

System Software for Energy-Efficient Computing

Benedict Herzog

Abstract

Manufacturers have consistently enhanced the performance and efficiency of computer hardware for decades. This has enabled computers to evolve from kilowatt machines into milliwatt systems while multiplying their performance. Although efforts to improve the hardware are ongoing, they have significantly slowed down or, in some cases, come to a halt. This is primarily due to constraints imposed by manufacturing technologies on electrical power and energy demand. At the same time, the demand for more computing power continues to grow rapidly, largely driven by the enormous power hunger of machine learning and artificial intelligence. However, the challenges of further improving the hardware, coupled with the pressing challenges of the climate crisis and technological limitations, render current approaches of *simply deploying more hardware* unsustainable. Instead, major improvements today are no longer found in the next hardware generation but rather at the software level, especially in the efficient coordination of software and hardware.

The energy efficiency of a system is defined by the energy consumed per unit of work. One prerequisite for an energy-efficient system is its ability to be aware of and respond to its energy demand. In turn, the system must be able to determine its energy demand to be energy aware. In this context, the operating system takes on a key role in the interaction between software and hardware. It serves as the layer that facilitates hardware accessibility to software on the one hand and controls and manages software execution on the other. Until now, general-purpose operating systems, such as Linux or Windows, aim to support as many systems as possible, sacrificing efficiency in favour of mass compatibility and commodity—not least because they are designed to run seamlessly across devices ranging from smartphones to data centres. This thesis posits the opposite: the operating system, hardware, and applications must be tailored to each other for an energy-efficient system.

To this end, this thesis presents, implements, and evaluates three components to support the operation and construction of energy-efficient systems. It contributes a *monitoring* component that tracks and attributes the energy demand of software at the operating system level. This component aids the optimisation efforts of system operators and programmers for existing and new software. The capabilities of the monitoring component are demonstrated through two use cases: firstly, it identifies and quantifies the previously unknown energy overhead of software mitigations against the Meltdown and Spectre hardware vulnerabilities; and secondly, it contributes a novel extension to the flame graph visualisation—*energy flame graphs*—that identifies code paths with excessive energy demand.

Building upon the monitoring component, the *specialisation* component enables automated system specialisation based on the currently executing software and hardware, thereby improving the energy efficiency of the system. It analyses the energy demand in an exploration phase, as tracked by the monitoring component, to create application and hardware profiles. In the subsequent optimisation phase, it acts upon this profile information to adapt the system's configuration to the applications and hardware. This component is implemented and evaluated for the Linux operating system.

The knowledge and tools acquired in the preceding two components are eventually employed in the *software construction* component. Its primary objective is to construct energy-efficient software from the outset rather than enhancing energy efficiency retrospectively. The component utilises software generation and energy models to automate software construction, thereby facilitating the development of energy-efficient software. Specifically, it is implemented for the development of energy-efficient data-processing pipelines for embedded systems.

In this thesis, these components form a comprehensive framework to support the operation and construction of energy-efficient systems by efficiently coordinating software and hardware. This makes it possible to effectively meet the growing demand for computing power and to fully exploit the capabilities of the employed hardware. In doing so, they complement hardware advancements and contribute to the sustainable improvement of performance and energy efficiency of current and future computer systems.

System Software for Energy-Efficient Computing

Benedict Herzog

Zusammenfassung

Hardwarehersteller haben die Leistungsfähigkeit und Effizienz von Computerhardware über Jahrzehnte hinweg kontinuierlich gesteigert. Dadurch konnten sich Computer von Maschinen mit einer Leistungsaufnahme im Kilowattbereich hin zu Geräten mit wenigen Milliwatt Leistungsaufnahme entwickeln—bei einem Vielfachen der Rechenleistung. Auch wenn die Anstrengungen zur Verbesserung der Hardware fortgeführt werden, hat sich die Entwicklung inzwischen erheblich verlangsamt. Hauptursache hierfür sind die technologischen Grenzen in der Hardwarefertigung, insbesondere im Hinblick auf den elektrischen Leistungs- und Energiebedarf. Gleichzeitig steigt der Bedarf nach mehr Rechenleistung kontinuierlich an, nicht zuletzt durch den enormen Leistungshunger maschinellen Lernens und Künstlicher Intelligenz. Die Herausforderungen bei der Weiterentwicklung der Hardware, gepaart mit der Klimakrise und technologischen Limitierungen, machen bisherige Herangehensweisen, wie *einfach mehr Hardware einzusetzen*, zunehmend unhaltbar. Stattdessen liegen heute Potenziale für signifikante Leistungssteigerungen nicht in der nächsten Hardwaregeneration, sondern im effizienteren Zusammenspiel von Hardware und Software.

Die Energieeffizienz eines Systems wird durch den Energiebedarf pro Arbeitseinheit definiert. Eine grundlegende Voraussetzung für ein energieeffizientes System ist, dass es sich seines Energiebedarfs gewahr ist. Dazu muss das System seinen Energiebedarf bestimmen können. In diesem Zusammenhang übernimmt das Betriebssystem eine Schlüsselrolle: Es fungiert als Vermittler, der einerseits die Hardware für die Software zugänglich macht und andererseits die Softwareausführung steuert und kontrolliert. Universalbetriebssysteme wie Linux und Windows sind bisher darauf ausgelegt, möglichst viele Systeme zu unterstützen und opfern dabei Energieeffizienz zugunsten breiter Kompatibilität und Massentauglichkeit. Diese Arbeit vertritt die gegenteilige Position: Betriebssystem, Hardware und Anwendungen müssen gezielt aufeinander abgestimmt werden, um ein energieeffizientes Gesamtsystem zu ermöglichen.

Zu diesem Zweck präsentiert, implementiert und evaluiert diese Arbeit drei Komponenten, die den Betrieb und die Entwicklung energieeffizienter Systeme unterstützen. Zunächst wird eine Komponente für das *Monitoring* auf Betriebssystemebene vorgestellt, die den Energiebedarf von Software erfasst und zuordnet. Diese Komponente unterstützt Systembetreiber und Programmierer bei der Optimierung existierender sowie neuer Software. Die Leistungsfähigkeit der Komponente wird durch zwei Anwendungsfälle demonstriert: Zum einen identifiziert und quantifiziert sie den bislang unbekanntem Energieoverhead von Softwaremitigierungen gegen die Meltdown und Spectre Sicherheitslücken. Zum anderen führt sie eine neue Erweiterung für die Visualisierung mittels Flame Graphs ein—*Energy Flame Graphs*—mit deren Hilfe besonders energieintensive Codepfade identifiziert werden können.

Aufbauend auf der Komponente für das Monitoring ermöglicht eine Komponente für die *Spezialisierung* die automatische Anpassung des Systems an die aktuell ausgeführte Software und Hardware, um die Energieeffizienz zu steigern. Dazu analysiert sie den Energiebedarf des Systems in einer Explorationsphase und erstellt Anwendungs- und Hardwareprofile. In der anschließenden Optimierungsphase wendet es die Informationen aus den Profilen auf die aktuelle Systemkonfiguration an. Diese Komponente wurde für Linux implementiert und evaluiert.

Das gewonnene Wissen und die Werkzeuge der beiden vorherigen Komponenten münden schließlich in der dritten Komponente für die *Softwareerstellung*. Deren Ziel ist es, bereits bei der Entwicklung von Software Energieeffizienz zu berücksichtigen, statt diese erst nachträglich zu optimieren. Sie nutzt Softwaregenerierung und Energiemodelle zur Automatisierung der Softwareentwicklung und erleichtert so die Erstellung energieeffizienter Anwendungen. Für diese Arbeit wurde diese Komponente für die Entwicklung energieeffizienter Datenverarbeitungspipelines für eingebettete Systeme realisiert.

Zusammen bilden die Komponenten dieser Arbeit ein umfassendes Rahmenwerk zur Unterstützung des Betriebs und der Entwicklung energieeffizienter Systeme. Dies ermöglicht es, der wachsenden Nachfrage nach mehr Rechenleistung wirksam zu begegnen und das Potenzial der eingesetzten Hardware vollständig auszuschöpfen. Zugleich kompensieren sie die langsameren Fortschritte in der Hardwareentwicklung und tragen zur nachhaltigen Steigerung der Leistungsfähigkeit und Energieeffizienz aktueller sowie zukünftiger Computersysteme bei.