

RESEARCH

Open Access



WISE: a computer system performance index scoring framework

Lorenzo Luciano^{1*} , Imre Kiss², Peter William Beardshear¹, Esther Kadosh¹ and A. Ben Hamza³

Abstract

The performance levels of a computing machine running a given workload configuration are crucial for both users and providers of computing resources. Knowing how well a computing machine is running with a given workload configuration is critical to making proper computing resource allocation decisions. In this paper, we introduce a novel framework for deriving computing machine and computing resource performance indicators for a given workload configuration. We propose a workload/machine index score (WISE) framework for computing a fitness score for a workload/machine combination. The WISE score indicates how well a computing machine is running with a specific workload configuration by addressing the issue of whether resources are being stressed or sitting idle wasting precious resources. In addition to encompassing any number of computing resources, the WISE score is determined by considering how far from target levels the machine resources are operating at without maxing out. Experimental results demonstrate the efficacy of the proposed WISE framework on two distinct workload configurations.

Keywords: Machine performance, Workload, Migration services, Compute performance, Virtual machines, Machine score, Cloud computing

Introduction

Determining how well a machine is performing with a given workload configuration is not easily determined, as it can be approached from many different angles. Depending on the workload, optimum performance levels may also differ. While one workload may require a certain memory buffer, another one may excel by pushing the central processing unit (CPU) utilization boundary. Choosing the right computing configuration is a challenging problem and often becomes more difficult as a result of the vast amount of virtual machine (VM) instance types available on cloud computing platforms. Each instance type varies the amount of a compute resource and the resource-to-resource ratio. For example, Compute intensive instances may have a higher CPU to random-access memory (RAM) ratio, while memory intensive instances may have higher RAM to CPU ratio, and similarly with all other resources.

The choice of resource amount and ratio is of vital importance, as it has a crucial impact on the performance of the machine for a workload configuration [1]. Given the number of possible workload configurations and the ever-growing list of instance types, it has become paramount in today's cloud computing environment to design a method for evaluating workload/machine combinations. Alipourfard et al. [2] address this issue by introducing CherryPick, a system that uses Bayesian Optimization to build performance models for various workloads that distinguish optimal or close to optimal VMs from the rest, with only a few test runs per workload configuration.

The different types of workload configurations can have varying effects on the physical computing resources seen through the resource utilization data. Understanding how different workload configurations affect computing resources is crucial for proper resource allocation. To efficiently allocate system resources for a workload, the capability to properly predict the characterization of a workload on the computing resource is essential whether

*Correspondence: lorluc@amazon.com

¹ Amazon, AWS, Boston, MA, USA

Full list of author information is available at the end of the article

that be for on-premises or on cloud computing environment. Koh et al. [3] use workload characterization by studying the effects of performance at system-level workload characteristics. By analyzing the collected data, they were able to identify different application clusters that generate certain types of performance. They subsequently developed models to predict the performance of a new application from its workload characterization. Khan et al. [4] identify repeatable workload patterns by exploring cross-VM workload correlations resulting from the dependencies among applications running on different VMs. By treating workload data samples as time series, they used a clustering technique to identify groups of VMs that exhibit correlated workload patterns. Then, they used a method based on hidden Markov models (HMM) to characterize the temporal correlations in the discovered VM clusters and to predict variations of workload patterns.

Understanding the characterizations of a workload configuration and how they affect computing resources is essential for improving compute resource allocation for the workload. However, customers of cloud computing providers are tasked with choosing a VM instance type from a large list of combinations of family types and sizes. Not only is the choice prohibitively difficult, but has enormous implications on both performance and cost for the given workload configuration. A fundamental void exists for a performance indicator that accurately, economically and consistently provides a performance score for a given workload and computing machine configuration. Such a performance indicator would help determine the computing configuration that would perform best with the workload configuration. Hsu et al. [5] show that there is often a single cloud configuration that is surprisingly near-optimal for most workloads. They introduce a collective-optimizer, MICKY, that reformulates the task of finding the near-optimal cloud configuration as a multi-armed bandit problem. MICKY efficiently balances exploration of new cloud configurations and exploitation of known good cloud configuration. More relevant work on the challenge of mapping workload categorizations to physical resources can be found in [6]. Yadwadkar et al. [7] address the problem of optimal VM selection with PARIS, a data-driven system that uses a hybrid offline and online data collection and modeling framework. PARIS predicts workload performance for different user-specified metrics, and also the resulting costs for a wide range of VM types and workloads across multiple cloud providers. While the aforementioned methods deliver a VM, the proposed WISE score, however, tells us whether we need to change our configuration or if it is good choice.

The proliferation of cloud computing availability and the mass adoption of cloud computing as a viable option to on-premises computing have triggered the need for

workload/machine performance indicators. Cloud computing offers high availability, scalable, efficient and cost saving computing performance for any workload configuration. These considerations require resource planning to determine the required compute resources and how/when compute resources need to grow or be scaled back. Compute resource planning requires a good understanding of both the computing resources capacities and the workloads resource utilization patterns. Rjoub et al. [8] address the task of guaranteeing performance while minimizing resource utilization from a task scheduling perspective. They propose a trust-aware scheduling solution, which consists of VMs' trust level computation, tasks priority level determination, and trust-aware scheduling. Also, Rjoub et al. [9] present an automated big data task scheduling approach for cloud computing environments. The approaches introduced in [8, 9] are for optimum resource utilization from a task scheduling perspective, whereas in our work we describe methods for scoring the performance of a workload on a particular machine.

From a workload characterization perspective, Mishra et al. [10] present an approach for workload classification by identifying the workload dimensions, constructing task classes using a clustering algorithm, determining the break points for qualitative coordinates within the workload dimensions, and merging adjacent task classes to reduce the number of workloads. They show that the duration of a task is bimodal, has either a short or a long duration, and that most tasks have a short duration. Also, that most compute resources are consumed by a few tasks with a long duration that have large demands for CPU and memory capacity. Downey et al. [11] present a characterization of a workload on a parallel system from an operating system perspective by investigating means to characterize the stream of parallel jobs submitted to the system, their resource requirements, and their behavior. A comprehensive survey of workload characterization can be found in [12].

In this paper, we present an integrated approach to derive computing machine and computing resource performance indicators for a given workload configuration. The proposed WISE framework indicates how well a computing machine is running with a specific workload configuration, and whether a different computing configuration could get better performance. Given the need for a computing performance indicator for a given workload configuration, we describe a novel method for scoring the performance of the machine while running the workload, meaning is the machine being under-utilized, over-utilized or is it running in a sweet spot? In particular, we consider how far off from described target levels the resources are running at, i.e. whether there exists resource waste or strain.

The rest of this paper is organized as follows. In “[Method](#)” section, we present the WISE framework for scoring the performance of a machine, given a specific workload, and we describe its main algorithmic steps. Experimental results using the WISE framework on distinct workloads are presented in “[Experiments](#)” section. Finally, we conclude in “[Validation](#)” section and point out some future work directions.

Method

In this section, we introduce the WISE framework for computing a fitness score for a workload/machine combination. The proposed approach encompasses described resource target levels and ranges, as well as the utilization distances from these targets into a single metric. This performance metric is used to easily determine how well a workload/machine combination is performing and also to diagnose possible issues. A key advantage of WISE is that we can use as many computing resources as necessary. In addition, computing resource can use as many aggregate utilization percentage rates as desired. Figure 1 illustrates hypothetical data points in two dimensions (CPU and memory), with machines closer to the targets having a value closer to 1. The WISE score encompasses this information into an index score that indicates the performance of the machine given a workload.

For each computing resource aggregate combination, we first define a target level and range, and then set the

limits as to what are acceptable running rates for such a resource using that aggregation. For example, in the case of CPU we can set the cpu utilization average running target at 40% with a range between -30% and 30%, meaning that any rate between 10% and 70% utilization is deemed to be acceptable, with a rate closer to 40% being better.

The benefits of the proposed machine scoring approach may be summarized as follows:

- Any number of resources can be used (CPU, memory, network, disk, etc..) depending on specific need or for general machine usage and any number of aggregations can be used for each resource (average, p95, p50, etc..). More aggregates allow a more specific machine-workload configuration. We discuss this in further detail in the “[Discussion](#)” section.
- Optimal machines are defined as ones falling within the defined acceptable range, but any range can be used depending on need.
- The WISE framework can be used to train models on machines that have scores above a certain threshold for better performance, as the models would be trained on machines that are running between defined ideal levels.
- Workload Validation is more thorough, as the WISE score allows evaluation of machines where utilization data exists from running a specific workload. This is demonstrated in the “[Validation](#)” section.

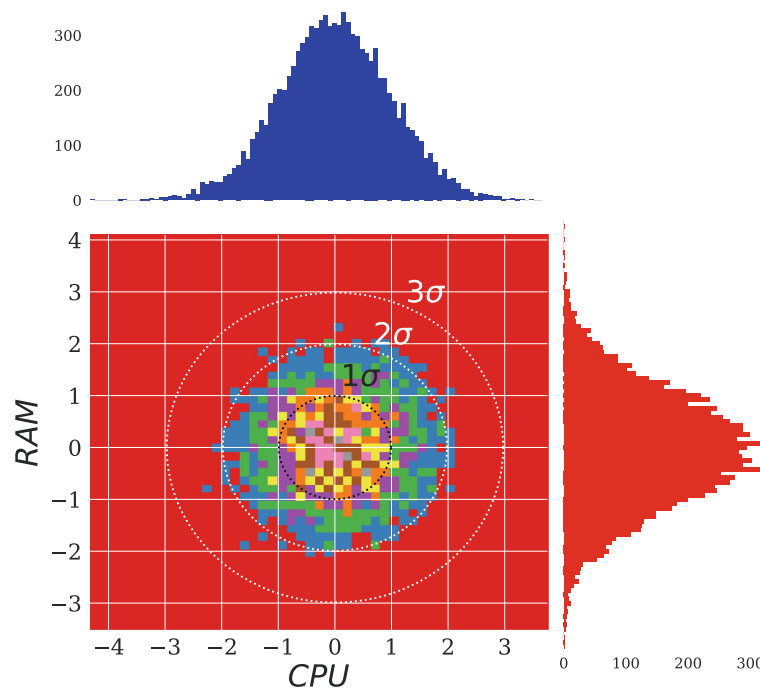


Fig. 1 Illustration of machine scoring in two dimensions (CPU and memory), with machines closer to the targets having a value closer to 1

- Users can define their own ideal running machine/resource rates and ranges, and subsequently evaluate their machines using the WISE score.
- Getting a WISE score pre and post configuration (vm or machine) change will validate the benefits with clear indicators.
- The individual resources' scores give an indication of how a resource is running with respect to the defined targets and ranges. For instance, if a machine has a low score, then we can look at the individual resources' scores and locate the origin of the problem.

Global vs Machine Specific: The running levels described next can be global or machine specific. Global resource running levels are applied to all machines, while machine specific types are applied only the that specific instance type. The machine specific defined running levels override the global levels for that machine specific type.

Setting targets and ranges

How to set good target and ranges for the different computing resources is one of the challenging issues to tackle while looking at computing performance given different workloads. The answer to what are acceptable performance levels will depend heavily on the use case [13, 14]. The targets and ranges are set after careful consideration on what the performance expectations are for the given a workload and for what the tolerance level of computing waste and/or unexpected workload changes are. Varying use cases may have different targets and ranges depending on tolerance. For instance, non-critical workloads have higher targets and ranges, as a slowdown due to an unexpected spike is not damaging (e.g. an online library or non-mission critical systems). In addition, critical workloads add more leeway in their targets and ranges to account for unexpected spikes in order to make sure there is no degradation in workload performance (e.g. in real time systems or mission critical systems).

Algorithmic steps

In this subsection, we describe in detail the main algorithmic steps of the proposed WISE framework.

Ideal resource running levels

In this first step, we define the ideal running levels for each resource r_i , where $i = 1, \dots, n$ and n is the total number of resources. More precisely, for each resource r_i (e.g. average memory utilization), we define the following parameters:

- The ideal target running utilization level μ_i for the given resource (r_i).

- The acceptable deviation levels σ_i from the ideal target level μ_i for the given resource r_i .
- An upper limit r_i^{\max} for the given resource r_i . Resources running above this level will be penalized with a zero score.
- If there is no ideal target running utilization level μ_i for the given resource r_i , but there is an upper limit r_i^{\max} , set the target and range to 0. Resources running above this level will be penalized with a zero score, otherwise the resource will have no affect on the score.

As an example, if we consider the average memory utilization as a resource with parameters $\mu = 50\%$, $\sigma = 30\%$ and $r_i^{\max} = 90\%$, then the average memory utilization running between 20% and 80% would be acceptable, with levels running closer to the defined target having a higher score. The further the running level is from the target level, the worse the score will be for that resource. Machines having a running level above the r_i^{\max} (e.g. 90% or above) would have the worst possible score for that resource.

Figure 2 (top) displays a normal distribution with target $\mu = 0$ and standard deviation σ equal to 1. It shows how a target level and ranges can be defined for resources. The darker blue area indicates ideal running levels, while the lighter blue indicates substandard running levels. Figure 2 (bottom) shows a plot of normal distributions with varying target levels and standard deviations.

Deviations from ideal running levels

In this second step, we define the resource z-score (also known as standard score) as

$$z_i = \frac{x_i - \mu_i}{\sigma_i}, \quad i = 1, \dots, n, \quad (1)$$

where x_i is the resource utilization rate, which is the actual utilization rate for a given resource r_i . The resource z-score gives the number of standard deviations from the target μ_i for each resource r_i .

Continuing with the example above, if a machine has an average memory utilization running level of 80%, then its z-score is equal to 1, while a machine with a running level of 20% has a z-score of -1.

Resources scores

In this third step, we define a resource score s_i for each resource r_i using the hyperbolic tangent and the exponentially monotone functions. The resource score provides a normalized index value for each resource. The resource score gives an index value for each resource, where the utilization rate falls within a normalized range. The closer the utilization is to the target the better the score will be and the further it is away from the target the worse the score will be.

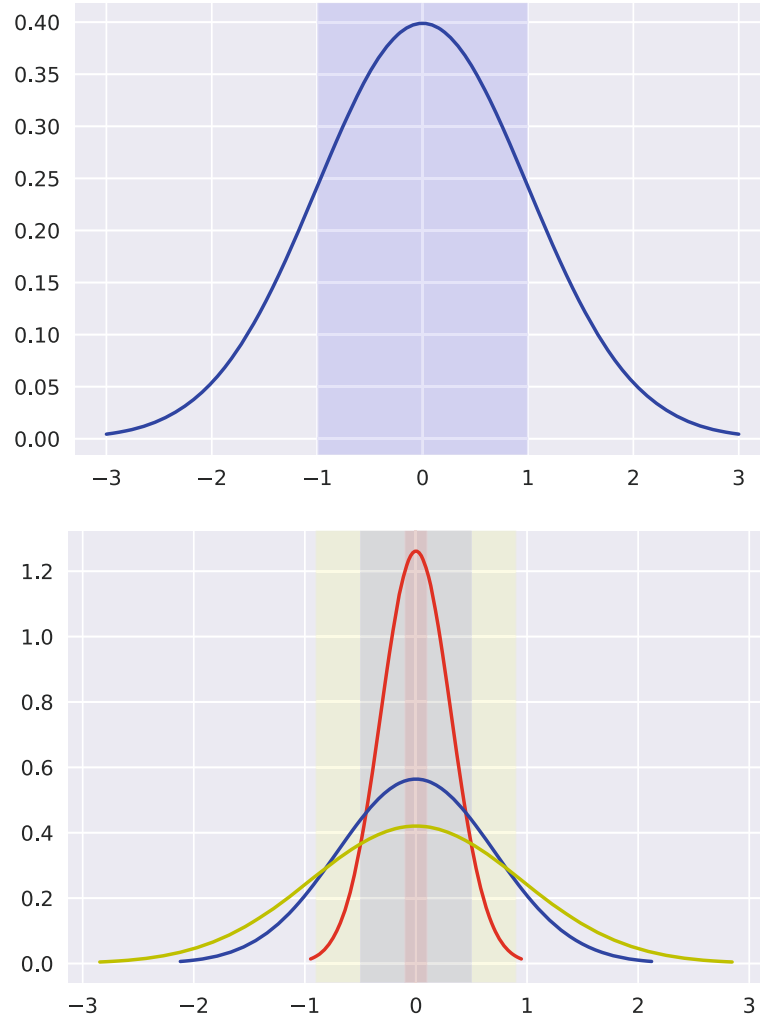


Fig. 2 Top: Defining a target and standard deviation. Bottom: Plot with different targets and standard deviations

Scoring functions: The scoring functions serve two main purposes: (1) Standardize the score to within a given level for both resources and the overall score. A scoring function converts the resource z-score to a predetermined range for all resources. This allows users of the WISE score for both resources and the overall score to examine the score without knowing a priori what the targets and ranges are; (2) Limit the negative influence that a sub-optimal performing resource can have on the overall score. The scoring functions limit the adverse effect that one resource can have on the overall score by smoothing the score when the resource z-score gets larger, as illustrated in Fig. 3. Such a situation may arise when a resource is running way off the target and range levels, resulting in a high z-score value. This would cause that specific resource and the overall machine to have sub-optimal WISE scores without a scoring function, when in fact if all other resources are running perfectly that one resource

should not determine the overall score outcome. To circumvent this limitation, we introduced a *penalty term* in the WISE framework to deal with the situation where a resource running above a certain level should cause the overall WISE score to be at a sub-optimal level to indicate potential issues that need to be addressed.

Using the hyperbolic tangent function: Given a resource z-score, we define the resource score as

$$s_i^{\tanh} = \tanh(z_i), \quad i = 1, \dots, n, \quad (2)$$

where $\tanh(\cdot)$ is the hyperbolic tangent function, which is commonly used as an activation function in neural networks and it produces outputs in the scale of $[-1, 1]$, as shown in Fig. 3 (top). A negative value indicates a resource or machine utilization rate below the ideal rate, while a positive value indicates a utilization rate above the target rate. A value of 0 indicates that the resource is running at

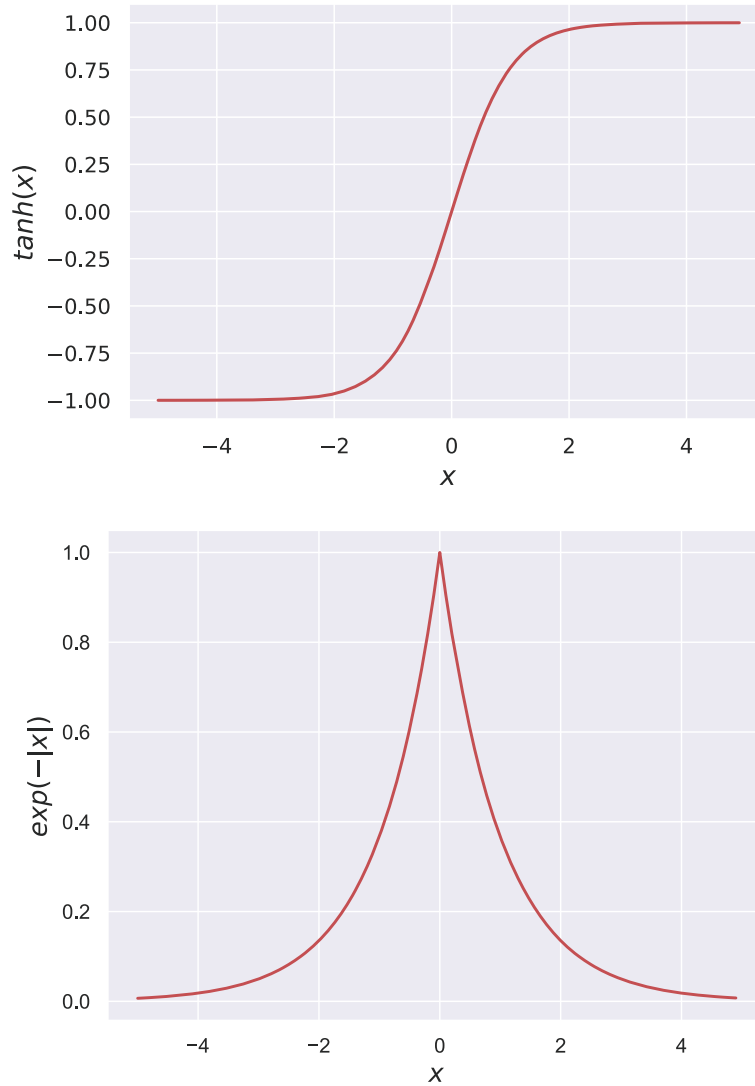


Fig. 3 Top: Hyperbolic tangent function. Bottom: Exponentially monotone function

the target level. A value of -1 indicates a resource that is at the extreme of under-utilization and a value of +1 indicates a resource at the extreme of over-utilization. Given this range of values for both the computing resources and an overall value for the machine allows customers to diagnose possible issues with the machine and/or resource usage.

The interesting aspect of using the hyperbolic tangent function to compute the resource score is that a negative score indicates an under-utilized resource with respect to the ideal target level, while a positive score indicates an over-utilized resource with respect to the ideal target level. This indication for each resource can be very helpful in diagnosing various issues with the computing machine and/or resources.

Using the exponentially monotone function: For each resource z -score, we define the resource score as

$$s_i^{\text{exp}} = \exp(-|z_i|), \quad i = 1, \dots, n, \quad (3)$$

where $\exp(-|t|)$ is the exponentially monotone function shown in Fig. 3 (bottom). This function produces outputs in the scale of $[0, 1]$, where the best score is 1 and the worst is 0. The interesting aspect of this score is that the best value would be 1 or close to one and the worst is 0, which is quite intuitive.

Notice that using the absolute value of z_i , the output of the exponentially monotone function is between 0 and 1. A value of 1 indicates that the resource is running at the target level. A value of 0 indicates a resource that is at the extreme of under- or over-utilization.

Resource weight: It is important to note that not all resources have equal importance on the computing environment. For instance, CPU and memory might have more influence on how well or bad a machine is behaving. This can also vary depending on the workload demands and goals on the machine. To this end, a predetermined weight w_i is assigned to a given resource score s_i . This weight can be used if we notice that certain computing resources, such as memory and CPU, are more important to the health of a machine than for example networking or disk. When no weight is assigned, all resources have equal weight in the resource score.

Resource penalty

In this fourth step, we introduce a resource penalty term, which affects the machine score when a given resource surpasses a maximum threshold. The reason behind using a penalty term is largely due to the fact that when some resources are stretched to a certain maximum level, the whole machine suffers and becomes unusable. We want to penalize resources running above an upper limit r_i^{\max} for the given resource so that the machine score is negatively influenced. To this end, we first subtract the upper limit r_i^{\max} from the resource running level x_i to get a positive value if the resource is running above the resource upper limit, 0 if it is equal to it and negative if it is running below it, i.e.

$$\text{sgn}(x_i - r_i^{\max}) = \begin{cases} -1 & x_i < r_i^{\max} \\ 0 & x_i = r_i^{\max} \\ 1 & x_i > r_i^{\max} \end{cases} \quad (4)$$

where $\text{sgn}(\cdot)$ is the sign function.

We define the penalty term for each resource as

$$\mathcal{P}(x_i) = H(x_i - r_i^{\max}), \quad i = 1, \dots, n, \quad (5)$$

where

$$H(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \quad (6)$$

is the Heaviside function (also referred to as unit step function). It is worth pointing out that the sign and Heaviside functions are related via the identity: $\text{sgn}(t) = H(t) - H(-t)$.

The penalty term returns a value of 0 or 1 depending on whether that resource is running above or below the pre-defined max utilization rate r_i^{\max} . If the resource is not above the resource limit, then the penalty term has a value of 0, and hence it does not have any affect on the resource score. On the other hand, if the resource utilization is above the resource limit, then the penalty term for that resource is equal to 1.

Using nonnegative penalty weight factor α for each resource r_i , we define the weighted penalty term for each resource as

$$\mathcal{P}_\alpha(x_i) = \alpha H(x_i - r_i^{\max}), \quad i = 1, \dots, n, \quad (7)$$

where an α value of 1 sets the resource penalty term to 1 for values greater than or equal to 1 and 0 otherwise. An α value smaller than 1 diminishes the affect of the penalty term, while an α value larger than 1 increases the affect of the penalty term. A high value for α ensures that a machine with a resource that is running at levels over the r_i^{\max} receives the worst possible machine score.

Workload/Machine index score

In this last step, we introduce four variants of the proposed WISE score using the hyperbolic tangent and exponentially monotone functions in conjunction with the weighted ℓ_1 - and ℓ_2 -norms. The WISE score gives a good indication on how well the computing machine is running given a specific workload.

Using tanh function and ℓ_1 -norm: We define the WISE score as

$$S_1 = \min \left[\frac{1}{n} \sum_{i=1}^n w_i |\tanh(z_i)| + \sum_{i=1}^n \mathcal{P}_\alpha(x_i), 1 \right], \quad (8)$$

where the minimum function is used to assure that a value greater than 1, which is the worst score on the positive side, is not returned even when more than one resource utilization rate falls above its upper limit rate.

The penalty term adds a value depending on whether or not any of the resource utilization rates are above their respective upper limit rate. If none of the resources is above its upper limit rate, then the penalty term is 0 and hence it has no affect on the WISE score. On the other hand, if there is at least one or more resources that are above their respective upper limit rates, then the penalty term has a value greater than 1, which adversely affects the machine score, indicating over-utilization. The WISE score has a value between 0 and 1, with a value of 0 being the best.

Using tanh function and ℓ_2 -norm: We define the WISE score as

$$S_2 = \min \left[\frac{1}{n} \sqrt{\sum_{i=1}^n (w_i \tanh(z_i))^2} + \sum_{i=1}^n \mathcal{P}_\alpha(x_i), 1 \right], \quad (9)$$

which returns values between 0 and 1, with a value of 0 being the best.

Using exponentially monotone function and ℓ_1 -norm: We define the WISE score as

$$S_3 = \max \left[\frac{1}{n} \sum_{i=1}^n w_i e^{-|z_i|} - \sum_{i=1}^n \mathcal{P}_\alpha(x_i), 0 \right], \quad (10)$$

Table 1 Parameters used in the experiments and validation

Resource (r)	Target (μ)	Range (σ)	Weight (w)	Resource Max (r^{\max})	Penalty Weight (α)
CPU/Avg	40%	30%	1	N/A	1
CPU/P95	70%	20%	1	N/A	1
RAM/Avg	50%	20%	1	90%	1
RAM/P95	70%	30%	1	N/A	1
Network/Avg	N/A	N/A	1	80%	1

where the maximum function is used to assure that the machine score is nonnegative, with a value of 0 being the worst score. The exponentially monotone function returns a value between 0 and 1 with 1 being the best, while the penalty term returns a value between 0 and the number of resources n times the parameter α , depending on the number of resource utilization rates that fall above the upper limit rate.

Using exponentially monotone function and ℓ_2 -norm:
We define the WISE score as

$$S_4 = \max \left[\frac{1}{n} \sqrt{\sum_{i=1}^n (w_i e^{-|z_{il}|})^2} - \sum_{i=1}^n \mathcal{P}_\alpha(x_i), 0 \right], \quad (11)$$

Experiments

In this section, we conduct extensive experiments by running a two distinct workloads on multiple Amazon AWS EC2¹ instances using the utilization data generated to compare the WISE scores for each instance. We evaluate the WISE score on two benchmarks, a MongoDB workload which is a cpu intensive workload and a second Streaming workload which is a networking intensive workload. To account for bursty workloads in the “Validation” section, we add a new third workload which is a reconfiguration of the MongoDB workload to have bursts of activity with periods of less activity.

Experimental setup

In all the experiments, we set the penalty weight factor α to 1 and used a uniform resource weight. We considered five resources (r) with target resource utilization levels (μ), acceptable deviation levels (σ) and upper limits (r^{\max}) as described in Table 1, which describes all the parameters that are used for the WISE score calculation. The optimal thresholds are set to the levels for one range (σ) deviation from the target (μ) as described in Table 2, although these can vary depending on a users use case. The table describes thresholds used for both functions (TanH and Exp.), which are used to determine if the workload/machine is performing within an acceptable range.

Results

In this subsection, we demonstrate the performance of our proposed WISE framework on two distinct workload configurations. Figure 4 displays the WISE score and resource scores for a MongoDB workload using the hyperbolic tangent function and ℓ_1 -norm, with the best score having a value of 0. The area between the light blue dotted lines indicate values that fall within the described acceptable ranges. Any score that falls outside of this region indicates over- or under-utilization from the acceptable ranges. A machine score can still fall within the acceptable range while having a resource that falls out of range. As average network utilization has only an upper limit (penalty), it can only affect the machine score when the utilization surpasses this limit. This is displayed in Fig. 4 by showing a dot on 0 when it has no affect. Figure 5 displays the same data as in Fig. 4, except that the overall machine score is computed using the hyperbolic tangent function and ℓ_2 -norm.

Figure 6 also displays the WISE score and resource scores for a MongoDB workload using the exponentially monotone function and ℓ_1 -norm, with the best score having a value of 1. The area above the light blue dotted lines indicate values that fall within the described acceptable ranges. Any value that falls below this region indicates over- or under-utilization from the acceptable ranges. A machine score can still fall within the acceptable range while having a resource that falls out of range. Figure 7 displays the same data as in Fig. 6, except that the overall machine score is computed using the exponentially monotone function and ℓ_2 -norm.

Figure 8 displays the WISE score and resource scores for a Streaming workload. The normalization function using the hyperbolic tangent function and ℓ_2 -norm, with the best score having a value of 0. The area between the

Table 2 Optimal thresholds used in the experiments and validation

Function	WISE Score	Threshold	Best Score
TanH	Resource	$-0.76 \leq r \leq 0.76$	0
TanH	Overall	$r \leq 0.76$	0
Exp.	Resource	$r \geq 0.36$	1
Exp.	Overall	$r \geq 0.36$	1

¹<https://aws.amazon.com/ec2>

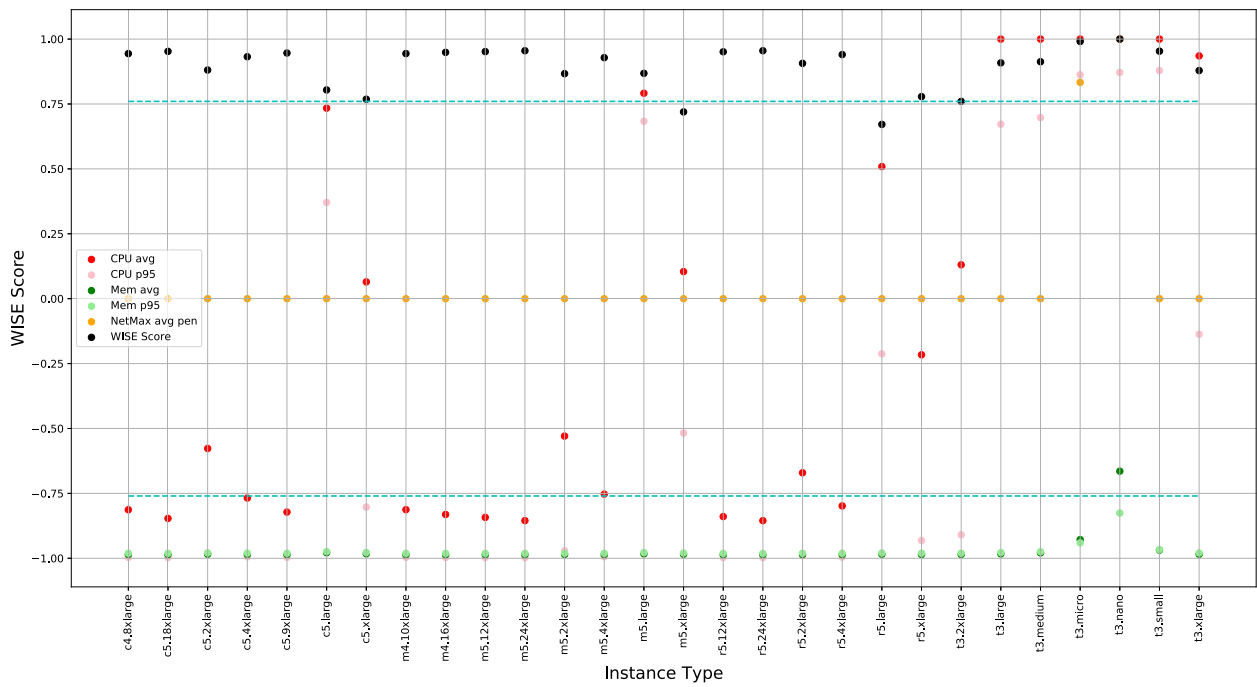


Fig. 4 WISE scores using the hyperbolic tangent function and ℓ_1 -norm for MongoDB Workload

light blue dotted lines indicates values that fall within the described acceptable ranges. Any value that falls outside of this region indicates over- or under-utilization from the acceptable ranges. A machine score can still fall within the acceptable range while having a resource that falls out of

range. As average network utilization has only an upper limit (penalty), it will only affect the machine score when the utilization goes above this limit. The plot displays this by showing a dot on 0 when it has no affect. Figure 9 displays the same data as in Fig. 8, except that the overall

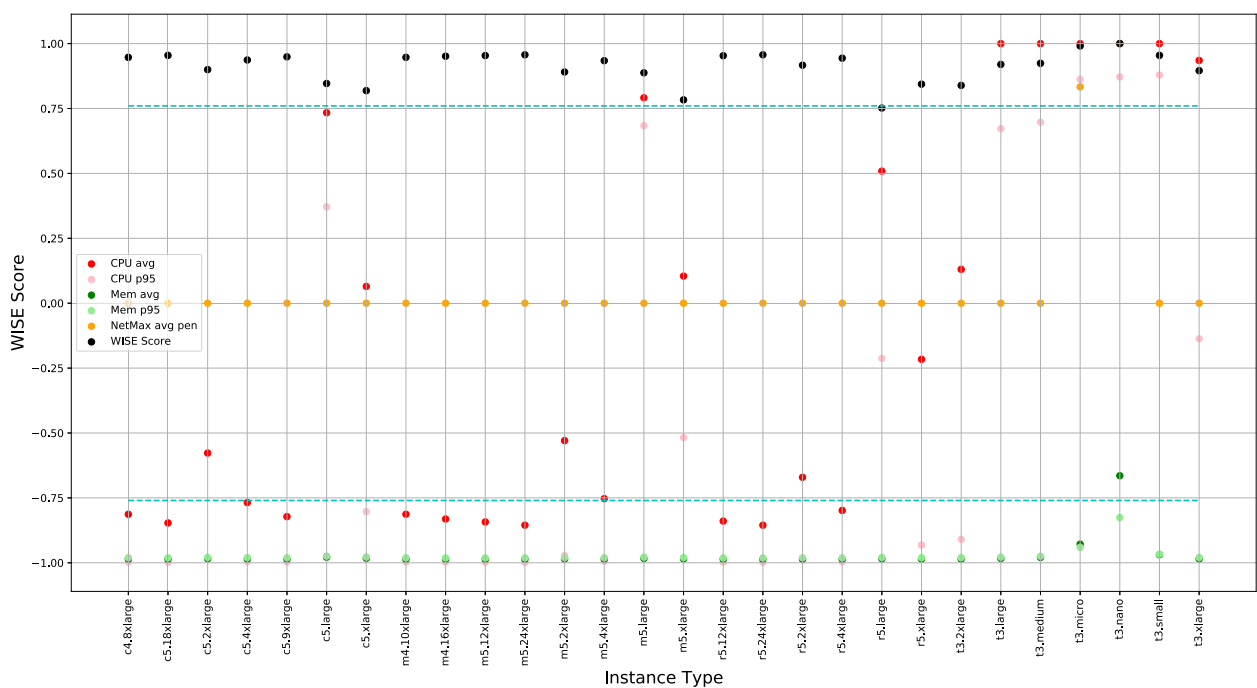
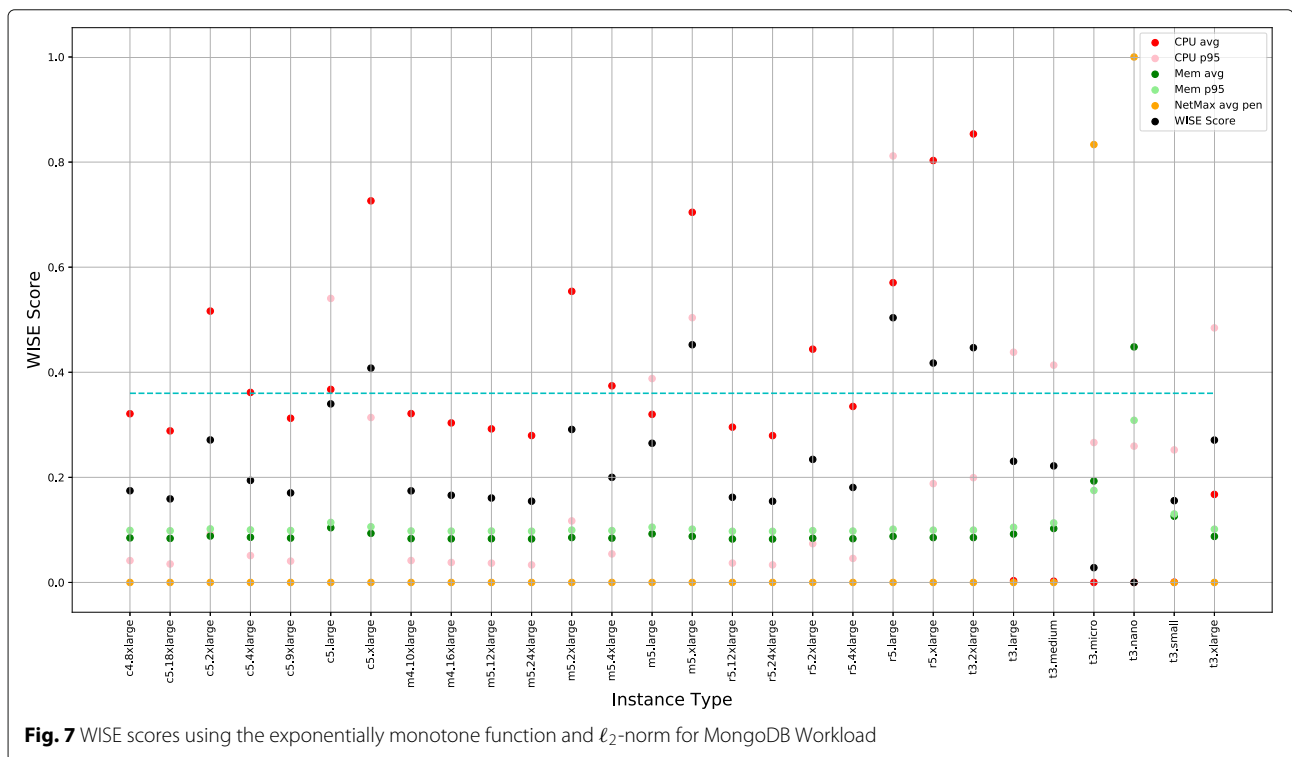
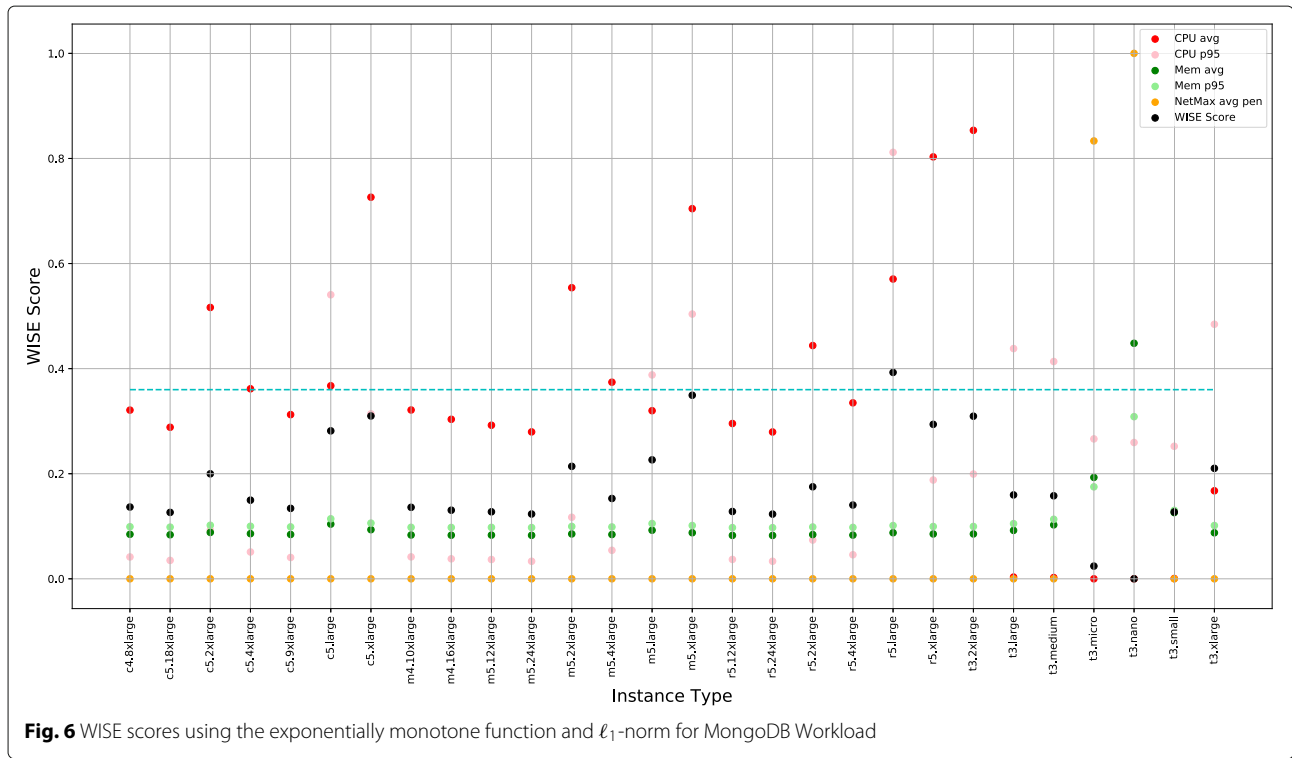
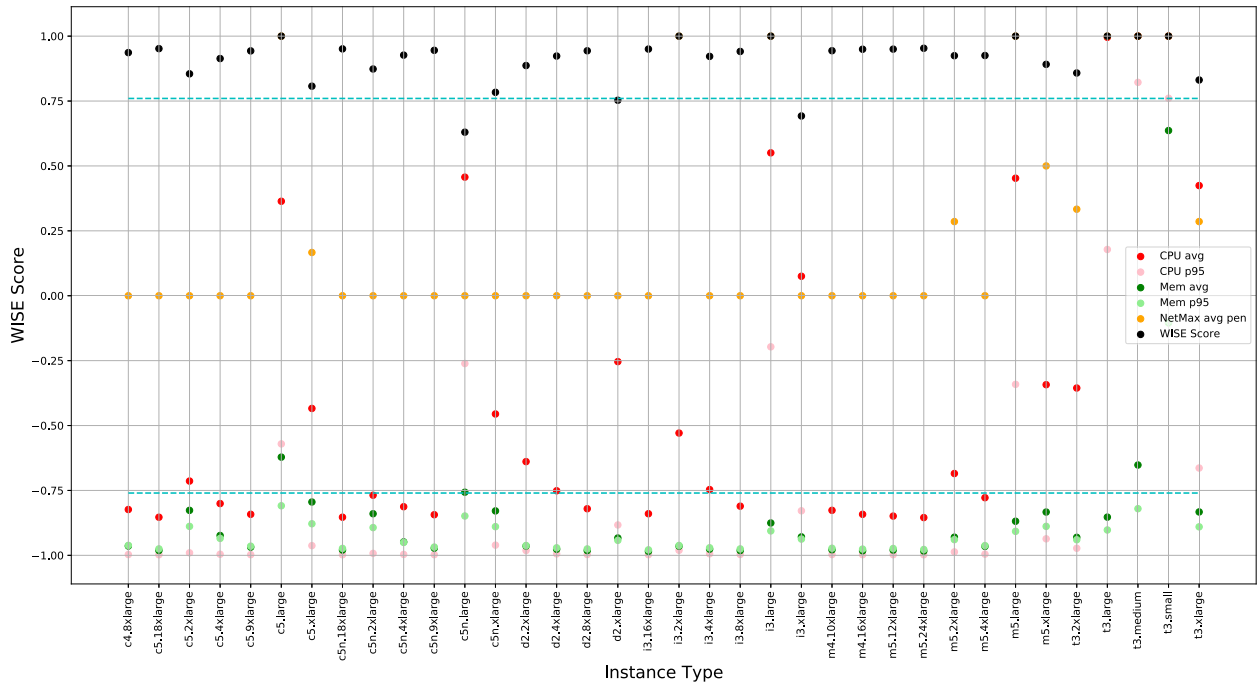
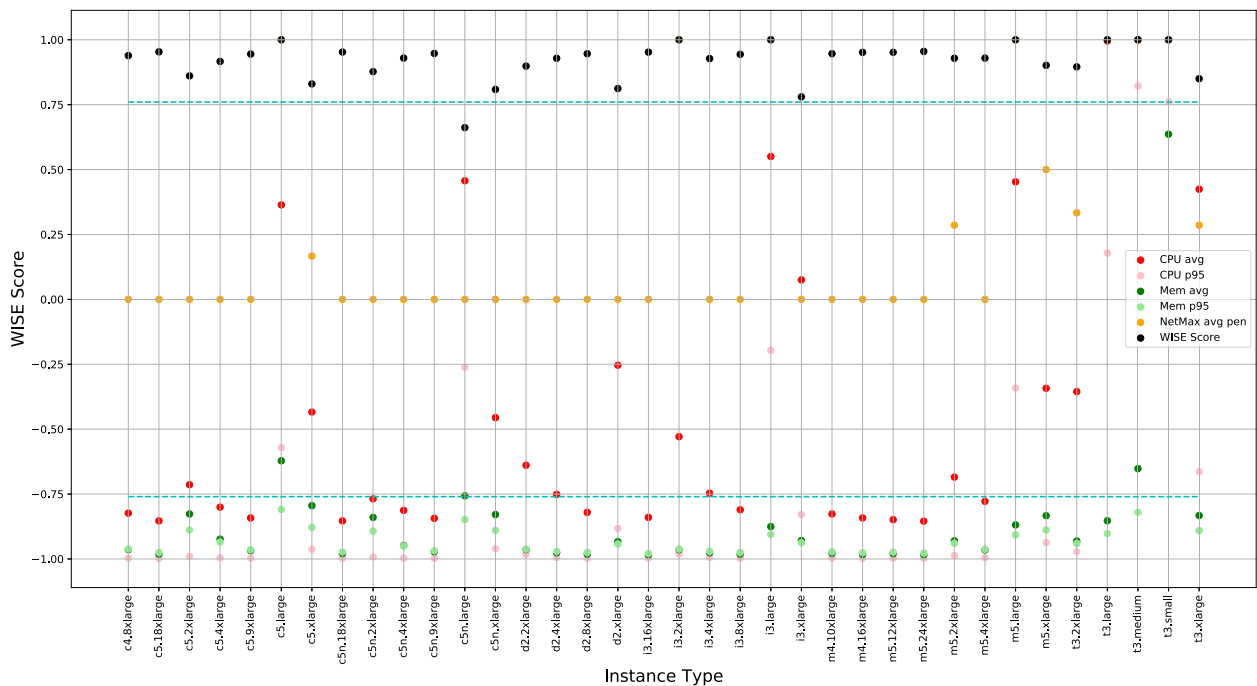


Fig. 5 WISE scores using the hyperbolic tangent function and ℓ_2 -norm for MongoDB Workload



**Fig. 8** WISE scores using the hyperbolic tangent function and ℓ_1 -norm for Streaming Workload**Fig. 9** WISE scores using the hyperbolic tangent function and ℓ_2 -norm for Streaming Workload

machine score is computed using the hyperbolic tangent function and ℓ_2 -norm.

Figure 10 displays the WISE score and resource scores for a Streaming workload, except the normalization function using the exponentially monotone function and ℓ_1 -norm, with the best score having a value of 1. The area above the light blue dotted lines indicates values that fall within the described acceptable ranges. Any score that falls below this region indicates over- or under-utilization from the acceptable ranges. A machine score can still fall within the acceptable range while having a resource that falls out of range. Figure 11 displays the same data as in Fig. 10, except that the overall machine score is computed using the exponentially monotone function and ℓ_2 -norm.

Validation

In this section, we describe and present results for a unique validation method that uses performance data from the benchmarks to validate the efficacy of the WISE score to determine well performing workload/machine combinations.

Method

We use two benchmarks to validate the WISE score 1) MongoDB, a CPU intensive benchmark. Configured to run steadily over time, 2) MongoDB, configured to run bursty with spikes of activity and 3) Streaming workload, a networking intensive benchmark. We first determine the optimal instances by using the performance metrics

that are generated by running a specific workload on many different configurations of virtual machines. Some of the performance metrics used are duration, latency and throughput. It is important to note that during this phase of validation the WISE score does not come into play and neither does the utilization data. Optimal virtual machines are determined solely on the performance metrics of the benchmarks. We then compare the optimal virtual machine selected by using the performance metrics with the optimal virtual machines generated by using the WISE score using utilization data to determine if indeed, the WISE score is a good indicator of workload/machine performance.

Performance metrics

Using the performance metrics generated for each workload/machine combination by the benchmark, the following criteria is used to determine optimal instances for a specified workload. First, using the statistical inter-quartile range outlier detection method, remove any data points that have a latency or duration of greater than $Q3 + 1.5 * IQR$. This removes any instances that are taking longer to execute with respect to duration and latency, they are outside of the range of most other instances. Second, the remaining instances are sorted by usage cost and the instances that fall within 3 times the usage cost of the cheapest instance (after part one) are selected. The first part removes instances that are under-provisioned and the second part removes instances that are over-provisioned.

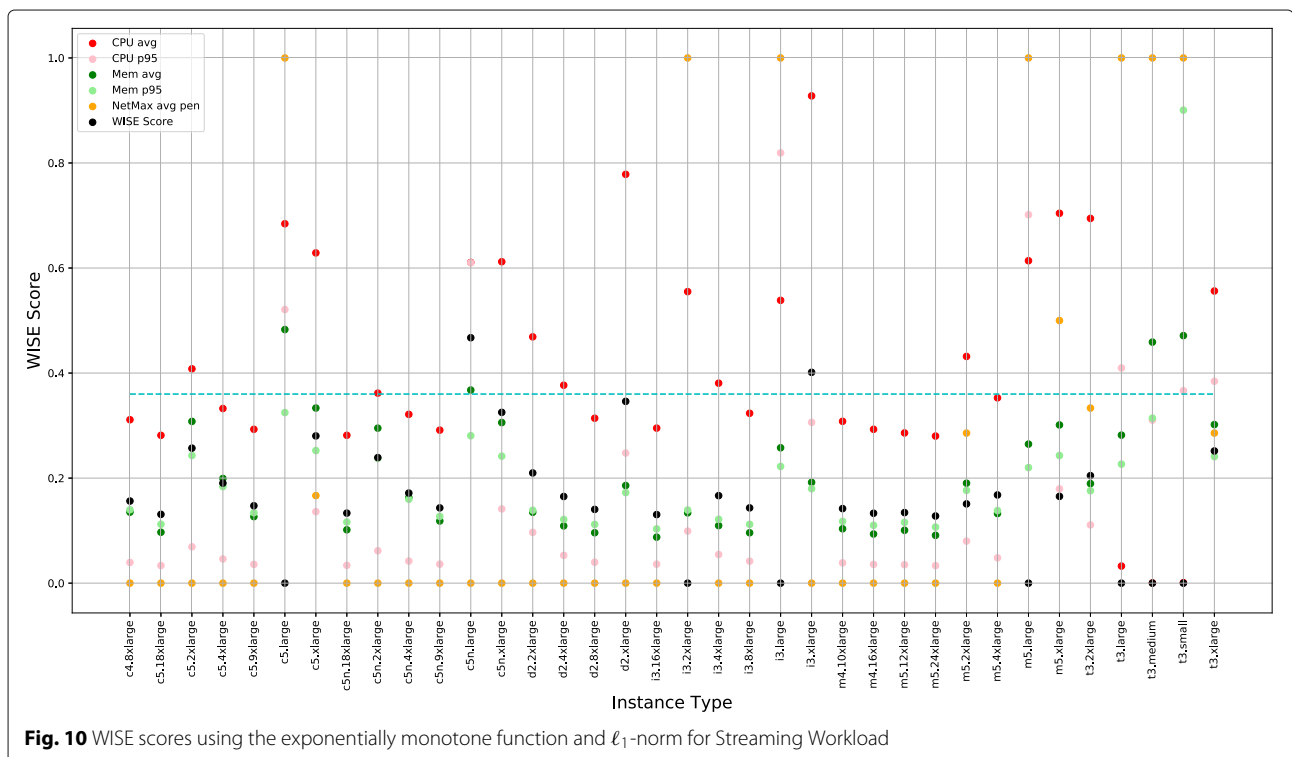


Fig. 10 WISE scores using the exponentially monotone function and ℓ_1 -norm for Streaming Workload

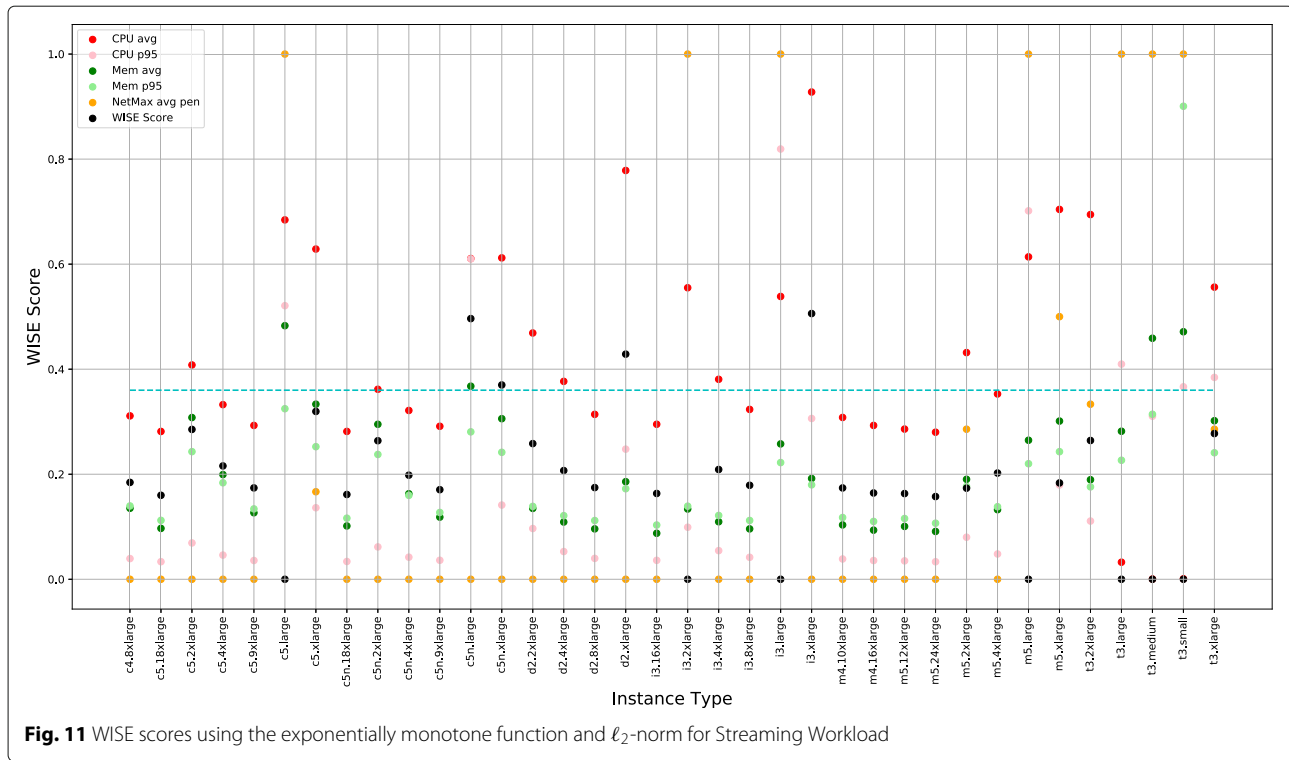


Fig. 11 WISE scores using the exponentially monotone function and ℓ_2 -norm for Streaming Workload

Resulting in instances that are capable of handling the load without being over-provisioned.

WISE score

During this phase, the WISE score is calculated for each workload/machine combination. So for every instance that the benchmark is run on, we take the utilization data generated and get a WISE score. None of the performance metrics used in the first phase are used to calculate the WISE score. The WISE score only uses utilization data, no specific metrics from the workload such as latency, duration are used in the WISE Score. We use the Streaming workload to validate the WISE score, by first coming up with optimal instances using the performance metrics generated by running the workload on many instances such as latency and duration. We then compare the optimal instances generated from the performance metrics with the optimal instances generated by the WISE score using various standard evaluation metrics.

WISE score - quality tenets

The quality tenants for the proposed WISE score can be summarized as follows:

1. Instances with a WISE score above threshold should be able to run the workload with acceptable performance. The precision metric is used as an

indicator of acceptable performance, indicating percentage of returned machines the run optimally according to performance metrics.

2. How to account for acceptable performing instances that are under-utilized? Although a machine has good performance metrics, it is possible that it is over provisioned for that specific workload. Therefore, from the list acceptable performing instances, only return those that are within 2 times the price of the cheapest instance.
3. How to account for acceptable performing instances that are over-utilized? Although a machine has good performance metrics, it is possible that it is over-utilized for the specific task, e.g. running very close to its capacity limits, but have not yet a resource wall yet. Use a tighter outlier cut-off point on the performance metrics. By using a tighter outlier cut-off point we will eliminate any instance types that may be close to that resource utilization wall. By looking at only the performance metrics and not the utilization data, we are not fully able to determine this, however, the solutions mentioned above mitigate some of the issues.
4. Use a ranking metric to determine how well the WISE score rankings compare to the ranked performance based list sorted by price. Note that this is not perfect as the WISE score does not take price

into account, and does not rank a cheaper instance higher. It will rank higher, an instance that runs within acceptable utilization ranges. For example in the performance metrics a cheap instance will be ranked higher even if it is over-utilized.

5. How well does the WISE score identify well performing instances at reasonable prices? Reasonable prices here would be defined as in a range of 2/3 times the cheapest well performing instance. We use recall to determine how many of these.

Results

Using the performance metrics and methods described above, for each workload we get a list of optimal performing virtual machines. The list derived by using the WISE scores will be compared to this optimal list. The precision and recall metrics are used along with a rank based metric, rank biased overlap [15] to evaluate the ordering of WISE scores. We used a standard cutoff of 0.36 for the exponential function and 0.76 for the tangent function. Results show that the WISE score consistently identifies optimal instance types for a workload, see Table 3 and the precision metric. Recall shows how many of the optimal instance types is the WISE score able to identify, the exponential function with the ℓ_2 performs best with this metric. The ranking metric shows how the WISE score orders the instance types by score. All of these metrics can be tweaked by changing the acceptable threshold parameter.

Discussion

The WISE framework may suffer from the curse of dimensionality if irrelevant dimensions (resources) are added. As less important resources are added, WISE can become inefficient as the more important resources become diluted by the less important ones. To circumvent

this issue, only necessary resources and aggregations that have a certain amount of information gain should be used, as the impact of one resource is diminished by the number of resources given, this can also be controlled by the weight factor. In essence it is important to add resources and aggregations that add value in determining performance. Moreover, it would be interesting to do this automatically by computing the information gain of each attribute and using only the most informative ones. This could be accomplished by first discarding all attributes whose information gains are below a pre-defined threshold and then measuring distance only in the reduced space.

We have designed the WISE score to be very flexible in its configuration possibilities. We have observed different configuration use cases and tolerances for over- and under-provisioning. Also, certain types of machines such as scientific supercomputers or GPU intensive computers can have different configuration levels. It is indeed a challenging issue to find those optimal values, and the existing baselines vary based on the use cases. There are various methods to set these levels, including through observation and expert knowledge, as well as understanding the needs of a workload, awareness of a business needs and cycles. By contrast, the proposed method selects a group of optimally running machines and trains the WISE model to output optimal scores for these machines. That configured WISE model can then be used to get tuned WISE scores for other machine/workload combinations.

Conclusion

In this paper, we proposed a novel approach for scoring a workload/machine combination representing the fitness of a machine running a particular workload. The WISE framework is powerful in that it produces an index between 0 and 1, indicating the level of fitness for the workload/machine combination. It is flexible in that

Table 3 WISE Score validation results using benchmark performance metrics

Benchmark	Function	Norm	Precision (%)	Recall (%)	Ranking
MongoDB (steady)	Tanh	ℓ_1	1.0	0.667	0.699
		ℓ_2	1.0	0.333	0.51
	Exp.	ℓ_1	1.0	0.333	0.51
		ℓ_2	0.858	1.0	0.897
MongoDB (bursty)	Tanh	ℓ_1	1.0	0.778	0.806
		ℓ_2	1.0	0.556	0.564
	Exp.	ℓ_1	1.0	0.556	0.564
		ℓ_2	0.889	0.889	0.849
Streaming	Tanh	ℓ_1	1.0	0.667	0.704
		ℓ_2	1.0	0.333	0.842
	Exp.	ℓ_1	1.0	0.667	0.704
		ℓ_2	0.4	0.667	0.575

Boldface numbers indicate the best evaluation metrics

customers can define individualized targets and ranges to suit their needs and then use the WISE score to test their fleet of machines. WISE also allows for general definitions of proper machine running levels to very sophisticated resource definitions. Experimental results showed the efficacy of the proposed framework on two distinct workload configurations. Validation results showed that the WISE score was able to deliver optimal instance types on three different benchmark configurations. For future work, we plan to learn the resources' weights given some ground truth data and then use the WISE framework with the learned weights to compute the WISE scores

Acknowledgements

We acknowledge the support of Amazon AWS for this research.

Authors' contributions

The authors have contributed equally to the research and writing of this paper. The author(s) read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

¹Amazon, AWS, Boston, MA, USA. ²Amazon, Alexa AI-Natural Understanding, Cambridge, MA, USA. ³Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada.

Received: 17 June 2020 Accepted: 16 December 2020

Published online: 20 January 2021

References

1. Jackson KR, Ramakrishnan L, Muriki K, Canon S, Cholia S, Shalf J, Wasserman HJ, Wright NJ (2010) Performance analysis of high performance computing applications on the Amazon web Services cloud. In: 2010 IEEE second international conference on cloud computing technology and science. IEEE. pp 159–168
2. Alipourfard O, Liu HH, Chen J, Venkataraman S, Yu M, Zhang M (2017) Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics. In: 14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI}) 17. pp 469–482
3. Koh Y, Knauerhase R, Brett P, Bowman M, Wen Z, Pu C (2007) An analysis of performance interference effects in virtual environments. In: 2007 IEEE International Symposium on Performance Analysis of Systems & Software. IEEE. pp 200–209
4. Khan A, Yan X, Tao S, Anerousis N (2012) Workload characterization and prediction in the cloud: A multiple time series approach. In: 2012 IEEE Network Operations and Management Symposium. IEEE. pp 1287–1294
5. Hsu C-J, Nair V, Menzies T, Freeh V (2018) Micky: A cheaper alternative for selecting cloud instances. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). IEEE. pp 409–416
6. Pietri I, Sakellariou R (2016) Mapping virtual machines onto physical machines in cloud computing: A survey. *ACM Comput Surv* 49(3):1–30
7. Yadwadkar NJ, Hariharan B, Gonzalez JE, Smith B, Katz RH (2017) Selecting the best vm across multiple public clouds: A data-driven performance modeling approach. In: Proceedings of the 2017 Symposium on Cloud Computing. pp 452–465
8. Rjoub G, Bentahar J, Wahab OA (2020) BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments. *Futur Gener Comput Syst* 110:1079–1097

9. Rjoub G, Bentahar J, Wahab OA, Bataineh A (2019) Deep smart scheduling: A deep learning approach for automated big data scheduling over the cloud. In: 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE. pp 189–196
10. Mishra AK, Hellerstein JL, Cirne W, Das CR (2010) Towards characterizing cloud backend workloads: insights from Google compute clusters. *ACM SIGMETRICS Perform Eval Rev* 37(4):34–41
11. Downey AB, Feitelson DG (1999) The elusive goal of workload characterization. *ACM SIGMETRICS Perform Eval Rev* 26(4):14–29
12. Calzarossa M, Serazzi G (1993) Workload characterization: A survey. *Proc IEEE* 81(8):1136–1150
13. Lilja DJ (2005) Measuring computer performance: a practitioner's guide. Cambridge university press
14. Henning JL (2000) SPEC CPU2000: Measuring CPU performance in the new millennium. *Computer* 33(7):28–35
15. Webber W, Moffat A, Zobel J (2010) A similarity measure for indefinite rankings. *ACM Trans Inf Syst (TOIS)* 28(4):1–38

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)