EASY-Assignment #2: Energy Models

In this exercise we implement different energy models. These energy models can be provided with different inputs and are based on energy measurements as conducted in the first assignment. We also examine the differences in precision and accuracy for different model parameters.

Goals of this assignment

- Create energy models for hardware and software
- Determine the influence of hardware states on software energy demand

2.1 Software Energy Model

Create an energy model for the `julia` program. Vary the number of pixels in the result image (`-W`, `-H`, `-R` parameters) and estimate the amount of energy needed to execute the program with these parameters. Use linear regression to form an energy model. Plot the energy measurements and the estimations of your model to visualise its precision and calculate the *mean squared error* (MSE).[1]

For assignments 2.1 to 2.4 you should fix the processor speed to 2.80 GHz using the `cpupower`[2] tool for the model creation process to allow for reproducible energy demand values. As baseline for your energy model use RAPL or physical energy measurements to determine the energy demand.

2.2 Simple Hardware Energy Model

Refine the per-pixel energy model: Use `perf stat` to count the number of instructions executed by a `julia` run. Estimate the amount of energy needed for each instruction, and create another energy model. Plot your results and calculate the MSE.

2.3 Model Comparison

Vary the number of iterations (`-I` parameter) of the `julia` program and evaluate both the per-pixel and the per-instruction energy model. Which model gives better results? Why?
Plot your results and calculate the MSEs.

2.4 Complex Hardware Energy Model

Similar to the per-instruction energy model, use `perf stat` with multiple performance monitoring counters (PMCs) to create a more precise energy model. Which PMCs do you consider suitable for an energy model? Why?
Plot your results and calculate the MSE.

The binary implementing your energy model should provide the same interface as the `julia` program and be named `julia_energy`. You can execute the `julia` program in your binary to obtain PMC values for a given set of parameters, but you should redirect the output to `/dev/null`. The result of the energy model (i.e., the energy demand estimation) should be printed on `stderr`. The submission of this assignment will be used for the competition (see below).

Example output:

```
$> ./julia_energy -W4 -H4 -X0 -Y0 -R512 -I1000
   12.34 J
```

2.5 Hardware State Models

Use `cpupower` to vary the processor speed. Use the default `julia` parameters (i.e., image width and height, center coordinates, resolution, and number of iterations) and create an energy model that describes how the processor frequency affects the energy demand. Plot your results and calculate the MSE.

[1] `https://en.wikipedia.org/wiki/Mean_squared_error`
[2] `cpupower frequency-set --governor performance --min 2.80GHz --max 2.80GHz`

Competition

Who creates the best (→ accuracy, precision) energy model? For the competition we use your submission of the complex hardware energy model as described in assignment 2.4. The best model of each semester will be displayed in a Hall of Fame on the website. You can utilize any information you want, except information provided by RAPL as the baseline measurements will be based on RAPL.

We compare all submissions using the following test environment:

- The program under test is the `julia` program, where the output is redirected to `/dev/null` to minimize the interference of `I/O` operations.
- The model is tested with different parameters for the `julia` program. The values are in same the order of magnitude as the default values are.
- The processor speed is fixed at 2.80 GHz.
- The metric to evaluate the model is the *mean squared error* (MSE).
- The baseline measurements are conducted with RAPL, using the package domain.

Notes

- Material: The `julia` program
- Deadline: 2020-07-02 12:00