4F-7

可変構造多重処理データベースマシンに於けるソーティングユニット

伏見信也 鈴木直信 喜連川優 田中英彦 元岡達
(東京大学 工学部)

§1. はじめに

リレーショナルデータベースに於る関係代数の処理は,データを予めソートしておくことにより高速化できることが知られている。この事実に鑑み,可変構造多重処理データベースマシンに於ては,Pモジュール内にソーティングユニットを置き,Mモジュールからシリアルに転送されてくるデータに同期した(即ち$O(N)$の)ソートを行なう構成[1]になっている。ここでは試作中のソータの内部構成について紹介する。

§2. ソート アルゴリズム

アルゴリズムはパイプライン化されたマージソートを基調としている[1]。今,$N(=K^\ell)$ケのレコードをソートするものとすると,K-wayマージを行なうプロセッサを$\ell(=\log_K N)$台用意し,これらを一次元状に結合する(Fig.1)。i番目のプロセッサは$K^{i-1}(K-1)$レコード分のメモリを持ち,i-1番目のプロセッサから送られてくるK^{i-1}レコードからなるソートされたストリングをK本マージしてK^iレコードからなる1本のストリングを生成し,i+1番目のプロセッサへ送出する。ストリングの入力,マージをオーバーラップさせ,プロセッサをパイプライン結合することにより$O(N)$の時間でソートを完了することができる。今回の実装にあたってはK=2とし,プロセッサ結合はバイトレベルとした(3.2参照)。

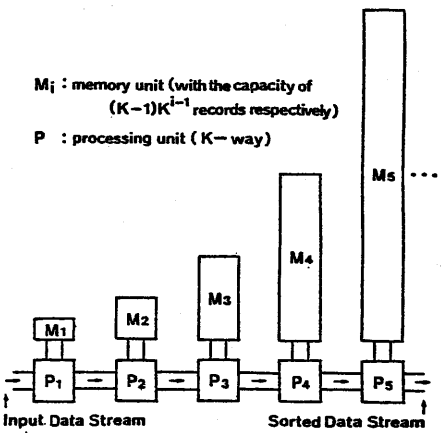


Fig.1 Global Architecture Of Stream Driven Sorter

§3. 実装方式

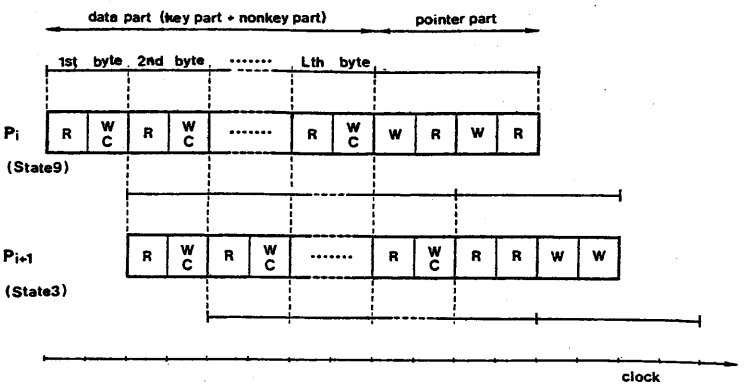
3.1 メモリ管理

各々のストリングは順にソートされてプロセッサに入力される為,メモリ中では何らかの方法でこの順序を維持する必要がある。この為のメモリ管理方式として,Double Memory Method, Block Division Method, Pointer Methodの3種の方式[1]を検討したが,今回は制御の容易さ等の点からPointer Methodを採用した。これはストリングの順序維持方式としては最も単純なもので,各レコードの最後尾にポインタ領域を付加し,全体をLinked listとするものである。ここに各レコードはキー部,非キー部を合わせたデータ部とポインタ部からなるフォーマットでメモリ中に格納される。ポインタ領域は現在2バイトである。キー部,非キー部を分離せず直接ソートする為データ部は比較的長く,ポインタ領域がメモリ全体に占める割合は小さい。

3.2 パイプラインの構成

プロセッサ間はバイトレベルで結合されており,1サイクル毎に1バイトのデータが各プロセッサから次段のプロセッサへ一斉に送出される。従って各プロセッサは1バイト/1サイクルの処理レートが要求され,これを達成する為にはメモリへのアクセス競合の解消が必須となる。そこで実装にあたっては1サイクル=2フロックとし,始めのフロックをメモリ読み出し(read)に,後のフロックをメモリ書き込み(write)に割り当てる方針を採った。全体としてはreadフロックでメモリからの必要なデータの読み出し,比較の準備を行ない,writeフロックでメモリへの書き込み及び比較を並行して行なう。一方,比較器への入力用レジスタを2つ設け(MTR,CR),その一方(MTR)には比較器の入力データの片方を常に保持させる。以上の構成により1サイクルを2回のメモリア

プロセスとする完全パイプラインを形成できる。(但し、ポインタ部を処理する2サイクルについては read ⇄ write のシーケンスは状態に応じて変化する。) Fig.2 にパイプラインの流れの一例を示す。



R: Memory Read
W: Memory Write
C: Comparison

Fig.2 Pipeline Flow Example

3.3 状態遷移

プロセッサはレコード間の大小比較の結果に従って Fig.3 に示すような内部状態遷移を繰り返す。各内部状態(10状態)の詳細については省略するが、図中に示す通り内部状態は大きく3つの Phase に分けられ、各々の次のような役割を持つ。

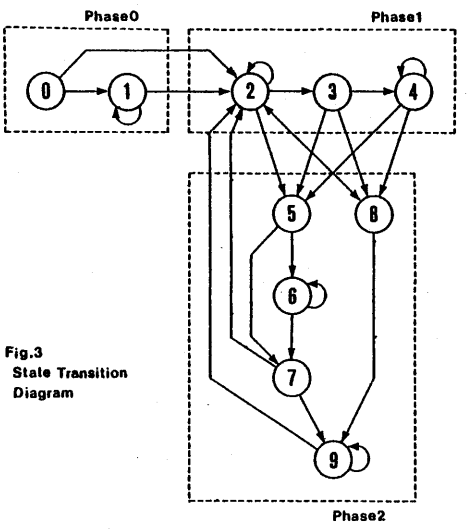


Fig.3 State Transition Diagram

Phase∅: 最初のストリング(string∅)のメモリへへのロードを行なう。ロード完了後 Phase 1へ。

Phase 1: メモリ内の string∅と入力されてくるストリング(string 1)のマージ及び出力を行なう。string 1の入力終了後 Phase 2へ。

Phase 2: メモリ内に残存する string∅, string 1のマージ、出力、及び次段のマージ用のストリング(string 2)のメモリ内へのロードを行なう。string 2の入力終了後、string 2を string∅とし Phase 1へ。

3.4 ハードウェア リソース

各プロセッサが持つ主なレジスタを以下に示す。

MAR (Memory Address Register)

LR (Link Register): linked list 生成に用いる。

STPi (ith String Top Pointer): string i の先頭レコードのアドレスを保持する。

CTRM (Master CounTeR): ストリング長管理を行なう。Phase 間遷移の条件を規定する。

CTRi (CounTeR for ith string): メモリ中に存在する string i のレコード数を保持する。

MTR (Merge Top PoinTer) } 比較器への入力
CR (Comparison Register) } データを保持する。

Fig.4 に動作中のプロセッサのスナップショットを示す。

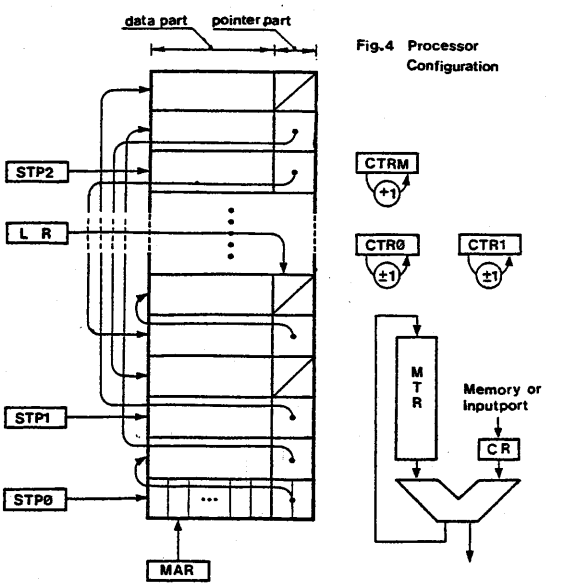


Fig.4 Processor Configuration

§4. おわりに

現在、レジスタトランスファレベルのシミュレーションを終え、ハードウェア実装を行なっている。また本リータはエンコードされた、限られたレコードを扱うのではなく、キー部、非キー部を分離しない任意長のレコードのソートを目標としており、この観点からの機能の拡張及び LSI 化を検討中である。

参考文献: [1] 喜連川他. 信学技法 EC81-15