



# Intel® Parallel Studio Инструменты Advisor и Inspector

Егор Казачков, Екатерина Антакова

[egor.kazachkov@intel.com](mailto:egor.kazachkov@intel.com)

[ekaterina.antakova@intel.com](mailto:ekaterina.antakova@intel.com)

# Содержание



- Трудности проектирования и реализации параллельного кода
- Поэтапное проектирование с помощью инструментов Intel
- Основные возможности на примере
- Новые возможности в версии 2015
- Вопросы и ответы



# 1. Трудности проектирования и реализации параллельного кода

# «Параллельная» реальность

- Процессоры становятся многоядерными – эти ресурсы надо использовать
- Никлаус Вирт:
  - «программы замедляются быстрее, чем ускоряется железо»
  - "software is getting slower more rapidly than hardware becomes faster"
- Параллельное программирование – сложная задача

# Трудности проектирования и реализации параллельного кода

## При проектировании

- Каков ожидаемый выигрыш от распараллеливания кода? Оправданы ли усилия разработчиков?
- Будет ли параллельный код масштабироваться с увеличением числа ядер и процессоров?

## В реализации

- Добиться желаемого ускорения программы
- Не сломать тесты
- Идентифицировать разделяемые ресурсы
- Отладить параллельный код и исправить трудновоспроизводимые ошибки

# Предпосылки

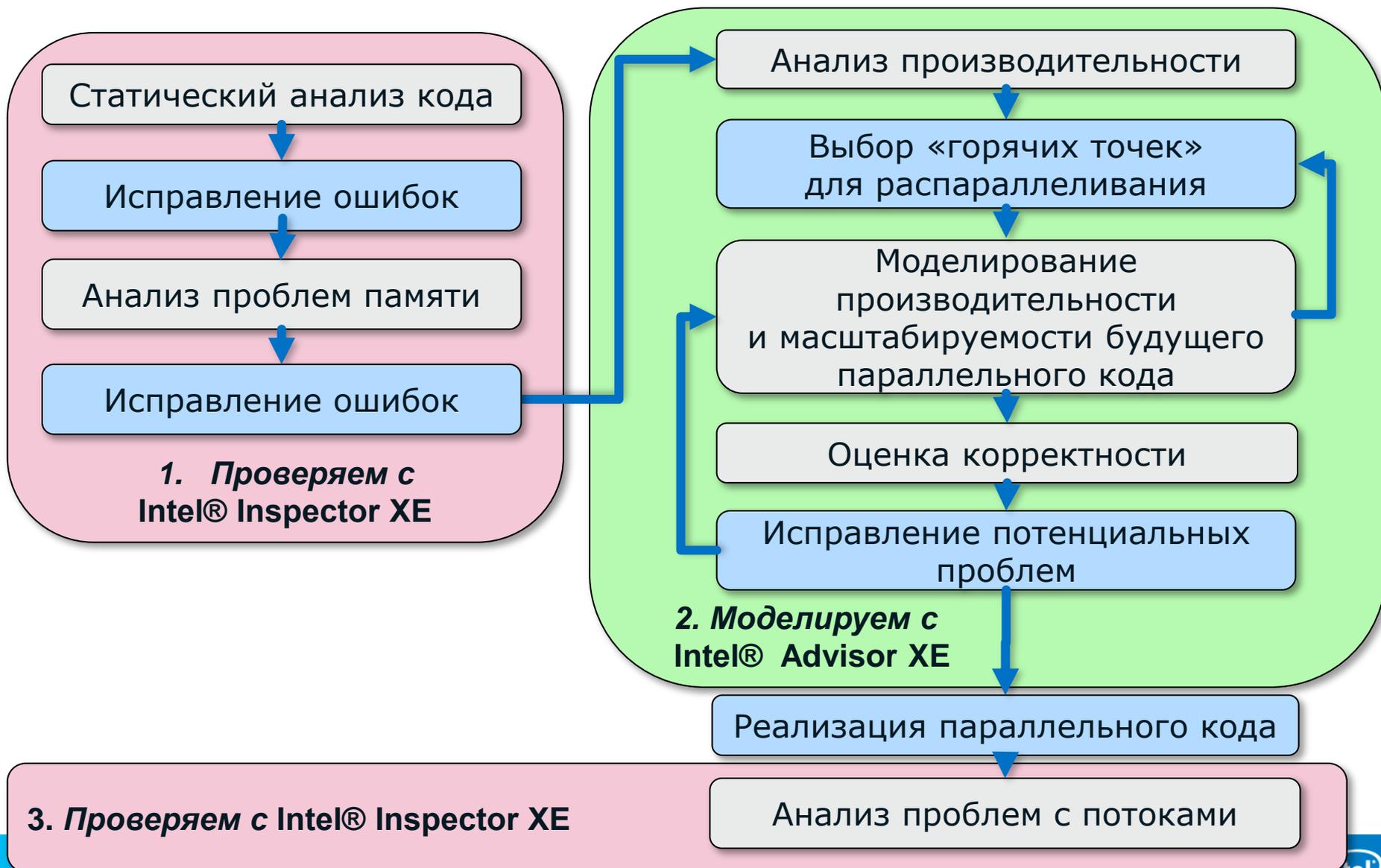
Имеет смысл использовать связку Advisor + Inspector, если:

- Есть работающая последовательная программа.
- Есть тесты или эталонные результаты.
- Требуется более полное использование вычислительных ресурсов.
- Требуется ускорить работу программы или обрабатывать больший объем данных за то же время.



## 2. Этапы проектирования параллельного кода и инструменты Intel для разработчиков

# Инструменты для разработчиков Intel и этапы проектирования параллелизма



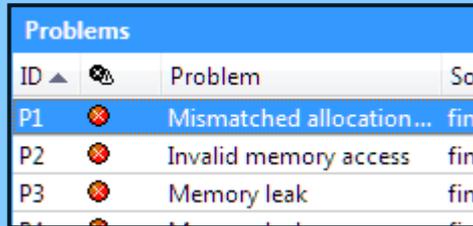
# Intel® Parallel Studio XE, Intel® Cluster Studio XE



# Intel® Inspector XE

Где в моём приложении проблемы в ...

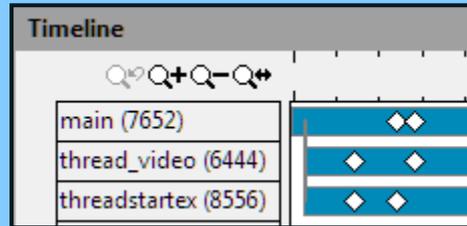
## памяти



ID	Problem	Source
P1	Mismatched allocation...	fin
P2	Invalid memory access	fin
P3	Memory leak	fin

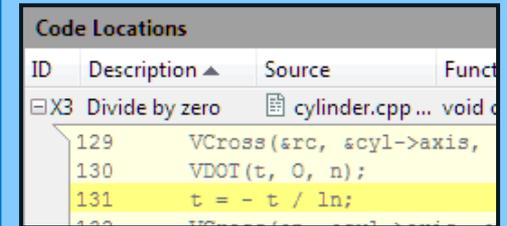
- Некорректный доступ
- Утечки памяти

## потоках



- Гонки данных
- Взаимные блокировки
- Доступ к чужим стекам

## безопасности



ID	Description	Source	Function
X3	Divide by zero	cylinder.cpp ...	void d
129		VCross(src, scyl->axis,	
130		VDOI(t, 0, n);	
131		t = - t / ln;	

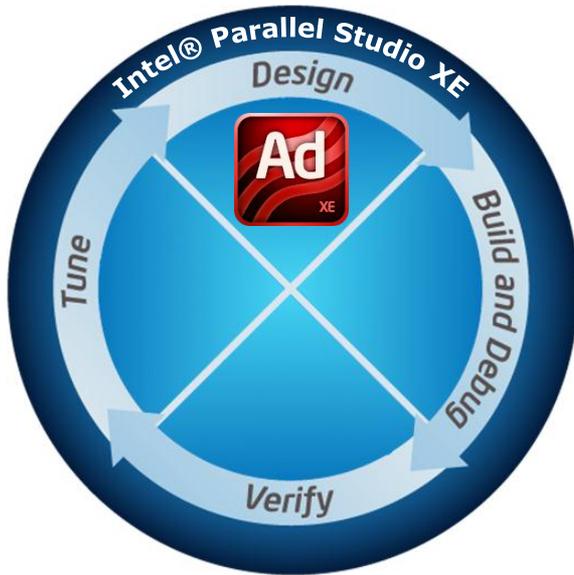
- Переполнение буферов
- Некорректные указатели

- Не требует специальной перекомпиляции программы
- Поддерживает остановку в дебаггере на найденной проблеме
- Работает на Windows\* и Linux\* 32-bit и 64-bit

Поиск трудноотлаживаемых ошибок на ранних стадиях разработки

# Intel® Advisor XE

## Средство этапа проектирования



- Оценка прироста производительности до реализации многопоточности
- Интеграция в Microsoft\* Visual Studio, отдельный GUI и CLI интерфейс
- C++, Fortran, C# .NET
- Windows\* и Linux\*



- Удобен для проектирования параллелизма с общей памятью

**Моделирование параллельного исполнения без риска «сломать» существующий код. Advisor XE увеличивает ROI распараллеливания**

# Intel® Advisor XE

- Ассистирует на каждом шаге процесса распараллеливания
- Знакомит с трудностями распараллеливания: взаимные блокировки, доступ к общим данным, вопросы масштабируемости
- Позволяет быстро экспериментировать с разными подходами к параллельности
- Даёт оценки потенциального прироста производительности для большого числа ядер

# Что учитывает модель Intel® Advisor XE

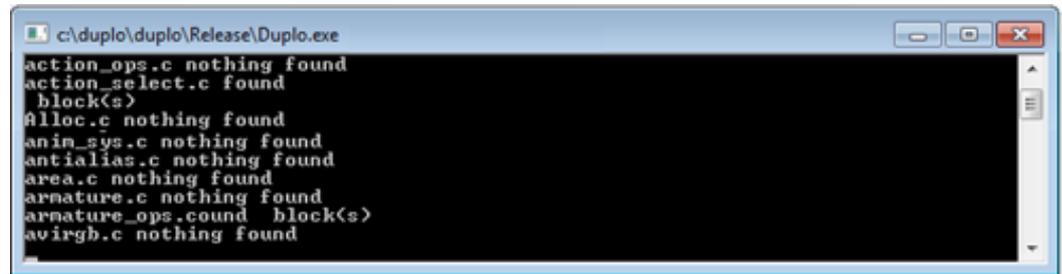
- Параллелизуемые CPU-bound вычисления (влияние последовательного кода и закон Амдала)
- Объём входных данных (закон Густафсона)
- Гранулярность задач, их объединение (chunking) и работа планировщика
- Распределение нагрузки по процессорам
- Ожидания в блокировках
- Накладные расходы на создание задач и потоков
- Возможности и ограничения аппаратного обеспечения



# 3. Основные возможности на примере

# Приложение *duplo*

<http://duplo.sourceforge.net/>



```
c:\duplo\duplo\Release\Duplo.exe
action_ops.c nothing found
action_select.c found
  block(s)
alloc.c nothing found
anim_sys.c nothing found
antialias.c nothing found
area.c nothing found
armature.c nothing found
armature_ops.c found  block(s)
avirgb.c nothing found
```

Поиск повторяющихся блоков текста в группе файлов

Алгоритм:

- сравнить все пары различных файлов
- для каждой пары файлов найти совпадающие строки и заполнить матрицу строк
- объединить строки из матрицы в блоки соседних строк
- вывести все найденные блоки в файл

# Особенности *duplo*

- Последовательное приложение
- Есть тестовые сценарии с эталонными выходными файлами
- Активное использование диска: запись строк в файл после проверки каждой пары файлов
- Входные текстовые файлы неоднородны

# Анализ проблем памяти с помощью Intel® Inspector XE

The screenshot displays the Intel Inspector XE 2013 interface. At the top, the title bar reads "Detect Memory Problems" and "Intel Inspector XE 2013". Below the title bar, there are navigation tabs: "Target", "Analysis Type", "Collection Log", and "Summary".

The main area is divided into several panels:

- Problems:** A table listing detected memory issues. The first row, P1, is highlighted with a red box. It shows a "Mismatched allocation/deallocation" error in "textfile.cpp" within the "Duplo.exe" module, with an object size of 1280 and a state of "New".
- Filters:** A panel on the right showing filters for "Severity" (Error, 5 item(s)), "Type" (Memory leak, 4 item(s); Mismatched allocation/deall..., 1 item(s)), and "Source".
- Code Locations:** A panel showing the specific code locations for the selected problem. It is titled "Code Locations: Mismatched allocation/deallocation".
- Timeline:** A panel on the far right showing a timeline of events, with "QueryEnvironment" visible.

The "Code Locations" panel is further divided into two sections:

- Mismatched deallocation site:** Located at "textfile.cpp:47" in the "readAll" function of "Duplo.exe". The code snippet shows:

```
45     std::ostringstream os;  
46     os.write(buffer, len);  
47     delete buffer;  
48     all = os.str();  
49     } else {
```

The line "delete buffer;" is highlighted with a red box.
- Allocation site:** Located at "textfile.cpp:42" in the "readAll" function of "Duplo.exe". The code snippet shows:

```
40     unsigned int len = inFile.tellg();  
41     inFile.seekg(0, std::ios::beg);  
42     char* buffer = new char[len];  
43     inFile.read(buffer, len);  
44     inFile.close();
```

The line "char\* buffer = new char[len];" is highlighted with a red box.

# Поиск горячих функций и циклов с помощью Intel® Advisor XE

Ad Where should I add parallelism?

Summary **Survey Report** Annotation Report Suitability Report Correctness Report

View Source

Function Call Sites and Loops	Total Time %	Total Time	Self Time	Top Loops	Source Location
Total	100.0%	89.5413s	0s		
pre_c_init	100.0%	89.5413s	0s		crtexe.c:225
_tmainCRTStartup	100.0%	89.5413s	0s		crtexe.c:410
main	100.0%	89.5413s	0s		duplo.cpp:295
Duplo::run	100.0%	89.5413s	0s		duplo.cpp:177
[loop at duplo.cpp:238 in Duplo::run]	97.7%	87.4985s	0.0321s		duplo.cpp:238
[loop at duplo.cpp:242 in Duplo::run]	97.7%	87.4664s	0s		duplo.cpp:242
Duplo::process	97.6%	87.4352s	0.4087s		duplo.cpp:91
[loop at duplo.cpp:102 in Duplo::process]	69.9%	62.5918s	0.0468s		duplo.cpp:102
[loop at duplo.cpp:104 in Duplo::process]	69.8%	62.4670s	9.6844s		duplo.cpp:104
SourceLine::equals	30.3%	27.1676s	27.1676s		sourceline.cpp:48
SourceFile::getLine	28.6%	25.6149s	25.6149s		sourcefile.cpp:168
SourceFile::getLine	0.1%	0.0780s	0.0780s		sourcefile.cpp:168
[loop at duplo.cpp:136 in Duplo::process]	17.1%	15.2795s	0.1091s		duplo.cpp:136
[loop at duplo.cpp:114 in Duplo::process]	10.2%	9.1552s	0.0623s		duplo.cpp:114
Duplo::isSameFilename	0.0%	0.0312s	0s		duplo.cpp:173
[loop at duplo.cpp:214 in Duplo::run]	2.3%	2.0428s	0s		duplo.cpp:214

# Исследование распараллеливания внешнего цикла

```
for(int i=0;i<(int)sourceFiles.size();i++){
    outfile << sourceFiles[i]->getFilename();
    int blocks = 0;

    for(int j=0;j<(int)sourceFiles.size();j++){
        if(i > j && !isSameFilename(sourceFiles[i]->getFilename(), sourceFiles[j]->getFilename())){
            blocks+=process(sourceFiles[i], sourceFiles[j], outfile);

        }
    }

    if(blocks > 0){
        outfile << " found " << blocks << " block(s)" << std::endl;
    } else {
        outfile << " nothing found" << std::endl;
    }

    blocksTotal+=blocks;
}
```

# Добавление аннотаций Intel® Advisor XE во внешний цикл

```
ANNOTATE_SITE_BEGIN(compare_files);
for(int i=0;i<(int)sourceFiles.size();i++){
    ANNOTATE_ITERATION_TASK(compare_files_task);
    outfile << sourceFiles[i]->getFilename();
    int blocks = 0;

    for(int j=0;j<(int)sourceFiles.size();j++){
        if(i > j && !isSameFilename(sourceFiles[i]->getFilename(), sourceFiles[j]->getFilename())){
            blocks+=process(sourceFiles[i], sourceFiles[j], outfile);
        }
    }

    if(blocks > 0){
        outfile << " found " << blocks << " block(s)" << std::endl;
    } else {
        outfile << " nothing found" << std::endl;
    }

    blocksTotal+=blocks;
}
ANNOTATE_SITE_END(compare_files);
```

# Моделирование масштабируемости внешнего цикла

Ad What are the performance implications of the annotated sites? 🗨

🌿 Summary   📊 Survey Report   💧 Annotation Report   🌿 Suitability Report   🔥 Correctness Report

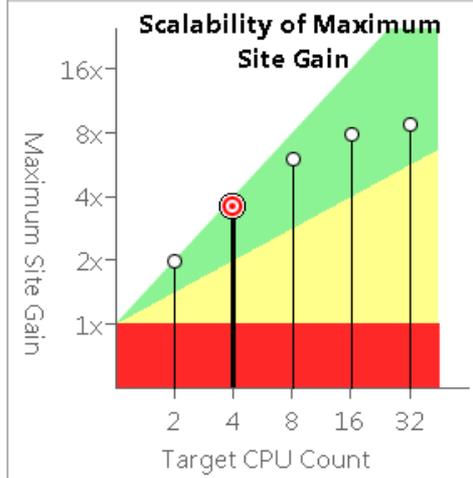
All Sites

Maximum Program Gain For All Sites: **3.41x**

Target CPU Count: 4   Threading Model: OpenMP

Annotation Label	Source Location	Maximum Site Gain	Maximum Total Gain	Average Instance Time	Total Time
compare_files	duplo.cpp:238	3.62x	3.41x	85.0414s	85.0414s

Selected Site



Changes I will make to this site to improve performance

Type of Change	Benefit if Checked	Loss if Unchecked	Recommended
<input type="checkbox"/> Reduce <a href="#">Site Overhead</a>			No
<input type="checkbox"/> Reduce <a href="#">Task Overhead</a>			No
<input type="checkbox"/> Reduce <a href="#">Lock Overhead</a>			No
<input type="checkbox"/> Reduce <a href="#">Lock Contention</a>			No
<input type="checkbox"/> Enable <a href="#">Task Chunking</a>			No

Annotation	Annotation Label	Source Location	Number of Instances	Maximum Instance Time	Average Instance Time	Minimum Instance Time	Total Time
Selected Site	compare_files	duplo.cpp:238	1	85.0414s	85.0414s	85.0414s	85.0414s
Task	compare_files_task	duplo.cpp:240	191	8.6420s	0.4452s	0.0004s	85.0413s

# Моделирование проблем корректности

Ad Did the annotated tasks expose data sharing problems? ▣

Summary Survey Report Annotation Report Suitability Report Correctness Report

## Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	compare_files	duplo.cpp	Duplo.exe	✓ Not a problem
P2	Memory reuse	compare_files	duplo.cpp; newaop.cpp	Duplo.exe	🚩 New
P3	Memory reuse	compare_files	duplo.cpp; sourceline.cpp	Duplo.exe	🚩 New

## Memory reuse: Code Locations

ID	Description	Source	Function	Module	State
X2	Write	duplo.cpp:101	process	Duplo.exe	🚩 New
<pre>99 100 // Reset matrix data 101 memset(m_pMatrix, NONE, m*n); 102 103 // Compute matrix</pre>					
X3	Read	duplo.cpp:120	process	Duplo.exe	🚩 New
<pre>118 int maxX = MIN(n, m-y); 119 for(int x=0; x&lt;maxX; x++){ 120     if(m_pMatrix[x+n*(y+x)] == MATCH) { 121         seqLen++; 122     } else {</pre>					

# Исправление проблем корректности - будущих проблем параллельного исполнения

- Глобальную матрицу совпадающих строк **m\_pMatrix** разбить на локальные для каждой итерации матрицы
- Добавить синхронизацию переменной **blocksTotal** – суммарное количество совпадающих блоков строк
- Синхронизировать вывод строк в файл

# Добавление аннотаций блокировки

```
ANNOTATE_SITE_BEGIN(compare_files);
for(int i=0;i<(int)sourceFiles.size();i++){
    ANNOTATE_ITERATION_TASK(compare_files_task);
    outfile << sourceFiles[i]->getFilename();
    int blocks = 0;
    std::stringstream task_stream;

    for(int j=0;j<(int)sourceFiles.size();j++){
        if(i > j && !isSameFilename(sourceFiles[i]->getFilename(), sourceFiles[j]->getFilename())){
            blocks+=process(sourceFiles[i], sourceFiles[j], task_stream);
        }
    }

    if(blocks > 0){
        task_stream << " found " << blocks << " block(s)" << std::endl;
    } else {
        task_stream << " nothing found" << std::endl;
    }
}
ANNOTATE_LOCK_ACQUIRE(0);
blocksTotal+=blocks;
outfile << task_stream.str();
ANNOTATE_LOCK_RELEASE(0);
}
ANNOTATE_SITE_END(compare_files);
```

# Моделирование масштабируемости внешнего цикла с блокировкой

Ad What are the performance implications of the annotated sites? ▢

🌿 Summary
📊 Survey Report
💧 Annotation Report
👤 Suitability Report
🔥 Correctness Report

All Sites

**Maximum Program Gain For All Sites:**

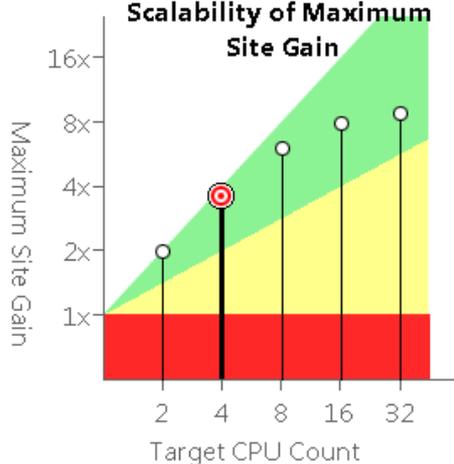
**3.40x**

Target CPU Count:  Threading Model:

Annotation Label	Source Location	Maximum Site Gain	Maximum Total Gain	Average Instance Time	Total Time
compare_files	duplo.cpp:238	3.62x	3.40x	85.1638s	85.1638s

Selected Site

**Scalability of Maximum Site Gain**



**Changes I will make to this site to improve performance**

Type of Change	Benefit if Checked	Loss if Unchecked	Recommended
<input type="checkbox"/> Reduce <a href="#">Site Overhead</a>			No
<input type="checkbox"/> Reduce <a href="#">Task Overhead</a>			No
<input type="checkbox"/> Reduce <a href="#">Lock Overhead</a>			No
<input type="checkbox"/> Reduce <a href="#">Lock Contention</a>			No
<input type="checkbox"/> Enable <a href="#">Task Chunking</a>			No

Annotation	Annotation Label	Source Location	Number of Instances	Maximum Instance Time	Average Instance Time	Minimum Instance Time	Total Time
Selected Site	compare_files	duplo.cpp:238	1	85.1638s	85.1638s	85.1638s	85.1638s
Task	compare_files_task	duplo.cpp:240	191	8.6719s	0.4459s	0.0007s	85.1637s
Lock		?	191	< 0.0001s	< 0.0001s	< 0.0001s	0.0004s

# Масштабируемость внутреннего цикла

Ad What are the performance implications of the annotated sites? ▢

Summary Survey Report Annotation Report **Suitability Report** Correctness Report

All Sites

**Maximum Program Gain For All Sites:**

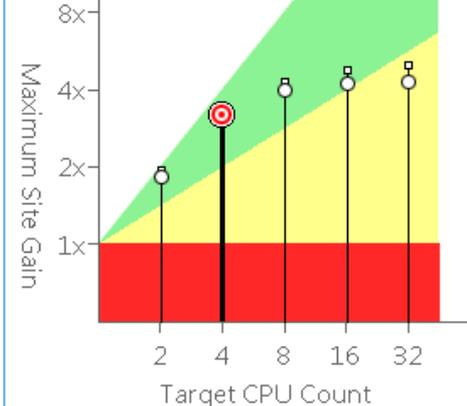
**3.01x**

Target CPU Count: 4 Threading Model: OpenMP

Annotation Label	Source Location	Maximum Site Gain	Maximum Total Gain	Average Instance Time	Total Time
inner loop	duplo.cpp:244	3.20x	3.01x	0.4356s	83.2056s

Selected Site

Scalability of Maximum Site Gain



Changes I will make to this site to improve performance

Type of Change	Benefit if Checked	Loss if Unchecked	Recommended
<input type="checkbox"/> Reduce <a href="#">Site Overhead</a>	0.02x		No
<input type="checkbox"/> Reduce <a href="#">Task Overhead</a>	0.16x		No
<input type="checkbox"/> Reduce <a href="#">Lock Overhead</a>			No
<input type="checkbox"/> Reduce <a href="#">Lock Contention</a>			No
<input type="checkbox"/> Enable <a href="#">Task Chunking</a>	0.16x		No

Annotation	Annotation Label	Source Location	Number of Instances	Maximum Instance Time	Average Instance Time	Minimum Instance Time	Total Time
Selected Site	inner loop	duplo.cpp:244	191	7.2384s	0.4356s	< 0.0001s	83.2056s
Task	inner loop task	duplo.cpp:246	36,481	2.3818s	0.0023s	< 0.0001s	83.2026s
Lock		?	18,145	0.0396s	< 0.0001s	< 0.0001s	0.0657s

# Реализация параллельного кода внешнего цикла с помощью OpenMP

```
#pragma omp parallel for schedule(guided) reduction(+:blocksTotal)
for(int i=0;i<(int)sourceFiles.size();i++){
    std::cout << sourceFiles[i]->getFilename();
    int blocks = 0;
    std::stringstream task_stream;

    for(int j=0;j<(int)sourceFiles.size();j++){
        if(i > j && !isSameFilename(sourceFiles[i]->getFilename(), sourceFiles[j]->getFilename())){
            blocks+=process(sourceFiles[i], sourceFiles[j], task_stream);
        }
    }

    if(blocks > 0){
        task_stream << " found " << blocks << " block(s)" << std::endl;
    } else {
        task_stream << " nothing found" << std::endl;
    }

    blocksTotal+=blocks;

    omp_set_lock(&omp_lock);
    outfile << task_stream.str();
    omp_unset_lock(&omp_lock);
}
}
```

# Анализ проблем потоков с помощью Intel® Inspector XE

Intel Inspector XE 2013

Locate Deadlocks and Data Races

Target Analysis Type Collection Log Summary

ID	Type	Sources	Modules	State
P1	Data race	Duplo.cpp	Duplo.exe	New
	Data race	Duplo.cpp:88	Duplo.exe	Not fixed
	Data race	Duplo.cpp:88	Duplo.exe	Not fixed

Filters: Severity (Error: 1 item(s)), Type (Data race: 1 item(s)), Source (Duplo.cpp: 1 item(s)), Module (Duplo.exe: 1 item(s)), State (New: 1 item(s))

Гонка данных при подсчёте суммы строк-дубликатов.  
**Решение:** блокировка или использование локальных переменных в каждом потоке с последующим суммированием

1 of 3 All Code Locations: Data race

Description	Source	Function	Module
Read	Duplo.cpp:88	reportSeq	Duplo.exe
86	for(int j=0;j<count;j++)	Duplo.exe!reportSeq - Du	
87	outFile << pSource1->	Duplo.exe!process - Dupl	
88	m_DuplicateLines++;	Duplo.exe!run - Duplo.cp	
89	}		
90	outFile << std::endl;		
Write	Duplo.cpp:88	reportSeq	Duplo.exe
86	for(int j=0;j<count;j++)	Duplo.exe!reportSeq - Du	
87	outFile << pSource1->	Duplo.exe!process - Dupl	
88	m_DuplicateLines++;	Duplo.exe!run - Duplo.cp	

Timeline: OMP Worker Thread #4 (5808), OMP Worker Thread #3 (4752)

# Сравнение последовательной и параллельной версии *duplo* в Intel® VTune™ Amplifier XE

Hotspots Hotspots by CPU Usage viewpoint (change) ?

Summary Bottom-up Caller/Callee Top-down Tree

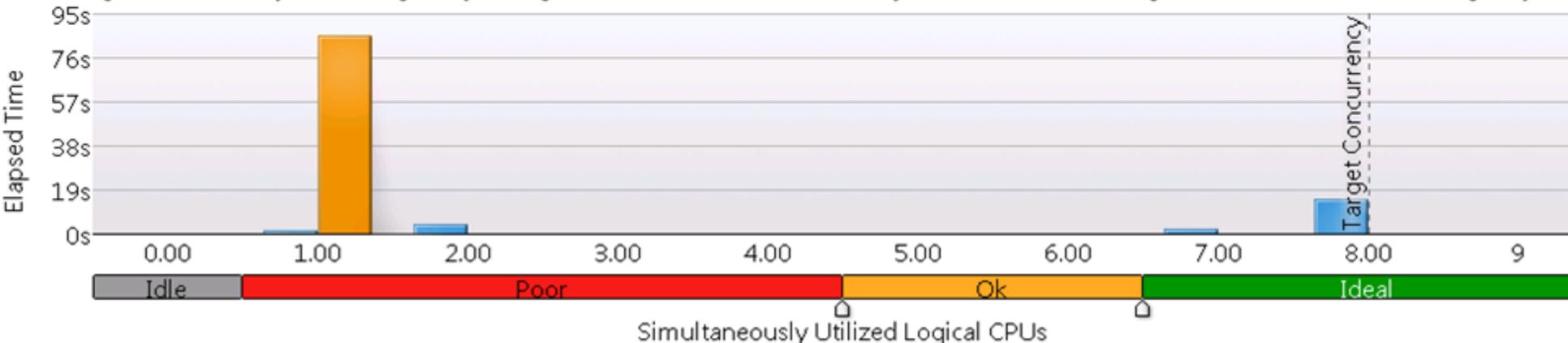
Elapsed Time: 28.413s - 89.427s = -61.014s

Total Thread Count: 8 - 1 = 7  
Overhead Time: Not changed, 0s  
Spin Time: 1.125s - 0s = 1.125s  
CPU Time: 169.360s - 85.397s = 83.963s  
Paused Time: Not changed, 0s  
Frame Count: Not changed, [Unknown]

Ускорение всего приложения в **3,15** раза на 4-ядерном процессоре за счёт распараллеливания внешнего цикла

## CPU Usage Histogram

This histogram represents a breakdown of the Elapsed Time. It visualizes what percentage of the wall time the specific number of CPUs were running simultaneously. CPU Usage may be higher than the thread concurrency if a thread is executing code on a CPU while it is logically...





# 4. Новые возможности Inspector и Advisor в Intel® Parallel Studio XE 2015 Beta

# Inspector и Advisor: Использование в среде MPI

- Запуск под MPI (Inspector):

```
mpirun -n 3 inspxe-cl -collect mi3 -r  
res_{at}_{mpirank} -- mpi_test.exe
```

- Просмотр результата (Advisor):

```
advixe-cl -project-dir <dir1> -import-dir <dir2>  
-mpi-rank 2
```

- Только в режиме командной строки
- Может понадобиться ручное копирование результатов с других MPI узлов.

# Inspector:

## Улучшенный алгоритм обнаружения чтения из неинициализированной памяти

Ограничение предыдущих версий:

- Ложные срабатывания, когда неинициализированная память читается, но не используется.

В Inspector XE 2015 Beta мы реализовали улучшенный алгоритм определения доступа к неинициализированной памяти.

**Detect Memory Problems** Copy

Medium scope memory error analysis type. Increases the load on the system and the time and resources required to perform analysis. Press F1 for more details.

- Detect uninitialized memory reads
- Revert to previous uninitialized memory algorithm
- Detect memory leaks upon application exit
- Detect resource leaks

# Advisor: Моделирование целевой платформы

- Подходит ли сопроцессор Xeon Phi™ для данной задачи?
- Какова производительность сопроцессора Xeon Phi™ по сравнению с основным процессором Xeon?
- Ограничение бета версии: не моделируется передача данных на сопроцессор. Эта возможность будет доступна в следующем обновлении.
- Экспериментальная возможность – чтобы включить, используйте переменную окружения:  
`set ADVIXE_EXPERIMENTAL=suitability_xeon_phi_modeling`  
Будет доступна по умолчанию в следующем обновлении.

# Advisor: Моделирование целевой платформы

[Summary](#)
[Survey Report](#)
[Annotation Report](#)
[Suitability Report](#)
[Correctness Report](#)

**Maximum Program Gain For All Sites: 5.37x**

Target System: Offload to Intel Xeon Phi Threading Model: Intel TBB Target CPU Count: 32

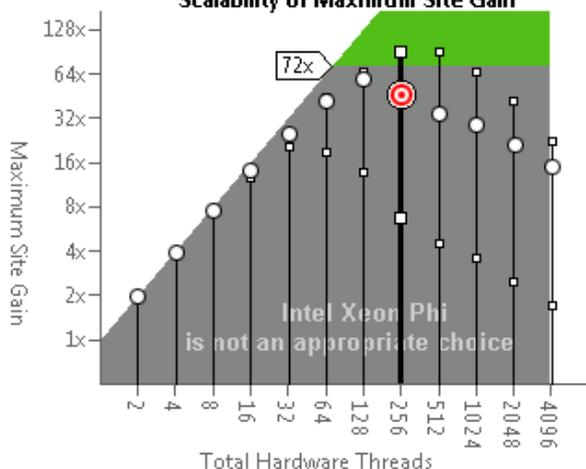
Threads: 256

Serial time: 22.2974s  
Predicted Parallel time: 4.1490s

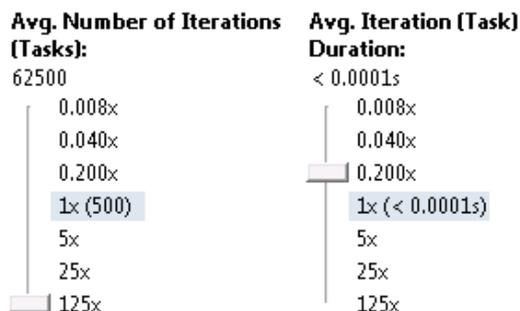
Site Label	Source Location	Offload to Intel Xeon Phi	Impact to Program Gain	Combined Site Metrics, All Instances			Average Site Instance Metr	
				Total Serial Time	Total Parallel Time	Site Gain	Average Serial Time	Ave
Site1	sdp_twosites. ...	<input checked="" type="checkbox"/>	1.46x	73.07s	0.3193s	228.81x	73.07s	0.31
Site2	sdp_twosites. ...	<input checked="" type="checkbox"/>	2.00x	142.47s	3.0859s	46.17x	0.2849s	0.00

Site Performance Scalability Site Details

Scalability of Maximum Site Gain



Loop Iterations (Tasks) Modeling



Runtime Modeling

- | Type of Change  | Gain Benefit if Checked |
|---|-------------------------|
| <input checked="" type="checkbox"/> Reduce <a href="#">Site Overhead</a>            |                         |
| <input checked="" type="checkbox"/> Reduce <a href="#">Task Overhead</a>            |                         |
| <input type="checkbox"/> Reduce <a href="#">Lock Overhead</a>                       |                         |
| <input checked="" type="checkbox"/> Reduce <a href="#">Lock Contention</a>          |                         |
| <input type="checkbox"/> Enable <a href="#">Task Chunking</a>                       | +15.85x                 |
| <input checked="" type="checkbox"/> Enable <a href="#">Ideal Code Vectorization</a> |                         |
- Apply

# Advisor:

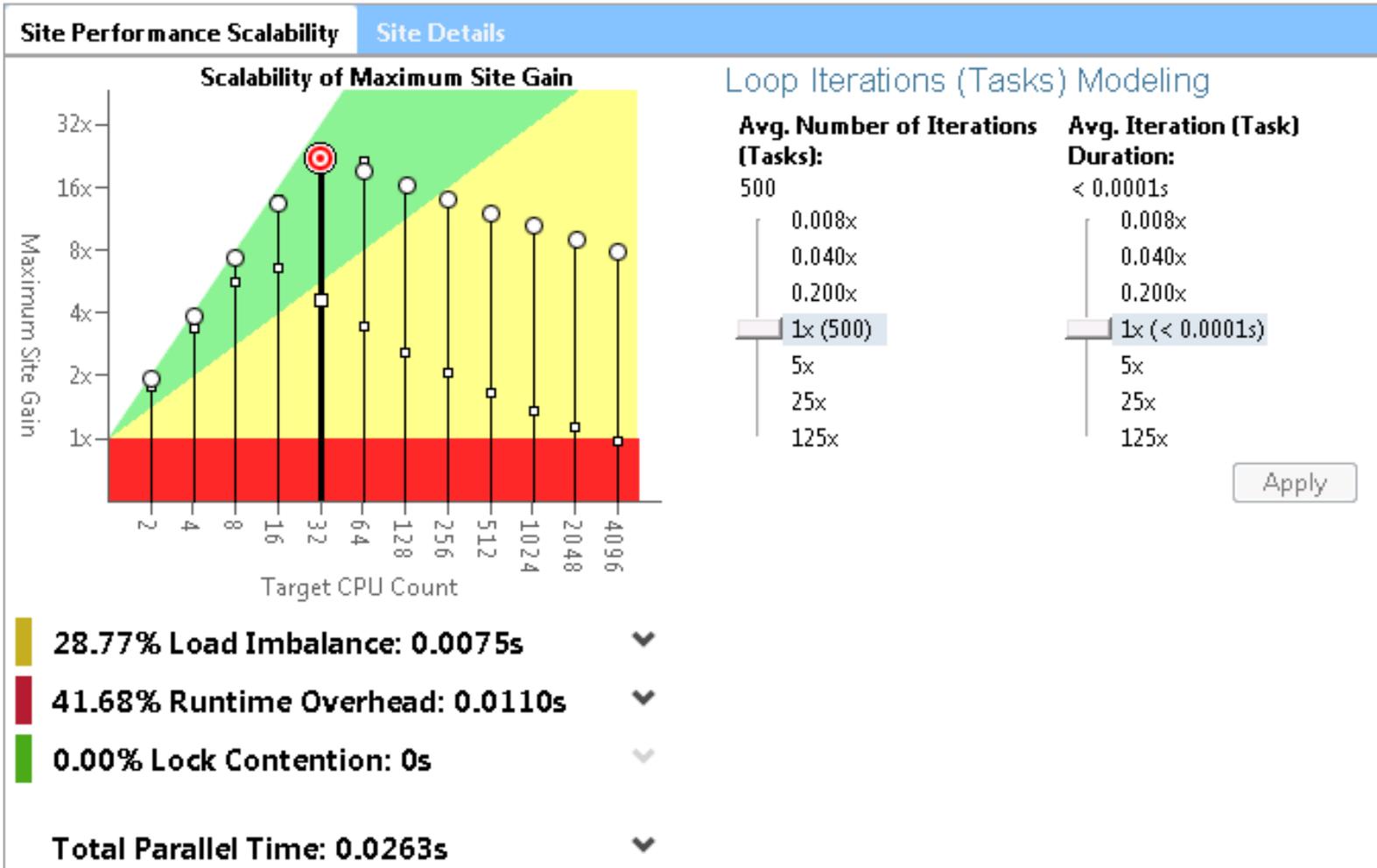
## Моделирование пространства итераций

- Как изменится масштабируемость при изменении размеров данных?
  - Изменение количества итераций в циклах
  - Изменение длительности итераций.
- Используйте небольшой набор данных для быстрого анализа, и используйте указанные опции для моделирования большого объема данных.

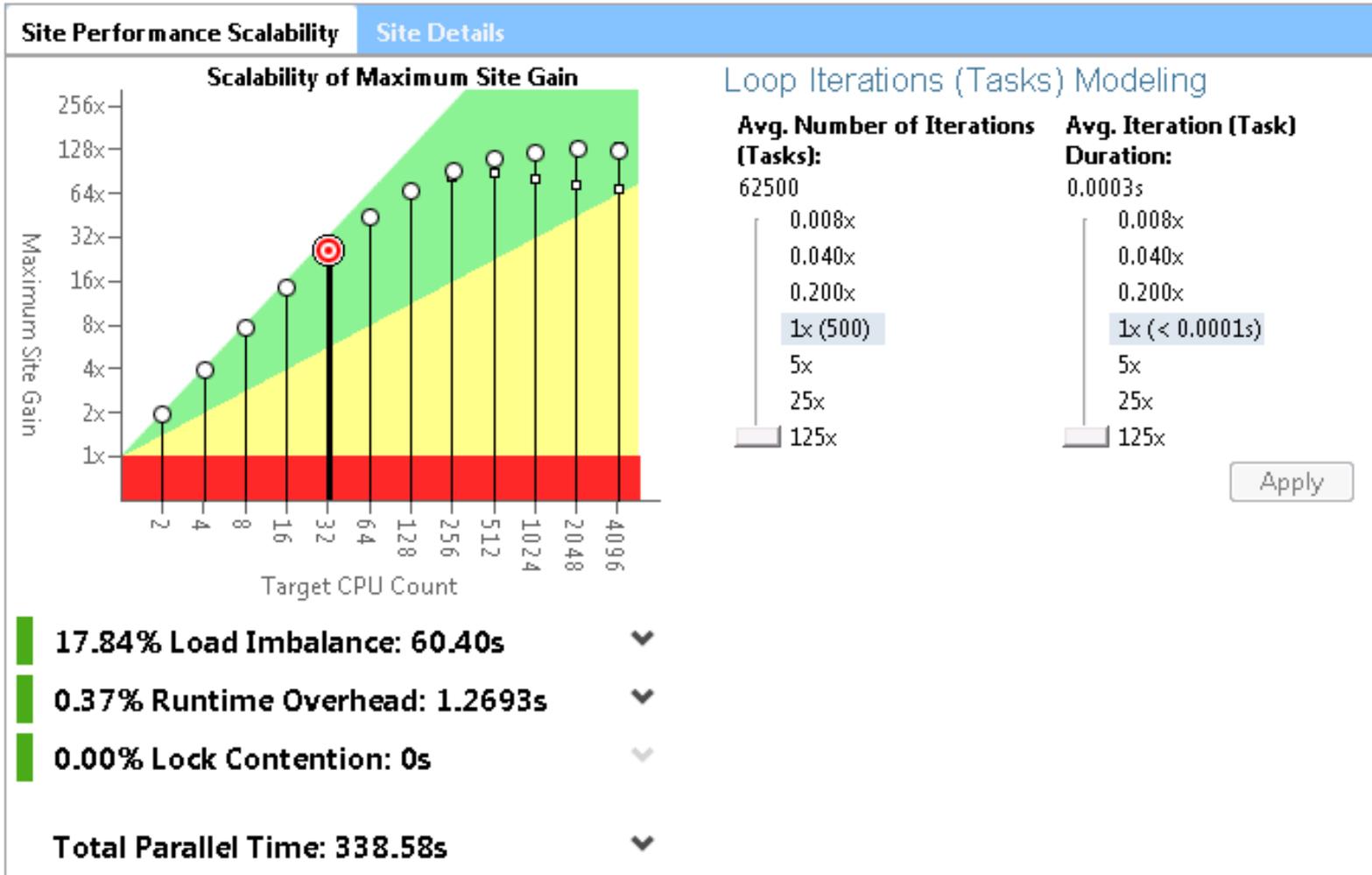
# Advisor: Сводные данные о потерях производительности

- Каковы основные факторы, ограничивающие производительность и масштабируемость?
  - Неравномерная нагрузка (load imbalance)
  - Накладные расходы на распараллеливание (runtime overhead)
  - Конфликты за семафоры/мьютексы (lock contention)

# Advisor: Моделирование пространства итераций



# Advisor: Моделирование пространства итераций



# Заключение

- Получение масштабируемого и корректного параллельного приложения требует экспериментов с различными подходами. **Intel® Advisor XE моделирует прирост производительности и возможные проблемы корректности ещё до реализации параллельного кода.**
- Дефекты использования памяти и многопоточности **нестабильны и трудно отлаживаемы**. Intel® Inspector XE помогает определить эти ошибки в последовательном и параллельном коде.
- Инструменты Intel® Parallel Studio XE и Intel® Cluster Studio XE для разработчиков программного обеспечения позволяют **проектировать параллелизм быстрее и сохранять корректность кода** при переходе к параллельной версии.



**Спасибо!**

[egor.kazachkov@intel.com](mailto:egor.kazachkov@intel.com)  
[ekaterina.antakova@intel.com](mailto:ekaterina.antakova@intel.com)

# 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 © , Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Core, VTune, and Cilk 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

