2002年度 卒業論文 発表

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

Implementation and Evaluation of a Thread Library for Multithreaded Architecture

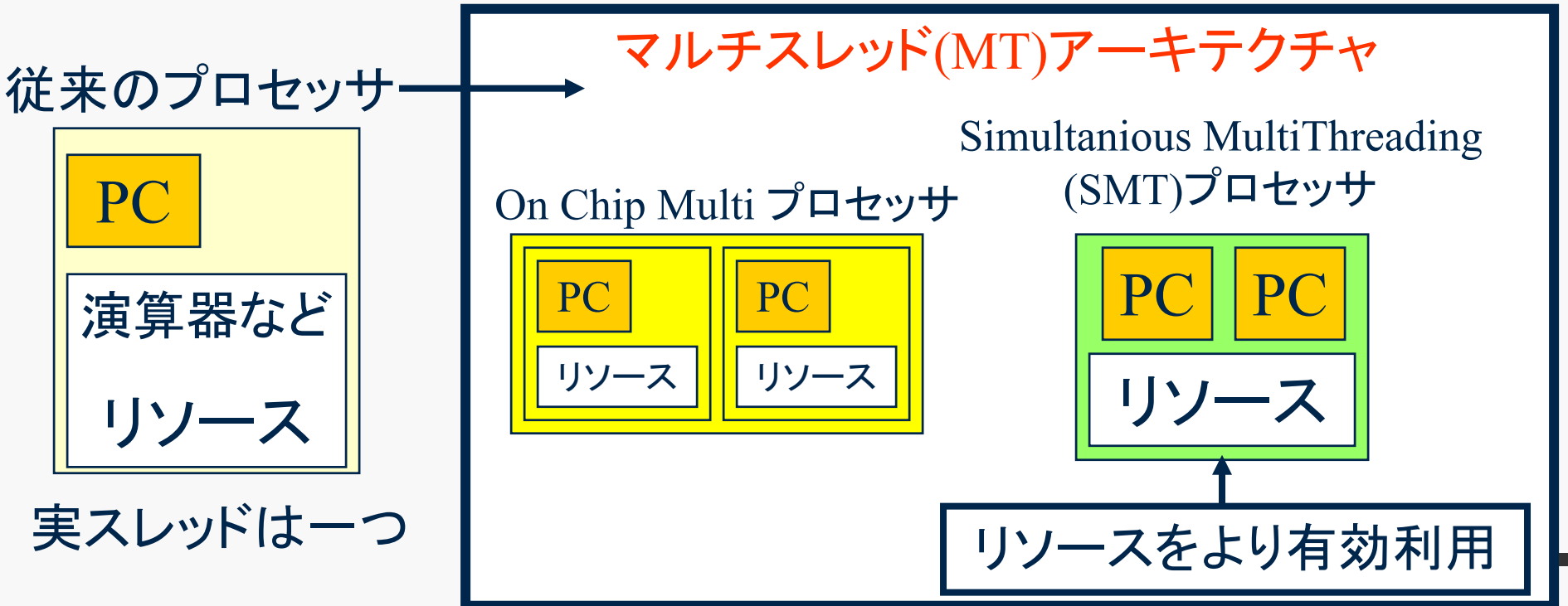
並木研究室

99349031 笹田 耕一

指導教官　並木 美太郎

マルチスレッドアーキテクチャ

- 1チップ上で複数の命令流を並列実行
  - スレッドレベル並列性(TLP) の向上が目的
  - この命令流を実スレッド(Architecture Thread)と定義



従来のスレッドライブラリの問題点

- Intel Xeonプロセッサの場合

  (2つの実スレッド　1.2倍の性能向上)

  - カーネルで実スレッド管理⇒制御に高負荷
  - 複数プロセスが1チップ ⇒ ワーキングセット増大

  ⇒性能低下の原因
- 従来のユーザレベルスレッドライブラリ
  - カーネルスレッドを利用
  - カーネル内でのブロックの問題
- 排他制御・同期機構
  - spin lock は競合が発生

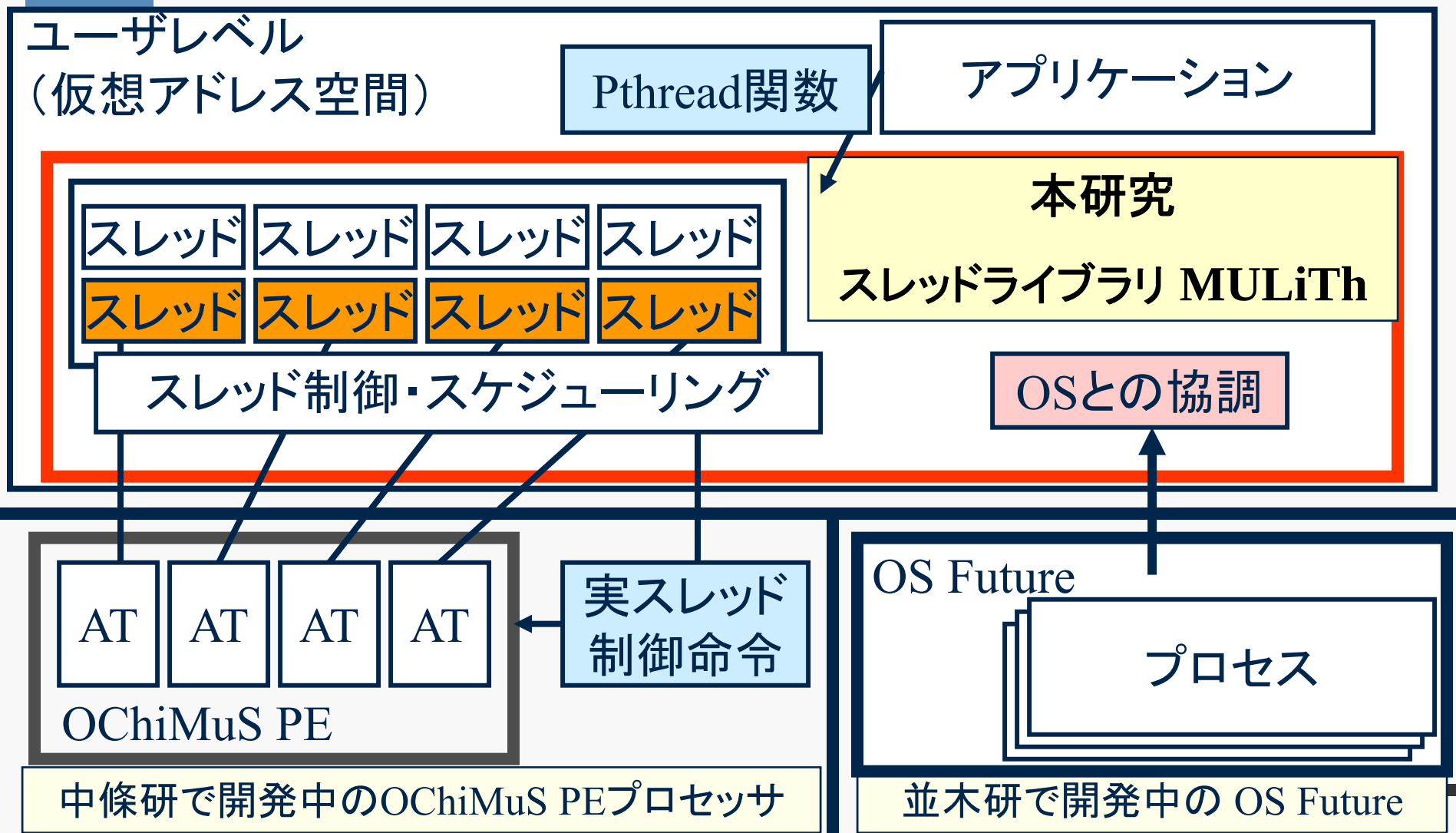
スレッドライブラリの目標

- **ユーザレベルで実行する高性能なライブラリ**
  - 実スレッドに割り当て、並列実行するスレッド
    - 1.5倍の性能向上を目標
  - プロセッサ実スレッド制御命令を利用
    - 軽量なスレッド制御
    - 100サイクル以下のスレッド生成
  - 効率的なOSとの協調
  - 利用しやすいスレッドライブラリ
    - 一般的なインターフェース

スレッドライブラリ MULiTh

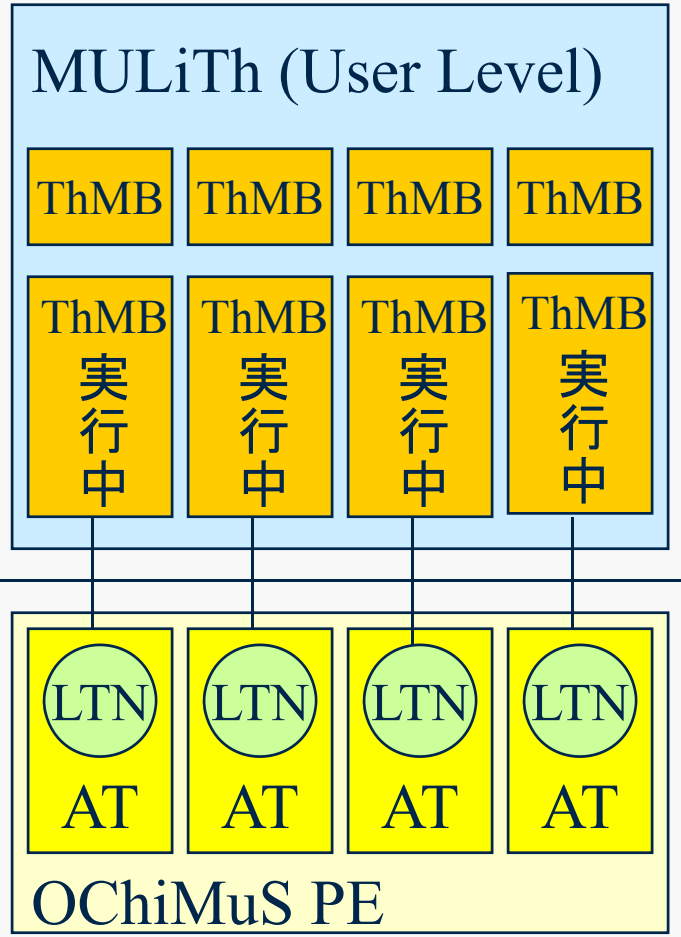
- MULiTh(Userlevel Thread Library for Multithreaded Architecture)
  - POSIX Thread インターフェース
  - 生成・削除・排他制御・同期機構を提供
- 実スレッドに割り当てスレッドを並列実行
- ユーザレベルで動作するスレッドライブラリ
  - プロセッサの実スレッド制御命令を利用
  - 軽量なスレッド制御
- OSと協調し効率的なスケジューリング

システムの全体像


ユーザレベル
（仮想アドレス空間）
Pthread関数
アプリケーション
本研究
スレッドライブラリ MULiTh
スレッド
スレッド
スレッド
スレッド
スレッド
スレッド
スレッド
スレッド
スレッド制御・スケジューリング
OSとの協調
AT
AT
AT
AT
実スレッド
制御命令
OChiMuS PE
OS Future
プロセス
中條研で開発中のOChiMuS PEプロセッサ
並木研で開発中の OS Future

設計：スレッドの管理

- スレッド管理ブロック(ThMB)
  - スレッドがそれぞれ持つ
  - スレッドの情報を保持
- スレッド識別子をThMBの先頭アドレスに
- スレッド識別子をLTNに
  - 実スレッド管理を一元化
  - 実行中スレッド管理不要



設計：スレッドの制御

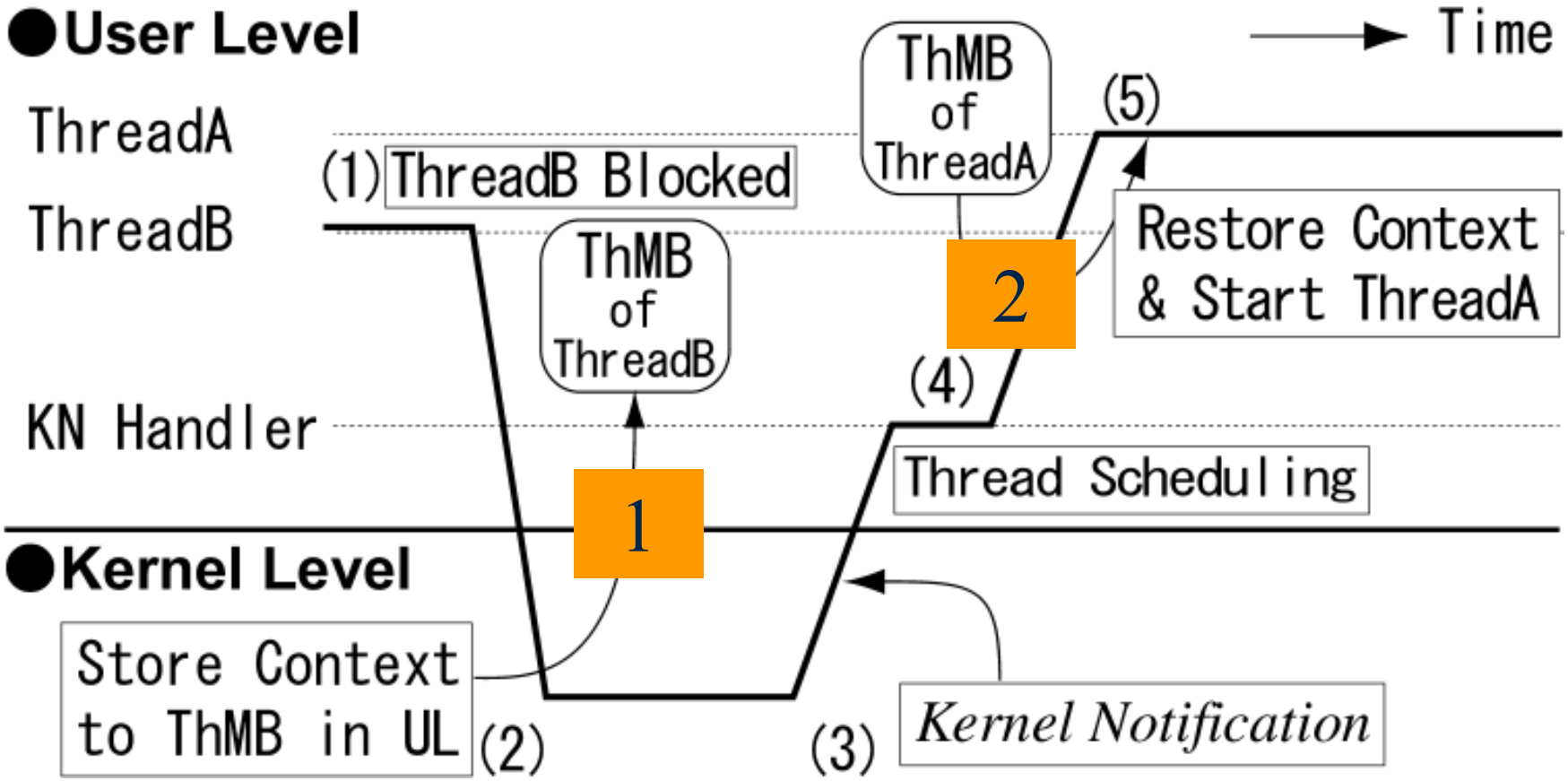
- プロセッサの実スレッド制御命令を利用
  - 並列実行するスレッドを生成
  - メモリアクセスなしでスレッド制御可
  - 命令が失敗したとき、従来のスレッド制御

  →軽量なスレッド生成
- ブロック状態を利用したスレッドの一時停止
  - スピンロックをせず、他実スレッドを実行
  - スレッド切り替えが不要

  →一時停止、解除のコストが低い

設計：OSとの協調(1)

- カーネルでの事象をユーザレベルへ通知が必要
  - I/O ブロック・ブロッキングの解除・シグナルなど
- 事象通知のためのKernel Notification 機構
  - カーネル遷移時、コンテキストをThMB に退避
  - ユーザレベルへの復帰時、ハンドラを起動
- 効率的な通知機構
  - コンテキストのコピー回数が少ない
  - スレッドのプリエンプションを実現可



コンテキストのコピーが2回のみ

- シグナルでの通知は 5回のコンテキストコピー
- Scheduler Activations では最低3回 + Kernel Thread 生成

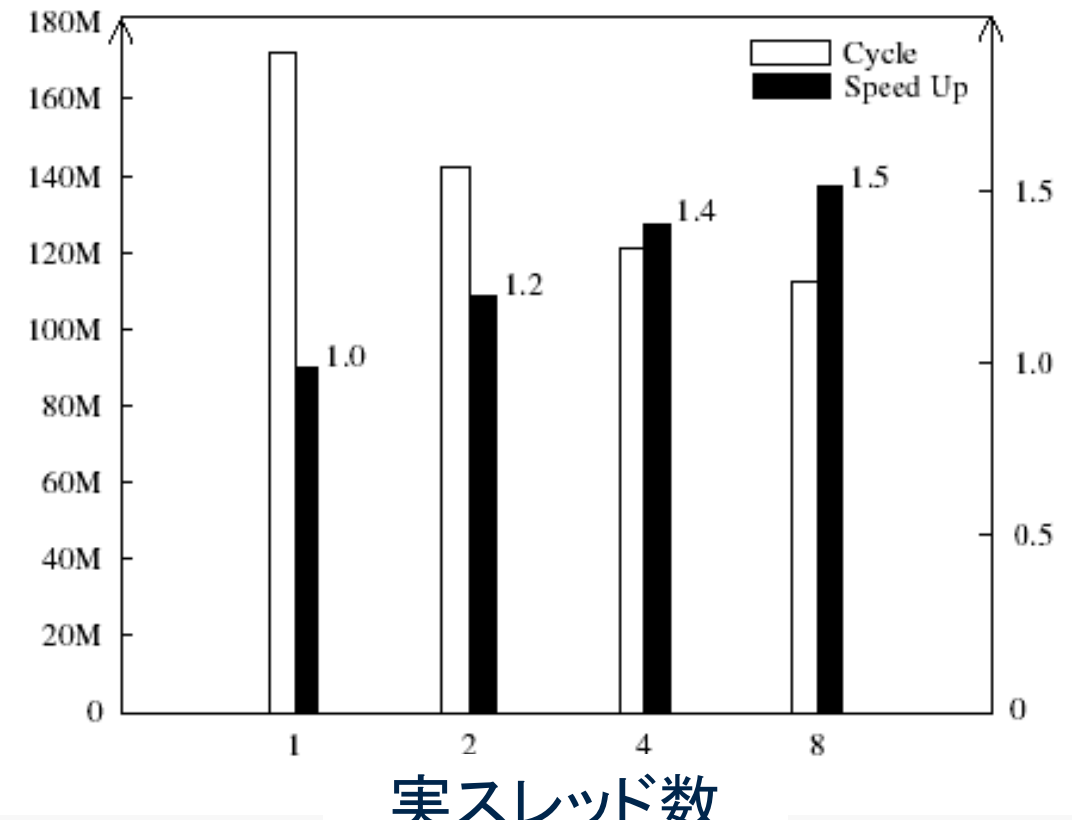
実装と評価

- 実装
  - ライブラリはC言語 10ファイル/ 2000行
  - MIPS アセンブラでの記述が約40個所
    - プロセッサ実スレッド制御命令
    - コンテキスト復帰、退避
    - Test and set による排他制御
- 評価はシミュレータ上で実施
  - MUTHASI(MultiTHread Architecture Simulater)
  - キャッシュ・TLBは利用せず

評価：アプリケーションの性能

並列化した画像縮小プログラム

最長スレッド実行時間(サイクル数)　　速度向上率



評価:スレッド制御の性能

	本研究	従来	速度比
スレッド生成	84	135	1.6倍
スレッド削除	51	223	4.4倍
排他制御	41461	46656	1.3倍
OSからの通知	373	522	1.4倍

単位:サイクル数

評価:考察

- 並列実行により性能向上
  - 実スレッドにスレッドを割り当て並列実行
  - CPUリソースの利用率が向上
- スレッド制御が軽量
  - ユーザレベルでのスレッド操作
  - プロセッサ制御命令を利用
  - 実スレッドブロック状態を利用
- 効率的なOSとの協調
  - 余計なコンテキストのコピーが無い

まとめ：成果

- 並列化により1.5倍の性能向上
- 軽量なスレッド制御
  - スレッド生成が最大1.6倍
- OSと効率的に協調動作
  - 従来の機構に比べ、少なくとも1.4倍の性能
- Pthread仕様スレッドライブラリ
  - 汎用的で利用が容易

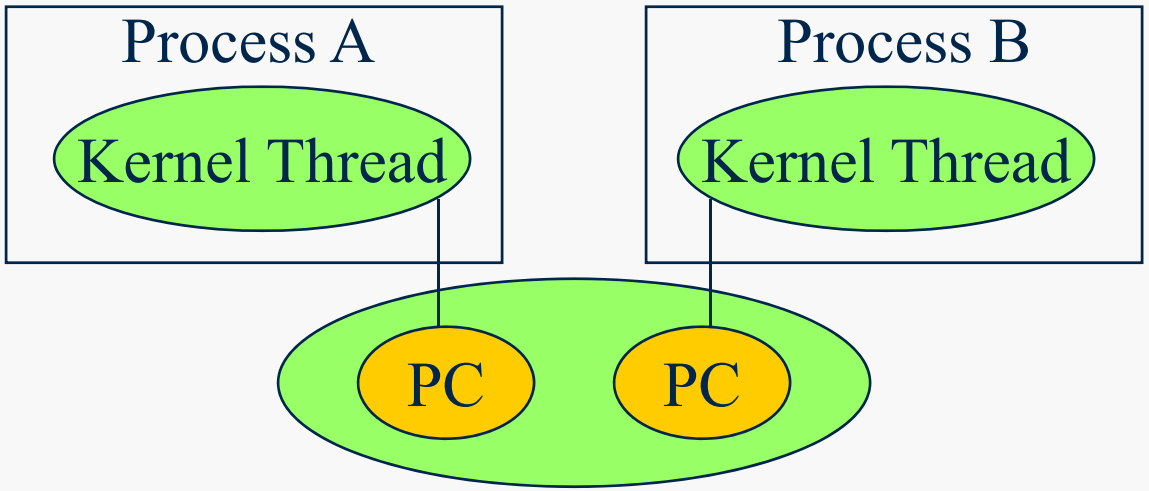
まとめ：今後の課題

- MULiTh の未実装部分の実装
- さらなる評価
  - キャッシュやTLBなどを含めた評価
  - その他のアプリケーションでの評価
- マルチスレッドアーキテクチャの利用法の検討
  - Pthread以外の並列化の検討
  - マルチスレッドアーキテクチャによる言語処理系の検討

実スレッドの管理(1)

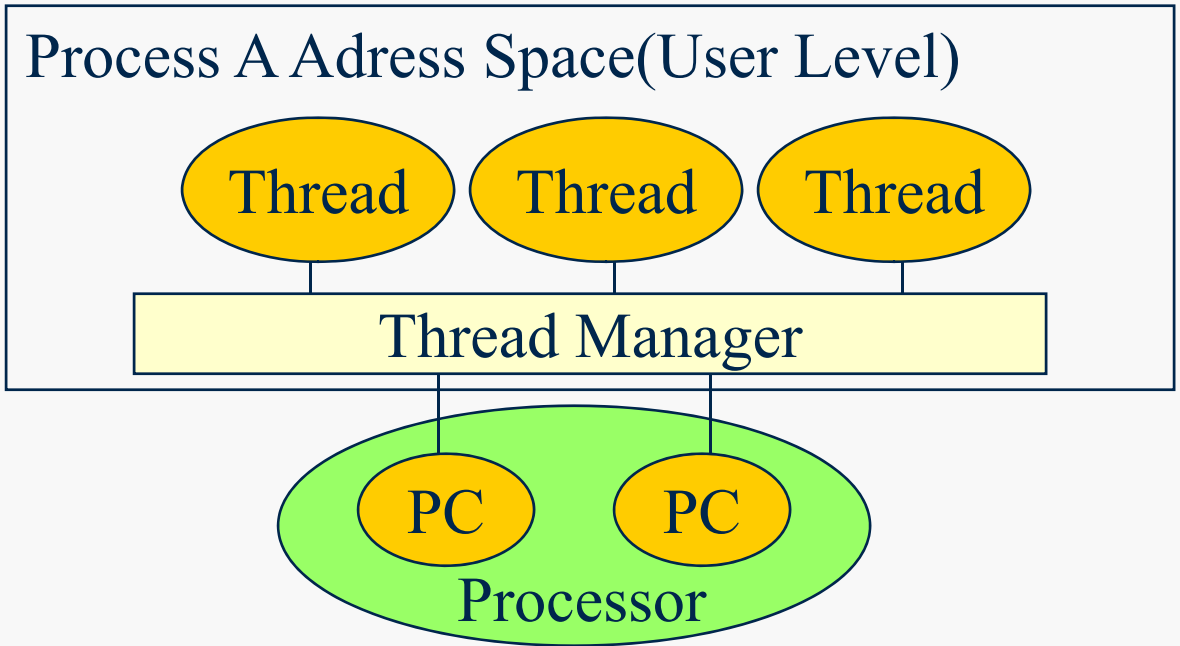
- 従来：OSがカーネルスレッドとして管理
  - 利点：SMP用カーネルが利用可能
  - 短所：ワーキングセット増大

    スレッド制御にシステムコールが必要



実スレッドの管理(2)

- 実スレッドをユーザレベルで管理
  - ユーザレベルで軽量なスレッド制御が可能
  - 専用システムソフトウェアが必要



OChiMuS PE プロセッサ

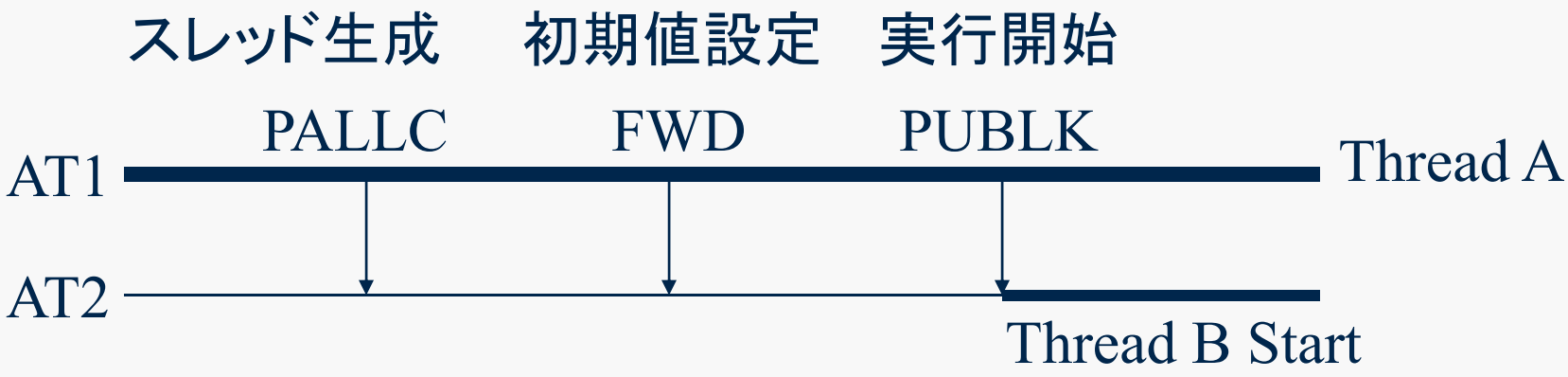
- 中條研究室で研究、開発中
- SMTアーキテクチャ
- 論理スレッド番号(LTN)での実スレッド制御
- 実スレッド制御命令はユーザレベルで利用可能
- 実スレッドは同一アドレス空間を共有

Future OS

- 並木研究室で研究、開発中
- プロセス管理
  - プロセスは OChiMuS PE 用
- スレッドライブラリと協調動作を行う

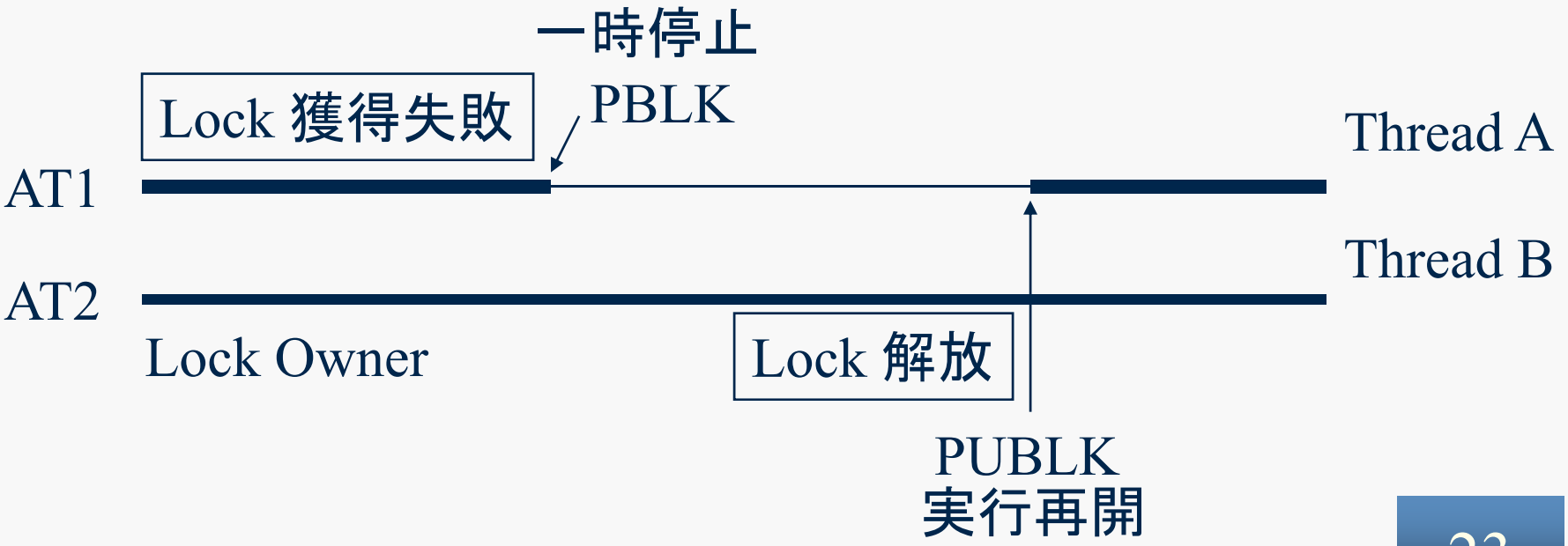
スレッドの制御:スレッド生成

- プロセッサのスレッド制御命令を発行
  - 成功:レジスタアクセスのみで生成可能
    生成したスレッドは即座に並列実行
  - 失敗:生成したスレッドを待ちスレッドに



スレッドの制御：排他制御・同期

- 実スレッドを一時停止する
- ロック解放時実行を再開させる



並列実行の評価詳細

実スレッド数	総サイクル数	向上率	IPC (Instruction per cycle)
1	171M	1.0倍	0.23
2	142M	1.2倍	0.27
4	121M	1.4倍	0.32
8	112M	1.5倍	0.35

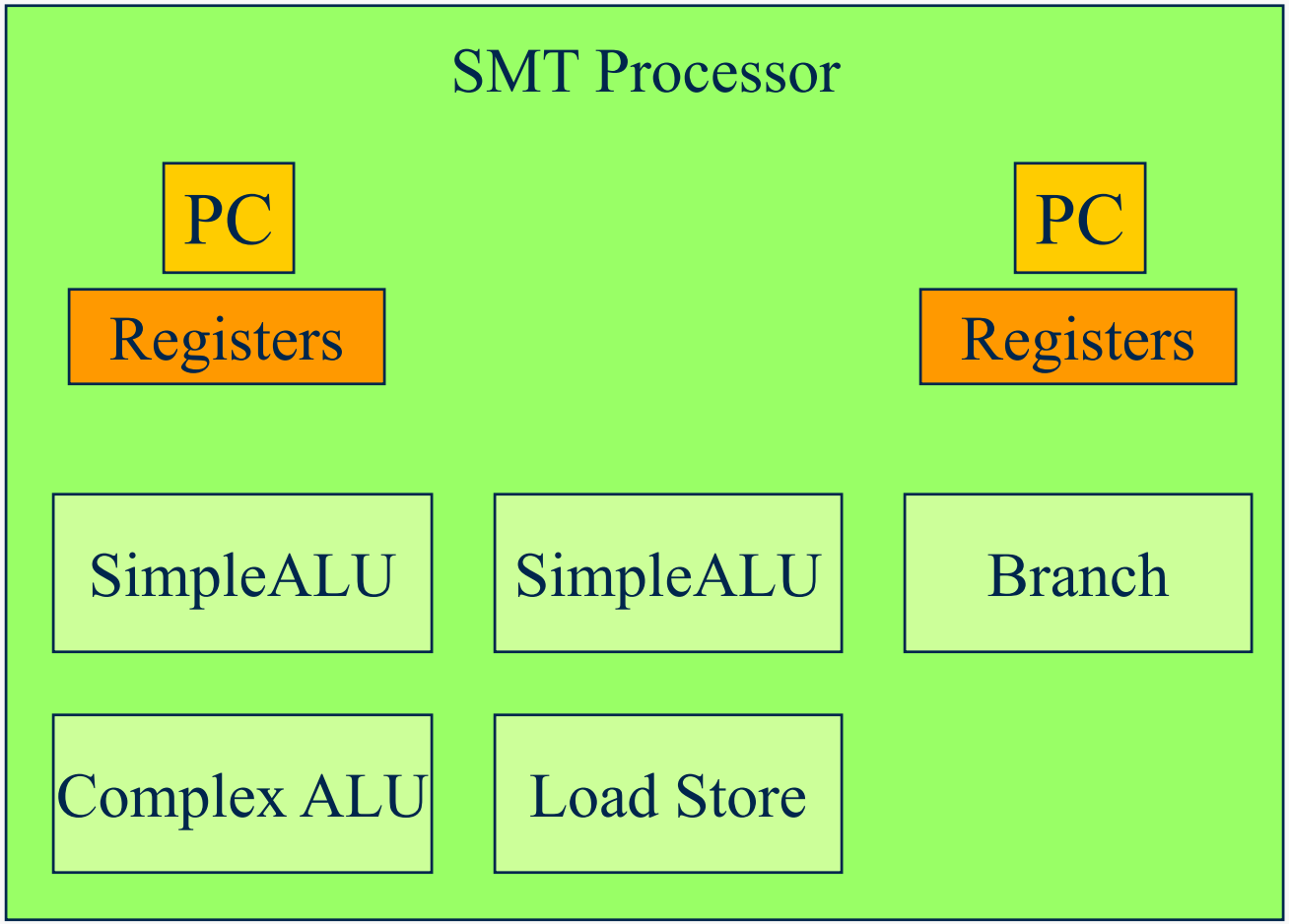
評価環境

- Simple ALU : 2 個
  - 一回の演算は1サイクル
- Complex ALU : 1個
  - 掛け算 12サイクル
  - 割り算 32サイクル
- キャッシュはなし

実装したPthread関数

- pthread_create スレッド生成
- pthread_exit スレッド終了
- pthread_join スレッド合流
- pthread_mutex_lock / unlock 排他制御
- pthread_cond_wait / signal 同期機構

SMTアーキテクチャ


SMT Processor
PC
Registers
PC
Registers
SimpleALU
SimpleALU
Branch
Complex ALU
Load Store