Optymalizacja Kodu Na Różne Architektury
Ćwiczenie 4

1 Wprowadzenie

Na zajęciach zapoznamy się z biblioteką Performance API (PAPI). Pozwala ona na zbieranie nisko-poziomowych metryk procesora takich jak m.in. licznik wykonanych instrukcji, transfery pamięci, licznik operacji zmiennoprzecinkowych, itp. Proszę zapoznać się z poniższymi linkami:

1. **biblioteka**,
2. **dokumentacja**,
3. **przykład wyliczania liczby operacji zmiennoprzecinkowych**,
4. **przykład użycia dostępnych liczników**.

2 Ćwiczenia

..

2.1 Serwer

Biblioteka PAPI została przygotowana na serwerze oknra.lab.ii.agh.edu.pl. Aby połączyć się z serwerem OKNRA z poza sieci AGH proszę tunelować się przez serwer BASTION (serwer dostępowy).

```
maciekw@Banach:~> ssh username@bastion.ii.agh.edu.pl
[mwozniak@bastion ~] $ ssh oknra.lab.ii.agh.edu.pl
[mwozniak@dc1a-lab-oknra:~] $
```

Oprogramowanie **PAPI 6.0.0** zostało zainstalowane w lokalizacji **/opt/icl.cs.utk.edu/papi**. Podkatalog **bin** jest dodany do zmiennej **PATH**. Podkatalog **lib** jest dodany do ścieżek wyszukiwania linkera dynamicznego (ld.so.conf). Konfiguracja **pkg-config** jest wgrana w odpowiednie miejsce. Przy kompilowaniu programu używającego PAPI proszę wpisać jako argument do **gcc** wynik wywołania *pkg-config –cflags –libs papi*, zamiast podawać ręcznie ścieżki do plików nagłówkowych i bibliotek.

```
[mwozniak@dc1a-lab-oknra:~] $ pkg-config --cflags --libs papi
-I/opt/icl.cs.utk.edu/papi/include -L/opt/icl.cs.utk.edu/papi/lib -lpapi
[mwozniak@dc1a-lab-oknra:~] $
```

2.2 Uruchomienie przykładu

Polecenie **papi_avail** pozwala zweryfikować dostępne liczniki procesora (kolumna *Avail*). Dodatkowo w kolumnie *Deriv* znajduje się informacja, czy licznik jest pochodną innego licznika. Większość współczesnych procesorów konsumenckich Intel-a ma zablokowane liczniki. Kolumna *name* zawiera nazwę licznika, której można użyć przy inicjalizacji i odczycie liczników w bibliotece PAPI.

```
[maciekw@Banach:] $ papi_avail
Available PAPI preset and user defined events plus hardware information.
--------------------------------------------------------------------------------
PAPI version             : 6.0.0.0
Operating system         : Linux 5.3.18-59.34-default
Vendor string and code   : AuthenticAMD (2, 0x2)
Model string and code    : AMD Ryzen 9 3900X 12-Core Processor (113, 0x71)
CPU revision             : 0.000000
CPUID                    : Family/Model/Stepping 23/113/0, 0x17/0x71/0x00
CPU Max MHz              : 3800
CPU Min MHz              : 2200
Total cores              : 24
SMT threads per core     : 2
Cores per socket         : 12
Sockets                  : 1
Cores per NUMA region    : 24
NUMA regions             : 1
Running in a VM          : no
Number Hardware Counters : 5
Max Multiplex Counters   : 384
Fast counter read (rdpmc): yes
--------------------------------------------------------------------------------
```

```
=========================================================================
  PAPI Preset Events
=========================================================================
    Name        Code      Avail  Deriv  Description (Note)
PAPI_L1_DCM  0x80000000   No     No    Level 1 data cache misses
PAPI_L1_ICM  0x80000001   Yes    No    Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002   No     No    Level 2 data cache misses
PAPI_L2_ICM  0x80000003   No     No    Level 2 instruction cache misses
```

Przykładowy kod programu *chol_papi.c* implementuje podstawową wersję faktoryzacji Choleskiego (patrz ćwiczenie 2) z dodaną funkcjonalnością mierzenia operacji zmiennoprzecinkowych (parametr uruchomieniowy *1*) oraz wybranych dwóch innych liczników procesora (parametr uruchomieniowy 2) - patrz punkty 3 i 4 z listy w pierwszej sekcji.

```
[maciekw@Banach:~] $ gcc $(pkg-config --cflags --libs papi) -lm chol_papi.c
[maciekw@Banach:~] $ ./a.out 100
Time: 1.139000e-03
[maciekw@Banach:~] $ ./a.out 100 1
Real_time: 0.001125 Proc_time: 0.001123 flpops: 338350 MFLOPS: 301.291199
[maciekw@Banach:~] $ ./a.out 100 2
The total instructions executed are 8058740, total cycles 2437809
[maciekw@Banach:~] $
```

2.3 Pomiary

Proszę zapoznać się z listą dostępnych liczników procesora oraz zaproponować wybór dostępnych liczników, które dadzą wgląd w rzeczywiste metryki wydajności programu. W wybranych typach liczników powinny znajdować się między innymi liczniki wskazujące transfery pamięci, jakość wykorzystania pamięci cache, czy GFLOPS.

Proszę wykorzystać kody programu powstałe podczas drugich laboratoriów i obudować je wywołaniami do PAPI. Proszę wykonać serię pomiarów wydajności dla poszczególnych etapów optymalizacji w konfiguracjach dla małej, średniej i dużej macierzy. UWAGA - jednorazowo na procesorze można mierzyć tylko dwa typy wskaźników.

Proszę wykonać porównanie wskazujące jak poszczególne kroki optymalizacji wpłynęły na uzyskane wyniki pomiarów i finalną wydajność.

3 Podsumowanie

Które metody optymalizacji dały najlepsze rezultaty?
Dlaczego? Jakie mechanizmy za nimi stały?