Xウィンドウシステム連動処理の実現

4Q－7

菊地　秀文，高桑　勇，末永　司
(株) 東芝　府中工場

1．はじめに

X-Window システムでは、クライアントがXサーバに対して、メッセージにより操作を要求する対象をリソースと呼ぶ。サーバ上のリソースは、リソース IDと呼ばれるリソースを識別するためのIDが割り当てられ、クライアントはXサーバへリソースIDを指定し、リソースに対する要求を送る。リソースは複数のクライアントで共有可能である。

また、X-Window システムは、ネットワークを介してクライアント・サーバが接続可能な、ネットワーク透過性という特徴を持つ。

“リソースの共有”、“ネットワーク透過性”という２つの特徴を利用することにより、複数のサーバ上で１クライアント処理の共有操作を実現できる。共有操作とは、複数のホスト上でクライアントを同時に操作できる事を示す。

しかし、この２つの特徴を利用したクライアントの共有操作は、共有開始時の処理性能において問題となる部分があった。本報告では、性能改善の手法として“リソース事前構築による高速化手法について述べる。

2．クライアント共有方式と性能問題

複数のサーバ間でのクライアントの共有操作は、以下のようにして実現できる（図１）。

・クライアントは用意した擬似的なサーバ（以下セッションサーバと呼ぶ）に接続し、リソース構築を含む要求を送受信する。
・セッションサーバは、クライアントを共有操作するネットワーク上の複数のXサーバに接続し、それぞれのXサーバでリソースを構築する。リソース構築時に割り当てられるリソースIDは、サーバにより異なるIDが割り当てられるため、セッションサーバはリソースIDを管理する。
・クライアントからの要求に対し、管理されたリソースIDにより、それぞれのXサーバに対応するリソースIDに変換し、要求を送信する。

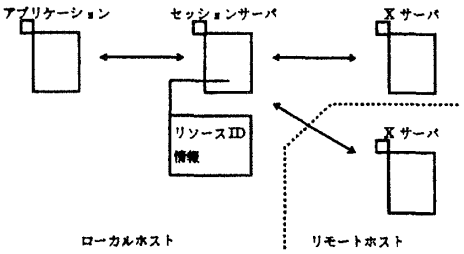

図 1. クライアント共有方式

セッションサーバは、実際のクライアントではないが、Xサーバ側からはリソースIDの変換により擬似的な要求を送受信する１つのクライアントのように見える。

この方式の場合、共有操作するためのリソースは、クライアントを共有するサーバに接続した後でしか構築できないため、クライアントの共有開始時に、共有操作に必要となるリソースを構築する必要がある。しかし、共有開始時には、共有クライアントからリソース構築要求が送信されるわけではないので、セッションサーバがリソースの構築要求を、共有操作するXサーバに対し擬似的に送信することが必要になる。また、セッションサーバは、共有操作するために、クライアントが使用しているリソース情報を管理する必要がある。

共有開始後の処理では、共有操作するXサーバに対し、リソース構築の完了を待つ必要があり、リソース構築が大量に必要なクライアントによっては、リソースの構築処理時間が分オーダーかかるという問題があった。

Implementation of synchronized operation on X-Window system.

Hidefumi KIKUCHI, Isamu TAKAKUWA, Tsukasa SUENAGA
Fuchu Works, TOSHIBA Corp.

3．リソース事前構築による高速化手法

前述のクライアント共有方式では、リソースの構築処理が性能ネックとなっていることが判明しており、リソースを事前に構築する方法に着目した。

セッションサーバがリモート上の X サーバに対して事前作成する方法では、共有させるリモートホストの候補が複数あり、共有開始以前には、処理するリモートホストが不定のため、リソースを事前構築することは難しかった。また、リソースを事前構築するプロセスがネットワーク上にある場合、通信の障害等でリソース資源が開放されてしまう危険性もあった。

そこで、高速化を図るために、共有操作されるクライアントが使用するリソースと同じリソースを構築する疑似クライアントをリモートホスト上で起動することにより、リソースを事前に構築する手法を取った（図2）。疑似クライアントは、リモートホスト上でそれぞれ起動されるため、通信障害によるリソースの開放の危険性も少ない。

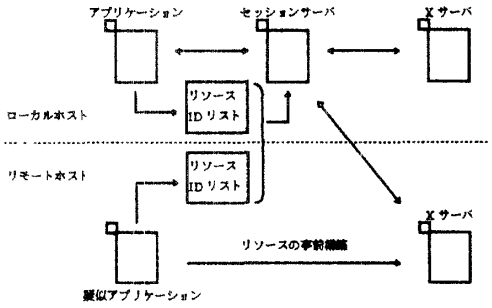

図 2. リソース事前構築による共有処理

この手法により、共有開始時のリソースの構築処理を省略することができ、共有開始時の初期性能の大幅な改善を実現できる。

4．リソース ID の変換手法

リモートホスト上の X サーバに要求を行う場合、それぞれの X サーバに対応したリソース ID への変換を行うことが必要である。

しかし、前述の高速化手法では、リソースを疑似クライアントが構築するため、セッションサーバは変換すべきリソース ID 情報を得ることができない。このため、ローカル上のリソース ID とリモート上のリソース ID を関係付けるための機構が必要となる。

この問題は次の“リソース ID リスト変換方式”により解決する。

＜リソース ID リスト変換方式＞

リソース ID は、サーバ内でクライアントを識別するための BaseID 部と、クライアント内でリソースを識別するための MaskID 部の2つから構成される。MaskID 部は、クライアント内でリソースが作成される度に1インクリメントされる。

このリソース ID の特性を利用し、以下の方法でローカル上のリソース ID とリモート上のリソース ID を対応付ける。

(1)疑似クライアントで構築するリソースは、実際のクライアントと同じ順番で構築する。

(2)BaseID 部とクライアントの対応リストを、クライアント側、疑似クライアント側で出力する。セッションサーバは2つの対応リストを比較し、ローカル側の BaseID 部を、対応するリモート側の BaseID 部に変換する。

(1)の方法により、クライアント内と疑似クライアント内では、対応するリソースに対し同じ MaskID 部が割り振られることになる。つまり、セッションサーバは(2)の BaseID 部のみ変換するだけでリソース ID の変換を行うことができる。

5．制限

疑似クライアントを利用したリソースの事前構築方式では、高速化を実現した一方で、以下のような制限を伴う。

・共有開始後に動的にリソースを構築するようなクライアントは共有操作できない。

これは、動的にリソースを構築するクライアントの場合、疑似クライアントが、動的に構築されるリソースを事前に構築できないためである。

6．おわりに

リソースの事前構築という方式を用いることにより、クライアントによっては、共有開始時の処理が数分オーダーかかっていたものが、数十秒のオーダーに改善された。