データベース・システム 54－4
（１９８６．７．２１）

分散処理向きデータベースマシンの考察

坂本　明史、川上　英、疋田　定幸
沖電気工業（株）

データベースマシンはデータベース管理システムの性能向上に効果がある。したがって、データベースマシンを用いた性能価格比の高い分散処理システムの構築が可能である。しかし、そのためにはデータベースマシンとホスト計算機との通信オーバヘッドを削減することが重要となる。筆者らは、この問題に対してデータベースマシンをプログラマブルな環境を持つ仮想データベース計算機とする方法でアプローチしている。仮想データベース計算機はデータベース管理命令、通信Ｉ／Ｏ命令などの仮想命令セットを持ち、応用システムに適したホストとのインタフェースを提供できる。

本稿では、今回インプリメントを行った仮想データベース計算機へのアプローチとそのアーキテクチャについて述べる。さらに、通信コストの改善について評価する。

A Distributed System Oriented Database Machine(in Japanese)

Akifumi SAKAMOTO, Suguru KAWAKAMI, Sadayuki HIKITA
OKI Electric Industry Co. Ltd,
4-11-22 Shibaura, Minato-ku, Tokyo 108, Japan

Database machines work as good components to organize distributed systems. Database machines have shown their performance and cost-effectiveness in commercial markets. However, problems of communication overheads between host computers and database machines have not fully resolved yet. In order to resolve this problem, authors propose the virtual database machine which provides users with a programmable environment. This machine has a virtual instruction set such as query instruction, input/output instruction etc. to make the interface flexible to applications.

In this paper, the motivation of the virtual database machine is described. Then the architecture of it is explained. Finally, its performance is evaluated using an example.

１．はじめに

機能分散型の分散処理システムでは従来一つの計算機で行っていたことを複数の専用計算機で行うことにより性能／価格比を上げることができる。しかし、そのためには計算機間の通信オーバヘッドを削減することが重要となる。

データベースマシンについても同じことが言える。たとえば、ＸＤＭＳ[CANA74]ではCODASYLのデータ操作命令がインタフェースの水準として低すぎたために、通信ボトルネックとなったと考えられる[植村80]。

関係データベースマシンでは、関係データベースのキュアリによる抽象度の高いインタフェースによって、通信ボトルネックを解決しようとしている[HIKI81][EPST80]。

ところが、実際の応用システムで応用プログラムがデータベースマシンにアクセスする場合には、検索、更新以外にもトランザクションの制御など多くのインタラクションが発生する。そこで応用システムにおいては、そのような通信コストをおさえることが要求される。

本稿では、通信コスト低減のため、データベースマシンのあるべきアーキテクチャを検討し、これを実際のデータベースマシンＦＲＥＮＤ[HIKI85a]へ適用した実例について述べる。

２．通信コストの改善

データベースマシンがデータベース管理システム（ＤＢＭＳ）の性能改善に効果があることはベンチマークテスト[BITI83]により示されている。これは以下のような理由による[HIKI85a][STON81][STON83]。

（１）応用プログラムに影響されることなく、データベース処理に適した処理構成をとることができる。
（２）バッファ管理をデータベース処理専用に設計できる。
（３）タスクスイッチなどのオーバヘッドの少ない専用ＯＳを用いることができる。

しかし、これらはあくまでＤＢＭＳ自体の改善であり、通信オーバヘッドの問題は解決されていない。特にデータベースマシンを共有サーバのように疎結合で用いる場合には、ホスト計算機との通信コストが高く問題が大きい。

通信オーバヘッドには、通信量によるオーバヘッドと通信回数によるオーバヘッドがある。低速の通信回線では、通信量が問題となる。一方、高速の通信回線では、通信回数が問題となる。共有サーバ型のデータベースマシンは公衆回線のような低速の広域ネットワークからイーサネット[METC76]などの高速のＬＡＮまで、種々の通信回線を通してアクセスされる。したがって、通信量と通信回数の両方のオーバヘッドを削減する方法が必要となる。

大量のデータをデータベースマシンに送ったり、逆にデータベースマシンから取り出す場合には、高速の回線を用いる以外に通信コストを削減する方法はない。しかし、それ以外では幾つかの方法が考えられる。

通信量を減らす方法として、ＡＮＳＩ標準のＳＱＬ[ANSI85]に導入された手続きの機構が利用できる。これは手続き名と仮引数で手続きの仕様を記述し、本体を一つのＳＱＬ文で記述するものである。実行時には、手続きは手続き名と実引数によって呼び出され本体のＳＱＬ文が実行される。データベースマシンにＳＱＬの手続きを置き、これをホストから呼び出すとすると、プログラム実行時にＳＱＬ文全体を送る必要がなくなり通信量を減らすことができる。

通信回数を減らす方法として、ＤＥＬＴＡ[SHIB84]のコマンド木がある。プリミティブなＤＥＬＴＡコマンドを個々に送るのではなく、複数のコマンドを含むキュアリ木として送るものである。この方法は、一回の転送で複数のコマンドを送ることができる。これにより、個々のプリミティブなコマンドとその応答が、ホストとデータベースマシン間で通信されるのをふせぐことができ、通信回数を減らすことができる。

しかし、ＳＱＬの手続きは、単一のＳＱＬ文を起動できるだけであり、ＤＥＬＴＡのコマンド木も単一のキュアリを構成できるだけである。データベースマシンのインタフェースとしては、単一のキュアリの通信量、通信コストを下げることはできるが、それ以上の改善は望めない。

データベース検索言語の機能として集合関数の機能がある。この機能を拡張し、キュアリの記述能力を高めることによっても通信オーバヘッドを減少させることが可能である。集合関数を用いれば、平均値、最大値、タプル数などを、一回の検索で求めることができる。２の巾乗、標準偏差などの関数をデータベースマシンのインタフェースに追加すれば、結果だけを転送することができ通

信コストを減少させることができる。しかし、実際には利用者の多様な要求をデータベースマシンが完全に満たすことはできない。

ＩＤＭ[UBEL85] [EPST80]のstored commandは、トランザクション開始、トランザクション終了といった制御用のコマンドや、検索のプリミティブコマンドをあらかじめ格納しておくことにより、実行時の通信コストを減少させることができる。

しかし、この機能はデータベース処理のプリミティブを格納するためのものであり、手続き的な処理はホスト計算機側で行うことになる[TSUR84]。したがって、応用システムの処理全体を考えた場合、ホスト計算機とデータベースマシンの通信オーバヘッドの問題は解決されない。

以上の分析により、データベースマシンにおける通信オーバヘッドの問題は次の点の解決が必要であるといえる。

（１）インタフェースが単一キュアリの記述能力では、トランザクションの制御や複雑な処理のためには複数の通信を必要とする。したがって、ＳＱＬの手続き、集合関数、コマンドのキュアリ木としての一括送信により単一キュアリの通信コストを下げただけでは処理全体の通信回数、通信量はそれほど減少しない。

（２）データベースマシンのコマンドセットは、データベース処理だけに基づいたもの（例えば関係代数演算やトランザクション制御）である。コマンド木やstored commandによってそれらのコマンドを組み合せても、条件判断や計算などの手続き的な処理を作れない。したがって、手続き的な処理を必要とする処理においては、通信コストは減少しない。

筆者らは、これらの問題に対して、データベースマシンをプログラマブルにする方法でアプローチし、データベースマシンＦＲＥＮＤ上でインプリメントを行った。以下で、筆者らが開発したデータベース処理のための仮想計算機環境を提供するプログラマブルデータベースマシンの実現法について述べる。

３．実現アーキテクチャ　－－　仮想データベースマシン

３．１　データベースマシンＦＲＥＮＤの拡張

図１にＦＲＥＮＤのシステム構成を示す。ＦＲＥＮＤは、パーソナルコンピュータやワークステーションをホスト計算機とするバックエンドマシンとして働く。ＦＲＥＮＤは関係ＤＢＭＳを持ち、データベース管理を行う。応用プログラムはワークステーション上で実行される。

両者はＦＲＥＮＤと通信ネットワークで接続される。キュアリやトラザクション制御などのコマンドと検索結果などのデータが通信ネットワークを介してやりとりされる。この際の通信オーバヘッドが２章で述べた問題点である。

この問題点の解決のために、リモートプロシジャコール[BIRR84]を用いる方法が考えられる。利用者は、ＦＲＥＮＤの中にリモートプロシジャとしてプログラムを記述・格納する。ホスト側にある応用プログラムは、このプロシジャをリモートプロシジャコールにより起動し、パラメタを渡す。リモートプロシジャはＦＲＥＮＤ上で実行され、ホストはその処理結果を受け取る。しかし、この機構を実現するためには、ＦＲＥＮＤ上で応用プログラムが実行できるようにＦＲＥＮＤのＣＰ

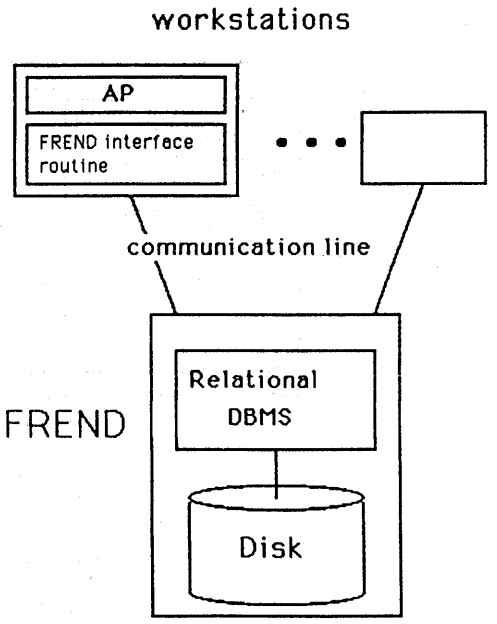


図１　ＦＲＥＮＤのシステム構成

Ｕを解放し、汎用のＯＳやプログラミング環境を提供する必要がある。これは、２章で述べたデータベースマシンＦＲＥＮＤの性能を向上させる専用化の特長に反する。汎用計算機の環境ではＦＲＥＮＤは効率よく動かない。

そこで、筆者らは、ＦＲＥＮＤのＣＰＵを利用者に解放せず、ＦＲＥＮＤをデータベース処理命令を持つ仮想計算機とする方式を用いた。以下この仮想計算機をＦＲＥＮＤに対してＶ－ＦＲＥＮＤと呼ぶ。Ｖ－ＦＲＥＮＤは、トランザクション開始命令、検索命令、更新命令などデータベース処理に対応した命令（データベース管理命令）の他に、条件分岐命令や文字列操作命令、演算命令、ロードストア命令などを持つ。これにより、データベースマシンのコマンドセットの制約から、手続き的な処理を実行する場合に生じる通信オーバヘッドの問題を解決している。しかも、ＤＢＭＳはデータベースマシンとして専用化された実行環境で動作できる。

利用者はＶ－ＦＲＥＮＤの命令を用いて、必要なプログラムを作成する。このプログラムはＦＲＥＮＤの内部に格納される。このプログラムはワークステーション上のＡＰによって起動され、Ｖ－ＦＲＥＮＤ上で実行され、結果がワークステーションに送り返される。以下では、Ｖ－ＦＲＥＮＤのアーキテクチャや実行手順の詳細について述べる。

３．２ 仮想データベース計算機のアーキテクチャ

Ｖ－ＦＲＥＮＤは、以下のユニットにより構成される（図２）。

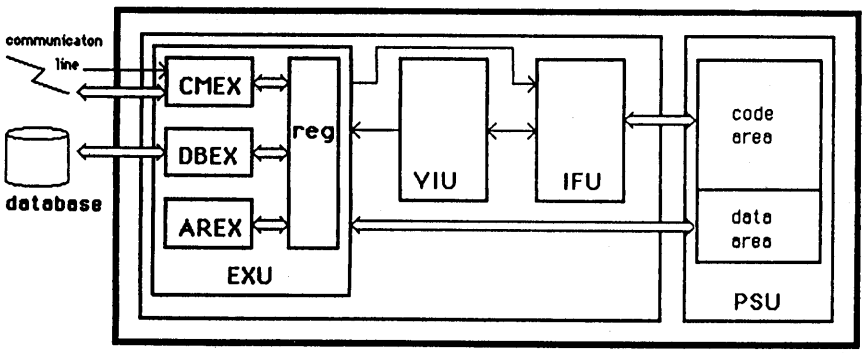
（１）命令フェッチユニット（ＩＦＵ）
次に実行するＶ－ＦＲＥＮＤ命令を取り出す。

（２）仮想命令ユニット（ＶＩＵ）
・Ｖ－ＦＲＥＮＤの命令セットに対してデコードを行い実行ユニットを起動する。

（３）実行ユニット（ＥＸＵ）
・実行ユニットにはデータベース管理実行部（ＤＢＥＸ）、通信実行部（ＣＭＥＸ）、演算実行部（ＡＲＥＸ）がある。それぞれＶ－ＦＲＥＮＤのデータベース管理命令、通信Ｉ／Ｏ命令、演算命令の実行を行う。

(3-1) データベース管理命令実行部（ＤＢＥＸ）
・内部のＤＢＭＳと対応する。ＤＢＭＳの機能を使う命令、例えばトラザクション開始・終了、検索、更新などの命令を実行する。

(3-2) 通信実行ユニット（ＣＭＥＸ）
・ワークステーション等ホスト計算機との通信命令を実行する。検索結果の転送やホストからのパ



Virtual Database Machine

⇔ flow of data
↔ flow of control

EXU: EXecution Unit
VIU: Virtual Instructon Unit
IFU: virtual Instruction Fetch Unit

CMEX: CoMmunication EXecution part
DBEX: DBMS EXection part
AREX: ARithmetic EXecution part
reg: registers
PSU : Program Strage Unit

図２　仮想データベース計算機のアーキテクチャ

ラメタの受取りは、このユニットによって行われる。

(3-3) 演算ユニット（ＡＲＥＸ）
・数の比較、演算、文字列操作等を行う。ループや条件分岐などの実行制御にも用いる。

（４）プログラム格納ユニット
・プログラムコードとデータを格納する。

実行ユニット内にはいくつかのレジスタがある。以下で主なものを説明する。

（１）入出力レジスタ
入出力レジスタはホスト計算機との通信に用いる。ホスト計算機からのデータは入力命令（ＩＮＰＵＴ命令）によって入力レジスタ（ＩＮＰＵＴ＿ＲＥＧ）にセットされる。ホストへの転送データは出力レジスタ（ＯＵＴＰＵＴ＿ＲＥＧ）にセットされた後、出力命令（ＯＵＴＰＵＴ命令）によって送出される。

（２）データ操作レジスタ・タプルレジスタ
データベース管理命令のオペランドはデータ操作レジスタ（ＤＭ＿ＲＥＧ）にセットされる。データベース管理命令の実行によりＤＢＥＸが起動される。検索されたタプルや更新するタプルはタプルレジスタ（ＴＵ＿ＲＥＧ）にセットされる。

（３）プログラムカウンタ
プログラムカウンタは次に実行するＶ－ＦＲＥＮＤの命令を指している。

（４）その他のレジスタ
利用者名レジスタ（ＵＮ＿ＲＥＧ）がある。ＵＮ＿ＲＥＧは、Ｖ－ＦＲＥＮＤを利用している利用者名を保持している。また、状態レジスタ（ＳＴ＿ＲＥＧ）があり、Ｖ－ＦＲＥＮＤ命令実行後の終了コードを保持している。

Ｖ－ＦＲＥＮＤへは複数の利用者が同時にアクセスできる。したがって、レジスタなどの実行環境は利用者毎に割り当てられる。一方、プログラムコードは、全利用者で共有される。現在これらの実行環境は、すべてソフトウェアにより実現されている。

３．３　Ｖ－ＦＲＥＮＤ命令の実行
ここでは簡単な例を挙げてＶ－ＦＲＥＮＤ上でのプログラムの動作を説明する。

Ｖ－ＦＲＥＮＤのＰＳＵのデータ領域に、以下に示すデータが格納されている。これは検索条件の一部である。

```
      ＮＡＭＥ
from  Ｒ
where ＣＩＴＹ　＝　’ＮＹ’
```

ホスト計算機は、実行時に決るＡＧＥ　＝　’２５’という条件を引数としてプログラムを呼び出す。Ｖ－ＦＲＥＮＤではこの引数とあらかじめ格納されていた検索条件から

```
      ＮＡＭＥ
from  Ｒ
where ＣＩＴＹ　＝　’ＮＹ’
      ＡＧＥ　＝　’２５’
```

という検索条件を生成する。次に、データベース検索命令（ＳＥＬＥＣＴ命令）を実行し、検索結果をホスト計算機に送り返す。この例ではタプル’ＳＭＩＴＨ’が検索され、正常終了コード’００’とともに送り返される。図３は、この例を示す。左側はプログラムで、右側はレジスタの内容

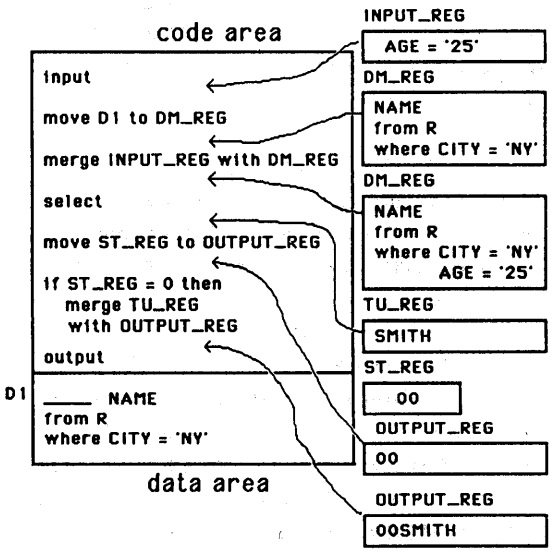


図３　仮想命令の実行

である。

(1) ＩＮＰＵＴ命令によりＣＭＥＸから起動時の引数を受け取る。ＡＧＥ＝’２５’がＩＮＰＵＴ＿ＲＥＧにセットされる。
(2) ＳＥＬＥＣＴ命令のパラメタがＰＳＵのデータ領域からＤＭ＿ＲＥＧに転送される。
(3) ＩＮＰＵＴ＿ＲＥＧの内容（ＡＧＥ＝’２５’）とＤＭ＿ＲＥＧの内容が、連結される。
(4) ＳＥＬＥＣＴ命令が実行される。ＤＭ＿ＲＥＧの内容を命令のオペランドとしてＤＭＥＸが起動され、検索が行われる。タプル’ＳＭＩＴＨ’が検索されＴＵ＿ＲＥＧに格納される。この例ではＳＥＬＥＣＴ命令は正常終了しているので、正常終了コード’００’がＳＴ＿ＲＥＧに設定される。
(5) ＳＴ＿ＲＥＧの内容がＯＵＴＰＵＴ＿ＲＥＧに転送される。
(6) ＳＥＬＥＣＴ命令の終了状態がチェックされる。プログラムのこれ以降の部分が実行されるかどうかを判断する。
(7) ＴＵ＿ＲＥＧの内容（ＳＭＩＴＨ）がＯＵＴＰＵＴ＿ＲＥＧに転送され、終了コードと連結される。
(8) 出力命令が実行され、ＯＵＴＰＵＴ＿ＲＥＧの内容がホスト計算機へ転送される。

Ｖ－ＦＲＥＮＤはあくまでデータベースマシンであり、目的はデータベース管理を高速化することにある。しかし、このように手続き的処理が記述できることで、応用システムへの対応を容易にすることができる。

３．４　プログラムの作成と実行

３．３節で述べたプログラムの作成と実行のためには、プログラムの作成、ＰＳＵへの格納、および起動が必要となる。この節では、それらについて述べる。

(1) プログラムの作成

プログラムの編集、コンパイルといった作業は、ワークステーション等のホスト計算機上で行う。ホスト計算機上のプログラミング環境として、コンパイラやアセンブラが必要となる。また、Ｖ－ＦＲＥＮＤ開発時においてはリモートにあるプログラムのデバッグツールが重要であった。このデバッグツールはプログラム開発時にも有効であろう。

ホスト計算機上で最終的にＶ－ＦＲＥＮＤ命令列となったプログラムは、Ｖ－ＦＲＥＮＤに格納する必要がある。プログラムは通常のデータを格納するのと同様に"insert"コマンドによりＶ－ＦＲＥＮＤのリレーションにインサートされる。
(2) プログラムの準備（ＯＰＥＮ）

リレーションに格納されたプログラムは、実行前にＰＳＵにロードする必要がある。この処理はホストからのＯＰＥＮコマンドによって行われる。まずコード領域、データ領域が確保される。次にコード領域にプログラムコードが読み込まれ、初期設定を必要とするデータが初期設定される。このプログラムはリエントラントであるので、既にロードされているプログラムがＯＰＥＮされた場合はプログラムコード部分は共有されるだけで新たにロードはされない。もちろんこの場合でもデータ領域は確保され初期設定される。
(3) 実行（ＲＵＮ）

ＰＳＵにロードされたプログラムはホストからのＲＵＮコマンドによって起動される。ＲＵＮコマンドには実行するプログラムの名前とプログラムに対する引数が含まれる。Ｖ－ＦＲＥＮＤは起動されたプログラムの仮想命令の実行を開始する。
(4) 終了（ＣＬＯＳＥ）

ＯＰＥＮされたプログラムはＣＬＯＳＥコマンドにより完了する。これによりプログラムのデータ領域が解放される。しかしこのプログラムが他の利用者に共有されている場合にはコード領域は解放されない。

以上のシーケンスを図４に示す。

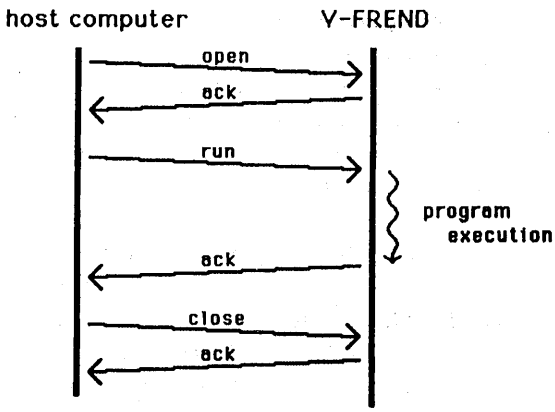


図４　プログラム実行のシーケンス

4．評価

簡単なトランザクション処理の例を用いてＶ－ＦＲＥＮＤによる通信オーバヘッドの改善を評価する。例として用いるトランザクションＴは次の通りである。

例：トランザクションＴは、リレーションＡにタプルを、追加する。次にリレーションＢのタプルを検索し、更新する。

トランザクションＴを、プログラマブルでないＦＲＥＮＤで実行する場合のホストとＦＲＥＮＤのインタラクションを示す（図５）。まず、ホスト計算機は、ＦＲＥＮＤに対して"start transaction"コマンドを出す。これによりリレーションＡとリレーションＢがオープンされ、ロックされる。次に"insert"コマンドにより、リレーションＡにタプルを追加する。さらに、"update"コマンドにより、タプルを検索し更新する。最後に"end_transaction"コマンドによりこれらの変更がコミットされ、リレーションＡとリレーションＢのロックが解除される。

一方、Ｖ－ＦＲＥＮＤでは、これらの処理は予めプログラムしてＶ－ＦＲＥＮＤ内に格納しておくことになる。このプログラムは以下のようになる。

(1) ホスト計算機から、追加するタプル、リレーションＢの検索条件値、およびタプルＢの更新後値をパラメタとして受け取り、リレーションＡ、Ｂをロックしてトランザクションを開始する。

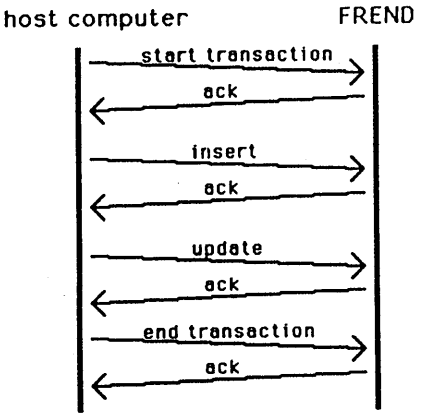


図５　ＦＲＥＮＤでのインタラクション

(2) 受け取ったパラメタによりリレーションＡにタプルを追加する。
(3) リレーションＢからタプルを検索し、これを更新する。
(4) トランザクションを終了する。
(5) 結果をホスト計算機に送り返す。

この例ではＶ－ＦＲＥＮＤによってインタラクションの数が４回から１回に減っている。

図６に通信回線の速度と応答時間を示す。このグラフはＦＲＥＮＤの実際のレスポンスタイム、通信量、およびＬＡＮのオーバヘッドにより求めたものである。これにより、実際の処理における通信オーバヘッドの割合が示される。

回線速度が遅い(1200bps、2400bps)場合には、ＦＲＥＮＤの応答時間はＶ－ＦＲＥＮＤの応答時間より大きい。回線速度が速くなるにしたがって両者のギャップは小さくなる。これは通信速度の増加によって通信量のオーバヘッドが減少するためである。

しかし、回線速度が無限大となっても、両者の間には全体の処理時間に対して無視できない差が残る。これは通信回数によって生じるＬＡＮのプロトコルオーバヘッドである。

Ｖ－ＦＲＥＮＤは、通信量と通信回数両方の削減を可能にする。したがって、このアプローチは低速の回線による広域ネットワークシステムにも、比較的高速のＬＡＮをベースにしたシステムにも有効であるといえる。

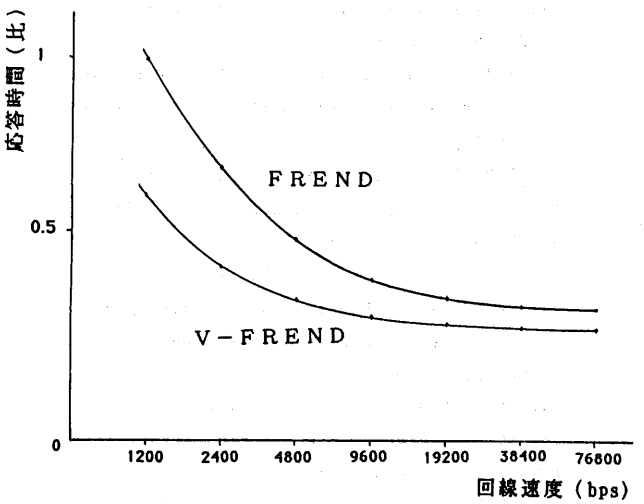


図６　応答時間の比較

5．結論

プログラマブルデータベースマシンへのアプローチとデータベースマシンＦＲＥＮＤでの実現例について述べた。仮想データベース計算機Ｖ－ＦＲＥＮＤは通信オーバヘッドを改善するためにプログラム実行環境を持つ。これにより応用システムとしての利用に適したデータベースマシンとホスト計算機のインタフェースを提供できる。しかもＦＲＥＮＤの持つ性能向上のための特長を保つことができる。

本稿では、クライアントサーバモデルだけについて述べた。複数のデータベースマシンによる分散データベースシステムでは”パイプラインジョイン”[HIKI85b]のようなデータベースマシンの協調動作が必要となる。仮想データベース計算機の考えは、このような分散データベースシステムにも適応できると思われる。

現在Ｖ－ＦＲＥＮＤはインプリメントが終り、機能と性能の評価を行っている。分散環境への適応が今後の課題である。

〈参考文献〉

[ANSI85] American National Standard Institute. "Draft Proposed American National Standard Database Language SQL", X3. 135-198x project: 363-D, 1985

[BIRR84] A. D. Birrel, et al., "Implementing Remote Procedure Calls", ACM TOCS, Vol. 2, No. 1, 1984 pp39-59.

[BITI83] D. Bitton, et al., "Benchmarking Database Systems A Systematic Approach", Proc. of VLDB, 1983, pp. 8-19.

[CANA74] R. H. Canaday, et al., "A Back-end Computer for Data Base Management", CACM, Vol. 17, No. 10, 1974, pp. 575-582.

[EPST80] R. Epstein, et al., "Design Decisions for the Intelligent Database Machine", Proc. AFIPS, Vol 49, 1980, pp237-241.

[HIKI81] S. Hikita, et al., "Optimization of the file access method in content addressable database access machine (CADAM)", Proc of AFIPS conference 1981, Vol 50, 1981, pp. 507-513.

[HIKI85a] S. Hikita, et al., "Database Machine FREND", Database Machines Forth International Workshop, Springer-Verlag, 1985, pp. 190-207.

[HIKI85b] S. Hikita, et al., "A Stepwise Approach to Distributed Database System by Database Machines", Proc. of 1985 ACM Symposium on Small Systems, 1985, pp. 18-24.

[METC76] R. M. Metcalfe, et al., Bogges, D. R., "Ethernet: Distributed Packet Switching for local computer networks, CACM Vol. 19, No. 7, 1976, pp. 395-404.

[SHIB84] S. Shibayama, et al., "A Relational Database Machine with Large Semiconductor Disk and Hardware Relational Algebra Processor" New Generation Computing 2, OHMSHA LTD. and Springer-Verlag, 1984, pp. 131-155

[STON81] M. Stonebraker, "Operating System Support for Database Management" CACM, Vol. 24, No. 7, 1981, pp. 412-418.

[STON83] M. Stonebraker, "Performance Enhancements to Relational Database System", ACM TODS, Vol. 8, No. 2, 1983, pp. 167-185.

[TSUR84] S. Tsur, et al., "An Implemetation of GEM - Supporting a Semantic Data", Proc. of SIGMOD '84, Vol. 14, No. 2, 1984, pp. 286-295.

[UBEL85] M. Ubell, "The Intelligent Database Machine (IDM)", Query Processing in Database Systems, Springer-Verlag, 1985, pp. 237-247.

[植村80] 植村、前川, "情報処理叢書１：データベースマシン", 1980, 情報処理学会, pp18-23.