Solar Data Tools Profiling Review

Summary

When creating workloads to run in the cloud, it is important to be aware of performance characteristics as they can have an impact on processing speed and overall cloud costs. As part of the effort parallelizing the solar-data-tools pipeline, the team profiled the memory and CPU utilization of the run_pipeline function in an effort to find potential optimizations. Through our analysis, we found the majority of the CPU time and memory usage occurred in CVXPY and was thus out of scope for optimization.

Methods

The profiling task involved loading 3 solar data files from disk and serially calling the run_pipeline function on each file. Sciagraph and Pyflame were used to run the analysis and generate flamegraph visualizations that show the relative amount of resources used by each module (note that they do not provide absolute performance times).

Sciagraph provides both CPU and memory profiling while Pyflame only offers CPU profiling. While Sciagraph can provide both CPU and memory results, Pyflame was used in conjunction with Sciagraph because the visual output was preferred. The tests were run on an AWS t2.xlarge EC2 instance.

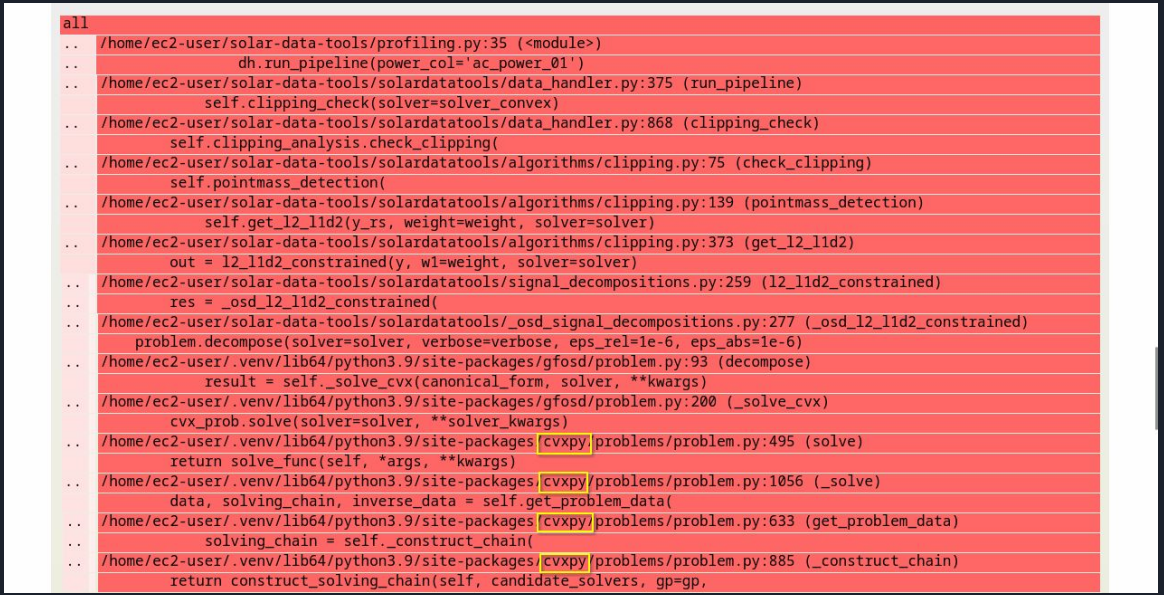
How to read a flame graph

When reading a flamegraph, the vertical elements represent stack frames, while the horizontal width of an element represents the relative amount of a resource consumed. Pyflame graphs (like the one below), are built from the bottom up. Therefore, the stack frame responsible for consuming the CPU at the time of measurement will be the closest to the top. Sciagraph graphs are build from the top down, so relevant stack frames will be near the bottom.

CPU Flamegraph - (Pyflame)

*Solar-data-tools code is color coded red. Note how most of the CPU time is spent on the blue and green code at the top of the graph.

Memory Flamegraph - (Sciagraph)



*This flamegraph is top down compared to the Pyflame graph. However, we can see that code within CVXPY utilized over 90% of the total memory used.

Testing Discrepancies

Prior to running the tests on AWS, the team attempted to profile the code running on our local development machines and received profiling results where an inordinate amount of time was spent on the clipping check. We believe these discrepancies may be caused by other software using resources on the development machines during testing. The following results are inconclusive but have been included in the following slides for completeness.

CPU Flamegraph - (Pyflame)

System Specifications

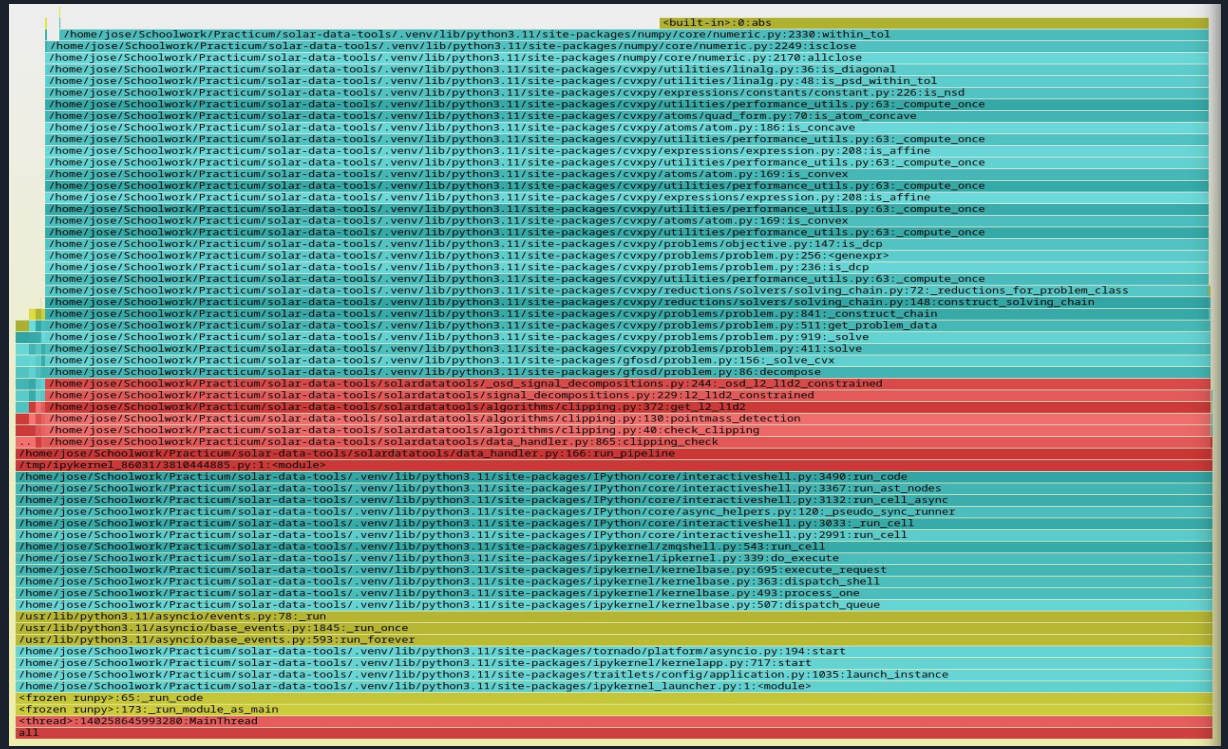
OS: Win11 Home Edition
RAM: 16 GB
CPU: AMD Ryzen 7

CPU Flamegraph - (Pyflame)

System Specifications

OS: MacOS
RAM: 16 GB
CPU: Intel i7

CPU Flamegraph - (Pyflame)



System Specifications

OS: Manjaro Linux
RAM: 32 GB
CPU: Intel i7 9700K