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 18, 2021

Abstract

Improving the performance and battery life of smart devices is modern technology's most vital goal. As lowing 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 addresses 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.

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 [8]

2 Literature Review

2.1 Different types of kernels

The current hardware technology is changing rapidly. The software is struggling to keep up with this rapid change. Thus it has become very challenging to optimize software for a specific hardware[1]. So we need a new OS structure that can be as scalable as needed. This is where LWKs and unikernels come in handy. 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 [8]. 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 single-purpose operating systems made from library os [unikernel]. As they are very specialized, it is easy to optimize them for hardware. Like LWKs, unikernels are also very lightweight, sometimes unikernels are lighter than LWKs [4]..

Figure 1 shows 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 [3], [2]. Another major issue of LWKs and unikernels are driver support. As they have a simple codebase mainly focusing on performance, they are missing many advanced features that FWKs have [3] [5].

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 both FWK and sub kernel side by side divides their tasks [2]. The sub kernel is usually responsible for the high-performance tasks, and FWK is there for ensuring compatibility, driver support and serves as a backup system [3]. In high-performance supercomputers, this system performs admirably.

Figure 2 shows the system architecture of a multikernel system.

This paper will explore if we can use this multi-kernel method on battery-powered

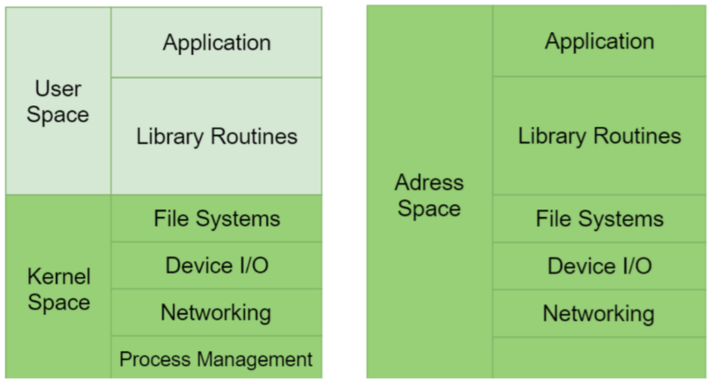


Figure 1: Application stack on monolithic OS vs Unikernel [6]

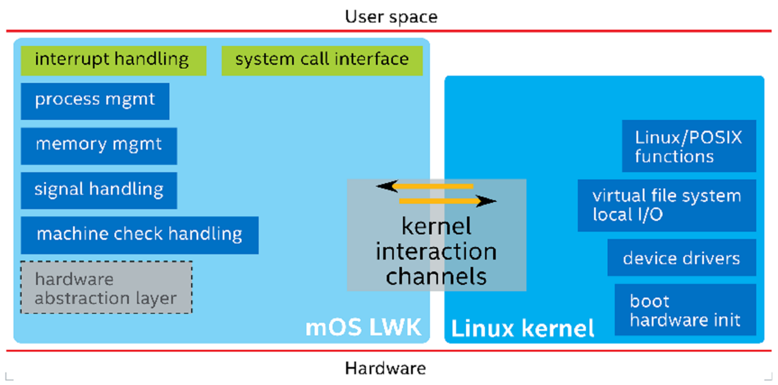


Figure 2: System Architectural Overview of Multi-kernel [8]

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.

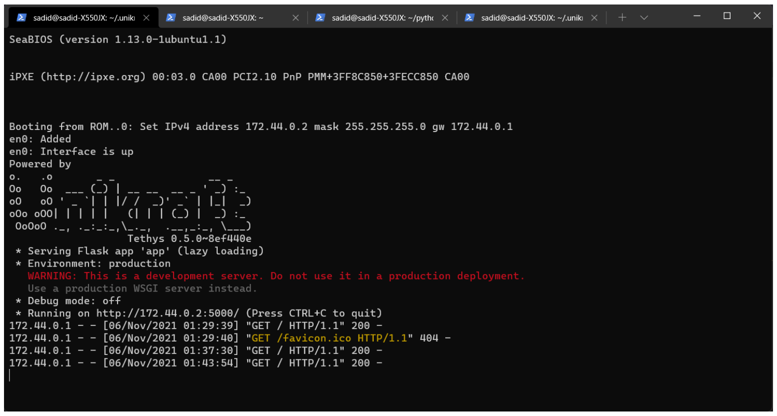
3 Problem Statement

Moore's Law [7] is reaching a physical limitation. Industry still trying to come up with clever methods to 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 are increasing exponentially. So enhancing performance and battery life takes top priority. As we are close to technical limitation of hardware, we need a 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 kernel simultaneously on a device.One main advantage of this system is, it has a potential to improve existing devices using software update, not only the new devices. We will evaluate if multikernel systems has potential to solve performance and battery life problem.

5 Experiments and Evaluation


sadid@sadid-X550JX: ~/.uniki
sadid@sadid-X550JX: ~
sadid@sadid-X550JX: ~/pytho
sadid@sadid-X550JX: ~/.uniki

```
SeaBIOS (version 1.13.0-1ubuntu1.1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+3FF8C850+3FECC850 CA00



Booting from ROM..0: Set IPv4 address 172.44.0.2 mask 255.255.255.0 gw 172.44.0.1
en0: Added
en0: Interface is up
Powered by
o.   .o       _ _            __ _
Oo   Oo  ___ (_) | __ __  __ _ ' _) :_
oO   oO ' _ `| | |/ /  _)' _` | |_|  _)
oOo oOO| | | | |   (| | | (_) |  _) :_
 OoOoO ._, ._:_:_,\_._,  .__,_:_, \___)
                   Tethys 0.5.0~8ef440e
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://172.44.0.2:5000/ (Press CTRL+C to quit)
172.44.0.1 - - [06/Nov/2021 01:29:39] "GET / HTTP/1.1" 200 -
172.44.0.1 - - [06/Nov/2021 01:29:40] "GET /favicon.ico HTTP/1.1" 404 -
172.44.0.1 - - [06/Nov/2021 01:37:30] "GET / HTTP/1.1" 200 -
172.44.0.1 - - [06/Nov/2021 01:43:54] "GET / HTTP/1.1" 200 -
```

Figure 3: Bootscreen of Unikraft running Flask server

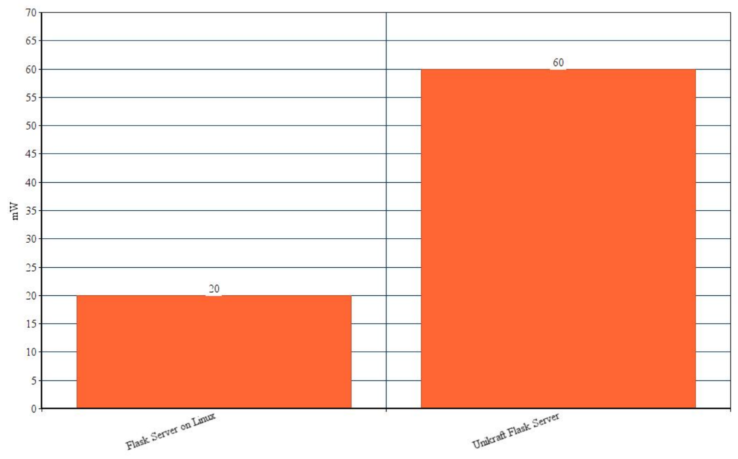


Figure 4: Unikraft Flask server vs Linux native Flask server (without OS) power consumption

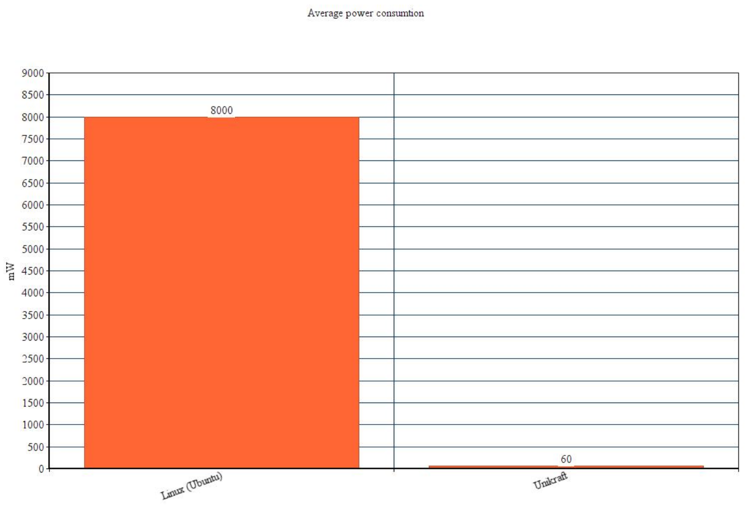


Figure 5: Unikraft Flask server vs Linux native Flask server (with OS) power consumption

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 CentOs depending on the situation. 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.

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, AMD Epyc, and Fujitsu A64FX [..................]. Almost all of the LWKs and unikernels and specialized for servers, so it is understandable that consumer-grade

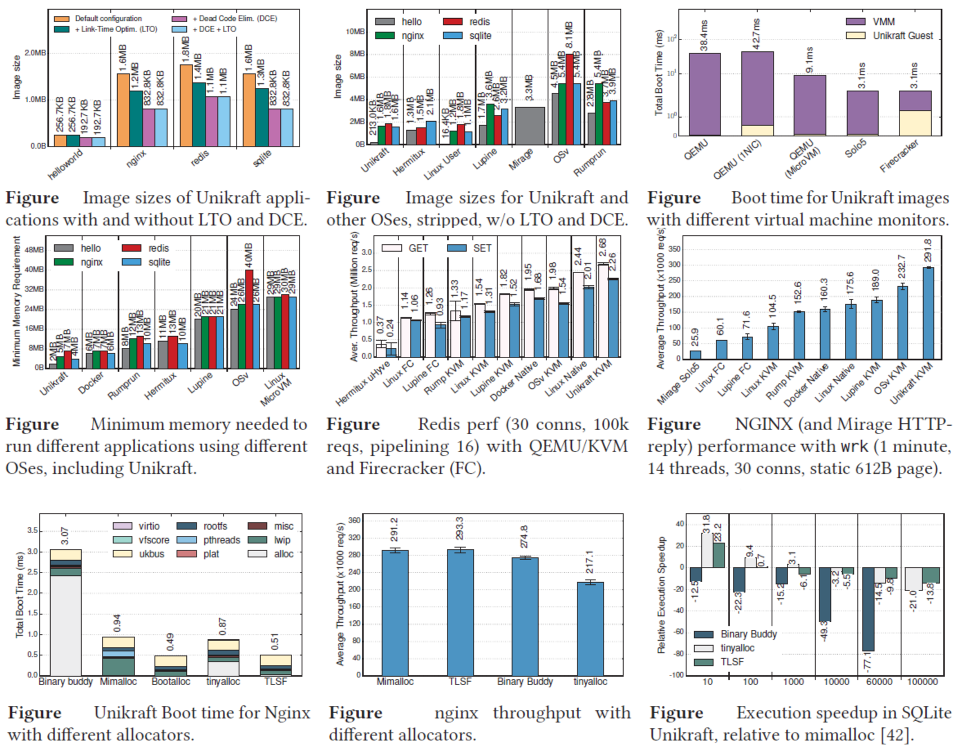


Figure Image sizes of Unikraft applications with and without LTO and DCE.

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[5]

CPUs are incompatible with them. So in the end, we were not able to fully implement multikernel system. While we were able to boot unikraft it was not 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 flask server on Unikraft. While testing, the boot time of unikraft was found 3 seconds on average. Unikraft flask server consumes 60 milliwatt on average. On the other hand native linux flask server process consumes 20 milliwatt on average. But if we also include Linux power consumption, total power consumption will be 8000 milliwatt on average on freshly booted system.

Figure 3 is the bootscreen of Unikraft running Flask server.

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

Figure 5 is comparison of power consumption between Unikraft Flask server and Linux native Flask server without including whole Linux power consumption.

Unfortunately, due to lack of proper hardware, we were unable to verify the full potential of of multikernel system. But this system still shows potential. Our experiment shows that this system uses less power if implemented properly. We have also seen from earlier research that this system also performs better in most cases [5].

6 Conclusions

6.1 Conclusions

In conclusion, multikernel system is a promising concept but needs more support. Currently, some kernels such as Unikraft, Include os, mirage os are receiving decent support but there are still room for improvement. As the support for these kernels gets better, they should also be more available to wider range of systems.

6.2 Summary of Contributions

In this paper, we tried to explore if the multikernel system is viable alternative method to improve performance and battery life. We tried to test several mainstream sub kernels on consumer level hardware such as generic laptop and raspberry pi. We also measured boot time and power consumption on Unikraft.

6.3 Future Work

Future work includes making multikernel system more available. To do that we need to make existing kernels work with more devices or create a new configurable kernel from ground up with large array of hardware support in mind.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.