PMlibのインストールとテスト

- PMlibの入手方法
- テストシステムへのログイン
- PMlibのインストール
- 動作確認プログラムの実行

PMlibのインストール　京コンピュータ(1)

- PMlibのインストール作業はログインノードでも計算ノードでも可能。本日はログインノードでインストール実施
- PMlibの利用は計算ノードアプリケーションが行う
- 任意のディレクトリでパッケージを展開。インストール先のディレクトリを　−−prefix で指定しconfigureの実施。自動作成されるMakefileを用いて、makeの実施。

```
$ tar -zxf PMlib-master.2.1.3.vsh.tar.gz
$ ls -l
$ drwxr-xr-x 8 a03155 ra000004     4096  5月 26 04:30 PMlib-master
$ cd PMlib-master
$ ./configure CXX=mpiFCCpx CC=mpifccpx FC=mpifrtpx \
    CFLAGS='-Kopenmp,fast  -Ntl_notrt’ \
    CXXFLAGS='-Kopenmp,fast -DUSE_PAPI  -Ntl_notrt' \
    --host=sparc64-unknown-linux-gnu   \
    --with-papi=yes  --with-example=yes \
    --prefix=${HOME}/pmlib/install_dir
$ make
$ make install
```

PMlibのインストール　京コンピュータ(2)

- 京でのインストール時間は数分で終了。正常にインストールされると以下のファイルができている。

```
$ ls -CF install_dir
bin/  doc/  include/  lib/  share/

$ ls -go install_dir/bin install_dir/include install_dir/lib

install_dir/bin:
total 4
-rwxr-xr-x 1 1563 May 26 19:18 pm-config

install_dir/include:
total 28
-rw-r--r-- 1 5798 May 26 19:18 PerfMonitor.h
-rw-r--r-- 1 6099 May 26 19:18 PerfWatch.h
-rw-r--r-- 1 1490 May 26 19:18 mpi_stubs.h
-rw-r--r-- 1  627 May 26 19:18 pmVersion.h
-rw-r--r-- 1 2079 May 26 19:18 pmlib_papi.h

install_dir/lib:
total 4136
-rw-r--r-- 1 4219910 May 26 19:18 libPM.a
-rw-r--r-- 1   11938 May 26 19:18 libpapi_ext.a
```

```
$ ls -go ./example/
total 2764
-rw-r--r-- 1   24025 May 26 19:17 Makefile
-rw-r--r-- 1    1649 May 23 00:14 Makefile.am
-rw-r--r-- 1   25696 May 23 00:14 Makefile.in
-rw-r--r-- 1    1130 May 23 00:14 Makefile_hand.fx10.login
-rw-r--r-- 1    1083 May 23 00:14 Makefile_hand.intel
-rwxr-xr-x 1 1062806 May 26 22:33 check_new_api
-rw-r--r-- 1    6096 May 23 00:14 check_new_api.c
-rwxr-xr-x 1 4409884 May 26 22:33 pmlib_test
-rw-r--r-- 1    2181 May 23 00:14 pmlib_test.cpp
-rw-r--r-- 1    1156 May 23 00:14 sub_kernel.c
```

PMlibを用いる　京コンピュータ(1)

- Example/ 以下のサンプルプログラムでPMlibを利用してみる

```
#!/bin/bash
set -x
date; hostname; /opt/FJSVXosPA/bin/xospastop
PMLIB=${HOME}/pmlib/PMlib-master
PMLIB_INCLUDE=-I${PMLIB}/include
PMLIB_LIB=${PMLIB}/src/libPM.a
PAPI_ROOT=/usr
PAPI_LIB="$PAPI_ROOT/lib64/libpapi.a $PAPI_ROOT/lib64/libpfm.a"
PAPI_EXT="$PMLIB/src_papi_ext/libpapi_ext.a”
CXXFLAGS="-Kfast,parallel,openmp ${PMLIB_INCLUDE} ${PAPI_INCLUDE}”
CCFLAGS="-std=c99 -Xg -Kfast,parallel,openmp ${PMLIB_INCLUDE} ${PAPI_INCLUDE}”
LDFLAGS="${PMLIB_LIB} ${PAPI_LIB} ${PAPI_EXT}”
SRC_DIR=${HOME}/pmlib/PMlib-master/example
WKDIR=/data/ra000004/a03155/tmp/check_pmlib ; mkdir -p $WKDIR
cd $WKDIR; if [ $? != 0 ] ; then echo '@@@ Directory error @@@'; exit; fi
cp $SRC_DIR/pmlib_main.cpp main.cpp
cp $SRC_DIR/sub_kernel.c sub.c
mpiFCC -c ${CXXFLAGS}  main.cpp
mpifcc -c ${CCFLAGS}  sub.c
mpiFCC    ${CXXFLAGS}  main.o sub.o ${LDFLAGS}
export OMP_NUM_THREADS=4
mpirun -np 2 ./a.out
```

Pmlibを用いる　京実行結果例 (1)

- 基本プロファイル＋MPIプロセス毎プロファイル

```
Report of Timing Statistics PMlib version 2.1.3
Operator  : RRR
Host name : QQQ
Date      : 2014/05/26 : 23:25:43
Parallel Mode                    :   Hybrid (2 processes x 4 threads)
Total execution time             = 7.231672e-01 [sec]
Total time of measured sections = 7.296531e-01 [sec]

Statistics per MPI process [Node Average]
Label    |     call     |            accumulated time                   |         flop | messages[Bytes]
         |              |  avr[sec]    avr[%]     sdv[sec]  avr/call[sec] |   avr         sdv         speed
---------+--------------+-----------------------------------------------+-------------------------------------
section1 :           2   4.918501e-01   67.41   2.4496e-02  2.459251e-01    0.000e+00   0.000e+00     0.00 Mflops
section2 :           1   2.378030e-01   32.59   1.9418e-03  2.378030e-01    4.000e+09   0.000e+00    16.82 Gflops
---------+--------------+-----------------------------------------------+-------------------------------------
Total    |               7.296531e-01                                     4.000e+09               5.48 Gflops
Performance|                                                                          10.96 Gflops

---------------------------------------------------
Elapsed time variation over MPI ranks

section1
MPI_rank       call       accm[s] accm[%]    waiting[s]  accm/call[s]    flop|msg     speed
#0    :           2  4.745290e-01   65.03  3.464222e-02  2.372645e-01   0.000e+00     0.00 Mflops
#1    :           2  5.091712e-01   69.78  0.000000e+00  2.545856e-01   0.000e+00     0.00 Mflops
section2
MPI_rank       call       accm[s] accm[%]    waiting[s]  accm/call[s]    flop|msg     speed
#0    :           1  2.391760e-01   32.78  0.000000e+00  2.391760e-01   4.000e+09    16.72 Gflops
#1    :           1  2.364299e-01   32.40  2.746105e-03  2.364299e-01   4.000e+09    16.92 Gflops
```

Pmlibを用いる　京実行結果例 (2)

- 環境変数を追加　export HWPC_CHOOSER=FLOPS

```
        Statistics per MPI process [Node Average]
        Label    |     call    |          accumulated time                   |        flop | messages[Bytes]
                 |             |  avr[sec]   avr[%]    sdv[sec]  avr/call[sec] |  avr        sdv         speed
        ---------+-------------+---------------------------------------------+-------------------------------------
        section1 :          2   4.843562e-01   67.16   9.8485e-03  2.421781e-01   8.123e+09   2.828e+00     16.77 Gflops
        section2 :          1   2.368304e-01   32.84   2.1871e-03  2.368304e-01   4.033e+09   7.071e-01     17.03 Gflops
        ---------+-------------+---------------------------------------------+-------------------------------------
        Total    |              7.211865e-01                                  1.216e+10             16.86 Gflops
        Performance|                                                                     33.71 Gflops

        ----------------------------------------------------
        PMlib detected the CPU architecture:
        The available Hardware Performance Counter (HWPC) events depend on this CPU architecture.
                Sun
                Fujitsu SPARC64 VIIIfx
        HWPC event values of the master rank, sum of threads. count unit in Giga (x 10e9)
        -        :      FP_OPS    [GFlops]
        section1 :       8.123      17.015
        section2 :       4.033      16.919

        ----------------------------------------------------
        Elapsed time variation over MPI ranks
section1
MPI_rank        call       accm[s] accm[%]    waiting[s]  accm/call[s]    flop|msg      speed
#0     :          2  4.773922e-01   66.20  1.392794e-02  2.386961e-01   8.123e+09    17.02 Gflops
#1     :          2  4.913201e-01   68.13  0.000000e+00  2.456601e-01   8.123e+09    16.53 Gflops
section2
MPI_rank        call       accm[s] accm[%]    waiting[s]  accm/call[s]    flop|msg      speed
#0     :          1  2.383769e-01   33.05  0.000000e+00  2.383769e-01   4.033e+09    16.92 Gflops
#1     :          1  2.352839e-01   32.62  3.093004e-03  2.352839e-01   4.033e+09    17.14 Gflops
```

Pmlibを用いる　京実行結果例 (3)

- 環境変数を追加　export HWPC_CHOOSER=FLOPS,VECTOR

```
PMlib detected the CPU architecture:
The available Hardware Performance Counter (HWPC) events depend on this CPU architecture.
        Sun
        Fujitsu SPARC64 VIIIfx
HWPC event values of the master rank, sum of threads. count unit in Giga (x 10e9)
-        :    FP_OPS   [GFlops]   VEC_INS   FMA_INS
section1 :     8.123     17.010     3.180     0.881
section2 :     4.033     16.758     1.581     0.435

        HWPC events legend: count unit in Giga (x 10e9)
                FP_OPS: floating point operations
                VEC_INS: vector instructions
                FMA_INS: Fused Multiply-and-Add instructions
                LD_INS: memory load instructions
                SR_INS: memory store instructions
                L1_TCM: level 1 cache miss
                L2_TCM: level 2 cache miss (by demand and by prefetch)
                L2_WB_DM: level 2 cache miss by demand with writeback request
                L2_WB_PF: level 2 cache miss by prefetch with writeback request
                TOT_CYC: total cycles
                MEM_SCY: Cycles Stalled Waiting for memory accesses
                STL_ICY: Cycles with no instruction issue
                TOT_INS: total instructions
                FP_INS: floating point instructions
        Derived statistics:
                [GFlops]: floating point operations per nano seconds (10^-9)
                [Mem GB/s]: memory bandwidth in load+store GB/s
                [L1$ %]: Level 1 cache hit percentage
                [LL$ %]: Last Level cache hit percentage
```

Pmlibを用いる　京実行結果例 (4)

- 環境変数を変更　export HWPC_CHOOSER=BANDWIDTH

```
Statistics per MPI process [Node Average]
Label    |     call     |            accumulated time                  |         flop | messages[Bytes]
         |              |  avr[sec]    avr[%]     sdv[sec]  avr/call[sec] |   avr        sdv        speed
---------+--------------+----------------------------------------------+------------------------------------
section1 :          2   4.823370e-01   66.93   3.7025e-03  2.411685e-01    1.007e+10   6.975e+08    19.45 GB/sec
section2 :          1   2.383659e-01   33.07   2.4961e-03  2.383659e-01    5.227e+09   6.785e+08    20.42 GB/sec
---------+--------------+----------------------------------------------+------------------------------------
Total    |              7.207029e-01                                    1.530e+10             19.77 GB/sec
Performance|                                                                       39.54 GB/sec

-----------------------------------------------------
PMlib detected the CPU architecture:
The available Hardware Performance Counter (HWPC) events depend on this CPU architecture.
          Sun
          Fujitsu SPARC64 VIIIfx
HWPC event values of the master rank, sum of threads. count unit in Giga (x 10e9)
-       :      L2_TCM    L2_WB_DM    L2_WB_PF  [Mem GB/s]
section1 :       0.082       0.000       0.000      22.025
section2 :       0.045       0.000       0.000      23.763

          HWPC events legend: count unit in Giga (x 10e9)
                    FP_OPS: floating point operations
                    VEC_INS: vector instructions
                    FMA_INS: Fused Multiply-and-Add instructions
                    LD_INS: memory load instructions
                    SR_INS: memory store instructions
                    L1_TCM: level 1 cache miss
                    L2_TCM: level 2 cache miss (by demand and by prefetch)
                    L2_WB_DM: level 2 cache miss by demand with writeback request
                    L2_WB_PF: level 2 cache miss by prefetch with writeback request
```

参考資料 OpenMPプログラムへの組み込み

```
#include <omp.h>
int main (int argc, char *argv[])
{
        matrix.nsize = nsize;
        set_array();

        int loop=3;
        for (int i=1; i<=loop; i++){
                subkerel();
        }

        return 0;
}

void subkerel()
{
int i, j, k, nsize;
float c1,c2,c3;
nsize = matrix.nsize;
#pragma omp parallel
#pragma omp for
        for (i=0; i<nsize; i++){
        for (j=0; j<nsize; j++){
            …
}
```

ソースプログラム全体は example/ ディレクトリにあります

```
#include <omp.h>
#include <PerfMonitor.h>
using namespace pm_lib;
PerfMonitor PM;

int main (int argc, char *argv[])
{
int my_id=0, num_threads, npes=1;
char parallel_mode[] = "OpenMP";
double flop_count, dsize;
matrix.nsize = nsize;
dsize = (double)nsize;
num_threads  = omp_get_max_threads();
PM.initialize();
PM.setParallelMode(parallel_mode, num_threads, npes);
PM.setRankInfo(my_id);
PM.setProperties("check_2", PerfMonitor::CALC);

set_array();
flop_count=pow (dsize, 3.0)*4.0;

PM.start("check_2");
int loop=3;
for (int i=1; i<=loop; i++){
      subkerel();
}
PM.stop ("check_2", flop_count, 1);
PM.gather();
PM.print(stdout, "London", "Mr. Bean");
PM.printDetail(stdout);
return 0;
}
```

ハンズオン

- ハンズオンプログラムについて
- プログラムの実行
- プログラムへのPMlibのくみこみ
- プログラムのPMlib統計情報の解釈と検討

ハンズオンプログラムについて

- 立方体領域の熱伝導問題
- 主要な計算はラプラス方程式の差分解法
- 解法のオプションとしてJacobi法とSOR法を選択可能
- 1プロセス実行用
- スレッド並列化用のOpenMP指示行含む
- プログラム言語
  - メインプログラム：C++
  - 主要な計算サブルーチン： Fortran

ハンズオンプログラムの構成

```
Main()
{
        bc_();
        src_dirichlet_();
        case JACOBI:
         for (loop=1; loop<=ItrMax; loop++)                    非排他測定区間
         {                                                     ”Jacobi”
                 jacobi_();        排他測定区間”Jacobi_kernel”
                 bc_();            排他測定区間”BoundaryCondition”
        }
        case SOR:
        for (loop=1; loop<=ItrMax; loop++)                     非排他測定区間
        {                                                      ” PointSOR”
                 psor_();          排他測定区間″Sor_kernel″
                 bc_();            排他測定区間”BoundaryCondition”
        }
}
```

ハンズオンプログラムについて

- ソースプログラム

```
$ ls -go *.cpp *.f90 *.h
-rw-r--r--  1    927 12 26 20:57 FortFunc.h
-rw-r--r--  1   1550 12 26 12:53 kernel_def.h
-rw-r--r--  1   5935 12 26 12:53 linear_solver.f90
-rw-r--r--  1   6344  1 15 22:47 main_PMlib.cpp
-rw-r--r--  1   6674  1 15 23:16 main_base.cpp
-rw-r--r--  1    476 12 26 20:56 realtype.h
```

- Makefile

```
 ls -go Makefile.*
-rw-r--r--  1   1324  1 16 01:02 Makefile.K
-rw-r--r--  1   1215  1 16 01:03 Makefile.intel
-rw-r--r--  1   1326  1 15 23:29 Makefile.macosx
```

プログラムのmakeと実行

- ハンズオンプログラム・ベース版
  - プログラムは実際にはPMlibを呼び出さないが、Makefileは PMlib使用版と同じ物を用いる
  - インストールしたPMlib用の環境変数が正しく設定されていることを確認後、makeする

```
$ PMLIB_ROOT=${HOME}/pmlib
$ export PMLIB_INCLUDES="-I${PMLIB_ROOT}/include"
$ export PMLIB_LDFLAGS="-L${PMLIB_ROOT}/lib -lPM"

$ PAPI_ROOT=/usr/local/papi/papi-5.3.2/intel
$ export PAPI_INCLUDE="-I${PAPI_ROOT}/include"
$ export PAPI_LDFLAGS="-L${PAPI_ROOT}/lib -lpapi -lpfm"
$ export PMLIB_LDFLAGS="-L${PMLIB_ROOT}/lib -lPM -lpapi_ext"

$ cp main_base.cpp main.cpp
$ make -f Makefile.K
$ time OMP_NUM_THREADS=1 ./kernel.ex 128 jacobi 1000
```

プログラムへのPMlibの組み込み

- ハンズオンプログラム・PMlib版の作成
  - main.cpp のソースプログラムでコメントアウトされている PMlib関数呼び出し箇所を有効化する（//if_use_Pmlibと書かれている22箇所）

```
//if_use_PMlib     PerfMonitor PM;
//if_use_PMlib     PM.initialize( PM_NUM_MAX );
```

```
PerfMonitor PM;
PM.initialize( PM_NUM_MAX );
```

  - 全て有効化するとソースプログラムは main_PMlib.cpp と同一内容となる
  - 実際にソースを編集してみると雰囲気がわかるので、無駄手間に思えても一度やってみることをオススメ

PMlib組み込みプログラムの実行

```
$ cp main_PMlib.cpp main.cpp
$ make –f Makefile.K
$ time OMP_NUM_THREADS=1 ./kernel.ex 128 jacobi 1000
```

参考資料

- プロセッサ固有のハードウエアパフォーマンスカウンタ（HWPC)について
- 京のHWPCとPAPIインタフェイス
- Intel XeonのHWPCとPAPIインタフェイス
- PAPI 高水準インタフェイスと低水準インタフェイス

ハードウエアカウンタについて

- 京・FX10 preset event

Available events and hardware information.

Vendor string and code : Sun (7)
Model string and code : Fujitsu SPARC64 IXfx (141)
CPU Revision : 0.000000
CPU Megahertz : 1650.000000
CPU Clock Megahertz : 1650
CPU's in this Node : 16
Nodes in this System : 1
Total CPU's : 16
Number Hardware Counters : 8
Max Multiplex Counters : 512

Name	Code	Deriv	Description (Note)
PAPI_L1_DCM	0x80000000	No	Level 1 data cache misses
PAPI_L1_ICM	0x80000001	No	Level 1 instruction cache misses
PAPI_L1_TCM	0x80000006	Yes	Level 1 cache misses
PAPI_L2_TCM	0x80000007	Yes	Level 2 cache misses
PAPI_CA_INV	0x8000000c	No	Requests for cache line invalidation
PAPI_CA_ITV	0x8000000d	No	Requests for cache line intervention
PAPI_TLB_DM	0x80000014	No	Data translation lookaside buffer misses
PAPI_TLB_IM	0x80000015	No	Instruction translation lookaside buffer misses
PAPI_TLB_TL	0x80000016	Yes	Total translation lookaside buffer misses
PAPI_MEM_SCY	0x80000022	No	Cycles Stalled Waiting for memory accesses
PAPI_STL_ICY	0x80000025	No	Cycles with no instruction issue
PAPI_FUL_ICY	0x80000026	No	Cycles with maximum instruction issue
PAPI_STL_CCY	0x80000027	Yes	Cycles with no instructions completed
PAPI_FUL_CCY	0x80000028	Yes	Cycles with maximum instructions completed
PAPI_HW_INT	0x80000029	No	Hardware interrupts
PAPI_BR_MSP	0x8000002e	No	Conditional branch instructions mispredicted
PAPI_BR_PRC	0x8000002f	Yes	Conditional branch instructions correctly predicted
PAPI_FMA_INS	0x80000030	Yes	FMA instructions completed
PAPI_TOT_IIS	0x80000031	Yes	Instructions issued
PAPI_TOT_INS	0x80000032	No	Instructions completed
PAPI_FP_INS	0x80000034	Yes	Floating point instructions
PAPI_LD_INS	0x80000035	Yes	Load instructions
PAPI_SR_INS	0x80000036	Yes	Store instructions
PAPI_BR_INS	0x80000037	No	Branch instructions
PAPI_VEC_INS	0x80000038	Yes	Vector/SIMD instructions
PAPI_TOT_CYC	0x8000003b	No	Total cycles
PAPI_LST_INS	0x8000003c	No	Load/store instructions completed
PAPI_L2_TCH	0x80000056	Yes	Level 2 total cache hits
PAPI_L2_TCA	0x80000059	Yes	Level 2 total cache accesses
PAPI_FP_OPS	0x80000066	Yes	Floating point operations

ハードウエアカウンタについて

- Intel Xeon E5 preset event

Hdw Threads per core : 1
Cores per Socket : 8
Sockets : 2
NUMA Nodes : 2
CPUs per Node : 8
Total CPUs : 16
Running in a VM : no
Number Hardware Counters : 11
Max Multiplex Counters : 32

Name	Code	Deriv	Description (Note)
PAPI_L1_DCM	0x80000000	No	Level 1 data cache misses
PAPI_L1_ICM	0x80000001	No	Level 1 instruction cache misses
PAPI_L2_DCM	0x80000002	Yes	Level 2 data cache misses
PAPI_L2_ICM	0x80000003	No	Level 2 instruction cache misses
PAPI_L1_TCM	0x80000006	Yes	Level 1 cache misses
PAPI_L2_TCM	0x80000007	No	Level 2 cache misses
PAPI_L3_TCM	0x80000008	No	Level 3 cache misses
PAPI_TLB_DM	0x80000014	Yes	Data translation lookaside buffer misses
PAPI_TLB_IM	0x80000015	No	Instruction TLBmisses
PAPI_L1_LDM	0x80000017	No	Level 1 load misses
PAPI_L1_STM	0x80000018	No	Level 1 store misses
PAPI_L2_STM	0x8000001a	No	Level 2 store misses
PAPI_STL_ICY	0x80000025	No	Cycles with no instruction issue
PAPI_BR_UCN	0x8000002a	Yes	Unconditional branch instructions
PAPI_BR_CN	0x8000002b	No	Conditional branch instructions
PAPI_BR_TKN	0x8000002c	Yes	Conditional branch taken
PAPI_BR_NTK	0x8000002d	No	Conditional branch not taken
PAPI_BR_MSP	0x8000002e	No	Conditional branch mispredicted
PAPI_BR_PRC	0x8000002f	Yes	Conditional branch correctly predicted
PAPI_TOT_INS	0x80000032	No	Instructions completed
PAPI_FP_INS	0x80000034	Yes	Floating point instructions
PAPI_LD_INS	0x80000035	No	Load instructions
PAPI_SR_INS	0x80000036	No	Store instructions
PAPI_BR_INS	0x80000037	No	Branch instructions
PAPI_TOT_CYC	0x8000003b	No	Total cycles
PAPI_L2_DCH	0x8000003f	Yes	Level 2 data cache hits
PAPI_L2_DCA	0x80000041	No	Level 2 data cache accesses
PAPI_L3_DCA	0x80000042	Yes	Level 3 data cache accesses
PAPI_L2_DCR	0x80000044	No	Level 2 data cache reads
PAPI_L3_DCR	0x80000045	No	Level 3 data cache reads
PAPI_L2_DCW	0x80000047	No	Level 2 data cache writes
PAPI_L3_DCW	0x80000048	No	Level 3 data cache writes
PAPI_L2_ICH	0x8000004a	No	Level 2 instruction cache hits
PAPI_L2_ICA	0x8000004d	No	Level 2 instruction cache accesses
PAPI_L3_ICA	0x8000004e	No	Level 3 instruction cache accesses
PAPI_L2_ICR	0x80000050	No	Level 2 instruction cache reads
PAPI_L3_ICR	0x80000051	No	Level 3 instruction cache reads
PAPI_L2_TCA	0x80000059	Yes	Level 2 total cache accesses
PAPI_L3_TCA	0x8000005a	No	Level 3 total cache accesses
PAPI_L2_TCR	0x8000005c	Yes	Level 2 total cache reads
PAPI_L3_TCR	0x8000005d	Yes	Level 3 total cache reads
PAPI_L2_TCW	0x8000005f	No	Level 2 total cache writes
PAPI_L3_TCW	0x80000060	No	Level 3 total cache writes
PAPI_FDV_INS	0x80000063	No	Floating point divide instructions
PAPI_FP_OPS	0x80000066	Yes	Floating point operations
PAPI_SP_OPS	0x80000067	Yes	Floating point operations; optimized to count scaled single precision vector operations
PAPI_DP_OPS	0x80000068	Yes	Floating point operations; optimized to count scaled double precision vector operations
PAPI_VEC_SP	0x80000069	Yes	Single precision vector/SIMD instructions
PAPI_VEC_DP	0x8000006a	Yes	Double precision vector/SIMD instructions
PAPI_REF_CYC	0x8000006b	No	Reference clock cycles

ハードウエアカウンタ Xeon E5 preset とnative

Intel Xeon E5ではPAPI_FP_OPSとPAPI_FP_INSは同じ内容を表示

```
Event name:                  PAPI_FP_OPS
Event Code:                  0x80000066
Number of Native Events:     2
Short Description:          |FP instructions|
Long Description:           |Floating point instructions|
Developer's Notes:          ||
Derived Type:               |DERIVED_ADD|
Postfix Processing String:  ||
 Native Code[0]: 0x4000001c |FP_COMP_OPS_EXE:SSE_SCALAR_DOUBLE|
 Native Event Description: |Counts number of floating point events, masks:Number of SSE double precision FP scalar uops executed|

 Native Code[1]: 0x4000001d |FP_COMP_OPS_EXE:SSE_FP_SCALAR_SINGLE|
 Native Event Description: |Counts number of floating point events, masks:Number of SSE single precision FP scalar uops executed|


$ papi_avail -e  PAPI_DP_OPS
Event name:                  PAPI_DP_OPS
Event Code:                  0x80000068
Number of Native Events:     3
Short Description:          |DP operations|
Long Description:           |Floating point operations; optimized to count scaled double precision vector operations|

 Native Code[0]: 0x4000001c |FP_COMP_OPS_EXE:SSE_SCALAR_DOUBLE|
 Native Event Description: |Counts number of floating point events, masks:Number of SSE double precision FP scalar uops executed|

 Native Code[1]: 0x40000020 |FP_COMP_OPS_EXE:SSE_FP_PACKED_DOUBLE|
 Native Event Description: |Counts number of floating point events, masks:Number of SSE double precision FP packed uops executed|

 Native Code[2]: 0x40000021 |SIMD_FP_256:PACKED_DOUBLE|
 Native Event Description: |Counts 256-bit packed floating point instructions, masks:Counts 256-bit packed double-precision|


$ papi_avail -e  PAPI_VEC_DP
Event name:                  PAPI_VEC_DP
Event Code:                  0x8000006a
Number of Native Events:     2
Short Description:          |DP Vector/SIMD instr|
Long Description:           |Double precision vector/SIMD instructions|

 Native Code[0]: 0x40000020 |FP_COMP_OPS_EXE:SSE_FP_PACKED_DOUBLE|
 Native Code[1]: 0x40000021 |SIMD_FP_256:PACKED_DOUBLE|
```

ハードウエアカウンタ SPARC64 VIIIfx preset とnative

```
$ papi_avail -e PAPI_FP_OPS

Event name:                   PAPI_FP_OPS
Number of Native Events:      4
Short Description:           |FP operations|
Long Description:            |Floating point operations|
Derived Type:                |DERIVED_POSTFIX|

 Native Code[0]: 0x40000010  |FLOATING_INSTRUCTIONS|
 Native Event Description:   |Counts the number of committed floating-point operation instructions.|

 Native Code[1]: 0x40000011  |FMA_INSTRUCTIONS|
 Native Event Description:   |Counts the number of committed floating-point Multiply-and-Add operation instructions.|

 Native Code[2]: 0x40000008  |SIMD_FLOATING_INSTRUCTIONS|
 Native Event Description:   |Counts the number of committed floating-point SIMD instructions of one operation in SIMD.|

 Native Code[3]: 0x40000009  |SIMD_FMA_INSTRUCTIONS|
 Native Event Description:   |Counts the number of committed floating-point SIMD instructions of two operation in SIMD.|
```