An Efficient Protocol Framework Solution for Resource – Constraint Mobile Devices Allocation in Cloud Computing Environments

Dr.P.Srimachari,
Assistant Professor & Head,
Department of Computer Applications
Erode Arts and Science College
Erode – 638001
srimanchari@gmail.com

Dr.G.Anandharaj,
Associate Professor & Head
Department of Computer Science
Adhiparasakthi College of Arts and Science
Kalavai, Vellore – 632 506
younganand@gmail.com

Abstract - **In cloud computing, data owners host their data on cloud servers, and users (data consumers) can access the data from the cloud servers. An excellent remote data authentication method should be able to collect information for statistical analysis, such as validation results. With the rapid development of network technology and cloud computing, more and more organizations and users outsource their data into the cloud server. In order to protect data privacy, the sensitive data have to be encrypted, which increases the heavy computational overhead and brings great challenges to resource-constraint devices.**

In the proposed scheme, several algorithms are designed to maintain and lookup CBF, while a pruning algorithm is used to delete the repeated items for saving the space. It ensures that the same relevance scores are encrypted into different bits, which can resist the statistical analyses on the cipher text of the relevance scores. Moreover, since the cryptosystem supports the addition of cipher text without the knowledge of the private key, the major computing work in ranking could be moved from user side to the cloud server side. Therefore, the proposed scheme has huge potentials in resource-constraint mobile devices such as 5G mobile terminals. Security analyses prove that the proposed scheme can prevent the information leakage. In the platform, an estimation module has been embedded to predict the future loads of the system, and then, two schedulers are considered to schedule the expected and unpredicted loads, respectively.

The proposed scheduler applies the column generation technique to handle the integer linear/quadratic programming optimization problem. Also, the cut-and solve-based algorithm and the call back method are proposed to reduce the complexity and computation time. Finally, numerical and experimental results are presented to validate our findings. To ensure cloud reliability and availability, a fault tolerance strategy should be developed and implemented. Most of the early fault tolerant strategies focused on using only one method to tolerate faults. This paper presents an adaptive framework to cope with the problem of fault tolerance in cloud computing environments. The framework employs both replication and check-pointing methods in order to obtain a reliable platform for carrying out customer requests. The results of the experiments show that the proposed framework improves the performance of the cloud in terms of throughput, overheads, monetary cost, and availability.

Keywords – Cipher text, Optimization, Resource, Overheads

I.INTRODUCTION

In recent years, with the development of computer science and technology and computer network, Internet of things, cloud computing [1]_[5] which has very high scalability and availability quickly has become the focus of extensive research attention in academia and industry. There is no doubt that cloud computing is the future of computing trend of development. Naturally, many large enterprises became interested in cloud computing, and the storage of data and information in the cloud is of great interest to major companies because it allows data owners to move data from their local computing systems to the cloud. Because of convenience and efficiency, the popularity of cloud storage has increased rapidly.

The Data Owner(s) would worry that the data could be tampered (or deleted) in the cloud. In addition, sometimes cloud service providers may be dishonest. The server may discard some blocks that have not been accessed or rarely accessed to save storage space and claim that all of the files are still intact. Gradually, the security of files has become a big problem in the field of cloud storage. Users are beginning to worry about the security of their files. The companies that provide cloud computing services are aware of this, and they understand that their businesses will collapse without reliable security. The original version consists of only two entities, i.e., the server and the client. First, client calculates the Tag of each block uploads the file and deletes local backup. Second, the client generates a challenge sequence and sends it to the server. In the same way, the cloud server should provide the similar function to the data users, while the data and search privacy should be protected.

More and more mobile applications produce massive sensitive data such as health and location information in cloud computing. Since relevance scores following different distributions, their relevance score transformation function needs to be rebuilt when a new item is inserted. Basically, the DInSAR technique allows generating spatially dense deformation maps with centimeter to millimeter accuracy by exploiting the phase difference between pairs of complex SAR images, usually referred to as Single Look Complex (SLC) images, relevant to acquisitions gathered at different times, but with nearly the same illumination geometry and from sufficiently close flight tracks, whose separation is typically referred to as baseline [1], [2], [17]. More specifically, the DInSAR methodology analyzes the so-called differential; they are generated through the difference between an and its topography-related phase component [17], the latter being calculated by exploiting the sensor orbital information and an external digital elevation model (DEM) of the illuminated scene, properly converted in the SAR coordinates system [17].

II. RELATED WORK

Relatively speaking, solutions have been discussed in previous papers, but [18] was not suitable for big data and batch processing because the computations of those schemes were too large. Some new tag calculation methods [18], [19] have been proposed in order to reduce the amount of computation. In addition to these problems, Zhu et al. [20] discussed the problem of multi-cloud storage in their paper. Brie_y, multi cloud storage is when different parts are stored on different servers. They divided the system into three layers, i.e., the storage layer, service layer, and express layer. All service providers are regarded as an aggregation through the three-layer mapping. Later, many authors solved this problem in a similar way.

Perhaps inspired by the multi-cloud, some people began to pay attention to the problem of multiple owners. Wang et al. [17] used a bilinear aggregate signature scheme [21] to solve this problem. It can aggregate several different signatures into a short signature, thereby reducing both the amount of computation and the amount of communication. In addition to PDP, Juels and Kaliski [22] defined another RDA technique based on sentinel, namely, proof of irretrievability (POR). Its basic idea is to insert a sentinel in the _le, and TPA asks the server to return to these random sentries. Because this scheme uses the error correcting code, the _le can be recovered. But it also has its disadvantages in that each challenge consumes a sentry, and there is no update mechanism, so the verification times are limited.

Also, the usage of space is increased by about 15% because of the sentinel and error-correcting code. Wang et al. [17] used a tree structure to store _les. The tree structure is more conducive to document validation and data update than the array structure. After the tree structure is used to verify the integrity of the data, some people aim to study how to improve the efficiency of the algorithm. Liu et al. [23] proposed the rank- based Merkle Hash Tree(RMHT) to improve the utilization efficiency of MHT [31]. Another way to do this is to improve the balance of the tree. However, such schemes will cause the auditor to incur high computational costs, especially for large-scale _les, because the verification party must calculate the whole tree according to the path. Sookhak et al. [24] proposed a new and efficient scheme to adapt to the era of big data. They created the following new data structure: Divide and Conquer Table(DCT) to perform dynamic update operations efficiently. DCT successfully improved the efficiency of insertion and deletion.

However, if the _le is frequently modified (especially the insertion operation), DCT continues to grow, thereby reducing the efficiency. In general, the data auditing methods assume that the data owner's secret key is secure. But, in fact, this is not necessarily correct. Jia et al. [25] proposed an auditing method to mitigate the damage of the data owners key exposure issue, in which the secret key of the data owner is updated by using the binary tree structure and the pre-order traversal technique will be reduced when multiple virtual machines are used to carry out the same customer's request. Recompilation of a request is avoided by performing multiple replicas of the request on different virtual machines at the same time.

III. PROBLEM ANALYSIS

In addition to the verification of the integrity of the data, most of the current PDP and POR schemes can support third party verification (public verification). In such schemes, there are three participating parties, i.e., the Data Owner, CSP, and TPA. The characteristic of this structure is that CSP is not sensitive to the identity of the authentication party. In fact, many times, that is not what we want to see. We know that both CSP and TPA are only semi-trusted by the Data Owner. Because CSP is semi-trusted, we have the RDA protocol. But the conventional three-entity structure cannot solve the problem of the third partys being semi-trusted. Because, in the old structure, the challenge message is very simple so that everyone can send one to the CSP, and the CSP cannot verify the identity of the challenge sender. Under this mechanism, the adversary can either get the related information about the Data Owners _le(s) or can gather statistical information about the CSPs service status. To this end, traditional PDP models cannot quite meet the security requirements of auditing-as-a service, even though they support public verifiability.

Our design goals can be summarized as the following:

Public auditability to assure the correctness of storage: to allow anyone, not just the clients who originally stored the _le on cloud servers, to have the capability of verifying the correctness of the stored data on demand. Dynamic data operation support: to allow the clients to perform block-level operations on the data _les while maintaining the same level of assurance concerning the correctness of the data The design should be as efficient as possible to ensure the seamless integration of public auditability and dynamic data operation support. Low computation complexity: to allow TPAs to perform auditing with minimum communication and computation overhead. But if the server becomes the bottleneck of the calculation process, it can transfer a part of the calculation load to the verification party

The Proposed Scheme:

After a series of attempts and summaries, we put forward our own solution as follow:

The first step is to generate the key. First, the client generates one pair of keys using some of the parameters. And this pair of keys will be used for the signature and decryption of the _le. Through the encryption/decryption operation and some other measures, the server can tell which users/TPAs are real users/TPAs (i.e., the ones that have the right to receive service). Then the client generates a second pair of keys. This pair of keys is used to generate the _le block identifier. Depending on the identity, the verification party can determine whether the _le has been changed. And the algorithm (Algorithm 1) is described as follows:

```
Algorithm 1 KeyGen
Input: 1k
Output: spk; ssk; v; _
1: compute V fspk; sskg 1k
2: randomly choose V _ 2 Zp
3: compute V v g_
4: record the public key set V fspk; vg
5: record the private key set V fssk; _g
```

After generating the keys, the user must make a digital signature for the _les that are to be uploaded.
(The file name should be unique.)
In addition, the user also must divide the _le into smaller blocks. Then he or she should generate a tag for each file block.

The SigTagGen algorithm (Algorithm 2) is described as follows:
When the tags of each file block have been calculated, the user sends them and the _le blocks to the CSP and then deletes the local backup (block-less verification). Now, the setting process is over. At the beginning of the validation phase, the parties must ensure that they are qualified. In order to obtain the appropriate permissions, the verification party (the TPA or user) should register her or his own identity. Simultaneously, in this program, we use a trusted public platform to process

```
Algorithm 2 SigTagGen
Input: mi; ssk; _
Output: t; _
1: divide the _le into smaller blocks
2: record the total number of blocks, represented by n V
F D (m1;m2; :::;mn)
```

```
3: randomly choose an element V u
4: compute V s D namejjnjjvjjSigssk (namejjnjjvjjujjT ) and
T D timejjversion
5: take s as the signature of this _le
6: for each block i do
7: compute V _i D (H(mi) _ umi )_
8: end for
9: output the _le signature and tag V t and _
```

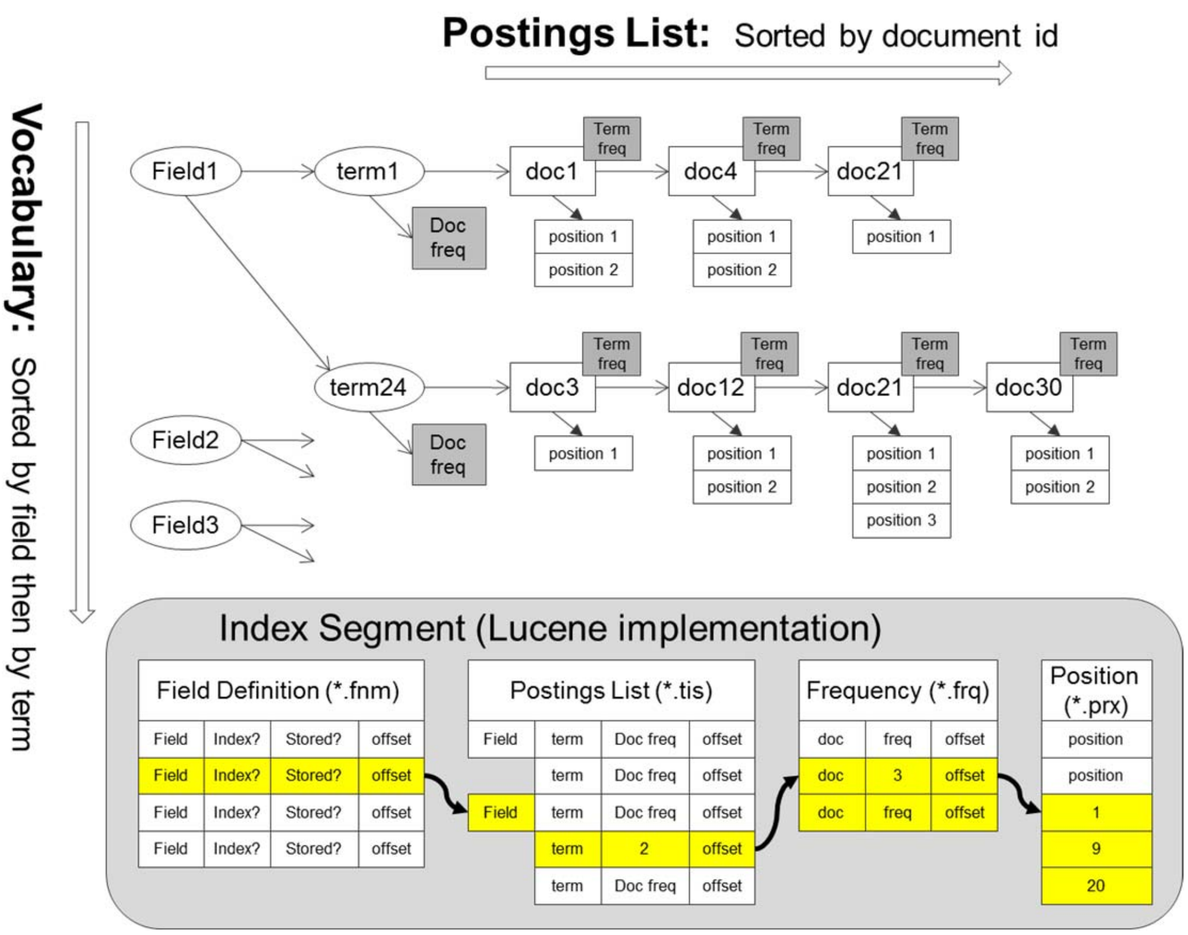
the registration information, and we think this platform is absolutely reliable.
And the Register algorithm (Algorithm 3) is described as follows:

```
Algorithm 3 Register
Input: info
Output: spk
1: generate a pair of keys V fpki; skig
2: compute V sigski (info)
3: send finfo; pki; sigski (info)g to platform
4: waiting for the response of platform V ans
5: if ans DD true then
6: failure to enroll
7: else ans DD false
8: recover spk with ski
9: end if
```

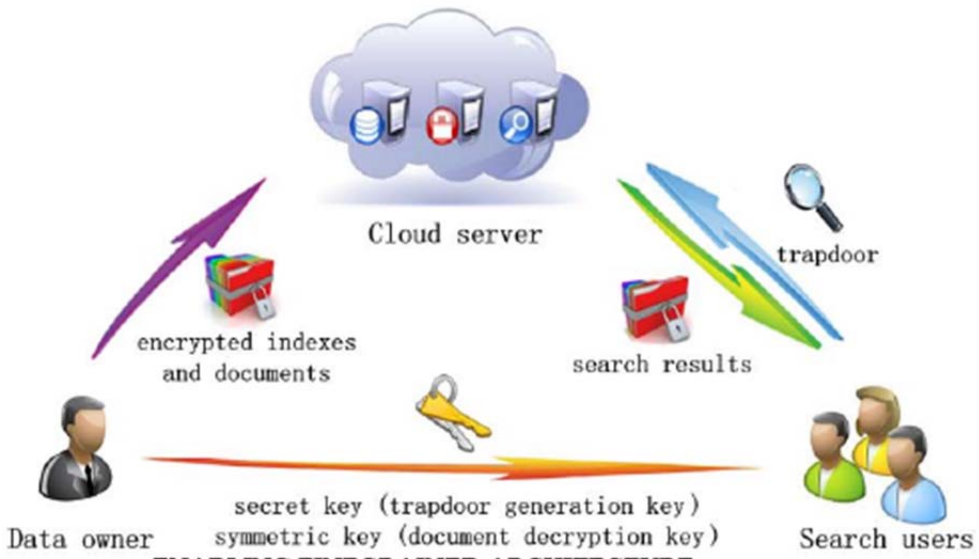
Through the above process, we can know that the public platform verify the identity of the third party and give the *spk* to the third party. And in the next stage, *spk* is a key point in the intercommunication between TPA and CSP.

As mentioned above, the TPA must interact with the CSP to prove its identity before presenting veri_cation requirements. After that, the TPA should generate the validation sequence and send it to the server.

In this program, *n* ensures that the generated random index is within the valid range; *vi* ensures that; *T* ensures that the TPA was successfully registered and specifies an effective time range to avoid replay and other attacks.



The cloud server knows the encrypted documents *C*, index *I'* and trapdoor *T* . The confidentiality of documents, index and trapdoor can be easily ensured by traditional cryptography, various search privacy requirements involved in retrieval procedure are more complex and difficult.


Cloud server
encrypted indexes and documents
trapdoor
search results
Data owner
secret key (trapdoor generation key)
symmetric key (document decryption key)
Search users

ENABLING FINEGRAINED ARCHITECTURE

DESIGN GOALS

In order to enable ranked searchable encryption for effective utilization of the encrypted data, we propose secure index based on CBF for ranked search over encrypted data with following goals:

Ranked Multiple Keywords Search: To design effective multiple keywords search scheme which support result relevance ranking for the encrypted data

Security Guarantee: To prevent the cloud server from directly obtaining or inferring additional information of encrypted documents, encrypted index and search keywords, we set a series of security requirements:

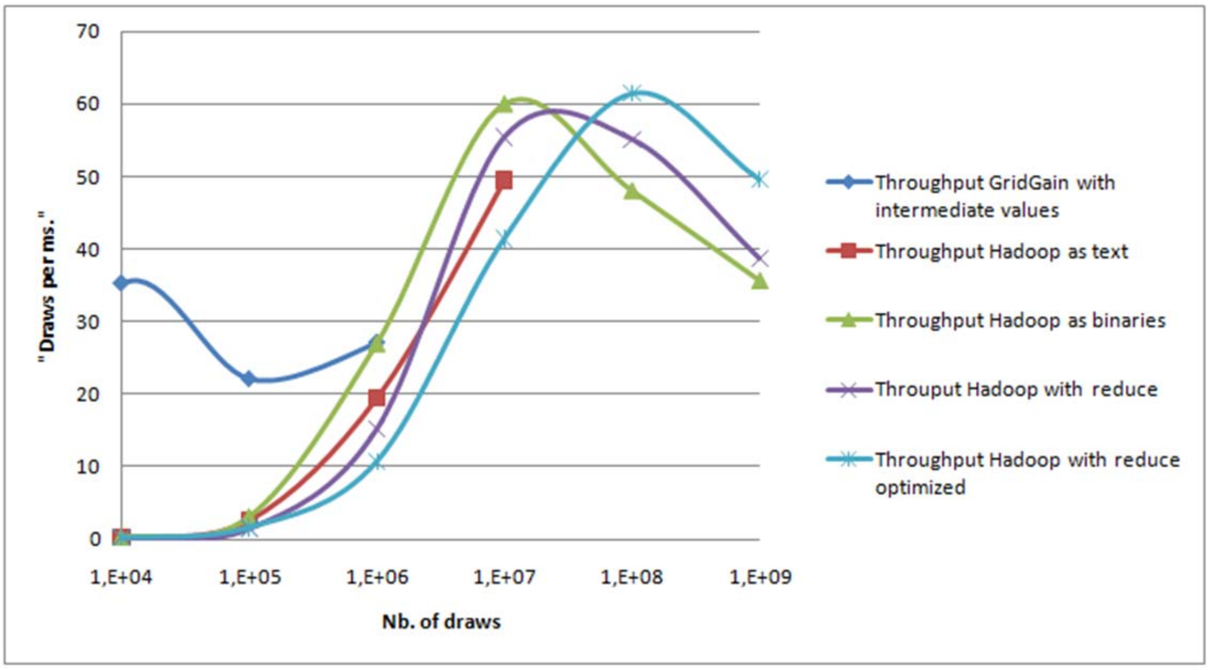
1) *Data confidentiality* presents the confidentiality of documents, index and trapdoor. It can be done by traditional cryptography.

2) *Index privacy* prevents the attacker from inferring the plaintext of the terms stored in index by statistical attacks. The cloud server can obtain the information such as the keyword frequency, relevance scores distribution. This information is helpful in statistical attacks.

3) *Keyword privacy* hides the search keywords by generating trapdoors which leak no information about the keywords. The trapdoor generation function should be a randomized one instead of being deterministic, i.e. even the same search request should be indicated by different trapdoor so as to prevent statistical attacks.

IV. RESULTS

There are many available cloud-simulator environments and Cloud Slim is one of the most of them [28], [29]. Among all classes and packages of the Cloud, there is no one that supports the implementation of fault-tolerant clouds. So, the creation of an extra package is needed in order to support the implementation of fault-tolerant methods in the cloud computing systems. This created package provides services of fault tolerance through allowing some virtual machines of cloud data centers to be faulty. The classes of the package allow the development of fault tolerant based algorithms that can monitor virtual machines in order to detect failures and resolve them. The package can implement both check pointing and replication techniques. The package provides the ability to measure throughput, availability, time overhead and monetary waste overhead. The cloud used in our experiments is generated with 100 heterogeneous virtual machines that are connected with fast Ethernet technology (100Mb/s). The number of data centers used in each experiment ranges from 5 to 10. Each data center contains 4 hosts. The size of each host's memory is 10 GB and the storage is 2TB. The processing capacity of computational units in each host is assumed to be in the range from 1000 to 10000 MIPS. The number of customer requests ranges from 500 up to 2500 requests. Each virtual machine has a memory of 4 GB and one computational unit. The size of data required for each request processing is randomly selected from 10 MB up to 1 GB. The price cost of the cloud computing unit is assumed to be in the range from $0.1 to $10 We evaluate the performance of our proposed framework by comparing it with the check pointing based algorithm proposed in [17], named optimal checkpoint interval (OCI) algorithm, which is based on using variable checkpoint intervals. Different simulation experiments have been conducted with a variable number of customers' requests. The performance metrics used in the comparison include throughput, availability, checkpoints overheads and the amount of monetary waste.



V. CONCLUSION

Failures are unavoidable in cloud computing environments. To treat this issue, an adaptive framework for tolerating faults in cloud computing environments has been proposed in this paper. The framework has one algorithm for selecting virtual machines to carry out customers' requests and another algorithm for selecting the suitable fault tolerance method. Both replication and check pointing methods are included in the framework. The performance of the framework is evaluated with a replication-based algorithm and also with a check pointing-based algorithm in terms of throughput, cloud overheads, monetary cost and availability

VI. REFERENCES

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, ``Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generat. Comput. Syst.*, vol. 25, no.6, pp. 599_616, Jun. 2009

[2] M. Chen, Y. Ma, J. Song, C. -F. Lai, and B. Hu, ``Smart Clothing: Connecting human with clouds and big data for sustainable health monitoring," *Mobile Netw. Appl.*, vol. 21, no. 5, pp. 825_845, Oct. 2016.

[3] A. Alhosban, K. Hashmi, Z. Malik, and B. Medjahed, ``Self-healing framework for cloud-based services," in *Proc. Int. Conf. Comput. Syst. Appl.*, May 2013, pp. 1_7.

[4] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, ``The cost of doing science on the cloud: The montage example," in *Proc. ACM/IEEE Conf. Supercomput.*, Austin, TX, USA, 2008, Art. no. 50.

[5] M. Armbrust *et al.*, ``Above the clouds: A Berkeley view of cloud computing," Univ. California at Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[6] L. Wang, M. Kunze, J. Tao, and G. Laszewski, ``Towards building a cloud for scienti_c applications," *Adv. Eng. Softw.*, vol. 42, no. 9, pp. 714_722, Sep. 2011.

[7] A. Gómeza, L. Carril, R. Valin, J. Mouriñoa, and C. Cotelo, ``Fault-tolerant virtual cluster experiments on federated sites using BonFIRE," *Future Generat. Comput. Syst.*, vol. 34, pp. 17_25, May 2014.

[8] *The Worst Cloud Outages of 2013*. (Apr. 2016). [Online]. Available: http://www.infoworld.com/slideshow/107783/the-worst-cloud-outagesof-2013-so-far-221831

[9] *The Worst Cloud Outages of 2014*. (Apr. 2016). [Online]. Available: http://www.infoworld.com/article/2606209/cloud-computing/162288-The-worst-cloud-outages-of-2014-so-far.html

[10] *Assessing Cloud Infrastructure Provider Perfor-mance in 2015*. (Apr. 2016). [Online]. Available:http://searchcloudcomputing.techtarget.com/feature/Assessing-cloudinfrastructure-provider-performance-in-2015

[11] K. Bilal *et al.*, ``Trends and challenges in cloud data centers," *IEEE Cloud Comput. Mag.*, vol. 1, no. 1, pp. 10_20, 2014.

[12] K. Ganga and S. Karthik, ``A fault tolerent approach in scientific workflow systems based on cloud computing," in *Proc. Int. Conf. Pattern Recognit.,Informat. Mobile Eng. (PRIME)*, Feb. 2013, pp. 378_390.

[13] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, ``Component ranking for fault-tolerant cloud applications," *IEEE Trans. Services Comput.*, vol. 5,no. 4, pp. 540_550, 4th Quart., 2012.

[14] I. Goiri, F. Julià, J. Guitart, and J. Torres, ``Checkpoint-based faulttolerant infrastructure for virtualized service providers," in *Proc. 12th IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Osaka, Japan, Apr. 2010, pp. 455_462.

[15] H. Hui *et al.*, ``An ef_cient checkpointing scheme in cloud computing environment," in *Proc. 2nd Int. Conf. Comput. Appl.*, Harbin, China, 2013, pp. 251_254.

[16] S. Limam and G. Belalem, ``A migration approach for fault tolerance in cloud computing," *Int. J. Grid High Perform. Comput.*, vol. 6, no. 2, pp. 24_37, Apr./Jun. 2014.

[17] J. Cao, M. Simonin, G. Cooperman, and C. Morin, ``Checkpointing as a service in heterogeneous cloud environments," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, Shenzhen, China, May 2015, pp. 61_70.

[18] W. Zhao, P. M. Melliar-Smith, and L. E. Moser, ``Fault tolerance middleware for cloud computing,'' in *Proc. 3rd Int. Conf. Cloud Comput. (CLOUD)*, Miami, FL, USA, Jul. 2010, pp. 67_74.

[19] P. Das and P. M. Khilar, ``VFT: A virtualization and fault tolerance approach for cloud computing,'' in *Proc. IEEE Conf. Inf. Commun. Tech- nol. (ICT)*, Apr. 2013, pp. 473_478.

[20] C. Assi, S. Chadi, S. Sebbah, and K. Shaban, ``Towards scalable traffic management in cloud data centers,'' *IEEE Trans. Commun.*, vol. 62, no. 3, pp. 1033_1045, Mar. 2014.

[21] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, ``Workload prediction using ARIMA model and its impact on cloud applications' QoS,'' *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449_458, Oct./Dec. 2015.

[22] S. Vakilinia, ``Performance modeling and optimization of resource allocation in cloud computing systems,'' Ph.D. dissertation, Dept. Elect. Comput. Eng., Concordia Univ., Montréal, QC, USA, 2015.

[23] S. Vakilinia, M. M. Ali, and D. Qiu, ``Modeling of the resource allocation in cloud computing centers,'' *Comput. Netw.*, vol. 91, pp. 453_470, Nov. 2015.

[24] D. K. Barry, *Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's Guide*. San Mateo, CA, USA: Morgan Kaufmann, 2003.

[25] S. Vakilinia, X. Zhu, and D. Qiu, ``Analysis and optimization of bigdata stream processing,'' in *Proc. IEEE Globecom*,Washington DC, USA, 2016, pp. 12_18.

[26] A. Thakar and A. Szalay, ``Migrating a (large) science database to the cloud,'' in *Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput.*, 2010, pp. 430_434.

[27] M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*. Cambridge, MA, USA: Springer, Feb. 2011.

[28] S. Vakilinia, D. Qiu, and M. M. Ali, ``Optimal multi-dimensional dynamic resource allocation in mobile cloud computing,'' *EURASIP J. Wireless Commun. Netw.*, vol. 2014, no. 1, pp. 1_14, Dec. 2014.

[29] S. J. Julier and J. K. Uhlmann, ``New extension of the Kalman _lter to nonlinear systems,'' *Proc. SPIE*, vol. 3068, pp. 182_193, Jul. 1997.

[30] W. Wei, W. Shyong, *Time Series Analysis*. Reading, MA, USA: Addison-Wesley, 1994.

[31] S. Karlin and M.-Y. Leung, ``Some limit theorems on distributional patterns of balls in urns,'' *Ann. Appl. Probab.*, vol. 1, no. 4, pp. 513_538, Nov. 1991.

[32] F. Dabek, N. Zeldovich, F. Kaashoek, D. Mazières, and R. Morris, ``Eventdriven programming for robust software,'' in *Proc. 10th ACM Workshop Eur. (SIGOPS)*, 2002, pp. 186_189.

[33] D.Kusic, J. O.Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, ``Power and performance management of virtualized computing environments via lookahead control,'' *Cluster Comput.*, vol. 12, no. 1, pp. 1_15, Mar. 2009.

[34] D. Bruneo, ``A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 560_569, Mar. 2014.

[35] H. Khazaei, J. Misic, and V. B. Misic, ``Performance of cloud centers with high degree of virtualization under batch task arrivals,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 12, pp. 2429_2438, Dec. 2013.

[36] J. Zhang, H. Huang, and X. Wang, ``Resource provision algorithms in cloud computing: A survey,'' *J. Netw. Comput. Appl.*, vol. 64, pp. 23_42,