分散システムにおけるジョブ特性を考慮したジョブの配送先決定法

谷内 典行，太田 剛，渡辺 尚，水野 忠則

静岡大学情報学部

概要

ネットワーク上に様々な性能の計算機が混在している環境では，ユーザは分散システムが本来持っている能力の恩恵を受けにくい．本研究の目的は，ジョブをその特性に適した計算機に自動的に配送することによってユーザがシステムを利用し易くすることである．その手段として，計算機をリソースの差異によってクラスに分類し，ジョブを記述したスクリプトをクラス間で配送するという方式を用いる．このとき，履歴情報から得たジョブの特性情報を利用する．

1　はじめに

計算機システムの形態は，メインフレームなどの大型計算機からワークステーションやパーソナルコンピュータなどで構成されたネットワークシステムへと移り変わりつつある．これは，高速なデータ通信を可能とするネットワークが普及したことと，計算機の性能及びコストパフォーマンスが飛躍的に向上したためである．この結果，計算機システムのユーザは，ネットワーク上に物理的に接続されたコンピュータ資源に自由にアクセスすることが可能になった．

しかし，実際のネットワーク上には様々な種類 / 性能の，あるいは互換性の無い計算機が混在していたり，一部の計算機上からしか使用できない計算機資源が存在するなど，ユーザを混乱させる原因となっている．ユーザは，自分が現在使っている計算機ではできないことをしたいときや，より高度な処理能力を必要とするジョブを実行したいときには，自分でそれを実行できる計算機を探し出さなければならない．現在のネットワークシステムでは意識的に他の計算機資源にアクセスするという作業を行なわなければならないが，ネットワーク上からユーザが自らの要求を処理するのに最適なホストを探し出すことは一般に困難である．ネットワークシステムが本来持っている能力を最大限に引出すためには，空いている計算機を利用して負荷を分散させることが必要である．

われわれは，ジョブがどのような計算機資源を使うかを考慮して，そのジョブを配送することを考える．この，どのような計算機をどのくらい使うかという情報を，そのジョブのジョブ特性と呼ぶ．ここでいう計算機資源とは，CPU やメモリや計算機の I/O 能力の他に，周辺機器やソフトウェアも含んでいる．

A Job Dispatching Method Based on Job Characteristics in Distributed Systems
Noriyuki Taniuchi, Tsuyosi Ohta, Takashi Watanabe, and Tadanori Mizuno
Faculty of Information, Shizuoka University

ジョブ特性を考慮してジョブを自動的に配送することができれば，ユーザは計算機資源をネットワーク上のどこからでも透過的にアクセスできるようになる．

ジョブの資源利用を考慮した負荷分散の研究は，既にいくつか報告されている [1],[3],[4]．しかし，ジョブの特性を有効に利用する方法はまだ得られていない．Kunz [3] は，Unix が与えるさまざまな統計情報をもとに閾値制御で配送先を決定する方法を検討したが，複数のジョブ特性は考慮しないほうが良いという結論に達している．また，プロセスの使用時間と入出力時間をベクトルで表現し，ホストの負荷と到着したジョブの内積が最小の計算機に送ると性能が向上することをシミュレーション結果で示した例もある [4]．これらの研究では，ジョブの資源利用情報を使って負荷分散することを目的としているが，われわれは，ジョブを処理する能力が均一でない，異なったアーキテクチャがネットワーク上に混在した状況も考慮している．同じように，異なったリソースを持つ計算機が混在するシステムで負荷分散をするソフトウェアに Task Broker [5] がある．Task Broker はワークステーション群を１つの仮想コンピュータ資源として使えることを目的としているが，各計算機が利用可能な資源はそれぞれで個別に管理している．われわれのシステムでは，ネットワーク上の計算機をクラスタリングすることにより，ジョブの最終的な配送先を決定することを容易にしている．このクラスタリングについては 2.1節で詳しく説明する．

上で述べたように，われわれはジョブの CPU や I/O 使用量といった負荷だけでなく，さまざまな計算機資源の利用もジョブの特性と考えている．その目的のために，計算機クラスという単位にネットワーク上の計算機をクラスタリングする．計算機のクラスタリングは Zhou らが提案する Utopia [2] でも行なわれている．Zhou らは，ネットワーク上の計算機をクラスタリングすることは，ジョブ配送の際の通信量を減らし，大規模なシステムにもシステムを適用させることができると主張している．

本研究は，単に負荷分散システムをつくることだけでなく，最終的にユーザの要求により柔軟に答えられるようなシステムを構築することを目指している．ただし今回は，システムの概観と，ジョブ特性を考慮したジョブの配送方式について述べ，今後の方向について報告する．

2 システムの概観

この章では，本稿で提案するジョブ配送方式を使用したシステムの構成について述べる．システムの概観図を図 1に示す．本研究では，どのような計算機資源を使うかという情報を考慮して，ジョブをより適した計算機に配送する方式を用いる．これによって，よりユーザが使い易いシステムを提案することが目的である．われわれは，ユーザが使い易い環境を作るという目的のために，ジョブ配送方式とともにユーザエージェントやジョブスクリプトなどといったいくつかの概念を導入したシステムを用いることを考えている．ただし，1 章で述べたように，本稿ではジョブの配送先決定法に重点をおいており，現段階ではユーザエージェントなどの機能は限定されている．以下でこのシステムを構成する要素について説明していく．

2.1 計算機クラス

計算機クラスとは，ネットワーク上の計算機の中でジョブを実行する能力が同じ計算機を分類 (クラスタリング) したものである．ネットワーク上の計算機を分類することは，ジョブを実行するのにより適した計算機を探し出すときに役立つ．本研究では，以下に示すような基準に基づいて計算機を分類し，それを

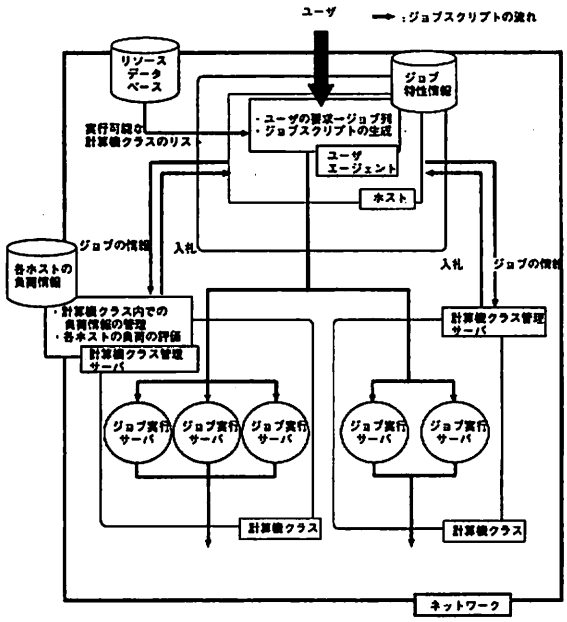

図 1: システム概観図

計算機クラスと呼ぶことにする．分類の基準は大きく分けて 2 つある．

第 1 の基準は，個々の計算機に使用されている CPU や OS の種類のような，ソフトウェアの実行形式の互換性に関するものである．この分類は，特定の計算機でしか使用できないジョブを実行するときに，そのジョブを実行可能な計算機を捜すために利用される．そして，その計算機クラスのリストを入手することにより，すべての計算機の互換性を把握しなくてもジョブを実行可能な計算機を認識することができる．

第 2 の基準は，CPU の能力，メモリ量，ファイル I/O 能力など，計算機のハードウェア能力に関する基準である．この分類は，あるジョブを実行するときに，そのジョブの特性に合った計算機を選ぶために行なわれる．ジョブの実行開始前にジョブ特性を予測するために，過去に実行したジョブの履歴情報を用いる．この情報はユーザインターフェースであるユーザエージェントが管理する．

Zhou らは，サーバなどの特別な計算機群を物理的なクラスタとは別に仮想クラスタとして二重に管理することにより，通信量を減らすことができるとしている [2]．この仮想クラスタは，計算機クラスの数を無意味に増やさないために，本方式でも有効な概念である．例えば，上に述べた 2 つの観点からは完全に同じでありながら，周辺機器の有無やサーバなどの特別な役割のために異なる計算機クラスとして分類しなければならないといった状況を避けることができる．本研究のジョブ配送法は，ジョブを配送する側とそのジョブを実行可能な計算機クラスとの間で入札を行なうので，同じジョブ実行能力を持つ計算機を細かく分類しすぎるのは好ましくない．従って，サーバとしての役割や特別な周辺機器を持つ計算機は，通常の計算機クラスとは別に，

仮想計算機クラスとして管理する．

各計算機クラスではクラス内の計算機の負荷情報が管理されている．ジョブを実行するのにより適した計算機を決定するためには，ジョブ特性の他に各計算機の現在の状態も考慮しなければならないからである．この目的のために，各計算機クラスにはクラス内の計算機の状態を管理するサーバ，計算機クラス管理サーバが1つずつ動いている．このサーバについては，2.5節で説明する．

2.2 ジョブスクリプト

ジョブスクリプトは，ユーザの要求を計算機が理解できるように展開したものを記述したものであり，ユーザエージェントによって作成される．ジョブスクリプトにはジョブを配送するために必要な情報が記述されている．その情報とはユーザの要求を実行するためのジョブのリストや，それぞれのジョブを実行できる計算機クラスのリスト，それぞれのジョブの特性などである．ジョブをどの計算機クラスで実行できるかという情報は，2.4節で説明するリソースデータベースが持っている情報から得ることができ，ジョブの特性は以前に実行した履歴情報から得られる．また，ジョブスクリプトを起動する時間や条件など，さらに細かい情報も付け加えることができる．

このジョブスクリプトはテキスト形式で作成される．ネットワーク上には様々な計算機が混在しており，その中に互換性の無い計算機も含まれていることを仮定しているため，このことは計算機の非互換性に影響されないために必要である．

作成されたジョブスクリプトはその結果をジョブスクリプトの中に追加しながらシステム内の計算機クラス間を移動し，要求を出したユーザエージェントに返ってくる．その結果，ユーザの要求はユーザの細かい判断を必要とせずに，そのジョブを実行可能な計算機の中の現在の状況に適した計算機で実行される．

このジョブスクリプトと同じように，ユーザの要求を他の計算機に処理させるためにネットワーク上を移動するものに，Telescriptがある [6]．各計算機で動作している，Telescript Engine と呼ばれるインタープリタに対して処理要求を投げつけるという方式は，ジョブスクリプトと似ている．しかし，ジョブスクリプトはより一般的なプログラム名が記述されているほか，利用したい資源をネットワーク上から探し出す，あるいは負荷を分散させるという目的を持っている点で，Telescript とは異なっている．

2.3 ユーザエージェント

ユーザエージェントは，ジョブ配送機構とユーザとのインタフェースである．ユーザの要求を受け取ってジョブスクリプトを作成し，返却されたジョブスクリプトからユーザが求める出力結果を生成する．

その主な仕事は，ユーザが出した抽象的な要求を計算機が理解できる具体的なジョブのリストに変換して，ジョブスクリプトを作成することである．ユーザエージェントはジョブの履歴を録っており，過去に実行したジョブならば，そのジョブの特性を知ることができる．このジョブ特性情報と，リソースデータベースから得られる情報を用いてジョブスクリプトを作成する．

2.4 リソースデータベース

リソースデータベースは，ネットワーク上に1つだけ存在し，ジョブを実行できるのはどの計算機クラスかといった情報を管理する．この情報はユーザエージェントがユーザの要求をジョブスクリプトに変換する際に用いられる．具体的には，そのジョブを実行できる計算機クラスのリストをユーザエージェントに与える．

2.5 計算機クラス管理サーバ

クラス管理サーバは，クラス内のホストの負荷を集中型で管理しているサーバである．ここでいう負荷とは，ホストが処理中のジョブのCPU量，メモリ量，I/O量のことである．これらの値は以前に実行したときの履歴情報から既に得られているものと仮定している．CPU量は，あらかじめ測定されているCPUの速度と，それを使用した時間の積で計算される．

また，クラス管理サーバは，処理すべきジョブスクリプトを抱えているホストからの処理要求を受け取り，入札する役割も持っている．処理要求には，処理すべきジョブの特性情報が含まれている．その情報をもとに，そのジョブをクラス内のどのホストで実行すればよいかを評価する．評価した結果は数値で表わされ，もっとも良い値を示したホストの評価値とアドレスを処理を依頼してきたホストに送り返す．ジョブを落札したかどうかは，負荷を管理する過程でホストがジョブを受け取ったことによって知ることになる．

3 ジョブの配送法

この章では，本研究で提案するジョブの配送法について述べる．本方式でのジョブの配送は全てジョブスクリプトを送りつけるという形式で行なう．以下に，ジョブを配送する手順を述べる．ただし，今回はユーザエージェントがジョブスクリプトを作成する部分は扱わないので，以前の実行時の履歴情報は既に得られており，ジョブスクリプトは既に作成されているものと仮定する．

1. ジョブスクリプトを保持しているホストは，計算機クラス管理サーバに対してジョブの処理要求をするメッセージを送る．そのメッセージには，実行したいジョブの特性に関する情報が記述されている．このとき，そのジョブを実行可能な複数のクラスに要求を送り付けることも可能である．ジョブを実行可能な計算機クラスのリストは，リソースデータベースが持つ情報により，既にジョブスクリプト中に記述されている．

2. 各計算機クラスの管理サーバは，入札アルゴリズムに従って，計算機クラスの中でそのジョブを処理するのに最も適していると予想されるホストのアドレスとその評価値を送り返す．

3. ジョブスクリプトを保持しているホストは，送り返されてきた情報に従って，スクリプトを配送する．複数の計算機クラスに要求を出したときは，この時点で最終的にどれに送るかを決める．具体的には，最も良い評価値を示すホストに配送する．

計算機クラスに分類することは，この配送方式において，入札時の処理要求にかかる通信を減らすとともに，計算機の現在の状態を管理することを簡単に行うことを可能にしている．また，計算機クラスサーバへの処理要求は，ジョブスクリプト中の1個目のジョブを実行時にはユーザエージェントが行ない，2個目以降にはジョブ実行サーバが行なう．

次に，処理要求を受けた計算機クラス内で，ジョブの実行に適したホストを選択するための配送法を示す．ここでは，ジョブが使用するCPU量(ジョブ量)，メモリ，I/O量を考慮している．これらの情報は以前に実行したジョブの履歴情報から得られ，ジョブスクリプトに記述されている．処理要求にはこれらのジョブの情報も含まれている．ただし，計算機クラスの性質上，送られてくる処理要求はクラス内のどのホストでも実行可能である．

配送法0. ジョブの個数が最小の計算機に送る．

配送法 1. 十分な空きメモリがあり，
$\max\{\frac{CPU_{host}}{S_{CPU}}, \frac{I/O_{host}}{S_{I/O}}\}$
が最小の計算機に送る．

配送法 2. 十分な空きメモリがあり，
$\max\{\frac{CPU_{job}+CPU_{host}}{S_{CPU}}, \frac{I/O_{job}+I/O_{host}}{S_{I/O}}\}$
が最小の計算機に送る．

配送法 3. 十分な空きメモリがあり，
$\max\{\frac{CPU_{job}+CPU_{host}}{S_{CPU}}, \frac{I/O_{job}+I/O_{host}}{S_{I/O}}\}$
あるいは，空きメモリが足りず (スワップ領域は空いている)，
$\max\{\frac{CPU_{job}+CPUhost}{S_{CPU}} + \frac{M_{job}-M_{host}}{S_{I/O}}, \frac{I/O_{job}+I/O_{host}}{S_{I/O}}\}$
が最小の計算機に送る．

S_{CPU}: ホストの CPU の速度
$S_{I/O}$: ホストの I/O 速度
CPU_{job}: 到着するジョブが使用する CPU 量

CPU_{host}: ホストが処理中の CPU 量
I/O_{job}: 到着するジョブが使用する CPU 量
I/O_{host}: ホストが処理中の CPU 量
M_{job} : 到着ジョブが使用するメモリ量
M_{host} : ホストの空きメモリ量

ここで使用されている到着するジョブに関する情報は，以前の実行時に得た履歴情報によって得たものである．配送法 0 は比較のために基準として用いるものである．配送法 1 は現在ホストが処理中の負荷のみに基づく評価法である．性能の異なる計算機のクラスからの情報と比較するため，CPU と I/O の速度で割って正規化している．配送法 2 は，ジョブの特性を考慮して，要求を受けたジョブを処理した場合の予想ターンアラウンドタイムが最小の計算機を選ぶ．配送法 3 は，さらにメモリが不足している場合のスワップにかかる時間も考慮したものである．

4 おわりに

本論文では，ユーザがネットワークシステムを使用する際に，より少ない努力で高い性能を得ることを目的として，ジョブの特性に基づいてジョブを配送するシステムの概観について述べた．また，負荷を分散させる手段として，簡単なジョブの配送法を紹介した．現在，本システムのインプリメント中であり，終了し次第，実測実験を行なって有効性を確認する予定である．

参考文献

[1] M. V. Devarakonda and R. K. Iyer, "Predictability of Process Resource Usage: A Measurement-Based Study on UNIX", IEEE Transactions on Software Engineering, Vol.15, No.12, (1989).

[2] S. Zhou, X. Zheng, J. Wang and P. Delisle "Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems", Software-Practice and Experience, Vol.23(12), pp.1305-1336, (1993).

[3] T. Kunz "The Influence of Different Workload Descriptions on a Heuristic Load Balancing Scheme", IEEE Transactions on Software Engineering. Vol 17, No.7, (1991).

[4] S. Ri, Y. Ji, J. Matukata and S. Asano "負荷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 D-1 Vol.J76-D-I No.3 pp.118-129, (1993).

[5] T. P. Graf, R. G. Assini, J. M. Lewis, E. J. Sharpe, J. J. Turner, M. C. Ward, 日経 エレクトロニクス, pp.143-154, 1993, 12.20 号

[6] 南 雅之:"Telescript のコンセプトと利用イメージ", 日経ニューメディア, 1994, 4.11 号