The Computability Hierarchy of Broadcast Abstractions

Matthieu Perrin
matthieu.perrin@univ-nantes.fr

Achour Mostéfaoui
achour.mostefaoui@univ-nantes.fr

Keywords: *Abstractions, Algorithmes, Calcul réparti, Calculabilité, Diffusion de messages, Fiabilité, Structures de données partagées, Tolérance aux pannes*

Contexte

Les systèmes répartis sont composés d'un ensemble d'entités indépendantes, qu'il s'agisse de réseaux décentralisés de machines informatiques indépendantes, d'applications multi-thread exécutées au sein d'une même machine multi-cœur, de superordinateurs composés de clusters de GPU, ou même de sociétés humaines complexes. Ils ont récemment été de plus en plus présents dans nos vies : En 2023, le monde compte désormais plus de 5,3 milliards d'utilisateurs d'Internet, le processeur de téléphone portable le plus simple d'aujourd'hui est multi-cœur, le cloud computing est désormais une technologie bien établie utilisée par d'innombrables entreprises grâce à l'explosion des services qu'il offre, la géo-réplication est largement utilisée pour garantir la persistance des données sensibles, etc.

Malgré toutes les applications possibles pour les systèmes répartis, les applications réparties sont notoirement difficiles à programmer, d'une part parce que chaque système réparti a ses propres spécificités qui rendent les algorithmes dépendants du système considéré, et d'autre part parce qu'il est actuellement très difficile de dissocier les aspects fonctionnels spécifiques à l'application en cours de développement de ceux relatifs aux problèmes de synchronisation spécifiques aux systèmes distribués.

Les structures de données partagées sont centrales dans les applications réparties car elles encapsulent les données partagées et abstraient les mécanismes de synchronisation basés sur des primitives de communication disponibles pour les processus, telles que la mémoire partagée ou les canaux de communication. Supposons, par exemple, que l'on doive mettre en œuvre un nouveau service de commerce électronique. Une structure de données centrale du projet serait un ensemble d'objets vendables, dans lequel des objets peuvent être insérés lorsqu'ils sont mis en vente, et supprimés après leur vente. D'un point de vue fonctionnel, l'implémentation de la structure de données doit se concentrer sur la facilité d'utilisation de cette structure de données, par exemple en fournissant une interface de requête agréable, permettant de filtrer facilement les éléments en fonction des achats précédents de l'utilisateur, ou d'optimiser les prix en utilisant des algorithmes d'apprentissage automatique de pointe. Cependant, en pratique, une grande partie du code de la structure de données devra se concentrer sur des mécanismes de synchronisation de bas niveau nécessaires pour la rendre sûre à utiliser dans un environnement distribué.

Par exemple, lorsqu'un article est vendu dans un centre de données, il doit devenir immédiatement invisible dans d'autres centres de données malgré l'asynchronie et les pannes, donc un

algorithme de synchronisation spécifique doit être mis en œuvre pour maintenir la cohérence (cela est similaire à la mise en œuvre d'un registre atomique booléen pour encoder si un objet spécifique est disponible ou non) qui nécessite l'envoi de messages et l'attente d'accusés de réception, éventuellement deux fois [2]. Différemment, évaluer une requête de lecture sur un état cohérent de l'ensemble nécessiterait des algorithmes de snapshot supplémentaires non intuitifs basés sur des stratégies de double collecte et des mécanismes d'aide complexes [1]... Cette méthodologie aboutit souvent à des algorithmes très compliqués qui ne peuvent être conçus que par des experts dans plusieurs domaines de l'informatique distribuée, et qui sont très difficiles à maintenir lorsque les besoins de l'application évoluent. Pire encore, les mécanismes de synchronisation décrits ci-dessus ne fonctionnent que si les hypothèses fortes faites dans [2] sont pertinentes dans le système sur lequel la boutique de commerce électronique est construite. Sinon, par exemple si des types de défaillances plus variés sont susceptibles de se produire, si les processus sont incapables de communiquer directement les uns avec les autres (par exemple, dans les systèmes pair-à-pair), ou si l'échelle du système rend l'attente de quorums impraticable, des algorithmes complètement différents doivent être conçus pour résoudre le même problème.

Abstractions de diffusion

Lorsque les processus communiquent en envoyant et en recevant des messages, la réplication est utilisée pour éviter une perte d'information en cas de crash. Une difficulté majeure résultant de la réplication est de maintenir des données cohérentes sur toutes les répliques lorsque des mises à jour sont effectuées concomitamment. La diffusion fiable garantit que tous les processus non fautifs reçoivent exactement le même ensemble de messages. Elle peut être utilisée pour s'assurer que toutes les répliques seront informées lorsqu'un processus effectue une opération. Cependant, la diffusion fiable ne fournit aucune forme de garantie d'ordre sur les messages, donc une synchronisation supplémentaire est nécessaire pour maintenir un état cohérent sur toutes les répliques.

La diffusion fiable peut être enrichie en restreignant les ordres possibles de réception des messages. Par exemple, la diffusion causale garantit que, si un processus p reçoit d'abord un message m et diffuse ensuite un message m', aucun processus ne reçoit m' avant m. La diffusion causale peut être utilisée pour résoudre certains problèmes de cohérence [13], mais pas tous. Un autre exemple est la diffusion d'ordre total, qui garantit que toutes les répliques recevront tous les messages dans le même ordre. La diffusion d'ordre total permet de résoudre tous les problèmes de synchronisation énoncés ci-dessus, mais à un coût beaucoup plus élevé que la cohérence causale. La mise en œuvre de la diffusion fiable, de la diffusion causale et de la diffusion atomique a été largement étudiée dans différents systèmes [3, 4, 8, 12].

La diffusion d'ordre total et la diffusion causale ne sont pas équivalentes, car la mise en œuvre de la diffusion d'ordre total sur la base de la diffusion causale nécessite des hypothèses fortes sur les modèles de défaillance et les délais de routage des messages, qui peuvent, ou non, être réalistes selon les spécificités du système considéré. De même, la diffusion d'ordre total peut être plus coûteuse que ce qui est requis pour le service de commerce électronique décrit ci-dessus (ou tout autre système partagé), ou peut fournir des garanties qui ne sont nécessaires que rarement.

Nous avons récemment découvert plusieurs abstractions intermédiaires entre la diffusion fiable et la diffusion atomique, mais équivalentes à divers problèmes centraux des algorithmes distribués, qui permettent des mises en œuvre très simples, élégantes et efficaces d'une variété de structures de données partagées. Par exemple, la diffusion mutuelle [6] capture la lecture et l'écriture dans

une variable partagée, la diffusion SCD [9] est équivalente à l'objet snapshot défini ci-dessus, et la diffusion par paires [6] est exactement l'abstraction requise pour s'assurer qu'un, et un seul, client peut acheter le dernier article du stock, lorsque plusieurs tentent de le faire simultanément.

Cela nous a convaincus que les primitives de diffusion sont « les bonnes abstractions » pour construire un « modèle standard » des problèmes de synchronisation, qui pourrait relever les deux défis de permettre des implémentations simples de structures de données partagées que les ingénieurs non-experts peuvent utiliser pour construire des systèmes complexes efficaces, ainsi que de fournir le fondement pour unifier les systèmes distribués spécifiques. En d'autres termes, les primitives de diffusion peuvent servir de blocs de construction qui permettent aux processus de synchroniser leur copies locales de données partagées (répliques), tout en apportant une étincelle d'homogénéité entre les systèmes.

Objectifs

Le recruté sera tenu d'effectuer des recherches avancées, avec l'intention de publier sur la question suivante : *Quels problèmes distribués peuvent être abstraits avec des abstractions de diffusion, et comment ?* Ce problème général peut être divisé en plusieurs problèmes plus simples, tels que :

- Étant donné un problème de synchronisation spécifique, (par exemple, le consensus entre k processus ou l'accord k-ensemble), existe-t-il une abstraction de diffusion qui le capture ?
- Existe-t-il des problèmes de synchronisation dont la complexité ne peut pas être capturée exactement par une abstraction de diffusion ? Comment le prouver ?
- Comment la hiérarchie des abstractions de diffusion se compare-t-elle à d'autres hiérarchies du calcul réparti, comme celle du plus faible détecteur de fautes, ou celle du numéro de consensus ?
- Comment mettre en œuvre efficacement les nouvelles abstractions trouvées dans divers modèles d'informatique distribuée ?

Après une phase de familiarisation avec le sujet et les problèmes qu'il soulève, le recruté devra choisir des questions ouvertes et tenter d'y répondre, soit en proposant un algorithme, soit en démontrant un théorème d'impossibilité. Ce poste nécessite des compétences avancées en raisonnement abstrait, et des compétences avancées en informatique distribuée, en particulier en ce qui concerne la théorie de la calculabilité distribuée.

Il se déroulera au *Laboratoire des Sciences du Numérique de Nantes* (LS2N) de l'Université de Nantes. L'équipe GDD (Gestion des Données Distribuées – Distributed Data Management) est internationalement reconnue dans les domaines de l'informatique distribuée et parallèle, du Web et des bases de données.

N'hésitez pas à nous contacter pour plus d'informations !

References

[1] Afek, Y. Hagit A. Dolev D. *et al. Atomic snapshots of shared memory.* Journal of the ACM, 1993.

[2] Attiya H., Bar-Noy A. and Dolev D. *Sharing memory robustly in message-passing systems.* Journal of the ACM, 1995.

[3] Birman K. P. and Joseph, T. A. *Reliable communication in the presence of failures.* TOCS, 1987.

[4] Birman K. P., Hayden M., Ozkasap O *et al. Bimodal multicast.* TOCS, 1999.

[5] Demers, A., Greene, D., Houser *et al. Epidemic algorithms for replicated database maintenance.* ACM SIGOPS Operating Systems Review, 1988.

[6] Déprés M., Mostéfaoui A., Perrin M. and Raynal M. *Send/Receive Patterns Versus Read/Write Patterns in Crash-Prone Asynchronous Distributed Systems.* DISC, 2023.

[7] Gilbert S. and Lynch N. *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.* ACM SIGACT News, 2002.

[8] Hadzilacos V. and Toueg S. *Reliable Broadcast and Related Problems.* Journal of Distributed Systems, 2013.

[9] Imbs, D., Mostefaoui, A., Perrin, M. and Raynal, M. *Set-constrained delivery broadcast: Definition, abstraction power, and computability limits.* ICDCS, 2018.

[10] Kleinberg, J. *The small-world phenomenon: An algorithmic perspective.* STOC, 2000.

[11] Lamport, L. *The part-time parliament.* TOCS, 1998.

[12] Nédelec, B. Molli P. and Mostefaoui A. *Breaking the Scalability Barrier of Causal Broadcast for Large and Dynamic Systems.* SRDS, 2018.

[13] Perrin, M., Mostefaoui, A. and Jard, C. *Causal consistency: beyond memory.* PPOPP, 2016.

[14] Voulgaris, S., Gavidia, D. and Van Steen, M. *Cyclon: Inexpensive membership management for unstructured p2p overlays.* Journal of Network and systems Management, 2005.