Oriented Consistency Algorithms for Virtual Organizations in Intensive Data Grids

Jesús Acosta-Elías[1] Enrique Stevens-Navarro[1] and Ulises Pineda-Rico[2]

[1] Facultad de Ciencias, Universidad Autonoma de
San Luis Potosí, San Luis Potosí, S.L.P., 78290, México,
Av. Salvador Nava s/n, Zona Universitaria.
Tel: +52 (444) 826 2316, Fax: +52(444) 826 2321
jacosta@uaslp.mx, estevens@galia.fc.uaslp.mx,
Ulises.Pineda@postgrad.manchester.ac.uk
[2] The University of Manchester, School of Electrical and Electronic Engineering.
Manchester, UNITED KINGDOM, M60 1QD.

Abstract. Replication can improve the reliability of a Computational Grid. Replication allows a resource to be used even if some of the instances are not running or are inaccessible. However the challenge is how maintain consistent the replicas. In this paper we propose and evaluate "Oriented Consistency" (OC), a weak consistency algorithm for the dissemination of changes considering application-level in computer Grids. The main aim of this algorithm is optimizing the propagation of changes introducing a preference for nodes in Virtual Organizations of the Grid. The algorithm has been simulated over ns-2, and evaluated with several simple and complex spatial distributions of the users. The impulse response of the algorithm has been characterized. We conclude that considering application parameters such as clustered users of Virtual Organizations in the event or change propagation mechanism gives a surprising improvement in the speed of change propagation perceived by most users. In other words, it satisfies the greatest number of users in the shortest amount of time.

1 Introduction

A growing number of Internet applications need to run on a changing and unreliable network environment with a very large number of clients. Grid based applications are examples of these, where roughly the same software is running at many or all nodes. Collaborative and other data grid intensive applications strongly depend on efficient access to reasonably up-to-date information that may be replicated and kept up to date since it is frequently subject to update messages. Replication is one way to provide service to clients with low delay response, high degree of availability and autonomy (independent of unexpected backbone delays or link failures), and good scalability[7]. This work is based on a model of service composed by a number of replicas running on hosts (nodes) at different locations. A replica is a host which provides exactly the same services

as the principal host. In this paper we will use the terms server and replica in the same sense. When changes in data are introduced, the distribution of changes or events to all nodes is required to keep all the replicas consistent, with the same content.

In replication with weak consistency each node from time to time chooses a neighbor to start an update session. In an update session two nodes mutually update their contents. At the end of the session both nodes will have the same content. In this paper it will be referred to simply as a "session". The usual metric principle to evaluate weak consistency algorithms is the amount of sessions necessary for a change brought about in a node to be propagated to all the others.

On the other hand, a Virtual Organization (VO) [5] is defined within a Computer Grid as a set of individual and/or institutions working on a collaborative basis to achieve a common goal. As an example, please refer to Fig.1. In this scenario, each network node provides service to a group of subscribers, and nodes are only required to know a few neighbor nodes (partial view). The oriented consistency (OC) algorithm gives priority to sessions with neighbors of the same VO. We have found considerable improvement with the exchange of very little additional signalling. This algorithm is validated by means of simulation over large grid systems.

To evaluate the performance of the algorithm presented in this paper, an OC and weak consistency algorithm[1] simulator has been constructed, over Network Simulator 2 (ns-2) [3]. To take into account the clustered clients(VO) at every node we use additional metrics: the rate of users satisfied with consistent information, an utility function (based on economic theory). We conclude that OC improves the distribution of changes by prioritizing nodes with greatest users of specific information. In other words, OC algorithm satisfies the greatest number of users in the shortest amount of time, while sending the same amount of messages (better value at the same cost).

At to date, exists works to applies the weak consistency algorithms for Data-intensive Grids [8], but they do not take into account the zones of the system with high number of subscribers to specific information(Virtual Organizations). The goal of this proposal is to send in priority way the changes to nodes with higher number of Subscribers for this information, see figure 1.

The rest of the paper is organized as follows: Section 2 describes our system model. Section 3 describes the "Oriented consistency" algorithm. In section 4 we explain the methodology of simulation of our algorithms in terms of network topology, workload and performance metrics. In section 5 we discuss the simulation results. The paper concludes in section 6.

2 System model

In data-intensive grids or a typical collaborative application, participants are clustered at different Virtual Organizations with one local server that knows

something (users in our case) about some neighbors but not all servers (partial knowledge), see Fig.1.

The model of our grid system consists of a number of N nodes (principals) that communicate only via message passing. We assume a fully replicated system, i.e., all nodes must have exactly the same content. Every node is a server that gives services to a number of local clients. Clients make requests to a server, and every request is a "read" operation, a "write" operation, or both. When a client invokes a "write" operation in a server, this operation (change) must be propagated to all servers (replicas) in order to guarantee its consistency. An update is a message that carries a "write" operation to the replica in other neighboring nodes.

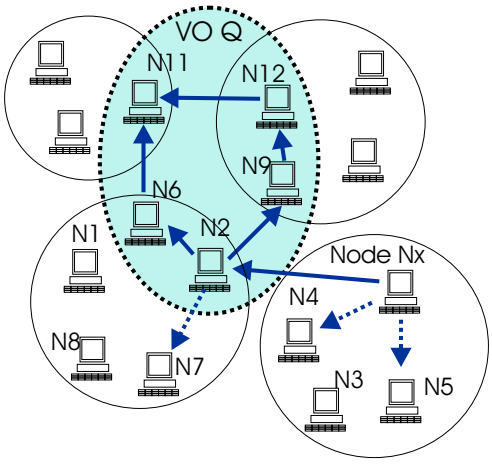

Fig. 1. In a grid the organizations can participate in one or more Virtual organizations. Each circle may correspond to an organization and the ellipse of dashed line represents a Virtual Organization(VO). The users in the VO may be sharing data and resources(ie CPU cycles, storage) for collaborate in the same projects. In this figure, is showed four organizations and one Virtual organization Q. In the node Nx a new message is generated(an object of VO Q was modified). In order to maintain the replicas in consistent state, this node sent the message(update) in priority way(Arrow with solid line) to nodes of VO Q and in normal priority(Arrow with dashed line) to the rest of the neighbor nodes.

The OC algorithm can be implemented as a daemon that interacts with a local server application(see fig.2). The daemon is responsible for propagating data, and thus maintaining consistent the external and local worlds. The daemon needs to be informed of the local interests (subscription to data of the VO), the value of local demand, and any new or changed data of external interest. It provides the local application with updates from the rest of the network. The internal state information consists on a vector of demand (local and neighbor

nodes), and data (state vector of data). At random time it selects with random neighbor peer nodes (with preference for nodes with greater demand) to exchange new data.

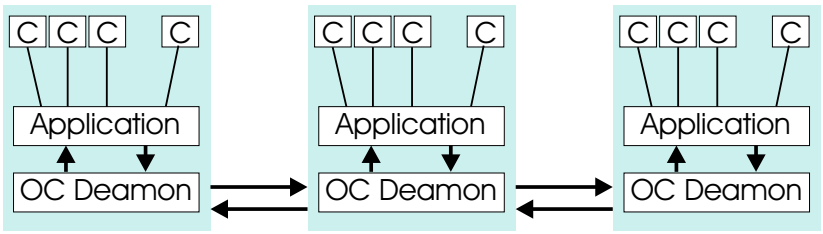

Fig. 2. Structure of a Grid application with OC replication

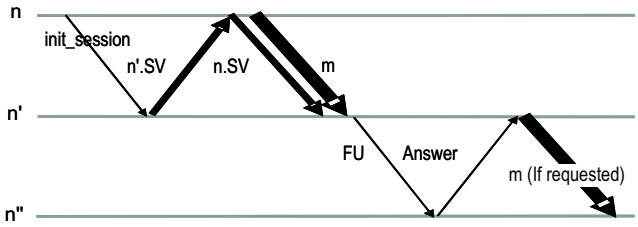

Fig. 3. An anti-entropy session followed by a fast-update notification to a node with higher demand.

3 The oriented consistency algorithm

The OC algorithm describes how changes propagate among network nodes with different value of demand(Users or subscribers).

The following section describes an extended Time-Stamped Anti-Entropy (TSAE) [2],[1],[6] weak consistency algorithm with a "fast update" step for faster propagation of changes to nodes of higher interest in this information.

3.1 The OC algorithm

In a Grid with an arbitrary large number of nodes N, every node n knows his interest of your users (demand) and the demand on t neighbors: $n.demand$ or d_n, and $n.neighbors[ni] = d0, d1, dn, dt,$. Demand at node $n : dn$, in our model and simulations it has been defined simply as the number of clients that

node n provides service. The value of t is typically in the range of $1..log(N)$. Every message m can be identified by a $MessageId$ and a Time-stamp: $m = m.id, m.tstmp, m.data$ Every node n has a summary vector of the history of messages it has received: $n.SV[]$.

Figure 3 describes the protocol with an example among three neighbor nodes, where n has a new message m and $n".demand > n'.demand > n.demand$, with all values of demand observed at the nodes at the same time. n randomly initiates a session of exchange of summary vectors with n', and n' immediately sends a fast update with $n"$ because it has greater demand.

The next text is the pseudo-code for the FC algorithm instantiated for fig. 3.

after n.wait(random time)
n'=n.select_neighbor(n.neigbors[ ],HIGHEST_DEMAND)
n.send_init_session(n')// message sent to n'
when n'.receive_init_session:
n'.send_SV(n,n'.SV[ ]) //SV: summary vector
when n.receive_SV:
n.send(n',n.SV[ ])
//Calculate diffV==messages n' not yet received
diffV=n.compareSV(n.SV[ ], n'.SV[ ])
foreach message m in diffV:
n.send(n',m)//send message m to n'
when n'.receive(n.SV[ ]):
diffV[ ]=n.compareSV(n'.SV[ ], n.SV[ ])
foreach message m **in** diffV[ ]:
n'.send(n, m)
when n'.receive(m):
n'.process(m) //and update n'.SV[ ]
foreach neighbor e **in** n'.neighbors[ ]
where n'.neighbors[e].demand ¿ n'.demand:
FU=new fast_update($m.id$, $m.tstmp$)
n'.send_fast_update(e,FU)
when $n"$.receive_fast_update(FU):
if $n"$.check_history(FU) == UNKNOWN):
$n"$.request(FU.id)
else $n"$.reject(FU.id)
when n'.receive(FU.id): //if requested
n'.send($n"$, m) //send message m to $n"$

4 Simulation methodology

To evaluate the performance of the Oriented consistency algorithm compared to the baseline TSAE algorithm [2], we simulate the behavior of the algorithms on large system Grid. In a simulation run each node originates a new message at $t = 0$. The simulation measures the number of sessions required to reach a

global consistent state: every node will receive the total number of messages in different number of sessions.

In the simulation we have discarded the influence of network performance (latency, bandwidth, congestion) because the time required to send a message from one node to another was assumed to be negligible compared to the time between anti-entropy sessions.

4.1 Demand Workload

The demand in our experiments corresponds with the number of clients who use a certain replica and have interest in a specific information. In certain applications the definition of what demand is can be different. For instance, in a publish/subscribe collaborative application, demand would correspond to the number of subscribers to a certain section. In other applications demand could be calculated from the history of past requests.

4.2 Performance Metric

The purpose of the "Oriented consistency" algorithm is to improve the performance of the weak consistency algorithms, with particular emphasis on increasing the speed with which these algorithms convey the changes to the zones of greatest demand(Virtual Organizations), so that a greater number of clients of specific information may have access to fresh content in a shorter period of time. It is for that reason that our experiments are centered on measuring these speeds.

At the beginning of a simulation, at $t = 0$, a new message is generated in every network node. The experiment ends when each node receives all the messages. The performance (speed) is measured in terms of the number of anti-entropy sessions needed for all the zones to receive the messages originated at every replica (node).

If the number of users in each node of the network is used as a measure of demand, then a node with a high number of users which reaches a consistent state will benefit the user community more than another node with a low number of users which reaches the same state at the same time. The availability of up-to-date information on a grid will be higher if high demand nodes have higher priority than low demand ones. Every simulation calculates the pair (d_i, c_i) for all nodes, where d_i is the demand at node i, and c_i is the time when node i has received all changes. This pair can be expressed by the $c(n_i, t)$ function (an impulse function of value d_i):

$$c(n_i, t) = \begin{cases} d_i : t = c_i \\ 0 \end{cases} \qquad C(t) = \sum_{i=0}^{N} c(n_i, t) \tag{1}$$

$C(t)$ is the sum of demand for all nodes that have reached a consistent state at a certain time t. In economic terms, we can define a utility function for each node

$u(n_i, t)$. It represents the value of demand satisfied with up-to-date information at time t (a step function of value d_i).

$$u(n_i, t) = \begin{cases} d_i : t \geq c_i \\ 0 \end{cases} \qquad U(t) = \sum_{i=0}^{N} u(n_i, t) \tag{2}$$

$U(t)$ is the sum of utility for all nodes that are consistent in time t. $U(t)$ expresses the satisfaction or benefit perceived by the community of users of our system. $U(t)$ roughly corresponds in economic terms with the Social Welfare function (SWF) defined in terms of global values as Benefit - Cost, given that the cost (total number of messages exchanged) does not change significantly. In time $t = 0$, all the nodes are in a non-consistent state, and as time passes more and more nodes will reach a consistent state and thus they will contribute to the SWF with their local demand d_i.

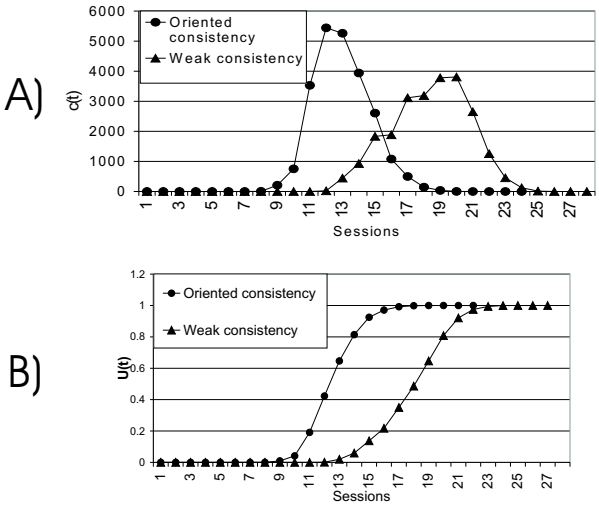

Fig. 4. $C(t)$ and $U(t)$ to show contributions of Oriented consistency vs weak consistency.

5 Simulation results

In this section, we evaluate the performance of the various parts of the algorithm on grid. Every experiment shows the mean value after more than 1000 executions to discard the effect of random choices. In figure 4A, its shown the behavior of $C(t)$ when the number of sessions is increased. The consistent state is achieved

sooner by the OC algorithm (less number of sessions) which imply that more users are able to access up to date information. On the other hand, in figure 4B, it is shown the behavior of the utility of all the nodes when the number of session is also increased. After 15 sessions the majority of the users in all the Grind are able to use up to date information.

6 Conclusions

In this paper, we study the problem of propagating changes on replicated data over a Grid using our Oriented Consistency (OC) algorithm. The principal aim of OC is the propagation of changes with preference for nodes and zones of the Grid which have the greatest demand of specific information. We evaluated the performance of the algorithm by simulation over a Grid. The results are dependent on the diameter of the network and rather independent on the number of nodes or the distribution of demand. In addition, it only requires local information, and there is no need for global coordination. Finally we conclude that our "Oriented consistency" algorithm optimizes the distribution of changes by prioritizing the nodes with greatest demand. In other words, it satisfies the greatest demand in the shortest amount of time.

References

1. Atul Adya, Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions, PhD thesis M.I.T., Department of Electrical Engineering and Computer Science, March 1999.
2. R. A. Golding, "Weak-Consistency Group Communication and Membership", PhD thesis, University of California, Santa Cruz, Computer and Information Sciences Technical Report UCSC-CRL-92-52, December 1992.
3. The Network Simulator: http://www.isi.edu/nsnam/ns/
4. V. Duvvuri, P. Shenoy and R. Tewari, "Adaptative Leases: A Strong Consistency Mechanism for the World Wide Web", IEEE INFOCOM 2000, pages 834-843.
5. I. Foster, C. Kesselman, S. Tuecke. International J. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. Supercomputer Applications, 15(3), 2001.
6. Kwok M, Wong JW, Scalability Analysis of the Hierarchical Architecture for Distributed Virtual Environments, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS Volume: 19 Issue: 3 Pages: 408-417 Published: MAR 2008
7. B. Clifford Neuman, "Scale in Distributed Systems. In Readings in Dist. Comp. Syst.", IEEE Computer Society Press, 1994
8. Changqin Huang, Fuyin Xu and Xiaoyong Hu, Massive Data Oriented Replication Algorithms for Consistency Maintenance in Data Grids, ICCS 2006.