Средства и системы параллельного программирования

Семинар #4. PAPI

Как работать на polus?

Host polus

    HostName polus.cs.msu.ru

    IdentityFile ~/.ssh/id_rsa

    User edu-cmc-sqi19-07

    Port 22

Это строчки, которые нужно вставить в config файл в ~/.ssh, указав свой логин и свой путь до ключа

Затем просто ssh polus

В лоб ssh edu-cmc-sqi19-07@polus.cs.msu.ru -i ~/.ssh/id_rsa

Для передачи файлов на Polus - утилита FileZilla https://filezilla-project.org/

После установки нужно перейти в Менеджер Сайтов, создать новый сайт, ввести адрес polus.cs.msu.ru, порт 22, протокол SFTP, в режиме входа "нормальный" указать свой логин

Для запуска заданий используем планировщик от IBM: http://hpc.cmc.msu.ru/node/250 (если параметры к программе через "--" не сработают, уберите "--"

Отслеживаем задания в очереди через bjobs, bjobs -u all

Как мы можем получать данные о работе приложения?

- Время выполнения (Wall time)
- htop (и прочие) - данные по памяти, загргузка CPU, число потоков, интенсивность ввода-вывода: по факту все подобные утилиты используют статистику из /proc/meminfo, /proc/loadavg и.т.д

Хватает ли этого?

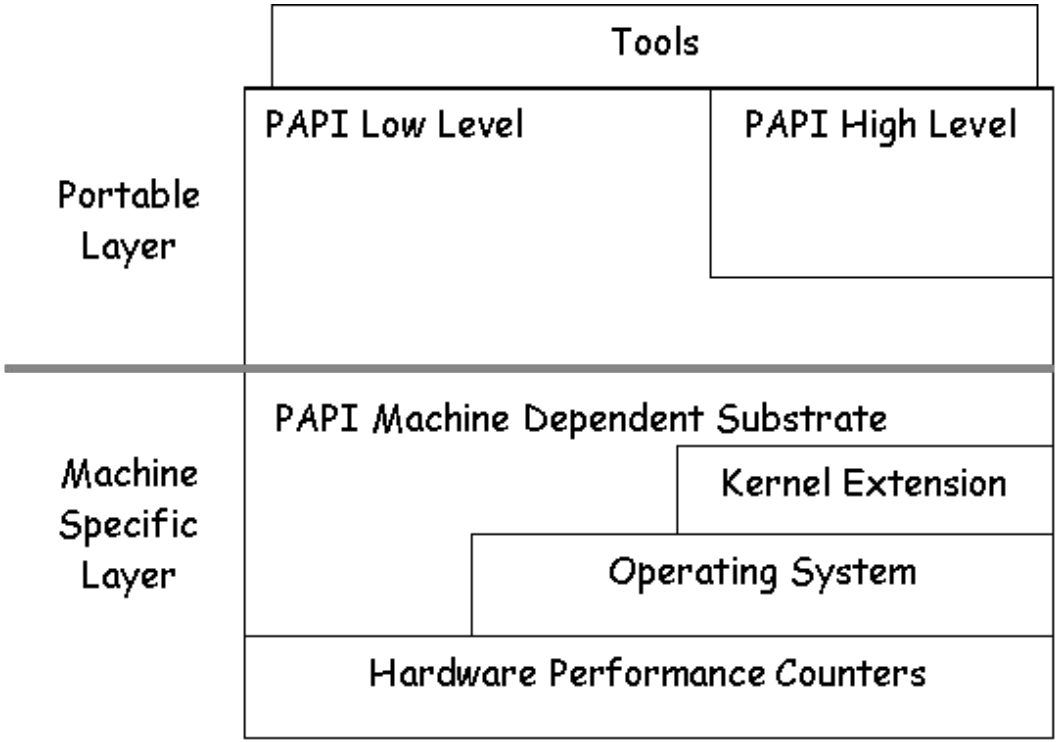
Performance monitoring units (PMU)

PMU — аппаратные регистры для подсчёта событий

Существует некоторый набор утилит, который в том числе позволяет собирать данные с этих счётчиков: perf, Intel Vtune, AMD uprof, LIKWID

PAPI

PAPI (Performance Application Programming Interface) - интерфейс для сбора данных с PMU.



" PAPI attempts to report per-process values, but there are a number of ways in which these might vary from what you would expect "

Установка и использование PAPI

https://github.com/icl-utk-edu/papi/releases/tag/papi-7-2-0b1-t - релиз последней (проверенной) версии

cd papi-7.2.0b1/

Установка: (подробный мануал лежит в INSTALL.txt)

./configure
make

gcc prog.c -lpapi -o prog -I/path/to/papi -L/path/to/papi

Paranoid

```
config.status: creating config.h
config.status: executing genpapifdef commands
configure: WARNING:

*******************************************************************
* Insufficient permissions for accessing any hardware counters.   *
* Your current paranoid level is 4.                               *
* Set /proc/sys/kernel/perf_event_paranoid to 2 (or less) or run as root. *
*                                                                 *
* Example:                                                        *
* sudo sh -c "echo 2 > /proc/sys/kernel/perf_event_paranoid"      *
*******************************************************************
```

sudo sysctl kernel.perf_event_paranoid=-1

Native & Preset events

Native events - события, определенные архитектурой процессора

Preset events - события, введенные в интерфейсе PAPI и каким-либо образом использующие Native events. Нужны для переносимости программ, использующих PAPI

papi_avail (papi_native_avail) - просмотр доступных preset (native) событий

papi_avail -e SOME_EVENT - детальный просмотр события

papi_decode - утилита для детального просмотра всех preset событий

papi_decode -a - то же самое, но только для доступных событий

Вывод papi_avail

```
dlichman@dimPC:~$ papi_avail
Available PAPI preset and user defined events plus hardware information.
--------------------------------------------------------------------------------
PAPI version             : 7.2.0.0
Operating system         : Linux 5.8.0-59-generic
Vendor string and code   : GenuineIntel (1, 0x1)
Model string and code    : 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (140,
0x8c)
CPU revision             : 1.000000
CPUID                    : Family/Model/Stepping 6/140/1, 0x06/0x8c/0x01
CPU Max MHz              : 4700
CPU Min MHz              : 400
Total cores              : 8
SMT threads per core     : 2
Cores per socket         : 4
Sockets                  : 1
Cores per NUMA region    : 8
NUMA regions             : 1
Running in a VM          : no
Number Hardware Counters : 19
Max Multiplex Counters   : 384
Fast counter read (rdpmc): yes
--------------------------------------------------------------------------------
```

EventSet

События считаются группами (**EventSet**) и упираются в число аппаратных счётчиков:

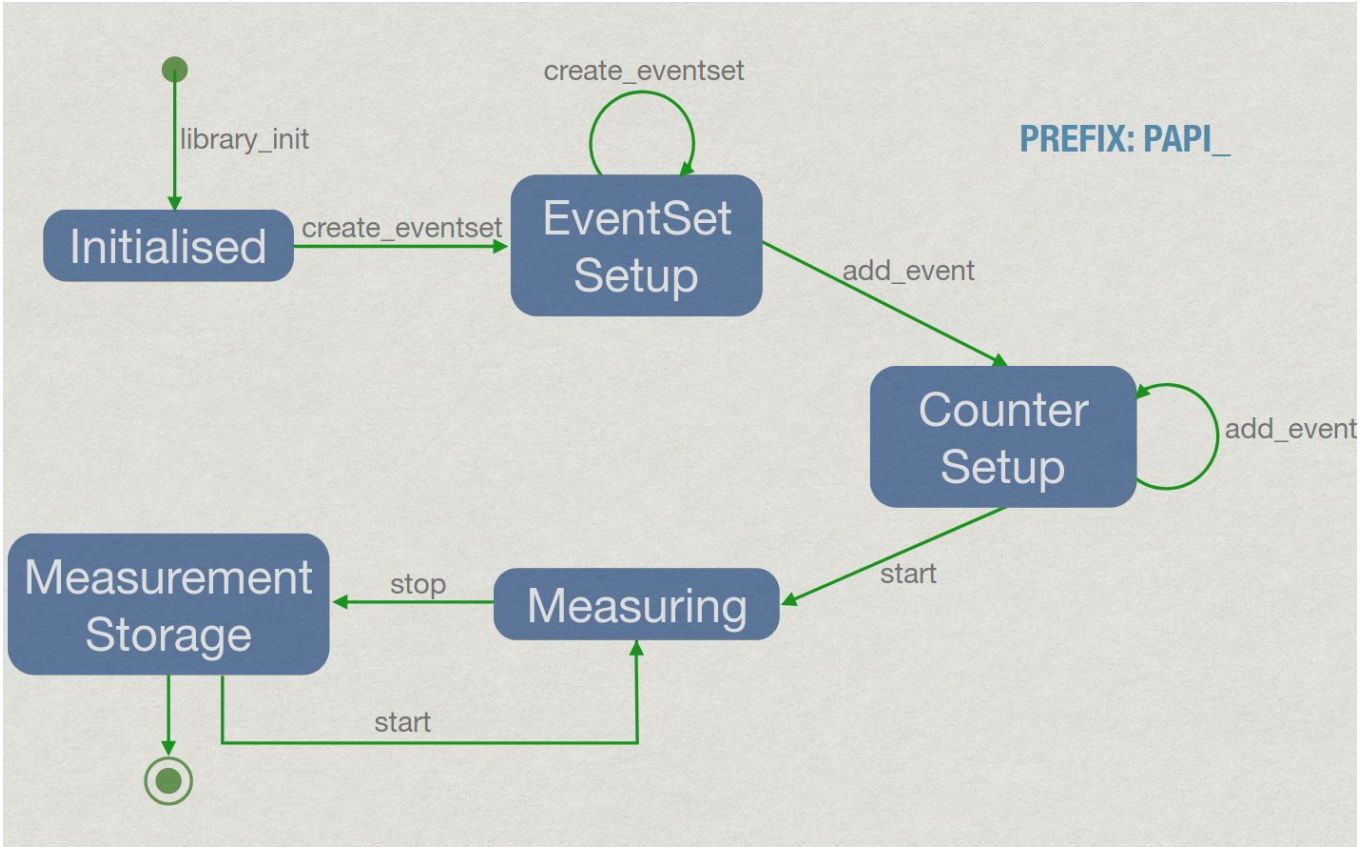
Некоторые события конфликтуют между собой в рамках одного EventSet - они хотят занять один и тот же PMU.

В другом случае, мы хотим собирать больше событий, чем число PMU

В обоих случаях на помощь приходит мультиплексирование

lscpu -1 , и в строке number of counters per logical processor будет число PMU на процессоре

Общая структура Low-level PAPI



Требуемые функции при работе с low-level API

1. `int PAPI_library_init(int version);` // инициализация библиотеки

2. `int PAPI_create_eventset (int *EventSet);` // инициализация EventSet

3. `int PAPI_add_event(int *EventSet, int EventCode);` // добавление события с кодом EventCode в EventSet
   `int PAPI_add_events(int *EventSet, int *EventCode, int number);`

4. `int PAPI_start(int EventSet);` // начало подсчёта event в EventSet

5. `int PAPI_read(int EventSet, long_long *values);` // чтение текущих значений с counters в values
   `int PAPI_accum(int EventSet, long_long *values);` // то же самое, но с прибавлением

6. `int PAPI_stop(int EventSet, long_long *values);` // остановка чтения, чтение значений с counters в values

7. `int PAPI_reset (int EventSet);` // очистка текущих значений в аппаратных counters

Продолжение

1. `int PAPI_cleanup_eventset(int *EventSet);` // удаление всех event из EventSet
2. `int PAPI_destroy_eventset(int *EventSet);` // освобождение данных, выделенных под EventSet. Можно вызывать только для пустого EventSet
3. `int PAPI_shutdown();` // освобождение всех ресурсов, выделенных PAPI

```
#include <papi.h>
```

Как использовать события из papi_avail_native ?

- `int PAPI_event_code_to_name(int EventCode, char *EventName);`
- `int PAPI_event_name_to_code(char *EventName, int *EventCode);`
- `int PAPI_query_event(int EventCode);` // проверка возможности сбора данного Event на данной архитектуре (0 = OK)

В качестве EventName можно использовать и Preset и Native events

High-level API

```
int PAPI_flops_rate(int event, float *rtime, float *ptime, long long *
flpops, float *mflops);
```

// Составная PAPI-команда, позволяющая снять с разработчика работу с Event

**rtime* -- total realtime since the first PAPI_flop_rate() call

**ptime* -- total process time since the first PAPI_flop_rate() call

**flpins* -- total floating point instructions since the first PAPI_flop_rate() call

***mflops -- Mflops/s achieved since the latest PAPI_flop_rate() call**

Мультиплексирование

papi_multiplex_cost - утилита для просмотра накладных расходов от использования мультиплексирования на выбранной архитектуре

```
int PAPI_multiplex_init (void); // инициализация мультиплексирования

int PAPI_set_multiplex(int *EventSet); // настройка EventSet на мультиплексирование
```

Задание

Реализовать программу, использующую интерфейс PAPI, с помощью которой экспериментальным путём определить размер L2-кеш памяти.

Примерный сценарий: (*при запуске программу через taskset -c 0 ./my_app лучше прибить к одному ядру, чтобы значение с PMU собиралось более точно*)

- Выделить и инициализировать массив размера N байт случайными значениями
- Провести сортировку массива (лучше написать свою, которая будет гарантированно однопоточной)
- Посмотреть на значение счетчика, соответствующего числу промахов в L2, за время выполнения сортировки
- Повторять эти действия для различных размеров массива (удобно начать с 4Кб и прибавлять по 4Кб каждый раз)
- Определить размер массива, при котором число промахов впервые выросло наиболее заметно

  Реализация своего, отличного от описанного, сценария приветствуется

  Помимо L2-события из preset событий, найдите интересное вам native событие (можно связанное с L2) и соберите его

  Написать отчёт, в котором будет описан ход ваших рассуждений и график (можно таблицу) значений счётчика, сравнить вывод по размеру L2 с правильным значением

  Подсказка: между 1 и 2 шагами можно очистить кеш (например, выделить ещё массив, заполнить его чем-то и посчитать максимум). Это должно позволить вам точнее определить предельный размер массива.

Дедлайн: 20.10, 27.10