# Intel® Parallel Studio
# Инструменты Advisor и Inspector

Егор Казачков, Екатерина Антакова

egor.kazachkov@intel.com

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

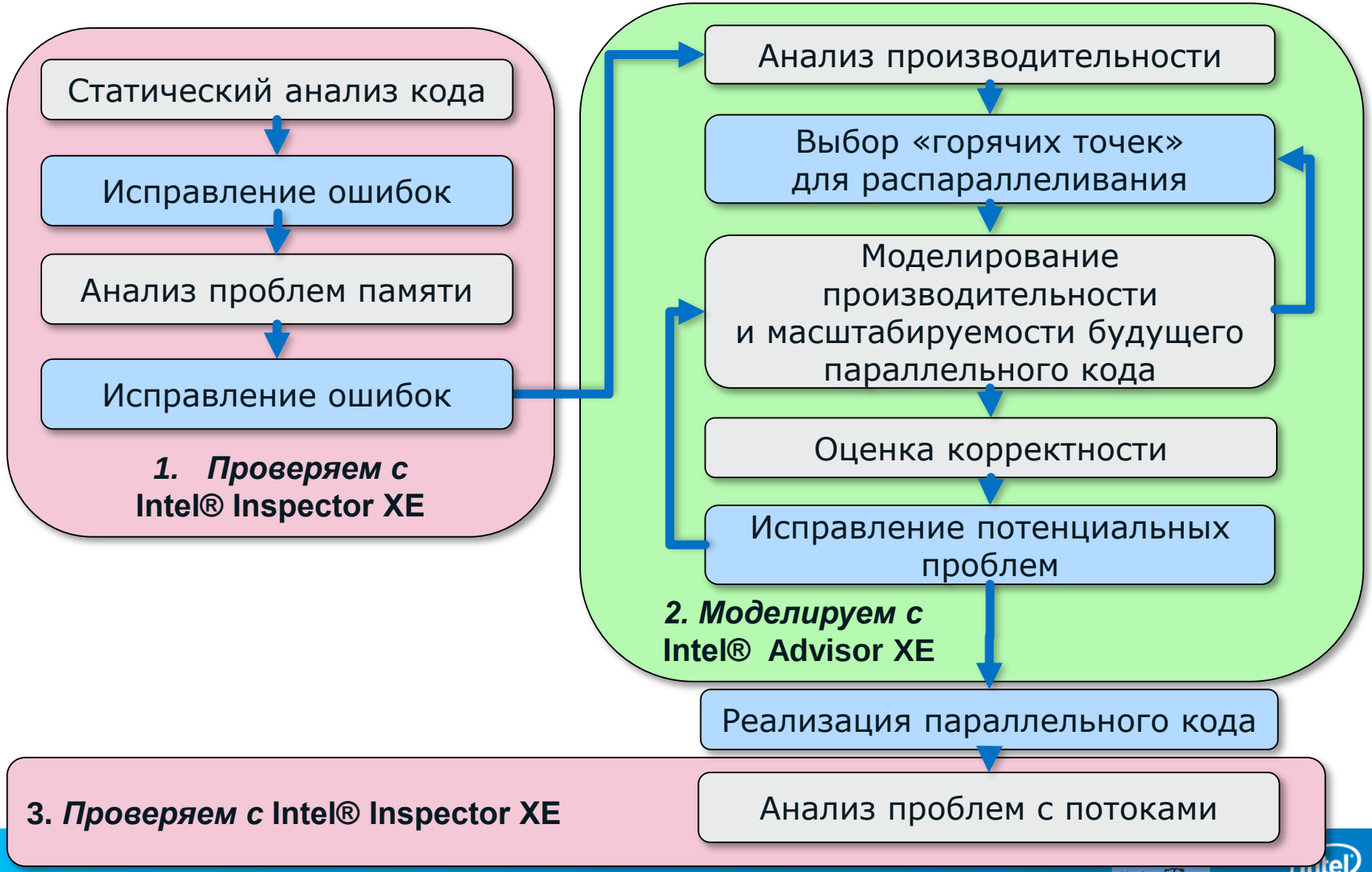
# Предпосылки

Имеет смысл использовать связку Advisor + Inspector, если:

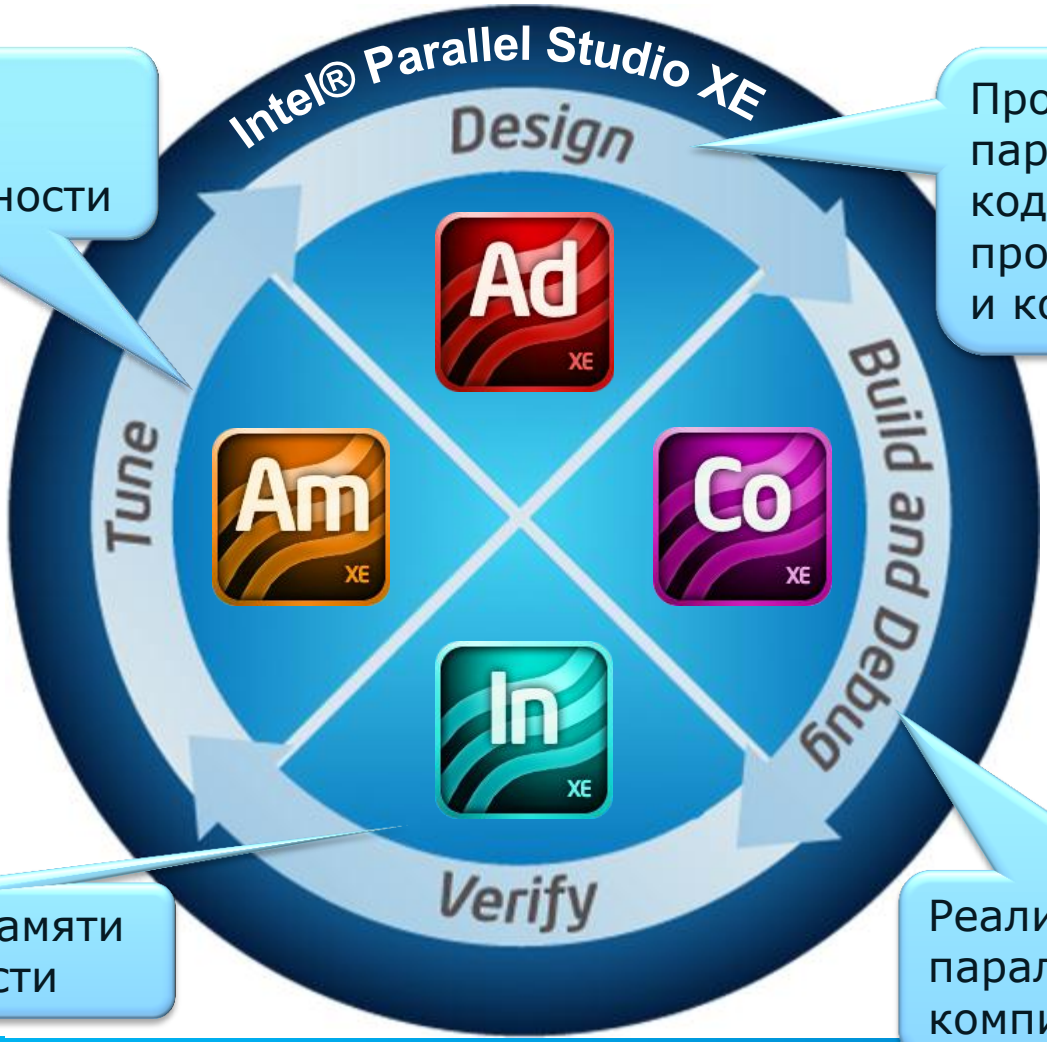
- Есть работающая последовательная программа.
- Есть тесты или эталонные результаты.
- Требуется более полное использование вычислительных ресурсов.
- Требуется ускорить работу программы или обрабатывать больший объем данных за то же время.

# 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




Оценка и оптимизация производительности

Проектирование параллельного кода, оценка его производительности и корректности

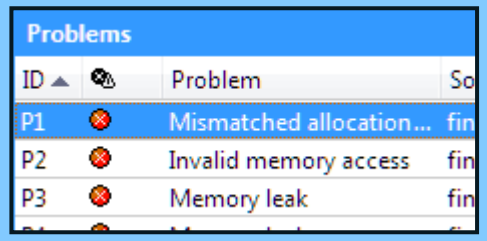
Поиск ошибок памяти и многопоточности

Реализация параллельного кода, компиляция и отладка

# Intel® Inspector XE

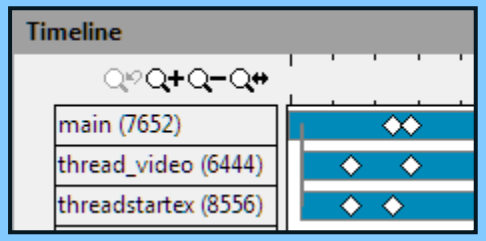
## Где в моём приложении проблемы в …

### памяти



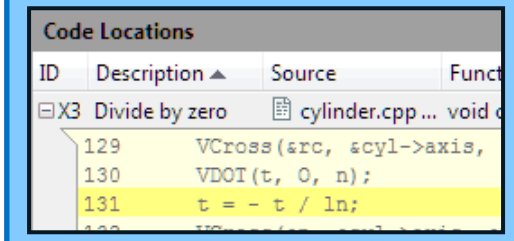

- Некорректный доступ
- Утечки памяти

### потоках




- Гонки данных
- Взаимные блокировки
- Доступ к чужим стекам

### безопасности

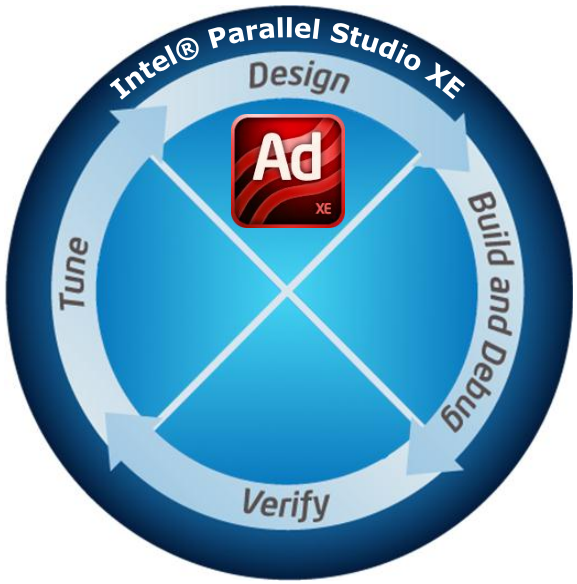



- Переполнение буферов
- Некорректные указатели

- 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

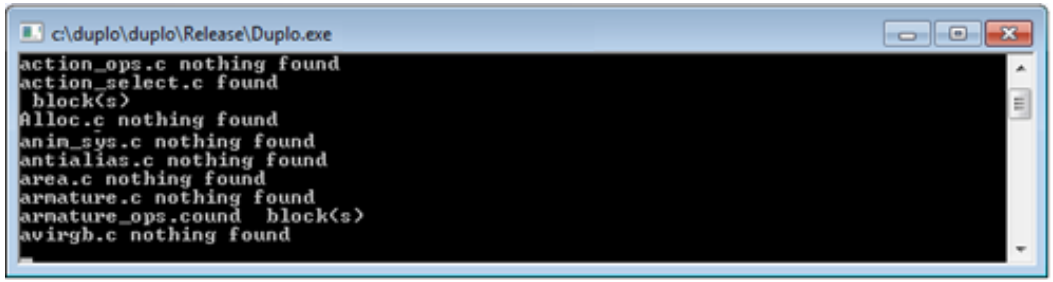
# Что учитывает модель Intel® Advisor XE

- 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/




Поиск повторяющихся блоков текста в группе файлов

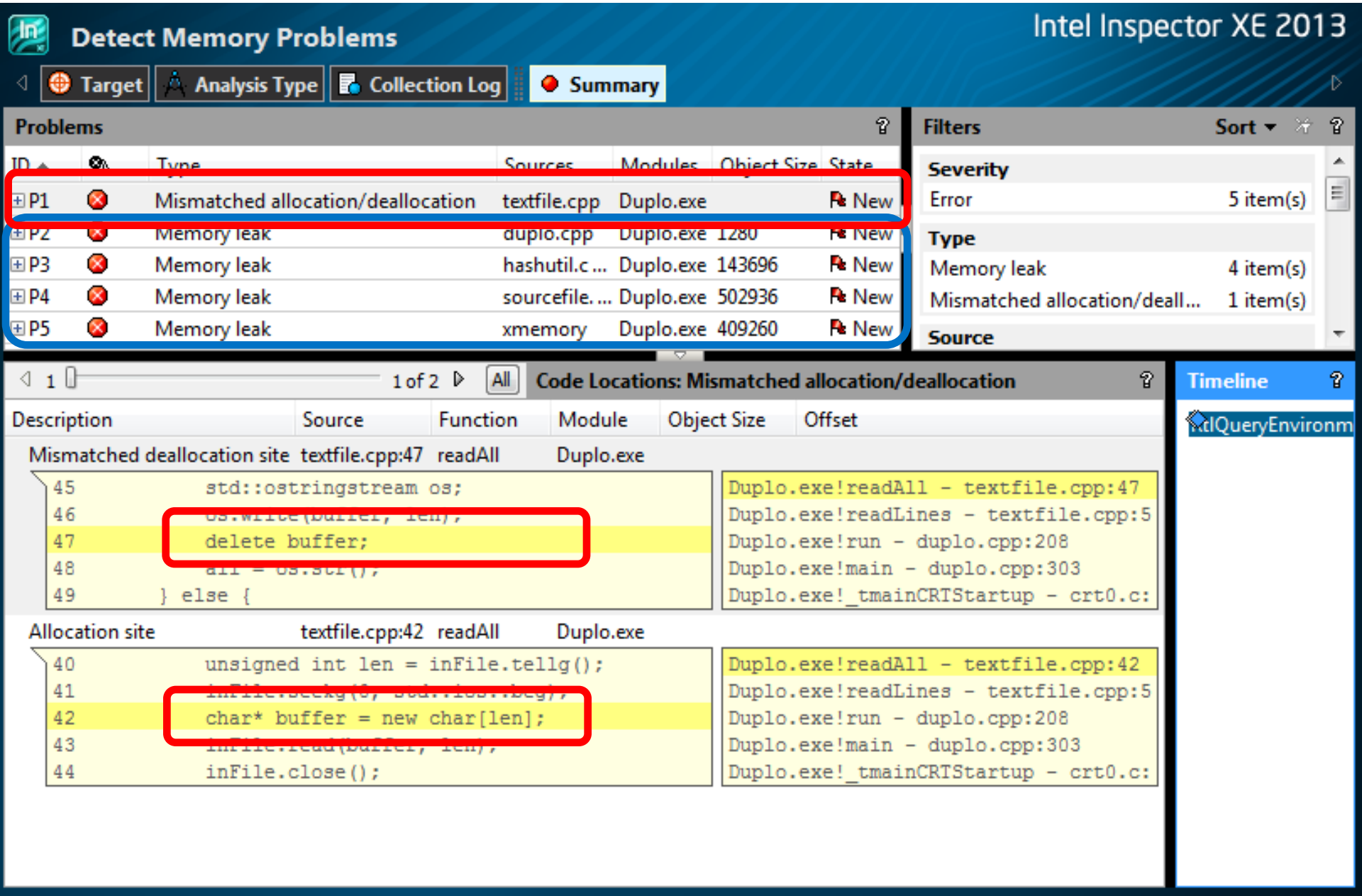
Алгоритм:

- 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

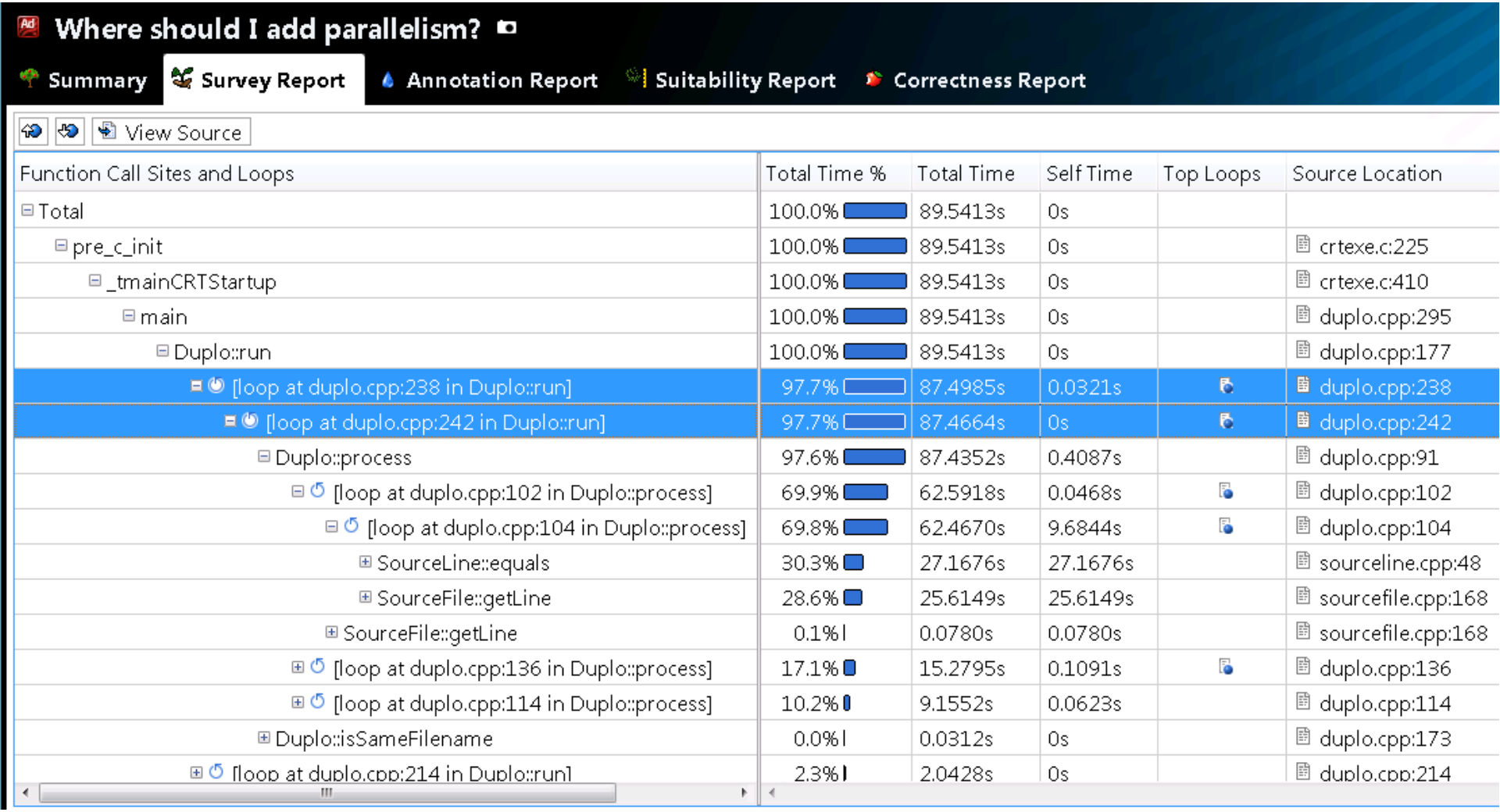
# Особенности *duplo*

- 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



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



# 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

**What are the performance implications of the annotated sites?**

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

**All Sites**

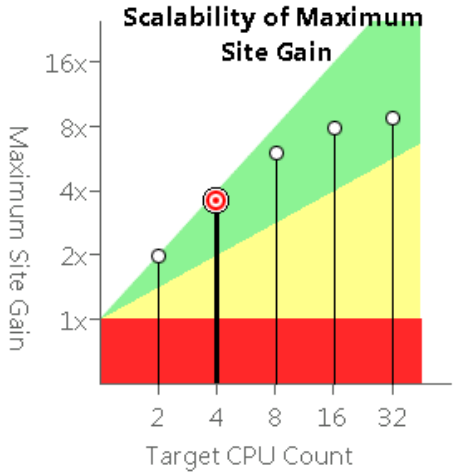
**Maximum Program Gain For All Sites:**

**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**

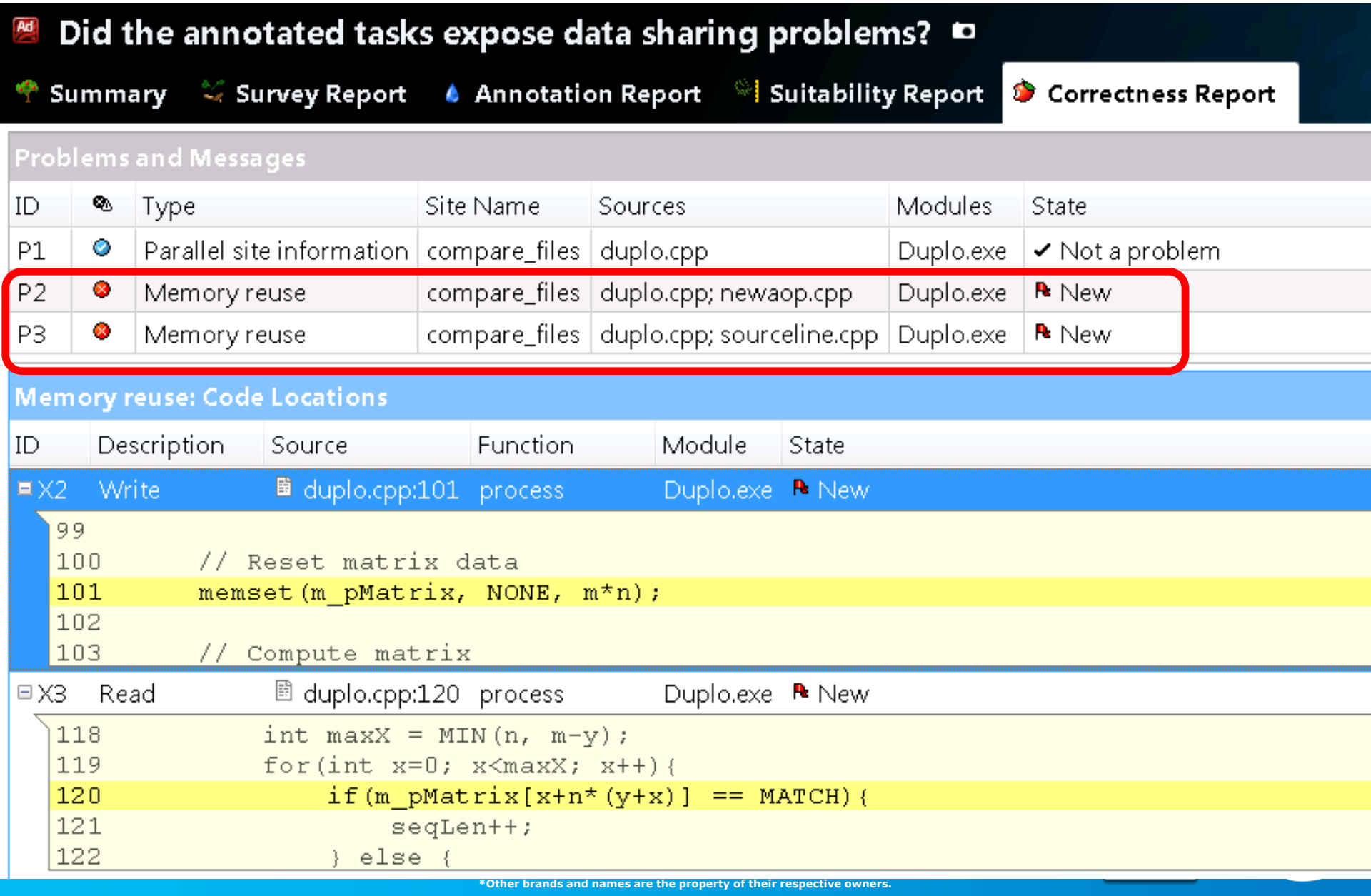



**Changes I will make to this site to improve performance**

| Type of Change | Benefit if Checked | Loss if Unchecked | Recommended |
|---|---|---|---|
| Reduce Site Overhead | | | No |
| Reduce Task Overhead | | | No |
| Reduce Lock Overhead | | | No |
| Reduce Lock Contention | | | No |
| Enable Task Chunking | | | 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



# 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

**What are the performance implications of the annotated sites?**

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

**All Sites**

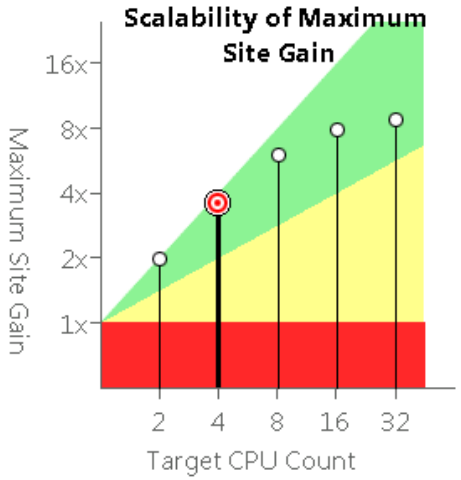
**Maximum Program Gain For All Sites:**

**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**




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

| Type of Change | Benefit if Checked | Loss if Unchecked | Recommended |
|---|---|---|---|
| Reduce Site Overhead | | | No |
| Reduce Task Overhead | | | No |
| Reduce Lock Overhead | | | No |
| Reduce Lock Contention | | | No |
| Enable Task Chunking | | | 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 внутненнего цикла

What are the performance implications of the annotated sites?

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

All Sites

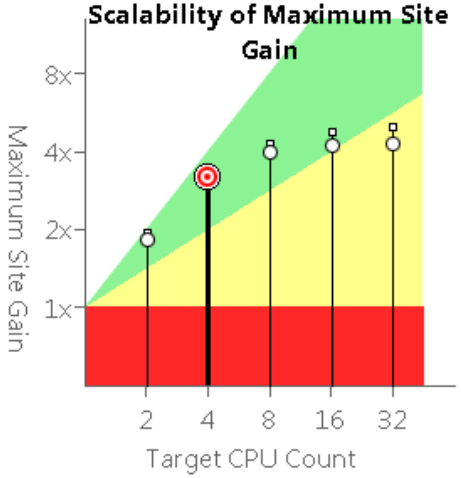
**Maximum Program Gain For All Sites:**

**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




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

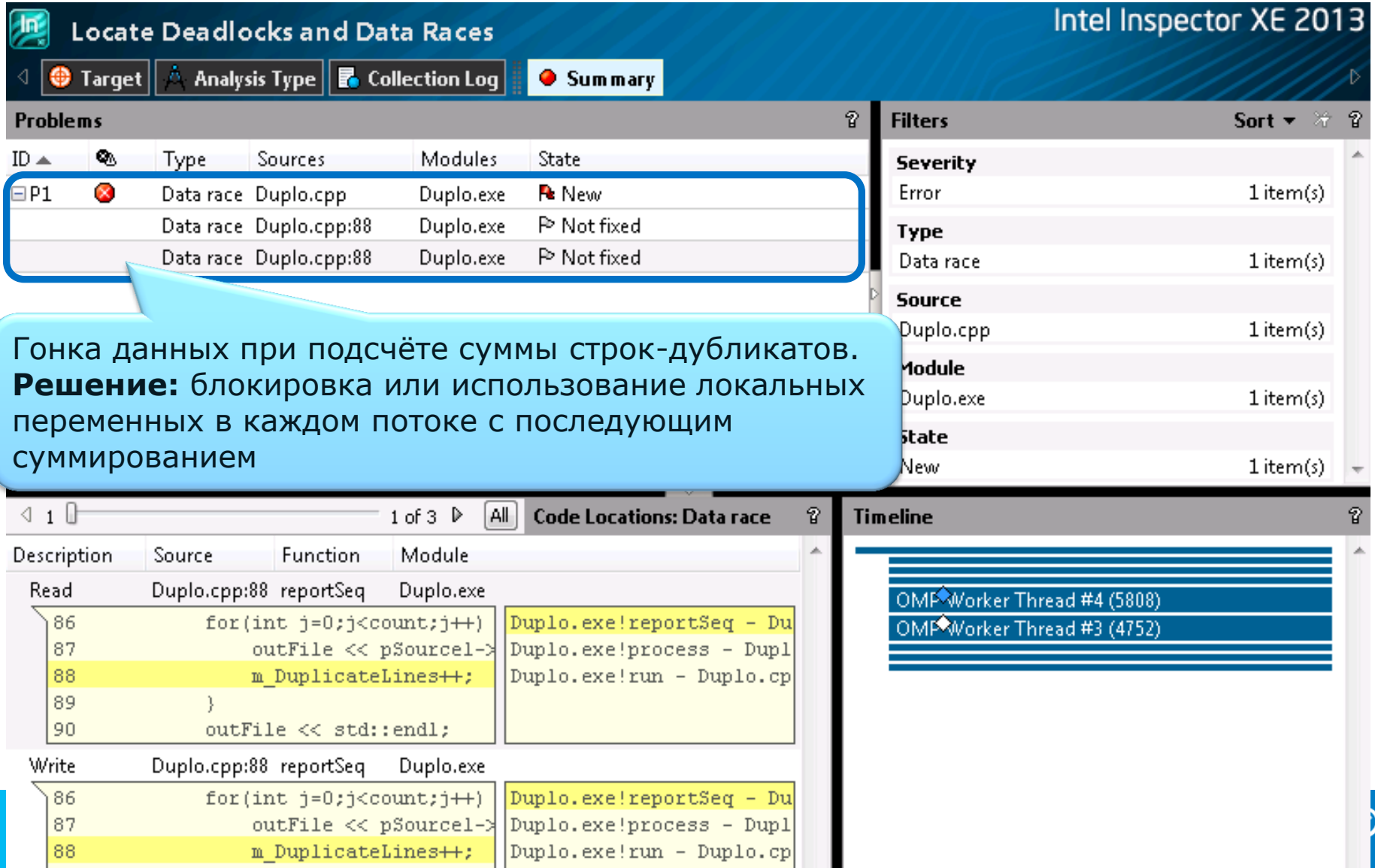
| Type of Change | Benefit if Checked | Loss if Unchecked | Recommended |
|---|---|---|---|
| Reduce Site Overhead | 0.02x | | No |
| Reduce Task Overhead | 0.16x | | No |
| Reduce Lock Overhead | | | No |
| Reduce Lock Contention | | | No |
| Enable Task Chunking | 0.16x | | No |

| 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



# 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

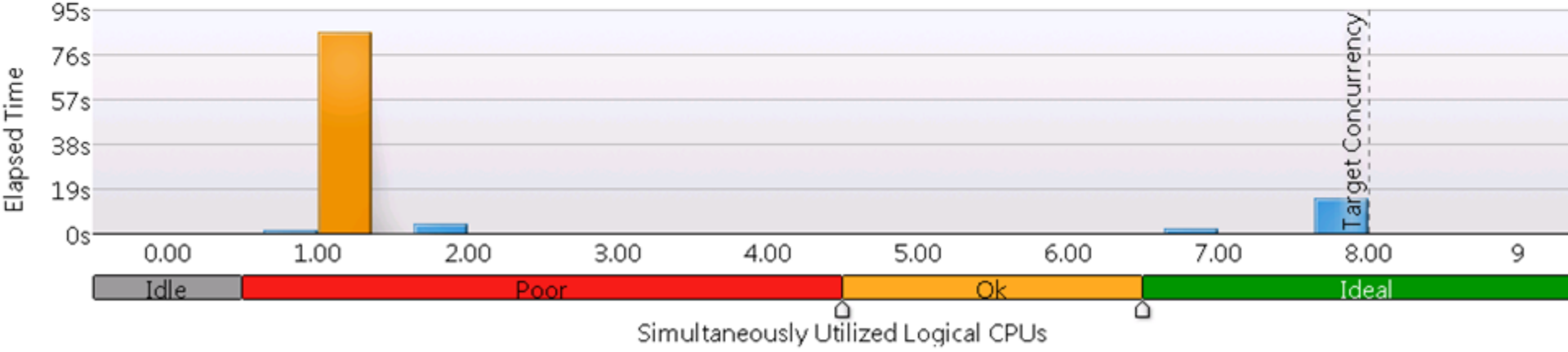
**Elapsed Time: 28.413s - 89.427s = -61.014s**

| | |
|---|---|
| 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
```

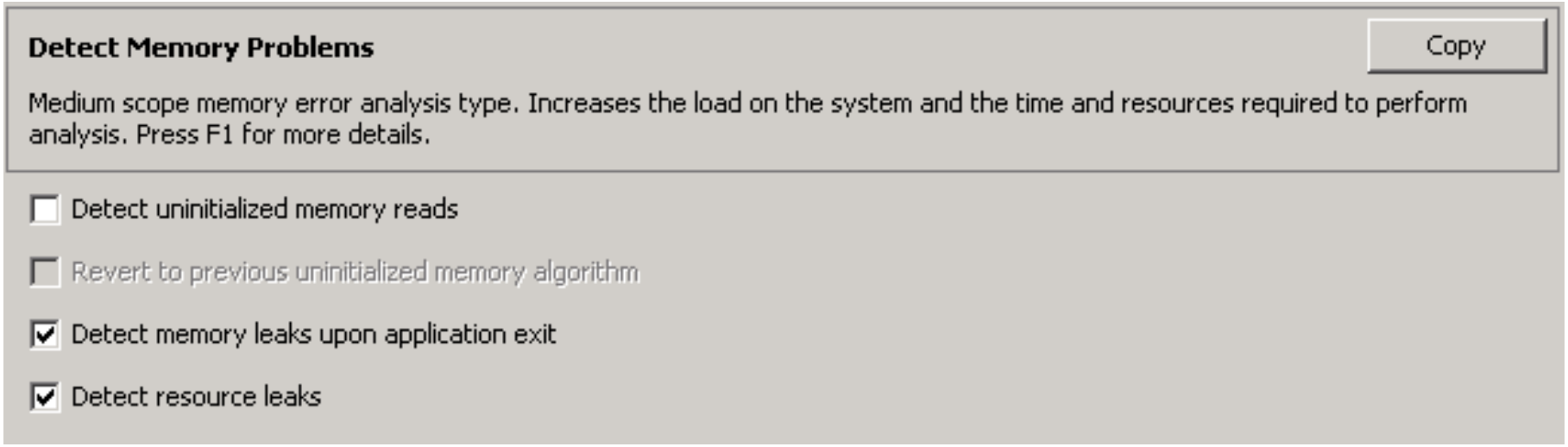
- Только в режиме командной строки
- Может понадобиться ручное копирование результатов с других MPI узлов.

# 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.



# 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

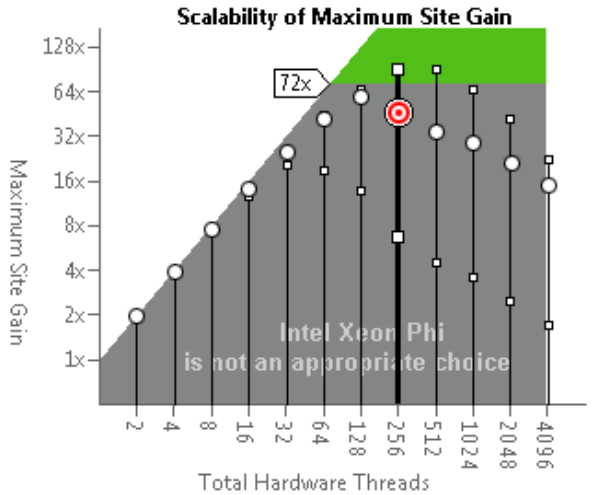
**Maximum Program Gain For All Sites: 5.37x**

Serial time: 22.2974s
Predicted Parallel time: 4.1490s

Target System: Offload to Intel Xeon Phi | Threading Model: Intel TBB | Target CPU Count: 32 | Threads: 256

| 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. ... | ☑ | 1.46x | 73.07s | 0.3193s | 228.81x | 73.07s | 0.31 |
| Site2 | sdp_twosites. ... | ☑ | 2.00x | 142.47s | 3.0859s | 46.17x | 0.2849s | 0.00 |

Site Performance Scalability | Site Details




Loop Iterations (Tasks) Modeling

**Avg. Number of Iterations (Tasks):**
62500
0.008x
0.040x
0.200x
1x (500)
5x
25x
125x

**Avg. Iteration (Task) Duration:**
< 0.0001s
0.008x
0.040x
0.200x
1x (< 0.0001s)
5x
25x
125x

Apply

Runtime Modeling

| Type of Change | Gain Benefit if Checked |
|---|---|
| ☑ Reduce Site Overhead | |
| ☑ Reduce Task Overhead | |
| ☐ Reduce Lock Overhead | |
| ☑ Reduce Lock Contention | |
| ☐ Enable Task Chunking | +15.85x |
| ☑ Enable Ideal Code Vectorization | |

# Advisor: Моделирование пространства итераций

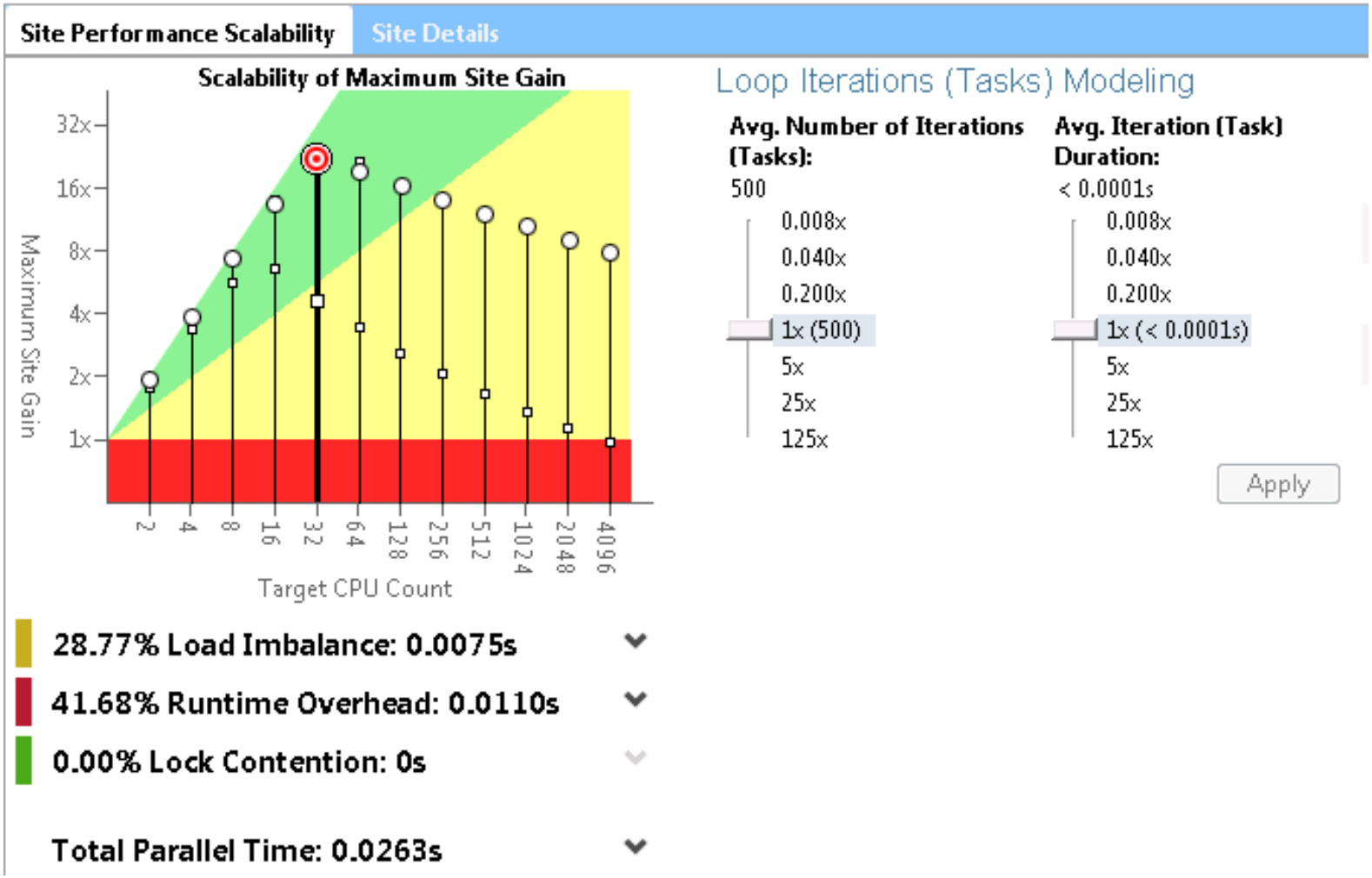
- Как изменится масштабируемость при изменении размеров данных?
  - Изменение количества итераций в циклах
  - Изменение длительности итераций.

- Используйте небольшой набор данных для быстрого анализа, и используйте указанные опции для моделирования большого объема данных.

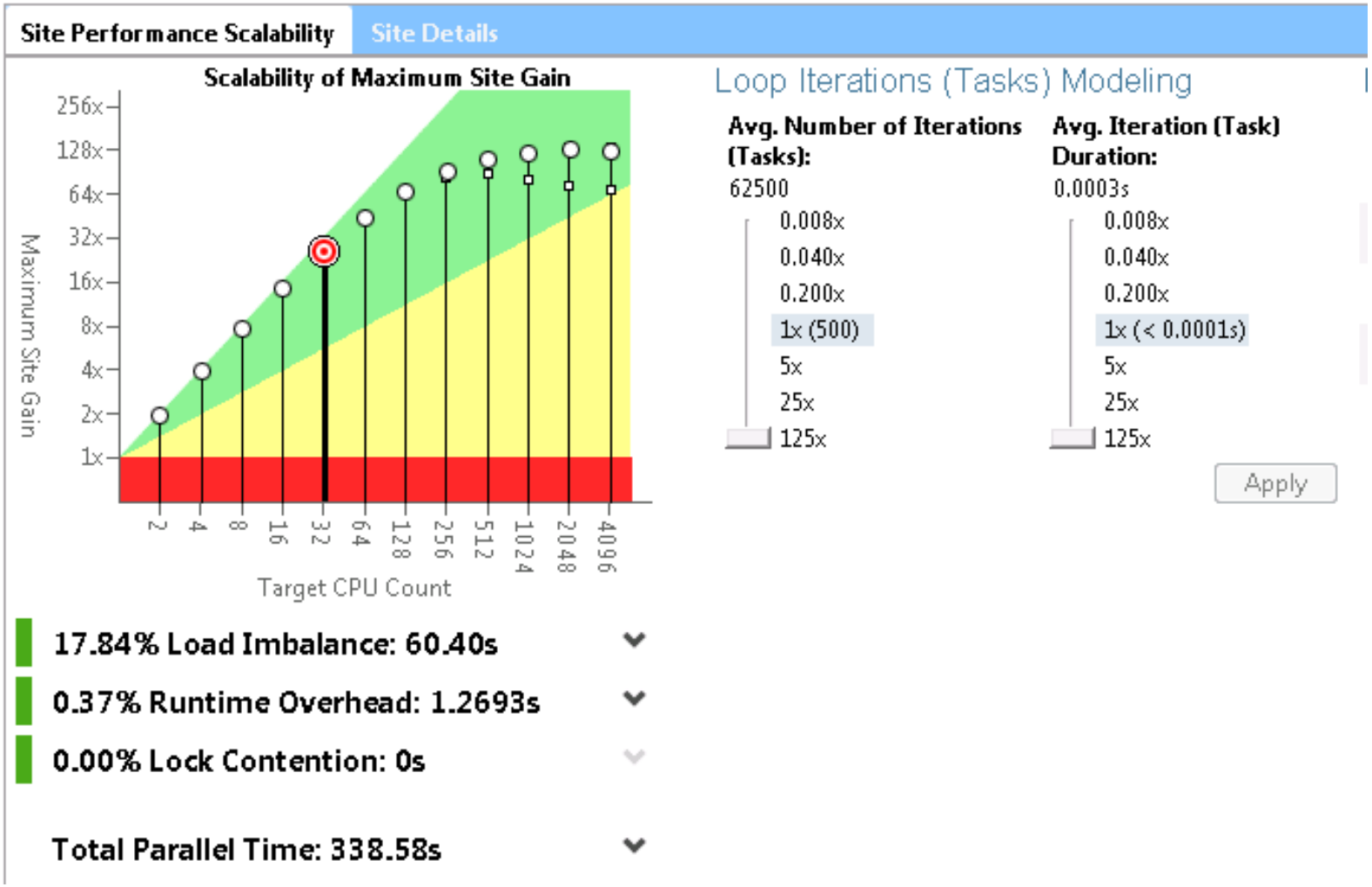
# Advisor: Сводные данные о потерях производительности

- 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
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.

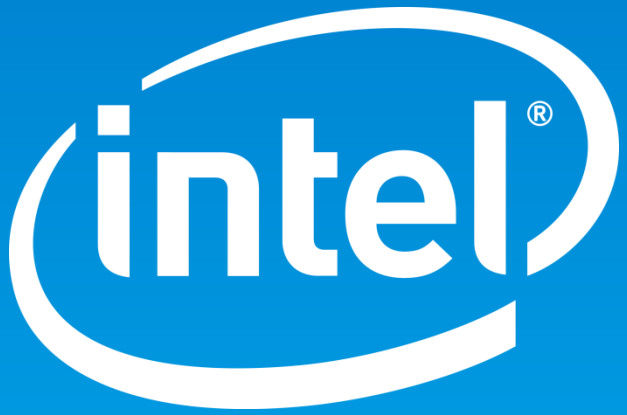
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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.



**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