Introduction: What is MoonCluster?

In short, MoonCluster is a personal computer (meant for mobile computing, i.e. smartphones, small notebooks, etc.) which can run up to a set number of tasks, at any one time, of any single-task program that may run on any personal computer (i.e. videogame, photoviewer, single browser tab, text editor, videoplayer, audioplayer, video editor, Netflix, Spotify, etc.).

Background: Why should you care?

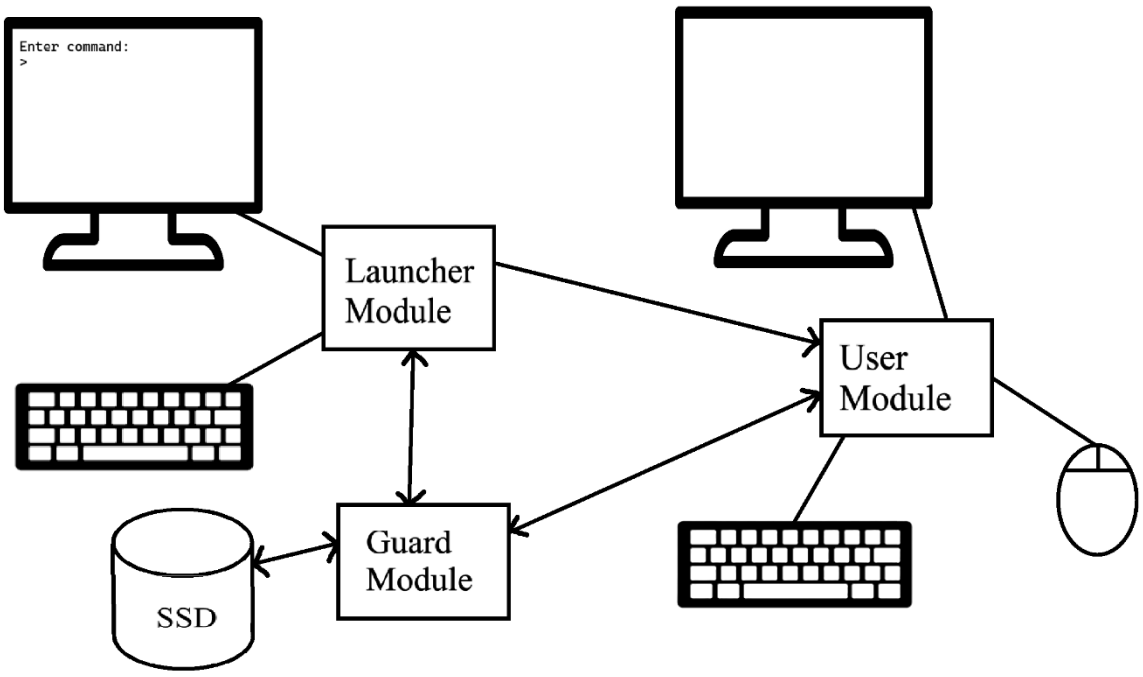
The security of modern PCs is dangerously poor, as evident by the the history of malware exploits, such as the recent Pegasus crisis. Modern PC OSes run on monolithic kernels with millions of lines of evolving, constantly updated code. The security crisis of unending vulnerabilities will persist as long as OSes are this complex. A simpler, more trustworthy alternative must be made available. There have been attempts to do this, such as the Minix 3 and seL4 microkernels, which aim to reduce the amount of trusted code of Oses as much as possible. The proposed research explores another potential alternative which I call MoonCluster.

Details: How does MoonCluster work?

Instead of a single physical computer system running multiple tasks, MoonCluster is comprised of multiple physical computer systems each running a single task.

The user can only run up to as many processes at once as there are "userspace modules," which are simply complete personal computers (each with a network card, monitor, mouse, keyboard).

To explain the architecture, imagine a system with one low-performance computer and two high-performance computers. The low-performance computer will be referred to as "launcher module (LM)," one of the two high-performance computers as "guard module (GM)," and the other as "userspace module (UM)." They are connected like so:



Launcher Module (LM):

LM is a low-performance computer whose only responsibility is running a very rudimentary Unix-like shell. LM has no peripherals/drivers other than USB 2.0, monitor, and keyboard (or touchscreen instead of monitor and keyboard depending on the implementation).

Through LM's interface, the user can launch programs on UM. The user can specify what files/directories the running process can and cannot access, and these permissions will be enforced by GM. For example, the user can run a browser tab such that it cannot access the user's `photos.dir` directory. GM will thereafter reject any file system calls made by UM which access files within photos.dir.

Once LM's terminal user interface (TUI) is defined, a more intuitive and efficient graphical user interface (GUI) can be developed with the exact same functionality. This GUI can be used in the mobile device implementation of MoonCluster.

LM has indirect access to the file system through GM.

User Module (UM):

UM is a high-performance computer on which the current userspace program will be run. When running a userspace program, the userspace program has **full and direct access** to the entire hardware of UM--over the network card, the monitor, the keyboard, the USBs, etc.. The only exception is that the module has no direct access to the SSD (the SSD is only directly accessible by GM). In other words, userspace programs

are themselves operating systems whose only purpose is to execute a single program. For example, a video game would essentially just be an operating system whose only purpose is to run a video game.

When we need to run a new program in UM, we have to ensure that the memory of the previous program is cleaned. So launching a new program is done as follows: completely wipe all volatile memory of UM by shutting off power. Reload UM's bootloader, and boot the new program (the new program is sent to UM from GM). Then execute the new program. Any malware, any undefined activity due to bugs in the previous program is completely gone, the slate perfectly cleaned. The newly loaded program is unaffected by the previous one.

Guard Module (GM):

GM is a high-performance computer, which has no peripherals/drivers other than the connector between GM and UM, the connector between GM and LM, and the SSD.

GM has a high-speed SSD directly connected to it. GM's purpose is to act as a server for the unit's file system. It executes file system calls sent to it from UM, enforcing restrictions set by the user. The file system can be UNIX-like.

In this way, guard **guards** its SSD from the UMs, which are potentially malicious, allowing access to only those directories which the user specifies.

As you can see, the architecture attempts to **physically sandbox** userspace programs to their own machines, as an attempt to reduce the complexity of the system code; the system code for guard is the most complicated part of the system, and all it does is maintain a unix-like file system and execute file system calls from UM. This (and LM, but LM is very simple) becomes the only part of the system that we need to verify.

MoonCluster in action:

Let me walk you through an example. Let's say I want to run a browser tab and restrict it to only the necessary files. I will type the command into launcher which is equivalent to "hey launcher, execute browser.exe and restrict it to assets.dir, uploads.dir, downloads.dir." Maybe something like “./browser.exe -confine assets.dir, uploads.dir, downloads.dir. LM sends the executable name and restrictions to GM, and wipes UM's memory (RAM, registers, cache, everything). GM sends the executable/image of the browser from the SSD to UM, and UM's bootloader loads it into its RAM. Then UM executes browser.exe. Browser's process can directly access all hardware within UM. Browser's process now must communicate with GM to access files, and GM is simple enough to reliably enforce the given restrictions. The browser process is **fully sandboxed**. Thus, in theory, malware will ever escape it.

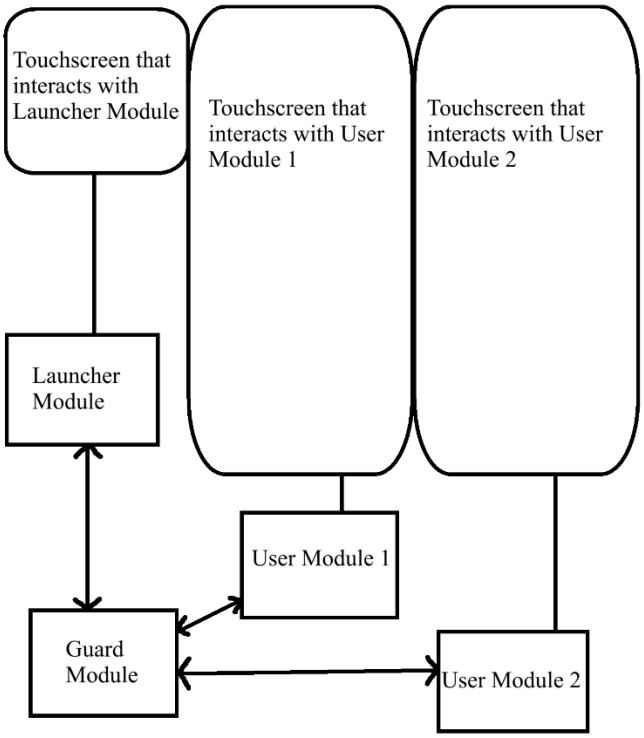
Limitations:

Obviously, this system's egregious limitation is its limited multitasking. But remember, smartphones and tablets are limited multitasking systems! Almost all of them can't run more than two apps at a time!

Thus, a MoonCluster system with two userspace modules and more graphical interface with launcher, is not at all far behind modern mobile devices in terms of functionality.

Here's a potential implementation of MoonCluster as a mobile device:

The number of UMs in the system determine how many tasks the user can run at once. If we extrapolate the design to have two UMs, both connected to GM, we could have a personal computer capable of running two tasks at once. GM can be programmed to handle multiple file requests at the same time. The interface with LM can be made graphical and efficient. LM will be connected to its own touchscreen. the UMs will each have their own touchscreen, microphone, speaker, camera, headphone jack, network card, GPS, etc.. For cellular, the user can connect to an LTE router and carry it with them (I know, it's janky). The internals of loading programs and file accesses will be as was described in previous sections.



What now?

There is a lot to do.

First I will buy two Jetson Orin Nanos and test how fast a file can transfer from one's SSD to the other's RAM through USB 3.2 gen 2. This is a test of the file transfer speed within MoonCluster. A speed >400 MB/s would be more than sufficient.

Second, I will develop a simulation of Launcher.

Third, I will develop a simulation of the entire system on very simple architectures.

And after that, I can start developing the actual system!