2016 年電子情報通信学会 総合大会に参加して

玉 井 智 規
Tomonori TAMAI
情報メディア学専攻修士課程　2015 年度修了

1. はじめに

私は，2016 年 3 月 16 日から 3 月 18 日に九州大学で開催された 2016 年電子情報通信学会総合大会に参加した．私はこの大会のコンピュータシステムのセッションで，「複数の計算機間でのユーザプロセスの共有」という題目で発表を行った．

2. 研究内容

近年，プロセッサのマルチコア化，メモリの大容量化といった計算機性能の向上に伴い，並列分散システムの領域が拡大してきている．これを受けて，分散システムの研究が行われている．しかし，これらシステムは，分散システムに最適化したアプリケーションへ変更，もしくは専用のインターフェースが必要となる．そこで，我々は，効率的に並列分散処理を行うためのプラットフォームである，プロセス共有機構というシステムの開発を行っている．本システムでは，実行対象であるアプリケーションへの変更を行わないことを目指している．また，本システム上で複数のプロセスを同時に実行でき，任意の計算機へスレッドを移送することを可能としている．

3. プロセス共有機構

プロセス共有機構は，マルチスレッドアプリケーションを分散処理する．本システムでは，プロセスが持つアドレス空間をネットワークに接続された複数のノード間で共有する．プロセス共有機構は，Linux カーネル上で動作し，プロセスを共有するための 2 つのシステムにより構成される．本章では，その 2 つのシステムについて説明を行う．

プロセス共有機構は，プロセスの共有を行うためにアドレス空間を共有する．プロセス共有機構は，アドレス空間を共有するためにメモリ管理部を持つ．メモリ管理部は，アドレス空間全体を複数のノードで共有するため，アドレス空間の情報を各ノードで同一のものにするよう一貫性制御を行なう．本システムの一貫性制御は，順序一貫性のメモリモデルを実現する．そのため，共有されたプロセスは，メモリの一貫性制御を意識することなく，複数のノード上で動作することができる．共有プロセスがメモリにアクセスを行った際，アクセス対象のページの内容が存在しない場合，ページフォルトが発生し，ページフォルトハンドラからメモリ管理部が実行され，一貫性制御を行う．

メモリ管理部は，ページを単位として管理する．メモリ管理部は，共有プロセスのメモリの状態を，常に以下の 2 つの状態になるよう管理する．

- 複数ノードが読み出し可，すべてのノードが書き込み不可
- 1 台のノードが書き込み可，他のノードは読み書き不可

メモリの読み出しは，同時に複数のノードがページの内容を持ち，参照することができる．また，メモリの書き込みは，1 つのノードでのみ行える．メモリに対して書き込みが行われるとき，他のノードで該当するページを無効化することで，一貫性を保つ．

プロセス共有機構は，プロセスを共有することで，複数のノードにスレッドを分散させ，協調動作させる．プロセスは単一ではなく，複数のプロセスを本システム上で動作させることができる．プロセス共有機構では，ノード，プロセス，スレッドの状況を逐次管理し，効率的にスレッドの分散と協調動作させるために，スレッド管理部を持つ．

スレッド管理部では，通信に必要な IP アドレスやポート番号，また，どのプロセスのスレッドがどこで処理されているかといった，システム全体の分散状況の把握を行うための管理表を作り，情報の管

理を行う．管理表では，ノード，プロセス，スレッドという枠組みで，それぞれに識別子を割り当てることで一意に参照することができる．また，管理表の更新は通信により行われ，すべてのノードで更新情報を共有する．

4. 評価

プロセス共有機構を用いて並列処理アプリケーションを動作させた際の台数効果を評価した．実験環境は表 1 に示す．

評価には姫野ベンチマークという CPU 処理速度測定ベンチマークを使用した．評価には，姫野ベンチマークをマルチスレッドで実装されたものを使用し，計算機 1 台に対して 1 スレッドを割り当て，処理時間を測定した．計算機台数を増やし，スレッド数を変化させたときの処理時間を測定した．姫野ベンチマークの問題サイズは 513×257×257 とした．今回の実験では，プロセス共有機構の性能観測だけでなく，MPI とマルチスレッドとして実装された姫野ベンチマークとの比較評価を行った．評価結果を図 1 に示す．

プロセス共有機構の結果を見ると，アプリケーションの処理時間を短縮していくことができている．また，本システムは，MPI と同傾向の台数効果を得ることができている．しかし，MPI と比べて，大きなオーバーヘッドが見受けられる．これは，プロセス共有機構では，MPI のようにアプリケーションを分散システムに最適化するよう変更を行っていないため，厳密なメモリアクセスを行ない一貫性を保持していることが原因と思われる．そのため，MPI に比べ多くの通信が発生していると考えられる．

表 1　実験環境

カーネル	Linux 3.10.12
プロセッサ	Intel Xeon E 3-1260 L 2.40 GHz
メモリ	4 GB

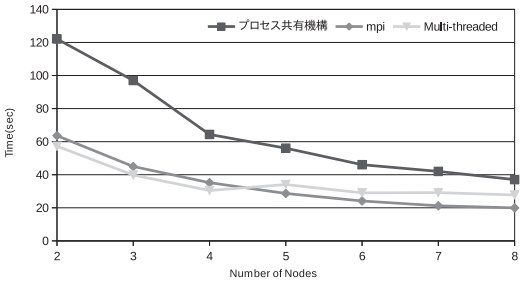

図 1　評価結果

本システムでは，分散環境を意識したアプリケーションへの変更を必要としないため，プログラマは導入コストを抑えて計算機資源を有効利用することが可能となる．

5. おわりに

今回の大会で多くの方々にご意見をいただき，有意義な発表となりました．また，別視点からの意見をいただくことができ，今後の研究に活かしたいと思います．最後に今回の発表にあたりご指導いただいた芝研究室の方々に深くお礼申し上げます．