データベース・システム 84−16
（１９９１．７．１８）

# ＳＣＭＰシステムでのＯＬＴＰ向け ＤＢＭＳ並列処理

林克己　齋藤一彦　林知博　三谷政昭　大里博志　小幡孝司　関根裕　浦満広　石井卓二
富士通㈱　情報システム事業本部第二ソフトウェア事業部

長期，連続稼働を必要とする大規模リアルタイム・オンラインサービスが広く利用されている．この種のシステムは，①メインフレームの多重化，②下位層を高信頼化する専用ハードウェア採用，等の手法で実現されている．少品種の小規模トランザクションが同時大量処理されるので，トランザクションを階層，マルチマスキング層として採用可能であるという特性は，ここでは十分活かされてはいない．我々は，これを活用した新発想によって，軽い冗長設計で高稼働，高性能システムを実現可能とする新手法を提案する．

# A Parallel Processing Feauture of a DBMS with SCMP for OLPT

Katzumi HAYASHI, Kazuhiko SAITOH, Tomohiro HAYASHI, Masa-aki MITANI, Hiroshi OHSATO, Takashi OBATA, Yutaka SEKINE, Mitsuhiro URA, and Takuji ISHI-I
2 nd Software div. Computer System Group FUJITSU LIMITED, Numazu-shi Shizuoka, Japan

Online services that have large capacity and realtime facility are widely in use these days. And such services need longrun and continuous operation. These types of systems are mainly implemented by the following methods: ① multiplied mainframes, ② specific hardware that makes the lower layers reliable. But in these systems, the characteristic that a transaction can be dealt with as hierarchical masking or multi-masking layer, has not been made good use, in the case of small transactions dealt with in parallel and in large quantities. In this paper, we propose a new idea that makes use of the above characteristic and realizes a system of high continuous operation and high performance with small redundancy.

## 1　はじめに

オンライン処理は社会の基盤構造になってきている．中でも，直接，エンドユーザや顧客窓口を扱うサービスにおいては，特に重要性が高い．

古くからお馴染みの列車，航空券の予約業務から，外国為替取引，株式債権取引，信販業務の与信確認，銀行取引といった経済システムの根幹を支える分野まで，ほとんどの業種で業務の一部または全部がオンライン・リアルタイム処理されるようになっている．

さらに，既にオンライン処理されている分野では，ホテル予約と乗車券購入の一括処理のような複合化による高付加価値化，休日深夜の銀行取引といった計算機処理による無人化ではじめて可能となるサービスも出現している．

この種のオンライン・リアルタイム・サービスの特色は，

① 稼働時間中の停止は不可．
② 十分な高速応答性が必須．
③ 個々の処理は小規模．
④ 総処理量を合算すると莫大．

の４点に要約できる．中でも，①が最重要であるので，この種のシステムはＨＡＳ(Highly Available System)(1)と呼ばれる．特に重要性の高い分野で，40年に２時間のダウンしか許容されないものすら報告されている(2)．

現在，ほとんどのＨＡＳは大型計算機（メインフレーム）のシステムレベルの冗長化によって実現されている．

今後，益々厳しくなる稼働率をシステムレベルの冗長化だけで実現しようとすると，いくら信頼性の高いシステムを採用しても，障害でダウンする単位を１ＴＣＭＰ(Tightly Coupled Multi-Processor)単位以下に押さえることは困難である．したがって，稼働条件の厳しいＨＡＳの構築のためには，少なくとも１セットの大規模システムを待機させなければならないことになる．

しかし，③の特色から，単位データ処理のためには，さほど高いプロセッサ能力は必要としない．実際，単位トランザクションに関しては３２ビット級の市販のプロセッサで十分な応答性を保証することが可能である．

そこで，多数の独立性の高い小規模プロセッサを緩やかに連携させることで，全体としては，必要な応答性を確保し，高いスループットを持ち，相互にバックアップすることで少ない余剰能力で高い冗長性を獲得できることが期待される．これによって所期の目的を経済的に達成できるのではないかと考えた．このアーキテクチャをＳＣＭＰ(System storage Coupled Multi-Processor)(3)と呼ぶ．詳細は後述する．

一方，大きな単位処理能力がないと十分な応答性の保証が困難な複雑なトランザクション処理には，単位処理能力の高いメインフレームを素材にシステムを構築すればよい．

また，トータルな生産性の観点からは，ここで提案する新方式よりも，従来型のメインフレームの利用形態に分がある場合があることも否定できない．

したがって，経済性，性能，信頼性等を総合的に考慮した最適なシステム構成は，一般にメインフレームと，ここで述べるＳＣＭＰシステムを上手に組み合わせた複合システムとなるであろう（図１参照）．

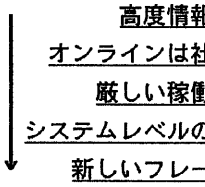


図１　小規模プロセッサによるＳＣＭＰ構築の必要性

なお，一方でＨＡＳ専用システムも，一部で実用化されている．これらによってハードウェアや，オペレーティングシステムといった低レベルのマスキングが可能となるので，我々の提案するトランザクション・レベルの方式と組み合わせれば，より良いシステムが構築可能となるだろう．

## 2　基本的な考え方

前述の様々なフォールト・トレラント・システムでは，ハードウェアやオペレーティングシステムといった下位レベルでの耐障害機能に重点がおかれている．これらと比べて，我々のアプローチの特色はトランザクション処理特有のアブストラクションでの対処を行った点にある．実際のシステムでは下位の機能も具備しているのであるが，ここでは，最も特徴的な部分に絞って議論する．

我々の課題は，一般的にＨＡＳをソフトウェア的に如何に構築するかということではない．これは非常に難しいため，実際に作成された例は余り耳にしない(4)．前項で述べた特殊な応用プログラムによるメッセージ処理が実現できさえすればよいというアプローチを採用したのである．

本題にはいる前に，用語法を定義する(5)．

隣接階層について，相対的に上位にある側をユーザ，下位をサーバと呼ぶ．この対に関して，次の２種類のサーバ異常への対応策が考えられる．

第一は，そのユーザが再試行，代替路の設定等の対処をする．この結果，更に上位からはサーバ以下は正常に機能しているようにみえる．これを階層的マスキングという．

第二は同位の正常な代替サーバに処理を継承させる．これをマルチマスキングという．

この枠組みに従って，我々のシステムの基本概念を説明する．上位層はユーザ，サーバ関係によって，順次，階層の障害に対し階層的マスキング，マルチマスキングを行う（図２参照）．

マクロには，端末からのオンライン要求に対して，二重線で囲まれた集中システムでサービスするというモデルである．そして，ミクロに見ればシステムは単位プロセッサと呼ばれる独立性の高いサブシステムの集合から

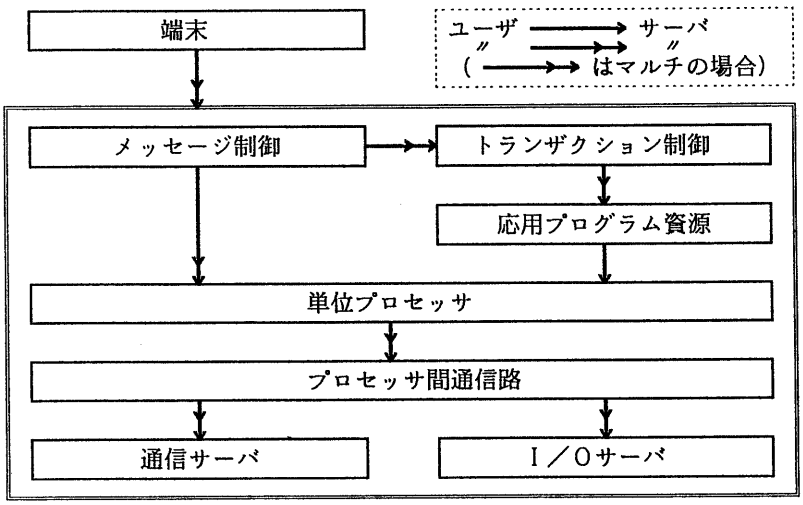


図２　ＨＡＳのユーザ，サーバ階層

なっている．

端末で，データベースの最上位のユーザとなるアクセス要求が発生する．メッセージとして通信サーバ等の下位層を経由して，メッセージ制御に渡る．詳細には端末群ごとに設定された通信サーバからプロセッサ間通信路経由で単位プロセッサ上のメッセージ制御に渡る．

メッセージ制御は各単位プロセッサ上のトランザクション制御へメッセージを振り分けると，トランザクション制御は単位プロセッサに配置された適当な応用プログラム資源のキューに接続して，その処理を委ねる．そして，データベース処理がＩ／Ｏ制御等の単位プロセッサ資源を利用して実行される．

我々の目標とするオンライン・リアルタイム・システムでは少品種，大量処理を想定しているので，大きなスループットを得るために，応用プログラムのコピーすべてが，予め単位プロセッサに，少なくとも論理的には配分済みである．したがって，原則として，どのメッセージを，どのプロセッサで実行することも可能であり，実際，メッセージ単位で多重処理される．

次に，異常時のマスキングの基本について述べる．

単位プロセッサのダウンが他のプロセッサに検出されると，相互のバックアップ規則に従って，（一台とは限らない）別の単位プロセッサ上のメッセージ制御に制御が継承される．このときのメッセージの脱冗送回避は，後述のトランザクション処理に準じて実行される（本稿では詳細は割愛）．

応用プログラムの処理異常はトランザクション制御によって階層的にマスクされ，別の応用プログラムで再実行される．また，単位プロセッサ異常に起因する場合は，メッセージ制御まで戻して，マスクされ，メッセージの再配分で再処理されることになる．

プロセッサ相互にも，応用プログラムのプロセス相互にも，アクティブにバックアップし合っており，マルチマスキングしていることに注意されたい．

なお，応用プログラム論理の障害については対処していない．

従来のメインフレームによるＨＡＳサポートでは，端末がユーザ，その他が計算機システム全体という１階層のマスキングしか行っていないという構図である．また，いわゆるフォールト・トレラント・システムでは，応用プログラムでの冗長化という階層はなく，オペレーティングシステムの機能要素というシステム全体として見たときに非常に局所的で断片的な要素のマスキングしか考慮されていない．

これらと比較すると，単位プロセッサ何台までの障害に対処できるかは，システムで固定的に設定されているのではなく，どれだけの資源を投入するかに関わる事になるという著しい違いがある．

## 3　ハードウェアの構成

我々の提案する上位アブストラクションでのＨＡＳを実現する方式のための最低限のトポロジカルな要件を示す（図３参照）．

基本要件としてＬＣＭＰ(Loosely Coupled Multi-Processor)システムであって，しかも，端末及びデータベースを格納する外部記憶装置が，どの単位プロセッサからも直接アクセス可能な構成で接続されていることが絶対条件となる．

そして，どの単位プロセッサが異常を起こしても，相互にバックアップを可能とするために，トランザクション状況を記録し継承するロギング領域に用いる共用記憶装置が必要である．なお，これは性能要件によって，デ

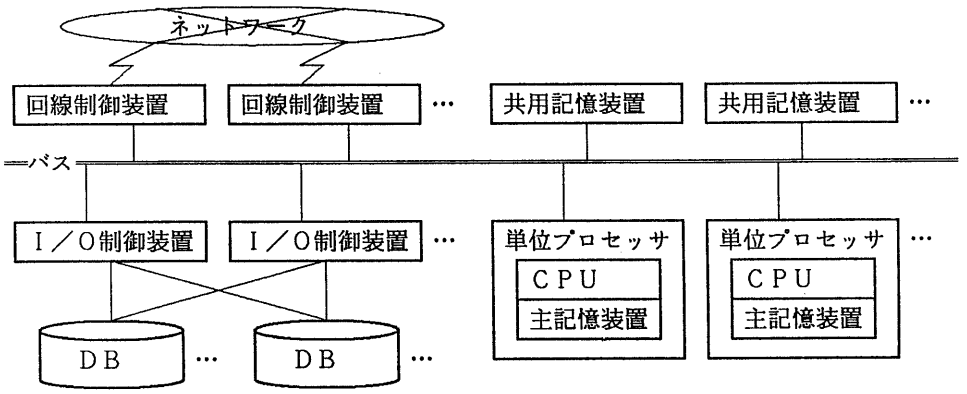


図3　ハードウェア構成（基本概念図）

ータベースと同様に二次記憶装置，半導体ディスク，高速記憶装置，専用プロセッサ等から選択可能である．

各装置は，単位プロセッサによって分担管理される．また，単位プロセッサは，通信路経由で相互監視される．

なお，基本構成と重複して，一般のフォールト・トレラント・システムと同様に下位アブストラクションにおけるマスキング機構が用意されていれば，尚，信頼性の高いシステムにすることができる．

本案と論理的に対応する構成要素があれば十分，上位レベルのＨＡＳを構築可能である．例えば，

① ＳＣＭＰ構成のメインフレーム利用(3)
② 専用プロセッサ（(4)の改造）
③ ＷＳ／ＬＡＮを構築（(6)の変形）

といった幅広い領域に応用可能である．特に②，③の分野は，有望である．

## 4　ソフトウェアの構成

ソフトウェア上の観点からハードウェア構成要件を更に整理すると，単位プロセッサ毎に独立したＤＢＭＳがあって，単位プロセッサ上の主記憶と，共用記憶装置を同時にアクセス可能であり，端末，データベースはすべて見えていることになる．すべての単位プロセッサ間の関係は完全にシンメトリックである（図4参照）．

データベースは，データの参照制約の閉包といった相互関係の完結した単位や，データベースのクロス参照のローカリティなどによってグループ化する．もちろん，自然な分割が存在しないような場合は，完全にランダム

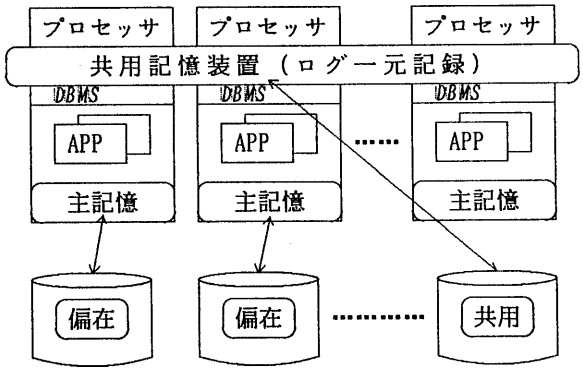


図4　ソフトウェア構成の概要

に分割することもありうる(7).

各グループは，それぞれ1つの単位プロセッサ（の主記憶装置），または共用記憶装置に対応させる．

単位プロセッサは，それに対応するグループのインテグリティに責任を持つ．また，共用記憶装置に対応するグループは，全プロセッサで共同管理する．

ただし，ログは，プロセッサ異常時の継続運転，記憶媒体異常時の回復処理のための整列の便宜を考慮して，システム全体で一元化された一連のデータとして共用記憶装置に追記される．この点で，プライマリ・コピー方式とは異なる(6).

なお，この共用記憶装置と後述の共用制御モードのための装置が同一である必要はない．

## 5　偏在制御モードと共用制御モード

単位プロセッサに対応するグループの制御方式を偏在制御モードと呼ぶ．共用記憶装置に対応するグループの制御方式を共用制御モードという．

偏在制御モードのグループの所属する単位プロセッサ上の応用プログラムからのアクセスは，ログが共用記憶装置に取得される以外は完全にローカルに実行される．これを局所処理と呼ぶ（図5の①参照）．

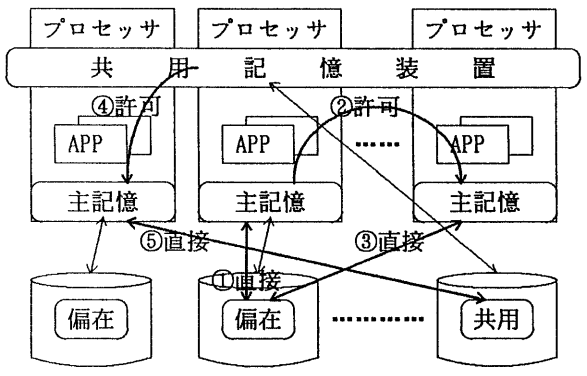


図5　偏在制御モードと共用制御モード

その他の単位プロセッサ上の応用プログラムからのアクセスは，所属プロセッサと通信して，アクセス許可を取得した後に，データベースのコピーが所属プロセッサにあれば転送してもらい，なければデータベースを直接アクセスすることによって実行される．処理結果は直接データベースに書き出す場合と，所属プロセッサに返す場合がある．これを連動処理と呼ぶ（図5の②③参照）．

共用制御モードは，どの単位プロセッサもグループに対して特権は持っていない．アクセスはグループの所属する共用記憶装置上でシリアライズされ，また，ここに存在するバッファを利用して実行，制御される．その意味で，各々の単位プロセッサが偏在制御モードの他のプロセッサと同様の連動処理を実行するともいえる．これを共用処理と呼ぶ（図5の④，⑤参照）．

## 6　制御モードの動的変更

ハードウェアの構成手法によって差があるけれども，一般的にいって，局所処理に比べて，連動処理と共用処理のアクセス・コストは大きいといえそうである．連動処理と共用処理の比較はアーキテクチャ依存で多様であろう．

共用処理は単位プロセッサ間の負荷を平等にするという利点を持つ．

また，偏在制御モードのグループを共用制御モードに移行する処理は，所属プロセッサのイニシアティブで，共用制御モードの制御表を完全に作り出した後，共用を宣言すれば可能である．逆も同様で共用記憶装置の一部を，ある単位プロセッサが占有した時点で偏在を宣言した後，段階的に偏在制御モードの構成に変更すればよい．

この仕組みを利用して，単位プロセッサの負荷状況に応じて，ダイナミックに偏在制御モードの担当プロセッサを変更したり，共用／偏在制御モード変更を容易に実現可能である．

例えば，メッセージ分配とデータベースのグループ割りの密接な対応関係を設定しにくい場合に，連動処理と局所処理の頻度統計をもとに，自動的に最適な共用／偏在制御モードの調整をすることが可能である．

## 7　プロセッサ異常のマスキング

単位プロセッサの異常がプロセッサ間の相互監視によって検出されると，そのプロセッサの処理を停止させ，メッセージ制御機能を他のバックアップ・プロセッサに予めさだめておいた規則にしたがって移管する．また，その単位プロセッサが偏在制御モードで支配していたグループは他のプロセッサの配下に移管される（図6参照）．

詳細なシナリオは次のようなものである．異常プロセッサ配下の偏在制御モードのグループ全体が瞬時，新規のアクセスを禁止される．この後，共用記憶装置上のログを利用してリカバリの必要な範囲に禁止範囲が縮小される．その後，暫くしてリドゥ，アンドゥされた最新のバッファ状態が復元され，現状復帰する．それぞれに要する時間は，専用プロセッサ（3の②参照）では，数ｍｓ，数秒の範囲に収まるであろう．

なお，偏在制御モードの移管完了を待てない連動処理中の応用プログラムが巻き添えで異常終了を宣言する可能性が若干ある．ただし，トランザクション制御が応用プログラムのマルチマスキングをしているのでマクロにはサービスの中断はない．

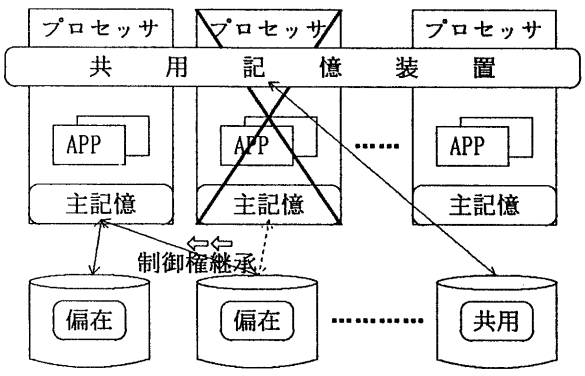


図6　偏在制御モードのデータグループのマスキング

## 8　おわりに

ＤＢＭＳのトランザクションの特性を活用することによって，上位のアブストラクションの冗長化によって，オンライン・リアルタイム処理向けの高信頼，高性能のシステムを何処にでもあるような比較的簡単なハードウェア構成を素材にして実現可能であることを示した．

この提案は，いわゆるフォールト・トレラント・システムで実現されている手法とは直交する概念である．そして弊社のＭ１８００のような超巨大マシンから，マイクロチップを数個並べただけのマシンまで広いシステムに応用できる．

しかし，素材の特性によって，それぞれ大幅に異なる問題を抱えていると予想される．今後の課題は，何が決定的なパラメータであり，どう寄与してくるのかを順次明らかにして行くことであろう．

現在，開発中のプロトタイプに基づいて，さらに検討を加えていきたいと考えている．

最後に本論文作成に当たって，様々の特許調査，文献検索に協力していただいた弊社の多くのスタッフの方々に謝意を表しておく．

## 参考文献

(1) Kim, Won, 'Highly Available Systems for Database Applications', ACM Computing Surveys, Vol.16, No.1, March 1984, pp71-98.

(2) Zorpette, Glenn, 'Computer that are 'never' down', IEEE Spectrum, pp46-54, April 1985.

(3) Shimizu, Kazuyuki et al., 'FUJITSU M-1800 モデルグループのシステム概要', FUJITSU, Vol.42, No.2. 1991-3, #242, pp113-124.

(4) Muramatsu, Hiroshi et al., 'システムを止めずに保守・運用が可能なＯＳを開発', NIKKEI ELECTRONICS 1991.2.18, pp209-223.

(5) Cristian, Flaviu, 'Understanding Fault-Tolerant Distributed Systems', CACM, Vol.34, No.2, pp56-78, Febrary 1991.

(6) Rahm, Erhard, 'PRIMARY COPY SYNCHRONIZATION FOR DB-SHARING', Infomation Systems, Vol.11, No.4, pp275-286.

(7) Hayashi, Katzumi et al., 'データベースの格納構造定義方式—(1)～(8)', 情報処理学会第41回（平成２年後期）全国大会講演論文集(4), 4-25～4-40, 1990.9.4.

(1991年04月08日執筆, 1991年06月14日加筆)