プロセスを共有するためのソフトウェア分散共有メモリの実現

小鍛治 翔太† 芝 公仁†† 岡田 至弘†††
†龍谷大学理工学部

1 はじめに

近年，計算機で処理する内容が複雑化し，扱うデータも大容量化してきている．これにともない，複数の計算機を協調させ処理を行うことが増えてきている．このような計算機間の協調処理で使用される手法のひとつに分散共有メモリがある．しかし，一般的な PC などで使用可能な，専用のハードウェアを必要としないソフトウェア分散共有メモリの多くは，データの共有を目的としており，プログラムを共有することはできない．

我々は，特定のメモリ領域を共有するのではなく，アドレス空間自体の共有を実現し，計算機間でのプロセスの共有を可能とするソフトウェア分散共有メモリシステムを構築した．アドレス空間自体を共有することによって，プロセス内の各スレッドは複数の計算機上で位置透過に動作することが可能となる．本稿では，本システムの構成とアドレス空間の共有を実現する一貫性制御の手法について述べる．

2 システムの構成

本手法ではネットワーク上の複数のノードでプロセスを共有できるソフトウェア分散共有メモリを実現する．本システムは図 1 に示すように複数のノードに置かれるターゲットプロセスを制御する制御プロセスと，制御プロセスがターゲットプロセスを操作するためのメモリ操作機構からなる．ターゲットプロセスとは複数のノードで共有されるプロセスである．

2.1 制御プロセス

制御プロセスはターゲットプロセスのアドレス空間のページをページ毎に以下の状態で管理する．

- NONE … ページ内容を持たず読み書きできない．
- READ_ONLY … ページ内容を持ち，読み出せるが書き込めない．
- READ_WRITE … ページ内容を持ち，読み書きできる．

制御プロセスが管理するページの状態はノード全体で共有される．制御プロセスは各ターゲットプロセスの仮想アドレス空間を順序一貫性 [1] のメモリモデルで一貫性制御を行う．このため本システム上で動くターゲットプロセスは，メモリの一貫性制御を意識することなく動作可能である．

制御プロセスは他ノードとの通信を行う．受け取ったメッセージは必要に応じてメモリ操作機構に通知される．各ノード間では以下の通信が行われる．

- ノードの参加・離脱
- メモリの保護の制御

メモリの保護の制御はノードがページの状態を変える際，別ノードの状態も変える必要があるときに通知される．これは指定したページの内容を全ノードで統一するためである．またターゲットプロセスがアクセス権のないページにアクセスした際，確認が可能となる．

制御プロセスは別のノードから通知された要求に従い，メモリ操作機構を用いてターゲットプロセスのメモリに変更を加える．また制御プロセスはターゲットプロセスを常に監視し，ターゲットプロセスでおこるシグナル，取得する．シグナルを確認した制御プロセスは，必要に応じて別ノードに通信を送りターゲットプロセスの仮想アドレス空間を操作する．

表 1　メモリ操作機構が提供するシステムコール

p_map(pid, addr, len)
p_munmap(pid, addr, len)
p_mprotect(pid, addr, len, plot)

2.2 メモリ操作機構

メモリ操作機構はカーネル内にあり，制御プロセスがターゲットプロセスの仮想アドレス空間を操作するためのシステムコールを提供する．メモリ操作機構が提供するシステムコールを表 1 に示す．

p_map はプロセス識別子 pid で指定したプロセスが持つアドレス空間のアドレス addr からサイズ len までを使用可能な領域として設定する．当該領域のメモリページには書き込み権も読み出し権も与えられない．

p_munmap はプロセス識別子 pid で指定したプロセスが持つアドレス空間のアドレス addr からサイズ len まで設定された内容を使用不可にする．

p_mprotect はプロセス識別子 pid で指定したプロセスが持つアドレス空間のアドレス addr からサイズ len までのアドレス範囲のメモリの保護属性を属性 prot に変更する．また addr はページサイズの整数倍でなければならない．ページの保護に違反するようなメモリアクセスを行おうとするとページフォールトが発生する．

Software Distributed Shared Memory for Sharing Processes
Shouta Kokaji†, Masahito Shiba††and Yoshihiro Okada†††
†Faculty of Science and Technology, Ryukoku University

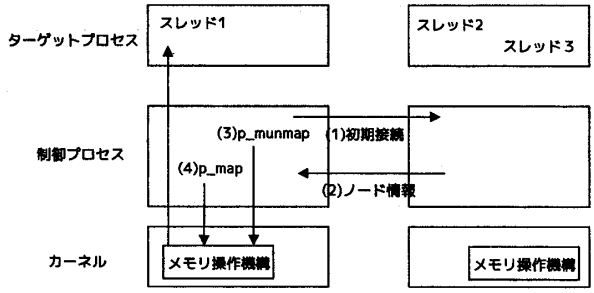

図 1　アドレス空間の共有

3　ページ状態

ページはページ毎にいるページオーナーによって管理され，別ノードの当該ページはページオーナーによって変更される．ページオーナーはそのページの READ_WRITE を持つノード，または最後に所持していたノードである．そのためページオーナーとなるのは接続ノードの中で 1 台のみとなり，1 つのノードが複数のページのページオーナーとなれる．最後に READ_WRITE を持ち，現在では READ_ONLY を持つノードを R_OWNER とする．

ターゲットプロセスの持つメモリのページは常に次の 2 つのいずれかである．

- 1 つのノードのページ状態が READ_WRITE を持ち，残りのノードの当該ページの状態は NONE である．
- 1 つのノードのページ状態が READ_OWNER を持ち，残りのノードは READ_ONLY または NONE である．

これら 2 つの状態によって順序一貫性のメモリモデルが実現される．

4　仮想アドレス空間の共有

仮想アドレス空間の共有は図 1 のような流れで行われる．

(1) 共有を開始したいノードは本システムを用いてプロセスを共有しているノードに接続する．
(2) 分散ネットワークに参加しているノード情報とターゲットプロセスのメモリマップを受け取る．
(3) fork で作った子プロセスに対して p_munmap を行い，マップしているファイルをを全て使用不可にする．
(4) fork で作った子プロセスに対して p_map を行い，ターゲットプロセスのメモリマップを複製する．

(2) において受け取るノード情報とは現在プロセスの共有を行っている全ノードの IP アドレス，ポートである．接続したノードはプロセスを共有しているノード全てに新たに接続ノードが増えたことを通知する．また (4) において有効にしたメモリ領域は全てアクセス禁止状態である．禁止された領域を参照するとページフォールトが起こり，カーネルはこれを制御プロセスに通知する．制御プロセスはこれを受け，書き込み権を持つノードにページの保護属性変更を要求する．要求するページの状態が読み込みであった場合，要求の許可と共に送られたページの内容をターゲットプロセスのアドレス空間にコピーする．許可を得た後は p_mprotect を行い，ページの保護属性をアクセス可能にし，ページフォールトを発生させたスレッドの実行を再開させる．

表 2　ページ遷移の処理時間

遷移前のページ状態	遷移後のページ状態	処理時間
NONE	READ_ONLY	10.62ms
READ_ONLY	READ_WRITE	12.01ms
READ_OWNER	READ_WRITE	6.21ms

5　性能評価

基本性能評価として，ページ状態の遷移を評価した．2 台のノードでページの状態を遷移させ，NONE から READ_ONLY，READ_ONLY から WRITE にかかった時間を表 2 に示す．NONE から READ_ONLY に遷移する際にはページを送るノードが送信バッファに書き込む処理のと，ページを要求したノードが受信バッファをターゲットプロセスに書き込みがあるため，時間がかかっている．また READ_ONLY から READ_WRITE の遷移に時間がかかるのは，ページオーナーを取得してから別ノードにページ無効化要求を送る必要があるため，通信回数が増えている．READ_OWNER から READ_WRITE の遷移は接続している全ノードにページ無効化要求を送るだけであり，通信回数が READ_ONLY から READ_WRITE に遷移するときに比べて 1 往復分少ない．このため処理時間が半分になっていると考えられる．

6　おわりに

本稿ではプロセスを共有可能なソフトウェア分散共有メモリシステムについて述べた．本システムではアドレス空間の共有が可能であるため，単一プロセス内の複数のスレッドを異なる計算機上で同時に動作させることができる．今後は，マルチプロセッサに対応した既存のアプリケーションを本システム上で動作させ，性能評価を行う予定である．

参考文献

[1] Lamport, L.: How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs, *IEEE Trans. Comput.*, Vol. C-28, No. 9, pp. 690–691 (1979).