2004－DPS－116　(16)
2004／1／30

# ポリシ先読みによるパラメータフィルタ高速化の試み

垣内 正年 †　森島 直人 †　砂原 秀樹 †

筆者らはルータにおける複雑なポリシを反映するパケット分類の問題に対し，KUPF アーキテクチャを提案・実装してきた．KUPF アーキテクチャのモデルは，パラメータフィルタをポリシ非依存の第 1 ステージとポリシ依存の第 2 ステージの 2 段階に分割する．KUPF, KUPF-VR はこの 2 段階処理のモデルをそのまま実装しているため，2 段階処理は処理速度低下の原因の 1 つとなっている．本稿では，第 1 ステージにおけるポリシ依存処理の先読み実行による，パラメータフィルタの高速化を提案する．本提案は，KUPF-VR の検索木の各ノードがその下位ノードのポリシ条件を保持することで，検索木の探索中にポリシ依存処理を可能とした．これにより，ポリシに適合しない部分木を検索対象から除くことで，検索処理の効率化を図った．

# An Improvement Method in the Speed of Parameter Filter by Pre-scanning Policy

KAKIUCHI Masatoshi†　MORISHIMA Naoto†　SUNAHARA Hideki†

We have proposed and implemented KUPF architecture to expose and resolve problem of packets classification which reflects complicated policy on routers. The model of this architecture divides parameter filter into two stages: the first stage which depends on no policies and the second stage which depends on policies. We implemented KUPF and KUPF-VR based on 2-phase selection strictly, and this selection causes processing speed fall. In this paper, we propose an improvement method in the speed of parameter filter by pre-scanning policy. Our proposal makes each node of search tree on KUPF-VR store conditions of policy for lower nodes, in order to take policy during searching tree. We excluded subtrees which don't agree with policy from targets of search, and the search becomes efficient.

## 1　まえがき

近年の複雑化するネットワークにおいて，高度化・多様化するサービスを実現するためには，ルータ，アプリケーションサーバ等のサービスを提供するネットワーク機器は，複数のパラメータから構成される複雑なフィルタルールに基づいてサービスを処理することが要求される．また，複数のサービスが並行して運用されるため，1 つのフィルタルールが同時に 1 つの入力データに適合する場合がある．この場合，ネットワーク機器はサービス毎の競合を適切に解決し，入力データに適用する処理を決定する必要がある．しかし，サービスのポリシ毎に競合関係は異なるため，フィルタルールのみに基づいて一意に適用する処理を選択できない．

これらの問題に対して，くまプロジェクト[*1]は，KUMA's Universal Parameter Filter (KUPF) [1, 2, 3] アーキテクチャを提案し，実装してきた．KUPF は，複数パラメータから構成される複雑なフィルタルールが処理可能なパラメータフィルタを実現した．さらに，KUPF-VR [4] は，多次元空間検索手法を用いることで KUPF の検索速度高速化を実現した．

本稿では，第 1 ステージ処理中に第 2 ステージを先読み的に実行することで，第 1 ステージと第 2 ステージの連携により検索処理を高速化した．提案手

† 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

[*1] くまプロジェクト – http://www.kuma-project.net/

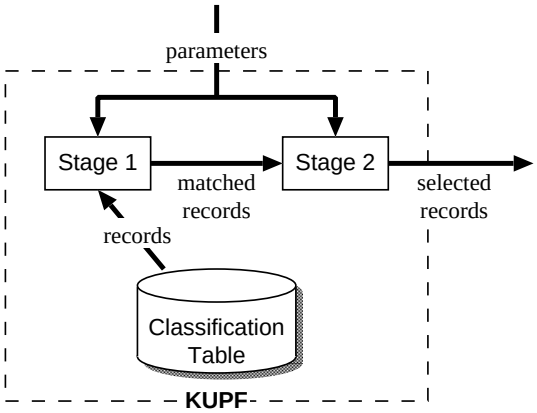


図 1　KUPF 概要

法では，最善一致検索に着目し，KUPF-VR の探索木の途中ノードに第 2 ステージ処理で必要とする情報を保持し，この情報を用いて第 2 ステージの処理を第 1 ステージ処理中に実行する．第 2 ステージにより選択される分類レコードを含まない部分木の探索を省略することで，第 1 ステージの処理を最適化する．本手法の有効性を示すために，パラメータフィルタを実装し，評価実験を行った．

## 2　パラメータフィルタ

インターネットサービスプロバイダ (ISP) はデータ転送の優先処理，帯域保証など品質を向上させるサービスを提供しようとしている．ISP は顧客とサービス品質保証契約 (SLAs; Service Level Agreements) を結び，SLA に沿ったパケット分類規則と処理内容を基幹ルータに設定する．一般的に，入力データを処理する場合，入力データに対応する処理を選択するための規則およびその処理自体の記述が必要となる．KUPF では，前者の規則をフィルタ規則，後者の記述をアクション，さらにその対を分類レコードと呼ぶ．KUPF モデルは，分類レコードを選択する機構であるパラメータフィルタを，サービス非依存処理とサービス依存処理の 2 段階に分割する．パラメータフィルタを 2 段階に分割したことで，以下が実現できた．第 1 ステージは，個々の分類レコードに対してフィルタ規則の条件が入力データを満たすか否かの判定をするのみで，サービスに依存せずに処理を行うため，汎用的に設計・実装できる．第 2 ステージは，第 1 ステージの結果からサービス毎のポリシに従って，入力データに適用するアクションを持つ分類レコードを選択する．サービス依存処理は第 2 ステージで完結しているため，第 2 ステージ置き換えのみによりパラメータフィルタを異なるサービスに応用できる．

表 1　最長一致検索分類レコード

| 分類レコード | IP アドレス | アクション |
|---|---|---|
| $R_1$ | 10.1.0.0/16 | $A_1$ |
| $R_2$ | 10.1.2.0/24 | $A_2$ |
| $R_3$ | 10.1.3.0/24 | $A_3$ |
| $R_4$ | 10.2.0.0/15 | $A_4$ |

表 2　最長一致検索矩形表現

| 分類レコード | IP アドレス | 矩形（区間） |
|---|---|---|
| $R_1$ | 10.1.0.0/16 | [10.1.0.0, 10.1.255.255] |
| $R_2$ | 10.1.2.0/24 | [10.1.2.0, 10.1.2.255] |
| $R_3$ | 10.1.3.0/24 | [10.1.3.0, 10.1.3.255] |
| $R_4$ | 10.2.0.0/15 | [10.2.0.0, 10.2.127.255] |

図 1 に KUPF の概要を示す．分類レコードは，分類表で管理される．第 1 ステージは，入力データのパラメータを入力すると，パラメータに一致するフィルタを持つ分類レコードを出力する．第 2 ステージは，入力データのパラメータと分類レコードを入力すると，サービス毎のポリシに従って分類レコードを選択し，出力する．

## 3　ポリシの先読み実行

KUPF においては，ポリシの抽出をサービス非依存に行う第 1 ステージが複数のパラメータを扱うため，Radix Tree[5]，ハッシュテーブル等の単一のパラメータに対するフィルタで培われてきた高速化手法が使えない．筆者らは，多次元空間上での効率的な検索手法として知られる R*-tree[6] を応用することで，第 1 ステージにおける検索処理を高速化する KUPF-VR[4] を開発した．KUPF-VR は，フィルタ規則を多次元空間上の矩形に変換する．この矩形を用いて探索木を構築する．

さらに，第 1 ステージ処理中に第 2 ステージの呼

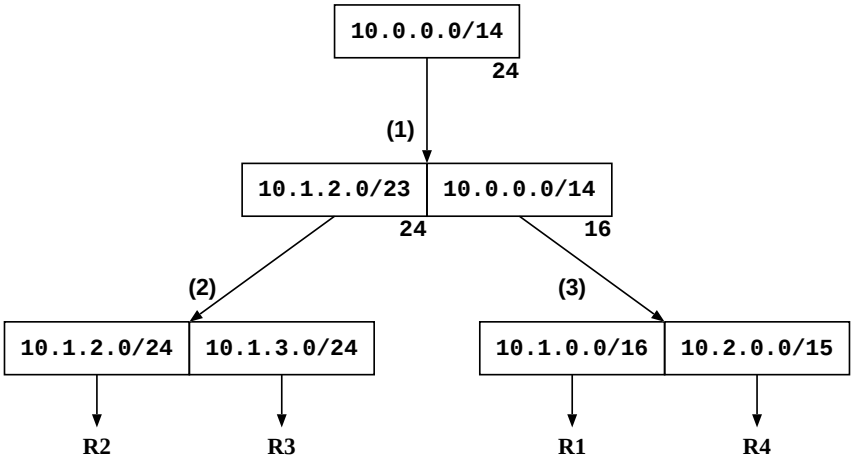


図 2　最長一致検索探索木

び出しを先読み的に実行することで，ステージ間連携による高速化の余地がある．例えば，最善一致検索において，ある分類レコードが入力データの条件を満たす場合，この分類レコードより一致度が低い分類レコード検索を省略することよる最適化が可能である．

KUPF-VR において，プレフィックス長付き IP アドレスでパケットを分類する場合を考える．この場合，フィルタ規則は 1 次元上の矩形，すなわち線分で表現される．表 1 に示す分類レコードがある場合，表 2 に示す区間の線分になる．これらの分類レコードの探索木の例を，図 2 に示す．10.1.2.0/24 と 10.1.3.0/24 および 10.1.0.0/16 と 10.2.0.0/15 は，それぞれより大きな範囲の 10.1.2.0/23 および 10.0.0.0/14 の子ノードとなり，さらにすべてを含む 10.0.0.0/14 の子ノードとなっている．この探索木を用いて, 最長一致のポリシにしたがって検索する．入力データの IP アドレスが 10.1.2.3 の場合，(1) では 10.1.2.0/23, 10.0.0.0/14 がともに 10.1.2.3 に適合する．(2) では 10.1.2.0/24，(3) では 10.1.0.0/16 が適合し，第 1 ステージでは $R_2, R_1$ が抽出される．第 2 ステージでは，最長一致則によりプレフィックス長の長い $R_2$ が選択される．

第 2 ステージの最長一致則で用いるプレフィックス長に着目する．中間ノードのエントリ 10.1.2.0/23 の下位ノードのプレフィックス長はともに 24，エントリ 10.0.0.0/13 の下位ノードのプレフィックス長は最長が 15 である．この下位ノードの最長プレフィックス長を各エントリに保持する．(2) の探索では 10.1.2.0/24 が得られる．次に (3) の探索を行う前に，下位ノードの最長プレフィックス長 15 と (2) で得られたプレフィックス長 24 を比較すると，下位ノードのプレフィックス長が短い．(3) の探索により得られる結果は，第 2 ステージの最長一致則では選択されないと判断できる．したがって，(3) の探索は必要ない．

最善一致検索において，第 2 ステージの判断基準となる値を分類レコードから抜き出し，下位ノード群から部分木の最善値を選択できれば，部分木の最善値を探索木の各エントリに保持することで，第 1 ステージ処理中に第 2 ステージを先読み的に実行できる．これにより，各ノードにおける探索処理は増えるが，探索ノードを省略することが可能となる．

## 4　実装

最善一致検索に特化したステージ間連携をおこなう，最善一致 KUPF-VR を NetBSD 1.6 上で C 言語を用いて実装した．

最善一致のためのノードエントリの構造を図 3 に示す．木構造構築のために，上位エントリ，下位エントリ，水平エントリへのノードエントリへの参照を保持する．最上位のエントリ（根ノードエントリ）は下位エントリへの接続のみ保持し，最下位のエントリ（葉ノードエントリ）は下位エントリの代わりに分類レコードへの参照を保持する．エントリの矩形は各次元毎の区間を表すパラメータ群，仮想包囲

表 3　プログラムインタフェイス

| 分類操作 | |
|---|---|
| 第 1 ステージ分類操作 | kupf_comp_table_collect(**分類表，キーパラメータ群**) |
| 第 2 ステージ分類操作 | kupf_select_best(**分類表，分類レコード群，優先順位**) |
| 最善一致分類操作 | kupf_comp_table_collect_best(**分類表，キーパラメータ群，優先順位**) |

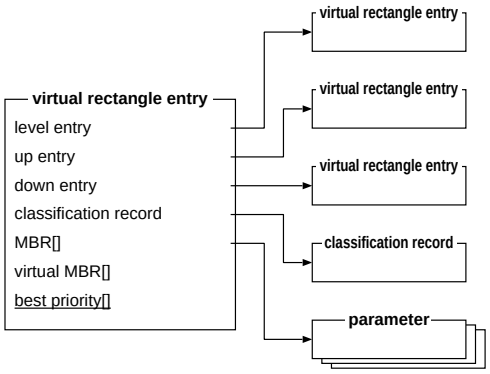


図 3　最善一致 KUPF-VR 探索木ノードエントリ

矩形は各次元毎の正規化された区間を表す整数値群で保持する．KUPF-VR 探索木ノードエントリに対し，下位ノードのパラメータ毎の最善優先度を保持する配列（下線部）を追加した．この値は，探索木にデータが挿入・削除されるたびに算出する．

分類操作は，第 1 ステージ分類操作と第 2 ステージ分類操作に分かれる．第 1 ステージ分類操作は，分類表とキーパラメータ群を与えると，返値として，分類表から，キーパラメータ群を条件として満たすのレコードパラメータ群を持つ全分類レコードが返る．第 2 ステージ分類操作は，サービス毎に依存するため個別実装が必要である．KUPF は，サービスに依存しない実装として最善一致 kupf_select_best を提供する．返値として，最善一致の分類レコードを返す．第 1 ステージで第 2 ステージの処理を実行するためには，第 1 ステージ呼び出し時に第 2 ステージ呼び出しのパラメータが必要がある．最善一致検索ではパラメータ間の優先順位が必要であるため，キーパラメータ群とともに優先順位を与えるインタフェイスを追加した（表 3）．返値として，最善一致の分類レコードを返す．

探索木からの検索は以下の手順で実行する．

1. ノードリスト **N** を根ノード，分類レコード $R$ を空とする
2. ノードリスト **N** から 1 ノード取り出し，ノード $N$ とする．ノードがなければ，分類レコード $R$ を結果として終了する．
3. 分類レコード $R$ が空でなければ，$R$ とノード $N$ の優先度を比較し，$R$ の優先度が高ければ 2 に戻る
4. キーパラメータからノード $N$ を基準とした VBR を算出する
5. ノード $N$ が葉ノードでなければ，キーパラメータとノード $N$ の各エントリを VBR 同士，キーパラメータとレコードパラメータで比較し，適合したエントリの下位ノードをノードリスト **N** に加える
6. ノード $N$ が葉ノードであれば，キーパラメータとノード $N$ の各エントリを VBR 同士，キーパラメータとレコードパラメータで比較し，適合したエントリの分類レコードと分類レコード $R$ の内，もっとも優先度が高い分類レコードを分類レコード $R$ とする．
7. 2 に返る

表 4　KUPF-VR 実験環境

| | |
|---|---|
| CPU | Pentium 4 (2 GHz) |
| メモリ | 512 MByts |
| OS | NetBSD 1.6.2_RC1 |

## 5　評価

本提案による速度性能向上の効果を検証するために，最善一致検索に特化した KUPF-VR を実装し，KUPF-VR との比較実験を行った．

表 5　実験に用いたフィルタ

| 入力インタフェイス | プロトコル | 終点アドレス | 終点ポート |
|---|---|---|---|
| 1 | UDP | fec0::/128 – fec0::/29 | 80 |

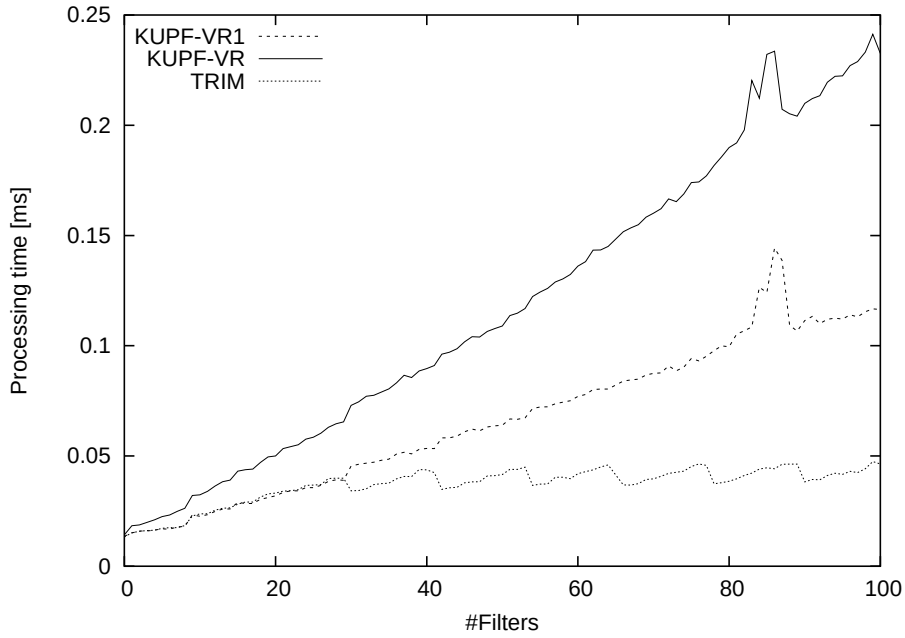


図 4　フィルタ数に対する処理時間変動

### 5.1　測定環境

性能測定環境を表 4 に示す．ユーザ空間で測定プログラムを実行した．KUPF-VR における第 1 ステージ，第 1 ステージと第 2 ステージ，最善一致に特化した KUPF-VR における第 1 ステージと第 2 ステージの処理時間を，`getrusage()` システムコールを用いて測定した．測定結果として，1,000 回実行の平均値 12 サンプル中の中間値 10 サンプルの平均値を用いた．1 ノードあたりの子ノード数の上限値は 3，下限値は 8 とした．仮想区間の上限値，下限値の精度は 16 ビット長整数とした．

### 5.2　測定結果

KUPF-VR との比較を行うため，表 5 の終点アドレスのプレフィックス長が異なる 0 から 100 個のフィルタを用いて，処理時間を測定した．キーパラメータ群は, 全てのフィルタに適合するパラメータ群を与えた．測定結果を図 4 に示す．グラフの KUPF-VR1，KUPF-VR はそれぞれ，KUPF-VR の第 1 ステージ，第 1 ステージと第 2 ステージの処理時間を示す．TRIM は最善一致に特化した KUP-VR の処理時間を示す．KUPF-VR はフィルタ数にほぼ比例した結果が得られた．フィルタ数の増加につれ探索ノード数が増え，適合レコードも増加したためと考えられる．これに対して, 最善一致に特化した KUP-VR は上下の変動があるものの，フィルタ数に関係なくほぼ一定の結果が得られた．全般に KUPF-VR に比べ優れた性能を示している．探索ノードの省略により，処理時間の増加が抑制されたためと考えられる．

### 5.3　考察

本提案では，第 1 ステージと第 2 ステージ連携による高速化を試みた．第 2 ステージはサービス依存処理となるため，汎用的なステージ連携は困難である．この問題を回避するため，最善一致を行う第 2 ステージに対象を限定した．各エントリはフィルタ規則のパラメータ毎に算出した最善優先度を保持するため，比較順序の変化に対応できる．

本実装は，検索処理中に最善ノードと中間ノード

を比較し，最善ノードに劣る部分木の探索を省略することで高速化を図っている．そのため，探索時に優先度の高いノードが含まれる部分木が先に探索されるように探索木を構築すれば，探索が省略されるノードが増え，さらに高速になる可能性がある．

部分木の最善優先度の値を算出する代わりに，部分木毎に第 2 ステージの処理を演算子として取り出すことが考えられる．これが可能であれば，最善一致検索以外にもステージ連携が適用可能となる．

## 6 むすび

本稿では，第 1 ステージ処理中に第 2 ステージを先読み的に実行することで，第 1 ステージと第 2 ステージの連携による検索処理高速化を試みた．本提案は，最善一致検索において，KUPF-VR の検索木の各ノードがその下位ノードのパラメータ毎の最善条件を保持することで，検索木の探索中に最善条件に適合しない部分木を検索対象から除外し，検索処理の効率化を図った．この結果，2 段階処理のモデルをそのまま実装した KUPF-VR に対して，ステージ間連携によりパラメータフィルタの高速化を実現した．これにより，ステージ間連携による検索処理の高速化の可能性を示した．

## 参考文献


[1] 垣内正年, 森島直人, 宇田仁, 中村豊, 砂原秀樹: 汎用的なパラメータフィルタの設計と実装, 電子情報通信学会技術研究報告, Vol. 102, No. 143, pp. 7–14 (2002).

[2] Kakiuchi, M., Morishima, N., Nakamura, Y., Fujikawa, K. and Sunahara, H.: KUPF: 2-Phase Selection Model of Classification Records, *Proc. 2003 Symposium on Applications and the Internet Workshops*, Orlando, United States, pp. 262–265 (2003).

[3] 垣内正年, 森島直人, 砂原秀樹: KUPF フレームワークの実装と応用, 情報処理学会研究報告, Vol. 2003, No. 118, pp. 61–66 (2003).

[4] 垣内正年, 森島直人, 砂原秀樹: 仮想包囲矩形に基づくパラメータフィルタの設計と実装, 電子情報通信学会和文論文誌. （条件付き採録）.

[5] Sklower, K.: A Tree-Based Packet Routing Table for Berkeley UNIX, *Proc. the Winter '91 USENIX Conference*, Dallas, United States, pp. 93–99 (1991).

[6] Beckmann, N., Kriegel, H.-P., Schneider, R. and Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles, *Proc. the 1990 ACM SIGMOD international conference on Management of data*, Atlantic City, United States, pp. 322–331 (1990).