大規模高信頼プラットフォームの構築 (1)
－高信頼性プラットフォームのアーキテクチャ

2S－1

内田 昭宏*、佐藤 孔一*、中川 いずみ*、中嶋 輝明*、横田 潔*
*NEC 流通・サービス業システム開発本部

1　はじめに

従来、オフコンと専用OSをベースとして開発されてきた大規模な業務システムにも、オープンシステムが使用されるようになってきた。

オープンシステムは、多様なアプリケーションを実行するプラットフォームとして優れている反面、定期的な人手による管理が必要とされたり、ユーザの不用意な操作がシステム障害を引き起こし易いなど大規模システムで安定運用を実現するためには問題となる点も多い。ネットワークを介して多数のマシンを管理する分散システム管理技術も研究・開発されているが、本稿で対象とする大規模な業務システムでの制約条件はほとんど考慮されていない。

本稿では、大規模展開システムに特有の制約下で高信頼を実現するため、我々が実際に適用した高信頼アーキテクチャについて説明する。

2　大規模システム

以下に、本稿で対象とする大規模システムの特徴を挙げる。

- 全国規模で大量に展開される、定型業務を主体とする業務情報システムであり、展開数は、数百から数万台にまで達することがある。
- 各設置箇所にはシステム管理者は配置されない。システム運用は集中管理のための保守センタから行う。保守センタとの間の通信回線は多くの場合、公衆あるいはISDNのダイアルアップ接続である。
- すべての設置箇所は基本的に同じ構成であり、同じアプリケーションが動く。エンドユーザ自身がソフトウェア/ハードウェアの追加/変更を行うことはない。
- ミッションクリティカルな業務情報を扱う、また、多くの場合、各拠点に設置された端末自身がローカルに業務情報を保持する。
- ほとんど教育を受けていないユーザでも、すぐにシステムを利用できる操作性が求められる。

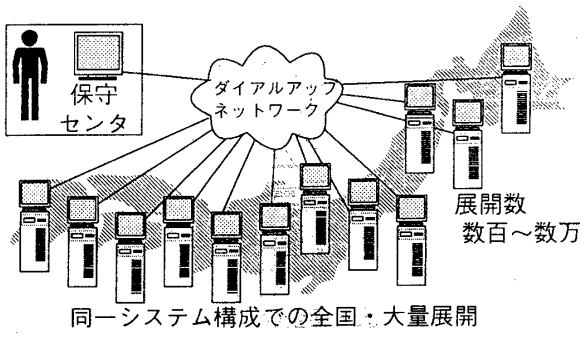

図 1: 大規模システム

このような特徴をもつシステムには全国に展開されるオフィスサーバや店舗サーバ、さらにKIOSK端末などがある。

3　高信頼への制約条件

ここで対象とするシステムは大規模展開という特徴から、信頼性をつくる上で以下の制約がある。

コスト制約:

大量展開ゆえにコスト制約は非常に厳しい。ハードウェアの冗長構成は、ディスクなどデータ保全に必要な最低限のコンポネントに限定される。また、リモートメンテナンスのための付加ハードウェアも採用できない。

保守センタ:

各設置箇所にはシステムの管理をできる人がいない。保守センタで大量のシステムを一括管理するため、障害の通報を受けるための回線数、障害対応をするための運用スタッフ数などには限りがある。

エンドユーザの操作性:

システム障害からの復旧、プログラムの更新などのシステム維持管理の操作に、エンドユーザの介在は期待できない。また、エンドユーザ向けの操作性向上のため、標準的なマウス＋キーボードという入力装置を持たないこともある。

4　高信頼アーキテクチャ

上記の制約下で高信頼を実現するため、以下のポイントに主眼を置いた。

1)障害の発生そのものを抑える

A Platform for Highly-reliable Large Scale Systems(1) - Architecture

*Akihiro Uchida, *Kouichi Satoh, *Izumi Nakagawa, *Teruaki Nakajima, *Kiyoshi Yokota *Distribution and Service Industories Systems Development Division, NEC Corporation

ハードウェア、ソフトウェアの稼動状況、運用に支障のない軽微な障害の発生回数を計測し、予防保守対策を講じる。

2)障害からの回復に人手による操作を必要としない

障害が発生した場合でも、可能な限り自動回復を試みる。また、OS の panic など回復不能の障害により業務処理が中断された場合でも、人手による操作なしに自動的に回復のための処置を講じる。

3)ユーザにシステム機能を見せない

プログラムのリビジョンアップなど運用管理のための処理にユーザの操作を要求しない。また、ユーザに対し不要なシステム操作を隠蔽し、障害通知も必要最低限のものに限定する。

これを実現するため我々が用いたアーキテクチャを図2に示す。

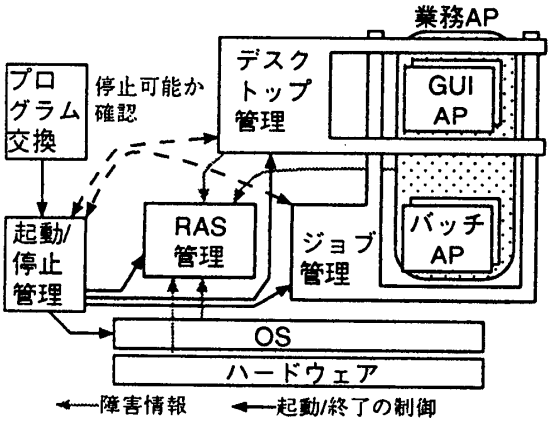

図 2: ソフトウェア構成

ハードウェア、OS を始め、全てのコンポネントからの運用情報や障害情報は RAS 管理機能で一元的に集中管理する。軽微な障害の発生回数や、ハードウェアの稼働時間などは予防保守情報として管理する。

システムの運用に支障をきたす障害であっても、RAS 管理機能が障害の内容を判断し人手を介することなく自動回復を試みる。そして、どうしても自動回復が不可能な必要な障害だけを保守センタに通報することで、保守センタへの通信回線、保守センタスタッフの手間を最低限に抑える。

次に問題となるのが、システムの起動 / 停止と、ソフトウェアのリビジョンアップ時の処理である。実行中の業務処理に影響をあたえずシステムを安全に停止する機能、さらに、重大な障害により安全な手順をつかえずにシステムが緊急停止した後に、安全に業務を再開する機能を実現している。

システムの安全な停止:

- ユーザ操作による指示をうけ、実行中の業務処理 (画面業務・バッチ処理) が安全に停止できるタイミングまで待ってシステムを停止させる。システムコンポネント間の依存関係記述により、システムを安全な順序で停止させる。また部分的な機能の停止も可能としている。
- UPS からの電源断信号などで、最低限のシステム保全のための手順で緊急にシステムを停止する。

障害からの安全な回復:

- システム運転停止の時間中にスケジュールされていたジョブを、電源投入時に順次起動する。
- 緊急停止でジョブが中断された場合、システム再起動時に途中のチェックポイントからジョブの実行を再開する。

ソフトウェアのリビジョンアップについては、OS を含めた交換操作を、できるだけ業務処理を妨げず自動的に行うことを可能にしている。プログラム交換は以下の手順で動作する。

1)デスクトップ管理、ジョブ管理と連携し、安全にプログラムを交換できるポイントを検出する。

2)業務アプリケーションの交換ならば、業務アプリケーションだけを停止 (デスクトップの操作はブロックする)、基盤プログラムならば、業務と基盤の停止、OS 交換ならばシステム全体停止というように、必要最低限の機能を停止する。

3)プログラム交換後、業務再開する。

5 まとめ

この高信頼アーキテクチャを実際のシステムに適用した。実際にフィールド展開を進めるうえでは当初想定していない障害の発生により復旧に手間取るようなことも幾度かあったが、基本的なアーキテクチャには手を触れることなく設定の変更で対応をすることができた。

本アーキテクチャは、大規模システム特有の制約をもった環境をファーストターゲットとしたが、今後、適用システムの領域をさらに広げ、本アーキテクチャの有効性を実証してゆく。

参考文献

[1]佐藤 他, “大規模高信頼プラットフォームの構築 (2)- - GUI アプリケーション構築のプラットフォーム,” 情報処理全国大会, 1996.

[2]坂田 他, “大規模高信頼プラットフォームの構築 (3)- - 自律分散システム構築のプラットフォーム,” 情報処理全国大会, 1996.