Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Workload Prediction in P4 Programmable Switches: Smart Resource Scheduling

BOYANG YAN, DEEPSHIKA KOVVALI, ANUSHA AKKIREDDY[1]
[1]NC State University (e-mail: byan4@ncsu.edu; dgkovval@ncsu.edu; aakkire@ncsu.edu)

ABSTRACT The rapid expansion of cloud services and their unpredictable workload demands present significant challenges in resource management. Traditional resource management approaches, primarily based on static rules and thresholds, often fail to ensure cost-effectiveness and optimal resource utilization. This research introduces a predictive model designed to forecast traffic demand, aiming to shift from a reactive to a proactive resource management approach. By integrating advanced predictive analytics with the capabilities of P4 programmable switches, this study seeks to enhance the efficiency of resource utilization and improve system robustness. The goal is to equip organizations with the agility and economic efficiency required to navigate the complexities of dynamic cloud environments effectively. This approach not only promises to refine microservice resource allocation but also supports the broader objective of fostering more resilient and efficient cloud infrastructures.

INDEX TERMS Smart Scheduling, Resource Management, Workload Predictions, Time Series Analysis

I. INTRODUCTION

The rapid proliferation of cloud computing has significantly altered the landscape of software development and deployment, presenting both new opportunities and complex challenges. As enterprises increasingly adopt cloud services to leverage the benefits of scalability, flexibility, and cost-effectiveness, the need for advanced resource management strategies becomes paramount. Traditional approaches, which largely depend on static allocation rules and simplistic threshold-based scaling, are proving inadequate in handling the dynamic and often unpredictable nature of cloud workloads. These conventional methods often lead to either resource over-provisioning, resulting in wasteful expenditure, or under-provisioning, which can compromise service quality and user experience. In response to these challenges, the field of cloud computing has seen a paradigm shift towards more agile and adaptive resource management techniques. Microservices architecture, a key component of modern cloud applications, exemplifies this shift. It involves decomposing applications into smaller, independently scalable services, each running in its own process and communicating with lightweight mechanisms, such as HTTP resource APIs. This architectural style enhances agility and speeds up deployment but also introduces significant complexity in resource management. Each microservice may scale differently based on its specific workload, requiring a nuanced approach to resource allocation that can anticipate and react to changes in demand in real-time. Predictive resource scheduling emerges as a crucial strategy in this context. By employing advanced algorithms that can forecast future demands based on historical data, cloud providers can dynamically allocate resources in a manner that optimizes cost and performance. This research focuses on developing a predictive model that not only addresses the inefficiencies of traditional resource management approaches but also leverages the unique capabilities of P4 programmable switches. These switches allow for highly flexible and programmable network behavior, which is critical in managing the high-speed data flows typical of modern data centers. The advent of smart scheduling techniques, which incorporate real-time data and predictive analytics, represents a significant advancement in cloud resource management. These techniques enable data centers to operate more efficiently, reduce operational costs, and improve the overall quality of service by minimizing latency and avoiding service disruptions. Our study is situated at this innovative juncture, aiming to harness the potential of P4 programmable switches to revolutionize resource scheduling in cloud environments. By integrating smart scheduling with predictive analytics, this research contributes to the ongoing evolution of cloud computing, pushing the boundaries of

what is possible in dynamic and complex cloud environments.

A. RESEARCH QUESTIONS

Predictive workload management and resource scheduling are increasingly critical in cloud computing, particularly as demand for cloud services grows and the architecture of applications becomes more complex. The literature on this subject is rich with various approaches aimed at enhancing the efficiency and responsiveness of cloud resources through predictive analytics and intelligent scheduling.

1) Historical Context and Evolution

Initially, research in this field focused on static resource allocation strategies that did not account for the fluctuating nature of cloud workloads. Early works by Calheiros et al. (2011) and Beloglazov et al. (2012) laid the groundwork by demonstrating the limitations of static resource allocation and the potential benefits of adopting dynamic resource provisioning based on real-time data analysis [1], [2]. These studies underscored the need for more adaptive resource management techniques that could respond to changes in workload demand without human intervention.

2) Advances in Predictive Modeling

As the field evolved, more sophisticated predictive models were introduced. Techniques such as time series analysis, machine learning, and neural networks began to be applied to forecast future demands in cloud environments. For instance, Lorido-Botrán et al. (2014) explored the use of autoregressive integrated moving average (ARIMA) models for predicting the workload in cloud systems, paving the way for more advanced predictive algorithms [3]. The introduction of machine learning techniques for workload prediction has been particularly transformative. Researchers like Tordsson et al. (2012) began employing various forms of regression analysis, support vector machines, and neural networks to predict the workload and thus optimize resource allocation [4]. These methods have proven effective in not only forecasting demand but also in reducing the operational costs associated with under or over-provisioning of resources.

3) Role of P4 Programmable Switches

The advent of P4 programmable switches has introduced new possibilities in network management and resource scheduling. The programmability of these switches allows for the implementation of complex algorithms directly into the network fabric, enabling real-time, adaptive control of data flows and resource allocation. This capability is particularly pertinent in high-traffic data center environments where traditional networking hardware struggles to keep pace with dynamic and high-speed requirements. Studies by Bosshart et al. (2014) and Kim et al. (2015) have explored the applications of P4 programmable switches in network traffic management and security [5], [6]. More recently, researchers have begun investigating their potential in facilitating smart scheduling and predictive resource management in cloud computing infrastructures. The flexibility of P4 switches to adapt to changing network conditions and their ability to process data packets with custom-defined rules makes them ideal for implementing predictive scheduling algorithms that require rapid adjustments to network behavior.

4) Emerging Trends and Current Research

The latest research in this area is focusing on integrating artificial intelligence (AI) with P4 programmable technologies to further enhance predictive capabilities. AI-driven approaches are being explored to handle even more complex scenarios involving multiple variables and highly dynamic environments. These studies aim to leverage the inherent strengths of P4 programmable switches to execute AI algorithms efficiently, ensuring that resource scheduling can be as responsive and proactive as possible. There are five open questions in this area. This project will only focus on scheduling for predictions of the VMs/Micro-services' expected resource utilization. In another ward, prediction dynamic workload demands.

- Smart scheduling. Before selecting servers to run a set of new VMs/Micro-services, the scheduler can contact the Real-time Controller for predictions of the VMs/Micro-services' expected resource utilization. With this information, the scheduler can select servers to balance the disk IOPS load or to reduce the likelihood of physical resource exhaustion in oversubscribed servers.
- Smart cluster selection. Before selecting a cluster in which to create a VM/Micro-services deployment, the cluster selection system can query the Real-time Controller for a prediction of maximum deployment size. With this information, this system can select a cluster that will likely have enough resources.
- Smart power oversubscription and capping. During a power emergency (when the power draw is about to exceed a circuit breaker limit), the power capping system can query the Real-time Controller for predictions of VM/Micro-services workload interactivity, before portioning the available power budget across servers. Ideally, VMs/Micro-services executing interactive workloads should receive all the power they may want, to the detriment of VMs/Micro-services running batch and background tasks. Alternatively, the VM/Micro-services scheduler can request interactivity predictions before selecting servers, so that interactive and delay-insensitive workloads are segregated in different sets of servers.
- Scheduling server maintenance. When a server starts to misbehave, the health monitoring system can query the Real-time Controller for the expected lifetime of the VMs/Micro-services running on the server. It can thus determine when maintenance can be scheduled, and whether VMs/Micro-services need to be live-migrated to enable maintenance without unavailability. In addition, the VM/Micro-services scheduler can use the

lifetime predictions to co-locate VMs/Micro-services that are likely to terminate roughly at the same time. This could facilitate other types of maintenance, such as OS updates.
- Recommending VM/Micro-services and deployment sizes. The cloud platform could provide a service to its customers that recommends the appropriate VM/Micro-services size and number of VMs/Micro-services at the time of each deployment. Using the Real-time Controller predictions of workload class and resource utilization, the service could recommend deployments where VMs/Micro-services predicted to be delay-insensitive would be more tightly sized than interactive VMs/Micro-services.

B. CONTRIBUTIONS

- Data center networks require high reliability and high-speed traffic (100Gbps), so our algorithms must be interpretable and cannot be black-box.
- We modify the decision tree (DT) to the Binary Decision Tree (BDT). Compared with the DT, our BDT:
  - Supports faster training.
  - Generates fewer rules.
  - Satisfies switch constraints better.
- Moreover, as learning algorithms (particularly deep learning) have shown their superior performance, some more complicated learning models are emerging for networking. However, their:
  - High computational complexity and
  - Large storage requirement

  are causing challenges in the deployment on switches.

C. STRUCTURE

This paper is organized as follows. We begin with a 'Background' section, which provides a comprehensive review of related work and a general overview of the field. This is followed by the 'Algorithms' section, where we describe the computational models and predictive algorithms employed in our research. Next, the 'Methods' section details the experimental setup and data preprocessing techniques. The 'Results' section presents the outcomes of our experiments, including data analysis and findings. Subsequent to this, the 'Discussion' section interprets the results, exploring the implications and limitations of our study. Finally, the 'Conclusion' section summarizes the key points of our research, offering insights and potential directions for future work.

II. BACKGROUND

In this section, we explore the architecture of microservices, followed by an examination of the P4 switch and its constraints. We then delve into a comparative analysis of eBPF and P4 switch technologies, highlighting the distinct features and operational differences between the two. This comparison is crucial for understanding the specific advantages that P4 switches offer in network management and resource scheduling within cloud computing environments. Through this discussion, we aim to clarify the suitability of each technology in various scenarios and their implications for modern network architectures.

A. MICROSERVICES ARCHITECTURE

Microservices architecture diverges from the traditional monolithic design paradigm, where all functional components of an application are interwoven into a single indivisible unit [7]. By contrast, microservices encapsulate discrete functionalities into independently deployable modules, facilitating modular development, deployment, and scaling. This architectural refinement aligns well with cloud computing platforms that excel in offering elastic computing resources, thereby enabling services to be scaled in or out based on real-time demand.

The crux of managing a microservices-based application lies in the effective orchestration of service replicas to ensure seamless scalability and resilience. Horizontal scaling, or the dynamic adjustment of service instances, emerges as a cornerstone strategy in this context. It ensures that each microservice can independently scale according to its workload, thereby optimizing resource allocation and minimizing latency.

Nonetheless, the stochastic nature of web traffic and service demands poses a significant challenge, often resulting in either resource over-provisioning — leading to unnecessary expenditure — or under-provisioning, which risks breaching service level agreements (SLAs) and deteriorating user experience.

Cloud computing complements the microservices architecture by providing a scalable and flexible execution environment that supports rapid provisioning and de-provisioning of resources in response to varying workloads. The synergy between microservices and cloud platforms encapsulates the potential for achieving high levels of operational efficiency, agility, and cost-effectiveness. Nevertheless, the ephemeral and volatile nature of microservices workloads necessitates a predictive approach to scaling that transcends reactive and heuristic based methods. Predictive scaling, underpinned by analytical and machine learning models, offers a prescient mechanism to forecast demand and proactively adjust resource allocations, thereby ensuring optimal service performance and resource utilization.

This research explores and validates the efficacy of predictive workload scaling within the microservices architecture, aiming to refine resource management strategies, enhance QoS, and reduce operational costs. Through the development and empirical assessment of predictive models, this study seeks to contribute to the burgeoning field of cloud-native application optimization, presenting a pragmatic pathway to harnessing the full potential of microservices and cloud computing technologies.

B. P4 SWITCH AND ITS CONSTRAINTS

A P4 switch refers to a network switch that is programmable using the P4 language. P4, which stands for "Programming Protocol-independent Packet Processors," is a high-level programming language designed specifically for networking applications. It enables network engineers and researchers to define how packets are processed by the data plane of a network switch, router, or network interface card (NIC), independent of the underlying hardware.

The flexibility provided by P4 programmability allows for the customization of network behaviors, optimization of performance, and rapid prototyping of new network features that are traditionally fixed in network devices. This adaptability is crucial in modern network environments, which require rapid changes in network policies and functions to support diverse applications and services.

P4 switches are often used in data centers, research facilities, and by network equipment vendors who require the ability to experiment with and deploy new network functionalities efficiently.

1) P4 Switch data plane

The data plane of a P4 switch incorporates a Protocol Independent Switch Architecture (PISA). Upon receipt of a packet, it undergoes an initial transformation into a Packet Header Vector (PHV) via the parsing mechanism. Subsequently, the PHV traverses through a pipeline structured with multiple stages of match-action units (MAUs). Within this pipeline, if a specific header field within the PHV, such as the destination port, aligns with an entry in a predefined table, it activates corresponding processes within the action unit linked to that entry. The PHV, after undergoing these designated processes, is then restructured back into a packet format by the deparser. PISA enables the configuration of custom processes, including defining tables and associated actions in the P4 language, which are then implemented within the MAUs.

2) Matching Constraints

For P4 language, the standard library `core.p4` defines three standard match kinds:

1) **Exact match:** The key must match exactly with the field in the rule.
2) **Ternary match:** The key is ANDed with a mask, then compared with the value for exact match.
3) **Longest prefix match (LPM):** This guarantees that the mask is a series of consecutive bits 1 followed by a series of consecutive bits 0. It is widely supported by diverse devices.

Other libraries (e.g., `v1model.p4`) may define additional match kinds such as range match, fuzzy match, but these are not available on many hardware targets.

3) Memory Constraints

Each stage is equipped with two high-speed types of memory:

- **TCAM:** Ternary Content-Addressable Memory suitable for fast table lookups.
- **SRAM:** Static Random Access Memory is used to store exact match table entries and stateful registers.

Packet processing in Match Action Units (MAUs) is limited to simple instructions like integer additions and bit shifts. The entire pipeline should occupy as few stages as possible to prevent packet delays or stalls.

C. EBPF (EXTENDED BERKELEY PACKET FILTER) AND P4 COMPARISON

eBPF is another advanced network programming mechanism that operates within the Linux kernel, allowing for the dynamic modification and enhancement of network behavior without recompiling the kernel. It is primarily used for high-performance packet processing, network monitoring, and security applications. eBPF programs can be written in a high-level language like C and compiled into eBPF bytecode, which runs in a virtual machine-like environment inside the kernel, offering extensive flexibility and safety [8]. Unlike eBPF, P4 is specifically designed for data plane programming, allowing developers to specify how devices process packets. This can be deployed in various platforms, including those supporting eBPF, to tailor the packet processing pipeline specifically to the needs of a network, often on programmable hardware such as ASICs and FPGAs. A notable implementation is the NIKSS, which leverages P4 for programming software-defined networking (SDN) data paths, and translates P4 programs to run in the eBPF execution environment, combining the strengths of both P4 and eBPF [9]. As a result, P4 is more widely in use, we will only be focused on P4 in this project.

III. ALGORITHMS

A. WORKLOAD PREDICTION (BASELINE)

There are five types of methods for Time Series Prediction, which are the Regression Method, Multiple Regression Method, Exponential Smoothing, Stochastic Forecasting Technique, Soft Computing Based Forecasting Technique [10].

1) Vector Auto Regression (VAR) [11]

Vector Auto Regression (VAR) is a statistical model used to capture the linear interdependencies among multiple time series. VAR models generalize the univariate autoregressive model by allowing for multivariate time series. It's an extension of the autoregressive (AR) model to multiple time series, which is why it's called 'vector' autoregression. VAR models are widely used for analyzing the dynamic impact of random disturbances on the system of variables. After estimating a VAR model, impulse response functions can be used to analyze the effect of a one-time shock to one of the innovations on current and future values of the endogenous variables. While VAR models are very flexible and can be used in a variety of contexts, they have limitations. They can

require a large amount of data to estimate accurately and can suffer from the "curse of dimensionality" when dealing with a large number of time series. A VAR model is formulated as follows for k time series and a VAR of order p:

$$Y_{i,t} = c_i + \sum_{j=1}^{p} \sum_{k=1}^{K} A_{ij} Y_{k,t-j} + \varepsilon_{i,t}$$

where $Y_{i,t}$ is the value of the i-th variable at time t, c_i is a constant, A_{ij} are the coefficients of the lagged values of the j-th time series, and $\varepsilon_{i,t}$ is the error term.

2) Support vector regression (SVR)

Support Vector Machines (SVMs) are a robust set of supervised learning methods used for classification, regression, and outliers detection [12]. The versatility of SVMs lies in their ability to handle both linear and non-linear data effectively, using a set of mathematical functions known as kernels. The traditional application of SVMs is in binary classification, but they can be extended to solve regression problems (known as Support Vector Regression). Support vector regression (SVR), introduced by Drucker et al. [13], is an extension of the well-known SVMs and was devised for single-output regression. SVR uses a quadratic program (QP) to obtain the predictions of a single output. SVR is particularly well-suited for situations where the prediction of continuous values is required, and the data may exhibit complex patterns that linear models cannot capture. SVR maintains all the main features that characterize the algorithm for classification (SVM): It uses the same principles of minimization of an error function, maximizing the margin, and controlling the model complexity using a regularization parameter. Similar to SVMs, SVR can perform linear regression in the original input space or it may map the input into high-dimensional feature spaces through the use of kernel functions. The kernel trick is particularly useful when the relationship between the input variables and the target variable is non-linear.

SVR offers several advantages for predictive modeling:

- **Robustness**: SVR is robust to outliers and is capable of handling non-inear relationships effectively. This makes it well-suited for real-world data which often contains noise and anomalies.
- **Flexibility**: Through the use of different kernels, such as linear, polynomial, and radial basis functions, SVR can model complex relationships without the need for extensive data transformation. This flexibility allows it to adapt to various types of data characteristics.
- **Regularization**: The regularization parameter in SVR provides a means to control overfitting. This parameter helps in maintaining a balance between achieving a low error on the training data and minimizing the model complexity, which ensures good generalization to unseen data.

3) Random Forest Regressor

The Random Forest Regressor is an ensemble learning technique for regression tasks, which builds upon the concept of bagging (bootstrap aggregating) and the decision tree algorithms [14]. It creates a forest of trees where each tree is slightly different from the others, and their collective output is used to make final predictions. This method is widely used due to its robustness, simplicity, and effectiveness across a wide range of datasets.

Random Forest Regressor offers several significant advantages:

1) **Bootstrap Sampling**: Each decision tree is trained on a random sample of the data, drawn with replacement from the original dataset, which may lead to some observations being repeated in each sample.
2) **Feature Randomness**: When constructing trees, the split at each node is made not from the best split among all features but from a random subset of features. This increases diversity among the trees and is crucial for reducing correlation between them.
3) **Tree Construction**: Trees are grown to their maximum length without pruning (fully grown and untrimmed) to capture complex patterns, although this can lead to overfitting.
4) **Aggregation for Prediction**: For regression tasks, the final prediction is the average of the predictions from all trees. This averaging reduces noise and generally improves the model's ability to generalize.

4) Gradient Boosting Machines (GBM)

Gradient Boosting Machines (GBM) are a powerful and widely-used ensemble learning technique that builds predictive models from an ensemble of weak learners, typically decision trees. The core idea behind GBM is to iteratively improve the prediction accuracy by focusing on the errors made by previous models.

5) Large Bayesian vector auto regressions [15]

Large Bayesian vector auto regressions (BVARs) are an advancement of the traditional vector auto regression (VAR) models, incorporating Bayesian statistical methods to handle systems with a large number of time series. BVARs address the parameter proliferation problem in large VARs by using shrinkage priors. This Bayesian approach helps in dealing with over-parameterization that often accompanies the inclusion of many variables in a model. Despite their complexity, large BVARs with shrinkage can produce credible impulse response functions, which are essential for interpreting the dynamic relationships among variables.

6) Explainable Boosted Linear Regression (EBLR) [16]

An iterative method that starts with a base model and explains the model's errors through regression trees. It incorporates nonlinear features by residuals explanation and extends to probabilistic forecasting through generating prediction intervals based on the empirical error distribution.

7) Time series forecasting using distribution enhanced linear regression [17]

This method explores enhancing the performance of linear regression for time series forecasting by employing a grouping-based quadratic mean loss function. The approach involves segmenting the input time series into groups and optimizing both the average loss of each group and the variance of the loss between groups over the entire series. This method aims to create a linear model that not only has low overall error but also exhibits robustness to outliers and sensitivity to distribution changes in the time series.

IV. METHODS

A. DATASET OVERVIEW

This section offers a succinct overview of the datasets employed in this research. We utilize seven publicly available datasets, categorized into four distinct types: VM Traces, Serverless Traces, Microservice Traces, and Traffic Request Traces. Each category has been selected to address different aspects of workload prediction and resource scheduling in cloud environments, providing a comprehensive data foundation for our analysis.

TABLE 1. Overview of Datasets used in the Project

Dataset Name	Type	Year	Cite
Azure [18]	VM Traces	2017	630
Azure Functions [19]	Serverless Traces	2020	475
Alibaba AMTrace [20]	Microservice Traces	2022	7
Google Borg [21]	Microservice Traces	2020	326
Calgary-HTTP [22]	Traffic request Traces	1994	54
ClarkNet-HTTP [22]	Traffic request Traces	1995	54
NASA-HTTP [22]	Traffic request Traces	1995	54
Saskatchewan-HTTP [22]	Traffic request Traces	1995	54

1) Traffic request Traces

The datasets are as follows: Calgary-HTTP, ClarkNet-HTTP, NASA-HTTP, and Saskatchewan-HTTP. Each dataset consists of the following five variables:

- **Host making the request:** Hosts are identified as either local or remote.
- **Timestamp:** In the format "DAY MON DD HH:MM:SS YYYY", where DAY is the day of the week, MON is the name of the month, DD is the day of the month, HH:MM:SS is the time of day using a 24-hour clock, and YYYY is the year.
- **Filename** of the requested item.
- **HTTP reply code.**
- **Bytes** in the reply.

a: Calgary-HTTP

The Calgary-HTTP dataset comprises a comprehensive record of HTTP requests made to the Department of Computer Science's WWW server at the University of Calgary, located in Calgary, Alberta, Canada. The dataset encompasses a period of approximately one year, from October 24, 1994, to October 11, 1995, totaling 353 days. It includes 726,739 HTTP requests, with timestamps recorded at a resolution of one second. This dataset provides a significant resource for analyzing web server traffic patterns over an extended period.

b: ClarkNet-HTTP

The ClarkNet-HTTP dataset includes detailed records of all HTTP requests made to the ClarkNet WWW server over a two-week period. ClarkNet is a comprehensive Internet service provider serving the Metro Baltimore-Washington DC area. The data collection was conducted in two phases: the first from 00:00:00 on August 28, 1995, to 23:59:59 on September 3, 1995, and the second from 00:00:00 on September 4, 1995, to 23:59:59 on September 10, 1995. Each phase lasted exactly seven days, culminating in a total of 3,328,587 HTTP requests captured over the 14-day span. All timestamps in the dataset are recorded with a one-second resolution, facilitating precise analysis of request timing and frequency.

c: NASA-HTTP

The NASA-HTTP dataset encapsulates two months of comprehensive HTTP request data from the NASA Kennedy Space Center WWW server in Florida. Data collection was methodically executed in two consecutive periods: the first from 00:00:00 on July 1, 1995, to 23:59:59 on July 31, 1995, spanning 31 days, followed by the second from 00:00:00 on August 1, 1995, to 23:59:59 on August 31, 1995, which inadvertently states seven days but correctly accounts for 31 days. Over this two-month duration, a total of 3,461,612 HTTP requests were logged. All timestamps in the dataset are recorded with one-second resolution, providing detailed temporal data for analysis of web server traffic patterns.

d: Saskatchewan-HTTP

The Saskatchewan-HTTP dataset comprises an extensive collection of HTTP requests spanning seven months to the University of Saskatchewan's WWW server, located in Saskatoon, Saskatchewan, Canada. The data were meticulously collected from 00:00:00 on June 1, 1995, through 23:59:59 on December 31, 1995, covering a total of 214 days. During this period, the server processed 2,408,625 HTTP requests. All timestamps in the dataset are captured with a resolution of one second, providing detailed insights into the server's request handling over the extended duration. This dataset serves as a valuable resource for analyzing web traffic dynamics and server load distribution.

B. CHECK FOR STATIONARITY

It is imperative that all time series variables within the dataset exhibit stationarity, a statistical property characterized by a constant mean and variance over time [23]. A prevalent method for assessing stationarity is the Augmented Dickey-Fuller (ADF) test. The ADF test operates under the null hypothesis that posits the time series as non-stationary. A p-value from the ADF test that is less than the predetermined significance level leads to the rejection of the null hypothesis,

thereby affirming the stationarity of the time series. Should the data prove to be non-stationary, one recommended approach is to apply differencing to the time series data to achieve stationarity.

C. PROPOSED SOLUTION

Recursive Random Projection (RRP) emerges as a robust technique for regressing high-dimensional data, drawing from foundational works [24]–[26]. RRP is fundamentally a Nyström algorithm, primarily employed in approximating the eigenvectors and eigenvalues of matrices [27]. Comparable to Principal Components Analysis (PCA) in its full implementation, Nyström's approach can be executed more rapidly in its variant forms, such as FASTMAP [28]. While RRP's applicability extends to various tasks, our study focuses exclusively on its utility in regression analyses, denoted herein as RRP Regression.

- Localization: take current example, walk it down the cluster tree to find its relevant leaf
- Anomaly detection: Anomalous if you are far away from everything else in your relevant leaf
- Classification: Localization + report mode class label in relevant leaf
- Regression: Localization + report median class label in relevant leaf

The pseudocode below illustrates the procedures involved in the Recursive Random Projection (RRP) Regression.

```
Algorithm: RandomProjectionRegression
Input: data, stop_depth, target_index
Output: predictions

Class RandomProjectionRegression:
   Initialize(data, stop_depth, target_index):
      self.data  data
      self.stop_depth  stop_depth
      self.target_index  target_index
      self.tree  BuildTree(data.to_numpy(), stop_depth)

   Function BuildTree(candidates, enough):
      if not isinstance(candidates, numpy_array):
         candidates  convert_to_numpy_array(candidates)

      if length_of(candidates)  enough:
         median_value  calculate_median(candidates[:, target_index])
         return {"type": "leaf", "median": median_value}

      east_pivot, west_pivot, east_items, west_items  Split(candidates)
      return {
         "type": "node",
         "east_pivot": east_pivot,
         "west_pivot": west_pivot,
         "east_child": BuildTree(east_items, enough),
         "west_child": BuildTree(west_items, enough)
      }

   Function Split(candidates):
      pivot   random_choice(candidates)
      east_pivot  FindFarest(pivot, candidates)
      west_pivot  FindFarest(east_pivot, candidates)
      distance_c  CalculateDistance(east_pivot, west_pivot)

      if distance_c = 0:
         mid_point  length_of(candidates) // 2
         return east_pivot, west_pivot, candidates[:mid_point],
                candidates[mid_point:]

      all_distance  []
      for candidate in candidates:
         distance_a  CalculateProjectionDistance(candidate, west_pivot,
                east_pivot, distance_c)
         append (distance_a, candidate) to all_distance

      sort all_distance by first element of each tuple
      mid_point  length_of(all_distance) // 2
      east_items  first_half_of_sorted(all_distance)
      west_items  second_half_of_sorted(all_distance)
      return east_pivot, west_pivot, east_items, west_items

   Function Predict():
      predictions  []
      for row in data:
         prediction  LocalizeAndPredict(convert_row_to_numpy(row), tree)
         append prediction to predictions
      return predictions

   Function LocalizeAndPredict(data_point, node):
      if node["type"] = "leaf":
         return node["median"]
      if CalculateDistance(data_point, node["east_pivot"]) <
            CalculateDistance(data_point, node["west_pivot"]):
         return LocalizeAndPredict(data_point, node["east_child"])
      else:
         return LocalizeAndPredict(data_point, node["west_child"])

   Static Function FindFarest(pivot, candidates):
      max_distance  0
      farest_point  pivot
      for candidate in candidates:
         distance  CalculateDistance(pivot, candidate)
         if distance > max_distance:
            max_distance  distance
            farest_point  candidate
      return farest_point

   Static Function CalculateDistance(p1, p2):
      return square_root(sum((v1 - v2) for each (v1, v2) in zip(p1, p2)))

   Static Function CalculateProjectionDistance(candidate, west_pivot,
         east_pivot, c):
      distance_a  CalculateDistance(candidate, west_pivot)
      distance_b  CalculateDistance(candidate, east_pivot)
      return (distance_a + c - distance_b) / (2 * c)
```

D. FRAMEWORK OVERVIEW

Our whole in-network intelligence framework is shown in Fig. For our packet prediction task. we obtain a trained BDT. Then, we encode the prediction rules of the BDT into ternary match table entries, and finally install these entries on the P4 program of the switch. Now, for an arriving packet on the switch, the whole prediction task is model-free, and the P4 program will look up the packet's header fields and find its matched table entry and the related class label.

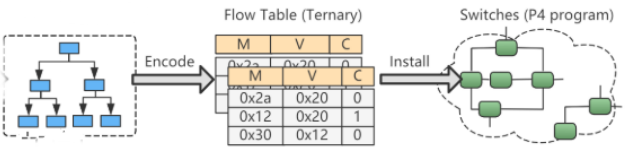

FIGURE 1. Framework Overview

E. EVALUATION METRICS

In this research, we mainly use two evaluation metrics: mean absolute error (MAE) and model size. Since most of the algorithms are stochastic, Scott-Knott is the statistical analysis we used in our study. All of the details are below.

1) Mean Absolute Error

To assess the accuracy of the time series models used in our forecasting, we compute the Mean Absolute Error (MAE). MAE has been widely favored by researchers across various fields as a robust metric for evaluating errors in time series analysis [29]–[31]. It measures the difference between two continuous variables, represented as paired observations X and Y, which correspond to the same underlying phenomenon. Common examples include comparing predicted values versus observed values, assessing changes over time, and comparing different measurement techniques. A scatter plot of n points, where each point i has coordinates (x_i, y_i), facilitates this comparison. The MAE is calculated as the

average distance between each point and the identity line $Y = X$, both vertically and horizontally:

$$MAE = \sum_{i=1}^{N} p_i|\hat{y}_i - y_i| = \sum_{i=1}^{N} p_i|e_i| \quad (1)$$

Here, N represents the total number of observations, p_i denotes the probability of occurrence for each unique value, y_i is the actual value, and $\hat{y}_i$ is the predicted value. Lower values of MAE indicate higher accuracy.

While some researchers [32] advocate for using other metrics like Mean Squared Error (MSE) or Mean Magnitude of Relative Error (MMRE) for evaluating forecast quality, we choose to focus on MAE due to its clarity and straightforward interpretation. This choice is further justified because:

- MAE directly represents the average absolute discrepancy between the forecasted $\hat{Y}_i$ and actual Y_i values, providing a clear and intuitive measure of prediction accuracy. This clarity of interpretation is highlighted and preferred in several studies [30], [31].
- MMRE can distort error assessment, especially when actual values are zero or close to zero, leading to disproportionately large error values that may misrepresent the model's accuracy.

Our focus on MAE supports a transparent and reliable approach to error measurement, mitigating common pitfalls associated with other metrics.

2) Model Size Consideration

In this research, the algorithms are designed to operate on networking devices, which typically have limited memory and computational resources. Consequently, the model size becomes a critical metric for evaluating the feasibility and performance of our approach [33]. It is imperative that our algorithms are optimized to have a minimal memory footprint, ensuring that they can be effectively deployed within the constraints of networking hardware. We anticipate that our refined algorithms will achieve a reduced model size, enhancing their suitability for real-time applications on network devices.

F. STATISTICAL ANALYSIS

Since most of the algorithms are stochastic, and the different train test splits may result in different samples generated by those algorithms, we run each algorithm 10 times and use statistical analysis to compare results through 10 repeats. Scott-Knott is the statistical analysis we used in our study.

More specifically, Scott-Knott recursively partitions the list of candidates l into two sub-lists l_1 and l_2. The split should maximize the expected mean value before and after the division [34]–[36]:

$$E(\Delta) = \frac{|l_1| * |\bar{l}_1 - \bar{l}| + |l_2| * |\bar{l}_2 - \bar{l}|}{|l|} \quad (2)$$

We check if two sub-lists differ significantly by using the Cliff's Delta procedure. The delta value is

$$\Delta = \frac{(\#(x > y) - \#(x < y))}{(|l_1| * |l_2|)} \quad (3)$$

for $\forall x \in l_1$, and $\forall y \in l_2$. Cliff's delta estimates the probability that a value in the sub-list l_1 is greater than a value in the sub-list l_2, minus the reverse probability [37]. Two sub-lists differ significantly if the delta is not a "small" effect ($\Delta \geq 0.147$) [38].

The reason we choose Scott-Knott because (a) it is fully non-parametric and (b) it reduces the number of potential errors in the statistical analysis since it only requires at most $O(\log_2(N))$ statistical tests for the $O(N^2)$ analysis. Other researchers also advocate for the use of this test [39] since it overcomes a common limitation of alternative multiple-comparison statistical tests (e.g., the Friedman test [40]) where treatments are assigned to multiple groups (making it hard for an experimenter to distinguish the real groups where the means should belong [41]).

V. RESULTS

This study evaluated the performance of various predictive modeling techniques for workload prediction in cloud environments using P4 programmable switches. The primary evaluation metric was the Mean Absolute Error (MAE), which directly measures predictive accuracy.

The MAE for each predictive model is summarized in the table below:

Algorithm	Mean Absolute Error (MAE)
Vector Auto Regression (VAR)	9,919.93
Boosted Linear Regression	13,368.73
Support Vector Regression (SVR)	8,410.90
Random Forest Regressor	15,207.49
Gradient Boosting Machines (GBM)	15,249.56
Recursive Random Projection Regression	11,127.49

TABLE 2. Comparison of predictive models based on Mean Absolute Error (MAE).

Support Vector Regression (SVR) emerged as the most accurate model, suggesting its strong capability in handling the non-linear patterns characteristic of cloud workload data. In contrast, models like Gradient Boosting Machines (GBM) and Random Forest Regressor recorded higher errors, which may indicate issues such as overfitting or inadequacies in handling complex data structures. A detailed analysis was performed to explore the discrepancies in performance among the models. The effectiveness of SVR can be largely attributed to its ability to efficiently manage non-linear relationships within the dataset through the kernel trick. Conversely, the ensemble methods, though robust across various scenarios, did not perform optimally, potentially due to the high dimensionality and noisy characteristics of the cloud computing data.

VI. DISCUSSION

Notable anomalies were observed during the evaluations. For example, the unexpectedly poor performance of the Boosted Linear Regression model suggests its linear assumptions may not be suitable for the complex, variable nature of cloud workload data. This highlights the need for further investigation into feature selection and transformation to potentially enhance its predictive accuracy.

VII. FUTURE WORK

The comparative analysis underscores the superiority of Support Vector Regression in forecasting cloud workloads within environments equipped with P4 programmable switches. Future research will focus on integrating these models into real-time systems for more refined and extensive cloud management applications.

VIII. CONCLUSION

The insights derived from this study are crucial for cloud service providers who aim to optimize resource allocation and enhance cost efficiency. Implementing SVR for predictive workload management could lead to significant reductions in resource mismanagement, thus improving service quality and user experience.

REFERENCES

[1] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In 2009 international conference on high performance computing & simulation, pages 1–11. IEEE, 2009.

[2] Anton Beloglazov and Rajkumar Buyya. Energy efficient allocation of virtual machines in cloud data centers. In 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 577–578. IEEE, 2010.

[3] Tania Lorido-Botrán, José Miguel-Alonso, and Jose Antonio Lozano. Auto-scaling techniques for elastic applications in cloud environments. Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09, 12:2012, 2012.

[4] Johan Tordsson, Rubén S Montero, Rafael Moreno-Vozmediano, and Ignacio M Llorente. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future generation computer systems, 28(2):358–367, 2012.

[5] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. ACM SIGCOMM Computer Communication Review, 44(3):87–95, 2014.

[6] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, Lawrence J Wobker, et al. In-band network telemetry via programmable dataplanes. In ACM SIGCOMM, volume 15, pages 1–2, 2015.

[7] S Newman. Building Microservices. O'Reilly Media, Inc., 2015.

[8] Marcos AM Vieira, Matheus S Castanho, Racyus DG Pacífico, Elerson RS Santos, Eduardo PM Câmara Júnior, and Luiz FM Vieira. Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications. ACM Computing Surveys (CSUR), 53(1):1–36, 2020.

[9] Tomasz Osiński, Jan Palimąka, Mateusz Kossakowski, Frédéric Dang Tran, El-Fadel Bonfoh, and Halina Tarasiuk. A novel programmable software datapath for software-defined networking. In Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies, pages 245–260, 2022.

[10] Ganapathy Mahalakshmi, S Sridevi, and Shyamsundar Rajaram. A survey on forecasting of time series data. In 2016 international conference on computing technologies and intelligent data engineering (ICCTIDE'16), pages 1–8. IEEE, 2016.

[11] Ken Holden. Vector auto regression modeling and forecasting. Journal of Forecasting, 14(3):159–166, 1995.

[12] Vladimir Vapnik and Alexey Chervonenkis. Theory of pattern recognition. 1974.

[13] Harris Drucker, Christopher J Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. Advances in neural information processing systems, 9, 1996.

[14] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. R news, 2(3):18–22, 2002.

[15] Marta Bańbura, Domenico Giannone, and Lucrezia Reichlin. Large bayesian vector auto regressions. Journal of applied Econometrics, 25(1):71–92, 2010.

[16] Igor Ilic, Berk Görgülü, Mucahit Cevik, and Mustafa Gökçe Baydoğan. Explainable boosted linear regression for time series forecasting. Pattern Recognition, 120:108144, 2021.

[17] Goce Ristanoski, Wei Liu, and James Bailey. Time series forecasting using distribution enhanced linear regression. In Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part I 17, pages 484–495. Springer, 2013.

[18] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In Proceedings of the 26th Symposium on Operating Systems Principles, pages 153–167, 2017.

[19] Mohammad Shahrad, Rodrigo Fonseca, Inigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In 2020 USENIX annual technical conference (USENIX ATC 20), pages 205–218, 2020.

[20] Kangjin Wang, Ying Li, Cheng Wang, Tong Jia, Kingsum Chow, Yang Wen, Yaoyong Dou, Guoyao Xu, Chuanjia Hou, Jie Yao, et al. Characterizing job microarchitectural profiles at scale: Dataset and analysis. In Proceedings of the 51st International Conference on Parallel Processing, pages 1–11, 2022.

[21] Jiechao Gao, Haoyu Wang, and Haiying Shen. Task failure prediction in cloud data centers using deep learning. IEEE transactions on services computing, 15(3):1411–1422, 2020.

[22] Fatemeh Ebadifard and Seyed Morteza Babamir. Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Cluster Computing, 24:1075–1101, 2021.

[23] Zhiping Lu, Ming Li, Wei Zhao, et al. Stationarity testing of accumulated ethernet traffic. Mathematical Problems in Engineering, 2013, 2013.

[24] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pages 245–250, 2001.

[25] Xiaoli Z Fern and Carla E Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In Proceedings of the 20th international conference on machine learning (ICML-03), pages 186–193, 2003.

[26] Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In Proceedings of the fortieth annual ACM symposium on Theory of computing, pages 537–546, 2008.
[27] John Platt. Fastmap, metricmap, and landmark mds are all nyström algorithms. In International Workshop on Artificial Intelligence and Statistics, pages 261–268. PMLR, 2005.
[28] Christos Faloutsos and King-Ip Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Proceedings of the 1995 ACM SIGMOD international conference on Management of data, pages 163–174, 1995.
[29] Rahul Krishna, Amritanshu Agrawal, Akond Rahman, Alexander Sobran, and Tim Menzies. What is the connection between issues, bugs, and enhancements? lessons learned from 800+ software projects. In Proceedings of the 40th international conference on software engineering: Software engineering in practice, pages 306–315, 2018.
[30] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. Climate research, 30(1):79–82, 2005.
[31] Cort J Willmott and Kenji Matsuura. On the use of dimensioned measures of error to evaluate the performance of spatial interpolators. International journal of geographical information science, 20(1):89–102, 2006.
[32] Ayman Amin, Lars Grunske, and Alan Colman. An approach to software reliability prediction based on time series modeling. Journal of Systems and Software, 86(7):1923–1932, 2013.
[33] Guorui Xie, Qing Li, Yutao Dong, Guanglin Duan, Yong Jiang, and Jingpu Duan. Mousika: Enable general in-network intelligence in programmable switches by knowledge distillation. In IEEE INFOCOM 2022-IEEE Conference on Computer Communications, pages 1938–1947. IEEE, 2022.
[34] Huy Tu, Zhe Yu, and Tim Menzies. Better data labelling with emblem (and how that impacts defect prediction). IEEE Transactions on Software Engineering, 48(1):278–294, 2020.
[35] Tianpei Xia, Rahul Krishna, Jianfeng Chen, George Mathew, Xipeng Shen, and Tim Menzies. Hyperparameter optimization for effort estimation. arXiv preprint arXiv:1805.00336, 2018.
[36] Huy Tu, George Papadimitriou, Mariam Kiran, Cong Wang, Anirban Mandal, Ewa Deelman, and Tim Menzies. Mining workflows for anomalous data transfers. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), pages 1–12. IEEE, 2021.
[37] Guillermo Macbeth, Eugenia Razumiejczyk, and Rubén Daniel Ledesma. Cliff's delta calculator: A non-parametric effect size program for two groups of observations. Universitas Psychologica, 10(2):545–555, 2011.
[38] Melinda R Hess and Jeffrey D Kromrey. Robust confidence intervals for effect sizes: A comparative study of cohen'sd and cliff's delta under non-normality and heterogeneous variances. In annual meeting of the American Educational Research Association, volume 1. Citeseer, 2004.
[39] CE Gates and JD Bilbro. Illustration of a cluster analysis method for mean separation 1. Agronomy Journal, 70(3):462–465, 1978.
[40] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the american statistical association, 32(200):675–701, 1937.
[41] Samuel G Carmer and William M Walker. Pairwise multiple comparisons of treatment means in agronomic research. Journal of Agronomic Education, 14(1):19–26, 1985.