計算機アーキテクチャ　93－14
(1992. 3. 23)

協調分散スケジューリング方式のリアルタイム保証性

神余浩夫　　藤田昌也　　竹垣盛一
三菱電機中央研究所
(661) 兵庫県尼崎市塚口本町 8-1-1

協調分散制御システムは、システム故障や負荷変動に対して、定常的リアルタイムタスクの実行を維持するように制御プロセッサが協調的にタスク再配置を行なう制御システムである。協調機構は、サブシステムにおけるタスク分散を行なう Remapping Scheduler と、プロセッサ内部のリアルタイム実行を保証する Local Scheduler により構成され、アプリケーション周期タスクは checkpoint 毎に他プロセッサと同期をとりながら、両スケジューラにより実行制御される。本論文は両スケジューラによる周期タスクのリアルタイム保証性について解析的評価を行ない、システム設計の基準を明確にする。

Schedulability of The Coordinative Decentralized Scheduling Policy

Hiroo Kanamaru　　Masaya Fujita　　Morikazu Takegaki
Central Research Lab. Mitsubishi Electric Corp.
8-1-1 Tsukaguchi-Honmachi Amagasaki, Hyogo, Japan

The Coordinative Decentralized Control System is that the reconfigurable system to keep its steady periodic tasks by the interaction of the controllers. Its coordination mechanism is a couple of Remapping Scheduler; which reallocates tasks in a subsyetem, and Local Scheduler; which is a realtime scheduler in each node. These schedulers make swap the running periodic tasks which keep consistencies with itselves on other nodes. We analyzed the schedulability of the coordination mechanism for realtime tasks and show the guideline of the system design.

1 諸言

発電プラント、化学プラント制御システムや生産ライン、ロボット制御装置などの産業分野における情報制御システムは、制御処理が監視対象の挙動や動特性に追従するように、タスク処理および応答が制御周期やタイムアウト時限などのデッドラインに間に合わなければならない、強リアルタイムシステムである。また、このような産業プラントが停止あるいは制御不能になると多大の損失を被るので、フォールトトレラント性も強く要求される。このような要求に対して、システム状況に応じてタスクを各制御要素に動的にスケジューリングを行なう、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を集めている [1]。

筆者らが提案している協調分散制御システムは、システムの各制御要素の処理余裕や故障状態あるいはタスクの負荷状況に応じて、動的にタスクを再配置することにより、情報制御システムの定常的機能とその応答性を維持するシステムである [2]。タスク再配置は各制御要素が周期的にその近傍系について行ない、この各近傍系の再配置結果が次第に全体へと波及することによって、全体システムは負荷分散的な平衡状態を形成する。すなわち、制御要素の相互作用による動的タスクスケジューリングがつくる全体システムの挙動(協調ダイナミクス)が安定となるように設計された分散システムである [3, 4]。

協調分散制御システムの各制御プロセッサは、一定周期のリマッピングポイント(RP)においてその周期の実行担当タスクをリマッピングスケジューラ(RS)から得て、各制御プロセッサのローカルスケジューラ(LS)によってタスク周期を満足するように実行制御される。タスク実行切替えに必要なタスク内部情報は各タスク中のチェックポイント(CP)毎に担当制御プロセッサから他プロセッサへと転送され、分散システム上の一貫性が保証される。

本論文では、協調分散制御システムがタスクのリアルタイム性を保証するための各種制約を議論する。まず、拡散モデルに基づく協調分散制御システムの概念とその協調機構について述べる。そして、リマッピングスケジューラの実装方法によるリアルタイムタスクのスケジューラビリティについて解析を行なう。次に、CP の設定間隔を通信量と実行切替え時の処理余裕を基準に評価を行なう。以上の結果から、協調機構の実装方式と制約条件について考察を行なう。

2 協調分散制御システムの概要

2.1 協調分散の目的

代表的な情報制御システムの運用例を図1に示す。分散制御要素群はプラントや生産ラインの機器とリモートI/O装置を介して周期的なサンプリングを行ない、制御周期などのデッドラインまでに計算処理などで求めた制御指示を出す。また、一定時間毎の履歴情報を運転監視盤(OPS)や運転管理計算機などに転送する。時には、運転員からの指示命令により非周期的なプラント入出力やデータ転送が発生する。最近のシステムでは、運転支援のグラフィック処理や保守支援のデータ検索などの非リアルタイムタスクの処理割合が増える傾向にある。情報制御システムには、機器制御を行なう周期および非周期のリアルタイムタスクの実時間保証と非リアルタイムタスクのできるだけ早い応答性が要求される。また、制御プロセッサの故障時には故障プロセッサの機能を他制御要素で速やかにバックアップしなければならない。

従来、フォールトトレラントを指向するとH/Wレベルで多重化構成とすることが多かったが [5]、非リアルタイムタスクを含む全ての処理を多重化してバンプレスに切替える必要はない。一部のクリティカルなタスクを除いて、所定の切替え時間以内に処理が継続的に再開できれば制御システムとしての性能に問題はない。そこで、協調分散制御システムはシステムの定常的な機能保証、すなわち周期タスクのリアルタイム性を保証するように、所定の切替え時間の周期で分散制御プロセッサ間のタスク割当を配置し直す。この動的再構成によって、システムの制御要素の故障やタスク負荷の増大といったシステム状況の変化に対して、柔軟な適応性と高い可用性を得ることができる。

2.2 協調分散の機構

一般的な分散リアルタイムシステムの動的スケジューリング問題は NP 困難であるので、現在提案されている分散リアルタイムスケジューリング方式は、各ノードにおけるタスクのローカルスケジューラと、ノード間の主に負荷分散を目的とした分散スケジューリングから構成されることが多い。

Ramamrithan らは、タスクのリアルタイム制約保証率をできるだけ高くするため、焦点配置法と入札法を組み合わせた手法を提案している [6, 7]。焦点配置法はノード

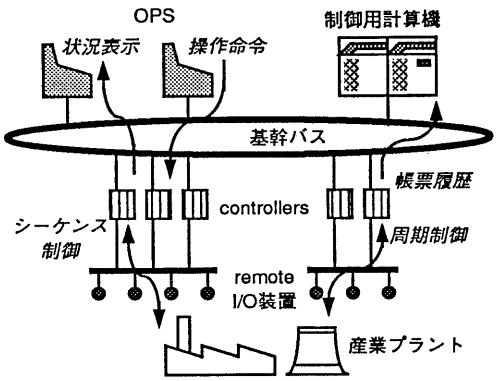

図 1: 情報制御システムの例

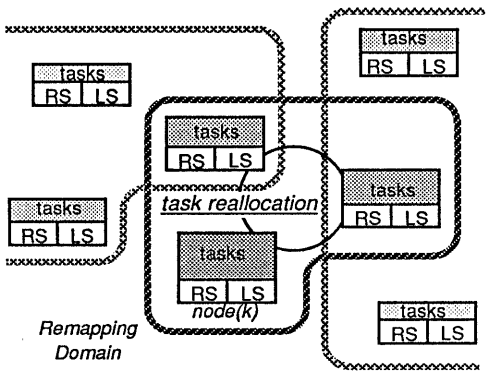

図 2: 協調分散システムの近傍系

で保証できないタスクがあれば、処理時間余裕があると予想されるノードを選んでタスクを送る。入札法はタスクの送り先を入札によって決定する。これらの手法をヒューリスティックに組み合わせて、効果的な分散タスクスケジューリングを行なう。Cheng らは、いくつかのタスクが同じデッドラインを持つ時、そのタスクグループをネットワーク上に分散し、焦点配置法と入札法を組み合わせてグループの実行制御を行なう手法を提案している [8]。

分散スケジューリングの各方式は、複雑性を回避するためのタスクとノードのサブシステム分割と、その中のタスク配置方式に特徴付けられる。筆者らの協調分散システムの拡散協調モデルでは、図 2 に示すように各ノード近傍系を一回当たりのタスク再配置範囲である *Remapping Domain(RD)* なるサブシステムとする。ここで、タスク再配置を行なうスケジューラが *Remapping Scheduler(RS)* であり、各プロセッサにおけるリアルタイムタスクの実行を制御するのが *Local Scheduler(LS)* である。隣接するノードの RD は互いにオーバーラップしているので、RS によるタスク再配置による部分的な変化はオーバーラップ部分を介して次第に周辺ノードへと拡散的に波及し、ついには全体が負荷分散的な平衡状態に達する。このような部分的再構成の繰り返しにより全体システムを協調的に安定とする協調分散システムの動作モデルを拡散協調モデルと呼んでいる [4]。分散システムが拡散協調モデルに従い協調的動作を行なうためには、RS に基づく全体システムの漸近的挙動が安定であることが条件となる [9]。筆者らは各制御ノードと隣接する両側のプロセッサを RD として負荷分散タスク分配を行なう二相拡散アルゴリズム [4] を RS としたときについて、そのタスク分配係数の安定条件を示している [10]。

RS により求められたタスク再配置結果は、RD について一定周期の *Remapping Point(RP)* において各制御プロセッサの LS に反映される。具体的には、その RP 周期のタスク割り当て状態が、LS が参照するタスク実行管理ブロック (TCB) に追加 / 削除される。周期的アプリケーションタスクは通常メモリに常駐し LS によって実行制御されるので、そのタスクのバックアップを行なう制御プロセッサでもタスクを常駐させておき、RP において実行プロセッサ切替えが生じた時には直ちに対応できるようにする。すなわち、そのタスクに関してバックアップ可能プロセッサにおいて cold stand-by 状態で待機している。この実行系と待機系のタスクの内部状態、処理の継続的切替えに必要な引き継ぎデータはアプリケーション内部に設定した *Cehck Point(CP)* 毎に実行系から待機系へと転送、複写される。従って、両者は CP のタイミングで一貫性が保たれる [11]。CP のタイミングと転送する CP データはアプリケーションタスク設計時に CP 毎に選定抽出して、一覧表として記述しておく必要がある。

協調分散制御システムの周期タスクはある制御プロセッサで実行、他のバックアップ可能なプロセッサでは待機状態となっているが、待機側は全てのタスクを待機状態としているのではなく、負荷分散的にタスクを担当実行している点が従来の待機冗長系とは異なる点である。しかし、RD をオーバーラップしないふたつの制御プロセッサとして、RS を負荷分散でなく片方に全てのタスクを集中実行するような方式を採用すると、この RD の制御プロセッサ構成

はcold stand-byの待機二重冗長構成のシステムとなる。このように、協調分散方式はRS,LSの方式によって幅広いシステム構成が可能であり、柔軟な適用性を有している。

3 協調分散方式のスケジューラビリティ

3.1 RSの実行タイミング

協調分散方式では特定の制御プロセッサだけがRSを実行するのではなく、全てのプロセッサがRSを相互通信により協調的に実行する。各プロセッサのRSはRPタイミングでLSに次のRP周期のタスク割当を与えなくてはならないので、RP間隔をデッドラインとする周期的リアルタイムタスクとして扱わなくてはならない。従って、RSはLSにより実行制御される。ここで、RSは各プロセッサにおいて次のふたつの実現方式がある。

- preemptive: RSタスクはより優先度の高いアプリケーションタスクによりpreemptされる。次のRPまでに終了すればよい。
- non-preemptive: RSは優先度の最も高いタスクとして、RPにおいて他タスクをブロックして実行される。結果は即座にLSに反映される。

この各々の場合について、各プロセッサにおけるRSおよび周期的アプリケーションタスクのスケジューリング条件を導出する。

3.2 preemptiveの場合

協調分散システムのあるRDにおいて、アプリケーション周期タスク$\tau_1,\cdots,\tau_n$が存在し、RDに属する全てのプロセッサが全てのタスクを実行可能であり、かつ、Rate Monotonic法[14]を採用したLSによりスケジューリング可能であるとする。プロセッサPにおいて、タスクセットがRM法によりスケジューリングできる必要十分条件は、

$$\begin{aligned} L_{Pi} &= \min_{(0\le t\le T_i)} \Sigma_{j=1}^{i} C_j\lceil t/T_j\rceil/t \\ L_P &= \max_{(1\le i\le n)} L_i \end{aligned}$$

としたとき、$L_P \le 1$である[15]。ただし、T_i, C_iはそれぞれτ_iの周期および計算時間である。ここで、tは次式で与えられる。

$$t \in S_i = \{k\ldots T_j | j=1,\cdots,i; = 1,\cdots,\lfloor T_i/T_j\rfloor\}$$

上の条件が成立して周期タスク$\tau_1,\cdots,\tau_n$がスケジューリング可能であるとき、さらにRSタスクτ_rを与えた時のスケジューリング条件について考える。τ_rは全ての周期タスクの中で最も長い周期を持つから、Leoczkyの定理は以下のようになる。

$$\begin{aligned} L_{Pr} &= \min_{(t\in S_r)} (\Sigma_{j=1}^{n} C_j\lceil t/T_j\rceil/t + C_r\lceil t/T_r\rceil/t) \\ &= \min_{(t\in S_r)} (\Sigma_{j=1}^{n} C_j\lceil t/T_j\rceil/t + C_r/t) \\ &= \min_{(t\in S_r)} (L_{n,r}(t) + C_r/t) \end{aligned}$$

$$L_{ki} \le 1, (1 \le i \le n)$$

$L_{Pr} \le 1$ならば、RSタスクを含めた全ての周期タスクがスケジューリング可能である。すなわち、図3に示すように、$\tau_1,\cdots,\tau_n$をスケジューリングしておいて、T_rの区間にC_rを終了させる余裕があることを条件とする。

いま、プロセッサPで全ての周期タスクセット、

$$\Upsilon = \{\tau_1,\cdots,\tau_n\}$$

を担当したが、協調分散方式ではこれらのタスクセットをRD内の他プロセッサと分担して協調的に実行する。従って、ひとつのプロセッサでn個全てのタスクをリアルタイムスケジューリングできなくてもよい。すなわち、Υの部分集合であるタスクセットΥ_Pについて上式が成り立つ時、プロセッサkは、Υ_PとRSタスクτ_rをリアルタイムスケジューリングできる。

$$\Upsilon_P = \{\tau_{1'},\cdots,\tau_{n'}\} \subseteq \Upsilon = \{\tau_1,\cdots,\tau_n\}$$

RD内のプロセッサ間の負荷分散が最適となるような、あるいはプロセッサ利用率が均一となるようなタスクセットΥ_Pを求めようとしても、組合せ的困難である。二相拡散アルゴリズムによるタスク分配方式は、RD内の負荷偏差が小さくなる方向にタスクを順次移動させる程度であり、問題の複雑化を回避している。なぜなら、協調分散の目的が定常的なリアルタイムタスクの実行を保証する程度に動的な負荷分散を行なうことであり、瞬時の全体最適負荷分散を目的としていないからである。

3.3 non-preemptiveの場合

RSタスクが最高優先度で他タスクによりpreemptされないとき、RPにおいて各プロセッサはRSタスク実行

時間 C_r の間ブロックされる。従って、図4に示すようにアプリケーション周期タスクの実行割り当て時間は、

$$C_r \leq t \leq T_r$$

となる。このとき、τ_i の最初の周期 ($t = T_i$) での CPU 利用率は、

$$C_i/(T_i - C_r)$$

であり、$T_i \leq C_r$ では明らかに τ_i のリアルタイム保証はできない。このとき、τ_i の RM スケジューリング可能条件は、

$$L'_{Pi} = \min_{(t \in S_i)} (C_r/t + \Sigma_{j=1}^{i} C_j \lceil t/T_j \rceil / t)$$

より、$\tau_1, \cdots, \tau_n$ がすべてスケジューリング可能である必要十分条件は、

$$L_P = \max_{(1 \leq i \leq n)} L_{Pi} \leq 1$$

である。

最初の周期に残された時間 $T_i - C_r$ で自分自身とより短い周期のタスクを実行しなければならず、先ほどの preemptive な RS 実行方式に比べてより厳しい条件となっている。non-preemptive な RS タスクは RD 内の他プロセッサとの通信処理の中で緩い同期機構を有している [12]。現実の分散システムでは、分散プロセッサ間の同期手段のような比較的長周期であるが他タスクをブロックする一種のシステムタスクが必要であり、上のスケジューリング条件はこのような高優先度のシステムタスクに関しても一般的に適用することができる。

preemptive,non-preemptive いずれの場合でも、周期タスク数が多くなると上式の判定条件に基づき周期タスクのスケジューラビリティを解析するのは容易ではなくなってくる。現実的なシステムでは、割り込みや非リアルタイムタスクを考慮しなければならない。このような状況におけるリアルタイムスケジューラビリティを評価するために、筆者らはタスクスケジューリングアナライザ (TSA) を開発を進めている [16, 17]。TSA を用いて設計時にリアルタイムシステムの性能評価ができるようになり、システム構築が効率化できる。

4 CP 間隔とスケジューラビリティ

4.1 タスク切替え動作

協調分散制御システムは RP のタイミングでアプリケーション周期タスクの実行を他プロセッサに切替えることができる。RP において実行中であった、あるいは preempt されたタスクも切替え対象となる。実行中のタスク実行が切り替わった場合、その preempt された時から処理を再開できればよいが、メモリ状態やレジスタ値など全ての情報を RP 時にプロセッサ間転送することはオーバーヘッドが大きくなり過ぎる。そこで、タスク中のチェックポイント (CP) で処理引き継ぎに必要なタスク内部データであるチェックポイントデータ (CP データ) を他プロセッサに転送し、タスク実行プロセッサが切り替わっても、CP タイミングでの実行系、待機系の内部データの一致を保証する。

CP による常用・待機系の内部データの同期手法は待機冗長によるフォールトトレラント方式では広く使われてい

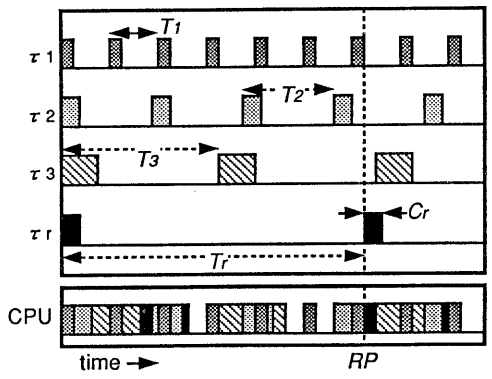

図 3: preemptive RS のスケジューリング

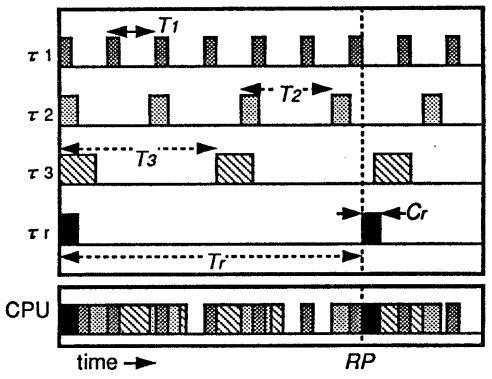

図 4: non-preemptive RS のスケジューリング

る方法である。待機冗長系におけるCPタイミングの代表的なものに以下が挙げられる。

- **両系メモリ書き込み法**: 常用系でCPデータをメモリ上に書き込む毎に待機系のメモリも更新し、両系のメモリ内容を常に一定に保つ。ハードウェアの工夫が必要。
- **周期転送法**: CPデータを一定周期で待機系に転送する。転送周期間にI/Oアクセスが行われるとき、待機系でその動作の再現が可能なように履歴(ジャーナル)を作成する必要がある。
- **I/O同期方式** [13]: ファイルの書き込みなどジャーナルが必要なI/Oアクセスをタイミングとして、待機系にCPデータを転送する。待機系は直前のI/Oアクセス終了時点から処理を再開できる。
- **CP転送命令方式** [11]: アプリケーションタスクにおいてCP転送命令によりCPデータを転送する。タスク終了時には命令が明示されてなくともCP転送を行う。

協調分散制御システムは疎に結合された分散システムであるので、両系メモリ書き込み方式は実現が難しい。また、タスクの入出力はプラントとのI/Oであり、ジャーナルを参照して再現できないものが多い。以上の理由から、協調分散制御システムにおけるCP転送方式は、I/O同期方式を包含したCP転送命令方式を用いる。タスク設計時に、出力が行われるなど再実行が無意味な時点でCPおよびCPデータを明示的に指定し、実行時のCP転送を制御する。タスク実行プロセッサが切り替わった場合は、直前のCPすなわち最新のCPデータを持っている時点から処理を開始する。

4.2 CP間隔と通信遅延

タスク設計者は各タスクについて、効率的通信と迅速な処理引き継ぎができるように、CPデータの転送量と処理再開のタイミングを考慮してCPを設定しなければならない。ここでは、CPの設定間隔と、それに伴う通信量および通信遅延について定量的に評価を行ない、CP設定の基準を求める。ここで、近傍系サブシステム内部の通信にブロードキャストを使用せず、サブシステム内のm個の制御要素に対してそれぞれメッセージ転送を行なうものとする。CP動作タイミングを図5に示す。

あるタスクに関するチェックポイントiにおいて転送するCPデータ量を$q_i$byteとし、このサブシステム内部の通信容量を一律Bbyte/sec、両ノードにおける通信プロトコルのオーバーヘッドをPsecとする。いま、近傍系の全ての制御要素に対してCP転送した時の通信時間は次式で与えられる。

$$L_i = m(q_i/B + P)$$

図5において、CP$(i-1)$とiの間隔をT_{Ci}とすると、iにおけるCP通信の時間占有率は次式で定義できる。

$$R_i = L_i/T_{Ci}$$

次に、このタスク中に設けたCP数を減らしてCP間隔を大きくすることを考える。設定したCP$(i = 1 \cdots n)$におけるCP転送を$i = n$でまとめて行なうとする。同じタスクに着目すれば、いくつかのCPで同一のCPデータを転送していることがあるので、まとめた時に転送すべきCPデータ量は減少する。各CPにおけるCPデータ転送量を等しいとし、任意のふたつのCPi, jの間でCPデータが重なる割合を係数αの一定とすると、

$$\alpha := const, (0 \leq \alpha \leq 1)$$

$$q = q_i, \quad i = 1, 2, \ldots, n$$

$\alpha = 0$のとき、CPデータに重なりはなく、1のときCPデータは一致していることを意味する。このとき、CP$i = n$における転送データ量は次式となる。

$$\bar{q}_n = q \sum_{i=1}^{n} nCi\alpha^{i-1}(1-\alpha)^{n-i}$$

αが0および1のとき、$\bar{q}_n$はそれぞれ、nq, qとなる。このとき、通信時間と占有率はそれぞれ次式となる。

$$L_n = m(\bar{q}_n/B + P) \leq m(nq/B + P)$$

$$R_n = m(\bar{q}_n/B + P)/nT_{Ci} \leq R_i$$

$$T_{Cn} = nT_{Ci}$$

すなわち、CPデータをまとめて転送した方が、平均的な通信効率を表す通信時間占有率が小さくなる。この効果

は通信プロトコルの処理が重く、CP データの重複率が高いほど期待できる。ただし、一回の CP における通信時間は大きくなるので、L_n が他のタスクのリアルタイム制約に干渉しないように設定しなければならない。また、CP 間隔を大きくすると、実行プロセッサ切替え時の再実行部分が大きくなり、LS のスケジューラビリティが悪化する。

4.3 CP 間隔と処理切替え

RP においてアプリケーション周期タスクが preempt されて、実行プロセッサ切替えが行なわれたとすると、切替え先のプロセッサではそのタスクに関して直前の CP、すなわち既に受信した最新の CP から処理を再開する。このとき、処理再開する CP から、実行切替え元のプロセッサでの preempt 時点までの処理が分散システムにおいて二度実行されることになるので、そのタスクの正味の計算時間がその周期においてのみ増加する。以下では、このような preempt されたタスク実行を切替えた場合のリアルタイム保証性について評価する。

いま、分散システムの各プロセッサは絶対時間に対して同期しており、時差がないものとする。ある RD をなすふたつのプロセッサ P, Q は、いずれも n 個のタスクからなるタスクセット Υ を、RP 周期の中で RM 法によりスケジューリングが保証されているとする。Υ に属するタスク τ_i に CP を l 個設定する。

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

$$c_i = \max_{(1 \le j \le l)} c_{ij}$$

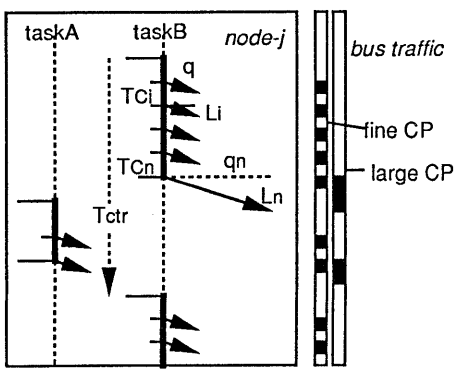

図 5: CP 間隔と通信量

ただし、CP(0),CP(l) はそれぞれ各周期におけるタスク開始点および終了点である。

いま、時刻 $t = kT_r$ の RP において、アプリケーション周期タスク τ_i がプロセッサ P から Q に RS により実行切替えが指示されたとすると、τ_i のその周期のデッドラインまでに残された時間は、

$$D_{ki} = \lceil kT_r/Ti \rceil - kT_r \ (k = 1, 2, \cdots)$$

となる。図 6 に示すように τ_i が P において、CP(h-1) と CP(h) の間で preempt されたとすると、少なくとも pre-empt 点以後を P は処理できることから、

$$C_i - \Sigma_{j=0}^{h+1} c_{ij} \le D_{ki} \ (h = 1, \cdots, l)$$

である。

τ_i が Q でなく、P において直前の CP から再開するための条件を考えると、二度実行される部分、c_h をその周期において処理する余裕を持つことが必要である。従って、τ_i のみかけの計算時間を次式で与える。

$$\hat{C}_i = C_i + c_i$$

このみかけの計算時間について τ_i がリアルタイムスケジューリング可能である時、どの preempt 時点でも D_{ki} までに c_i の処理余裕を持つことになる。従って、全てのタスクを preempt 状態で実行切替えを行なって、その時のデッドラインを満足するための十分条件は、

$$\hat{L_{Pi}} = \min_{(0 \le t \le T_i)} \Sigma_{j=1}^{i} \hat{C_j} \lceil t/T_j \rceil / t$$

$$\hat{L_P} = \max_{(1 \le i \le n)} \hat{L_i}$$

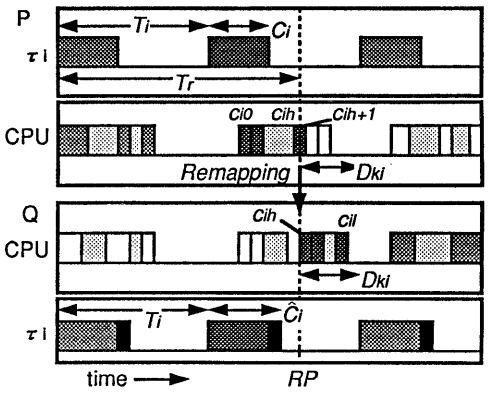

図 6: CP 間隔と処理切替え

$$t \in S_i = \{k \ldots T_j | j = 1, \cdots, i; = 1, \cdots, \lfloor T_i/T_j \rfloor\}$$

としたとき、$\hat{L_P} \leq 1$である。ただし、一回のRSで移動するタスクセット数が限定されるので(二相拡散の場合、最大担当タスク数の1/2)、移動するタスクセットの最悪状態について成立すればよい。

すなわち、CPデータの転送効率向上のためCPを大きくすると、実行切替え時のタスクのリアルタイム保証のための処理余裕を含んだみかけの計算時間$\hat{C_i}$が大きくなり、RM法によるスケジューラビリティが悪くなる。

5 結論

協調分散制御システムのリアルタイムタスクと協調機構の実現方式と、そのリアルタイム保証性について議論を行なった。拡散モデルに基づく協調分散システムでは、協調機構を各ノードの近傍系に関する周期的なRSと各ノード内部でのRate-Monotonic法に基づくLSにより実現され、RSの実現方式とそのときのアプリケーションタスクのスケジューラビリティについて必要十分条件を導出、比較した。また、タスク実行切替え時の実行再開点を示すタスクのCPについて、その設定間隔とLSに対する影響を解析的に評価した。その結果を定性的に述べると、プロセッサ処理余裕に対し通信遅延が大きい場合はCP間隔を大きく、通信容量に余裕がある場合はCPを小さくすることが望ましい。

今回の評価モデルは、分散システム間の完全な同期を前提とするほか、実際システムで解決しなければならない問題を前提条件としている。今後は、これらの課題についてモデルによる解析的方法と、TSAを用いたシミュレーション的方法によって解決を図る予定である。

参考文献

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

[2] 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)

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

[4] M.Takegaki, H.Kanamaru, M.Fujita : The Diffusion Model Based Task Remapping for Distributed Real-Time Systems, 12th IEEE Real-Time Systems Symp. (1991)

[5] 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)

[6] K.Ramamritham and J.Stankovic : Dynamic Task Scheduling in Distributed Hard Real-Time Systems, IEEE Software, 1-3, 65/75 (1984)

[7] J.Stankovic et.al : Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems, IEEE Trans. on Computers, C-34,12, 1130/1143 (1985)

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

[9] 深尾 : 分散システム論, 昭晃堂, 93/133 (1987)

[10] 神余, 竹垣 : 拡散モデルに基づく協調分散制御システム, 情報処理学会研究報告, 92-DPS-53, 69/76 (1992)

[11] 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)

[12] 神余, 竹垣 : 協調分散システムにおける同期機構, 第34回システム制御情報学会研究発表講演会論文集, 65/66 (1990)

[13] 真矢 他 : 分散システムにおける高速回復方式の提案, 電情通論文誌, Vol.J74-D-I, No.10, 729/738 (1991)

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

[15] 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)

[16] 藤田, 竹垣 : 制御計算機用タスクスケジューリングアナライザの試作, 第43回情報処理学会全国大会講演論文集, 6-359/360 (1991)

[17] 藤田, 竹垣 : 分散制御システムへの非周期タスクサーバーの適用性, 電子情報通信学会研究会 第1回実時間処理ワークショップ (1992)