
(Digest)

高橋征義

現狀

Apache

- 比較的大きい
- 静的なコンテンツを大量・高速に配信したい場合には不利
- Fork処理が重い
- プラットフォーム依存度を低くするため、プラットフォーム依存の技はあまり導入されていない



The **Apache Software Foundation**

<http://www.apache.org/>

PHP

- お手軽（使いこなせばそれなりに高機能）
- 実行速度はあまり早い
 - ↑ 実際の開発では、実はあまり問題にならない
 - むしろDBなどが問題になることが多い
- 便利なライブラリが少ない
 - O/Rマッパー（DBとの連携）
 - アプリケーションフレームワーク
 - → 複雑なものの省力化に不利



PostgreSQL

- めちゃめちゃ高速、というほどではない
- 機能は充実しつつある
- 標準のレプリケーションツールがない
 - 標準じゃなければいくつかはある



PostgreSQL

The world's most advanced
open source database.

Linux 2.4

- そろそろ交替の時期
 - Redhat Enterprize Serverも最新版で2.6対応に



新技術

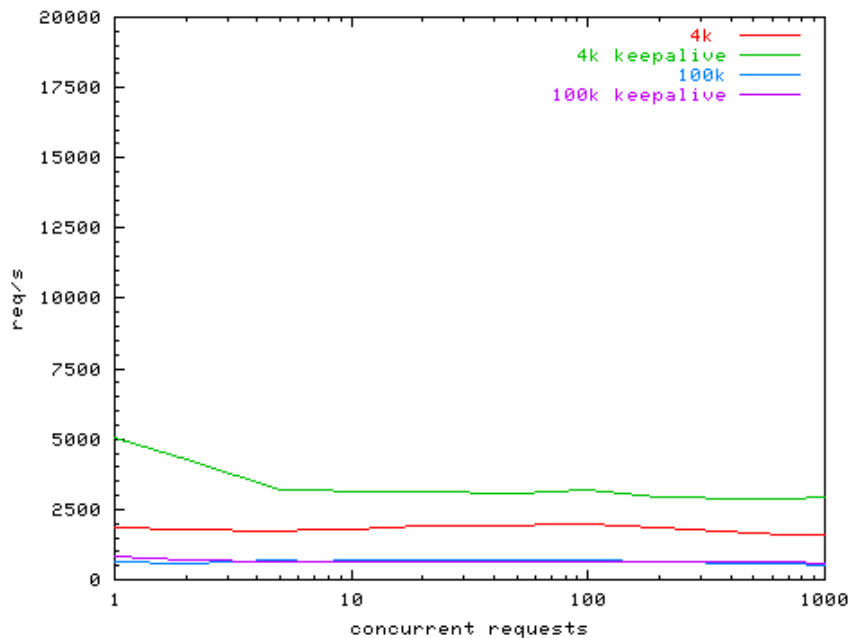
Lighttpd

lighttpd

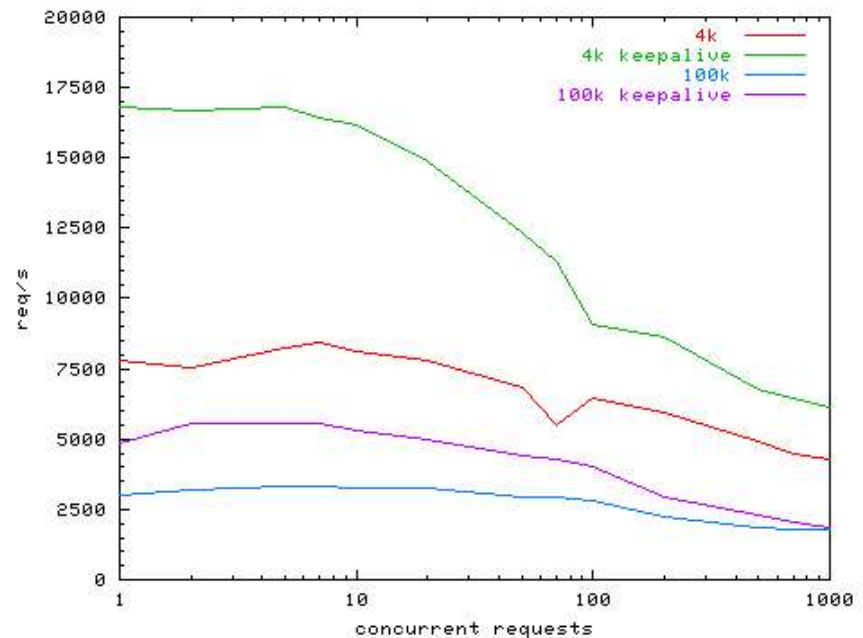
- 高速・軽量なWebサーバ
- Forkしない(signalを使ったり、プラットフォームによってepollやkqueueなどを使ったりする)
- Moduleもいくつかある(Apacheほどではない)
 - mod_fastcgi
 - mod_rewrite
- 設定ファイルも小さい

Lighttpd評価

- 実測ではApacheと同程度？
 - ↓Linux 2.6系じゃないとだめかも



apache 1.3.28



lighttpd 1.0.2 with poll

FastCGI

- 実はApacheでも使える既存の技術
- CGIのように、別プロセスで起動
 - あらかじめ複数プロセスを用意しておく
- アクセスごとにプロセスを起動する必要がない
→実行のコストが小さい

FastCGI

Ruby



- 比較的素直なスクリプト言語
- オブジェクト指向プログラミング言語(OOPL)
 - 今となってはOOできないスクリプト言語の方が少数なのでそれだけではあまり比較対象にはならない
- 高度な記述力
 - ブロックつきメソッド呼び出し(手軽なクロージャ)
 - リフレクション(実行時にもクラス定義を変更できる)
 - 充実したリテラル

Ruby on Rails

- MVCなWebアプリケーションフレームワーク
- オールインワン(DB、テンプレート、コントローラ、メールなど全て内包)
- 去年の夏ごろから一部で話題
- 今年に入り海外でブレイク中
- 巧みなマーケティング
 - 「開発効率はJavaの10倍」
 - blog、ビデオによる宣伝



ActiveRecord

- Ruby on Railsの一部
 - 独立して使用することも可能
 - 最小限の記述で動作する
 - Conversion over Configuration(設定よりも規約を重視)
 - カラム名を書く必要もない
 - プライマリキーも指定しなくてよい
 - その分制約に従わないと面倒になる
 - 自然な記述で表現できる
-

ActionView

- ほとんどPHPなみのシンプルなテンプレート
 - なんでも書けるが、書けすぎかも？
 - 批判や代替案も多い
 - が、とりあえずみんなそのまま使っている
-

JavaScript

- Ajax(Asynchronous JavaScript and XML)で一躍ブームに
 - 最近のブラウザに限定すれば、比較的仕様が安定しつつある
 - とはいえ、古い環境を使う必要があるなら意味がない
-

MySQL



- とにかく高速なDB
- PostgreSQLに比較すると、高速ではあるが機能は乏しい
 - まさに「早ければなんでもいい」という感じ
- 最近トランザクションも可能になったが、トランザクションが使えないDBエンジンでの利用が根強い

言 詳 糸 田

DBアクセスのコード(その1)

- テーブルとカラムを指定する

```
class Foo < ActiveRecord::Base  
end
```

↑ これだけ。2行。すごい。

- これでFoosテーブル参照・登録の準備ができる
 - カラムの情報などは実行時にDBから取得するため
- DB名やアカウント・パスワードは別途指定

DBアクセスのコード(その2)

- 検索する

```
@foos = Foo.find_all(where, " id ", 300)  
( whereにはWHERE節の文字列が入る)
```

- 登録する(バッチ)

```
foo = Foo.new()  
foo.aa = cols[1]  
foo.bb = s2e(cols[2])  
.....  
foo.kkk = s2e(cols[9])  
foo.save
```

パフォーマンス

- 悪くない(が、すごく良いわけでもない)
- DBの検索だけなら0.01秒を切ることもある(前方一致なら0.0029秒とか)
 - 表示系がちょっと遅い

