TU Wien
Faculty of Informatics
Research Group for Parallel Computing

High Performance Computing
Project 1

Summer semester 2018
Submission deadline: 14 May 2018, 23:55 (strict)
Contact: hunold@par.tuwien.ac.at

1 Obtain Roofline Model for Two Machines (8 points = 2 * (2+1+1) points)

The task is to obtain a Roofline model for one (multicore)-machine of your choice and for `mars`. Do the following steps for each of the two machines (thus, the number of points refers to one machine):

1. Refer to the specification of the processor
   - to calculate the theoretical peak performance (GFLOP/sec) of your processor (**all physical cores**),

   and run the STREAM or NUMA-STREAM[1] benchmark (on all cores)
   - to measure the peak memory bandwidth.

   (2 points)
2. Document the parameters used to compile the STREAM/NUMA-STREAM benchmark and include the output of the benchmark. (1 points)
3. Plot the Roofline model for the given machine as shown in the lecture. Label the plots correctly. (1 points)

[1] `https://github.com/larsbergstrom/NUMA-STREAM`

2 Determine the arithmetic intensity (AI) of one benchmark kernel (17 points)

2.1 Preparation

- This exercise will be done on `mars`.
- Download `Rodinia 3.1` from `http://www.cs.virginia.edu/~kw5na/lava/Rodinia/Packages/Current/rodinia_3.1.tar.bz2`.
- Compile the `kmeans` OpenMP benchmark using `gcc` with OpenMP support.

2.2 Tasks

1. Document the compiler (`gcc`) version and the compilation flags used to compile the benchmark. (1 point)
2. Determine the **arithmetic intensity** (single precision floating point) empirically using `LIKWID`. (2+2 points)
   a) Experimentally obtain the values for the GFLOPS done and GBytes transferred for all given inputs (`100, 204800.txt, 819200.txt, kdd_cup`).

   ```
   /opt/likwid/bin/likwid-perfctr -f  -C 60-69 -g FLOPS_SP ./kmeans_openmp/
       ↪ kmeans -n 10 -i ../../data/kmeans/819200.txt
   /opt/likwid/bin/likwid-perfctr -f  -C 60-69 -g MEM ./kmeans_openmp/kmeans
       ↪  -n 10 -i ../../data/kmeans/819200.txt
   ```

   b) Compute the AI (for which you need the GFLOPS) for all inputs (`100, 204800.txt, 819200.txt, kdd_cup`) and for 10 and 80 cores (physical cores, we do not use hyper-threading here).

   ```
   /opt/likwid/bin/likwid-perfctr -f -C 0-9  -g FLOPS_SP ..
   /opt/likwid/bin/likwid-perfctr -f -C 0-79 -g FLOPS_SP ...
   ```

3. Plot the GFLOPS obtained for each input file into the Roofline models for `mars`. There should be one Roofline model for 80 cores (the model that was created in the previous task) and one model for one socket (10 cores). (2 points)
4. Inspect the `kmeans` code and discuss what the expected maximum performance of this code could be! (hint: check whether we have add/mul imbalance or vectorized operations (AVX, SSE)) Insert this new ceiling into your Roofline plot and explain your experimental findings by comparing to this new ceiling. (2 + 2 points)
5. Since your measurements will have covered the entire code and not just the `kmeans` kernel, we will now only measure the performance of that kernel using the `LIKWID` marker API. See `https://github.com/RRZE-HPC/likwid/wiki/TutorialMarkerC` for an example of how to use this API.

- Use the `LIKWID` marker API to limit the performance measurements with `LIKWID` to the specific `kmeans` kernel (the call to `kmeans_clustering`). (2 points)
- Obtain the AI for `kmeans` (with the marker API) for all input files and for 10 and 80 cores, and create new Roofline plots (one for 10 and one for 80 cores) with these values. (2 points)
- Compare the results obtained with and without the marker API (2 points).

Some Hints

- When using the `LIKWID` marker API, you need to use `#include "likwid.h"` in all source files that use these `LIKWID` macros.
- You need to set the `PATH` and `LD_LIBRARY_PATH` to `LIKWID`.

  ```
  export PATH=/opt/likwid/bin:$PATH
  export LD_LIBRARY_PATH=/opt/likwid/lib
  ```

- You will need to link the `LIKWID` library (`-llikwid`).

Additional Bonus Points

You can obtain 3 additional points by writing the entire report in a Jupyter notebook. In this case, use the same headings as shown in the LaTeX template. These points can be used to compensate point deficits in the second assignment.

What to hand in?

Hand in the source code of all algorithms including a Makefile. Zip or tar all files and name your file like this: `your_lastname_hpc_project1.{tar.gz|zip}`. Write a report (approximately 4–8 pages) that answers the questions in detail.