An LOF-based Adaptive Anomaly Detection Scheme for Cloud Computing

Tian Huang, Yan Zhu, Qiannan Zhang, Yongxin Zhu, Dongyang Wang
School of Microelectronics,
Shanghai Jiao Tong University,
Shanghai, P. R. China
zhuyongxin@sjtu.edu.cn,

Meikang Qiu
Department of Electrical and Computer Engineering,
University of Kentucky
U.S.A.,
mqiu@engr.uky.edu

Lei Liu
China Unionpay Corporate,
Shanghai, P. R. China,
liulei1@unionpay.com

***Abstract*—One of the most attractive things about cloud computing from the perspective of business people is that it provides an effective means to outsource IT. The behaviors of business applications on cloud are constantly evolving due to technical upgrading, cloud migration as well as social outbreaks. These changes bring the challenge of detecting anomalies during the change of applications on cloud. LOF (Local Outlier Factor) algorithm has already been proven as the most promising outlier detection method for detecting network intrusions. To improve the performance of detection, LOF needs a complete set of normal behaviors of business applications, which is usually not available in cloud computing. We present an adaptive anomaly detection scheme for cloud computing based on LOF. Our scheme learns behaviors of applications both in training and detecting phase. It is adaptive to the change during detecting. The adaptability of our scheme reduces demand of efforts on collecting training data before detecting. It also enables the ability to detect contextual anomalies. Experimental results show that our scheme can effectively detect contextual anomalies with relatively low computational overhead.**

Keywords—anomaly detection; Cloud Computing; LOF; adaptive; contextual anomaly.

I. INTRODUCTION

It is gradually known to the public that cloud computing can be interpreted as "a style of computing where massively scalable IT-enabled capabilities are delivered 'as a service' to external customers using internet technologies" [1]. By combining the functionality of business applications delivered by cloud computing and by leveraging components of Business Process Outsourcing (BPO), executive leadership of a company can realize tremendous profits at a price that is very compelling.

Security and reliability are two important factors that companies mostly concern after shifting business processes onto cloud computing. Although there exist supportive tools for both cloud providers and users to handle anomalies and network intrusions, they still have difficulties keeping BPO agreement promises and application properness due to diversities of cloud services and evolving behaviors of cloud applications [2][3]. These issues impose the demand of adaptability of anomaly detection method.

Local outlier factor (LOF) is an anomaly detection algorithm proposed by Breunig et al. [4]. The key idea of LOF is to compare the local density of a point's neighborhood with that of its neighbors. It is capable of detecting unknown anomalies of various service models [5] of cloud without knowing a complete set of anomaly types in advance. Lazarevic et al. [6] stated that LOF approach is the most promising technique for detecting network intrusions in DARPA'98 dataset [7]. To the best of our knowledge there is no reference support that any method completely beat LOF in the scenario of real dataset in IT domain.

However, LOF still has weakness to be fit for detecting anomalies in service-oriented cloud systems. LOF usually incurs significant efforts in collecting the complete set of normal behaviors before detection starts. DARPA'98 dataset includes seven weeks of networks' performance statistics so that almost all normal behaviors of some designated applications had been collected into a knowledge base. For service-oriented cloud systems with a large variety of applications, the training phase need to be even longer, which precludes effective surveillance.

Another weakness of LOF with static knowledge base is that it cannot detect contextual anomalies [8]. The normal behaviors of a business application on cloud may change due to technical reasons, e.g. cloud migration and software/hardware upgrading, as well as nontechnical aspects, e.g. seasonal events and social headlines. The normal behaviors in the past could be anomalous in the perspective of current context. This situation is termed as contextual anomaly. Detection methods should adapt to the changes of cloud computing and detect contextual anomaly during these unceasing changes.

In this paper, we propose an LOF-based adaptive anomaly detection scheme which can be aware of evolving behaviors of business applications on cloud. We introduce adaptability to LOF by performing knowledge base adaption after each LOF test. The adaptability feature improves our detection scheme in the following ways:

- Reducing efforts in collecting training data before detecting

978-0-7695-4987-3/13 $26.00 © 2013 IEEE
DOI 10.1109/COMPSACW.2013.28

- Adapting to the smooth changes of cloud computing
- Enabling effective detection of contextual anomalies

The paper is organized as follows. Section 2 briefly introduces some state of the art related works. . Section 3 gives an overview of our anomaly detection scheme. Section 4 describes the implementation of adaptability in detail. Section 5 carries out experiments and analysis. Section 6 draws conclusions.

II. RELATED WORK

The authors in [9] proposed a cooperative intrusion detection system for service oriented systems to reduce the impact of DoS attack. The intrusion detection component is used to collect network packets and analyze these packets. If the type of packet belongs to a type of anomalous packet defined in the signature comparison rules, the packets will be identified as an anomaly. The system is restricted to the limited anomalies type known in advance.

The authors in [11] presented an autonomic mechanism for anomaly detection in compute cloud systems. The author use dimensionality reduction in the progress of feature space construction. Such method efficiently reduce the overhead of the detection scheme, but may have fatal problem when business application changes. Principal components may shift to the dimension previously discarded, which means this scheme may lose most important cues of anomalies during the detection phase.

Wang et al. [12] also use LOF to perform workload-aware anomaly detection. Their method is claimed adaptive to different context of workload. However, their work has following shortages. 1. They impose additional overhead of PCA, clustering and recognition besides the calculation of LOF. The performance of their method significantly replies on fine tuning the parameters of these steps, which is not feasible to the diversity of cloud environment. 2. Their work is based on the observation of "session" to judge whether the workload has changed, while many cloud services [13][14], such as IaaS (infrastructure as a service), PaaS (platform as a service) [5] are not session-oriented.

Comparing to the methods above, our scheme is able to detect unknown anomalies without any hypothesis of infrastructure, platform, and software. Our scheme requires less effort of gathering normal behaviors during training phase. It can adapt to changing behaviors, which is very common during evolving of cloud applications. The computational overhead of our scheme is low so that online surveillance on cloud services can be enabled.

III. SCHEME OVERVIEW

Every application has its own pattern of system resource utilization. Our detection scheme makes use of this characteristic to decide whether an application is in proper status. A sample of various system performance statistics can be seen as a multivariate vector, or a multi-dimension point in a feature space. Normally, these points should be close to each other. An anomaly results in unusual system resource utilization, thus deviate the point from normal status. Our detection scheme will observe these deviations and detect these anomalies. Fig. 1 shows the flow of our scheme. It consists of two phases: training and detecting.

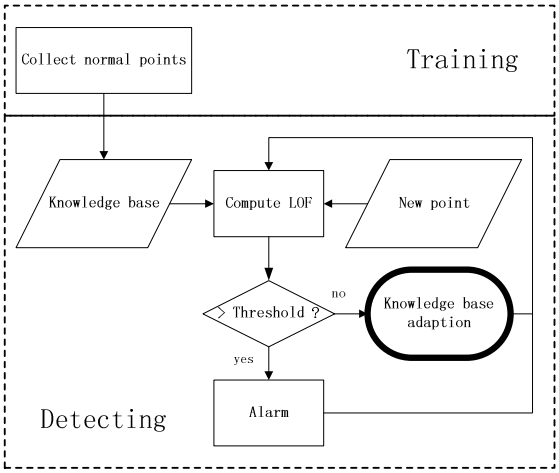

Figure 1 Overview of our adaptive detection scheme

At training phase, we log the system statistics periodically until there are enough logs for anomaly detection. Empirically, a training phase lasting for one day would be fine, and the logging interval should be no less than five seconds to reduce noise. Statistics such as CPU utilization, disk I/O activity, network activity, etc. [15] are collected. Note that time-variant statistics such as storage used and memory free are not suitable for outlier detection method and thus not included here.

A record in statistics log looks like a point whose coordinate are representatives of system statistics. The two phrases "collecting statistics" and "sampling a point" are interchangeable in the rest of this paper. Having enough normal points, we normalize these points and construct a database using R-tree [16][17] data structure for future testing.

At detecting phase, new points are kept being sampled periodically. LOF of each incoming new point will be computed according to the knowledge base we previously constructed. If the LOF value of a new point is smaller than a given threshold we consider it as normal and perform knowledge base adaption. Otherwise we consider it as anomalous and do nothing to the base. Unless specified, we set Threshold to an empirical value of 2.0 in the rest of the paper. During the detection, a specific value of parameter K is used, which denotes the number of neighbors involved in NNS (nearest neighbor search) [18] required by LOF. The detection performance of LOF is not very sensitive to the value of K according to [4]. A typical value of K=30 is used in the rest of the paper.

IV. KNOWLEDGE BASE ADAPTION

Knowledge base should always keep up to date with the changing behaviors. Normal behaviors of a cloud application form one or more clusters in feature space. Since the normal behaviors often change due to migration, seasons, social headlines, technology and other environmental factors, clusters of normal behaviors may smoothly move in feature

space. We apply adaption to knowledge base so that our scheme can capture the changes properly without raising false alarms.

Knowledge base is stored in the R-tree data structure [16], which provides a quick way for LOF algorithm to search nearest neighbor of a given point in database [19]. Table I below shows the pseudo-code of adaption.

A. Knowledge updating

Lines 2~8 are the procedure of the updating phase. Deleting (line 4) and adding (line7) are used to replace the oldest point in the knowledge base with the incoming new point. Deleting is a conditional operation being conducted only when the knowledge base is full. This branch allows knowledge base to keep growing after training phase until a specified capacity upper bound is reached.

With updating, we are able to detect contextual anomaly. In our scheme, Knowledge base acts as a time window sliding along the history of normal behaviors. The oldest normal behavior expires when new one moves in. As a normal cluster moves, a point is possible to be regarded as anomalous even if it belonged to previous normal cluster. This point is now identified as contextual anomaly after new normal behaviors are incorporated into the cluster. Only recent normal behaviors are kept in knowledge base and involved in LOF calculation. As such our scheme has the ability to detect contextual anomalies.

Normalization (line 6) is critical in practice before adding a new point into knowledge base. Breunig et al. [4] did not mention normalization when they proposed the LOF algorithm. We repeat their experiments and find their results only hold when normalization is applied during the establishment of knowledge base. Without normalization, LOF algorithm will be too sensitive to the dimensions with larger variance and could be ineffective to detect anomalies in dimensions with smaller variance.

In our detection scheme, we offset and scale the range of every dimension to 0~1. After this normalization process, the MBR (Minimum bounding rectangle, also refers to bounding box) [20] of the knowledge base is a unit hypercube (unit square in 2-dimension space, unit cube in 3-dimension space).

TABLE I. THE PSEUDO-CODE OF KNOWLEDGE BASE ADAPTION

```
1   /* Knowledge base adaption */
2       /* Update */
3           If knowledge base is full {
4               Delete oldest point in R-tree
5           }
6           Normalize the new point
7           Add new point into R-tree
8       /* End Update */
9       /* Renormalize */
10          If MBR (Minimum Bounding Rectangle) is not
        similar to a hypercube {
11                  Renormalize entire database
12                  Rebuild entire R-tree
13          } Else {
14                  Continue
15          }
16      /* End renormalize */
17  /* End adaption */
```

B. Renormalizing after updating

The updating procedure raises new issues to the normalization of knowledge. It stretches or squeezes the MBR (Minimum bounding rectangle) of the knowledge base when normal behavior keeps changing. This will have negative effect on the performance of LOF algorithm. Fig. 2 shows an example. (a) is a 2D cluster following Poisson distribution. $\lambda_x = \lambda_y$. The point p1 and p2 are at the border of the cluster and have similar LOF value. Suppose the behavior of an application evolves, $\lambda_y >> \lambda_x$. The cluster is stretched in y dimension as shown in (b). p3 and p4 are at the border of the cluster, but p4 has larger LOF value than p3 does. LOF algorithm becomes too sensitive to y coordinate of a point and could be ineffective to detect anomalies in x dimension.

To tackle this issue, MBR checking and renormalization is performed after each updating procedure to ensure that MBR is always similar to hypercube (line 10). The definition of "similar" is as follows: If the longest edge of MBR is 10% longer than the shortest edge, then we say MBR is not similar to a hypercube. During renormalization, every dimension of the database is scaled to 0~1 (line 11). Since renormalization changes relative distance among the points in database, the R-tree spatial index should also be rebuild according to the new database (line 12). After renormalization, the MBR of the knowledge base would become unit hypercube again.

C. Be aware of contextual anomalies

As an additional effect of adaptability, our scheme can detect contextual anomaly. Because we only keep recent normal behavior of application in knowledge base, behavior that never appears or has not appeared for a long time will be

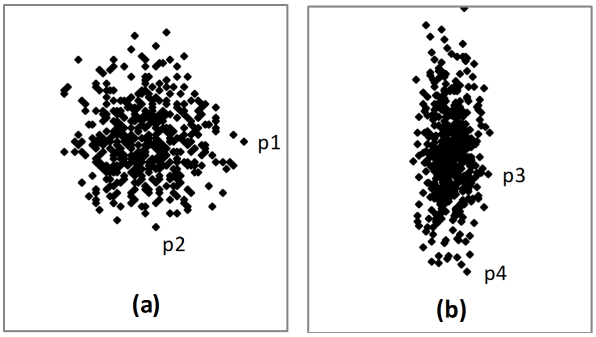

Figure 2 (a) is a 2D cluster following Poisson distribution. $\lambda_x = \lambda_y$. As the behavior of application evolves, λ_y goes beyond λ_x. The cluster is stretched in y dimension as shown in (b). LOF algorithm will be too sensitive to y dimension and could be ineffective to detect anomalies in x dimension.

considered as anomaly. Contextual anomaly can be divided into three categories according to the contextual attributes [8] used during detection: (1) spatial; (2) sequential; and (3) graph. LOF is originally a method capable of detecting contextual anomaly with respect to spatial attributes. With the improvement of adaptability, our scheme has the ability to detect contextual anomaly with spatial and sequential attributes simultaneously.

V. EXPERIMENTS AND ANALYSIS

Our detection scheme is adaptive to the changes of cloud computing. In this section, we first use synthetic datasets to quantitatively identify the adaptability of our scheme. Then we use real data of typical cloud application to verify the ability of detecting contextual anomalies.

A. Identifying adaptability

Identifying adaptability of our scheme is equivalent to the following question: How fast does a cluster of normal behaviors move without triggering alert of our detection scheme. We introduce "stride" as a measurement of how fast a cluster moves. Stride is defined as the Euclidean distance of two adjacent points. We generate synthetic sequence with different strides to identify the adaptability of our scheme. LOF algorithm is capable of detecting high-dimension anomalies. To simplify the experiments, we only use 3-dimension datasets. Relevant variables are defined as follows.

Training sequence *A* for knowledge base:

$$A = \left\{ \left(a_{i,1}, a_{i,2}, a_{i,3} \right) \mid i \in [1,200] \right\} \quad (1)$$

A is a cluster consists of 200 3-dimension points. Each point represents a normal behavior.

For $\forall i \in [1,200], j \in [1,3], a_{i,j}$ are randomly generated following Poisson distribution, $a_{i,j} \sim Pois(100)$

Testing sequence *B* and *C*: *B* and *C* are two sequences of points with different strides.

$$B = (101,101,101), (102,102,102), \ldots \quad (2)$$

$$C = (101,101,101), (103,103,103), \ldots \quad (3)$$

Testing sequence *D*: *D* is also a sequence of points whose stride is a geometric progression with base number α=1.04.

$$D = \left\{ \left(d_i, d_i, d_i \right) \mid \forall i \in [1,\infty), d_i = 100 + \alpha^{i-1} \right\} \quad (4)$$

$$\alpha = 1.04 \quad (5)$$

A is a sequence of 3-dimension points with each dimension following Poisson distribution. The mean and variance of the Poisson distribution is 100. To simplify the experiment, we use *B*, *C* and *D* instead of sequences generated by random variables to represent the worst case of changing. *B*, *C* and *D* are three sequences starting from the center of cluster consisting of *A*. The strides of these testing datasets are different.

In the following experiment we treat *A* as a cluster of normal behaviors and respectively perform LOF detection on *B*, *C* and *D*. During the detection, we set threshold of LOF to 999 so that any point will be added into the knowledge base. As a contrast we also perform detection on *B* without knowledge base adaption. The testing result is shown in Fig. 3.

In Fig. 3, X axis represents the distance of test points to the center of cluster consisting of *A*. Y axis represents corresponding LOF values of the test points. The curve "*B*, noadapt" stands for the detection on *B* without applying knowledge base adaption. This curve keeps growing and does not converge to a certain value, implying that LOF without updating does not adapt to the move of the test point.

In contrast, the curves denoted as *B* and *C* finally converge to an LOF value of 1.3, implying that it is possible for LOF to adapt to the move of test point. We see from Fig. 3 there are peak values of curve denoted as *B* (around 150) and *C* (around 130). This is because as a test point moves out of cluster, the density around it is lower than the density inside the cluster so its LOF value is relatively high. As the test point goes further away from the cluster, only new points get involved in LOF calculation. Since the density of new points are constant and lower than that inside the cluster, LOF value drops from the peak value and stabilizes at 1.3. In fact, we have tested arithmetic progressions with other strides. Experiment shows that the stride only has effect on the peak (larger stride, higher peak value), LOF value will eventually converge to 1.3 regardless of the stride.

The convergence of LOF value at 1.3 in Fig. 3 implies our scheme has the ability to adapt to constant change of behaviors. In practical, the testing sequence mostly appears as a moving cluster rather than a straight line like *B* or *C*. So there is not obvious peak value. The LOF value will grow monotonically to 1.3. For instance, if the workload of a web server grows at a constant rate in its life cycle, it will not be considered as anomalous by our detection scheme with a LOF threshold of 1.5.

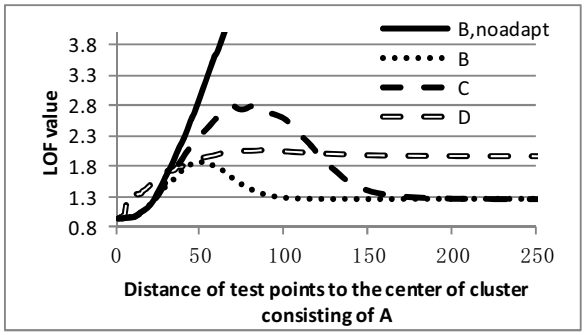

Figure 3 Detecting sequence with different strides

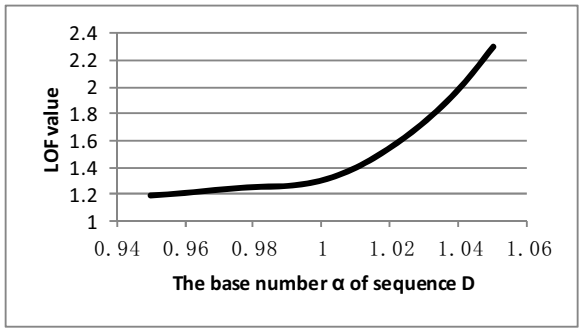

Figure 4 LOF value versus the base number α. With different α, the curve "D" in fig. 3 will converge at different LOF value

The curve denoted as *D* in Fig. 3 also has a peak value around 140. The reason is similar with that of *B* and *C*. But it converges at 1.98 rather than 1.3. This is because the stride of sequence *D* is not constant, the distance of test point in *D* to the center of cluster consisting of *A* grows exponentially. So the ratio of density (definition of LOF) changes according to the base number α of sequence *D*. We also performed detections on sequence *D* with other α and plot a figure to show the relationship between α and LOF value.

From Fig. 4 we see as the base number α increases, the curve denoted as *D* in fig. 3 will converge at higher LOF value. This means our scheme has the ability to adapt to exponential growth of a cloud application. We can use LOF threshold to fine tune the limitation of growing rate. For example, if the workload of a web server grows 105% at every sampling period, it will not be considered as anomalous by our detection scheme with an LOF threshold of 2.5.

B. Detecting contextual anomalies

In this part we establish a web server and setup an access scenario to demonstrate the ability of our scheme to detect contextual anomaly.

The number of HTTP requests per seconds towards the web server follows Poisson distribution. The mean of Poisson distribution keeps dropping in the first half of the experiments and holds in the last part. We inject anomalies by using *malicious port scan* towards the web server in the second half. This kind of anomaly mainly causes performance deviation in terms of network devices. We recorded 8 system performance statistics of the web server and used two halves as training and testing dataset respectively.

Fig. 5 shows the trace of transmitted data speed of a web site we established. Number 1~500 sampling points of the history together with other 7 performance statistics (not shown in fig. 5) are used as knowledge base for LOF algorithm. The rest of sampling points (#501~1000) are used as testing point. From Fig. 5 we can see some sampling points are especially high compared to the average value. In the second half of the history, we inject anomalies into the web site by issuing *malicious port scan* towards the web servers. We apply our scheme onto the real dataset to see if contextual anomalies can be detected.

TABLE II. PERFORMANCE OF CONTEXTUAL ANOMALY DETECTION WITH/WITHOUT ADAPTION

	Without adaption	**With adaption**
Detection rate	10%	70%
False alarm rate	0.2%	0.4%

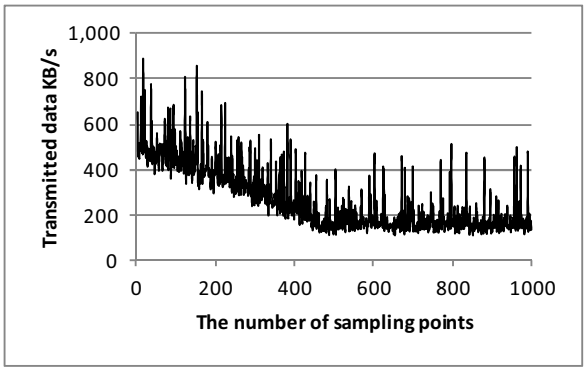

Figure 5 The trace of transmitted data (KB/s) of a web server. The first half of the history is used as training data. Anomalies to be detected are injected in the second half. Other 7 statisticses are not shown in the figure.

We first perform detections on the testing data without knowledge base adaption. Since the knowledge base contain sampling points ranging from 100~800 KB/s, LOF algorithm failed to detect most anomalies we injected in the testing dataset.

Then we enable knowledge base adaption and repeat detection. The experimental result is shown in table II. 70% anomalies are detected while the false alarm penalty is 0.4%. 30% anomalies are not detected because they appear at the beginning of the detecting phase. The training dataset in knowledge base has not been completely replaced by testing sequence. The high-value sampling points in history make anomalies look like normal behaviors. As high-value sampling points being replaced by new testing points, the LOF values of anomalies gradually grow and show up to our detection scheme.

Renormalization is the major overhead of the adaption. During 500 LOF calculations on testing points, 7 times of renormalizations are performed, occupying 9.9% of total time consumption. This overhead of adaption is low enough so our scheme is capable of online anomaly detection for cloud computing.

VI. CONCLUSION

Cloud computing provides an effective means to outsource business processes. The changing cloud environment for business applications imposes new challenges to anomaly detection. In this paper, we proposed an LOF-based adaptive anomaly detection scheme. The adaptability feature of our scheme reduces efforts in collecting training data before detecting and enables the ability to detect contextual anomalies. To evaluate our

scheme, we presented the implementation details of adaptability and quantitatively analyzed its properties. Experiments show that we enabled adaptability without compromising the performance of LOF algorithm. The overhead of adaptability is small enough so that online surveillance can be enabled.

Adaptability is a double-edged sword as it captures the smooth changing of normal behavior, but also treats some anomalies based on gradual change as normal behaviors, e.g., gradual memory leakage or a computer worm that gradually increase the workload will be ignored by the adaptability. Our further efforts will be dedicated to distinguish these anomalies from the changes of normal behaviors.

ACKNOWLEDGMENT

This paper is partially sponsored by the National High-Technology Research and Development Program of China (863 Program) (No. 2009AA012201), National Development and Reform Commission (NDRC) Project(Grant No: C73623989020220110006) and NSF CNS-1249223 for Meikang Qiu.

REFERENCES

[1] Heiser, J., "What you need to know about cloud computing security and compliance," online: www.gartner.com/id=1071415, 2009

[2] Roschke, S., Cheng, F., & Meinel, *C*. (2009, December). Intrusion detection in the cloud. In Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on (pp. 729-734). IEEE.

[3] Almorsy, M., Grundy, J., & Müller, I. (2010). An analysis of the cloud computing security problem. In the proc. of the 2010 Asia Pacific Cloud Workshop, Colocated with APSEC2010, Australia.

[4] Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000, May). LOF: identifying density-based local outliers. In ACM Sigmod Record (Vol. 29, No. 2, pp. 93-104). ACM.

[5] Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications, 34(1), 1-11.

[6] Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., & Srivastava, J. (2003, May). A comparative study of anomaly detection schemes in network intrusion detection. In Proceedings of the third SIAM international conference on data mining (Vol. 3, pp. 25-36). Siam.

[7] Lippmann, R., Cunningham, R. K., Fried, *D*. J., Graf, I., Kendall, K. R., Webster, S. E., & Zissman, M. A. (1999, September). Results of the DARPA 1998 offline intrusion detection evaluation. In Recent Advances in Intrusion Detection (Vol. 99, pp. 829-835).

[8] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 15.

[9] Lo, *C. C*., Huang, *C. C*., & Ku, J. (2010, September). A cooperative intrusion detection system framework for cloud computing networks. In Parallel Processing Workshops (ICPPW), 2010 39th International Conference on (pp. 280-284). IEEE.

[10] Abdullah A. H., bin Abu Bak, K., bin Nagadi M.A., Dahlan *D*., Chimphlee W., "A Novel Method for Unsupervised Anomaly Detection using Unlabelled Data", 08 International Conference on Computational Sciences and Its Applications, 2008, pp: 252-260.

[11] Smith *D*., Qiang Guan, Song Fu, "An Anomaly Detection Framework for Autonomic Management of Compute Cloud Systems", 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, 2010, pp: 376-381.

[12] Wang, T., Zhang, W., Wei, J., & Zhong, H. (2012, July). Workload-Aware Online Anomaly Detection in Enterprise Applications with Local Outlier Factor. In Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual (pp. 25-34). IEEE.

[13] U.S. Department of Commerce. "The NIST Definition of Cloud Computing". National Institute of Science and Technology. Retrieved 24 July 2011.

[14] Voorsluys, W., Broberg, J., & Buyya, R. (2011). Introduction to cloud computing. Cloud Computing: Principles and Paradigms, Wiley Press, New York, 3-41.

[15] sysstat. Available at: http://pagesperso-orange.fr/sebas- tien.godard/

[16] Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching (Vol. 14, No. 2, pp. 47-57). ACM.

[17] JSI (Java Spatial Index) RTree Library. Available at http://jsi.sourceforge.net/

[18] Roussopoulos, N., Kelley, S., & Vincent, F. (1995). Nearest neighbor queries. ACM sigmod record, 24(2), 71-79.

[19] Hwang, S., Kwon, K., Cha, S., & Lee, *B*. (2003). Performance evaluation of main-memory R-tree variants. Advances in Spatial and Temporal Databases, 10-27.

[20] Papadias, D., Sellis, T., Theodoridis, Y., & Egenhofer, M. J. (1995). Topological relations in the world of minimum bounding rectangles: a study with R-trees (Vol. 24, No. 2, pp. 92-103). ACM.