様式Ｆ－１９

科学研究費助成事業（学術研究助成基金助成金）研究成果報告書

平成２５年３月３１日現在

機関番号：３２６８９
研究種目：挑戦的萌芽研究
研究期間：2011 ～ 2012
課題番号：23650053
研究課題名（和文）　超高速ストレージ環境でのファイルＩ／Ｏ処理におけるＣＰＵボトルネックの除去
研究課題名（英文）　Eliminating CPU Bottleneck of File I/O on Ultra High Speed Storage Environments
研究代表者
上田　高徳（UEDA　TAKANORI）
早稲田大学・ＩＴ研究機構・助手
研究者番号：90546863

研究成果の概要（和文）：
本研究の成果には、関係データベースにおいて重要な演算である Sort や Join について、超高速 SSD 環境における新たな実行方法の可能性を示唆した点がある。研究成果は DEIM2013 において発表した。また、データストリームの超低レイテンシ並列処理手法も提案した。処理に必要な通信回数を削減することで超低レイテンシ処理を実現する。提案内容を DEIM 2012 において発表したところ最優秀論文賞を受賞し、高い評価を受けた。また、電子情報通信学会論文誌にも採録されている。

研究成果の概要（英文）：
The achievements of this research include proposing a new execution method of sort and join that are important operations in relational database systems. The details were presented at DEIM 2013. Also, this research contributes to build a low latency parallel processing method for data streams. It reduces the number of communications between CPU cores and realizes low latency processing. The proposed method was presented at DEIM 2012 and it won a best paper award in the forum. In addition, it was published in an IEICE journal.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
交付決定額	3,100,000	930,000	4,030,000

研究分野：データベース・ストレージ
科研費の分科・細目：情報学、メディア情報学・データベース
キーワード：ストレージ、ファイル I/O、I/O ボトルネック、メニーコア CPU、DBMS、DSMS

１．研究開始当初の背景

研究開発当初、CPU のメニーコア化に伴い CPU1 コアあたりの性能向上が緩やかになってきていた。その一方で、SSD の登場によりストレージ性能は向上していた。この傾向は現在も継続しており、今後は、OS がファイル I/O 処理に使うシステム時間の割合が増加し、CPU 1 コアあたりの I/O 処理負担が増える。従って、超高速なストレージ環境におけるファイル I/O のボトルネックは、ストレージそのものではなく、OS の I/O 処理になると考えられた。

多くの OS では、システムコールを呼び出したプロセス上でカーネル内処理が動作し、ファイル I/O に関する OS レベルの処理が並列化されない。そのため、メニーコア CPU の性能を生かすことができない。そこで本研究では、これまで考慮されなかった超高速ストレージ環境での CPU ボトルネックの存在を明示すると共に、I/O 処理を並列化する手法を開発することを目指した。

２．研究の目的

本研究の目的は、これまで考慮されなかった超高速ストレージ環境での CPU ボトルネックの存在を示すと共に、CPU ボトルネックを軽減し、DBMS や DSMS といったアプリケーションを対象に性能向上方法を提案することである。

超高速ストレージを持つ近未来の計算機におけるファイル I/O では、ストレージではなく CPU がボトルネックになると考えられた。ストレージのスループットが向上すると I/O 処理に必要なシステム時間が増加するが、現在一般的な OS では、1 度のシステムコール呼び出し内の I/O 処理を並列化しておらず、メニーコア CPU の特性を生かせない。本研究は、これまでストレージがボトルネックになると考えられてきた「ファイル I/O」において、将来的には CPU がボトルネックになると仮定する研究である。

３．研究の方法

本研究の実験を行うために、市販の SSD と RAID カードを複数用いて、可能な限り高速なストレージ環境を構築した。構築した環境において、Linux を用いて研究を行った。研究体制としては、OS・アーキテクチャ・並列処理の専門家である早稲田大学の山名教授から助言を受け、同教授の修士学生からも協力を得た。修士学生に手法評価の一部補助を依頼した。

具体的な手順として、まず、(1) CPU ボトルネックになる環境・状況の明確化を行った。ここでは、どのような環境・クエリにおいて CPU ボトルネックが発生するか明確化した。次に、(2) DBMS の処理で重要な Sort と Join オペレータを対象として、CPU ボトルネックが発生し、既存のクエリ最適化技法の改良が必要なことを示した。最後に、(3) データストリーム処理を対象として低レイテンシ並列実行手法の開発を行った。

４．研究成果

本研究の成果には、関係データベースにおいて重要な演算である Sort や Join について、超高速 SSD 環境における新たな実行方法が必要なことを示唆した点がある。研究成果は DEIM2013 において発表した。また、データストリームの超低レイテンシ並列処理手法も提案した。処理に必要な通信回数を削減することで超低レイテンシ処理を実現する。提案内容を DEIM 2012 において発表したところ最優秀論文賞を受賞し、高い評価を受けた。また、電子情報通信学会論文誌にも採録されている。

(1) CPU ボトルネックの提示

まず、超高速環境におけるファイル I/O において CPU がボトルネックになることを示すために、Linux OS を対象として、高性能ストレージ環境下におけるアクセス速度の測定と、その際の CPU 負荷を計測した。スループットとカーネル内 CPU 使用率の結果を、図 1 に示す。

この処理では、256GB のファイルに対し 2GB のブロックサイズで、Direct I/O によるシーケンシャルな読み込みを行っている。Direct I/O とは、Linux カーネルが管理するページキャッシュ（ファイルキャッシュ）を経由した I/O とは異なり、ストレージからユーザ空間に直接データを転送する方式である。DBMS などのアプリケーションでは、独自のページキャッシュ機構を持つため、通常のページキャッシュを利用した I/O では、キャッシュの処理が二重になり、無駄な遅延が発生することになる。そこで Direct I/O のような機構が利用されている。また OLAP においては、データスキャン型のクエリが多いられるため、ファイルキャッシュが有効に機能しない場合も多く、Direct I/O によってファイル I/O スループットを向上することができる。

実験結果の図 1 より Direct I/O を利用した読み込み処理におけるスループットは毎秒 5,000～5,500MB であり、その際のカーネルの CPU 使用率（%system）は 70～80% ほどであったことがわかる。実際のカーネルの処理では、システムコールの read に起因する処理が大半を占めていた。以上のことから、超高速ストレージ環境においては、ファイル I/O によって CPU ボトルネックが発生することが確認された。この事実は、ストレージがボトルネックになるというこれまでの考え方とは異なる結果である。

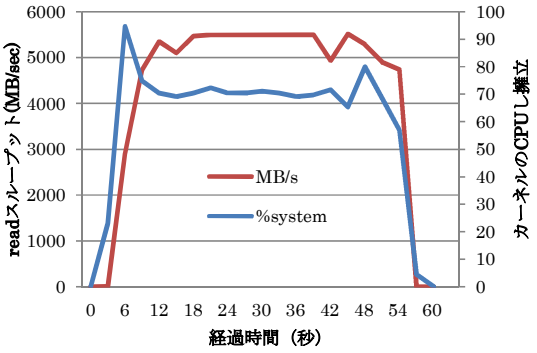

図 1 Direct I/O での read スループットとカーネルの CPU 使用率

(2) DBMS の処理を対象とした研究

次に、ファイル I/O の方法によって、DBMS にどのような性能差異が生じるか明らかにすることを目指した。DBMS のベンチマークに一般的に利用されている TPC-H クエリを、独自に実装した関係代数オペレータを用いて、

超高速ストレージ環境で実験を行った。

実験クエリとして TPC-H の LINEITEM 表に対する演算である、クエリ 1 を利用した。スケールファクタ（SF）を 10 から 100 まで変化させた。ファイル I/O は Linux OS のシステムコールを用いて実装した。システムコール毎に、クエリ処理性能を測定した。比較に利用したシステムコールは以下のとおりである。なお全てのアクセスは Direct I/O で行う。

- read()による同期 I/O 処理
- Linux カーネルの非同期 I/O（aioread）
- pread による、ファイルオフセットを指定したマルチスレッド（pthread）による処理

いずれもアクセスのブロックサイズは 2GB である。また pread による処理では、スレッドを 2,4,8,16 の 4 パターンで変更して、計測を行った。実験結果を図 2 に示す。

システムコールの read()は，1 スレッドの処理に関わらず、pread による処理よりも高速なケース（SF = 20, 30）が存在した。1 スレッドによる非同期 I/O の aioread については、クエリ処理のプロセス本体とは別に I/O 処理を行うが、今回のケースでは I/O スループットが、元々read()に比べ低いため性能が低下したと考えられる。また、マルチスレッド下での pread での処理は、read()ほど先読みの効果が働かず、スレッド同期のオーバーヘッドが大きいことも影響して、性能が劣化する場合があると考えられる。メモリなどのリソースの観点からも read による処理が効率的であると考えられる。

以上から、現在の Linux OS の実装上では、システムコールの read を用いて、非同期 I/O な処理をするスレッドと I/O 以外の DBMS クエリの処理を行うスレッドによる処理が効率的であると確認できた。

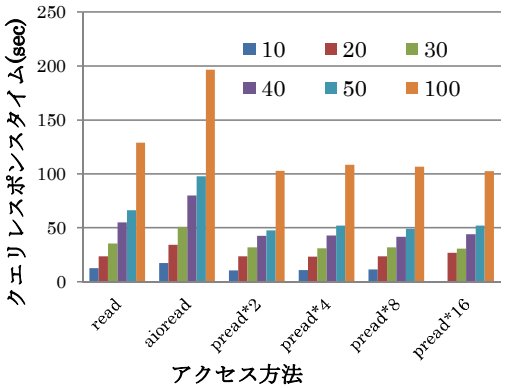

図 2. システムコールとクエリレスポンスタイムの関係

そこで、DBMS クエリ中のアルゴリズムとして、ソートマージ Join を対象にファイル I/O の高速化を検討した。本実験ではアルゴリズム中の各 I/O の性能を測定し、クエリ最適化に用いられる既存のアクセスコストの妥当性を検証することが目的である。既存のクエリ最適化は一般にストレージアクセスがボトルネックと仮定して、ファイル I/O を最小化するものであるから、CPU がボトルネックになる場合が存在すれば、クエリモデルの改良が必要となる。

ソートマージ Join の I/O に関する処理は、全部で 5 つに大別できる。

① 全データを含むファイルからメモリへの read（非同期）
② メモリ上の sort 後、サブリストへの write（同期）
③ メモリで mergesort を行う際、各サブリストから行う read（同期）
④ メモリ上の先頭データの write（同期）
⑤ 結合処理を行う際の read（非同期）

実験データには TPC-H の LINEITEM 表と ORDER 表（それぞれスケールファクタ 10）を用いて、l_orderkey と o_orderkey を結合キーとする。また、I/O バッファ：2GB（同期 I/O)、2GB×4（非同期 I/O）とする。以上の I/O 処理について測定した実験結果を表 1 に示す。

実験の結果、I/O 処理の①、⑤においては Linux の先読み機構が充分に機能して CPU バウンドな処理となっていた。残りの②、③、④は I/O バウンドな処理となっていることが確認できた。したがって DBMS の Sort や Join のアルゴリズムにおいても、将来的な高速ストレージ環境においては、従来のストレージへのアクセスコストのみを想定したコストモデルでは不十分であると考えられ、今後は他のオペレータを含め、I/O 処理に占める CPU 時間の増加の影響を考慮したコストモデルの検討が必要であることが分かった。

以上の研究は DEIM 2013 において発表されている。

表 1：アルゴリズム中の I/O 性能

I/O 処理	スループット MB/s（最高値）
①	4000~5000
②	2000
③	100
④	≒0
⑤	4000~5000

(3) データストリームを対象とした研究

次に、データストリーム処理の並列化検討を行った。上記の結果から、データストリーム処理にファイル I/O オペレータが存在した時、ファイル I/O のみでも CPU 時間を大量に消費することが分かる。そのため、ボトルネックを解消するためには、クエリをパイプライン化し、処理の並列化を行う必要がある。しかし、並列数を増やしすぎては、スレッド間通信に伴って、データストリーム処理のレイテンシが延びてしまう。本研究では、処理あふれを起こさない範囲でレイテンシを削減するための CPU コア割り当て手法を提案した。

研究結果では、CPU アーキテクチャやスレッド起床のオーバーヘッドを考慮し、マルチコア環境における処理レイテンシの発生原因について検討した。コア間通信によって発生するレイテンシを削減するために、可能な限り少数のコアで処理を実行し、通信経路を制御すべきであることが分かった。そして、データストリーム処理のレイテンシ定義を与え、動的計画法によりモデル上の最適解が求まることを示した。さらに、ストリームの入力データレート変化に応じてオペレータを再配置する際、ストリーム処理を停止せずに、オペレータへのタプル適用順を守ってオペレータを再配置する方法を提案した。

実験結果を図 3 に示した。10 秒までは定常状態にないため省略している。図中の Static はスレッド割り当てを実験中に変化させなかった場合であり、データレートが変化するとレイテンシ性能が悪化していることが分かる。提案手法はデータレートが変化するとオペレータへの CPU コア割り当てを変更するため、レイテンシを削減できている。10 秒から 15 秒、20 秒から 25 秒、30 秒から 35 秒の間である。提案手法は入力レートが変化してからオペレータを再配置するため、レイテンシ性能が一時的に悪化する。これは、図 3 においてもオペレータ再配置時の性能変化から確認できる。しかし、再配置を行った場合でも、再配置しない場合のレイテンシを超えていないことから、再配置手法が有効に機能し、再配置のオーバーヘッドが少ないことを確認できる。なお、10 秒において提案手法のレイテンシ性能が特に悪いのは、初回のオペレータ再配置なため、JIT コンパイラの影響によりレイテンシが悪化していると考えられる。

以上の提案内容を DEIM 2012 において発表したところ最優秀論文賞を受賞し、高い評価を受けた。また、電子情報通信学会論文誌にも採録されている。

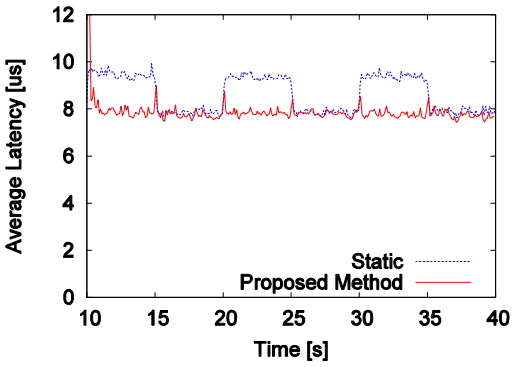

図 3 平均レイテンシの変化（10 秒～15 秒、20 秒～25 秒、30 秒～35 秒はパターンが異なる）

５．主な発表論文等

〔雑誌論文〕（計 2 件）

- **上田高徳**, 秋岡明香, 山名早人,「マルチコア CPU 環境における低レイテンシデータストリーム処理」, 電子情報通信学会論文誌 D, vol.96, no.5, May 2013.
- **上田高徳**, 佐藤亘, 鈴木大地, 打田研二, 森本浩介, 秋岡明香, 山名早人,「Producer-Consumer 型モジュールで構成された並列分散 Web クローラの開発」, 情報処理学会論文誌 Vol.6, No.2, pp.85-97, Mar. 2013.

〔学会発表〕（計 1 件）

- 鈴木大地, **上田高徳**, 山名 早人,「高速ストレージ環境における DBMS クエリの I/O 並列化に関する検討」, 第 5 回データ工学と情報マネジメントに関するフォーラム (DEIM), Mar. 2013.

６．研究組織

(1)研究代表者

上田　高徳　（早稲田大学）

研究者番号：90546863