推定伸長率に基づく動的負荷分散方式

久保秀士†

コンピュータシステムの構成は分散型に移行しつつあるが，ここから性能上のメリットを得るには負荷の分散方式が重要である．本論文ではクラスタ型システム上でトランザクション処理を行う場合を対象に負荷分散方式を提案し，シミュレーション評価によって従来方式を超える応答性が得られることを示す．ノードの応答性を動的に表現する指標として，走行中のジョブ集合の CPU-I/O 特性まで反映でき，収集の容易なデータに基づいて計算できる**推定伸長率**を導入し，これをベースとして動的負荷分散を行う．データ精度が落ちる場合の影響についても調査し，許容範囲と対策を示す．

Load Distributing Strategy Based on Estimated Expansion Ratio

Hidehito Kubo†

To make full use of clustered systems to get high performance, operating system should distribute appropriately the load of the system among its processing nodes. We propose dynamic load distributing strategies for transaction processing system, compare the performance of conventional methods and ours by simulation and show that proposed methods can offer better responsibility. The "expected expansion ratio" is introduced as dynamic performance index of a node, which represents dynamic performance ability of the node reflecting executing job set characteristics and can be calculated based on measured macroscopic data.

1. まえがき

コンピュータシステムは分散システム化の方向に進んでいる．サーバ内部における分散化は高性能化・高信頼化を目指しているが，高性能化の達成には適切な負荷分散が必要である．すなわち，各計算機ノードがつねに他と同水準の負荷を分担するように，仕事の配分を行う必要がある．負荷分散方式のうち静的負荷分散は，各ノードの性能，負荷内容や到着特性などの先験的知識をベースとして事前に決定した方法に沿って，負荷を確率的にあるいは固定的方法で配分する．動的負荷分散は，つねに各ノードの負荷状況を把握しつつ，この現状の知識に基づいて負荷を動的に配分して均衡を図る．静的方式は定常負荷を均衡させうるが，定常均衡下でも現実の負荷は変動しており，瞬間瞬間には不均衡が存在する．これを抑制し，さらなる性能改善を狙うのが動的方式である[1),2)]．動的方式をさらに分類すると，実行前のジョブの転送のみか，実行途中のジョブの移送まで行うかに分けられる．静的方式は理論的に解析可能な場合が多い[3)]が，動的方式は一般に解析が困難であり，理論的には最適性を保証できないヒューリスティックな制御方式となる．ここでは開始前転送のみの動的負荷分散を中心に扱う．

負荷分散を自由度高く動的に行えるためには，構成ノード間の情報伝達が低コスト・高速であるとともに，個々のファイル装置がどのノードからも性能的に等距離にあることが望ましい．汎用機はこの条件を満たすクラスタ型構成への移行をほとんど終えた．オープン系サーバも大型領域ではクラスタ構成へと向かっており，コンピュータ間結合網の最近の高速・大容量化とファイル装置群の SAN（Storage Area Network）化の方向が，理想的な負荷分散を可能にする環境を整えつつある．本論文の目的はこのような状況に適した負荷分散の方式を提案し，その有効性を示すことにある．対象システムは上述のような広い意味でのクラスタ型システムとし，その上で，ビジネス用途で中心的な形態であるトランザクション処理が実行されているとする．向上の目的とする性能指標は応答性である．

負荷分散に対する問題意識は並列処理・分散処理の発生とともにあり，30 年近くにわたって多くの研究の対象となってきた．しかし，従来の研究はほとんど UNIX などの TSS が対象であり，また，文献 4) などを除きリソースとしてはノードの内容まで踏み込まないものが大部分であった．ジョブの特性はノード処理時間だけで表現され，**CPU-I/O 使用特性**を考慮するものは非常に少なかった．ビジネス用途ではファイルアクセスの頻度が高いので，入出力を考慮せずに適

† 日本電気株式会社情報通信メディア研究本部
Computer & Communication Media Research, NEC Corporation

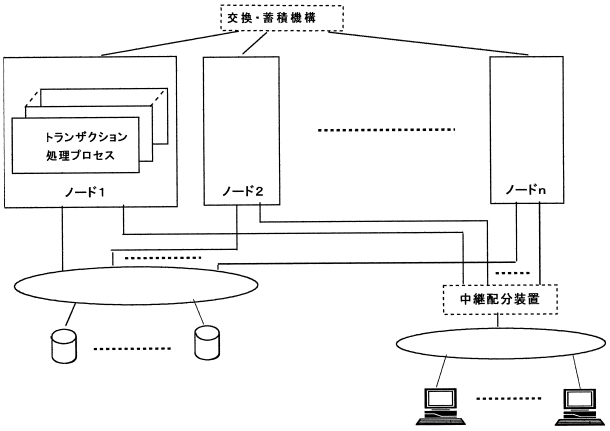

図 1 対象システム
Fig. 1 System model for load distributing.

切な負荷配分は困難と思われる．もちろん，制御のダイナミック性を上げ，負荷状況把握の基礎データを増やしさえすれば良い制御になるとは限らない．負荷指標の選択に代表される収集情報を適切に生かす制御アルゴリズムと，収集データの精度を上げる工夫とがないと，かえって性能を下げることにもなりかねない．

制御の判断基準となる**負荷指標**については，負荷というイメージには素直に対応し，間接的には性能指標値に影響するノード上のジョブ数などが多く使われていた[2),5)]．たとえば文献 2) ではノード上のジョブ数を負荷指標とし，情報収集や転送先選択の様々な方式の比較評価をしている．また文献 6) は，それ以前に行った様々な負荷指標を用いた実測結果から最適とされた CPU 待ち行列長現在値を採用し，情報収集方策の比較評価を行っている．しかし，負荷の指標は性能目標に直結していることが望ましい．また，推定応答時間を負荷指標としていても，個別最適化にとどまり全体最適化になっていない場合が多かった．さらに，LAN などによる分散システムを対象としていたので，負荷情報の交換やジョブ転送のオーバヘッドが大きく，これを回避しつつ負荷分散の効果を上げることに力が注がれていた[2),7)]．**負荷情報の精度**が結果に及ぼす影響についての検討もされたが[1),6)]，十分ではなかった．現在の特にクラスタ型システム内部における負荷分散問題は，上述のように状況が異なっており，動的なデータ収集が容易になっているのでこれを生かし，また，精度についての検証も行うことが望まれる．

以上から，現在の状況に適合し，より動的制御の効果が上がるような新しい負荷分散方式の確立が期待される．本論文ではこの要請に対応し従来法の欠点を克服する方式として，**推定伸長率**に基づく負荷分散方式を提案する．各ノードで実行中のジョブの集まりのマクロなジョブ特性を反映する応答性指標である伸長率（純処理時間に対する実処理時間の倍率）を導入し，無理なく測定可能なデータからこれを推定し，得られた推定伸長率をベースとして各ノードの負荷レベルを推定して負荷分散を行う．本論文ではシミュレーションによって提案方式を評価し，想定したシステム環境でジョブ特性の異なる複数種のトランザクションが混在する負荷の場合，評価した範囲では従来方式を超える性能が得られたことを示している．

論文の構成としては，2 章で対象とするシステムの構成などの環境条件を述べ，3 章で推定伸長率に基づく負荷分散方式について説明する．4 章で提案方式とその他の方式との性能の比較，負荷データの精度の応答性能への影響などの評価結果を述べる．

2. 対象システム

以降における議論の前提として，想定する対象システムについて，その構成とソフトウェア条件の性能面から見たとらえ方（モデル）の概要，負荷分散のやり方に関する共通的な前提などを述べる．

2.1 システム構成

対象とするシステムの概念図を**図 1** に示す．それほど厳密でないクラスタ型システムを想定する．基本イメージは大型汎用機であるが，オープンサーバ製品もノード間結合やディスク共有の機構については大規模領域からこれに近づきつつある．

複数ノードからなるシステムの結合形態として，並列システム，分散システムもあるが，ビジネス用途で中心になりつつあるクラスタ型システムを対象とする．クラスタ型の特徴として次の 2 点があげられる．

(1) ファイル装置はシステム全体として共用し，どのノードからも同じ速度でアクセス可能．
(2) ノード間の結合が比較的緊密で，相互間のデータのやりとりが高速にできる．

各ファイル装置がそれぞれ特定のノードにだけ直結されていると，性能的観点から処理ノードを選択する際の自由度が非常に狭くなるので，(1)は負荷分散に望ましい特性である．また，動的負荷分散を行うにはノード間の頻繁な負荷情報交換が必要であり，トランザクションを他ノードに転送することも必要なので，ノード間通信のオーバヘッドは大きな制約となる．よって(2)も負荷分散に望ましい特性である．従来の分散システムを対象とした研究では，特に後者の制約を強く意識して情報収集方策に注力したものも多かった[2),7)]．次に，ノード構成の均質性について選択肢がある．負荷分散可能なために機能的には均質が前提となるが，性能的にも同一性能のノードからなる均質なシステムとする．図1の交換・蓄積機構は，ノード間の比較的高速なデータ交換手段である．中継配分装置は強力な負荷配分機能は持たず，せいぜい静的な配分だけを行う．動的負荷分散は処理要求を受けたノードが判断して行うものとする．

トランザクション処理のために直接的に使われるノード内の資源にはCPU（時間）と主記憶（容量）があり，ノード間共用の資源としてファイル装置（時間および空間）がある．従来は各ノードを分解できない1つのリソースとし，ファイル装置はノード内外に意識しないモデルが多かった．ノード上の動きはジョブを単位とするFCFSか，並列的動作でもPS（Processor Sharing）などでモデル化されていた．ビジネス用途ではファイル入出力が多いので，ここではCPUおよびディスクへのアクセスを区別して意識する．ただし，リソースの空間的容量は十分にあり，制約条件にならないとする．また，競合によるI/Oアクセスの遅延は考慮しない☆．ノード間の結合網は通信の速度・容量とも相当に大きいと想定しており，ジョブ転送時のオーバヘッドだけを考慮する．

2.2 トランザクション処理のモデル

トランザクション処理要求は端末装置から送り出され，端末が指定した/中継配分装置が決定したノードへ送られ，受けたノードが負荷分散を行う．どのノードでもすべての種類の処理が可能（基本的に，それらの処理ソフトが常駐）とする．

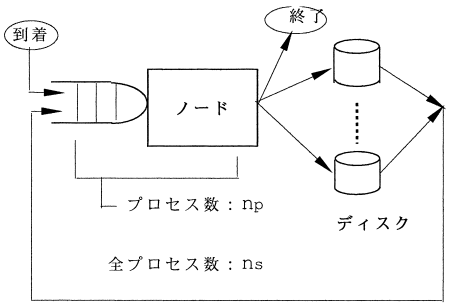

図2 ノードから見たシステム
Fig. 2 Node model in the system.

1つのトランザクションの処理は，割り当てた1つの処理プロセスによりすべて実行されるとする．ノード上では複数の処理プロセスが，**図2**のようにマルチタスキングで走る．処理プロセスは，実行するトランザクションの種類で定まるジョブ特性を持って，CPU使用とファイル装置使用を繰り返し終了する．

各々のトランザクションは処理の上で相互に独立とする．ジョブ特性は，従来の論文ではこれを意識する場合でも**純処理時間**（資源を実際に使用している総時間）だけで表現される例が多かった．ここでは，純処理時間とそれを構成するCPU時間と入出力時間の内訳などの形で表現する．論理的資源としてファイルレコード（排他制御対象）などが考えられるが無視する．複数種類のジョブ特性（トランザクションクラス）が混在する不均質ワークロードを想定する．図2はここで考慮する性能面から見たモデルであり，任意のノードから見たシステムを表す．ディスク群はすべてのノードから同じ位置づけにあり，同一性能特性で共用される．

2.3 負荷分散における前提

性能目標は応答時間とし，これが方式の評価尺度となる．一般に**平均応答時間**だけを見る場合が多いが，平均と標準偏差が小さいことのほかに，極端に長いものが少ないことなども性能指標として考慮する．制御法として動的制御を考察の対象とするが，比較のために静的制御も評価は行う．基本的に分散型制御とする．

処理要求は個々に各ノードへ到着する．負荷分散は一般に，到着した要求を処理前/処理中に他のノードへ転送することで実現される．実行途中で行うジョブ転送は移送（migration）と呼ばれ，その効果が検討されている[1),4),8)]．しかし，ここで対象とするトランザクションは一般に処理時間が短い（典型的には1秒以下）ので，移送はオーバヘッドの割に効果が小さいと考えられ[8)]行わない．開始前に判断して必要なら処

☆ 共用資源であるI/O装置へのアクセス時間はノード間負荷分散方式による違いは生じないので，I/Oでの競合はモデルから除外する．また，オンラインシステムではディスクは競合が問題にならない程度に時間的使用率を低くすべきとされている．

理要求のみを転送する．判断を起動する主体に関して，転送元ノード，転送先ノード，混合主導の 3 種が知られている[2]．後の 2 者は必然的に移送をともなうことになるので，ここでは**転送元ノード主導**とする．

以上は，負荷分散方式を検討するにあたって出発点とした共通的な前提である．これら以外に方式を構成する要素として，負荷の指標，負荷データの収集，転送の決定基準などがある．これらは各負荷分散方式を特徴づけるものとして検討されることになる．

3. 伸長率に基づく負荷分散方式の提案

まず，従来の文献における提案を振り返りつつ，トランザクション処理における負荷分散方式として望ましい特性を，特に上述の負荷指標とデータ収集について検討する．次に，これらの特性を備えると考えられる推定伸長率に基づく負荷分散方式の内容を検討する．まず，負荷指標のベースとする処理時間の伸長率について，定義を述べその推定方法を検討し，次にこの推定を行うのに必要な情報方策（データの収集とその精度向上処理）を，そして，以上を用いて計算可能になる負荷指標について検討する．最後に，これらをまとめて提案方式の実現アルゴリズムを示す．

3.1 トランザクション負荷分散の方針

方針としては何を負荷指標とするかが最も重要なのでこれの持つべき特性を考え，次に各ノードについて負荷指標の値を知るために必要となる情報方策の方針を検討する．

負荷分散対象の**ジョブ特性**に関しては，従来は個々のジョブまで考慮する場合は少なく，処理時間の内訳（CPU，入出力）まで考慮したのはさらに少数派，これを制御に反映させる検討はあった[4),9)]が，非常に少なかった．文献 4) では走行中の各ジョブの動作情報を収集して用いているが，十分に効果を上げたとはいい難い．本論文ではジョブの特性として CPU と入出力使用とを識別して扱い，この上に立つマルチタスキングも意識し，特性の異なる複数種のトランザクションの共存を想定し，この条件に対応する方式を考える．

動的負荷分散ではノードの現在負荷量の大小に応じてジョブの分散を行うので，何を**負荷の指標**とするかが重要である．個々のジョブ特性を意識する動的制御の論文の半数は特定ジョブの応答時間を予測し，これをノード負荷の指標として用いている．それ以外の，ジョブ特性を意識しないものを含む多くの文献では，「負荷」というイメージには合うが応答性にリニアに結びつくとはいい難い様々な指標が使われていた[2),5)]．負荷指標としては，制御の目標である応答性に直結するものであることが望ましいと考えられる．

応答性を負荷指標とする場合，**応答性の指標**について，多くは，各ノードで実行中の全ジョブの特性と進捗を知りうるとし，既知特性の新ジョブを追加実行させた場合の応答時間を得ている[10)]．しかしこれには，ジョブ特性の実行前把握と全ノードの最新情報の常時獲得の両面で無理がある．ミクロな個々のジョブの特性データに頼るのでなく，そのときに各ノードで実行中の複数の処理の，ジョブ群としてのマクロな測定可能な特性から応答性を推定できることが望ましい．また，負荷指標として推定応答時間を導入しても，上述のように「転送対象ジョブについての推定処理時間」をそのまま各ノードの負荷指標とするものが多かった．これは対象ジョブにとっての**個別最適化**であり，システムとしての**全体的最適化**になっている保証はない[11)]．全システム的な負荷指標を導入する必要がある．

動的負荷分散では，現在負荷レベルを知る必要がいつでも生じうるので，最新の負荷**基礎データ**をつねに備える必要があり，その**精度**が問題となる．**情報方策**は負荷分散方式を構成する重要な要素である[1),6)]．データ収集の手を抜くと負荷分散の効果が上がらないばかりか負荷の振動を起こし，逆に性能を下げることもある．過去の研究で，優れているはずの制御方式が簡単な方式より劣る結果となる場合が稀ではなかったが，精度の低いデータに依存したことによる場合が多いとみられる．トランザクション処理のように小さいジョブが頻繁に出入りするシステムでは，負荷の変化が激しいので特に要注意である．1 つにはデータの**鮮度**の点があり，収集オーバヘッドとのトレードオフが問題となる．想定環境は情報交換オーバヘッドが小さいので，このメリットを生かすべきである．もう 1 つはデータの**代表性**の問題であり，「必要なときに，その時点の現状を正確につかめること」が重要である．過去のデータを使うしかないが，それから現在を推定したり，知りうる現在データを用いた補正で最終測定後の変化を反映する，などして現状に近づく必要がある．

3.2 伸長率の推定方式の検討

負荷指標のベースになるものとして提案する応答性指標は処理時間の**推定伸長率**（Expansion Ratio）である．応答時間（待ち時間も含む実際の処理時間）の純処理時間（CPU と I/O の使用時間の和）に対する倍率であり，応答性そのものといえる．この概念自体は，負荷分散の最終的評価尺度としては方々で使われていた（slowdown[8)]，stretch factor[10)]，response ratio など）．しかし，動作中の状況の表現（したがって，動的制御の判断材料）に使われた例は知られていない．

伸長率を導出する対象システムは2章で規定したものを前提とする．全体は図1のような構成である．各ノードは図2のようにモデル化されるが，これは開かれた待ち行列ネットワークとなり，広範囲に適用できる解が知られている[12]．伸長率を推定するうえでの，したがって提案方式を導く前提とする，**ノードのモデル**はやや簡略化した次のようなものである．「外部からのジョブの到着はランダムであり，ノードのCPUは1台で短いタイムスライスによる均等サービスがされ，（2.1節で述べたように）ディスクには競合がない」．ジョブあたりの平均値として，処理時間（応答時間）を T，純処理時間を t，CPU使用時間を s_p，ディスク使用時間を s_d とする（$t = s_p + s_d$）．統計的平衡における平均値として，図2のように対象ノードの滞在ジョブ数を n_s，そのうちCPU系に滞在する（CPU使用中または待ちの）ジョブ数を n_p，ディスク系に滞在するジョブ数を n_d とし（$n_s = n_p + n_d$），CPU使用率を ρ とすると，上記のモデルでは，たとえば文献3)の式8.1，8.3にみるように，統計的平衡における平均処理時間 T は次の式で表される．

$$T = \frac{s_p}{1-\rho} + s_d = \frac{\rho\, s_p}{1-\rho} + t. \qquad (1)$$

$$T = s_p(n_p + 1) + s_d = s_p n_p + t. \qquad (2)$$

静的負荷分散は平衡平均値に基づく制御なので，これらの式をベースとする論文も多い．しかし，動的負荷分散は「（瞬間的な）現在の状態」に基づく制御であり，ρ あるいは n_p については測定で現在値を知って用いることができるとしても，ノード上でその瞬間に実行中のジョブの集まりの特性を表す t，s_p，s_d に関しては知る手段がない，という問題がある．静的と動的を比較した文献3)，13)では，動的方式として式(2)に基づく方式を試みているが，ジョブは同一クラスに属するとして平均値を用いていた．多クラスのジョブが存在する一般の環境では，そのときに走行中のジョブの集まりの特性を反映できる必要がある．ここでは式(2)をベースとして，観測可能な値のみを用いて，ジョブ特性を反映した**処理時間の伸長率**（平衡平均値）E を表す式(3)を得た．式の導出については付録に示す．CPU使用率 ρ と n_s を用いた表現も可能である．n_p が同一（式(8)よりCPU使用率が同一）ならば，n_s が大きい☆ほど伸長率は低いという関係などが表現されている．式(3)により伸長率に基づく動的負荷分散が可能になった．

$$E = \frac{T}{t} = \frac{n_s(n_p+1)}{n_s(n_p+1) - {n_p}^2}. \qquad (3)$$

☆ I/Oバウンドのジョブが多いとこうなる．

この式によれば，特定ノードについて現状を示す n_s，n_p の推定値が得られれば，その時点の推定伸長率が得られ応答性を推定可能と期待できる．ただ，求められた伸長率は平衡状態を前提としており，短期的な状況判断を行う動的制御に適用可能かという問題を持つ．これについては，基礎データの収集とそれに基づく現状の推定によりカバーすることを考え，その推定方式を3.3節で述べる．ここでは，この問題がクリアされれば，式(3)による伸長率の概念は動的負荷分散のための応答性指標として，3.1節で述べた望ましい条件をほぼ満たすものといえることを確認しておく．1）多クラスジョブ下において，走行中のジョブの集まりのCPU-I/O使用比まで含むジョブ特性を反映した応答性指標である．2）個々のジョブの情報を知らなくても，ノード上の現状を反映した応答性を表現できる．3）無理なく観測可能なデータに基づいており現実的である．4)「倍率」というとらえ方はリニアな指標であり，負荷の高低比較に適している．ただし，伸長率の式は簡単化されたモデルの上で得られたものであり，これをベースとした制御アルゴリズムの有効性は現実に近い場での検証を必要とする．

3.3 情報方策の検討

伸長率を求めるのに必要な基礎データは，ノードごとの処理中トランザクション数 n_s と各CPU系の滞在数 n_p である．つねに現在値を用いるとすると，全ノード分が必要な測定実施の面と代表性の面とから，特に n_p に関しては問題がある．そこで，一定時間ごとにサンプリングを行い，測定時点の現在値を得ることにする．測定の間隔はデータの鮮度に直結する☆☆．測定値は観測時点の瞬間値であり，実際にそれを使うときにもその値である保証はない．また，現在の瞬間値よりは，もう少し長期的な現状を表す平衡平均値の推定が望まれる（代表性）．長期間の平均をとれば安定性・信頼性は増すが，それは過去の推定値にすぎない．そこで，次の式を用いて**推定値**を得ることにする．e は推定値，n は最初からの測定回数，a は重みパラメータ（$0 < a \leq 1.0$ の固定値），m は測定値とする．

$$e_n = a\, m_n + (1-a) e_{n-1}, \qquad e_0 = m_1. \quad (4)$$

この式では，推定値は近い過去ほど重視する形（a が大きいほどそう）で過去の測定値を全部取り込んでいる．得られた推定値を n_{sE}，n_{pE} と表記する．

この推定値は信頼度が高いが，測定の中間には同一の推定値を使い続けることになる．実際には，この間

☆☆ 4章の評価例では n_s は100 ms程度の間隔で，n_p はその10倍以上の頻度で変化している．

表 1 ノードの負荷指標
Table 1 Load indexes for a node.

使用伸張率 / 負荷指標	スケジュール前 推定伸張率	スケジュール後推定伸張率 特性未知	特性推定
伸張率そのもの	$L_{p1} = E_p$	$L_{a1} = E_n$	$L_{k1} = E_k$
Tr. 数 × 伸張率	$L_{p2} = n_{sr} \times E_p$	$L_{a2} = n_{sN} \times E_n$	$L_{k2} = n_{sN} \times E_k$
総伸張率の増分	—	$L_{a3} = L_{a2} - L_{p2}$	$L_{k3} = L_{k2} - L_{p2}$

にもトランザクションの開始・終了はあり，ノード上のジョブミックスや n_s は変化する．動的制御に用いる**現在値**にはこの変化を反映すべきであり，補正が必要である．n_s は変化の頻度が低いので現在値としての信頼性は高く，掌握のオーバヘッドも小さい．n_s の現在値を n_{sp} と表記し，これを用いて補正を行う．n_{sp} は知りうるとするが，平衡平均値としての性格を持たせるため次のように**補正値** n_{sr} を得る．

$$n_{sr} = w\, n_{sp} + (1-w) n_{sE}. \qquad (5)$$

ここで w は重み係数（$0 < w \leq 1.0$）で，1.0 に近い値がよい．CPU 系滞在数については次のようにする．

$$\begin{aligned} n_{pr} &= n_{pE} + (n_{sr} - n_{sE}), \quad n_{sr} \geq n_{sE}, \\ &= n_{pE}(n_{sr}/n_{sE}), \quad n_{sr} < n_{sE}. \end{aligned} \qquad (6)$$

式 (6) は過去の測定値から推定した n_{pE} を，同条件で推定した n_{sE} と現時点の状況を表す信頼できる補正値 n_{sr} との関係から補正するものである．

以上を用いて計算時点の伸長率が得られるが，負荷指標として伸長率を考える場合，到着ジョブの各ノードへの配分後の状況を予測して判断する方法も考えられる．そのためには，式 (3) から，配分後の処理中数 n_{sN} と CPU 系滞在の平均数を予測すればよい．n_{sN} は，式 (5) における w を用いて $n_{sr}+w$ とする．CPU 系滞在数については，新ジョブの特性が推測不能な場合は予測値を上と同様 $n_{pN} = n_{pr} + w$ とし，可能な場合の予測値は n_{pK} と表記する．新ジョブの t に対する CPU 時間 s_p の推定比率 p_0 と，式 (10) から得られる配分前のノードのジョブ群としての CPU 時間比率 p_1 から配分後の比率 p_2 が得られる．さらに式 (10) を配分後の状態に適用すると次式が成立する．

$$p_2 y^2 + (1 - p_2 n_{sN}) y - p_2 n_{sN} = 0 \qquad (7)$$

これを y について解くと予測値 n_{pK} が得られる．

3.4 採用する負荷指標の検討

以上で得られる推定伸長率をベースとして，ノードの**負荷指標**（負荷分散の判断基準）をどう規定するか検討する．指標の分類軸として次の 2 つが考えられる．

- どの時点の負荷を基準とするか：配分前/後
  さらに，配分後の場合，到着ジョブのジョブ特性を推測可能か未知か，という区分がある
- 推定伸長率そのものか，さらに加工するか
  加工する場合の候補として，処理中ジョブ数を乗ずる/乗じた結果の配分前後の差

これらを組み合わせて**表 1** に示す 8 種類が可能であり，実施にあたってはこのうちの 1 つを選んで採用する．推定伸長率に関し，配分前の値（ベースは n_{sr}，n_{pr}）を E_p，到着ジョブ特性が推定不能な場合の配分後の値（同 n_{sN}，n_{pN}）を E_n，推定可能な場合の値（同 n_{sN}，n_{pK}）を E_k と表記する．負荷の指標は L で表し，配分前/後（未知・既知）を第 1 添字 $p/a/k$ で，伸長率の用い方を第 2 添字 1/2/3 で区別する．これらの指標は値が小さいほど負荷が低いことを表す．

配分前/後については，各ノードに配分してみた場合を予測して判断するのが望ましいはずである．ただし，新着ジョブの特性を知りうるとしても到着ジョブ個々は無理で，所属クラスの平均値程度であろうし，特性が不明なら予測の精度が落ちるので逆効果にもなりかねない．推定伸長率そのものを負荷指標とする L_{x1} に類する，新着ジョブにとっての最適配分の方法は多く行われてきた．伸長率を用いると，それだけでなく，応答性をノード間で均衡させる方向にもなるが，個別最適がシステム全体の最適につながるとは限らないとか，その時点では応答性が均衡しても近い将来までみた負荷均衡の観点からどうか，という問題を持つ．表 1 中段の負荷指標 L_{x2} は，ノード上の全トランザクションの伸長率の総和（**総伸長率**）という性格を持つ．その時点の応答性だけでなく，ノードとしての応答負荷（システム全体としての応答性への影響度）の状況を表しており，これに注目した制御は応答負荷の均衡を目指すといえそうである．下段の負荷指標 L_{x3} は，ノードにおける総伸長率に関し，新着ジョブを配分した際にそれによる増加分が最小であるようなノードを選択する意図を持つ．ノードの伸長率を能動的に変化できるチャンスのたびに，その時点のシステム全体としての伸長率総和の増加を最小にする選択となり，全体としての伸長率総和をつねに最小に保つ**全体最適化**を目指すことになる．

3.5 実現アルゴリズム

以上，制御方式の重要な要素についての検討と結果を述べてきたが，ここで，提案する負荷分散方式を整

理し具体的な動作手順として述べる．各ノードは**表 2** のようなデータ表を持つことが想定される．以下で，(1) はデータ収集であり，制御とは独立のタイミングで行われる．発生したトランザクション処理要求はいずれかのノードへ送られる．負荷分散の判断は処理要求到着を受けてそのノードで開始され，自分が処理するか，どのノードへ転送するかを決定する．転送の判断では，効果の不確実な転送を避け，また頻度を下げるために 2 つの閾値判断を導入している．(2) 以降はこれらの手順であり，負荷指標として表 1 のどの L_{yz} を採用する場合にも使える．

(1) 各ノード上で，定められた一定時間ごとに自ノードの負荷基礎データ n_s，n_p を測定し，式 (4) で最新の推定値 n_{sE}，n_{pE} を計算する．現トランザクション数 n_{sp} は各ノードが自分の値をつねに把握しておく．以上の値は変化する度に自ノードに記憶するとともに他の全ノードに通知して記憶させる☆．

(2) 各ノードの n_{sp}，n_{sE}，n_{pE} を用いて，すべてのノードについて式 (5)，(6) により補正値 n_{sr}，n_{pr} を得て式 (3) に代入し，自分の現推定伸張率 E_p と，採用する負荷指標 L_{yz}（表 1）で必要な種類の推定伸長率 $E_{p/x}$ を全ノードについて求める．

ここで，配分後の予測推定伸長率 E_x を必要とする場合は，3.3 節の方法で配分後の n_{sN} および n_{pN}/n_{pK} を予測して求める．

(3) 到着ノードの現在の推定伸長率 E_p が小さい値（たとえば 1.3 以下）ならば，無条件で自ノードでの実行とする．そうでない場合は次へ．

(4) 全ノードについて，(2) で得られた推定伸長率 $E_{p/x}$ を用い，採用した負荷指標の値 L_{yz} を求める．

(5) 負荷指標値 L_{yz} が最低のノードと到着ノードとで差が小さいなら到着ノードで処理する．大きいなら L_{yz} が最小のノードへ転送する．

負荷指標として L_{x3} 以外を採用する場合は倍率閾値とする．最小の 1.5 倍以内なら到着ノードで処理，などである．L_{x3} の場合は，2 ノード間の負荷指標値の差をシステム内の全トランザクション数で除したものが閾値を超えるか否かで判断する．

4. 性能評価

提案方式の性質・有効性を知るためにシミュレーションによる性能評価を行った．静的確率的には最適な配分をベースとし，そのうえでこれ自体と提案方式および従来の代表的な負荷指標であるノード滞在中ジョブ数，CPU 系に滞在中のジョブ数を用いた方式も同一条件で評価し，性質・有効性を比較した．特に，まえがきで述べたように，CPU-I/O 特性の異なる複数のクラスの小さい負荷が多数到着するようなビジネス用途環境に対する適性に注目した．また，採用した負荷指標による違いだけでなく，情報方策の効果も調べた．もちろん，システムそのものについても負荷内容についても，あらゆる場合を尽くしているわけではない．

表 2 負荷推定データ表
Table 2 Data table for load estimate.

ノード番号	1	...	i	...	n
現ジョブ数	n_{sp1}	...	n_{spi}	...	n_{spn}
推定ジョブ数	n_{sE1}	...	n_{sEi}	...	n_{sEn}
CPU 系の推定ジョブ数	n_{pE1}	...	n_{pEi}	...	n_{pEn}
配分前推定伸長率	E_{p1}	...	E_{pi}	...	E_{pn}
配分後推定伸長率	E_{x1}	...	E_{xi}	...	E_{xn}
推定負荷指標値	L_{yz1}	...	L_{yzi}	...	L_{yzn}

$x = n/k,\ y = p/a/k,\ z = 1/2/3$

表 3 トランザクション・クラス
Table 3 Transaction classes.

クラス	cpu 比率	av.cput/回	回数	av.cput/trn
Light	0.05	0.8 ms	28	22.4 ms
Medium	0.30	6.43 ms	21	135 ms
Heavy	0.60	22.5 ms	12	270 ms

4.1 シミュレーションモデル

4.1.1 システムモデル

2 章の記述を前提としてシミュレーションモデルを作成した．システム全般は SLAMII[14]，負荷分散のアルゴリズムは FORTRAN で記述した．システムは 8 ノード（同一性能の CPU 各 1 台を含む）からなる．入出力における競合はなし，CPU 割当てでは 20 ms のタイムスライスありである．

処理の実行モデルは 2.2 節で述べたとおりである．トランザクションとして，**表 3** に示す典型的な長さと CPU-I/O 特性を持つ L，M，H の 3 種類を想定した．純処理時間に占める CPU 時間の割合（s_p/t）が，平均 5%，30%，60%と異なる．純処理時間は各クラスとも平均 450 ms 程度，I/O 時間は 1 回あたり平均 15 ms とした．以上から，1 トランザクションについての CPU 使用—入出力の繰返し回数（固定値とする）など，表 3 に示す特性値が定まる．CPU 時間はトランザクション発生時に個々に，クラス平均値を持つ 10 相アーラン分布☆☆により 1 回あたりの平均値を発生して持たせる．

☆ 図 1 の交換・蓄積機構が蓄積機能を持つなら，ここに置いて全ノードから共用してもよい．

☆☆ 標準偏差が平均値の $1/\sqrt{10}$ で，分布の形は正規分布に近い．

CPU 使用のたびに，この平均値を持つ指数分布で使用時間を定める☆．ジョブミックスとしては，現実のトランザクションシステムで代表的と思われる，これらのクラス L，M，H の発生比率が 5 : 3 : 2 であるパターン A を標準とし，このほかに特性の違いが際だつ 2 クラスからなるパターン B（7 : 0 : 3）も用意した．

トランザクションの処理要求は，ノード別に，同一の平均値を持つ指数分布間隔で発生させ，確率的にクラスを割り当てる．静的には最適な確率的負荷分散が最初からなされており，その上で行う動的制御の効果を見ることができる．**負荷率**（システム負荷のレベル）は，CPU の平均使用率でとらえ，処理要求の到着間隔を変化させてこれを制御する．

トランザクション処理要求のスケジュールにかかわるオーバヘッドは，初期処理（到着ノード）と終了処理は各々10 ms，処理要求を他ノードへ転送する場合は送り側に 5 ms 追加，受け側も 5 ms（以上 CPU 時間），さらに 1 KB 程度と考えられる要求情報の転送の間に 5 ms の遅延（転送，排他制御）を設定した．いずれも固定値である．2.3 節で述べたように移送は採用しない．情報収集のオーバヘッドについては，測定そのものは本方式では小さく，ノード間の情報交換についても 2.1 節で述べたように無視しうる程度なので考慮しない．

4.1.2 評価対象とする負荷分散方式

性能評価対象として，動的分散なしの場合と伸長率に基づく提案方式のほかに，従来から提案されている方式などを含めて，負荷指標の異なる次の 6 種類について比較を行った．いずれも，3.5 節で述べたようにサンプリング測定は各ノードが自分について行い，制御は処理要求到着時に，その要求を処理するノードを決定するために到着ノードで実施される．

NC 動的制御なし．静的確率的には最適な配分．基本ケースとなる．

Ns ノード滞在中のトランザクション数 n_s による方式．データ収集は 3.5 節と同様に行う．n_s に関し，最小ノードが到着ノードより n_1 以上小さかったらそこへ転送する．

Np ノードの CPU 系に滞在中のトランザクション数 n_p による方式．n_p に関し，Ns と同様にする．

Lp1 L_{p1}（現推定伸長率）に基づく配分．

La2 L_{a2}（スケジュール後総推定伸長率）に基づく配分．

Lk3 L_{k3}（総伸長率の増分予測）に基づく配分．

n_s を負荷指標とする Ns 方式的なものは動的負荷分散の研究で多く採用されていた．たとえば文献 2) は 1 章で触れたように n_s を負荷指標とする様々な方式を比較している．n_p は CPU の負荷状況を反映していると考えられ，Np 的な研究例も比較的多かった．文献 6) では，1 章で触れたように n_p を負荷指標とし，様々な情報方策を比較している．到着ジョブを n_p が最小のノードに配分するのは，間接的に走行中ジョブミックスのジョブ特性を考慮し，それにとっての個別最適化を狙うことになる．最後の 3 方式は 3 章で述べた伸長率に基づく方式であり，代表的な 3 種類を選んだ．Lp1 は 3.4 節で定義したうち，最も基本的な負荷指標を使用し，La2 は到着トランザクション特性の知識が不要な場合の，試行結果が最良のものを使用した．Lk3 は 3.4 節で定義したうち，到着トランザクション特性の知識を使用する場合の，試行結果が最良の負荷指標を採用した．ここでは，到着時にクラスは検出可能とし，クラスの平均特性（5%，30%，60%）を既知とした．

ランダム均等到着を前提とした基本方式に加えて，図 1 の中継配分装置が最初に固定的な配分（定順で配分先を回す）を行い，受けた各ノードが動的負荷分散を行う場合も評価した．定順自体も，実システムでは負荷分散によく採用されている．これらのケースを c-NC など “c-” をつけて表示する．c-NC（定順のみ）は，動的測定・判断が不要な最良の方式として，比較の対象としても用いる．

シミュレーション評価は上の各方式を対象に行った．情報方策は負荷分散方式を特徴づける要素であるが，本評価では提案方式以外で使用する基礎データの入手についても，後述のように，現在値/推定値（式 (4)）採用の比較，推定用パラメータ値の事前評価など行い，許される範囲で最良の方法を用いるようにした．1 ケースにつきモデル上の時間で 30 分間，負荷率 0.8 の場合で 91,000 トランザクション程度を走らせた．

4.2 標準ジョブミックスの結果

標準ジョブミックス（3 クラス混合のパターン A）の下で，負荷率を変化させてシミュレーション評価を行った．制御のベースとなる負荷データの測定は，各方式にとって現実的に最適な条件とした．すなわち，全ノードの n_s 現在値をつねに知りうるとし，サンプリングは 10 ms 間隔とした．各方式の制御パラメータについては負荷率 0.8 において試行し，応答性が最も良かった値を採用した．広い負荷率範囲でほぼ最適であったので，各方式内ではそれぞれ同一値を用いた．

☆ どのシミュレーション実施でも，n 番目に発生したジョブどうしはまったく同じ動作系列を行うように設定した（同一乱数法）．

表 4 負荷率と平均応答時間の関係（3 クラス）
Table 4 Average response time vs. load level (3 class jobmix).

負荷率	NC	Ns	Np	Lp1	La2	Lk3	c-NC	c-Ns	c-Np	c-Lp1	c-La2	c-Lk3
0.10	481	475	480	477	474	474	467	467	467	467	467	467
0.20	501	484	498	490	484	484	473	473	473	472	473	473
0.30	527	494	517	504	496	496	484	484	484	484	484	484
0.40	567	512	539	522	511	511	503	502	502	500	500	499
0.50	619	537	563	544	532	531	532	528	527	523	522	521
0.60	706	572	592	576	562	560	577	563	559	554	551	551
0.70	847	624	632	623	608	605	652	615	607	601	597	596
0.75	971	664	667	660	644	639	714	652	642	637	630	629
0.80	1130	717	717	709	691	686	806	701	693	683	676	675
0.85	1420	805	809	792	772	763	964	782	781	763	748	745
0.90	1920	976	1010	952	914	902	1230	920	971	910	886	880

事前配分なしの場合の制御パラメータ選択に簡単に触れる．Ns，Np の両方式については負荷データの現在値を使用する場合と，現在値と過去の値とを用いた式 (4) による推定値とを試みた．Ns については現在値，Np では推定値（a=0.2）の結果が最良であり，それらを採用した．伸長率方式における推定パラメータは最良の a=0.1 を用いた．事前配分ありの場合や以後の節における評価でも，同様に，対象方式についてそこで許される範囲で最良の結果となる制御パラメータを選んで適用した．

事前配分なしとありの場合の，負荷率の変化に対する各方式の平均応答時間を**表 4** に示す．**図 3** には事前配分なしの場合を中心に平均応答時間のグラフを，**図 4** には同じく応答時間の標準偏差を示す．以下，数値的な比較は特に断らない限り負荷率 0.8 で行う．

平均応答時間で見ると，静的配分に比較して動的負荷分散は方式によらず大幅に向上した．静的配分の中では確率的最適配分（NC）よりも定順（c-NC）の方が良く，その向上率は 30%である．動的負荷分散を行う場合，各方式について事前定順配分を行うとやや向上している（2〜4%程度）．事前配分なしでも，動的分散の結果は NC に比較して 35〜38%の向上，c-NC に対しても 9〜15%の短縮である．動的方式の中では，伸長率に基づく方式の性能が明らかに良い．事前配分なしで，従来方式の代表的な Ns と比較して Lk3 は 4.3%の短縮であり，これは CPU 待ち時間だけを見ると 11.6%の短縮にあたる．事前配分なしの場合の結果が良い順序は，高負荷領域では，ほぼ Lk3，La2，Lp1，Ns，Np，c-NC，NC の順である．低負荷では，c-NC が最良でそれ以外は高負荷とほぼ同一順序だが，Lp1 と Ns が逆転している．事前配分ありでは全般的にやや向上したが，傾向としては事前なしの場合と類似している．動的方式間の格差が縮小された点と，c-NC 方式も含めて全方式が負荷率 0.4 までの低負荷では横一線

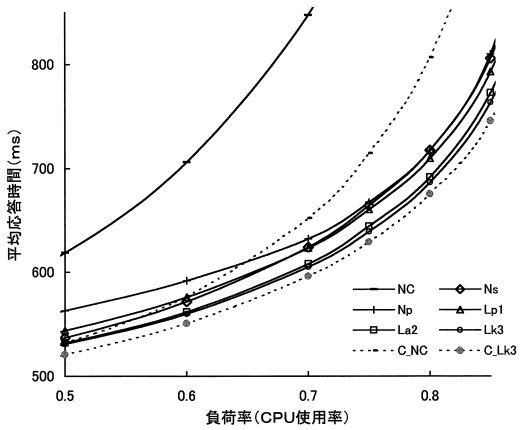

図 3 負荷率と応答時間の関係（3 クラス，部分）
Fig. 3 Response time vs. load level (3 class, partial).

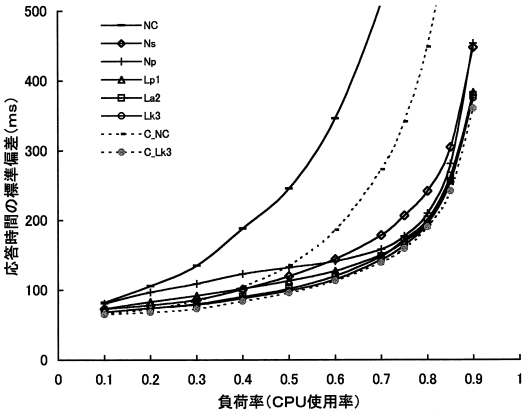

図 4 負荷率と標準偏差の関係（3 クラス）
Fig. 4 Standard deviation vs. load level (3 class).

の結果になった点くらいが変化で，負荷率 0.5 以上での方式間順位も，Np と Ns が入れ替わっただけである．

応答時間のばらつき（標準偏差）についても，静的配分に比較して動的分散は全般に大幅に向上している（図 4）．平均応答時間そのものより向上が顕著であり，

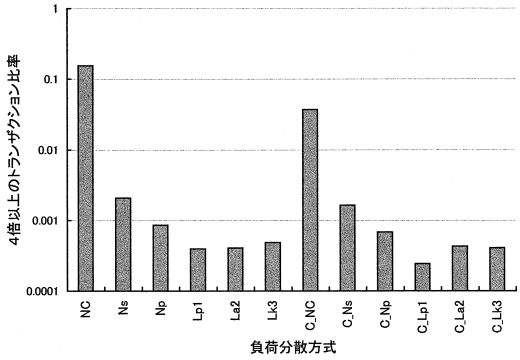

図 5 伸長率＞4 のトランザクション比率（負荷率：0.8）
Fig. 5 Ratio of transactions with ER>4 (load level: 0.8).

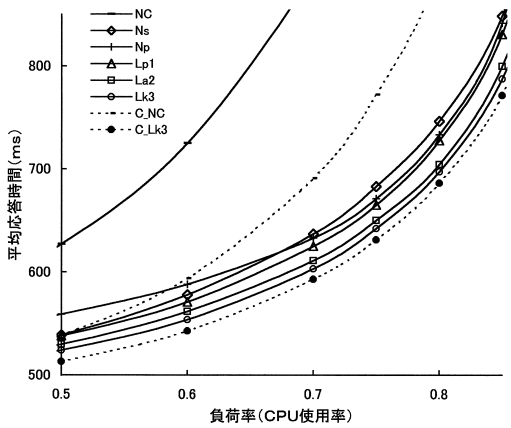

図 6 負荷率の変化と応答時間の関係（2 クラス，部分）
Fig. 6 Response time vs. load level (2 class, partial).

NC の 1/4，c-NC の 1/2 程度に減少している．静的配分としては優れている定順も，負荷率 0.5 を超えると大きく後退している．動的方式の中では，伸張率に基づく方式が全負荷範囲で安定的に良い．事前配分なしの場合，負荷率 0.4 までの低負荷では良い順にほぼ，Lk3，La2，Ns，Lp1，Np である．最悪であった Np は 0.75 あたりから伸張率方式に近くなっている．事前配分ありの場合の結果は，なしの場合より小幅ながら良くなった．負荷率 0.8 における，応答時間が純処理時間の 4 倍以上と極端に長いものの出現比率（対数表示）を**図 5** に示す．これは利用者の立場から見て直感的に分かりやすい性能指標と感じられる．伸長率方式は 0.05%以下であり，Np ではこの 2 倍，Ns で 5 倍，NC では 40 倍となっており，伸長率方式が明瞭な差を持って良い．

以下，使用した負荷指標の影響とデータの精度の 2 つの観点から結果を検討する．応答性全般について伸張率方式が優れている．なかでも Lk3 が最良となったのは，最多種の情報を用い全体最適化の指標も採用した結果で期待どおりである．一方，応答時間では La2 もほとんど差がなく，到着ジョブ特性情報なしでもいける場合があることが示された．Lp1 は高負荷では応答時間について La2 とやや離れて悪いが，ばらつきでは優っており，これは負荷指標の性格が明瞭に反映されたと考えられる．Lp1 は個別最適化を La2 は応答負荷のバランスを狙っていた（3.4 節）．ともに従来方式である Ns と Np の比較では，低負荷では Ns が圧倒的に良いが高負荷では応答時間は僅差，ばらつきでは Np の方が良くなっている．これは推定データの精度によるところが大きい．n_s は安定したデータ，n_p は特に低負荷では相対的に変化の激しい不安定なデータである（低負荷で Lp1 のばらつきが良くないのも，n_p 値への依存度が高いからか）．しかし，高負荷では推定（式 (4)）の効果もあって信頼性が高まり，Np 本来の CPU 負荷を直接的に均衡させようとする特性が，結果的にばらつきの小ささとして表れたものであろう．図 5 で Np，Lp1 が良いのも，個別応答最適化狙いの効果であろう．逆に Ns は数だけによる均衡なので，中高負荷でばらつきが悪くなっている．最後に，事前定順配分を行った場合の目立った結果として，低負荷では方式間の差がなくなったことがあるが，本環境で低負荷だと，定順配分を行うとそれ以上の動的制御は必要ないというのが実状（制御のための転送が非常に少ない）であった．また，方式として Np が Ns よりも良くなったが，定順配分と Ns とは同種の制御であり両者の組合せの効果は小さいが，Np は別種の制御なので相乗効果が出たものと考えられる．

4.3 2 クラス混合ジョブミックスの結果

ジョブ特性にコントラストのある 2 種類のクラスのみが存在するパターン B の下で，前節と同様に評価を実施した．図 3 と同様なグラフを**図 6** に示す．標準ケースと対比して大まかな傾向としては変っていない．平均応答時間は全般的にやや悪化したが，静的配分に対する動的負荷分散の向上度は拡大した．事前配分なしでも，動的分散の結果は NC に対して 39〜43%の向上，c-NC に対しても 16〜22%の短縮となっている．動的方式の中では方式間の格差が拡大され，伸長率方式の効果がさらに顕著になった．Ns に対して Lk3 は 6.5%の短縮率に向上した．標準偏差は標準ケースよりも全般的に相当に（16〜20%）悪化した．

以上の結果は，負荷を構成するジョブ特性の組合せが，より不均質であったことから説明できる．不均質さが著しいと応答性は一般に悪化するが，負荷分散方式によって不均質性への対応の程度に差があり，伸長

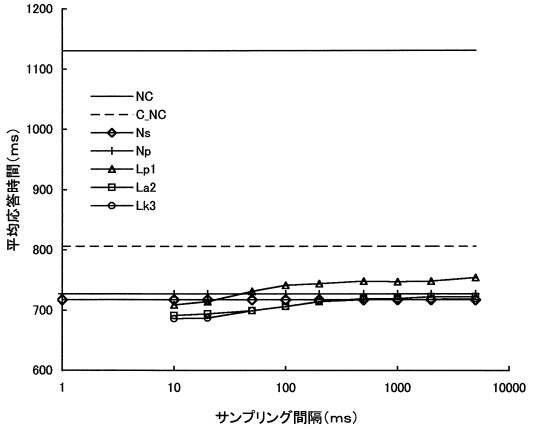

図 7 測定間隔の影響（現 n_s 可知）
Fig. 7 Response vs. sampling interval (present n_s known).

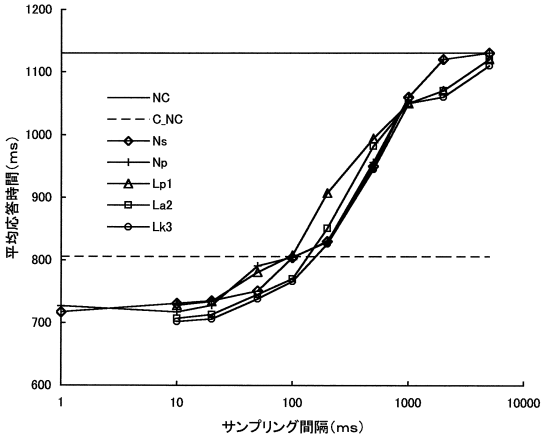

図 8 測定間隔の影響（現 n_s 不可知）
Fig. 8 Response vs. sampling interval (present n_s unknown).

率方式では悪化の度合いを小さく抑えることができ，これで Ns などとの差が広がったと解釈できる．Ns ではジョブの数だけに注目しているのに対し，伸長率方式における実行中ジョブ群の特性の反映させかたの妥当性がこの結果に表れていると思われる．Ns は順位的にも 4 位から 5 位に落ちた．

4.4 基礎データの精度の影響

以上では負荷データの測定は理想的に行えるという条件で評価した．実際には測定の困難さやオーバヘッドの問題から，そうはいかない場合も多いであろう．データ精度の悪化に対する耐性も方式の評価項目となる．ここでは使用する基礎データの精度が落ちた場合に，動的負荷分散の効果にどの程度の影響がでるものかを評価し，各方式の適性や性能劣化の回避方法などを検討した．標準負荷の下で負荷率 0.8 において，データの精度を落とす要因としてはサンプリング間隔の増大と，ノード上の現在トランザクション数データ n_s の利用に限度がある場合を取り上げた．

図 7 に，n_s，n_p の測定・推定間隔（対数表示）だけを変化させた場合の結果を示す．別に，必要なときにすべての n_s 現在値が得られるとし，これを用いる補正（式 (5)，(6)）は行われる．これに合わせて，Ns，Np の両方式では n_s，n_p 現在値をつねに使用可能とした．測定間隔は 10 ms から開始してほぼ倍増のペースで 5 秒までの 9 点である．各方式の制御パラメータは，測定間隔ごとに試行により最適値を決定した．図には参照のために，静的配分である NC と c-NC の結果も示した．図から，サンプリングを要する全方式について測定間隔は短いほど良い．3.3 節の式 (4) による推定は適用したものの間隔増大による劣化は伸長率方式ではそれほどでなく，10 ms と 5 秒で比較しても 30 ms（4.5%）の悪化にとどまっている．5 秒間隔でも，NC より 35%，c-NC と比較しても 10%短い．伸長率方式の La2，Lk3 は間隔 100 ms までは代表的な従来方式である Ns を上回り，5 秒でもほぼ互角に競えることが示された．つねに全現在値が得られるなら，サンプリングを要しない Ns，Np も使える方式といえるが，頻繁に変化する n_p 現在値の全ノードについての収集には問題がありそうである．

伸長率方式が測定間隔を長くしても強いのは n_s 現在値を用いた補正（式 (5)，(6)）の効果が大きいと考えられる．そこで，n_s（n_p も）の現在値がまったく使えない場合について評価した．伸長率方式は補正ができず，Ns，Np 方式もサンプリングに基づく推定値（式 (4) による）だけを用いる．平均応答時間の結果を**図 8** に示す．全 n_s 現在値を知りうる場合と異なり，全方式について間隔増大による劣化が著しい．知りうる場合と比較し，応答時間の悪化の度合いは間隔 10 ms では 2%台，100 ms では 10%程度，1 秒では 40%台，5 秒では 50%台となっている．NC と比較すると，伸長率方式は間隔 2 秒までは良く，5 秒で互角，Np，Ns はさらに悪い．c-NC と比較すると，Ns，Np は 100 ms で互角，伸長率方式も 200 ms で互角であり，動的制御として意味があるのは間隔 100 ms までといえる．Ns，Np 方式は，図 7 では現在値を用いたので，両図の差は直接的に測定間隔の影響を表している．伸長率方式については，n_s 現在値の使用可否による変化は補正の有無に反映されるので，両図の差は**補正の効果**を示すものと見ることができ，その大きさが分かる．

n_s 現在値がまったく使えない場合は悪化が激しいが，現実的には，他ノードは別として制御を行うノー

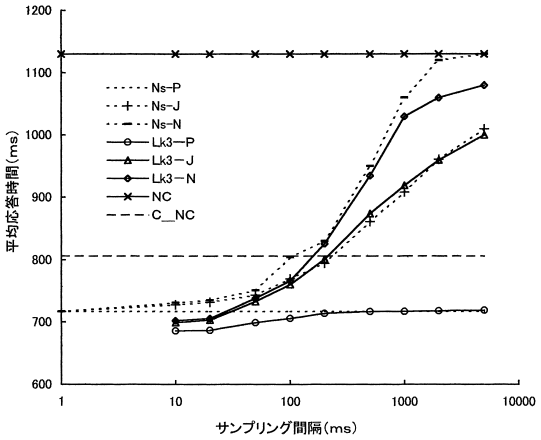

図 9 3 タイプの比較（Ns，Lk3 方式）
Fig. 9 Effect of n_s data usage (Ns and Lk3 methods).

ド自身の値は使えるはずなので，この場合についても評価した．伸長率方式では自ノードのデータにだけは補正をかけ，Ns，Np 方式では自ノードのみ現在値そのものを使う．最低限見込める効果として，測定間隔が長い場合にありがちな，過負荷であったノードが制御の結果過小負荷に陥る形の振動を防げることがある．このタイプを J タイプ，全 n_s 使用可能を P タイプ，全然使えないのを N タイプと呼ぶ．

図 9 に代表として Ns，Lk3 方式の各 3 タイプと，参照として静的配分を取り上げ，平均応答時間を示す．全般的に J タイプは N タイプより相当に良くなり，間隔 5 秒でも平均応答時間が NC を 10%ほど下回ったが，特に長間隔の場合 P タイプには大きく劣っている．定順（c-NC）との対比から，J タイプが意味があるのは測定間隔 200 ms までといえる．Ns 方式と Lk3 方式を比較すると，Lk3 は N タイプでは全般的に優り，J タイプでは 50 ms まで優るが以後はやや悪い，P タイプでは 200 ms 以上では互角だが 100 ms 以下では優っている．Lk3 方式の P タイプ・間隔 100 ms 以下は，本評価結果において最良の領域であり，動的制御の効果が最も現れている．

想定した環境では，到着ジョブ特性が推定可能なら Lk3 方式を，不確実なら La2 方式を，できれば P タイプ，測定間隔 100 ms 以下程度で適用するのが最良であるといえる．

4.5 想定以外のワークロード環境の考察

以上，代表的なジョブ特性のトランザクション 3 種類の環境で負荷分散方式を評価し，伸長率方式の有用性を示した．ここでは，ワークロードが想定と異なる場合を検討し，提案方式が有効な範囲を考察する．本論文で重視したジョブ特性は CPU-I/O 比率である．伸長率方式は，この特性の異なるジョブが混在する環境に有効な動的負荷配分を目指し，狙いに沿う特性を実現できたといえる．代表的と考えた 3 クラスの環境よりも，特性の違いが際だつ 2 クラス環境での方が従来方式との差が大きいことがこれを示している．特性がもっと異なれば，従来方式との差はさらに広がったであろう．一方，CPU 比率特性に大きな違いのないジョブミックスの環境では伸長率方式の特徴は薄まり，従来方式との差が縮小すると予想される．

想定ではジョブの長さを全クラス一律に平均約 450 ms としたが，長さのばらつきがより大きい場合を考える☆．動的負荷分散ではその時々の現負荷状況に応じた制御を行う．したがって，本論文のいずれの動的方式も，伸長率方式を含めて，走行中ジョブの余命の長さに影響を受けることはないはずである☆☆．これから，得られる性能の方式間の差は本評価の結果と同じ傾向に保たれると考えられる．4.4 節で評価した妥当なサンプリング間隔については，ジョブの長さは影響を持つと思われる．平均が長ければジョブ群特性の変化が緩やかになるので，測定間隔を延ばしても同等の効果を持つ制御が可能になるはずである．

5. むすび

計算機ノードが比較的緊密に結合されたシステムでトランザクション処理を行う場合に関し，推定伸長率に基づく動的負荷分散方式を提案した．推定伸長率は走行中ジョブ集合の CPU-I/O 特性まで反映し，収集の容易なデータで計算でき，ノードの各時点の応答性を扱いやすい形で表現できる指標である．制御の効果はデータの質に依存するので，収集データから現状を推定する方法も提案方式に含めた．8 ノードの処理環境を設定しシミュレーション評価を行った．データの精度が落ちる場合の影響も評価した．想定環境における評価では，提案方式は従来方式より優れており，負荷内容の不均質さが著しい場合に特に有効であること，データの精度が落ちる場合も，補正などでかなりカバーできることなどが示された．想定環境では，最良なのは提案方式を測定間隔 100 ms 程度以下で適用する場合であった．

提案方式の有効性は示されたと考えるが，適用場面を考えると確認・解決すべき点もある．システムとして，ノードが SMP 構成を含む性能的に不均質な場合を扱

☆ 静的方式は一般にクラス平均値の知識に基づいて負荷配分するので，クラス内のばらつきが大きいと有効性が損なわれる．
☆☆ 移送を行う動的負荷配分では，余命の長いジョブを選んで移送対象とするのがよいことが知られている[8]．

う必要がある．負荷内容に関しては，バッチ処理と共存する場合の負荷分散法が必要であり，ジョブ特性や到着特性の不均質性が大きい場合の確認・対応も必要であろう．また，集中制御型の適用も検討すべきであろう．今後はこのような問題にアプローチしていきたい．

参 考 文 献

1) 朴 圭成，芦原 評，清水謙多郎，前川 守：分散オペレーティングシステムにおけるプロセス移送の方式，情報処理学会論文誌，Vol.31, No.7, pp.1080–1090 (1990).
2) Shivaratri, N.J., Krueger, P. and Singhal, M.: Load Distributing for Locally Distributed Systems, *IEEE Comput.*, Vol.25, No.12, pp.33–44 (1992).
3) Kameda, H., Li, J., Kim, C. and Zhang, Y.: *Optimal Load Balancing in Distributed Computer Systems*, Springer (1997).
4) 李相てつ，計 宇生，松方 純，浅野正一郎：分散ベクトルを用いた負荷分散方式の検討，電子情報通信学会論文誌，Vol.J76-D-I, No.3, pp.118–129 (1993).
5) Casavant, T.L. and Kuhl, J.G.: A Taxonomy of Scheduling in General-Purpose Distributed Computer Sysems, *IEEE Trans. Soft. Eng.*, Vol.14, No.2, pp.141–154 (1988).
6) Zhou, S.: A Trace-Driven Simulation Study of Dynamic Load Balancing, *IEEE Trans. Soft. Eng.*, Vol.14, No.9, pp.1327–1341 (1988).
7) 渡辺 尚，太田 剛，水野忠則，中西 暉：双方向ピギーバックに基づいた動的負荷分散方式，電子情報通信学会論文誌，Vol.J78-D-I, No.3, pp.302–312 (1995).
8) Harcol-B., M. and Downey, A.B.: Exploiting Process Lifetime Distributions for Dynamic Load Balancing, *SIGMETRICS '96*, Vol.24, No.1, pp.13–24 (1996).
9) Aoki, Y., Suranauwarat, S. and Taniguchi, H.: A Load Distribution Scheme For a New Transation Service Considering the Pre-loaded Services, *IEICE Trans.*, Vol.E82-D, No.11, pp.1447–1456 (1999).
10) Goswami, K.K., Devorakanda, M. and Iyer, K.: Prediction-Based Dynamic Load-Sharing Heuristics, *IEEE Trans. Par. and Dis.*, Vol.4, No.6, pp.638–648 (1993).
11) Zhang, Y., Kameda, H. and Shimizu, K.: Parametric Analysis of Optimal Static Load Balancing in Distributed Computing Systems, *J. Information Processing*, Vol.14, No.4, pp.433–441 (1992).
12) 亀田壽夫，紀 一誠，李 頡：性能評価の基礎と応用，共立出版 (1998).
13) Zhang, Y., Hakozaki, K., Kameda, H. and Shimizu, K.: A Performance Comparison of Adaptive and Static Load Balancing in Heterogeneous Distributed Systems, *28th Annual Simulation Symposium*, pp.332–340 (1995).
14) 森戸 晋，中野一夫，相沢りえ子：SLAMII によるシステム・シミュレーション入門，共立出版 (1993).

付録 伸長率の式の導出

2 章で示した環境モデルを前提とし，3 章で規定した記号を用いる．以下，単独のジョブ特性クラスを扱う．式 (1) と式 (2) の対比から分かるように次の関係がある．

$$\rho = n_p/(1+n_p). \tag{8}$$

CPU，ディスクの平均使用率はジョブのそれぞれの平均使用時間に比例するはずである．ディスクでは待ちがないので，平均使用率（同時使用数）と平均滞在数は等しく，次が成り立つ．

$$s_d/s_p = n_d/\rho. \tag{9}$$

式 (8) と式 (9) から，$s_d n_p/(1+n_p) = s_p n_d$.

これに $s_d = t - s_p$ と $n_p + n_d = n_s$ の関係を適用して整理すると，

$$s_p = \frac{n_p t}{n_p + n_d + n_p n_d} = \frac{n_p t}{n_s(n_p+1) - {n_p}^2}. \tag{10}$$

式 (10) を式 (2) に代入すると式 (11) が得られ，これから式 (3) が得られる．

$$T = \frac{{n_p}^2 t}{n_s(n_p+1) - {n_p}^2} + t = \frac{n_s(n_p+1)t}{n_s(n_p+1) - {n_p}^2} \tag{11}$$

式 (1) に式 (8)，(9) を適用すると，同様に ρ による表現式が得られる．以上ではジョブクラスは単一であるとしているが，ポアソン到着のマルチクラスの場合にも，全体の平均値について成立する関係である．

(平成 12 年 12 月 15 日受付)
(平成 13 年 4 月 6 日採録)

久保 秀士（正会員）

1941 年生．1965 年東京大学工学部計数工学科卒業．同年日本電気（株）入社．研究所にてオペレーティングシステム，システム性能評価等の研究開発に従事．2001 年 2 月退職．著書「OS 概論」（共立出版）．昭和 46 年情報処理学会論文賞受賞．電子情報通信学会会員．

E-mail: h-kubo@mtf.biglobe.ne.jp