Vol.2013-DPS-157 No.10
2013/10/18

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]　村上瑛美[†1]　秋山晴彦[†1]　村井信哉[†1]

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，一秒間に 6 万 5 千回のデータ参照要求に応答することができた．また，その時の応答遅延は，約 500 μ 秒であった．

Implementation of the SSD Cache Server using the Direct Transfer HW Engine from storage to network

MASATAKA GOTO[†1]　EIMI MURAKAMI[†1]
HARUHIKO AKIYAMA[†1]　SHINYA MURAI[†1]

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 65000 times per second to GET requests from clients. And, the average time of response delay is about 500 microseconds.

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．そのひとつに，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，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，ユーザの利便性を損ねない高い応答性能を 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一つで，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．DRAM の利用が前提で，単純な操作のみ対応しているため，応答性能が非常に高い．非常に参照頻度の高いデータについて，メモリキャッシュサーバで保持し，Web サーバが画面を構築する際に参照する使い方となる．今般では，頻繁に更新される膨大なデータから画面を構築する Web サービスが増え，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．

本稿では，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

†1　(株)東芝
TOSHIBA Corp.

転送エンジンの概要を述べる．4 章では，直接転送エンジンを適用した SSD キャッシュサーバの構成を述べ，5 章で評価，6 章で考察を述べる．

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．

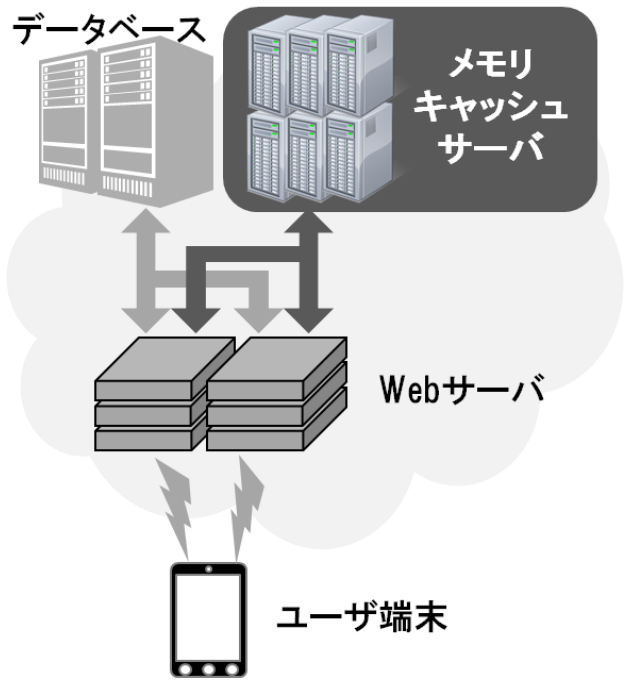

図 2-1 Web サービスシステムの例

図 2-1 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一斉に更新するような機能は有さない場合が多い．

表 2-1　データ読み書きのコマンド

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

表 2-1 に, memcached プロトコル[2]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 2-2 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．

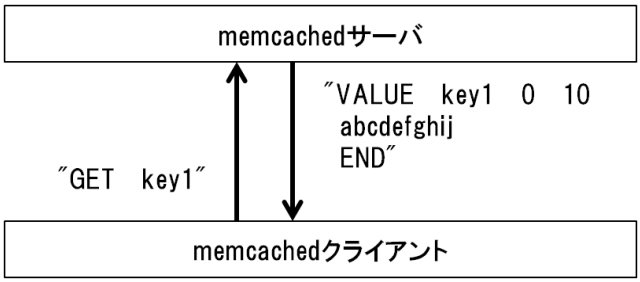

図 2-2　GET コマンドのやり取りの例

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，複数台のサーバを用いたシステムとなりがちで，ランニングコストも無視できない．一方，HDD や SSD などのスト

レージ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が，高頻度のデータ参照が行われる状況で，応答遅延を短く保証することが非常に困難である．

我々は，Web サービスシステムのメモリキャッシュサーバを前提に，SSD を記憶領域とする低応答遅延キャッシュサーバの開発を進めている．コンシューマ用途の SSD を用い，一秒間に数万回以上の高頻度アクセスが行われている状況で，2, 3 ミリ秒程度の応答遅延の実現を目指している．

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 3-1 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)の他に，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[3][4][5]．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．

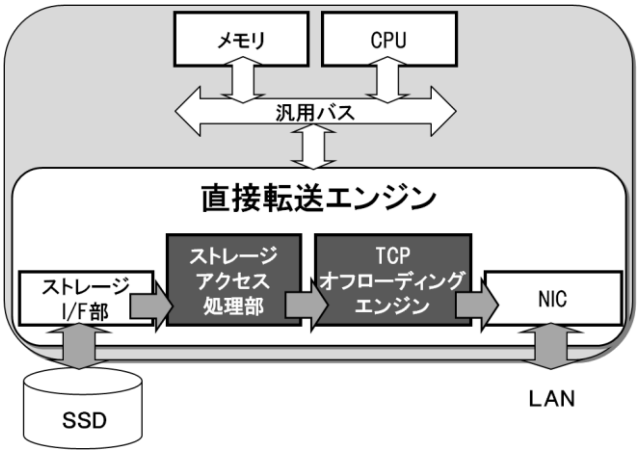

図 3-1　直接転送エンジンの構成

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 記憶のための 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．

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 3-2 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．

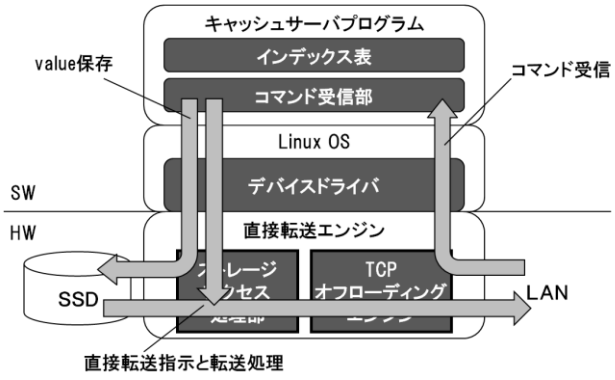

図 3-2　SSD キャッシュサーバの構成

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．本稿では，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．

まず，SET コマンドについて述べる．SET コマンドを受信すると，SSD 上の記憶場所を割り当てる．次に，SET コマンド中に含まれる value を取り出し，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．該当するエントリが既にある場合は，インデックス表の該当するエントリの値を，割り当てた value の位置，長さで更新する．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．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，長さを直接転送エンジンの直接転送機能を呼び出す．

今回の試作では，この直接転送機能の 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た TCP セッションで送出する 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．

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，一旦メモリに読み出し，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，ファイルの削除を伴う．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 4-1 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 4-1 に評価環境の機器構成を示す．あらかじめ，4KB の 1000 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した．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，応答遅延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．

表 4-1　SSD キャッシュサーバスペック

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)

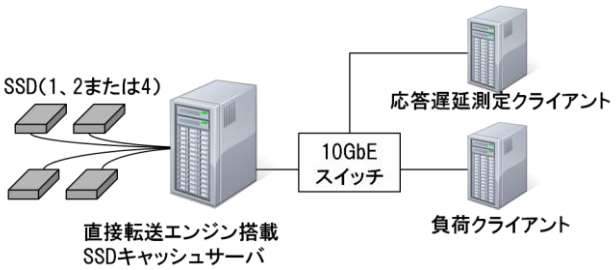

図 4-1　評価環境の機器構成

4.2 測定結果

図 4-2 に，SSD ドライブ数と一秒間の応答回数の測定結果を示す．応答回数は，負荷クライアントからの GET コマンドに対して，試作した SSD キャッ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．

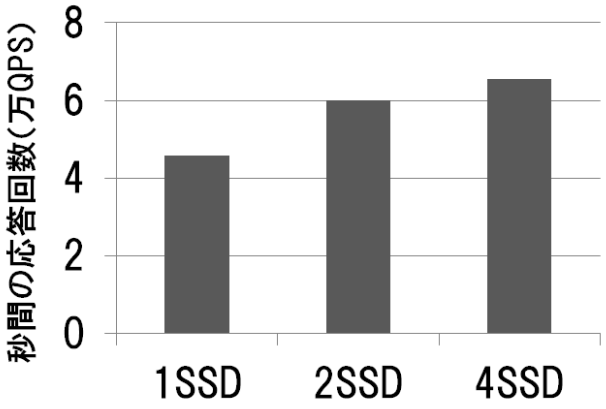

図 4-2　SSD ドライブ数と応答回数の関係

次に，図 4-3 に応答遅延の測定結果を示す．応答遅延時間は，応答遅延測定クライアントから GET 要求を送信した時刻から，4KB の応答がすべて受信されるまでの時間である．図 4-2 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．

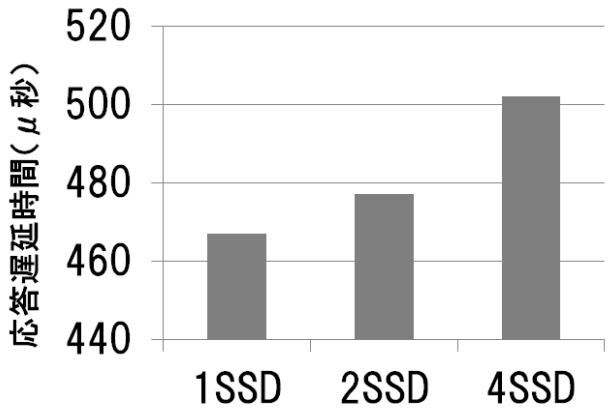

図 4-3　SSD ドライブ数と応答遅延時間の関係

最後に，に負荷クライアントへの応答回数と応答遅延測定クライアントで測定した応答遅延の関係を示す．従来の DRAM 使用のメモリキャッシュサーバの応答遅延も測定し，併記した．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，高頻度のアクセスに対して応答中であっても，応答遅延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，応答回数の上昇とともに，応答遅延時間も延びていく結果となっている．

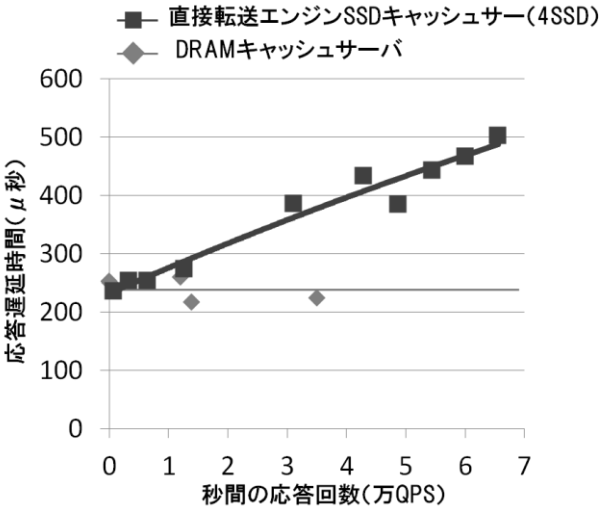

図 4-4　応答回数と応答遅延の関係

5. 考察

5.1 応答回数

図 4-2 より，SSD のドライブ数の増加と共に，一秒間の応答回数が増加している．SATA には，データの読み出しなどのコマンドをキューイングして受け付ける NCQ という機能がある．この応答回数の増加，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, 4KB 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，応答回数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 4KB のため，一秒間に 6 万回の応答であっても，TCP/IP の送信データは 2Gbps 強に留まる．評価環境の LAN の通信帯域である 10Gbps 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づらい．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 応答遅延

図 4-3 より，SSD のドライブ数の増加と共に，応答遅延が増加している．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, 応答遅延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．図 4-4 によると，SSD のドライブ数が 4 台であっても，4 万 QPS 近辺では，400～450μ秒の応答遅延である．このことから，応答遅延時間は，SSD のドライブ数によらず，秒間の応答回数に依存しており，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．

また，図 4-4 より DRAM 版との応答時間の比較を考える．低 QPS では，DRAM 版との応答遅延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 HW 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．評価により，SSD を 4 台使用する場合において，一秒間の GET コマンドの応答回数は約 6 万 5 千回，応答遅延時間は約 500 μ 秒を実現できることを示した．

今後は，ボトルネックの詳細な分析を行う．また，GET コマンドのみでなく，その他のコマンドを併用した場合の性能の傾向を把握する．

参考文献

1) memcached，http://memcached.org，(参照 2013-09-16)
2) memcached プロトコル仕様，https://github.com/memcached/memcached/blob/master/doc/protocol.txt，(参照 2013-09-16)
3) 山浦，田中，菅沢，鎌形，“高効率 TCP/IP オフロードエンジン NPEngine™ の試作と評価，”信学総大，B-7074，Mar. 2009
4) 山浦，田中，菅沢，村井，“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，”信学ソ大，B-6-39，Sep. 2011
5) 田中, 山浦, 山口, 菅沢, 谷澤, 渋谷, “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 52(12), 3715-3728, 2011-12-15

なお，本論文に掲載の商品，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．