International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 7, July - 2013

Global State Recognition Algorithms In Distributed Systems

T. Swathi[1] M. Padma[2] M. Lakshmi Neelima[3]

Asst.Prof,CSE Dept Asst.Prof , CSE Asst.Prof , CSE Dept

G.Pulla Reddy Engineering College, Kurnool, Andhra Pradesh

Abstract**—***Large Scale distributed systems and peer-peer networks are envisioned to be deployed for a wide range of applications. Processes in distributed system communicate by sending and receiving messages. A process can record its state and message it sends and receives.*

Computing the global state of a system is a difficult task in distributing computing. Global states are also useful in monitoring the stable properties of the system. There are many existing algorithms for computing global state which have high response time and require large number of messages.

This paper presents a suite of algorithms DPREU and DPRRU by which a process in distributed system can determine a global state of the system and that data is accessed through different clusters. These algorithms are fast and require less number of messages.

Keywords-Distributed Systems, Minimal Spanning Trees, Clusters, Message Communication Systems, Global States.

I. INTRODUCTION

A distributed computing system consists of spatially separated process that do not shares a common memory and communicate asynchronously with each other by passing messages over communication channel. Each component of a distributed system has a local state. The state of process is characterized by the state of its local memory and a history of its activity. The state of channel is characterized by the set of messages sent along the channel less the messages received along the channel. The global state of the distributed system is a collection of the local states of its components.

Recording the global state of a distributed system is an important paradigm and it finds applications in several aspects of distributed system design. A global state of a system consists of $<U_i\{LS_i\}U_{i.j}\{SC_{i.j}\}>$,Where Ls_i is a local state of a processor p_i and $Sc_{i.j}$ is the state of the channel $c_{i.j}$ from the processor p_i to p_j In the system with non-FIFO channels, $Sc_{i.j}$={ messages sent up to Ls_i}\{ messages received up to Ls_j}.

Recording the distributed global states is a fundamental problem in asynchronous distributed systems.

The global states have applications in fault tolerance of long running programs by providing an intermediate Checkpoint of the system. In case of a failure, the system can restart from the checkpoint instead of the beginning of the program.

There are many mechanisms for taking checkpoints in Distributed systems; two broad categories are coordinated and uncoordinated check pointing. In uncoordinated check pointing, processes autonomously take a checkpoint. The drawback of uncoordinated checkpoints is that checkpoints may be wasted because they are not consistent with checkpoints taken by other nodes. Further, multiple checkpoints need to be recorded at each process because the latest checkpoint at a process may not be consistent with the latest checkpoints at other process. Worse, this requires rollback to a much older system state and redoing computation from that older system state. In coordinated check pointing, the nodes collectively record a consistent global state of the system so that no checkpoint is wasted. Coordinated check pointing based on recording global snapshots is the preferred way of check pointing in the high performance massively parallel systems community.

Global states are also useful in monitoring stable properties of the system. A property is stable if, once it becomes true, it stays true. Some examples of stable properties are termination, deadlock, loss-of-a-token, etc. By repeatedly computing the global snapshot and evaluating the property on the computed snapshot, one can detect any stable property.

Existing algorithms for computing global states like seminal algorithm by Chandy and Lamport [1] requires sending a special control message called the marker message on each of the logical channels in the system. In the typical case where there exists a fully connected overlay on the network graph, this amounts to a $O(n^2)$message.

In this paper we present two algorithms which are fast and require less number of messages. This makes the algorithm highly scalable, in addition to having the property that the roles of all the processes are symmetry. Process symmetry implies greater potential for balanced workload and congestion-freedom. These algorithms assume non-FIFO channels.

This paper is organized as follows. Section-2 provides related work on global states over Non-FIFO channels and Section-3 provides related work on MST, Section-4 provides DPREU and DPRRU algorithms. Finally section-5 provides Conclusions and Results.

II RELATED WORK ON GLOBAL STATE OVER NON-FIFO CHANNELS

The Lai-Yang algorithm [2] works as follows:
1. Each process is initially white and turns red while taking a local snapshot.
2. A white (red) process sends white (red) colored messages.
3. Each process takes a local snapshot at any time before receiving a red message.
Each process keeps a log of all messages sent and received along the channel. After the local snapshots are collected at an initiator, the in-transit messages $SC_{i,j}$ are computed as the set-theoretic difference per channel $C_{i,j}$

The Chandy-Lamport algorithm [1] for a FIFO system and its variant by Mattern [3] for a non-FIFO system use a marker per logical channel. The deficiency counting algorithm which are introduced by Mattern [3] works as follows.

The algorithm uses the white/red coloring and does not use Markers. Each processor keeps a counter that counts the number of messages sent minus the number of messages received. The counter gets recorded as part of the local state recording. When the initiator collects the local snapshots, it knows how many white messages are in transit to the processors in the snapshot. Each red processor is required to report to the initiator each in-transit (white) message that is received. This continues until the deficit count is matched at the initiator.

The two-dimensional grid-based algorithm by Garg [4, 5] uses $O(n^{3/2})$ messages, each of size $\sqrt{n}$, by assuming a logical grid overlay on the underlying architecture and by using message coloring. The centralized and tree-based algorithms by Garg are based on deficit counting of the sum of intransit messages, as used by Mattern in his deficiency counting algorithm.

III CONSTRUCTING MINIMAL SPANNING TREES

The distributed minimum spanning tree (MST) problem is one of the most important problems in the area of distributed computing. There has been a long line of research to develop efficient distributed algorithms for the MST problem starting with the seminal paper of Gallager[6] that constructs the MST in O(n logn) time and O(|E|+n logn) messages. The communication (message) complexity of the Gallager[6] algorithm is optimal, but its time complexity is not optimal. Hence further research concentrated on improving the time complexity. The time complexity was first improved to O(n log logn) by Chin and Ting [7], further improved to $O(n \log^* n)$ by Gafni [8], and then improved to *existentially optimal* running time of O(n) by Awerbuch[9]. The O (n) bound is existentially optimal in the sense that there exists graphs where no distributed MST algorithm can do better than $\Omega(n)$ time.

Computing the low weight spanning sub graphs of a given graph G (V; E) with non-negative edge weights is a fundamental problem in network design. One important problem in this setting is the k-vertex connectivity problem (henceforth simply the k-connectivity problem) find a spanning sub graph of minimum weight that is k-vertex-connected, i.e., there exists k vertex-disjoint paths between every pair of vertices. Finding an optimal k-connected spanning sub graph is NP-hard for k ¸ 2 even if the weights of the edges satisfy the triangle inequality, or even when the graph is a complete Euclidean graph. There has been a lot of work on designing approximation algorithms for the k-connectivity problem. Most of these algorithms are centralized algorithms which are quite sophisticated and their main goal is to obtain a polynomial time algorithms with the best possible approximation ratio.

With the emergence of the new networking technologies such as ad hoc and sensor networks, there is an increasing need for distributed algorithms that are simple and easily implementable, have low communication complexity, and perform reasonably well. Such simple local algorithms are desirable even for the MST problem, where optimal distributed algorithms are known because these algorithms are quite complex, involve a lot of message complexity and synchronization to implement in a light weight and unreliable environment, such as ad hoc networks. This motivates the question of developing simple, local control, approximate algorithms.

This also adds a new dimension to the design of distributed algorithms for such networks: we can potentially tradeoff *optimality* of the solution to the amount of resources (messages, time etc) consumed by the algorithm. This is the motivation for the relatively new area of distributed approximation (Elkin [10]).

We now mention the previous work on distributed algorithms. Most of these algorithms assume that the graph is unweighted and the goal is to find a sparse k-connected subgraph. The algorithm of Cheriyan [11] finds k edge-disjoint breadth fist (BFS) forests, which gives a k-connected subgraph. The distributed implementation of this algorithm has time and message complexity as $O(knlog^3n)$ and $O(k|E|+knlog^3n)$ respectively. Thurimella [12] improved the time complexity to $O(kD+kn^{0.614})$where D is the diameter of G, but the message complexity was ignored and can be much larger than the previous algorithm. Using similar ideas, Jennings developed a distributed algorithm for the k-vertex connected subgraph problem which takes O(n) time and O(|E|) messages. In the same paper, they also presented a distributed algorithm for the k-edge connectivity problem which takes $O((k+D)log^3n)$ time and $O(|E|+knlog^3n)$ messages. All the above algorithms produces a k-connected subgraph with O(kn) edges from an unweighted k-connected graph G.

IV DATA PROVISION RESOURCE EXPLOIT AND UTILIZATION (DPREU) AND DATA PROVISION RESOURCE REPLICATION AND UTILIZATION (DPRRU)

The advent of high speed distributed systems has made it feasible for a large number of geographically dispersed computers to cooperate and share objects. An attempt is made with this set of algorithms to minimize communication delays with efficient use of network resources and providing access to shared objects.

The task of designing efficient algorithms for supporting access to shared objects over wide area networks is challenging both for practically and theoretically. The main source of difficulty in designing an access scheme that is efficient in both time and space is the computing considerations of these measures. In this paper a simple randomized access scheme is designed that exploits locality and distributes control information to achieve low memory.

The centralized and tree-based algorithms by Garg [4, 5] limited their solution to deal with resource requirements of the single clusters.

In this model two algorithms are designed that manages multiple clusters, one is for resource discovery and utilization in between node groups that labeled as "Grid level Data Provision resource exploit and utilization [DPREU] model" and other one is for resource replication and utilization between nodes with in node groups that labeled as node group level "Data provision resource replication and utilization" [DPRRU] model.

A network is modeled as an undirected weighted graph $G = (V;E;w)$ where V is the set of the nodes (vertices) and E is the set of the communication links between them and $w(e)$ is the weight of the edge e *2* E. Each node hosts a processor with limited initial knowledge.

The communication between any two nodes happens by sending/receiving messages along the edge between them in the graph G. We assume that the communication is synchronous and occurs in discrete pulses (time steps). This assumption is not essential for time complexity analysis. One can use a *synchronizer* to obtain the same time bound in an asynchronous network at the cost of some increase in the message complexity [13]. We assume that $O(\log n)$ bits can be transferred in one step per edge and a node can send messages through its entire incident links at the same time.

Initially all the nodes in the distributed systems are grouped into clusters. Minimal spanning trees are constructed separately for each cluster. Inter cluster graph is build by reading the coast between each cluster. By applying DPREU a cluster is selected as root where global data is placed. DPRRU replicates the data in each cluster that is accessed by all the nodes present in each cluster. By using DPREU average cost between each cluster is noted. Inter cluster graph is constructed which is MST for the whole network. With this the time complexity can be reduced. DPRRU places the data in each and every cluster at the root node of minimal spanning tree. Thus providing access of data for other local nodes.

V CONCLUSION AND RESULTS

A simple randomized approximation scheme for constructing a low-weight k-connected spanning sub graph with this paper, it also presents an efficient implementation in a complete network of processors. The proposed algorithm has low time and message complexity while giving a relatively good approximation ratio for the metric graphs, random geometric graphs, and random edge-weight graphs. It is interesting to see whether the ideas in this paper can be used to design an efficient distributed algorithm for the more challenging problem of finding a k-connected sub graph in an arbitrary general graph.

The local nature of the NN-scheme seems suitable for designing a simple and efficient *dynamic* algorithm (especially in a distributed setting), where the goal is to maintain a k-connected graph of good quality, as nodes are added or deleted. This looks promising for future work.

These algorithms are also suitable for constructing Minimal spanning tree using multiprocessor computer systems. There are many reasons for constructing minimal spanning trees in computer communications networks since minimal spanning tree routing is useful in distributed operating systems for performing broadcast in adaptive routing algorithms for transmitting delay estimates, and in other networks like packet Radio Networks.

Below figure (Fig.1) is a result of DPREU and DPRRU algorithms. An Inter cluster graph is constructed considering all the clusters. The time and message considerations are done by taking 80 nodes, grouping those to nine clusters. In the second step MST are constructed for each cluster separately. Then DPREU is applied to construct Inter cluster graph reading the cost between clusters and making one cluster as root (the highlighted cluster) considering the threshold. At last DPRRU places the data at root node for each cluster.

As there is an overhead in distributed communication as the processors may not be stable, this paper suggest an idea to group the nodes into clusters thus making communication management easier which leads to less number of messages sent and received between processors . These algorithms are also scalable as the new nodes can be placed or deleted in any cluster. Additionally these algorithms are congestion free and have balanced workload as all the processors share the work equally.

Finally these algorithms are used as distributed termination detection this is known when local states are of no interest then no messages are sent or received and the message counter is equally zero.

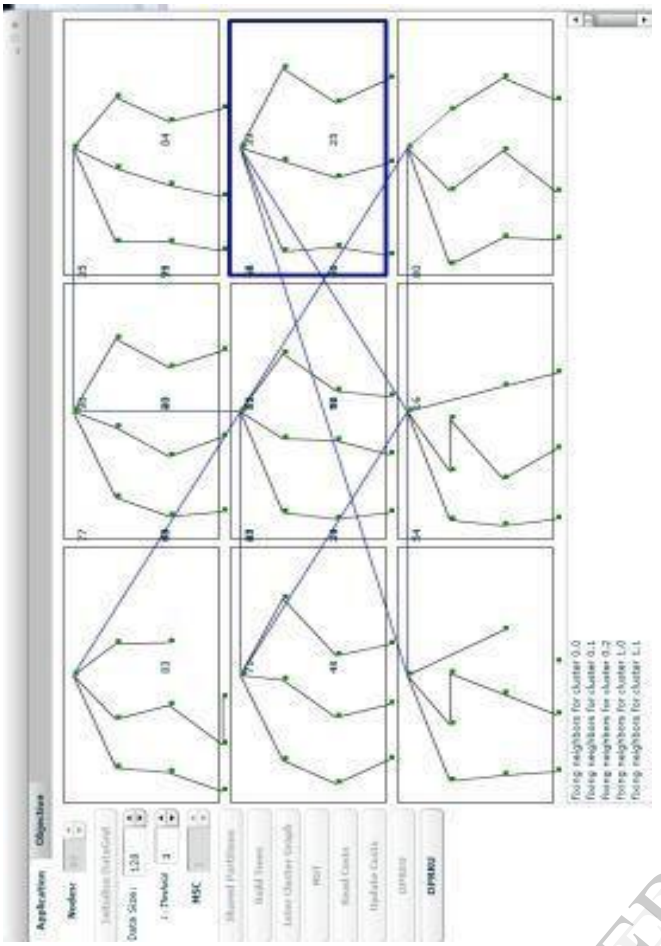

Fig.1
INTERCLUSTER GRAPH CONSTRUCTED BY APPLYING DPREU ALGORITHM

REFERENCES

[1] K.M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," ACM Trans. Computer Systems, vol. 3, no. 1, Feb. 1985.
[2] T.-H. Lai and T. Yang, "On Distributed Snapshots," Information Processing Letters, vol. 25, no. 3, pp. 153-158, 1987.
[3] F. Mattern, "Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation," J. Parallel and Distributed Computing, vol. 18, no. 4, pp. 423-434, 1993.
[4] S.Agarwal, R. Garg, M. Gupta, and J. Moreira, "Adaptive Incremental Checkpointing for Massively Parallel Systems," Proc.Int'l Conf. Supercomputing, pp. 277-286, 2004.
[5] R. Garg, V. Garg, and Y. Sabharwal, "Scalable Algorithms for Global Snapshots in Distributed Systems," Proc. 20th Ann. Conf.Supercomputing, pp. 269-277, Nov. 2006.
[6]R.Gallager, P. Humblet, and P. Spira, "A distributed algorithm for minimum weight spanning tree,". *ACM Transactions on Programming Languages and Systems*, January 1983.
[7] F.Chin and H.F. Ting "An almost linear time and $O(n \log n + e)$ messages distributed algorithm for minimum-weight spanning trees" , In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science (FOCS)*.
[8] E. Gafni, "Improvements in the time complexity of two message-optimal electionalgorithm" , In *Proceedings of the 4th Symposium on Principles of Distributed Computing (PODC)*, 1985.
[9] B. Awerbuch." Optimal distributed algorithms for minimum weight spanningtree, counting, leader election, and related problems" , In *Proceedings of the 19thACM Symposium on Theory of Computing (STOC)*, May 1987..
[10] M. Elkin, "Unconditional lower bounds on the time-approximation tradeoff fort he distributed minimum spanning tree problem" , In *Proceedings of the ACMSymposium on Theory of Computing (STOC)*,June 2004.
[11] J.Cheriyan, M. Kao, and R. Thurimella, " Scan-first search and sparse certi¯cates:an improved parallel algorithm for *k*-connectivity" *SIAM Journal of Computing*, 1993.
[12] R.Thurimella, "Sub-linear distributed algorithms for sparse certificates and biconnected components", In *Proceedings of the Symposium onPrinciples of Distributed Computing (PODC)*, August 1995
[13] D. Peleg. *Distributed Computing, " A Locality-Sensitive Approach",* SIAM, 2000.