サーバー統合基準の確立と作業アセット化によるTCO削減

加藤 洋　三谷 隆

Establishment of Server Consolidation Standards and Asset Management of Work

Hiroshi Katoh Takashi Mitani

分散コンピューティングの普及によるシステムの大規模化, 複雑化に伴い, 維持管理総経費(TCO)増大という負の側面が顕在化してきている. オンデマンド時代に適応した企業となるためにはサーバー統合によるTCO削減は重要手段になっている. しかし, お客様環境で実際にサーバー統合を実践する際には情報テクノロジーやメソドロジーとともに全体最適の視点が必要である. そこで本論文では, 2点の具体的施策を提案し, その有用性を示す. 1つは「サーバー統合基準の確立」であり, 5つの側面の評価軸を利用し, 重要性や優先度を数値化することでサーバー統合方針を決定する手段を示す. もう1つは「基盤作業のアセット化」であり, 設計書やテスト項目, ガイドやツールなどをアセットとして管理する方法を示す. 2つの施策は全体最適化によるTCO削減に効果があることを, 実践事例の評価で示す.

As systems become larger and more complicated by the spread of distributed computing, increase in TCO has become evident on the negative side. TCO reduction by server consolidation is one of the critical steps that a corporate can take to win in the on-demand era. When practicing server consolidation in the customer environment, it is imperative to apply the viewpoint of overall optimization as well as IT technology and methodology. This paper proposes two specific methods and describes their effectiveness. One method is "Establishment of server consolidation standards." It provides 5 evaluation standards to numerically assess the server consolidation in terms of importance and priority, and helps in setting server consolidation policies. The other method is "Asset management of work". It shows the ways to maintain design documents, test content, guides and tools as assets. These two methods effectively reduces TCO by overall optimization and we present them along with the evaluations of actual cases.

Key Words & Phrases：サーバー統合, TCO削減, リファレンスアーキテクチャー, EA
server consolidation, TCO reduction, reference architecture, enterprise architecture

1. はじめに

1980年代後半から1990年代にかけて手軽さ, 柔軟さ, 低コストといったメリットから分散コンピューティングが幅広く普及し, システムが大規模化, 複雑化するにつれ, 維持管理総経費(TCO)増大という負の側面が顕在化してきている.

お客様企業において, オンデマンド時代に適応するためには, スピード, さまざまな要求や変化に対応できる柔軟性を持ちながらも, コストを最小化していくIT環境にしていなければならない. 短期的には, 新規案件を凍結することでITコストを抑制することも可能であるが, TCO削減を実現しない限りITコストの削減, 企業競争力の強化はかなわない. 確実なTCO削減のためにはサーバー統合は重要な手段となっている.

サーバー統合によりTCO削減効果を得るためには論理分割(LPAR)をはじめとする情報テクノロジー[1]やZodiac[2]などのメソドロジー適用だけでなく, 全体最適の視点を考慮する必要がある. なぜなら, サーバー統合の仕方によっては, H/W, S/WなどのIT資産のコスト削減は出来ても, 運用やメンテナンスにかかる人件費が削減されなかったり, 統合するため

提出日：2004年08月31日　再提出日：2005年8月30日

の一時費用の増加やユーザー要望への柔軟な対応を阻害する危険性があるからである. TCO削減のために, 企業全体の視点でサーバー統合を進める必要がある.

2002年の日経コンピュータ調査[3]によると, IT予算の6〜7割は既存システムの保守・運用によるものである. また, 保守優先は時代の流れである[4]とも言われる. サーバー統合は保守・運用費の削減だけに着眼しがちだが, システム的および人的リソースの有効活用が大命題であり, 新規システム構築時の効率化や保守切れによる機器やソフトウェア更改の効率化なども着眼し, 全社最適化を図る必要がある.

サーバー統合による削減効果が期待される費用としては大きくシステム資源と人件費である. サーバー統合を成功に導き, TCO削減を実現させるには, H/W, S/Wなどのシステム資源コストを削減させる視点, システム運用・メンテナンスなどの基盤系作業にかかる人件費コストを削減させる視点の2つの視点で考える必要がある.

本論文では, この2つの視点に対する施策として,「サーバー統合基準の確立」と「基盤作業のアセット化」を提案する. 以下, 2章でサーバー統合における課題を整理し, 3章4章で具体的な2つの施策について述べる. また実践事例と評価結果を5章で述べる.

2. サーバー統合の必要性と課題

2.1 サーバー統合に向けた課題

サーバーを統合するにあたり, 各企業はさまざまな課題を持っている. 筆者らの経験から各企業がサーバー統合を行う上で抱えている典型的な課題を示す.

(1) サーバー統合といってもアプリケーションやDBまで統合するケースやLPARなどのH/Wだけを統合するケースなど統合レベルが複数あり, 担当者では統合レベルの妥当性を決めるのが困難.
(2) 統合後の運用, メンテナンスコストが増大してしまう危険性がある. 統合後も複数アプリケーションシステムの運用, メンテナンスを強いられ, 結果としてコストが削減されない.
(3) サーバー統合に基準がなく始めてしまうと, 企業全体の視点でサーバー統合に関してトレースできなくなってしまう. 企業全体で最適化されない.

上記のような課題を解決し, 企業全体の視点で, サーバー統合を進めTCO削減を実現するためには, サーバー統合に対する統合基準の確立をすることが必要である.

2.2 システム資源削減における課題

統合の実施にあたっては, システム資源削減効果だけではなく統合によるデメリットやリスクが存在しないかといった点も重要なポイントとなる.

(1) 統合の安全性

サーバーを集約化することによりシステム間の独立性は制約されることとなる. 具体的にはサービス時間の違いによるH/Wメンテナンス時間の減少, 将来的な拡張性を損なう, 障害時の影響範囲拡大といった点を考慮する必要がある.

(2) 企業全体での最適化

プロジェクト最適化だけの視点で判断されていないか, 企業内の他システムとの重複間や無駄がないかといったエンタープライズ・アーキテクチャー(EA)[5]に代表されるような企業全体としての評価が必要である.

(3) 削減効果算定の妥当性

メンテナンスワークロードが減少することだけに着眼していないか, 統合構築するワークロードを加味しても効果があるのか.

これら視点による統合の妥当性についての項目はシステム毎に重要性や優先度が違い, また割り切りが許されるシステムとそうでないシステムなどさまざまである. 重要な点はトレーサビリティであり, 確立された「統合基準」で指し示す必要がある. 割り切り事項や拡張性をどこまで見込んだかなどを明示しておかないと, 将来の拡張時に制約が発生した場合や障害による影響が出た場合に問題となるといった可能性があるためである.

2.3 基盤作業(人件費)における課題

お客様企業においてサーバー統合する場合, H/W, S/Wなどのシステム資源コストだけでなく人件費を削減しなければ真のコスト削減につながらない. 筆者らの経験から人件費, 特に基盤作業に関する典型的な課題を以下に示す.

(1) 企業内においてサーバー統合を複数・平行的に行う場合, 重複した作業が発生し統合ワークロードがかかってしまう. WBS(Work Breakdown Structure), 設計, テストなど統合に関する重複する作業ワークロードが統合毎に発生してしまう.
(2) 統合作業時より, 統合後の運用, メンテナンス作業を意識しないと統合後の人件費削減につながらない.

上記の課題を解消し, 人件費を削減しつつ安定したシステムを提供するためには「基盤作業のアセット化」としてアーキテクチャーのリファレンス(RA)[6][7]およびWBS, 設計書, 運用手順書, テストケース, CFIA (Component Failure Impact Analysis)などの作業のアセット化が必要である.

3.サーバー統合基準の確立

サーバー統合基準を確立するためには, システム特性による重要性・優先度だけではなく, そのシステムに係わる複数のステークホルダーである業務開発部門, 基盤構築部門, 運用部門, 実際の投資を行うエンドユーザー部門といった利害関係の異なる関係者間での合意が重要となる. そして, 各々の部門が重要視する項目をすべて包含した比較軸の明確化と, それぞれの比較軸のプライオリティ付けによる評価の数値化こそが重要なポイントとなる.

3.1 統合比較軸の5つの側面

比較軸については各ステークホルダーを意識し5つの側面を定めた. 以下では具体的な比較軸と解説をバンキングにおけるWebアプリケーションサーバーの場合において, 既存システムへ統合するか否かといった視点で検討した際の例をベースに述べる. また, 本節では紙面の都合上Webアプリケーションサーバーについてのみ述べるが, DBサーバーなどについてもアーキテクチャー側面の内容が一部変更になるのみであり, 考え方や比較軸については適用可能であることを実践で立証している.

3.1.1 アーキテクチャー側面

主に基盤構築部門が考慮する側面でありサーバー特性により比較軸は異なる. Webアプリケーションにおいては**図1**のように, JVMまで同一にするか, JVM分割する場合には筐体まで分けるかといった選択肢での比較となる.

(1) セッション共有の必要性

セッション情報を共有したい場合には同一JVMに限られる. 他の方法をとる場合には, 引数による受け渡しなどが考えられる. トップメニュー間のリンク程度で良い場合には, どの案もとりうる.

(2) JVMスレッド数制御(流量制御)

JVM毎に同時稼動スレッド数を設定可能. 業務毎に同時実行数を制限したい場合, それぞれの高負荷時にも業務を優先させたいなどの場合にはJVM分割に限られる.

(3) JVMヒープサイズ

JVM毎に割り当てられるヒープ領域は, JVMの上限やガベージ・コレクション実行時間から実質1.5GB程度が上限となる. 同時ログイン数とアプリケーションが使用するヒープ量から, 同一JVMにのせることが可能かアセスが必要.

(4) 他システム連動経路

ホストや他システム連動が必要な場合には, 連動経路によっては1サーバーでの設定経路数に上限があるものもある. また経路を利用形態により分割し独立性を高めたい業務などの場合には筐体分離が望ましい.

(5) 採用ソフトウェア

ソフトウェア構成が同一であれば問題ないが, パッケージ活用などの場合には別サーバーといった競合回避が望ましいケースがある.

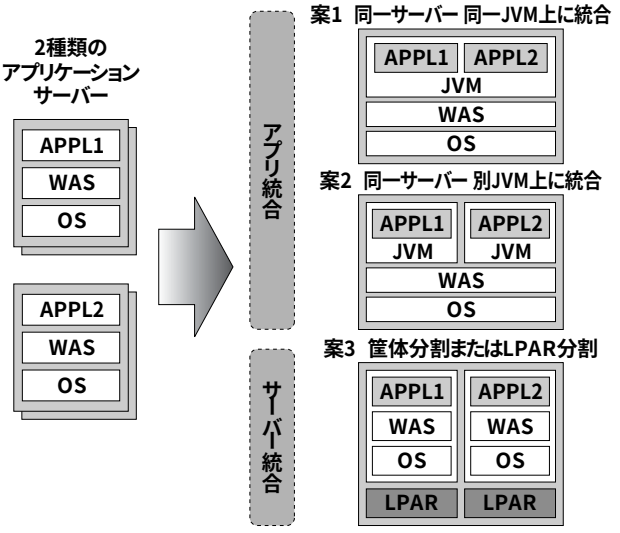

図1. Webアプリケーション統合パターン

3.1.2 業務特性側面

主にエンドユーザー部門と基盤構築部門間で考慮される側面である.

(1) CPU占有

スレッド単位でのCPU割り当て制御は不可. JVM単位でのCPU割り当て制御も確実な分割とはならないため, 他業務によるCPU浪費などを考慮しCPU単位での独立性を維持したい場合には筐体分割が望ましい.

(2) 利用ユーザー

双方のアプリケーションが同一利用者の場合には, ピークが同一であっても完全に加算されるわけではない.

(3) ピーク時間

ピーク日・時間が異なる業務の場合には, リソース有効活用のために統合が理想.

(4) ピーク処理能力

ピーク処理に必要な能力の合計を正しく把握する. 既存サーバーへ統合する場合には, 現行機器でまかなえるのか否かや, 能力増強可能なのかといった視点が必要.

(5) サービス時間

サービス時間の著しく異なる場合(計画停止時間帯など)は, JVM分割案となる. 影響度はシステムの開閉局の仕組みにも依存.

(6) 拡張性
トランザクション量の伸びが大きいと想定される場合には, JVM単位で増やせるJVM分割案が良い.

3.1.3 システム運用側面

主に運用部門が考慮する側面である.
(1) 運用負荷
統合しJVMも同一であればシステム運用負荷は増加しない. JVM分割では既存システムの運用変更, 筐体分割では新規システム運用が追加となる. LPAR追加の場合にも筐体分割と同等負荷が増加することが一般的.
(2) ランニングコスト
差異がどの程度あるか算定し配点すべき.
(3) 監視システム
業務のドメインにより監視システムが異なる場合など, どの監視下にするべきなのかによってはサーバー分割となる.

3.1.4 プロジェクトリスク側面

プロジェクト推進面より考慮される側面である.
(1) テスト環境
本番機を繰り回し後にリリースするといったやりかたが必要なシステムではサーバー分割となる. キャパシティパフォーマンスや運用検証までは不要あるいは机上検証で可能である場合には統合による問題なし.
(2) 業務開発担当チーム
組織的な側面であり, 双方のシステムが同一開発チームである場合には, プログラム・ライブラリ管理を含め一元化可能であり統合が理想. 別チームによりサーバー統合をする場合には, 各種ネーミングルールや標準化などの密な調整が必要であり, 結果として開発生産性を下げる可能性もある点を考慮すべき.
(3) 双方業務の開発状況
相手業務側が安定稼動の状態なのか, 並行開発となるかにより状況が異なる. 並行開発となる場合には統合リスクが高くなる.
(4) 移行リスク
既存JVMへの統合の場合には単純な業務リリースと同様となるためリスクは少ない. また別筐体の場合にも独立性が高くリスクは少ないが, 既存機器へのJVM追加の場合には本番機での移行作業が多く発生しリスクが高まる.

3.1.5 投資側面

(1) エンドユーザー部門
組織的な側面であり, 能力増強などの際のオーナー部門が同一か否か. 別の場合には共有資源への増設などの場合の負担方法が課題であり協議が必要.
(2) 初期コスト
各案で差異がどの程度でるかにより配点.

3.2 統合基準比較軸のプライオリティ付け

通常比較検討する際には各項目と案のマトリックスにおいて簡易的に○△×といった評価を行うが, サーバー統合においては関係部門による重要性・優先度の考え方が異なるため配点方式とすべきである. 比較軸それぞれにおいて各案の評価として○は2点, △は1点, ×は0点と配点とし, 各比較軸の重要性・優先度をプライオリティとして0〜3として影響度高:3点, 影響度中:2点, 影響度低:1点, 該当しない:0点とした. これにより各案の評価(0〜2点)×比較軸のプライオリティ(0〜3点)により各案の合計評価得点が出るようにし, マトリックス表を作成する.
このマトリックス表を利用し各関連部門とすり合わせを実施することで部門間の利害の違いをすり合わせることが可能であり, 全体としてのプライオリティ付けを合意できるとともに, 統合しないという結論になった場合にも新規サーバー構築の追加投資を何のためにねんしゅつするのか, どの評価軸のためにいくらぐらいの投資をすることとなるのか判断が可能となる.

4.基盤作業のアセット化

サーバー統合に関する作業, 統合後の運用, メンテナンス作業における人件費の削減のためには, リファレンスアーキテクチャー(RA)の考え方を拡張して, 再利用を徹底することが重要である.

4.1 基盤S/W,H/Wのマスター化

サーバー統合を行う場合, 基盤部分のマスター化を行う. **図2**のようにOS, 主要ミドルウェアの組み合わせを基本マスターとする. Web系アプリケーションサーバーであれば, AIX®+WebSphere®+Webフレームワークなどを基本マスターとする. これに, オプションマスター

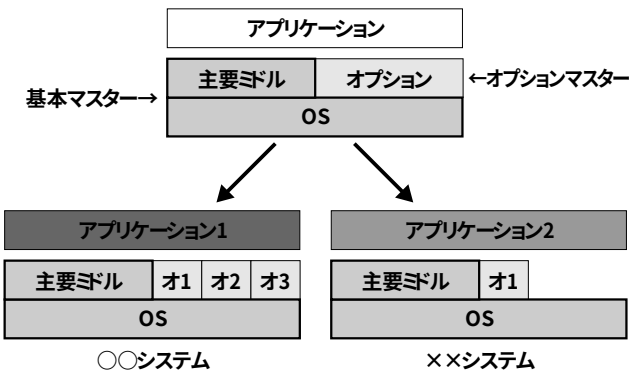

図2. 基盤マスター化の考え方

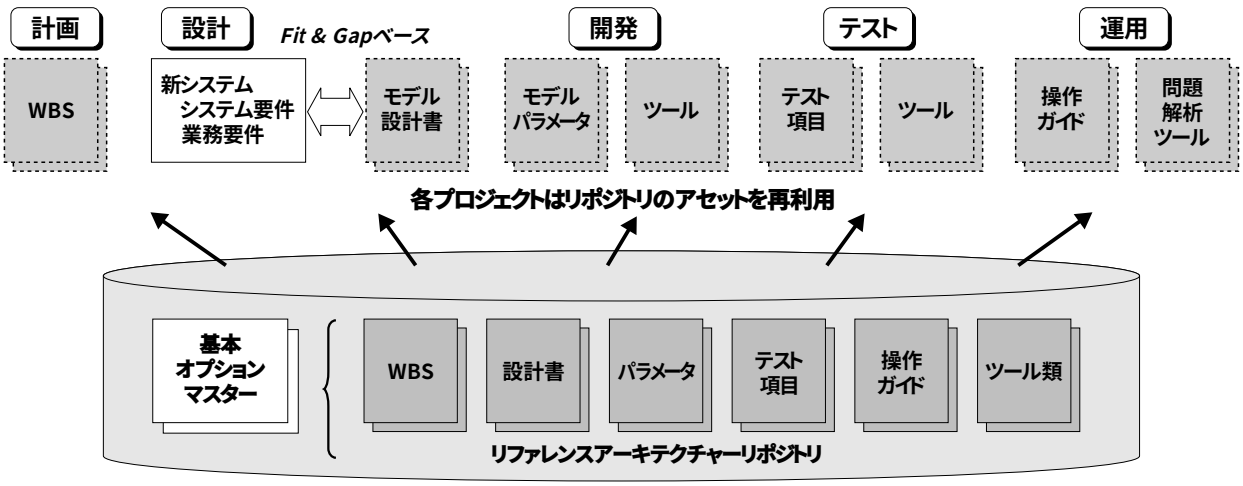

図3.　基盤系作業のアセット化

として, ファイル転送S/Wやレガシーシステムとの接続S/Wなどを追加導入する. 実績のある基本マスターを再利用することで, 個別プロジェクト毎に設計, テストするよりも低コストで高品質な基盤を構築できる.

4.2　基盤作業のアセット化

実績のあるアーキテクチャーの設計および基盤作業をアセット化し再利用できる仕組みを構築する. アーキテクチャー毎にアセットとしてWBS, 設計書, パラメータ, テスト項目, ガイド, ツール類などの全開発工程の成果物をリポジトリに格納する. (**図3**) 各プロジェクトでは, そのリポジトリからリファレンスしたいアーキテクチャー, 例えばWebアプリケーションサーバーやDBサーバーなど, を選択し, 基盤系の一連作業のアセットを取り出す. そのアセットと新システムの要件のFit&Gap分析を行い, Gapを中心に設計, テストを行う. これにより, 各プロジェクトでの作業コストを軽減できるとともに, 品質の高い基盤を構築できる. この施策をサーバー統合時に適用することにより, 統合時の作業負荷軽減だけでなく, 統合後の運用, メンテナンス作業負荷も軽減できる. また, 企業全体でアーキテクチャーの整合性が取れ, ビジネスの変化に柔軟に対応できるサーバー統合を実現できる.

5．施策に基づくサーバー統合の実践と評価

前説で述べた施策をもとに, 実際に筆者が担当するお客様において実践した事例を紹介するとともに, その効果について以下に述べる.

5.1　「サーバー統合基準」の実践と評価

イントラネットの統合事例と, お客様向けインターネットサービスの非統合事例の2事例を紹介する.

5.1.1　イントラネット業務事例

営業活動を支援するSFAアプリケーションの新規構築案件において, プロジェクト計画フェーズでのEAレビューがサーバー統合検討のスタートであった. EAレビューでは保有データや利用ユーザーの重複からすでに稼動している融資稟議業務のサーバーを有効活用すべきではとの案が指摘された. 開発部門としては既存業務への影響や開発部門が異なるといった点で同一サーバーの活用には後ろ向きであり, コスト負担をするユーザー部門も異なるために独立したサーバーを設置する方向で検討していた. 運用部門としては分散系サーバーがすでに乱立していることもあり, サーバー追加には難色を示していた. そこで前述のサーバー統合判断基準のマトリックス表を作成し, 評価を行った. その結果, 以下のような総合点数となった.

案1：既存JVMへの業務追加　22点
案2：既存機器へのJVM追加　19点
案3：新規サーバーでの構築　11点

部門間でのすり合わせの結果, 案1の既存JVMへの業務追加が選択され, 結果としてコスト削減となった.

5.1.2　お客様向けインターネット業務事例

お客様へ外為サービスを提供するバンキングアプリケーションの新規構築案件において, 機器構成検討時点でサーバー統合の検討がされた.

開発部門としてはすでに提供している預金系サービスとはサービス時間も異なり, 1取引のレスポンスタイムにも差があることから独立したサーバー構成とし障害時の共倒れを防ぎたいと考えていた. しかし基盤構築部門は同一サーバーにより構築負荷を軽減したいと考えており, ユーザー部門も投資を圧縮したいと思っていた. 利害関係が対立しているに加え

て, 双方の言い分も定性的なものであるため議論は進まず, 前述のサーバー統合判断基準のマトリックス表を作成し, 評価を行うこととなった. 結果としては以下のような結果となった.

案1：既存JVMへの業務追加　24点
案2：既存機器へのJVM追加　35点
案3：新規サーバーでの構築　52点

サーバー統合時に比較してコスト増となったが, 何のための対価として投資するのかが明確になったことで, 各部門は新規サーバー構築とすることで合意できた.

5.1.3 「サーバー統合基準」の評価

これまでは○×による比較表と文書による総合評価による判断であったために, 何をだれがどのように重視した結果であるのか, 何を犠牲あるいは重要視してコスト増あるいは削減となったのか不明朗であった. それは関係部署間の調整の場も同様で, 空中戦になりがちであった. しかし, 確立された統合基準をもとに各案を数値化することで「そこまで重要視する必要はない」とか「業務特性がどの程度違うのか」といった部署間でのすり合わせもしやすく, 結果としてどの案でいくべきかといった判断にも不公平感がなく有効であった. 開発部門としては経営への報告に際しても有効に機能したとの意見もいただいた.

5.2 「基盤作業のアセット化」の実践と評価

基盤作業のアセット活用の事例としては, 営業店向けCRMアプリケーション構築時の基盤設計・WBS・障害分析・運用機能などのアセットを再利用した事例を紹介する.

5.2.1 イントラネット業務構築の事例

経費管理業務の更改プロジェクトが立ち上がり, 前述のCRMアプリケーションと同様に典型的なイントラWebアプリケーションであったため基盤作業のアセット再利用に取り組むこととなった. まずプロジェクト計画フェーズにおいてはWBSを流用し, 作業計画の立案とアセット利用により省略できる作業の洗い出しを行った. 設計フェーズではアセットをベースに差分設計となるため効率化されるだけではなく, 設計事例をベースにお客様とも確認をしていくため漏れが発生することも防げた. 構築・テストフェーズにおいては導入・設定作業は同様に負荷がかかるが, シェルなどの運用機能開発は大幅に削減できた. また, テストケースなどを流用できるため負荷削減になるとともにケースの漏れなども回避できた. 結果として基盤構築負荷の30%強の工数削減となった.

5.2.2 「基盤作業のアセット化」の評価

アセットを活用することで効率化や品質維持に有効に働いたことは当然であるが, 企業内のシステム間の整合性も維持されお客様の満足度も高かった. 運用部門においても要員共通化が可能となり, 開発部門はこれまでプロジェクト毎に基盤保守要員を残していたが, 基盤共通設計により保守要員共通化も実現した.

6. おわりに

サーバー統合において, サーバー統合基準の確立と基盤作業のアセット化がTCO削減に大きく貢献することを実証することができた. サーバー統合基準を確立することで, ステークホルダーをはじめ, 企業全体で統合に対するメリットを理解できる. 最終的に企業全体での最適化がなされ, TCO削減が可能になる.

また, 基盤作業をアセット化することで, 統合及び統合後の運用, メンテナンスに関する人件費の削減を実現することができる. 合わせて実績のあるアーキテクチャーを再利用することにより, システムの安定稼動や企業全体での最適化が実現できる.
2つの施策を実施することで, 統合後のビジネス変化に柔軟に対応するシステム構築が可能となり, オンデマンド時代のニーズに応えるサーバー統合が可能になる.

参考文献

[1] IBM Server consolidation, http://www-1.ibm.com/servers/solutions/serverconsolidation/about/types.html, 2004.06.14
[2] Zodiac, http://www-6.ibm.com/jp/servers/eserver/zodiac, 2005.07.05
[3] 戸川尚樹他,「不良IT資産を洗い出せ」, 日経コンピュータ, no.558, pp.40-54, 2002
[4] 西村崇他,「もうシステムを錆びつかせない」, 日経コンピュータ, no.604, pp.52-64, 2004
[5]「特集：エンタープライズ・アーキテクチャー」, ProVISION No.41, 2004, http://www-6.ibm.com/jp/provision/no41/index.html, 2005.08.10
[6] フランク・ブッシュマン他, ソフトウェアアーキテクチャ, 近代科学社, ISBN 4764902834, 2000
[7] Martin Fowler et al., *Patterns of Enterprise Application Architecture*, Addison-Wesley Pub, Boston, ISBN 0321127420, 2002

日本アイ・ビー・エム株式会社
ICP ITアーキテクト

加藤 洋 Hiroshi Katoh

[プロフィール]
1990年に日本IBM入社．入社以来，銀行担当のSEとして，分散系，Web系を中心に数多くのプロジェクトに参画．現在は，都市銀行のお客様担当のクライアントITアーキテクト（CITA）として銀行全体のアーキテクチャー構築に従事．
hiroshik@jp.ibm.com

日本アイ・ビー・エム株式会社
ICP ITアーキテクト

三谷 隆 Takashi Mitani

[プロフィール]
1991年に日本IBM入社．製造業・流通業・金融機関など多彩なお客様の担当SEを経験．1999年より都市銀行のお客様担当SEとして分散系，Web系を中心とした各種プロジェクトに携わり，現在はクライアントITアーキテクト（CITA）としてアーキテクチャー構築全般に従事．
mitaka@jp.ibm.com