Использование ICC

Установил icc в Fedora 7 (только компилятор). Пришлось дополнительно yum install compat-libstdc++-33.i386

Документация: /opt/intel/cc/10.0.023/doc/main_cls/whnjs.htm

man: man -M /opt/intel/cc/10.0.023/man icc

Векторизация

Включается ключом -x... или -ax... В примере – -xP. Перед этим робовал -axT, но выяснил, что автоопределение типа процессора работает странно. С -axT Core Duo T2300 работал по коду сопроцессора, а не по SSE.

В совокупности с ippsSin время 5.808/кадр.

Изменения в Makefile и циклах:

```
diff -ruN src-org/Makefile src-icc/Makefile
--- src-org/Makefile    2007-09-17 17:43:08.000000000 +0400
+++ src-icc/Makefile    2007-11-03 22:28:21.000000000 +0300
@@ -53,22 +53,23 @@

 OBJ = .o

-CC         = gcc
+CC         = icc
 CPLAT      =
 CPROC      =
 CINC       = -I$(SRC_DIR)
 CDEFS      =
 COBJ       = -c -o$(OBJ_DIR)/$@
-CDEFOPT    = -O2
+CDEFOPT    = -O3 -xP -fp-model fast -fp-speculation fast -fno-math-errno -g
 COPT       =
 CFLAGS     =
 CFLAGS_ALL = $(CFLAGS) $(CINC) $(CDEFS) $(CDEFOPT) $(CPROC) $(CPLAT)

 LD         = g++
-LDPLAT     =
-LDFLAGS    =
+LDPLAT     =
+LDFLAGS    = -L/opt/intel/cc/10.0.023/lib
 LDOUTOPT   = -o "$(OUT_DIR)/$(BENCHMARK)"
-LIBS       = -lm -lc
+LIBS       = -lm -lc -lirc -limf -lsvml -lippcore -lippvm -lipgo
+# -lompstub -lomp_db -lguide
```

```
 LIBS_ALL   = $(LIBS)

 endif
diff -ruN src-org/sunset.cpp src-icc/sunset.cpp
--- src-org/sunset.cpp  2007-09-16 12:04:44.000000000 +0400
+++ src-icc/sunset.cpp  2007-11-05 11:38:25.000000000 +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))
@@ -730,6 +731,9 @@
 !!!!!!!!!!!!!!!! Water surface modelling !!!!!!!!!!!!!!!!!!!!!!!!
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 */
+
+                pFlTmp = flArgSin[currentthread].aptr;
+
                 for(t = 0; t < NKMAX; t++)
                 {
                     OT  = flOmega[t] * flTime;
@@ -739,17 +743,16 @@
                     for(l = 0; l < iAngleHarmNum; l++)
                     {
                         iSinIndex1 = t * iAngleHarmNum + l;
-                        flArgSin[currentthread].aptr[iSinIndex1] = OT -
+                        pFlTmp[iSinIndex1] = OT -
                             KX1 * flAzimuthCosFi[l] - KY1 *
flAzimuthSinFi[l] +
-                            flRandomPhase[t*iAngleHarmNum + l];
+                            flRandomPhase[iSinIndex1];
                     } /* end for l */
                 } /* end for t */

-                pFlTmp = flArgSin[currentthread].aptr;
-
-                #pragma ivdep
-                for(t=0; t<iWaveMeshSize; t++)
-                    pFlTmp[t] = (float)sinf(pFlTmp[t]);
+                ippsSin_32f_A11(pFlTmp,pFlTmp,iWaveMeshSize);
+                //#pragma ivdep
+                //for(t=0; t<iWaveMeshSize; t++)
+                //    pFlTmp[t] = (float)sinf(pFlTmp[t]);

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

Оптимизация по результатам профилирования (PGO)

В CDEFOPT добавить -prof-gen. Откомпилировать, запустить. Появится файл с результатами профилирования. Заменить -prof-gen на -prof-use. Откомпилировать, запустить.

Существенной разницы не заметил.

OpenMP

В коде интересная ловушка: #pragma omp parallel for **ordered**. Насколько я понял из спецификации OpenMP, ordered в цикле позволяет помещать внутри него секции ordered, которые будут выполняться всегда в одном и том же порядке. Однако, в нашем коде нет таких секций. Убрал ordered. Скорость возросла как и ожидалось, почти вдвое, при сохранении точности. Странно, что gcc слово ordered не смутило.

В совокупности с ippSin и векторизацией – 2.843/кадр.

```
diff -ruN src-org/sunset.cpp src-icc/sunset.cpp
--- src-org/sunset.cpp  2007-09-16 12:04:44.000000000 +0400
+++ src-icc/sunset.cpp  2007-11-05 13:44:08.000000000 +0300
@@ -656,7 +657,7 @@
         flOmegaTime[i] = flTime * flOmega[i];

 #if defined (_OPENMP)
-    #pragma omp parallel for ordered \
+    #pragma omp parallel for \
         private(k, i, l, t, iPointIndex, iColorIndex, j) \
         private(T2, FI2, CT2, ST2, chStatus0, flR) \
         private(CFI2, SFI2, flDeltaFi2, flDeltaT2, flSunPoint) \
```

From:
http://wiki.osll.ru/ - **Open Source & Linux Lab**

Permanent link:
http://wiki.osll.ru/doku.php/etc:common_activities:intel_students_cup:icc

Last update: **2008/01/03 02:32**