高信頼 UNIX「風雅」の故障管理機構*

6M－2

中島 淳†, 伊藤 栄信‡, 片山 朝子§
(株) 富士通研究所¶
{nakajima, hidenobu, archan}@flab.fujitsu.co.jp

1　はじめに

我々はUNIXの信頼性を大幅に向上させるために，高信頼UNIX「風雅」を開発した．風雅は，カーネル内のOSサーバ（以後単にアクタという）と呼ばれるサブシステムを現用と待機とに二重化することにより，ソフトウェアおよびハードウェアの故障からの回復ができるような枠組を提供している．

風雅は，クラスタなどのような複数のノード上でSSI (single system image) を提供するUNIXであり，それらの上で動作するアクタ間のIPCを使って，OSの機能が実現されている．本稿では，風雅でのノード，アクタの故障管理を議論する．

2　故障管理モデル

風雅での故障管理モデルは，正常動作，故障検出，故障隔離，故障回復，システム再構成，というサイクルに基づいている．

故障検出

風雅の枠組における故障検出は，前述のようにノード，アクタの故障検出を行なう．本稿では，ノードの故障検出機構は述べないが，誤検出を中心をとして難しい問題が多数ある．ノードの故障検出機構によって，1つのノードが故障するとその上のノードのすべてのアクタも故障することを保証している．

アクタの単体の故障検出も単純ではない．特に単純な論理矛盾以外に，無限ループ，デッドロックなどまでも検出することになると，複雑な機構が必要になる．クライアント／サーバ型モデルでは，(1) サーバ・アクタ内だけで検出する，(2) クライアント・アクタがタイムアウトを検出するの2つがある．(2) の方が広い検出範囲を持つが，タイムアウト値などの調整，誤検出の回避などの別の難しい問題があったので，今回は (1) を採用した．

故障隔離

風雅では，性能のために，1つのノードにあるすべてのアクタは，アドレス空間を共有している．アクタ故障が生じても，データ破壊の範囲を特定できないので，ノード単位で故障隔離を行なう．つまり，アクタ故障はそのノードの故障として扱う．

故障回復

故障回復は，待機アクタが引継ぎを行なう．故障回復が正常に終了すると，その待機アクタは，新しい現用になる．また，故障回復が失敗することもあり，その場合には，消失したことになる（後述）．他ノードからも参照されている資源を管理しているアクタの消失は，システム全体に波及する可能性もあるが，そのアクタ自身ではその後のシステム全体の振舞い（例えば，システム全体をクラッシュさせる）を決めるのは難しいので，この決定は故障管理機構が行なう．また，風雅ではいわゆる部分機能閉塞はまだサポートしていない．

また，一般には，複数の現用アクタが同時に故障した場合には，回復の順番が問題になることがあるが，回復時に依存関係を作ると回復が複雑になるので，風雅では（アクタ間のIPCによるプロトコルを変えてでも）依存関係をすべて解消するという戦略を用いた．

システム再構成

現用・待機アクタのいずれが故障した場合でも，結果的には待機アクタが失われるので，新しい待機アクタを配置・生成する必要がある．一般に待機アクタの配置は，デバイスアクセスの可否，負荷などの様々な要因によって決められる．待機アクタの生成は，ノードまたは

*Fault Management for Highly Available UNIX "FUGA"
†Jun Nakajima
‡Hidenobu Ito
§Asako Katayama
¶Fujitsu Laboratories Ltd.

待機アクタのみの動的な再起動によって実現する.

3 Fault Manager

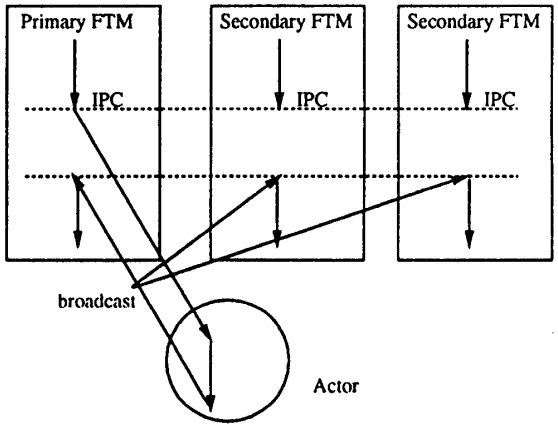
風雅では, Fault Manager (FTM) という1つのアクタが, 上のモデルに基づいて故障を管理している. 主な機能として, (1) 現用・待機アクタの管理, (2) 引継ぎ制御, (3) 新待機アクタの用意, (4) 資源回収のための消失通知, である. さらにFTMの耐故障性を保証するために, FTMは各ノードに複製配置される (後述).

FTMの機能

ノードの故障検出機構により故障検出はFTMに通知され, 現用・待機アクタの故障が判断され, 適切な待機アクタに引継ぎ指示が送られる. 一般には, ノード故障では複数の現用・待機アクタが故障し, FTMは複数の引継ぎ指示を異なるノードに存在する待機アクタに送ることになる. 現用アクタは, 引継ぎ指示を受ける必要はない. 待機アクタからの回復通知により, FTMはその待機アクタを新しい現用アクタに変える. その後, FTMは, システム再構成の要件に基づいて新待機アクタを配置・生成する. 待機アクタの故障のときも, 別の待機アクタを配置・生成する.

また, クライアント・アクタが消失したときは, サーバ・アクタは自分の管理資源を回収するため, 消失通知を受けとる必要がある. 同様に, サーバ・アクタが (耐故障性を持たないか, 回復が失敗して) 消失したときは, クライアント・アクタも消失通知を受けとる必要がある. このためには, (1) すべてのアクタにFTMが通知する, または, (2)FTMがアクタ間のクライアント/サーバの関係を保持し, 関係あるアクタにのみ通知する, の2つの方法がある. 実現は, (1) が単純であるが, スケーラビリティ, 効率を考えると, 一般には (2) の方が優れているので, (2) を使っている.

設計バグ, コーディングバグがある場合には, 正常な故障管理が行なわれないので, FTM自身は, 一般のコードに比べ格段に高い品質を要求される. 設計においては, 単純さを特に目標においた. FTM自身は, 正常時にはほとんど動作せず, 性能要求が厳しくないので, 耐故障性の実現がなるべく単純になるように設計した.



FTMの耐故障性

回復が失敗すると, 同時に複数のノードが故障することがある. したがって, それ自身の耐故障性を保証するために, FTMを全ノードに配置する.

IPC同期方式という方法で, FTMを単一スレッドにし, IPCの入出力を放送することにより, 同じ状態を複製する. FTMが同期型IPCのみで他アクタと通信することを利用している. 初期値を同じにし, 1つの現用 (primary) FTMに従って, 残りの待機 (secondary) (複数の) FTMが, 現用FTMの他のアクタへのIPCを繰返す. 現用FTMのIPCは実際に実行されるが, 待機FTMのIPCは実行されず現用FTMが得た結果が放送される (図参照). また, 現用FTMへのIPCは, すべての待機FTMに放送され, 現用FTMのみ結果を返す. この結果, すべてのFTMがIPCが終了した段階では同じ状態になっている. 現用FTMが故障すると, その通信層があるルールによって次の現用を決める. なお, FTMが回復できなくなると, システムを停止 (クラッシュ) させる.

新しいノードがシステムに加わった場合には, 現用FTMの状態をコピーする必要がある. 故障中はメモリ獲得が不可能なときがあるので, FTMはメモリを静的に獲得しており, FTMのメモリ管理の情報を使うことにより, 新しいノード上のFTMも同じ状態になり, 現用FTMに同期して動作するようになる.

参考文献

[1] 岸本他,"高信頼Unix「風雅」", 第52回情報処理全国大会論文集 6M-1,3-6