論文番号: B5-3

アイドル計算機を用いた分散 ODB 処理の効率化の一検討

A consideration on efficient processing ways of distributed ODBs with idle machines

吉田裕介 †‡，有次正義 †，金森吉成 †

† 群馬大学工学部情報工学科

‡ 群馬県工業試験場 技術支援部

{yosida, aritsugi, kanamori}@dbms.cs.gunma-u.ac.jp

今までに，我々はマルチクライアント・マルチサーバの分散オブジェクトデータベース環境の効率化手法について提案してきた．その効率化手法は，サーバから永続オブジェクトを移動しクライアントで実行するデータマイグレーションと，クライアントからメソッドを移動しサーバで実行するメソッドマイグレーションを組合わせることで実現している．各サイトの負荷，処理するデータサイズなどを基に，最も小さいレスポンスタイムを与えるメソッドマイグレーション・データマイグレーションの組合わせを求める方法について考察してきた．本稿では，データマイグレーション，メソッドマイグレーションに加え，永続オブジェクトとメソッドを共にサーバ・クライアントとは異なるサイトに移動・処理し，結果のみをクライアントで受け取る方法を考え，これらを用いた処理の効率化を検討する．

1 はじめに

分散オブジェクトデータベースの効率化のためには，データマイグレーションだけでなくメソッドマイグレーションを組合わせることが有効であると報告されている [4, 5, 8].

今までに，我々はマルチクライアント・マルチサーバの分散オブジェクトデータベース環境の効率化手法について提案した [1, 2, 8, 9]．その効率化手法は，サーバから永続オブジェクトを移動しクライアントで実行するデータマイグレーションと，クライアントからメソッドを移動しサーバで実行するメソッドマイグレーションを組合わせ，それらの組合わせから最小のレスポンスタイムを与える組合わせを選択することによって実現している．マルチサーバ・マルチクライアントの環境ではデータマイグレーションとメソッドマイグレーションの組合わせが複数考えられ，その組合わせの中から最小のレスポンスタイムを与える組合わせを求めた.

文献 [8] の結果から効率の良い組合わせには，サーバサイトの負荷が大きく影響していることが解った．すなわち，より負荷の低いサイトでメソッド実行することで，更なる効率化を達成できると考えられる．

本稿では，従来のメソッドマイグレーションとデータマイグレーションに加え，永続オブジェクトとメソッドを共に移動し，サーバでもなくクライアントでもないアイドル状態のサイトで実行し，従来と比較してより効率的な処理を実現する方法ついて考察する．

2 データマイグレーションとメソッドマイグレーション

本節では，データマイグレーションとメソッドマイグレーションを行ったときのレスポンスタイムを求めるコストモデルについて説明する．

サーバサイト $S_i(1 \leq i \leq n)$ はサイズが $D_{S_i}(pages)$ のデータを管理している．適用するメソッドのサイズを $M(pages)$ とする．サーバサイトとクライアントサイト C の間のネットワークバンド幅を $NW(pages/sec)$, S_i のディスク I/O 速度を $DW_{S_i}(pages/sec)$, C の

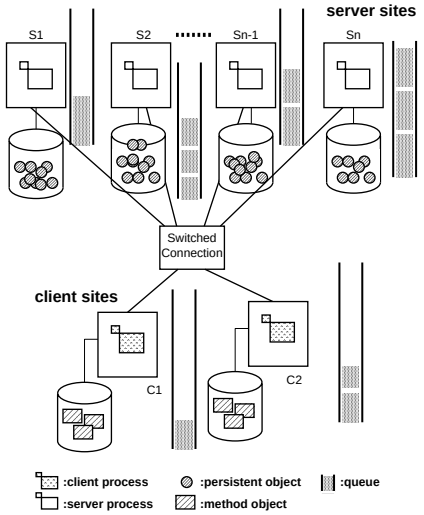

図 1: 分散データベース環境の例

ディスク I/O 速度を $DW_C(pages/sec)$, C が1秒あたりに処理できるデータのサイズを $PT_C(pages/sec)$, S_i が 1 秒あたりに処理できるデータのサイズを $PT_{S_i}(pages/sec)$ で表す．図 1に示すように，各サイトはスイッチング装置で接続されており，文献 [6] の設定と同様に，サイト間の通信が他のサイト間の通信と競合することはないものと仮定する．図 1の中心にある長方形はスイッチングハブを表現しいる．

以下では，メソッドを実行するサイトにメソッドを移動し，プロセスにロードする時間を T_M で表す．式 (1) 中の DW はメソッドが存在するサイトのディスク I/O 速度を表す．第 2 項はメソッドの実行が行われるサイトにメソッドを移動するためのネットワークの転送時間であり，メソッドが存在するサイトとメソッドを実行するサイトが同一の場合は 0 になる．

$$T_M = M \times \frac{1}{DW} + M \times \frac{1}{NW} \quad (1)$$

ここでは，次節の説明に必要なサーバサイトが 1 台 (S_1)，クライアントサイトが複数台の環境について説明する．マルチサーバ・マルチクライアントの説明ではここでは省略する．本稿で省略した詳細については文献 [8] を参照されたい．

C が S のデータをデータマイグレーションで処理するときのレスポンスタイム T_{dm} は，以下のようになる．

$$T_{dm} = T_M + D_S \times (\frac{1}{DW_S} + \frac{1}{NW} + \frac{1}{PT_C}) \quad (2)$$

クライアントサイトでメソッドが管理されている時，メソッドが実行されるサイトとメソッドが存在するサイトは同一であるため，第 1 項は C が $M(pages)$ のメソッドをディスクから読み出す時間に相当する．第 2 項は S が $D_S(pages)$ のデータをディスクから読み込み，ネットワークで転送し，C がメソッドを実行する時間の合計である．

C が S のデータをメソッドマイグレーションで処理するときのレスポンスタイム T_{mm} は，以下のようになる．

$$T_{mm} = T_M + D_S \times (\frac{1}{DW_S} + \frac{1}{PT_S} + \frac{f}{NW}) \quad (3)$$

クライアントサイトでメソッドが管理されている時，メソッドが実行されるサイトとメソッドが存在するサイトは異なる．したがって，第 1 項は C が $M(pages)$ のメソッドをディスクから読み出しサーバへ転送する時間になる．第 2 項は $D_S(pages)$ のデータを S がディスクから読み込み，メソッドを実行し，その実行結果を転送する時間である．ここで，$f(0 \leq f \leq 1)$ を用いて，メソッドの実行結果のサイズを fD_S と表す．

メソッドマイグレーションで処理する場合，他のクライアントの処理要求によるサーバサイトの処理能力の低下を考慮する必要がある．ここでは，サーバサイトの処理能力低下を考慮した場合，1 秒あたりに CPU が処理できるデータサイズ，ディスク I/O 速度はそれぞれ PT'_S, DW' で表すことにする．

3 アイドル計算機の有効利用

データマイグレーションとメソッドマイグレーションを組合わせた場合，メソッドを実行するサイトの負荷がレスポンスタイムに大きく影響することがわかっている [8]．したがって，効率化を考えた場合，メソッドを実行するサイトとして負荷が低いサイトを選ぶのが適切である．

本節では，永続オブジェクトとそのメソッドを共に移動する新しい処理手法について議論する．データマイグレーション，メソッドマイグレーションではメソッドを実行するサイトの候補としては，永続オブジェクトが保存されているサーバ (S)，あるいはメソッド実行のリクエストを発行するクライアント (C) に限られていた．新しい処理手法では，メソッド実行する候補となるサイトとしてアイドル計算機 (I) を考える（図 2）．本稿ではまず，サーバ，クライアント，アイドル計算機がそれぞれ各一台の場合を考える．

前節のコスト式に従い，レスポンスタイムは次の式 (4), (5), (6) で表せる．T_S, T_C, T_I はそれぞれサーバ (S)/ クライアント (C)/ アイドル計算機 (I) でメソッド実行した場合のレスポンスタイムを表す．

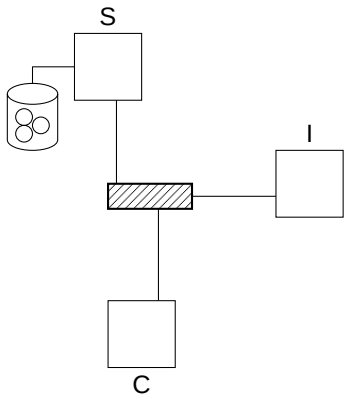

図 2: メソッド実行の候補となるサイト

$$T_S = T_M + D_S \times (\frac{1}{DW'_S} + \frac{1}{PT'_S}) + D_S \times f \times \frac{1}{NW} \quad (4)$$

$$T_C = T_M + D_S \times \frac{1}{DW'_S} + D_S \times (\frac{1}{NW} + \frac{1}{PT_C}) \quad (5)$$

$$T_I = T_M + D_S \times (\frac{1}{DW'_S} + \frac{1}{NW}) + D_S \times \frac{1}{PT_I} + D_S \times f \times \frac{1}{NW} \quad (6)$$

各サイトでメソッド実行した場合のレスポンスタイムを比較するために，式 (4), (5), (6) の差を以下のように表す．

$$Diff(S,C) = T_S - T_C \quad (7)$$
$$Diff(C,I) = T_C - T_I \quad (8)$$
$$Diff(S,I) = T_S - T_I \quad (9)$$

どのサイトでメソッドを実行するのが最も効率が良いかは，式 (7), (8), (9) の正負の組合わせから表 1のようになる．表 1からデータマイグレーションとメソッドマイグレーションの 2 者のみの組合わせと比較して，さらに効率化が達成できると考えられる．

式 (7), (8), (9) を比較するために PT_C を基準として $PT'_S = \frac{1}{\alpha}PT_C, PT_I = \frac{1}{\beta}PT_C$ と表し，両辺を D_S で割ると，式 (7), (8), (9) は以下のように表せる．

$$\frac{Diff(S,C)}{D_S} = (\alpha - 1)\frac{1}{PT_C} + (f-1)\frac{1}{NW} \quad (10)$$

$$\frac{Diff(C,I)}{D_S} = (1-\beta)\frac{1}{PT_C} - f\frac{1}{NW} \quad (11)$$

$$\frac{Diff(S,I)}{D_S} = (\alpha - \beta)\frac{1}{PT_C} - \frac{1}{NW} \quad (12)$$

PT_C, NW は計算機の性能によってあらかじめ決定されるので，メソッド実行サイトとして適切なサイトの選択は，以下の 3 つの要因から決定される．

1. クライアントサイトとサーバサイトの CPU の処理能力の比率 (α)
2. クライアントサイトとアイドル計算機の CPU の処理能力の比率 (β)
3. メソッド返り値の割合 (f)

	$Diff(S,I) > 0$	$Diff(S,I) < 0$
$Diff(C,I) > 0$	$I(T_I < T_C < T_S)$	-
$Diff(C,I) < 0$	$C(T_C < T_I < T_S)$	$C(T_C < T_I < T_S)$

$Diff(S,C) > 0$ の場合

	$Diff(S,I) > 0$	$Diff(S,I) < 0$
$Diff(C,I) > 0$	$I(T_I < T_S < T_C)$	$S(T_S < T_I < T_C)$
$Diff(C,I) < 0$	-	$S(T_S < T_C < T_I)$

$Diff(S,C) < 0$ の場合

表 1: 効率的なメソッド実行サイト

4 実験

4.1 実験環境

本稿の考察を検証するために実験を行った．実験環境を表 2に示す．サーバとアイドル計算機には同じ性能を持った計算機を用いた．クライアントサイトにはサーバ / アイドル計算機よりも性能の低い計算機を用いた．各計算機は 100Mbps のイーサネットにスイッチングハブで接続されている．

実験のため，我々は `name`， `age`， `salary`， `x` と 2Kbytes のビットマップ画像の `image` の属性をもつクラス `Person` のオブジェクト集合を用いた．これらの属性の数は， OO1 ベンチマーク [3] の `Part` の持つ属性のうち，リスト構造である `to` と `from` を除いた個数により決定した．属性 `age` は， 0 から 99 の範囲のランダムに生成された整数で，各サーバサイトでの値は一様に分布している．各サーバサイトでは，5000 個の `Person` オブジェクトを要素に持つ集合を生成した．また，属性 `salary` は `age` から導出した値を用いた．実験では， `Person` オブジェクトは生成

Site	S_1, I	C
Type	Sun Ultra30	Sun Ultra1
CPU	UltraSparcII	UltraSparcI
Clock	248MHz	167MHz
Memory	128MB	128MB
Disk	SUN4.2G	SUN2.1G
Page size	8192(bytes)	
OS	Solaris2.6	
Java 処理系	J2SE 1.3.1	
DBMS	Berkeley DB 4.0.14	

表 2: 実験環境

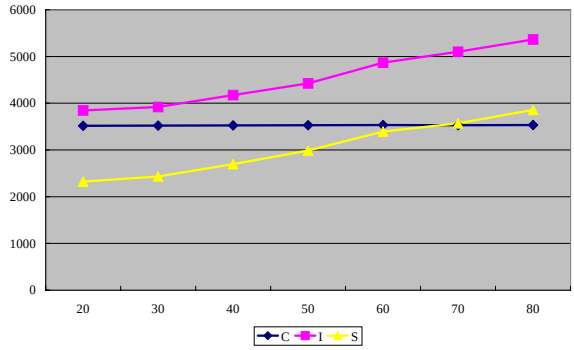

図 3: $\rho = 0.0$

順に集合オブジェクトに挿入され，インデックス等は考えていない．

Rodríguez-Martínez 等の報告の通りメソッドをあらかじめ分散しておくことは困難である [5] と考え，実験ではメソッドのあるサイトをクライアントサイトとした．

4.2 負荷の生成

サーバにおける他のクライアントの要求を処理することによる影響を模倣するために，CPU およびディスク I/O の使用率を調整するプロセスをサーバサイトで生成する [7]．このプロセスは，CPU（ディスク I/O）の利用を $\rho_{cpu}(\rho_{disk})$ 秒間繰り返す処理と，$1-\rho_{cpu}(1-\rho_{disk})$ 秒の WAIT 状態を繰り返す．したがって，このプロセスは CPU（ディスク I/O）処理能力の $\rho_{cpu}(\rho_{disk})$ を使用する．CPU の使用率を調整するプロセスは次のように実装されている．1) 変数 loop に 1 を代入し，初期化する．2) SIGALRM シグナルのハンドラを設定する．シグナルハンドラは変数 loop に 0 を代入する関数である．3) ρ_{cpu} 秒間有効なタイマを設定する．4) 変数 loop が非 0 である間，特定の変数の加算を繰り返す．5) タイマにより 3) のループを抜けた後，SIGALRM シグナルのハンドラを再び設定する．シグナルハンドラは変数 loop に 1 を代入する関数である．6)$(1-\rho_{cpu})$ 秒間有効なタイマを設定する．7) シグナルを受信するまで休眠し，5) のタイマにより覚醒する．8) 2)-7) を無限に繰り返す．ディスク I/O 使用率を調整するプロセスの実装は次の点を除いて CPU の場合と全く同じである：上記説明の ρ_{cpu} を ρ_{disk} に置き変え，4) において 8192 バイトのバッファをファイルへ書き込む操作を繰り返し実行する．

シグナルハンドラの設定には `signal()` 関数，タイマの設定には `setitimer()` システムコール，プロセスの休眠は `pause()` システムコール，ファイルの書き込みは `write()`, `lseek()` システムコールを用いている．本稿の実験では ρ_{cpu} と ρ_{disk} を設定し，他のクライアントの要求処理による影響の大きさを調整している．本稿では ρ_{cpu} と ρ_{disk} は同じ値としている．

4.3 実験結果と考察

実験ではデータの返す割合を 20, 30, ···, 80 まで変化させながら，メソッドを 1) サーバで実行した場合 2) アイドル計算機で実行した場合 3) クライアントで実行した場合 についてレスポンスタイムを計測した．さらに，サーバの負荷 ρ を変化させた場合について同様の計測を行った．ρ は 0.0, 0.2, 0.5, 0.8 の場合を用いた．図 3, 4, 5, 6に実験結果を示す．各グラフの縦軸はレスポンスタイムを表し，横軸は実行結果の割合 f を表す．

表 3にパラメータと表 1から決定される効率的なメソッド実行サイトの関係を示す．クライアントとサーバの CPU の処理能力の比率 α は，表 2から $\alpha = PT_C/PT'_S = 167/(248 \times (1-\rho))$ と表せる．また，クライアントとアイドル計算機の CPU の処理能力の比率 β は，$167/248 \simeq 0.673$ と計算される．また，PT_C, NW は実験に用いた環境で実測し，$PT_C = 520.0$, $NW = 273.6$ としている．

実験結果のグラフ図 3, 4, 5, 6と表 3を比較すると，多くの場合で効率の良いメソッド実行の候補サイトを同定している．ただし，サーバの負荷 ρ が大きいときと実行結果の割合 f が大きいときにコストモデルの誤りが見られる．これは，サーバで負荷がかかったときの影響を，コストの見積もりが過小評価しているためであると考えられる．今後，コストモデルを調整する

ρ	α	β	f	$Diff(S,C)$	$Diff(C,I)$	$Diff(S,I)$	効率的なサイト
0.0	0.673	0.673	20,$\cdots$, 80	負	負	負	S
0.2	0.841	0.673	20,$\cdots$, 80	負	負	負	S
0.5	1.346	0.673	20,$\cdots$, 80	負	負	負	S
0.8	3.366	0.673	20,$\cdots$, 80	正	負	正	C

表 3: パラメータとメソッド実行の候補サイト

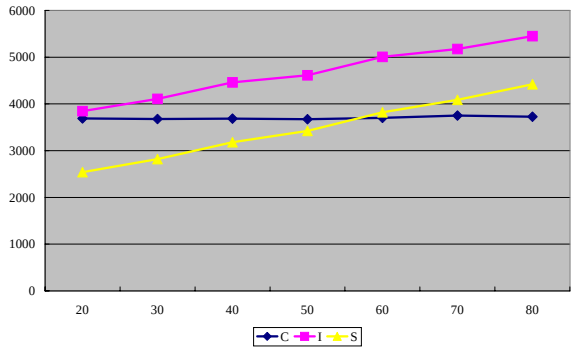

図 4: $\rho = 0.2$

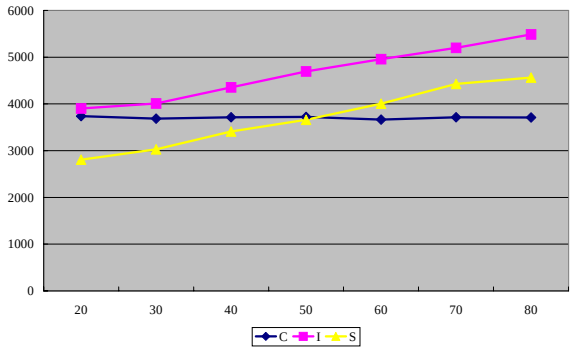

図 5: $\rho = 0.5$

必要がある．

また，サーバ / アイドル計算機 / クライアントが各一台の環境では，アイドル計算機が効率の良いサイトとなることは無かった．これは，アイドル計算機にデータを転送する手間をかけるよりも，サーバあるいはクライアントでメソッドを実行してしまったほうが効率が良いからであると考えられる．つまり，アイドル計算機でメソッドを実行する際のオーバヘッドが，アイドル計算機で実行することによる効率化を上回っているからである．今後は，マルチサーバ環境でアイドル計算機を有効利用する方法について検討する予定である．

5 まとめ

分散オブジェクトデータベース処理において，メソッドを実行するサイトとしてクライアント / サーバ / アイドル計算機を候補としたとき，最も効率の良いサイトを選択する基準について考察し，定式化した．多くの場合で効率の良いサイトを選択することができた．さらに精度を高めるにはコストモデルの修正が必要である．また，サーバ / アイドル計算機 / クライアントが各一台の環境ではアイドル計算機でメソッド実行することは効率化に寄与しない．今後は，マルチサーバ環境でアイドル計算機の有効利用を検討する予定である

参考文献

[1] M. Aritsugi, Y. Yoshida, T. Nakamura, and Y. Kanamori, “Method migration in object database environments,” Proc. 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI2000), vol. VIII, pp.125–130, Jul. 2000.

[2] 有次 正義, 吉田 裕介, 中村 知久, 金森 吉成, “分散メソッドの提案,” 情処学 DBS 研報, pp.189–196, Jul. 1998.

[3] R.G.G. Cattell and J. Skeen, “Object operations benchmark,” ACM Trans. Database Syst., vol.17, no.1, pp.1–31, 1992.

[4] M.J. Franklin, B.T. Jónsson, and D. Kossmann, “Performance tradeoffs for client-server query processing,” Proc. ACM SIGMOD Conf., pp.149–160, June 1996.

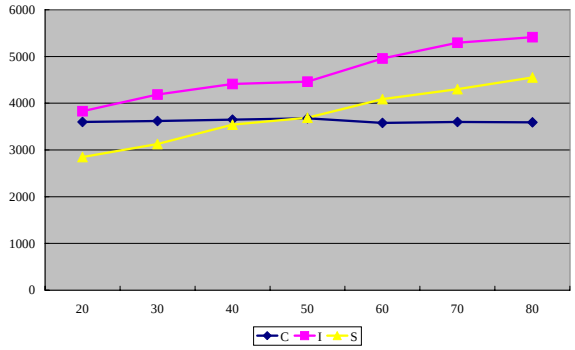

図 6: $\rho = 0.8$

[5] M. Rodríguez-Martínez and N. Roussopoulos, “MOCHA: A self-extensible database middleware system for distributed data sources,” Proc. ACM SIGMOD Conf., pp. 213–224, May 2000.

[6] K. Voruganti, M. T. Özsu, and R. C. Unrau, “An adaptive hybrid server architecture for client caching object DBMSs,” Proc. 25th Intl. Conf. on Very Large Data Bases, pp.150–161, Sept. 1999.

[7] 谷口秀夫, “入出力性能の制御によりプログラム実行速度を調整する制御法の実装方式による比較評価,” 信学論 (D-I), vol.J84-D-I, no.9, pp.1362–1371, Sep. 2001.

[8] Y. Yoshida, M. Aritsugi, and Y. Kanamori, “Performance evaluation of combining data migration and method migration in object database environments,” Proc. 13th Australasian Database Conf.(ADC2002), Jan. 2002.

[9] 吉田 裕介, 中村 知久, 有次 正義, 金森 吉成, “分散環境でのデータ移動とメソッド移動,” 電子情報通信学会データ工学専門委員会, 第 9 回データ工学ワークショップ, March 1998.