A Flexible Hybrid Concurrency Control Model for Collaborative Applications in Large Scale Settings

Guillaume Pierre
Guillaume.Pierre@inria.fr

Mesaac Makpangou
Mesaac.Makpangou@inria.fr

INRIA Projet SOR - BP 105 - 78153 Le Chesnay Cedex - France

Abstract

Large-scale collaborative applications are difficult to build because of their high concurrency control needs and the heterogeneity of the underlying architecture. Due to these difficulties, only a few large-scale applications have been developed, such as Usenet or irc. To facilitate the realisation of such applications, we propose a more precise definition of the application's needs, in order to provide a good "quality" of cooperation when it is needed, and cheaper cooperation when it is acceptable. The model of LaSCoW (Large Scale Collaborative Work) allows the applications to be partitioned into separate consistency domains, each domain implementing its own collaboration policy.

1 Introduction

The success of the World-Wide Web has boosted the demands concerning world wide collaboration. In particular, collaborative applications are expected to become commonplace in the future.

We consider a collaborative application to be any application allowing multiple participants (users or computer programs) to access a shared document concurrently. Examples of applications include collaborative engineering (e.g. a group of architects cooperating to design a building), games involving several participants (e.g. a chess game on the Internet with players, spectators and commentators) and collaborative editing of a multimedia document (e.g. a magazine with reporters, correspondents, consultants, and assistants located world wide). As illustrated by these examples, a shared document can be of multiple nature: a text, a building plan, or even a chess board.

Due to the characteristics of large-scale settings (i.e. frequent disconnections or crashes, network partitions and high latency), it is very attractive for availability reasons to replicate documents that are shared by participants located world-wide. The replication raises however one difficult issue: how to maintain efficiently the consistency of replicas, in presence of concurrent updates ? The main problem is to find a good tradeoff between the cooperation policy (especially the concurrency control policy) to be enforced and the quality of the service (i.e. performance and availability) to be guaranteed to the users.

Participants to a large-scale collaborative application are aware of technical matters such as network bandwidth, throughput, and latency. They are also aware of the impact of technical limitations on the quality of service, and of the necessity to make a tradeoff between the concurrency control policy and the quality of service. The quality of service that each participant is prepared to accept mainly depends on the characteristics of connections with its peers and on his role in the collaborative session. As all participants do not play the same role and are not located on a uniform network, they have different demands. For example, one can expect a very good quality of cooperation with a user located on the same LAN, but can accept a worse quality for cooperation with a WAN-connected partner.

The diversity of demands makes it difficult, or even impossible, to define one satisfying uniform cooperation policy. We propose a cooperation model based on an hybrid concurrency control which permits to

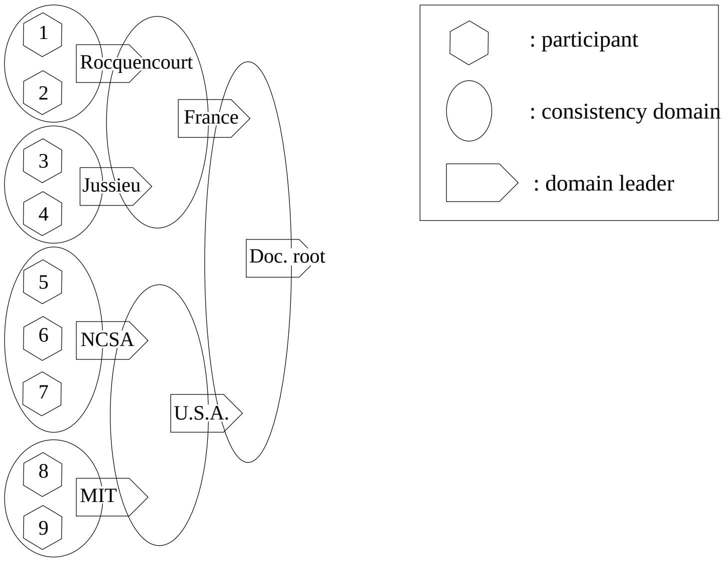

Figure 1: An example of consistency domain hierarchy

cope efficiently with the diversity of participants' demands regarding the quality of service. For example, in a collaborative text editing application, it is often possible to find groups of users which interact closely and actually need a good cooperation scheme, but with fewer interaction between groups. It is not useful, and often not possible, to provide to the entire application the cooperation policy needed by each group individually. Specifying which groups need which policy allows us to save significantly on resources. With respect to a given participant, partners should be classified in separate groups corresponding to different qualities of service. Each class of partners could then be associated to the appropriate cooperation policy.

Section 2 discusses this model. Section 3 points out the plan for future work. Then, Section 4 compares this proposal with existing collaborative editing support systems and Section 5 draws some conclusions.

2 A hybrid concurrency control model

To cope with the varying demands of the applications and the environment, LaSCoW (Large Scale Cooperative Work) partitions participants of a collaborative editing session into separate *consistency domains*.

A consistency domain is a group of participants who agree on one concurrency control policy concerning their own accesses to the documents. Each consistency domain has a leader which represents this domain to other domains. A leader cooperates with its peers to propagate the local modifications to the outside world and also to update its domain's view of the shared document. It can be viewed as a gateway between its consistency domain and the rest of the world.

Leaders can be grouped together in another, separate, consistency domain. One can iterate this process any number of times. We end up with a consistency domain hierarchy with the root domain containing the top-level leaders.

Figure 1 shows an example of a consistency domain hierarchy designed to cope with the heterogeneity of the network. Each participant cooperates using a strong concurrency control protocol with participants located on their LAN, an intermediate protocol with participants located in their country, and a weaker protocol with other participants.

The example represents a collaborative editing session which involves two participants at Rocquencourt, two at Jussieu, three at NCSA and two at MIT. The root consistency domain represents the world and has two members: the French and American leaders. Within this domain, one will likely enforce an optimistic concurrency control policy. Such a choice conforms to the characteristics of international connections (high latency, high frequency of faults such as messages loss and network partitions).

The second level represents countries. The France consistency domain contains the leaders of Rocquencourt and Jussieu; the USA consistency domain contains the MIT and NCSA leaders. At this level, one can enforce a coarse grain locking: each campus will lock a large portion of the document (e.g. the sections on which its participants are working).

Finally, the Rocquencourt, Jussieu, NCSA and MIT consistency domains are composed of participants located in the same campus. Members of these domains are likely connected by high speed networks. The appropriate concurrency control policy here would be fine grain locking.

The consistency domain model raises many questions that must be solved. Section 2.1 discusses different partitioning policies and Section 2.2 discusses the dynamic reconfiguration of the domains structure.

2.1 Partitioning criterion

An important issue regarding the definition of consistency domains is to be addressed: how do we determine the partitioning criterion and the appropriate policy to enforce within each domain? For the moment, we believe that there is no such thing as a universal solution that is appropriate for all collaborative editing sessions. Rather, this will depend on the nature of the application, the number and the distribution of participants in a session, the supporting network characteristics, and the tradeoffs that participants are prepared to accept at a given period in time. The optimal criterion could even change due to application, participants or network evolution.

The example given in Section 2 shows a possible construction based on network characteristics. The structure has been built in order to make the network quality between all the members of a consistency domain homogeneous. Furthermore, connections within a domain must be significantly better than those of higher-level domains. The example is based on the assumption that the cost of distributed collaborative editing applications is essentially due to synchronous lock/unlock requests and update propagation. Such applications usually spend a lot of time waiting for these kinds of acknowledgments. Hence the reduction of this traffic, especially for slow connections, will yield better performance.

Another possible construction could be based on the access patterns of the participants. The rationale of such a decomposition is that the way people interact with one another greatly depends on their roles: some groups of users could choose to suffer poor performance in order to get a closer cooperation (e.g. a strong consistency model instead of a weak one). In an application allowing participants to edit a magazine collaboratively, one could create a consistency domain per section (e.g. sports, fashion, politics, and economics). For example, the sports domain is likely to constitute a few journalists at the magazine offices coordinating reports from correspondents, covering different events. Within this domain, reporters abroad do not conflict with one another. Coarse grain locking could satisfy participants of this domain. On the other hand, the economics domain may be composed of a few experts located at the magazine offices. Economists need to interact closely to confront the different analyses. To permit this close interaction between participants within the economics domain, a fine grain concurrency is preferable, even if a poor network performance slows the application down.

One advantage of the second partitioning criterion compared to the first is that it reduces the concurrency between domain leaders, since participants in different domains are likely to access different sections of the document. However, if most participants have the same role, then the first partitioning will yield better performance than the second. We are contemplating the possibility of LaSCoW providing a formalism that permits the specification of the partitioning criterion and the policy associated with each domain. Alternatively, we are considering the definition of a default policy based on locality and network characteristics.

2.2 Dynamic reconfiguration

In order to be really flexible, LaSCoW must be able to deal with the changes on the quality of service of the underlying environment, or concerning the activity of the participants. LaSCoW should be able to adapt dynamically the domain hierarchy accordingly to these changes.

The adaptation should be transparent to the users. For example, if a journalist working on one section of a magazine is to be moved to another, this operation should not require that the entire application be restarted, thereby disturbing all the other users.

More precisely, LaSCoW will provide two elementary features:

- *The replacement of the replication protocol* inside a consistency domain. This feature mainly deals with a global synchronisation inside the domain concerned. Such an operation should not impact the other domains, so the cost of such a change should be limited.

- *The movement of participants* from one consistency domain to another. This feature seems much more difficult to achieve because the source and destination domains are not supposed to have a totally coherent view of the document. If a user is to be moved from a domain to another, this implies a synchronisation of both domains, certainly through a chain of third-party domains. In the example of section 2, if the user "1" needs to be moved from the "Rocquencourt" to the "Jussieu" domain, the synchronisation will certainly concern "Rocquencourt", "France" and "Jussieu" domains. That means that the cost of such a change is likely to be much higher than the previous one.

One related problem is who decides on the appropriate time for adapting the domain hierarchy and where he finds the necessary informations for taking the right decision. On simple answer to this question is to rely on an administrator who is in charge of the application's configuration. When this administrator observes any change on the expected quality of service, or when any change occurs in the participants' activity or location, he will interact with LaSCoW for adapting the domain hierarchy accordingly.

An alternative answer is for LaSCoW to monitor the application behavior and to react essentially to the changes on the quality of service of the underlying environment. It seems reasonable that each domain leader be responsible for using simple rules like those described in [8] in order to choose the optimal protocol inside its domain. But a global view of the situation is indispensable for being able to take overall decisions. We are considering making each domain leader responsible for all of the hierarchy dependent on it. So the root domain leader would be in charge of taking "strategic" decisions whereas lower level domains leaders would take "tactical" decisions.

3 Current and Future Work

Our current work is to evaluate the benefits that applications can gain from the use of our model. We expect to show that most large-scale applications will benefit from the LaSCoW cooperation model in order to handle appropriately the diversity of user requirements and the heterogeneity of the architecture.

Our approach consists of evaluating the quality of service offered to the participants that share some replicated objects, depending on the underlying network characteristics, the replication protocol (i.e. the concurrency control policy and the propagation strategy) and the participants access pattern. We consider only objects that can be effectively shared by participants located world-wide, for example a structured text document. The participants' access patterns are derived from different traces of real executions of existing applications. These simulations will help to identify interesting partitioning criteria and the appropriate policies to enforce in order to provide a given quality of service to the application.

Following the simulation phase (and provided that the results of these simulations conform to our intuition), we plan to develop LaSCoW. The short-term objectives will be a running LaSCoW platform with a library of replication protocols and dynamic reconfiguration features. No help for configuration is provided in this prototype. We will then add tools for monitoring the network infrastructure and the users access pattern, and for analysing consistency domains configurations. Lastly we plan to add a module for automatic (re)configuration.

4 Related work

Our work on LaSCoW deals with many different domains: large-scale collaborative applications, flexible systems, protocols evaluation, etc. We will only address the first two aspects in this section.

4.1 Large-scale cooperative applications

Most existing large-scale collaborative applications (e.g. Usenet [6], irc [9], Web caches [3]) avoid the concurrency control problem by trading increased efficiency against a reduced level of consistency management. Such a trade is made possible because of the nature of these applications. Usenet and irc only allow users to add information to the shared data (adding articles in Usenet or messages in irc), so the only possible inconsistency is a misordering of messages, which has been chosen to be ignored. Web caches allow the modification of documents, but don't try to manage update concurrency (only the owner of a Web page is able to modify it). These characteristics allow a very light cooperation between caches and servers, without any world-wide concurrency control. The aim of LaSCoW is to allow the development of large-scale application which can't afford such a uniform tradeoff, and need a real complex concurrency control.

The World-Wide Web is an attractive platform for building more complex large-scale applications because of its very large diffusion and uniform access to resources. Some projects aim at building CSCW (Computer Supported Collaborative Work) applications on top of the Web but its communication model limits the possibilities of such systems: the systems using CGI scripts are necessarily centralised on the server. These extensions are most often shared workspaces [2] or annotation systems, neither of which provides concurrency control. Applets might be an interesting basic feature for large-scale collaborative editing because they enable code migration and client-side computing. Some interesting projects have experimented with this on top of the HotJava browser [12], but they are only experimental applications and could not handle a realistically complex collaborative application. Although the applet technology doesn't seem mature enough for handling the LaSCoW infrastructure, we expect that future developments will let us integrate LaSCoW in the World-Wide Web environment.

4.2 Flexible systems

A number of collaborative systems have been developed during the last ten years. These are in fact complete applications: collaborative editors, chatting applications, etc. They are in general specific programs, each of which solves a particular problem (e.g. GroupDesign [1], Alliance [4]).

Toolkits have also been proposed recently. They provide essentially some basic features to allow programmers to build exactly the system that suits their needs. For example, Bayou [11] lets programmers insert application-specific code into the provided generic optimistic concurrency control. DistEdit [7] allows any text editor to be ported into CSCW by defining generic text editing primitives.

These systems and toolkits provide only one concurrency control policy. This choice limits each of them to a few specific well-suited uses. Some projects such as GroupKit [10] address this limitation by providing a library of consistency protocols among which programmers can choose the one most suited. This run-time choice make these systems much more flexible. But what makes them unsuitable for large-scale environments is the fact that they don't take into account the differences between the participants in order to give the best possible service to each of them.

As far as we know, the only work which has already tried to run several replication protocols at the same time are only intended to provide fault-tolerance via embedded transactions [5]. The approach is to decompose a transaction providing one-copy serialisability into a group of one-copy serialisability sub-transactions. LaSCoW's model extends this paradigm to all kinds of consistency criteria and to non-transactional computation models.

Adaptive systems are able to react to environmental modifications in order to optimise their behavior. The only adaptive replication model we know is [8]. The originality of this work is that during the execution with any concurrency protocol, the system is able to compute the cost that other protocols would introduce if they were used instead of the current one. This is done by using simple protocols whose cost can be deterministically computed from the users' access pattern. LaSCoW intends to integrate protocols which don't fit to these requirements, but it will certainly use similar algorithms inside its automatic adaptation modules.

5 Conclusion

This paper presented a hybrid concurrency control model. This model aims at offering to each participant of a collaborative application a quality of service that matches his needs and that can be obtained at reasonable cost.

Though a number of important issues are still open (e.g. how to determine a partitioning criterion, how to capture the application semantics and how to reconfigure dynamically the domain hierarchy), we believe that this model permits efficient implementation of a number of large-scale collaborative applications. Simulations are being carried out to comfort, or infirm, this intuition.

References

[1] Beaudouin-Lafont, M., and Karsenty, A. Transparency and awareness in a real-time groupware system. In *Proceedings of the ACM Symposium on User Interface Software and Technology UIST'92* (Nov. 1992). ftp://ftp.cs.unc.edu/pub/users/jeffay/papers/CSCW-92.ps.Z.

[2] Bentley, R., Horstmann, T., Sikkel, K., and Trevor, J. Supporting collaborative information sharing with the World-Wide Web : the BSCW shared workspace system. In *Proceedings of the 4th international WWW conference* (1995). http://orgwis.gmd.de/˜bscw/papers/boston-95/BOSTON.ps.

[3] Chankhunthod, A., Danzig, P. B., Neerdaelsand, C., Schwartz, M. F., and Worrell, K. J. A hierarchical internet object cache. Tech. Rep. 95-611, Computer Science Department, University of Southern California, Mar. 1995.
ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/HarvestCache.ps.Z.

[4] Decouchant, D., Quint, V., and Salcedo, M. R. Structured cooperative editing and group awareness. In *Proceedings of the HCI International'95, 6th International Conference on Human-Computer Interaction* (July 1995). ftp://ftp.imag.fr/pub/OPERA/doc/HCI95_alliance.ps.gz.

[5] El Abbadi, A., and Toueg, S. The group paradigm for concurrency control protocols. *IEEE transactions on knowledge and data engineering 1*, 3 (Sept. 1989), 376–386.

[6] Kantor, B., and Lapsley, P. Network News Transfer Protocol: a proposed standard for the stream-based transmission on news. RFC 977, Feb. 1986. ftp://ftp.inria.fr/rfc/rfc9xx/rfc977.Z.

[7] Knister, M. J., and Prakash, A. DistEdit : a distributed toolkit for supporting multiple group editors. In *Proceedings of the CSCW conference* (Oct. 1990), pp. 343–355.

[8] Le Sergent, T., and Matthews, D. C. J. Adaptive selection of protocols for strict coherency in distributed shared memory. Tech. Rep. ECS-LFCS-94-306, LFCS, University of Edinburgh, Sept. 1994. http://www.dcs.ed.ac.uk/publications/lfcsreps/EXPORT/94/ECS-LFCS-94-306/index.html.

[9] Oikarinen, J., and Reed, D. Internet Relay Chat Protocol. RFC 1459, May 1993.
ftp://ftp.inria.fr/rfc/rfc14xx/rfc1459.Z.

[10] Roseman, M., and Greenberg, S. GroupKit : a groupware toolkit for building real-time conferencing applications. In *Proceedings of the CSCW conference* (Nov. 1992), pp. 43–50.
http://www.cpsc.ucalgary.ca/projects/grouplab/papers/gk_cscw92.ps.

[11] Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J., and Hauser, C. H. Managing update conflicts in Bayou, a weakly connected replicated storage system. In *Proceedings of the 15th ACM symposium on Operating Systems Principles* (Dec. 1995). http://www.parc.xerox.com/csl/projects/bayou/pubs/sosp-95/BayouConflictsSOSPPreprint.ps.gz.

[12] Walther, M. Supporting development of synchronous collaboration tools on the web with GroCo. In *Proceedings of the CSCW and the Web ERCIM workshop* (Jan. 1996). http://www11.informatik.tu-muenchen.de/~walther/groco/gmdabstract.html.