Replica Divergence Control Protocol in Weakly Connected Environment

Ahmed Jebali, Mesaac Makpangou

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Replica Divergence Control Protocol in Weakly Connected Environment

Ahmed Jebali — Mesaac Makpangou

N° 4217

Juin 2001

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4217--FR+ENG

Replica Divergence Control Protocol in Weakly Connected Environment

Ahmed Jebali [*] , Mesaac Makpangou [†]

Thème 1 — Réseaux et systèmes
Projet SOR

Rapport de recherche n° 4217 — Juin 2001 — 15 pages

Abstract: This paper presents a consistency control protocol destined for groups of cooperative processes located world-wide. We assume that processes cooperate by sharing a replicated object. We consider processes with well-known profiles; however the actual behaviour of each process could diverge from the profile. The protocol aims at offering a good quality of service to cooperative processes, while guaranteeing replicated object consistency in the presence of disconnections and partitions. We are interested in the bounded consistency condition: the divergence between each replica and the "real object" remains smaller than a fixed bound during the lifetime of the application. We decompose this consistency condition into a set of local criteria concerning the effects of unexpected operations initiated by each process. We then associate to each process a consistency manager that enforces this process local consistency criterium. To experiment with this protocol, we developed a highly available allocation service relying on it. The experimental results confirm that this approach provides a high degree of autonomy to each replica, especially when the actual behaviour of that replica conforms to the profile.

Key-words: Distributed System, Network Service, Replication Protocol, Divergence Control, Bounded Consistency

[*] Ahmed.Jebali@inria.fr
[†] Mesaac.Makpangou@inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

Protocole de Contrôle de Divergence dans les Environnements Faiblement Connectés

Résumé : Nous présentons dans ce papier un protocole de cohérence destiné à des groupes de processus coopérants répartis à large échelle. Nous admettons que les processus coopèrent en partageant un objet répliqué. Nous considérons des processus avec des profils d'accès bien définis ; toutefois le comportement réel de chaque processus peut diverger par rapport à son profil. Le protocole vise à offrir aux processus coopérants une bonne qualité de service, tout en garantissant la cohérence de l'objet répliqué en présence des déconnexions et des partitions réseaux. Nous nous interessons à la condition de divergence bornée : la divergence entre les réplicas et le «réplica réel» reste inférieure à une borne fixée. Nous décomposons ce critère de cohérence en un ensemble de critères locaux concernant les effets des opérations imprévues initiées par chaque processus. Nous associons ensuite à chaque processus un gestionnaire de cohérence qui assure le respect du critère local de cohérence de ce processus. Pour experimenter ce protocole, nous l'avons utilisé pour développer un service d'allocation hautement disponible. Les résultas d'expérimentations comfirment que notre approche permet d'avoir un degré d'autonomie élevé pour chaque réplica, surtout quand le comportement du réplica est conforme à son profil.

Mots-clés : Systèmes distribués, service distribué, protocole de réplication, contrôle de divergence, divergence bornée

1 Introduction

Thanks to the success of the World-Wide Web, a wide range of large-scale network applications are deployed on the Internet. Examples concern various application domains, mainly selling, resources allocation and service supply. With the success of these applications and the rapid deployment of portable machines and PDAs, an increasing number of users want certain network applications to remain continuously accessible. To achieve such a level of availability, applications are replicated on a large-scale.

Due to the characteristics of large-scale networks, one could not enforce strong consistency guarantees while still offering a satisfying level of responsiveness to users. Fortunately, a number of large-scale cooperative applications can continue to behave correctly even if the replicated data they use are inconsistent. However the divergence between replicas should remain within a certain limit. Recent studies [ABM90, PL91, YV00b] show that one could exploit this tolerance to device efficient replication protocol that enforce bounded consistency conditions.

However, none of the existing bounded consistency protocols addresses the issue when replicas can be disconnected. For that, one needs a way to determine without communicating with partners operations that they have initiated. For applications with a sufficient historic, data analysis and data mining techniques [Vap98] can be used to extract an application profile; such a profile could then serve to approximate each replica's behaviour.

This paper presents a bounded divergence control protocol that benefits from the availability of the application profile to augment the autonomy of each replica. Roughly, we decompose the global consistency criterium in a group of local criteria that can be evaluated by replicas. The evaluation requires only local information. Synchronizations are however necessary when the actual behaviour of the application diverges considerably from the predicted profile.

The rest of the paper is organized as follows. Section 2 presents the replication model. Section 3 presents how to decompose the global consistency condition in local criteria. In section 4 we give the algorithm performed at each site, guidelines to configure the protocol and its implementation. Section 5 presents an example of a cooperative application that relies on the proposed protocol and discusses the conclusions that we draw from performance evaluation of this example. Section 6 compares our proposal with related work. Finally, section 7 presents our perspectives and draws some conclusions.

2 Replication Model

An application is a collection of cooperative processes $(p_1, \ldots, p_n)$ located respectively on sites $(S_1, \ldots, S_n)$. These processes cooperate through a shared replicated object. There is one object replica on each site. Each replica of the shared object is associated with a consistency manager that controls operations that are performed on that replica. An *operation* is an event chain that leads to a particular access to the shared object.

We assume the cooperative replication model defined by Georges Brun-Cottan [BC98]. Each process accesses the shared replicated object by initiating operations, thanks to the local consistency manager. Each operation initiated by one process is eventually propagated to all consistency managers attached to replicas of the shared object. Each manager decides the order on which its associated replica of the shared object performs operations initiated both locally and by remote process. We note $I_i(t',t) = (op_1, op_2, \ldots, op_k)$ the ordered sequence of operations initiated by process p_i during the time interval $[t',t]$. We simply note $I_i(t)$ if t' is the start of the applications. With respect to a process $p_j, j \neq i$, the history of p_i is composed of two parts:

- $I^n_{i,j}$ the sequence of operations initiated by p_i and already seen on site S_j.
- $I^l_{i,j}$ the sequence of operations initiated by p_i yet to be propagated to the consistency manager on site S_j or yet to be acknowledged by that consistency manager.

Using a concatenation operator $\oplus$[1] on histories, we have the following equality: $I_i(t) = I^n_{i,j}(t') \oplus I^l_{i,j}(t',t)$. Note that the model doesn't specify the way operations initiated on one site happen to be seen by the consistency manager located on a different site, nor the acknowledgement mechanism. One can use the anti-entropy protocol [PST+97], gossip messages [LLSG92], or any reliable propagation protocol.

The primary role of a consistency manager is to make sure that new operations can be initiated by the local process without compromising application consistency. For that, each consistency manager is parameterized with an applicative consistency criterium that is evaluated when the local process wants to initiate a new operation. If the evaluation succeeds, the requested operation is accepted and can therefore be scheduled on this site regardless of the activity on other sites. Otherwise, a synchronization with all or a subset of other consistency managers is required. The synchronization allows consistency managers to reduce the divergence between their local views of the application's state and the real one (that is the set of operations initiated so far by the group of processes composing the application).

3 Bounding Replica Divergence

We attach to each operation *op* a measure $\mu(op)$ that represents the impact of its application on the shared object. For example, one could attach to operation $get(amount)$ on a stock $(-amount)$ as its measure since its impact is to decrement the stock by the value *amount*. We define the measure of a history as the sum of measures of operations of this history. For a given history $\mathcal{H} = (op_1, op_2, \ldots, op_k)$ the measure is: $\mathcal{M}(\mathcal{H}) = \mu(op_1) + \mu(op_2) + \ldots + \mu(op_k)$. Note that the measures of histories composed of the same operations are the same, regardless the ordering of these operations within these histories.

[1] The $\oplus$ operator concatenates histories. $h_1 \oplus h_2$ is the history formed by operations of h_1 followed by operations of h_2.

We assume that one can predict the sequence of operations that each process is likely to initiate during the application's lifetime. We assume that the anticipated behaviour could be characterized by a profile function. We define a profile function of process p_i, denoted $Prof_i$, to be a function from the set of time intervals into the set of execution histories that maps to each time interval the history of operations that are expected from process p_i during that time interval.

The idea is to allow consistency managers to use profile functions to approximate remote process histories, without having to communicate with them, while still guaranteeing application consistency. Since prediction is not always accurate, these approximations could diverge from the actual behaviours of processes. The more accurate the prediction provided by the profile functions, the smaller that divergence. Our goal is to provide consistency managers with a way to bound this divergence to a value B.

Consider the tuple RH that represents the real histories that actually occurred on processes $(p_1, \ldots, p_n)$;

$$RH(t) = (I_1(t), I_2(t), \ldots I_n(t))$$

A consistency manager on site S_i approximates this global state with EH_i as follows:

$$EH_i(t) = (I_1(T_i[1]) \oplus \mathcal{PV}_1(T_i[1], t), \ldots, I_i(t), \ldots,$$

$$I_n(T_i[n]) \oplus \mathcal{PV}_n(T_i[n], t))$$

where $T_i[j]_{1 \leq j \leq n}$ is the occurrence date of the last operation initiated by process p_j and already seen by p_i; $\mathcal{PV}_j$ is the profile function of process p_j.

We define the divergence (d_i) between RH and EH_i as follows:
$d_i = |\mathcal{M}(I_1(t), I_2(t), \ldots I_n(t)) - \mathcal{M}(I_1(T_i[1]) \oplus \mathcal{PV}_1(T_i[1], t), \ldots, I_i(t), \ldots, I_n(T_i[n]) \oplus \mathcal{PV}_n(T_i[n], t))|$
rewritten to:

$$d_i = |\Sigma_{j \neq i}^{n} \mathcal{M}(I_j(t)) - \mathcal{M}(I_j(T_i[j]) \oplus \mathcal{PV}_j(T_i[j], t))|$$

Since $I_j(t) = I_j(T_i[j]) \oplus I_j(T_i[j], t)$, the divergence is:

$$d_i = |\Sigma_{j \neq i}^{n} \mathcal{M}(I_j(T_i[j], t) - \mathcal{M}(\mathcal{PV}_j(T_i[j], t))|$$

Now, consider the following definitions.

Definition 3.1 (Effective Effect) *The effective effect of process p_i on the shared object during the time interval $[t', t]$, noted $\mathcal{EE}_i(t', t)$ is the measure of operations initiated by p_i during that interval. $\mathcal{EE}_i(t', t) = \mathcal{M}(I_i(t', t))$.*

Definition 3.2 (Predicted Effect) *The predicted effect of process p_i during the time interval $[t', t]$, noted $\mathcal{PE}(t', t)$ of p_i, is the measure of operations predicted by the profile of p_i during that interval. $\mathcal{PE}_i(t', t) = \mathcal{M}(Prof_i(t', t))$.*

Given these definitions, d_i can be written as follows.

$$d_i = |\Sigma_{j \neq i}^{n} \mathcal{EE}_j(T_i[j], t) - \mathcal{PE}_j(T_i[j], t)|$$

Using the properties of absolute values, the present equation implies that

$$d_i \leq \Sigma_{j \neq i}^{n} |\mathcal{EE}_j(T_i[j], t) - \mathcal{PE}_j(T_i[j], t)|$$

Hence, to guarantee that

$$d_i < B \tag{1}$$

it suffices to find a tuple $(b_1, \ldots, b_n)$ such that

$$\forall j,\ b_j \geq 0 \ and \ \Sigma_{j=1}^{n} b_j \leq B \tag{2}$$

$$\forall j,\ |\mathcal{EE}_j(T_i[j], t) - \mathcal{PE}_j(T_i[j], t)| \leq b_j \tag{3}$$

Let's now assume that each consistency manager agrees with its peers on their view of operations initiated on its site; that is, $\forall 1 \leq i, j \leq n, V_j[i] = T_i[j]$ where $V_j[i]$ is the occurrence date of the last operation initiated by process p_j and already acknowledged by p_i. We derive from (2) and (3) the following sufficient condition for inequality (1):

$$\forall j,\ b_j \geq 0 \ and \ \Sigma_{j=1}^{n} b_j \leq B \tag{4}$$

$$\forall j,\ |\mathcal{EE}_j(V_j[i], t) - \mathcal{PE}_j(V_j[i], t)| \leq b_j \tag{5}$$

4 Divergence Control Protocol

4.1 Algorithm

The protocol bounds the divergence of the actual history of each process with respect to approximations that are made by other partners. We consider a tuple $(b_i, \ldots, b_n)$ such that $\Sigma_{j=1}^{n} b_j \leq B$. We assign to the consistency manager on site S_j the value b_j. When a process p_j wants to execute an operation op at date t, it informs its consistency manager, which in turn performs the following algorithm:

1. If for all i, the condition $|\mathcal{EE}_j(V_j[i], t) + \mu(op) - \mathcal{PE}_j(V_j[i], t)| \leq b_j$ holds then the operation is accepted; its measure is added to this process effective effect.
2. Otherwise, a synchronization is required. The local consistency manager sends to each partner k such that $|\mathcal{EE}_j(V_j[k], t) + \mu(op) - \mathcal{PE}_j(V_j[k], t)| > b_j$, operations initiated since the last operation it has acknowledged.

A process can also send notification at any time, this is not necessary to ensure the divergence condition, but increases system performance. When receiving a notification message from p_j, p_k updates the corresponding entry in its notification date vector T_k and executes notified operations on its own replica.

4.2 Protocol Configuration

To run the protocol, we first need to determinate the bound B. In essence, this bound is used to model the risks that many real-world applications are confronted with. More formally, we refer to the divergence bound as an error window tolerated by the application. Hence, the suitable value of bound B depends on the application metric and on the error window that is is considered as tolerable. Take the case of airlines reservation, overbooking flights is a common practice. This permits airlines companies to augment the occupancy of their flights. However, this practice has a cost, especially when companies are forced to refuse passengers that were booked. To make sure that this cost won't exceed some fixed budget, one could fix the limit for overbooked seats per flight. This overbooking limit is a good candidate for bound value B.

Once the bound parameter is found, we need now to determine a tuple $(b_1, \ldots, b_n)$ for which condition 4 holds. One simple solution is to partition B in n equal parts ($b_i = \frac{B}{n}$). Note that this partitioning does not have to remain unchanged along the time. One could at any time adapt the partitioning provided that the sum of all parts remains less or equal to B. For the airline reservation example, we affect to each agency a part of B proportional to the total number of requests directed to this agency, compared to the total number of requests. Furthermore, one could increase or decrease the bound B. An increase of B augments the degree of liberty of each cooperative server, while a decrease force them to synchronize more often.

4.3 Implementation

The divergence control protocol is implemented by a group of cooperative consistency managers. Each consistency manager exports an interface that enables local application processes, sharing the replicated data controlled by the group of managers, to declare operations that they want to perform. Each operation is a first class object, capable in particular to indicate its measure. For an application process to access the local replica of the shared data, it first creates the operation describing the intended action that it wants to perform on the shared object then it invokes the local consistency manager, passing the previously created operation; the consistency manger checks if this operation can be integrated within the local history without exceeding the bound. Once the operation can be performed, the local manger returns the control to the application process.

To ease the development, we rely on the Ensemble group communication platform [Hay98]. Managers use the *Cast* primitive of Ensemble to send notifications. Managers require reliable and totally ordered delivery guarantee. A notification includes the list of operations performed by the sender manager and not yet seen by other managers. When a manager receives a notification it adds these notified operations to its local history.

5 Experimentation and Performance Evaluation

To experiment our implementation prototype, we develop a simplistic version of a distributed electronic market space. Roughly, we consider the case of a distribute community of producers and consumers that cooperate thanks to the service offered by a distributed electronic market space. Producers register their offers and consumers request resources.

For the sake of simplicity, we make the folowing assumptions: (1) the number of sites where the electronic market is deployed is known and is equal to n; (2) all resources are equivalent, consequently the pool of resources can be represented by a numerical value corresponding to the number of resources available in the shared pool; (3) the behaviour of consumers and producers served by one site one site can be approximated with two linear functions: $cons_i(t) = f_{cons_i} \times t$ and $prod_i(t) = f_{prod_i} \times t$: f_{cons_i} is the average arrival frequency of consumption requests on process p_i and f_{prod_i} is the average arrival frequency of production requests on process p_i.[2]

The distributed electronic market space is an interesting example of cooperation that could benefit from the divergence control protocol. The central issue for each site here is to serve as many consumers as possible without causing the overall system to violate the consistency.

We developed a consistent distributed market space relying on the divergence control protocol as follows. Firstly, on each market site, we have one application process. Its role is to treat requests from the local consumer and producer. This process interacts with the consistency manager as described in Section 4.1. Secondly, on each market site, an observer process is running. It monitors the producer and the consumer profile. When a significant change is noticed, it initiates a procedure to update the current profile known to the group of consistency mangers.

We run this application to evaluate the impact of protocol parameters on the performance. We are especially interested on evaluating how the value of bound B and inaccuracy of the predicted profile impact the overall performances of the application.

5.1 Experiment Conditions

We run 10 allocators on 3 Pentium based PCs running Linux Redhat 6.2 interconnected with a local area network. We run a gossip daemon on each machine to optimize message exchange. Each allocator receives a total number of 10000 requests from each client (consumer and producer). The bound B is expressed in terms of resources; this corresponds to the maximum number of conflictual allocations that are tolerated. In this experiment, the observer acts as a configurator: it generates profiles and notifies all concerned partners(consumer, producer and consistency manager). For experiments where we assume the absence of prediction error, all partners will behave accordingly to the profile announced by the configurator. To simulate a prediction error, a consumer or a producer will deviate from what the observer has announced.

[2] Note that one could define a combined profile function $Prof_i$ as the sum of these two functions.

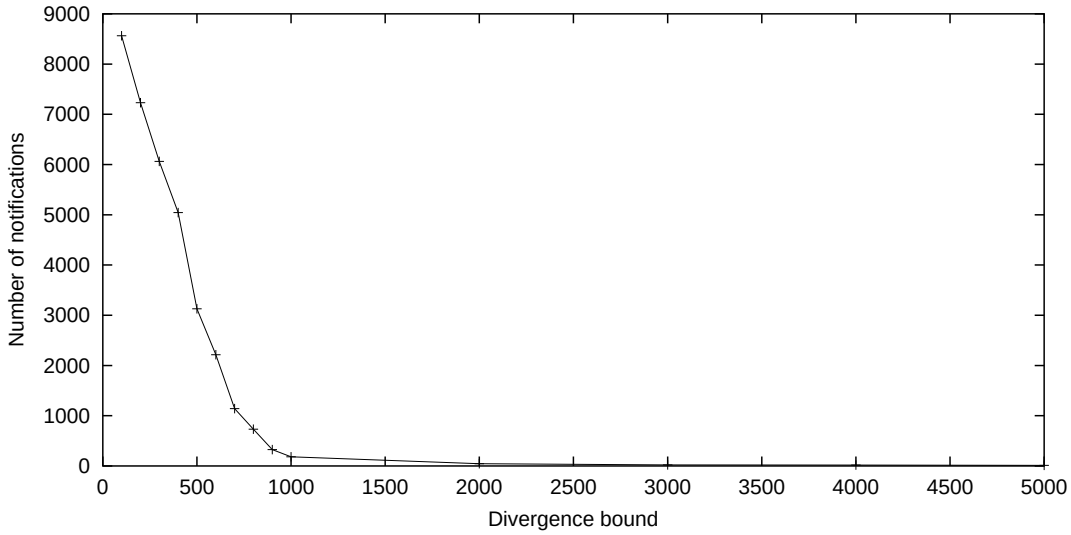

Figure 1: Network traffic in case 1

The metric we focus on are mainly the network traffic and the system throughput. The network traffic is the total number of notification messages exchanged between allocators and the application throughput is the total number of resources allocated. As the protocol relies on profiles prediction we anticipate that its performance is strongly related to the accuracy of such prediction. Thus we measured the performance of the protocol in three cases:

1. The perfect case: at each site the consumption and the production rate conform to the predicted profile;
2. Production miss case: consumers behave accordingly to their predicted profiles, but producers create less resources than predicted;
3. The general case: the real behavior doesn't conform to the prediction.

5.2 Experimental results

In the first experimentation case, all consistency managers are notified the profile of each allocator (that is the consumptions and productions that are expected from each site). Each consumer and producer conforms to its predicted profile, ie. allocation and production rates are conform to the profile. We plot in figure 1 for different values of the divergence bound B the total number of notification messages exchanged by allocators.

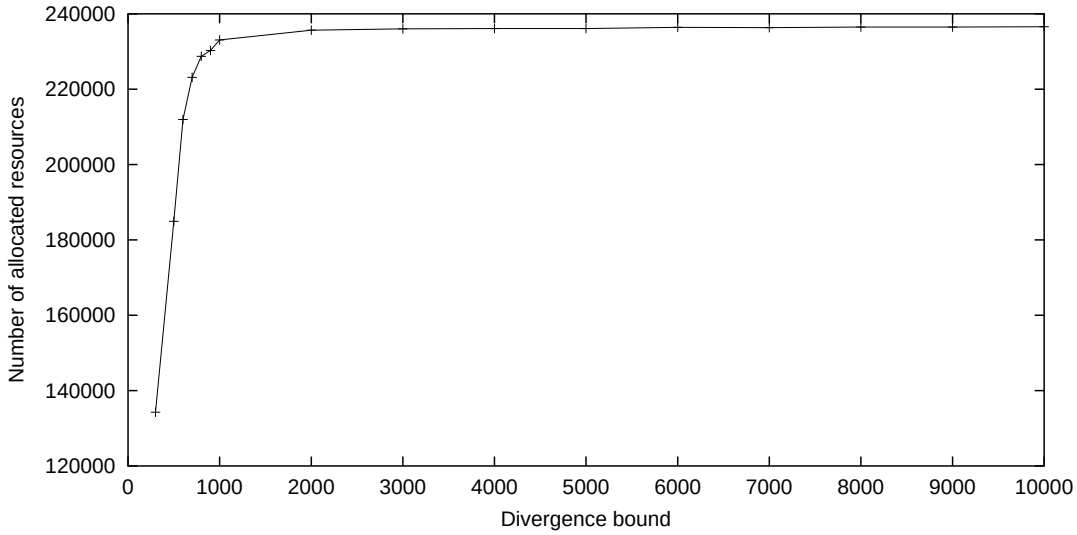

Figure 2: Application throughput in case 1

The figure shows that the network traffic is quasi null when the divergence exceeds some limit (here 5000). This indicates that, for a reasonable value of bound B, there is no need for allocators to communicate when each one behaves conformably to the profile declared to consistency managers. If however the bound is too low, the network traffic remains important, even if there is no prediction error. A detailed analysis of the case of low bound values reveals that approximatively 50% of allocation requests cause the effect to exceed the local bound b_i. We conclude from this observation that our protocol is more useful when the application can afford large bound values. When bounds are too low, the required guarantee is almost the strong consistency, hence this becomes as costly as the strong consistency protocol.

The figure 2 shows for the same configuration, the total number of allocated resources. We observe that there is a degradation of the system throughput for bound less than 5000. As seen in the previous paragraph more than the half of allocation requests cause a notification, furthermore this causes the request to be rejected because in our implementation allocators do not defer the request treatment after the synchronization procedure.

In the second experiment, each producer generates resources with a frequency $m\%$ less than its announced frequency. The value of m indicates the prediction error. We plot in figure 3 for various values of m the resulting network traffic.

First we note that the traffic decreases when the divergence bound increases. Therefore, for a fixed divergence bound the network traffic augments as the prediction error becomes higher. This appears clearly when divergence bounds are more restrictive ie. low. But in the other case the difference between network traffic for the different values of prediction

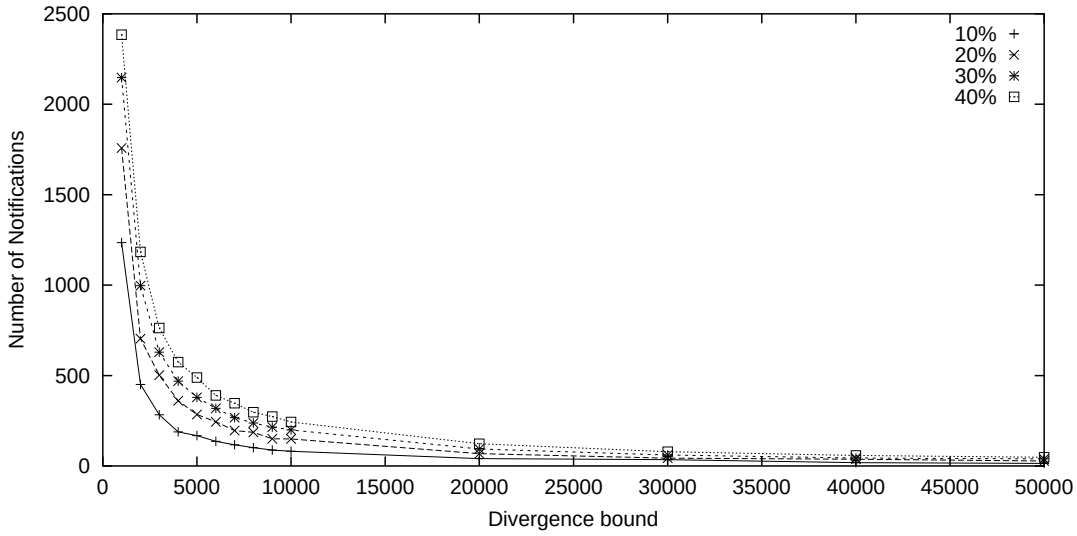

Figure 3: Network traffic in case 2

error are not excessive for high bounds. Since the effect due to the producers behaviour is annihilated within the permitted divergence. This graph could be used to parameterize the system in the following fashion: the system administrator fix the network maximum traffic to generate and evaluates periodically the miss percentage of producers, and then choose the divergence bound accordingly to the observed network traffic. A similar study can be performed for the consumer profiles to obtain a list of configuration that can be used to reconfigure the system.

The third case study considers the case where the consumer and the producer will not conform to their announced profiles. To simulate that, the observer process generates random profiles and uses them to reconfigure the consistency managers. Figure 4 draws out the network traffic generated in this case. It indicates an economy of network traffic even if profiles diverge from real behaviour.

Figure 5 shows the system throughput in this case. When relaxing divergence bounds the throughput is increased. This result, combined with the network traffic generated demonstrates that there is a real gain of using the divergence control protocol.

6 Related Work

In the area of weak consistency, several protocols have been proposed. Bayou proposes an optimistic replication protocol that addresses the specific needs of collaborative editing in a disconnected environment [TTP+95]. Bayou proposes an anti-entropy protocol for propa-

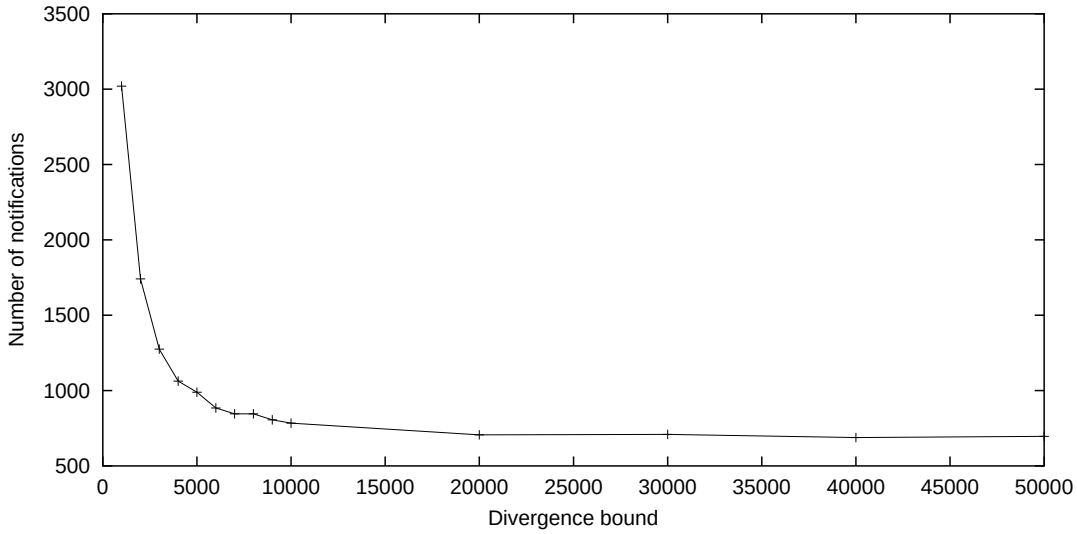

Figure 4: Network traffic in case 3

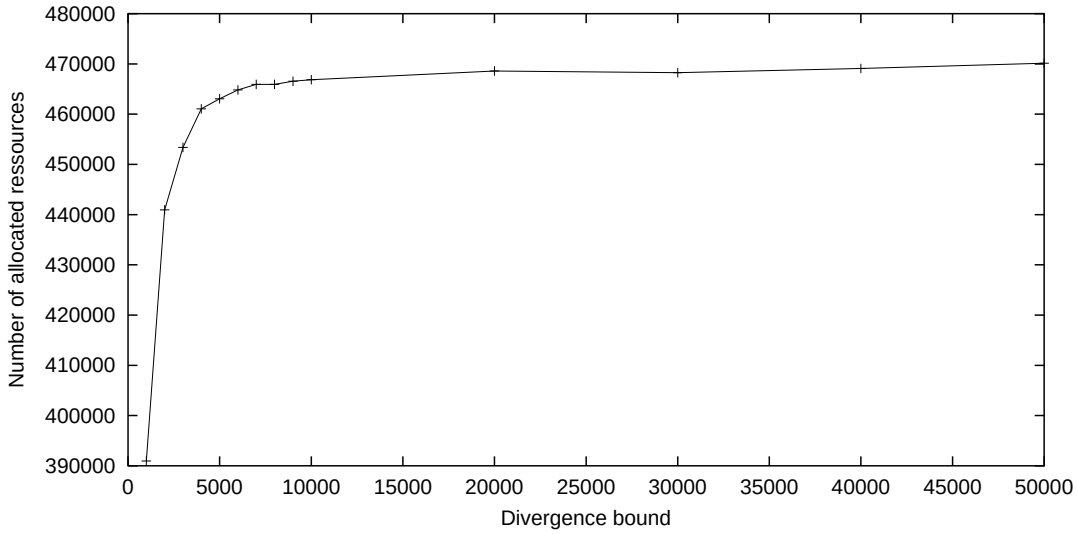

Figure 5: Application throughput in case 3

gating updates. For each update a certification procedure and a reconciliation procedure are performed. It implements session guarantees to maintain consistency for clients switching from a replica to another. Like Bayou, our proposal is adapted for the cooperation between disconnected replicas. Unlike Bayou, we want to guarantee a bound on the divergence observed on each process.

Epsilon serializability [PL90, PL91] implements a transaction model on queries and updates. Each query or update introduces an inconsistency in the data. Two inconsistency accumulators are associated with each epsilon transaction: the first indicates the total amount of imported inconsistency, while the second indicates the inconsistency exported by the transaction. A transaction is not performed when those accumulators exceed a certain limit.

Alonso and al. propose the concept of "quasi-copy" caching [ABM90]. Updates are propagated to cached copies according to specified coherency conditions. Four types coherency conditions are proposed: arithmetic, delay, periodic and version. These conditions define the divergence that is permitted between the primary copy of an object and cached copies. These conditions can be implemented in our framework provided that one can define the suitable measure functions and can predict the primary copy access profile.

The formulation of consistency constraints in the demarcation protocol [BGM94] has some resemblance to the way we express our consistency criterium. Our protocol differs from the demarcation protocol in the way each replica determines its degree of liberty. In our case, replicas rely on their knowledge of the expected behaviour of others and on their allowable divergence set at the start time, whereas in the demarcation protocol a replica has to ask other replicas to change its allowable divergence.

Krishnakumar and Bernstein N-ignorance protocol [KB94] controls the divergence between database replicas by bounding to N the number of conflicting transactions of the same type that the system is allowed to run in parallel on different replicas. N is an inconsistency bound definition. This concept does not capture the effective impact of transactions that are initiated concurrently on different replicas. Our protocol is interested to bound that effect rather than the number of transactions.

Our divergence control protocol is closely related to the TACT error bounding algorithm [YV00b, YV00a]. We attach a measure to each operation, which is similar to the notion of weight in TACT. However, relying on profiles allows to maintain the bound, even when replicas can not communicate. This might lead to a degradation of the service, rather than the denial of the service due to the inaccessibility of some partner.

7 Conclusion

We presented a divergence control protocol based on application profiles. The protocol controls on each replica the effect of the unexpected operations and ensures that this remains smaller than a given local bound. The protocol was tested with an allocation service. The performance analysis in terms of network traffic and system throughput show that the protocol is efficient when profiles used match the observed reality. In the near future we

plan to investigate how to capture the actual behaviour of the application. We are also interested to studying how to adapt bounding criterium in order to optimize the overall performance.

Acknowledgment

The replication model presented in this paper was defined in collaboration with Georges Brun-Cottan [BC98].

References

[ABM90] Rafael Alonso, Daniel Barbara, and Hector Garcia Molina. Data caching issues in an information retrieval system. *ACM Transactions on Database Systems*, 15(3):359–384, 1990.

[BC98] Georges Brun-Cottan. *Cohérence de données répliquées partagées par un groupe de processus coopérant à distance.* Thèse de doctorat, Université Paris 6, Pierre et Marie Curie, Paris (France), September 1998.

[BGM94] Daniel Barbara and Hector Garcia-Molina. The demarcation protocol: A technique for maintaining constraints in distributed database systems. *The International Journal on Very Large Data Bases*, 3(3):325–355, 1994.

[Hay98] Mark Hayden. The ensemble system. Technical Report TR98-1662, Cornell University, January 1998.

[KB94] Narayanan Krishnakumar and Athur Bernstein. Bounded ignorance: A technique for increasing concurrency in replicated system. *ACM Transaction on Data Base Systems*, 19(4):586 – 625, February 1994.

[LLSG92] Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Transactions on Database Systems*, 10(4):360–391, 1992.

[PL90] Calton Pu and Avraham Leff. Epsilon-serialisability. Technical Report CUCS-054-90, Columbia University, 1990.

[PL91] Calton Pu and Avraham Leff. Replica control in distributed system. an asynchronous approach. In *the 1991 International conference on the managment of Data, ACM SIGMOD Record*, pages 377–386. ACM SIGOPS, ACM press, May 1991.

[PST+97] Karin Petersen, Mike J. Spreitzer, Douglas B. Terry, Marvin M. Theimer, and Alan J. Demers. Flexible update propagation for weakly consistent replication.

In *16th ACM Symposium on Operating Systems Principles*, Saint Malo, France, October 1997.

[TTP+95] Douglas B. Terry, Marvin M. Theimer, Karin Petersen, Alan J. Demers, Mike J. Spreitzer, and Carl H. Hauser. Managing update conflicts in bayou, a weakly connected replicated storage system. *the 15th ACM symposium on Operating System Principles, Operating System review*, 5(29):172–183, December 1995.

[Vap98] Vladimir Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[YV00a] Haifeng Yu and Amin Vahdat. Design and evaluation of a continuous consistency model for replicated services. In *Fourth Symposium on Operating Systems Design and Implementation*, October 2000.

[YV00b] Haifeng Yu and Amin Vahdat. Efficient numerical bounding for replicated network services. In *the 26th International Conference on Very Large Databases (VLDB)*, September 2000.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399