計算機アーキテクチャ　93－13
（1992. 3. 23）

分散制御システムへの非周期タスクサーバの適用性

藤田昌也　　竹垣盛一
三菱電機中央研究所
兵庫県尼崎市塚口本町 8-1-1

周期、非周期タスクが混在するリアルタイムシステムにおいて、高いスケジューラビリティと高速応答性を確保する手法に、非周期タスクサーバアルゴリズムがある。制御用マンマシンシステムのような実システムに同手法を適用する時、システムクロック粒度による性能劣化が問題となる。そこで、システムクロックをパラメータとして非周期タスクサーバによる周期、非周期タスクのスケジューラビリティについてシミュレーションを行った。その結果、実システムのシステムクロック領域でもタスク応答性が悪化せず、十分適用可能であることが明らかになった。

Avalability of Aperiodic Task Server to Distributed Control System

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

For the real-time system which processes periodic tasks and aperiodic tasks, the aperiodic task server algorithm is able to guarantee their deadlines and keep fast responses of these tasks. Applying this algorithm to the operator's terminal of an industrial control system, it is a major issue that the long time clock of the real system makes its schedulability worse. We simulated the behaviors of tasks under the server implemented on various system clocks. According to the simulation, the differneces of system clock have no relation with the responses of aperiodic tasks, and it shows the server is available for the real system.

1 はじめに

現在の制御用実時間システムではEIC統合に伴うシステムの大規模化や、リスク分散、機能分散などを目的とした分散化が進んでいる。そのため、システムの構造が複雑となり、全体を把握することは非常に困難になってきている。

特に制御用システムではプラント制御などのために、時間的な制約を持ったタスクとそうでないタスクの処理が混在しており、システムの時間的ふるまいを把握することが重要であるが、システムの複雑化に伴って、それを設計段階で把握することは非常に困難なものとなっている。

このような状況下で、制御システムの設計者は、経験的手法、試行錯誤的方法や、シミュレーションによる検証などに頼って設計を行っており、設計から現地調整にいたるまでの経費的・時間的浪費が大きくなってきている。

これらの問題は、実時間スケジューリングの理論を設計のベースとすることによって解決され得るものであり、システムのより効率的な設計を可能とする。実時間スケジューリング理論の目的の一つは、システムの時間的ふるまいを予測し、実時間システムに与えることのできるタスク セットを決定するための指標を与えることにある。

しかし、これまでの実時間スケジューリング手法の議論においては、実際のシステムとは異なるいくつかの単純化が行われている。一つは、あるタスク τ_H がエントリーしてきた時、τ_H のプライオリティが実行可能状態にあるタスクの中でもっとも高いプライオリティであった場合には、現在実行中のタスクの処理が中断され、すぐに τ_H の処理が開始されるという仮定である。実際のシステムでのタスク スケジューリングはシステムの最少時間単位であるシステム クロック毎にしか行われない場合が多い。今回我々は、実時間スケジューリング理論の中で提案されている手法であるRate Monotonic スケジューリングについて、タスク スケジューリングがシステム クロック毎にしか行えない様な離散的なスケジューリングの場合のスケジューリング性能に現れる影響について検討を行った。特に、マンマシン システムの性能改善に有効であると思われる、Rate Monotonic スケジューリングをベースとした非周期タスク サーバについて、その効果がシステム クロックを考慮した場合にどのような影響を受けるかを検討したので報告する。

2 分散制御システムにおけるタスク管理の問題

現在の制御用コンピュータは、H/W の劇的な性能向上から高度な処理能力を持つようになり、マン マシン インタフェースを含めた電気制御、計装制御、計算機制御のシステム統合化が急速に進んでいる。それに伴い、各コントローラではミリ秒から秒オーダの周期を持つタスクを時間制約を守りながら処理しなければならないのと同時に、オペレータからの処理要求に応じた非周期なタスクもこなさなければならない。オペレータの操作によって発生する非周期タスクは、クリティカルなデッドラインを持つものではないが、システムの的確な操作を保証すると言う点で、高速な応答性が要求される。

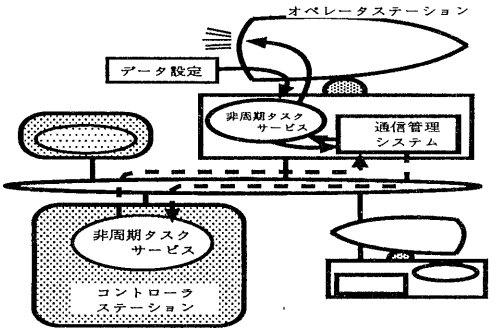

図 1: 制御用マンマシン システムの構成

例えば、制御用マンマシン システムはシステムの状態表示、オペレータからのデータ入力処理等を行なうオペレータ ステーションと、システム制御を行なうコントローラ ステーションで構成される(図1)。オペレータ ステーションでは帳票収集やトレンド収集などの周期タスクと、データ設定などのイベント駆動の非周期タスクの処理が主に行なわれている。オペレータの操作によって発生する非周期タスクは、コントローラ ステーションとのコミュニケーションを通して処理されるため、その応答性はコントローラ ステーションで非周期タスクがどの様に処理されるかに影響される。また、周期タスクにも他のタスクとのメッセージ通信によって処理を実行するものがある。この場合、メッセージを受け取ったタスクは非周期タスクとして処理が行われるために、やはり、その応答性が問題となる。

一方、コントローラ ステーションではプラントを制御するための周期タスクが高プライオリティで実行されている。そのため、オペレータ ステーションからの非周期なタスクは通常バックグラウンドで処理され、その応答は遅いものとなってしまう。しかし、コントローラ ステーションからの応答性を向上させるために、非周期タスク処理を割り込み(ポーリングなど)で行なえば、周期タスクの時間制約を守った処理を保証することができなくなる。

オペレータによるシステムの的確な操作、オペレータとシステムとの円滑なコミュニケーションを実現するためには、これら非周期タスクの応答性を向上させることが必須であり、周期タスクの処理への影響を最小にしてコントローラ ステーションでの非周期タスクの処理を効率化しなければならない。

3 非周期タスクのスケジューリング

3.1 周期タスクのスケジューリング

実時間スケジューリングの理論は、他のシステムにおけるスケジューリング理論とは異なり、対象とするシステムに対してどのようなタスク セットを与えた場合に、タスクのデッドラインを守って処理可能であるか、といったシステムの予測可能性に重点がおかれている。

Liu と Layland は、その後のリアルタイム スケジューリング理論においてベースとなる二つのスケジューリング手法を 1973 年に提案している [4]。それらのスケジューリング手法は周期タスクを対象としたものである。一つはタスク スケジューリング時に処理要求を出しているタスクの中で、デッドラインの最も迫っているタスクに最高のプライオリティを与えるものであり、Earliest Deadline First (EDF) と呼ばれる。タスク τ_i の実行時間と周期が C_i、$T_i\,(i=1,...,n\ \ T_i \leq T_j)$ と表し、稼働率 U を

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

とした時、EDF によるスケジューリングがフィージブルであるための必要十分条件は $U \leq 1$ を満たせば良く、100% の CPU 稼働率が得られることが示されている。しかし、EDF は各タスクのデッドライン計算、タスク スイッチが頻繁に発生しオーバーヘッドが大きく、あまり現実的なものと見られていない。

もう一つは、Rate Monotonic(RM) と呼ばれる手法である。このスケジューリング法では、対象とする周期タスクの周期が短いものから順に高プライオリティを設定する。したがって、各タスクのプライオリティはシステムの起動時に一度計算する(もしくは設計者が設定する)必要があるだけで、タスク スケジューリングのためのオーバーヘッドのあまり大きくない現実的な手法である。また、数学的にもシンプルであるために、RM をベースとした研究が数多く発表されている。

RM によって、m 個のタスクをスケジュールした場合、

$$U = m(2^{1/m} - 1) \qquad (1)$$

で表される稼働率よりも小さな値を持つタスク セットについては、タスクの実行時間、周期によらず、各タスクのデッドラインが保証されることを彼らは示している。したがって、タスクの数が十分に多い場合には $U \simeq 0.7$ となり、どのようなタスクセットであっても、その稼働率 U が 70% を越えない場合にはデッドラインを越えること無くタスクの実行を終了することができる。

さらに Liu たちの結果から、Lehoczky らは RM を用いた場合のデッドライン保証のための必要十分条件を与えている [3]。あるタスク セットに対して

$$L = max_{1 \leq i \leq n} L_i \leq 1 \qquad (2)$$

の不等式が満たされれば、そのタスク セットはスケジュール可能であることを示している。ここで

$$L_i = min_{t \in S_i} \sum_{j=1}^{i} C_j \cdot \lceil t/T_j \rceil / t, \qquad (3)$$

$$S_i = \{k \cdot T_j | j = 1, ..., i; k = 1, ..., \lfloor T_i/T_j \rfloor\}$$

である。S_i は τ_i のスケジューリング ポイントであり、τ_i よりも高いプライオリティをもつタスクによって及ぼされる、臨界時刻でもある。

3.2 非周期タスクのスケジューリング

現在の実時間システムでは、制御用マンマシン システムのように周期タスクの時間制約を満足し、加えてオペレータからのシステム データ表示要求等の非周期タスク処理の高速応答も要求されるものが多い。上述の RM スケジューリングは、周期

タスクの時間制約保証を対象としており、非周期タスクはバックグラウンドサービスで処理されるために高速な応答はのぞめない。また、非周期タスクの応答性を向上させるためにポーリングサービスなどの割り込み処理などを行なった場合には、周期タスクのデッドラインを保証することができなくなる。

Lehoczky らの提案した Deferrable サーバ (DS)、Sporadic サーバ (SS) はこれらの問題を解決する一つの方法である [2] [5]。これらのサーバは、周期タスクのスケジュールを RM で行うことが前提となっており、Lehoczky らは、インプリメントが容易であること、オーバーヘッドが少ないことから RM スケジューリングを選んだ。

DS は、周期タスクに対応してキャパシティ、周期の二つの量を持つ。キャパシティは周期タスクの実行時間に対応しており、非周期タスクをサービスするためにあらかじめ確保された CPU 時間である。非周期タスクが到着すると、非周期タスクは DS の周期でスケジューリングされ、キャパシティの時間分だけの処理が実行される。キャパシティは非周期タスクを処理するのに要した CPU 時間に比例して減少していき、処理中にその値が 0 になった場合には、その時点で非周期タスクの処理は中断さる。このキャパシティの値は、サーバの周期毎に一定値に補填される。中断されていた非周期タスクの処理は、この補填が行なわれた時点でまた処理可能となり、再びスケジューリングされることになる。

SS のアルゴリズムでも、DS と同様に非周期タスクをサービスするための周期タスクを生成する。DS と異なる点は、キャパシティの補填時刻の設定の方法とその補填量である。SS での補填時刻と補填量の決定は、システムが処理しているタスクのプライオリティ レベルの変化と、その時点においてサーバが消費したサービス時間 (すなわちキャパシティ減少量) とで決定される。簡単には、サーバの補填時刻はサーバが非周期タスクの処理を始めた時刻 t_0 からサーバの一周期後に設定され、補填量は、t_0 に始まった非周期タスクのサービスが終了するまでに要した時間分 (途中でより高いタスクにプリエンプトされた場合はその時点まで) である。

3.3 非周期タスク サーバの実システムへの適用

彼らの提案する非周期タスクサーバの手法を実際のシステムに採り入れた時に、非周期タスクの応答時間はどれだけ改善されるだろうか。この場合、RM スケジューリングが持つ仮定によってプリディクタビリティがどのような影響を受け、それによって非周期タスク サーバを用いた場合の非周期タスクの応答性にどのような変化があるのかを見なければならない。

RM スケジューリングをベースとして提案された多くの研究では、タスク プリエンプションが任意の時刻に行われることを許してスケジューラビリティの検討が行われている。しかし実際のシステムにおける周期タスクのプリエンプション、もしくはタスク スケジューリングのタイミングはユーザ (設計者) の設定したタイム インターバル (システム クロックの整数倍) 毎でしかない。処理されるタスクの周期が、スケジューリング インターバルに対して十分に大きい場合には、周期タスクのエントリーとそのスケジューリングの時刻をほとんど同時とみなすことができる。しかし、タスクの周期がスケジューリング インターバルと同程度の高周期である場合には、離散的なスケジューリングの効果はスケジューラビリティに少なからぬ影響を及ぼすと考えられる。

このような点から、Katcher らは Lehoczky らの条件式 2を拡張し、タスク スイッチやタイマーをハンドルするためのオーバーヘッドを含めた式

$$\forall i,\ 1 \leq i \leq n$$

$$min_{(k,l)\in R_i}\{(\sum_{j=1}^{i}\frac{C_j+C_{preempt}+C_{exit}}{lT_k}\lceil\frac{lT_k}{T_j}\rceil) + \frac{lT_k}{T_{tic}}\frac{C_{timer}}{lT_k} + \frac{T_{tic}}{lT_k}\} \leq 1 \quad (4)$$

$$R_i = \{(k,l)|1 \leq k \leq i, l = 1, ..., \lfloor\frac{T_i}{T_k}\rfloor\} \quad (5)$$

を提案している [1]。

ここで、$C_{preempt}$、C_{exit}、C_{timer} は、タスク プリエンプション、タスク終了時、タイマー割り込み処理のオーバーヘッドを各々表しており、C_{tic} はタイマー割り込み間のブロッキングを表している。

彼らの条件式は実システムでの状況をより正確に評価できるものとなっている。しかしながら、タスク、特に非周期タスクの処理状況や応答性を確認するためにはシミュレーションによる検討が必要となるだろう。

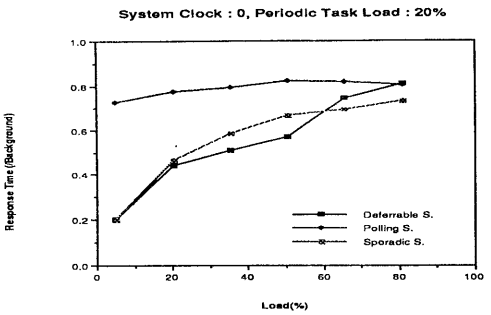

図 2: 周期タスクの負荷 20% での応答時間

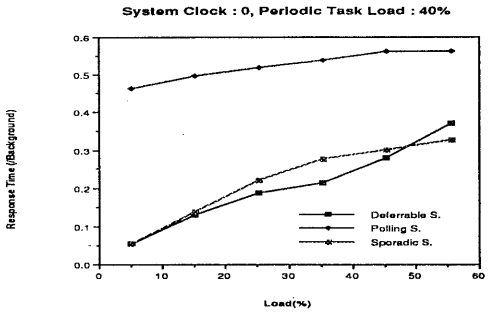

図 3: 周期タスクの負荷 40% での応答時間

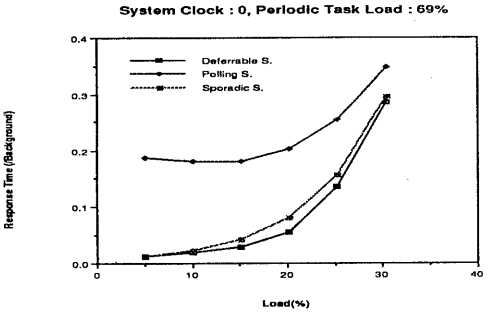

図 4: 周期タスクの負荷 69% での応答時間

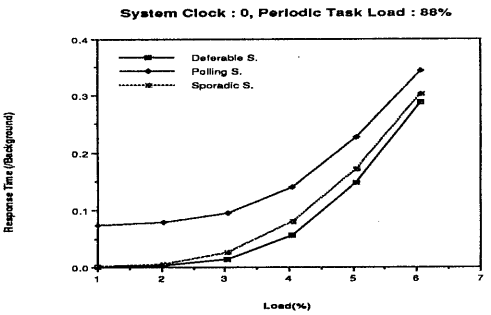

図 5: 周期タスクの負荷 88% での応答時間

4 システムクロックを考慮した非周期タスクの応答性

4.1 シミュレーションによる評価

制御用マンマシン システムにおいて、オペレータによるシステムの的確な操作を実現させるためは、オペレータの操作に対するシステムの応答が十分に高速なものとなっている必要がある。そこで、オペレータの操作によって発生した非周期タスクの応答に影響を及ぼす、コントローラステーション側での非周期タスクの処理に、非周期タスクサーバ (DS、SS) を用いた場合の応答性改善を検討した。特にコントローラステーションでのタスクスケジューリングが、インタバルタイマーによって離散的に実行されている場合に、タスクの実行にどのような影響が現れ、非周期タスクの応答性がどのように変化するかを見た。

タスクのプリエンプションをシステムクロック毎 (離散的) に行うことによって、非周期タスクサーバの性能がどのような影響を受けるのかを見るために TSA(Task Scheduling Analyzer) を用いてシミュレーションを行った [6]。TSA は各タスクの周期、実行時間、デッドラインといったエレメンタルな量を与えることによって、タスクのスケジューラビリティ、応答性、非周期タスクの応答性のチェックを行なうことができる、イベント駆動型のシミュレータである。

非周期タスクは Deferrable サーバ (DS)、Sporadic サーバ (SS)、Polling、バックグラウンドを用いて処理を行った。ここでのシミュレーションは、周期タスクのスケジューリングを Rate Monotonic (RM) で行い、これに対応したスケジューラの起動はシステムクロック毎とする。非周期タスクはネットワークを介して割り込みとして発生し、スケジューラを起動し、非周期タスクサーバの持つプライオリティを受けてすぐにスケジューリングされるものとする。

シミュレータには、周期 50 ～ 1200 単位時間の周期タスクを与え、そのデッドラインは各々の周期に等しいものとする。非周期タスクの発生間隔は、平

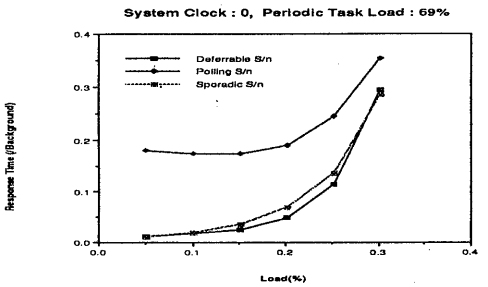

図 6: システムクロック 0 での応答時間

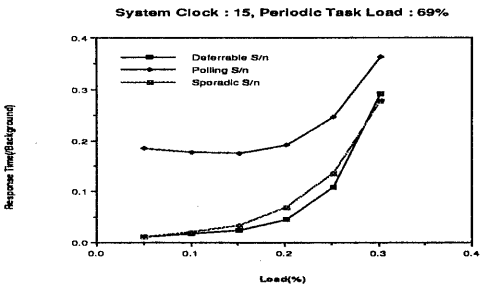

図 7: システムクロック 15 での応答時間

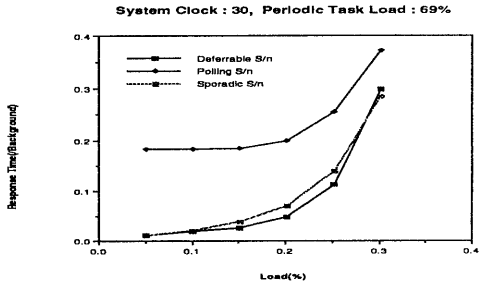

図 8: システムクロック 30 での応答時間

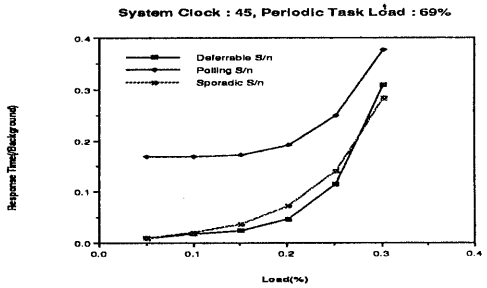

図 9: システムクロック 45 での応答時間

均到着時間間隔を 18 単位時間としたポアソン分布によって与え、その実行時間は、指数分布を用いてランダムに与えるものとする。非周期タスクサーバの周期は 50 単位時間とする。スケジューリングインターバルはもっとも周期の短いタスクの周期を越えないこととして、15,30,45 単位時間の場合についてシミュレーションを行い、周期タスクの負荷を 20% ～ 88% に変化させ、対する非周期タスクの平均応答時間の変化を見た。

4.2 周期タスクのデッドライン

非周期タスクサーバを含めたタスクセットは、Katcher らの条件式 4 を満たしており、デッドラインはどのタスクに関しても守ることができた。シミュレーションを行ったタスクセットの場合、各タスクの待ち時間の平均はスケジューリングインタバルに対して 30 ～ 40% の増加が見られた。

4.3 非周期タスクの応答時間

DS と SS による、非周期タスクの応答時間改善の効果を図 2～5 に示した。縦軸は各サーバでの応答時間を表しているが、バックグラウンドでの処理に対する応答時間の改善を見るために、各サーバの応答時間をバックグラウンドでのそれによって規格化してある。周期タスクの負荷があまり高くない場合において、応答時間に顕著な改善が見られるのは、サーバの持つキャパシティに対する非周期タスクの実行時間が十分に小さいためである。しかし、実行時間がサーバのキャパシティを越えるような非周期タスクが頻繁に発生する場合であっても、その応答時間はバックグラウンドのみでの処理の場合の 40% 以下となっており、非周期タスクサーバを用いる場合の可能性を示している。

図 6～9 は周期タスクの負荷を 69% に設定した場合の非周期タスクの応答性である。横軸は非周期タスクの負荷である。今回のシミュレーションでは、非周期タスクサーバの周期が全周期タスクの中で最も短いものの一つである。したがって、非周期タスクサーバは最高優先度を持っており、キャパシティが完全消費されていない場合、非周期タスクは割り込み処理によってすぐにスケジューリングされることから、スケジューリングインタバルが 0 の場合 (連続なスケジューリング) との顕著な差は現

れていない。

非周期タスクサーバの実システムへ適用を考えた場合、システムクロックによる影響は主に周期タスクのスケジューラビリティに対して現る。したがって、非周期タスクサーバを含めた形でタスクセットがスケジュール可能であるならば、非周期タスクの応答性を向上させるために、非周期タスクサーバは有効であることがわかる。

5 おわりに

今回我々は、制御用マンマシンシステムへの非周期タスクサーバの適用性を検討するために、システムクロックを考慮し、タスクが離散的にスケジューリングされる場合に、非周期タスクサーバによる非周期タスクの応答性がどのように変わるのかを検討した。その結果、システムクロックによる影響は周期タスクのスケジューラビリティに対して現れ、非周期タスクの応答性を向上させるためには非周期タスクサーバが有効に働くであろうことが得られた。

実際のシステムへの適用を考えた場合には、非周期タスクサーバのキャパシティ管理や補填時刻の管理を行わなければならず、そのことによるオーバーヘッドをさらに考慮しなければならない。今後はリアルタイムUNIX系のOSへの非周期タスクサーバのインプリメントを行い、その有効性の検討を行って行く予定である。

参考文献

[1] D. Katcher, H. Arakawa, and J. Strosnider. Bridging the gap between sceduling theory and reality. *Proc. Workshop on Architecture Supports for Real-Time Systems*, pages 86–90, 1991.

[2] J. P. Lehocxky, L. Sha, and J. K. Stronsnider. Enhanced aperiodic responsiveness in hard-real-time environments. In *Proc. IEEE Real-Time Systems Symposium*, pages 261–270. San Jose, 1987.

[3] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. *Proc. IEEE Real-Time Systems Symposium*, pages 166–171, 1989.

[4] C. L. Liu and W. James Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of ACM*, pages 46–61, 1973.

[5] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard-real-time systems. *The Journal of Real-Time Systems*, 1:27–60, 1989.

[6] 藤田 昌也, 竹垣 盛一 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 情報処理学会第*43*回全国大会講演論文集 *(分冊 6)*, pages 359–360, 1991.