部分的なメモリ故障への耐性を有するインメモリデータベース

下村剛志[†]　山田浩史[†]

In-Memory Key-Value Store (In-Memory KVS) はデータを key と value の組で管理するデータベースの中で, 特に key, value を全てメモリ上に展開するものを指す. In-Memory KVS では管理しているデータがメモリ上に存在するので従来の HDD や SSD などのディスク装置を用いるデータベースよりも高い性能が実現できる. In-Memory である性質上, 大量のメモリを消費することが特徴で用途によっては TB 単位のメモリを使用することもある. 採用例としては実際に Amazon では DynamoDB[1] が, Facebook では memcached を用いて秒間 10 億を超えるリクエストを処理している[5].

メモリは使用しているとモジュール自体は正常であるにも関わらずビットが反転してしまうソフトエラーや, モジュール自体に異常が起き正常にデータを読み出すことができなくなるハードエラーが発生することがある[6][3]. メモリにエラーが発生し, 記録したデータを正しく呼び出すことができなくなればプログラムは正常に動作を続けることが困難になる. 前者のソフトエラーは誤り訂正符号 (ECC) を持ったメモリを用いてエラー発生時に適切に反転してしまったビットを訂正することである程度抑制することができるが, ハードエラーが発生すると該当する領域を用いていたアプリケーションは正しいメモリの内容を取得できなくなり, 予期せぬ動作をする可能性があるため, たとえ他のメモリページが正常だとしても OS に強制的に終了させられてしまう. 特に近年, 搭載メモリの大容量化やプロセスルールの微細化が進みハードエラーの発生確率が向上しており[6], 問題となっている.

特に In-Memory KVS では大量のメモリを使用することが多く, 多数の key, value を管理するほど他のアプリケーションに比べて相対的にメモリエラーに遭遇する確率が高くなってしまう. 更に, In-Memory KVS でアプリケーションの再起動を行うと, メモリ上に存在していた key や value などのデータ群すべてを失うため, その間のサービスの可用性が著しく低下することが知られている. また, In-Memory KVS を Hot な状態にするには時間がかかり, Facebook では元の性能を発揮するまでに 90 分以上要する[2]. In-Memory KVS 上にソフトウェア的に ECC を実装[4] することでメモリエラーへの耐性を向上させる研究もあるが対象はソフトエラーであり, メモリモジュールの一部が破損してしまうハードエラーには対応することができない.

本研究ではマスターデータが別のストレージに保存されており In-Memory KVS をキャッシュとして用いる場合を対象として, In-Memory KVS の一部のデータがメモリエラーで欠損してもアプリケーションの再起動の必要無しに動作を継続することができる機構を提案する. これにより In-Memory KVS の再起動によるダウンタイムや, メモリ上に展開されていたキャッシュがリセットされることによるスループットの低下を軽減させることが可能になる.

メモリエラーが起きても In-Memory KVS の動作を継続させるために, 故障したメモリページ上に存在したメモリオブジェクトを破棄し, その他のメモリオブジェクトを活用して動作を維持させる. 例えば, 一部の key, value のペアが消失したとしても, そのペアさえ取り除けば, In-Memory KVS としては動作可能である. これを実現するために, Linux Kernel と協調し In-Memory KVS が使用しているメモリ領域でエラーが発生した際にはプロセスをキルせずにエラー領域の仮想アドレスを In-Memory KVS に通知する. In-Memory KVS ではそのアドレスに対応する領域を検索し, その領域に保存されているデータの特性によって予め決めた破棄/再構築/再読み込みのいずれかを行い, エラーの起きた領域を修復する. 例えば実際のデータを保管している key, value は別の場所にマスターデータが存在するので破棄, key から value を検索する為に用いられる hash table は key のデータより再構築, 起動時からの統計情報はコピーを取得しておき再構築などが想定される. 本アプローチであると key, value の領域が破損してしまった場合にはアクセスできない要素が発生してしまうが, 本研究のターゲットである In-Memory KVS のキャッシュ用途での利用であればディスク装置から再度読み込みを行えばよい. 厳密にはディスクからの読み込みを行う分だけシステム全体としての性能は劣化してしまうが In-Memory KVS 全体を再起動するよりは低い性能劣化で済むと考えている.

現在, Linux 4.13.9 と memcached 1.4.39 を基盤と

† 東京農工大学
Tokyo University of Agriculture and Technology

して提案方式の実装を行っている．今後は, 実装を完成させ, 実際にメモリにエラーを混入させて性能劣化の度合いを測定する実験を実施する予定である．

参 考 文 献

1) G. DeCandia, D. Hastorun, M. Jampani, G.Kakulapati, A.Lakshman, A.Pilchin, S.Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon ’s Highly Available Key-value Store. In *Proc. of the 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP ’07)*, pages 205–220, 2007.

2) A. Goel, B. Chopra, C. Gerea, D. Matani, J. Metzler, F. U. Haq, and J. Wiener. Fast Database Restarts at Facebook. In *Proc. of the 2014 ACM SIGMOD Int ’l Conf. on Management of Data (SIGMOD ’14)*, pages 541–549, 2014.

3) A. A. Hwang, I. A. Stefanovici, and B. Schroeder. Cosmic Rays Don ’T Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design. In *Proc. of the17th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS ’12), year = 2012, pages = 111–122,*.

4) Y.Li, H.Wang, X.Zhao, H.Sun, and T.Zhang. Applying software-based memory error correction for in-memory key-value store: Case studies on memcached and ramcloud. In *Proceedings of the Second International Symposium on Memory Systems*, pages 268–278. ACM, 2016.

5) R.Nishtala, H.Fugal, S.Grimm, M.Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, et al. Scaling memcache at facebook. In *nsdi*, volume 13, pages 385–398, 2013.

6) B. Schroeder, E. Pinheiro, and W.-D. Weber. Dram errors in the wild: a large-scale field study. *Communications of the ACM*, 54(2):100–107, 2011.