Decoupling Energy Consumption and Execution Time in High Performance Computing

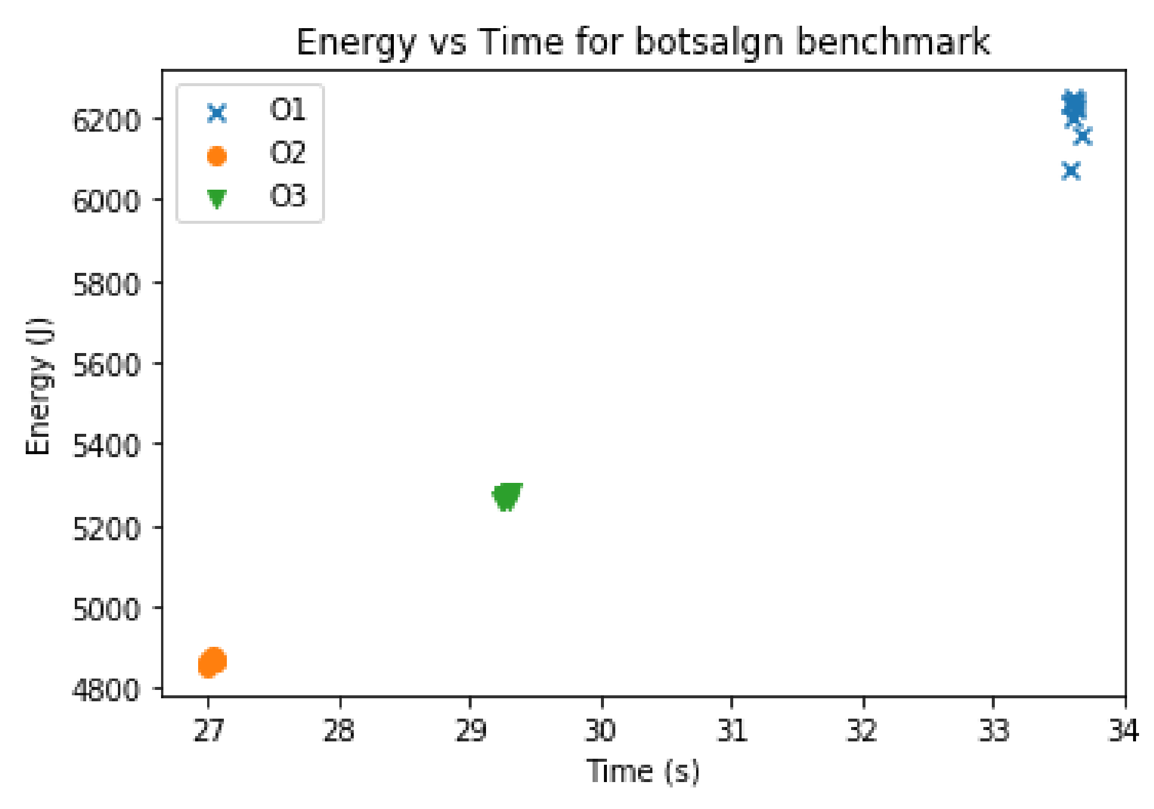
University of Bristol
Department of Computer Science

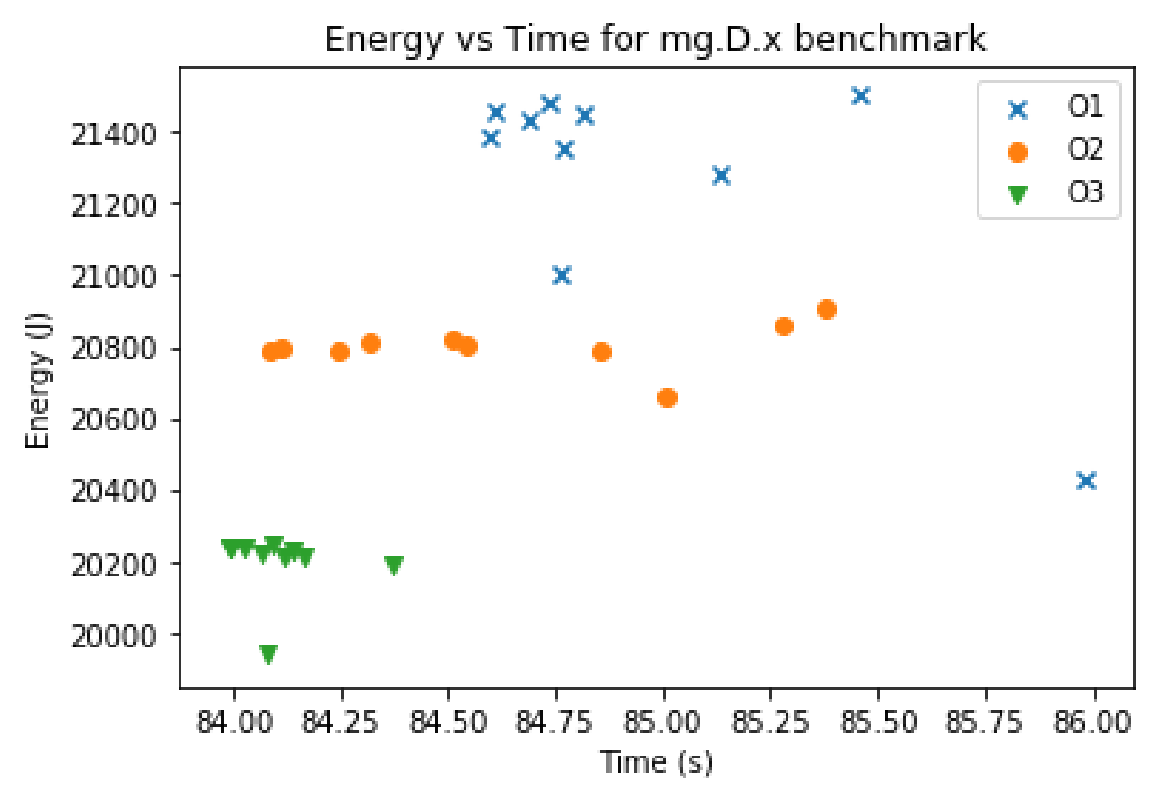
Overview

- Supercomputers use an enormous amount of **energy** which
  - is expensive
  - has an environmental impact
  - is a limiting factor in scaling of HPC systems.
- The aim of this project is to **reduce energy consumption** in multi-threaded, HPC programs using **compiler optimisations.**
- The project focuses on the **GCC compiler** and follows the method of the **Milepost** study [1] but focussing on energy consumption not just execution time.
- **Hypothesis**: optimising for energy consumption will yield greater reductions than optimising for execution time.

Benchmark Programs

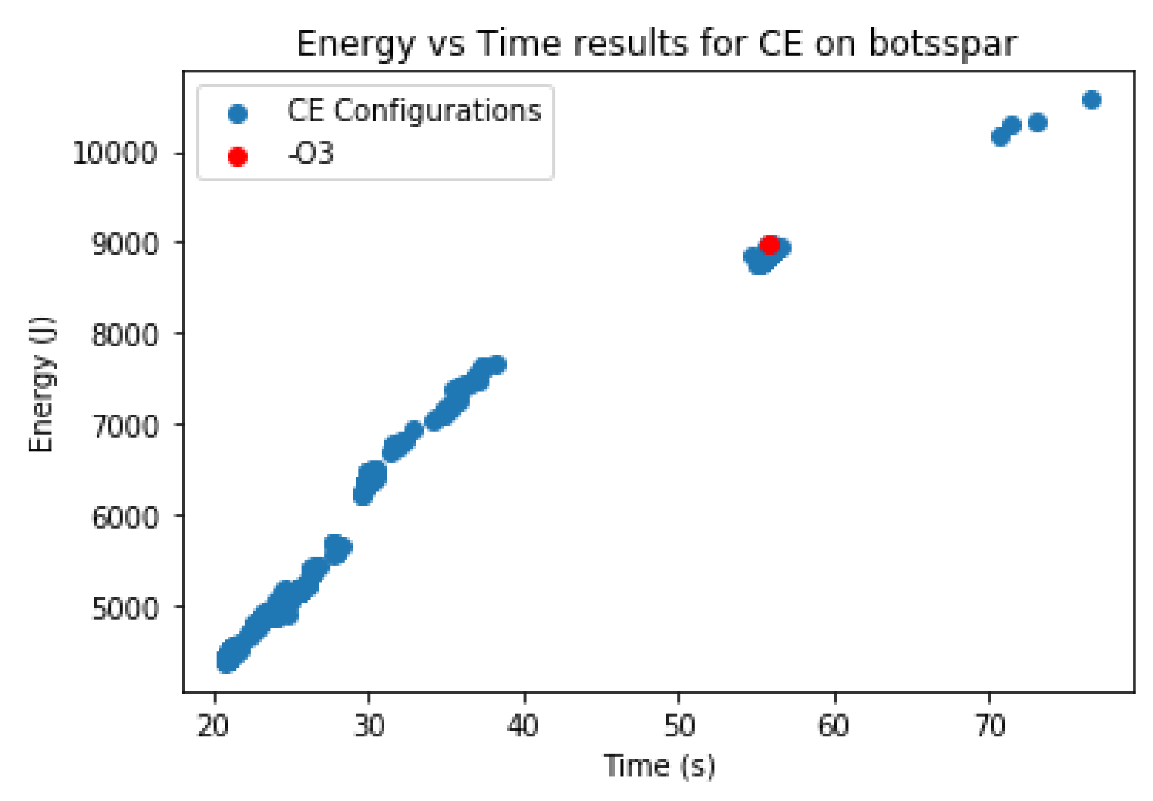
- Using **OpenMP** benchmark suites
  - NAS Parallel Benchmarks
  - SPEC OMP2012
- Subset of benchmarks chosen based on **stability** of runtime and energy:





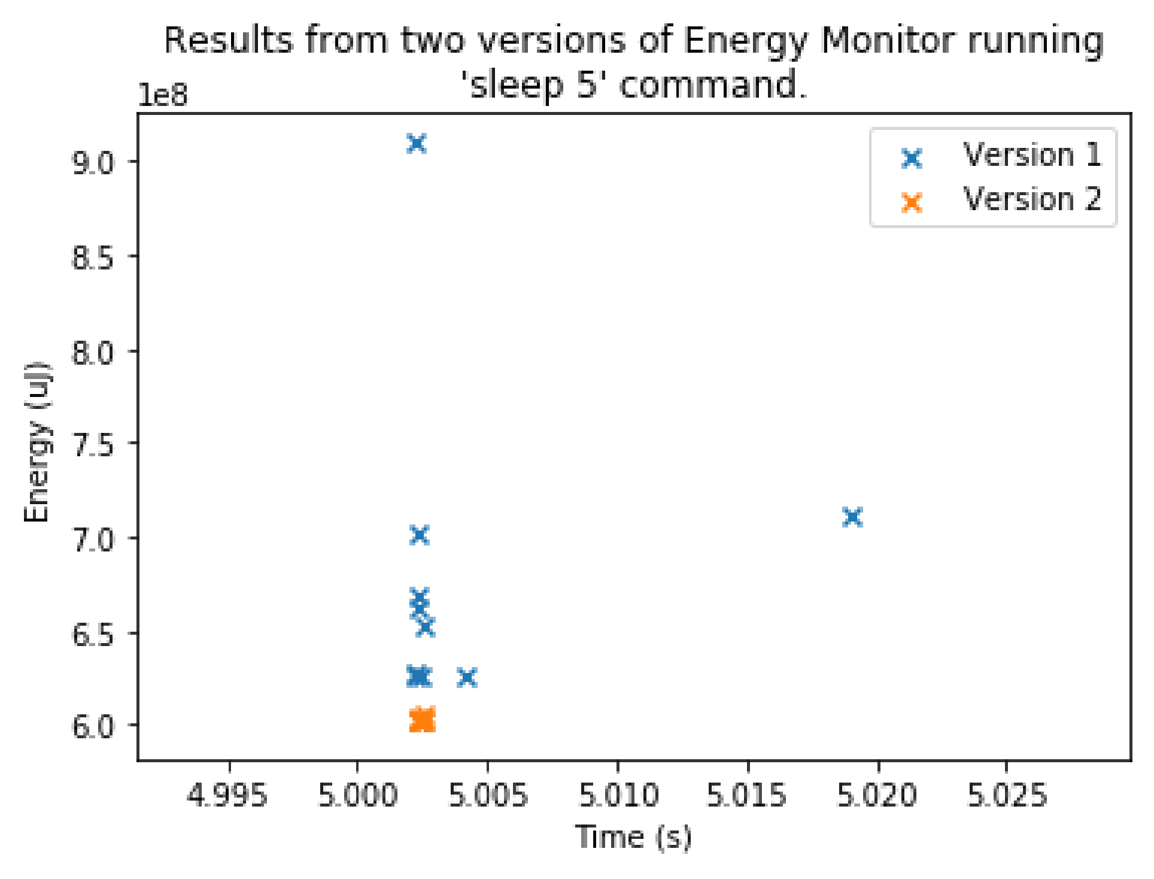
Iterative Compilation

- Iterative compilation algorithm **Combined Elimination** [2] used to search compiler optimisation space.
- Space is very large (195 optimisations in GCC 7.2). Cannot exhaustively search.
- Explores the how compiler optimisations affect energy and time.
- Provides training data for machine learning techniques.



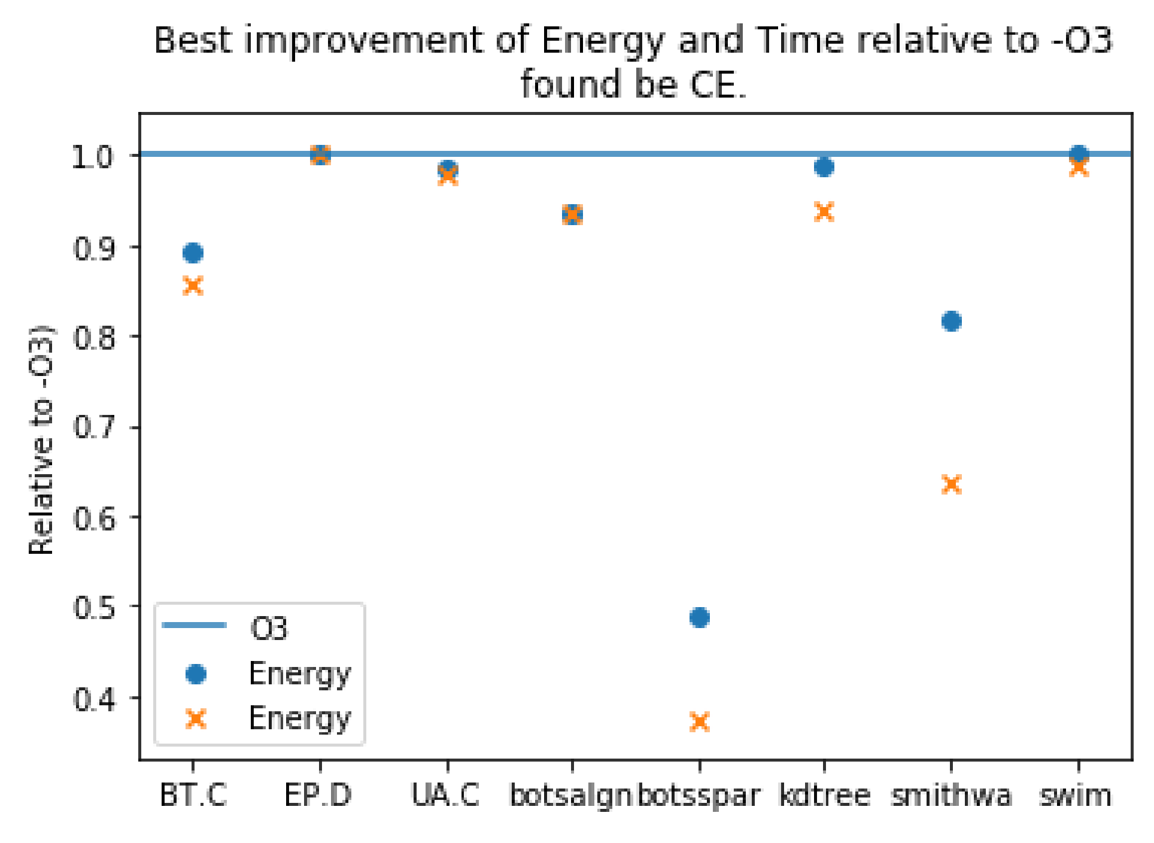
Energy Measurements

- Energy measurements available through the Intel Running Average Power Limit (**RAPL**) feature.
- Measures processor package and DRAM energy consumption.
- Energy monitoring tool written in C.
- Multiple versions of monitoring tool developed and tested for **stability**:



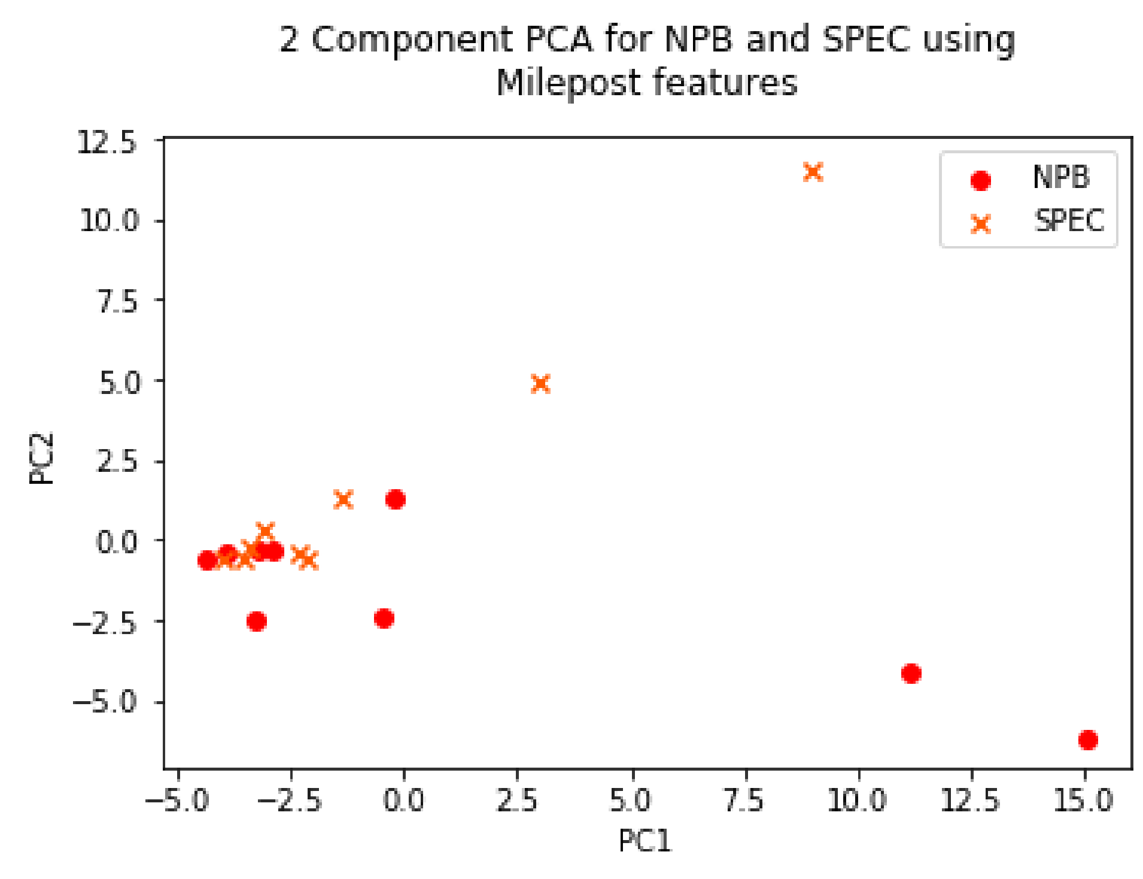
Energy vs Time

- Energy vs Time relationship currently unknown in multi-threaded applications.
- Previous work has shown strong linear correlation in single threaded applications [3].
- Results of this project indicate
  - a linear correlation between energy and time
  - energy reduction not always equal to time reduction



Machine Learning

- Use the **1-Nearest-Neigbour** (1NN) technique to **predict** good compiler configurations.
- Use 65 features identified in Milepost study [1] extracted from source code.
- Explore a new feature using the **Seven Dwarfs** classification system [4].
- Benchmarks analysed for **suitability** to use in 1-NN:



References

[1] G. Fursin, Kashnikov, Y., and A. Memon, "Milepost GCC: Machine Learning Enabled Self-tuning Compiler," Int J Parallel Prog, vol. 39, no. 3, pp. 296–327, 2011.

[2] Z. Pan and R. Eigenmann, "Fast and Effective Orchestration of Compiler Optimizations for Automatic Performance Tuning," in Proceedings of the International Symposium on Code Generation and Optimization, ser. CGO '06. New York, USA: IEEE Computer Society, 2006, pp. 319–332.

[3] J. Pallister, S. Hollis, and J. Bennett, "Identifying Compiler Options to Minimise Energy Consumption for Embedded Platforms," The Computer Journal, vol. 58, no. 1, 2013.

[4] K. Asanovic et. Al. "The Landscape of Parallel Computing Research: A View from Berkeley," Electrical Engineering and Computer Sciences, University of California at Berkeley, Technical Report UCB/EECS-2006-183, Dec. 2006.

David Greasley
Supervisor: Dr. Oliver Ray