



Real-Time ML Integration with CFS to Improve Power Efficiency in Non-Hybrid Linux Systems

Dissanayake M.D.
(MS24035026)

A THESIS
SUBMITTED TO
SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

December 2025

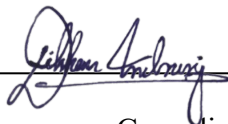
I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



4/1/2026

Prof. Samantha Rajapaksha

Approved for MSc. Research Project:



MSc in IT Programme Co-ordinator, SLIIT

Approved for MSc:

Head of Graduate Studies, FoC, SLIIT

DECLARATION

This is to certify that the work is entirely my own and not of any other person, unless explicitly acknowledged (including citation of published and unpublished sources). The work has not previously been submitted in any form to the Sri Lanka Institute of Information Technology or to any other institution for assessment for any other purpose.

Sign:

A handwritten signature in blue ink, appearing to read 'M. Dissanayake', enclosed within a light blue rectangular border.

Dissanayake M.D.

Date: 30/12/2025

ABSTRACT

Real-Time ML Integration with CFS to Improve Power Efficiency in Non-Hybrid Linux Systems

Dissanayake M.D.

MSc. in Information Technology

Supervisor: Prof. Samantha Rajapaksha

December 2025

Modern non-hybrid Linux systems rely on the Completely Fair Scheduler (CFS) and Dynamic Voltage and Frequency Scaling (DVFS) to balance performance and energy efficiency. However, this default approach has key limitations: CFS distributes tasks evenly across cores without distinguishing between workload types, while DVFS adjusts frequency based on aggregate load. As a result, CPU-bound and I/O-bound tasks often share the same cores, leading to unnecessary frequency boosts and wasted energy for tasks that don't require high frequencies. This thesis proposes a machine learning-enhanced scheduling framework that integrates task-type awareness into non-hybrid Linux systems. A decision tree classifier predicts whether tasks are CPU-bound or I/O-bound using lightweight runtime features such as execution time, waiting time, context switches, priority, and niceness values. Classified CPU-bound tasks are consolidated onto selected cores running at higher frequencies, while I/O-bound tasks are grouped on separate cores maintained at lower frequencies. To prevent thermal hotspots and performance degradation, periodic rotation of CPU-bound and I/O-bound core groups is implemented. The scheduler is evaluated against the default CFS using controlled CPU-intensive, I/O-intensive, and mixed workloads generated with Sysbench and FIO. Energy consumption, performance, and Energy-Delay Product (EDP) are measured using perf and turbostat. Experimental results show that the ML-enhanced scheduler reduces energy consumption and improves EDP significantly, while maintaining performance levels comparable to the default scheduler. These findings highlight the potential of ML-based task classification to improve DVFS utilization and deliver more energy-efficient scheduling in general-purpose, non-hybrid Linux systems.

ACKNOWLEDGEMENT

I wish to extend my deepest gratitude to my supervisor for their invaluable guidance, constructive feedback, and continuous encouragement throughout the course of this research. Their expertise and support have been instrumental in shaping the direction and outcome of this work.

I am also grateful to the faculty and staff of my university for providing the academic environment and resources necessary for the successful completion of this study.

Finally, I would like to acknowledge the unwavering support and understanding of my family and friends, whose encouragement has been a constant source of strength during this research journey.

TABLE OF CONTENTS

DECLARATION	3
ABSTRACT	4
ACKNOWLEDGEMENT	i
TABLE OF CONTENTS	ii
List of Figures	viii
List of Tables	ix
Chapter 1 Introduction	1
1.1 Background and Motivation	1
1.1.1 Computing Systems and Energy Efficiency	1
1.1.2 Processor Architecture and Scheduling in Non-Hybrid Systems	2
1.1.3 Linux Scheduling and DVFS in Non-Hybrid Systems	3
1.2 Research Gap	5
1.2.1 Limitations in Current Literature	5
1.2.2 Need for ML-Based Scheduling in Non-Hybrid Systems	5
1.3 Research Problem Context	6
1.3.1 Default CFS + DVFS Behavior	6
1.3.2 Limitations in Current Scheduling Approaches	7
1.3.3 Impact of Inefficiency	8
1.4 Research Objectives	9
1.4.1 Main Objective	9
1.4.2 Specific Objectives	9
1.5 Research Contributions	11
1.5.1 Proposed Framework	11
1.5.2 Novelty in this Research	14
1.5.3 Key Contributions	14
Chapter 2 Literature Review	16

2.1 Background	16
2.1.1 Energy Scaling in Modern Computing.....	16
2.1.2 Dynamic Voltage and Frequency Scaling (DVFS)	17
2.1.3 Rising Compute vs. Energy Constraints.....	17
2.1.4 Why Task-Aware Scheduling Matters	18
2.2 Linux Scheduling on Non-Hybrid CPUs	18
2.2.1 CFS: Fairness, vruntime, Load Balancing, and Migration Overhead	18
2.2.2 DVFS in Linux: cpufreq Drivers, Governors, and Scaling Granularity	20
2.2.3 Measurement Tools and Methodology Challenges	22
2.3 Energy-Aware Scheduling: What We Know	25
2.3.1 Heuristic task placement & consolidation (pack/idle, cluster-idle).....	25
2.3.2 Frequency capping/uncapping policies and perf-energy trade-offs	27
2.3.3 What doesn't translate from hybrid (big.LITTLE) to non-hybrid.....	27
2.4 Task Characterization & Prediction	28
2.4.1 Defining CPU-bound vs I/O-bound Tasks	28
2.4.2 Signature Features and Dynamic Indicators	29
2.4.3 Dataset and Labeling Strategies in OS Scheduling Research.....	30
2.4.4 Challenges and Gaps	31
2.5 Machine Learning for Power and Scheduling.....	31
2.5.1 ML for DVFS Prediction (Supervised & Reinforcement).....	31
2.5.2 On-device Models, ONNX Runtime & Userspace Constraints	33
2.6 Gaps in the Literature.....	34
2.6.1 Dominance of Hybrid / Asymmetric Architectures in Prior Work	34
2.6.2 Lack of Task-Type-Aware Placement Coupled with DVFS in General Linux	35
2.6.3 Lack of Reproducible Evaluation with Open Tools & Standard Workloads	36
Chapter 3 Research Design and Methodology.....	38
3.1 Research Design.....	38

3.2 Data Collection and Feature Extraction	40
3.2.1 Data Collection	40
3.2.2 Data Cleaning and Preprocessing	40
3.2.3 Feature Engineering	41
3.2.4 Labeling Strategy	42
3.2.5 Dataset Summary	42
3.3 Machine Learning Model Development	42
3.3.1 Model Selection Rationale	43
3.3.2 Training Pipeline	43
3.3.3 Model Evaluation	43
3.3.4 Model Export and Deployment	44
3.4 Integration into Linux Scheduling	44
3.4.1 Architecture of the ML-Enhanced Scheduling Approach	44
3.4.2 Periodic Process Monitoring	46
3.4.3 ML-Based Classification of Tasks	46
3.4.4 Core Rotation to Avoid Hotspots	47
3.5 Workload Design for Evaluation	47
3.5.1 CPU-Bound Workloads	47
3.5.2 I/O-Bound Workloads	48
3.5.3 Mixed Workloads	48
3.5.4 Justification for Workload Selection	48
3.6 Experimental Setup	49
3.6.1 Hardware Configuration	49
3.6.2 Software Environment	49
3.6.3 Monitoring and Measurement Tools	50
3.6.4 Baseline vs. ML-Enhanced Setup	50
3.7 Evaluation Metrics	51

3.7.1 Energy Consumption	51
3.7.2 Performance.....	51
3.7.3 Energy-Delay Product (EDP)	52
3.7.4 Statistical Testing	52
Chapter 4 Results and Discussion.....	53
4.1 Introduction	53
4.2 Workload Validation.....	54
4.2.1 Workload Composition.....	54
4.2.2 Observed Behavior of Workloads	54
4.3 Baseline Results (Default CFS + DVFS)	55
4.3.1 Performance Metrics.....	56
4.3.2 Energy Consumption Metrics	56
4.3.3 Key Observations	57
4.4 ML-Enhanced Scheduler Results	57
4.4.1 Performance Metrics.....	57
4.4.2 Energy Metrics	58
4.5 Comparative Analysis Between Baseline and ML Schedulers	58
4.5.1 Energy Consumption Comparison.....	58
4.5.2 Performance Metrics Comparison	61
4.5.3 Analysis of Results	61
4.6 Discussion of Findings.....	62
4.6.1 Relating Back to Research Objectives.....	62
4.6.2 Key Insights.....	63
4.6.3 Addressing Limitations.....	63
4.6.4 Conclusion of Findings.....	63
4.7 Implications of Results.....	65
4.7.1 Implications for Non-Hybrid Systems.....	65

4.7.2 Implications for Datacenters and Large-Scale Systems	65
4.7.3 Implications for Future Research	65
4.8 Summary of Results	65
Chapter 5 Conclusion and Future Work	67
5.1 Key Findings and Contributions	67
5.1.1 Energy-Delay Product	67
5.1.2 Frequency Scaling	67
5.1.3 Energy Efficiency	67
5.1.4 Performance Impact.....	68
5.1.5 Statistical Validation.....	68
5.2 Contribution Summary	68
5.3 Implications of the Study	69
5.3.1 Implications for Non-Hybrid Linux Systems	69
5.3.2 Implications for Data Centers and Large-Scale Systems	70
5.3.3 Implications for the Research Community	70
5.3.4 Implications for Sustainability and Green Computing	71
5.4 Future Work	71
5.4.1 Kernel-Level Integration	72
5.4.2 Model Enhancement.....	72
5.4.3 Expanded Evaluation.....	73
5.4.4 Integration with Cloud and Edge Systems	73
5.4.5 Energy-Aware Reinforcement Learning	74
Chapter 6 References	76
Appendix.....	80
Appendix 1: Data preprocessing, feature engineering and labeling script.....	80
Appendix 2: ML model training script.....	84
Appendix 3: ML enhanced scheduler script.....	86

Appendix 4: Workload for evaluation.....	89
Appendix 5: Evaluation metrics collection script	90
Appendix 6: Statistical significance testing script	92

List of Figures

Figure 1.1 Conceptual Framework	13
Figure 3.1 Research Methodology Flow Chart.....	38
Figure 4.1 Energy-Delay Product (EDP) Comparison	59
Figure 4.2 Frequency Comparison.....	59
Figure 4.3 Energy Comparison	60
Figure 4.4 Busy Rate Comparison	60

List of Tables

Table 4.1 Performance Metrics of Baseline CFS Scheduler.....	56
Table 4.2 Energy Metrics of Baseline CFS Scheduler	56
Table 4.3 Performance Metrics of ML-enhanced Scheduler	57
Table 4.4 Energy Metrics of ML-enhanced Scheduler.....	58
Table 4.5 Energy Metrics Comparison	58
Table 4.6 Performance Metrics Comparison	61