7K−2

連続運転システム SURE SYSTEM 2000 のOS SXO (1) — 概要 —

伊達政広，吉田浩
富士通（株）

1. SURE SYSTEM 2000の位置付け

90年代の計算機システムを取り巻く環境に対して，当社は，メインフレームシステムコンセプトMISSION/DCを提案している。その通信処理領域に向けて，連続運転性，拡張性，接続性を持ち，次の役割を果たすSURE SYSTEM 2000（以下SURE）を開発した。

1) メインフレームシステムの通信処理を，連続運転能力を持つSUREが分担することによって，ユーザから見た全体の連続運転性を実現する（図1）。
2) SUREによるメッシュネットワークなど，SUREの連続運転能力を生かした無停止データ系ネットワークの構築を可能とする（図2）。

図1の適用形態では，当社の既存のメインフレームシステム(Mシリーズ+CCP)で，業務処理の一部としてホスト上のアプリケーションプログラムで実施していた業務選択，プロトコル変換，コード変換などの通信処理を，SURE上のプログラムが実施する。既存のいわゆるFTC(Fault Tolerant Computer)では，資源をすべて自身で管理して無停止化を図るが，SUREでは，ホストのMシリーズとの連携により，既存のホスト上のアプリケーションや資産を移行することなく，システム全体の連続運転を実現できる。

2. SUREの設計コンセプト

90年代の計算機システムへのニーズとして，

1) 銀行の営業時間延長，商店の終夜営業など，種々のサービスが常時提供されることが要求され，その前提となる情報システムが，社会生活の基盤（インフラストラクチャ）化しつつあること
2) 国際化の進展により，世界中の拠点を結ぶネットワークが構築され，時差を越えた24時間の情報管理が必要となっていること

などがあり，ここから，計算機システムに求められる重要な能力の一つとして，連続運転があげられる。

連続運転の阻害要因の一つに，ハードウェア，ソフトウェアの故障がある。これを救済するいわゆるFTCが商品化されているが〔Serlin84〕，ソフトウェア障害によるシステムダウンは，依然として大きな問題であるといえる〔規格協90〕。一方，余り知られていないが，保守作業からの停止要因として，OSプログラムの保守や，システム規模の拡張があり，システム稼働中のプログラム修正や，ハードウェア機器増設に伴うOSの構成定義，システムパラメタなどの変更というソフトウェアの活性保守性が重要となる。

そこで，既存のFTCのカバーする耐故障性に加えて，さらに高度なソフトウェアの耐故障性と，それにとどまらずに，ソフトウェアの活性保守性や拡張性などシステム保守・管理まで含めた，より広い意味での連続運転の実現をSUREの設計目標とした。これをコンセプトとして再整理すると，次のようになる。

①耐故障性：ハードウェア故障に加えて，特にソフトウェア障害に対して，システムを停止させない。②活性保守性：プログラム修正，構成変更などの保守作業を，システム稼働中に行える。③拡張性：CPU，通信回線，ディスクを増設してシステム規模を拡張でき，しかも，その作業を稼働中に行える。

3. SXOの連続運転実現技術

SUREのOS SXO(SURE SYSTEM 2000 eXpandable OS)は，上記の目標を実現するために，基本アーキテクチャから新たに設計されたOSである。ここでは，上記の

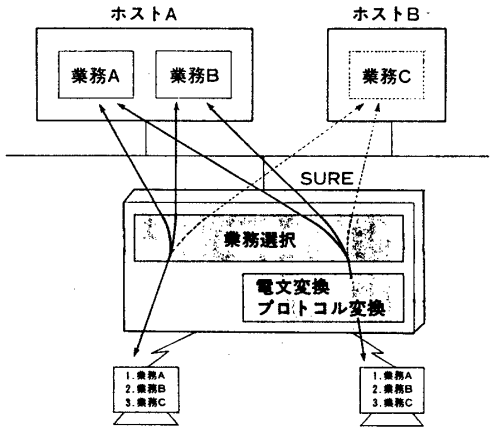

図1　メインフレームシステムの通信処理の実現

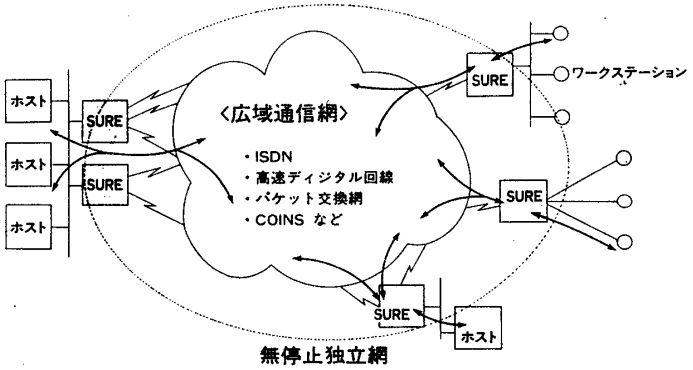

図2　無停止データ系ネットワークの構築

Operating System SXO for Continuously Operable SURE SYSTEM 2000 (1) - An Overview
Masahiro DATE and Hiroshi YOSHIDA
FUJITSU Ltd.

耐故障性，活性保守性，拡張性の実現に向けて体系的なアプローチをとり，多くの技術を有機的に関連付けて採用している。以下に，これらのSXO の技術的なポイントをまとめるが，本論文に続く論文(2)〜(7)で，個々のテーマに関する詳しい報告がある。

3.1 OSの基本アーキテクチャとAPI

SXO では，連続運転を実現する基礎として，「ソフトウェアを交換可能な部品で構成する」という考え方に基づき，OSの個々の機能を，相互に独立性の高いサーバに分解して実現している。これを支える基本アーキテクチャとして，メッセージベースインタフェースを採用した（後続論文(2)）。

一方，OSのAPI(Application Program Interface)ではメッセージを直接見せず，既存のプログラミングスタイルとの親和性を重視している。具体的には，システムコールやユーティリティは，WS/PC 系の国際規格POSIX 〔IEEE88〕に沿ったものとし，トランザクション制御や，端末の画面制御などはホストのM シリーズのDB/DC 機能であるAIM のAPI を継承している。

3.2 耐故障性の実現

メッセージで通信し合うサーバでOSを構成するアーキテクチャの上で，サーバの冗長化と切替えによる耐故障化を行った。ここでは，既存のFTC で採用されている手法に加えて，不揮発共用メモリを使用した遅延引継ぎとエッセンス引継ぎ方式を採用し〔黒羽90〕，さらに，OS機能のアトミックアクション化の考え方によって，たとえば，通信処理におけるコネクション継続のように，外部に故障を見せない高度な耐故障性を低オーバヘッドで実現すると共に，ソフトウェア活性保守実現の基礎とした（後続論文(2)(5)）。

ところで，ソフトウェア障害に対する耐故障化手法として，従来から，Nバージョンプログラミング〔Avizienis 85〕やリカバリブロック〔Randell 75〕などが提案されている。しかし，OSのように，入出力装置などの状態やタイミングに依存して動く大規模なソフトウェアに対する体系立った方法論は確立されていない。そこで，SXO では，障害の発生から，検出，切替え，引継ぎ，再実行という一連の救済処理の流れを，救済過程図という新たな設計パラダイムによって分析しながら設計することにより，ソフトウェア障害に対する高い耐故障性を実現している（後続論文(3)）。

3.3 活性保守性の実現

ソフトウェア活性保守の対象としては，OSプログラムの修正作業と，従来，システムジェネレーションと呼ばれる作業で実施されていたシステム機器の構成定義や各種OSパラメタなどのOS定義体の設定・変更作業がある。これを実現するために，耐故障性の項で述べたOSの部品化と，遅延引継ぎを中心とする交換技術に加えて，①OSプログラムや定義体の版数管理と差分格納，②OS定義体の一元管理，③DBMS指向の定義体管理（正規化やビューの概念，ディクショナリ的アプローチによる重複排除と一貫性保証），④複数サーバ間で一貫性を持った2相コミットベースの動的反映などの技術を採用した（後続論文(4)）。

また，保守作業の高度化・迅速化の一環として，顧客と保守センタの高速回線経由の接続による遠隔監視・保守サービスを実現している（後続論文(7)）。

3.4 拡張性の実現

機能の部品化と複数プロセッサへの均等分散配置を基礎に，OS定義体の活性保守，OSIシステム管理の概念〔OSI 89〕に基づく装置状態管理などの技術を組み合わせることによって，処理量の増加に対して，システムを停止せずに能力を拡大できるようにしている。すなわち，システムの運用中のCPU，ディスク，回線関連機器の追加と，対応するOS定義体の変更が可能であり，これに従って，新たな機器を含めたシステムの負荷分散が行われる（後続論文(6)）。

4. おわりに

我々は，90年代のユーザニーズを満たすには，単に故障に強いだけではなく，ネットワークの構成要素やホストコンピュータを含めて，システム運用のあらゆる局面において連続運転を実現するシステムが必要と確信しており，これに応えるために，以上のアプローチを一層強力に進めてゆきたいと考えている。

【参考文献】

〔黒羽90〕黒羽，加藤，田中：エッセンス情報引継ぎ方式によるOSのフォールトトレラント化，第40回情報処理学会全国大会講演論文集Ⅱ，pp.750-751 (1990).

〔規格協90〕"システムの高信頼性技術に関する調査研究（電子応用システム）報告書，" 日本規格協会，東京 (1990).

〔Avizienis 85〕Avizienis, A., "The N-Version Approach to Fault-Tolerant Software," IEEE Trans. Softw. Eng., Vol.11, No.12, pp.1491-1501 (1985).

〔IEEE88〕"Portable Operating System Interface for Computer Environments," IEEE, New York(1988).

〔ISO 89〕"Information Processing Systems - Open System Interconnection - System Management - Part 2 : State Management Function (ISO/IEC/JTC1/SC21/N3926)", ISO DP 10164-2 (1988).

〔Randell 75〕Randell, B., "System Structure for Software Fault Tolerance," IEEE Trans. Softw. Eng., Vol.1, No.2, pp.220-232 (1975).

〔Serlin84〕Serlin,O.,"Fault-Tolerant System in commercial Application," Computer, Vol.17, No.8, pp.19-30 (1984).