メニーコア混在型並列計算機向け大域仮想アドレス空間モデル「Multiple PVAS」のためのメモリ管理方式

佐藤 未来子[†1]　深　沢　　豪[†1]　島　田　明　男[†2]
吉　永　一　美[†2]　辻　田　祐　一[†2]
堀　　敦　史[†2]　並木　美太郎[†1]

1. はじめに

スーパーコンピュータの性能は日々向上し続けており，近年ではシステムに搭載するコア数を増加させて高性能化を図っている．本研究でも，マルチコアプロセッサとメニーコアプロセッサを混在させた「メニーコア混在型並列計算機」において，性能向上を目的としたシステムソフトウェアの研究開発を進めている．ノード内の PCIe バスにホスト CPU(Intel 社 Xeon) とメニーコア CPU(Intel 社 Xeon Phi) を接続し，各 CPU に備えるメモリを PCIe 経由でアクセス可能な構成である (**図 1**)．このようなメニーコア混在型並列計算機では，マルチコアとメニーコアの特性に応じたタスク並列化と，軽量なコア間通信が可能なプログラム実行基盤が重要となる．本研究では，ホスト CPU とメニーコア CPU を併用する新しいスタイルで実アプリーションを構築し，メニーコア CPU を有効活用するためのプログラム実行基盤を提案し，実現している．これまでに，メニーコア向け新プロセスモデル PVAS (Partitioned Virtual Address Space)[1]，機能並列によるメニーコア CPU の有効活用を追求する PVAS Agent モデル[2]，それらをメニーコア混在型並列計算機へ適用するための Multiple PVAS[3] を提案している．本発表では，Multiple PVAS の概要およびその実現のためのメモリ管理方式について示す．

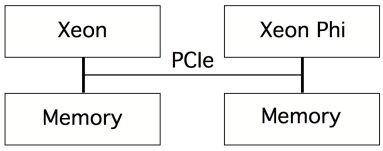

図 1　メニーコア混在型並列計算機の概念図[2]

†1 東京農工大学
Tokyo University of Agriculture and Technology
†2 独立行政法人理化学研究所計算科学研究機構
RIKEN AICS

2. PVAS と Multiple PVAS

PVAS はプロセスとスレッドの中間に位置する新しいタスクモデルである．PVAS においてコアを割り当てる実行実体であるプロセスを “PVAS Task” と呼び，**図 2** 中央に示す「PVAS 空間」という一つの仮想アドレス空間に PVAS Task の各アドレス空間を配置し，ページテーブルを共有する．これにより，PVAS Task 間では共有メモリを確保することなく，仮想アドレス参照による Task 間データ共有を可能としている．

Multiple PVAS はマルチコア CPU，メニーコア CPU ごとに管理される PVAS 空間を包括する形へ拡張した大域仮想アドレス空間モデルである．すなわち，図 2 左端の Virtual Address Map に示すとおり，異なる CPU の PVAS 空間を単一の仮想アドレス空間へマップし，Multiple PVAS に属する PVAS Task をすべて同じ仮想アドレス空間において実行可能とする．これにより，Multiple PVAS 空間内の仮想アドレスによる情報の受け渡しが，異なる CPU 上の PVAS Task 間でも行えるようになり，複数 CPU で稼働する PVAS Task や Agent システムを自由に構成できる．

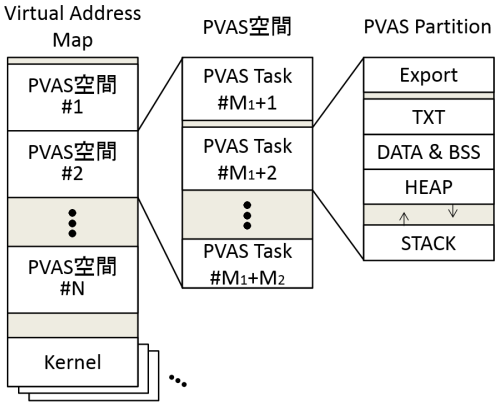

図 2　Multiple PVAS の仮想アドレス空間[3]

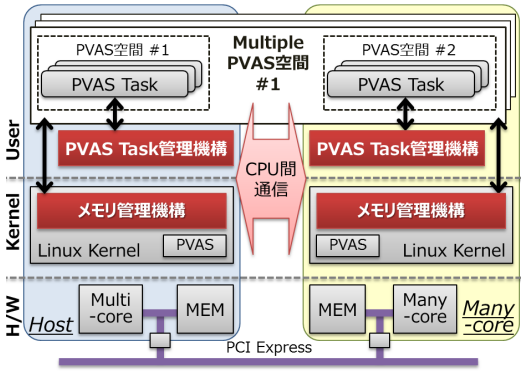

図 3　Multiple PVAS を含む全体構成[3)]

3. メモリ管理方式

図 3 に Multiple PVAS を含む全体構成を示す．マルチコア CPU(ホスト) 側とメニーコア CPU 側において個別の Linux Kernel が稼動し，カーネルに Multiple PVAS の大域仮想アドレス空間を構築・維持するための “メモリ管理機構” を備え，ユーザレベルに PVAS Task の生成等を行う “PVAS Task 管理機構” を備える．

メモリ管理機構で管理するページテーブルの構成を図 4 に示す．Multiple PVAS では，各 CPU の OS は自身の CPU 上の PVAS 空間に対する物理メモリのアサインとページテーブル管理を行い，必要に応じて，リモート CPU 上の PVAS 空間の仮想アドレス変換情報をリモート CPU のページテーブルからコピーする (図 5)．ページテーブルの一貫性維持に関しては，ページテーブルのエントリ (多段ページテーブルの末端) 単位で，CPU をまたいだメモリアクセスが行われたエントリに限定して行う．本設計により，Multiple PVAS 空間全域の仮想アドレス変換情報を管理するオーバヘッドを最小限にとどめている．また，同一 CPU 上の全ての PVAS Task は一つのページテーブルを共有するため，一度物理メモリがアサインされた仮想アドレスは，以降同一 CPU 上の PVAS Task であればページフォルトを起こすことなく，ローカル CPU・リモート CPU 上のいずれの物理メモリへも効率よくアクセスできる．

なお，Multiple PVAS でリモート CPU のメモリをアクセスする場合，CPU 命令で直接アクセスする MMIO 方式でアクセスすることにより，データの一貫性を保障できる．PVAS Task 間で大量のデータ転送を広帯域に実施する場合には，Xeon Phi に備える DMA を用いたデータコピーが有効であると考える．

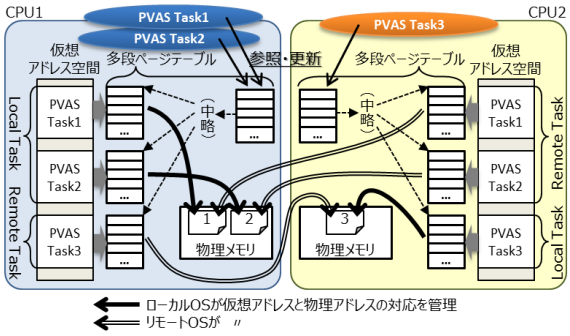

図 4　ページテーブルの構成[3)]

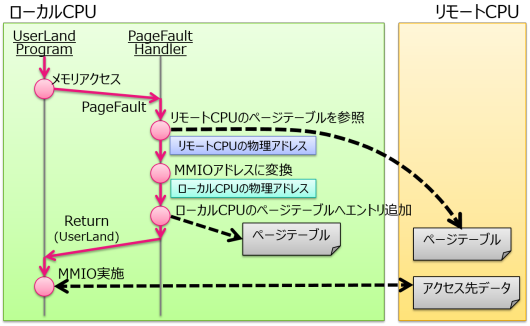

図 5　リモート CPU へのメモリアクセス

4. 実装および今後の展望

Xeon Phi を搭載したメニーコア混在型並列計算機において Multiple PVAS を実装し，Xeon Phi から Host へ 2〜12us 程度，Host から Xeon Phi へ 2〜8us 程度でリモート CPU 上の PVAS Task への処理依頼が可能であるという結果を得た．これらの評価結果については，ポスターにて説明を加える．今後は本指標を元に各種 Agent を設計・開発し，実アプリケーションの実行性能向上を目指す．

謝辞　本研究は，JST CREST における研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」研究課題「メニーコア混在型並列計算機用基盤ソフトウェア」によるものである．

参　考　文　献

1) Shimada, A., et al.: Proposing a new task model towards many-core architecture, MES ’13, pp.45–48 (online) (2013).
2) 堀敦史，他: メニーコア用 Agent プログラミング環境の提案，情処 HPC 研究報告，Vol.2013-HPC-140, No.32, pp.1–8 (2013).
3) 深沢豪，他: メニーコア混在型並列計算機向け大域仮想アドレス空間モデル Multiple PVAS の提案，情処 HPC 研究報告，Vol.2013-HPC-141, No.7, pp.1–10 (2013).