IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 13
ISSN: 2320 - 8791
www.ijreat.org

An Efficient Processing Of Regular Register Prone To Continuous Change

Deepak Singh P[1], Anish Kumar R[2],Dr.P.N.Jebarani Sargunar[3]

[1,2&3]**C.S.E Department, S.A.Engineering College, Chennai-77, Tamilnadu, India**

Abstract

In generally dealing with failures has been one of the main challenges in the construction of real reliable applications able to work in a distributed system. Implementations of Shared objects, in modern distributed systems, have to take into account the fact that almost all services, implemented on top of distributed infrastructures, are no longer fully managed due to their size or their maintenance cost. This system addresses the construction of a multi writer/multi reader regular register in an eventually synchronous distributed system affected by the continuous arrival/departure of participants. In particular, a general protocol implementing a regular register is proposed ,It has been formally proved that a regular register can be implemented in an eventually synchronous distributed system. Also we performed in here the enhancement by using Consensus algorithm. Consensus is the process of agreeing on one result among a group of participants. In here reconfiguration service is implemented by a distributed algorithm that uses distributed consensus to agree on the successive configurations. In addition we enhanced Elo rating system, which is a method for calculating the relative skill levels of players in two-player games such as chess or more player
process.

Keywords:*churn, regular register, consensus algorithm, Elo-rating.*

1. Introduction

1.1 Perspective

A distributed system is a set of physically separate processors connected by one or more communication links. User should be presented with the "single-system image" regardless any particular hardware platform, operating system and network. Air traffic control, telecommunication, banking systems, and e-booking systems are just a few examples of such application domains. The composition of the system is modified only when a new process is added or either a process crashes. Therefore, if a process does not crash, it lives for the entire duration of the computation.

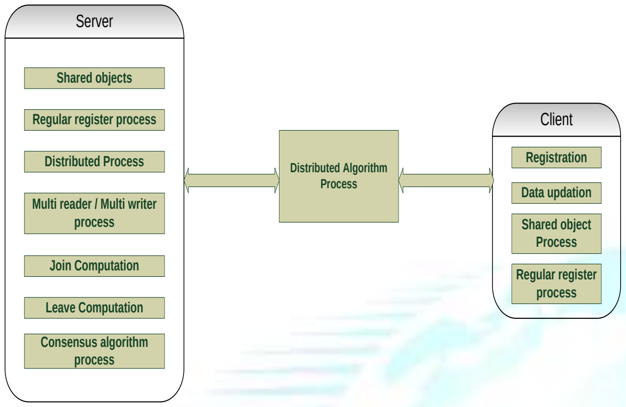
1.2Drive

Distributed system now requires some degree of self management. In general, every player is eager to know the competitor status in a competition. Knowing that some may quit and some may proceed with more effort. Here in this paper the same [5] churn notion is used as parameter to make tractable systems having their composition evolving along the time.

1.3 Involvement

No process is guaranteed to participate for-ever in the distributed computation. To check this churn notion c as a parameter is used [6]. [5].A regular register can have any number of writers and any number of readers [12].with this we make an enhancement by consensus algorithm and Elo rating system,from which agreement and relative skill of the participant is known.

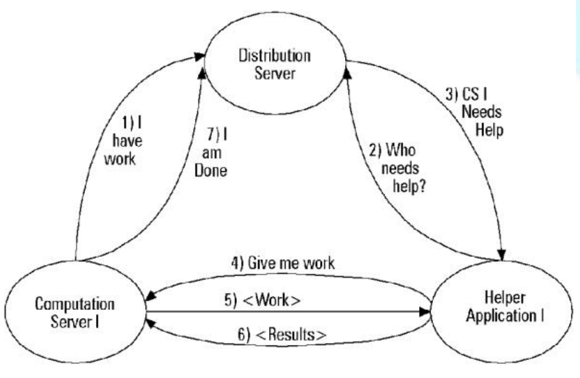
2. Proposed system model



2.1 Distributed Computing

Distributed computing is the next step in computer progress, where computers are not only networked, but also smartly distribute their workload across each computer so that they stay busy and don't squander the electrical energy they feed on. If enough users sign up, these linked computers — often referred to as virtual parallel machines — can surpass the fastest supercomputer by as much as four times for a fraction of the supercomputer's cost. More power for less money. All the computers on the Internet may be powerful when combined, but there needs to be something to combine and coordinate all of them to work towards one goal. Server computers are still needed to distribute the pieces of data andcollect the results from participating clients. Air trafficcontrol, telecommunication, banking systems, and e-government systems are just a few examples of such application domains.

Figure 1 Communication procedure



Two protocols join and leave computation are the one to make a process in and out of the distributed system.

2.2 The challenges of distributed systems

1. Secure communication over public networks (ACI: who sent it, did anyone see it, did anyone change it).
2.Fault-tolerance (Building reliable systems from unreliable components nodes fail independently; a distributed system can "partly fail".
3. [11] A distributed system is one in which the failure of a machine I've never heard of can prevent me from doing my work. (Replication, caching, naming).
4. Placing data and computation for effective resource sharing, hiding latency, and finding it again once you put it somewhere (Coordination and shared state, "What should the system components do and when should they do it?").
5. Once they've all done it, can they all agree on what they did and when?.

2.3 Regular Register Process

The notion of a regular register defined in the introduction has to be adapted to a dynamic system. We consider that a protocol implements a regular register in a dynamic system if the following properties are satisfied.
* **Liveness**: If a process invokes a read or a write operationand does not leave the system, it eventually returns from that operation.
* **Safety**: A read operation returns the last value writtenbefore the read invocation, or a value written by a write operation concurrent with it.
It is easy to see that these properties boil down to the classical definition if the system is static. Moreover, it is assumed that a process invokes the read or write operation only after it has returned from its join() invocation.

2.4 Distributed Algorithm

Distributed algorithms are typically executed concurrently, with separate parts of the algorithm being run simultaneously on independent processors, and having limited information about what the other parts of the algorithm are doing. One of the major challenges in developing and implementing distributed algorithms is successfully coordinating the behavior of the independent parts of the algorithm in the face of processor failures and unreliable communications links. The choice of an appropriate distributed algorithm to solve a given problem depends on both the characteristics of the problem, and characteristics of the system the algorithm will run on such as the type and probability of processor or link failures, the Kind of inter process communication that can be performed, and the level of timing synchronization between separate processes.

2.5 Multireader/Multiwriter

We presented an implementation of the regular register for a synchronous distributed system. Such implementation is based on the following considerations: 1) the join register () operation is executed once from each process and 2) read () and write () operations are executed frequently. In here we also enhanced into the Consensus algorithm. Consensus is the process of agreeing on one result among a group of participants. In here reconfiguration service is implemented by a distributed algorithm that uses distributed consensus to agree on the successive configurations. In addition we enhanced Elo rating system, which is a method for calculating the relative skill levels of players in two-player games such as chess or more player process.

3. Consensus Algorithms

In distributed computing, a classic problem is achieving consensus among multiple parties. In a design with multiple parties, there may be times when a majority (or potentially all) of the processes involved need to agree. The protocols relating to achieving this agreement are typically called consensus algorithms. One of the harder problems with achieving consensus is dealing with and recovering from failure of processing during the consensus activity. Most algorithms are focused on achieving as consistent and as reliable a consensus while anticipating failure. The goals of most consensus algorithms usually include:

Validity—The final answer that achieves consensus is a valid answer.
Agreement—All processes agree as to what the agreed upon answer was by the end of the process.
Termination—The consensus process eventually ends with each process contributing.
Integrity—Processes do not vote more than once.

Many consensus algorithms contain a series of events (and related messages) during a decision-making round. Typical events include Proposal and Decision.

According to paxo's algorithm[14]

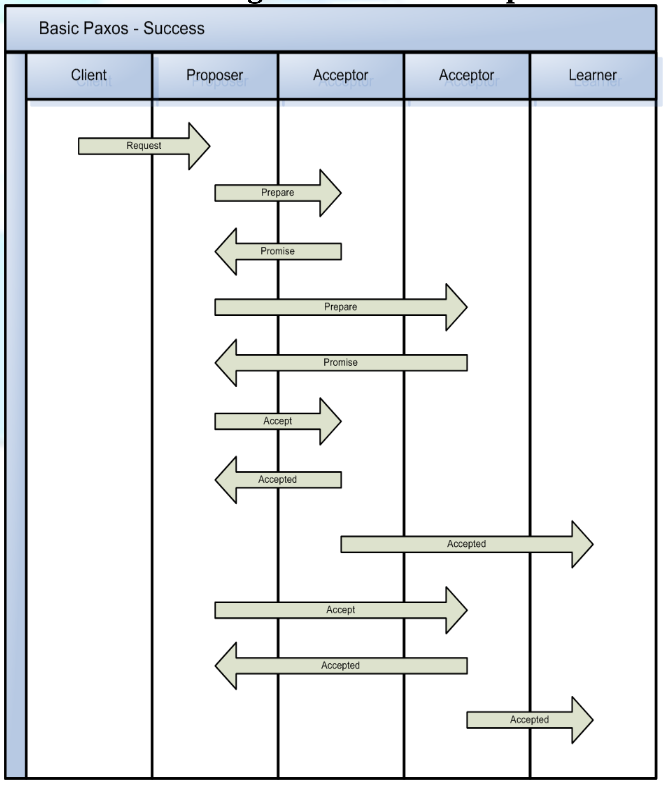
1. Prepare—The Leader and Proposer sends a Prepare message to the Quorum of Acceptors. This Prepare message includes a proposal number, n.
2. Promise—The Acceptors respond with a Promise to not accept Proposals with numbers less than the proposal number in the Prepare message. They must first verify that they have not already made a similar Promise with a proposal number greater than *n*.

If they cannot Promise, they send a denial message. If they can Promise, they may optionally send a proposed value for the Proposal they believe is the correct one.

3. Accept—The Proposer sends a value for the Quorum to agree upon. If no values were included in the Promise messages, the Proposer is free to choose any value it believes is correct. If values were returned with the Promise messages, the Proposer must select one of those.

4. Accepted—This is the response message from the Acceptors within the Quorum, indicating they have accepted the value. This message is sent by the Acceptors to the Proposer and by each Acceptor to each Learner (think of the Leaner as asecondary storage mechanism in this case.)

An illustrative diagram of consensus process



4.Conclusion

When change occurs continuously in a distributed system, it creates consistency violations. To track the churn notion as referred in [5] is implemented and also global acceptance of the data is made using the consensus algorithm [14]Consensus algorithms can be used directly to implement an atomic data service by allowing participants to agree on a global total ordering of all operations. In contrast, we use consensus to agree only on the sequence of configurations and not on the individual operations. Also, in Consensus algorithm, processed into the termination of consensus affects the terminations of reconfiguration, but not of read and write operations. In addition enhanced into the information about others updating process of impact and also provide Elo rating system, here the Elo rating system is a method for calculating the relative skill levels of players in two-player games such as chess or more player process.

References

[1] R. Baldoni and A.A. Shvartsman, "Theoretical Aspects of Dynamic Distributed Systems: Report on the Workshop," SIGACTNews, vol. 40, no. 4, pp. 87-89, 2009.
[2] T. Chandra and S. Toueg, "Unreliable Failure Detectors for Reliable Distributed Systems," J. ACM, vol. 43, no. 2, pp. 225-267, 1996.
[3] M.K. Aguilera, I. Keidar, D. Malkhi, and A. Shraer, "DynamicAtomic Storage without Consensus," Proc. 28th Ann. ACM Symp.
Principles of Distributed Computing (PODC), pp. 17-25, 2009.
[4] Roberto Baldoni, Member, IEEE, Silvia Bonomi, and Michel Raynal "Implementing a Regular Register in an Eventually Synchronous Distributed System Prone to Continuous Churn"
[5] R. Baldoni, S. Bonomi, A.M. Kermarrec, and M. Raynal, "Implementing a Register in a Dynamic Distributed System," Proc. 29th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '09), June 2009.
[6] R. Baldoni, S. Bonomi, and M. Raynal, "Regular Register: An
Implementation in a Churn Prone Environment," Proc 16th Int'l Colloquium on Structural Information and Comm. Complexity(SIROCCO), pp. 15-29, 2009.
[7] B. Godfrey, S. Shenker, and I. Stoica, "Minimizing Churn in Distributed Systems," Proc. Conf. Applications, Technologies, Architectures,and Protocols for Computer Comm. (SIGCOMM), pp. 147-158, 2006.
[8] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the Presence of Partial Synchrony," J. ACM, vol. 35, no. 2, pp. 288-323,
1988
[9] S. Ko, I. Hoque, and I. Gupta, "Using Tractable and Realistic Churn Models to Analyze Quiescence Behavior of Distributed Protocols," Proc. 27th IEEE Int'l Symp. Reliable Distributed Systems(SRDS '08), 2008. 108 ieee transactions on parallel and distributed systems, vol. 23, no. 1, january 2012[9,12,23]
[10] F. Kuhn, S. Schmid, J. Smit, and R. Wattenhofer, "A Blueprint forConstructing Peer-to-Peer Systems Robust to Dynamic Worst-Case Joins and Leaves," Proc. 14th IEEE Int'l Workshop Quality ofService (IWQoS), 2006.
[11] L. Lamport, "On Interprocess Communication, Part 1: Models,
Part 2: Algorithms," Distributed Computing, vol. 1, no. 2, pp. 77-101, 1986.
[12] C. Shao, E. Pierce, and J. Welch, "Multi-Writer Consistency
Conditions for Shared Memory Objects," Proc. 17th Int'l Symp.
[13] H. Attiya, A. Bar-Noy, and D. Dolev, "Sharing Memory Robustlyin Message-Passing Systems," J. ACM, vol. 42, no. 1, pp. 129-142,1995.
[14]Consensus Algorithms with EC2.