超高速 DB アルゴリズムの超並列環境下のシミュレータ開発

Evaluation of the Ultra High Speed DB algorithm (LFM) on Massive Parallel Environments

古庄 晋二
Shinji FURUSHO

株式会社ターボデータラボラトリー
（〒231-0004 神奈川県横浜市中区元浜町3－21－2 ヘリオス関内10階 E-mail: furusho@digo-tech.com）

ABSTRACT. Today, many people, company or organization need to operate huge data. But it is very rare if the system satisfy the user. That is because until now the unified theory for processing gigantic data is not known. LFM (Linear Filter Method) is that theory found. With LFM based application and PC, people can search, sort, make cross tabulation and do many transforming operations on tables within sub-second that have millions of records. One can create joined tables having up to 2 billion records and can search, sort and do more in several seconds. This software is already available, but you can accelerate these operations more than 100 times. Because LFM is not good for current computer architecture, instead is suitable for massive parallel architecture. The mission for this time project was to prove this by simulator. The mission was very successful and we've got great knowledge.

1．背景

今日では至る所で大規模なデータの処理が必要とされているが、そのための一般的で幅広く適用できる手法は開発されていなかった。ために、現実のシステムではユーザーニーズを大幅に制約し、特定目的のみの高速化が図られてきた。

用途を制約する例では、DWH や OLAP のような集計処理に特化したもの、SCM のように特定処理に特化したものなどがある。

また、手法を限定するものもあり、まずデータファイル全体にアクセスを行い、そのときにインデックス付けやその他の必要な情報をメモリ上に展開し処理するものなどがある。（バッチ処理にしか使用できない。）

しかし、このようなアプローチは適用範囲の狭いその場しのぎの方法であり、決して満足なものではない。

そもそもユーザは本来幅広く情報を活用したいという強い願望を持っているものだが、このようにシステムの使用方法を限定する手法は根本的にこれと相容れない。

さらに依然としてスピードが不十分である。

このような問題を解決するためのアプローチが 1998 年に始まった。下記にその基本方針を記す。

方針1. データの基本形式をテーブルとする。
方針2. 表を3つの情報成分（表現、位置、値）に分解して保持する。このデータ構造を FAST(Filter Array Structure)と呼ぶ。
方針3. 検索や集計やソートに限らず、全てのデータ処理を FAST 上で組み立てる。これらのアルゴリズム群を LFM (Linear Filter Method)と呼ぶ。

情報が成分に分解されると一切のインデックスが不要になり、インデックスというアプローチ自体に起因する様々な問題が解決できる。

1）**項目間の均等性**：インデックスの付加された項目、付加されていない項目という区別が無くなり、どの項目にも等しく超高速にアクセスできる。この特性により、今まで不可能であった大規模データとの対話型の編集･加工･分析が可能になった。また、システム設計工数を大幅に削減できるようになった。
2）**FAST 構造の再生成性**：JOIN などの操作により生成されたテーブルではそのまま FAST 構造が再生産されるので、インデックスを付加するという高コストのステップを経ることなく、継続して超高速アクセスができる。
3）**部分集合の効率性**：インデックスはテーブルの全レコードに対して生成されたものであり、部分集合に対しては特性が悪いが、LFM にはこのような欠点がない。
4）**データの圧縮**：インデックスはデータサイズを増加させるが、FAST はデータサイズをコンパクトにする。
5）**単一構造**：インデックスは用途に応じていくつものタイプがあり、システムを複雑化させるが、FAST は1つの構造しか持たない。

また、位置の情報成分と値の情報成分が分離されたことで、LFM でのソートは O(n)で実現が保証されるようになった。他にも処理量における線型性が現れるようになり、これが Linear という言葉を使用する由来となった。

上記の特性により、特に下記の場合に効果があることが実際のデータを用いたベンチマークなどで実証されている。

1） 多段階の処理を必要とするプロセスの高速化

数百万行の大規模なデータの BOM, MRP などでは凡そ 500 倍くらい高速であった。

２）どのような処理を行うかが予め想定できない、大規模データの対話型加工･分析システム

2. 目的

(基本的な問いかけ)

FAST+LFM の開発とベンチマークテストなどを通じて、これまで大規模データの処理性能が低かったのは CPU へ高速にデータをフィードする手法が確立されていなかったためであることが明らかになってきた。

すると自然と次の問いにつながってゆく。

『超並列コンピューティング環境においても、各 CPU へ効率的にデータをフィードすることができるか？』

もしこの問いに Yes という解を出すことができれば、実用的かつ汎用的な超並列コンピューティング装置を設計することが可能になる。

(目標)

そこで本プロジェクトでは、以下の目標を据えた。

1. FAST+LFM の拡張
   超並列環境下で効率的に機能するように FAST+LFM を拡張すること。

2. モデルアーキテクチャの決定
   下記の要件を満たす適切な超並列アーキテクチャの決定を行うこと。
   要件1. TB級の大規模なメモリを保持可能
   要件2. 検索･集計･ソート等の基本処理が高速
   最低値保証可能であること
   要件3. ユーザロジックも高速に実行可能

3. 超並列環境下での高速性の確認
   ソフトウェアより100倍以上高速であること
   例：1億行を0．1秒未満のソートを保証

タイトルにあるシミュレータは上記目的を達成するためのツールとして製作･使用するが、目標にはしなかった。

(実現イメージ)

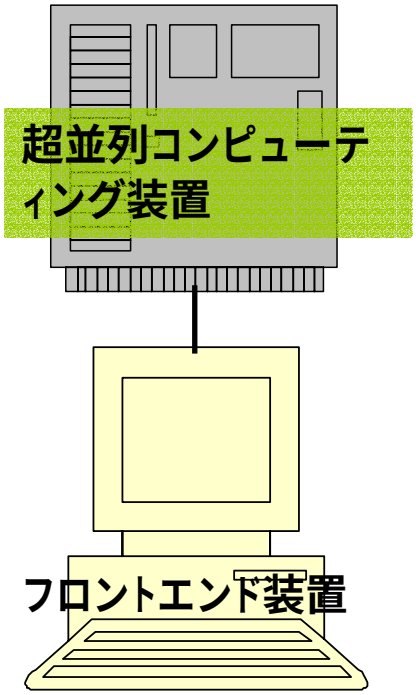
現在、グラフィックアクセラレータは 100G Flops の性能があるといわれ、これは汎用の CPU の 1G Flops より100倍速い。このような性能があってこそ、高解像度ディスプレーの上で自在に画像を描画できる。

同様に、誰もが自由自在に大規模なデータを編集･加工･分析集計できるシステムを本プロジェクトの達成イメージとした。1億行程度のテーブル群なら1クリックで自在に操作できるようにしたい。

3. モデルアーキテクチャの決定

(ロッカー型装置を想定)

ターゲットとするシステムは、TB級のメモリを保持する必要性から当面、グラフィックアクセラレータボードのようなコンパクトなものではあり得ない。そこで、汎用コンピュータのバックエンドに、別の筐体に収容された超並列装置を接続することを想定した。(図1)



(図1) ロッカー型装置を想定

(分散メモリ型)

超並列であっても効率を落とさず運用できるアーキテクチャの実現を目指すことから、分散メモリ型とした。

(分散メモリモジュール間接続方式の決定)

各分散メモリモジュール間の効率的な接続方式を研究した。得られた接続方式は安価で実現も容易だが、理論から導かれる検索･集計･ソート等の基本的な処理の性能をきちんと実現できるものになった。

(モジュールの内部構成の決定)

各分散メモリモジュールの内部構成を決めた。主なデバイスとして以下のものが搭載される。

1. 通信プロセッサ
   他のモジュールと通信を行いながら、モジュール内部のデータを適切に更新する機能を備える。

2. 汎用プロセッサ
   ユーザロジックを含む多様な処理を実現するためのもの。

3. メモリ
   処理の最低値の保証を行えることは重要であるため、キャッシュなどの統計的メカニズムを一切使用しない設計とした。これにはアーキテクチャがシンプルになるという副次的効果もあった。

4. データ構造とアルゴリズムの拡張

(データ構造の拡張)

多数の分散メモリモジュールによる超並列環境での運用でも効率をほとんど落とさないようにデータ構造：FAST の拡張を行うことができた。

(アルゴリズムの拡張)

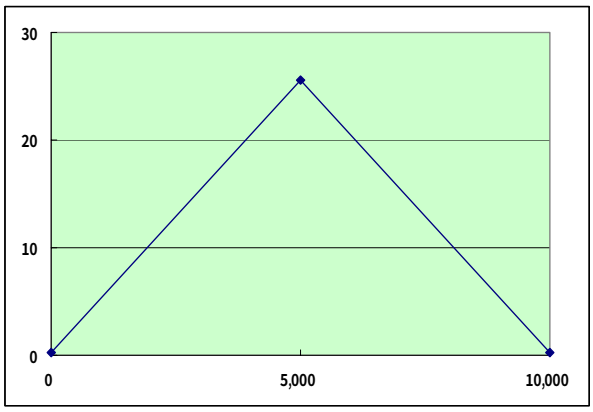
FAST の拡張に伴って LFM も拡張することに成功した。

5．高速性の確認

検索･集計･ソート･JOIN の各処理について高速性を確認した。

(検索)

1億行のテーブルから検索を実行する時間を調べた。グラフ1の縦軸は検索の要した時間（ミリ秒）、横軸はヒット件数（単位は万）である。



グラフ1．1億行からの検索実行時間

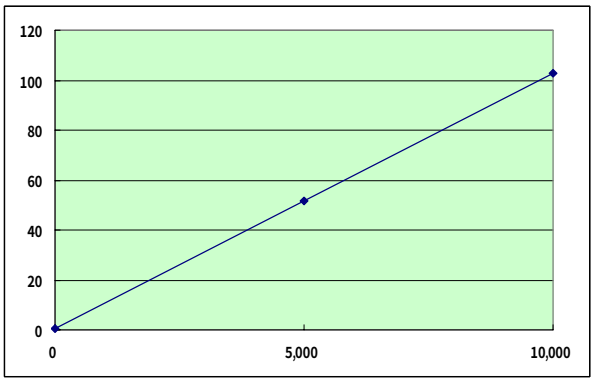
また、5000 万行がヒットした場合の処理時間の内訳は表1のようになった。合計で 25.9 ミリ秒になった。

消費時間－1	0.1 mSec
消費時間－2	0.1 mSec
消費時間－3	25.6 mSec
消費時間－4	0.1 mSec

表1．1億件中5千万行ヒット時の所要時間

(ソート)

1億行のテーブルの部分集合をソートする時間を調べた。グラフ2の縦軸はソートに要した時間（ミリ秒）、横軸は部分集合のサイズ（単位は万）である。



グラフ2．1億行のテーブルのソート時間

また、1億行をソートした場合の処理時間の内訳は表2のようになった。合計で 102.8 ミリ秒になった。

消費時間－1	0.1 mSec
消費時間－2	0.1 mSec
消費時間－3	0.0 mSec
消費時間－4	0.2 mSec
消費時間－5	102.3 mSec
消費時間－6	0.1 mSec

表2．1億行全行ソート時の所要時間

(そのほかの性能)

上記の検索･ソート性能の他、集計、JOIN などの諸性能の評価を行い、いずれも目標を達成できるものであることが分かった。

5．今後の展望

(機能の拡充)

長い間、大規模データを処理するシステムではその基盤ツールとしてＲＤＢＭＳを使用することが多かった。

本来は大規模データ処理には大切であるのにＲＤＢＭＳでは存在しない機能では、その必要性が認識されていないものがたくさんある。一例に項目転送という機能がある。これは JOIN で結合された2つのテーブル間で項目を転送する機能である。あるいは、表計算の埋め込み計算式のようにセル単位で自動計算する機能などである。

このようなさまざまな機能を実現し、かつユーザが自ら超並列環境上のプログラムを開発して実行できるようにできれば真の高速化が達成される。

(マーケットへの提供)

第1に現在のソフトウェアによる大規模データ処理エンジンを普及させ、本プロジェクトの成果としての超並列装置のニーズを確定すること、第2に超並列装置の開発･販売･サポートのできる組織を作ることが必要である。

そのためマーケットへの提供（発売）までには数年以上の時間がかかるが、着実に進めてゆく。

6．参加企業

財団法人京都高度技術研究所（プロジェクト管理組織）
株式会社ターボデータラボラトリー（開発支援作業）

7．謝辞

本プロジェクト実施の機会を与えてくださった喜連川プロジェクトマネージャー、契約作業などに適切なアドバイスを下さった財団法人京都高度技術研究所、未踏のソフトウェアのプロジェクトを行っている情報処理振興事業協会の皆様に心から感謝を申し上げます。

本プロジェクトの成果を製品化し皆様のご厚意に報いるべくこれからも努力を続ける決意です。

8．参考文献

特になし