Process-in-Process
新しいノード内並列実行モデル

PCクラスタワークショップ in 柏2017
2月17日（金）
理化学研究所 計算科学研究機構
堀　敦史

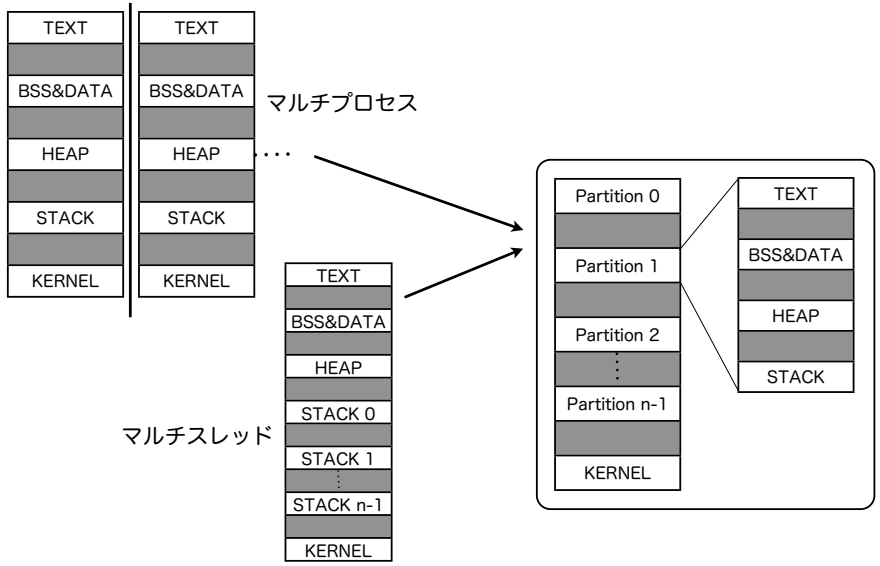
In PCクラスタワークショップ in 大阪2015（2015年2月）

- 新しい並列実行モデル：PVAS（ピーヴァス）
  - プロセスでもない、スレッドでもない
  - プロセス並列とスレッド並列の「いいとこ取り」
- 欠点
  - Linux カーネルを入れ替える必要あり
    - センター運用されている場合は、不可能
- なんとかユーザレベルで実現できないだろうか？

[復習] PVAS [ピーヴァス]

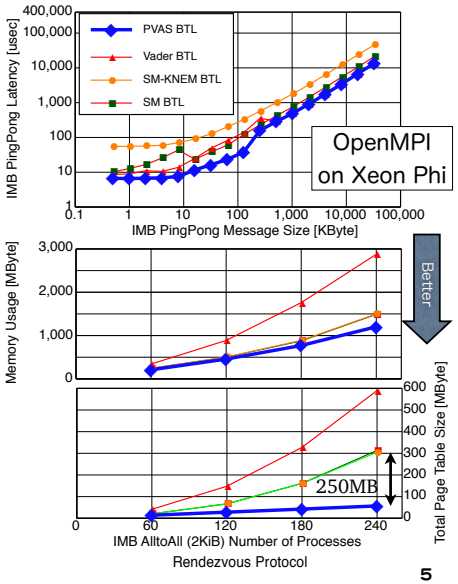
- Partitioned Virtual Address Space
  - プロセスとスレッドの中間的なもの
    - 実行の単位を「PVASタスク」と呼ぶ
  - 変数は全てPVASタスク固有
  - しかしながら、同じアドレス空間なので、共有したければ共有できる
    - 普通に Load/Store 命令でアクセスできるため高速
    - 共有部分のみ、必要なら排他制御

[復習] プロセスとスレッドの中間



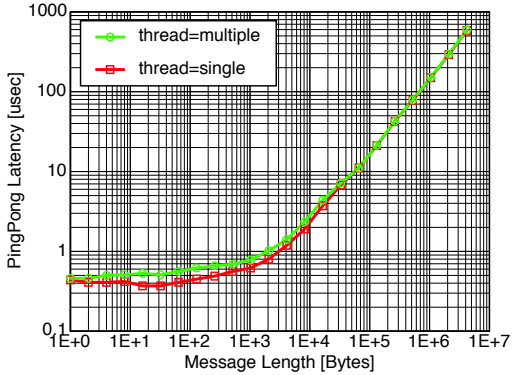
[復習] PVASを用いたMPIの実装

- PVASを用いることで
  ノード内通信の効率化
  - 高速、かつ
  - 省メモリ
- なぜ？
  - 1copyで通信可能
  - ページテーブルが小さくて済む
- CPUコア数が大きい程有利
  - メニーコア



マルチスレッドの問題

- 排他制御のコストが無視できない
  - このグラフは MPICH のコンフィグレーションを変えた時の性能の差



アイデア自体は古くからある

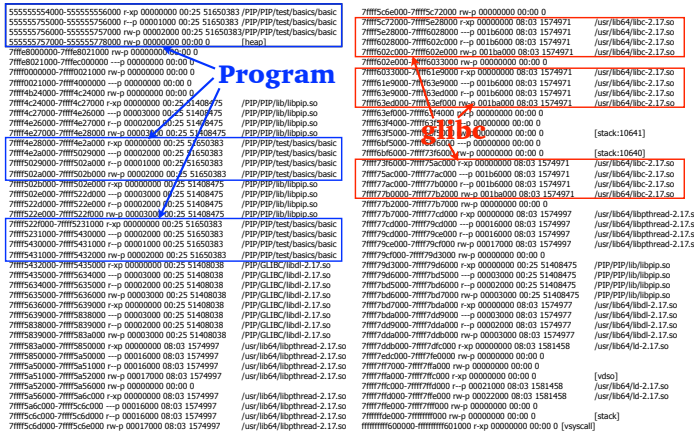
- プロセスからのアプローチ
  - 新しい OS カーネル：Kitten OS/smartmap
  - カーネルモジュール：XPMEM
  - Linux にパッチ：PVAS
- スレッドからのアプローチ
  - コンパイラ／プリプロセッサで、変数を「非」共有にする
  - MPC（仏CEA）
  - Thread-based MPI
    - TOMPI、TMPI、などなど

ユーザレベル実装

- PVAS 同様の機能を Linux の標準機能のみを使って実現できることが判明
  - clone() と dlmopen()
- 「プロセスの中に複数のプロセスがある」
  ➡ “Process-in-Process” (PiP)
  - PVAS に変わる新しい名称
- 以降で、PiP の機能を検証する

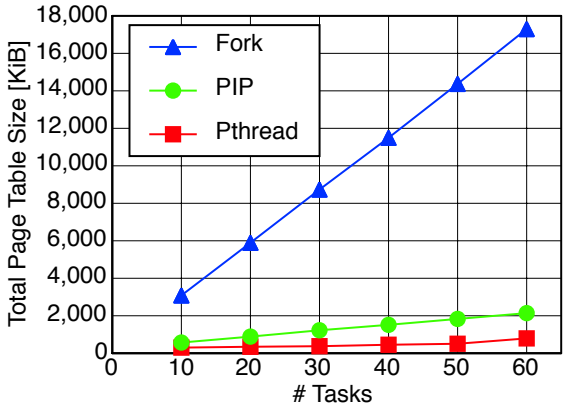
メモリマップ

- PiP で3プロセスを実行した時のメモリマップの例：同じライブラリが複数ロードされている



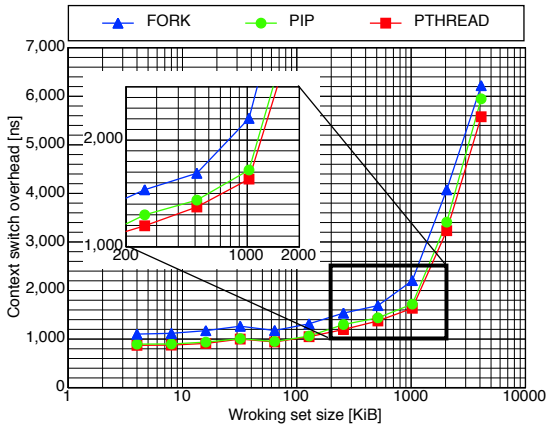
ページテーブルの大きさ

- 128 MiB のデータをノード内のプロセスで共有した時、ページテーブルの大きさを比較



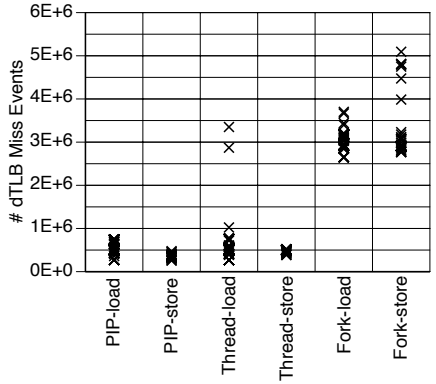
プロセス切り替えの時間

- 同一コア上で、50プロセスを走らせて、プロセス切り替えを発生させる（それぞれ1,000回）



TLBミス回数の比較

- 前ページと同じ条件の時の dTLBミス回数を比較する



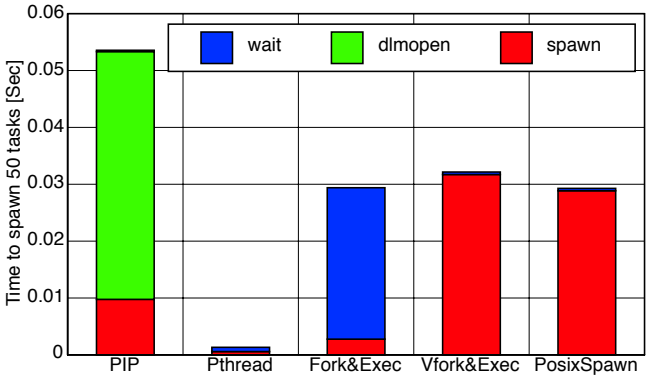
Number of `load_cr3` function calls

PIP	Pthread	Fork
74.1	53.0	794535.4

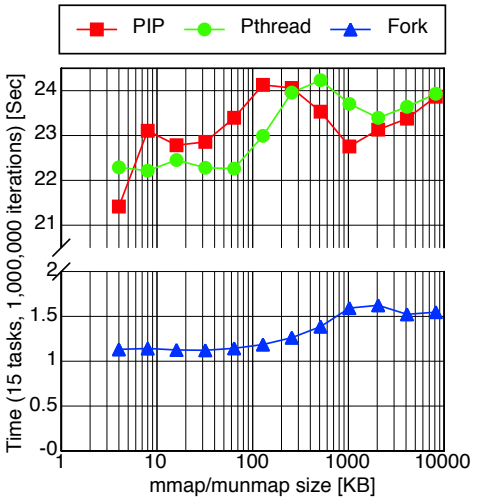
load_cr3関数はLinux（x86）でアドレス空間を切り替える際に呼ばれるカーネル関数
副次的に、キャッシュやTLBが無効になり、オーバヘッドが大きい

プロセス生成の時間

- 同じコア上で15プロセス／スレッドを生成するのに要した時間の比較



mmap/munmap 時間の評価



アプリの実行時間の比較

- PiP上でのアプリ実行で遅くなることはない

PiPのまとめ

Functional View	
Address Space	Thread
PID, wait(), exit()	Process
Signal	Process
File Descriptors	Process*
Session	Process

* depending on clone() flag setting

Performance View	
Context Switch	Thread
Page Table	Process
mmap/munmap	Thread

共同研究

- 米アルゴンヌ研究所
  - Casper + PiP
- 仏 CEA
  - MPC + PiP
- 米テネシー大学
  - Open MPI + PiP

最後に

- 現在 PiP を用いた論文を執筆中
  - 採択されたらすぐに公開（オープンソース）