Aaron Tan
71575415
E/I Yr.3

Comparing the usage of system resources of industry-leading text editors

Introduction:

Text editors are one of the key tools used by programmers to effectively write code. At their most basic, they facilitate the action of writing text to files. Many will also however include syntax highlighting, auto-indentation, and auto-completion. Integrated development environments take things a step further by including debugging, compilation, and execution features as well. This report will examine three leading text editors; Sublime Text, Atom, and VS Code. I have been constantly switching between these three editors in an effort to find one that fits my workflow the best. I prefer them to IDEs because they each have plugin support that makes them more extensible than a bare-bones text editor, and yet are not as feature-bloated and therefore slow as a full-fledged IDE. I decided to test the memory and CPU usage of each application as I primarily develop on a macbook that is now several years old and want to determine the most lightweight of the three applications. A program has been written to log the CPU and memory usage of each text editor and tests will be run under different conditions to assess the efficiency of each application.

Tests will be conducted under five different scenarios:

- Idle, Clean
  - Measure each application running as is without user interaction, *no* plugins installed.
- Idle, Plugins
  - Measure each application running as is without user interaction, *with* plugins installed.
- Heavy
  - Measure each application as it attempts to open a large text file (without plugins).
- Realistic, clean
  - Measure each application with a realistic project folder open, user interaction, and *no* plugins installed.
- Realistic, plugins
  - Measure each application with a realistic project folder open, user interaction, and *with* plugins installed.

Aaron Tan
71575415
E/I Yr.3

Additional notes:

While VS Code and Atom are written in Javascript/Typescript and run Electron, which is an instance of the Chrome v8 engine, Sublime Text is written in C++ and python. This could factor into the base resource usage as Sublime Text will not be hindered by needing to constantly run Electron in the background. This can be confirmed by checking the process list. VS Code and Atom both run five additional ‘helper’ processes alongside the main process. It can also be noted that Atom is open-source and therefore is regarded higher in terms of privacy and security. However, for the purposes of this test, that will not be taken into consideration.

In order to ensure the program can be run on different operating systems in case others would like to replicate the tests, a python library named *psutil* was used as a cross-platform system monitoring tool. Using *psutil* a program was written that first searches the process list for processes that relate to each text editor that is to be examined. This was initially done manually until the process names were found and could be used to automate the gathering of processes. The program then records the total CPU and memory usage of an editor’s processes as a percentage of the maximum use limit. All of the gathered data is written to a CSV file for data processing.

Aaron Tan
71575415
E/I Yr.3

Data:

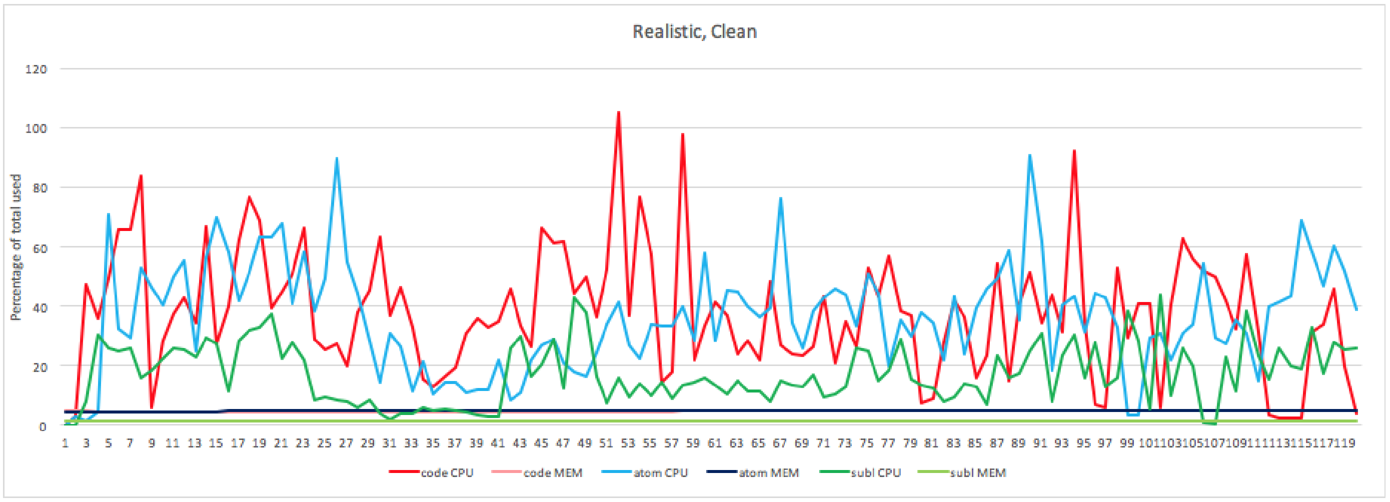

Fig 1. Tests with a realistic project open, switching between files, editing lines, without plugins.

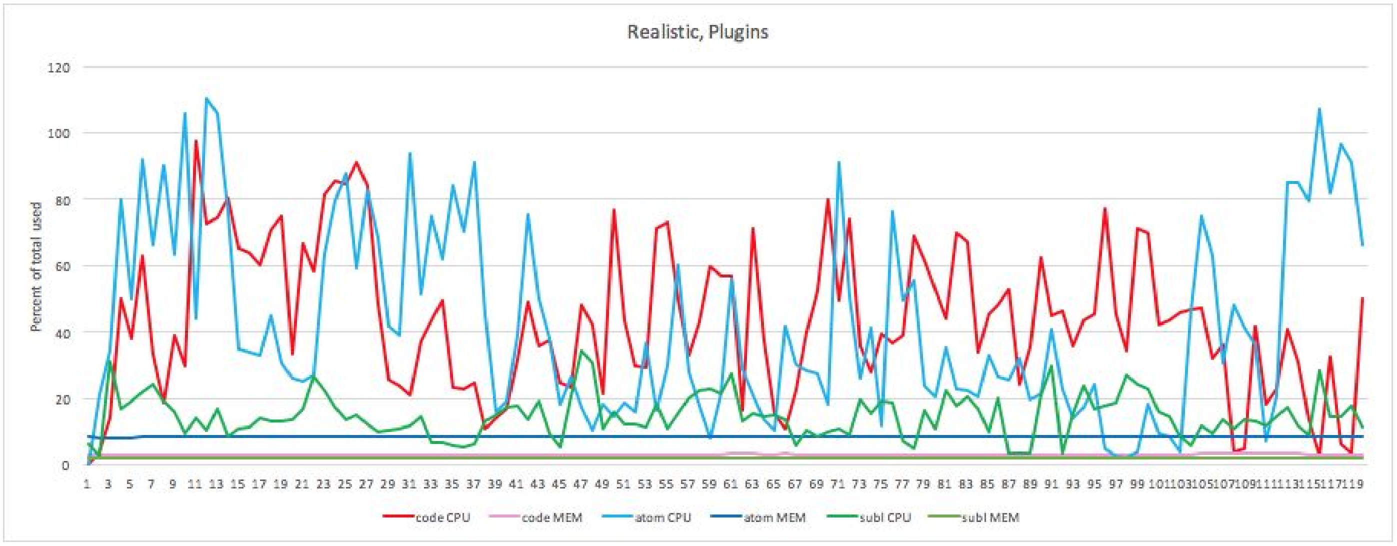

Fig 2. Tests with a realistic project open, switching between files, editing lines, with plugins.

Aaron Tan
71575415
E/I Yr.3

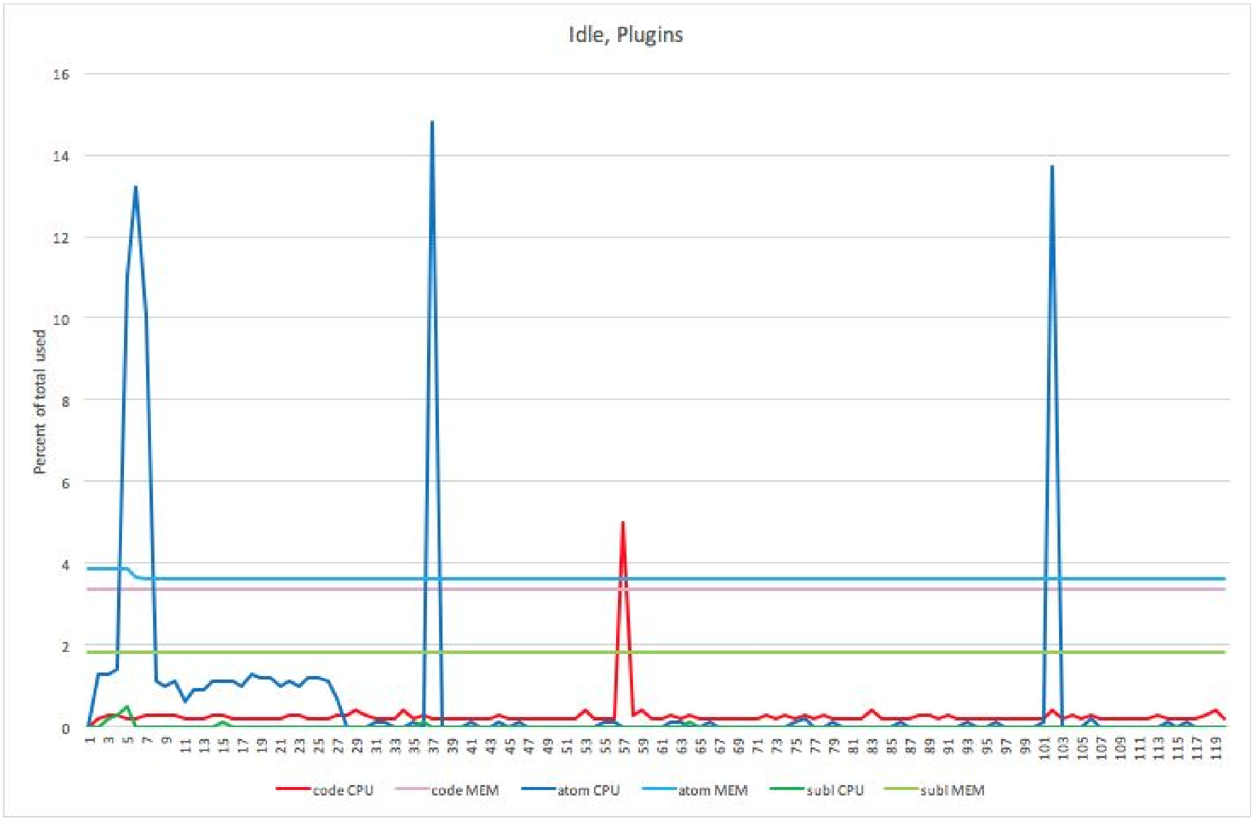

Fig 3. Tests with a blank document open, no user interaction, with plugins.

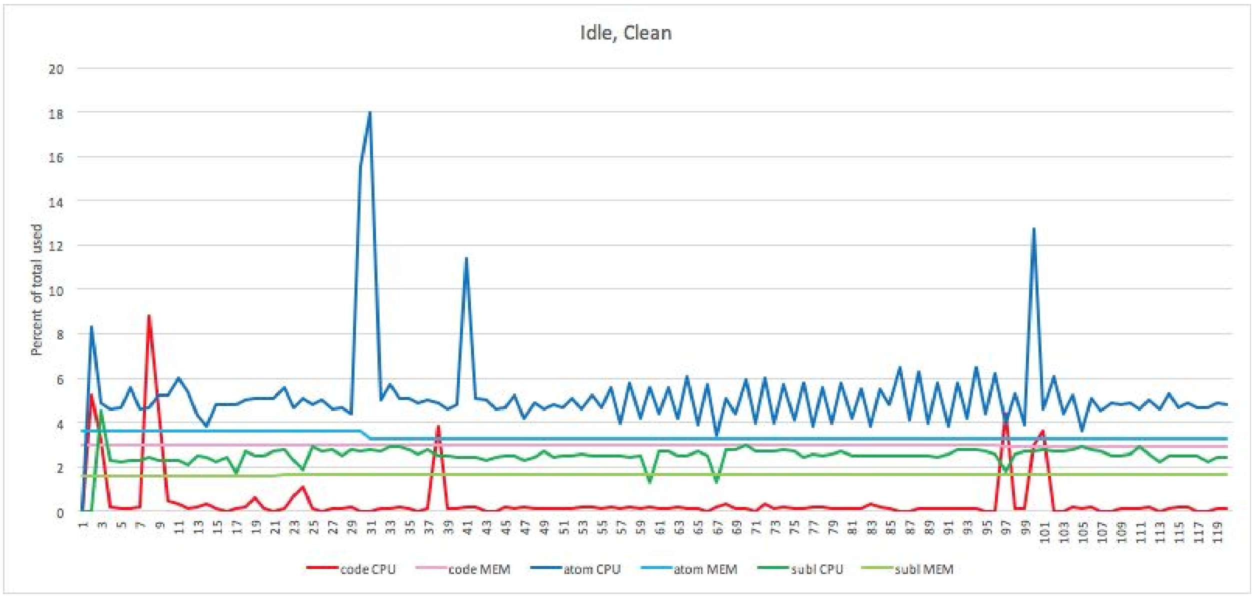

Fig 4. Tests with a blank document open, no user interaction, without plugins.

Aaron Tan
71575415
E/I Yr.3

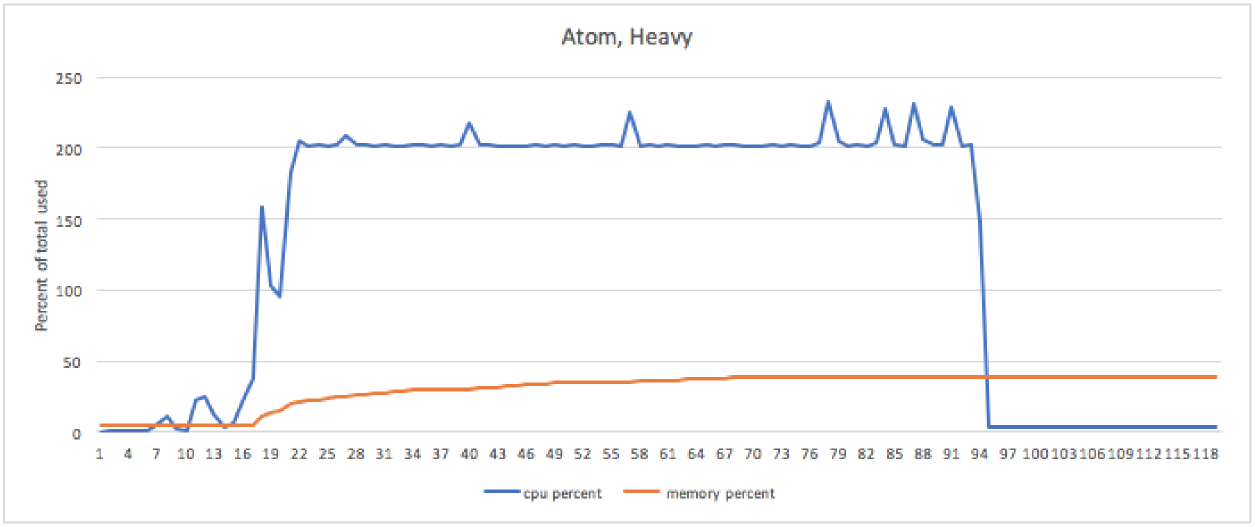

Fig 5. Atom opening a 250MB text file. Crashes at 95 seconds.

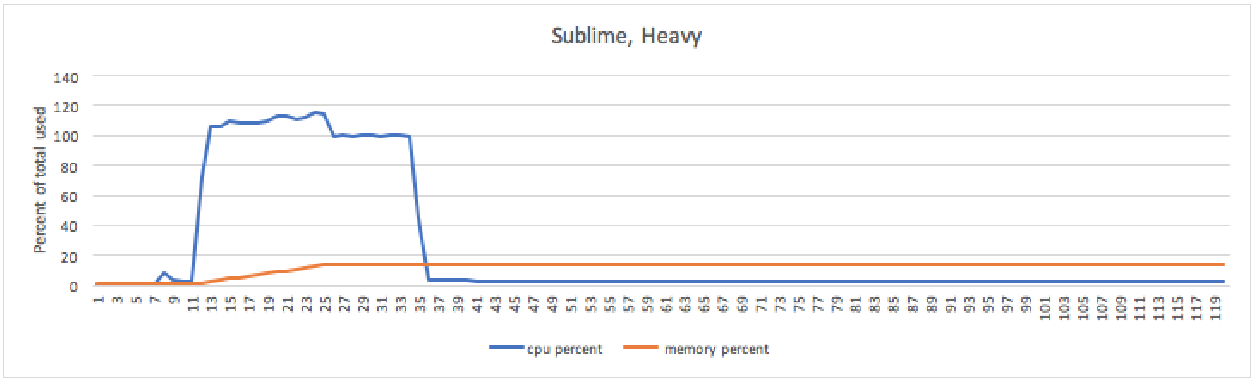

Fig 6. Sublime opening a 250MB text file. Successfully loaded in 23 seconds.

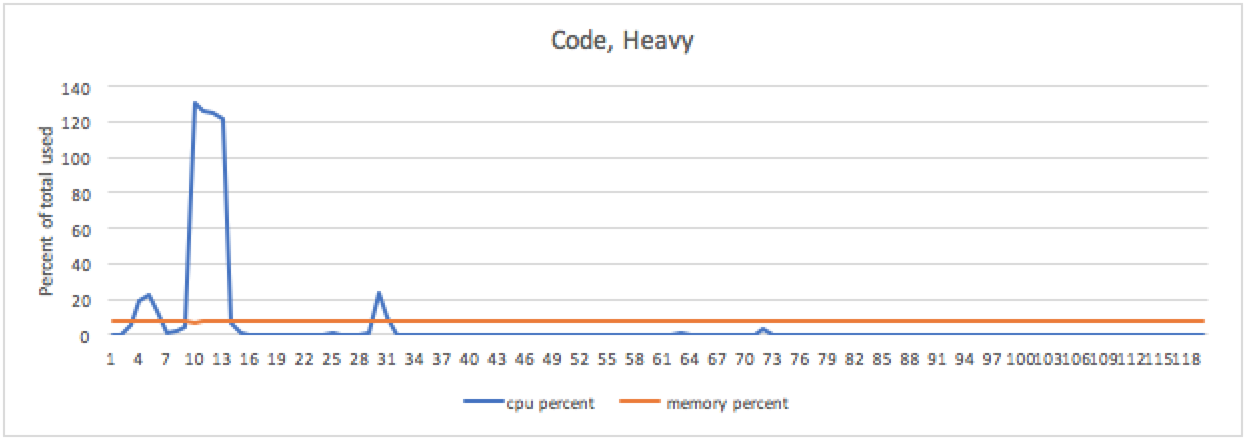

Fig 7. VS Code opening a 250MB text file. Successfully loaded in 4 seconds.

Aaron Tan
71575415
E/I Yr.3

Discussion:

Figures 1 and 2 show similar behavior between the three applications, the one difference being the slightly higher average CPU usage with the inclusion of plugins. This test was run editing a javascript file and CSS file, switching between the two. The same actions were mimicked across all three applications.
Figures 3 and 4 suggests different behaviors when idling as a clean install and idling with plugins installed. Figure 3 shows stable CPU usage with occasional spikes from both Atom and VS Code. Sublime Text remains consistent throughout.
Figures 5, 6, and 7 demonstrate the speed at which each application can handle a large (250MB) text file. This was done to stress test each application against an unreasonable use case. Atom performs poorly, crashing at the 95 second mark. The file was opened at 19 seconds, meaning it ran for 76 seconds. Sublime performed well; it loaded the file in 23 seconds. VS Code performed best, loading the file in only 4 seconds. The memory usage of each application follows a similar order. Atom hovered around 39% memory usage while not being able to open the file. Sublime used 14% and VS Code 8%.

Throughout these tests, Sublime Text appears to be the most memory and CPU efficient of the three. In all tests bar the edge-case test with a heavier file load, Sublime Text managed to run with the least amount of CPU and memory usage.

Issues:

As *psutil* is a cross-compatible with multiple operating systems, revisiting this test and also comparing overall performance across different OSes would be an interesting approach. I had attempted to do so but encountered difficulties dual-booting linux on my machine and had to resort to a single OS approach.

Due to previously mentioned limitations regarding operating system choice, this test was not able to measure the I/O count of each application. While it is possible to do so using *psutil* for other operating systems, macOS does not provide the proper calls to do so. Future attempts to recreate this test should also implement the reading of input/output counts for each application on non-macOS systems. This could give further insight to explain their performance.

Aaron Tan
71575415
E/I Yr.3

Bibliography

“A Hackable Text Editor for the 21st Century.” Atom, atom.io/.

“A Sophisticated Text Editor for Code, Markup and Prose.” Sublime Text, www.sublimetext.com/.

“Psutil.” PyPI, pypi.org/project/psutil/.

“Visual Studio Code - Code Editing. Redefined.” VS Code, Microsoft, 14 Apr. 2016, code.visualstudio.com/.