

マルチプラットフォーム対応 並列プログラミングフレームワーク

渡真利 勇飛 (並列信頼研究室)

研究概要

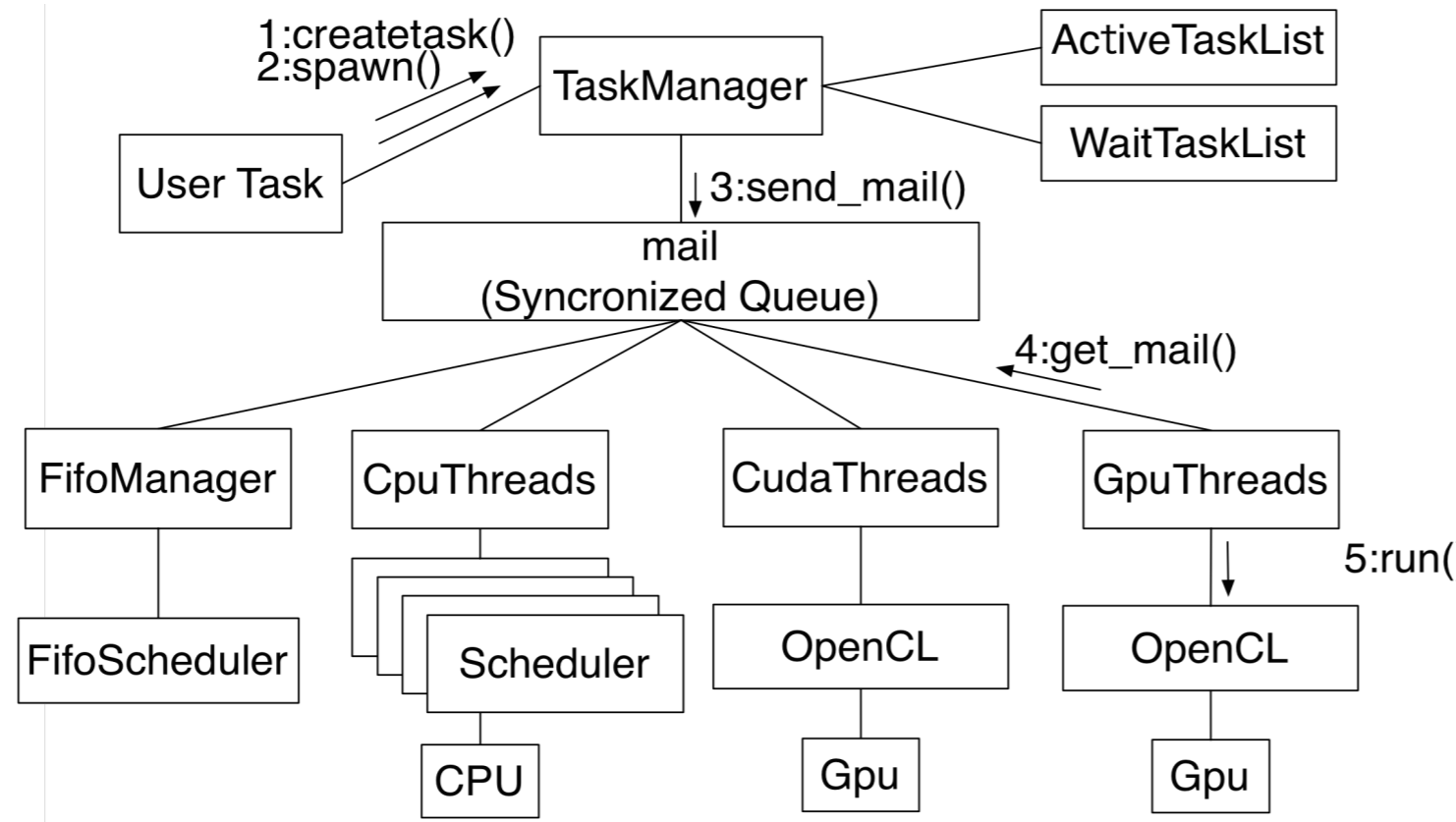
- 消費電力や発熱、クロックの限界といった問題からCPUの性能を上げることによる処理性能の向上は難しい
- マルチコアCPUやGPUを含んだヘテロジニアス構成が主流
- そういったマルチプラットフォームに対応したフレームワークが必要
- 並列プログラミングフレームワークCeriumを開発
- プラットフォームに最適な形でプログラムを並列に動作させる
- Ceriumの並列実行機構の測定・評価を行った

並列プログラミングフレームワークCerium

- CeriumはLinux、MacOSX上で動作する汎用計算用のフレームワーク
- CeriumはマルチコアCPUとGPUにおける並列プログラミングを可能にする

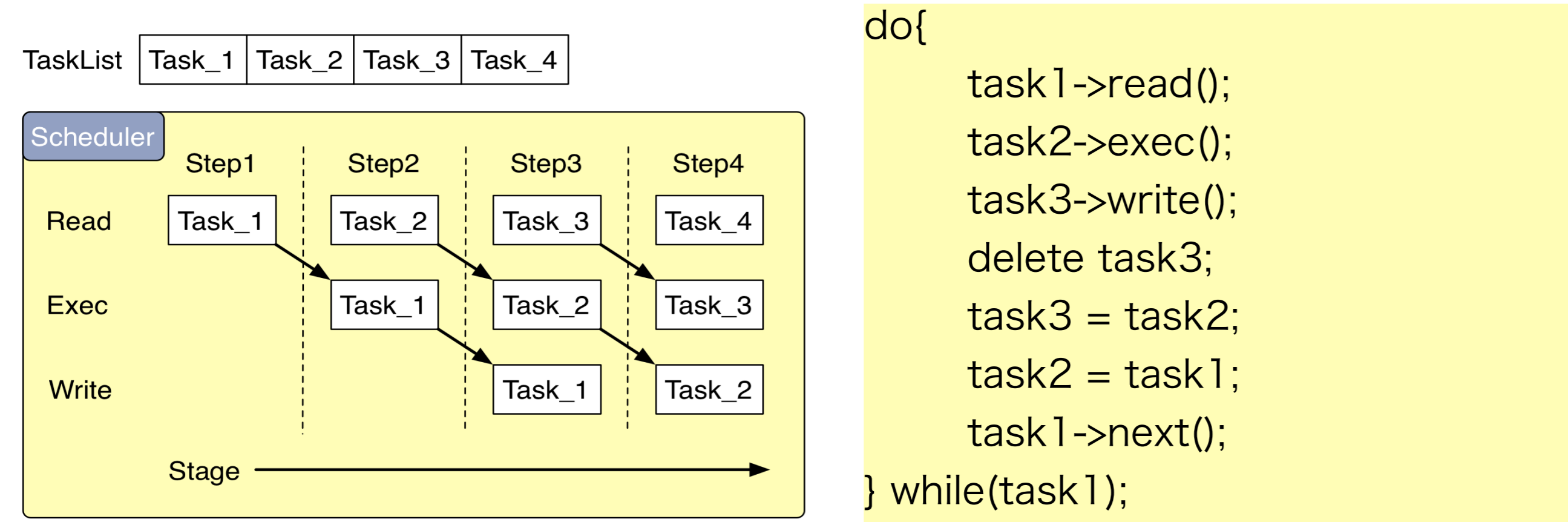
TaskManager

- Ceriumはユーザーが定義したTaskを受け取り、管理する機構TaskManagerを持つ
- TaskManagerとThreadsの間にはSynchronisedなMailQueueがある



- 依存関係の解決したTaskはTaskManagerからMailQueueに送られる
- ThreadsがMailQueueからTaskを取得し、並列実行していく

パイプラインによる並列実行



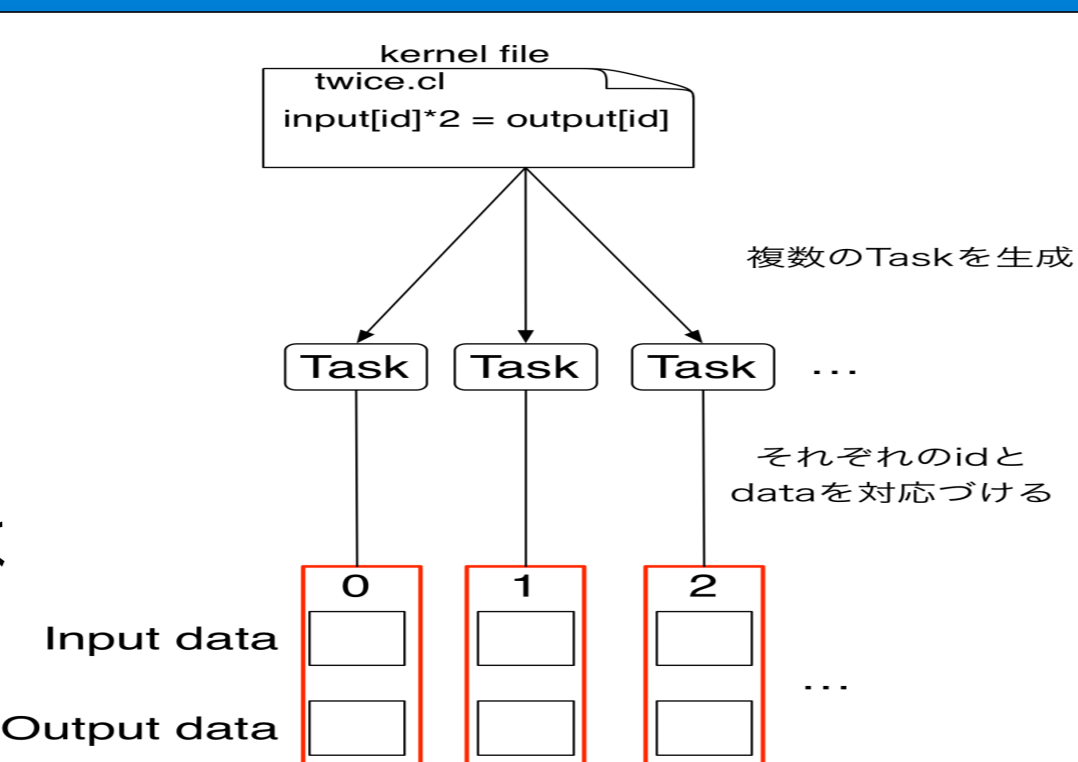
- マルチコアCPU上での並列実行はSynchronizedQueueとパイプラインにより構成されている
- TaskManagerで依存関係を解決されたTaskはSchedulerに送信され、パイプラインに沿って並列実行される

GPGPUへの対応

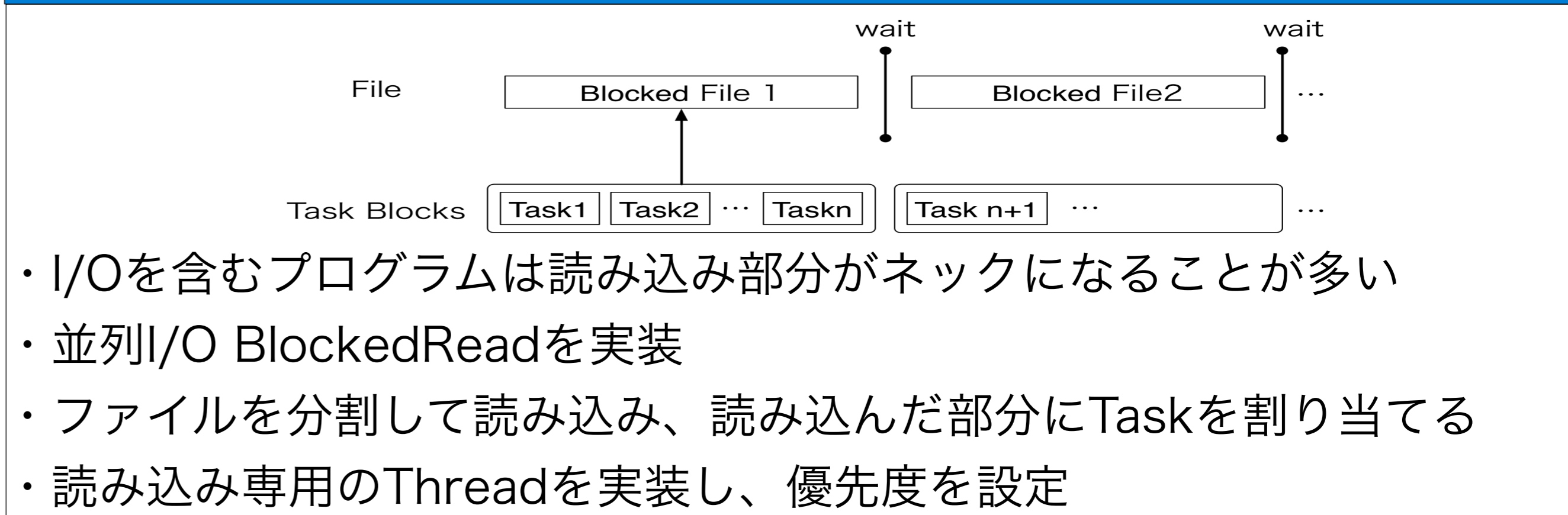
- OpenCL、CUDAを用いてCeriumをGPGPUへ対応
- TaskManagerから受け取ったTaskやデータをOpenCL、CUDAのAPIを介してGPUに転送する機構を実装
- OpenCL、CUDAにはGPUを制御するCommandを格納するCommandQueueがある。これを通してGPGPUを行う
- CommandQueueを2つ用意し、それを使ってパイプラインングしている

データ並列実行

- データ並列用のAPI、iterateを実装
- 1つの記述から複数のTaskを生成する
- 生成した複数のTaskにIDとデータを割り当てる
- Taskの持つIDと割り当てられるデータは対応があり、IDから担当範囲を算出できる

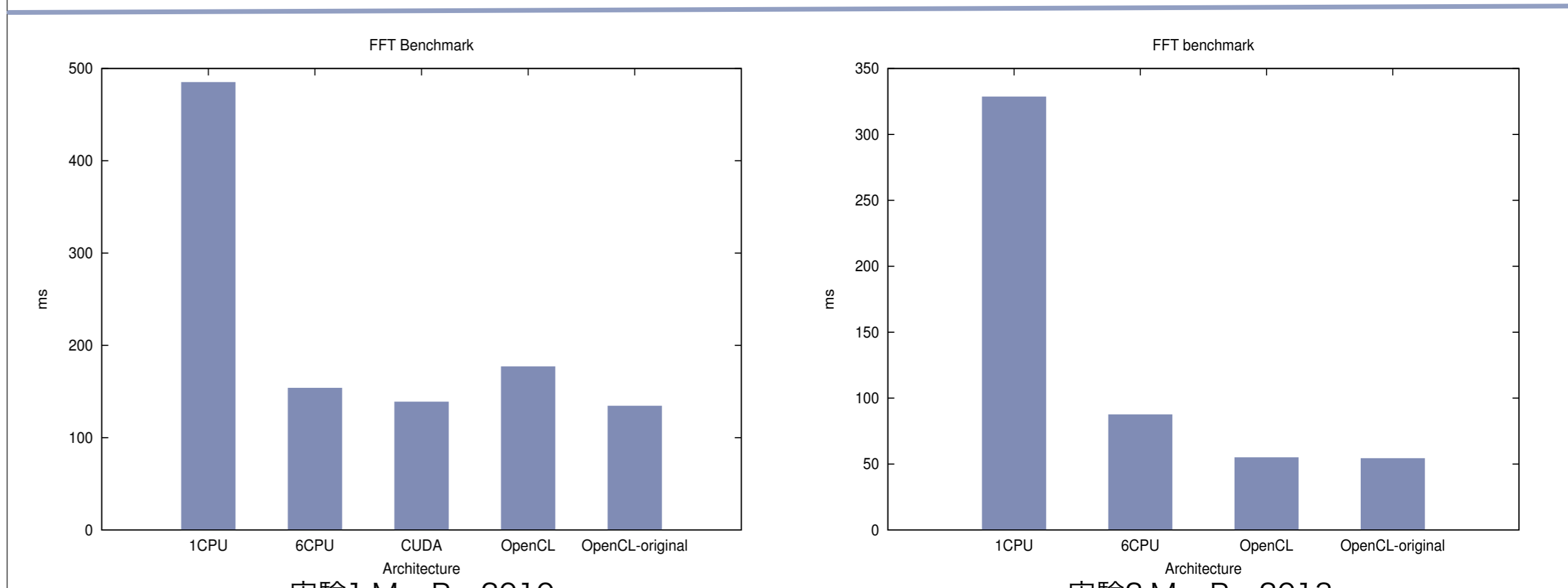
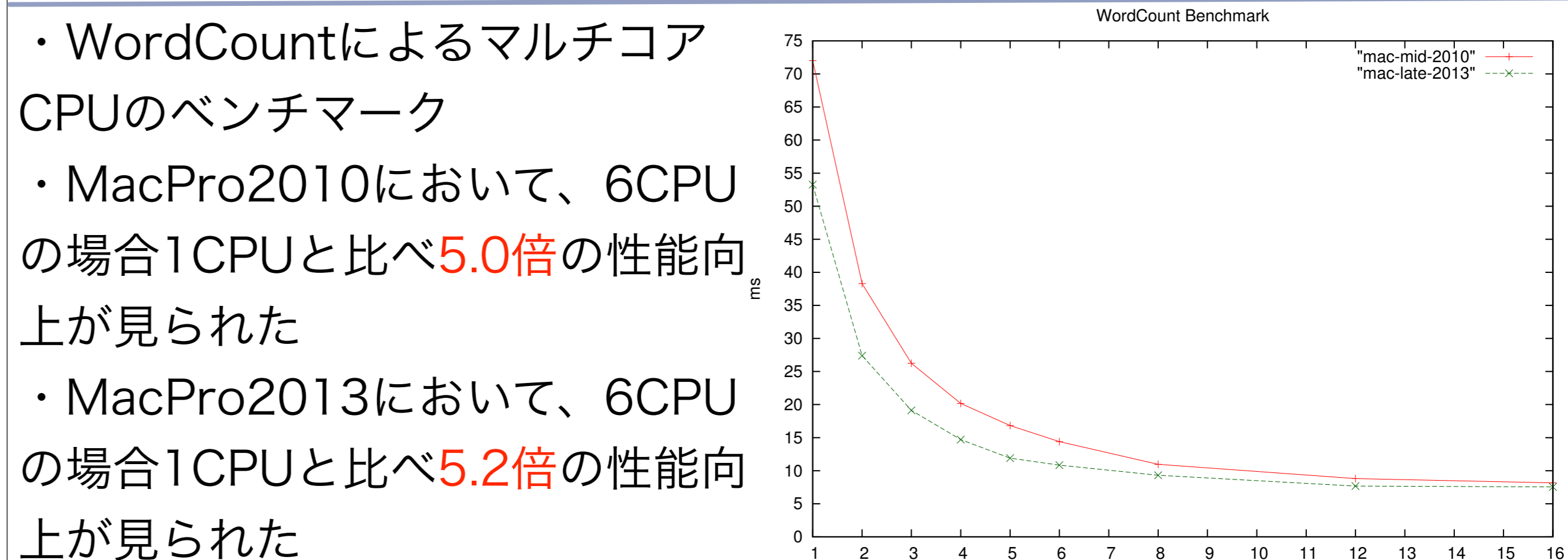


並列処理向けI/O BlockedRead



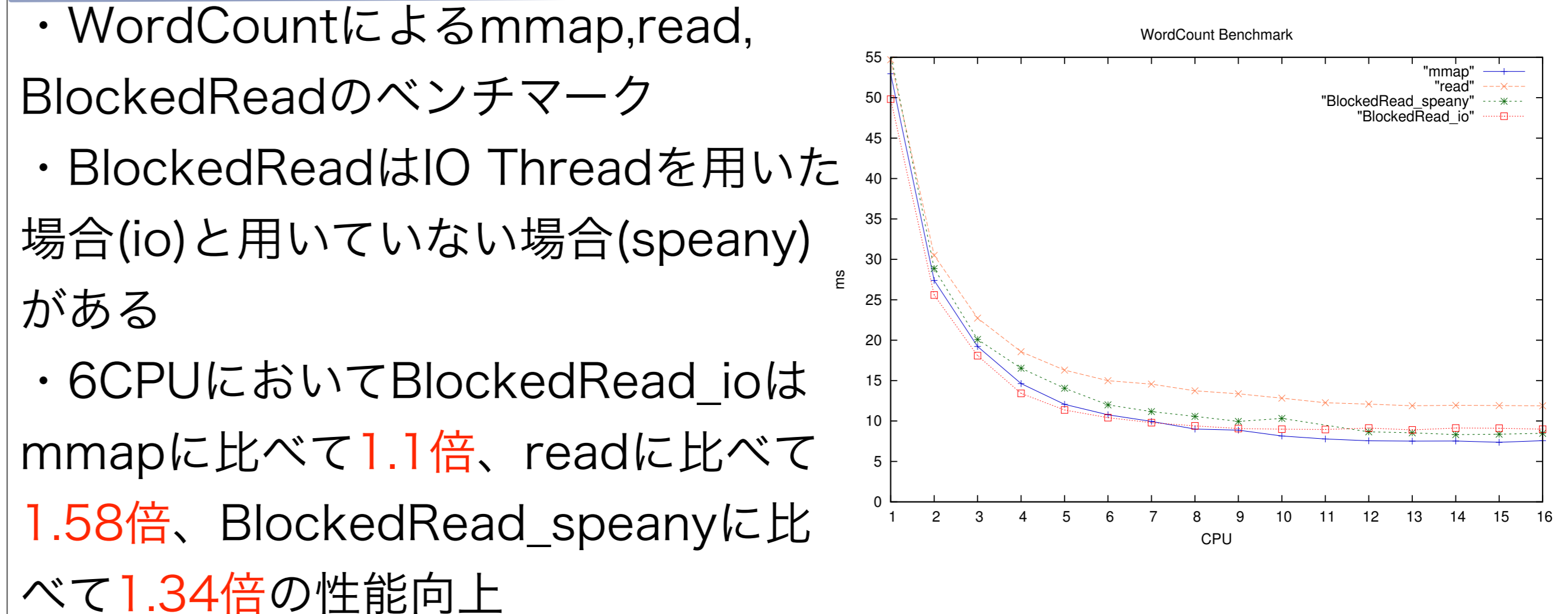
- I/Oを含むプログラムは読み込み部分がネックになることが多い
- 並列I/O BlockedReadを実装
- ファイルを分割して読み込み、読み込んだ部分にTaskを割り当てる
- 読み込み専用のThreadを実装し、優先度を設定

BlockedRead



- FFTによるCUDA、OpenCLによるGPGPUのベンチマーク
- 【実験1】 CUDAは6CPUと比べて1.1倍の性能向上
- 【実験1】 しかしOpenCLは6CPUと比べ0.76倍の性能低下
- 【実験1】 OpenCLのみのFFTと比べても0.76倍の性能低下
- 【実験2】 より高性能な計算機で測定
- 【実験2】 OpenCLは6CPUと比べ1.6倍の性能向上
- 【実験2】 OpenCLのみの場合と比べても同等の性能

GPUの性能を活用できている事がわかる。マルチコアCPUの処理性能も上がっていることから、GPUよりも上のレイヤでの最適化(パイプライン)が有効に働いていると考えられる。



今後の課題

- 依存関係の記述方式にDataDependencyを追加
- GpuSchedulerのパイプラインの改良

確認することリスト

- 載せるベンチマークはこれでいいか
- ベンチマークはラインで区切るのではなく別の項目にわけべきか
- 今後の課題、ポスターではこれくらいにして口頭で説明すべきか