耐障害に向けたサービスアプリケーション多重実行方式の提案

中村 暢達　　加藤 清志　　平池 龍一
ＮＥＣ インターネットシステム研究所

1. はじめに

今日の情報化社会では、IT サービスは必要不可欠であり、社会インフラの１つとなっている。そのため、常時サービスを継続し、たとえ障害が発生しても障害を顕在化させないサービス指向の耐障害性が要求されている。

それを実現するには、これまでハードウェアの領域で強化されてきた耐障害性の機能を十分発揮できるようにシステムソフトウェア側でサポートし、またできるかぎり自動的に障害復旧するフレームワークが必須となる。しかしながら、これまでの耐障害実装はプロプライエタリであり、アプリケーション非依存／汎用的な耐障害フレームワークは発展途上である。

本稿では、耐障害フレームワークの１つとして、サービスアプリケーション多重実行方式を提案する。提案方式は、さまざまな耐障害機能、例えば１つの計算機において障害が発生しても、瞬時に別の計算機で、サービスを継続させるようなフェイルオーバー機能を実現する。本稿では、提案方式の構成および動作について述べる。

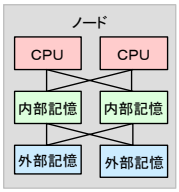
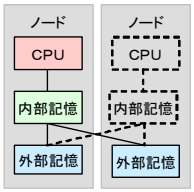
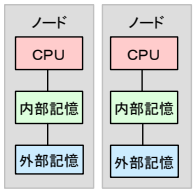
2. アプリケーション多重実行方式

2.1. 既存システムの耐障害

１つの計算機(ノード)における障害に対する対処方式としては、FT サーバ、クラスタ・ソフトウェアが知られている[1][2]。表１に、それらと提案方式との構成の差異を示す。

FT サーバやクラスタ・ソフトウェアにおいては、各部が冗長構成され、各々が同じ状態となる(同期する)ように構成される。今後、CPU などの処理速度が上がり、内部処理が高度化すれば、厳密な同期を行うことは困難となる。

表 1 既存耐障害システムとの構成の差異

FT サーバ	ｸﾗｽﾀ･ｿﾌﾄｳｪｱ	提案方式
		

Application Redundant-processing Mechanism for Service-oriented Fault Tolerance
Nobutatsu NAKAMURA, Kiyoshi KATO, Ryuichi HIRAIKE
Internet Systems Research Laboratories, NEC Corp.

クラスタ・ソフトウェアにおいては、冗長系をホットスタンバイしておき、障害発生時に、フェイルオーバーする手法が一般的である。しかしながら、外部記憶部(HDD 等)とプロセッサ部(CPU、内部記憶)とが、別個に管理されるので、これらの間にしばしば不整合が生じる。

このような問題点を鑑み、ノード(プロセッサ部－外部記憶部の組)単位で処理プログラムを多重実行し、厳密な同期を行わない冗長構成で処理を行う方式について提案する。厳密な同期を行わないとは、ノードの入出力さえ同一であれば、ノード内部の処理状態は問わないということである。

2.2. 従来方式

処理プログラムを多重化することで、システムの耐障害性を向上させる方式としては、N バージョン方式が知られている[3]。

N バージョン方式は、主にソフトウェアのバグに対して、有効な方式とされている。異なる開発工程(開発者)で、同一機能のソフトウェアを開発することで、同時にバグが顕在化する可能性を低減させ、いずれかのソフトウェアが障害に陥ってもサービスを継続する。しかしながら、開発コストが増大すること、多重実行するソフトウェアの出力を検証する機能の実装に手間がかかることから、実用性に問題がある。

2.3. 提案方式

我々の提案方式では、同一のソフトウェアを用いながら、その実行環境を多様化することで、同時にバグが顕在化する可能性を低減させる。また、アプリケーション入出力インタフェースのプロトコルとして、最も汎用的に使われている HTTP を対象とし、処理要求の多重化や、処理応答の比較を行う。このようにして、耐障害性の高いサービスシステムを容易に構築できるようにする。

ここで、実行環境の多様化とは、異なる OS、異なるメモリ量、異なるアプリケーション設定などもあるが、特にバグの同時顕在化の回避という意味で、異なる実行タイミングで冗長系を動作させることに着目する。実行タイミングを遅らせることで、現用系において障害が発生した後、冗長系でその障害が顕在化する前に、何らかの障害対処のアクションを実行できるようにする。

図 1 に、本提案の基本構成を示す。クライアントからの処理要求を現用系と冗長系に分配する。各系で、多重に処理が行われるが、冗長系への処理要求は、バッファを経由する。基本的に現用系の処理結果のみがクライアントに送信される。このようにバッファを含む構成とすることで、冗長系と現用系とが非同期で動作することを可能とする。図では、ミドルウェアとして実装しているが、プロキシサーバ型の実装でもよい。

冗長系での障害対処のアクションとしては、

1) 問題の処理要求を拒否し、処理を継続
2) ログレベルを変更して、処理を継続
3) ダンプ解析、デバッグ

などがある。障害の発生直前であることが分かっているので、単に障害を回避するだけでなく、障害原因の究明を含めた障害対処が可能である。

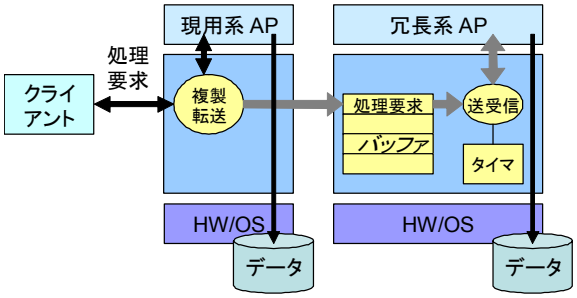


図 1 提案方式の基本構成

3. 障害対処動作

前章で述べた提案方式が、どのように機能するかについて述べる。システム全体の構成を図 2 に示す。通常、クライアントからの要求トランザクション(TX)は、現用系と冗長系で多重処理される。システムの状態は、自律管理フレームワーク[4]において、統合監視・分析が行われる。異常があった場合には、設定されたポリシーに従って、障害対処アクションが決定され、実行される。

システムの動作例を図 3 に示す。現用系は、要求 TX を受信し、応答 RX を送信する。冗長系は、現用系の処理を確認後に複製された要求 TX を処理し、検証用応答 RX を返す。そのため、冗長系は現用系に比べ、ある時間 t だけ遅延して処理を実行する。遅延の度合いは、処理トランザクション数を設定してもよいし、時間を指定してもよい。

現用系において、パニック障害を検知した場合、冗長系において、実行状態のイメージ保存を実行する。実行状態のイメージ保存は、各系の実装に仮想マシンを用いており、仮想マシンモニタの機能を利用している。冗長系は、ネットワークの設定のみを変更することで、現用系として機能を開始する。

代替リソース(別の仮想マシンホスト)では、前述の実行状態のイメージを活性化して、新たな冗長系として機能を開始する。活性化するまでに現用系の処理が進んでいても、バッファに保持された処理要求を順次処理することで、漸次通常状態に復帰する。

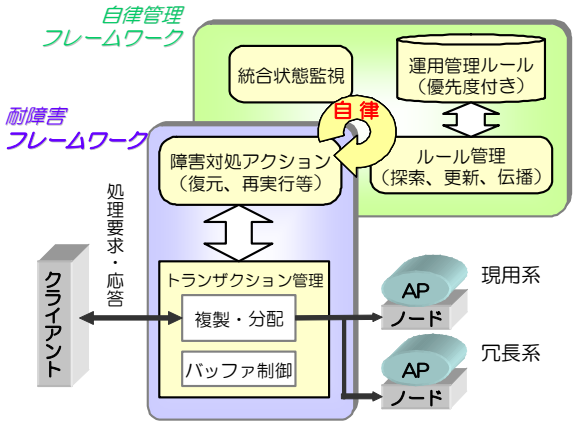


図 2 システム全体構成

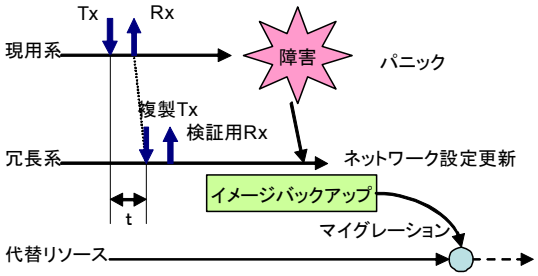


図 3 耐障害シナリオ例

4. おわりに

本稿では、サービスアプリケーションを多重に実行することで、システムの耐障害性を高める技術に関して、HTTP 層で処理要求を多重化し、多様な実行環境で、特に実行タイミングを遅らせて処理するフレームワークについて提案した。

今後は、アプリケーションのマルチスレッド処理対応、データベースなどの外部アクセスへの対応など、任意のアプリケーションを多重に実行できるように機能拡張を進める予定である。

参考文献

[1] “フォールトトレーラントシステムの設計”, 当麻,南谷,藤原,pp:150-172,槇書店,1991
[2] 例えば、http://www.beowulf.org/
[3] “Fault tolerance by design diversity: Concepts and experiments”, Avizienis, A., etc., IEEE Computer vol.17, no.8, pp:67-80, 1984
[4] “自律運用管理に向けたポリシー適用優先度の制御に関する一考察”,加藤,中村,平池,情処 66 全大, 6D-6, 2004.