Exercises for Architectures of Supercomputers

4th Exercise, 26./27.11.2019

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Intro to hardware performance counters

- Modern processors feature **hardware performance counters**, which enable logging of certain **events** that occur on the processor
- Mode of operation
  - Performance counters can be programmed with a particular event (e.g., the number of branch instructions executed)
  - Once programmed, the counter is incremented whenever the event (e.g., a branch instruction) occurs
  - Some events can be recorded per core, others for the whole processor
  - The construction of complex metrics is possible by combining different events (processors typically feature multiple counters)
  - The number of supported events is significant (see, e.g., https://download.01.org/perfmon/IVB/)

Accessing counters with LIKWID

- The Tool `likwid-perfctr` from the LIKWID tool suite (developed at RRZE) provides an easy way for developers to program and read performance counters
- It expects the following command-line arguments:
  - `-g GROUP` where `GROUP` is a pre-defined performance group (you can list all available groups with `likwid-perfctr –a`); **OR**
  - `-g EVENT:PMC`*x* where `EVENT` is a particular event and `PMC`*x* the performance counter to be programmed with the specified event. Multiple event-counter entries are separated by a comma
  - `-C <core>` Core/cores on which the event should be measured
  - `<bin> <bin args>` An executable binary and its arguments
- **Note:** To get access to the hardware performance counters, you must allocate a node with the `likwid` property:

```
$ qsub -lnodes=1:ppn=40:f2.2:likwid,walltime=1:30:00 -I
```

Instrumentation with LIKWID

- By default, `likwid-perfctr` will measure the events during execution of the whole binary

- However, in most cases, you will be interested in the processor's behavior during a particular loop of your program

- To this end, `likwid-perfctr` allows the **instrumentation** of your code. This means you can add **markers** to your code which instruct `likwid-perfctr` to measure events only for regions surrounded by these markers

Using LIKWID's marker API

- To use the marker API, you need to include `likwid.h` as additional header
  - Because the header not in a directory known to the compiler, you must tell the compiler where to find it using the `-I<path>` command-line argument, where `<path>` is the directory containing the header
- Additionally you have to link against the LIKWID and pthread libraries, which can be done by specifying the `-llikwid` and `-pthread` command-line arguments when compiling
  - Because the LIKWID library is not in a directory known to the compiler, you must tell the compiler where to find it using the `-L<path>` command-line argument, where `<path>` is the directory containing the header
- To enable the marker API, the `LIKWID_PERFMON` macro must be set
  - This can be done by supplying the `-DLIKWID_PERFMON` command-line argument to the compiler
- To measure events only for the instrumented part of you code, use the `-m` (for marker API) command-line argument for `likwid-perfctr`

Using LIKWID's marker API

- Instrumenting your code
  - You need to call `LIKWID_MARKER_INIT` once before you can use any markers
  - Surround your region of interest by the `LIKWID_MARKER_START` and `LIKWID_MARKER_START` markers
  - Finally, before your program ends execution, you must call `LIKWID_MARKER_CLOSE`
- You can find an example on how to use the marker API here: https://github.com/RRZE-HPC/likwid/wiki/likwid-perfctr#using-the-marker-api

Exercise

- Run your code from last week and measure
  - The total number of branch instructions executed in the `benchmark()` function that uses branches
  - The total number of branch mispredictions in the `benchmark()` function that uses branches
  - Because you are only interested in the events occurring inside the `benchmark()` function, you need to instrument your code using LIKWID's marker API

- To get the required data, use program
  - `PMC0` with the event `BR_INST_RETIRED_ALL_BRANCHES`; and
  - `PMC1` with the event `BR_MISP_RETIRED_ALL_BRANCHES`

- From the result, derive the misprediction ratio (i.e., the percentage of total mispredictions) in the `benchmark()` function

Exercise: Hints

- To use LIKWID, you must load it using module system
  - Use `likwid/4.3.4`
  - To determine the paths of LIKWID's header file and libraries, you can use the `which` command:
    - `emmy$ which likwid-perfctr`
      `/apps/likwid/4.3.4/bin/likwid-perfctr`
    - If the `likwid-perfctr` binary is in `/some/path/bin` the
      - LIKWID headers will be in `/some/path/include` and the
      - LIKWID libraries will be in `/some/path/lib`