

# Intel Advisor и Roofline Анализ

Ануфриенко Андрей

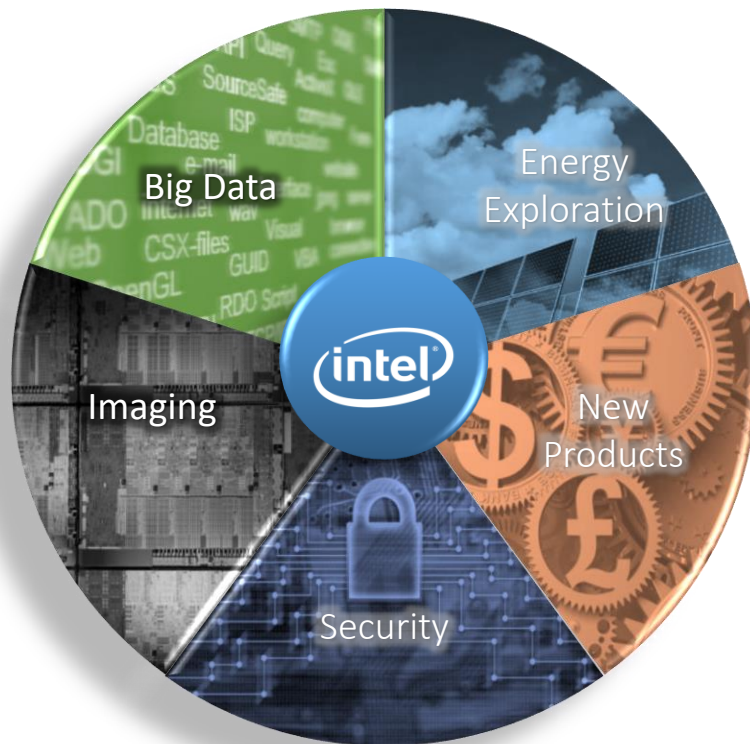
"Intel® Software 2017: HPC Code  
Modernization and Artificial Intelligence"

15 сентября 2017



# HPC :

## Для того чтобы конкурировать нужно вычислять

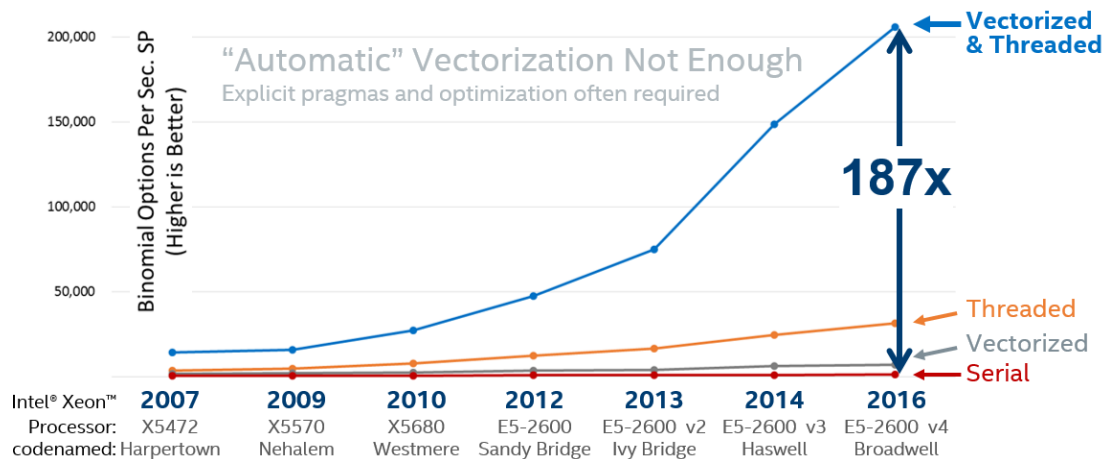


# Последовательные приложения



# Intel® Advisor – Модернизируйте ваш код

## Векторизация и прототипирование многопоточности



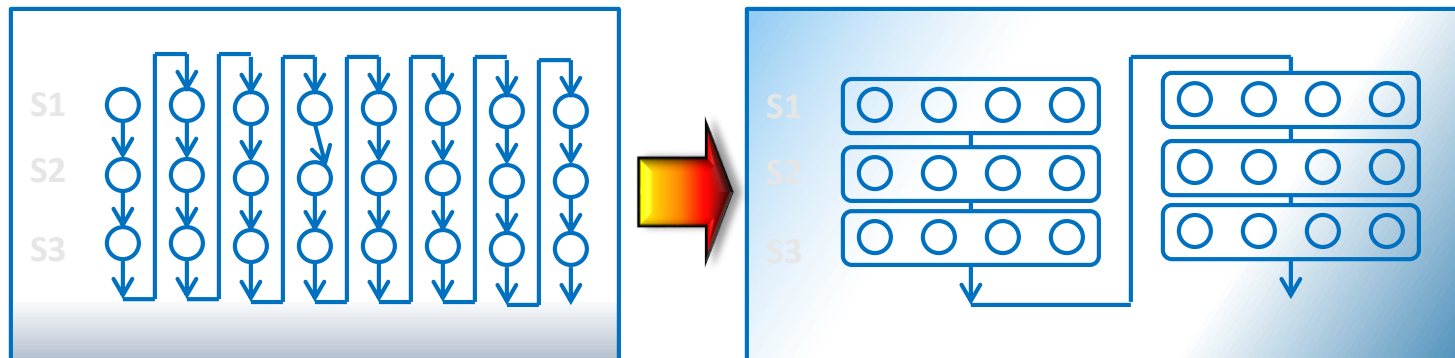
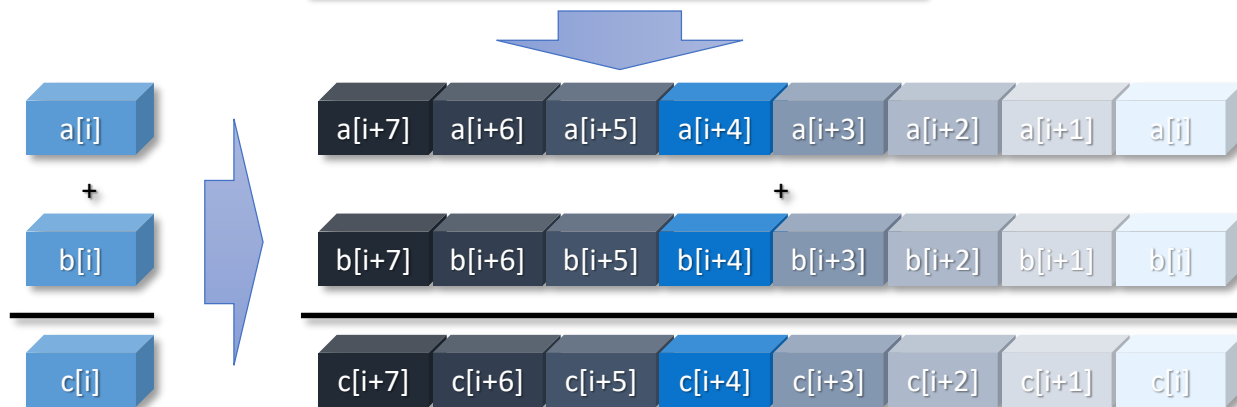
- Векторизуйте и распараллеливайте для максимальной производительности приложений на современных процессорах
- Получайте информацию о числе итераций, зависимостях по данным, особенностях доступа к памяти
- Следуйте простому рабочему процессу оптимизации с подсказками для получения более быстрого кода

**Производительность увеличивается с каждым новым поколением железа**

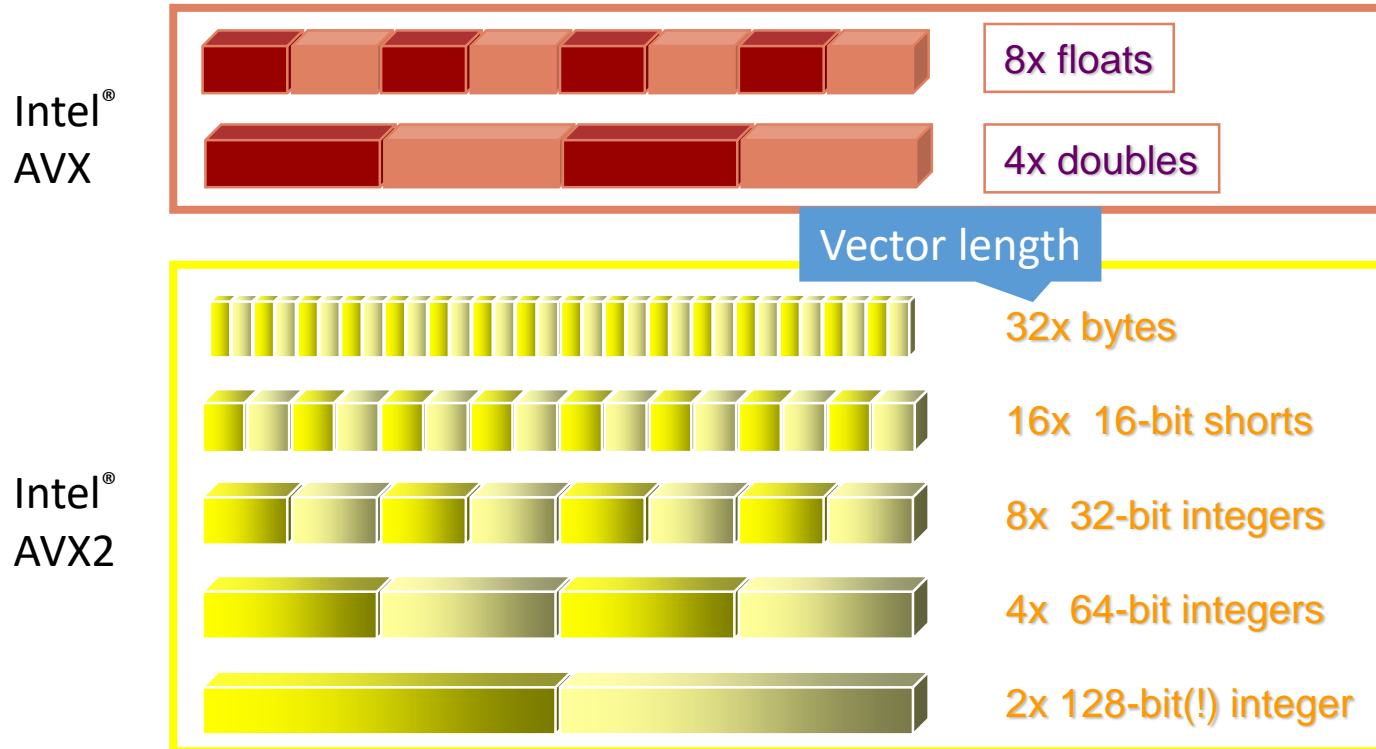
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to <http://www.intel.com/performance> [Configurations](#) at the end of this presentation.

# Векторные операции (SIMD)

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



# Intel® Advanced Vector Extensions (Intel® AVX)

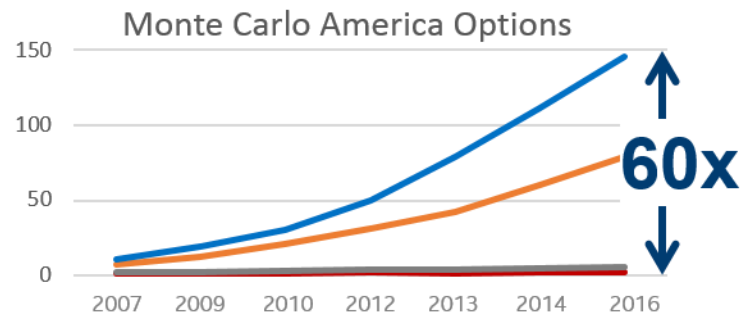
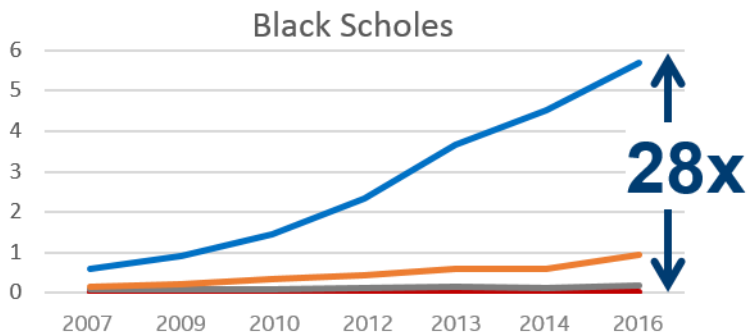
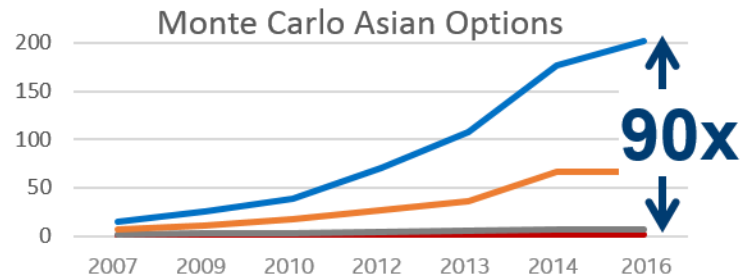
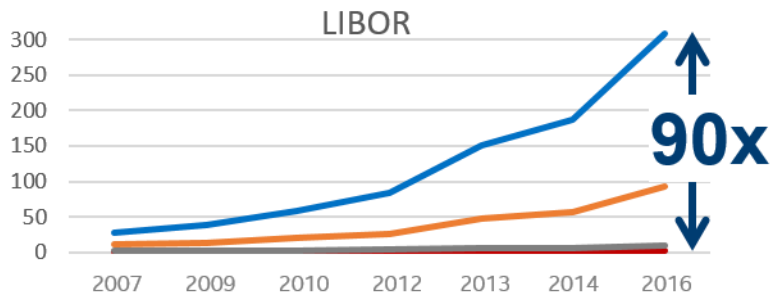


# Используйте весь параллелизм



| Intel Advisor XE: | Threading | Vectorization |
|-------------------|-----------|---------------|
|                   | ✓         | ✓             |

# Векторизация и многопоточность критична для современных вычислительных систем



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to <http://www.intel.com/performance> [Configurations](#) at the end of this presentation.



# Автоматической векторизации часто недостаточно

- Компилятор не всегда может векторизовать ваш код
  - Проверьте наличие цикловых зависимостей используя [Intel® Advisor](#)
  - Все понятно? Векторизуйте принудительно.  
C++: `pragma simd`,  
Fortran use: `SIMD directive`
- Не всякая векторизация одинаково эффективна
  - последовательный доступ эффективнее чем доступ с шагом. Анализируйте с [Intel® Advisor](#).
  - Рассмотрите возможность изменения структуры данных. [Intel® SIMD Data Layout Templates](#) могут помочь

Бенчмарки на предыдущих слайдах были ускорены не только за счет автовекторизации. Были использованы директивы для принудительной векторизации и т.п.

Массивы структур удобны для интуитивной организации данных, но значительно более эффективной для векторизации является организация данных как структура массивов. Использование [Intel® SIMD Data Layout Templates](#) (Intel® SDLT) позволяет маппировать данные в более эффективные структуры для векторизации.

# Быстрый код быстрее с дизайном управляемым данными

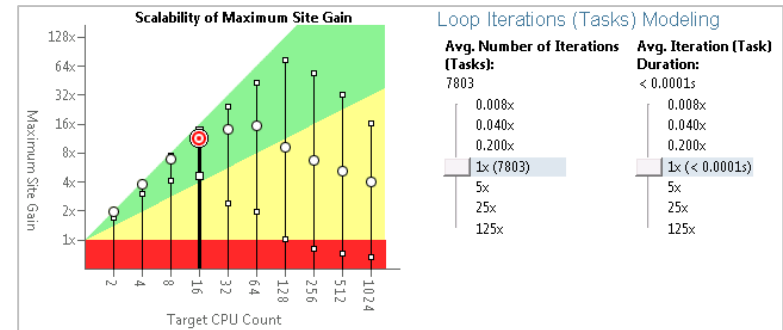
Intel® Advisor – векторизация и прототипирование многопоточности

- Наилейффективнейшая параллелизация:
  - Векторизуем там, где это приносит наибольшую выгоду
  - Быстрое определение блокирующих проблем
  - Подсказки для эффективной векторизации
  - Безопасная векторизация
  - Оптимизация доступа к памяти



| Function Call Sites and Loops        | Vector Issues | Self Time | Type             | FLOPS  | GFLOPS | AI | Why No Vectorization? | Vectorized Loops | Gain... |
|--------------------------------------|---------------|-----------|------------------|--------|--------|----|-----------------------|------------------|---------|
| [loop in S252 at loops90.f:1172]     |               | 3.158s    | Vectorized ...   | 0.167  | 0.1070 |    | 1 vectori...          | AVX2             | 1.7%    |
| [loop in S2101 at loops90.f:1749]    | 2 Ineffi...   | 2.875s    | Scalar           | 0.1361 | 0.0625 |    | vectorizat...         |                  |         |
| [loop in S126 at loops90.f:447]      | 2 Prov...     | 0.997s    | Scalar           | 0.3971 | 0.1667 |    | vector de...          |                  |         |
| [loop in S343 at loops90.f:2300]     | 2 Assu...     | 0.875s    | Scalar           |        |        |    | vector de...          |                  |         |
| [loop in s141_SompSparallel_for ...] | 2 Assu...     | 0.824s    | Scalar           | 0.0611 | 0.0833 |    | vector de...          |                  |         |
| [loop in S353 at loops90.f:2381]     | 1 Possi...    | 0.719s    | Vectorized (...) | 2.771  | 0.1250 |    |                       | AVX2             | 3.5%    |
| [loop in s232_SompSparallel_for ...] | 3 Prov...     | 0.693s    | Scalar Versions  | 0.2881 | 0.2220 |    | 1 vector d...         |                  |         |

- Прорыв для многопоточного дизайна:
  - Быстрое прототипирование
  - Масштабирование проекта на больших системах
  - Поиск ошибок синхронизации до имплементации многопоточности
  - Проектирование без утомительной разработки



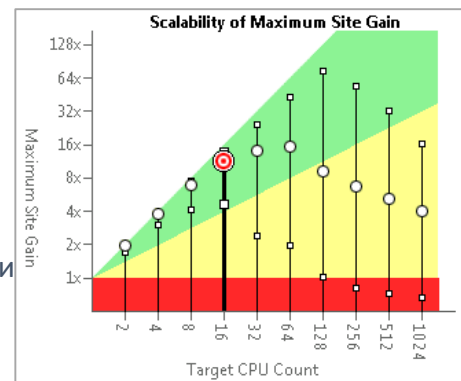
Добавляйте параллелизм с наименьшими усилиями и риском и наибольшей отдачей

Part of Intel® Parallel Studio  
<http://intel.ly/advisor-xe>

# Сделай быстрый код быстрее! Intel® Advisor

Прототипирование многопоточности

- Случалось ли у вас, что:
  - Параллелизация приложения дает скромный эффект?
  - Достигли «барьера масштабируемости»?
  - Отложили релиз из-за ошибок синхронизации?
- Параллелизация управляемая данными:
  - Быстрое прототипирование
  - Масштабирование проекта на больших системах
  - Поиск ошибок синхронизации до имплементации многопоточности
  - Проектирование без утомительной разработки



**Добавляйте параллелизм с наименьшими усилиями и риском и наибольшей отдачей**

“Intel® Advisor has allowed us to quickly prototype ideas for parallelism, saving developer time and effort”

*Simon Hammond*  
Senior Technical Staff  
Sandia National Laboratories

<http://intel.ly/advisor-xe>

# Спроектируй и затем реализуй

Intel® Advisor прототипирование многопоточности



# Добавь параллельную реализацию

Summary of predicted parallel behavior | Intel Advisor XE 2015

Summary | Survey Report | Annotation Report | Suitability Report | Correctness Report

Maximum program gain<sup>®</sup>: 5.20x (8 CPUs, Intel TBB Threading Model)

These annotated parallel sites were detected:

| Parallel Site                                     | Maximum Site Gain <sup>®</sup> | Correctness Problems |
|---|--------------------------------|----------------------|
| <a href="#">solve (nqueens_annotated.cpp:113)</a> | <a href="#">6.51x</a>          | 0 2 1 0              |

Consider adding parallel site and task annotations around these time-consuming loops found during Survey an

| Loop                     | Source Location                           | CPU Total Time <sup>®</sup> |
|--------------------------|---|-----------------------------|
| <a href="#">setQueen</a> | <a href="#">nqueens_annotated.cpp:96</a>  | 1.8252s                     |
| <a href="#">solve</a>    | <a href="#">nqueens_annotated.cpp:117</a> | 1.8252s                     |
| <a href="#">setQueen</a> | <a href="#">nqueens_annotated.cpp:69</a>  | 0.1976s                     |
| <a href="#">setQueen</a> | <a href="#">nqueens_annotated.cpp:69</a>  | 0.1877s                     |
| <a href="#">setQueen</a> | <a href="#">nqueens_annotated.cpp:69</a>  | 0.1346s                     |

- Список позиций в коде
- Шаблоны для популярных методов реализации многопоточности

## Intel® Advisor

- Содержит основные метрики для популярных параллельных реализаций
- Быстрое прототипирование и оценка альтернатив
- Детальные подсказки для популярных многопоточных реализаций

| Serial Code with Intel Advisor Annotations  | Parallel Code using Intel TBB  |
|---|--|
| <pre>// Locking ANNOTATE_LOCK_ACQUIRE(); Body(); ANNOTATE_LOCK_RELEASE();</pre>   | <pre>// Locking can use various mutex types provided // by Intel TBB. For example: #include &lt;tbb/tbb.h&gt; ... tbb::mutex g_Mutex; ... {   tbb::mutex::scoped_lock lock(g_Mutex);   Body(); }</pre> |
| <pre>// Do-All Counted loops, one task ANNOTATE_SITE_BEGIN(site); For (I = 0; I &lt; N; ++I) {   ANNOTATE_ITERATION_TASK(task);   {statement;} } ANNOTATE_SITE_END();</pre>   | <pre>// Do-All Counted loops, using lambda // expressions #include &lt;tbb/tbb.h&gt; ... tbb::parallel_for(0, N, [&amp;](int I) {   statement; });</pre>   |
| <pre>// Create Multiple Tasks ANNOTATE_SITE_BEGIN(site); ANNOTATE_TASK_BEGIN(task1); statement-or-task1; ANNOTATE_TASK_END(); ANNOTATE_TASK_BEGIN(task2); statement-or-task2; ANNOTATE_TASK_END(); ANNOTATE_SITE_END();</pre> | <pre>// Create Multiple tasks, using lambda // expressions #include &lt;tbb/tbb.h&gt; ... tbb::parallel_invoke(   [&amp;]{statement-or-task1;},   [&amp;]{statement-or-task2;});</pre>                 |

Threading Model:

- Intel TBB
- Other
- Intel TBB
- Intel Cilk Plus
- OpenMP
- Microsoft TPL

# Vectorization Advisor

Intel® Advisor – Векторизация и прототипирование многопоточности

- **Наиэффективнейшая параллелизация:**
  - Векторизуем там, где это приносит наибольшую выгоду
  - Быстрое определение блокирующих проблем
  - Подсказки для эффективной векторизации
  - Безопасная векторизация
  - Оптимизация доступа к памяти
- **Данные и необходимая помощь:**
  - Диагностика компилятора + данные о производительности + SIMD эффективность
  - Определить проблемы и рекомендовать их решения
  - Анализ цикловых зависимостей
  - Анализ методов доступа к памяти

| Function Call Sites and Loops            | Vector Issues | Self Time          | Total Time | Type             | FLOPS  |        | Why No Vectorization? | Vectorized Loops |            |         |        | Trip Counts |
|--|---------------|--------------------|------------|------------------|--------|--------|-----------------------|------------------|------------|---------|--------|-------------|
|  |               |                    |            |                  | GFLOPS | AI     |                       | Vector...        | Efficiency | Gain... | VL ... |             |
| [loop in S343 at loops90.f:2300]         | 1 Assu...     | 0.875s <b>2.5%</b> | 0.875s     | Scalar           |        |        | vector de...          |                  |            |         |        |             |
| [loop in s141_SompSparallel_for@569 ...] | 1 Assu...     | 0.824s <b>2.4%</b> | 0.824s     | Scalar           | 0.061  | 0.0833 | vector de...          |                  |            |         |        | 500         |
| [loop in S353 at loops90.f:2381]         | 1 Possi...    | 0.719s             | 0.719s     | Vectorized ...   | 2.771  | 0.1250 |                       | AVX2             | 35%        | 2.78x   | 8      | 62; 4       |
| [loop in s232_SompSparallel_for@955 ...] | 2 Prove ...   | 0.693s             | 0.693s     | Scalar Versio... | 0.288  | 0.2220 | 1 vector d...         |                  |            |         |        | 249; 3      |
| [loop in S222 at loops90.f:907]          | 2 Prove ...   | 0.672s             | 0.672s     | Scalar           | 1.775  | 0.3000 | vector de ...         |                  |            |         |        | 999         |
| [loop in S352 at loops90.f:2356]         | 2 Possib ...  | 0.656s             | 0.656s     | Scalar           | 3.048  | 0.2500 | vectorizat ...        |                  |            |         |        | 200         |
| [loop in s114_SompSparallel_for@211 ...] | 2 Ineffe ...  | 0.632s             | 0.632s     | Vectorized (...) | 0.156  | 0.0769 |                       | AVX              | 24%        | 1.94x   | 8      | 62; 3       |

**Оптимизация для AVX-512 с и без доступа к AVX-512 железу**

<http://intel.ly/advisor-xe>

# Правильные данные на кончиках ваших пальцев

Получите всю информацию необходимую для эффективной векторизации

The screenshot displays the Intel Advisor XE 2016 interface with a table of vectorization results. The table columns include Function Call Sites and Loops, Vector Issues, Self Time, Total Time, Trip Counts, Loop Type, Why No Vectorization?, and Vectorized Loops (with sub-columns for Vecto..., Efficiency, and Vector L...). Callouts highlight specific features: 'Фильтр для векторизованных циклов!' points to the 'Vectorized' filter; 'Кол-во итераций' points to the 'Trip Counts' column; 'Что препятствует векторизации?' points to the 'Why No Vectorization?' column; 'Фокус на горячих циклах' points to the 'Function Call Sites and Loops' column; 'Имеющиеся векторизационные проблемы.' points to the 'Vector Issues' column; 'Используемые векторные инструкции.' points to the 'Vectorized Loops' column; and 'Эффективность кода.' points to the 'Efficiency' column.

| Function Call Sites and Loops            | Vector Issues                | Self Time | Total Time | Trip Counts | Loop Type         | Why No Vectorization?         | Vectorized Loops |            |             |
|--|------------------------------|-----------|------------|-------------|-------------------|-------------------------------|------------------|------------|-------------|
|  |                              |           |            |             |                   |                               | Vecto...         | 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     | 12          | 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              | ~96%       | 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       |                  |            |             |

**Сделай быстрый код быстрее!**

# Сделай быстрый код быстрее! Intel® Advisor

## Векторизация



### • Случалось у вас, что:

- Перекомпилировали для AVX2 с маленьким улучшением?
- Не понимали где векторизовать?
- Переписывали интринсики для новых архитектур?
- Мучались с компиляторными отчетами?

- Векторизация управляемая данными:
  - Что векторизовать в первую очередь?
  - Что и почему мешает векторизации?
  - Мои циклы векторизовались успешно?
  - Можно реорганизовать данные для улучшения производительности?
  - Безопасно ли использование прагмы simd?

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

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

| Function Call Sites and Loop  | Vector Issues           | Self Time | Total Time | Trip Counts | Loop Type     | Why No Vectorization?  | Vectorized Loops |            |
|-------------------------------|-------------------------|-----------|------------|-------------|---------------|------------------------|------------------|------------|
|                               |                         |           |            |             |               |                        | Vecto...         | Efficiency |
| [loop at stl_algo.h:4740 i... |                         | 0.170s    | 0.170s     |             | Scalar        | non-vectorizable I ... |                  |            |
| [loop at loopstl.cpp:2449...  | 2 Ineffective peeled... | 0.170s    | 0.170s     | 12; 4       | Collapse      | Collapse               | AVX              | ~100%      |
| [loop at loopstl.cpp:2...     |                         | 0.150s    | 0.150s     | 12          | Vectorized (B |                        | AVX              |            |
| [loop at loopstl.cpp:2...     |                         | 0.020s    | 0.020s     | 4           | Remainder     |                        |                  |            |
| [loop at loopstl.cpp:7900...  |                         | 0.170s    | 0.170s     | 500         | Scalar        | vectorization possi... |                  |            |
| [loop at loopstl.cpp:35...    | 1 High vector regi...   | 0.160s    | 0.160s     | 12          | Expand        | Expand                 | AVX              | ~69%       |

"Intel® Advisor's Vectorization Advisor permitted me to focus my work where it really mattered. When you have only a limited amount of time to spend on optimization, it is invaluable."

*Gilles Civario*  
Senior Software Architect  
Irish Centre for High-End Computing



# 5 шагов к эффективной векторизации - Vector Advisor

### 1. Диагностика компилятора + Данные о производительности + Информация об эффективности векторизации

| 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>::... | 0.031s    | 0.031s     | Vectorized (Body)                              |

Vectorized SSE: SSE2 loop processing Float32; Float64 data type  
Peeled loop: loop stars were reordered

| Function Call Sites and Loops   | Self Time | Total Time |
|---|-----------|------------|
| [loop in std::basic_string<char,struct_std::char_traits<char>,class_std::alloc... | 0.000s    | 5...       |
| [loop in std::basic_string<char,struct_std::char_traits<char>,class_std::allo...  | 0.000s    | 5...       |
| [loop in std::num_put<char,class_std::ostreambuf_iterator<char,struct_st...       | 0.000s    | ...        |

### 2. Руководство: определить проблемы и дать рекомендации для их исправления

**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

Use one of the memory accesses in the source loop does not align with the byte boundary of the kernel loop. Tell the compiler your memory access is aligned.

SIZE\*sizeof(float), 32);

### 3. Анализ кол-ва итераций + FLOPs: оценить эффективность векторизации

| Total Time | Median | Min | Max | Iteration Duration | Call Count |
|------------|--------|-----|-----|--------------------|------------|
| 3.151s     | 1      | 1   | 1   | 3.1509s            | 1          |
| 0.440s     | 1      | 1   | 1   | < 0.0001s          | 2408000    |
| 0.010s     | 1      | 1   | 2   | < 0.0001s          | 207596     |
| 0.010s     | 1      | 2   | 9   | < 0.0001s          | 1173619    |
| 0.010s     | 1      | 3   | 5   | < 0.0001s          | 1312315    |

### 4. Анализ цикловых зависимостей

| 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           |

### 5. Анализ типа доступа к памяти

| Site Name     | Site Function | Site Info             | Loop-Carried Dependencies | Strides Distribution     | Access Pattern           |
|---------------|---------------|-----------------------|---------------------------|--------------------------|--------------------------|
| loop_site_203 | runCrawLoops  | runCrawLoops.cxx:1063 | RAW:1                     | No information available | No information available |
| loop_site_139 | runCrawLoops  | runCrawLoops.cxx:622  | No information available  | 39% / 36% / 25%          | Mixed strides            |
| loop_site_160 | runCrawLoops  | runCrawLoops.cxx:925  | No information available  | 100% / 0% / 0%           | All unit strides         |

| ID  | Stride  | Type            | Source               | Modules  | Alignment |
|-----|---|-----------------|----------------------|----------|-----------|
| P22 | 0; 0; 1   | Unit stride     | runCrawLoops.cxx:637 | lcal.exe |           |
| P23 | 0; 0  | Unit stride     | runCrawLoops.cxx:638 | lcal.exe |           |
| P30 | -1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801 | Variable stride | runCrawLoops.cxx:628 | lcal.exe |           |

```

635     j2 = ( j2 & 64-1 ) ;
636     p[ip][0] += y[i2+32];
637     p[ip][1] += z[j2+32];
638     i2 += e[i2+32];
639     j2 += f[j2+32];
    
```

```

626     i1 = 64-1;
627     j1 = 64-1;
628     p[ip][2] += b[j1][11];
    
```

# Дополнительные материалы

- Intel® Advisor:

- [Страница продукта](#) – обзоры, новинки, вопросы, поддержка ...
- [Тренинги](#) – фильмы, технические обзоры, документация...
- [Руководства](#) – шаг за шагом через тернии к звездам
- [Обзоры](#)

- Другие инструменты для анализа:

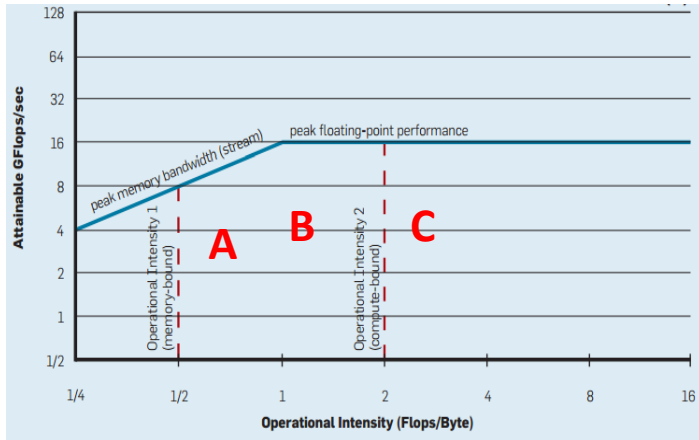
- [Intel® VTune Amplifier](#) – профилировщик производительности
- [Intel® Inspector](#) – исследователь потоков и памяти / отладчик

- Другие продукты для разработчиков:

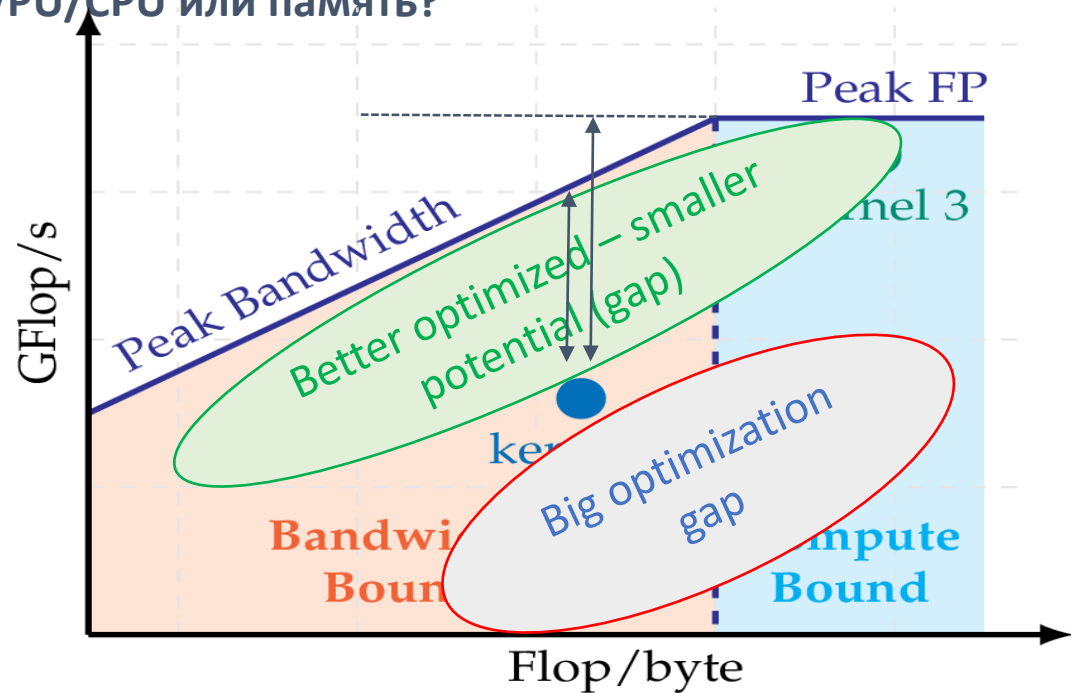
- [Intel® Software Development Products](#)

# Roofline модель: Что ограничивает производительность?

VPU/CPU или память?



Что делает циклы  
**A, B, C** различными?



# Roofline Модель

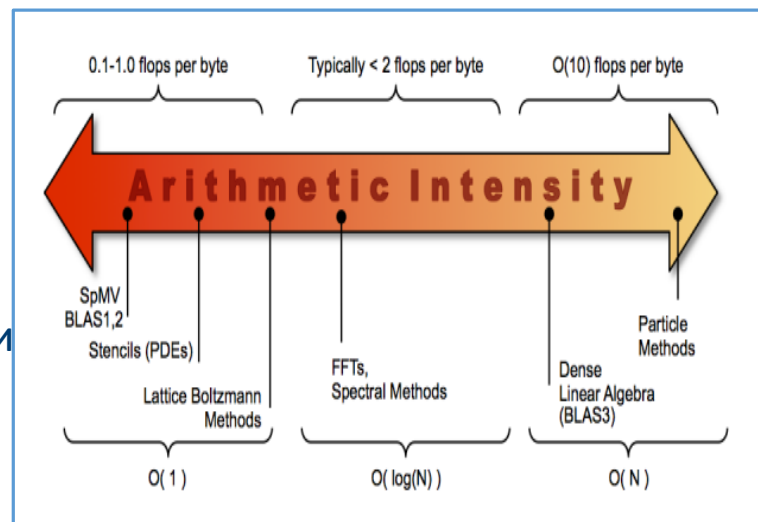
## ВИЗУАЛЬНО ИНТУИТИВНАЯ МОДЕЛЬ ПРОИЗВОДИТЕЛЬНОСТИ

Берет в расчет

- Использование/спрос памяти
- CPU использование

Участвуют в анализе производительности и моделировании пространства совместно

$$AI = \# \text{ FLOPs} / \# \text{ BYTEs}$$



# Advisor Roofline: под капотом

Профилирование приложения:

Axis Y:  $\text{FLOP/S} = \# \text{FLOP} / \# \text{Seconds}$

Axis X:  $\text{AI} = \# \text{FLOP} / \# \text{Bytes}$

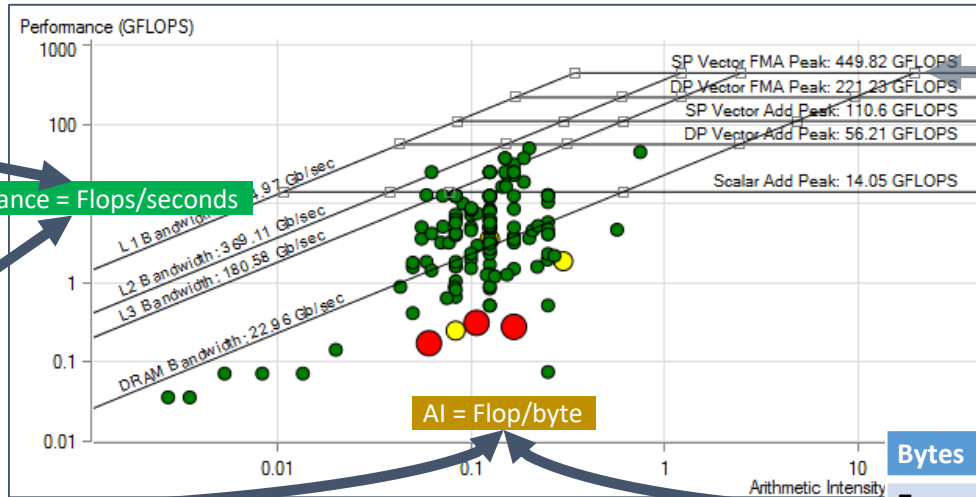
Seconds

Сэмплирование уровня пользователя  
*Root access not needed*

Performance = Flops/seconds

#FLOP

Бинарная инструментация  
Не зависит от счетчиков CPU



Roofs

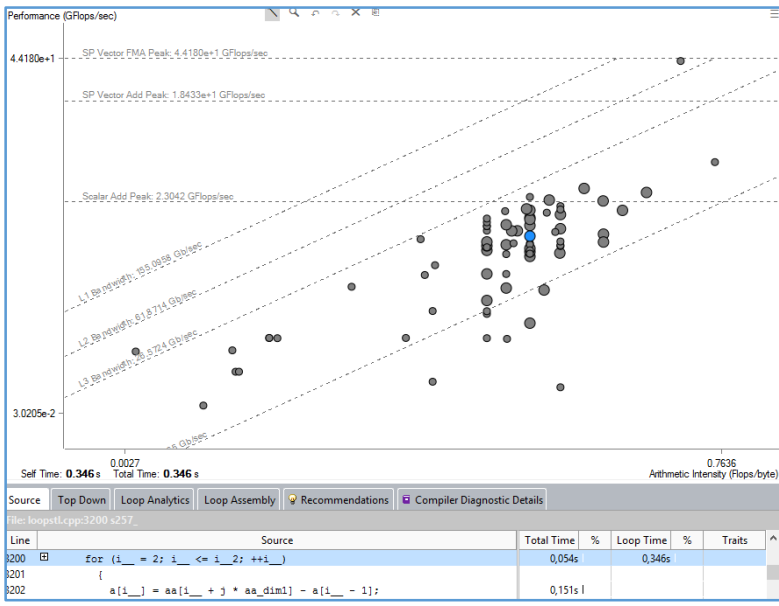
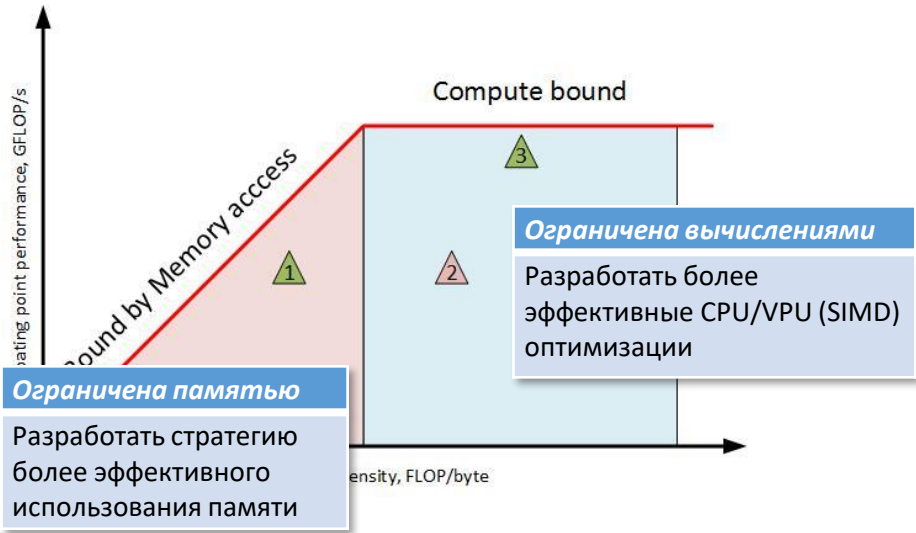
Микробенчмарки:  
Максимальная производительность для текущей архитектуры

Bytes

Бинарная инструментация  
Считает размер операндов (не кэш-линии)

# Интерпретация данных Roofline анализа.

Разработка стратегии оптимизации



# Дополнительные ссылки

Roofline model proposed by Williams, Waterman, Patterson:

<http://www.eecs.berkeley.edu/~waterman/papers/roofline.pdf>

“Cache-aware Roofline model: Upgrading the loft” (Ilic, Pratas, Sousa, INESC-ID/IST, Thec Uni of Lisbon) <http://www.inesc-id.pt/ficheiros/publicacoes/9068.pdf>

At Intel:

**Roman Belenov, Zakhar Matveev**, Julia Fedorova  
SSG product teams, Hugh Caffey,  
in collaboration with **Philippe Thierry**

# QA