ビッグメモリ環境におけるインメモリ KVS の定量的性能評価

庄 司 直 樹 † 山 田 浩 史 †

インメモリ Key-Value Store(KVS) はその性能の高さから様々な場面で広く利用されるようになってきた．インメモリ KVS では，主たる処理をメモリ上で行なうため，従来のディスクベースのデータベース管理システムよりも高い性能を発揮することができる．Redis や memcached，Masstree などがその代表的な例である．インメモリ KVS は今日の Web サービスには欠かせないコンポーネントの１つであり，SNS サービス[6] や検索サービス[1]，ショッピングサイト[2),8)] を構成する要素となっている．

インメモリ KVS が広く普及してきた一方で，近年のサーバマシンは数百ギガバイトからテラバイトサイズのメモリを搭載できるようになってきた．たとえば，HP ProLiant シリーズでは最大 6TB のメモリを搭載できる．また，不揮発性メモリの発達により，x86 がサポートしている 256 TiB の仮想アドレス空間よりも大きいサイズのメモリを搭載可能なマシンが登場すると言われている[4]．こうした巨大なメモリを搭載したマシンを用いれば，インメモリ KVS は大量のデータを扱うことができる．そのため，インメモリ KVS 単体の性能向上が期待できる一方で，現行のインメモリ KVS がそうした巨大なメモリを効率的に管理および利用できるか未だ不明な点である．

これまでインメモリ KVS の挙動を解析した研究がなされてきた．CloudSuite[3] では，memcached などの scale-out ワークロードに着目し，PERSEC などの CPU インテンシブなベンチマークとの挙動比較を行なっている．しかしながら，ビッグメモリ環境での挙動は比較されていない．Karakostas ら[5] の調査では scale-out ワークロードにとって MMU の性能がどのように影響しているかを定量的に示している．Huge page など，メモリの設定を変更しながら実験をしているが数百 GB クラスのメモリサイズでの実験は報告されていない．Park ら[7] の調査では，インメモリ分散処理フレームワークである Spark に着目し，様々なベンチーマークを NUMA マシン上で Huge page を用いながら稼動させ，その性能を解析している．本調査においても，数百 GB クラスのメモリサイズでの実験は報告されておらず，またインメモリ KVS は用いていない．

本研究では，インメモリ KVS を対象に巨大なメモリを搭載するマシン上での挙動調査および定量的な解析を行なう．この結果を大きいメモリを有するマシンを利用するインメモリ KVS の構成法の足がかりとする．インメモリ KVS が数百 GB のメモリを与えられることで大量のデータをメモリ上に展開して操作可能になる反面，大量のデータを管理することによる検索の非効率化や TLB ミスの多発など，性能向上を妨げる要因は多々あると考えられる．

本調査では，Redis，memcached，および Masstree を調査の対象とする．利用するマシンは 256 GB のメモリを搭載したマシンである．第一歩として，これらの KVS にセットする KV ペアの量を変更しながら，GET メソッドに必要となる時間を計測した．結果はいずれも数十 GB までのテーブルサイズであれば GET メソッドのレイテンシはほぼ同じであるものの，100 GB を超えたところでレイテンシが数倍にも増加した．特に Masstree の性能劣化は著しく，2 倍以上の性能劣化が観測された．

各 KVS の詳細な挙動を確認するために，現在パフォーマンスカウンタを取得しながら性能劣化の原因を探っている．今後は，パフォーマンスカウンタの結果を踏まえ，KVS それぞれの内部構成を確認し，性能ボトルネックの要因を探る予定である．

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

参 考 文 献

1) F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *Proc. of the 7th USENIX Symp. on Operating Systems Design and Implementation (OSDI '06)*, pages 205—218, 2006.
2) 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.
3) M. Ferdman, A. Adileh, O. Kocberber, S. Vo-

los, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In *Proc. of the 17th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS '12)*, pages 37–48, 2012.

4) I. E. Hajj, A. Merritt, G. Zellweger, D. Milojicic, R. Achermann, P. Faraboschi, W. mei Hwu, T. Roscoe, and K. Schwan. SpaceJMP: Programming with Multiple Virtual Address Spaces. In *Proc. of the 21st Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16)*, pages 353–368, 2016.
5) V. Karakostas, O. S. Unsal, M. Nemirovsky, A. Cristal, and M. Swift. Performance analysis of the memory management unit under scale-out workloads. In *Proc. of the 2014 IEEE Int'l Symp. on Workload Characterization (IISWC '14)*, pages 1–12, 2014.
6) R.Nishtala, H.Fugal, S.Grimm, M.Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani. Scaling Memcache at Facebook. In *Proc. of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13)*, pages 385–398, 2013.
7) J. Park, M. Han, and W. Baek. Quantifying the performance impact of large pages on in-memory big-data workloads. In *Proc. of the 2016 IEEE Int'l Symp. on Workload Characterization (IISWC '16)*, pages 1–10, 2016.
8) S. Sivasubramanian. Amazon dynamoDB: A Seamlessly Scalable Non-relational Database Service. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12*, pages 729–730, 2012.