# Haskell で Web サーバーを 実装してみました

山本和彦

IIJ INNOVATION INSTITUTE

# なぜ Web サーバーを実装したのか？

去年の秋、研究のために、自由に
変更できる Web サーバーが必要になった。

Apache も考えたが、大きくて複雑。
C を読んだり書いたりするのは、もううんざり。

Haskell で書いた Web サーバーが欲しい。
でも、そんなものの存在は知らなかった。

そこで一から書き始めた。
名前は Mighttpd。Mighty と発音する。

# 3つのゴール

**機能**

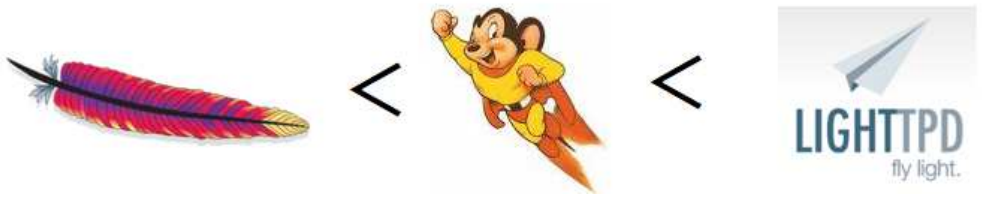
Mighttpd は、僕のドメイン Mew.org の Apache を置き換えるに十分な機能を提供すること。

**部品化**

Mighttpd は、僕の研究のために簡単に変更できること。

**性能**

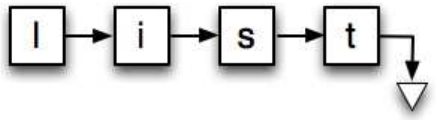
Mighttpd は、静的コンテンツに対して Apache よりも性能を出すこと。



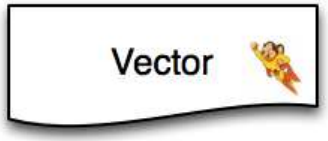
# 性能を出す2つのアイディア

ByteString

伝統的な Haskell の **`String`** はとても遅い。




**`ByteString`** は C の **`char[]`** と同程度に速い。




ユーザースレッド

カーネルスレッドは重い。

ユーザースレッドは軽い。

# HTTP とスレッド

| | | |
|---|---|---|
| ネットワーク<br>プロトコル | メッセージ<br>指向 | ストリーム<br>指向 |
| | DNS | SMTP, HTTP |
| ネットワーク<br>プログラミング | イベント<br>駆動 | スレッド |
| | `select,kqueue,epoll` | `fork,pthread_create` |

ストリーム指向をイベント駆動で書くと
見通しが悪い。

ストリーム指向をスレッドで書くと
見通しがよい。

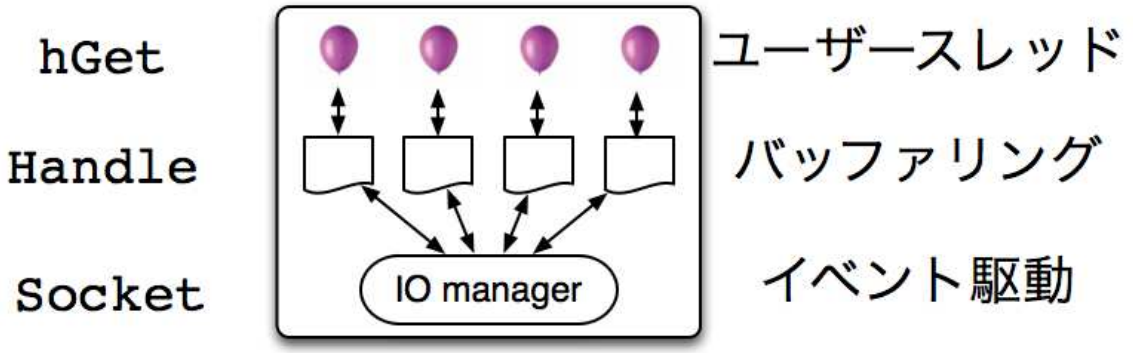
HTTP はスレッドで書きたい。
簡潔なことは、よいことだ。

# ユーザースレッドは本当のスレッド

GHC にはユーザースレッドで実装された
イベント駆動の IO マネージャーがある。

バッファリングしたり、ブロックしている
ユーザースレッドを起こしたり。

ユーザースレッドを使っても、真の
スレッドプログラミングができる。

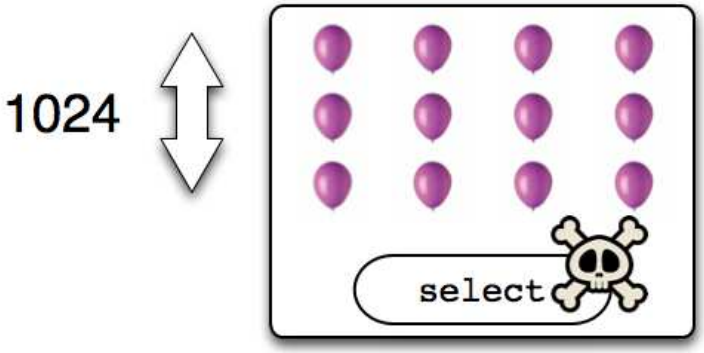


# 1,024 コネクションの壁

IO マネージャーは `select` を使って実装されている。

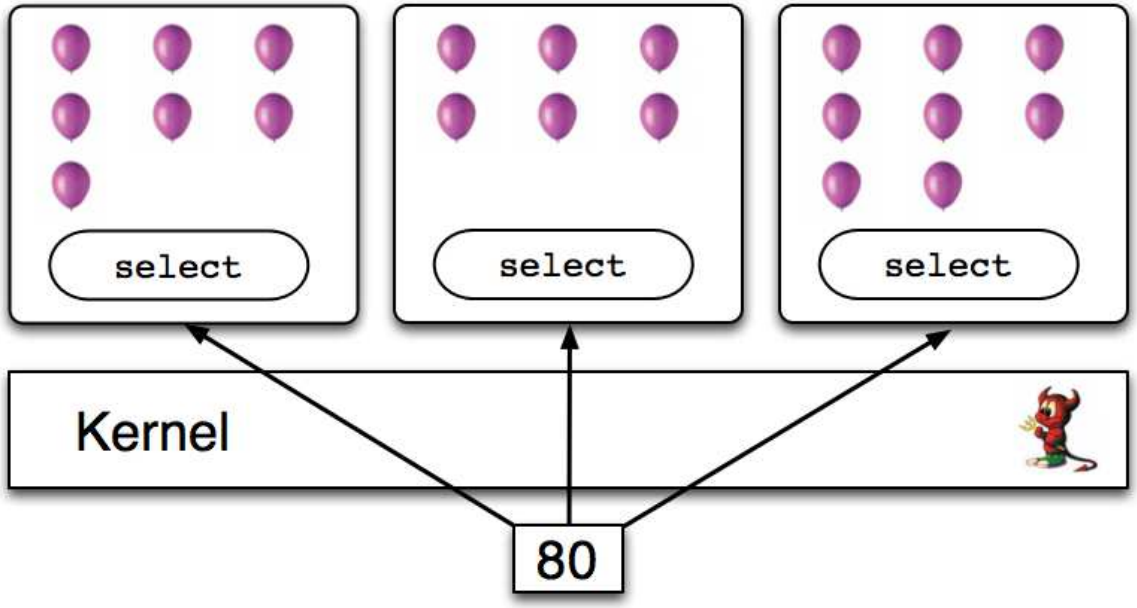
`select` は 1,024 以上のファイル/コネクションを扱えない。

もし GHC 6.12 が 1,024 以上のコネクションを受け取ると、資源枯渇の例外が発生する。



# Prefork ライブラリ

Prefork は、fork したプロセス間で
リスニングポートを共有する技術。




これで GHC 6.12 でコネクションを
何本でも受けられる。

# Mighttpd の実装

パケージ名

| | | | |
|---|---|---|---|
| mighttpd | File base | KVS base | リリースしていない |
| webserver | HTTP, session, redirect, CGI | | |
| c10k | prefork | | |

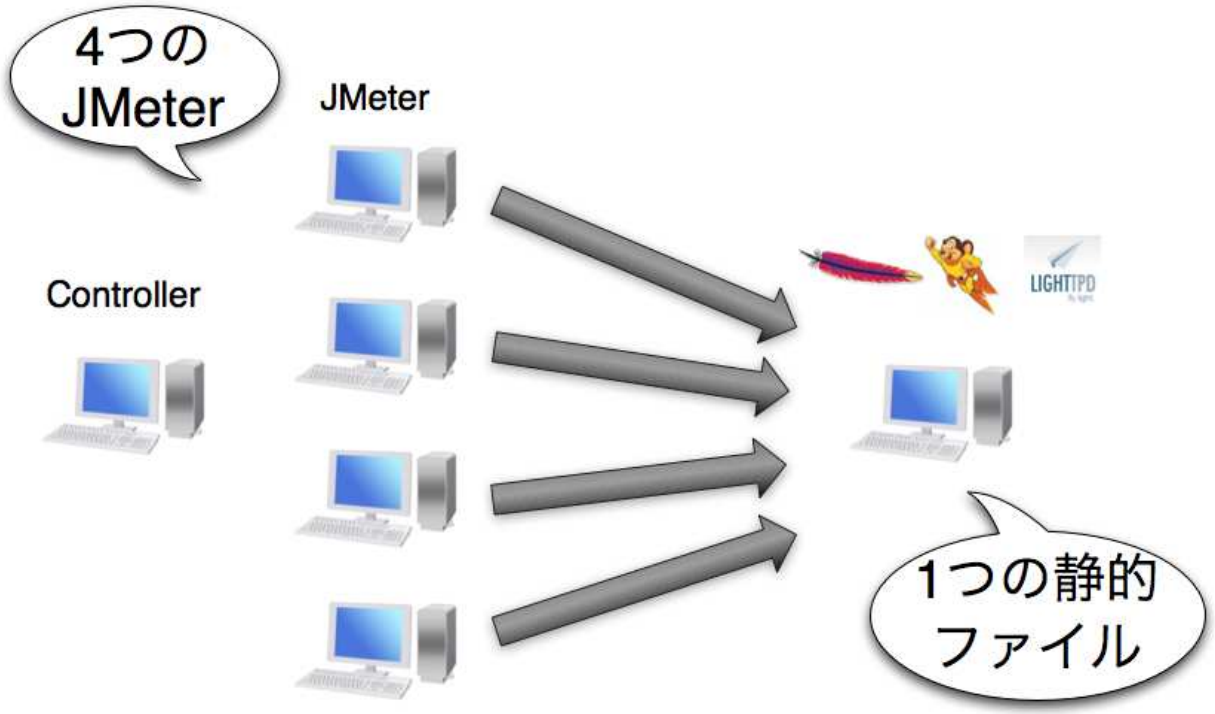
部品化

"webserver" は、いろいろなストレージを利用できるように設計。

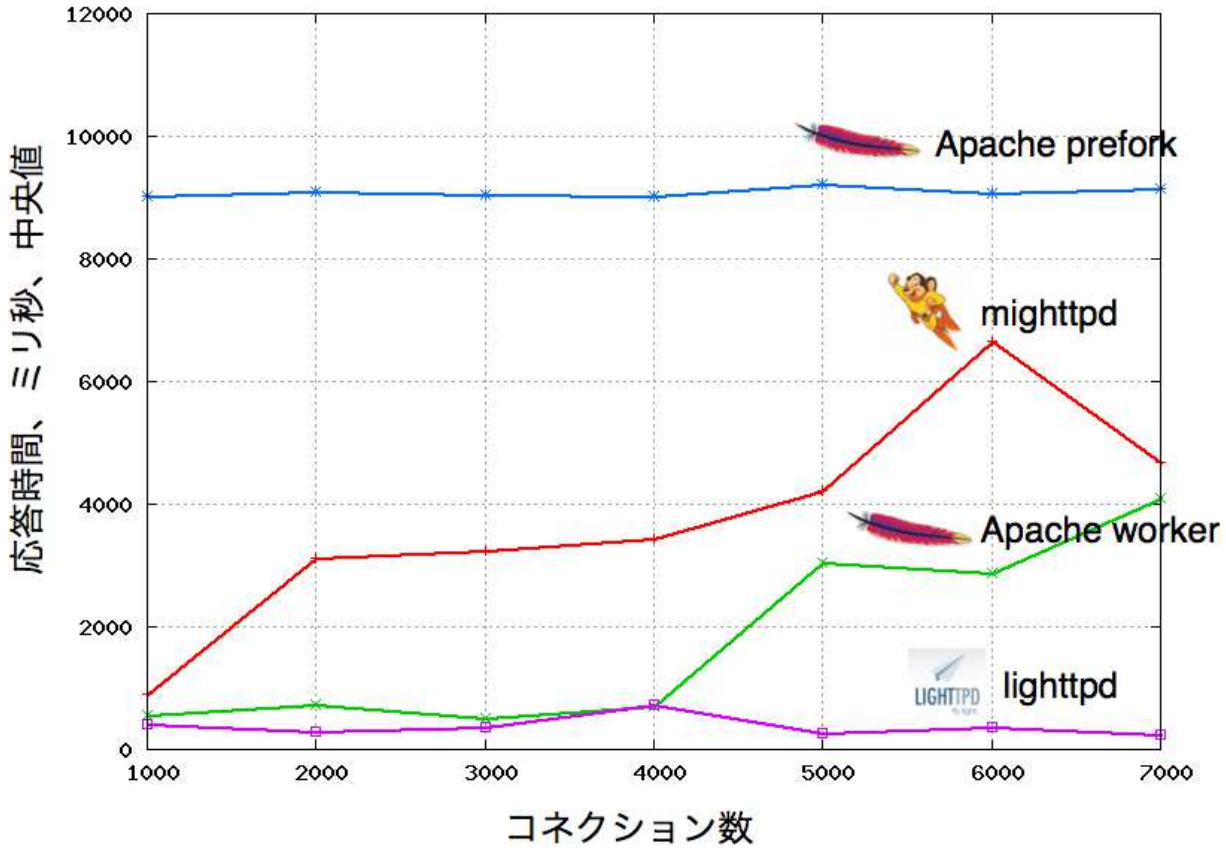
機能

"mighttpd" は、Mew.org で動いている。

# ベンチマーク環境



# ベンチマークの結果




- ベンチマークは不安定なので、あまり信用しないで下さい

# プロファイル

ファイルの IO が支配的。
なぜ、Mighttpd は Apache より遅いのか？

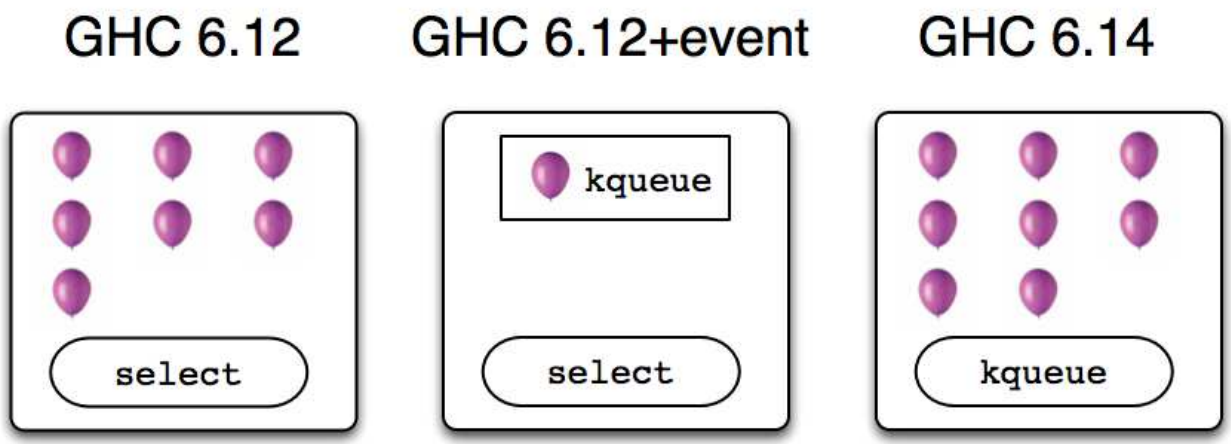
```
% ab -n 2000 -c 200 -k http://localhost/

COST CENTRE    MODULE    %time %alloc

fileGet        File      73.3   37.4
mighty         File      20.0   57.9
fileInfo       File       6.7    2.9
fileMapper     File       0.0    1.1
```

あー、select のせいだ。

なにか方法はないのか？

# 解決策

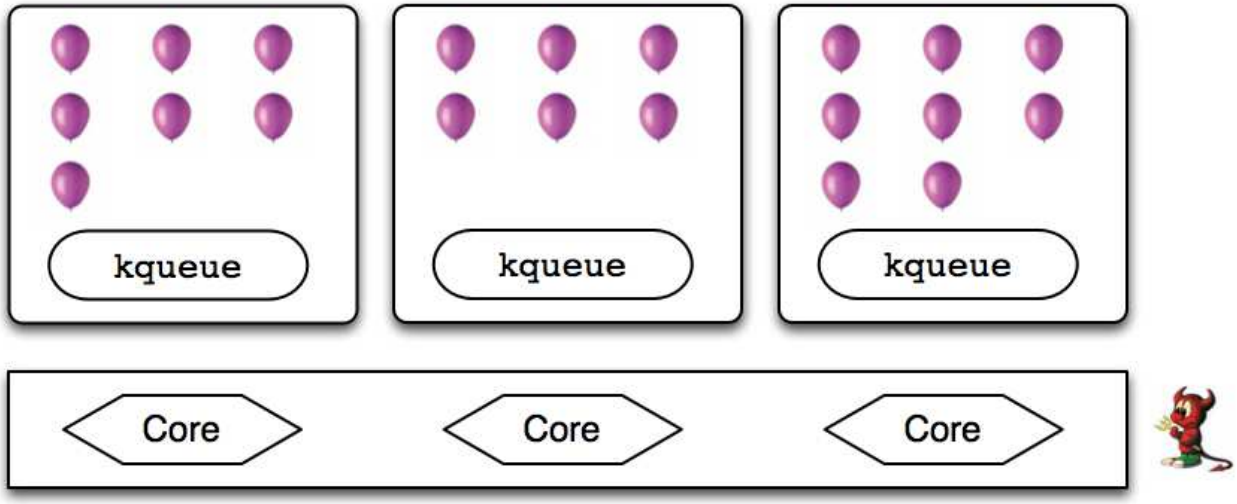
- Tibbe と Bos が `kqueue` と `epoll` を使う "event" ライブラリを開発中。
- 現時点では、それをイベント駆動のプログラミングに利用可能。
- GHC 6.14 では IO マネージャーに統合する予定。



# 将来のアーキテクチャー

IO マネージャーは1つしかないので
GHC 6.14 はマルチコアを活用できない。

しかし prefork 技術を使えば、
マルチコアを活用できるだろう。



# 結論

Haskell のユーザースレッドを使った
ネットワークプログラミングは楽しい！

`select` のせいで、GHC 6.12 はネット
ワークプログラミングに弱い。

GHC 6.14 では、この問題が解決される。
ネットワークプログラミングを楽しもう。

Prefork ライブラリを使えばマルチコアを
活用できるだろう。

# 関連リンク

- Mighttpd
  - http://www.mew.org/~kazu/proj/mighttpd/
- My github
  - http://github.com/kazu-yamamoto
- JL Smiley
  - http://jamlog.podzone.org/