# An effective way to optimize performance and battery life of smart devices using multi-kernel approach

Sadid Rafsun Tulon
School of Computer Science
University of Ottawa
Ottawa, Canada
*stulo080@uottawa.ca*

December 19, 2021

**Abstract**

Enhancing the performance and battery life of smart devices is modern technology's most vital aspect. As reducing the transistor size becomes increasingly complex and battery technology has plateaued, other methods to increase performance and decrease power consumption are being explored relentlessly. While there are many techniques available, this paper focuses on how to improve the performance and battery life of modern battery-powered smart devices using a new multi-kernel system technique. The main goal of this paper is to examine the potential of multi-kernel systems and compare them against Linux systems.

**Keywords:** multicore processors, multi-kernel, unikernel, light weight-kernel

## 1 Introduction

Smart devices have become an integral part of our life. Many of the smart devices are battery-powered. The current battery technology is the most significant handicap of portable smart devices right now. As CPUs and systems on a chip(SoC) improve their efficiency, things are getting better. However, modern software also demands more processing power, which faster drains the battery. The lightweight kernels can extend battery life much further, but their software compatibility is far from ideal. Now that battery technology has reached a plateau and transistors in the semiconductors are also close to reaching their size limit, we are in need of an alternate way of achieving better battery life without compromising performance and compatibility. A new approach using multiple operating system (OS) kernels can help mitigate this problem. Almost all modern chips are multi-core. We can use these multi-core chips to run multiple OS kernels at the same time. Ideally, one of the kernels is a lightweight kernel for better performance and battery life and a monolithic kernel for better compatibility. The system can switch between the kernels, run them simultaneously, or stop them from running depending on the need. This way, we will be able to get better battery life while still retaining the software compatibility. This research is an effort to optimize performance, compatibility, and battery life by utilizing a multi-kernel approach while using existing battery technology.

In this project, the idea is to research if we fit two operating systems for both cores of multi-core processors individually and how they perform.

| Design goal | LWK | FWK |
|---|---|---|
| Target | massively parallel systems | laptops, desktops, servers |
| Support | scalable applications | everything under the sun |
| Dev. environment for | parallel applications | business, games, commerce, etc. |
| Emphasis | effciency | functionality |
| Resources | maximize use | fair sharing, QoS |
| Time to completion | minimal | when needed |

Table 1: Design goals of LWKs and FWKs [17]

# 2 Literature Review

## 2.1 Different types of kernels

The current hardware technology is changing rapidly. But they are still limited by physical limitations. Performance mainly depends on data transfer speed and process time. But these are limited by electron flow rate[1]. As transistors are becoming smaller, the electron has to travel less. But shrinking transistor sizes are becoming more challenging. The size of a silicon atom is .02 nanometre and we are reaching this limit [8]. So we need a new OS structure that would be more resource-efficient without compromising compatibility. Multi-core processors are used in the majority of current systems. We can leverage this to our advantage.

Light Weight Kernels (LWKs) are not a new concept. They have been here for at least 30 years [17]. However, they are only gaining popularity in these past couple of years because of High-Performance Computing (HPC). HPC necessitates tremendous scalability, which the Full-Weight Kernel (FWKs) take a long time to adjust to. LWKs work exceptionally well with highly scalable systems as they have a simple codebase.Table 1 describes the Design goals of LWKs and FWKs

There is also a new type of single address space operating system called unikernels. They are minimalistic custom single-purpose operating systems made from library os that only contains the functionality required for specific operation [11] [9]. As they are very specialized, it is easy to optimize them for hardware. Unikernel uses library operating system. Like LWKs, unikernels are also very lightweight, sometimes unikernels are lighter than LWKs [9]. Moreover, like LWKs they are also extremely resource-efficient [4].
Figure 1 shows the application stack on monolithic OS vs Unikernel.

In this paper, we will be treating both unikernal and LWK as the sub kernel as they both have better performance and efficiency than FWKs and have limited compatibility with the standard APIs [6], [5]. Additionally, LWKs and unikernels lack driver support. As they have a simple codebase mainly focusing on performance, they are missing many advanced features that FWKs have [6] [11].

## 2.2 Multi-Kernel systems

Multi-kernel operating systems have been presented as a way to improve system performance and efficiency without sacrificing software and hardware compatibility. Multikernel OS runs

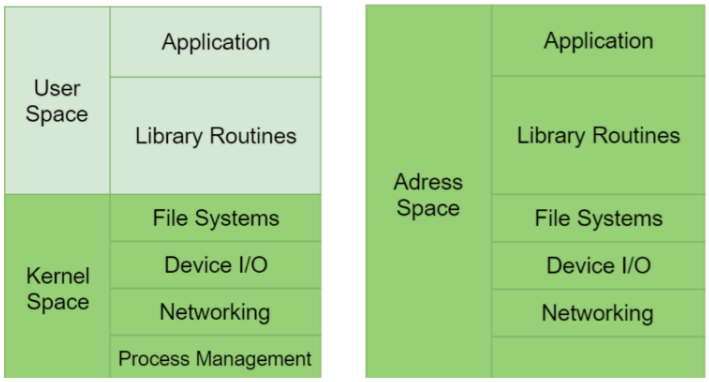


Figure 1: Application stack on monolithic OS vs Unikernel [12]

both FWK and sub kernel side by side divides their tasks [5]. The sub kernel is usually responsible for the high-performance tasks, and FWK is there for ensuring compatibility and driver support [6]. In high-performance supercomputers, this system performs admirably. The kernels communicate with each other using message-passing or shared-memory [3]. There are also different types of implementation 1) booting both kernels simultaneously when the system starts and 2) booting the LWK after Linux [5].

Figure 2 shows the system architecture of a multikernel system.

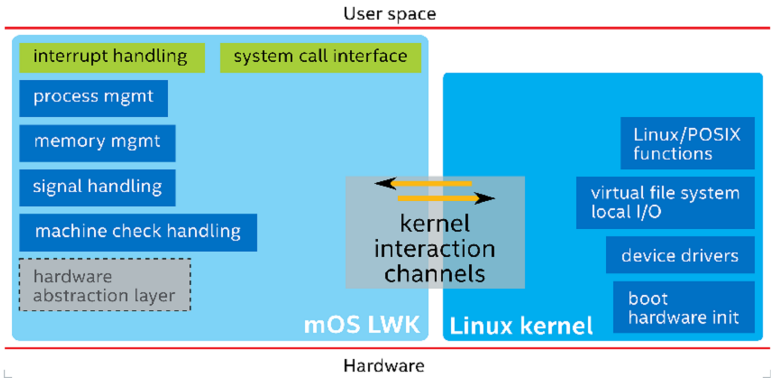


Figure 2: System Architectural Overview of Multi-kernel [17]

Three general goals of multikernel OS are 1) Achieving scalability of LWK, 2) Maintaining Linux compatibility, and 3) Being able to support Linux development so that the system does not fall behind. Achieving all three goals are extremely hard as they conflict with each other [5]. The multikernel method tries to solve this conflict. As Linux is a mature operating system, now the goal is to build a LWK that plays well with the Linux kernel and does not need heavy modification of the Linux kernel to accommodate it. As mentioned previously, there are two types of implementation of multikernel. If both kernels boot simultaneously, Linux has to be modified heavily making maintaining Linux kernel such as updating or fixing laborious. However, this performs better as the subkernel can

access more resources without depending on Linux. On the other hand, the second approach makes maintaining Linux easier but the subkernel has to request resource allocation from Linux which reduces performance[5].

This paper will explore if we can use this multi-kernel method on battery-powered smart devices. We will compare performance and power efficiency between a single Linux kernel system and a multi-kernel system. As batteries have a very small capacity, ensuring longer battery life while maintaining consistent performance is crucial for this project. Thus performance and power efficiency will be the main focuses of this research project.

## 3 Problem Statement

Moore's Law [14] is reaching a physical limitation. The industry still trying to come up with clever methods to shrink it even further but those methods are really expensive and complex. Battery technology has also slowed down. But demand for battery-powered devices is increasing exponentially. So enhancing performance and battery life take top priority. As we are close to the technical limitation of hardware, we need an approach to improve performance and battery life that is not constrained by hardware.

## 4 Proposed Solution

Most of our modern chips are using multiple cores. Multi-kernel system is a new promising method that can leverage these multi-core chips. As mentioned before multikernel system is a method of running two kernels simultaneously on a device. One main advantage of this system is, it has the potential to improve existing devices using software updates, not only the new devices. We will evaluate if multikernel systems have the potential to solve performance and battery life problems.

## 5 Experiments and Evaluation

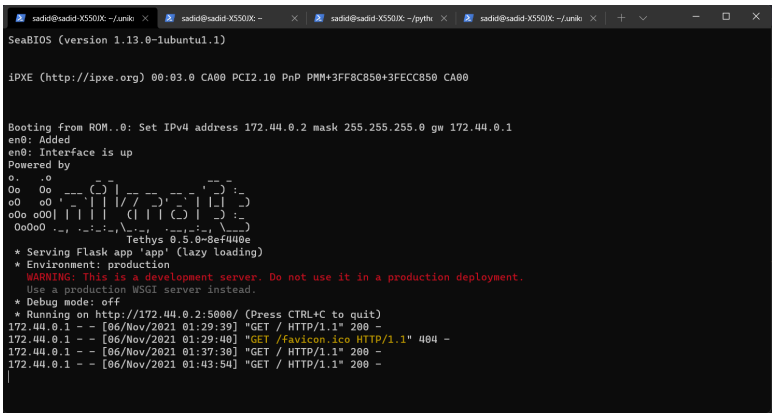


Figure 3: Bootscreen of Unikraft running Flask server

The following experiments are done with a machine using Intel core I5 4500H 4 core CPU clocked at 2.50 GHz, 8 gigabytes of ram. Primary operating systems were Ubuntu and

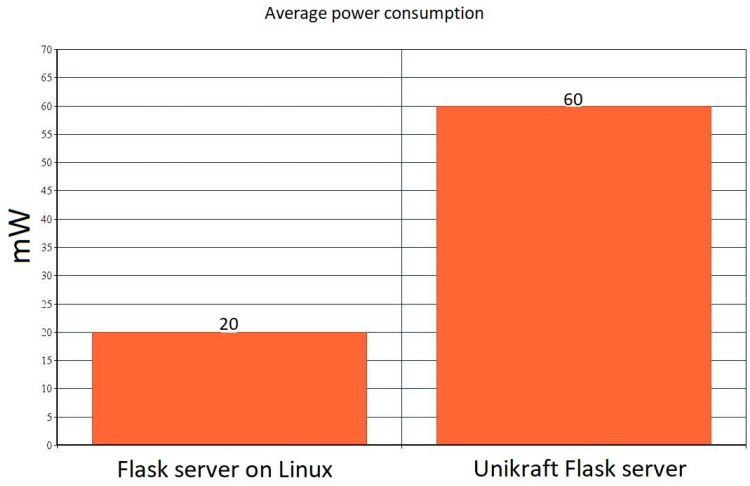


Figure 4: Unikraft Flask server vs Linux native Flask server (excluding OS) power consumption

CentOs depending on the situation. The system was freshly installed with Linux operating systems before each attempt. We tried to compile and test various open-source LWKs and Unikernels such as Azalea, HermitCore, mckernel, MirageOs, Unikraft. Table 2 describes the summery of each experiment.

| Kernel | Main Linux OS | Architecture | Status | Result |
|---|---|---|---|---|
| Azalea | CentOs | x86 | Failed to install | Fail |
| Hermitcore | Ubuntu | x86 | Failed to compile on machine | Fail |
| mckernel | CentOs | x86 | Failed to compile on machine | Fail |
| mckernel | CentOs | arm64 | Failed to compile on machine | Fail |
| MirageOs | Ubuntu | x86 | Successfully compiled on machine but failed to boot anything other than helloworld | Partial Success |
| MirageOs | RaspbianOS | arm64 | Failed to compile on machine | Fail |
| Unikraft | Ubuntu | x86 | Successfully compiled and ran flask server | Success |
| Unikraft | RaspbianOS | arm64 | Successfully compiled on machine but failed to boot anything other than helloworld | Partial Success |

Table 2: Experiment with various kernels

From Table 2, we can see that only Unikraft fully worked on our system. The rest of the kernels refused to compile or run.

The biggest challenge we faced while trying to compile these kernels was a lack of support. Most of the guides the very difficult to follow. Moreover, if any error occurred, it was extremely difficult to debug because of the complex code struct and lack of supporting documents. Furthermore, as our system was not officially supported, our failure rate was even high. We followed [18], [7],[16], [13]and [19] to install Azalea, HermitCore, mckernel,

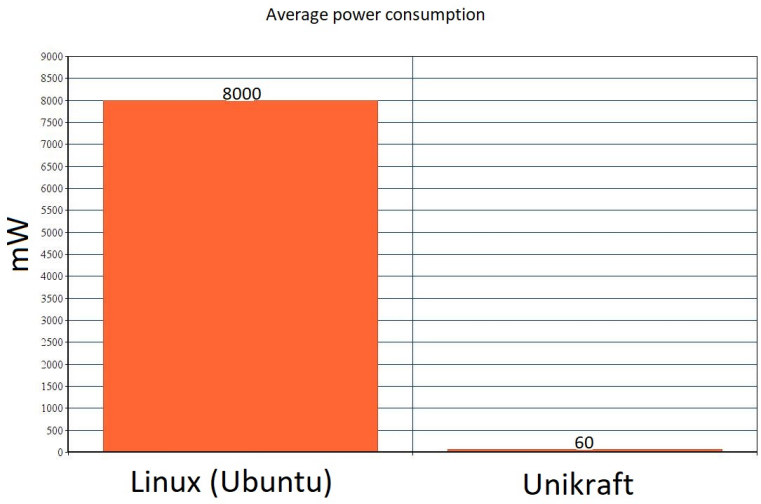


Figure 5: Unikraft vs Linux power consumption

MirageOs, Unikraft respectively. Azalea, HermitCore, and mckernel failed to compile on our system. MirageOs and Unikraft compiled successfully but we could only run helloworld example on MirageOs.

We followed "Python Flask on Unikraft" to install the Unikraft Flask server [2]. As unikraft specifically supports python 3.7.4, we had to manually install that version using pyenv [15]. We could then boot the Unikraft kernel using qemu kvm. We could not find any way to boot Unikraft natively on our system.

We also tried to compile and test the kernels mentioned above on the Raspberry pi 4 4 gigabyte model. Only helloworld module of Unikraft ran properly.

We suspect such a situation occurred because these kernels are built for server CPUs such as Intel Xeon and Fujitsu A64FX [5] [3] [17]. Almost all of the LWKs and unikernels and specialized for servers, so it is understandable that consumer-grade CPUs are incompatible with them. So in the end, we were not able to fully implement the multikernel system. While we were able to boot unikraft it was not a true multikernel system as unikraft was running on kvm.

As only Unikrft ran properly on our main system, we will mainly focus on Unikraft. We have been able to run the flask server on Unikraft. While testing, the boot time of unikraft was found 3 seconds on average. Unikraft flask server consumes 60 milliwatts on average. On the other hand native Linux flask server process consumes 20 milliwatts on average. But if we also include Linux power consumption, the total power consumption will be 8000 milliwatts on average on a freshly booted system. For measuring power consumption, powerstat [10] and PowerTOP [20] was used.

Figure 3 is the boot screen of Unikraft running Flask server.

Figure 4 is a comparison of power consumption between Unikraft Flask server and Linux native Flask server without including whole Linux power consumption.

Figure 5 is a comparison of power consumption between Unikraft and Linux.

Unfortunately, due to a lack of proper hardware, we were unable to verify the full potential of the multikernel system. But this system still shows potential. Our experiment

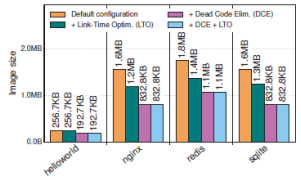


**Figure** Image sizes of Unikraft applications with and without LTO and DCE.

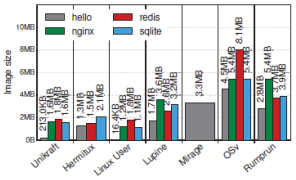


**Figure** Image sizes for Unikraft and other OSes, stripped, w/o LTO and DCE.

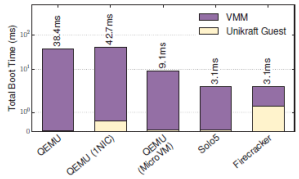


**Figure** Boot time for Unikraft images with different virtual machine monitors.

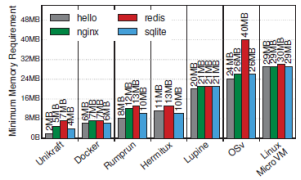


**Figure** Minimum memory needed to run different applications using different OSes, including Unikraft.

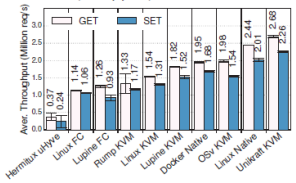


**Figure** Redis perf (30 conns, 100k reqs, pipelining 16) with QEMU/KVM and Firecracker (FC).

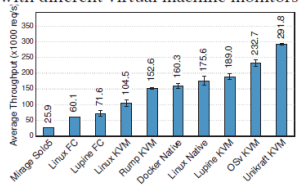


**Figure** NGINX (and Mirage HTTP-reply) performance with `wrk` (1 minute, 14 threads, 30 conns, static 612B page).

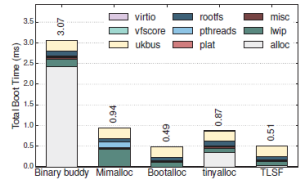


**Figure** Unikraft Boot time for Nginx with different allocators.

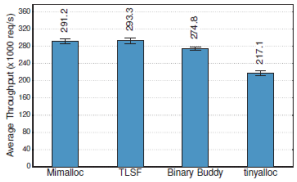


**Figure** nginx throughput with different allocators.

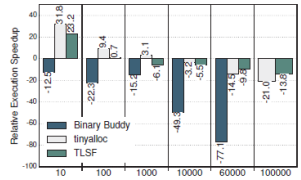


**Figure** Execution speedup in SQLite Unikraft, relative to mimalloc [42].

Figure 6: Performance overview of Unikraft and comparison against competitors[11]

shows that this system uses less power if implemented properly. We have also seen from earlier work in [11] that this system also performs better in most cases.

# 6 Conclusions

## 6.1 Conclusions

In conclusion, multikernel system is a promising concept but needs more support. Previous work in [5] shows that 9% to 280% performance improvement was achieved using multikernel. In our experiment, we can see that Unikernel used 1/133 of power than Linux even while running on kvm. Though it was not running a true multikernel system, we can theoretically reduce power consumption drastically if we can implement a multikernel system properly. As this concept is relatively new, a lot of work needs to be done before this concept can mature. For now, most of the multikernel projects are open-source test projects. These are far from being stable. Currently, some kernels such as Unikraft, Include os, mirage os are receiving decent support but there is still room for improvement. As the support for these kernels gets better, they should also be more available to a wider range of systems.

## 6.2 Summary of Contributions

In this paper, we tried to explore if the multikernel system is a viable alternative method to improve performance and battery life. We tried to test several mainstream sub kernels on

consumer-level hardware such as generic laptops and raspberry pi. We also measured boot time and power consumption on Unikraft. But we could not boot anything meaningful on raspberry pi. As our main target system for the project is battery-powered devices, arm-powered raspberry pi should have been the main target system. Unfortunately, due to lack of support, raspberry pi was not able to boot anything that can be measured properly.

### 6.3 Future Work

Future work includes making multikernel systems more available and making them stable. To do that we need to make existing kernels work with more devices or create a new configurable kernel from the ground up with a large array of hardware support in mind. Previous work in [3] talks about hardware neutral multikernel, but we have not found anything that could be considered hardware neutral. Most kernels are intended for HPCs. The majority of the battery-powered devices have low-powered arm CPUs in them. So for this project goal to succeed, proper arm CPU support is necessary. As mentioned earlier, we also faced difficulty compiling some kernels on our system. In addition, proper support documents would be beneficial for trying, testing, and contributing to these projects. We faced a lot of issues while trying various kernels and the lack of a proper guide made it more challenging. Decent community and industry support would mitigate these issues.

## References


[1] Computers are becoming faster and faster, but their speed is still limited by the physical restrictions of an electron moving through matter. what technologies are emerging to break through this speed barrier?, Oct 1999 available at https://www.scientificamerican.com/article/computers-are-becoming-fa/.

[2] Fredrik Bakken. Python flask on unikraft, Sep 2021 available at https://medium.com/@fredrik.bakken/python-flask-on-unikraft-ced8ac327c07.

[3] A. Baumann, P. Barham, PE. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. The multikernel: a new os architecture for scalable multicore systems. In *SOSP '09: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 29–44. Association for Computing Machinery, 2009.

[4] A. Bratterud, A. A. Walla, H. Haugerud, P. E. Engelstad, and K. Begnum. Includeos: A minimal, resource efficient unikernel for cloud services. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 250–257, 2015.

[5] B. Gerofi, R. Riesen, M. Takagi, T. Boku, K. Nakajima, Y. Ishikawa, and R. W. Wisniewski. Performance and scalability of lightweight multi-kernel based operating systems. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 116–125. IEEE Computer Society, 2018.

[6] B. Gerofi, A. Santogidis, D. Martinet, and Y. Ishikawa. Picodriver: fast-path device drivers for multi-kernel operating systems. In *27th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '18)*, pages 2–13. Association for Computing Machinery, 2018.

[7] HermitCore. libhermit. *GitHub repository*, 2017, available at https://github.com/hermitcore/libhermit.

[8] M. Hopkinson. With silicon pushed to its limits, what will power the next electronics revolution?, 2015 available at https://phys.org/news/2015-08-silicon-limits-power-electronics-revolution.html.

[9] S. H. Jeon, S. J. Cha, Ramneek, Y. J. Jeong, J. M. Kim, and S. Jung. Azalea-unikernel: Unikernel into multi-kernel operating system for manycore systems. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1096–1099, 2018.

[10] C. I. King. powerstat. *GitHub repository*, 2013, available at https://github.com/ColinIanKing/powerstat.

[11] S. Kuenzer, V. A. Badoiu, H. Lefeuvre, S. Santhanam, A. Jung, G. Gain, C. Soldani, C. Lupu, S. Teodorescu, C. Raducanu, C. Banu, L. Mathy, R. Deaconescu, C. Raiciu, and F. Huici. Unikraft: fast, specialized unikernels the easy way. *Proceedings of the Sixteenth European Conference on Computer Systems*, 2021.

[12] G. Longree. unikernels. *GitHub repository*, 2018, available at https://github.com/cetic/unikernels.

[13] MirageOS. mirage. *GitHub repository*, 2013, available at https://github.com/mirage/mirage.

[14] G. E. Moore. Progress in digital integrated electronics [technical literaiture, copyright 1975 ieee. reprinted, with permission. technical digest. international electron devices meeting, ieee, 1975, pp. 11-13.]. *IEEE Solid-State Circuits Society Newsletter*, 11(3):36–37, 2006.

[15] pyenv. pyenv. *GitHub repository*, 2013, available at https://github.com/pyenv/pyenv.

[16] RIKEN-SysSoft. mckernel. *GitHub repository*, 2018, available at https://github.com/RIKEN-SysSoft/mckernel.

[17] R.Riesen, A. Barney Maccabe, B. Gerofi, DN. Lombard, JJ. Lange, K. Pedretti, K. Ferreira, M. Lang, P. Keppel, RW. Wisniewski, R. Brightwell, T. Inglett, Y. Park, and Yutaka Ishikawa. What is a lightweight kernel? In *5th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS '15)*, pages 1–9. Association for Computing Machinery, 2015.

[18] Operating systems lab. Azalea. *GitHub repository*, 2018, available at https://github.com/oslab-swrc/Azalea.

[19] Unikraft.org. Unikraft. 2017, available at https://github.com/unikraft/unikraft.

[20] A. V. D. Ven. powertop. *GitHub repository*, 2010, available at https://github.com/fenrus75/powertop.