Конференция CEE-SECR 2009

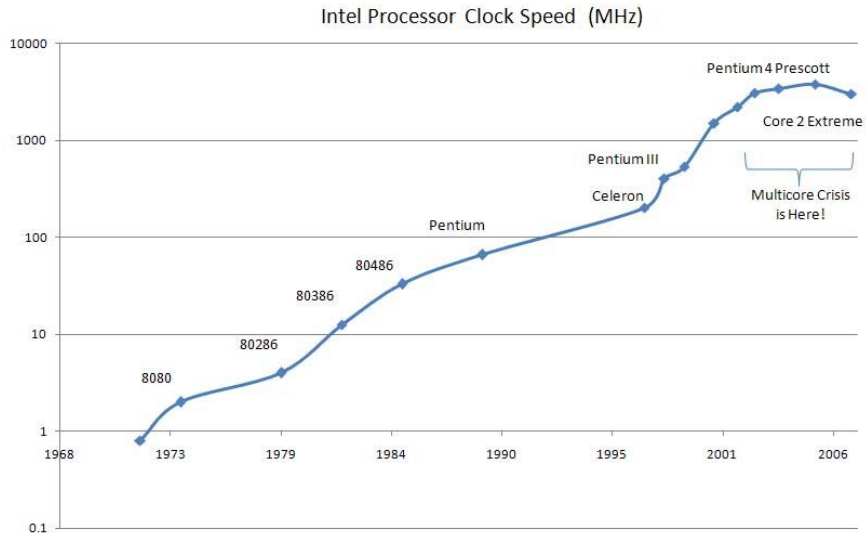
Ранняя диагностика ошибок в параллельных программах

ООО «Системы программной верификации»

Авторы: Карпов Андрей
Рыжков Евгений

www.viva64.com

Параллельность – новая эра в программировании



Рекламные ссылки

Intel® Parallel Studio
Интел предлагает простой и удобный параллелизм в Visual Studio C/C++
www.Intel.com

Разместите здесь свою рекламу »

- Рост вычислительной мощности в ближайшее десятилетие будет осуществляться за счет увеличения количества ядер в микропроцессоре.
- Лидеры рынка активно продвигают решения в области параллельности.
- Переход к параллельному программированию неизбежен для рядовых программистов.

Одна из основных сложностей параллельного программирования – выявление ошибок и отладка

Гейзенбаг - термин, используемый в программировании для описания программной ошибки, которая исчезает или меняет свои свойства при попытке её обнаружения. К гейзенбагам относятся ошибки синхронизации в многопоточных приложениях.

Статический анализ кода

Статический анализ кода (англ. static code analysis) — анализ программного обеспечения, производимый без реального выполнения исследуемых программ. Является альтернативой методологии обзора кода (англ. code review).

- **Преимущества:**
  - Раннее выявление ошибок
  - Независимость от среды исполнения
  - Анализ кода редко получающего управление
  - Хорошая масштабируемость
  - Функция обучения

- **Недостатки:**
  - Невозможность выявления ряда ошибок
  - Ложные срабатывания
  - Высокий уровень самодисциплины

Инструменты статического анализа для выявления ошибок параллельности

- Не следует путать возможность выявления ошибок в параллельном коде и выявление ошибок связанных с использованием параллельности.
- Инструменты статического анализа для выявления ошибок параллельности в начале пути развития.
- Рассмотрим три инструмента: PC-Lint, Intel C++ (“Parallel Lint”), PVS-Studio (VivaMP).

Разные инструменты для разных технологий параллельного программирования

- POSIX threads (Pthreads);
- MPI;
- OpenMP;
- Intel Threading Building Blocks (TBB);
- Windows Threads.

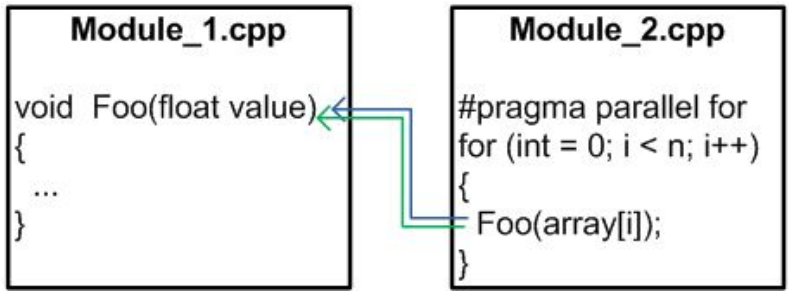
PC-Lint

- PC-Lint – продукт компании Gimpel Software. Не имеет GUI, но его можно самостоятельно интегрировать в Visual Studio 2005/2008 или использовать оболочку Visual Lint.
- В девятой версии Gimpel Software PC-lint реализована диагностика ошибок в параллельных программах, построенных на основе технологии POSIX threads или схожих с ней.

Принцип работы PC-Lint

- Каждый файл обрабатывается отдельно
- Необходимо расставить «подсказки»
  Пример:

```
//lint -sem(f, thread)
void Foo(float value) {
  static int n = SlowFoo();
  /* ... */
}
```



- Поддержка POSIX threads

PC-Lint. Демонстрационный пример

(PC-lint/FlexeLint 9.0 Manual Excerpts)

```
//lint -sem( lock, thread_lock )
//lint -sem( unlock, thread_unlock )
extern int g();
void lock(void), unlock(void);
void f()
{
  lock();
  unlock(); // great
  //-------------
  lock();
  if( g() )
  {
    unlock();
    return; // ok
  }
  unlock(); // still ok
  //-------------
  lock();
  if( g() )
    return; // Warning 454
  unlock();
```

```
if( g() )
  {
    lock();
    unlock();
    unlock(); // Warning 455
    return;
  }
  //-------------
  if( g() )
    lock();
  { // Warning 456
    // do something interesting
  }
  if( g() )
    unlock();
} // Warning 454 and 456
```

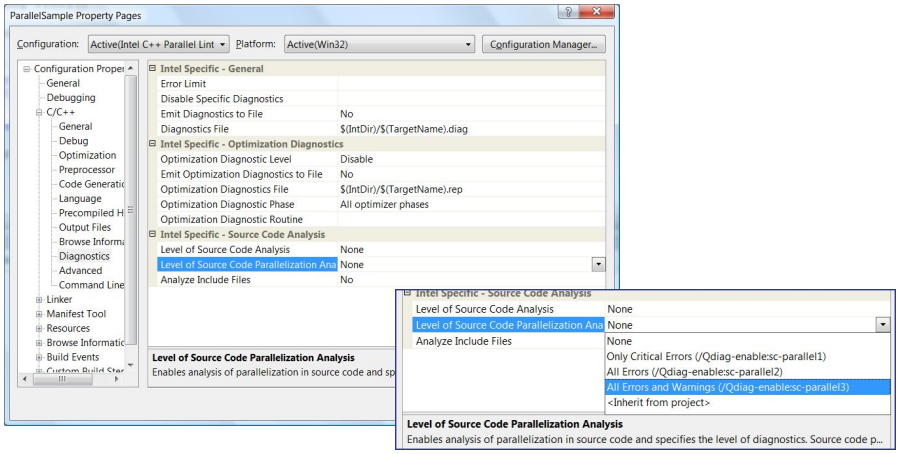
454 – Возврат управления при захваченном мьютекс

455 – Освобождение не захваченного мьютекса

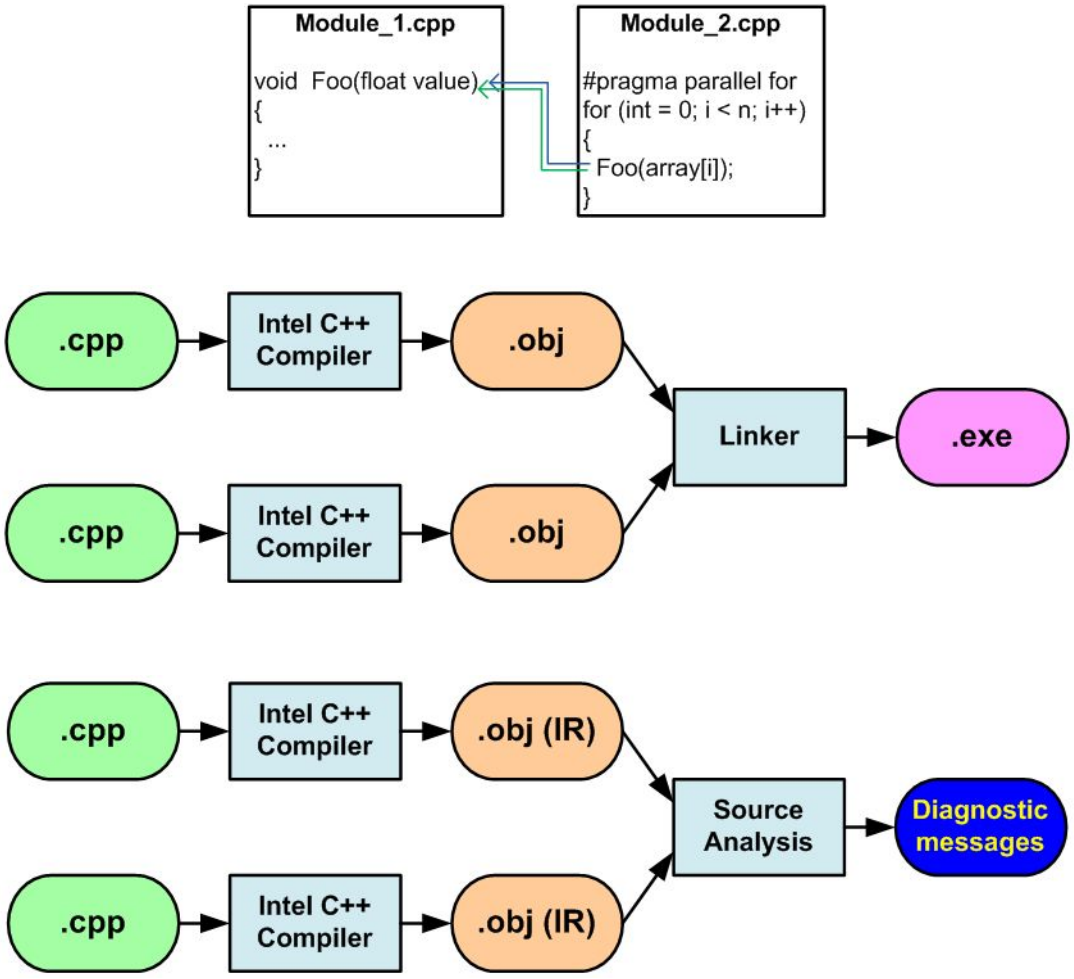
456 – Возможна ошибка синхронизации, так как работа с мьютексом зависит от условия

Intel C++ “Parallel Lint”

- Начиная с версии 11.1 в состав Intel C++ Compiler входит подсистема статического анализа OpenMP кода, получившая название “Parallel Lint”.
- Компилятор Intel C++ может интегрироваться в среду Visual Studio.



Принцип работы Intel C++ (“Parallel Lint”)



Intel C++. Демонстрационный пример

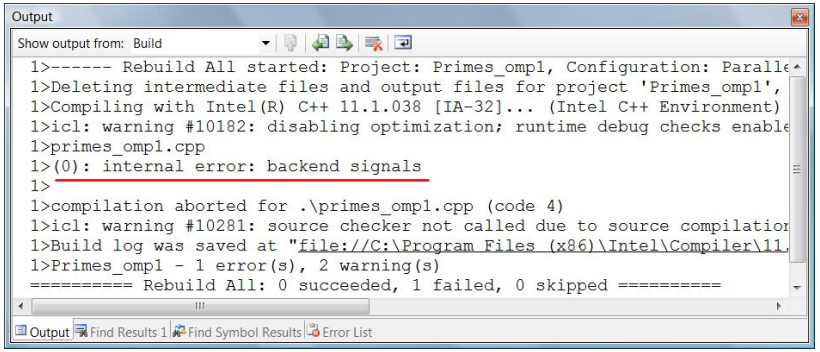
(проект parallel_lint из дистрибутива Intel C++)

```
int main(int argc, char *argv[])
{
      int i, j, limit;
      int start, end;
      int      number_of_primes=0;
      int number_of_41primes=0;
      int number_of_43primes=0;

      start = 1;
      end = 1000000;

      #pragma omp parallel for
      for(i = start; i <= end; i += 2) {
        limit = (int) sqrt((float)i) + 1;
        int prime = 1; /* assume number is prime */
        j = 3;
        while (prime && (j <= limit)) {
              if (i%j == 0) prime = 0;
              j += 2;
        }

        if (prime) {
              number_of_primes++;
              if (i%4 == 1) number_of_41primes++;
              if (i%4 == 3) number_of_43primes++;
        }
      }
      printf("\nProgram Done.\n %d primes found\n",number_of_primes);
      printf("\nNumber of 4n+1 primes found: %d\n",number_of_41primes);
      printf("\nNumber of 4n-1 primes found: %d\n",number_of_43primes);

      return 0;
}
```

Теоретически:

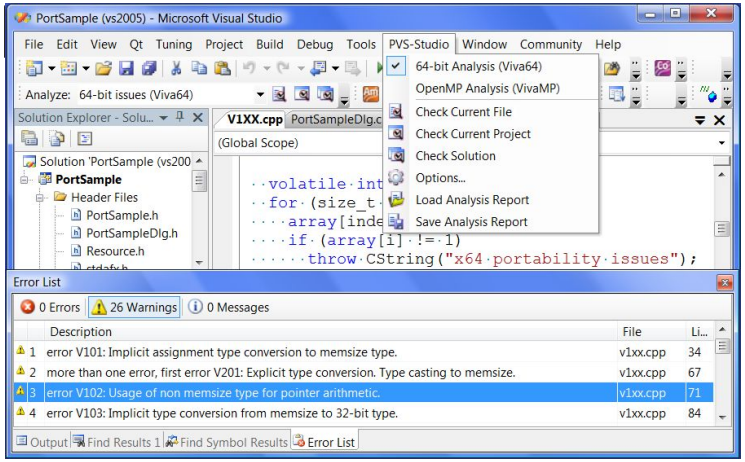
primes_omp1.cpp(70): warning #12246: flow data dependence from (file:primes_omp1.cpp line:70) to (file:primes_omp1.cpp line:73), due to "limit" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(70): warning #12248: output data dependence from (file:primes_omp1.cpp line:70) to (file:primes_omp1.cpp line:70), due to "limit" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12246: flow data dependence from (file:primes_omp1.cpp line:75) to (file:primes_omp1.cpp line:73), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12246: flow data dependence from (file:primes_omp1.cpp line:75) to (file:primes_omp1.cpp line:74), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12246: flow data dependence from (file:primes_omp1.cpp line:75) to (file:primes_omp1.cpp line:75), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12248: output data dependence from (file:primes_omp1.cpp line:75) to (file:primes_omp1.cpp line:72), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12248: output data dependence from (file:primes_omp1.cpp line:75) to (file:primes_omp1.cpp line:75), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(79): warning #12246: flow data dependence from (file:primes_omp1.cpp line:79) to (file:primes_omp1.cpp line:79), due to "number_of_primes" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(79): warning #12248: output data dependence from (file:primes_omp1.cpp line:79) to (file:primes_omp1.cpp line:79), due to "number_of_primes" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(80): warning #12246: flow data dependence from (file:primes_omp1.cpp line:80) to (file:primes_omp1.cpp line: 80), due to "number_of_41primes" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(80): warning #12248: output data dependence from (file:primes_omp1.cpp line:80) to (file:primes_omp1.cpp line:80), due to "number_of_41primes" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(81): warning #12246: flow data dependence from (file:primes_omp1.cpp line:81) to (file:primes_omp1.cpp line:81), due to "number_of_43primes" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(81): warning #12248: output data dependence from (file:primes_omp1.cpp line:81) to (file:primes_omp1.cpp line:81), due to "number_of_43primes" may lead to incorrect program execution in parallel mode

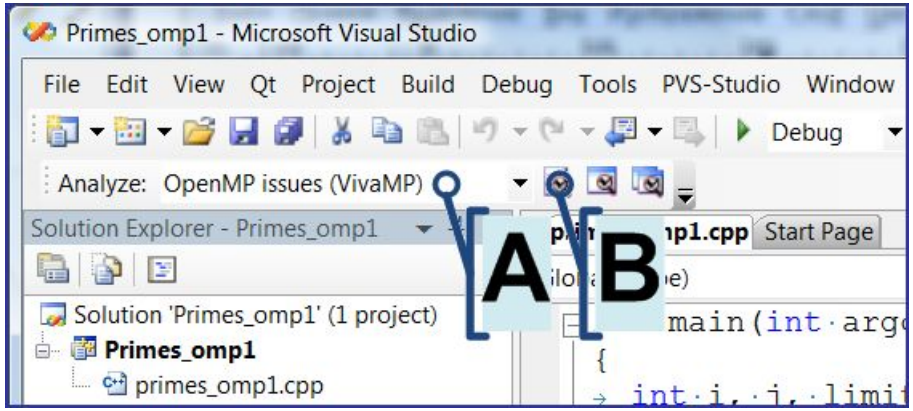
Практически:



PVS-Studio (VivaMP)

- VivaMP – статический анализатор кода для проверки OpenMP программ. Входит в состав PVS-Studio.
- Интегрируется в Visual Studio 2005/2008. Справочная система интегрируется в





Принцип работы PVS-Studio (VivaMP)

- Технология OpenMP способствует возникновению локальных распараллеленных участков кода. В результате большую часть ошибок можно выявить обрабатывая каждый файл отдельно.
- Не требуется какой либо адаптации проекта или специальных настроек.
- Фильтрация сообщений в режиме on-the-fly. Сохранение и загрузка списка предупреждений.

PVS-Studio. Демонстрационный пример

```
#pragma omp parallel for
for (size_t i = 0; i != n; ++i) {
  float *array =
    new float[10000];
  delete [] array;
}
```

Анализатор VivaMP диагностирует в данном коде ошибку: “V1302. The ‘new’ operator cannot be used outside of a try..catch block in a parallel section.” Ошибка связана с бросанием исключения из параллельного блока.

```
  pragma omp parallel
  {
    static int st = SlowFoo();
    ...
  }
```

Статическая переменная начнет процесс инициализации сразу в нескольких потоках, что может привести к неопределенному результату. В отличии от распараллеливания с использованием POSIX Threads, здесь анализатор легко может понять, что код находится в параллельной секции.

Сравнение Intel C++ (“Parallel Lint”) и PVS-Studio (VivaMP)

Сравнение на базе parallel_lint из дистрибутива Intel C++ (всего 6 ошибок):

```
primes_omp1.cpp(70): warning #12246: flow data dependence from (file:primes_omp1.cpp line:70) to
   (file:primes_omp1.cpp line:73), due to "limit" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(70): warning #12248: output data dependence from (file:primes_omp1.cpp line:70) to
   (file:primes_omp1.cpp line:70), due to "limit" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12246: flow data dependence from (file:primes_omp1.cpp line:75) to
   (file:primes_omp1.cpp line:73), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12246: flow data dependence from (file:primes_omp1.cpp line:75) to
   (file:primes_omp1.cpp line:74), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12246: flow data dependence from (file:primes_omp1.cpp line:75) to
   (file:primes_omp1.cpp line:75), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12248: output data dependence from (file:primes_omp1.cpp line:75) to
   (file:primes_omp1.cpp line:72), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(75): warning #12248: output data dependence from (file:primes_omp1.cpp line:75) to
   (file:primes_omp1.cpp line:75), due to "j" may lead to incorrect program execution in parallel mode
primes_omp1.cpp(79): warning #12246: flow data dependence from (file:primes_omp1.cpp line:79) to
   (file:primes_omp1.cpp line:79), due to "number_of_primes" may lead to incorrect program execution in
parallel mode
primes_omp1.cpp(79): warning #12248: output data dependence from (file:primes_omp1.cpp line:79) to
   (file:primes_omp1.cpp line:79), due to "number_of_primes" may lead to incorrect program execution in
parallel mode
primes_omp1.cpp(80): warning #12246: flow data dependence from (file:primes_omp1.cpp line:80) to
   (file:primes_omp1.cpp line: 80), due to "number_of_41primes" may lead to incorrect program
execution in parallel mode
primes_omp1.cpp(80): warning #12248: output data dependence from (file:primes_omp1.cpp line:80) to
   (file:primes_omp1.cpp line:80), due to "number_of_41primes" may lead to incorrect program execution
in parallel mode
primes_omp1.cpp(81): warning #12246: flow data dependence from (file:primes_omp1.cpp line:81) to
   (file:primes_omp1.cpp line:81), due to "number_of_43primes" may lead to incorrect program execution
in parallel mode
primes_omp1.cpp(81): warning #12248: output data dependence from (file:primes_omp1.cpp line:81) to
   (file:primes_omp1.cpp line:81), due to "number_of_43primes" may lead to incorrect program execution
in parallel mode
```

Parallel Lint (Найдено 6 ошибок из 6. Количество предупреждений: 13)

```
Warning  1  error V1205: Data race risk. Unprotected concurrent operation with the
'limit' variable.  r:\primes\primes_omp1\primes_omp1.cpp  69
Warning  2  error V1205: Data race risk. Unprotected concurrent operation with the
'j' variable.  r:\primes\primes_omp1\primes_omp1.cpp  71
Warning  3  error V1205: Data race risk. Unprotected concurrent operation with the
'j' variable.  r:\primes\primes_omp1\primes_omp1.cpp  74
Warning  4  error V1205: Data race risk. Unprotected concurrent operation with the
'number_of_primes' variable.  r:\primes\primes_omp1\primes_omp1.cpp  78
Warning  5  error V1205: Data race risk. Unprotected concurrent operation with the
'number_of_41primes' variable.  r:\primes\primes_omp1\primes_omp1.cpp  79
Warning  6  error V1205: Data race risk. Unprotected concurrent operation with the
'number_of_43primes' variable.  r:\primes\primes_omp1\primes_omp1.cpp  80
```

VivaMP (Найдено 6 ошибок из 6. Количество предупреждений: 6)

Пункт сравнения	PVS-Studio (VivaMP)	Intel C++ ("Parallel Lint")
1. Забытая директива parallel.	Да	Нет
2. Забытая директива omp.	Да	Да
3. Забытая директива for.	Да	Нет
4. Множественное распараллеливание циклов.	Да	Нет
5. Забытые order директивы.	Да	Нет
6. Не подключен заголовочный файл <omp.h>.	Да	Нет
7. omp_set_num_threads внутри параллельной секции.	Да	Нет
8. Нечетность количества блокировок и разблокировок.	Да	Нет
9. omp_get_num_threads в арифметических операциях.	Да	Нет
10. omp_set_nested внутри параллельной секции.	Да	Нет
11. Параллельное использование общих ресурсов.	Да	Да
12. flush для указателя.	Да	Нет
13. Использование threadprivate.	Да	Нет
14. Состояния гонки. Статические переменные.	Да	Да
15. Состояние гонки. Общая переменная.	Да	Да
16. Состояние гонки. Доступ к массиву.	Нет	Да
17. Состояние гонки. Опасная функция.	Да	Нет
18. Состояние гонки. Модификация объекта.	Да	Нет
19. Приватные указатели.	Да	Нет
20. Неосторожное применение lastprivate.	Да	Нет
21. Бессмысленный flush. Локальные переменные.	Да	Нет
22. Бессмысленный flush. Присутсвует неявный flush.	Нет	Нет
23. Исключения. Явный throw.	Да	Нет
24. Исключения. Оператор new.	Да	Нет
25. Исключения. Функции.	Да	Нет
26. Забытая инициализация. omp_init_lock.	Нет	Нет
27. Забытая инициализация. Локальные переменные.	Нет	Да
28. Забытая инициализация после параллельного региона.	Нет	Да
29. Отсутствие конструктора копирования.	Нет	Нет
30. Неэффективное использование критических секций.	Нет	Нет
31. Бессмысленная защита локальных переменных.	Нет	Нет
32. Лишний reduction.	Нет	Да
Процент покрытия известных автору некорректных паттернов параллельного кода:	**72 %**	**25 %**

Некорректная таблица сравнения Intel C++ ("Parallel Lint") и PVS-Studio (VivaMP)

Вывод:

Parallel Lint удобен тем, кто уже использует Intel C++ для сборки своих проектов

PVS-Studio (VivaMP) подходит для разработчиков, использующих Visual C++

Преимущества от ранней диагностики ошибок

	Время обнаружения дефекта				
Время внесения дефекта	Выработка требований	Проектирование архитектуры	Конструирование (кодирование)	Тестирование	После выпуска ПО
Выработка требований	1	3	5-10	10	10-100
Проектирование архитектуры	-	1	10	15	25-100
Конструирование (кодирование)	-	-	1	10	10-25

Средняя стоимость исправления дефектов в зависимости от времени их внесения и обнаружения (данные для таблицы взяты из книги С. Макконнелла "Совершеннй Код").

Приложение. Дополнительные примеры

void Foo() { **#pragma omp for** **for (int i = 0; I < n; ++i)** **…** **}**	**Цикл, вопреки ожиданиям программиста, будет выполняться только одним потоком.**

#pragma single
{
…
}
	Забытая директива omp. И хотя компилятор VC++ выдаст сообщение "warning C4068: unknown pragma", это предупреждение может быть проигнорирована в больших сложных проектах, или быть вообще отключено.

#pragma omp parallel
{
omp_set_num_threads(2);
…
}
	Количество потоков нельзя переопределять внутри параллельной секции. Это приводит к ошибке во время выполнения программы и ее аварийному завершению.

Приложение. Дополнительные примеры

int a = 0; **#pragma omp parallel for num_threads(4)** **for (int i = 0; i < 100000; i++) {** ** a++;** **}**	**Состояние гонки возникает тогда, когда несколько потоков многопоточного приложения пытаются одновременно получить доступ к данным, причем хотя бы один поток выполняет запись. Состояния гонки могут давать непредсказуемые результаты, и зачастую их сложно выявить. Иногда последствия состояния гонки проявляются только через большой промежуток времени и в совсем другой части приложения. Кроме того, ошибки такого рода невероятно сложно воспроизвести повторно.**

int a = 1;
#pragma omp parallel for private(a)
for (int i = 10; i < 100; ++i) {
  #pragma omp flush(a);
  ...
}
	Неэффективным следует считать использование директивы flush для локальных переменных (объявленных в параллельной секции), а также переменных помеченных как threadprivate, private, lastprivate, firstprivate.
	Директива flush не имеет для перечисленных переменных смысла, так как эти переменные всегда содержат актуальные значения. И дополнительно снижает производительность кода.

Дополнительная информация

Статьи по тематике презентации:

- Андрей Карпов. Что такое "Parallel Lint"? http://www.viva64.com/art-3-1-17137191.html
- Алексей Колосов, Евгений Рыжков, Андрей Карпов. 32 подводных камня OpenMP при программировании на Си++ http://www.viva64.com/art-3-1-464379766.html

Доклад подготовлен сотрудниками компании ООО «СиПроВер»
Компания СиПроВер занимается разработкой программного обеспечения в области анализа исходного кода программ. Основные направления наших работ: верификация программ, статический анализ кода, развитие открытой библиотеки разбора и анализа Си/Си++ кода, создание инструментария для тестирования программных продуктов.

Контактная информация:
300027, Россия, Тула, Металлургов 70-1-88.
Web: www.viva64.com
E-mail: support@viva64.com
Телефон: +7 (4872) 38-59-95
Рабочее время: 09:00 – 18:00 (GMT +3:00)