LB-7 *Tender* における粒度が小さい資源の位置透過な操作方式

Location-transparent Operation Method for Fine-grained Resource on *Tender*

石井 陽介† 谷口 秀夫‡

Yousuke ISHII† and Hideo TANIGUCHI‡

1 はじめに

分散環境では，各計算機上の計算機資源を位置透過に，かつ，効率よく操作できる方式を提供することが必要不可欠である．本稿では，我々が開発している ***Tender*** [1] における粒度が小さい資源の位置透過な操作方式について述べる．

2 位置透過な資源操作

2.1 要求条件と実現方針

分散環境において，計算機資源を位置透過に利用可能にするためには，次の条件を満たす必要がある．

（条件１）システム内の資源を一意に識別できる．

（条件２）システム内の全資源を同一形式で操作できる．

（条件１）を満たすためには，システム内に存在する全資源に対して，文字列による名前（資源名），または，数字による番号（資源識別子）を付与する方法がある．資源名を用いる場合は，応用プログラム（AP）を記述する際の利便性が高い．資源識別子を用いる場合は，オペレーティングシステム（OS）内部で操作資源を特定する処理が速い．ここでは，二つの方法の長所を取り入れるために，全資源に対して，資源名と資源識別子を付与し，その対応関係を管理し，一方を他方へと変換する機能を提供するようにする．さらに，これらの中に，当該資源の場所情報を持たせる．これにより，資源操作時に，対象資源の存在場所を意識しなくてはならないものの，システム内の資源は一意に識別可能になる．

次に，（条件２）を満たすために，資源操作要求を行う計算機（ローカル計算機）内における資源操作のインタフェースと，別の計算機（リモート計算機）上の資源操作のインタフェースを同一形式にする．操作対象計算機の判別には，資源名，もしくは資源識別子の中に含まれる資源の場所情報を利用する．これにより，分散環境における AP の資源操作を，非常に簡易なものとできる．

2.2 実現時の課題

位置透過な資源操作を実現するためには，次の課題への対処を行う必要がある．

（課題１）システム内における計算機の場所情報の管理

（課題２）ローカル計算機内における資源操作の効率化

（課題１）については，計算機の場所情報をシステム全体で共有し，管理する必要がある．計算機の場所情報には，計算機名，計算機番号，およびそれらとシステム内計算機との対応関係がある．計算機の場所情報の管理法には，集中管理法と分散管理法がある．ここでは，計算機の場所情報の追加や更新作業を一元化し，情報の一貫性を保持しやすくするため，集中管理法を用いる．

（課題２）については，ローカル計算機内で資源操作を行う際，リモート計算機に対する資源操作と同様に，対象となる資源の場所情報を指定して操作を要求することになるため，逐一，計算機の場所情報との整合処理が必要となり，処理効率が悪くなってしまう問題がある．そこで，資源操作を要求する際，操作対象資源が，ローカル内に存在するのか否かを陽に指定できるようにする．これにより，資源操作処理は図１のようになる．資源操作要求を受けた際，最初に，対象資源が陽にローカル指定されているかどうかを調べる．陽に指定されている場合は，そのままローカル計算機内で資源操作を行う．そうでない場合は，資源の場所情報と計算機の場所情報との整合を取ることで，当該資源が存在する計算機を判別する．操作対象計算機がローカル計算機であった場合は，対象資源を陽にローカル指定し直して，再度，資源操作要求を行う．操作対象計算機がリモート計算機であった場合は，当該計算機に対して，遠隔手続呼出により処理の依頼を行う．なお，依頼を受けたリモート計算機は，依頼された処理を代行して行う．以上より，ローカル，ならびにリモートの資源を透過的に操作でき，さらに，ローカル計算機内における資源操作の処理効率を改善できる．

3 *Tender* における実現方式

3.1 *Tender* オペレーティングシステム

Tender は，プログラム構造を重視し，OS の操作対象を資源として分離し，独立化させている．例えば，「プロセス」や「プログラム」，「実メモリ」などを資源化している．***Tender*** では，資源を管理しているプログラム部分（資源管理処理部）を独立化させるため，表プログラム構造という機構を持つ．表プログラム構造とは，プログラム部品と，プログラム部品へのポインタを持つプログラムポインタ表からなる．プログラムポインタ表の行要素と列要素は，操作する資源の種類と操作内容に対応している．資源管理処理部は，資源への操作を，資源の生成（open 系），削除（close 系），入力（read 系），出力（write 系），および制御（control 系）の５つに分類し，各々をプログラム部品として実現する．プログラムポインタ表は，***Tender*** 特有の部分である資源インタフェース制御によって管理される．資源インタフェース制御は，プログラム部品の登録，削除，および変更を行う機能を提供し，プログラム部品への呼び出しを制御している．つまり，各資源の操作に必要な資源管理処理部の呼び出しは，資源インタフェース制御へ依頼し，資源インタフェース制御がプログラムポインタ表を用いてプログラム部品を呼び出すようにしている．

資源には，図２に示す資源識別子と資源名を付与する．資源識別子は，資源の場所と種類と同一種類内の通番を情報として有する数字である．資源名は，場所名と種類名と固有名からなる文字列である．場所とは，資源が存在する計算機を指し，その数字や名前はシステム環境設定情報としている．この中には，ローカルを示す番号，ならびに名前を確保しておく．種類については，OS で数字や名前を規定している．同一種類内通番は OS が資源生成時に決定する．固有名は AP が指定する．資源識別子と資源名の変換機能は，資源インタフェース制御によって提供される．

3.2 資源操作方式

Tender における資源操作処理の様子を図３に示す．資源操作は，表１の提供インタフェースを利用して，ローカル計算機の資源インタフェース制御に依頼する．資源インタフェース制御では，引数の資源名，もしくは資源識別子を調べて，当該の要求が陽にローカルを指定しているのか否かの判別を行う．陽にローカル計算機を指定している場合は，要

† 九州大学大学院システム情報科学府

‡ 九州大学大学院システム情報科学研究院

Graduate School of Information Science and Electrical Engineering, Kyushu University

表1 資源操作のための提供インタフェース

操作の種類	形式	機能 (pid:共通引数, 処理要求プロセスのプロセス識別子)
open	open_rsc(rsc_name, pid, args, mod)	引数 args を利用して資源を生成し，操作権 mod と資源名 rsc_name を付与する．
非 open	***_rsc(rid, pid, args)	引数 args を利用して資源識別子 rid で指す資源に***操作を行う．ここでは，*** = “close, read, write, control”．

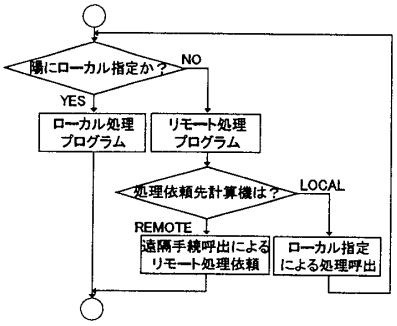

図1 位置透過な資源操作処理の様子

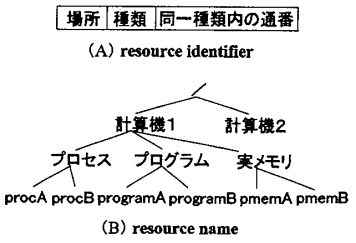

図2 資源識別子と資源名

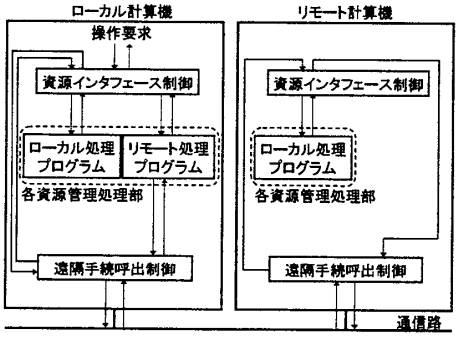

図3 ***Tender*** における資源操作処理の様子

求された処理を行う資源管理処理部のローカル処理プログラムを呼び出し，その戻り値を返す．逆に，陽にローカルを指定していない場合は，要求された処理を行う資源管理処理部のリモート処理プログラムを呼び出す．リモート処理プログラムは，リモート計算機に対して処理の依頼をするため，遠隔手続呼出制御を呼び出す．ここで，***Tender*** における遠隔手続呼出制御は，リモート計算機にある手続きを呼び出す際，データの送受信を行う入出力とのインタフェース整合処理，自計算機内での手続呼出代行処理，および呼出し結果の返送処理を制御している．この遠隔手続呼出制御では，引数の資源名，もしくは資源識別子と，計算機の場所情報との整合を取ることにより，処理の依頼先計算機を判別する．この際，依頼先がローカル計算機である場合は，その処理要求を陽にローカル指定し直して，再度，ローカル計算機の資源インタフェース制御に処理の依頼を行う．逆に，依頼先がリモート計算機である場合は，通信路を介して，当該のリモート計算機に対して処理の依頼をする．リモート計算機では，依頼された処理要求を，ローカルを陽に指定するように変更して，自計算機の資源インタフェース制御に依頼する．資源インタフェース制御では，当該の要求は陽にローカルを指定していると判別し，要求された処理を行う資源管理処理部のローカル処理プログラムを呼び出し，その戻り値を返す．この戻り値は，処理依頼とは逆の手順で，依頼元のローカル計算機に返される．

3.3 評価と考察

表2に，***Tender*** ver.7.1 における資源操作処理時間の測定結果を示す．処理内容は，プログラムの大きさが 4 KB であるプロセス生成処理（open 系），ならびにプロセス削除処理（close 系）とした．また，処理形態は，陽にローカル指定したローカル処理（Ｌ１），ローカル処理（Ｌ２），およびリモート処理（Ｒ）とした．測定環境として，計算機（PentiumII 450MHz）を二台，通信路に Myrinet（1.28Gbps）を用いた．

測定では，各計算機上に計算機の場所情報を持たせ，資源の場所情報との整合処理を行うようにした．したがって，実際には，形態（Ｌ２）と形態（Ｒ）の処理時間には，計算機の場所情報の獲得処理時間も付加される．この点を考慮にいれると，形態（Ｌ１）の処理時間は，形態（Ｌ２）の処理時間と比べて高速になることがわかる．また，各処理において，形態（Ｌ１）と形態（Ｌ２）の処理時間の差と，形態（Ｌ２）と形態（Ｒ）の処理時間の差に違いがある．この差

表2 資源操作処理時間（単位 [μsec.]）

処理内容 \ 処理形態	（Ｌ１）	（Ｌ２）	（Ｒ）
プロセス生成	1,801	1,810	2,533
プロセス削除	281	285	966

は，資源操作時に行う資源識別を，資源名，あるいは資源識別子のどちらを用いるのかということに起因している．

4 関連研究

分散 OS Sprite[2] は，位置透過なネーミング機構を実現しており，ファイルシステムと入出力デバイスを透過的に利用できる．また，プロセス移動も実現している．しかし，大域的なプロセス識別子が存在しないため，位置透過なプロセス操作を行うことは難しい．また，V[3] は，プロセス管理やメモリ管理などを，各計算機上に存在するカーネルサーバと呼ばれるサーバの形で実現している．資源は各サーバ内で管理され，資源操作は，プロセス間通信を用いて当該サーバに依頼する形式を用いている．しかし，管理する資源の粒度が大きく，AP が細かな資源操作を行うことは難しい．

Tender では，資源の粒度が小さいので，細かな資源操作を実現できる．また，仮想記憶空間や実メモリのように，従来，識別や操作が制限され，プロセスの存在を前提に存在していた資源を，個別に生成し，操作することを可能にしている．さらに，資源インタフェース制御を介することで，AP による細かな資源操作を可能にしている．ただし，資源の細分割を行っているため，処理効率の低下が懸念される．しかし，ローカル処理の効率化，ならびに資源の事前用意や保留による資源の生成や削除に伴う処理の高速化を行うことにより，処理効率の低下を防いでいる．

5 おわりに

Tender における粒度が小さい資源の位置透過な操作方式について述べた．残された課題として，システム内で計算機の場所情報を管理し，共有して利用する機構の実現がある．

参考文献

[1] 谷口 秀夫, 青木 義則, 後藤 真孝, 村上 大介, 田端 利宏: “資源の独立化機構による ***Tender*** オペレーティングシステム,” 情報処理学会論文誌, Vol.41, No.12, pp.3363–3374 (2000)

[2] Ousterhout J., Cherenson A., Douglis F., Nelson M. and Welch B.: “The Sprite Network Operating System,” *IEEE Computer*, Vol.21, No.2, pp.23–36 (1988)

[3] Cheriton D. R.: “The V Distributed System,” *Communications of the ACM*, Vol.31, No.3, pp.314–333 (1988)