AN EFFICIENT DECISION TREE BASED APPROACH FOR ADAPTIVE MODELING IN BIG DATA APPLICATIONS

M Kiran Kumar[1] Raju Korandla[2] SNVASRK Prasad[3] China Guravaiah Makkena[4]
[1234]Assistant Professor, Dept of Computer Science and Engineering, SICET, Ibrahimpatnam, Hyd.

Abstract—

The approach of the Big Data time has brought forth an assortment of new designs going for applications with expanded versatility, strength and adaptation to non-critical failure. In the meantime, these models have entangled application structure, prompting exponential development of their setup space and expanded trouble in anticipating their execution. In this work, we depict a novel, robotized profiling system that makes no presumptions on application structure. Our approach uses slanted Decision Trees so as to recursively parcel an application's setup space in disjoint districts, pick a lot of delegate tests from each sub-area as indicated by a characterized arrangement and restore a model for the whole space as an organization of straight models over each sub-locale. A broad assessment over genuine applications and engineered execution capacities grandstands that our plan beats other best in class profiling approaches. It especially exceeds expectations at reflecting anomalies and discontinuities of the execution work, just as distinguishing the parameters with the most astounding effect on the application's conduct.

Keywords: *Adaptive Modeling, Big Data, Decision Tree*

I. INTRODUCTION

Execution Performance modeling is an all around explored issue [1], [2], [3]. The ID of an application's conduct under various setups is a key factor for it to have the capacity to satisfy its targets. As the application scene develops, mostly because of the rise of the Big Data period, new models and configuration designs have empowered an expanding number of applications to be sent in a circulated way and advantage from the benefits of this approach: Scalability, vigor, and adaptation to internal failure are a portion of the properties that render appropriated stages appealing and clarify their wide selection. By goodness of their structure, appropriated applications are usually conveyed to cloud foundations [4], so as to join their natural qualities with the intensity of the cloud: Seemingly unending figure and capacity assets, powerfully designated and obtained, empower a Big Data application, i.e., an application that forms Big Data, to scale in a savvy way. Nonetheless, the appropriation of the conveyed worldview builds the intricacy of the application engineering: Many helping software modules that help coordination, bunch the executives, and so forth, are basic for the application to run legitimately. However, every module can be designed from multiple points of view. All things considered, the application setup space has endlessly extended. Subsequently, the issue of modeling an application execution likewise called an application profile, has turned out to be especially convoluted.

Clearly, the robotized estimation of an application profile would demonstrate very useful. The extent of the setup space renders comprehensive approaches that investigate a monstrous piece of the design space unreasonable since they involve both a restrictive number of organizations and a colossal measure of calculation. A few approaches focus to demonstrate application execution in a logical way [5], [6], [7], [8]. These approaches are viable for known applications with a particular structure and they are based on reproduction or copying systems [9], [10]. To defeat the inflexibility of these plans, different strategies (e.g., [11], [12], [13]) take a "discovery" approach, in which the application gets a lot of information sources, comparing to the distinctive elements of the design space and delivers at least one yields, relating to its execution. These approaches endeavor to recognize the connection between the info and the yield factors for a subset of the arrangement space (using testing) and sum up the discoveries with Machine Learning methods (modeling).

Such profiling approaches require different application organizations for unmistakable designs so as to efficiently investigate the arrangement space and catch its conduct. Notwithstanding, the arrangement utilized for choosing these designs is an unequivocal factor. This element of the profiling issue, however, isn't tended to by flow look into works, as a large portion of the recommended application profiling approaches accept an irregular setup determination arrangement, suggesting that all information measurements have a similar effect on the application's conduct, and go for limiting the modeling blunder through expanding the quantity of inspected designs. Along these lines, the work of an adaptive design choice arrangement, that admirably picks the tried setups regarding examining the most "delegate" application setups, would result in progressively exact application profiles with less tried setups, quickening the profiling procedure and limiting the individual expense.

In this work, we propose an adaptive approach to consequently create an application profile for any Big Data application with obscure structure, given a particular number of organizations, especially concentrating on fittingly choosing the tried setups so as to augment the modeling exactness. Our work handles the examining and modeling ventures unifiedly: First, we present an exactness driven testing method that favors areas of the arrangement space which are not precisely approximated. Second, we disintegrate the setup space in disjoint districts and use distinctive models to estimated the application execution in every one of them. The premise of our approach lies on the mechanics of Classification and Regression Trees [14] that utilization a recursive parceling of the application's setup space. Each segment is allocated with various setups to be sent, with the procedure being iterated until a pre-characterized most extreme number of test arrangements is come to. The quantity of setups apportioned at every locale is adaptively chosen by the guess mistake and the measure of each segment. At long last, the whole space is approximated by direct models per segment, based on the conveyed setups. Naturally, our approach endeavors to "zoom-in" to areas where application execution isn't precisely approximated, paying explicit enthusiasm to every one of the anomalies and discontinuities of the execution work. By using angled Decision Trees [15], we can catch designs that are influenced by different setup parameters at the same time. In this work we make the accompanying commitments:

We propose an adaptive, exactness driven profiling system for Big Data applications that uses slanted Decision Trees. Our strategy disintegrates the multi-dimensional info space into disjoint districts, normally adjusting to the mind boggling execution application conduct in a completely computerized way. Our plan uses three one of a kind highlights in respect to the standard Decision Tree calculation: First, it proposes a novel development calculation that builds sideways Decision Trees by looking at whether the acquired examples fit into a direct model. Second, it enables designers to give a bargain between investigating the setup space and misusing the recently gotten information. Third, it adaptively chooses the most precise modeling plan, based on the accomplished exactness.

We play out a broad test assessment over differing, certifiable applications and manufactured execution elements of different complexities. Our outcomes exhibit that our system is extremely compelling, accomplishing modeling correctnesses even 3× higher than its rivals and, in the meantime, having the capacity to make models that reflect anomalies and discontinuities of the execution work requests of size all the more precisely. Moreover, our inspecting technique turns out to be especially advantageous for direct classifiers, as straight models prepared with tests picked by our plan present up to 38% lower modeling mistake.

Application profiling can be detailed as a capacity estimate issue [16], [12]. The application is seen as a black-box that gets various information sources and delivers a solitary (or more) output(s). The principle thought behind building the execution demonstrate is to anticipate the connection between the information sources and the yield, without making any suspicion in regards to the application's engineering. Sources of info mirror any parameters influencing application execution, for example, the number and nature of various sorts of assets (e.g., centers/memory, number of hubs, and so forth.), application-level parameters (e.g., store utilized by a RDBMS, HDFS square size, and so forth.), remaining task at hand explicit parameters (e.g., throughput/kind of solicitations) and dataset-explicit parameters (e.g., measure, dimensionality, dispersion, and so forth.).

Accept that an application contains n information sources and one yield. We expect that the ith input, 1 = I = n, gets values from a predefined limited arrangement of qualities, signified as di. The Cartesian result of all di, 1 = I = n is the Deployment Space of the application D = d1 × d2 × • × dn. Likewise, the application's yield reflects values that relate to an execution metric, characteristic of its capacity to satisfy its destinations. The arrangement of the application's yield will be alluded to as the application's Performance Space. Based on the meanings of D and P, we characterize the execution display m of an application as a capacity m : D ? P . The estimation of the execution display involves the estimation of the execution esteem bi? P for every ai ? D. In any case, |D| increments exponentially with n (as |D| =identification of all execution esteems ends up restrictive, both as far as time and cost. A typical approach to handle this test is the extraction of a subset Ds ? D (|Ds| |D|) and the estimation of the execution focuses Ps for every ai ? Ds. Utilizing Ds and Ps, demonstrate m can be approximated, making an inexact model m . While m ? m, the guess is progressively precise.

Note that this plan is legitimate just under the supposition that unmistakable organizations are reproducible, i.e., for the situation where a given Deployment Space point is redeployed, the deliberate result is indistinguishable or if nothing else fundamentally the same as. For some reasons, for example, the obstruction [17], arrange glitches, and so on., such a suspicion can be abused when the application is sent to cloud conditions, due to the presented flightiness that contorts the application's conduct. The treatment of this component of the issue is outside the extent of our work. These works handle the multifaceted nature presented by the intemperate dimensionality of the Deployment Space. The introduced technique can be, along these lines, specifically connected to genuinely unsurprising conditions with diminished obstruction, for example, private cloud establishments that, as indicated by [18], remain incredibly famous, since they will have half of the client produced outstanding burdens for 2017, keeping up the pattern from the earlier years.

Order and Regression Trees (CART) [14], or Decision Trees (DT), are an extremely prominent arrangement and relapse approach. They are framed as tree structures containing middle (test) and leaf hubs. Each test hub speaks to a limit of the data space and each leaf hub speaks to a class if the DT is utilized for characterization or a straight model is utilized for relapse. The limits of the DT partition the first space into a lot of disjoint areas. The development of a DT is based on recursively apportioning the space of the data to make disjoint gatherings of focuses that amplify their intra-amass homogeneity and limit their between gathering homogeneity. The homogeneity metric of a gathering contrasts among the current calculations: GINI contamination has been utilized by the CART calculation [14], Information Gain has been utilized by ID3 and C4.5 [19] for order, though the Variance Reduction [14] is usually utilized for relapse. These heuristics are connected to each leaf to choose which measurement ought to be parceled and at which esteem. The end state of the DT development changes between various calculations as the tree stature is either pre-characterized or progressively chose, i.e., the tree develops until the point when new leaves hardly advantage its exactness.

Every limit of a DT is parallel to one hub of the data since it includes a solitary component of the data space, i.e., the limit line is communicated by a standard of the frame xi = c, where c is a steady esteem. Summing up this standard into a multivariate line, we acquire the diagonal DTs [15] that comprise of lines of the frame:

The multivariate limits support the expressiveness of a DT, since non-hub parallel examples can be perceived and communicated. In this paper, we use sideways DTs in two different ways: First, they are utilized to make a surmised model of the execution work. Second, their development calculation is altered to adaptively test the Deployment Space of the application, concentrating more on districts where the application shows a mind boggling conduct and disregarding areas where it will in general be effectively unsurprising.

II. RELATED WORK

Execution modeling is a strikingly inquired about zone. The particular approaches used to show the conduct of a given application can be allocated in three classes: (a) reproduction based, (b) imitating based and (c) approach including the benchmarking of the application and take a "discovery" see. In the main case, the approaches are based on known models of the cloud stages [30] and upgrade them with realized execution models of the applications under profiling. CDOSim [5] is an approach that objectives to display the Cloud Deployment Options (CDOs) and reproduce the expense and execution of an application. CloudAnalyst [6] is a comparative work that mimics extensive disseminated applications and concentrates their execution for various cloud arrangements. At long last, WebProphet [7] is a work that has practical experience in web applications. These works expect that execution models with respect to both the foundation and the application are known, rather than our approach that makes no suppositions neither for the application nor for the framework.

The possibility of copying based approaches is to send the application and catch execution follows for different situations, which are then "replayed" to the framework so as to foresee the execution. CloudProphet [31] is an approach utilized for relocating an application into the cloud. It gathers follows from the application running locally and replays them into the cloud, anticipating the execution it ought to accomplish over the cloud framework. /Trace [9] is an approach having some expertise in foreseeing the I/O conduct of a parallel application, distinguishing the causality between I/O designs among various hubs. In [10] a comparative ap-proach is displayed, in which a lot of benchmark appli-cations are executed in a cloud framework, estimating microarchitecture-autonomous attributes and assessing the connection between an objective and the benchmarked application. As per this relationship, an execution forecast is removed. At long last, [8] practices to I/O-bound BigData applications, producing a model of the virtualized stockpiling through small scale benchmarking and summing it up to anticipate the application execution. In spite of the fact that imitating approaches can be amazingly efficient for catching explicit parts of an application's conduct, they can't be commonly connected if the application structure is obscure.

The last classification of profiling approachs sees the application as a black box that gets various information sources and delivers various yields, comparing to execution measurements. The application is sent for some delegate arrangement setups and execution measurements are acquired, used by machine learning systems to delineate design space to the application execution. In [11], [16] a nonexclusive system is proposed used to derive the application execution of based on delegate arrangements of the design space. The approach handles the issue of summing up the execution for the whole arrangement space however does not handle the issue of picking the most suitable examples from the organization space, as the recommended approach. Frenzy [12] is a comparative work, that tends to the issue of picking agent focuses amid testing. This approach supports the focuses that have a place with the steepest locales of the Deployment Space, based on the possibility that these districts describe most suitably the whole execution work. Notwithstanding, it is excessively centered around the variations from the norm of the Deployment Space and the proposed approach beats it. Likewise, the issue of picking delegate tests of the Deployment Samples is additionally tended to by Active Learning [25].

III. PROPOSAL METHOD

This hypothetical model presents the term of vulnerability for a classifier that, basically, communicates its certainty to mark a particular example of the Deployment Space. Dynamic Learning favors the locales of the Deployment Space that present the most astounding vulnerability and, as PANIC, neglect to precisely rough the execution work for the whole space, as likewise demonstrated by our exploratory assessment. At long last, in [13] two progressively nonexclusive discovery approaches are given, using distinctive machine learning models for the estimate. Neither of these works, however, address the issue of picking the suitable examples, since they are progressively centered around the modeling issue.

A. Method overview

The fundamental thought of the proposed calculation is to parcel the Deployment Space by gathering tests that better fit diverse straight models, construe information about the execution work through examining the Deployment Space, sending the chose designs and, when a predefined number of organizations is achieved, demonstrate the execution work using distinctive direct models for each segment. In particular, at each progression, the Deployment Space is parceled by gathering the as of now acquired examples as indicated by their capacity to make a direct model that precisely approximates the application execution. Evaluations are then made in regards to the intra-assemble homogeneity, which relates to the expectation precision of the execution work for the predefined district. In this manner, inadequately approximated districts should be additionally examined so as to be better approximated, though precise areas require not be additionally investigated. Instinctively, the proposed calculation is an endeavor to adaptively "zoom-in" to zones of the Deployment Space where the conduct of the execution work is progressively dark, as in it is flighty and difficult to demonstrate. This empowers the quantity of enabled application arrangements to be progressively dispersed inside the Deployment Space, prompting increasingly precise forecasts as more examples are gathered for execution regions that are more enthusiastically to rough. This shrewd dispersion of the conveyed tests requires the closer examination of districts of the Deployment Space freely, something that is naturally directed by DTs. Their properties, for example, their adaptability to numerous measurements, power, and their intrinsic gap and-vanquish usefulness render them an ideal fit for the application profiling issue and encourage the decay of the Deployment Space in an instinctive and efficient way. In Algorithm 1, we give the pseudocode of the proposed philosophy.

DT-Based Adaptive Profiling Algorithm

```
1: procedure DTADAPTIVE($D$, $B$, $b$)
2:   $tree \leftarrow$ TREEINIT($\emptyset$), $samples \leftarrow \emptyset$
3:   while $|samples| \leq B$ do
4:     $tree \leftarrow$ PARTITION($tree$, $samples$)
5:     $s \leftarrow$ SAMPLE($D$, $tree$, $samples$, $b$)
6:     $d \leftarrow$ DEPLOY($s$)
7:     $samples \leftarrow samples \cup d$
8:   $model \leftarrow$ CREATEMODEL($samples$)
9:   return $model$
```

Algorithm 1 DT-based Adaptive Profiling Algorithm

Our Algorithm takes three info parameters: (a) the application's Deployment Space D, (b) the most extreme number of tests B and (c) the quantity of tests chose at every calculation emphasis b. The tree variable speaks to a DT, while the examples set contains the got tests. While the quantity of got tests is not as much as B, the accompanying advances are executed: First, the leaves of the tree are inspected and tried whether they can be supplanted by subtrees that further segment their areas (Line 4). The leaves of the extended tree are, at that point, examined with the SAMPLE work (Line 5), the picked tests are conveyed (Line 6) as indicated by the example's arrangement setup and execution measurements are acquired. Note that, s ? D though d ? D × P , i.e., s contains the examined setups and tests contains the organization arrangements alongside their execution esteem. At last, when B tests have been picked, the last model is made (Line 8).

B. Space Partitioning

Space apportioning happens through the development of the DT, i.e., the supplanting of the DT's leaves with test hubs with two new kids/leaves. The substitution of a leaf involves the ID of a split line that boosts the homogeneity between the two recently leaves. Review that, the use of slanted DTs enables the test hub to be represented by a multivariate line and upgrades their flexibility to various execution capacities. In any case, figuring an ideal multivariate split line is NP-finished [15]. Since we need to abuse the expressiveness of the diagonal segments without presenting a restrictive calculation cost for their estimation, we express the issue as an improvement issue [15] and use Simulated Annealing (SA) [20] to distinguish a close ideal line. In Algorithm 2, we present the PARTITION work. For each leaf of the tree (Line 3), SA is executed to distinguish the best multivariate split (Line 5), considering exclusively the examples whose Deployment Space-related measurements (d.in) lie inside the predefined leaf (Line 4). The examples of the predetermined leaf are then apportioned in two disjoint sets as per their position (Lines 7-11), in which the image shows that an example s is situated underneath the line. At long last, another test hub is created (Line 12), supplanting the first leaf hub of the tree and the new tree is returned.

Partitioning Algorithm:

```
1: procedure PARTITION($tree$, $samples$)
2:   $newTree \leftarrow tree$
3:   for $l \in$ leaves($tree$) do
4:     $v \leftarrow \{d | d \in samples, d.in \in l\}$
5:     $line \leftarrow SA(v)$
6:     $L_1 \leftarrow \emptyset, L_2 \leftarrow \emptyset$
7:     for $s \in v$ do
8:       if $s \preceq line$ then
9:         $L_1 \leftarrow L_1 \cup s$
10:      else
11:        $L_2 \leftarrow L_2 \cup s$
12:    $testNode \leftarrow \{line, L1, L2\}$
13:    $newTree \leftarrow$replace($l$, $testNode$)
14:  return $newTree$
```

Algorithm 2 Partitioning Algorithm

The foundation of SA's adequacy is the score work that evaluates the viability of every applicant arrangement. The strategies used by different DT development calculations make no supposition with respect to the idea of the data. In spite of the fact that this improves their flexibility to various issue spaces, their usage in this work brought about inadequately parceled Deployment Spaces. The data that our work endeavors to demonstrate have a place with an execution work. Naturally, if all the execution work focuses were accessible, one would envision that a closer perception of a particular area of the Deployment Space would exhibit a roughly direct conduct, as "neighboring" designs are foreseen to create a comparative execution. When concentrating on a solitary neighborhood, this "comparable execution" could be condensed by a straight hyperplane. A split line is viewed as "great" when the created leaves are best outlined by direct relapse models or, equally, if the examples situated in the two leaves can deliver straight models with low modeling blunder. Given a Deployment Space D, a line l and two sets L1, L2 that contain D's examples in the wake of parceling it with l (as in lines 6-10 of Algorithm 2), we gauge two straight relapse models for L1 and L2 and gauge their residuals utilizing the coefficient of assurance

R^2. l's score is: $Score(l) = -\frac{|L_1|R^2_{L_1} + |L_2|R^2_{L_2}}{|L_1|+|L_2|}$.

ideal fit for the direct model. Score(l) is limited when both L1 and L2 create very precise direct models or, equally, if the two sets can be precisely spoken to by two straight hyperplanes. We weight the significance of each set by the quantity of tests they contain as a mistaken model with more examples has a more prominent effect to the exactness than an off base model with less examples. The negative sign is utilized so as to stay lined up with the writing, in which SA looks for least focuses. At last, one SA property not examined up to this point is the capacity to accomplish an adaptable bargain between the modeling exactness and the required calculation. In particular, before SA's execution, the client characterizes a most extreme number of cycles to be executed for the distinguishing proof of the best part line. When one needs to boost the nature of the parceling, numerous emphasess are directed. Despite what might be expected, the quantity of emphasess is set to bring down qualities so as to consider a fast part line estimation.

C. Modeling

After B tests are returned by the profiling algorithm, they are used by the CREATE MODEL (Algorithm 1, Line 8)function to prepare another DT. The decision of preparing another DT as opposed to growing the one utilized amid the inspecting stage is made to amplify the precision of the last model. At the point when the primary test hubs of the previous DT were made, just a short segment of the examples was accessible to the profiling algorithm and, subsequently, the first DT may have at first made incorrect segments. Additionally, in situations where the quantity of got tests is equivalent to the dimensionality of the Deployment Space, the quantity of developed leaves is amazingly low and the tree deteriorates into a direct model that covers sizeable locales of the Deployment Space with diminished exactness. To conquer this restriction, alongside the last DT, a lot of Machine Learning classifiers are likewise prepared, keeping the one that accomplishes the most minimal Cross Validation mistake. Be that as it may, when the DT is prepared with enough examples, it outflanks the various classifiers. This is the principle explanation behind picking the DT as a base model for our plan: The capacity to give higher expressiveness by making different straight models in territories of higher unusualness, settle on them an ideal decision for modeling an execution work. Note that, the linearity of this model does not trade off its expressiveness: Non-straight execution capacities can likewise be precisely approximated by a piecewise direct model, through further apportioning the Deployment Space.

At last, we give a case of execution of Algorithm 1 for the Wordcount administrator. The administrator is executed for a 100G manufactured dataset, on a Hadoop bunch of fluctuating sizes, i.e., the Deployment Space is 1-d and speaks to the quantity of hubs of the Hadoop group (4– 256 hubs) and the Performance Space speaks to the execution time. We execute our procedure for a financial plan of B = 14 arrangements and b = 7 for every cycle. In Figures 1 (an) and (b) we delineate the genuine and approximated execution capacities for the first and second algorithm emphasess, separately.

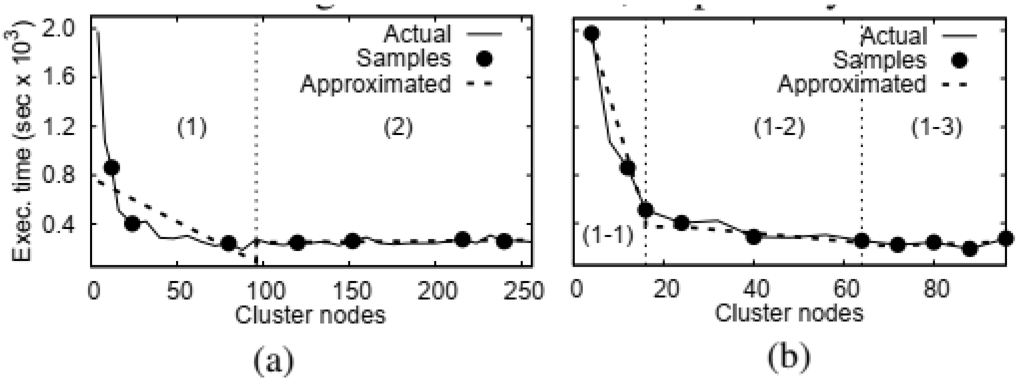

Figure 1.Algorithm execution for Wordcount

Amid the principal emphasis, 7 tests are arbitrarily picked and the primary dividing of the Deployment Space happens (for a group of 96 hubs) creating the regions (1) and(2) of Figure 1 (a). Note that, the execution work in the zone (2) presents straight conduct and, subsequently, is very precisely approximated with just 4 tried setups. In actuality, the execution work is firmly non-direct in the region (1) and, henceforth, inadequately approximated. The scores of the two Deployment Space zones are, at that point, determined, and in the accompanying cycle, all the 7 accessible designs are allocated to the zone (1), on the grounds that the mistake of region (2) is for all intents and purposes zero (and wsize = 0). In Figure (b), just region (1) is delineated, as whatever remains of the space stays flawless. The new examples authorize space parceling which, makes the model precisely surmised the genuine execution work utilizing a piecewise straight capacity. The last model approximates the genuine execution work amazingly precisely, analyzing just a minor 5% of the Deployment Space. This precedent is characteristic of our strategy's capacity: Our gap and-vanquish approach enables us to test and model every district independently, appointing an alternate dimension of detail to various areas of the Deployment Space. Additionally, the piecewise direct capacity is incredibly viable for the execution elements of ordinary Big Data applications and administrators, since the conduct saw in this model (i.e., the "Genuine" line) is generally experienced, particularly when the Deployment Space comprises of asset related measurements. This empowers us to furnish very precise approximations with an insignificant number of organizations. At last, notwithstanding while tending to execution elements of high multifaceted nature and unequivocally non-direct conduct, the use of progressively nitty gritty parcels (as in the zone (1) of Figure 1 (an)) ensures that the genuine line can be precisely approximated through the sending of more examples.

CONCLUSION

In this work, we returned to the issue of execution modeling of Big Data applications. Their arrangement space can develop exponentially extensive, making the required number of organizations for good exactness restrictively huge. We proposed an approach that uses angled Decision Trees to recursively segment and test the Deployment Space, expecting a most extreme number of organizations. Our approach figures out how to adaptively concentrate on regions where the model neglects to precisely rough application execution, accomplishing prevalent exactness under few organizations. We showed that our technique better approximates both reality and engineered execution capacities.

REFERENCES

[1] N. H. Kapadia, J. A. Fortes, and C. E. Brodley, “Predictive application-performance modeling in a computational grid environment,” in High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on. IEEE, 1999.

[2] C. Stewart and K. Shen, “Performance modeling and system management for multi-component online services,” in Pro-ceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. USENIX Association, 2005.

[3] D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasser-man, and M. Gittings, “Predictive performance and scalability modeling of a large-scale application,” in Proceedings of the 2001 ACM/IEEE conference on Supercomputing.ACM,2001.

[4] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghal-sasi, “Cloud computing - The business perspective,” Decision support systems, vol. 51, no. 1, 2011.

[5] F. Fittkau, S. Frey, and W. Hasselbring, “CDOSim: Simulating cloud deployment options for software migration support,” in Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the. IEEE, 2012.

[6] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, “Cloud-analyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications,” in Ad-vanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE, 2010.

[7] Z. Li, M. Zhang, Z. Zhu, Y. Chen, A. G. Greenberg, and Y.-
M. Wang, “WebProphet: Automating Performance Prediction for Web Services.” in NSDI, vol. 10, 2010.
[8] I. Mytilinis, D. Tsoumakos, V. Kantere, A. Nanos, and N.Koziris, “I/O Performance Modeling for Big Data Ap-plications over Cloud Infrastructures,” in Cloud Engineering(IC2E), 2015 IEEE International Conference on. IEEE, 2015.
[9] M. P. Mesnier, M. Wachs, R. R. Simbasivan, J. Lopez, J. Hen-dricks, G. R. Ganger, and D. R. O’hallaron, “//Trace: parallel trace replay with approximate causal events.” USENIX, 2007.
[10] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John, and K. De Bosschere, “Performance prediction based on inherent program similarity,” in Proceedings of the 15th international conference on Parallel architectures and com-pilation techniques. ACM, 2006.
[11] M. Gonc¸alves, M. Cunha, N. C. Mendonca, and A. Sampaio, “Performance Inference: A Novel Approach for Planning the Capacity of IaaS Cloud Applications,” in Cloud Comput-ing (CLOUD), 2015 IEEE 8th International Conference on. IEEE, 2015.
[12] I. Giannakopoulos, D. Tsoumakos, N. Papailiou, and N. Koziris, “PANIC: Modeling Application Performance over Virtualized Resources,” in Cloud Engineering (IC2E), 2015 IEEE International Conference on. IEEE, 2015.
[13] S. Kundu, R. Rangaswami, A. Gulati, M. Zhao, and K. Dutta, “Modeling virtualized applications using machine learning techniques,” in ACM SIGPLAN Notices, vol. 47, no. 7. ACM, 2012.
[14] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, Classification and regression trees. CRC press, 1984.
[15] D. Heath, S. Kasif, and S. Salzberg, “Induction of oblique decision trees,” in IJCAI, 1993.
[16] M. Cunha, N. Mendonc¸a, and A. Sampaio, “Cloud Crawler: a declarative performance evaluation environ-ment for infrastructure-as-a-service clouds,” Concurrency and Computation: Practice and Experience, 2016.
[17] M. Garc´ia-Valls, T. Cucinotta, and C. Lu, “Challenges in real-time virtualization and predictable cloud computing,” Journal of Systems Architecture, vol. 60, no. 9, 2014.
[18] “RightScale 2017 State of the Cloud Report,” https://www.rightscale.com/lp/2017-state-of-the-cloud-report.
[19] J. R. Quinlan, “Induction of decision trees,” Machine learn-ing, vol. 1, no. 1, 1986.
[20] P. J. Van Laarhoven and E. H. Aarts, “Simulated annealing,” in Simulated Annealing: Theory and Applications. Springer, 1987.
[21] S. Geisser, Predictive inference. CRC press, 1993, vol. 55.
[22] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 1998.
[23] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with YCSB,” in Proceedings of the 1st ACM symposium on Cloud computing. ACM, 2010.
[24] I. Giannakopoulos, D. Tsoumakos, and N. Koziris, “A De-cision Tree Based Approach Towards Adaptive Profiling of Distributed Applications (Extended Version),” arXiv preprint arXiv:1704.02855, 2017.
[25] B. Settles, “Active learning literature survey,” University of Wisconsin, Madison, vol. 52, no. 55-66, 2010.
[26] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: an update,” ACM SIGKDD explorations newsletter, vol. 11, no. 1, 2009.
[27] L. Rokach, “Ensemble-based classifiers,” Artificial Intelli-gence Review, vol. 33, no. 1-2, 2010.
[28] C. Dismuke and R. Lindrooth, “Ordinary least squares,” Methods and Designs for Outcomes Research, vol. 93, 2006.
[29] J. R. Quinlan, “Bagging, boosting, and c4. 5,” in AAAI/IAAI, Vol. 1, 1996.
[30] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simu-lation of cloud computing environments and evaluation of resource provisioning algorithms,” Software: Practice and Experience, vol. 41, no. 1, 2011.
[31] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, “CloudProphet: towards application performance prediction in cloud,” in ACM SIGCOMM Computer Communication Review, vol. 41, no. 4. ACM, 2011.