計算機アーキテクチャ　89－8
（１９９１．　７．　１９）

協調分散型制御におけるフォールトトレラント方式

神余浩夫　　竹垣盛一
三菱電機株式会社　中央研究所

協調分散型制御システムは管理系も待機冗長系も持たない、自律的な制御要素から構成される均質な分散システムである。制御要素の相互作用によって対域的秩序を形成し、システムの状況や環境変化に対してその秩序を自己組織的に維持できる。例えば、ある制御要素が故障すると、まずその隣接装置がバックアップを行なう。局所的なタスク分配によってタスクは遠い制御要素へと拡散していき、最後に全体が処理を分担する状態に落ち着く。協調分散型制御システムアーキテクチャ:CODAは、このような協調バックアップ方式により耐故障性を高めている。

The Fault Tolerant Method of the Coordinative Decentralized Control System

Hiroo Kanamaru and Morikazu Takegaki
Mitsubishi Electric Corporation
8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo 661, Japan

The coordinative decentralized control system is a homogenious system that consists of symmetrically coupled autonomous controllers, and has neither supervisory element nor reserved one. The total system has a grobal reguration made of the ineractions among the elemental controllers, and can keep it in changing its conditions and environments by coordinating of them. For example, when one controller is broken, firstly its neighbor controllers backup its tasks. The local distributions of tasks in every controllers-pairs propagats the tasks to far controllers in gradually. Finally, the total system gets the stable load balancing state that all controllers take on same loads of tasks. The COordinative Decentralized control system Architecture: CODA which is we proposed is a well fault tolerant system, as taking on such the coordinative backup.

1.はじめに

産業用計算機制御システムは汎用計算機システムに比べリアルタイム性や信頼性に対する要求が厳しいが、特に大規模かつ複雑化するほど、多様性、柔軟性、耐故障性が重要となる。具体的には、プラントや生産ライン運転中のシステム保守点検、無停止機能拡張や故障時の縮退運転などの機能である。このような柔軟で融通性の高いシステム構築を可能にするためには、システムの構成要素が均質で対称的であることが望ましい。しかし、従来のフォールトトレラント構成は常用系の他に待機系や冗長系が存在する非均質であったため、制御システムの状態や対象プラントの状況に応じて制御システムを動的に構成することが困難であった。筆者らが提案する協調バックアップ方式[1]は、全ての制御要素に制御処理を割り当てて待機系を排除し、それぞれの時間および資源余裕により相互にバックアップする。制御要素間の相互作用は隣接するものとのみ局所的に行われるので、バックアップ動作は故障装置を中心に拡散的に進み、ついにはシステム全体に均等に処理分散された状態に落ち着く。

このような、均質な要素の自律的動作によりシステム全体が機能するシステムとして自律分散システムが提案されている[2][3]。しかし、制御要素故障時の障害隔離と健全要素の動作性を指向したため、システム全体の機能維持や協調機構に関する議論が希薄である。自律分散に対して要素間の相互作用を重視したシステムを、協調分散システムという。システムの要素群が協調的に動作しているとき、全体システムには平衡な大域的秩序が形成されていると考えることができる。制御要素の故障や負荷変動といった状況変化に対して制御システムが自己組織的に適応するためには、全体システムの秩序形成の挙動、協調ダイナミクスが安定であることが条件となる。筆者らは以上の原理に基づく協調分散制御アーキテクチャ: CODA (COordinative Decentralized control system Architecture) を提案している[4][5]。CODA において協調バックアップを達成する制御要素の自律動作、相互作用は協調プロトコルとして定義されており、現在ワークステーションを使った機能検証が行われている。

本文では制御システムにおける協調分散概念と、それに基づく具体的なアーキテクチャであるCODAを紹介する。特に、CODAで特徴的な柔軟性と可用性を高めたフォールトトレラント機能である協調バックアップ方式とその実現について述べる。

2.制御システムのフォールトトレラント

2.1.従来方式の分析

システムのフォールトトレラント方式について、産業用制御計算機および計測制御システムに焦点を絞り、そこで行われている処理(タスク)のバックアップ動作について考える。従来方式は以下の３種類に分類できる。

・冗長実行方式

ハードウェア的な多重化構成においてそれぞれが同一の処理を行う。故障の際には正常装置の応答だけが出力される。故障検出に連動する出力阻止回路を持つものと、多重装置の間に出力比較回路を持つものがある。同一の処理を異なるアルゴリズムにより処理し、その結果を比較する手法もここに含まれる。信頼性と高速応答性に対する要求が厳しい場合に採用される。

・待機冗長方式

常用装置に対して待機装置が存在し、常用装置の故障時には待機装置が処理を継続する。待機冗長系では故障検出後にタスク実行を待機系に切り替える時間分の遅延が発生する。従って、高速応答性が要求される場合には適していない。タスクのプログラムは待機系が予め保持している場合、常用系と共有している場合および故障後転送する方式がある。また処理引継に必要なデータは、物理的に共用する場合、常時適当なタイミングで転送する場合、および故障後常用系から読みだして待機系に転送する方式がある。

・再試行方式

ノイズによる信号エラーか誤動作が原因でデータ転送や計算処理が中断した場合、その処理を再試行してエラー回復が可能である。再試行方式はハードウェア的な冗長性が不要である反面、処理をやり直す時間だけの遅延が生じる。また、再試行ができない故障の場合は対応できない。従って、応答性と信頼性への要求が比較的弱い場合でのみ有効である。

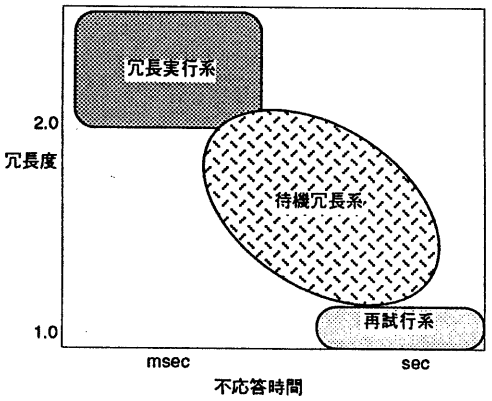

Fig.1 フォールトトレラント方式の比較

これらの方式の比較をFig.1に示す。縦軸はシステムの多重度、横軸は故障後の復旧速度あるいは不応答時間を表している。フォールトトレラント構成における多重度、冗長性は信頼性に対する保険のようなものであり、高信頼性を要求するとその設備コストは増大する。しかし、最近の制御計算機は部品数低減によって単独構成でかつての二重系と同等MTBFを達成しており、その処理速度、データ転送速度も大幅に向上している。そこで、要求される信頼性と性能を低多重度で柔軟に構築できる冗長構成が求められている。

2.2.制御アプリケーションの種類

Fig.1に示したように、高速応答性と信頼性に対する要求が非常に厳しい場合、冗長実行方式しかシステム構成方法はない。しかし、制御アプリケーションによって要求する応答性、信頼性は異なるので、それに適したフォールトトレラントシステム構成を採用するべきである。ここでは、Table.1に示した制御問題に適したシステム構成を分析する。

通常、制御周期が1msec以下の高速応答性を要求される場合、故障検出回路と切り替え装置の高速性だけでなく、CPUと I/Oを密に接続する必要がある。このため、待機冗長構成としても二重化とほとんど同じ構成になってしまう。従って、ロボット制御や信号処理のようなアプリケーションは低多重度の柔軟なシステム構築が難しい。

次に、制御周期が数msecのプロセス制御の場合、常用系から待機系への切り替えにおけるデータ転送が1msec程度で完了するならば、Fig.1の待機冗長方式の範疇に含むことができる。もちろん、プロセスデータや制御装置間通信に高速なものが要求される。最後の化学プラントや圧延工場などの一般産業プロセスのセルコントロールやモニタリングでは制御周期が数100msecから秒の規模である。このようなシステムはLANと制御用計算機で構成されることが多く、それだけ処理内容が複雑になっている。応答性に対する要求が緩く、制御周期程度の時間で故障復旧または処理切り替えができれば十分満足できる。低冗長度のフォールトトレラント構成が最も効果的なアプリケーションであり、協調分散型制御システムが対象とするのもこの種類である。

事務処理用の計算機システムでは、故障により中断したサービスや処理の確実な継続を重視する。一方、リアルタイム制約のある制御システムでは、制御装置の故障中でも制御対象との周期的入出力タイミングを遵守するために、故障中の処理をすべて再試行することはない。筆者の経験によると、故障診断回路により故障検出を行なっている待機冗長型制御システムでは、故障時の予備系切り替え時間を制御周期の倍として設計している場合が多い。制御応答出力間際に故障してその周期の出力ができず、故障検出できてからでは次の周期における出力が間に合わない場合、二制御周期の不応答時間となるからである。一般に、プロセス制御のような周期制御タスクの制御周期は制御対象の動特性の

Table.1 制御アプリケーションの特徴

種類	応答時間	複雑性	タスク数
ロボット制御	μ～1msec	単純	<5
プロセス制御	1～10msec	中	5～20
機械制御 装置制御	<1sec	中	10～30
セルコントロール モニタリング 帳票処理	200msec ～1sec	複雑	50～100

時定数に比べ十分小さいので、制御応答が一周期欠落したからといって制御対象が致命的状態に陥ることはない。従って、周期的制御システムでは故障中の処理を無視することにより、低多重度のフォールトトレラント構成が可能である。

3.協調分散システム概念

3.1.協調バックアップ方式

待機冗長システムの多重度を下げていくと、n 個の常用系に対して待機系がひとつといった構成になるが、このとき待機系は全てのタスクをバックアップ可能なように全てのタスクのプログラムと内部状態を保持するために、メモリ資源と通信量が集中する。システムにおいて待機系が特殊な役割を与えられているため、常用系、待機系相互の融通性に欠ける。柔軟で自由度の高いシステム構成を行なうには、構成要素が均質かつ対称であることが望ましい。

協調バックアップ方式(coordinative backup)とは、システム中から待機系を排除し待機系にも制御処理を割り当て常用系として扱う。それぞれの制御装置は待機系に処理を回した分だけの処理時間の余裕を持ち、故障時にはその余裕分により故障装置のバックアップを行なう。すなわち、システムの健全要素が協調してバックアップ動作を果たす。システムの全ての制御要素が処理切り替えやデータ転送の処理を共用でき、その対称性は要求される多重度に適した構成変更を容易にする。

具体的な動作としては、それぞれの制御装置がメモリ上に常駐させた制御タスクについて任意装置が実行担当し、他はそのタスクを休止させておく。適当なタイミングで実行担当装置はタスクの状態を他装置に転送しておいて、故障時には処理を他の何れかの装置に切り替える。これはソフトウェアの待機冗長方式と考えることができ、よく似た例が商用フォールトトレラント計算機に採用されている。GUARDIAN[6]は異なるプロセッサ上に主系と予備系からなるプロセス対を生成し、入出力操作などが行われると主系から予備系に操作に関する情報をメッセージとして転送する。このチェックポイント(check point)情報には主系の故障の際にその処理を引き継ぐために必要なすべての状態情報が含まれており、予備系は故障直前のチェックポイントから処理を再開する。チェックポイントの設定は、間隔を短くし過ぎるとデータ転送のオーバーヘッドが大きくなり、長いと処理再開の時間遅延が大きくなる。効果的な設定はアプリケーションの性質とフォールトトレラント要求に依存する。協調バックアップ方式ではタスク間のデータ一貫性だけでなく、制御要素群が協調的にバックアップ動作を行なうためのタイミング、バックアップポイント(backup point)も重要である。

3.2.局所スケジューリング

協調バックアップ方式ではタイミングの問題ともうひとつ、タスクの割当(スケジューリング)問題を解かなくてはならない。協調分散システムは制御要素の均質性を特徴とするので、全体を管理するスケジューラは排除している。また、分散システム全体のスケジューリングを制御システムのリアルタイム制約下で解決することは困難である。そこで、それぞれの制御要素はバックアップ動作を局所的範囲、隣接する制御要素間についてのみ行なうとする。すると、故障直後はその両側の制御要素によりバックアップが行なわれ、その次のバックアップポイントでさらに遠くの要素がバックアップに参加する。これを繰り返すことにより次第にタスクは拡散的に分散され、最後は全ての制御要素がバックアップに参加してシステム全体にほぼ均等なタスク分散状態となる。協調バックアップの動作例をFig.2に示す。それぞれの要素が扱うスケジューリング問題は両隣の要素との均等負荷分散のためのタスク分配問題であるから、協調動作一回当りの処理時間のオーバーヘッドがリアルタイム制約に悪影響を及ぼすことはない。そのかわり、全体が落ち着くまでに数回の協調動作が必要である。

このような自律的に動作する制御要素群により構成され、それらの自律的な相互作用により全体システムが協調的に機能するシステムを筆者らは協調分散システムと呼んでいる。協調バックアップ方式は協調分散システムにおけるフォールトトレラント方式の一例である。

3.3.協調ダイナミクス

柔軟性、拡張性に優れたシステム構成法としては、制御要素が他の要素に対し自律的に動作できる自律可制御性と、ある要素の故障に係わらず残りの要素が動作できる自律可協調性を特徴とする、自律分散システムが知られている()。しかし、これらの性質は故障時の要素の動作性を保証するが、故障装置のバックアップなど全体システムとしての協調機構について明言していない。

協調分散システムが協調的であるとき、システム全体における大域的秩序が形成されている。システム状態が変動しても秩序状態を自己適応によって維持するためには、秩序状態が状態空間において安定であることが必要である。すなわち、分散要素群がつくるポテンシャル場が勾配系であり、大域的秩序状態がポテンシャル場の基底状態となっている場合である。もし、ポテンシャル場が勾配系でなかったり、期待する状態が安定ではなかったら、自律要素の相互作用は期待しない秩序状態に遷移したり、無秩序な発散状態に向かったりする。この全体システムのポテンシャル挙動を協調ダイナミクス(coordinative dynamics)と呼んでいる。協調ダイナミクスが安定であるとき、秩序状態は勾配ポテンシャル場の安定点に位置するので、自律要素の相互作用によってシステムは秩序状態に有限時間で到達できる。

自律分散システムにおける局所的相互作用のダイナミクスと全体の大域的秩序の関係と、その協調原理に関する理論は湯浅、伊藤によって構築されつつあるが[13]、安定な秩序状態を形成する制御要素の自律的動作および相互作用を設計するのに十分な成果はまだ得られていない。しかし、定義した制御要素の動作に基づく秩序形成を全体システムの協調ダイナミクスの安定性解析によって検証することは可能であり、協調分散システムのボトムアップ的な設計における効果的な解析、検証手法になる。

4.協調分散制御アーキテクチャ

4.1.システム概要

これまでの協調分散システム概念に基づいた制御システムアーキテクチャ、CODAについてその概要を説明する。

CODA は周期制御が数10～100msecのセルコントロールやモニタリングを対象とした分散制御システムであり、集中管理系やハードウェア的な冗長系を持たない水平分散型システム構成を特徴とする。CODA のシステム構成例を Fig.3 に示す。それぞれの制御装置は自律的に制御タスクを処理できるように、制御タスクとOSおよび協調動作のプログラムをメモリに保持する。対象プラントの各種センシング情報はリモートI/O装置からデータバスにブロードキャストされるので、制御装置はタスク実行に必要な全ての

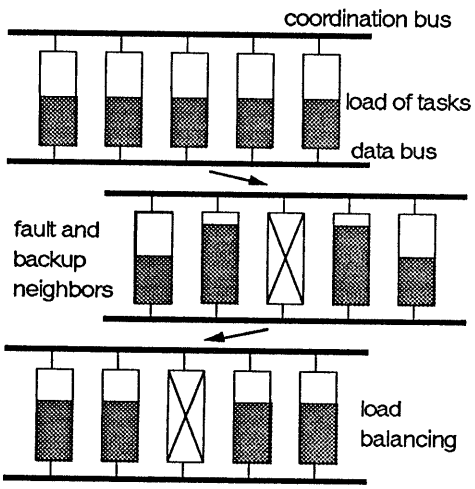

Fig.2 協調バックアップ方式の動作

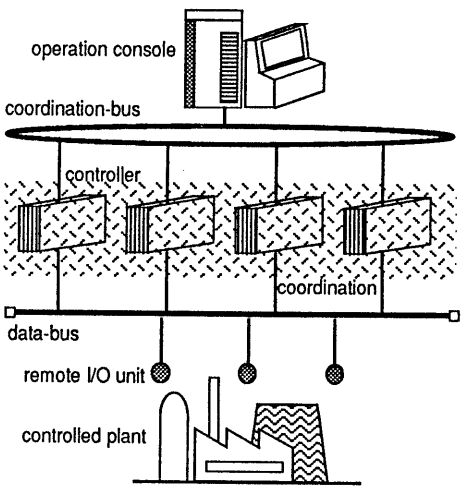

Fig.3 協調分散型制御システムの構成

プロセスデータを毎制御周期に得ることができる。各制御装置は協調バス(coordination bus)に接続されており、制御装置間の通信や相互作用はこのバスを介して行われる。制御装置の故障検出はそれぞれに装備されたウオッチドックタイマなどの診断回路により検知され、隣接制御装置に通信される。データバスまたは協調バスが十分な通信容量を有していればひとつのバスで共用してもよい。図では運転員の制御盤もバスに接続されているが、プラントおよびシステムのモニタリングと特権的操作を行い、制御装置の通常の協調動作には関与しない。

制御タスクは対象に直接作用する制御システムにおけるアプリケーションである。特別なハードウェアを必要としたりリアルタイム制約が極端に厳しいタスクはCODAに適さない。また、タスクの実行時間がデータ転送時間に対し比較的大きく、バックアップの際に受け渡すデータが少ないものの方が、CODAにとって扱いやすい。協調バックアップ方式を適用するので、同一のタスクが複数装置においてスタンバイ状態になっていて、隣接装置との処理時間としての負荷分散スケジューリングによって実際に実行するタスクを決める。チェックポイントはタスク終了時であり、そのときリモートI/Oへの出力と隣接装置へのチェックポイントデータ転送を行う。バックアップポイントはそれぞれの装置で一定時間周期で生起して、隣接装置とのタスク分散と自分の実行するタスク選択を行う。

4.2.協調プロトコル

協調分散制御システムにおいて目的とする秩序形成を可能にするには、制御要素の自律的動作および相互作用による協調ダイナミクスが安定であることが必要条件である。そして、このために制御要素が遵守すべく動作と相互作用に関する手順が協調プロトコルである[8]。協調プロトコルの構成をFig.4に示す。協調プロトコルはリアルタイムOSとアプリケーションの間に位置し、データ転送、スケジューリングや同期機構など協調化に必要な機能を提供する。

・通信プリミティブ層

実装において通信バスやネットワークの低レベルプロトコルの差異を吸収し、上位層が必要

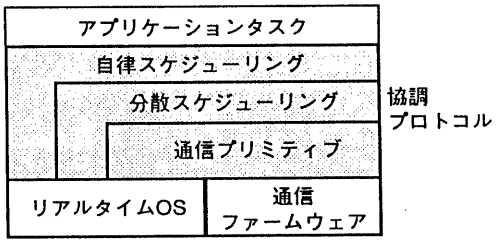

Fig.4 協調分散プロトコルの階層構造

とする通信機能を提供する機能層である。均質で自律的な協調分散構成の通信モデルには、サーバー・クライアント型あるいはマスタ・スレーブ型のような非対称型でなく、非同期のメッセージ交換モデルを採用している。データバスのプラントデータ通信ではブロードキャスト、協調バスの隣接装置間通信および故障通知ではメッセージ転送をサポートする。隣接装置の故障が通知されると故障装置との通信を切断し他の(ひとつ向こう側の装置を新たな隣接装置とみなして通信を再開するが、この隣接装置の切り替えもこの層で行っている。

・分散スケジューリング層

協調バックアップ方式の制御装置は通信プリミティブ層の機能を用いて隣接装置と相互通信を行い、両者間の負荷均等化のために装置間タスク分配を行う。この隣接装置間の相互作用を分散スケジューリング層はサポートする。タスクがチェックポイントに到達すると、処理引継に必要なデータをまとめて予備系でもある両隣接装置に転送する。受信側はそのデータを自メモリに展開する処理を、割り込み処理する。バックアップポイントでは、隣接装置と Fig.5 のアルゴリズムにより負荷分散となるようにタスクの割当を行う。自己の担当していたタスクをふたつに分け、それぞれを左右両隣の装置とのタスク割当に提供する。このとき、装置間で担当タスクを教え合うので、両隣接装置により故障装置が担当していたバックアップすべきタスクを認識できる。このバックアップポイント処理は制御タスクに対して優先的に先取り処理される。チェックポイントとバックアップポイントの関係をFig.6に示す。

```
0:Separates tasks into two groups.
1:Send one group to the right controller.
2:Recieve the task group from the left.
3:Scheduling tasks of two groups for load
    balancing between this controller and the left.
4:Send tasks allocated on the left
                    to the left controller.
5:Receive tasks from the right.
6:Combinate two groups allocated
                    to this controller.
                .....
    [to the autonomous scheduling]
```

Fig.5 分散スケジューリングのアルゴリズム

・自律スケジューリング層

制御装置に割り当てられたタスクの実行順序割当、管理を行う。リアルタイムシステムでは異なる周期のタスクの刻限(deadline) までの終了を保証するタスクスケジューラが不可欠である[9][10]。制御装置に処理時間余裕があれば問題はないが、全てのタスクを処理できない過負荷状態の時は、担当するタスクの中で重要性の低いタスクを次のバックアップポイントまで休止させる。次の分散スケジューリングで制御装置に余裕ができれば、休止していたタスクは実行可能になる。すなわち、一時的な優先度の低いタスクの実行抑制により優美機能縮退(graceful degradation)を発揮する。CODAが扱う周期的制御タスクは動的に生成されることなく、しかも実行時間が設計時に与えられるので、静的スケジューリングで十分である。ただし、タスク割当が変わるバックアップポイント毎にスケジューリングする必要がある。

4.3.CODAの協調ダイナミクス

協調分散システムは制御要素の相互作用によって、設計者の目的とする大域的秩序を形成し協調的な要素動作を実現する。この秩序形成の挙動は全体システムの協調ダイナミクスの安定性解析によって検証できる。CODA の制御要素の相互作用は分散スケジューリング層で定義される隣接装置間のタスク分散であるから、制御装置の負荷(タスク処理時間)分散状態のポテンシャル関数をそれから構成し、負荷分散状態の安定性を調べる。

制御周期t回目における制御装置i,jの負荷 をそれぞれxit, xjtとして、その周期の分散スケジューリングによる負荷移動量をa(xit−xjt)とする。ここで、aは設計時に与えらえれる定数である。毎周期にすべての隣接装置間で分散スケジューリングが並行して行われるので、制御装置 n 台の負荷(x1, x2, …, xn) は次式で表される。

$$\begin{bmatrix} x1 \\ x2 \\ \vdots \\ xn \end{bmatrix}_{t+1} = \begin{bmatrix} -a & a & & 0 \\ a & -a & & \\ & & \ddots & \\ 0 & & & 0 \end{bmatrix} + \begin{bmatrix} 0 & & & 0 \\ & -a & a & \\ & a & -a & \\ 0 & & & 0 \end{bmatrix} + \cdots + \begin{bmatrix} 0 & & & 0 \\ & \ddots & & \\ & 0 & 0 & \\ & & -a & a \\ 0 & & a & -a \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ \vdots \\ xn \end{bmatrix}_t + \begin{bmatrix} x1 \\ x2 \\ \vdots \\ xn \end{bmatrix}_t$$

$$= a \begin{bmatrix} -1 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 0 & & & 1 & -1 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ \vdots \\ xn \end{bmatrix}_t + \begin{bmatrix} x1 \\ x2 \\ \vdots \\ xn \end{bmatrix}_t$$

$$\therefore X_{t+1} = aA_n X_t + X_t = [\, aA_n + I \,] X_t$$

ただし、An(n×n)、X(n)であり、Iは単位行列である。いま、[aAn ＋I] の特性方程式を考える。

$$F_n = \det [\, \lambda I - [\, aA_n + I \,] \,]$$

$$= a_n \det [\{ (\lambda - 1) / a \} I - A_n \,]$$

An の固有値が、2 cos(k π /n) − 2 , k=0,1,…,n-1 であることから(14)、特性方程式の根は次式となる。

$$\lambda = 2a (\cos(k \pi / n) - 1) + 1 , \quad k=0,1, \cdots, n-1$$

ただし，k=0は Σxi ; (i=1,n)＝constに対応する固有値である。有限回の分散スケジューリングでXtがある状態に収束し安定となるのは、[aAn ＋I] のk=0 以外の固有値が単位円内に存在するとき(|λ|<1)であるから[11]、(0<a<1/2)のときである。すなわち、システム全体が安定収束するためには、隣接装置間の負荷移動量は

両者の負荷の差の半分以下としなければならない。

5.結論

自律的な制御要素の局所的な相互作用により全体システムの大域的秩序を形成し、システム全体が目標状態を自己組織化する協調分散システムと、その性質の応用例としての協調バックアップ方式について述べた。CODAの協調プロトコルはUDP(User Datagram Protocol)[12]を用いてUNIXワークステーション上に実現されており、ダミーアプリケーションによる協調プロトコルの機能評価および協調ダイナミクスのシミュレーションを行なっている。チェックポイントやバックアップポイントの設定と動作はアプリケーション依存性があるので、ダミーでない制御アプリケーション例題を用いた実験を予定している。

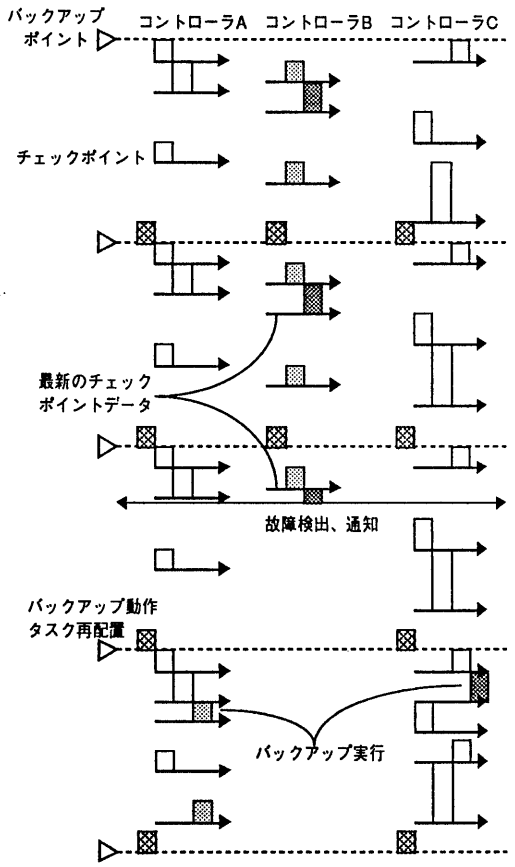

Fig.6 チェックポイントとバックアップポイントの動作

参考文献

1)神余,竹垣:協調分散制御システムの設計手法に関する考察,第9回自律分散システム研究会講演論文集,17/20(1989)

2)森,宮本,井原:自律分散概念の提案,電気学会論文誌C,104-12,303/310(1984)

3)伊藤他:自律分散システム特集,計測自動制御学会学会誌,29-10,877/952(1990)

4)神余,竹垣: 協調分散制御システムアーキテクチャ:CODAの概念モデル,計測自動制御学会論文誌,27-4,458/465 (1991)

5)H.Kanamaru,M.Takegaki: The Coordinative Decentralized Control System Based On Local Communications, 16th Annual Conference of IEEE IES, 535/538(1990)

6)J.Gray et.al.,渡辺榮一編訳:フォールトトレラントシステム,p.128,マグロウヒル(1986)

7)K.Mori et.al.: Autonomous Decentralized Software Structure and Its Application, Fall Joint Computer Conference Nov.1986, 1056/1063(1986)

8)神余,竹垣:リアルタイム性を指向した協調分散プロトコル,第1回自律分散システムシンポジウム論文集,63/68(1990)

9)J.A.Stankovic et.al.: Evaluation of a flexibletask scheduling algorithm for distributed hard real-time systems, IEEE Trans. Computers, 34-12,1130/1143(1985)

10)C.L.Liu,J.W.Layland: Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, J.ACM, 20-1,46/61(1973)

11)R.Bellman: Introduction to Matrix Analysis,p.66 TATA McGrow-Hill(1974)

12)D.Comer: Internetworking With TCP/IP,p.120,Printice-Hall(1980)

13)湯浅,伊藤:自律分散システムの構造理論,計測自動制御学会論文集,25-12,1355/1362(1989)