A Novel Method for the Synthetic Generation of Non-I.I.D Workloads for Cloud Data Centers

Furkan Koltuk
Electronic Systems Group, TÜBİTAK SAGE, Ankara, Turkey.
furkan.koltuk@tubitak.gov.tr
Department of Electrical and Electronics Engineering,
Middle East Technical University, Ankara, Turkey.
furkan.koltuk@metu.edu.tr

Ece Güran Schmidt
Department of Electrical and Electronics Engineering,
Middle East Technical University,
Ankara, Turkey.
eguran@metu.edu.tr

***Abstract*—Cloud data center workloads have time-dependencies and are hence non-i.i.d (independent and identically distributed). In this paper, we propose a new model-based method for creating synthetic workload traces for cloud data centers that have similar time characteristics and cumulative distributions to those of the actual traces. We evaluate our method using the actual resource request traces of Azure collected in 2019 and the well-known Google cloud trace. Our method enables generating synthetic traces that can be used for a more realistic evaluation of cloud data centers.**

***Index Terms*—cloud computing, model-based workload generation, distribution fitting.**

I. Introduction

Cloud data centers provide users virtual machines as Infrastructure as a Service (IaaS) or run user submitted tasks as Software as a Service (SaaS). Cloud data centers perform these services by assigning available hardware and software resources to the users. On the one hand, the satisfactory and efficient realization of these services must be ensured by the correct management of the existing resources. On the other hand, capacity planning must be executed to meet the increasing user demand and offer new services.

These requirements call for the comprehensive performance evaluation of cloud data centers under *realistic* workloads that reflect the user demands and usage patterns. Collecting workload traces from the field is a costly process. When such collected traces are used, customized results are obtained and it is difficult to investigate different cases [1]. Instead of real traces, *synthetic traces* with similar statistical properties to the actual traces can be generated using a model-based approach. Furthermore, the parameters of the model can be set to generate synthetic traces that represent different scenarios.

The fidelity of the workload model depends on the included statistical parameters. In addition to the distributions of the actual trace parameters, the time dependencies within the trace should also be considered. A number of previous studies in model-based cloud workload generation [2]–[4] assume that the modeled trace is independent and identically distributed (i.i.d) and ignore such time dependencies. However, there are time dependent variations in the cloud workload because of the users' day and night differences, possible different pricing and task scheduling policies of the cloud service provider. Modeling time-dependent changes in the cloud workload is important for capacity planning and evaluation of methods that make predictions about the future by looking at the current situation. [5], [6] study generating cloud workloads with time-dependent features however they do not provide a general model-based method that also includes the cumulative distribution of the modeled parameter.

In this paper, we propose a new model-based cloud workload generation method which generates synthetic traces that both match the cumulative distribution function (CDF) and the autocorrelation function (ACF) of the actual trace that is modeled. To this end, we assume that the modeled workload trace is stationary. A result of matching the ACF to that of the actual trace, the synthetic traces have very close Index of Dispersion for Counts (IDC) [7] and Hurst parameter (H) which measure the burstiness, self-similarity and long range dependency. We evaluate our method using the hourly user resource request counts in a very recent cloud workload trace published in 2019 [8] and in another popular trace [9] to show its generality.

The remainder of the paper is organized as follows. We present the previous work on cloud workload analysis and generation together with preliminaries to define the ACF, IDC and H in Section II. We present our two step method that first generates a synthetic trace with matching CDF and second with a matching ACF to the original trace in Section III. Section IV describes the application of our method to actual traces and presents the results. Finally Section V presents our conclusions and the outline of the next steps in our research.

II. Previous Work and Preliminaries

A. Workload Analysis and Modeling

[10]–[13] study the workloads of different cloud data centers. They provide CDFs for requested amount of re-

This work was supported by the Scientific and Research Council of Turkey (TUBITAK) [Project Code 117E667-117E668].

sources and life times of the jobs together with time series for resource utilization. Workload temporal properties are analyzed for the purpose of estimation with Autoregressive Moving Average-ARMA and Autoregressive Integrated Moving Average-ARIMA methods in [14], [15] respectively.

Cloud workload generation studies in the literature are mostly based on distribution fitting without considering the the time dependencies. To this end the resulting synthetic workloads have independent identical distribution (i.i.d). [2] models the amount of resources required for user tasks and hourly resource requests for the second version of the Google cloud traces published in 2011 [13]. Their models are based on distribution analysis and clustering on the users and tasks. [3] models resource utilization, task lifetimes and request arrival time intervals of a Web-based workload using distribution analyzes. In our previous work [4], we propose model-based trace generation for IaaS requests using the [16] trace. Our model includes clustering of the requests according to their resource demands and lifetimes and then fitting the cumulative distribution functions (CDF) of the intra-cluster requests. [5] states that the request arrivals to the data center are not Poisson Distributed. This work identifies the periodicities and inter-dependencies among different tasks to generate synthetic traces. Most of the modeling is using CDFs of interarrival times. [6] uses Markov Modulated Poisson Process (MMPP) processes to generate synthetic cloud workload traces with desired burst and self-similarityparameters. However, the CDFs for the arrival rates or interarrival times are not considered as model parameters.

B. Preliminaries for Time-dependent Workload Properties

Let $X = X_1, \cdots X_n$ be a stationary time series with sample average μ_X and sample variance σ_X^2.

The Autocorrelation Function (ACF) value $\rho_X(k)$ at lag $k = 0, \cdots, n-1$ for X is defined as follows:

$$\rho_X(k) = \frac{1}{\sigma_X^2 \cdot n} \cdot \sum_{t=1}^{n-k} (X_t - \mu_X)(X_{t+k} - \mu_X) \qquad (1)$$

We denote $\rho_X(k)$ for all lags $k = 0, \cdots, n-1$ with ρ_X. ρ_X measures the average relation of the data points to the preceding data points in X. Here a positive $\rho_X(k)$ indicates that data points separated by k stay above/below μ_X whereas a negative auto-correlation indicates that the data points separated by k tend to alternate about μ_X. Significant $\rho_X(k)$ values indicate periodicity with period k. The confidence bounds for the autocorrelation designate the significance of $\rho_X(k)$ according to the standard error of the Gaussian White Noise which is an i.i.d process. In this paper we define these bounds as twice the standard error; $\pm 2 \cdot \frac{1}{\sqrt{(n)}}$ [1], [17].

Burstiness of the workload is an important time-related property which implies fluctuations between high and low values. One measure for burstiness of a time series is the Index of Dispersion for Counts (IDC) metric [7]. Let $X_j^{(m)}$ show the elements of the time series X^mthat are obtained by m-fold aggregation of X as defined in Eq.(2).

$$X_j^{(m)} = \sum_{i=(j-1)m+1}^{jm} X_i \qquad (2)$$

Accordingly, the estimated IDC for X^m is defined as $I_X^m = C_{X^{(m)}}$ where $C_X = \frac{\sigma_X^2}{\mu_X}$. Here;

$$I_X^m = C_X \left[1 + 2 \sum_{k=1}^{m-1} \left(1 - \frac{k}{m}\right) \rho_X(k) \right]. \qquad (3)$$

IDC is a relative burstiness measure of X with respect to a Poisson series where $I_X^m = 1$. If there is a temporal structure in the time series, $\sigma^2_{X^{(m)}} > m \cdot \sigma_X^2$. Clustering of small and large sample values in the aggregated time series results in such larger dispersion of values. Note that, if X is i.i.d, $I_X^m = C_X$ since $\rho_X(k) = 0, \forall k$.

Computer workloads often display *self-similarity* and *long-range dependence* which manifest as burstiness [1]. An exactly self similar process $X(t)$ has the same finite dimensional distribution as $\frac{1}{a^{H_X}} X(at)$ implying that the bursts occur in all time scales. Here H_X is the *Hurst Parameter* for X which characterizes the degree of self similarity. Typically self-similarity results from long-range dependence that implies large $\rho_X(k)$ values although k is large. Poisson Process has $H = 0.5$ whereas $H > 0.5$ indicates self similarity and long-range dependency.

III. Proposed Method

Our proposed method is applicable to any workload trace that is presented as time series. We assume that the cloud data center workload is stationary [1]. In this paper, we apply and evaluate our method for the number of virtual machine requests received per hour in the cloud data center. Accordingly, the actual trace is a time series $X = X_1, \cdots X_n$ that consists of n samples. Each sample X_i shows the number of requests during hour i. We denote the un-ordered set of samples in this series with $\mathcal{X}$.

Our method has two steps. In the first step, we generate an i.i.d synthetic sample set $\hat{\mathcal{X}}$ with a *matching cumulative distribution* to $\mathcal{X}$. In the second step, we algorithmically construct the synthetic workload trace as a time series $\hat{X}^{\text{fit}}$ out of $\hat{\mathcal{X}}$ such that $\rho_{\hat{X}^{\text{fit}}} \approx \rho_X$.

We note that there is no limit to the number of samples in the synthetic trace $\hat{X}^{\text{fit}}$ where as the largest lag k that can be matched is limited to $n-1$. Accordingly, we keep the $\hat{\mathcal{X}}$ sample count the same as the $\mathcal{X}$ sample count to simplify the comparison of results.

A. Cumulative Distribution Function Matching

In the first step, we identify a cumulative distribution function that fits to the actual trace sample set $\mathcal{X}$. In this work, we choose Anderson-Darling (AD) test [18] as the Goodness of Fit (GoF) test similar to [3] and [2]. We apply the AD test on a subset of samples that are randomly selected among $\mathcal{X}$ if necessary as GoF tests are not suitable for large data sets

[1]. We fit the continuous distributions in [19] to $\mathcal{X}$ using MATLAB built-in function `fitdist` which determines the distribution parameters that would result in a best fit using maximum likelihood estimation. We then apply Anderson-Darling test using using MATLAB built-in function `adtest` to compare each fit distribution to $\mathcal{X}$. We select the *candidate fit distributions* with the $AD < cv$ and $p > 0.05$. Here, $AD < cv$ is an indicator of a good fit where the critical value, cv, is constant for a distribution with specified parameters [20].

It is important to note that some distributions are defined by parameters that are indirectly related to mean and variance. Accordingly, among the candidate fit distributions we select the distribution that has a large p, small AD and σ^2/μ that is close to C_X. After selecting the distribution, we generate a set of randomly generated samples $\hat{\mathcal{X}}$ out of the selected distribution. The first step of our method concludes by randomly ordering the elements of $\hat{\mathcal{X}}$ into a time series $\hat{X}^{\text{init}}$. We remind that $\hat{X}^{\text{init}}$ has a matching CDF to the actual time series X.

B. Autocorrelation Function Matching

We propose an iterative, heuristic Algorithm 1 that changes the time instants of $\hat{X}^{\text{init}}$. We denote the time series under construction in iteration q with Y^q where $Y^0 = \hat{X}^{\text{init}}$. Subsequently, in each iteration q, we construct a new time series Y^q by shuffling the time instants of S samples in Y^{q-1} among each other. We then compare ρ_{Y^q} to ρ_X with the MSE function. If ρ_{Y^q} yields less error than $\rho_{Y^{q-1}}$, the next iteration goes on with Y^q, otherwise Y^{q-1} is used. The iterations terminate when MSE value is less than a selected error threshold ϵ_ρ. Note that, for our work presented in this paper, we are matching the ACF values of all $k = 0, \cdots, n-1$ with ρ_X. It is possible to apply the method only for a group of selected lags.

Algorithm 1 Autocorrelation Matching Algorithm

```
1:  Inputs: $X$, $\hat{X}^{\text{init}}$, $S$, $\epsilon_\rho$
2:  Output: $\hat{X}^{\text{fit}}$
3:  Iteration: $q$
4:  Initialization: $q = 0$, $Y^0 = \hat{X}^{init}$, $e_{\text{min}} = MSE(\rho_X, \rho_{Y^0})$
5:  while $e_{\min} > \epsilon_\rho$ do
6:      $q = q + 1$
7:      Copy: $\hat{Y^q} = Y^{q-1}$
8:      $l$: A list of randomly selected $S$ sample time instants out of $\{1, \ldots, n\}$
9:      $v = \hat{Y^q}[l]$: list of the the sample values at $l$ instants
10:     Construct list $\hat{v}$ by randomly shuffling the values in $v$
11:     Place the values in the list $\hat{v}$ at the time instants $l$: $Y^q[l] = \hat{v}$
12:     $e = MSE(\rho_X, \rho_{Y^q})$
13:     if $e < e_{\min}$ then
14:         $e_{\min} = e$
15:     else
16:         $Y^q = Y^{q-1}$
17: return $\hat{X}^{\text{fit}} = Y^q$
```

IV. Application of the Method

We apply our method to hourly incoming resource requests for different cloud services from two different data centers: Azure [8] and Google [9] in order to show its generality. [8] consists of $n = 720$ hourly virtual machine request and usage information within the scope of Infrastructure as a Service-IaaS. This anonymized trace was collected for 30 days in 2019. [9] consists of $n = 695$ hourly task requests to run on servers as part of Software as a Service-SaaS. This trace has a similar length to [8], and was collected in 2011. We investigate a 10-day, part of the Azure trace with $n = 240$ samples for easy inspection on less number of samples. To this end, we call the traces we evaluate as Azure 10 Days, Azure 30 Days and Google 30 Days.

Fig. 1 (c), 2(c) and 3(c) show the time series and Fig. 1 (d), 2(d) and 3(d) show the autocorrelation values for the actual traces for Azure 10 Days, Azure 30 Days and Google 30 Days, respectively.

First of all, we observe that these workloads are not i.i.d as each of them has a large number of lags k with $\mid \rho_X(k) \mid > 2 \cdot \frac{1}{\sqrt{(n)}}$ as defined in Section II-B. Furthermore, we observe that particularly for Azure 2019 traces, the autocorrelation value is large around small lags and around 24 hour-daily period.

Consistent with these autocorrelation values, the persistence of request counts can be observed in the time series. Accordingly, there are no sudden large changes between the previous and next sample values.

Our second observation is that there is no increasing or decreasing values of the actual time series as a trend. Hence, there is no evident violation of the stationarity assumption.

We apply the first step of our method as described in Section III-A. The fit distributions and their estimated parameters are presented in Table I. Here, α, c and k are the scale, first shape and second shape parameters respectively for the Burr distribution. Similarly, k, σ and μ are the shape, scale and location parameters respectively for the Generalized Extreme Value distribution.

TABLE I: Fit Distributions, Estimated Parametes, GoF Test Results

	Azure 10 Days	**Azure 30 Days**	**Google 30 Days**
Distribution	Generalized Extreme Value	Burr	Generalized Extreme Value
Parameters	$k = -0.2054$ $\sigma = 666.23$ $\mu = 3242.23$	$\alpha = 3868.87$ $c = 6.24$ $k = 1.90$	$k = 0.16$ $\sigma = 607.94$ $\mu = 2419.61$
p	0.94	0.92	0.97
AD, cv	$0.30, 2.493$	$0.33, 2.4926$	$0.25, 2.4928$

We show the analytical CDF that is fit with the parameters that are in Table I in Fig. 1(a), 2(a), 3(a), respectively. We also show the empirical CDF (ECDF) values for the actual trace, and the two different synthetic traces that we generate from this trace are shown in the same Figure. We observe that ECDF values of the synthetic traces are similar to each other and to that of the analytical CDF values.

We create two examplary i.i.d traces for each actual trace. The resulting time series are shown in Fig. 1(b), 2(b), 3(b), respectively. We observe that despite having the same CDF with the actual series, the autocorrelation values at almost all lags are within the confidence bounds for these traces.

We then apply the second step of our method described in Section III-B. To this end, we apply the Autocorrelation Matching Algorithm with an error threshold of $\epsilon_\rho = 10^{-4}$. We display the resulting time series for these two exemplary synthetic traces in Fig.1 (e) and (g), 2 (e) and (g),3 (e) and (g), respectively. We observe that, although synthetic traces are similar to each other and the actual trace, they are not identical.

Fig.1 (e) and (g), 2 (e) and (g),3 (e) and (g) show the ACF values of the two synthetic traces produced with our method. We observe that $\rho_{\hat{X}^{\text{fit}}} \approx \rho_X$ for both synthetic traces.

We compare the Index of Dispersion (IDC) values of the actual traces to the IDC values of the synthetic traces produced by our method. We select the aggregation values $m = 5, 10, 20$. The IDC values computed for these selected m are approximately unity for a Poisson time series with the same number of samples and mean of μ_X. Here we note that, we get $I^m_X \approx \overline{I^m_{\hat{X}^{\text{fit}}}}$, $\forall m = 2, \cdots, n$.

To this end, we generate 10 i.i.d traces $\hat{X}^{\text{init}}$ and the resulting synthetic traces $\hat{X}^{\text{fit}}$. We then compute the *average* IDC and H values denoted by $\overline{I^m_{\hat{X}^{\text{init}}}}$, $\overline{I^m_{\hat{X}^{\text{fit}}}}$,$\overline{H_{\hat{X}^{\text{init}}}}$, and $\overline{H_{\hat{X}^{\text{fit}}}}$ respectively. We estimate H using the re-scaled range method [1].

We report the confidence bounds around the true mean for a 95% confidence for each trace in Table II. In addition, we report the observed maximum deviations of IDC and H between the original and synthetic trace defined as follows

$$\Delta_{\text{IDC}} = \max_{m=5,10,20} \{100 \cdot \mid I^m_X - \overline{I^m_{\hat{X}^{\text{fit}}}} \mid / I^m_X\}, \qquad (4)$$

$$\Delta_H = 100 \cdot \mid H_X - \overline{H_{\hat{X}^{\text{fit}}}} \mid / H_X. \qquad (5)$$

We observe that $\overline{I^m_{\hat{X}^{\text{fit}}}}$ is very close to I^m_X for the traces Azure 10 Days and Azure 30 Days and sufficiently close to I^m_X for the trace Google 30 Days. $\overline{H_{\hat{X}^{\text{fit}}}}$ is very close to H_X for all traces.

Tables III, IV and, V show the detailed evaluation of the IDC and H values for Azure 10 Days, Azure 30 Days and Google 30 Days, respectively. Here we observe that $\overline{I^m_{\hat{X}^{\text{init}}}} \approx C_X$ because of the i.i.d property. Similarly, $\overline{H_{\hat{X}^{\text{init}}}}$ is very close to 0.5. There is a deviation from C_X caused by the finite sample count for the relatively shorter Azure 10 Days trace.

We would like to note that [6] takes the IDC and H values as inputs and accordingly creates synthetic traces. They report a deviation of the IDC and H of the generated series from that of the actual series between less than 1% to 10% and 1% to 2%.

TABLE II: Mean IDC and H evaluation for $\hat{X}^{\text{fit}}$

	Azure 10 Days	**Azure 30 Days**	**Google 30 Days**
95% **confidence interval for** $\overline{I^m_{\hat{X}^{\text{fit}}}}$	$\pm 2.3\%$	$\pm 3\%$	$\pm 4\%$
Δ_{IDC} [%]	2%	2%	7%
95% **confidence interval for** $\overline{H_{\hat{X}^{\text{fit}}}}$	$\pm 2.5\%$	$\pm 1\%$	$\pm 1.5\%$
Δ_H [%]	3%	2.5%	1%

TABLE III: IDC and H values for Azure 2019 10 Days Trace.

m	I^m_X	$\overline{I^m_{\hat{X}^{\text{fit}}}}$	$\overline{I^m_{\hat{X}^{\text{init}}}}$, $C_X = 135.42$
5	424.47	426.98	133.282
10	657.2	662.58	126.63
20	883.73	893.01	116.55

	H_X	$\overline{H_{\hat{X}^{\text{fit}}}}$	$\overline{H_{\hat{X}^{\text{init}}}}$
	0.801	0.826	0.550

TABLE IV: IDC and H values for Azure 2019 30 Days Trace.

m	I^m_X	$\overline{I^m_{\hat{X}^{\text{fit}}}}$	$\overline{I^m_{\hat{X}^{\text{init}}}}$, $C_X = 186.63$
5	640.67	628.528	183.159
10	1036.1	1021.147	181.303
20	1467.6	1458.18	182.923

	H_X	$\overline{H_{\hat{X}^{\text{fit}}}}$	$\overline{H_{\hat{X}^{\text{init}}}}$
	0.91	0.887	0.567

TABLE V: IDC and H values for Google 30 Days Trace.

m	I^m_X	$\overline{I^m_{\hat{X}^{\text{fit}}}}$	$\overline{I^m_{\hat{X}^{\text{init}}}}$,$C_X = 360.01$
5	1176.8	1093.47	348.394
10	1822.1	1705.57	350.191
20	2648.2	2492.51	345.156

	H_X	$\overline{H_{\hat{X}^{\text{fit}}}}$	$\overline{H_{\hat{X}^{\text{init}}}}$
	0.845	0.835	0.571

V. Conclusion and Future Work

In this paper, we propose a novel two-step method to generate realistic synthetic workload traces with the same Cumulative Distribution (CDF) and similar Autocorrelation Function (ACF) values as the actual trace. When we evaluate the ACF for actual resource request traces represented as time series, we observe that while they show stationarity they are not independent and identically distributed (i.i.d) as their ACF values have significant lags particularly reflecting the daily patterns. To this end, our proposed method enables realizing the time dependencies of the actual workload trace in the generated synthetic trace. Furthermore, as an outcome, the synthetic traces posses similar burstiness, self-similarity and long-range dependence properties to the actual traces which are measured by the Index of Dispersion for Counts (IDC) and the Hurst parameter metrics. We apply our method on a cloud data center request trace that is published in 2019 together with a legacy trace that is published in 2011 and demonstrate that we can generate synthetic traces with very similar CDF and ACF to the actual traces.

In the next step of this research, we will construct a comprehensive cloud workload model that includes the virtual machine utilization time series. The time-dependent parameters of the model will be generated using our method presented in this paper to generate realistic synthetic traces with desired

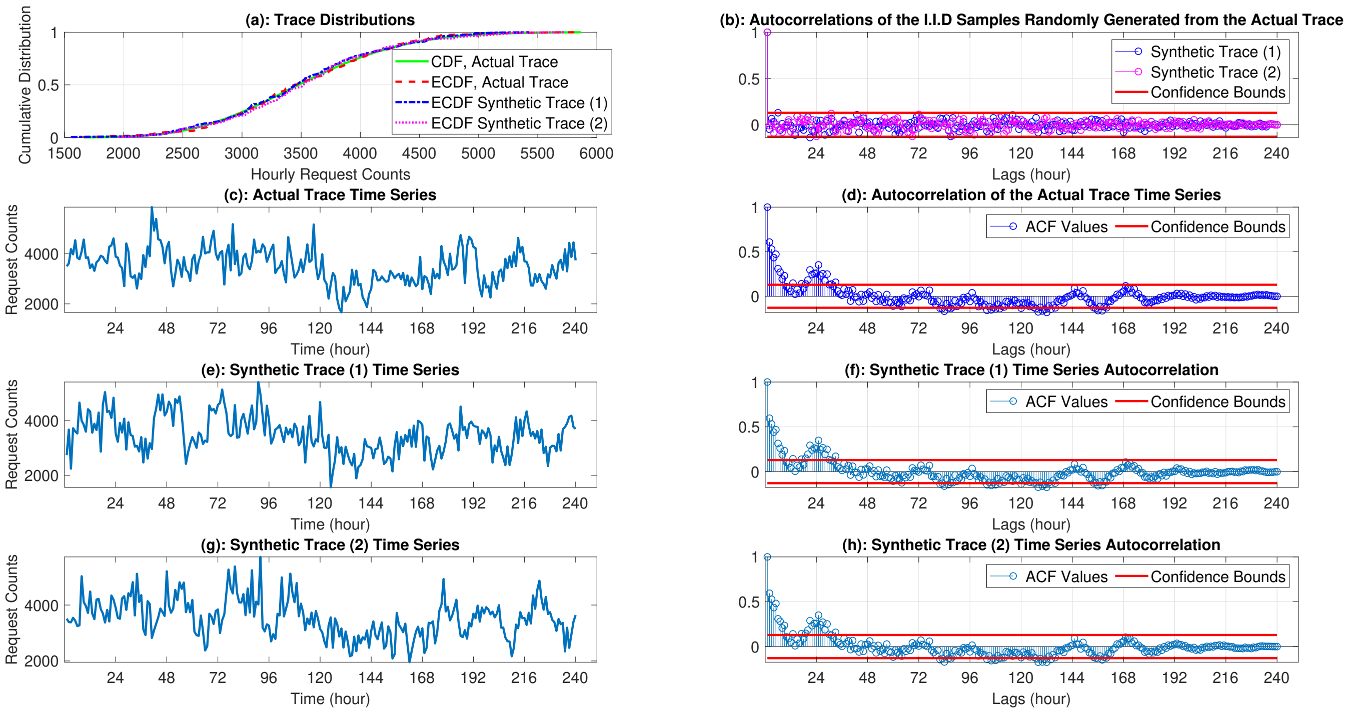

Fig. 1: Azure 2019 10 days trace, 2 example generated synthetic traces.

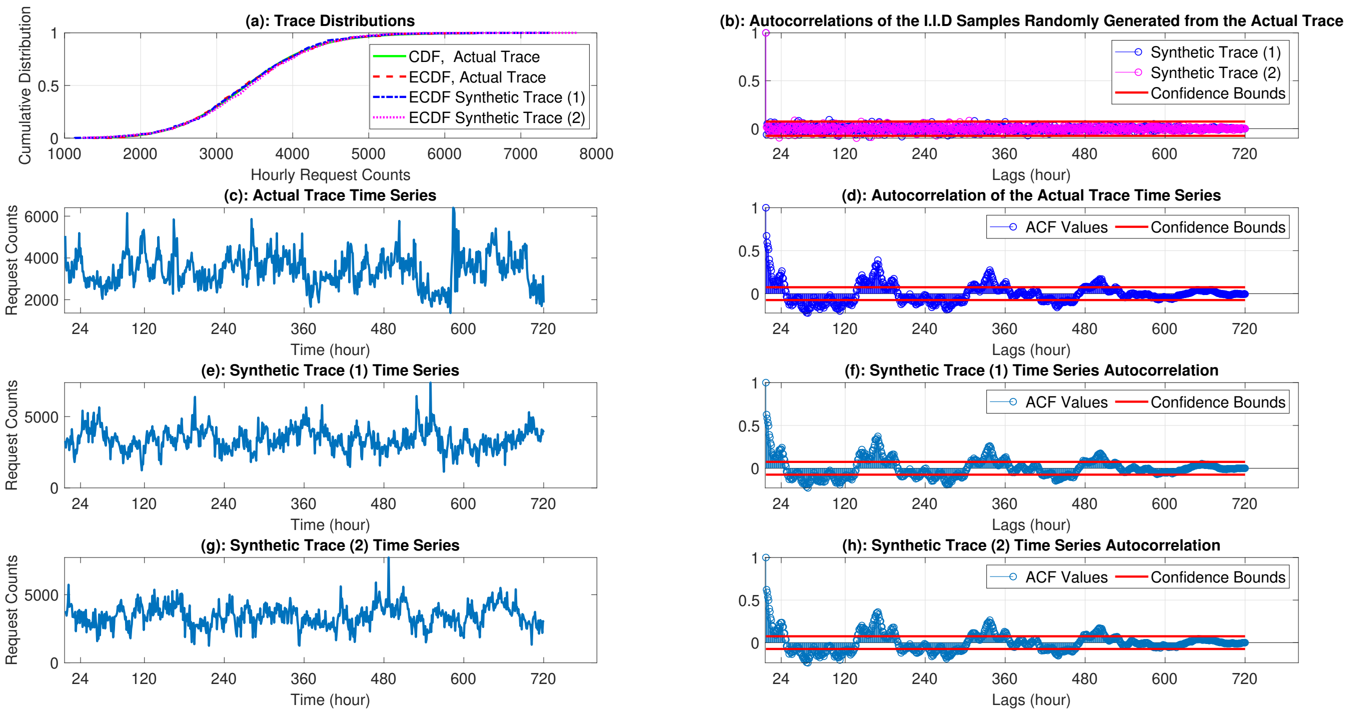

Fig. 2: Azure 2019 30 days trace, 2 example generated synthetic traces.

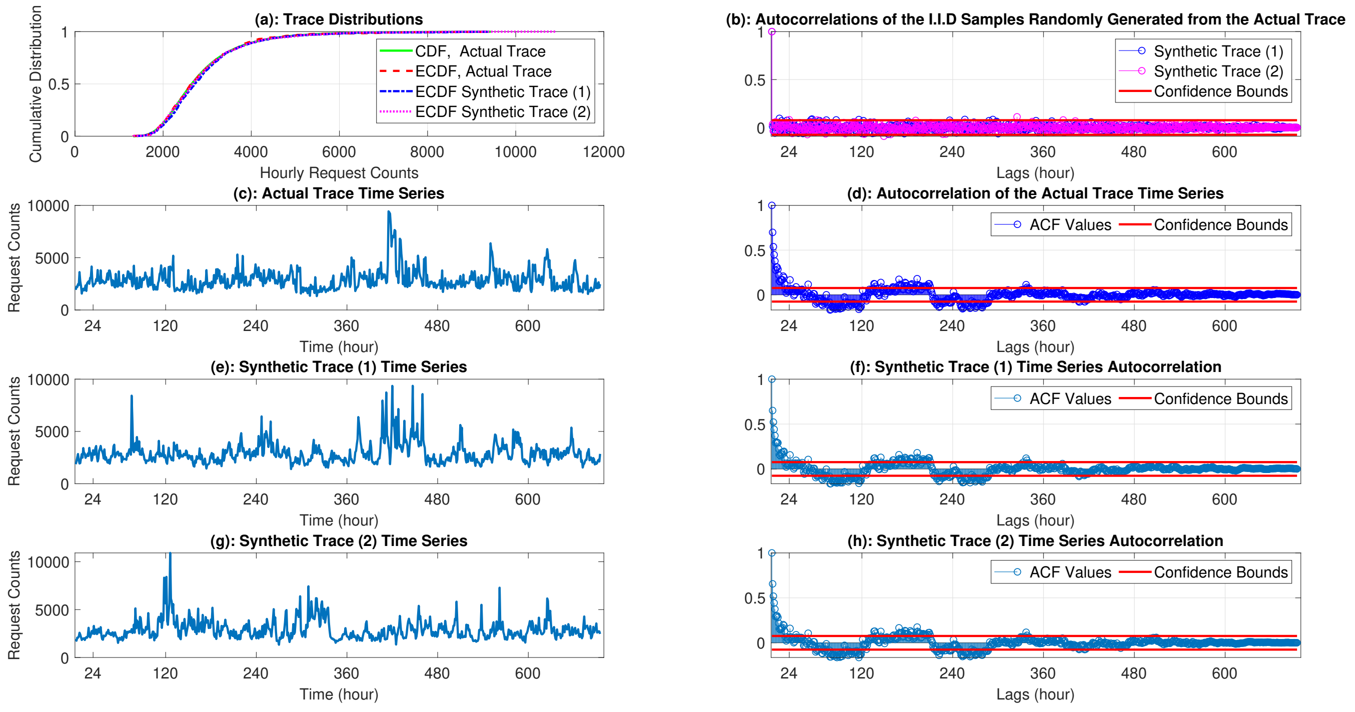

Fig. 3: Google 30 days trace, 2 example generated synthetic traces.

level of detail. The realistic synthetic traces will be used to test novel cloud resource allocation and resource demand prediction methods to be tested on a laboratory test-bed for a hardware accelerated cloud.

References

[1] D. G. Feitelson, *Workload modeling for computer systems performance evaluation*. Cambridge University Press, 2015.

[2] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 208–221, April 2014.

[3] D. Magalhães, R. N. Calheiros, R. Buyya, and D. G. Gomes, "Workload modeling for resource usage analysis and simulation in cloud computing," *Computers & Electrical Engineering*, vol. 47, pp. 69–81, 2015.

[4] F. Koltuk, A. Yazar, and E. G. Schmidt, "Cloudgen: Workload generation for the evaluation of cloud computing systems," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2019, pp. 1–4.

[5] D.-C. Juan, L. Li, H.-K. Peng, D. Marculescu, and C. Faloutsos, "Beyond poisson: Modeling inter-arrival time of requests in a datacenter," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2014, pp. 198–209.

[6] J. Yin, X. Lu, X. Zhao, H. Chen, and X. Liu, "Burse: A bursty and self-similar workload generator for cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 668–680, 2014.

[7] R. Gusella, "Characterizing the variability of arrival processes with indexes of dispersion," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 2, pp. 203–211, 1991.

[8] "Azure Public Dataset 2019," accessed: 2019-07-30. [Online]. Available: https://github.com/Azure/AzurePublicDataset/blob/master/AzurePublicDatasetV2.md

[9] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, pp. 1–14, 2011, revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.

[10] G. Amvrosiadis, J. W. Park, G. R. Ganger, G. A. Gibson, E. Baseman, and N. DeBardeleben, "On the diversity of cluster workloads and its impact on research results," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 533–546.

[11] C. Jiang, G. Han, J. Lin, G. Jia, W. Shi, and J. Wan, "Characteristics of co-allocated online services and batch jobs in internet data centers: a case study from alibaba cloud," *IEEE Access*, vol. 7, pp. 22 495–22 508, 2019.

[12] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 153–167.

[13] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 7.

[14] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE 4th International Conference on Cloud Computing*. IEEE, 2011, pp. 500–507.

[15] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2014.

[16] "Azure Public Dataset 2017," accessed: 2018-06-30. [Online]. Available: https://github.com/Azure/AzurePublicDataset/blob/master/AzurePublicDatasetV1.md

[17] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: with R examples*. Springer, 2017.

[18] "Nist/sematech e-handbook of statistical methods," accessed: 2018-12-21. [Online]. Available: http://www.itl.nist.gov/div898/handbook/

[19] "Mathworks documentation," accessed: 2019-01-10. [Online]. Available: https://www.mathworks.com/help/stats/fitdist.html?s_tid=doc_ta

[20] "The anderson darling statistic." [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/a079807.pdf