システムソフトウェアと
オペレーティング・システム 69−17
（１９９５．６．２３）

分散システムにおける プロセスバッファを用いた動的負荷分散方式

鈴木 伸夫†, 山本 幹††, 岡田 博美††, 池田 博昌††

† 三菱電機マイコン機器ソフトウエア(株)

〒661 兵庫県尼崎市猪名寺 2-5-1

†† 大阪大学工学部通信工学科

〒565 大阪府吹田市山田丘 2-1

あらまし　分散システムにおける動的負荷分散方式は、現実的な問題から未実行プロセスを移動対象とし、プロセス投入時点で移動するか否かを決定することが一般的である。本稿では、計算機の処理能力を考慮し、さらにプロセスの移動時点をプロセス投入時点に制限するという制約を除くことにより、柔軟な負荷分散を可能とする方式としてプロセスバッファを用いた動的負荷分散方式を提案する。本方式では、まず処理能力に応じて計算機で実行するプロセス数を決定し、それ以外のプロセスを仮想的に設けたバッファに収容する。バッファ内のプロセスは、プロセス移動の対象となる。このように処理速度に比例して実行プロセス数を決定することで、各プロセスに均等にＣＰＵ資源を分配する。また、バッファ内のプロセスを移動対象とすることで、システム内の負荷の不均衡に柔軟に対応できる。

和文キーワード　動的負荷分散方式, プロセスバッファ, 分散システム

Dynamic Load Balancing Method using Process Buffer Scheme for Distributed System

Nobuo Suzuki†, Miki Yamamoto††, Hiromi Okada††, Hiromasa Ikeda††

† Mitsubishi Electric Micro-Computer Application Software Co.,Ltd.

2-5-1, Inadera, Amagasaki, Hyougo 661, Japan

†† Faculty of Engineering, Osaka University

2-1, Yamadaoka, Suita, Osaka 565, Japan

Abstract　In this paper, we propose a dynamic load balancing method using process buffer for distributed system. The method proposed here takes into account processing capability of each processor and remove a restriction that process can be migrated at the process input epoch. Our method determine the maximum number of processes which can be executed in each processor in proportion to its computation power. When the number of processes executed is equal to this maximum number, arrival processes are stored in a process buffer which is virtually settled. Processes in process buffer are candidates for process migration. As a result, CPU resourse are evenly shared, and it flexibly balances load in a system.

英文 Key Words　Dynamic Load Blancing, Process Buffer, Distributed System

1 まえがき

近年高度情報化社会の発展に伴い、種々の計算機をネットワークにより接続した分散システムが一般的となっている。しかし、各計算機間で負荷の不均衡が生じ、計算機能力を十分に活用しているとは言い難い。このような状況を改善する方法として、負荷分散方式の研究が数多くなされてきた。負荷分散方式は、大きく静的負荷分散方式 [1][2] と動的負荷分散方式 [3][4][5] の二つに分けることができる。静的負荷分散方式は実行時の負荷の動的変動が考慮されていないため、性能は動的負荷分散方式より劣る事が多い。

以上のような理由から、動的負荷分散方式が多数研究されている。これらの負荷分散方式では、プロセスの投入段階で負荷評価を行いプロセス移動が行われる。プロセスの投入時点は、ユーザが決定するパラメータであり、計算機の負荷状況の変化とは全く独立である。従って、プロセスの投入時という時間軸上の一点における負荷がたとえ重くても、長期的にみればその計算機の負荷は統計的に軽い場合がある。そのため、プロセスの投入時点でプロセスの移動先を決定することは必ずしも最適と考えられない。

そこで本稿では、上記の問題点を解決するためにプロセスバッファを用いた動的負荷分散方式を提案し、その有効性を示す。まず、処理能力に比例して計算機で実行するプロセス数を決定し、それ以外のプロセスをバッファに収容する。そして、バッファ内のプロセスを負荷評価およびプロセス移動の対象とし、周期的にプロセス移動を行う。このようにプロセス移動に猶予を持たせることで、プロセス移動の機会を増やすことができ、統計的な計算機負荷を考慮した負荷分散を行うことが可能となる。以下、2 章ではシステムや本稿で用いられる用語の説明を行い、3 章では提案する動的負荷分散方式について説明を行う。そして、4 章ではその性能評価を行い、最後に簡単なまとめを述べる。

2 動的負荷分散アルゴリズム

Fig.1に、本方式で想定するシステム構成を示す。本方式では、複数の計算機 (以下、ノード) が LAN (Local Area Network) により接続された分散システムを想定する。処理能力は各ノードごとに異なり、これらのノードがバス型 LAN により接続されているものとする。ノードの処理能力は、さまざまな尺度ではかることができるが、本稿では MIPS(Million Instruction Per Second) 値を用いる。

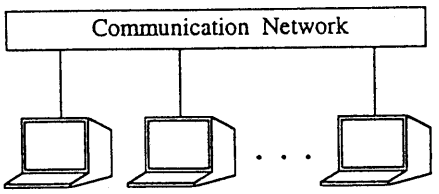

Fig. 1 システム構成

本稿で提案する動的負荷分散方式では、各ノードの負荷の均衡化とプロセス移動の機会を増やすために、ノードである一定数以上のプロセスが実行中の場合、到着したプロセスを待機させるプロセスバッファを設ける。プロセスが自ノードユーザからの発生プロセスであるか、他ノードからの移動プロセスであるかによって投入プロセスバッファと移動プロセスバッファの 2 種類のプロセスバッファを用意する。

以下、2.1 節では負荷分散アルゴリズムの説明を行い、2.2 節では最大実行プロセス数、投入プロセスバッファ、移動プロセスバッファの説明を行う。そして、2.3 節でノードの負荷評価に関して説明を行う。

2.1 負荷分散アルゴリズム

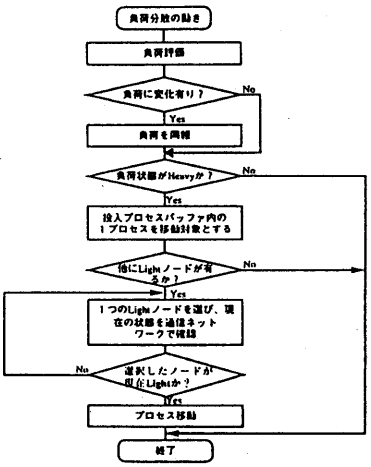

Fig. 2 負荷分散アルゴリズム

プロセスの実行および移動を決定する際のアルゴリズムを Fig.2に示す。

ユーザがノードにプロセスを投入すると、一旦投入

プロセスバッファに投入される。ノードは2.3節で述べる方法により周期的に負荷評価を行い、その結果負荷が変化していれば負荷メッセージを同報通信する。同報通信により得られた負荷は、各ノードで負荷情報テーブルとして保持する。自負荷がHeavyの時、負荷情報テーブルよりLightのノードを探し出す。Lightのノードが複数ある場合、その中よりランダムに選択する。Heavyのノードは Lightのノードに対しプロセス処理要求を送る。ノードはプロセス処理要求を受け取ると、依然として Lightであればプロセス移動許可を返す。プロセス処理要求を出したノードがプロセス移動許可を受け取ると投入プロセスバッファより先頭プロセスを取り出し移動する。移動されたプロセスは、移動プロセスバッファに収容される。また、ノードで実行されているプロセス数が最大実行プロセス数以下であれば、2種類のプロセスバッファからプロセスが取り出され実行される。

2.2 プロセスバッファ

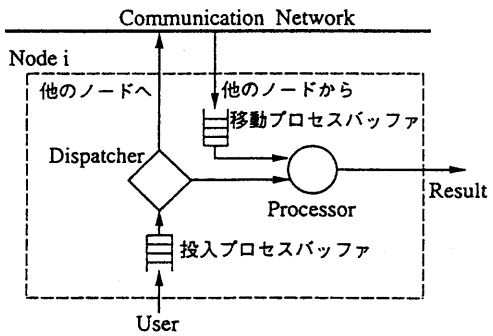

Fig. 3 ノードモデル

Fig.3に、ノードモデルを示す。プロセスバッファ内のプロセスは未実行プロセスである。ただし、バッファ内のプロセスは、今後該当ノードで実行される可能性のあるプロセスとして考え、そのノードの負荷評価の対象に加える。

最大実行プロセス数

本研究では、ノードの処理能力を考慮した負荷均衡を目的として、各ノードで実行するプロセス数をノードの処理能力 (CP_i:Computing Power) に比例して決定する。そのために、しきい値 Cを用いる。

ノード iにおける最大実行プロセス数 ($MNPC_i$: Maxmum Number of Processes in Computer) は、

$$MNPC_i = CP_i \times C$$

で求められる。この最大実行プロセス数を超えたプロセスがノードに入って来た場合、2種類のプロセスバッファに収容する。以下に、それぞれのプロセスバッファに関する説明を行う。

投入プロセスバッファ(Input Process Buffer)

ユーザからノード iにプロセスが投入されたとき、ノード i内に最大実行プロセス数以上のプロセスが存在すれば、投入プロセスバッファに収容する。それ以外はノード iで実行される。具体的には、

$$NEP_i \geq MNPC_i$$

のとき、プロセスは投入プロセスバッファに収容される。また、

$$NEP_i < MNPC_i$$

のとき、投入プロセスはノード iで実行される。

NEP_i : 実行プロセス数
(Number of Executed Processes)

移動プロセスバッファ(Migrated Process Buffer)

移動プロセスバッファは、他ノードより移動されたプロセスを収容する。このプロセスバッファ内のプロセスは再び移動対象とはしない。

ノードでのプロセスの実行に際して、各プロセスバッファはFIFO規律に従う。もし、両バッファ双方にプロセスが収容されている場合は、移動プロセスバッファ内のプロセスが優先的に実行される。

2.3 ノードの負荷

ノードの負荷を決定するために、さまざまな尺度が用いられている。しかし、実際にプロセスの要求仕事量を知ることができないため、プロセス数を用いる方法が一般的となっている。しかし、プロセス数を用いた場合、プロセス移動が、一つのノードに集中するなどの弊害が生じる場合がある。

本研究においてもノードの負荷を決定するために、プロセス数を用いる。そして、冗長性をもたせるために3種類の負荷状態に分類する [1]。このようにすることで、低負荷のノードにプロセス移動が集中することを防ぐ。

本研究では、未実行のプロセスもノードの負荷として換算する。つまり、一定間隔ごとにノード内のプロセス数とプロセスバッファ内のプロセス数を用いて負

荷の評価を行う。その後、その負荷をランク付けする。

$$WorkLoad_i = \frac{NPIB_i + NPMB_i + NEP_i}{CP_i}$$

$$LoadState_i = \begin{cases} Light \\ (WorkLoad_i \leq LightLoad) \\ Normal \\ (LightLoad < WorkLoad_i < NormalLoad) \\ Heavy \\ (NormalLoad \leq WorkLoad_i) \end{cases}$$

$NPIB_i$: 投入プロセスバッファ内プロセス数
(Number of Processes in Input Process Buffer)

$NPMB_i$: 移動プロセスバッファ内プロセス数
(Number of Processes in Migrated process Buffer)

このように得られた $LoadState_i$を同報通信を用いて他ノードに知らせる。ただし、$LoadState_i$が変化しなければ同報通信を行わない。$LoadState_i$を離散状態表現し、これが変化した時のみ通知するという手法を用いる事により、通信量を減少させる事ができる。また、他のノードからの負荷情報は、各ノードで負荷情報テーブルに保持する。

3 性能評価

3.1 シミュレーションモデルおよび条件

本稿では、2章で提案した負荷分散アルゴリズムに対して、シミュレーションを用いた性能評価を行う。シミュレーションにおいては、以下の仮定を設ける。

1. 処理能力の異なる3台のノード($node_0, node_1, node_2$)がネットワークで接続されており、それぞれの処理能力$MIPS_i$は30,40,50[MIPS]とする。
2. ノード間の伝搬遅延はノードによらず10[μsec]とする。
3. プロセスの処理要求量とファイルサイズは、それぞれ平均5.0×10^7[instructions],200[Kbytes]の指数分布に従うものとする。
4. プロセスは$node_i$に平均到着率λ_iでポアソン到着する。
5. ノードのオペレーティングシステムのスケジューリング方式はラウンドロビンとする。
6. 負荷の評価に用いられるしきい値 $LightLoad$, $NormalLoad$は、それぞれ0.1,0.2とする。

プロセスを移動する場合、ネットワーク遅延が性能に大きな影響を及ぼす。遅延が大きい場合には、あるノードが出した負荷情報メッセージのもつ情報が、現在の状態と大きく異なることがある。また、ネットワークを介してプロセスをメモリにロードする場合にも応答時間の増大という形で影響が現れる。このように、負荷分散におけるネットワーク遅延の影響は大きい。そこで、本稿ではネットワーク負荷によるネットワーク遅延の増減を考慮した性能評価を行う。

本稿ではネットワークとして現在もっとも一般的に用いられているLANであるEthernetを仮定している。そこで、Ethernetのアクセス方式であるCSMA/CDの解析式[6]を用いる。具体的には、ネットワーク負荷を観測し、このネットワーク負荷による遅延を[6]の解析式から求め、その値をネットワーク遅延として用いる。

シミュレーションでは、2台のノード($node_1, node_2$)の平均プロセス到着率を0.3に固定し、$node_0$の平均プロセス到着率を変化させる。そして、$node_0$の平均応答時間を測定する。

3.2 比較方式

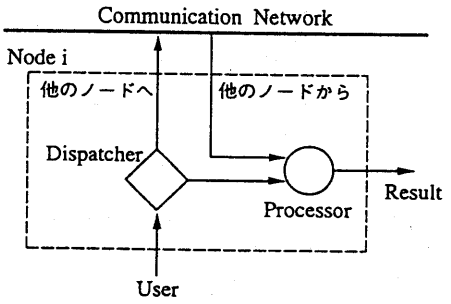

Fig. 4 比較方式のノードモデル

本稿では、プロセスバッファを用いない場合との比較を行う。Fig.4に比較方式のシミュレーションモデルを示す。比較方式は、プロセスが投入されると自ノードの負荷評価を行う。その結果、負荷がHeavyであれば他ノードに対してプロセス移動を行い、負荷がLightであれば自ノードでプロセスを実行する。その他の仮定は、提案アルゴリズムと同様である。

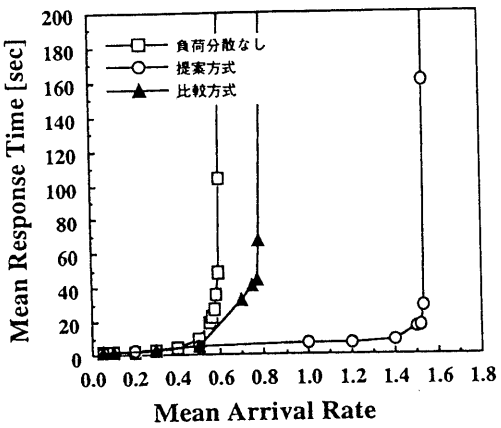

Fig. 5 平均応答時間の比較

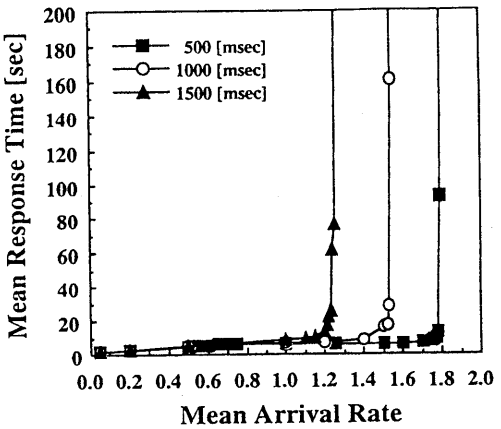

Fig. 6 同報通信間隔に対する平均応答時間

3.3 シミュレーション結果・考察

前節の条件下でシミュレーションを行い、評価・考察を行った。

Fig.5に、負荷分散を行わない場合、プロセスバッファを用いない負荷分散方式の場合と提案方式を用いた場合の平均応答時間を示す。同報通信間隔は、1000[$msec$]である。提案方式はいずれの平均到着率においても優れていることが分かる。また、比較方式と提案方式との間には、かなりの性能差が見られる。この理由は以下のように考えられる。

比較方式では、プロセスがノードに投入されると、その時点の負荷を評価する。そして負荷が $Heavy$の場合、$Light$ ノードに対してプロセスを移動するが、$Light$ ノードがなければ、自ノードで実行する。いったん実行したプロセスは移動できないため、このプロセスの実行後にあるノードが $Heavy$から $Light$ に状態遷移してもプロセスを移動し、負荷を均衡化することが出来ない。

それに対し、提案方式ではプロセスはプロセスバッファに収容されるため、$Light$ ノードが現れた時点で再びプロセス移動を試みることが出来る。このように提案方式は、プロセス到着時点以外の複数の時点でプロセス移動を行うことができるため、より柔軟な負荷分散が可能となる。

Fig.6に同報通信間隔を変化させた場合の提案方式の平均応答時間特性を示す。図に示されるように、同報通信間隔を小さくするほど性能が向上している。これは、同報通信を行うときに、負荷評価とプロセス移動を行うためである。つまり、同報通信間隔が短くなるほどプロセス移動の機会が増えるために、このような結果となっている。しかし、同報通信間隔が短くなるほどネットワーク負荷が高くなる。そのため、同報通信間隔とネットワーク負荷とのトレードオフを考慮する必要がある。

Fig.7にしきい値 Cを変化させた場合の特性を示す。同報通信間隔は 1000[$msec$] である。Cは最大実行プロセス数を決定するしきい値であり、Cが小さいほど最大実行プロセス数は少なくなる。この図で示されるように、Cが小さくなるほど平均応答時間は小さくなっている。Cが小さくなることで、ノード内で同時に実行されるプロセス数が少なくなる。その結果、平均応答時間が小さくなるものと考えられる。しかし、$C = 0.1, 0.2$ のとき、その平均応答時間は、ほぼ同一である。これは、Cが小さくなることで平均応答時間は短くなるが、その分プロセスバッファ内での待ち時間が長くなるためと思われる。

また、Fig.7において平均応答時間がいったん極値をもち、その後再び減少している。これは、プロセス平均到着率が 0.75 までのときは、プロセス移動を行うよりも、自ノード内のプロセスバッファにプロセスが収容される傾向があるためである。平均到着率が 0.75 を超えるとプロセス移動が増え、平均応答時間が小さくなるものと思われる。

Fig.8は、$C = 0.1$ のときのノードで実行されるプロセス数である。この図で上述の仮説を証明することが

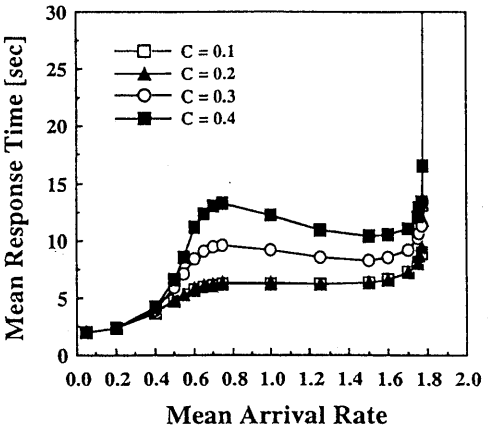

Fig. 7 しきい値 Cに対する平均応答時間

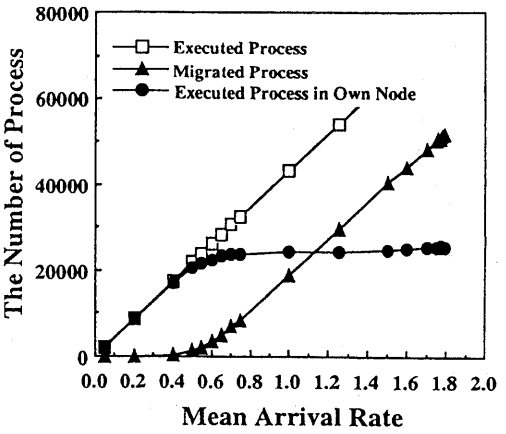

Fig. 8 実行プロセス数の割合

できる。到着率が0.75まで、自ノードで実行されるプロセス数は増えており、そのため平均応答時間は大きくなる。しかし、その前後から移動するプロセス数が増えるため平均応答時間は小さくなる。また、この図から負荷が重くなると自ノードで実行するプロセス数を一定に保ち、それ以外のプロセスを効率的に移動していることが確認できる。

4 まとめ

本稿では、処理能力が不均一な分散システムにおいて、ノードの処理能力に応じたノード内実行プロセス数の最大値を設定し、これを超えたプロセスを仮想的に設けたバッファに収容する方式を提案した。

シミュレーションの結果、提案方式は従来の負荷分散方式と比べ、高負荷まで性能を維持することがわかった。これより、プロセスバッファを用いることで、より柔軟なプロセス移動が可能となり、各計算機の処理能力を十分に利用できることを確認した。今後の課題としては、本方式を現実の分散システムに実装し、性能評価を行うことである。

参考文献

[1] K. Efe, "Heuristic models of task assignment scheduling in distributed systems", *IEEE Computer*,pp. 50–56, June 1982.

[2] V. M. Lo, "Heuristic Algorithms for Task Assignment in Distributed systems", *IEEE Trans. on Comput.*, vol. 37, No. 11, Nov. 1988.

[3] Tony T. Y. Suen, Johnny S. K. Wong, "Efficient Task Migration Algorithm for Distributed Systems", *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, No. 4, July 1992.

[4] 渡辺 尚, 太田 剛, 西藤 浩二, "ピ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, IN92-94, OFS92-37, 1993.

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

[6] W.Bux, "Local-Area Subnetwork:A Performance Comparison", *IEEE Trans. of Comm.*,vol. COM-29, No. 10, Oct. 1981.