オペレーティング・システム 56−14
(1992. 8. 19)

協調タスクバックアップ方式のリアルタイム性能評価

神余浩夫
三菱電機(株)中央研究所
(661) 兵庫県尼崎市塚口本町 8-1-1

協調バックアップ方式とは、近傍系において周期的な動的再構成を繰り返し、処理装置が相互にタスクをバックアップするリアルタイム分散システムである。この機構としてタスク処理の引き継ぎデータを転送する checkpoint (CP) 機構と、タスク再配置を与える Remapping Scheduler (RS) と、処理装置内のタスクのリアルタイム実行制御する Local Scheduler (LS) を有する。同方式でタスクのリアルタイム性を保証するには、CP データ転送やタスク再実行などの処理余裕を評価する必要がある。そこで、協調タスクバックアップ方式の種々のオーバーヘッドを定量的に解析し、そのうえでリアルタイムスケジューリング条件を導出したので、報告する。

Real-Time Schedulability Analysis of the Coordinative Task Backup Policy

Hiroo Kanamaru
Central Research Lab. Mitsubishi Electric Corp.
8-1-1 Tsukaguchi-Honmachi Amagasaki, Hyogo, 661 JAPAN

The coordinative task backup policy is a fault tolerant method that elemental controllers are reconfigured by themselves in neighbors periodically, and backup their tasks with each others. For such behavior, the distributed real-time system has following mechanisms; a checkpoint data transfer protocol, Remapping Scheduler(RS) which gives task allocations in neighbors and Local Scheduler(LS) which executes realtime tasks in each one. To guarantee real-time tasks, we need to verify and analyze the schedulability of tasks with overheads and capacities for these mechanisms. This paper shows the quantitative analysis of these overheads, and the scheduling condition to guarantee realtime tasks under the coordinative task backup policy.

1 諸言

プラント運転制御システム、CIM/ FA ロボット制御装置、航空管制システムや音声動画マルチメディアシステムなど、現実世界との応答を行なう情報処理システムは、情報処理の論理的正当性だけでなく、応答の時間的制約も満足しなければならないリアルタイムシステムである。特に産業分野における情報制御システムは、制御対象の動特性に追従するリアルタイム性だけでなく、システム故障に際して制御機能が損なわれないフォールトトレランス性も同時に要求される。従来、フォールトトレランス方式としてホットスタンバイ / コールドスタンバイの多重冗長化構成が多く適用されてきた [1]。しかし、計算機システムの処理能力および耐久性の著しい向上により、従来なら制御用途に特化したアーキテクチャでなければ得られなかった性能が比較的一般的なアーキテクチャにより実現可能となった。同時に、産業分野における情報処理も、マンマシンインタフェースなど非リアルタイム処理の割合が急激に増加して情報制御システムと呼ばれる形態へと変革している [2]。全体システムの構築容易性や保守性までを考慮した、新しい情報制御アーキテクチャが期待されている。

そこで、冗長構成系を分散システムと考え、タスクのリアルタイム性を保証する動的分散スケジューリングをフォールトトレランスにまで拡張する、リアルタイム分散システムからのアプローチが最近注目されている [3, 4]。筆者が提案する協調分散システムアーキテクチャも、フォールトトレラント性と柔軟な状況適応性を持つリアルタイム分散システムである [5, 6]。特定の待機冗長系や全体を制御するノードを持たず、個々の分散制御要素はその局所的な近傍要素とのみ負荷分散に基づくタスク分散を周期的に行なう。この局所アルゴリズムは、全体システムの漸近的挙動である協調ダイナミクスを安定にする。従って、制御要素の故障や部分的な負荷変動が発生すると、その制御要素の近傍系を中心にタスク再配置が拡散的に進み、最終的に全体システムとしての平衡状態に収束させる。

この協調分散アーキテクチャに基づくフォールトトレラント構成が協調タスクバックアップ方式であり、複数タスクをメモリ上にスタンバイ状態にした制御要素が相互に処理を引き継ぐ。処理引き継ぎに必要なトラッキングデータなどは指定されたチェックポイント (CP) で、他装置へと転送される。タスク切替えは各装置の再配置スケジューラの協調動作により決定され、各装置においてリアルタイム実行制御される [9]。このようなチェックポイントデータ転送方式では、CP データ転送や切替え時のタスク再実行のための処理余裕が必要であり、これを考慮した上で各タスクのリアルタイム性を保証しなければならない。

本論文は、協調タスクバックアップ方式について、タスクのリアルタイム性を保証するために必要な処理余裕と最適なチェックポイント設定について定量的解析とその評価結果について報告する。

2 協調タスクバックアップ方式

従来、産業分野における情報制御計算機システムでは、故障時の切替え速度を得るために、常用系に対して専用のバックアップ系を用意したフォールトトレラント構成をとっていた。例えば、二重 / 多重冗長系や n 台の装置に 1 台の待機系がある n 対 1 バックアップ方式などである。協調タスクバックアップ方式はシステムにおいて専用の待機系を持たず、故障装置の処理していたタスクをその隣接装置間の協調的なタスク分散配置 (Remapping Scheduler :RS) により、装置全体の処理余裕分でバックアップする。二相拡散アルゴリズムによる RS は周期的な Remapping Point (RP) 毎に起動される [7]。

情報制御システムで処理されるタスクは PID 制御などの周期的タスク、または、シーケンス制御などのイ

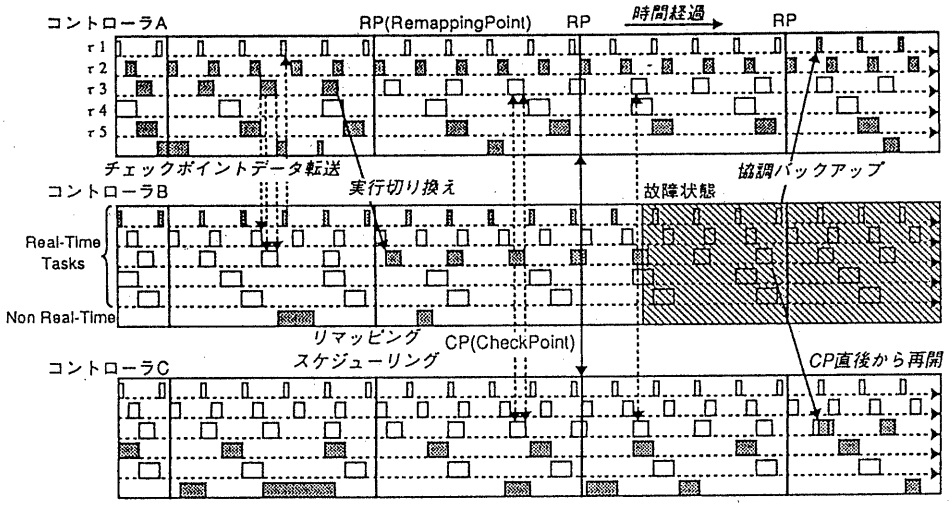

図 1: 協調タスクバックアップの動作例

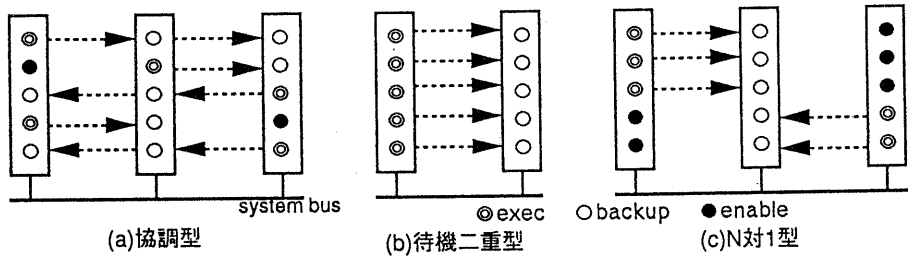

図 2: 協調タスクバックアップ方式の種々の構成例

ベントタスクが主であり、前者は次のプラントデータサンプリングまで、後者は次のイベント受け付け時間までのデッドラインを有する。これらのリアルタイムタスクはメモリに常駐し、タイマおよびイベントを要因に各プロセッサの Local Scheduler(LS) により実行制御される。協調タスクバックアップ方式ではバックアップタスクもメモリ常駐しておき、RS に割り当てられたタスクのみを LS が実行する。どのタスクが近傍系のどの装置でバックアップ可能で、どのタスクが RS により割り当てられたかは、タスク配置管理テーブル (TACT) により管理される。一方、制御に直接関係しない非リアルタイムタスクは LS によりリアルタイム処理の空き時間に処理される。

情報制御システムのリアルタイムタスクが周期的な処理を行なうためには、制御対象や外部からのサンプリングデータやイベント情報と、これまでの処理の結果保存されている積分データやトラッキングデータや内部状態フラグなどのタスク内部データを必要とする。タスク処理を他装置で引き継ぐ場合、前者は制御対象から採集できるが、後者は、実行側で保持する内容の複製をバックアップ側でも保持しなければならない。このタスク引き継ぎデータの常用 / 待機系の整合性維持方式にはハードウェアの構成方法を含めて、周期転送法や I/O 同期方式など様々な方式が提案されている [13, 14]。筆者のアーキテクチャは分散制御装置の接続方式をそれぞれの自立性を高めるために密結合でなくシステムバス接続としたので、タスク中に指定された checkpoint (CP) に引き継ぎデータをバックアップ可

能な装置に転送するCP転送方式を採用した。分散制御装置が相互に補完する、協調タスクバックアップの動作例を図1に示す。

協調タスクバックアップ方式は、RPとRSアルゴリズムの選び方によって様々なバックアップ方式と対応がとれる。一般的な協調型の場合、TACTは相互にバックアップ可能なタスクをホットスタンバイとする図2(a)のように設定される。TACTを図2(b)のように設定して、RSアルゴリズムを奇数偶数番の装置間での全タスク切替とすると、このバックアップ方式はホットスタンバイの待機二重系と同一となる。また、図2(c)では、右端の装置が全タスクのバックアップを引き受け待機しているn対1バックアップ方式と同じ構成となる。システムに要求される仕様に応じて、柔軟な構成が可能であることも同方式の柔軟性を示している。

3 性能解析モデルと評価

3.1 近似解法

チェックポイントに基づくタスクバックアップ方式は、CPデータ転送とリカバリのためのタスク再実行のためのオーバーヘッドを必要とする。一般に、CP間隔を大きくすると再実行のための時間が大きくなり、CP間隔を小さくするとデータ転送にかかる時間が大きくなる。従って、CPデータ転送と再実行のための合計時間が最小となるようなCP間隔が存在する[10, 11]。

Youngによる適正なCP間隔の近似解は次式で与えられる[12]。CP間隔をT_c、CPデータ転送時間をT_sとし、故障はランダム生起を仮定しその平均故障間隔をT_fとすると、同装置において再実行とCPデータ転送の合計時間T_lは最終的に次式で表される。

$$T_l = T_f + T_c/[1 - exp((T_c + T_s)/T_f)]$$

同式において、T_lを最小とするT_cは、$T_s \ll T_f$の

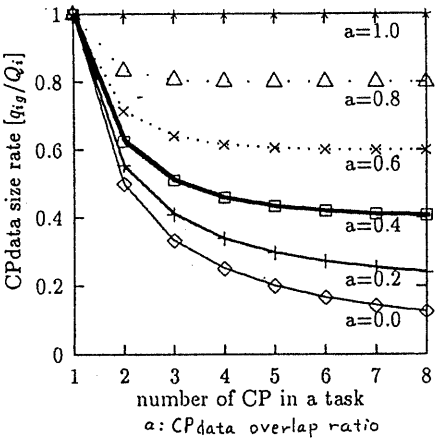

図3: CP数と各CPのデータ転送数

とき、

$$T_c = \sqrt{2T_c \cdot T_f}$$

と近似的に得られる。

より厳密な議論のためには、処理装置の能力、通信容量、タスクの特徴やCPデータの大きさを考慮して解析モデルを組むことが必要である。ここでは、情報制御システムの周期的なリアルタイムタスクを前提として、適切なCP間隔を導くための解析モデルをつくり、その評価を行なう。

3.2 CPデータ数

周期的なリアルタイムタスクは、次の起動時にも参照するトラッキングデータを、タスク空間内部あるいは複数のタスクが共用する空間に保持する。従って、これらをCPデータとして、各周期において最低一回は転送する必要がある。タスク中に複数のCPを設定すると、あるCPにおいて直前のCP以降で値が更新されたものは全て待機側に転送しなくてはならない。すなわち、異なるCPにおいて同一のトラッキングデータを転送することになるから、CPを多くするほど転送データ量は増加する。

あるリアルタイムタスクτ_iが次の起動において引き

継がなければならないデータ点数を Q_i とする。いま、異なるふたつの CP の間で送るべき CP データの重複する割合をいずれも a で一定であるとする。$a = 0$ のとき、CP で転送するデータに重なりはなく、$a = 1$ だと全ての CP で全 CP データ Q_i を転送することを意味する。タスク中に CP を g 箇所設定した時の平均 CP データ数を、q_{ig} とする。このとき、g と q_{ig} の関係は次式となる。

$$\begin{aligned} Q_i &= q_{i1} \\ &= q_{i2}\{2(1-a)+a\} \\ &= q_{i3}\{3(1-a)^2+3a(1-a)+a\} \\ &\cdots \\ &= q_{ig}\Sigma_{k=1}^{g}\, gCk\, a^{k-1}(1-a)^{n-k} \end{aligned}$$

ここで、$\Sigma gCk\ a^{k-1}(1-a)^{n-k} = 1/O_{(g,a)}$ とおくと、$O_{(g,a)}$ は全体 CP データに対する平均 CP データ量の比率を表す。

$$q_{ig}/Q_i = O_{(g,a)}$$

このとき全転送 CP データ量は、次式となる。

$$g \cdot q_{ig} = gQ_iO_{(g,a)}$$

タスク周期あたりの CP 数と平均 CP データ比率の関係を図 3 に示す。$\alpha = 1$ のとき、$O_{(g,\alpha)} = 1$ であり、$\alpha = 0$ のとき、$O_{(g,\alpha)} = 1/g$ となる。同じ引き継ぎデータがタスク途中で何度も更新されないようにタスクのプログラムを作成することが、CP 設定において効率的となることを図は示している。

協調バックアップ方式における CP 転送データは実行以前に CP 毎に明示的にアドレス指定する必要がある。すなわち、処理引き継ぎに必要な CP データは、静的なアドレスを持ったデータであり、一時的に存在する変数（例えばスタック変数）であってはならない。なぜなら、一時変数をバックアップ側のタスクに転送しても、タスクが実行されていない状態ではその変数

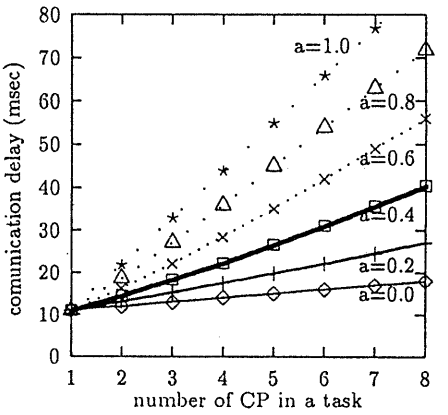

図 4: CP 数と転送時間合計 (CP 重複率の影響)

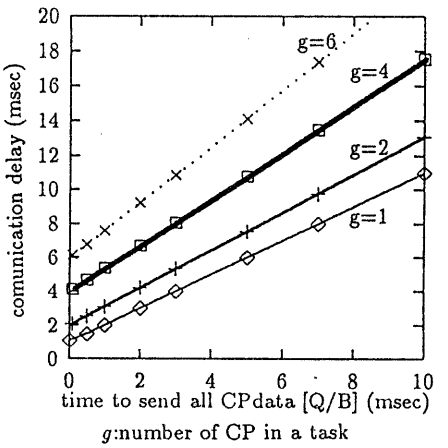

図 5: CP 数と転送時間合計 (通信量の影響)

が存在していないため、参照も設定もできないからである。

3.3 CP 転送遅延

次に、CP データを転送するのに必要な通信時間について考察する。$q$byte のデータを m 個の制御要素に対して転送するのに必要な通信時間 W を次式で定義する。ここで、通信バンド幅を B byte/sec、通信プロトコルのオーバーヘッドを P sec とする。

$$W = m(q/B + P)$$

すると、g 個の CP を持つリアルタイムタスク τ_i の CP 転送時間合計は次式で表される。

$$W_{ig} = mg(q_{ig}/B + P)$$
$$= mgQ_iO_{(g,a)}/B + mgP$$

CP 数と転送時間の関係を図 4 に示す。ここで、制御装置を接続するシステムバスは Ethernet 程度のものを仮定し、通信容量を 1MByte/sec、プロトコル遅延を 1 msec、タスクの全 CP データを 10 Kbyte とする。また、近傍系においてバックアップ可能な装置を 2 台としている。図 3 で示したように、CP 重複率が高くなると CP 分割数に対する通信量の増加率が高くなるので、通信量と通信容量の比で与えられるより急勾配となっている。通信時間を決定づけるのは通信量と容量の比である。図 5 は全 CP データ数 Q_i をパラメータとして、CP 数との関係を示したものである。ここで、CP 重複率 $a = 0.2$ としている。

3.4 再実行余裕

実行中のリアルタイムタスクが RS により装置切替が指示されたとすると、新たに処理を受け持った処理装置は受信している最新の CP から処理を再開する。従って、その直前の CP から RP までのタスク処理は両装置において二度実行される。リアルタイムシステムでは、この再実行部分を考慮した上でタスクのデッドラインを保証しなければならない。

いま、周期リアルタイムタスク τ_i に CP を周期あたり g 個設定したとすると、各 CP 間の部分タスクについて以下の関係式が成立する。

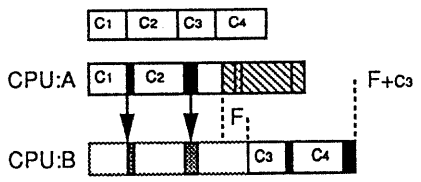

図 6: preemptiveタスク切替え

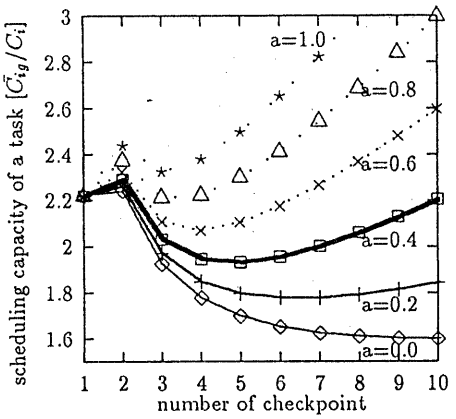

図 7: CP 数と処理余裕の関係
(データ重複率による比較)

$$C_i = \Sigma_{j=0}^{g} c_{ij}, \quad (c_{i0} = 0)$$

ただし、C_i は τ_i の周期あたりの処理時間であり、c_{ij} は (j-1,j) の CP 間の部分タスクの処理時間である。ここで、CP(0) はタスクの起動時、CP(g) はタスク終了時を表す。

はじめに、待機冗長系でのタスク再実行のための処理余裕を考える。図 6 のように装置 A で τ_i の CP (h) 直前で故障が発生したとすると、そのバックアップは故障検出および切替え処理に必要な時間 F の後、最新の CP (h-1) から処理が再開される。CP 転送中に故障が発生した場合、再送できないのでやはり CP (h-1) から処理を再開する。このとき、タスクのデッドラインを守るために装置 B に必要とされる処理余裕は、$F + c_{ih}$ である。従って、リアルタイムスケジューリングに際して τ_i の実効最大処理時間 $\bar{C}_{ig}$ を次式とすればよい。

$$\bar{C}_{ig} = C_i + \max_{(i \le j \le g)} c_{ij}$$

図 7 にタスク中の CP 設定数とスケジューリングに必要な処理余裕を示す。縦軸は、$\bar{C}_{ig}/C_i$ を表してい

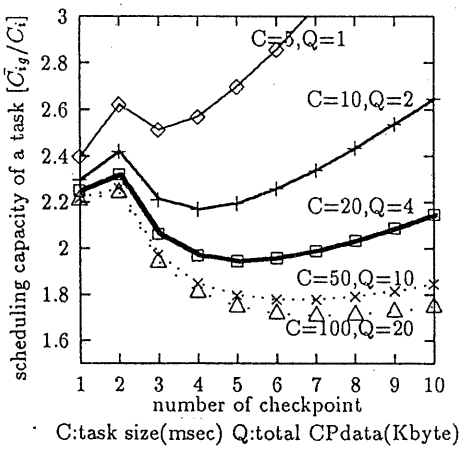

図 8: CP 数と処理余裕の関係
(タスク処理時間による比較)

る。ここで、全 CP データ数を 10 Kbyte、通信容量を 1M byte/sec、通信プロトコルオーバーヘッドを 1 msec、故障検出時間を 1 msec、タスク実行時間を 50 msec としている。また、最長の CP 間隔は平均 CP 間隔の係数倍を上回らないと仮定し、ここでは $k = 1.5$ とした。

$$\max_{(i \le j \le g)} c_{ij} \le kC_{ig}/g \quad (1 \le k \le g)$$

CP 数が小さいときは再実行のための処理余裕を大きくする必要があるが、CP 数が多くなると通信時間のための処理余裕が大きくなる。従って、両者の合計が最小となる CP 数が存在し、リアルタイムタスクにおける最適な CP 間隔の基準となる。図中では極小点として現れている。また、CP データ重複率が大きくなると、CP 数増加に従い通信量が増加するため、後者の影響が強く現れる。そのため、最適な CP 数は小さくなる。

次に、タスク処理時間をパラメータとして処理余裕を議論する。一般に、タスク処理時間が長くなると扱う CP データ数も多くなる。ここでは、その比 (Q_i/C_i) を一定 (0.2 Kbyte /msec) とした。CP データ重複率は 0.2 とし、その他の変数は図 7 と同じである。その計算結果を図 8 に示す。タスク処理時間に対する転送データ量の比が同じでも、プロトコルオーバーヘッドの負担がタスグ処理時間が大きくなると比率的に軽くなる。そのため、タスク処理時間の大きい方が最適な CP 数は増加するが、最適な CP 間隔としては処理時間の短いものより長くなっている。

4 リアルタイム保証条件

これまでの議論から、協調分散方式においてタスクセットのリアルタイム性を保証するための必要十分条件を導出する。

協調バックアップを行なうひとつの制御装置が一度に受け持つ全リアルタイムタスクセットを $(\tau_1, \cdots, \tau_n)$ とする。通常はこれらのうちいくつかのタスクは待機中で、それを実行する装置から CP データを受信しメモリ上の CP データを更新する。いま、制御装置単体でのタスクスケジューリング (LS) を Rate Monotonic 法 [15] とすると、同装置でタスクセットがリアルタイムスケジューリングできる必要十分条件は Lehoczky により与えられている [16]。

協調バックアップ方式では、タスク再配置は RP 周期毎の RS により実行される。この動作は再構成を行なう近傍系で同期的に行なわれなければならないので、RS は他のタスクに対して高優先度で実行される。RS を高優先度とした協調分散制御要素のリアルタイム保証条件は次式となる [8]。

$$\begin{aligned} L_{Pi} &= \min_{(t \in S_i)} (C_r \lceil t/T_r \rceil / t + \Sigma_{j=1}^{i} C_j \lceil t/T_j \rceil / t) \\ L_P &= \max_{(1 \le i \le n)} L_{Pi} \le 1 \\ & \quad L_{Pr} \le 1 \end{aligned}$$

$$t \in S_{ir} = \{k \cdot T_j | j = 1, \cdots, i, r; k = 1, \cdots, \lfloor T_i/T_j \rfloor\}$$

ここで、T_r, C_r は RS の周期および実行時間である。

協調タスクバックアップ方式で各タスクが RP において他装置より実行切替されて、最新 CP より再実行してなお元装置でのデッドラインを保証するには、各

タスクは全節で求めた実行処理余裕をもってリアルタイムスケジューリング保証されなければならない。タスクのバックアップ動作、すなわち CP 受信とデータ更新処理は、実行側の処理時間より短いと仮定すると、タスクのリアルタイム実行が保証されるならばバックアップ動作もリアルタイム保証できる。従って、次式が協調タスクバックアップ方式において、タスクセット $\tau_1, \cdots, \tau_n$ がリアルタイムスケジューリング可能な条件となる。

$$
\begin{aligned}
L_{Pi} &= \min_{(t \in S_i)} (C_r \lceil t/T_r \rceil / t + \Sigma_{j=1}^{i} \bar{C}_{jg} \lceil t/T_j \rceil / t) \\
L_P &= \max_{(1 \le i \le n)} L_{Pi} \le 1 \\
& \quad L_{Pr} \le 1
\end{aligned}
$$

故障が発生した場合は、次の RP までタスクのバックアップ動作が行なわれない。すなわち、故障時刻を t_f として、故障回復時間は次式となる。

$$F = T_r \lceil t_f / T_r \rceil - t_f$$

F はタスク処理余裕 $\bar{C}_{ig}$ に加算的に影響を与えるので、RP 周期を小さくして F を小さくしなければ、実行処理余裕が大きくなりスケジューリング条件が厳しくなる。

5 結論

分散制御要素が相互にバックアップする協調タスクバックアップ方式について、適用に際して必要な処理余裕について定量的解析を行ない、Rate-Monotonic 法に基づくタスクのリアルタイム保証条件について導出した。今回のシミュレーションでも示されたように、CP 転送方式を用いるシステムでは、総合的な通信速度が性能を支配する。また、RS/LS のスケジューリング方策によりシステム / タスクの応答性が決定付けられる。この二点について手法の検討を進めるとともに、分散リアルタイムシステムの評価解析ツールの整備も行なっていく予定である。

参考文献

[1] 森下巖 編: ディジタル計装制御システム, 計測自動制御学会, 63/76 (1983)

[2] 片岡、香川、古屋: 統合制御システム MELTAS, 三菱電機技報, 63,5, 347/354 (1989)

[3] S.Cheng et.al. : Scheduling Algorithms for Hard Real-Time Systems, Hard Real-Time Systems, IEEE CS Press, 150/173 (1988)

[4] J.A.Stankovic : Distributed Real-Time Computing: The Next Generation, 計測自動制御学会誌, 31, 7, (1992)

[5] 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)

[6] H.Kanamaru, M.Takegaki : The Coordinative Decentralized Control SYstem Based on Local Communications, Proc. of IECON'90 16th Annual Conf. of IEEE IES, 533/538 (1990)

[7] M.Takegaki, H.Kanamaru and M.Fujita : The Diffusion Model Based Task Remapping for Distributed Real-Time Systems, Proc. of 12th IEEE Real-Time Systems Symp, 2/11 (1991)

[8] 神余, 竹垣, 藤田 : 協調分散スケジューリング方式のリアルタイム保証性, 情報処理学会研究会 RTP '92, 92-ARC-93, 99/107 (1992) 92-DPS-53, 69/76 (1992)

[9] 神余, 竹垣 : 協調分散型制御システムのフォールトトレラント方式, 情報処理学会研究会 SWoPP'91, ARC89-8, 57/64 (1991)

[10] 南谷 : フォールトトレラントコンピュータ, オーム社, 143/165 (1991)

[11] K.M.Chandy and C.V.Ramamoorthy : Rollback and Recovery Strategies for Computer Programs, IEEE Trans. Comput., C-21, 6, pp546/556 (1972)

[12] J.W.Young : A First Order Approximation to the optimum checkopint interval, Commun. ACM, 17, 9, 530/531 (1974)

[13] 向殿政男 : コンピュータシステムの信頼性評価, 電子情報通信学会誌, 73, 11, 1142/1148 (1990)

[14] 真屋 他 : 分散システムにおける高速回復方式の提案, 電子情報通信学会論文誌, Vol.J74-D-I, No.10, 729/738 (1991)

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

[16] J.Lehoczky, L.Sha and Y.Ding : The Rate Monotonic Scheduling Algorithm : Exact Characterization and Average Case Behavior, Proc.of IEEE Real-Time Systems Symposium, 166/171 (1989)