マルチメディア通信と分散処理　69－4
（１９９５．　３．　２）

分散システムにおけるデータ更新性能解析と更新アルゴリズムの提案

宮西 洋太郎　　中村 健二　　渡辺 尚　　水野 忠則
三菱電機　静岡大学工学部　静岡大学工学部　静岡大学工学部

分散システムにおけるデータベース更新時には、並行処理制御や一貫性維持のために、ロック操作が用いられる場合が多い。データベースが複製を有する場合に通常は、全ての複製へのロック操作の後に全複製の更新処理が行なわれる。全複製へのロック操作および更新処理は応答時間低下の原因の一つとなりうる。

本論文では、待ち行列解析により近似的に更新処理の応答時間を求める方法を示し、シミュレーションの結果と対比する。また、応答時間性能の低下を避けるために、全複製にわたる広範囲なロックおよび更新処理を行なわないで、データ更新を行なう方式を提案する。この方式はデータの内容の意味を用いているので適用分野には制限があるが、応答時間の改善とネットワーク分断時のデータ更新に対して有効である。提案アルゴリズムについてもシミュレーションにより従来の方式との応答時間の比較を行なう。

Analysis of Data Update Performance in Distributed Systems and A Proposal of a Data Update Algorithm

Yohtaro Miyanishi*, Kenji Nakamura+, Takashi Watanabe+, and Tadanori Mizuno+

* Mitsubishi Electric Co.
7-10-4 Nishigotanda Shinagawa,Tokyo,141

\+ Shizuoka University
3-5-1 Johoku, Hamamatsu, 432, Japan

LOCK operations are usually used in the case of updating the distributed database. When the database has several replicas, the data may be updated after all replicas has been locked. Locking and updating of all replicas tend to degrade the performance.

We analyze the response time for updating data by using queueing model, and compare the results with the results of some simulations.

Then we propose a new updating algorithm which avoids locking and updating of all replicas. As this algorithm uses the semantics of data, the applications will be limited. However, it will be effective for keeping the performance well and it will be applicable to updating in the situation of network partition.

１．はじめに

分散システムにおけるデータベース更新において通常、ロック操作の後に更新処理が行なわれる。複製を有する場合には、全複製へのロック操作および更新処理はデータ更新応答時間性能低下の原因の一つとなりうる。

ここでは、ロック動作および更新処理が応答時間に与える影響について、待行列解析により近似的に応答時間を求める方法を示し、シミュレーションの結果と対比する。また、応答時間性能の低下を避ける一つのデータ更新方法を提案する。

２．関連する研究

分散データの更新処理において、データの並行処理制御(concurrency control)やデータ相互間・複製間の一貫性(consistency)維持のために、ロック操作を用いた、2PL(two phase lock)方式、2PC(two phase commit)方式が実用化されている。このようにロックを用いて複製を含むデー

タを更新する場合の性能の問題が本発表での検討対象である。分散システムの性能解析の多くは待行列モデルを用いて行われている。[1)2)]

本発表の前半では更新処理を単純化し、「全複製へのロックを行ない、全複製の更新を行ない、その後ロックを解除する」というモデルで、更新応答時間性能に焦点を絞って待行列モデルによる性能解析を行なう(４．)。

一方、性能向上を目的とした方式として、楽観的(optimistic)並行処理制御方式が研究されている[3)]。またネットワーク分断時の複製維持方式として投票-定数(vote-quorum)方式[4)]やデータの内容を利用する更新方式も研究されている[5)]。

楽観的制御方式では、衝突が発生して一貫性が侵害された場合には訂正操作(un-do)が必要であるが、外部への影響などは訂正ができない。定数方式はネットワーク分断時に、それぞれのサイトに存在する複製を更新するのを許容するか否かを判断するのに用いられる、一貫性は維持されるがアベイラビリティは低下する。

本発表の後半では応答性の向上、ネットワーク分断時のアベイラビリティ向上の目的で一つのデータ更新アルゴリズム案を提案する(５．)。

３．対象データのモデル

(1) 対象データのアクセス

検討の対象となる分散システム、データベースをそれぞれ図1.、図2.に示す。

データのアクセスには読込み操作と書込み操作があるが、本発表では書込み操作(以下更新と称する)を検討する。また一般的には書込みデータを生成するために、事前に読込みを行うことが必要であるが、ここでは簡単のため読込み時間と書込みの時間の和が一つの確率変数として指数分布に従うものとする。またアクセス要求の発生はポアソン分布に従うものとする。

対象となるデータは、データアイテム(j)単位に更新され、また 更新要求は地域的に分散した複数サイトの内の一つに到着し、そのサイト(i)で更新の管理がなされるものとする。システム全体の動作を定める次の変数を定義する。

λ_{ij} ： サイトiにデータアイテムjの更新要求が到着する到着率(1/Sec)(ポアソン過程)

X_{ij} ： 複製存在変数(2値変数)

X_{ij}=1ならばサイトiにアイテムjの複製あり

X_{ij}=0ならばサイトiにアイテムjの複製なし

s_{ij} ： サイトiに複製がある場合の、この複製単独の平均更新時間(Sec)(指数分布を仮定)

μ_{ij} ： 処理率、$\mu_{ij}=1/s_{ij}$

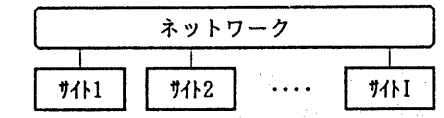

図1. 対象分散システム

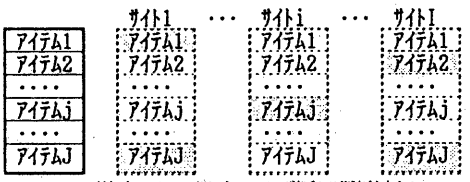

図2. 対象データベース 例えば、ハッチング部分には複製が存在する

(2) 対象データの特性

データの複製を配置する際、対象データの統計的特性を利用するが、５．で提案する更新アルゴリズムにおいては、さらに、そのデータの持つ意味も利用することを考える。

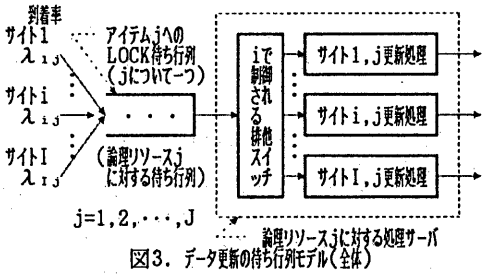

図3. データ更新の待ち行列モデル(全体)

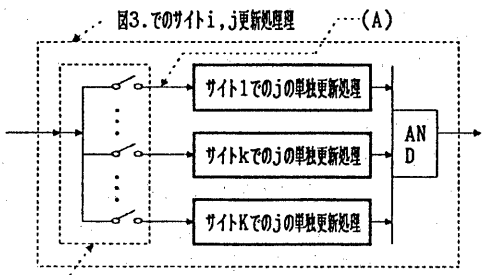

図4. データ更新の待ち行列モデル(サイトiでのjの更新処理)

4．待ち行列モデル

図3.及び図4.に待ち行列モデルを示す。まず、単一のデータアイテムへの複数の更新要求が到着する場合を検討する。次に複数データアイテムへの複数の更新要求が到着する場合を検討する。

4．1 単一データアイテム更新要求

(1) サイトiでの更新処理時間

サイトiでのアイテムjの更新処理時間は、当該アイテムjの複製が存在するサイトにおける単独の更新処理が全て完了するまでの時間である。すなわち、図4.での並行する複数の指数分布に従う処理が全て完了するまでの時間である。

サイト間での通信時間は無視できるものとする。またロックおよび解除の処理は優先的に行なわれ、それ自体の処理時間は無視できるものとする。また単一データアイテムの更新要求の場合であるので、ロックがなされていればサイトは空きの状態である。従ってサイトへの待ち時間は無い、すなわち図4.の(A)の箇所に待ち行列を発生しない。

アイテムjについて、複製が存在するサイトをk=1,2,,..,Kと番号を再付番する。例えば、アイテムjを省略して、サイト番号i=1の$X_{ij}=1$ならば、サイト番号 k=1番とする、以下同様にサイト番号をk=1,2,･･･,Kと付番する。

サイトiでのアイテムjの更新処理時間t の確率密度関数 $f_{ij}(t)$ を求める。

サイトkで、アイテムjについて$\tau \leqq t$に単独の更新処理が完了している確率すなわち、確率分布$p_k(t)$は、

$$p_k(t) = 1 - \exp(-\mu_k t) \qquad (1)$$

ただし、ここでのkは再付番したサイト番号。

全ての単独更新処理が t 以内で完了している確率分布$P_i(t)$は、同時確率であり、独立事象とみなし$p_1(t)\cdot\cdot\cdot p_K(t)$の確率分布の積で表される。$P_i(t)$はiによらず、アイテムjの複製の存在するサイトkに依存する。

$$P_i(t) = \prod_{k=1}^{K} p_k(t)$$
$$= (1-\exp(-\mu_1 t))\cdot\cdot(1-\exp(-\mu_K t))$$
$$= 1 + \sum_{r=1}^{K} (-1)^r Rr \qquad (2)$$

ただし、μ_k のサイト番号kは再付番のもの、

$$Rr = \sum_{k_1=1}^{K-r+1} \sum_{k_2>k_1}^{K-r+2} \sum_{k_3>k_2}^{K-r+3} \cdots \sum_{k_r>k_{r-1}}^{K} \cdot \exp(-(\mu_{k1}+\mu_{k2}+\mu_{k3}+\cdots+\mu_{kr})t) \qquad (3)$$

確率密度関数 fij(t) は、

$$f_{ij}(t) = \frac{d}{dt} P_i(t) = \sum_{r=1}^{K} (-1)^r R_r' \qquad (4)$$

ただし、'は時間での微分を表し、

$$R_r' = \sum_{k_1=1}^{K-r+1} \sum_{k_2>k_1}^{K-r+2} \sum_{k_3>k_2}^{K-r+3} \cdots \sum_{k_r>k_{r-1}}^{K} -(\mu_{k1}+\cdots+\mu_{kr})\cdot\exp(-(\mu_{k1}+\cdots+\mu_{kr})t) \qquad (5)$$

$1/\mu_{kr}$：サイトk_rに存在するアイテムj複製への単独の平均更新時間

このように、各サイトiについてのアイテムjの並列更新処理時間の確率密度$f_{ij}(t)$を求める。

以下、K＝3(複製の数の最大数が3)の場合について検討する、

$$f_{ij}(t) = \mu_1\exp(-\mu_1 t)+\mu_2\exp(-\mu_2 t) +\mu_3\exp(-\mu_3 t)-(\mu_1+\mu_2)\exp(-(\mu_1+\mu_2)t) -(\mu_2+\mu_3)\exp(-(\mu_2+\mu_3)t) -(\mu_3+\mu_1)\exp(-(\mu_3+\mu_1)t) +(\mu_1+\mu_2+\mu_3)\exp(-(\mu_1+\mu_2+\mu_3)t) \qquad (6)$$

ただし、μ_k のサイト番号kは再付番のもの、

この確率密度について、

一次モーメント(平均処理時間) $\overline{t}$

二次モーメント(処理時間の2乗平均) $\overline{t^2}$

を求める。これをSijとMijとする。

$$Sij = \frac{1}{\mu_1}+\frac{1}{\mu_2}+\frac{1}{\mu_3}-\frac{1}{(\mu_1+\mu_2)} -\frac{1}{(\mu_2+\mu_3)}-\frac{1}{(\mu_3+\mu_1)}+\frac{1}{(\mu_1+\mu_2+\mu_3)} \qquad (7)$$

$$Mij = \left(\frac{2}{\mu_1^2}+\frac{2}{\mu_2^2}+\frac{2}{\mu_3^2}-\frac{2}{(\mu_1+\mu_2)^2} -\frac{2}{(\mu_2+\mu_3)^2}-\frac{2}{(\mu_3+\mu_1)^2}+\frac{2}{(\mu_1+\mu_2+\mu_3)^2}\right) \qquad (8)$$

これを元のサイト番号とX_{ij}を用いて表す。

$$Sij = \sum_{n=1}^{I} X_{nj}\cdot(1/\mu_{nj}) - \sum_{n_1=1}^{I-1} \sum_{n_2>n_1}^{I} X_{n1j}\cdot X_{n2j}\cdot(1/(\mu_{n1j}+\mu_{n2j})) + \sum_{n_1=1}^{I-2} \sum_{n_2>n_1}^{I-1} \sum_{n_3>n_2}^{I} X_{n1j}\cdot X_{n2j}\cdot X_{n3j} \cdot(1/(\mu_{n1j}+\mu_{n2j}+\mu_{n3j})) \qquad (9)$$

$$M_{ij} = 2\sum_{n=1}^{I} X_{nj}\cdot(1/\mu_{nj})^2 - 2\sum_{n_1=1}^{I-1}\sum_{n_2>n_1}^{I} X_{n_1j}\cdot X_{n_2j}\cdot(1/(\mu_{n_1j}+\mu_{n_2j})^2) + 2\sum_{n_1=1}^{I-2}\sum_{n_2>n_1}^{I-1}\sum_{n_3>n_2}^{I} X_{n_1j}\cdot X_{n_2j}\cdot X_{n_3j}\cdot(1/(\mu_{n_1j}+\mu_{n_2j}+\mu_{n_3j})^2) \quad (10)$$

(2)ロック待ち時間

各アイテムについてのロック及び解除は複製を持っている各サイトに対して同期して行われる(サイト間の通信時間、ロック処理自体の処理時間はここでは無視する)ので、論理的に一つのリソースに対してロック及び解除を行っているものとみなせる。

図3において、「論理リソースjに対する処理サーバ」と記した部分の処理時間の確率密度分布は、(1)で求めた「サイトiでの更新処理」確率密度分布をアクセス到着率λ_{ij}でウエイト付けて合成されたものである。この分布は指数分布ではないが、待ち時間はM/G/1のポランチェック-ヒンチンの公式[例えば6)]により求めることができる。

サイト共通の平均待ち時間W_j

$$W_j = \frac{\rho_j\cdot S_j}{2\cdot(1-\rho_j)}\left(1+\left(\frac{\sigma_j}{S_j}\right)^2\right) \quad (11)$$

ただし、

$$S_j = \sum_{i=1}^{I}\frac{\lambda_{ij}}{\lambda_j}\cdot S_{ij}\ ,\quad \lambda_j = \sum_{i=1}^{I}\lambda_{ij} \quad (12)\ (13)$$

$$\rho_j = \sum_{i=1}^{I}\lambda_{ij}\cdot S_{ij} \quad (14)$$

$$\sigma_j^2 = \sum_{i=1}^{I}\frac{\lambda_{ij}}{\lambda_j}\cdot M_{ij} - (S_j)^2 \quad (15)$$

サイトi の平均応答時間T_{ij}は次式で求まる。

$$T_{ij} = W_{ij} + S_{ij} \quad (16)$$

T_{ij}、W_{ij}、S_{ij}の値はiによらず、jのみによる値となる。

4．2 複数データアイテム更新要求

この場合にはサイトには異なるデータアイテムの更新処理が到着するので、図4の(A)の箇所に待ち行列が発生する。ただし前段でロックが行なわれるので同一データアイテムの更新要求については待ち行列の中または処理サーバの中に1個のみ受け付けられる。このような待ち行列の解析には、有限数の状態の間での遷移確率から定常状態での状態確率、それによる平均待ち時間、平均応答時間を計算する方法が適用されるが、ここではより簡易的な計算方法を示す。

アイテムjの更新要求に対応するサイトiの処理能力c_{ij}は、

$$c_{ij} = 1 - \sum_{i=1}^{I}\sum_{l=1,l\neq j}^{J}\lambda_{il}\cdot s_{il} \quad (17)$$

である。すなわち、能力1から自アイテム(j)以外のアイテムに費やされる能力が差し引かれた能力である。

近似として、この能力のサーバがデータアイテムjに専属していると考える。仮想的に専属しているのでここでは、仮想専属サーバと仮称する。

この仮想専属サーバはjに専属するサーバであるので、即時に処理に入れる。従って、4．1での計算方法が、サーバの能力を低下させるという処置のみで、そのまま同様に待ち時間、応答時間を計算することができる。すなわちs_{ij}をつぎのように変更し、同様に計算する。

$$s'_{ij} = s_{ij}/c_{ij} \quad (18)$$

この仮想専属サーバの近似方法が適切か否かはシミュレーションによって確認するものとする。

4．3 シミュレーションとの比較

シミュレーションとの比較の例を以下に表1.および図5.に記す。

I=3, J=4, λ_{ij}は同一値、$1/\mu_{ij}$も同一値(=1)の場合、次の表に示す結果が得られた。

シミュレーションは期間20000秒を0.1秒ステップでサンプリングする離散時間型で行ない、各サイトの応答時間の平均値を求めた。

表1. 解析結果とシミュレーション結果
ρ_S:処理サーバ占有率(処理中率), ρ_L:各データアイテムのロック中率

λ_{ij} 1/秒	解析値			シミュレーション値		
	応答時間 秒	ρ_S	ρ_L	応答時間 秒	ρ_S	ρ_L
.001	1.857	0.01	0.006	1.873	0.01	0.005
.01	2.106	0.12	0.06	2.101	0.12	0.06
.02	2.479	0.24	0.13	2.447	0.24	0.13
.03	3.027	0.36	0.23	2.926	0.36	0.22
.04	3.919	0.48	0.34	3.705	0.49	0.33
.05	5.675	0.60	0.50	5.609	0.61	0.49
.06	11.092	0.72	0.72	10.243	0.73	0.70
.065	23.712	0.78	0.86	21.418	0.78	0.83
.068	93.397	0.82	0.96	41.784	0.81	0.90

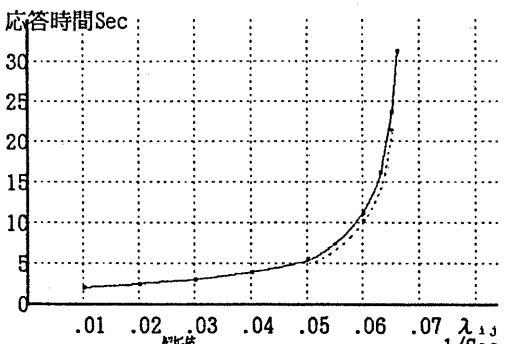

図5. 解析値とシミュレーション値のグラフ

5．データ更新アルゴリズム案

ここに提案する方式は「権限分割方式」とも呼べる方式で、各サイト毎にデータ更新の限界値を設定し、その範囲内で更新を許容する方式である。動作はmodestであるので、modestly optimistic concurrency control (MOCC)(制限付楽観的制御方式)と仮称することにする。

この方式が有効である適用分野は預金口座残高、在庫量、座席予約用座席数等のように共通の量を増減するような分野である。

通常のアクセス時は、上限値の範囲内で更新を行ない、夜間等のアクセスの閑散時に全体の一貫性回復するという方法である。またネットワーク分断時にも上限値範囲内での更新が可能である。

5．1 MOCCアルゴリズム

データアイテムのロック操作および更新処理は次の種類があるものとする。

・広域ロック(仮称)：アイテムjの全複製についてロックを行なう。

・狭域ロック(仮称)：自サイトにアイテムjの複製が存在していれば、自サイトの複製のみにロックを行なう。

・狭域更新処理(仮称)：狭域ロックを行ない、自サイトの複製を更新する。

・広域更新処理(仮称)：広域ロックを行ない、全複製を更新する。

(1) 準備状態

・データベースの維持を担当するサイト(ホストと称する)を定めておく。

・データアイテム毎またはアイテムの群毎に、ある一つのデータタイプ属性を持たせ、当該アイテムまたは当該アイテム群に対して本アルゴリズムを適用可能か否かを表現する。

データタイプ 0 ：本アルゴリズム適用不可

データタイプ 1 ：本アルゴリズム適用可能

・アイテムjの現在値をx_jとする。

・サイト毎、データアイテム毎に更新の限界率(limit rate)r_{ij}を持つ。

$$\sum_{i=1}^{I} r_{ij} = 1 \qquad (19)$$

・サイト毎、データアイテム毎に更新の限界値(limit value)u_{ij}を持つ。

$$u_{ij} = r_{ij} \cdot x_j \qquad (20)$$

・限界率は、適切なアルゴリズムによって定める。

例：・アクセス頻度に応じて重み付けを行う。

・特定サイトに重みを持たせる。

(2) アクセス要求発生時

サイトi に次のアイテムj 更新要求が発生する場合を考える、

更新要求：$x_j \rightarrow x_j - \Delta x_j$ (21)

Δx_j はx_j の変化分

次ぎのとき、本アルゴリズムの処理を行う。

・アクセス対象のデータアイテムが本アルゴリズム適用対象である(データタイプで判断)。

・データ更新処理

・ネットワークが正常の場合

・更新要求が

$$\Delta x_j \leqq u_{ij} \qquad (22)$$

ならば、狭域更新処理を行う。

・更新要求が

$$\Delta x_j > u_{ij} \qquad (23)$$

ならば、広域更新処理を行なう。

・ネットワークが異常の場合

(要求受け付けサイトとホストの間でのコミュニケーションが途絶えている場合)

・更新要求が

$$\Delta x_j \leqq u_{ij} \qquad (24)$$

の範囲内ならば狭域更新処理を行う。

・更新要求が

$$\Delta x_j > u_{ij} \qquad (25)$$

ならば、更新要求を棄却する。

(3) 回復処理(同期処理)

タイミング：

1) 同期時刻

2) ネットワーク復旧時

データベースのホストの役割をあるサイトに定めておき、一定時刻にデータベース整合のための処理を起動する。この処理は通常の広域ロックを行なってから、更新処理を行なう。

・ホスト：「回復要求」の発行

・各サイト：「回復要求」を受け取ると各サイトは現在の限界値u_{ij}(j=1,･･･,J)をホストに回答する。

・ホスト：回答されたu_{ij}を集計する。回答が無かったサイトi については前の限界値が保持されているものとみなす。回答があったサイトのみ限界値を再計算する。

$$SUM = \sum_{i=1}^{I} \delta i \cdot u_{ij} \qquad (26)$$

ただしサイトi から回答があれば$\delta i=1$、なければ$\delta i=0$とする。

回答があったサイトi に対して、次の値を新しい限界値u_{ij}とする。

$$u_{ij}=SUM \cdot r_{ij} / \left(\sum_{i=1}^{I} \delta i \cdot r_{ij}\right) \qquad (27)$$

これを回答があった全てのサイトに送信する。

・各サイト：送られてきたu_{ij}をサイトiでの新しい限界値u_{ij}とする。

5．3 シミュレーション結果

このアルゴリズムでは他サイト複製の更新処理は回復時以外には無くなるので、処理サーバへの負荷が軽減される。従って応答時間が改善されることは容易に予想できる。以下にシミュレーション結果の例を示す。

表2. 解析結果とシミュレーション結果
1/10回復アリ：データ更新要求頻度λijの10分の1の頻度で回復処理を行なう

λij 1/秒	従来方式解析 応答時間 秒	提案方式解析 応答時間 秒 回復処理ナシ	提案方式シミュレーション結果 応答時間 秒 回復処理ナシ	提案方式シミュレーション結果 応答時間 秒 1/10回復アリ
.01	2.106	1.042	1.051	1.020
.05	5.675	1.250	1.238	1.292
.06	11.092	1.316	1.300	1.366
.10	—	1.667	1.660	1.838
.15	—	2.500	2.587	3.368
.20	—	5.000	5.069	17.139
.21	—	6.250	6.275	89.232
.24	—	25.000	27.811	—
.245	—	50.000	46.760	—

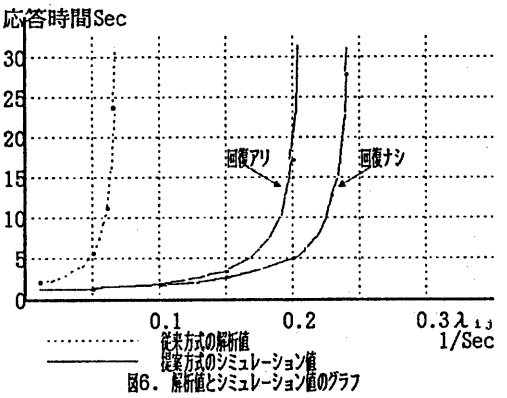

図6. 解析値とシミュレーション値のグラフ

6．おわりに

(1) ロックを用いて複数の複製を更新する場合の応答時間性能についての近似的な解析的評価方法を示した。またシミュレーションとの対比により近似方法がほぼ妥当であることを示した。

(2) 全複製ロックおよび更新処理に起因する性能低下を回避する更新アルゴリズムの概要を示した。

今後の課題として下記を検討する予定である。

・読み込みを含めた総合的性能評価方式の検討。

・サイト間通信時間、ロック自体の処理時間などを含めた実際のネットワーク構成に近い分散システムにおける性能評価方式の検討。

・性能評価と複製配置の関連の検討等。

最後に、有益な議論をいただいた静岡大学工学部佐藤文明助教授に感謝の意を表します。

7．参考文献

1)Kleinrock L.,"On the Modeling and Analysis of Computer Network," Proceedings of IEEE. V.81,No.8. '93

2)Singhal M.,"Update Transport: A New Technique for Update Syncronization in Replicated Database Systems" IEEE Transactions on Soft.Eng.,V.16,No.12 '90

3)Kung H.T.et al.,"On Optimistic Methods for Concurrency Control" ACM Transactions on Database System V.6,No.2 '81

4)Thomas R.H.,"A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases" ACM Trans. on Database System V.4,No.2 '79

5)Reading in DISTRIBUTED COMPUTING SYSTEMS Chapter 9,10,11 IEEE Comp.Soc.Press,1994

6)森村,大前「応用待ち行列理論」日科技連'75