平成 12 年度

学士学位論文

データ駆動方式による
超高速 IP アドレス検索の実現法

A Data-Driven Implementation of
IP Address Lookup

1010454　　森川 大智

指導教員　　岩田 誠

2001 年 2 月 5 日

高知工科大学 情報システム工学科

要 旨

データ駆動方式による
超高速 IP アドレス検索の実現法

森川 大智

近年、インターネットの急速な普及によって、ネットワーク中でデータ通信量が急激に増加している。伝送技術に関しては DWDM などの技術により大容量化されつつあるが、Tbps 超の性能を達成するルータ技術に関してはまだ研究段階にある。さらに、ネットワークの利用形態の多様化に伴い、複数のプロトコルを扱う要求が高まっている。従来ルータでは、各プロトコル処理向きの専用 LSI で対応してきたため、新しいプロトコルに対して LSI を新規に設計する必要があり、開発コストが増加する。

本論文では、ルータの主要なボトルネックの一つである IP アドレス検索処理の超高速実現法を提案する。提案方式は、検索速度を支配する外部メモリアクセス回数を、パイプライン並列処理により隠蔽し効率的なメモリアクセスを行うことによって高い検索レートを維持する。性能評価の結果、本実現法により最大 18Gbps の性能を達成可能なことが確認された。

キーワード　超高速ルータ、IP アドレス検索、レベル圧縮トライ、データ駆動方式、マルチプロセッサ、自己タイミング型パイプライン

Abstract

A Data-Driven Implementation of IP Address Lookup

Daichi MORIKAWA

Recently, the data traffic within the information communication network is significantly increasing along with rapid diffusion of the Internet. Although high bandwidth links are realized by the advanced transmission technology such as DWDM, the router technology operating beyond T bps throughput is still opened as a challenging theme. Further, recent routers are required to handle one or more protocols suitable for the individual application. Conventional routers have usually employed multiple ASIC devices each of which copes with a specific protocol. However, every time a new protocol then comes out, you must pay higher development cost to design LSI chip again newly.

key words high-speed router, IP address lookup, level-compressed trie, data-driven, multi-processor, self-timed pipeline

目次

図目次

表目次

第 1 章

序論

近年、ネットワークにおけるデータ通信量が増大しており、今後の超高速ネットワークにおいてネットワークの大容量化が急務となっている。ネットワークにおける物理リンクに関しては DWDM(Dense Wavelength Division Multiplexing) 等の技術の発達で数 Tbit/sec の性能が出せるようになっており、1Gbit/sec 以上のリンク速度のギガビットイーサネット (IEEE 802.3z)、OC-48(2.4Gbit/sec) 等が登場している。さらに、近い将来 OC-192(9.6Gbit/sec) レベルのインタフェースの開発も積極的に議論されており、このようなリンク速度に追従できる超高速ルータの開発が急務となっている。

ルータはパケットを受信すると、パケット長、チェックサム等を検査する。正当な IP パケットと判断されると、宛先 IP アドレスを取り出す。次に、LPM(Longest Prefix Match) 検索により、フォワーディングテーブルの検索を行い、出力すべきインターフェース情報を得る。最後に、TTL(Time To Live) の減算、チェックサム再計算の処理を行い、パケットを出力インターフェースへフォワードする。また、QoS 制御処理に対応するため、上位ヘッダも合わせて検索する機能が求められる。これらの処理の中で、ヘッダの正当検査など、テーブル検査処理以外の処理は必要な計算量が小さく、並列化も可能なため、高速化は容易である。しかし、テーブル検査処理は検索アルゴリズムが複雑になり計算量が大きくなる上、検索対象であるテーブルの容量が大きく、並列化が困難で、フォワーディング処理のボトルネックとなっている。

IP アドレス検索に使用される IP フォワーディングテーブルは、宛先フィールドとそれに対応する出力インタフェース情報フィールドの組から構成される。被検索対象となるのは宛先フィールドで、プレフィックスと呼ばれる、IP アドレスを MSB から任意の長さのビット

分だけ指定したビット列で表現される。IP アドレスと一致するプレフィックスの中で、プレフィックス長がルーティングテーブル中で最長のものを検索する方法を LPM 検索と呼ぶ。LPM 検索はテーブルの宛先フィールドとパケットの宛先 IP アドレスを、エントリのプレフィックスで指定した部分にのみ着目して比較し、一致するエントリを検索結果とする。

LPM 検索に対する高速化については様々な研究がされており、スループットも平均的にはマルチギガビットのリンク速度に近づきつつある。研究により提唱されている方式は、主に専用ハードウェアを用いて行うものと、汎用プロセッサ上でソフトウェア的に処理するものがある。専用ハードウェアによる検索方法は、高速な処理が可能な反面、複数プロトコルなどに対する場合にハードウェア構成を再構築する必要があり、柔軟性に欠けるという問題がある。一方、汎用プロセッサを用いた検索方式は、ソフトウェアにより実現可能なため、アルゴリズムにもよるが、柔軟性は得られる。しかし、処理速度は専用ハードウェアに比べて遅い。

本稿では、従来のこうした問題点を解決する、専用ハードウェアの高速性と、ソフトウェアの変更のみで処理内容の変更可能な柔軟性を得るために、専用ハードウェアと同程度の高速性を持ち、かつソフトウェア処理で処理が行える、データ駆動プロセッサ [1] を用いる。

まず、2 章で現在のルータで実現されている IP アドレス検索法について示し、高速化のための問題点を述べる。

3 章では、2 章で示した特性および問題点を基に、高速処理が可能で、かつ柔軟性を持つ IP アドレス検索アルゴリズムをデータ駆動プロセッサで実現する方法について述べる。

4 章では、IP アドレス検索処理における要求条件を提示し、性能評価を行う。

第 2 章

既存の方式とその問題点

2.1 緒言

既存の IP アドレス検索方法は、大きく分けると

- キャッシュによる検索方法
- ハードウェアによる完全一致検索方法
- ツリー構造のデータ構造を用いた検索方法

の 3 つが挙げられる。

以下に、各検索方法について述べ、ハードウェア上での実現においての問題点を示す。

2.2 キャッシュ方式

高速な汎用プロセッサのキャッシュメモリにフォワーディングテーブルをコンパクトに格納する方法である [2][8][9]。これらの方法は、平均検索時間としては、マルチギガビットのリンク速度に近づいているが、 検索のキーとして与える IP アドレスとルーティングテーブルの組み合わせによってヒット率が異なり検索時間にばらつきが大きい。さらに、この方式は図??が示すように、現在普及している IPv4 におけるプレフィックス分布を元にキャッシングを行っているため、動的なトラヒックの分布への適応をが困難である。さらに、IP アドレス長の変更によるプレフィックス分布にの変化にも対応する必要があり適応性が低い。

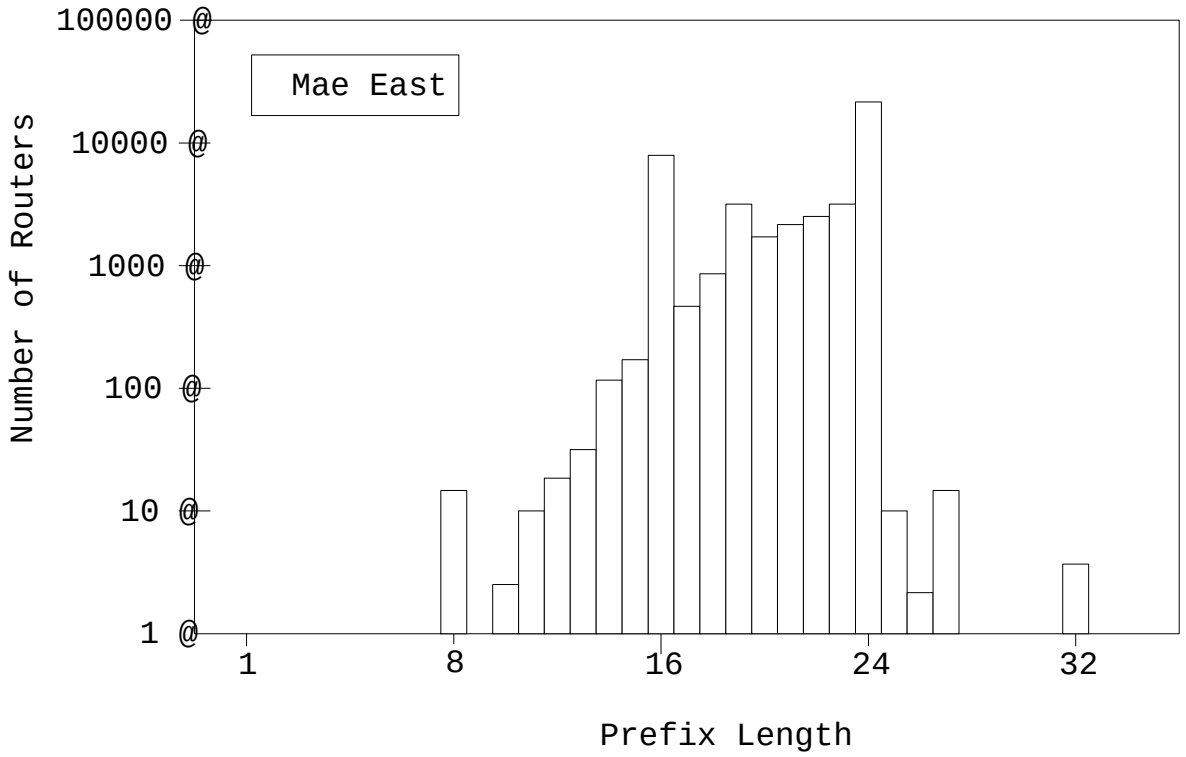


図 2.1 IPv4 におけるプレフィックス分布

2.3 ハードウェアによる完全一致検索方式

ハードウェアによる IP アドレス検索は、キャッシュ方式同様プレフィックス分布における 24bit 長のプレフィックスが多いということを活用し、プレフィックスの完全一致を行っている [3][4][10]。これにより、高速な検索とメモリ要領の削減を達成できるが、プレフィックス分布に依存したプロセッサ構成になってしまうという問題点がある。ハードウェアよる検索法の一つである DIR-24-8-BASIC アーキテクチャを例に示す (図 2.2)。

この方法は、24bit 長のプレフィックスがメモリ (TBL24) のアドレスに対応しており、宛先 IP アドレスの 24bit 分をポインタとしてメモリ参照を行う。一致するものがあるプレフィックスに対しては TBL24 は NextHop を出力する。一致しないものはさらにプレフィックス検査を行うため、TBL Long へのポインタを出し、宛先 IP アドレスの残り 8bit をオフセットとしてテーブル参照し NextHop を得る。このように、特定のプロトコルに対して設計が行われているため、製造時に想定したプロトコル以外のプロトコル (IPv6 等) が登場し

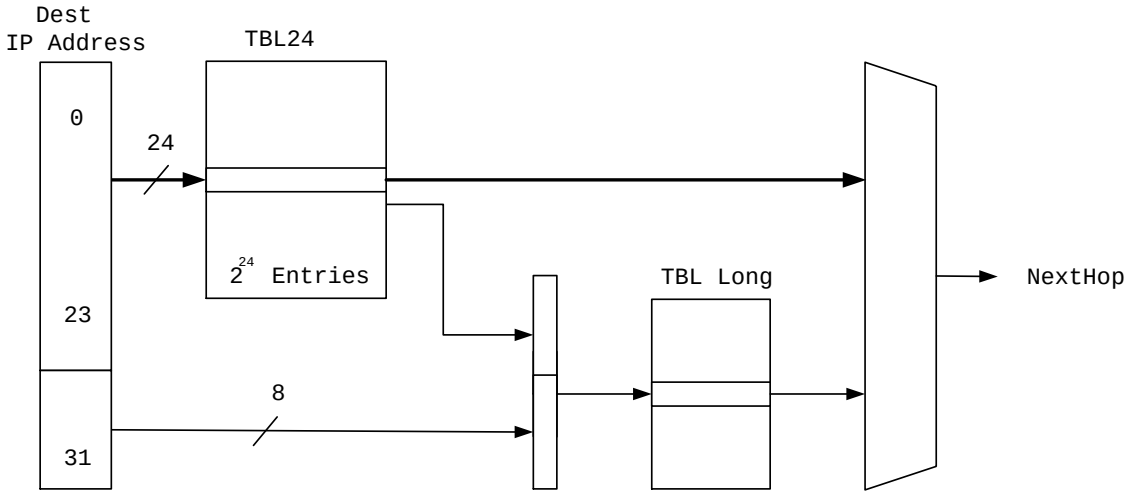


図 2.2 DIR-24-8-BASIC

たとき、新たにハードウェアを用意する必要があり、柔軟性に欠け、さらに製造コストもかかる。

2.4 ツリー方式

フォワーディングテーブルをツリー構造で表現し、検査 bit に基づいて親ノードから小ノードへ分岐をし、検索を行う方式である [5][6][7]。この検索方法では、ツリーをたどるときの枝分かれ毎に、比較判定と、時間のかかるメモリアクセスが必要なため、検索時間がかかる。また、この方式は汎用プロセッサ上でソフトウェアとして実行されるため、専用ハードウェアによる実現法ほど高速化が望めない。

ルーティングテーブルを Trie 構造に写像すると図 2.3 のようになるが、これではノード数が大きくなり検索にかかる平均深度 (ルートからリーフまでのパスの平均長) が長くなる。

この問題を解決するために現在用いられている手法の一つが Path Compress 法である。この手法は、Binary Tree 木における子ノードが一つしかない中間ノードを削除し、パス検索に必要ないビットの長さをノードに保持することでパス長を縮小する方法であり、一般に Patricia Tree と呼ばれる。図 2.3 の Trie を Pass Compress することによって生成された Patricia Tree を図 2.4 に示す。Patricia Tree では、木構造におけるリーフの数を n

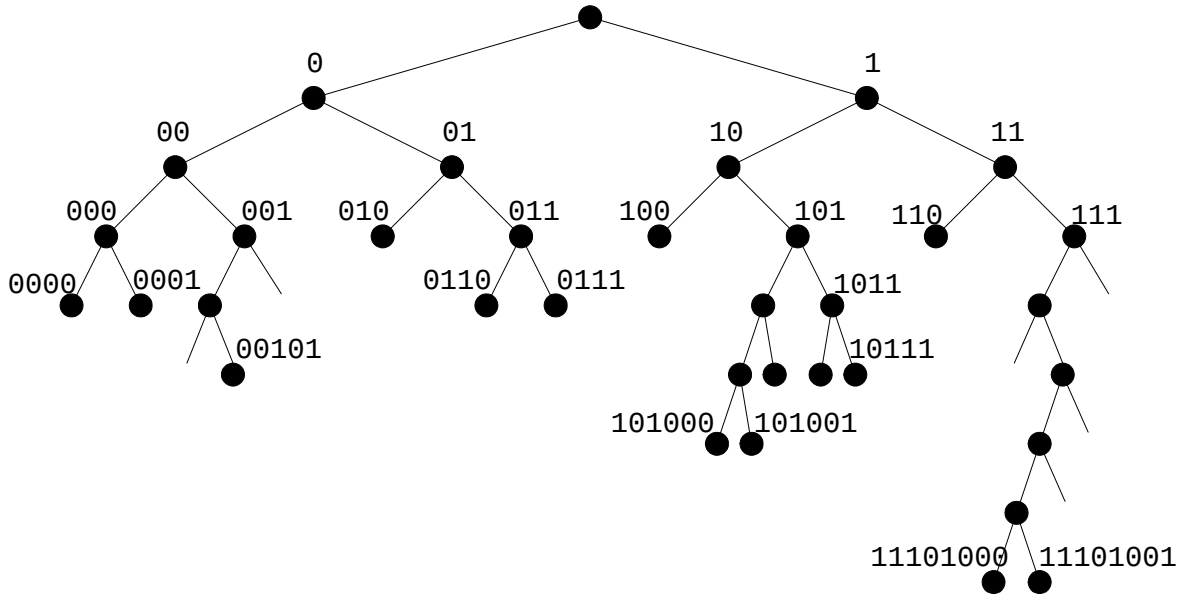


図 2.3 Trie

としたとき、総ノード数は必ず $(2n-1)$ になる。この統計的な特徴はよく知られている。しかし、ツリー構造に写像する対象データの長さが長くなった場合に、Path Compress 法ではデータ数に比例して深さ方向にノード数が増え、平均深度の増加を抑制することができず、メモリアクセス回数、およびメモリ使用量の削減には至らない。

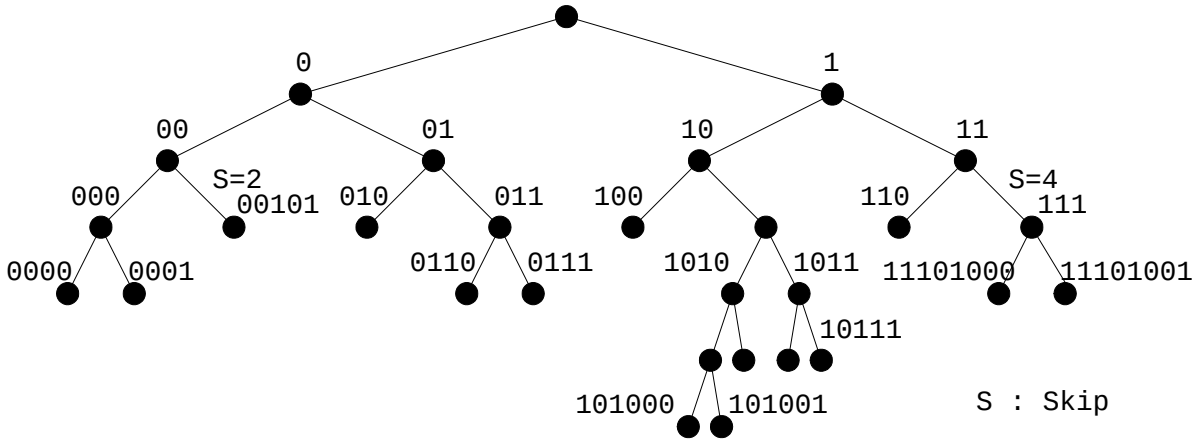


図 2.4 Path Compressed Trie

この問題を解決する方法が Level Compress 法である。Level Compress 法は各ノードが 2^k の子ノードを保持する MultiWay ツリー構造をとることでツリーを深さ方向にさらに圧

縮する方式であり、

- 検索木の深さはフォワーディングテーブルのエントリ数に対し $O(\log \log n)$ で増加する。
- 検索木の深さはアドレス帳と独立しているため、大きなプレフィックスにも対応。
- 同じデータ構造でユニキャスト、マルチキャストに対応可能。

という特徴を持つ。これより得られた LC(Level Compressed)-Trie は平均深度の劇的な削減を可能とし、ツリー検索に必要なメモリアクセス回数の削減およびメモリ使用量の削減が望める。

図 2.4 の Pass Compressed Trie を Level Compress した図を 2.5 に示す。

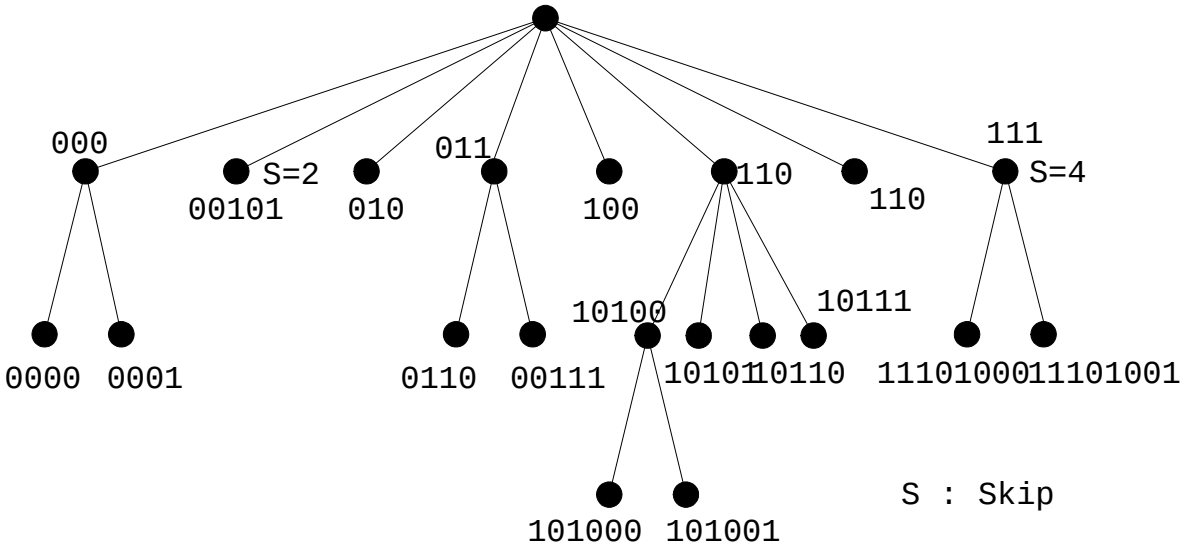


図 2.5 Level Compressed Trie

LC-Trie の各ノードは検索処理を可能とするため、以下のデータを持つ。

- Branch： 値が k のとき、ノードは 2^k 個の子ノードを持つことを示す。また、$k = 0$ のときは、ノードが終端 (リーフ) であることを示す。
- Skip： Path Compress を行った際に省かれた深さ方向のノード数を示す。検索時に検索 bit を Skip 分だけ飛ばす。
- Pointer： ノードが持つ一つの子ノードへのポインタである。図 2.5 の LC-Trie 木を想

定した場合、ポインタはノードが持つ子ノードの左端を指す。リーフの場合は、プレフィックスを格納しているテーブルへのポインタを返す。

図 2.5 のデータ構造を配列表現を表 2.1 に示す。また、そのときの BaseVector を表 2.2 に示す。

上記の LC-Trie 木で、入力値 $10110111_{(2)}$ の場合の検索例を説明する。まず、表 2.1 のノード番号 0 より、Skip=0 なので入力値の先頭から見て、Branch=3 より入力値の 3bit 分を参照する。それから、3bit 分より得た値 5 と Pointer=1 との加算を行い、6 を得る。これで、1 レベル分の分岐が完了する。次に、前レベルの計算結果が 6 なので、ノード番号 6 を参照する。Skip=0、Branch=2 より入力値の先頭 4bit 目から 2bit を取得し、その値と Pointer の加算より 15 を得る。そして、加算結果 15 を参照する。このとき Branch=0 であることから、検索木の終端に達したことになる。そこで、Pointer=10 が指す値を表 2.2 から取得し、得たプレフィックス 10110 と入力値 5bit 分を比較する。比較した結果一致したので検索は成功したことになる。

ルーティングテーブルを LC-Trie に写像する場合、各リーフは BaseVector と呼ばれるテーブルへのポインタを持つ。BaseVector には、

- プレフィックス
- NextHopTable(NextHop を格納したテーブル) へのポインタ
- PrefixVector(プレフィックス再検査用テーブル) へのポインタ

が格納されている。

宛先アドレスとプレフィックスが一致すると、NextHop ポインタにより NextHopTable 内から NextHop を取得する。一致しないときは、PrefixVector へのポインタの指す位置に格納されているプレフィックスとの比較を行う。ポインタが-1 の場合は、ルーティングテーブルに宛先アドレスと一致するプレフィックスが存在しないことを示す。

LPM 検索を行うときメモリアクセスの大部分は LC-Trie 木の検索で起こる。IPv4 の場合、LC-Trie 木の検索における平均深度は 2 以下であり、各ノードは 1 回のメモリアクセス

表 2.1 LC-Trie 木の配列表現

	Branch	Skip	Pointer
0	3	0	1
1	1	0	9
2	0	2	2
3	0	0	3
4	1	0	11
5	0	0	6
6	2	0	13
7	0	0	12
8	1	4	17
9	0	0	0
10	0	0	1
11	0	0	4
12	0	0	5
13	1	0	19
14	0	0	9
15	0	0	10
16	0	0	11
17	0	0	13
18	0	0	14
19	0	0	7
20	0	0	8

表 2.2 プレフィックスの配列表現

nbr	string
0	0000
1	0001
2	00101
3	010
4	0110
5	0111
6	100
7	101000
8	101001
9	10101
10	10110
11	10111
12	110
13	11101000
14	11101001

で参照される。次に、BaseVector 参照のためメモリアクセスが行われる。ここで、宛先 IP アドレスと BaseVector に格納されているプレフィックスが一致すれば、NextHopTable にて NextHop を得るため最終メモリアクセスが行われる。もし、一致しなければ PrefixVector を参照するため、数回メモリアクセスが起こる。ただし、PrefixVector 内での参照頻度は低く、1 回以上のメモリアクセス以上起こることは希である。

表 2.3 各方式におけるメモリアクセス回数、メモリ使用量、柔軟性

	メモリアクセス回数	メモリ使用量	柔軟性
キャッシュ方式	少	多	×
ハードウェア方式	少	少	×
ツリー方式	多	多	○
ツリー方式 (LC-Trie)	少	少	○

2.5 結言

既存ルータで実現されている IP アドレス検索方式について説明し、その問題点を述べた。

IP アドレス検索処理のアルゴリズムのボトルネックであるメモリアクセス回数に注目した場合、現在用いられている検索方式においてはキャッシュを用いた方式もしくはハードウェアによる方式が挙げられる。しかし、表 2.3 が示すように、キャッシュ方式はメモリ使用量が多い上、プレフィックス分布に依存したアルゴリズム構造のため柔軟性が低い。ハードウェア方式は、メモリ使用量は少ないが、柔軟性に関してはキャッシュ方式同様低い。一方、ツリー方式は、柔軟性は高いものの、メモリ回数およびメモリ使用量は多くなってしまう。ただし、LC-Trie を適用することで柔軟性を維持しつつ、メモリアクセス回数、メモリ使用量の削減が図れる。

第 3 章

データ駆動プロセッサによる実現法

3.1 緒言

超高速 IP アドレス検索を実現は、検索アルゴリズムにおけるボトルネックであるメモリアクセス回数を削減することにより可能である。本研究では、2 章よりメモリアクセス回数およびメモリ使用量が少なく、適応性のある LC-Trie 法を採用する。実現法としては、高速処理が可能で 複数プロトコル処理をソフトウェアにより実現可能な柔軟なアーキテクチャとしてデータ駆動プロセッサを用いた。データ駆動プロセッサは自己タイミングパイプライン機構により構成されているため、アルゴリズムを実装すると、自然にパイプライン並列化が行われる。LC-Trie 法における複数メモリアクセスがパイプライン化することにより、検索遅延時間に依存せずに検索レートを向上できる。

本章では、現行データ駆動プロセッサ上で LC-Trie 法を実現し、そのデータを基にさらなる高速化を図るために追加した命令について説明する。そして、追加命令を効率よく動作させるために構成されたプロセッサコアについて説明する。

3.2 LC-Trie のパイプライン並列化

LC-Trie を用いた LPM 手法の場合、メモリアクセスは

1. LC-Trie 検索
2. Prefix 検査
3. Prefix 再検査

4. NextHop 参照

の各処理で起こる。

1 メモリアクセスを 1 ステージとし、パイプライン化したフローを図 3.1 に示す。LC-Trie 検索部分は、IPv4 を LC-Trie 木に写像した場合、最大 5 レベルの検索深度で検索可能であることから、5 つの検索ステージに分けてある。検索が終了次第、LC-Trie 検索ステージを抜け、Prefix 検査に移行する。Prefix 検査の結果プレフィックスが一致したならば、NextHop 取得ステージに移行する。一致しない場合は Prefix 再検査ステージに移行し、しかる後に NextHop 取得ステージに移行する。

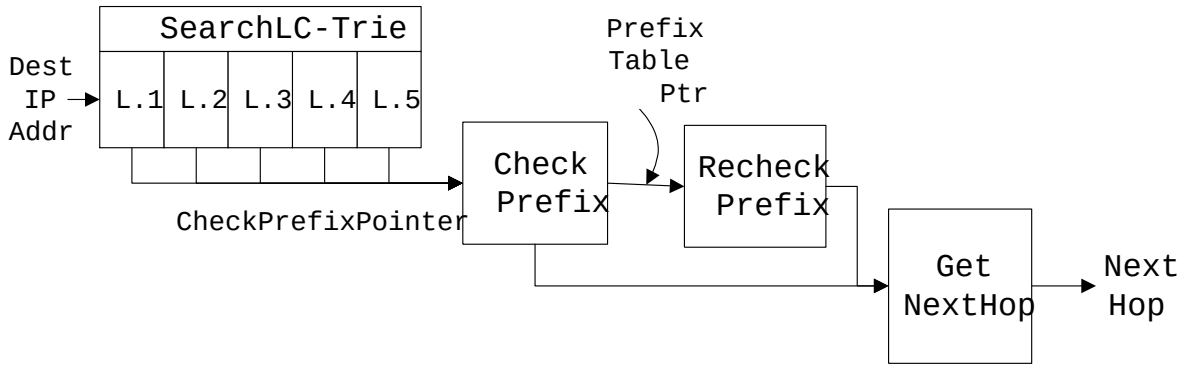


図 3.1　パイプライン化した LC-Trie

データ駆動プロセッサでは、その特性上自然にパイプライン化が図れる。そこで、LC-Trie 検索をメモリアクセスのある処理ごとにモジュール化し、現行のデータ駆動プロセッサ DDMP-4G 上で動作するように設計した。このときの命令接続線数 (arc)、およびランク (rank) を表 3.1 に示す。

汎用プロセッサを使用した UltraSPARC II 450MHz と現行データ駆動プロセッサ DDMP の平均検索レート、および遅延時間の比較を行う。なお、DDMP は 2 つの整数演算プロセッサコア (INTnPE) と 1 つのテーブル演算プロセッサコア (TBLnPE) を持つ。プロセッサコアの仕様を表 3.2 に示す。

表 3.1 使用命令数と命令接続線数、およびランク

(a) LC-Trie 検索レベル 1

	INTnPE	TBLnPE
1 入力命令	12	0
2 入力命令	2	1
arc	19	2
rank	7	1

(b) LC-Trie 検索レベル 2〜5

	INTnPE	TBLnPE
1 入力命令	13	0
2 入力命令	8	1
arc	29	2
rank	10	1

(c) Prefix 検査、Prefix 再検査

	INTnPE	TBLnPE
1 入力命令	8	0
2 入力命令	4	1
arc	16	2
rank	6	1

(d) NextHop 出力

	INTnPE	TBLnPE
1 入力命令	1	0
2 入力命令	0	1
arc	2	1
rank	1	1

表 3.2 データ駆動プロセッサ DDMP 仕様

	動作速度	ステージ数
INTnPE	120	30
TBLnPE	40	35

動作速度：MHz, 1 ステージ：5nsec

検索レート

$$\sum_{l=1}^{5} min(\frac{120MHz}{INTarc} \times INTnPE, \frac{40MHz}{TBLarc} \times TBLnPE) \times LevelFreq$$

遅延時間

$$\sum_{l=1}^{5} (INTrank \times INTstage + TBLrank \times TBLstage) \times StageDelay \times LevelFreq$$

とすると、

表 3.3 汎用プロセッサおよび DDMP における検索レート、遅延時間

	UltraSPARC II 450MHz	DDMP 4G
検索レート	3.8	2.5
遅延時間	11.0	4.65

検索レート:MLookup/sec, 遅延時間:msec

これより、検索遅延時間は短縮されるものの、検索レートは低い結果が得られた。これは、プログラム中における命令数の多さが原因である。次節で IP アドレス検索に適した命令を追加することで、プログラム中の命令数を減らしクリティカル長を短くする。

3.3 複合命令

3.2 で述べた実現では、命令セットが単純なためプログラム全体の命令数およびクリティカルパス長が大きくなり、LPM 検索を効率良く実現できない。そこで、

1. 命令の複合化によりプログラム中の命令数を極小化し、検索レートの向上と同時に検索時間の短縮を図る。
2. 異なるプロトコルにより宛先アドレスのデータ長が異なっても柔軟に対応できるように、入出力データを IP アドレスと Trie 検索に必要なデータに分けて扱う。

の 2 点に留意して IP アドレス検索用命令を追加した。

3.3.1 SearchTrie 命令

SearchTrie 命令は、LC-Trie 木の 1 レベル分の検索を行うための複合命令である (図 3.2)。

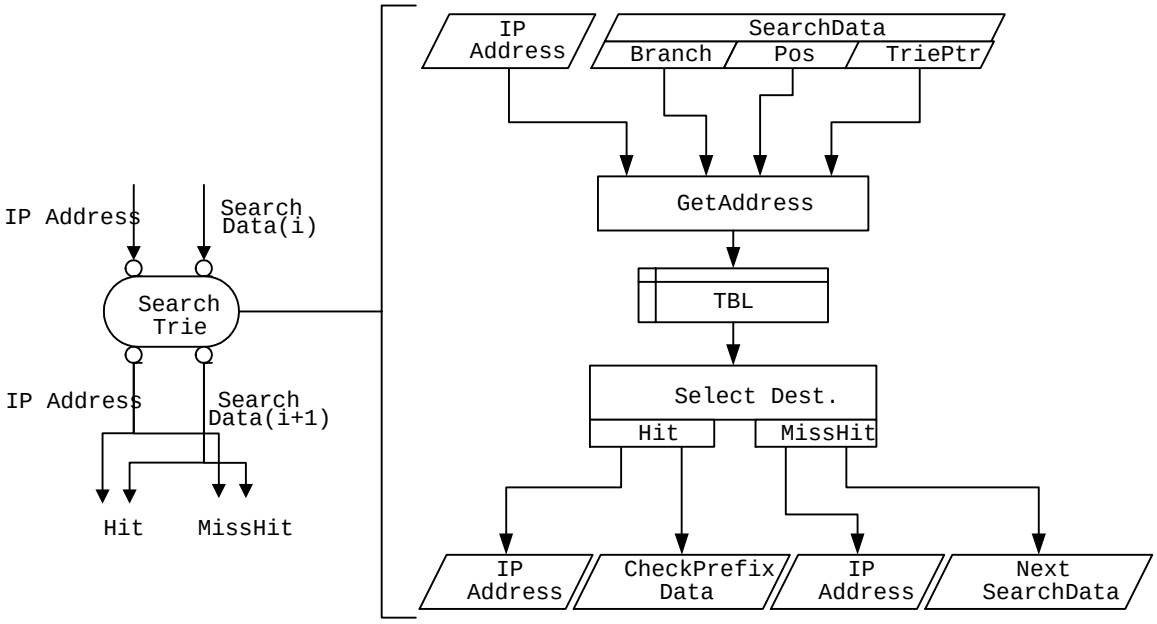
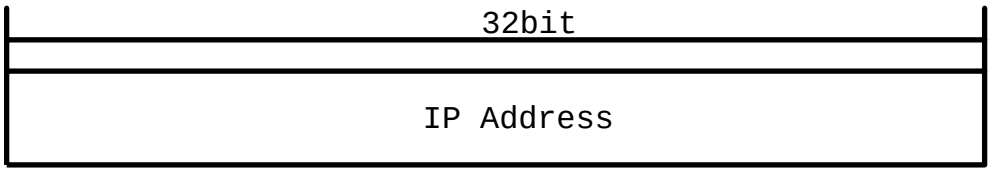


図 3.2 SearchTrie 命令

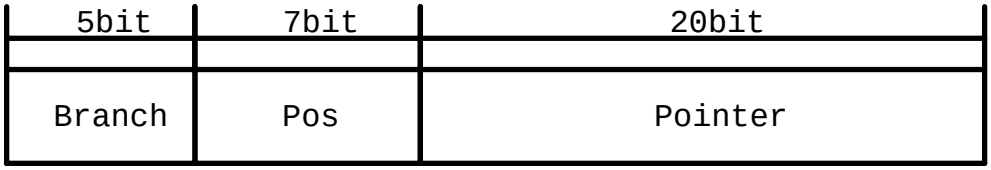
IP アドレス (図 3.3-(a)) と LC-Trie 検索用データ (図 3.3-(b)) が入力されると、まず、GetAddress で必要なノードが格納されているテーブルの位置を計算し、TBL(図 3.4) でノード情報を取得する。そして、Select Dest で得られたノード情報を元に検索が終了したかどうかを判定する。検索が終了した場合は、IP アドレスとプレフィックス検査用データ (図 3.3-(c)) を出力する。検索が終了しない場合は、IP アドレスと次の検索用データを出力する。

この命令において、IP アドレスと SearchTrie 検索用データを分けて扱っているのは、図 3.3 にあるように 1 つのデータの長さが 32bit であり、現行のデータ駆動プロセッサのバンド幅が 32bit であるため、変更なしに追加が可能であるためである。さらに、今後異なるプロトコルにより宛先アドレスのデータ長が異なっても柔軟に対応できるようにするためである。

(a) IP Address



(b) Seach Data



(c) CheckPrefixPointer

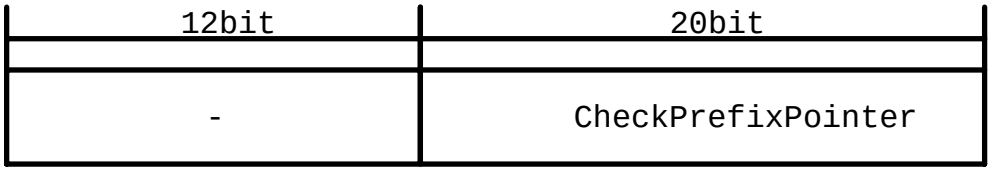


図 3.3 SearchTrie 命令入出力データ構造

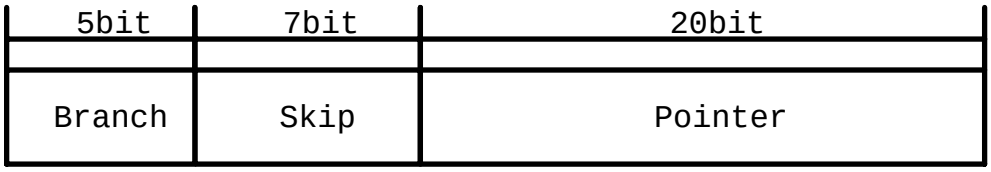


図 3.4 SearchTrie 命令用テーブルデータ構造

3.3.2 CheckPrefix 命令

CheckPrefix 命令は、SearchTrie 命令によって得られたプレフィックスの正当性を検査し、NextHop を出力する命令である (図 3.5)。

IP アドレス (図 3.6-(a)) と BaseVector(図 3.6-(b)) へのポインタが入力されると、TBL#0(図 3.7-(a)) から BaseVector に格納されたプレフィックスを、TBL#1(図 3.7-(b))

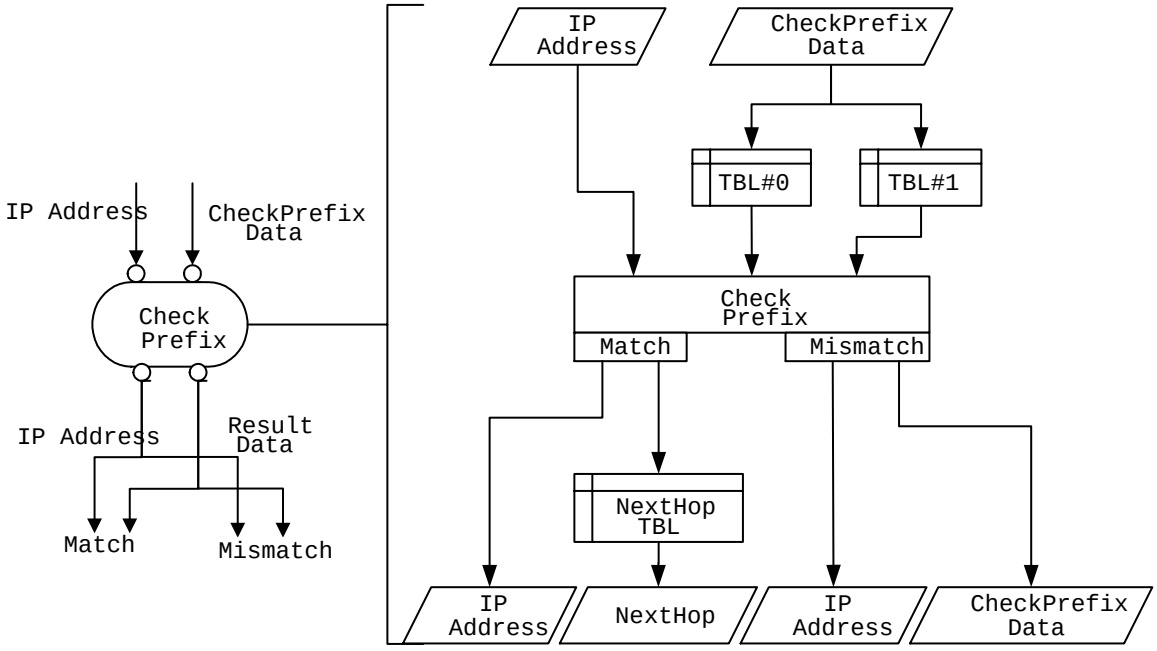


図 3.5 CheckPrefix 命令

からプレフィックス検査データ (図 3.6-(c)) を取得する。そして、CheckPrefix 部で入力された IP アドレスとプレフィックスを比較し、一致すれば NextHopTBL から NextHop を取得し、出力する。もし、一致しなければプレフィックスの再検査を行うため再度 Chkeck-PrefixPointer を出力する。

SearchPrefix 命令と同様の理由により、IP アドレスと SearchTrie 検索用データを分けて扱っている。

3.4 プロセッサコア構成

実現法の実装に際しては、専用ハードウェア実現法の欠点である柔軟性の欠如、汎用プロセッサによる実現法の欠点である低性能の両者を克服するために、次の要件を満足するプロセッサコアを構成する必要がある。

- 必要最低限のハードウェア追加で複合命令を実現でき、既存の命令セットも実行可能なこと。

- メモリアクセスの並列パイプライン化によって、メモリ参照遅延を隠蔽できること。
- ルーティングテーブル内のプレフィックス分布に応じて、最適なプログラム、およびメモリ割当てを柔軟に設定可能なこと。

本研究で使用するデータ駆動プロセッサは、整数演算用プロセッサコア、およびテーブル参照用プロセッサコアから成るマルチプロセッサコア構成で成っている。基本構成は現行のデータ駆動プロセッサ DDMP と同等であり、整数演算用プロセッサコアは、既存の整数演算用プロセッサコアを使用する。IP アドレス検索処理において最も使用頻度の高いテーブル参照プロセッサコアは、既存のテーブル参照命令セットを継承しつつ、高速化のための複合命令を追加し、さらにメモリアクセス I/F の変更を行っている。

3.4.1 テーブル参照ナノ PE

テーブル参照ナノ PE(図 3.8) は、既存のテーブル参照用プロセッサコアに LC-Trie 検索用複合命令 SearchTrie を追加実装したものである。このプロセッサコアはオンチップメモリ (内部テーブル) 用とオフチップメモリ (外部テーブル) 用のアクセス I/F を有する。

SearchTrie 命令としては、内部テーブルアクセス用と外部テーブルアクセス用がある。これによって、LC-Trie 構造中の各レベル毎のデータ数、参照頻度に応じて内部テーブルもしくは外部テーブルに格納された Trie データを、プログラムによって選択的にアクセスする (図 3.9)。

3.4.2 拡張テーブル参照ナノ PE 構成

拡張テーブル参照専用ナノ PE 構成 (図 3.10) は 2 つの外部テーブルへのアクセス I/F と 1 つの内部テーブルを有する拡張プロセッサコアである。拡張テーブル参照専用ナノ PE では Prefix 検査に必要なデータを外部テーブルから取得する。このとき、2 つの外部テーブルに同時並行にアクセスし、1 回のメモリアクセスと同等の時間で 2 ワードを参照する。

MM に一組の IP アドレスと Prefix 検査用ポインタが到達すると、Prefix 検査用データは

MAU#0、MAU#1 に送られ、それぞれ外部テーブルよりプレフィックスとプレフィックス検査用データを取得する。プレフィックス検査用データは、プレフィックス長、NextHopTable へのポインタ、及び Prefix 再検査用ポインタから成る。次に、IP アドレスとプレフィックス検査用データは SBV により比較され、一致した場合は NextHopTable への IP アドレスと NextHopTable へのポインタを出力する。一致しない場合は IP アドレスと Prefix 再検査用ポインタを出力する。比較が一致した場合は、さらに MAU#2 より内部テーブルに格納されている、NextHop を取得する。

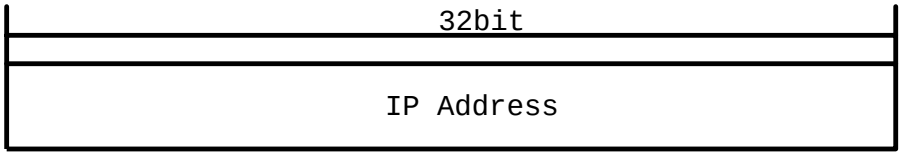
3.4.3 プロセッサコアの接続構成

各プロセッサコアはプロセッサコア間でデータフローの制御を行うルータ (Interprocessor Router) によって接続されており、複数個のプロセッサコアの接続が可能なマルチプロセッサコア構成をとることができる。

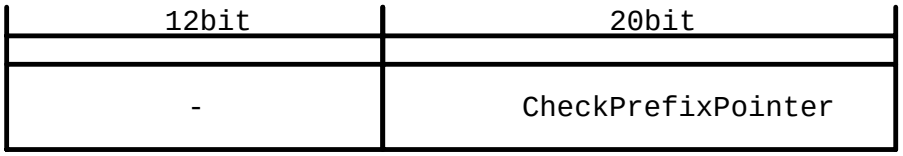
3.5 結言

本章では、LC-Trie をデータ駆動プロセッサ上でパイプライン並列に実現する方法について述べた。そして、超高速 IP アドレス検索を行うために、LC-Trie 木を検索する SearchTrie 命令と、Prefix 検査を行い NextHop を出力する CheckPrefix 命令の 2 種類の複合命令を新たに追加実装した。さらに、SearchTrie 命令をテーブル参照プロセッサコアの命令セットに追加し、CheckPrefix 命令を実行するために拡張テーブル参照プロセッサコアを構成した。

(a) IP Address



(b) CheckPrefixPointer



(c) Next Hop

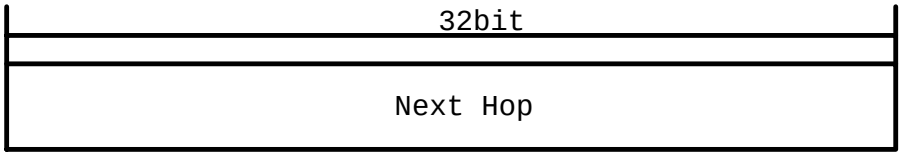
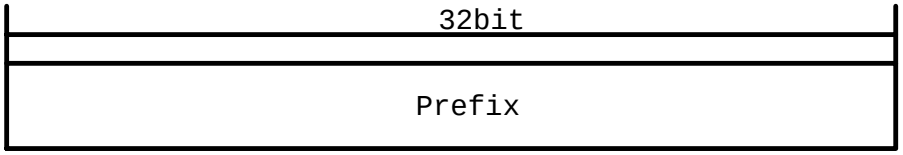


図 3.6 CheckPrefix 命令入出力データ構造

(a) Prefix



(b) CheckPrefixData

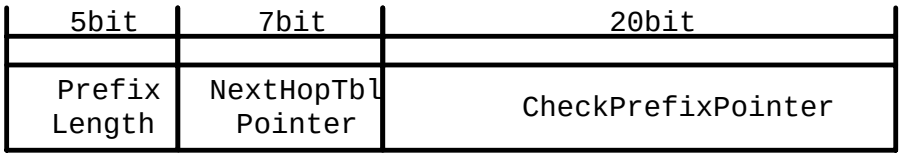


図 3.7 CheckPrefix 命令用テーブルデータ構造

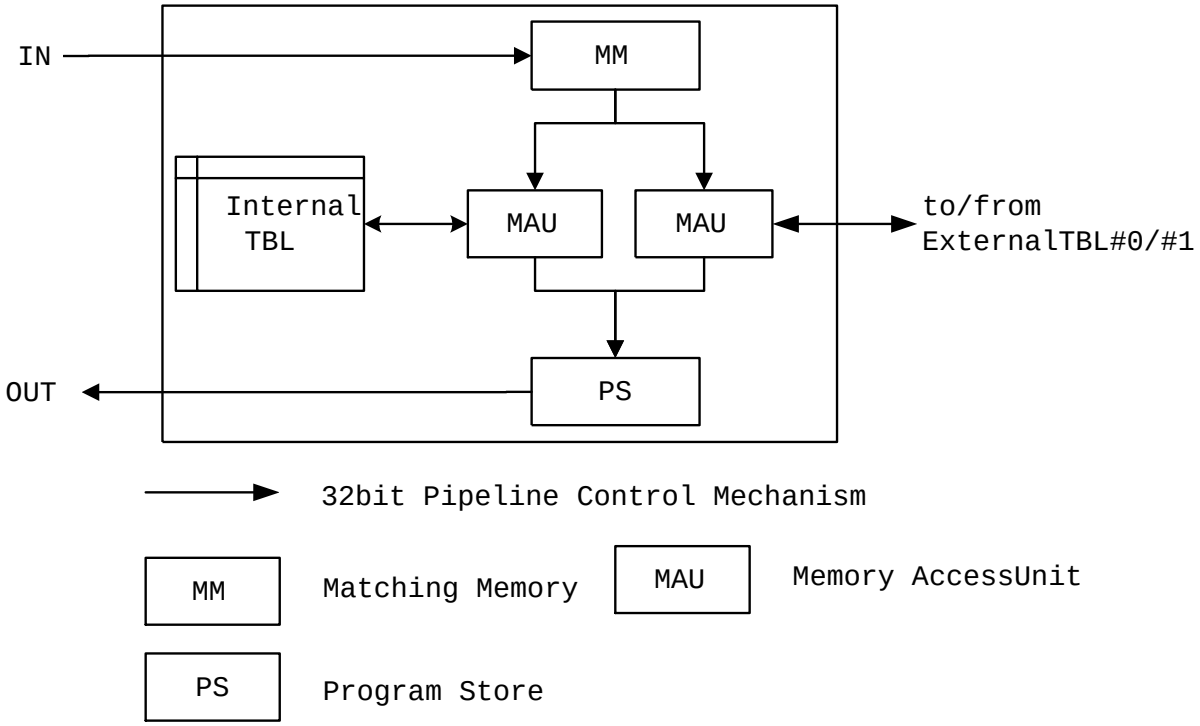


図 3.8 テーブル参照専用ナノ PE 構成

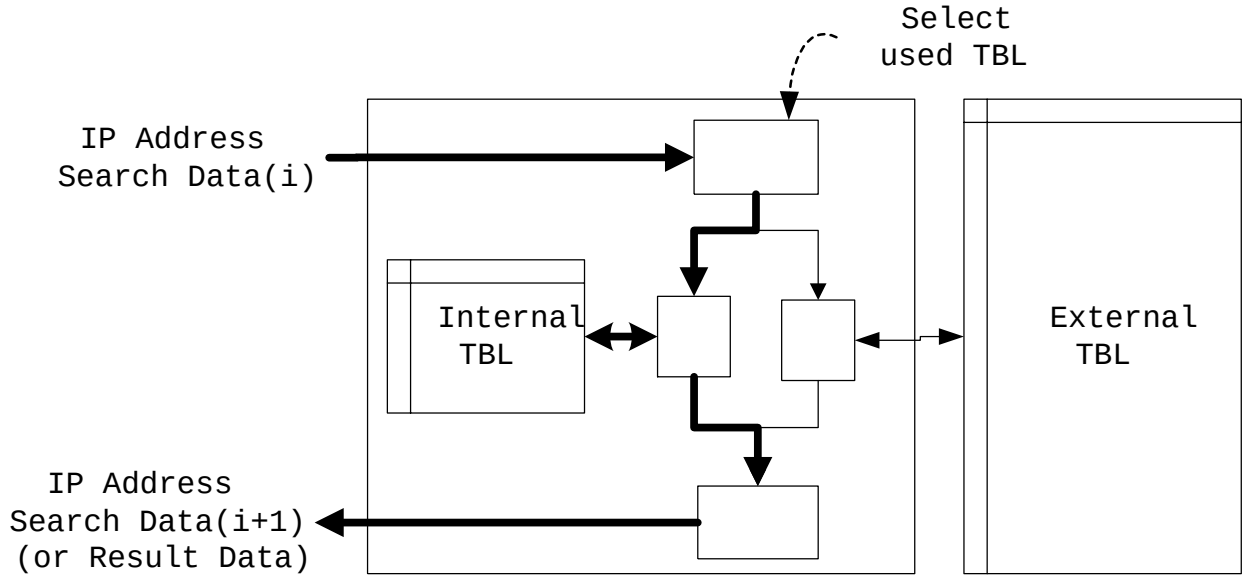


図 3.9 内部テーブル使用時のデータの流れ

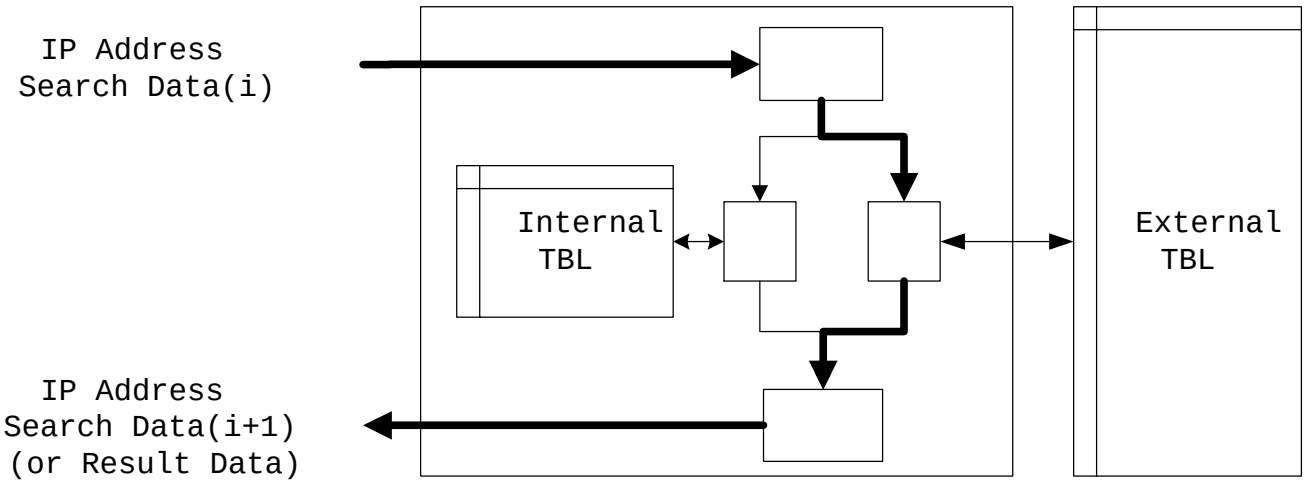


図 3.10　拡張テーブル参照専用ナノ PE 構成と流れるデータ

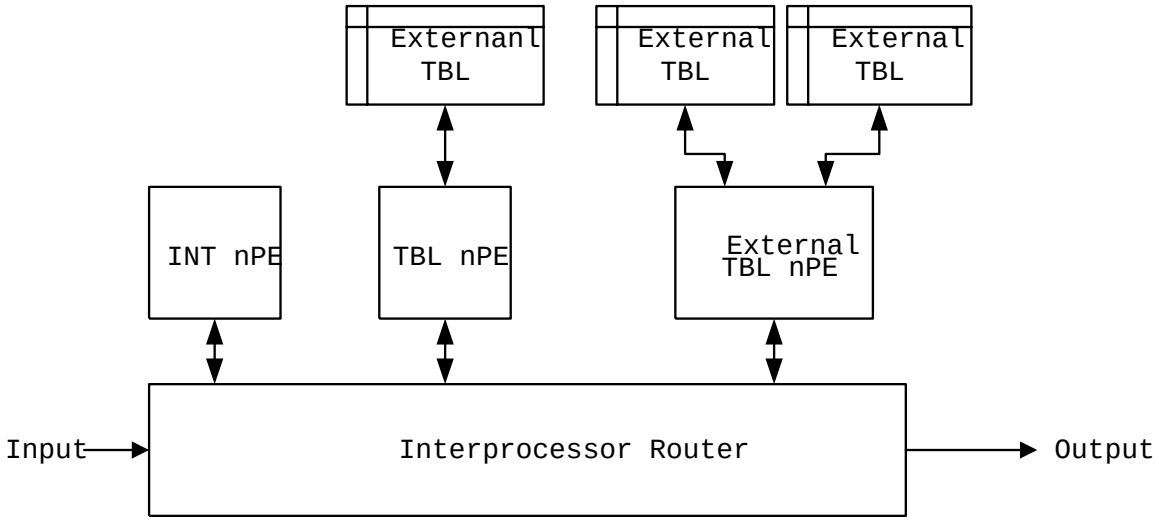


図 3.11　プロセッサコアの接続関係

第 4 章

性能評価

4.1 緒言

本実現法によって、大規模なルーティングテーブルに対しても高い LPM 検索レートを達成可能なことを確認するために、バックボーンのルーティングテーブルを対象に既存の汎用プロセッサ、現行データ駆動プロセッサ、および本実現法での平均検索レートの比較を行う。さらに、ルーティングテーブルが増大した場合の本実現法における平均検索レートの変化、必要内部メモリ容量の増加について評価する。

4.2 評価モデル

対象とするルーティングテーブルは、北米インターネットバックボーンノード MaeEast の IPv4 データ、55K エントリを使用した。さらに、ルーティングテーブルの増加に伴う検索レートの推移を確認するために、110K エントリおよび 550K エントリのルーティングテーブルを作成し用いた。

汎用プロセッサは、Dec Alpha 600MHz を搭載した UltraSPARCII 上で、C 言語により記述された検索アルゴリズムを用いた。現行データ駆動プロセッサである DDMP-4G では、専用開発環境でプログラムを行い、シミュレータを用いて性能を求めた。本実現法は、現行 DDMP と同等の集積化技術を用いると仮定して、プロセッサ内の処理対象データのビット長 32bit、自己タイミング型パイプラインの動作速度 150M データ/秒、内部テーブル容量は各テーブル参照用プロセッサコアに 128KB、拡張テーブル参照専用に 1KB、外部テーブ

ルへのアクセスレート 66M ワード/秒とした時の値を求めた。

平均検索レートは M 検索/秒、メモリ容量は KByte 単位とする。

4.3 評価内容

ルーティングテーブルのエントリが、統べて内部テーブルに格納可能であるとすると、外部テーブルへのアクセスは拡張テーブル参照プロセッサコアでの Prefix 検査処理でのみ発生する。このとき、Prefix 検査処理は 2 つの外部テーブルへアクセスするが、2 つは互いに独立しており、アクセスも独立して行われる上、テーブル参照プロセッサコアからのアクセスがないため、外部テーブルアクセスレートを最大限に維持することが可能である。

また、MaeEast 等のルーティングテーブルは巨大なため、内部テーブルにすべて格納しきれない。そこで、最も頻繁にアクセスされる LC-Trie 検索木のレベル 1 のみを 2 つのテーブル参照プロセッサコア内の内部テーブルに分割格納し、レベル 2 からレベル 5 までは外部テーブルに格納する。このとき、IP アドレス検索遅延時間は、内部テーブルへのアクセス遅延が外部テーブルへのアクセス遅延を越えないようにルーティングテーブルを格納配置すると、外部テーブルへのメモリアクセスに制約される。外部テーブルへのアクセスはテーブル参照プロセッサコアと拡張テーブル参照プロセッサコアの両方から発生するため、メモリアクセスの調停が起こりメモリアクセスレートが下がる。

本研究において提案したプロセッサは、現行データ駆動プロセッサのマルチプロセッサコア構成を継承しており、7 個の整数演算用プロセッサコア、2 個のテーブル参照プロセッサコア、および 1 個の拡張テーブル参照プロセッサコアから成る (図 4.1)。

また、テーブル参照プロセッサコアと拡張テーブル参照プロセッサコアは、オンチップメモリ (内部テーブル) とオフチップメモリ (外部テーブル) へのアクセスが可能である。本プロセッサは外部テーブルとして 2 系統の独立したメモリを搭載している。1 つの外部テーブルには、テーブル参照プロセッサコアから 1 つ、拡張テーブル参照プロセッサコアから 1 つの I/F があり、各プロセッサコアからのアクセスは調停回路によって制御される。

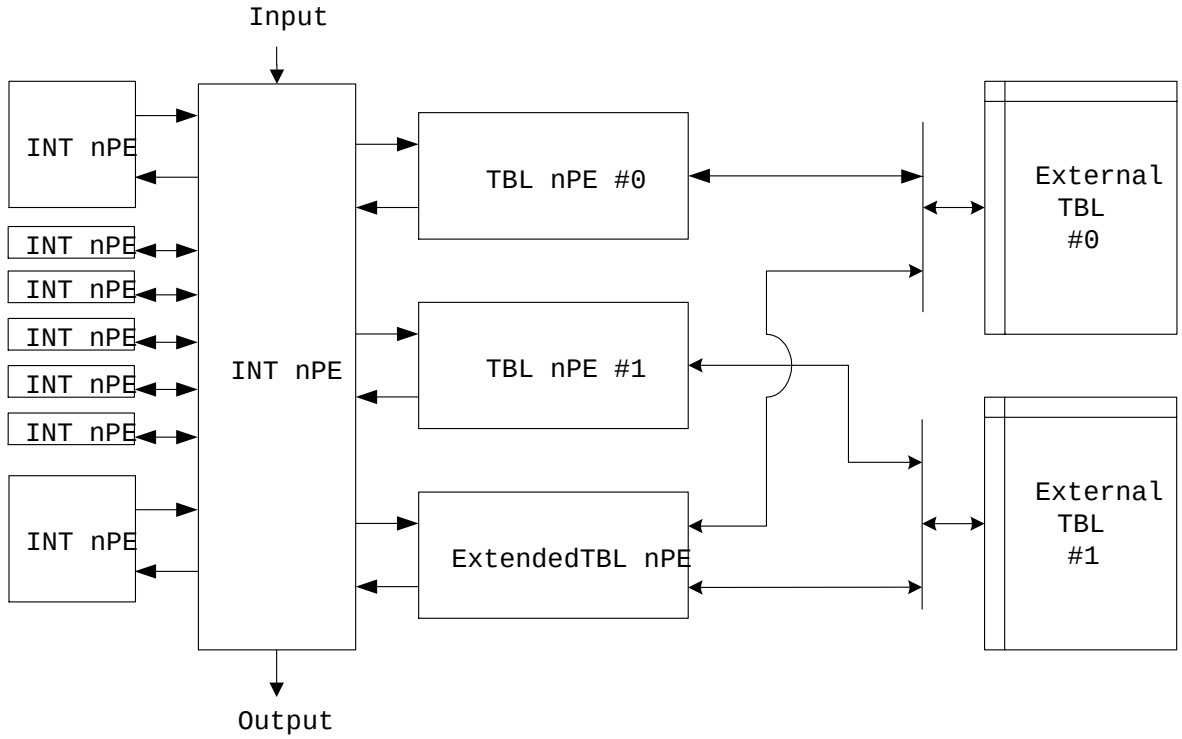


図 4.1　マルチプロセッサコア構成

MaeEast のルーティングテーブルを LC-Trie に写像したとき、各レベルの平均検索頻度は表 4.1 のようになる。

このとき、Prefix 再検査におけるメモリアクセスは全体の検索処理に対して、きわめて少ないので数字上誤差範囲内であるため、省略してある。

表 4.1　MaeEast ルーティングテーブルにおけるレベル毎の平均検索頻度

Level	Leaves
1	62038.33
2	31087.67
3	33839.83
4	3951.00
5	126.67

以上のことをふまえて、IP アドレス検索を現行汎用プロセッサ Dec Alpha 600MHz 上で行った場合、現行データ駆動プロセッサ DDMP-4G 上で行った場合、および本実現法で行った場合について検証した。検証結果を表??に示す。

表 4.2　平均検索レート

Dec Alpha 400MHz	DDMP 4G	本実現法	本実現法 (内部 TBL のみ)
2.5	17.5	51.06	66.00

また、ルーティングテーブルが増大したときの、平均検索レートと必要内部メモリ容量の比較を行い、表 4.3 に示す結果が得られた。

表 4.3　平均検索レートと必要内部メモリ容量

エントリ (KEntry)	55	110	550
平均検索レート	51.06	43.28	36.68
内部メモリ容量	242	260	1572

この結果より、本実現法を用いた場合、ルーティングテーブルの増加に伴い、必要な内部メモリ容量は増加するが、平均検索レートの低下はきわめて緩やかであることが確認された。

以上のことより、最大検索レート 66M 検索/秒、平均検索レート 51M 検索/秒が達成可能であり、最小 IP パケット長 47 バイトの場合でも、18Gbps〜13Gbps の性能を達成できる可能性があることが確認された。

4.4　結言

本実現法の実装によって、大規模なルーティングテーブルに対しても高い LPM 検索レートを達成可能なことを確認するために、バックボーンのルーティングテーブルを対象に性能

を見積もった。

その結果、本実現法を用いた場合、最大約 66Mlookup/sec、平均で 51Mlookup/sec の検索レートを達成することが確認できた。さらに、ルーティングテーブルの増加に伴い、必要な内部メモリ容量は増加するが、最小 IP パケット長 47 バイトの場合でも、18Gbps〜13Gbps、平均 IP パケット等 188 バイトの場合、72Gbps〜52Gbps の性能を達成できる可能性があることが確認された。

なお、今回は現行データ駆動プロセッサの性能仕様を基に、提案した実現法の性能を求めた。よって、今後さらに厳密なシミュレートを行って、性能を実証する。

第 5 章

結論

ネットワークにおけるデータ通信量の増大に伴い、伝送路は DWDM 等の技術により飛躍的に性能が向上している。一方で、伝送路を収容するノードにおいてはルータの処理速度、特に IP アドレス検索がボトルネックとなっており、高速化が急務となっている。

本研究では、超高速伝送路を収容可能な超高速ルータに必須である IP アドレス検索のデータ駆動方式による高速化の実現法を提案した。

まず、2 章では既存のルータで用いられている IP アドレス検索法について主にキャッシュ検索方式、ハードウェアによる完全一致検索方式、ツリー検索方式について具体例を挙げて、各方式の動作原理を説明し、高速化するにあたっての問題点を述べた。そして、本実現法で採用した LC-Trie 法の動作原理を説明し、その有効性について述べた。

3 章では、複数メモリアクセスをパイプライン並列で処理することにより検索遅延時間に依存せずに検索レートの向上が可能であることに着目し、LC-Trie 検索法をパイプライン並列化したアルゴリズムを説明した。そして、現行データ駆動プロセッサ上で実現した場合の使用命令数、命令接続線数、ランクを示し、メモリアクセスを行う命令に対する周辺処理回数の多さから、パイプライン効率が芳しくないことを指摘した。そこで、パイプライン効率の向上を図るために検索処理の複合命令を提案した。さらに、プロセッサ内のマルチプロセッサコアの構成を行い、複合命令を効果的に動作させるために最適なプロセッサコアの構成を提案した。

4 章では、今回提案した実現法の実現可能性を実証するために、現行のデータ駆動プロセッサの性能を基に、本実現法により大手バックボーンルータのルーティングテーブルを用いて IP アドレス検索を行った場合の検索レートを求めた。さらに、ルーティングエントリ

が増大したときの平均アクセスレートの変化と必要内部メモリの容量の増加を確認した。

今後の課題としては、今回提案したモデルは、かなり処理対象を簡略化したものであるため、本実現法の詳細設計を行い、回路レベルの評価を実施する予定である。また、今後主流になるであろう IPv6 への対応を考慮し、プロセッサコア構成の最適化および検索アルゴリズムの最適化も同時に行う予定である。

他にも、本実現法によりバックボーンノードからアクセスノードまでプログラムを更新することで同一のプロセッサにより対応可能なため、製造コストが削減できる。さらに、アクセスノードなど IP アドレス検索に多くのリソースを必要としないルーティング環境においては、同一プロセッサに QoS やスケジューリング処理を与えることができ、1 チップでルータ機能を提供することができる。また、大きなデータの取り扱いが必要な場合に対しても、マルチプロセッサ構成をとることによって容易に対応可能である。

以上のことより、本実現法は、IP アドレス検索処理として実現可能な問題に対してはそのまま的要することができるため、基幹ルータをはじめ家庭用ルータにいたる幅広い応用が考えられる。

謝辞

本研究において、懇切なる御指導、御鞭撻を賜った 岩田 誠助教授 に心より感謝の意を表します。

本研究の基礎としている自己タイミング型データ駆動マルチプロセッサの考え方を提唱され、様々な御示唆を賜った 寺田 浩詔教授 に心より感謝の意を表します。

本研究を進めるにあたり、適切な御助言、御指導を賜った 大森 洋一助手 に深く感謝の意を表します。

本研究で使用したデータ駆動プロセッサシミュレータを提供して頂いたシャープ株式会社の方々ならびにパシフィックソフトウェア開発株式会社の関係各位に深く感謝の意を表します。

日頃から暖かい御支援を頂いた 細美 俊彦氏 に感謝の意を表します。

日頃から御支援頂いた情報システム工学科岩田研究室の方々、橋本 正和氏、別役 宣奉氏、古家 俊之氏、 前田 庸介氏、森安 亮氏に感謝の意を表します。

参考文献

[1] Hiroaki Terada, Souichi Miyata, Makoto Iwata, "DDMP's: self-timed super-pipelined data-driven multimedia processors", Proceedings of the IEEE, 87(2), 282-296, 1999.

[2] Marcel Waldvogel, George Varghese, Jon Turner, Bernhard Plattner, "Scalable High Speed IP Routing Lookups", ACM SIGCOMM '97, pp.26-36, 1997

[3] PankajGupta, Steven Lin, and Nick McKeown, "Routing Lookups in Hardware at Memory Access Speeds", IEEE INFOCOM '98, 10B-1, 1240-1247, 1998.

[4] Graig Partridge, Philip P. Carvey, Ed Burgess, Isidro Castineyra, Tom Clarke, Lise Graham, Michael Hathaway, Phil Herman, Allen King, Steve Kohalmi, Tracy Ma, John Mcallen, Trevor Mendez, Walter C. Milliken, Ronald Pettyjohn, John Rokosz, Joshua Seeger, Michael Sollins, Steve Storch, Benjamin Tober, Gregory D. Troxel, David Waitzman, and Scott Winterble, "A 50Gb/s IP Router", IEEE/ACM Transactions On Networking, 6(3), 1998.

[5] Stefan Nilsson and Gunnar Larlsson, "IP-address lookup using LC-Tries", IEEE Journal On Selected Areas In Communication, 17(6), 1083-1092, 1999.

[6] Mikael Degrmark, Andrej Brodnik, Svante Carlsson, and Stephen Pink, "Small Forwarding Tables for Fast Routing Lookups", ACM SIGCOMM '97, 27, 3-14, 1997.

[7] Henry Hong-Yi Tzeng and Tony Przygienda, "On Fast Address-Lookup Algorithms", IEEE Journal On Selected Areas In Communications, 17(6), 1067-1081, 1999.

[8] 村瀬勉, 小林正好, "Longest Prefix Match キャッシュを用いた IP フォワーディングテーブル検索方法の提案", 電子情報通信学会信学技報, SSE2000-20, 73-78, 2000.

[9] 羽賀雅則, 塩本公平, ”ポインタキャッシュを用いた最長一致に基づくルーティングテーブル検索法”, 電子情報通信学会信学技報, SSE98-26, 7-12, 1999.

[10] 小林正好, 村瀬勉, 小倉直志, 栗山敦, ”マルチギガビット IP フォワーダ用 50Mpps Longest Prefix Match 検索 LSI”, 電子情報通信学会信学技報, IN98-119, 7-12, 1998.

[11] 森川大智, 大森洋一, 岩田誠, ”データ駆動方式による IP アドレス検索処理の一実現法”, 平成 12 年電気関係学会四国支部連合大会, 12-19, 207, 2000

[12] 森川大智, 岩田誠, 寺田浩詔, ”IP アドレス検索処理の超高速パイプライン実現法”, 第 62 回情報処理学会全国大会, 5T-07, 2001.