

Девятая независимая  
научно-практическая конференция  
«Разработка ПО 2013»

23 - 25 октября, Москва



## Проверяем, моделируем и снова проверяем

Три практических совета по облегчению разработки  
параллельных программ с помощью инструментов Intel

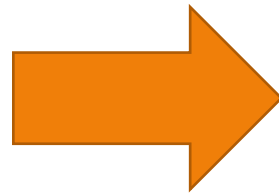
Екатерина Антакова

Intel Corporation

# План доклада

1. Трудности проектирования и реализации параллельного кода
2. Поэтапное проектирование с помощью инструментов Intel и три совета
3. Пример: распараллеливание программы поиска строк-дубликатов
4. Вопросы и ответы

# 1. Трудности проектирования и реализации параллельного кода



# Трудности проектирования и реализации параллельного кода

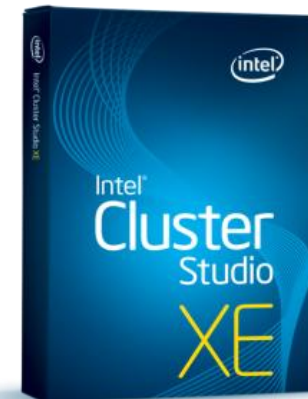
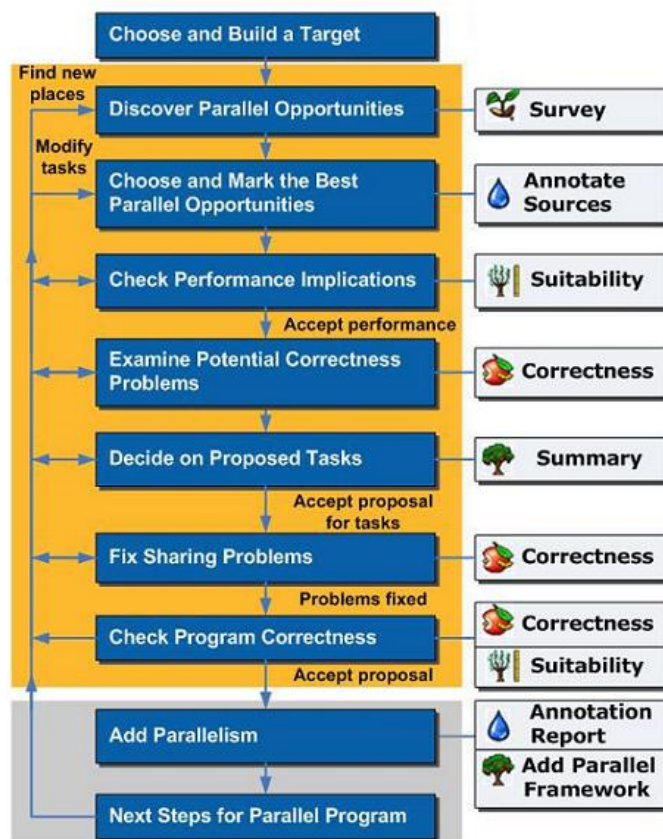
- **При проектировании**

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

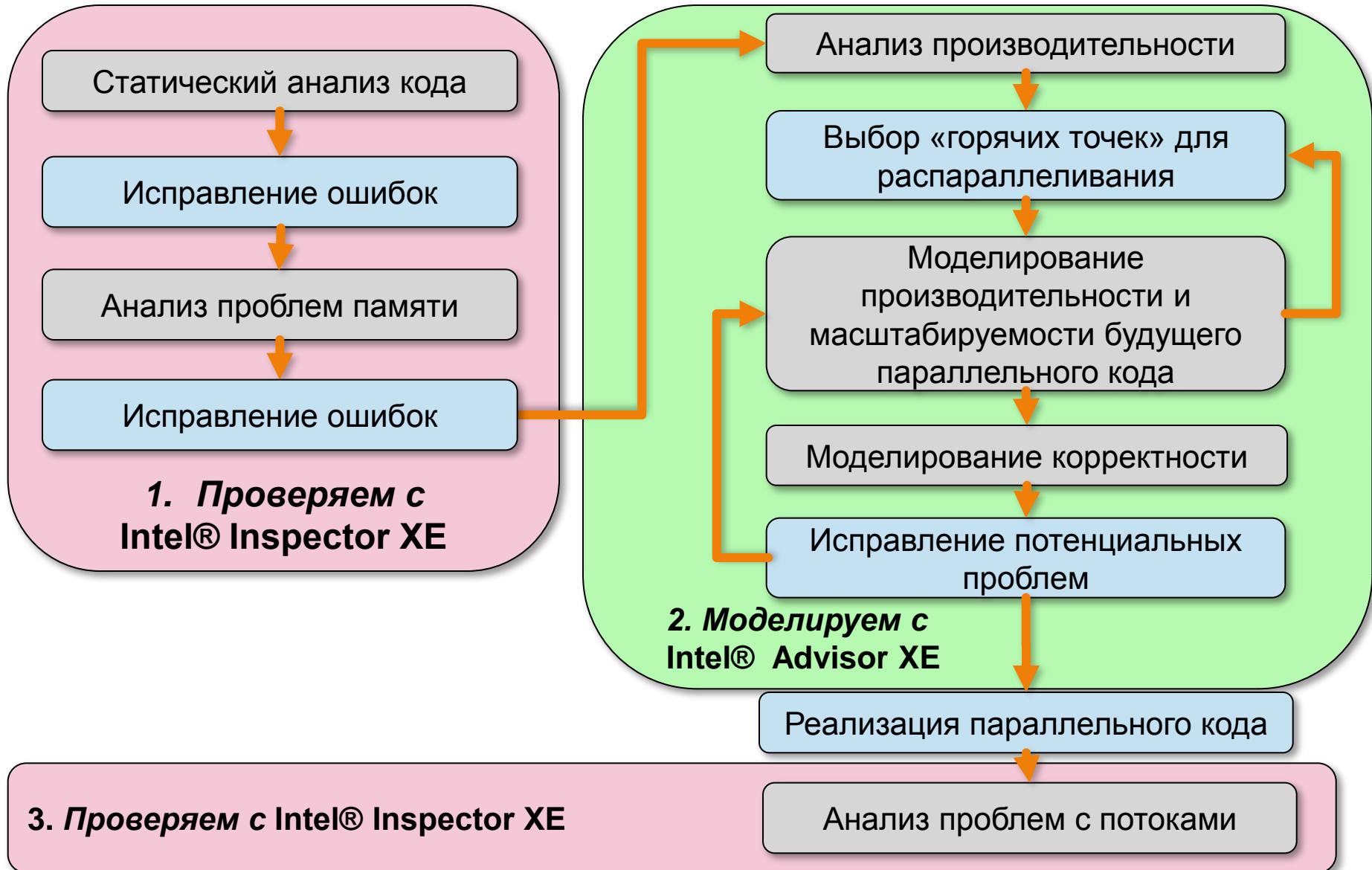
- **В реализации**

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

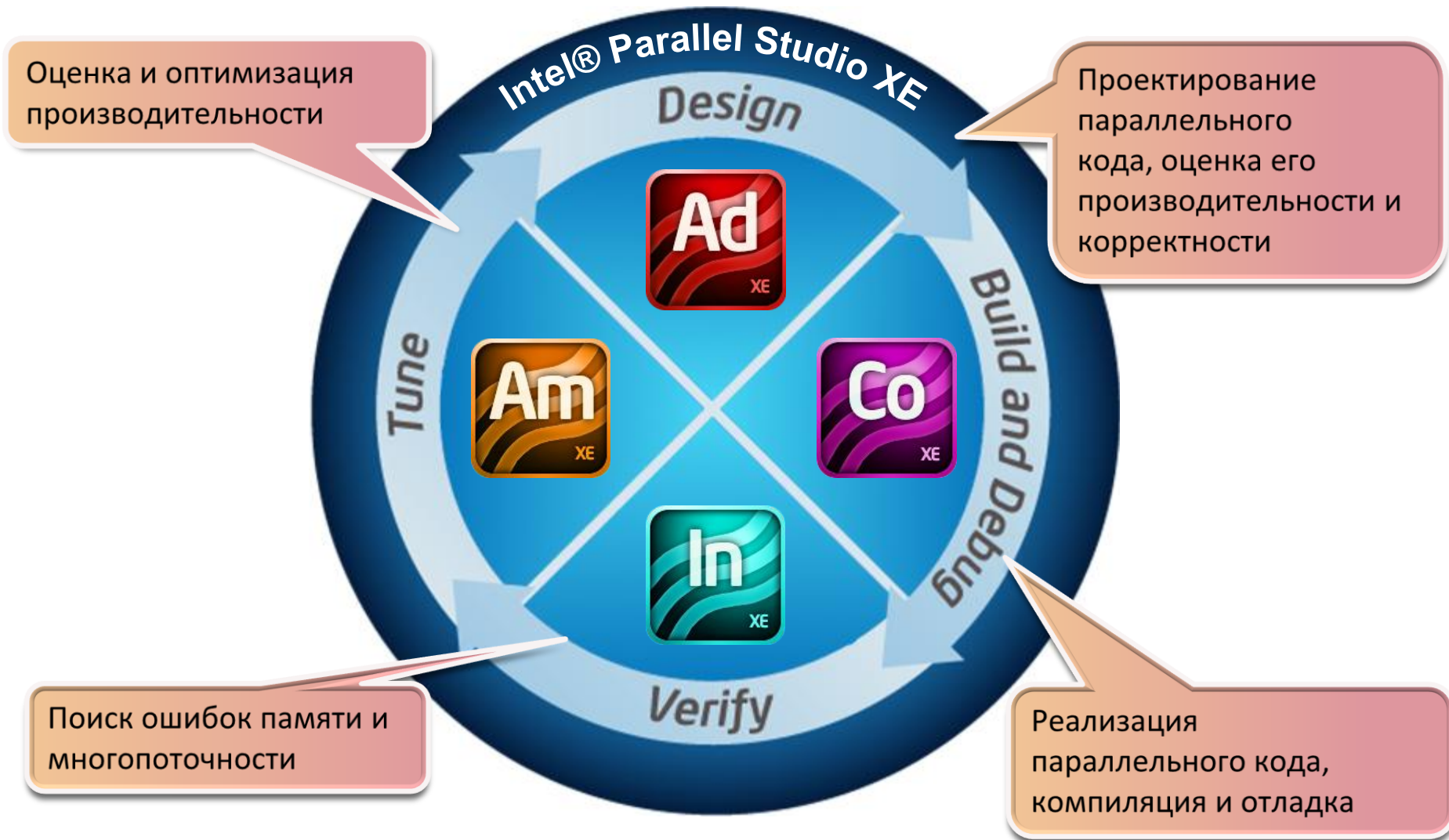
## 2. Этапы проектирования параллельного кода и инструменты Intel для разработчиков



# Инструменты для разработчиков Intel и этапы проектирования параллелизма



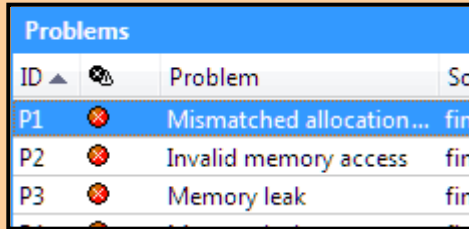
# Intel® Parallel Studio XE, Intel® Cluster Studio XE



# Intel® Inspector XE

Где в моём приложении проблемы в ...

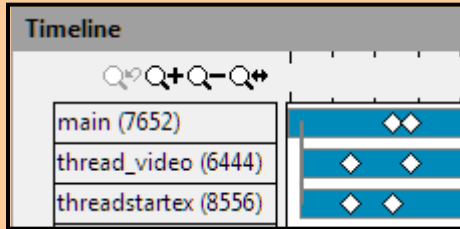
## памяти



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

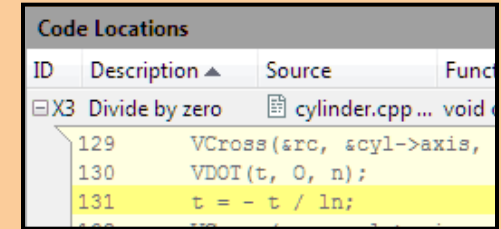
- Некорректный доступ
- Утечки памяти

## потоках



- Гонки данных
- Взаимные блокировки
- Доступ к чужим стекам

## безопасности



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

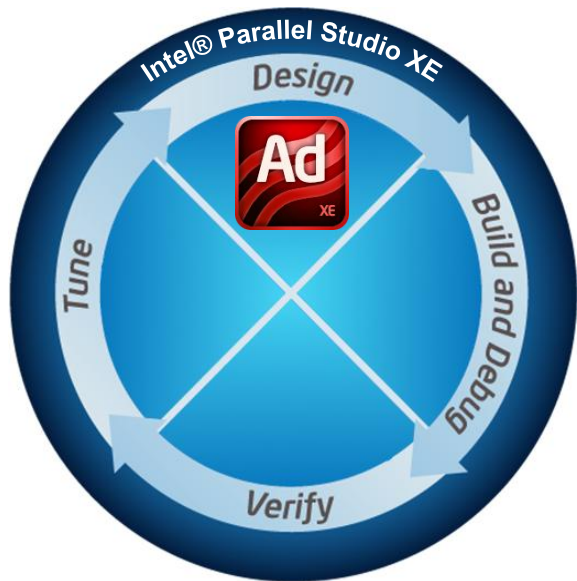
- Переполнение буферов
- Некорректные указатели

- Не требует специальной перекомпиляции
- Поддерживает процессоры Intel и x86-совместимые процессоры
- Операционные системы 32-bit и 64-bit, Windows\* и Linux\*

**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

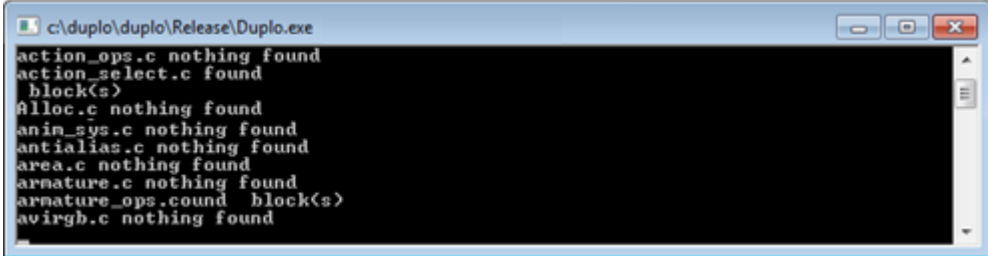
Моделирование параллельного исполнения без риска «сломать» существующий код.  
Advisor XE увеличивает ROI распараллеливания

### 3. Пример распараллеливания



# 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
anin_sys.c nothing found
antialias.c nothing found
area.c nothing found
armature.c nothing found
armature_ops.cound block(s)
avirgb.c nothing found
```

- Поиск повторяющихся блоков текста в группе файлов
- Алгоритм:
  - сравнить все пары различных файлов
  - для пары файлов найти совпадающие строки и заполнить матрицу строк
  - объединить строки из матрицы в блоки соседних строк
  - вывести все найденные блоки в файл

# Особенности приложения *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 for detecting memory problems. The main window is titled "Detect Memory Problems" and shows a list of detected issues in the "Problems" pane. The "Summary" tab is selected, showing a list of five problems (P1 to P5). Problem P1 is highlighted with a red box and is of type "Mismatched allocation/deallocation". The "Filters" pane on the right shows the current filter settings: Severity (Error), Type (Memory leak, Mismatched allocation/deallocation), and Source.

ID	Type	Sources	Modules	Object Size	State
P1	Mismatched allocation/deallocation	textfile.cpp	Duplo.exe		New
P2	Memory leak	duplo.cpp	Duplo.exe	1280	New
P3	Memory leak	hashutil.c...	Duplo.exe	143696	New
P4	Memory leak	sourcefile. ...	Duplo.exe	502936	New
P5	Memory leak	xmemory	Duplo.exe	409260	New

The "Code Locations: Mismatched allocation/deallocation" pane shows the following details:

Description	Source	Function	Module	Object Size	Offset
Mismatched deallocation site	textfile.cpp:47	readAll	Duplo.exe		
Allocation site	textfile.cpp:42	readAll	Duplo.exe		

The code snippets for the deallocation and allocation sites are shown below:

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

```
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 "Timeline" pane on the right shows the execution flow, including "QueryEnvironment".

# 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

# Выбор внешнего цикла для распараллеливания

```
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

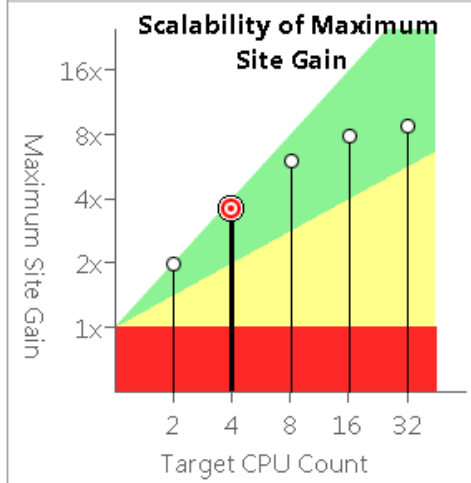
All Sites

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

- 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:**

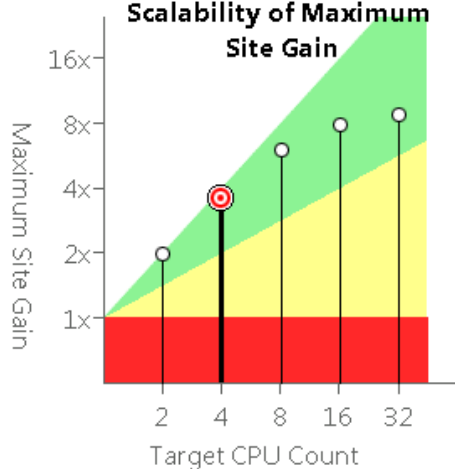
**3.40x**

Target CPU Count: 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	3.40x	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:**

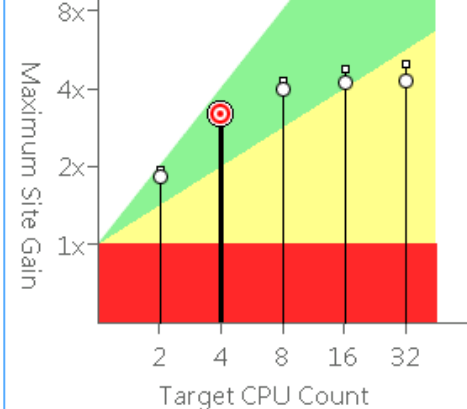
**3.01x**

Target CPU Count:  Threading Model:

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

The screenshot shows the Intel Inspector XE 2013 interface. The main window is titled "Locate Deadlocks and Data Races". The "Problems" pane on the left shows a list of detected issues. One issue, labeled "P1", is highlighted with an orange box. It is a "Data race" in "Duplo.cpp" at line 88, with a state of "New". The "Filters" pane on the right shows the current filter settings: Severity: Error (1 item(s)), Type: Data race (1 item(s)), Source: Duplo.cpp (1 item(s)), Module: Duplo.exe (1 item(s)), and State: New (1 item(s)).

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

**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

The screenshot shows the "Code Locations: Data race" pane and the "Timeline" pane. The "Code Locations" pane shows the source code for the data race, with lines 86-90 highlighted. The "Timeline" pane shows the execution of the code by two threads: OMP Worker Thread #4 (5808) and OMP Worker Thread #3 (4752).

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

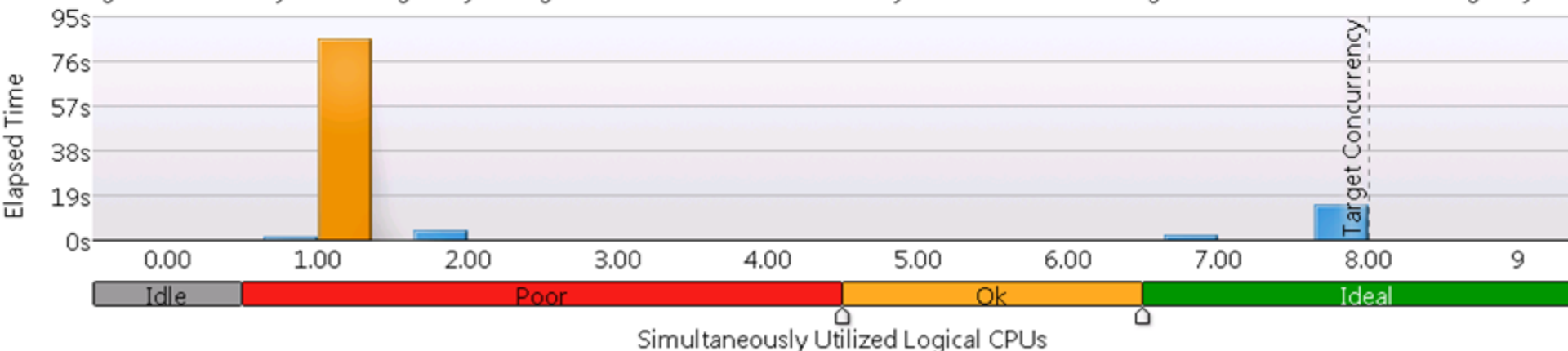
Elapsed Time: 28.413s - 89.427s = -61.014s

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...



# Заключение

- Получение масштабируемого и корректного параллельного приложения требует экспериментов с различными подходами к параллельности. **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 за внимание!

# 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