

Профилирование программ

# Профилирование

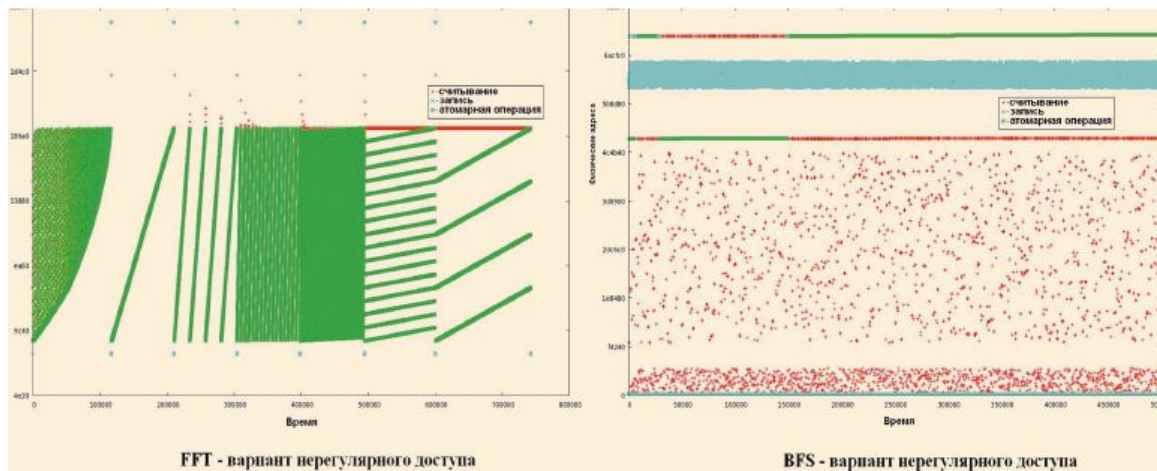
- Профилирование – сбор характеристик работы программы
- Цели профилирования
  - Оценка эффективности работы программы или вычислительной системы
  - Определение критических участков программы (hotspots) или компонентов вычислительной системы

# Результаты профилирования

- Характеристики работы программы
  - Путь исполнения (покрытие кода, дерево вызовов подпрограмм, количество вызовов подпрограмм, ...)
  - Время работы участка программы (функции, оператора языка, машинной команды, ...)
  - Количество событий, произошедших в системе (исполненных команд, промахов кэша, неправильно предсказанных переходов, ...)
  - Распределение времени и событий по коду программы
  - Загрузка ресурсов системы (процессор, память, диск, сеть, ...)
- Анализ характеристик
  - Критические участки программы (hotspots, critical path)
  - Оценка достигнутой производительности, причины потери, варианты улучшения

# Результаты профилирования

- Профиль обращений к памяти



Пример профиля работы с памятью для задачи DIS-класса (SpMV)

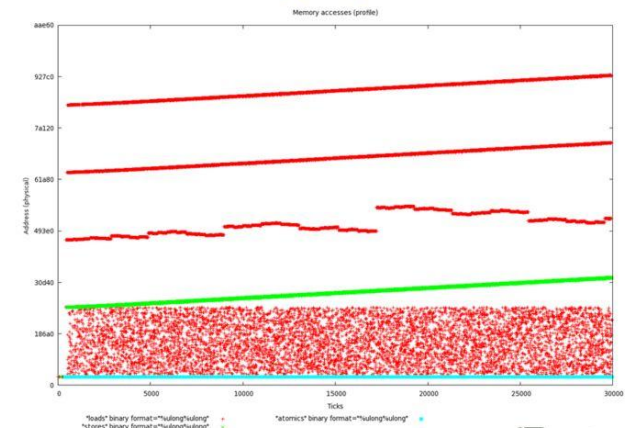
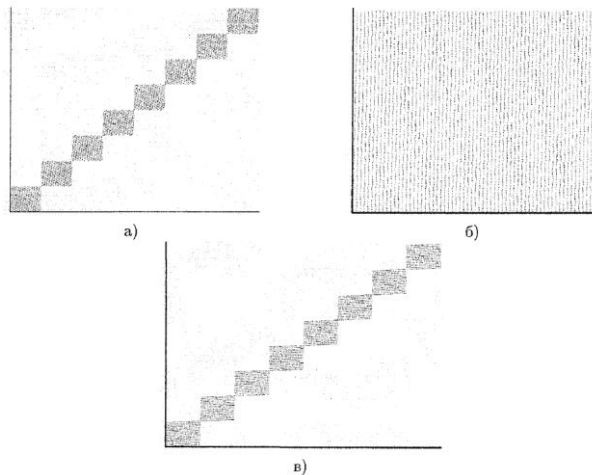


Рис. Профиль обращений к памяти для теста SpMV (A\*p=q)



# Способы профилирования

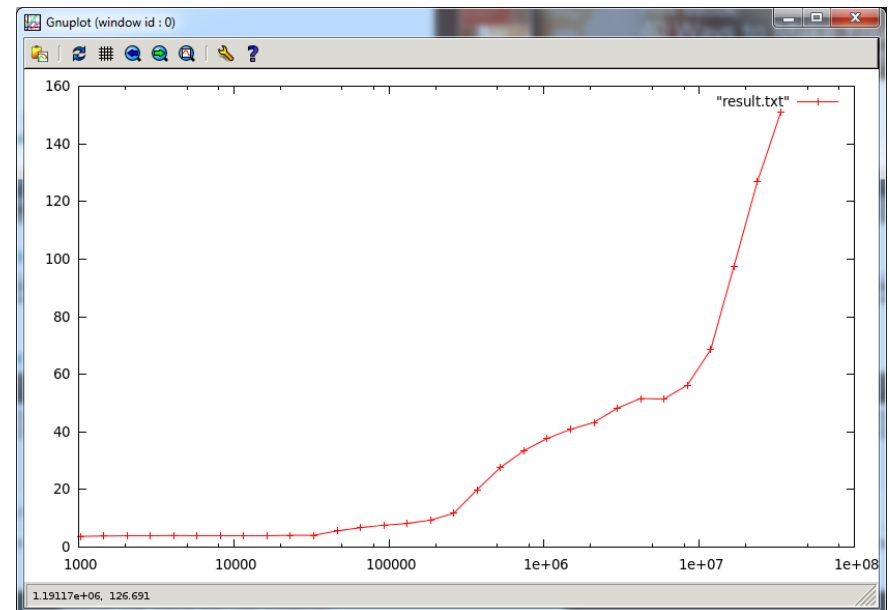
- Инструментирование кода – добавление кода для сбора статистики
  - Просто реализуется
  - Сильное влияние на характеристики программы
  - Позволяет собрать информацию о пути исполнения, времени работы участка кода
- Обработка прерываний – код профилировщика вызывается при срабатывании определенных прерываний
  - Меньшее влияние на характеристики программы
  - Позволяет определять низкоуровневые характеристики системы (аппаратные таймеры, счетчики событий), распределение событий по коду программы
- Эмуляция выполнения программы
  - Большое время профилирования
  - Не учитывается «реальная» ситуация
  - Повторяемость результата
  - Позволяет получить любую информацию

# Профилировщики

- gprof, gcov
- Oprofile  
(<http://oprofile.sourceforge.net>)
- Intel Vtune Amplifier  
(<http://software.intel.com/en-us/intel-vtune-amplifier-xe/>)
- AMD CodeAnalyst Performance Analyzer  
(<http://developer.amd.com/tools-and-sdks/heterogeneous-computing/archived-tools/amd-codeanalyst-performance-analyzer/>)
- Dtrace  
(<http://dtrace.org>)
- Valgrind  
(<http://valgrind.org>)
- ...

# Тестируемая программа

- Программа: определение времени доступа к элементам массива в случайном порядке в зависимости от размера массива
- Этапы работы:
  - Заполнение массива
    - $a[j] = \text{rand}()$
  - Обход массива
    - $\text{for } (i=\dots) j = A[j]$



# gprof

- Компиляция программы
  - `$g++ -O1 -g -pg -o cache cache.c`
  - Создаётся файл: `cache`
- Запуск программы
  - `$/cache`
  - Создается файл: `gmon.out`
- Получение профиля программы и графа вызовов
  - `$gprof ./cache >cache.profile.txt`
  - Создается файл: `cache.profile.txt`
- Получение аннотированного листинга
  - `$gprof -A cache >cache.source.txt`
  - Создается файл: `cache.c.gcov`



# gcov

- Компиляция программы
  - `$g++ -O1 --coverage -o cache cache.c`
  - Создаются файлы: `cache`, `cache.gcn`
- Запуск программы (много раз)
  - `./cache`
  - Создаётся файл: `cache.gcda`
- Получение аннотированного листинга
  - `$gcov -b cache.c`
  - Создаётся файл: `cache.c.gcov`

# Оптимизация в gcc с помощью профилирования

- Компиляция программы
  - `$g++ -O1 -fprofile-generate -o cache cache.c`
  - Создаётся файл: `cache`
- Запуск программы (много раз)
  - `./cache`
  - Создаётся файл: `cache.gcda`
- Компиляция программы с оптимизацией
  - `$ g++ -O1 -fprofile-use -o cache_opt cache.c`
  - Создается файл: `cache_opt`

# Оптимизация в gcc с помощью

```
$time ./cache
```

```
real      0m40.409s
```

```
user      0m40.323s
```

```
sys       0m0.016s
```

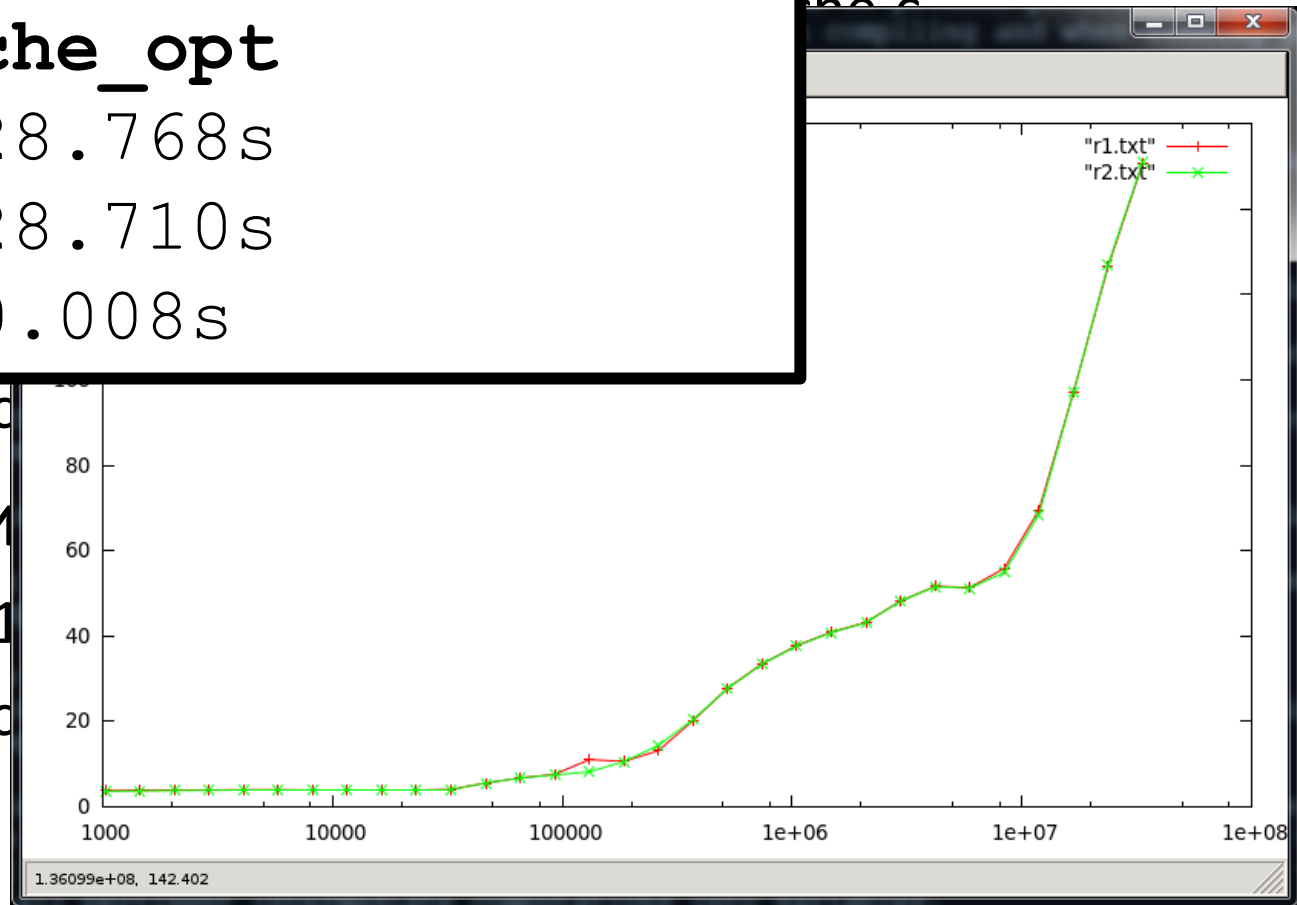
```
$time ./cache_opt
```

```
real      0m28.768s
```

```
user      0m28.710s
```

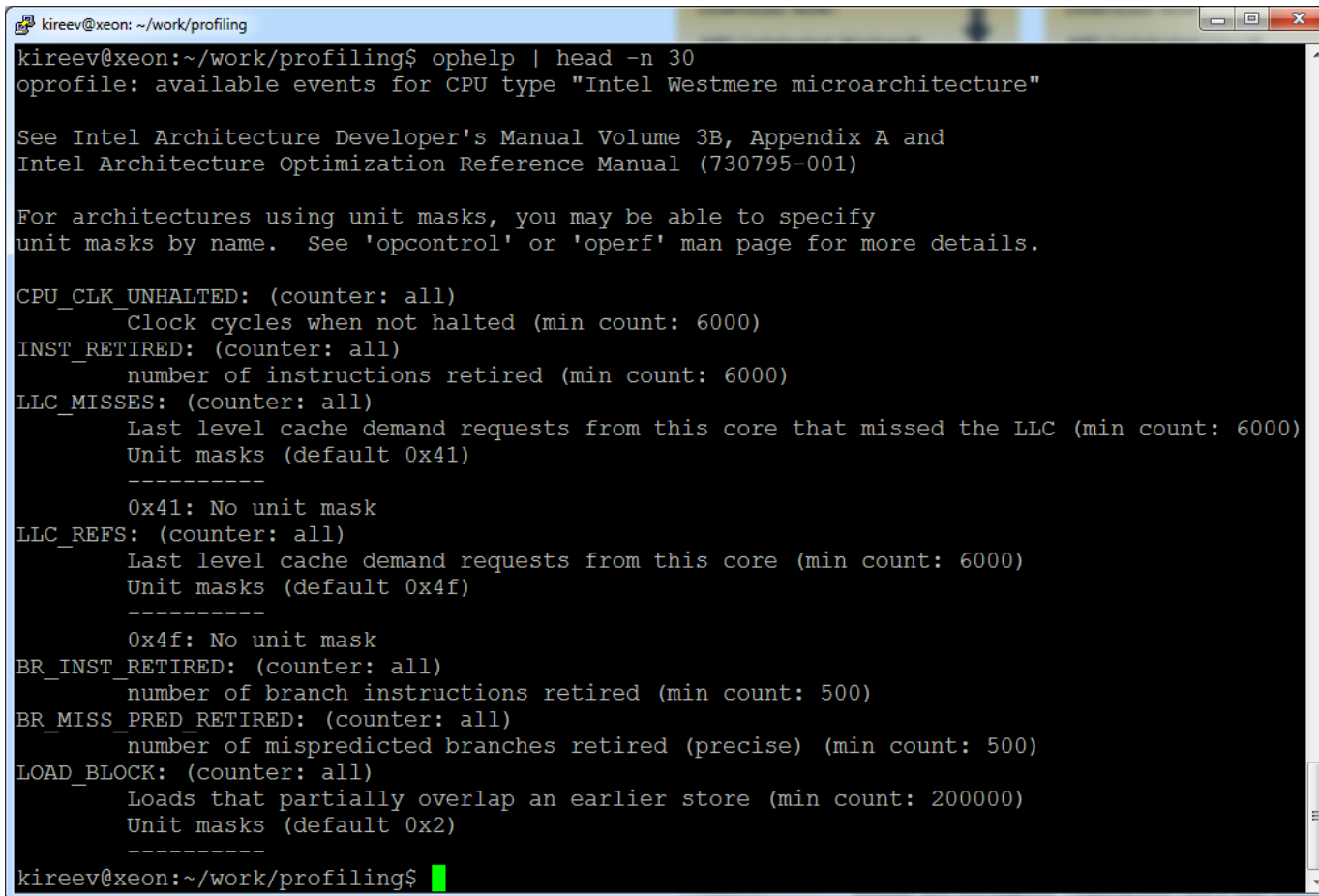
```
sys       0m0.008s
```

- Создаётся
- Компиляция
- \$ g++ -O1
- Создается



# OProfile

- Позволяет получать аппаратные счетчики производительности процессора:

A terminal window titled 'kireev@xeon: ~/work/profiling' displays the output of the command 'oprofile: ophelp | head -n 30'. The output lists various hardware performance events available for Intel Westmere microarchitecture, including CPU\_CLK\_UNHALTED, INST\_RETIRED, LLC\_MISSES, LLC\_REFS, BR\_INST\_RETIRED, BR\_MISS\_PRED\_RETIRED, and LOAD\_BLOCK. Each event is followed by a brief description and its minimum count.

```
kireev@xeon: ~/work/profiling$ ophelp | head -n 30
oprofile: available events for CPU type "Intel Westmere microarchitecture"

See Intel Architecture Developer's Manual Volume 3B, Appendix A and
Intel Architecture Optimization Reference Manual (730795-001)

For architectures using unit masks, you may be able to specify
unit masks by name. See 'opcontrol' or 'operf' man page for more details.

CPU_CLK_UNHALTED: (counter: all)
    Clock cycles when not halted (min count: 6000)
INST_RETIRED: (counter: all)
    number of instructions retired (min count: 6000)
LLC_MISSES: (counter: all)
    Last level cache demand requests from this core that missed the LLC (min count: 6000)
    Unit masks (default 0x41)
    -----
    0x41: No unit mask
LLC_REFS: (counter: all)
    Last level cache demand requests from this core (min count: 6000)
    Unit masks (default 0x4f)
    -----
    0x4f: No unit mask
BR_INST_RETIRED: (counter: all)
    number of branch instructions retired (min count: 500)
BR_MISS_PRED_RETIRED: (counter: all)
    number of mispredicted branches retired (precise) (min count: 500)
LOAD_BLOCK: (counter: all)
    Loads that partially overlap an earlier store (min count: 200000)
    Unit masks (default 0x2)
    -----
kireev@xeon:~/work/profiling$
```

# OProfile

- Получение списка аппаратных счетчиков
  - `$ophelp`
- Компиляция программы
  - `$g++ -O1 -g -o cache cache.c`
- Запуск программы и профилирование
  - `$operf -e CPU_CLK_UNHALTED:100000:0:0:1 ./cache`
  - Создаётся каталог: `./oprofile_data`
- Получение информации об исполнении программы
  - `$oreport ./cache` - профиль по модулям
  - `$oreport -l ./cache` - профиль по функциям
  - `$oreport -c ./cache` - граф вызовов
  - `$orannotate -s ./cache` - аннотированный исходный код
  - `$orannotate -a ./cache` - аннотированный ассемблерный код
  - `$orgprof` - генерация `gmon.out` для `gprof`

# OProfile

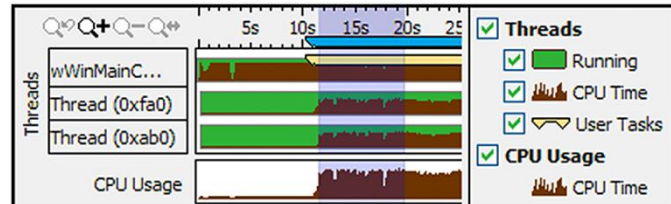
- \$perf -e \  
CPU\_CLK\_UNHALTED:100000:0:1:1,\  
LLC\_REFS:100000:0x4f:0:1:1,\  
LLC\_MISSES:100000:0x41:0:1:1 ./cache

# Intel Vtune Amplifier

- Получение информации с точностью до команд ассемблера
- Построение графа вызовов
- Получение значений аппаратных счетчиков производительности (Intel)
- Динамика загрузки ядер процессора
- Динамика работы многопоточных программ (ожидание на блокировках)
- Подсказки вариантов оптимизации
- Удобный графический интерфейс
- Поддержка только процессоров Intel
- ...еще много плюсов...
- От 899\$

/Function /Call Stack	CPU Time
initialize_2D_buffer	11.768s
grid_intersect	5.916s
intersect_objects	5.431s
grid_intersect ← intersect_objects	0.485s
sphere_intersect	5.044s

Line	Source	CPU Time
579	cur = g->cells[voxindex];	0.204s
580	while (cur != NULL) {	0.048s
581	if (ry->mbox[cur->obj->id] != NULL)	1.611s
582	ry->mbox[cur->obj->id] = ry->	1.025s
583	cur->obj->methods->intersect	1.098s

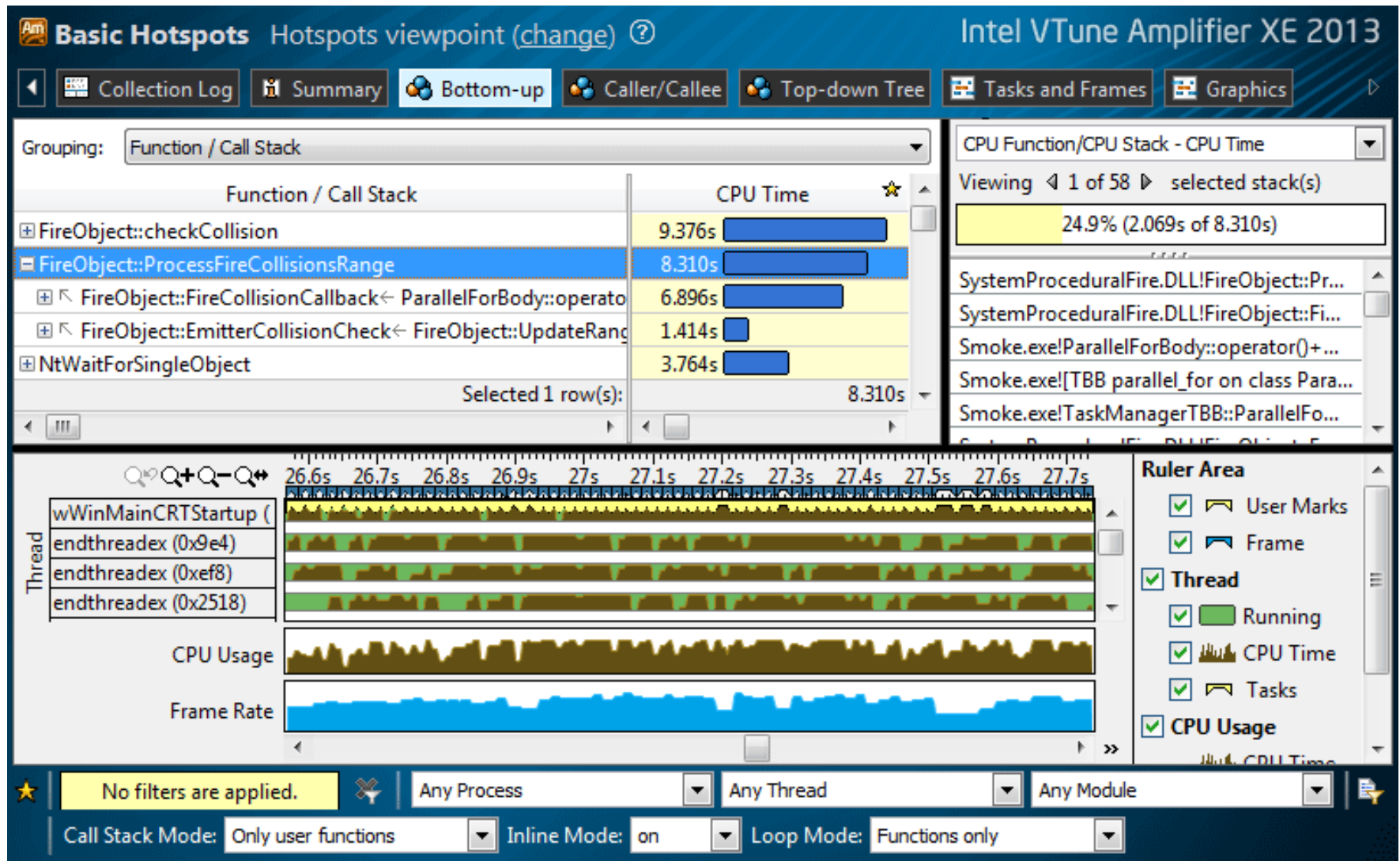


/Sync Object /Function /Call Stack	Wait Time	Wait Count
Manual Reset Event 0xbe5a38e	36.070s	2
GdipCreateSolidFill	36.070s	1
video::~video	0.000s	1
Multiple Objects	20.966s	515

/Function	PMU Event Count		CPI	Branch Mispredict
	CPU_CLK...	INST_RETIRE...		
initialize_2D_buffer	22,566,000,000	51,210,000,000	0.441	0.040
grid_intersect	11,304,000,000	10,778,000,000	1.049	0.205
sphere_intersect	11,030,000,000			
grid_bounds_intersec	1,580,000,000			

The CPI may be too high. This could be instruction starvation, branch mispredict or the other hardware-related metrics to it

# Intel Vtune Amplifier





# AMD CodeAnalyst Performance Analyzer

- Получение информации с точностью до команд ассемблера
- Получение значений аппаратных счетчиков производительности (AMD)
- Динамика загрузки ядер процессора
- Динамика работы многопоточных программ (ожидание на блокировках)
- Удобный графический интерфейс
- Поддержка только процессоров AMD
- Встроенный эмулятор процессоров
- ...еще много плюсов...

