第Ⅴ章　コンピュータシステムの性能と信頼性

第Ⅴ章　コンピュータシステムの性能と信頼性

学習目標

1．コンピュータシステムの高性能化技術を整理して理解させる。
2．RISCコンピュータが使われてきている理由を理解させる。
3．多重プロセッサの構成の仕方を理解させる。
4．並列処理とプログラミングの関係を理解させる。
5．コンピュータシステムの性能を大ざっぱに理解させる。
6．コンピュータシステムの高信頼化技術を整理して理解させる。
7．コンピュータシステムの稼動率に関する簡単な計算ができるようにする。

全体概要

情報システムの大規模化複雑化に伴い、性能の問題が大きくクローズアップされている。また、ネットワーク環境の拡大も急速に進んでおり、データの発生源が多様化しピーク特性も我々が予測できないほどになってきており、大量のトランザクション処理を伴うことになる。したがって、システムの性能設計の重要性がさらに増しており、システム容量の設計や効率化の設計にはレベルの高い技術が求められるようになっている。

この問題とともに、高度情報化社会の進展に連れコンピュータへの依存度がどんどん増しており、コンピュータの障害が及ぼす不安が大きく指摘され、システムの高信頼性への要求はさらに高まっている。情報システムの高信頼性実現のための設計技法を大いに学ばなくてはならない。

内容のあらまし

節　・　項	内　容
1．コンピュータの高性能化	
（1）コンピュータの性能測定	TPC、 SPEC、 BAPCo
（2）高性能化技術	素子、 回路構成、 処理の並列化 バッファ、コンパイラの最適化
（3）命令のパイプライン処理とRISC	パイプライン処理、RISC
（4）スーパーコンピュータ	ベクトルコンピュータ、ベクトル化率
2．多重プロセッサシステム	
（1）密結合多重プロセッサシステム	
（2）疎結合多重プロセッサシステム	完全疎結合、不完全疎結合
（3）アレイコンピュータシステム	
（4）タンデム多重プロセッサシステム	
3．コンピュータの信頼性	
（1）RASIS	RASIS、RASISの評価尺度
（2）信頼性を高める構成	デュプレックス構成、デュアル構成 分散処理構成
（3）無停止コンピュータ	
（4）稼動率の計算	直列系システムの稼動率 並列系システムの稼動率 N－out－of－Mの稼動率 直立系と並列系が混在するシステムの稼動率
4．集中から分散へ	
（1）集中システムの長所	
（2）集中システムの問題点	
（3）分散戦略の利点	
5．主要用語	

1．コンピュータの高性能化

（1） コンピュータの性能測定

コンピュータの性能測定の最重要な項目の一つとして、実行速度がある。これを計測する単位として、MIPS、FLOPS（浮動小数点演算の命令実行回数）がある。商用のコンピュータは、これらのMIPS、FLOPS値を公表して速さでの競争を優位にしようとしている。CPU速度の評価手法としては、命令ミックス（Instruction Mix）とベンチマークテストがあげられる。

① 命令ミックス

典型的なプログラムを実行して命令の使用頻度を計り、それを重みとして、一命令当たりの平均実行時間を評価する。これは、プログラムの応用分野によって傾向が異なるため、科学技術計算用の命令の使用頻度から平均実行時間を出したものをギブソンミックスといい、事務計算用から出したものをコマーシャルミックスという。この考え方は、現在では陳腐化しており、この値で全てを評価することはありえない。

② ベンチマークテスト

標準的なプログラムを実際に実行して速度を計る方法である。Whetstoneテストや Dhrystoneテストはよく使われてきた。前者は、浮動小数点を頻繁に使うプログラムを使って浮動小数点性能を評価するもの。後者は、システムプログラムのような数値計算の少ないプログラムをサンプルとして命令の使用頻度を出し、整数演算性能を測定する。ただし、これらの方法はキャッシュメモリの拡大化にともない意義は薄れてきた。

③ SPEC（Systems Performance Evaluation Cooperative：システム性能の評価・標準化協会）

SPECが開発したベンチマーク手法である。

CINT92およびCFP92というベンチマークであり、それぞれ整数演算性能用、浮動小数点演算性能用に何本かのテストプログラムにより平均値の評価を行う。性能値をそれぞれSPECint92、SPECfp92と呼んでいる。

④ TPC（Transaction Processing Performance Council：トランザクション処理性能評議会）

OLTP（オンライントランザクション処理）システムの性能を評価する基準としてTPC－A、TPC－B、TPC－Cなどを開発している。

a．TPC－A：Bank of Americaベンチマークを起源とするATMを用いた銀行の入出金システムを対象としたベンチマーク

b．TPC－B：TPC－AからDBMS部分だけを取り出したもの。

c．TPC－C：実運用されている受発注業務システムをモデルとし、特定業務の性能を評価するもの。

（2） 高性能化技術

高性能コンピュータを実現する要素技術として、素子技術、回路構成、処理の並列化、バッファの使用、コンパイラの最適化が挙げられる。

① 素子

素子の微細化や高集積化により半導体素子はどんどん高速化してきた。だが、最近は加工寸法の限界、消費電力の増大により、大きな壁に突き当たっている。限界を超えるべく様々な素子技術の研究が行われている状況である。いくつかの例を挙げよう。

- 21世紀には、数ギガビットクラスの集積度の実現を目指した超々LSIの研究開発
- 高速化と低消費電力を実現する素子として、超伝導素子への期待が大きい。
- 現在はまだアイディア段階ではあるが、光素子の研究が始まっている。例えば、500THzの

光は5GHzの電波と比べ10の15乗倍もの情報を伝送できる。

② キャッシュメモリ

キャッシュメモリは、CPUの演算速度に見合うよう、アクセス速度が6～8ナノ秒程度のSRAMで構成される。容量は、CPU内蔵キャッシュの場合64～256Kビットである。キャッシュには、プログラムの実行に必要な命令やデータを含むメモリーブロックを予め読み込む。必要な内容がキャッシュになければ主記憶から転送する。この転送時間を短縮するためには、128ビットあるいは256ビットのバス幅で主記憶の内容をキャッシュに転送する。

さらに、高速化をしたければ、2次キャッシュをキャッシュと主記憶の間に付加する方法がある。これは、数メガバイトの容量で12～15ナノ秒前後のSRAMを使う。

③ 処理の並列化

並列処理マシンや超並列プロセッサとは異なるが、多数の演算機、プロセッサ、メモリを高速に相互結合することによって高速化が行われる。

④ バッファの使用

主記憶装置とディスクのバス上にキャッシュメモリを置き、データの先読みや一括書き込みを行うことによって実際のI／O回数を減らすとディスクI／O時間が短縮できる。

⑤ コンパイラの最適化

ソフトウェアの高速化技術も重要である。特にコンパイラの最適化機能は欠かせないし、最適化の度合いもますます要求が高くなっている。

これには、従来のCISCマシンにおいてコンパイラ開発技術の一環として行われてきたが、マルチプロセッサ、RISCマシン、並列マシン、超並列マシンに対しては並列処理機能を十分に使うオブジェクトコードが生成される必要がある。

余談にはなるが、最近はコンパイラの最適化だけではなく、既存ソフトウェアの部分的な切り出しを行った上での並列マシンへの適用が重要になっており、これはコンパイラの役割よりは、ソフトウェアの最適化技術の向上に期待されるところである。

（3） 命令のパイプライン処理とRISC

CISC型のプロセッサは高速化にやや限界が見えている。RISC型プロセッサは大方のUNIXコンピュータで採用されているが、下記のような高速化に寄与することがポイントである。

① 単純な命令セット

命令の数、アドレスモードを最小限にとどめている。これにより命令解読時間の減少、命令フォーマットを単純化してワイヤードロジックにより高速化を行える。

② パイプライン処理

一つの命令の実行を取り出し、解読、実行、書き込みのようなステップに分解し、それらを独立的に同時実行させることにより高速化を図る。

パイプラインがよく機能するために、各命令の実行サイクルが同じでなければならない。このため、命令を単純化して大部分の命令を1サイクルで終了するよう設計されている。

③ メモリアクセス頻度の削減

CISCプロセッサと異なる点は、メモリ上のデータを命令のオペランドとせずに、演算に必要なオペランドは必ずレジスタにロードしておき全ての演算がレジスタ上で行われる点である。各命令の実行時間が一定に保たれ、パイプラインが有効に機能するため、高速化が図れることになる。

④ レジスタの拡充

メモリのアクセス回数を減らすためには、CISCよりも多くのレジスタを持つ必要がある。その個数は、決まっているわけではなく、数十個ないし百個を超えるケースもある。また、その使い途も演算やデータの出し入れだけでなく、他のモジュールの呼出を効率的に行うような

局面でも対象とされている。

⑤　コンパイラの最適化機能

CISCはある意味でコンパイラがコードを生成し易い命令形式である。勿論、CISC型のコンピュータに対するコード生成においてかなりの最適化が行われている。しかしながら、RISC型のコンピュータに対するコード生成はややむづかしい。その上、RISCプロセッサの高速化機能を活かした最適化を行う必要があり、コンパイラの最適化機能は非常に重要である。

a．レジスタの割当て

主記憶へのアクセスを最小化するためのレジスタの有効利用が欠かせない。例えば、プログラムをグローバルに解析して変数、テンポラリ変数は演算時にできるだけレジスタに載っているようにレジスタの割当てを行う必要がある。

b．パイプラインの効率的な利用

主メモリからレジスタへロードするための消費時間は、パイプライン待ちを起こす可能性があり、同時に別の命令を実行して効率を上げることが望ましい。

ロード命令前後の命令シーケンスが入れ替え可能で、そのデータに無関係であればロード命令の直後に置き換える。これによりパイプラインの遅れはかなり解消される。

（4）　スーパーコンピュータ

米国において、気象データの解析をはじめ、宇宙・航空、軍事、原子力の工学的な計算処理需要によりスーパコンピュータの技術が発展してきた。IBMは、1960年代後半には既にアレイプロセッサを接続していた。また、CDCは高速コンピューティングに注力してきたがセイモア・クレイ氏がCray Research社を作りCRAY－1を出すと、パイプライン方式のベクトルプロセッサとして高い地位を獲得できた。

①　ベクトルコンピュータ

複数個のベクトル演算器（パイプラインともいう）の並列使用により高速化を果たす方式である。このためには、ベクトルレジスタとマスクレジスタを装備し、ベクトルレジスタにはベクトルデータを置いて、メモリ負荷を軽減したり、パイプラインへ高速にデータを送れる必要がある。

ベクトルデータの演算自体の高速化とデータをメモリからレジスタにロードしたり、ストアしたりするオペレーションの高速化も欠かせない。現在のスーパーコンピュータでは、パイプライン化されたベクトルデータのロード／ストア機構を複数個載せて転送速度を上げている。

②　ベクトル化

科学技術系の計算処理プログラムでは、配列データに対して同一の演算を繰り返し実行させる場合が多い。これにはいわゆるDOループが用いられている。このDOループ内でのベクトル演算に注目して高速化を図る方法をベクトル処理方式という。この機構をできるだけ使うために2通りの工夫が行われる。

一つは、プログラムのソースコードの中からベクトル処理化が可能な部分をできるだけ多く見つけ、ベクトリ処理オブジェクトモジュールを作り出す必要があり、これはベクタライズコンパイラが行う。もう一つは、スカラ演算部分をベクトル演算に変換し、さらにベクトル化率を上げることである。

2．多重プロセッサシステム

プロセッサの処理能力をあげる技術としては、一つには単一プロセッサの能力向上と、もう一つは複数プロセッサの協調動作により全体の性能をあげる方式とがある。単一プロセッサ技術の高度化については以前から限界があると指摘されてきた。集積技術、実装技術にお金がかかりすぎるという事態である。しかしながら、米国が複数プロセッサ技術で高速化を図ってきたのと対照的にむしろ日本では単一プロセッサの製造技術により高性能化を果たしてきた歴史がある。まだまだ、単一プロセッサ技術にも技術向上の余地はある。

一方、多重プロセッサ技術は、ここ数年日本でも盛んになってきており、今後はこの方式への技術革新が中心となっていこう。この方式では、多数のプロセッサが交互に通信を行いつつ並列動作を行う。技術的な焦点は、この方式が、プロセッサ間の通信量が増大化し過ぎたり、その仕掛けに適合するアルゴリズムやソフトウェア開発が容易でない点である。プロセッサ間の結合は、何種類か有りそれぞれ特徴を持っている。

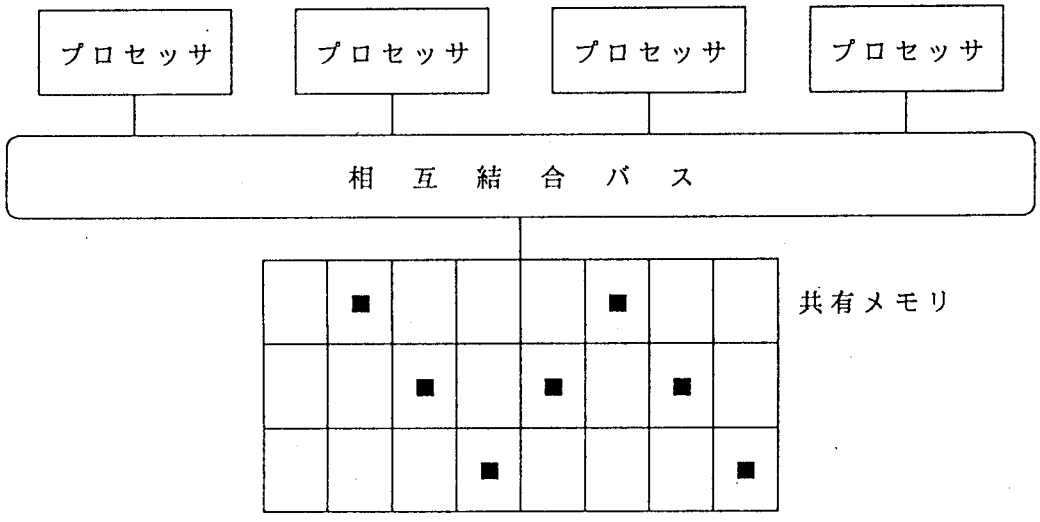
多重プロセッサでは、負荷や機能の分散による高速化、信頼性の向上、プロセッサ数の調整による適正な性能を引き出す目的がある。マルチプロセッサにおいて各プロセッサの結合方式は“密結合”と“疎結合”とに分類できる。

（1） 密結合多重プロセッサシステム

密結合多重プロセッサシステムでは、主メモリやディスク装置を全てのプロセッサが共有してアクセスできる。各プロセッサの処理が円滑に進められるためにプロセッサ間で相互に情報を交換し合うことを基本としている。このプロセッサ間の通信は、送信プロセッサ側が共有メモリの特定場所に情報を書き、受信プロセッサがこれを読んで行われる。

この方式では、複数のプロセッサが同時に同一メモリにアクセスすると競合が発生する。この競合状態発生時にアクセス待ちが多くなると、全体的には並列処理の効率が低下してしまう点に欠点がある。

図表Ⅴ－1　密結合多重プロセッサシステム



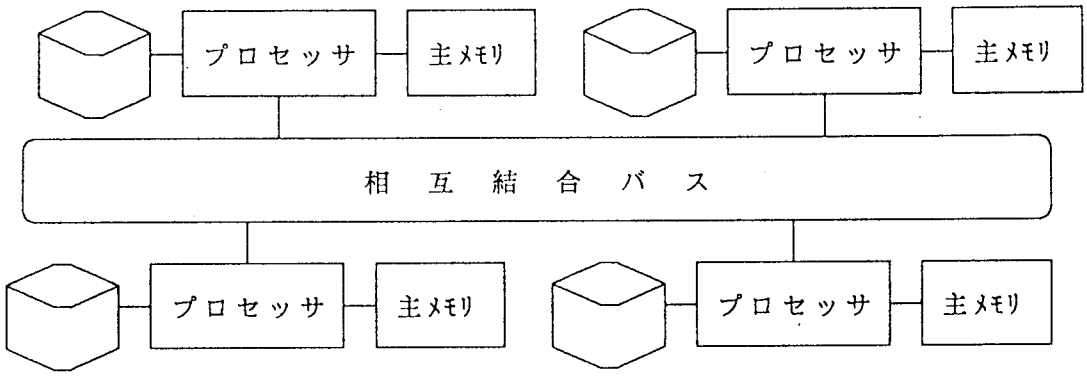
（2） 疎結合多重プロセッサシステム

疎結合多重プロセッサシステムでは、各プロセッサが固有のメモリを持っている。プロセッサ間での共有メモリは持たない。各プロセッサ間での共通情報の交換は送信プロセッサがメッセージを作り受信プロセッサが受け取ることによって行われる。

プロセッサ間通信に関しては、メッセージを第三者のプロセッサが中継を行う場合もある。メッセージには宛名が付加されパケット形式で送受信される。各プロセッサは、独立して通信処理を行う分散処理システムである。

この方式ではメモリの競合はおきないが、中継プロセッサを介す場合にはそのためのオーバヘッドが問題となる。この結合では、通信路が複数個あるためプロセッサ数が大きくとも通信路空き待ちは少ないであろう。したがって、プロセッサ数が多いほど密結合よりも有利となる。通信は、各プロセッサが自分で制御するため、故障時の復旧、プロセッサの増減が容易である。

図表Ⅴ－2　疎結合多重プロセッサシステム

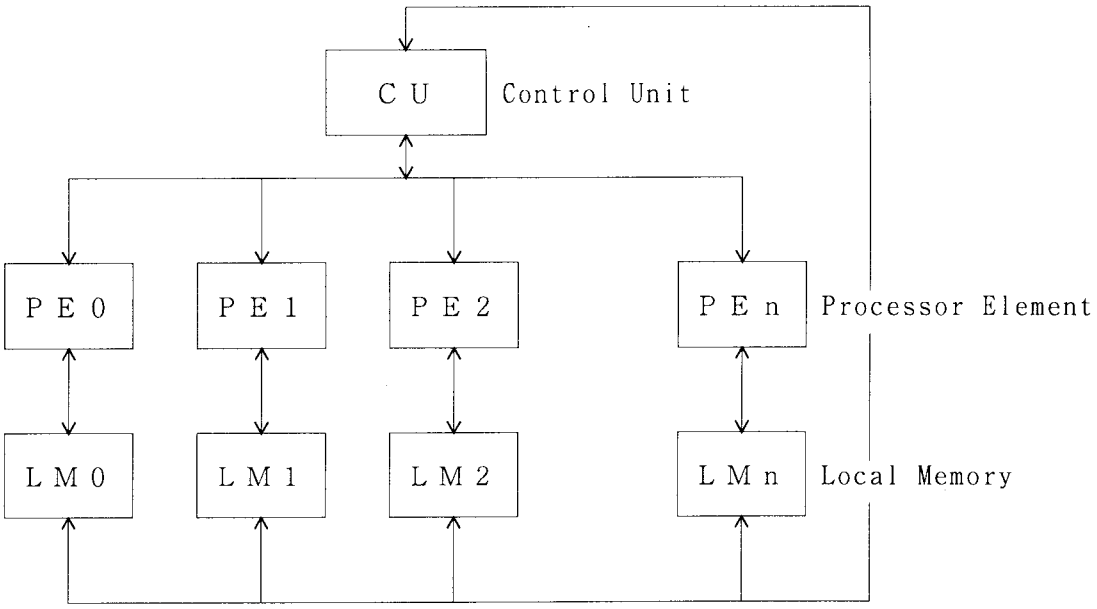


上図において、主メモリもディスク装置もプロセッサに従属している場合完全疎結合、ディスクは各プロセッサ間で共有するような場合は不完全疎結合と呼ぶ。

（3）　アレイコンピュータシステム

一つの制御ユニット（CU）のコントロールにより配下の複数個のプロセッサに演算指示が送られる。それらのプロセッサはデータは異なるものの同じ種類の演算を一斉に行う。例えば、巨大配列同士の演算を高速に行うのに効果がある。

図表Ⅴ－3　アレイコンピュータシステム構成

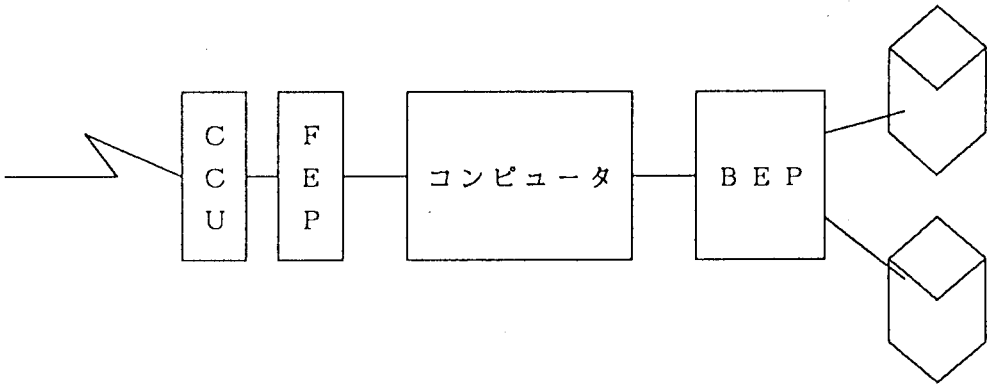


［出典：情報システムハンドブック編集委員会編；「情報ハンドブック」、培風館、1989年、Ｐ４－326］

（4） タンデム多重プロセッサシステム

複数台のプロセッサを直列に接続した多重プロセッサシステムの一種である。基本的にはそれぞれのプロセッサ毎に特定の処理機能を割り当て、付加を分散することを目的とする。システムの構成例としては、メインフレームコンピュータにおけるトランザクション処理で、端末処理を専門とするFEP（Front End Processor）とトランザクション処理を行う本体、およびデータベースアクセスを専門に行うBEP（Back End Processor）とをつないだようなシステム、あるいは、米国タンデムコンピュータ社が商品化しているGURDIAN90は無停止を目標に、1秒毎に各プロセッサが信号を発し、他のプロセッサで信号を検知し、正常かどうかを調べる。返信号がこないときはダウンしたものとみなし他のプロセッサに処理の切換を行うシステムである。

図表Ⅴ－4　タンデム結合多重プロセッサシステム構成



3．コンピュータの信頼性

情報技術が今日のように企業、行政、社会、家庭にこれほど浸透している現在、我々の社会活動、経済活動、産業活動は大きくその力に依存してしまっている。家庭における娯楽のツール的な位置づけにあるものはともかくも、システム化されたものの信頼性は機能の高度性よりもずっと重要視されるべき要素となっている。

現在の情報通信技術に支えられた活動基盤は、電力における停電を最小化することと同等くらいの信頼性の保証が求められている。以下で高信頼性を実現する技法について考えよう。

（1） RASIS

① RASISについて

システムの信頼性に関連して、総合的に信頼度を定量的にとらえる概念の一つとして［RASIS］がある。

図表Ｖ－５　高信頼性技術の構成要素

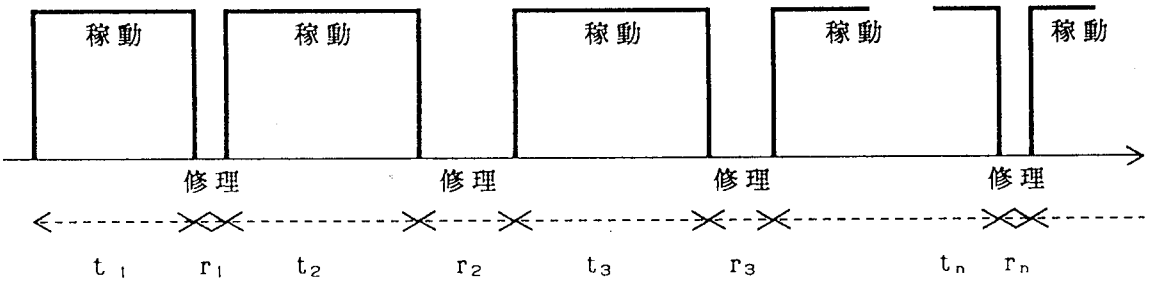
システム信頼性の構成要素	概　　要
信頼性（Reliability）	システムが、規定された期間中に、要求された機能を一定の条件下で果たし続ける能力 ハードウェア部品の高品質性、部品の定期点検・交換、ソフトウェアの耐久試験による高品質性が求められる。
可用性（Availability）	システムの一部に故障が発生していても、利用者からみると全体的には正常動作をできる度合い。 ハードウェアのデュプレックス構成、デュアル構成化
保全性（Serviceability）	システムに故障が生じたときに保守が容易になされること。保守性ともいう。 エラー情報が得られること、自己診断が可能なこと、データベースなどの復旧などがなされることが求められる。
完全性（Integrity）	情報の入力から出力に至り、データの内容、データ項目間の正当性、整合性が保たれる度合い。 故障の回復後などは当然、システムの状態は故障前と同一であることも求められる。
機密性（Security）	システムに対する不正な侵入やデータの破壊を防ぐ度合い。

②　RASISの評価尺度

a．MTBF（Mean Time Between Failures）：システム全体の平均故障間隔

前回の故障が復旧し、次の故障で停止するまでにシステムが稼動している時間の平均値

図表Ｖ－６　システムの稼動と故障による停止



$$MTBF = \frac{t_1 + t_2 + \cdots\cdots t_n}{n} = \left\langle \frac{\text{稼動時間}}{\text{故障回数}} \right\rangle$$

b．MTTR（Mean Time To Repair）：平均故障修理時間

$$MTTR = \frac{r_1 + r_2 + \cdots\cdots r_n}{n} = \left\langle \frac{\text{修理時間}}{\text{故障回数}} \right\rangle$$

c．稼動率（Availability）：ある期間中にシステムが稼動している時間の比率

$$A\ (\text{稼動率}) = \frac{MTBF}{MTBF + MTTR}$$

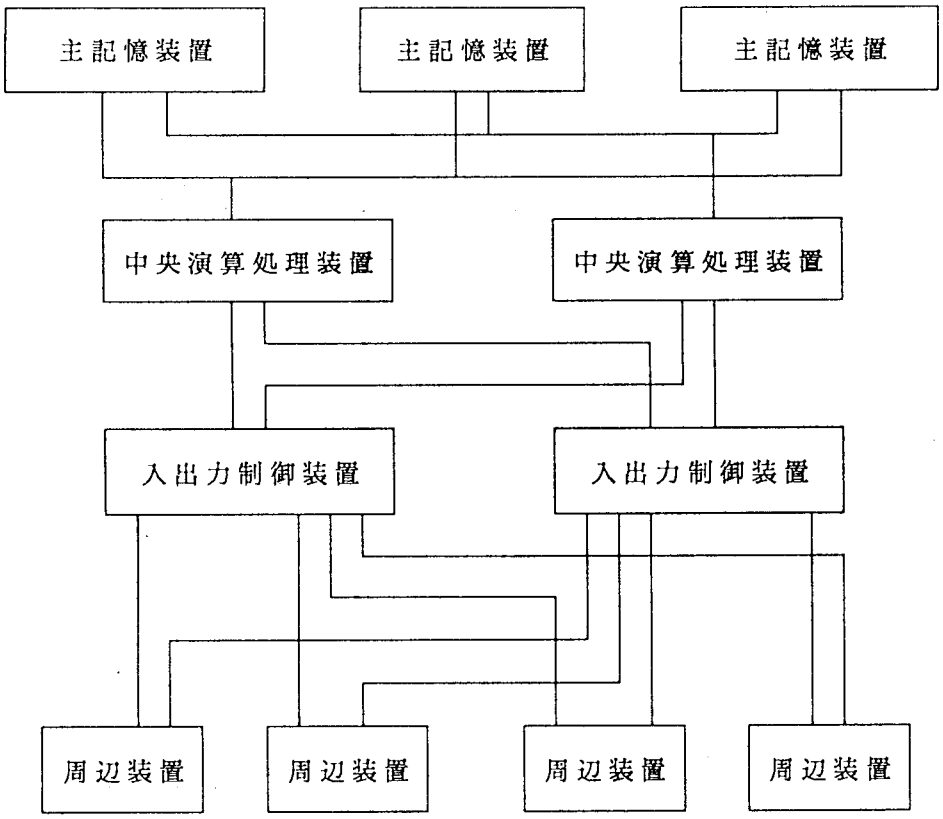
（２）　信頼性を高める構成

システムの信頼性を高めるには、最初に考えるべきは構成要素である各種装置の信頼性を上げることであり、そのためには個々の装置に使用される部品の信頼度の改善が第一である。しかしながら、個々の部品だけでは障害を完全に回避することは不可能であり、バックアップ機などを設置してシステムとして装置を多重化することにより信頼性の向上を図る必要がある。

基本は、障害を起こした装置をシステムから切り離し、システムを再構成して運転を続行させる考え方である。

① マルチシステム

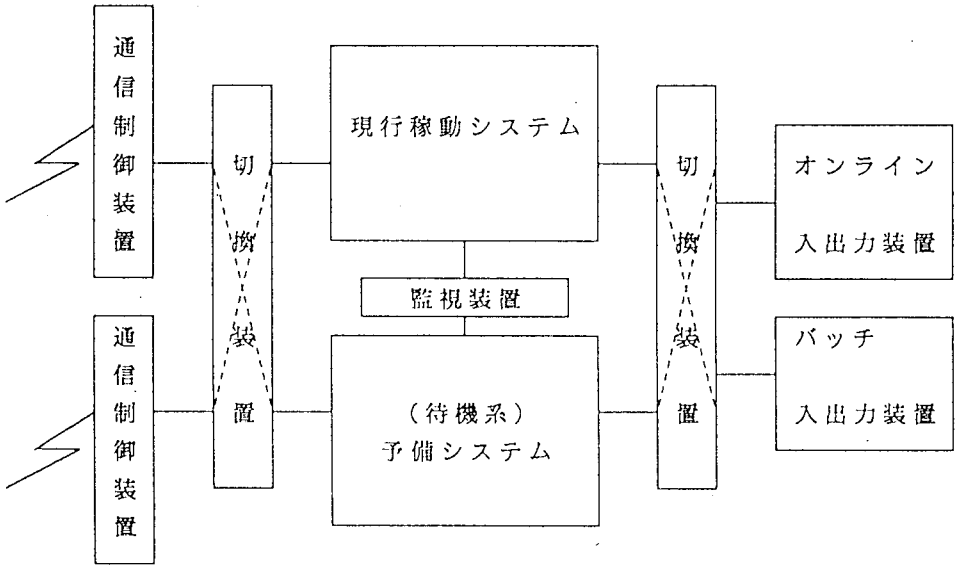
ハードウェア冗長化の典型的な例である。複数の中央演算処理装置が複数の主記憶装置、周辺装置を共有するシステム。演算装置の制御は一つのオペレーティングシステムで行う。どの装置に障害が発生しても処理能力の低下は避けられないものの残りの装置だけで処理を続行できる。



② デュプレックスシステム

2台のCPUによって待機型の冗長システムを構成する。例えば、1台をオンライン業務システムとして運用し、他の1台をバッチシステムとして用いる。オンライン業務システムに故障が生じるとバッチシステムをオンラインシステムに切換て稼動させる。バッチシステムの方はいわゆるバックアップ機（あるいは待機系）と呼ぶ。この待機系の運用の違いによってコールドスタンバイとホットスタンバイとに分類される。

図表Ｖ－７　デュプレックス構成のコンピュータシステム構成例



ａ．コールドスタンバイ

待機系は通常は他のバッチ的な業務を行っている。稼動系に障害が発生すると、その処理を中断し、システム立ち上げを行ってから稼動系の処理を再開する。このため、代行開始までに時間差がでる。

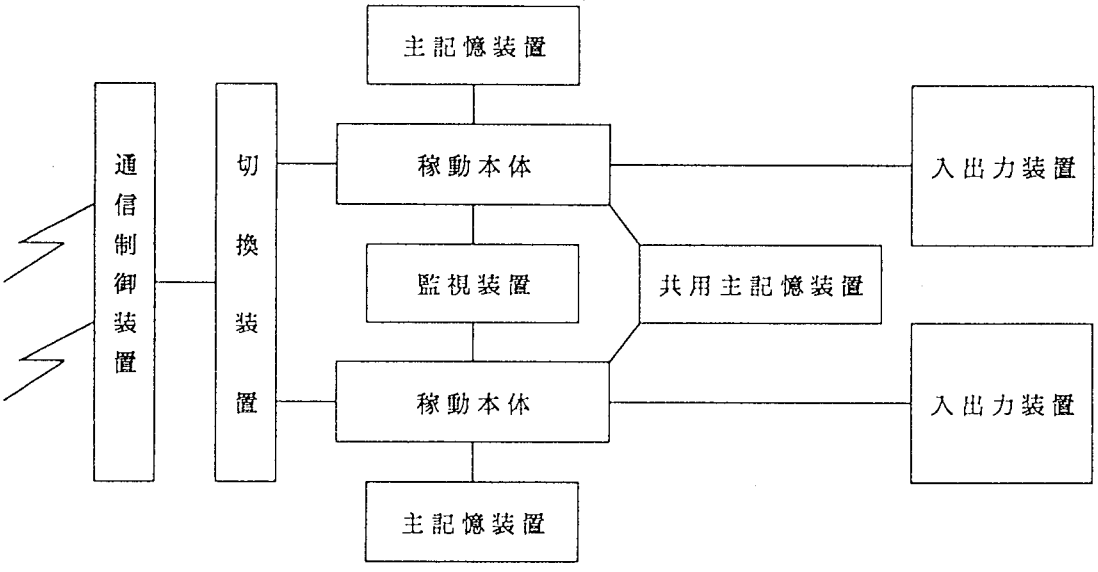
ｂ．ホットスタンバイ

稼動機に障害が発生したときに、待機系では稼動機の処理を速やかに引き継げるように準備をしておく（例えば稼動機のIPLなどは当然終わっている。）。待機はリカバリ情報をもとに復旧処理を即行う。

③　デュアルシステム

全く同じ二つのシステム構成で一つの業務を同時に処理させ、結果を照合比較して正常動作をしているか確認するシステム。非常に高度な信頼性を求められるシステムに採用され、いずれかの系に故障が発生しても中断なく他の系で運転が継続できる仕組み。いわゆる並列型の冗長システム構成である。このシステムは、緊密な情報連絡を必要とするため、同一場所に設置されなければならない。

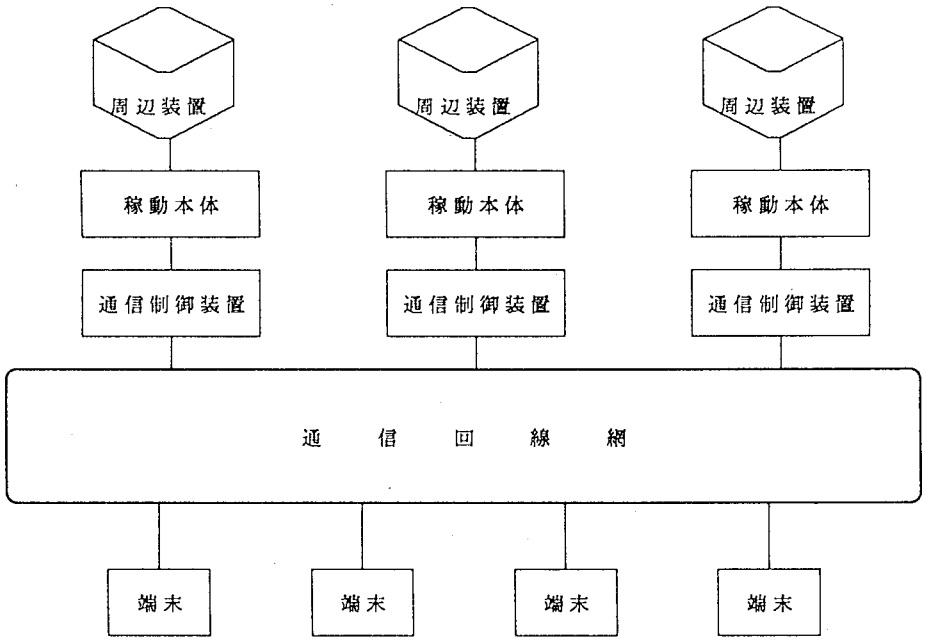
図表Ｖ－８　デュアル構成のコンピュータシステム構成例



④　分散処理構成

デュプレックス構成の考え方を拡張し、切換装置の機能を通信回線網と接続する通信制御装置に持たせ、分散処理システム構成をとって稼動性を高めようとするもの。

図表Ｖ－9　デュプレックス構成の発展型分散処理システム

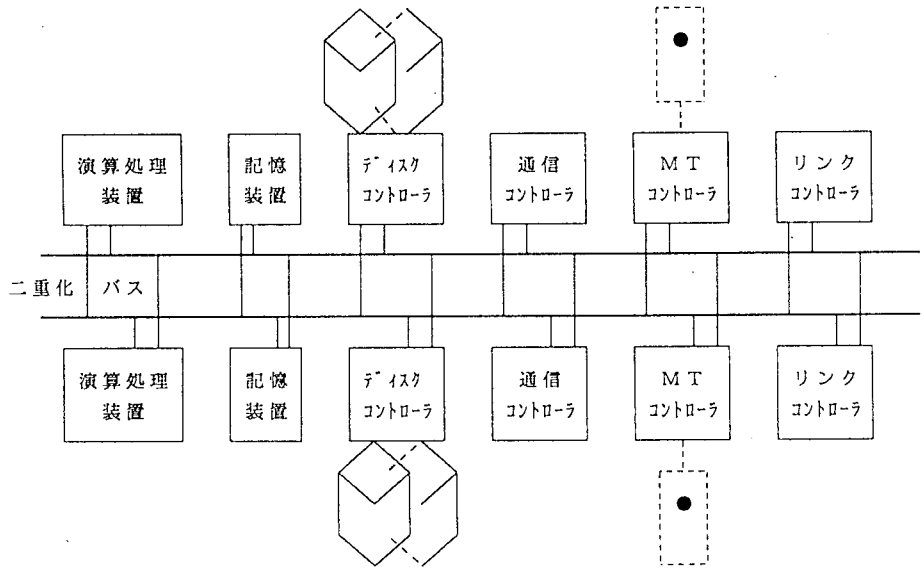


（3）　無停止コンピュータ

限りなく連続運転が可能となるよう設計されたコンピュータであり、システムの一部の要素が故障しても自動的に障害が検知され、その切り離しが行われ、代替装置へ切換が行われることによって支障を来さず業務を続けられる。このコンピュータは信頼性や性能に最も要求の厳しい航空・宇宙の分野で発展していった。1980年代には、銀行のATMやクレディットカード、株式会社の売買などデータベースをリアルタイムに検索したり、紹介したり、また更新をしたりするシステムでフォールトトレラント能力が要求されるようになった。

商品化された代表例としてストラタス社のシステムが有名であり、マイクロプロセッサを多用し、接続される全てのハードウェアの完全な二重化を実現している。

図表Ｖ－10　無停止コンピュータのシステム構成例

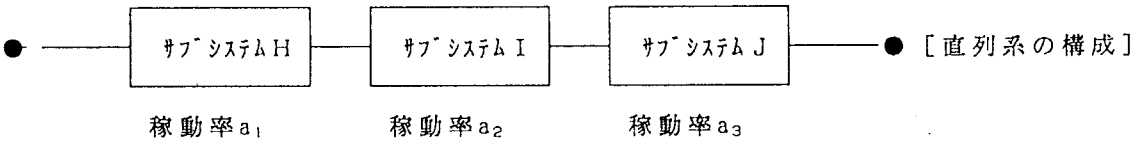


（4） 稼動率の計算

コンピュータシステムを構成する単位（これをサブシステムと呼ぶことにする。）の稼動率が分かっていれば、コンピュータシステム全体における稼動率が計算できる。これは、コンピュータシステムの高性能性の実現に非常に重要である。

① 直列系システムの稼動率

直列系システムでは全サブシステムが稼動していることを前提としており、下図のシステムの稼動率は$A = a_1 \times a_2 \times a_3$、また、n個のサブシステムから成る直列システムの稼動率は、A（直列システムの稼動率）$= a_1 \times a_2 \times a_3 \times \cdots \times a_n$となる。



直列系システムではサブシステムの数に比例して稼動率が低下する。また、信頼性の低いサブシステムが存在するとシステム全体の信頼性も低下してくる。

② 並列系システムの稼動率

並列系システムでは、いずれかのサブシステムが稼動していれば、システムとして稼動していると見なされる。下図左のシステムの稼動率は$A = 1 - (1 - a_1) \cdot (1 - a_2)$、下図右のシステム稼動率は、

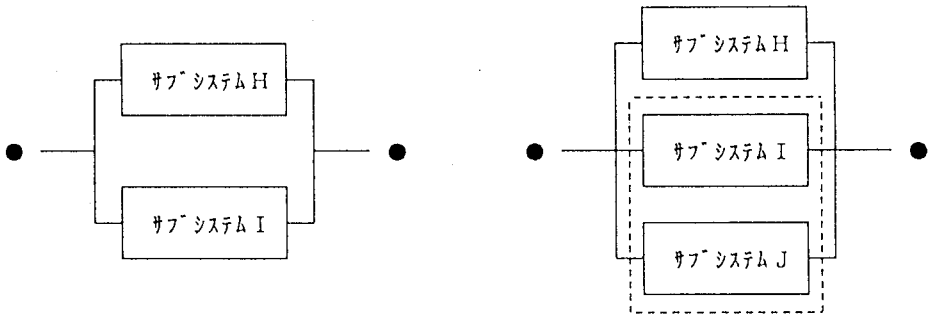
$$A = 1 - (1 - a_1) \times [1 - \{1 - (1 - a_2) \times (1 - a_3)\}]$$
$$= a_1 + a_2 + a_3 - a_1 \times a_2 - a_2 \times a_3 - a_1 \times a_3 + a_1 \times a_2 \times a_3$$
$$= 1 - (1 - a_1) \times (1 - a_2) \times (1 - a_3)$$

となる。

また、n個のサブシステムから成る並列システムの稼動率は、

A（並列システムの稼動率）$= 1 - (1 - a_1) \times (1 - a_2) \times (1 - a_3) \cdot\cdot \times (1 - a_n)$

となる。



③ N out of M系システムの稼動率

並列に接続されているM台のサブシステムのうちN台のサブシステムが稼動していれば、正常な運転とみるシステム構成をN out of M系システムと呼ぶ。このシステムの稼動率は、N台のサブシステムが稼動している確率からはじめ、N＋1台、・・・M台のサブシステムが稼動している確率を求めてそれらを加算すれば求められる。

同一の稼動率のサブシステムがM台並列に接続されている場合のN out of Mシステムの稼動率は、

$$A\ (\text{N out of M システムの稼動率}) = \sum_{i=N}^{M} {}_M C_i \times h^i \times (1 - h)^{(M-i)}$$

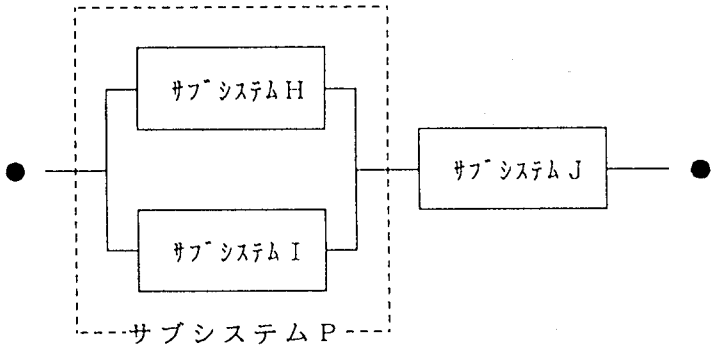
④ 直列系と並列系が混在するシステムの稼動率

a．並列系システムが他のサブシステムと直列系を構成する例

A（並列システムの稼動率）＝1－（1－a_1）×（1－a_2）

A（直列システムの稼動率）＝p_1 ×a_3

A（混在システムの稼動率）＝｛1－（1－a_1）×（1－a_2）｝×a_3



並列系システムが他のサブシステムと直列系を構成する例

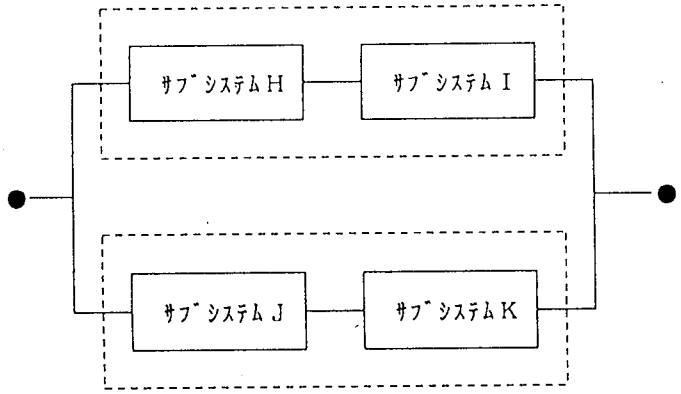
b．直列システムが並列系を構成する例

A（直列システムの稼動率）＝a_1×a_2

A（直列システムの稼動率）＝a_3×a_4

A（並列システムの稼動率）＝1－（1－p）（1－q）

A（混在システムの稼動率）＝1－（1－a_1×a_2）×（1－a_3×a_4）



4．集中から分散へ

（1） 集中システムの長所

語り尽くされてきたこととして、集中システムの利点は、全てのコンピュータ資源を中央のコンピュータシステム近隣に直結させ、一つの基準と処理ソフトウェアによってその運用と管理を行う方式である。このタイプのシステムはこれを意識して開発されてきたのではなく、もともとコンピュータ同士を結ぶ通信処理が発達しておらず、また、分散コンピュータが存在しなかったこともあり、必然的に集中化されていた経緯がある。

分散コンピュータが出現した後もしばらくは、中央コンピュータと分散コンピュータとの論理的なつながりは少なかった。しかし、データの発生地点や量がどんどん膨れる中、通信コストの削減、データベースの分散化の必要性が高まってくるとともに、分散処理の利点を生かしたシス

テムが次第に増えるようになった。

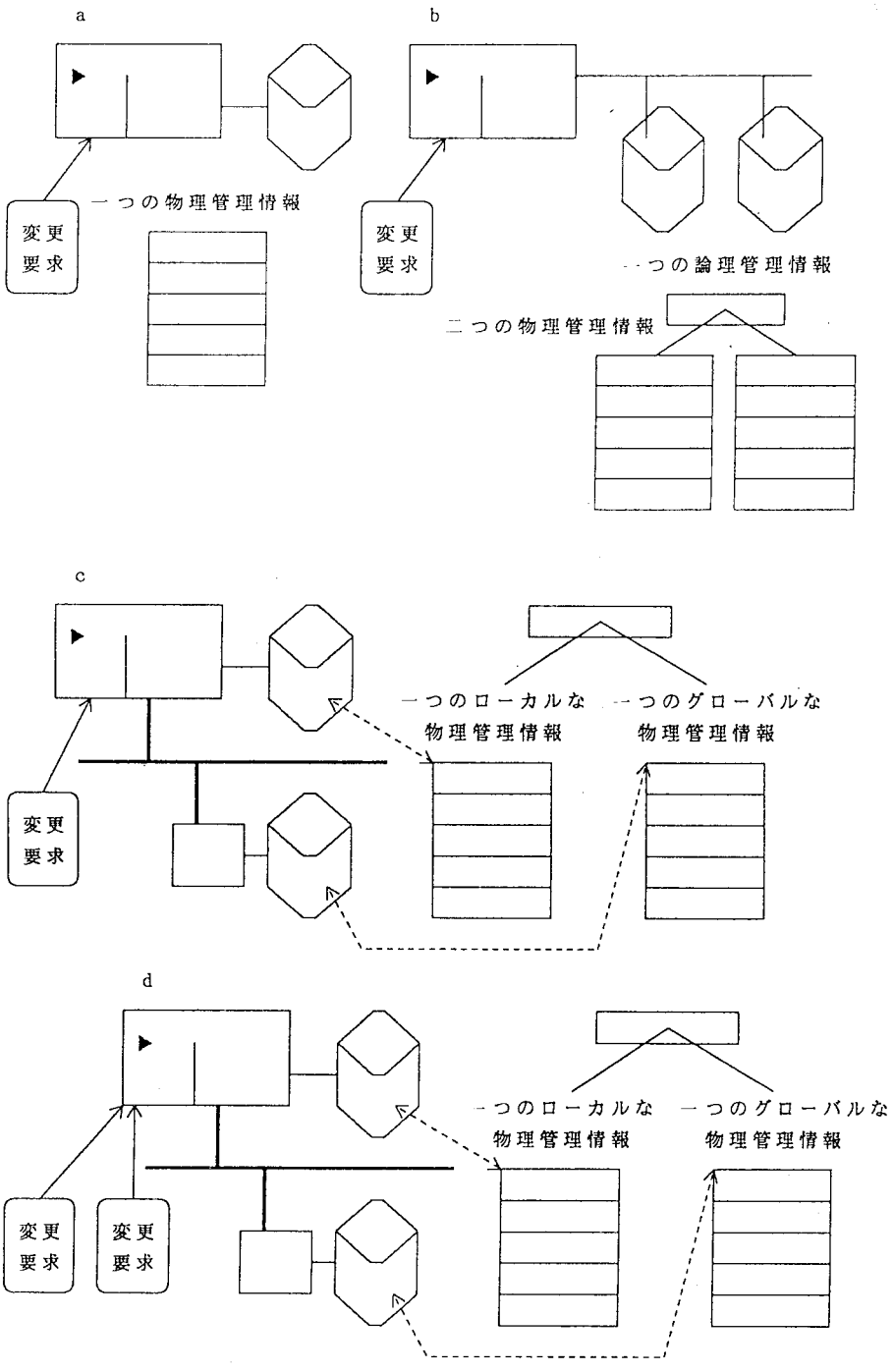
分散処理技術の高度化する中で、中央コンピュータの役割が次第に小さくなり影が薄くなってきたように思えるが、それでもなおかつどういう局面を集中処理で、どこを分散システムとするかの戦略作りは相変わらず重要である。

以下で、集中システムの長所について考えよう。

① データの一貫性保持の容易性

データの一貫性を保つためには、データに対する変更要求をどう裁くか、論理的なだけでなく物理的な整合性をどう維持するかが重要である。下記のａ～ｄのシステムを比較すれば一目瞭然、ａあるいはｂが変更管理が容易である。ｃ、ｄでは物理的に離れたデータの管理と同時変更に関する制御が容易ではないことがわかる。

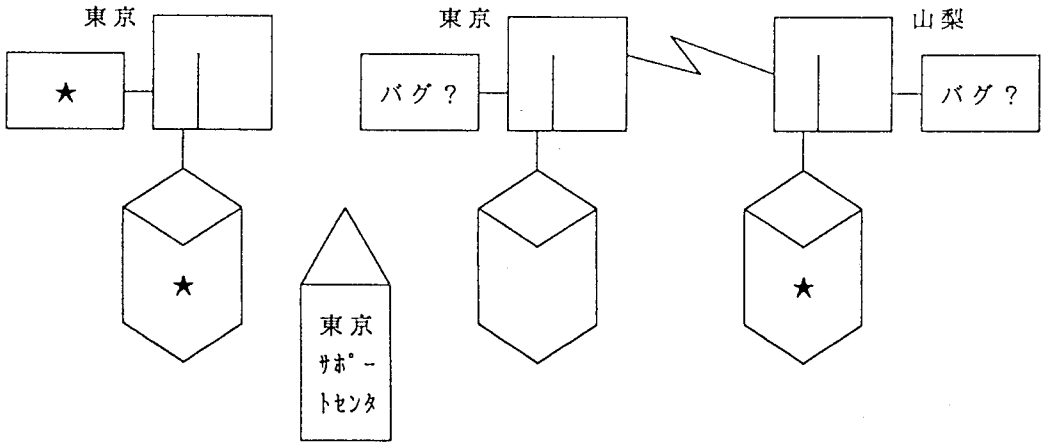
図表Ⅴ－11　データの管理



② 故障に対する対応の容易性

ハードウェアの故障に対する修理、ソフトウェアの障害に対する原因特定の時間と対応策に関して、明らかに集中処理の方が速い対応が可能である。

図表Ⅴ－12 ハードウェアの故障、ソフトウェアの障害



③ 資源の効率的な運用・管理

資源を多くの同時利用者により効率よく稼動させることができる。データのバックアップ自身も物理的に一箇所で行える。拡張計画なども容易に行えるなど多くの利点がある。

a．コンピュータ資源を１箇所に集中でき、スペース的に得である。

b．コンピュータ資源を１箇所に集中でき、資源不足が起こったら比較的単純に追加すれば良い。

c．コンピュータ資源を１箇所に集中でき、オペレータの少人数で済む。

d．コンピュータ資源を１箇所に集中でき、不正使用に対する防御が比較的楽である。

e．コンピュータ資源を１箇所に集中でき、ソフトウェアも１種類で済む。

f．コンピュータ全体の制御のためのソフトウェアを比較的軽く作ることができる。

（２） 集中システムの問題点

メインフレームコンピュータは利用ニーズの増加、大型化に支えられ、また情報技術の革新性の高さによりハードウェア的にもソフトウェア的にも高成長を遂げてきた。従来のハードウェア技術では、「コンピュータの性能とコストの関係は、性能がコストの自乗に比例する」というグロッシュの法則が成り立っており、機能と性能を集中させることにより大きな利益を得てきた。

しかしながら、一度限界にぶつかり始めると、大規模化したOSの機能追加や保守費用の膨大化、多種多様なコンピュータ操作ニーズの要請により分散管理の優れた点に次第に技術が注がれるようになっていった。集中システムにはいくつかの問題点が出てきた。

① システムの柔軟性

資源、運用、管理の集中化は、一つのアンバランスが生じただけでもシステム全体には大きな影響を与える。

a．拡張性の低下

拡張それ自身が全く不可能になるケース、拡張してもパフォーマンスが落ちすぎて使い勝手が低下するケース、拡張できたとしても急にコスト高になるなど負の面が大きくなってくる。

b．故障による波及

部分的な故障が全体に大きな悪影響を及ぼす。ときとして、システム全体の停止が必要と

なる。故障対策のためのソフトウェア機能が膨れ上がり、パフォーマンスが落ちる。

c．変更の自由度の喪失

システム構成の大きな変更に対してＯＳやアプリケーションが対応できなくなるほどのソフトウェア改造が必要になることがある。

② ソフトウェア開発と維持

ソフトウェアコスト増大化の要因は、

a．ソフトウェアの量が多くなると開発の歴史の長いソフトウェアは、複数人の手が入っており、ソフトウェア開発技法の変化する中で、機能追加や不具合の改修に相当の労力をつぎ込まなければならなくなる。

b．情報の複雑性、多量性は、ソフトウェアの処理論理を複雑にし、処理性能を低くするとともに、テスト時間も増大化する。

c．ソフトウェアの改良や新規導入において、本番システムを長時間止めておくわけにはいかず、オンライン試験を行う必要があるが、非常に困難なことである。

バーチャルマシン（IBMのVMなどに相当）などのOSであれば、ある程度本番にほぼ近い環境でのテストは可能であるが、本番試験をゼロにすることはできない。

d．ソフトウェア開発の変化

利用者が手元でコンピュータを操作する時代になり、ユーザ環境の改善のために中央コンピュータOSの負荷が次第に高くなっていったが、そこで処理する必然性をとっくに超えている。

③ 信頼性

集中型コンピュータでの信頼性を上げるために、システムの二重化、多重化して冗長性により対処しているが、ハードウェア費用が膨大になったり、深夜運転時の影響が日中に出てくるなどマイナス面が多くなっている。

（3） 分散戦略の利点

① 処理効率の高さ

コンピュータの分散により、計算処理の負荷分散が行われ、システム全体としての処理能力が上がり得る。

② 経済的な面

グロッシュの法則が崩壊して、分散化により、開発や運用の局面で経済性が発揮されるようになった。

a．開発コスト

システム開発の範囲が限定され、システム開発の生産性の高まりにより開発コストを削減できる。また、分散化により機能別開発が可能となり、並行開発などにより納期短縮、開発要員の平均化などが図れコスト削減につながる。

b．運用コスト

分散地点でのローカルなデータ処理により通信コストを削減できる。そもそもハードウェアが分散しているため、システム自身が冗長となっており、運用コストを落とせる。

③ 拡張性の向上

a．処理ニーズの高いところへの部分的な拡張が可能である。

b．機能の追加や変更要求に対して、他の分散マシンに影響を与えずに実施できる。

c．分散地点固有のニーズに対応できる。

④ 信頼性の向上

システム全体でみると、ある分散地点で故障があっても、一部の機能は低下するものの、他の地点への影響は少ない。また、災害や不正使用に対して、被害がローカル化されるので全体

的な被害は抑えられる。但し、ある分散地点での不正使用などが発見されないままホストコンピュータや他の分散地点へのデータ変更を実行すると、全体に影響を及ぼすことも有り得ることに留意しなければならない。

5．主要用語

アレイコンピュータシステム	MTTR	MTBF
稼動率	MIPS	FLOPS
タンデム結合	デュアル構成	デュプレックス構成
無停止コンピュータ	パイプライン	フロントエンドプロセッサ
バックエンドプロセッサ	多重プロセッサシステム	CISC
RISC	RASIS	ギブソンミックス
命令ミックス	Whetstone	スーパーコンピュータ
集中システム	分散システム	