ネットワークスタックと不揮発性メモリの統合による永続性Key-Value Storeの高速化

味曽野　雅史[†]　品川　高廣[†]

1. はじめに

計算機を構成するハードウェアには，CPU やメモリの他，ストレージやネットワークなどの I/O デバイスがある．従来は CPU の処理速度が圧倒的に高速であったため，I/O デバイスの性能を引き出すために複雑なソフトウェアスタックを用いてきた．しかし，10Gbps のネットワークインターフェイスカード（NIC）や不揮発性メモリ（NVM）[4] の登場で I/O デバイスが高速化したため，ソフトウェアで複雑な処理をおこなうと I/O デバイスの性能に追いつかなくなってしまう[1]．従って，ハードウェア性能を最大限に生かす為には，従来のソフトウェアスタックを再構成する必要がある．

本研究では，代表的な Web アプリケーションの一つである永続性 Key-Value Store (KVS）を対象とし，その高速化をおこなう新しいソフトウェア構造を提案する．KVS は多くの Web サービスにおいてバックエンド処理で利用されており[2),3)]，その処理の高速化はサービス全体の性能向上に寄与する．

従来の KVS を 10G NIC や NVM を搭載した環境で動作させると，(1) ネットワークスタックの処理，(2) ストレージスタックの処理，(3)CPU 負荷の集中の 3 点でボトルネックが発生する．(1) では，10G NIC で最小サイズのパケットを送受信すると，1 パケットあたり約 70ns しか処理時間がないため，従来の TCP/IP やソケットなどの処理を OS でおこなっていると，ハードウェア性能を引き出すことが難しい．(2) も同様に，メモリと同程度のアクセス速度がある NVM では，従来のファイルシステムを経由するとソフトウェアのオーバーヘッドが大きい．(3) については，最近の CPU はマルチコアであるため，コア間で適切に処理を分散することが重要であるが，従来は処理の内容に基づくコアの選択をおこなっていないため，コア間で処理が偏ったりキャッシュを有効活用できなかったりする．

† 東京大学
The University of Tokyo

本研究では，永続性 KVS に特化することでハードウェア性能を最大限引き出すことが出来るソフトウェア構造を提案する．(1) については，KVS に特化して TCP/IP などのプロトコル処理を簡略化し，ゼロコピーで NIC から直接ストレージに書き込めるネットワークスタックを構築する．(2) については，NIC から NVM に対して直接 DMA で書き込めるようなデータ構造を構築する．(3) については，Intel Flow Director を活用して，KVS クライアント側でキーに基いたコア選択をおこなって，処理を適切に分散できるようにする．

2. 提案手法

高速な永続性 KVS 実現のために，主に以下の 3 つの手法で KVS のソフトウェアスタックを構築する．

2.1 KVS に特化したネットワークスタック簡略化

ネットワークスタック処理での不要なデータコピーを削減するため，ユーザとカーネル空間で共有する巨大なメモリ空間を用意する．ネットワークスタック及びアプリケーションはこの共有領域を直接参照するようにする．また NIC の受信キューもこの共有メモリ領域上に用意するようにする．これにより無駄なデータコピーを削減する．今回は KVS がハードウェアを占有していると仮定しているので，この方式でも他のアプリケーションに影響はない．

また，KVS にとって不要なネットワークスタックの処理は省略する．具体的には，TCP/IP 通信でおこなわれる到着順が前後したパケットの並び替え処理や，分割したパケットの再構成処理等である．KVS サーバはバックエンドサーバに近い所に位置し，また大容量のデータをやりとりは少ないので，こうした問題はめったに発生しないと仮定した最適化が可能である．万が一発生したときだけ，例外処理として対処する．

2.2 KVS 固有の NVM 上のデータ構造の利用

NVM は DRAM と同様に NIC から DMA 可能な領域である．今回は動作するアプリケーションは永続性 KVS であり，データを保存する必要があるので，

NIC から直接 NVM へ DMA するようにする．ネットワークスタックやアプリケーションは NVM 上のデータを直接参照するようにする．これによりファイルシステムを経由したデータの書き込みがなくなり，メモリからディスクへのデータコピーも排除できる．また，KVS のデータ構造としてはパケットをそのまま保存するような形とする．データ送信時はこの NVM 上のパケットを直接 DMA して送信するようにする．

一部の Intel のプロセッサでは Intel Data Direct I/O Technology (DDIO)[5] の機能により，割り込みで受信したデータは直接プロセッサの LLC に配置される．したがって，この場合データを NVM 上に永続化させるには明示的なキャッシュフラッシュが必要になる．一方で，この機能を使用することで永続化する必要がないデータに対して DMA 先を NVM にしても速度低下の影響は低くなると考えられる．

2.3 キー空間ごとのコアの割り当て

キャッシュを有効活用した割り込みを処理を実施するには，パケットの割り込み時点でキー空間ごとに処理をコア間で分散させるべきである．Intel NIC の一部は Flow Director[6] によって IP アドレスやポート番号等の組から計算されるハッシュ値でパケット受信割り込みのコアをハードウェア的に分散させることができるが，パケットの中身まで見て割り込みを分散させる機能は存在しない．

提案手法では，クライアントと協調することで，キー空間ごとのコアの割り当てを実現する．具体的には，キー空間ごとにクライアントが送り先ポートを変えることでおこなう[7]．ポート番号を変えることで結果として Flow Director が計算するハッシュ値がキー空間ごとに変化する．クライアントはキーを知っており，また今回 KVS が NIC を占有しているためこのような処理が可能となる．

3. 関 連 研 究

ネットワークスタック最適化の研究としては IX[1] や mTCP[8] などがある．これらの研究では汎用的なネットワークスタックの高速化手法を提案しており，主な内容にカーネルとユーザ空間でのメモリの共有や，処理のバッチング，スケジューリングの改善等である．これらの手法は今回の KVS アプリケーションにも有用である．しかし IX や mTCP はストレージスタックや KVS 固有の特性は考慮していない．

ネットワークスタックと NVM の統合することでアプリケーションの高速化を図る研究として，PASTE[9] がある．PASTE では NIC が受信したパケットは DMA で NVM に直接転送される．したがって DMA をした段階でデータの永続化が完了する．アプリケーションは永続化された領域を後で参照することで必要な処理をおこなうことができる．PASTE はこれを一般的なトランザクション処理におけるログ先行書き込み部分の高速化をおこなっているが，KVS などのアプリケーション全体の最適化はおこなっていない．また，TCP/IP のプロトコルスタックはアプリケーション向けに最適化していない．

MICA[7] は揮発性 KVS の高速化に関する研究であり，クライアントと協調してキャッシュを有効活用する処理分散を実現している．また，高速なロギング実現のためのジャーナリングファイルシステムの構造を利用している．MICA ではネットワークスタック処理部分やデータの永続化は研究の対象外である．

4. まとめと今後の予定

本研究では NVM 上で高速な永続性 KVS を実現するためのソフトウェアスタック構築手法を提案した．提案方式では，ネットワークとストレージスタックの簡略化及びキー空間ごとのコア割り当てをおこなう．現在提案手法のプロトタイプの開発を進めている．

参 考 文 献

1) A. Belay, et al. IX: A Protected Dataplane Operating System for High Throughput and Low Latency. In *OSDI '14*, pp. 49–65, 2014.
2) Memcached. `http://www.memcached.org`.
3) Apache Cassandra. `http://cassandra.apache.org`.
4) M. Nanavati, et al. Non-volatile Storage. *Commun. ACM*, Vol. 59, No. 1, pp. 56–63, 2016.
5) Intel. Intel Data Direct I/O Technology. `http://www.intel.co.jp/content/www/jp/ja/io/data-direct-i-o-technology.html`.
6) Intel. Intel Ethernet Flow Director and Memcached Performance. `http://www.intel.co.jpcontent/www/jp/ja/ethernet-products/converged-network-adapters/ethernet-flow-director.html`.
7) H. Lim, et al. MICA: A Holistic Approach to Fast In-Memory Key-Value Storage. In *NSDI '14*, pp. 429–444, 2014.
8) E. Y. Jeong, et al. mTCP: A Highly Scalable User-level TCP Stack for Multicore Systems. In *NSDI '14*, pp. 489–502, 2014.
9) M. Honda, et al. PASTE: Network Stacks Must Integrate with NVMM Abstractions. In *HotNets '16*, 2016.