

Fig. 2. Creation of feature vector

clustering, starting from the simple and widespread *k-means* to more complex kernel-based solutions, up to clustering based on spectral analysis [21], [22]. In this proposal of a methodology to automatically cluster similar VMs, we adopt one of the most popular solutions for clustering, that is the *k-means* algorithm [21]. The *k-means* algorithm core is a loop with two steps. The first step starts with a set of random centroids and the set of VMs to cluster, each described as a point in a multi-dimensional feature space (the coordinates of a VM  $n$  in the feature space are represented by its feature vector  $V^n$ ). Each VM is assigned to the cluster of the centroid that is closer to the VM position in the feature space. In the second step, the centroids are updated so that the centroid coordinates are the mean value of the coordinates of the VMs belonging to the corresponding cluster. The loop ends when the assignments of VMs to clusters no longer change.

It is worth to note that the *k-means* algorithm starts with a random set of centroids. To ensure that the *k-means* result is not affected by local minimums, we iterate the *k-means* multiple times, then we compare the ratio between inter-cluster distances (sum of squares of distances between elements belonging to different clusters) and intra-cluster distances (sum of squares of distances between elements of the same cluster). Finally, we select the best solution across multiple *k-means* runs as the solution that maximize inter-cluster distances and minimize intra-cluster distances. The use of the popular *k-means* clustering technique leaves unaddressed the problem of automatically identify the number of clusters to be used to classify the considered VMs. More advanced techniques, such as techniques based on the Spectral Analysis, can be used to this aim [23], but we leave the comparison of different clustering algorithms as a future work.

Once the clustering is complete, we can select some representative VMs for each class to the purpose of simplifying the monitoring task. Clustering algorithms such as *k-means* provide as additional output the coordinates of the centroids for each identified class. In our scenario, the representative VMs can be selected as the VMs closest to the centroids. The choice to consider more than one representative for each class is due to the possibility that a selected class representative changes its behavior with respect to the class it belongs to. When more than one representative is used, quorum-based techniques can be exploited to identify a misbehaving VM

within the list of representatives, as suggested in the case of byzantine fault tolerance [24].

#### IV. CASE STUDY

To evaluate the results of the proposed methodology, we consider a case study based on a dataset coming from an enterprise datacenter supporting one customer Web-based application deployed according to a multi-tier architecture. The data center is composed of 10 nodes on a Blade-based system and exploits virtualization to support the Web application. The nodes host 110 VMs that are divided between Web servers and back-end servers (that are DBMS).

We collect detailed data about the resource usage of every VM for different periods of time, ranging from 5 to 180 days. The samples are collected with a frequency of 5 minutes. For each VM we consider 11 metrics describing the usage of different resources (such as CPU, memory, disk, and network). The complete list of the metrics is provided in Table I along with a short description.

TABLE I  
VIRTUAL MACHINE METRICS

Metric	Description
$X_1$	SysCallRate Rate of system calls [req/sec]
$X_2$	CPU CPU utilization [%]
$X_3$	DiskAvl Available disk space [%]
$X_4$	CacheMiss Cache miss [%]
$X_5$	Memory Physical memory utilization [%]
$X_6$	UserMem User-space memory utilization [%]
$X_7$	SysMem System-space memory utilization [%]
$X_8$	PgOutRate Rate of memory pages swap-out [pages/sec]
$X_9$	InPktRate Rate of network incoming packets [pkts/sec]
$X_{10}$	OutPktRate Rate of network outgoing packets [pkts/sec]
$X_{11}$	ActiveProc Number of active processes

It is worth to note that, to collect data about 11 VM metrics with a frequency of 1 sample every 5 minutes, we need to manage a volume of data in the order of  $3.1 \times 10^3$  samples per day per VM. Considering a data center hosting 110 VMs, the total amount of data is in the order of  $3.5 \times 10^5$  samples per day. After the clustering, we need to continue monitoring every 5 minutes only a few representatives per class, while the remaining VMs can be monitored with a coarse time granularity, for example of 1 sample every few hours. Assuming to select 3 representatives for each of the 2 classes, the amount of data to collect after clustering is reduced to  $1.9 \times 10^4$  samples per day for the class representatives; for the remaining 104 VMs, assuming to collect 1 sample every 6 hours for VM, the data collected are reduced to  $4.5 \times 10^3$  samples per day. From this example we observe that our proposal may reduce the amount of data collected by nearly a factor of 15, from  $3.5 \times 10^5$  to  $2.35 \times 10^4$ .

#### V. EXPERIMENTAL RESULTS

Let us now describe the application of the proposed methodology to the considered case study. The methodology has been

implemented using popular technologies for data management and analysis. Specifically, we use the R language<sup>4</sup> for the statistical analysis functions, Python<sup>5</sup> for the task of reading and writing data, and as a wrapper for the R core. Finally, we use Bourne shell<sup>6</sup> to invoke the main steps of the methodology. These choices ensure that our proposal can be easily deployed directly in currently available cloud infrastructures.

For each considered VM of the data set, we compute the correlation between each pair of measured metrics. As discussed in Section III, the resulting correlation matrix is used to build a feature vector, which describes the VM behavior and is given as input to the subsequent VM clustering step. As the k-means algorithm starts each run with a set of randomly-generated cluster centroids, we run the final clustering  $10^3$  times, then we select the best solution  $C$  as described in Section III. Finally, we compare the output of the clustering step with the ground truth represented by the correct classification of VMs (we consider that Web servers and DBMS servers are divided into two different clusters) to evaluate the performance of the methodology. Specifically, we aim to evaluate how many VMs are correctly identified as Web and DBMS servers. To this purpose, we consider the clustering *purity* [25], that is one of the most popular measures for clustering evaluation. The clustering purity is obtained by comparing the output of the clustering algorithm  $C$  with the vector  $C^*$ , which represents the correct clustering solution. Purity is thus defined as:

$$purity = \frac{|\{c^n : c^n = c^{n*}, \forall n \in [1, N]\}|}{|C|}$$

where  $|\{c^n : c^n = c^{n*}, \forall n \in [1, N]\}|$  is the number of VMs correctly clustered and  $|C| = N$  is the number of VMs.

In our experiments, we evaluate the purity of the clustering as a function of the length of the time series expressing the metric measurements. Furthermore, we provide an insight on the computational costs of the proposed methodology for varying number of VMs and considered metrics.

#### A. Analysis for different time series length

The histogram in Figure 3 presents the clustering purity as a function of the time series length. We show that, given a very long time series, the clustering achieves perfect results, meaning that every Web server and every DBMS is correctly identified. On the other hand, the purity significantly decreases as we reduce the amount of data used to create the correlation matrix. In particular, when the time series is below 20 days, the purity is below 0.7, reaching 0.65 for a time series of only 5 days. From this first analysis, we observe that the straightforward application of the methodology is likely to provide poor results when short time series of measurements are available. However, the capability of the methodology to achieve acceptable performance for short time series is particularly important for our scenario because it allows us to reduce the period of time during which we need a fine-grained monitoring of the VM resources. This motivates a

more in-depth analysis to understand how the methodology performance can be improved for short time series of data.

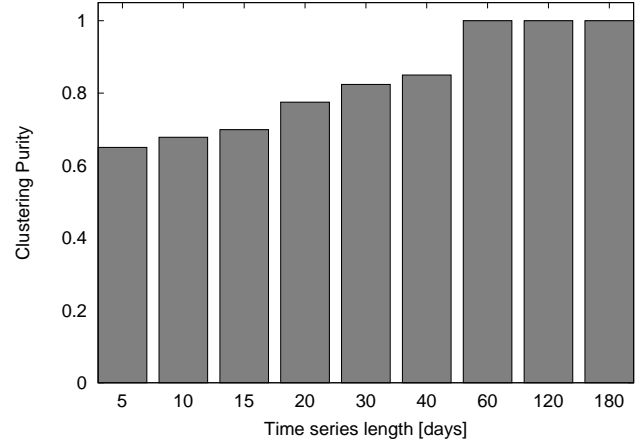


Fig. 3. Clustering purity for different time series length

#### B. Filtering of idle periods

Analyzing the reasons for the poor performance with time series shorter than 20 days, we observe that some VMs present a bimodal behavior, showing periods where the VM is mostly idle mixed with periods where the VM is heavily utilized. When we consider short intervals (e.g., 5 days), we notice that some time series are composed almost exclusively by idle periods. This is the reason for the poor performance of the methodology: during the idle periods, the correlation between the metrics describing the VM behavior is significantly reduced, thus leading to wrong clustering. To avoid this effect, we consider a different approach, where we filter the time series in order to extract a sequence of samples with no idle periods in between. In Figure 4, we compare the results of our filtered data against the time series of the same length used previously.

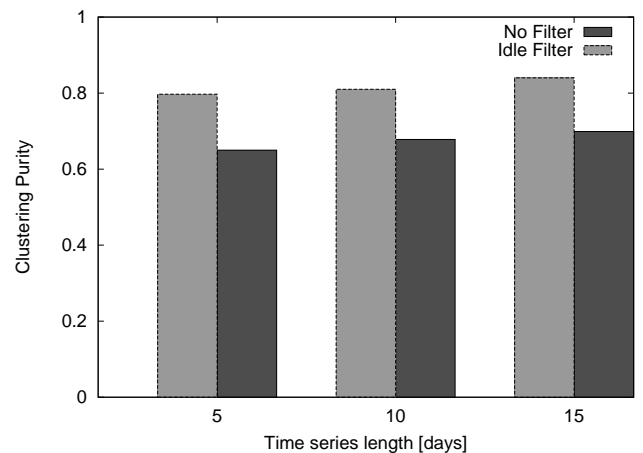


Fig. 4. Impact of idle data filtering on clustering purity

The histograms of Figure 4 clearly show the benefit derived from the filtering of data. The clustering purity is increased

<sup>4</sup>R project home page: <http://www.r-project.org/>

<sup>5</sup>Python home page: <http://www.python.org/>

<sup>6</sup>Bourne shell home page: <http://www.gnu.org/software/bash/>

by 14% for every considered time series length. This results is important because it shows that applying filtering to the resource measurements can increase the purity of the clustering up to values higher than 0.8 even for the shortest considered time series.

### C. Impact of metric reduction

We now aim to evaluate the impact on the methodology performance of a reduction in the number of the metrics considered in the description of VM behavior. The main reason to perform this analysis is the comparison with common state-of-the-art approaches for data center monitoring and management [5], [6], [7], [8], which tend to consider only few VM resources, typically CPU and memory. Specifically, we evaluate if the metrics that are typically used in these approaches are sufficient for our methodology to perform an accurate clustering of VMs. We should also consider that the choice to reduce the number of metrics fed into the clustering algorithm could provide a twofold benefit: first, it reduces the amount of information that is necessary to collect before applying the methodology; second, it reduces the computational cost of the clustering operation as it reduces the size of the feature vectors  $|\mathbf{V}^n|$  given as input to the final clustering step.

In this experiment, we compare the clustering purity when different sets of metrics are used to generate the feature vectors for the clustering step. We compare the proposed methodology, where all the 11 metrics are considered, with two alternatives: the *2 Metrics* case, where only CPU and memory ( $X_2$  and  $X_5$  metrics in Table I, respectively) are considered as in state-of-the-art approaches, and the *4 Metrics* case, where we include also two network-related metrics measuring the rate of input and output packets ( $X_2$ ,  $X_5$ ,  $X_9$ ,  $X_{10}$  in Table I). Figure 5 shows, for different time series lengths, the purity achievable when considering every metric (*All Metrics*), only CPU and memory (*2 Metrics*), and four metrics related to CPU, memory, and network (*4 Metrics*). On the right side of the figure, we report the results for the filtered time series, labeled as X-F, where X represents the time series length expressed in number of days.

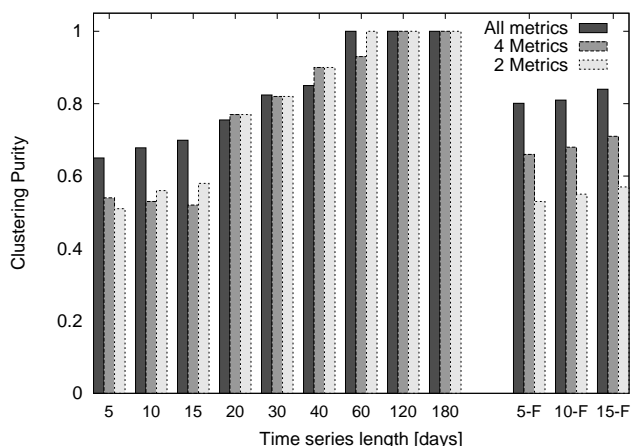


Fig. 5. Clustering purity for different sets of metrics

The histograms in Figure 5 show that reducing the amount of metrics considered by the clustering is not a viable option for VM clustering. Using only four metrics leads to a performance degradation up to 25.6% and 17.6% for unfiltered and filtered time series, respectively. Considering the case where only two metrics are taken into account, the clustering performance is even worse, especially because the penalty in the achieved purity is more evident for the shortest time series, with a decrease up to 21.5% and 33.8% for unfiltered and filtered time series, respectively. Furthermore, when only CPU and memory are considered the application of filtering techniques to short metric time series does not lead to any improvement of the clustering performance with respect to the unfiltered case. This result confirms our claim that for clustering VMs on the basis of similar behaviors we need to consider multiple metrics, differently from common approaches for data center monitoring and management, where only CPU and memory are typically considered.

### D. Evaluation of methodology computational cost

As reducing the number of considered metrics is not a viable option for VMs clustering, we aim to evaluate the computational cost of the proposed methodology as the number of VMs grows in order to investigate the scalability of our approach. To this purpose, we measure the execution times of the two steps of the methodology on an Intel Xeon 2GHz node.

We observe that the time required for the first step of the methodology, which basically computes the correlation matrix describing the VM behavior, always remains in the order of few seconds: it reaches 3 seconds for time series of 180 days and 11 metrics, which is the most expensive case from the computational point of view for this step. As regards the second step of the methodology, we should consider that the computational cost of the clustering phase depends on two elements: the number of considered metrics, which determines the length of the feature vectors given in input to the clustering algorithm, and the number of VMs to cluster. Figure 6 shows the execution times of the clustering step as a function of the number of metrics and of VMs to cluster, considering filtered metric time series of 15 days. In particular, we consider a number of metrics ranging from 2 to 11 and a number of VMs increasing from 10 to 110.

The graph shows a linear growth of the time required for clustering as the number of VMs increases, while the growth of clustering time as a function of the number of metrics is super-linear. This result is consistent with the theoretic computational complexity of the k-means algorithm [26], which is in the order  $\mathcal{O}(ICNV)$ , where  $I$  is the number of iteration of the clustering,  $C$  is the number of clusters,  $N$  is the number of VMs, and  $V$  is the length of the feature vectors used to describe each VM. The length of the feature vector  $V$  has a quadratic dependence from the number of metrics  $M$ , thus explaining the super-linear growth of the clustering time with respect to  $M$ .

We also evaluate the purity achieved for varying metric and VM number. We observe that the purity is mostly unaffected by the number of VMs to cluster, ranging from 0.83 to 0.85

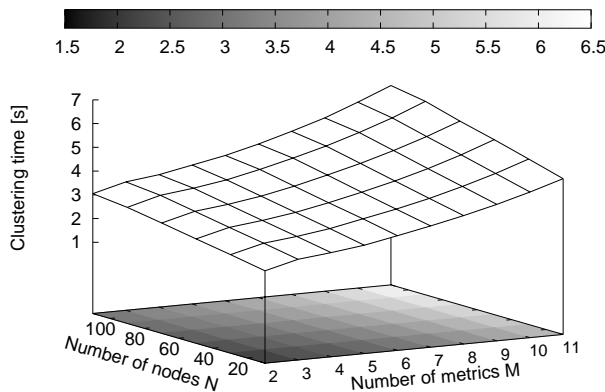


Fig. 6. Clustering time for varying number of metrics and Vms

when all the metrics are considered. On the other hand, purity is highly affected by the the number of considered metrics, as discussed in Section V-C, with values ranging from 0.84 for the whole set of metrics to 0.57 for the case of only two metrics. The proposed analysis of computational costs proves that our methodology for clustering VMs on the basis of their resource usage is scalable and does not pose performance issue for the applicability even in the case of large data centers.

## VI. CONCLUSIONS AND FUTURE WORK

Modern data centers supporting Infrastructure as a Service cloud present major challenges in terms of scalability issues for the monitoring and management of the system resources. In this paper we propose a methodology for automatically clustering VMs into classes sharing similar behavior with the aim to improve the scalability of data center monitoring and management. To capture the VM behavior, the proposed methodology exploits the correlation among the usage of multiple resources, ranging from CPU to storage and network. The application of the proposed methodology to a real data center hosting multi-tier Web applications shows that the purity of VMs clustering ranges between 100% and 80% for every considered scenario and can reduce the amount of data collected by a factor of 15.

This study is just a first step towards the definition of a general methodology for the automated classification of VMs in cloud data centers. As a future work we plan to apply our methodology to a more complex scenario where the data center hosts not only two classes of VMs but several different applications. In this scenario, the methodology will be extended to automatically determine the number of classes to identify during the clustering phase. Furthermore, we intend to investigate the use of alternative techniques to determine the similarity of VMs behavior.

## REFERENCES

- [1] M. Andreolini, M. Colajanni, and S. Tosi, "A software architecture for the analysis of large sets of data streams in cloud infrastructures," in *Proc. of the 11th IEEE International Conference on Computer and Information Technology (IEEE CIT 2011)*, Cyprus, Aug.-Sept. 2011.
- [2] T. Setzer and A. Stage, "Filtering multivariate workload non-conformance in shared IT-infrastructures," in *Proc. of IFIP/IEEE International Symposium on Integrated Network Management (IM'11)*, Dublin, Ireland, may 2011.
- [3] R. Singh, P. J. Shenoy, M. Natu, V. P. Sadaphal, and H. M. Vin, "Predico: A System for What-if Analysis in Complex Data Center Applications," in *Proc. of 12th International Middleware Conference*, Lisbon, Portugal, Dec. 2011.
- [4] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. of the 4th USENIX Conference on Networked systems design and implementation*, ser. NSDI'07, Cambridge, MA, Apr. 2007.
- [5] A. Beloglazov and R. Buyya, "Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers," in *Proc. of (MGC'10)*, Bangalore, India, Dec. 2010.
- [6] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Resource pool management: Reactive versus proactive or let's be friends," *Computer Networks*, vol. 53, no. 17, Dec. 2009.
- [7] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, "Energy-Aware Automatic Resource Allocation in Multitier Virtualized Environments," *IEEE Transactions on Services Computing*, vol. 5, no. 1, pp. 2–19, Jan.-Mar. 2012.
- [8] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW'07, Banff, Alberta, Canada, May 2007.
- [9] R. Lancellotti, M. Andreolini, C. Canali, and M. Colajanni, "Dynamic Request Management Algorithms for Web-Based Services in Cloud Computing," in *Proc. of 35th IEEE Computer Software and Applications Conference (COMPSAC'11)*, Munich, Germany, Jul. 2011, pp. 401–406.
- [10] C. Canali, V. Cardellini, M. Colajanni, R. Lancellotti, and P. S. Yu, "A two-level distributed architecture for Web content adaptation and delivery," in *Proceedings of The IEEE Symposium on Applications and the Internet (SAINT 2005)*, Trento (IT), Jan./Feb. 2005.
- [11] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environment via Lookahead," *Cluster Computing*, vol. 12, no. 1, pp. 1–15, Mar. 2009.
- [12] T. Setzer and A. Stage, "Decision support for virtual machine reassignments in enterprise data centers," in *Proc. of IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS'10)*, Osaka, Japan, Apr. 2010.
- [13] J. Tan, P. Dube, X. Meng, and L. Zhang, "Exploiting Resource Usage Patterns for Better Utilization Prediction," in *Proc. of the 31st International Conference on Distributed Computing Systems Workshops (ICDCSW'11)*, Minneapolis, USA, Jun. 2011.
- [14] C. Canali and R. Lancellotti, "Automated Clustering of VMs for Scalable Cloud Monitoring and Management," in *Proc. of 20th International Conference on Software, Telecommunications and Computer Networks (SOFTCOM'12)*, Split, Croatia, Sept. 2012.
- [15] G. Golovinsky, S. Johnston, and D. Birk, "Syslog Extension for Cloud Using Syslog Structured Data," Internet-Draft, Internet Engineering Task Force, March 2011. [Online]. Available: <http://tools.ietf.org/html/draft-cloud-log-01>
- [16] W.-C. Chung and R.-S. Chang, "A new mechanism for resource monitoring in Grid computing," *Future Generation Computer Systems*, vol. 25, no. 1, pp. 1–7, 2009.
- [17] A. N. Naeem, S. Ramadass, and C. Yong, "Controlling Scale Sensor Networks Data Quality in the Ganga Grid Monitoring Tool," *Communication and Computer*, vol. 7, no. 11, pp. 18–26, Nov. 2010.
- [18] C.-Y. Tu, W.-C. Kuo, W.-H. Teng, Y.-T. Wang, and S. Shiau, "A Power-Aware Cloud Architecture with Smart Metering," in *Proc. of 39th International Conference on Parallel Processing Workshops (ICPPW'10)*, San Diego, CA, Sept. 2010.
- [19] E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite, and W. Zwaenepoel, "Performance Comparison of Middleware Architectures for Generating Dynamic Web Content," in *Proc. of the 4th International Middleware Conference*, Jun. 2003.
- [20] S. Kavalanekar, D. Narayanan, S. Sankar, E. Thereska, K. Vaid, and B. Worthington, "Measuring database performance in online services: A trace-based approach," in *Performance Evaluation and Benchmarking*, R. Nambiar and M. Poess, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 132–145.
- [21] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.
- [22] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recognition*

*Letters*, vol. 41, no. 1, pp. 176–190, Jan. 2008. [Online]. Available: <http://dx.doi.org/10.1016/j.patcog.2007.05.018>

- [23] G. Sanguinetti, J. Laidler, and N. Lawrence, “Automatic determination of the number of clusters using spectral algorithms,” in *Machine Learning for Signal Processing, 2005 IEEE Workshop on*, sept. 2005, pp. 55–60.
- [24] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” in *Proc. of the 3rd Symposium on Operating Systems Design and Implementation (OSDI’99)*, New Orleans, Louisiana, United States, Feb. 1999.
- [25] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, “A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints,” *Journal of Information Retrieval*, vol. 12, no. 4, pp. 461–486, Aug. 2009.
- [26] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.



**Claudia Canali** Claudia Canali is a researcher at the University of Modena and Reggio Emilia since 2008. She received Laurea degree summa cum laude in computer engineering from the same university in 2002, and Ph.D. in Information Technologies from the University of Parma in 2006. During the Ph.D. she spent eight months as visiting researcher at the AT&T Research Labs in Florham Park, New Jersey. Her research interests include cloud computing, social networks, and wireless systems for mobile Web access. On these topics, Claudia Canali published

about forty articles on international journals and conferences. She is member of IEEE Computer Society.

For additional information: <http://weblab.ing.unimo.it/people/canali>



**Riccardo Lancellotti** Riccardo Lancellotti is a researcher at the University of Modena and Reggio Emilia since 2005. He received the Laurea Degree in computer summa cum laude in computer engineering from the University of Modena and Reggio Emilia in 2001 and the Ph.D. in computer engineering from the University of Roma “Tor Vergata” in 2003. In 2003, he spent eight months at the IBM T.J. Watson Research Center as a visiting researcher. His research interests include geographically distributed systems, peer-to-peer systems, social networks and

cloud computing. On these topics he published more than fifty papers on international journals and conferences. He is a member of the IEEE Computer Society and ACM.

For additional information: <http://weblab.ing.unimo.it/people/lancellotti>