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 5, 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 [5]

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 [5]. 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 [azealea..........................].

Figure 1 shows application stack on monolithic OS vs Unikernel.

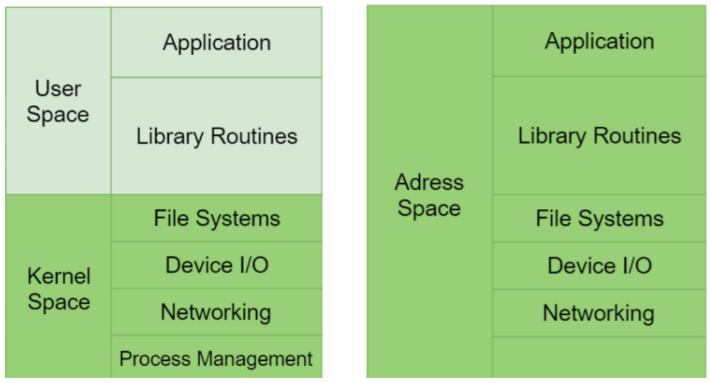

Figure 1: Application stack on monolithic OS vs Unikernel [4]

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] [unikernel].

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.

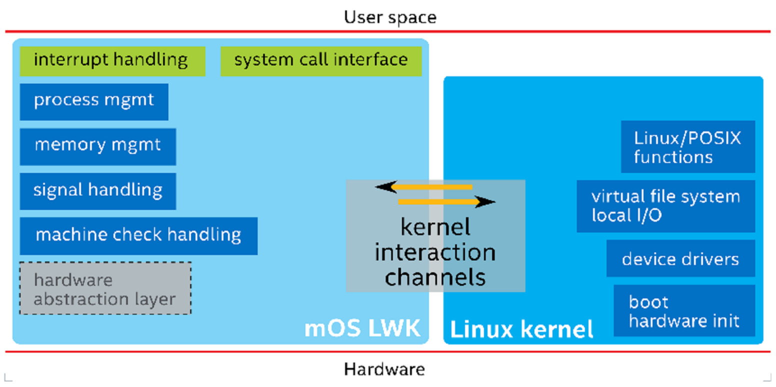

Figure 2: System Architectural Overview of Multi-kernel [5]

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

1. A concise statement of the question that your paper tackles.
2. Justification, by direct reference to your Literature Review, that your question is previously unanswered.
3. Discussion of why it is worthwhile to answer this question.

4 Proposed Solution: ...

This part of the paper is much more free-form. It may have one or several sections and subsections. But it all has only one purpose: to convince the reader that you answered the

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

Table 2: Experiment

question or solved the problem that you set for yourself. In this section you will for example present new algorithms you developed and your implementation of these new algorithms.

Figure 3 is an example of a drawing created with *mdraw* or *epsfig*.

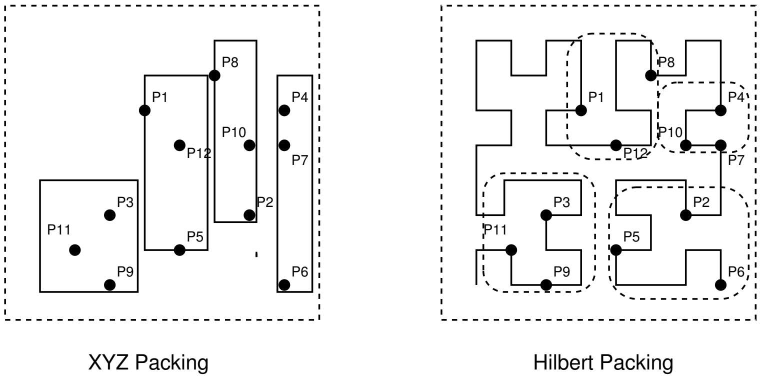

Figure 3: XYZ and Hilbert Packings

5 Experimental Evaluation

This section is not mandatory for all papers (for example theory papers) but typically required for papers in the field of parallel computing. After all, parallel computing is all about compute performance. Here you present performance data obtained from e.g. running your newly developed algorithms and code on a parallel machine using some benchmark input data. Typically, you need to describe the parallel machine you used and the data that you used as input. The main results are usually performance graphs, typically speedup curves. You want to evaluate your code on different input data sets highlighting the strengths and weaknesses of your code. Don't just use best case scenarios. People will call you on that. Discuss the results obtained.

Figure 4 is a typical example of an experimental evaluation result. Such graphs are ususally created with GnuPlot.

6 Conclusions

You generally cover three things in the Conclusions section.

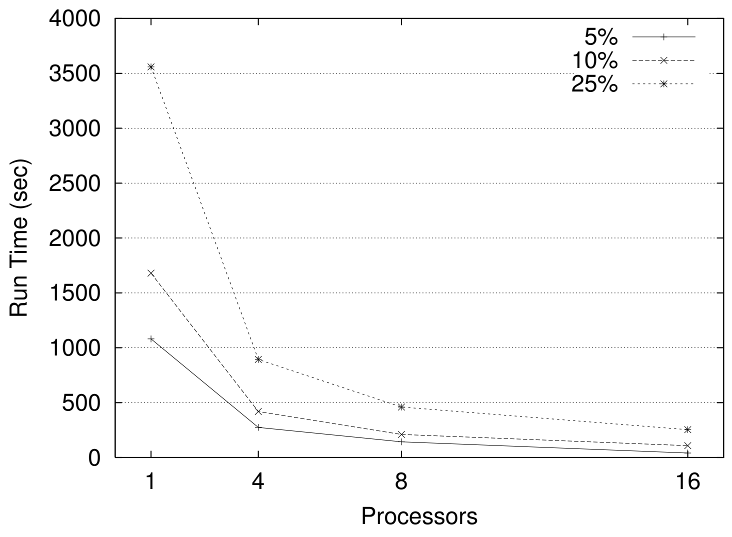

Figure 4: Measured Running Times Of Some Unknown Algorithm Implementation

1. Conclusions
2. Summary of Contributions
3. Future Research

Conclusions are not a rambling summary of the thesis: they are short, concise statements of the inferences that you have made because of your work. All conclusions should be directly related to the research question.

The Summary of Contributions will be much sought and carefully read by the readers. Here you list the contributions of new knowledge that your paper makes. Of course, the paper itself must substantiate any claims made here. There is often some overlap with the Conclusions, but that's okay.

The Future Research should indicate interesting new problems arising from your work. No paper ever solves everything. In fact, the best research papers lead to new research questions for other researchers to work on.

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

[5] 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.