低コスト・低消費電力 TCAM における効率的なルーティングテーブル管理法

電子情報通信学会　ネットワークシステム研究会　2008年1月

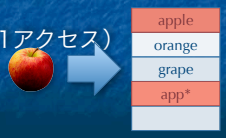
阿多 信吾（大阪市立大学）
黄 恵聖（大阪大学）
山本 耕次（ルネサステクノロジ）
井上 一成（ルネサステクノロジ）
村田 正幸（大阪大学）

発表の内容

- 研究の背景と目的
  - TCAM の概要とその問題点
  - 関連研究
  - 本研究の目指すもの
- 低コスト・消費電力を実現する TCAM
  - 概要
  - 技術的課題
- 効率的なルーティングテーブル格納のための インデックス決定法
  - 基本方針
  - アルゴリズム詳細

TCAM とは

- TCAM = Ternary Content Addressable Memory
  - RAM と逆の動作を実現する記憶素子
    - RAM:（入力）アドレス→（出力）内容
    - CAM:（入力）内容→（出力）アドレス
  - 完全一致に加え部分一致をサポート (*)
- 特徴
  - 情報検索が非常に高速（1アクセス）



TCAM の用途

- コア・エッジルータに幅広く利用
  - アドレス検索
    - パケットごとに宛先アドレスをチェックし、経路表に基づいてパケットを次ノードに転送
  - アクセス制御
    - 不正なトラヒックのパケットかどうかを決められたルールに基づき検証し、通過・拒否を決定
  - トラヒックモニタリング
    - パケットをフローごとに分類し、統計量を調査

TCAM によるアドレス検索

- 宛先アドレスはネットワークごとに集約
  - プレフィックス
    - 192.168.10.0/24
  - 最もプレフィックス長が長いエントリ検索

4.36.200.0/21	1101*
4.38.104.0/23	110010*
4.67.64.0/22	11101011*
4.71.0.8/30	01001010100*
4.78.192.96/27	…
4.78.192.112/28	11101001001010010011*
4.79.181.0/24	11011011110101110001*
4.128.0.0/9	
6.1.0.0/16	
6.2.0.0/22	
6.3.0.0/18	

TCAM の問題

- TCAM の構造的な問題
  - 高コスト
    - 歩留まりが悪い
  - 高消費電力
    - 検索ごとに全領域に通放電
- ルータの要求
  - ルーティングテーブルの増加
    - IPv4 で25万エントリ
  - アクセス制御リスト (ACL) の細分化
    - 必要最小限の通信のみ許可したい

TCAM の低消費電力化

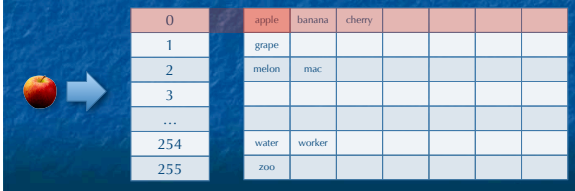
- エントリ全体を複数のグループに分割
  - グループごとに別々の CAM に格納
  - 検索キーから適切なグループを決定
    - 範囲による分割
    - ハッシュ値

問題点と本研究の目的

- 既存研究の問題点
  - TCAM デバイスは変更しない
    - 外部回路あるいはソフトウェアによる制御
    - I/O のボトルネック（速度・コスト）
    - ソフトウェアのオーバーヘッド
  - 分割ポリシーは単一
    - ルーティングテーブルの局所性に対応できない
- 本研究の目的
  - TCAM デバイス内で実現 (TACT)
    - 外から見れば従来 TCAM と同じ→更新が容易
  - より詳細な分割ポリシー

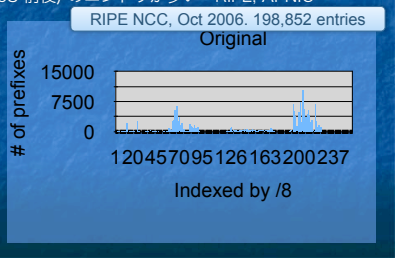
TACT の概要

- TCAM エントリ全体を分割
  - 256 サブグループ
  - 8ビットインデックステーブル によるサブグループ選択



インデックス値最適化の必要性

- 先頭８ビットによる分割
  - グループ間に大きな偏り
    - > 10,000 と 0 が混在
    - 一部 (63, 208 前後) のエントリが多い→RIPE, APNIC



インデックス値の決定アルゴリズム

- 目標
  - サブグループ内のエントリ数の最大値を最小化したい
    - 8ビットインデックスによる256分割
  - サブグループ間のエントリ数分布をできる限り平滑化
  - インデックスとして使用できる8ビットを選択

ハードウェアの概要

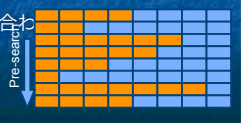
- TACT デバイス部分を複数並列化
  - TACT デバイスごとにインデックス値を変更
    - 3 TCAM デバイス
    - 4ビットインデックス
  - エントリを分割
    - プレフィックス長により３分割
      - /8〜/16
      - /16〜/24
      - /24〜/32
    - インデックスによりサブグループ化
      - ４ビット→16分割

テーブルの分割

- 均等分割 (Equal)
  - 1/n of prefixes are stored into single TACT
  - Simple, equal utilization, but little bit complex
- プレフィックス長による分割 (Prefix length based)
  - Partition by the length of prefix
  - Easy to implement, but low flexibility, uneven utilization
- 偏向分割 (Biased)
  - Use more TACTs to store shorter prefixes
  - Can reduce the maximum # of prefixes (show later)

平滑化のためのインデックス決定法

- 目的：サブグループ間エントリ数の平滑化
- 制約条件
  - プレフィックスに “* (Don’t care)” が含まれないこと
- 先頭ビットの選択
  - エントリ全体を二等分できるのが望ましい
  - “0”:”1” = 50%:50%
- ２ビット目の選択
  - 1ビット目と２ビット目を組み合わ
  - １ビット目との関係は？



インデックスビット選択例

- Six 4-bit prefixes
- For the first bit:
  - Count # of 1
  - (from left) 2, 4, 3, 2
  - 3rd is best (50% of total)
- How about second?
  - Consider not only count, but correlation with 1st

0100
0011
1100
0010
0101
1110

2ビットによる四分割

- 1ビット目と2ビット目の関連
  - 4つの値 (00, 01, 10, 11) が均等に現れる
  - 相関が強いと2ビット目で分割されない
  - 逆相関が強くても同じ
  - 相関が最も弱いビットがよいはず
  - 2ビット目も 0, 1 が 50%:50% であることが望ましい

Two conditions for selecting 2nd bit

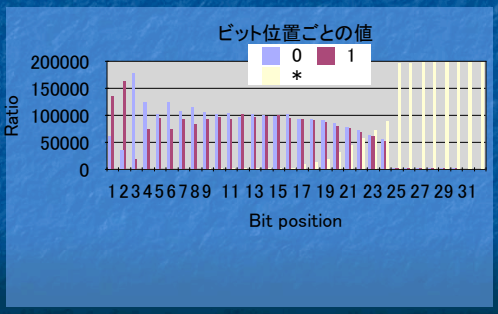
- Numbers of 0/1 are even
  - P_A = abs(count(bit[j]==“0”) - count(bit[j]==“1”))
  - Smaller P_A is better
- Correlation with 1st bit (i) is weakest
  - P_B = abs(count(bit[i]==bit[j]) - total/2)
  - Smaller P_B is better
- 2nd bit should satisfy both conditions
  - Both P_A, P_B should be smaller

インデックスビット決定アルゴリズム

- Selection of candidates
  - Ignore any positions if there is a prefix which has don’t care “*” at the position
- Selection of 1st bit
  - For each candidate count number of “0” and “1”
  - P_{Aj} = abs(count(bit[j]==“0”) - count(bit[j]==“1”))
  - Select position j if P_{Aj} is smallest
- Selection of 2nd and after
  - For each candidate j
    - P_{Aj} = abs(count(bit[j]==“0”) - count(bit[j]==“1”))
    - P_{Bj} = abs(count(bit[i]==bit[j]) - total/2), i: pos. of 1st
    - Choose position j where P_{Aj} P_{Bj} is smallest

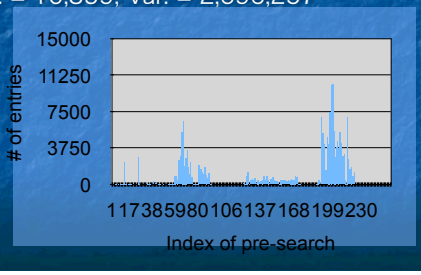
数値評価

- RIPE (198,852 entries)



単一インデックスの場合

- 先頭8ビット以外使用できない
  - /8 プレフィックスは 9 ビット目以降すべて *
  - Max. = 10,359, Var. = 2,696,257

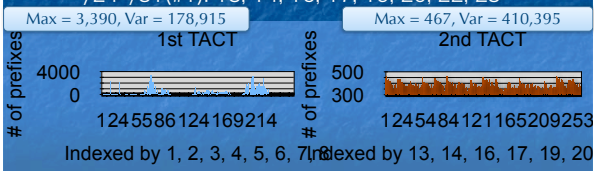


2 TACT デバイス

- Two groups having 99,492 entries
  - /1~/24 (#2), /24~/31 (#1)
- Selected bits
  - /1~/24 (#2): 1, 2, 3, 4, 5, 6, 7, 8
  - /24~/31(#1): 13, 14, 16, 17, 19, 20, 22, 23
- Results
  - #1: Max = 3,390, Var = 178,915
  - #2: Max = 467, Var = 410,395

2 TACT デバイスの場合

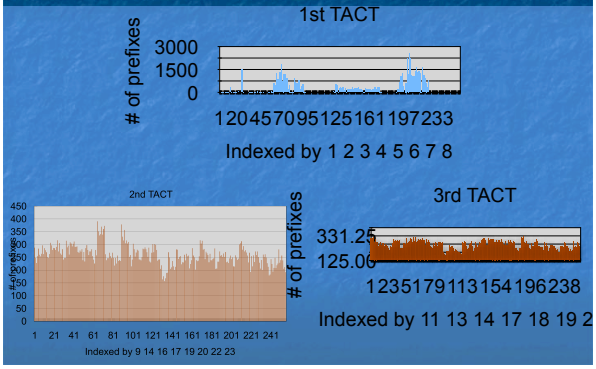
- 99,492 エントリずつ2分割
  - /1~/24 (#2), /24~/31 (#1)
- Selected bits
  - /1~/24 (#2): 1, 2, 3, 4, 5, 6, 7, 8
  - /24~/31(#1): 13, 14, 16, 17, 19, 20, 22, 23



3 TACT chips

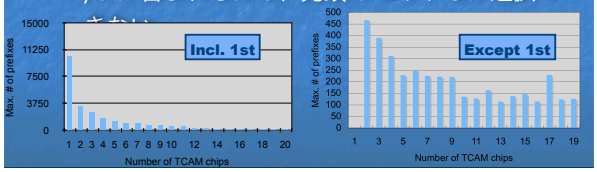
- Equal partition (66,328 entries)
  - #3: /1~/22
  - #2: /22~/24
  - #1: /24~/32
- Bit selection
  - #3: 1~8, 1 2 3 4 5 6 7 8
  - #2: 13~20, 9 14 16 17 19 20 22 23
  - #1: 14~21, 11 13 14 17 18 19 20 21
- Results
  - #3: Max = 2,562, Var = 204,893
  - #2: Max = 359, Var = 1,085
  - #1: Max = 326, Var = 795

3 TACT chips (cont'd)

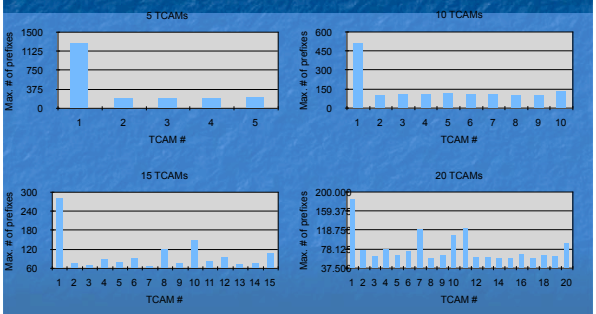


最適な TACT デバイスの数は？

- デバイス数の影響
  - デバイス数を増やすことで最大エントリ数を大幅に削減することは可能
  - 10デバイス程度が妥当
- 先頭デバイスのみ最大エントリ数が多い
  - /8 が含まれるため、先頭 8 ビットしか選択できない

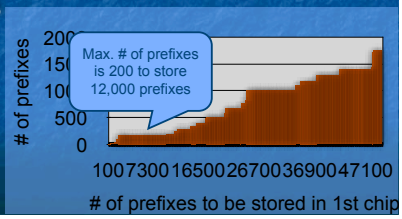


Distribution of max. # of prefixes



偏向分割 (biased partitioning)

- 先頭デバイスのみがばらつきが大きい
  - No bit selection: only 1-8
    - existence of /8 prefixes
  - 先頭デバイスへの格納プレフィックス数をより少なくする



まとめと今後の課題

- サブグループへの分割とインデックス決定の効率化により低コスト化
  - Max. number of prefixes can be reduced to about 1/100 (10,000 to 100) by using 10~20 TACTs
  - Short prefixes should be stored independently in order to reduce max. # of prefixes
- 今後の課題
  - テーブルの更新時における評価