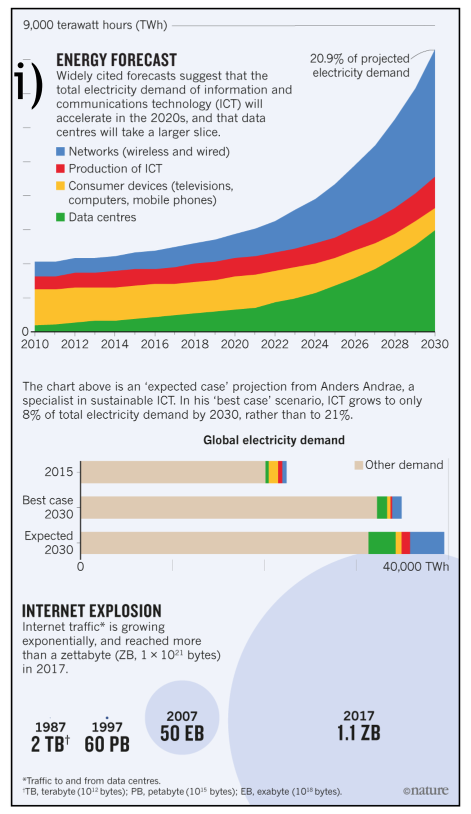
Applying Machine Learning to Linux Kernel Configurations for Performance and Energy

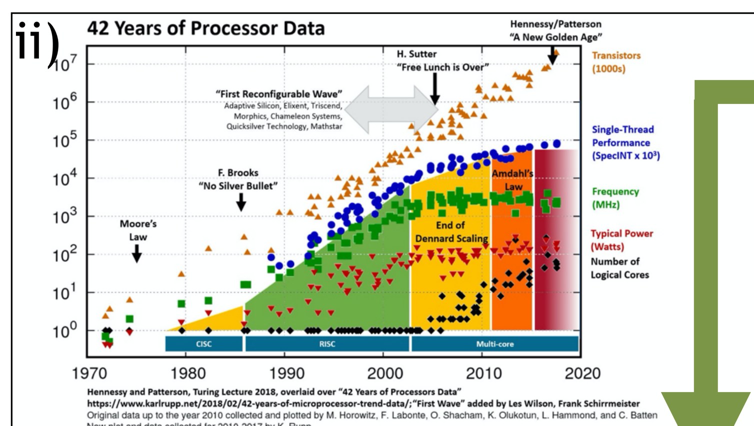
Han Dong[1+], Jonathan Appavoo[1], Sanjay Arora[2]
[1]Department of Computer Science, Boston University. [2]Red Hat, Inc.
[+]PhD candidate (graduating Fall'22)
SESA@BU http://sesa.github.io

Motivation

Increasingly constrained **i)** energy budgets and stagnation in **ii)** silicon technology poses new challenges of how modern software and hardware can meet the performance and energy requirements of modern network-based applications.

We identify two fundamental properties in network processing and demonstrate how a ML-based approach can be used to extract more value out of existing software and hardware.

i) 

ii) 

Main Findings

In order to effectively exploit ML, we conducted a rigorous operating system (OS) experimental study and discovered that using two hardware mechanisms to tune **1) the speed of network interrupts and 2) the speed of request processing**, enables application performance and energy to be controlled in a well-structured manner. This structure can then be exploited by a ML technique, such as Bayesian optimization, to automatically configure Linux in order to improve its performance and energy efficiency by over 50% for a broad class of network-driven applications.

Findings **1**, **2**, and **3** detail the main discoveries that document our research approach; from characterizing the two mechanisms, detailing its effects, and to the deployment of Bayesian optimization for automatic tuning.

Finding 3: Applying ML to tune Linux

Our study also reveals application behavior that is stable and well-structured, and these behaviors can then be exploited by ML techniques. Fig. 3 demonstrates that a Linux memcached server can be automatically tuned using Bayesian optimization to minimize energy use for a real world workload trace from Twitter*. Our method allows Linux-tuned to use over 50% less power than default Linux and has the potential to radically improve real world data-center applications and total-cost-of-ownership (TCO) savings.

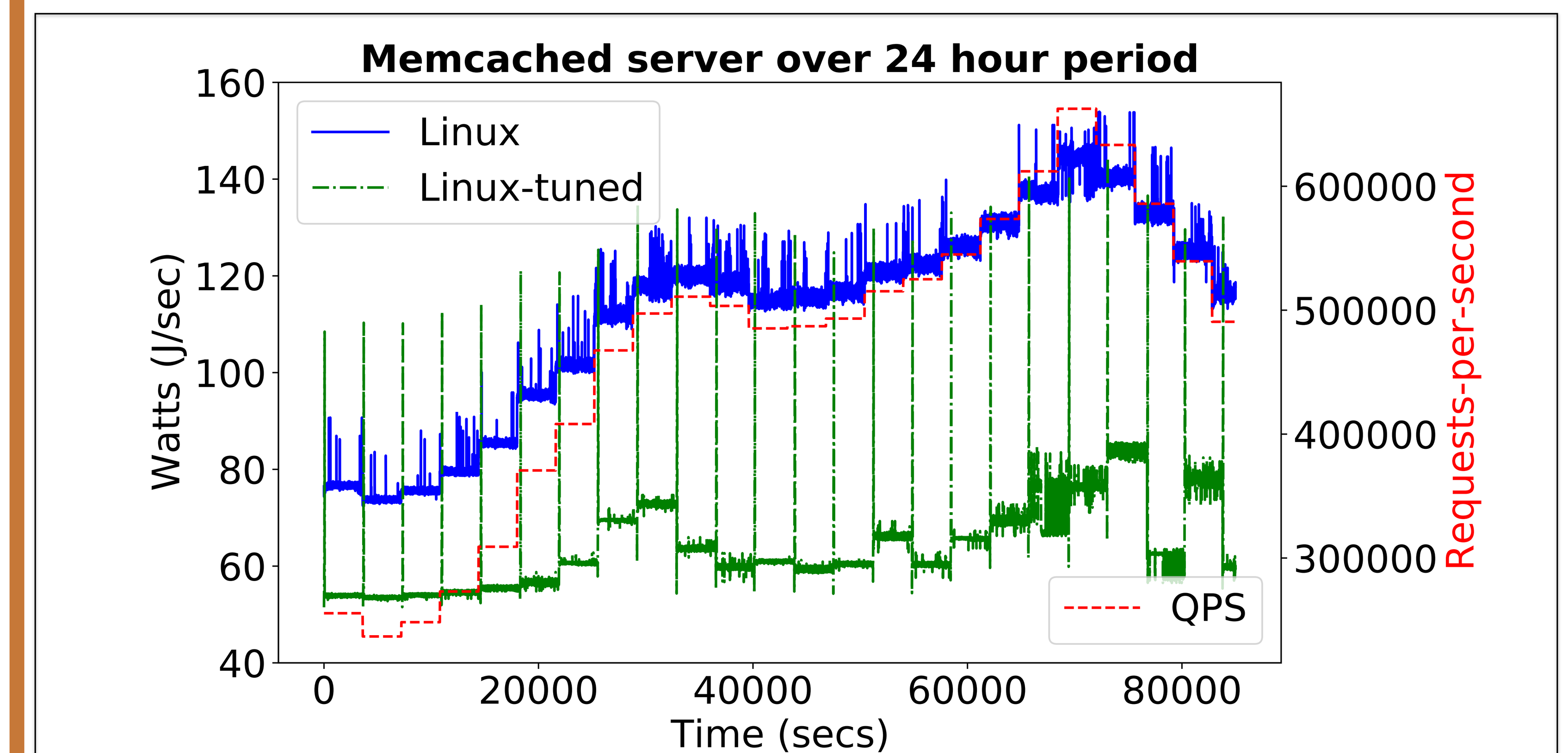


Fig 3: Requests-per-second shows the changing request rate on a hourly basis over a whole day. This figure shows the watt (joules/sec) usage between default Linux and ML-enhanced Linux-tuned during this period.

*https://github.com/twitter/cache-trace

Finding 1: Improving Linux Performance and Energy Efficiency with Two Hardware Mechanisms

Our experimental study involved four broad classes of network applications; from a NodeJS web server to a high performance memcached-database server. From the study, we find that Linux can be configured in new ways to improve its performance and energy efficiency. As an example, Fig. 1 shows that tuning Linux can result in both improved performance and energy efficiency for a NodeJS web server by over 10%.

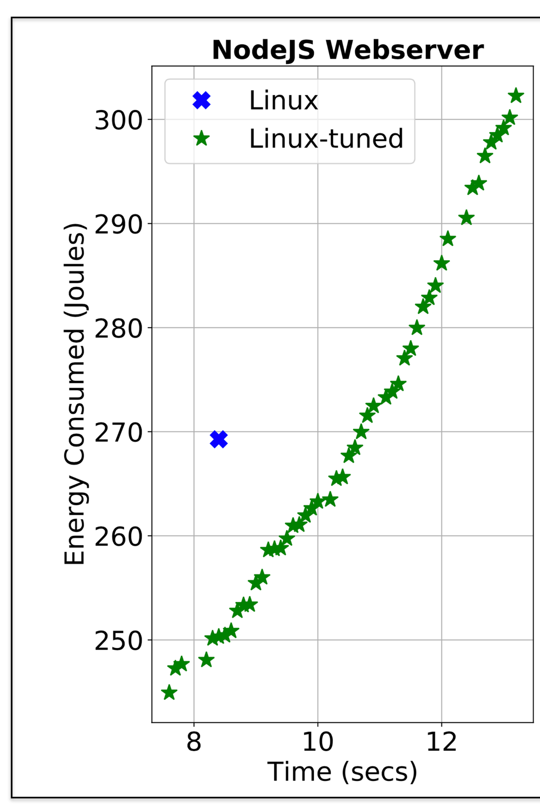


Fig. 1: Linux is running kernel version 5.5 and Linux-tuned is using the same kernel but with the two mechanisms controlled in new ways. Each marker is a single experimental run of the NodeJS web server. Energy consumed (Joules) and performance measured (Seconds) are shown for both systems (lower is better).

Finding 2: Performance and Energy Opportunities from Optimizing the Kernel

We find the effects of tuning in a specialized OS can further magnify its benefits. Fig. 2 illustrates how OS specialization via an entirely different OS (LibOS) results in over 100% improved performance and energy efficiency. This finding suggests there can be dramatic gains by adopting more specialization techniques in modern kernels.

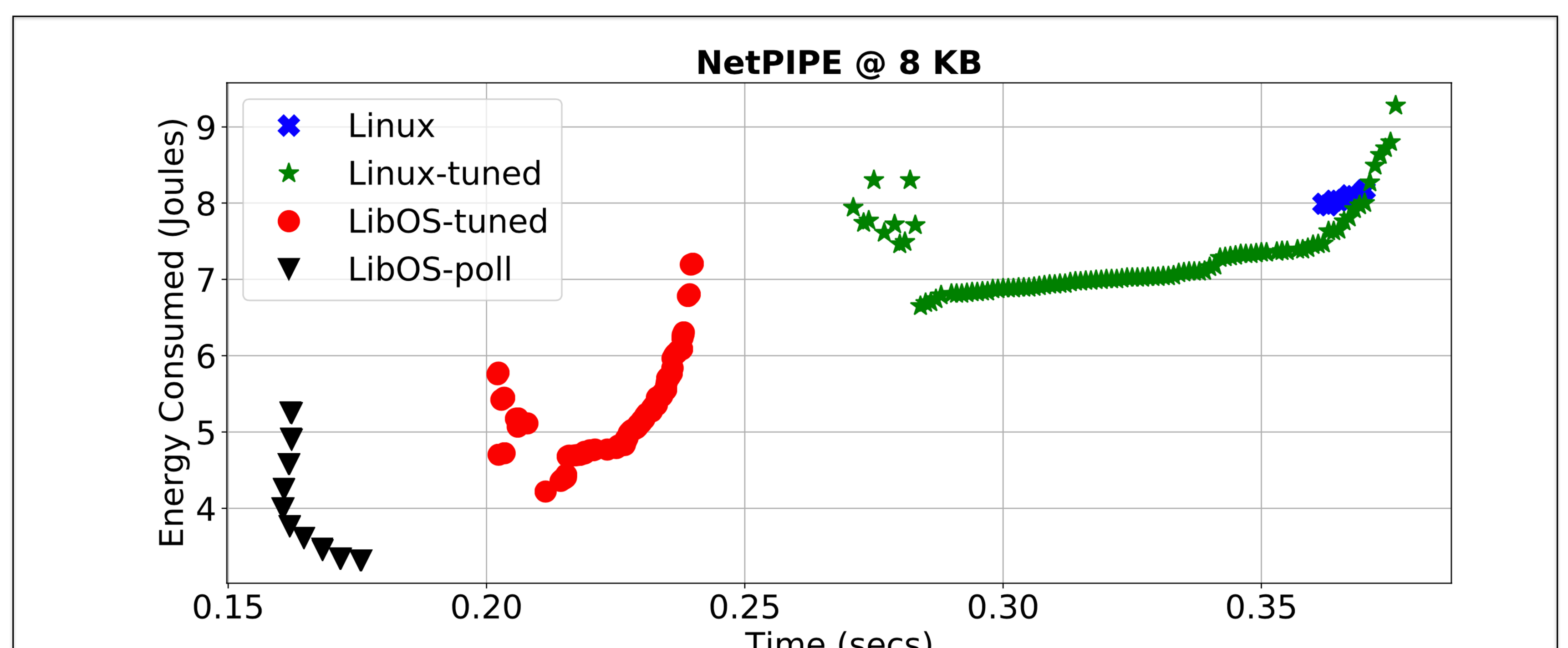


Fig. 2: NetPIPE is a simple application that ping-pongs fixed sized messages between two nodes. LibOS-tuned is an OS written from scratch with optimized network processing paths and LibOS-poll uses a per-core polling loop instead of being interrupt-driven.

Future Work

The structured approach we undertook in this project enabled us to exploit ML for the OS in a meaningful way to achieve dramatic gains in modern datacenter applications. For each of the three findings, we detail exciting next steps in this work:

Finding 1: While we've only focused on two mechanisms; different types of hardware (SSDs, RAM, GPUS, etc.) and software (e.g. Linux sysctl, .config) settings suggest there are further opportunities to optimize Linux.

Finding 2: New opportunities from OS specialization in the LibOS suggest open source Linux projects (e.g. Unikernel Linux, Chronokernel) can integrate new energy efficient optimizations within the kernel.

Finding 3: The benefits of Bayesian optimization approach suggests one can implement simple load balancers in datacenters that drive requests to correctly pre-configured servers to minimize energy. Further, it is another step towards self-adapting systems to enable a new generation of system policies that are completely automated.

handong@bu.edu, saarora@redhat.com, jappavoo@bu.edu