B-025

SAN ブート環境におけるシステム高信頼化技術の開発

Study of High-reliability method for SAN boot system.

畑崎　恵介†
Keisuke Hatasaki

１．はじめに

近年、IT システムの肥大化・複雑化・ブラックボックス化により、システム構築・運用にかかるコストが増大している。このため、各企業はシステムコストの削減を目指し、サーバコンソリやオープンシステムの導入を積極的に行っている。一方、企業の IT 依存度が高まる中、企業イントラシステムであっても、障害からの早期回復が求められるようになってきている。このような背景の中、ディスクの冗長化によりデータ保持の信頼性を高める RAID システムの利用や、サーバ障害時に別のサーバへと業務を引き継ぐサーバ冗長化構成を低コストで構築するニーズが急速に高まっている。

しかし、代表的なサーバ冗長化構成である HA クラスタシステムでは、サーバ障害時に別サーバへ高速な業務の引き継ぎが可能であるが、余分なサーバやライセンス費用が必要となり、高コストとなる。このため、コスト削減を迫られるユーザにとって、HA クラスタほど高速な業務の引き継ぎを実現できなくても、必要最低限のリソースで、低コストなサーバ冗長化機能を求める声が高まっている。

そこで、本研究では、サーバ冗長化方式として N+1 冗長化システムに着目し、(1)低コストで、(2)I/O オーバヘッドが無く、かつ(3)高速な業務引き継ぎが可能な N+1 冗長化方式の実現を目指した。

2.サーバ冗長化

システム内で業務を稼動中の複数のサーバのうち、いずれかのサーバで障害が発生した場合、直ちに別のサーバが業務を引き継ぐことで、障害の影響を最小限に抑える方法として、「サーバ冗長化」がある。サーバ冗長化を実現する代表的な方法として、「HA（高可用性）クラスタシステム」と「N+1 冗長化システム」が存在する。これらの方式を比較すると、本研究の狙いである低コストなサーバ冗長化方式としては、「N+1 冗長化システム」が適していることが分かった。以下に、各方式の詳細と、その比較について述べる。

2.1　HA クラスタシステム

HA クラスタシステムは、業務を稼動中のサーバと、待機系サーバとを用意しておき、両方のサーバにあらかじめ同じ OS や業務アプリケーションをインストールしておく方法である。この方法では、待機系サーバは電源 ON 状態で待機（ホットスタンバイ）しており、お互いのサーバの状態を監視する。動作手順として、①業務稼動中のサーバで障害が発生すると、②待機系サーバが障害の発生を検知し、③IP アドレスを引き継ぐことで、業務の引き継ぎを行う。

2.2　N+1 冗長化システム

N+1 冗長化システムは、業務を稼動中の N 台のサーバが、１台の待機系サーバを共有する冗長化システムである。待機系サーバには、OS や業務アプリケーションのインストールは不要であり、かつ電源は OFF 状態で待機する。動作手順として、①N 台の業務稼動中サーバのいずれかで障害が発生した場合、②障害が発生した業務稼動サーバの利用していたディスクイメージを待機系サーバへと引き継ぎ、③そのディスクイメージから待機系サーバをブートする。これにより、業務の引き継ぎを実現する。

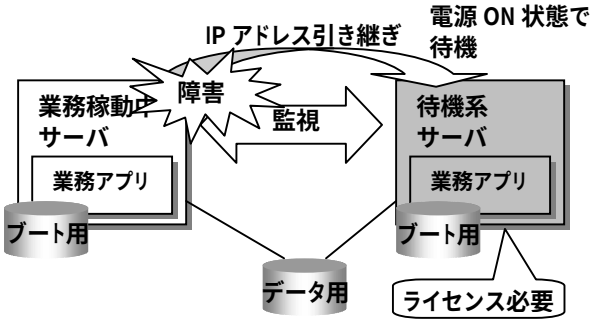

図 1　HA クラスタシステム

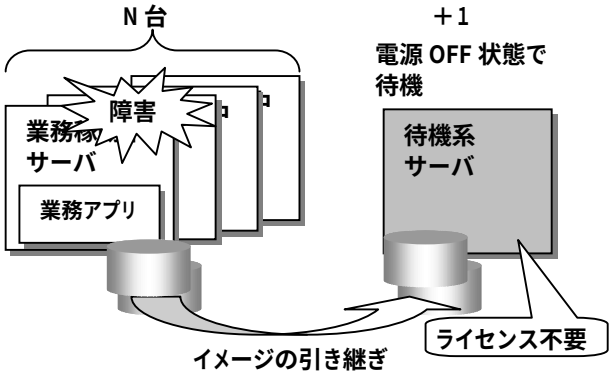

図 2　N+1 冗長化システム

2.3　サーバ冗長化方式の比較

上述のサーバ冗長化方式の比較を表 1に示す。表に示すとおり、HA クラスタは高速に業務の引き継ぎが可能である。しかし、複数の業務系サーバで稼動する全ての業務アプリケーションを待機系サーバでも構築する必要がある。かつクラスタ対応の業務アプリケーションが必要となる。また、HA クラスタソフトや待機系サーバのソフトウェアライセンスが必要となるため、高価なシステムとなる。

これに対して N+1 冗長化による業務の引き継ぎでは、イメージの引き継ぎのための時間と、電源 OFF 状態の待機系サーバを起動するための時間が必要となる。このため、HA クラスタに比べて障害からの回復時間が増大し、信頼性は劣る。しかし、複数台の業務系サーバに対して待機系サーバのアプリケーション構築工数が不要である。また、待機系サーバ分のソフトウェアライセンスが不要であるため、安価なコストでシステムの高信頼化が可能である。

上記の比較の結果、本研究の狙う、低コストなサーバ冗長化方式としては、N+1 冗長化が適していることが分かった。次に、N+1 冗長化の実現方式について検討する。

†（株）日立製作所　中央研究所

表 1　サーバ冗長化方式の比較

項目	HA クラスタ	N+1 冗長化
引き継ぎ時間	サーバ起動時間不要 セッション引き継ぎ可能	サーバ起動時間が必要 セッション引き継ぎ不可
構築工数	待機サーバへの構築要	待機サーバの構築不要
ソフトウェア	クラスタソフト必須	不要
ライセンス	待機サーバにも必要	待機サーバ分は不要

3. N+1 冗長化システムの実現方式

本研究の狙うサーバ冗長化方式として、N+1 冗長化が適していることを先に述べた。本章では、本研究の狙う (1)低コストで、(2)I/O オーバヘッドが無く、かつ(3)高速な業務引き継ぎが可能な方式を検討するため、N+1 冗長化の実現方法として、次の 3 つの実現案について述べる。

(a) ディスクイメージコピー方式
(b) I/O 切り替えサーバ方式
(c) ストレージパス切り替え方式

3.1 ディスクイメージコピー方式(図 3(a))

業務系サーバのバックアップイメージを準備しておき、ディスクイメージの引き継ぎを、あらかじめ準備しておいたバックアップイメージを待機系サーバのディスクへとコピーすることで実現する方式である。

手順：①業務系サーバのディスクのバックアップを取得、②運用開始、③障害発生、④待機系サーバへバックアップイメージをコピーして起動。

3.2 I/O 切り替えサーバ方式(図 3(b))

全てのサーバの I/O をネットワーク上に転送し、外部ディスクなどのデバイスとサーバの I/O とのマッピングを「I/O 切り替えサーバ」にて行う。これにより、障害発生時のディスクイメージ引き継ぎを、I/O 切り替えサーバの I/O のマッピングの変更で実現する方式である。

手順：①業務系サーバの I/O を外部ディスクにマッピング、②運用開始、③障害発生、④待機系サーバへ業務系サーバのディスクを再マッピングして起動。

3.3 ストレージパス切り替え方式(図 3(c))

SAN ブート環境では、サーバから SAN 上のディスクへのアクセスを制限するため、外部ディスク装置には、サーバと外部ディスク装置のディスクとをマッピングするセキュリティ機能が存在する。このセキュリティ機能を利用して、サーバとディスクとのマッピングを切り替えることで、ディスクイメージの引き継ぎを実現する方式である。

手順：①業務系サーバに外部ディスク装置のディスクをアクセス可能とするセキュリティ設定を実施、②運用開始、③障害発生、④業務系サーバからディスクをアクセス不可とし、待機系サーバからのみアクセス可能とするように変更して、待機系サーバを起動。

4.N+1 冗長化方式の比較

前述の３つの N+1 冗長化方式案を比較し、表 2にまとめた。表に示すように、ディスクイメージコピー方式は、比較的容易に実現できるが、切り替え時間が発生するため、業務の復旧までに時間を要していた。さらにバックアップイメージを取得する必要があるため、運用工数が増大する。I/O 切り替えサーバ方式では、ディスクイメージコピー方式の問題点を回避しているが、I/O を集中管理するサーバで I/O オーバヘッドが発生するため、システムの性能を犠牲にする。このため、システムの性能を維持するためには、サーバの追加が必要となる。これに対して、ストレージパス切り替え方式は、他の方式の課題を解決できており、もっとも優れた方式といえる。よって、ストレージパス切り替え方式を利用することで、(1)低コストで、(2)I/O オーバヘッドが無く、かつ(3)高速な業務引き継ぎが可能な N+1 冗長化システムを実現できるという結論を得た。

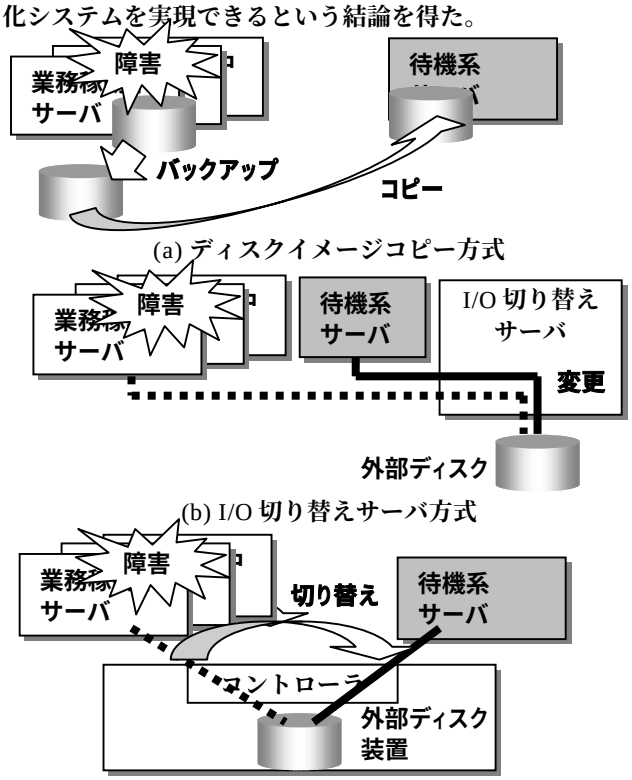

図 3　N+1 冗長化実現方式案

表 2　N+1 冗長化方式の比較

項目	ディスクイメージコピー方式	I/O 切り替えサーバ方式	ストレージパス切り替え方式
切り替え時間	× コピー時間必要 信頼性低下	○ コピー不要	○ コピー不要
運用前バックアップ	× 必要 運用工数増大	○ 不要	○ 不要
I/O オーバヘッド	○ なし	× I/O 性能低下 性能維持のため サーバ追加必要	○ 外部ディスク装置のコントローラで処理するため殆ど無し

5.まとめ

本稿では、システムの高信頼化を抵コストで実現するサーバ冗長化方式として N+1 冗長化方式に着目し、その実現方式案の比較を行った。この結果、外部ディスク装置のセキュリティ設定を利用する N+1 冗長化方式により、(1)低コストで、(2)I/O オーバヘッドが無く、かつ(3)高速な業務引き継ぎが可能なサーバ冗長化方式を実現した。

なお、本方式は、日立ブレードシステム BladeSymphony[1]にて実装済みである。

参考文献：[1]統合サービスプラットフォーム BladeSymphony
http://www.hitachi.co.jp/products/bladesymphony/index.html