(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2005-502957
(P2005-502957A)
(43)公表日　平成17年1月27日(2005.1.27)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 11/20	G06F 11/20	310E	5B034
G06F 12/00	G06F 12/00	533J	5B082
G06F 13/00	G06F 12/00	545A	5B089
	G06F 13/00	357Z	

審査請求 未請求　予備審査請求 未請求　（全 78 頁）

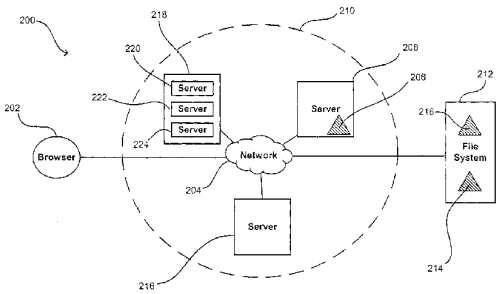
(21)出願番号　特願2003-527615(P2003-527615)
(86)(22)出願日　平成14年9月5日(2002.9.5)
(85)翻訳文提出日　平成16年3月8日(2004.3.8)
(86)国際出願番号　PCT/US2002/028199
(87)国際公開番号　WO2003/023633
(87)国際公開日　平成15年3月20日(2003.3.20)
(31)優先権主張番号　60/317,718
(32)優先日　平成13年9月6日(2001.9.6)
(33)優先権主張国　米国(US)
(31)優先権主張番号　60/317,566
(32)優先日　平成13年9月6日(2001.9.6)
(33)優先権主張国　米国(US)
(31)優先権主張番号　10/234,693
(32)優先日　平成14年9月4日(2002.9.4)
(33)優先権主張国　米国(US)

(71)出願人　500105160
ビーイーエイ　システムズ，　インコーポレイテッド
ＢＥＡ　Ｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国　カリフォルニア　９５１３１，　サン　ノゼ，　ノース　ファースト　ストリート　２３１５
２３１５　Ｎｏｒｔｈ　Ｆｉｒｓｔ　Ｓｔｒｅｅｔ，　Ｓａｎ　Ｊｏｓｅ，　ＣＡＬＩＦＯＲＮＩＡ　９５１３１　Ｕ．Ｓ．Ａ．
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100067013
弁理士　大塚　文昭

最終頁に続く

(54)【発明の名称】厳密に一回のキャッシュフレームワーク

(57)【要約】
クラスター化されたネットワーク内のオブジェクトを管理するためのシステムは、データオブジェクト（２０８）の少なくとも１つのコピーを包含するファイルシステム（２１２）を含む。本システムは、ファイルシステム（２１２）と通信するいくつかのクラスター化されたサーバを含むことができる。リードサーバが選択され、それは、ホストサーバ（２０６）を選択するための分散型合意アルゴリズムを含み、アルゴリズムのラウンドを実行中にマルチキャスティングを利用する。選択されたホストサーバ（２０６）は、ローカルキャッシュなどにデータオブジェクト（２０８）のコピーを含むことができ、クラスター内の任意の他のサーバにローカルコピー（２０８）へのアクセスを提供する。また、ホストサーバ（２０６）によってホスティングされた項目に対して為されたいかなる変更もファイルシステム（２０２）において更新することができる。ホストサーバ（２０６）がオブジェクトをホスティングすることができなくなった場合、分散型合意アルゴリズムを使用して新しいホストを選択することができる。他のサーバ（２１６、２１８



【特許請求の範囲】
【請求項１】
各々がネットワークデータソースと通信するようになった複数のネットワークサーバと、
前記複数のネットワークサーバにおけるリードサーバと、
を含み、
前記リードサーバは、前記複数のネットワークサーバからホストサーバを選択するための分散型合意アルゴリズムを包含し、
前記ホストサーバは、前記ネットワークデータソース内のデータ項目に関連したオブジェクトを包含し、それによって、該データ項目にアクセスする必要がある前記複数のネットワークサーバのいずれもが該ホストサーバ上の該オブジェクトにアクセスすることができる、
ことを特徴とする、ネットワーク上のオブジェクトを管理するためのシステム。
【請求項２】
前記ネットワークサーバは、ハードウエアクラスターサーバ及びソフトウエアクラスターサーバから成る群から選択されることを特徴とする請求項１に記載のシステム。
【請求項３】
前記分散型合意アルゴリズムは、前記リードサーバと前記複数のサーバとの間に、前記複数のネットワークサーバの大部分が前記ホストサーバについて同意するまで継続するメッセージのラウンドを含むことを特徴とする請求項１に記載のシステム。
【請求項４】
前記ホストサーバは、前記ネットワークデータソースからのデータのコピーを含むデータオブジェクトを包含することを特徴とする請求項１に記載のシステム。
【請求項５】
前記ホストサーバは、前記ネットワークデータソース内の前記データ項目に対する唯一のアクセスポイントとして働くデータオブジェクトを包含することを特徴とする請求項１に記載のシステム。
【請求項６】
前記データ項目は、トランザクションログであることを特徴とする請求項１に記載のシステム。
【請求項７】
前記分散型合意アルゴリズムは、「Ｐａｘｏｓ」アルゴリズムであることを特徴とする請求項１に記載のシステム。
【請求項８】
各々がネットワークデータソースと通信するようになった複数のネットワークサーバと、
前記複数のネットワークサーバにおけるリードサーバと、
を含み、
前記リードサーバは、前記複数のネットワークサーバからホストサーバを選択するための分散型合意アルゴリズムを包含し、
前記ホストサーバは、前記ネットワークデータソースに位置するデータ項目のコピーを包含し、それによって、該データ項目にアクセスする必要がある前記複数のネットワークサーバのいずれもが該ホストサーバ上の該コピーにアクセスすることができる、
ことを特徴とする、ネットワーク上のオブジェクトを管理するためのシステム。
【請求項９】
各々がネットワークデータソースと通信するようになった複数のネットワークサーバと、
前記複数のネットワークサーバにおけるリードサーバと、
を含み、
前記リードサーバは、前記複数のネットワークサーバからホストサーバを選択するための分散型合意アルゴリズムを包含し、
前記ホストサーバは、前記ネットワークデータソースに位置するデータ項目への唯一のアクセスポイントを包含し、従って、該データ項目にアクセスする必要がある前記複数のネ

ットワークサーバのいずれもが、該ホストサーバを通じて該データ項目にアクセスしなければならない、
ことを特徴とする、ネットワーク上のオブジェクトを管理するためのシステム。
【請求項１０】
データ項目の少なくとも１つのコピーを包含するファイルシステムと、
前記ファイルシステムと通信する複数のサーバと、
前記複数のサーバからホストサーバを選択するための分散型合意アルゴリズムを包含する、該複数のサーバにおけるリードサーバと、
前記データ項目のローカルコピーを包含する、前記複数のサーバにおけるホストサーバと、
を含み、
前記ホストサーバは、前記ローカルコピーへのアクセスを前記複数のサーバのいずれにも提供し、該ローカルコピーに更新が為された時はいつでも前記ファイルシステム内の前記データ項目の前記コピーを更新するようになっている、
ことを特徴とする、ネットワーク上のオブジェクトを管理するためのシステム。
【請求項１１】
前記ホストサーバは、更に、前記ローカルコピーをローカルキャッシュに記憶するようになっていることを特徴とする請求項１０に記載のシステム。
【請求項１２】
前記複数のサーバは、クラスターを含むことを特徴とする請求項１０に記載のシステム。
【請求項１３】
前記ファイルシステムは、複数のディスク上で前記データ項目を複製することを特徴とする請求項１０に記載のシステム。
【請求項１４】
データ項目の少なくとも１つのコピーを包含するファイルシステムと、
前記ファイルシステムと通信する複数のサーバと、
前記複数のサーバ内に位置するハードウエアクラスターサーバを包含し、かつ該ハードウエアクラスターサーバの中からリードサーバを選択するための分散型合意アルゴリズムを包含するハードウエアクラスターと、
前記複数のサーバからホストサーバを選択するためのアルゴリズムを包含する、前記ハードウエアクラスターサーバにおけるリードサーバと、
前記データ項目のローカルコピーを包含する、前記複数のサーバにおけるホストサーバと、
を含み、
前記ホストサーバは、前記ローカルコピーへのアクセスを前記複数のサーバのいずれにも提供し、該ローカルコピーに更新が為された時はいつでも前記ファイルシステム内の前記データ項目の前記コピーを更新するようになっている、
ことを特徴とする、ネットワーク上のオブジェクトを管理するためのシステム。
【請求項１５】
前記ホストサーバは、前記ハードウエアクラスター内にあることを特徴とする請求項１４に記載のシステム。
【請求項１６】
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
ファイルシステムから前記ホストサーバにデータ項目のコピーを引き抜く段階と、
ネットワーク要求を処理するのに使用される前記データ項目のコピーを前記ホストサーバが包含することを他のネットワークサーバに通知する段階と、
を含むことを特徴とする、ネットワーク上のオブジェクトを管理する方法。
【請求項１７】
前記ホストサーバ上の前記コピーが修正される時に前記ファイルシステム内の前記データ

を更新する段階を更に含むことを特徴とする請求項１６に記載の方法。
【請求項１８】
前記ファイルシステムにアクセスするのに前記ホストサーバを通過するように他のネットワークサーバを制限する段階を更に含むことを特徴とする請求項１６に記載の方法。
【請求項１９】
前記ファイルシステムの外側に前記データ項目の１つのコピーのみが存在することを確実にする段階を更に含むことを特徴とする請求項１６に記載の方法。
【請求項２０】
前記ファイルシステムの外側に前記データ項目の１つのコピーが常に存在することを確実にする段階を更に含むことを特徴とする請求項１６に記載の方法。
【請求項２１】
前記ホストサーバが、もはや前記オブジェクトをホスティングすることができない場合に、分散型合意アルゴリズムを使用して複数のネットワークサーバの中から新しいホストサーバを選択する段階を更に含むことを特徴とする請求項１６に記載の方法。
【請求項２２】
前記分散型合意アルゴリズムを使用して選択された前記ホストサーバが、もはや前記オブジェクトをホスティングすることができない場合に、ファイルシステムから前記新しいホストサーバにデータ項目のコピーを引き抜く段階を更に含むことを特徴とする請求項１６に記載の方法。
【請求項２３】
前記ホストサーバが、もはや前記オブジェクトをホスティングすることができない場合に、ネットワーク要求を処理するのに使用される前記データ項目のコピーを新しいホストサーバが包含することを他のネットワークサーバに通知する段階を更に含むことを特徴とする請求項１６に記載の方法。
【請求項２４】
各々がデータオブジェクトをキャッシュに入れることができる複数のサーバと、
データ項目の少なくとも１つのコピーを包含するファイルシステムと、
前記データオブジェクトのコピーをキャッシュに入れるホストサーバを前記複数のサーバの中から選択するための分散型合意アルゴリズムと、
前記データオブジェクトのコピーをホストコンピュータが包含することを前記ネットワーク上のサーバに通知するための分配システムと、
を含むことを特徴とする、ネットワーク上のオブジェクトを管理するためのフレームワーク。
【請求項２５】
分散型合意アルゴリズムを使用して、ホストサーバを複数のネットワークサーバの中から選択する段階と、
特定の期間に亘ってデータ項目への唯一のアクセスを提供するために割り当てられた前記ホストサーバにデータオブジェクトを割り当てる段階と、
ファイルシステムから前記ホストサーバにデータ項目のコピーを引き抜く段階と、
ネットワーク要求を処理するのに使用される前記データ項目のコピーを前記ホストサーバが包含することを他のネットワークサーバに通知する段階と、
を含むことを特徴とする、オブジェクトをネットワーク上のサーバにリースする方法。
【請求項２６】
前記特定の期間が終了した状態で、前記データオブジェクトを別の期間に亘って前記ホストサーバに割り当てる段階を更に含むことを特徴とする請求項２５に記載の方法。
【請求項２７】
前記特定の期間が前記ホストサーバ上で終了した状態で、前記データオブジェクトを別の特定の期間に亘って新しいホストサーバに割り当てる段階を更に含むことを特徴とする請求項２５に記載の方法。
【請求項２８】

ハードウエアクラスター内の複数のハードウエアクラスターサーバの中からリードサーバを選択する段階と、
前記リードサーバ上の分散型合意アルゴリズムを使用して、複数のネットワークサーバからホストサーバを選択する段階と、
特定の期間に亘ってデータ項目への唯一のアクセスを提供するために割り当てられた前記ホストサーバにデータオブジェクトを割り当てる段階と、
ファイルシステムから前記ホストサーバにデータ項目のコピーを引き抜く段階と、
ネットワーク要求を処理するのに使用される前記データ項目のコピーを前記ホストサーバが包含することを他のネットワークサーバに通知する段階と、
を含むことを特徴とする、オブジェクトをネットワーク上のサーバにリースする方法。

【請求項２９】
ハードウエアクラスター内の複数のハードウエアクラスターサーバの中からリードサーバを選択する段階と、
前記リードサーバ上の分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
ネットワーク上のデータオブジェクトへの唯一のアクセスを提供するために割り当てられた前記ホストサーバにデータオブジェクトを割り当てる段階と、
ファイルシステムから前記ホストサーバにデータオブジェクトのコピーを引き抜く段階と、
ネットワーク要求を処理するのに使用される前記データオブジェクトのコピーを前記ホストサーバが包含することを他のネットワークサーバに通知する段階と、
を含むことを特徴とする、ネットワーク上のオブジェクトの所有権を割り当てる方法。

【請求項３０】
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
ネットワーク上で「ＪＭＳ」に対する唯一のアクセスポイント及びメッセージ待ち行列を提供する「ＪＭＳ」メッセージストアを含む「ＪＭＳ」オブジェクトを前記ホストサーバに割り当てる段階と、
前記ホストサーバが前記唯一の「ＪＭＳ」メッセージストアをホスティングしていることを前記ネットワークサーバに通知する段階と、
を含むことを特徴とする、ネットワーク上の「Ｊａｖａメッセンジャーサービス（ＪＭＳ）」をホスティングする方法。

【請求項３１】
前記複数のネットワークサーバの１つに向けたメッセージの有無に関して前記「ＪＭＳ」メッセージストアを検査する段階を更に含むことを特徴とする請求項３０に記載の方法。

【請求項３２】
「ＪＭＳ」メッセージをネットワークサーバから前記ホストサーバ上の前記「ＪＭＳ」メッセージストアに送信する段階を更に含むことを特徴とする請求項３０に記載の方法。

【請求項３３】
前記ホストサーバ上の前記「ＪＭＳ」メッセージストア内のメッセージを「ＪＭＳ」構成要素に送信する段階を更に含むことを特徴とする請求項３０に記載の方法。

【請求項３４】
複数のサーバにおけるホストサーバを使用して、データオブジェクトへのアクセスを提供する段階と、
前記ホストサーバが前記データオブジェクトへのアクセスを提供することができない場合に、分散型合意アルゴリズムを使用して前記複数のサーバの中から新しいホストサーバを選択する段階と、
前記データオブジェクトへのアクセスを提供するのに必要な情報を前記新しいホストサーバに引き抜く段階と、
新しいホストサーバが前記データオブジェクトへのアクセスを提供していることを前記複

数のサーバの他のサーバに通知する段階と、
を含むことを特徴とする、クラスター内のオブジェクトの存在を確実にする方法。
【請求項３５】
複数のサーバの中からリードサーバを選択する段階と、
前記リードサーバ上の分散型合意アルゴリズムを使用して、複数のサーバの中から管理サーバを選択する段階と、
データソース内及び前記管理サーバ上の情報を調整するために、データソースから該管理サーバに管理情報を引き抜き、該データソース内の管理情報を更新する段階と、
前記クラスター内の他のサーバに前記管理サーバのアイデンティティを通知する段階と、
を含むことを特徴とする、クラスター内の管理サーバの利用可能性を確実にする方法。
【請求項３６】
前記データソースから管理情報を引き抜く段階は、管理情報をファイルシステムから引き抜く段階を含むことを特徴とする請求項３５に記載の方法。
【請求項３７】
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
データ項目への唯一のアクセスを提供するために割り当てられた前記ホストサーバにデータオブジェクトを割り当てる段階と、
ネットワーク要求を処理するのに使用される前記データ項目のコピーを前記ホストサーバが包含するという通知を、クラスター内の他のサーバに対してマルチキャストする段階と、
を含むことを特徴とする、クラスター内のオブジェクトを分配する方法。
【請求項３８】
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
前記クラスター内の各サーバに連絡して、前記選択されたホストがそのサーバにとって容認可能か否かを判断する段階と、
前記クラスター内の全てのサーバが、前記選択されたホストが容認可能であることに同意した場合、新しいホストサーバの選択を委託する通知を該クラスター内の他のサーバにマルチキャストする段階と、
を含むことを特徴とする、クラスター内のオブジェクトを分配する方法。
【請求項３９】
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択するための手段と、
データ項目への唯一のアクセスを提供するために割り当てられた前記ホストサーバにデータオブジェクトを割り当てるための手段と、
ファイルシステムから前記ホストサーバにデータ項目のコピーを引き抜くための手段と、
ネットワーク要求を処理するのに使用される前記データ項目のコピーを前記ホストサーバが包含することを他のネットワークサーバに通知するための手段と、
を含むことを特徴とするコンピュータ可読媒体。
【請求項４０】
ネットワーク上のオブジェクトを管理するサーバコンピュータによって実行されるコンピュータプログラム製品であって、
分散型合意アルゴリズムを使用して複数のネットワークサーバの中からホストサーバを選択するためのコンピュータコードと、
ファイルシステムから前記ホストサーバにデータ項目のコピーを引き抜くためのコンピュータコードと、
ネットワーク要求を処理するのに使用される前記データ項目のコピーを前記ホストサーバが包含することを他のネットワークサーバに通知するためのコンピュータコードと、
を含むことを特徴とする製品。

【請求項４１】
分散型合意アルゴリズムを使用して複数のネットワークサーバの中からホストサーバを選択するための手段と、
ファイルシステムから前記ホストサーバにデータ項目のコピーを引き抜くための手段と、
ネットワーク要求を処理するのに使用される前記データ項目のコピーを前記ホストサーバが包含することを他のネットワークサーバに通知するための手段と、
を含むことを特徴とする、クラスター内のオブジェクトを分配するためのシステム。
【請求項４２】
プロセッサと、
前記プロセッサによって実行されるオブジェクトコードと、
を含み、
前記オブジェクトコードは、
分散型合意アルゴリズムを使用して複数のネットワークサーバの中からホストサーバを選択し、
ファイルシステムから前記ホストサーバにデータ項目のコピーを引き抜き、
ネットワーク要求を処理する際に使用される前記データ項目のコピーを前記ホストサーバが包含することを他のネットワークサーバに通知する、
ように構成される、
ことを特徴とするコンピュータシステム。
【請求項４３】
「Ｐａｘｏｓ」アルゴリズムを使用して、ソフトウエアクラスター内の複数のネットワークサーバの中からホストサーバを選択する段階と、
データオブジェクトを前記ホストサーバに割り当てる段階と、
を含み、
前記データオブジェクトは、ネットワーク内の前記ホストサーバ上にのみ存在する、
ことを特徴とする、ネットワーク上のオブジェクトを管理する方法。
【請求項４４】
前記データオブジェクトのためのデータをファイルシステムから引き抜く段階を更に含むことを特徴とする請求項４３に記載の方法。
【請求項４５】
ネットワーク要求を処理するのに使用される前記データ項目のためのオブジェクトを前記ホストサーバが包含することを他のネットワークサーバに通知する段階を更に含むことを特徴とする請求項４３に記載の方法。
【請求項４６】
前記新しいホストサーバの識別を前記他のネットワークサーバにマルチキャストする段階を更に含むことを特徴とする請求項４３に記載の方法。
【請求項４７】
前記「Ｐａｘｏｓ」アルゴリズムを使用してソフトウエアクラスター内の複数のネットワークサーバの中からホストサーバを選択する段階は、情報のラウンドを前記他のネットワークサーバにマルチキャストする段階を含むことを特徴とする請求項４３に記載の方法。
【請求項４８】
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
ネットワーク上で「ＪＭＳ」に対する唯一のアクセスポイント及びメッセージ待ち行列を提供する「ＪＭＳ」メッセージストアを含む「ＪＭＳ」オブジェクトを前記ホストサーバに割り当てる段階と、
前記ホストサーバが前記唯一の「ＪＭＳ」メッセージストアをホスティングしていることを前記ネットワークサーバに通知する段階と、
を含むことを特徴とする、ネットワーク上の「Ｊａｖａメッセンジャーサービス（ＪＭＳ）」をホスティングする方法。

【請求項４９】
前記複数のネットワークサーバの１つに向けたメッセージの有無に関して前記「ＪＭＳ」メッセージストアを検査する段階を更に含むことを特徴とする請求項４８に記載の方法。
【請求項５０】
「ＪＭＳ」メッセージをネットワークサーバから前記ホストサーバ上の前記「ＪＭＳ」メッセージストアに送信する段階を更に含むことを特徴とする請求項４８に記載の方法。
【請求項５１】
前記ホストサーバ上の前記「ＪＭＳ」メッセージストア内のメッセージを「ＪＭＳ」構成要素に送信する段階を更に含むことを特徴とする請求項４８に記載の方法。
【請求項５２】
ネットワーク上で「ＪＭＳ」に対する唯一のアクセスポイント及びメッセージ待ち行列を提供する「ＪＭＳ」メッセージストアを含む「ＪＭＳ」オブジェクトへのアクセスを、複数のサーバ内のホストサーバを使用して提供する段階と、
前記ホストサーバが前記「ＪＭＳ」オブジェクトへのアクセスを提供することができない場合に、分散型合意アルゴリズムを使用して前記複数のサーバの中から新しいホストサーバを選択する段階と、
前記「ＪＭＳ」オブジェクトへのアクセスを提供するのに必要な情報を前記新しいホストサーバに引き抜く段階と、
新しいホストサーバが前記「ＪＭＳ」オブジェクトへのアクセスを提供していることを前記複数のサーバの他のサーバに通知する段階と、
を含むことを特徴とする、クラスター内の「ＪＭＳ」オブジェクトの存在を確実にする方法。
【請求項５３】
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
前記クラスター内の各サーバに連絡して、前記選択されたホストがそのサーバにとって容認可能か否かを判断する段階と、
前記クラスター内の全てのサーバが、前記選択されたホストが容認可能であることに同意した場合、新しいホストサーバの選択を委託する通知を該クラスター内の他のサーバにマルチキャストする段階と、
を含むことを特徴とする、「ＪＭＳ」オブジェクトをクラスター内のサーバに割り当てる方法。
【請求項５４】
分散型合意アルゴリズムを使用して複数のネットワークサーバの中からホストサーバを選択するための手段と、
「ＪＭＳ」への唯一のアクセスを提供するために割り当てられた前記ホストサーバに「ＪＭＳ」オブジェクトを割り当てるための手段と、
前記ホストサーバが「ＪＭＳ」への唯一のアクセスを提供することを他のネットワークサーバに通知するための手段と、
を含むことを特徴とするコンピュータ可読媒体。
【請求項５５】
ネットワーク上のオブジェクトを管理するサーバコンピュータによって実行されるコンピュータプログラム製品であって、
分散型合意アルゴリズムを使用して複数のネットワークサーバの中からホストサーバを選択するためのコンピュータコードと、
「ＪＭＳ」への唯一のアクセスを提供するために割り当てられた前記ホストサーバに「ＪＭＳ」オブジェクトを割り当てるためのコンピュータコードと、
前記ホストサーバが「ＪＭＳ」への唯一のアクセスを提供することを他のネットワークサーバに通知するためのコンピュータコードと、
を含むことを特徴とする製品。

【請求項５６】
分散型合意アルゴリズムを使用して複数のネットワークサーバの中からホストサーバを選択するための手段と、
「ＪＭＳ」への唯一のアクセスを提供するために割り当てられた前記ホストサーバに「ＪＭＳ」オブジェクトを割り当てるための手段と、
前記ホストサーバが「ＪＭＳ」への唯一のアクセスを提供することを他のネットワークサーバに通知するための手段と、
を含むことを特徴とする、クラスター内のオブジェクトを分配するためのシステム。
【請求項５７】
プロセッサと、
前記プロセッサによって実行されるオブジェクトコードと、
を含み、
前記オブジェクトコードは、
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択し、
「ＪＭＳ」への唯一のアクセスを提供するために割り当てられた前記ホストサーバに「ＪＭＳ」オブジェクトを割り当て、
前記ホストサーバが「ＪＭＳ」への唯一のアクセスを提供することを他のネットワークサーバに通知する、
ように構成される、
ことを特徴とするコンピュータシステム。
【請求項５８】
「Ｐａｘｏｓ」アルゴリズムを使用して、ソフトウエアクラスター内の複数のネットワークサーバの中からホストサーバを選択する段階と、
「ＪＭＳ」メッセージストアを前記ホストサーバに割り当てる段階と、
を含み、
前記「ＪＭＳ」メッセージストアは、前記ホストサーバ上にのみ存在する、
ことを特徴とする、ネットワーク上の「ＪＭＳ」メッセージストアを管理する方法。
【請求項５９】
前記ホストサーバが「ＪＭＳ」メッセージストアをホスティングすることを他のネットワークサーバに通知する段階を更に含むことを特徴とする請求項５８に記載の方法。
【請求項６０】
新しいホストサーバの識別を他のネットワークサーバにマルチキャストする段階を更に含むことを特徴とする請求項５８に記載の方法。
【請求項６１】
前記「Ｐａｘｏｓ」アルゴリズムを使用してソフトウエアクラスター内の複数のネットワークサーバの中からホストサーバを選択する段階は、情報のラウンドを他のネットワークサーバにマルチキャストする段階を含むことを特徴とする請求項５８に記載の方法。
【請求項６２】
複数のネットワークサーバと、
前記複数のネットワークサーバにおけるリードサーバと、
を含み、
前記リードサーバは、前記複数のネットワークサーバからホストサーバを選択するための分散型合意アルゴリズムを包含し、
前記ホストサーバは、「ＪＭＳ」オブジェクトを包含し、従って、「ＪＭＳ」にアクセスする必要がある前記複数のネットワークサーバのいずれもが、該ホストサーバ上の該「ＪＭＳ」オブジェクトにアクセスしなければならない、
ことを特徴とする、ネットワーク上の「ＪＭＳ」オブジェクトを管理するためのシステム。
【請求項６３】

前記ネットワークサーバは、ハードウエアクラスターサーバ及びソフトウエアクラスターサーバから成る群から選択されることを特徴とする請求項６２に記載のシステム。
【請求項６４】
前記分散型合意アルゴリズムは、前記リードサーバと前記複数のサーバとの間にメッセージのラウンドを含み、
前記ラウンドは、前記複数のネットワークサーバの大部分が前記ホストサーバについて同意するまで継続される、
ことを特徴とする請求項６２に記載のシステム。
【請求項６５】
前記分散型合意アルゴリズムは、「Ｐａｘｏｓ」アルゴリズムであることを特徴とする請求項６４に記載のシステム。
【請求項６６】
「ＪＭＳ」構成要素と、
前記「ＪＭＳ」構成要素と通信する複数のサーバと、
前記複数のサーバからホストサーバを選択するための分散型合意アルゴリズムを包含する、該複数のサーバにおけるリードサーバと、
「ＪＭＳ」メッセージストアを包含し、前記「ＪＭＳ」構成要素へのアクセスを前記複数のサーバのいずれにも提供するようになった、該複数のサーバにおけるホストサーバと、
を含むことを特徴とする、ネットワーク上の「ＪＭＳ」を管理するためのシステム。
【請求項６７】
前記複数のサーバは、クラスターを含むことを特徴とする請求項６６に記載のシステム。
【請求項６８】
「ＪＭＳ」構成要素と、
前記「ＪＭＳ」構成要素と通信する複数のサーバと、
前記複数のサーバ内に位置するハードウエアクラスターサーバを包含し、かつ該ハードウエアクラスターサーバの中からリードサーバを選択するためのアルゴリズムを包含するハードウエアクラスターと、
前記複数のサーバからホストサーバを選択するための分散型合意アルゴリズムを包含する、前記ハードウエアクラスターサーバにおけるリードサーバと、
「ＪＭＳ」メッセージストアを包含し、「ＪＭＳ」へのアクセスを前記複数のサーバのいずれにも提供するようになった、該複数のサーバにおけるホストサーバと、
を含むことを特徴とする、ネットワーク上の「ＪＭＳ」を管理するためのシステム。
【請求項６９】
前記ホストサーバは、前記ハードウエアクラスター内にあることを特徴とする請求項６８に記載のシステム。
【請求項７０】
各々が「ＪＭＳ」メッセージストアをホスティングすることができる複数のサーバと、
「ＪＭＳ」メッセージストアをホスティングするためのホストサーバを、前記複数のサーバの中から選択するための分散型合意アルゴリズムと、
ホストコンピュータが前記「ＪＭＳ」メッセージストアをホスティングしていることをネットワーク上のサーバに通知するための分配システムと、
を含むことを特徴とする、ネットワーク上の「ＪＭＳ」を管理するためのフレームワーク。
【請求項７１】
分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
特定の期間に亘って「ＪＭＳ」への唯一のアクセスを提供するために割り当てられた前記ホストサーバに「ＪＭＳ」オブジェクトを割り当てる段階と、
前記ホストサーバが前記「ＪＭＳ」オブジェクトをホスティングしていることを他のネットワークサーバに通知する段階と、

を含むことを特徴とする、「ＪＭＳ」オブジェクトをネットワーク上のサーバにリースする方法。
【請求項７２】
前記特定の期間が終了した状態で、前記「ＪＭＳ」オブジェクトを別の期間に亘って前記ホストサーバに割り当てる段階、
を更に含むことを特徴とする請求項７１に記載の方法。
【請求項７３】
前記特定の期間が前記ホストサーバ上で終了した状態で、前記「ＪＭＳ」オブジェクトを別の特定の期間に亘って新しいホストサーバに割り当てる段階、
を更に含むことを特徴とする請求項７２に記載の方法。
【請求項７４】
ハードウエアクラスター内の複数のハードウエアクラスターサーバに中からリードサーバを選択する段階と、
前記リードサーバ上の分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
特定の期間に亘って「ＪＭＳ」への唯一のアクセスを提供するために割り当てられた前記ホストサーバに「ＪＭＳ」オブジェクトを割り当てる段階と、
前記ホストサーバが前記「ＪＭＳ」オブジェクトをホスティングしていることを他のネットワークサーバに通知する段階と、
を含むことを特徴とする、「ＪＭＳ」オブジェクトをネットワーク上のサーバにリースする方法。
【請求項７５】
ハードウエアクラスター内の複数のハードウエアクラスターサーバの中からリードサーバを選択する段階と、
前記リードサーバ上の分散型合意アルゴリズムを使用して、複数のネットワークサーバの中からホストサーバを選択する段階と、
「ＪＭＳ」への唯一のアクセスを提供するために割り当てられた前記ホストサーバに「ＪＭＳ」オブジェクトを割り当てる段階と、
前記ホストサーバが前記「ＪＭＳ」オブジェクトをホスティングしていることを他のネットワークサーバに通知する段階と、
を含むことを特徴とする、ネットワーク上のオブジェクトの所有権を割り当てる方法。
【発明の詳細な説明】
【技術分野】
【０００１】
優先権の請求
本出願は、本明細書に組み込まれる以下の出願に対する優先権を請求するものである。
２００１年９月６日出願のディーン・バーナード・ヤコブズ及びエリック・ハルパーンに付与された「厳密に一回のキャッシュフレームワーク」という名称の米国特許仮出願第６０／３１７，７１８号、
２００２年９月４日出願のディーン・バーナード・ヤコブズ及びエリック・ハルパーンに付与された「厳密に一回のキャッシュフレームワーク」という名称の米国特許出願、
２００１年９月６日出願のディーン・バーナード・ヤコブズ及びエリック・ハルパーンに付与された「厳密に一回のＪＭＳ通信」という名称の米国特許仮出願第６０／３１７，５６６号、及び
２００２年９月４日出願のディーン・バーナード・ヤコブズ及びエリック・ハルパーンに付与された「厳密に一回のＪＭＳ通信」という名称の米国特許出願。

相互参照例
以下の米国特許出願は、相互参照されて本明細書において引用により組み込まれる。
２００１年７月１６日に出願のディーン・バーナード・ヤコブズ、リト・クレイマー、及びアナンサン・バラ・スリニバサンに付与された「データ複製プロトコル」という名称の米国特許出願号第６０／３０５，９８６号。
本発明は、オブジェクトをネットワーククラスター内のサーバ間に分配するための技術に関する。
【背景技術】
【０００２】
分散型コンピュータシステムにおいては、いくつかのサーバ及び／又はネットワーキングノードが協働する必要がある場合が多い。マシンが単一のエンティティとして機能することができるようにマシン間で共有すべきネットワーキング情報が通常存在するので、これらのサーバ及びノードは調整されるべきである。マシン調整の一般的な手法は、リソース及び効率に関して非常に経費が掛かる可能性がある。
【０００３】
一般に、ノード間で伝達されるいくつかのメッセージがあると考えられるので、ノードが同意するように何らかの同期化が必要である。しかし、この同期化要件は、クラスター化されたネットワーキング環境では望ましくないかもしれない。多くのクラスター化された環境においては、このようないかなる同期化要件を課すことも単に回避される。しかし、同意が必要な用途も存在する。
【０００４】
同意が必要とされる１つの場合においては、クラスターが独占的なアクセスを望むことがある装置が存在することができる。１つのこのような装置は、ファイルシステム上のトランザクションログである。トランザクションが進行中の時は常に、永続的な方法で保存する必要があるいくつかのオブジェクトがあり、それによって、故障が起こった場合にその永続的保存オブジェクトを回復することができる。
【０００５】
１ヵ所に保存する必要があるこれらのオブジェクトについては、通常、そのクラスター又はドメイン内の各サーバ上で作動するトランザクションモニタがあり、それは、次にローカルファイルシステムを使用してそのオブジェクトにアクセスする。各サーバは、永続性に関する問題がほとんどないか又は全くないように独自のトランザクションマネージャを有することができる。従って、各サーバがトランザクションマネージャを有するので調整の必要もない。
【発明の開示】
【発明が解決しようとする課題】
【０００６】
例えば、各々がトランザクションマネージャを有する３つのサーバを含むクラスターが存在することができる。これらのサーバの１つは、クラスターがそのサーバを利用することができなくなるような故障又は他の問題が発生する可能性がある。故障したサーバは、特定のトランザクションログにアクセス可能な唯一のサーバであるために、その特定のログ内の全てのトランザクションは、クラスターがそのサーバを利用することができるようになるまで再び回復することはできない。サーバに関する問題は、解決するのにかなりの時間が掛かる可能性があるために、ログの回復は、難しいか又は少なくとも非効率的なものになる可能性がある。重大なサーバ上の問題には、サーバ上のマザーボードからの短絡又は電源の消耗の発生などを挙げることができる。
【課題を解決するための手段】
【０００７】
本発明は、ネットワーク上又はクラスター内のサーバに記憶することができるような、オブジェクトを管理するためのシステムを含む。本システムは、クラスターの内側又は外側に位置することができる、ファイルシステム又は「Ｊａｖａ（登録商標）メッセージサー

ビス」構成要素のようなデータソース、アプリケーション、又はサービスを含む。本システムは、高速ネットワーク接続などを通じてファイルシステム又はアプリケーションと通信するいくつかのサーバを含むことができる。

【０００８】

本システムは、他のサーバが同意することができるようなリードサーバ(lead server)を含む。リードサーバは、ハードウエアクラスター又はソフトウエアクラスター内に含めることができる。本システムは、ハードウエアクラスターマシンに内蔵されたアルゴリズムのような、サーバの中からリードサーバを選択するためのアルゴリズムを含むことができる。リードサーバは、それ自体、「Ｐａｘｏｓ」アルゴリズムのようなホストサーバを選択するための分散型合意アルゴリズムを含むことになる。リードサーバを選択するのに使用されるアルゴリズムは、ホストサーバを選択するのに使用されるアルゴリズムと異なるか又は同じものとすることができる。

【０００９】

ホストサーバは、ローカルキャッシュに記憶することができるのような項目又はオブジェクトのコピーを含むことができる。ホストサーバは、ネットワーク上又はクラスター内の任意のサーバへローカルコピーアクセスを提供することができる。ホストサーバはまた、ファイルシステムに記憶されたオブジェクトに対して唯一のアクセスポイント、又は、アプリケーション又はサービスに対して唯一のアクセスポイントを与えることができる。また、ホストサーバがキャッシュに入れるか、ホスティングするか、又は所有する項目に対して行われた任意の変更は、ファイルシステム、アプリケーション、又はサービス内で更新することができる。

【００１０】

ホストサーバがオブジェクトをホスティングすることができなくなった場合、分散型合意アルゴリズムを使用して新しいホストを選択することができる。その後、新しいホストは、ファイルシステム又はサービスからそのオブジェクトの必要なデータを引き抜くことができる。新しいサーバがそのオブジェクトをホスティングすることを、クラスター内の他のサーバに通知することができる。サーバへの通知は、２地点間接続又はマルチキャスティングなどによる任意の適切な手段によって行うことができる。

【発明を実施するための最良の形態】

【００１１】

本発明によるシステムは、データオブジェクトを所有するサーバがサーバクラスターに対して利用不能になった時のような利用可能性に関する問題の解決策を提供することができる。１つのこのような解決策は、データオブジェクトの所有権を引き継ぐためのクラスター内の別のサーバを考慮するものである。しかし、両方のサーバ上にデータオブジェクトを複製する必要なくデータオブジェクトを両方のサーバにアクセス可能にする際に問題が生じる。

【００１２】

ファイルシステム、データストア、又はデータベース（全てを総称して「ファイルシステム」という）がデータを永続的に記憶するためにクラスターによって使用され、また、ファイルシステムが１つよりも多いサーバからアクセス可能である場合、第２のサーバは、オブジェクトを所有する第１のサーバが問題に遭遇した場合に自動的にデータオブジェクトアクセスのタスクを引き継ぐことができる。代替的に、その項目の所有権を取るようにサーバに命令するためにクラスター又はクラスター内のサーバによって利用されるアルゴリズムが存在することが可能である。しかし、別の根本的な問題には、どのサーバが現在リソース又はオブジェクトを所有するかをクラスターに同意させること、又は、サーバ間の「合意」を達成することが絡んでくる。

【００１３】

図１は、トランザクションログ１１４などのオブジェクトがファイルシステム１１２に記憶される、本発明によるクラスターシステム１００の一例を示す。ファイルシステム１１２には、クラスター１１０内の全てのサーバ１０６、１１６、及び１１８がアクセス可能

であるが、一度にこれらのサーバの１つだけがログ１１４にアクセス可能である。クラスター１１０内のサーバ間のホストサーバ１０６は、ログ１１４のコピー１０８を記憶するか、又は、ファイルシステム１１２内でログ１１４に対する全てのアクセスを提供するなどにより、ログ１１４を「所有する」又は「ホスティングする」ことになる。クラスター１１０内の他のいずれかのサーバ１１６及び１１８は、ログのコピー１０８にアクセスすることができ、及び／又は、ホストサーバ１０６を通じてログ１１４にアクセス可能である。例えば、クライアント又はブラウザ１０２は、クラスター１１０内のサーバ１１６に方向付けられたネットワーク１０４に対する要求を行うことができる。そのサーバは、ネットワーク１０４を通じてホストサーバ１０６上のトランザクションログのコピー１０８にアクセス可能である。トランザクションログを更新する必要がある場合は、ファイルシステム１１２上のオリジナルログ１１４と共にコピー１０８を更新することができる。

【００１４】

サーバは、例えばデータオブジェクトのリポジトリーとして機能する時、データオブジェクトのコピーをローカルキャッシュに記憶してそのコピーをクラスター内の他のサーバに利用可能にするか、又は、クラスター内の全ての他のサーバがホストサーバを通じてオブジェクトにアクセスする必要があるように、ファイルシステム、サービス、又はアプリケーション内のオブジェクトへの直接アクセスを有する唯一のサーバになるなどにより、そのオブジェクトを「所有する」又は「ホスティングする」ことができる。これによって、サーバクラスター内にオブジェクトが確実に「厳密に一回」存在する。

【００１５】

図３は、オブジェクトのホスティングを確立するために使用することができる１つの処理３００を示す。ホストサーバは、「Ｐａｘｏｓ」アルゴリズムのような分散型合意アルゴリズム３０２を使用して選択することができる。このようなアルゴリズムは、クラスター内のサーバが、クラスターサーバ間でオブジェクトを分配する方法に関して全体的に同意するか又は合意に至るべきであるから、「分散型合意」アルゴリズムと呼ばれる。

【００１６】

ホスティングされているオブジェクトが、例えばホスティングサーバ上でキャッシュに入れられる場合は、データオブジェクトのコピーをファイルシステムからホストサーバに引き抜いて、ローカルキャッシュ３０４内のオブジェクトとして記憶することができる。その後、ネットワーク上又は適切なクラスター内の他のサーバは、オブジェクトのローカルコピーがホスティングサーバ上に存在すること、及び、そのローカルコピーが今後のネットワーク要求３０６を処理する際に使用されるべきであることをホストサーバなどによって通知される。

【００１７】

分散型合意アルゴリズムの一例である「Ｐａｘｏｓ」アルゴリズムにおいては、ネットワークサーバは、ホストサーバ又はリードサーバとして機能するサーバを選択することができ、ネットワークサーバが一連の「合意ラウンド」をリードする。これらのラウンドの各々においては、新しいホストサーバ又はリードサーバが提案される。ラウンドは、提案されたサーバの１つがサーバの大部分又は定足数によって受け入れられるまで続行される。どのサーバもラウンドを初期化することによってホストサーバ又はリードサーバを提案することができるが、システムは、リードサーバが常にホストサーバ選択のためのラウンドを初期化するように構成することができる。異なる選択のためのラウンドを同時に実行することができる。従って、ラウンド選択は、ラウンド番号、又は１つの値がラウンドを示し、１つの値がそのラウンドをリードするサーバを示すような値の対によって識別することができる。

【００１８】

１つのこのようなラウンドの手順は以下の通りであるが、他の手順及び／又は手法がいくつかの状況又は用途では適切な場合がある。第一に、リーダーが「回収」メッセージをクラスター内の他のサーバに送ることによってラウンドを開始することができる。回収メッセージにより、クラスター内のサーバから、サーバが参加した以前に為されたラウンドに

関する情報が回収される。また、この特定の選択処理について以前の合意ラウンドがあった場合は、回収メッセージは、サーバに以前のラウンドから選択を行わないように通知する。リーダーは、クラスターサーバの少なくとも半分から回答を収集すると、例えば、次のラウンドで提案すべき値を判断し、この提案を「開始」メッセージとしてクラスターサーバに送ることができる。リーダーがこの手法で提案すべき値を選択するためには、サーバから初期値情報を受信する必要がある。

【００１９】

サーバは、リーダーから開始メッセージを受信すると、「受理」メッセージを送ることによって回答し、サーバが提案されたホスト／リードサーバを受理することを示すことができる。リーダーがサーバの大部分又は定足数から受理メッセージを受信した場合、リーダーは、出力値をラウンドで提案された値に設定する。リーダーが所定の期間内に大部分又は定足数による受理（合意）を受信しなかった場合、リーダーは、新しいラウンドを開始することができる。リーダーが合意を受信した場合、リーダーは、サーバが選択されたサーバに委託すべきであることをクラスターサーバ又はネットワークサーバに通知することができる。この通知は、２地点間接続又はマルチキャスティングなどによる任意の適切な同報通信技術によってネットワークサーバに同報通信することができる。

【００２０】

合意手法の同意条件は、以前のラウンドに関する情報を利用する選択を提案することによって確実にすることができる。この情報は、任意の２つのラウンドについて両方のラウンドに参加したサーバが少なくとも１つあるように、ネットワークサーバの少なくとも大部分からのものであることが必要になる可能性がある。

【００２１】

リーダー（ｌｅａｄｅｒ）は、各サーバにそのサーバが値を受理した最新ラウンドの番号を尋ねることにより、また、恐らくは受理された値を求めることにより、新しいラウンドの値を選択することができる。リーダーがこの値をサーバの大部分又は定足数から取得すると、回答間の最新ラウンドの値に等しい値を新しいラウンドについて選択することができる。また、リーダーは、以前のラウンドに関与したサーバがない場合は初期値を選択することができる。リーダーが、最終受理ラウンドが例えばｘであって現在のラウンドがｙであるという回答を受信した場合、サーバは、一貫性を維持するために、ｘとｙの間のどのラウンドも受理されないであろうということを暗示することができる。

【００２２】

ラウンドリーダー及びネットワークサーバ間のサンプル対話は、以下のメッセージを伴う。

【００２３】

（１）「回収」－新しいラウンド「ｒ」が開始されているというメッセージがサーバに送られる。このメッセージは、ｍ＝（“Ｃｏｌｌｅｃｔ”、ｒ）という形を取ることができる。

【００２４】

（２）「最終」－受理された最終ラウンド「ａ」及びそのラウンドの値「ｖ」を与えるメッセージがネットワークサーバからリーダーに送られる。このメッセージは、ｍ＝（“Ｌａｓｔ”ｒ、ａ、ｖ）の形を取ることができる。

【００２５】

（３）「開始」－ラウンドｒの値を知らせるメッセージがサーバに送られる。このメッセージは、ｍ＝（“Ｂｅｇｉｎ”、ｒ、ｖ）の形を取ることができる。

【００２６】

（４）「受理」－ラウンドｒの値を受理するメッセージがサーバからリーダーに送られる。このメッセージは、ｍ＝（“Ａｃｃｅｐｔ”、ｒ）の形を取ることができる。

【００２７】

（５）「成功」－ラウンドｒの値ｖの選択を知らせるメッセージがサーバに送られる。このメッセージは、ｍ＝（“Ｓｅｃｃｅｓｓ”、ｒ、ｖ）の形を取ることができる。

【００２８】
（６）「確認」－サーバがラウンドｒに関する決定を受信したことを確認するメッセージが、サーバからリーダーに送られる。このメッセージは、ｍ＝（“Ａｃｋ”、ｒ）の形を取ることができる。
【００２９】
ハードウエアクラスター又はソフトウエアクラスターの内側又は外側のいずれかに位置する、サーバから分離されたファイルシステムが存在することができる。このファイルシステムは、トランザクションログを第１のディスクに記憶してファイルシステム内の第２のディスクに複製するなどにより、トランザクションログを永続的に記憶することができる。第１のディスクがクラッシュした場合、ファイルシステムは、クラスター及び／又はサーバからそのクラッシュを隠し、第２のディスクからログ情報を取得することができる。また、ファイルシステムは、第２のディスクのバックアップの役目を果たすことができる第３のディスクにそのログを複製することを選択することができる。
【００３０】
クラスター内のサーバの観点からは、ファイルシステムは、単一のリソースとすることができる。一実施形態においては、サーバは、単一のサーバがいつの時点でもファイルシステムを所有することだけに注意するであろう。
【００３１】
本発明によるシステムの別の例は、サーバクラスター内のキャッシュを伴う。単一のキャッシュにクラスター内のサーバに対してデータオブジェクトを表すようにすることが、ネットワーク性能上の理由などからクラスター化された環境では望ましいものであろう。クラスター内のサーバは、継続的に永続的記憶装置に戻る必要なくキャッシュにアクセス可能なので、単一のキャッシュに項目を保持することが有利になる可能性がある。データベース又はファイルシステムへのヒットが相対的に時間集約的になる可能性があるので、メモリ内の既にある項目を引き抜くことができると、このようなシステムの効率を大幅に上げることができる。
【００３２】
しかし、単一のキャッシュに関する１つの問題は、メモリに記憶されたオブジェクトをファイルシステムのディスク上に記憶されたものと確実に同じものにする必要があり得ることである。このような一貫性を必要とする１つの理由は、キャッシュに入れられた項目上で行われる任意の演算又は計算で確実に正しい結果が出るようにするためである。別の理由は、キャッシュがクラッシュしたり、又はそれ以外に痛んだり又は利用不能になった場合に、キャッシュをファイルシステムから復元することが必要になる可能性があることである。
【００３３】
クラスター内でこの種のキャッシュを処理する主たる方法は、少なくとも２つあるとすることができるが、特定の用途については、他の方法も少なくとも同様に良好に機能することができる。１つの方法は、複数の場所でキャッシュを複製することである。この手法は、キャッシュに入れられる項目に変更があった場合、キャッシュを複製する全てのサーバがその変更に同意するか又は少なくともその変更を知っていることが必要であるから、問題になる可能性がある。これは、結果的にリソース及び性能の点で非常に経費が掛かるものになる可能性がある。
【００３４】
本発明による代替手法は、クラスター内のキャッシュの所有者となるように特定のサーバを割り当て、そのキャッシュへの全てのアクセスは、その特定のサーバを通過する。クラスター内の任意のサーバが、このようなキャッシュをホスティングすることができる。各サーバは、１つ又はいくつかのキャッシュをホスティングすることができるか、又は、全くキャッシュをホスティングすることができない。キャッシュは、単一のサーバ上でホスティングされるか、又は、クラスター内の一部又は全てのサーバ間に分散することができる。クラスター自体は、ハードウエアクラスターか、又はソフトウエアアプリケーション

によって所定の「ソフトウエア」クラスター内にあるように指定されたサーバのグループのような任意の適切なクラスターとすることができる。

【００３５】

トランザクションログ及び／又はキャッシュであるいずれかの例を、システム上のどこかにあるような種類のオブジェクトとして考えることが可能であろう。任意のこのようなオブジェクトが確実にクラスター内で一回のみ存在するようにし、そのオブジェクトを常に利用可能なものにすることが望ましいであろう。また、オブジェクトをホスティングするサーバが故障した場合は、確実にオブジェクトを別のサーバ上で復元させることができ、確実にそのオブジェクトがクラスターに利用可能になることも望ましいであろう。

【００３６】

回復の１つの方法４００を図４に示す。本方法においては、サーバがまだネットワークに対して利用可能であるか否かなど、ホストサーバが引き続きオブジェクト４０２をホスティングすることができるか否かの判断が行われる。ホスティングすることができない場合は、分散型合意アルゴリズムを使用して新しいホストが選択される。この選択は、オリジナルホスト４０４を選択するために使用された方法に従って行うことができる。データオブジェクトのコピーは、ファイルシステムから新しいホストに引き抜かれると、ローカルキャッシュ４０６に記憶することができる。新しいホストサーバがオブジェクトのコピーを含むこと、及び、任意の今後のネットワーク要求４０８を処理する際にはそのローカルキャッシュを使用すべきであることが、ネットワーク上又は適切なクラスター内の他のサーバに通知される。

【００３７】

本発明によるシステム及び方法においては、クラスター内の厳密に１つの場所に存在するオブジェクトを定めることができ、確実にそれらのオブジェクトが常に存在するようにすることができる。サーバの観点からは、トランザクションログなどのオブジェクトがファイルシステムなどにより鏡像化又は複製されるか否かは問題ではないであろう。サーバの観点からは、クラスター内の任意のサーバによってアクセス可能な永続的記憶装置が常に１つある。本システムは、定期的にオブジェクトの存在を検査することができ、又は、ネットワーク上又はクラスター内の何らかのマシン上での存在を確保するためにオブジェクトが頻繁に再割り当てされることになるように、短期間に亘ってオブジェクトの所有権を割り当ててもよい。

【００３８】

ハードウエアクラスターは、各々が複数のサーバを実行することができるマシンの列を含むことができる。また、各マシンの後にファイルシステムが存在することができる。ハードウエアクラスター内のサーバは、より素早く意思決定を行ってハードウエアクラスター内のサーバ欠陥を処理することができるように、一般的に配線されている。ハードウエアクラスターは、サーバを含むマシンの物理的ハードウエアに大きさを限定することができる。ハードウエアクラスター内のサーバは、ソフトウエアクラスター内のサーバとして使用することができ、また、マシン上の個々のサーバがネットワークに対して利用可能であるから、ネットワークサーバも含むことができる。

【００３９】

これらのマシンの１つに対する共有ファイルシステムは、高速ネットワークなどを通じて、クラスター内の全てのサーバに利用可能とすることができる。ファイルシステムはまた、冗長であることができる。一実施形態においては、この冗長は、ファイルシステム用の複数のデータディスクを使用して行われる。このような冗長の実行においては、オブジェクトがファイルシステムに書き込まれる時はいつでも、複数のディスクに亘ってオブジェクトを複製することができる。このようなファイルシステムは、「ブラックボックス」として見た時、ディスクのいかなる故障にも耐え、依然としてクラスター内の任意のサーバからのデータ項目のアクセスを提供することができる。

【００４０】

「厳密に一回」のフレームワークという本発明によるフレームワークは、メモリに保持さ

れたこれらのオブジェクトが、常に信頼性のある永続的記憶機構によって支援されると仮定して構築することができる。例えば、トランザクションログを表すオブジェクトが存在することができる。オブジェクトが呼び出された時、対応するトランザクションログを更新することができる。これには、データベースから読み取られるか、又はデータベースに書き込まれる呼び出しを挙げることができる。そのトランザクションログを表すオブジェクトは、ホストサーバのようなクラスター内のサーバの１つに存在することができる。厳密に一回のフレームワークは、クラスター内のサーバの少なくとも１つが立ち上がって実行されている限り、別のサーバが故障した場合に確実にサーバがログの所有権を引き継ぐことができるようにすることができる。

【００４１】

キャッシュを表す１つのオブジェクトが存在してもよい。また、キャッシュが更新される度に、更新内容を永続的記憶装置に再度書き込むことができる。サーバの１つがデータ項目を使用する必要がある時は、そのサーバは、このオブジェクトを通過する必要がある可能性がある。キャッシュを表すオブジェクトをホスティングするサーバが故障した場合、そのオブジェクトを別のサーバで復活させることができる。復活したオブジェクトは、全ての必要な情報を永続的記憶装置から引き抜くことができる。

【００４２】

厳密に一回のフレームワークは、クラスターによる使用のためのメモリバッファとして機能することができる。フレームワークは、信頼性のある永続的記憶装置によって支援された本システム内のデータを表す単一のキャッシュを提供するることができる。データがキャッシュから読み取られる時は、永続的記憶装置にアクセスする必要なく読取りを行うことができる。しかし、更新内容がキャッシュに書き込まれる時は、故障の場合に本システムが回復することができるように、永続的記憶装置を通じて再度書き込む必要がある可能性がある。

【００４３】

厳密に一回のフレームワークの１つの重要な態様は、この手法が抽象化される方法に関連し、これは、用途及び／又は実施に依存して変わる可能性がある。「厳密に一回のオブジェクト」と呼ぶことができるような新しい種類の分散オブジェクトが作成される。厳密に一回のオブジェクトは、例えば、ファイルシステム内のデータ項目のローカルにキャッシュに入れられたコピー、又はクラスター内のサーバのためのこのようなデータ項目の唯一のアクセスポイントとすることができる。この抽象化を実行する根底にある技術も重要になる可能性がある。

【００４４】

本発明のシステムは、上述の「Ｐａｘｏｓ」アルゴリズムを使用する方法のような、分散型合意に有用ないくつかの方法のいずれかを利用することができる。複数のノード及び／又は分散型ノードがオブジェクトの１つの値に同意するための効率的な方法を提供するようなアルゴリズムを選択することができる。このアルゴリズムは、たとえノードが故障し、及び／又は、同意処理中に戻ったとしても機能するように選択することができる。

【００４５】

ネットワークのクラスター化の一般的な手法は、全てのメッセージが所期の受信者に配信されるか、又は少なくとも全ての所期の機能しているサーバに配信されることが保証される信頼性のある同報通信を利用する。信頼性のある同報通信では受信者が次のメッセージ又は受信者に移る前にメッセージに応答する必要があるので、この手法は、システムを並列処理することを非常に難しいものにする可能性がある。マルチキャスティングを利用する分散型アルゴリズムは、マルチキャスティングは全てのサーバがメッセージを受信することを保証しないので、保証数を低減する場合がある。しかし、マルチキャスティングは、各サーバからの応答を待たずに単一メッセージを同時に全てのクラスターサーバに対してマルチキャストすることができるので、システムが並列処理を行うことができるようにこの手法を簡素化することは確かである。マルチキャストメッセージを受信しないサーバは、後でその情報をリードサーバか又は別のクラスターサーバ又はネットワークサーバか

ら引き抜くことができる。本明細書で使用される時のネットワークサーバは、ハードウエアクラスター内、ソフトウエアクラスター内、又はいずれかのクラスターの外側であるかを問わず、ネットワーク上の任意のサーバを意味することができる。

【００４６】

厳密に一回のアーキテクチャの重要な態様は、合意上の困難が低減されることである。本発明によれば、分散型合意実施の性能は、分散型合意アルゴリズムと共にマルチキャストメッセージングを使用することによって向上させることができる。この手法は、全てのサーバが同意する上で必要とされるメッセージ交換及び／又はネットワークトラヒックの最少化を考慮することができる。

【００４７】

マルチキャスティング時には、いくつかの手法の１つを取ることができる。「一相分散」と呼ぶことができる第１の手法においては、リードサーバは、「Ｐａｘｏｓ」アルゴリズムのラウンドで使用するか、又はオブジェクトのための新しいホストが選択されたことを示すために使用することができるように、ネットワーク上の全ての他のサーバに対してメッセージをマルチキャストすることができる。この手法においては、リードサーバは、ネットワーク上で利用可能な任意のサーバに転送することができる１つのメッセージを送る必要があるだけである。サーバが一時的にネットワークから外される場合、サーバは、ネットワーク上に戻った後で新しいホストの識別を要求することができる。

【００４８】

「二相分散」と呼ぶことができる別のマルチキャスト手法を使用すると、リードサーバは、適切なアルゴリズムを使用してホストサーバを予め選択することができる。しかし、オブジェクトをそのホストに割り当てる前に、リードサーバは、サーバがその新しいホストサーバの選択に同意しているか否かを判断するために、クラスター内の他の全てのサーバに連絡する可能性がある。リードサーバは、２地点間接続によって各サーバに連絡することができ、又は、マルチキャストされた要求を配信した後に各サーバが回答するのを待つことができる。サーバがそのホストの選択に同意しなかった場合、リードサーバは、アルゴリズムを使用して新しいホストを予め選択することができる。リードサーバは、次に、別のラウンドで新たに予め選択されたホストのアイデンティティと共に別のマルチキャスト要求を配信するであろう。

【００４９】

全てのサーバが予め選択されたホストに同意した場合、リードサーバは、そのオブジェクトをホストサーバに割り当てることができる。その後、リードサーバは、コミットメッセージをマルチキャストし、新しい変更が発効したのでサーバはサーバの情報を相応に更新すべきであることをサーバに知らせることができる。

【００５０】

厳密に一回のフレームワークはまた、「リース」機構を利用することができる。このような機構を使用する際は、分散型合意を使用するなどにより、クラスターサーバにリードサーバについて同意させるためのアルゴリズムを使用することができる。そのリードサーバは、選択されるとクラスター内の様々なサーバへの厳密に一回のオブジェクトの割り当てを担当することができる。システムは、既存のリードサーバが故障した場合にクラスターサーバが常に新しいリーダーに同意することになるように設定することができる。

【００５１】

リードサーバがアクティブの間は、リードサーバは、システム内に存在すべき全ての厳密に一回のオブジェクトを知ることができる。リードサーバは、どのサーバが各オブジェクトをホスティングすべきかを決めることができ、その後、そのオブジェクトを選択されたサーバに「リース」することができる。オブジェクトがサーバにリースされた時、そのサーバは、リース期間継続中のようなある一定期間に亘ってそのオブジェクトを所有するか又はホスティングすることができる。リードサーバは、これらのリースを定期的に更新するように構成することができる。この手法は、サーバが故障したり、又は何らかの方法で接続を断たれたり、又はそれ以外にクラスター内で適正に作動していない場合、サーバが

そのリースを更新されないことを保証にする方法を提供することができる。

【００５２】

故障中の分散型システムに関する問題の大部分は、故障したサーバと単に応答していないサーバとの違いを判断することが難しいということである。ネットワークから何らかの理由で遮断されたいかなるサーバも、もはやオブジェクトをホスティングすることはできない。そのサーバは、それがクラスターには利用可能でなくても、リース期間の後でいかなるオブジェクトのホスティングも辞めることができることを依然として知っていることになる。そのサーバは、クラスターに利用可能ではないので、そのリースは更新されないことになる。

【００５３】

リードサーバはまた、ある一定の時間内にホストサーバに到達することができない場合、ホストサーバがオブジェクトの所有権を放棄することになることを知っている。リース期間は、数秒間のような任意の適切な時間とすることができる。リース期間は、クラスター内の全てのサーバについて同じものとすることができ、又はオブジェクト間で変えることもできる。

【００５４】

厳密に一回のアーキテクチャを使用するシステムはまた、これを緊密にすることもできる。オペレーティングシステムは、多くの場合、ハードウエアのより近くに構築されてより多くの制御を与えることができる特殊なマシンを提供する。しかし、この手法を用いる１つの問題は、利用可能なハードウエアによって限定される可能性があるということである。例えば、サーバのハードウエアクラスターが有することができるサーバ数は、約１６程度である。これらのシステムでは何らかの緊密なハードウエア結合が要求されるので、クラスターに含むことができるサーバ数に対して制限がある可能性がある。

【００５５】

一方、厳密に一回のフレームワークは、これら専用のハードウエアクラスターが処理することができるよりも遥かに大きなクラスターを処理することができるであろう。フレームワークは、専用クラスターの１つから利用可能であるサービス品質の何らかの活用を可能にし、それによってより大きなクラスターを考慮することができる。サービス品質の違いは、例えば、メッセージが２地点間接続などにより確実なプロトコルによって送られるか、又はマルチキャスティングのような信頼性は劣るがリソースに優しいプロトコルによって送られるかを含むであろう。厳密に一回のフレームワークを使用する利点は、ユーザがシステムを特定用途の必要性に適合させることができるように故障許容性と拡張性の均衡を取ることができる点である。

【００５６】

ハードウエアクラスターマシンのような従来技術によるシステムは、第２のマシンによって支援された単一マシン（であるとクラスターには見えるもの）を有することにより、利用可能性の高い解決策を試すことができる。第１のマシンが故障した場合、引継ぎを行う「相棒」がおり、第１のマシン上で実行されていたいかなるソフトウエアも第２のマシンに持ってこられる。

【００５７】

本発明による厳密に一回のフレームワークは、リーダーの故障に対する処理をソフトウエアクラスター内での処理よりも迅速にすることができるように、リードサーバをこれらのハードウエアクラスターの１つにあるサーバに割り当てることができる。しかし、このリードサーバは、サーバがハードウエアクラスター内か又はソフトウエアクラスター内かを問わず、これらのサーバにリースを分配することができる。この構成は、より迅速なリードサーバの回復をもたらし、同時にハードウエアクラスターよりも大きいが依然としてハードウエアクラスターを含むソフトウエアクラスターを考慮することができる。

【００５８】

１つのこのようなシステム２００を図２に示す。ハードウエアクラスター２１８は、複数のサーバ２２０、２２２、及び２２４を包含する単一マシンを含むことができる。効率を

向上させることができるように、ハードウエアクラスターを使用してリードサーバ２２０をそのマシン上のサーバの中から選択することができる。リードサーバ２２０が選択された状態で、このリードサーバは、ソフトウエアクラスター２１０の内側又は外側に位置することができる、ファイルシステム２１２内のオブジェクト２１４に対するホスト２０６を選択することができる。ファイルシステム２１２自体は、永続性をもたらすことができるように、オブジェクト２１４をファイルシステムの別のディスク上の第２のオブジェクト２１６に複製することができる。オブジェクト２１４は、ホスト２０６上でキャッシュに入れられたオブジェクトのコピー２０８と共に、新しいホスト２０６によってファイルシステム２１２から引き抜くことができる。サーバ２０６、２１６、及び２２０のようなサーバにブラウザ又はクライアント２０２からネットワーク２０４を通じて要求が受信された時、そのサーバは、サーバがオブジェクト２０８のキャッシュに入れられたコピーへのアクセスを必要とする場合、ホストサーバ２０６に連絡することを知っていることになる。

【００５９】

このようなシステム５００を使用する１つの方法を図５に示す。ハードウエアクラスター５０２のアルゴリズムを使用してリードサーバを選択する。このアルゴリズムは、例えば、ハードウエアクラスターマシンの専用アルゴリズムとすることができ、又は、ハードウエアクラスターサーバのみに亘る合意を必要とする分散型合意アルゴリズムとしてもよい。その後、リードサーバ５０４と共に「Ｐａｘｏｓ」アルゴリズムのような分散型合意アルゴリズムを使用してホストサーバを予め選択することができる。その後、ハードウエアクラスター５０６を包含するソフトウエアクラスター内の他のサーバに対して、予め選択されたホストのアイデンティティをマルチキャストすることができる。リードサーバは、現在作動中でクラスター５０８に接続されている各サーバから承認又は否認を受信することができる。サーバが予め選択されたホストサーバを承認した場合、予め選択されたホストが今や項目をホスティングすることをサーバに知らせるコミットメッセージがクラスターサーバにマルチキャストされ、そうでなくサーバが承認しなかった場合は、新しいホストが予め選択されて処理が再び始まる（５１０）。

【００６０】

厳密に一回のフレームワークは、例えばトランザクションログ又はキャッシュを処理するために使用することができる。また、このようなフレームワークを使用して、例えば、管理サーバを厳密に一回のオブジェクトとして定め、管理サーバが決して故障しないようにこの管理サーバをリースすることができる。

【００６１】

図６は、オブジェクト６０８が「Ｊａｖａ（登録商標）メッセージサービス（ＪＭＳ）」６１２のためのメッセージストアとして機能する、本発明によるクラスターシステム６００の別の例を示す。クラスター６１０内の全てのサーバ６０６、６１４、及び６１６は、「ＪＭＳ」を使用することができるが、それらは、ネットワーク６０４を通じてメッセージをメッセージストア６０８に送り、メッセージストア６０８からいかなるメッセージも収集する必要がある。クラスター６１０内のサーバのホストサーバ６０６は、メッセージストア６０８を「所有する」又は「ホスティングする」ことになる。クライアント又はブラウザ６０２は、クラスター６１０内のサーバ６１６に向けられた要求をネットワーク６０４に出すことができる。そのサーバ６１６は、ネットワーク６０４を通じてホストサーバ６０６上のメッセージストア６０８にメッセージを送ることによってのみ「ＪＭＳ」にアクセスすることができる。

【００６２】

図７は、本発明の構成要素に対して又は本発明の方法を実施するために使用することができるコンピュータシステムのブロック図７００を示す。図７のコンピュータシステムは、プロセッサユニット７０４及びメインメモリ７０２を含む。プロセッサユニット７０４は、単一のマイクロプロセッサを含むことができ、又は、コンピュータシステムを多重プロセッサシステムとして構成するための複数のマイクロプロセッサを含むことができる。メ

インメモリ７０２は、部分的に、プロセッサユニット７０４による実行のための命令及びデータを記憶する。本発明が全体的又は部分的にソフトウエアに実装された場合、メインメモリ７０２は、作動時に実行可能なコードを記憶することができる。メインメモリ７０２は、ダイナミック・ランダム・アクセス・メモリ（ＤＲＡＭ）、高速キャッシュメモリ、及び、当業技術で公知の他の種類のメモリの列を含むことができる。

【００６３】

図７のシステムは、更に、大容量記憶装置７０６、周辺装置７０８、ユーザ入力装置７１２、携帯記憶媒体ドライブ７１４、グラフィックサブシステム７１８、及び出力ディスプレイ７１６を含む。簡素化のために、図７に示す構成要素は、単一バスを通じて接続されるものとして示されている。しかし、当業者には明らかであろうが、構成要素は、１つ又はそれ以上のデータ搬送手段を通じて接続することができる。例えば、プロセッサユニット７０４及びメインメモリ７０２は、ローカルマイクロプロセッサバスを通じて接続することができ、大容量記憶装置７０６、周辺装置７０８、携帯記憶媒体ドライブ７１４、及びグラフィックサブシステム７１８は、１つ又はそれ以上の入力／出力（Ｉ／Ｏ）バスを通じて接続することができる。磁気ディスクドライブ、光学ディスクドライブ、及び、当業技術で公知の他のドライブを用いて実施することができる大容量記憶装置７０６は、プロセッサユニット７０４によって使用されるデータ及び命令を記憶するための不揮発性記憶装置である。一実施形態においては、大容量記憶装置７０６は、本発明を実施するためのソフトウエアを、メインメモリ７０２に読み込む目的で記憶する。

【００６４】

携帯記憶媒体ドライブ７１４は、フレキシブルディスクのような携帯不揮発性記憶媒体と共に作動し、図７のコンピュータシステムに対してデータ及びコードを入力及び出力する。一実施形態においては、本発明を実行するためのシステムソフトウエアは、このような携帯媒体上に記憶され、携帯記憶媒体ドライブ７１４を通じてコンピュータシステムに入力される。周辺装置７０８は、付加的な機能性をコンピュータシステムに追加するための入力／出力（Ｉ／Ｏ）インタフェースのような任意の種類のコンピュータサポート装置を含むことができる。例えば、周辺装置７０８は、コンピュータシステムをネットワークに接続するためのネットワークインタフェース、及び、モデム、ルータ、又は当業技術で公知の他のハードウエアのような他のネットワーキングハードウエアを含むことができる。

【００６５】

ユーザ入力装置７１２は、ユーザインタフェースの一部分を形成する。ユーザ入力装置７１２は、英数字及び他の情報を入力するための英数字キーパッド、又は、マウス、トラックボール、スタイラス、又はカーソル方向キーのようなポインティング装置を含むことができる。テキスト及びグラフィック情報を表示するために、図７のコンピュータシステムは、グラフィックサブシステム７１８及び出力ディスプレイ７１６を含む。出力ディスプレイ７１６は、ブラウン管（ＣＲＴ）ディスプレイ、液晶ディスプレイ（ＬＣＤ）、又は他の適切な表示装置を含むことができる。グラフィックサブシステム７１８は、テキスト及びグラフィック情報を受信し、ディスプレイ７１６に出力するためにその情報を処理する。更に、図７のシステムは出力装置７１０を含む。適切な出力装置の例には、スピーカ、プリンタ、ネットワークインタフェース、モニタ、及び当業技術で公知の他の出力装置が含まれる。

【００６６】

図７のコンピュータシステムに含まれる構成要素は、本発明のいくつかの実施形態と共に使用するのに適切なコンピュータシステムで一般的に見られるものであり、当業技術で公知のコンピュータ構成要素の広い部類を表すものである。すなわち、図７のコンピュータシステムは、パーソナルコンピュータ、ワークステーション、サーバ、ミニコンピュータ、メインフレームコンピュータ、又は任意の他のコンピュータ装置とすることができる。コンピュータシステム７００はまた、異なるバス構成、ネットワーク化されたプラットフォーム、及び多重プロセッサプラットフォームなどを組み込むことができる。「ユニックス」、「リナックス」、「ウインドウズ」、「マッキントッシュＯＳ」、「パームＯＳ」

、及び他の適切なオペレーティングシステムを含む様々なオペレーティングシステムを使用することができる。

【００６７】

本発明の好ましい実施形態の以上の説明は、例証と説明の目的で為されたものである。それは、限定的つまり開示された正確な形態に本発明を限定するものではない。明らかに、多くの修正及び変形が当業者には明らかであろう。これらの実施形態は、本発明の原理及びその実際的な応用を最も良く説明し、それによって他の当業者が様々な実施形態に関して、かつ予想される特定の使用に適切な様々な修正と共に本発明を理解することを可能にするように選択して説明したものである。本発明の範囲は、特許請求の範囲及びその均等物によって規定されるものとする。

【図面の簡単な説明】

【００６８】

【図１】本発明の一実施形態による分散型オブジェクトシステムの図である。

【図２】本発明の一実施形態による別の分散型オブジェクトシステムの図である。

【図３】本発明によるホストサーバを選択する方法の流れ図である。

【図４】本発明による新しいホストサーバを選択する方法の流れ図である。

【図５】本発明によるリードサーバを利用する方法の流れ図である。

【図６】本発明の一実施形態による「ＪＭＳ」メッセージ記憶システムの図である。

【図７】本発明に従って使用することができるコンピュータシステムの構成要素を示すブロック図である。

【符号の説明】

【００６９】

２００　システム
２０６　ホストサーバ
２０８、２１４、２１６　オブジェクト
２１０　ソフトウエアクラスター
２１２　ファイルシステム
２１８　ハードウエアクラスター
２２０　リードサーバ

【図１】

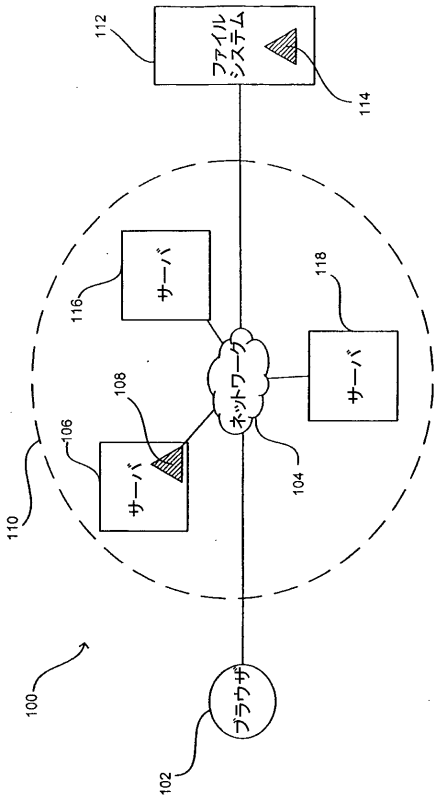


Figure 1

【図２】

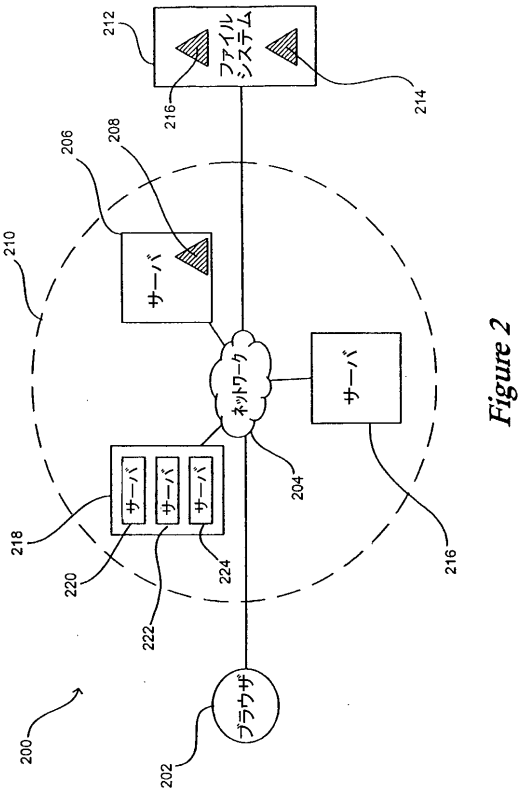


Figure 2

【図３】

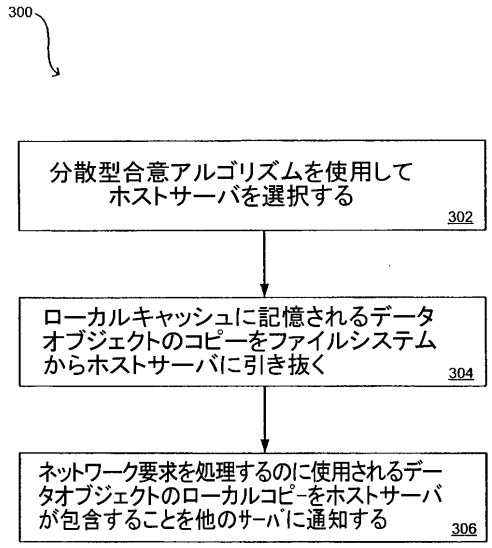


Figure 3

【図４】

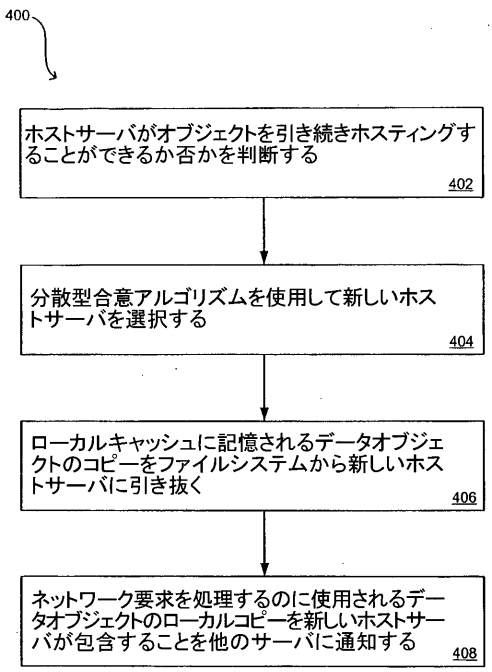


Figure 4

【図５】

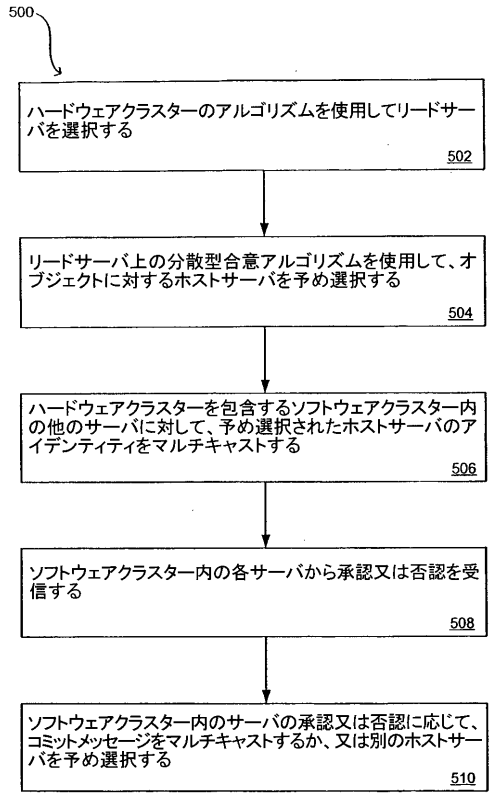


Figure 5

【図６】

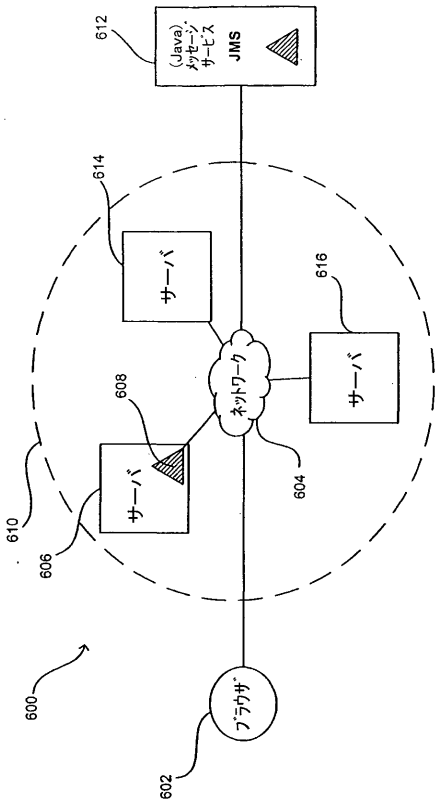


Figure 6

【図７】

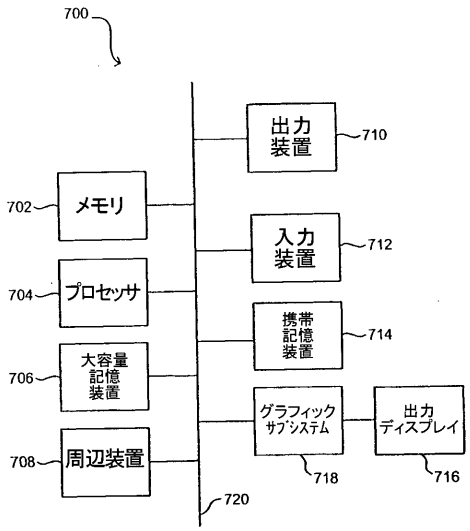


Figure 7

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 March 2003 (20.03.2003)

PCT

(10) International Publication Number
WO 03/023633 A1

(51) International Patent Classification[7]: G06F 15/16, 15/173

(21) International Application Number: PCT/US02/28199

(22) International Filing Date:
5 September 2002 (05.09.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/317,718	6 September 2001 (06.09.2001)	US
60/317,566	6 September 2001 (06.09.2001)	US
10/234,693	4 September 2002 (04.09.2002)	US
10/234,597	4 September 2002 (04.09.2002)	US

(71) Applicant: **BEA SYSTEMS, INC.** [US/US]; 2315 North First Street, San Jose, CA 95131 (US).

(72) Inventors: **JACOBS, Dean, Bernard**; 1747 Madera Street, Berkeley, CA 94707 (US). **HALPERN, Eric**; 160 Delmar Street, San Francisco, CA 94117 (US).

(74) Agents: **MEYER, Sheldon, R.** et al.; Fliesler Dubb Meyer and Lovejoy LLP, Suite 400, Four Embarcadero Center, San Francisco, CA 94111-4156 (US).

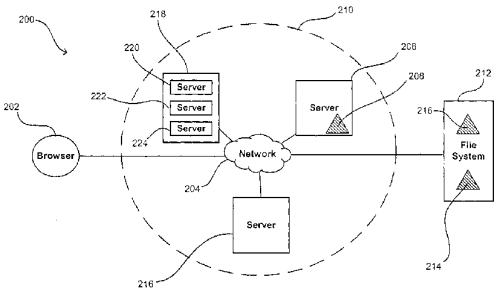
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
with international search report

[Continued on next page]

(54) **Title:** EXACTLY ONCE CACHE FRAMEWORK



(57) **Abstract:** A system for managing objects in a clustered network includes a file system (212) containing at least one copy of a data object (208). The system can include several clustered servers in communication with the file system (212). A lead server is selected, which contains a distributed consensus algorithm for selecting a host server (206), and which utilizes multicasting while executing rounds of the algorithm. The selected host server (206) can contain a copy of the data object (208), such as in local cache, providing access to the local copy (208) to any other server in the cluster. Any change made to an item hosted by the host server (206) can also be updated in the file system (212). If the host server (206) becomes unable to host the object, a new host can be chosen using the distributed consensus a algorithm. The other severs (216, 218) are then notified of the new host by multicast messaging.

WO 03/023633 A1

WO 03/023633 A1

before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

EXACTLY ONCE CACHE FRAMEWORK

CLAIM OF PRIORITY

[0001] This application claims priority to the following applications which are incorporated herein:

[0002] U.S. Provisional Patent Application No. 60/317,718 entitled "EXACTLY ONCE CACHE FRAMEWORK," by Dean Bernard Jacobs and Eric Halpern, filed September 6, 2001.

[0003] U.S. Patent Application entitled "EXACTLY ONCE CACHE FRAMEWORK," by Dean Bernard Jacobs and Eric Halpern, filed September 4, 2002.

[0004] U. S. Provisional Patent Application No. 60/317,566 entitled "EXACTLY ONCE JMS COMMUNICATION," BY Dean Bernard Jacobs and Eric Halpern, filed September 6, 2001.

[0005] U. S. Patent Application entitled "EXACTLY ONCE JMS COMMUNICATION," by Dean Bernard Jacobs and Eric Halpern, filed September 4, 2002.

COPYRIGHT NOTICE

[0006] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document of the patent

disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

CROSS-REFERENCED CASES:

[0007] The following U.S. Patent Application is cross-referenced and incorporated herein by reference:

[0008] U.S. Patent Application No. 60/305,986 entitled "DATA REPLICATION PROTOCOL," by Dean Bernard Jacobs, Reto Kramer, and Ananthan Bala Srinivasan, filed July 16, 2001.

TECHNICAL FIELD

[0009] The present invention is related to technology for distributing objects among servers in a network cluster.

BACKGROUND

[0010] In distributed computer systems, it is often the case that several servers and/or networking nodes need to work together. These servers and nodes have to be coordinated, as there is typically networking information that needs to be shared among the machines in order to allow them to function as a single entity. Typical approaches to machine coordination can be very expensive in terms of resources and efficiency.

[0011] In general, some synchronization is required for the nodes to agree, as there may be several messages passing between the nodes. This requirement for synchronization may, however, be undesirable in a clustered networking

environment. Many clustered environments simply avoid imposing any such synchronization requirement. There are applications, however, where agreement is necessary.

[0012] In one case where agreement is needed, a device can exist to which a cluster may want exclusive access. One such device is a transaction log on a file system. Whenever a transaction is in progress, there are certain objects that need to be saved in a persistent way, such that if a failure occurs those persistently-saved objects can be recovered.

[0013] For these objects that need to be saved in one place, there is typically a transaction monitor that runs on each server in that cluster or domain, which then uses a local file system to access the object. Each server can have its own transaction manager such that there is little to no problem with persistence. There is then also no need for coordination, as each server has its own transaction manager.

[0014] For example, there can be a cluster including three servers, each server having a transaction manager. One of those servers can experience a failure or other problem causing the server to be unavailable to the cluster. Because the failed server is the only server having access to a particular transaction log, none of the transactions in that particular log can be recovered until the server is again available to the cluster. Recovery of the log can be difficult or at least inefficient, as a problem with the server can take a significant amount of time to fix. Significant server problems can include such occurrences as the shorting out of a motherboard on the server or a power supply being burnt out.

BRIEF SUMMARY

[0015] The present invention includes a system for managing objects, such as can be stored in servers on a network or in a cluster. The system includes a data source, application, or service, such as a file system or Java Message Service component, which can be located inside or outside of a cluster. The system can include several servers in communication with the file system or application, such as through a high-speed network connection.

[0016] The system includes a lead server, such as can be agreed upon by the other servers. The lead server can be contained in a hardware cluster or in a software cluster. The system can include an algorithm for selecting a lead server from among the servers, such as an algorithm built into a hardware cluster machine. The lead server in turn will contain a distributed consensus algorithm for selecting a host server, such as a Paxos algorithm. The algorithm used for selecting the lead server can be different from, or the same as, the algorithm used to select the host server.

[0017] The host server can contain a copy of the item or object, such as can be stored in local cache. The host server can provide local copy access to any server on the network or in a cluster. The host server can also provide the only access point to an object stored in a file system, or the only access point to an application or service. Any change made to an item cached, hosted, or owned by the host server can also be updated in the file system, application, or service.

[0018] If the host server becomes unable to host the object, a new host can be chosen using a distributed consensus algorithm. The new host can then pull the necessary data for the object from the file system or service. The other servers in the cluster can be notified that a new server is hosting the object. The servers can

be notified by any appropriate means, such as by point-to-point connections or by multicasting.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] Figure 1 is a diagram of a distributed object system in accordance with one embodiment of the present invention.

[0020] Figure 2 is a diagram of another distributed object system in accordance with one embodiment of the present invention.

[0021] Figure 3 is a flowchart of a method for selecting a host server in accordance with the present invention.

[0022] Figure **4** is a flowchart of a method for selecting a new host server in accordance with the present invention.

[0023] Figure 5 is a flowchart of a method for utilizing a lead server in accordance with the present invention.

[0024] Figure 6 is a diagram of JMS message store system in accordance with one embodiment of the present invention.

[0025] Figure 7 is a block diagram depicting components of a computing system that can be used in accordance with the present invention.

DETAILED DESCRIPTION

[0026] Systems in accordance with the present invention can provide solutions to availability issues, such as when a server owning a data object becomes unavailable to a server cluster. One such solution allows for another server in the cluster to take over the ownership of the data object. A problem arises, however, in making the data object accessible to both servers without having to replicate the

data object on both.

[0027] If a file system, data store, or database (all collectively referred to hereinafter as "file system") is used by the cluster to persistently store data, and the file system is accessible from more than one server, the second server can automatically take over the task of data object access if the first server owning that object encounters a problem. Alternatively, there can be an algorithm utilized by the cluster or a server in the cluster to instruct a server to take ownership of the item. Another fundamental problem, however, involves getting the cluster to agree on which server now owns the resource or object, or achieving a "consensus" amongst the servers.

[0028] **Figure 1** shows one example of a cluster system **100** in accordance with the present invention, where an object such as a transaction log **114** is stored in a file system **112**. The file system **112** is accessible to all servers **106**, **116**, **118** in the cluster **110**, but only one of these servers can access the log **114** at a time. A host server **106** among the servers in the cluster **110** will "own" or "host" the log **114**, such as by storing a copy **108** of the log **114** or by providing all access to the log **114** in the file system **112**. Any other server **116**, **118** in the cluster **110** can access the copy **108** of the log, and/or can access the log **114** through the hosting server **106**. For example, a client or browser **102** can make a request to a network **104** that is directed to server **116** in cluster **110**. That server can access the copy **108** of the transaction log on the host server **106** through the network **104**. If the transaction log needs to be updated, the copy **108** can be updated along with the original log **114** on the file system **112**.

[0029] A server can "own" or "host" a data object when, for example, it acts as a repository for the object, such as by storing a copy of the data object in

local cache and making that copy available to other servers in the cluster, or by being the sole server having direct access to an object in a file system, service, or application, such that all other servers in the cluster must access that object through the hosting server. This ensures that an object exists "exactly once" in the server cluster.

[0030] **Figure 3** shows one process **300** that can be used to establish the hosting of an object. A host server can be selected using a distributed consensus algorithm **302**, such as a Paxos algorithm. Such an algorithm is referred to as a "distributed consensus" algorithm because servers in a cluster must generally agree, or come to a consensus, as to how to distribute objects amongst the cluster servers.

[0031] If a hosted object is, for example, to be cached on the hosting server, a copy of the data object can be pulled from a file system to the host server and stored as an object in local cache **304**. The other servers on the network or in the appropriate cluster are then notified, such as by the hosting server, that a local copy of the object exists on the hosting server, and that the local copy should be used in processing future network requests **306**.

[0032] In a Paxos algorithm, one example of a distributed consensus algorithm, a server can be selected to act as a host or lead server by a network server, the network server leading a series of "consensus rounds." In each of these consensus rounds, a new host or lead server is proposed. Rounds continue until one of the proposed servers is accepted by a majority or quorum of the servers. Any server can propose a host or lead server by initiating a round, although a system can be configured such that a lead server always initiates a round for a host server selection. Rounds for different selections can be carried out at the same time. Therefore, a round selection can be identified by a round number or pair of

values, such as a pair with one value referring to the round and one value referring to the server leading the round.

[0033] The steps for one such round are as follows, although other steps and/or approaches may be appropriate for certain situations or applications. First, a round can be initiated by a leader sending a "collect" message to other servers in the cluster. A collect message collects information from servers in the cluster regarding previously conducted rounds in which those servers participated. If there have been previous consensus rounds for this particular selection process, the collect message also informs the servers not to commit selections from previous rounds. Once the leader has gathered responses from at least half of the cluster servers, for example, the leader can decide the value to propose for the next round and send this proposal to the cluster servers as a "begin" message. In order for the leader to choose a value to propose in this approach, it is necessary to receive the initial value information from the servers.

[0034] Once a server receives a begin message from the leader, it can respond by sending an "accept" message, stating that the server accepts the proposed host/lead server. If the leader receives accept messages from a majority or quorum of servers, the leader sets its output value to the value proposed in the round. If the leader does not receive majority or quorum acceptance ("consensus") within a specified period of time, the leader can begin a new round. If the leader receives consensus, the leader can notify the cluster or network servers that the servers should commit to the chosen server. This notification can be broadcast to the network servers by any appropriate broadcasting technology, such as through point-to-point connections or multicasting.

[0035] The agreement condition of the consensus approach can be

guaranteed by proposing selections that utilize information about previous rounds. This information can be required to come from at least a majority of the network servers, so that for any two rounds there is at least one server that participated in both rounds.

[0036] The leader can choose a value for the new round by asking each server for the number of the latest round in which the server accepted a value, possibly also asking for the accepted value. Once the leader gets this information from a majority or quorum of the servers, it can choose a value for the new round that is equal to the value of the latest round among the responses. The leader can also choose an initial value if none of the servers were involved in a previous round. If the leader receives a response that the last accepted round is x, for example, and the current round is y, the server can imply that no round between x and y would be accepted, in order to maintain consistency.

[0037] A sample interaction between a round leader and a network server involves the following messages:

[0038] (1) "Collect" - a message is sent to the servers that a new round "r" is starting. The message can take the form of m=("Collect", r).

[0039] (2) "Last" - a message is sent to the leader from a network server providing the last round accepted, "a", and the value of that round, "v". The message can take the form of m=("Last", r, a, v).

[0040] (3) "Begin" - a message is sent to the servers announcing the value for round r. The message can take the form of m=("Begin", r, v).

[0041] (4) "Accept" - a message is sent to the leader from the servers accepting the value for round r. The message can take the form of m=("Accept", r).

[0042] (5) "Success" - a message is sent to the servers announcing the selection of value *v* for round *r*. The message can take the form of *m*=("Success", *r*, *v*).

[0043] (6) "Ack" - a message is sent to the leader from a server acknowledging that the server received the decision for round *r*. The message can take the form of *m*=("Ack", *r*).

[0044] There can be a file system that is separated from the servers, located either inside or outside of a hardware or software cluster. This file system can persistently store the transaction log, such as by storing the log on a first disk and replicating the log to a second disk within the file system. If the first disk crashes, the file system can hide the crash from the cluster and/or server and get the log information from the second disk. The file system can also choose to replicate the log to a third disk, which can serve as a backup to the second disk.

[0045] From the perspective of a server in the cluster, the file system can be a single resource. In one embodiment, the server may only care that a single server owns the file system at any time.

[0046] Another example of a system in accordance with the present invention involves caching in a server cluster. It may be desirable in a clustered environment, such as for reasons of network performance, to have a single cache represent a data object to servers in the cluster. Keeping items in a single cache can be advantageous, as servers in the cluster can access the cache without needing to continually return to persistent storage. Being able to pull an item already in memory can greatly increase the efficiency of such a system, as hits to a database or file system can be relatively time intensive.

[0047] One problem with a single cache, however, is that it may be

necessary to ensure that the object stored in memory is the same as that which is stored on a disk of the file system. One reason for requiring such consistency is to ensure that any operations or calculations done on a cached item produce the correct result. Another reason is that it can be necessary to restore the cache from the file system in the event that the cache crashes or becomes otherwise tainted or unavailable.

[0048] There can be at least two primary ways to handle this type of caching in a cluster, although other ways may work at least as well for certain applications. One way is to replicate the cache in multiple places. This approach can be problematic, as any change to an item being cached requires that all servers replicating the cache agree to, or are at least aware of, the change. This can prove to be very expensive in terms of resources and performance.

[0049] An alternative approach in accordance with the present invention assigns a particular server to be the owner of a cache in the cluster, and all access to the cache goes through that particular server. Any server in a cluster can host such a cache. Each server can host one, several, or no caches. The caches can be hosted on a single server, or spread out among some or all of the servers in the cluster. The cluster itself can be any appropriate cluster, such as a hardware cluster or a group of servers designated by a software application to be in a given "software" cluster.

[0050] It may be possible to think of either example, a transaction log and/or a cache, as a type of object that sits somewhere on a system. It may be desirable to ensure that any such object exists only once in a cluster, and that the object is always available. It may also be desirable to ensure that the object can be recovered on another server if the server hosting the object fails, and that the object

will be available to the cluster.

[0051] One method **400** for recovery is shown in **Figure 4**. In this method, a determination is made whether the host server can continue to host an object **402**, such as whether the server is still available to the network. If not, a new host is selected using a distributed consensus algorithm. This selection may be performed according to the method used to select the original host **404**. A copy of the data object is pulled from a file system to the new host, and can be stored in a local cache **406**. The other servers on the network or in the appropriate cluster are notified that the new host server contains a local copy of the object, and that the local copy should be used in processing any future network requests **408**.

[0052] Systems and methods in accordance with the present invention can define objects that exist in exactly one place in a cluster, and can ensure that those objects always exist. From a server's perspective, it may not matter whether an object such as a transaction log is mirrored or replicated, such as by a file system. From the server's perspective, there is always one persistent storage accessible by any server in the cluster. The system can periodically check for the existence of an object, or may assign ownership of objects for short periods of time such that an object will be reassigned frequently to ensure existence on some machine on the network or in the cluster.

[0053] A hardware cluster can comprise a bank of machines, each machine being capable of running multiple servers. There can also be a file system behind each machine. Servers in a hardware cluster are typically hardwired, such that they are able to more quickly make decisions and deal with server faults within the hardware cluster. Hardware clusters can be limited in size to the physical hardware of the machine containing the servers. Servers in a hardware cluster can be used

as servers in a software cluster, and can also comprise network servers, as the individual servers on the machines are available to the network.

[0054] The shared file system for one of these machines can be available to all servers in a cluster, such as through a high-speed network. The file system can also be redundant. In one embodiment, this redundancy is implemented through the use of multiple data disks for the file system. In such a redundant implementation, an object can be replicated across multiple disks any time the object is written to the file system. Such a file system, when viewed as a "black box," can be able to withstand failures of any of the disks and still provide access to a data item from any of the servers in the cluster.

[0055] A framework in accordance with the present invention, referred to as an "exactly-once" framework, can be built on the assumption that these objects kept in memory are always backed by a reliable, persistent storage mechanism. For example, there can be an object that represents a transaction log. Whenever a call is made to the object, the corresponding transaction log can be updated. This may include a call that either reads from, or writes to, a database. An object representing that transaction log can be sitting on one of the servers in the cluster, such as the host server. An exactly-once framework can ensure that, as long as at least one of the servers in the cluster is up and running, a server will be able to take over ownership of the log if another server fails.

[0056] There may be one object that represents a cache. Every time the cache is updated, the update can also be written back to persistent storage. When one of the servers needs to use a data item, that server can be required to go through this object. If the server hosting the object that represents the cache fails, the object can be resurrected on another server. The resurrected object can pull all

the necessary information from persistent storage.

[0057] An exactly-once framework can act as a memory buffer for use by the cluster. The framework can provide a single cache representing data in the system that is backed by a reliable, persistent storage. Whenever data is read from the cache, the read can be done without needing to access persistent storage. When an update is written to cache, however, it can be necessary to write back through the persistent storage, such that the system can recover if there is a failure.

[0058] One important aspect of an exactly-once framework involves the way in which the approach is abstracted, which can vary depending upon the application and/or implementation. A new type of distributed object is created, such as can be referred to as an "exactly-once object." An exactly-once object can be, for example, a locally-cached copy of a data item in a file system, or the sole access point to such a data item for servers in a cluster. Underlying techniques for implementing this abstraction can also be important.

[0059] Systems of the present invention can utilize any of a number of methods useful for distributed consensus, such as a method using the aforementioned Paxos algorithm. Such an algorithm can be selected which provides an efficient way for multiple nodes and/or distributed nodes to agree on one value of an object. The algorithm can be chosen to work even if nodes fail and/or return during an agreement process.

[0060] A typical approach to network clustering utilizes reliable broadcasting, where every message is guaranteed to be delivered to its intended recipient, or at least delivered to every intended functioning server. This approach can make it very difficult to parallelize a system, as reliable broadcasting requires a recipient to acknowledge a message before moving on to the next message or

recipient. A distributed algorithm utilizing multicasting may reduce the number of guarantees, as multicasting does not guarantee that all servers receive a message. Multicasting does simplify the approach such that the system can engage in parallel processing, however, as a single message can be multicast to all the cluster servers concurrently without waiting for a response from each server. A server that does not receive a multicast message can pull the information from the lead server, or another cluster server or network server, at a later time. As used herein, a network server can refer to any server on the network, whether in a hardware cluster, in a software cluster, or outside of any cluster.

[0061] An important aspect of an exactly-once architecture is that consensus difficulties are reduced. In accordance with the present invention, the performance of a distributed consensus implementation can be improved by using multicast messaging with a distributed consensus algorithm. This approach can allow for minimizing the message exchange and/or network traffic required for all the servers to agree.

[0062] When multicasting, one of several approaches can be taken. In a first approach, which may be referred to as "one-phase distribution," a lead server can multicast a message to all other servers on the network, such as may be used in a round of a Paxos algorithm, or used to state that a new host has been selected for an object. In this approach, the lead server only needs to send one message, which can be passed to any server available on the network. If a server is temporarily off the network, the server can request the identification of the new host after coming back onto the network.

[0063] Using another multicast approach, which may be referred to as a "two-phase distribution," the lead server can pre-select a host server using an

appropriate algorithm. Before assigning an object to that host, however, the lead server can contact every other server in the cluster to determine whether the servers agree with the choice of the new host server. The lead server can contact each server by a point-to-point connection, or can send out a multicast request and then wait for each server to respond. If the servers do not agree on the selection of the host, the lead server can pre-select a new host using the algorithm. The lead server would then send out another multicast request with the identity of the newly pre-selected host in another round.

[0064] If every server agrees to the pre-selected host, the lead server can assign the object to the host server. The lead server can then multicast a commit message, informing the servers that the new change has taken effect and the servers should update their information accordingly.

[0065] An exactly-once framework can also utilize a "leasing" mechanism. In using such a mechanism, an algorithm can be used to get the cluster servers to agree on a lead server, such as by using distributed consensus. Once selected, that lead server can be responsible for assigning exactly-once objects to various servers in the cluster. The system can be set up such that the cluster servers will always agree on a new leader if an existing lead server fails.

[0066] While the lead server is active, the lead server can be aware of all the exactly-once objects that need to exist in the system. The lead server can decide which server should host each object, and can then "lease" that object to the selected server. When an object is leased to a server, that server can own or host the object for a certain period of time, such as for the duration of a lease period. The lead server can be configured to periodically renew these leases. This approach can provide a way to ensure that a server will not get its lease renewed

if it fails or becomes disconnected in any way, or is otherwise not operating properly within the cluster.

[0067] Much of the problem with distributed systems under failure is that it is difficult to tell the difference between a server that has failed and one that is simply not responding. Any server that has been somehow cut off the network can no longer host an object. That server will still know, even though it is not available to the cluster, that it can drop its hosting of any object after the lease period. As the server is not available to the cluster, it will not get its lease renewed.

[0068] The lead server also knows that, if the lead server is unable to reach the host server within a certain amount of time, the host server will relinquish its ownership of the object. The lease period can be for any appropriate time, such as for a matter of seconds. The lease period can be the same for all objects in the cluster, or can vary between objects.

[0069] A system using an exactly-once architecture can also be tightened down. Operating systems often provide special machinery that is built closer to the hardware and can offer more control. One problem with this approach, however, is that it can be limited by the hardware available. For example, a hardware cluster of servers can have on the order of 16 servers. Because these systems require some tight hardware coupling, there can be limitations on the number of servers that can be included in the cluster.

[0070] An exactly-once framework, on the other hand, may be able to handle clusters much larger than these proprietary hardware clusters can handle. A framework can allow for some leveraging of the qualities of service that are available from one of the proprietary clusters, thereby allowing for a larger cluster. Differing qualities of service may include, for example, whether messages are sent

by a reliable protocol, such as by point-to-point connections, or are sent by a less reliable but more resource-friendly protocol, such as multicasting. An advantage to using an exactly-once framework is the ability to balance scalability with fault tolerance, such that a user can adapt the system to the needs of a particular application.

[0071] Prior art systems such as hardware cluster machines can attempt high availability solutions by having (what appears to the cluster to be) a single machine backed up by a second machine. If the first machine goes down, there is a "buddy" that takes over, and any software that was running on the first machine is brought up on the second machine.

[0072] An exactly-once framework in accordance with the present invention can assign the lead server to a server in one of these hardware clusters, such that dealing with leader failure can become faster than dealing with it in a software cluster. This lead server can, however, dole out leases to servers whether those servers are in the hardware cluster or the software cluster. This arrangement may provide for faster lead server recovery, while allowing for a software cluster that is larger than, but still includes, the hardware cluster.

[0073] One such system **200** is shown in **Figure 2**. A hardware cluster **218** can comprise a single machine containing multiple servers **220, 222, 224**. The hardware cluster can be used to choose a lead server **220** from among the servers on that machine, such as may improve efficiency. Once a lead server **220** is selected, the lead server can select a host **206** for an object **214** in a file system **212**, which can be located inside or outside of the software cluster **210**. The file system **214** itself can replicate the object **214** to a second object **216** on another disk of the file system, such as may provide persistence. The object **214** can be pulled from

the file system **212** by the new host **206** with a copy **208** of the object cached on the host **206**. When a request is received from a browser or client **202** to a server through the network **204**, such as servers **206**, **216**, and **220**, that server will know to contact host server **206** if the server needs access to the cached copy of the object **208**.

[0074] One method **500** for using such a system is shown in **Figure 5**. The lead server is selected using an algorithm of a hardware cluster **502**. This algorithm may be, for example, a proprietary algorithm of the hardware cluster machine, or may be a distributed consensus algorithm requiring consensus over the hardware cluster servers only. A host server can then be pre-selected using a distributed consensus algorithm with the lead server **504**, such as a Paxos algorithm. The identity of the pre-selected host can then be multicast to the other servers in a software cluster containing the hardware cluster **506**. The lead server can receive approval or disapproval from each server that is presently operational and connected to the cluster **508**. If the servers approve the pre-selected host server, a commit message is multicast to the cluster servers informing the servers that the pre-selected host now hosts the item; otherwise, if the servers do not approve a new host is pre-selected and the process begins again **510**.

[0075] An exactly-once framework can be used, for example, to handle transaction logs or caching. Such a framework can also be used, for example, to define an administration server as an exactly-once object and lease the administration server such that the administration server never goes down.

[0076] **Figure 6** shows another example of a cluster system **600** in accordance with the present invention, where an object **608** acts as a message store for Java Message Service (JMS) **612**. All servers **606**, **614**, **616** in the cluster **610**

can use JMS, but they must send messages to the message store **608** and pick up any messages from the message store **608** through the network **604**. A host server **606** of the servers in the cluster **610** will "own" or "host" the message store **608**. A client or browser **602** can make a request to a network **604** that is directed to server **616** in cluster **610**. That server **616** can access JMS only by sending a message to the message store **608** on the host server **606** through the network **604**.

[0077] **Figure 7** illustrates a block diagram **700** of a computer system which can be used for components of the present invention or to implement methods of the present invention. The computer system of **Figure 7** includes a processor unit **704** and main memory **702**. Processor unit **704** may contain a single microprocessor, or may contain a plurality of microprocessors for configuring the computer system as a multi-processor system. Main memory **702** stores, in part, instructions and data for execution by processor unit **704**. If the present invention is wholly or partially implemented in software, main memory **702** can store the executable code when in operation. Main memory **702** may include banks of dynamic random access memory (DRAM), high speed cache memory, as well as other types of memory known in the art.

[0078] The system of **Figure 7** further includes a mass storage device 706, peripheral devices **708**, user input devices **712**, portable storage medium drives **714**, a graphics subsystem **718**, and an output display **716**. For purposes of simplicity, the components shown in **Figure 7** are depicted as being connected via a single bus **720**. However, as will be apparent to those skilled in the art, the components may be connected through one or more data transport means. For example, processor unit **704** and main memory **702** may be connected via a local microprocessor bus, and the mass storage device **706**, peripheral devices **708**,

portable storage medium drives **714**, and graphics subsystem **718** may be connected via one or more input/output (I/O) buses. Mass storage device **706**, which may be implemented with a magnetic disk drive, optical disk drive, as well as other drives known in the art, is a non-volatile storage device for storing data and instructions for use by processor unit **704**. In one embodiment, mass storage device **706** stores software for implementing the present invention for purposes of loading to main memory **702**.

[0079] Portable storage medium drive **714** operates in conjunction with a portable non-volatile storage medium, such as a floppy disk, to input and output data and code to and from the computer system of **Figure 7**. In one embodiment, the system software for implementing the present invention is stored on such a portable medium, and is input to the computer system via the portable storage medium drive **714**. Peripheral devices **708** may include any type of computer support device, such as an input/output (I/O) interface, to add additional functionality to the computer system. For example, peripheral devices **708** may include a network interface for connecting the computer system to a network, as well as other networking hardware such as modems, routers, or other hardware known in the art.

[0080] User input devices **712** provide a portion of a user interface. User input devices **712** may include an alpha-numeric keypad for inputting alpha-numeric and other information, or a pointing device, such as a mouse, a trackball, stylus, or cursor direction keys. In order to display textual and graphical information, the computer system of **Figure 7** includes graphics subsystem **718** and output display **716**. Output display **716** may include a cathode ray tube (CRT) display, liquid crystal display (LCD) or other suitable display device. Graphics

subsystem **718** receives textual and graphical information, and processes the information for output to display **716**. Additionally, the system of **Figure 7** includes output devices **710**. Examples of suitable output devices include speakers, printers, network interfaces, monitors, and other output devices known in the art.

[0081] The components contained in the computer system of **Figure 7** are those typically found in computer systems suitable for use with certain embodiments of the present invention, and are intended to represent a broad category of such computer components known in the art. Thus, the computer system of **Figure 7** can be a personal computer, workstation, server, minicomputer, mainframe computer, or any other computing device. Computer system **700** can also incorporate different bus configurations, networked platforms, multi-processor platforms, etc. Various operating systems can be used including Unix, Linux, Windows, Macintosh OS, Palm OS, and other suitable operating systems.

[0082] The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to the practitioner skilled in the art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

CLAIMS

What is claimed is:

1. A system for managing objects on a network, comprising:

a plurality of network servers, each network server adapted to communicate with a network data source; and

a lead server in said plurality of network servers, the lead server containing a distributed consensus algorithm for selecting a host server from said plurality of network servers, the host server containing an object related to a data item in the network data source such that any of said plurality of network servers needing to access the data item can access the object on the host server.

2. A system according to claim 1, wherein said network servers are selected from the group consisting of hardware cluster servers and software cluster servers.

3. A system according to claim 1, wherein said distributed consensus algorithm comprises rounds of messages between said lead server and said plurality of servers, the rounds continuing until a majority of said plurality of network servers agrees on the host server.

4. A system according to claim 1, wherein the host server contains a data object comprising a copy of data from the network data source.

5. A system according to claim 1, wherein the host server contains a data object serving as the sole access point for the data item in the network data source.

6. A system according to claim 1, wherein said data item is a transaction log.

7. A system according to claim 1, wherein said distributed consensus algorithm is a Paxos algorithm.

8. A system for managing objects on a network, comprising:

a plurality of network servers, each network server adapted to communicate with a network data source; and

a lead server in said plurality of network servers, the lead server containing a distributed consensus algorithm for selecting a host server from said plurality of network servers, the host server containing a copy of a data item located in the network data source such that any of said plurality of network servers needing to access the data item can access the copy on the host server.

9. A system for managing objects on a network, comprising:

a plurality of network servers, each network server adapted to communicate with a network data source; and

a lead server in said plurality of network servers, the lead server containing a distributed consensus algorithm for selecting a host server from said plurality of network servers, the host server containing the sole access point to a data item located in the network data source such that any of said plurality of network servers needing to access the data item must access the data item through the host server.

10. A system for managing objects on a network, comprising:

a file system containing at least one copy of a data item;

a plurality of servers in communication with the file system;

a lead server in said plurality of servers, the lead server containing a distributed consensus algorithm for selecting a host server from said plurality of servers; and

a host server in said plurality of servers, said host server containing a local copy of the data item, said host server adapted to provide access to the local copy to any of said plurality of servers and update the copy of the data item in the file system any time an update is made to the local copy.

11. A system according to claim 10, wherein said host server is further adapted to store the local copy in a local cache.

12. A system according to claim 10, wherein said plurality of servers comprise a cluster.

13. A system according to claim 10, wherein said file system replicates the data item over multiple disks.

14. A system for managing objects on a network, comprising:

a file system containing at least one copy of a data item;

a plurality of servers in communication with the file system;

a hardware cluster containing hardware cluster servers located in said plurality of servers, said hardware cluster containing a distributed consensus algorithm for selecting a lead server from among said hardware cluster servers;

a lead server in said hardware cluster servers, the lead server containing an

algorithm for selecting a host server from said plurality of servers; and

a host server in said plurality of servers, said host server containing a local copy of the data item, said host server adapted to provide access to the local copy to any of said plurality of servers and update the copy of the data item in the file system any time an update is made to the local copy.

15. A system according to claim 14, wherein said host server is in said hardware cluster.

16. A method for managing objects on a network, comprising:

selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

pulling a copy of a data item from a file system to the host server; and

notifying other network servers that the host server contains a copy of the data item to be used in processing network requests.

17. A method according to claim 16, further comprising the step of:

updating the data item in the file system when the copy on the host server is modified.

18. A method according to claim 16, further comprising the step of:

restricting other network servers to pass through the host server to access the file system.

19. A method according to claim 16, further comprising the step of:

ensuring that only one copy of the data item exists outside the file system.

20. A method according to claim 16, further comprising the step of:
ensuring that one copy of the data item always exists outside the file system.

21. A method according to claim 16, further comprising the step of:
selecting a new host server from among a plurality of network servers using a distributed consensus algorithm if the host server is no longer able to host the object.

22. A method according to claim 16, further comprising the step of:
pulling a copy of a data item from a file system to the a new host server if the host server is no longer able to host the object, the host server selected using the distributed consensus algorithm.

23. A method according to claim 16, further comprising the step of:
notifying other network servers that a new host server contains a copy of the data item to be used in processing network requests if the host server is no longer able to host the object.

24. A framework for managing objects on a network, comprising:
a plurality of servers, each server capable of caching a data object;
a file system containing at least one copy of a data item;
a distributed consensus algorithm for selecting a host server from among said plurality of servers, the host server to cache a copy of the data object; and

a distribution system for notifying servers on the network that the host computer contains a copy of the data object.

25. A method for leasing an object to a server on a network, comprising:
selecting a host server from among a plurality of network servers using a distributed consensus algorithm;
assigning a data object to the host server, the host server assigned to provide sole access to a data item for a specific period of time;
pulling a copy of a data item from a file system to the host server; and
notifying other network servers that the host server contains a copy of the data item to be used in processing network requests.

26. A method according to claim 25, further comprising the step of:
assigning the data object to the host server for another period of time once the specific period of time expires.

27. A method according to claim 25, further comprising the step of:
assigning the data object to a new host server for another specific period of time once the specific period of time expires on the host server.

28. A method for leasing an object to a server on a network, comprising:
selecting a lead server from among a plurality of hardware cluster servers in a hardware cluster;
selecting a host server from among a plurality of network servers using a

distributed consensus algorithm on said lead server;

assigning a data object to the host server, the host server assigned to provide sole access to a data item for a specific period of time;

pulling a copy of a data item from a file system to the host server; and

notifying other network servers that the host server contains a copy of the data item to be used in processing network requests.

29. A method for assigning ownership of an object on a network, comprising:

selecting a lead server from among a plurality of hardware cluster servers in a hardware cluster;

selecting a host server from among a plurality of network servers using a distributed consensus algorithm on said lead server;

assigning a data object to the host server, the host server assigned to provide sole access to a data object on the network;

pulling a copy of a data object from a file system to the host server; and

notifying other network servers that the host server contains a copy of the data object to be used in processing network requests.

30. A method for hosting Java Messenger Service (JMS) on a network, comprising:

selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

assigning a JMS object the host server, the JMS object comprising a JMS message store providing the sole access point and message queue for JMS over the network; and

notifying the network servers that the host server is hosting the sole JMS

message store.

31. A method according to claim 30, further comprising the step of:

checking the JMS message store for messages intended for one of the plurality of network servers.

32. A method according to claim 30, further comprising the step of:

sending a JMS message from a network server to the JMS message store on the host server.

33. A method according to claim 30, further comprising the step of:

sending messages in the JMS message store on the host server to a JMS component.

34. A method for ensuring the existence of an object in a cluster, comprising:

providing access to a data object using a host server in a plurality of servers;

selecting a new host server from among the plurality of servers using a distributed consensus algorithm if the host server is unable to provide access to the data object;

pulling information to the new host server needed to provide access to the data object; and

notifying other servers in said plurality of servers that a new host server is providing access to the data object.

35. A method for ensuring the availability of an administration server in a cluster,

comprising:

selecting a lead server from among a plurality of servers;

selecting an administration server from among a plurality of servers using a distributed consensus algorithm on the lead server;

pulling administration information from a data source to the administration server and updating administration information in the data source in order to coordinate information in the data source and on the administration server; and

notifying other servers in the cluster of the identity of the administration server.

36. A method according to claim 35, wherein the step of pulling administration information from a data source comprises pulling administration information from a file system.

37. A method for distributing objects in a cluster, comprising:

selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

assigning a data object to the host server, the host server assigned to provide sole access to a data item; and

multicasting a notification to other servers in the cluster that the host server contains a copy of the data item to be used in processing network requests.

38. A method for distributing objects in a cluster, comprising:

selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

contacting each server in the cluster to determine whether the selected host is acceptable to that server; and

multicasting a notification to other servers in the cluster to commit the selection of a new host server if all servers in the cluster agree that the selected host is acceptable.

39. A computer-readable medium, comprising:

means for selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

means for assigning a data object to the host server, the host server assigned to provide sole access to a data item;

means for pulling a copy of a data item from a file system to the host server; and

means for notifying other network servers that the host server contains a copy of the data item to be used in processing network requests.

40. A computer program product for execution by a server computer for managing objects on a network, comprising:

computer code for selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

computer code for pulling a copy of a data item from a file system to the host server; and

computer code for notifying other network servers that the host server contains a copy of the data item to be used in processing network requests.

41. A system for distributing objects in a cluster, comprising:
means for selecting a host server from among a plurality of network servers using a distributed consensus algorithm;
means for pulling a copy of a data item from a file system to the host server; and
means for notifying other network servers that the host server contains a copy of the data item to be used in processing network requests.

42. A computer system comprising:
a processor;
object code executed by said processor, said object code configured to:
select a host server from among a plurality of network servers using a distributed consensus algorithm;
pull a copy of a data item from a file system to the host server; and
notify other network servers that the host server contains a copy of the data item to be used in processing network requests.

43. A method for managing objects on a network, comprising:
selecting a host server from among a plurality of network servers in a software cluster using a Paxos algorithm; and
assigning a data object to the host server, the data object existing only on the host server in the network.

44. A method according to claim 43, further comprising the step of:
pulling data for the data object from a file system.

45. A method according to claim 43, further comprising the step of:

notifying other network servers that the host server contains an object for the data item to be used in processing network requests.

46. A method according to claim 43, further comprising the step of:

multicasting the identification of the new host server to the other network servers.

47. A method according to claim 43, wherein said step of selecting a host server from among a plurality of network servers in a software cluster using a Paxos algorithm comprises multicasting rounds of information to the other network servers.

48. A method for hosting Java Messenger Service (JMS) on a network, comprising:

selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

assigning a JMS object the host server, the JMS object comprising a JMS message store providing the sole access point and message queue for JMS over the network; and

notifying the network servers that the host server is hosting the sole JMS message store.

49. A method according to claim 48, further comprising the step of:

checking the JMS message store for messages intended for one of the plurality of network servers.

50. A method according to claim 48, further comprising the step of:
sending a JMS message from a network server to the JMS message store on the host server.

51. A method according to claim 48, further comprising the step of:
sending messages in the JMS message store on the host server to a JMS component.

52. A method for ensuring the existence of a JMS object in a cluster, comprising:
providing access to a JMS object using a host server in a plurality of servers, the JMS object comprising a JMS message store providing the sole access point and message queue for JMS over the network;
selecting a new host server from among the plurality of servers using a distributed consensus algorithm if the host server is unable to provide access to the JMS object;
pulling information to the new host server needed to provide access to the JMS object; and
notifying other servers in said plurality of servers that a new host server is providing access to the JMS object.

53. A method for assigning a JMS object to a server in a cluster, comprising:
selecting a host server from among a plurality of network servers using a

distributed consensus algorithm;

contacting each server in the cluster to determine whether the selected host is acceptable to that server; and

multicasting a notification to other servers in the cluster to commit the selection of a new host server if all servers in the cluster agree that the selected host is acceptable.

54. A computer-readable medium, comprising:

means for selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

means for assigning a JMS object to the host server, the host server assigned to provide sole access to JMS; and

means for notifying other network servers that the host server provides sole access to JMS.

55. A computer program product for execution by a server computer for managing objects on a network, comprising:

computer code for selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

computer code for assigning a JMS object to the host server, the host server assigned to provide sole access to JMS; and

computer code for notifying other network servers that the host server provides sole access to JMS.

56. A system for distributing objects in a cluster, comprising:

means for selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

means for assigning a JMS object to the host server, the host server assigned to provide sole access to JMS; and

means for notifying other network servers that the host server provides sole access to JMS.

57. A computer system comprising:

a processor;

object code executed by said processor, said object code configured to:

select a host server from among a plurality of network servers using a distributed consensus algorithm;

assign a JMS object to the host server, the host server assigned to provide sole access to JMS

notify other network servers that the host server provides sole access to JMS.

58. A method for managing a JMS message store on a network, comprising:

selecting a host server from among a plurality of network servers in a software cluster using a Paxos algorithm; and

assigning a JMS message store to the host server, the JMS message store existing only on the host server.

59. A method according to claim 58, further comprising the step of:

notifying other network servers that the host server is hosting a JMS

message store.

60. A method according to claim 58, further comprising the step of:

multicasting the identification of the new host server to the other network servers.

61. A method according to claim 58, wherein said step of selecting a host server from among a plurality of network servers in a software cluster using a Paxos algorithm comprises multicasting rounds of information to the other network servers.

62. A system for managing a JMS object on a network, comprising:

a plurality of network servers; and

a lead server in said plurality of network servers, the lead server containing a distributed consensus algorithm for selecting a host server from said plurality of network servers, the host server containing a JMS object such that any of said plurality of network servers needing to access JMS must access the JMS object on the host server.

63. A system according to claim 62, wherein said network servers are selected from the group consisting of hardware cluster servers and software cluster servers.

64. A system according to claim 62, wherein said distributed consensus algorithm comprises rounds of messages between said lead server and said plurality of servers, the rounds continuing until a majority of said plurality of network servers

agrees on the host server.

65. A system according to claim 48, wherein said distributed consensus algorithm is a Paxos algorithm.

66. A system for managing JMS on a network, comprising:

a JMS component;

a plurality of servers in communication with the JMS component;

a lead server in said plurality of servers, the lead server containing a distributed consensus algorithm for selecting a host server from said plurality of servers; and

a host server in said plurality of servers, said host server containing a JMS message store, said host server adapted to provide access to the JMS component to any of said plurality of servers.

67. A system according to claim 66, wherein said plurality of servers comprise a cluster.

68. A system for managing JMS on a network, comprising:

a JMS component;

a plurality of servers in communication with the JMS component;

a hardware cluster containing hardware cluster servers located in said plurality of servers, said hardware cluster containing an algorithm for selecting a lead server from among said hardware cluster servers;

a lead server in said hardware cluster servers, the lead server containing a

distributed consensus algorithm for selecting a host server from said plurality of servers; and

a host server in said plurality of servers, said host server containing a JMS message store, said host server adapted to provide access to JMS to any of said plurality of servers.

69. A system according to claim 68, wherein said host server is in said hardware cluster.

70. A framework for managing JMS on a network, comprising:

a plurality of servers, each server capable of hosting a JMS message store;

a distributed consensus algorithm for selecting a host server from among said plurality of servers, the host server to host a JMS message store; and

a distribution system for notifying servers on the network that the host computer is hosting the JMS message store.

71. A method for leasing a JMS object to a server on a network, comprising:

selecting a host server from among a plurality of network servers using a distributed consensus algorithm;

assigning a JMS object to the host server, the host server assigned to provide sole access to JMS for a specific period of time; and

notifying other network servers that the host server is hosting the JMS object.

72. A method according to claim 71, further comprising the step of:

assigning the JMS object to the host server for another period of time once the specific period of time expires.

73. A method according to claim 72, further comprising the step of:
assigning the JMS object to a new host server for another specific period of time once the specific period of time expires on the host server.

74. A method for leasing a JMS object to a server on a network, comprising:
selecting a lead server from among a plurality of hardware cluster servers in a hardware cluster;
selecting a host server from among a plurality of network servers using a distributed consensus algorithm on said lead server;
assigning a JMS object to the host server, the host server assigned to provide sole access to JMS for a specific period of time; and
notifying other network servers that the host server is hosting the JMS object.

75. A method for assigning ownership of an object on a network, comprising:
selecting a lead server from among a plurality of hardware cluster servers in a hardware cluster;
selecting a host server from among a plurality of network servers using a distributed consensus algorithm on said lead server;
assigning a JMS object to the host server, the host server assigned to provide sole access to JMS; and
notifying other network servers that the host server is hosting the JMS object.

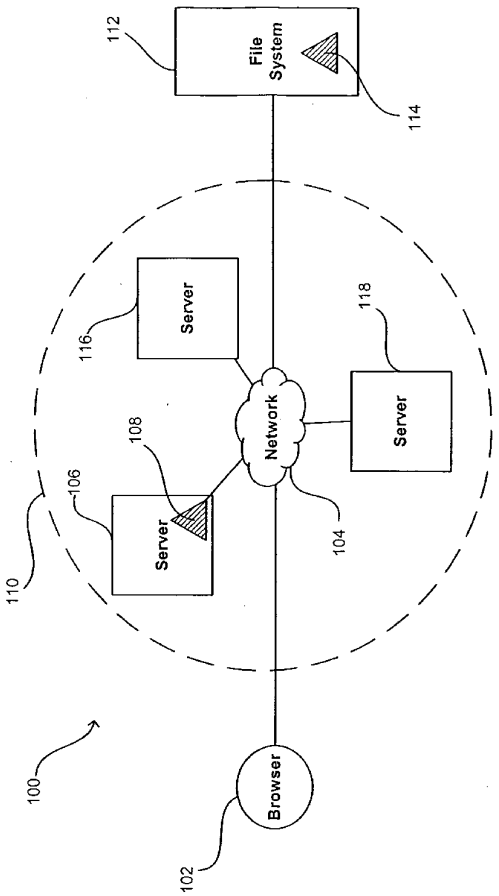

112
File System
114
Server
116
118
Server
Network
108
106
Server
104
110
100
Browser
102

Figure 1

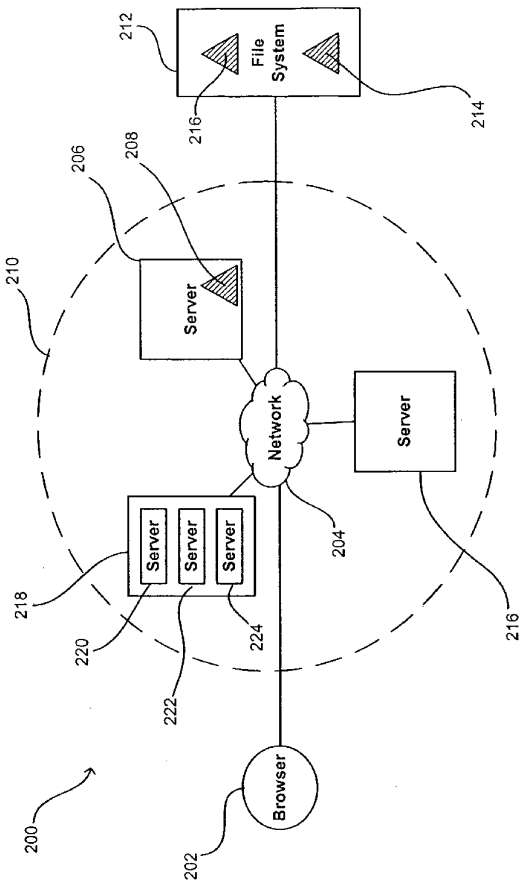

212
File System
214
216
206
208
210
Server
Network
Server
Server
Server
Server
218
220
222
224
204
216
200
Browser
202

Figure 2

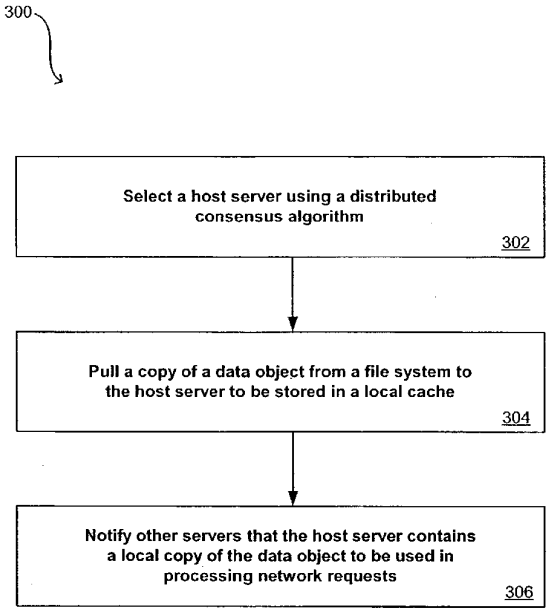

300
Select a host server using a distributed consensus algorithm
302
Pull a copy of a data object from a file system to the host server to be stored in a local cache
304
Notify other servers that the host server contains a local copy of the data object to be used in processing network requests
306

Figure 3

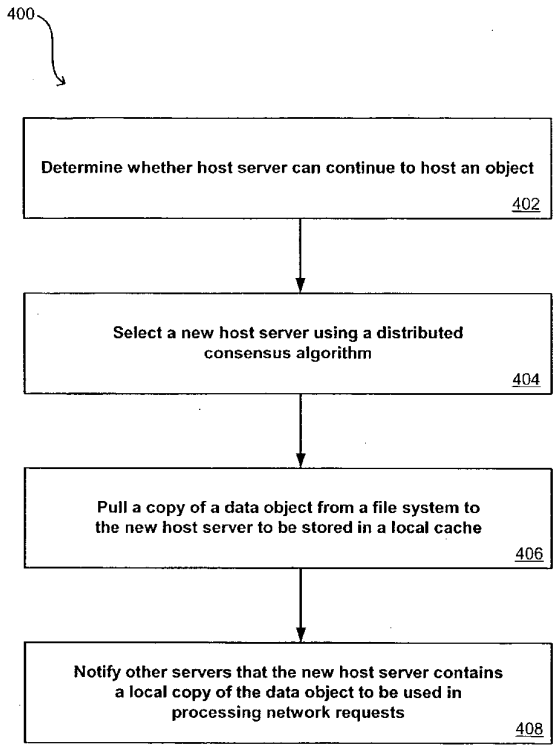

400
Determine whether host server can continue to host an object
402
Select a new host server using a distributed consensus algorithm
404
Pull a copy of a data object from a file system to the new host server to be stored in a local cache
406
Notify other servers that the new host server contains a local copy of the data object to be used in processing network requests
408

Figure 4

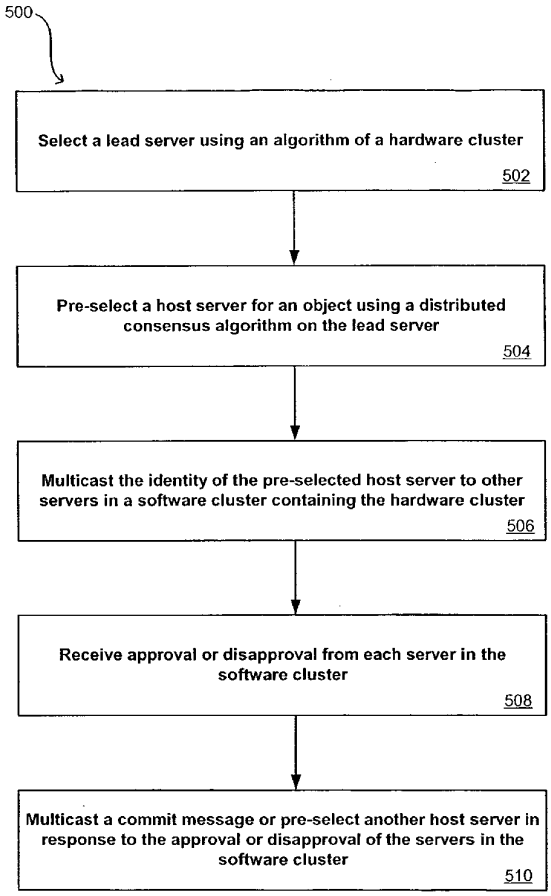

500
Select a lead server using an algorithm of a hardware cluster
502
Pre-select a host server for an object using a distributed consensus algorithm on the lead server
504
Multicast the identity of the pre-selected host server to other servers in a software cluster containing the hardware cluster
506
Receive approval or disapproval from each server in the software cluster
508
Multicast a commit message or pre-select another host server in response to the approval or disapproval of the servers in the software cluster
510

Figure 5

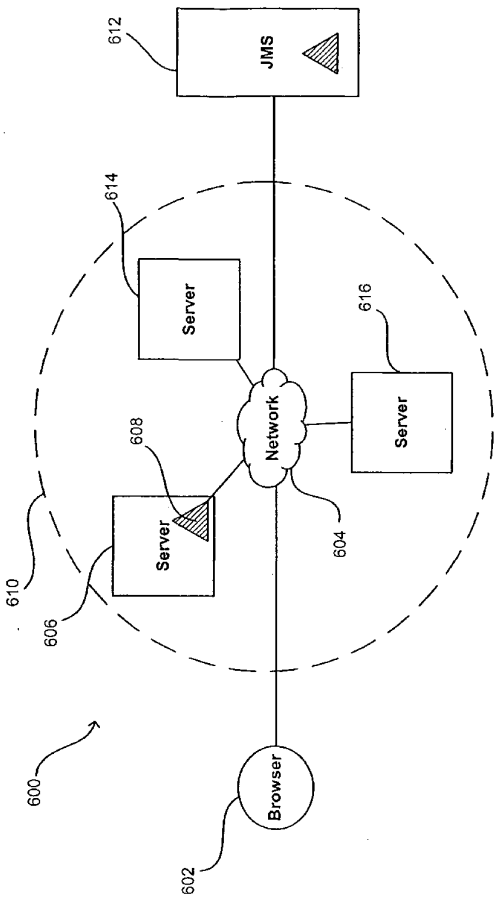

612
JMS
614
Server
616
Server
Network
608
Server
604
610
606
600
Browser
602

Figure 6

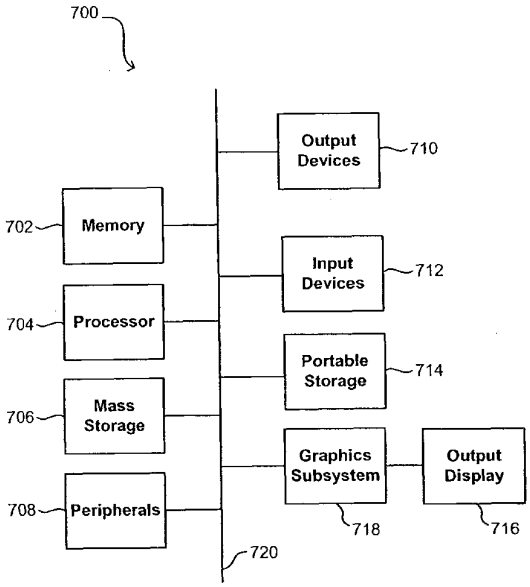

700
Output Devices
710
Memory
702
Input Devices
712
Processor
704
Portable Storage
714
Mass Storage
706
Graphics Subsystem
Output Display
Peripherals
708
718
716
720

Figure 7

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/28199

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 15/16, 15/173
US CL : 709/201, 202, 217, 223
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 709/201, 202, 217, 223

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Please See Continuation Sheet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,067,477 A (WEWALAARACHCHI et al) 23 May 2000 (23.05.2000), col. 3, line 26 - col. 5, line 38 and col. 9, line 30 - col. 10, line 41.	1-75
Y	US 5,802,291 (BALICK et al) 01 September 1998 (01.09.1998), col. 3, lines 6-35 and col. 4, line 24 - col. 5, line 65.	1-75

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
26 January 2003 (26.01.2003)

Date of mailing of the international search report
14 FEB 2003

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703)305-3230

Authorized officer
Paul H Kang Peggy Harod
Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

PCT/US02/28199

Continuation of B. FIELDS SEARCHED Item 3:
EAST
search terms: network, management, Java Message Service, host

フロントページの続き

(31)優先権主張番号 10/234,597
(32)優先日 平成14年9月4日(2002.9.4)
(33)優先権主張国 米国(US)

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,OM,PH,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TN,TR,TT,TZ,UA,UG,UZ,VN,YU,ZA,ZM,ZW

（特許庁注：以下のものは登録商標）
リナックス
マッキントッシュ
ＪＡＶＡ

(74)代理人 100074228
弁理士 今城 俊夫
(74)代理人 100086771
弁理士 西島 孝喜
(72)発明者 ジェイコーブズ ディーン バーナード
アメリカ合衆国 カリフォルニア州 ９４７０７ バークリー マデラ ストリート １７４７
(72)発明者 ハルパーン エリック
アメリカ合衆国 カリフォルニア州 ９４１１７ サン フランシスコ デルマー ストリート １６０

Ｆターム(参考) 5B034 BB02 CC01
5B082 DD04
5B089 JA35 KA12 KB04 KC23

【要約の続き】
）は、次に、マルチキャストメッセージングによって新しいホストについて通知される。