Time and Energy-Aware Computation

Yara Awad[1] (awadyn@bu.edu) Han Dong[1] (handong@bu.edu) Jonathan Appavoo[1] (jappavoo@bu.edu) Sanjay Arora[2] (saarora@redhat.com)

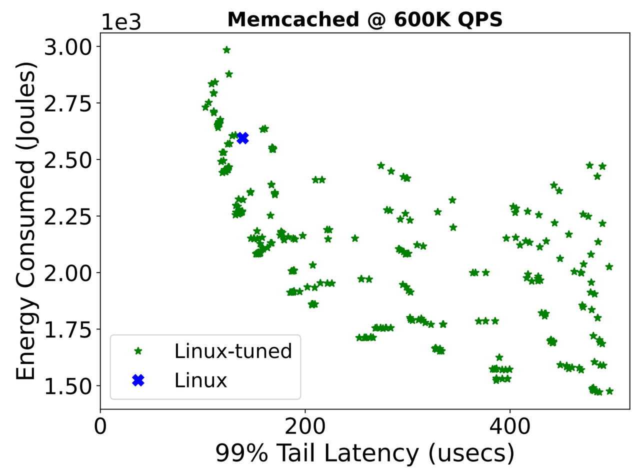
[1]Department of Computer Science, Boston University [2]Red Hat, Inc.

Motivating Work

Each point is a run where a unique energy/performance setting was issued through the ITR and DVFS registers.

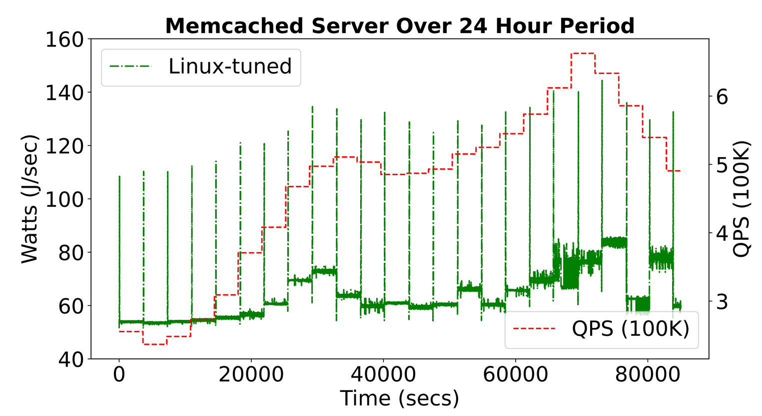
In Green: latency achieved and energy consumed with a unique, stable setting of ITR and DVFS parameters.

In Blue: mean latency/energy achieved when the default system control algorithms set ITR and DVFS parameters.



A curve of energy - *joules* - and performance - *99% tail latency* - of many runs of a Memcached/Linux software stack subject to a rate of 600K requests per second (QPS).

Structure is revealed *in the curve of varying energy/performance points achieved as the system responds to* ***controlled changes of ITR and DVFS settings.***



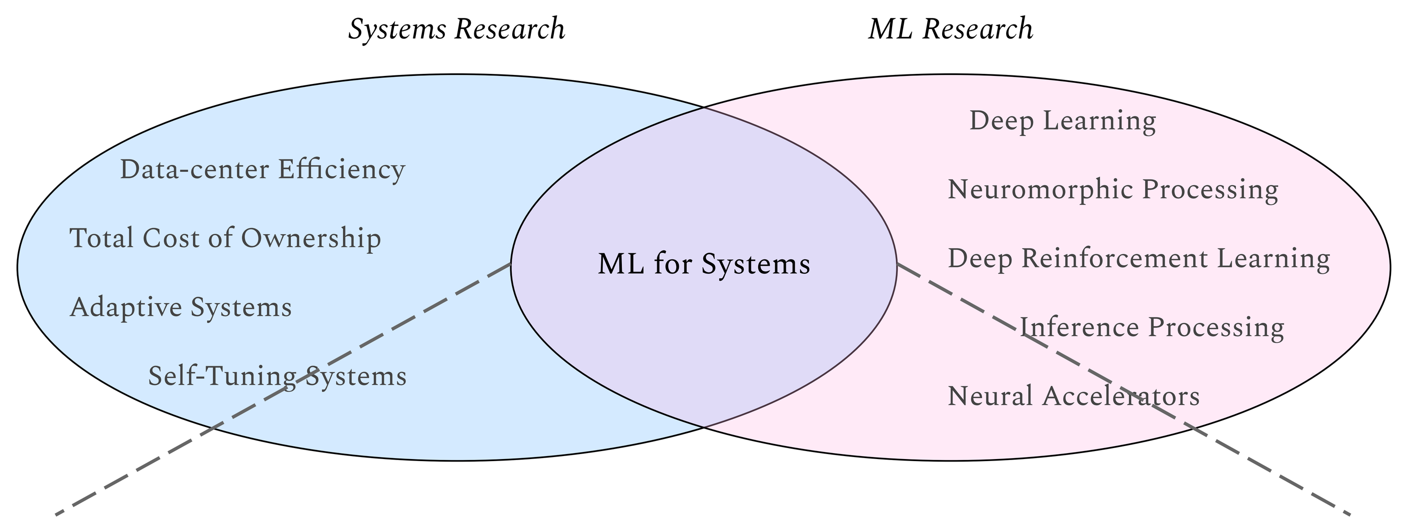
Each spike-and-dip pair represents the process of a Bayesian optimizer as it responds to a newly introduced QPS rate and ultimately finds a new optimal ITR/DVFS setting.

In Red: a step function showing moments at which the QPS rate is manually changed across a 24 hour period.

In Green: the energy/performance achieved as a Bayesian optimizer searches for optimal ITR/DVFS settings.

A Bayesian optimization algorithm traversing different curves of energy/performance targets, where each curve corresponds to the software stack behavior subject to a different QPS rate.

A learnable relationship is revealed *between ITR and DVFS settings and the resulting energy/performance achieved. When directed to do so, a Bayesian optimization algorithm is able to exploit this relationship to find the most optimal settings for execution.*



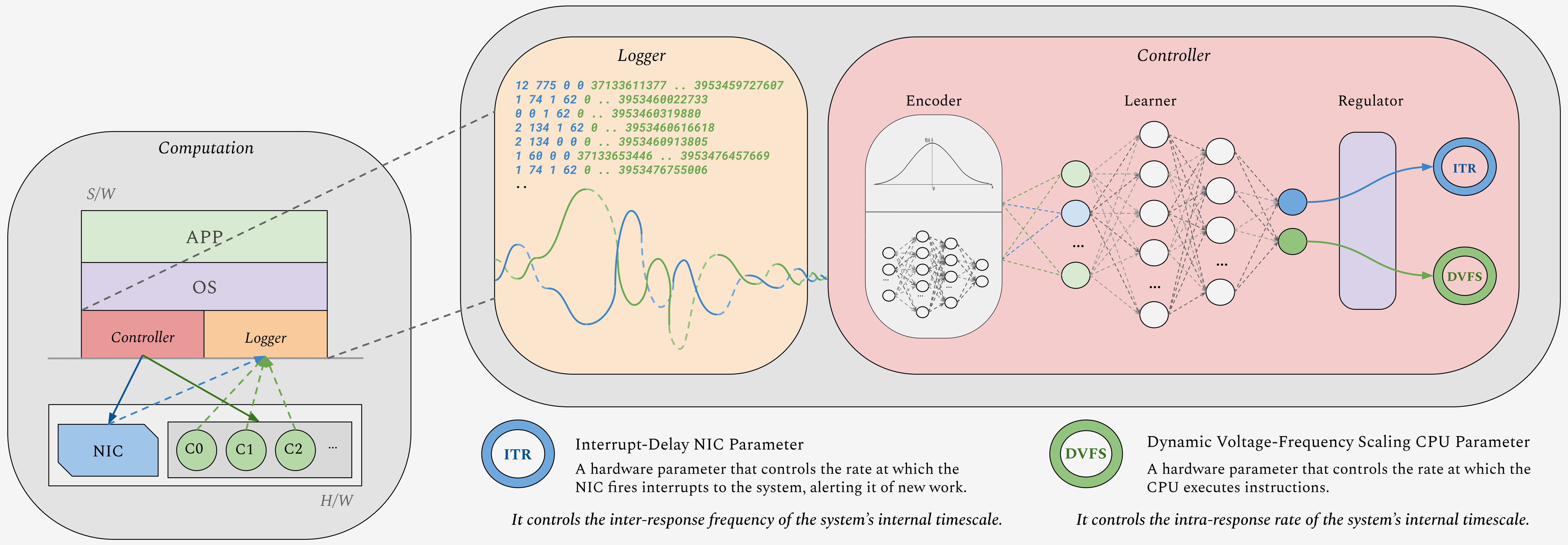
Structure and learnability promote the hypothesis that ***deep learning techniques integrated with fine-grain, meaningful system logs can enable the development of a self-regulating system,*** *capable of detecting internal and external changes and adapting to them by automatically adjusting energy/performance settings to optimize execution.*

Idea

We deduce that network-bound software stacks, influenced externally by varying QPS rates, will operate within different ***timescales***, or ***time signatures***, defined by an **inter-response frequency and intra-response rate**.

We believe that *there exists a timescale for which a target execution consumes minimal energy and exhibits optimal or sub-optimal performance.*

We propose *impedance matching, or rate matching,* as the missing system primitive that would match the *internal timescale of the system*, via energy/performance configuration, to the external timescale of the world it is responding to.



Approach

The methodology lies in developing a system component that is

1. able to learn the properties of ideal timescales for target software stacks and
2. automatically improve the system's energy and performance by configuring it to that ideal timescale under any QPS rate.

This component must be designed as a **dynamic control mechanism**, based on a model that can learn from a **time-varying, multidimensional, signal-based** interpretation of the target execution.

Objectives: *Toward an Architecture for Learnable Energy/Performance Control Policies*

Given prior data that exposes log-based execution signals for a set of software stacks, three stages align toward developing a dynamic energy/performance controller:

1. Developing a numerical encoding of the execution signal
2. Learning, from encodings of executions subject to different QPS rates, the characteristic energy/performance behavior of a target software stack
3. Configuring the host system, through some feedback cycle from the controller to ITR and DVFS system drivers, toward a more optimal timescale

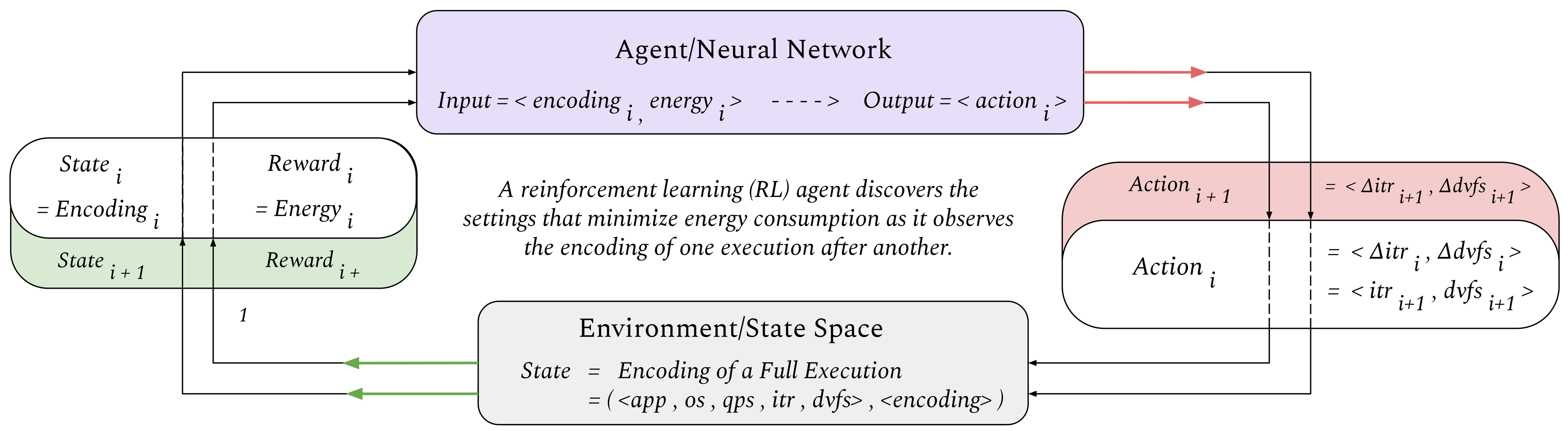
1. Encoding

From an execution log to a multi-dimensional vector representative of the log's energy and performance characteristics.

1. Manual statistics, characterizing logs through their constituent percentiles.
2. Automated statistics, computed by a recurrent neural network (RNN) as it observes execution logs across time and produces a vector representative of the full execution.

2. Learning

From the state-space of execution encoding to a mapping between execution and optimal ITR/DVFS settings.



3. System Regulation

From one execution state to another, improving energy performance through a more optimal setting of ITR and DVFS.

$$itr_{i+1} = itr_i + + / - -$$
$$dvfs_{i+1} = dvfs_i + + / - -$$

For an arbitrary *(app, os, cpu, nic)*, the controller has knowledge about optimal execution settings for arbitrary QPS rates such that it is able to always set the system settings for an optimal energy/performance.