情報処理学会論文誌 Vol.56 No.2 525–533 (Feb. 2015)

推薦論文

ストレージ・ネットワークの直接転送エンジンを用いたSSDキャッシュサーバの提案

後藤 真孝[1,a)] 村上 瑛美[1] 秋山 晴彦[1] 村井 信哉[1]

受付日 2014年5月9日, 採録日 2014年11月10日

概要：本稿は，Web システム等で用いられるメモリキャッシュサーバを大容量化するため，SSD を記憶領域として用いた SSD キャッシュサーバについて述べる．ストレージ上のデータを，TCP/IP ネットワークに CPU 処理を介すことなく直接転送できるハードウェアエンジンを用い，低応答遅延の SSD キャッシュサーバを試作し評価した．メモリキャッシュサーバの概要，ハードウェアエンジンの概要について述べ，試作したキャッシュサーバの構造について述べる．また，実機評価について述べる．SSD を 4 台使用することにより，4 KB のサイズのデータ参照への応答を，1 秒間に 6 万 5 千回実施することができた．また，そのときの応答時間は，約 500 μ 秒であった．

キーワード：memcached，SSD，TOE，ハードウェアエンジン，FPGA

Implementation of the SSD Cache Server Using the Direct Transfer HW Engine from Storage to Network

Masataka Goto[1,a)] Eimi Murakami[1] Haruhiko Akiyama[1] Shinya Murai[1]

Received: May 9, 2014, Accepted: November 10, 2014

Abstract: In this document, we describe the low-latency SSD cache server which uses the direct transfer HW engine from storage to a network. We assume that it can be used as a large capacity memory cache server in many Web service systems. A direct transfer HW engine can send data on SSD to the client host via TCP/IP network without TCP/IP software protocol stack processing by CPU. And, we are developing the SSD cache server application program which uses such direct transfer HW engine effectively. We introduce the generic memory cache server and the structure of a HW engine and we describe the structure of SSD cache server using it. And, we show the performance our SSD cache server. Using 4 SSD drives, it can respond 65,000 times per second of 4 KB data to GET requests from clients. And, the average time of response time is about 500 microseconds.

Keywords: memcached, SSD, TOE, HW Engine, FPGA

1. はじめに

近年，Web サービスは日常生活に欠かせないものとなっている．以前は家から PC で利用する場合が多かったが，スマートフォン等のモバイル機器からの利用が爆発的に増えた．その結果，人気の高い Web サービスには，大量のアクセスが集中する．アクセス集中も一過性だけではなく，SNS のように，膨大な数のユーザが情報を発信し参照することが定常的に行われている．

Web サービスの応答性には様々な要因がある．その 1 つに，データベース（DB）のデータ参照の遅延がある．発信された情報は，内容の一貫性を維持するために DB に登録され，ユーザによる情報参照時には，DB からデータの読み出しが行われる．この情報の登録と参照の繰り返しが，

[1] 株式会社東芝
TOSHIBA Corporation, Yokohama, Kanagawa 247–8585, Japan
[a)] masataka2.goto@toshiba.co.jp

本稿の内容は 2013 年 10 月のマルチメディア通信と分散処理研究発表会で報告され，同研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

Web サービスの応答性に与える影響は大きい．

DB は，大容量化を実現するために，HDD 等の低コストであるストレージの利用が一般的である．しかし，上述のように高い応答性能を実現するためには，HDD では不十分であり，DRAM 等のメインメモリを活用する高速化手法が開発され続けてきた．しかしながら，近年のように膨大なユーザが利用する Web サービスに関しては，ユーザの利便性の要求に応えられる高い応答性能を，DB のみで実現するのは難しくなってきている．

そこで，この問題を解消する方法として，メモリキャッシュサーバが利用されるようになってきた．メモリキャッシュサーバは，高速なデータ参照を実現する方法の 1 つである．キーバリューストア（KVS）と呼ばれる，単純な操作を前提とし，DRAM 等のメインメモリのみを用いてデータを記憶するサーバである [1]．DRAM の利用が前提で，単純な操作のみ対応しているため，従来の DB に比べ応答性能が非常に高い．メモリキャッシュサーバの処理を FPGA でハード化し，ソフトウェアの実装に比較して広帯域な KVS サーバの検討も進められている [2], [3]．

今般のデータセンタでは，参照頻度が非常に高いデータをメモリキャッシュサーバで保持し，Web サーバが動的に画面を構築する際にそれらのデータを参照するシステム構成が一般的である．特に近年は SNS 等のサービスのように，頻繁に更新される膨大なデータから動的に画面を構築する Web サービスが増えており，メモリキャッシュサーバの大容量化の要求も高まっている．そこで，記憶領域として記憶密度の高いフラッシュメモリも利用する方式の検討がさかんである [4], [5], [6]．さらに KVS をストレージシステムの基盤とし，上位に DB やファイルシステムを搭載する検討も進められており，高速な KVS への需要は高まっている [7], [8], [9], [10]．

本稿では，我々が試作し評価した，ストレージ上のデータを，CPU 処理を介すことなく直接ネットワーク送出できるハードウェアエンジンを適用した，SSD キャッシュサーバについて述べる．2 章ではメモリキャッシュサーバの概要を述べ，3 章では直接転送エンジンの概要と提案するキャッシュサーバの設計について述べる．4 章で今回の試作の評価について述べ，5 章で考察する．

2. メモリキャッシュサーバの概要と課題

2.1 メモリキャッシュサーバ

本稿では，キー・バリュー型の DB のうち，データの記憶領域にメインメモリを使用しているものをメモリキャッシュサーバと呼ぶ．代表的なメモリキャッシュサーバには，memcached [1] がある．memcached は，ブログサービスや SNS サービスのユーザへの応答性能の向上を目的として，頻繁に参照されるデータを高速に読み書きするために用いられる．

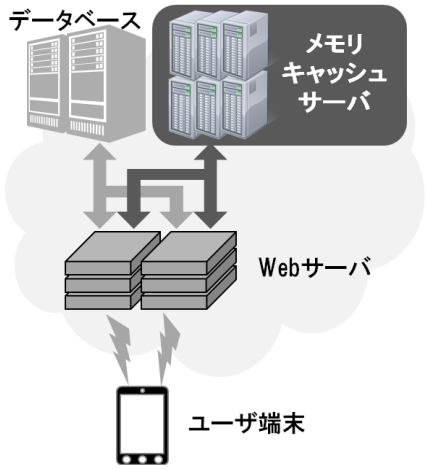


図 1 Web サービスシステムの例
Fig. 1 Web service system.

表 1 データ読み書きのコマンド
Table 1 Commands of memcached protocol.

コマンド名	処理内容
ADD	新規に value を格納する
SET	value を格納する
GET	value を参照する
DELETE	value を削除する
GETS	リビジョンチェック付き GET コマンド
CAS	リビジョン指定付き SET コマンド
REPLACE	value を書きかえる
APPEND	value を追記する
PREPEND	value を挿入する
INCR	value に算術加算する
DECR	value から算術減算する

図 1 に，メモリキャッシュサーバを用いた Web サービスシステムの例を示す．トランザクションを必要とするデータの一貫性や完全性の保証は，DB により担保されており，特にユーザの操作性に関わるようなデータをメモリキャッシュサーバに保存するような使用法で，DB システムの一翼を担う．

次に，メモリキャッシュサーバの通信プロトコルについて述べる．キー・バリュー型とは，保持したいデータ（Value）と，それを一意に識別する識別子（key）を一対で記憶する DB である．key を指定して value を保存するセット操作，key を指定して保存した value を読み出すゲット操作が基本となる．一般的な DB にある，パターンマッチングでの読み出し操作や，複数のデータを不可分操作でいっせいに更新するような機能は有さない場合が多い．

表 1 に，memcached プロトコル [11] で使用できるデータ読み書きに関するコマンド一覧を示す．これらのコマンドは，クライアントからサーバに送信される．図 2 に，GET コマンドについての，クライアントとサーバのメッ

セージのやり取りの例を示す．クライアントは，key1 という key のデータの参照を要求している．サーバは，key1 という key で記憶した value である abcdefghij を応答している．memcached プロトコルにおけるサーバからの応答は，コマンドごとに応答が異なる．GET コマンドへの応答は，問い合わせを受けた key を保持している場合のみ，VALUE で始まるメッセージを応答する．GET コマンドへの応答の最後は，END で締めくくる．

2.2 メモリキャッシュサーバの課題

メモリキャッシュサーバは，低遅延での応答を保証するためにメインメモリ上にデータを保持する．このため，大容量化を行うにはコストが高いという問題がある．また，サーバ 1 台あたりのメインメモリ量にも限界があるため，数テラバイト級の記憶容量を必要とする場合には，複数台のサーバを用いたシステムとならざるを得ず，ランニングコストも無視できない．

一方，SSD 等のストレージの中でも比較的高速なものでデータを保持すれば，低コストで大容量化を行うことができる．SSD は，DRAM と比較するとストレージ単価は数分の一であり [12]，大容量メモリキャッシュサーバの低コスト化に SSD は有効である．しかし，高頻度のデータ参照が行われる状況では，CPU によるストレージアクセス処理とネットワークのソフトウェア処理をともなうため，DRAM に比べ，安定的に応答時間を短く保証することが困難である．

公開されているアクセス統計情報によると [13]，世界規模の Web サービスの 1 つであるフリー百科事典では，1 秒間に 2,000 回から 3,000 回のアクセスが行われている[*1]．1 回のページの表示に，20 回から 30 回の memcached アクセスが行われるとすると，memcached サーバは，1 秒間に約 10 万回のアクセスに応答する能力が必要である．また，Web ページが表示されるまでの遅延を考慮すると，memcached サーバの応答性能は，1 ミリ秒から 2 ミリ秒程度に抑えられておくことが必要である．

我々は, SSD 等のストレージ上のデータを, CPU 処理を介さずに TCP/IP ネットワークへ送出することができるハードウェアオフロードエンジンを開発している [14], [15], [16]．このエンジンを Web サービスシステムのメモリキャッシュサーバへ適用し，ストレージアクセスとネットワーク処理を一括でハードウェア処理することで，安定した低応答遅延を可能とする，SSD を記憶領域とする低応答遅延キャッシュサーバの実現を検討している．コンシューマ用途の SSD を用い，1 秒間に数十万回の高頻度アクセスが行われている状況で，1，2 ミリ秒程度の応答時間の実現を目指している．

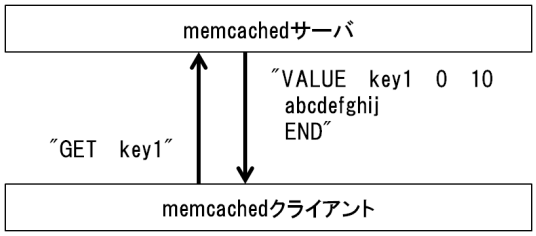


図 2 GET コマンドのやり取りの例
Fig. 2 GET command transaction.

[*1] 2014/6 の統計では，英語版ページに 1 時間あたり 9,145,657,338 回のアクセスがあった．

3. 直接転送エンジン SSD キャッシュサーバ

3.1 直接転送エンジンの概要

本節では，ストレージ上のデータを TCP/IP ネットワークに直接転送するハードウェアエンジンについて述べる．**図 3** に，直接転送エンジンの構成を示す．直接転送エンジンとは，ストレージインタフェース（I/F）部，ネットワークインタフェース（NIC）のほかに，ストレージアクセス処理部と TCP オフローディングエンジンを持つ，ハードウェアオフロードエンジンである．CPU やメモリといったホストシステムとは，汎用バスで接続される．ストレージアクセス処理部は，ストレージ I/F 部を介して SSD からデータを読み出し，TCP オフローディングエンジンにデータを渡す処理を行う．TCP オフローディングエンジンは，ストレージアクセス処理部から受け取ったデータを，TCP セグメントとしてネットワークに送出する処理を行う．これらの処理部により，CPU によるソフトウェア処理を介さないため，高速，広帯域でデータの送出が行える．

また，ストレージアクセス処理部は，ホストシステムの汎用ストレージ I/F としての機能も有する．CPU 上で動作する OS 等のソフトウェアから通常のストレージとして SSD にアクセスできるためである．Value を記憶するための SSD へのデータ書き込みは，CPU 上のソフトウェアから行う．

同様に，TCP オフローディングエンジンは，ホストシス

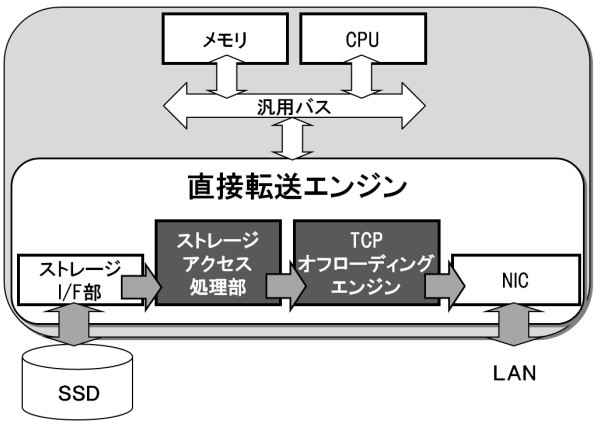


図 3 直接転送エンジンの構成
Fig. 3 Structure of direct transfer hardware engine.

表 2 直接転送エンジンの実装規模
Table 2 The number of logic module.

モジュール	論理ユニット数
ストレージ I/F （SATA 3.0 の 8 ポート構成）	約 3 万
ストレージアクセス処理部	約 1 万 5 千
TCP オフローディングエンジン	約 1 万 1 千
NIC （10GBASE-SR の 4 ポート構成)	約 6 千
FIFO	約 3 万 2 千

テムの TCP/IP プロトコルスタックの処理をオフローディングする機能も有する．ストレージアクセス処理部と同様に，CPU 上で動作する OS 等のソフトウェアから通常のネットワーク I/F として，データ通信を行い，キャッシュプロトコルのコマンド受信等を行うためである．

直接転送エンジンは，FPGA を用いて試作した．ストレージ I/F には，汎用の SSD を使用できるように，SATA 3.0 を採用した．また，NIC は，10GBASE-SR を，CPU との汎用バスは，PCI EXPRESS® [*2]の Generation2 を 4 レーン使用した．ストレージアクセス処理部，および，TCP オフローディングエンジンは，200 MHz の 128 bit 幅内部バスにより接続されており，FPGA 内部でのデータ転送のボトルネックはない．参考として，我々が試作した直接転送エンジンの実装規模を**表 2** に示す．

3.2 直接転送エンジン適用 SSD キャッシュサーバ

本節では，SSD を value の記憶領域として用いた SSD キャッシュサーバに直接転送エンジンを適用したシステムの構成について述べる．**図 4** に SSD キャッシュサーバの構成を示す．キャッシュサーバプログラムは，メインメモリ上に，key と value の対応を示すインデックス表を持つ．インデックス表の各エントリは，key の文字列，SSD 上の value の位置と長さを持つ．value の位置は，SSD のブロック位置で保持する方法やファイルとして保持する方法がある．直接転送エンジンの API に合わせて，適した形式で保持する．

我々の設計では，value を SSD 上のファイルに保持することとした．また，直接転送エンジンの API には，sendfile システムコールを用いた．sendfile システムコールは，Linux® に標準的に実装されている．引数で渡される，「ファイルディスクリプタ，オフセット，データ長」で指定されたデータを，「ソケットディスクリプタ」で指定される TCP セッションに送出する API である．この sendfile システムコールを変更し，直接転送エンジンの直接転送機能の呼び出しに利用した．

[*2] なお，Intel, Intel Core は，アメリカ合衆国および/またはその他の国における Intel Corporation の商標です．Linux は，Linus Torvalds 氏の日本およびその他の国における登録商標または商標です．Ethernet は，富士ゼロックス社の日本における登録商標です．PCI EXPRESS は，Peripheral Component Interconnect Special Interest Group の商標です．また，その他の本稿に掲載の商品，機能等の名称は，それぞれ各社が商標として使用している場合があります．

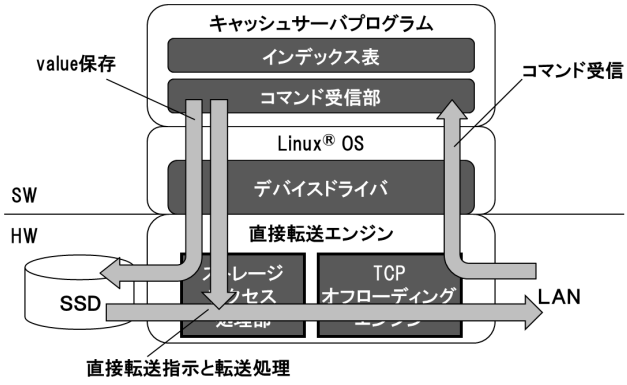


図 4 SSD キャッシュサーバの構成
Fig. 4 Structure of SSD cache server.

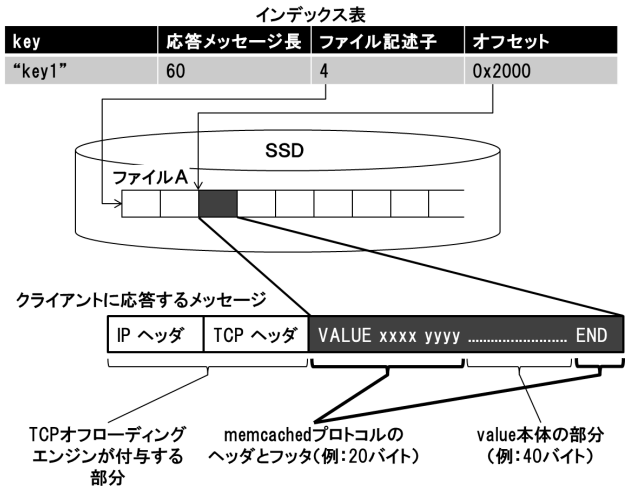


図 5 key と value の保持方法の例
Fig. 5 Data structure and relation between key and value.

3.3 SSD キャッシュサーバのデータ保持方法

図 5 に key と value の保持方法を具体的に示す．インデックス表には，key の文字列，応答メッセージ長（バイト長），value を保持しているファイルのファイル記述子，ファイル先頭からのオフセットをエントリとして保持している．ファイル上には，value のデータ列だけではなく，memcached プロトコルで規定されているヘッダとフッタを含めて記録する．つまり，GET コマンドの応答の形式である，VALUE から始まり END で終了する，応答メッセージそのものをファイル上に記述ことで，クライアントからの GET コマンドに対し，key が一致するインデックスエントリのファイル記述子，オフセット，応答メッセージ長を引数に含んだ sendfile システムコールのみで，応答を完了できるようにするためである．TCP ヘッダと IP ヘッダは直接転送エンジンの TCP オフローディングエンジン

が付与する．

ファイル上の応答メッセージは，アクセス単位でアラインメントされる．これにより，直接転送エンジンのストレージアクセス処理部が SSD からデータを読み出した冒頭から TCP オフローディングエンジンを通じて応答が行えるため，応答遅延を抑えることができる．ストレージアクセス処理部は，読み出し単位を 4 KB として実装したため，応答メッセージは 4 KB でアラインメントすることとした．その結果，応答メッセージ長に比べ，アラインメントの単位が大きい場合は，記憶用ファイルは内部断片化を起こすことになる．応答時間とストレージ効率のトレードオフであり，今回は応答時間に有利になるようにアラインメントを優先した．

3.4 SSD キャッシュサーバのコマンド処理

クライアントからのコマンドは，TCP オフローディングエンジンを通じて，コマンド受信部で受信し，解釈される．まず，メモリキャッシュサーバの主要な操作である SET コマンドの処理と，GET コマンドの処理について述べる．

SET コマンドを受信すると，SSD 上の記憶場所を割り当てる．記憶場所の割り当ては，先に述べた 4 KB アラインメントに基づいて割り当てられる．次に，SET コマンド中に含まれる value を取り出し，memcached の GET コマンドの応答メッセージを生成し，そのメッセージを割り当てた記憶場所に保存する．次に，key の文字列と一致するエントリをインデックス表から検索する．既存のエントリがない場合は，新規のエントリを割り当てて，key の文字列，value の位置，長さを記憶する．該当するエントリがすでにある場合は，該当する key に関してインデックス表の更新を行う．最後に，クライアントへ SET コマンドが正常に完了したことを知らせるために，STORED を応答する．

次に GET コマンドについて述べる．GET コマンドを受信すると，コマンドで指定された key でインデックス表を検索し，該当するエントリを得る．エントリに記憶されている value の位置，長さを引数に含め，sendfile システムコールを発行し，直接転送エンジンの直接転送機能を呼び出す．

以降で，その他のコマンドについて概略を述べる．INC コマンド，DEC コマンドは，value に対して算術処理を要するため，SSD 上の value をメモリに読み出し，計算を行った後，SSD に書き戻す．同様に APPEND コマンド，PREPEND コマンドも，いったんメモリに読み出し，value の追加や挿入を行った後，SSD に書き戻す．GETS コマンドや CAS コマンドは，リビジョンの一致性を確認する処理を加える以外は，GET コマンド，および，SET コマンドと同様に実現できる．ADD コマンド，REPLACE コマンドは，インデックス表の検索により，重複するエントリの有無によってエラーを判断する処理を加え，SET コマンドと同様に処理する．DELETE コマンドは，インデックス表から該当するエントリを削除して対応する．削除時には，該当する記憶場所を未使用場所として管理し，SET コマンドや，ADD コマンドで再利用できるようにする．ファイルとして保持している場合は，ファイルの削除をともなう．その他のコマンドは，通常のメモリキャッシュサーバと同様に実現する．

4. 評価

4.1 評価環境

試作した SSD キャッシュサーバの性能評価を行った．**表 3** に評価で使用した SSD キャッシュサーバのスペックを示す．SSD ドライブは 1 台，2 台，または，4 台使用した．2 台，または，4 台の場合は，RAID0 を構成し，一方の SSD にアクセス負荷が集中しないようにした．**図 6** に評価環境の機器構成を示す．あらかじめ，4 KB の 1,000 万件のデータを SSD キャッシュサーバに登録しておき，負荷クライアントから GET コマンドを休みなく送信する．GET コマンドで参照する key は，登録された key の中からランダムに選択しており，参照ミスはない．また，その最中に，応答時間測定クライアントから，3 秒間隔でランダムな key の GET コマンドを送信し，応答を受信完了するまでの時間を測定した．

4.2 測定結果

図 7 に，SSD ドライブ数と 1 秒間の応答回数（QPS）の測定結果を示す．QPS は，負荷クライアントから休みなく送信される GET コマンドに対して，試作した SSD キャッ

表 3 SSD キャッシュサーバスペック
Table 3 Server PC spec.

項目	内容
CPU	Intel® Core™ i7(3.5GHz)
メインメモリ	16GB
OS	Linux® 2.6.32
SSD ドライブ	SSDN-3T240B
SATA IF	SATA3.0 6Gbps (直接転送エンジン搭載)
NIC	10Gigabit Ethernet®(直接転送エンジン搭載)

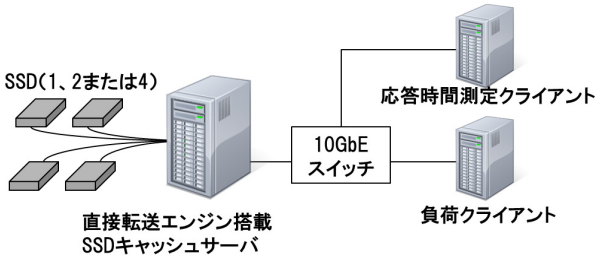


図 6 評価環境の機器構成
Fig. 6 Network topology for evaluation.

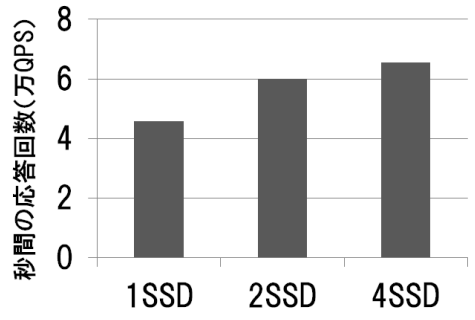


図 7 SSD ドライブ数と応答回数（QPS）の関係
Fig. 7 The number of SSD drives vs. QPS (10 k).

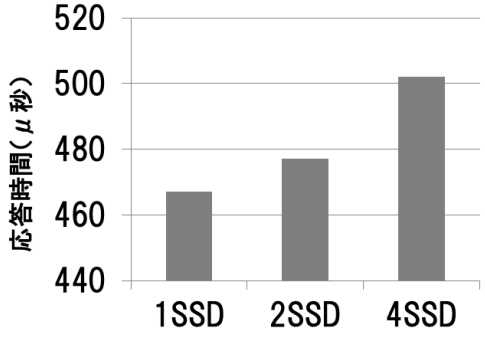


図 8 SSD ドライブ数と応答時間の関係
Fig. 8 The number of SSD drives vs. response time (usec).

シュサーバが一秒間に応答した回数である．SSD ドライブ 1 台では 4 万 5 千回，2 台では 6 万回，4 台では 6 万 5 千回であった．

次に，**図 8** に応答時間の測定結果を示す．応答時間は，負荷クライアントから休みなく送信される GET コマンドに対して応答中のサーバに対し，応答時間測定クライアントから GET 要求を送信した時刻から，4 KB の応答がすべて受信されるまでの時間である．図 7 に示したアクセス負荷がかかっているときの応答時間を測定した．20 回の平均時間を示している．SSD ドライブ 1 台では 467 μ 秒，2 台では 477 μ 秒，4 台では 502 μ 秒であった．標準偏差はそれぞれ SSD ドライブ 1 台では 85.0 μ 秒，2 台では 35.6 μ 秒，4 台では 38.0 μ 秒であった．

最後に，**図 9** に QPS と応答時間の関係を示す．比較のため，従来の DRAM 使用のメモリキャッシュサーバの応答時間も同様に測定し，併記した．測定で用いた負荷クライアントは，図 7 および図 8 の測定で使用したものを，負荷パラメータである QPS を加減するため，GET コマンドの送信間隔の待ち時間を可変にして用いた．結果で示している QPS は，サーバが実際に応答した QPS であるため，パラメータが等間隔の測定結果とはなっていない．また，DRAM 使用のメモリキャッシュサーバの測定は，傾向を把握する目的のためサンプリング数は最低限となっている．直接転送エンジン搭載 SSD キャッシュサーバは，SSD ドライブを 4 台使用した場合の結果である．

DRAM 使用のメモリキャッシュサーバは，測定した範囲

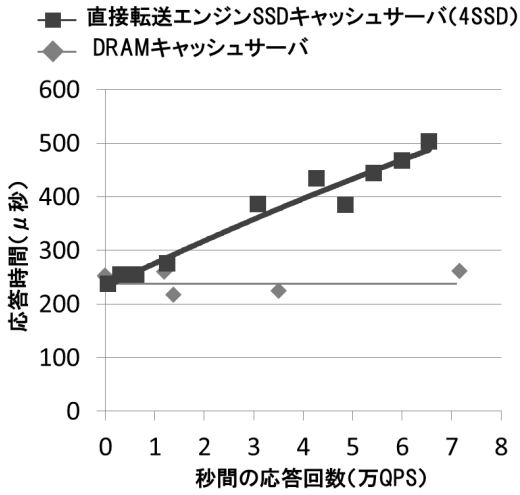


図 9 応答回数（QPS）と応答時間の関係
Fig. 9 QPS (10 k) vs. response time (usec).

では QPS に依らず，応答時間はほぼ一定であった．一方，直接転送エンジン搭載 SSD キャッシュサーバは，QPS の上昇とともに，応答時間も延びていく結果となっている．

5. 考察

5.1 応答回数

図 7 より，SSD のドライブ数の増加とともに，1 秒間の応答回数（QPS）が増加している．SATA には，データの読み出し等のコマンドをキューイングして受付ける NCQ（Native Command Queueing）という機能がある．この QPS の増加，すなわち，スループットの増加は，この NCQ の効果が表れていると考えられる．IO ベンチマークの結果によると，今回の評価で使用した SSD は，4 KB のランダム読み出しの秒間 IO 回数性能は，4 万 9 千という情報がある．SSD が 1 台の場合の結果は，SSD のランダム読み出し性能の約 92%の性能を引き出している．

しかし，SSD のドライブ数の増加の割合に比べ，QPS の増加の割合は低い．これは，ドライブの増加により NCQ のキューの容量が相対的に増加している一方で，SSD からのデータ応答以外の部分の処理時間の影響が増大していることが考えられる．今回の評価はデータサイズが 4 KB のため，1 秒間に 6 万回の応答であっても，TCP/IP の送信データは 2 Gbps 強にとどまる．評価環境の LAN の通信帯域である 10 Gbps に比べ十分小さく，ネットワークがボトルネックとなっているとは考えにくい．キャッシュサーバプログラムが，GET コマンドを解釈するたびに，直接転送指示を行っているが，この一連の処理や，直接転送エンジンのデバイスドライバによる割り込み処理がボトルネックとなっている可能性がある．さらなる高性能を実現するには，より詳細な分析が必要である．

5.2 応答時間

図 8 より，SSD のドライブ数の増加とともに，応答時間

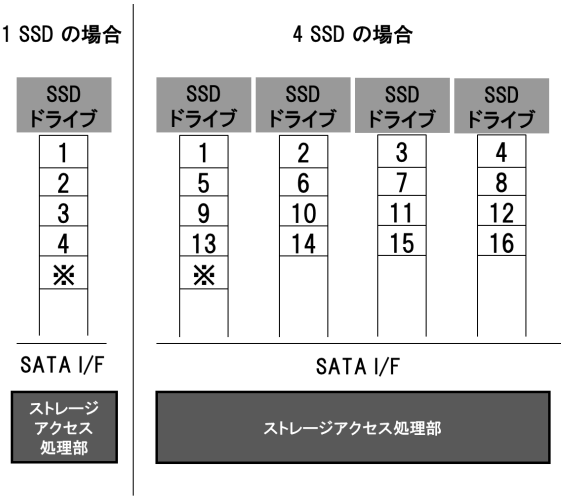


図 10 高負荷時の NCQ のコマンドキューイングの状態（推定）
Fig. 10 NCQ command queuing condition in high-load (supposition).

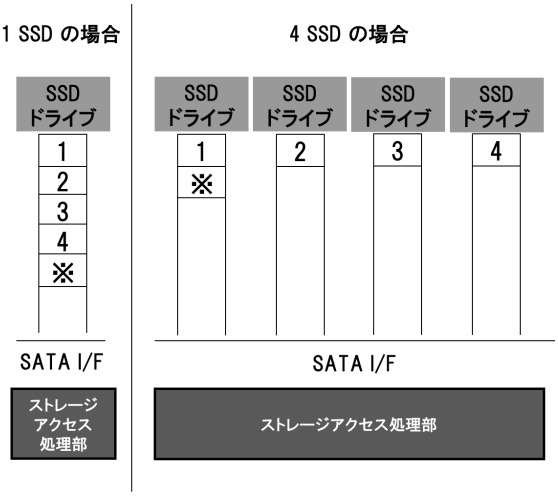


図 11 約 4 万 QPS の場合のキューイングの状態（推定）
Fig. 11 NCQ command queuing condition in 40 k QPS load (supposition).

が増加している．これは，SSD のドライブ数の増加が直接影響しているわけではないと思われる．ドライブ数の増加により総量が増加した NCQ が，負荷クライアントからの高頻度の GET 要求で埋まり，応答時間測定の要求に関する SSD アクセス要求が，総量が増加した NCQ の最後尾に配置されるため，SSD からの応答待ち時間が延びるためであると考えられる．その様子を**図 10** に示す．

今回の評価で最高の負荷をかけた際，各 SSD ドライブの NCQ のコマンドのキューイング状態は，図 10 に示すような状態であると考えられる．図 10 には，SSD ドライブ数が 1 台の場合と SSD ドライブ数が 4 台の場合の双方についてのイメージを示している．数字は，発行されたコマンドの順番を便宜的に示している．SSD ドライブ数が 1 台の場合は，1 つのキューに，順々にコマンドがキューイングされる．図では，1 つのドライブのキューの長さを 5 で示しているが，実際のドライブでは，32 までキューイングできるものもある．SSD ドライブ数が 4 台の場合は，RAID0 のストライピングを構成している関係で，コマンドの順は，SSD ドライブを横断するようになる．実際は，クライアントからの GET コマンド要求に基づいて sendfile システムコールが発行され，直接転送エンジンのストレージアクセス処理部からの SATA の読み出しコマンドが，各 SSD ドライブへキューイングされる限り詰め込まれるため，sendfile の引数で示されるファイルの位置によってコマンドが発行される SSD が定まる．このため，図 10 で示すようにすべてのドライブを横断して順々にはコマンドが並ぶわけではないが，多数のクライアントから，ランダムな key にアクセスがあると仮定すると，それぞれの SSD ドライブのキューは埋まり，図 10 と同等の状態になると考えられる．

SSD ドライブ数が 4 台の場合は，SSD ドライブ数が 1 台の場合に比べ，単純にコマンドをバッファできる数が多くなる．今回の評価のように，応答時間測定の読み出しを別途行っている場合は，図中の※に示したように，その他の高負荷アクセスによるコマンドで埋まったキューの最後尾に置かれるため，応答時間が延びていると考えられる．

これを考慮して図 9 を見ると，SSD のドライブ数が 4 台であっても，約 4 万 QPS のアクセス負荷では，400〜450 μ 秒の応答時間であり，SSD ドライブ数が 1 台の場合の約 4 万 QPS で応答時間 467 μ 秒と同等の応答時間であることが分かる．この場合の NCQ のコマンドのキューイングの状態は，**図 11** のような状態であると考えられる．このように，応答時間は，SSD のドライブ数によらず，秒間の QPS に依存しており，NCQ にキューイングされた待ちコマンド数の傾向であることが分かる．一方で，応答時間の標準偏差から，ドライブ数が 1 台より，2 台または 4 台の方が，安定した応答時間を得られていることも分かる．

以上の応答時間の振舞いは，我々の試作の 2 つの要因が関係している．1 つ目は，直接転送エンジンのストレージアクセス処理部は，SSD ドライブへ発行した順にコマンドの応答を受け取る仕様となっていることである．このため，キューイングされたコマンドの数に依存して，応答時間が延びる．しかし，仮に，ストレージアクセス処理部が，コマンドの発行順ではなくドライブからの応答順に処理を行うように設計したとしても，TCP/IP のストリーミングプロトコルの特徴のため，ネットワークの送出までにどこかでシリアライズされる必要がある．このため，応答時間は，キューイングされたコマンド数に依存する傾向は残ると思われる．

2 つ目は，我々の試作の応答時間のボトルネックが，SSD のアクセス性能ではなく，直接転送エンジン内のシリアライズされた処理内部にあることである．仮に SSD のみに

ボトルネックがある場合は，各 SSD ドライブのアクセス時間が応答時間に対応し，図 7 の示す結果が SSD の台数に比例して QPS は伸びていくはずである．その場合は，SSD のコマンドの処理遅延は QPS の逆数となり，図 8 は SSD 台数によらず応答時間は一定になるはずである．つまり，さらに SSD の台数を増やし，QPS の増加を狙うためには，直接転送エンジン内のシリアライズされた処理内部のボトルネックの解消が必要である．

最後に，図 9 から，DRAM 版との応答時間の比較を考える．低い QPS では，DRAM 版と比較しても応答時間に大きな差はない．つまり，直接転送エンジンを用いれば，SSD からデータを読み出してネットワークに送出するまでの時間は，DRAM を用いたメモリキャッシュサーバに匹敵するポテンシャルがある．つまり，上述の処理内部のボトルネックの解消がなされれば，SSD ドライブ数を増加させることで，高い QPS になった場合でも NCQ による応答時間の増大を考慮しても，1 秒間に数十万回以上の高頻度アクセスが行われている状況で，1，2 ミリ秒程度の応答時間の実現は可能であると考えられる．

6. おわりに

本稿では，Web サービスシステムのメモリキャッシュサーバの大容量化を実現する，低応答遅延の SSD キャッシュサーバを提案した．ストレージ上のデータを TCP/IP ネットワークに直接転送するハードウェアエンジンを適用し，SSD 上にデータを保持するキャッシュサーバプログラムを試作した．SSD 上のデータは，memcached プロトコルの応答メッセージの形式となるように，ヘッダとフッタを含めた形で保持する．また，データは，ファイル内を SSD の読み出し単位でアラインメントして保持することで，SSD からの読み出し遅延が最小になるようにした．

評価により，SSD を 4 台使用する場合において，1 秒間の GET コマンドの応答回数は約 6 万 5 千回，応答時間は約 500 μ 秒を実現できることを示した．SSD ドライブのコマンドキューイングの状況を考察し，応答時間がキューイングされたコマンド数に依存していることを示した．また，今回の試作では，直接転送エンジン，あるいは，デバイスドライバに処理性能上のボトルネックがあり，秒間の応答回数が SSD のアクセス性能を十分に発揮するレベルに達していないことが分かった．ただし，低い秒間の応答回数では，DRAM 版との応答時間に大きな差はなく，本提案手法が，応答時間においては DRAM を用いたメモリキャッシュサーバに匹敵するポテンシャルがあることを示した．

一方，本稿の評価はデータサイズを 4 KB に固定し，GET コマンドの QPS と応答時間の観点のみとなっている．一般的にメモリキャッシュサーバは，数十バイト程度の長さのデータを大量に保持する用途もある．SSD からの読み出し単位未満の長さのデータの扱いについても，本方式の実効性の確認が必要である．また，データの算術処理，PUT コマンド等のデータ書き込みを併用した場合の性能についても，明らかにする必要がある．

今後は，ボトルネックを解消し 1 秒間に数十万回の高頻度アクセスを実現し，実際のサービスへの適用性，実用性を評価する．

参考文献

[1] memcached, available from ⟨http://memcached.org⟩ (accessed 2013-09-16).

[2] Blott, M., Karras, K., Liu, L., Vissers, K., Bar, J. and Istvan, Z.: Achieving 10 Gbps Line-rate Key-value Stores with FPGAs, *Presented as part of the 5th USENIX Workshop on Hot Topics in Cloud Computing* (June 2013).

[3] Chalamalasetti, S.R., Lim, K., Wright, M., Young, A.A., Ranganathan, P. and Margala, M.: An FPGA memcached appliance, *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (Feb. 2013).

[4] Debnath, B., Sengupta, S. and Li, J.: FlashStore: High throughput persistent key-value store, *Proc. VLDB Endowment*, Vol.3, No.1-2 (Sep. 2010).

[5] Debnath, B., Sengupta, S. and Li, J.: SkimpyStash: RAM space skimpy key-value store on flash-based storage, *Proc. 2011 ACM SIGMOD International Conference on Management of Data* (June 2011).

[6] Lim, H., Fan, B., Andersen, D.G. and Kaminsky, M.: SILT: A memory-efficient, high-performance key-value store, *Proc. 23rd ACM Symposium on Operating Systems Principles* (Oct. 2011).

[7] AlchemyDB, available from ⟨https://code.google.com/p/alchemydatabase/⟩ (accessed 2013-05-09).

[8] InfoFrame Relational Store, available from ⟨http://jpn.nec.com/infoframe/relationalstore/⟩ (accessed 2013-05-09).

[9] A File-System Friendly Key Value Store, available from ⟨http://www.slideshare.net/MaxiScale/cloud-lab-kv02⟩ (accessed 2013-05-09).

[10] Trinity, available from ⟨http://research.microsoft.com/en-us/projects/trinity/⟩ (accessed 2013-05-09).

[11] memcached プロトコル仕様, 入手先 ⟨https://github.com/memcached/memcached/blob/master/doc/protocol.txt⟩ (参照 2013-09-16).

[12] Yoon, J.H., Hunter, H.C. and Tressler, G.A.: DRAM & Flash, *Proc. Flash Memory Summit 2013* (Aug. 2013), available from ⟨http://www.flashmemorysummit.com/English/Collaterals/Proceedings/2013/20130813_C12_Yoon.pdf⟩ (accessed 2014-08-25).

[13] Wikipedia Statistics, available from ⟨http://stats.wikimedia.org/EN/Sitemap.htm⟩ (accessed 2014-08-25).

[14] 山浦隆博, 田中信吾, 菅沢延彦, 鎌形映二：高効率 TCP/IP オフロードエンジン NPEngine^{TM} の試作と評価，信学総大，B-7074 (Mar. 2009).

[15] 山浦隆博，田中信吾，菅沢延彦，村井信哉：ストレージと TCP/IP オフロードエンジン間の直接転送を用いたデータ送信サーバに関する提案，信学総大，B-6-39 (Sep. 2011).

[16] 田中信吾，山浦隆博，山口健作，菅沢延彦，谷澤佳道，渋谷尚久：Gigabit/10 Gigabit Ethernet に対応した高効率 TCP/IP オフロードエンジン，情報処理学会論文誌，Vol.52, No.12, pp.3715–3728 (2011).

推薦文

本稿は，SSD キャッシュサーバの低応答遅延を実現するために，TCP オフロードエンジンに，SATA アクセスのハードウェアエンジンを組み合わせ，CPU 処理とメインメモリを介さない SSD キャッシュサーバの実現手法が提案されている．低応答遅延の SSD キャッシュサーバは有用性が高く，実現手法も実現可能性が高い提案である．以上より，本研究会からの推薦に値する．

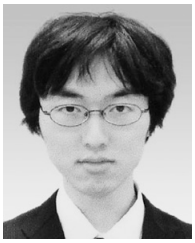
（マルチメディアと分散処理研究会主査　勝本道哲）

後藤 真孝　（正会員）

平成 9 年九州大学大学院システム情報科学研究科情報工学専攻修士課程修了．同年（株）東芝入社．以来同社にて，オペレーティングシステム，通信プロトコル，ストレージシステムに関する研究開発に従事．

村上 瑛美

平成 23 年早稲田大学大学院先進理工学研究科電気・情報生命専攻修士課程修了．同年（株）東芝入社．現在，（株）東芝研究開発センターネットワークシステムラボラトリーで通信プロトコル，ネットワークストレージ関連の研究開発に従事．



秋山 晴彦　（正会員）

平成 24 年東京農工大学大学院情報工学専攻博士前期課程修了．同年（株）東芝入社．ネットワーク・ストレージシステムの研究開発に従事．

村井 信哉　（正会員）

平成 4 年京都大学工学部電気工学科卒業．平成 6 年同大学工学研究科修士課程修了．同年（株）東芝入社，以来同社にて，グループウェア，アドホックネットワーク等のネットワークシステムの研究開発に従事．電子情報通信学会会員．