Summer Research Proposal: MoonCluster

Nicholas Belotserkovskiy

March 2025

1 Introduction: What is MoonCluster?

MoonCluster is a novel computing framework designed for mobile devices, such as smartphones and compact notebooks, that can efficiently run a limited number of tasks simultaneously. Each task corresponds to a single-purpose application that can be executed on a typical personal computer. Examples include applications such as video games, photo viewers, web browsers, text editors, media players, and streaming services like Netflix or Spotify.

2 Background: Why should you care?

The security of modern PC operating systems remains alarmingly weak, as evidenced by the rise in malware exploits, such as the widely publicized Pegasus attack. Modern operating systems rely on complex monolithic kernels, composed of millions of lines of constantly evolving code. The inherent complexity of these systems continues to contribute to a persistent security crisis, with frequent vulnerabilities emerging in the software ecosystem.

Efforts have been made to simplify OS design and enhance security, such as with the Minix 3 and seL4 microkernels, which aim to minimize the amount of trusted code within operating systems. However, there is still a need for more robust alternatives. This research introduces MoonCluster as a potential solution, offering a simpler and more secure approach to computing.

3 Details: How does MoonCluster work?

Unlike traditional computer systems that rely on a single machine to run multiple tasks, MoonCluster consists of multiple physical devices, each running a single task. The number of tasks a user can run simultaneously is directly limited by the number of available ”userspace modules,” which are essentially independent personal computers, each equipped with a network card, monitor, mouse, and keyboard.

To illustrate this architecture, consider a system comprised of one low-performance computer and two high-performance computers. The low-performance

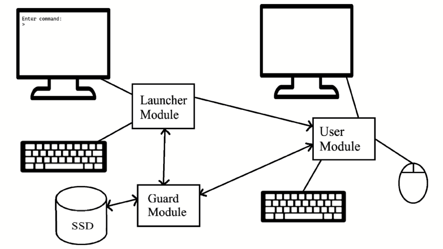

Figure 1: MoonCluster setup

unit is designated the "Launcher Module" (LM), one of the high-performance computers is the "Guard Module" (GM), and the other is the "Userspace Module" (UM). These modules are interconnected as illustrated in figure 1.

4 Launcher Module (LM):

The Launcher Module (LM) is a low-performance computer that serves as the primary interface for the user, running a simple Unix-like shell. It lacks advanced peripherals and drivers, except for basic components such as USB ports, a monitor, and a keyboard (or potentially a touchscreen, depending on the specific implementation).

The LM allows the user to initiate programs on the Userspace Module (UM). Additionally, it enables the user to specify access restrictions for the running processes, such as limiting the files or directories that a program can access. These restrictions are enforced by the Guard Module (GM). For example, if the user runs a web browser, the browser may be restricted from accessing a specific directory, such as the user's photo library. GM ensures that no unauthorized file system calls are made by the UM.

While LM's terminal interface is simple, it can be further developed into a more user-friendly graphical interface (GUI) for mobile device implementations. LM also interacts with the file system indirectly through GM.

5 User Module (UM):

The Userspace Module (UM) is a high-performance computer designed to run a single user application. When executing a program, the UM has full access to its own hardware, including the network card, monitor, keyboard, and USB ports. However, the UM does not have direct access to its local storage (e.g., the SSD), which is exclusively controlled by GM.

In essence, each userspace program behaves as a minimal operating system designed solely to execute a single application. For instance, a video game running on the UM is an operating system that manages only the resources necessary for the game.

Before launching a new program, the UM's memory is completely wiped by powering down the system, ensuring a fresh environment for the new application. The program is then loaded from the Guard Module (GM), effectively isolating any remnants of previous tasks, thus preventing potential issues such as malware persistence.

6 Guard Module (GM):

The Guard Module (GM) is another high-performance computer responsible for managing the file system and enforcing access control between the UM and its storage. The GM is not directly connected to any user peripherals; it primarily handles network communications between LM and UM, as well as managing file access via a high-speed SSD.

The GM acts as a server, executing file system calls sent by the UM, ensuring that user-defined access restrictions are adhered to. This module ensures that the UM only has access to the files and directories that the user has explicitly allowed, preventing unauthorized data access or malicious activity.

The architecture of MoonCluster simplifies the code complexity by physically isolating userspace applications to their own hardware, while the GM's primary responsibility is the enforcement of a secure and simplified file system.

7 MoonCluster in action:

To better illustrate the operation of MoonCluster, consider the scenario where a user wishes to run a web browser while restricting access to certain files. The user would input a command into the LM, specifying the executable and the directories it is permitted to access, for example:

```
./browser.exe -confine assets.dir(Read), uploads.dir(Write),
downloads.dir(Read and Write)
```

This command might run the browser so that it can only read from assets.dir, write to uploads.dir and read and write to downloads.dir. It will have no access to any other directories.

The LM sends this command and access restrictions to the GM. It then wipes the memory of the UM to ensure no residual data from previous tasks remain. GM transmits the browser executable from its SSD to the UM, where the program is loaded into RAM and executed.

Once the browser is running, it has full access to UM's hardware, but any requests to access restricted files will be blocked by the GM. This sandboxing ensures that any potential malware contained within the browser process cannot escape or access unauthorized data.

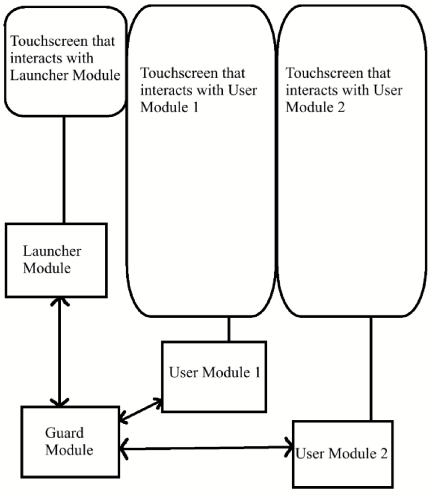
Touchscreen that interacts with Launcher Module
Touchscreen that interacts with User Module 1
Touchscreen that interacts with User Module 2
Launcher Module
User Module 1
Guard Module
User Module 2

Figure 2: Mobile Device setup

8 Limitations:

While MoonCluster offers significant security benefits, it does come with some limitations, notably in terms of multitasking. Since each userspace module can only run a single task, the system's multitasking capabilities are restricted. However, this is not a major drawback when compared to modern mobile devices, which often have limited multitasking capabilities. For instance, many smartphones can only run two applications simultaneously, which aligns well with the capabilities of a MoonCluster system with two userspace modules.

A potential implementation of MoonCluster as a mobile device could involve two userspace modules connected to the GM, allowing the device to run two tasks at once. The system's interface could be graphical, with LM providing a touchscreen for user interaction. Each UM would have its own suite of peripherals, including touchscreens, microphones, speakers, cameras, and network cards, enabling the device to function as a fully-fledged mobile computer. A cellular connection could be established via an external LTE router. Figure 2 is a diagram of this design.

9 What now?

Several steps are required to bring the MoonCluster concept to life:

1. **Hardware Testing**: First, I will acquire two Jetson Orin Nanos to test the file transfer speed between the SSD and RAM via USB 3.2 Gen 2. Achieving a transfer speed greater than 400 MB/s would be sufficient for the system's needs.

2. **Software Simulation**: Next, I will develop a simulation of the Launcher Module to test its functionality and interaction with other components of the system.

3. **Full System Simulation**: Following the Launcher simulation, I will create a comprehensive simulation of the entire MoonCluster architecture using simple hardware platforms to test the interaction between the LM, GM, and UM.

4. **Documentation and Research Paper**: A main goal of this endeavor is to thoroughly document the system, publish a research paper that collates all relevant results, and effectively publicize the idea.

5. **Physical System Development**: After successful simulations, I will begin designing and constructing the actual hardware implementation of MoonCluster.

Through these steps, I aim to demonstrate the feasibility and security advantages of MoonCluster as an alternative approach to modern computing.