ASP 基盤のためのアプリケーションサービスの自動開閉処理

Automated Opening and Closing of Application Services for ASP Infrastructures

五十嵐　智

要　約　Kiban@asaban では，アプリケーションサーバが提供するサービスを，ゲートウェイサーバを経由してインターネットへ提供する．ゲートウェイサーバではアプリケーションサーバの障害発生と障害復帰を検知し，自動的にサービス画面の開閉処理を行っている．これを汎用的に実現するために必要となる要素について検討し，Kiban@asaban での実装方法を述べる．

Abstract　Kiban@asaban discloses the services provided by an application server to the Internet via an application geteway server. The gateway server detects the occurrence of an application failure and the completion of the error recovery, and performs opening-closing processes of service automatically.

This paper discusses factors required for implementing the opening-closing prosesses, and introduces its implementation.

1. はじめに

Kiban@asaban[1][2]では，アプリケーションサーバ（AP サーバ）が提供するサービスを，ゲートウェイサーバ（GW サーバ）を経由してインターネットへ公開する．公開されたサービスは 24 時間 365 日，常にアクセスされる可能性があり，アプリケーションサービスが停止している場合には，サービス停止中であることを示す閉塞画面を表示して利用者に知らせる仕組みが必要である．また，アプリケーションサービスが再開した場合には，閉塞画面の表示を止め，速やかにサービス提供を再開することが求められる．すなわち，GW サーバは AP サーバのサービスを監視することにより，障害発生時には閉塞画面を自動的に表示し，障害復旧時には自動的にサービス提供をする．

Kiban@asaban では，ASP というサービス形態から導き出される主だった項目について，あらかじめ要求を定義し，GW サーバの構成を標準化することで，設計と構築に掛かるコストを少なくし，短期間での立ち上げを可能にしている[3]．GW サーバによる自動開閉処理を行なう際に問題となるのは，ASP 事業者の提供するサービス形態によりネットワーク構成が異なるため，GW サーバによる AP サーバの監視が複雑化することである．そこで，標準化された Kiban@asaban の GW サーバ構成を前提とすることで複雑化を避け，その上で自動開閉処理の阻害要因とその解決策を論じる．こうして標準化された自動開閉処理は，Kiban@asaban のみならず，同様のネットワークアーキテクチャを持つネットワーク基盤に適用可能である．

本稿では，まず前提条件（2 章）を記述し，その後，自動開閉処理の仕組み（3 章）と実装における問題点（4 章）を提示する．次に問題点の解決方法と実現方法（5 章）

について記述し，最後に Kiban@asaban での実装方法（6 章）について述べる．

2. 前 提 条 件

ここでは，本稿で述べる自動開閉処理を適用するための前提条件を記述する．

2.1 基 本 構 成

Kiban@asaban のネットワークアーキテクチャは，論理的に非武装ネットワーク（DMZ）とファイアウォールから成るゲートウェイ（GW）層，アプリケーションサービスおよび ASP 共通サービス等を提供するアプリケーション（AP）層，データベース機能を提供するデータベース（DB）層，バックアップ機能を備えたストレージ（SAN/NAS）層に分かれている（図 1）．

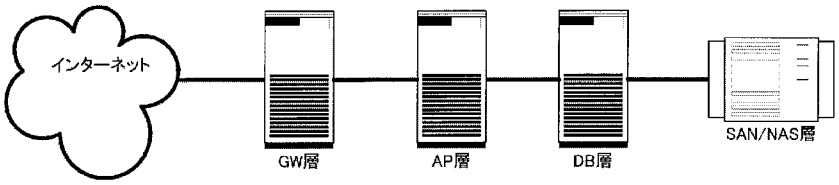

図 1　Kiban@asaban の論理ネットワークアーキテクチャ

Kiban@asaban では，GW 層は文字通り，アプリケーションゲートウェイとして機能する．AP 層で提供するサービスを，GW 層を介してユーザに提供する．提供するサービスは，一般に HTTP や HTTPS を用いた Web によるサービスである．

このように，GW 層と AP 層が分離され，GW 層が AP 層のフロントエンドとしてサービスを提供するネットワークアーキテクチャを持つシステムを基本構成とする．

2.2 サービスの状態

サービスの状態を示す用語を定義する．サービスに関する状態には，提供状態，閉塞状態，停止状態の三種類がある．

2.2.1 提 供 状 態

サービスの「提供状態」とは，サーバまたはシステム全体がサービスを公開し，期待通りにサービスを提供できることを指す．AP サーバが提供状態で，GW サーバも提供状態のとき，システム全体としてサービスを公開できる提供状態になる．

2.2.2 閉 塞 状 態

サービスの「閉塞状態」とは，AP サーバが提供するサービスではなく，GW サーバ自身が用意した閉塞用のサービスを提供することを指す．例えば，AP サーバで障害が発生し，GW サーバが「メンテナンス中」等の画面を提供している状態がこれに相当する．これにより AP サーバの状態を外部からのアクセスに対して隠蔽し，障害やメインテナンスに対応することができる．

2.2.3 停 止 状 態

サービスの「停止状態」とは，GW サーバ自体が機能していない場合や，GW サーバは動作してもサービスを提供するプログラムが機能していない場合であり，サービスを提供できない状態にあることを指す．

例えば，GW サーバ上のサービスを提供するプログラムで障害が発生し，閉塞画面さえも提供できない状態がこれに相当する．GW サーバ自体に障害が発生している場合もサービスの停止状態である．AP サーバが正常であっても，GW サーバが機能停止していると，サービスは提供できない．

GW サーバ自身に障害が発生している状態と，サービスを提供するためのプログラムが停止している状態の違いは，GW サーバ自身が能動的に処理を行えるか否かに現れる．単にプログラムを停止してサービスを停止状態にしている場合は，状況の変化に応じて，サービスを提供状態や閉塞状態に遷移させることが可能である．

2.3 クラスタ構成

ネットワークに対するクラスタ構成は，サービス提供においてネットワークの高可用性を実現する．サービス IP アドレスと呼ばれる一つの IP アドレスに対し，複数のサーバでサービスを提供することで，単一サーバに障害が起きた場合でも他のサーバによって，サービスを提供しつづけることが可能となる．このクラスタ構成の前提条件を示す．

2.3.1 ゲートウェイ層におけるクラスタの構成

GW 層で構成するクラスタと開閉処理とは分離して実現するが，GW 層では GW サーバの障害を検知する，何らかのクラスタが構成されていることを前提とする．

GW 層で構成するクラスタには負荷分散型やアクティブスタンバイ型などが考えられるが，同一のサービスを提供する GW サーバ群の開閉処理の仕組みには一切影響しない．ただし，負荷分散装置（ソフトウェア）やアクティブスタンバイのためのクラスタリングソフトウェアが GW サーバのサービス停止を検知した場合，負荷分散装置やクラスタリングソフトウェアがその GW サーバをクラスタからはずすことを前提としている．DNS のラウンドロビンによるアクセスのように，GW サーバの状態には全く関知せずに負荷分散を行うような場合は，本稿の自動開閉処理の範囲外である．

したがって，複数の GW サーバで構成される GW 層では，GW サーバを監視して負荷分散を行う，あるいはアクティブとスタンバイを切り替えるような，GW サーバのサービス障害を検知するクラスタが構成されていることが前提である．

2.3.2 アプリケーションサーバの障害に対する考え方

高可用性の考え方では，基本的には障害が同時に二箇所以上で発生した場合には高可用性が成立しないと考えるのが一般的である．しかし，自動開閉処理においては AP サーバに障害が発生している場合でも，GW 層では独立して高可用性を維持するものとして考える．つまり，GW 層にとっては，AP サーバの障害は，一つの状態遷移の形態であり，GW 層に障害が発生した時に初めて一点の障害が発生したとみなす．

2.4 ゲートウェイサーバとアプリケーションサーバのネットワーク構成の考え方

基本構成の前提条件について 2.1 節で述べたが，ここでは特に GW サーバと AP サーバとのネットワーク構成に着目して考え方を記述する．

AP 層でクラスタを構成せずに複数の AP サーバで同一のサービスを提供する場合，GW サーバと対になってサービスを提供することを前提とする．つまり，GW サーバと AP サーバが論理的に一対一で結びついていると考える．この場合，各 AP サ

ーバ間は物理的に接続されている必要はない．すなわち，GW サーバと AP サーバの関係のみが重要であり， GW サーバと AP サーバの数がそれぞれ N 台ずつであれば，1×1 構成（図 2）が複数あるとみなすことができる（図 3）．

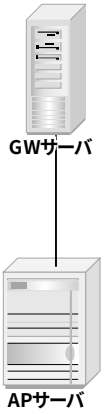

図 2　1×1 構成

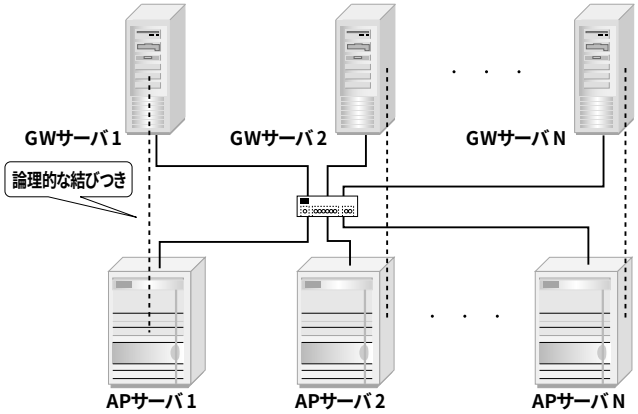

図 3　N×N 構成

AP 層でクラスタを構成する場合は，AP 層のサービス IP アドレスによって提供されるサービスのみが開閉処理を行う上で必要であるため，GW 層から見ると，単一の AP サーバとして扱える．GW サーバが N 台の場合，GW サーバと AP サーバの数は N 対 1 の関係になるが，完全に同一状態の AP サーバが仮想的に N 台あると考えることができる．そのため，自動開閉処理を考える上では N×1 構成は N×N 構成の場合と同じネットワーク構成であるとみなしてよい．

また，GW サーバと AP サーバが N×M（N≠M）構成の場合についても，N×1 構成と N×N 構成の組み合わせと考えることで，応用可能である．

したがって，GW サーバと AP サーバのネットワーク構成を考える場合，N×N 構成（N≧1）について検討すれば十分である．

3. 自動開閉処理の仕組み

AP サーバが提供するサービスを GW サーバを経由して公開する場合，アプリケーションに障害が発生した場合は，AP サーバの設定によりエラー画面を表示することが可能である．しかし，AP サーバ自身や GW サーバから AP サーバに至るネットワーク上に障害が発生すると，AP サーバの設定は効力をなさなくなる．この場合，GW サーバ側で障害に対応する仕組みが必要となる．

AP サーバやネットワークに障害が発生したことを検知するためには，AP サーバが提供するサービスを監視することが不可欠である．他の GW サーバでサービスを提供することが可能な場合，AP サーバの障害を検知した GW サーバは停止状態に遷移しなければならない．これは，クラスタリングソフトウェア（あるいは装置）に，その GW サーバではもはやサービスを提供できないことを検知させるためである．提供状態にあった最後の一台の GW サーバが AP サーバの障害を検知した場合は，

予め設定した閉塞画面を表示して閉塞状態となる．このとき，停止状態にあるGWサーバは閉塞状態へ遷移して，APサーバの障害の復旧を待つ．停止状態から閉塞状態へ遷移するのは，GW層での高可用性を維持するためである．

提供状態にある最後の一台のGWサーバに障害が発生した場合も同様に，他の停止状態のGWサーバは閉塞状態に遷移する．

GWサーバがAPサーバの障害の復旧を検知した場合，自動的にサービスの提供を再開し提供状態となる．閉塞状態のGWサーバが存在した場合は，そのGWサーバはAPサーバの状態に応じて閉塞状態から提供状態または停止状態へ遷移する．

このように，GWサーバ間での状態の同期をとるために，各GWサーバは状態が遷移したときに他のGWサーバへ遷移した状態を通知する．また，提供状態あるいは閉塞状態の最後のGWサーバに障害が発生していないことを確認するために，相互に監視することが必要である．

4. 実装における問題点

この章では実装する上で特に問題となる点を述べる．

4.1 アプリケーションサーバの状態監視の問題点

複数のGWサーバが単一のAPサーバを監視する場合，監視タイミングによる不整合の問題が発生する．

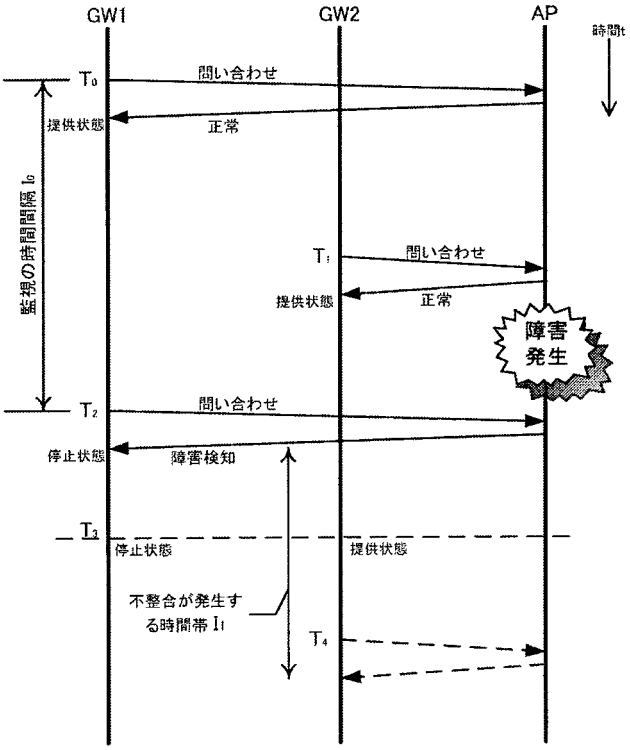

図 4　監視のタイミングと不整合

GW サーバはそれぞれが時間間隔を設定して AP サーバが提供するサービスを監視するため，それぞれの GW サーバで AP サーバのサービスを検査するタイミングが異なる．図 4 は単一の AP サーバに二台の GW サーバが状態検査を行うタイミングを示している．T_3 において，GW_1 は AP サーバに障害が発生していることを検知しているが，GW_2 はまだ検知できていない．検知した状態が異なると，GW 層全体で見たときに，正しくサービスを提供できない場合が発生する．つまり，図の I 1 で示す時間帯では，GW_1 は AP サーバの障害を検知して停止状態になっているが，GW_2 ではまだ AP サーバの障害を検知できていないために提供状態のままである．この間に GW_2 へアクセスすると，エラーが発生する．

4.2 ゲートウェイサーバ間の相互監視の問題点

GW サーバが監視している AP サーバについては，単純にサービスが提供されているかどうかを検査すればよい．しかし，GW サーバについては，クラスタ構成を用いている場合，単純には検査できない場合がある．これは，クラスタによってサービス IP アドレスが割り当てられ，そのサービス IP アドレスに対してアクセスした場合にのみ，GW サーバ上のサービスが正常に提供されるような構成が存在するためである[*1]．

このような場合には，検査対象となる GW サーバの実アドレスを用いてサービスを確認しようとしても，得られる結果は実際のサービスの提供状態とは異なる[*2]．また，クラスタのサービス IP アドレスに対して監視しても，必ずしも対象となる GW サーバの状態を検査できるとは限らない．従って，特定の GW サーバが提供するサービスを，外部から検査する方法は無いと考えなければならない[*3]．

5. 実 現 方 法

ここでは，4 章で述べた問題点の解決方法とサービス状態の遷移について述べる．

5.1 サービス監視の不整合の解決

各 GW サーバの AP サーバに対する監視のタイミングのずれを全くなくすことは，事実上不可能であるが，最小限に抑える必要がある．時刻同期を利用する方法と通知による方法が考えられる．

時刻同期を利用する方法は，GW サーバ間で時刻同期を行い，AP サーバへの監視を同一時刻に行うことにより，タイミングのずれをなくし，GW サーバ同士の検知の差異を微小にすることが可能になる．

通知による方法は，いずれかの GW サーバが AP サーバの障害を初めて検知したときに，他の GW サーバへその旨を通知する．通知を受けた GW サーバでは，通常の監視スケジュールとは別に，即座に AP サーバのサービスを検査し，最新の AP サーバの状態を把握する．

通知による方法は時刻同期を利用する方法に比べて，不整合の時間がわずかに長くなる．しかし，時刻同期の方法では AP サーバの負荷が集中することや，監視のタイミングのずれをなくすことで，逆に監視をしていない時間帯が長くなることが考えられる．そのため，本稿では通知による方法を採用する．

5.2 ゲートウェイサーバ間の相互監視の方法

GW サーバの相互の監視は，問い合わせと応答によって行う．問い合わせを行う GW サーバは，定められた時間間隔で，他のすべての GW サーバに対して状態を問い合わせる．状態の問い合わせを受けた GW サーバは，自身のサービスの状態を応答として返す．応答を受けた GW サーバは，当該の GW サーバの状態を記録する．一定時間内に応答が返らなかった GW サーバについては，停止状態にあると仮定すると共に，障害が発生していることを記録する．

5.3 サービス状態の遷移

GW サーバのサービス状態は，AP サーバのサービス状態と他の GW サーバのサービス状態によって決まる．以下に，遷移する状態が決まる条件を示す．

1）AP サーバが正常にサービスを提供していて，他の GW サーバのいずれかが提供状態の場合は，提供状態を維持する．

2）AP サーバが正常にサービスを提供していて，他の GW サーバのいずれも提供状態ではなく，かつ，他の GW サーバのいずれかが閉塞状態の場合は，監視 GW サーバが自分自身を提供状態にする．

3）AP サーバが正常にサービスを提供していて，他の GW サーバがすべて停止状態の場合は，監視 GW サーバが自分自身を提供状態にする．

4）AP サーバに障害が発生し，他の GW サーバのいずれかが提供状態の場合は，監視 GW サーバが自分自身を停止状態にする．

5）AP サーバに障害が発生し，他の GW サーバのいずれも提供状態ではなく，かつ，他の GW サーバのいずれかが閉塞状態の場合は，監視 GW サーバが自分自身を閉塞状態にする．

6）AP サーバに障害が発生し，他の GW サーバがすべて停止状態の場合，監視 GW サーバが自分自身のサービスを閉塞する．

これらの動作をまとめたものを表 1 に示す．

表 1　AP サーバおよび他 GW サーバの状態と遷移すべき状態

AP の状態	他 GW サーバの状態		
	1 以上の提供状態の GW サーバがある	提供状態の GW サーバが 0，1 以上の閉塞状態の GW サーバがある	提供状態の GW サーバが 0 閉塞状態の GW サーバが 0
正常	1）提供状態	2）提供状態	3）提供状態
障害	4）停止状態	5）閉塞状態	6）閉塞状態

GW サーバが，あるサービス状態から他のサービス状態へ遷移するとき，他の GW サーバに対して通知する．例えば，停止状態にある一台の GW サーバが他の GW サーバを監視すると，提供状態の GW サーバが 0，いずれかの GW サーバが閉塞状態のときに，AP サーバが正常にサービスを提供しているという状況が稀に発生する．このとき監視を行った GW サーバ自身は提供状態になるが，他の GW サーバが閉塞状態のままでは，不整合が起きてしまう．このため，他の GW サーバに対して提供状態になった旨を通知し，その通知を受け取った GW サーバは，提供状態か停止状

態へ遷移することになる．

6. Kiban@asaban での実装方法

この章では，Kiban@asaban での主な実装方法について紹介する．

6.1 状態遷移の方法

状態遷移の方法について，インタフェースと状態遷移を行うエージェントについて述べる．

6.1.1 状態遷移のインタフェース

Kiban@asaban では，状態を遷移させるためのインタフェースとして，SNMP を用いる．SNMP のエージェントを置き，状態遷移の指示に対して状態遷移エージェントを起動する．

一つのサービスに対し，以下の 3 個のインタフェースを用意する．

・open：サービスの提供開始

・close：サービスの閉塞

・shutdown：サービスの停止

上記に加え，状態保持と状態保持の解除を指示するために二つの修飾子を用意する．

・Keep：サービスの状態保持

・No keep：サービスの状態保持の解除

これらのインタフェースの組は，サービス単位に用意する．

6.1.2 状態遷移エージェント

サービスの状態を変更するために，状態遷移エージェントを置く．状態遷移エージェントは SNMP エージェントから起動される．変更した状態を記録し，履歴を保存する役割も持つ．状態遷移エージェントはサービスの状態を変更することに専任し，他の GW サーバへの指示等は行わない．

6.2 状態検知の方法

監視エージェントが SNMP により状態の問い合わせを行い，各サーバの状態を検査する．問い合わせた結果が状態保持のいずれかであった場合，状態遷移の決定においては，その GW サーバの状態は応答がない状態であるとみなす．これは，状態保持が外部からの指示によるものであること，それによる不整合状態は外部から指示を出した者（人，プログラム）の責任によって解消されるべき状態であることと，考えるためである．

6.3 状態の通知

状態の問い合わせが行われた場合，SNMP エージェントが応対し，監視エージェントに問い合わせを行う．監視エージェントは状態遷移エージェントによって記録された状態を読み，SNMP エージェントへ返答する．SNMP エージェントは監視エージェントからの返答を問い合わせ元に返答する．

また，監視している AP サーバの障害を初めて検知した時，監視エージェントが他の GW サーバへ通知を行う．

6.4 状態遷移の決定プロセス

状態遷移の決定を行うのは，監視エージェントの役割である．監視エージェントは，

あらかじめ定められたスケジュールに従い，また，他の GW サーバからの指示に従って AP サーバの監視と他の GW サーバの検査を行う．監視を行い，遷移すべき状態を決定した後，監視エージェントは，SNMP エージェントに対して状態の遷移を指示する．

6.4.1 監視のスケジューリング

提供状態では，あらかじめ定めた時間間隔で監視を行う．閉塞および停止状態の場合，提供状態よりも短い間隔で監視を行う．これは障害復旧に対して迅速に対応するためである．

状態保持の場合は，状態保持が解除されるまで監視のスケジューリングを停止する．

6.4.2 他のゲートウェイサーバの指示による状態の検査

GW サーバは AP サーバの障害を検知した場合，各 GW サーバ間での不整合を防ぐため，通知を行う．この通知を SNMP エージェント経由で受け取った GW サーバの監視エージェントは，即時に AP サーバの状態検査を行う．

また，状態が遷移した GW サーバは，他の GW サーバへ状態の確認を SNMP トラップによって指示する．SNMP エージェント経由で指示を受けた監視エージェントは，即時に状態遷移の決定プロセスを実行する．

6.4.3 状態遷移の履歴の検査

遷移する状態を決定する際，ログファイルに記録された状態遷移の履歴を検査する．これは，一定時間に状態が著しく変化するような状況を検知するためである．AP サーバ側のネットワークが不安定な場合など，自動的な障害検知では対応できない障害が発生している可能性がある．例えば，5 分間に 10 回も提供状態と閉塞状態を繰り返すような場合には，明示的にサービスの閉塞状態を保持させるためである．

6.5 実装方法のまとめ

Kiban@asban の実装に使用したエージェントについて表 2 にまとめた．また，エージェント間の関連については，図 5 に示した．

表 2 実現方法のまとめ

エージェント	役　　割
SNMP エージェント	• 状態遷移に対する指示を受け取り，状態遷移エージェントへ伝達する． • 状態の問い合わせに対し，監視エージェントを起動し，現在のサービス状態を通知する．
状態遷移エージェント	• SNMP エージェントから起動され，サービスの状態を変更する． • 遷移した状態を履歴として記録する．
監視エージェント	• AP サーバおよび他の GW サーバを監視する．検知したステータスに従って，遷移すべき状態を決定する． • SNMP エージェントへの指令を出す．

7. 課　　題

本稿で述べた GW サーバ間の相互監視では，いくつかの条件下で正しく相互の状態を把握できない場合があることがわかっている．このような状態への対応を引き続き検討しなければならない．

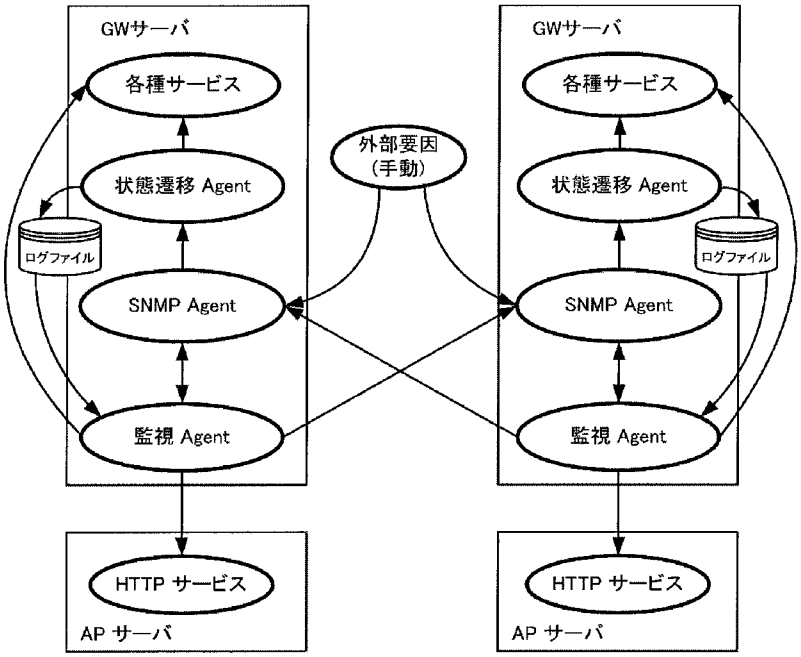

図 5　エージェントの関連

AP サーバが提供するサービスと DB 層および SAN/NAS 層との連携に対する開閉処理は，それぞれのサービス提供の形態に応じて，個別に対応しているのが現状である．よりコストを抑え，迅速な構築サービスの提供と，スケールメリットを生かした保守を行うために，AP–DB 連携部分の開閉処理を標準化する必要がある．

また，Kiban@asaban で実装する上で，SNMP を主体に構築しているが，より堅固にするために TCP/IP 上の独自のプロトコルを用いたエージェントを構築する方法も検討する必要がある．

8. おわりに

Kiban@asaban と同様の構成におけるアプリケーションサービスの自動開閉処理について記述した．AP サーバ自身が障害を検知した場合については，本稿の方法で応用が可能なため，特に記述しなかった．

本稿では標準化を行うための方法を紹介したが，このような単純ではあるが一般に適用できる方策の組み合わせにより，省コストと短期間での構築が実現可能となる．ASP 事業は今後も拡大すると思われるが，スケールメリットを生かしたコスト削減と工期短縮がますます望まれるため，今後も引き続き検討を行う．

＊ 1　例えば，Apache[7] の IP アドレスベースの VirtualHost を使用している場合．
＊ 2　仮想アドレスと実アドレスで同一のサービスを提供している場合にのみ同一結果が得られる．

＊ 3　クラスタリングのソフトウェアを改造することで可能であるが，クラスタリングのソフトウェアの構造に依存し，またGWサーバ外部の負荷分散装置などを使用した場合には，必ずしも改造できるとは限らないため，この方法は使用しない．

参考文献

［1］　川辺治之,“ ASP 基盤 Kiban@asaban ”, UNISYS 技報, 第 72 号, 2002 年 3 月
［2］　保科剛,“ ASP サービス基盤「Kiban@asaban」”, UNISYS 技報, 第 68 号, pp 135–143, 2001 年 3 月
［3］　布村知靖,“ ASP に求められるゲートウェイ構築 ”, UNISYS 技報, 第 72 号, 2002 年 2 月
［4］　Linux–HA Project, http : //linux–ha.org/
［5］　The Linux Virtual Server Project—Linux Server Cluster, http : //www.linuxvirtualserver.org/
［6］　Linux Network Address Translation, http : //www.linas.org/linux/load.html
［7］　Apache, http : //www.apache.org/

執筆者紹介

五十嵐　智（Satoshi Ikarashi）
1990 年新潟大学工学部電気工学科卒業．同年日本ユニシス(株)に入社．UNIX オペレーティングシステムの受入/保守に従事．現在 asaban.com 事業部技術企画室に所属し，Kiban@asaban の設計/開発/保守に従事．