システムソフトウェアと
オペレーティング・システム 77－14
マルチメディア通信と分散処理 87－14
（1998. 2. 26）

分散オブジェクト複製配置プロトコルRORPについて

森 章文　渡辺 尚

静岡大学 理工学研究科　静岡大学 情報学部

分散オブジェクト指向技術はオブジェクト単位で開発、修正が可能であるという利点があるものの、広域網で用いると通信遅延による処理性能が低下する。本研究ではオブジェクトの複製を適切なサイトに動的に配置することによって処理性能を向上させる複製再配置プロトコルRORPを提案する。複製はクライアントもしくはサーバによって更新される。RORPはオブジェクト複製の蓄積効率を考慮し複製の予約、共有を行なう。本論文では、それらの特徴について述べ、Caching goes Replicationあるいは、Squidを用いた多段キャッシュとの比較を行う。

Replicated Object Relocation Protocol RORP

Akifumi Mori　Takashi Watanabe

Graduate School of Science and Engineering, Shizuoka University
Faculty of Information, Shizuoka University

This paper proposes a relocation protocol of replicated object in a widely distributed object system, which is called RORP. RORP creates a replicated object and allocates it on an intermediate site to reduce latency of object retrieval. A replicated object is updated by a client (lazy update) or by a server (immediate update). RORP also supports reservation of replication of an object, and it allows several clients to share a replicated object to enhance storage efficiency. The protocol is summarized and compared with Caching goes Replication (CgR) protocol and multiple stage caching protocol, Squid.

1　はじめに

近年、分散環境における技術として分散オブジェクト指向技術[1][2][3][4]が発展しつつある。分散オブジェクト指向では、図1で示されるように

- オブジェクト同士がクライアントとサーバという関係を持つ
- クライアント・オブジェクトがサーバ・オブジェクトに対して処理要求を発行し、その要求をサーバ・オブジェクトが処理し結果をクライアントに返すという仕組みを提供する
- ネットワークを1つの巨大なコンピュータであるかのように動作させることができる

といった特徴を持つ。

この技術にはオブジェクト単位で開発、修正が可能であるという利点がある反面、WANで用いたときには通信遅延の影響による問題が生じる。例えば、クライアントが処理要求を発行してからサーバに要求が伝達されるまでに大きな通信遅延がかかり、そのため効果的にユーザの要求を処理できない可能性がある。本研究では、クライアントとサーバ間の通信遅延、サーバの処理性能、オブジェクトの参照頻度、複製の転送時間などを考慮し適切なサイトにオブジェクト複製を転送し配置することによって処理性能を向上させるプロトコルRORP(Replicated Object Relocation Protocol)を提案する。

2　RORPの概要

簡単に言うと、RORPは処理要求、結果返却にかかる通信遅延、オブジェクトの参照頻度、サイトの処理速度等を考慮し適切なサイトに複製を再配置する。例えば、図2のように通信遅延、オブジェクトの参照頻度、サイトの処理速度を考慮してsiteへ、serverに存在しているオブジェクトの複製を配置する。複製再配置後は複製の配置されたサイトに対してオブジェクトの参照を行なう。RORPは、

- ドメイン, 複製数の制約を考慮した複製の再配置
- 複製の更新
- 複製配置予約, 共有

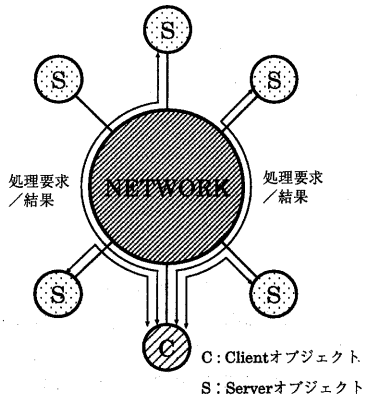

図 1: 分散オブジェクト指向

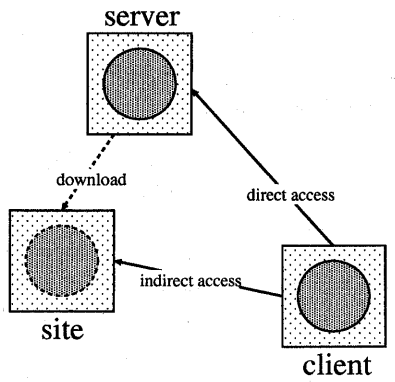

図 2: RORP の複製再配置機能

といった特徴をもつ。それぞれの特徴を次に説明する。

3 RORP 特徴

3.1 複製の再配置

再配置のタイミング

クライアントは、再配置間隔 (t_{rel}) ごとに候補となる各中間サイト I_i に対して

$$f_i(n) = n(s_c + c_c) - n(s_i + c_i) - t_i$$

を計算する。ここで、n はオブジェクトの参照頻度 (再配置間隔あたりのオブジェクトの参照回数) 、s_i はサイトの処理時間、c_i はクライアントとサイトとの通信遅延、t_i はサイト c からサイト i への複製転送時間を示している。また、s_c, c_c は現在参照しているサイトの処理時間、通信遅延を表している。クライアントは、$f_i(n) > 0$ を満たす中間サイトの中で最も大きい $f_i(n)$ を持つサイト I_i を選択する。選択したサイトに複製が配置されれば再配置間隔あたりの全通信遅延は小さくなることが期待される。そこで、クライアントはオブジェクト複製を選択したサイトに配置することを要求し、以後そのサイトに対してオブジェクトの参照を行なうことになる。なお、後に述べるように、配置を許可しないドメイン、複製の総数を考慮し配置する。

オブジェクト配置テーブル (ODT)

クライアントは表 1に示すようなオブジェクト配置テーブル (object distribution table; 以下 ODT と略す) を持つ。ODT は、それぞれのオブジェクトが現在どのサイトに存在しているかを示すカレントサイト、t_{rel} 間にオブジェクトが参照される頻度を示す参照頻度、オブジェクトサイズ、オブジェクト処理時間、またクライアントとカレントサイト間の通信遅延、オブジェクトのオリジナルサーバ再配置間隔 t_{rel}、オブジェクトを共有している他のクライアントを示す共有クライアント、更新間隔を含んでいる。

中間サイトは、オブジェクトが有効であるかを示す有効フラグ、共有クライアント、オリジナルサーバ、更新権所有クライアントを含む ODT を持つ。その ODT を表 2に示す。

サーバは、中間サイト、共有クライアント、参照頻度、更新間隔 (t_u)、規制数、予約待ち行列、スワップ間隔を含むオブジェクト配置テーブル ODT を持つ。これを表 3に示す。

複製配置シーケンスの例

図 3に複製再配置のシーケンスの例を示す。ここで、$C_{c1} + C_{c2} = C_c$ である。Client は再配置間隔 (t_{rel}) を過ぎると $f_i(n)$ を計算し、Site i を選択し、そのサイトに対し *relocation* メッセージを送り再配置を要求する。Site i はカレントサイト Site 1 に対して *transmission* メッセージを出すのと同時に original server に *being relocation* メッセージを出し現在再配置を実行中であることを伝える。*transmission* メッセージを受け取った Site 1 は Site i にオブジェクトを転送する。転送が終了すると Site i は *completion* メッセージを Client, Site i,original server に送り転送が終了したことを伝え、同時に ODT を更新する。*completion* メッセージを受け取った Client、Site i、original server は ODT を更新する。

3.2 複製の更新

更新のタイミングと更新権

RORP ではネットワーク上に配置された多数の複製の一貫性を維持する必要性がある。RORP では、

オブジェクト名	カレントサイト	参照頻度	オブジェクトサイズ	処理時間	通信遅延	オリジナルサーバ	再配置間隔 (t_{rel})	共有クライアント	更新間隔 (t_u)
object1	site1	5	50k	50msec	0.3 sec	server X	3 hr	c2,c3	24hr
object2	site1								

表 1: クライアントの ODT

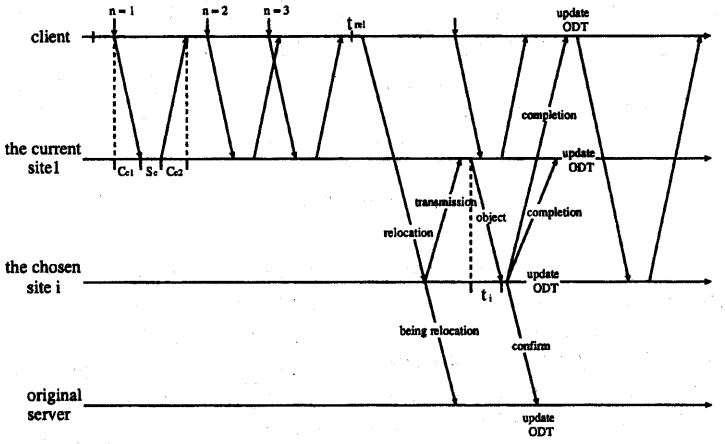

図 3: 複製再配置シーケンス

複製の一貫性を維持するために次のとき複製の更新を行なう。

- lazy update: クライアントが更新間隔 (t_u) ごとに更新要求を複製配置サイトに対して出す。
- immediate update: サーバの変更が重要であるとき複製配置サイトの複製を更新する。

lazy update では、クライアントが更新要求を出すことによってオブジェクトが更新される。後に述べるように複製が共有されることがある。この時は共有されているクライアント群の中で 1 つのクライアントが更新する権利 (更新権) を持つ。更新権を持つクライアントは複製を持つ中間サイトによって選択され与えられる (例えば、最初にそのオブジェクトにアクセスしたクライアント)。もし更新権のあるクライアントがその複製の参照をやめた場合は、中間サイトは更新権を持つクライアントを再度選択する。

更新間隔の動的変更

更新間隔 (t_u) ごとに更新権を持つクライアントは中間サイトを経由してサーバに対して更新要求を出す。もし、オブジェクトの変更がなければ *reject* メッセージをクライアントに返す。*reject* メッセージを受け取ったクライアントは更新間隔 (t_u) を 2 倍に書き換える。変更があれば、1 つ前の変更時間 (T1) と共に新しいオブジェクトを中間サイトに送る。クライアントは中間サイトから T1 を受け取ると更新要求を出した時間 (T2) を比較し、T1 のほうが大きいとき更新間隔 (t_u) をに $\frac{1}{2}$ にする。

複製更新シーケンスの例

複製更新のシーケンスの例を図 4に示す。図に示される例では、１５秒にクライアントが更新要求を出し、オブジェクトに変更がないためサーバはクライアントに *reject* メッセージを返す。*reject* メッセージを受け取ったクライアントは更新間隔 (t_u) を 2 倍にし、３１秒に次の更新要求を出す。この時オブジェクトには変更があったため、サーバは中間サイトの複製を更新するのと同時に T1(この場合２０) をクライアントに送る。T1 を受け取ったクライアントは T2(この場合１５) と比較し T1 の値のほうが大きいため更新間隔 (t_u) を $\frac{1}{2}$ にし３９秒に更新要求を出す。またこの例では、４５秒に重要な変更があったのですぐに immediate update を行なっている。

3.3 複製の共有と予約

単純に多数の複製をネットワーク上に配置するとネットワーク全体から見た蓄積効率の低下、オブジェ

オブジェクト名	有効フラグ	共有クライアント	オリジナルサーバ	更新権所有クライアント
object1	Y	c1,c2,c3	server X	c1
object2	N			

表 2: 中間サイトの ODT

オブジェクト名	中間サイト	共有クライアント	参照頻度	更新間隔 (t_u)	規制数	予約待ち行列	スワップ間隔	制限ドメイン
object1	site A	c1		24hr.	3	c3	10min	
	site B	c2,c3						
	site C	-						

表 3: サーバの ODT

クト複製の一貫性保持の難しさといった2つの欠点が生じる。RORP では、これらの問題を解決するために分散されたオブジェクトの総数規制を行なう。複製の総数規制を行なうと、複製がすでに規制数配置された状態のとき次のような問題について考慮する必要性がある。(問題1) クライアントは再配置要求を出しサーバはクライアントに対して *reject* メッセージを返すということを繰り返し続ける。(問題2) どれかのオブジェクトが開放されるまで、クライアントは複製を再配置することができない。

予約

(問題1) に対しては、サーバはクライアントに複製配置要求の予約を許すことで対処する。すなわち、サーバはクライアントからの再配置要求を受け取った時、もしすでに規制数以上の複製を配置済みでその要求に応じられなければ、その要求を予約待ち行列に入れ、そのことをクライアントに伝えクライアントはその後、要求は出さない。サーバはスワップ時期になると予約待ち行列の中から1つのクライアントを選択し、まだ複製を必要としているのかをクライアントに尋ね、必要なときはあるサイトの複製の使用を無効としクライアントに再配置を許可する。

図5は予約のシーケンスの例を示している。クライアントは *relocation* メッセージを中間サイトを経由しサーバに対して出す。しかし、規制数の複製がすでに配置されているためサーバはその要求を予約待ち行列に入れクライアントに *enqueueing* メッセージを送る。スワップ時期になるとサーバは再配置がまだ必要かという asking メッセージをクライアントに対して送る。この場合クライアントは複製を必要とするためサーバに *still_need* メッセージを送る。これを受け取ったサーバは再配置可能であることを伝える *relocatable* メッセージをクライアントに送る。そして、クライアントは再度 *relocation* メッセージを中間サイトを経由してサーバに出すことで複製が再配置される。

共有

(問題2) に関しては、複数のクライアントがオブジェクト複製を共有して解決する。複製を共有することで最適ではないが、現在よりアクセス効率のよいサイトの複製に対してオブジェクトの参照を行なうことができる。

RORP では、次のようなとき複製の共有を行なう。これはサーバからのメッセージによりクライアントが判断する。

- クライアントが出した複製配置要求が予約待ち行列に入れられたとき
- スワップ時期にそのクライアントが参照していたサイトの複製が無効とされたとき

このようなとき、サーバはクライアントに各サイトの情報を送り、そのデータをもとにクライアントはその中で最も適したサイトを選択し、そのサイトに対してオブジェクトの参照を行なう。

図6に示すように、Client1 の複製配置要求が予約待ち行列に入れられると Server からの共有サイト情報をもとに Site1 を共有サイトと決定する。また、複製がスワップされると Site2 は、他のサイト情報をそのサイトにオブジェクトの参照を行なっている全てのクライアントに転送し、各クライアントはその情報から適切なサイトを決定する。

図7に共有の例を示す。スワップ時に予約待ち行列からクライアント C1 が選択された結果、クライアント C3 がスワップアウトされている。その後、C3 は別のサイト S2 を選択している。

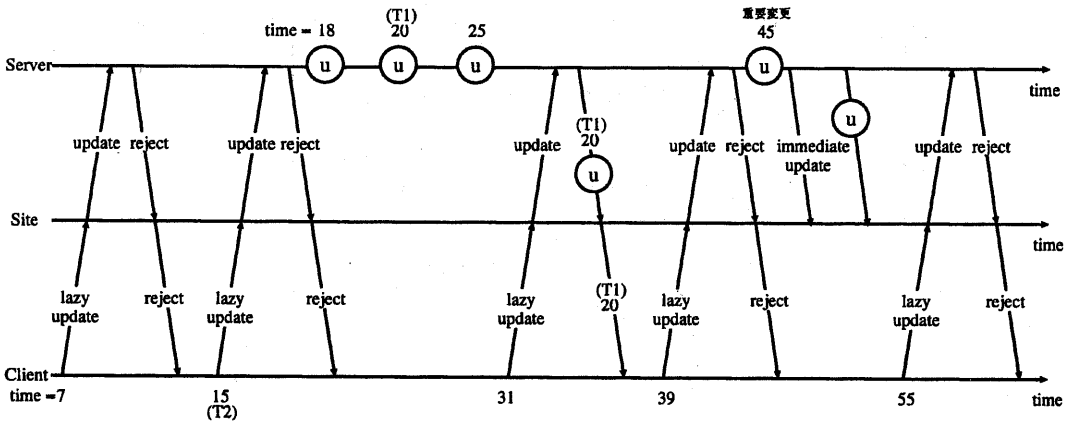

図 4: 複製更新シーケンス

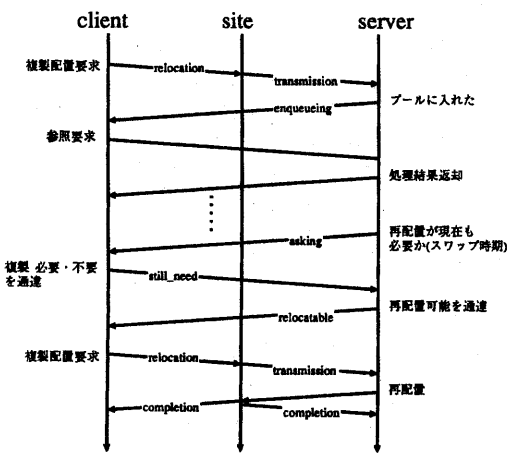

図 5: 予約シーケンス

4 関連研究との比較

4.1 CgR との比較

本研究に近いものとしては Primary Server(PS) を中心に Replicated Server(RS) を階層的にもつ仕組みを持つ CgR(Caching goes Replication)[7] が提案されている。CgR ではクライアント (ユーザ) は Client-side Proxies(CP) を通じて PS、もしくは RS にアクセスする。CP は WLIS など [8] のネーミングサービスにより RS のロケーションを知ることができる。CP は負荷分散などの目的によりアクセスする RS をスイッチしている。

RORP と CgR では、状況に応じてクライアント側から選択的にアクセスするサイトが変わるという点で似ている。しかし、RORP ではクライアントの知るオブジェクトのロケーションはカレントサイトのみであるのに対し、CgR では CP が複数の RS を知っているという違いを持つ。つまり、RORP ではクライアントが適切なサイトに複製を動的に再配置し、それにアクセスする手段を提供するのに対して、CgR は配置された複製を CP を通じてどのようにアクセスするかに重点をしぼっている。また、CgR は意図的に負荷分散されるためサーバ側のメリットが大きいのに対し RORP では、クライアントにとって適切なサイトにアクセスされるのでクライアント側に大きなメリットがあると言える。

4.2 Squid との比較

RORP の再配置機能を、www に応用することが可能である。例えば、参照頻度によってホームページを適切なサイトに配置しておくことができる。www の世界ではトラフィックを軽減するためにプロキシ・キャッシュサーバの運用が普通である。プロキシ・キャッシュサーバの代表的なものとして Harvest[5]、Squid[6] などが挙げられる。Squid は、複数の親プロキシ・サーバを選択的に設定でき、隣接プロキシ・キャッシュサーバも設定できるといった特徴をもつ。このような特徴によりキャッシュ内のリソースの再利用を複数のサーバに分散することを可能にした。また Squid では、クライアントから要求を受けたリソースが存在しないときは隣接するサーバにそのリソースの有無を確認しあればそこからリソースを得ることができ、なければ親サーバさらにその上と多段にキャッシュを行なうことができる。

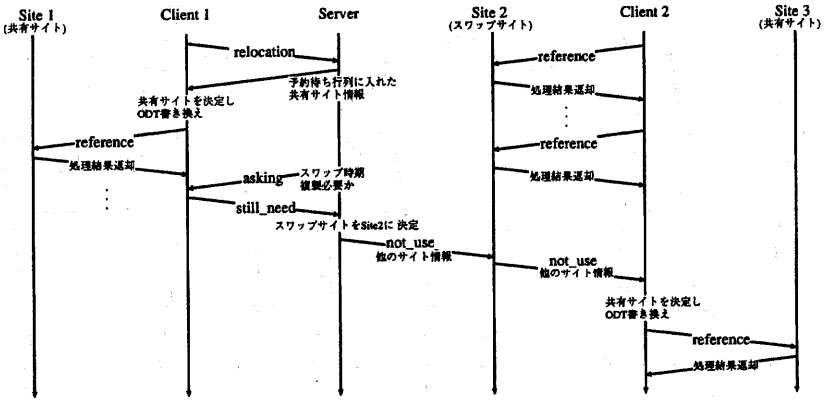

図 6: 共有シーケンス

RORP と Squid では、リソースを複数のサーバに分散するといった点では同じであるがリソースの配置の仕方やアクセス方法などにおいて異なる。プロキシ・キャッシュサーバを用いるのに比べ RORP はリソースの再利用率が低く、参照頻度の低いオブジェクトに関しては遠くのサイトにアクセスするため回答に時間がかかる。しかし、RORP はクライアントにとって適切なサイトにオブジェクト複製を動的に配置することができ、CGI-bin などのプログラムをオブジェクトとして扱うことができるといった利点を持つ。

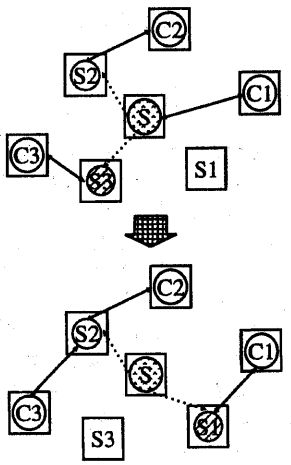

図 7: オブジェクトの共有

5 おわりに

分散オブジェクト指向技術を広域網で用いたときには、通信遅延を無視することができない。したがって、通信遅延をできるだけ減少させるために複製を適切なサイトに配置するプロトコル RORP を提案した。RORP は、複製再配置、複製更新、複製予約、複製共有といった特徴をもつ。現在 HORB[3][4] を用いたプロトタイプを作成中であり、これを用いた機能評価が今後の課題である。

参考文献

[1] "Distributed Object Computing with CORBA" ,November 1997,Available from <URL:http://www.cs.wustl.edu/ schmidt/ corba.html>

[2] "DCOM",November 1997,Available from <http://www.microsoft.com/com/dcom.htm>

[3] 松原 敦, 藤田 憲治,"WWW がもたらす分散オブジェクトの世界", 日経バイト, No.154,1996.

[4] 中原 真則, 平野 聡,"飛べ, オブジェクト！ HORB プログラミングマジック",bit,vol.28, No.10,1996.

[5] Mic Bowman, Peter Danzig, Darren Hardy, Udi Manber, Mike Schwartz, and Duane Wessels,"HARVEST EFECTIVE USE OF INTERNET INFORMATION" <URL:http://harvest.transarc.com/> September 2, 1996

[6] "Squid Internet Object Cache",December 1997,Available from <URL:http://squid.nlanr.net/Squid/>

[7] Fifth International World Wide Web Conference, Paris, France, " Introduction Application-level Replication and Naming into today's Web," May 1996, <URL:http://www5conf.inria.fr/fich_html/papers /P3/Overview.html>

[8] "Object-Oriented Real-Time CORBA Naming Service on Distributed Enviroment",Sungjune Hong, Youngjae Kim, Minsoon Kweon, Dugki Min, Sunyoung Han,ICOIN-12,pp637-640(1998.1).