THE UNIVERSITY OF CHICAGO

FRAMEWORKS FOR PROVIDING SUPPORT FOR GENERAL-PURPOSE ADAPTATION IN COMPUTING SYSTEMS

A DISSERTATION PROPOSAL SUBMITTED TO
THE FACULTY OF THE PHYSICAL SCIENCES DIVISION
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY

AHSAN PERVAIZ

CHICAGO, ILLINOIS

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Modern computer systems are becoming increasingly complex. Apart from functional correctness, they are expected to provide strict guarantees on quality-of-service (QoS), expressed as goals, in face of unpredictable changes in operating conditions and workload. Furthermore, these systems have a plethora of configurable variables that, combined with unpredictable external conditions, impact the system's QoS. All aforementioned factors make it difficult to optimally deploy and operate modern systems.

The systems community views adaptation as a crucial capability for systems to deliver reliable quantitative behavior in the presence of dynamic operating conditions. Prior work suggests programming frameworks that can simply be instantiated by system. The framework then monitors the system behavior and changes its configurations on behalf of the system to ensure that the quantifiable goals are met despite unpredictable external changes. However, A major limitation of prior frameworks is that they are implemented for a specific, narrow set of goals and knobs. Hence, they cannot be used for complex adaptive systems that must meet different goals using different sets of knobs for different deployments, or different execution stages of one deployment. For such scenarios developers are expected to embed different frameworks in their systems to support different goals. This, in turn, increases the size of the system code base and makes development and deployment more difficult.

In our research we explore the benefits of providing a single generalized adaptation framework that is agnostic of knobs and goals. We show the deployment and runtime benefits of using such a framework in a number of real-world systems including a networked video analytics pipeline. Our research shows that it is not only possible to implement a generalized adaptation framework but that using such a framework is more favorable from a development, deployment and performance perspective.

CHAPTER 1

THESIS STATEMENT

Adaptation is crucial in modern computer systems to meet quality-of-service goals in the presence of unpredictable changes in operating conditions and workload. However, implementing adaptive systems is a specialized skill. Requiring application developers to implement adaptive modules for their systems creates a burden for developers who must be experts, not only in their domain, but also in building adaptive systems and their underlying principles, e.g. control theory or machine learning. Prior work suggests adaptation frameworks to alleviate this burden. However, several fundamental limitations in the suggested frameworks restrict the scenarios and applications for which they can be used. Hence, in this body of research we propose a fully generalized adaptation framework that, with minimal development effort, can be used to add adaptation to systems irrespective of the underlying configurable components or goals.

CHAPTER 2

DDS: SERVER-DRIVEN STREAM FOR VIDEO ANALYTICS

Internet video must balance between maximizing application-level quality and adapting to limited network resources. This perennial challenge has sparked decades of research and yielded various models of user-perceived quality of experience (QoE) and QoE-optimizing streaming protocols. In the meantime, the proliferation of deep learning and video sensors has ushered in new analytics-oriented applications (*e.g.,* urban traffic analytics and safety anomaly detection [9, 3, 6]), which also require streaming videos from cameras through bandwidth-constrained networks [7] to remote servers for deep neural nets (DNN)-based inference. We refer to it as *machine-centric video streaming*. Rather than maximizing human-perceived QoE, machine-centric video streaming maximizes for DNN *inference accuracy*. This contrast has inspired recent efforts to compress or prune frames and pixels that may not affect the DNN output (*e.g.,* [73, 18, 19, 17, 75, 42, 70, 22]).

A key design question in any video streaming system is *where to place the functionality of deciding which actions can optimize application quality* under limited network resources. In traditional streaming schemes, it is the content source that decides how the video should be best compressed and streamed. Such schemes can be referred to as *source-driven* approaches. For example, in internet videos streaming (*e.g.,* YouTube, Netflix), the server (the source) encodes a video at several pre-determined bitrate levels, and although the mainstream protocol, DASH [4], is dubbed a client-driven protocol, the client does not provide any instant user feedback on user-perceived QoE to let server re-encode the video. Current machine-centric video streaming relies largely on the camera (the source) to determine which frames and pixels to stream.

We argue that it is suboptimal for analytics-oriented applications. The source-driven approach hinges on two premises: (1) the application-level quality can be estimated by the video source, and (2) it is hard to measure user experience directly in real time. Both need

to be revisited in machine-centric video streaming.

First, it is inherently difficult for the source (camera) to estimate the inference accuracy of the server-side DNN by itself. Inference accuracy depends heavily on the compute-intensive feature extractors (tens of neural network layers) in the server-side DNN. The disparity between most cameras and GPU servers in their compute capability means that any camera-side heuristics are unlikely to match the complexity of the server-side DNNs. This mismatch leads to the suboptimal performance of the source-driven protocols. For instance, some works use inter-frame pixel changes [17] or cheap object detectors [75] to identify and send only the frames/regions that contain new objects, but they may consume more bandwidth than necessary (*e.g.,* background changes causing pixel-level differences) and/or cause more false negatives (*e.g.,* small objects could be missed by the cheap camera-side object detector).

Second, while eliciting real-time feedback from human users may be hard, DNN models can provide *rich* and *instantaneous* feedback. Running an object-detection DNN on an image returns not only detected bounding boxes, but also additional feedback for free, like the confidence score of these detections, intermediate features, etc. Moreover, such feedback can be extracted on-demand by probing the DNN with extra images. Such abundant feedback information has not yet been systematically exploited by prior work.

In this work, we explore an alternative *DNN-driven* approach to machine-centric video streaming, in which video compression and streaming are driven by how the server-side DNN reacts to real-time video content. DNN-driven video streaming follows an *iterative* workflow. For each video segment, the camera first sends it in low quality to the server for DNN inference; the server runs the DNN and derives some *feedback* about the most relevant regions to the DNN inference and sends this feedback to the camera; and the camera then uses the feedback to re-encode the relevant regions in a higher quality and sends them to the server for more accurate inference. (The workflow can have multiple iterations though this work only considers two iterations). Essentially, by deriving feedback directly from the

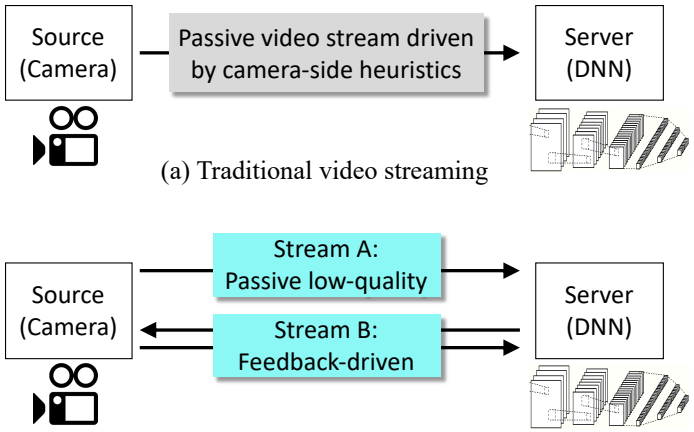


Figure 2.1: Difference DNN-based and traditional video streaming protocols.

server-side DNN, it sends high-quality content only in the minimal set of relevant regions necessary for high inference accuracy. Moreover, unlike prior work that requires camera-side vision processing or hardware support (*e.g.,* [75, 17, 42]), we only need standard video codec on the camera side. Figure 2.1 illustrates the difference between traditional video streaming frameworks and a DNN-driven video streaming framework.

The challenge of DNN-driven protocols, however, is *how to derive useful feedback from running DNN on a low-quality video stream.* We present *DDS* (*D*NN-*D*riven *S*treaming), a concrete design which utilizes the *feedback region* derived from DNN output on the low-quality video during the first iteration and sparingly uses high-quality encoding for the relatively small number of regions of interest for the second iteration. We apply DDS to three vision tasks: object detection, semantic segmentation, and face recognition. The insight is that the low-quality video may not suffice to get sufficient DNN inference accuracy, but it can produce surprisingly accurate feedback regions which intuitively require higher quality for the DNN to achieve desirable accuracy. Feedback regions are robust to low-quality videos because they are more akin to binary-class tasks (i.e. *whether* a region might contain an object and need higher quality) than to more difficult tasks such as classifying *what* object is in each region. Moreover, DDS derives feedback regions from DNN output without extra

GPU overhead.

DDS is not the first to recognize that different pixels affect DNN accuracy differently, *e.g.,* prior works also send only selected regions/frames to trigger server-side inference [75, 49]. But unlike DDS, these regions are selected either by simple camera-side logics [75] which suffer from low accuracy, or by region-proposal networks (RPNs) [49] which are designed to capture where objects are likely present, rather than where higher quality is needed (*e.g.,* large targeted objects will be selected by RPNs but they do not need high video quality to be accurately recognized). Using RPNs also limits the applications to object detection and does not generalize to other tasks such as semantic segmentation. In a broader context, DDS is related and complementary to the trend in deep learning of using *attention* mechanisms (*e.g.,* [55, 68])—attention improves DNN accuracy by focusing *computation* on the important regions, while DDS improves *bandwidth efficiency* by sending only a few regions in high quality to achieve the same DNN accuracy as if the whole video is sent in the highest quality.

We evaluate DDS and a range of recent solutions [73, 70, 17, 75, 49] on three vision tasks. Across 49 videos, we find DDS achieves same or higher accuracy while cutting bandwidth usage by upto 59%, or uses the same bandwidth consumption while increasing accuracy by 3-9%.

Furthermore, like other video streaming protocols, DDS must adapt its bandwidth usage to handle fluctuations in available bandwidth. There are several effective control knobs that affect the bandwidth usage of DDS. However, we empirically find that these knobs affect the bandwidth-accuracy tradeoff in a similar way, so DDS only tunes quality levels that are used to encode the video for the two iterations of DDS.

To tune the low and high quality levels, we implement a feedback control system (illustrated in Figure 2.2). Our controller is based on prior work that proposes a virtual, adaptive control system that can be customized for specific deployments [54, 12]. To instantiate this

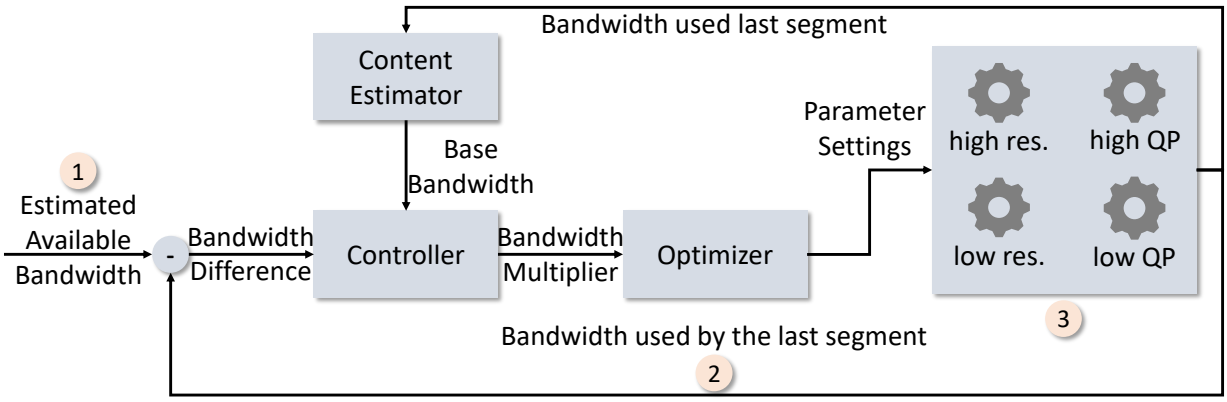


Figure 2.2: DDS's adaptive feedback control system dynamically tunes the low and high quality configurations based on the difference between the estimated available bandwidth for the next segment and that used for the previous segment.

controller, DDS needs to specify three things: a bandwidth constraint to be met, feedback for monitoring bandwidth usage, and the tunable parameters that affect bandwidth usage. For DDS, the bandwidth constraint is the estimated available bandwidth for the next *segment* (labelled (1) in the figure), the feedback is the total bandwidth usage (for both low and high quality) from the previous time segment (labelled (2) in the figure), and the tunable parameters are the resolution and quantization parameters (i.e. the QP in Figure 2.2) of both the low and high quality (labelled (3) in the figure). We define a segment as a fixed number of consecutive frames that are processed by DDS as a single batch. The controller continually estimates the *base* bandwidth usage; i.e. the previous segment's bandwidth usage if the *default* parameter settings had been used. The default parameter settings are defined as the settings that consume the least amount of bandwidth. The controller then takes this base behavior as well as the difference between the desired bandwidth constraint for the next segment and the achieved bandwidth usage for the previous segment and computes a scaling factor for the base bandwidth. This scaling factor is passed to an optimizer which finds the low and high quality settings that deliver the scaled bandwidth usage while maximizing F1 score.

DDS's dynamic bandwidth adaptation has several useful formal properties based on its

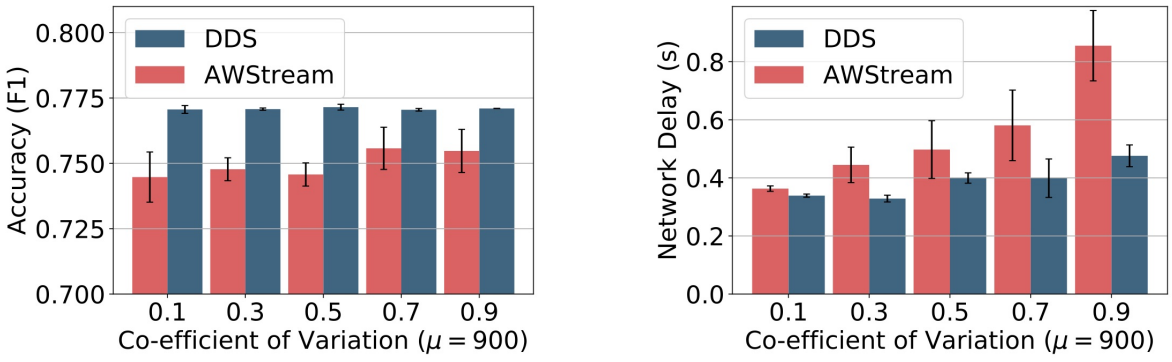
use of feedback control.

First, the content estimator can handle *dynamic* video content which changes the relationship between the parameters and bandwidth usage. The adaptation mechanism uses a Kalman Filter [71] to continually estimate the *base* bandwidth usage. Hence, when the video content changes, the control model—that captures the relationship between the parameters and bandwidth usage—will update itself, allowing DDS to capture unmodeled non-linearities in the relationship between quality settings and bandwidth use. Intuitively, we can think of the relationship between bandwidth usage and quality parameters as a curve and the base bandwidth (estimated by the Kalman filter) as a tangent to that curve. When adjusting the quality parameters, the DDS controller uses this tangent as a linear approximation to the true behavior. Using this formulation, the bandwidth usage converges to the bandwidth constraint in time proportional to the logarithm of the error in this estimation [54]. This adjustment technique provides robustness in the face of shifts and variations in the system including when there does not exist a single control model that captures all system dynamics [24].

Second, the optimizer finds the highest quality given the bandwidth usage specified by the controller. This optimality is achieved by scheduling configurations over multiple segments. As the system has a small, constant number of constraints (simply respecting the bandwidth requirement), an optimal solution can be found in constant time [45].

We evaluate DDS's ability to handle bandwidth variation against the state-of-the-art AWStream [73]. We compare the systems under both synthetic and real-world bandwidth traces.

For the synthetic trace, the available bandwidth of is drawn from a normal distribution of $900 \cdot N(1, \sigma^2)$Kbps while we increase σ from 0.1 to 0.9. Figure 2.3 shows the accuracy and network delay of DDS and AWStream under increasing bandwidth variance. DDS is able to maintain a higher accuracy than AWStream. Although DDS and AWStream



(a) Accuracy

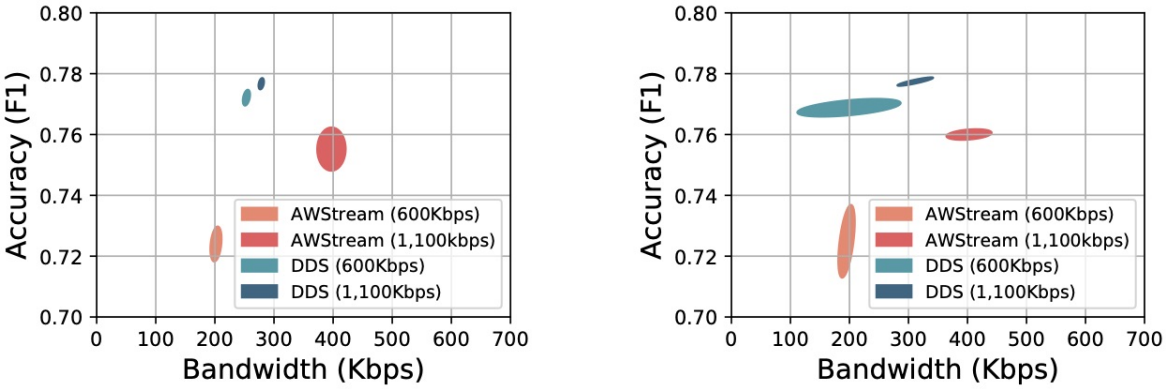
(b) Network Delay

Figure 2.3: DDS can handle bandwidth variance and maintain a sizeable gain over AWStream even under substantial bandwidth fluctuations.

use the same bandwidth estimation strategy (average of the last two segments), DDS uses the available bandwidth more efficiently because DDS's feedback control system continually adapts the model configuration parameters to bandwidth, allowing DDS to choose the best possible configuration parameters at each instant. Even under high variance DDS maintains a relatively low response delay while AWStream's delay increases significantly.

Next, we compare the performance of DDS and AWStream under two 4G/LTE networks traces [1]. However, the available bandwidth in the traces on average is greater than the bandwidth needed to stream the original video. Hence, we scale the available bandwidth by a constant factor to mimic settings where multiple cameras share the bottleneck bandwidth. In particular, we scale the average bandwidth of traces to 1,100kbps and 600kbps, while retaining the relative bandwidth variance in the trace. Figure 2.4 shows that DDS consistently achieves higher accuracy under both network traces because its adaptive control system is able to utilize the estimated available bandwidth more efficiently. Hence, DDS's adaptation mechanism highlights the advantages of principled adaptation over heuristic-based approaches.

While designing DDS's controller we realized that not only does DDS need to meet the bandwidth constraint but it also needs to update the bandwidth constraint itself as the



(a) Network Trace 1

(b) Network Trace 2

Figure 2.4: Performance of DDS and AWStream under real network traces

available bandwidth changes during execution. Hence, DDS needs to update the constraint of the controller after the controller has been initialized. After implementing this capability for DDS, we realized that most real-world systems have similar requirements. And while prior work has proposed *adaptation frameworks* to add adaptation to existing systems, to the best of our knowledge, no existing framework allows the system to interact with any aspect of adaptation during execution. Hence, we recognized that there is a clear gap between adaptation requirements of modern computer systems and the capabilities existing solution.

In the following part of this body of work we take a closer look at the limitation of existing solution for adaptation and illustrate how these limitations lead to suboptimal behavior in several real-world systems. Finally, we propose a novel approach to address the limitations of existing solutions by providing support for general-purpose adaptation while allowing systems to interact with all aspects of adaptation as first-class programming objects. We illustrate the effectiveness of our proposed approach using a number of real-world systems and also examine how our proposed approach can be used to easily add adaptation in DDS.

CHAPTER 3

GOAL: GOAL-ORIENTED ADAPTATION LANGUAGE

Logical correctness is no longer the only guarantee that modern software systems are expected to provide. Increasingly, these systems—from mobile to cloud—must meet quality-of-service *goals*, expressed as constrains and objectives on *metrics*; *e.g.*, request latency, energy consumption and result accuracy [64, 8, 31, 53, 16]. While developing software to meet such diverse requirements is already a challenge, the problem is further complicated by the fact that all of these quantifiable metrics are affected not just by the application code, but also by factors beyond program's control like changing operating environment, input workload, and user needs. Successful system deployment now requires applications that are both logically correct and capable of *adapting* to meet a specified goal to deliver predictable quantifiable behavior even in unpredictable dynamic environments [47, 44].

Implementing adaptive systems requires an *adaptation logic* (AdaptLog) that can efficiently—at runtime—convert observed metrics into knob settings that meet the goals [72, 50, 65, 67, 35]. However, implementing a reliable and robust AdaptLog is difficult because it requires software developers to be experts in their specific domains along with a wide range of factors that could impact the system's ability to meet the specified goals. Furthermore, it requires the developers to acquire specialized knowledge of the underlying principles used to implement the AdaptLog, *e.g.,* control theory, machine learning etc.

Existing systems employ either a *heuristic* or a *declarative* pattern to implement their adaptation.

In heuristic approaches (Figure 3.1a), the AdaptLog is written using existing language constructs and inter-mixed with application code that provides core functional behavior. Heuristics have the benefit of putting the application in full control of all adaptation; *i.e.,* they do not rely on some independently developed module to produce the correct behavior. However, designing robust heuristics is quite difficult and the heuristics are not easy to port

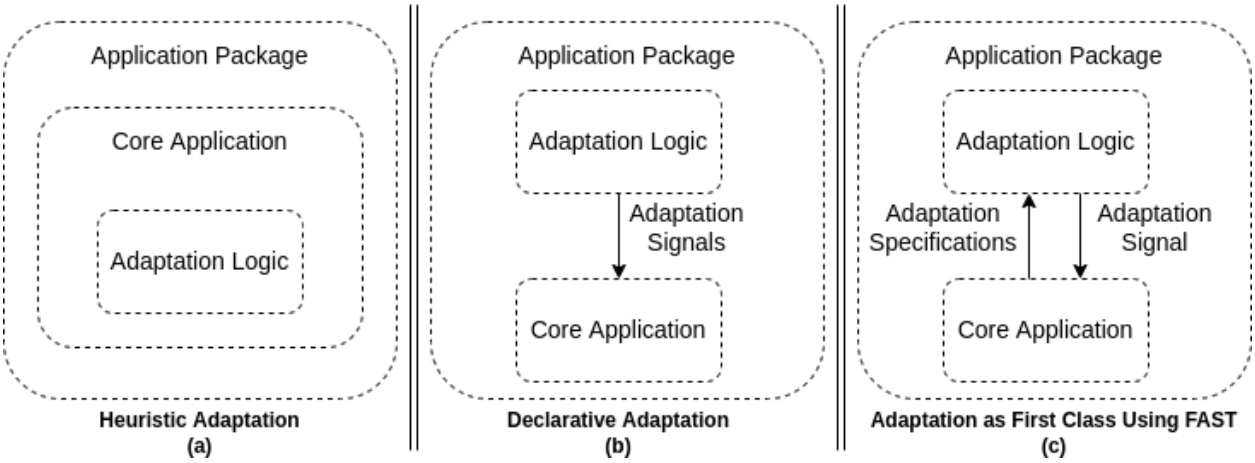


Figure 3.1: Types Of Adaptation.

to other systems [47, 44, 59, 27, 62]. For example, when the hardware for the Samsung Galaxy S9 was upgraded for the S9+, the achieved performance and energy efficiency was worse despite the better hardware [30]. The problem was tracked down to misconfigurations in the AdaptLog of the HMP scheduler [29]. Furthermore, heuristic adaptation makes software unwieldy and harder to maintain as embedding adaptation logic in the core program violates separation of concerns by mixing core functionality with code for adapting to deliver quantifiable behavior. Additionally, even minor changes in the application code can significantly impact the efficacy of the heuristics.

In declarative approaches (Figure 3.1b), developers use an existing *adaptation framework*, packaged in the form of libraries or language runtimes [14, 60, 76, 43, 46, 15, 23, 40, 39, 32, 21, 20], to add adaptation to their applications. Developers use the frameworks' interface to instantiate the packaged AdaptLog and provide an *adaptation specification* (AdaptSpec): a declaration of the system's goals and the knobs that can be configured to achieve it. The framework's internal AdaptLog then tunes the knobs in response to any runtime changes. The instantiated AdaptLog operates outside of the core program functionality, allowing developers to focus on their application domain and delegate adaptation. Recent proposals package a machine learning [16, 13, 41, 48, 69] or control theory [26, 37, 38, 32, 74, 28]

based AdaptLog that is instantiated by the larger software system. In contrast to heuristic approaches, declarative approaches are quite robust and achieve good separation of concerns.

Unfortunately, all existing declerative frameworks focus on a narrow, predefined set of possible AdaptSpecs (i.e., one or two metrics and a small collection of knobs). For example, Eon [63] only supports accuracy and energy metrics using alternative method implementations as knobs. Similarly, PowerDial [38] only supports accuracy and throughput tradeoffs using application-level parameters as knobs.

This lack of generality arises because prior adaptation frameworks develop their AdaptLog using specialized *models* that relate specific metrics to specific knobs. Whether the AdaptLog is based on machine learning, control theory, or heuristics, the model is essential to predict how metrics will change in response to changes in the knob configurations, which then guides the AdaptLog to set the knobs to ensure the goals are met. However, because the model relates specific knobs to specific metrics, the relevant knobs and metrics need to be enumerated before the model is constructed and that model must be reconstructed for use with a different knobs and/or metrics. This reliance on a narrowly defined model makes it difficult to implement a *general* adaptive system that can be deployed with different goals in different environments.

The use of fixed models also prevents a system from *dynamically* changing AdaptSpecs during execution. We define the runtime alteration of an AdaptSpec as *meta-adaptation*. For many applications it is not enough to just adjust knob configurations; meta-adaptation is necessary as the goals themselves must be changed in response to external conditions [34, 58]. For example, consider a CCTV camera installed with a backup battery [61]. It must always meet a target frame rate to prevent data loss, but its other goals vary depending on power source. On line power the system must maximize quality, however, during a power outage, it must minimize energy to prolong battery life [2]. Running this system in either AdaptSpec for its entire execution is suboptimal, either wasting energy on battery or lowering quality

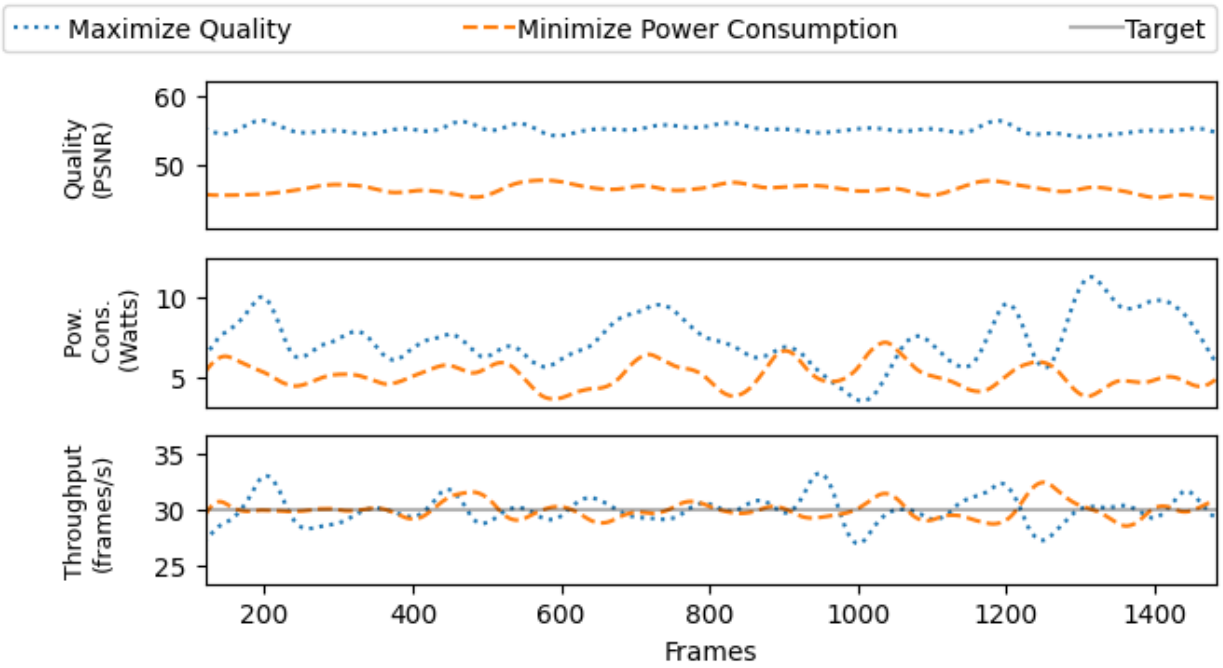


Figure 3.2: Existing declarative approaches result in suboptimal behavior.

on line power. Figure 3.2 shows the execution of the CCTV system using both of the goal. During both executions the system is able to meet its throughput constraint of 30 frames/sec. However, not surprisingly, the AdaptSpec that maximizes quality improves peak signal-to-noise-ratio (to 56 dB) at a cost of an average increase of 2.2 Watts, or roughly 1.5x increase in power consumption. On the other hand, the AdaptSpec that minimizes power consumption degrades quality unnecessarily by roughly 1.37x.

Developers add adhoc support for meta-adaptation using existing declarative approaches by embedding several different adaptation frameworks in their application. During execution, the system then switches between the AdaptLogs by destroying one and initializing another to meet a different goal or tune a different set of parameters. This makes the development of adaptive systems more difficult and requires the developer to enumerate all adaptation requirements at the time of development. Furthermore, having to debug and maintain code that uses interfaces of several different adaptation frameworks puts significant burden on developers.

Name	Platform	# App. Knobs	# Sys. Knobs	Initial Objective	Initial Constraint	Change after Meta-adaptation
CCTV Camera [37]	Embedded	3	2	max(quality)	thruput == 30	min(power)
Video Object Detector [51]	Embedded	4	2	min(power)	thruput == 20	Restrict QP
Service Oriented Architecture (SOA) [26]	Server	3	N/A	max(reliability)	latency == 0.5	min(cost), rel == 0.6
Synthetic Aperture Radar (SAR) [66]	Embedded	4	2	max(quality)	thruput == 80	max(thruput), quality == 0.7
AES Encryption [33]	Embedded	1	2	max(thruput)	power == 1.5	max(thruput / power), block size == 256
Search Engine [11, 38]	Server	1	2	min(power)	thruput == 18.0	Restrict searched doc.
Optical Character Recognition (OCR) [5]	Server	1	2	min(power)	thruput == 8.0	max(quality), thruput == 8.0.

Table 3.1: Properties of applications used to evaluate GOAL.

To support more general and dynamic adaptation—including meta-adaptation—this work presents GOAL: Goal-Oriented Adaptation Language. GOAL provides novel adaptation framework implemented in Swift [10]. Its key components are its (1) runtime AdaptLog and (2) its interface for writing AdaptSpecs.

Central to GOAL's design is its AdaptLog, which takes the form of a *virtualized, time-variant, adaptive* control system. Unlike prior approaches, GOAL's *virtual* control system is independent of any specific model relating metrics to knobs; instead, it is parameterized by a model which is passed in at runtime. Furthermore, GOAL's controller continually adjusts itself at runtime while also carefully exploiting structure of optimization problems so that it can control non-linear systems with a series of linear approximations.

GOAL's AdaptSpecs are written using a novel domain specific language (DSL), which is compiled just-in-time (JIT), separating the AdaptSpec declaration from system implementation. This separation allows different AdaptSpecs to be used with the same binary for deployments with different requirements or even for changing the requirements while the system is running. GOAL also provides a Library API so users can declare knobs and metrics and alter these values during execution. These features support complex adaptive behavior that would have been difficult and inefficient to implement with existing frameworks.

To demonstrate GOAL's effectiveness, we re-implement seven adaptive applications from the literature. Collectively, these case studies cover a wide range of metrics (throughput, latency, accuracy, power, cost, reliability and efficiency) and knobs (at both the application and system level, including two different hardware systems with distinct knobs). Our re-

sults show that GOAL's generalized approach meets goals just as well as prior work that synthesizes AdaptLogs specifically for each application's narrow goals and knobs [25]. To highlight GOAL's benefits, we then modify each application to perform meta-adaptation. Table 3.1 outlines the implemented applications and the adaptation and meta-adaptation requirements of each application. We observe that due to GOAL's ability to support a wide range of AdaptSpecs and meta-adaptation, GOAL-based applications exhibit a 1.69× average improvement in corresponding metrics after meta-adaptation is performed, compared to prior approaches that cannot support meta-adaptation. Furthermore, we show that GOAL incurs negligible overhead and is robust to errors in profiling, changing workloads and operating conditions.

This work makes the following contributions:

- Motivates the benefits of support general purpose adaptation and meta-adaptation.
- Proposes a general adaptation logic and runtime that supports a wide range of knobs, metrics and goals.
- Proposes a programming framework and DSL for writing adaptation specifications.
- Implements GOAL and releases it as open source.

As mentioned earlier, control theoretic AdaptLogs provide robust performance while meeting a higher level user-specified goal. However, using control theory for adaptation poses three challenges: (1) AdaptLogs are difficult to design and implement, (2) AdaptLogs are often developed for a narrow set of knobs and metrics, and (3) such AdaptLogs provide incorrect behavior when operating in parallel with other AdaptLogs. GOAL provides solutions to the first two challenges. However, GOAL does not address the third challenge. The next part of this body of research aims to extend GOAL to tackle the challenge of coordinating AdaptLogs so that they meet their respective goals.

CHAPTER 4

WASL: COORDINATING CONTROLLERS

Control theoretic AdaptLogs allows systems to self-adapt to deal with unpredictable changes in the operating conditions and workloads. Control theory provides a framework for formally reasoning about the controlled system's behavior. With the rise in usage of control theory for adaptation a new challenge has emerged: how to ensure that multiple colocated adaptive *modules* achieve their goals.

Prior approaches have been suggested to allow multiple modules with non-competing goals to be operate correctly on a system [36]. However, synchronizing behavior between two modules with competing goals is an open and challenging research problem. As an example, consider a search engine running on a server machine. The application module needs to meet a target energy per query while maximizing the quality of search results while the system module is expected to meet the same energy per query target while maximizing performance of the system. This is a concrete example of two modules with competing goals. Decisions taken by one module impair the other modules ability to meet its goals. Hence, it is obvious that they cannot be allowed to operate in an uncoordinated manner.

Figure 4.1 shows the execution of the aforementioned multi-module search engine application. The application and the system modules are required to meet a target 0.5 Joules energy per iteration while maximizing quality and maximizing performance respectively. The figure shows that when only the system module and only the application module are operating they are able to meet their goal quite well. They are able to meet their goal with a MAPE of roughly 7%. However, when they are both executing in parallel, the MAPE of the entire system increases by roughly 4×. This shows that both modules cannot be allowed to execute in parallel in an uncoordinated manner.

Existing research has suggested synthesizing a monolithic controller that actuates knobs for modules colocated on the system [46, 52, 57]. However, this makes such a controller

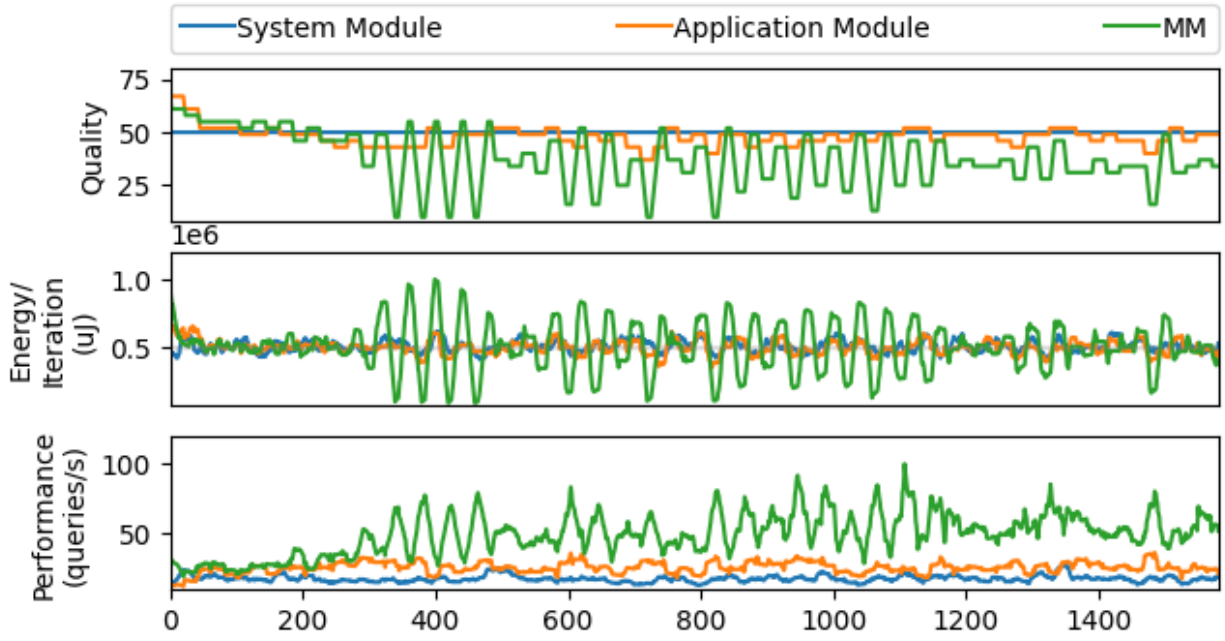


Figure 4.1: Multi-Module Search Engine.

difficult to design, maintain and port to other systems. Another approach is to assign priorities to controllers. This means that modules with the lowest priority sacrifice their goal satisfaction for modules with higher priorities. This can lead to needlessly sacrificing on the goal of lower priority agents if the goal is feasible. However, the problem at the heart of these solutions is that all of these approaches require conflicts to be resolved at design time making the design and implementation of multi-module adaptive systems difficult. Furthermore, dynamically changing the modules or the requirements of a single module becomes very difficult.

In this work, we propose a novel approach for online coordination of multiple adaptive modules. Using our approach, individual modules share their models, knobs, goals and gain with a coordination server. The server then negotiates contracts between individual modules ensuring that decisions taken by one module do not negatively impact the other module. The key insight behind the coordination is that the control actions of the module that can impact the constrained metric more drastically (module A) should be curtailed or

slowed down by reducing the gain appropriately. This allows the other module (module B) to view changes made by the other controller as a gradual change in the operating conditions which allows it to actuate knobs accordingly to continue meeting its goals. Hence, when both modules are operating module B views the changes made by module A as a gradual change in its operating point, while module A views changes made by module A as disturbances. Our approach allows all modules to be developed independently, because all conflicts are resolved at runtime. The only requirement for the modules is that the underlying adaptation algorithm should be a model-based closed-loop feedback controller. This provides a number of common properties that are use by the coordination server during the negotiation process. Additionally, since our approach resolves conflicts at runtime, the user can change the number of operating modules and their goals dynamically.

CHAPTER 5

REMAINING WORK AND TIMELINE

The WASL project still requires a significant amount of work. There are several questions that need to be answered for the core algorithm. During the rest of this and the following academic year we hope to be able to answer those questions and hopefully address any related challenges. During the first phase of evaluation, we aim to show the efficacy of our system on the case-studies that were used for GOAL's evaluation.

To further motivate the use of our system we will identify benchmarks from recently published relevant works. Our current aim is to predominantly consider embedded application benchmarks. For example, sensing applications would be great candidates for our system because they need to manage performance, accuracy and energy tradeoffs. On such systems, the sensing application will have goals that are competing with the goal of the underlying system. Hence, these would be straightforward use-cases for our proposed approach. We have already identified OpenDataCam [56] as one such application. Furthermore, we aim to extend the individual sensor node case-study to sensor networks in which our system will be used to coordinate nodes to meet node level goals while meeting a global network level goal.

While we will predominantly focus on embedded applications, we will also evaluate our approach for server applications. For example, we aim to employ our approach for pipelined systems that provide ML-as-a-service.

By evaluating our system not only for different applications but for different architectures as well, we hope to be able to show the generality of our system which is a core part of this body of research.

In accordance with the work still left to be done for the project we propose the following timeline:

Goal	Expected Completion
Finalize controller prediction accuracy metric	Feb 2022
Finalize gain scheduling scheme	Feb - March 2022
Test system on GOAL case-studies	March 2022
Implement Server & API	March 2022
Find benchmarks from recent related work	Feb - May 2022
Implement benchmarks from related work	March - May 2022
Collect additional results	May 2022
Iterate on Design of System	May - June 2022
Finalized Results	June 2022
Away for internship	June - Sep 2022
Preparing Paper	Sep - Oct 2022
Submit Paper	Autumn - Winter 2022/3
Prepare Thesis Document	Jan 2023
Defense	Feb 2023
Formally File Thesis	Apr 2023

Table 5.1: Expected Timeline

REFERENCES

[1] 4g/lte bandwidth logs. http://users.ugent.be/~jvdrhoof/dataset-4g/.

[2] Arlo: Wire-free hd and hdr smart home security cameras.

[3] Can 30,000 cameras help solve chicago's crime problem? https://www.nytimes.com/2018/05/26/us/chicago-police-surveillance.html.

[4] Dashjs. https://github.com/Dash-Industry-Forum/dash.js.

[5] Tesseract ocr.

[6] Video surveillance: How technology and the cloud is disrupting the market. https://cdn.ihs.com/www/pdf/IHS-Markit-Technology-Video-surveillance.pdf.

[7] Wi-fi vs. cellular: Which is better for iot? https://www.verypossible.com/blog/wi-fi-vs-cellular-which-is-better-for-iot.

[8] Mai Abusair, Antinisca Di Marco, and Paola Inverardi. Context-aware adaptation of mobile applications driven by software quality and user satisfaction. In *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 31–38, July 2017.

[9] Ganesh Ananthanarayanan, Victor Bahl, Peter Bodik, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath Sivalingam, and Sudipta Sinha. Real-time video analytics - the killer app for edge computing. *IEEE Computer*, October 2017.

[10] Apple. Swift. https://developer.apple.com/swift/, 2019.

[11] Woongki Baek and Trishul M. Chilimbi. Green: A framework for supporting energy-conscious programming using controlled approximation. In *Proceedings of the 31st ACM Conference on Programming Language Design and Implementation*, PLDI '10, pages 198–209, New York, NY, USA, 2010. ACM.

[12] S. Barati, F. A. Bartha, S. Biswas, R. Cartwright, A. Duracz, D. Fussell, H. Hoffmann, C. Imes, J. Miller, N. Mishra, Arvind, D. Nguyen, K. V. Palem, Y. Pei, K. Pingali, R. Sai, A. Wright, Y. Yang, and S. Zhang. Proteus: Language and runtime support for self-adaptive software development. *IEEE Software*, 36(2):73–82, March 2019.

[13] Josep Ll. Berral, Íñigo Goiri, Ramón Nou, Ferran Julià, Jordi Guitart, Ricard Gavaldà, and Jordi Torres. Towards energy-aware scheduling in data centers using machine learning. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 215–224, New York, NY, USA, 2010. ACM.

[14] James Bornholt, Todd Mytkowicz, and Kathryn S. McKinley. Uncertain¡t¿: A first-order type for uncertain data. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, page 51–66, New York, NY, USA, 2014. Association for Computing Machinery.

[15] Anthony Canino and Yu David Liu. Proactive and adaptive energy-aware programming with mixed typechecking. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2017, pages 217–232, New York, NY, USA, 2017. ACM.

[16] Anthony Canino, Yu David Liu, and Hidehiko Masuhara. Stochastic energy optimization for mobile gps applications. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, pages 703–713, New York, NY, USA, 2018. ACM.

[17] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168. ACM, 2015.

[18] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. Adascale: Towards real-time video object detection using adaptive scaling. *arXiv preprint arXiv:1902.02910*, 2019.

[19] Sandeep P Chinchali, Eyal Cidon, Evgenya Pergament, Tianshu Chu, and Sachin Katti. Neural networks meet physical networks: Distributed inference between edge devices and the cloud. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 50–56. ACM, 2018.

[20] Christina Delimitrou and Christos Kozyrakis. Paragon: Qos-aware scheduling for heterogeneous datacenters. *SIGPLAN Not.*, 48(4):77–88, March 2013.

[21] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-efficient and qos-aware cluster management. *SIGPLAN Not.*, 49(4):127–144, February 2014.

[22] John Emmons, Sadjad Fouladi, Ganesh Ananthanarayanan, Shivaram Venkataraman, Silvio Savarese, and Keith Winstein. Cracking open the dnn black-box: Video analytics with dnns across the camera-cloud boundary. In *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, pages 27–32, 2019.

[23] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Architecture support for disciplined approximate programming. *SIGARCH Comput. Archit. News*, 40(1):301–312, March 2012.

[24] Antonio Filieri, Henry Hoffmann, and Martina Maggio. Automated design of self-adaptive software with control-theoretical formal guarantees. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 299–310, New York, NY, USA, 2014. ACM.

[25] Antonio Filieri, Henry Hoffmann, and Martina Maggio. Automated design of self-adaptive software with control-theoretical formal guarantees. In *Proceedings of the 36th*

International Conference on Software Engineering, ICSE 2014, pages 299–310, New York, NY, USA, 2014. ACM.

[26] Antonio Filieri, Henry Hoffmann, and Martina Maggio. Automated multi-objective control for self-adaptive software design. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 13–24, New York, NY, USA, 2015. ACM.

[27] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolás D'Ippolito, Ilias Gerostathopoulos, Andreas B. Hempel, Henry Hoffmann, Pooyan Jamshidi, Evangelia Kalyvianaki, Cristian Klein, Filip Krikava, Sasa Misailovic, Alessandro Vittorio Papadopoulos, Suprio Ray, Amir Molzam Sharifloo, Stepan Shevtsov, Mateusz Ujma, and Thomas Vogel. Software engineering meets control theory. In *10th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015, Florence, Italy, May 18-19, 2015*, pages 71–82, 2015.

[28] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolás D'Ippolito, Ilias Gerostathopoulos, Andreas Berndt Hempel, Henry Hoffmann, Pooyan Jamshidi, Evangelia Kalyvianaki, Cristian Klein, Filip Krikava, Sasa Misailovic, Alessandro Vittorio Papadopoulos, Suprio Ray, Amir Molzam Sharifloo, Stepan Shevtsov, Mateusz Ujma, and Thomas Vogel. Control strategies for self-adaptive software systems. *TAAS*, 11(4), 2017.

[29] Andrei Frumusanu. Improving the exynos 9810 galaxy s9: Part 1. https://www.anandtech.com/show/12615/improving-exynos-9810-galaxy-s9-part-1, 05 2018.

[30] Andrei Frumusanu. The samsung galaxy s9 and s9+ review: Exynos and snapdragon at 960fps. https://www.anandtech.com/show/12520/the-galaxy-s9-review/5, 03 2018.

[31] Alessio Gambi, Mauro Pezzè, and Giovanni Toffetti. Kriging-based self-adaptive cloud controllers. *IEEE Transactions on Services Computing*, 9(3):368–381, May 2016.

[32] Ashvin Goel, David Steere, Calton Pu, and Jonathan Walpole. Swift: A feedback control and dynamic reconfiguration toolkit. Technical report, 1998.

[33] J. Goodman, A. P. Dancy, and A. P. Chandrakasan. An energy/security scalable encryption processor using an embedded variable voltage dc/dc converter. *IEEE Journal of Solid-State Circuits*, 33(11):1799–1809, Nov 1998.

[34] Google. Android Power Management: Battery Saver. `https://developer.android.com/about/versions/pie/power#battery-saver`, 2020.

[35] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving dnns like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 443–462. USENIX Association, November 2020.

[36] Henry Hoffmann. Coadapt: Predictable behavior for accuracy-aware applications running on power-aware systems. In *2014 26th Euromicro Conference on Real-Time Systems*, pages 223–232, 2014.

[37] Henry Hoffmann. Jouleguard: Energy guarantees for approximate applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 198–214, New York, NY, USA, 2015. ACM.

[38] Henry Hoffmann, Stelios Sidiroglou, Michael Carbin, Sasa Misailovic, Anant Agarwal, and Martin Rinard. Dynamic knobs for responsive power-aware computing. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, page 199–212, New York, NY, USA, 2011. Association for Computing Machinery.

[39] C. Imes, D. H. K. Kim, M. Maggio, and H. Hoffmann. Poet: a portable approach to minimizing energy under soft real-time constraints. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 75–86, April 2015.

[40] Connor Imes and Henry Hoffmann. Bard: A unified framework for managing soft timing and power constraints. In *Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), 2016 International Conference on*, pages 31–38. IEEE, 2016.

[41] Connor Imes, Steven Hofmeyr, and Henry Hoffmann. Energy-efficient application resource scheduling using machine learning classifiers. In *Proceedings of the 47th International Conference on Parallel Processing*, ICPP 2018, pages 45:1–45:11, New York, NY, USA, 2018. ACM.

[42] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, 2017.

[43] Aman Kansal, Scott Saponas, A.J. Bernheim Brush, Kathryn S. McKinley, Todd Mytkowicz, and Ryder Ziola. The latency, accuracy, and battery (lab) abstraction: Programmer productivity and energy efficiency for continuous mobile context sensing. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*, OOPSLA '13, pages 661–676, New York, NY, USA, 2013. ACM.

[44] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, Jan 2003.

[45] D. H. K. Kim, C. Imes, and H. Hoffmann. Racing and pacing to idle: Theoretical and empirical analysis of energy optimization heuristics. In *2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications*, pages 78–85, 2015.

[46] Minyoung Kim, Mark-Oliver Stehr, Carolyn Talcott, Nikil Dutt, and Nalini Venkatasubramanian. Xtune: A formal methodology for cross-layer tuning of mobile embedded systems. *ACM Trans. Embed. Comput. Syst.*, 11(4), January 2013.

[47] Robert Laddaga. Guest editor's introduction: Creating robust software through self-adaptation. *IEEE Intelligent Systems*, 14(3):26–29, May 1999.

[48] X. Lin, Y. Wang, and M. Pedram. A reinforcement learning-based power management framework for green computing data centers. In *2016 IEEE International Conference on Cloud Engineering (IC2E)*, pages 135–138, April 2016.

[49] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[50] Kiwan Maeng and Brandon Lucia. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 129–144, Carlsbad, CA, October 2018. USENIX Association.

[51] M. Maggio, A. V. Papadopoulos, A. Filieri, and H. Hoffmann. Self-adaptive video encoder: Comparison of multiple adaptation strategies made simple. In *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 123–128, May 2017.

[52] Martina Maggio, Alessandro Vittorio Papadopoulos, Antonio Filieri, and Henry Hoffmann. Automated control of multiple software goals using multiple actuators. In *Proceedings of the 12th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, ESEC/FSE 2017. ACM, 2017.

[53] Sasa Misailovic, Michael Carbin, Sara Achour, Zichao Qi, and Martin C. Rinard. Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels. In *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*, OOPSLA '14, pages 309–328, New York, NY, USA, 2014. ACM.

[54] Nikita Mishra, Connor Imes, John D. Lafferty, and Henry Hoffmann. Caloree: Learning control for predictable latency and low energy. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '18, page 184–198, New York, NY, USA, 2018. Association for Computing Machinery.

[55] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[56] Opendatacam. Opendatacam/opendatacam: An open source tool to quantify the world.

[57] Raghavendra Pradyumna Pothukuchi, Sweta Yamini Pothukuchi, Petros Voulgaris, and Josep Torrellas. Yukta: Multilayer resource controllers to maximize efficiency. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 505–518, 2018.

[58] Redhat. Tuned: Tuning Profile Delivery Mechanism for Linux. `https://tuned-project.org/`, 2020.

[59] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42, May 2009.

[60] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. Enerj: Approximate data types for safe and general low-power computation. In *Proceedings of the 32Nd ACM Conference on Programming Language*

Design and Implementation, PLDI '11, pages 164–174, New York, NY, USA, 2011. ACM.

[61] Homeland Security. Cctv technology handbook. Online Documen, 2013.

[62] Vítor E. Silva Souza, Alexei Lapouchnian, and John Mylopoulos. System identification for adaptive software systems: A requirements engineering perspective. In Manfred Jeusfeld, Lois Delcambre, and Tok-Wang Ling, editors, *Conceptual Modeling – ER 2011*, pages 346–361, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[63] Jacob Sorber, Alexander Kostadinov, Matthew Garber, Matthew Brennan, Mark D. Corner, and Emery D. Berger. Eon: A language and runtime system for perpetual systems. In *In Proceedings of The Fifth International ACM Conference on Embedded Networked Sensor Systems (SenSys '07), Syndey*, 2007.

[64] Mbarka Soualhia, Foutse Khomh, and Sofiène Tahar. Atlas: An adaptive failure-aware scheduler for hadoop. In *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, Dec 2015.

[65] Akshitha Sriraman and Thomas F. Wenisch. µtune: Auto-tuned threading for OLDI microservices. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 177–194, Carlsbad, CA, October 2018. USENIX Association.

[66] Xin Sui, Andrew Lenharth, Donald S. Fussell, and Keshav Pingali. Proactive control of approximate programs. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '16, Atlanta, GA, USA, April 2-6, 2016*, pages 607–621, 2016.

[67] Chunqiang Tang, Kenny Yu, Kaushik Veeraraghavan, Jonathan Kaldor, Scott Michelson, Thawan Kooburat, Aravind Anbudurai, Matthew Clark, Kabir Gogia, Long

Cheng, Ben Christensen, Alex Gartrell, Maxim Khutornenko, Sachin Kulkarni, Marcin Pawlowski, Tuomas Pelkonen, Andre Rodrigues, Rounak Tibrewal, Vaishnavi Venkatesan, and Peter Zhang. Twine: A unified cluster management system for shared infrastructure. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 787–803. USENIX Association, November 2020.

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[69] Yanzhi Wang and Massoud Pedram. Model-free reinforcement learning and bayesian classification in system-level power management. *IEEE Transactions on Computers*, 65(12):3713–3726, Dec 2016.

[70] Yiding Wang, Weiyan Wang, Junxue Zhang, Junchen Jiang, and Kai Chen. Bridging the edge-cloud barrier for real-time advanced vision analytics. In *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, 2019.

[71] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, USA, 1995.

[72] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. Neural adaptive content-aware internet video delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 645–661, Carlsbad, CA, October 2018. USENIX Association.

[73] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A Lee. Awstream: Adaptive wide-area streaming analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 236–252. ACM, 2018.

[74] Ronghua Zhang, Chenyang Lu, Tarek F. Abdelzaher, and John A. Stankovic. Controlware: a middleware architecture for feedback control of software performance. In *Proceedings 22nd International Conference on Distributed Computing Systems*, pages 301–310, July 2002.

[75] Tan Zhang, Aakanksha Chowdhery, Paramvir Victor Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 426–438. ACM, 2015.

[76] Yuhao Zhu and Vijay Janapa Reddi. Greenweb: Language extensions for energy-efficient mobile web computing. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, pages 145–160, New York, NY, USA, 2016. ACM.