5J-1

マルチコア・メニーコア混在型計算機における資源管理コア向け OS 間連携機構の試作

深沢 豪[†1]　長嶺 精彦[†2]　佐藤 未来子[†2]　並木 美太郎[†3]
東京農工大学工学部情報工学科[†1] / 東京農工大学大学院工学府[†2] / 東京農工大学大学院工学研究院[†3]

1 はじめに

我々は，x86 アーキテクチャのコアを多数集積したメニーコア CPU[1]を，既存のマルチコア CPU と組み合わせた計算機を対象に，エクサスケールを目指した次世代スーパコンピュータの開発を行っている[2].

CPU のリソースを最大限にタスクへ割り当てるため，タスクの実行基盤を，コア管理に特化した軽量な OS としたい．そこで，マルチコアとメニーコアを，それぞれ「資源管理コア」・「演算コア」とし，ディスク I/O 等の処理を資源管理コア上で動作させる汎用 OS が担当し，演算コアで軽量な OS を動作させ，両 OS が相互に連携し，タスクを実行する環境を構築する.

本研究では，演算コア側から受信した OS 間連携要求を，マルチコア CPU の各コアで並列に処理することで演算コア側の待ち時間を削減する，資源管理コア側 OS 間連携機構を試作し，ラウンドトリップ時間を評価した.

2 OS 間連携の概要

資源管理コア上 OS と，演算コア上 OS との間で行う 2 種類の連携について，以下に概要を示す．また，本 OS 間連携における，資源管理コア側 OS 間連携機構の位置づけを図 1 に示す.

(1) 演算コアで発生した I/O の代行

演算コア上のタスクで発生したディスクや他ノードに対する I/O を，資源管理コア上 OS で処理するために OS 間連携を行う．演算コア側から資源管理コア側に対する，ファイルのオープン/クローズ，Read/Write 要求を可能とする.

(2) 演算コア上プロセスの管理

演算コア上タスクの実行単位である「軽量プロセス」を，資源管理コア側から管理するために OS 間連携を行う．資源管理コア側から演算コア側に対する，実行バイナリの指定，軽量プロセスの生成/終了，および停止/再開要求を可能とする．また，軽量プロセスの実行状態が変化した際に，演算コア側から資源管理コア側に対する通知を行う.

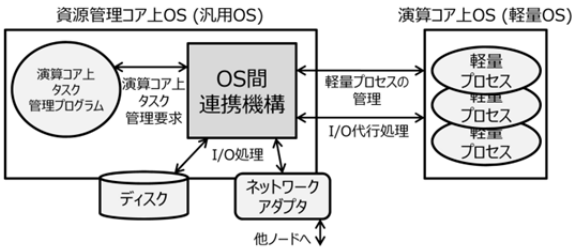

図 1 資源管理コア側 OS 間連携機構の位置づけ

3 本研究の目標

マルチコアとメニーコアではコア数が大きく異なるため，演算コアから資源管理コアに向けて，同時に複数のOS間連携要求が発行された際に待ち時間が生じる恐れがある．本研究では，演算コア側での待ち時間を最低限に抑え，高演算性能が求められるHPCで利用可能なOS間連携機構の実現を目指す.

4 設計

4.1 通信方法

本研究では，マルチコア・メニーコア間での割り込み(IPI)機能，および両 CPU から共に読み書き可能な共有メモリを有する計算機を対象に試作を行うため，通信データの受け渡しに共有メモリを，通信開始のトリガーに IPI を用いる．共有メモリを用いたゼロコピー通信を行うことで，通信遅延の削減を図る．また，共有メモリ上に全二重の待ち行列を構成することで，両コア上の OS 間連携機構間での同期処理を排除する．さらに，共有メモリと IPI を併用することで，ポーリングによる受信側 CPU リソースの消費を抑える．なお，通信の帯域幅を向上させるため，IPI 受信に対する ACK 信号は発行しない.

4.2 OS 間連携プロトコル

OS 間連携を行うにあたり，資源管理コア側 OS 間連携機構と軽量プロセス間でコネクションを設ける．本コネクションは軽量プロセスの生成/終了に付随して発生/消滅する．複数のコネクションが存在する際，共有メモリへのアクセスを排他制御なしで行うために，各々のコネクションで専用の待ち行列を確保する.

本プロトコルでは，「連携要求パケット」，「通知パケット」，および「返答パケット」という単位でデータの送受信を行う．連携要求パケットを受信した場合，返答

Prototyping of Multi-OS Environment on a Resource Management Core for a Multi/Many-core Hybrid Architecture
Go Fukazawa[†1], Kiyohiko Nagamine[†2], Mikiko Sato[†2], Mitaro Namiki[†3]
[†1] Department of Computer and Information Sciences, School of Engineering, Tokyo University of Agriculture and Technology
[†2] Graduate School of Engineering, Tokyo University of Agriculture and Technology
[†3] Institute of Engineering, Tokyo University of Agriculture and Technology

パケットを用いて要求の受理結果を返答するが，処理に時間を要する連携要求に関しては，別途通知パケットを用いて処理の完了を通知することで，非同期に OS 間連携を行うことができる．

4.3 OS 間連携機構の設計

資源管理コア側 OS 間連携機構では，I/O 代行機能と軽量プロセス管理機能を，Linux を用いて実装する．ハードウェア操作と Linux プロセスとの連携を共に行うため，IPI と共有メモリを扱うデバイスドライバ，演算コア上タスク管理プログラムへの I/F となるライブラリに分離した設計とする．軽量プロセス管理機能はデバイスドライバ・ライブラリを共に用いるが，Linux プロセスとの連携を要さない I/O 代行機能は，デバイスドライバ内で処理を完結させることで，特権レベルの切り替えに要する時間を排除する．

本連携機構は，図 2 で示すように，IPI 受信ブロック，I/O 代行ブロック，軽量プロセス管理ブロックの 3 ブロックで構成される．演算コアにおいて，IPI 送信先の CPU コアを分散させることで，IPI 受信ブロック内での IPI ハンドリング処理を並列化する．また，I/O 代行ブロックではファイルの最大取り扱い数分，軽量プロセス管理ブロックでは軽量プロセス数分のスレッドを走らせることで，複数の OS 間連携要求を同時に処理する．各ブロックでの並列処理により，単位時間当たりの OS 間連携処理数を増加させることで，演算コア側の待ち時間削減を図る．

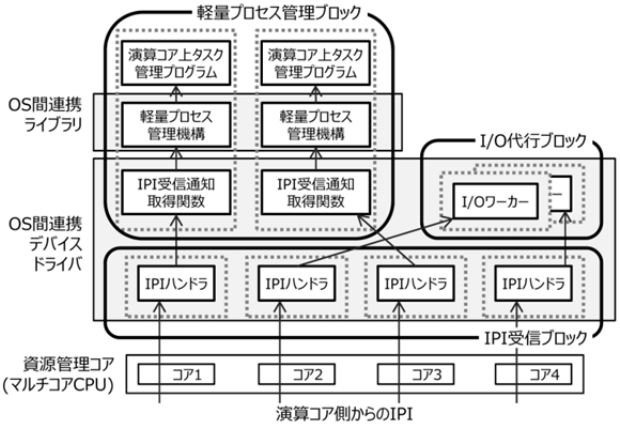

図 2 資源管理コア側 OS 間連携機構の構成

5 実装と評価

資源管理コア側 OS 間連携機構を構成する，デバイスドライバおよびライブラリを Linux 上へ実装した．デバイスドライバからライブラリに対する IPI の受信通知には，ブロッキング I/O を用いた．通知を受け取るライブラリ内のスレッドにて，デバイスファイルに対する I/O 命令を発行しておく．IPI 受信時に，IPI ハンドラから I/O 命令のブロッキングを解除することで，ライブラリに対して低遅延に IPI 受信通知を行う．

演算コア側から資源管理コア側に対する OS 間連携のラウンドトリップ時間を評価するにあたり，Intel Xeon X5690 (6 コア，3.47GHz)を 2 個搭載した NUMA 型計算機上に評価環境を構築した．一方のコアを資源管理コアと見立て，資源管理コア側 OS 間連携機構を動作させた．他方のコアは演算コアに見立て，演算コア中の各コアから同時に IPI を発行する機能，および返答の IPI を受信する機能を備えた評価用ドライバを動作させた．

演算コア側から同時に複数の OS 間連携要求を発行し，資源管理コア側 OS 間連携ライブラリからの返答を受信するまでのラウンドトリップ時間の測定結果を図 3 に示す．同時に要求を発行する軽量プロセス数を 1～6，資源管理コア側 OS 間連携機構の使用コア数を 1，3 とした際の，6 万回試行時の中央値を算出した．

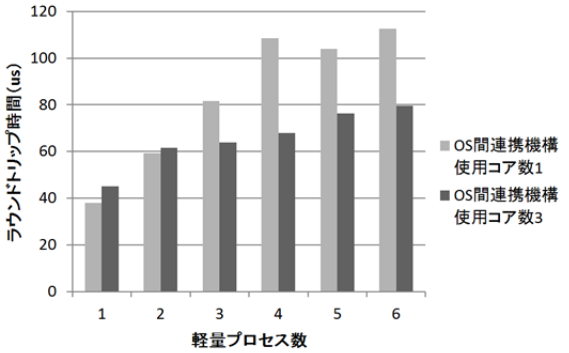

図 3 評価結果

軽量プロセス数が 2，4，6倍に増加した際のラウンドトリップ時間は，資源管理コア側 OS 間連携機構の使用コア数が1の場合は 1.56，2.85，3.00 倍と増大するのに対し，使用コア数が 3 の場合は 1.36，1.50，1.76 倍に抑えられている．これは，資源管理コア側 OS 間連携機構における分散並列処理が，ラウンドトリップ時間の削減に寄与していることを示している．

6 おわりに

本研究では，マルチコア・メニーコア混在型計算機で OS 間連携を行うための資源管理コア側 OS 間連携機構を試作した．OS 間連携要求を分散並列に処理することで，演算コア側の待ち時間を削減した．

今後の課題は，I/O 代行機能の実装と，性能向上に向けた内部構造の単純化である．

参考文献

[1] Intel. “Many Integrated Core (MIC) Architecture - Advanced”. Intel Corp. http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html, (accessed 2012-01-11).

[2] 吉永一美ほか:“メニーコア混在型並列計算機における MPI 通信基盤の提案”，情報処理学会研究報告，2011-HPC-132(5)，1-6 (2011.11)