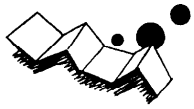
解　説

システム障害回復技法†



久保隆重†† 堀越　彌†††

1. はじめに

計算機システムは適用分野の拡大，利用の高度化を背景に大型化複雑化している．この傾向が強くなればなるほど，システム障害の社会的影響が大きくなり，より高水準のシステム信頼度が要求される．一方で固有の故障率をもつコンポーネントの数と種類が増加することにより，システムとしての信頼度は低下し，高信頼化に対するニーズと実現レベルとのギャップは拡大を余儀なくされる．

このギャップの拡大を防止除去するための技術としていわゆるフォルトトレラント技術（耐故障性強化技術）があり，計算機システムの出現以来，特に大型システム発展のための必須の技術として発展を続けている[2)~8)]．

本稿では汎用大型計算機システムのハードウェア，ソフトウェアの障害に対するシステム（もしくはサブシステム）レベルの回復技法に焦点をあてて，比較的最近の技法とその問題点等について解説する．

2. システム回復技法の現状と動向

2.1 システム回復の基本概念

システム回復の目的は，システムにおいて発生しうる障害のできるだけ多くの要因に対して，システムへの影響度を最小化することである．一般的に影響度はシステムサービスへの影響範囲と回復時間で評価される．システム回復の焦点は，ある率で障害が各種のコンポーネントにおいて発生する時に，それによるシステムのサービス停止をいかに回避するかというところにある．最近の大型計算機の RAS 技術も，この命題のもとに改善されており，特にその中心はシステム回復技法 (System Recovery) である[16)]．

† System Recovery Techniques by Takashige KUBO (Systems Development Laboratory, Hitachi Ltd.) and Hisashi HORIKOSHI (Central Research Laboratory, Hitachi Ltd.).

†† (株)日立製作所システム開発研究所
††† (株)日立製作所中央研究所

（1） 障害の要因とシステム回復のアプローチ

大型計算機システムにおいて発生しうる障害要因は多種多様である．同一の種類の障害が発生したとしても，その障害コンポーネントの機能的役割が動的に変化していることを常とする汎用計算機システムにおいては，システムへの影響度は同一にはならないので単にコンポーネントの種類と故障モードによる分類だけでは不十分であるが，まずは障害要因をコンポーネントの種類別に分類しておくことは一般的である．

また回復のためのアプローチを考慮するためには，障害の特性による分類が有効であり[9)]，**表-1** に示すような分類が用いられる．

（2） システムへの影響

マルチプログラミングを基本とする計算機システムに対する障害の影響度は，システム設計の立場からはユーザへの影響度の順に次のように分類される[14)]．

- 影響なし：障害からの回復に成功し，システムサービスに何らの悪影響をもたない．
- 機能縮退により罹障ユーザプログラム続行：障害の発生要因を切り離すことによりユーザプログラムを続行する．
- 罹障ユーザプログラム異常終了：罹障ユーザプログラム（タスク/ジョブ）のみ異常終了する．
- サブシステム異常終了：システム全体が複数のサブシステムから構成されていて，罹障サブシステムのみ異常終了する．

表-1 障害の分類と回復のためのアプローチ

性　格	持続時間	発生コンポーネント	主たるアプローチ
動作障害	過渡（1時）障害（間欠障害）	ハードウェア	ハードウェア冗長性（動的，静的） ソフトウェア冗長性 時間冗長性
	永久（固定）障害	ハードウェア	ソフトウェア冗長性 ハードウェア冗長性
設計障害	永久（固定）障害	ハードウェア ソフトウェア	設計検査 ソフトウェア冗長性

○システム異常終了：システムの全サービス停止となり，いわゆるシステムクラッシュとかシステムダウンとよばれる．

これらの他に異常を検出しえなかった場合は，その時点では影響度は評価不可能であるが，データロスなどを伴うとユーザにとってはシステムクラッシュ以上の悪影響を与えるケースがあり，最も影響度の高いケースとして分類しておかねばならない．

(3) システム回復の基本機能と目的

システム回復の基本機能は障害の検出と回復である．ある特定の種類の障害を対象としたときのシステム回復の基本フローを **図-1** に示す[7]．

システム回復の目的は，あらゆる障害に対して 図-1 に示した η_1, η_2, η_3 を1に近づけることであるということができる．

ここで注意すべきことは，従来システムの信頼度は障害の発生頻度 x により測られてきたが，システム回復技法の強化により，障害が発生してもほとんどの場合システムクラッシュに直結することはなくなってきている．この意味でシステムクラッシュの頻度または間隔をシステム信頼度の最も重要な尺度として使用するようになってきているが，これでも種々の曖昧性が残る．これを除去するため，すべての IPL (Initial Program Load) から，Scheduled IPL を除いた Unscheduled IPL の頻度を尺度とすることも多い．

2.2 システム回復技法の現状と動向

システム回復のアプローチにはハードウェア冗長性，ソフトウェア冗長性，時間冗長性があり，また，数十万のゲート規模の大型 CPU や，数百万ステップにも及ぶ超大型 OS における種々の障害を対象とした回復技法の体系は概観することすら容易でない．特にソフトウェア冗長性の中心的存在である超大型 OS の回復終了管理プログラムはそれだけでも 100 K～200 K ステップに及ぶ大規模ソフトウェアであり，その機能も多様である．

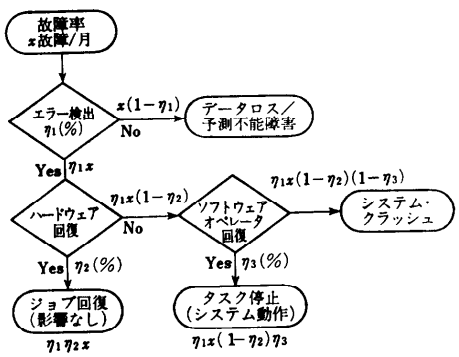


図-1 システム回復の基本フロー

ここでは 3.5 世代および 3.75 世代システムのシステム回復技法の体系 (**表-2**) とそのシステム信頼度の実績データの例 (**表-3**) で現状の説明にかえさせていただく[1)~16)]．なお信頼度データは日米のオンラインシステムから収集したものである．また用語は，主として IBM 社の大型システムで使用されているものを用いた．

システム回復技法の最近の特徴は次のものである．

(1) システム回復機構の階層化によるシステム化

可用性の向上のためには，システムのできる限り下位レベルでの回復が，障害の早期検出や影響度の最小化に最も有利であることと，また上位レベルでの回復がより機能的な回復手法が用い得るという考え方が組み合わされてシステム階層の各レベルに回復機構が配置され，それらが階層化された回復システム (Recovery Hierarchy) としてシステム化が進展している[7),14)]．

(2) ソフトウェアの設計障害[9)]に対するシステム回復

システムクラッシュの原因として，ハードウェア以上にソフトウェア特に制御プログラムと運用操作上の要因が増大している．このため，ソフトウェアの不良に対するシステムクラッシュの回避策が色々と組み込まれるようになっている[14)]．

(3) マルチプロセッサシステムの回復機能の強化

オンライン・システムを中心とする高い信頼度の要請に応えるため密結合，疎結合マルチプロセッサシステム構成用のシステム回復技法の成功率を高める方式が実用化されてきている[11),14)]．

(4) 多重化構成の利用度増大

障害検出率や回復率を高めるため，VLSI, LSI 技術によるモジュール・レベルの二重化等の多重化構成の利用度が増大している[15)]．

以上は主な特徴といえるが，**表-2** に示した従来からの種々の障害回復技法も対象とする障害の適用条件の緩和や影響度を最小化するためそれぞれ精緻化され，着実に拡張がなされていることも注目しておかねばならない．

これらの特徴をもつ種々レベルの障害回復技法は，**図-2** のように構成される．図中の下方向の矢印は各レベルの回復に失敗した時にたどる過程を示し，上方

表-2　汎用計算機システムのシステム回復技法の体系

障害の種類	ハードウェア障害		ソフトウェア障害	
実現手段	ハードウェア	ソフトウェア	ハードウェア	ソフトウェア
誤り検出	パリティチェック 制御部シーケンスチェック，etc ⊙二重化による照合チェック (回路，モジュール，装置レベル)	時間監視機構 MIH (Missing Interrupt Handler)	記憶保護機構 ・セグメントプロテクション ⊙・多重仮想空間，etc プログラムチェック機構	各種妥当性チェック， インタフェースチェック，etc 時間監視機構
誤り訂正	エラー訂正コード ハミングコード サイクリック・チェックコード 命令再試行 入出力命令再試行 (コマンド)	回復管理サポート MCH (Machine Check Handler) CCH (Channel Check Handler) ⊙ACR (Alternate CPU Retry) (マルチプロセッサによる回復) エラー回復サポート ERP (Error Recovery Program) APR (Alternate Path Retry) DDR (Dynamic Device Reconfiguration)		⊙回復終了管理 タスク/ジョブ回復制御 タスク/ジョブ終了制御 ⊙階層型回復ルーチンサポート FRR (Functional Recovery Routine) ESTAE (Extended STAE)
再構成	自動縮退機構 バッファ記憶 (BS, BAA) TLB 制御記憶 (CS) 再構成機構 メモリ，チャネル	システム・ファイル2重化 システム・ファイルのチェックポイント システム領域一部2重化		
障害報告／記録	マシンチェック割り込み機構 拡張障害情報ログアウト	OBR (Out Board Recoder) SDR (Statistical Data Recoder)	プログラムチェック割込機構	各種メモリ・ダンプ ログ・レコーディング
診断	マイクロ診断機構 ⊙ステージ・トレーサ ⊙SVP (SerVice Processor)	EREP (Environment Recording Edit Prog.) OLTEP (On Line Test Executive Prog.) ⊙RETAIN (リモート診断)，FLT	PER (Program Event Recording)	各種トレース，トレース解析 ⊙IPCS (Interactive Problem Control System)
リスタート	リスタート・キー	Check Point/Restart	リスタート・キー	ジョブ管理サブ・システムリスタート ⊙DSI (Dynamic System Interchange) Warm Start/Cold Start

⊙: 3.75 世代システムに特徴的な回復技法

向は回復に成功したことを示している．

3. ハードウェアによるシステム回復機構

3.1 障害検出機構および誤り訂正機構

本体系障害の検出は障害処理 (Machine Check Handling) の第1歩であり，

- パリティチェック
- レシデュチェック
- 制御部のシーケンスチェック
- 正当性チェック
- k out of n チェック
- 時間監視チェック
- 2重化による照合チェック

等々，各種の方式がすでに広く用いられており，特に演算部にモジュールの2重化による検出方式を採用したものもあらわれてきた．検出機構の問題点はどの程度実現すれば必要かつ十分かという問題であるが(図-3 参照)，一般法則はなく，経験的に付加されているのが実状のようである．

しかし，いずれにしても障害検出機構と，引続く状態の凍結機構は（これは後のトラブル・シューティングに必要）大型計算機の方式の中でかなりの重みをもっている．

さらに一歩進んだ誤り訂正機構の分野では，主記憶装置のデータにいわゆるハミング・コードを付加して，1ビットのエラーを修正するのは，常識になっているが，記憶装置以外の分野ではまだ適用例はないようでむしろ，次の命令再試行の一環として自動訂正を行うようになる傾向にあると思われる．

3.2 命令およびコマンドの再試行機能

命令（入出力動作ではコマンド）の実行中に障害を検出した場合，動作を再試行して，正常処理を続行し

表-3 システム信頼度の例（第3.5世代システム）

System ID	A	B	C
Life Cycle Stage	Stable	New	Stable
System Configuration	1・L・UP*	1・M・UP*	2・L・UP*
Transaction Volume/Sec.	42 (Average)	3 (Peak)	22 (Peak)
Operating Hours/Day	24	11	10
Min. to switch to Backup Processor	6	NA.	NA.
Shortest Restart Time	2	NA.	NA.
Unsheduled Downtime (%)	0.4	0.3	0.1
MTBC	19.5	153.1	458.8
MTTR	4.4	25.0	38.1
% Crashes by type			
Hardware	41	27	53
Software	40	22	26
Operations	5	51	21
Others	14		
% Crash Time by Type			
Hardware	39	27	41
Software	37	28	37
Operations	5	45	22
Others	19		

*: backup processor,
NA: Not Available
MTBC: Mean Time Between Crash, MTTR: Mean Time To Repair
L: Large-scale System, M: Medium-scale System.

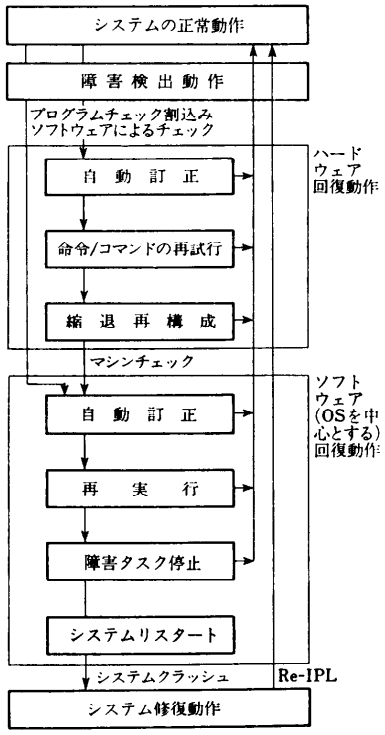


図-2 障害からの回復方式とレベル

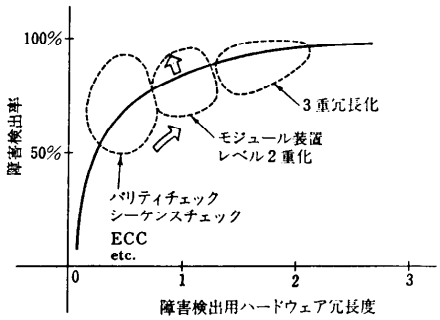


図-3 障害検出におけるハードウェア冗長性の効果

ようと試みることが，大型計算機システムでは一般に行われている．もちろん再試行はすべての場合に可能なわけではなく，これらの判定に必要な情報は，通常動作時に各種の状態レジスタに退避しておくことになる．また再試行しても回復に成功するとは限らない．再試行機能を阻害する要因には次のようなものが考えられる．

（1） 命令の開始時や，命令のチェックポイントにおいて，再試行に必要十分なデータが退避できない時．

（2） システム内で多くの動作が並行して進行しているため，障害検出時にその影響範囲を限定するのが困難な時．

（3） 再試行に移っても，一般に試みられる 10 回前後の再試行の間に回復する障害でない（永久障害）時は当然回復しない．

（4） 再試行制御部に問題があったり，検出回路が不備で，故障発生時点からかなり経てから検出機構にひっかかった場合には，再試行が成功しないだけでなく一見成功したように見えて間違った処理を行ってしまう可能性もあって問題である．

これらのケースの発生比率を減少させるため，退避用のデータ・レジスタを増大させ，より複雑な命令の再試行も可能になってきているが，性能や装置規模の兼ね合いで多くの困難にぶつかっている．また(2)に記したように大型計算機では並行処理度が高いため検出された障害と影響をうける命令との関連をとることもきわめて難しい．さらにマシン・サイクルが極度に小さいため，障害検出時に必要な情報を凍結することも困難になっている．

しかしソフトウェアで用いられる命令とその組合せは基本的なものが多いので，実際上はほとんどの場合に再試行可能である．

3.3 自動縮退機能

機能単位や回路が複数個あり，障害を発生した部分が外部機能に対して透明である場合は，その部分を取り除き，縮退して動作を続行する．このような例としては，

（1） 処理装置がアドレス変換の高速化のために保持している TLB (Translation Lookaside Buffer),

（2） 処理装置内の BS (Buffer Storage : Cache ともいう) およびこれを制御する BAA (Buffer Address Array)

の縮退等にすでに適用されている．

この種の機構は，

- 見通しがつけやすいこと
- 処理装置内の障害のかなりの部分がメモリ素子によっていること

等から，記憶装置を中心に実用化していると考えられる．しかしこれらの作業用の記憶装置の縮退も方式によっては，かなり性能低下をもたらすので，縮退の程度を最小限にとどめるような考慮が払われていないと，実際上はシステムクラッシュに結びついてしまう．

3.4 マシンチェック割込み

これまでにのべた障害回復処理は，基本的にはすべてハードウェアがソフトウェアに知らせることなく，自動的に行う．この結果を制御プログラムに報告するのがマシンチェック割込みである[3]．

マシンチェック割込みの概念は近年最も進歩したものの1つである．今までのべた障害回復機能は基本的には第3世代機でも存在したが，制御プログラムが高度の木目の細かい回復サポートをしようとすると，情報が不足していたり，色々なところから情報を収集しなければならないということが起きた．これらの経験から現在の大型計算機では，次の概念に分けてマシンチェックを制御プログラムに報告するようになっている．

（1） システム障害
（2） 命令実行障害
（3） 機能縮小障害
（4） システム回復障害
（5） タイマ，外部系等の障害
（6） 警　告

これらの報告に合わせて，各種の処理状態もログ・アウト・エリアに退避して報告される．

このような概念の導入により，より統一的に制御プログラムでのシステム回復サポートが可能になりつつある．しかしこれらを逐一ハードウェアでサポートするのはかなり困難な面もあり，継続的な質の改善が必要である．

4. ソフトウェアによるシステム回復機構

最近のソフトウェアによるシステム回復方式の特長は，システムの下位レベルの回復機構において回復に成功しなかった場合でも，より上位の回復機構による機能的回復を試みることにより，回復の成功率を高める階層的回復の概念が確立し，各社の OS でそれが実現されるようになってきたことであろう[4),5),14)]．

4.1 シングルプロセッサにおけるシステム回復

第3.5世代のオペレーティング・システムの特色は，システムの構成要素のどこにおいて障害が発生しても，その影響をシステム全体でなく，タスク，ジョブといった作業単位にとどめ，可能な限りシステムを稼動させようということにある．第 3.75 世代システムにおいては，この回復の対象となる障害は，制御プログラムも例外ではなく，ディスパッチャ，割込みハンドラといった，従来システムでは核と考えられていた制御プログラムでも各種の障害の発生を予期して，それに対処する機構を組み込むようになった．

具体的には，個々の通常処理ルーチン（機能モジュール）に対応して，それぞれ回復ルーチンを用意する．従来のオペレーティングシステムでは，システムを回復するために必要なクリーンアップ処理を，一括した異常処理 (ABEND; Abnormal End 処理) として行ってきたが，回復処理ルーチンを処理ルーチン（機能）対応にもつことにより，障害発生前に実行していた処理の取消し，無効化を行い，システムを矛盾のない正常な状態に回復させる確率を増大させた．

これらのシステム回復を行うためには，

- 回復終了制御プログラム
- 各タスクの異常終了を処理する TAE (Task Abnormal Exit) ルーチン
- 制御プログラムの機能モジュール単位に回復を行う FRR (Functional Recovery Routine)

等が用意されていて，**図-4** に示すような手順で次々と回復処理が行われる．

回復ルーチンの処理はたとえば，次のようなものである．

- キューを更新中の場合などの，システム共通制御情報間の矛盾の解消

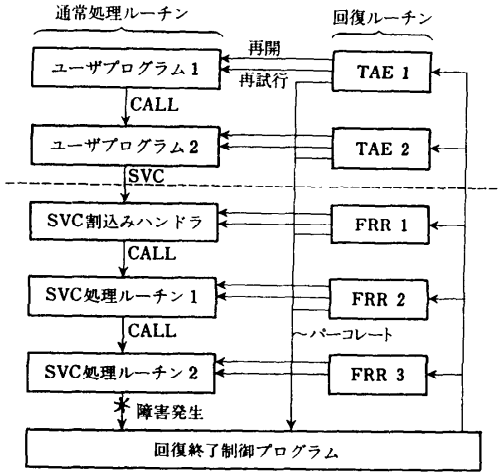


図-4 階層的機能回復の概念図

○システムの作業域やリソースの解放

○デバッグ情報の収集

このような回復処理の後，状況を判断して(i)再試行，(ii)再開，(iii)パーコレート (percolate: より上位モジュールに対応する回復ルーチンでの回復処理の続行)のいずれかの動作を選択する．いずれかのTAEやFRRで再試行や再開が成功すれば，システム回復が成功したことになる．

またFRRレベルで成功しなくても，TAEまでパーコレートできれば，最悪の事態でもタスクの異常終了で済むので一応成功といえる．

この階層的機能回復の考え方と機構により，制御プログラム実行中にハードウェア障害が発生しても，ソフトウェア障害が発生してもシステムクラッシュを防止できる可能性が増したことになる．

これらの回復機能の効果は，回復ルーチンにどの程度用意されているか，回復ルーチンにどの程度情報が渡されるか等によるが，制御プログラムの不良の半分程度は回復できそうである．FRRのクリーンアップ処理をより詳細にすることにより，さらにその効果を高めることができようが，一方ではFRRが正しく機能するための前提となるプログラムやデータの正当性の保証が難しい問題となる．

4.2 密結合マルチプロセッサにおけるシステム回復

密結合したマルチプロセッサシステム (Tightly Coupled Multiprocessor) では一方のプロセッサの故障が他のプロセッサをも含めたシステムクラッシュに至らしめる．いわゆる“同情病”をいかにして防止するかが問題である．従来の密結合マルチプロセッサの中には単一のプロセッサの障害のかなりの部分が，全システムのクラッシュを引き起しているような例もある．これを避けるためには，**4.1** 節で述べたような手法等を用いて，シングルプロセッサでの回復をまず徹底することが重要である．その理由は，

○制御プログラム実行中の障害がただちにシステムクラッシュに結びつくようなOSでは，密結合マルチプロセッサ構成にしても，事態は何も改善されず，その構成自身が“障害”の増幅機構となってしまう．また実際問題としても，システム使用時間のうちで制御プログラムの占める割合が増加しており，その障害が発生する確率は高くなっている．

○またシステム回復全般にいえることであるが，障害の影響の範囲が当初から限定できない時は，結局システムをクラッシュさせ，IPLからやり直すことになる．これを避けるためには，システムの構造がそのように構築されていなくてはならないことになる．

密結合マルチプロセッサシステムの回復技法としては，ACR (交代CPU回復機構: Alternate CPU Recovery) と呼ばれる技法[5),11),14)]が開発され，一般化しているが，その回復手順は概略次のようである．

(1) 障害を発生した処理装置側では，独自のシステム回復を開始する．

(2) 障害処理装置側での独力の回復が不成功となると緊急指令を正常処理装置側へ発する．

(3) 正常処理装置側では，

○障害を発生している処理装置の環境を設定し，

○実行中であった回復処理をシミュレートし，

○実行途中であった入出力処理も終結させる．

これらの回復機能は，障害を発生した処理装置が，回復機能を破壊しない限り，ほぼ有効に機能すると考えられる．しかし，いずれにしてもシステム性能は半減するので，これが許容できないシステムでは，実質的にシステムクラッシュになる．

4.3 疎結合マルチプロセッサにおけるシステム回復

疎結合マルチプロセッサシステム (Loosely Coupled Multiprocessor) ではプロセッサ間の分離度が高く，全体として信頼度を向上する手段としてもっとも一般的に用いられているが，運用や操作が煩雑になるので，その面での技法が色々試みられている．たとえば疎結合マルチプロセッサをシングルシステムイメージ

で運転させるジョブ管理システムにおいては，主プロセッサ（グローバルプロセッサ）に障害が発生すると，系全体のクラッシュになるので，これを防止するためこの主プロセッサの交代機能が必要となり，このため DSI (Dynamic System Interchange) と呼ばれる方式が開発されている．さらに操作ミスを防止するための自動 DSI 方式も試みられている．

5. むすび—今後の展望にかえて—

以上，汎用大型計算機システムにおけるシステム回復技法を，特にシステムのサービス停止（システムクラッシュ）を回避するために重要性を増しつつあるシステムレベルの回復技法に重点を置いて解説した．

大型計算機システムに要求される信頼度は今後益々高くなろうが，コンポーネントの信頼度の向上とシステム回復技法の強化と発展により，1年に1回程度のクラッシュ率の実現は近い将来に可能である．

特に，一般化しつつあるマルチプロセッサ構成を利用したシステム回復技法も，フィールドにおける多くの知見を加えて精細化され，数年〜十年といった MTBC も可能になってくると思われる．この場合，

- 許容されるサービス停止時間
- ユーザシステム開発時の擾乱

等も問題になってこよう．

システム回復機能の強化は，何らかの冗長性を伴い必然的にコストの増加を伴うので，商用の汎用機においては，信頼性の向上効果とコストとのトレードオフが計られようが，今後に要求される信頼度水準を実現するためには，このためのシステム信頼度の現実的な予測および設計手法の確立が必要である．

またシステム回復技法に代表される耐故障性強化技術と相補的な技術としてのフォルト・イントレラント技術（非耐故障性技術）へのリソース配分のバランスの改善と，システム回復技法の統一的体系化のもとでの，冗長性実現のためのハードウェア，ファームウェア，ソフトウェア間のリソース配分のバランスの改善のための技術課題への挑戦が継続され，ユーザの求める高信頼化技術が進展してゆくであろう．

参 考 文 献

1) 情報処理学会編：コンピュータ・システムの高信頼化 (1977).
2) 北村，富田：RAS 技術の現状，情報処理，Vol. 12, No. 8, pp. 497-504 (1971).
3) IBM: System/370 Principles of Operation, IBM マニュアル GA 22-7000-8 (Oct. 1981).
4) Scherr, A. L: Functional Storucture of IBM Virtual Storage Operating Systems Part II, IBM Syst. J., Vol. 12, No. 4, pp. 368-381 (1973).
5) Sprague, S. C: OS VS 2 Release 2 and MP Hardware Recovery Facilities, IBM GUIDE 37 Proc. pp. 633-642 (Oct. 1973).
6) Biersack, F.: M VS SP Release 3 RAS Items, IBM Guide 52 Proc., pp. 32-41 (May 1981).
7) Hsiao, M. Y. 他：Reliability, Availability, Serviceability of IBM Computer Systems: A Quater Century of Progress, IBM J. Res. Dev., Vol. 25, No. 5, pp. 453-465 (Sept. 1981).
8) Auslander, M. A.: The Evolution of the MVS Operating System, J. Res. Dev. Vol. 25, No. 5, pp. 471-482 (Sept. 1981).
9) Avizienis, A.: Fault Torelant Systems, IEEE Trans. Comput., Vol. C-25, No. 12, pp. 1304-1312 (Dec. 1976).
10) 当麻：高信頼化技術，信学会誌，Vol. 62, No. 11, pp. 1260-1269 (Nov. 1979).
11) 松崎：3.75 世代 OS の思想と実現手法，日経エレクトロニクス，Vol. 141-143 (1976).
12) Denning, P. J.: Fault Tolerant Operating Systems, Comput. Surv. Vol. 8, No. 4, pp. 359-390 (Dec. 1976).
13) Randell, B. 他：Reliability Issues in Computing System Design, Comput. Surv., Vol. 10, No. 2 (Jun. 1978).
14) 情報処理学会：計算機システムの解析と制御研究会資料 7-1〜6 (1979 年 12 月 17 日号).
15) 田中他：装置ないしモジュールのレベルで 2 重化を図ったコンピュータ，日経エレクトロニクス，1977 年 3 月 21 日号，pp. 100-121 (1977年).
16) 堀越，久保：最近の大型計算機システムの RAS 機構とシステム信頼度：信学技法，Vol. 78, No. 26, pp. 19-23 (May 1978).

(昭和 57 年 1 月 20 日受付)