ウェアラブルコンピューティングのための追記型ファイルシステムの実装

An Implementation of an Append-only File System for Wearable Computers

宮崎 純◇ 小野田 英樹♡
波多野 賢治◇ 植村 俊亮◇

Jun MIYAZAKI Hideki ONODA
Kenji HATANO Shunsuke UEMURA

ウェアラブルコンピューティング環境では，小型コンピュータ上で能動型データベースを利用することにより，コンピュータが個人アシスタントのように動作することが望ましい．我々は，これまで能動型 DBMS を小型ディスクドライブ上に構築することを検討してきた．能動型 DBMS は自動的にデータベースアクセスを行なうため，高速なディスクアクセスが要求される．しかし，デスクトップ用ディスクドライブに比べ，小型ディスクドライブはシーク時間および回転速度が著しく劣るためデータアクセス速度に問題がある．本稿では，小型ディスク上で高速なデータアクセスを実現するために，ログを利用したファイルシステムを提案し，従来のページ単位アクセスによるファイルシステムとアクセス性能を比較することで提案手法の有効性を示す．

Users desire their electronic assistances with wearable computers. In order to recommend information to users automatically, use of an active database is effective. Since an active database requires a large number of disk accesses, the system has to be able to access data on a tiny disk drive at high speed. However, the data access speed of tiny disks are slow because their seek time and rotational latency are inferior to those of desktop models. We make use of a log-structured file system for the sequential access that decreases the latency. In this letter, we compare the proposed data access method with a conventional page-based one, so as to show the good efficiency of our method.

1. はじめに

近年，小型のノートパソコンや PDA(Personal Digital Assistant)，携帯電話 (以下，これらをモバイル機器と呼ぶ) などの登場により，日常的にモバイル機器を携帯することが可能となってきている．モバイル機器は小型化，高機能化が進んでおり，将来的には GPS(Global Positioning System) やカメラを内蔵したウェアラブルコンピュータとして利用されると予想される．モバイル機器が，利用者のアシスタントとして動作する事への要求が高まりつつあるため，利用者が求めている情報を，状況に応じてモバイル機器が判断し，能動的に情報を提示する必要がある．モバイル機器が利用者をアシストする場面は，日常生活の場から災害救助現場など幅広く求められており，ネットワークが利用できない環境においてはモバイル機器単体で動作することが求められる．機器単体で動作するためには，必要なデータをすべて蓄えることが可能な大容量なストレージが必要であり，低価格の小型ディスクドライブを利用することは適当である．

利用者が必要とする情報を能動的に提示するには，あらかじめ登録したルールに基づき動作し，ルールにより提示する情報のカスタマイズが可能な能動型データベースの利用が有効である．しかし，能動型データベースは通常のデータベースに比べ，イベント処理，コンディションの評価およびアクションがシステムにより自動的に実行されるため，データの参照，イベントの監視など，ディスクアクセスが頻繁に発生し処理コストが高くなる．ところが，小型ディスクドライブはサーバ用ハードディスクドライブと比べシーク時間，回転速度が劣るため，データアクセス速度は遅い．そのため，迅速な情報の提供には小型ディスクドライブの特性に見合ったファイルシステムが必要となる．

本稿では，ウェアラブルコンピュータ向けの高速なデータベースを実現するために，ログファイルシステム [6] を利用し，すべてのデータの更新をログとしてシーケンシャルに追記するファイルシステムを採用する．更新はシーケンシャルアクセスのため，ディスク I/O の高速化が図られる一方，データは頻繁に更新され，アクティブなデータ集合が必ずしも連続して存在しているとは限らず，データの検索がシーケンシャルに行なわれるとは限らない．ここでは，小型ディスク内のディスク制御プロセッサとバッファメモリを利用して，検索時のディスクアクセスの最適化を小型ディスク内で処理させることにより，この問題の解決を試みる．以降で，提案するログ追記型ファイルシステムにおけるデータ格納方法，検索処理方法，ならびにディスク I/O の最適化のためのインデクスの利用方法を詳細に述べる．

2. ウェアラブルコンピュータ向けデータベースと小型ディスクドライブ

2.1 データベース

一般にデータベースを利用する時，重要な特性が耐障害性である．データベースの障害によりデータの消失が発生すると，データを有効に利用できないばかりか，正しくない情報を提示してしまう可能性がある．通常，データベースはログを利用してデータ消失の問題を解決している．同時に，ログ書込みはシーケンシャルアクセスであり，データベースの速度向上に貢献している．

ウェアラブルコンピュータでは，上記の耐障害性の必要性はもちろんのこと，過去のデータを時系列に参照したり，テンポラルクエリが実行できることが望ましい．よって，このような用途では，全てのデータを上書きしないファイルシステムが要求される．更に，使用するディスクドライブは，大きさが小型で低消費電力のものが要求される．

2.2 小型ディスクドライブ

小型ディスクドライブ，例えば，日立 GST のマイクロドライブは，構造的には従来のハードディスクドライブと同様ではあるが，回転速度などが十分であるとは言えない (表 1 参照 [7, 3, 4])．

一般に，小型ディスクドライブは消費電力や耐衝撃性などの問題からディスクの回転速度やヘッドのシークを高速にできないため，ディスクアクセスが低速である．従って，ディスクヘッドが目

◇ 正会員 奈良先端科学技術大学院大学情報科学研究科
{miyazaki,hatano,uemura}@is.naist.jp
♡ 学生会員 奈良先端科学技術大学院大学情報科学研究科，現在，(株) 日本ユニシス・ソフトウェア Hideki.Onoda@usk.co.jp

的のレコード位置に移動するアクセス待ち時間 (平均シーク時間+平均回転待ち時間) はマイクロドライブの場合，約 20.33ms とシステム側からみると長時間を要する．一方で，年々記憶密度が上がり，同一回転数でもシーケンシャルアクセス速度は改善されてきている．つまり，小型ディスクドライブを効率よく利用するには，いかにヘッドシークを減少させるかが重要な鍵となる．

表 1: ディスクドライブの性能 [HGST の HP より抜粋]
Table 1: Performance of disk drives [Courtesy of HGST]

	マイクロドライブ		サーバモデル
容量 (GB)	1	4	73.9
密度 (Gb/in^2)	15.3	56.5	31.2
転送レート (Mb/s)	36–60	57.1–97.9	684–969
連続 R/W 速度 (MB/s)	2.5–4.1	4.3–7.2	—
平均シーク時間 (ms)	12	12	3.9
平均待ち時間 (ms)	8.33	8.33	1.99
回転速度 (rpm)	3600	3600	15037
バッファ(KB)	128	128	8192
I/F 転送速度 (Mb/s)	11.1	33	—
消費電力 (mW)	420	360	12000

3. 提案するファイルシステム

半導体の集積技術の発展により，小型ディスクドライブ上に高速な CPU，大容量のメモリを搭載する事は，技術的に問題がなくなってきている．図 1 に示すように，小型ディスク内のディスク制御プロセッサをデータベース管理システムの処理を行うプロセッサとして利用し，小型ディスクドライブ内のバッファを汎用メモリとして利用することは技術的に可能である [8]．データベース管理システムを小型ディスクドライブ上で閉じて処理を実行することが可能となるので，モバイル機器に依存しない高レベルのインタフェースを提供することで，高いポータビリティが実現可能である．

本稿では，上記の前提の下で小型ディスク内の資源，特にメモリを利用した効率の良いデータアクセスを実現するファイルシステムを提案する．

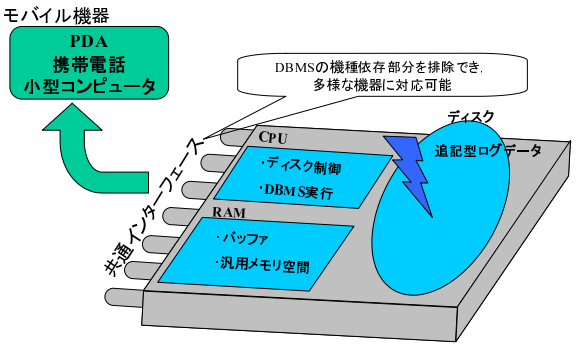

図 1: 小型ディスク上のシステム構成
Fig. 1: System Configuration on a Tiny Disk

3.1 記録方式

小型ディスクドライブで，アクセス待ち時間を短縮するためにログファイルシステム [6] を利用する．提案システムでは，データを追加する場合，ディスクに連続して追記していく手法を採る．このため，ディスクを連続的に利用することができ，データ追加時には，従来のデータベース管理システムで起こるディスクのランダムアクセスが発生しないため，効率の良いデータアクセスが実現できる．データベースは，2.1 節で述べたようにログが保存されていればリカバリが可能であるため，データを上書きせず，ディスクにログのみを追記していく方式はリカバリの観点からも有効である．

また，サイズの小さいデータを頻繁にディスクに書き込む場合，ディスクアクセスのオーバヘッドが高くなるため，シーケンシャルアクセスの利点が十分に得られない可能性がある．このため，まとまったデータを一括して書き込むグループコミット [2] の手法を採用し，効率的な更新処理を実現する．メモリ上にログバッファを設け，更新するデータをログバッファに書き込んでいく．ログバッファがデータで満たされると，ディスクにデータを書き込む．ウェアラブル機器で扱うデータには，位置情報など，たとえデータが消失しても前後のデータから補間が可能なものがある．これらのデータは重要度が低いため，データの消失が発生しても大きな問題は生じない．一方，電子商取引など，重要度の高いデータをメモリ上に蓄えると，障害によりデータが消失する可能性があるため，このような場合はディスクに強制的に書き込みを行う．すなわち，データの重要度 (プライオリティ) に応じてディスクフラッシュの制御を行うことにより効率のよい更新処理が可能となる．

追記型ファイルシステムの利点は，更新処理がシーケンシャルに行なわれるため高速である点である．また，更新によりデータが上書きされないため，ヒストリカルデータを扱う必要があるウェアラブルコンピュータに向いている．その反面，検索処理は必要なデータが連続しているとは限らず，高速にアクセスできるとは限らない．そこで，インデクスを利用した高速な検索方式について次節で議論する．

3.2 インデクス

小型ディスクドライブのメモリ上に，ディスクに格納されているアクティブなレコードの物理位置をエントリに持つインデクスを配置する．インデクスはディスク上のデータを検索する際，高速にアクセスするために利用されるが，インデクスは頻繁に更新されるため，ディスク上に作成するとインデクスの操作自体がランダムアクセスとなり，シーケンシャルアクセスの利点を十分に利用できない．

インデクスのサイズに関して考えると，実際にウェアラブル機器に提案したディスクを装着して利用した場合を想定すれば，例えば位置情報が 10 秒毎に追加されるとしても，高々8640 レコードが 24 時間で蓄えらる．このレコードに対するインデクスは 35KB であり，小型ディスク内のメモリに十分収めることが可能である．それ以外のほとんど更新のないレコードにしても，利用者の個人的に必要なデータであるためインデクスのサイズはそれほど大きくはならないと考えられる．

インデクスの作成方法は 2 種類考えられる．データの変更が生じた場合，変更の対象となるレコード全体を複製し変更後のレコードをディスクに追記する**複製方式**と，データ更新時に更新された属性情報のみを追記する**差分方式**である．図 2 は複製方式のデータの格納とアクティブなデータを指すインデクスの関係を示している．データ操作時の詳細は 3.3 節で示す．一方，差分方式はもともとのレコードへのポインタを保持したまま，更新された属性へのポインタを新たに作成し元レコードへのポインタにリストとして追加する方式であり，元データへのポインタと変更データの格納先へのポインタによりアクティブなレコードが再構成される．

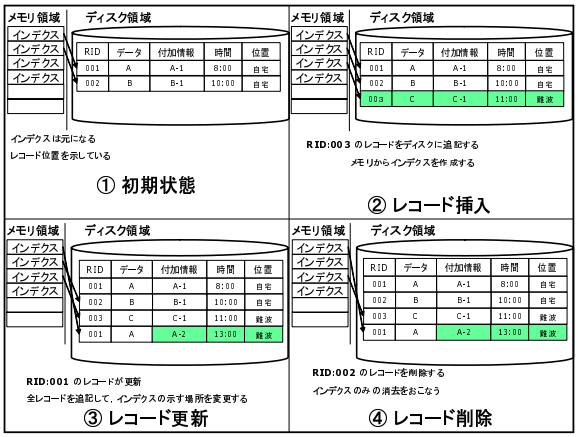

図 2: 複製方式のデータ格納とインデクス
Fig. 2: Data Placement and Index for the Replication Method

3.3 データ操作

本節では，実現が容易な複製方式におけるデータの操作方法を説明する．差分方式も複製方式と同様な方法でデータ操作を行うことができる．

挿入: 新規レコードを挿入する際，新しいレコード ID 発行し，そのレコードに ID を付与して連続データとしてディスクに追記する．データ格納後，このレコードに関するインデクスのエントリを追加する (図 2(2) 参照)．従来のファイルシステムでは，データを挿入する際，格納可能な空き領域を検索するためランダムアクセスが発生するが，提案手法ではデータを格納する追記場所が予め決まっているため，ランダムアクセスは発生しない．そのため，効率よくデータを挿入できる．

更新: インデクスを利用して対象レコードを特定し，そのレコード ID を取得する．取得したレコード ID，複製したレコードをディスクに追記する．更新前のデータをディスク上に残したまま，インデクスは追記されたレコードを指すように変更する (図 2(3) 参照)．

削除: レコードはすべて追記して書かれるため，削除の際はそのレコードが削除された事実をログとして記録する必要がある．そうすれば，万一システムがクラッシュしても全てのデータを一貫した状態に回復できる．最も簡単に削除を実現するためには，削除レコードの ID を追記すれば良い．実際の処理は，削除レコードの ID を追記した後，そのレコードのインデクスエントリを削除すれば良い (図 2(4) 参照)．

検索: 複製方式ではインデクスにアクティブなレコード集合の物理的な格納位置が記録されているため，アクティブなレコード位置を検索前に特定できる．ディスクアクセスを行う前に，インデクスを用いて検索対象レコードの物理的な位置順にソートを行う．インデクスはメモリ上にあるため，このソートはディスクのアクセス時間に比べて短時間で行え，並べ替えによりランダムアクセスがシーケンシャルアクセスに変換される．レコードの物理位置順にソートした後，順にレコードをシーケンシャルにアクセスしていく．しかし，次にアクセスするレコードがシーケンシャルアクセスで読み飛ばすよりもシークした方がより短時間でアクセスできる場合，つまり，平均アクセス待ち時間に相当する時間内にシーケンシャルアクセスできるデータのサイズよりもレコード間隔が離れている場合はシークを行う．このディスクアクセスの最適化により，検索が高速化される．

4. 提案する追記型ファイルシステムの評価

現在，内部リソースを利用できる小型ディスクドライブは存在しないため，小型ディスクドライブとして表 1 の日立のマイクロドライブ (1GB) を用い，Linux の raw デバイス上にファイルシステムの実装を行なった．なお，Linux は gentoo1.4.1(kernel Version 2.6)，PC として IBM ThinkPad (CPU:pentium3，メモリ:384MB) を使用し，ファイルシステムの処理は PC の CPU およびメモリを使用して処理を行わせた．実験に使用したデータは，ウィスコンシンベンチマークで使用される 1 レコード 208 バイトのデータ集合である．

ディスク I/O の最適化のためのパラメタを求めるために予備実験を行なったところ，使用したマイクロドライブは，4KB のページ単位で I/O を行えば最適であり，実測値のアクセス待ち時間は平均 19.84ms であることが分かった．この結果からシーケンシャルアクセスを選択するかランダムアクセスを選択するかの分岐点はレコード間のギャップが約 112KB，もしくはウィスコンシンベンチマークのデータでは約 28 ページ (おおよそ 600 レコード分に相当) の時となる．

提案する追記型ファイルシステムでは，通常，ヘッドはディスクの追記開始トラック上にあるため，挿入処理を実行する場合，平均回転待ち時間で書き込みを行うことが可能である．一方，データの検索時はディスクスキャンが必要となり，提案手法では 3.2 節で示したメモリ上のインデクスを利用することによりディスクアクセスの高速化を図っている．実験では検索性能に焦点を当て，その性能評価を行なう．

4.1 実験内容

十分な数のレコードを用意し，300 回の検索処理の応答時間の測定を行なう．第 n 回目の施行では n 個のレコードを検索する．この時，各レコードの間隔は $\{1, 3, 5, \cdots, 2n+1\}$ を重複なくランダムに選択し，レコード間隔を人工的に与えた．つまり，n が小さいときは，検索されるレコードが密に存在し，シーケンシャルアクセスによる読み飛ばしが利用される．一方，n が大きい時，レコード間隔が密な部分ではシーケンシャルアクセスが選択されるが，疎な部分ではランダムアクセスが選択される．

実際には，予備実験から得られたディスクの特性を反映し，レコード間隔が約 112KB，すなわち 208 バイトのレコードでは約 28 ページ以上のギャップがあればランダムアクセスに切り替わるよう設定した．

4.2 実験結果と考察

実験結果は図 3 の通りである．横軸は検索するレコード数 n を，縦軸はその検索の応答時間を示している．提案手法は実線で示している．比較のため，全てシーケンシャルアクセスの場合 (粗点線)，および全てランダムアクセスの場合 (細点線) も掲載している．

検索レコード数 n が小さい時，すなわちレコード間隔が密の時はシーケンシャルアクセスが選択されている．n が増加するに従い，レコード間隔が疎である部分が増加し，シーケンシャルな読み飛ばしにより応答時間が二次関数的に増加している．レコード間隔が 112KB を過ぎるとシーケンシャルアクセスに比べランダムアクセスによりディスクヘッドを移動させたほうが目的レコードに到達する時間が早くなる．その分岐点からランダムアクセスを利用する方が高速であることがグラフから読み取れる．検索レコード数をさらに増やした場合，28 ページ以上のレコード間隔はランダムアクセスとなる．グラフから明らかなように，提案手法によりディスクアクセスが最適化されていることが分かる．

ウェアラブルコンピュータで小型ディスクドライブを利用するに際し，消費電力の問題を考慮する必要がある．ディスクドライブが電力を消費するのは，ヘッドの移動や回転トルクである．提案手法では，ディスクを可能な限りシーケンシャルにアクセスするため，データアクセスを高速化しつつヘッドの移動回数を減少させている．また，ウェアラブルコンピュータでは，頻繁に GPS

データなどのセンサーデータが追加されるため，ディスクは常に回転しており回転トルクの変動が少ない．従って，消費電力の面においても提案システムは有効であると言える．

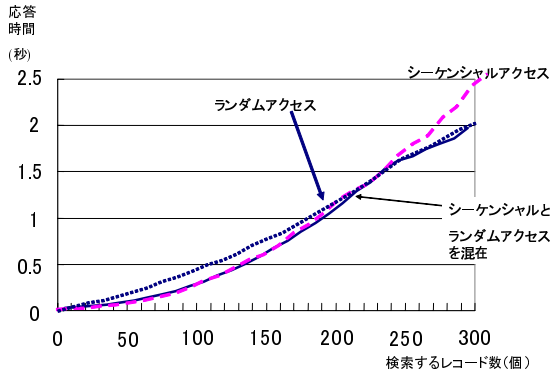

図 3: 検索処理の性能比較
Fig. 3: Comparisons of Query Processing

5. 関連研究

通常のディスクドライブでも，少数のリクエストについてファームウェアレベルでアクセス最適化が行なわれているが [5]，提案手法では大域的なデータ集合についてアクセスの最適化を行なう点で優れている．更に，現在のヘッド位置を判断し，より柔軟なデータアクセスの最適化が可能である．

Rosenblum は，ログ型のファイルシステムを提案している [6]. ファイル削除が実行されると，ストレージに未使用ブロックが発生し，結局ガベージコレクションにより未使用ブロックを回収する．我々の提案は，ウェアラブルコンピュータ向けの個人用ファイルシステムであり，過去の記録を参照したり，テンポラルクエリを実行することを考慮して，削除は論理的であり，削除されたデータは実際には消去されない点で異なる．

Bobineau らは 1MB を超える非接触型スマートカード上で効率よく問合わせを処理する PicoDBMS[1] を提案している．しかし，我々のファイルシステムはディスクを前提としており，メモリを前提とする PicoDBMS のリング型ストレージモデルとは異なる．

石墨らはフラッシュメモリ上でのファイルシステムの構築手法について提案している [9]. フラッシュメモリは記録済み部分への直接の上書きができず，上書き時は一度メモリブロックごとデータを削除する必要がある．この削除時間は比較的長時間要するだけでなく，ブロック消去回数にも制限がある．この問題を解決するためにリンクリストを利用したファイルシステムを提案し，特定のブロックに消去が集中することを回避している．我々の提案手法は，インデクスの格納に RAM を使用できれば，データ本体は消去しないため，フラッシュメモリにも適用可能である．

倉光らは，非接触型スマートカード上に小型の DBMS を構築する方法について論じている [10]. 非接触型スマートカードは記憶容量が小さいため，提案手法のように大量のデータアクセスについては考慮されていない．

横田らはディスク装置のプロセッサやメモリを利用することでストレージ機器内で負荷分散，障害対策，障害回復を実現する手法を提案しており [8]，我々のアプローチと類似する．しかし我々は，ファイルシステムへのアクセス制御に特化してストレージ上の資源を利用している．

6. おわりに

本稿では，小型ディスクドライブに適した追記型ファイルシステムを提案し，その評価を行った．データの追加の際はシーケンシャルアクセスとなり，高速にアクセスが行える．更に，検索の際もディスク内のメモリにアクティブなレコードの物理位置を持つインデクスを格納することにより，位置情報を検索実行時にソートして，ディスクアクセスを最適化できることを明らかにした．この最適化は，消費電力の大きいヘッドの移動を減らせるため，低消費電力にも貢献する．

データ複製方式ついては検証を行ったが，今後，差分方式の検証も行う必要がある．また，データの挿入と検索が並行して起こる場合も考慮する必要があり，これらの最適化や検証を行っていく予定である．

[謝辞]

本研究の一部は，科学技術振興機構戦略的創造研究推進事業 (CREST)「高度メディア社会の生活情報技術」，科学研究費補助金 (課題番号: 15300029，15700090)，ならびに情報ストレージ推進機構 (SRC) の支援による．ここに記して謝意を表す．

[文献]

[1] C. Bobineau, L. Bouganim, P. Pucheral, and P. Valduriez. PicoDBMS: Scaling down Database Techniques for the Smartcard. In *Proc. of International Conference on Very Large Databases*, pp. 11–20, 2000.

[2] P. M. Chen, W. T. Ng, S. Chandra, C. Aycock, G. Rajamani, and D. Lowell. The Rio File Cache: Surviving Operating System Crashes. In *Proc. of 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 74–83, 1996.

[3] HITACHI マイクロドライブ製品仕様 3K4 シリーズ.

[4] HITACHI マイクロドライブ製品仕様 DSCM シリーズ.

[5] Serial ATA Native Command Queuing – An Exciting New Performance Feature for Serial ATA –. A Join Whitepaper by Intel and Seagate, July 2003.

[6] M. Rosenblum. *The Design and Implementation of a Log-Structured File System*. Kluwer Academic, 1995.

[7] HITACHI サーバ向ハードディスクドライブ製品仕様 Ultrastar 15K73 シリーズ.

[8] H. Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments*, pp. 441–448, November 1999.

[9] 石墨紀孝, 最所圭三, 福田晃. 組み込みシステム向けフラッシュメモリファイルシステムの設計 s. 電子情報通信学会論文誌, Vol. J84-D-I, No. 1, pp. 90–99, 2001.

[10] 倉光君朗, 坂村健. 非接触型スマートカード上のユビキタスデータベース. 情報処理学会論文誌：データベース, Vol. 43, No. SIG5(TOD14), pp. 110–117, 2002.

宮崎 純　Jun MIYAZAKI
奈良先端科学技術大学院大学情報科学研究科助教授．データベースシステムの研究に従事．情報処理学会，電子情報通信学会，日本データベース学会，ACM SIGMOD，IEEE CS 各会員.

小野田 英樹　Hideki ONODA
奈良先端科学技術大学院大学情報科学研究科博士前期課程修了，現在，日本ユニシス・ソフトウェア株式会社に勤務．在学中はウェアラブルデータベースの研究に従事.

波多野 賢治　Kenji HATANO
奈良先端科学技術大学院大学情報科学研究科助手．情報検索システムの研究に従事．情報処理学会，電子情報通信学会，日本データベース学会，ACM，IEEE CS，各会員.

植村 俊亮　Shunsuke UEMURA
奈良先端科学技術大学院大学情報科学研究科教授．データベースシステムの研究に従事．情報処理学会，電子情報通信学会および IEEE フェロー.