Resource and Performance Distribution Prediction for Large Scale Analytics Queries

Alireza Khoshkbarforoushha
Australian National University & CSIRO
Canberra, Australia
a.khoshkbarforoushha@anu.edu.au

Rajiv Ranjan
Newcastle University, UK
raj.ranjan@ncl.ac.uk

ABSTRACT

Efficient resource consumption and performance estimation of data-intensive workloads is central to the design and development of workload management techniques. Recent work has explored the efficacy of using distribution-based estimation of workload performance as opposed to single point prediction for a number of workload management problems such as query scheduling, admission control, and the like. However, the proposed approaches lack an efficient workload performance distribution prediction in that they simply assume that the probability distribution function (pdf) of the target value is already available. This paper aims to address this problem for an inseparable portion of big data analytics workloads, Hive queries. To this end, we combine knowledge of Hive query executions with the novel usage of mixture density networks to predict the whole spectrum of resource and performance as probability density functions. We evaluate our technique using the TPC-H benchmark, showing that it not only produces accurate pdf predictions but outperforms the state of the art single point techniques in half of experiments.

Keywords

Query performance prediction; Distribution prediction; Hive

1. INTRODUCTION

Data-intensive workload management strategies including resource provisioning, workload scheduling, and admission control need the cost of a request to be specified a priori. Recent studies [14, 5, 4] detailed the advantages of distribution-based estimation as opposed to single point prediction of workload performance for profit-oriented admission control [14], efficient query scheduling [5], and cost-optimized cloud resource provisioning [4]. These studies simply assume that the probability distribution function (pdf) of the target value (e.g. CPU, Response time, etc.) is already available. On the other hand, the state of the art estimators [2, 6, 7, 12, 8] model resource and performance of data-intensive workload as a single point value.

ICPE'16, March 12-18, 2016, Delft, Netherlands
© 2016 ACM. ISBN 978-1-4503-4080-9/16/03…$15.00
DOI: http://dx.doi.org/10.1145/2851553.2851578

This leaves us with one key question to answer in this paper: *How can we predict the resource and performance distribution of data-intensive workloads?* Answering this question is important for the increasingly common data-intensive platforms where efficient resource usage prediction is a key operating criterion for proper cluster and resource utilization and service level agreement (SLA) management.

Moreover, in a shared multi-system cluster, having a mix of different applications and workloads (e.g. Pig, Hive) running concurrently is a trivial practice to utilize resources cost efficiently, while challenging accurate workload performance prediction. In this context, we argue that with distribution-based prediction of data-intensive workloads we are able to tackle properly the inevitable performance variances in the presence of resource contention.

To this end, in the subsequent section we derive an optimal proposal model for *CPU* and *Runtime* distribution prediction of the major big data workloads, Hive queries. Apache Hive (https://hive.apache.org/) is a data warehouse infrastructure built on top of Hadoop that facilitates querying and managing large datasets residing in distributed storage. It provides a mechanism to project structure onto this data and query the data using a SQL-style language, HiveQL.

1.1 Overview of the Proposed Approach

Our approach combines knowledge of Hive query processing with Mixture Density Networks (MDN) [3], a flexible technique to modelling real-valued distributions with neural networks. For this purpose, we firstly execute training Hive workloads and log their CPU usage and runtime values along with predefined query features. Secondly, we input the query features to the MDN model. Finally, the MDN statistically analyses the feeding features' numbers and actual observation of the resource consumption and runtime of training data and predicts the probability distribution parameters (i.e. mean, variance, and mixing coefficients) over target values (i.e. CPU and query execution time).

To illustrate the gains possible by using the proposed approach, consider Fig. 1 which displays two sample predicted pdfs for CPU usage and runtime for one of the experiments conducted on TPC-H queries (www.tpc.org/tpch/) in this paper. The predicted pdfs correspond to a test input from Template-7 (Q7) of TPC-H against 100GB database size. To demonstrate the whole possible range of performance values under Q7, the histograms for 30 instance queries based on Q7 from the test set are shown as well.

As we can see, the predicted pdfs properly estimate the

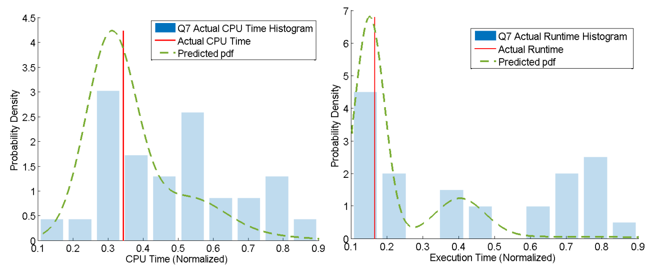

Figure 1: Two sample predicted distributions for (a) CPU and (b) Execution Time for a sample input from Q7 of TPC-H. The histograms show respectively the actual CPU and Runtime values for 30 different instance queries generated based on template-7 and executed in the cluster.

CPU and Runtime distribution in which they show high probability around the target value. More importantly, they provide information about the whole spectrum of performance and resource usage. Specifically, the predicted pdf in Fig. 1(b) shows highly probable Runtime in ranges (0.1, 0.2) and (0.3, 0.5) which are consistent with the actual distribution, though the predicted pdf corresponds to one input, meaning that the resulting uncertainty of pdf for the range (0.8, 0.9) is defensible. Similarly, the predicted pdf for CPU time (Fig. 1a) provides a complete description of the statistical properties of the CPU usage through which we are not only able to capture the observation point, but the different range of resource usage. In contrast, a best prediction from existing single point techniques [8, 2, 12] merely estimates the point which is visualized by solid vertical line through which, unlike the pdf, we are not able to directly extract valuable statistical measures including variance, expectation, and confidence interval about the target.

1.2 Contributions

Distribution-Based Prediction: This paper transfers solid techniques from other computer science fields to the distributed systems community. Although other approaches such as Conditional Density Estimation Network and Random Vector Functional Link are also available to estimate the pdf, the benefit of using MDN is its ability to model unknown distributions. Put differently, the MDN does not hold any assumption about the final shape of resource and performance distribution of workloads.

Resource Modelling of Hive Queries: We develop a set of black-box models for predicting CPU and runtime distribution of Hive query workloads. The models are trained based on a set of SQL and MapReduce specific features and the corresponding data input statistics appeared in the HiveQL query execution plan.

Evaluation: We evaluate our approach on the TPC-H state of the art decision support benchmark, showing that it approximates accurate pdf predictions using proper error metrics for evaluating distribution predictions.

2. RELATED WORK

Query processing runtime and resource usage estimation has been investigated in the context of DBMS or MapReduce [2, 7, 12, 8, 1, 6]. In the majority of related work, different statistical ML techniques are applied for query performance estimation. Specifically, techniques such as Kernel Canonical Correlation Analysis (KCCA), Multiple Additive Regression-Trees, and Support Vector Machine (SVM) have been respectively built upon query plan features [7] operator level features [2, 12] or both [2]. These approaches build statistical models using past query executions and a representative set of query features which have high predictive power in terms of resource or performance estimation.

In terms of concurrent workloads, [1] uses various regression models to predict the completion times of batch query workloads when multiple queries are running concurrently. Along similar lines, [6] argues that the buffer access latency metric is correlated with the query runtime, and they use linear regression techniques for mapping buffer access latency to the execution times. Though the above approaches primarily use statistical ML techniques, they apply fine-grained models in a different context, that of massively parallel data processing in the MapReduce environment.

Another related paper applies KCCA to the Hive workload using two different set of features [8]. In their initial job feature vector they consider features corresponding to the number of occurrences of each operator in a job's execution plan. The obtained results suggest that Hive operator occurrence counts are insufficient for modelling Hive query *runtime* which is somehow consistent with what we will report and discuss in 4.4 (Fig. 4b and 5b). Following that, they include another set of low level features pertaining to Hive query execution such as the number of maps and reduces, bytes read locally, bytes read from HDFS, and bytes input to the map stage, which lead to good prediction accuracy. However, the provided low level features are not available before the query is executed, so that it can not be used for performance prediction of new incoming queries.

Note that all of the above studies approximate the performance of workload as a single point value which is neither expressive enough nor does it capture performance variances.

3. PERFORMANCE MODELLING OF HIVE

To approach the problem of resource and performance distribution prediction of Hive workloads, we use knowledge of Hive query execution in Hadoop combined with statistical machine learning (ML) techniques.

3.1 Query Execution in HiveQL

A key to the accuracy of a prediction model is choosing the most predictive features from the available set of features to train the model. Therefore, we need to identify a set of potential features that would affect the performance and the query resource usage. To identify the potential features we need to dissect the way a HiveQL statement is being executed on top of a Hadoop cluster.

Once a Hive query is submitted against the chunks of data residing in distributed file systems (e.g. HDFS, GFS), the Hive engine compiles it to workflows of MapReduce jobs, in which the SQL operators are plugged into map and reduce functions of the job. At the end of each map and reduce phase, the intermediate results are materialized on disk. During a Hive query execution, SQL specific operators (e.g. table scan, select) which are implemented inside map and reduce functions along with MapReduce specific tasks (e.g. read, spill, shuffle, write) are the main computation tasks which use cluster resources and impact the query completion latency. The overhead of the latter is in fact the function of the number of mappers and reducers spawned across a cluster to execute the query's operators against the data blocks. This number is itself dependent on input data to each query processing stage, job configurations, and available free resources in a multi-system cluster.

As we aim at resource and runtime distribution prediction of Hive queries before any actual execution takes place, we inevitably need to stick with the data provided by the Hive query execution plan. Unlike conventional database systems, the Hive execution plan is an intermediate step before determining the number of mappers and reducers to be executed as a MapReduce job. However, assuming constant configuration, the estimated input data size is a proper predictive feature for alleviating the issue of mappers/reducers numbers and their corresponding Hadoop phases (e.g. reading, spilling, shuffling, writing). Thus, our feature set includes SQL and MapReduce operator counts along with the input record number and data size as specified in Table 1.

Table 1: Feature set for resource modelling of Hive queries.

Feature Name	Description
SQL Operator No	Number of SQL operators (e.g. Table Scan) which appear in the HiveQL query plan.
SQL Operator Input Records	Input Row Numbers for each operator as per the query plan.
SQL Operator Input Byte	Input Data Size to SQL operator.
MapReduce Operator No	Number of MapReduce operators (e.g. Reduce Output Operator), appear in the HiveQL query plan.
MapReduce Operator Input Records	Input Row Numbers for each operator as per the query plan.
MapReduce Operator Input Byte	Input Data Size to the MapReduce specific operator.

We note that the resource contention among different concurrent workloads will impact the performance and following its estimation. However we put forward the claim that with distribution prediction of data-intensive workloads, we are able to tackle properly the inevitable performance variances in the presence of resource contention and runtime configurations. We will discuss this issue in detail in 4.4 and 5.

3.2 Mixture Density Networks

One of the challenging decisions when using statistical models is the choice of the underlying ML technique itself. This is why identifying the most accurate prediction model without training and testing multiple models is hardly possible. Nevertheless, the focus of our work which is conditional probability density prediction, alleviates the problem of picking the right model. We use Mixture Density Networks as an underlying ML technique in the proposed approach. Our decision is backed up by i) its successful applications in speech synthesis or meteorological domain, and ii) its flexibility in capturing skewed and multi-modal distributions, as exhibited by runtime and resource usage distributions in a multi-system cluster.

A classic MDN fuses a Gaussian mixture model (GMM) with multilayer perceptron (MLP). In MDN, the distribution of the outputs t is described by a parametric model whose parameters are determined by the output of a neural network, which takes x as inputs. Fig. 2 gives an overview of MDN in which the neural network is responsible for mapping the input vector x to the parameters of the mixture

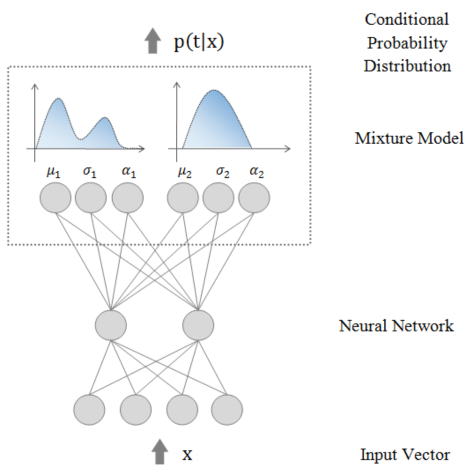
p(t|x)
Conditional Probability Distribution
Mixture Model
μ_1 σ_1 α_1 μ_2 σ_2 α_2
Neural Network
x
Input Vector

Figure 2: MDN approximates distribution parameters, conditioned on the input vector.

model $(\alpha_i, \mu_i, \sigma^2)$, which in return provides the conditional distribution. An MDN, in fact, maps input features x to the parameters of a GMM: mixture weights α_i, mean μ_i, and variance σ^2, which in turn produces the full *pdf* of an output feature t, conditioned on the input vector $p(t|x)$. Thus, the conditional density function takes the form of GMM:

$$p(t|x) = \sum_{i=1}^{M} \alpha_i(x)\phi_i(t|x) \tag{1}$$

where M is the number of mixture components, ϕ_i is the ith Gaussian component's contribution to the conditional density of the target vector t as follows:

$$\phi_i(t|x) = \frac{1}{(2\pi)^{c/2}\sigma_i(x)^c} exp\left\{-\frac{||t-\mu_i(x)||^2}{2\sigma_i(x)^2}\right\} \tag{2}$$

The MDN approximates the GMM as:

$$\alpha_i = \frac{exp(z_i^\alpha)}{\sum_{j=1}^{M} exp(z_j^\alpha)} \tag{3}$$

$$\sigma_i = exp(z_i^\sigma) \tag{4}$$

$$\mu_i = z_i^\mu \tag{5}$$

where z_i^α, z_i^σ, and z_i^μ are the outputs of the neural network corresponding to the mixture weights, variance, and mean for the ith Gaussian component in the GMM, given x [3]. To constrain the mixture weights to be positive and sum to unity, a softmax function is used in Eq. 3 which associates the output of corresponding units in the neural network to the mixing coefficients. Similarly, the variance parameters (Eq. 4) are related to the outputs of the neural network which constrains the standard deviations to be positive.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

Infrastructure Setup. We evaluate our models on CSIRO Big Data cluster. The cluster comprises of 14 worker nodes connected with fast Infiniband network, each featuring 2 x Intel Xeon E5-2660 @ 2.20 GHz CPU (8 cores), 128 GB RAM and 12 x 2 TB NL-SAS HD making up the total disk space of 240 TB. All experiments were run on top of HiveQL 0.13.1, and Hadoop 2.3.0 in Yarn mode on.

Workloads. We test our approach on TPC-H benchmark. We execute TPC-H queries on six scaling factors: 2, 5, 25, 50, 75, and 100 GB. All databases are generated in Apache Parquet data file format. The TPC-H workload consists of all queries except the queries that are either super slow (including Q2, Q8, Q9) or failed (e.g. Q19[1]), thereby we run the super slow queries for solely 2 and 5 GB database size to keep the overall experiment duration under control.

There are approximately 11 queries from each template in six databases. Thus, the resulting data set we used contains 995 queries. Note that our cluster is shared by multiple users in the organization who submit different ranges of applications (e.g. Spark, MapReduce) for processing. Moreover, queries are either run *sequentially* or in *parallel* without any pre-defined ordering to simulate real world conditions as much as possible.

Training and Testing Settings. To assess how the result of a predictive model would be generalized to an independent data set, we divide the TPC-H workload randomly into training and testing datasets with 66% and 34% respectively. Before training and testing, the input and output features are normalized using z-score and min-max normalization with range (0.1-0.9). For training and testing, we use a Netlab toolbox [13] which is designed for the simulation of neural network algorithms and related models, in particular MDN. The implemented MDN model uses the MLP as a feed forward neural network.

4.2 Error Metrics

To determine whether a probabilistic model performs well, we need to implement a set of appropriate error metrics. Therefore, in the following subsection three error metrics including continuous ranked probability score (CRPS), negative log predictive density (NLPD), and root mean-square error (RMSE) are defined. The first two measures are proper metrics for evaluating the accuracy of a distribution prediction. We also use RMSE to compare the MDN with the state of the art single point prediction techniques.

The goal of a probabilistic prediction is to maximize the sharpness of the predictive distributions subject to calibration [9]. Sharpness refers to the concentration of the predictive distributions. Calibration refers to the statistical consistency between the pdfs. The objective is to predict pdfs that closely estimate the region in which the target lies with proper sharpness. Thus, the CRPS [9] is a proper metric to evaluate the accuracy of pdfs. The CRPS takes the whole distribution into account when measuring the error:

$$CRPS(F,t) = \int_{-\infty}^{\infty} \left[F(x) - O(x,t)\right]^2 dx \qquad (6)$$

where F and O are the cumulative distribution functions (cdfs) of prediction and observation distributions respectively. $O(x,t)$ is a step function that attains the value of 1 if $x \geq t$ and the value of 0 otherwise.

To calculate CRPS both the prediction and the observation are converted to cumulative distribution functions. The CRPS compares the difference between cumulative distributions of prediction and observation as given by the hatched area in Fig. 3. It can be seen that the area gets smaller if the prediction distribution concentrates probability mass near the observation, i.e. the better it approximates the step

[1]This issue is also reported by users in https://issues.apache.org/jira/browse/HIVE-600

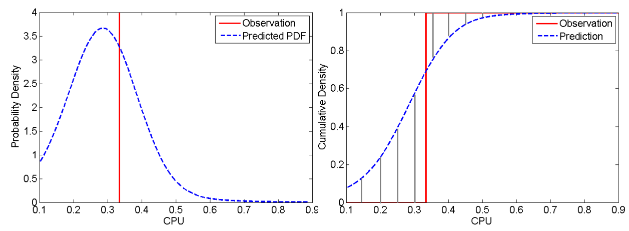

Figure 3: (a) predicted pdf and the observation (b) schematic sketch of the CRPS as the difference between cdfs of prediction and observation.

function. Moreover, the small CRPS value shows that the prediction captures the sharpness of prediction accurately.

After calculating the CRPS for each prediction, we need to average the values to evaluate the whole input set:

$$CRPS = \frac{1}{n}\sum_{i=1}^{n} CRPS(F_i, t_i) \qquad (7)$$

To evaluate the spread of predictive density in which our targets lie, the average NLPD [10] error metric is used. Note that unlike CRPS, it is not sensitive to distance:

$$NLPD = \frac{1}{n}\sum_{i=1}^{n} -log(p(t_i|x_i)) \qquad (8)$$

where n is the number of observations. The NLPD evaluates the amount of probability that the model assigns to targets and penalizes both over- and under-confident predictions.

The last metric is the RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - m_i)^2} \qquad (9)$$

where m refers to the mean of the pdfs as point predictions for the MDNs. This metric allows us to compare the proposed estimation technique with single point competitors.

4.3 State of the Art Techniques

In order to compare the performance of distribution-based prediction with single point estimators, we study REPTree, SVM, and MLP as the alternative techniques. REPTree and SVM are the main prediction techniques used in [14] and [2] respectively. Moreover, [12] also uses a variant of regression trees as a core predictor. Since classical MDN uses MLP in its neural network layer, we can expect that MDN as a single point prediction shows almost the same performance and accuracy as MLP. Thus, we report and discuss the results under MLP as well. These three algorithms are implemented in the well-known Weka package [11].

4.4 Evaluation: Single Point Estimators

Before presenting and discussing the results under the MDN technique, let us first investigate how accurately the Hive query performance and resource usage could be estimated in terms of the proposed feature set (Table 1) using well-established ML techniques.

Fig. 4 displays the performance of REPTree in CPU and runtime estimation of Hive workloads where it approximates resource usage more successfully than runtime. We will discuss this issue later in this section, by then we are interested in the performance of other competing techniques as well. Because, in general, identifying the most accurate prediction model without training and testing multiple models is

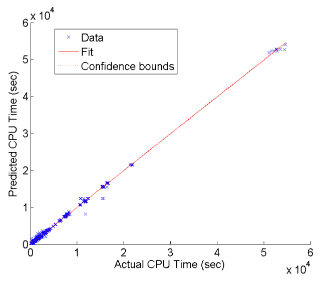

Figure 4: (a) CPU and (b) Response time prediction for Hive queries, modelled using the Table 1 feature set.

hardly possible, thereby the Relative Error (%) of CPU and Runtime estimation of $\sim$ 1000 Hive queries using all three alternative techniques (i.e. REPTree, SVM, and MLP) are evaluated and shown in Fig. 5.

As we can see, REPTree outperforms the other predictors in both CPU and Response Time estimation with relative errors of 4.83% and 13.28% respectively. More importantly, our classifiers are more successful in resource estimation than runtime. The main reason behind this observation is the *resource contention* issue in a shared cluster of machines. As stated earlier, our cluster is shared by multiple users and various applications concurrently processing GBs or TBs of data. Therefore, when multiple jobs and queries are submitted to the cluster, they compete for common resources such as disk, memory, or CPU which might negatively impact the performance. In terms of resource modelling, the contention is not a challenge because our models capture the CPU time which is the amount of time for which CPUs are used for processing instructions, as opposed to, for example, waiting for I/O operations. In contrast, interference of other workloads inevitably hit the query runtime.

We argue that with the distribution of query performance we are able properly to capture and express the whole spectrum of performance (i.e. here response time) and any possible variances in presence of resource contention. To capture the impact of the concurrency and interference in performance, there are some proposals [1, 6] for query executions in DBMSs. However, the proposed techniques are not applicable to the Hive workloads in a multi-system cluster due to i) different abstraction level of query processing in Hive, and ii) lack of control on the type of concurrent workloads in a cluster where they typically hold some assumptions about the mixture of queries running concurrently. Nevertheless, our approach relaxes such constraints and more importantly it is able to estimate the performance while the concurrent workloads are not even from the same platform, for example, where a certain Hive query is competing with Spark jobs for the CPU shares.

4.5 Evaluation: Distribution-Based Prediction

We now discuss the accuracy of the proposed approach. The results for both the proposed approach using MDN and the single point estimators under CRPS, NLPD, and RMSE metrics are shown in Table 2. Note that the number of Gaussian components is a hyper-parameter in MDN and needs to be specified beforehand. To do so, we report the results under 1, 3, and 5 mixture components (M).

All three metrics are negatively oriented scores; hence the smaller the value the better. Let us first study the accuracy of the MDN per se using CRPS and NLPD metric errors. As the small numbers under CRPS and NLPD indicate, the

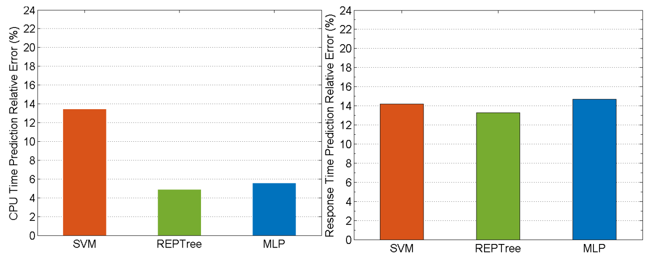

Figure 5: Relative error (%) for (a) CPU and (b) Response time prediction using SVM, REPTree, and MLP techniques.

proposed model is an appropriate estimator for both CPU and Runtime distribution prediction of Hive workloads. Unlike single point estimators, the MDN shows slightly better performance in Runtime prediction rather CPU. Another interesting observation is that in the TPC-H workload sophisticated MDN architecture with 3 and 5 mixture components led to increased fidelity of results.

To compare the proposed approach with the competing techniques, we need to treat it as a single point estimator, thereby we use RMSE metric error for comparison. According to Table 2, the MDN outperforms SVM in CPU prediction, albeit REPTree has the lowest RMSE value. Similarly, REPTree outperforms the others in Response Time (RT) estimation. However, this is not the whole story.

Taking the output corresponding to the mean of the predicted pdfs is almost equivalent to using an MLP with linear output activation function, trained with a least-squares error function. It means that the MLP classifier accuracy is comparable to the MDN. A closer look at the data indicates that the RMSE values under the MLP are in between SVM and REPTree. This observation is consistent with what we saw in RMSE values for MDN.

However, the question may arise *"Why tharee RMSE values under MDN and MLP totally different?"*. This observation is sourced from the different normalization and configuration parameters used in MLP implementation in Netlab toolbox [13] and Weka [11] which are respectively employed for the MDN and MLP (as a standalone technique) training. To test our hypothesis, we replaced the default MLP configuration of Weka with what is used in Netlab, observing almost the same RMSE errors.

In summary, our approach outperforms the state of the art single point techniques in 2 out of 4 experiments conducted using SVM and REPTree. This result is quite promising because it shows that our approach is not only able to predict the full distribution over targets accurately, it is also a reliable single point estimator.

4.6 Training Times and Overhead

Table 3 denotes the training times regarding different workload sizes. As the results indicate, the training cost is very

Table 2: MDN performance compared with its competitors.

MDN					**REPT.**	**SVM**	**MLP**
Targ.	M	CRPS	NLPD	RMSE	RMSE	RMSE	RMSE
CPU	1	0.093	-1.2	**0.077**	0.005	0.08	0.048
	3	0.091	-2.54	0.08			
	5	**0.024**	**-2.65**	0.081			
RT	1	0.064	-1.1	**0.077**	0.01	0.073	0.031
	3	0.031	-2.68	0.079			
	5	**0.017**	**-3.2**	0.08			

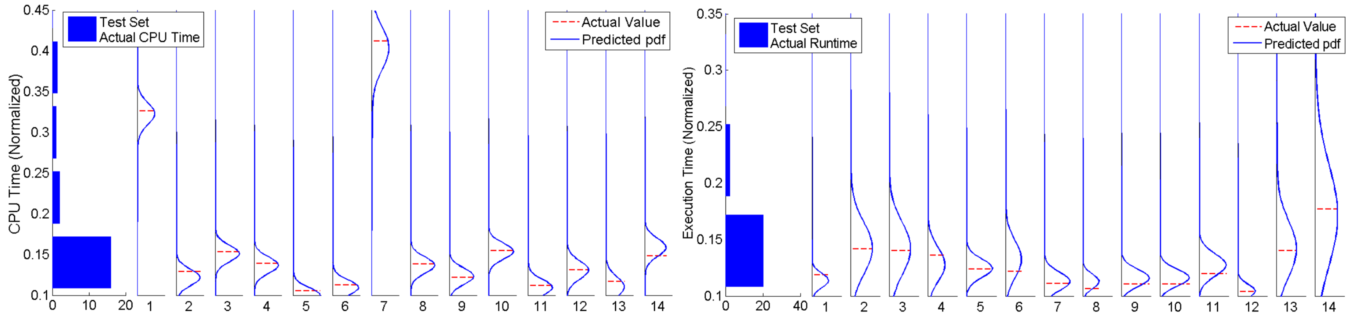

Figure 6: Sample pdf predictions for (a) CPU and (b) Execution Time of Hive queries based on TPC-H workload.

small and it grows linearly in terms of the training set size.

Table 3: Training times in seconds with regard to different workload sizes for 500 iterations.

Workload Size	**1K**	**2K**	**4K**	**8K**	**16K**
Elapsed Time (sec)	1.47	1.9	2.63	3.84	7.83

Apart from reasonable training time, low overhead in invoking a trained model at runtime is yet another critical parameter because it has to be quick enough to get the estimates ready in time for decision making modules of the workload management strategies at runtime, where unreasonable delays may lead to SLA misses. To this end, we measured the elapsed time for evaluating an MDN model for a given input feature set on a 2.80GHz Intel Core i7, and obtained an overhead of about 0.2 ms for each call. To put these numbers in perspective, the execution plan generation in Hive (using EXPLAIN command) for say Q1 of TPC-H takes 4.97 seconds, meaning that invoking the MDN model for each new incoming query would not be a significant factor in the overall workload management cost.

5. PREDICTION UTILIZATION

This section provides a clear picture of how the provided prediction could be utilized and employed in workload management of data-intensive applications. We have visualized some sample predicted pdfs from the test set of the TPC-H workload as shown in Fig. 6. In particular, the figure plots 14 random sample predicted pdfs for CPU and execution time. The histograms show the actual CPU and runtime values for the whole test dataset. Each pdf may (not) belong to different queries as they were randomly selected from the test set, meaning they are conditioned on different inputs. The dotted vertical line shows the observation value.

As the figures show, the pdfs accurately approximate the resource usage and performance distributions which are primarily within the range (0.1, 0.4) and (0.1, 0.25) for CPU and runtime respectively. In a consistent manner, the models for CPU and execution time beyond the values 0.5 and 0.3 are much more uncertain. Put differently, the tendency of all CPU and runtime pdfs is to the right hand side of diagram and this is consistent with the plotted histograms of actual resource and performance values in which, for example, we hardly face resource demand above 0.5.

These sample pdfs demonstrate that the MDN is also a reliable classifier in the classic point estimate sense, where the pdfs cover the observation points with high probability in all figures but pdfs number 14 in 6(a) and 8 in 6(b). However, they locate the shape of distributions precisely.

We also argue that distribution-based prediction gives the resource and workload management systems a concise yet lucid way of interpreting workload behaviour. Such capability is crucial for a number of resource management activities such as run-time performance isolation or diagnosis inspection. In particular, upper and lower bounds of resource usage simplify the task of performance isolation, since for example our predictions in all figures capture the dominant CPU time precisely. When it comes to performance inspection, diagnosing abnormal behaviour as per the predicted numbers is also viable. Specifically, Fig. 6(a) reports that for a given set of queries we will not face peak CPU time ($>$0.5) very often, hence a higher peak CPU time indicates the possible presence of a fault in the software or cluster.

6. CONCLUSIONS AND FUTURE WORK

This work presented a novel approach of using mixture density networks for CPU and runtime distribution prediction of large-scale analytics queries. We evaluated our approach on TPC-H, showing that it outperforms the state of the art techniques in half of experiments. For future work, we plan to devise a distribution-based admission control and query scheduler on top of Apache Yarn to avoid resource usage spikes on busy clusters.

7. REFERENCES

[1] M. Ahmad et al. Predicting completion times of batch query workloads using interaction-aware models and simulation. In *EDBT*, pages 449–460. ACM, 2011.

[2] M. Akdere et al. Learning-based query performance modeling and prediction. In *ICDE*, pages 390–401. IEEE, 2012.

[3] C. M. Bishop. Mixture density networks. 1994.

[4] S. Chaisiri et al. Optimization of resource provisioning cost in cloud computing. *TSC*, 5(2):164–177, 2012.

[5] Y. Chi et al. Distribution-based query scheduling. *VLDB*, 6(9):673–684, 2013.

[6] J. Duggan et al. Performance prediction for concurrent database workloads. In *SIGMOD*, pages 337–348. ACM, 2011.

[7] A. Ganapathi et al. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *ICDE*, pages 592–603. IEEE, 2009.

[8] A. Ganapathi et al. Statistics-driven workload modeling for the cloud. In *ICDEW*, pages 87–92. IEEE, 2010.

[9] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[10] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 107–114, 1952.

[11] M. Hall et al. The weka data mining software: an update. *SIGKDD*, 11(1):10–18, 2009.

[12] J. Li et al. Robust estimation of resource consumption for sql queries using statistical techniques. *VLDB*, 5(11):1555–1566, 2012.

[13] I. Nabney. *NETLAB: algorithms for pattern recognition.* Springer Science & Business Media, 2002.

[14] P. Xiong et al. Activesla: a profit-oriented admission control framework for database-as-a-service providers. In *SoCC*, page 15. ACM, 2011.