ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 9(11), DOI: 10.17485/ijst/2016/v9i11/89268, March 2016*

A Comprehensive Survey on Taxonomy and Challenges of Distributed File Systems

D. Sathian[1*], R. Ilamathi[2], R. Praveen Kumar[3], J. Amudhavel[2] and P. Dhavachelvan[4]

[*1]Department of CSE, Pondicherry University, Kalapet, Puducherry – 605014, Pondicherry, India; dsathian@gmail.com
[2]Department of CSE, SMVEC, Madagadipet, Puducherry – 605107, Pondicherry, India; ilam.mathi21@gmail.com, info.amudhavel@gmail.com, rathnapriyadevi89@gmail.com
[3]Department of MCA, SMVEC, Madagadipet, Puducherry – 605107, Pondicherry, India; praveenkoumar@gmail.com
[4]Pondicherry University, Pondicherry – 605014, Tamil Nadu, India; dhavachelvan@gmail.com

Abstract

Background/Objectives: To analyze, find and interpret the current challenges in Distributed File Systems for research. **Method/Statistical Analysis:** With the immense growth of network applications a large amount of data existence is managed and stored. To process a huge amount data that is to be stored, a backend framework is required and for that DFS provides a central store for those data. As the amount of data being increased, the issue of proving efficient and easy to use solution arises and this issue is overcome using distributed file systems. **Findings:** In this research, an overview about various evolution of file system and how distributed file system is being used effectively in wide variety of network applications. DFS aims at sharing data and storing resources which is physically distributed on computers with a common file system. The main issues concerning the design of DFS such as fault tolerance, architecture, synchronization, consistency and replication and few others are also discussed in detail. It alsogives a comprehensive taxonomy of DFS in detail. **Applications/Improvements:** The encouraging results observed from this work serve as the motivation to apply future implementation of DFS within the context of current developments in distributed computing.

Keywords: Challenges, Distributed File System, Fault Tolerance, Scalability, Transparency

1. Introduction

Long term storage is the purpose of file system which is the subsystem of operating system. It is used to manage how a file is stored and retrieved out. As the need of storage and processing of data increased, the various evolution of file system emerged. Various progression of file system is discussed here. Distributed file systemis a client -server based application that allows processing of data stored on the server which makes client to access them[1]. The demand of storing large amount of data has developed over recent years. These data should be stored for future change or sharing among users. Local file system could not handle huge data for providing services. DFS allows sharing huge data over a period of time using multiprocessors in a secured and a reliable way. DFS has a distributed implementation but appears to provide a centralized file system view. In DFS files are accessed by a number of remote clients that are stored in one or more central server stores. Replication can be used for achieving better system performance and reliability. Multi-computer file sharing system in DFS does not need IPC or RPC[2]. File sharing between user will be in a hierarchical and in a unified view. DFS contains software residing on both network servers and clients which links shared file located on different file server transparently into a single namespace in improving load sharing and data availability. Initially, file appears to be a normal file on a client machine that retrieves file from the server and the client machine can work on the same file as if it is stored locally on to its workstations. After finishing the work with the file then it is returned back to its server in an altered form and those file can be used for future references. In consistency maintenance each of update that made to file are being consistently maintained at each replica servers where the file is being located. Various

*Author for correspondence

requirements of DFS compared to a local file system are First, Fault tolerance is well implemented, ability of how the data are being recovered during or after failure. Second, it can store and handle huge amount of data[3]. Here, file servers are divided into file and blocks (i.e.) to reduce size of data handled by one operations (GB's to MB's) and it requires an additional mapping procedure. Finally, files are in write-once and read-many format and in handling metadata DFS assigns a certain node through which data can be retrieved faster.

1.1 Structure of Distributed File System

The schema of distributed file system structure contains the following:

1.1.1 Service

It uses software that runs on one or more machines in offering a typical function to remote clients in prior.

1.1.2 Server

It provides service by running the software on a particularly single machine.

1.1.3 Client

It is used to invoke the services provided by its server by a client interface by performing certain set of operations.

The rest of the paper is organized as follows: section 2 provides a detailed overview of issues in the distributed file system. Section 3 presents taxonomy of DFS is reviewed in detail respectively. Section 4 is concludes the discussion of the paper.

2. Motivation

DFS plays a vital role in managing huge cluster of data store on to the network. In this paper we would try to discuss various evolution of file system and using that in we try to present various issues and taxonomy of DFS alone in detail.

3. Evolution of File Systems

We could find a various kinds or types of file system that are being listed in long range, in market we could now find enormous types of file system being evolved with that emergence, large amount of unstructured data, which are to be stored up and managed to match up real time problem solving existence[4-6]. The major category or classification file systems are listed below.

3.1 Local File System

Applications are present within the server. Data contents are available only at their individual file system and no sharing of data is made possible. One way for sharing the data is using scale up. This scaling can be done in two ways they are vertical scaling and horizontal scaling.

3.2 Shared File System

Different application or multiple instances of the same application will be running on different servers. Multiple servers are connected simultaneously through the physical storage devices which are called shared storage devices[7,8]. Therequest made by the client I/O is transactional in this file system. The transactions are made through three steps. In the first step the request will be made by client from metadata server. In the second step client will receive the response from the metadata server. In the third step the data from the file are read or write from the storage device by the client.

3.2.1 SAN File System

The san file system is a master slave architecture. The master role will be played by the metadata servers. The dedicated low latency system area network is used to connect the clients and MDS[9]. In the SAN file system the clients are widely dispersed among the same operating system.

3.2.2 Cluster File System

Cluster file system is another variation in the shared file system. In the cluster file system the dedicated machine is not the Meta data server. The metadata services are distributed among all clients within the ecosystem. So the cluster file system is distributed database which requires cluster technology to handle the cache coherency and locking.

3.2.3 Network File System

The network file system is similar to the SAN file system, but in SAN file system a block based protocol like ATA or SCSI is used by the client in order to access the storage devices[10].

3.2.4 Distributed File System

From the file server the clients read and write the files which appear like a single file server, but several file servers are used for distributing the files and directories in DFS[11].

3.2.5 Distributed Parallel File System

In distributed parallel file system the file servers are used to distribute the files in addition to that the files are segmented into small parts and are distributed over the file server. By which the granularity is smaller. A single or global namespace is provided to all file servers[12].

4. Issues Considered in Design of Distributed File System

During designing a distributed file system various issues are being taken into considerations. The issues are:

4.1 Transparency

Transparencyis hiding processes and resource distribution held physically across the network from the user. Transparency does not provide the user of the system what is happening behind the system (i.e.) what processing and interaction is held between the client machine and server machines[13]. The various transparencies are described in the Table 1 and Table 2:

4.2 Flexibility

Flexibility can be achieved by using a monolithic kernel or microkernel on each machine. To mange activities like process management, resource management, memory management kernels are used. Monolithic kernel provides functionality of kernel by not considering whether all machine use it or not, the approach used is (kernel does it all). Micro-kernels provides accessibility to other services as needed using minimalist modular approach[14-16].

Table 1. Different forms of transparencies

Transparency	Description
Access	Resource access and data representation differences are hidden
Location	Resource location is hidden
Migration	Moving of resources from one location to another is hidden
Relocation	Location of resource moved out during use is hidden
Replication	Resource is Replicated are hided
Concurrency	Resource are hidden that is may be shared by multiple competitive users
Failure	Failure and recover of resource is hidden

Table 2. Abbreviations of taxonomy

DFS	Distributed file system	REPL	Replication
PRO	Process	RAID	Redundant arrays of inexpensive disk
SEC	Security	CKS	Checksum
FT	Fault tolerance	CSC	Client-side caching
CR	Consistency and replication	SSR	Server side replication
SYNC	Synchronization	FLS	File-locking system
NAM	Naming	HYA	Hybrid
COMU	Communication	ATT	Atomic transactions
ARC	Architecture	CMS	Central metadata server
AUTHE	Authentication	MDS	Metadata distribution in all nodes
AUTHO	Authorization	N-I	Network independence
PRIV	Privacy	UDP	User datagram protocol
TCP	Transmission control protocol	ASY	Asymmetric
SY	Symmetric	CB	Cluster based
PL	Parallel	CS	Client server
WO	Write-once	HDFS	Hadoop distributed file system
RM	Read-many	KFS	Kosmos file system
PSC	Perform all operations synchronously	PVFS2	Parallel virtual file system
GFS	Google file system	RGFS	Red hat global file system
NFS	Network file system	EXCEP	Exception

4.3 Reliability

Reliabilityis availability of data when and where it is required on time without any variation and alternation. Distributed system allows multi-processes which are preferred by users as it can protect against single processors system crashes. In case of failure replicated copies can be used .if we use replicated copy, the copy present should be consistent with its content. Thus even on a failure, a backup of contents are available[17].

4.4 Performance

Performance metrics can be measured using response time, throughput, system utilization and amount of network capacity used. Response time is time of system or a functional unit to respond to the given input or request. Throughput is successful message delivery with respect to time measured out in (bps) bits per second. System utilization reports server OS configuration and its associated utilization information. Network capacity is maximum capacity of network path to convey data from one network location to another[18]. Applications that run should be presented as if it running on a single processor. In LAN message transmission takes over milliseconds. Performance can be increased by reducing the number of message transmitted[19].

4.5 Scalability

Scalabilityin which resources get added or removed in a network at any time. More clients are connected with the system day to day. An efficient system is needed in handling those users or client. There can be many CPU's getting added to the distributed system. While designing, DFS can be made in two ways: one is centralized and the other is decentralized architecture. Centralized architecture requires more administration while scaling up in the distributed file system. Decentralized architecture scaling can be managed by an administrator itself[20,21].

4.6 Security

Securityof the system can be achieved using three main aspects they are: confidentiality, integrity and availability. Confidentiality uses authentication techniques to protect our system from unauthorized users. Integrity uses message digest to identify and protect our system against data corruption. Availability protects system against failure and making the system always accessible[22].

4.7 Fault Tolerance

Fault tolerancehas to be provided when a failure occurs (can be a hardware or software) in a distributed file system. It must be provided with fault-tolerant capability such that the system should be able to recover and tolerate faults that occur. Consider if a system has multiple servers, if any one fails, then the load has to be distributed among the servers in transparent way[23].

5. Taxonomy of Distributed File Systems

In the Figure 1, the taxonomy of distributed file systems is to be organized most appropriately and suitable file system (the detail that establish the distributed file system) has to be considered for better performance, fault tolerant and to be secured one. Based on various factors the taxonomy is being discussed below:

5.1 Architecture

There are five types of architecture in distributed file system they are:

5.1.1 Client-Server Architectures

It maintains a uniform view of local file system (e.g. Sun Microsystems's Network File System). It appears with a set of communication protocols which shares a common file system in a machine that allows clients to access files stored on different processes running on different operating system. This is largely independent of local file system. The main issue of the system is it cannot be used in MS_DOS for its short file names[24-26].

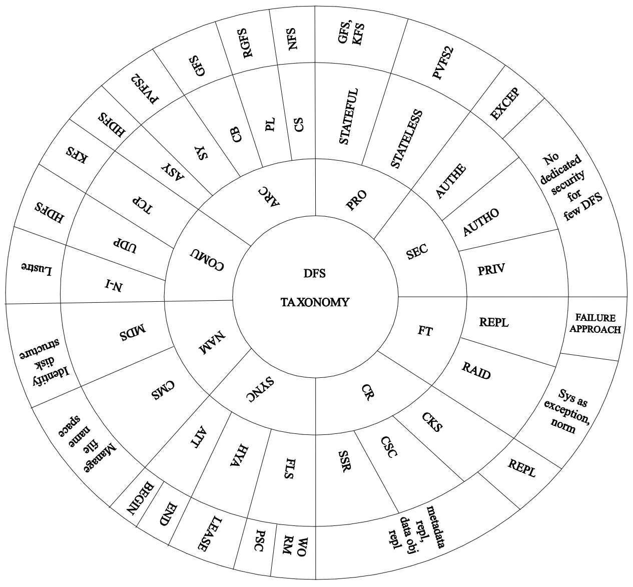

Figure 1. Taxonomy of Distributed File System.

5.1.2 Cluster-Based Distributed File System

It consists of multiple chunk servers contained in a single server and each chunk is divided into 64 Mbytes (e.g. Google File system). The advantage is that a single master which manages and controls few hundred chunks of server. In designing cluster-based DFS three important features are taken into Considerations they are: Decoupled metadata and data, Reliable autonomic distributed object storage and dynamic distributed metadata management[27].

5.1.3 Symmetric Architecture

It is based on peer-to-peer technology. Here clients also host the metadata manager code which results in understanding each of the disk structure. It uses a chord-DHT for distributing data combined up with a key based look-up mechanism[28] .

5.1.4 Asymmetric Architecture

It uses one or more metadata managers to maintain the file system and its associated disk structure (e.g. panasa active scale, luster, NFS). Parallel architecture contains data blocks that are stripped in parallel across multiple storage devices and servers[29]. Parallel application supports node access to same file at same time (i.e. CRCW) concurrent read and writes capabilities.

From the above survey it is found that multi-layers architecture allows flexibility and so functions or protocols layer can be added at ease[30].

5.1.5 Process

When concerning processes, a process can be either state-full or stateless. State-full or stateless report whether a computer program is designed to mark and extract one or more leading event in a given sequence of interaction with the users[31].

5.1.6 State-Full

It keeps track on the state of interactions (i.e.) it stores session state and keeps track of which client used the file, current read and write of the files and which client has blocked which file is being maintained[32].

5.1.7 Stateless

It has no note or log of previous interactions or interaction request has to be managed based on information that appears. It does not store any session state and each and every client is treated independently. The advantage of this process is if a client fails or resume it leaves without disturbing the entire or whole system (e.g. PVFS2)[33].

5.1.8 Communication

Remote procedure call is a communication method used in Distributed file system .it makes the system call independent to its underlying operating systems, network and transport protocols. It uses two different approaches they are TCP and UDP. Another approach for handling communication is plan 9 where all files are accessed based on syntax[34].

5.1.9 Naming

It is a process of mapping between logical objects and physical objects. Each object plays a major role that has an associated logical path and physical address. A location transparency is provided by DFS (i.e.) where the file is located is not provided to its user and mobility of file names is made without and alternation[35]. (Name of a file). To access the information of the DFS an object address is required. To access a remote file system contents a complete transparency has to be provided to client. The two approaches are central metadata server and metadata distributed in all nodes. A collection of logical path names constitute a distributed name space and it is logical separated into domain[36].

5.1.9.1 Central Metadata Server

It provides in managing the file name space. So partitioning of the metadata is made in improving the file namespace and in breaking synchronization problems.

5.1.9.2 Metadata Distributed in All Nodes

It makes nodes identifies the structure of the disk .On basis of security issues the name spaces are not shared between the users and file sharing becomes difficult.

5.1.9.3 Synchronization

Semantics of read and write is necessary when a same file is accessed and shared between two or multiple users. Semantics of files sharing is conceptually easy and difficult during implementation. Apart from semantic various approaches are available they are file locking system, hybrid approaches, and atomic transactions.

5.1.9.4 Atomic Transaction

It is based on two signals they are begin transaction and end transaction.

5.1.9.5 File Locking System

It is based on write-once-read-many and producer/single-consumer. Some systems choose between locks on objects to clients and perform all operations synchronously on the server to support their access model.

5.1.9.6 Hybrid Model

It uses leases to lock on object to the client and it controls parallel access to distributed file system.

5.1.10 Consistency and Replication

Checksum is typically used to authenticate the data that are sending to communication network. Furthermore, replication and caching plays a vital role in DFS. Replication of data provides high availability and consistency of data is maintained using caching, it is performed to improve system performance. It can be executed in client-side caching and server-side replication. Client-side caching asks the server whether the data cached is valid and validates the content requested by the client is being served with the desired information needed. Server-side replication where every open is notified by the server for example if a file is accessed or during any modification if caching is stuck for other clients of that file. Replication of data should consider the metadata replication and data object replication. Four places where data are held are on the server's disk, cache in server memory, in the client's memory, and on the client's disk[37]. Availability and recoverability of the data in ensured in DFS, which provides proxy of metadata server with that of the snapshots of metadata along with its transaction logs. Asynchronous replication method is used on a high bandwidth and is consumed when data objects gets replicated on various servers.

5.1.11 Fault Tolerance

A system has to provide a fault-tolerant capability to tolerate faults and recovering from faults that occur during hardware and software failure. To provide excess fault tolerance techniques like replication and redundant arrays of inexpensive disk (RAID) are used. Availability and transparency of failure is provided to users using replication. Failure as exception systems and failure as norm systems are methods for handling Fault tolerance[38]. In failure as an exception be made by eliminating or by isolation of failure node such that it could be easily recover the system from last normal running state. And replication concept is applied in failures as norm for all types of data and replicates when replication rate becomes uncertain.

5.1.12 Security

The key security issues in DFS are Authentication and access control. Security are provided in DFS using authentication, authorization and privacy[39]. Whereas few DFS does not need in any security mechanisms because they trust in communication laid between nodes and clients.

6. Conclusion

DFS is the classic model of sharing file system through which multiple users can store and share resources and it is permanent storage medium. In this paper an overview of various evolutions file system and overview of their processing details of each of the individual files are being discussed. Based on this study, a detailed study or survey of how to design a distributed file system, with that of the issues related in designing the system and with that taxonomy related to DFS are presented.

Finally, the encouraging results observed from this work serve as the motivation to apply future implementation of DFS within the context of current developments in distributed computing.

7. References

1. Alonso R, Barbara D, Cova LL. Using stashing to increase node autonomy in distributed file systems. Huntsville, AL: Proceedings of 9th Symposium on Reliable Distributed Systems. 1990; p.12–21.
2. Hac A. Modelling parallel access to shared resources in a distributed file system using queueing networks. Journal of Systems and Software. 1986; 6(1-2):61–9.
3. Akinlar C, Mukherjee S. A scalable distributed multimedia file system using network attached autonomous disks. San Francisco, CA: Proceedings of 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. 2000; p. 180–87.
4. Martini B, Choo KKR. Distributed filesystem forensics: XtreemFS as a case study, Digital Investigation. 2014; 11(4):295–313.

5. Gaidioz B, Koblitz B, Santos N. Exploring high performance distributed file storage using LDPC codes. Parallel Computing. 2007; 33(4-5):264–74.
6. Bian J, Seker R. The Jigsaw secure distributed file system. Computers & Electrical Engineering. 2013; 39(4):1142–52.
7. Amudhavel J, et al. An robust recursive ant colony optimization strategy in VANET for accident avoidance (RACO-VANET). International Conference on Circuit, Power and Computing Technologies (ICCPCT); Nagercoil. 2015. p. 1–6.
8. Amudhavel J, et al. A krill herd optimization based fault tolerance strategy in MANETs for dynamic mobility. International Conference on Circuit, Power and Computing Technologies (ICCPCT); Nagercoil. 2015. p. 1–7.
9. Amudhavel J, Prabu U, Dhavachelvan P, Moganarangan N, Ravishankar V, Baskaran R. Non-homogeneous hidden Markov model approach for load balancing in web server farms (NH2M2-WSF). Global Conference on Communication Technologies; Thuckalay. 2015. p. 843–5.
10. Mecozzi D, Minton J. Design for a transparent distributed file system. Monterey, CA, USA: Eleventh IEEE Symposium on Mass Stroage Systems, Digest of Papers. 1991; p. 77–84.
11. Yu J, Wu W, Li H. DMooseFS: Design and implementation of distributed files system with distributed metadata server. Shenzhen: IEEE Asia Pacific Cloud Computing Congress (APCloudCC). 2012; p. 42–7.
12. Fridrich M, Older W. Helix: The Architecture of the XMS Distributed File system. IEEE Software. 1985; 2(3):21–9.
13. Zhang T, Sun XZ, Xue W, Qiao N, Huang H, Shu J, Zheng W. ParSA: High-throughput scientific data analysis framework with distributed file system. Future Generation Computer Systems. 2015; 51:111–19.
14. Lanjewar U, Naik M, Tewari R. Glamor: An architecture for file system federation. IBM Journal of Research and Development. 2008; 4(5):329–39.
15. Carretero J, Perez F, de Miguel F, Alonso GL. Performance increase mechanisms for parallel and distributed file systems. Parallel Computing. 1997; 23(4,5):525–42.
16. Hac A. A benchmark for performance evaluation of a distributed file system. Journal of Systems and Software. 1989; 4(9):273–85.
17. Hac A. A validated performance model for distributed file systems. Journal of Systems and Software. 1989; 10(3):169–85.
18. Perez F, Carretero J, Garcia F, de Miguel F, Alonso L. Evaluting ParFiSys: A high-performance parallel and distributed file system. Journal of Systems Architecture. 1997; 43(8):533–42.
19. Zhou X, He L. A Virtualized Hybrid Distributed File System. Beijing: Proceedings of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberCC). 2013; p. 202–05.
20. Cho JY, Jin HW, Lee M, Schwan K. Dynamic core affinity for high-performance file upload on Hadoop Distributed File System. Parallel Computing. 2014; 40(10):722–37.
21. Liu T-J, Chung C-Y, Lee C-L. A high performance and low cost distributed file system. Beijing: 2011 Proceedings of IEEE 2nd International Conference Software Engineering and Service Science (ICSESS). 2011; p. 47–50.
22. Hung-Chang H, Hsueh-Yi C, Haiying S, Yu-Chang C. Load Rebalancing for Distributed File Systems in Clouds. IEEE Transactions on Parallel and Distributed Systems. 2013; 24(5):951–62.
23. Lee W, Su D, Srivastava J. QoS-based evaluation of file systems and distributed system services for continuous media provisioning. Information and Software Technology. 2000; 42(15):1021–35.
24. Thanh TD, Mohan S, Choi E, Kim SB, Kim P. A Taxonomy and Survey on Distributed File Systems. Gyeongju: Procedings of 4th International Conference on Networked Computing and Advanced Information Mangement, NCM'08. 2008; p. 144–49.
25. Hac A. Performance-reliability issues in distributed file systems. Journal of Systems and Software. 1986; 6(3):219–24.
26. Hurley RT, Yeap SA, Wong JW, Black JP. Potential benefits of file migration in a heterogeneous distributed file system. Sudbury, Ont.: Proceedings 5th International Conference on Computing and Information, ICCI'93. 1993; p. 123–27.
27. Peng H, Wang L-R, Wang J, Hagiwara I. Design and Implement of File Linked Distributed File System. Los Angeles, CA: WRI World Congress on Computer Science and Information Engineering. 2009; 7:305–09.
28. Verma A, Sharma U, Rubas J, Pease D, Kaplan M, Jain R, Devarakonda M, Beigi M. An architecture for lifecycle management in very large file systems. Proceedings of 22nd IEEE/13th NASA Goddard Conference on Mass Stroage Systems and Technologies. 2005; p.160–68.
29. Dwivedi K, Dubey SK. Analytical review on Hadoop Distributed file system. Noida: Proceedings of 5th International Conference on Confluence the next Generation Information Technology Summit (confluence). 2014; p. 174–81.
30. Wu Y, Ye F, Chen K, Zheng W. Modeling of Distributed File Systems for Practical Performance Analysis. IEEE Transactions on Parallel and Distributed Systems. 2014; 25(1):156–66.
31. Chen P, Li J, Gou XR. Research of distributed file system based on massive resources and application in the network teaching system. International Conference on Advanced Intelligence and Awareness Internet (AIAI 2011). 2011; p. 154–58.
32. Sharma N, Irwin D, Shenoy P. A distributed file system for intermittent power. Arlington, VA: 2013 International Green Computing Conference, (IGCC). 2013; p. 1–10.

33. Tseng F-H, Chen C-Y, Chou L-D, Chao H-C. Implement a reliable and secure cloud distributed file system. New Taipei: Proceedings of International Symposium on Intelligent Signal Processing and Communications Systems, ISPACS. 2012; p. 227–32.
34. Chu C-C, Hsu C-H. A Performance-Effective and High-Scalable Grid File System. Busan: Proceedings of International Conference on Multimedia and Ubiquitous Engineering, MUE'08. 2008; p. 460–65.
35. Hiroyasu T, Minamitani Y, Miki M, Yokouchi H, Yoshimi M. Distributed PACS using distributed file system with hierarchical meta data servers. San Diego, CA: 2012 Annual International Conference of the Engineering in Medicine and Biology Society (EMBC). 2012; p. 5891–94.
36. Rani LS, Sudhakar K, Kumar SV. A Survey International Journal of Computer Science and Information Technologies. 2014; p. 123–256.
37. Lauf AP, Peters RAL, Robinson WH. A distributed detection system for resource-constrained devices in ad-hoc networks. Ad-Hoc Networks. 2010; 8(3):253–66.
38. Gharehchopogh FS, Arjang H. A Survey and Taxonomy of Leader Election Algorithms in Distributed Systems. Indian Journal of Science and Technology. 2014 Jan; 7(6):815–30.
39. Kim BS, Kim TG, Song HS. Parallel and Distributed Framework for Standalone Monte Carlo Simulation using MapReduce. Indian Journal of Science and Technology. 2015 Oct; 8(25):1–8.