クエリプランを利用した先読み技術の開発と初期評価

出射 英臣　茂木 和彦　西川 記史　大枝 高

（株）日立製作所 システム開発研究所
〒215-0013　神奈川県川崎市麻生区王禅寺 1099
E-mail: {idei, mogi, norifumi, oeda}@sdl.hitachi.co.jp

あらまし

近年、急速に普及したデータウェアハウス等の IT システムにおける各種アプリケーションの基盤として、DBMS(RDBMS)の重要性は益々高まっている。DBMS が管理するデータは年々爆発的に増加しており、大規模データベースにおける検索性能の向上は更に増して重要となっている。以上の背景から、RDBMS がクエリ実行時に作成するクエリプランを基に先読みを実施するクエリプラン利用先読み技術を研究している。本技術では、B-Tree 索引を利用した検索処理において、RDBMS が今後アクセスするデータを特定し、そのデータをストレージキャッシュに先読みすることで検索性能の向上を図る。今回、本技術を実装したプロトタイプシステムを開発し、初期評価を実施した。その結果、本技術により、B-Tree 索引を利用した検索処理を含むクエリの実行速度が最大 5.9 倍に向上可能なことを実証した。

キーワード　ストレージ，RDBMS，クエリプラン，先読み，性能評価

Development and Premininary Evaluation of Prefetch Technique Using Query Plan.

Hideomi Idei,　Kazuhiko Mogi,　Norifumi Nishikawa　and　Takashi Oeda

Systems Development Laboratory, Hitachi, Ltd.
1099, Ouzenji, Asao-ku, Kawasaki-shi, Kanagawa, 215-0013, Japan
E-mail: {idei, mogi, norifumi, oeda}@sdl.hitachi.co.jp

Abstract

In recent years, the importance of DBMS(RDBMS) is increasing as a base software of the various application programs in IT systems such as datawarehouses which have become into wide use.　Because the amount of the data managed by the DBMS is growing explosively every year, the improvement of the performance to search the data stored in a large database is quite important.　From the above background, we have studied a prefetch technique based on the query plan made by the RDBMS at the time of query execution.　With this technique, the data which may be accessed by the RDBMS using B-Tree indexes are specified based on the query plan and prefetched into a storage cache in order to improve the search performance. We developed a prototype system of this technique and carried out a preliminary evaluation.　The evaluation result shows that the execution time of the query with the data search using B-Tree indexes with this technique become 1/5.9 shorter than that without this technique at the best case.

Keyword　Storage, RDBMS, Query Plan, Prefetch, Performance Evaluation

1. はじめに

近年、急速に普及したデータウェアハウス、CRM、SCM、OLAP、データマイニング等の IT システムにおける各種アプリケーションの基盤として、DBMS の重要性は益々高まっている。DBMS が管理するデータは主に大容量のストレージに記憶されるが、そのデータ量は年々爆発的に増加しており、大量のデータの中から必要なデータを検索する処理における検索性能の向上は更に増して重要となっている。

ストレージに記憶してあるデータにアクセスする一連の過程の中で、ボトルネックになりやすい処理として、ストレージ内のディスクからデータを実際に読み出す処理があげられる。CPU 内部の処理等が数マイクロ秒～数十マイクロ秒オーダーで完了するのに対し、この処理ではディスクシーク等の機械的な動作を伴う為、処理時間が数ミリ秒オーダーになってしまうことがその理由である。ストレージでは、このボトルネックを解消する為、装置内に大容量のキャッシュメモリを設け、データの再利用や先読み処理によって性能を向上している。しかし、現在、この先読み処理によって得られる効果はシーケンシャルアクセスの場合に限定される。ランダムアクセスの場合、ストレージはどのデータを先読みして良いか判定する手段がなく、アクセス要求を受け付けてからデータを読み出すことになる。

以上の背景から、当社ではストレージの高性能化を図る技術を研究している。その一技術として、RDBMS がクエリ実行時に作成するクエリプラン（どの表や索引を、どの順で、どの様にアクセスするといったクエリ実行手順に関する情報。RDBMS は、このクエリプランに沿って処理を進める。）を基に RDBMS が今後アクセスするデータを特定し、ストレージキャッシュに先読みを行うことで、高性能化を図る機能（以下、クエリプラン利用先読み機能）の検討を行っている。本機能を用いた場合、上記ランダムアクセス時でも先読みを実施することができる為、キャッシュヒット率を高め、リード I/O 性能を向上することが可能となる。今回、クエリプラン利用先読み機能を実装したプロトタイプシステムを開発し、本機能の効果を実証する初期評価を実施した。本論文では、その内容をまとめる。

RDBMS のクエリプランを利用して先読みを行う技術に関するその他の研究例として、“高機能ディスクにおけるアクセスプランを用いたプリフェッチ機構に関する評価”をあげることができる[1]。この論文では、アクセスプラン（クエリプラン）を用いたプリフェッチ（先読み）によって、どの程度の性能向上が見込まれるか調べる為にシミュレーション実験を行っている。本論文では、索引解析による先読み処理のアルゴリズムを明確にし、実機によって本機能の評価を実施している。

また、データを利用するアプリケーションの内部に手を加え、先読みすべきデータを指示することによって先読みを行う技術が“Informed Prefetching and Caching”では論じられている[2]。この論文では、先読みによりリード I/O 性能は向上するが、アプリケーションの大幅な改造が必要となる。一方、本機能では RDBMS に手を加えずに（場合によっては若干手を加える必要がある）先読みを実施することができる。

以下、2 章では、クエリプラン利用先読み機能プロトタイプシステムの概要、本機能における先読み処理の概要、本機能による検索性能の考察について述べる。3 章では、性能測定環境、初期評価結果について述べる。4 章では、本論分のまとめについて述べる。

2. クエリプラン利用先読み機能プロトタイプシステム

2.1. 概要

クエリプラン先読み機能は、RDBMS がクエリ実行時に作成するクエリプランを基に索引データを解析し、RDBMS が今後アクセスするデータを特定する。続いて、それらのデータをストレージキャッシュ上に先読みしておくことにより、リード I/O 時間を短縮して DB 検索性能の向上を図る機能である。本機能のプロトタイプシステムの概略を図 1 に示す。

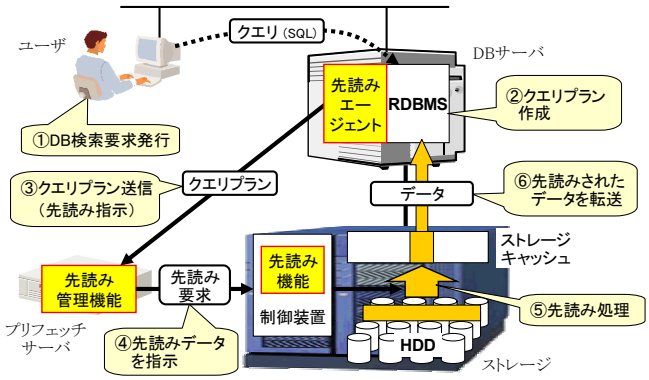


図 1　クエリプラン利用先読み機能
プロトタイプシステム概略

本機能は、図 1 に示す様に、RDBMS が動作する DB サーバ、DB データを格納するストレージ、先読み処理の主要となるプリフェッチサーバ（以下、PF サーバ）を接続したシステム構成において、DB サーバ、ストレージ、PF サーバに以下 3 つの機能をそれぞれ配した機能構成となる。尚、本プロトタイプシステムで用い

るストレージは、複数のディスクドライブ、大容量キャッシュメモリ、制御装置を有する高性能ストレージである。

(1) 先読みエージェント機能

DB サーバ上で動作する。先読み処理に必要となる情報を RDBMS や OS から取得し、PF サーバの先読み管理機能に送信する機能である。取得・送信を行う情報を表 1 に示す。

表 1　先読みエージェント機能の取得情報

#	情報名	情報内容
1	マッピング情報	DB サーバにおける DB オブジェクト（表・索引）－論理ユニット（ストレージ上のデータ記憶単位）間のデータマッピングに関する情報である。本機能の起動時に、RDBMS や OS から取得し、先読み管理機能に送信する。
2	索引情報	索引の定義に関する情報である。本機能の起動時に、RDBMS から取得し、先読み管理機能に送信する。
3	クエリプラン情報	クエリプランに関する情報である。クエリ実行開始前に、RDBMS から取得し、先読み管理機能に送信する。
4	I/O 先情報	RDBMS が発行した I/O 先に関する情報（論理ユニット識別子、アクセス先 LBA、データ長）である。I/O が発行される毎に、OS（デバイスドライバ）から取得し、先読み管理機能に送信する。

(2) 先読み管理機能

PF サーバ上で動作する。先読みエージェント機能から受信した情報に基づいて、索引ページを取得、解析し、今後 RDBMS がアクセスするデータを特定した後、それらのデータの先読み指示（本プロトタイプシステムでは SCSI プリフェッチコマンドを使用）をストレージの先読み機能に送信する機能である。

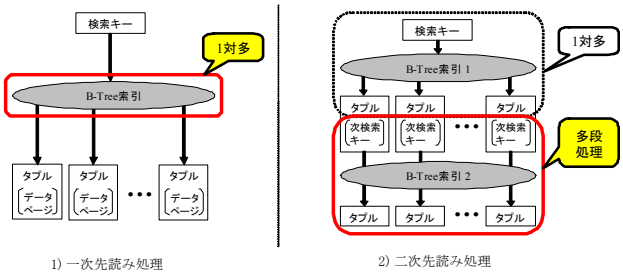
(3) 先読み機能

ストレージ上で動作する。先読み管理機能から受信した先読み指示に従い、対象のデータをディスクからストレージキャッシュ上に読み出す機能である。

2.2. 先読み処理方式

2.2.1. 先読み処理の概略

クエリプラン利用先読み機能における先読み処理は、上記 PF サーバの先読み管理機能が実行する。処理方式には、B-Tree 索引の索引ページ解析をベースに行う方式を採用し、その適用条件は、大きく次の二つに分類される。先読み処理方式の概略を図 2 に示す。



1) 一次先読み処理　　2) 二次先読み処理

図 2　先読み処理方式概略

1) 一次先読み処理

B-Tree 索引を利用して 1 つの検索キー値に対応した複数のデータを読み出す検索処理（1 対多検索処理）において、最初のデータが判明した時点で残りのデータの先読みを行う。

2)二次先読み処理

上記 1 対多検索処理後、読み出したデータを基に B-Tree 索引を利用して更に複数のデータを読み出す検索処理（多段検索処理）において、一次先読み処理で先読みしたデータを用いて次段の索引解析を行い、その結果を基にデータの先読みを行う。

クエリプラン利用先読み機能は、上記先読み処理の過程でディスク I/O を発行し、先読み対象となる複数のデータをストレージキャッシュ上に読み出す。この読み出しは、ストレージ内部でディスクドライブ毎に並列に行われる。従って、RDBMS からの次リード I/O までに対象となるデータをストレージキャッシュ上に読み出しておくことが可能となり、リード I/O 性能を向上させる。

従来の RDBMS では、1 つのクエリを実行する際に、I/O を一つずつ逐次的に発行するか（並列性なし）、もしくは、Oracle®の Data Block Prefetching[4]にみられるように、ある検索キー値に対応する複数のデータを取得する I/O を一括して発行する（上記一次先読み処理に相当する）。本機能の先読み処理では、二次先読みまで実施することにより、更に多くの I/O をストレージ内部で並列に処理することができる。つまり、ストレージ内部のディスクを高い並列度でアクセスし、より高い性能を得ることができる。

本機能における先読み処理は、索引ページを解析して先読みを行う為、RDBMS がアクセスした索引ページを取得する必要がある。しかし、本先読み処理は、DB サーバとは別の機器（PF サーバ）上で動作する為、RDBMS がどの索引ページを読み出したか判断できない。そこで、RDBMS がデータアクセスを行う毎に、DB サーバ上の先読みエージェント機能がそのデータ

アクセス先に関する情報（I/O 先情報）を取得し、PF サーバ上の先読み管理機能に送信する。先読み管理機能は、受信した I/O 先情報と、前もって受信しているマッピング情報を照合することにより、アクセス先のデータの内容を知ることができる。尚、先読みエージェント機能が I/O 先の情報を取得する方式としては、RDBMS から直接取得する方式と、OS（デバイスドライバ）から取得する方式が考えられるが、実験を容易にする為に後者の方式を採用した。

2.2.2. 先読み先決定アルゴリズム

前述の一次先読み処理、二次先読み処理における先読み先決定アルゴリズムについて、以下記述する。

(1) 一次先読み処理アルゴリズム

一次先読み処理のアルゴリズム概略を図 3 に示す。一次先読み処理では、以下の処理を実施する。

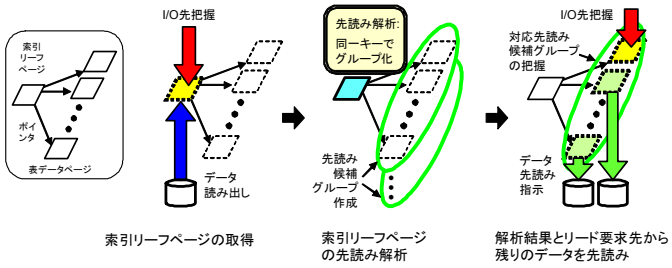


図 3　一次先読み処理アルゴリズム概略

1) 先読みエージェント機能から I/O 先情報を取得する。
2) 取得した I/O 先情報とマッピング情報を照合し、その I/O 先が索引のリーフページであるか判定する。その結果、I/O 先が索引リーフページの場合は、その索引リーフページをストレージから読み出す。
3) 読み出した索引リーフページの内容を解析し、その索引リーフページがポイントしているデータページをキー値によってグループ化した先読み候補グループを作成する。
4) データページへの I/O 先情報を取得した場合、その I/O 先のデータページを含む先読み候補グループを割り出し、そのグループに属するデータページの先読みを指示する。

(2) 二次先読み処理アルゴリズム

二次先読み処理では、上記の一次先読み処理に続いて、更に以下の処理を実施する。

1) 前段の一次先読み処理によってデータページを先読みすると同時に、そのデータをストレージから取得する。
2) 取得したデータページの内容を解析し、次段で検索キーとして利用される属性値を割り出す。
3) 割り出した検索キー値によって利用される索引を解析して、次にアクセスされる索引ページ、及びデータページを把握し、そのデータの先読みを指示する。尚、次にアクセスされると把握したページが索引ページの場合には、その索引ページ内部を解析して次の先読み処理に利用する為、先読み指示を行うと同時に、該当の索引ページを読み出す。

2.3. 本機能による検索性能の考察

本機能の先読み処理による検索性能は、前述した処理方式から概算することができる。今回の初期評価では、その考察を裏付ける評価を中心に実施する。

2.3.1. 選択行数

I/O が多発する検索処理の実行時間において、ストレージ内部のディスク I/O に要する時間が多くを占める。従って、従来処理では、選択行数が増加すれば（DB の規模が大きくなれば）ディスク I/O 数もそれに伴って増加する為、選択行数増加に比例して検索処理の実行時間も大きく伸びる。

本機能が有効に機能する条件下において、先読み処理を行った場合、ディスク I/O によってディスクから読み出していたデータを、ストレージキャッシュから読み出すことになる為、選択行数が増加しても検索処理の実行時間は、上記従来の場合と比較して比例係数の小さい伸びとなることが推測できる。

2.3.2. 先読み並列度

本論文では先読み並列度を、1 索引ページの解析から実際にどれだけのデータページの先読みが並列に行われたかを示す値とする。

本機能の先読み処理では、キー値毎にアクセスされた索引ページがポイントしているデータページをグループ化し、あるグループのデータページにアクセスがあった場合に同グループに属するデータページの先読みを並列に行う。この時、同グループに属しているデータページが多い程、同時に先読みを行うデータページも多く（先読み並列度も高く）なり、キャッシュヒットの割り合いも高くなる為、検索性能はより向上する。但し、並列度が高くなるにつれ、ディスク競合が原因でディスクがボトルネックとなり、本機能を用いた場合の検索性能は飽和すると推測できる。

3. 初期評価

3.1. 性能測定環境

前述のクエリプラン利用先読み機能を実装したプロトタイプシステムを開発し、そのプロトタイプシステムを用いて、初期評価に必要となる性能に関するデータを採取した。以下、性能測定の環境について述べる。

3.1.1. 機器構成

本プロトタイプシステムは、DB サーバーPF サーバ間を 100Mbps Ethernet と FC(Fibre Channel)で、DB サーバーストレージ間、及び PF サーバーストレージ間を FC で接続したシステム構成である。プロトタイプシステムのシステム構成を図 4、構成機種を表 2 に示す。

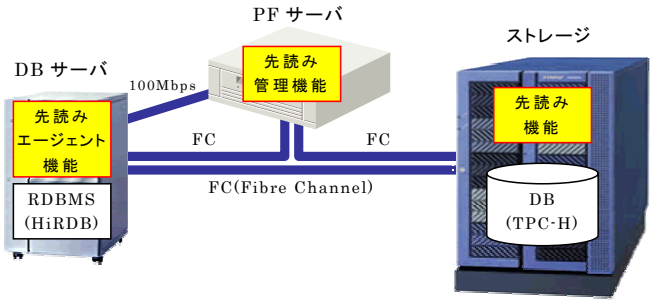


図 4　プロトタイプシステム機器構成

表 2　プロトタイプシステム構成機種

【DB サーバ】	
プロセッサ	Pentium3 1.13GHz
OS	Linux (Kernel 2.4.24)
RDBMS	HiRDB Version7
【PF サーバ】	
プロセッサ	Xeon 3.06GHz
OS	Linux (Kernel 2.4.24)
【ストレージ装置】	
機種	SANRISE 9570V
キャッシュ容量	2GByte
LU 構成	RAID5(4D＋1P)×3 グループ

3.1.2. DB 環境

性能測定に用いた DB は、解析系 DB システム性能を測る業界標準ベンチマークである TPC-H[4]のモデルを流用した。尚、DB 規模は SF(Scale Factor)=3、総データ量約 3GByte、索引ページサイズは 4KByte、データページサイズは 16KByte である。性能測定用のクエリには、本機能の効果が見込める TPC-H Query8（以下、Q8）を選択した。Q8 のクエリプランを図 5 に示す。

図 5 に示した様に、Q8 において、PART 表を検索した後、その結果と LINEITEM 索引（LINEITEM 表に付加された本処理で利用する索引）によって LINEITEM 表とネストループ結合する部分が一次先読み処理の対象範囲となり、更にその結果と ORDERS 索引(ORDERS 表に付加された本処理で利用する索引）によって ORDERS 表とネストループ結合する部分が二次先読み処理の対象範囲となる。その先は、ハッシュ結合である為、本機能における先読み処理の対象にはならない。

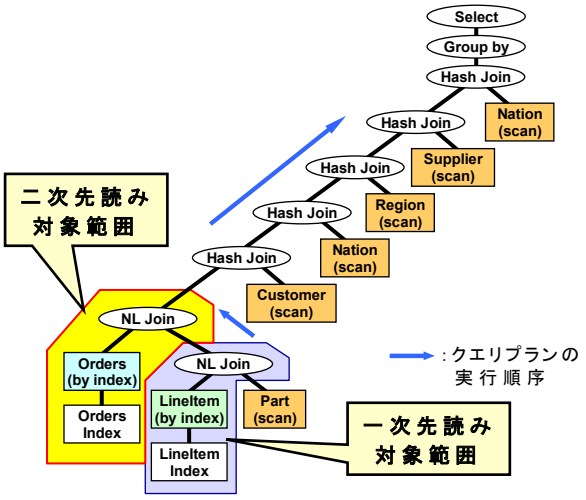


図 5　Q8 のクエリプラン

3.1.3. 性能測定に関して

(1) ディスクアクセスの並列性を確保する為、15 台のディスクを用いた。本機能の先読み処理ではデータの先読みを並列に行う為、ディスク台数が少ないとディスク競合が発生し、検索性能が十分に得られない可能性がある為である。

(2) DB バッファは Q8 の処理内容と先読み処理内容に合わせて、ORDERS 索引用 DB バッファ、LINEITEM 索引用 DB バッファ、その他のスキーマ用 DB バッファの 3 つを使用する。また、Terabyte クラスの大規模 DB を想定した場合、DB バッファは全データサイズの 0.数%～数%程度しか確保できないことから、ORDERS 索引用 DB バッファは ORDERS 索引データサイズの 5%（約 10MByte）、LINEITEM 索引用 DB バッファは LINEITEM 索引データサイズの 5%（約 15MByte）、その他のスキーマ用の DB バッファはその他のスキーマのデータサイズの 1%（約 29MByte）とする。

(3) 性能測定を実施する際、それぞれの測定を同一条件で行う為、測定毎に DB バッファとストレージキャッシュをクリアする。DB バッファ、及びストレージキャッシュにデータが残っていた場合、そのデータにヒットしてしまう為である。

(4) 測定結果のばらつきを考慮して、各々の測定は 3 回行い、その平均値を各測定の結果とする。

3.2. 初期評価結果

3.2.1. 先読み処理と検索性能に関する評価

本機能による先読み処理が有効に機能しているか実証するため、FC モニタを使用して、Q8 実行時に DB サーバがストレージに発行する I/O と、PF サーバがストレージに発行する I/O の I/O トレースを採取し、先読み処理の有無による I/O アクセスパターンの変化について解析を行った。

I/O トレースは、先読み処理なし、一次先読み処理実施、二次先読み処理実施の各場合について採取した。先読み処理に関係するスキーマの I/O トレースの解析結果を図 6 に示す。尚、縦軸は LBA（論理ブロックアドレス）に対応するスキーマ名、横軸は経過時間を示している。

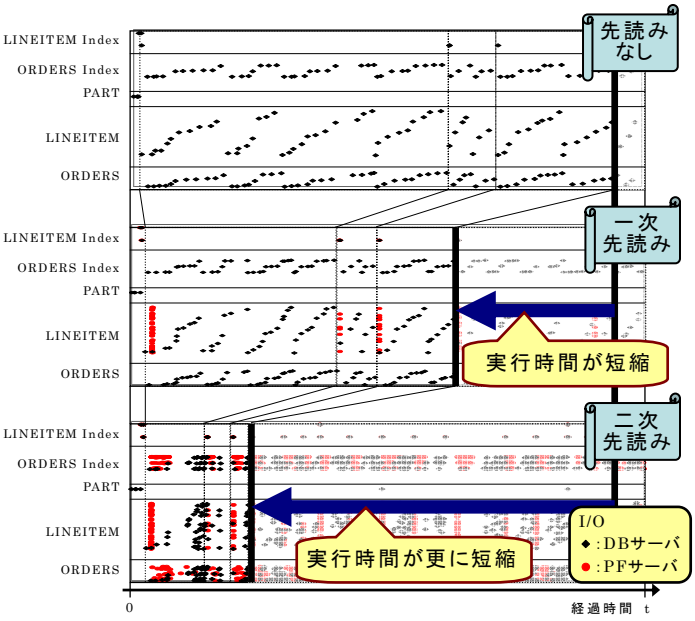


図 6　I/O トレース解析結果

図 6 では、先読みなし、一次先読み処理、二次先読み処理の各場合において、Q8 実行中のある一部の処理の I/O トレースを上から順番に並べて比較している。

図 6 の一次先読み処理を実施した時の I/O トレースを見ると、PF サーバが LINEITEM 索引ページを取得した後、LINEITEM データページの I/O(一次先読み指示)をほぼ同時に発行しているのが判る。また、DB サーバの I/O（データ読み出し）は、先読みなしの場合に比べて、その発行間隔が狭くなっている。これは、上記 LINEITEM データページが先読みされていることにより、I/O 時間が短縮された為である。

次に、図 6 の二次先読み処理を実施した時の I/O トレースを見ると、上記一次先読み処理を実施した時と同じく、PF サーバが LINEITEM 索引ページを取得した後、LINEITE データページの I/O をほぼ同時に発行し、それに続いて、ORDERS 索引ページと ORDERS データページの I/O（二次先読み指示）を発行していることが判る。図 2 で示した様に、二次先読み処理では、一次先読み処理の結果から更にその先の先読みを行う為、先読み対象となるデータ量は一次先読みのそれと比較してはるかに増加する。この先読み対象のデータ量の差が、図 6 の一次先読みと二次先読みの DB サーバの I/O 発行間隔に現れている。二次先読みを実施した場合の I/O 発行間隔は、一次先読みを実施した場合の I/O 発行間隔と比較して更に狭まっている。つまり、それだけ実行時間が短縮されているということである。

これらの I/O トレースを採取した時の Q8 実行時間のグラフを図 7 に示す。縦軸は先読みなしの Q8 実行時間を基準にした相対値である。

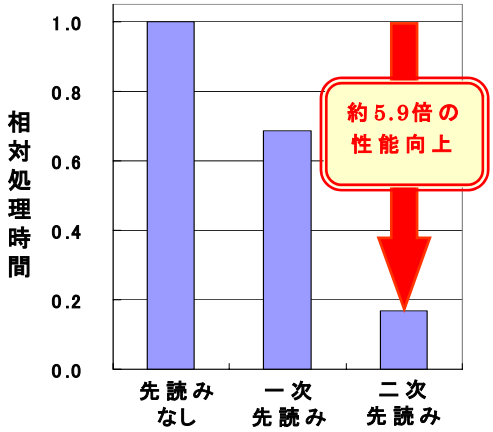


図 7　Q8 実行時間

次先読み処理を実施した場合、先読み処理を行わない場合と比較して Q8 の実行速度が約 5.9 倍に向上していることが図 7 から判る。以上の結果から、本測定環境における Q8 の実行において、本機能の先読み処理により、検索性能が向上したことを実証した。

3.2.2. 選択行数と検索性能に関する評価

続いて、2.3.1 章で考察した様に、選択行数が増加することによってクエリの実行時間も比例して向上すること、またその比例係数は、先読み処理を行わない場合と比較して先読み処理を行った方が小さいことを明確にする為、以下の評価を行った。

Q8 において選択行数を増減させるには、Q8 が PART 表の検索条件に合致する行からそれ以降の検索処理を行う為、PART 表の選択行数を変えることによって可能となる。そこで、Q8 における PART 表の検索条件を以下の様に変えて PART 表の選択行数を変更し、先読みなしと先読みあり（二次先読み処理）の場合について Q8 の実行時間を測定した。PART 表の検索条件と、その検索条件による PART 表の選択行数、選択行割り

合いを表 3 に示す。尚、今回の DB 環境において、PART 表の行数は 600,000 行である。また、TPC-H が提供する Q8 の実際の PART 表検索条件とその選択行数は、表 3 の最後の行にあたる。

表 3　PART 表の検索条件と選択行数

PART 表選択条件	PART 表 選択行数	PART 表 選択行割り合い
P_container = 'SM CASE' and p_size < 3	580	0.1%
P_container = 'SM CASE' and p_size < 5	1190	0.2%
P_container = 'SM CASE' and p_size < 7	1776	0.3%
P_container = 'SM CASE' and p_size < 9	2382	0.4%
P_container = 'SM CASE' and p_size < 11	2986	0.5%
P_container = 'SM CASE' and p_size < 13	3611	0.6%
P_type = 'ECONOMY ANODIZED STEEL'	4187	0.7%

測定結果をまとめたグラフを図 8 に示す。縦軸は Q8 実行時間、横軸は PART 表の選択行割り合いを示している。尚、縦軸の Q8 実行時間は、先読みありの PART 表選択行割り合い 0.1%の時の Q8 実行時間を基準にした相対値である。

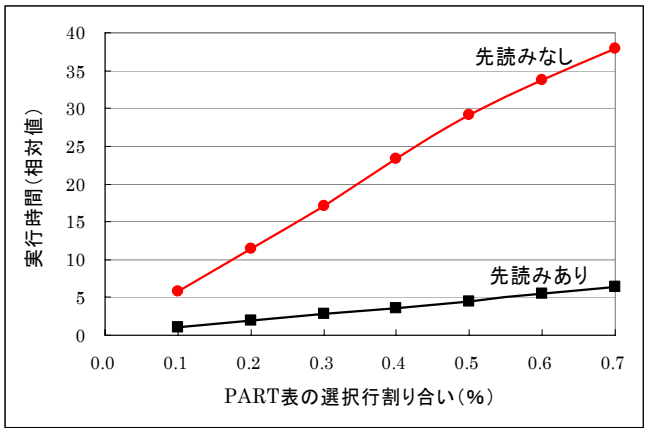


図 8　選択行数増加による Q8 実行時間

先読み処理を行わない場合、PART 表の選択行割り合いが増加するに従って Q8 実行時間も比例して伸びていることが図 8 から判る。選択行数が増加したことにより、処理時間を最も要するディスク I/O が増えた為だと考えられる。

また、先読み処理を行った場合も、PART 表の選択行割り合いが増加するに従って Q8 実行時間も比例して伸びているが、先読み処理を行わない場合と比較するとその比例係数は小さいことが図 8 から判る。先読み処理を行わない場合には逐次的に発行されていたディスク I/O が、本先読み処理によって並列に実行される為である。

以上のことから、先読み処理を実施した場合、選択行数が多い（DB 規模が大きい）程、先読み処理を実施しない場合と比較して検索性能は向上すると考えられる。

3.2.3. 先読み並列度と検索性能に関する評価

続いて、2.3.2 章で考察した様に、本機能の先読み処理を実施した場合、先読み並列度が高くなると検索性能も向上することを実証する為、以下の評価を行った。

先読み並列度の変更は、LINEITEM 表と PART 表との結合演算時の結合キー分布を変えることにより実現した。つまり、先読み並列度毎に異なるデータを用いている。今回、測定した先読み並列度は、1、2、4、8、16、30 である。尚、TPC-H で用いるデータの先読み並列度は多少ばらつきがあり、平均で約 30 となる。

測定結果をまとめたグラフを図 9 に示す。縦軸は性能向上倍率、横軸は先読み並列度を示している。尚、性能向上倍率は、先読みなしの Q8 実行時間を先読みありの Q8 実行時間で割った値である。

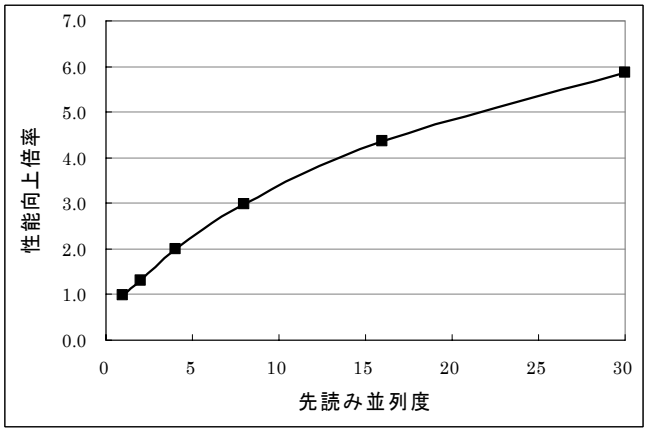


図 9　先読み並列度増加による性能向上率

2.3.2 章で推測した様に、先読み並列度が低い時は、性能向上倍率も低く、先読み並列度が高くなるにつれて性能向上倍率も高くなることが図 9 から判る。

先読み並列度が低い場合は、先読み対象となるデータページも少なくなり、先読み処理によってストレージキャッシュに読み出しておくデータも少なくなる為、本先読みの効果が得られにくくなる。従って、先読み処理を実施した場合としなかった場合のクエリ実行時間の差は小さく、性能向上倍率も低くなると考えられる。先読み並列度が高くなるにつれて性能向上倍率が高くなるのは、先読み処理によってストレージキャッシュに読み出しておくデータが多くなり、ストレージキャッシュのヒット率も高くなる為だと考えられる。

また、先読み並列度に比例して性能向上倍率も伸びるのではなく、先読み並列度が 5 を超えた辺りから性能向上倍率の伸びが緩やかになっていることも図 9 から判る。2.3.2 章で考察した様に、先読み並列度が高くなるにつれて、ディスク競合が発生するケースが多くなり、その影響がクエリ実行時間に現れた為だと考えられる。

4. まとめ

RDBMS がクエリ実行時に作成するクエリプランを基に先読みを実施するクエリプラン利用先読み機能を研究している。本機能は B-Tree 索引を利用した検索処理において、RDBMS が今後アクセスするデータを予測し、そのデータストレージキャッシュに先読みすることで検索性能の向上を図る。今回、本機能を実装したプロトタイプシステムを開発し、初期評価を実施した。その結果、本機能により、B-Tree 索引を利用した検索処理を含むクエリの実行速度が最大 5.9 倍に向上可能なことを実証した。

謝辞

本研究に関して御指導いただきました東京大学の喜連川教授に感謝致します。

本論文で記された技術には、文部科学省が実施するリーディングプロジェクト「e-Society 基盤ソフトウェアの総合開発」のストレージ・データベース融合技術（東大，日立）で技術開発された成果が反映されています。

参考文献

[1] 向井景洋，根本利弘，喜連川優，“高機能ディスクにおけるアクセスプランを用いたプリフェッチ機構に関する評価”，DEWS00 講演論文集 3B-3

[2] R. H. Patterson, et al. “Informed Prefetching and Caching,” In Proc. of the 15th SOSP, pp. 79-95, Dec. 1995.

[3] Oracle Corporation, “Performance and Scalability in DSS Environment with Oracle9i An Oracle Technical White Paper,” pp.12-13, Apr. 2001.

[4] TPC BENCHMARK™ H Standard Specification Revision 1.3.0 仕様書 http://www.tpc.org/tpch