TypeOS:A Power-Saving-Centric Operating System for Writers

Lorenzo Gomez
OS Theory CS519

In my spare time I write fiction, in fact many people around the world do. In the world of today writers, like myself, go to their OS(Windows, Linux, OSX, MSDOS, etc) and fire up a word processor. Word processors are awesome tools; they are these blank canvases in which one can paint a whole world with new characters, worlds and adventures. *Most* of the time, word processors are very reliable; once words are written they will never be erased, even if they are filesystems today journal most of the input to be able to recover data after unexpected crashes. And in truth, this is awesome for writers! However, once one starts writing in mobile devices like laptops there is a problem that OS designers still grapple with today—power consumption. Most word processors like MS Word, LibreOffice and Pages are not very power consumption friendly. MS Word has an auto-save feature—this is the absolute *worst* for power consumption. It's great for reliability of data, but these saves are just requesting a lot of writes to disk—whether that'd be an SSD or HDD. For the purposes of this paper and because I have a machine with an HDD, when I say "disk" I'll be referring to a mechanical disk. My approach to power consumption should also be applicable to SSDs. This isn't only application-specific, filesystems are also part of the problem as they journal most things—rightfully so to ensure stability—but things like journaling essentially means more writes—which in turn means higher power consumption. The OS as a whole struggles to be power efficient on laptops as it strives for performance efficiency above all else in most cases. In the world of servers, power users and containers this makes a lot of sense. However, when a writer like myself wants to, as Dr. Kannan so eloquently stated, go out in the woods for eight hours without interruption—MS Word, Pages or LibreOffice can only keep one writing for so long. Based on my own personal experience, one is lucky to get 4.5 hours of battery juice while creating new worlds. That's quite some time, the equivalent of a trip to Boston with good traffic. There's nothing wrong with Boston—it is a great city, but too cold! However, what if we want to get to somewhere a lot warmer like Virginia! All jokes aside, it will be more feasible for writers, including myself, to go out into the woods if we could get something closer to 7.5 hours(the travel time to Virginia) from our writing machines—from our blank canvas!

Power consumption is a multi-faceted problem. Disk writes; context switches; network I/O; screens; multiple CPUs; these are all components that factor into the power consumption of an OS. For the purposes of this paper, I will be focusing on filesystems. I will be focusing on filesystems because they surround things like writes from/to disk and memory, which in most general-purpose OSes, are two of the main causes for power consumption. I am not stating that making a more power-efficient filesystem is the silver bullet, but it can help us extend the battery life of our Typers. For the purposes of this paper, I will be referring to computers as Typers to enforce the technical and philosophical idea of a power-efficient laptop for writers. When discussing my filesystem-centric solutions for power consumption, they will be put into the context of TypeOS. Imagine an OS that when you boot it, you stare at your blank canvas. The canvas, cursor and your imagination going wild as you discover new worlds and ideas. Imagine an operating system that is stable enough where you can write a 40-page short story without worrying about power. A simple OS with just the blank canvas word processor and a power-efficient filesystem—that's it! No Office Suite non-sense, no auto-save(though the filesystem will be stable), no weird in-the-middle-of-writing lag because it is auto saving. I spoke to Dr. Kannan about this and he said that the amount of writes can work as a good proxy for power-consumption. I will be focusing on the number of writes that the OS(TypeOS) executes. One way to approach this problem in Linux is to create a virtual hard disk on Linux to isolate a part of the drive in a file and work from there. I could also partition my drive and install a Linux Distro like

Ubuntu Mate to just conduct experiments from there. Tools like powerstat can help me analyze the raw power being consumed by the Typer. A tool like cpufreq allows me to turn off CPU cores and benchmark power using 2 or 4 cores—I have a dual-core machine with 4 threads. There is research out there done by several researchers; Kester Li from Berkely and Yeri Yin Hom from Rutgers University.

In particular, Kester Li from Berkely came up with an idea that caught my eye—an infinite file cache. Now, as the title implies, it is not technically feasible to have *infinite* cache of anything, however, let's not discard this idea so easily. In TypeOS, we have context—we know all there is, in a sense, are just "words". Just a quick glance at caches today, any commodity machine has anywhere between 3MB to 6MB of cache—that is six million words! To put that in perspective, if one were to take the average of all the words in Harry Potter novels that is 154,000 words! That's barely 155KB, which is nothing relative to cache sizes today. A guttural(possible) solution that comes to mind is staying in cache *most* of the time. The idea is to minimize the amount of time we do writes to memory, and go to disk *only* when we absolutely have to. Now, to be fair, general-purpose OSes do some version of this. However because they establish a performance-centric approach , as they should, power consumption suffers—they end up writing things twice because of journaling and in the case of word processors applications it gets worse as developers stuff them with features that don't take into account power consumption. This won't be the case with our new filesystem—our new filesystem for TypeOS will use power-consumption as a unit of measurement for current state. Power consumption will be prioritized over performance, as long as the reduction in performance does not disturb the user's flow while typing into their canvas. My deliverables will incluse some version of this filesystem which would use *infinite* cache to reduce the amount of writes to memory/disk. I will also be be benchmarking my machine with tools like cpufreq to see how far I can push linux while writing some stories using a word processor like LibreOffice Writer.