コンピュータシステムの障害回復技術

福本 聡，海生 直人，尾崎 俊治

1. はじめに

現在のコンピュータシステムでは，障害に備えて様々な冗長性が導入されている．プロセッサや記憶装置を多重化したり，予備を設けて再構成を可能にするなどがそれである．各システムに要求される信頼性と性能は，これらの冗長性を用いたシステム構成技術によって実現される．

しかしながら，実際のコンピュータシステムでは，障害が発生した後に，物理的な再構成や修理だけで直ちに稼働を再開できる訳ではない．一般には，障害によるデータの損失または汚染を仮定しなければならないからである．すなわち，システムの論理的な一貫性を回復するための操作が必要となる．

本稿では，コンピュータシステムの回復技術とその評価について概説する．回復技術は，上に記した構成技術と密接に関係しており，システムがどのような目的で，また，どのような要素で構成されるかによって議論の内容は異なる．ここではまず，次節において基本的な障害回復の原理と評価について述べる．つづいて3節で，現在のメインフレームの中心的な応用であるOLTP(On Line Transaction Processing, オンライントランザクション処理)システムの障害回復技術を取り上げることにする．

ふくもと さとし 愛知工業大学 情報通信工学科
〒470-01 豊田市八草町八千草 1247
かいお なおと 広島修道大学 商学部 管理科学科
〒731-31 広島市安佐南区沼田町大塚 1717
おさき しゅんじ 広島大学 工学部 第二類(電気系)
〒724 東広島市鏡山 1-4-1

2. 障害回復の基本

一口にコンピュータシステムの障害と言っても，その発生箇所や大きさ，性質などは様々である．従って，それによる損失の程度や状態もまた様々であるが，障害回復の手法そのものは，基本的に次のふたつに大別することができる．ひとつは前向き回復 (Forward Recovery) であり，もうひとつは後ろ向き回復 (Backward Recovery) である [1].

前向き回復は，障害によってもたらされた不正な状態に適当な補償を加えて，新しい正常な状態を直接作り出す手法である[1]．一方，後ろ向き回復は，状態を障害の発生や故障の潜伏が起こる前の正常な状態まで戻す手法である．前向き回復は，その原理上，障害の特性とシステムの状態推移を予め特定できる場合しか実現しない．これに対して後ろ向き回復は，全ての障害と不正な状態に対して適用することができる．

本稿では，以下，何れも後ろ向き回復による回復技術について考える．

2.1 チェックポイント

後ろ向き回復では，目的とする状態に復帰するために，ロールフォワード (Roll Forward) とロールバック (Roll Back) のふたつの技法がある [2]．前者は，目的とする状態よりも前の時点の状態を起点として，時間的に順方向に処理を重ねて状態復帰するものである．逆に後者は，目的とする状態より後の時点の状態を起点とする．これに時間的に逆方向に，処理の取り消しを重ねながら復帰する．これらによって，任意の時点のシステムの状態を再構築できるが，そのためのオーバヘッドはかなり高いものになる可能性がある．とり

[1]前向き回復またはフォワードリカバリなる述語は，ここで示した意味ではなく，ロールフォワードと同様の意味で用いられることもある．

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

わけ，システムの稼働初期からのロールフォワードなどは，多くの場合に実用的とは言えない．

そこで，その様な大きなオーバヘッドを軽減するための冗長的手段として，チェックポイント (Checkpoint) が生成される．チェックポイントとは，予め正常な時点で取得した状態の写しを言う．実際の回復処理では，生成された複数のチェックポイントから目的とする状態と論理的に (または時間的に) 最も近いものを基点に選び[2]，ロールフォワードまたはロールバックの復帰処理を実行すればよい．図1にこれらの概念を図示する．なお，チェックポイントと言う用語は状態の写しをとる時刻の意味でもしばしば用いられる．また，取得した写し(上記のチェックポイント)をチェックポイントデータと呼ぶこともある．

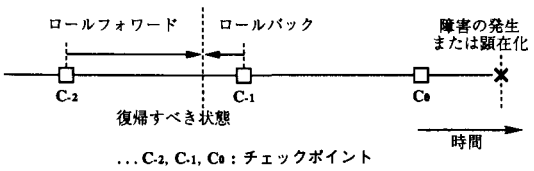

図 1: 回復処理とチェックポイント

さて，チェックポイントを用いて具体的に回復機構を設計する場合，主な関心は，チェックポイントをどのような場所に退避させるかということと，チェックポイントの生成をどのような頻度で行うかということである．チェックポイントの退避場所は，対象とする障害の特性と回復機構の規模によって決る．後述する様に，それはCPU内のレジスタであったり，大容量の磁気記憶ディスクやテープであったりする．また，マルチプロセッサのうちのバックアップに指定されたプロセッサユニットの場合もある [2].

一方，チェックポイントの生成頻度の決定も同様に様々ではあるが，典型的にはチェックポイントの生成オーバヘッドと障害回復処理オーバヘッドとのトレードオフの問題に帰着できるものが多い．すなわち，チェックポイントが自由に設定できる場合，その頻度を高くすれば回復処理に要するオーバヘッドは少なくなるが，当然チェックポイントのためのオーバヘッドは多くなる．逆に，頻度を低くすればチェックポイントのオーバヘッドは少なくてすむが，回復処理のオーバヘッドは多くなる．これらの関係を図示すると図2の様になる．

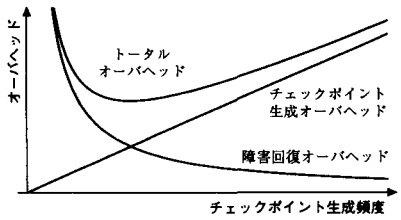

図 2: チェックポイント生成頻度とオーバヘッド

2.2 再試行

チェックポイントによる最も小規模な障害回復は，コマンドレベルの再試行であろう [2]．再試行は，主に過渡障害や間欠障害 [1], [2] からの回復を目的に実行される繰返し処理である．例えば，主記憶装置や二次記憶装置のリード・ライトエラーは，同一の命令の実行によって自然回復することが期待できる．繰り返しの命令実行には時間的オーバヘッドを伴うため，再試行は，時間冗長技術と呼ばれる．

再試行を実行するためには，障害が認められた命令の実行前の状態に復帰する必要がある．従って，チェックポイントは自動的に決り，命令を発行したCPUのレジスタ等の内容によって構成される．それは一定の時間，または，一定の再試行実行規定回数だけ保持され，その間は再試行の実行が保障される．

2.3 評価例

前記のチェックポイント生成頻度の決定について，最も基本的な解析例を示す．Young [3] は，永久ジョブを処理するシステムで，一定時間周期 T_c 毎に処理を中断してチェックポイントを取得する場合について解析している．これは，障害発生後の再処理時間と，障害発生までの全てのチェックポイント生成時間との総オーバヘッド時間を評価したものである．その結果，平均故障時間間隔を T_f とし，チェックポイントの取得時間 T_s がこの T_f より十分小さいことを前提にして，総オーバヘッドを最小とするチェックポイント生成周期 T_c^* は

$$T_c^* = \sqrt{2T_sT_f} \tag{1}$$

として近似的に得られることを示している．例えば，$T_s = 30$[秒], $T_f = 100$[時間] ならば，求めるチェックポ

[2] このチェックポイントによる状態復帰を指してロールバックと呼ぶことがある．

© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.

イント生成周期はおよそ$T_c = 77$[分]となる．

もうひとつ，極めて身近な回復操作に関する例を示そう．著者らは，文献[4]で一定量のデータN[Byte]をキーボードから入力する場合のチェックポイント設定効果について考察した．キーボード入力操作におけるチェックポイント生成の特徴は，入力操作が進み，主記憶データ量が増加するほどデータの書出し(チェックポイント生成)オーバヘッドも増加する点にある．このことを考慮して，チェックポイント生成頻度の変化と，チェックポイント退避場所の分割とを設計のパラメータとする7つのチェックポイント設定方式について解析した．それらのうち，一定のデータ量T[Byte]を入力する毎にチェックポイントを生成する方法で，全てのデータN[Byte]を入力し終るまでの総期待操作時間$L(N)$を評価した場合について示す．解析結果によれば，データの入力速度をv[Byte/秒]，x[Byte]のデータを二次記憶ファイルについて読み書きするための入出力オーバヘッドを$c_u \cdot x + c_c$，また，入力操作中の障害の発生頻度をλとすれば，$L(N)$を最小とするチェックポイント生成データ間隔は

$$T_c^* = \sqrt{\frac{c_u \cdot N + 2c_c}{\lambda(1/v - c_u)}} \qquad (2)$$

として得られる．このとき，$L(N)$は最小値

$$L^*(N) = \frac{N}{v} + c_u \cdot N + c_c + \lambda N(c_u \cdot \frac{N}{2} + c_c) + N\sqrt{\lambda(1/v - c_u)(c_u \cdot N + 2c_c)} \qquad (3)$$

となる．例えば，日本語フロントエンドプロセッサによる文字入力について，[4]の調査結果からv=1.2 [Byte/秒], c_u=1.9× 10^{-4}[秒/Byte], c_c=7.0× 10^{-2}[秒], $\lambda = 10^{-5}$[回/Byte]のとき，$N = 10^5$[Byte]ならば，T^* =495 [byte], $L^*(N)$=2時間19分36秒となる．四百字詰め原稿用紙で言えば，12枚程度の日本語データを入力するとき，およそ250文字毎にデータの書出しを行えばよいことになる．

3. OLTPシステムの障害回復

現在では，為替・預金業務や鉄道・航空の座席予約サービスなどの支援が，メインフレームコンピュータの中心的な応用を占めるようになっている．このように，データベースを共有する大規模で社会性の強い処理形態をOLTP(On Line Transaction Processing, オンライントランザクション処理)と言う．また，それを実現するシステムをOLTPシステムと呼ぶ．本節では，このOLTPシステムの障害回復技術について述べる．なお，以下で用いる述語の多くは，基本的にHaerderとReuter[5]の導入した用語の枠組みによっている．

3.1 システムの構成

通常のOLTPシステムの構成は，図3の様な概念図で表すことができる．複数のユーザから発送されたトランザクションは，ホストコンピュータ上のデータベース管理システムによって並列処理される．個々のトランザクションは，典型的には，データベースのいくつかのページを参照または更新する命令を含む．図4は，トランザクション処理の状況を例示している．例えばトランザクションiは，処理の開始後，ページAを更新してA'とし，その後更にページBを更新してB'とする．そして，システムにコミット，すなわち処理が成立したものとして結果を付託する．システムは，一度コミットされたトランザクションの結果について，如何なる障害に対してもその効果を保障しなければならない．

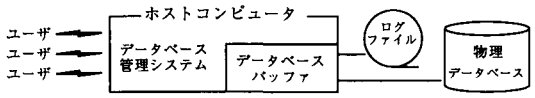

図3: OLTPシステムの構成

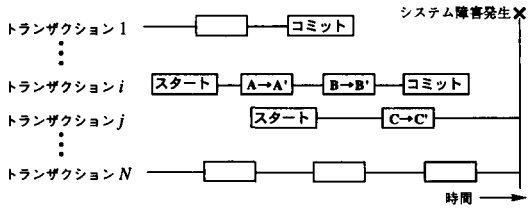

図4: トランザクションの並列処理

預金業務などを例にとれば，上記のページAのA'への更新とページBのB'への更新は「預金者Aの口座からある金額を引出し，それを預金者Bの口座へ入れる」などの操作を考えればよい．ユーザへの保障として，このような取り引き(トランザクション)の結果が，全てのコンピュータ側の障害から護られねばならないことは明かであろう．

更新の対象となるページは，物理データベース(二

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

次記憶媒体)上で直接更新されず，一旦バッファ(主記憶)に読み込まれて更新される．それによって新しいイメージとなったページは，しかるべき時点でログファイルと物理データベースに書き出される．ログファイルは，ジャーナルまたは監査トレイルなどとも呼ばれ，どのトランザクションがどのページをどの様に更新したのかという情報を全て記録している．そして，その書込みは，必ずページの更新が物理データベースに反映される(書き出される)前に行われる．

3.2 障害の分類

OLTPにおける障害は，その原因と影響範囲によって，以下のように大別できる．すなわち，トランザクション障害，システム障害，媒体障害の3つである．

トランザクション障害は，個々のトランザクションの不正な中断を意味する．従って，その影響範囲は，当該トランザクションと，高々それに関係するいくつかのトランザクションだけである．トランザクション障害は，不正な入力や，デッドロック等によって発生し，その発生率は全トランザクションのうちの数パーセントと言われる．

システム障害は，システムの稼働を無制御に停止させる障害である．これは，OSやデータベース管理システムのエラー，オペレータの操作ミスなどが原因となって発生する．また，その頻度は，HaerderとReuter[5]によれば週に数回発生するものと仮定される．システム障害の発生時には，主記憶のデータは保持されないため，一般には，データベースバッファの内容は全て失われるものと考えなければならない．

あらゆる障害の中で，最も影響範囲の大きな障害が媒体障害である．これは，物理データベースである二次記憶媒体(多くの場合磁気ディスク)の損失を意味する．原因としては，二次記憶装置のヘッドクラッシュなどのハードウェア故障や，媒体の劣化などが考えられる．媒体障害は，通常，極めて希な障害であり，1年に1度から数年に1度の発生頻度が仮定される．

以上の障害のうち，回復時間の点で主に問題となるのは，システム障害と媒体障害である．ここでは，この2つ障害からの回復操作について考察する．

3.3 システム障害からの回復

障害回復の観点から言えば，トランザクションには「完全なトランザクション」と「不完全なトランザクション」とがある．障害の発生までにシステムで取り扱われているトランザクションのうち，障害発生の時点で既にコミットされているトランザクションが完全なトランザクションであり，まだコミットされていないトランザクションが不完全なトランザクションである．図4を例にとれば，トランザクションiは完全なトランザクションであり，トランザクションjは不完全なトランザクションである．

システム障害によってデータベースバッファの内容が失われたとき，これらの2種類のトランザクションはデータベースの論理的一貫性にそれぞれ次のような不具合をもたらす．

トランザクションの更新したページがいつ物理データベースに反映されるかは，バッファの管理方法によって決る．それは，明かにシステム障害の発生するタイミングとは無関係であるから，更新されたページのなかには，完全なトランザクションによって更新されたページであるにも拘らず，データベースに反映されずに消失するものが在り得る．一方，同様の理由によって，不完全なトランザクションによって更新されたページであるにも拘らず，障害発生以前に物理データベースに書出されてしまうものが在ることを仮定しなければならない．図4の例で言えば，完全なトランザクションiによって更新されたページA'およびB'は，障害の発生前に物理データベースに書出されていれば好都合であるが，そうではないかもしれない．また不完全なトランザクションjによって更新されたページC'は，障害の発生まで書出されずバッファ内に留まっていれば好都合であるが，そうではないかもしれない．

これらの不具合によって失われた論理的一貫性を回復するための操作が，REDOとUNDOである．REDO操作は，完全なトランザクションによって更新されたページのうち，障害発生までに反映されていないイメージを物理データベースに書出す．また，UNDO操作は，物理データベースに書出されたページイメージのうち，不完全なトランザクションによって更新されたものを更新前イメージに戻す．これらの操作は，ログファイルのページの更新前後の情報によって可能となる．

3.4 REDO操作とチェックポイント

前述のとおり，バッファ内で更新されたページが物理データベースに書出される順序やタイミングは，バッファの管理方法に従って決る．そのため，外側から観

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

れば，個々のページは不規則な時点で書出されるように思われるかも知れない．しかし，大局的にはバッファ内のページは，ほぼ更新された時期の早いものから書出されていく．データベースバッファの容量は，定常的には，参照または更新のために読み込まれた多くのページで満たされている．そのため，新たなトランザクションの要求によってページを読込むには，バッファ内の参照ページを棄却するか，古い更新後ページを書出さねばならないのである．一方，各トランザクションは，必要なページを更新した後に，そのうちそれほど遠くない時点でコミットする．

以上のことから，基本的にREDO操作とUNDO操作の量は，障害発生時点とは無関係にほぼ一定と見なせることになる．コミットしたトランザクションに更新されたページのうち，書出されずバッファ内に留まっているものは定常的にほぼ一定であり，また，ページを更新したトランザクションがコミットしていないにも拘らず，書出されているものもほぼ一定の筈だからである．

ところが，多くの場合，実際のREDO操作の量は一定とはならない．バッファ内には，複数のトランザクションによって非常に頻繁に更新されるページが存在するからである．この様なページはホットスポットページ(Hot Spot Pages)と呼ばれ，長いあいだ物理データベースに書出されないため，障害発生時には多くの更新情報を反映することなく消失する．従ってREDO操作の量は，稼働開始から障害発生までのトランザクション処理量に比例した量を含むことになる．

この様なREDO操作によるオーバヘッドの増大を制限するためにチェックポイントが生成される．トランザクション処理システムにおけるチェックポイントの生成は，バッファ内の更新ページ情報を安全な二次記憶媒体に保存することを意味する．それによってREDO操作の対象は，チェックポイント生成以後のトランザクション処理による更新ページに限定される．なお，ここで用いる二次記憶媒体が，物理データベースであるのか，ログファイルであるのか，または他の別の媒体であるのかは，各システムの構成と設定によって異なる．

具体的なチェックポイントとしては，トランザクション向けチェックポイント，トランザクション一貫チェックポイント，動作一貫チェックポイントなどを挙げることができる[5]．トランザクション向けチェックポイントでは，各トランザクションがコミットするとき，それらが更新したページを全て二次記憶に書出す．この場合，REDO操作は全く必要ない．トランザクション一貫チェックポイントでは，新たなトランザクションの実行を中止し，その時点での全てのトランザクションの実行が終了するのを待って，バッファ内の全更新ページを書出す．動作一貫チェックポイントでは，新たなページ更新を中止し，その時点での全てのページ更新が終了したとき，バッファ内の全更新ページを書出す．

3.5 チェックポイントの評価

障害回復機構を含めたシステムの設計や運用においては，その評価が極めて重要である．多くの場合，評価尺度には，平均稼働率[6],[7]やトランザクションスループット[8]など，性能に対する回復処理オーバヘッドの影響を考察するための量が用いられる．

回復機構の特性は，ページの書出し方式やタイミングなどの具体的な規律によって決る．前記のチェックポイントは，これらの規律の要因のひとつであるが，その設定によって評価尺度の値が大きく変化するため，とりわけ多くの関心が集められてきた．

トランザクション向けチェックポイントは，REDO操作を不要にする反面，そのためのチェックポイント生成オーバヘッドが大きく，通常稼働中の性能の低下が著しいため大規模な応用には向かない．これに対して，トランザクション一貫チェックポイントと動作一貫チェックポイントとは，オーバヘッドが比較的小さいため，障害回復より性能を重視する場合には有利である．特に，動作一貫チェックポイントは，チェックポイント生成時のトランザクションの処理待ちが少なく，より効果的である．

定量的な評価モデルとしては，トランザクション一貫チェックポイント，または，動作一貫チェックポイントに対応する解析的モデルが多く提案されている[6]–[11]．それらは，以下の2種類のオーバヘッドを基本的なモデル要素とする．ひとつはチェックポイント生成の累積オーバヘッドである．一回のチェックポイント生成オーバヘッドは，バッファ内の更新ページ数が定常的に一定であることから定数でモデル化され，これが障害発生までの生成回数だけ累積される．もうひとつは，障害回復オーバヘッドである．これは，チェックポイントの生成から障害発生までの処理量または時間をXとして

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

$$R = hX + c \qquad (4)$$

の様にXの比例項と定数項とでモデル化される．比例項は，上記のホットスポットページに関する項であり，定数項は，ページの書出しとトランザクションのコミットとが定常的に一定であることに対応する．これまでに提案された多様なモデルは，それぞれシステムの構成や運用に関する視点の違いから論じられたものである．しかし何れにおいても，チェックポイントの決定の原理が，上記の2種類のオーバヘッドのトレードオフにあることは，前節で述べたとおりである．

ところで，こういった最適化によるチェックポイントの設定に対して，「障害の発生は希な事象であるから，そのための準備に大きなオーバヘッドを割くべきではなく，許容される時間内に回復操作を終了するために必要な最小限のチェックポイント生成頻度に留めるべきである」という主張がある[5]．これは，実際のOLTPシステムの立場から言えば正論である．障害の発生が希であるということは，発生までの時間間隔が大きいこと，また通常，そのばらつきも大きいことを意味する．つまり，その様な不確定なパラメータを予測して確率的に総合オーバヘッドを評価するよりも，障害回復時間の上限を決定論的に評価してチェックポイントを設定した方がよいという訳である．仮に，予想よりも高い頻度でシステム障害が発生するとしても，多くのシステムで「回復処理が制限時間内に完了するならば，正常時のトランザクション処理性能を回復機能に優先してもよい」という運用上の合意を得られるであろう．

3.6 媒体障害からの回復

最後に，媒体障害に対する回復機構の基本的な考え方と，二次記憶媒体の新しいアーキテクチャについて述べよう．

媒体障害が発生した後，新たな二次記憶媒体にデータベースを再構築するには，データベースのバックアップコピーとログファイルを用いたロールフォワードを実行するのが原則である．物理データベースのバックアップコピーは，2節で述べた広義のチェックポイントのひとつである．通常，データベースそのもののバックアップコピーは，膨大な量のデータ転送オーバヘッドを必要とするため，トランザクション処理業務の終了後，1日，1週間，1ヶ月などの時間間隔で作成される．多くの場合，バックアップコピーは，記憶媒体の磁気的劣化を考慮して，複数の世代に渡って保管される．最新のコピーからのロールフォワードに失敗したときは，より古い世代のコピーを用いて回復操作が試みられる．

この様な標準的な障害回復とは別に，冗長構成によって媒体障害からの回復を容易にしようとする技法がある．最もよく知られているものはミラーディスクであろう．この構成では，同一のふたつのディスクによってデータベースが二重化されるため，単一ディスクの媒体障害に対しての回復操作は不要となる．極めて高い信頼性を達成できる反面，記憶効率が悪くコストが高くなる欠点がある．

最近では，冗長構成をとるディスク装置の新しいアーキテクチャとしてディスクアレイが注目されている[12]．ディスクアレイとは，比較的廉価な複数台のディスクユニットを並列冗長構成して，性能，信頼性，コストの改善を可能としたディスク装置である．一般にそれは，RAID (Redundant Array of Inexpensive Disks) と呼ばれ，Pattersonらによって，RAID1からRAID5までの5つの記憶・構成方式に分類されている[12]．

RAID1 は上記のミラーディスクの構成を指し，RAID2 はディスクアレイのうちの数台をハミング符号の格納に用いる方式，RAID3はディスクアレイのうちの1台をパリティ符号の格納に用いる方式である．RAID4 および RAID5 は，本節で論じているトランザクション処理向けのディスクアレイである．RAID4は，比較的小さなデータブロックを含むデータディスクとそれらに対するパリティディスクおよびスペアディスクから構成される．現在，より実用化が進んでいるのはRAID5であり，RAID4のパリティブロックをアレイ内に分散させた構成をとる (図5参照)．これらのディスクアレイでは，複数ディスクへの並列アクセスによる負荷分散効果が性能を向上させる．一方，障害回復の面では，アレイ内の単一のディスク障害について，ログファイル等の付加的な資源を用いることなく媒体の回復が可能となる．すなわち，図6の様にパリティブロックを含めた残りのディスクの情報から，スペアディスク上に障害発生ディスクの内容を再現することができるのである．

ディスクアレイは，性能や障害回復機能の点で様々な改良が試みられており，高性能・高信頼ディスク装置として今後の本格的普及が期待される．

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

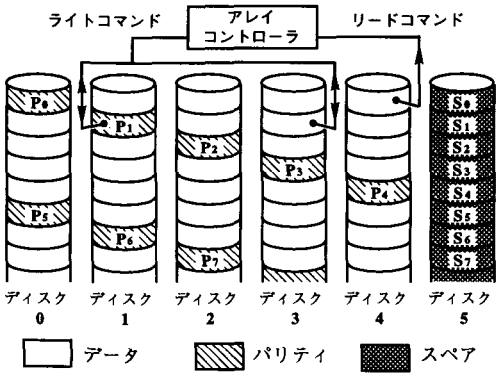

図 5: RAID5 型ディスクアレイの構成

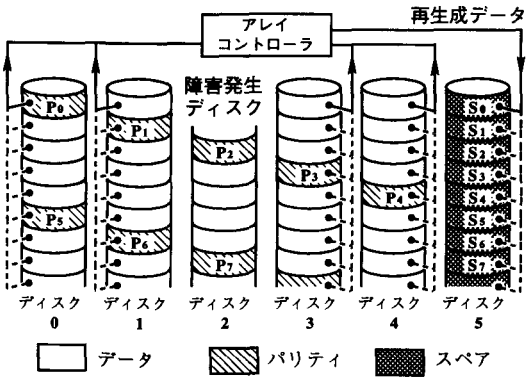

図 6: RAID5 型ディスクアレイの障害回復

4. おわりに

本稿では，コンピュータシステムの障害回復技術の概要について述べた．回復技術がシステムの構成と密接に関連することは，はじめに記したとおりであり，本文の後半では OLTP システムの構成の観点から論じた．同様に近年注目されるトピックとして，分散システムの障害回復があるが，今回は紙面の都合で取り上げていない．これについての優れた文献としては [13] などがあることを付け加えておく．

参考文献

[1] D. P. Siewiorek and R. S. Swarz (eds.): *The Theory and Practice of Reliable System Design*, Digital Press, Bedford, Massachusetts (1982).

[2] 当麻善弘監修, 向殿政男: “コンピュータシステムの高信頼化技術入門”, 日本規格協会 (1988).

[3] J. W. Young: “A First Order Approximation to the Optimun Checkpoint Interval”, *Commun. ACM*, Vol. 17, No. 9, pp. 530-531 (1974).

[4] 福本聡, 石井誠一, 前沢敦司, 海生直人, 尾崎俊治: “キーボードデータ入力操作におけるチェックポイント設定方式とその有効性に関する考察”, 信学論, Vol. J76-D-I, No. 7, pp. 390-404 (1993).

[5] T. Haerder and A. Reuter: “Principles of Transaction-Oriented Database Recovery”, *Comput. Surv.*, Vol. 15, No. 4, pp. 287-317 (1983).

[6] K. M. Chandy, J. C. Browne, C. W. Dissly and W. R. Uhrig: “Analytic Models for Rollback and Recovery Strategies in Data Base Systems”, *IEEE Trans. Softw. Eng.*, Vol. SE-1, No. 1, pp. 100-110 (1975).

[7] E. Gelenbe: “On the Optimum Checkpoint Interval”, *J. ACM*, Vol. 26, No. 2, pp. 259-270 (1979).

[8] A. Reuter: “Performance Analysis of Recovery Techniques”, *ACM Trans. Database Syst.*, Vol. 9, No. 4, pp. 526-559 (1984).

[9] S. Toueg and Ö. Babaoğlu: “On the Optimum Checkpoint Selection Problem”, *SIAM J. Comput.*, Vol. 13, No. 3, pp. 630-649 (1984).

[10] S. Fukumoto, N. Kaio, S. Osaki: “Evaluation for a Database Recovery Action with Periodical Checkpoint Generations”, *Trans. IEICE of Japan* Vol. E-74, No. 7, pp. 2076-2082 (1991).

[11] S. Fukumoto, N. Kaio, S. Osaki: “A Study of Checkpoint Generations for a Database Recovery Mechanism”, *Computers Math. Applic.* Vol. 24, No. 1/2, pp. 63-70 (1992).

[12] Patterson, D. A., Gibson, G. and Katz, R. H. :“A Case for Redundant Arrays of Inexpensive Disks (RAID)”, *Proc. of ACM SIGMOD*, pp. 109-116 (June 1988).

[13] 南谷崇: “フォールトトレラントコンピュータ”, オーム社 (1991).

© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.