I.J.Modern Education and Computer Science, **2014, 6, 37-45**
Published Online June 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2014.06.05

An Automatic Approach to Detect Software Anomalies in Cloud Computing Using Pragmatic Bayes Approach

Nethaji V
Karpagam University, Computer Science Department, Coimbatore, 642120, India
Email: nethaj.babu@gmail.com

Chandrasekar C
Periyar University, Computer Science Department, Salem, 636011, India
Email: ccsekar@gmail.com

Abstract—Software detection of anomalies is a vital element of operations in data centers and service clouds. Statistical Process Control (SPC) cloud charts sense routine anomalies and their root causes are identified based on the differential profiling strategy. By automating the tasks, most of the manual overhead incurred in detecting the software anomalies and the analysis time are reduced to a larger extent but detailed analysis of profiling data are not performed in most of the cases. On the other hand, the cloud scheduler judges both the requirements of the user and the available infrastructure to equivalent their requirements. OpenStack prototype works on cloud trust management which provides the scheduler but complexity occurs when hosting the cloud system. At the same time, Trusted Computing Base (TCB) of a computing node does not achieve the scalability measure. This unique paradigm brings about many software anomalies, which have not been well studied. This work, a Pragmatic Bayes approach studies the problem of detecting software anomalies and ensures scalability by comparing information at the current time to historical data. In particular, PB approach uses the two component Gaussian mixture to deviations at current time in cloud environment. The introduction of Gaussian mixture in PB approach achieves higher scalability measure which involves supervising massive number of cells and fast enough to be potentially useful in many streaming scenarios. Wherein previous works has been ensured for scheduling often lacks of scalability, this paper shows the superiority of the method using a Bayes per section error rate procedure through simulation, and provides the detailed analysis of profiling data in the marginal distributions using the Amazon EC2 dataset. Extensive performance analysis shows that the PB approach is highly efficient in terms of runtime, scalability, software anomaly detection ratio, CPU utilization, density rate, and computational complexity.

Index Terms—Cloud Environment, Pragmatic Bayes approach, Gaussian mixture, Marginal Distributions, Trusted Computing Base, Profiling Data

I. Introduction

Cloud Computing is being altered and distorted to a new model consisting of services that are commoditized and delivered in a fashion analogous to conventional utilities. In such a model, customers access services based on their necessities without knowing from where the services are hosted or how they are distributed. Cloud computing denotes the infrastructure as a Cloud from which commerce and clients are experienced and proficient to access applications from anywhere in the world using on demand techniques. The Cloud based Computing Service Model is based on three primary factors such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). All IT functions with applications, networking, security, storage space and software are developedfor users to work in a service, based on the client server model.

Dynamic Adaptive Automated Software Engineering (DAASE) as shown in [13], whose goal is to implant optimization deployed software to produce self-optimizing adaptive systems. The distribution needs to be carefully chosen, so that it satisfies the testing objectives expressed in terms of functional or structural properties of the software. The major security, privacy and trust issues in current existing cloud computing environments as described in [13] help users to be familiar with the tangible and intangible threats connected with users. Privacy and trust issues still occur in cloud computing environments by a quantifiable approach and also fails to satisfy the security demands.

Cloud computing is now on the limit of being hold as a serious usage model. However, outsourcing services and workflows into the cloud provides undeniable benefits in terms of flexibility of costs and scalability. A slight advancement in security influences reliability, precision and incident handling. Advanced Cloud Protection System (ACPS) as illustrated in [3] is expected for guaranteeing amplified security to cloud resources. ACPS deployed on several cloud solutions effectively monitor the integrity of guest and infrastructure mechanism while remaining completely transparent to virtual machines and to cloud users.

An innovative remote attestation framework called DRAFT as illustrated in [7] for efficient measuring of target system based on an information flow-based integrity model. The integrity model is not more flexible and systematic way. The high integrity processes of a system are first measured and established, and these processes are then confined from accesses initiated by low integrity processes. An efficient cryptographic protocol as shown in [12] that enforces keystroke integrity by utilizing on-chip Trusted Computing Platform (TCP) prevents the counterfeit of fake key events by malware under reasonable assumptions. A reasonable assumption is difficult in accessing a host's kernel, and the facility to build application-level fine-grained detection solutions.

As described in [5] bread and butter of data forensics and post investigation in cloud computing is characterized by providing the information privacy on sensitive documents. The documents stored in the cloud, unidentified authentication on user access, and provenance tracking on disputed documents demonstrated the provable security schemes. The Secure cloud storage system as depicted in [10] supports privacy-preserving public auditing which performs audits for multiple users concurrently and proficiently. Public auditability for cloud storage is of serious consequence so that users resort to a Third Party Auditor (TPA) as shown in [13] check the integrity of outsourced data. To securely establish an efficient TPA, the auditing process brings in novel vulnerabilities towards user data privacy.

An increase of service activity, service calls at unusual hours, abnormal users, noticeable by a gradually increasing number of document requests and apprehensively active hosts. But change occurs in flow performance of cloud service calls and network hosts of web-services. Cloud computing allows to outsource part of their calculation to server farms, typically owned by a cloud provider. It promises many benefits, such as dropping infrastructure investments, the capability to rapidly acclimatize and compute power according to the demands and the adoption of a pay-as-you-go billing model. Trusted Third Party tasked with assuring specific security characteristics inside a cloud environment as demonstrated in [13] concerned in data communications but does not achieve the quality of services.

Most of the common applications and operating systems are occupied of security flaws at many levels. These vulnerabilities allow an attacker to gain unauthorized privileges, gain unauthorized access to confined data or interfere with the work of others. Detection attempts to compromise the integrity, privacy, or accessibility of computing and communication networks are a tremendously demanding problem. A decentralized mechanism for such self-adaptation as depicted in [6] uses market-based heuristics which does not enrich in CloudSim. Decentralized mechanism also fails in systematically add these modification to scrutinize their effect on the collective infrastructure adaptation.

Nefeli, a virtual infrastructure gateway as demonstrated in [8] provide deployment hints on the probable mapping of VM to physical nodes. Such hints include the collocation and anti-collocation of VM, the existence of possible performance bottlenecks, and the existence of underlying hardware features. Nefeli investigate alternative constraint satisfaction approaches to address scalability issues present in large infrastructures and fails to offer deployment hints that efficiently handle the deployment of virtual infrastructures in the background of real large cloud installations. SBSE for the cloud as formulated in [13] challenges in way of addressing search based software engineering. Cloud providers share analogous goals in reducing resource usage, but they are less focused on uphold their service level agreements on intrusions.

Intrusion detection and avoidance normally refers to a broad range of strategies for defending against malicious attacks. Deploy protection mechanisms in [13] against automated tools exploits failures of systems for support access to system servers, compromising their safety. Software anomaly detection classifies misuse detection techniques which build signatures of all known intrusions. The intrusions use these signatures for detecting challenge and the main drawback of such systems is that they cannot notice new intrusions whose intrusion patterns are unidentified. The need for storing the intrusion signatures for each type of intrusion and the requirement of instant updating of the intrusion signatures impress rigorous performance bottleneck on misuse detection techniques.

In Pragmatic Bayes (PB) approach, two component Gaussian mixture is used to perform deviations. PB monitors the massive number of cells which is useful in streaming scenarios with Bayes per section error rate procedure. A novel feature of PB approach is the capability to restrain deviations that simplify the consequence of sharp changes in the marginal distributions. PB motivated by the need to extract critical application information and business intelligence from the daily logs with higher scalability ratio.

The contribution of Pragmatic Bayes approach is to present a PB framework to detect software anomalies in imbalanced classified data streams with potentially large number of cells. PB framework performs multiple testing using a hierarchical Bayesian model and suppresses redundant alerts caused due to changes in the marginal distributions. Empirically illustrate the superiority of PB approach by comparison to a Statistical Process Control and Trusted Computing Base (TCB) on OpenStack prototype and illustrated with benefits.

The structure of paper is as follows. In Section 1, describe about the definition of software anomaly with existing works. In Section 2, demonstrates the Pragmatic Bayes (PB) approach with the Gaussian mixture to detect the software anomaly by comparing the current history. Section 3 explains about the PB experimental approach with parametric factors. Section 4 analysis the result through table and graph values and section 5 illustrates the related existing works with limitations. The conclusion is described in Section 6.

II. Pragmatic Bayes Theoretical Framework

Pragmatic Bayes aims to detect the software anomalous behavior by comparing data in the current block based on historic data. However, PB approach is interested in detecting software anomalous patterns rather than detecting abnormal software records. The design of PB approach is centredon the concept of monitoring statistical measures which are computed for combinations of definite attributes in the database. Considering definite attributes combinations gives rise to multiple software testing at each time interval. In order to achieve multiple software testing,Bayes per section error rateis evaluated on each cell, where each dimension corresponds to the levels of a categorical variable. The framework of PB is shown in Fig 1.

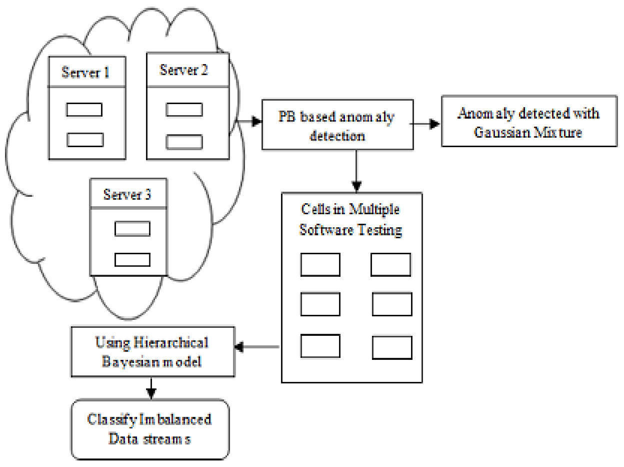
Server 1
Server 2
Server 3
PB based anomaly detection
Anomaly detected with Gaussian Mixture
Cells in Multiple Software Testing
Using Hierarchical Bayesian model
Classify Imbalanced Data streams

Fig1: Architecture Diagram of Pragmatic Bayes approach

As shown in Fig 1, PB classifies the imbalanced data streams. Imbalanced data stream gets computed from call logs that are added to the current database on a daily basis. When monitoring cells for deviations, it is prudent to regulate the sharp changes in the marginal statistics. Software anomalies which are direct consequences of changes in a small number of margins are detected with Gaussian mixture.

For ease of evaluation, the PB approach proceeds with the assumptions that the multidimensional software tests consist of two categorical variables with ‘A’ and ’B’ levels respectively. ‘A’ and ‘B’ levels note the generalization of higher dimensions software anomalies. In practice, PB takes the suffix of the ‘A’ and ‘B’ levels for the first and second categorical variables respectively at time ‘t’. Let X_{ABT} denote the observed value to follow a Gaussian distribution. Often, a certain level of transformation is required for the original data to ensure approximation of true value.

In order to ensure approximate software normality, the counts are observed with the help of a square root transformation. In general, PB square root transformation is denoted as

$$((x+l)^{q}\text{-}1)/\text{q}) \tag{1}$$

In (1), ‘l’ and ‘q’ are chosen to ’stabilize’ the variance and depends on the mean recommended in software anomaly detection. Moreover, ‘q’ is constrained to lie between 0 and 1, and q→0 implies a log transformation while detecting the software anomalies. In fact, the values of these parameters are chosen with a reasonable value with the help of the initial training data. For time interval ‘t’ PB approach detects the software anomalies after regulating the changes with marginal means. The process of PB approach starts with the formal definition of software anomaly followed by the mechanism Pragmatic Bayes approach explained with the help of an algorithm.

A. Formulation of PB Software Anomaly

Consider a 2*2 table, the levels of the row feature being I, J and the levels of the column feature being i, j respectively. PB denotes the 4 cell entries corresponding to (Ii, Ij, Ji, Jj) by a vector of length 4.

$$\begin{matrix} Ii & Ij \\ Ji & Jj \end{matrix} \tag{2}$$

Two values are measured in PB approach to analyse the scalability issue, namely, the expected values and the observed values. The deviations after adjusting the changes in the row and column means are (0, 0, 0, 0), ensuring that it produces no software anomalies in cloud environment. The significant values in the PB approach and the non-adjusted changes is described to be a drop in the first row mean and a rise in the second row mean. Hence, non-adjusted cell modification contains redundant information which results in a situation where the adjusting for margins is desirable.

However, marginal adjustments are not guaranteed to produce a prudent explanation of change in all situations. However, if the marginal drops are attributed to a few specific cells in PB multiple software testing and the goal are to identify the unadjusted software version from cloud zone. In PB approach, software track changes in the margins separately using simple process control techniques and run both adjusted and unadjusted versions with careful interpreting of results.

In fact, the adjusted software version detects changes in interactions among the levels of categorical variables which are the focus of several applications. Also, in higher dimensions PB approach adjust for higher order margins, which is routine in PB approach framework. For instance, adjusting two-way margins in a three dimensional form detects changes in third order interactions also.

Let C_{t-1}denote the current historical information up to time t-1. Deviations at time t are detected by comparing the observed values X_{ABT} with the corresponding posterior analytical Pragmatic Bayes distributions. The PB expected distribution of data at time t is based on current historic data until t-1. Gaussian mixture with Pragmatic Bayes mean and variances is computed as given below:

$$\mu_{ABT} = P(X_{ABT}|C_{t-1}) \tag{3}$$

$$\sigma^2_{ABT} = Var(X_{ABT}|C_{t-1}) \tag{4}$$

μ_{ABT} represent the mean of PB approach and σ^2_{ABT} represent the variance factor of the PB approach. X_{ABT} denotes the observed value which is assumed to follow a Gaussian distribution mixture. The Gaussian denotes current historical information up to time t-1 for detailed analysis of profiling data in cloud environment. A Pragmatic Bayes approach generally used in process control estimate Δ_{ABT} with δ_{ABT} and declares the ab^{th} cell of a software anomaly is,

$$|\delta_{ABT}/\sigma_{ABT}| > L1 \tag{5}$$

The central idea of the Hierarchical Bayesian model is to classify the imbalanced data streams. $H_{mixture}$ assumes Δ_{ABT} which are random samples from Gaussian mixture distribution at time 't', GM_t. The form of GM_t is known but depends on the density rate parameter. A Bayesian hierarchical model is one that is written with software modularity. It is often useful to think of the analysis of data streams using PB inside-unit analysis, and another model for across-unit analysis. The inside-unit model describes the behavior of individual respondents over run time, while the across-unit analysis describes the density and complexity of the units. The two models, inside-unit and across-unitb combine to form the hierarchical model, and Pragmatic Bayes algorithm is used to integrate the pieces mutually and report for all the uncertainty and anomalies.

B. Mechanized Pragmatic Bayes Approach

A Pragmatic Bayes approach makes inference about Δ_{ABT} by using approximation of the hyper parameters in order to perform the deviations. The software inference is obtained by numerically integrating with respect to the posterior of Δ_{ABT} using an adaptive Gaussian mixture quadrature. PB Gaussian posterior distribution of Δ_{ABT} depends directly on δ and indirectly on the other δ's through the posterior of the hyper parameters in order to reduce the complexity issue. Generally, such adaptation of strength makes the posterior means of Δ_{ABT} regress toward each other and automatically builds penalty for conducting multiple software tests.

In fact, replacing σ^2_{ABT} by their PB approach mean gives us a constant S_t which provides a good measure of the global penalty imposed by $H_{mixture}$ at time t. The popular approaches used to capture current history C_t are Gaussian mixture and exponential smoothing. In former, a size Gaussian mixture 'w' is fixed apriority and the distribution at't' is assumed to depend only on data in the distribution stream. Extensive research on computational complexity is maintained to down weight current historic data with the weights dropping exponentially in the past.

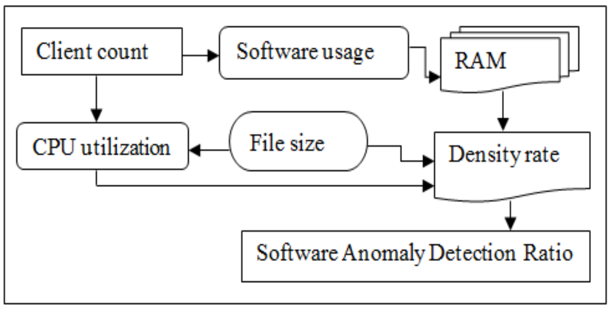

Fig 2: Model for Software Anomaly using PB

Fig 2 describes the detection of software anomalies based on the Pragmatic Bayesian approach. The client count in cloud environment identifies the CPU utilization and software usage factor. The software stored and RAM usage is resolute. The basic model of PB approach is analyzed and finally density rate and software anomaly detection ratio is evaluated. The cell penalty increases with software predictive variance. Also, the general penalty of the procedure at time 't' depends on the hyper parameters which are predictable from data.

C. Pragmatic Bayes Approach Algorithmic Form

Pragmatic approach used to generate data whose statistical properties are close to that of actual data during the training period, followed by the detection of software anomalies and then score the PB approach with existing system. The algorithmic step of PB approach is shown below,

```
Input: Cloud servers with software's
Output: Detect Software anomalies in cloud environment
Start
Step 1: Generate $\mu_i$ for Gaussian distribution 'G'
Step 2: Cell computed from training data and fitted with a log-normal Gaussian distribution
Step 3: Mean and variance of cells in multiple software testing is generated
Step 4: For each i, simulate H ($\mu_{ABT}, \sigma^2_{ABT}$)
Step 5: At time't', PB randomly selects the data streams, and then software anomalies are generated.
Step 6: Detect the software anomalies at time't' using $H_{mixture}$
Step 7: Hierarchical Bayesian Model classifies the imbalanced data streams till the last row and column level of features.
End
```

In principle, PB model provide an estimate of posterior Gaussian predictive means and variances to obtain $\mu_{ABT}, and\ \sigma^2_{ABT}$. However, elaborating on appropriate Gaussian mixture on PB approach has been chosen and the software is trained analytically by the user. Also, to be useful in data streaming scenarios, the PB model is easily adapted to new data. Gaussian mixture effectively captures C_t and detects the software anomalies in cloud environment. Then, the posteriorpredictive mean μ_{ABT} is the sample mean and the posterior predictive variance $\sigma^2_{ABT} is$ replaced by its estimator s^2_{ABT} for effective variance on each X_{ABT}. In order to adjust effects, a separate Gaussian mixture is maintained for each iteration.

III. PBExperimental Approach With Parametric Factors

Pragmatic Bayes (PB) approach in cloud environment uses JAVA estimate to the performance of system with varying parameters. Experiments are evaluated using Pragmatic Bayes (PB) approach, Statistical Process Control (SPC) framework and Trusted Computing Base (TCB) with Open Stack prototype on Amazon EC2 dataset. Amazon Elastic Compute Cloud (Amazon EC2) presents resizable calculating capability in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates the necessity to spend in hardware up front, so that it is easy to enlarge and organize applications faster.

Amazon EC2 provides a broad compilation of instance types optimized on top form diverse use cases. Instance types include untrustworthy mixtures of memory, CPU, storage, and networking capability and present the litheness to decide the suitable mix of resources for the required applications. Every instance type comprises one or more example ranges, permitting to improve the software anomaly detection to the supplies of the target workload. It provides a repository of public data sets that readily integrated into AWS cloud-based applications.

Performance metric for evaluation of PB approach is measured in terms of runtime, scalability, software anomaly detection ratio, CPU utilization, density rate, and computational complexity. Runtime factor is defined as the amount of time consumed to perform the software anomalies detection, measured in terms of seconds (sec). Scalability factor measures the quality of services provided using the Pragmatic Bayes approach, measured in terms of percentage (%).A characteristic of a PB system that describes its capability to cope and perform an increased detection service.

Software anomaly detection ratio in PB is measured as the amount of time consumed to perform the operations on cloud using Gaussian mixture to detect the anomalies whereas the CPU utilization is amount of CPU cycles undergone to perform the anomalies detection operation, measured in terms of Kilobits per second (Kbps). Finally, the density rate is the average speed of detecting the anomalies, measured in terms of percentage (%).Computational complexity factor is a division of the theory of computation in theoretical PB framework that focuses on classifying imbalanced data according to their inherent hierarchical Bayesian form.

IV. Result Analysis Of PB Approach

Pragmatic Bayes (PB) approach in cloud environment is compared against the existing Statistical Process Control (SPC) framework and Trusted Computing Base (TCB) with Open Stack prototype on Amazon EC2 dataset. The experimental value through table and graph describes the software anomalies detection parametric factors on cloud environment.

Table 1.Tabulation of Runtime

No. of users	**Runtime (sec)**		
	SPC Framework	**TCB**	**PB approach**
5		80	75
10	133	121	108
15	218	201	187
20	241	222	210
25	425	389	348
30	535	480	445
35	835	799	728

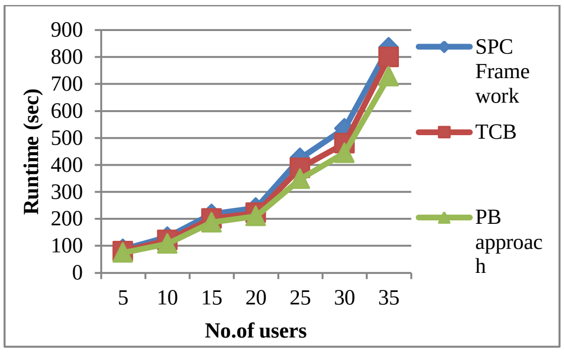

Fig 3: Measure of Runtime

Fig 3 describes the runtime based on the user count. The inside unit model in PB approach describes the behavior of individual respondents over run time and reduces the percentage by 12%- 18% when compared with SPC Framework [1]. The models combine to form the hierarchical model, and Pragmatic Bayes algorithm is used to integrate the pieces mutually and report for all the uncertainty and anomalies within 5 – 10 % limited runtime when compared with TCB [2].

Table 2. Tabulation of Scalability

Technique	**Scalability (%)**
SPC Framework	87
TCB	90
PB Approach	96

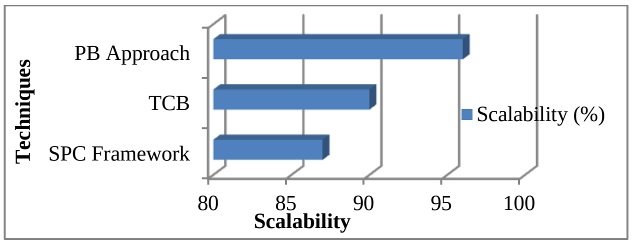

Fig 4. Measure of Scalability

Fig 4 describes the scalability in software detection anomalies. Scalability is measured in SPC Framework, TCB and PB approach. The marginal drops are attributed to a few specific cells in PB multiple software testing and unadjusted software version from cloud zone are identified with higher scalability percentage. The scalability is improved by 6 % with the help of the observed value using the Gaussian distribution mixture when compared with TCB [2] and 9 % when compared with SPC Framework [1] in cloud environment.

Table 3.Tabulation for Software Anomaly Detection Ratio

Run Id	Software Anomaly Detection Ratio (Success %)		
	SPC Framework	TCB	PB approach
20	66	60	70
22	71	69	76
24	76	74	80
26	82	76	86
28	86	80	89
30	91	87	95
32	92	89	98

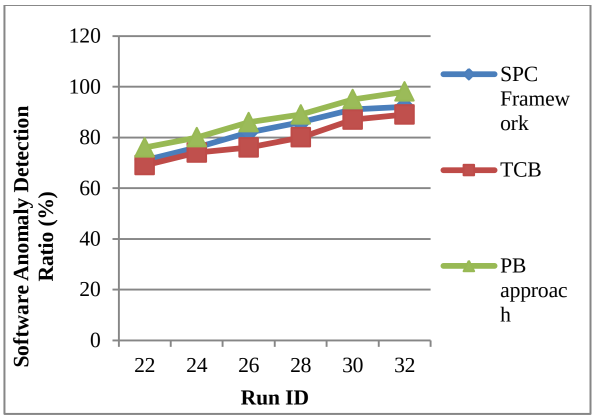

Fig 5:Software Anomaly Detection Ratio Measure

The software anomaly detection ratio is measured based on the Run Id count. PB approach improves the detection ration by 3 – 7 % when compared with SPC Framework [1] because square root transformation is adequate, for proportions arcsine which detect the software anomaly. In PB approach, software track changes in the margins separately using simple process control techniques and run both adjusted and unadjusted versions and results in 8 – 16 %improved detection ratio when compared with TCB [2].

Table 4.Tabulation for CPU utilization

File Size (KB)	CPU utilization (Kbps)		
	SPC Framework	TCB	PB approach
35	6664	5652	4641
62	6737	6731	5722
90	7537	6528	5822
124	7851	6385	5923
189	8167	7154	6149
225	8233	7516	6597
387	9551	8529	7524

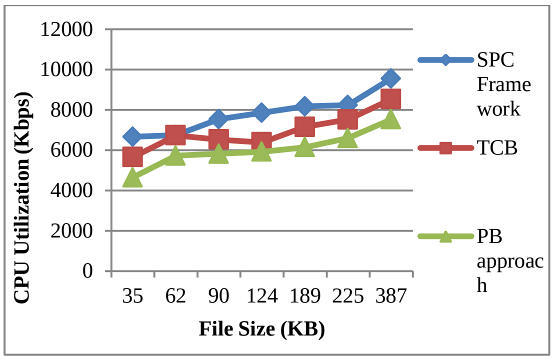

Fig 6: Performance of CPU utilization

Table 4 and Fig 6 illustrate the CPU utilization based on the File size. The size of file is measured in terms of Kilobytes (KB). Pragmatic Bayes approach used in process control estimate Δ_{ABT} with δ_{ABT} and declare the ab^{th} cell of a software anomaly to reduce the CPU utilization. Deviations at time t are detected by comparing the observed values with the corresponding posterior analytical Pragmatic Bayes distributions, so that it results in minimum CPU utilization of 15 – 30 % when compared to SPC Framework [1] and reduces from 7-17% when compared with TCB [2].

Table 5.Tabulation for Density Rate

Performance Counter	Density Rate (%)		
	SPC Framework	TCB	PB approach
10	62	65	69
20	68	69	74
30	75	77	81
40	75	81	85
50	76	80	84
60	77	81	86
70	85	89	93

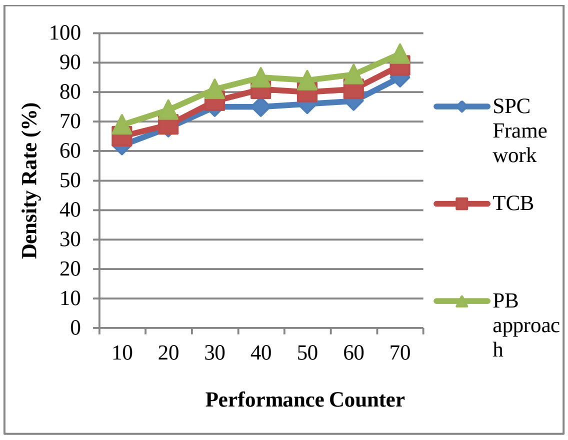

Fig 7: Density Rate Measure

Fig 7 shows the density rate which is measured based on the performance counter. The performance count ranges from 10, 20, 30…up to 70. The density rate is improved by imbalanced classified data stream from call logs that are added to the current database. When monitoring cells for deviations, the PB approach regulates the marginal statistics and increases density rate by 8 – 13 % when compared with SPC Framework [1]. Software anomalies which are direct consequences of changes in a small number of margins are detected with Gaussian mixture, and improve the density rate from 4 – 7 % when compared to the TCB [2].

Table 6.Tabulation of Computational Complexity

Problem Size (Bytes)	**Computational Complexity (bps)**		
	SPC Framework	**TCB**	**PB approach**
250	442	461	386
500	671	676	551
750	954	1004	852
1000	1223	1245	1092
1250	1710	1798	1527
1500	2089	2180	1828
1750	3446	3488	3147

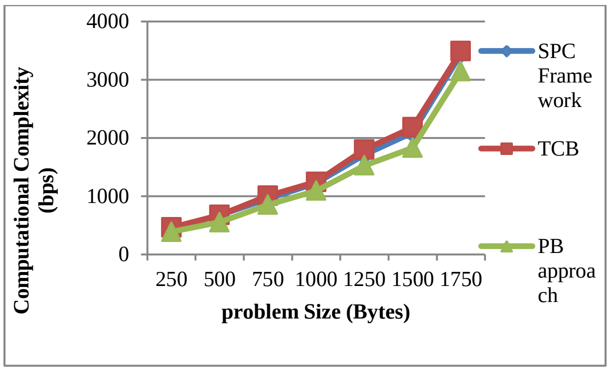

Fig 8:Measure of Computational Complexity

Fig 8 describes the computational complexity based on the problem size where the problem size is measured in terms of Bytes. The PB approach performs multiple testing using a hierarchical Bayesian model and suppresses redundant complexity alerts caused with the changes in the marginal distributions. Pragmatic Bayes algorithm removes the computational complexity by 8 – 17 % when compared with SPC Framework [1] and reduces the complexity from 9 – 18 % when compared with TCB [2].

PB approach simultaneously conducted the Bayes per section error rate which increases the software anomaly detection ratio. However, imbalanced classified data streams used the hierarchical Bayesian framework to reduce the computational complexity and run time factor. PB detects software anomalies in massive imbalanced classified data streams to reduce redundancy by adjusting marginal changes. PB solves the multiple software testing problems using a hierarchical Bayesian model to prove the superiority of $H_{mixture}$ on cloud environment.

V. Related Works

Software anomaly based techniques performed practically to detect intrusions. However, software anomaly detection techniques undergo from higher false alarm rate compared to mishandling intrusion detection techniques. Although many anomaly detection techniques have been proposed to date, no single technique successfully notice all types of intrusions under various scenarios. GenProg framework as illustrated in [13] contains individual variants with crossover design and mutation operators. EC-based program repair approach is expected to be sensitive to these parameter choices. A direct comparison of two source-level representations and an analysis of features supply most extreme success rate were also presented.

Statistical Process Control (SPC) cloud charts framework as described in [1] performs routine anomalies and differential profiling identifies their root causes. By automating the tasks within the framework it remove most of the manual overhead in detecting software anomalies and reduce the analysis time but detailed analysis of profiling data were not performed in most of the cases. Entropy based System with Anomaly detection System providing multilevel Distributed Denial of Service (DDoS) in [4] initially allowed passing through router in network site and detects for legitimate user. It incorporates Knowledge and behavior analysis to identify specific intrusions but fails to detect those attacks that are not included in daily basis updating database task.

Many-Task Computing (MTC) users as illustrated in [9] make use of loosely coupled applications comprising of many tasks. The task achieves their scientific goals with additional analysis of the other services offered by clouds, and in particular storage and network of workloads occur with dissimilar characteristics with the diverse population of cloud users. The outsourcing decryption technique and key private proxy re-encryption as adapted in [14] results in a severe risk on both clients' privacy and intellectual property. The intellectual property of monitoring service providers which could discourage the extensive adoption of mHealth technology but CAM are not design under these stronger models.

Cloud scheduler which solves both the user requirements and infrastructure properties as shown in [2] focuses on guaranteeing the user with their virtual resources. Resources are hosted using physical resources that match their necessities without receiving users involved with understanding the details of the cloud infrastructure. Entrusted computing nodes found by the re-attestation are not immediately detached from the database and would need to re-enrol in the system. Fine-grained access control on sensitive data as stored in [10] assures the confidentiality of the data from the cloud and conserves the privacy of users who are authorized to access the data.

VI. Conclusion

In this paper, a software anomaly detection model called the Pragmatic Bayes approach is presented which aims to detect the software anomalies in massive imbalanced classified data streams. The PB framework then reduces redundancy by adjusting marginal changes and solves the multiple software testing problems using hierarchical Bayesian model within a decision theoretic framework. PB approach proves the superiority of $H_{mixture}$ through simulation using the two component Gaussian mixture for deviations in cloud environment. Furthermore, the PB approach works on combining adjusted and unadjusted $H_{mixture}$ to automatically produce software anomalies detection. In addition, with the introduction of Bayes per section error rate procedure through simulation also suppresses the deviations and provides the detailed analysis of profiling data in the marginal distributions. Experimental result attains the 9.347% minimal runtime, CPU utilization and computational complexity. PB also improves the density rate, scalability, and software anomaly detection ratio on Amazon EC2 dataset.

Acknowledgment

I am greatly indebted to my guide, DR.C.Chandrasekar M.C.A.,Ph.D, Associate Professor in Department of Computer Science, Periyar University, for his unstinted encouragement and guidance for the successful completion of this manuscript.

I have to thanks my parents, wife, son and friends for their encouragement and timely suggestion to successfully prepared this valuable manuscript.

References

[1] Donghun Lee., Sang K. Cha., and Arthur H. Lee., "A performance anomaly detection and analysis framework for DBMS development," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 8, AUGUST 2012.

[2] Imad M. Abbadi., and AnbangRuan., "towards trustworthy resource scheduling in clouds," IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 8, NO. 6, JUNE 2013.

[3] Flavio Lombardi., RobertoDiPietro., "Secure virtualization for cloud computing," Journal of Network and Computer Applications, Elsevier journal., 2010.

[4] A.S.SyedNavaz., V.Sangeetha.,C.Prabhadevi., "Entropy based anomaly detection system to prevent DDoS attacks in cloud," International Journal of Computer Applications (0975 – 8887) Volume 62– No.15, January 2013.

[5] Rongxing Lu., Xiaodong Lin., Xiaohui Liang., and Xuemin (Sherman) Shen., "Secure provenance: The essential of bread and butter of data forensics in cloud computing," ACM journal., 2010.

[6] VivekNallur., Rami Bahsoon., "A decentralized self-adaptation mechanism for service-based applications in the cloud," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 2012.

[7] WenjuanXu, Xinwen Zhang., Hongxin Hu., Gail-JoonAhn., and Jean-Pierre Seifert., "Remote attestation with domain-based integrity model and policy analysis," TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 9, NO. 3, IEEE, 2012.

[8] KonstantinosTsakalozos., MemaRoussopoulos., and Alex Delis., "Hint-based execution of workloads in clouds with nefeli," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 7, JULY 2013.

[9] AlexandruIosup., Simon Ostermann,NezihYigitbasi., RaduProdan., ThomasFahringer., and Dick Epema., "Performance analysis of cloud computing services for many-tasks scientific computing," IEEE TPDS, MANY-TASK COMPUTING, NOVEMBER 2010.

[10] Mohamed Nabeel., Elisa Bertino., "Privacy-preserving fine-grained access control in public clouds," IEEE Computer Society Technical Committee on Data Engineering, 2012.

[11] KuiXu., HuijunXiong, Chehai Wu., Deian Stefan., and Danfeng Yao., "Data-provenance verification for secure hosts," IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 9, NO. 2, MARCH/APRIL 2012.

[12] Qian Wang., Cong Wang., KuiRen., Wenjing Lou., and Jin Li., "Enabling public auditability and data dynamics for storage security in cloud computing," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 5, MAY 2011.

[13] Mark Harmana., KiranLakhotiaa., Jeremy Singerb., David R. Whiteb.,Shin Yooa., "Cloud engineering is search based software engineering too," The Journal of Systems and Software., Elsevier journal., 2013.

[14] DimitriosZissis., DimitriosLekkas., "Addressing cloud computing security issues," Future Generation Computer Systems., Elsevier journal., 2012.

Authors' profiles

V.Nethaji,Tamilnadu, India, 01 November 1981. I received Bachelor of Science (B.Sc) in Computer Technology (Regular) from K.S.Rangasamy College of Technology, Tiruchengode, Tamilnadu, India in 2003. Master of Computer Applications (M.C.A) in Computer Application (Regular) from Periyar University, Salem, Tamilnadu, India in 2006. Master of Philosophy (M.Phil) in Computer Science (Part-Time) from Periyar University, Salem, Tamilnadu, India in 2008.

I have 7 years' experience in Software Testing. Currently I am working in American Megatrends India P Ltd, Chennai as Test Engineer. Previously I have worked in Computer Sciences Corporation as Associate Test Engineer.

Dr.C.Chandrasekar,Tamilnadu, India, 16 May 1972. He received Master of Computer Applications (MCA) from Madurai Kamarajar University, Madurai, Tamilnadu, India and Doctor of Philosophy(PhD) from Periyar University, Salem, Tamilnadu, India.

He has 17 years of experience in both Teaching and research. Currently he is working as Associate Professor in Department of Computer Science from Periyar University, Tamilnadu, India. Previously he worked as Assistant Professor in K.S.Rangasamy College of Technology.