

恒常的な並列度を維持するフレームワーク  
Cerium Task Manager における  
マルチコア上での並列実行機構の実装

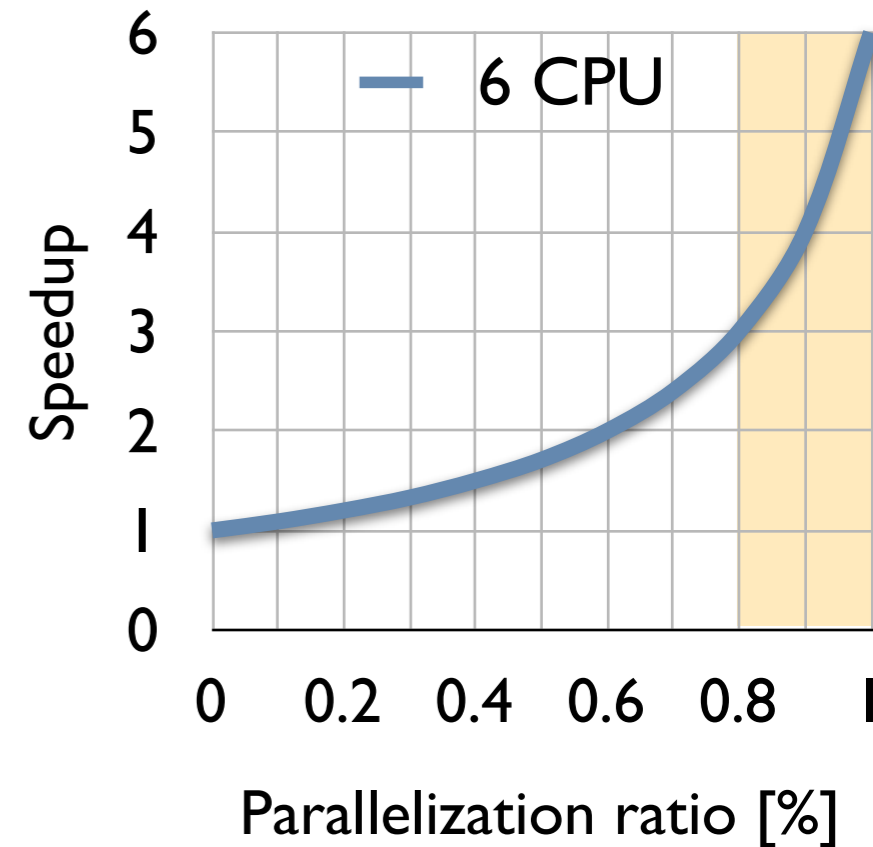
當眞大千

並列信頼研

# ○ 研究概要

処理全体で高い並列度を保たなければ  
処理性能は著しく下がる(アムダール則)

Cerium Task Managerでは  
Taskという単位で処理を管理し  
高い並列度を保つように実行する



本研究ではCell用に開発されたCerium Task Managerを  
マルチコアプロセッサ上での並列実行に対応

# ○ Ceriumのマルチコア環境への対応

---

Cellとマルチコアで同じプログラムが動作する

アーキテクチャの違いを気にせずに  
並列プログラムを書くことが可能に

将来的にはGPGPUにも対応予定

対応にあたって、Cellと一般的なマルチコアの違いを  
考慮して新しいTask Managerを設計した

# Cellとマルチコアの比較

共有メモリか、そうでないかの重要な違いがある

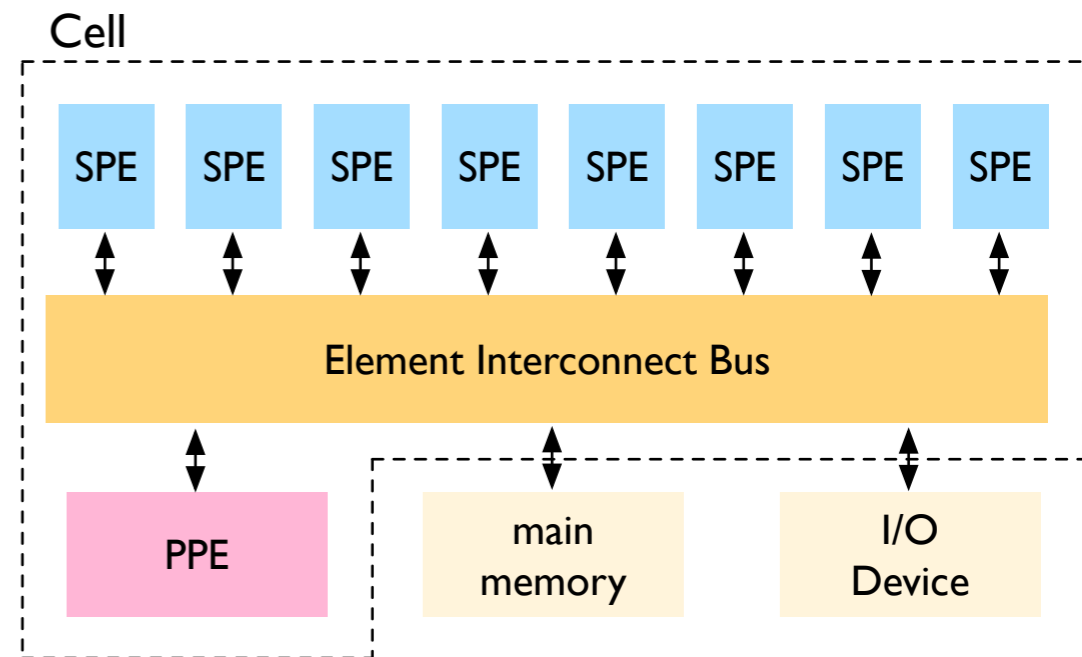
## Cell

**PPE** 汎用コア

**SPE** 演算用コア

SPEからはメインメモリに直接アクセスできない  
**DMA命令を用いて**

データを各SPEに転送する必要がある

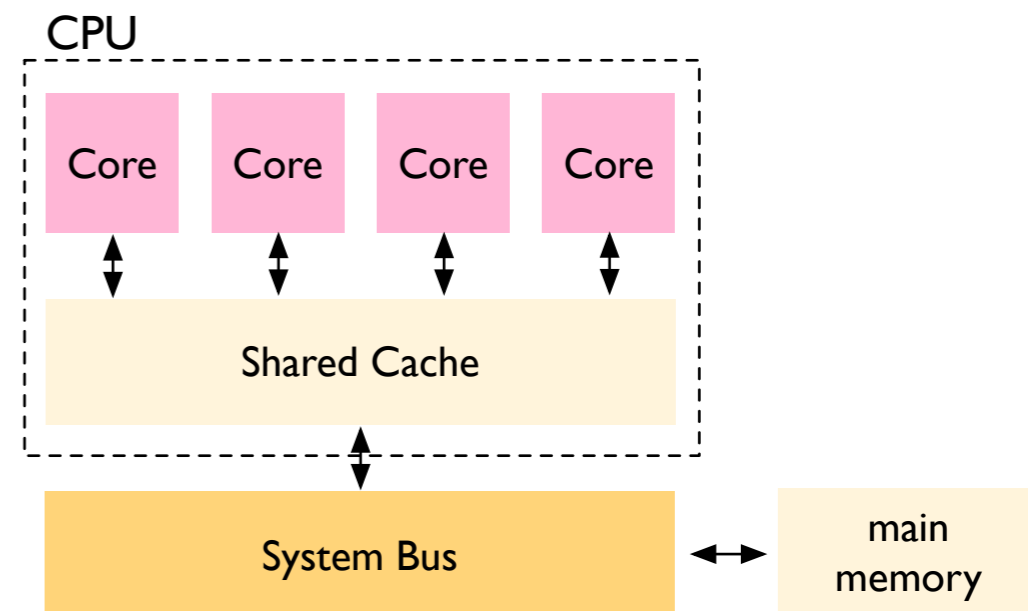


## 一般的なマルチコアCPU

汎用コアを複数持つ

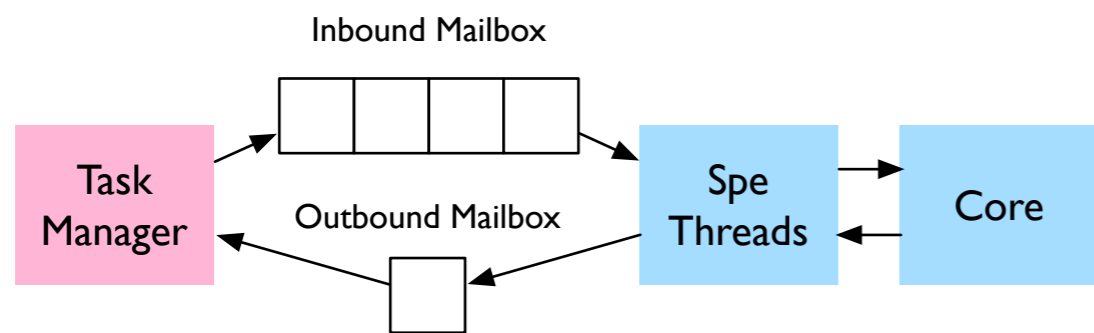
メインメモリに直接アクセスできる

データのコピーは必要ない

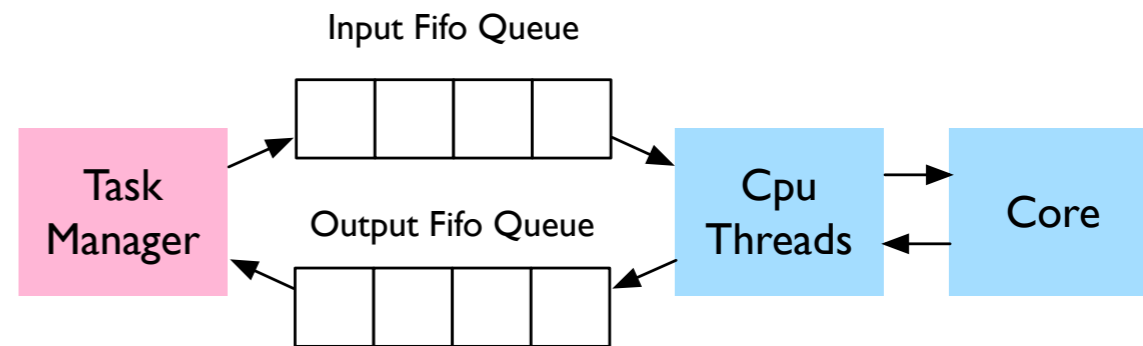


# マルチコア上での並列実行機構

Cell



マルチコア



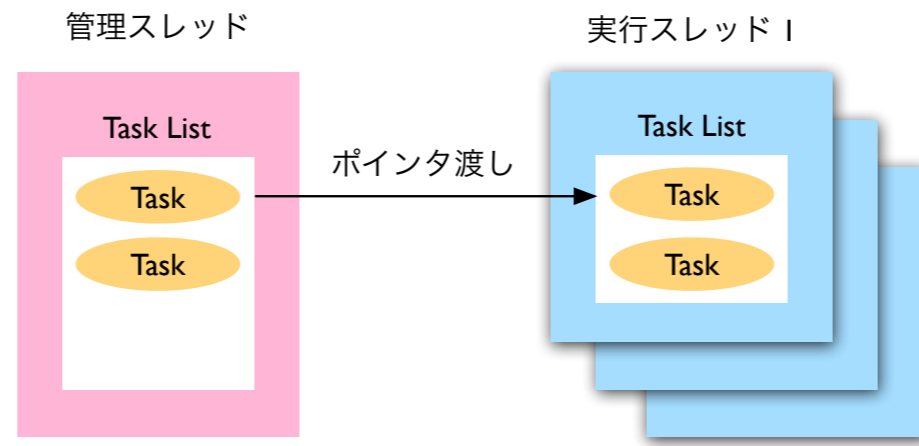
インタフェースを修正し、Spe Threadsに対応するCpu Threadsを実装

Cpu Threadsは、CPUコア数に対応する数を作成し  
並列に実行できる

Cellでは、メッセージをやり取りするために各SPEにMailboxという  
機構があり、Taskの割り振りに利用している

マルチコアでは、Mailboxに対応する形でCpu Threadsに  
2つのSynchronized Queueが渡されるようになっている

# ○ 実行スレッドへのTaskの割り当て



各CPUは、同じメモリ空間を利用できるので

DMA転送命令の代わりに**ポインタ渡し**を利用

さらにPrefetch命令も利用したが、有意な効果は見られなかった

6 CPU  
30万入力のソート

実行オプション	実行時間
ポインタ渡しを利用しない	1457 ms
ポインタ渡しを利用	1385 ms
ポインタ渡し & Prefetch	1383 ms

ポインタ渡しを利用した場合、約**5.2%**の速度向上が見られる

# ベンチマーク

	Word Count	Sort	Prime Counter
1 CPU / Cell	2381 ms	6244 ms	2081 ms
6 CPU / Cell	1268 ms	1111 ms	604 ms
1 CPU / Xeon	354 ms	846 ms	266 ms
6 CPU / Xeon	70 ms	163 ms	50 ms
12 CPU / Xeon	48 ms	127 ms	36 ms
24 CPU / Xeon	40 ms	100 ms	31 ms

## CentOS / Xeon

OS	CentOS 6.0
CPU	Intel Xeon X5620 @2.67GHZ * 2
Memory	128GB
Compiler	GCC 4.4.4

## PlayStation 3 / Cell

OS	Yellow Dog Linux 6.1
CPU	Cell Broadband Engine @ 3.2GHz
Memory	256MB
Compiler	GCC 4.12

## Word Count

100MBのテキストファイルの単語数カウント

## Sort

10万入力のソート

## Prime Counter

100万までの範囲の素数を全て数え上げ

CentOS / Xeon上で6CPUを利用した時、1CPUを利用した時と比較して

Word Countの例題で約**4.6**倍、Sortの例題で約**5.5**倍

Prime Counterの例題で約**4.8**倍の速度向上

しかしながら、24CPUを利用した時、12CPUを利用した時と比較して

速度は上がっているものの速度向上率が落ちている

並列化率が低いためと考えられる

並列化率の向上は今後の課題

# ○ まとめ

---

## Ceriumをマルチコアに対応させた

マルチコアではTaskの受け渡しに  
コピーは必要ないので、コピーを除いたバージョンと  
Prefetchのバージョンを実装し、計測した

今後の課題として、並列化率を向上させ  
プロセッサ数が増えた時の速度向上率を改善する