ジョブ特性を考慮した動的負荷分散の実装について

7D−2

城谷 貴志　渡辺 尚　太田 剛　水野 忠則

静岡大学　工学部

1　はじめに

コンピュータネットワークは技術的にここ数年、飛躍的に進化している。しかし、個々のコンピュータの処理能力の向上が、ユーザの要求する資源や情報に対して十分な応答性能を与えているとは言いがたい。また、ネットワークで接続されたコンピュータ群を1つのシステムとして見ると、ローカルシステムごとの負荷に不均衡が生じている。これは、ジョブの特性を考慮して、動的に負荷を分散をする手法が十分に確立されていないことが原因である。

本研究では、種々のコンピュータが混在する環境において、ネットワーク上の各コンピュータで刻々と変わってゆく環境情報 (負荷情報とジョブの特性) に基づき、動的に負荷を分散することで、個々のコンピュータのCPUパワーを有効に利用する環境を提供することを目的とする。特に、ジョブの特性を考慮した配送法をTCPを用いて実装する。

2　実現する環境

実装にあたって、2つのデーモン (ディスパッチデーモン,CPUサービスデーモン) と3つのデータベース (環境データベース, ジョブデータベース, リソースデータベース) を考える。ディスパッチデーモンは、ユーザから依頼されたジョブの配送先を決定し配送する。CPUサービスデーモンは、ディスパッチデーモンから配送されてくるジョブを処理し結果を返す。リソースデータベースは、コンピュータの静的属性を格納しておく。環境データベース, ジョブデータベースは、予想ターンアラウンドタイムを計算し、ジョブの配送先を決定する際に用いられる。(図1)

3　各データベースの役割

各データベースの例を図2に示す。

環境データベース (EDB)

EDBは、各サーバの状況 (ジョブの予想処理時間の合計, ネットワーク遅延など) を保存する。EDBを検索することで、ディスパッチデーモンは各サーバの待ち時間を知ることができる。

ジョブデータベース (JDB)

各ユーザに1つのJDBを用意する。JDBにはユーザのジョブの履歴を保存し、ジョブの名前と実行ディレクトリによってジョブの同一性と処理時間を予想する。

Implementation of an adaptive load sharing strategy with job characteristic
Takashi Shirotani, Takashi Watanabe, Tsuyoshi Ohta and Tadanori Mizuno
Faculty of Engineering, Shizuoka University
3-5-1 Johoku, Hamamatsu, Shizuoka

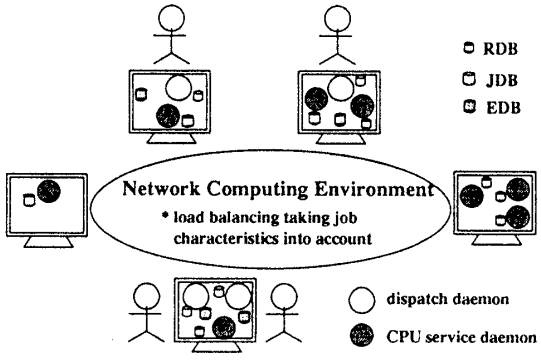

図 1: 提供する環境

リソースデータベース (RDB)

RDBには、個々のコンピュータの静的な要素 (OS等) と、そのコンピュータの待ち行列長 (ジョブの予想処理時間の合計) を保存する。ディスパッチデーモンは、起動されるとこのデータベースと通信することで各サーバのハードウェア情報を得る。

4　実装方針

ディスパッチデーモンは各ユーザごとに起動される。RDBはCPUサービスデーモンに対して1つ存在する。また、EDBは各コンピュータに1つずつあれば十分であるが、ここでは実装を簡単にするためにディスパッチデーモンに対して1つ用意する。ディスパッチデーモンへ、ユーザが投入したジョブは、EDBとJDBの情報をもとに配送先が決定される。ジョブの処理が行なわれる配送先のコンピュータには、CPUサービスデーモンが起動されている。

ジョブ配送の際、2つのプロセス間でTCPストリームコネクションが確立され、ジョブの配送が終了すると閉じられる。ジョブを受けとったCPUサービスデーモンは、いったんそれをスプールディレクトリへ格納した後に、到着順にバッチ処理を行なう。処理が完了したジョブから、結果が配送元のディスパッチデーモンへ返送される。この間にもTCPストリームコネクションが確立され、結果の返送が終了すると閉じられる。(図3)

5　環境情報の獲得と配送

5.1　環境情報の獲得方法

ネットワーク内に分散しているコンピュータの環境情報を得る方法として、

環境データベースの一例

ホストネーム	スピード	ネットワーク遅延	待ち行列長	プロセッサ名	OS	. . . . .
ホストA	12.9	3.54	130	sparc	4.1.3	. . . . .
ホストB	9.83	0.25	45.8	mc68040	5.0	. . . . .
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	

ジョブデータベースの一例

ジョブ名	実行ディレクトリ	配送先	実行時間	システム時間	ユーザ時間	. . . . .
sim1	/usr/local	B	45.8	10.3	3.56	. . . . .
command	/tmp/exec	A	2.44	0.58	0.21	. . . . .
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	

リソースデータベースの一例

スピード	待ち行列長	プロセッサ名	OS	. . . . .
12.9	130	sparc	5.1	. . . . .

図 2: データベースの一例

- ブロードキャストやマルチキャストによる方法 [1]
- メッセージパッシングによる方法 [1]
- ピギーバックによる方法 [3]

などが考えられる。本研究では環境情報を得る際のコストが比較的小さく、そして自分の配送したジョブの特性のみを知ればよいことを考え、ピギーバックにより情報を得る方法を採用した。つまり、結果が返送される際に環境情報を持ち返り、EDB や JDB の更新を行なう。また、過去に配送したジョブのうち、まだ結果の返ってきていないものの予想処理時間の合計を考慮することにより、より正確な負荷の分散がなされることが期待できる。

5.2 配送先の決定方法

配送先を決定するには、

- ランダムに配送する方法 [1]
- サイクリックに配送する方法 [1]
- スループットを最大にする方法 [2]
- 待ち行列長が最小のサーバへ配送する方法 [1]
- ターンアラウンドタイムが最小になるサーバへ配送する方法 [2]

などが考えられる。ここでは、予想ターンアラウンドタイムを最小にするようなサーバへジョブを配送する方法の実装を行なった。

サーバは n 台あるとする。EDB を検索して得られるサーバ $S_j(1 \leq j \leq n)$ での待ち時間 W_j とネットワーク遅延 D_j、そのサーバへ過去に配送してまだ結果が返ってきていないジョブの予想処理時間 R_j の 3 つの合計を予想ターンアラウンドタイムとする。ここで、サーバ S_j へ配送したジョブの結果がすべて返ってきていれば $R_j = 0$ になり、サーバ S_j において考えられる待ち時間は、W_j で予想せざるを得ないことに注意する。

JDB を検索し、全く同じ名前と実行ディレクトリを持つエントリからジョブ量 J_a(実行時間・配送先の処理速度) と配送しようとしているサーバ S_j の処理速度 c_j を取り出し、予想ターンアラウンドタイム T_j を以下のように計算する。

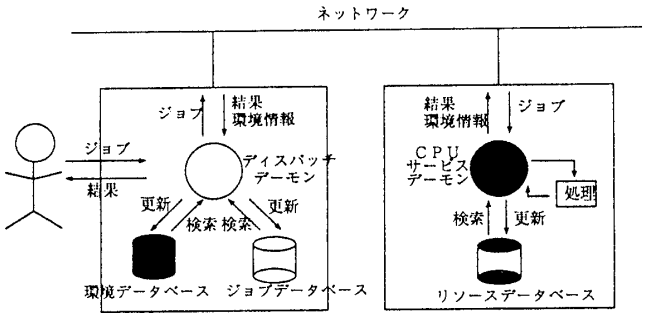

図 3: 実装の仕様

$$T_j = \frac{J_a}{c_j} + D_j + W_j + W_j$$

この T_j を最小にするサーバの集合を S_{min} とし、以下のように求める。

$$l = min\{T_j | 1 \leq j \leq n\}$$

$$S_{min} = \{S_j | T_j = l\}$$

ディスパッチデーモンは、S_{min} の中からランダムに 1 つの配送先を決定する。

6 まとめ

以上の方法でジョブを配送するシステムを SunOS5.1 を OS とする 20 台のワークステーション上で C 言語により実装し、動作を確認した。その結果、1 台で処理した場合と比べて処理時間が約 1/17 となることを観測した。

今後は、他の配送方法の実装、および情報獲得法の改良を行なう。

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 ans Walter H.Kohler : "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System", IEEE Trans. on Comput., Vol.28, No.5, 1979.

[3] 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.