Green Lab - LAB 1

Experiment Runner & EnergiBridge

Hui & Andrei

Before getting started

Requirements:

A Linux or MacOS PC is needed for today's lab; ex3 requires python 3.11; ex1 and 2 need 3.8

Quick Look:

Lab Instructions Link

Experiment Runner & EnergiBridge Video Demo

VU

Road Map

We will go through 3 topics:

- Setting up the environment (if not done already);
- Getting familiar with Energibridge & Experiment-Runner
- Conduct a small experiment on your own.

Goal: Learn to use both tools to conduct experiments and collect data

Link: GitHub Repo

https://github.com/DaydreamCreator/green-lab-lab1/tree/main

VU

General Overview

- **ExperimentRunner**: An Experiment Controller
It is a generic framework to automatically execute measurement-based experiments across platforms. The experiments are user-defined, can be completely customized, and expressed in python code.

- **EnergiBridge**: An Energy Measurement Tool
It is a cross-platform energy measurement utility that provides support for Linux, Windows, and MacOS, as well as Intel, AMD, and Apple ARM CPU architectures. This tool is designed to collect resource usage data for a command to execute and to output the data in a CSV format.

VU

Setup and Example

Experiment Runner Setup

Step 1- Fork the Experiment Runner repo and clone the fork to your computer, follow the installation instructions of the repo

```
git clone https://github.com/S2-group/experiment-runner.git
cd experiment-runner/
pip install -r requirements.txt
```

Step 2- Try running the following command from the root directory of the repo to check if the example program ran correctly (i.e., no errors)

```
python experiment-runner/ examples/hello-world/RunnerConfig.py
```


VU

Setup and Example

EnergiBridge Setup

Step 1- Follow the installation instructions for your platform from the [EnergiBrigde repo]

Step 2- Check if EnergiBridge is installed

```
energibridge -h
```

EX 1 – Test out EnergiBridge

We will measure the total CPU energy and the total memory used of 5 seconds:

```
energibridge --summary -o test.csv sleep 5
```

Example Output in terminal:

```
Energy consumption in joules: 253.86045820617682 for 5.1230927
sec of execution
```

Open the generated `test.csv` file to check details of raw EnergiBridge file. Energy consumption calculated by EnergiBridge is **system dependent**, and output can change accordingly.

VU

EX 2 – Test out Experiment Runner

We will run the example found in `examples/hello-world-fibonacci/RunnerConfig.py`

The goal of this experiment is to collect execution metrics of the Fibonacci sequence with different input sizes.

Take a look at the the `config_run_table_model()` in the `RunnerConfig` class to understand how the experimental variables and parameters are defined.

EX 2 – Test out Experiment Runner

To run this example, we use the following command:

```
python3 experiment_runner/ examples/hello-world-fibonacci/RunnerConfig.py
```

This will create:

- The output directory (at the path specified in the `RunnerConfig.results_output_path`);
- Subdirectories for each run containing the `energibridge.csv` file with raw data;
- The `run_table.csv` data with all aggregated data

VU

EX 2* – Go further:

Run it yourself

* Change factor levels (e.g., add more Fibonacci sizes) and rerun.

* Increase or decrease repetitions to see how result stability changes.

* Add new data columns (e.g., `cpu_temp`) and extend `populate_run_data()`.

* Compare performance between Fibonacci implementations (`iter` vs `mem` vs `rec`).

Small experiments

1. Predict which Fibonacci type (`iter`, `mem`, or `rec`) performs best at large problem sizes. Run the experiment and check if your prediction holds.

2. Try disabling/enabling shuffling—does the order of runs affect performance or stability?

3. Visualize your results (runtime vs. problem size) in a simple plot.

VU