マルチメディア通信と分散処理　66－3
（１９９４．　７．　７）

双方向ピギーバックを用いたジョブ配送法について

染葉佳代子　　渡辺尚　　太田剛　　水野忠則

静岡大学

分散型ディスパッチャを用いる負荷分散方式として、著者らはLR-PB方式を提案してきた。LR-PB方式は、双方向ピギーバックを用いて環境観測を行い予想応答時間が最小であるサーバをジョブの配送先に決定する方式である。本稿では、LR-PB方式をランダム方式、変形サイクリック方式および可変閾値方式と比較し検討する。その結果、サーバの処理速度やジョブの到着率が均質なシステムでは変形サイクリック方式が有効であり、たとえ通信遅延が無視できない場合にでも、システムが不均質であればLR-PB方式は他の方式と比べて有効であることを示す。

A load balancing strategy with bidirectional piggybacking

Kayoko Someha　Takashi Watanabe
Tsuyoshi Ohta　Tadanori Mizuno
Faculty of engineering, Shizuoka University
3-5-1, Johoku, Hamamatsu, 432 Japan

The authors have proposed an adaptive load balancing strategy called LR-PB in a distributed computer system which consists of several pairs of a heterogeneous server and a dispatcher. In LR-PB policy, a dispatcher selects a server with least response time based on information acquired by bidirectional piggybacking. In this paper, the policy is evaluated against several policies ; random selection policy, revised cyclic selection policy and variable threshold policy. Simulated results show that the revised cyclic selection policy is effective in homogeneous system and that LR-PB policy is effective in a heterogeneous system, even if the environment includes relatively large network delay.

1 はじめに

個々に資源を有する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ついての主要な論点の１つは負荷分散とシステム性能との関係である。負荷分散を行う場合、システムに到着したジョブはディスパッチャによりサーバへ配送される。このディスパッチャは、配置形態により集中型と分散型に分類することができる。

集中型ディスパッチャでは、システムに投入された全てのジョブを一つのディスパッチャにより管理する。集中型ディスパッチャによる負荷分散については多くの研究がなされている。たとえば、次のジョブが到着するまでの間のシステムのスループットを最大にする方式 [2]、サーバの待ち行列長とサーバの処理速度の比が最小となるサーバへジョブを配送する方法 [3]、サーバの利用状況により閾値を変化させる方式 [4] などが提案されている。

分散型ディスパッチャでは、各計算機ごとにディスパッチャを配置し、それぞれに投入されたジョブを管理する。分散型ディスパッチャにおける負荷分散方式の論点は、1) 環境情報の収集方法、2) ジョブの受け渡し相手の決定方法である。分散型ディスパッチャにおける負荷分散方式には、ジョブを投入されたソース側が配送するサーバを決定するソースイニシアチブと、サーバ側がどのソースにあるジョブを引き受けるか決定するサーバイニシアチブの 2 つのタイプがある。ソースイニシアチブでは、最適なジョブの分配率を求めるアルゴリズム [14] や、環境情報を用いずサイクリックな配送を行う方式 [7]、モニターで管理している情報を参照して配送先の決定を行う方式 [10]、自分が閾値を越えたときに閾値を越えていないか各サーバに尋ねる方式などが研究されている [6] [12]。サーバイニシアチブでは、自分が閾値を越えていないときに各ソースに閾値を越えているかどうか尋ねる方式などがある [6][12]。また、ソース・サーバイニシアチブの両方を用いた方式も提案されている [6][12]。

本研究では以下のような理由から、分散型ディスパッチャを用い、ソースイニシアチブな配送について研究する。

- 本研究が最終的に対象とするシステムは、数 10 ～数 100 の計算機がインハウスネットワークまたは中・広域ネットワークによって接続されているシステムである。これらの資源とユーザが投入するジョブを一元管理することは、ネットワークトラフィック、ディスパッチャ管理の点から困難である。
- ジョブの特性はユーザによって大きく異なる。また、ユーザが使用する計算機群はネットワーク接続されている計算機群の部分集合であり、それらはユーザによって異なる。従って、ユーザの要求に柔軟に対処するためには、ユーザに依存したジョブの配送を行なわねばならない。さらに、ユーザに依存する部分はユーザに解放してカスタマイズを許すべきである。ジョブのクラスにより最適なサーバを予想する方法が [15] [13] で研究されている。

我々はこれまでに、双方向ピギーバックを用いて環境観測を行い予想応答時間が最小であるサーバをジョブの配送先に決定する LR-PB 方式を提案し、この方式が閾値による制御を必要としないことを示している [9] [16] [18]。本稿では、LR-PB 方式と多くの文献で検討されているランダム方式、サイクリック方式および集中型ディスパッチャで有効であったものを分散型ディスパッチャへ拡張した方式との比較、検討を行う。

2 モデル

対象とするシステムを図 1のようにモデル化する。

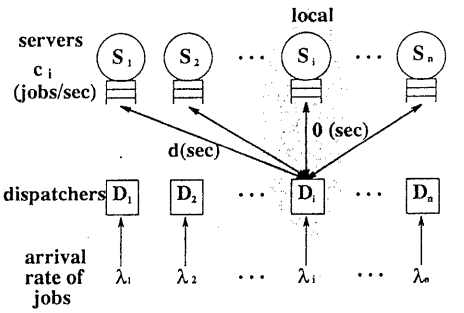


図 1: システムモデル

- システムは n 個のディスパッチャと n 個のサーバにより構成される。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。
- S_i の処理速度を c_i 命令／秒、$\vec{C} = (c_1, c_2, \ldots, c_n)$ とする。
- ディスパッチに要する時間は 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、$\vec{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ 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。
- サーバに割り当てられたジョブは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。

3 ジョブ配送アルゴリズム

本稿で提案し、検討する負荷分散方式について説明する。

3.1 双方向ピギーバックを用いる方式

各サーバの負荷状況を考慮してジョブの配送先を決定する場合、各ディスパッチャは各サーバの負荷状況を把握している必要がある。情報の収集方法としては、サーバがブロードキャストによって自分の情報を伝達するという方法が考えられる [5] [11]。しかし、これにはネットワークトラフィックが増大するという欠点がある。また、ジョブの特性を考慮した配送へ拡張することを考えると、実行して始めてわかるジョブ量などの情報が得られるようにしたい。そこで本研究では、以下に説明する双方向ピギーバックを用いる。

本研究では、直接ピギーバックと間接ピギーバックにより情報伝達を行う（図 2）。直接ピギーバックは、S_j が処理し終ったジョブの結果に自分の負荷情報を付けて

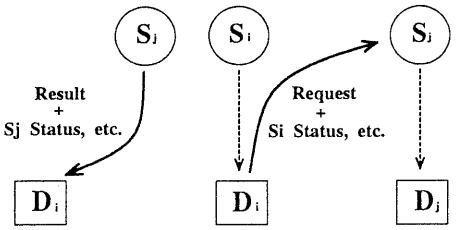
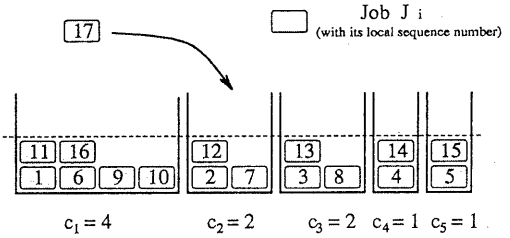


図 2: 直接ピギーバックと間接ピギーバック



$\vec{y} = (c_1, c_2, c_3, c_4, c_5, c_1 - 1, c_2 - 1, c_3 - 1, c_1 - 2, c_1 - 3)$
$= (4, 2, 2, 1, 1, 3, 1, 1, 2, 1)$

図 3: TCW 方式

D_i に返送する方法である。一方、間接ピギーバックは D_i がリモート S_j へジョブを配送する時、ローカル S_i の負荷情報をジョブに付けて配送し、S_j に届いた情報をそのローカルな D_j が得るという方法である。これらにより、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は更新される。

LR-PB 方式

我々が提案する LR-PB(least response time based on piggyback information) 方式は、双方向ピギーバックによって環境観測を行い、予想応答時間が最小のサーバをジョブの配送先に決定する方式である。

D_i にジョブが到着したら、D_i は環境観測で得た情報に基づいて、各サーバへジョブを配送した場合の予想応答時間 R_j、$j = 1, 2, ..., n$、を求め、それが最小のサーバをジョブの配送先に決定する。予想応答時間は次式で求められる。

$$R_j = \frac{(q_{ij} + 1) * \mu}{c_j} + 2 * d$$

ここで、q_{ij} は D_i がピギーバックによって得た情報に自分が配送しているジョブを考慮した S_j の待ち行列である。q_{ii} は S_i がローカルサーバであるため常に正確な値である。R が最小のサーバが複数ある場合は、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 R 中の任意のサーバへ配送する。

なお、LR-PB 方式の位置づけを付録に示す。

3.2 比較対象とする方式

3.2.1 ランダム方式

TR(τ) 方式 TR(threshold control and random selection) 方式は、閾値制御を行い、すべてのサーバの中から処理速度の比に相当する確率で配送先を決定する方式である。環境観測は行わない。

他の研究においては、閾値として待ち行列にならんでいるジョブの個数を用いているものが多い [7] [8]。本研究では応答時間を閾値に用いる。これは、サーバの処理速度の違いに対応するためである。ディスパッチャ D_i は、ローカルサーバ S_i へ配送した場合の予想応答時間 R_i と閾値 τ を比較し、$R_i \leq \tau$ であればローカルサーバへジョブを配送する。ここで、ディスパッチャはローカルサーバの負荷状況を瞬時に知ることができるので、S_i のその時の待ち行列長を q_i とすると、ここで R_i は次のようにして求められる。

$$R_i = \frac{(q_i + 1) * \mu}{c_i}$$

つまり、D_i はジョブが到着したら R_i を計算し、

$R_i \leq \tau$: ローカル S_i へ配送する。

$R_i > \tau$: 処理速度の比に相当する確率でサーバを選択し配送する。

閾値を τ とした場合、TR(τ) と表す。

3.2.2 変形サイクリック方式

TCW(τ) 方式 TCW(threshold control and cyclic selection weighted by service rate) 方式は、自分の履歴から環境を予測し、TR 方式と同様に閾値制御を行い、サーバの処理速度に合わせて重み付けされたサイクルに従ってジョブの配送先を決定する方式である。

D_i はジョブが到着したら、R_i を計算し、

$R_i \leq \tau$: ローカル S_i へ配送する。

$R_i > \tau$: あらかじめ定められたサイクルに従って配送する。
サイクルは次のように決定する。
$c_i \geq c_{i+1}$ であり、c_i は整数であると仮定する。
D_i は $\vec{x}$ を

$$\begin{aligned}\vec{x} &= (c_1, \ldots, c_n, c_1 - 1, c_2 - 1, \ldots, c_1 - k, \\ &\quad \ldots, c_n - k, \ldots, c_1 - c_1, \ldots, c_n - c_1)\end{aligned}$$

とする。ここで、$0 \leq k \leq c_1$ である。
$\vec{x}$ の要素から正であるものを取り出し $\vec{y}$ とする。

$$\begin{aligned}\vec{y} &= (y_1, \ldots, y_l) \\ &= (c_1, \ldots, c_n, \ldots, c_h - k, \ldots, 1)\end{aligned}$$

ここで、$l = \sum_i^n c_i$ である。ジョブが D_i でサイクルに従って配送される u 番目のジョブであれば、D_i は $y_{u\,mod\,l} = c_h - k$ である S_h を選ぶ。

図 3に例を示す。この場合、今到着したジョブが D_i に到着しサイクルにしたがって配送される 17 番目のジョブであれば、D_i はジョブを S_2 に配送する。D_i は自分が配送したジョブの履歴に従って配送し、他のディスパッチャの履歴は考慮しないことに注意する。

3.2.3 可変閾値方式

DDER 方式 本方式は、閾値を動的に変化させる方式である [17]。閾値の変更は、Shenker と Weinrib によって集中型ディスパッチャに対する方式として提案された D 方式 [4] を基にする。

DDER(distributed deterministic based on equivalent service rate) 方式は、ディスパッチャに待ち行列を形成し、各ディスパッチャが自分がどのサーバを利用しているかに基づいてジョブの配送先を決定する方法である。D_i は、ジョブが到着した時、自分が利用していないサーバの中で最も速いものが S_j であれば閾値 τ_{ij} を計算する。すなわち、到着したジョブを含めた D_i の待ち行列長が閾値 τ_{ij} 以上であれば S_j へ配送し、そうでなければ待ち行列に並べる。待ち行列に並べられたジョブは、ジョブを処理中である速いサーバが空くのを待つ。速いサーバが空いたら、待ち行列の先頭のジョブから空いたサーバへ配送される。

閾値は以下のように決定される。

$c_{open} \equiv$ (未使用の中で最も速いサーバの処理速度)$*\frac{1}{n}$ とし、c_{open} よりも速いサーバの数を N_{faster}、$c_{ave} \equiv$ (N_{faster}の平均処理速度) $* \frac{1}{n}$ とする。ここで、$\frac{1}{n}$ を掛けるのは、各ディスパッチャは他のディスパッチャが自分と同程度のジョブを配送していると仮定し、c_i を実際の $\frac{1}{n}$ であると考えるためである。

このとき、平均遅延 D_τ は、

$$D_\tau = \frac{N_\tau}{N_{faster}\rho c_{ave}} + \frac{Z_\tau}{c_{open}}$$

で表せる。ここで、N_τ は M/M/N_{faster}/k システムにおけるジョブ数の平均、Z_τ はブロック確率であり、$\rho = \frac{\lambda}{N_{faster}c_{ave}}$ である。呼損率を示すアーラン B 関数：$B(N_{faster}, N_{faster}\rho)$ を B で表すと、N_τ, Z_τ は、

$$N_\tau = \frac{N_{faster}\rho + BN_{faster}\rho(-1+\frac{1-\rho^\tau}{1-\rho}) + B\rho(\frac{1-(1+\tau)\rho^\tau+\tau\rho^{\tau+1}}{(1-\rho)^2})}{1+B\rho\frac{1-\rho^\tau}{1-\rho}}$$

$$Z_\tau = \frac{B\rho^\tau}{1+B\rho\frac{1-\rho^\tau}{1-\rho}}$$

となる。次に、差分関数 $\Delta(\tau) \equiv D_\tau - D_{\tau-1}$ を以下のように定義する。

$$\Delta(\tau) \propto \tau + N_{faster}(R-1)(\rho-1) + B\rho(N_{faster}(1-R) + \frac{\tau}{1-\rho} + \frac{\rho^\tau - 1}{(1-\rho)^2})$$

ここで、$R \equiv \frac{c_{ave}}{c_{open}}$、$\tau = \tau_{ij}$ である。$\Delta(\tau) < 0$ である最大の整数値 τ が最適閾値である。対象とするジョブを含めた待ち行列長が閾値以上であれば、利用していないサーバの中で最も速いサーバへジョブを配送する。

3.2.4 準理想的な方式

AQ 方式 AQ(actual queue-length) 方式では、ディスパッチャはすべての環境情報を瞬時に得ることができると仮定し、正確な情報を用いた予想応答時間が最小のサーバを配送先に決定する。

この方式は双方向ピギーバックで得られる環境情報の信頼性を検討するための比較対象とする。ただし、通信遅延を無視できない場合、D_i が S_j にジョブを配送しているとしてもそのジョブが S_j に到着するまでは他のディスパッチャはそれを知ることができないためジョブが集中し、AQ 方式は理想的であるとは言えない。

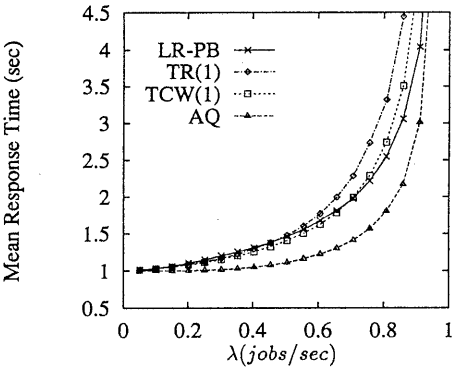


図 4: 均質網における各方式の比較

$\vec{C} = (1,1,1,1,1)$ $\vec{\lambda} = (\lambda,\lambda,\lambda,\lambda,\lambda)$ $d = 0$

4 シミュレーション結果

シミュレーションは、次のようなシステムを仮定して行なう。

1. 全てのサーバの処理能力が等しく、全てのディスパッチャへのジョブの到着率が等しいシステム。
2. 全てのサーバの処理能力は等しく、ディスパッチャへのジョブの到着率は異なるシステム。
3. サーバの処理能力は異なり、ディスパッチャへのジョブの到着率は等しいシステム。
4. 通信遅延を無視できないシステム。

ここでは、$\mu = 1$ とする。

各配送方式の特性を、ディスパッチャへのジョブ到着率と平均応答時間の関係を示すグラフで表し、各仮定ごとに検討する。TCW 方式、TR 方式に関しては、最も特性の良い閾値のもののみ示す。

4.1 処理能力・到着率ともに均質な場合

処理速度が 1(jobs/sec) のサーバが 5 台とディスパッチャ 5 台が通信遅延を無視できる回線でつながれているシステムで、各ディスパッチャにはジョブが均等に到着する場合に対するシミュレーションを行った。図 4 に、シミュレーション結果を示す。低負荷においては各方式に大きな差はないが、低・中負荷では TCW(1) が良く、高負荷になると LR-PB が良いといえる。

4.2 到着率が不均質な場合

各ディスパッチャへのジョブ到着率が異なる場合を考える。ここで $D_2 \sim D_5$ のジョブ到着率 λ_2 は 0.2(jobs/sec) に固定し、D_1 の到着率 λ_1 を変化させて特性を調べた。D_1 へ到着するジョブの平均応答時間を平均応答時間 1、$D_2 \sim D_5$ のそれを平均応答時間 2 とする。各方式を比較すると、図 5 に示すように、平均応答時間 1 は中・高負荷において LR-PB が良い。LR-PB は高負荷になるに従って AQ に近づき双方向ピギーバックによる情報の信頼性が高くなるといえる。これは、λ_1 が λ_2 より大きくなると、D_1 がリモートサーバに頻繁にジョブを配送し情報の更新回数が増すのに加えて、$S_2 \sim S_5$ の状態があ

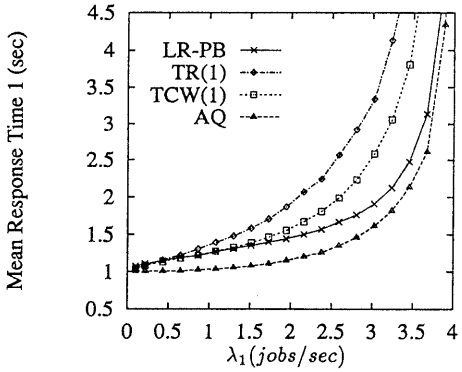


図 5: ジョブ到着率が不均質な場合での各方式の平均応答時間 1 の比較
$\vec{C} = (1,1,1,1,1)$ $\vec{\lambda} = (\lambda_1, 0.2, 0.2, 0.2, 0.2)$ $d = 0$

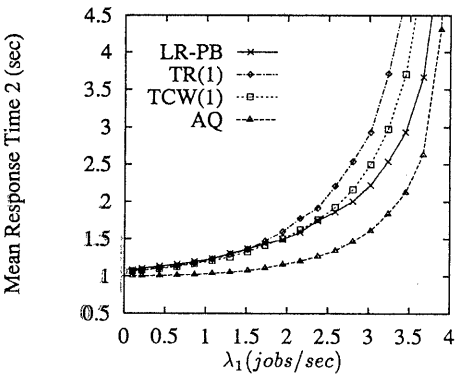


図 6: ジョブ到着率が不均質な場合での各方式の平均応答時間 2 の比較
$\vec{C} = (1,1,1,1,1)$ $\vec{\lambda} = (\lambda_1, 0.2, 0.2, 0.2, 0.2)$ $d = 0$

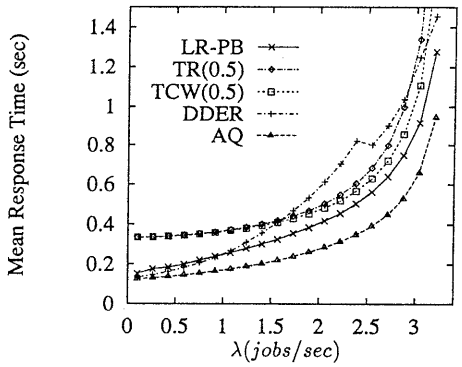


図 7: サーバが不均質なシステムにおける各方式の比較
$\vec{C} = (1,1,2,2,2,2,4,4,8,8)$ $\vec{\lambda} = (\lambda, \lambda, \lambda, \lambda, \lambda)$ $d = 0$

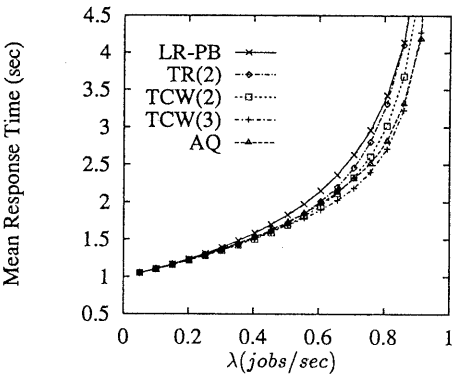


図 8: 通信遅延のある均質網における各方式の比較
$\vec{C} = (1,1,1,1,1)$ $\vec{\lambda} = (\lambda, \lambda, \lambda, \lambda, \lambda)$ $d = 0.5$

まり変化しないためである。また図 6 に示すように、平均応答時間 2 は低・中負荷においては各方式ともほぼ同じであるか若干 TCW(1) が良いが、高負荷になると LR-PB が良い。

4.3 処理能力が不均質な場合

サーバの処理速度が均質でない場合について検討する。ここでは、10 台のサーバの処理速度が $\vec{C} = (1,1,2,2,2,2,4,4,8,8)$ であるシステムを考える。通信遅延は無視できるものとし、ディスパッチャへのジョブの到着率は全て同じとする。各方式を比較すると、図 7 に示すように、低負荷においては DDER が良いが、ほとんどの領域において LR-PB が良い。

4.4 通信遅延が無視できない場合

通信遅延が無視できない場合について検討する。ここでは $d = 0.5$ とする。

均質なシステム 各方式を比較すると、図 8 に示すように、低負荷においてはほとんど差はないが、TCW(2)、TCW(3) が良いといえる。LR-PB は、D_i が予想応答時間が最小の S_j にジョブを配送しているとしても、そのジョブが S_j に到着するまでは他のディスパッチャはそれを知ることができない。この仮定においては通信遅延がかかるため、他のサーバも S_j にジョブを配送してしまう。そのため S_j にジョブが集中する結果となる。また、ビギーバックによる情報が通信遅延により古いものとなってしまうため、さらに性能が悪くなる。

処理能力が不均質なシステム 不均質なシステムにおいても均質なシステムと同様に LR-PB の性能は悪くなるが、TCW 方式も速いサーバが通信遅延のかかるリモートであっても多くのジョブを配送するため性能が悪い。LR-PB 方式は、ビギーバックによる情報の信頼性が落ちるが、配送先を決定するときに通信遅延を考慮することが有効に働き、図 9 に示すように、各方式を比較するとほぼ全域において LR-PB が良い。

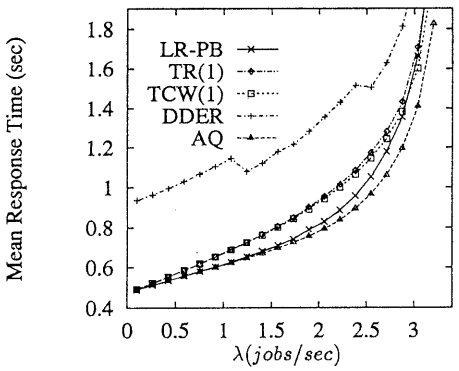


図 9: 通信遅延のある不均質なシステムにおける各方式の比較
$\vec{C} = (1,1,2,2,2,2,4,4,8,8)$ $\vec{\lambda} = (\lambda,\lambda,\lambda,\lambda,\lambda)$
$d = 0.5$

5 結論

本研究では、分散計算機システムにおいて、資源の有効利用と応答特性の改善のために必要な負荷分散の方式について検討した。

我々は双方向ビギーバックを用いて環境観測を行い、その情報に基づいた予想応答時間が最小のサーバへジョブを配送する LR-PB 方式をすでに提案している。この LR-PB 方式を、重み付けをおこなってサイクリックな配送を行なう TCW 方式、ランダムな配送を行なう TR 方式、集中型ディスパッチャで有効な D 方式を分散型へ拡張した DDER 方式とシミュレーションを用いて比較、検討した。その結果、均質網では TCW 方式が有効であり、双方向ビギーバック情報がはずれる可能性のある LR-PB 方式は TCW 方式ほど特性が良くなかった。しかし、サーバの処理速度とジョブの到着率がアンバランスなほど LR-PB 方式は有効であることがわかった。これは、通信遅延がある場合にも成立する。

LR-PB 方式の改良点として、ビギーバックによる情報の信頼性の向上があげられる。低負荷のときの情報、また、ジョブをあまり配送しないサーバの情報はあまり更新されず正しい情報を把握しにくい。通信遅延が大きい場合にも、情報が古くなってしまう。そこで情報の更新回数を多くするために、自分の情報あるいはローカルサーバの情報だけでなく、ビギーバックによって得た他のサーバの情報もビギーバックさせる方法が考えられる。しかしこれを行うには、ビギーバックさせる情報の増加によるネットワークトラフィックへの影響について検討しなければならない。

また本稿では、ある程度大きなジョブを前提としているためディスパッチ時間を 0 秒とした。しかし、プロセスマイグレーションのように小さなジョブに対してこの仮定は成立しない。今後は、ディスパッチに要する時間の影響の考察が必要であると思われる。

参考文献

[1] Yung-Terng Wang and Robert J.T.Morris, "Load Sharing in Distributed Systems," IEEE Trans.on Comput., vol.C-34, No.3, 1985.

[2] Yuan-Chieh Chow and Walter H.Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," IEEE Trans.on Comput., vol.28, No.5, 1979.

[3] S.A.Banawan and J.Zahorjan, "Load Sharing in Heterogeneous Queueing Systems," IEEE INFOCOM '89. 1989.

[4] Scott Shenker and Abel Weinrib, "The Optimal Control of Heterogeneous Queueing Systems:A Paradigm for Load-Sharing and Routing," IEEE Trans.on Comput., vol.38, No.12, 1989.

[5] Katherine M.Baumgartner and Benjamin W.Wah, "GAMMON:A Load Balancing Strategy for Local Computer Systems with Multiaccess Networks," IEEE Trans.on Comput., vol.38, No.8, 1989.

[6] Ravi Mirchandaney,don Towsley and John A.Stankovic, "Analysis of the Effects of Delays on Load Sharing," IEEE Trans.on Comput., vol.38, No.11, 1989.

[7] 中西暉, 井上健, 手塚慶一, "複数待ち行列システムにおける負荷分散について," 京大数理解析研究所講究録 452, 1982.

[8] 中西暉, 加茂博史, 真田英彦, 手塚慶一, "分散情報処理網におけるジョブ配送について," 京大数理解析研究所講究録 519, 1984.

[9] 渡辺尚, 太田剛, 西藤浩二, "ビギーバック情報に基づいた動的負荷分散方式の考察," 信学技報 SSE92-129.

[10] John A.Stankovic, "An Application of Bayesian Decision Theory to Decentralized Control of Job Scheduling," IEEE Trans.on Comput., Vol.C-34, No.2, 1985.

[11] Benjamin W.Wah and Jie-Yong Juang, "Resource Scheduling for Local Computer Systems with a Multiaccess Network," IEEE Trans.on Comput., Vol.C-34, No.12, 1985.

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

[13] Kumar K.Goswami, Murthy Devarakonda, and Ravishankar K. Iyer, "Prediction-Based Dynamic Load-Sharing Heuristics," IEEE Tr.on SMC, vol.4, No.6, 1993.

[14] Asser N.Tantawi and Don Towsley, "Optimal Static Load Balancing in Distributed Computer Systems," JACM., Vol.32, No.2., 1985.

[15] Murthy V.Devarakonda and Ravishankar K.Iyer, "Predictability of Process Resource Usage: A Measurement-Based Study on UNIX," IEEE Trans.on Software Engineer., Vol.15, No.12, 1989.

[16] T. Watanabe,T.Ohta and T.Mizuno, "Adaptive Load Balancing with Distributed Dispatchers using Piggybacked information," International Joint Workshop on Computer Communication, 1993 12.

[17] 染葉佳代子, 渡辺尚, 太田剛, 水野忠則, "分散型ディスパッチャを用いたジョブ配送法について," 情報処理学会第 48 回全国大会 7D-3, 1994.

[18] 渡辺尚, 太田剛, 水野忠則, "双方向ビギーバックを用いた動的負荷分散方法について," 電子情報通信学会春季大会 B-678,1994.

付録 LR-PB 方式の位置づけ

Y.T.Wang と R.J.T.Morris は、サーバイニシアチブとソースイニシアチブおよび配送先、配送元を決定する際に利用する情報のレベルにより負荷分散アルゴリズムを 14 に分類した [1]。この分類法に従うと LR-PB は、ソースイニシアチブであり、サーバの待ち行列長までのレベルの情報に依存するレベル 5 に属する。

また N.G.Shivaratri らは、動的な負荷分散アルゴリズムを 4 つの要素から分類した [12]。LR-PB 方式をこれに当てはめるとつぎのようになる。ノードがジョブを他のノードに送る側であるか他のノードから受け取る側であるかを決定する transfer policy は relative transfer policy である。どのジョブを移動するかを決定する selection policy はノードを送る側にする新たに発生したジョブに決定する。どのノードにジョブを送るかを決定する location policy はビギーバック情報に基づいた予想応答時間が最小のノードを選択する方法である。彼らの論文ではブロードキャストが登場しているが、ビギーバックにはふれられていない。どのようにシステム内の他のノードの情報を収集するかを決定する information policy は送る側、受け取る側になったときに情報を収集する demand-driven policy である。