Vol.2014-ARC-211 No.10
2014/7/28

パケット処理キャッシュに特化した 低回路コストなキャッシュエントリ制御手法の実現

八巻隼人[†1]　　西宏章[†2]

近年のネットワークトラフィックの増加は，ルータに対する負荷の増大をもたらしている．これに対し，パケット処理においてボトルネックとなる各種のテーブル検索処理を，キャッシュを用いることで高速化するパケット処理キャッシュ機構が提案されてきた．パケット処理キャッシュにおける処理遅延はキャッシュミス率に左右され，メモリ容量に制限のあるパケット処理キャッシュにとって，キャッシュエントリの効率的な配置によるミス削減は重要な課題であった．本報告では，パケット処理キャッシュの性質に特化し，ネットワーク中のショートフローをキャッシュから迅速に追い出し，ロングフローを優先的に残す Hit Priority Cache の提案を行った．実ネットワークトレースを用いたシミュレーションにより，本手法は従来の 4way LRU キャッシュに対し最大で 15.9%のキャッシュミス削減を可能とし，平均テーブル検索処理遅延を最大で 14.5%改善できることを示した．また，本手法のハードウェア実装における実装コストが同エントリ数の 4way LRU キャッシュと同等であることを示した．

Cache Entry Control Mechanism Specialized for Packet Processing Cache with Low Circuit Costs

YAMAKI HAYATO[†1]　HIROAKI NISHI[†2]

Growing network traffic in recent years makes increase of routers loads. Following this, Packet Processing Cache, which achieves fast table lookup that can be the bottle neck of packet processing by using cache architecture, has proposed. The table lookup delay by using Packet Processing Cache depends on the cache-miss rate, and it is an important task for Packet Processing Cache to set a cache entry to an efficient position. In this report, we proposed Hit Priority Cache, which is specialized for Packet Processing Cache and can control the cache entry appropriate. It can reject short flows fast in the cache and save long flows preferentially. In simulations with real network traffics, we showed Hit Priority Cache could reduce the cache-miss by at most 15.9% compared with the normal 4way LRU cache and improve the average delay of the table lookup processing by at most14.5%. In addition, hardware implementation costs of this mechanism are the same as the 4way LRU cache that has the equal entries.

1. はじめに

近年のネットワークでは，プロセッサ性能の向上や携帯端末の普及，それに伴う高容量なデータを用いたサービスの出現によりネットワークトラフィックが急速に増加している[1]．ムーアの法則やギルダーの法則に予想されるプロセッサ、通信帯域の成長速度は緩やかになっているが、トラフィックの増加は加速しており、今後もパケット総量の増加が予想される．このような傾向から，パケット処理を担うルータの負荷は増加する一方である．

従来のルータは，ネットワーク処理に特化したハードウェアを持つ ASIC やネットワークプロセッサといったモジュールを搭載し，高速なネットワーク処理を実現してきた．その内部処理においては，パケットを個々に独立して処理可能であるという事実を基に，プロセッシングエレメント（Processing Element: PE）と呼ばれる小規模なパケット処理専用コントローラを集積し，パイプラインやマルチスレッディングといった並列化技法を併用することで処理の高速化を図ってきた[2]．このような PE 集積に依存した処理手法は，消費電力増大やリーク電流に伴うトランジスタ小型化の限界といった問題を抱えており，処理能力の向上は次第に困難になると考えられている[3]．今後のネットワーク機器において，内包する PE 数を極力増加させることなく処理スループットを改善するパケット処理機構が必要となっている．

上述した問題点を解決する一つの方向性として，パケット処理においてボトルネックとなる各種のテーブル検索を，キャッシュを用いることで高速化する手法が提案されている[4][5][6]．以降，本手法をパケット処理キャッシュと呼ぶ．キャッシュとは，頻繁に用いられるデータを高速アクセスが可能なキャッシュメモリに保存し，次のデータ参照時に使うことで，データ参照の高速化を行う手法である．パケット処理のテーブル検索においては，フローという単位に分けたパケット群毎に検索結果が等しくなるという事実がある．そこで，パケット処理キャッシュではフローの初回パケットのテーブル検索結果をキャッシュメモリへ保存することで，以降の同一フローのパケットに対し，キャ

†1 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†2 慶應義塾大学理工学部システムデザイン工学科
Dept.of System Design, Keio University

ッシュを用いた高速な処理を可能とする．一般的にキャッシュの性能は，キャッシュメモリ中に該当するデータが存在しないキャッシュミスの発生率に大きく左右される．パケット処理キャッシュにおいても，テーブル検索処理速度はキャッシュミス率により決定されるため，処理速度の向上にはキャッシュミスの削減が重要な課題であった[5]．本論文では，まずパケット処理キャッシュにおけるキャッシュミス削減の足掛かりとして，パケット処理キャッシュ特有のデータアクセス傾向の分析を行う．分析結果をもとに，キャッシュミス削減に効果的なキャッシュ手法の提案を行い，実ネットワークトラフィックのトレースを用いたソフトウェアシミュレーションによって手法の評価を行う．

2. パケット処理キャッシュ機構

ルータは IP フォワーディングやファイアウォール，NAT，QoS 制御といった様々なネットワークアプリケーションをワイヤレートで処理しなければならない[7][8]．当初これらの処理は，DRAM 等のメモリに格納されたフォワーディングテーブルやフィルタリングテーブルといった各種テーブルに対し，パケットヘッダの特定フィールドの値を用い，検索を行うという手法であった．DRAM のアクセス速度が遅いことや，最長一致検索（Longest Prefix Match: LPM）[9]を行うために複数回のメモリアクセスを要することから，このようなテーブル検索処理はパケット処理の中でもボトルネックとなりうることが知られている[8][10][11]．特に，ネットワークの高度化に伴いルータでは単純なルーティングテーブルのみならず，QoS テーブル等，複雑なテーブル検索処理が増加しており，テーブル検索処理の高速化は重要な課題となっている．テーブル検索を高速に行う手法としては，CAM（Content Addressable Memory）を用いることでメモリアクセス回数を削減する手法が提案されてきた[12]．しかしながら、CAM の回路コストは DRAM に比べ遥かに大きく，経路数の増大に対し、半導体技術の成長速度が停滞していることから回路規模は増加する一方である。更に、近年標準化のなされた 100G Ethernet が普及した場合には、CAM であってもアクセス速度が不足することが考えられる。そのような中で，近年ではキャッシュを用いたテーブル検索の高速化手法が提案されている[3][4][5]．

パケット処理におけるテーブル検索では，パケットヘッダ中の一つ又は複数のフィールドの値をもとにテーブル内の検索が行われる．従って，このフィールドの値が等しいパケット群では同一のテーブル検索結果が返される．例えば，ルーティングテーブル検索では，宛先 IP の等しいパケット群に対し，同一の検索結果が返されることになる．この事実を利用し，パケット処理キャッシュでは，多くのテーブル検索で用いられる 5 つのフィールド（Source IP，Destination IP，Source Port#，Destination Port#，Protocol#）の組をフローと定義し，フローの初回パケットのテーブル検索結果をキャッシュメモリへと保存する．そして，以降の同一フローのパケットはキャッシュ内のテーブル検索結果を適用することで高速に処理を行う．このようなヘッダ値の等しい同一フローのパケットは，短時間に連続して訪れるという時間的局所性が存在することが知られており，キャッシュを用いることで効率的に処理を行うことが可能となる[13]．本手法の概要を図 1 に示す．本手法では，従来 PE を割り当て行っていたテーブル検索処理に対し，事前にキャッシュ検索を行う．当該フローの処理結果が存在するキャッシュヒット時には，キャッシュの内容を適用することで, PE の割り当てやテーブルメモリの検索を省略し，高速に処理を完了することが可能となる．一般的なキャッシュに用いる SRAM は数 ns オーダでのメモリアクセスが可能であり，数十から数百 ns オーダのアクセス遅延を要する DRAM に比べ，10 倍以上テーブル検索処理遅延を短縮することが可能である．

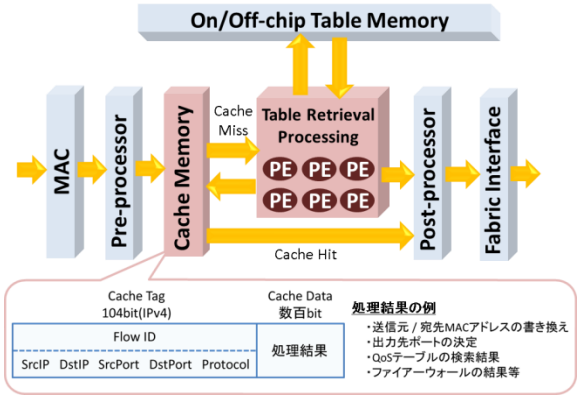

図 1　パケット処理キャッシュ機構
Figure 1　Architecture of Packet Processing Cache.

このようなパケット処理キャッシュ機構では，該当フローの処理結果がキャッシュに存在しないキャッシュミスが発生すると，従来通り処理遅延の大きいテーブル検索処理を行わなければならない．従って，本手法の処理遅延はキャッシュミス率に左右され，処理遅延の改善にはキャッシュミスの削減が重要であった．キャッシュにおいて最も一般的なキャッシュミス削減手法は，メモリ容量を大きくし，キャッシュ可能なエントリ数を増やすことである．しかしながら，メモリ容量とアクセス遅延の間にはトレードオフの関係があり，容量を増やすことでキャッシュミスの削減を行っても全体的な遅延が改善されない場合がある．そこで，まず我々は実ネットワークトレースを用いたシミュレーションにより，パケット処理キャッシュにおける最適なメモリ容量を検討した．図 2 は，後述するパケット処理キャッシュシミュレータを用いた，メモリ容量毎のキャッシュミス率の測定結果を示している．シミュレーションでは，WIDE(Widely Integrated Distributed Environment)とアメリカ間の 150Mbps 国際回線で取得された WIDE トレース及び日

表 1　実ネットワークトレースの詳細
Table 1　Detail of Real Network Traces.

トレース	回線	採取日時	パケット数
WIDE	150Mbps	2009/4/2/0:00-1:00	22,151,899
Academic	10Gbps	2010/6/17/14:26-15:07	630,688,024

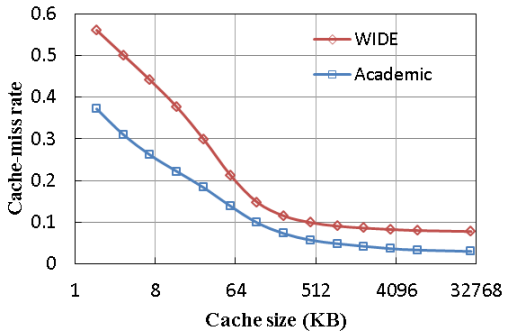

図 2　キャッシュ容量とミス率の関係
Figure 2　Relation of Cache Size and Cache-miss Rate.

本国内の大規模ネットワークの 10Gbps 回線で取得された Academic トレースの二つを用いた．各トレースの詳細は表 1 に示す．図 2 によると，ネットワークに依らず 64KB 程度まではメモリ容量を増加することで，キャッシュミスが線形に改善されていく．しかしながら，これ以上のメモリ容量の増加では僅かなキャッシュミス改善しか得られない．上記のことから，パケット処理キャッシュでは近年のプロセッサの L1 キャッシュと同程度である 32KB または 64KB の高速なメモリを用いることが効果的であると考えられる．

本論文での議論は，L1 キャッシュメモリを想定したパケット処理キャッシュにおいて生じる 15%～20%程度のキャッシュミスを，メモリ容量を増やすことなく削減することである．

3. 関連研究

従来からパケット処理の高速化を目的としたキャッシュにおけるキャッシュミス削減の研究は数多くなされてきた．しかしながら，その多くはルーティングテーブル検索結果のみを保存するキャッシュに関する研究[4][13]であり，近年研究のなされているフローを用いたパケット処理キャッシュに関する研究は少ない．以降では，フローを用いたパケット処理キャッシュのキャッシュミス削減手法の方向性について，主な研究をまとめる．

Georgia Institute of Technology の Li らは，パケット処理キャッシュにおける最適なキャッシュ構成について検討を行った[11]．Li らによると，キャッシュの連想度は 4way がパフォーマンスと実装コストの両面から見て最適であるとし，キャッシュインデックス生成のためのハッシュ関数は実装が容易な XOR ベースを用いることで，複雑な SHA-1 と同程度のキャッシュヒット率を得られると述べている．また，4way キャッシュにおけるキャッシュ追い出しアルゴリズムとして，Least Frequently Used（LFU）やラウンドロビンより Least Recently Used（LRU）を用いることで，高い性能を得られることを示している．

大阪市立大学の阿多らは，パケット処理キャッシュにおける 5 属性のフロー情報が IPv4 でも 104bit と大きくなる問題点に着目し，フロー情報の圧縮手法を提案した[14]．阿多らによると，多くのテーブル検索で用いられるのは送信元 IP，宛先 IP，小さいほうのポート番号の 3 属性であるとし，メモリ使用の効率化を図っている．しかしながら，近年におけるテーブル検索は複雑性を増しており，例えば QoS テーブルでは一般的に前述した 5 属性の値が使われている．このことからも，今後のテーブル検索処理を高速化するためには 3 属性の値では不十分であると考えられる．

同様のキャッシュミス削減手法として，OHSU の Chang らは，104bit のフロー情報を 32bit のハッシュ値へと圧縮することで，同メモリ容量でのエントリ数を増やす Digest Cache の提案を行っている[6]．Digest Cache では，32bit のハッシュ値を生成する回路や，フロー情報を圧縮したことによる衝突への対策が必要となり，全体的なハードウェアコストが一概に削減されるとは言えない．このように，フロー情報を圧縮することでエントリ数を増やすことを目的とした手法は，圧縮により弊害を生じる可能性がある．

一方で，キャッシュエントリを効果的に管理することでキャッシュミスの削減を行う研究がある．多くのキャッシュは，一つのキャッシュインデックスに対し複数のキャッシュデータを割り当てるセットアソシアティブ方式を採用していることが一般的である．上述したように，パケット処理キャッシュにおいても，本方式はキャッシュミス削減に有効であることが知られている．本方式のキャッシュにおいて新たなデータを格納する場合，既に格納されている複数のデータから一つを選んで追い出さなければならない．このような追い出しエントリの決定アルゴリズムは，キャッシュ追い出しアルゴリズムと呼ばれており，最適な追い出しアルゴリズムを適用することで，少ないハードウェアの変更に対し，キャッシュミスを削減できることが知られている．Akamai Technologies の Girish らはパケット処理キャッシュにおいて最適な追い出しアルゴリズムとして Saturating Priority Cache を提案している[15]．また，Girish らは L1 キャッシュ程度のメモリを想定したパケット処理キャッシュシミュレーションにおいて，本手法により 7%程度キャッシュミスを改善可能であることを示している．このように，追い出しアルゴリズムの改善はキャッシュミス削減に有効な手法ではあるが，Girish らも示しているようにトラフィックの種類やエントリ数の違いにより効果も大きく異なり，また大幅なキャッシュミスの改善を望むことはできない．キャッシュミスの大幅な削減には，新たな方向性を見つけることが必要であると考えられる．

4. キャッシュミス削減手法の検討

従来，CPU キャッシュやページキャッシュといった様々なキャッシュにおいて，キャッシュミス削減の研究がなされてきた[16,17]．しかしながら，これらの研究により提案された手法を，そのままパケット処理キャッシュに適用することは，アクセスパターンの違いから良い手段とはいえない．CPU キャッシュやページキャッシュに利用されるデータアクセスの局所性とは異なり，パケットは独特の局所性を有している．そこで本報告では，まずネットワークトラフィックに存在するパケット処理キャッシュ特有の傾向を分析することで，キャッシュミス削減の足掛かりとする．

4.1 パケット処理キャッシュのデータアクセス分析

パケット処理キャッシュにおいてキャッシュミスを増大させる要因の一つに，少数パケットで構成されるフロー（以降ではショートフローと呼んで扱う）によるキャッシュ汚染の問題がある．特に 1 パケットで構成されるフローはキャッシュヒットの機会がなく，不要にキャッシュエントリを圧迫する[18]．図 3 は実ネットワークトラフィックの解析により得られた，フロー構成パケット数の分布を示している．図 3 によると，大部分のパケットが 10 パケット以下で構成されていることがわかる．また，トラフィックに依るが，1 パケットで構成されるフローはフロー全体の半数程度であり，このようなフローがキャッシュに与える影響は大きい．パケット処理キャッシュでは，ショートフローを早く追い出すことが重要となる．

一方で，ネットワークには動画トラフィックのような多数のパケットにより構成されるフロー（以降ではロングフローと呼んで扱う）も存在する．図 3 に示したトラフィック解析では，1 時間で 10 万パケットを超えるフローが複数存在している．このような，多数のショートフローと少数のロングフローがネットワークトラフィックを占める傾向は，Elephant and Mice Phenomenon として知られている[19]．ロングフローは少数ではあるが，パケット数の観点から見るとトラフィックの大部分を占めており，ネットワークへ大きな影響を与えている．上記の知見をもとに，我々はパケット処理キャッシュにおけるロングフローの挙動に着目した．図 4 は，パケット処理キャッシュシミュレーションにおける，ロングフローのヒット状況の一例を示している．多くのロングフローでは一連のパケット群全てがヒットしているわけではなく，途中にキャッシュミスが発生している．これは即ち，キャッシュ内でロングフローエントリの追い出しと再登録が繰り返され，キャッシュにとって効率の悪いアクセスが頻繁に発生しているということである．パケット処理キャッシュでは，ショートフローの排除と同時に，ロングフローを優先的にキャッシュに残す手法が必要である．

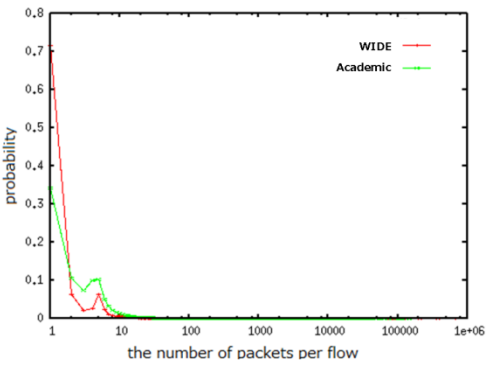

図 3　フロー構成パケット数の分布

Figure 3　Distribution of the Number of Packets per Flow.

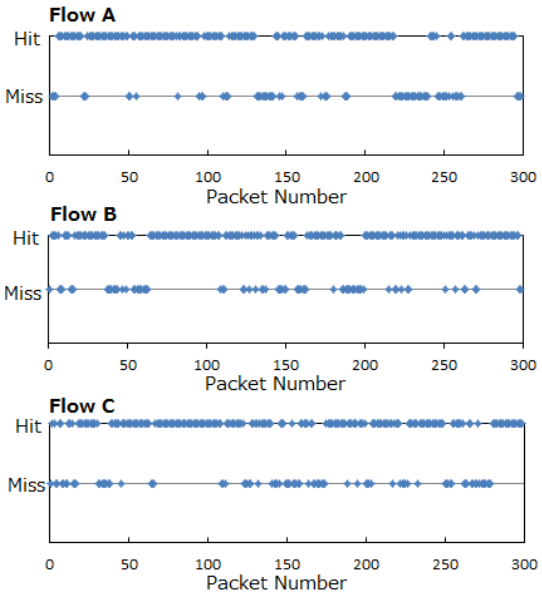

図 4　キャッシュにおけるロングフローの挙動

Figure 4　Behavior of Long Flow in Cache.

4.2 Hit Priority Cache の提案

ネットワーク解析で得られた結果を基に，本報告ではショートフローエントリを迅速に排除し，なおかつ，ロングフローエントリを優先的にキャッシュに残す Hit Priority Cache (HPC)の提案を行う．ロングフローを決定することは困難であるため，HPC では連続的にヒットするエントリをロングフローであると推測し優先的に残す．HPC の概要を図 5 に示す．

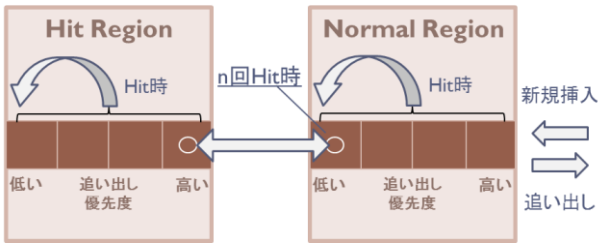

図 5　Hit Priority Cache の処理概要

Figure 5　Processing Outline of Hit Priority Cache.

本手法はキャッシュメモリを半分に分け，通常領域とヒット領域とし，通常領域で一定回数キャッシュヒットしたエントリをヒット領域に移動することで複数回ヒットしたエントリを優先的に残す．一方で，新規エントリ挿入時には，通常領域の最も追い出されやすい位置へと挿入することでショートフローを迅速に追い出すことを可能とする．通常領域とヒット領域のエントリ入れ替えは，通常領域で n 回目のキャッシュヒットをし，最も追い出し優先度が低くなったエントリと，ヒット領域の最も追い出し優先度の高いエントリ間で行われる．これにより，領域間でのエントリ入れ替え時にヒット領域の追い出し順位が変わることなく，本手法の実装コストを削減することが可能となる．実装コストの検討については第 5 章で後述する．

5. 性能評価

本章では HPC のキャッシュミス削減効果及び実装コストに関して検討を行う．従来多くのキャッシュにおいて，ヒットしたエントリに特別な優先度を設けることでキャッシュミス削減を行う手法が提案されてきた[20]．本報告では，HPC と同様にヒットしたエントリを特別な領域へ移動する Segmented LRU（SLRU）を比較対象として用いる[21]．SLRU の概要を図 6 に示す．SLRU では，新規エントリは追い出し優先度の最も低い MRU ポジションに挿入される．そして，キャッシュヒット時には Protected Segment の MRU ポジションへと移動が行われる．SLRU はキャッシュを二つの領域に分けてはいるが，キャッシュの挙動としては 8way キャッシュにおいて新規エントリの挿入位置を 4 番目に追い出されやすい位置へと変更した，LRU の改良手法に等しい．

5.1 キャッシュミス削減性能のシミュレーション

まず，HPC におけるキャッシュミス削減性能を評価するため，C++で作成したシミュレータによるパケット処理キャッシュシミュレーションを行った．シミュレータの構成を図 7 に示す．本シミュレータはネットワークトレースのタイムスタンプ値をもとに，時間経過をシミュレートし処理を行う．経過時間がタイムスタンプ値と一致したパケットは，Flow Extractor によりフローID の抽出が行われる．

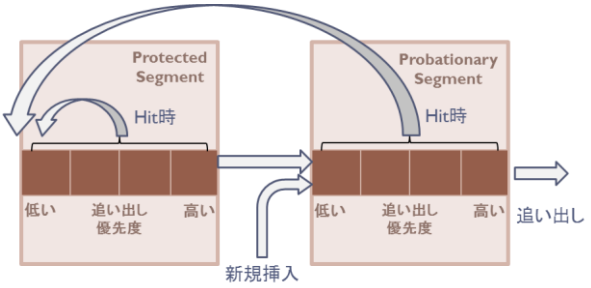

図 6　Segmented LRU の処理概要
Figure 6　Processing Outline of Segmented LRU.

更に，フローID は CRC により 7bit にハッシュ化され，キャッシュインデックスとして用いられる．得られたハッシュ値とフローID からパケット処理キャッシュの検索が行われる．キャッシュメモリ容量は 32KB とし、総エントリ数を 1,024 エントリとした。ここでは，キャッシュヒットした場合には PE に処理が割り当てられず，当該パケットの処理が完了になると仮定している．一方で，キャッシュミスした場合には，Cache Miss Table（CMT）へと処理が送られる．CMT は，キャッシュミスしたパケットの PE 処理が完了する前に同一フローの後続パケットが到着した際，連続してキャッシュミスが発生してしまうことを防ぐために，PE で処理中のパケットを CMT にリストし，PE 処理が完了するまで後続パケットを Cache Miss Queue（CMQ）に蓄えておく機構である[22]．CMT にリストされていないフローは，PE への処理の割り当てが行われ， PE の処理遅延が課される．本報告では PE の数を 64 個，PE のテーブル検索遅延を 100ns としている．遅延分の時間が経過した後，当該パケットの処理は完了となり，同時に CMQ 内の同一フローパケットの処理も完了となる．更に，この時当該フローのキャッシュエントリを新規エントリとしてキャッシュに挿入する．本シミュレータにおけるキャッシュミスはキャッシュ及び CMT でミスをしたパケットとなる．

本シミュレーションによる，WIDE 及び Academic トレースを用いたパケットミス率の測定結果を図 8 に，4way LRU キャッシュに対するパケットミスの改善率を図 9 に示す．HPC のシミュレーションでは，通常領域からヒット領域へエントリを移動するために必要なヒット回数を 1, 2, 4, 8, 16 に設定し，それぞれを HPC1～HPC16 と呼ぶことにした．シミュレーション結果によると，HPC のキャッシュミス削減効果は，同様にヒットエントリを特別に扱う SLRU よりも高く，WIDE トレースでは 15%程度，Academic トレースでは 11%程度のキャッシュミスが改善されている．また，HPC における通常領域とヒット領域のエントリ移動は，トラフィックに依らず 4 回が最も効果的であり，これ以上の回数ではミスが増大することがわかった．本シミュレーシ

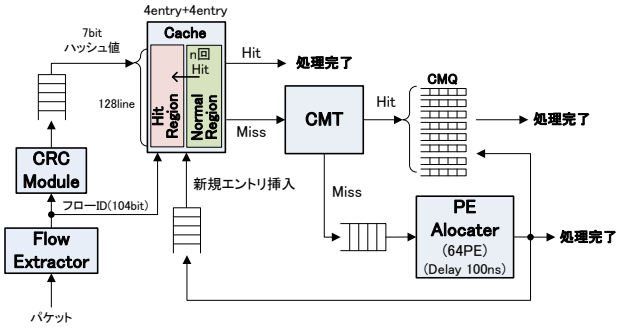

図 7　パケット処理キャッシュシミュレータの構成
Figure 7　Composition of Packet Processing Cache Simulator.

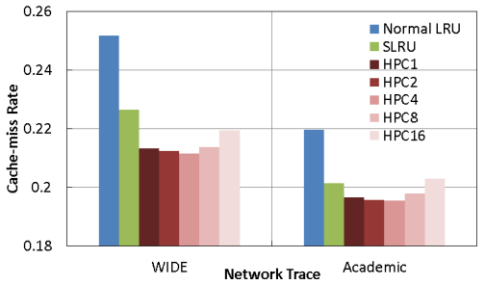

図 8　キャッシュミス率の測定結果

Figure 8　Measurement Results with Cache-miss Rate.

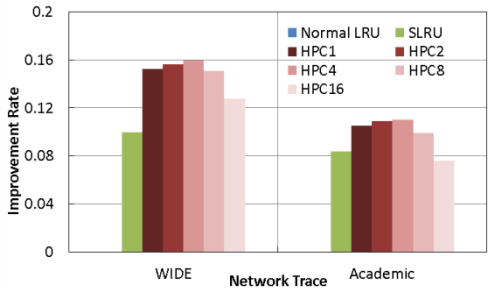

図 9　キャッシュミス改善率の測定結果

Figure 9　Measurement Results with Improvement Rate of Cache-miss.

ョンにより，パケット処理キャッシュの性質に特化した HPC4 を用いることで大幅にキャッシュミス削減が可能であることが示せた．

5.2 実装コストの検討

キャッシュのハードウェア実装における実装コストは，メモリ容量による回路規模とエントリ優先度の管理コストが大部分である．HPC は一つのキャッシュインデックスに対し，8 つのデータが入った疑似的な 8way キャッシュ構造である．一般に 8way キャッシュは，得られるキャッシュミス削減効果に対する実装コストが高いことが知られている[23]．近年の 4way 以下のキャッシュにおける LRU 実装は，キャッシュライン毎のエントリ優先度の順位を優先度管理メモリに格納しておき，ヒットがあった際に当該ラインの優先度順位を遷移させ，優先度管理メモリを更新する手法が用いられている[24]．8way キャッシュではエントリ優先度の順位が 40,320 通りであり，更に，各 way でヒットすることによる遷移パターンは 322,560 通りである．このように，8way 以上のキャッシュではエントリ優先度管理のための実装コストが大幅に増加する．SLRU も上記の問題を解決する手法ではなく，ハードウェアへの実装が必要なキャッシュには向いていない．パケット処理キャッシュでは，キャッシュミス削減効果だけでなく実装コストについても検討を行う必要がある．

HPC は疑似的な 8way 構造のエントリ優先度を 4way と等しい実装コストにより管理することができる．HPC4 の

図 10　HPC4 の優先度管理の例

Figure 10　Example of Priority Management on HPC4.

優先度管理の例を図 10 に示す．HPC では通常領域とヒット領域を分け，それぞれ 4way として優先度を付ける．更に，通常領域のエントリにヒットカウンタを設け，ヒット領域とのエントリ交換に用いる．ヒットカウンタは HPC2 では 1 エントリあたり 1bit，HPC4 では 2bit となり，例えば 1K エントリを持つ HPC8 では 3bit のカウンタを 512 エントリ分で 192Byte が必要となる．通常領域のエントリがヒットし，カウンタが溢れた時に，通常領域とヒット領域それぞれのキャッシュメモリ内のデータを入れ替える（図 10 中のフローA とフローG）．この際，エントリ優先度の遷移は通常領域のみで起こり，ヒット領域の優先度順位を変更する必要はない．ヒット領域でキャッシュヒットが起こった際には，ヒット領域のみ優先度の遷移を行う．また，新規エントリの挿入は，通常領域の優先度の最も低いエントリとキャッシュデータを入れ替えるだけであり，優先度の遷移は起こらない．このように，HPC におけるエントリ優先度の遷移は，キャッシュヒットのあった領域で 4way キャッシュと同様に遷移を行うだけで良く，優先度管理にかかる実装コストは LRU を用いた 4way キャッシュと同等である．

5.3 平均検索遅延の検討

パケット処理キャッシュによるテーブル検索処理では，処理遅延はキャッシュミス率に左右される．State University of New York の Gopalan らによると，キャッシュミス率を m，L1 キャッシュのアクセス遅延を t_h，テーブル検索処理の遅延を t_m としたとき，平均検索遅延は $t_h \times (1-m) + t_m \times m$ により計算される[5]．また，カリフォルニア大学の Talbot らによると，t_h を 2ns，t_m を 100ns とすることができる[25]．本節では，上記の手法をもとに HPC の平均検索遅延について検討を行う．

HPC は，通常のキャッシュ機構とは異なり，通常領域のエントリが複数回ヒットした時にヒット領域のエントリと交換が行われる．このようなキャッシュ更新は，パケット

表 2　エントリ交換発生率と平均検索遅延の測定結果
Table 2　Measurement Results with Rate of Entry Change and Average Lookup Delay.

	WIDE			Academic		
手法	ミス率	交換発生率	平均遅延[ns]	ミス率	交換発生率	平均遅延[ns]
LRU	0.252	0	26.7	0.220	0	23.5
SLRU	0.227	0	24.2	0.201	0	21.7
HPC1	0.213	0.0858	23.2	0.197	0.0650	21.5
HPC2	0.212	0.0436	23.0	0.196	0.0411	21.3
HPC4	0.212	0.0177	22.8	0.196	0.0206	21.2
HPC8	0.214	0.00682	23.0	0.198	0.00876	21.4

到着に余裕がある時は次パケットの到着までに行うことで遅延とならないが，キャッシュに対して待ち行列ができている際には遅延となる．ここで，エントリ交換にかかる遅延を 4ns と仮定した場合，HPC によるエントリ交換を含めた平均検索遅延は次式により計算される．

$$2\text{ns} \times (1-m) + 55\text{ns} \times m + 4\text{ns} \times c$$

式中の c は，エントリ交換が発生する割合である．HPC では，エントリ交換を行うためのヒット回数 n によって，エントリ交換の発生率が異なるため，n の値が大きい程エントリ交換遅延の影響は少なくなる. これらの議論をもとに，エントリ交換の発生率をシミュレーションにより測定し，平均検索遅延の計算を行った結果を表 2 に示す．WIDE と Academic トレースでは，共に HPC4 によるパフォーマンスが最も高い．WIDE トレースでの平均検索遅延は 22.8ns で LRU に対し 14.5%遅延を改善している．また，Academic トレースでの平均検索遅延は 21.2ns で LRU に対し 9.71%遅延を改善している．これらの結果から，パケット処理キャッシュではトラフィックに依らず HPC4 を用いることで，10%以上のテーブル検索遅延の改善を見込めることが示された．

6. 結論

近年のネットワークトラフィックの増加に対し，ルータは，従来の PE 集積度向上による並列処理手法のみでスループットを向上することは困難となっている．そこで，パケット処理においてボトルネックとなるテーブル検索の処理結果をキャッシュすることで，テーブル検索処理を高速化するパケット処理キャッシュが提案されてきた．パケット処理キャッシュにおける処理遅延はキャッシュミス率に左右され，容易にキャッシュ容量を向上できないパケット処理キャッシュにとって，エントリの適切な制御によりキャッシュミスを削減することは重要であった. 本報告では，パケット処理キャッシュの性質に特化したエントリ制御手法である Hit Priority Cache（HPC）を提案することでキャッシュミス削減を行った．

HPC では，新規エントリは通常領域の LRU ポジションへと挿入され，複数回ヒットすることでヒット領域へと移動する．これにより，ロングフローに高い優先度を充て，尚且つ，ショートフローを積極的に追い出すことが可能となり，15%程度のキャッシュミスを改善できることがシミュレーションにより示された．また，本手法のハードウェア実装コストは，一般的な 4way キャッシュにヒットカウンタの数百 Byte を追加した程度であり, 比較的容易に実装可能である. HPC では, 連続してパケットが到着した際に，エントリ交換に伴うメモリ読み書きのための遅延が生じる．この遅延の影響は，エントリ交換するためのヒット回数を多く設定する程小さくなるが，キャッシュミス率を考慮すると，4 回のヒットによりエントリ交換する HPC4 を用いることが最適である．HPC4 を用いることで，テーブル検索処理の平均遅延は一般的な LRU を用いた 4way キャッシュに対し，10%以上改善できることが示された．

謝辞

この研究は、文部科学省　社会システム改革と研究開発の一体的推進気候変動に対応した新たな社会の創出に向けた社会システムの改革プログラム「グリーン社会 ICT ライフインフラ」、文部科学省科学技術研究費補助金基盤研究 B「機能維持性を高める建物・複数機器の協調制御」(24360230)ならびに「コンテンツベース・スマートコミュニティインフラの構築と展開」(25280033)、の一環としてなされた。

参考文献

1)　我が国のインターネットにおけるトラヒックの集計・試算 http://www.soumu.go.jp/main|_content/000279409.pdf
2)　Crowly, P. and Onufryk, P.Z.: Network Processor Design: Issues and Practices, vol.1, Morgan Kaufmann (2003).
3)　Ceuppens, L. and Kharitonov, D.: Power Saving Strategies and Technologies in Network Equipment Opportunities and Challenges, Risk and Rewards, International Symposium on SAINT, pp.381-384 (2008).
4)　Feldmeier, D.C.: Improving gateway performance with a routing-table cache, In Proceedings of IEEE INFOCOM,

pp.298-307(1988).
5) Gopalan, K. and Tzi-cker, C.: Improving Route Lookup Performance Using Network Processor Cache, In Proceedings of ACM/IEEE SC, pp.16-22 (2002).
6) Francis, C. and Kang, L.: Efficient Packet Classification with Digest Caches, In HPCA Workshop on Network Processors (NP3 (2004).
7) Baer, J.L., Low, D., Crowley, P., and Sidhwaney, N.: Memory Hierarchy Design for a Multiprocessor Look-up Engine, In Proceedings of PACT, p.206 (2003).
8) Taylor, D.E.: Survey and taxonomy of packet classification techniques, In Proceedings of ACM Computing Surveys, Vol.37, No.3, pp.238–275 (2005).
9) Douglas, E.C.: Network Systems Design using Network Processors, Pearson Education Inc., p.68(2004).
10) Gupta, P. and McKeown, N.: Algorithms for packet classification, IEEE Network, Vol.12, No.2, pp.24-32 (2001).
11) Kang, L. et al.: Architectures for packet classification caching, In Proceedings of ICON, pp.111-117 (2003).
12) Kobayashi, M. et al.: A longest prefix match search engine for multi-gigabit IP Processing, In Proceedings of IEEE ICC, Vol.3, pp.1360-1364 (2000).
13) Chvets, I. L. and MacGregor, M.H.: Multi-zone caches for accelerating IP routing table lookups, IEEE Workshop on HPSR, pp.121-126 (2002).
14) Ata, S. et al.: Efficient cache structures of IP routers to provide policy-based services, In Proceedings of IEEE ICC, vol.5, pp.1561-1565(2001).
15) Girish, C. and Govindarajan, R.: Improving performance of digest caches in network processors, In Proceedings of HiPC, pp.6-17(2008).
16) Moinuddin, K.Q. et al.: Adaptive insertion policies for high performance caching, In Proceedings of ACM ISCA, pp.381-391(2007).
17) Chuanjun, Z. and Bing, X.: Divide-and-conquer: a bubble replacement for low level caches, In Proceedings of ACM ICS, pp.80-89(2009).
18) Yamaki, H. and Nishi, H.: An Improved Cache Mechanism for Cache-based Network Processor, Journal of Communication and Computer, Vol.10, No.3, pp.277-286 (2013).
19) Tatsuya, M. et al.: Identifying elephant flows through periodically sampled packets, In Proceedings of ACM SIGCOMM IMC, pp.115-120 (2004).
20) Guangdeng, L. et al.: A new IP lookup cache for high performance IP routers, In Proceedings of ACM/IEEE DAC, pp.338-343(2010).
21) Ramakrishna, K. et al.: Caching strategies to improve disk system performance, Journal of Computer, Vol.27, No.3, pp.38-46(2004).
22) Okuno, M. and Nishi, H.: Network-Processor Acceleration-Architecture Using Header-Leraning Cache and Cache-Miss Handler. In Proceedings of SCI2004, Vol.3, pp.108-113(2004).
23) Hassan, G. and Fatemi, S.O.: Pseudo-FIFO Architecture of LRU Replacement Algorithm, In Proceedings of IEEE INMIC, pp.1-7 (2005).
24) Hassan, G. et al.: Modified pseudo LRU replacement algorithm, International Symposium and Workshop on IEEE ECBS, pp.376-381(2006).
25) Talbot, B. et al.: IP caching for terabit speed routers, In Proceedings of GLOBECOM, Vol.2, pp.1565-1569 (1999).