Avdanced ECHMM-Based Machine Learning Tools for Complex Applications

Alfredo Cuzzocrea
DIA Dept.
University of Trieste and ICAR-CNR
Trieste, Italy
alfredo.cuzzocrea@dia.units.it

Enzo Mumolo
DIA Dept.
University of Trieste
Trieste, Italy
mumolo@units.it

Gianni Vercelli
DIBRIS Dept.
University of Genova
Genova, Italy
gianni.vercelli@unige.it

***Abstract*—We present a novel approach for accurate characterization of workloads. Workloads are generally described with statistical models and are based on the analysis of resource requests measurements of a running program. In this paper we propose to consider the sequence of virtual memory references generated from a program during its execution as a temporal series, and to use spectral analysis principles to process the sequence. However, the sequence is time-varying, so we employed processing approaches based on Ergodic Continuous Hidden Markov Models (ECHMMs) which extend conventional stationary spectral analysis approaches to the analysis of time-varying sequences.**

In this work, we describe two applications of the proposed approach: the on-line classification of a running process and the generation of synthetic traces of a given workload. The first step was to show that ECHMMs accurately describe virtual memory sequences; to this goal a different ECHMM was trained for each sequence and the related run-time average process classification accuracy, evaluated using trace driven simulations over a wide range of traces of SPEC2000, was about 82%. Then, a single ECHMM was trained using all the sequences obtained from a given running application; again, the classification accuracy has been evaluated using the same traces and it resulted about 76%. As regards the synthetic trace generation, a single ECHMM characterizing a given application has been used as a stochastic generator to produce benchmarks for spanning a large application space.

***Index Terms*—Workload characterization, HMM, cepstral coefficients.**

I. Introduction

Performance evaluation of computer systems requires to test different alternatives under identical conditions. However, a real computing environment is generally not repeatable, and for this reason it is necessary to characterize the workload by developing a workload model that can be used repeatedly. Once a workload model is available, changes in the workload and in the system can be studied under controlled conditions.

As pointed out in [8], workload characterization using a model plays a fundamental role in many areas, namely to understand the key resource usage of applications, to tune computer architectures, to validate trace reduction mechanisms, to guide the selection of programs for obtaining benchmark sets, to generate synthetic traces to span application spaces, and to create abstract program behavior models for performance studies of computer systems.

Workloads are typically modeled as stochastic processes and analyzed with statistical techniques [10] [11]. This is because different benchmarks are obtained from a single application for different inputs, and the only way to describe all the potential application space is through the extraction from the running application of suitable parameters which describes the main features of the workload.

A running application thus produces a huge amount of data; the only way to analyze such data is by means of statistical techniques. In this paper we propose to use ergodic Hidden Markov Models as statistical models of workloads. Our approach is based on the idea to treat the sequences of memory page references produced by a running application as time-varying discrete-time series of data and to analyze them with statistical techniques using spectral parameters. The proposed methodology operates as follows: the page references sequences obtained from a running application is divided into segments of some hundreds of page numbers, and each piece is then described with a vector of spectral parameters. Chunks of references are formed by some hundreds of such vectors; the chunks are then used to estimate the parameters of a Hidden Markov Model. Repeating this operation for each running application, we compute a HMM model of the application. The accuracy of such models has been estimated as quite good.

By considering a number of workloads obtained from the same type of application, and re-estimating the parameters of a single Hidden Markov Model, a statistical model of that type of application can be computed. In this way, we have obtained models for several application types, as described in I-A. In this paper, models have been used in two ways: to determine to which application type belongs a running application and to generate synthetic traces. Both these points are very important from a computer architecture perspective. As regards the benchmark classification, it is important to note that using our approach the classification is possible in run-time, i.e. during the application execution, since the computational complexity is quite low. As regards the synthetic traces generation, HMMs can indeed be viewed as generators of observations, in our case allowing to cover a large application space for computer architecture studies and designs [12].

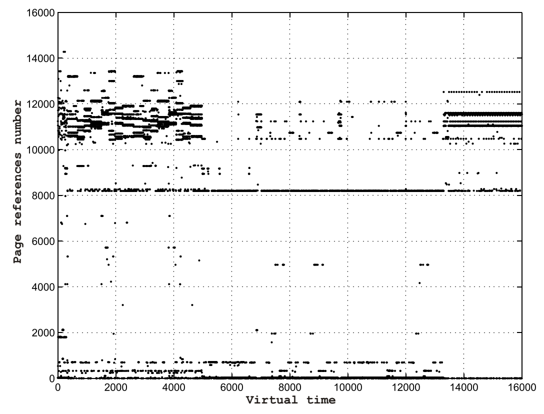

Fig. 1. Graphical view of a portion of a sequence of page references.

A. Methodology

We used traces-driven simulations to test the proposed approach. The traces are obtained from the SPEC2006 benchmark suite [9], as reported in Tab. I.

Benchmark	Category
464.h264ref	Video coding
403.gcc	C Compiler
400.perlbench	Interpreter
401.bzip2	File Compressor
445.gobmk	Go Game Playing
429.mcf	Vehicle Scheduling
456.hmmer	Profile Hidden Markov Models
458.sjeng	Chess Playing
462.libquantum	Quantum Computer Simulation
471.omnetpp	Discrete Event Simulation
473.astar	Path-finding Library
483.xalancbmk	XML to HTML

TABLE I
THE WORKLOADS USED IN THIS WORK.

CPU address traces have been obtained by running the applications of Tab. I with different input data; several executions of each application have been considered. The applications of Tab. I run on a Intel Core i7 quad-core 2,8 GHz, under Linux operating system. In Fig. 1 a part of a page references trace (16000 virtual time instants) is shown. This figure illustrates the time-varying characteristic of the trace. From Fig. 1, it clearly follows how modeling and capturing such phenomena represents a challenging problem for modern machine learning applications.

The rest of this paper is organized as follows. Section II reports some significative papers related to this work. In Section III the operation principles of the algorithm described in this paper are described together with the used parameters. In Section IV the workload classification methodologies based on HMM are described while in Section V the generation of synthetic traces with HMMs is briefly reported. Finally, in Section VI some final remarks are reported. This paper significantly extends the copyright-free paper [21], where the proposed methodology has been firstly introduced.

II. Related Work

A Markov Model is a classical workload characterization tool. Markov Models contain a number of system states linked by probabilistic transitions. A simple way to use a Markov Model for describing a process behaviour during execution is to make states equal to the software resources used by the process and transitions probabilities between states equal to the actual use of the resources during execution.

Song *et al.* propose a Markov Chain based model of a Parallel computer workload [1]. The workload is described in terms of the allocation of the process nodes to the jobs. Two independent Markov Chains are used, one for modeling the sequence of the requested processing nodes and for modeling the runtimes required. The authors describe algorithms for combining the two Markov Chains and for computing the correlation between them. The quality of the model is tested by generating synthetic traces and comparing them with the original one.

Normally the system states of a Markov Model are directly observable. However, in many practical cases we can observe only values which are functions of the states. A Markov Chain where the states are not observable is an Hidden Markov Model (HMM). HMM have been successfully used in a huge number of practical applications, including Workload modeling and Performance Evaluation [2].

[3] trains an HMM for finding the type of application workload from Network File System (NFS) file system operations and their parameters by unknown processes.

[4], instead, uses HMM for finding the network protocol used by a running application from packet size and destination.

[5] estimates an HMM from I/O data gathered from storage operations. The authors then use the HMM to create representative traces to test the performance of storage systems. Synthetic traces are very useful to drive simulation based performance studies without the need to acquire suitable traces. In particular, in [5] the synthetic traces are used to study Flash memories performance.

Khan *et al.* proposes in [6] an HMM based model for characterizing and predicting Virtual Machines workload. The purpose of this paper is workload characterization and prediction of computing resources in cloud computing.

[7] propose an adaptive HMM algorithm for estimating statistical models of storage systems. The algorithm has been tested with several traces formed by a time stamp and I/O commands like read/write. Also in this case the authors use the HMM model to generate synthetic traces of storage system. This proposal embeds interesting ideas, as the whole HMM-based framework can be extended to these metaphors (e.g., [22]) as to gain *flexibility* in the main processes.

Of course a process is in several states during execution. The states can be initialization, I/O, computation. During computation there are sequential, periodic or random states.

In our paper we estimate an HMM from parameters obtained from memory address reference generated during process execution. The observable values are memory reference addresses and the HMM states are the states of the process in execution. We cannot know the exact meaning of the HMM states exactly; we can only experimentally find what is the best number of states.

III. Hidden Markov Models for Workload Classification

A. Operating Principles

The sequence of memory address references is transformed to a virtual page sequence. The Virtual page sequence is divided into 4096 pages frames. For each frame a cepstral vector is computed and from each vector the4 first ten cepstral coefficients are given as input to HMM. For Discrete HMM computation the cepstral coefficients are vector quantized. An Hidden Markov Model is normally designed as $\lambda = (A, B, \pi)$ where A is the transition probability matrix among states, B is the emission probability distribution and π is the state initial probability distribution. Given an unknown observation O, an HMM is evaluated by computing the likelihood $P(O|\lambda)$.

B. HMM Features

The page references are produced at a CPU instruction clock rate, because each virtual memory address is translated to a virtual page reference. This information rate is too high to make reasonable workload evaluations, and consequently the number of page references is too large. Therefore, some feature extraction must be performed for getting rid of the redundant information and for reducing the data rate. According to the idea of considering the page references sequence as a signal, we use a spectral description of the page references sequences. Characteristics in the sequences, such as for examples loops or sequential program behaviors, are indeed described in the spectrum. For instance, loops introduce peaks in the spectrum while a sequential address sequence produces a DC component. For example, representing the sequence of Fig. 1 in the log spectral domain, we obtain the data shown in Fig. 2.

Since the page references sequence is time varying, as suggested in Fig. 1, the result of Fig. 2 is obtained with short-time spectral analysis. In particular, the sequence of virtual memory pages is divided into short sections – 120 references long – and analyzed by means of a discrete Fourier transform.

It is worth noting that Fig. 2 reports a log-spectral view of the page references trace shown in Fig. 1. In Fig. 2 it is possible to see how the change of behavior in the trace of Fig. 1 at about 5000 virtual time instants reflects in the spectral domain. As in the proposed approach a fundamental issue is related to the comparison between log-spectral data, it is important to define a log-spectral distance between two spectra. To show how to define the log-spectral distance, let

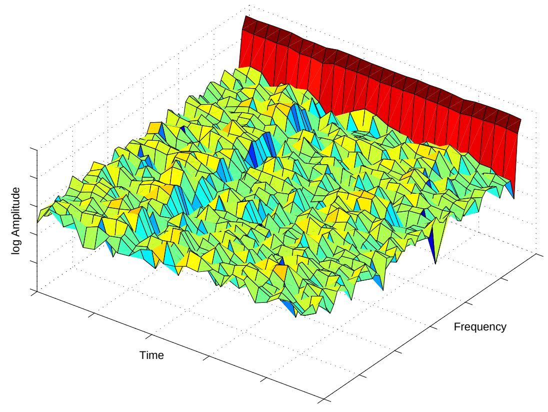

Fig. 2. Log-spectral data of the portion of the page references sequence shown in Fig. 1.

us start with the Euclidean distance definition between the log spectra of two sequences, x_n and y_n:

$$\begin{aligned} e(\omega) &= log\,|X(\omega)|^2 - log\,|Y(\omega)|^2 = \\ &= 2(log\,|X(\omega)| - log\,|Y(\omega)|) = \\ &= 2Re\left[log\,(X(\omega)) - log\,(Y(\omega))\right] \end{aligned}$$

where $X(\omega) = \sum_{n=-\infty}^{+\infty} x_n e^{j\omega n}$ is the spectrum of the x_n sequence. On the other hand, $\log(X(\omega)) = \sum_{n=-\infty}^{+\infty} c_n e^{j\omega n}$ where c_n is the cepstrum sequence [14] which is obtained applying an inverse Fourier transform to the log spectrum of the input page references sequence. Hence, calling c_n^x and c_n^y the cepstrum of the x_n and y_n sequences respectively,

$$e(\omega) = 2Re\left[\sum_{n=-\infty}^{+\infty} (c_n^x - c_n^y)\, e^{j\omega n}\right] = \sum_{n=-\infty}^{+\infty} (c_n^x - c_n^y)\, e^{j\omega n}$$

because the c_n sequences are symmetrical since the input reference page sequence is real. Finally, the spectral distance between two sequences x_n and y_n is

$$d(X, Y) = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^2(\omega)d\omega = \sum_{n=-\infty}^{+\infty} (c_n^x - c_n^y)^2\,.$$

In conclusion, the spectral distance between the log spectra is simply the Euclidean distance between the cepstal sequences.

On the basis of this consideration, we described the page references sequences with cepstral coefficients. In Fig. 3 the cepstral representation of the page references sequence of Fig. 1 is reported. As shown in Fig. 2 the change of trace behavior at about 5000 time instants is reflected in the cepstral domain. In fact, the slow spectral characteristics are seen in the part around zero in the cepstral domain, in Fig. 3 we can see that the initial part of the cepstrum is more spiky around zero reflecting in this way the change of trace behavior seen in Fig. 1. On the basis of that, it is useless to consider all the cepstral coefficient to represent traces; for this reason we used only the first 10 cepstral coefficients.

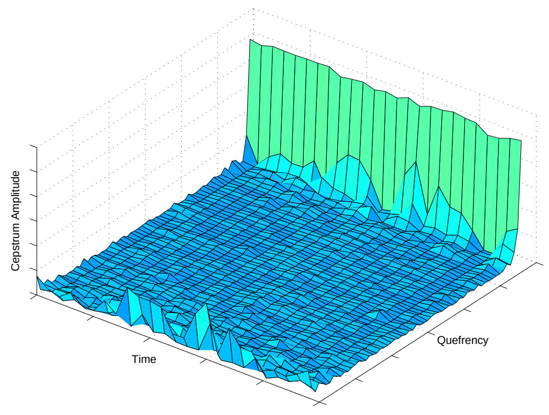

Fig. 3. Cepstral description of the portion of the page references sequence shown in Fig. 1.

IV. Workload Classification

For dynamic characterization of processes, the address field of the traces has been extracted. In this way we have obtained a sequence of virtual addresses generated by the processor during the execution of the processes. For converting the trace of addresses into trace of virtual pages, the sequence of addresses has been divided by the page dimension, which we set to 4096 pages.

Once the sequence of virtual pages has been obtained from every trace and thus for every process, we have tried to use discrete HMMs for their classification. Even if the sequence of pages is a discrete sequence, it can not be used for processes classification using discrete HMMs, as it contains a too high number of symbols.

In order to face this problem, the sequence of virtual pages has been turned into a sequence of few symbols, without loosing meaningful data. The sequence of virtual pages has been turned into sequence of cepstral coefficients by the short time analysis process described in Sec. III-B.

A. Single Trace Classification

The sequences of cepstral coefficients are real number sequences. For analyzing cepstral sequences using a discrete HMM, vector quantization is needed. In this process some degradation is introduced and the training lacks its efficiency.

A continuous HMM can use an input sequences of 10-dimensional cepstral vectors and vector quantization is not needed. The results obtained in this way usually perform better than using the discrete model.

The multivariate Gaussian density is used for describing the cepstral observation. The 10-dimensional cepstral vector is described using a multivariate density having 10 dimensions, and it is specified by means of the mean and covariance matrixes. Using this approach it is supposed that the 10-cepstral coefficients are uncorrelated and so the covariance matrix is diagonal.

In order to choose the optimum number of states and the topology of the HMMs, several tests have been performed. The number of states needed for continuous HMM is lower than in the discrete ones. Considering topology, ergodic models score better results.

In Fig. 4 the mean classification rate versus the number of states for ergodic and left-right models is depicted.

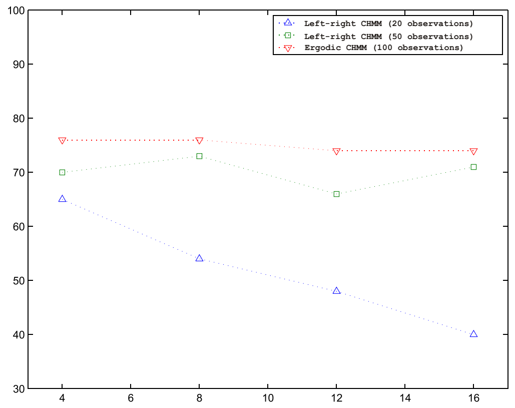

Fig. 4. Mean classification rate versus the number of states of ergodic and left-right models.

As the models using 4 states provides better results using a lower number of observation, we have repeated experiments using this configuration increasing the number of observations.

Using 100 observations for every model, in the ergodic case the recognition mean of single traces is about 82%, in the left-right case this mean is 65%. In Fig. 5 and in Fig. 6 these results are depicted, gathering the traces per workload and computing for every traces group the mean recognition rate.

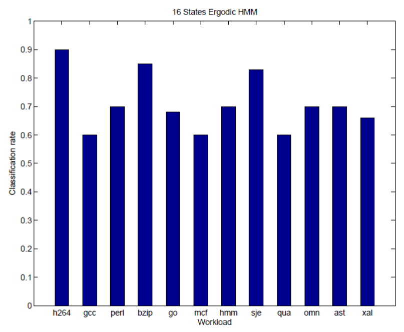

Fig. 5. Average classification rate for all the traces using 16-state ergodic discrete HMMs.

The ergodic continuous HMMs have been trained using 100 observations for every model. The recognition rate varying the number of states and using all the traces is reported in Fig. 6.

As it is shown in Fig. 5 and Fig. 6, four states give better results than sixteen states. In Fig. 7 is reported the classification rate using left-right continuous HMM with 4 states.

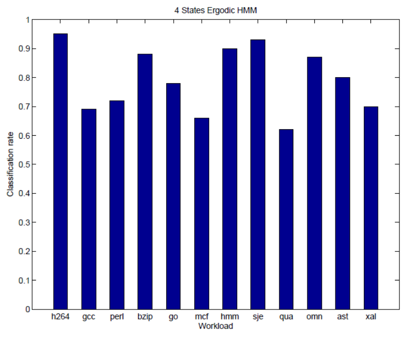

Fig. 6. Average classification rate for all the traces with 4-state ergodic continuous HMMs.

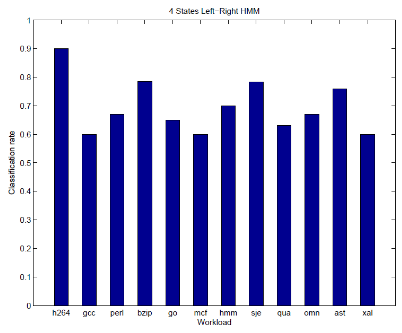

Fig. 7. Average classification rate for all the traces with 4-state left-right continuous HMMs.

The results obtained using such statistical models demonstrated the effectiveness of this dynamic processes modeling approach. Cepstral coefficient obtained from the virtual pages sequences are a good parameter for describing traces of programs during execution.

B. Program Behavior Modeling

Dynamic classification of the traces, taking as parameter the virtual pages, has obtained satisfactory results. As seen in IV-A, the traces of a single application have been obtained processing such application with different inputs, or processing different functions of the same program.

Then, we have classified the workloads, gathering the traces of the same workload using a single HMM trained with several traces representing the same workload.

Using several traces of the same workload for incremental training, it is possible to classify the program behavior. Of course the performances are lower than the Single Trace Classification. The performance obtained with ergodic discrete and continuous HMM, are reported in Fig. 8 and in Fig. 9.

The mean accuracy results obtained in the case of ergodic discrete and ergodic continuous HMMs are reported in Tab. II: ergodic continuous models obtain better classification accuracy than the discrete ones.

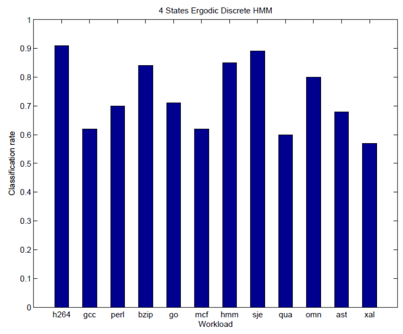

Fig. 8. Program behaviour classification using ergodic discrete HMM.

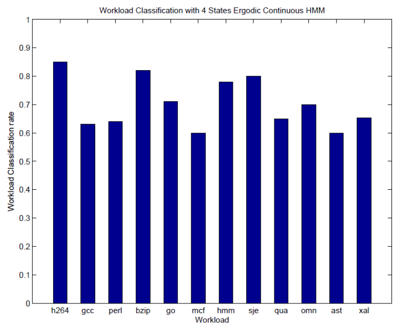

Fig. 9. Program behaviour classification using ergodic continuous HMM.

V. Synthetic Trace Generation

A Hidden Markov Model can be used as a generator of a stochastic process. In our case the HMM generates vectors of cepstral coefficients. However, the cepstral coefficients can be inverted to give page numbers as shown in Fig. 10.

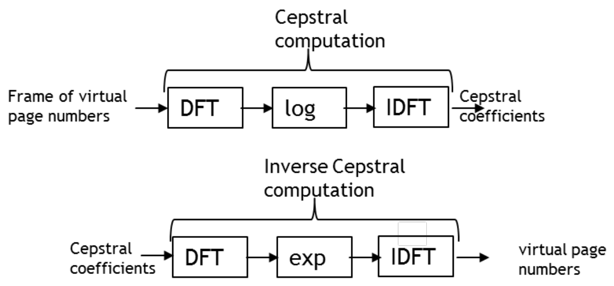

Fig. 10. Inversion of cepstral coefficients to page numbers.

The procedure is the following:

	Ergodic Discrete HMM	Ergodic Continuous HMM
Mean Accuracy	64%	70%

TABLE II
Mean program behaviour classification accuracy.

1) Choose an initial state i according to the initial distribution π.
2) Set $t = 1$.
3) Generate a N-dimensional random variable according to the characteristic of the multivariate Gaussian distribution in state i.
4) Perform a state transition according to the transition probabilities $a_{i,j}$.
5) Set $t = t + 1$. If $t < T$ go to 3, else terminate.

The random variable generated in step 3 is a vector of cepstral coefficients. This vector must be inverted to obtain a set of page references.

A result is reported in Fig. 11, where the log-spectral data of a synthetic trace produced with the above procedure and the HMM trained with the trace of Fig. 1 is reported. Fig. 11

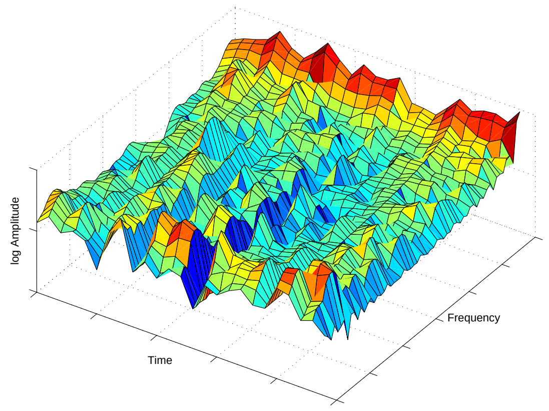

Fig. 11. Example of synthetic trace generated using a continuous ergodic HMM represented in the spectral domain.

should be compared with Fig. 2.

VI. Conclusions and Future Work

In this paper we describe an approach for workload characterization using ergodic hidden Markov models. The page references sequences produced by a running application are divided into short virtual time segments and used to train a HMM which models the sequence and is then used for run-time classification of the application type and for synthetic traces generation. The main contribution of our approach are on one hand that a run-time classification of the running application type can be performed and on the other hand that the applications behavior are modeled in such a way that synthetic benchmarks can be generated. Using trace-driven simulation with SPEC2000 benchmarks, the mean classification rate is about 82% for each traces and about 76% using a single HMM to model a single application type. Many future developments of our approach are possible since what we propose in this paper – to use time-varying non-linear processing techniques to treat sequences produced by programs during execution – is a novel approach in computer architecture studies. In addition to this, we believe that another interesting line of research is represented by the adaption of the proposed framework to novel *big data trends* (e.g., [18]–[20]).

References

[1] B. Song, C. Ernemann and R. Yahyapour, *Parallel Computer Workload Modeling with Markov Chains*. Job Scheduling Strategies for Parallel Processing, 10th Int. Workshop, pages 47–62, 2004

[2] E. de Souza e Silva, R.M. Meri Leão and R.R. Muntz, *Performance Evaluation with Hidden Markov Models*, Performance Evaluation of Computer and Communication Systems, Int. Workshop, Vienna, Austria, pages 112–128, 2010

[3] N. J. Yadwadkar, C. Bhattacharyya, K. Gopinath, T. Niranjan and S. Susarla, *Discovery of Application Workloads from Network File Traces*, 8th USENIX Conference on File and Storage Technologies, pages 183–196, 2010

[4] T. Umut, *Hidden Markov models to analyze user behaviour in network traffic*. Technical report, Bilkent University, Ankara, Turkey, 2005.

[5] P. G. Harrison, S. K. Harrison, N. M. Patel and S. Zertal, *Storage workload modelling by hidden Markov models: Application to Flash memory*, Perform. Eval., vol.69, n.1, pages 17-40,2012

[6] A. Khan, X. Yan, S. Tao and N. Anerousis, *Workload characterization and prediction in the cloud: A multiple time series approach*, IEEE Network Operations and Management Symposium, pages 1287–1294, 2012

[7] T. S. Chis and P. G. Harrison, *iSWoM: The Incremental Storage Workload Model Based on Hidden Markov Models*, 20th Int. Conference on Analytical and Stochastic Modelling Techniques and Applications, pages 127–141, 2013

[8] R. Han, S. Zhan, C. Shao, J. Wang, L. K. John, J. Xu, G. Lu, Lei Wang, *BigDataBench-MT: A Benchmark Tool for Generating Realistic Mixed Data Center Workloads*. Lecture Notes in Computer Science 9495, Springer 2016.

[9] Standard Performance Evaluation Corporation, *https://www.spec.org/cpu2006/*.

[10] M.C. Calzarossa, L.Massari and D. Tessera, *Workload Characterization: A Survey Revisited*. ACM Comput. Surv., 48, 3, Page(s):1–43, 2016.

[11] K.J. McDonell, *Benchmark Frameworks and Tools for Modelling the Workload Profile*. Performance evaluation 22(1), Page(s):23–42, 1995.

[12] J.P. Singh, H.S. Stone, D.F. Thiebaut, *A Model of Workloads and Its Use in Miss-Rate Prediction for Fully Associative Caches*. IEEE Transactions on Computer, vol.41(7), 1992.

[13] L.R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Proceedings of the IEEE, vol.77(2), 1989.

[14] L.R. Rabiner, B.H. Juang, *Foundamentals of Speech Recognition*. Prentice Hall Signal Processing Series, 1993.

[15] Y. Bengio, *Markovian Models for Sequential Data*. Neural Computing Surveys 2, 1999.

[16] Y. Bengio, *Markovian Models for Sequential Data*. in "Machine Learning for Audio, Image and Video Analysis: Theory and Applications", Springer London, Page(s):265–303, 2008.

[17] J.A. Bilmes, *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*. Technical Report, TR-97-021, 1997.

[18] A. Cuzzocrea, *Privacy and Security of Big Data: Current Challenges and Future Research Perspectives*. ACM PSBD 2014 Proceedings, 2014.

[19] A. Cuzzocrea, U. Matrangolo, *Analytical Synopses for Approximate Query Answering in OLAP Environments*. DEXA 2004 Proceedings, 2004.

[20] A. Cuzzocrea, G. Fortino, O.F. Rana, *Managing Data and Processes in Cloud-Enabled Large-Scale Sensor Networks: State-of-the-Art and Future Research Directions*. CCGRID 2013 Proceedings, 2013.

[21] A. Cuzzocrea, E. Mumolo, G. Vercelli, *Ergodic Hidden Markov Models for Workload Characterization Problems*. DMSVLSS 2017 Proceedings, 2017.

[22] M. Cannataro, A. Cuzzocrea, A. Pugliese, *XAHM: An Adaptive Hypermedia Model based on XML*. SEKE 2002 Proceedings, 2002.