

## **Исследование вопроса**

пока я выложу свои изыскания. если появятся альтернативы – сделаем разные странички, или секции – как будет удобнее.

## **План**

1. Найти hotspots
  2. Сравнить их относительный вклад во время работы
  3. Наметить пути оптимизации

# Hotspots

Профилировал gcc/gcov. Для этого патчил Makefile (cd src ; patch -p1 Makefile-profiling.patch):

```
diff -ruN src-org/Makefile src1/Makefile
--- src-org/Makefile 2007-09-17 17:43:08.000000000 +0400
+++ src1/Makefile 2007-10-31 22:58:53.000000000 +0300
@@ -59,14 +59,14 @@
CINC      = -I$(SRC_DIR)
CDEFS     =
COBJ      = -c -o$(OBJ_DIR)/$@
-CDEFOPT   = -O2
+CDEFOPT   = -g -pg -fprofile-arcs -ftest-coverage
COPT      =
-CFLAGS    =
+CFLAGS    = -O3
CFLAGS_ALL = $(CFLAGS) $(CINC) $(CDEFS) $(CDEFOPT) $(CPROC) $(CPLAT)

LD        = g++
LDPLAT   =
-LDFLAGS  =
+LDFLAGS  = -g -pg -ax -fprofile-arcs -ftest-coverage
LDOUTOPT = -o "$(OUT_DIR)/$(BENCHMARK)"
LIBS      = -lm -lc
LIBS_ALL  = $(LIBS)
```

[Результаты](#) (`./sunset -cfg ..../input/Sample01.cfg`) – основное:

```
-:  
729:!!!!!!!!!!!!!!  
-: 730:!!!!!! Water surface modelling  
!!!!!!  
-:  
731:!!!!!!  
-: 732:*/
```

```
247782400: 733:           for(t = 0; t < NKMAX; t++)
    - : 734:           {
240217600: 735:               OT  = flOmega[t] * flTime;
240217600: 736:               KX1 = flK[t] * flDecartX[i][j];
240217600: 737:               KY1 = flK[t] * flDecartY[i][j];
    - : 738:
7927180800: 739:           for(l = 0; l < iAngleHarmNum; l++)
    - : 740:           {
7686963200: 741:               iSinIndex1 = t * iAngleHarmNum + l;
    - : 742:
flArgSin[currentthread].aptr[iSinIndex1] = OT -
    - : 743:                   KX1 * flAzimuthCosFi[l] - KY1 *
flAzimuthSinFi[l] +
7686963200: 744:                   flRandomPhase[t*iAngleHarmNum +
l];
    - : 745:               } /* end for l */
    - : 746:           } /* end for t */
    - : 747:
7564800: 748:           pFlTmp = flArgSin[currentthread].aptr;
    - : 749:
    - : 750: #pragma ivdep
7753920000: 751:           for(t=0; t<iWaveMeshSize; t++)
7746355200: 752:               pFlTmp[t] = (float)sinf(pFlTmp[t]);
    - : 753:
    - : 754: /* initialize the values of derivation */
7564800: 755:           flDerivX = 0.0f;
7564800: 756:           flDerivY = 0.0f;
    - : 757:
    - : 758: /* dot product to compute derivation */
7753920000: 759:           for(t = 0; t < iWaveMeshSize; t++)
    - : 760:           {
7746355200: 761:               flDerivX += pFlTmp[t] *
flAmplitudeX[t];
7746355200: 762:               flDerivY += pFlTmp[t] *
flAmplitudeY[t];
    - : 763:
    - : 764:
```

Интерпретация: алгоритм проходит по всем точкам изображения (7564800 действий). Для прообраза каждой точки изображения, находящегося на поверхности воды рассчитывается  $iWaveHarmNum * iAngleHarmNum$  дополнительных значений (во всех примерах  $- 32*32 == 1024$ ). Это – аргументы синусов, сами синусы и скалярные произведения амплитуд на эти синусы (7746355200 действий).

Начальное время на моей машине: 45.457/кадр

## Вклад во время

закомментировал блок целиком:

```
diff -ruN src-org/sunset.cpp src1/sunset.cpp
--- src-org/sunset.cpp 2007-09-16 12:04:44.000000000 +0400
+++ src1/sunset.cpp 2007-10-31 23:34:04.000000000 +0300
@@ -730,6 +730,11 @@
 !!!!!!!!!!!!!!! Water surface modelling !!!!!!!!!!!!!!!
 !!!!!!!!!!!!!!!
 */
+         /* initialize the values of derivation */
+         flDerivX = 0.0f;
+         flDerivY = 0.0f;
+
+#if 0
+         for(t = 0; t < NKMAX; t++)
+         {
+             OT = flOmega[t] * flTime;
@@ -751,17 +756,13 @@
+             for(t=0; t<iWaveMeshSize; t++)
+                 pFlTmp[t] = (float)sinf(pFlTmp[t]);
+
-         /* initialize the values of derivation */
-         flDerivX = 0.0f;
-         flDerivY = 0.0f;
-
-         /* dot product to compute derivation */
-         for(t = 0; t < iWaveMeshSize; t++)
-         {
-             flDerivX += pFlTmp[t] * flAmplitudeX[t];
-             flDerivY += pFlTmp[t] * flAmplitudeY[t];
-         }
-
+#endif
+         /* Near horizont area correction */
+         flDerivX *= P2;
+         flDerivY *= P2;
```

Время: 1.225/кадр.

1. Оставил только расчет аргументов. Время: 4.771/кадр.
2. Оставил только расчет скалярного произведения. Время: 2.436/кадр.

Результат: основное время уходит в тригонометрию, затем – в генерацию аргументов, затем – в скалярное произведение.

## Пути оптимизации

### Очевидные шаги

1. включить openmp, заготовки которого уже есть в коде (: (ускорение пропорционально количеству вычислительных ядер))
2. очень много математики. однако нет ни специальных значений аргументов, ни проверок matherr, ничего такого. Включить на полную оптимизацию математических вызовов. (ускорение до 35.943/кадр или ~ в 1.2 раза)

```
diff -ruN src-org/Makefile src/Makefile
--- src-org/Makefile 2007-09-17 17:43:08.000000000 +0400
+++ src/Makefile 2007-10-23 22:11:17.000000000 +0400
@@ -59,14 +59,14 @@
CINC      = -I$(SRC_DIR)
CDEFS     =
COBJ      = -c -o$(OBJ_DIR)/$@
-CDEFOPT   = -O2
+CDEFOPT   = -g -pg -fprofile-arcs -ftest-coverage
COPT      =
-CFLAGS    =
+CFLAGS    = -O3 -ffast-math -ffinite-math-only -fno-math-errno -funsafe-
math-optimizations -fno-trapping-math -march=prescott -fopenmp
CFLAGS_ALL = $(CFLAGS) $(CINC) $(CDEFS) $(CDEFOPT) $(CPROC) $(CPLAT)

LD         = g++
LDPLAT    =
-LDFLAGS   =
+LDFLAGS   = -g -pg -ax -fprofile-arcs -ftest-coverage -fopenmp
LDOUTOPT  = -o "$(OUT_DIR)/$(BENCHMARK)"
LIBS      = -lm -lc
LIBS_ALL  = $(LIBS)
```

### Менее очевидные шаги

1. ускорить счет синуса. кустарные методы отдыхают (:

```
diff -ruN src-org/Makefile src1/Makefile
--- src-org/Makefile 2007-09-17 17:43:08.000000000 +0400
+++ src1/Makefile 2007-11-02 00:49:53.000000000 +0300
@@ -56,19 +56,19 @@
CC        = gcc
CPLAT    =
CPROC    =
-CINC     = -I$(SRC_DIR)
+CINC     = -I$(SRC_DIR) -I/opt/intel/ipp/5.2/ia32/include
CDEFS    =
```

```

COBJ      = -c -o$(OBJ_DIR)/$@
-CDEFOPT  = -O2
+CDEFOPT  = -g -pg -fprofile-arcs -ftest-coverage
COPT      =
-CFLAGS   =
+CFLAGS   = -O3 -ffast-math -ffinite-math-only -fno-math-errno -funsafe-
math-optimizations -fno-trapping-math -march=prescott
CFLAGS_ALL = $(CFLAGS) $(CINC) $(CDEFS) $(CDEFOPT) $(CPROC) $(CPLAT)

LD        = g++
LDPLAT   =
-LDFLAGS  =
+LDFLAGS  = -g -pg -ax -fprofile-arcs -ftest-coverage -
L/opt/intel/ipp/5.2/ia32/sharedlib
LDOUTOPT = -o "$(OUT_DIR)/$(BENCHMARK)"
-LIBS     = -lm -lc
+LIBS     = -lm -lc -lippcore -lippvm -lguide
LIBS_ALL  = $(LIBS)

endif
diff -ruN src-org/sunset.cpp src1/sunset.cpp
--- src-org/sunset.cpp 2007-09-16 12:04:44.000000000 +0400
+++ src1/sunset.cpp 2007-11-02 00:34:38.000000000 +0300
@@ -45,6 +45,7 @@
 #include <omp.h>
 #endif
 #include "sunset.h"
+#include <ippvm.h>

#define MIN(x,y)    (((x) < (y)) ? (x) : (y))
#define MAX(x,y)    (((x) < (y)) ? (y) : (x))
@@ -747,9 +748,10 @@

```

```

pFlTmp = flArgSin[currentthread].aptr;

-
#pragma ivdep
-    for(t=0; t<iWaveMeshSize; t++)
-        pFlTmp[t] = (float)sinf(pFlTmp[t]);
+    ippsSin_32f_A21(pFlTmp,pFlTmp,iWaveMeshSize);
+
//#pragma ivdep
+    //for(t=0; t<iWaveMeshSize; t++)
//        pFlTmp[t] = (float)tab_sinf(pFlTmp[t]);

/* initialize the values of derivation */
flDerivX = 0.0f;

```

Результат - 10.044/кадр, 0.7% отличий. Понижение точности до 11 бит дает 9.139/кадр, 1% отличий. Однако, по-простецки с openmp оно дружить не захотело - segfault.

Откомпилировать другими компиляторами. если времена для одного и того же кода будут заметно отличаться - смотреть в ассемблер. или просто использовать другой компилятор и

дальше (.

В части использования инструментов intel можно применить следующее:

1 - Провести компиляцию с их [компилятором\(Описание\)](#):

а) должен позволить автоматически использовать SIMD команды (SSE, SSE2...) для оптимизации вычислений в основном в циклах

б) возможно подскажет где что можно ещё распараллелить

2 - Для тригонометрии использовать [Intel Performance Primitives\(Описание\)](#):

а) как заявляется производительность растёт в том числе и за счёт оптимизации библиотеки под различные модели процессоров

б) можно попробовать использовать оттуда не только тригонометрию но и функции работы с изображениями 2D

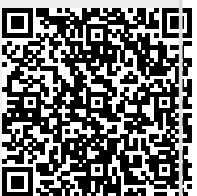
P.S. Менеджер проекта этой библиотеки из Нижнего Новгорода откуда и сам sunset :)

3 - Попытаться использовать [Math Kernel Library\(Описание\)](#):

а) Как более глубокий вариант оптимизации с учётом того что библиотека в основном хорошо реализует матричные операции

From:

<http://wiki.osll.ru/> - Open Source & Linux Lab



Permanent link:

[http://wiki.osll.ru/doku.php/etc:common\\_activities:intel\\_students\\_cup:tour2?rev=1193954453](http://wiki.osll.ru/doku.php/etc:common_activities:intel_students_cup:tour2?rev=1193954453)

Last update: 2008/01/03 02:32