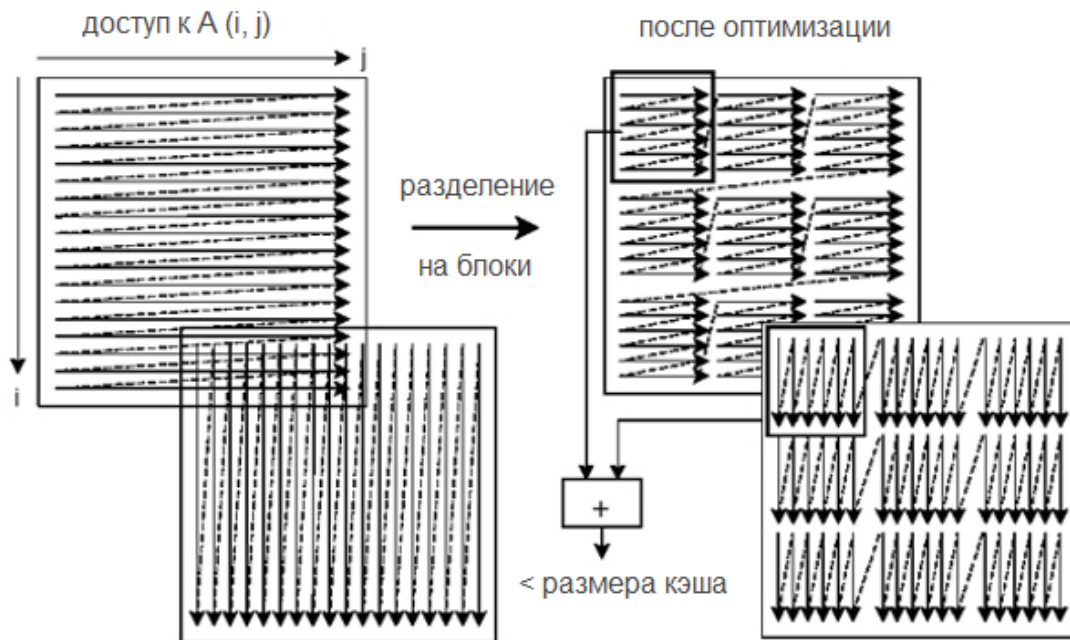
```
double A[MAX, MAX], B[MAX, MAX];
for (i=0; i< MAX; i++)
    for (j = 0; j< MAX; j++)
        A[i, j] = A[i, j] + B[j, i];
```



```
for (i = 0; i< MAX; i += block_size)
    for (j = 0; j< MAX; j += block_size)
        for (ii = i; ii<i + block_size; ii++)
            for (jj = j; jj<j + block_size; jj++)
                A[ii, jj] = A[ii, jj] + B[jj, ii];
```

# Оптимизация циклов

## Loop blocking



# Оптимизация циклов

## Директива `block_loop`



C++: `#pragma block_loop [clause]`  
`#pragma noblock_loop`

Fortran: `!DIR$ BLOCK_LOOP [clause]`  
`!DIR$ NOBLOCK_LOOP`

## Уровень оптимизации O3

```
#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;
}
```

# Оптимизация циклов

## Директива block\_loop



C++: `#pragma block_loop [clause]`  
`#pragma noblock_loop`

Fortran: `!DIR$ BLOCK_LOOP [clause]`  
`!DIR$ NOBLOCK_LOOP`

```
for (jj = 1; jj < n / 256 + 1; jj + ) {  
  for (ii = 1; ii < n / 256 + 1; ii++) {  
    for (j = (jj - 1) * 256 + 1; j < 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;  
    }  
  }  
}
```

# Подмодули из F2008

## Проблема



```
module bigmod
...
contains
subroutine sub1
...
function func2
...
subroutine sub47
...
end module bigmod
```

```
! Source source1.f90
...
Call sub1
```

```
! Source source2.f90
...
x = func2(...)
```

```
! Source
source47.f90
...
call sub47
```



# Подмодули из F2008

## Проблема



```
module bigmod
...
contains
subroutine sub1
...
function func2
...
subroutine sub47
edit
...
end module bigmod
```

```
! Source source1.f90
...
Call sub1
```

```
! Source source2.f90
...
x = func2(...)
```

```
! Source
source47.f90
...
call sub47
```



# Подмодули из F2008

## Проблема



```
module bigmod
```

```
...
```

```
contains
```

```
subroutine sub1
```

```
...
```

```
function func2
```

```
· КОМПИЛЯЦИЯ!
```

```
subroutine sub47
```

```
edit
```

```
...
```

```
end module bigmod
```

```
! Source source1.f90
```

```
· КОМПИЛЯЦИЯ!
```

```
Call sub1
```

```
! Source source2.f90
```

```
· КОМПИЛЯЦИЯ!
```

```
x = func2(...)
```

```
! Source
```

```
source47.f90
```

```
· КОМПИЛЯЦИЯ!
```

```
call sub47
```

# Подмодули из F2008

Решение



```
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
```

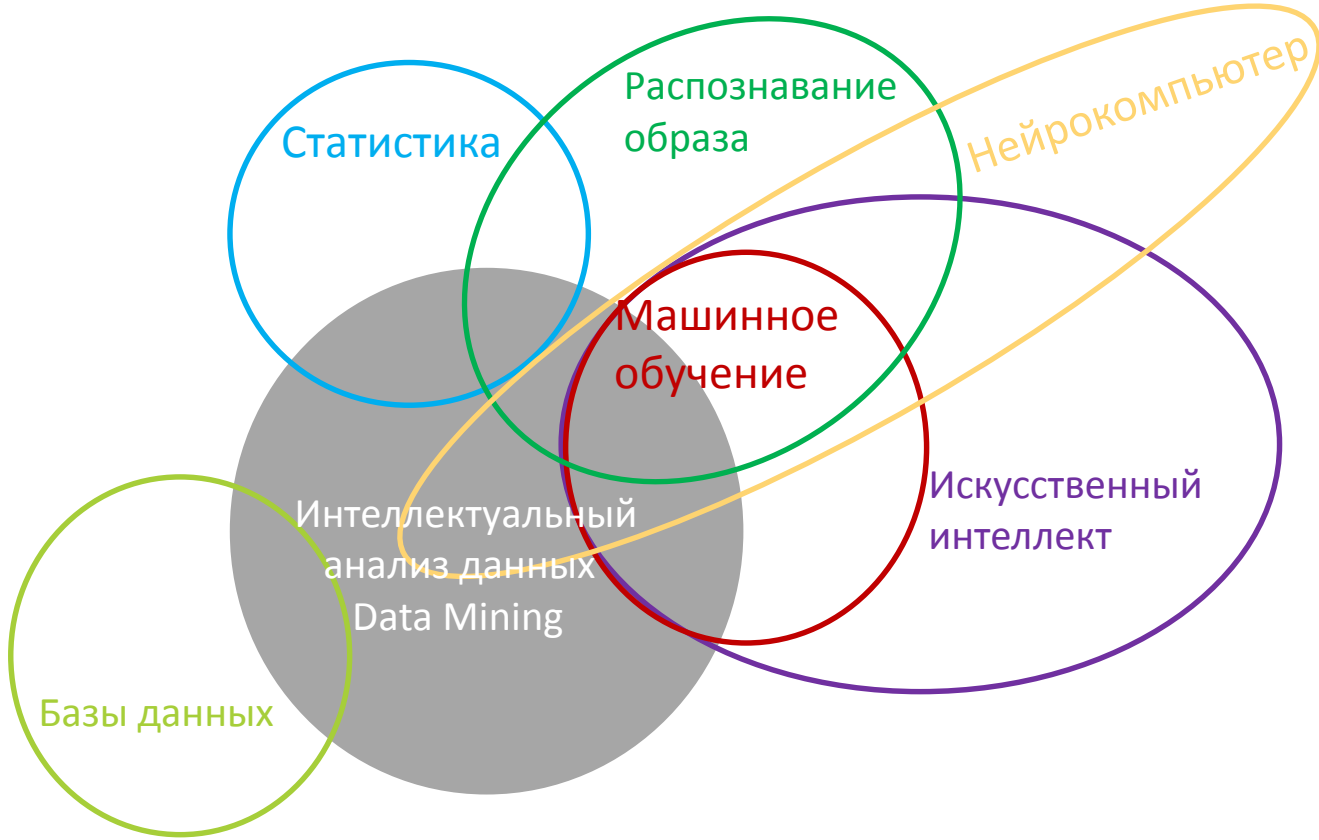
# Интероперабельность C и Фортрана

F2015



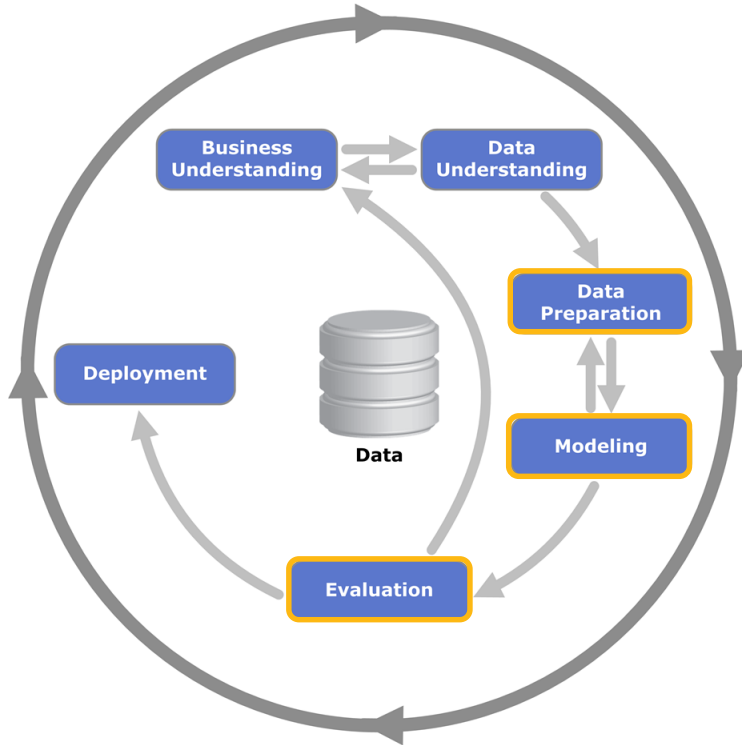
- *TS29113 (Further Interoperability of Fortran with C)*
  - передача в C функции указателей (pointer), перенимающих конфигурацию (assumed-shape) и выделяемых (allocatable) переменных
  - интероперабельность с C типом void \*
  
- *TYPE(\*)*
- *DIMENSION(..)*
- *C descriptor*

# Библиотека Intel® Data Analytics Acceleration Library 2016



# Intel® DAAL

## CRISP-DM Cross Industry Standard Process for Data Mining



- Итеративный процесс
- DAAL фокусирован на этапах подготовки данных, моделирования и оценки результатов



1) Analyze it.



2) Design it.  
(Compiler ignores these annotations.)



3) Tune it.



4) Check it.



5) Do it!



Advisor XE Workflow

- 1. Survey Target**  
[Where](#) should I consider adding parallelism? Locate the loops and functions where your program spends its time, and functions that call them.  
 Collect Survey Data  
 View Survey Result
- 2. Annotate Sources**  
Add Intel Advisor XE annotations to [identify](#) possible parallel tasks and their enclosing parallel sites.  
+ Steps to annotate  
 View Annotations
- 3. Check Suitability**  
Analyze the annotated program to check its predicted parallel [performance](#).  
 Collect Suitability Data  
 View Suitability Result
- 4. Check Correctness**  
[Predict](#) parallel data sharing problems for the annotated tasks. [Fix](#) the reported sharing problems.  
 Collect Correctness Data  
 View Correctness Result
- 5. Add Parallel Framework**  
+ Steps to replace annotations  
 View Summary



### 1. Диагностика SIMD циклов

Function Call Sites and Loops	Self Time	Total Time	Compiler/Vectorization	
			Loop Type	Why No Vectorization?
[loop in runCforallLambdaLoops]	0.094s	0.094s	Scalar	vector dependence prevents vector...
[loop in runCforallLambdaLoops]	0.140s	3.744s	Scalar	inner loop was already vectorized
[loop in std::complex_base<double,struct_C_double_complex>::i...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations				
Peeled loop; loop stats were reordered				
[loop in std::basic_string<char,struct_std::char_traits<char>,class_std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct_std::char_traits<char>,class_std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::min_put<char,class_std::ostreambuf_iterator<char,struct st...	0.000s	0.234s	Scalar	nonstandard loop is not a vectoriza...

### 2. Рекомендации

**Issue: Peeled/Remainder loop(s) present**

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials, Utilizing Full Vectors...](#)

**Recommendation: Align memory access**  
 Projected maximum performance gain: High  
 Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

```
float *array;
array = (float *)_mm_malloc(ARRAY_SIZE*sizeof(float), 32);

// Somewhere else
_assume_aligned(array, 32);
// Use array in loop
```

### 3. Анализ зависимостей

Problems and Messages						
ID	Type	Site Name	Sources	Modules	State	
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem	
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P7	Write after read dependency	site2	dqtest2.cpp, idl.e.h	dqtest2	✗ New	

### 4. Анализ доступов к памяти

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_203	runCRawLoops	runCRawLoops.cox1063	RAW1	No information available	No information available
loop_site_139	runCRawLoops	runCRawLoops.cox622	No information available	39% / 36% / 23%	Mixed strides
loop_site_160	runCRawLoops	runCRawLoops.cox925	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns	Correctness Report				
ID	Stride	Type	Source	Modules	Alignment
P22	0; 0; 1	Unit stride	runCRawLoops.cox637	lcal.exe	
<pre>635 j2 = ( j2 + 64 - 1 ) ; 636 p[4p][0] += y[12+32]; 637 p[4p][1] += z[12+32]; 638 i2 += e[12+32]; 639 j2 += f[12+32];</pre>					
P23	0; 0	Unit stride	runCRawLoops.cox638	lcal.exe	
P30	-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.cox628	lcal.exe	
<pre>626 i1 += 64 - 1; 627 j1 += 64 - 1; 628 p[4p][2] += b[j1][11];</pre>					

# 1. Диагностика SIMD

Function Call Sites and Loops	Self Time	Total Time	Compiler/Vectorization	
			Loop Type	Why No Vectorization?
[loop in runCforallLambdaLoops]	0.094s	0.094s	Scalar	vector dependence prevents vector...
[loop in runCforallLambdaLoops]	0.140s	3.744s	Scalar	inner loop was already vectorized
Loop in std::complex_base<double,struct _C_double_complex>::i...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations				
Peeled loop; loop stmts were reordered				
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	0.234s	Scalar	nonstandard loop is not a vectoriza...



# Диагностика SIMD циклов



Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 54.44s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops		
							Vector...	Efficiency	Vector L...
[loop at stl_algo.h:4740 in std:tr...]		0.170s	0.170s		Scalar	non-vectorizable loop ins...			
[loop at loopstl.cpp:2449 in s234_]	2 Ineffective peeled/rem...	0.170s	0.170s	12; 4	Collapse	Collapse	AVX	~100%	4
[loop at loopstl.cpp:2449 in s...]		0.150s	0.150s		Vectorized (Body)		AVX		4
[loop at loopstl.cpp:2449 in s...]		0.020s	0.020s	4	Remainder				
[loop at loopstl.cpp:7900 in vas_]		0.170s	0.170s	500	Scalar	vectorization possible but...			4
[loop at loopstl.cpp:3509 in s2...]	1 High vector register ...	0.160s	0.160s	12	Expand	Expand	AVX	~69%	8
[loop at loopstl.cpp:3891 in s279_]	2 Ineffective peeled/rem...	0.150s	0.150s	125; 4	Expand	Expand	AVX	~98%	8
[loop at loopstl.cpp:6249 in s414_]		0.150s	0.150s	12	Expand	Expand	AVX	~100%	4
[loop at stl_numeric.h:247 in std...]	1 Assumed dependency...	0.150s	0.150s	49	Scalar	vector dependence preve...			

Top Down Source Loop Assembly Assistance Recommendations Compiler Diagnostic Details

File: loopstl.cpp:3509 s273\_

Line	Source	Total Time	%	Loop Time	%
3504	forttime_ (&tl);				
3505	i_1 = *ntimes;				
3506	for (nl = 1; nl <= i_1; ++nl)	0.010s		0.200s	
	[loop at loopstl.cpp:3506 in s273_] Scalar Loop. Not vectorized: inner loop was already vectorized No loop transformations were applied				
3507	{				
3508	i_2 = *n;				
3509	for (i_ = 1; i_ <= i_2; ++i_)	0.010s		0.160s	
	[loop at loopstl.cpp:3509 in s273_] Vectorized AVX Loop processing Float32; Float64; Int32 data type(s) having Inserts; Extracts; Masked St				
	Selected (Total Time):	0.010s			

# Диагностика SIMD циклов



Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 54.44s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops		
							Vector...	Efficiency	Vector L...
[loop at stl_algo.h:4740 in std:tr...]		0.170s	0.170s		Scalar	non-vectorizable loop ins ...			
[loop at loopstl.cpp:2449 in s234_]	2 Ineffective peeled/rem...	0.170s	0.170s	12; 4	Collapse	Collapse	AVX	~100%	4
[loop at loopstl.cpp:2449 in s...]		0.150s	0.150s		Vectorized (Body)		AVX		4
[loop at loopstl.cpp:2449 in s...]		0.020s	0.020s	4	Remainder				
[loop at loopstl.cpp:7900 in vas_]		0.170s	0.170s	500	Scalar	vectorization possible but...			4
[loop at loopstl.cpp:3509 in s2...]	1 High vector register ...	0.160s	0.160s	12	Expand	Expand	AVX	~69%	8
[loop at loopstl.cpp:3891 in s279_]	2 Ineffective peeled/rem...	0.150s	0.150s	125; 4	Expand	Expand	AVX	~98%	8
[loop at loopstl.cpp:6249 in s414_]		0.150s	0.150s	12	Expand	Expand	AVX	~100%	4
[loop at stl_numeric.h:247 in std...]	1 Assumed dependency...	0.150s	0.150s	49	Scalar	vector dependence preve...			

Top Down Source Loop Assembly Assistance Recommendations Compiler Diagnostic Details





File: loopstl.cpp:3509 s273\_

Line	Source	Total Time	%	Loop Time	%
3504	forttime_ (&tl);				
3505	i_1 = *ntimes;				
3506	for (nl = 1; nl <= i_1; ++nl)	0.010s		0.200s	
	[loop at loopstl.cpp:3506 in s273_] Scalar Loop. Not vectorized: inner loop was already vectorized No loop transformations were applied				
3507	{				
3508	i_2 = *n;				
3509	for (i_ = 1; i_ <= i_2; ++i_)	0.010s		0.160s	
	[loop at loopstl.cpp:3509 in s273_] Vectorized AVX Loop processing Float32; Float64; Int32 data type(s) having Inserts; Extracts; Masked St...				
	Selected (Total Time):	0.010s			



# Диагностика SIMD циклов



Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization:	Efficiency	Vector Length
0.170s	0.170s		Scalar	<input checked="" type="checkbox"/> non-vectorizable loop ins ...		
0.170s	0.170s	12; 4	<a href="#">Collapse</a>	<a href="#">Collapse</a>		4
		12	Vectorized (Body)			4
		4	Remainder			
0.170s	0.170s	500	Scalar	but...		4
<b>0.160s</b>	<b>0.160s</b>	<b>12</b>	<b>Expand</b>	<b>Expand</b>		<b>8</b>
0.150s	0.150s	125; 4	<a href="#">Expand</a>	<a href="#">Expand</a>		8
0.150s	0.150s	12	<a href="#">Expand</a>	<a href="#">Expand</a>		4
0.150s	0.150s	49	Scalar	<input checked="" type="checkbox"/> vector dependence preve ...		

Время ЦПУ

Тип инструкций

Длина вектора

Количество итераций

Скалярный или векторный

# Диагностики в исходном коде



File: fractal.cpp:164 <lambda1>::operator()

Line	Source	Total Time	%
163	<pre>for (int x = x0; x &lt; x1; ++x) {</pre>		
	<pre>[loop at fractal.cpp:163 in &lt;lambda1&gt;::operator()] Scalar Loop. Not vectorized: outer loop was not auto-vectorized: consider us No loop transformations were applied</pre>		
164	<pre>for (int y = y0; y &lt; y1; ++y) {</pre>		
	<pre>[loop at fractal.cpp:164 in &lt;lambda1&gt;::operator()] Scalar Loop. Not vectorized: vectorization possible but seems inefficient. Us Loop was unrolled by 2</pre>		
165	<pre>fractal_data_array[x - x0][y - y0] = calc_one_pixel(x, y, t</pre>	10.822s	<input type="checkbox"/>
166	<pre>}</pre>		
167	<pre>}</pre>		
168	<pre>for (int y = y0, y_temp = 0; y &lt; y1; ++y, ++y_temp) {</pre>		
169	<pre>area.set_pos(0, y - y0);</pre>		
170	<pre>for (int x = x0, x_temp = 0; x &lt; x1; ++x, ++x_temp) {</pre>		
171	<pre>area.put_pixel(fractal_data_array[x_temp][y_temp]);</pre>		
172	<pre>}</pre>		
173	<pre>}</pre>	0.196s	





# Эффективность векторного цикла

Loops	Vecto...	Efficiency ▲	Estimated Gain	Vect...
⊕ [loop at lbpSUB.cpp:1280 in fPropagationS ...]	AVX	13%	0,53	4
⊕ [loop at lbpGET.cpp:152 in fGetFracSite]	AVX	30%	2,38	8
⊕ [loop at lbpGET.cpp:42 in fGetOneMassSite]	AVX	36%	2,86	8
⊕ [loop at lbpGET.cpp:78 in fGetTotMassSite]	AVX	36%	2,86	8
⊕ [loop at lbpGET.cpp:334 in fGetOneDirecSp ...]	AVX	38%	3,05	8
i> [loop at lbpBGK.cpp:840 in fCollisionBGK]	AVX	100%	2,05	2



# Части векторного цикла



**Ad** Where should I add vectorization and/or threading parallelism? 📷

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 8,52s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	🔥	💡 Vector Issues	Self Time	Total Time	Loop Type
🔍 [loop at fractal.cpp:179 in <lambda1>::op ...		💡 4 High vector ...	0,013s	12,020s	<u>Collapse</u>
↳ [loop at fractal.cpp:179 in <lambda1>::o ...	☑	💡 4 Serialized use ...	0,013s	11,281s	Vectorized (Body)
↳ [loop at fractal.cpp:179 in <lambda1>::o ...	☑	💡 2 Data type co ...	0,000s	0,163s	Peeled
↳ [loop at fractal.cpp:179 in <lambda1>::o ...	☑	💡 2 Data type co ...	0,000s	0,576s	Remainder
↳ [loop at fractal.cpp:177 in <lambda1>::oper ...	☐	💡 2 Data type co ...	0,010s	12,030s	Scalar



# Число итераций



« Ad Where should I add vectorization and/or threading parallelism? 📷

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Program time: 12.82s Vectorized Not Vectorized FILTER: All Modules

Function Call Sites and Loops	Self Time	Total Time	🔥	💡	Trip Counts			
					Median	Min	Max	Call Count
[-] v [loop at Multiply.c:53 in matvec]	11.898s	11.898s						
i> v [loop at Multiply.c:53 in matvec]	11.851s	11.851s	<input type="checkbox"/>			101	101	12000000
i> v [loop at Multiply.c:53 in matvec]	0.047s	0.047s	<input type="checkbox"/>			3	3	1000000
i> [loop at Multiply.c:53 in matvec]	0.413s	0.413s	<input type="checkbox"/>		101	101	101	2000000
[+] v [loop at Multiply.c:45 in matvec]	0.109s	12.373s		💡 1				
i> [loop at Driver.c:146 in main]	0.016s	12.483s	<input type="checkbox"/>	💡 1	1000000	1000000	1000000	1

Число  
ВЫЗОВОВ

Число  
итераций

# 1. Диагностика SIMD

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization	
			Loop Type	Why No Vectorization?
[loop in runCforallLambdaLoop]	0.094s	0.094s	Scalar	vector dependence prevents vector...
[loop in runCforallLambdaLoop]	0.140s	3.744s	Scalar	inner loop was already vectorized
[loop in std::complex<bool, double, struct_C_double_complex>::...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations				
Peeled loop; loop starts were recompiled				
[loop in std::basic_string<char, struct_std_char_traits_cchar, class_std_allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char, struct_std_char_traits_cchar, class_std_allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::num_put<char, class_std_ostreambuf_iterator<char, struct.st...	0.000s	0.234s	Scalar	nonstandard loop is not a vectoriza...

# 2. Рекомендации



## Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials, Utilizing Full Vectors...](#)

### Recommendation: Align memory access

Projected maximum performance gain: High  
Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

```
float *array;
array = (float *)_mm_malloc(ARRAY_SIZE*sizeof(float), 32);

// Somewhere else
_assume_aligned(array, 32);
// Use array in loop
```



# Рекомендации



Найдены "Vector Issues"

Elapsed time: 8,81s   Vectorized   Not Vectorized   FILTER: All Modules   All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?	Vectorized Loops		
						Vecto...	Estim...	Vector Len
i> [loop at market.cpp:476 in tb...			1,460s	Scalar				
i> [loop at arena.cpp:88 in tbb::tbb::]		0,000s	11,460s	Scalar				
[loop at fractal.cpp:179 in <lambda1>::op...	5 Ineffective ...	0,000s	2,022s	Collapse	Collapse			
i> [loop at fractal.cpp:179 in <lambda1>::o ...	2 Data type co ...	0,000s	2,022s	Remainder				

Top Down   Source   Loop Assembly   Assistance   Recommendations   Compiler Diagnostic Details

**Issue: Ineffective peeled/remainder loop(s) present**

All or some [source loop](#) iterations are not executing in the [loop body](#). Improve performance by moving source loop iterations from [peeled/remainder](#) loops to the loop body.

Disable unrolling

The [trip count](#) after loop unrolling is too small compared to the [vector length](#). To fix: Prevent loop [unrolling](#) or decrease the unroll factor using a [directive](#).

ICL/ICC/ICPC Directive	IFORT Directive
#pragma nounroll	!DIR\$ NOUNROLL
#pragma unroll	!DIR\$ UNROLL

Read More:

- [User and Reference Guide for the Intel C++ Compiler 15.0 > Compiler Reference > Pragmas > Intel-specific Pragma Reference > unroll/nounroll.](#)

# Рекомендации



Найдены "Vector Issues"

Elapsed time: 8,81s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?	Vectorized Loops		
						Vecto...	Estim...	Vector Len
[loop at market.cpp:476 in tb...			460s	Scalar				
[loop at arena.cpp:88 in tbb::tbb::]		0,000s	11,460s	Scalar				
[loop at fractal.cpp:179 in <lambda1>::op ...]	5 Ineffective ...	0,000s	2,022s	Collapse	Collapse			
[loop at fractal.cpp:179 in <lambda>::o ...]	2 Data type co ...	0,000s	2,022s	Remainder				

Top Down | Source | Loop Assembly | Assistance | Recommendations | Compiler Diagnostic Details

**Issue: Ineffective peeled/remainder loop(s) present**  
All or some [source loop](#) iterations are not executing in the [loop body](#). Improve performance by moving source loop iterations from [peeled/remainder](#) loops to the loop body.

Disable unrolling  
The [trip count](#) after loop unrolling is too small compared to the [unroll factor](#) or decrease the unroll factor using a [directive](#).

ICL/ICC/ICPC Directive	IFORT Directive
#pragma nounroll	!DIR\$ NOUNROLL
#pragma unroll	!DIR\$ UNROLL

Read More:

- [User and Reference Guide for the Intel C++ Compiler 15.0 > Compiler Reference > Pragmas > Intel-specific Pragma Reference > unroll/nounroll.](#)

Подробное описание типичных проблем



# Рекомендации



Top Down Source Loop Assembly Assistance Recommendations Compiler Diagnostic Details



2

## Issue: Serialized user function call(s) present

User-defined functions in the [loop body](#) are not vectorized.



2

### Enable inline expansion

Inlining of user-defined functions is disabled by compiler option. To fix: When using the `Ob` or `inline-level` compiler option to control inline expansion, replace the `0` argument with the `1` argument to enable inlining when an `inline` keyword or attribute is specified or the `2` argument to enable inlining of any function at compiler discretion.

Windows* OS		Linux* OS	
<a href="#">ICL Option</a>	<a href="#">IFORT Option</a>	<a href="#">ICC/ICPC Option</a>	<a href="#">IFORT Option</a>
<code>/Ob1</code> or <code>/Ob2</code>	<code>Ob1</code> or <code>Ob2</code>	<code>-inline-level=1</code> or <code>-inline-level=2</code>	<code>-inline-level=1</code> or <code>-inline-level=2</code>

**Read More:**

# Рекомендации



Top Down Source Loop Assembly Assistance Recommendations Compiler Diagnostic Details

**Issue: Serialized user function call(s) present**  
User-defined functions in the [loop body](#) are not vectorized.

2

2

Enable inline expansion

Inlining of user-defined functions is disabled by compiler option. To fix: When using the `Ob` or `inline-level` compiler option to control inline expansion, replace the `0` argument with the `1` argument to enable inlining when an `inline` keyword or attribute is specified or the `2` argument to enable inlining of any function at compiler discretion.

Windows* OS		Linux* OS	
ICL Option	IFORT Option	ICC/ICPC Option	IFORT Option
/Ob1 or /Ob2	Ob1 or Ob2	-inline-level=1 or -inline-level=2	-inline-level=1 or -inline-level=2

Read More:

Source Top Down Loop Assembly Recommendations Compiler Diagnostic Details

**Issue: Inefficient memory access patterns present**

There is a high of percentage memory instructions with irregular (variable or random) stride accesses. Improve performance by investigating an

Recommendation: Use SoA instead of AoS

An array is the most common type of data structure containing a contiguous collection of data items that can be accessed by an ordinal index. Structures (AoS) or as a structure of arrays (SoA). While AoS organization is excellent for encapsulation, it can hinder effective vector processing using SoA instead of AoS.

Read More:

## 1. Диагностика SIMD циклов

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization	
			Loop Type	Why No Vectorization?
[[loop in runCForsAllambdaLoop]	0.094s	0.094s	Scalar	vector dependence prevents vector...
[[loop in runCForsAllambdaLoop]	0.140s	3.744s	Scalar	inner loop was already vectorized...
[[loop in std::complex<bool, double, struct C_double_complex>::...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop starts were reinserted				
[[loop in std::basic_string<char, struct std::char_traits<char>, class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[[loop in std::basic_string<char, struct std::char_traits<char>, class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[[loop in std::num_put<char, class std::ostreambuf_iterator<char, struct st...	0.000s	0.234s	Scalar	nonstandard loop is not a vectoriza...

## 2. Рекомендации



### Issue: Peeled/Remainder loops present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials, Utilizing Full Vectors...](#)

### Recommendation: Align memory access

Projected maximum performance gain: High  
Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary.

```
Float *array;  
array = (Float *)_mm_malloc(ARRAY_SIZE*sizeof(Float), 32);  
  
// Somewhere else  
_mm_assume_aligned(array, 32);  
// Use array in loop
```

## 3. Анализ зависимостей

### Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	🚫 New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	🚫 New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	🚫 New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	🚫 New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	🚫 New
P7	Write after read dependency	site2	dqtest2.cpp, idle.h	dqtest2	🚫 New

# Зависимости по данным

```
DO I = 1, N
  A(I) = A(I-1) * B(I)
ENDDO
```

```
void scale(int *a, int *b)
{
  for (int i = 0; i < 1000; i++)
    b[i] = z * a[i];
}
```

## Issue: Assumed dependency present

The compiler assumed there is an anti-dependency (Write after read – WAR) or true dependency (Read after write – RAW) in the loop. Improve performance by investigating the assumption and handling accordingly.

### 🔍 Enable vectorization

Potential performance gain: Information not available until Beta Update release

Confidence this recommendation applies to your code: Information not available until Beta Update release

The Correctness analysis shows there is no real dependency in the loop for the given workload. Tell the compiler it is safe to vectorize using the `restrict` keyword or a [directive](#).

ICL/ICC/ICPC Directive	IFORT Directive	Outcome
<code>#pragma simd</code> or <code>#pragma omp simd</code>	<code>!DIR\$ SIMD</code> or <code>!\$OMP SIMD</code>	Ignores all dependencies in the loop
<code>#pragma ivdep</code>	<code>!DIR\$ IVDEP</code>	Ignores only vector dependencies (which is safest)

**Read More:**

# Анализ зависимостей



Intel Advisor XE 2016

Where should I add vectorization and/or threading parallelism?

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Program time: 12.82s Vectorized Not Vectorized FILTER: All Modules All Sources

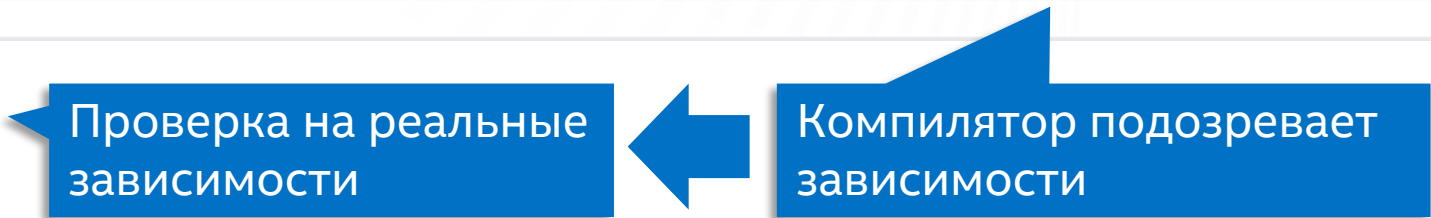
Function Call Sites and Loops	Self Time	Total Time	🔥	💡	Trip Counts	Compiler Vectorization	
						Loop Type	Why No Vectorization?
↳ [loop at Multiply.c:53 in matvec]	0.047s	0.047s	<input type="checkbox"/>		3	Vectorized (Body)	
↳ [loop at Multiply.c:53 in matvec]	0.413s	0.413s	<input type="checkbox"/>		101	Scalar	
↳ [loop at Multiply.c:45 in matvec]	0.109s	12.373s	<input type="checkbox"/>	💡 1		<a href="#">Collapse</a>	<a href="#">Collapse</a>
↳ [loop at Multiply.c:45 in matvec]	0.078s	11.930s	<input type="checkbox"/>		12	Vectorized (Body)	
↳ [loop at Multiply.c:45 in matvec]	0.031s	0.444s	<input type="checkbox"/>		2	Remainder	
↳ [loop at Driver.c:146 in main]	0.016s	12.483s	<input checked="" type="checkbox"/>	💡 1	1000000	Scalar	vector dependence prevents vectoriza ...

2.1 Check Correctness

Identify and explore loop-carried dependencies for marked loops. Fix the reported problems.

▶ 📄

Command Line



# Реально существующие зависимости

Memory Access Patterns Report	Correctness Report
-------------------------------	--------------------

## Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	loop_site_6	main.cpp	test_1.exe	✓ Not a problem
P3	Read after write dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	🚩 New
P4	Write after write dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	🚩 New
P5	Write after read dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	🚩 New

### Write after read dependency: Code Locations

ID	Description	Source	Function	Module	State
X17	Read	main.cpp:22	main	test_1.exe	🚩 New
		20	k += a[9];		
		21	k *= a[8];		
		22	k -= a[7];		
		23	k += a[6];		
		24	k *= a[5];		
X18	Read	main.cpp:23	main	test_1.exe	🚩 New
		21	k *= a[8];		
		22	k -= a[7];		
		23	k += a[6];		



## 1. Диагностика SIMD циклов

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization	
			Loop Type	Why No Vectorization?
[[loop in runCForAllLambdaLoop]	0.094s	0.094s	Scalar	vector dependence prevents vector...
[[loop in runCForAllLambdaLoop]	0.140s	3.744s	Scalar	inner loop was already vectorized...
[[loop in std::complex<bool, double, struct...>::Cdouble::complex::c...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations				
Peeled loop; loop starts were recompiled				
[[loop in std::basic_string<char, struct std::char_traits<char>, class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[[loop in std::basic_string<char, struct std::char_traits<char>, class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[[loop in std::num_put<char, class std::ostreambuf_iterator<char, struct st...	0.000s	0.234s	Scalar	nonstandard loop is not a vectoriza...

## 3. Анализ зависимостей

### Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P7	Write after read dependency	site2	dqtest2.cpp, idl.h	dqtest2	🔴 New

## 2. Рекомендации

### Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials, Utilizing Full Vectors...](#)

#### Recommendation: Align memory access

Projected maximum performance gain: High

Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary.

```
Float *array;
array = (Float *)_m_malloc(ARRAY_SIZE*sizeof(Float), 32);

// Somewhere else
__assume_aligned(array, 32);
// Use array in loop
```

## 4. Анализ доступов к памяти

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_203	runCRawLoops	runCRawLoops.cox1063	RAW1	No information available	No information available
loop_site_139	runCRawLoops	runCRawLoops.cox622	No information available	39% / 96% / 23%	Mixed strides
loop_site_160	runCRawLoops	runCRawLoops.cox925	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns		Correctness Report			
ID	Stride	Type	Source	Modules	Alignment
P22	0; 0; 1	Unit stride	runCRawLoops.cox637	lcal.exe	
635 j2 = ( j2 + 64 - 1 ) ;					
636 p[4p][0] += y[12+32];					
637 p[4p][1] += z[12+32];					
638 i2 += e[12+32];					
639 j2 += f[32+32];					
P23	0; 0	Unit stride	runCRawLoops.cox638	lcal.exe	
P30	-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.cox628	lcal.exe	
626 i1 += 64 - 1;					
627 j1 += 64 - 1;					
628 p[4p][2] += b[j1][11];					

# Шаблоны доступа к памяти

## Unit-Stride access

```
for (i=0; i<N; i++)  
    A[i] = C[i]*D[i]
```



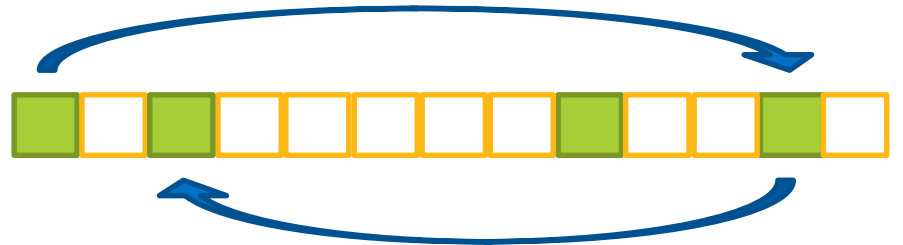
## Constant stride access

```
for (i=0; i<N; i++)  
    point[i].x = x[i]
```



## Variable stride access

```
for (i=0; i<N; i++)  
    A[B[i]] = C[i]*D[i]
```



# Анализ шаблонов доступа



Where should I add vectorization and/or threading parallelism?

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 8,52s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?
[loop at fractal.cpp:179 in <lambda1>::op ...	4 High vector ...	0,013s	12,020s	Collapse	Collapse
i> [loop at fractal.cpp:179 in <lambda1>::o ...	4 Serialized use ...	0,013s	11,281s	Vectorized (Body)	
i> [loop at fractal.cpp:179 in <lambda1>::o ...					
i> [loop at fractal.cpp:179 in <lambda1>::o ...					
i> [loop at fractal.cpp:177 in <lambda1>::oper ...					

Выделяем интересующие нас циклы

2.2 Check Memory Access Patterns

Identify and explore complex memory accesses for marked loops. Fix the reported problems.

Command Line

Запускаем анализ шаблонов доступа

# Шаблоны доступа

Unit/Constant/Variable



Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_133	grid_intersect	grid.cpp:559	No information available	23% / 1% / 76%	Mixed strides
loop_site_131	grid_intersect	grid.cpp:581	No information available	21% / 4% / 75%	Mixed strides
loop_site_145	grid_intersect	grid.cpp:562	No information available	21% / 4% / 75%	Mixed strides
loop_site_135	initialize_2D_buffer	find_hotspots.cpp:92	No information available	42% / 0% / 58%	Mixed strides

Memory Access Patterns Report    Correctness Report

ID	Icon	Stride	Type	Source	Modules	Alignment
P17	📄	8	Constant stride	intersect.cpp:141	find_hotspots.exe	
P19	📄	8	Constant stride	intersect.cpp:141	find_hotspots.exe	

```
139
140  intstruct->num++;
141  intstruct->list[intstruct->num].obj = obj;
142  intstruct->list[intstruct->num].t = t;
143 }
```

Constant stride, "Array of Structures"

P1.	📄	0	Unit stride	intersect.cpp:141	find_hotspots.exe	
P1.	📄	0	Unit stride	intersect.cpp:141	find_hotspots.exe	
P2.	📄	-8; -2; 0; 1; 2; ...	Variable stride	intersect.cpp:141	find_hotspots.exe	
P2.	📄	-8; -2; 0; 1; 2; ...	Variable stride	intersect.cpp:141	find_hotspots.exe	
P21	📄	4	Constant stride	intersect.cpp:142	find_hotspots.exe	

Unit stride access

Variable or random access

# Intel Advisor XE



Все диагностики в одном месте

Рекомендации по оптимизации

Проверка зависимостей

Анализ шаблонов доступа

Location	Count	Time	Category	Architecture	Percentage	
[loop in tbb::internal::tbb::internal:: at mark...	0,000s	11,534s	Scalar			
[loop in tbb::tbb:: at arena.cpp:88]	0,000s	11,534s	Scalar			
[loop in <lambda1>::operator() at fractal...	2 Hi...	0,000s	12,534s	Vectorized: Coll...	AVX2	-10%
[loop in <lambda1>::operator() at fracta...	2 Ser...	0,000s	12,201s	Vectorized (Body)	AVX2	
[loop in <lambda1>::operator() at fracta ...		0,000s	0,333s	Remainder		

Issue: Serialized user function call(s) present

User-defined functions in the [loop body](#) are not vectorized.

- Enable inline expansion

Inlining of user-defined functions is disabled by compiler option. To fix: When using the `Ob` or `inline-level` control inline expansion, replace the `0` argument with the `1` argument to enable inlining when an `inline` specified or the `2` argument to enable inlining of any function at compiler discretion.

P1	Parallel site information	loop_site_6	main.cpp
P3	Read after write dependency	loop_site_6	crtexe.c; main.cpp
P4	Write after write dependency	loop_site_6	crtexe.c; main.cpp
P5	Write after read dependency	loop_site_6	crtexe.c; main.cpp

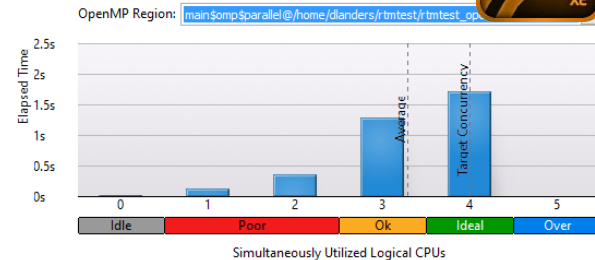
```
0: 0; 0 Unit stride
P30 -1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801 Variable stride
626 i1 += 64-1;
627 j1 += 64-1;
628 p[ip][2] += b[j1][i1];
629 p[ip][3] += c[j1][i1];
630 p[ip][0] += p[ip][2];
```

# Intel® VTune™ Amplifier XE

- Масштабируемость OpenMP
- Гибридный анализ с MPI
- OpenCL & GPU анализ
- Улучшенный анализ доступа к памяти
- Быстрее и проще
- Добавлена поддержка VM
- Последние процессоры & ОС

## OpenMP Region CPU Usage Histogram

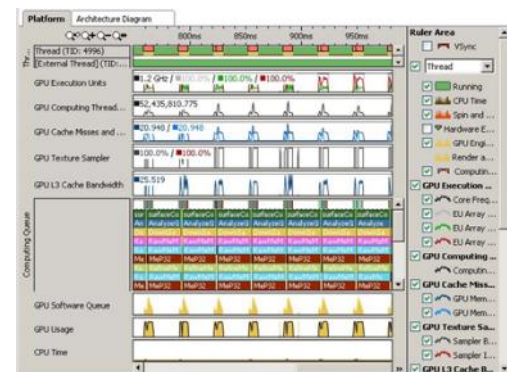
This histogram displays a percentage of the wall time the specific number of simultaneously in an OpenMP region.



## Top OpenMP Processes by MPI Communication Spin Time

This section lists process with the lowest MPI communication spin time.

Process	PID	MPI Communication Spinning (s)	(%)	OpenMP Potential Gain (%)	Serial Time (s)	(%)	
<a href="#">bt-mz.B.4</a>	125904	0.020s	0.2%	3.392s	31.2%	0.251s	2.3%
<a href="#">bt-mz.B.4</a>	125902	0.040s	0.4%	3.431s	31.6%	0.291s	2.7%
<a href="#">bt-mz.B.4</a>	125905	0.321s	3.0%	3.025s	27.9%	0.659s	6.1%
<a href="#">bt-mz.B.4</a>	125903	0.441s	4.1%	3.147s	29.0%	0.608s	5.6%





# Intel® VTune™ Amplifier XE

## Эффективное использование OpenMP



1) **OpenMP Analysis. Collection Time: 14.490**

Serial Time (outside any parallel region): 4.020s (27.7%)  
Serial Time of your application is high. It directly impacts application Elapsed Time and scalability. Explore options for parallelization, algorithm or microarchitecture tuning of the serial part of the application.

2) **Parallel Region Time: 10.469s (72.3%)**

Estimated Ideal Time: 7.115s (49.1%)  
Potential Gain: 3.354s (23.1%)  
The time wasted on load imbalance or parallel work arrangement is significant and negatively impacts the application performance and scalability. Explore OpenMP regions with the highest metric values. Make sure the workload of the regions is enough and the loop schedule is..

3) **Top OpenMP Regions by Potential Gain**

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	Potential Gain (%)	Elapsed Time
<a href="#">conj_grad_omp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:514:695</a>	3.294s 22.7%	10.208s
<a href="#">MAIN__omp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:185:231</a>	0.059s 0.4%	0.260s

4)

Ответы на интересующие вопросы:

- 1) Насколько велика последовательная часть приложения?
- 2) Сколько можно «выжать» при оптимизации OpenMP?
- 3) На какие OpenMP регионы/циклы/барьеры обратить внимание?
- 4) Что неэффективного в каждом регионе?

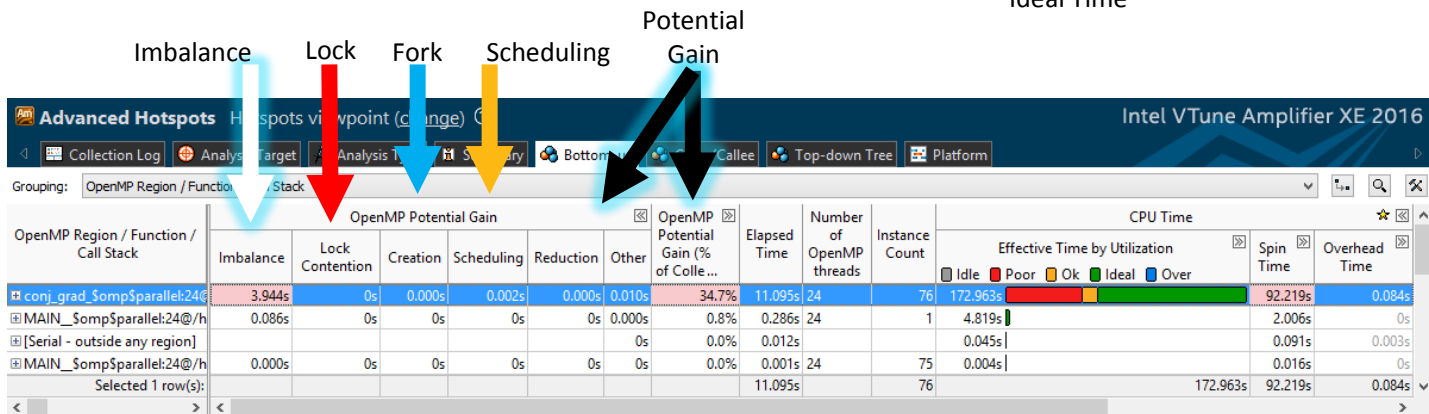
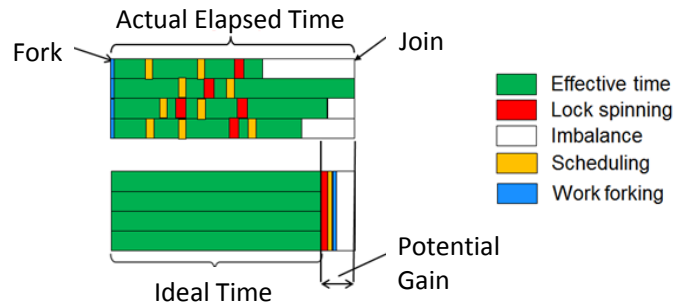


# Intel® VTune™ Amplifier XE

## Эффективное использование OpenMP



- Показывает то, что важно
  - Какой регион неэффективен?
  - Насколько можем улучшить?
  - Почему неэффективен?  
Дисбаланс? Планировка? Lock spinning?
  - Поддержка Intel® Xeon Phi



# Intel® VTune™ Amplifier XE

## Определяем неэффективные регионы OpenMP



Advanced Hotspots Hotspots viewpoint (change) ?

Collection Log Analysis Target Analysis Type Summary Bottom-up Caller/Callee Top-down Tree Tasks and Frames

Grouping: OpenMP Region / Function / Call Stack

OpenMP Region / Function / Call Stack	OpenMP Potential Gain						OpenMP Potential Gain (% of Collection Time)					
	Imbalance	Lock Con...	Crea...	Sch...	Redu...	Other	Imbalance (%)	Lock Con...	Crea...	Sch...	Redu...	Other (%)
conj_grad_Somp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:514:695	3.944s	0s	0.000s	0.002s	0.000s	0.010s	34.6%	0.0%	0.0%	0.0%	0.0%	0.1%
MAIN_Somp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:185:231	0.086s	0s	0s	0s	0s	0.000s	0.8%	0.0%	0.0%	0.0%	0.0%	0.0%
[Serial - outside any region]						0s						0.0%
MAIN_Somp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:339:345	0.000s	0s	0s	0s	0s	0s	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MAIN_Somp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:361:365	0.000s	0s	0s	0s	0s	0s	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MAIN_Somp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:263:269	0.000s	0s	0s	0s	0s	0s	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Дисбаланс

Advanced Hotspots Hotspots viewpoint (change) ?

Collection Log Analysis Target Analysis Type Summary Bottom-up Caller/Callee Top-down Tree Tasks and Frames

Grouping: OpenMP Region / Function / Call Stack

OpenMP Region / Function / Call Stack	OpenMP Potential Gain						OpenMP Potential Gain (% of Collection Time)					
	Imbalance	Lock Con...	Creation	Scheduling	Reduc...	Other	Imbalance (%)	Lock Con...	Crea...	Scheduling (%)	Redu...	Other (%)
conj_grad_Somp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:514:695	0.206s	0s	0.000	3.128s	0.001s	0.002s	1.7%	0.0%	0.0%	25.9%	0.0%	0.0%
MAIN_Somp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:185:231	0.075s	0.000s	0s	0s	0s	0.000s	0.6%	0.0%	0.0%	0.0%	0.0%	0.0%
[Serial - outside any region]						0s						0.0%
MAIN_Somp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:339:345	0.000s	0s	0s	0s	0s	0.000s	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Возможная причина:  
Накладные расходы от динамической планировки



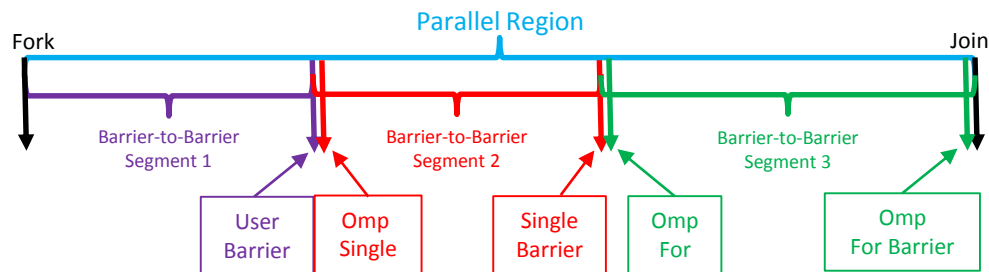
# Intel® VTune™ Amplifier XE



Определяем причину проблем

- Подробный анализ (Barrier to Barrier)
  - Рассматриваем каждый сегмент отдельно
  - Проще видеть возможности к улучшению

```
#pragma omp parallel
{
  ...
#pragma omp barrier
#pragma omp single
{
  ...
{
#pragma omp for
{
```



Grouping: OpenMP Region / OpenMP Barrier-to-Barrier Segment / Function / Call Stack

OpenMP Region / OpenMP Barrier-to-Barrier Segment / Function / Call Stack	OpenMP Potential Gain						OpenMP Potential Gain (% of Collection ...)
	Imbalance	Lock Contention	Creation	Scheduling	Reduction	Other	
3.3-OMP/CG/cg.f:514:695	3.944s	0s	0.000s	0.002s	0.000s	0.010s	34.7%
VPB3.3.1/NPB3.3-OMP/CG/cg.f:580	3.725s	0s	0s	0.000s	0s	0.008s	32.8%
VPB3.3.1/NPB3.3-OMP/CG/cg.f:683	0.149s	0s	0s	0s	0s	0.000s	1.3%
VPB3.3.1/NPB3.3-OMP/CG/cg.f:664	0.014s	0s	0s	0.000s	0s	0.000s	0.1%

# Intel® VTune™ Amplifier XE

Он стал быстрее!



	Initial Release 2015 Mid-Size Data	Initial Release 2015 Large Data	Initial Release 2016 Both Sizes
Open Summary	10-40 seconds	1-2 minutes	~4-5 seconds
Open Timeline	10-40 seconds	2-5 minutes	~1-3 seconds
Zoom Timeline	5-10 seconds	1-2 minutes	~ 1 second
Grid Node Expand	5-15 seconds	>1 minute	<1 – 2 seconds
Finalization			Usually <2x slower Sometimes faster

- При работе с большими размерами данных получаем наибольшее ускорение.
- При маленьких размерах – незначительное улучшение.

Совет: для лучшей производительности избегайте медленной графики VNC. Запускайте UI локально. Импортируйте данные с удаленной системы.



Данные получены по результатам внутреннего тестирования. Ваши результаты могут отличаться. Результаты зависят от характеристик системы, приложения и размера данных.

# Отзывы разработчиков

"Intel® VTune™ Amplifier XE analyzes complex code and helps us identify bottlenecks rapidly. By using it and other Intel® Software Development Tools, we were able to improve PIPESIM performance up to 10 times compared with the previous software version."

*Rodney Lessard  
Senior Scientist  
Schlumberger*

*Carlos Boneti  
HPC software engineer,  
Schlumberger*

"Intel® Advisor XE has been extremely helpful in identifying the best pieces of code for parallelization. We can save several days of manual work by targeting the right loops. At the same time, we can use Advisor to find potential thread safety issues to help avoid problems later on."

Intel® Inspector XE has dramatically sped up our ability to find/fix memory problems and track down difficult to isolate threading errors before our packages are released to the field.

*Peter von Kaenel, Director, Software  
Development, Harmonic Inc.*

# Call to Action

- Пробуем сегодня!

<https://software.intel.com/en-us/intel-parallel-studio-xe/try-buy>

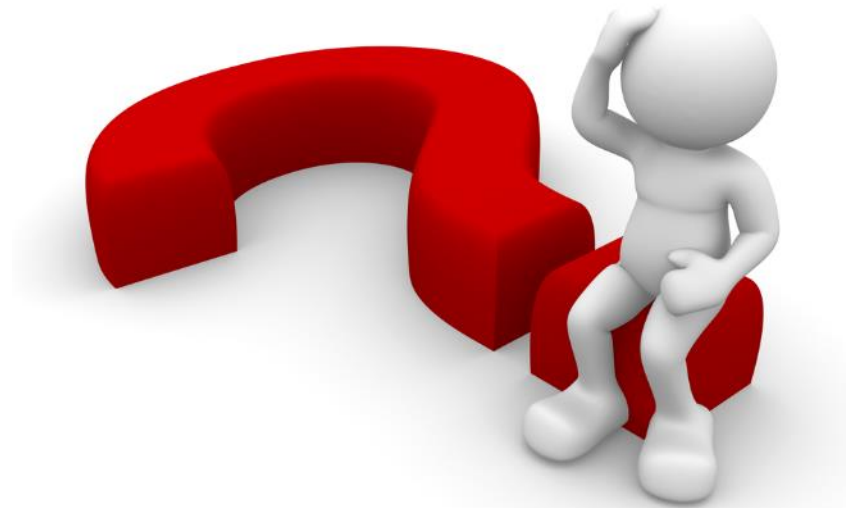
- Intel® Premier Support

- Ваш отзыв по использованию

Intel® Parallel Studio XE 2016 важен!



# Q&A



# 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



