(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5385977号
(P5385977)

(45)発行日　平成26年1月8日(2014.1.8)　(24)登録日　平成25年10月11日(2013.10.11)

(51)Int.Cl.　　F I
G06F 9/48 (2006.01)　　G06F 9/46 452H
G06F 13/10 (2006.01)　　G06F 13/10 340B

請求項の数 34　(全 31 頁)

(21)出願番号　特願2011-511688 (P2011-511688)
(86)(22)出願日　平成21年5月8日(2009.5.8)
(65)公表番号　特表2011-525010 (P2011-525010A)
(43)公表日　平成23年9月8日(2011.9.8)
(86)国際出願番号　PCT/US2009/043387
(87)国際公開番号　WO2009/148764
(87)国際公開日　平成21年12月10日(2009.12.10)
審査請求日　平成23年1月31日(2011.1.31)
(31)優先権主張番号　12/130,892
(32)優先日　平成20年5月30日(2008.5.30)
(33)優先権主張国　米国(US)

(73)特許権者　500551079
ソニー　コンピュータ　エンタテインメント　アメリカ　リミテッド　ライアビリティ　カンパニー
アメリカ合衆国、カリフォルニア州　９４４０４－２１７５、フォスター・シティー、セカンド・フロアー、イースト・ヒルスデイル・ブルバード　９１９
(74)代理人　100105924
弁理士　森下　賢樹
(72)発明者　ターラー、アンドリュー、アール.
アメリカ合衆国、カリフォルニア州　９４４０４、フォスター・シティー、セカンド・フロアー、イースト・ヒルスデイル・ブルバード　９１９

最終頁に続く

(54)【発明の名称】ファイル入出力スケジューラ

(57)【特許請求の範囲】
【請求項１】
プロセッサユニット、メモリ、およびメディアデバイスをもつシステムにおいて、前記プロセッサユニットは、メインプロセッサと、関連づけられたローカルメモリをもつ補助プロセッサとを含み、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法であって、
ａ）メディアデバイスに対してデータを転送するために、プロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取るステップと、
ｂ）前記入力Ｉ／Ｏリクエストを完了するための予測時間T_pを計算するステップと、
ｃ）前記入力Ｉ／Ｏリクエストのスケジュール内の位置が予測時間T_pに少なくとも部分的に依存するように、前記入力Ｉ／Ｏリクエストをメモリ内に具体化されたスケジュールに挿入するステップと、
ｄ）メディアデバイスからローカルメモリにデータを転送することによって、前記スケジュールにしたがって前記入力Ｉ／Ｏリクエストをサービスし、補助プロセッサを用いてローカルメモリのデータを変換するステップとを含み、
前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップは、
複数の異なるＩ／Ｏリクエストをサービスするための合計時間Ｔを決定するステップと、
前記複数のＩ／Ｏリクエストの２以上の異なる順序に対して前記合計時間Ｔの２以上の異なる値を計算するステップと、

前記合計時間の最適値にもとづいて前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップとを含み、
ステップｄ）は、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカンダリデータユニットで置き換えられることを特徴とする方法。
【請求項２】
ステップｂ）は、メディアデバイスのパラメータと前記Ｉ／Ｏリクエストから決定される係数を用いた性能特性式（ＰＣＥ）にしたがって予測時刻T_pを計算するステップを含む請求項１の方法。
【請求項３】
前記パラメータは、リクエストオーバーヘッドｘ、ヘッド運動レートｙ、およびスループットｚを含む請求項２の方法。
【請求項４】
前記係数は、オーバーヘッド係数Ａ、ヘッド移動距離Ｂ、および転送データ量Ｃを含む請求項３の方法。
【請求項５】
前記性能特性式は、$T_p = Ax + By + Cz$の形である請求項４の方法。
【請求項６】
ステップｃ）は、前記スケジュール内の前記複数の異なるＩ／Ｏリクエストに対する異なる係数をもつ前記性能特性式を繰り返すことで前記複数のＩ／Ｏリクエストをサービスするための前記合計時間Ｔを決定するステップを含む請求項２の方法。
【請求項７】
前記合計時間Ｔの最適値の値は、前記複数のＩ／Ｏリクエストの２以上の異なる順序に対する合計時間Ｔの最小値である請求項１の方法。
【請求項８】
ステップｂ）は、Ｉ／Ｏリクエストに対してかかった現実の時間Ｔａを測定するステップと、前記現実の時間Ｔａを前記係数とともに用いて、未知パラメータの値に対してＰＣＥを反復して解くステップとを含む請求項２の方法。
【請求項９】
ステップｄ）は、圧縮されたネストしたアーカイブから特定のデータを抽出するステップをさらに含む請求項１の方法。
【請求項１０】
前記システムはさらにグラフィックスプロセッサと関連づけられたグラフィックスメモリとを含み、ステップｄ）は、メディアデバイスからグラフィックスメモリにデータを読み込むステップと、グラフィックスメモリからローカルメモリにデータを転送するステップと、補助プロセッサを用いてデータ上でデータ変換を実行して、変換されたデータを生成するステップと、変換されたデータをグラフィックスメモリに返送するステップとを含む請求項１の方法。
【請求項１１】
前記プライマリデータユニットのソースと前記セカンダリデータソースは同じメディア上にある請求項１０の方法。
【請求項１２】
前記プライマリデータユニットのソースと前記セカンダリデータソースは異なるメディア上にある請求項１０の方法。
【請求項１３】
前記セカンダリデータソースは仮想的なデータソースである請求項１０の方法。
【請求項１４】
セカンダリデータユニットはコールバックによって指定される請求項１３の方法。
【請求項１５】
前記システムは高速ストレージデバイスをさらに含み、ステップｄ）は、メディアデバ

イスから高速ストレージデバイスへ１以上のデータユニットをプリフェッチするステップと、それに続いて高速ストレージデバイスからそのデータユニットを転送するステップとを含む請求項１の方法。
【請求項１６】
高速ストレージデバイスはハードディスクドライブである請求項１５の方法。
【請求項１７】
メディアデバイスは大容量の光ディスクドライブである請求項１６の方法。
【請求項１８】
１以上のデータユニットをプリフェッチするステップは、Ｉ／Ｏがアイドルであるとき、ある時間間隔の間、１以上のデータユニットをプリフェッチする請求項１５の方法。
【請求項１９】
１以上のデータユニットをプリフェッチするステップは、メディアデバイスから１以上のデータユニットの第１グループをプリフェッチするステップと、メディアデバイスがアイドルかどうかを調べるステップと、もしＩ／Ｏがアイドルなら、メディアデバイスから１以上のデータユニットの第２グループをプリフェッチし、もしＩ／Ｏがアイドルでないなら、前記第２グループのプリフェッチを遅延させるステップとを含む請求項１８の方法。
【請求項２０】
ステップｂ）は、前記システムに関連づけられたメディアデバイスよりも低速のメディアのオペレーションをエミュレートするために前記メディアデバイスのパラメータを調整するステップを含む請求項１の方法。
【請求項２１】
プロセッサユニットと、
前記プロセッサユニットと結合したメモリと、
前記プロセッサユニットと結合したメディアデバイスと、
前記メモリに具体化されたプロセッサ実行可能な命令セットとを含むシステムであって、
前記命令は実行されたときに、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法を実行するように構成され、前記方法は、
ａ）メディアデバイスに対してデータを転送するために、プロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取るステップと、
ｂ）前記入力Ｉ／Ｏリクエストを完了するための予測時間T_pを計算するステップと、
ｃ）前記入力Ｉ／Ｏリクエストのスケジュール内の位置が予測時間T_pに少なくとも部分的に依存するように、前記入力Ｉ／Ｏリクエストをメモリ内に具体化されたスケジュールに挿入するステップと、
ｄ）前記スケジュールにしたがって前記入力Ｉ／Ｏリクエストをサービスするステップとを含み、
前記プロセッサ実行可能な命令セットは、デアーカイバレイヤ、ＲＡＭキャッシュレイヤ、スケジューラキャッシュレイヤ、オーバレイレイヤ、カタログキャッシュレイヤ、データ変換レイヤ、またはシミュレートされたデータレイヤを含む１以上のメディアフィルタレイヤをさらに含み、
前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップは、
複数の異なるＩ／Ｏリクエストをサービスするための合計時間Ｔを決定するステップと、
前記複数のＩ／Ｏリクエストの２以上の異なる順序に対して前記合計時間Ｔの２以上の異なる値を計算するステップと、
前記合計時間の最適値にもとづいて前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップとを含み、
ステップｄ）は、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカン

ダリデータユニットで置き換えられることを特徴とするシステム。
【請求項２２】
メインプロセッサと、関連づけられたローカルメモリをもつ補助プロセッサと、メインメモリと、メディアデバイスとを含むシステムにおいて、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法であって、
ａ）メディアデバイスに対してデータを転送するために、メインプロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取るステップと、
ｂ）前記入力Ｉ／Ｏリクエストをメインメモリ内に具体化されたスケジュールに挿入するステップと、
ｃ）スケジュールと１以上のフィルタにしたがって前記入力Ｉ／Ｏリクエストをサービスするステップとを含み、１以上のフィルタの少なくとも１つは補助プロセッサによって実行され、
前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップは、
複数の異なるＩ／Ｏリクエストをサービスするための合計時間Ｔを決定するステップと、
前記複数のＩ／Ｏリクエストの２以上の異なる順序に対して前記合計時間Ｔの２以上の異なる値を計算するステップと、
前記合計時間の最適値にもとづいて前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップとを含み、
ステップｄ）は、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカンダリデータユニットで置き換えられることを特徴とする方法。
【請求項２３】
プロセッサユニット、メモリ、およびメディアデバイスをもつシステムにおいて、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法であって、
ａ）メディアデバイスに対してデータを転送するために、プロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取るステップと、
ｂ）前記入力Ｉ／Ｏリクエストを完了するための予測時間$Ｔ_{p}$を計算するステップと、
ｃ）前記入力Ｉ／Ｏリクエストのスケジュール内の位置が予測時間$Ｔ_{p}$に少なくとも部分的に依存するように、前記入力Ｉ／Ｏリクエストをメモリ内に具体化されたスケジュールに挿入するステップと、
ｄ）前記スケジュールにしたがって前記入力Ｉ／Ｏリクエストを実行するステップとを含み、ステップｄ）は、圧縮されたネストしたアーカイブから特定のデータを抽出するステップを含み、
前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップは、
複数の異なるＩ／Ｏリクエストをサービスするための合計時間Ｔを決定するステップと、
前記複数のＩ／Ｏリクエストの２以上の異なる順序に対して前記合計時間Ｔの２以上の異なる値を計算するステップと、
前記合計時間の最適値にもとづいて前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップとを含み、
ステップｄ）は、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカンダリデータユニットで置き換えられることを特徴とする方法。
【請求項２４】
プロセッサユニット、メモリ、およびメディアデバイスをもつシステムにおいて、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法であって、
ａ）メディアデバイスに対してデータを転送するために、プロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取るステップと、
ｂ）前記入力Ｉ／Ｏリクエストを完了するための予測時間$Ｔ_{p}$を計算するステップと、

ｃ）前記入力Ｉ／Ｏリクエストのスケジュール内の位置が予測時間T_pに少なくとも部分的に依存するように、前記入力Ｉ／Ｏリクエストをメモリ内に具体化されたスケジュールに挿入するステップと、
ｄ）前記スケジュールにしたがって前記入力Ｉ／Ｏリクエストを実行するステップとを含み、ステップｄ）は、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカンダリデータユニットで置き換えられ、
前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップは、
複数の異なるＩ／Ｏリクエストをサービスするための合計時間Ｔを決定するステップと、
前記複数のＩ／Ｏリクエストの２以上の異なる順序に対して前記合計時間Ｔの２以上の異なる値を計算するステップと、
前記合計時間の最適値にもとづいて前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップとを含み、
ステップｄ）は、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカンダリデータユニットで置き換えられることを特徴とする方法。
【請求項２５】
前記プライマリデータユニットのソースと前記セカンダリデータソースは同じメディア上にある請求項２４の方法。
【請求項２６】
前記プライマリデータユニットのソースと前記セカンダリデータソースは異なるメディア上にある請求項２４の方法。
【請求項２７】
前記セカンダリデータソースは仮想的なデータソースである請求項２４の方法。
【請求項２８】
セカンダリデータユニットはコールバックによって指定される請求項２７の方法。
【請求項２９】
プロセッサユニット、メモリ、高速ストレージデバイス、およびメディアデバイスをもつシステムにおいて、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法であって、
ａ）メディアデバイスに対してデータを転送するために、プロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取るステップと、
ｂ）前記入力Ｉ／Ｏリクエストを完了するための予測時間T_pを計算するステップと、
ｃ）前記入力Ｉ／Ｏリクエストのスケジュール内の位置が予測時間T_pに少なくとも部分的に依存するように、前記入力Ｉ／Ｏリクエストをメモリ内に具体化されたスケジュールに挿入するステップと、
ｄ）前記スケジュールにしたがって前記入力Ｉ／Ｏリクエストを実行するステップとを含み、ステップｄ）は、メディアデバイスから高速ストレージデバイスへ１以上のデータユニットをプリフェッチするステップと、それに続いて高速ストレージデバイスからそのデータユニットを転送するステップとを含み、
前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップは、
複数の異なるＩ／Ｏリクエストをサービスするための合計時間Ｔを決定するステップと、
前記複数のＩ／Ｏリクエストの２以上の異なる順序に対して前記合計時間Ｔの２以上の異なる値を計算するステップと、
前記合計時間の最適値にもとづいて前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップとを含み、
ステップｄ）は、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカン

ダリデータユニットで置き換えられることを特徴とする方法。
【請求項３０】
高速ストレージデバイスはハードディスクドライブである請求項２９の方法。
【請求項３１】
メディアデバイスは大容量の光ディスクドライブである請求項３０の方法。
【請求項３２】
１以上のデータユニットをプリフェッチするステップは、Ｉ／Ｏがアイドルであるとき、ある時間間隔の間、１以上のデータユニットをプリフェッチする請求項３１の方法。
【請求項３３】
１以上のデータユニットをプリフェッチするステップは、メディアデバイスから１以上のデータユニットの第１グループをプリフェッチするステップと、メディアデバイスがアイドルかどうかを調べるステップと、もしＩ／Ｏがアイドルなら、メディアデバイスから１以上のデータユニットの第２グループをプリフェッチし、もしＩ／Ｏがアイドルでないなら、前記第２グループのプリフェッチを遅延させるステップとを含む請求項３１の方法。
【請求項３４】
プロセッサユニット、メモリ、高速ストレージデバイス、およびメディアデバイスをもつシステムにおいて、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法であって、
ａ）メディアデバイスに対してデータを転送するために、プロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取るステップと、
ｂ）前記入力Ｉ／Ｏリクエストを完了するための予測時間T_pを計算するステップと、
ｃ）前記入力Ｉ／Ｏリクエストのスケジュール内の位置が予測時間T_pに少なくとも部分的に依存するように、前記入力Ｉ／Ｏリクエストをメモリ内に具体化されたスケジュールに挿入するステップと、
ｄ）前記スケジュールにしたがって前記入力Ｉ／Ｏリクエストを実行するステップとを含み、ステップｂ）は、前記システムに関連づけられたメディアデバイスよりも低速のメディアのオペレーションをエミュレートするために前記メディアデバイスのパラメータを調整するステップを含み、
前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップは、
複数の異なるＩ／Ｏリクエストをサービスするための合計時間Ｔを決定するステップと、
前記複数のＩ／Ｏリクエストの２以上の異なる順序に対して前記合計時間Ｔの２以上の異なる値を計算するステップと、
前記合計時間の最適値にもとづいて前記入力Ｉ／Ｏリクエストを前記スケジュールに挿入するステップとを含み、
ステップｄ）は、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカンダリデータユニットで置き換えられることを特徴とする方法。
【発明の詳細な説明】
【技術分野】
【０００１】
［優先権主張］
本出願は、２００８年５月３０日に出願された、同一譲受人による米国特許出願第１２／１３０，８９２号の優先権の利益を主張し、開示内容全体を参照により組み入れる。
【０００２】
［技術分野］
本発明の実施の形態はコンピュータゲームおよび関連するアプリケーションに関し、特に、コンピュータゲームおよび関連するアプリケーションにおけるファイル入出力（Ｉ／Ｏ）管理に関する。

【背景技術】
【０００３】
ビデオゲームのような多くのソフトウェアアプリケーションは、アプリケーション内のメディアアクセスをより効率よくかつ信頼あるものとするためのファイル入力／出力（入出力；Ｉ／Ｏ）スケジューラを含む。ファイル入出力システム（file I/O system: FIOS）は、いくつかのパーツからなるファイルにアクセスするための、スケジューラおよび任意の入出力フィルタレイヤを含むミドルウェアレイアである。スケジューラは、典型的には任意の期限および優先度制約による条件の下、可能な限り最短時間で完了するように入出力リクエストを最適に並べ替えるように設計されている。
【０００４】
多くの異なるゲームコンポーネントは、ストレージメディア内のファイルへの入出力アクセスを要求する。オーディオコンポーネントはオーディオファイルを読み込み、ゲームプレイのエンジンはレベルの定義を読み込み、グラフィックコンポーネントはテクスチャマップおよびモデルを読み込み、動画コンポーネントは音声映像ファイルを読み込み、サブシステムは巨大なＷＡＤファイルを読み込む。メディアは、ＵＭＤ（universal media disc）、ＣＤ（compact disc）、ＤＶＤ（digital video disc）、Ｂｌｕ－ｒａｙ（登録商標）ディスク等のような光学ディスク、ハードディスクのような中間ストレージメディア、あるいは発達するプラットフォームの等の他のメディアタイプのような、ゲーム配信メディアであってもよい。
【０００５】
ビデオゲームのような単一のアプリケーションは、典型的には多数のコンポーネントを持っており、各コンポーネントは独自の入出力要求を持っている。あるものはメデイアへのストリーミングアクセスを要求する。ここで入出力システムは、コンポーネントがゲームプレイヤにストリーム化されたデータを連続的に提示できるようにファイル内のデータをストリーム化する。例えば、音声コンポーネントは典型的にはサウンドトラックの再生のためにオーディオファイルをストリーミングする。動画コンポーネントはプレイヤに動画を再生するために音声映像コンテンツをストリーミングする。他のコンポーネントは、そのコンポーネントが処理するためのデータをファイルからチャンクという形で取ってくる非ストリーミングアクセスのみを必要とする。これらのコンポーネントは安定したデータフローを要求しないが、チャンクの配信のタイミングはしばしば極めて重要となる。もしアプリケーションがデータが必要なときにそれを受け取れないと、パフォーマンスを下げるかも知れない。
【０００６】
ビデオゲームはしばしばかなりの量の入出力を行う。ゲームのイベントはしばしば必ず間に合わなければならない期限を持つタイムドリブン型のものである。例えば、もしプレイヤがＡ地点からＢ地点まで移動すると、ゲームはプレイヤがＢ地点に到達する前にＢ地点に関連するデータ用のファイルを持っておく必要がある。ゲームコンポーネントは低水準の入出力基本命令（primitives）を使ってメディアからデータを読み出すことができるが、これらの基本命令は多数のコンポーネントが同じデバイスからのデータを同時に要求するときにデバイス接続を扱うことはしない。混雑した入出力リクエストはストリーミングデータフローを中断したり、必要なときにコンポーネントがデータのチャンクを取得することを妨げたりする可能性がある。９８９ＳｏｕｎｄのＳｔｒｅａｍｓａｆｅのような高レベルシステムは少なくともファイルに対して信頼できるストリームアクセスを供給するため、ストリーム化されたデータは中断されることはない。残念ながら、そのようなシステムは一つだけデータストリームを割り当てるので、非ストリームデータに対する競合をうまく扱えない。ＳＣＥＡは普通のデータアクセスに対する競合を解消する標準的なレイヤを提供しない。
【発明の概要】
【発明が解決しようとする課題】
【０００７】

アプリケーションのコンポーネントは、典型的にはお互いの入出力活動を関知していないので、それらの入出力リクエストはかなりの量の過度な探索（データを探すために行きつ戻りつすること）を含む、とても非効率な入出力パターンにしばしば陥る。ゲームが大きくなると、過度な探索および非効率なデータ取得が増加する。ＣＤのような配布メディアに対するディスクのマスター化では、頻繁にアクセスされるデータブロックを重複させたり、ディスク全体にデータブロックを分散させることで、ストレージデバイスの読み取りヘッドがどこにあってもコピーがいつも近くにあるようにする技術を通してこの非効率性をある程度避けることができる。ファイルシステムは、データが物理的にディスクにある場所を明らかにするとは限らないので、Ｉ／Ｏシステムはそのデータを最大限活用することができるとは限らない。

【０００８】

本発明の実施の形態はこのような文脈の中で生じた。

【課題を解決するための手段】

【０００９】

本発明の実施の形態によれば、プロセッサユニット、メモリ、およびメディアデバイスをもつシステムにおいて、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法が実装される。メディアデバイスに対してデータを転送するために、プロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取る。前記入力Ｉ／Ｏリクエストを完了するための予測時間T_pを計算する。前記入力Ｉ／Ｏリクエストをメモリ内に具体化されたスケジュールに挿入する。前記入力Ｉ／Ｏリクエストのスケジュール内の位置は予測時間T_pに少なくとも部分的に依存する。前記スケジュールにしたがって前記入力Ｉ／Ｏリクエストをサービスする。

【００１０】

ある実施の形態では、プロセッサ実行可能な命令セットがメモリに具体化される。前記命令は実行されたときに、上述の方法を実行するように構成されてもよい。プロセッサ実行可能な命令セットは１以上のメディアフィルタレイヤを含む。メディアフィルタレイヤは、デアーカイバレイヤ、ＲＡＭキャッシュレイヤ、スケジューラキャッシュ（たとえば、ハードディスクドライブ（ＨＤＤ)キャッシュ）レイヤ、オーバレイレイヤ、カタログキャッシュレイヤ、データ変換レイヤ、またはシミュレートされたデータレイヤを含んでもよい。

【００１１】

ある実施の形態では、メディアデバイスのパラメータと入力Ｉ／Ｏリクエストから決定される係数を用いた性能特性式（performance characteristic equation；ＰＣＥ）にしたがって予測時間T_pを決定してもよい。一例として、一般性を失うことなく、前記パラメータは、リクエストオーバーヘッドｘ、ヘッド運動レートｙ、およびスループットｚを含んでもよい。そのような例において、前記係数は、オーバーヘッド係数Ａ、ヘッド移動距離Ｂ、および転送データ量Ｃを含んでもよい。前記性能特性式は、$T_p = Ax + By + Cz$の形であってもよい。

【００１２】

ある実施の形態では、Ｉ／Ｏリクエストに対するＰＣＥ係数はＩ／Ｏリクエストと初期メディア状態にもとづいて決定されてもよい。初期メディア状態は、性能に影響するデバイスの状態に関する情報を含む。一例として、一般性を失うことなく、その情報には、デバイスのレディ状態とアクセスされる最後の論理ブロックアドレス（ＬＢＡ）が含まれる。ある実施の形態では、Ｉ／ＯリクエストとＩ／Ｏリクエストと初期メディア状態からのＰＣＥ係数の計算は、更新されたメディア状態を返し、これは別のＰＣＥ計算に対する初期メディア状態として用いることができる。

【００１３】

ある実施の形態では、ＰＣＥ係数は、所与のＩ／Ｏリクエストに対する予測時間T_pを計算するために、ＰＣＥにおける未知の値と結合されてもよい。その後、Ｉ／Ｏリクエストの順序付けされたリストに対する合計時間Ｔは、リスト内の各Ｉ／Ｏリクエストに対し

てＰＣＥとメディア状態を反復し、その結果得られる時間$Ｔ_{p}$を加算することにより計算されてもよい。この処理は、Ｉ／Ｏリクエストの順序付けされたリストのある順列またはすべての順列に対して繰り返されてもよい。リストの好ましい順序付けは、各順序づけに対して計算された時間Ｔと場合によっては他の基準とにもとづいて、複数の順列の中から選択されてもよい。ある実施の形態では、好ましい順序付けは、最小時間Ｔを与える順列であってもよい。順序付けに影響する付加的な基準は、これに限定しないが、Ｉ／Ｏリクエスト優先度、ストリームバッファ状態、およびレーテンシ要件であってもよい。

【００１４】

ある実施の形態では、Ｉ／Ｏリクエストに対してかかった現実の時間$Ｔ_{a}$を測定し、前記現実の時間$Ｔ_{a}$を前記係数とともに用いて、未知パラメータの値に対してＰＣＥを反復して解いてもよい。これにより、モデルが現実のドライブ性能で絶えず更新される。

【００１５】

ある実施の形態では、Ｉ／Ｏリクエストを実行するステップは、圧縮されたネストしたアーカイブから特定のデータを抽出するステップを含んでもよい。

【００１６】

ある実施の形態では、前記プロセッサユニットは、メインプロセッサと、関連づけられたローカルメモリをもつ補助プロセッサとを含んでもよい。そのような実施の形態では、Ｉ／Ｏリクエストを実行するステップは、メディアデバイスからローカルメモリにデータを転送するステップと、補助プロセッサを用いてローカルメモリのデータを変換するステップとを含んでもよい。

【００１７】

ある実施の形態では、前記システムはさらにグラフィックスプロセッサと関連づけられたグラフィックスメモリとを含んでもよい。そのような実施の形態では、Ｉ／Ｏリクエストを実行するステップは、メディアデバイスからグラフィックスメモリにデータを読み込むステップと、グラフィックスメモリからローカルメモリにデータを転送するステップと、補助プロセッサを用いてデータ上でデータ変換を実行して、変換されたデータを生成するステップと、変換されたデータをグラフィックスメモリに返送するステップとを含んでもよい。

【００１８】

ある実施の形態では、Ｉ／Ｏリクエストを実行するステップは、データオーバレイを実行するステップを含み、メディアデバイスとメモリ間で転送されるプライマリデータユニットは、セカンダリデータソースからのセカンダリデータユニットで置き換えられてもよい。前記プライマリデータユニットのソースと前記セカンダリデータソースは同じメディア上にあっても、異なるメディア上にあってもよい。前記セカンダリデータソースは仮想的なデータソースであってもよい。セカンダリデータユニットはコールバックによって指定されてもよい。

【００１９】

ある実施の形態では、前記システムは高速ストレージデバイスをさらに含んでもよい。そのような実施の形態では、Ｉ／Ｏリクエストを実行するステップは、メディアデバイスから高速ストレージデバイスへ１以上のデータユニットをプリフェッチするステップと、それに続いて高速ストレージデバイスからそのデータユニットを転送するステップとを含んでもよい。一例として、高速ストレージデバイスはハードディスクドライブであり、メディアデバイスはブルーレイ（登録商標）ディスクドライブのような大容量光ディスクドライブであってもよい。Ｉ／Ｏがアイドルであるとき、ある時間間隔の間、１以上のデータユニットをプリフェッチしてもよい。ある実施の形態では、プリフェッチするステップは、メディアデバイスから１以上のデータユニットの第１グループをプリフェッチするステップと、メディアデバイスがアイドルかどうかを調べるステップと、もしＩ／Ｏがアイドルなら、メディアデバイスから１以上のデータユニットの第２グループをプリフェッチし、もしＩ／Ｏがアイドルでないなら、前記第２グループのプリフェッチを遅延させるステップとを含んでもよい。

【００２０】
ある実施の形態では、予測時間T_pを計算するステップは、前記システムに関連づけられたメディアデバイスよりも低速のメディアのオペレーションをエミュレートするために前記パラメータを調整するステップを含んでもよい。
【００２１】
本発明の実施の形態によれば、メインプロセッサと、関連づけられたローカルメモリをもつ補助プロセッサと、メインメモリと、メディアデバイスとを含むシステムにおいて、メディアデバイスに対する入出力（Ｉ／Ｏ）をハンドリングする方法が実装される。この方法は、ａ）メディアデバイスに対してデータを転送するために、メインプロセッサ上で動作するアプリケーションから入力Ｉ／Ｏリクエストを受け取るステップと、ｂ）前記入力Ｉ／Ｏリクエストをメインメモリ内に具体化されたスケジュールに挿入するステップと、ｃ）スケジュールと１以上のフィルタにしたがって前記入力Ｉ／Ｏリクエストをサービスするステップとを含み、１以上のフィルタの少なくとも１つは補助プロセッサによって実行される。
【図面の簡単な説明】
【００２２】
本発明の実施の形態は、添付の図面と併せて、以下の詳細な説明を考慮することによって容易に理解される。
【図１】本発明の実施の形態に係るファイル入出力システム（ＦＩＯＳ）を実装するシステムを示すブロック図である。
【図２】本発明の別の実施の形態に係るＦＩＯＳを実装する別のシステムを示すブロック図である。
【図３】本発明の実施の形態に係るファイル入出力システム（ＦＩＯＳ）ソフトウェアのブロック図である。
【図４】本発明の実施の形態に係るＦＩＯＳのオペレーションの流れを例示するデータフロー図である。
【図５】本発明の実施の形態に係るＦＩＯＳで使用されるソフトウェアスケジューラの例を示すフロー図である。
【図６】本発明の実施の形態に係るＦＩＯＳにおけるプリフェッチの例を説明する模式図である。
【図７Ａ】本発明の実施の形態に係る関連づけられたローカルメモリをもつ補助プロセッサを用いたＩ／Ｏバッファリングの例を説明するフロー図である。
【図７Ｂ】本発明の実施の形態に係る関連づけられたローカルメモリをもつ補助プロセッサを用いたＩ／Ｏバッファリングとデータ変換の例を説明するフロー図である。
【図８】本発明の実施の形態とともに用いられるネストしたアーカイブに対するデータ構造を説明するブロック図である。
【図９】本発明の実施の形態に係るデアーカイバを説明するフロー図である。
【発明を実施するための形態】
【００２３】
以下の詳細な説明には、説明のための多くの具体的な詳細事項が含まれるが、それらに対する多くの変更や修正も本発明の技術範囲に含まれることは、当業者であれば誰でも理解するであろう。したがって、以下に説明する本発明の例示的な実施の形態は、クレームされた発明の一般性をなんら損なうことなく、また限定も行うことなく記載されている。
【００２４】
本発明の実施の形態は、システム用の全ての入出力が通過する集中型のレイヤを提供するファイル入出力システム（file I/O system: FIOS）によって実装されうる。ファイル入出力システムは、ここの入出力リクエストの最適な処理のための木構造を作成する命令と、入出力リクエストをスケジュールし、入出力リクエストが最も効率的にサービスを提供する順序を決定する命令とを含む。
【００２５】

例として、図１に示すように、コンピュータで実装されたシステム１００が本発明の実施の形態に係るファイル入出力システム（file I/O system: FIOS）を実装するように構成されてもよい。例として、一般性を失うことなく、システム１００は本発明の実施の形態の実行に適したパーソナルコンピュータ、ビデオゲームのコンソール、ＰＤＡ（personal digital assistant）、または他のデジタルデバイスとして実装されてもよい。システム１００はＣＰＵ１０５と、処理ユニット１０５に接続するメインメモリ１０６とを含んでもよい。ＣＰＵ１０５はソフトウェアアプリケーションおよび、オプションとしてオペレーティングシステムを実行するように構成されている。セルプロセッサアーキテクチャの例は、「セルブロードバンドエンジンアーキテクチャ」に詳細が記述されている。これは、ＩＢＭ、ソニー・コンピュータエンタテインメント、東芝の２００５年８月８日付けの著作権で保護されており、コピーはｈｔｔｐ：／／ｃｅｌｌ．ｓｃｅｉ．ｃｏ．ｊｐ／からダウンロード可能であり、その内容全体を参照によってここに組み入れる。

【００２６】

メインメモリ１０６は、ＣＰＵ１０５に使用されるアプリケーションおよびデータを格納する。メインメモリ１０６は、例えばＲＡＭ、ＤＲＡＭ、ＲＯＭやそれと同様の集積回路の形態であってもよい。コンピュータプログラム１０１は、ＣＰＵ１０５上で実行可能な命令の形態でメインメモリ１０６内に格納される。プログラム１０１の命令は他のもの、すなわち以下に記載する特徴を持ったファイル入出力システム（file input/output system: FIOS）で実装するよう構成されてもよい。メモリ１０６は、例えばスタックまたはキューの形態で、ファイル入出力プログラム１０１に使用される受信、スケジュール、発行、完了、および解放の入出力リクエストである入出力キュー１０１Ｑを含む。それらのキューの例もまた以下に記載される。

【００２７】

例として、ＦＩＯＳプログラム１０１は、ａ）メディアデバイス１１８に関わる入力される（incoming）入出力（Ｉ／Ｏ）リクエストをアプリケーション１０３から受信し、ｂ）その入出力リクエストを完了する予測時間T_pを計算し、ｃ）少なくとも部分的に予測時間T_pにもとづいてメモリ１０６内のスケジュールのある位置にその入力された入出力リクエストを挿入し、ｄ）そのスケジュールにしたがって入出力リクエストを実行する命令を含む。

【００２８】

ＦＩＯＳプログラム１０１は、インタラクティブな環境を実装するように構成された命令とともに機能する。一例として、そのような命令は、ビデオゲームプログラムのような、メインプログラム１０３のサブルーチンまたは呼び出し可能な関数である。あるいは、メインプログラム１０３は、仮想世界とインタフェースをもつためのプログラムである。メインプログラム１０３は、ビデオディスプレイ上でカメラＰＯＶ（視点）からシミュレートされた環境の一部のシーンを表示し、シミュレートされた環境とのユーザの相互作用の間、カメラ経路に沿ってカメラＰＯＶの動きに応じてカメラＰＯＶが変化するにつれてシーンを変化させるように構成される。メインプログラムは、物理シミュレーション１０４とカメラ管理１０７に対する命令を含む。メインプログラム１０３は、ＦＩＯＳプログラム１０１、物理シミュレーション命令１０４、カメラ管理命令１０７、および広告インプレッション報告命令１０９をたとえば関数またはサブルーチンとして呼び出してもよい。

【００２９】

クライアントシステム１００はさらに、入出力（Ｉ／Ｏ）装置１１１、電源（Ｐ／Ｓ）１１２、クロック（ＣＬＫ）１１３およびキャッシュ１１４などの公知のサポート機能５１０を備えてもよい。クライアントシステム１００は、アプリケーションとデータ用の不揮発性のストレージを提供するハードディスクのような高速ストレージ１１５をさらに備えてもよい。高速ストレージ１１５は、他のものの間で、より遅いメディアデバイス１１８から読み出されたファイル１１６の一時的または長期間のストレージに使用されてもよい。さらに、高速ストレージ１１５上のファイル１１６は、より遅いメディアデバイス１

１６以外の他のソースから来てもよい。限定されるものではないが、例えば、ファイル１１６はオペレーティングシステムのファイル、アプリケーションによって作成されたテンポラリファイル、写真、音声、映像のようなユーザのデータ、ダウンロードされたコンテンツ、その他を含んでもよい。例として、高速ストレージ１１５は固定ディスクドライブ、リムーバブルディスクドライブ、フラッシュメモリデバイス、あるいはテープドライブであってもよい。より遅いメディアデバイス１１８は、例えばＣＤ－ＲＯＭドライブ、ＤＶＤ－ＲＯＭドライブ、ＨＤ－ＤＶＤ（high-definition digital versatile disc）ドライブ、Ｂｌｕ－Ｒａｙ（登録商標）ドライブ、ＵＭＤドライブ、または他の光学ストレージデバイス等の大容量光学ディスクドライブであってもよい。メディアデバイス１１８からのプリフェッチファイル１１６は、メインメモリ１０６に素早く読み出されるために、高速ストレージ１１５のハードウェアキャッシュに一時的に格納されてもよい。

【００３０】

１またはそれ以上の入力デバイス１２０は、１またはそれ以上のユーザからのユーザ入力をシステム１００に伝えるために使われてもよい。例として、１またはそれ以上のユーザ入力デバイス１２０は入出力エレメント１１１を経由してクライアントデバイス１００と接続されていてもよい。適切な入力デバイス１２０の例は、キーボード、マウス、ジョイスティック、タッチパッド、タッチスクリーン、光学ペン、静止または動画カメラ、および／またはマイクロフォンを含む。クライアントデバイス１００は電子通信ネットワーク１２７を介してコミュニケーションを促進するためのネットワークインタフェース１２５を含んでもよい。ネットワークインタフェース１２５はローカルエリアネットワークおよびインターネットのようなワイドエリアネットワーク上の有線または無線通信を実装するように構成されていてもよい。システム１００は、データを送信および受信し、および／またはネットワーク１２７を介して１またはそれ以上のメッセージパケット１２６を通じたファイルをリクエストしてもよい。

【００３１】

システム１００は、グラフィックプロセッシングユニット（ＧＰＵ）１３５とグラフィックスメモリ１４０とを有するグラフィックス・サブシステム１３０を更に含む。グラフィックスメモリ１４０は、出力イメージの各画素に対する画素データを格納するために使われるディスプレイメモリ（例えばフレーム・バッファ）を含む。グラフィックスメモリ１４０は、ＧＰＵ１３５と同じデバイス内に統合されてもよく、別のデバイスとしてＧＰＵ１３５に結合されてもよく、メモリ１０６内に実装されてもよい。画素データは、ＣＰＵ１０５から直接グラフィックスメモリ１４０に提供される。あるいは、ＣＰＵ１０５は、所望の出力イメージを定めるデータやインストラクションをＧＰＵ１３５に提供する。ＧＰＵ１３５はそれを用いて１またはそれ以上の出力イメージの画素データを生成する。所望出力イメージを定めるデータやインストラクションは、メモリ１０６やグラフィックスメモリ１４０に格納される。ある実施の形態では、ＧＰＵ１３５は、例えば、適当なプログラミングまたはハードウェア・コンフィギュレーションによって構成され、シーンのためにジオメトリ、ライティング（照明）、シェーディング、テクスチャリング、モーション、および／またはカメラパラメータを定めるインストラクションとデータから出力イメージに対する画素データを生成するための３Ｄレンダリング機能をもつ。ＧＰＵ１３５は、シェーダプログラムを実行することができるプログラム可能な実行ユニットを更に含む。

【００３２】

グラフィックス・サブシステム１３０は、表示装置１５０で表示されるべきグラフィックスメモリ１４０からのイメージに対する画素データを周期的に出力する。表示装置１５０は、システム１００からの信号に応じてビジュアル情報を表示することができる任意のデバイスであり、ＣＲＴ、ＬＣＤ、プラズマ、およびＯＬＥＤディスプレイを含む。コンピュータシステム１００は、表示装置１５０にアナログまたはデジタル信号を提供する。一例として、表示装置１５０は、テキスト、数字、グラフィカルシンボル、または画像を表示するＣＲＴまたはフラットパネルスクリーンを含む。さらに、ディスプレイ１５０は

、可聴もしくは検知可能な音を出すオーディオスピーカーを含んでもよい。そのような音の生成を容易にするために、クライアントデバイス１００は、ＣＰＵ１０５、メモリ１０６、および／またはストレージ１１５によって提供されたインストラクションおよび／またはデータからアナログまたはデジタル音声出力を生成するために適したオーディオプロセッサ１５５を更に含む。

【００３３】

ＣＰＵ１０５、メモリ１０６、支援機能１１０、データストレージデバイス１１５、ユーザ入力デバイス１２０、ネットワークインタフェース１２５、およびオーディオプロセッサ１５５を含むシステム１００の構成要素は、データバス１６０を介して互いに機能的に接続されている。これらの構成要素は、ハードウェア、ソフトウェアまたはファームウェア、あるいはこれらのうちの２つ以上の組合せで実装される。

【００３４】

本発明のある実施の形態は、セルプロセッサアーキテクチャやそれと類似するプロセッサアーキテクチャをうまく利用する。図２は、本発明の別の実施の形態に係るファイル入出力システムを実装するセルプロセッサ２００を示す。セルプロセッサ２００は、メインメモリ２０２、ひとつのパワープロセッサ要素（ｐｏｗｅｒ　ｐｒｏｃｅｓｓｏｒ　ｅｌｅｍｅｎｔ：ＰＰＥ）２０４、および８つのシナジスティックプロセッサ要素（ｓｙｎｅｒｇｉｓｔｉｃ　ｐｒｏｃｅｓｓｏｒ　ｅｌｅｍｅｎｔ：ＳＰＥ）２０６を備える。図４の例では、セルプロセッサ４００は、１個のＰＰＥ４０４と８個のＳＰＥ４０６とを備える。このような構成では、７個のＳＰＥ４０６が並列処理に用いられ、１個は他の７個の一つが故障したときのバックアップとして確保されている。例示として、ＰＰＥ２０４は、関連するキャッシュを持った６４ビットのパワーＰＣプロセッサユニット（ＰＰＵ）であってもよい。ＰＰＥ２０４は、システム管理リソース（例えば、メモリ保護テーブルなど）にアクセス可能な汎用処理装置である。ハードウェアリソースは、ＰＰＥに見られるように実アドレス空間に明示的にマップされてもよい。したがって、ＰＰＥは適当な実効アドレス値を用いて、直接これらのリソースのいずれにでもアドレス指定できる。ＰＰＥ２０４の主な機能は、セルプロセッサ２００のＳＰＥ２０６に対するタスク管理とタスク割り当てである。ハードウェアリソースは、ＰＰＥ２０４に見られるように実アドレス空間に明示的にマップされてもよい。したがって、ＰＰＥ２０４は適当な実効アドレス値を用いて、直接これらのリソースのいずれにでもアドレス指定できる。ＰＰＥ２０４の主な機能は、異なるＳＰＥ２０６に対するタスク管理とタスク割り当てである。ＰＰＵはＦＩＯＳプログラム２０５のコード化された命令を実行する。

【００３５】

なんらシステム管理機能を果たさないＳＰＵ２０６は、ＰＰＥ２０４よりは簡単な構造のコンピュータ装置である。ＳＰＥ２０６のそれぞれは、ときにシナジスティックプロセッサユニット（ｓｙｎｅｒｇｉｓｔｉｃ　ｐｒｏｃｅｓｓｏｒ　ｕｎｉｔ：ＳＰＵ）として参照されるプロセッサユニットと、関連するローカル記憶領域ＬＳ（local store）とを含む。ＳＰＵ２０６は、一般にＳＩＭＤ（ｓｉｎｇｌｅ　ｉｎｓｔｒｕｃｔｉｏｎ，ｍｕｌｔｉｐｌｅ　ｄａｔａ）機能を有し、通常はデータ処理を行い、割り当てられたタスクを行うために（ＰＰＥ２０４により設定されたアクセス特性にしたがって）要求されたデータ転送を開始する。ＳＰＥ２０６はファイル入出力プログラム１０１の一部を実装するローカルストア命令２０７を自身のローカルストア内に格納する。ＳＰＵ２０６の目的は、より高いコンピュータ装置密度を要求するアプリケーションを可能として、提供された命令セットを有効に利用できるようにすることである。本例では８個のＳＰＥが示されているが、セルプロセッサ２００は任意の数のＳＰＥを用いて設定される。図２を参照すると、メモリ２０２、ＰＰＥ２０４、およびＳＰＥ２０６は、リングタイプの要素相互接続バス２１０を介して、お互いにおよび入出力デバイス２０８と通信可能である。メモリ２０２は、メモリインタフェースコントローラ（ｍｅｍｏｒｙ　ｉｎｔｅｒｆａｃｅ　ｃｏｎｔｒｏｌｌｅｒ：ＭＩＣ）を介してＰＰＥ２０４およびＳＰＥ２０６からアクセスされる。

【００３６】
メモリ２０２は、以下に記載するような受信（incoming）、スケジュール、発行、完了、解放、およびプリフェッチキュー等の入出力キュー２０３を含む。メモリ２０２は、本明細書に記載されているＦＩＯＳプログラム１０１と同様の特徴を持っているＦＩＯＳプログラム２０９の部分も含む。ＳＰＥ２０６の少なくとも１つは、以下に述べるデータ圧縮解凍を実装するように構成されたコード２０７をローカルストア内に有する。ＰＰＥ２０４は内部キャッシュ（Ｌ１）および外部キャッシュ（Ｌ２）を含んでもよい。ＰＰＥはこの内部キャッシュ（Ｌ１）にＦＩＯＳプログラム２０５の部分を格納する。ＦＩＯＳプログラム２０５とＳＰＥに実装されたファイル転送命令２０７、またはその一部とはまた、必要なときにＳＰＥ２０６とＰＰＥ２０４によってアクセスされるためにメモリ２０２に格納されてもよい。

【００３７】
例として、ＦＩＯＳプログラム１０１／２０５は、ストレージデバイス１１５のようなハードウェアとのインタフェースを容易にするためにメディアスタックを含んでもよい。メディアスタックは図３に示すように実装される。具体的には、メディアスタック３００は、スケジューラ３０２、１またはそれ以上のメディアフィルタレイヤ３０４、デバイスメディアレイヤ３０６、プロセッサファイルシステム（ＦＳ）読み込みレイヤ３０８、およびハードウェアレイヤ３１０を含んでもよい。デバイスメディアレイヤ３０６（ときどきメディアアクセスレイヤと呼ばれる）は、メディアフィルタレイヤ３０４から受け取ったメディアリクエストに応答し、要求されたデータを受け取り、そのリプライをスタックに返すように構成される。

【００３８】
メディアフィルタレイヤ３０４は、デアーカイバレイヤ３０１、ＲＡＭキャッシュレイヤ３０３、スケジューラキャッシュレイヤ３０５、オーバレイレイヤ３０７、カタログキャッシュレイヤ３０９、１以上のデータ変換レイヤ３１１、シミュレートされたデータレイヤ３１３、および速度エミュレーションレイヤ３１５を含む。

【００３９】
デアーカイバ３０１は、圧縮されたアーカイバから特定のアセットファイルを抽出することを手助けするのに用いられてもよい。ＲＡＭキャッシュレイヤ３０３は、例えばメインメモリ１０６、２０２内にデータをキャッシュすることを実装するために用いられてもよい。スケジューラキャッシュ３０５は、光学ディスクのようなより遅いソースからのデータを一時的に格納するための、単純なディスクからディスクへの（disc-to-disc）キャッシュであってもよい。本明細書で用いられているように、「スケジューラキャッシュ」という用語は、メディアデバイス１１８へのアクセスよりも速いアクセスであるストレージメディア内への一時的なデータの格納をいう。スケジューラキャッシュ３０５内の全てのデータがプリフェッチされる必要はなく、いくらかは要求に応じて読み出されてキャッシュ内にコピーされてもよい。

【００４０】
限定されるわけではないが、例として、スケジューラキャッシュレイヤ３０５は、そのような一時的な格納場所を生み出すために高速ストレージメディア１１５を利用する。高速ストレージメディアがハードディスクドライブ（hard disk drive: HDD）である特別なケースでは、スケジューラキャッシュ３０５はときにハードディスクドライブキャッシュと呼ばれる。

【００４１】
スケジューラキャッシュ３０５は、単一のファイルまたは多数のファイルとして保持されてもよい。加えて、スケジューラキャッシュ３０５のコンテンツは、欠ける部分のない全体のファイルである必要はない。多数のファイルのキャッシュは、多くのデータを犠牲にすることなく必要なときにインテリジェントにディスクスペースを空けるために、いくつかの個々のファイルを消去することによって部分的に消去されてもよい。対称的に、単一ファイルのキャッシュは、典型的には切り捨てられるかあるいは完全に消去される。複

数ファイルのキャッシュは典型的には追加の帳簿作業（bookkeeping work）を要求するため、単一ファイルのキャッシュは複数ファイルのキャッシュよりも高いパフォーマンスを提供する。帳簿作業は追加の入出力を要求し、ホストファイルシステム自体の内部で行われる。

【００４２】

オーバーレイレイヤ３０７は、以下で述べられるように、ファイルシステムレベルにおいてファイルおよびディレクトリの任意のオーバーレイを可能にするために用いられてもよい。カタログキャッシュレイヤ３０９は、プロセッサユニットまたはセルプロセッサのためのオペレーティングシステムが適切にキャッシュしないであろうデータ（例えばファイルの存在やサイズ、格納場所等）をキャッシュするために用いられてもよい。データ変換レイヤ３１１は、メディアデバイス１１８から読み出したり、メディアデバイス１１８に書き込まれるデータに暗号、暗号解読、圧縮または圧縮伸張のようなデータ変換を実行する。シミュレートされたデータレイヤ３１３は、たとえば、そのようなシミュレートされたデータを要求するＩ／Ｏリクエストに対してゼロまたは乱数を生成するために使われる。速度エミュレーションレイヤ３１５は、たとえばソフトウェアの開発過程で、以下で説明するように、メディアデバイスの異なる速度をシミュレートするために用いられる。

【００４３】

スケジューラ３０２のオペレーションは、以下に述べられる図４を参照することで理解されよう。クライアントアプリケーションの観点から、入出力リクエスト４０３は比較的直接的な方法で実行される。例えば、ビデオゲームのようなアプリケーション４０１内のスレッド４０２は、関数（例えば、readFile()）を呼び出すことによってＦＩＯＳ１０１／２０５を通して入出力をリクエストしてもよい。この関数は、ＦＩＯＳリクエスト４０３およびリクエストが完了されるべき期限を特定してもよい。この関数は、バッファ４０５へのポインタを提供してもよい。そのような実装は、Ｉ／Ｏリクエスト４０３に対するＩ／Ｏオペレーションの構造体を自動的に割り当て、新しく割り当てられたオペレーションを受信（incoming）キュー４０４に移動させる。例として、受信キューはＦＩＦＯ（first-in-first-out）キューであり、キュー４０４内に最初に置かれたオペレーションが、スケジューラ３０２によって最初にスケジューリングされるオペレーションである。受信キュー４０４内へのリクエスト４０３の挿入は、非同期の入出力を要求するときにスレッドがブロックされることを防ぐために、アトミックな操作（atomic operation）によって実装されてもよい。ある実施の形態では、受信キュー４０４は、アトミックな交換（atomic exchange）とともに補助プロセッサ１０５Ｂにしたがうアトミックなスタック（atomic stack）の形態であってもよい。

【００４４】

限定されないが、例として、入出力リクエスト４０３が読み込み命令の場合、クライアントアプリケーション４０１はメディアデバイスから読み込んだデータを受信するためのバッファ４０５を提供してもよい。入出力リクエストが完了したとき、クライアントアプリケーションは提供したバッファから読み出されたデータを読み込んでもよい。クライアントが入出力リクエスト４０３の処理を終了したとき、クライアントは、将来の入出力での使用においてリソースが利用可能となるように、オペレーションの構造体の割り当てを解除（de-allocate）してもよい。

【００４５】

入出力リクエスト４０３は、スケジューラ３０３がまだ起動していない場合、スケジューラ３０３を起動してもよい。その後、アプリケーション４０１は入出力リクエスト４０３が完了するまで待機してもよい。例えば、クライアントアプリケーション４０１は、アトミックなメソッド（atomic method、例えばisDone()）を用いて周期的にポーリングしてもよい。もう一つの方法としては、クライアントは、入出力リクエスト４０３が完了したときにファイル入出力プログラム１０１が呼び出すコールバック関数４１１を待ってもよい。

【００４６】

受信キュー４０４内に入出力リクエスト４０３が挿入された後に、ＦＩＯＳプログラム１０１／２０５はスケジュール挿入４０７を実行するためにスケジューラ３０２を呼び出してもよい。スケジューラによって実装された演算の順序は図５を参照することで理解されるであろう。５０２で示すように、スケジューラ３０２は、扱うべき入出力がない場合、スリープ（例えば、依然として起動していない）状態であってもよい。新しい入出力リクエスト４０３が受信キュー４０４に入ると、スケジューラはスリープを解除してそれを扱う。ひとたびスケジューラがスリープを解除（あるいは既に解除）すると、受信キュー内の新しいオペレーションに気づいて、そのリクエストをスケジューリングキュー４０６内の適切な場所に移動する。

【００４７】

５０３に示すように、入出力リクエスト４０３のキューの位置を決定するために、スケジューラ３０２はオプションとして、例えばデバイスメディアレイヤ３０６やＦＩＯＳスタック３００内の他のレイヤに問い合わせることにより、メディアデバイスの現在の状態を決定してもよい。状態データは、ＦＩＯＳスタック３００によって扱われるメディアデバイスのタイプおよびスタック内にある様々なレイヤに依存して変わりうる。例としては、もっとも最近にアクセスされた論理ブロックアドレス（logical block address: LBA）、ＲＡＭキャッシュの最後にアクセスされたパスおよびオフセット、現在のディスクレイヤ、先頭の位置、ストリーミングのモード、およびストリーミングではないモード（non-streaming mode）が含まれる。

【００４８】

本発明の実施の形態によれば、スケジューラ３０２はメディアデバイスのパフォーマンスモデル３０２Ａに基づいてもよい。現存の入出力スケジューラとは異なり、ドライブモデル（drive model）は、リクエスト４０３の最適なスケジュールを決定する際にメディアデバイス１１８のパフォーマンスに関するデータを考慮に入れる。ドライブモデルを用いて入出力リクエストをスケジューリングすることは、ＣＤを生成する際のマスタ化処理の逆と比較されてもよい。パフォーマンスモデル３０２Ａは、オーバーヘッド、ディスクの移動時間、読み込み時間、およびそれと同様のもののような要素を考慮するように構成されていてもよい。パフォーマンスモデル３０２Ａは、スループット、レーザの細かな揺れ（wiggles）、読み取りヘッドの動き、レーザの変更、およびリクエストのオーバーヘッドのようなメディアデバイス１１８の任意の複雑な特性をモデル化してもよい。パフォーマンスモデル３０２Ａはまた、入出力リクエストによって引き起こされたメディアデバイス１１８によって読み込まれたり書き込まれたりする特定のメディアの他のパラメータを考慮に入れてもよい。例えば、パフォーマンスモデル３０２Ａは、デバイスが単一のレイヤのディスクを読み込んでいるのか、あるいは多数のレイヤのディスク（例えばＢｌｕ－ｒａｙ（登録商標）ＤＭＤのような二重のレイヤのディスク）を読み込んでいるのかを考慮してもよい。

【００４９】

スケジューラ３０２はまた任意であるが、符号５０４で示すように、新しいＩ／Ｏオペレーションの時間要件、優先度、および見込まれる効率性を調べる。符号５０６で示すように、スケジューラは、受信Ｉ／Ｏリクエストを完了する予測時間T_pを計算するように構成される。好ましい実施の形態によれば、スケジューラ３０２は、メディアデバイス１１８に関するパラメータとＩ／Ｏリクエスト４０３に関する係数を変数としてもつ性能特性式（performance characteristic equation；ＰＣＥ）にしたがって、Ｉ／Ｏリクエストを実行するための予測時間T_pを決定してもよい。

【００５０】

ＰＣＥは次の一般的な形式である。

$T_p = Af_1(x) + Bf_2(y) + Cf_3(z) + Df_4(w) + Ef_5(v) + \ldots$

ここで、Ａ、Ｂ、Ｃ、ＤおよびＥは係数であり、ｘ、ｙ、ｚ、ｗおよびｖはメディアデバイスの性能特性に関するパラメータを表し、$f_1(x)$、$f_2(y)$、$f_3(z)$、f_4

（ｗ）およびf_5（ｖ）はｘ、ｙ、ｚ、ｗおよびｖの関数である。一例として、限定しないが、ＰＣＥは１以上の項の和の形式の線形式であり、各項はパラメータ値（ｘ、ｙ、ｚなど）を表す変数を乗じた係数（Ａ、Ｂ、Ｃなど）である。そのような式は次の形式である。

$T_p = Ax + By + Cz + Dw + Ev + \ldots$

【００５１】

上記式は任意の数の項目を有してもよい。変数は、要求されたタスクを実行するための時間に影響するメディアデバイスの任意のパラメータに対応する。そのようなパラメータの例には、これに限定しないが、オーバーヘッド、シーク時間、データ転送レートが含まれる。実際上、ＰＣＥは３より大きい数の項目、たとえば１０以上の項目を用いる。たとえば、一つの読み取り（リード）オペレーションを記述するためにはしばしば６つの重要な項目が必要である。１．ファイルオープン、２．オーバーヘッド、３．シーク、４．転送、５．圧縮解凍、６．ファイルクローズ。他のオペレーション（オープン、クローズ、アンリンク（unlink）、リードディレクトリ（read directory））には、典型的には異なる項目の集合が必要である。

【００５２】

線形式の代わりとして、１以上の項の和であり、各項は何乗かした対応する変数によって乗じた係数（Ａ）である一般的な多項式にＰＣＥを拡張してもよい。たとえば、ＰＣＥは次の形式をもつ。

$T_p = Ax + By + Cz^2 + Dw^{10} + Ev^{-2} \ldots$

【００５３】

別の代案として、１以上の項の和として表され、各項は対応する変数の任意の関数によって乗じた係数（Ａ）である一般的な式にＰＣＥを拡張してもよい。たとえば、ＰＣＥは次の形式をもつ。

$T_p = Ax + B\log y + Ce^z + D(w^2 + 2w + 1) + \ldots$

【００５４】

単純な例として、線形ＰＣＥは$T_p = Ax + By + Cz$の形式であり、ここで、ｘはリクエスト４０３に対するオーバーヘッドに関するパラメータであり、ｙはセクタ単位のヘッドの動きに関するパラメータであり、ｚはデータスループットレート（たとえば、所与のデータ量を読み出すまたは書き込む時間）に関するパラメータである。係数Ａ、ＢおよびＣの値は、スケジュールキュー４０４内のリクエスト４０３の位置に依存する。

【００５５】

一例として、係数Ａ、Ｂ、Ｃは、状態（ステート）だけでなくリクエスト４０３内の情報からも決定される。たとえば、一般性を失うことなく、その情報には、デバイスのレディ状態とアクセスされる最後の論理ブロックアドレス（ＬＢＡ）が含まれる。最後の論理ブロックアドレスは、ドライブの読み取り（リード）ヘッドの位置を示し、これにより、シーク距離（ヘッド移動距離）を計算することが可能になる。

【００５６】

ある例では、係数Ａはオーバーヘッド係数と呼ばれる。この係数は、オーバーヘッドが部分的には、Ｉ／Ｏリクエスト４０３の性質だけでなく、メディアデバイス１１８の特性にも依存するという事実を考慮に入れている。係数Ｂは、データが読み取られるべきメディア上の位置に到達するために、メディアデバイス１１８上の読み取りヘッドが取らなければならないヘッドの運動量に関する。たとえば、リクエスト４０３は、メディアデバイス１１８がセクタ１０００からスタートする１０セクタを読むように要求する。そのような場合において、データ係数Ｃは読み取るべきセクタ数である。運動係数Ｂは、デバイスメディアレイヤ３０６から取得されるステート（状態）情報から決定される。ステート情報には、Ｉ／Ｏリクエストの開始時における読み取りヘッドの初期位置が含まれる。たとえば読み取りヘッドが初期的にセクタ５００にあるなら、運動係数Ｂは、読み取りが開始するべきセクタ（セクタ１０００）と現在のセクタ（セクタ５００）の差、たとえばＢ＝１０００－５００＝５００から決定される。スケジューラ３０２は、Ｉ／Ｏリクエスト４

０３を完了した後で最終ステートを決定する。たとえば、１０セクタが１０００番から読み取られるなら、最終ステートは、読み取りヘッドがセクタ１０１０で終わることを示す。

【００５７】

ＰＣＥはまたＩ／Ｏリクエストを書き込むためのありうる異なるシナリオを考慮に入れてもよい。たとえば、メディアデバイス１１８が複数のレイヤ媒体を用いるなら、ＰＣＥは、あるレイヤから別のレイヤに切り替えて読み取りヘッドを動かす予測時間T_pに与える影響を考慮に入れる。

【００５８】

ある実施の形態において、任意のＩ／Ｏリクエストに対する現実にかかった時間T_aを測定し、ＰＣＥを反復して解いて未知変数の値を求めるためにＰＣＥ係数とともに用いてもよい。これにより、モデルを現実のドライブ性能で絶えず更新することができる。

【００５９】

いったん予測時間T_pが計算されると、スケジューラ３０２は同様の方法でスケジュールキュー４０６における他のリクエストに対する予測時間を決定する。この処理は、符号５０７に示すように、リクエスト４０３に対する現実の時間と最適なスケジュールキューの順序が決定されるまで、複数の異なるキュー順序で反復して繰り返される。リクエスト４０３に対する初期状態は、スケジュール順位における以前のリクエストに対する最終状態から決定され、スケジュール順位における次のリクエストに対する初期状態は、リクエスト４０３の最終状態から決定される。

【００６０】

スケジューラ３０２は、スケジューリングキュー内のリクエストを検証して、既にキュー内にあるリクエストに対してリクエスト４０３の特性を比較する。スケジューラ３０２は、キュー内におけるリクエスト４０３の考え得る各位置を試し、優先度を無視していること、期限を過ぎること、およびタイミングの判断を探す。スケジューラ３０２は、リクエストされた入出力オペレーションがいずれも期限を過ぎない順序を探すことによって、考え得る最適な新しいキューの順序を決定してもよい。もし１またはそれ以上のオペレーションが絶対に期限を途過するという場合、スケジューラは異なる基準を用いてキューの順序を決定してもよい。スケジュールキュー４０６内のリクエストの順序に影響を与える追加の基準は、限定はされないが、入出力リクエストの優先度、ストリームバッファの状態、および要求されるレイテンシを含んでもよい。ある実施の形態では、T_p計算に対するＰＣＥ係数は、別のＰＣＥ係数計算に対する初期メディア状態として使うことができる更新されたメディア状態を返すために使われる。

【００６１】

一例によれば、スケジューラ３０２は、異なる挿入順列（permutation）を調べる。たとえば、オペレーション１、２、３を含む既存スケジュールにオペレーション４を挿入する。各順列は異なる合計時間Ｔをもたらす。ありうる順列と時間には以下がある。

４，１，２，３→T_1
１，４，２，３→T_2
１，２，４，３→T_3
１，２，３，４→T_4

【００６２】

スケジューラ３０２は、どの順列が最小予測合計時間Ｔを生成するかを決め、この順列を好ましいものとして使う。

【００６３】

さらに、スケジューラ３０２は、たとえば、スケジュールされた各アイテムに対する時間を加算することにより、徒過が予測される期限を探し、これにより、いずれかのスケジュールされたアイテムが自分の期限を徒過することになるかどうかを調べる。これには、スケジュールされた各オペレーションに対する完了時間を決定し、その完了時間をそのオペレーションに対する期限と比較する。いずれかの期限が徒過されるならば、オペレーシ

ョンはキュー内のより早い位置に再スケジュールされる。

【００６４】

ある実施の形態では、例えば期限が全て解決できない場合、優先度が使用される。例えば、ある期限が否応なく徒過する場合、最良のキューの順序は、最も優先度の低いリクエストがその期限を徒過する順序であろう。前述の考察を満たす多数の可能なキューの順序がある場合、同じ優先度である期限を徒過するリクエストの数が最小となるキューの順序が最良の順序であろう。前述した考察を全て満たす多数の可能なキューの順序がある場合、可能な限り少ない時間で全てのキューを実行するキューの順序が最良の順序であろう。いくつかの場合、高い優先度のリクエストが期限に間に合う限りにおいて、低い優先度のリクエストは高い優先度のリクエストの前にスケジューリングされてもよい。

【００６５】

前述した考察を全て等しく満たす多数の可能なキューの順序がある場合、スケジュールキュー４０６内の最新のリクエストはキューの末尾に来る順序が最良の順序であろう。スケジューラ３０２がキューに新しいオペレーションを挿入するために合計キュー実行時間を見積もらなければならない場合は、多数の異なるファクタを考慮する。たとえば、スケジューラは、キュー順序の評価の直前にスケジューラ３０２が取り出したデバイスドライバの現在の状態を考慮に入れる。それに加えて、スケジューラ３０２は、メディアフィルタレイヤとＦＩＯＳスタック３００における他のレイヤのよって報告された性能係数を考慮する。これらの値には、たとえば、ヘッドの動き、ヘッドアライメント、ＤＭＡセットアップ、データ転送レート、ストレージレイヤに対するレイヤ変更などが含まれる。これらの値は、Ｉ／Ｏ性能のランタイム測定によって供給されるので、見積もりを可能な限り真実なものにすることができる。

【００６６】

ひとたびスケジューラ３０２がリクエスト４０３におけるオペレーションに対してキュー４０６内の最適な位置を決定すると、図４の４０７および図５の５０８に示されるように、オペレーションはそこに挿入される。図４を再び参照すると、もし入出力リクエストを実行可能なリソースが存在すれば、４０９に示されるように、スケジューラ３０２はスケジュールキュー４０６内の最初のオペレーションを発行キュー４０８に移動する。オペレーションの関連づけられたメディアリクエストあ発行キュー４０８から実行される。メディアリクエストを実行するために、スケジューラ３０２は、ＦＩＯＳスタック３００内の最初のレイヤにリクエストを渡す。

【００６７】

スケジューラ３０２以下の各レイヤは、それよりも上のレイヤから渡されたメディアリクエスト４０３を見る。適宜、レイヤはそのデータを処理する。例えば、リクエストが読み出し（read）オペレーションの場合、デアーカイバレイヤ３０１は開かれているアーカイブのコンテンツに対して与えられたパスネームを確認し、もしファイルを見つけた場合、圧縮されたデータの読み出し（read）にそのリクエストを再マッピングする。ＦＩＯＳスタック３００の各レイヤは、リクエストが最終的にハードウェアレイヤ３１０に到達するまで、処理されたあるいは処理されていないメディアリクエストを次の下位のレイヤに渡す。ハードウェアレイヤ３１０がリクエストに応答するとき、その応答はスタック３００内の各レイヤを下から上に向かって進み、各レイヤは適宜読み出したデータを処理する。例えば、デアーカイバレイヤは返されたデータは解凍されなければならないことを知っているので、応答をスタックに返す前にデータを解凍する。応答は最終的にスケジューラ３０２に戻されて完了する。

【００６８】

読み出されたデータがスタックに戻るとき、スケジューラ３０２がそれを受信して入出力オペレーションを完了キューに移動する。完了キューはアプリケーション４０１へのコールバック関数の契機となる（コールバックがアプリケーションによって設定されていた場合）。別の方法として、アプリケーション４０１は入出力オペレーションが完了したか否かを決定するためにＦＩＯＳスタック３００をポーリングしてもよい。ひとたび入出力

オペレーションが完了すると、フリーオペレーションプール４１２に移動される。フリーオペレーションプール４１２は使用されていない入出力オペレーション構造体の集合を含む。そのようなオペレーションには、クライアントアプリケーションに割り当てられていなかったリものや、クライアントアプリケーションによって使用された後、再び使用するために解放されたものが含まれる。クライアントアプリケーションが入出力リクエストを作ると、スケジューラ３０２はこのプールからクライアントに入出力オペレーションを割り当てる。フリー入出力プール４１２はスタックとして実装されてもよい。フリー入出力リクエストはフリー入出力プール４１２からポップされ、受信キュー４０４にプッシュされる。このような方法で、フリー入出力リクエストは再利用される。

【００６９】

本発明の実施の形態によれば、スケジューラ３０２は以下のようにスケジュールループを動作する。

１．入出力の完了を調べる。
２．新しい入出力リクエストを発行する。
３．入出力コールバックを発行する（もしあれば）。
４．スケジュールに挿入する（受信からスケジュールに挿入する）。
５．再度新規に発行する。
６．予想される期限の途過を調べる。
７．１に戻る。

【００７０】

各反復におけるスケジュールの挿入の数は、ある最大数（例えば１６）で制限されていてもよい。

【００７１】

本発明のある実施の形態では、補助プロセッサユニット１０５Ｂ（例えばＳＰＥ２０６）が、受信する入出力用のリクエストを、メインプロセッサ１０５ＡまたはＰＰＥ２０４に関してアトミックな交換を通じて受信キュー４０４に追加することによって、自分自身に入出力をリクエストすることができる。例えば、従来型のセルプロセッサの実装において、ＳＰＥ２０６はいずれの入出力能力も持っていない。しかしながら、受信キュー４０４が共通のデータ要素である場合、補助プロセッサ１０５Ｂは、メインプロセッサ１０５との標準的なアトミックな交換およびメインプロセッサ１０５に通知することを通じて、入出力リクエストをキューに追加することができる。

【００７２】

多くの従来技術における実装では、補助プロセッサ１０５Ｂが即時に処理するデータを必要とする場合、そのデータはメインメモリになければならない。これとは対称的に、本発明の実施の形態では、補助プロセッサ１０５Ｂはファイル入出力システムメディアスタック３００に、ハードメディアデバイス１１８やネットワーク１２７を介して得た所望のデータを取りに行かせる。

【００７３】

限定するわけではないが、例として、受信キュー４０４、完了キュー４１０、およびフリー（解放）プール４１２はアトミックなスタックであってもよい。補助プロセッサ１０５Ｂはフリープール４１２からオペレーションをポップし、オペレーションのパスとオフセットを書き込み、受信キュー４０６にオペレーションをプッシュし、さらに同期を実行し、スケジューラ３０２をウェイクアップする。補助プロセッサ１０５Ｂは入出力オペレーションの完了のための間欠的なポーリングの間に別の仕事を行ってもよい。

【００７４】

例として、セルプロセッサシステム２００内において、入出力リクエストはＳＰＥ２０６によって、次のタイプの命令シーケンスを用いてサービスされる。

```
Op = pop(FREE)
Op path = …..
Offset = …..
```

Push (op, incoming)
【００７５】
ＳＰＥ２０６によってリクエストされたデータは、ＰＰＥ２０４によってアドレス可能な場所であればどこでも送信される。ＳＰＥ２０６がロックされている場合、ＦＩＯＳ３００はＳＰＥ２０６のローカルストア（local store: LS）にデータを直接書き込むことができる。ある実施の形態では、ひとつのＳＰＥ２０６はデアーカイバに他のＳＰＥを用いて圧縮解凍するようリクエストしてもよい。これは例えばプロセッサ間の遠隔手続呼び出し（remote procedure call: RPC）を用いて実装されてもよい。この例では、ＳＰＥ２０６がＰＰＥ２０４に何かをするように要求する。より従来型の遠隔手続呼び出しではこれと逆である。
【００７６】
［プリフェッチ］
ある実施の形態では、スケジューラキャッシュ３０５（例えばハードディスクキャッシュ）は、ファイルが後に素早く読み出せるように、メディアデバイスからのファイルをストレージデバイス１１５内のキャッシュ１１７にプリフェッチするために用いられる。そのような場合、プリフェッチオペレーション４１３は発行キュー４０２に直接挿入されてもよい。プリフェッチは、ＰｏｗｅｒＰＣのＣＰＵアーキテクチャにおける「dcbt」命令のような多くの種類のキャッシュによって実装される標準的な手動プリフェッチであってもよい。本発明の実施の形態によれば、プリフェッチは上述のスケジューラループの一部としてキューに入れられ、かつ実行されてもよい。プリフェッチは、比較的高いレイテンシでかつ低いスループットを伴う、他の入出力リクエストが完了しているときにアクセスされなければならない比較的遅いソースメディア（例えばＢｌｕ－ｒａｙ（登録商標）やＵＭＤ）の仕事の手助けをする。
【００７７】
スケジューラ３０２は、メディアデバイス１１８がアイドル状態で、かつ他の入出力リクエストを邪魔しない、使用されていない時期にのみ実行するような、比較的低い優先度のプリフェッチを実装するように構成されてもよい。例えば、システム３００は遅いメディアデバイス（例えばＢｌｕ－ｒａｙ（登録商用）ディスク（ＢＤ）のような光学ディスク）と、ハードディスクのようなより高速なストレージデバイス１１５とを持っているとする。スケジューラキャッシュレイヤ３０５は、光学ディスクドライブからハードディスクへの非同期なコピーのために用いられてもよい。ファイルが後にアクセスされるとき、入り低速な光学ディスクのスピード（例えば８ＭｉＢ／ｓ）の代わりに、より高速なハードディスクドライブのスピード（例えば２０ＭｉＢ／Ｓ）で読み込まれる。
【００７８】
キャッシュプリフェッチは通常ハードディスクキャッシュで行われるが、プリフェッチはメインメモリ１０６、２０２で行われてもよい。プリフェッチオペレーションは、スケジューリングが完了した後に受信された順序でスケジュールキュー４０６の末尾に含まれてもよい。ある実施の形態では、プリフェッチオペレーションは、必要なときに遅延されてもよい。ある実施の形態では、スケジューラ３０２によって使用される性能特性式には、状態情報の一部として、キャッシュ１１７に格納されるものについての情報が含まれる。
【００７９】
プリフェッチオペレーショントは他の入出力オペレーションと同様の方法ではスケジューリングされない。プリフェッチオペレーションは、スケジュールキュー４０６が空のときのみ実行される別個のプリフェッチキュー４１３に保管される。プリフェッチオペレーションは必要があれば遅延されてもよい。そうでなければ、プリフェッチは受信された順序で実行してもよい。スケジューラ３０２はキューされたプリフェッチリクエストを保持し、入出力サブシステムが特定の長さの時間だけアイドル状態であるときにのみそれらのプリフェッチリクエストを実行する。これにより、プリフェッチが通常の入出力リクエストを邪魔するのを防ぐ。加えて、プリフェッチは単一のキャッシュブロックには限定され

ず、任意のサイズでよく、キャッシュにファイルのはじめから最後まで全体をロードすることを伝える特別な「全ファイル（whole-file）」値を渡してもよい。さらに、プリフェッチは任意のサイズでよいが、プリフェッチは、ＦＩＯＳスタック３００がアイドル状態を継続しているかを調べるためにスケジューラ３０２に戻る前には、せいぜいひとつのキャッシュブロックしか満たされないように実装されてもよい。

【００８０】

ＦＩＯＳスタック３００は、不必要なプリフェッチを回避する方法でプリフェッチを実装するように構成される。たとえば、図６に示すように、どんなブロックもキャッシュ６０１にないならば、ＦＩＯＳスケジューラキャッシュレイヤ３０５は、最初のブロックを埋めて、アイドル状態をチェックするために戻る。もしストレージデバイス１１５上のキャッシュ１１７にブロック１からＮがあるならば、（Ｎ＋１）番目のブロックがメディアデバイス１１８からのデータで埋められる。スケジューラキャッシュレイヤ３０５はアイドル状態をチェックするために戻る。このプロセスは受信キューに新しいＩ／Ｏが到着するまで繰り返される。これらの機能は任意の数のプリフェッチがキューされることを可能にするため、通常のスケジュールされたＩ／Ｏリクエストを妨げることなく、アプリケーションがメディアデバイス１１８からストレージデバイス１１５（またはメインメモリ１０６）へ大きな量のデータを高速かつ効率的にプリフェッチすることができるようになる。

【００８１】

たとえば、ゲームアプリケーションにおいて、ゲームデータはしばしば、メディアデバイス１１８によって読まれるＣＤ－ＲＯＭまたはＨＤ－ＤＶＤのようなメディアに格納されている。ゲームは、プレイヤが特定の行き先に動いていることを判定し、６０秒もすればそこに到着するであろうと計算する。その時間の間、ＦＩＯＳスタック３００は、メディアデバイスからその行き先に関してプリフェッチする。プレイヤが向きを変えて異なる行き先に向かうことを決定したなら、アプリケーションはそのプリフェッチをキャンセルする。他の実施の形態では、アプリケーションは、キャッシュ１１７にある必要とされるデータが上書きされないように防ぐために事前通知を利用する。

【００８２】

上述のキャッシュプリフェッチは、ビデオゲームのような入出力駆動型（I/O driven）のアプリケーションや、現代的なビデオゲームコンソールのような特定の入出力を要するプラットフォームにおける入出力のパフォーマンスを向上しうる。特に、ゲームデータはしばしば光学ディスクやネットワークファイルサーバのような低速なメディアに格納されるが、ゲームはハードディスクドライブのような高速なローカルストレージへのアクセスをもってもよい。

【００８３】

［補助プロセッサを用いたデータ変換］

別の実施の形態によれば，ＦＩＯＳは，メインプロセッサと，関連づけられたローカルメモリをもつ補助プロセッサとを利用するあるプロセッサアーキテクチャを活用する。たとえば，圧縮伸張または暗号解読のようなデータ変換のためにそのような実施の形態が用いられる。比較のために，ある以前のアーカイブ解凍システムは，標準的なバックグランドスレッドの中で動作する（ｚｌｉｂのような）単純な圧縮伸張アルゴリズムを用いてきた。本発明の実施の形態は，それとは対照的に，標準的なデコンプレッサ（decompressor）実装と，セル（Ｃｅｌｌ）プロセッサアーキテクチャのような関連づけられたローカルメモリをもつ補助プロセッサを含むプロセッサアーキテクチャを活用するように構成された非標準的な実装の両方を可能にする抽象的なコーデックインタフェースを含む。そのような実施の形態によれば，デコンプレッサオブジェクトが多数のフラグと制約条件を設定することが可能になり、デアーカイバ（de-archiver）３０１は，それを活用して、メモリ使用と速度を最適化する。特別な取り扱いをすることになるこれらの制約条件と環境の多くは、ゲームプログラミング、ファイルシステムＩ／Ｏ圧縮伸張、またはシステム１００のアーキテクチャにとって固有のものである。

【００８４】
ビデオゲームアプリケーションのようなあるアプリケーションは，限られた量の利用可能なメモリをもち，可能な限り少ないメモリを用いることが望ましい。一つの共通の定式化は，Ｉ／Ｏ圧縮伸張のために３つのＩ／Ｏバッファをもつことである。一つのバッファは読み取りによって使われ，２つまでの一時バッファ（一つは入力用，もう一つは出力用）はデータ変換オペレーション（たとえば圧縮伸張や暗号解読）によって使われる。一例としてこれに限定しないが，ある実施の形態は，データ変換のために関連付けられたローカルメモリ（たとえばセルプロセッサ２００におけるＳＰＥ２０６）をもつ補助プロセッサを用いることにより，一時バッファ全体を使用することを回避してもよい。
【００８５】
たとえば，圧縮伸張がメインプロセッサ（たとえば，ＰＰＥ２０４）によって扱われるなら，典型的には、圧縮されたデータを一時的に格納するために一つのバッファが必要であり、伸張されたデータを一時的に格納するために別のバッファが必要である。十分なローカルメモリ１０６Ｂをもった補助プロセッサ１０５Ｂを利用できるなら、一時的なバッファの代わりにローカルメモリ１０６Ｂを用いて同じ圧縮伸張を行うことができる。圧縮されたデータと伸張されたデータの両方をローカルメモリ１０６Ｂに格納してもよい。一例として、あるセルプロセッサアーキテクチャにおいて、ＳＰＥ２０６は２５６キロバイトの容量のローカルストアを有する。これは、圧縮された入力、伸張された出力、および圧縮伸張アルゴリズムに対するコードのために十分なメモリスペースである。比較すれば、圧縮伸張がメインプロセッサ１０５Ａ（たとえば、ＰＰＥ２０４）によって扱われるなら、圧縮されたデータは第１バッファに読み込まれ、伸張され、伸張されたデータは第２バッファに書き込まれる。その後、伸張されたデータは、第２バッファから宛先へコピーされる。
【００８６】
図７Ａおよび７Ｂは、補助プロセッサを利用してデータ変換を実装するものの中で２つの例を示す。具体的には、図７Ａにおいて説明されるように、Ｉ／Ｏリクエストを遂行する過程で、メインプロセッサ１０５Ａ上で動作するＦＩＯＳ３００が、たとえば、データ変換レイヤ３１１を介してデータ変換を起動する。符号７０２で示すように、メインプロセッサ１０５Ａは補助プロセッサ１０５Ｂをセットアップする。具体的には、メインプロセッサ１０５Ａは補助プロセッサ１０５Ｂに命じて、補助プロセッサのローカルメモリ１０６Ｂにデータ変換を埋め込むためのコード化された命令７０１をロードさせる。符号７０４で示すように、その後、メインプロセッサまたは補助プロセッサはＦＩＯＳスタック３００を検査して、メディアデバイス１１８からメインメモリ１０６にデータ７０３を転送する。補助プロセッサ１０５Ｂはその後、符号７０６に示すようにデータ７０３をローカルメモリ１０６Ｂにロードする
【００８７】
次に補助プロセッサは、符号７０８で示すように、データ７０３に変換を実行し、変換されたデータ７０５を生成し、これをローカルメモリ１０６Ｂに格納する。一例として、コード化された命令７０１の実行により、データ７０３は暗号解読または圧縮伸張される。変換されたデータ７０５は、符号７１０で示すように、メインメモリ１０６に転送される。あるいは、変換されたデータ７０５は、他の宛先、たとえばストレージデバイス１１５に配信される。Ｉ／Ｏリクエストによって特定されるすべてのデータが処理されるまで、オペレーションの前述のシーケンスをデータの次のブロックに対して繰り返してもよい。いったん入力データ７０３がローカルメモリ１０６Ｂに転送されると、そのデータは、新しい入力データまたは変換されたデータ７０５によって、メインメモリにおいて上書きされる。
【００８８】
一例として、これに限定しないが、図２で図示したようなセルプロセッサアーキテクチャを用いて、図７Ａのデータ圧縮伸張を実装してもよい。ＳＰＵプログラミングの特徴（データがＳＰＥ２０６のローカルストアＬＳにストリーム伝送され、その後、いったん処

理が完了するとストリーム返送される）は、ファイルシステムＩ／Ｏ圧縮伸張の要求と組み合わされ、Ｉ／Ｏ性能を改良する機会を与える。たとえば、圧縮伸張のためにＳＰＥ２０６を用いることにより、付加的なバッファを解放することの結果、性能が３７Ｍｂｐｓから４２Ｍｂｐｓに改善する。ソニープレイステーション２において使用されるプロセッサアーキテクチャを用いて類似した圧縮伸張を実装してもよい。そのような場合、メインプロセッサはエモーションエンジン（ＥＥ）と呼ばれ、補助プロセッサの一つは、関連づけられたローカルメモリをもつＩ／Ｏプロセッサ（ＩＯＰ）として知られる。そのような場合、デアーカイバ３０１はＥＥ上で動作し、圧縮伸張は、図７Ａに図示された方法にしたがってＩＯＰ上で実行される。あるいは、図７Ａに図示されて技術は、独立してアドレス可能なメモリをもつ２つのプロセッサが関わる分散コンピューティングアーキテクチャに対して使用することもできる。

【００８９】

前述の例は、変換レイヤによって実装されたデータ変換を用いているが、図７Ａに関して説明した方法は、メディアフィルタレイヤ３０４のいずれかを実装する際に用いることもできることに留意する。それとは対照的に、先行技術のファイルＩ／Ｏシステムは典型的には、メインプロセッサ（たとえば、セルプロセッサの例ではＰＰＵ）で実装される。

【００９０】

代替的な実装では、メディアサービス１１８から読まれたデータの宛先が低速メモリ（たとえば、ビデオランダムアクセスメモリ（ＶＲＡＭ）のようなグラフィックスメモリ１４０）であり、コーデックがその出力にランダムアクセスを実行する必要がある場合、ＦＩＯＳデアーカイバは、一時的な出力バッファを割り当てる必要がある。しかし、コーデックがランダムアクセスする必要がないなら（この例で言えば、たとえばＳＰＵ実装されたデコンプレッサの場合）、ＦＩＯＳデアーカイバ３０１は、一時的な出力バッファを割り当てることを避けて、データ７０３を圧縮伸張して直接、宛先に送ることができる。さらに、ファイルシステム圧縮伸張には「スロップ（slop）」すなわち未使用バイトが必要である。すなわち、６４キロバイト全体を圧縮するとき、ほんのわずかなバイトが必要とされる。コーデックが、圧縮されたブロックの一部を出力することができる（この例では、ＳＰＵ実装されたデコンプレッサを用いて）ことを示しているなら、ＦＩＯＳデアーカイバは一時的な出力バッファを割り当てることを回避できる。それに加えて、関連づけられたローカルメモリ１０６Ｂをもつ補助プロセッサ１０５Ｂを上述の環境の任意のすべての組み合わせで用いることができる。

【００９１】

図７Ｂは、補助プロセッサ１０５Ｂとローカルメモリ１０６Ｂを低速グラフィックスメモリ３４０とともに使用する例を示す。ＦＩＯＳ３００はメディアデバイス１１８からグラフィックスメモリ３４０（たとえば、ＶＲＡＭ）へデータを読み出す。バッファを用いたＶＲＡＭへのデータ転送は低速になる傾向がある。しかし、ＶＲＡＭデータをあるタイプの補助プロセッサのローカルメモリ（たとえば、ＳＰＥ２０６のローカルストアＬＳ）に読み込むことは問題ではない。図７Ｂに示す例では、ＦＩＯＳ３００は主としてメインプロセッサ１０５Ａ上で実装される。未変換データ７０３はグラフィックスメモリ３４０から、補助プロセッサ１０５Ｂに関連づけられたローカルメモリ１０６Ｂに転送される。ＦＩＯＳ３０１は、補助プロセッサ１０５Ｂによって実行するために変換命令７０１をローカルメモリ１０６Ｂにロードする。補助プロセッサ１０５Ｂは、命令７０１を実行することによってデータ７０３を変換されたデータに変換し、変換されたデータ７０５を生成する。変換されたデータはその後、グラフィックスメモリ３４０に転送される。この実施の形態のバリエーションの中で、未変換（たとえば暗号化または圧縮された）データ７０３は、メディアデバイス１１８から読み出され、ローカルメモリ１０６Ｂに転送される前にメインメモリ１０６のバッファに格納される。

【００９２】

［ネストしたアーカイブ］

以前のＦＩＯＳ実装には、ランダムアクセスデアーカイブが含まれており、それによっ

て、圧縮されたアーカイブのコンテンツがファイルシステム階層に仮想的に現れ、読み出しと圧縮伸張をオーバーラップさせることによって最適に効率化した方法でこれらのアーカイブからデータを読み出すことができた。本発明の実施の形態は、単一のデアーカイバレイヤ３０１をもち、性能ペナルティのないネストした（入れ子になった）アーカイブを取り扱う能力をもつことにより、このシステムを改善するものである。ここで使われるネストしたアーカイブという用語は、他のアーカイブ内に格納されるアーカイブのことを言い、他のアーカイブはさらにネストの任意の深いレベルにまで別のアーカイブに格納されていてもよい。

【００９３】

ネストしたアーカイブの支援は、ゲームＩ／Ｏおよびデータレイアウトの固有の要求を満たすために特に有益である。ネストしたアーカイブの使用は、スケジューラ３０２によって使用されるドライブモデル３０２Ａを用いることと相乗作用がある。具体的には、大きなファイルは、ドライブヘッドの動きのより良いモデルを考慮に入れる。ネストしたアーカイブの支援は、圧縮されたアーカイブフォーマットがランダムアクセスであるならば、大いに役立つ。それとは対照的に、典型的には圧縮されたアーカイブはランダムアクセスではない。たとえば、ｚｉｐ、ｇｚｉｐ、７ｚ等の圧縮フォーマットは典型的には、アーカイブの始まりから線形アクセスすることが必要である。

【００９４】

ここで「アーカイブ」という用語は、単一の大きなファイルに結合された複数のデータファイルの集合のことである。ソースデータは変更されることなく格納されるか、圧縮または暗号化のような１以上のデータ変換がデータの任意の部分に適用される。インデックス（ときには目次と呼ばれる）は、アーカイブ内のパスおよびオフセットのような、ソースファイルの少なくともある一面を記述するものであり、アーカイブとともに維持されることで、ファイルを個別に調べてアクセスすることができるようになる。「ネストしたアーカイブ」という用語は、１以上のソースデータファイルが別のアーカイブ（またはネストしたアーカイブ）であるようなアーカイブのことである。このネスト構造のおかげで、ソースデータの所与のブロックは、それに適用された１以上のデータ変換を有する。たとえば、２つのレベル（レベル１とレベル２）をもつビデオゲームアプリケーションは、「level1.psarc」と「level2.psarc」と名付けられた２つのアーカイブを含む「game.psarc」と名付けられたネストしたアーカイブを用いる。これは、ゲーム開発者がデバッグやテストの目的で単一のアーカイブ（「level1.psarc」や「level2.psarc」など）を用いることを可能にし、いつも大きな「game.psarc」というアーカイブを生成するために時間を費やさなくてもよくなることで開発を容易にする。

【００９５】

開発者はネストしたアーカイブを生成することができるが、従来のＩ／Ｏリーダは、複数のレベルをもつアーカイブを読むことができないか、効率的には読むことができない。これを克服するために、アーカイブは図８に示すように構成される。特に、外側のアーカイブ８００は、ゲームの個別のレベルに対するデータをもったファイルが含まれる内側のアーカイブ８０１、８０２を含む。一例として、これに限られないが、内側のアーカイブ８０１、８０２は、そのレベルで利用可能な地図、武器やそのレベルで出会う敵のような特定のゲームレベルに関連するデータを含む。内側のアーカイブ８０１、８０２に対するデータは、個別に圧縮され、共通するデータ８０３とともにバンドルされる。外側のアーカイブ８００内のファイルは、ブロック単位で圧縮される。各アーカイブは、デアーカイバが、内側のアーカイブ８０１、８０２における任意の場所を含めて、外側のアーカイブ８００内の任意の場所の任意のデータブロックを見つけることを可能にするインデックスを含む。インデックスは、アーカイブに対する目次と等価なものとみなすことができる。

【００９６】

ＦＩＯＳ３００がアーカイブにある特定のファイルに対するリクエストを発行するとき、リクエストはRequest: /game/level2/map(1000,10)のような形になる。「/game/level2/map」の部分はパスであり、特定のファイルを見つけるために使われる。「(1000,10)」

の部分はリクエストのオフセットと長さを特定する。外側のアーカイブ８００またはその一部が読み出されるとき、デアーカイバ３０１は利用可能なファイルのリストを考慮する。このリストは、どのアーカイブを開いているかに依存して異なる。たとえば、外側のアーカイブ８００も内側のアーカイブ８０１、８０２も開いていないなら、リストは次のようである。

/game.psarc

外側のアーカイブ８００を開いているなら、リストは次のようである。

/game (folder)
/game/level1.psarc
/game/level2.psarc
/game/common.dat

内側のアーカイブを開いているなら、リストは次のようである。

/game/level1/map
/game/level1/weapons
/game/level1/enemies
/game/level2/map
/game/level2/weapons
/game/level2/enemies
/game/common.dat

【００９７】

各アーカイブ８００、８０１、８０２は、コンテンツのパス（または部分的なパス）のハッシュのリストをもつ。ハッシュリストは、アーカイブに対する目次（table of contents；ＴＯＣ）を生成するために使われる。たとえば、外側のアーカイブ８００に対する目次は次のように構成される。

TOC: hash (/level1.psarc)
hash (/level2.psarc)
hash (/common.dat)

ここで、hash()は括弧内の量のハッシュのことである。任意の適切なハッシュ、たとえばＭＤ５ハッシュを用いることができる。

【００９８】

同様に、レベル１の内側のアーカイブ８０１に対する目次は次のように構成される。

TOC: hash(/map)
hash(/weapons)
hash(/enemies)

【００９９】

図８に示されたタイプのネストしたアーカイブが使われる場合、デアーカイブ３１１は、それらを取り扱うように構成される。一例として、デアーカイバ３１１は図９に示したアーカイブ解除方法９００を実装する。この方法９００は、符号９０２に示すように、アーカイブにおける初期ルックアップで始まる。一例として、これに限らないが、初期ルックアップは、探しているデータに対する初期パス、オフセット、長さ、および要求されるデータ変換の順序リストを返す。具体的には、Ｉ／Ｏリクエストは次のようである。

(/game/level2/weapons, offset 0, length 50000, none)

このファイルに対する初期ルックアップは次を返す。

(/game/level2.psarc, offset 20000, length 30000, zlib decompress)

【０１００】

初期ルックアップの後、デアーカイバ３１１は、符号９０４で示すようにネストを解消する。ネストを解消する際、デアーカイバ３１１は、アーカイブ８０２が別のアーカイブ内に格納されているかどうか、それは圧縮されて格納されているか、圧縮されないで格納されているかを判定しようとする。

【０１０１】

ネストを解消する際、デアーカイバ３１１は、探しているデータに対するパス、オフセット、長さ、および要求されるデータ変換の順序リストを返す。たとえば、パス/game.psarc、オフセット120000、長さ30016、暗号解読＋ｚｌｉｂ圧縮伸張のような形式をもつ。スタックの制限を避けるために再帰よりもループでネストを解消する。ネストがいったん解消されると、メディアデバイス１１８からデータが転送される。一例として、これに限られないが、デアーカイバ３１１は、符号９０５で示すようにブロック単位でデータを繰り返し読み出して変換する。具体的には、デアーカイバは、符号９０６で示すようにデータのブロックを読み出し、その後、符号９０８で示すように読み出されたブロックにデータ変換を適用する。ネストしたアーカイブの複数のレベルが単一のインタリーブした読み取り－変換ループに解消され、もっとも外側のアーカイブからブロックを読み出すことで、ソースメディアを最大限効率的に利用する。各ブロックが読み取られ、変換された後、符号９１０に示すように、望ましい部分が適当な宛先、たとえば、メインメモリ１０６、ストレージデバイス１１５または他の場所にコピーされる。ブロックの望ましい部分は任意のサイズであり、たとえば、１バイトのように小さくても、ブロック全体であってもよい。

【０１０２】

［オーバレイレイヤ］

上述のように、ＦＩＯＳスタック３００におけるメディアフィルタレイヤ３０４は、オーバレイレイヤ３０７を含む。本発明の実施の形態によれば、オーバレイレイヤ３０７は、ファイルシステムレベルにおいてファイルとディレクトリの任意のオーバレイができるように構成される。一般的なコンセプトは、より一層柔軟性があるという点を除けば、Ｕｎｉｘ（登録商標）のユニオン（union）マウントと比較することができる。典型的なユニオンマウントは、あるファイルシステムにおけるディレクトリを別のファイルシステムの全内容とマージすることにより実現される。オーバレイレイヤ３０７は同様のことをするが、もっと柔軟であり、たくさんの特徴がある。たとえば、オーバレイレイヤ３０７は、ディレクトリ、ファイルまたはファイル内のバイトのレンジの粒度で動作する。ディレクトリは隠され、そのコンテンツはオーバレイレイヤ３０７を用いて置き換えられる。さらに、オーバレイレイヤ３０７は、プライマリデータソースから要求されるファイルをセカンダリデータソースからの他のファイルで置き換えるために使われる。セカンダリデータソースは、ファイルまたはディレクトリであり、それは、プライマリデータソースと同じメディアまたは別のメディア上にある。さらに、第２データソースは仮想的なデータソースであり、データがコールバックによって指定されてもよい。この特定の文脈において、「コールバック」という用語は、他のコードへの引数として渡される実行可能なコードのことである。

【０１０３】

さらに、オーバレイを生成するアプリケーションは、オーバレイがオーバレイレイヤ３０７によってどのようにして、いつ適用されるべきかの基準を特定する。たとえば、アプリケーション１０３が、ディレクトリＢをディレクトリＡ上にオーバレイしているならば、基準は、「オーバレイ」＝最初にＢ内のファイルを探す、「アンダーレイ」＝最初にＡ内のファイルを探し、次にＢ内のファイルを探す、「より新しい」＝両方の場所にファイルが存在するなら、より最新の更新日をもつ方を使う、「より古い」＝両方の場所にファイルが存在するなら、より古い更新日をもつ方を使うなどである。

【０１０４】

オーバレイレイヤ３０７は、特定のソースから別のソースへのＩ／Ｏリードに対するリクエストの任意のマッピングを可能にするように構成される。たとえば、メディアデバイス１１８から特定のファイルを読み出すリクエストは、ストレージデバイス１１５などにある対応するファイルにマップされる。ファイル内の特定のバイトから全体のディレクトリまでの範囲のそのような特徴データを用いていることは、隠されているか、入れ替えられている。どのディレクトリ、ファイル、またはバイトを置き換える必要があるかを特定することにより、オーバレイレイヤ３０７は、特定のＩ／Ｏリクエストを処理するために

ＦＩＯＳスタック３００によってなされなければならない仕事量を最小化することができる。

【０１０５】

オーバレイレイヤ３０７は、たとえば、ソフトウェアおよび／またはデータに対するアップデートを実装するために使われる。多くの先行技術のアップデートシステムでは、実行可能なリンク可能ファイル（executable linkable file；ＥＬＦ）がメディアデバイス１１８からストレージデバイス１１５にコピーされ、更新されたコードまたはデータでパッチが当てられる。本発明の実施の形態を用いれば、これとは対照的に、アプリケーション１０３が、メディアデバイス１１８からのデータや、ストレージデバイス１１５のような他の場所からの他のバイトレンジを用いて、あるバイトレンジを埋めるようにＦＩＯＳ１０１に指示する。したがって、パッチを当てるために全体のファイルをコピーする必要がない。その代わり、パッチを当てることは、Ｉ／Ｏシステムレベルで実装される。

【０１０６】

たいていの先行するパッチの実装は、ファイル全体を置き換えることにもとづいている。これは実装するにはもっとも単純な方法であり、したがって、もっとも普通の方法によるものである。オーバレイレイヤ３０７によって実装される強化したパッチは、それとは対照的に、圧縮され、かつ、暗号化されたアーカイブ内でパッチを当てるといった、より複雑で有益なパターンを可能にする。

【０１０７】

［速度エミュレーションレイヤ］

本発明のある実施の形態では、速度エミュレーションレイヤ３１５は、低速Ｉ／Ｏデバイスの性能をエミュレートするために開発過程でより速いストレージの速度を低下させるために用いられる。速度エミュレーションレイヤ３１５は、低速Ｉ／Ｏデバイスのシーク時間、スループット、および他の性能特性のモデルを利用する。たとえば、速度エミュレーションレイヤ３１５は、ストレージデバイス１１５（たとえばハードディスクドライブ）を低速化して、ＨＤ－ＤＶＤやブルーレイ（登録商標）ドライブのようなメディアデバイス１１８の低速のシークタイムおよび／またはスループットをエミュレートする。

【０１０８】

本発明の実施の形態にしたがって、速度エミュレーションレイヤ３１５は、以前のセッションの間に集めたドライブ性能データを将来のセッションに与えることによって実装される。上述のように、スケジューラ３０２によって使用される性能モデルは、スループット、レーザの小刻みな動き、リードヘッドの動き、レイヤの変更、およびリクエストのオーバーヘッドのような複雑なドライブの特徴を適宜モデル化するように構成される。速度エミュレーションレイヤ３１５はこの情報を用いて、特定のメディアデバイスの高度に正確なエミュレーションを提供する。このエミュレーションは、ランタイム性能にもとづくため、シーク時間のようなより複雑な詳細を無視してスループットを単純に合わせようとする単純かつ素朴なエミュレーションレイヤよりもずっと正確である。

【０１０９】

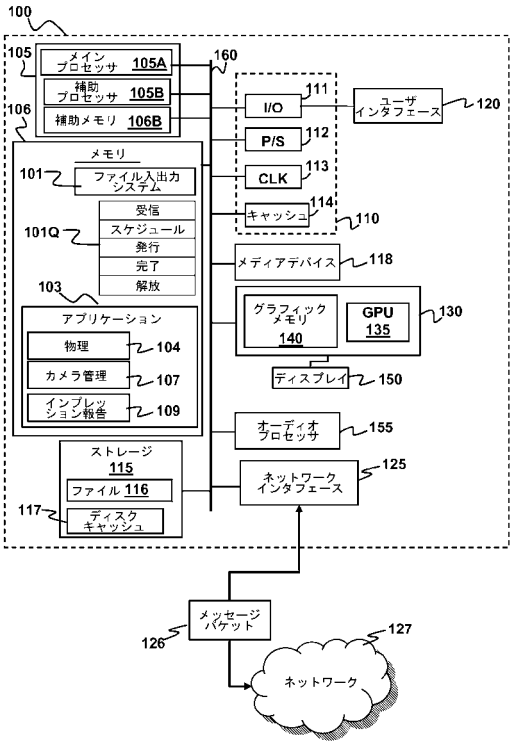
本発明の実施の形態は、かなりの量のＩ／Ｏを利用するアプリケーションとシステムにおける改良されたＩ／Ｏ性能を提供する。上述のように、本発明の実施の形態は、ビデオゲームアプリケーションおよびビデオゲームシステムにおいて特に有益である。しかしながら、本発明の実施の形態はそのようなアプリケーションやシステムに限定されるものではない。

【０１１０】

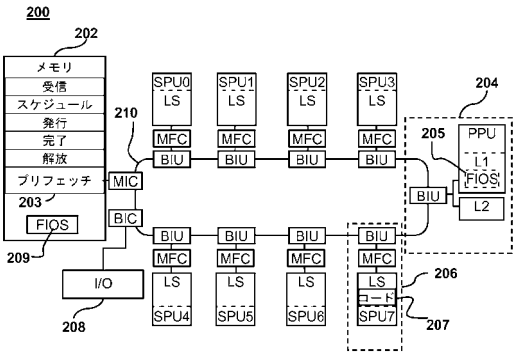
本発明の好ましい実施の形態を完全な形で説明してきたが、いろいろな代替物、変形、等価物を用いることができる。したがって、本発明の範囲は、上記の説明を参照して決められるものではなく、請求項により決められるべきであり、均等物の全範囲も含まれる。ここで述べた特徴はいずれも、好ましいかどうかを問わず、他の特徴と組み合わせてもよい。請求項において、明示的に断らない限り、各項目は１またはそれ以上の数量である。請求項において「～のための手段」のような語句を用いて明示的に記載する場合を除いて

、請求項がミーンズ・プラス・ファンクションの限定を含むものと解してはならない。

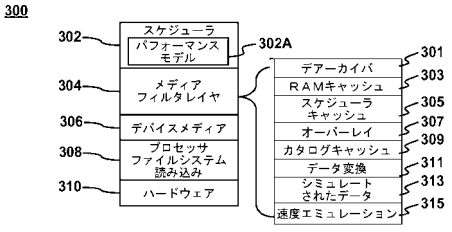
【図１】



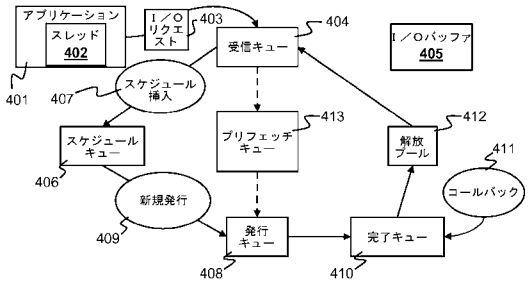
【図２】



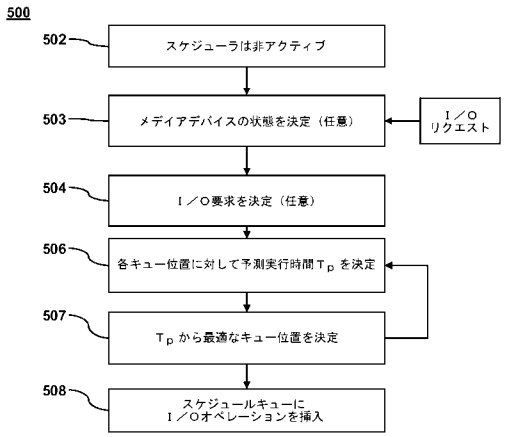
【図３】



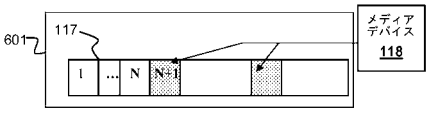
【図４】



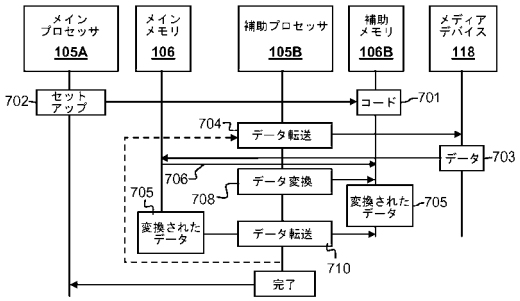
【図５】



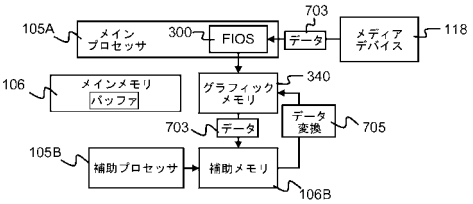
【図６】



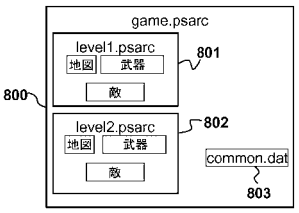
【図７Ａ】



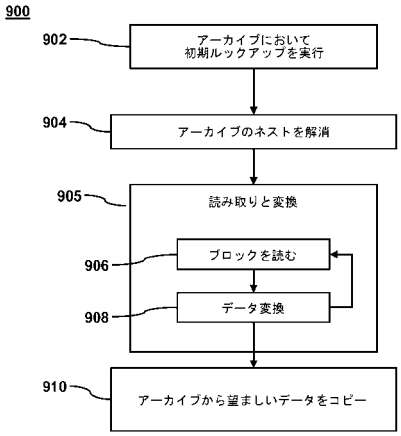
【図７Ｂ】



【図８】



【図９】



フロントページの続き

(72)発明者　ラーナー、エドワード、アダム
アメリカ合衆国、カリフォルニア州　９４４０４、フォスター・シティー、セカンド・フロアー、イースト・ヒルスデイル・ブルバード　９１９
(72)発明者　ミカル、ロバート、ジェイ.
アメリカ合衆国、カリフォルニア州　９４４０４、フォスター・シティー、セカンド・フロアー、イースト・ヒルスデイル・ブルバード　９１９

審査官　篠塚　隆

(56)参考文献　特開２００８－６５４７８（ＪＰ，Ａ）
特開平８－２１２０９０（ＪＰ，Ａ）
特開２００１－３４４０７３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ９／４８
３／０６
１３／１０