ES 7000 を使用した流通系基幹業務システム構築

Development of Mission Critical System for Physical Distribution using ES 7000

田　中　　敦

要　約　1990 年代の後半，WindowsNT 4.0 のリリースにより，Windows サーバを使用した業務システムが急速に普及した．しかし，当時は安定性が低いという理由により企業の基幹業務サーバは依然 UNIX が主流であった．こうした中，2000 年に“ Windows 2000 Data Center Server ”と“ UNISYS Enterprise Server ES 7000 ”が登場し，Windows サーバでミッションクリティカルな企業システム構築が現実のものとなった．

筆者の担当顧客では発注系・商品管理系業務といった基幹業務を UNIX サーバに実装しオープン化を実現していたが，導入から 5 年が経過し，老朽化及び性能限界によるサーバ構成の見直しが必要となっていた． 当初 UNIX サーバのグレードアップが検討されていたが，Windows 2000 Data Center Server＋ES 7000 の持つ性能・可用性・拡張性・TCO（Total Cost of Ownership）削減といった点が注目され，ES 7000 の採用が決定した．

本稿では，UNIX サーバからの移行作業を通じ，基幹業務システムをどの様に Windows サーバに実装したか，又，ES 7000 が TCO 削減についてどの様に貢献したかについて明らかにする．

Abstract　Data processing applications running on the windows-based server have quickly widespread with the release of Windows NT 4.0 since mid-1990s. However, at that time, UNIX operating system still had the advantage of the enterprise server in most companies because of poor stability of Windows operating system. Under such a circumstances, Microsoft Windows 2000 Data Center Server and UNISYS Enterprise Server ES 7000 were announced in 2000, and implementation of the mission critical business application of the Windows-based server has became true.

The customer site the author was involved implemented mission-critical business systems such as the order entry and merchandise management systems and operated them on open UNIX-based systems. However, five years have passed since installation of server systems, and the review of server configuration was required at standpoint of the aging and limitation of performance of processing system and the upgrade of UNIX-based server was initially pursued. But the high-performance, high-availability, extensibility, and reduction of TCO(Total Cost of Ownership), which Windows 2000 Data Center Server＋ES 7000 has, drew the customer's attention, and adoption of ES 7000 was determined.

This paper describes how the mission-critical business systems were implemented on the Windows-based server, and how ES 7000 has contributed to reduction of TCO through experience of the system migration from the UNIX based server.

1. は じ め に

企業システムは，汎用機システムによる集中処理と，オープン化による業務別サーバの設置や大規模ネットワークの構築による分散処理を経て成長してきた. その結果，システムの複雑化を招き，コンピュータ管理部門や開発部門に大きな負担を強いる事となった．こうした中，間接的な運用コストを含んだ Total Cost of Ownership（総

合保有コスト：以下 TCO）をいかに削減できるかが，情報システム部門の命題となっている．本稿では UNISYS Enterprise Server ES 7000（以下，ES 7000）及び Windows 2000 Data Center Server（以下，Windows 2000 DCS）を使用し，複数のシステム環境の統合を行った事例を紹介する．

本稿のもう一点の主旨は，UNIX から Windows への業務システムの移行事例である．企業システムの業務サーバの OS（以下，OS）として，UNIX と Windows のどちらを採用すべきかという論争は長い間論議され続けている．1990 年代において，オープンシステムの OS は UNIX が主体であり，Windows NT 4.0 のリリース後も，UNIX はミッションクリティカルな業務のサーバ OS として位置付けられていた．その最大の理由は，OS としての安定性にあった．例えば，Windows NT 4.0 でメモリリークによるアプリケーション停止やハングアップの発生頻度が高いことにより，企業システムの OS として安定している UNIX を使用すべきという選択である．しかしながら，Windows OS は，クライアント PC の爆発的な普及を足がかりとして利用人口が増加したことで，サーバとしての操作性やハードウェア価格で UNIX より優位に立ち，シェアを伸ばしてきた．また，2000 年以降，大規模な Windows サーバが登場して，最大 32 CPU の大規模な環境で稼働する事が可能となり，両 OS の特性やハードウェア性能の格差は狭まってきている．このようにハードウェアの選択肢が広まった事により，総合的なシステム機能やコスト対効果，管理者の負荷といった問題をいかに解決するかという点が OS 選択の重要なポイントとなってきている．

本稿の背景となった顧客事例では，UNIX サーバ 6 台（内一台はホットスタンバイのバックアップ専用サーバ）で業務システムを構築していた．このシステムは，稼働後 5 年が経ち，個々のサーバのハードウェア性能に起因して処理能力不足が表面化してきた．解決策として，UNIX サーバの最新機種へのアップグレードの検討が行われていたが，各サーバの導入時期が異なるため，UNIX やデータベースエンジンとして使用している Oracle の詳細バージョンの違いが問題となっていた．また，全サーバを一度に交換できないため，ローテーションを組んで UNIX サーバをアップグレードしていくという計画では，作業負荷が非常に大きく，実現性に問題があった．

この対応策として，ES 7000 と Windows 2000 DCS への移行を提案したが，顧客としては，UNIX から Windows への業務システム移行の実現性を最も懸念していたので，実際に一つのサブシステムの移行を実験的に実施することにした．また，ES 7000 によるサーバ統合によりどの程度 TCO の削減が可能となるか，Windows 2000 DCS への安定性や ES 7000 の拡張性が確保できるかの懸念事項もあったが，結果として払拭することができた．本稿では，第 2 章で UNIX から Windows 2000 DCS への移行の実験作業で発生した問題の解決策や考慮点を提示し，第 3 章で ES 7000 を使用したサーバ統合をベースにした可用性や拡張性の確保とどのような分野で TCO 削減ができたかを記述する．

2. UNIX から Windows への業務システム移行

本稿の背景となった顧客事例では，複数の UNIX サーバで稼働している業務システムを ES 7000 と Windows 2000 DCS に置き換える上で，システム構成の妥当性や

移行作業工数を明確にする必要があった．また，移行にあたって，ES 7000 上で使用するプロダクトの選定や移行及び改修の方針を決めた上で，業務アプリケーションの具体的な改修作業量を把握する必要がある．このため，事前に一つのサブシステムを使用して実験的に移行を実施した．

この移行実験は，UNIX サーバと ES 7000 で同一処理を実行し，処理効率の比較と作業手順の確立を行うことが目的である．移行実験の準備作業として，移行や改修の方針を決めるために，業務アプリケーションの全ソースプログラムの内容を解析し，非互換項目の調査を実施した．その結果として，移行実験作業の中で，以下の項目を調査検討することになった．

- UNIX 上で業務アプリケーションの実行制御をするシェルコントロールを Windows 上で実現する方法の検討
- UNIX と Windows の C 言語のシンタックスレベルの非互換項目や実行時の非互換項目の洗い出しとその対応方法の検討
- Windows へ移行したアプリケーション実行効率の調査と必要となるハードウェアリソースの確認
- 業務システム全体を移行するための作業ボリュームの把握

また，実際の移行作業と平行して，Windows 2000 DCS についての安定性の評価を実施した．

本章では，業務システムを UNIX から Windows 2000 DCS へ移行するにあたって調査検討を行った上記の項目について記述する．

2.1 UNIX のシェルコントロール代替

UNIX の業務システムは，アプリケーションの実行制御が UNIX のシェルで記述され，移行対象となるシェルの本数が多いため，シェルコントロールの Windows 上での実現方法の選択が作業工数全体に大きな影響を与える事になる．このシェルコントロールの代替機能として，VB（Visual Basic）Script やバッチコマンド等の Windows 上で動作するスクリプトを検討した．しかし，前提としてシステム全体の移行期間が実質 6 ヶ月であったため，作業工数面の負担が大きくなり選択できなかった．この制約は，顧客側のシステム開発凍結期間が 6 ヶ月以上確保できない事に起因している．

結果として，Windows 上でシェルコントロールを実現するために，Microsoft 社の Windows Service for UNIX バージョン 2.0（以下 SFU）を使用する事にした．SFU を導入する事で，Windows 2000 上で基本的な UNIX シェルコマンドが実行可能となる．表 1 に，SFU を導入する事で使用可能となる UNIX シェルコマンドを示す．また，SFU が対応している UNIX シェルコマンドでは機能的に不足する場合は，SFU の拡張版である MKS 社の MKS Toolkit を使用することで対応できる．MKS Toolkit を導入する事で対応可能となるコマンドを表 2 で示す．この他に UNIX の MOTIF 対応の GUI アプリケーションを移植が必要であれば, 同じく MKS 社の Nuts Cracker を使用することで対応できる．

SFU の導入により，シェルコントロールの移行工数は最低限に抑えられる．しかし，SFU を利用しても UNIX と Windows との非互換として以下のような項目があり，シェルコマンドの記述の変更や解決策が必要となる．

表 1　Service for UNIX の対応 UNIX コマンド

alias	autodfs	autotune						
basename	bgjob	break						
cat	cd	chmod	chown	:（コロン）	continue	cp	cron	crontab
cut								
date	diff	dirname	dos2unix	.（ドット）	du			
echo	egrep	environ	eval	exec	exit	export		
false	fc	fgrep	find	getopts	grep	gwshare		
hist	head							
iconv								
jobs								
kill								
let	ln	ls						
mapadmin	mkdir	more	mount	mv				
nfsadmin	nfsshare	nfsstat	nice	nis2ad	nisadmin	nismap		
od								
paste	perl	print	printenv	printf	ps	pwd		
rcmd	read	readonly	renice	return	rm	rmdir		
rpcinfo	rshsvc							
sdiff	sed	set	sh	shift	showmount	shpc		
sleep	sort	split	strings	su				
tail	tar	tee	telnet	test	time	times		
tnadmin	top	touch	tr	trap	true	typeset		
umask	umount	unalias	uname	uniq	unix2dos	unset	uudecode	uuencode
vi								
wait	wc	whence	which					
xargs								
ypcat	ypclear	ypmatch	yppush					

表 2　MKS Toolkit で拡張される対応 UNIX コマンド

[	[[							
64decode	64encode							
a2p	appc	ar	asa	ash	assoc	autorun	awk	awkc
banner	bc	bdiff	bindres					
c	cal	calendar	cc	chacl	chgrp	cksum	clear	cmp
col	comm	command	compress	config	cpio	cppstdin	crypt	csplit
ctags	cut							
date	db	dc	dd	dde	deroff	desktop	dev	df
diff	diff3	diffh	dircmp	dirs	dlg	domain	du	
ed	egrep	env	ex	expand	expr			
fgrep	file	filebox	filever	flip	fmt	fold	ftype	functions
gdf	gdir	getconf	getopt	ghist	gps	gres	groupinfo	gset
gvar								
halt	help	history	htdiff	htsplit				
iconv	id	if	integer					
join								
lc	line	logname	look	lsacl				
m4	mailx	make	man	manstrip	mapimail	member	mkscgi	mksdiag
mksinfo	mkszip	msgbox	mt					
nl	nm							
od								
pack	paste	patch	pathchk	pax	pcat	pg	popd	pr
printf	ps	PScript	pushd					
r	red	regexp	registry	rev				
security	select	service	shedit	shortcut	sid	size	sleep	smtpmail
spell	split	start	stat	strerror	strings	strip	stty	su
sum	sync	sysinf						
tar	tb	timezone	tksched	tr	tsort	tty	type	
ugrep	uname	uncname	uncompres	unexpand	unpack	until	url	userinfo
uudecode	uuencode							
VDiff	viw	VPax						
wcopy	web	which	who	windir	wpaste	ws	wstart	xargs
zcat								

1）　ファイル名記述におけるパス表記上の違い

ファイル名を記述する場合，UNIX は全てのパスを/（ルート）を起点とした木構造としているが，Windows はドライブレター（C:，D：等）を起点とした

構造になっている．注意点として，パス区切り文字はSFUを使用した場合，Windows表記の￥ではなく，UNIX表記の/を使用する．

ファイル記述例）

UNIXの場合　　Windowsの場合

/home/oracle　→　c：/home/oracle

2）シェルコマンドの応答結果に表示されない情報がある

シェルコマンドの中には，応答結果としてUNIXの場合に戻されていた情報が，WindowsのAPI（Application Program Interface）でサポートしていないために戻されないものがある．下記の例は，UNIXサーバとSFUを実装したWindowsサーバでOracleのsqlplusを実行し，その時のプロセスをps-ef（プロセス状態表示）コマンドで実行した時の出力結果である．

例）ps-ef　コマンドの処理実行結果

UNIXの場合

oracle 5576 5565 0 12：03：36 pts/4　　0：00 sqlplus scott/tiger

Windows上のSFU場合

--coopm　1208　-- 8 -77 e 69264　6260　- -　0：00 sqlplus

この例では，SFUの場合sqlplusがプロセスとして稼働している事は取得できるが，UNIXの応答のようにsqlplusのプロセスに対する詳細パラメータ情報（上記例ではユーザscott/tigerでOracleに接続している）が戻されない．これはUNIXとWindowsのAPI仕様の違いであり，Windowsではこの様な付加情報を取得する手段自体がない事に起因している．このため，代替策をアプリケーションレベルで解決する必要があった．

こういったプロセス情報を取得する必要があるコマンドの代替策として，専用のバッチコマンドファイル（複数のコマンドを一括して実行できるファイル）を作成した．sqlplus［ユーザ名］．batという様に，バッチコマンド自体にプロセス情報が特定できる名前を付けた．

このバッチコマンドから実際のsqlplusを起動する事により，どのユーザがOracleに接続しているかの判断が可能となる．ftp等，ユーザ接続情報を判断する必要があるコマンドは全てこの対応を実施し，対象プロセスの特定ができる様にした．

3）特定の2バイト文字が正しく処理されないコマンドがある．

文字列定数をechoコマンドで表示させた際，特定の2バイト文字の表示でエラーが発生した．下位1バイトがシングルコーテーション「'」と同じになる文字処理が正しく処理されないため，文字列定数区切り文字をダブルコーテーション「”」に変更する事により解決した．

2.2　C言語非互換項目対応

業務システムをWindowsに移行するにあたって，アプリケーションの一部がUNIXのANSI-CのC言語で作成されているため，WindowsのVisual Cとの非互換に対しての調査，検討を行った．非互換項目としては，コンパイルエラーの文法上の問題と実行結果の問題に分けられる．コンパイル時のエラーについては，表3に提示

してあるエラーメッセージを解析することで対応でき，機械的に修正すればよい内容であった．また，Windows 上での実行時の動作結果として以下のような不具合が判明した．

1) 変数の初期化を明示的に宣言しなかった場合，UNIX の ANSI-C では暗黙的にクリアされるが，Windows の Visual C ではクリアされないので，メモリ空間に残存した内容がそのまま使用され，処理結果が一定しない現象があった．

2) サイズの一致しない変数間の項目移送により，結果項目の内容が破壊される．特に漢字項目の項目移送で有効データ長以降に半角・全角スペースの充填ミスが発生した．

以上の様な文法エラーとならない非互換項目については，実際に動作検証するまで問題に気が付かないケースが多い．こうした問題点を完全にクリーンアップするためには，Programming Research Ltd 社の QAC/QAC＋＋等の文法解析ツールの使用が有効である．QAC を使用する事により，コンパイルエラーにならずに実行時エラーとなる箇所を特定できる．但し，詳細解析を実施するためには QAC の実行と結果の解析に期間を必要とする．今回の事例では作業期間の制限があったため，事前調査内容の検証目的で実験的使用に留めた．

表 3　C プログラムの主な非互換項目

主な非互換項目(C言語)	内容
カーソル未宣言	文法エラー
識別子未定義	文法エラー
;抜け	文法エラー
PL/SQLﾌﾟﾛｼｰｼﾞｬ不明	文法エラー
#include絶対パス指定	文法エラー
識別子不明	文法エラー
不正なスペース	文法エラー
;抜け	文法エラー
dllimport関数を定義した(itoa)	文法エラー
ｲﾝｸﾙｰﾄﾞﾌｧｲﾙ不明(ｺﾒﾝﾄｱｳﾄ可)	文法エラー
ｲﾝｸﾙｰﾄﾞﾌｧｲﾙ不明(対処不明)	文法エラー
C標準ﾍｯﾀﾞﾌｧｲﾙのｲﾝｸﾙｰﾄﾞ不足	文法ワーニング
extern宣言なし	文法ワーニング
プロトタイプ宣言なし	文法ワーニング
変数未使用	文法ワーニング
ラベル未使用	文法ワーニング
代入、比較における型の不整合	文法ワーニング
間接参照のﾚﾍﾞﾙ不一致(NULLの違い)	文法ワーニング
値が割りあてられていない変数の参照	文法ワーニング
戻り値のない関数/値を返さない関数	文法ワーニング
値を返さないｺﾝﾄﾛｰﾙﾊﾟｽ	文法ワーニング
ローカル変数のアドレスを返している	文法ワーニング
関数宣言と定義の不一致(itoa)	文法ワーニング
引数が異なる関数の再宣言(itoa)	文法ワーニング
マクロ再定義	文法ワーニング
プログラム上の作用なし	文法ワーニング
strcatの引数が多すぎる	文法ワーニング
PROCソースをｲﾝｸﾙｰﾄﾞしている	文法ワーニング
認識できないｴｽｹｰﾌﾟｼｰｹﾝｽ	文法ワーニング

QAC での C 言語ソース解析結果のサンプルを表 4 に示す．

表 4　QAC での C 言語ソースプログラム解析結果

Description
[C] 'size_t' は同じ有効範囲で 以前の宣言と異なる型を持ちます.
[C] 'memchr' は同じ有効範囲で 以前の宣言と異なる型を持ちます.
[C] 'memcmp' は同じ有効範囲で 以前の宣言と異なる型を持ちます.
[C] 'memcpy' は同じ有効範囲で 以前の宣言と異なる型を持ちます.
[C] 'memset' は同じ有効範囲で 以前の宣言と異なる型を持ちます.
[C] 'wchar_t' は同じ有効範囲で 以前の宣言と異なる型を持ちます.
[C] 'size_t' は同じ有効範囲で 以前の宣言と異なる型を持ちます.

2.3　アプリケーション効率調査と必要ハードウェア・リソースの確認

UNIX と Windows の非互換項目の洗い出し結果を踏まえて，対処すべき作業項目の内容と業務処理結果の正当性を確認するために，一つのサブシステムを実際に移行して評価を行った．また，移行実験を行うサブシステムは，ハードウェア構成の性能を確認するため，業務アプリケーションの中で最も効率要件の厳しい発注サーバの締め処理を対象にした．該当業務は，発注先とのデータ送信の関係上，処理開始から終了まで 30 分以内に完結する必要性があり，移行前の UNIX サーバではリソースが不足していたため，要求の 30 分ぎりぎりの処理時間がかかっていた．ES 7000 に移行して処理速度を実測した結果，表 5 で示す様に約 5 倍の処理速度向上が確認できた．

比較した UNIX サーバはかなり処理能力が低いため，この結果は一般的な UNIX サーバと Windows サーバの性能の比較にはならない．しかし，客観的な比較として，Windows サーバでも高速なバッチ処理が可能であるという点と，UNIX サーバのバージョンアップ提案と比較した際のコストパフォーマンス面での優位性が証明された．これは，提示された UNIX サーバの TPC-C ベンチマーク測定値は ES 7000 の半分だが，価格面では ES 7000 と同程度のためである．また，本番環境と全く同一のデータベース及びトランザクションを使用したため，CPU やメモリ使用量等のハードウェア・リソースの消費量も確認する事ができた．

表 5　UNIX サーバと Windows サーバの処理時間比較

入力件数	出力件数	処理名	UNIXサーバー 処理時間	ES7160 処理時間
82895	46865	発注締め処理（14：00締め分）	0:26:43	0:04:33

2.4　全体移行作業のボリューム把握

移行実験の結果を元に，移行作業全体のボリューム予測を実施した．全ソースの非互換項目の洗い出しは，事前の調査作業で判明していたため，今回のサブシステムによる移行実験で得られた必要作業量を元に，システム全体の作業量を算出した．

作業期間設定は前述の通り 6 ヶ月間であったが，移行実験の作業期間実績が 1 ヶ月未満で終了した結果から，充分であると判断した．

2.5　Windows 2000 Data Center Server の安定性

Windows 2000　DCS についての安定性を検証するため，移行作業を実施している 6 ヶ月の間に ES 7000 と Windows 2000　DCS の評価を実施した．目的は ES 7000 と Windows 2000 を安定稼働させるために必要な設定内容の確認である．主管部との共

同作業で各プロダクトを実装し，動作確認を実施していった．結果として，ES 7000 及び Windows 2000　DCS を安定稼働させるためには，実装するプロダクトについて Windows 2000 DCS での動作保証がされている事が前提となる．下記は実装したプロダクトの一覧である．

・Oracle 8 i（8.1.6）Enterprise Edition
・Oracle fail Safe
・Microsoft Windows Service for UNIX
・Veritas Backup Exec

これらのプロダクトは米国 UNISYS 及び日本ユニシス（以下，当社）で Windows 2000　DCS 上で動作が認められたプロダクトである．これ以外のプロダクト（プリンタ・ドライバ等も含む)は, 一切 ES 7000 上に搭載していない. 今日現在では Windows 2000　DCS 上で動作確認が取れているプロダクトは着々と増加している．現状で動作保証されている他のプロダクトについては Windows 2000　DCS を販売している各ハードウェアベンダーに確認が必要である. また, プロダクトによっては Windows 2000 DCS で実行可能ではあるが，制限事項が付加されている可能性がある．このため，巻末に添付した参考文献等の確認が必要となる．

2. 6　UNIX からの移行についてのまとめ

前項までの内容を元に UNIX から Windows サーバに対して実際の業務システム移行を実現したが，予想より非互換項目は少なかった．その主たる要因は，UNIX サーバと ES 7000 の双方でデータベースに Oracle を使用した事である．移行プロセスにおけるデータの整合性維持は困難な課題であるが，今回は UNIX サーバと ES 7000 の Oracle データベース間で Snapshot 機能を使用して対応した．Snapshot 機能とは，異なる Oracle データベース間でテーブルの変更内容（insert・update・delete 等の変更要素）を複製する機能である．Snapshot の使用により，UNIX サーバと ES 7000 の Oracle データベースの内容を完全に一致させる事ができた．このため，システムテストと併行稼働期間において，処理結果の比較により移行結果の正当性評価が可能となり，短期間で移行を完了する事ができた．

本稿の事例では，言語やデータベースを変更せずに，非互換をプロダクトで吸収するという移行アプローチを取った．全く別のアプローチとして，SQL　Server 等に代表される Microsoft プロダクトへの全面的変更といった手段もある．実際に今回の事例でも，売上分析系のデータウェアハウスの構築には Oracle から SQL　Server の全面変更を実施している．どちらの手法を取るにしても，使用するプロダクトの特性に合わせ，移行方針を確立する事が重要である．

3.　ES 7000 を使用したサーバ統合

UNIX サーバから ES 7000 への移行に際して, UNIX サーバを統合することができた. 本章では，ES 7000 を使用する事により実現したサーバ統合について記述する．このサーバ統合は，サーバ運用管理を軽減するための物理的な統合と，統合されたサーバを業務毎に分割する方法を併用して実施した．この様な構成を取り，システムの可用性を高めるために ES 7000 の特徴の一つであるパーティショニング機能と Microsoft

Cluster Service（以下 MSCS）を併用した．

3.1 パーティション機能を使用したサーバ統合とシステム分割

UNIX サーバを統合する手段として，ES 7000 上にどの様な形で実装するかについて構成検討を実施した．第 2 章の移行時の結果から，UNIX サーバ 6 台分の業務を ES 7000 に集中させた場合，能力的な問題は無いが，各業務処理が集中する事によりトラブル発生時のリカバリー処理の負荷と復旧手順の複雑性といった運用面の問題が発生する可能性があった．トラブルが発生した時の影響度を最低限にするために，ES 7000 のパーティショニング機能を使用して各業務システムを物理的に分割した．すなわち，物理サーバ毎に Windows 2000 DCS を配置し，別々のアプリケーションを稼働させる事により，一つのパーティションで障害が発生しても他のパーティションの業務は全く影響を受けずに業務処理を続行できる．このようにパーティショニング機能を使用する事で，32 CPU の構成であれば最大八つの物理サーバを一台の ES 7000 に構築可能である．

今回の事例では，システム追加やパーティション構成の拡張に耐えられる構成を求められていたため，最大の 32 CPU 構成を取らず，16 CPU 構成の ES 7000 を 2 台にし，それぞれを四つのパーティションに分割した．また，データ容量の増大や業務システムの追加要件に耐えられる様，パーティションあたりのメモリと HDD（Hard Disk Drive）容量を UNIX サーバの 2 倍にした．

こうして 6 台の UNIX サーバで構築されていたシステムを 2 台の ES 7160 DD（16 CPU 構成の ES 7000）に統合した結果，設置スペースの小面積化，使用電源量・空調機容量といった運用コストを削減する事ができた．システム稼働当時（UNIX からの移行時）の ES 7000 構成を図 1 に示す．この構成図にある論理サーバを括るクラス

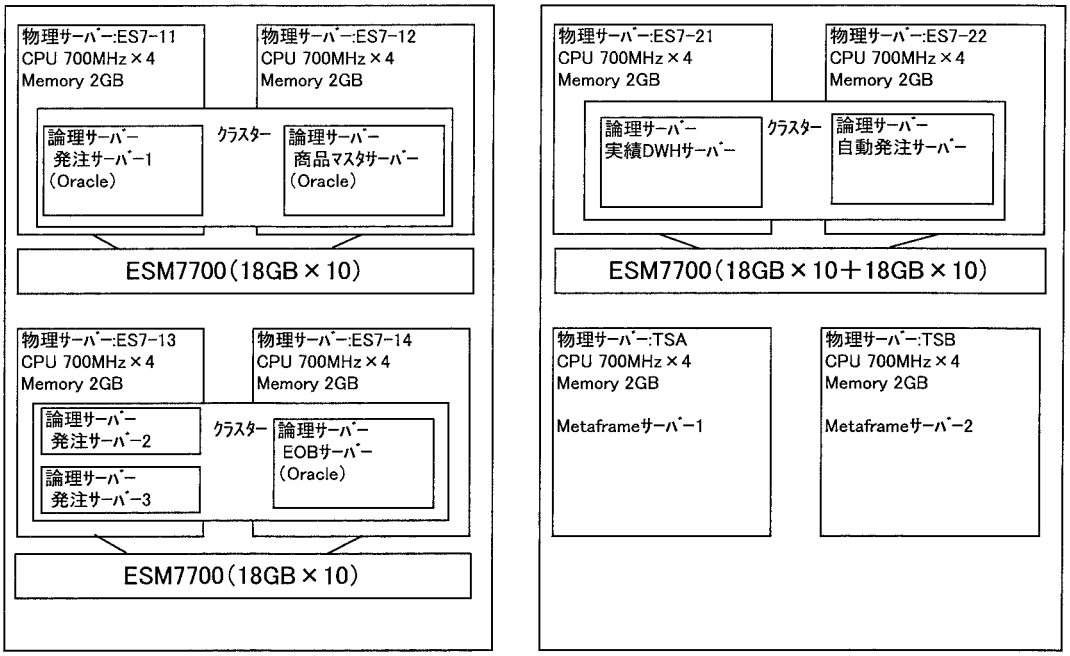

図 1　ES 7000 構成（システム稼働当時）

タの矩形は，MSCS によるクラスタ構成を示している．これについては，事項で記述する．また，各物理サーバは外付けディスクとして ESM 7700 を使用してクラスタの共有ディスクとして使用している．

3.2 MSCS を使用したクラスタ構成

Windows 2000 DCS は，業務システムの可用性を高めるために，OS のサービスとして MSCS を使用してクラスタを構成できる．すなわち，MSCS によるクラスタ構成にすれば，システム障害時にデータベース，ネットワーク ID，共有ディスク等がクラスタ内の別のサーバに移動する（以後フェイルオーバ）ので，そのサーバ上で業務を継続することができる．クラスタリングによるフェイルオーバ時のアプリケーションの動作状況を表 6 に示す．

表 6　クラスタフェイルオーバー時のアプリケーション動作

No.	アプリケーション	フェイルオーバー発生時の動作	フェイルオーバー実行後の動作	備考
1	ORACLEデータベース 使用アプリケーション	・ORACLEがシャットダウン中のエラー発生 ・トランザクションはロールバック	・以後起動されるセッションは通常 ・コミットの発生ポイントによるリカバリーが必要	TNSNAMESの記述は論理サーバー名/IPアドレスにする必要有り
2	SQL Serverデータベース 使用アプリケーション	・接続エラーが発生 ・トランザクションはロールバック	・以後起動されるセッションは通常 ・コミットの発生ポイントによるリカバリーが必要	接続記述は¥¥共有サーバー名¥データベース名にする必要有り
3	クライアントからのftp実施	・接続中のセッションが切断される	・再接続・再実行必要	論理サーバー名／論理IPアドレスでの接続が必要
4	他サーバーへのftp（クライアント）実施	・接続中のまま	・プロセスが無効となる為、別プロセスでの実行が必要	論理サーバー名／論理IPアドレスでの接続が必要
5	クライアントからのtelnet接続	・接続中のセッションが切断される	・再接続・再実行必要	論理サーバー名／論理IPアドレスでの接続が必要
6	EXCELからのODBC接続アプリケーション	・アプリケーション的に再接続されるまでODBC接続エラーが発生する	・再接続が実行される個所まで（メニュー画面等）戻る必要有り	ODBCリソース設定は論理サーバー名の設定が必要
7	ACCESSからのODBCリンク接続アプリケーション	・フェイルオーバー終了までODBC接続エラーが発生する	・フェイルオーバー終了後はそのまま実行が可能	ODBCリソース設定は論理サーバー名の設定が必要
8	ファイル共有	・「リソースが見つからない」エラーとなる	・フェイルオーバー終了後共有可能	リソース設定は倫理サーバー名／論理IPアドレスでの設定が必要

図 1 に示すように ES 7000 の構成では，二つの物理サーバを一つのクラスタとして，三つのクラスタ構成を形成している．各クラスタは，複数の論理サーバが設定されていて，二つの物理サーバに配置されている．例えば，発注サーバー 1 の配置された物理サーバに障害が発生すれば，商品マスタサーバの配置された物理サーバにフェイルオーバして，一つの物理サーバ上で二つの論理サーバが稼働することになる．このため，サーバ障害時の代替用として，予備サーバの設置は不要となった．しかし，MSCS によるクラスタ構成があらゆる種類の障害に対応できる訳では無い．MSCS のクラスタ構成で対応できない障害事象の一例として，データベースの物理的破壊がある．

データベース（Oracle,SQL Server 共）の物理的な破壊が発生した場合には，クラスタ構成であっても共有ディスクが障害を受けているので，フェイルオーバしてもデータベースのサービスが異常終了して，業務システムは継続できない．この様な状態は，クラスタ構成でないデータベースと同じく，バックアップ媒体からのデータのリカバリーが必要となる．HDD の媒体障害の対応として，HDD 構成をパリティ付きストライプセットとミラーリングにして，HDD 側でカバーする事で対応している．

一方，MSCS のクラスタリングを有効に利用するには，ソフトウェアが Cluster API をサポートする必要がある．この API をサポートしていないソフトウェア，つまりクラスタに対応していないソフトウェアを利用する場合には，フェイルオーバ時に処理継続できないことがある．例えば，処理結果をクラスタ構成の共有ディスクに出力

できないソフトウェアや，メモリ上に更新情報を保持して共有ディスクに出力していないソフトウェアは，ファイルオーバすると障害時までの処理がすべて消失することになる．その一例として，タスクスケジュールサービスがある．これは，スケジュール登録した内容がメモリ上に保持されるため，フェイルオーバした場合その内容が継承されないのでスケジュール再登録が必要となる．このため，クラスタリングに対応したジョブ制御機能を持つ運用管理ソフトの JP 1（日立製）等を使用すれば代用可能である．

3.3 サーバ統合による効果

ES 7000 を利用したサーバ統合は，単に 6 台の UNIX サーバを物理的に統合するだけで無く，物理的な構成を組替え可能なパーティション機能と，論理的な構成を相互バックアップするクラスタ構成を組むことで，構成の自由度及び高可用性を実現する事が出来た．さらに，サーバ統合の結果として，バックアップやセキュリティといった面でシステム管理負荷を軽減する事が出来た．例えば，データのバックアップは一台の DLT 装置で採取可能となり，セキュリティは Windows ドメインを利用する事で管理操作性の向上が得られた．この様に，Windows 2000 DCS の MSCS と ES 7000 のパーティショニング機能を組み合わせる事により，単なる物理的なサーバ統合では実現不可能な自由度の高いシステム構成をとることが可能となる．

実際に ES 7000 へ移行後，ES 7000 をアップグレードし，今後の追加システムの開発構想に対応するために，システム構成を図 2 の様に変更した．こうした構成変更を

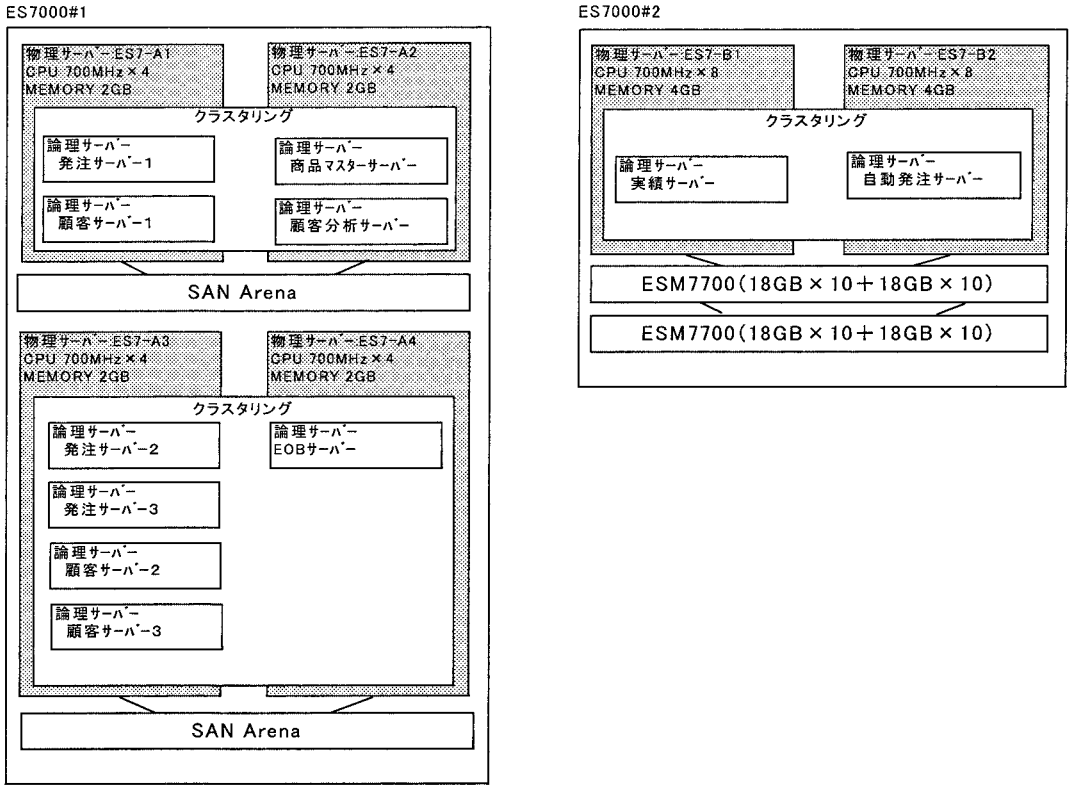

図 2　ES 7000 構成（2002.6 月現在）

実現できるのは ES 7000 の最大の特徴である．今後予定されている動的パーティショニング機能が使用できるようになれば，更に構成の変更やシステム拡張が容易になると予想される．

3.4　TCO の削減効果

本章で提示した，6 台の UNIX サーバから 2 台の ES 7000 に統合したことによる運用コストの削減に加えて，UNIX から Windows 2000 DCS に移行に伴って，以下のような TCO の削減効果を得ることができた．

1）　Windows ドメイン環境に業務サーバを含める事による管理コスト削減

ES 7000 導入までは，単に Windows クライアントの管理にのみ Windows ドメイン環境を使用していた．Windows へ移行後は，業務サーバを含めた統一管理が可能になり，サーバへのログイン管理に始まり，グループポリシーの適用によるセキュリティ管理やリソース管理までが実現可能となった．現在では Windows 2000 での Active Directory による管理を実現している．

2）　Metaframe の使用によるクライアント PC の TCO 削減

今回の移行前までは，サーバサイド・コンピューティングを実装できる Citrix 社の Metaframe を実験的に使用していたが，ES 7000 の導入を契機に業務システムへの適用を行った．このことにより，老朽化したクライアント PC でも快適な速さで動作するようになった．さらに，クライアントのアプリケーションは，サーバ上に配置して集中管理するので，各クライアント PC へ配置していたアプリケーションモジュールの管理コストを削減することができた．また，比較的安価な Windows　CE ベースのクライアントを使用した無線 LAN（WAN）業務の実現も可能となった．

3）　システム開発と保守要員の教育コストの低減

UNIX のシステム管理は，ベンダーシステム保守要員の常駐による 2 名体制で実施していたが，ES 7000 の導入により常駐システム保守体制は不要となった．これはハードウェア障害やフェイルオーバ発生時の自動通知体制環境が整った事に起因している．また，システム開発要員の教育面で，Windows 2000 DCS の操作がクライアント PC で使い慣れた Windows のユーザ・インタフェースに準じているため，基本的操作習得コストが不要であった．システム管理者に対しては Windows 2000 Server の管理用の教育資材が市販されているため，基本的な管理要件は充分習得可能である．

4.　お　わ　り　に

ここまで ES 7000 及び Windows 2000　DCS の長所を記述してきたが，いくつかの課題も残っている．その一つにプロダクトのライフサイクルの問題がある．Windows に限らず，近年のオープン系プロダクトはライフサイクルが非常に短くなっているため，本事例の様にライフサイクルの長い基幹系業務をオープン系プロダクトで構築する場合は，OS やデータベース製品のバージョンアップをすることを前提にその対応方法を充分に検討しておく必要がある．

また，ES 7000 でも障害発生時の原因追求に必要となる資料は，イベントログ，各

プロダクトのログやメモリダンプ等で，従来の Windows と同一である．ES 7000 には，これらを一括採取する機能を持つ ESS（Enterprise Server Software）の Analysis Manager が提供されている．このソフトウェアを利用するには，付帯している VB Script のサンプルコーディングを参考にして，システムの環境応じた VB Script 作成する必要がある．しかし，障害のケースによって収集すべきデータが異なるため，各々のパターンで VB Script を開発して運用に組み込むには，作業負荷が高くなる．この機能を一歩進めて，汎用的に，障害時の資料収集ができるようになることを期待したい．

本稿で紹介した ES 7000 の移行事例では，UNIX からの移行とサーバ統合により，ES 7000 上で TCO 削減を実現する事ができた．筆者にとっても貴重な体験であったが，特記すべきは，顧客での ES 7000 に対する評価が高い点である．Windows 2000 DCS と ES 7000 の導入に対する不安要素は，現在完全に払拭されている．この状態を維持し，ES 7000 でのシステム構築を継続するために，Windows OS 自体の更なる成熟を願う．

最後に，UNIX から ES 7000 の移行作業について，多大なる支援をいただいた顧客システム部，社内関係者各位に謝辞を表し，結びとしたい．

参考文献 ［1］ Microsoft Product Support Service 文書番号 JP 266650「Windows **2000 Data** center Server コンピュータ上での BackOffice プログラムのサポートに関する情報」

執筆者紹介 田　中　　敦（Atsushi Tanaka）
1986 年 3 月仙台電子専門学校情報システム工学科卒業．同 4 月日本ユニシス㈱入社．1988 年より流通小売系アプリケーション開発・保守・サポートに従事．現在東北支店 I & C システム室に所属．