ＩＴ特別講座
システム運用の切札、
Linuxの仮想化入門

～今、現場で求められている技術とは～

LA-Linux専任講師 矢越昭仁

2011/04/09

インターネットが普及することで、膨大な数のサーバが利用される環境となりましたが、サーバが増えれば増えるほど、それらの効率よい管理＝「運用」が求められています。このコースではその答えの一つである「仮想化」とは何か、なぜ注目されているのかについて解説します。

目次

1.システム運用とは 3
1.1 システムのライフサイクルと業務 4
1.2 運用に関わる要素 5
1.3 システム基盤 7
1.4 高可用システム 8
1.5 開発・運用の分離 8
1.6 運用ルール 9
2.運用の現場から 12
2.1 定常監視 12
2.2 混在環境 12
2.3 バージョン・ライセンスの管理 13
2.4 リソースの適正配分 14
2.5 バックアップ 14
3.仮想化 15
3.1 ハイパーバイザ 15
3.2 完全仮想化と準仮想化 16
3.3 運用上のメリット 16
3.4 グリッド・クラスタ 17
4 実習手順 18
参考資料 20

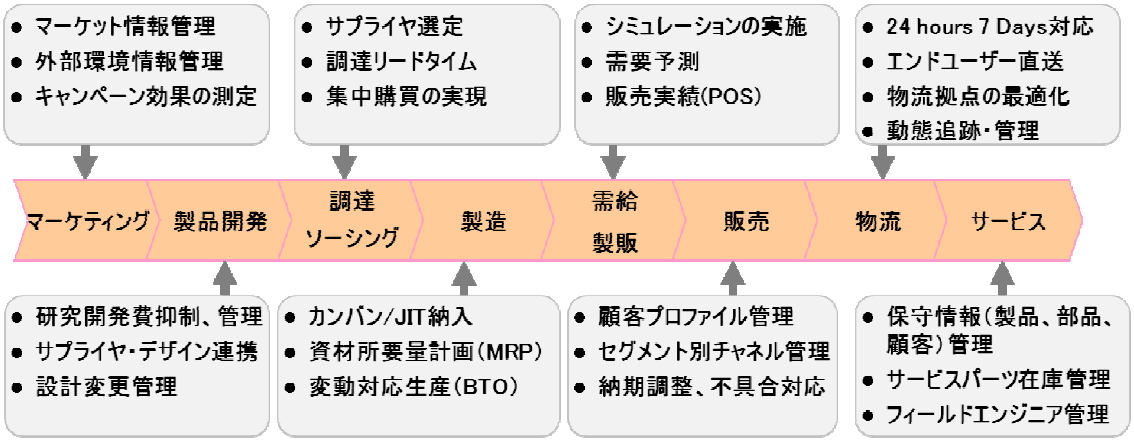
1.システム運用とは

コンピュータ・システムのない社会が今の皆さんに想像できるでしょうか？インターネットで情報を検索する、パソコンで文書作成を作成する、音楽をダウンロードするといった直接コンピュータを使う事だけでなく、普段の生活にコンピュータ・システムはなくてはならない世の中となっています。

ＡＴＭ・ネットバンキング、株式売買などの金融サービス、コンビニや商店の品揃え通信販売といった物流、工業製品の開発・設計、工場で完成品に組み上げる製造行、ガスや電気・水道などのインフラ、携帯・電話・ＴＶ放送などの通信など例をあげればきりがありません。
具体的に電化製品の物の流れを考えると、部品メーカから製品メーカが部品を調達し製品を製造、それを物流会社が販売店へと運び消費者へと届けます。この物の流れのどこにコンピュータが関わるかを考えれば、コンピュータが停止してしまう危険性、社会的影響の大きさがわかると思います。

下図は製造業の物の流れを表しています。仕入れたもの（供給：Supply）を加工し、販売することからこの流れを「サプライチェーン」と呼びます。製造業でなくとも、なにかを仕入れそれに付加価値を加え販売するといった考え方はすべての産業の基本です。サービス業では加工・製造ではなく付加価値に注目することから、Value Chain という言い方もされています。

図１： Supply Chain － サプライチェーン



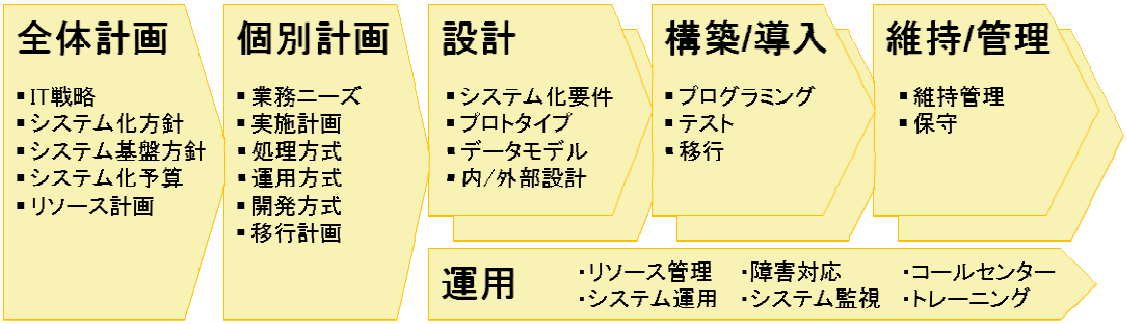
JIT：Just In Time / 必要な時に必要な量を仕入れる
MRP：Material Requirements Planning
BTO: Build to Order / 注文に応じ部品を組み合わせ製造する（PC のカスタマイズ受注）
POS: Point of Sales / 販売時点情報管理

サプライチェーンでは、数多くの企業が商品とそれに付随する情報を交換する必要があり、それぞれの企業の中でも人・物・金といった情報を維持管理する必要があります。その情報管理のすべての局面でコンピュータが活躍しています。

1.1 システムのライフサイクルと業務

コンピュータ・システムが作りあげられ、実際に利用できるまでには数々の工程を踏むことになります。

図 2：システムのライフサイクル



- 全体計画(Strategy)

  企業としてどんなシステムが必要かを見極め、その実現方法を検討し計画を立てます。いわゆる「IT 戦略」と言われるもので、３～５年の期間で計画を立てます。「経営企画」部門が立案する事が多く、IT 投資が盛んな企業では「IT 企画部署」が取り仕切る事もあります。

- 個別計画(Plan)

  IT 戦略に基づき、具体的にどのような仕組みが必要で、それを実現するためには何が必要かを明確にします。たとえば SCM を導入するために必要なコンピュータ、POS レジといった機材や、集計する機能だけなく、店舗から情報を取り込むためのネットワーク、仕入先との調整、取得する情報の定義（顧客層：性別・年齢、天候・気温、仕入先など)、展開する地域や商材など直接コンピュータに関係ない事柄も計画に含まれます。

  大企業などでは要件が膨大になるため、全体の設計方針・設計思想を EA(Enterprise Architecture)としてまとめることもあります。

- 設計(Design)

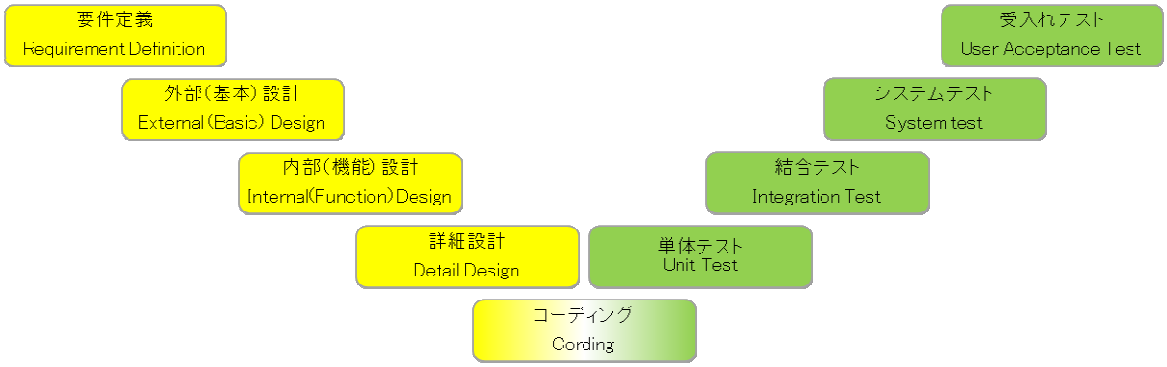
  コンピュータ・システムの設計を行います。大規模なシステムでは要件定義から始まり概要から詳細へと何段階に分割して設計を行いますし、共通機能を切り出してそれと他のモジュールとのインタフェースを設計するようになります。またプログラムだけではなく、データ構造や意匠デザイン、設計ルールといったものも用意する必要があります。

- 構築／導入(Development/Implement)

  実際のプログラミング作業を行います。システム構築で最も人出がかかる工程になります。設計では概要から詳細へと細分化していきましたが、構築では個別のプログラム作成、テスト（単体テスト）した後、それらをつなぎ合わせ動作確認し（連結テスト)、最終的にシステム全体の動作を確認します（システムテスト)。作業量は人月（一

人のエンジニアが１か月かかる仕事量）がよく用いられます。
また構築では「成果物（作ったもの）」が当初の目的とおり正しく動作するかを「テスト」を通してします。この成果物とテストは階層になっており、それぞれが対応する構造になっています。これはドイツで発案されＶモデルと呼ばれています。

図３：システム設計Ｖモデル



- 維持／管理(Maintenance/Administration)
  一度導入したシステムがそのままの状態で使われ続けることはまずありません、実際に使用して初めてわかる入力項目の過不足や、法律改定に伴う計算方法の変更、組織変更など常に修正が必要となります。稼働後のシステム変更やユーザ登録などの作業を維持・管理、または保守業務といいます。

- 運用(Operation)
  導入したシステムが安定稼働するよう、普段からシステムの状態を監視することや、利用ルールの整備・周知徹底を行う事を運用といいます。運用は実際に利用されているシステムだけでなく、開発に必要となる「開発環境」や設計から導入までの手順・規則の整備も含まれます。特にシステムのトラブルは自社だけでなく、サプライチェーンで繋がる多くの企業にも影響し最悪の場合は社会問題にまで発展してしまいます。

1.2 運用に関わる要素

コンピュータ・システムが停止しないよう、普段からシステムの状況を監視し、トラブル発生を予測、それに備えることを「システム運用」と呼びます。この運用で重要なのは「システムが不意に停止しない」「必要な時にいつでも利用できる」という事です。
システム運用で必要となる要素は大きく５つあり、それぞれの頭文字をとって、RASIS といいます。

- R：Reliability（信頼性）
  コンピュータ・システムが故障しないこと。ハードウエアが故障しないことはありえないなので、連続稼働させることが可能な度合い（最長不倒距離）。運用規則を設計するためには平均故障間隔(MTBF: Mean Time Between Failure)という指標が用いられます。MTBF は稼働時間（つまり特定の期間のうち修理・トラブルのため利用できなかった時間を除いた時間）を表し、システムがどれだけ停止せずに稼働するかの指針で、以下の計算式になります。

MTBF ＝ 稼働時間 ／ 故障回数

- A: Availability（可用性）
  MTBF と MTTR(後説)から、システムを稼働させている時間のうち実際に利用できる時間を求めることができます。この 3 指標は特に重要で良く用いられ、RAS と呼ばれます。

  可用率 ＝ MTBF ／ （ MTBF ＋ MTTR ）

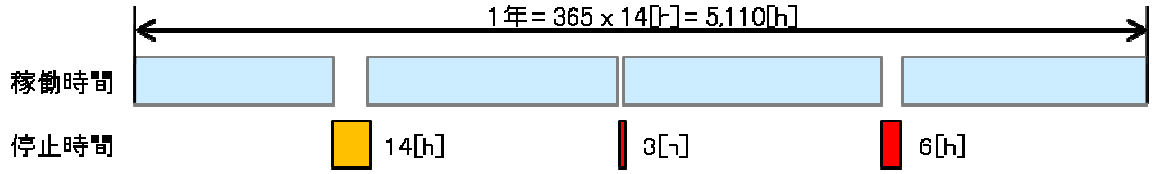
- S: Serviceability（保守性）
  修理のしやすさの指針です。よく用いられる指標は、修理を行う際に、システムが停止する時間の平均である、平均修理時間（Mean Time To Repair、MTTR）です。修理にかかった時間ではなく、修理に伴いシステムが停止した時間である点に注意してください。

- I: Integrity（保全性）
  外部要因によるシステムダメージを食い止めることを表し、一般的な指針はありません。たとえば落雷によるサージ電流流入の回避や瞬時停電に備え無停電電源装置（UPS: Uninterruptible Power Supply ）を導入する、振動による転倒を防止するといった設備の導入だけでなく、定期点検によるトラブルの発生防止、操作運用の手続き・ルールの徹底とその検証といった作業も含まれます。データセンターはコンピュータ・システムを保全するための施設だという事もできます。

- S: Security（機密性）
  本来は保全の中に含まれるものでしたが、インターネットの普及によるコンピュータの一般化、利用者の増加により指針として別に切り出されるようになりました。これも RAS のような一般的な指標はありませんが、情報の「盗聴・傍受」「改ざん」「なりすまし」の防御に必要な事柄を指します。最近では個人情報保護法に見られるように、情報の流出防止や正確な管理（本人による個人情報開示請求への対応）も必要となってきています。

RAS の例：
「あるシステムの稼働状況を調査したところ、2010 年 1 年間で 2 回トラブルが発生しており、それぞれ 3 時間、6 時間のシステム停止であったことが分かっている。また 3 月の第 1 日曜日にビルの計画停電があり、24 時間停止していた。このシステムは毎日 09:00 から 22:00 までの 14 時間サービスを提供している。」

図 4：システム稼働状況（例）



このような場合、稼働率は以下のように考えることができます。

稼働時間　＝3,760 – 14 – 3 – 6 = 5,087[h]

MTBF　= 5,087 / 2 = 2,543.5[h]（計画停電はトラブルではないので含まない）

MTTR　= (3 + 6) / 2 = 4.5[h]

可用率　= 2,543.5 / (2,543.5 + 3.5) = 99.82%

このシステムは年間稼働率 99.82% となります。

インターネットの物販サイトのように 24 時間 365 時間稼働が求められるシステムを稼働率 99.99%で稼働すると、年間に 80 分の停止時間が発生することになります。99.99%は銀行 ATM システムに求められる性能です。

1.3 システム基盤

高い可用性を確保するには、システムの構成要素と利用範囲の特定が必要となります。たとえば使用するハードウエア、オペレーティングシステム、データベースの選択や、それらを使って構築したシステムの用途は何かを明確にします。用途によっては必要となる可用性や保全性が大きく異なります、たとえば単なる会社で使う報告書を印刷するのであれば、プリンタを何台か用意しておけば、1 台が故障していても出力先を変えればよいですし、紙詰まりなどは再度印刷すればかまいません。しかし印刷するものがコンサートのチケットだっらどうでしょう？再印刷を勝手にされては困りますし、誰が何を何枚印刷（チケット発行）したかを追跡できる機能が必要となります。

同様に自分のブログを公開するシステムはたまに停止しても全く問題ないでしょうし（そもそも、毎日更新することが大変です)、早朝や夜中に停止していても気になりません。しかし EC サイトや金融システムがダウンすることは許されません。

このように情報サービスを提供する上で必要となる要件は異なります。情報システムが必要とする要件を満たすコンピュータ資源、環境を総じて「システム基盤」と呼びます。会社のシステムをひとつのシステム基盤で賄うこともありますが、用途に応じたとえば基幹系、情報系、インターネット系など複数用意している企業も多く存在します。

図 5：システム基盤の構成要素

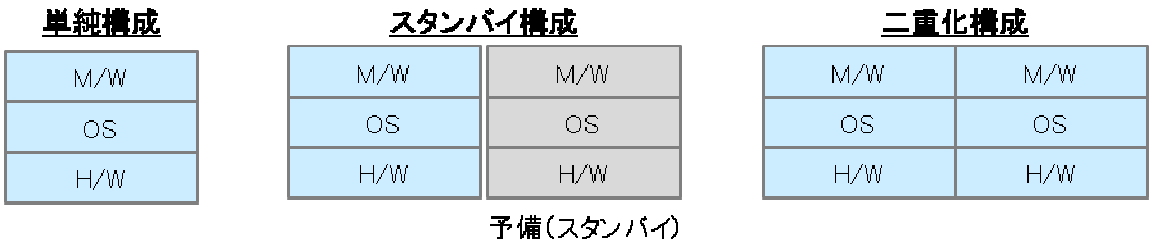
構成要素	解説
ミドルウエア	複数のシステムで共有されるソフトウエア。データベース、メッセージ交換(EDA/EDI)などがある。
オペレーティングシステム	Windows や Linux といったOS
ハードウエア	CPU, Memory といった同一筐体に格納されているものだけでなく、外部HDD、プリンタ、ネットワークなどを含む。

システム基盤は複数のアプリケーションから参照されるため、その決定は十分に吟味する必要があります。また影響範囲が大きくなるため、変更方法や構成管理についても併せて決定しておきます。

1.4 高可用システム

可用性の高いシステムには HA(High Availability)や、FT(Fault Tolerance)などがあります。単純にシステムの可用性を上げるためには、必要なシステムひと揃いを２式用意しておくことが考えられます。そのうち、トラブル発生時に切り替えるものをスタンバイ構成（デュプレックス・システム）、常時二式とも稼働させておき、不具合があった場合にそれを切り離すものを二重化構成またはデュアル・システムと呼びます。これらは HA と呼ばれます。

図 6：HA システム例



スタンバイ機を停止させておくものはコールド・スタンバイ、つねに稼働させておき障害発生時に切り替えるものをホット・スタンバイと呼びます。

HA の中でも、トラブルへの対処に関する考え方で Fail Safe システムと Fault Tolerance システムに分かれます。FS はトラブル発生時に不要不急なサービスは停止させても安全を優先し、不要不急なサービスを停止した状態を「縮退運用(Degrade mode)」と呼んでいます。FT に至ってはトラブルが発生しても多少の性能劣化はあるものの、サービスを提供し続けるシステムを指します。

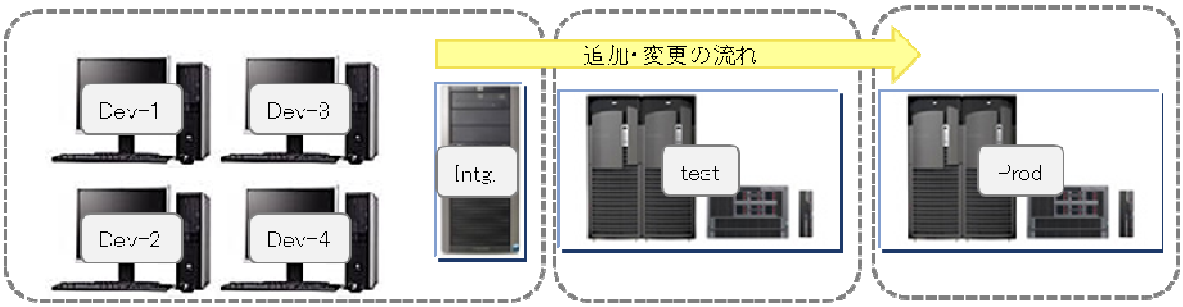
ちなみに退役が決まっているスペースシャトルの制御システムは４セットのシステムからなり、１台がエラーを発すると他の３台で多数決をとり、その処理を行う複雑な冗長化を採用しています。

1.5 開発・運用の分離

多くの IT システムでは、利用するシステムだけではなく開発するためのシステムも必要です。システムは稼働した瞬間から修正作業が始まります、いわゆるバージョンアップに向けた作業です。しかし実際に利用しているシステムを直接修正すると、バグによるサービ

ス停止やデータの破損などが発生するため、別のシステム基盤を用意しそこで開発や保守を行います。
実際にサービスを提供するシステムを業界では「本番環境(Production System)」、プログラムやデータをバージョンアップや不具合修正のために変更するシステムを「開発環境（Development System)」と呼び、運用方法も使い分けています。
基本的に本番環境はよほどの事がないかぎり、停止・修正を行いません。一方開発環境は必要に応じ OS やミドルウエアのバージョンアップや、プログラム・データの修正が行われます。そして開発環境で変更した内容を本番環境へ反映する際に、変更内容を確認する「テスト環境（Test System)」も必要となります。
また Web サイトの世界では、開発環境が複数社に分散していることがあります。そのような複数の開発環境から目的となるひとつのシステムへ組み上げる環境として「統合環境(Integration System)」も最近では増えています。
このように複数の環境がセットになって実際のサービスが提供されています。この環境の集合を「システム・ランドスケープ」（独 SAP 社）と呼ぶことがあります。

図 7：システム・ランドスケープ例



運用方法の視点では、各環境のルールを厳密に規定することが重要です。開発している人が勝手に本番環境を操作することは好ましくありません。特に金融業界では厳しく制限され、扱う組織さえも情報システム開発 1 部、運用 2 部というように分離しています。
開発担当者がプログラムを修正し、それをテスト担当に引き渡しテスト結果が良好であれば、本番運用担当に引き渡しサービスが提供されます。この本番環境で新しい機能が稼働することをリリースと呼んでいます。
本番環境での動作を確実なものとするため、テスト環境は本番環境と全く同じ構成にする事が望ましいですが、コストの関係から HA 構成のバックアップ環境の流用や、若干グレードの低いシステム（特に CPU 性能、ディスク容量など）を使う場合もあります。

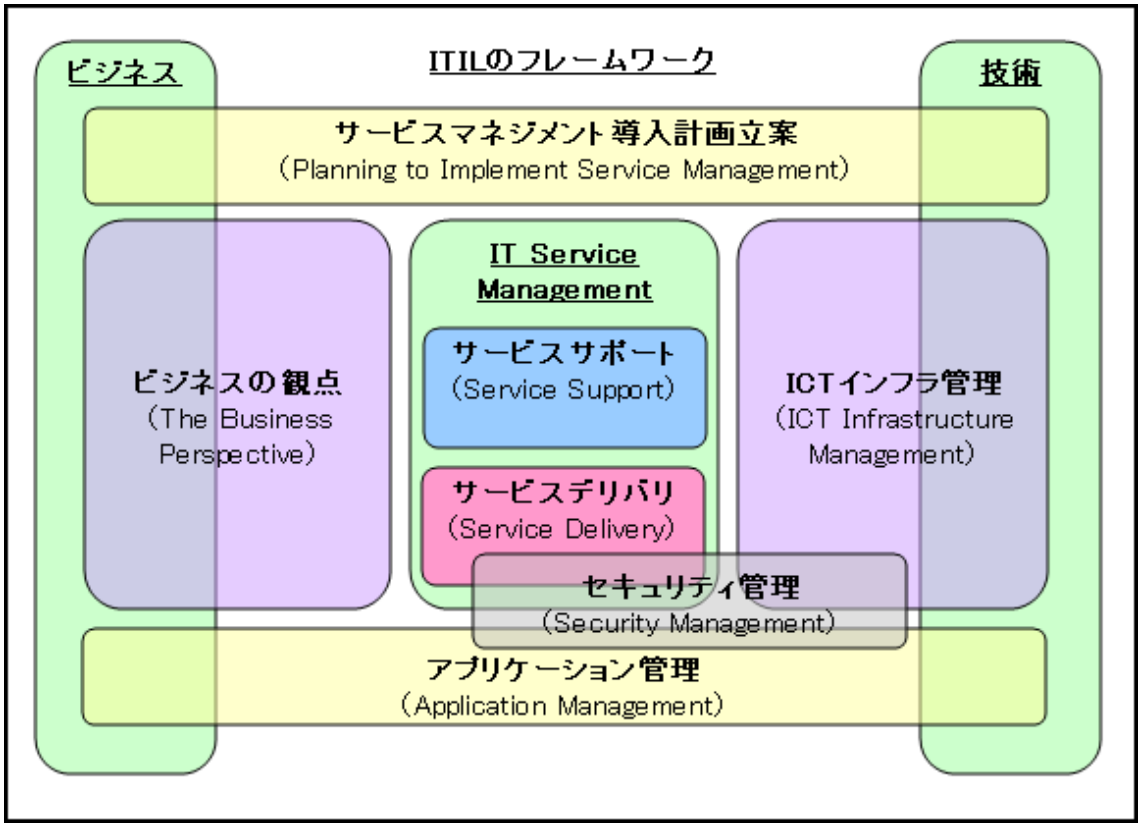
1.6 運用ルール

コンピュータ・システムを安定動作させるには、HA 構成の検討や定期的なチェックは欠かせませんが、それらを実際に行う人の作業手順も重要です。本番環境で改善要求があれば、それの内容を確認し、適切なものかどうか判断のうえ開発環境で追加・変更を行います。追加変更が正しいものであるか、他への影響はないかをテスト環境で最終確認し最後には本番環境でリリースします。
企業においてこのような作業手順が正しく行われているか、指揮命令系統が正しく作動し

ているかを表すのが「コーポレート・ガバナンス」や「内部統制」です。上場企業に内部統制が行われている事の証明を求めるのが SOX 法です。IT 分野でも同様の規律が求められ、IT ガバナンスや IT 統制（IT 全社統制/ITLCL: Company Level Control、IT 全般統制/ITGC: General Controls、IT アプリケーション統制/ITAC: Application Controls）があります。

この一連の流れが正しく行われているか、その証跡を取得し適切な人物（責任者）が判断を下しているか、といった作業の流れを規定したガイドラインが ITIL です。
ITIL(Information Technology Infrastructure Library)は 1989 年にイギリスで誕生し、多くの国々で規準として採用されています。

図 8：ITIL　V2.フレームワーク



ITIL は当初英国規格 BS 15000 でしたが、その後国際規格に採用され ISO/IEC 20000 となっています。日本ではさらに ISO/IEC20000 に準拠した JIS Q 20000 が公開されています。検定試験もあり IT 業界とくに運用に携わる人には必須となりつつあります。

図 9：ITIL 資格一覧

資格グレード	内容
マネージャ	最上位資格で論述試験やマネージャ認定コースの受講などが義務づけられています。
プラクティショナ	ITIL のフレームワークに関する資格であり運用設計を行うレベルの人向けの内容です。
ファンデーション	基本的な内容であり、運用者以外でも IT 業界では常識として取得を推奨している会社も多数存在します。

他に運用に関する規格としては、以下のものがあります。IT サービス業として取得を推奨・義務づけられているものも多く存在します。

図 10：システム運用関連規格

規格名	概要
ISMS （Information Security Management System）(ISO/IEC 27001, JIS Q 27001)	企業や団体における情報セキュリティを管理する仕組み。企業単位で認定を受けることができる。
COBIT(Control Objectives for Information and related Technology)	情報システムコントロール協会（ISACA)と IT ガバナンス協会（ITGI)が制定したフレームワークで、企業内の IT ガバナンスや内部統制のガイドとして用いられる。
CMMI(Capability Maturity Model Integration)	組織がプロセスをより適切に管理できるようになることを目的として遵守するべき指針を体系化したものでレベル１～５の認定を受けることができる。（レベル３以上）
ITSMS(IT Service Management System) (ISO/IEC 20000, JIS Q 20000)	IT サービスマネジメントに関する国際規格。ITIL に似ているが管理を主眼として編纂されている。
BCMS(Business Continuity Management System) (BCM/BS25999)	事業継続性に関する仕組みと、その適用に関する認定。
ISO 9001 （JIS Q 9001:2000）	製造やサービスに関する品質の基準。何度か改定があり現在は ISO 9001:2000 が適用されている。
ISO 15001 (JIS Q 15001)	個人情報保護に関する規格であり、認定されると「プライバシーマーク」の使用が認められる。
SAS70(Statements on Auditing Standards No.70)	内部統制が正しく実施されている事を評価する監査基準。業務委託を行う場合に基準となる場合がある。
産業省　システムインテグレータ登録、システムオペレーション等企業認定	SI 登録制度は、システム開発を円滑に遂行できる企業かどうかの審査登録制度。SO 認定制度は、システムのアウトソーシング能力を認定する制度。

2.運用の現場から

システム運用とは何かを解説してきましたが、ここでは実際に現場で起こっていることを例示し、その対応方法を検討します。

2.1 定常監視

運用の基本は、情報システムが正しく動作しているかどうかを常時監視することにあります。トラブル発生をユーザから指摘されて動くのではなく、トラブルの発生を未然に食い止めるためにシステムの挙動を常日頃監視します。

ネットワーク負荷、CPU・メモリ・ディスクの使用率、サーバの疎通確認(ping 監視、死活監視などとも)、各種エラーログの確認といったコンピュータで収集可能なものから、サーバが設置されている部屋の温度チェック、配線や USB 装置などの有無、ハードディスクなどの騒音といった実際に担当者が目視して行う確認などもあります。

毎回各サーバにログインしログや利用状況をチェックするのは大変なので、多くの場合は専用のプログラムを用意し自動化を行います。単に手間を省くための自動化ではなく、担当者のスキルに依存しない、チェック漏れがないといった目的からも自動化が推奨されています。

サーバの稼働状況をチェックするプログラムは、運用担当がシェルスクリプトなどで手作りする場合もありますが、多くは専用のツールを用いています。とくにサーバが複数になってくると、1 か所で集中管理できる統合監視ツールがよく用いられます。

有名なものとしては以下があります。

図 11：主要統合システム管理ソフト一覧

商品名*1	概要
JP1	日立製作所、国内での実績が多くデファクトスタンダードに近い
CA	CA。世界的に有名なツールであり導入実績も非常に多い。
Tivoli	IBM、IBM の純正ツールという事もあり IBM 導入企業での採用、特に金融業界では多い。
ZABBIX	オープンソースのシステム管理ツール

*1)プロプライエタリ（オープンソースの対語）製品は非常に多くの製品群を形成しており、ブランド名とした。

2.2 混在環境

システム運用の中で最も手間がかかる環境は、OS やハードウエアがバラバラの状態だといえます。OS が異なれば監視する方法も当然異なるでしょうし、そもそも操作方法が異なり要因の確保が難しくなります。IT 業界では複数のメーカや OS が混在した環境を「ヘテロジニアス（異質、異種）環境」と呼びます。

このような環境の場合は、機種に依存しない運用監視ツールなどを導入することで、運用担当者が OS やメーカの違いを意識せずに操作ができるよう工夫する必要があります。操作方法が同一である度合いや、データ交換が可能な尺度を「相互運用性（Interoperability）」と呼びます。

Java は機種ごとに VM(Virtual Machine)を用意することで、ひとつのプログラムを複数のシステムで動作させることができます(Write once Run anywhere)。またアプリケーションソフトウエアでも機種に依存しにくい設計の物があります。たとえば独 SAP 社の ERP ソフト R/3 や mySAP は、OS、DBMS、開発言語をひとまとめにした BASIS 層と呼ぶ仮想化インフラを提供することで OS、DB に依存せず動作する事が出来きます。Oracle DB も複数の OS で動作可能ですし、DB の場合はプログラムの DB 問合せ部分を機種非依存の ODBC(Open Database Connectivity)や JDBC を使って実装できるか開発を行う前に検討します。

図 12：SAP R/3 BASIS － システム抽象化層

このようにヘテロジニアス環境になる場合には、その上で動作するミドルウエア部分を機種非依存にしたシステム基盤を設計する必要があります。
また OS や機種を統一するという考え方もあります、たとえば Google は Linux をベースにしたオリジナル OS を自社で製作したハードウエアに搭載し運用しています。ちなみに 2011 年 3 月現在で Google が持つ Web サーバ数は 1,500 万サイトで月に 50～70 万サイト増加しています。

2.3 バージョン・ライセンスの管理

システム運用で次に問題となるのが OS などの基盤にかかわるソフトウエアの管理です。基盤で利用されるソフトの多くはライセンス（利用許諾）が厳しく管理されています。登録ユーザ数やインストール対象マシンの制限などに抵触しないよう十分な管理が必要です。企業によってはクライアントソフトを含めライセンスの棚卸を定期的に実施しています。
しかし、本当に問題になるのはそれらソフトのバージョン管理です。Windows XP と Vista や 7 では動作に違いがあり、仕事で使うアプリケーションの前提条件となります。
例えばＷｅｂサーバは Windows 7 が前提、勤怠管理ソフトは XP が前提となるような場合は、それぞれに基盤を用意しなければなりません。
さらに、バージョンによってはサポート期間が設定されていたり、対応するハードウエアの組み合わせが指定されている場合があります。このように基盤ソフトとアプリケーションソフトのバージョンの組み合わせを把握し管理する必要が生じます。

このようなハードウエアやソフトウエアおよびバージョンの組み合わせの管理のことを「構成管理」と呼び、システムの更改には十分な配慮が必要となります。

2.4 リソースの適正配分

アプリケーションにより前提条件となるシステム基盤（およびバージョン）が異なる事があります。また導入した時期によっては使われなくなっているサーバや、負荷が集中しているサーバなど、その利用状況がまちまちになってしまうことがあります。

例えばハードディスクなどは、「共有ファイルサーバは常に空き容量不足、ところが個々のサーバを調べてみると、意外と空き容量がある。」といった事が良く発生します。

図 13：サーバ構成例

資源	ファイルサーバ	メールサーバ	Web サーバ
ディスク（利用率）	120GB(80%)	300GB(50%)	100G(90%)
Memory	4GB(12%)	1GB(20%)	2GB(88%)
CPU	4core (20%)	2core (40%)	4core(30%)

このようにならないためにも、コンピュータ資源を包括的に管理し過不足がないよう調整する必要があります。またディスクなどは柔軟に割り当てられるよう各サーバから切り離し共有化された NAS(Network Attached Storage),SAN(Storage Area Network)などの製品を活用する手立ても考えられます。

コンピュータ資源を管理し適切に割り当てることを「キャパシティ・プランニング」とか「サイジング（必要資源見積もり、性能見積もり）」と言います。

2.5 バックアップ

システムのバックアップはシステム管理の中でも重要であり、非常に手間がかかるものです。特にデータ統合が進むとディスクには種々雑多なデータが蓄積しトラブル発生時に、どの範囲のデータをいつまで遡って回復すればよいのかを見極める必要があります。またデータ容量が増大するなか、バックアップにかかる時間も長期化し、一晩で終わらないことさえ発生します。

またバックアップデータをテープやＤＶＤといった取り外し可能なメディアに取得した場合、その格納場所を管理する必要があります。特に重要なデータは暗号化するといった工夫も必要になります。

バックアップはデータだけでなく、プログラムやシステムその物を含む場合もあります。最も大規模な例ではデータセンターごと複数配置し、災害・紛争が発生した場合でもネットワークの仕向け先を変えることでサービス提供を継続します。災害時にも備えたバックアップをディザスタリカバリと言い地理的に離れた拠点間で実施します。

システムをコンピュータ・システムだけでなく会社機能全てをバックアップすることを事業継続計画(BCP: Business Continuity Plan)といい、市場における企業評価にもつながります。

3.仮想化

既に記述したように、システム運用は非常に複雑な物へと変化してきています。そんな中、「仮想化」が注目されています。

仮想化とは１つのハードウエアをソフトウエアによって、複数のハードウエアがあるように見せかける技術で、古くは1970年代に汎用機で実装されVMモニタと呼ばれていました。当時は高価な大型計算（ハードウエア）を複数用意する事が難しく、少ない資源を有効に活用するために編み出されました。現在の仮想化ブームの火付け役ともいえるVMwareはx86(_64)用仮想化ソフトとして1998年に誕生しました。

現在よく用いられる主力製品としては以下の４つがあります。

図 14：主要仮想化ソフト一覧

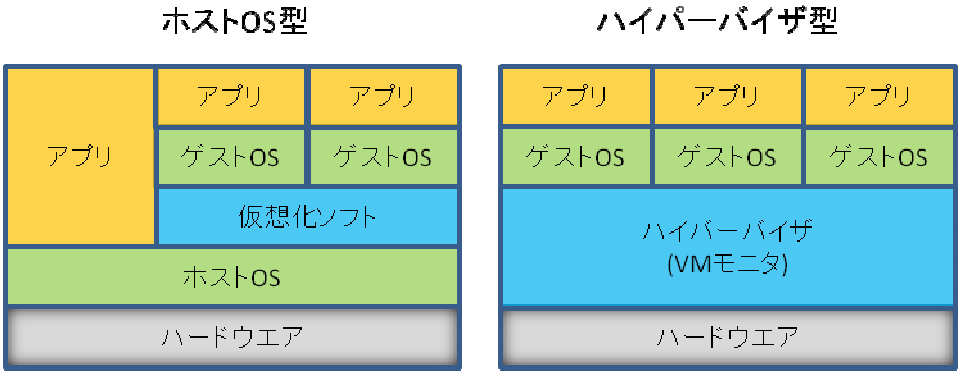
	VMware vSphere	Xen Server	Hyper-V	KVM
ベンダー	EMC(VMware)	Citrix(Xen)	MicrOSoft	Red Hat
特徴	最古参で安定しておりサポートが充実。 高スケーラビリティ	オープンソースの高い拡張性。 準仮想化による高パフォーマンス。	Windows Server 2008 添付。 高コストパフォーマンス	カーネル組込で高パフォーマンスが期待できる。

最近ではサーバ側だけでなく、クライアントの環境を仮想化しサーバに集約する「仮想化ディスクトップ」と「シン・クライアント(Thin Client)」の組み合わせも普及しています。

3.1 ハイパーバイザ

仮想化ソフトは大きくホストOS型とハイパーバイザ型の２種類に分かれます。

図 15：仮想化ソフトの違い



ホストOS型は、既に動作しているOS（ホストOS）上で仮想化ソフトを動作させ、さらに、その上にゲストOSと呼ばれるOSをインストールします。VMware Workstation / Server や、MS Virtual PC などがこの方式です。

ハイパーバイザ型は仮想化ソフトウエアをOSとしてインストールし、その上にゲストOSをインストールして動作させます。Xen やKVM、VMware ESX、MS Hyper-Vはこのタイプになります。ハイパーバイザはホストOS型に比べ無駄な処理が少なく処理効率が高いという効果があります。

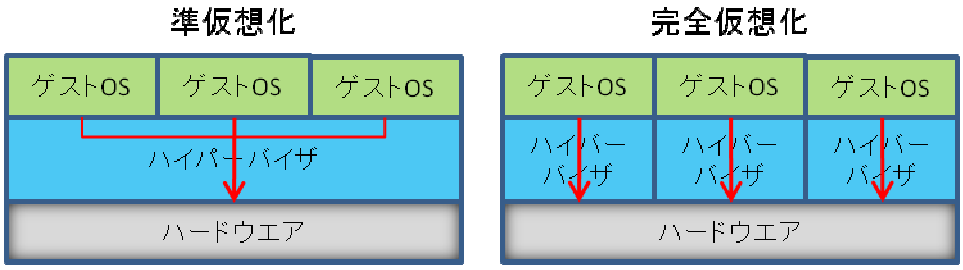
なおハイパーバイザ型は仮想ソフトが直接ハードウエアを駆動することから「ベアメタル(Bare Metal、金属むきだしの意)」や、「VM(仮想マシン)モニタ」とも呼ばれます。またOSの別名がスーパーバイザだったことから、その上位に位置づくVMモニタをハイパーバイザと呼んでいます。

3.2 完全仮想化と準仮想化

ハイパーバイザ型はさらに完全仮想化と準仮想化の2種類に分類されます。完全仮想化はゲストOSから見た仮想ハードウエア（＝ハイパーバイザ）が、実ハードウエアと区別できないほど完全に置き換えた状態を指します。ゲストOSをそのまま利用できるというメリットがありますが、仮想ハードウエアの処理が足かせとなってパフォーマンスが低下するデメリットがあります。

準仮想化はゲストOSに手を加えることで、HDDやネットワークといった本当に物理装置に対する入出力以外はハイパーバイザ内で最適化します。パフォーマンスは向上しますが、ゲストOSに手を加えることができない場合は動作させることができません。

図16：準仮想化と完全仮想化



3.3 運用上のメリット

仮想化によるメリットとしては以下のものがあります。

・リソース有効利用

もともとハードウエアは1つなので一元集中管理が可能で、資源をハイパーバイザやホストOSを通じ、ゲストOSへ動的に割り当てる事ができます。余裕のあるサーバから負荷の高いサーバへ資源を移動することで、資源利用の平滑化ができます。

またハードウエアを集中させることで、管理対象を絞り込む事ができます。さらに最近は環境負荷軽減も注目されています。

・レガシーマイグレーション（過去資産の移行）

ゲストOSが必要とするハードウエアそのものではなく、それに準ずる環境をソフトウエアで提供（エミュレーション）するため、販売終了・保守期限切れのハードウエアを前提とするソフトウエアでも稼働させることが可能です。Win XP SP-1以前のOSを最新のハードウエアで動作させるためにはドライバが対応しないなど大変な苦労を伴いますが、仮想化することで作業負荷を軽減できます。

・可用性の向上

ハードウエアに障害が発生し縮退運転となった場合でも、仮想化されたゲスト OS を他のハードウエアへ移動して継続する「マイグレーション」機能、システム基盤の共通部分を予めイメージファイルとして作成・保存しておき、サーバ追加時は固有部分を追加・変更しすばやく立ち上げる事ができます。また同様の機能を用いてシステムを丸ごとコピーやバックアップする事ができます。このようにサーバをまるごとコピーする機能を「クローン」と呼びます。

3.4 グリッド・クラスタ

クラウドコンピューティングでは仮想化ともう一つグリッド・コンピューティングがあります。仮想化は１つのハードウエアで仮想的に複数のシステムを動作させましたが、グリッド・コンピューティングでは複数のコンピュータで１つのシステムを動作させます。

ネットワークを使い複数のコンピュータを接合する「疎結合クラスタ」、より科学技術演算に特化した「HPC: High Performance Computing」。CG やスーパーコンピュータの演算部に多くみられる「Vector Processor」はバスを使って多数の CPU を接続した「密結合クラスタ」といわれます。スマートグリッドは町中にある PC や PC が搭載された自動車、店舗などを一つの大きなコンピュータの集合としてとらえ、効率的なエネルギー利用を目指しています。

4 実習手順

実習では OSS でパフォーマンスがよく、動作環境の制限が少ない Xen を用いる事にしました。Xen は CentOS に組み込まれたハイパーバイザとして動作し、完全仮想化・準仮想化で利用することができます。

0. BIOS の設定

（BIOS はメーカ、機種によって内容が異なります）

HDD の設定を高速(SATA モード)、CPU の仮想化オプションを有効にしておきます。

System – Drivers

-- SATA Operation ［**Normal**］ Legacy

Performance

-- Virtualization ［**On**］ Off

1. ゲスト OS キットの準備

i. DVD をセット、自動的にマウントされた場合はアンマウントしておく。

```
#  umount  /dev/scd0
```

ii. Web サーバの公開ディレクトリを作成

```
#  mkdir  /var/www/html/CentOS
```

iii. 公開ディレクトリへ DVD をマウント

```
#  mount  /dev/scd0  /var/www/html/CentOS
```

iv. Web サーバを起動

```
#  /etc/init.d/httpd  start
```

2. 仮想マシンの作成

i. 「仮想マシンマネージャ」起動

最上行を選択し、[+新規(N)]ボタンをクリックし「新規の仮想マシンを作成」を起動

ii. 「仮想システムの名前を指定」

「システム名（N）」は任意。今回は vm01 とする。

iii. 仮想化方式

「準標準化」を選択

iv. インストール方法

「ネットワークのインストールツリー」を選択

v. ＵＲＬの指定

・サーバは自ＩＰ（192.168.xx.xx など。不明な場合は /sbin/ifconfig で確認）

・ＵＲＬは(1)で作成したディレクトリ「/CentOS」

vi. ストレージ

規定値を採用（４ＧＢ必要）

vii. ネットワーク

仮想ネットワークを採用

viii. メモリとＣＰＵ割り当て

メモリは最大を 512MB、起動時 256MB

ix. 終了

[終了(F)]を押すと、ディスクイメージが作成されインストールが開始される。

3. ゲスト OS インストール

i. 言語選択：日本語(Japanese)

ii. キーボード：日本(JP106)

iii. ネットワーク設定：IPv4 を選択

iv. (インストールメディア： 自ＩＰと/CentOS を指定)

v. パッケージ選定（必要最小限を選択）

・ディスクトップ環境：GNOME のみ選択

・アプリケーション：エディタ、グラフィカルインターネットのみ選択

・開発：選択しない

・サーバ：印刷サポートのチェックを外す

・ベース：ダイアルアップネットワークを外す

演習

・スナップショット

ある時点でのゲスト OS の状況をまるごとファイルとして記録することをスナップショットと呼びます。ノートＰＣの休止モードに似た機能で、Xen では仮想マシンモニタから実行できます。

メニュー：仮想マシン（M）＞保存

スナップショットを保存する場合は、仮想システムを一時停止させる必要があります。
保存したスナップショット(dump)は、同様にメニューから呼び出すことができます。

・クローン

Xen でクローンを作成するには、virt-clone コマンドを用います。

```
# virt-clone --original vm01 --name vm02 --file /var/lib/xen/images/vm02.img
```

参考資料

- ✓ 高信頼システム構築標準教科書（LPI-JAPAN）
  http://www.lpi.or.jp/linuxtext/ha
- ✓ @IT Linux フォーラム
  http://www.atmarkit.co.jp/flinux/
- ✓ Linux Academy 関連資料
  http://ycOS.sakura.ne.jp/LA