システムコンバージョンにおける基盤設計事例
－メインフレームからAIXへの移行－

テクニカルコンピテンシー部
基盤技術チーム
ITスペシャリスト

鈴木　俊之

Toshiyuki　Suzuki
toshiyuki-suzuki@exa-corp.co.jp

業務ノウハウが蓄積されたアプリケーションに極力手を入れず、ハードウェアを老朽化したメインフレームから価格性能比の高いオープン系システムに置き換えるシステムコンバージョンが注目されている。当社でもシステム・トランスフォーメーション・サービスの一環として取り組んでいるが、その中から金融系業務システムのコンバージョンにおける基盤設計事例について報告する。そこではメインフレームの持つ高い信頼性をオープン系システムで如何に実現するかが技術的な課題となる。信頼性を構成する可用性、リアルタイム性、あるいは運用といった技術要素の移行検討を行い、移行先となるオープン系システムのシステム基盤アーキテクチャの設計方法や設計時の考慮事項などを整理・体系化し、提示した。

1. はじめに

システムコンバージョンビジネスが市場の注目を浴びている。一昔前のシステムダウンサイジングによる分散化で企業内に散らばったハードウェア資源を統合することでTCO削減を狙ったもの、寿命が尽きたハードウェアを置き換えることにより業務アプリケーションシステムの延命を狙ったものなど、企業の置かれたシステム環境によりニーズは様々である。ここで共通するのは、業務ノウハウが蓄積されたアプリケーションに手を加えず、単純にハードウェアを移行して使い続けたいという要求である。アプリケーションを単純移行したいという要求から当然ではあるが、移行に必要なコストは最小化されることが望ましい。

このような要求に応えるため、当社では様々な形態のシステム・トランスフォーメーション・サービスを提供してきた。特に、メインフレームからオープン系システムへ移行する場合、アプリケーションを単純に移行するだけでも、システム基盤の観点からは解決すべき多くの課題が存在する。中でも、現行システムの信頼性確保の問題が重要である。メインフレームの特徴はその高い信頼性にある。ハードウェアレベルで冗長化されており、障害が発生しても処理の継続が可能となっている。加えて、プロセス管理機能が充実しており、リアルタイム性を持つ処理に対応できる。一方で、オープン系コンピュータは高い価格性能比が特徴である。しかし、一般にはハードウェアレベルでの冗長化がなされておらず、障害が発生すると処理が打ち切りとなる。また、プロセス管理機能が簡素化されておりリアルタイム性に欠けるなど信頼性の面ではメインフレームにはおよばない。すなわち実行するアプリケーションの特性を考慮しつつ、メインフレームの持つ高い信頼性をオープン系システムで如何に実現するかが、移行設計の要となる。

本稿では、この視点から特に可用性とリアルタイム性の確保に注目して実施したシステム基盤アーキテクチャの設計方法と内容を事例に基づき紹介する。対象は金融系の業務システムで、メインフレームを価格性能比の高いオープン系システムに置き換えることによりアプリケーションを延命させ、同時に運用コストの削減を図ったものである。移行設計においては、現行の機能・非機能要件を保証するという制約から、ある程度設計方法の一般化が行える。要求事項の導出、設計方針の検討、設計の進め方および設計時の考慮事項などを極力整理、体系化して記述するよう努めた。

まず第２章で、移行先のオープン系システムのアーキテクチャ設計に必要な要求事項や設計方針の導出について述べる。続く第３章では、第２章の方針・要件に基づくアーキテクチャ設計の方法と内容について述べ、最後に第4章で結果を評価する。

2. アーキテクチャ設計の要求事項と基本方針

システム基盤の移行設計に必要な要求項目は、業務アプリケーション機能から定まる機能要件と、機能をシステムとして実現するための性能や運用にかかわる非機能要件に大別される。システム移行の大前提は現行保証であり、機能移行だけでなく現行稼動しているシステムの性能や運用を正確に把握し、非機能要件として明確化することが重要である。また、機能移行については移行コストが大きい業務アプリケーションの変更を最小に抑えることが前提となる。

ここでメインフレームからオープン系システムへのシステム移行を対象に、システム基盤として検討すべき要求項目と検討ポイントを整理・体系化し表1にまとめた。同表には今回事例として取り上げた、移行対象となるIBMメインフレーム上のシステム要件も併記した。以降は表の検討項目に従い、移行設計に必要な要求事項の明確化と、設計の基本方針策定を検討した結果について述べる。

2.1. 機能要件

メインフレーム上の業務システムの処理は、オンライン機能とバッチ機能に大きく分けられる。オンライン機能の基盤はトランザクションモニタであり、バッチ機能はジョブが基本となっている。オンラインもバッチもデータ管理機能（DB管理システム）が管理するデータを利用している。DB管理システムには、表形式のRDB型とファイル型のものがある。本事例では新規に追加となった機能に業務端末のWeb化がある。PC上のアプリケーションで表示していた業務端末をWeb化し、ブラウザ上で業務端末を使用する。これ以外に見落としてはいけない機能に対外接続がある。以下では、これら各機能をオープン系システムに移行する場合に明確化すべき要求事項や、設定すべき基本方針について述べる。

表１　メインフレームからオープン系システムへのシステム基盤の移行検討項目

大分類	検討項目	検討ポイント	移行対象メインフレーム要件
機能要件	オンライン	・トランザクションモニタ	・CICS
	バッチ	・ジョブ	・JCL
	DB 管理	・RDB ・ファイル型/階層型 DB	・DB2 ・VSAM
	端末	・端末の Web 化	・3270 端末
	対外接続	・ファイル転送 ・文字コード	・CMT/SNA ・EBCDIC
非機能要件	性能	・CPU 能力/アーキテクチャ ・ストレージ容量 ・キャパシティプランニング	・60MIPS ・0.6TB ・3 年間、2 倍
	可用性	・停止許容時間 ・停止タイミング ・対障害性/復旧性	・ATM/CD は無停止（FEP となる専用機あり） ・障害発生時は一定時間停止可
	リアルタイム性	・リソース競合	・OS と CICS が一体化制御
	運用	・ジョブ管理 ・バックアップ ・障害監視 ・運用管理ツール	・毎日計画停止 ・DB バックアップはオフライン ・バッチは夜間 ・障害監視はシステム標準機能

2.1.1. オンライン

オンライン（ネットワーク接続）処理として、管理用業務端末、ATM（Automated Teller Machine:現金自動預け払い機）およびCD（Cash Dispenser:現金自動支払機）などの処理がある。このオンライン処理において、トランザクションと呼ばれる不可分な操作を保証する仕組みが必要である。例えば、ある口座から別の口座にお金を移動させる場合、引き落とし操作と入金操作が両方とも成功するか両方とも失敗しなければならない。個々の操作ごとにロジックを組んで保証することは難しいため、トランザクションモニタと呼ばれる、トランザクションを一括管理できるミドルウェアを使用するのが一般的である。

本事例では、トランザクションモニタとしてIBM社製のCICS（Customer Information Control System）を使用していた。CICSはその名のとおり、キャラクターベースのユーザインタフェース機能とトランザクション管理機能を不可分に一体化してある。このため、別のトランザクションモニタを使用する場合は、GUIおよびトランザクション管理部分の設計から行うこととなり、業務アプリケーションプログラムを単純に移植できなくなる。これは、移行コストの大きい業務アプリケーションの変更を最小に抑えるという前提条件に反する。

このため、トランザクションモニタとして、CICSをオープン系コンピュータに移植したTXSeriesという製品を採用することとした。ただし、CICSはメインフレームのアーキテクチャに特化した部分が多いため、TXSeriesとして移植されているのは基本部分だけで完全互換ではない。それでも、全面作り直しより非互換部分の変更の方が作業量は少ないと判断した。

2.1.2. バッチ

バッチは、JCL（Job Control Language）でジョブを組んでオペレータが投入していた。オープン系システムでは、JCLをシェルスクリプトに変換することになる。コンバージョンツールも存在するが、ファイルやデータ資源に対するアクセス方法や表記方法が異なるため人手による修正が必要であり、かつ一般にジョブの数が多いため、移行時の見積り等で注意が必要である。また、多数のジョブをコン

バートしたシェルスクリプトの実行および結果を管理する仕組みも必要となり、ジョブスケジューラを導入して管理することとした。

2.1.3 データ管理

現行の金融系業務システムではデータ管理にDB2とVSAM(Virtual Storage Access Method)を使用していたが、オープン系システムへの移行に際し、データ管理システムの統合化を検討した。複数のデータ管理システムを継続し維持・運用するのは、システムのライフサイクルを考えると好ましくないと判断し、VSAMをDB2 UDBに置き換え、データ管理をDB2 UDBで統合することとした。これにより、VSAMを使うプログラムをDB2 UDBに対応させ書き換える必要が生じるが、データ管理の一元化による効果の方が大きいと判断した。

2.1.4 Web端末

メインフレームではPC上の3270端末エミュレータを経由して、端末画面単位で入出力を行っている。端末との入出力制御はCICS固有の機能で行っており、画面定義をmapと呼ばれる方法で記述している。

本事例では、3270端末をより汎用性の高いWeb端末に置き換えるという新たな機能要件が追加された。このため、ブラウザで端末の入出力を行えるようWebアプリケーションサーバを導入し、mapを流用しWeb化するためのフレームワークを新規にJavaベースで開発することとした。

トランザクションモニタとして採用したTXSeriesとの親和性を考慮し、Webアプリケーションサーバは、同じIBMのWAS(WebSphere Application Server)を採用することとした。

2.1.5. 対外接続

メインフレームでは、外部(他社)とのデータのやり取りにファイル転送、CMT(Cartridge Magnetic Tape)およびSNA(Systems Network Architecture)経由の電文を使用していた。現行の機能を保障するためには、オープン系システムのファイル転送も転送完了の保証ができる方式が必要である。また、CMTは外部とのやり取りの他に、レポート作成のためにDBのバックアップを読み込んでいる。

対外接続の通信はオープン系システムではSNAではなくTCP/IPとなるため、通信の信頼性を考慮するとミドルウェアとしてMQ(IBM WebSphere MQ)を用いるのが望ましい。ただし、本事例では、FEP(Front End Processor)となる専用機との接続方式に制約があり、MQは用いていない。

対外接続において文字コードの相違が問題となる。メインフレームの文字コードの標準はEBCDICで日本語はIBM漢字である。オープン系コンピュータではUTF-8、EUC-JP、Shift_JISが文字コードの標準となっているが、トランザクションモニタとして選定したTXSeriesはUTF-8をサポートしていないこと、およびWindowsサーバとの接続性を考慮し、オープン系システムの文字コードはShift_JISとした。

移行後にも、メインフレームとのファイル転送が残っている。ファイル転送には、転送の完了の保証が行えないオープン系コンピュータの標準のFTPは使わず、HULFT(セゾン情報システムズが開発したファイル転送ソフト)を採用した。また、メインフレームとのやり取りに必要な文字コード変換にもHULFTの機能を用いた。

また、オープン系システムへの移行に合わせて、過去のCMTを用いたバックアップデータはメディア変換し、他社とのCMTのやり取りはメディアを変えるまたはネットワーク経由に変更することで、CMTを廃止する方針とした。

2.2. 非機能要件

非機能要件には、性能、可用性、リアルタイム性および運用がある。前節と同様、それぞれに対する要求事項や基本方針について述べる。

2.2.1. 性能

現行のメインフレーム性能を保証するだけでなく、将来の業務拡大による負荷およびデータ量の増加に対応できなければならない。増加量は年25%の増加で３年間推移し、３年後に約２倍になることを想定した。現行のメインフレームのスペックは約60MIPS(Million Instructions Per Second)程で負荷的に余裕がない状態であった。また、ストレージの実使用量は0.6TB程であった。約２倍に対応するにはおのおの120MIPSに1.2TBとなる。

なお、トランザクションの処理速度は、1日の総数より、３件/秒程度と推定した。

2.2.2. 可用性

ATMやCDなど、365日24時間でリアルタイム性を必要とする端末は無停止が前提となる。現行でもメインフレーム単体ではなくフォールトトレラントな専用機をFEPに使ってサービス無停止を実現していた。移行後も同じFEPを継続して用い、サービス無停止とする。

一方、管理用業務端末のサービスは金融機関の営業時間だけであり、かつ短時間であれば停止も許されていた。よって、オープン系システムでも、障害発生時は一定時間の停止を認めサービスが継続できればよい。これを満足するため、稼動系/待機系の２系からなる高可用性クラスタ構成を採用した。通常時は稼動系でサービスを行い、障害発生時は待機系に切替えサービスを継続する。

2.2.3. リアルタイム性

オンライン機能には、リアルタイム性が必要となる。しかし、メインフレームに比べオープン系コンピュータのOSはプロセスの管理が簡略化されており、オープン系コンピュータのOSに於ける優先度の指定はCPUのサービス優先度の指定でしかない。オープン系コンピュータのOSでも、リソースの使用可能サイズの制限はかけられる。しかし、これは一つのプロセスが異常にリソースを消費して、他のプロセスを圧迫しないようにするための安全弁であって、優先処理には使えない。制限を超えてリソースを使おうとしたプロセスはOSに強制終了されてしまう。

プロセス実行の優先度を上げても、プロセス間でシステムのリソースの競合が起きた場合、リソースの取得は早い者勝ちとなり、優先処理とはならない。

競合が起きやすいシステムのリソースに物理メモリがある。オープン系のOSでは、物理メモリをどのプロセスに優先的に幾らあたえるかを実行時に指定できない。物理メモリで競合が起きると早い者勝ちとなりリアルタイム性が失われてしまう。

このような考察から、オープン系コンピュータのOSでは優先度の異なるプロセス間でリソースの競合を起こさないよう、サーバを分ける方法がリアルタイム性を維持する単純かつ効果的な方法となる。そこで、リアルタイム性を確保するため、システムを構成するミドルウェアごとに独立したサーバを立てるという方針を採用した。

2.2.4. 運用

メインフレームはハードウェアの信頼性が高く、障害発生頻度が低い。このため、障害対応も手順を定め、人手による運用としているケースが多い。しかし、オープン系システムではハードウェアの信頼性が低いため障害発生頻度が高く、ミスが伴う人手による運用にすべてを委ねることは危険である。そこで、極力運用管理パッケージを導入し、運用業務の効率化を図ると同時に、フールプルーフ設計を行うことを基本方針とした。

高可用性を実現する構成にはシステムの機能を監視する障害監視が必要である。OSレベルの障害検出と対応は高可用性クラスタパッケージに任せ、サービスレベルの障害検出と対応を運用管理パッケージの監視機能で対処し、統合コンソールに各サーバの監視状態をまとめて表示することとした。また、検出した障害に対応した復旧操作を実行管理する仕組みも必要となり、運用管理パッケージのジョブスケジューラで管理することとした。

現行のメインフレームの運用に合わせてオープン系システムでも毎日計画停止を行うことを前提とする。朝にオンラインを開局し夜間にオンラインの閉局を行い夜間バッチとDBバックアップおよびログのローテーションを行う。DBのバックアップは毎日の計画停止時間にDBを止めてオフラインで行う。バックアップも運用管理パッケージを用い管理する。

3. アーキテクチャ設計と製品選択

個々の要求事項とそれから導かれる基本方針同士をすり合わせ、最も実現しやすく運用しやすい構成を設計する。移行設計のフローを図１に示すが、２章で検討した要件・基本方針を入力とし、製品の選定や構成を出力としている。また、設計のフローは、プラットフォームの選定、論理クラスタ構成、物理クラスタ構成、共有ストレージおよび保守環境の順に展開した。以下では、この設計フローに従い、設計時の検討事項や内容について述べる。

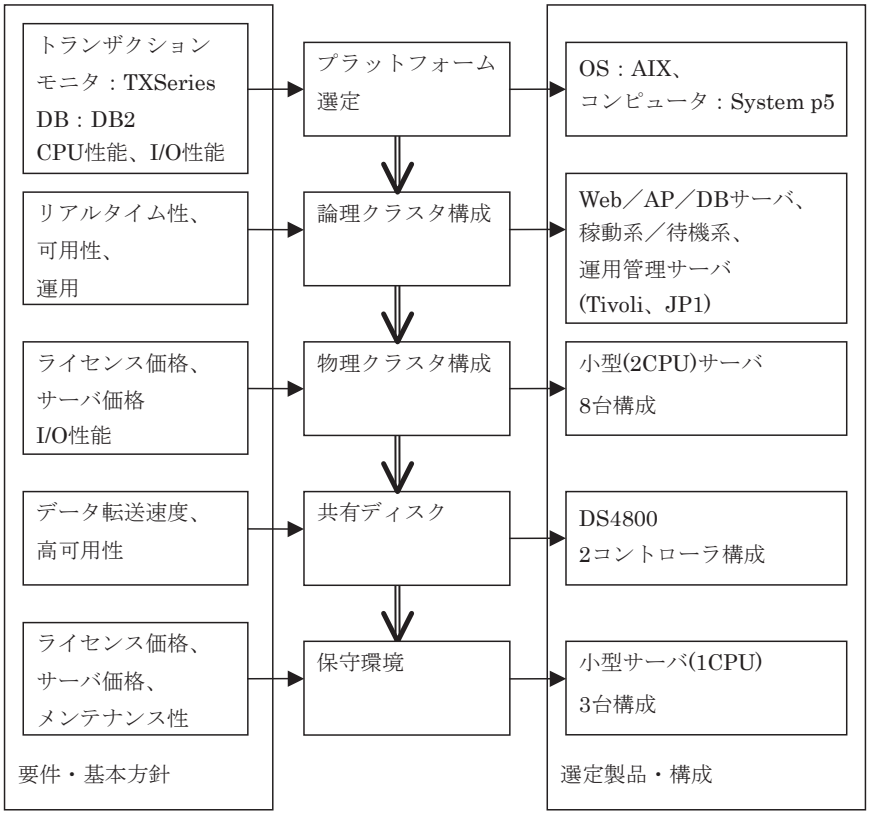

図１　メインフレームからオープン系システムへのシステム基盤の移行設計フロー

3.1. プラットフォームの選定

トランザクションモニタとしてTXSeriesを採用したことから、システム構築時のサポートを考え、同じIBM製品のAIXを選択した。OSをAIXとした場合、ハードウェアはSystem p5となる。ここでは、System p5を対象として、主にCPU性能と構成について検討する。

現行のメインフレームのCPU性能60MIPSに対し、移行プロジェクト開始時点のSystem p5のCPUクロックは1.65GHz以上であり性能的に問題にならないように思える。ただし、メインフレームとオープン系コンピュータのCPUを単純にクロック値で比較するわけにはいかない。CPUに関する検討の留意点について以下に述べる。

(1)　CPUアーキテクチャの差異

CPUアーキテクチャの違いのため、単純にクロック比で性能を評価できない。メインフレームのCPUアーキテクチャはCISC（Complex Instruction Set Computer）で、処理速度が処理内容に影響を受け難く、安定した性能を発揮する。対してオープン系コンピュータで使われているRISC（Reduced Instruction Set Computer）アーキテクチャは処理速度が処理の内容に大きく依存する。

経験上、単純な配列を使った処理などではCISCの１命令はRISCの1.5 から２命令になる。RISCのCPUクロックが1.65GHzであれば、約２分の１の800MIPS見当となり、要求性能の120MIPSは、1CPUで十分満たせると判断した。

(2)　COBOL利用時の問題

pSeriesに不利な点としてCOBOLがある。メインフレームの業務アプリケーションのほとんどはCOBOLを用いている。COBOLでは数値計算はpacked decimalの演算となる。ホストはpacked decimalのハードウェア演算命令を持っているが、System p5は持っておらずエミュレーションで処理している。このため、packed decimalの演算を多用する処理では性能が大きく低下する。演算の頻度を業務アプリケーション担当者に確認し、問題とならないと判断した。

(3)　I/Oの問題

GHzのクロックスピードで命令を実行できるのはデータがCPUのレジスタにある時だけであり、データがディスク上にあれば処理速度はディスクの処理速度となる。メインフレームではディスクおよびネットワークとの入出力をCPUではなくチャネルと呼ばれる接続専用プロセッサが行っている。

一方、AIXではCPUが入出力を直接行っている。しかし、入出力特有の割り込みはRISCアーキテクチャのCPUにとって大きなペナルティとなる。RISCアーキテクチャのCPUはパイプラインを使い同時に複数の命令を処理し高速化を図っている。割り込みを受けると、パイプラインで処理していた命令が無効となってしまう。割り込みにCPUを１個取られても、パイプラインの乱れないCPUが残るように2Way以上のCPU構成とした。

3.2.論理クラスタ構成

リアルタイム性、可用性および運用の要件を満たすようクラスタの論理的構成を設計した。リアルタイム性確保のためミドルウェアごとに独立したサーバを建てるという方針に基づき、Webサーバ、AP(アプリケーション)サーバ、DB(データーベース)サーバ、運用管理サーバを配置し、稼動系/待機系の２系統構成とした。（図２）

3.2.1. リアルタイム性の考慮

必要なミドルウェアとして、WAS 、TXSeriesおよびDB2 UDBがある。ミドルウェアごとに、それぞれ独自のメモリ管理を行っているため、リアルタイム性を維持する単純かつ効果的な方法はサーバを分ける方法となる。そこで、ベースとなるミドルウェア(WAS、TXSeries、DB2 UDB)でサーバを分け、Webサーバ、APサーバ、DBサーバの構成とした。

3.2.2. 可用性

Webサーバ、APサーバ、DBサーバそれぞれを稼動系/待機系の２台構成とし、常時OSを立ち上げておき、障害時サービスを稼動系から待機系に切替えることとした。データの引継は共有ディスクを用いて行う。待機のサーバを１台に集約するN対１構成もあったが、運用作業が単純化できるように、1対1の構成を採用した。高可用性クラスタのパッケージを使用することとし、IBMのHACMP(High Availability Cluster Multiprocessing)を選択した。HACMPは障害時のサービス切替えおよび復旧後の切戻し操作のシェルスクリプトの運用が必要になる。サーバ間の依存関係に伴う操作まではHACMPでは管理できず、DBサーバ(DB2 UDB)がフェイルオーバした場合はAPサーバ(TXSeries)の再立ち上げが必要となる。

サービスを切替えるシェルスクリプトの実行管理のため、ジョブスケジューラとして、弊社で実績のある、JP1(日立の統合システム運用管理)を採用した。

3.2.3. 運用

通常運用のバッチをスケジュールする他に、障害の監視

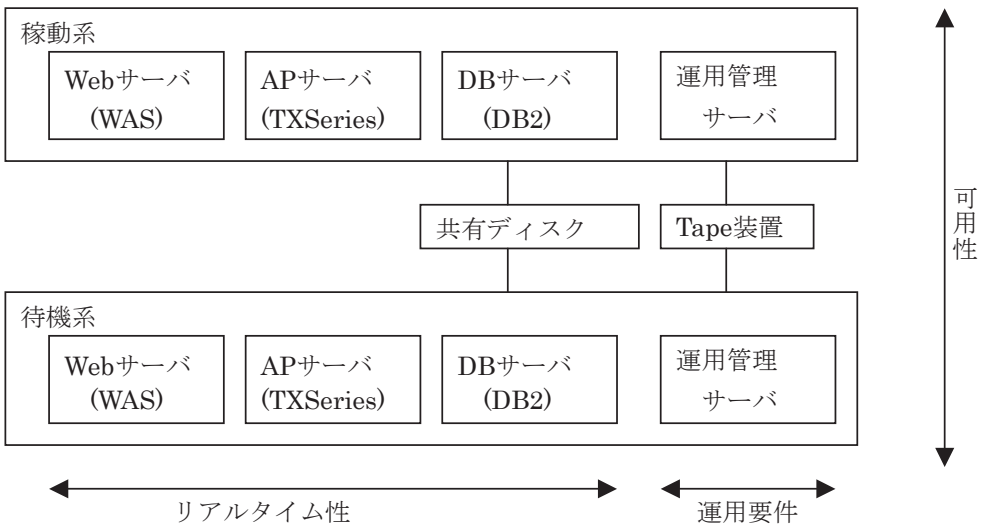

図２　クラスタの論理的構成

および障害時の稼動系から待機系へサービスを切替えるスケジュールも行う。またシステム障害に備えたバックアップの採取もスケジュールを組んで行う。

障害の監視やスケジューラはシステム負荷が大きい場合でも機能する必要がある。システム負荷が大きい場合でも障害を検知しアラームを上げなければならない。リアルタイム性を持った監視を行うため運用管理サーバを独立に立てる。

監視はTivoliの監視モニタを導入した。ミドルウェアによる一点障害を避けるため、実行時間を監視できるものは監視し、異常を監視モニタへ送る方針とした。監視モニタは、異常のレベルに応じ、クラスタのフェイルオーバからアラームまでのアクションを起こす設計とした。しかし、客先要件でフェイルオーバはせず、アラームの出力のみの実装にとどめた。

DBのバックアップは、DBの停止時間を最小にしてネットワーク負荷を抑えるため、一旦ディスクに落としテープに吸い上げる運用とした。DBサーバで共有ディスクにDBのバックアップを取り、運用管理サーバで共有ディスク上のDBバックアップをテープに落とす、サーバをまたがるジョブのスケジュールとなった。

3.3. 物理クラスタ構成

論理クラスタ構成と予算および性能要件より、物理クラスタ構成を決めた。Webサーバ、APサーバ、DBサーバ、運用管理サーバの稼動/待機系論理的クラスタ構成を物理クラスタ構成に展開する。

多数のCPUを実装した２台の大型のサーバを論理的に分割(LPAR)し実装する方法と、８台の小型のサーバで実装する方法がある。IBMのWAS、DB2 UDB、TXSeriesの必要ライセンス数は利用可能なCPUの数で定まるため、どちらの構成であってもミドルウェアのライセンス価格に影響はない。

ハードウェアの値段は、大型機を論理的に分割するより、小型機をネットワーク接続して使う方が安い。一方で、小型機では拡張性が乏しく、性能不足の場合はリプレースになってしまう。ただし、集積回路の集積密度は18～24ヶ月ごとに倍になるというムーアの法則を考慮すると、CPU性能の向上により拡張よりリプレースの方が安価になる可能性が高い。

このような考察から、小型機の構成を採用しWebサーバ、APサーバ、DBサーバ、運用管理サーバの２系統８台構成とした。おのおののCPU構成は、入出力特有の割り込み処理の影響を考慮し、最小の2CPU構成とした。各サーバ間は、業務データ用のネットワークの他に運用管理用のネットワークも設け二重に接続した。それぞれのネットワークはL3スイッチで冗長化した。

3.4. 共有ディスク

現行のメインフレームのディスクI/Oは17Mbyte/secのchannel２本で行っている。これに対し、オープン系コンピュータでは200Mbyte/secのFC(Fiber Channel) ２枚を使うこととした。インタフェースには大分余裕があるように見える。しかしボトルネックで比べるとあまり差はない。オープン系のディスク単体で見ると140Gbyte 、15000回転/分の高速ディスクでも連続転送速度96Mbyte/sec程である。メインフレームのインタフェース転送速度の34Mbyte/secに比べ、オープン系のディスク装置がそれほど速いわけではない。

容量の問題もあり、AIXへの接続をサポートするIBMブランドの共有ディスク装置の内、構成検討時点で最高性能のものを選択した。それでも性能の余裕はCPUに比べ少ない。

一点障害を避けるため共有ディスク本体を除くすべての機器は二重化した。サーバ側インタフェースFCは各２枚とし、接続経路にFCスイッチを入れ多重化した。また、共有ディスク本体も、ディスクコントローラを２台持つ構成として、内部で二重化した。

DBのバックアップのため､共有ディスク上にバックアップ用の論理ディスクを設け、DBサーバと運用管理サーバで接続を切替えて使う方式とした。具体的には、DBサーバがDBのバックアップをファイルとしてバックアップ用論理ディスクに出力する。出力後、バックアップ用論理ディスクを運用管理サーバにつなぎ変え、運用管理サーバがバックアップ用の論理ディスク上のファイルをテープに落とす。

3.5. 保守環境

運用開始後のプログラム開発および機能検証のために保守環境を準備した。当初、待機系をそのまま保守環境として用い、費用を抑えることを計画していた。しかし、待機

系を保守環境とするとミドルウェアのライセンス追加が必要となる。保守環境では、性能、高可用性およびリアルタイム性を確保する必要がなく１CPUでよいこと、１CPUのサーバよりミドルウェアの１CPUライセンスの方が高価であることから別途もう１系統構成した。別立てにするメリットとして稼動系/待機系の障害時の切替え運用が単純になることもある。

保守機は、プログラムの実行環境が稼動機と一致する方がよい。Webサーバ、APサーバおよびDBサーバの３台構成とした。また、検証のための実業務データの取り込みを考慮し、稼動系の共有ディスクにも接続している。

3.6 その他の機器

２章の対外接続で検討し、技術的にはCMTの使用は廃止する方針としたが、実際には他社とのCMTのやり取りが残り、CMT装置の導入を行った。IBMではAIX用CMT装置を扱っていないためサードパーティ製品を導入した。CMTにはLABELと呼ばれる固有の情報があり対応ソフトウェアも合わせて導入した。

4. 設計結果の評価

実稼動したシステムに対し、重要な設計ポイントとした機能要件の業務アプリケーションの移行性と非機能要件の可用性、リアルタイム性および性能の４つの面から評価した結果を以下に示す。いずれの項目についても目的を達成しており､本稿で提示した方針設定や設計方法が的確であったと言うことができる。

(1) 業務アプリケーションの移行性

通信部分の作り込みを除けば、業務プログラムについての部分的手直しで、メインフレーム上のオンラインとバッチの全機能をオープン系システムに移植することができた。なお、本稿では基盤システムの移行を中心に述べてきたが、業務アプリケーションの移行については、機会があれば紙面を改めて報告したい。

(2) 可用性

高可用性の実装については、カットオーバ前の障害試験で問題なくフェイルオーバし、サービスが継続できることを確認した。なお幸いなことにオンライン稼動系のハードウェア障害がまだ発生しておらず、実環境での確認はこれからである。

(3) リアルタイム性

オンライン処理のリアルタイム性が確保できていることを、カットオーバ前のオンライン負荷試験で確認した。想定負荷の10倍となる30電文/秒まで負荷をかけても処理遅れは発生しなかった。

(4) 性能

リスク回避のためにかなり余裕のあるシステム構成としたこともあるが、カットオーバ後のオンライン稼動系の、各サーバのCPU負荷は10%以下に収まっている。時間帯による負荷の変化もあまりなく安定している。しかし、想定した以上に管理サーバの負荷が重く常時10%弱となっているが、要件の範囲内であり問題ではない。また、DBサーバのCPU負荷は５%程度と軽い。

5. おわりに

本稿では、金融系業務システムのダウンサイジングの問題点とシステム構成方針について述べた。そこで得られた重要な知見をまとめると次のようになる。

(1) メインフレームからオープン系システムへの単純移行案件においては、業務アプリケーション実装の基盤となるミドルウェアの選択で、プラットフォーム設計の基本がほぼ定まる。特に、オンライン機能が含まれた場合、この実装に必要なDBとトランザクションモニタの選定がシステム設計の基本を決めることになる。

(2) データ管理システムが複数種類ある場合、その一元化を図ることがオープン系システムでの運用を考慮すると望ましい。例えば、運用にかかわる操作およびフェイルオーバ時の操作の単純化にも役立つ。本事例でも、VSAMを廃止してDB2 UDBに集約した。

(3) プラットフォームが定まると必要とする性能を満足するためのシステム構成がほぼ決まってくる。この時に、オープン系コンピュータで使われているRISCアーキテクチャの処理速度が処理内容に大きく依存する点に注意が必要である。

(4) 高可用性を実現するために、コンピュータ単位で冗長化し、稼動待機の密結合クラスタ構成をとることが推奨される。また、リアルタイム性確保のために、Webサーバ、APサーバ、DBサーバなど機能ごとにサーバを分割し、ミドルウェアの実行環境を分離して相互の干渉を排除することが必要である。

(5) 共有ディスクはシステムのボトルネックとなりやすいので、十分に余裕を持って設計、選定する必要がある。また、システム全体の可用性を保証する最も重要な部分であるため、ハードウェアの価格に占める割合が大きく高価となるが、投資を惜しんではいけない。接続経路と内部コントローラの二重化は必須である。

(6) DBおよびトランザクションモニタとして用いる製品は、運用を決める大きな要素となる。障害対応のためのジョブも含めたすべてをシステムに盛りこむことは難しく、費用対効果を著しく悪くする。人手による運用でカバーする割り切りも必要となる。しかし、メインフレームと異なりオープン系システムではハードウェア障害の発生頻度が高い。人手による運用は誤りを伴うため極力フールプルーフにする必要があり、運用管理ツールの導入が必須となる。

今後の課題として、今回提示した設計方法を改善し、さらに設計精度を向上させ、次のシステム構築に役立てていくことが必要と考えている。また、システムを構成する製品も日進月歩である。継続的に調査および評価を行い、技術動向をとらえ、より良いシステム設計でプロジェクトの成功に寄与して行きたい。
