Intel Parallel Studio XE 2015

Воробцов Игорь

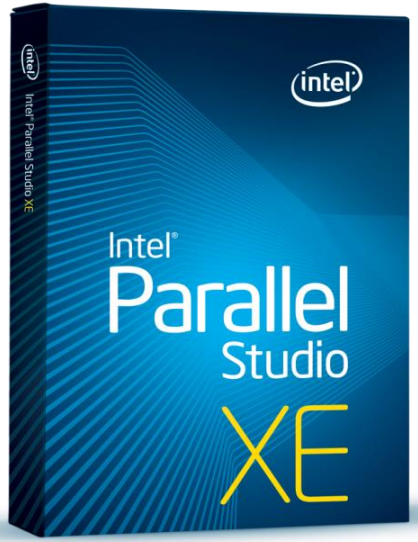

Scaling
KNC
PARDISO
ExplicitVectorization
AVX2
Broadwell
OSX
OpenMP4.0
Haswell
IntelOpenCL
C99
Fortran2008
Fortran
IntelMPI
Optimization
C++03
Fortran90
Fortran66
C++98
150k+processes
IntelMKL
Resources
IntelThreadingBuildingBlocks
C++11
Windows
C++TR1
Fortran2003
Multicore
KNL
Fortran77
Many-core
Microarchitecture
Compilers
IntelVTuneAmplifierXE
Innovation
CodeAnalysis
AVX-512
IntelCilkPlus
IvyBridge
C++
Coding
3XFasterCode
MPIScaling
TSX
Linux
ThreadingErrorAnalysis
latency

Как максимально использовать «железо»?

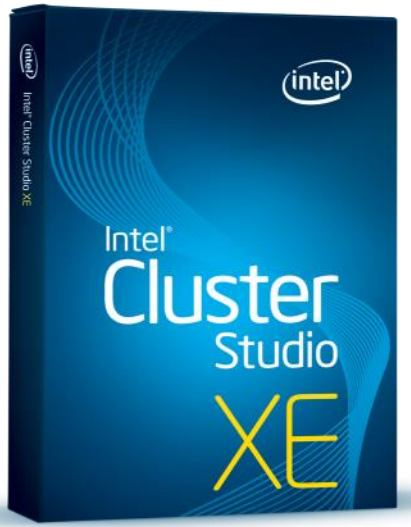
Что делать?	Требуемые усилия
Использовать новые инструкции	
Добиться стабильности базовой версии	
Распараллелить	
Добиться стабильности параллельной версии	
Найти узкие места, сдерживающие производительность	
Оптимизировать под микроархитектуру	

Продукты для разработчиков
“имена минувших дней”

Компиляторы и библиотеки



Пакет средств для систем с общей памятью



Пакет средств для систем с распределенной памятью

Продукты для разработчиков

“новые старые герои”

Intel® Parallel Studio XE 201
Composer Edition

- Intel® C++ Compiler
- Intel® Fortran Compiler
- Intel® Threading Building Blocks
- Intel® Integrated Performance Primitives
- Intel® Math Kernel Library
- Intel® Cilk™ Plus
- Intel® OpenMP*

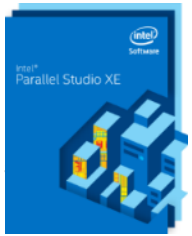
Компиляторы и библиотеки

Intel® Parallel Studio XE 201
Professional Edition

- Intel® C++ Compiler
- Intel® Fortran Compiler
- Intel® Threading Building Blocks
- Intel® Integrated Performance Primitives
- Intel® Math Kernel Library
- Intel® Cilk™ Plus
- Intel® OpenMP*

- Intel® Advisor XE
- Intel® Inspector XE
- Intel® VTune™ Amplifier XE

Пакет средств для систем с общей памятью



Intel® Parallel Studio XE 201
Cluster Edition

- Intel® C++ Compiler
- Intel® Fortran Compiler
- Intel® Threading Building Blocks
- Intel® Integrated Performance Primitives
- Intel® Math Kernel Library
- Intel® Cilk™ Plus
- Intel® OpenMP*

- Intel® Advisor XE
- Intel® Inspector XE
- Intel® VTune™ Amplifier XE

- Intel® MPI Library
- Intel® Trace Analyzer and Collector

Пакет средств для систем с распределенной памятью

Intel® Parallel Studio XE
Professional Edition

Intel® VTune™ Amplifier XE

Профилировка производительности

Intel® Advisor XE

Моделирование параллелизма

Intel® Inspector XE

Поиск ошибок

Intel® Parallel Studio XE Composer Edition

Компиляторы и библиотеки

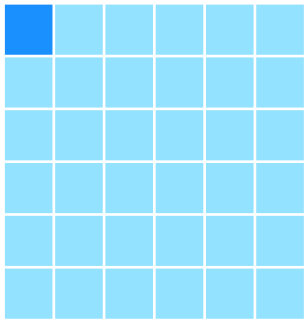
Как максимально использовать «железо»?

Что делать?	Требуемые усилия
Использовать новые инструкции	
Добиться стабильности базовой версии	
Распараллелить	
Добиться стабильности параллельной версии	
Найти узкие места, сдерживающие производительность	
Оптимизировать под микроархитектуру	

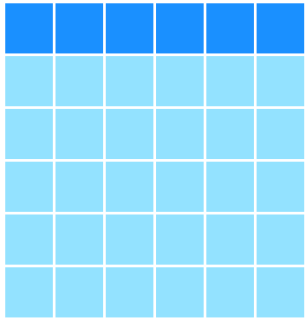
Максимум производительности

Где он?

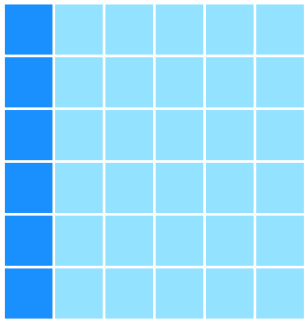
Скалярный код



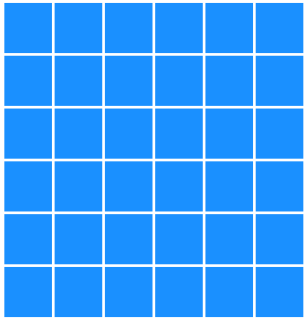
Векторный код



Параллельный код



Вместе - лучше



Максимум производительности

Как достичь?

- Гонка мегагерц

  Производительность «даром»

- Ширина команды SIMD **по данным (DLP)**
  - Перекомпиляция
  - Возможно, локальная переработка кода
  - Изредка, перепроектирование
  - Перекомпиляция для каждого нового поколения

- Многоядерность **по задачам (TLP)**
  - Перепроектирование
  - Возможно, глобальная переработка кода
  - Перекомпиляция
  - В дальнейшем «бесплатная» масштабируемость

Почему компилятор Intel® ?

Параллелизм!

Векторизация

- Улучшенная автоматическая векторизация
- Векторные конструкции в языке (Cilk™ Plus)
- Низкоуровневые возможности

Многопоточность и многоядерность

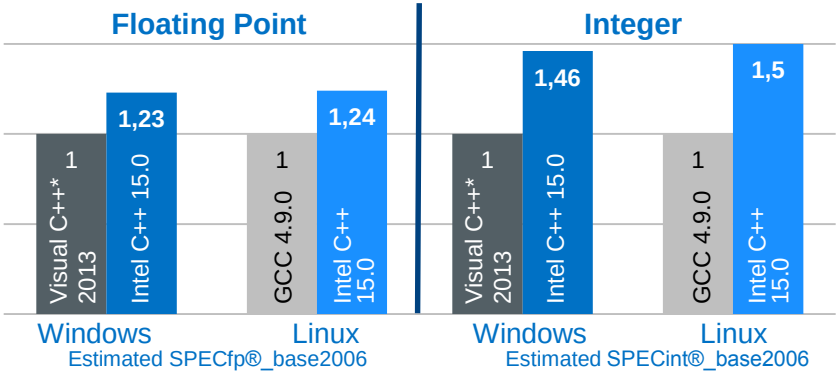
- Расширения языка (Cilk™ Plus, OpenMP 4.0)
- C++ библиотеки для многопоточности (TBB)
- Улучшенная автоматическая параллелизация

Оптимизированные библиотеки – IPP, MKL

Производительность!

“Citius, Altius, Fortius!”

Intel® C++



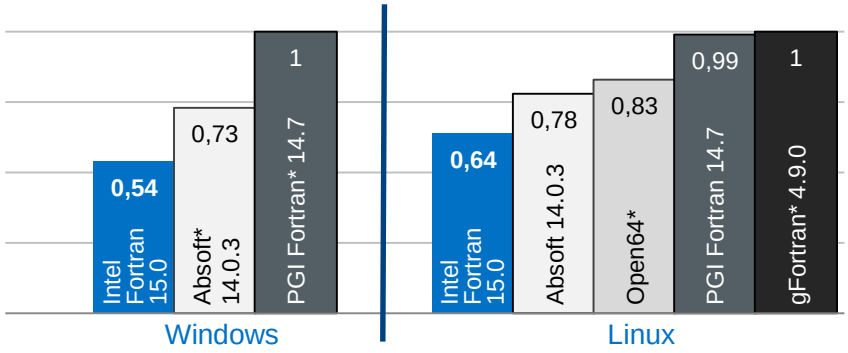
Relative geomean performance, SPEC* benchmark - higher is better

Configuration: Hardware: HP ProLiant DL360p Gen8 with Intel® Xeon® CPU E5-2680 v2 @ 2.80GHz, 256 GB RAM, HyperThreading is on. Software: Intel C++ compiler 15.0, Microsoft Visual C++ 2013, GCC 4.9.0. Linux OS: Red Hat Enterprise Linux Server release 6.5 (Santiago), kernel 2.6.32-431.el6.x86_64. Windows OS: Windows 7 Enterprise, Service pack 1. SPEC* Benchmark (www.spec.org).

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. * 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 .

Intel® Fortran Compiler



Relative geomean performance, Polyhedron* benchmark– lower is better

Configuration: Hardware: Intel® Core™ i7-4770K CPU @ 3.50GHz, HyperThreading is off, 16 GB RAM. . Software: Intel Fortran compiler 15.0, Absoft*14.0.3,. PGI Fortran* 14.7, Open64*, gFortran* 4.9.0. Linux OS: Red Hat Enterprise Linux Server release 6.4 (Santiago), kernel 2.6.32-358.el6.x86_64. Windows OS: Windows 7 Enterprise, Service pack 1. Polyhedron Fortran Benchmark (www.polyhedron.com). 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 -mfpmath=sse -flto -march=native -funroll-loops -ftree-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 -mso –apo.

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

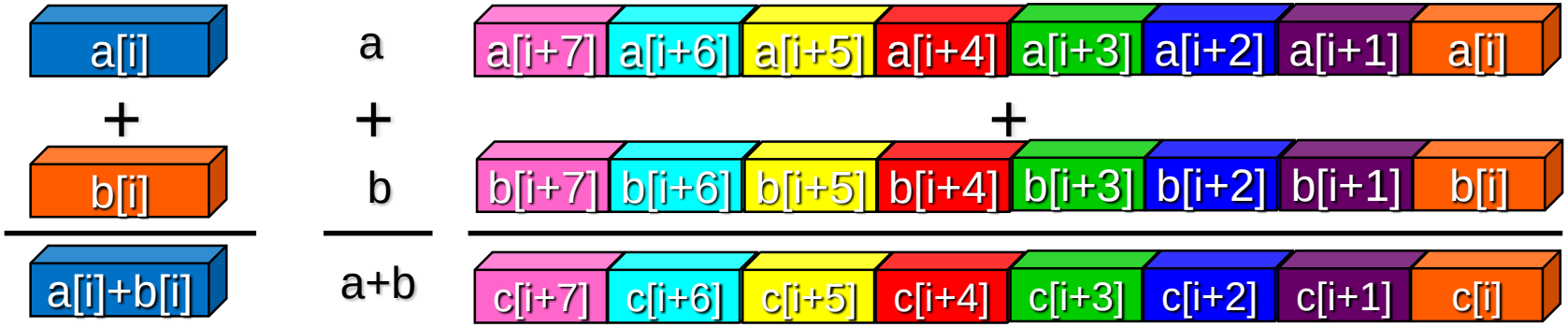
Optimization Notice: Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, 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 .

Векторизация

SIMD – одна инструкция для вектора данных

- Скалярное выполнение
  - x86/x87/SSE
  - одна инструкция дает один результат
- Векторное выполнение
  - SSE или AVX инструкции
  - одна инструкция дает вектор результатов

```
for (i=0;i<=MAX;i++)
    c[i]=a[i]+b[i];
```



Различные способы векторизации

Компилятор:
Авто-векторизация (без изменения кода)

Компилятор :
Директивы для векторизации

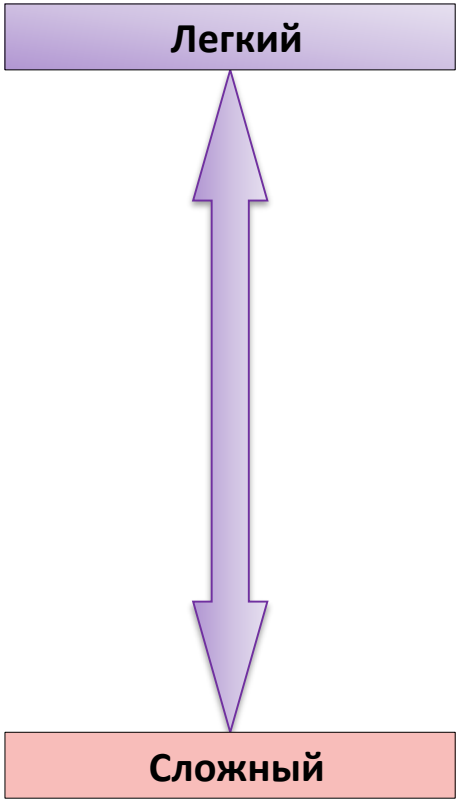
Компилятор:
#pragma simd / omp simd

Компилятор:
Intel® Cilk™ Plus Array Notation

Классы SIMD intrinsic
(например: `F32vec`, `F64vec`, …)

Векторные intrinsic функции
(например : `_mm_fmadd_pd(…)`, `_mm_add_ps(…)`, …)

Чистый ассемблер
(например : `[v]addps`, `[v]addss`, …)



Intel® C, C++ & Fortran Compilers

Векторизация с pragma simd

- Всего **одна строчка** (!) и максимальная производительность от SSE и AVX наша
- Игнорируются другими компиляторами => код портируем

```
for (int x = x0; x < x1; ++x) {
    float div = coef[0] * A_cur[x]
                + coef[1] * ((A_cur[x + 1] + A_cur[x - 1])
                + (A_cur[x + Nx] + A_cur[x - Nx])
                + (A_cur[x + Nxy] + A_cur[x - Nxy]))
                + coef[2] * ((A_cur[x + 2] + A_cur[x - 2])
                + (A_cur[x + sx2] + A_cur[x - sx2])
                + (A_cur[x + sxy2] + A_cur[x - sxy2]))
                + coef[3] * ((A_cur[x + 3] + A_cur[x - 3])
                + (A_cur[x + sx3] + A_cur[x - sx3])
                + (A_cur[x + sxy3] + A_cur[x - sxy3]))
                + coef[4] * ((A_cur[x + 4] + A_cur[x - 4])
                + (A_cur[x + sx4] + A_cur[x - sx4])
                + (A_cur[x + sxy4] + A_cur[x - sxy4]));
    A_next[x] = 2 * A_cur[x] - A_next[x] + vsq[s+x] * div;
}
```

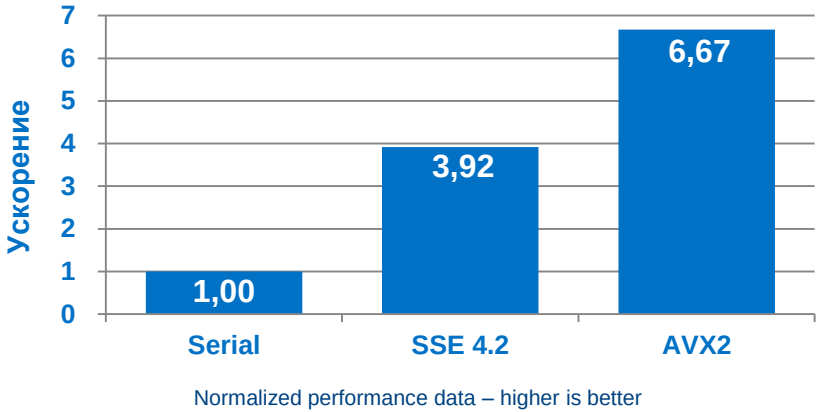
Векторизация с pragma simd

- Всего **одна строчка** (!) и максимальная производительность от SSE и AVX наша
- Игнорируются другими компиляторами => код портируем

```
#pragma simd vectorlength(8)
  for (int x = x0; x < x1; ++x) {
       float div = coef[0] * A_cur[x]
                        + coef[1] * ((A_cur[x + 1] + A_cur[x - 1])
                        + (A_cur[x + Nx] + A_cur[x - Nx])
                        + (A_cur[x + Nxy] + A_cur[x - Nxy]))
                        + coef[2] * ((A_cur[x + 2] + A_cur[x - 2])
                        + (A_cur[x + sx2] + A_cur[x - sx2])
                        + (A_cur[x + sxy2] + A_cur[x - sxy2]))
                        + coef[3] * ((A_cur[x + 3] + A_cur[x - 3])
                        + (A_cur[x + sx3] + A_cur[x - sx3])
                        + (A_cur[x + sxy3] + A_cur[x - sxy3]))
                        + coef[4] * ((A_cur[x + 4] + A_cur[x - 4])
                        + (A_cur[x + sx4] + A_cur[x - sx4])
                        + (A_cur[x + sxy4] + A_cur[x - sxy4]));
       A_next[x] = 2 * A_cur[x] - A_next[x] + vsq[s+x] * div;
  }
```

Intel® C, C++ & Fortran Compilers

RTM-Stencil



Normalized performance data – higher is better

Configuration: Intel® Xeon® CPU E3-1270 v3 @ 3.50 GHz system (4 cores with Hyper-Threading On), running at 3.50GHz, with 32.0GB RAM, L1 Cache 256KB, L2 Cache 1.0MB, L3 Cache 8.0MB, 64-bit Windows* Server 2012 R2 Datacenter. Compiler options:, SSE4.2: -O3 –Qipo –QxSSE4.2 or AVX2: -O3 –Qipo –QxCORE-AVX2. For more information go to http://www.intel.com/performance

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. * 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 .

Векторизация с OpenMP 4.0

- Всего **две строчки** (!) и максимальная производительность от SSE и AVX наша

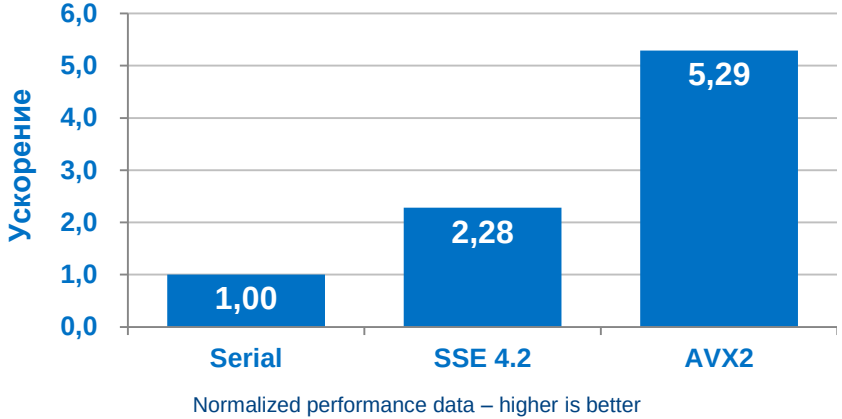

```
typedef float complex fcomplex;
const uint32_t max_iter = 3000;

uint32_t mandel(fcomplex c, uint32_t max_iter)
{
    uint32_t count = 1; fcomplex z = c;
    while ((cabsf(z) < 2.0f) && (count < max_iter)) {
       z = z * z + c; count++;
    }
    return count;
}
uint32_t count[ImageWidth][ImageHeight];
 …… ….  …….
      for (int32_t y = 0; y < ImageHeight; ++y) {
        float c_im = max_imag - y * imag_factor;

        for (int32_t x = 0; x < ImageWidth; ++x) {
          fcomplex in_vals_tmp = (min_real + x *
real_factor) + (c_im * 1.0iF);
          count[y][x] = mandel(in_vals_tmp, max_iter);
        }
      }
```

Векторизация с OpenMP 4.0

- Всего **две строчки** (!) и максимальная производительность от SSE и AVX наша

```
typedef float complex fcomplex;
const uint32_t max_iter = 3000;
#pragma omp declare simd uniform(max_iter), simdlen(16)
uint32_t mandel(fcomplex c, uint32_t max_iter)
{
    uint32_t count = 1; fcomplex z = c;
    while ((cabsf(z) < 2.0f) && (count < max_iter)) {
       z = z * z + c; count++;
    }
    return count;
}
uint32_t count[ImageWidth][ImageHeight];
 …… …. ……
      for (int32_t y = 0; y < ImageHeight; ++y) {
        float c_im = max_imag - y * imag_factor;
#pragma omp simd safelen(16)
        for (int32_t x = 0; x < ImageWidth; ++x) {
          fcomplex in_vals_tmp = (min_real + x *
real_factor) + (c_im * 1.0iF);
          count[y][x] = mandel(in_vals_tmp, max_iter);
        }
      }
```

Вычисление множества Мандельброта



Normalized performance data – higher is better

Configuration: Intel® Xeon® CPU E3-1270 v3 @ 3.50 GHz system (4 cores with Hyper-Threading On), running at 3.50GHz, with 32.0GB RAM, L1 Cache 256KB, L2 Cache 1.0MB, L3 Cache 8.0MB, 64-bit Windows* Server 2012 R2 Datacenter. Compiler options:, SSE4.2: -O3 –Qipo –QxSSE4.2 or AVX2: -O3 –Qipo –QxCORE-AVX2.

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

Ключевые слова Intel Cilk™ Plus

ДО: последовательный код

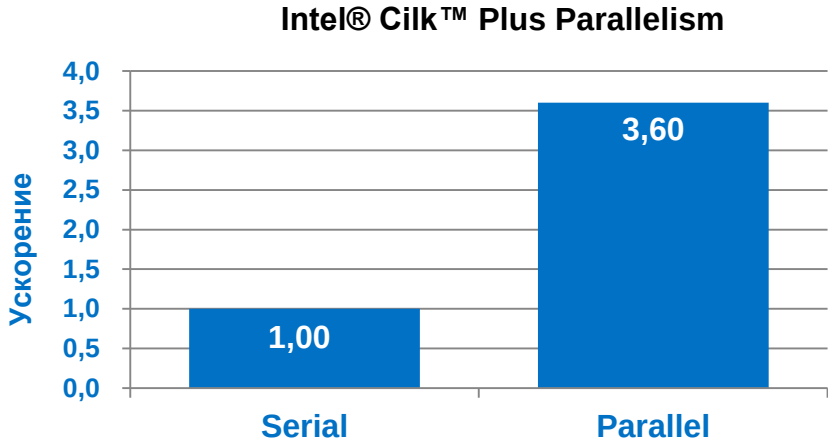
```
void Search( const Board& b ) {
  …
    } else {
     Search( Board(b,i,0) );
     Search( Board(b,i,1) );
    }}}
```

30.2 Seconds

ПОСЛЕ: параллельный код

```
void Search( const Board& b ) {
  …
    } else {
     cilk_spawn Search( Board(b,i,0)
);
     Search( Board(b,i,1) );
    }}}
```

8.4 Seconds



Normalized performance data – higher is better

Configuration: Intel® Core™ i7-4770K (Haswell) CPU, 3.50GHz, 4 cores, Hyper-Threading On, 16GB RAM, 12M cache, 64-bit Ubuntu. For the full code set: https://software.intel.com/en-us/articles/cilk-plus-solver-for-a-chess-puzzle-or-how-i-learned-to-love-rejection

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

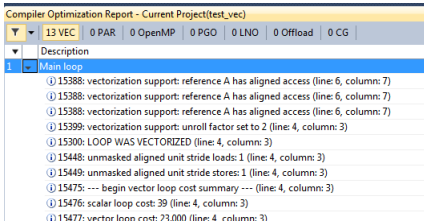
Intel Parallel Studio XE 2015 Composer Edition

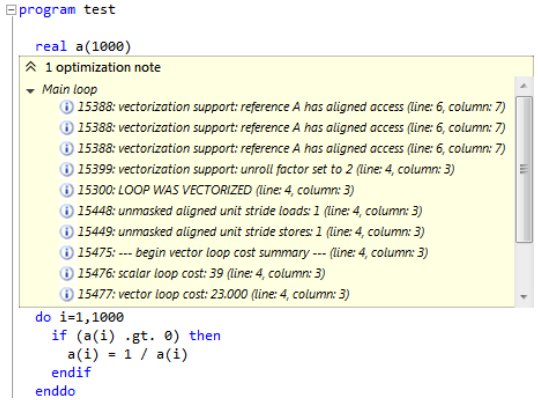
что нового?

- Полная поддержка стандартов:
  - C++11
  - Fortran 2003
- Почти весь OpenMP* 4.0 (нет редукций, определяемых пользователем)
- Частичная поддержка Fortran 2008
- Оффлодинг с помощью Intel® Graphics Technology
- Новые отчеты об оптимизации и векторизации
- Поддержка Fortran в gdb*
- Удобная онлайн установка
- Ещё более совершенная библиотека MKL

Отчёты по оптимизации

- Единая опция для всех отчетов -opt-report (объединение -vec-report, -par-report, -openmp-report и др.)
- Старые опции всё ещё поддерживаются
- Файл с отчётом для каждого объектника.
  - Можно изменить с помощью –opt-report-file=<filename|stderr|stdout>
- Более понятные и читаемые отчеты
- Красивая интеграция с Microsoft Visual Studio*





Отчёт о векторизации

```
int size();
 void foo(double *restrict a,
double *b){
  int i;
  for (i=0;i<size();i++){
    a[i] += b[i];
  }
}
```

icpc -c -O3 -restrict -opt-report x.cpp

14.0 compiler:

x.cpp(6) (col. 15) remark: loop was not vectorized: unsupported loop structure

Отчёт о векторизации

```
int size();
 void foo(double *restrict a,
double *b){
  int i;
  for (i=0;i<size();i++){
    a[i] += b[i];
  }
}
```

```
icpc -c  -O3 -restrict -opt-report  x.cpp

15.0 compiler:
LOOP BEGIN at x.cpp(6,15)
   remark #15523: loop was not vectorized: cannot compute loop iteration count before executing the loop.
LOOP END
```

Intel VTune Amplifier XE 2015
что нового?

Больше профилировки на CPU и GPU

- Поддержка OpenCL на Windows
- Анализ TSX транзакций
- GPU оффлодинг на Windows
- Анализ масштабируемости OpenMP

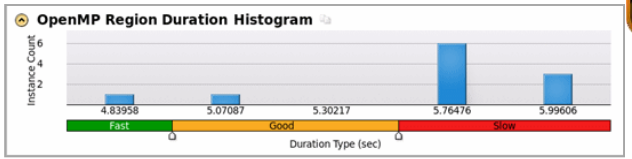
Мощный анализ данных

- Импорт внешних данных из CSV и поддержка внешних коллекторов
- Группировка во временной шкале

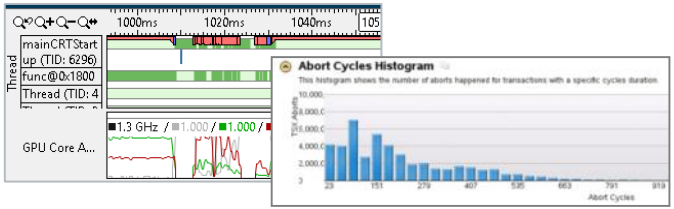
Удобство использования

- Анализ удалённых Linux* и Android систем
- Просмотрщик для Mac OS X*

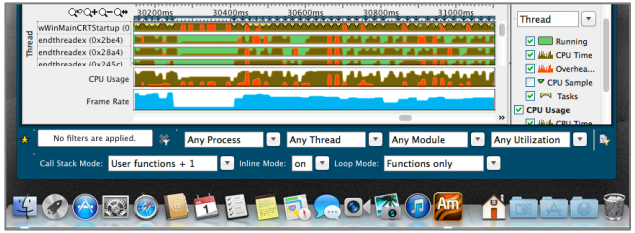
Анализ масштабируемости OpenMP



GPU offload & TSX Sync.

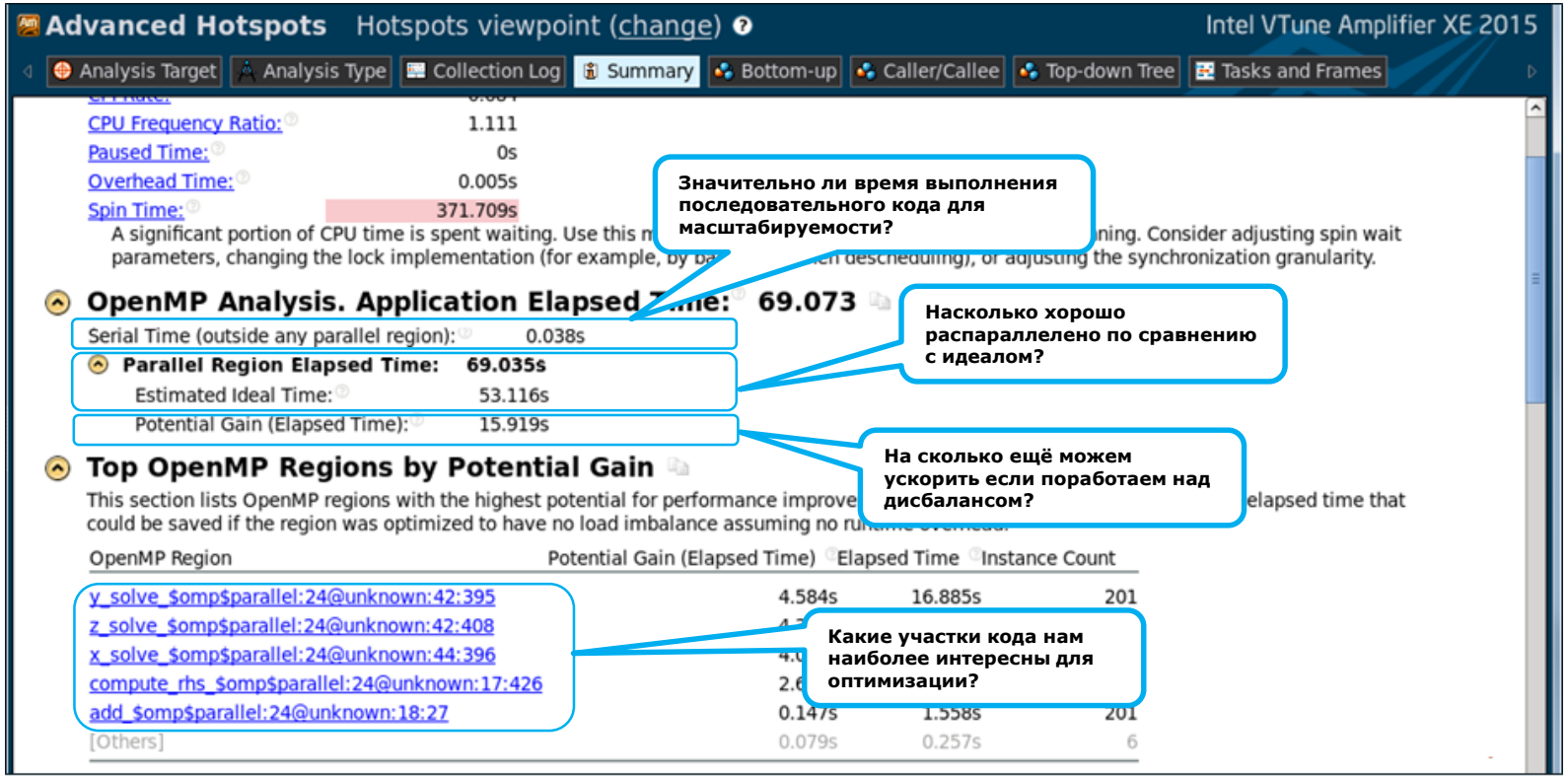


Удаленный анализ на Mac OS X*



Анализ масштабируемости OpenMP*

Intel® VTune™ Amplifier XE

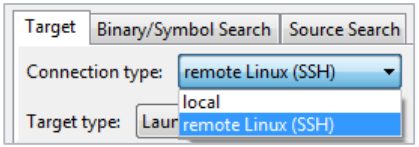


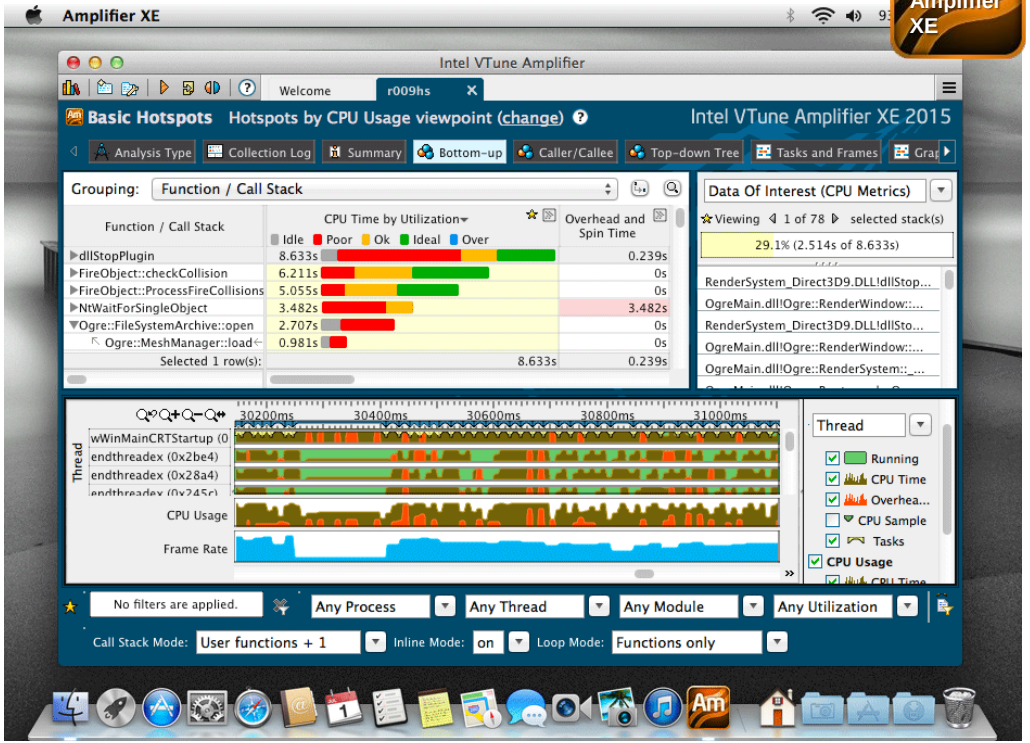
Просмотр на OS X*
Intel® VTune™ Amplifier XE

Поддержка Mac OS X
- Анализ профилей с Linux*
- Анализ профилей с Windows*
- Нет профилировки на OS X

Почти “даром”
- Отдельный установочный пакет
- Использует лицензию Windows или Linux

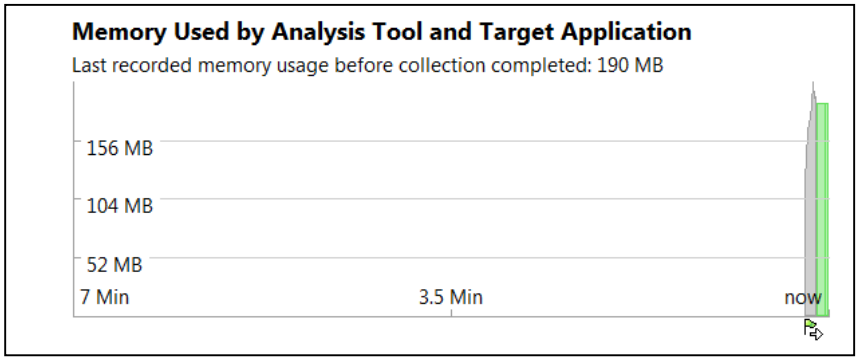
Удаленная профилировка
- По SSH к Linux



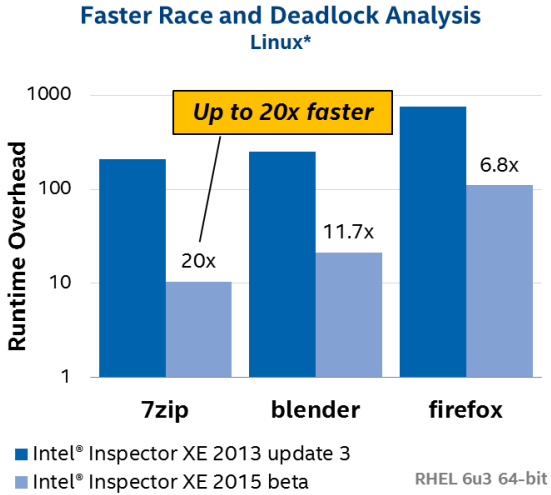


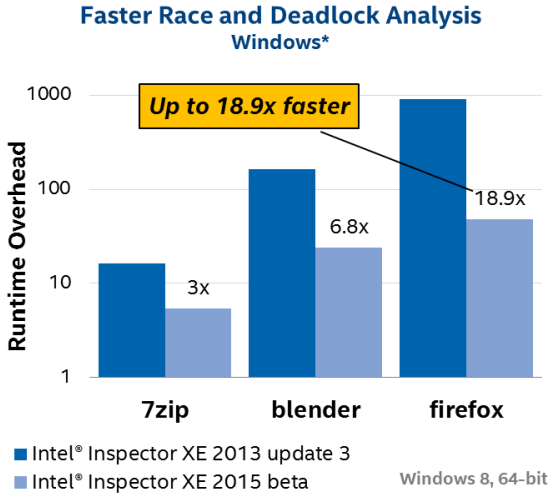
Intel Inspector XE 2015
что нового?

Граф использования памяти в реальном времени



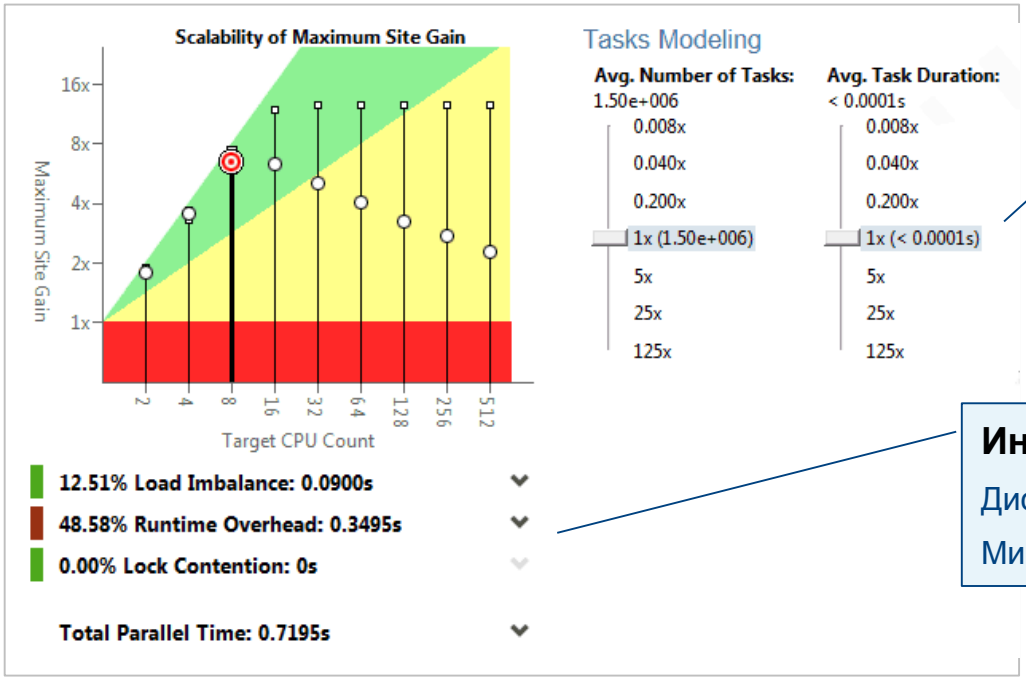
Проверка потоков ускорена до 20 раз – использование памяти тоже сократилось!





View configuration information at end of this presentation: Click to view

Intel Advisor XE 2015
что нового?



Моделирование размера входных данных

Количество итераций

Размер задачи

Оценка большой нагрузки на маленьких тестах

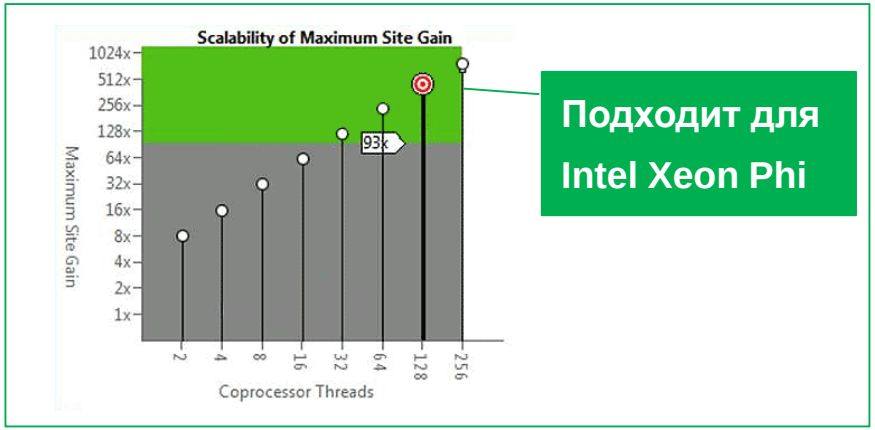
Информация о задачах

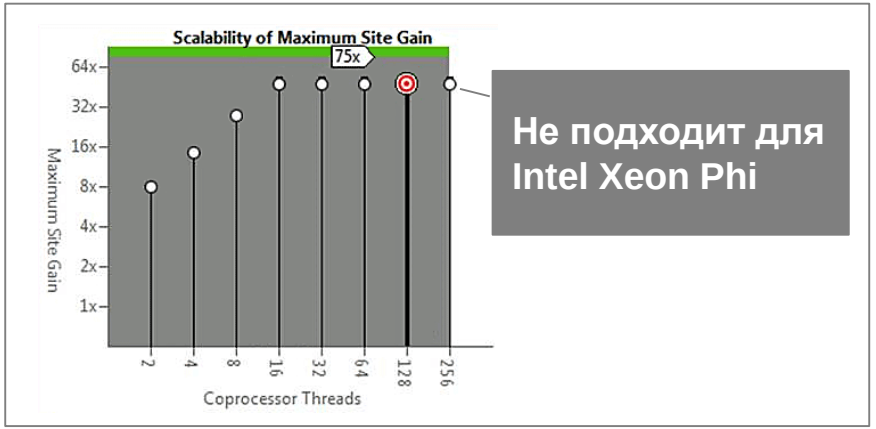
Дисбаланс

Минимальное и максимальное время

Intel Advisor XE 2015
что нового?

Моделирование для Xeon и Xeon Phi!





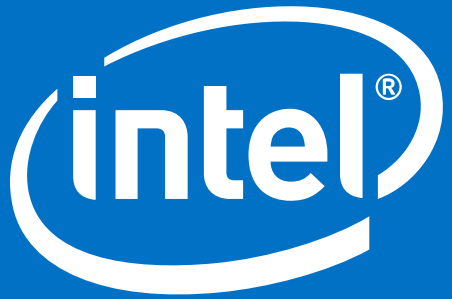
Итого

- Параллелизм – ключ к высокой производительности!
  - Средства Intel позволяют проектировать и эффективно реализовать параллелизм
- Новые версии – новые возможности!
  - code ***faster*** code

Попробуйте – вам понравится!

https://software.intel.com/ru-ru/intel-software-evaluation-center

Q&A



Software