2008－OS－107 (4)
2008／1／30

耐ネットワーク分断性を持つ インターネットサービス向け分散オブジェクト

池嶋 俊†品川 高廣†杉木 章義‡加藤 和彦†‡

†筑波大学大学院 システム情報工学研究科 ‡科学技術振興機構,CREST

要 旨

インターネット上で提供されるサービスは多様化し，さらにその重要度を増してきている．しかし，そのインターネットに依存しているサービスは，その経路上の全てのホストおよびネットワーク機器が正常に動作していなければ使用できないため，ローカルで実行するアプリケーションに比べ高い可用性を保証する事は難しい．一部のインターネットサービスは信頼性の低いインターネット上でも継続してサービスが提供できるように設計されているが，新しく作るサービスを継続して提供できるようにする事は開発コストがかかったり，開発自体が困難であったりする場合もある．本研究では，継続動作するサービスを提供できるようにするための分散オブジェクトライブラリを提案する．提案するライブラリを用いる事で，インターネットの信頼性に関する問題はライブラリ層以下で隠蔽され，容易に継続して動作するサービスを開発する事ができる．

Network-partition Aware Distributed Objects for Internet Services

Syun Ikejima†Takahiro Shinagawa†Akiyoshi Sugiki‡Kazuhiko Kato†‡

†Graduate School of Systems and information Engineering, University of Tsukuba
‡CREST, Japan Science and Technology Agency

Abstract

The number and types of Internet services are consistantly growing. Due to unpredictable network outages, it is difficult to make these services highly available. Currently, few highly-available network services exist because the creation of these services require a high development cost. We propose making Internet services highly available through distributed objects. With our system, if a network outage occurs, our library will ensure the concurrency of the distributed objects. Thus, the service author does not need to be concerned about micro-managing objects.

1 はじめに

近年，インターネット上で提供されるサービスは多様化してきている．特にWebサイトを通して提供されるサービスにおいては，過去には専用システムとして閉じたホスト上で動作させていたものまでも，インターネットを介したサービスとして提供されるようになってきている．さらに，業務システムなどもインターネット上のサービスとして提供されるようになってきており，さらにそのサービスの可用性に対する要求はさらに高まってきている．

しかし，インターネットサービスにおいて高い可

用性を実現するのは難しい．インターネットは小規模なネットワークの集合体であるという性質上，通信の可用性は低い．特に，それぞれの中小ネットワークは正常に動作していても，小規模なネットワークどうしを接続するネットワークに障害が発生すれば，ネットワークの分断が発生する可能性がある．

インターネットサービスがインターネット上に設置された単一のサーバ計算機や拠点内クラスタによって提供されている場合，このサーバーが置かれているネットワークが分断してしまうと，サービスを提供し続ける事ができなくなってしまう．このような問題に対処するために，NetNews，DNS などは複数のサーバーで分散してサービスを提供しているが，このような構成は一方でデータの一貫性管理の問題が発生する．

インターネットサービスは次々新しいサービスが誕生し，短期間で開発することが求められている．これら全てに対してネットワーク分断に対応する事は難しい．また，障害に対処していたとしても，多くはコストの関係からホストの障害までにとどまる．

これらの問題を解決するために，インターネットサービスにおいてネットワーク分断に対処するためのフレームワークが望まれる．このフレームワークを利用してサービスを開発する事で，簡単に可用性の高いサービスを提供できるようになることが考えられる．

よって本研究では，インターネットサービスのためのネットワーク分断に用意に対処できる分散オブジェクトを提案する．本研究で提案する分散オブジェクトでは，オブジェクトを仮想的にホスト群に配置し，複数のホストから利用する．ネットワーク分断がホスト間で発生しても，予め分散オブジェクトごとに決められたルールに基づいて障害回復後のデータのマージ処理が行われる．それぞれの分散オブジェクトはインターネットサービスの特性を分析して作成されており，さまざまなサービスの記述が用意になっている．サービスの分散は分散オブジェクトシステムが隠蔽する．サービス開発者はサービスの状態を分散オブジェクト内のみに置くことで，意識せずサービスの分散が図られる．

本論文では，まず，典型的なインターネットサービスを分析し，分散オブジェクトライブラリに必要とされる機能や構造を示す．次に，分析に基いた分散オブジェクト群の設計案を示す．最後に，提案ライブラリを分析したアプリケーションに適用してみる事で，提案ライブラリが有用である事を示す．

2 分析

本章では，サービスを分断に耐えられるようにするにはどうしたら良いか議論し，実際にアプリケーションを分断に耐えられるようにするためにはどうすれば良いか分析を行う．

2.1 耐分断性

単一のホストで提供されているサービスにおいて，ネットワーク分断が発生した場合，そのサービスを受けられないユーザーが発生する．極端な場合，サービスを提供しているホストがネットワークから分断されてしまうと，ほぼ全てのユーザーがサービスを受ける事ができなくなる．この問題に対処する方法として，サービスを複数のホストで提供するという方法がある．

一つのホストで提供しているサービスを複数のホストで実行するためには，各ホスト上で動作するサービスの内部状態を共有する必要がある．しかし，広域なインターネット上で厳密に一貫性を取る事はコストが大きい．

また，サービスを実行するホスト間で常に一貫性を取り続けていると，ネットワーク分断時にサービスを継続させる事はできなくなり，ネットワーク分断に耐えるサービスを提供するという目的を果す事ができない．

そこで，単一の内部状態を持たずホストごとに別の内部状態を持ち，サービスの実行とは非同期に各ホストの内部状態の同期を取るというモデルが考えられる．このモデルに合わせてサービスを作る事ができれば，そのサービスは耐分断性を持つ事ができる．

では，次にこのモデルに合わせてサービスを作るためには何が必要かについて考えたい．このモデルに合わせられるサービスとは，このモデルで動作させても内部状態に矛盾が生じない必要がある．そのためには，次の 2 つの条件を満たす必要がある．

第一に，同期操作を行わずにサービスを行う必要がある．分断が発生している間，通信は行えないため，同期操作を行う事は難しい．そのため，サービスに同期操作が必要である場合は，分断が発生している間はサービスを停止せざるおえない．

第二に，別々に動作していたサービスの内部状態のマージを矛盾無く行う事ができる必要がある．これは、分断が発生したため，通信が行えなくなり，別々に動作していた 2 つのサービスの内部状態を，分断終了後に一つに戻す事が可能である必要がある

ためである．

これらの2つの条件を満たすサービスは耐分断性を持たせる事ができる．

2.2 サービスの分析

本節では，本論文が対象とするインターネットサービスについて分析を行う．分析の対象としては，次の種類のサービスを対象とする．まず，一人のユーザーがデータを送信し他のユーザーがこれを受信する物 (1 対多)．次に，多数のユーザーがデータを送信し，他の多数のユーザーがこれを受信する物 (多対多)．最後に，多数のユーザーがデータを送信し，一人のユーザーがこれを受信する物 (多対 1)．この 3 種類のサービスに対して，具体的に次のようなサービスを分析する．1 対多のサービスとして静的な Web サーバーを，多対多の簡単な例として，掲示板システムを，多対 1 の例として，メールサービスについて分析を行う．また，多対多の更に複雑な例として，オークションサービスを分析する．これらのサービスに対して，どのようなデータ構造を持ち，サービスで行える各操作が，耐分断性を持たして行えるかについて議論する．

2.2.1 静的 Web サーバー

まず，簡単な例として，静的な Web ページを提供する Web サーバーについて議論する．

Web サーバーは，内部状態として Web ページを保持し，ページを閲覧するユーザーからの要求に応じて Web ページを返すサービスである．静的な Web ページとは，ページを閲覧するユーザーからの要求によって内容が変更される事の無い Web ページの事である．しかし，静的 Web ページは必ずしも更新が行われない訳ではない．1 人，または 1 グループのサイトの所有者によって更新が行われる．

Web サーバーが持つ Web ページデータは次のような特徴がある．まず，多くの場合，更新は頻繁ではない．そのため，沢山のホスト上に分散してデータのコピーを配置しておけば，それぞれのホストからデータを読み出す事ができる．これにより，いくつかのホストとの通信が行えない状況になっても，コピーを持つホストのうち一つと通信を行う事で，サービスを継続させる事ができる．

一方，Web ページの更新について考える．一般に記録されたデータの変更を行う場合は次のような一貫性の問題がある．まず，データの書き込み中にそのデータを読むと中途半端な状態が読まれてしまうという問題 (読み込みと書き込みの衝突) がある．また，データを書き込み中にデータを書き込みを行うと，双方の書き込み後の状態が不定であるという問題 (書き込みと書き込みの衝突) もある．通常，1 ホスト内で動作するシステムの場合，これらの問題に対しては，同期操作を用いて解決を図っている．しかし，本論文では，インターネットを通じて地理的に分散した複数地点に置かれたホスト上での動作を目指している．そのため，同期操作を使う事は実用的ではない．そのため，同期操作を使わずこの 2 つの問題を解決する必要がある．

まず，前者の読み込みと書き込みの衝突に関しては，操作をまとめて扱う事で解決を図る．更新処理をアトミックにする事で，読み込み側では中途半端な状態を見ず，更新前，もしくは，更新後のいずれかのデータを取得する事ができる．

次に，後者の書き込みと書き込みの衝突については，この静的 Web サイトに関してはこの問題は発生しないと仮定できる．なぜなら，Web ページを変更する人，もしくは団体は 1 つであるため，システムの外側で一貫性は取られている．そのため，システム内で書き込みが衝突する事は無いと仮定できる．また，行われた書き込みは，書き込みが行われた時刻が後である物が最終的に修正後のものである事がわかる．そこで，もし複数の更新が発生した場合，発生時刻が最後の物を採用する．

2.2.2 掲示板システム

次に，Web 掲示板システムに関して議論する．Web 掲示板システムは，Web 上に実装された掲示板で，全てのユーザーが掲示板に記事を投稿する事ができる．また，全てのユーザーが投稿された記事の一覧を見る事ができる．

掲示板は記事の集合としてデータ構造作る事ができる．通常，掲示板は投稿された時刻が新しい順番に表示される．

掲示板には記事の投稿を行う事ができる．これは，掲示板を記事の集合とした場合，集合に記事を追加した事として扱う事ができる．また，掲示板システムでは，記事の追加は他の記事に対して排他的に行う必要や，投稿順序に厳密に順序を求める必要はないため，そのまま分散して行う事ができる．また，分断後のマージに関しては，双方の記事を重複無く集めた物を作る事で，1 ホストで掲示板システムを動作させている場合と最終的な結果が同じになる．一方，読み込みに関しても，集合内のデータを返す事で実装できる．また，掲示板は必ずしも最新の内部状態をユーザーが見れる訳ではなく，Web ブラウザによって画面更新操作を行うまでは投稿された記

事がユーザーの手元では反映されない．そのため，内部状態の一貫性を厳密に行う必要はない．以上のように記事の集合として掲示板を扱うことができるが，ユーザーは最新の記事を読む事が多いため性能上，記事を投稿時刻順に並び替えて持つ事が有利である．

2.2.3 メールサービス

インターネットを介したメールサービスについて議論する．メールサービスはサーバー内にユーザーごとのメールボックスを持ち，それぞれのメールボックスはメールアドレスによって識別される．全てのユーザーは全てのメールボックスにメールを入れる事ができる．また，そのメールボックスの持ち主のユーザーは，メールボックスからメールを取り出す事ができる．

各メールボックスはメールの集合としてデータ構造を作る事ができる．

また，上で記したようにメールボックスにはメールを入れる操作と，メールを取り出す操作ができる．メールボックスにメールを入れる操作の場合，前節の掲示板に記事を投稿する場合と同じように処理を行う事ができる．そのため，同期操作は必要ない．次にメールを取り出し，削除を行う方法について考える．メールの集合に含まれるあるメールに対して削除という操作を行う事ができるのは1度だけである．これを分散して行うと，2箇所で削除という操作を行う事ができてしまい，元の1ホストの場合とセマンティックに差が発生する．しかし，メールボックスの場合は，その中を読むのはユーザー一人であるため，2重の削除は発生せず問題にならない．また，万が一この問題が発生した場合でも，影響の範囲は，ユーザーが同じメールを2通受け取ってしまう事のみであり，データの損失などは発生しない．

分断後のマージに関しては，掲示板と同じにする事はできない．分断中にメールが削除された場合，掲示板と同じ手法でマージを行うと，削除したメールが復活してしまうという問題がある．この問題を解決するためには，削除などの操作ログを残し，マージ時にはそれを再実行する事によって，一貫性を保つ事ができる．

2.2.4 オークションサービス

複雑な例として，オークションサービスについて議論する．オークションサイトを利用者は出品者と入札者に分けられる．出品者は，サイトに品物を出品する．入札者は，その品物に入札を行う．そして期限終了後に，最高額で入札した人が落札者に決定する．

表 1: データ構造の分類

マージ方法	使用例
変更無し	Web ページの HTML，掲示板の投稿
新しい物優先	Web サイト，サイト内のユーザ情報
集合の和	掲示板，オークションの入札群
操作のログ	メールボックス
整数	アクセスカウンタ

オークションサイトのデータモデルは，出品リストの中に出品情報があり，各出品情報に対して入札のリストがあるといった形で定義する事が可能である．

オークションに対しては，出品，入札，落札という3つの操作がある．このうち，出品に関しては，掲示板と同様に，出品リストへの出品情報の追加と考える事ができる．そのため，出品に関して上記の掲示板システムと同様の方法で解決できる．入札に関しても同様である．

オークションサイトでは，オークション終了後に落札者を算出するが，この処理に関しては注意を要する．落札者を決定するには全ての入札情報が揃っている必要がある．しかし，全てのホストに入札情報を問い合わせるのは現実的ではない．そこで，最後に全ホストの情報が揃った時間を記録し，オークションの締切がそれより前である事をチェックする．これによって，落札者の算出処理を分散で行う事ができる．

2.3 分類

前節の分析によって，データ構造の種類はマージ方法によって次の5種類に分けられる事が分かった．これを，表1にまとめる．また，最後に同期が取られた時刻が得られる必要がある事がわかった．

3 設計

本章では，提案する分散オブジェクトライブラリの概要とクラス階層について説明する．

3.1 概要

本論文で提案するシステムの概観を図1に示す．

提案システムではネットワーク分断時にもオブジェクトにアクセスできるようにするために，オブジェクトへのアクセスは基本的にローカルに保持し

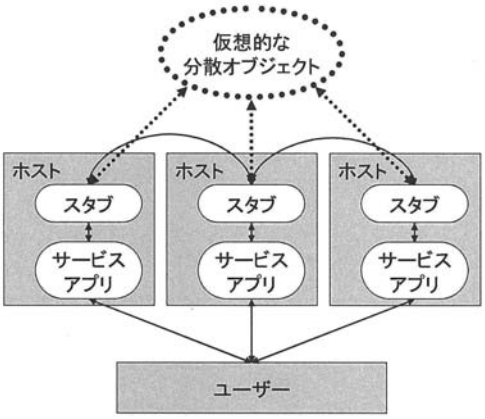

図 1: システムの概観

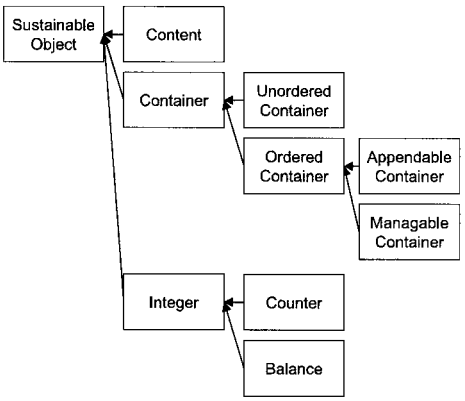

図 2: 提案ライブラリのクラス図

ているコピーに対しておこなう．一般的な分散オブジェクトシステムでは，RMI のようにネットワークを通して他のホスト上のオブジェクトにアクセスするため，ネットワークが分断するとオブジェクトにアクセスできなくなってしまうが，提案システムではローカルのコピーにアクセスできるため，ネットワーク分断時でもサービスを継続して提供することができる．

各サイトでローカルのコピーを更新した場合，オブジェクトの一貫性の維持が問題となる．すなわち複数のサイトで更新されたオブジェクトのマージ処理が必要となる．マージ処理はオブジェクトの構造に依存する処理であり，あらゆるオブジェクトに対してマージ処理を定義することは難しい．そこで本システムでは，前章で分析したインターネットサービスの性質を利用して，マージ処理が容易におこなえるようなオブジェクトクラスをいくつか定義する．インターネットサービスを構築する際に，このクラスを用いてサーバプログラムを記述することにより，マージ処理を容易に行うことができるようになり，ネットワーク分断時にも継続してサービスを提供することが容易になる．

また，本システムでは，サーバプログラムから分散オブジェクトを取り出すための分散レジストリサービスを提供する．このレジストリサービスでは，各サイトにおいて名前を指定してオブジェクトを取り出すことができる．

3.2 クラス階層

本論文で提案する分散オブジェクトのクラス階層を図 2 に示す．本クラス階層では，基底クラスとして SustainableObject クラスがある．また，これを継承するクラスは大きく分けて Content クラス，Container クラス，Integer クラスの 3 つに分けられる．以下にそれぞれのクラスについて説明する．

3.3 SustainableObject クラス

SustainableObject クラスは，本クラス階層における全てのクラスの基底クラスである．このクラスには，データを保持する機能は無く，サーバプログラムからはこのクラスの派生クラスを利用する．

SustainableObject クラスには，全てのクラスで共通に持つ基本機能として，最後にそのオブジェクトが他のサイトと同期を取った時刻を返す機能を持たせている．これは，アプリケーションによっては，最後に一貫性をとった時刻を確認してからでないと処理が継続できない場合があるためである．例えば，オークションサービスでは，落札時刻が終了した後に一度でも一貫性がとられたことを確認してからでないと落札者を決定することができない．最終同期時刻の取得はオブジェクトの種類によらず必要となるため，基底クラスの機能としている．

3.4 Content クラス

Content クラスは，インターネットサービスで使われる各種コンテンツを格納するためのクラスである．例えば，Web サービスにおける静的コンテンツや掲示板サービスにおける「記事」などを格納することを想定している．Content オブジェクトを用

いることで，サーバプログラムは必要な時にいつでもコンテンツにアクセスできる．

Content オブジェクトは，いったん作成された後は変更されることのない Immutable なオブジェクトである．オブジェクトはコンテンツのハッシュ値に基づく ID によって一意に識別されるため，ハッシュ値が同じコンテンツのオブジェクトをいくつ作成しても，それらの実体は同じものとみなされる．

Content オブジェクトは，Immutable であることを利用してネットワーク分断への対応を容易にしている．例えば，オブジェクトはコンテンツのハッシュ値で識別されるため，ネットワーク分断時にも各サイトで自由にオブジェクトを生成することができる．ネットワーク分断から復旧した際には，全てのサイトに存在する Content クラスのオブジェクトの集合をマージして，重複するものを削除すればよい．

また，あるサイトでオブジェクトが生成された場合には，速やかに他のサイトにコピーを伝搬しておくことで，ネットワーク分断時にもコンテンツを継続して利用できる可能性を高くすることができる．コンテンツはハッシュ値で識別されていてお互いに依存関係がないため，複数のコンテンツを並列にコピーしたり，任意の順番でコピーすることが可能である．なお，具体的にどのようなタイミングでどの程度コピーをおこなうかは実装依存であり，本フレームワークのレベルでは規定しない．

3.5 Container クラス

Container クラスは，Content クラスのオブジェクトへの参照を単数または複数格納するためのクラスである．Container クラスは，各オブジェクトに一意の順番が付けられた OrderedContainer と，順番の付けられていない UnorderedContainer の 2 つに大きく分けられる．

3.5.1 OrderedContainer クラス

OrderedContainer クラスは，参照を格納しているオブジェクトに対して全サイトで一意となるような順番が付けられるものである．例えば，掲示板サービスにおける記事の一覧を格納した「掲示板」や，オークションサービスにおける「入札リスト」などを格納することを想定している．

OrderedContainer クラスでは，オブジェクトに順番を付けることでネットワーク分断への対応を容易にしている．例えばネットワーク分断時に複数のサイトでオブジェクトが追加された場合でも，ネットワーク分断から復旧した際にオブジェクトの順番に基づいたマージソートをおこなえばよい．オブジェクトの順番はグローバルに一意であるため，どのサイトでマージ処理をしても順番は同じになる．

オブジェクトの順番としては，典型的にはタイムスタンプを用いることを想定している．例えば，掲示板であれば記事が投稿された時のそのサイトの時刻を用いる．全てのサイトの時刻を厳密に同期させることは困難であるが，インターネットは通信遅延が大きく，もともと厳密な時刻同期を前提にしたサービスを実現することは難しい．従って，NTP などを用いて誤差が一定以下に抑えられていれば，多くのインターネットサービスにおいては多少の誤差は許容されると考えられる．

OrderedContainer クラスは，削除を許可するかどうかによって，さらに AppendOnlyContainer と AppendDeleteContainer の 2 つに分けられる．AppendOnlyContainer クラスでは，オブジェクトの追加のみを可能とすることにより，マージを容易にしている．一方 AppendDeleteContainer では，オブジェクトの追加だけでなくオブジェクトの削除も可能とする．この場合，単純にマージをすると削除されたオブジェクトが復活してしまうことがある．従って，オブジェクトを削除するだけではなく削除操作のログを保持しておく．

3.5.2 UnorderedContainer クラス

UnorderedContainer クラスも，複数のオブジェクトへの参照を格納するが，それらのオブジェクトに一意の順番がつけられないものである．例えば，Web サービスにおけるディレクトリ構造のように，順番だけでは各オブジェクトの関係を規定できないものを想定している．

UnorderedContainer クラスのオブジェクトは，ネットワーク分断時に複数サイトで同時に更新すると，マージ処理を行うのが難しい．これはオブジェクトの構造がアプリケーション依存であるため，汎用的なマージ手順を定めることが難しいためである．

そこで UnorderedContainer クラスのオブジェクトでは，更新を許可するユーザを 1 人に限定する．例えば，Web サービスの例では，Web のディレクトリ構造の変更は 1 人の管理者しかおこなえないこととする．ネットワーク分断時に 1 人の管理者が複数のパーティションで更新処理をおこなうことは考えにくいため，一貫性の問題は基本的には生じない．ネットワーク分断時におこなわれたオブジェクトの更新は，ネットワークの回復時に速やかに他のサイトに伝搬する．

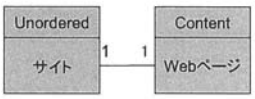

図 3: 静的 Web サービスのクラス図

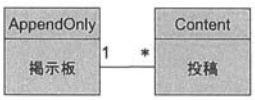

図 4: 掲示板サービスのクラス図

3.6 Integer クラス

Integer クラスは，数値を保持するためのクラスである．保持する内容を数値に限定することによって，加算や減算といった演算を定義して，ネットワーク分断後のマージ処理を容易にすることができる．Integer クラスは，Counter クラスと Balance クラスの 2 つに分けられる．

Counter クラスは，値の加算のみができる数値のクラスである．例えば，アクセスカウンタなどを想定している．演算操作を加算のみに限定することにより，マージ処理を容易にすることができる．例えば，ネットワーク分断時に 1 つの Counter オブジェクトに対して複数サイトで加算がおこなわれた場合，回復時には全てのサイトで加算された値を合算したものを最終的な値とすればよい．

Balance クラスでは，値の加算及び減算をおこなうことができる．ただし値はマイナスの数値にはならない．例えば，商品の在庫数やポイントの残高などを想定している．Balance クラスでは，ネットワーク分断時にも値の減算を可能にしつつ値がマイナスになることを防ぐために，オブジェクトの値とは別にサイトごとに減算可能な数の上限値を設定できるようにする．例えばオンラインショッピングでは，ネットワーク分断時でも一定の個数までは販売することが可能になる．ネットワークが接続されている場合には，サイト全体で値の一貫性をとりつつサイト別の減算可能な上限値を適切に設定することにより，ネットワーク分断に対応する．

4 アプリケーション例

本章では，3 章で述べた分散オブジェクトライブラリを用いて，2 章のインターネットサービスをネットワーク分断に対応させる例を示す．

4.1 静的 Web サービス

静的 Web サービスを提案システムで構築した場合のクラス図を図 3 に示す．静的 Web サービスで扱うデータとしては，「静的コンテンツ」と，URL から静的コンテンツへの「対応表」の 2 つがある．

静的コンテンツは，内容が変更されないため，Content クラスを用いる．これにより，ネットワーク分断時にも静的コンテンツへのアクセスや新規作成が可能になる．静的コンテンツは複数の HTML ファイルや画像ファイルなどをまとめたアーカイブとすることもできる．

対応表は，URL を静的コンテンツのオブジェクトへの参照に変換するデータ構造である．コンテンツを更新する際には，対応表の参照先を新しいコンテンツのオブジェクトへの参照に更新することでおこなう．この更新作業はマージが難しいが，更新作業を許可するユーザを管理者 1 人に限定することは妥当である考えられる．従って対応表としては，UnorderedContainer クラスを用いる．URL から UnorderedContainer オブジェクトを取り出すためには，分散レジストリサービスを用いる．

静的コンテンツと対応表をどの程度細かい単位とすべきかは運用による．例えばサイト全体で 1 つの静的コンテンツオブジェクトとした場合，サイト全体を一貫性を保った状態で一度に更新することが可能になるが，サイトの管理者は全体で一人となり，コンテンツの一部を変更する場合でも静的コンテンツオブジェクトを丸ごと変更することになる．従って運用に応じて適切な粒度を選択する必要がある．

4.2 掲示板サービス

掲示板サービスを提案システムで構築した場合のクラス図を図 4 に示す．掲示板サービスで扱うデータとしては，ユーザが投稿する「記事」と，投稿された記事を時間順に並べた「掲示板」の 2 つがある．

記事はいったん投稿されたら内容が変更されることはないため，Content クラスのオブジェクトとして格納する．掲示板は記事を時間順に複数並べたものであるため，AppendOnlyContainer クラスを用いる．記事のオブジェクトが生成されたら，そのサイトにおけるオブジェクト生成時のタイムスタンプを付けて掲示板オブジェクトに追加する．

4.3 メールサービス

メールサービスを提案システムで構築した場合のクラス図を図 5 に示す．メールサービスで扱う

図 5: メールサービスのクラス図

図 6: オークションサービスのクラス図

データとしては,「メールの本文」と各ユーザごとの「メールボックス」, サーバ内の「メールボックス一覧」の 3 つがある.

メール本文の内容は受信後に変更されることはないため, Content クラスを用いる. 各ユーザのメールボックスは, サーバによって受信したメール本文が追加されるほか, ユーザが不要なメールを削除する操作があるため, AppendDeleteContainer を用いる. メールボックスの一覧は AppendOnlyContainer を用いる. ユーザが削除されることは現時点では想定してない.

4.4 オークションサービス

オークションサービスを提案システムで構築した場合のクラス図を図 6 に示す. オークションサービスで扱うデータとしては,「出品物リスト」,「出品物」,「入札」がある.

出品物リストは, 出品物の一覧を保持するものであるため, AppendOnlyContainer を用いる. 出品物に対しては, 複数のユーザからその出品物に対する入札を保持するため, AppendOnlyContainer を用いる. 入札はいったんおこなわれたら変更されることはないため Content を用いる.

オークションの落札処理を行う場合は, SustainableObject クラスの最終同期時刻取得機能を利用する. 最終同期時刻がオークション終了時刻より前である場合は, 落札処理を遅延させる.

5 関連研究

インターネットサービスにおいてネットワーク分断に対応しているものには Gao らの研究 [1] がある. Gao らの研究では, 本研究と同様に分散オブジェクトを用いることでネットワーク分断時にもサービスを継続できる. しかし Catalog object や Order object など書籍販売サイトに特化した設計をおこなっており, 汎用的なサービスで利用するのは難しい.

Gribble ら [2] では, インターネットサービスの記述を容易にするために, ハッシュ構造を用いた汎用な分散データオブジェクトを設計・実装している. しかしこの研究では, 2 相コミットをストレージの書き込みに用いており, ホスト障害までしか対応することはできない.

Bayou[3] では, マージ手順をアプリケーションごとに SQL に近い文法で記述可能としており, ネットワーク分断に対処している. この手法は汎用性は高いが, サービスごとの開発にかかる手間が大きい.

6 おわりに

本論文では, ネットワーク分断に耐えられる性質を持つインターネット用分散オブジェクトについて述べ, 分散オブジェクトライブラリを提案した. 提案分散オブジェクトライブラリを利用する事によって, 高可用性なインターネットサービスを提供する事ができる. また, 提案オブジェクトライブラリに必要なクラスについて論じるために, 一般的なインターネットサービスについて分析した. 高い可用性を持つインターネットサービスを作るために, 提案分散オブジェクトライブラリが有用である事を示した.

参考文献

[1] Lei Gao, Mike Dahlin, Amol Nayate, Jiandan Zheng, and Arun Iyengar. Application specific data replication for edge services. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pp. 449–460, New York, NY, USA, 2003. ACM.

[2] Steven D. Gribble, Eric A. Brewer, Joseph M. Hellerstein, and David Culler. Scalable, distributed data structures for internet service construction. In *OSDI'00: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*, pp. 22–22, Berkeley, CA, USA, 2000. USENIX Association.

[3] Karin Petersen, Mike Spreitzer, Douglas Terry, and Marvin Theimer. Bayou: replicated database services for world-wide applications. In *EW 7: Proceedings of the 7th workshop on ACM SIGOPS European workshop*, pp. 275–280, New York, NY, USA, 1996. ACM.