Vol.2011-ARC-194 No.7
2011/3/11

オンルータデータベースにおけるインデックス生成機構の実装

西田雄介† 牧野友昭† 川島英之‡ 西宏章†

データベースマネジメントシステムをハードウェア化し，ネットワークストリームを直接処理することでネットワークデバイスがエンドホストに対してサービスとして提供できる可能性がある．本報告ではネットワークストリームをデータベース化することを想定したインデックス構造の検討を行い，マルチアクセス環境に対応したインデックス生成機構を FPGA 上に実装した. Xilinx FPGA Vertex5 を利用して論理実装した結果，パケットサイズが 50byte のときに 12.7[Gbps], 1036byte のときに 20.3[Gbps]のスループットを達成し, 数 Gbps 程度のネットワークで，ワイヤレートでの動作が可能であることを確認した．

Implementation of Index Generation Mechanism for Database on Router

YUSUKE NISHIDA† TOMOAKI MAKINO†
NOBUYUKI KAWASHIMA‡ HIROAKI NISHI†

By implementing DBMS as a hardware circuit, there is a possibility to provide a service to end-hosts by using network devices when the devices process network streams directly. In this paper, we examined the structure of index of on-router database management system and implemented it by using FPGA. It proves that the throughput of the proposed architecture meets a wire-rate processing of network by fulfilling the throughput of 12.7Gbps in 50 byte packets and 20.3 Gbps in 1036 byte pakcets.

1. はじめに

近年，Google，Amazon，Flickr，YouTube など，API として無償で提供した web サービスを利用することで, 容易に高機能な web ページを作成することが可能となった．この背景には，インターネット上の複数発信源の情報を組み合わせたマッシュアップなどの手法の確立や，多角的価値のあるコンテンツの増加がある．今後インターネットの発展に伴い，より豊かなサービスに対する需要が高まることが考えられる．このようなサービスの発展に伴い，交換されるコンテンツの内部を扱うことでユーザの嗜好にあった, もしくはより高度かつ包括的なサービスを提供することが望まれている．

ネットワークで交換されるパケットは，サービスから見た場合，情報の断片でしかない．比較的大きな情報を交換する場合，複数のパケットに分断されて通信が行われるが，これらのパケット全体をストリームと呼ぶ．例えば，TCP/IP によって管理されたまとまりのあるデータにより作成されるパケット群を TCP ストリームと呼ぶが, 同様にあるまとまりで意味を成すパケット群をネットワークにおけるストリーム，すなわちネットワークストリームと呼ぶ．ネットワークにおけるサービスの提供では，ネットワークストリームという単位で情報を管理・解析することが必要である．パケット単体では情報の断片として扱わなければならず，単純にパケットの並びとして情報を扱うと，パケット間にまたがる情報の取得が難しくなることや，インターネットにおいてはパケット到着の FIFO 性が保障されない．この問題を解決するためネットワークストリームの形に変換して扱う研究や提案が過去に行われている[1]．このようなネットワークストリームという形で情報を利用するメリットは大きいが，未だエンドホスト以外，例えばゲートウエイやルータなどといった部位でのネットワークストリームの利用例は限定的である．ネットワークストリームはインターネットユーザの行動履歴や，ストリームの出現時刻など，今日までコンテンツとして利用されていない情報を多く含むため，コンテンツとして活用できれば従来に無いサービスを提供できる可能性がある．

そこで，データベースマネジメントシステムをハードウェア化し，ネットワークストリームを直接処理可能とすることでサービスへと転化することを考える．ネットワークストリームをコンテンツとして利用するには，ストリームから必要なデータのみを抽出してデータベースへ格納し，セレクションを考慮してインデックスを管理する必要があるが，そのためにはインデックス生成機構のワイヤレートでの動作が必要不可欠である．本稿ではネットワークストリームのインデックスをワイヤレートで生成し，特に複数クエリが混在する環境において，効率よく管理することを想定したインデックス生成機構をハードウェア実装することを目的とする．また，決定した構造に基づくインデックス生成機構を設計し，FPGA 上に実装したうえで，回路規模や動作速度といった評価を行う．

2. 関連研究

膨大なストリームデータに対し，外部記憶装置に蓄積することなくデータ検索を可能にする取り組みとしてストリームデータベースの研究がある．ストリーム処理エン

† 慶應義塾大学
Keio University
‡ 筑波大学
Tsukuba University

ジン(SPE)に関する研究は，およそ3世代に大別される．

第一世代SPEでは，ストリームを扱うためのデータモデルを提案し，同モデル上における性能向上とメモリ使用量の制御が研究された[2] [3]．第二世代SPEでは，第一世代SPEで提案されたデータモデルに対して，分散処理を導入することで，高性能化および耐故障化に関する研究が行われた[4]．そして第三世代SPEでは，RFIDやネットワークトラフィックの監視といったアプリケーションを想定した，専門用途システムの構築が研究されている[5][6][7]．また，ストリームデータからイベント・パターンを発見しアプリケーションを駆動するイベントストリームプロセッシングの研究もなされている[8][9][10]．

ストリームデータ処理の具体例として，RFID やセンサなど，高いレートで生成される継続的データのリアルタイム処理が挙げられる．そのため，ストリームデータベースには高速処理が求められる．また，本報告の想定するネットワークストリームを有効なサービスとして利用する場合には，ワイヤレートで流れるネットワークストリームを継続的に取得・管理する必要がある．これらのことから，ストリームデータベースとネットワークストリームの有効利活用の実現には，データ挿入処理の高速化という共通目的が存在する．

これらのハードウェア化に関する研究もおこなわれているが，すべてセレクションにおけるデータベース内部基本操作のハードウェア化がメインであり，インサーションに関する研究は，我々以外では未だ行われていない．

3. インデックス生成機構の設計

ネットワークストリームをデータベースへ格納し，サービスとして提供するとき，ストリーム全体を抽出するのではなく，ユーザが指定した抽出項目に従って選択的に抽出する方が，より効率的であると考えられる．そのため，各ユーザが指定した抽出項目に従ってデータが管理され，これらのデータの読み出し処理においても，ユーザ単位での一括取得が可能であるほうが望ましいと考えられる．すなわち，ユーザごとに管理されたインデックス構造を有することが必要である．さらに，この構造においては,ユーザ数の増大に対してある程度スケーラブルな構成であることが必要である．本節では実装するインデックス構造について検討を行い，インデックス生成機構の設計を行う．

3.1 インデックス構造の検討

複数のクエリに重複してマッチすることを考慮したインデックス構造として，以下の 3 つが考えられる．

1, 重複データを重複分コピーする．コピーしたそれぞれのデータに対してインデックスを生成し，それぞれのインデックスに対して単一の usr_next ポインタ(データ間の連結を管理するポインタ)を付加する．
2, 重複データに対して，個別に ID を与えることで，全体として重複なく管理する．usr_next ポインタは新たに付加した ID ごとに管理する．
3, 一つのインデックスに対して複数の usr_next ポインタを付加する．

本節では上記の 3 つの手法について検討を行い，実装するインデックス構造の検討を行う．以下では 3 つの手法を検討するにあたり，例としてユーザ A は Google に関するパケットを取得するクエリを発行し，ユーザ B はパケットサイズが 1Kbyte 以上のパケットを取得するクエリを発行した場合について考える．

まず，手法 1 について述べる．手法 1 のインデックス構造は図 1 のようになる．手法 1 は重複したデータをコピーし，それぞれのデータに対して各ユーザの usr_next ポインタを付加する．この手法は単純な手法であるため管理が容易で，読み出し処理が簡略化できるが，重複数に比例してデータをコピーする回数が増加し，メモリ使用効率が低下する．また，コピーを実行している間に次の抽出データが到着した場合は，メモリへのデータの書き込み処理を,待ち合わせのために遅延させる必要があるため，データの取りこぼしが生じる．

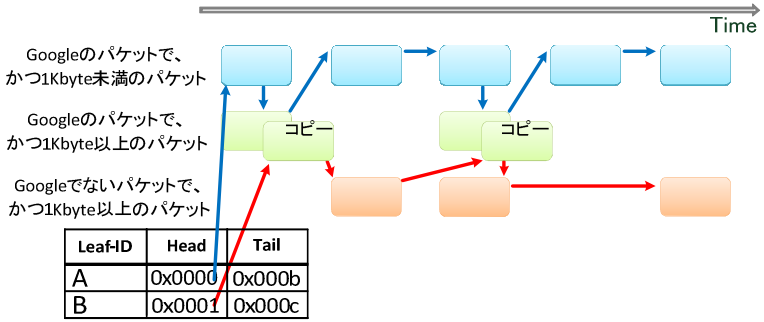
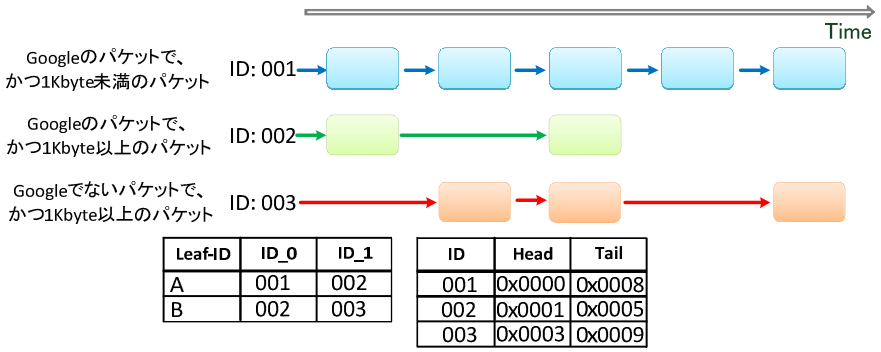


図 1：手法 1 におけるインデックス構造

次に手法 2 について述べる．手法 2 のインデックス構造は図 2 のようになる．発行したクエリによって生じ得るすべての組み合わせについて，ユーザ ID とは別の仮想 ID を付加しデータを管理する．この手法は，データの読み出し処理を考慮すれば，ユーザ ID と仮想 ID との関連を管理するテーブルが必要であり，仮想 ID ごとに保存領域を管理するため，保存領域リストの先頭アドレスと末尾アドレスを記録するテーブルも管理する必要がある．この手法は手法 1 のように抽出データ取りこぼしが生じる

問題は発生しないが，クエリ数が増えた場合には付加すべき ID 数が急増する可能性がある．具体的には，クエリを発行するユーザ数を n とするとワーストケースでO(n!)で付加すべき ID 数が増大する．また，新たなクエリを発行した場合には，すでにあるクエリとの組み合わせを考慮して付加する ID の情報を更新しなければならず，テーブル管理コストが高いといえる．

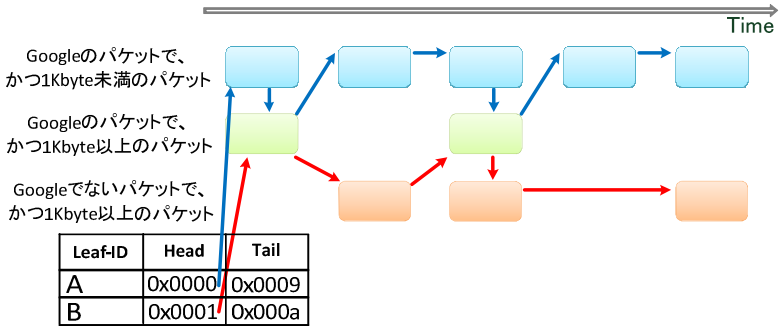
最後に手法 3 について述べる．手法 3 のインデックス構造を図 3 に示すように，一つのインデックスに対して複数の usr_next ポインタを付加する．この手法は，データのコピーを伴わないためデータの取りこぼしの問題は発生しない．また，クエリを発行するユーザ数とテーブルで管理するユーザ数が一致するため，手法 2 よりもテーブル管理コストは低い．一方で，一つのデータブロックに対して複数の usr_next ポインタを付加させるため，インデックス管理が複雑となり，管理コストが増大と処理速度の低下が想定される．



Leaf-ID	ID_0	ID_1
A	001	002
B	002	003

ID	Head	Tail
001	0x0000	0x0008
002	0x0001	0x0005
003	0x0003	0x0009

図 2：手法 2 におけるインデックス構造



Leaf-ID	Head	Tail
A	0x0000	0x0009
B	0x0001	0x000a

図 3：手法 3 におけるインデックス管理手法

ストリームからデータを抽出する際には，できるだけデータを取りこぼさないことが望ましい．したがって，ネットワークの状況によってデータの取りこぼしが頻繁に生じ得る手法 1 は本研究が想定する状況において不適切であるといえる．また，手法 2 と手法 3 では，クエリ数が少ない場合は手法 2 によるインデックス管理が有効であり，クエリ間での重複が多い場合は手法 3 によるインデックス管理が有効である，すなわちクエリ重複数とのトレードオフになると考えられる．

3.2 設計

本稿ではユーザの発行するクエリ数が多い時を想定し，手法 3 の構造のインデックス生成機構を設計し，実装した．インデックス生成機構のインデックス生成手順は以下のようになる．

1, ユーザ ID 情報と Timestamp をこの順に受け取り，インデックス生成を開始する．
2, Index Memory から Free Address を取得し，Timestamp，時間リストポインタ等の情報を，Index Memory へ書き込む．
3, ユーザ ID を用いてユーザテーブルへアクセスし，ユーザの末尾アドレスを取得する．
4, 3 で参照したユーザの末尾アドレスにおけるインデックスに，今回の書き込み先アドレスを usr_next ポインタとして記録する．その後，ユーザテーブルの末尾アドレスを更新する．
5, 3，4 を繰り返し，到着データに付加されたユーザ ID が無くなり次第，インデックス生成処理を終了する．

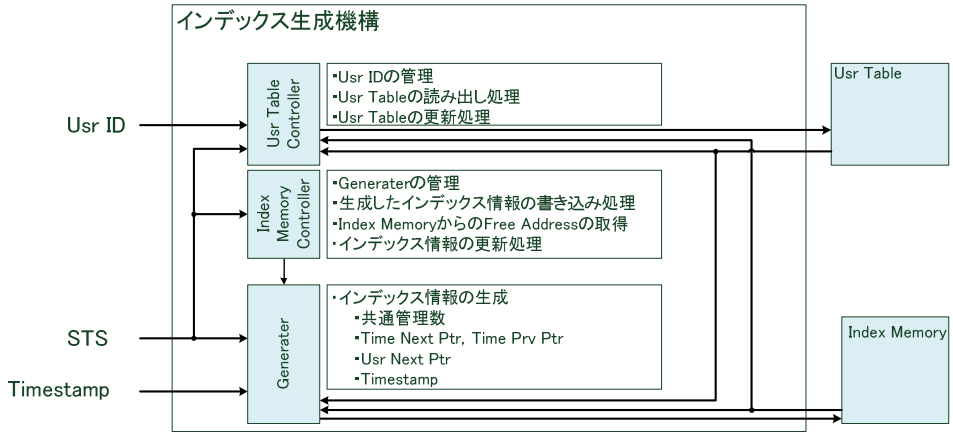


図 4：手法 3 におけるインデックス生成機構のアーキテクチャ

4. 実装・評価

3.2 項で設計したインデックス生成機構を Xilinx ISE Design Suite 12.3 で論理合成し，Xilinx の FPGA Vertex5 XC5VLX330T 上に実装した．また，FREEPDK45n テクノロジを用いて，Synopsys Design Compiler 2005.09 で ASIC として論理合成した．また，それぞれの論理合成における回路規模，動作遅延，最大動作周波数を求めた．

表 1，表 2 にそれぞれのソフトにおける論理合成結果を示す．また，重複数が 2 のときにおける，パケットサイズが 50byte のときと，1306byte のときのインデックス生成機構の対応可能なスループットを表 3 に示す．なお，50byte はパケットサイズとして想定される最少のサイズであり，1306byte は WIDE MAWI の 2009/7/12/14~14:15 の HTML パケットの平均サイズである. 表 3 より, どちらのパケットサイズにおいても，インデックス生成機構は数 Gbps 程度のネットワークではワイヤレートでの動作が可能であることが確認できた．

表 1：ISE Design Suite による論理合成結果

回路規模(セル数)	動作遅延(ns)	最大動作周波数(MHz)
2,818	3.14	318

表 2：Design Compiler による論理合成結果

回路規模(μm^2)	動作遅延(ns)	最大動作周波数(MHz)
2.08×10^4	1.97	508

表 3：それぞれのパケットサイズにおける対応可能なスループット

パケットサイズ	ISE Desing Suite	Design Compiler
50byte	12.7Gbps	20.3Gbps
1036byte	20.3Gbps	32.3Gbps

5. 結論

本報告は，ネットワークストリームをサービスとして利用することを想定し，データベースマネジメントシステムをハードウェア化することにより実現することを考えた．ストリームをサービスとして利用するには，データベースのインデックス構造の検討が不可欠でるため，本報告は実装するインデックス構造の検討を行い，それに基づくインデックス生成機構の設計を行った．

設計したインデックス生成機構を FPGA により実現した場合ではパケットサイズが 50byte と 1306byte のそれぞれにおけるスループットは 12.7[Gbps], 20.3[Gbps]となり，FREEPDK45n テクノロジを利用した ASIC で実現した場合はそれぞれ 20.3[Gbps], 32.3[Gbps]となることを確認した. この結果, 設計したインデックス生成機構が数 Gbps 程度のネットワークではワイヤレートで動作することが可能であることを確認した．

謝辞 この研究は，独立行政法人情報通信研究機構の高度通信・放送研究開発委託研究「ネットワーク仮想化を活用したデータサービスアプリケーション基盤技術に関する研究開発」並びに「新世代ネットワーク技術戦略の実現に向けた萌芽的研究」および文部科学省科学技術研究費補助金基盤研究C「安心・安全な情報提供を可能とするインターネット基盤の構築に関する研究」の一環としてなされたものである．

参考文献

[1] 菅原豊，千本潤介，稲葉真理，平木敬. 10 ギガビットイーサネットを対象とした再構成型ネットワークプロセッサ．第 1 回リコンフィギャラブルシステム研究会論文集. pp.249-256. 2003

[2] S. Chandrasekaran, O. Cooper, A. Deshpande, and M. J. telegraphcq continuous dataflow processing for an uncertain world. In SIGMOD, pp. 668–668, 2003.

[3] B. Zhao, L. Huang, J. Stribling, S. C Rhea, A. D Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications, pp. 41–53, 2004.

[4] J.-H. Hwang, Y. Xing, U. Cetintemel, and S. Zdonik. A cooperative, self-configuring high-availability solution for stream processing. In International Conference on Data Engineering, 2007.

[5] B. Gedik, H. Andrade, K.-L. Wu, P. S. Yu, and M. Doo. Spade: the system s declarative stream processing engine. In SIGMOD, pp. 1123–1134, 2008.

[6] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: a stream database for network applications. In SIGMOD, pp. 647–651, 2003.

[7] L. Girod, Y. Mei, R. Newton, S. Rost, A. Thiagarajan, H. Bal akrishnan, and S. Madden. A regular expression processor embedded in service-friendly router for future internet. In Conference on Innovative Data Systems Research, 2007.

[8] L. Brenna, A. Demers, J. Gehrke, M. Hong, J. Ossher, B. Pand a, M. Riedewald, M. Thatte, and W. White. Cayuga: a high-performance event processing engine. In 2007 ACM SIGMOD International Conference on Management of Data International Conference on Management of Data, 2007.

[9] A/J.Demer, M. Hong, M.Riedewald, and W.M White. Towards expressive publish/subscribe systems. In EDBT, 2006.

[10] E.Wu, Y.Diao, and S.Rizvi. High-performance complex event processing over streams. In SIGMOD `06: Proceedings of the 2006 ACM SIGMOD, p.407418,2006.