【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成21年7月16日(2009.7.16)

【公表番号】特表2006-503370(P2006-503370A)
【公表日】平成18年1月26日(2006.1.26)
【年通号数】公開・登録公報2006-004
【出願番号】特願2004-544839(P2004-544839)
【国際特許分類】
Ｇ０６Ｆ　1/26　(2006.01)
Ｇ０６Ｆ　12/00　(2006.01)
【ＦＩ】
Ｇ０６Ｆ　1/00　３３５Ｃ
Ｇ０６Ｆ　12/00　５１４Ｋ
Ｇ０６Ｆ　12/00　５３１Ｒ
Ｇ０６Ｆ　12/00　５４５Ｂ

【誤訳訂正書】
【提出日】平成21年4月8日(2009.4.8)
【誤訳訂正１】
【訂正対象書類名】明細書
【訂正対象項目名】全文
【訂正方法】変更
【訂正の内容】
【発明の詳細な説明】
【発明の名称】利用者データの永続的な格納を提供するための方法およびシステム
【発明の詳細な説明】
【０００１】
[発明の分野]
この発明はパワーが故障した時のデータ記憶に関する。より明確には、この発明は、第２の電源を用いたパワーが故障し、そして、パワーが復旧した時に、データが不当に上書きされないことを保証するためにプライベートリストを利用するデータ記憶に関する。
【０００２】
[発明の背景]
ファイルサーバの性能における重要な基準は、クライアント要求を処理するのにかかる時間である。優れたサーバは最も短い時間でクライアント要求に応じるであろう。ファイルシステムへの書き込み時の場合、すべての利用者データが最終的にディスクアレイに書き込まれることを保証することができるまで、サーバは、要求の承諾を許可しない。そのデータがサーバの中に揮発性メモリに保存されているなら、電力供給停止は、書き込みデータの完全な喪失を引き起こし、データ保全保証に反する全損を引き起こす。常時利用できるように、サーバを無停電電源装置に取り付けることは、費用がかかり、スペースの面で禁止されることがある。
【０００３】
この発明は、システムへのパワーが長期間の間、遮断されても、すべてのクライアントからのゆだねられた利用者データが、最終的にディスクに保存されることを保証する。それはどんなシステム規模のバッテリーデバイスも必要としない。バッテリーでバックアップされたメモリのわずかなバンクだけがシステムにおける永続的な格納に供される。
【０００４】
[発明の概要]
この発明は、ネットワークからクライアントのデータに役立つためのファイルサーバに

関係する。そのサーバはデータを保存するためのディスク手段を備える。そのサーバは、ネットワークからのデータを受信し、そして、データがディスク手段に格納される前に、データクライアントに保存されたことの通知をネットワークを通じて送るための手段を備え、その受信手段はディスク手段と接続状態にある。サーバは、データがディスク手段で保存されるまでデータを保存するためのメモリを備え、その受信手段はディスク手段と接続状態にある。そのサーバは、電気をディスク手段に供給する第１の電源と、メモリおよび、受信手段を備え、前記第１の電源は、ディスク手段、メモリおよび受信手段と電気接続状態にある。そのサーバは、第１の電源が故障した時、電気をメモリに供給する第２の電源を備え、この第２の電源はメモリと接続状態にある。

【０００５】

この発明はネットワークからクライアントのデータに役立つための方法に関係する。その方法は、第１の電源により給電されたファイルサーバで、ネットワークからデータを受信するステップを備える。データがディスクアレイに格納される前に、データがネットワークを通してサーバのディスクアレイに保存されたことの通知をクライアントに送信するステップがある。第１の電源が故障した時に、データが喪失しないように、第１の電源が故障した時に、データがディスクアレイで保存されるまで第２の電源により給電されるメモリにデータを格納するステップがある。データをディスクアレイ内に保存するステップがある。

【０００６】

添付図面では、この発明の好ましい実施例およびこの発明を実施する好ましい方法が図示される。

【０００７】

[詳細な説明]

参照する図面において、いくつかの視点中で同様の、または、同じ部分については同じ参照番号を用いている。特に図１から４を参照すると、ネットワーク12からクライアントのデータに役立つためのファイルサーバ10が示される。そのサーバ10は、データを保存するためのディスク手段14を備える。そのサーバ10は、ネットワーク12からのデータを受信し、そして、そのデータがディスク手段14に格納される前に、そのデータがネットワーク12を通じてクライアントに保存されたという受信確認（アクノレジメント）を送信するための受信手段を備え、その受信手段16はディスク手段14と接続状態にある。サーバ10は、データがディスク手段14で保存されるまでデータを保存するためのメモリ18を備え、受信手段16はメモリ18と接続状態にある。そのサーバ10は、電気をディスク手段14に給電するための第１の電源、メモリ18および受信手段16を備え、第１の電源20はディスク手段14、メモリ18、および受信手段16と接続状態にある。サーバ10は、第１の電源20が故障した時、メモリ18に電気を供給する第２の電源22を備え、第２の電源22はメモリ18と接続状態にある。

【０００８】

望ましくは、受信手段16は、クライアントからいずれかの順序でデータを受信し、そして、データを不正常な順序で受信した時、そのデータがメモリ18に格納される前に第１の電源20が故障した時、データがメモリ18に不正に上書きされることを防止する。望ましくは、そのディスク手段14はディスクアレイ24を含む。望ましくは、そのメモリ18はNVRAM26を含む。望ましくは、受信手段16は、クライアントから受け取った他の情報から書き込みデータを分離し、そのデータをNVRAM26へ送信する、要求マネージャ28を含む。

【０００９】

望ましくは、要求マネージャ28は、クライアントからの書き込み要求が受信手段16により完了される正常な順序でデータを識別するプライベートリストを有する。要求が不正常な順序で処理され、そしてデータがディスクアレイ24に書き込まれる前に第１の電源20が故障したとしても、プライベートリストは、データがメモリ18内で前記正常な順序で回復するのを保証するために要求マネージャ28によって使用される。望ましくは、受信手段16は、クライアントから要求を処理するiノードマネージャ30を含む。望ましくは、受信手

段16は、ディスクアレイ24を管理して、そのディスクアレイ24に対してデータの読み書きを行うディスクマネージャ32を含む。望ましくは、受信手段16は、ディスクアレイ24に保存されなかった、メモリ18に格納されたファイルの少なくとも一部を管理するキャッシュコントローラ34を含む。

【００１０】

望ましくは、NVRAM26は、第１の電源20に故障がある時、データの適切な再構成を保証するためにシステムメタデータへのすべての変化を追跡する、ディスクアレイ24内に配列されたログファイルのテール（末端部）を持つ。望ましくは、NVRAM26は、NVRAMバッファ36、NVRAM記述子38、および回復レジスタ40を備える。望ましくは、NVRAMバッファ36は、クライアントからデータを受け取った時、そのデータを保存し、NVRAM記述子38は、回復データに適切な情報を、関連するNVRAMバッファ36に記録し、NVRAMバッファ36のフリーリスト、および、回復レジスタ40は、フリーリストのヘッド（先頭部）とカウントを保持する。第２の電源28は、好ましくはバッテリを含む。

【００１１】

望ましくは、要求マネージャ28は、要求を特定する要求番号を、ネットワーク12からのクライアントから受信手段16で受け取られたファイルシステム要求に割り当てる。望ましくは、受信手段は要求番号状態メモリおよび他の情報を含み、そして、望ましくは、コールパラメータおよびファイル名を含み、そして、要求マネージャ28は、コールパラメタおよびファイル名を要求番号状態メモリ42に送信し、そして、その要求が処理の準備ができていることを示しているというメッセージをiノードマネージャ30へ送る。iノードマネージャ30が要求を処理する一方で、要求マネージャ28がクライアントからの要求を継続的に受信出来るように、望ましくは、要求マネージャ28およびiノードマネージャ30は、互いに独立して動作する。

【００１２】

望ましくは、iノードマネージャ30は、どんなタイプの操作が要求によって要求されるかを決定するために、要求番号状態メモリ42から要求のコールパラメータを読むことによって、要求を処理し始める。望ましくは、受信手段は、キャッシュRAMを含み、そして、キャッシュコントローラ34は、iノードマネージャ30とディスクマネージャから、要求されたデータブロックのために、キャッシュRAM44内に配置されたキャッシュ状態テーブルをキャッシュコントローラ34にサーチさせる検索メッセージ、データブロックに関連した現在のデータをディスクアレイ24からフェッチする検索メッセージ、どこに書き込みデータが位置するかを示すキャッシュポインタをキャッシュ状態テーブルに戻す検索メッセージ、キャッシュポインタを含むメッセージおよび命令(関連したキャッシュブロックの状態で何をすべきかの命令)をフェッチする検索メッセージを受け取る。キャッシュRAM44内でデータブロックを特定して、NVRAMバッファ36からの書き込みデータをコピーするために、要求マネージャ28へメッセージを送信した時、望ましくは、iノードマネージャ30は、(データブロックが変化し、ディスクアレイ24に書き込まなければならないという)変化メッセージを、キャッシュコントローラ34へ送信する。

【００１３】

望ましくは、ディスクマネージャ32は、一時的に変更を防ぐ、関連するキャッシュポインタをロックするキャッシュコントローラ34へフェッチメッセージを発行した後に、キャッシュポインタからの書き込みデータをディスクブロックに転送する。望ましくは、キャッシュコントローラ34は、開放されるべきことが必要なNVRAMバッファ36とリンクするNVRAM26開放リストを格納する。望ましくは、NVRAM記述子38は、異なったキャッシュブロック内に保存される、そのNVRAMバッファ36内のデータの第１の部分と第２の部分を追跡する。

【００１４】

この発明は、ネットワーク12からのクライアントのデータに役立つための方法に関する。その方法は、第１の電源20により給電されるファイルサーバ10にてネットワーク12からデータを受信するステップを備える。データがディスクアレイ24内に格納される前に、デ

ータがネットワーク12を通してサーバ10のディスクアレイ24内にデータが保存されたという受信確認（アクノレジメント）をクライアントに送るステップがある。第１の電源20が故障した時にデータが喪失しないように、そのデータがディスクアレイ24に格納されるまで、第１の電源20が故障した時に、第２の電源22によって給電されるメモリ18内にデータを格納するステップがある。そのデータをディスクアレイ24に格納するステップがある。

【００１５】

望ましくは、受信ステップは不正常な順序でデータを受信するステップを含み、そして、格納ステップは、データが不正常な順序で受信された、そして、データがメモリ18に格納される前に第１の電源20が故障した時、データがメモリ18に不正に上書きされることを防止するステップを含む。望ましくは、受信ステップは、サーバ10の要求マネージャ28によって、プライベートリストを維持するステップを含む。このプライベートリストは、書き込み要求がサーバ10により完了された正常な順序でデータを識別し、要求が不正常な順序で処理されたとしても、また、データがディスクアレイ24に書き込まれる前に第１の電源20が故障したとしても、メモリ18内でデータが前記正常な順序で回復されることを保証するために要求マネージャ28により用いられる。

【００１６】

望ましくは、メモリ18内にデータを保存するステップは、データをNVRAM26内に格納するステップを含む。望ましくは、受信ステップは、ネットワーク12を通じてクライアントから受け取られた他の情報から書き込みデータを分離するステップを含む。

【００１７】

望ましくは、受信ステップは、要求番号を、データに関連する要求に割り当て、そして、書き込みデータを保持するために、十分なNVRAMバッファ36をデキュー(待ち行列からデータを取り出す)させる。

【００１８】

望ましくは、サーバ10の要求マネージャ28によるステップは、フリーリストのヘッドからNVRAM26ポインタをデキューさせるステップがあり、記述子が表すNVRAMバッファ36の状態をチェックするためにNVRAM26に関する記述子を読む。望ましくは、NVRAM26に関連づけられたサーバ10のキャッシュRAM44の第１のキャッシュブロックも第２のキャッシュブロックも、"保護された"状態にない限り、要求マネージャ28により、記述子のビット状態をゼロに合わせるステップがあり、要求マネージャ28により、記述子の次のポインタフィールドを回復レジスタ40にコピーし、そして、回復レジスタ40がアップデートされる前に、第１の電源20が中断した場合、記述子の次のポインタフィールドを保有する。書き込みデータを格納するステップは、望ましくは、書き込みデータをNVRAMバッファ36に書き込み、そして、要求番号状態メモリ42内にそれぞれのNVRAMバッファ36へのバッファポインタを保存する。

【００１９】

望ましくは、要求番号が要求に割り当てられていることを要求マネージャ28によりサーバ10のiノードマネージャ30に通知するステップがあり、要求に関する書き込みデータを保持するNVRAMバッファ36を保護し、そして、iノードマネージャ30がクライアントからの要求に対して、承認をクライアントに発行することを許可する。望ましくは、保護ステップは、要求に対する書き込みデータを保持するのにどのNVRAMバッファ36が使用されているかを決定するために、iノードマネージャ30で要求番号状態メモリ42のポインタフィールドを見るステップを含み、NVRAM記述子38のために物理アドレスをそれぞれのポインタフィールドに発生させ、iノードマネージャ30でNVRAMバッファ36に記述子を書き、そして、第１および第２のキャッシュブロックを、"保護された"に設定する。望ましくは、ブロック開放不正メッセージを持つそれぞれの不正キャッシュブロックを、iノードマネージャ30から、各不正キャッシュブロックが開放される時にどのNVRAMバッファ36が開放されなければならないかを識別するキャッシュコントローラ34へ、開放するステップがある。

【００２０】

望ましくは、キャッシュブロックに関連しているあらゆるNVRAM26ポインタがクリーン

にされた後で、そのポインタを、要求マネージャ28により、開放し、そして、要求マネージャ28は、キャッシュコントローラ34を通してディスクマネージャ32からブロッククリーン応答メッセージを受け取るステップがある。

【００２１】

望ましくは、前記開放するステップは、キャッシュコントローラ34から要求マネージャ28へ、NVRAM26ポインタリストを転送するステップを含み；要求マネージャ28がNVRAM26ポインタをプライベートリストに結びつける一方、要求マネージャ28がNVRAM26ポインタを受信した時、要求マネージャ28でもって、ポインタリスト上の各NVRAM26ポインタの、各キャッシュコントローラ34により指示された、第１または第２の部分の状態を"開放する"状態に変え；要求マネージャ28でもって、キャッシュコントローラ34によって示されなかったプライベートリスト上のそれぞれのNVRAM26ポインタの第１または第２の部分のいずれが、"保護された"状態であるかを決定し;要求マネージャ28により、NVRAM26ポインタが、第１または第２の部分が、"保護された"状態にあるならばプライベートリストのヘッドに受け取られた時、または、第１と第２の部分のいずれも、"保護された"状態にないならばプライベートリストのテールに受け取られた時、そのNVRAM26ポインタを待ち行列に入れ；プライベートリストを、現在のフリーリストのヘッドに追加して；フリーリストの新しいヘッドを示すために、回復レジスタ40をアップデートし；そして、フリーリストから"保護された"状態でないすべての記述子をデキューする(待ち行列から除く)。

【００２２】

この発明の動作では、ファイルサーバは、一般用途のプロセッサおよびカスタムASICが実行する機能により分離された、一般用途のプロセッサおよびカスタムASICの分散システムである。これらの機能的なユニットの管理に不可欠なものは、サーバ内のすべてのユニットに対して32バイトメッセージを待ち行列にできるメッセージシステムである。４個の機能ユニットは、ファイルシステムデータの管理を含み、そのデータのディスクサブシステムへの転送は、要求マネージャ、iノードマネージャ、ディスクマネージャおよびキャッシュマネージャである。これらの実体の各々は、それらがファイルシステム要求を処理しながら、メッセージを互いに送ることができる。図１は、これらのユニットがいかにして、互いに適合するかを示し、それらをサポートするメモリを示す。太い実線は、クライアントからディスクアレイへのデータの流れを示す。細い点線は、データの流れを制御するための通信チャンネルを示す。

【００２３】

ファイルシステム要求がネットワークのクライアントから到着すると、そのファイルシステム要求は、直ちに要求マネージャによって処理される。新しい要求を受信すると、要求マネージャは、処理中にそれを特定するために用いられるであろう要求番号(RNUM)を割り当てる。要求マネージャは、コールパラメータ、ファイル名、および書き込みデータを切り離すために、ありとあらゆるファイルシステム要求を分析する。パラメータおよびファイル名は、RNUM状態メモリに送出される一方、コールに存在するすべての書き込みデータは、分離され、そして停電に対する保護のための不揮発性のRAMに送出される。構文解析がいったん完了すると、要求マネージャは、メッセージをiノードマネージャに送り、新しいクライアント要求に対して処理の準備ができていることをiノードマネージャに知らせる。要求マネージャおよびiノードマネージャは、メッセージ待ち行列化システムによって切り離されるので、互いから応答に対して待つことなく、自由に自身の機能を処理することができる。言い換えれば、iノードマネージャが難しくて遅いファイルシステム操作に遭遇しても、要求マネージャは、急激なクライアント要求を自由に処理することができる。要求マネージャは、システム内のプログラムされたRNUMの最大番号によってのみ制限される。

【００２４】

iノードマネージャは、ファイルシステムコマンドの中央プロセッサであり、その処理を完了するためにメッセージを様々なユニットに送る。iノードマネージャは、要求マネージャからのメッセージの受け取りから、要求が待機であることが知らされる。そして、

それは、どんなタイプの操作が要求されているかを決定するために、RNUM状態メモリからコールのパラメータを読むことによって、処理を開始する。このプロセッサは、クライアントから到着するかもしれないすべてのタイプのファイルシステム要求を扱うようにプログラムされる。処理の間、iノードマネージャは、メッセージをキャッシュコントローラに送り、iノードおよびデータブロックがディスクアレイからとってフェッチされることを要求する。そして、それは、クライアント要求の満足のために必要に応じてデータをキャッシュRAMへ送信し、またキャッシュRAMからデータを受け取るために、キャッシュコントローラから受け取るポインタを使用する。それが１つのキャッシュブロックのコンテンツを変えたなら、iノードマネージャは、メッセージをキャッシュコントローラに送り、データブロックがアップデートされ、ディスクに書き戻さなければならないことを、このユニットに知らせる。

【００２５】

キャッシュコントローラは、クライアントにより近くにあり、ディスクアレイよりはるかに少ない待ち時間を持っているメモリ内に格納されたファイルシステムの一部を管理する。それは、ファイルシステム要求を処理するとき、iノードマネージャおよびディスクマネージャからのメッセージの２つの基本的なタイプを受け取る。最初のタイプは、ルックアップメッセージであり、キャッシュコントローラに、要求されたデータブロックに対するキャッシュ状態テーブルを探させ、必要なら、ディスクからデータをフェッチして、そしてキャッシュポインタをデータが置かれた箇所に戻す。２番目のタイプは、キャッシュポインタおよび、キャッシュポインタの状態で何をすべきかの命令を含むフェッチメッセージである。異なった機能的なユニットがキャッシュエントリーで動作するとき、それらが不意に取り替えられないことを保証するために、それらは、自身に対する参照を(キャッシュメッセージを通じて)取り込む。

【００２６】

ディスク活動が要求されるときはいつも、ディスクマネージャは、キャッシュコントローラからのメッセージを扱う。参照メッセージがキャッシュミスを引き起こしたとき、キャッシュコントローラは、リードメッセージをディスクマネージャへ送る。キャッシュエントリーが変更されたとき、ディスクに書き戻されるように、キャッシュコントローラは、最新のデータが位置するキャッシュポインタでもってディスクマネージャにライトメッセージを送る。ディスクマネージャがキャッシュポインタからディスクブロックへデータを移動するために、それは、最初にフェッチメッセージをキャッシュコントローラに発行し、ポインタ上でロックを求める。ロックが許可された時、ディスクマネージャは、キャッシュポインタからのデータをディスクアレイにコピーのみをする。これは、転送が始まる前にデータブロックの内容が安定していることを保証する。ディスクマネージャは、キャッシュRAMからディスクアレイへデータを移動するためにコマンドをキャッシュコントローラに送る。それは、その決定を、潜在的な高い待ち時間のディスクの効率を最適化するように設計されたアルゴリズムの結果に基礎をおく。不正ページを追跡することは別として、キャッシュRAMからデータをいつ移動するかについてのその決定は、システムのクライアント側に起こるイベントの処理をほとんど扱わない。

【００２７】

要求マネージャがネットワークから要求を受け入れるとき、要求マネージャは、サーバ内の要求の有効期間の間に用いられるべきコンテキストIDを、前記要求に割り当てる。このIDはRNUMと参照されるであろう。RNUM状態メモリは、要求の処理の間に蓄積された情報を保持するために使用される。それは、停電後に、揮発性で、無効と認定される。RNUMのために到来するクライアントデータを保持するのに使用されるNVRAMバッファへのポインタは、この領域に格納される。

【００２８】

ファイルサーバは、クライアントの近くで低待ち時間メモリのファイルシステムの部分を維持するためにキャッシュRAMを使用する。このメモリはディスクアレイよりはるかに速いが、それは、一般に、多くのパワーを消費して、したがって、揮発性のメモリとして

実装され、電力供給停止の間、すべてのコンテンツを喪失する。キャッシュコントローラは、改善された効率のために、書き戻しアルゴリズムを使用することでキャッシュRAMを管理する。ディスクへの書き戻しを遅らすことにより、ファイルサーバは、同様の物理的な位置からのデータを、１つのより大きいは要求にグループ化し、性能を向上させる。それは、ディスクへのデータの書き戻しを延期するので、キャッシュRAMは、ファイルシステムはディスクアレイと比較して、より最新のバージョンを持つであろう。したがって、書き戻しが起きるまで、そのコンテンツはNVRAMによって保護されなくてはならない。

【００２９】

キャッシュRAMに保存された別のアイテムはNVRAM開放リストである。このメモリは、１つのキャッシュブロックのクリーン時の開放されるべき必要のあるNVRAMバッファをリンクするために、キャッシュコントローラによって使用される。最大64kのエントリーはこの機能のためにサポートされる。それぞれのエントリーは、以下の通り8バイトであって以下のようにフォーマットされている。

【００３０】

【表１】

NVRAM 開放エントリー (8 バイト)

4 バイト	フラグ： [31-2]: 準備済み [1]: 第１の部分が開放されるべき時にクリア。第２に設定 [0]: 開放リスト内で最後のエントリーの時に設定
2 バイト	NVRAM バッファポインタ このNVRAMに関係したアクチュアルのNVRAM バッファポインタ 開放エントリー
2 バイト	次の開放リストのエントリー このエントリーがリスト上で最後のものでないとき、この領域は開放リスト上の次のエントリーへのポインタを含む。

【００３１】

キャッシュコントローラが、開放リストを必要とする第１のNVRAMポインタの通知を得たとき、キャッシュコントローラは、それのためのエントリーを作成し、それがリスト上の最後であると指定するためにそのフラグ(０)をセットする。リストに置かなければならないあらゆるその後のポインタは、リストのヘッドに待ち行列に入れられ、前のヘッドを指す。

【００３２】

クライアントからの書き込みデータ帯域幅に耐えることの出来るいずれかの低パワーメモリ技術を用いることでこのシステムの不揮発性RAM(NVRAM)を実施できる。停電の時にバッテリバックアップへスイッチされるように、そのパワー・システムは、システムの残りの部分から切り離さなければならない。そのNVRAMには、２つの別々の目的がある。まず最初に、それはファイルシステムメタデータへのすべての変化を追跡するログファイルのテールを保持する。メタデータが変更されるとき、サーバ故障の場合に適切な復旧を保証するために古いデータおよび新しいデータの処理がログファイルに保持される。ディスクアレイ内でファイルの過半数が保たれる一方、最新の処理がログファイルのテールに追加されるとき、その最新の処理はNVRAMに維持される。一旦、十分な処理が蓄積されると、そのテールは、大きくかつ効率よく瞬時にディスクに移動される。高速メモリ内にログファイルのテールを持つことにより、そのファイルシステムは最も効率的にそれを維持することができる。

【００３３】
NVRAMの２番目の使用は、クライアントからの書き込みデータをディスクアレイに書く前に、一時的に記憶するためのものである。このタスクを達成するために、３つの別々の構造がNVRAM内に維持される。これらは4kBのNVRAMバッファ、32バイトのNVRAM記述子および4バイトの回復レジスタである。クライアントから書き込みデータが到着した時、その書き込みデータを正確に格納するために、NVRAMバッファがサーバにより使用される。ディスクブロックとキャッシュRAMバッファのサイズを合わせるために、4kBのバッファサイズが選ばれた。しかしながら、選択的にサイズを決める別の可能性は、バッファを、クライアントからの利用者データの最も大きい可能な転送と等しくすること含む。更に別のアプローチは、もし多数の小さい書き込みが到着するなら、メモリの効率を増すために、小さいのと大きいサイズの双方を備える。各バッファは、電力供給停止の後に書き込みデータを回復することができるように、NVRAMバッファ内に保持された書き込みデータの詳細のすべてを指定するために、バッファに関連するただ一つの記述子を持つ。また、その記述子は、クライアント要求を処理する一方、要求マネージャが引き出すことができる、バッファのフリーリストを維持するために使用される。キャッシュからの書き込みデータでディスクブロックがアップデートされるので、NVRAMバッファは、フリーリストの最後に戻される。回復レジスタは、フリーリストのヘッドとカウントを保持するために用いられる。図２はファイルサーバが、クライアントからの書き込みデータを保持するために、NVRAMバッファをいかにして追跡するかを示す。
【００３４】
そのサーバは、64kのNVRAMバッファをサポートするように設計されているので、16ビットのポインタは、各バッファとその記述子を区別し参照するために用いられる。それが与えられると、回復レジスタは単に、そのNVRAMフリーリストのヘッドにある記述子への16ビットのポインタ、および、前記フリーリスト上の記述子の16ビットのカウント値を含む。そのNVRAM記述子のコンテンツは、ここで提案された強固な格納アーキテクチャーの操作に非常に重要である。このフィールド(情報を表す単位)は、書き込みデータを受け取ると初期化され、そして、回復手順の間、検査される。NVRAM記述子の内容は表２に示される。
【００３５】

【表２】

NVRAM 記述子（３２バイト）

12 バイト	FID: バッファ中のデータが属するファイルID
8 バイト	オフセット ファイル中のデータの開始バイトのオフセット
8 バイト	シーケンス番号: 回復しなければならないデータをバッファが含むとき、この分野は到着したとき要求に割り当てられたシーケンス番号を含む。 これは回復の間、正当性を保証するために到来する書き込みをオーダーするために使用される。
4 バイト	状態フラグ(4ビット): [3-2]: 第１のキャッシュブロック状態ビット [1-0]: 第２のキャッシュブロック状態ビット '00': *不正* ．不正コンテンツ '01': 未決定 '10': *開放* ． 記述子がフリーリスト内に無いならコンテンツを回復 '11':*保護される*. 全ての状態でコンテンツを回復
	データ長（１２ビット): NVRAMバッファ中の有効データの量
	次のポインター（２バイト): NVRAM バッファがフリーリスト内にあるとき、この領域はリスト上の次のNVRAMバッファへのポインターを含む。ゼロの値はリストの終わりを指定する。

【００３６】

記述子にみられるように、それぞれのNVRAMバッファには、保護の異なった状態にあるかもしれない第１および第２の部分がある。この概念に関する説明のために、以下の例を考察する。クライアント要求が8kBの書き込みデータを担うサーバに到着するなら、前記データの受信のために、二つの4kBのNVRAMバッファを準備しなければならない。前記書き込みデータがそのファイル内に1kBの開始用のオフセットを持つなら、完全な8kBのデータは、結局、1kB～9kBをオフセットするために書かれる。すべてのキャッシュブロックが4kB境界を始めるので、書き込みデータがディスクアレイに戻される前に、この書き込みデータは３つの個別のキャッシュブックに書かれる。図３に示されるように、書き込みデータは、NVRAMバッファ１および2に書かれる。処理の間、キャッシュポインタ100、200および300が得られ、そして書き込みデータを受け取るために用いられる。キャッシュブロックの4kB境界のために、NVRAMバッファは実際には第１および第２の２つの別々の部分に分割される。キャッシュブロック100は、NVRAMバッファ１の第１の部分からのデータを保持するが、キャッシュブロック200は、NVRAMバッファ１の第２の部分およびNVRAMバッファ２の第１の部分からのデータを保持し、以下同様である。キャッシュブロック300が他のものより前にクリーンにされた場合、システムは、NVRAMバッファ２の第２の部分だけを開放してもよいことを認識し、そのバッファをフリーリストから離し、更に第１の部分に対し保護を維持する。

【００３７】

クライアントからの書き込み要求の存続期間の間に、NVRAM構造の維持についての操作の３つの段階を以下述べる。以下で概説された各段階は、２つの小区分を持つ。第１の小区分は、段階を処理するとき、サーバが直面するステップについて説明する。第２の小区

分は、段階の間に起こるかもしれない電力供給停止の衝撃の分析を提供する。この供給停止分析は、NVRAM構造の完全さを約束することなく、いつでも停電できることを保証するのに必要である。

【００３８】

段階１：ユーザーデータの到着

ユーザーデータ到着の説明

ファイルサーバがクライアントから書き込み要求を受けた時、それは、要求マネージャによって専用要求番号(RNUM)に割り当てられて、分析されて、分配される。要求が到着したことを、iノードマネージャに通知する前に、全体の書き込みデータコンテンツを保持するために、最初に、十分な4kBのNVRAMバッファをデキューしなければならない。書き込み要求の最大サイズが8kBであるので、２つ未満のバッファが要求毎に必要となる。

【００３９】

要求マネージャがフリーリストのヘッドからのNVRAMポインタをデキューするとき、それは、最初に、それが表すバッファの状態をチェックするために、その記述子を読むであろう。第１および第２のキャッシュブロックのいずれもが"保護されていない"状態なら、要求マネージャは、記述子において状態ビットをゼロにする。それは、フリーリストの新しいヘッドであるので、それは、その後、新たにデキューされた記述子の次のポインタフィールドを、NVFREE_HEAD回復レジスタにコピーする。次のポインタフィールドは、NVFREE_HEADレジスタをアップデートする前にパワーを中断するといけないので、デキューされた記述子内に保有される。新たにデキューされた記述子が完全にゼロにされたなら、フリーリストは、NVFREE_HEAD回復レジスタをアップデートするまで、無効の瞬間があるであろう。NVRAMバッファを"無効"状態に置くことは、フリーリスト外のNVRAMバッファが回復されないことを保証する。

【００４０】

要求マネージャが、"保護された"に設定された第１または第２のキャッシュブロックを見つけると、それは、ポインタが完全に開放されず、フリーリストからそれを全体で取り除くことを知るであろう。それが記述子をフリーリストから移動した時、それは、回復を防ぐために"開放する"状態内のすべての部分をも"無効"に変える。その結果、記述子の次のポインタフィールドは、上述のように、NVフリーヘッド回復にコピーされるであろう。

【００４１】

要求マネージャは、次に、利用者データを新たに取得されたバッファに書き込み、そして、RNUM状態メモリ内の事前に定義された位置のバッファにポインタを収納するであろう。このクライアントは、データが遂行されたという承認を受けていないので、これらのバッファはまだ保護されていない。どんなフリーのNVRAMバッファも利用可能でないなら、十分にフリーになるまで、要求マネージャのクライアントインターフェースは完全に停止するであろう。

【００４２】

利用者データ到着の供給停止分析

利用者データ到着の間のNVRAM構造への唯一の変更は、フリーリストのヘッドからのNVRAMバッファをデキューすることである。このプロセスの間、極めて特別なオーダーで２つの構造だけをアップデートする。まず最初に、記述子の状態フラグは、"開放する"から"無効"に変化し、どんな"保護された"の部分も変化されない。記述子の他のフィールドが全くこの操作の間に変化しないので、アップデートの間の電力遮断はその状態に全く影響を持たないであろう。"無効"か"開放する"であるフリーリスト上のエントリーは、定義により決して回復されないであろう。NVフリーヘッド回復レジスタがアップデートされる前で、記述子がアップデートされた後で、パワーが無くなると、回復手順は、まだフリーリストのヘッド上の最近にデキューされた記述子を単に見つけ、それを回復しないであろう。次のポインタフィールドが変化しないので、フリーリストの保全はまだ維持されている。

【００４３】

段階２：利用者データ収容
利用者データ収容の説明
要求マネージャが、新しいRNUMが書き込み要求に割り当てられたことをいったんiノードマネージャに通知すると、iノードマネージャは、応答をクライアントに送り返すべきである時間になるまで、このコールのためのその手順を実行する。この応答を送る前に、iノードマネージャは、確実にバッファコンテンツを保護しなければならない。このタスクを達成するために、この要求に対する書き込みデータを保持するのにどのバッファが用いられたかを見つけるために、RNUM状態メモリのポインタフィールドを最初に探す必要がある。書き込み要求が4kBかそれ未満なら、１個のポインタだけが有効である。iノードマネージャは、32バイトのNVRAM記述子に対する物理アドレスを発生させるのに各ポインタを使用しなければならない。iノードマネージャは、メモリに記述子の全体のコンテンツを書き、第１および/又は第２の状態が保護されるために設定しなければならない。さらに、それは、それが保護した最後のNVRAMバッファよりさらに１つ以上の8バイトのシーケンス番号を割り当てなければならない。このポイントで、バッファが保護されて、iノードマネージャが書き込み要求に対する応答を発行することが受入れられる。
【００４４】
その応答をクライアントに送ったとき、iノードマネージャは、ブロック開放不正メッセージと共に各不正キャッシュブロックをキャッシュコントローラに開放するであろう。このメッセージでは、プロセッサは、ブロックがクリーンにされたとき、どのNVRAMバッファを開放しなければならないかを指定しなければならない。最大２つのNVRAMバッファが、iノードマネージャにより、１つのキャッシュブロックに関連している必要があるので、ブロック開放不正コールは、これらのポインタを通過させるために十分なスペースを備える。さらに、このコールは、NVRAMバッファの第１または第２の部分がこのキャッシュブロックに関係するかを指定するために、NVRAMポインタあたり１ビットを提供する。
【００４５】
新しいNVRAMポインタが割り当てられるキャッシュブロックが既に他のもののトラックを維持しているかもしれないことに注目することは重要である。したがって、キャッシュコントローラは、キャッシュブロックがクリーンにされたとき、開放を必要とするNVRAMポインタのリストを保持しなければならない。これを達成するために、キャッシュコントローラは、NVRAM開放リストと呼ばれるキャッシュRAMの領域で8バイトのエントリーの別々のプールを維持する。ブロック開放不正コールは、到達し、キャッシュコントローラはこのメモリ内のNVRAMバッファのただ一つのリンクされたリストを形成する。NVRAMバッファがあるのと同じくらい多くのエントリーがこのリストにあるに違いない。このリストはキャッシュRAM内に保持されるので、NVRAM開放リストと呼ばれる。
【００４６】
段階３：利用者データ開放
利用者データ収容の供給停止分析
この段階の間、書き込みデータを保護するために、iノードマネージャがNVRAM記述子を満たしたとき、NVRAM構造への唯一の変化が起こる。記述子の状態ビットは、他のすべてのフィールドが最初に書かれるまで"保護された"への状態変化が起こらないのを保証するために、最終的なワード内に置かれる。この段階の始めで、記述子が"無効"状態にあるので、すべてのフィールドが有効な情報を含むまで、記述子は回復の候補でなくなるであろう。
【００４７】
利用者データ開放の説明
なぜ、NVRAMバッファの開放リストがキャッシュバッファの背後に維持されるのか、また、すべての時間で電力遮断を考慮しているとしても、なぜ、サーバが、このリストが任意のオーダーで作成されることを許容するのかの理由は、次に説明される。ファイルシステム上で独立して、又は同時に動作する種々のソフトウエアおよびハードウエアをサーバに持たせるために、不正常な順序で機能を完了することを可能にする十分な柔軟性がなけ

ればならない。言い換えれば、コンテンツが最終的にディスクに書き込まれた時に開放するべきNVRAMバッファのリストをキャッシュブロックが持つとする。バッファのこのリストが、1-2-5-4-3の順であったとする。数字はクライアントから到達したデータの順を表す。クライアントは５回連続で極めて迅速にファイルに書き、(要求#5からの最新の書き込みデータが実際にはキャッシュバッファの中にあっても)サーバは最後に書かれた３番目の書き込みが開放リストの最後に書くようにこれらの要求を処理することを考える。開放リストが処理される前にパワーが停止したなら、NVRAMがまだすべてで５つのNVRAMバッファを持っていて、それらを整理して、最新のデータがディスクに行くことを保証するので問題が全くないであろう。しかしながら、開放リストの最初の３つのエントリーがNVRAMから取り除かれ、その後、パワーが停止したらどうなるのか。システムが戻るとき、転送#3および#4がNVRAMにあって、それがそれらをディスクに書き込み、不当に、転送#5からの最新のデータを上書きすることがわかるであろう。プライベートリストが要求マネージャによって維持されている状態で、バッファを開放するとき、この問題は解決される。

【００４８】

より明確に、そのキャッシュコントローラは、メッセージをクリーンにするため、キャッシュブロックがいつクリーンにされるかが、ブロックの動作を通じて報告される。このコールが受け取られたとき、キャッシュコントローラは、ブロックの状態で参照されるあらゆるポインタを開放するために、要求マネージャと通信する。ファイルサーバがマルチスレッド化の処理用エンジンなので、開放されたNVRAMバッファの命令が、要求がネットワークから到着する命令と同じであるという保証は全くない。したがって、要求マネージャは、NVRAM記述子をフリーリストに置く時に、それらのNVRAM記述子をゼロにする簡潔な解決法を取ることができない。チップが不適切な記述子をフリーリストに返している間、パワーが無くなるなら、回復手順は、解放されなかったより古いバッファが実際に最新のデータを含むと考えるであろう。

【００４９】

順序づけの制限を避けるために、サーバは制御された方法でNVRAMバッファを開放する。まず最初に、全体のポインタリストは、キャッシュコントローラから要求マネージャにいずれかの順序で移動する。要求マネージャがポインタを受け取った時、それは、第１、または、第２の部分(キャッシュコントローラによって示されたように)を"開放"状態に変える一方、プライベートリストの中でそれらを結びつける。NVRAMポインタの反対の部分がまだ"保護された"であることを、要求マネージャが見つけたなら、それは、ポインタをプライベートリストのヘッドにエンキューする(待ち行列に入れる)。さもなければ、それはポインタをプライベートリストのテールにエンキューする。キャッシュコントローラが開放するべきNVRAMポインタの全体のリストを送った後に、要求マネージャは、新しいリストを現在のフリーリストのヘッドに追加し、そして新しいヘッドを示すためにNVFREEヘッド回復レジスタをアップデートする。NVRAMポインタを二度、フリーリストに置くことを避けるために、要求マネージャは、キャッシュコントローラからいずれかの別の"開放"を受け入れる前に、フリーリストから"保護された"のデータを持つすべての記述子をデキューしなければならない。それは、正常なデキューの手順に従うことでこれを実行する。

【００５０】

利用者データ開放の停電分析

開放プロセスの間に現れるNVRAM構造の第１の変更は、プライベートリストの作成である。ポインタがプライベートリストのヘッドかテールに書かれているときはいつも、状態ビットは、次のポインタにアップデートされるのと同時に、開放することに変化する。プライベートリストの生成の間、フリーリストへの変化は全くない。したがって、プライベートリスト生成の間にパワーが失われたとき、NVRAMポインタがプライベートリストに対するものか否かに関係なく、新たにクリーンにされたキャッシュブロックの背後のあらゆるNVRAMポインタが回復されるであろう。

【００５１】

プライベートリストがフリーリストのヘッドに動かされるとき、NVRAM構造の第２の変

化が起きる。このとき、プライベートリストのテールのエントリーの次のポインターは、現在のフリーのヘッドになるように、アップデートされる。次のステップは、プライベートリストのヘッドへ向けるためにNVフリーヘッド回復レジスタをアップデートする。これは有効的に、新たに開放される各々のNVRAMポインタを、フリーな待ち行列へ移動する。NVフリーヘッド回復レジスタをアップデートする前にパワーを失うなら、全体のプライベートリストがフリーリストで見つけられないので、回復ステップは、更に全体のプライベートリストを回復するであろう。

【００５２】

パワー損失時の手順

パワーの損失時、ファイルサーバは優雅にすべてのNVRAMの活動に終わりにもたらすであろう。データをNVRAMに送るメモリコントローラは、要求マネージャおよびiノードマネージャからのどんな新しい要求も受け入れるのを止め、そして、それを持っているすべての書き込み位置バッファも洗い流すであろう。書き込みがメモリコントローラにより受け入れられた後で、しかし、それが実際にNVRAMになされる前に、要求マネージャがメモリコントローラにより受け入れられるので、その洗い流しのステップは重要である。そのメモリコントローラは、その後、電源をバッテリバックアップに移し、そしてメモリを低パワーモードに置く、シャットダウン手順が続くであろう。

【００５３】

回復手順

システムでパワーが復帰するとき、全体のサーバが新しい要求を受け入れる準備ができるまで、要求マネージャはディスエーブル状態で留まる必要がある。正常な初期化ルーチンが完了した後、どのバッファがデータを保護するか、そして、それらがどんな順序で受け取られたかを決定するために、iノードマネージャは、NVRAM内のあらゆる記述子を通じて実行することが期待される。

１．"開放する"または"保護された"の状態にデータの１部分を持っているすべてのポインタのために、全体のNVRAM記述子空間をスキャンさせる。これらの部分を"保護される"のリストの中に置く。

２．NVフリーヘッド回復レジスタを読む。

３．次に、"保護された"のリストからどんな"開放"の部分も取り除きつつ、バッファのリンクされたリストに従う。

４．全体の"保護された"のリストをスキャンし、そして、シーケンス番号に従って、それらを順序づける。

【００５４】

"保護された"のデータの最終的なリストがいったん決定されると、iノードマネージャは、対象のディスクブロックの現在のコンテンツを、ディスクアレイからキャッシュRAMに読み込ませるために、キャッシュコントローラと協力する。次に、記述子で示されたバイトオフセットでスタートして、NVRAMバッファ中に格納されたデータでそれらのキャッシュブロックを上書きする。すべてのデータがいったんキャッシュブロックにコピーされると、ディスクにそのデータを書き戻し、ファイルシステムを最新版にアップデートする。このとき、そのサーバは、クライアントから新しいネットワークトラフィックを受けることが可能にされてもよい。

【００５５】

図５に示したような好ましい実施例では、32Gbpsバックプレーン80上のクライアントからの要求は、構造インターフェース81によって受け取られる。構造インターフェース81は、バックプレーン80から受け取ったデータのフォーマットを、要求待ち行列マネージャ82が動作できるフォーマットに変換する。その構造インターフェース81および要求待ち行列マネージャ82は、要求マネージャを形成する。要求待ち行列マネージャ82はDRAMコントローラ83にデータを書き、DRAMコントローラ83は次にデータを外部の共有されたDRAMメモリ92へ、最終的にNVRAMに伝える。また、要求待ち行列マネージャ82は、情報をZBTコントローラ84に供給し、そのコントローラは、外部の、共有されたSRAMメモリ85とのインターフ

ェースとして機能する。
【００５６】
要求待ち行列マネージャ82は、メッセージ待ち行列管理マネージャ85と通信し、メッセージ待ち行列管理マネージャ85はシステムの様々な部品の間のすべてのメッセージのためのハブとして機能する。メッセージ待ち行列マネージャ85からのメッセージは、外部のiノードマネージャとディスクマネージャと通信する高速経路の上位インターフェース86に提供する。ディスクマネージャはBMAPリードプロセッサおよびBMAPライトプロセッサから形成される。ディスクマネージャは、仕事量をよりよく分配するためにリードプロセッサとライトプロセッサに分解される。
【００５７】
メッセージ待ち行列マネージャ85は、キャッシュコントローラと通信し、このキャッシュコントローラは次ぎに、外部のキャッシュ状態DRAM95とインターフェースする。メッセージ待ち行列コントローラ85およびキャッシュデータDRAMコントローラ93は、外部記憶装置に保存されたデータに関する情報を得るためにディレクトリマネージャ87と通信する。また、メッセージ待ち行列コントローラ85は、システムのメインテナンスのために外部のブレード制御プロセッサ89と通信する低速経路インターフェース88と通信する。ファイバーチャンネルマネージャ90は、ディスクおよびiノードに対してデータを転送するために、チップ94から外部へ広がるファイバーチャンネル91で通信する。また、要求待ち行列マネージャ82は、キャッシュデータメモリ96とインターフェースするキャッシュデータDRAMコントローラ93と通信する。
【００５８】
この発明は図示の目的のための以上の実施例で詳細に説明したが、そのような詳細は単に前記目的のためのものであり、以下のクレームで述べられたことを除き、この発明の本旨および範囲からそれることなく、当業者により種々の変形が可能であることが理解されるであろう。
【図面の簡単な説明】
【００５９】
【図１】ファイルシステムの機能ユニットのブロック図を示す。
【図２】NVRAMのデータバッファ構造を示した概略図を示す。
【図３】NVRAMバッファの第１/第２の部分を視覚化した概略図を示す。
【図４】この発明のファイルシステムの概略を示す。
【図５】この発明のファイルシステムの好ましい実施例の概略を示す。
【符号の説明】
【００６０】
10：ファイルサーバ
12：ネットワーク
14：ディスク手段
16：受信手段
18：メモリ
20：第１の電源
22：第２の電源
24：ディスクアレイ
26：NVRAM
28：要求マネージャ
30：iノードマネージャ
34：キャッシュコントローラ
36：NVRAMバッファ