マルチメディア通信と分散処理 83－4
（1997. 6. 6）

ジョブ配送システム JAM/JC におけるユーザエージェントのジョブ特性推測方法の検討

久田 なつみ† 渡辺 尚†† 太田 剛†† 水野 忠則††

†静岡大学 理工学研究科 ††静岡大学 情報学部

分散計算機環境における多様なリソースをユーザが効率的に利用するためには、希望するサービスを提供できるリソースの位置を自ら把握し、現時点でのリソースやネットワークの状態に応じて要求の配送先を決定する必要がある。この問題に対し著者らは、ユーザやジョブの特性を考慮するジョブ配送システム JAM/JC を提案してきた。本研究では、JAM/JC におけるユーザエージェントのジョブ特性推測方法を検討する。本方式のユーザエージェントは、ユーザが実行したジョブの特性を履歴として保存する。新たなコマンドが投入されると、そのコマンドと履歴を比較し、MBR を用いて候補履歴を選択し、その候補履歴から現在のジョブ特性を推測する。本稿では、候補を選択する方法として、候補を一つにしぼる方式、複数候補の加重平均をとる方式について検討する。

Estimation methods of job characteristics in a job allocation agent in JAM/JC

Natsumi Hisada† Takashi Watanabe†† Tsuyoshi Ohta†† Tadanori Mizuno††

†Graduate School of Science and Technology, Shizuoka University
††Faculty of Information, Shizuoka University

In distributed computing environment, to efficiently utilize resources, a user needs to know the location of the resources, which can meet the demand, and to decide where user's requests send to. To the problem we proposed JAM/JC, which takes into account a user and job characteristics. We investigate methods to estimates job characteristics of a user-dependent agent in JAM/JC in this paper. The user-agent retains job characteristics which a user has run so far. When the user runs a new job as a request, the user-agent compares items in historical database with the new request and selects candidates which have less distance to the new request and estimates job characteristics of the request from the candidates. In this paper, we evaluate a method using one candidate and a method using several candidates.

1 はじめに

近年、複数の計算機資源をネットワークで接続した分散計算機システムが急激に普及しつつある。これは、計算機の高性能化、ネットワーク技術の進歩などによるものである。この結果、ユーザはネットワークで物理的に接続されたさまざまな計算機資源(リソース)にアクセスすることが可能となった。

しかし現実には、ユーザは利用可能なリソースの一部しか利用していないことが多い。これは、さま

ざまなリソースを効率的に利用するためにはネットワーク環境に関する多くの知識が必要となり、ユーザにとってこれらの情報をすべて把握し適切に処理することは困難であるためである。

一方、あるユーザは同じようなジョブを繰り返し投入する傾向があると考えられる。性質の異なるジョブはそれぞれ異なるリソースを要求するため、あるジョブを効率良く処理するためには、そのジョブの性質に最も適したリソースを持つホストを選択することが必要になる。つまり、システムの持つ性能を十分に活用するためには、ユーザは各リソースの特徴を知ると同時に、自分が投入するジョブの性質も把握する必要がある。

従来、分散計算機システムに関しては、システム側に立った研究が多い。例えば、各計算機の能力や現在の負荷状況などを観察し、ローカルシステムごとの負荷を均等化することによって、システム全体の処理効率の改善を目的とした多くの負荷分散方式が提案されている [1],[3],[4]。負荷分散は、負荷分散戦略の主導権を、ジョブを受ける側が取るか、送る側が取るかによって、ソース主導型・サーバ主導型に分けられ、さらに両方を用いる混合型も存在する。Shivaratri らは [1] で、ソース主導型・サーバ主導型・混合型それぞれで、しきい値制御を行う負荷分散について検討している。

しかし、本研究で対象としているユーザごとに異なる要求への対処法を検討する研究はほとんどない。Utopia [2] は、ユーザ側に立った研究の一つであり、分散システム上での総合的な利用環境を実現することで、ジョブ単位の負荷分散を実現している。しかし、ユーザごとのジョブ類似性など、ユーザ単位の特性は考慮されていない。

著者らは、リソースの特性とともにジョブの特性も考慮して自動的にジョブに適したリソースを選択し、そこへジョブを配送するシステム JAM/JC (Job Allocation Method based on Job Characteristics) を提案している [5]。ジョブ配送システム JAM/JC は、分散システムの本来の性能をユーザの負担を増やすことなく簡単に利用できるようにすることを目的としている。JAM/JC では、いかにユーザの投入するジョブの特性を得るか、そしてそれをいかに利用するかが問題となる。この作業を行うのがユーザエージェントである。本研究では特に、ユーザエージェントの構成法について考察する。

2 ジョブ配送システム JAM/JC

ジョブ配送システム JAM/JC の全体の概観を図1に示す。JAM/JC では、計算機ネットワーク上のホストをネットワークグループとリソースクラスによって分類し、それぞれのネットワークグループやリソースクラスごとにその構成や状態を管理する

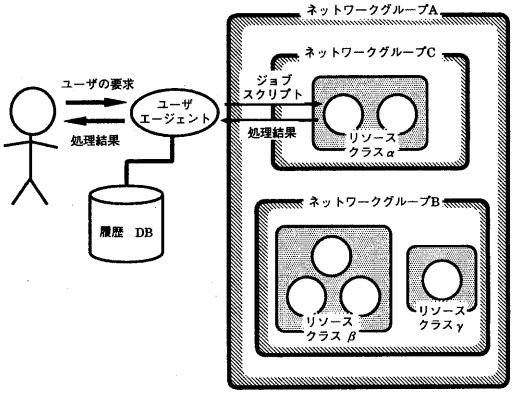

図 1: JAM/JC のジョブ配送システムの概観図

このように分類された計算機ネットワークに、ユーザからの要求が投入されると、各ユーザごとに設けられたユーザエージェントがそれを受け取り、そのジョブの特性を過去の履歴に照らして推測する。その後、ユーザエージェントによって作成されたジョブスクリプトは、計算機ネットワークに投入され実行される。ジョブスクリプトは、ジョブを実行するのに必要な情報やジョブの特性に関する情報を記述したもので、ジョブを配送する際に実際に送られるものである。

ここでいうジョブの特性には、以下のようなものが含まれる。

- ジョブが必要とする CPU 量・ I/O 量・メモリ量などの予測値
- アカウント等も含め どの計算機で実行可能か
- 制御端末を必要とするか
- X window system のクライアントであるか

3 ユーザエージェント

3.1 エージェントの役割

ユーザエージェントの役割を大きく分けると以下のように分類される。

- ユーザが実行したジョブについて実行時の特性を履歴として記録・保存する
- ユーザから投入されたジョブがどのような特性 (リソース使用量) を持っているのかを推測する

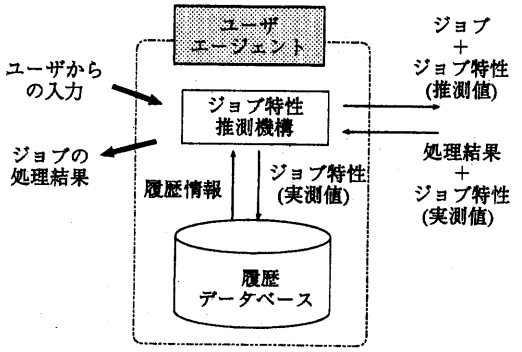

図 2: ユーザエージェントの構成

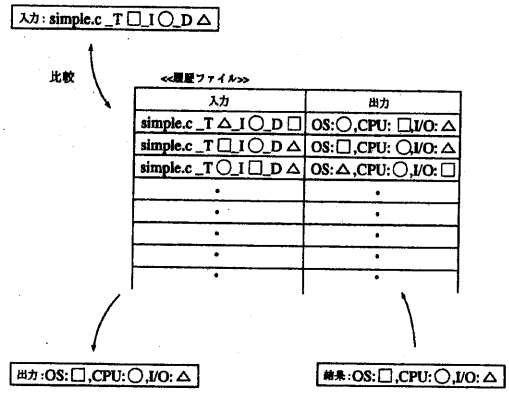

図 3: ユーザエージェントの動作

- 各リソースクラスに問い合わせてジョブの配送先を決定する

本稿では特に、ユーザエージェントの役割の一つであるジョブ特性の推測について検討する。

3.2 エージェントの構成

ユーザエージェントは、ジョブ特性推測機構と履歴データベースからなり、ユーザごとに用意される (図 2)。つまり、ユーザ個別の履歴データベースを持ち、そこからそのユーザのジョブ特性を推測する。

履歴データベース
: ユーザがそれまでに投入したジョブについて、その名前・実行時の性質 (選択されたリソース名・CPU 処理量・I/O 処理量など) が記録される。

ジョブ特性推測機構
: 履歴データベースに記録されたジョブの特性情報に基づいて、ユーザが新しく投入したジョブの特性を推測する。

従って、ユーザエージェントの基本的な動作は、以下の通りである (図 3)。

- ユーザから投入されたジョブを、その名前 (コマンドライン) を手掛かりに、履歴データベースのデータと比較する。その違い (差) の小さかったデータの特性から、新しく投入されたジョブの特性を推測し、配送先決定機構に渡す。
- ユーザエージェントは、ジョブの実行結果とともに、そのジョブの実行時の性質に関する情報を返送することを配送先決定機構に要求する。
- 結果が返送されてきたら、その結果はユーザに渡し、結果とともに返送されてきたジョブの性質に関する情報は履歴データベースに記録する。

4 ユーザエージェントの推測方法

本節では、ユーザエージェントの具体的な推測方法について説明する。

本研究では、複数のパラメータが存在し、そのパラメータの値がコマンドラインに反映されているようなジョブを対象とする。コマンドラインは、以下のようにプログラム名の後ろにパラメータが並んだ形であると仮定し、パラメータ部分のスペースで区切られた各部分をプレディクタフィールドと呼ぶ。

```
simple e4 s100 10.7 c1
```

プログラム名の部分を n で表し、プレディクタフィールドの部分を p で表す。

また、データベース内のジョブ履歴は、以下のようにコマンドラインの後ろに、リソース使用量などのジョブ特性が並んだ形式であり、このジョブ特性の部分をゴールフィールドと呼び、g で表す。

```
simple e4 s100 10.7 c1 : CPU 10, I/O 8
```

カレントジョブのプログラム名、プレディクタフィールドをそれぞれ $c.n, c.p$ と表し、データベース内のジョブ履歴のプログラム名、プレディクタフィールド、ゴールフィールドをそれぞれ $d.n, d.p, d.g$ と表す。具体的な手順としては、MBR(Memory-based reasoning) [6] を応用する。

(1) 各フィールドには、ジョブの特性を推測するうえでの重要性を反映した重みをつける。具体的な重みの計算方法は次の通りである。

- データベースを $[d.n = c.n]$ で制限する
- 各フィールドごとに $[d.p = c.p]$ となるデータをカウントする
- その中でさまざまな $d.g$ の頻度を調べる
- 頻度の二乗を足し合わせ その平方根を取り これを重みとする

(2) フィールドごとの重みを計算した後、改めて履歴データベース内のジョブ履歴とカレントジョブをフィールドごとに比較する。フィールドが一致するならば、その差を"0" とし、一致しない場合は (1) で計算したフィールドの重みをその差とする。

(3) (2) を繰り返して、コマンドラインの全てのフィールドに関する差を求め、それを足し合わせたものをジョブ履歴とカレントジョブとの差とする。

(4) このようにして、履歴データベース内のすべてのジョブ履歴とカレントジョブとの差を計算し、差の小さいものを検索し、そのジョブ特性情報からカレントジョブの特性を推測する。

以上を式に表すと以下のようになる。

$$\Delta(D,c,d) = \sum_{allp} \delta(D,c.p,d.p)$$

$$\delta(D,c.p,d.p) = \begin{cases} c.p = d.p & 0 \\ \text{otherwise} & \omega(D,c.p) * d(c.p,d.p) \end{cases}$$

$$\omega(D,c.p) = \sqrt{\sum_{v \in V_g} \left(\frac{|D[d.p = c.p][d.g = v]|}{|D[d.p = c.p]|}\right)^2}$$

ここで、$\Delta(D,c,d)$ は、データベース D における、コマンドライン c とジョブ履歴 d の隔たりの大きさを示している。また、$d(c.p,d.p)$ はコマンドのフィールド $c.p$ とジョブ履歴のフィールド $d.p$ の差を示している。

5 候補選択方式の検討

5.1 候補選択方式

5.1.1 TOLD 方式

この TOLD(Take One candidate with the Least Distance) 方式では、4 章の手順で計算された差が最も小さいジョブ履歴をカレントジョブに一番近いと判断し、そのリソース使用量をカレントジョブの予測リソース使用量とする。つまり、一番近いジョブ履歴一つのみを考慮する方法である。

5.1.2 WANT 方式

この WANT(calculate weighted Average value from multiple caNdidates within the Threshold) 方式では、推測手順の最後で差が小さいと判断する方法として閾値を用いる。ある閾値を設定し、4 章の手順で計算された差が、その閾値の範囲内にあるジョブ履歴をすべてカレントジョブに近い履歴とみなし、カレントジョブのリソース使用量予測の候補履歴と考える。さらに、各候補履歴のリソース使用量を、カレントジョブとの差が小さいもの程大きく反映させ、差が大きいもの程小さく反映させるような加重平均をとり、この結果をカレントジョブの予測リソース使用量とする方法である。この方法では、複数のジョブ履歴情報を考慮することになる。

5.2 実験結果と考察

4 章の方法を用いて、カレントジョブの予測リソース使用量を計算するとともに、実際のリソース使用量も測定する。予測リソース使用量と実際のリソース使用量とを比較することによって、リソース使用量の予測値が実測値にどれだけ近い値を予測できるか、つまり予測リソース使用量の正確さを検討する。具体的には、リソース使用量の予測値と実測値の差を示す。

今回検討するジョブには、パラメータに関して様々な組み合わせが存在する。この組み合わせすべてを網羅するのは不可能であるので、組み合わせの一部を保持するデータベースを用意する。このときデータベースの大きさによる比較を行うために、保持するジョブ履歴の数を変えた様々なサイズのデータベースを用意する。

5.2.1 TOLD 方式

図 4は、TOLD 方式のデータベースのサイズに対する、予測リソース使用量の正確さ (= リソース使用量の予測値と実測値の差) を示している。

一般には、以下のことが期待できる。すなわち、データの数が多い場合には、カレントジョブに近いデータも多く含まれるので、最小差のデータが、カレントジョブに近い可能性は高くなり、実測値に近いリソース使用量を予測できる。しかし、データの数が少なくなるにつれて、カレントジョブに近いデータがそのデータベースに存在する確率が小さくなる

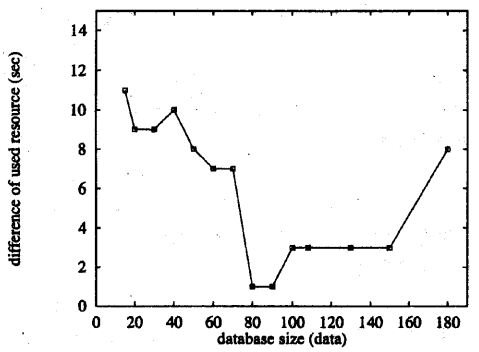

図 4: 予測値の精度 (TOLD 方式)

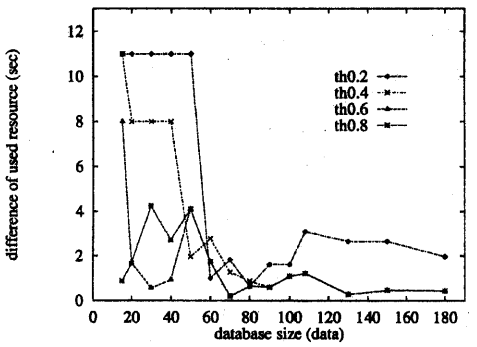

図 5: 予測値の精度 (WANT 方式)

ので、差が最小となるデータであっても、カレントジョブと似ていない可能性が高くなる。

しかし図 4のように、データベースのサイズが大きい場合でも、実測値とはかけ離れた値を予測してしまい、実測値との差が大きくなりこともある。当然、データベースのサイズが小さくなるにつれて、実測値に対する予測値の差はさらに大きくなる。この結果、予測値の正確さは非常に不安定といえる。

5.2.2 WANT 方式

図 5 は、WANT 方式のデータベースサイズに対する予測リソース使用量の正確さを示す。結果から、データベースのサイズが大きい場合には、実測値に近い値を予測できることがわかる。一方、データベースのサイズが小さい場合でも、ある程度以下になるまでは、予測値の正確さは、徐々に悪くなる程度である。つまり、TOLD 方式に比べると安定していることがわかる。

データベースのサイズが大きいときはもちろん、データベースのサイズが小さくなっても、一つのジョブ履歴だけでなく複数のジョブ履歴を考慮するので、加重平均をとる候補履歴の中にカレントジョブに近いジョブ履歴が存在する可能性が高い。

また、しきい値が 0.8(th0.8) の時に、データベースサイズによる変化があまり見られないのは、以下の理由である。候補履歴は、カレントジョブとの差をしきい値と照らしあわせて選択されるので、データベースが大きくなれば候補履歴の数も多くなり、データベースが小さくなれば候補履歴の数も少くなるため、データベースサイズの変化に適応できるためと考えられる。

6 ジョブ実行シーケンスを用いた実験

6.1 実験方法

本節では、実際にユーザがコマンドを次々と入力している状況において、ユーザエージェントがいかに実行時間を予測できるかを検討する。なお、しきい値およびデータベースの最大サイズは固定する。

この実験では、データベースはほとんどジョブ履歴を持たない状態から始まり、ジョブを実行するにしたがって、データベースに追加されていくので、サイズは徐々に大きくなる。そして、サイズが最大サイズに達すると古い情報の一部を変更し、常に最大サイズを保つ。このような設定のもとで実験を行い、データベースの更新によって予測リソース使用量の正確さがどのように変化するかを考察する。

候補を絞る方法として WANT 方式を採用し、データベースの最大サイズは 100、しきい値は 0.8 とした。また、データベースの更新方法としては、一番古い情報を新しい情報に置き換えるという方法を用いた。具体的には 5.2 節と同様に、リソース使用量の予測値と実測値の差を求め、その差のリソース使用量の実測値に対する比で検討する。

6.2 実験結果と考察

実行するジョブとしては、以下のような複数のパラメータを持つ待ち行列網シミュレーションを実行ジョブとするコマンドシーケンスを考えた。

連続法 最後のパラメータから昇順に変化させていく複数のシミュレーションを逐次実行する場合。

補間法 シミュレーション結果の概観を早く知るために、実行するジョブのパラメータ (例えば $a_1 < a_2 < a_3 < a_4 < a_5$) を、小・中・大・小・中・大の順番、つまり $(a_1, a_3, a_5, a_2, a_4)$ と補間しながら実行する場合。

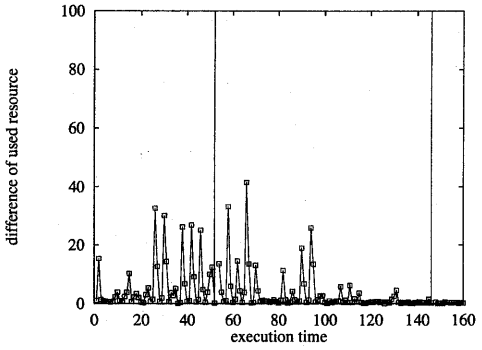

図 6: 予測値の精度 (補間法)

実験に用いたシーケンスは、4コのパラメータを持つ同一の実行形式プログラムが180コ連なったものである。

図6は、補間法でさまざまなパラメータの値を変化させてシミュレーションを実行した場合の結果である。横軸は時間の変化 (100 まではデータベースのサイズと等しい) を表しており、縦軸は予測リソース使用量の精度を示している。

結果から、データベースサイズが 70 ～ 80 までは、予測値が実測値に近いこともあるが、定期的あるいは不定期に大きくかけ離れた結果が観測され、不安定な状態が続いている。しかしそれ以降は、大きくかけ離れた結果が観測される度合いは小さくなり、安定してくるのがわかる。

データベースサイズが小さいということは、予測のための情報が少なく、カレントジョブに近いデータが存在しない可能性が高い。このため、予測されるリソース使用量も正しくない可能性が高く、予測リソース使用量の正確さは不安定となると考えられる。またジョブ履歴の数が少ないことから、各フィールドの重み計算も正確ではなく大雑把な値となってしまい、予測に重要なフィールドを把握できないため、予測の正確さが低くなると考えられる。

このようにユーザエージェントは、ユーザがコマンドシーケンスを入力するにつれ、徐々に正確な予測ができることがわかる。

7 おわりに

本研究では、ジョブ配送システム JAM/JC におけるユーザエージェントの構成を目的とし、その重要な役割の一つであるジョブ特性の推測方法について検討した。

ユーザエージェントは過去のジョブ履歴を記録した履歴データベースを検索して、カレントジョブと各ジョブ履歴との差を計算する。候補履歴を選択するための方法として、差の最も小さいジョブ履歴ひとつにしぼり、そのリソース使用量のみを考慮する TOLD 方式、差が一定範囲内にある複数のジョブ履歴を候補として選択し、そのリソース使用量の加重平均をとる WANT 方式について検討した。

両者を検討した結果、データベースサイズが大きい場合には、どちらの方法も実測値に近いリソース使用量を予測できた。しかし、データベースサイズが少し小さくなると、TOLD 方式では、予測リソース使用量の正確さが不安定になるという結果を示した。

また、実際の待ち行列網シミュレーションを実行ジョブとして考え、パラメータが異なるコマンドシーケンスに対して、ユーザエージェントがいかに振る舞うかを検討した。その結果、コマンドを入力していくにつれ、ジョブ履歴の数が増えていくので、予測の正確さが増していくことが確認された。

今後は、データベースの更新方法について、さらに検討していく必要がある。また、検討中のユーザエージェントを JAM/JC の中に組み込んで、その性能を確認する必要もある。

参考文献

[1] Niranjan G.Shibaratri, Phillip Krueger and Mukesh Singhal, "Load Distributing for Locally Distributed Systems", IEEE Computer, Dec 1992.

[2] S.Zhou, X.Zheng, J.Wang and P.Delisle, "Utopia: A load sharing facility for large, heterogeneous distributed computer", Software-Practice and Experience, vol.23(12), pp.1305–1336(1993).

[3] S.Ri, Y.Ji, J.Matsukata 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-I, vol.J76-D-I no.3 pp.118–129(1993).

[4] 染葉佳代子, 太田 剛, 渡辺 尚, 水野 忠則, "選択的多重ビギーバックを用いた負荷分散とその特性", 信学技報, SSE95-223, IN95-165, pp43-48(1995).

[5] 谷内 典行, 太田 剛, 渡辺 尚, 水野 忠則, "ジョブ配送システム JAM/JC の構成とプロトタイプ", 情報処理学会第 54 回全国大会論文集 4M-08, March 1997.

[6] C.Stanfill and D.Waltz, "Toward Memory-Based Reasoning," Communication of the ACM, vol.29, no.12, pp.1213–1223 (1986).