INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

ORGANISATION OF MULTIPLE FILE ENTRIES IN SECURE COMPUTING USING AN UNDERSTANDABLE THRUST METHODOLOGY WITH COMPOUND QUALITIES

B.Kalpana[1], Dr. S. Uma[2]

[1] PG Scholar, PG CSE Department, Hindusthan Institute of Technology, Coimbatore, Tamil Nadu, India, kalpanabalu1982@gmail.com
[2] Head of the Department, PG CSE Department, Hindusthan Institute of Technology, Coimbatore, Tamil Nadu, India, umakarunahind@gmail.com

Abstract: - Thrust is a consent authority methodology designed for organizing multiple file entries. It is more comprehensible than the existing traditional methods and provides an enhanced counterpart for constructing a sensible system. It is used to produce equivalent result oriented system, and it is also used for providing ingenious solutions for arrangement which is different from the existing system. In order to provide an enhance security thrust is used to separate the basic elements such as leader choice voting and replication log finding. It produces an excellent quality of coherency to reduce the number of solutions. The paper demonstrates in detail about thrust which includes a fresh methodology using overlaps mainstream tie to provide authentication for changing the group membership.

1 Introduction

The thrust methodology is used for assembling of the system to work as a dependable set which can endure the failures of sum of its existing members [1]. They are used for playing a main role in constructing dependable large scale software systems. The traditional technique has dominated the disputes of thrust methodology over the last decade. Most of the evaluations of thrust are based on traditional technologies which have become the major medium used to teach the learners about the thrust. But unluckily the traditional systems are very complicated to understand the practical systems cannot be built with the existing technology which needs major corrections to provide an effective running absolute system. Therefore the programmer and learners battle with the traditional system.

Thrust is very much comparable in countless ways to the existing replication methodology. But it has numerous narrative features.

- Head selection: Thrust user's random timer to select the head/leader. It makes only small corrections to the existing layout which is required for any replication methodology by solving major conflicts which appear constantly.

- Sole head: Thrust users a stronger structure of organization method than any other replication methodology. For example, the record entries will only flow from the head to other servers. This methodology simplifies the organization of replicated data and makes thrust very much easier for the learners and the programmers to understand.

- Member switching: Thrust methodology is used for changing the set of servers in a crowd by using a novel approach by configuring a system during state transitions. This allows the group members to operate normally during the course of chances in configuration.
  The reminder of the paper introduces the methodology to achieve fault tolerance in replicated systems; problems with traditional technologies, designing thrust methodology and log cleansing.

2. Methodology to achieve fault tolerance in replicated systems

The thrust methodology basically emerges in the frame work of virtual state systems. In this technique the system are designed by combining multiple servers and by combining identical copies of same types of systems which enable to function even if n number of servers is down. The replicated state systems are evaluated by means of using a replicated plot entries. The log entries are stored by means of the server by using a string of commands in sequence in which the system executes in order. Each of the log based system consists of identical command in identical order so that each system has the same string of commands. Since the state systems are deterministic, they are used to compute similar sequence of outputs by not disturbing the traditional system. The agreement module on the server will receive unique commands from the clients which are then added to the log system. The agreement modules are communicated to ensure that every log system will eventually contain identical request in original order even if some of the servers fail. Once the commands are properly replicated each dedicated server of the system will process them in a particular order and clients are supplied with the outputs. Therefore the servers will form highly reliable state systems. The basic methodology is shown in the following figure 1.

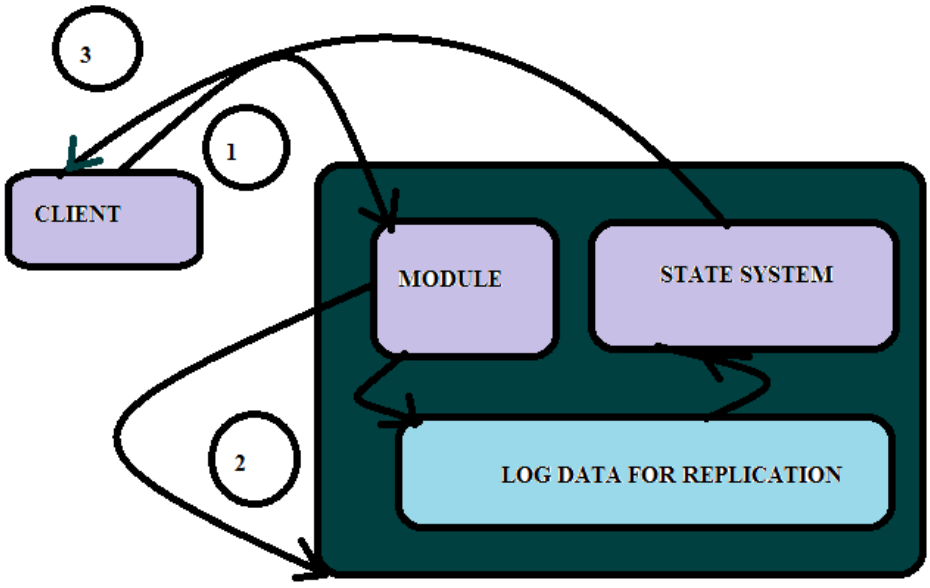

Figure 1: Basic methodology of Replicated System

The thrust methodology for realistic stimulated system will have the following classical properties

- They provide safety under any non-Byzantine conditions which includes packet loss, reordering, duplication of records, disk partitions and normal network delays by returning a concrete result in any situation.

- They are fully functional.

3. Drawbacks of traditional system for replicated data

Some of the existing system defines a protocol which is capable of reaching to an agreement on a single decision such as simple single log replicated entries. These types of systems combine multiple instances of some protocol and are used to facilitate series of decisions over the past decade Leslie Lamport's [2] protocol is most similar to thrust methodology. This is the most commonly thought protocol and most of its evaluation are very similar. It ensures protection by supporting chances in the crowd membership. The correctness of this methodology is proven successfully and it is used in many of the normal cases and has to considerable and remarkable drawbacks.

The first drawback is that they are extraordinarily difficult to understand and implement. The full description of the system given is extremely not clear and only very few programmers land up in understanding with immense efforts. The simple single log replicated entries are very dense and subtle which is broadly divided into two stages and which cannot be understood independently. So it is extremely tricky to develop simple single log replicated entries. The composition rules [3] for multiple logs will add considerable amount of complexity and additional subtleness. It is also believed that the whole problem of reaching upon a single decision leads to chaos and hence an alternate solution has to be derived using multiple decisions.

The second drawback is that it will not provide a strong basic foundation for constructing real time implementation. The one reason for using this methodology is that there no such commonly agreed upon methodology for replicated data. Lamport's descriptions are commonly used for single level replicated systems. He has derived some of the possible approaches but many of the fine line details are missing which is a considerable drawback

4 Designing of thrust methodology

4.1 Goals and understandability

It is very important to understand the methodology for larger audience with greater comfort. N number of points is to be considered for designing thrust among the existing approaches. There are many goals for designing thrust which ends up in providing complete and suitable base for system constructions. It also reduces the amount of frame work needed for the developers which leads to safe operating conditions and efficient basic operations of all this basic important goal and the most challenging goal is understandability fortunately the more evident approach is also more crisp in almost all cases. It leads to analyses which prove that two techniques are commonly applicable for a very higher degree of subjectivity.

The first technique is to simplify the space occupied by reducing the number of considerable space by taking no deterministic values [4] whenever possible. For example, empty entries are never allowed in the log which is considered to be inconsistent.

The second technique is the most commonly known methodology of problem breakdown/decomposition: i.e. the problems are divided into small individual modules which can be solved separately and the solutions can be combined together to provide a relatively independent solution.

4.2 The thrust methodology

The thrust [5] uses a method of remote procedure calls (RPCs) to solve the problem of replicated data by using a group of servers. It is performed by first selecting an eminent head/leader and then assigning the head with absolute task of managing the replicated data. The head initially accepts the log entries from the client and duplicate

them on the server, and tells the server the time to apply the log entries to the state systems. Why doing so the head has the capability to simplify the management of replicated data. For example the head has authority to decide where to keep the new entry in the log without consulting other server systems.

Given with the head approach the thrust will decompose major problems into considerably independent sub problems such as,

- Head selection
- Replication of log

Head selection: Every time when an existing head fails by disconnecting it from the server a new head will be elected to perform the job.

Replication of log: It is the responsibility of the head to duplicate the data across the crowd of data by accepting the log entries from the client system.

4.3 Thrust basics

The thrust consist of several server systems which has the capability to tolerate a maximum of twin failures. During this time each of the servers can be in any one of the three states. Such as head, applicant or a follower. In a normal operation mode there will be exactly only one head and the other servers are considered to be followers. The followers are in passive mode that is, they will not issue any RPC on their own but they will respond to the RPCs generated by the heads and applicants. The head will handle all of the requests given by the clients and contacts the followers. The follower again redirects to the head. When the existing head fails the applicant has the capability to elect a new head.

The paper divides the time into terms which are numbered with constitutive set of integers. Each term will begin with a selection of one or more applicant who attempts to become a head. If an applicant wins the election [6] then it serves as the head for rest of the term. In some cases an election will end up in crack vote. It is the case where the term will end with no head and a new term/new election will begin shortly. Thrust ensures to see that the server state followers will respond only to the request from other server. If a distinguished follower receives no such communication it becomes an applicant and will initiate the process of election. Most of the time only one head will be given for a single term. The term some time will act as logical clock in thrust and allow the thrust server to find outdated information such as stale head. Each of the servers has the capability to store the current term number which will keep on increasing monotonic over time. The current term are inter exchanged whenever the server wish to communicate. If any one of the servers current term is smaller than the then it drags itself and updates its current terms to the nearest large value.

When the applicant or the head discovers that its term is out of date it will then immediately revert to the follower. If the server receive a request with a stale term member it will immediately rejects the request. The entries of the RPCs append by means of the head to replicate the data entries which in term form a seasaw data. A seasaw data is used to trigger head election which is initiated by the applicant during election. The server will remind in a follower mode until it receives a valid RPCs [7] from a head or an applicant. The head will consistently send seasaw (RPC APPEND DATA WITH NO LOG ENTRY).

To all its followers so that they can maintain their own authority. When a follower receives no information for some period of time then it is called as time out session and it is considered that there is no suitable head and begins the election for the head. An applicant can remain in any one of the following states.

- An applicant can win the election and become the head
- A volunteering server establishes itself to be a head.
- A considerable amount of time with no winner/no head.

4.4 Replication of log entries

Once a head is been elected then it starts servicing the request from the client. Each of the requests from the client consists of specific commands to be executed by the replicated data. The head will append the command to

the existing log as a fresh new entry and will issue the seasaw data in parallel to other servers to replicate the data entry.

When the particular entry has been replicated safely then the head will apply the entry to state system and will return the results produced during execution to the communicating clients. If the followers run slowly or when the network packets are lost or the follower's crash or when the head tries/retries to send a seasaw data then all the followers will store all the log entries [8]. The logs are organized as shown in the figure 2.

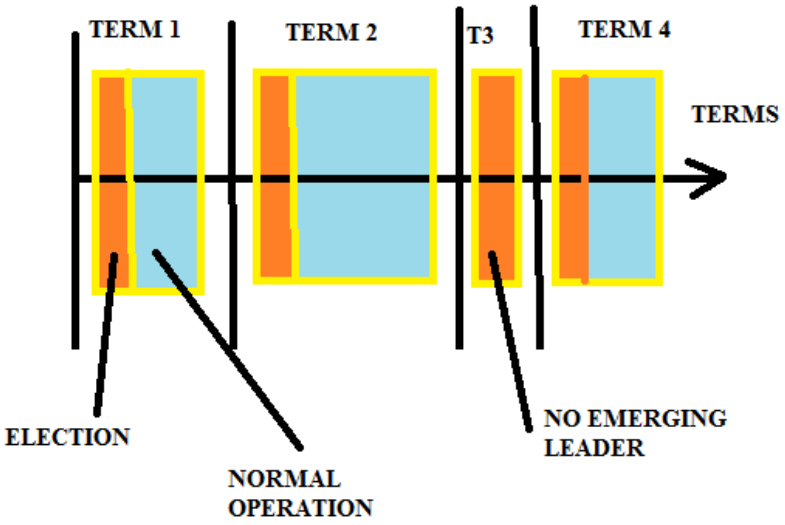

Figure 2: Replicated Log Time Division

Each log entry will store the state system command along with the term number and will add a detail such as the time when it is obtained from the head. In order to detects the inconsistency between the log entries the term numbers are used. The position of the log is identified by the means of the index integer. The head also decides the time which is safe to apply a log data entry into the state system and such an entry is called as COMMIT-COMMIT [9]. Thrust guarantees that COMMIT-COMIMIT entries are strong, robust and will always be executed during the availability of state system. Thrust maintains the following set of properties such as

- The logs are said to be identical, if two entries in different logs have the same index and the term.
- The logs are used to store the same command, if two entries in different logs have the same index and the term.

The first property is cross checked by simple consistency checks performed by the RPCs. If the follower does not find the entry with the same index and the term it will refuse for new entry addition.

The second property is crossed checked by the fact that the head will create almost one entry and will never change the position in the log entry.

5. Security and authentication

5.1 Protection of replicated system

The previous sections describe specifies how thrust selects a new head and how it stores the replicated [10] log data. This section will complete the thrust methodology with two additional features:

- Restricting which server can be a head.
- Restricting which entry can be a COMMIT-COMMIT.

These restrictions enable the head for any specified term containing all the entries COMMIT-COMMIT [11] in the previous terms they also contain additional mechanism to identify missing entries and forward it to the new head either during the selection process or shortly later. Thrust uses the simple approach that the log entries will be

forwarded only in one direction from the head to the followers. The heads will never ever over right the existing the log entries of the data.

5.2 Applicant and follower crash

In the previous sections of the paper I have focused on head failure. Applicant and Follower crashes are much easier to handle than head crashers. When the applicant and the follower crashers further request for seasaw data will fail. Thrust will handle this type of failures by infinite retries.

The server will resume itself as a follower and generate RPC successfully. If the server crashes after completing the RPC request then it will receive the same RPC again and again. Therefore the thrust RPCs are harmless and the system will not produce incorrect result.

6. Compaction of the log

The thrust log cannot grow up infinite in the real time systems which will occupy more space in memory and will take large amount of time to reply. There are two important approaches for log compaction.

- Log cleansing
- Log snapping

Log cleansing is the method of investigating the log data entries to determine whether they are alive. The head of the log is rewritten with live data entries and large volume of deceased logs is freed. This process of cleansing the log is said to be efficient by determining which portion of log can be cleaned and find out which live entries are complex.

Log snapping is the method of taking snap shot of the original log.

Therefore thrust can rely on seesaw data for a safe organization of multiple file entries.

7. Conclusions

Many of the methodology are designed with dominant factors with correctness, efficiency to satisfy the basic goals of the system. Even though there are many such important goals I believe that understandability is equivalently important a goal can be achieved only with clear perception, proper understanding and real time execution. Until and unless the developers have a knowledges of understanding the developed methodology and cannot create a clear frame work and will end up in difficult evaluation. In this paper I have given a clear idea of organizing and managing the replicated logs which has always remained a challenge for developers and programmers for many years.

REFERENCES

[1] BOLOSKY, W. J., BRADSHAW, D., HAAGENS, R. B., KUSTERS, N. P., AND LI, P. Paxos replicated state machines as the basis of a high-performance data store. In Proceedings of the 8th USENIX conference on Networked systems design and implementation (Berkeley, CA, USA, 2011), NSDI'11, USENIX Association, pp. 11-11.

[2] LAMPORT, L. Time, clocks, and the ordering of events in a distributed system. Commun. ACM 21, 7 (July 1978), 558-565.

[3] LAMPORT, L. The part-time parliament. ACM Trans. Comput.Syst. 16, 2 (May 1998), 133-169.

[4] LAMPORT, L. Paxos made simple. ACM SIGACT News 32, 4 (Dec. 2001), 18-25.

[5] LAMPORT, L. Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, 2002.

[6] LAMPORT, L. Generalized consent and paxos. http://research.microsoft.com/apps/pubs/ default.aspx?id=64631, 2005.

[7] LAMPORT, L. Fast paxos. http://research.microsoft.com/apps/pubs/default.aspx?id=64624, 2006.

[8] LAMPSON, B. W. How to build a highly available system using consent. In Distributed Methodology, O. Baboaglu and K. Marzullo, Eds. Springer-Verlag, 1996, pp. 1-17.

[9] LAMPSON, B. W. The abcd's of paxos. In Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (New
York, NY, USA, 2001), PODC 2001, ACM, pp. 13-13.

[10] LISKOV, B., AND COWLING, J. Viewstamped replication revisited. Tech. Rep. MIT-CSAIL-TR-2012-021, MIT, July 2012.

[11] Log Cabin source code. http://github.com/logcabin/logcabin.

A Brief Author Biography

B Kalpana is an Assistant Professor and Post Graduate Scholar of PG Department of Computer Science and Engineering at Hindusthan Institute of Technology, Coimbatore, Tamil Nadu, India. She received her Bachelor of Technology Degree in Information Technology from Sri Venkateswara College of Engineering Chennai with First Class and Distinction affiliated to Madras University in 2003 and is currently pursuing M E degree (Part Time) at Hindusthan Institute of Technology Coimbatore affiliated to Anna University. She received her Diploma Degree in Computer Science from Panimalar Polytechnic Chennai with First Class and Distinction with a Gold Medal affiliated to Directorate of Technical Education in 2000.She has several high-level involvements in the area of Artificial Intelligence and Big data. She has nearly 9 years of academic experience in the field of Engineering and guided many under graduate projects.

Dr. S.Uma is Professor and Head of PG Department of Computer Science and Engineering at Hindusthan Institute of Technology, Coimbatore, TamilNadu, India. She received her B.E., degree in Computer Science and Engineering in First Class with Distinction from PSG College of technology in 1991 and the M.S., degree from Anna University, Chennai, TamilNadu, India. She received her Ph.D., in Computer Science and Engineering Anna University, Chennai, TamilNadu, India with High Commendation. She has nearly 24 years of academic experience. She has organized many National Level events like seminars, workshops and conferences. She has published many research papers in National and International Conferences and Journals. She is a potential reviewer of International Journals and life member of ISTE professional body. Her research interests are pattern recognition and analysis of nonlinear time series data.