5J-01

パーソナライズされたコンテンツ配信のための低遅延分散KVSの構築

上田 義明 †　　碓井 利宣 ‡
†Supership 株式会社／放送大学　　‡ 東京大学 生産技術研究所

1 はじめに

ユーザの利便性向上や効果的なマーケティングのため，ユーザごとにパーソナライズされたコンテンツ配信が拡大している．パーソナライズされたコンテンツ配信では，ユーザのリクエストに合わせて配信するコンテンツを動的に決定して配信する．こうしたコンテンツ配信では，ユーザからの要求を受信してから，ユーザに配信するコンテンツを決定し，データストアから動的に取得して配信する必要があり，高速性が要求される．そのため，データストアには分散処理による高速化が期待できる分散 Key-Value Store (KVS) が用いられる．しかしながら，ユーザ数が増加すると，データ要求ごとの KVS ノードへの通信やデータ更新ごとの KVS ノード間の同期により，遅延が発生する問題があった．

本研究では，パーソナライズされたコンテンツ配信の特性を利用し高速応答が可能な分散 KVS の構築手法を提案する．提案手法では，アプリケーションと KVS を同一ノードに配置し，ネットワーク遅延を削減する．また，ユーザごとのデータの一貫性が保たれていれば良いことに着目し，ユーザごとに接続先を常に特定の KVS ノードに固定して，一貫性を損なわずにデータ同期の遅延を排する．本手法を用いた PoC の性能を評価し，分散 KVS の応答性能の向上を確認した．

2 要件

パーソナライズされたコンテンツ配信の一つに，配信する広告を広告事業者間でのリアルタイムの入札によって動的に決定する Real Time Bidding (RTB) がある．RTB では応答速度に要件があり，事業者間でリクエスト送信から規定時間内での応答が必須で，多くの事業者で 100 ミリ秒である．それには事業者間ネットワークにおける遅延も含まれるため，アプリケーションの処理時間は 50 ミリ秒程度に抑える必要があり，そのうちデータストアのアクセスに費やせるのは 10 ミリ秒程度であると言われている．この要件は，パーソナライズされたコンテンツ配信の中でも一般的なものの一つであるため，本研究では，この要件の実現を目指す．

したがって，コンテンツ配信に用いる分散 KVS では以下を満たす必要がある．

1. 10 ミリ秒以下で安定した高速な応答性能
2. 単一のユーザに一貫性のあるデータの提供

3 従来手法

従来の分散 KVS を構築する場合，2 つの構成が考えられる．更新と参照が可能なマスターノードと参照のみ可能なスレーブノードを用いるマスター／スレーブ構成と，全ノードが参照と更新が可能でキーごとに対応するノードを固定するクラスタ構成である．

しかし，マスター／スレーブ構成・クラスタ構成ともに分散 KVS への問い合わせ時，あるいは分散 KVS 間のデータ同期時にネットワークの遅延が発生するため，高速な応答性能や一貫性が要求される用途には向かないという欠点がある．

4 提案手法

図 1 に提案手法の概要を示す．提案手法では，KVS をコンテンツ配信アプリケーションと同一のノードに配置し，プロセス間通信でデータを交換することで，ネットワーク遅延を排した高速な応答を実現する．また，ロードバランサでユーザのコンテンツ配信アプリケーションへの接続を常に同一ノードに固定することで，KVS 間の同期を要さずにユーザに対するデータの一貫性を維持する．これは，パーソナライズされたコンテンツ配信においては，実時間で各ユーザから見たデータの一貫性が保たれていればよいという特性に着目しており，クライアント中心一貫性 [1] を確保していることに等しい．

また，耐障害性を高速に確保するため，提案手法ではマスター KVS と更新キューを配置する．分散 KVS の更新は一旦更新キューにエンキューされ，一定間隔でまとめてデキューされて，同一のキーへの書き込み処理があった場合に最新の値のみマスター KVS に書き込まれる．分散 KVS はキーの更新時刻を保持し，一定間隔でマスター KVS に問い合わせて，保持しているデータより新しいデータがある場合は値を更新する．

Building Low-latency Distributed KVS Optimized for Personalized Content Delivery
†Yoshiaki UEDA　‡Toshinori USUI
†Supership Inc./The Open University of Japan
‡Institute of Industrial Science, The University of Tokyo

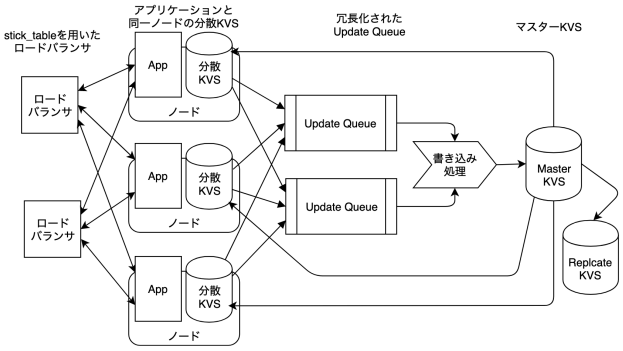

図 1: 提案手法の概要

5 評価

提案手法を評価するため，プロトタイプを実装した．図 1 の分散 KVS には memcached を採用し，テキストプロトコル互換の通信方式で通信させた．プロセス間通信には UNIX ドメインソケットを用いた．また，ロードバランサには HAProxy [2] を用い，ユーザの接続先ノードの固定にはその stick_table の機能を用いた．stick_table とは，接続毎の識別子において，前回の接続先ノードを記憶し，一定期間同一ノードに接続先を固定する機能である．ここではユーザごとに一意な UUID を生成して識別子とし，Cookie に保持させた．コンテンツ配信アプリケーションには，測定のため memtier_benchmark[3] を用いた．Update Queue は KeyDB[4] のリスト型を用いて実装した．Master KVS および Replcate KVS には KeyDB[4] を採用した．Update Queue から Master KVS への書き込み処理は Go 言語で実装し，1 分間隔とした．

実験環境を表 1 に示す．このマシン上に分散 KVS を構築し，応答時間と一貫性を評価する実験をそれぞれ実施した．

表 1: 実験環境

CPU	Intel Xeon E5-2620 @2.00GHz * 2
メモリ	48GB
OS	Linux Ubuntu 18.04.3 64-bit
ネットワーク	1Gbps イーサーネット

5.1 応答時間

提案手法および比較対象の既存手法である Redis の応答時間を評価する．この実験では，既存手法・提案手法ともに単一ノードで実験した．memtier_benchmark では，clients=1 threads=1000 requests=100 の引数を用いた．

実験の結果，提案手法と既存技術の Redis の平均応答時間はそれぞれ 0.72 ミリ秒，15.62 ミリ秒であり，提案手法による 10 ミリ秒以上の応答の高速化を確認した．また，既存技術では 2 章の応答時間の要件を満たせない一方，提案手法では要件を容易に充足できることも確認した．

5.2 一貫性

提案手法でユーザ単位のデータの一貫性を確認する．確認にはユーザ毎に異なる識別子を Cookie に保持させ，識別子ごとに何回のアクセスがあったかを分散 KVS に記録しレスポンスする実験用アプリケーションを作成した．100 並列の各 100 回のリクエストを行いクライアント側が計測しているリクエスト数と分散 KVS で計測された数を比較し一貫性の評価を行った．提案手法の分散 KVS と実験用アプリケーションを各 2 台を用意しロードバランサの配下に配置して負荷分散される環境を用意した．

実験の結果，クライアントがリクエストした回数とレスポンスの結果は常に一致しており，クライアント自身が書き込んだ値を読み出すことができる一貫性が正しく機能していることを確認した．

6 おわりに

本論文ではパーソナライズされたコンテンツ配信に用いるデータ特性を利用した低遅延な分散 KVS の構築手法を提案した．実験では応答速度の要件を満たし，既存技術と比較し応答速度が 10 ミリ秒以上高速化された．また，一貫性においてはクライアント中心一貫性 [1] が正しく確保できていることを確認した．ノードダウン時や入替時の一貫性の担保が今後の課題である．

参考文献

[1] Yuqing Zhu and Jianmin Wang. Client-centric consistency formalization and verification for system with large-scale distributed data storage. *Future Generation Computer Systems*, Vol. 26, pp. 1180–1188, 2010.

[2] HAProxy. The reliable, high performance tcp/http load balancer. http://www.haproxy.org/. (最終アクセス日: 2020-01-09).

[3] RedisLab. memtier_benchmark. https://github.com/RedisLabs/memtier_benchmark. (最終アクセス日: 2020-01-09).

[4] John Sully. Keydb. https://github.com/JohnSully/KeyDB. (最終アクセス日: 2020-01-09).