Mio: A High-Performance Multicore IO Manager for GHC

Haskell Symposium 2013

Andreas Voellmy
Junchang Wang
Paul Hudak

Yale Univ

Kazu Yamamoto

IIJ-II

概要

問題点

GHC 7.6.3 以前でコンパイルした
並行(concurrent)プログラムが
マルチコア環境でスケールしない

解決方法

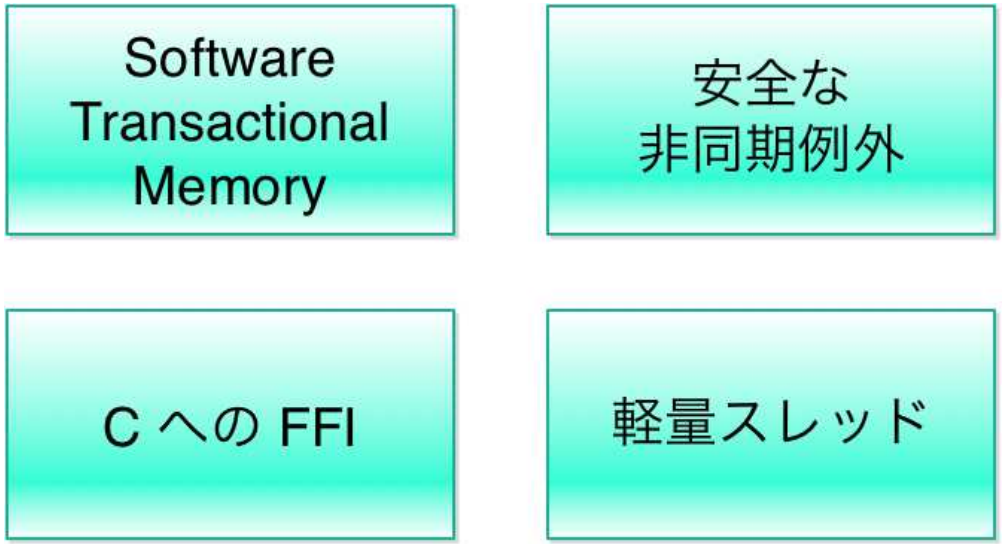
入出力を多重化するIOマネージャ
に存在したボトルネックを3つ
解消した

成果

GHC 7.8.1 以降で並行プログラム
をコンパイルすればマルチコア
環境でスケールする

並行 Haskell

- Haskell は並行プログラムを書くのに最適な言語

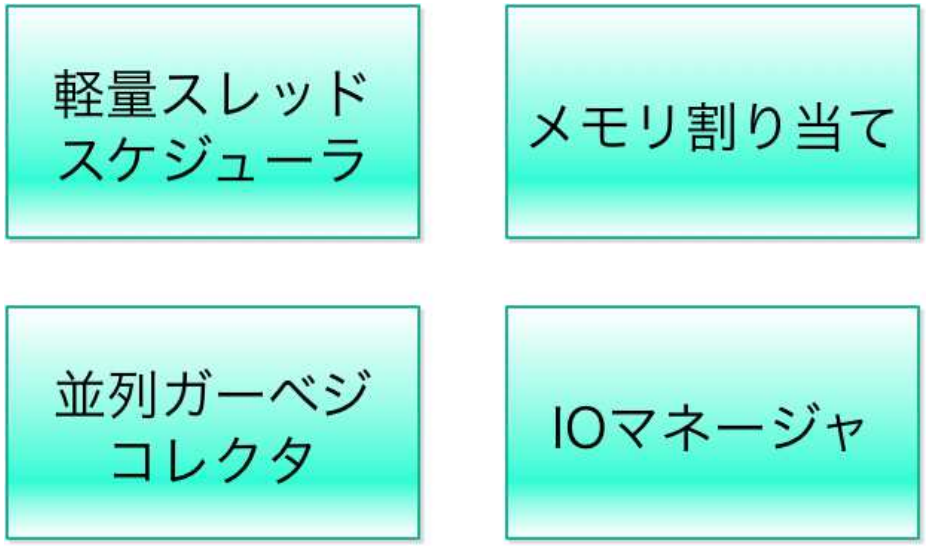


スレッドとイベント駆動

	コードの見通し	性能	
ネイティブ・スレッド	○	×	・制御を占有したコード ・コンテキストスイッチのオーバーヘッドが大きい
イベント駆動	×	○	・コードをコールバックに分割し状態を管理 ・コンテキストスイッチのオーバーヘッドがない
軽量スレッド	○	○	・制御を占有したコード ・コンテキストスイッチのオーバーヘッドがない

GHC のランタイム・システム

- マルチコア環境を思考した実装



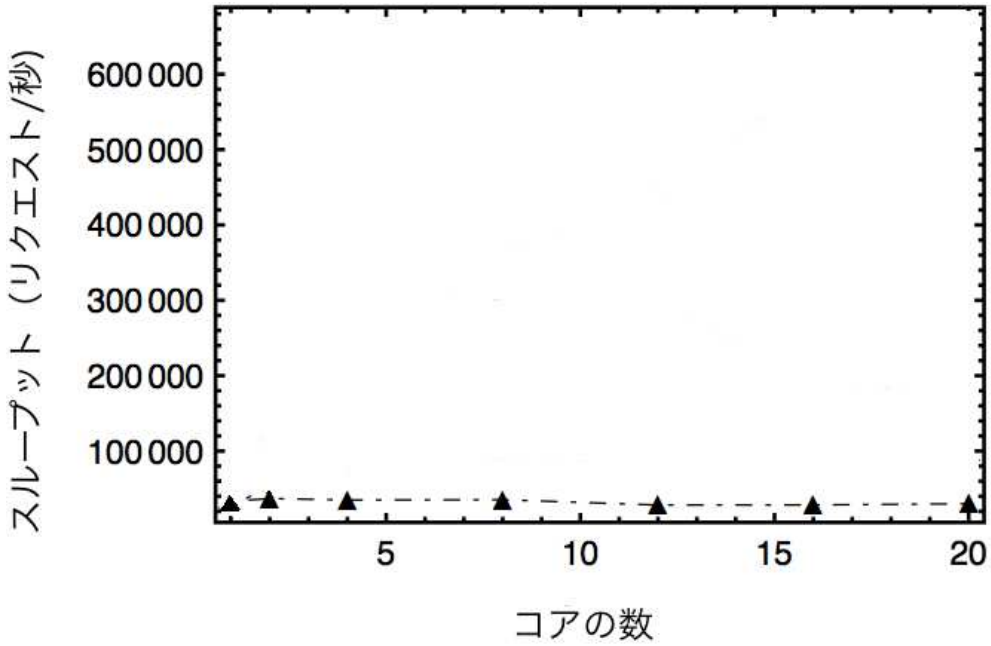
GHC でコンパイルし
ランタイム・システムをリンクした
並行プログラムは
マルチコア環境でスケールすべき

問題点

実際は、並行プログラムが
マルチコア環境でスケールしない

エコー・サーバのスループット

- コネクションごとに軽量スレッドを生成
- 極力オーバーヘッドを削除
- 400 コネクション、計 500,000 リクエスト



原因

IOマネージャにボトルネックがある

IOマネージャ

同期入出力

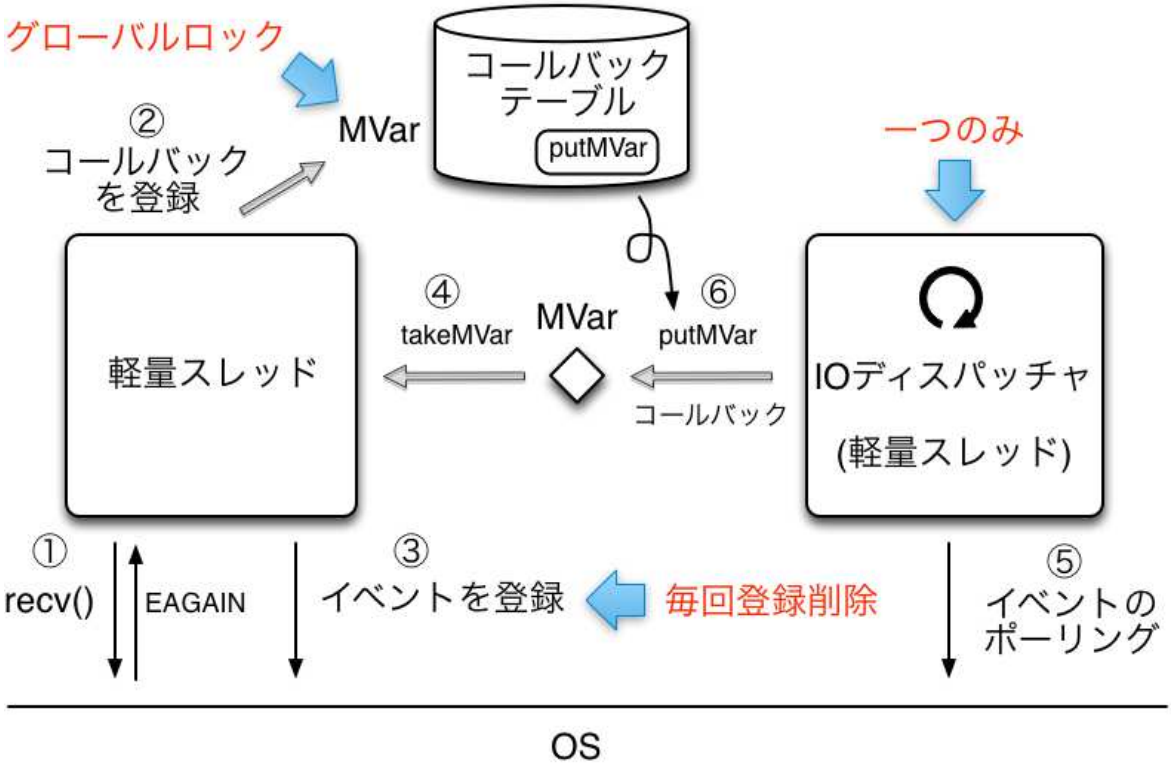
非同期入出力(ノンブロッキング)の上に同期入出力を構築

軽量スレッドは論理的にブロックされる

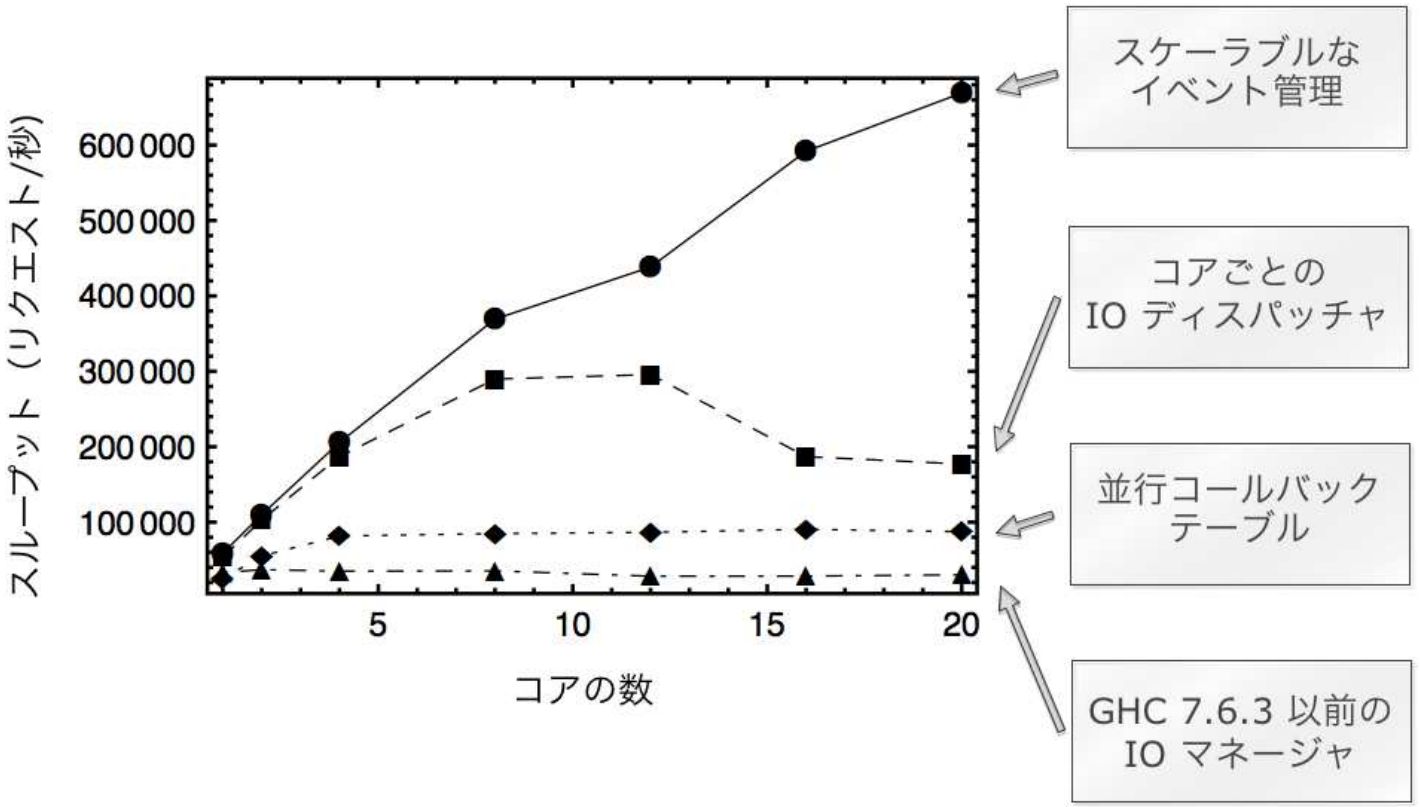
入出力の多重化

入出力イベントを多重化してポーリングし、軽量スレッドに通知

IO マネージャの仕組み

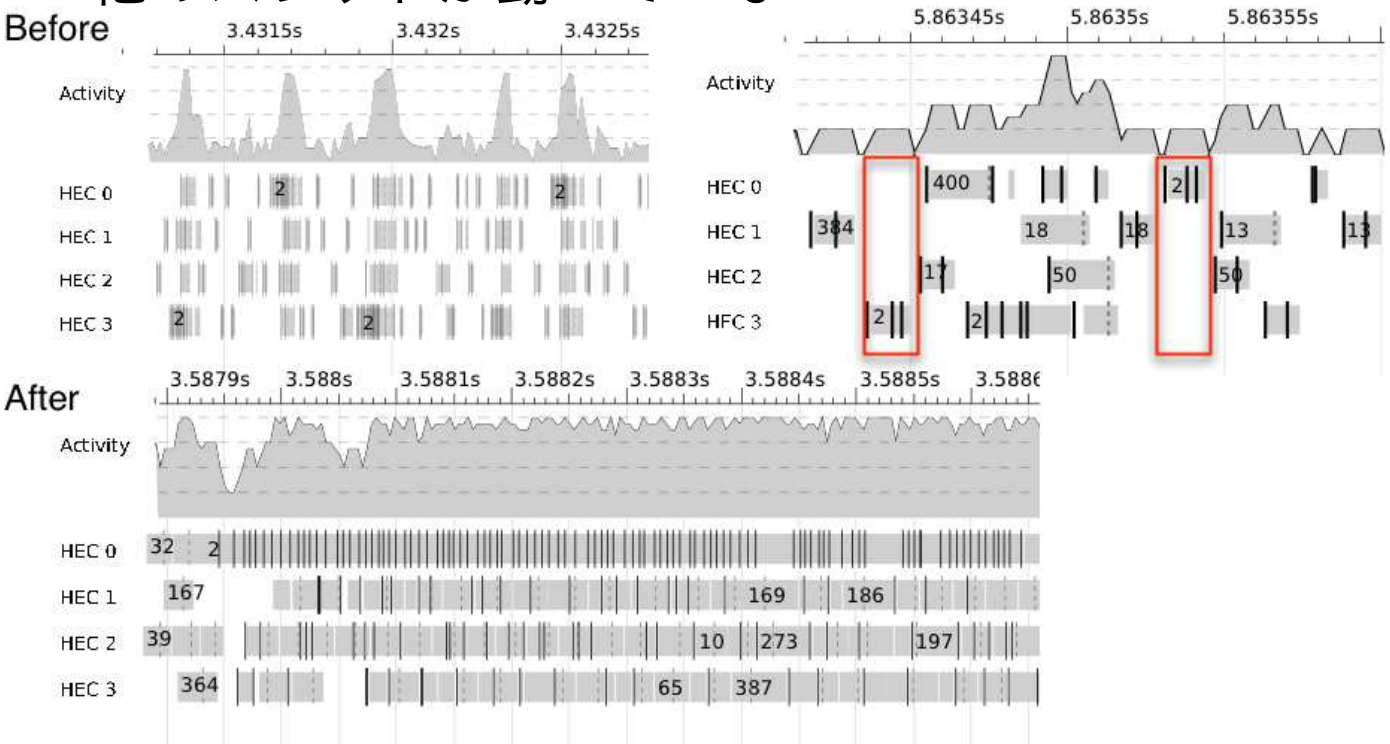


解決方法


スループット（リクエスト/秒）
600 000
500 000
400 000
300 000
200 000
100 000
5
10
15
20
コアの数
スケーラブルな
イベント管理
コアごとの
IO ディスパッチャ
並行コールバック
テーブル
GHC 7.6.3 以前の
IO マネージャ

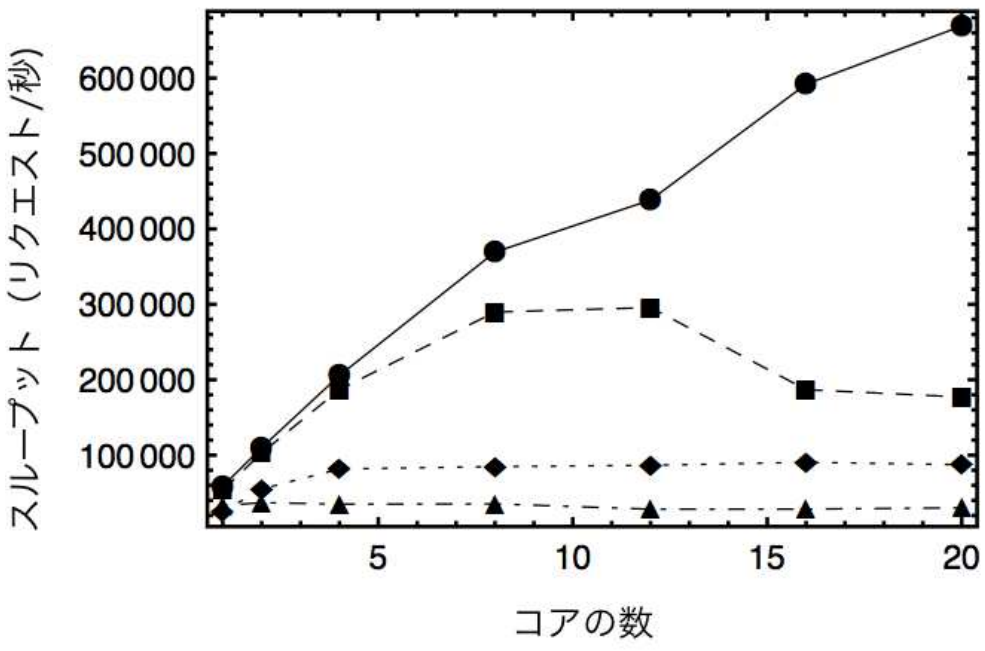
解決方法1: 並行コール・バックテーブル

- 他のスレッドが動いていない



- グローバルなジャイアントロックを分割

エコー・サーバのスループット(再掲)



スケーラブルな
イベント管理

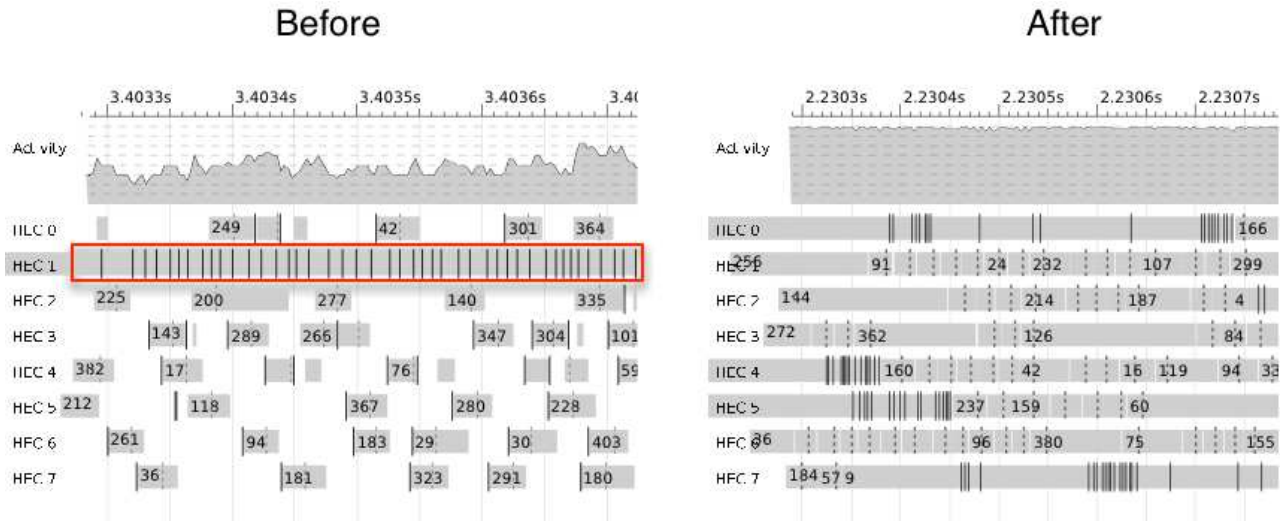
コアごとの
IO ディスパッチャ

並行コールバック
テーブル

GHC 7.6.3 以前の
IO マネージャ

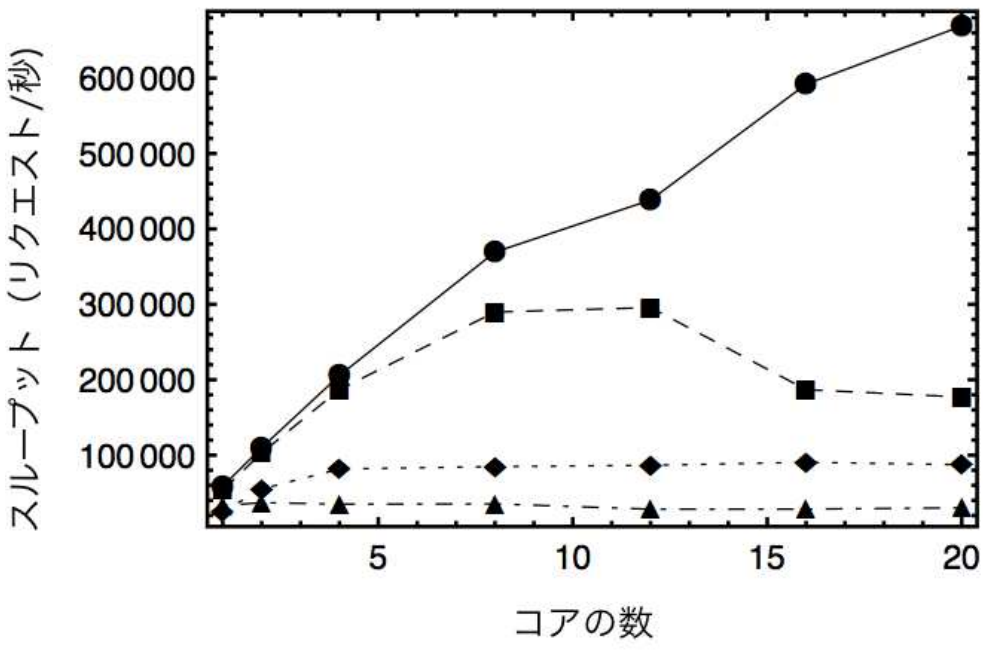
解決方法2: コアごとのIOディスパッチャ

- コアの数が増えると IO ディスパッチャだけが動いている



- コアごとの IO ディスパッチャを導入

エコー・サーバのスループット(再掲)



スケーラブルな
イベント管理

コアごとの
IO ディスパッチャ

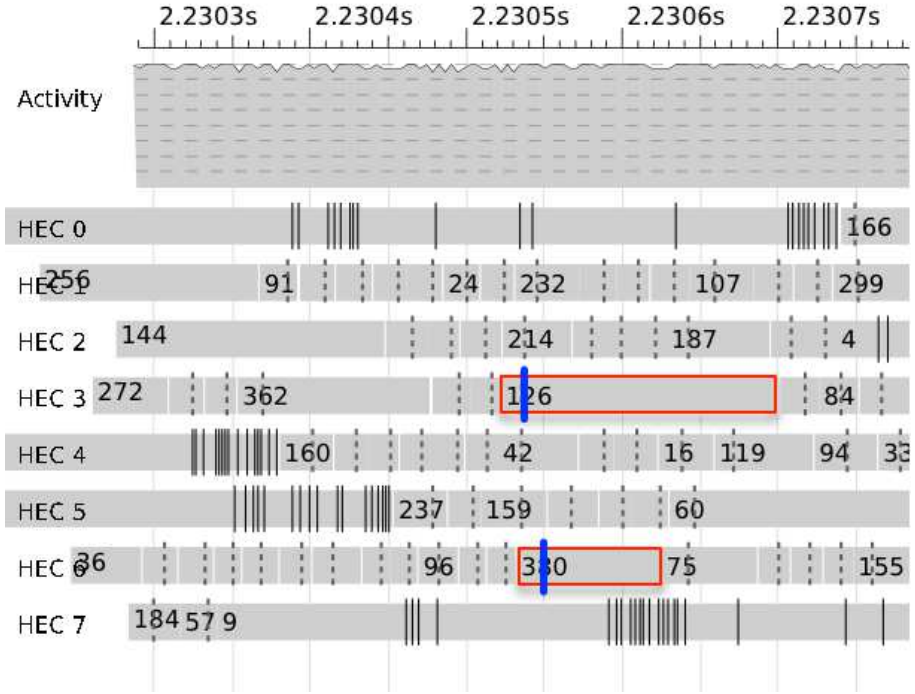
並行コールバック
テーブル

GHC 7.6.3 以前の
IO マネージャ

解決方法3: スケーラブルなイベント管理

- イベントの登録時に軽量イベントが待たされる

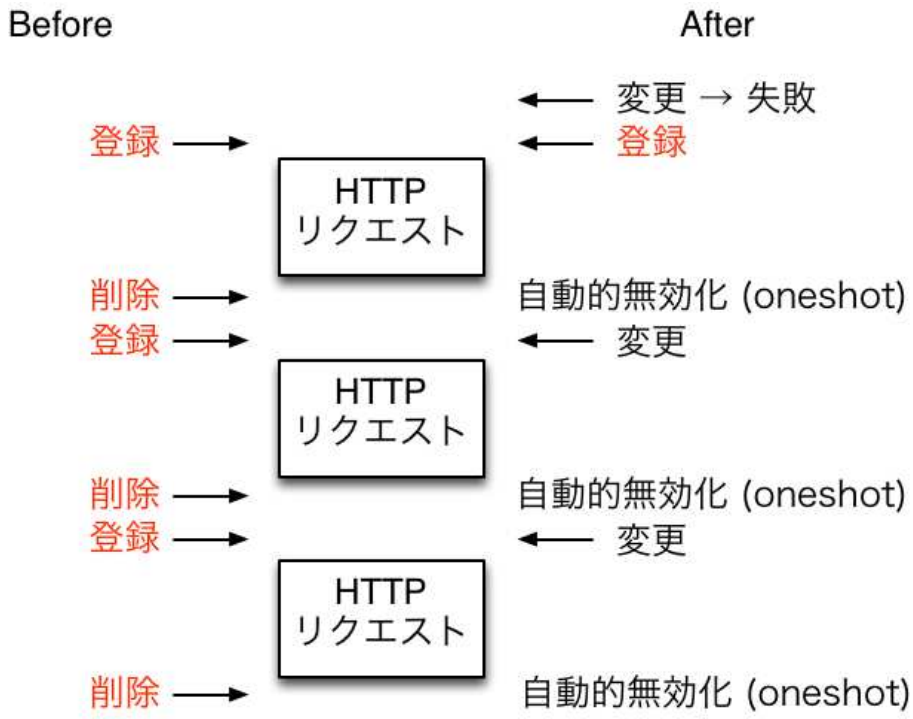
Before



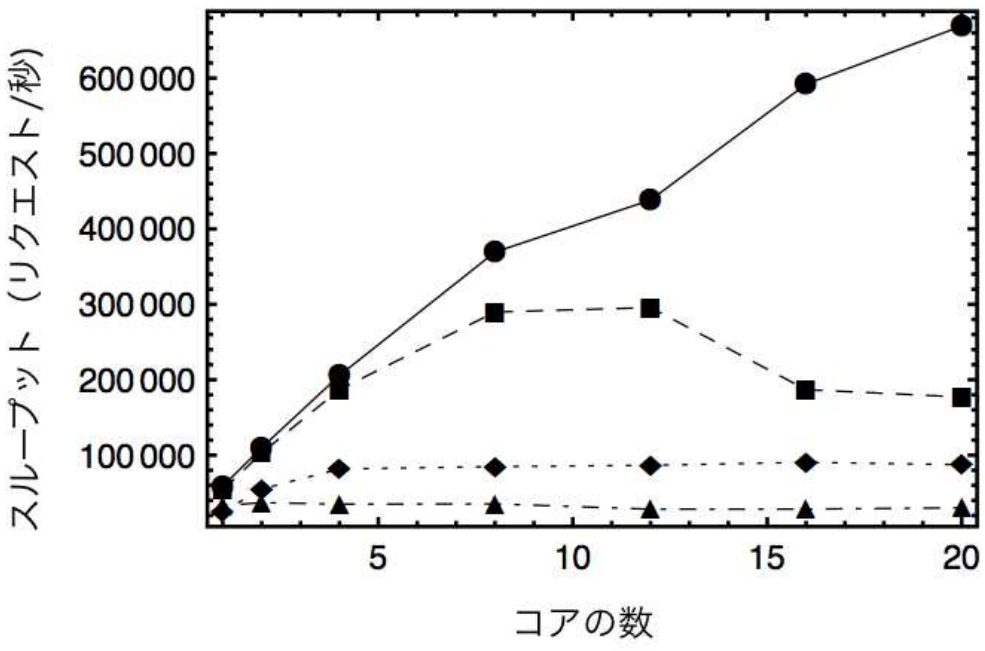
- OS はイベントのエントリを挿入/削除する際にグローバル・ロックを取る

スケーラブルなイベント管理 (2)

- OS はイベントのエントリを変更する際は
  グローバル・ロックを取らない



エコー・サーバのスループット(再掲)



スケーラブルな
イベント管理

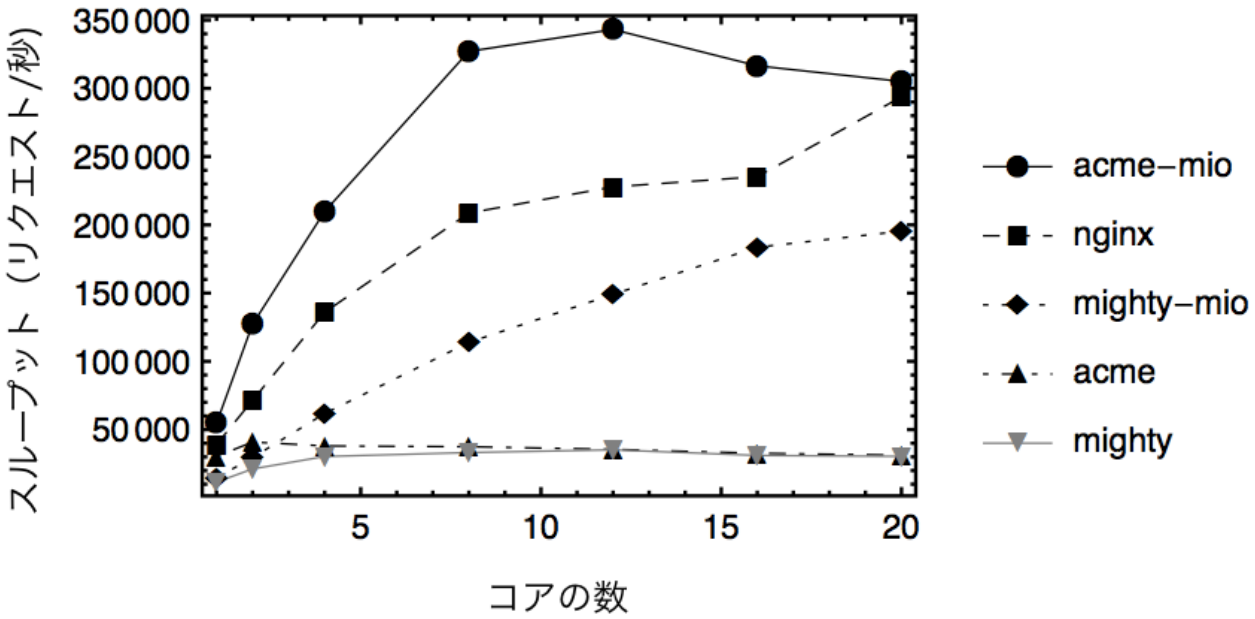
コアごとの
IO ディスパッチャ

並行コールバック
テーブル

GHC 7.6.3 以前の
IO マネージャ

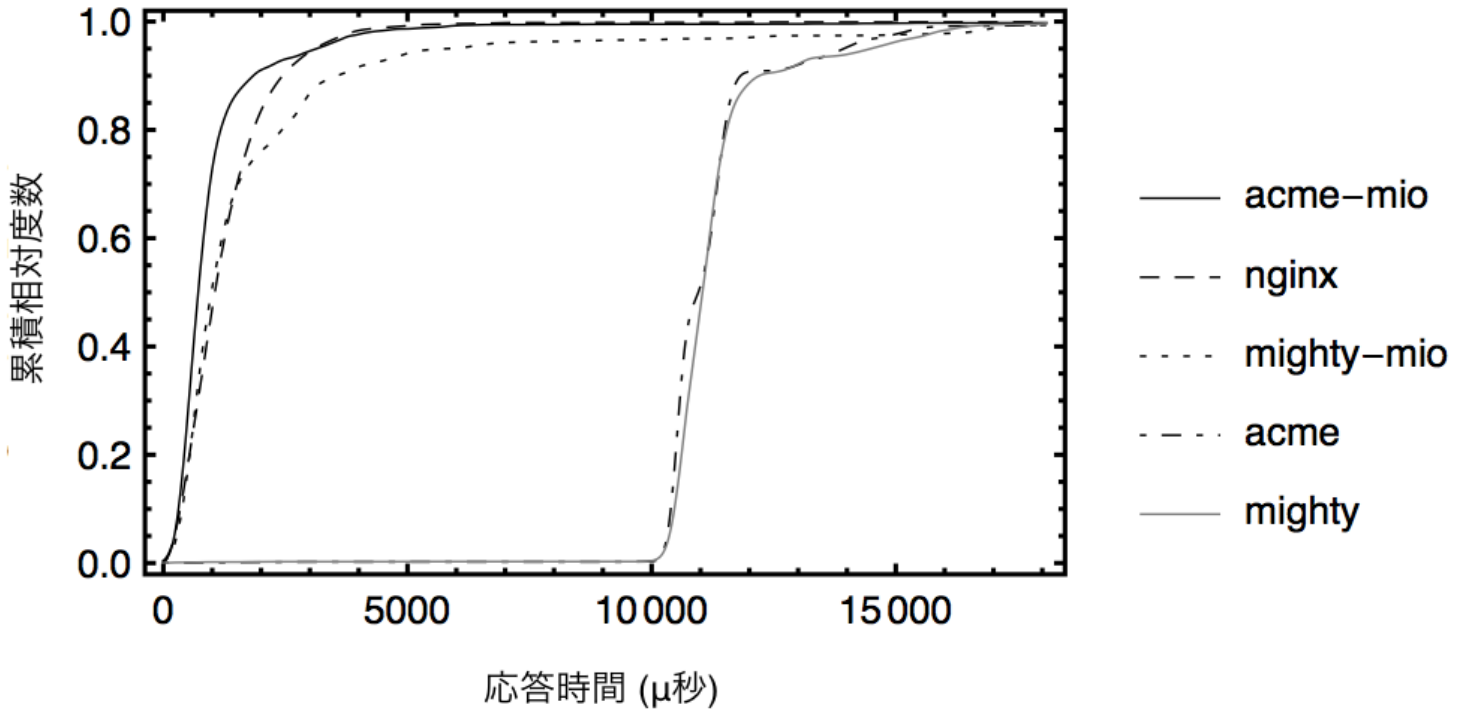
実践的なHTTPサーバのスループット

- サーバ機： HT なしの 80 コア
- クライアント機： HT ありの 8 コア
- ネットワーク：10Gpbs イーサ
- 計測ソフト： `weighttp`
- 400 コネクション、計 500,000 リクエスト



実践的なHTTPサーバの応答時間

- 同じ環境
- サーバ機：12コアを利用
- 計測ソフト：改良した `weighttp`



実装

- ほとんどの部分を Haskell で記述
  - 874 行追加
  - 359 行削除
- たくさんの OS をサポート
  - Linux では epoll
  - FreeBSD では kqueue
  - (Mac を含む)他のOS では poll
- GHC HEAD にマージ済み
  - もうすぐ GHC 7.8.1 の一部としてリリースされる
- たくさんのバグを発見
  - GHC IO マネージャのバグ
  - GHC RTS のバグ
  - GHC ビルドシステムのバグ
  - epoll の深刻なバグ

今後の課題

- Yield 関数がスループットを劇的に改善

```
loop = do
  recv
  send
  loop
```

```
recvfrom(13, )       -- Haskell thread A
sendto(13, )         -- Haskell thread A
recvfrom(13, ) = -1  -- Haskell thread A
epoll_ctl(3, )       -- Haskell thread A
recvfrom(14, )       -- Haskell thread B
sendto(14, )         -- Haskell thread B
recvfrom(14, ) = -1  -- Haskell thread B
epoll_ctl(3, )       -- Haskell thread B
```

```
loop = do
  recv
  send
  yield
  loop
```

```
recvfrom(13, )   -- Haskell thread A
sendto(13, )     -- Haskell thread A
recvfrom(14, )   -- Haskell thread B
sendto(14, )     -- Haskell thread B
recvfrom(13, )   -- Haskell thread A
sendto(13, )     -- Haskell thread A
```

  - IO マネージャを軽量スレッド・スケジューラに組み込む