2015 International Conference on Cloud and Autonomic Computing

Proactive Management of Systems via Hybrid Analytic Techniques

Ji Xue
College of William and Mary
Williamsburg, VA, USA
xuejimic@cs.wm.edu

Feng Yan
College of William and Mary
Williamsburg, VA, USA
fyan@cs.wm.edu

Alma Riska
NetApp
MA, USA
alma.riska@netapp.com

Evgenia Smirni
College of William and Mary
Williamsburg, VA, USA
esmirni@cs.wm.edu

***Abstract*—In today's scaled out systems, co-scheduling data analytics work with high priority user workloads is common as it utilizes better the vast hardware availability. User workloads are dominated by periodic patterns, with alternating periods of high and low utilization, creating promising conditions to schedule data analytics work during low activity periods. To this end, we show the effectiveness of machine learning models in accurately predicting user workload intensities, essentially by suggesting the most opportune time to co-schedule data analytics work. Yet, machine learning models cannot predict the effects of performance interference when co-scheduling is employed, as this constitutes a "new" observation. Specifically, in tiered storage systems, their hierarchical design makes performance interference even more complex, thus accurate performance prediction is more challenging. Here, we quantify the unknown performance effects of workload co-scheduling by enhancing machine learning models with queuing theory ones to develop a hybrid approach that can accurately predict performance and guide scheduling decisions in a tiered storage system. Using traces from commercial systems we illustrate that queuing theory and machine learning models can be used in synergy to surpass their respective weaknesses and deliver robust co-scheduling solutions that achieve high performance.**

I. Introduction

High utilization of resources is a design target of scaled-out systems. Often in such systems, data analytics workloads co-exist with high priority user workloads that operate within strict service level objectives (SLOs). Data analytics workloads, e.g., personalized advertising, sentiment analysis, product recommendation, database replication, dominate many systems today. Different than traditional internal work (e.g., garbage collection, snapshots, upgrades), data analytics work requires faster reaction in order to provide timely information [1], [2], [3], e.g., a delayed advertisement event update could cause reduced income or a product recommendation should occur before the user leaves the site. Since data analytics to enhance user experience and regular user traffic share the same hardware, their effective resource management can greatly enhance business value and user satisfaction.

Scheduling user traffic and data analytics work in the same system is a challenging task. Scheduling data analytics too aggressively may cause user traffic to suffer from SLO violations. If scheduled too conservatively, data analytics work could not finish in time, thus could loose its value. With user workload traffic that is repeatable and periodic across different time scales [3], [4], [5], it is natural to interleave data analytics work with user workload at periods of low user demands.

Key to the effective deployment of any policy that interleaves data analytics with SLO-bound user level work, is the prediction of fluctuations in the user workload and especially identifying a priori heavy or spiky loads. Here, we propose *NeuQ*, a neural network/queuing hybrid solution: the queueing model that is the basis of co-scheduling decisions is significantly enhanced by neural networks to improve its accuracy. The queueing models that we use predict the magnitude of the potential performance interference of co-scheduling user and data analytics workloads. An important parameter of the queueing model that greatly affects its prediction accuracy is a priori knowledge of the upcoming arrival intensity, this is successfully provided by the neural network.

To illustrate the effectiveness of *NeuQ*, we consider tiered storage systems as a use case. In storage systems flash-based technologies (e.g., DRAM, SSD) are widely integrated into data centers and scaled-out cloud storage systems [6], [7], [8]. Despite their performance advantages over the traditional disk- or tape-based storage technologies (e.g., HDD), their much higher cost per byte prevents flash-based storage technologies to completely replace traditional disk or tape based storage devices. Hybrid, tiered architectures that integrate flash with various disk technologies are common alternatives.

Tiered storage systems adopt a hierarchical design: fast tiers using flash storage devices aiming at boosting performance and slow tiers using HDD devices for the purpose of balancing capacity and cost, as well as for improving reliability [9], [10]. Their efficiency is based on moving data to the right tier based on statistics of data access frequencies. Data moving, i.e., *what* portion of data ought to be transfered and *when* this should take place, affect greatly performance as access times and capacities across different tiers differ by orders of magnitude. This greatly depends on how the upcoming workload intensity is known in advance and is vital for *NeuQ*'s success in offering co-scheduling decisions, as the effects of tier warming greatly depend on timely information on this measure.

978-1-4673-9566-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCAC.2015.31

NeuQ guarantees user SLOs while maximizing data analytics throughput. To this end, we do not treat data analytics simply as a best effort workload. Instead, our aim is to schedule it as aggressively as the system allows without violating user SLOs. We stress that the above performance targets are by no means a limitation of the proposed hybrid model. Incorporating different targets (e.g., deadlines) for completion of data-analytics workload are also easily handled, as we show here.

The proposed approach is fully automatic and robust. We validate its correctness and efficiency via trace-driven simulation using two case studies: 1) enterprise traces from Wikipedia [4] and 2) arrival traces in a scaled-out storage back-end of a mid-size web service provider that we have also used in prior work[1]. Our extensive experimental evaluation shows that the prediction of the user traffic arrival intensity and data analytics completion time is remarkably accurate. More importantly, compared to other state-of-the-art scheduling approaches, the proposed solution strictly meets user SLOs while serving aggressively data analytics workloads.

This paper is organized as follows. Section II presents experimental numbers of a multi-tiered storage system that motivate this work. Section III presents the hybrid model: the machine learning model for traffic intensity prediction and the queueing model. Section IV presents extensive experimental evaluation via trace driven simulations to verify the correctness and effectiveness of the proposed methodology. Section V discusses related work. We conclude in Section VI.

II. Motivation

We showcase the challenges of performance interference by presenting some results from an experimental testbed that consists of a server with a disk enclosure attached to it, providing data services to a host. Its memory is 12GB and the disk enclosure has 12 SATA 7200RPM HDDs of 3TB each. In our experiments the system memory emulates the fast tier and the disk enclosure is used as the slow tier where the bulk of the data resides. The workload is generated and measured at the host machine. We use `fio` [11] as the IO workload generator and generate two types of IO workloads, user and data analytics, which differ on the active working set size rather than on their access patterns. The working set size for the user workload is 1GB such that it always fits into the memory of the server that emulates the fast tier. The data analytics work has an active working set of 24GB, i.e., it does not fit fully into the fast tier and the large slow tier is accessed to retrieve the data. The access pattern for both user and data analytics workload is 100% small random reads to emulate common enterprise workloads that benefit from prefetching (warm-up) only if the working set can fully (or almost) fit in the high performing tier (i.e., the SSD). We also assume that the system is provisioned in such a way that the fast tier can fit the entire (or the majority of) the active user working set. The fast tier is warmed up via a sequential read of the user working set.

Using `fio`, we generate a random reads workload accessing data stored in our server. The intensity of user IO requests is shown in Figure 1 and it emulates very closely the load pattern of user requests of the Wikipedia and the storage workloads that we use to drive our experiments in Section III. Note that without any data analytics work, the response time of user IO requests remains in the same range of about 150ms. All IOs are served from the fast tier. The user throughput however does increase by one order of magnitude as the arrival intensity increases. This confirms that the storage system does not suffer from queuing delays and it has the capacity to sustain more user load.

We add on the same experiment some data analytics work. Since the user workload pattern has a clear periodicity, we contend that this can be easily captured by a machine learning model (indeed, we show the effectiveness of machine learning models for predicting such periodic patterns in Section IV.) Having observed the time pattern of the first 60 minutes, we co-schedule a data analytics workload at around the 65th minute, i.e., when the user traffic subsides significantly. While the data analytics workload immediately gains a lot of throughput, the performance effect of co-scheduling on the user work is huge: response times increase by two orders of magnitude. Data analytics work is stopped around the 70th minute, but we observe that poor user RTs are sustained for about 6 minutes. This is because the data analytics work has evicted part of the user working set from the fast tier/cache.

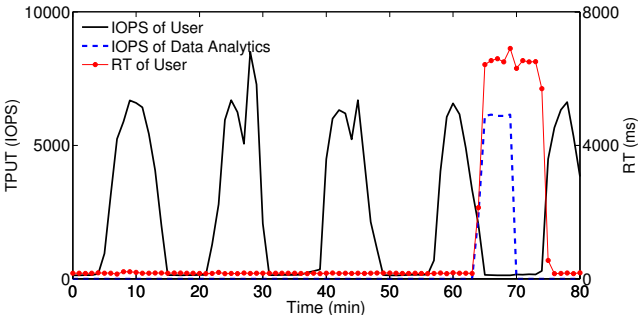


Figure 1: User and data analytics IOPS (throughput) and user response time over time.

What could have made the difference in user performance is the timely fast-tier warm up. Figure 2 captures this effect. In this experiment, we use a small data set of 1GB. We perform two experiments: one with explicit cache warm up (via a sequential read to the user working set) and the other

[1]The Wikipedia traces are publicly available [4]. Due of confidentiality agreements, the storage system trace or provider details can not be made publicly available.

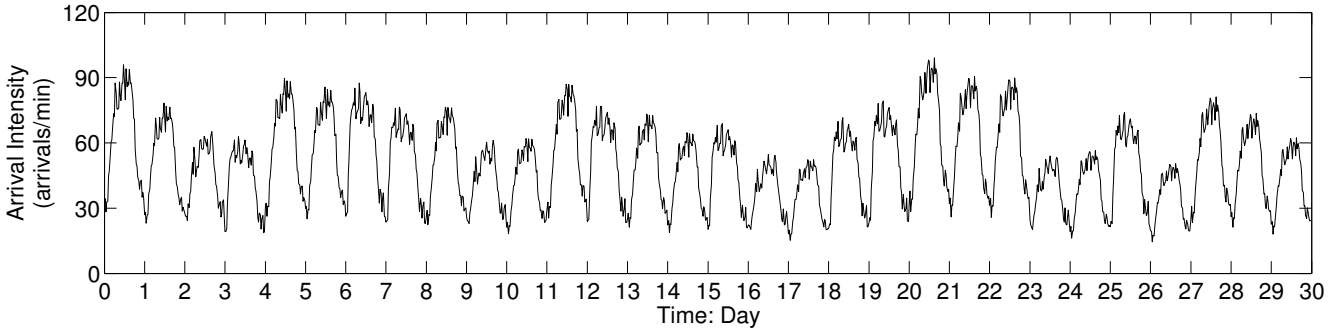


Figure 3: User request arrival intensity of storage workload over one month.

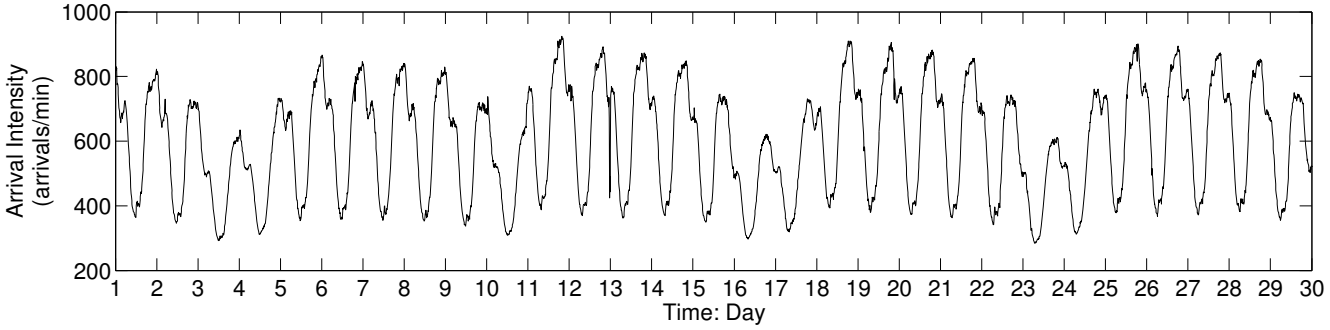


Figure 4: User request arrival intensity to Wikipedia over one month in 2007.

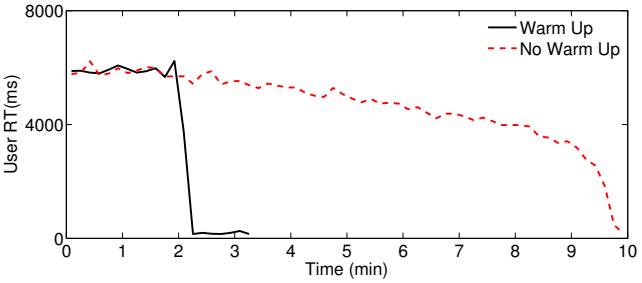


Figure 2: Response time with warm up and without warm up across the experiment time.

one that brings the workload to cache when needed (no explicit warm up). While it takes only about 2 minutes to bring 1GB of data into our fast tier by reading it sequentially, it takes about 10 minutes to fully warm up the cache by the user workload alone. This number is longer than what we observe in Figure 1, this is because in Figure 1 the cache is not completely cold. These experiments demonstrate the importance of incorporating the effects of workload interference and tier warming into any scheduling methodology. The point here is that naively scheduling data analytics during the low period (whose timing can be predicted) does not reach optimized performance. What we need here is a methodology that incorporates tier-warming and accurately predicts when this should start. We address these issues in the following sections.

III. Methodology

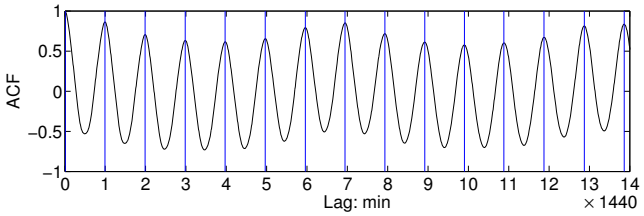
In this section, we start with the description of the user workloads used in this work. Then we illustrate how to use a neural network to build a traffic prediction model. Finally, we introduce in details of the *NeuQ* scheduler that is powered by a queueing model.

A. Workload

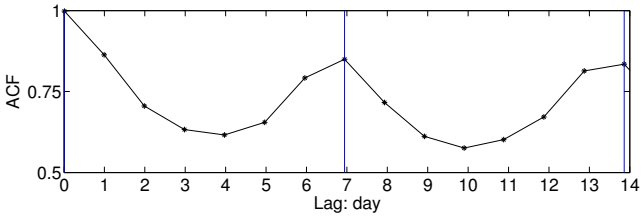
Data center workloads often follow periodic patterns across time [12], [13], [5], [14]. In Figure 3, we demonstrate the arrival intensity of the storage system workload during one month period [5]. In Figure 4, we present the arrival intensity of requests to Wikipedia during October 2007 [4].

Intuitively, the workload of Figure 4 shows a distinctive day/night pattern as well as weekday/weekend pattern. To capture this, we carry out some statistical analysis by calculating the autocorrelation of the time series of the arrival process. Autocorrelation is the cross-correlation of a signal with itself [15]. Intuitively, it captures the similarity between observations as a function of the lag between them. Autocorrelation values are in the $[-1,1]$ range, the closer the lag autocorrelation to 1, the closer the two observations. Zero values indicate no relationship among the observations, while negative values indicate that the range of values of the two observations is diametrically different.

The autocorrelation of user arrival intensity for the Wikipedia workload at varying time lags is shown in Figures 5(a) and 5(b), that report on autocorrelations at the minute lag (10-minute granularity) and day lag, respectively. The figures verify clear daily and weekly patterns, as also observed in Figure 4. Autocorrelation reaches a maximum value per day across all lags illustrating a clear daily pattern, ditto for Figure 5(b) that illustrates a clear weekly pattern. Similar autocorrelation patterns are also observed for the storage workload. In the following section, we use these properties to train a neural network that can model user arrival intensity.



(a) Minute granularity



(b) Day granularity

Figure 5: Autocorrelation of arrivals for different granularities. Note for (a), the points are at 10-minute granularity, the ticks in x-axis is multiplied by 1440, e.g., 2 represents $2 * 1440$ minutes or 2 days, so the lag is up to 14 days.

B. Neural Network Model

Neural networks are a class of machine learning models that can capture periodicity within different time scales. Typically, traditional time series models such as ARMA/ARIMA [16] and Holt-Winters exponential smoothing [17] are limited by the linear basis function. Neural networks can instead model non-linear functions of the input, which makes them effective for time series modeling [18]. A time series analysis consists of two steps: first building a model that represents a time series, and then using the model to forecast future values. If a time series has a regular pattern, then a value of the series should be a function of previous values. If X is the target variable for prediction, and X_t is the value of X at time t, then the goal is to create a model of the form:

$$X_t = f(X_{t-1}, X_{t-2}, X_{t-3}, \cdots, X_{t-n}) + \varepsilon_t, \qquad (1)$$

where ε_t is an error term. An artificial neural network consists of multiple nodes, or *neurons*, which are interconnected to mimic a biological neural network [19]. These neurons have adaptive weights, tuned by a learning algorithm, which enables neural networks to approximate non-linear functions of their inputs. The adaptive weights describe the connection strengths between neurons, activated during training and prediction.

To build a neural network model for a time series, selecting the most *relevant* input that can describe the *trend*, *season*, and *noise* is utterly important. To take care of *trend* and *season*, or in other words, to capture the *long-term* pattern, we make use of the correlogram in Figure 5(b). The figure shows that when the lag equals to seven days, the user traffic is highly and positively correlated. Therefore, as input to our traffic model, we choose the user arrival intensities of exactly one week ago. To capture the temporal change or *noise*, which can be seen as a *short-term* pattern, we look into Figure 5(a) and see that for lag = 10 min the arrival intensities have the highest correlation value. This suggests to consider the user arrival intensities of 10 min ago as another input to the model. With the above inputs as features and current observations, we can train a neural network. When training a neural network, the data are divided into three distinct subsets [20]: the training set, that is used to train the weights (model parameters) of neural network; the validation set, which is used to minimize overfitting thus ensure a generalized solution; and the test set, which is used for evaluating the accuracy of the trained neural network model. Here we use the neural network toolbox provided by MATLAB [21] to train our traffic prediction models.

When training a neural network, even with the same data, the trained model can have different accuracy depending on the configured parameters, e.g., different initial weight values, different divisions of data used for training, validation, and test samples. Our primary attempt to avoid using badly-behaved neural networks (that result in poor prediction), was to train several neural networks on the same data set and select the one with the smallest mean squared error. However, since the future largely remains unknown, it is possible that the "best" neural network fails to do a good prediction in the future. To mitigate this potential problem, we use an ensemble method [22], which averages predictions from different neural networks. Even though the ensemble method may not always provide the optimal prediction, it consistently produces accurate predictions without requiring manual checking.

Last but not least, the computational complexity of the neural network training is not significant, as the prediction model can forecast upcoming traffic for up to a week ahead, suggesting that it is sufficient to update the neural network model as often as once per week for the data in hand. The frequency of training can be adjusted based on different needs.

C. Co-scheduling

As Figure 4 shows, user traffic demonstrates peaks and valleys, suggesting that one could aggressively co-schedule data analytics work during the "low" user periods. Our intention is to quantify how much additional work one could co-schedule such that overall system utilization increases while user SLOs are respected. Because we are aiming to provide a scheduling approach that is easy to deploy and integrate with other management tools, we refrain from making scheduling decisions for user requests too frequently, as this could result in significant overhead. Therefore, our framework divides the time into small windows T_w and makes scheduling decisions only at the beginning of each window[2].

1) Performance model for user traffic in a tiered storage system: To simplify presentation, we assume that the time window that the user SLO is computed is the same as the scheduling window size T_w.

Arrival process: Although in large time windows the arrival process shows a periodical pattern, within each small time window T_w, the arrival process can be viewed as a Poisson process [15].

Service process: In a 2-tiered storage system[3], if the working set of the coming IO request is in the fast-tier (e.g., SSD), the request is served in fast tier. Otherwise, the coming IO request is served in the slow-tier (e.g., disk). Here we assume the service process for each tier follows an exponential distribution. The service process for the 2-tiered system can be described by a hyper-exponential model [23] with mean service time:

$$E[s] = h \times E[s_1] + (1-h) \times E[s_2], \tag{2}$$

where $E[s_1]$ and $E[s_2]$ are the mean service times for the fast tier (tier 1) and slow tier (tier 2), respectively. h is the fast tier hit rate defined as the probability that a request is served from the fast tier. The maximum value of the fast tier hit rate h_{max} is determined by the workload characteristics and the capacity of the fast tier, e.g., if the entire working set can loaded into the fast tier, then the maximum hit rate is 1, in which case all the requests are served in the fast tier. Based on Eq. 2, the expectation of the squared service time can be computed as follows [23]:

$$E[s^2] = 2! \times (h \times E[s_1]^2 + (1-h) \times E[s_2]^2), \tag{3}$$

where $E[s_1]^2$ and $E[s_2]^2$ are the square of mean service times for the fast tier (tier 1) and slow tier (tier 2), respectively.

Queuing model: Based on the above assumptions, the 2-tired storage system can be modeled by a $M/H_2/1$ queue for each small time window. We use the Pollaczek-Khinchine formula [24] to compute the average response time of an $M/H_2/1$ queue:

$$RT = E[s] + \frac{\lambda \times E[s^2]}{2(1-\rho)}, \tag{4}$$

where $E[s]$ is the mean service time, $E[s^2]$ is the mean of squared service time, λ is average arrival intensity, and ρ is the system utilization

$$\rho = \lambda \times E[s]. \tag{5}$$

Combining Eqs. 2, 3, and 5 into Eq. 4, we have:

$$RT = h \times E[s_1] + (1-h) \times E[s_2] + \frac{\lambda \times (h \times E[s_1]^2 + (1-h) \times E[s_2]^2)}{1 - \lambda \times (h \times E[s_1] + (1-h) \times E[s_2])}. \tag{6}$$

From Eq. 6, it is clear that within each time window T_w, the average user response time RT is a function of the average arrival intensity of user requests λ, the mean service time for each tier ($E[s_1]$ and $E[s_2]$), and the fast tier hit rate h.

2) NeuQ Scheduler: Co-scheduling data analytics work may have a performance impact on the user traffic. From our performance model above, the first three parameters (λ, $E[s_1]$, and $E[s_2]$) only depend on the characteristics of user workload and performance of hardware devices, so co-scheduling has no impact on these parameters. However, co-scheduling data analytics work does impact the fast tier hit rate h because it evicts the user working set from the fast tier and reduces the probability of serving user requests in the fast tier. Therefore, in order to meet the user SLO (RT_{target}), the fast tier hit rate of user traffic needs to be maintained above a threshold that is computed from the user SLO and arrival intensity. We compute the threshold (h_{target}) based on Eq. 6 by representing the fast tier hit rate as a function of the user response time target and arrival intensity:

$$h_{target} = \frac{\lambda \times (RT_{target} + E[s_1] - E[s_2]) - \sqrt{P}}{2\lambda \times (E[s_1] - E[s_2])}, \tag{7}$$

where:

$$P = \lambda^2 \times (RT_{target}^2 + 2RT_{target} \times E[s_1] + 2RT_{target} \times E[s_2] + (E[s_1] - E[s_2])^2) - 2\lambda \times (RT_{target} - E[s_1] - E[s_2]) + 1. \tag{8}$$

Note that the fast tier hit rate depends on both user response time target and arrival intensity because the latency is composed of service time and queuing waiting time, e.g., during periods of high arrival intensity, a higher fast tier hit rate is needed to achieve the same response time target than during low arrival intensity periods.

Now the question becomes how to make scheduling decisions such that the data analytics work can be completed as fast as possible without affecting the user SLO. Because the data analytics work is usually measured by the average completion time of submitted jobs or the throughput, it is important to quantify and maximize the (cumulative) time

[2]We assume $T_w = 1$ minute in our experimental evaluation, but this could be adjusted according to the specific system requirement.

[3]For presentation reasons, we use a 2-tiered storage system to explain our methodology, but it could be easily extended to storage systems with more tiers. This discussion also applies to caching.

slot allocated to the analytics work within each time window T_w. Figure 2 and its corresponding analysis in Section II shows the benefit of explicit warm up compared to no explicit warm up. When the system is not as busy with user requests, there are 2 choices: (i) scheduling the data analytics work for time t_{DA}, which reduces the fast tier hit rate or (ii) explicit warming up the fast tier, which recovers the fast tier hit rate.

In order to maximize the time slot allocated to data analytics work but maintain the fast tier hit rate above the threshold h_{target}, different scheduling choices need to be made based on the arrival intensity in the near future. Recall that the future arrival intensity can be predicted using the neural network model introduced in Section III-B. Therefore, we can estimate the fast tier hit rate h_{DA} after scheduling for t_{DA} time units data analytics work:

$$h_{DA} = h_{begin} - \frac{t_{DA}}{t_{evict}}, \tag{9}$$

where h_{begin} is the fast tier hit rate at the beginning of a time window and t_{evict} is the time to evict the entire user working set from a fully warmed up fast tier. Since the data analytics work is very intensive, t_{evict} can be approximated by the time to explicitly warm up the fast tier from completely cold to fully warmed up, which we define as $t_{warmup1}$, this can be easily obtained by a quick profiling experiment such as the one shown in Figure 2. Assuming that the remaining of idle time is used for explicit warm up of the fast tier, the fast tier hit rate h_{warmup} is:

$$h_{warmup} = min\{h_{begin} + \frac{(1-\lambda \times E[s]) \times t_w - t_{DA}}{t_{warmup1}}, h_{max}\}, \tag{10}$$

recall that h_{max} is the maximum value of the fast tier hit rate. In addition, serving user requests also changes the fast tier hit rate (no explicit warm up):

$$h_{user} = h_{begin} + \frac{\lambda \times E[s] \times t_w}{t_{warmup2}}, \tag{11}$$

where $t_{warmup2}$ is the time to non-explicitly warm up the fast tier from completely cold to fully warmed up, which can be easily obtained by a quick profiling experiment. Combining Eqs. 9, 10, and 11, we have the fast tier hit rate at the end of the time window h_{end}:

$$h_{end} = min\{h_{begin} - \frac{t_{DA}}{t_{warmup1}} + \frac{(1-\lambda \times E[s]) \times t_w - t_{DA}}{t_{warmup1}} + \frac{\lambda \times E[s] \times t_w}{t_{warmup2}}, h_{max}\}. \tag{12}$$

Since each time window is very small, we assume that the fast tier hit rate only changes at the end of the window, but stays the same within the window. To meet the SLO, the fast tier hit rate needs to be maintained at or above the threshold i.e., $h_{end} \geq h_{target}$, therefore

$$t_{DA} \leq ((h_{begin} - h_{target}) \times t_{warmup1} \times t_{warmup2} + (1 - \lambda \times E[s]) \times t_w \times t_{warmup2} + \lambda \times E[s] \times t_w \times t_{warmup1}) \times \frac{1}{2 \times t_{warmup2}}. \tag{13}$$

From the above inequality, we can quantify the maximum amount of time to be allocated to data analytics work without violating user SLOs. The co-scheduling decisions are based on Eq. 13. We stress that this inequality is *critical* for the success of workload co-scheduling as it regulates the amount of data analytics work that the system can systain in order to not violate the user SLO.

Because the targeted fast tier hit rate changes with the arrival intensity, scheduling decisions need to be evaluated well in advance. Here we determine how early a scheduling decision needs to be evaluated such that there is enough time to fully warm up the fast tier during this period. We define $t_{advance}$ as:

$$t_{advance} = (h_{max} - h_{current}) \times t_{warmup1}, \tag{14}$$

where $h_{current}$ is the current fast tier hit rate. Based on the predicted arrival intensity after $t_{advance}$, a correct scheduling decision (the time allocated to data analytics work) for the current time window can be made.

In addition, based on the time slot allocated to the data analytics work (Eq. 13), we can estimate the throughput of data analytics work $Throughput_{DA}$ as follows:

$$Throughput_{DA} = \frac{t_{DA}}{s_{DA}} \tag{15}$$

where s_{DA} is the average service time of data analytics work.

If the scheduling target is meeting deadlines for data analytics work, then the throughput of data analytics work needs to meet the above requirements. Assume that the throughput requirement is $Throughput_{target}$, then we have the following:

$$Throughput_{DA} = \frac{t_{DA}}{s_{DA}} \geq Throughput_{target}, \tag{16}$$

therefore,

$$t_{DA} \geq Throughput_{target} * s_{DA}. \tag{17}$$

The above inequality indicates the minimum amount of time to be allocated to the data analytics work in order to meet the deadlines.

IV. Performance Evaluation

An integral part of the effectiveness of workload co-scheduling is the ability to predict accurately the upcoming user workload. We first evaluate the accuracy of the prediction model and then compare the performance of the proposed approach with other methods in the literature. We

also show several scenarios that illustrate the effectiveness of *NeuQ*. Finally, we demonstrate another scheduling target archived by *NeuQ*

A. Traffic Prediction

We drive our simulations using the storage workload in [5] and the Wikipedia trace that is shown in Figure 4 (both workloads with granularity of 1 minute). We trained neural networks for the two workloads and used the trained models to predict incoming traffic for the time period of the next three upcoming days. We also illustrate the model with two different prediction lengths, i.e., 4 hours and 24 hours. Here, if the prediction length equals to K hours, the neural network directly predicts the arrival intensities in the next K hours. In the proposed *NeuQ* scheduler, the traffic intensity information is usually needed only a few hours ahead, so such prediction length can fully satisfy the scheduler's needs. Consistent with the workload analysis of Section III-A, for each observation during training, we select the observations from the most recent time window and the one from one week ago as features. In Figure 6, we show the traffic predictions with two prediction lengths, as well as the actual traffic. The figure illustrates that for both of the storage and Wikipedia traces, the shorter the prediction length, the more accurate the predictions. Yet, even predicting 24 hours ahead, the overall prediction accuracy is still good.

To quantify better the quality of the two prediction lengths, we use the *absolute percentage error* (APE) metric which is defined as:

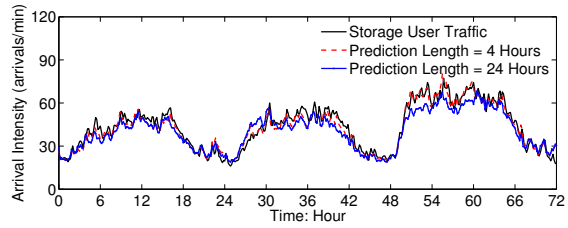
$$APE = \frac{|Prediction - Actual|}{Actual} \times 100\%. \qquad (18)$$

When APE is close to 0, it suggests an accurate prediction. Figure 7 illustrates the CDFs of the absolute percentage error for the two traffic predictions. For the storage trace, Figure 7(a) shows that the 80 percentile of the APE for the 4 and 24-hour predictions, is less than 12 and 18 percent respectively. For the Wikipedia trace, Figure 7(b) shows that for the 4-hour prediction length, almost all the APEs are no more than 10 percent, i.e., predictions are very accurate. For the 24-hour case, over 70 percent of the APEs are less than 10 percent, and nearly 100 percent of them are no more than 20 percent.

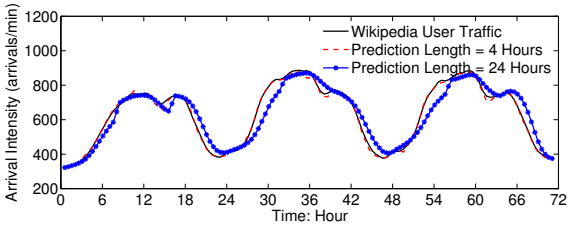
B. Scheduler Comparisons

Here, we compare our *NeuQ* scheduler with two other state-of-the-art scheduling approaches in the literature. We also compare to a scheduling approach that only uses neural network traffic prediction without any help from queueing models.

- *On-line Feedback*
  The on-line feedback method [25] collects measurements during the most recent time window to check whether user performance requirements are met or violated. If met and provided that the system is in a low utilization state in the current window, then data analytics work is added. In the interest of showing the maximum benefit of this scheduler, we assume that we know all future upcoming workload, i.e., we are certain about the length of low/high utilization times due to

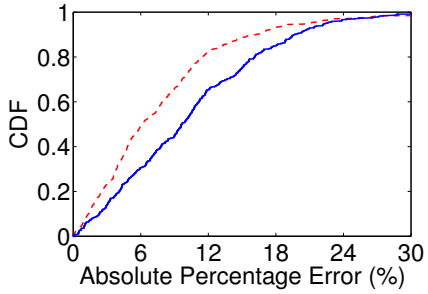


(a) Storage Trace

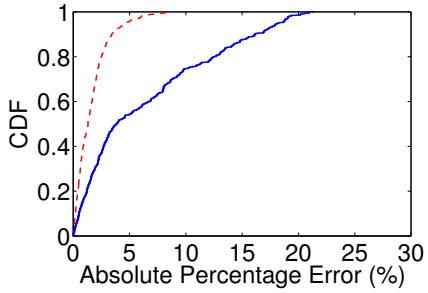


(b) Wikipedia Trace

Figure 6: Traffic predictions from neural networks with different prediction lengths.



(a) Storage Trace



(b) Wikipedia Trace

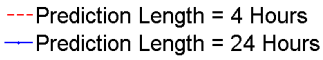
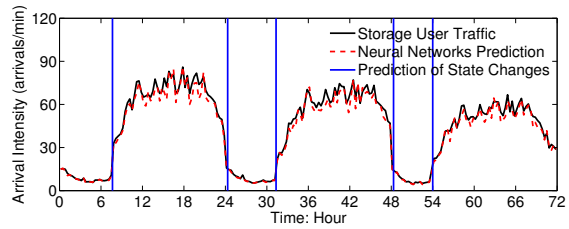
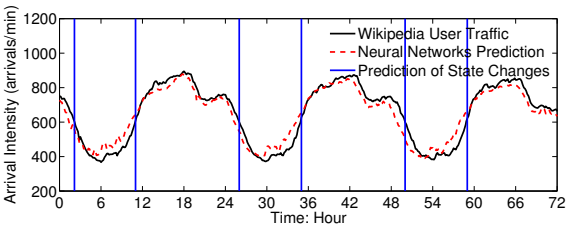


Figure 7: CDF of absolute percentage error of the neural network predictions.



(a) Storage Trace



(b) Wikipedia Trace

Figure 8: Storage system and Wikipedia user traffic for simulation, with the state changes and traffic predictions.

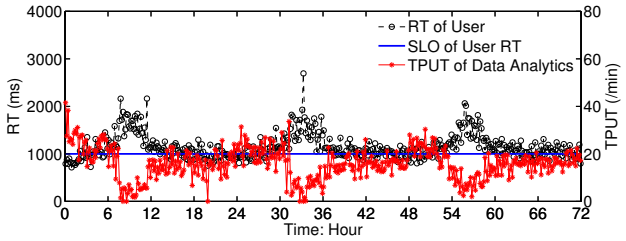
user workload *a priori*.

- *State Prediction*
  The state change prediction based method [5] divides the traffic into high/low states, based on the average arrival intensity. It makes use of a Markovian Modulated Poisson Process (MMPP) model to predict when the high state starts and ends. Based on the state prediction, this method only schedules the data analytics work in the low utilization state, while pro-actively warming up the fast tier with the active user working set right before the arrival of a high utilization state.
- *NeuralNet Scheduler*
  The NeuralNet scheduler decides the amount of data analytics work to schedule based on the user arrival intensity prediction. Here the amount of data analytics work to schedule in each time window increases/decreases linearly with the predicted user arrival intensity in that time window. For example, if for the current window w_1, λ_1 is the average user arrival intensity, and the amount of scheduled data analytics work is n_1, then for the incoming time window w_2 with predicted average user arrival intensity equal to λ_2, the NeuralNet scheduler schedules $\frac{\lambda_1}{\lambda_2} \times n_1$ data analytics work in w_2. Intuitively, if the user arrival intensity is predicted to increase, then the NeuralNet scheduler schedules less data analytics work in the incoming time window. Otherwise, more data analytics work is scheduled.
- *NeuQ Scheduler*
  The *NeuQ* scheduler makes scheduling decisions (how much and when to schedule data analytics work) based on the hybrid model developed in Section III.

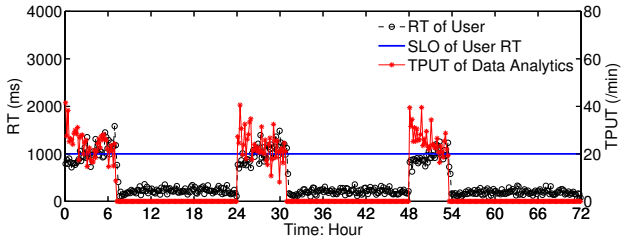
We conduct trace-driven simulations on the storage workload in [5] and the Wikipedia user traffic (days 26, 27, and 28 in Figure 4). Because the *NeuQ* scheduler and the state prediction based scheduler depend on the accuracy of their workload model, we illustrate in Figure 8 how well the MMPP model predicts changes in system state as well as the neural network prediction, both methods are quite effective.

We start by comparing how fast data analytics work each method can complete, given an SLO for user performance. We assume that the service times for the fast and slow tiers are exponential distributed, and that the average service time between the two tiers differs by two orders of magnitude. For the data analytics work, we assume that the average time to finish one unit of work equals to the mean slow tier service time and we use throughput to measure how fast it can be scheduled. Figure 9 illustrates the user RTs and data analytics throughputs for the four policies. In each graph, the horizontal lines represent the pre-defined user SLOs. Figure 9(a) shows that during heavy user traffic periods (e.g., the 6th, 30th and 54th hour), user SLOs are consistently violated due to the aggressive scheduling of data analytics. Similar phenomena are observed in Figure 9(e). The system recovers after the violation is detected. The MMPP-based state prediction method, see Figures 9(b) and 9(f), is more conservative and refrains from scheduling data analytics work while the system is in high utilization. Since this scheduler pro-actively warms up the fast tier, it contains SLO violations. However, the stochastic nature of the underlying Markovian-based model results in unavoidable inaccuracies for the *exact* time that the system changes, which results in SLO violations when the high state approaches. In Figures 9(c) and 9(g), the NeuralNet scheduler is evaluated. The figures show that although less data analytics work is scheduled in heavy user traffic periods than during hours of low traffic, the user SLO is still violated. Figures 9(d) and 9(h) illustrate the performance of *NeuQ*. With *NeuQ*, there are no SLO violations while data analytics work is served aggressively, and with even higher throughput as with on-line feedback. We also list the percentage of user SLO violations, throughput of data analytics, and its coefficient of variation (CV) for the above four policies in TABLE I. *NeuQ* achieves $2\times$ to $3\times$ higher data analytics throughput without any user SLO violation, while for the other policies, the percentage of user SLO violations is much higher and still less efficient on data analytics throughput.

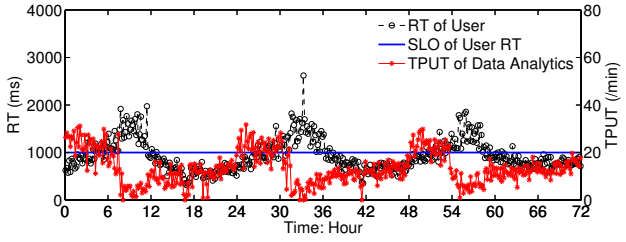
To further illustrate the advantages of our proposed method, we present the CCDFs that illustrate the tails of average user response times of each time window in Figure 10. *NeuQ* manages to strictly respect the user SLO (1000ms for the storage trace and 75ms for the Wikipedia trace).



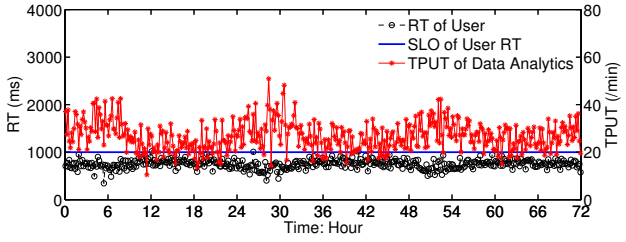
(a) Storage Trace: On-line Feedback



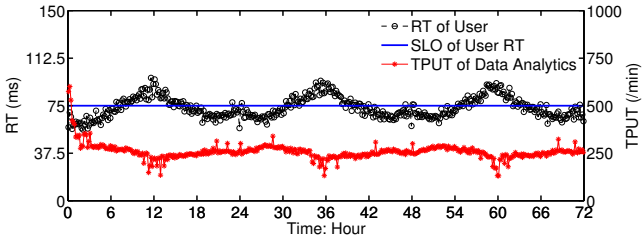
(b) Storage Trace: State Prediction



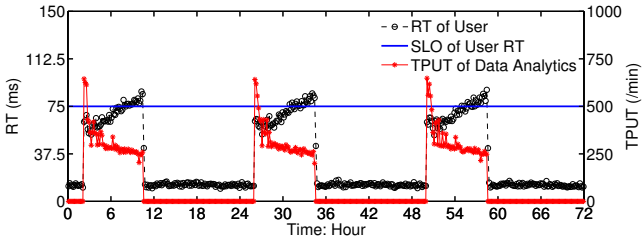
(c) Storage Trace: NeuralNet Scheduler



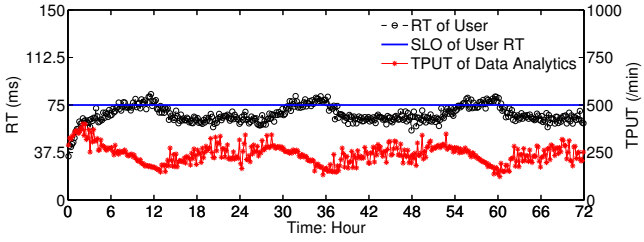
(d) Storage Trace: *NeuQ* Scheduler



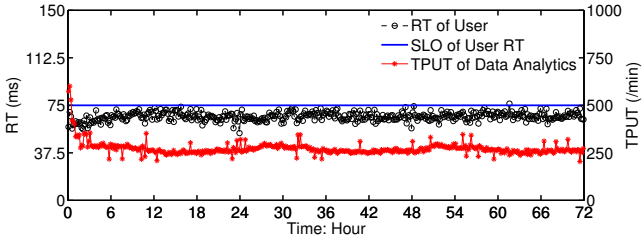
(e) Wikipedia Trace: On-line Feedback



(f) Wikipedia Trace: State Prediction



(g) Wikipedia Trace: NeuralNet Scheduler



(h) Wikipedia Trace: *NeuQ* Scheduler

Figure 9: Performance comparisons via simulation.

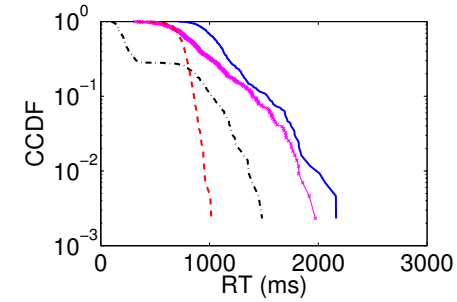
Table I: Average performance analysis for simulations

	Storage Trace		
	% User SLO Violations	$TPUT_{DA}$ *(/min)*	*CV of* $TPUT_{DA}$
On-line Feedback	68.29	18.24	0.42
State Prediction	11.11	7.58	1.71
NeuralNet Scheduler	32.41	15.53	0.52
NeuQ Scheduler	0.00	26.45	0.23
	Wikipedia Trace		
	% User SLO Violations	$TPUT_{DA}$ *(/min)*	*CV of* $TPUT_{DA}$
On-line Feedback	40.05	257.38	0.16
State Prediction	11.81	110.99	1.45
NeuralNet Scheduler	15.51	240.71	0.21
NeuQ Scheduler	0.00	271.69	0.13

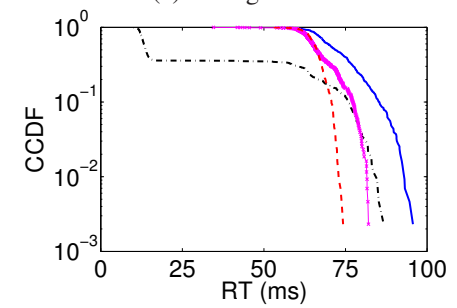
C. Model Effectiveness

The performance prediction model that is developed in Section III is the core of the proposed *NeuQ* scheduler. Recall that the model allows us to regulate the amount of data analytics work to be co-scheduled such that a certain SLO is met for the user work. Similarly, if one needs to increase the throughput of data analytics, this would result in affecting the user RT as well.

Figure 11 illustrates this relationship between user RT (x-axis) and data analytics throughput (y-axis) for the storage and Wikipedia workloads. The figure plots the relationship of these measures as obtained both by simulation and by using the prediction model of Section III (Eq. 15). Both model and simulation numbers are in good agreement, well-capturing the relationship trends of both measures. Further, by using Figure 11 one could estimate the maximum data analytics throughput to be achieved given a certain SLO or conversely the sustained average user RT if a certain throughput for data analytics work is expected.



(a) Storage Trace



(b) Wikipedia Trace

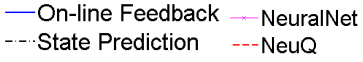
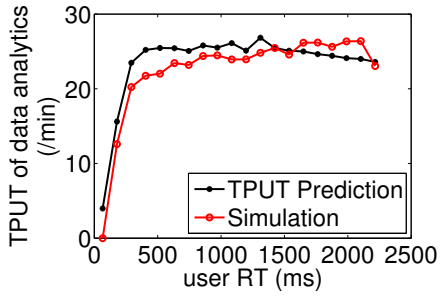
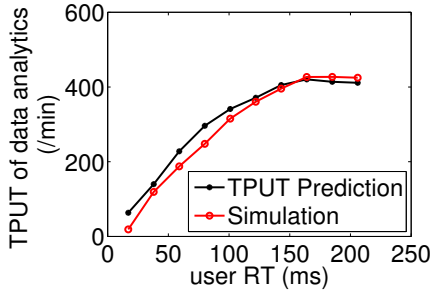


Figure 10: CCDF of average user response time.



(a) Storage Trace



(b) Wikipedia Trace

Figure 11: Throughput of data analytics versus user response time: model (Eq. 15) and simulation (*NeuQ* scheduler).

D. Different Scheduling Targets

NeuQ can be used to support different scheduling targets. Here we demonstrate another scheduling target scenario of finishing the data analytics work by the pre-defined deadlines while preserving the performance of user workload as much as possible. We show two different scheduling targets:

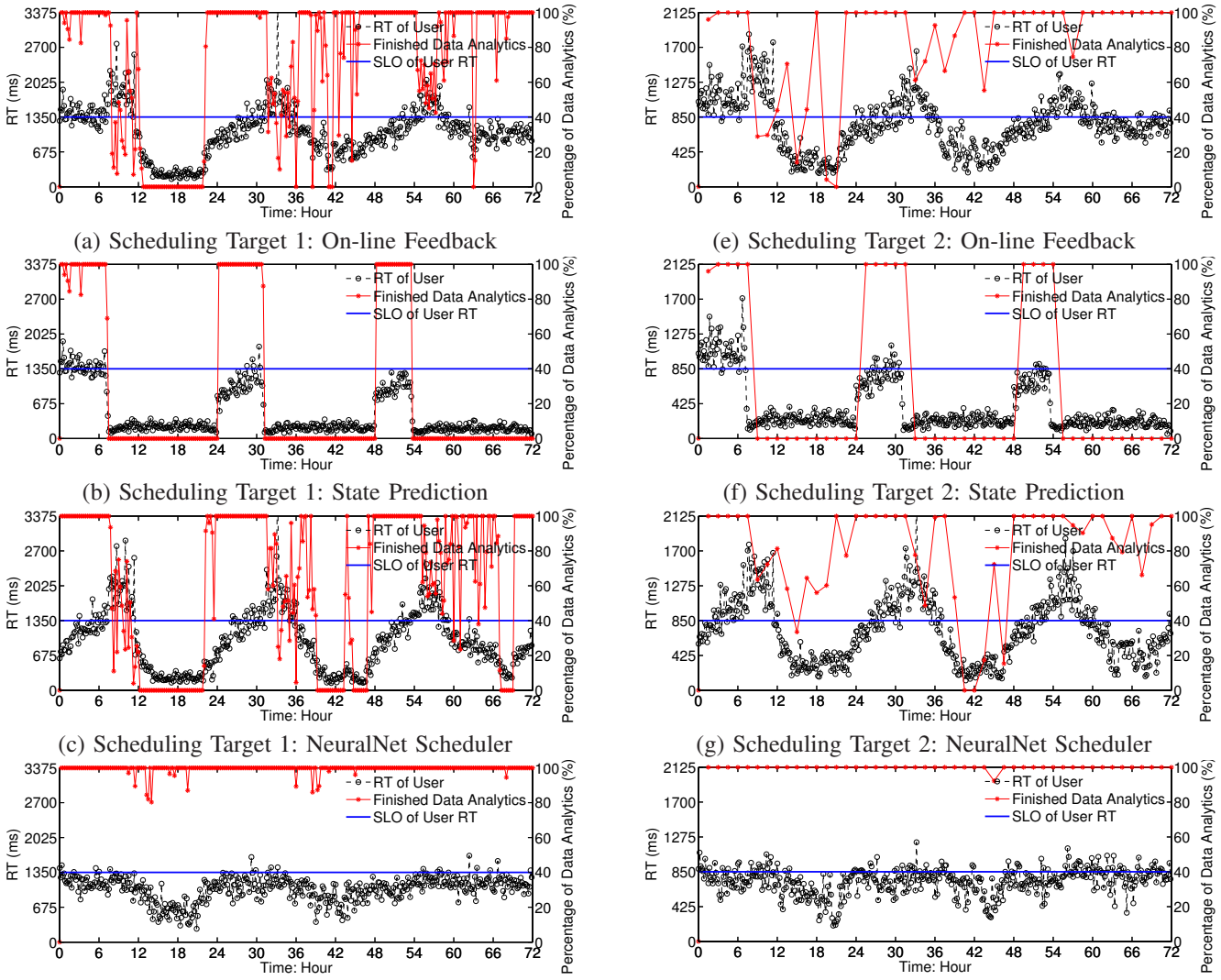
- *Scheduling Target 1*: the deadline of data analytics work is 15 minutes for 500 units of work. The SLO of user workload is SLO 1350ms.
- *Scheduling Target 2*: the deadline of data analytics work is 1.5 hours for 1500 units of work. The SLO of user workload is SLO 850ms.

Scheduling Target 1 has a tighter deadline for data analytics work while relatively loose SLO for the user workload, and Scheduling Target 2 is on the contrary. Both scheduling scenarios consider meeting the deadline of data analytics work as first priority and meeting SLO of user workload as a secondary target. The data analytics work that is not finished by the deadline is dropped so that its impact is not propagated to the future analytics work.

The results are presented in Figure 12. The left y-axis is the user response time measured in ms and right y-axis is the percentage of the finished data analytics work. The x-axis represents the elapsed time (3-day period). For Target 1 (left column of graphs), the results suggest that only *NeuQ* can meet the deadlines of data analytics work (there are a few exceptions, but very few and all above 80%) and all other methods fail. Meanwhile, *NeuQ* also consistently archives the SLO of user workload, this is not the case for other methods. Target 2 (right column of graphs) shares the same requirement for meeting deadlines of data analytics work. Given the stricter SLO, none of the methods can meet the deadlines of data analytics work while also achieving the SLO for user workload. However, it is clear that *NeuQ* results in smallest violation of the SLO of user workload among these methods. The above experiments suggest that *NeuQ* has great use potential in reaching other targets. For the experiments presented in Figure 12, especially for Scheduling Target 1, it is necessary to tolerate high user SLOs to meet the 15 minutes deadline. If low deadlines and low SLOs are to be met, then it is necessary to increase the capacity of the system. *NewQ* can be easily extended to support a capacity planning component.

V. Related Work

Analytical and simulation models have been widely used to quantify the impact of workload changes to application and/or system performance, see [26], [9], [5], [13], [27], [28], [29] and references therein. [5] uses a probabilistic model to define “workload states" via hierarchical clustering. After state discovery, the observed workload is used to parameterize a Markov Modulated Poisson Process that can accurately predict the duration of each state as well as transitions from state to state. ARMA/ARIMA [16] have been adopted in [3] to predict the user traffic overtime in



(a) Scheduling Target 1: On-line Feedback

(e) Scheduling Target 2: On-line Feedback

(b) Scheduling Target 1: State Prediction

(f) Scheduling Target 2: State Prediction

(c) Scheduling Target 1: NeuralNet Scheduler

(g) Scheduling Target 2: NeuralNet Scheduler

(d) Scheduling Target 1: *NeuQ* Scheduler

(h) Scheduling Target 2: *NeuQ* Scheduler

Figure 12: Performance comparisons via simulation. Scheduling Target 1 (left column of graph): the deadline of data analytics work is 15 minutes for 500 units of work and the user workload SLO is 1350ms. Scheduling Target 2 (right column of graph): the deadline of data analytics work is 1.5 hours for 1500 units of work and the user workload SLO is 850ms.

order to achieve cost-efficient capacity planning. However, this prediction method is limited to the linear basis function.

Machine learning techniques are used to overcome the limitation of the linear basis function of the ARMA/ARIMA models, and are used for effective characterization of TCP/IP [30] and web server views [31]. Machine learning techniques [32] have been also used for performance prediction of total order broadcast, a key building block for fault-tolerant replicated systems. Ensembles of time series models have been used to project disk utilization trends in a cloud setting [22].

In general, analytical models are restricted by their simplified assumptions while machine learning models are effective in predicting performance for scenarios that have already been observed in the past and fail when new patterns are observed. A gray-box performance model that combines analytical modeling with machine learning has been proposed [33]. The authors advocate the use of analytical models to lower the initial training time of machine-learning based predictors or enhance the accuracy of the analytic model by adjusting its error with the help of machine learning. In contrast to this work, what we propose here is the usage of machine learning to accurately predict specific inputs of a queueing model, which in turn we use to derive scheduling decisions.

VI. Conclusion and Future Work

Co-scheduling data analytics workloads with user workloads in multi-tiered storage systems is a challenging problem. In this paper, we propose *NeuQ* scheduler, a hybrid co-scheduling approach using machine learning and queueing models, that applies neural networks to predict user workload intensities and then appropriately adjusts the input to a queueing model in order to consistently meet user SLOs. Trace-driven simulations show that *NeuQ* can effectively reach performance targets under different user workloads and different performance/scheduling targets from commercial systems while maximizing the throughput of data analytics work. In the future, we intend to evaluate our *NeuQ* scheduler with different user workloads. We also plan to extend *NeuQ* to support strict SLO for data analytics work by adding a capacity planning component.

Acknowledgments

This work is supported by NSF grant CCF-1218758.

References

[1] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: a self-tuning system for big data analytics." in *CIDR*, vol. 11, 2011, pp. 261–272.

[2] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton, "Mad skills: new analysis practices for big data," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1481–1492, 2009.

[3] Z. Zhuang, H. Ramachandra, C. Tran, S. Subramaniam, C. Botev, C. Xiong, and B. Sridharan, "Capacity planning and headroom analysis for taming database replication latency: experiences with linkedin internet traffic," in *Proceedings of the 6th ACM/SPEC ICPE*, 2015, pp. 39–50.

[4] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, vol. 53, no. 11, pp. 1830–1845, 2009.

[5] J. Xue, F. Yan, A. Riska, and E. Smirni, "Storage workload isolation via tier warming: How models can help," in *Proceedings of the 11th ICAC*, 2014, pp. 1–11.

[6] M. Peters, "3PAR: optimizing I/O service levels," *ESG White Paper*, 2010.

[7] B. Laliberte, "Automate and optimize a tiered storage environment - FAST!" *ESG White Paper*, 2009.

[8] "Amazon ElastiCache," http://aws.amazon.com/elasticache, last visited on: Mar 11th, 2015.

[9] J. Guerra, H. Pucha, J. S. Glider, W. Belluomini, and R. Rangaswami, "Cost effective storage using extent based dynamic tiering," in *FAST*, 2011, pp. 273–286.

[10] Y. Oh, J. Choi, D. Lee, and S. H. Noh, "Caching less for better performance: balancing cache size and update cost of flash memory cache in hybrid storage systems," in *FAST*, 2012, pp. 313–326.

[11] "FIO Benchmark," http://www.freecode.com/projects/fio, last visited on: Mar 11th, 2015.

[12] M. Bjorkqvist, L. Y. Chen, and W. Binder, "Opportunistic service provisioning in the cloud," in *5th IEEE CLOUD*, 2012, pp. 237–244.

[13] D. Ansaloni, L. Y. Chen, E. Smirni, and W. Binder, "Model-driven consolidation of java workloads on multicores," in *42nd IEEE/IFIP DSN*, 2012, pp. 1–12.

[14] R. Birke, M. Björkqvist, L. Y. Chen, E. Smirni, and T. Engbersen, "(Big)data in a virtualized world: volume, velocity, and variety in cloud datacenters," in *FAST*, 2014, pp. 177–189.

[15] L. M. Leemis and S. K. Park, *Discrete-event simulation: A first course*. Pearson Prentice Hall Upper Saddle River, NJ, 2006.

[16] B. George, *Time Series Analysis: Forecasting & Control, 3rd ed*. Pearson Education India, 1994.

[17] P. Goodwin, "The holt-winters approach to exponential smoothing: 50 years old and going strong," *Foresight*, pp. 30–34, 2010.

[18] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems*, vol. 31, no. 1-3, pp. 91–103, 2001.

[19] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, 1st ed. Cambridge, MA, USA: MIT Press, 1995.

[20] T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," *Management Science*, vol. 42, no. 7, pp. 1082–1092, 1996.

[21] H. Demuth, M. Beale, and M. Hagan, "Neural network toolbox^{TM} 6," *User's Guide*, 2008.

[22] M. Stokely, A. Mehrabian, C. Albrecht, F. Labelle, and A. Merchant, "Projecting disk usage based on historical trends in a cloud environment," in *Proceedings of the 3rd workshop on ScienceCloud*, 2012, pp. 63–70.

[23] S. M. Ross, *Introduction to probability and statistics for engineers and scientists*. Academic Press, 2009.

[24] H. C. Tijms, *A first course in stochastic models*. John Wiley and Sons, 2003.

[25] T. Cucinotta, F. Checconi, L. Abeni, and L. Palopoli, "Self-tuning schedulers for legacy real-time applications," in *Proceedings of the 5th EuroSys*, 2010, pp. 55–68.

[26] A. J. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame *et al.*, "Optimis: A holistic approach to cloud service provisioning," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.

[27] R. Singh, P. Shenoy, M. Natu, V. Sadaphal, and H. Vin, "Analytical modeling for what-if analysis in complex cloud computing applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 53–62, 2013.

[28] Q. Zhang, L. Cherkasova, and E. Smirni, "A regression-based analytic model for dynamic resource provisioning of multi-tier applications," in *Proceedings of the 4th ICAC*, 2007, pp. 27–36.

[29] F. Yan, A. Riska, and E. Smirni, "Busy bee: how to use traffic information for better scheduling of background tasks," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. ACM, 2012, pp. 145–156.

[30] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Multi-scale internet traffic forecasting using neural networks and time series methods," *Expert Systems*, vol. 29, no. 2, pp. 143–155, 2012.

[31] J. Li and A. W. Moore, "Forecasting web page views: methods and observations," *Journal of Machine Learning Research*, vol. 9, no. 10, pp. 2217–2250, 2008.

[32] M. Couceiro, P. Romano, and L. Rodrigues, "A machine learning approach to performance prediction of total order broadcast protocols," in *4th IEEE SASO*, 2010, pp. 184–193.

[33] D. Didona, F. Quaglia, P. Romano, and E. Torre, "Enhancing performance prediction robustness by combining analytical modeling and machine learning," in *Proceedings of the 6th ACM/SPEC ICPE*, 2015, pp. 145–156.