4ZA-2

マルチスレッドアーキテクチャ向けOS「Future」における メモリ管理方式の提案

佐藤未来子　内倉 要　笹田 耕一　加藤 義人　大和 仁典　中條 拓伯
並木 美太郎
東京農工大学大学院工学教育部

1　はじめに

今日，1チップで同時に複数の命令流を扱うことが可能なChip MultiProcessor（CMP）およびSimultaneous Multithreading (SMT) アーキテクチャなどのプロセッサが商用化され，またUnix，Linuxといった汎用OSにおいても，1つの仕事（タスク）を複数の命令流（スレッド）で並列実行する「マルチスレッドモデル」がサポートされており，マルチスレッドアーキテクチャプロセッサの処理性能を活かし細かい粒度で効率よく並列実行させたいという要求が高まっている．

既存のマルチスレッドモデルでは，メモリモデルとして，複数のスレッドが一つの仮想アドレス空間内でコード，データ，スタックを共有することでメモリ共有を達成してきた．しかし，アプリケーションによってはスレッドごとに保護可能なメモリを扱いたいという要求がある．本稿では，同一仮想アドレス空間内で同一アドレスに対するスレッド固有メモリを提供して保護を実現すると同時に，ユーザレベルで軽量にスレッド固有メモリを管理する方式について述べる．

2　従来のスレッド固有メモリ管理方式

Linuxなどで扱うマルチスレッドモデルでは，スレッド固有のスタックやデータ領域を，同一アドレス空間上の別仮想アドレスへ配置しているのが現状である [1]．この場合，スレッド固有メモリが他のスレッドからもアクセス可能となっているため，意図しないアクセスによるエラーが発生する可能性がある．

また、スレッド固有メモリの保護を実現する方式として、河野等が提案するマルチプロテクションページテーブル方式 [2] では，従来のカーネルレベルページテーブルのエントリに各スレッドのページアクセス属性を一元管理することで保護を実現している．また，新井等が提案するユーザレベル記憶管理方式 [3] では，ユーザレベルで仮想アドレス空間を再構築し，この空間上でスレッドを実行することで保護を実現している．いずれの方式も，保護対象の仮想アドレスのページやセグメントをカーネルレベルで管理しているため，システムコールを用いて陽にスレッド固有メモリの保護に関わる設定を指示する必要があり，オーバヘッドが大きい．

3　スレッド固有メモリ管理方式

3.1　目標および設計方針

仮想アドレス空間を共有するスレッドに対してスレッド固有メモリを提供し，これを軽量に管理し保護することを目標としている．以下に設計方針を示す．

- **ユーザレベルスレッドモデルに適したメモリ管理**
  スレッド固有メモリの確保や管理に関わる処理をユーザレベルで行い，スレッド制御時に余分なカーネル遷移が発生しないようにする．
- **スレッド固有メモリの保護**
  スレッド固有に持たせるべきスタックやスレッド固有のデータ領域に，同一の仮想アドレスを割当てることで，意図しないスレッド固有メモリへのアクセスを防ぐ．

3.2　システム構成

図1に筆者等が研究開発を進めているマルチスレッドアーキテクチャ向けシステムの構成を示す．マルチスレッドアーキテクチャプロセッサ，メモリなどの計算機資源を割り当てる単位をプロセスとし，プロセスの資源を共有して実行する実行実体を論理スレッドと定義

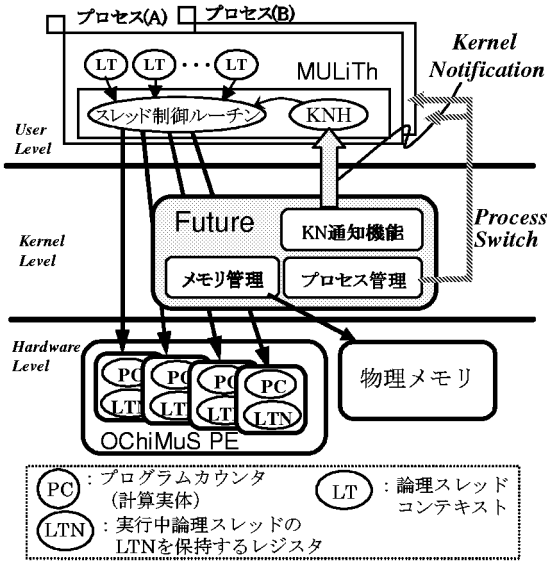

図 1: システム構成図

Memory Management of the Operating System “Future” for On Chip Multithreaded Architecture
Mikiko Sato, Kaname Uchikura, Koichi Sasada, Norito Kato, Masanori Yamato, Hironori Nakajo, Mitaro Namiki
Graduate School of Technology, Tokyo University of Agriculture and Technology

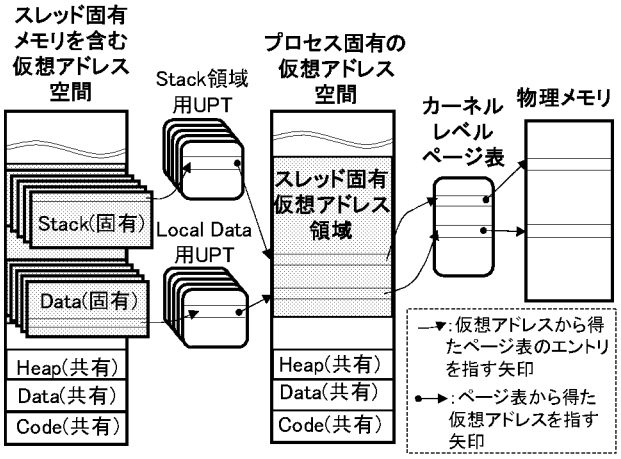

図 2: スレッド固有メモリの仮想アドレス変換の概念図

している．OS **Future**[4] はプロセスを管理し，論理スレッドの制御やコンテキストを全てユーザレベルライブラリ **MULiTh**[5] が管理する．**MULiTh** では論理スレッドのコンテキスト表のアドレスを論理スレッド番号（Logical Thread Number，以下 LTN）と定めており，**OChiMuS** プロセッサ [6] には実行中の論理スレッドの LTN をユーザレベルから設定・参照可能とする機構を備えている．

3.3 メモリ管理の概要

Future はカーネルレベルページ表にてプロセス固有の仮想アドレスに対する物理メモリを管理する．スレッド固有メモリに関しては，図 2 に示すように，プロセス固有の仮想アドレス空間内にスレッド固有の仮想アドレスでアクセスする領域を定めておき，**MULiTh** がその領域内へスレッド固有メモリを再配置する．この対応関係を「ユーザレベルページ表（User-level Page Tabl，以下 UPT)」で保持する．本 UPT は，論理スレッドコンテキストの一部として **MULiTh** がユーザ空間内で管理し，実行中の論理スレッドの LTN から UPT を検索可能とする．

3.4 論理スレッド番号を利用した仮想アドレス変換

スレッド固有メモリの仮想アドレス（Taddr）から物理アドレス（Paddr）への変換は，以下の手順で行う．

1. 論理スレッドの LTN から UPT を参照する．
2. UPT を用いて Taddr をプロセス固有の仮想アドレス (Vaddr) へ変換する．
3. Vaddr をカーネルレベル・ページ表を用いて Paddr へ変換する．

2 重のアドレス変換に伴う管理オーバヘッドに関しては，従来よりプロセッサに備えている TLB に，Taddr から Paddr へのアドレス変換情報をキャッシュしておくことで，本アドレス変換による速度低下が全体性能に及ぼす影響を抑えられると考えている．また，MIPS に見られる TLB のみを備えるアーキテクチャの場合，

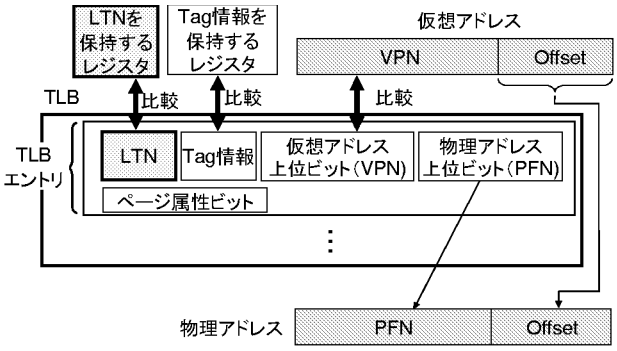

図 3: LTN を含む TLB エントリの構成

TLB の書き換え作業は OS の仕事であり，作業時に参照するページ表の一部がユーザ空間上の UPT であっても実現可能である．

TLB に関しては，空間切替えの効率化のために従来より用いられている TLB タグと共に，LTN を TLB の連想検索に加える．スレッド固有の仮想アドレスであれば LTN と TLB タグを用い，プロセス固有の仮想アドレスであれば TLB タグのみで TLB の連想検索を行う．TLB に LTN を導入することで，スレッド切替え時にスレッド固有メモリのみを軽量に切替え可能とするため，スレッド制御の軽量化にもつながる．

4 まとめ

同一仮想アドレス空間内で同一アドレスに対するスレッド固有メモリを管理する方式について提案した．ユーザレベル・ページ表とカーネルレベル・ページ表の両方を用いるスレッド固有の仮想アドレス変換方式，スレッド番号情報を TLB 連想検索に加えた空間切り替え方式などを示した．今後は，これらの方式評価を進めていく予定である．

参考文献

[1] Ulrich Drepper : ELF Handling For Thread-Local Storage, http://people.redhat.com/ drepper/tls.pdf

[2] 河野 健二他:細粒度保護ドメインのための多重保護ページテーブルの提案と実装, 情処研報 99-OS-81-16, pp. 89–94, 1999.

[3] 新井 克也他:ユーザプログラムによるカーネル外仮想記憶管理, 情処研報 91-OS-53-1, 1991.

[4] 佐藤未来子他:マルチスレッドアーキテクチャ向け OS「Future」におけるプロセス管理, 情報処理学会論文誌, Vol.45, No.SIG 3(ACS5), pp. 38–49, 2004.

[5] 笹田耕一他:マルチスレッドアーキテクチャにおけるスレッドライブラリの実装と評価，情報処理学会論文誌，Vol. 44, No.SIG 11(ACS3), pp. 215–225, 2003.

[6] 河原章二他:システムソフトウェアとの協調を目指すオンチップマルチスレッドアーキテクチャの構想，コンピュータシステムシンポジウム，Vol. 2002, No. 18, pp. 1–8, 2002.