Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 158–167

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Clustering Cloud Workloads: K-Means vs Gaussian Mixture Model

Eva Patel[a,*], Dharmender Singh Kushwaha[a]

[a]*Motilal Nehru National Institute of Technology Allahabad, Prayagraj 211004, India*

Abstract

The growing heterogeneity due to diverse Cloud workloads such as Big Data, IoT and Business Data analytics, requires precise characterization to design a successful capacity plan and maintain the competitiveness of Cloud service providers. K-Means is a simple and fast clustering method, but it may not truly capture heterogeneity inherent in Cloud workloads. Gaussian Mixture Models can discover complex patterns and group them into cohesive, homogeneous components that are close representatives of real patterns within the data set. This work compares K-Means and Gaussian Mixture Model to evaluate cluster representativeness of the two methods for heterogeneity in resource usage of Cloud workloads. Experiments conducted with Google cluster trace and business critical workloads by Bitbrains reveal that clusters obtained using K-Means give a very abstracted information. Gaussian Mixture Model provides better clustering with distinct usage boundaries. Although, Gaussian Mixture Model has higher computation time than K-Means, it can be used when more fine-grained workload characterization and analysis is required.

Keywords: Cloud Computing; Workload characterization; Clustering; Expectation-Maximization; K-Means Clustering; Gaussian Mixture Model

1. Introduction

On-demand compute power and data storage are the key reasons behind rising Cloud adoption, emergence of new computing paradigms such as Volunteer computing, Fog computing, and Serverless computing, and a horde of new workloads such as Big Data, IoT and Business Data analytics [1, 2], hosted on geographically distributed resources [3]. Heterogeneity of workloads is now the norm in Cloud Computing environment [4, 5]. Identifying workload characteristics is therefore crucial for effective utilization of the infrastructure capacity and guaranteed Cloud services.

A fundamental step in Cloud infrastructure capacity planning is resource usage monitoring and workload characterization. *Workload characterization* is the process of identifying individual workload components and its describing attributes with respect to a characterizing objective [6]. A good characterization is one that truly represents workload behaviour in terms of an objective of interest. However, dynamic resource requirements, unforeseen demand surges, and disparate computational and configuration requirements of Cloud applications - transactional, scientific, stream-

* Corresponding author. Tel.: +91-808-557-5111.
E-mail address: evapatel08@gmail.com

1877-0509
10.1016/j.procs.2020.04.017

ing, social networking - introduce heterogeneity, making the workload characterization process extremely complex. Resource usage not only at higher level but also at the level of individual CPU cores may need to be considered [7].

K-Means and Gaussian Mixture Model (GMM) are unsupervised clustering techniques. K-Means groups data points using distance from the cluster centroid [8] - [16]. GMM uses a probabilistic assignment of data points to clusters [17] - [19]. Each cluster is described by a separate Gaussian distribution. The generated clusters correspond to distinct heterogeneous data patterns. In this work, Cloud workloads from Google cluster and business critical workloads from Bitbrains are clustered using K-Means and GMM to compare the representativeness of generated clusters for heterogeneity of resource usage patterns of Cloud workloads. The contributions are:

1. Clustering of workload traces using K-Means and GMM.
2. An experimental evaluation of the two methods for representativeness of heterogeneity in real workloads.

In Section 2 we discuss previous works on workload characterization. Section 3 briefly describes the concepts of clustering, Expectation-Maximization, K-Means Clustering and GMM. Experiments conducted are detailed in Section 4 and results obtained are discussed in Section 5. Section 6 concludes the work with future scope.

2. Related Work

Cluster analysis is the fundamental step for identifying workloads with similar resource usage patterns. Several works have been proposed for clustering workloads using K-Means. [8, 9] characterize one-month Google cluster trace by CPU and memory usage using the Forgy method for centroid initialization. Optimal number of clusters is determined using a merge ratio threshold. Moreno et al. [10] use K-Means clustering to cluster workloads based on user behaviour and task characteristics. Number of clusters is determined by comparing the variability of all elements within a cluster against a threshold value. [11] partition virtual machines (VM) based on CPU utilization. Each cluster is analysed to extract peak CPU usage, number of occurrences of the peak, range of variations in the number of peaks and sum of the width of peaks. [12] propose enhancements to K-Means method to improve clustering performance and convergence rate of the algorithm. Authors in [13] improve initial guesses of cluster centers for K-Means and then cluster Cloud workloads based on their latency sensitivity requirements. [14] and [15] classify VMs using resource usage and compare the performance of K-Means with that of Neural networks. [16] cluster TPC-H benchmark data to evaluate the performance of data analytics applications on available processor designs, using features that are related to and independent of micro-architecture. Authors in [17] characterize batch and streaming workloads via Gaussian Mixtures based on the observation that clusters need not be disjoint. [18] use CPU, memory, disk and network usage to cluster workloads using a mixture of Gaussians.

K-Means is widely used in scientific and industrial applications due to its simplicity and speed. However its use of Euclidean distance as similarity measure limits its ability to identify complex non-linear usage structures [20]. A GMM is an effective generative as well as probabilistic clustering method [20]. We conduct experiments on Google Cluster trace and business workloads by Bitbrains, to compare the representativeness of clusters to varying resource usage behaviour.

3. Background Concepts

3.1. Clustering

Clustering is the process of grouping data objects using a similarity measure [19]. Clustering can be hierarchical or partitional, exclusive, overlapping or fuzzy, and complete or partial [20]. K-Means is a *partitional clustering* technique; data objects are divided into non-overlapping groups. Clusters may be well-separated, prototype-based, graph based or density based. K-Means is a prototype-based clustering while GMM generates density-based clusters. A *prototype-based cluster* is represented by a prototype such that all members within a cluster are close to the corresponding prototype. Centroid and medoid are two commonly used prototypes. A *density-based cluster* is a region of objects with high density, surrounded by low-density regions. Grouping data into clusters simplifies identification and summarizing of properties of interest. Workloads with similar usage patterns can help design capacity plans that

accommodate future resource requirements while maintaining Service Level Agreement (SLA) for currently running services.

3.2. Maximum Likelihood and the Expectation Maximization Algorithm

Maximum likelihood is a method of estimating model parameters which maximize the likelihood that the model generates the observed data [20]. Machine learning algorithms minimize an error function by evaluating negative log of the likelihood function. The *Expectation-Maximization* (EM) algorithm is an iterative method of computing maximum likelihood estimations for probabilistic models with latent variables i.e. variables that are inferred from observed data. The EM algorithm involves following steps:

1. *Initialization*: Obtain an initial estimate of the model parameters, usually through a random initialization.
2. *Expectation Step*: Estimate the values of the latent variables, with values of the model parameters fixed.
3. *Maximization Step*: Estimate new values for the model parameters that minimize an error function.
4. *Termination*: Repeat steps 2 and 3 until a convergence criterion is met.

3.3. K-Means

K-Means clustering learns properties of a set of data points and forms partitions called *clusters*, that represent data with similar properties [19]. For continuous data, each cluster is represented by the centroid which is the mean of cluster members. For categorical data, medoid, which corresponds to the most frequently occurring object, is used as cluster prototype. K-Means uses squared Euclidean distance as the similarity measure for cluster membership:

$$d_{sq} = \sum_{i=1}^{D} (x_i - y_i)^2 \tag{1}$$

Here x and y are points in the D-dimensional space. Number of clusters k is determined by minimizing the *Sum of Squared Errors* (SSE), which is the sum of squared error of each data point and its nearest centroid. It is given by (2)

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{i,j} \left\| x_i - c_j \right\|^2 \tag{2}$$

where c_j is the centroid of j^{th} cluster, and $w_{i,j} = 1$ if the data point x_i is in cluster j and $w_{i,j} = 0$, if x_i is not in cluster j [20].

3.3.1. Expectation Maximization for K-Means Clustering

K-Means uses the iterative EM algorithm to determine the cluster membership that minimizes the SSE [20]. Given k, the number of clusters, data points within the set are grouped into clusters using following steps:

1. Randomly initialize k centroids c_j, $j \epsilon \{1, ..., k\}$ from the data set.
2. Repeat until convergence:

   (a) *E-step*: For each data point, recompute the distance to each centroid c_j, $j \epsilon \{1, ..., k\}$ and assign each point to the cluster with nearest centroid. In this step, the $c_j's$ in equation (2) are fixed and $w_{(i,j)}'s$ are updated.
   (b) *M-step*: Recompute the mean for each cluster and update the cluster centroids $c_j's$, keeping the $w_{(i,j)}'s$ fixed.

The algorithm is said to converge when either there is no change in cluster assignment, a predefined tolerance value for distortions is reached or the algorithm has iterated a maximum number of times.

3.3.2. Limitations

1. K-Means has no mechanism to handle the uncertainty when a data point is close to more than one cluster centroid.
2. K-Means fails to produce optimal clusters for complex, non-linear decision boundaries.
3. It is sensitive to initial guess of centroids. Different initializations may lead to different clusters.

3.4. Gaussian Mixture Model

Gaussian distribution, also called the *Normal distribution* , is a continuous probability distribution [20], given by

$$\mathcal{N}(X|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}} exp\left\{-\frac{(X-\mu)^T\Sigma^{-1}(X-\mu)}{2}\right\} \tag{3}$$

where μ is a D-dimensional mean vector, Σ is a D $\times$ D covariance matrix, which describes the shape of the Gaussian and $|\Sigma|$ denotes the determinant of Σ.

Gaussian distribution is symmetrical about the mean and is described by the mean and the standard deviation. However, the unimodal property of a single Gaussian distribution cannot model the multiple density regions within a multimodal data set found in practical situations and therefore, it cannot learn the heterogeneous resource usage behaviour of Cloud workloads. Complex, multimodal distributions can be appropriately modelled using a mixture of Gaussian distributions.

A GMM is an unsupervised clustering technique that forms ellipsoidal shaped clusters based on probability density estimations using the Expectation-Maximization. Each cluster is modelled as a Gaussian distribution. The mean and the covariance rather than only the mean as in K-Means, give GMMs the ability to provide a better quantitative measure of fitness per number of clusters.

A GMM is represented as a linear combination of the basic Gaussian probability distribution and is expressed as

$$p(X) = \sum_{k=1}^{K} \pi_k \mathcal{N}(X|\mu_k,\Sigma_k) \tag{4}$$

where, K is the number of components in the mixture model and π_k is called the *mixing coefficient*, which gives an estimate of the density of each Gaussian component. The Gaussian density given by $\mathcal{N}(X|\mu_k,\Sigma_k)$, is called a *component* of the mixture model. Each component k is described by a Gaussian distribution with mean μ_k, covariance Σ_k and the mixing coefficient π_k.

3.4.1. Maximum Likelihood for the Gaussian Mixture Model

Given a set of N independent and identically distributed observations $\{x_1, x_2, ...x_N\}$, the log likelihood function for a mixture of Gaussians is given by

$$\ln p(X|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln\left\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu,\Sigma)\right\} \tag{5}$$

The log of likelihood function avoids numerical underflow due to product of a large number of small probabilities [20].

3.4.2. Expectation Maximization for the Gaussian Mixture Model

The EM algorithm gives maximum likelihood estimates for GM in terms of the mean vector μ, the covariance matrix Σ and the mixing coefficients π as in equation (5) [20]. Multiple random initializations is one way to prevent EM from converging to local maxima for log likelihood function with multiple local maximum.

4. Experimentation

4.1. Datasets

Google Cluster trace (GCT: The trace contains resource usage of tasks for a seven-hour period [21]. Each job consists of multiple tasks and all tasks of a single job run on a single machine. The data set records time (in seconds) of the start of trace collection, a unique JobID, a unique TaskID, type of job (0, 1, 2, 3), average number of CPU cores and average memory used by each task. The CPU and memory utilizations are linearly transformed.

Business Critical Workloads of Bitbrains: This workload records usage details of 1750 VMs, organized into two traces, fastStorage and Rnd [22]. Experiments have been performed on the fastStorage data set, consisting of 1,250

VMs. CPU usage records number of virtual CPU cores provisioned, and capacity and usage of CPU, both in MHZ and in percentage. Memory usage is given by memory provisioned and actually used, measured in KBs.

The workloads are clustered for CPU usage and, CPU and memory usage. Cluster means obtained with K-Means and GMM are then compared for cluster analysis. The experiments are conducted in the Python programming language.

4.2. Feature Scaling

Feature scaling ensures that each feature contributes equally when estimating model parameters. Normalization and standardization are two feature scaling methods. Normalization scales features in the range [0, 1]. In the standardization, the features are centered with mean 0 and standard deviation 1. Normalization and standardization are given by equation (6) and (7), respectively.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{6}$$

$$x_{std} = \frac{x - \mu_x}{\sigma_x} \tag{7}$$

where x is a feature of the data points, x_{min} and x_{max} are the minimum and maximum values of x, respectively, and μ_x and σ_x are sample mean and standard deviation.

4.3. Key challenges with K-Means clustering

1. *Optimal Number of Clusters*: The Elbow method is a technique for finding number of clusters, k. In the plot of within-cluster SSE for different number of clusters, k is that value beyond which the distortions begin to approach a constant value. In scikit-learn, SSE is obtained by the *inertia* attribute of the KMeans model.
2. *Initial Centroids Selection*: Initial cluster centroids are determined using the K-Means++ algorithm, which is a technique for obtaining initial guesses for centroids [23]. An inappropriate choice of initial centroids may lead to bad cluster quality and slow convergence rates.
3. *Convergence Rate*: This can be controlled by setting a maximum number of iterations for EM. In scikit-learn, the K-Means algorithm stops if it converges earlier than the maximum number of iterations. In situations where K-Means does not converge, the algorithm stops when changes in within-cluster SSE is less than a tolerance value. In the experiments, this value is set to 1e-05.

4.4. Clustering with Gaussian Mixtures

Optimal number of Components: GMM is a generative model that gives a probability distribution for the data set. An optimal number of components avoids overfitting or underfitting and can be determined by evaluating the model likelihood using cross-validation or analytic criterion. The Akaike Information Criterion (AIC) and the Bayesian Information criterion (BIC) are analytic methods that estimate the goodness-of-fit of statistical models relative to each other for a given data set. They provide a quantitative measure of how general the model is, in terms of accuracy of representing future data using the process that generated the current data. AIC and BIC use a penalty for overfitting and under-fitting and this value is larger for BIC than that by AIC.

5. Results and Discussion

5.1. Google Cluster Trace

The normalized CPU core per task in GCT, is further normalized using equation (7) and converted as a fraction of 100, rounded to two places of decimal, to make the values more intuitive. Next, mean CPU core usage and mean memory usage for each job for its entire run, is computed. The scatter plots of Google cluster trace for CPU usage and CPU and memory usage in Figure 1 indicate that a very large number of jobs have low CPU and memory requirements,

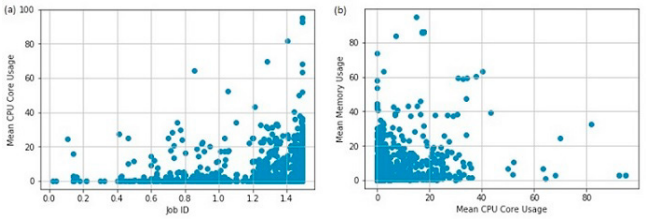

Fig. 1. Google Cluster Trace - Scatter Plot (a) CPU Usage (b) CPU and Memory Usage

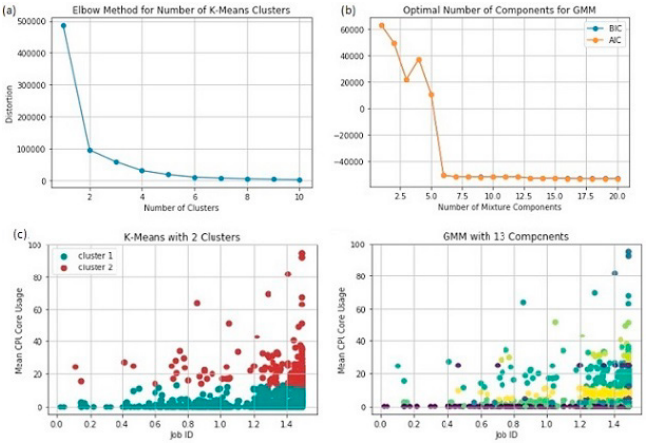

Fig. 2. Google Trace - Clustering with CPU Usage (a) Elbow Method for K-Means (b) AIC and BIC for GMM (c) Clustering with K-Means (d) Clustering with GMM

very few jobs have high CPU but low memory requirements and very few jobs have less CPU and high memory requirements.

5.1.1. Clustering with CPU Usage

Using the plots in Figure 2(a) and 2(b), the number of clusters for K-Means is set to 2, and 13 for GMM. Clustering with K-Means and Gaussian Mixtures are shown in Figure 2(c) and 2(d).

Table 1. Cluster means of CPU usage for Google Cluster Trace.

Clustering Method	Number of Clusters	Cluster Means
K-Means	2	2.60, 24.82
Gaussian Mixture Model	13	0.00, 1.45, 2.50, 5.00, 7.75, 12.50, 16.33, 25.00, 29.07, 33.44, 49.41, 66.44, 91.34

Usage means in Table 1 indicate that while K-Means learns maximum mean CPU core usage as 25 (approximately), GMM gives highest core usage as 91 (approximately). GMM is able to learn jobs with extremely low CPU core usage and identifies 13 different usage levels. K-Means identifies only two such groups of jobs. The details obtained with K-Means may be insufficient for meaningful characterization.

5.1.2. Clustering with CPU and Memory Usage

Number of K-Means clusters and GMM components, when clustering workloads based on CPU and memory usage is shown in Figure 3. The plot in Figure 3(a) shows a steep decrease in distortions till 4 clusters beyond which it approaches a constant value. Therefore, for K-Means the number of clusters is set to 4. Using Figure 3(b), number of GMM components is set to 18 as the AIC and BIC are constant for values higher than 17.5. Clustering with K-Means and GMM are shown in Figure 3(c) and 3(d).

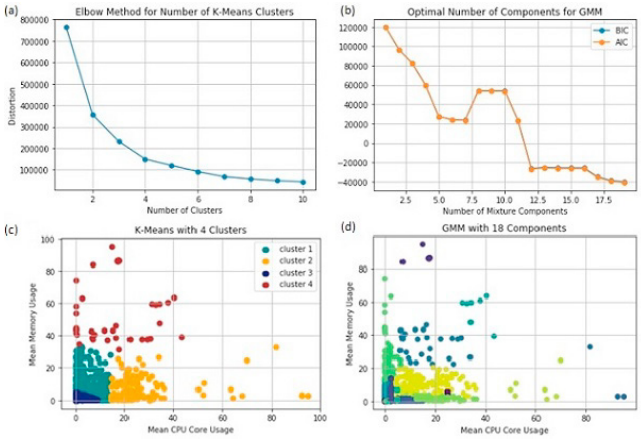

Fig. 3. Google Trace - Clustering with CPU and Memory Usage (a) Elbow Method for K-Means (b) AIC and BIC for GMM (c) Clustering with K-Means (d) Clustering with GMM

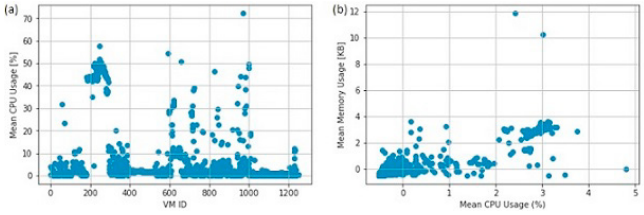

Fig. 4. Business Critical workloads - Scatter Plot: (a) CPU usage (b) CPU and Memory usage

From Table 2, CPU core and memory usage per job, as learned by K-Means, lie between [1.46, 0.62] and [25.05, 5.47]. The GMM technique identifies 18 different usage trends. Some jobs have nearly same CPU usage but differ in memory usage, as in the set of values ([5.00, 0.84], [5.00, 2.62], [5.00, 6.67]) and ([14.17, 1.83], [14.30, 10.08]). The usage trends are more clearly discovered with GMM as compared to that with K-Means.

Table 2. Cluster means of CPU and Memory usage for Google Cluster Trace.

Clustering Method	Number of Clusters	Cluster Means
K-Means	4	[1.46, 0.62], [4.23, 6.37], [15.26, 58.23], [25.05, 5.47]
Gaussian Mixture Model	18	[0.00, 0.19], [0.005, 2.5], [1.00, 0.56], [1.80, 21.88], [1.82, 5.18], [2.50, 3.91], [5.00, 0.84], [5.00, 2.62], [5.00, 6.67], [6.03, 0.98], [14.18, 1.83, [14.30, 10.08], [16.14, 30.81], [16.61, 86.58], [25.00, 5.3], [36.10, 55.76], [59.96, 8.00], [91.35, 8.86]

5.2. Business Critical workloads by Bitbrains

For the Bitbrains data set, mean usage of CPU and memory of each VM is computed. Figure 4(a) and 4(b) give scatter plot of CPU and, CPU and memory usage of 1250 VMs.

5.2.1. Clustering with CPU Usage

From the plot of distortions of the Elbow method in Figure 5(a), the optimal number of clusters for K-Means is 2. Both AIC and BIC in Figure 5(b) are minimum for fifteen mixture components. Beyond this value, the AIC is almost constant while BIC shows an increase. Also, for GMM with less than fifteen components, both AIC and BIC show high values. Therefore, fifteen mixture components were used to fit the GMM to the data set.

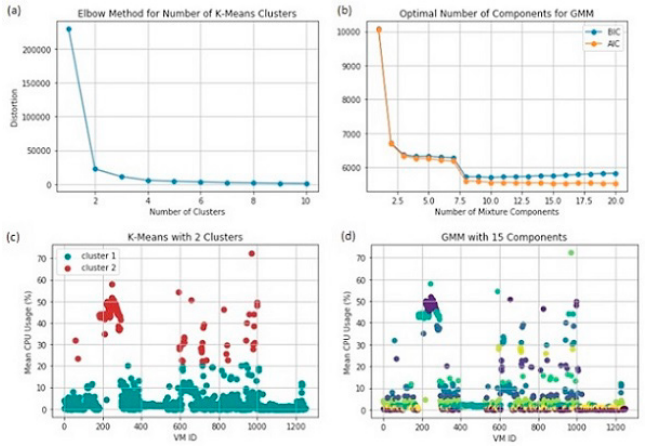

Fig. 5. Bitbrains Workload - Clustering with CPU Usage (a) Elbow Method for K-Means (b) AIC and BIC for GMM (c) Clustering with K-Means (d) Clustering with GMM

Table 3. Cluster means of CPU usage for Bitbrains Workload.

Clustering Method	Number of Clusters	Cluster Means
K-Means	2	2.3, 42.0
Gaussian Mixture Model	15	0.1, 0.8, 1.7, 3.3, 5.9, 9.4, 13.0, 20.2, 28.0, 31.2, 37.6, 43.5, 48.2, 56.0, 72.2

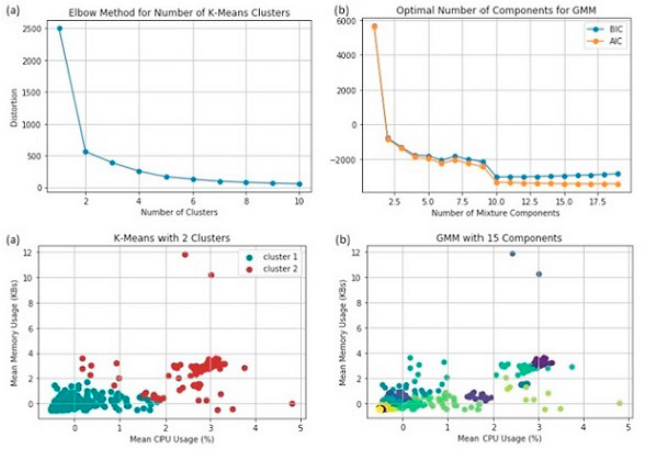

Fig. 6. Bitbrains Workload - Clustering with CPU and Memory Usage (a) Elbow Method for K-Means (b) AIC and BIC for GMM (c) Clustering with K-Means (d) Clustering with GMM

Table 3 shows that K-Means identifies two cluster means: 2.32% and 42.04%. This is a very abstracted view of CPU usage of the VMs, because the scatter plot in Figure 4(a) shows that there is a VM with mean CPU usage above 70%, some VMs have mean CPU usage between 10 and 40% which is not captured with just two clusters. The lone VM with CPU usage above 70% may be viewed as an outlier when considering the entire data set, but it is also important to be able to identify extreme resource requirements for informed capacity planning decisions. GMM gives a more comprehensive grouping of VMs, leading to a more meaningful clustering, with component-wise mean CPU usage ranging from 0.13% to 72.22%. These range of values clearly describe the varying CPU usage of different VMs.

Table 4. Cluster means of CPU and Memory usage for Bitbrains Workload.

Clustering Method	Number of Clusters	Cluster Means
K-Means	2	[2.6, 233179.26], [42.56, 3204834.23]
Gaussian Mixture Model	15	[0.11, 24605.12], [1.19, 114643.91], [2.55, 277393.46], [3.76, 692389.23], [6.33, 1189712.14], [9.46, 395735.23], [10.4, 1647517.52],[14.09, 3447012.51], [19.06, 760721.8], [29.94, 1136761.07],[43.4, 3675652.51], [43.5, 2228744.34], [43.91, 1295045761], [47.06, 1257120.34], [48.26, 4113764.52]

Table 5. Computation Time for Clustering.

Workload Trace	Clustering with CPU Usage (ms)		Clustering with CPU and Memory usage (ms)	
	K-Means	GMM	K-Means	GMM
Google	46.86	312.03	93.20	1218.04
Bitbrains	15.62	62.48	30.82	249.98

5.2.2. Clustering with CPU and Memory Usage

As shown in Figure 6(a) and 6(b), the optimal number of clusters is 2 for K-Means and 15 for GMM. Clustering with K-Means and GMM are shown in Figure 6(c) and 6(d). Cluster means in K-Means clustering is again too abstract and is given in Table 4 as (2.60%, 233179.26) and (42.56%, 3204834.23). With the GMM model, the component-wise means of CPU and memory usage are: VMs with very low CPU and memory usage - as in (0.11%, 24605.12) and (1.19%, 114643.91), VMs with low CPU and moderate memory usage - as in (2.55%, 277393.46) and (6.33%, 1189712.14), VMs with low CPU usage and high memory usage- as in (3.76%, 692389.23), (9.46%, 395735.23), (10.40%, 1647517.52), (14.09%, 3447012.51), (19.06%, 760721.80), and VMs with moderate CPU requirements and high memory usage - as in (29.95%, 1136761.07), (43.44%, 3675652.51), (43.45%, 2228744.34), (43.91%, 1295047.61), (47.06%, 1257120.34) and (48.26%, 4113764.52), which is indeed the case with the Bitbrains data set [2]. K-Means Clustering fails to discover these details.

5.3. Clustering Time

Table 5 shows the clustering time required by K-Means and GMM, given the number of clusters/components. As is clear from the table, K-Means requires much less time to discover and group the workloads into required number of clusters than required by GMM for corresponding number of Gaussian components. Although GMM takes more time than K-Means, the usage details are more descriptive of resource usage trends. As determined through experiments, K-Means provides very summarized usage information. Lack of finer usage details may lead to sub-optimal resource provisioning policies leading to serious implications in terms of unfulfilled resource requests, delayed resource provisioning and performance degradation of applications which in turn leads to SLA violations, customer dissatisfaction and even loss of clientele. With the clustering of workloads as generated by the GMM model, CPU and memory usage patterns are captured more clearly. Workload characterization that reveals workload characteristics with greater degree of details are crucial for intelligent placement of workloads with non-competing CPU usage and memory requirements. This results in better utilization of CPU and memory, and also a capacity plan that can accommodate unforeseen resource requests while maintaining performance of currently running services. But such details come at the cost of higher computation time.

6. Conclusion and Future Scope

This work compares the cluster effectiveness of K-Means and GMM, for characterization of Cloud workloads based on CPU and memory utilization. Euclidean distance, the within-cluster similarity measure in K-Means is unable to discover complex non-linear usage patterns. Membership of data points to clusters in GMM is probabilistic as against

the non-probabilistic, hard clustering approach of K-Means, thus resolving the membership ambiguities that may arise with overlapping clusters. The experiments reveal a more meaningful clustering of workloads with GMM than with K-Means, enabling a detailed characterization of resource usage requirements of Cloud workload. However, GMM is much slower as compared to clustering with K-Means.

Increasing heterogeneity due to rising Cloud adoption, emergence of new computing paradigms and new workloads requires identifying workload characteristics in precise terms, for effectively utilizing the infrastructure capacity and maintaining the competitiveness of Cloud providers in the Cloud infrastructure market. Gaussian Mixture Model can be used when more fine-grained workload behaviour analysis is required. This work will be further explored to derive a workload modelling and prediction framework for Cloud workload resource usage patterns.

References

[1] Varghese, B. and Buyya, R. (2018) "Next generation Cloud computing: New trends and research directions." *Future Generation Computer Systems* **79** pp.849-861.

[2] Shen, S., van Beek, V. and Iosup, A., 2015, May. Statistical characterization of business-critical workloads hosted in Cloud datacenters. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on* (pp. 465-474). IEEE.

[3] Pippal, S.K., Kumari, A. and Kushwaha, D.S., 2011, September. CTES based Secure approach for Authentication and Authorization of Resource and Service in Clouds. In *2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011)* (pp. 444-449). IEEE.

[4] Zhang, Q. and Boutaba, R., 2014, May. Dynamic workload management in heterogeneous Cloud computing environments. In *2014 IEEE Network Operations and Management Symposium (NOMS)* (pp. 1-7). IEEE.

[5] Pippal, S.K. and Kushwaha, D.S., 2013. A simple, adaptable and efficient heterogeneous multi-tenant database architecture for ad hoc Cloud. *Journal of Cloud Computing: Advances, Systems and Applications*, **2**(1), p.5.

[6] Menasce, D. A. and Almeida, V. A. F., 2001. Capacity Planning for Web Services: Metrics, Models, and Methods. *Prentice Hall*.

[7] Patel, E. and Kushwaha, D.S., 2019. Analysis of Workloads for Cloud Infrastructure Capacity Planning. In *Data and Communication Networks* (pp. 29-42). Springer, Singapore.

[8] Di, S., Kondo, D. and Cappello, F., 2014. Characterizing and modeling Cloud applications/jobs on a Google data center. *The Journal of Supercomputing*, **69**(1), pp.139-160.

[9] S. Di, D. Kondo, and F. Cappello, "Characterizing Cloud applications on a google data center," in Parallel Processing (ICPP), *2013 42nd International Conference on. IEEE*, 2013, pp. 468–473.

[10] Moreno, I.S., Garraghan, P., Townend, P. and Xu, J., 2014. Analysis, modeling and simulation of workload patterns in a large-scale utility Cloud. *IEEE Transactions on Cloud Computing*, **2**(2), pp.208-221.

[11] Wamba, G.M., Li, Y., Orgerie, A.C., Beldiceanu, N. and Menaud, J.M., 2017, October. Cloud workload prediction and generation models. In *Computer Architecture and High Performance Computing* (SBAC-PAD), 2017 29th International Symposium on (pp. 89-96). IEEE.

[12] Cui, H., Ruan, G., Xue, J., Xie, R., Wang, L. and Feng, X., 2014, May. A collaborative divide-and-conquer K-means clustering algorithm for processing large data. In *Proceedings of the 11th ACM Conference on Computing Frontiers* (p. 20). ACM.

[13] Lu, Y., Liu, L., Panneerselvam, J., Zhai, X., Sun, X. and Antonopoulos, N., 2019. Latency-based Analytic Approach to Forecast Cloud Workload Trend for Sustainable Datacentres. *IEEE Transactions on Sustainable Computing*.

[14] Wamba, G.M. and Beldiceanu, N., 2018, September. CP-based Cloud workload annotation as a preprocessing for anomaly detection using deep neural networks. In *International Conference on Time Series and Forecasting*.

[15] Patel, E., Mohan, A. and Kushwaha, D.S., 2018, November. Neural Network Based Classification of Virtual Machines in IaaS. In *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)* (pp. 1-8). IEEE.

[16] Panda, R. and John, L.K., 2014, December. Data analytics workloads: Characterization and similarity analysis. In *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)* (pp. 1-9). IEEE.

[17] Khoshkbarforoushha, A., Ranjan, R. and Strazdins, P., 2015, September. Resource Distribution Estimation for Data-Intensive Workloads: Give Me My Share & No One Gets Hurt!. In *European Conference on Service-Oriented and Cloud Computing* (pp. 228-237). Springer, Cham.

[18] Zhu, H., Dai, H., Yang, S., Yan, Y. and Lin, B., 2017, November. Estimating Power Consumption of Servers Using Gaussian Mixture Model. In *2017 Fifth International Symposium on Computing and Networking (CANDAR)* (pp. 427-433). IEEE.

[19] Tan, P.N., Steinbach, M. and Kumar, V., 2006. Cluster analysis: basic concepts and algorithms. Introduction to data mining, **8** (pp.487-568).

[20] Bishop, C.M., 2006. Pattern recognition and machine learning. springer.

[21] https://github.com/google/cluster-data/blob/master/TraceVersion1.md

[22] http://gwa.ewi.tudelft.nl/datasets/Bitbrains

[23] Arthur, D. and Vassilvitskii, S., 2007, January. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.