Optimal Functionality and Domain Data Clustering based on Latent Dirichlet Allocation

Stoyan Garbatov and João Cachopo
Software Engineering Group
Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento, INESC-id
Lisbon, Portugal
stoyangarbatov@gmail.com and joao.cachopo@ist.utl.pt

***Abstract* — This work presents a new approach for clustering domain data and application functionality, based on the Latent Dirichlet Allocation. The methodology, developed here, performs an optimal clustering by identifying input values that lead to the best possible clustering output. The optimal solutions are identified through the use of the Silhouette technique. A validation of the work is performed based on the TPC-W benchmark. The new approach is flexible enough to be applied to any object-oriented application where identifying meaningful clusters of its domain data and functionality is desired.**

Keywords-clustering; Latent Dirichlet Allocation; stochastic model; Silhouette.

I. INTRODUCTION

The problem of clustering has been considered and analysed in many different disciplines' contexts, illustrating its relevance and usefulness in a variety of circumstances.

Clustering corresponds to an unsupervised classification of patterns (data, observations, etc) into sets or groups (clusters). Clustering algorithms organize pattern aggregates based on similarity criteria according to which these may be classified.

A pertinent situation requiring clustering would be in the context of large-scale object-oriented applications (e.g dynamic content web applications). There, it can be interesting to identify meaningful subsets of application functionality that display high affinity with regards to the domain data that is manipulated within their scope. If such information is available, then it may be feasible to carry out techniques such as load balancing or partial data replication to improve the application performance and scalability.

Based on what can be seen from recent research, there has been some effort spent in this area, but the great majority of these approaches display only partial automation in their mode of operation. Many approaches require user intervention at one, if not more, points of the analysis procedure, making the approaches more prone to errors and leading to non-optimal results, due to the subjectivity induced by the user interaction in the decision making process.

In contrast to these supervised approaches, we believe that a wholly automatic approach would lead to better results, by avoiding the problems identified above. This paper describes the development and validation of a fully automated system capable of identifying the domain data manipulated during the execution of the target system's functionality and, based on that information, of performing optimal partitioning of the application's methods/services according to the domain data used within their runtime scopes. The partitioning of the application's functionality (represented by its services and/or methods) is performed by employing the Latent Dirichlet Allocation [1]. The optimality of the solutions is guaranteed through the use of the Silhouette technique, [2].

The article has the following structure. The related works are discussed in Section II. The description of the system is covered in Section III. The results and evaluation of the system are given in Section IV. The concluding remarks are presented in Section V.

II. RELATED WORK

Based on the nature of the work presented here, it is possible to identify two related research areas. The first one covers the development and analysis of clustering algorithms, whilst the second one encompasses works seeking to develop performance improvement techniques in the context of dynamic content web applications.

It was not possible to find any work that takes an at least comparable approach for the problem at hand. As such, the discussion of works strictly related to clustering algorithms will be restricted to the relevant references that are present in the system description.

It is important to discuss some of what has been done in the context of dynamic content web applications [3-8], so as to better appreciate the contribution of the current work. A rather thorough study and comparison of load balancing and scheduling strategies, for the type of applications identified above, can be seen in the work of Amza et al. [9] .

The work of Elnikety et al. [7] introduced a memory-aware load balancing method for dispatching transactions to replicas in systems employing replicated databases. The algorithm uses information about the data manipulated in transactional contexts with the goal of assigning transactions to replicas so as to guarantee that all necessary data for their execution is in memory, thereby reducing disk I/O. For guiding the load balancing technique, the authors developed an auxiliary approach for estimating the volume and type of data manipulated during transactions. An additional

 ISBN: 978-1-61208-165-6

contribution of their work is an optimization designated *update filtering* for decreasing the overheads due to the propagation of updates between replicas.

The work of Amza et al. [5] presents a novel lazy replication technique, intended for scaling database back-ends of dynamic content site applications operating on top of computer clusters. The approach developed by Amza et al. is referred to as conflict-aware scheduling and provides throughput scaling and one-copy serializability. This technique exploits the fact that, in the context of database clusters, there is a scheduler responsible for processing all incoming requests. By making use of information regarding the domain data accessed within transactions, Amza et al. [3] developed a conflict-aware scheduler that provides one-copy serializability, as well as reducing the rate at which conflicts occur. This is achieved by guiding incoming requests to nodes based on the data access patterns that are expected to be performed during the execution of the associated transactions.

Gao et al. [6] developed an edge service replication architecture for e-commerce applications using application specific distributed objects. The authors exploit application specific behaviour to manage subsets of shared domain data through distributed objects. Higher system availability and efficiency is achieved by tolerating lower consistency among distributed objects.

Shen et al. [4] performed an analysis over the clustering of replicated services with high ratios of write operations. With their work, the authors developed an infrastructural middleware called Neptune, which allows the agglomeration and replication of a system's service modules. The middleware supports multiple alternative persistence mechanisms and is capable of maintaining consistency dynamically, independently of the location and availability of a particular replica.

Zuikeviciute and Pedone present in [8] a hybrid approach for conflict-aware load balancing for systems with database replication. The authors analyzed the effects of the often opposing requirements (from an engineering point of view) of maximizing transaction parallelism and minimizing conflict ratios. The work led to the development of a load balancing technique that finds a good compromise between parallelism and conflict minimization, accomplishing better results than approaches concentrating solely on one of the above requirements.

As can be seen from the above works, there are indeed very promising results for improving the performance and scalability of large scale applications, through the use of load balancing, replication techniques, adaptive scheduling, and other related approaches. Yet, there is still significant room for improving the full automation of existing solutions, both at the level of analyzing the behavior of target applications, as well as in the identification of meaningful functionality and domain data subsets on which the approaches are to be applied.

Thus, we believe that a system capable of performing a completely automated analysis of a target application's behavior (with regard to domain data manipulations performed in runtime), and of performing an optimal clustering of the application's functionality and domain data (through the use of the current state-of-the-art multivariate clustering algorithm), would constitute an important contribution within this research area.

III. SYSTEM DESCRIPTION

The system developed with this work is composed of two parts: a data acquisition and analysis module and an optimizing clustering module. The first module is responsible for capturing the target application behaviour, for analysing it, and for generating predictions about what are the most likely domain data types to be needed by the application when it is in a specific execution context (e.g., method, service, etc). The full description of the implementation, functionality, and properties of this module has already been presented and discussed in detail in [10-12]. The prediction functionality is of no relevance for the work presented here. The key aspects of this module are that it provides the input necessary for the optimal clustering module, and that the data collection task performed is done with relatively low overheads (an average of 5-8% overheads in comparison with the original version of the target application performance), in an online fashion. Moreover, all modifications necessary for the acquisition of the behavioural data are performed in a completely automated manner by the system presented here.

The second module is responsible for identifying the optimal clustering of the target application's functionality and domain data, based on the data access pattern behaviour observed in runtime. For the clustering itself, we use the Latent Dirichlet Allocation algorithm, while the optimal clustering solution is guaranteed through the use of the Silhouette technique. Both of them shall be discussed in detail in the following subsections.

A. *Latent Dirichlet Allocation*

The data acquisition module is responsible for supplying the clustering module with the observed target application conduct. This corresponds to the application's domain data access behaviour, and is expressed in terms of the frequencies of the domain object manipulation operations observed when executing application *functionality*. For simplicity, the abstraction capturing this functionality shall be referred to as the *methods* of the application, but any other appropriate concept can be used instead (e.g., functions, services, etc).

When supplied with this input, the clustering module employs the Latent Dirichlet Allocation (LDA) algorithm, generating a probabilistic description of the contents of the clusters.

The decision of using LDA as the clustering algorithm was based on several factors. The first of these is the fact that LDA corresponds to the current state of the art in terms of clustering algorithms. Additionally, LDA consists in a three-level hierarchical Bayesian model. This shall be discussed in greater detail further on, but suffice it to say that LDA provides semantically richer results than other alternative methods, making it thus more useful for the purpose of the work presented here.

 ISBN: 978-1-61208-165-6

For this work, the contents of the clusters correspond to application methods that are strongly correlated in terms of the domain data manipulated within their runtime scopes. As such, the LDA will seek to populate the clusters in such a way as to maximize the intra-cluster similarity and minimize the inter-cluster similarity. This similarity is, once again, expressed in terms of the domain data used in the methods being clustered. It should be noted that the LDA, being a multivariate clustering model, provides a secondary result. This secondary result consists of a clustering of the application domain data. The cluster identities are the same as the ones for the application methods, with the difference that they are characterized by a stochastic description built-in function of the predominant domain data present in them.

The LDA does not estimate the optimal number of clusters that are to be found in the set of methods composing the target application. The number of clusters is supplied as input to the algorithm. The procedure for identifying the optimal value for the number of clusters shall be discussed at length in section III.B. In the remaining of this section, the theoretic bases of the LDA shall be considered.

Latent Dirichlet Allocation was developed and first presented by Blei et al. [1]. LDA can be generally described as a generative probabilistic model for collections of discrete data. In the probability analysis, a generative model corresponds to a model that can generate randomly observable data, based on some hidden parameters. The generative model specifies a joint probability distribution over observation and label sequences. Keeping this into account, LDA is a three-level hierarchical Bayesian model. The hierarchical Bayesian model corresponds to an elaborate model in modern Bayesian Analysis and allows the modelling of complex situations in a better way than simpler models.

Given data x and parameters ϑ, a simple Bayesian analysis starts with a prior probability (prior) $p(\vartheta)$ and likelihood $p(x|\vartheta)$ to compute a posterior probability:

$$p(\vartheta|x) \propto p(x|\vartheta)\,p(\vartheta) \tag{1}$$

The prior on ϑ depends, in turn, on other parameters φ that are not mentioned in the likelihood. So, the prior $p(\vartheta)$ must be replaced by a prior $p(\vartheta|\varphi)$, and a prior $p(\varphi)$ on the newly introduced parameters φ is required, resulting in a posterior probability:

$$p(\vartheta,\varphi|x) \propto p(x|\vartheta)\,p(\vartheta|\varphi)\,p(\varphi) \tag{2}$$

This procedure may be performed repeatedly, if any of the parameters employed up until now depends on additional parameters, requiring its own priors. The process terminates when priors that do not depend on any further unmentioned parameters have been reached.

The latent multinomial variables shall be referred to as clusters. The latent multinomial variables can be associated without any issue to different concepts.

In the LDA model, each collection item (e.g., method) is modelled as a finite random mixture over an underlying set of clusters. The clusters are modelled as an infinite mixture over a set of underlying cluster probabilities and are characterized by a distribution over domain data types. From the point of view of domain modelling, the cluster probabilities consist in an explicit representation of a method. The approximation inference techniques employed for LDA are based on variational methods and an estimation maximization algorithm for empirical Bayes parameter estimation.

LDA assumes the following generative process for each method m in an application A:

1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$.
3. For each of the N data types m_n:

   (a) Choose a cluster $z_n \sim \text{Multinomial}(\theta)$.

   (b) Choose a data type m_n from $p(m_n|z_n,\beta)$, a multinomial probability conditioned on the cluster z_n.

There are a few simplifying assumptions made in this model, among which is that the dimensionality k of the Dirichlet distribution (and the dimensionality of the cluster variable z) is assumed known and fixed. The Poisson assumption is not crucial for any part of the model and other more appropriate method length distribution may be employed if deemed necessary.

A k-dimensional Dirichlet random variable θ can take values in the $(k-1)$-simplex (a k-vector θ lies in the $(k-1)$-simplex if $\theta_i \geq 0, \sum_{i=1}^{k}\theta_i = 1$), and has the following probability density on this simplex:

$$p(\theta|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{k}\alpha_i\right)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\theta_1^{\alpha_1-1}\ldots\theta_k^{\alpha_k-1} \tag{3}$$

where the parameter α is a k-vector with components $\alpha_i > 0$, and where $\Gamma(x)$ is the Gamma function. The Dirichlet distribution, as a distribution on the simplex, has several useful properties that make it easier to develop algorithms for inferring and estimating parameters for the LDA. The Dirichlet distribution belongs to the exponential family; it has finite sufficient dimensional statistics and is conjugate to the multinomial distribution.

Given the parameters α and β, the joint distribution of a cluster mixture θ, a set of N clusters z, and a set of N data types m is given by:

$$p(\theta,z,m|\alpha,\beta) = p(\theta|\alpha)\prod_{n=1}^{N}p(z_n|\theta)\,p(m_n|z_n,\beta) \tag{4}$$

where $p(z_n|\theta)$ is simply θ_i for the unique i such that $z_n^i = 1$. By integrating over θ and summing over z, the marginal distribution of a method is obtained:

$$p(m|\alpha,\beta)=\int p(\theta|\alpha)\left(\prod_{n=1}^{N}\sum_{z_n}p(z_n|\theta)\,p(m_n|z_n,\beta)\right)d\theta \quad (5)$$

Finally, taking the product of the marginal probabilities of single methods, the probability of the set of application methods is obtained:

$$p(D|\alpha,\beta)=\prod_{d=1}^{M}\int p(\theta_d|\alpha)\left(\prod_{n=1}^{N_d}\sum_{z_{dn}}p(z_{dn}|\theta_d)\,p(m_{dn}|z_{dn},\beta)\right)d\theta_d \quad (6)$$

A graphical representation of the probabilistic model of LDA can be observed in Fig. 1. As can be seen, the LDA representation has three levels. The parameters α and β are application-level parameters and are sampled once in the process of generating an application. The variables θ_d correspond to method-level variables, which are sampled once per method. Lastly, the variables z_{dn} and m_{dn} are variables at the domain data-level. These are sampled once for each domain datum per method.

With regards to the actual implementation employed for the work presented here, it is a Java port of the original LDA implementation presented by Blei et al. [1], with no modifications or extensions performed over the model itself here. The contribution of this work, regarding the use of LDA, resides in the new semantic interpretation given to the model and its associated concepts. This made possible the use of the LDA algorithm, for the first time, to the best of our knowledge, to perform clustering of an object-oriented application's functionality, based on the domain data manipulated within its scope.

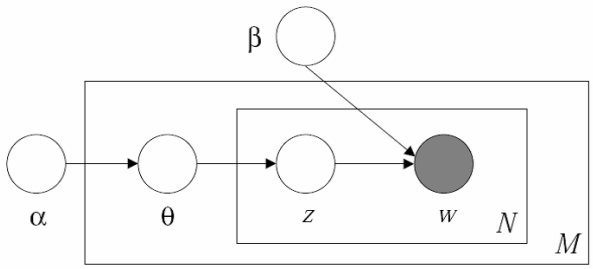

Figure 1. Graphical model representation of LDA

B. *Optimal Clustering Solution*

As indicated in the LDA model description, the algorithm does take some additional input parameters, apart from the occurrence frequencies of the data being modelled. These parameters are the number of clusters among which the data is to be split and the α coefficient value, which is also known as the Dirichlet parameter. The Dirichlet parameter controls the shape of the Dirichlet distribution and, subsequently, the likelihood of a given cluster being selected during the algorithm execution. In practice, high alpha values (close to 1) lead to many clusters being associated to each method, whereas a low value makes it so that few clusters are associated to each method.

As has been previously stated, what we intend with this work is an optimal clustering solution. This makes it necessary to find the additional input parameters' values that lead to the best clustering solutions. To evaluate the effects of the parameter values, we resorted to a well-known and recognized clustering model comparison technique. The technique is known as Silhouette, as reported by Rousseeuw [2]. Intuitively, good clusters have the property that cluster elements are close to each other and far from the elements of other clusters. The Silhouette technique captures this notion and provides an indicator value of how good a particular clustering is.

The Silhouette approach functions as follows. For each data element i, let $a(i)$ be the average dissimilarity between i and all other elements belonging to the same cluster. The approach is independent of the dissimilarity criteria, allowing any appropriate measure to be employed. The value of $a(i)$ can be considered as a measure of how well the element i is matched to the cluster. The smaller the value of $a(i)$, the better the matching is.

Afterwards, for every cluster where i does not belong, an average measure of dissimilarity is calculated, between the data elements of the cluster and i. The minimum of these dissimilarity measures is denoted by $b(i)$. The cluster to which $b(i)$ is associated with is called the "neighbouring cluster" of i, because it is the second best cluster where i could be placed, from among all available clusters. Based on this, $s(i)$ can be defined as:

$$s(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)\}} \quad (7)$$

where $-1\le s(i)\le 1$. When $s(i)$ is close to 1, this means that the datum i is properly clustered. When $s(i)$ is close to -1, the interpretation is that i would have been better placed in its neighbour, instead of the cluster where it is currently placed. If $s(i)$ is close to 0, then it means that the datum is

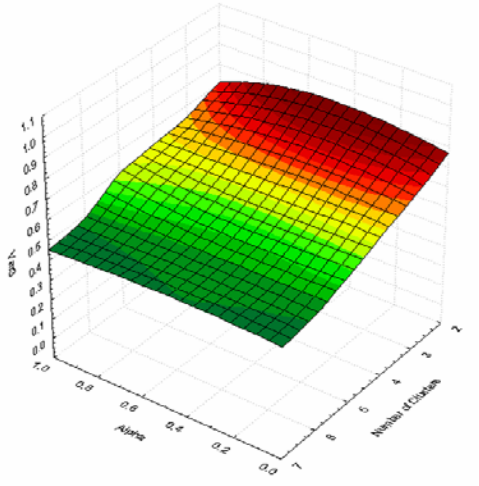

Figure 2. Silhouette coefficient values

placed somewhere "in between" the two clusters.

The average $s(i)$ of all the data placed in a cluster is an indicator of how tightly grouped all cluster data is. The average $s(i)$ for all clusters is a measure of how properly the data has been clustered.

To find the optimal values of the input parameters for the LDA, our system calculates the average $s(i)$ from several executions of the LDA algorithm for every combination of input parameters, within their valid range of values. Once the $s(i)$ coefficients are available for all the evaluated scenarios, the pair of input values which produced the closest to 1 $s(i)$ corresponds to the optimal input scenario that leads to the best possible clustering.

Regarding the similarity measure employed to calculate the $s(i)$, it is based on the gamma values generated by the LDA itself. The gamma values indicate the affinity between the data and the clusters where the data is placed. These affinity coefficients are normalized so that the sum of their values equals 1 for a given method. The dissimilarity measure $a(i)$ of a given application method is set to 1 minus the normalized gamma value for the associated best cluster, whereas the $b(i)$ is set to 1 minus the normalized gamma of the second best cluster.

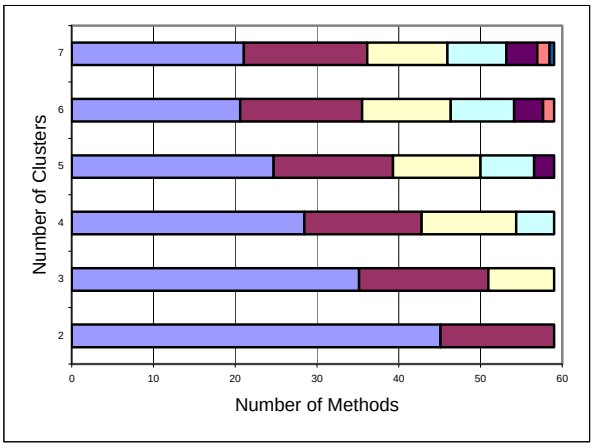

Figure 3. Distribution of average number of methods per cluster

IV. RESULTS AND EVALUATION OF THE SYSTEM

The TPC-W benchmark was selected to serve as a test-case for the demonstration of the new approach presented in this paper. The TPC-W benchmark was introduced by Smith [13]. This benchmark specifies an e-commerce workload that simulates the activities of a retail store website, where emulated users can browse and order products from the website. This particular benchmark was chosen for two main reasons. First of all, it has a reasonably rich application domain model and functionality. Secondly, due to the fact that the benchmark fits well with the type of applications that are most likely to benefit from optimizations that employ the results generated by the system developed with the current work. As previously stated, such optimizations would include dynamic load balancing schemes, conflict-aware approaches for partial or full data replication approaches, among others.

The Silhouette coefficients achieved for the evaluated range of values for the input parameters of the LDA algorithm, when applied to the methods and domain data accessed within them, for the TPC-W benchmark, can be seen in Fig. 2.

The z axis represents Silhouette coefficients, where the valid range of values is [-1,1]. Every point of the surface plotted in Fig. 1 corresponds to an average calculated from 20 independent LDA executions with the same combination of input values. This was done in order to have representative results of the non-deterministic behaviour of the LDA model.

As can be seen from Fig. 2, the input parameter controlling the number of clusters has been varied from 2 to 7, whilst the alpha parameter was varied from 0.01 to 1. Even though there are 59 benchmark methods within which domain data accesses take place, the number of clusters has been varied only up to 7 because, even though the LDA algorithm takes as input the maximum number of clusters among which the methods are to be partitioned, the algorithm decides by itself what is the optimal solution, within the possibilities given by its actual input parameters. Consequently, it is possible for the effective clustering result to consists in a solution where only a portion of the maximum number of clusters have been allocated any elements. This is an increasingly frequent occurrence as the maximum number of clusters increases, and, to a smaller degree, for the lower possible limit of clusters as well.

By analysing the results depicted in Fig. 2, we may conclude that, with regards to the maximum number of clusters, the best Silhouette coefficients (closest to 1) are those associated to 2 and 3 clusters. Regarding the optimal alpha values, even though they do not seem to exert a significant influence over the Silhouette coefficients, the best results are achieved when alpha is in the range of]0.4, 0.5[. The sensitivity analysis study performed by Park in [14] reached the same conclusion, with regards to the effect of the optimized alpha value on the general quality of the clustering results.

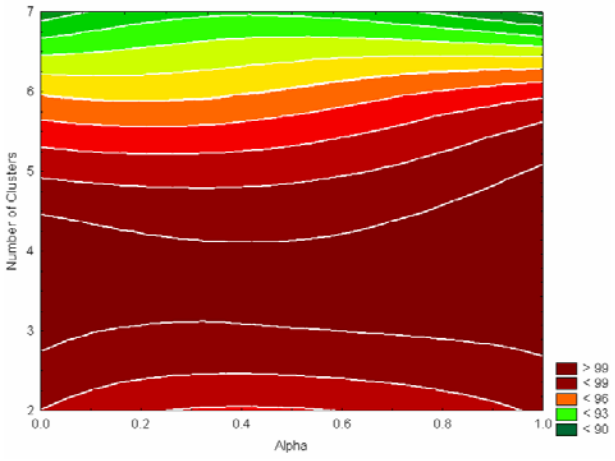

Figure 4. Effective clustering

 ISBN: 978-1-61208-165-6

The most commonly observed number of methods per cluster can be seen in Fig. 3. Each histogram bar was calculated as the average of 20 independent LDA executions, for the same total number of clusters. The x axis represents the number of methods present in a given cluster, while the y axis indicates the total number of clusters among which the data has been partitioned. These results show that, as the total number of clusters increases, the "new" clusters tend to be very small. This is an indicator that a lower total number of clusters is more appropriate, where the number of methods per cluster is more balanced.

A summary of the effective ratio of non-empty clusters can be seen in Fig. 4. The chart represents a 2D projection of the tri-dimensional surface describing the dependency between effective cluster number ratio and the LDA control parameters. The ratio has been calculated as the number of non-empty clusters divided by the maximum number of clusters supplied as input. The highest clustering ratios are achieved for 3 to 4 clusters and alpha values in the range of]0.4, 0.5[.

Combining the results of Silhouette coefficients with the effective cluster number ratio, we can deduce that the input parameter values that most consistently lead to the best clustering results are alpha in the]0.4, 0.5[range and a total of 3 clusters.

V. CONCLUSIONS

This work presented an innovative approach for clustering domain data and application functionality. The algorithm employed is the current state of the art multivariate Latent Dirichlet Allocation. The methodology performs an optimal clustering by fitting input control parameters so as to achieve the best possible clustering result. The optimal solutions are identified through the use of the Silhouette technique. A demonstration of system's capabilities is done based on the TPC-W benchmark. The approach is flexible enough to be applied to any object-oriented application where identifying meaningful clusters of its domain data and functionality is desired.

ACKNOWLEDGMENT

This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds and by the Specific Targeted Research Project (STReP) Cloud-TM, which is co-financed by the European Commission through the contract no. 257784. The first author has been funded by the Portuguese FCT (Fundação para a Ciência e a Tecnologia) under contract SFRH/BD/64379/2009.

REFERENCES

[1] Blei, D. M., Ng, A. Y. and Jordan, M. I., 2003, Latent dirichlet allocation, Journal of Machine Learning Research, 3, pp. 993-1022.

[2] Rousseeuw, P. J., 1987, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of computational and applied mathematics, 20, pp. 53-65.

[3] Challenger, J., Iyengar, A., Witting, K., Ferstat, C. and Reed, P., 2000, A publishing system for efficiently creating dynamic web content, IEEE, Vol. 2, pp. 844-853 vol. 842.

[4] Shen, K., Yang, T., Chu, L., Holliday, J. A. L., Kuschner, D. A. and Zhu, H., 2001, Neptune: Scalable replication management and programming support for cluster-based network services, USENIX Association, pp. 17-29.

[5] Amza, C., Cox, A. L. and Zwaenepoel, W., 2003, Conflict-aware scheduling for dynamic content applications, USENIX Association, pp. 6-20.

[6] Gao, L., Dahlin, M., Nayate, A., Zheng, J. and Iyengar, A., 2003, Application specific data replication for edge services, ACM, pp. 449-460.

[7] Elnikety, S., Dropsho, S. and Zwaenepoel, W., 2007, Tashkent+: Memory-aware load balancing and update filtering in replicated databases, ACM SIGOPS Operating Systems Review, 41, (3), pp. 399-412.

[8] Zuikeviciute, V. and Pedone, F., 2008, Conflict-aware load-balancing techniques for database replication, ACM, pp. 2169-2173.

[9] Amza, C., Cox, A. L. and Zwaenepoel, W., 2005, A comparative evaluation of transparent scaling techniques for dynamic content servers, IEEE, pp. 230-241.

[10] Garbatov, S., Cachopo, J. and Pereira, J., 2009, Data Access Pattern Analysis based on Bayesian Updating, Proceedings of the First Symposium of Informatics (INForum 2009), Lisbon, Paper 23.

[11] Garbatov, S. and Cachopo, J., 2010, Importance Analysis for Predicting Data Access Behaviour in Object-Oriented Applications, Computer Science and Technologies, 1, pp. 37-43.

[12] Garbatov, S. and Cachopo, J., 2010, Predicting Data Access Patterns in Object-Oriented Applications Based on Markov Chains, Proceedings of the Fifth International Conference on Software Engineering Advances (ICSEA 2010), Nice, France, pp. 465-470.

[13] Smith, W. TPC-W: Benchmarking An Ecommerce Solution. Intel Corporation, 2000.

[14] Park, L. and Ramamohanarao, K., 2009, The sensitivity of latent dirichlet allocation for information retrieval, Machine Learning and Knowledge Discovery in Databases, pp. 176-188.

 ISBN: 978-1-61208-165-6