B-014

マルチコア *Tender* のコア間遠隔手続呼出における代行プロセス処理の高速化
Fast Proxy Process Processing in Inter-core Remote Procedure Call on Multi-core *Tender*

藤戸 宏洋† 山内 利宏† 谷口 秀夫†
Hiromi Fujito Toshihiro Yamauchi Hideo Taniguchi

1. はじめに

マルチコアプロセッサを搭載した計算機において OS の性能を向上させるためには，排他制御オーバヘッドを小さくすることが重要である．そこで，***Tender*** では，OS カーネルをコアごとに配置し，OS が制御し管理する対象である資源をコアごとに管理し，自コアが管理する資源を排他制御無しで利用する方式を実現している．この方式では，他コアが管理する資源を利用するときに，コア間の遠隔手続呼出制御（以降，RPCC: Remote Procedure Call Controller）[1]による代行操作が必要であり，その高速化が望まれる．

本稿では，コア間の遠隔手続きを代行処理するプロセスを依頼ごとに生成と消滅させず，代行処理を複数回実行可能にすることにより，代行処理を高速化する方式を述べる．

2. マルチコア *Tender* のコア間遠隔手続呼出

2.1 基本方式

Tender では，OS が制御し管理する対象を資源と呼び，資源の分離と独立化を行っている．たとえば，既存 OS におけるプロセスを，資源「プロセス」や資源「プログラム」に分離している．各資源を操作するインタフェースは資源インタフェース制御（以降，RIC: Resource Interface Controller）により提供されている．

また，***Tender*** では，資源操作権制御（以降，RCC: Resource Capability Controller）により，各資源に対する操作権を制御している[2]．RIC において資源を生成するとき，RCC により資源を生成した所有者と資源に対する操作権を登録する．RIC は，資源を操作する前に，RCC により資源を操作する権限があるか判定する．資源を操作するユーザの特定には，***Tender*** における流れ識別子 flowid を利用している．flowid は OS 動作の可視化のため利用されており，OS 処理開始時に付与されている値である．

Tender では，マルチコアプロセッサ対応方式の一つとして，個別型 ***Tender*** を実現している．個別型 ***Tender*** では，資源をコアごとに管理し，各コア上の OS やプロセスが別のコアが管理している資源を直接操作することを禁止している．これにより，資源操作時のコア間の排他制御を撤廃し，コア数が増加した場合の排他制御オーバヘッドの増加を抑制している．

個別型 ***Tender*** においてコア間で協調して処理を実行するために，各コア上の OS やプロセスが他コアで管理している資源を操作するときには，コア間でメモリを介した RPCC を利用する．個別型 ***Tender*** においてコア間の RPCC を利用する様子を図 1 に示す．図 1 において，RPCC

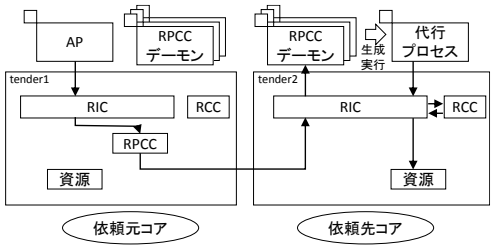

図 1 コア間の RPCC

デーモンとは RPCC により送信されたパケットの受信と受信内容に応じた処理を実行するデーモンプロセスであり，代行プロセスとは代行で資源操作を実行するプロセスである．

RIC に対し他コアで管理する資源の操作を要求すると，RIC は RPCC に対し代行処理を依頼する．RPCC は，資源を管理するコアを特定し，代行処理依頼を送信する．代行処理依頼は RPCC デーモンによって受信され，RPCC デーモンは代行プロセスを生成して実行する．代行プロセスは資源を代行操作し，資源操作結果を依頼元のコアに対し送信後，消滅する．依頼元のコアでは RPCC デーモンが代行資源操作結果を受け取り，RPCC から RIC に代行処理結果が返却され，RIC から依頼元のプロセスに資源操作結果を返却する．このように，個別型 ***Tender*** では RIC と RPCC を介することで，他コアで管理する資源を自コアで管理する資源と同じインタフェースで利用できる．

また，コア間の RPCC により資源操作依頼を送信するとき，流れ識別子 flowid に依頼元のコア情報を付与して送信する．代行処理では，送信された flowid を利用して資源の代行操作を実行する．このため，RPCC による代行処理において，flowid からユーザを特定することが可能である．

2.2 問題点

コア間の RPCC による資源操作において，1 回の代行処理あたり約 0.70ms のオーバヘッドが存在する[1]．コア間の RPCC では，1 回の資源操作の代行処理ごとに代行プロセスを生成実行し，代行処理を完了したプロセスは 1 回ごとに消滅する．1 回の代行プロセスの生成実行時間は約 0.62ms であり，代行処理オーバヘッドに占める割合が大きい．

また，コア間の RPCC による資源操作では，RCC による資源操作権制御の判定をしていない．このため，他コアで管理するよう生成した資源を，他のユーザから操作されないように設定することができていなかった．

†岡山大学大学院自然科学研究科，Graduate School of Natural Science and Technology, Okayama University

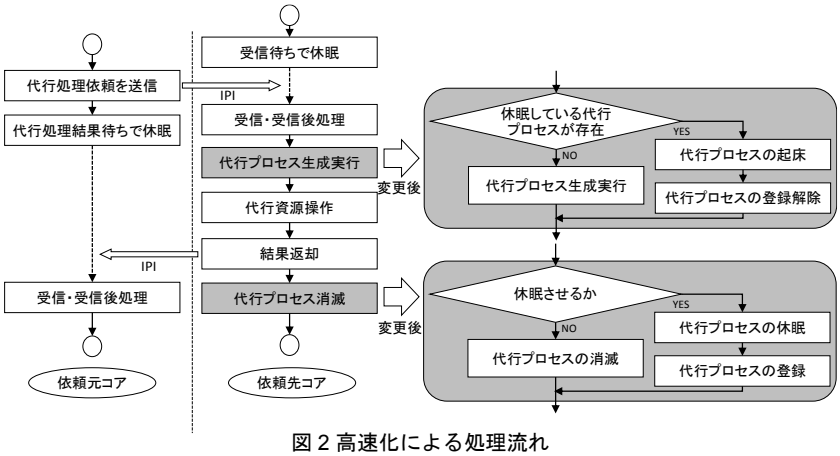

図 2 高速化による処理流れ

表 1 代行プロセス生成時に渡している引数

引数	利用方法
agent_num	受信回数から算出される値により，RPCCデーモンから代行プロセスにRPCCパケットの場所を連絡
flowid	RPCCデーモンのflowidを引き継ぎ，代行プロセスにおいて代行処理を開始するまで利用
machine_num	依頼元の計算機番号により，代行プロセスにおいてRPCCで管理するデータを取得

3. 代行プロセス処理の高速化

3.1 提案方式

代行プロセス生成と消滅の時間を削減し，他コアが管理する資源利用の高速化を目指す．そこで，提案方式では，代行プロセスを代行処理終了後に休眠させ，次に代行処理を依頼されたときに休眠している代行プロセスを利用可能にする．これにより代行プロセスの生成実行回数を減らし，代行プロセス処理の高速化を実現する．

実現における課題を以下に示す．

(課題1) 代行プロセスの生成時に渡す引数

(課題2) 代行プロセスの管理方法

(課題3) 代行プロセスの資源操作権限

(課題 1) について，代行プロセスの生成実行処理を代行プロセスの起床処理に置き換えるため，代行プロセス生成時に渡す引数を代行処理 1 回ごとに渡すことができなくなる．代行プロセス生成時に渡している引数を表 1 に示す．表 1 の引数について，それぞれ対処を行う必要がある．

引数 *flowid* は RPCC デーモンごとに異なり，引数 *machine_num* は依頼元のコアごとに異なる．ここで，RPCC デーモンは依頼元のコアと依頼先のコアの対応ごとに存在し，それぞれ独立して代行プロセスを生成実行している．このため，休眠した代行プロセスについて，それぞれの代行プロセスを生成した RPCC デーモンからのみ利用可能にすることで，代行プロセス生成時に渡した引数 *flowid* と引数 *machine_num* を複数回の代行処理で連続して利用できる．

引数 *agent_num* について，代行プロセスに代行処理依頼を保管している場所を連絡するために利用している．代行処理依頼を保管する場所は受信回数により算出しているため，代行処理依頼 1 回ごとに異なる．そこで，代行処理依頼を保管する場所の決定方法を変更し，代行プロセスのプロセス識別子によって管理する．これにより，引数 *agent_num* を代行プロセスに渡す必要がなくなる．

(課題 2) について，(課題 1)への対処により，休眠した代行プロセスはそれぞれ 1 つの RPCC デーモンからのみ利用可能である．そこで，RPCC デーモンごとに休眠した代行プロセスを管理することで対処する．

(課題 3) について，代行プロセスの flowid を 1 回の代行処理ごとに依頼元から送られた flowid に更新する．また，他コアで管理する資源を生成する代行処理依頼において，生成する資源に対し登録する資源操作権を送信する．これにより，RCC による資源操作権制御の判定を可能にする．

代行処理の高速化による処理流れの変更内容について図 2 に示す．従来方式の代行プロセス消滅処理について，代行プロセスによる資源操作結果返却後に代行プロセスを休眠させる場合は，代行プロセスを休眠させて休眠中代行プロセスとして登録する処理を追加する．また，従来方式の代行プロセス生成処理について，休眠している代行プロセスが存在する場合は，代行プロセスを起床させて休眠中代行プロセスとしての登録を解除する処理を追加する．

3.2 期待される効果

提案方式では，代行プロセス生成実行時間を代行プロセス起床時間と代行プロセス登録解除時間に置き換え，代行プロセスの消滅時間を代行プロセス休眠時間と代行プロセス登録時間に置き換える．プロセスの起床時間と休眠時間はプロセスの生成時間と削除時間に比べ，それぞれ短く，また代行プロセス登録時間と代行プロセス登録解除時間はそれらに比べ十分短いと推察できる．このため，提案方式により，代行処理時間を短くし，他コアで管理する資源利用の高速化が期待できる．

ただし，代行プロセスの消滅については，依頼元コアにおける受信・受信後処理と並行して実行される．このため，依頼先コアにおける負荷が小さい場合，代行プロセスの消滅における処理時間の短縮はほとんど影響がない．

また，提案方式では，RCC により資源の操作権を制御することができる．これを利用して，他コアで管理するよう生成した資源を，他のユーザから操作されないように設定することが可能になる．

4. おわりに

コア間の遠隔手続呼出制御において代行プロセスを複数回利用することにより，代行処理を高速化し，他コアで管理する資源の操作を高速化できることを述べた．この方式では，RCC による資源操作権の制御を利用することにより，コア間の権限移行を行う．

残された課題として，提案方式の実現と評価がある．

参考文献

[1] 藤戸宏洋, 山内利宏，谷口秀夫：マルチコア ***Tender*** におけるメモリを介した遠隔手続呼出制御方式， 情報処理学会研究報告, Vol.2019-OS-145，No.1，pp.1−8 (2019).

[2] 山本淳，谷口秀夫：***Tender*** における統一的な資源操作権制御方式，情報処理学会第 63 回全国大会講演論文集，Vol.2001，No.1，pp.81−82 (2001).