

Evaluation du caractère adaptatif d'un protocole de consensus de type "Fast Paxos"

I. Moise¹ and M. Hurfin² and J.-P. Le Narzul³ and F. Majorczyk¹

¹Université de Rennes 1, IRISA / INRIA, Campus de Beaulieu, 35042 Rennes Cedex, France

²INRIA Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes Cedex, France

³Institut Télécom/Télécom Bretagne, 2 rue de la Châtaigneraie, 35576 Cesson Sévigné, France

Afin de réduire le temps de convergence vers une valeur de décision, certains protocoles de consensus anticipent l'absence de collision. Ce principe de conception a notamment été étudié par Leslie Lamport qui a proposé une variante au protocole *Paxos*, appelée *Fast Paxos*. Si toutes les valeurs proposées sont identiques, le nombre d'étapes de communication est réduit et le temps de convergence est ainsi diminué. Dans cet article, nous évaluons le protocole Paxos-MIC qui propose une optimisation mettant en œuvre ce principe de conception. Le protocole exécute une séquence d'instances de consensus ; avant chaque nouveau consensus, la décision d'activer l'optimisation est prise localement et dynamiquement en évaluant un critère. Ce critère d'activation se doit d'être le plus précis possible car en cas d'activation à tort de l'optimisation, une procédure de recouvrement s'avère nécessaire ; le coût de cette procédure est bien plus élevé que le gain obtenu par une activation justifiée de l'optimisation. Le critère d'activation peut revêtir différentes formes et éventuellement s'appuyer sur la connaissance du passé récent afin d'évaluer le risque d'occurrence d'une collision lors du prochain consensus. Nous avons mené un travail portant sur un serveur WEB sécurisé où le consensus est utilisé pour ordonner des requêtes. En utilisant le log du serveur WEB d'une grande école sur 15 jours d'activité, nous avons évalué les gains de performance obtenus en fonction du choix du critère d'activation.

Keywords: Problème d'accord, Consensus, Paxos, Fast Paxos, Collision, Protocol adaptatif

1 Introduction

Nous considérons le problème du consensus dans un système distribué asynchrone sujet à des défaillances. La communication entre les nœuds se fait par échange de messages. Les défaillances sont de type panne franche et les messages peuvent être dupliqués ou perdus mais pas corrompus. Nous nous intéressons ici aux protocoles de consensus de la « famille » Paxos [Lam01, Lam06, HMLN11] basé sur une élection de leader ultime. Au cours d'une instance de consensus, des valeurs initiales (potentiellement distinctes) peuvent être proposées par un ou plusieurs nœuds appelés des *auteurs de propositions*. Le ou les auteurs communiquent leur proposition au même sous-ensemble de n nœuds du système. Ces n nœuds que nous appellerons par la suite les *acteurs* sont directement impliqués dans l'exécution du protocole de consensus. Ils doivent collaborer entre eux pour assurer que le protocole de consensus qu'ils exécutent converge inéluctablement (propriété de terminaison) vers une valeur de décision unique (propriété d'accord uniforme) qui doit nécessairement être l'une des valeurs proposées (propriété de validité). La valeur de décision est ensuite retournée vers tous les auteurs de propositions qui ont fourni (ou fourniront) une valeur initiale concernant cette instance de consensus particulière.

Depuis la version initiale [Lam01], deux optimisations majeures ont été proposées. Nous dénotons la première, O_1 . La seconde, notée ici O_2 , est au cœur de la solution Fast Paxos présentée par Lamport dans [Lam06]. Les deux optimisations sont compatibles et visent à réduire le nombre d'étapes de communication qui doivent impérativement être effectuées entre l'instant où un premier auteur transmet une proposition initiale et l'instant où il recevra la valeur de décision. Nous avons mis en œuvre ces deux optimisations dans le protocole Paxos-MIC, décrit dans [HMLN11]. La section 2 présente brièvement Paxos-MIC. La section 3 donne des mesures de la durée moyenne d'exécution du protocole. Enfin, la section 4 évalue son caractère adaptatif.

2 Le protocole Paxos-MIC

Deux rôles (*Coordinateur* et *Accepteur*) sont définis et chaque acteur joue un ou deux rôles. Les n acteurs (et de fait, une majorité de nœuds corrects) jouent le rôle d'accepteur alors qu'au moins $f + 1$ acteurs (et de fait, au moins un nœud correct) jouent le rôle de coordinateur. Un coordinateur n'est actif que lorsque le service d'élection de leader le désigne comme leader. En pensant alors agir comme un leader unique (ce qui n'est pas forcément le cas), le coordinateur tente d'imposer une valeur de décision aux autres acteurs. Dans les protocoles Paxos, un numéro de tour r est associé à chaque tentative. Dans la version originale de Paxos, un leader tente d'imposer une valeur de décision en exécutant successivement deux phases au cours d'un tour r . Au cours d'une première phase de *Préparation*, le leader s'assure que la valeur qu'il soumettra lors de la seconde phase n'est pas incompatible avec celles éventuellement soumises par d'autres coordinateurs ayant agi comme leader au cours du même consensus. La phase de préparation implique la diffusion d'un message du leader vers les accepteurs puis la collecte, par le leader, de réponses favorables en provenance d'une majorité d'accepteurs (2 étapes de communication). La seconde phase est une phase de *Proposition*. Elle débute dès que le leader a identifié une valeur qu'il peut soumettre sans risquer de violer les propriétés de sûreté (Accord et Validité). La valeur est alors diffusée par le leader vers les accepteurs puis le leader attend de collecter une majorité de réponses favorables avant de pouvoir considérer que cette valeur soumise est la valeur de décision (2 étapes de communication). Un accepteur est une entité passive dont l'état fait référence à la dernière phase de préparation acceptée et à la dernière phase de proposition acceptée.

Dans sa version originale, le protocole requiert 4 étapes de communications entre les acteurs plus 2 étapes pour la diffusion des valeurs initiales des auteurs vers les coordinateurs et la diffusion de la décision du leader vers les auteurs. Le chemin de communication est : auteur \triangleright coordinateurs (leader) \triangleright accepteurs \triangleright leader \triangleright accepteurs \triangleright leader \triangleright auteurs. L'optimisation O_1 consiste à supprimer une partie du calcul (phase de préparation) lorsqu'elle est inutile. Un tour r comporte alors une seule phase de préparation, qui sera suivie de plusieurs phases de proposition, toutes lancées par le leader avant qu'il ne soit destitué. En conséquence, plusieurs instances de consensus peuvent donc être exécutées durant le même tour. Lorsque le leader élu reste stable (pas de défaillance, interactions avec les acteurs suffisamment synchrones), le chemin de communication est de longueur 4 : auteur \triangleright coordinateurs (leader) \triangleright accepteurs \triangleright leader \triangleright auteurs.

L'optimisation O_2 consiste à exécuter de façon anticipée la partie de la phase de proposition correspondant à la diffusion par le leader d'une requête et à sa réception par les accepteurs. L'objectif est de tirer profit du fait que, durant la plupart des instances de consensus, un seul auteur participe au consensus et donc une seule valeur initiale est disponible. Les auteurs externes qui ne savent pas si O_2 est activée doivent désormais diffuser systématiquement leur valeur initiale à l'ensemble des coordinateurs et à l'ensemble des accepteurs. Le protocole exécuté par les n acteurs est lui aussi modifié. Lorsque l'exécution du consensus $k - 1$ se termine, un leader qui dispose déjà d'une proposition concernant le consensus k n'active pas O_2 et effectue un consensus en n'utilisant que O_1 : nous dirons dans ce cas que l'activation de O_2 n'a pas de sens. Si au contraire, aucune proposition n'est disponible, le leader est inactif et peut décider durant cette période d'inactivité d'activer O_2 . Dans notre protocole, ce choix est fait par le leader en évaluant un *test d'activation*. Si une proposition parvient au leader avant que le test n'autorise l'activation de O_2 , le consensus k s'exécute alors en n'utilisant que O_1 . Si le test est satisfait, le leader qui ne connaît aucune valeur initiale pour le consensus k adopte une valeur par défaut (ANY) qu'il soumet lors de la phase de proposition. Tout accepteur recevant cette valeur fictive est alors autorisé à la remplacer par une vraie valeur initiale reçue d'un auteur. Lorsque l'activation de O_2 est un succès, le chemin de communication est de longueur 3 et correspond à : auteur \triangleright accepteurs \triangleright leader \triangleright auteurs. Sans O_2 , une seule valeur initiale est soumise durant une phase de proposition, et donc tous les accepteurs qui acceptent une valeur durant cette phase adopte de fait la même valeur. Avec O_2 , cette propriété n'est plus toujours vérifiée dès lors que 2 auteurs fournissent 2 valeurs initiales distinctes durant l'instance k . Ceci a deux conséquences majeures. D'une part, la définition d'un quorum est plus contraignante : le leader doit collecter plus d'acceptations. Dans notre mise en oeuvre, le nombre de retours attendus passe de $\lceil n/2 \rceil$ à $\lceil 3n/4 \rceil$. D'autre part, O_2 est un échec quand les retours collectés par le leader font référence à plus d'une valeur ; on parle alors de *collision*. Un recouvrement est nécessaire : un nouveau tour est démarré par le coordinateur (exécution d'une phase de préparation) et une nouvelle phase de proposition est lancée mais cette fois, sans activer O_2 .

3 Mesures de la durée d'un consensus

Paxos-MIC a été mis en œuvre en Java ; nous avons ensuite utilisé cette mise en œuvre pour réaliser des mesures sur la grille GRID 5000[†] et obtenir ainsi les durées moyennes d'un consensus, en fonction de l'activation (ou pas) de O_2 et du succès (ou de l'échec) de cette activation. Nous avons considéré diverses configurations en termes de nombres d'accepteurs et de placement des auteurs : C_{RR05} avec auteurs et 5 accepteurs placés sur le site de Rennes, C_{RR11} avec auteurs et 11 accepteurs à Rennes également et enfin, C_{OR05} qui se différencie du contexte C_{RR05} de par le positionnement des auteurs sur le site d'Orsay. Pour cette dernière configuration, la latence observée entre le site où sont placés les auteurs et celui où sont placés les accepteurs est environ 70 fois supérieure à la latence intra-site. Pour chacune de ces configurations, nous avons une durée moyenne pour un consensus sans activation de O_2 (DN pour Durée Normale), une durée moyenne pour un consensus avec activation de O_2 sans collision (DR pour Durée Réduite) et une durée moyenne pour un consensus avec activation de O_2 conduisant à une collision (DD pour Durée Dégradée). La table 1 donne ces durées pour les 3 configurations considérées. Ces mesures montrent que les gains obtenus grâce à O_2 et les pertes subies en raison d'une collision dépendent fortement de la configuration. Le ratio entre le gain ($DN - DR$) et la perte ($DD - DN$) peut ainsi varier de 2 (C_{RR05}) à 9 (C_{RR11}).

	DR	DN	DD	ratio (perte/gain)
C_{RR05}	1.312ms	1.915ms	3.149ms	2
C_{RR11}	1.775ms	2.121ms	5.175ms	9
C_{OR05}	9.399ms	9.741ms	11.505ms	5

TABLE 1: Durée moyenne d'un consensus

4 Évaluation du caractère adaptatif de Paxos-MIC

Nous avons analysé un log de requêtes HTTP issu du serveur WEB d'une grande école sur une durée de 15 jours. Le consensus est utilisé par un protocole de diffusion atomique qui établit un ordre unique sur les requêtes. Sur la base de ce log, nous avons simulé l'exécution du protocole pour ordonner les requêtes HTTP et nous avons calculé le temps moyen d'un consensus en fonction de divers critères d'activation de O_2 et pour les différentes configurations identifiées précédemment. Nous avons utilisé 5 critères d'activation ("*jamais*", "*toujours*", "*temps*", "*résultat*" et "*aléatoire*") de O_2 . Avec le critère "*toujours*", O_2 est toujours activée dès lors qu'il y a une période d'inactivité. Avec le critère "*jamais*", O_2 n'est jamais activée. Avec le critère "*temps*", O_2 est activée lorsque, suite à la réception d'une requête R_i (ayant déjà déclenché un consensus), la requête suivante R_{i+1} n'est pas reçue avant un délai de Δ ms. On considère alors que l'on est dans une période d'inactivité devant permettre d'activer avec succès l'optimisation O_2 pour le consensus suivant (déclenché pour R_{i+1}). Avec le critère "*résultat*", par défaut, l'optimisation O_2 est activée. Dès lors qu'un consensus enclenché pour une requête R_i subit une collision, O_2 n'est pas activée pour les requêtes R_{i+1} et R_{i+2} . Le consensus pour R_{i+3} sera ensuite déclenché en activant O_2 . Dans le cas du critère "*aléatoire*", O_2 est activée de façon aléatoire selon une loi binomiale : pour 80% des consensus, O_2 est activée. Ce test qui ne prend pas en compte le passé récent peut être vu comme une stratégie intermédiaire entre "*jamais*" et "*toujours*". Dans l'ensemble de nos calculs, le risque de collision est toujours évalué de la même façon en fonction de la concomitance des requêtes HTTP présentes dans le log.

Au lieu de donner les valeurs brutes obtenues lors de la simulation (temps moyen d'un consensus en ms), nous préférons présenter nos résultats sous la forme de gains par rapport à une valeur optimale. Cette valeur optimale est une valeur idéale où on considère qu'il n'y a aucune collision, c'est-à-dire que O_2 a toujours été activée à bon escient, avec une durée du consensus égale à DR et que pour les cas où O_2 n'a pas été

[†]. Les expériences présentées dans cet article ont été réalisées en utilisant la plateforme expérimentale Grid 5000, issue de l'Action de Développement Technologique (ADT) Aladdin pour l'INRIA, avec le support du CNRS, de RENATER, de plusieurs Universités et autres contributeurs (<https://www.grid5000.fr>).

activée, la durée du consensus est DN. La valeur optimale t_{opt} est obtenue en calculant la durée moyenne d'un consensus au prorata du nombre de consensus réalisés en DR ms et de ceux réalisés en DN ms. De même, nous calculons la valeur d'un consensus « au pire » t_{pire} en considérant que l'optimisation O_2 n'est déclenchée que dans les cas où elle ne devrait pas l'être. Pour un critère donné, le gain par rapport à la valeur optimale est alors calculé de manière à (i) être égal à 0% lorsque le temps moyen du consensus est égal à la valeur « au pire » et ii) être égal à 100% lorsque ce temps moyen est égale à la valeur optimale. Ainsi, si on considère t , le temps moyen d'un consensus pour une activation de O_2 selon un critère donné, le gain est égal à $(t_{pire} - t)/(t_{pire} - t_{opt})$. Les gains calculés sont présentés dans la table 2 pour les diverses configurations décrites à la section 3. La partie gauche du tableau correspond à une analyse sur les 15 jours du log alors que la partie droite correspond à une analyse sur une période plus courte, de forte activité.

	Log complet (15 jours)					Période de forte activité (4 heures)				
	"jamais"	"toujours"	"temps"	"résultat"	"aléatoire"	"jamais"	"toujours"	"temps"	"résultat"	"aléatoire"
C _{RR05}	4.84%	95.06%	90.22%	93.82%	76.96 %	12.48%	87.08%	76.83%	83.73%	71.75%
C _{RR11}	19.90%	79.96%	77.58%	80.11%	67.89%	42.25 %	57.24 %	55.57 %	57.48 %	54.53 %
C _{OR05}	27.93%	71.94%	71.88%	72.25%	63.12%	54.05 %	45.59 %	45.69 %	47.41 %	47.67 %

TABLE 2: Gains obtenus en fonction des critères et des configurations

Dans le cas du log complet, nous observons que pour la configuration C_{RR05}, les gains obtenus sont très élevés avec les 4 critères activant O_2 (avec une diminution notable pour "aléatoire"). La comparaison de ces gains avec celui obtenu avec le critère "jamais" (aucune activation) montre sans ambiguïté l'intérêt de l'optimisation O_2 . Concernant les deux autres configurations (C_{RR11} et C_{OR05}), les gains "toujours", "temps", "résultat" et "aléatoire" diminuent par rapport à ceux obtenus pour la configuration C_{RR05} tout en restant toujours nettement supérieurs à celui obtenu avec "jamais". Pour la configuration C_{RR11}, l'explication réside dans le ratio perte/gain qui est en augmentation significative (il faut beaucoup d'activations réussies pour compenser des échecs). Pour C_{OR05}, l'augmentation du ratio explique partiellement la diminution du gain ; la réduction du nombre d'activations de O_2 [‡], participe aussi à cette diminution.

Nous avons également procédé aux mêmes calculs de gains en considérant une période plus restreinte du log de requêtes (4 heures au lieu de 15 jours) pendant laquelle l'activité était plus importante. Pour la configuration C_{RR05}, les gains avec les critères "toujours", "temps", "résultat" et "aléatoire" diminuent en raison d'un nombre d'erreurs (collisions) plus important. Ici, il est surtout intéressant de constater que pour la configuration C_{OR05}, le critère "jamais" (pas d'activation de O_2) devient alors un meilleur choix que le critère "toujours" (activation systématique de O_2). Les critères basés sur un test du passé récent ("temps" et "résultat"), donnent des résultats intermédiaires et témoignent ainsi de leur intérêt, aussi bien lors de périodes de forte activité que lors de périodes d'activité variable. D'autres tests réalisés (non présentés ici, faute de place) en combinant les critères "temps" et "résultat" ou encore en faisant varier le Δ du critère "temps" (qui peut être vu alors comme un réglage permettant de faire évoluer le critère sur une échelle allant de "toujours" à "jamais") donnent des résultats du même ordre. Nous avons également observé qu'en fonction de la période retenue dans la trace (jour/nuît par exemple), certains critères s'avèrent plus pertinents et donnent donc de meilleurs résultats. Il pourrait donc être intéressant d'effectuer une phase d'apprentissage de façon à définir a priori le meilleur test possible en fonction de la période considérée.

Références

- [HMLN11] M. Hurfin, I. Moise, and J.-P. Le Narzul. An Adaptive Fast Paxos for Making Quick Everlasting Decisions. In *Proc. of the 25th IEEE Int. Conf. on Advanced Information Networking and Applications (AINA)*, 2011.
- [Lam01] L. Lamport. Paxos Made Simple. *ACM SIGACT News*, 32(4) :51–58, Dec. 2001.
- [Lam06] L. Lamport. Fast Paxos. *Distributed Computing*, 19(2) :79–103, 2006.

‡. Le nombre d'activations de O_2 n'est pas apparent dans la table 2 de même que les pourcentages de succès (pas de collision) et d'erreur (collisions) calculés pour les trois configurations et les différents critères d'activation.