状態機械複製プロトコルRabiaにおけるネットワーク分断耐性強化

木田 碧[1,a)]　川島 英之[2,b)]

1. 背景

現代のアプリケーションでは, ユーザーへのサービスを中断させないために高可用性が必要とされている. ダウンタイムは重大な経済的損失と評判の低下をもたらす可能性があるため, 現代のアプリケーションを支える基盤にとって高可用性は重要な要件となっている. 状態機械複製は強力な一貫性を保証し, 部分的な障害を許容するため, 様々なシステムの基盤に広く採用されている. しかし Multi-Paxos [1,2] や Raft [3] などの既存の状態機械複製プロトコルは, 理論および実装上の困難さにより, 実世界のシステムに適用するのが難しい. このような課題に対して, Rabia [5] は乱択二値合意アルゴリズム [4] を使用して設計と実装を簡素化した状態機械複製プロトコルである.

2. 研究課題

状態機械複製プロトコルにおいて, ネットワーク分断への耐性は重要な要件である. 我々は Rabia の部分的なネットワーク分断 [8, 9] への脆弱性を明らかにした. 部分的なネットワーク分断に直面した場合 Rabia の進行は停止する. Rabia プロトコルは完全なネットワーク分断 [6, 7] に対する耐性を示す一方で, 部分的なネットワーク分断に対処するために必要なメカニズムが欠如している. 部分的なネットワーク分断とはクラスタ内の特定のノード間の直接通信が中断される一方で, 他のノードは完全な接続性を維持するという複雑なネットワーク障害である. 部分的なネットワーク分断障害は誤設定されたファイアウォール, 故障したネットワーク機器, 不整合なルーティングテーブルなど様々な要因から生じる. この障害への脆弱性は状態機械複製の意図する高可用性を損なうため問題であると考える.

[1] 慶應義塾大学大学院 政策・メディア研究科
[2] 慶應義塾大学 環境情報学部
[a)] aoikida@sfc.keio.ac.jp
[b)] river@sfc.keio.ac.jp

3. 提案手法: Qsync

我々は部分的なネットワーク分断によって引き起こされる停止状態を検出し, その状態から回復することを可能にする手法, Qsync (Algorithm 1) を提案する. 進行停止は, 部分的なネットワーク分断によってレプリカ間のリクエストを格納するキューの状態が異なってしまい, 各レプリカから異なるリクエストが提案され, 特定の提案への合意に至らないという過程が繰り返されることで引き起こされる. したがって Qsync では連続して合意に失敗する数が通常発生し得ない数になった場合, 部分的なネットワーク分断が存在していると想定する. 閾値を超えると, 各レプリカは自身のキューの先頭のリクエストをすべてのレプリカに送信する (2-3 行目). レプリカはリクエストを受信すると, そのリクエストが自身のキューに既に存在せず, かつ現在提案中のリクエストとも異なる場合にのみ, そのリクエストをキューに追加する (5-7 行目). Qsync は部分的なネットワーク分断の発生を推定し, すべてのレプリカ間でキューの状態を同期することによって, 部分的なネットワーク分断による停止状態の克服を可能にする.

Algorithm 1 提案手法: Qsync

Input: $threshold$ ▷ for consecutive NULL agreements
Input: $request$ ▷ first element of PQ
Input: PQ ▷ replica's priority queue
Input: $proposal$ ▷ replica's current proposal
1: **procedure** DetectPartition
2: **if** ConsecutiveNULLCounts $\geq threshold$ **then**
3: Send ($request$)
4: **procedure** SynchronizeQueue
5: Upon receiving $request$
6: **if** $request \neq proposal$ **and** $request \notin PQ$ **then**
7: PQ.push($request$)

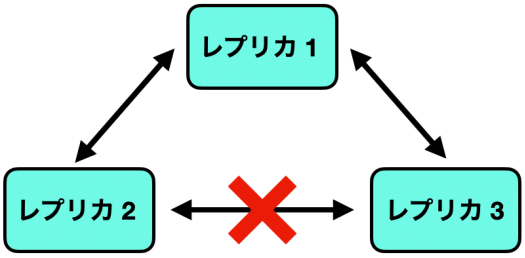

図 1: 部分的なネットワーク分断の例

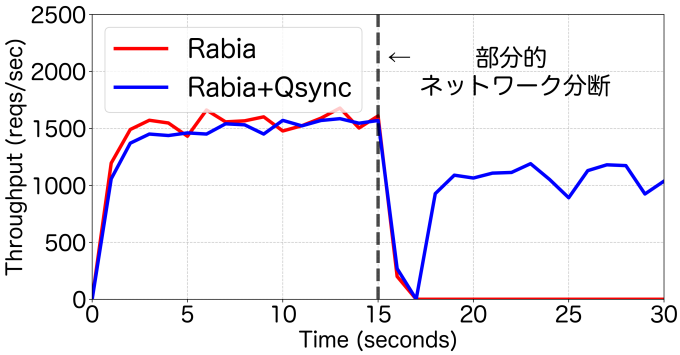

図 2: 部分的ネットワーク分断時のスループット比較

4. 評価

4.1 実験環境

Qsync を Rabia のオリジナルのコードベース [10] に実装した. 追加および修正されたコードは合計で約 50 行であり, これは Rabia の単純性を損なうものではない. レプリカ数は 3 つ, クライアントノード数は 3 つで, 1 クライアントノードあたり 2 つのクライアントスレッドを動作させた. 計 6 つのノードを Amazon EC2 t2.large インスタンス上で実行する. 基本的な機能を確認するため, いかなる形式のバッチ処理も使用せずに Rabia を評価した. 図 1 は本実験が想定する部分的なネットワーク分断である. レプリカ 2, レプリカ 3 間が分断され, 互いに通信できないが, レプリカ 1 はレプリカ 2, レプリカ 3 両方と通信できる. 本実験では停止状態からの回復を可視化するために, 部分的ネットワーク分断を検知する閾値を 1000 と高く設定している.

4.2 性能評価

部分的なネットワーク分断に対する Qsync の有効性を評価するため, オリジナルの Rabia プロトコルと, Rabia プロトコルに Qsync を組み込んだ拡張プロトコルの比較を行った (図 2).

部分的なネットワーク分断が発生する前は, オリジナルの Rabia と Qsync を組み込んだ Rabia の両方が同等のスループットを示している. 部分的ネットワーク分断が発生すると, オリジナルの Rabia はスループットが即座にゼロまで低下している. 一方 Qsync を組み込んだ Rabia は一時的な性能低下の後に, スループットは分断前の約 2/3 まで性能が回復している.

Qsync を組み込んだ Rabia の分断後の性能が分断前の性能にまで回復しないのは, クライアントからのリクエスト数が減少しているからである. 分断の影響で一つのレプリカが合意に参加できない状態に陥り, そのレプリカに接続しているクライアントのリクエストは提案されない. クライアント数が 2/3 になることによって, スループットが約 2/3 になっている. また分断後の一時的な性能低下は, 部分的ネットワーク分断を検知する閾値を高めに設定しているため, 回復までに時間がかかっているからである. 閾値をより小さな値にすれば, より早い性能回復が見込める.

謝辞 本研究の成果は, 国立研究開発法人新エネルギー・産業技術総合開発機構 (NEDO) の委託業務「ポスト 5G 情報通信システム基盤強化研究開発事業」(JPNP20017) 及び (JPNP16007), 日本学術振興会科研費（22H03596）及びセコム科学技術振興財団の助成により得られたものである.

参考文献

[1] Leslie Lamport. The part-time parliament. ACM Transactions on Computer Systems (TOCS), 16(2):133–169, 1998.
[2] Leslie Lamport. Paxos made simple. ACM Sigact News, 32(4):18–25, 2001.
[3] D. Ongaro, J. Ousterhout. 2014. In search of an understandable consensus algorithm. In USENIX Annual Technical Conference. 305–320.
[4] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. 1983. In Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing, PODC. 27–30.
[5] Haochen Pan, Jesse Tuglu, Neo Zhou, Tianshu Wang, Yicheng Shen, Xiong Zheng, Joseph Tassarotti, Lewis Tseng, and Roberto Palmieri. Rabia: Simplifying state-machine replication through randomization. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, SOSP ’21, 472–487.
[6] Seth Gilbert and Nancy Lynch. 2002. Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. SIGACT News 33, 2 (June 2002), 51–59.
[7] Brian A Coan, Brian M Oki, and Elliot K Kolodner. 1986. Limitations on database availability when networks partition. In Proceedings of the fifth annual ACM symposium on Principles of distributed computing (PODC ’86). 187–194.
[8] Mohammed Alfatafta, Basil Alkhatib, Ahmed Alquraan, and Samer Al-Kiswany. 2020. Toward a generic fault tolerance technique for partial network partitioning. In Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation (OSDI’20). USENIX Association, USA, Article 20, 351–368.
[9] Basil Alkhatib, Sreeharsha Udayashankar, Sara Qunaibi, Ahmed Alquraan, Mohammed Alfatafta, Wael Al-Manasrah, Alex Depoutovitch, and Samer Al-Kiswany. 2023. Partial Network Partitioning. ACM Trans. Comput. Syst. 41, 1–4, Article 1 (November 2023), 34 pages.
[10] Aoi Kida. 2024. Code of Rabia with Qsync. Retrieved October 15, 2024 from `https://github.com/aoikida/Rabia_with_Qsync`.