BANCS 接続システム（2）
――補完機能開発で実現した HA システム構築

Development of HA system on ES 7000/W2KDCS
――Challenge to Mission Critical Banking System

溝 上 昌 宏

要 約 M 銀行と日本ユニシス株式会社が再構築した BANCS システムはハードウェアにユニシス社の ES 7000，オペレーティングシステムにマイクロソフト社の Windows 2000 Datacenter Server（以下, W2KDCS）を採用した勘定系ミッションクリティカルシステムである．

本稿では，システムの可用性（Availability）に焦点を当て，ES 7000 と W2KDCS の組み合わせで，如何にして高可用性（HA : High Availability）を実現したかを報告する．BANCS システムの HA 基盤として採用したクラスタサービスは Windows 2000 の標準的なコンポーネントであるが，厳しい可用性要件を実現するには標準的な機能だけでは不十分であったため，補完機能を開発してよりきめ細かな稼働監視と障害時対応を実現した．補完機能としては，アプリケーションや SQL Server などのプロセスの異常状態を詳細に検知する機能や複雑なフェイルオーバ条件の監視と制御を行う機能を実装した．

Abstract The BANCS system is the mission critical banking system that M bank and Nihon Unisys, Ltd. reconstructed, adopting ES 7000 of Unisys Corp. as hardware, and windows 2000 Datacenter Server(hereinafter, W2KDCS) of Microsoft Corp. as the operating system.

This paper discusses how the high availability(HA) system was implemented in the combination of ES 7000 and W2KDCS, focusing on the availability of the system. Although the cluster service, adopted as HA infrastructure of the BANCS system, was the standard component of Windows 2000, the standard function had not enough adequacy to meet the severe requirements for the availability. Therefore, we developed some complementary functions and implemented finer system monitoring and contingency actions, for example, the function that detects the abnormal state of a process within applications, SQL Server and others in detail, and the function to perform the monitoring and control of complicated fail-over conditions.

1. は じ め に

M 銀行と日本ユニシス株式会社（以下，当社）は，SA 銀行と SU 銀行の合併に伴い，両行でそれぞれ稼働していた BANCS システムを統合し，新対外（BANCS）システムとして再構築した．BANCS システム全体で共通なシステム概要とシステム要件に関しては本誌掲載の論文「BANCS 接続システム（1)-プロジェクト成功の鍵：POC 実践報告（山岸重雄著)」（以下，山岸論文）に記述されている．

山岸論文の 3 章で記述されているように，本システムで求められるシステム要件には性能要件，信頼性要件，可用性要件そして保守性要件があるが，本稿では可用性要件を主体に ES 7000 および W 2 KDCS の組み合わせで，如何にして HA（High Availability）システムを実現したかを述べる．

本システムの HA 基盤は Windows 2000 のクラスタサービスであるが，標準機能で

は不十分なところはそれを補完する機能を開発している．本稿では，まず始めに高可用性を実現するアーキテクチャの要点と障害対策の方針を述べる．次に，補完機能の必要性について述べ，具体的な稼働監視と障害時対応の実現方式を紹介する．

2. 高可用性を実現するアーキテクチャの要点

本章では，高可用性を実現するためのアーキテクチャの要点を述べる．

2.1 クラスタ構成

図 1 に示すように，HA システムの基盤は 2 ノードによるクラスタ構成と Windows 2000 のクラスタサービスである．2 ノードは双方同時に業務アプリケーションが稼働する，いわゆるアクティブ-アクティブ状態での運用が行われる．クラスタサービスはシェアード・ナッシング・モデルであり，両方のノードでアクセス可能な共有ディスクなどのクラスタ内のリソースは，一度に一つのノードによってのみ所有される．一方のノードで障害が検知されると，正常なノードが障害ノードで行っていた業務処理とデータを引き継ぐ．このことをフェイルオーバという．

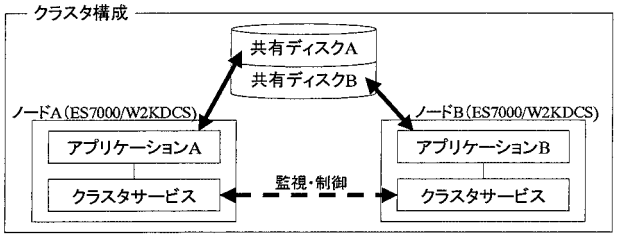

図 1 クラスタ構成

2.2 システムの配置とバックアップの形態

ノード障害時，センタ障害時のバックアップの形態を図 2 に示す．ここでいうバックアップとはノードあるいはセンタの片系のサービスが停止したときに，他系が処理を引き継ぐことを指す．

本システムには全体で 4 組のアプリケーション（AP 11，AP 12，AP 21，AP 22）が存在し，両センタ合わせて四つあるノードのうちいずれかで稼働する．

センタ内の片系ノードで障害が発生したときは，正常なノードが障害となったノードで行っていた処理とデータを自動的に引き継ぐ．図 2(a）の例で言えば，Node B で AP 12 が稼働している状態で，Node B にノード障害が発生すると，フェイルオーバにより Node A で AP 12 のバックアップが行われる．

災害やその他の重大障害によってセンタのシステム全体が停止したときは，中継センタ（外部）で BANCS 回線の切り替えが行われ，正常なセンタが被災したセンタで行っていた処理をバックアップする．図 2(b）の例で言えば，センタ 2 の Node C で AP 21，Node D で AP 22 が稼働している状態で，センタ 2 が被災した場合は，センタ 1 の Node A で AP 21，Node B で AP 22 のバックアップが行われる．センタバックアップではデータ（ディスク）は引き継がない．

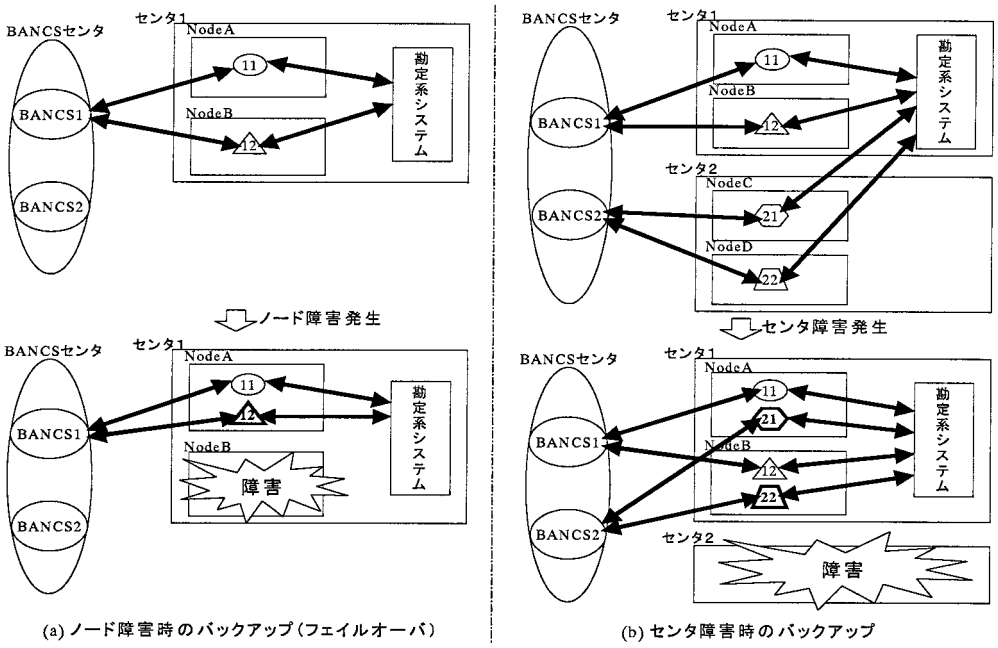

図 2　システムの配置とバックアップの形態

3.　障害対策方針

本章では，システムを構成する各コンポーネントに対する障害対策の方針とそれを実現するためのクラスタサービスの構成について述べる．

3.1　ハードウェアの障害対策

ハードウェアの障害対策は冗長構成が基本となる．障害時は予備のコンポーネントに自動的に切り替えられるため，サービスの停止を引き起こさず，業務への影響はほとんどない．ES 7000 はハードウェアの主要な部位において二重化を可能としている．本システムでは，電源，冷却ファン，Fiber Channel，LAN アダプタなどにおいて冗長構成をとり，可用性を高めている．

冗長構成要素がすべて障害となった場合はそのノードでの処理継続は不可能となる．この場合はクラスタサービスによって検知され，他のノードにフェイルオーバして処理が継続される．

3.2　ソフトウェアの障害対策

ソフトウェアの障害対策はクラスタサービスによる稼働監視が基本となる．

OS やクラスタサービスの障害はただちにノード全体の障害となるため，フェイルオーバするほかはない．クラスタを構成するノード間ではクラスタサービスが内部通信によって互いの死活監視を行っており，他ノードの障害を検知したらフェイルオーバが発動される．

SQL　Server や WinSAM などのミドルソフトウェアに関しては，同一ノード上のクラスタサービスが稼働監視を行い，障害を検知したら同一ノード上で再立ち上げを試み，規定時間以内に規定回数以上障害となった場合は，他のノードにフェイルオーバして処理を継続させる．

アプリケーションに関しては，可能な限りスレッドやプロセスを多重化し，単独障害ではサービスが停止しないようなプロセス構造になっている．プロセスの稼働監視は同一ノード上のクラスタサービスが行い，障害を検知したら同一ノード上で再立ち上げを試みる．規定時間以内に規定回数以上障害となった場合はそのプロセスは恒久的な障害となるが，多重化している別プロセスによってサービスは継続される．フェイルオーバは，多重化しているプロセスが複数障害となり，そのノード上でのサービス継続が困難であると判断したときにはじめて発動する．このように，フェイルオーバ発動をできる限り少なくすることで業務への影響が最小限になるようにする．

稼働監視と障害時対応はクラスタサービスの機能だが，標準機能では不十分なところは補完機能を開発して組み込む．補完機能の必要性と実現方式については，4章にて詳細を述べる．

表1に主なソフトウェアコンポーネントの障害対策を示す．同一ノード上でのプロセスの再立ち上げによるサービス停止時間は1分以内，フェイルオーバによるサービス停止時間は3分以内が障害対策の要件とされた．

表 1　ソフトウェアコンポーネントの障害対策

コンポーネント	障害内容	対策	障害発生時に起こる現象	停止時間(注)
OS	アボート	クラスタサービス	フェイルオーバ	3分以内
SQL Server	アボート	クラスタサービス	フェイルオーバ	3分以内
WinSAM	プロセス障害	クラスタサービス	①同じノード上での再立ち上げ ②フェイルオーバ	1分以内 3分以内
BANCS リスナ BANCS センダ	プロセス障害	多重化（回線数分）	別プロセスでのサービス継続	−
		クラスタサービス	同じノード上での再立ち上げ	1分以内
	閾値以上のプロセス障害	クラスタサービス	フェイルオーバ	3分以内
勘定系 FEP 接続インタフェース	プロセス障害	多重化(勘定系 FEP分)	別プロセスでのサービス継続	−
		クラスタサービス	同じノード上での再立ち上げ	1分以内
	全プロセス障害	クラスタサービス	フェイルオーバ	3分以内
BANCS アプリケーション	スレッド障害	スレッドの多重化	別スレッドでのサービス継続	−
	プロセス障害	クラスタサービス	①同じノード上での再立ち上げ ②フェイルオーバ	1分以内 3分以内

(注)停止時間はサービス停止時点からサービス再開時点までの時間（目標数値）

3.3　クラスタリング構成

ソフトウェア，ハードウェアの各コンポーネントの可用性を高めるために，それらをクラスタリング環境で構成し，管理する．クラスタサービスの配下で管理される実体のことをクラスタリソースという．システムの任意の構成要素をクラスタリソースとすることができる．例えば，IPアドレス，ネットワーク名（仮想サーバ名），物理ディスク，ファイル共有，プロセス，サービスなどである．

クラスタリソースの論理的な集合をクラスタグループという．依存関係のあるクラスタリソースはすべて同じクラスタグループに収められる．クラスタグループは同時に一つのノードだけに所有され，その単位でノード間を移動する．

本システムでは，各センタにそれぞれ四つのクラスタグループを定義した．1セン

タ内でのクラスタグループ構成を表 2 に示す．

表 2　クラスタグループ構成

クラスタグループ	実行可能ノード	優先ノード	含まれる主なクラスタリソース	通常時の状態
BANCS センタ接続 I/F 群#1	Node1	Node1	WinSAM，BANCS リスナ／センダ	○
BANCS センタ接続 I/F 群#2	Node2	Node2	WinSAM，BANCS リスナ／センダ	○
SQL Server & AP 群#1	Node1, Node2	Node1	SQL Server 群（SQL Server，物理ディスク）	○
			通常 AP 群(AP，勘定系 FEP 接続 I/F)	○
			被災 AP 群(AP，勘定系 FEP 接続 I/F)	△
SQL Server & AP 群#2	Node1, Node2	Node2	SQL Server 群（SQL Server，物理ディスク）	○
			通常 AP 群(AP，勘定系 FEP 接続 I/F)	○
			被災 AP 群(AP，勘定系 FEP 接続 I/F)	△

（注）通常時の状態：○（オンライン）△（オフライン）

サービス開始時は優先ノードがクラスタグループの所有権を持つが，片系ノードで障害が発生したときは自動的に正常な方のノードに所有権が移動する．すなわち，クラスタグループがフェイルオーバの単位となる．ただし，クラスタグループ「BANCS センタ接続インタフェース群（#1，#2）」は，実行可能なノードを固定にしており，障害が発生しても他のノードへ移動することはない．これは，WinSAM が同一ノード上で複数インスタンスを起動させることができないためである．BANCS リスナ，BANCS センダは WinSAM との依存関係を設定する必要があるため同一のグループとした．

クラスタグループ「SQL Server & AP 群（#1，#2)」に関しては，その中をさらに SQL Server 群，通常 AP 群および被災 AP 群という三つのクラスタリソース群(仮想的なクラスタグループ）に分類し，その単位でオンライン/オフラインするという運用を行う．通常サービス時は被災 AP 群を除くすべてのリソースがオンラインになっており，被災対応時には被災 AP 群もオンラインにする．当初の設計では，これら三つのクラスタリソース群を独立したクラスタグループとしていたが，フェイルオーバ時に関連するグループ間の同期処理が複雑になるため，フェイルオーバの単位とクラスタグループが完全に一致するように変更した．

4.　クラスタサービスを補完する稼働監視と障害時対応機能

本章では，クラスタサービスを補完する目的で開発した稼働監視と障害時対応の機能について，その必要性と実現方式を紹介する．

4.1　補完機能の必要性

Windows 2000 が標準で備えるクラスタサービスは，登録されているクラスタリソースの稼働監視を行い，障害を検知した場合は，事前定義にしたがって再起動やフェイルオーバといった制御を行う．表 3 は，各ソフトウェアコンポーネントに関して，検知しなければならない障害の内容と必要なアクション，そしてそれらがクラスタサービスの標準機能で実現可能かどうかを示している．

表 3　ソフトウェアコンポーネントの障害対策における補完機能の必要性

コンポーネント	障害内容	必要なアクション	標準	補完
OS	ブルースクリーン	フェイルオーバ	○	−
	クラスタサービスの異常終了	フェイルオーバ	○	−
SQL Server	プロセスの異常終了	フェイルオーバ	○	−
	ハングアップ	フェイルオーバ	○	−
	システムデータベースの異常	フェイルオーバ後，恒久的な障害状態となる→1DB 運用発動(手動)	×	○
	トランザクションログの異常	フェイルオーバ後，恒久的な障害状態となる→1DB 運用発動(手動)	×	○
	業務データベースの異常	フェイルオーバ後，恒久的な障害状態となる→1DB 運用発動(手動)	×	○
WinSAM	プロセスの異常終了	サービスの再起動	○	○
	ハングアップ	サービスの再起動	×	○
	規定時間以内に規定回数以上障害	フェイルオーバ	×	○
BANCS リスナ BANCS センダ	プロセスの異常終了	プロセスの再起動	○	○
	ハングアップ	プロセスの再起動	×	○
	規定時間以内に規定回数以上障害	恒久的な障害状態とする	○	−
	規定回線数以上の BANCS リスナ，BANCS センダが恒久的な障害となった	フェイルオーバ	×	○
勘定系 FEP 接続インタフェース	プロセスの異常終了	プロセスの再起動	○	○
	ハングアップ	プロセスの再起動	×	○
	規定時間以内に規定回数以上障害	恒久的な障害状態とする	○	−
	全プロセスが恒久的な障害となった	フェイルオーバ	×	○
BANCS アプリケーション	プロセスの異常終了	プロセスの再起動	○	○
	ハングアップ	プロセスの再起動	×	○
	規定時間以内に規定回数以上障害	フェイルオーバ	○	−

(注) 標準：○(標準機能で実現可能)　×(標準機能では実現不可)
　　補完：○(補完機能を実装)　−(標準機能を使用)

4.1.1 障害の検知

クラスタサービスは標準的にいくつかのクラスタリソースの種類をサポートしている．しかし，これらを本システムの監視対象コンポーネントに適用した場合，次のような点が不十分となる．

1) 外部システムとの接続インタフェースやアプリケーションのプロセス群をクラスタリング環境で動かすには，クラスタサービスが標準提供している「汎用アプリケーション」というリソースの種類を使って登録するのが簡単である．しかし，この方法では，障害の検知はプロセスが死んだことでしか行えない．例えば，ハングアップや無限ループなどアプリケーション的にプロセスが機能していない状態は検知できない．

2) SQL Server 2000 は標準的にクラスタサービスに対応しており，専用のリソースの種類が提供されている．しかし，障害の検知はアプリケーションがデータベースに接続できない全事象をカバーしていない．例えば，システムデータベースやトランザクションログ，業務データベースの障害は検知できない．

3) WinSAM はクラスタサービスに対応したソフトウェアではないため，クラス

タサービスが標準提供している「汎用サービス」というリソースの種類を使って登録することで，クラスタサービスからサービスの開始/終了が行える．しかし，障害の検知はサービスが停止したことを検知できるだけであって，実際にアプリケーションが WinSAM とのデータの送受信を行えることを保証するものではない．

以上のように，クラスタサービスの標準機能だけでは，本システムにおいて検知が必要とされる障害事象を完全にはカバーできない．

4.1.2 障害の復旧

BANCS リスナ，BANCS センダおよび勘定系 FEP 接続インタフェースのプロセスは可用性を高めるため多重化されており，フェイルオーバ条件は複雑である．フェイルオーバ発動をできるだけ減らすため，単独障害ではフェイルオーバさせず，一定の条件に達したときにフェイルオーバさせなければならない．

クラスタサービスにはクラスタリソースの障害を検知したときにとるアクションとして，表 4 に示すいずれかを設定できる．

表 4　障害時のアクションの種類

アクションの種類	再起動	フェイルオーバ
RestartNotify	既定時間以内に既定回数までは再起動	所属するグループをフェイルオーバする
RestartNoNotify	既定時間以内に既定回数までは再起動	フェイルオーバしない（恒久的な障害となる）
DontRestart	再起動はしない	フェイルオーバしない

これらのアクションはクラスタリソース単位でしか設定できないため，多重化された複数のプロセスの状態を判断してフェイルオーバさせることはできない．

また，WinSAM や BANCS リスナ，BANCS センダの障害によってフェイルオーバする場合，自身が属するクラスタグループはフェイルオーバの対象ではないためフェイルオーバしないが，当該ノード上で稼働していた他のクラスタグループをフェイルオーバさせなければならない．このようなこともクラスタサービスの標準機能では実現できない．

4.2 稼働監視の補完機能

4.2.1 クラスタサービスとカスタマイズの機構

Windows 2000 のクラスタサービスは次の三つの主要なコンポーネントから構成される．

・クラスタサービス（本体）
・リソースモニタ
・リソース DLL

クラスタサービスは各ノードに一つずつあるインスタンスで，他ノードのクラスタサービスとの通信，イベント通知処理，フェイルオーバ操作，クラスタオブジェクトの構成管理などを担当する中核コンポーネントである．

リソースモニタはクラスタサービスとクラスタリソースの間に位置するコンポーネントで独立したプロセスとして動作する．クラスタサービスからクラスタリソースへ

の要求やクラスタサービスへのイベントの報告はリソースモニタが中継する．

リソース DLL はリソースモニタから呼び出される API（エントリポイント）の実装である．リソースが受け付けるオンライン，オフライン，ポーリング（監視）といった要求には，それぞれ対応したエントリポイントが決まっており，関数の実体はリソース DLL に実装されている．リソース DLL は特定のリソースの種類に対応しており，リソース固有の処理が実装されている．

図 3 に Windows 2000 のクラスタサービスのアーキテクチャを示す．

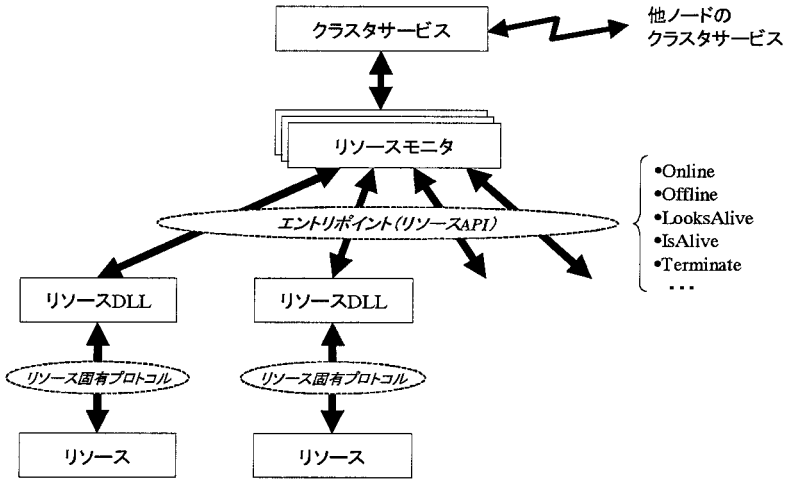

図 3　クラスタサービスのアーキテクチャ

標準提供されているリソース DLL に加えて，新たに独自のリソース DLL を作成することで，リソースの種類を追加したり，リソース DLL を置き換えたりすることができるようになっている．この仕組みを利用すれば，監視対象に最適な障害検知ロジックを組み込むことができる．

リソース DLL に実装しなければならないエントリポイントの主なものを表 5 に示す．

表 5　リソース DLL のエントリポイント

エントリポイント	リソースモニタから呼び出されるタイミング	処理内容
Startup	クラスタサービスの開始時などリソース DLL がロードされるとき	他のエントリポイントの関数テーブルをリソースモニタに返す．
Open	クラスタサービスの開始時やリソース作成時	リソース固有情報の初期化処理
Close	クラスタサービスの終了時やリソース削除時	リソース固有情報の終了処理
Online	リソースのオンラインを指示したとき	リソースの立ち上げ
Offline	リソースのオフラインを指示したとき	リソースの正常終了
LooksAlive	一定間隔でのポーリング（頻度大）	リソース死活のおおまかな評価
IsAlive	一定間隔でのポーリング（頻度小）	リソース死活の厳密な評価
Terminate	リソースの異常検知時	リソースの緊急終了

リソースとしてプロセスを例にあげれば，プロセスの生成は Online に，プロセスの安全な終了は Offline に実装する．プロセスの死活監視は LooksAlive や IsAlive に

実装する．プロセスの強制的な停止は Terminate に実装する．

4.2.2 稼働監視の補完機能実現方式

本システムにおける稼働監視の補完機能に対する実現方式を図 4 に示す．

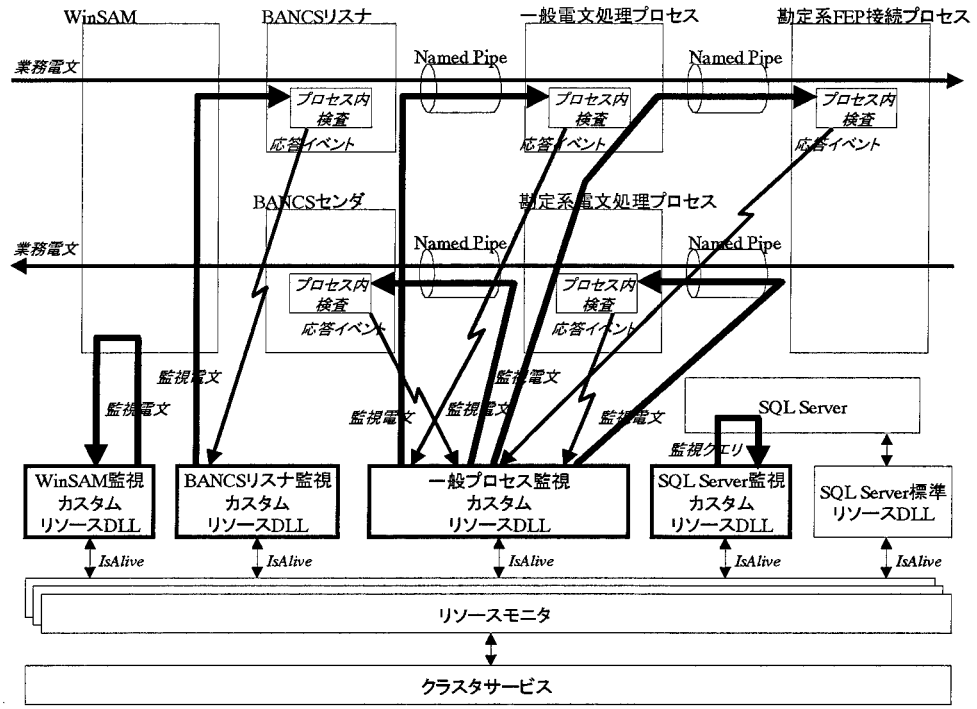

図 4　稼働監視の補完機能実現方式

標準機能を補完する障害検知ロジックはカスタムリソース DLL として実装され，ステータスはリソースモニタを経由してクラスタサービスに通知される．リソース DLL 内の監視スレッドからは一定間隔で対象リソースに監視電文が送信され，規定時間以内に応答を受信すれば監視対象は機能していると判断される．

BANCS センダ，勘定系 FEP 接続プロセス，BANCS アプリケーションなどのプロセスに対しては，実際に業務電文の受信に使われる名前付きパイプに対して監視電文を送信する．これは各プロセスが使用する名前付きパイプの検査にもなっている．例外として，BANCS リスナは業務電文を WinSAM から受信するため，監視電文も WinSAM 経由で送信する．各プロセスでは電文を受信後，それが監視電文だったら速やかに必要な検査を行い，イベントの送信をもって応答を返す．

SQL　Server については，標準で提供されるリソース DLL はそのまま使用し，それを補完する形でカスタムリソース DLL を作成した．サービスの起動，終了や基本的な監視は SQL Server 標準のリソース DLL が行う．カスタムリソース DLL では，監視スレッドから一定間隔で，毎回 SQL Server に接続し，指定されたストアドプロシージャを実行させることによって，標準のリソース DLL では検知が不可能なデータベース障害などを検知する．

WinSAMについては，WinSAMでの折り返し電文の送受信によって，WinSAMが機能していることを確認する．

4.2.3 障害検知の実装

本システムで作成したカスタムリソースDLLにおける障害検知の実装上の要点を述べる．障害検知は次の二つの手法によって実現している．

・監視電文によるポーリング

・非同期障害検知（Asynchronous Failure Detection）

監視電文によるポーリングは，監視対象が生きていても機能していない事象を検知するために使われている．監視電文によるポーリングは，独立したスレッド内で実行して結果を保持しておき，監視結果は後からIsAliveエントリポイント関数でリソースモニタに返す．IsAliveエントリポイント関数の中で実際のポーリングを行わないのは，リソースモニタには300ミリ秒以内にステータスを返す必要があるためである．このほか，OnlineやOfflineなどにおいてもリソースDLL内で時間がかかる場合は，別スレッドを作成して処理する必要がある．リソースモニタはIsAliveのステータスがFALSEだと，ただちにTerminateを呼び出す．Terminateエントリポイント関数では，終了要求イベントを送信してプロセスの終了を待ち，規定時間以内に終了しない場合は，プロセスを強制終了する．プロセスを強制終了する前にプロセスダンプを出力させることもできるようにしている．監視対象のプロセスには，終了要求イベントを受信したら必要な終了処理を行うような実装がなされている．

プロセスの異常終了に関しては，速やかにそれを検知するために非同期障害検知と呼ばれる仕組みを利用している．これは，IsAliveによる監視とは非同期に，異常を検知したら直ちにリソースモニタにイベントを通知することができる仕組みである．

4.3 障害時対応の補完機能

4.3.1 フェイルオーバ条件監視

主なクラスタリソースにおける障害時のアクションの設定を表6に示す．

表 6 クラスタリソースの障害時のアクション

クラスタグループ	クラスタリソース	アクション	再開しきい値	期間(秒)
BANCSセンタ接続インタフェース群	WinSAM	RestartNoNotify	1	900
	BANCSリスナ	RestartNoNotify	1	900
	BANCSセンダ	RestartNoNotify	1	900
SQL Server&アプリケーション群	物理ディスク	RestartNotify	0	900
	IPアドレス	RestartNotify	0	900
	SQL Server	RestartNotify	0	900
	BANCSアプリケーション	RestartNotify	1	900
	勘定系FEP接続インタフェース	RestartNoNotify	1	900

RestartNotifyが設定されているクラスタリソースは，恒久的な障害になると自身が属するクラスタグループが自動的にフェイルオーバする．

RestartNoNotifyが設定されているWinSAM，BANCSリスナ，BANCSセンダ，勘定系FEP接続インタフェースについては，再起動は行われるが，恒久的な障害になってもクラスタサービス自身の機能ではフェイルオーバしない．これらのクラスタ

リソースについては，表 7 に示すようなフェイルオーバ条件とアクションが必要である．

表 7　フェイルオーバ条件とアクション

フェイルオーバ条件	アクション
WinSAM が恒久的な障害になった	障害ノード上で所有権を持つ移動可能なクラスタグループをすべてフェイルオーバさせる．
BANCS リスナまたは BANCS センダが規定回線数以上恒久的な障害になった	障害ノード上で所有権を持つ移動可能なクラスタグループをすべてフェイルオーバさせる．
多重化されている勘定系 FEP 接続プロセスがすべて恒久的な障害になった．	自身が所属するクラスタグループをフェイルオーバさせる．

本システムでは，これら三つのフェイルオーバ条件を監視し，フェイルオーバ条件に達したらフェイルオーバさせる外部プログラム（フェイルオーバ条件監視プログラム）を作成した．このプロセス自身も汎用アプリケーションとしてクラスタサービスに登録している．

フェイルオーバ条件監視プログラムは，クラスタ API を使ってクラスタサービスに対する監視と制御を行うアプリケーションである．クラスタオブジェクトに関するイベントはクラスタ通知ポート（Notification　Port）と呼ばれるインタフェースを用いて監視できる．WinSAM や BANCS リスナ，BANCS センダ，勘定系 FEP 接続インタフェースのリソースの状態変化をクラスタ通知ポートによって検知し，その障害状況を評価し，フェイルオーバ条件に達していたら，クラスタサービスにフェイルオーバを実行させる．

4.3.2　ランチャー機能

本システムでは，サービスの開始時やフェイルオーバ直後に UST の振り分け先 IP アドレスを設定するコマンドや回線オープンコマンド，業務のリカバリバッチなどを実行させる必要があった．これはコマンドやバッチをクラスタリソースとして登録し，オンライン開始時に実行させることで実現可能である．この機能をランチャー機能と呼ぶ．ランチャー機能はクラスタサービス標準機能でも汎用アプリケーションリソースとして登録することで実現は可能だが，プロセスが正常終了しても死んだと判断され，リソースが障害状態になってしまうため具合が悪い．このため，コマンドやバッチのような非常駐プロセスを登録するためのカスタムリソース DLL を実装した．このカスタムリソース DLL の実装では Online エントリポイント関数で指定されたコマンドを実行し，正常終了したらリソースの状態をオンラインとする．IsAlive に対しては常に TRUE を返し，リソースへの稼働監視は行わない．正常終了しなかった場合は障害状態とする．

4.3.3　障害サーバシャットダウン

クラスタサービス障害が発生すると，フェイルオーバしても障害ノード上にプロセスが残ってしまい，アプリケーションが予期せぬ動作をする可能性がある．例えば，フェイルオーバ後に勘定系 FEP 接続インタフェースのプロセス群が障害サーバ上に残っていると，勘定系 FEP 上のメッセージキューから電文を横取りしてしまうとい

うことがある．SQL Server もクラスタサービス障害時に障害ノードにプロセスが残ってしまうことがわかっている．

この問題を解決するため，フェイルオーバが発生したら，即時に障害ノードをシャットダウンさせる．この機能は，優先ノード以外でオンラインとなったら他ノードをシャットダウンさせるコマンドとして実装し，前項のランチャー機能を使ってクラスタサービスに登録している．

5. 評価と考察

HA システム構築の基盤技術は Windows 2000 クラスタサービスであるが，よりきめ細かな稼働監視と障害時対応を目的として補完機能の開発を行った．高可用性という観点で補完機能が特に有効であったのは，標準機能では検知不可能な障害が検知できるようになり，早期にシステムの異常状態を発見できるようになったことである．

システム全体の可用性を数量的に評価することは難しいが，ここでは一つの指標として，障害時の再立ち上げおよびフェイルオーバにおけるサービス停止時間をとりあげる．代表的な障害ケースについて測定した結果を表 8 に示す．

表 8 サービス停止時間測定結果

障害内容	再立ち上げ（秒）	フェイルオーバ（秒）
バックボーン側 LAN アダプタ 2 枚障害	−	232
BANCS センタ側 LAN アダプタ 2 枚障害	−	101
OS 障害	−	69
クラスタサービス障害	−	68
SQL Server 障害	−	97
WinSAM 障害	25	113
BANCS リスナ障害	1 以下	76
勘定系 FEP 接続インタフェース障害	1 以下	66
一般電文処理プロセス障害	1 以下	62

再立ち上げ 1 分以内，フェイルオーバ 3 分以内という目標値はほぼ達成できたが，唯一達成できなかったのはバックボーン側 LAN アダプタ 2 枚障害時のフェイルオーバ時間である．ネックになっているのは，SQL Server と SQL Server Agent のオフラインで合計 100 秒かかっていることである．これは SQL Server と SQL Server Agent はオフライン時にドメインコントローラにアクセスしようとするが，ドメインコントローラがバックボーン LAN につながっているため，タイムアウトするまで終了できないことが原因であることがわかった．もっとも，LAN アダプタは冗長化されているため，これによって可用性が大きく低下することはないだろう．

障害箇所によって時間にばらつきがあるが，特に，障害を検知したリソースに依存しているリソースが多い場合は，障害を検知してから再立ち上げやフェイルオーバを開始するまでに時間がかかることがわかった．クラスタサービスはリソースの障害を検知すると，それに依存しているリソースを順番にオフラインにした後で，再開しきい値を検査し，再起動するか，恒久的な障害とするかを判断する，という動きをする．WinSAM には依存しているクラスタリソースが 25 個もあり，WinSAM 障害時には

それらを逐次にオフラインするために時間がかかっている．

6. おわりに

ES 7000 と W2KDCS の組み合わせで，勘定系ミッションクリティカルシステムに求められる高可用性要件を如何にして実現したかを述べてきた．

更なる高可用性要件を実現するには，早期に障害を見つけ，迅速な回復を行うために，きめ細かな稼働監視と障害時対応が必要となる．そういう意味では，Windows 2000 のクラスタサービスが標準で提供している機能だけでは不十分で，補完機能の開発を考える必要がでてくる．

なお，今回開発した HA 機能は BANCS システムだけに有効なものではなく，これを発展させ HA システム構築を支援する汎用性に富んだ信頼性の高いミドルソフトウェアとして新たに開発済みである．

Windows プラットフォームでのシステム構築において高可用性を要求されるケースは今後も増えていくであろう．本稿がその際の参考になれば幸いである．

参考文献　［1］ 小川康博，Unisys e-@ction Enterprise Server ES 7000 のメインフレーム同等の RAS 技術，Unisys 技報，第 66 号
［2］ Microsoft Cluster Server（MSCS）リソースダイナミックリンクライブラリ（DLL）の作成，Microsoft White Paper

執筆者紹介　溝　上　昌　宏（Masahiro Mizogami）
1965 年生．1989 年名古屋大学工学部情報工学科卒業．同年日本ユニシス(株)入社．人工知能，オブジェクト指向技術支援，CORBA 製品開発，BANCS システム開発等を経て，現在，インテグレーションサービス部 .NET ビジネスディベロプメントに所属．