

双方向ピギーバックを用いた動的負荷分散における負荷情報の更新手法

久田 なつみ[†] 太田 剛^{††} 渡辺 尚^{††} 水野 忠則^{††}

[†]静岡大学 理工学研究科 ^{††}静岡大学 情報学部

分散計算機システムにおける各計算機の負荷の不均衡を解消し、リソースを有効利用するために種々の動的負荷分散制御が提案されてきた。動的負荷分散においては、他のノードの状態を知る方法が重要な問題の一つである。著者らは、トラフィックへの影響が少ない点、実行して初めてわかるジョブ量などに応じた負荷分散への拡張性の利点に注目して、双方向ピギーバックによって得た状態情報を得る方式を提案した。しかしこの方式では、状態情報の更新がおこなわれにくいノードが存在する。本稿ではこの点を解決するために、得られてから時間が経過した情報を更新する四つの方式を提案し、シミュレーションによって評価する。その結果、時間の経過に従って予想内客数を減少させる方式が最も短い応答時間を示した。

Updating strategies of load information for adaptive load balancing with bidirectional piggybacking

Natsumi Hisada[†] Tsuyoshi Ohta^{††} Takashi Watanabe^{††} Tadanori Mizuno^{††}

[†]Graduate School of Science and Technology, Shizuoka University

^{††}Faculty of Information, Shizuoka University

In distributed computer systems, several load balancing strategies have been proposed to balance the load of resources. In dynamic load balancing, an important problem is how to collect other nodes' load information. We have already proposed an adaptive load balancing strategy, which uses bidirectional piggybacking. Though it has an advantage of giving little influence to network traffic and enables load balancing based on the real quantity of jobs which is unpredictable before the execution, some nodes have out-dated information on others nodes. In this paper, we propose the four improved strategies updating out-dated load information and evaluate them. As a result, a strategy shows the shortest response time which decreases the predicted number of customers in proportion to elapsed time since its receipt.

1 はじめに

複数の計算機をネットワークで接続した分散計算機システムにおいて、各計算機の能力やそこに到着するジョブの大きさ、到着率などが異なるため、ローカルシステムごとの負荷に不均衡が生じ、システム全体の処理効率が悪くなるという状況が発生する。そこで、ローカルシステムごとの負荷を均等化させることによってこれを解消する、多くの負荷分散方式が提案されている [1],[2]。

負荷分散を行なう場合、その負荷分散アルゴリズムによって、静的負荷分散方式と動的負荷分散方式の二つに分類することができる。前者は、あらかじめ統計処理などによってシステムの動作状況を評価

しておき、その情報に基づいて固定された負荷分散アルゴリズムを用いる方式である。一方、後者は動作中のシステム状態を常に監視、評価し、その結果をもとに負荷の分散を行なう方式である。したがって、負荷状況が変化しても、そのときの状況に応じた負荷分散を行うので、どのような場合にも効果が期待できる [3],[4],[5]。

動的負荷分散を行なう際、システム内の負荷情報を収集し、その情報をもとにシステムに投入されたジョブの配送先を決定するディスパッチャが必要となる。このディスパッチャの形態は2種類に分類できる。集中型ディスパッチャは、単一のディスパッチャが、システム内のすべての負荷情報を収集し、それ

に基づいてすべてのジョブの配送先を決定するという形態である。このため、集中型ディスパッチャを用いる場合、収集された情報は正確であり、正しい負荷に基づいて適切なサーバにジョブを配送することが可能である。しかし、一箇所で情報を扱うために、扱うネットワークが大きくなると膨大な量の情報を一箇所に持つことになり、そのディスパッチャの負担が大きくなったり、そのディスパッチャだけでは情報を処理しきれなくなってしまうという欠点がある。これに対して分散型ディスパッチャは、システム内に複数のディスパッチャが存在し、それぞれが独立に負荷情報を収集し、その情報に基づいてジョブの配送先を決定するという形態である。したがって、各ディスパッチャが持つ情報は正確であるとは限らないので、その情報に基づいて行なわれる配送も適切なものであるとは限らない。しかし、ネットワークが大きくなっても、個々のディスパッチャが持つ情報の量を抑制することができる。

これらの点から、本研究では分散型ディスパッチャを用いた動的負荷分散方式についてシミュレーションによる検討を行なう。その際、サーバに到着したジョブを先着順に処理するバッチ式のシステムと一定時間ごとに交代でサーバを占有する時分割式の二つについて検討を行う。

2 LR-PB 方式

2.1 システムモデル

本研究では、LR-PB 方式 [6] を用いる。この方式では、システムを図 1 のようにモデル化する。

- 対象とするシステムは、ネットワークで結合された n 台のマシンから成る。
- 各マシンには、1 台のサーバと 1 台のディスパッチャがある。サーバ S_i をディスパッチャ D_i のローカルサーバ、 S_j を D_i ($i \neq j$) のリモートサーバと呼ぶ。
- ジョブの配送先決定に要する時間は 0 秒とする。
- D_i はすべての S_j と全二重回線で接続されている。
- 通信遅延はローカル間を 0 秒、リモート間を一定値 d 秒とする。
- ディスパッチャはローカルサーバの負荷状況が瞬時にわかる。
- D_i にはジョブが平均 λ_i 個 / 秒でポアソン到着する。
- ジョブの処理要求量は平均 $\frac{1}{\mu}$ 命令の指数分布に従う。

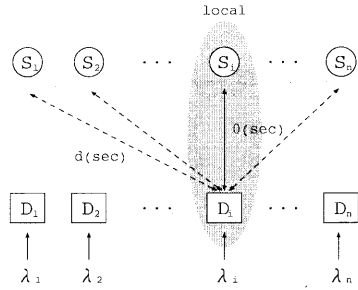


図 1: システムモデル

- ディスパッチャからサーバへ配送されたジョブは、その処理が終わるまでそのサーバから動かず、処理が終了するとそのディスパッチャへ返送される (ジョブの再配置は行わない)。
- S_i の処理速度を C_i 命令 / 秒とする。

2.2 シミュレーション条件

分散型ディスパッチャを用いた動的負荷分散を行う場合、問題となるのは次の二点である。

- 各ディスパッチャはシステム内の各サーバの負荷状態を把握している必要がある。この負荷情報をどのように収集するか。
- 各ディスパッチャは、持っている負荷情報をもとにジョブの配送先を決定する。このとき、どのようなサーバを配送先として選択するか。

このそれぞれについて、本研究で用いた方式を以下に示す。

2.2.1 負荷情報の収集

LR-PB 方式では、負荷情報の収集に双方向ビギンバックを用いる。これは、直接ビギンバックと間接ビギンバックの 2 種類のビギンバックからなる (図 2)。直接ビギンバックは、サーバ S_j がディスパッチャ D_i から配送されたジョブを処理した後、その時点でのサーバ S_j の負荷情報を処理結果に付加して D_i に返送する方法である。一方、間接ビギンバックは、到着したジョブをディスパッチャ D_i がリモートサーバ S_j に配送する際、ローカルサーバの負荷情報をジョブに付加して配送する方法である。サーバ S_j に届いた負荷情報は、そのローカルなディスパッチャ D_j に送られ利用される。これらによって、ジョブの配送が行われるたびに、ディスパッチャの持つ負荷情報が更新される。

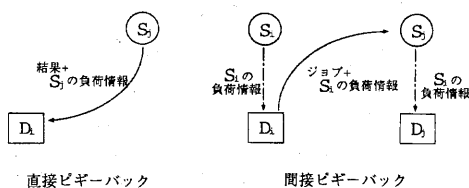


図 2: 直接フィードバックと間接フィードバック

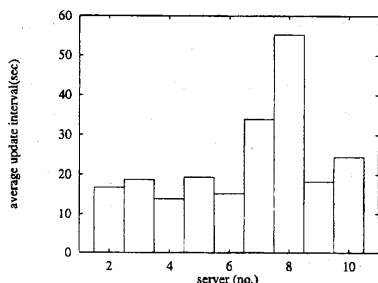


図 3: 負荷情報の更新間隔 (到着率 0.6)

2.2.2 ジョブの配送先決定法

ジョブが到着すると、各ディスパッチャは負荷分散方式に従ってそれらを配送するサーバを決定する。今回の実験で用いた LR-PB 方式では、ジョブが到着したディスパッチャ D_i は、双方向フィードバックによって収集した負荷情報とそのサーバの処理能力、さらに通信遅延を考慮して、各サーバへジョブを配送した場合の応答時間 $R_j (j = 1, 2, \dots, n)$ を予測し、それが最小となるサーバへジョブを配送する。 R が最小のサーバが複数ある場合、その中にローカルサーバが含まれていればローカルサーバへ、含まれていなければそのうちの任意のサーバへランダムに配送する。

3 情報の更新間隔

前章で述べた条件のもとでシミュレーションを行なった場合に、負荷情報の収集やジョブの配送が予想通りに行なわれているかどうかを調べるため、各ディスパッチャにおける情報の更新間隔を測定した。上述の配送方式では、各ディスパッチャに到着したジョブは、その時点でディスパッチャの持っている情報から予想応答時間が一番短いと判断されたサーバに配送され、ディスパッチャの情報は、サーバから戻ってきたジョブによって更新される。

更新間隔の測定結果を図 3 に示す。これは、ディ

スパッチャへのジョブの到着率: 0.6 のとき、ディスパッチャ 1 が保持する各リモートサーバ (ローカルサーバ 1 を除く) の負荷情報の更新間隔の平均を測定したものである。この図から、サーバ 8 の値が、他のサーバの値に比べて大きくなっているのがわかる。この原因を追求したところ、サーバ 8 の更新間隔が常にはではなく、頻度は少ないが極めて大きな更新間隔が存在するためであることがわかった。つまり、各情報は必ずしも同程度の間隔で更新されるとは限らず、しかも更新間隔の非常に大きいものが存在する。更新間隔の非常に大きいサーバとは、そのディスパッチャからはほとんど利用されていないことを意味する。そこで、これを解消するために、更新間隔がある程度大きくなった負荷情報は古いとみなし、別の方法を用いてこの情報を更新することとした。この際、1. 情報の更新間隔が大きいと判断する指標、2. その判断が成された場合の情報更新方法、3. 判断するルーチンと呼ぶタイミングという三点について考慮する必要がある。

3.1 判断する指標

これは、情報を古いとみなす基準であり、以下のものが考えられる。

- (1) 1 台のサーバの情報のみで判断する場合
ディスパッチャ D_i において、サーバ $S_j (j \neq i)$ の情報更新間隔が以下のものを越えた場合に、その情報は古いと判断する。
 - t 方式 (threshold[t])
あらかじめ定められたしきい値 t
 - ave 方式 (average[k])
(D_i における S_j の過去の平均更新間隔) の k 倍
- (2) 他のサーバの情報と比較して判断する場合
ディスパッチャ D_i において、サーバ $S_j (j \neq i)$ に関する負荷情報の最新更新時刻 (U_j) のうち、最も古いものから現在までの、各サーバについての情報更新回数を比較し、以下の条件を満たした場合にその情報は古いと判断する。
 - ua 方式 (under-average[k])
平均情報更新回数が k 回に達した場合、その平均に満たない情報
 - um 方式 (under-maximum[k])
最大情報更新回数が k 回に達した場合、それ以外の情報
 - m 方式 (minimum[k])
最大情報更新回数が k 回に達した場合、更新回数が最小の情報

		threshold	average	under average	under maximum	minimum
R方式	arrival	R-t-arr	R-ave-arr	R-ua-arr	R-um-arr	R-m-arr
	interval	R-t-int	R-ave-int	R-ua-int	R-um-int	R-m-int
A方式	arrival	A-t-arr	A-ave-arr	A-ua-arr	A-um-arr	A-m-arr
	interval	A-t-int	A-ave-int	A-ua-int	A-um-int	A-m-int
O方式	arrival	O-t-arr	O-ave-arr	O-ua-arr	O-um-arr	O-m-arr
	interval	O-t-int	O-ave-int	O-ua-int	O-um-int	O-m-int
D方式	arrival	D-t-arr	D-ave-arr	D-ua-arr	D-um-arr	D-m-arr
	interval	D-t-int	D-ave-int	D-ua-int	D-um-int	D-m-int

表 1: 方式の組み合わせ

3.2 情報更新方法

これは、3.1節の各基準に基づいて古いと判断された情報を更新する方法である。今回は、以下の方法を考察対象とした。

- R方式 (Reset information)
負荷情報を J_{ij} にリセットする方式。(ここで J_{ij} は、 D_i から S_j へ配送したがまだ返送されていないジョブ数である。)
- A方式 (Ask to the server)
配送ジョブとは別に、相手サーバに直接、負荷情報のみを尋ねる方式。(サーバが返送する際のオーバーヘッドは無視できるものとする。)
- O方式 (ask to Other server)
第三者のサーバに、ジョブ配送時に尋ねる方式。これは、 D_i から S_k へジョブを配送した場合、 D_k が持っている $S_j (j \neq k)$ の負荷情報をビギンバックするものである。
- D方式 (Decrease information)
古いと判断された負荷情報、つまり予想内容数を時間の経過にともなって減少させる方式。(但し、 J_{ij} 以下にはならないものとする)

3.3 タイミング

これは、3.1節の各判断を行なうルーチンと呼ぶタイミングであり、以下のものを考える。

- arr(arrival) ジョブ到着時
- int(interval[I]) 一定時間 I ごと

以上の組み合わせを表1にまとめて示す。なお、以降では各方式をこの略称で呼ぶ。

4 シミュレーション結果

第3.2節で提案した各方式の性能をシミュレーションを用いて評価する。システムモデルは、第2.1節で示したものをを用いる。簡単のため $\mu = 1$ 、 $n = 10$ とする。また、サーバに割り当てられたジョブがそのサーバごとに待ち行列を形成し、先着順に処理されるバッチ式と、一定時間 (q : クォンタム) だけサーバを占有して待ち行列の後ろに戻る時分割式の両方でシミュレーションを行った。時分割式でのクォンタムは $q = 0.02$ とする。

まず、同じ処理能力を持つノード群が通信遅延を無視できる回線につながれており、各ノードへのジョブの到着が均等であるシステムについて考察する。すなわち $d = 0$ 、すべての i について $C_i = 1$ 、 $\lambda_i = \lambda$ の場合を図4、図5に示す。次に、各ノード間の通信遅延が無視できない場合 ($d = 0.5$) の結果を図6、図7に示す。

4.1 方式ごとの評価

[R方式] この方法はリセット間隔に依存する。すなわち、 I が大きすぎると情報がなかなか更新されなくなる一方、小さすぎると頻繁にリセットされてしまうために負荷を実際よりも低く見積ってしまう。よって、システムの状態に応じて最適な I が存在すると予想される。仮定した条件のもとでは、低負荷・中負荷において $I = 1(\text{sec})$ のとき、高負荷において $I = 10(\text{sec})$ のときの応答時間が最適となった。リセットする間隔が小さいほど負荷を低く見積るのである、妥当な結果といえる。

[A方式] この方法では、ジョブ配送時に正しい値を得られるものほど、応答時間が改善される。このためタイミングとして“ジョブ到着時”を用いた場合に最適な結果を示す。通信遅延を無視できる状況では、この方式の結果は、ほぼ理想的といえる。しかし、通信遅延を無視できないシステムでは、情報が届くまでの遅延のために、ジョブの配送先決定時には、正しい情報ではなくなっている。したがって、応答時間も理想的とは言えない結果となる。

[O方式] 第三者のサーバへジョブを配送する際、直接ビギンバックで得られる処理サーバの負荷情報に加えて、処理サーバが持っている他のリモートサーバの負荷情報もビギンバックするという方法である。したがって、有効と考えられるタイミングは“ジョブ到着時”のみである。利点として、本来の目的であるネットワークトラフィックを増加させずに、他サーバの情報を

得ることが可能となる。しかし、そのリモートサーバが持っている情報は不確かなものであるので、応答時間が改善されるとは限らない。

〔D方式〕時間の経過にともなって負荷情報つまり予想内容数を減少させる。つまりタイミングとして“ジョブ到着時”を用いた場合にも、それまでの時間の経過を考慮して負荷情報の値を減少させる。このため、ジョブの配送先決定時に、この情報の更新が反映されるという理由から、定期的に行うよりも“ジョブ到着時”をタイミングとして用いた場合に応答時間が短くなることがかかる。

4.2 通信遅延を無視できる均質システム

各方式において最適なパラメータを用いた方式と比較する。この結果、通信遅延を無視できるシステムでは、A方式を用いた場合に応答時間が最も改善された。通信遅延の無視できるシステムで正しい値を得て配送を行うため、良い結果となるのは当然と言える。しかし、ネットワークトラフィックを増加させずに、他サーバの負荷情報を得るという本来の目的に反する。

A方式以外では、D方式を用いた場合の応答時間が最も短い結果となった。これは、ジョブ数を減少させることによってそのサーバの負荷を軽く予想することになるので、そのサーバにジョブが配送されやすくなり、新しく正しい負荷情報が得られるためと考えられる。また、リセットするのではなく、徐々に減少させることによって、実際の負荷により近い値を予想できると考えられる。

さらにバッチ式(図4)と時分割式(図5)を比較した場合、A方式の応答時間に関しては大きな違いが見られる。A方式は、バッチ式においてはほぼ理想的な応答時間を示すのに対し、時分割式においては、短い応答時間ではあるが理想的な値(AQ方式)とは言えない結果であった。これは、もとのLR-PB方式で負荷情報としてサーバの内容客数を用いているためと考えられる。例えば、内容客数“3”の場合を考える。バッチ式であれば、これは全く処理されていない2個のジョブと処理中のジョブが1個あることを意味する。しかし、時分割式ではサーバに到着しているジョブが一定時間ずつ交代で処理されるので、この3個のジョブはすべて処理中であり、実際の未処理量はわずかであるかもしれない。このように時分割式におけるA方式は、真の負荷よりも高い情報を持つ可能性が高い。このため、応答時間がバッチ式ほど改善されないと考えられる。

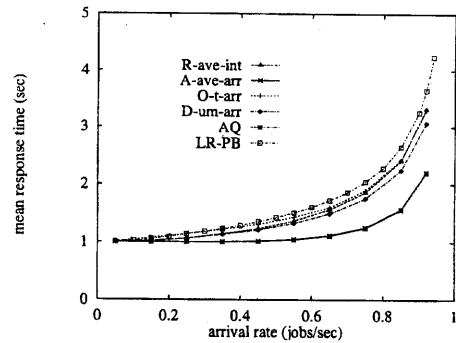


図 4: 遅延を無視できる場合 (バッチ式)

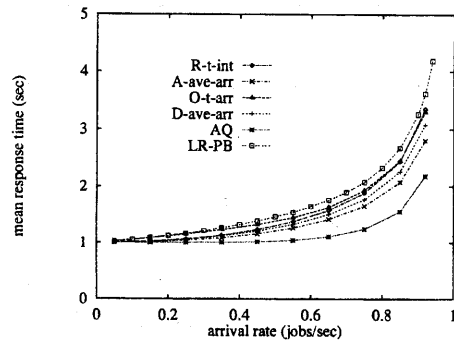


図 5: 遅延を無視できる場合 (時分割式)

4.3 通信遅延のある均質システム

通信遅延の無視できないシステム($d = 0.5$)では、R方式、D方式、A方式を用いた場合の応答時間が良い結果を示した。なかでも、中負荷ではA方式が、高負荷ではD方式がよい性能を示した。他の状況として遅延 $d = 0.3, 0.8$ の場合を評価した結果、遅延が大きくなるにしたがって、A方式の性能は悪くなり、D方式がよい性能を示すことが確認された。この理由は、以下のように考察できる。A方式は、古くなった情報の更新方法として通信を行うので、通信遅延の無視できないシステムでは、遅延が含まれてしまう。これによって、得られた情報が正しい情報ではなくなるため、性能が悪くなると考えられる。一方、D方式では通信遅延の影響を受けずに、古くなった情報を更新するので、その効果は失われない。

バッチ式と(図6)時分割式(図7)を比較した場合、どの方式に関しても大きな違いは見られなかった。

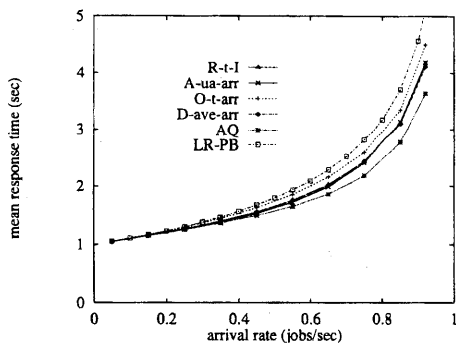


図 6: 遅延がある場合 (バッチ式)

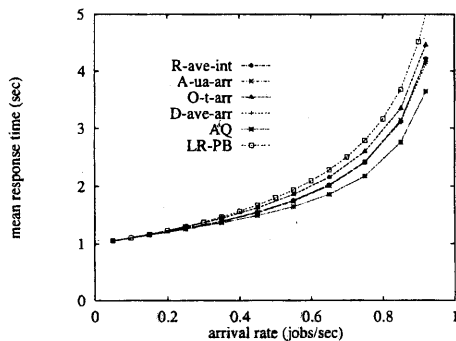


図 7: 遅延がある場合 (時分割式)

バッチ式における A 方式は、通信遅延がある場合には届いた情報が古くなっているため理想的な特性は示さない。

5 おわりに

LR-PB 方式ではそれぞれのディスパッチャに、負荷情報の更新間隔が非常に長くなるサーバが存在した。本稿では、これを改善するために、古くなった情報を別の方法で更新する方法について検討した。

古いと判断された情報をリセットする R 方式、古いと判断されたサーバの負荷情報を、その相手サーバと通信することで得る A 方式、第三者のサーバにジョブを配送する際に、古いと判断されたサーバの負荷情報もビジーバックさせる O 方式、古いと判断された負荷情報の値を、時間の経過にともなって減少させていく D 方式をシミュレーションを用いて検討した。

その結果、A 方式、O 方式、D 方式では、LR-

PB 方式と比較して応答時間が改善された。通信遅延の無視できるシステムでは、A 方式がほぼ理想的な応答時間を示し、遅延の無視できないシステムでは、D 方式が最適な応答時間を示した。O 方式では、パラメータの値によって、LR-PB 方式よりも応答時間が悪くなるので、適切なパラメータの値を選択しなければならない点が問題である。また、どの方式においても、システムの状態によって、判断の指標やタイミングの最適値は異なることがわかった。さらに各方式は、バッチ式、時分割式どちらにおいても同じ効果を挙げるため、応答時間の改善もほぼ等しいものとなった。

今後は、負荷を分散させることだけでなく、負荷分散を行うことによって生じる CPU オーバヘッドについても検討していく必要がある。なお、本研究は科学研究費 (08204205) の助成のもとで行われた。

参考文献

- [1] D.Ferrari, S.Zhou, "An empirical investigation of load indices for load balancing applications", Proc.of the 12th Int'l Symp. on Computer Performance Modeling, Measurement, and Evaluation, North Holland Publishers, Amsterdam, The Netherlands 1988, pp.515-528.
- [2] Gil-Haeng Lee, Heung-Kyu Lee, Jung-Wan Cho, "A sender-initiated adaptive load balancing scheme based on predictable state knowledge", IEICE Trans. inf.& syst., vol.E79-D, no.3, MARCH 1996.
- [3] 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).
- [4] 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).
- [5] 渡辺 尚, 太田 剛, 水野 忠則, 中西 暉, "双方向ビジ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 D-I, vol.J76-D-I no.3 pp.302-312(1995).
- [6] 染葉佳代子, 太田 剛, 渡辺 尚, 水野 忠則, "選択的多重ビジ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).