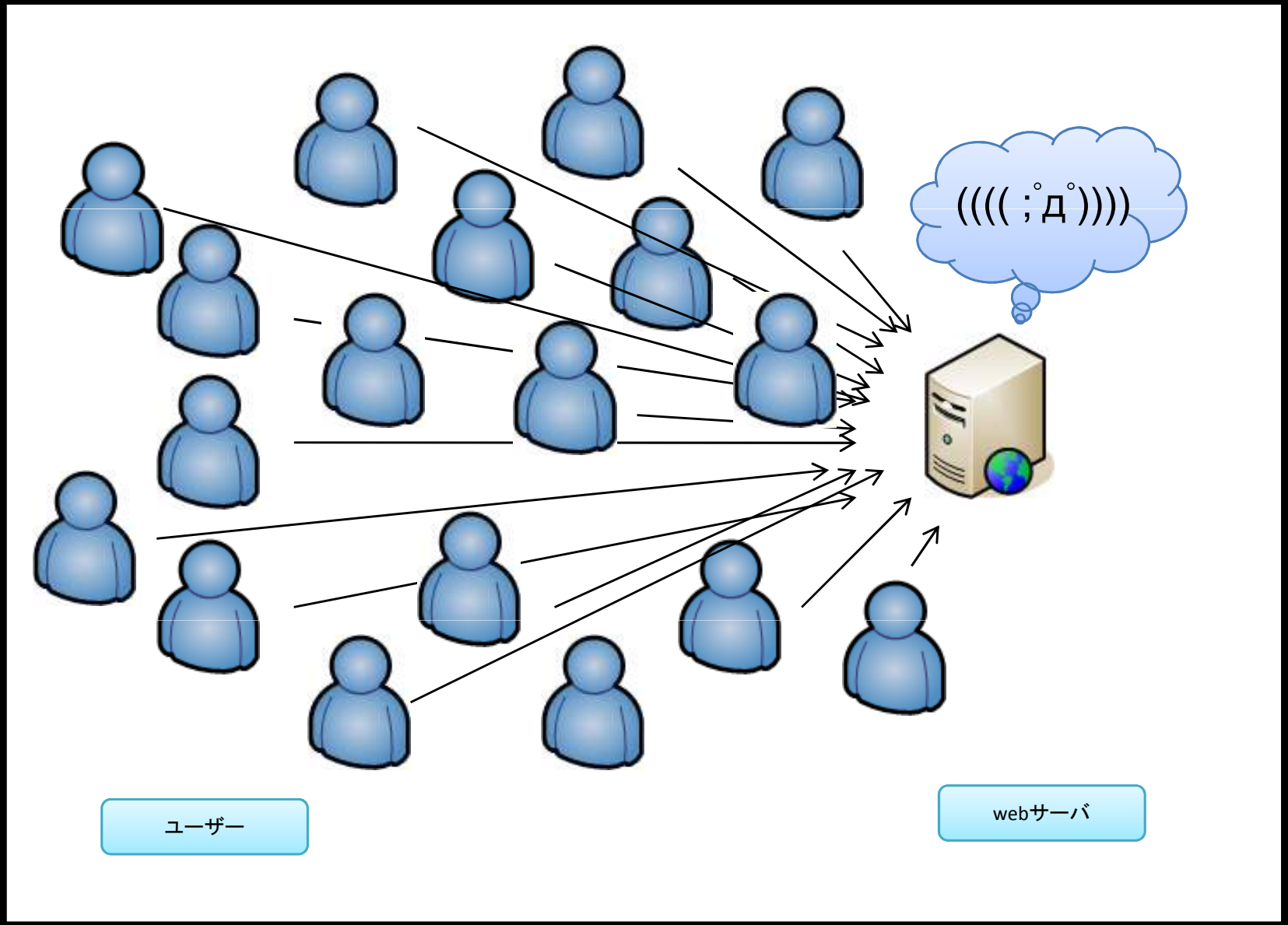
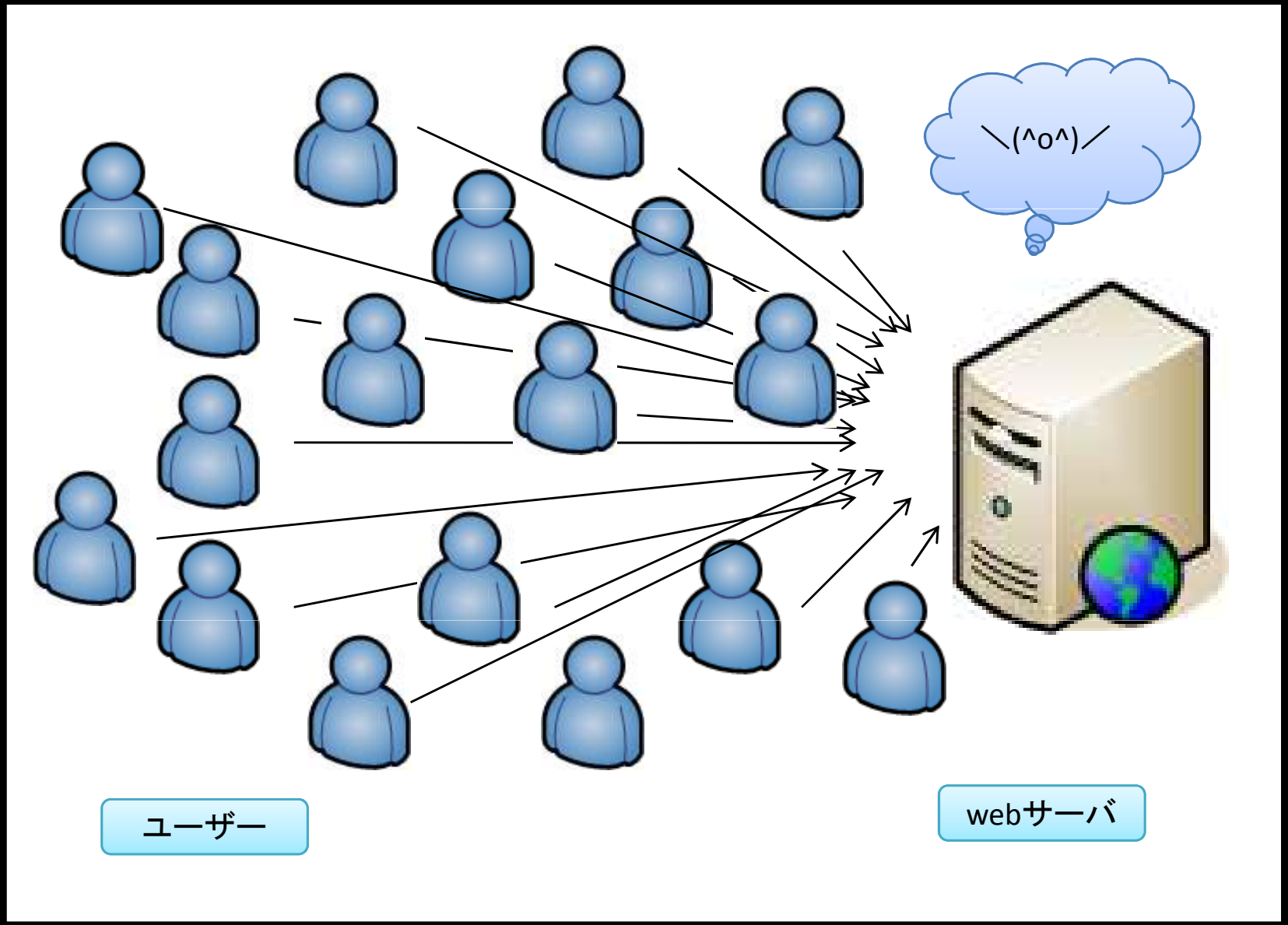
快適なwebアプリケーション構築のための基礎知識

増えていく負荷とサーバと機能に
どう向き合うかの巻

前回の復習

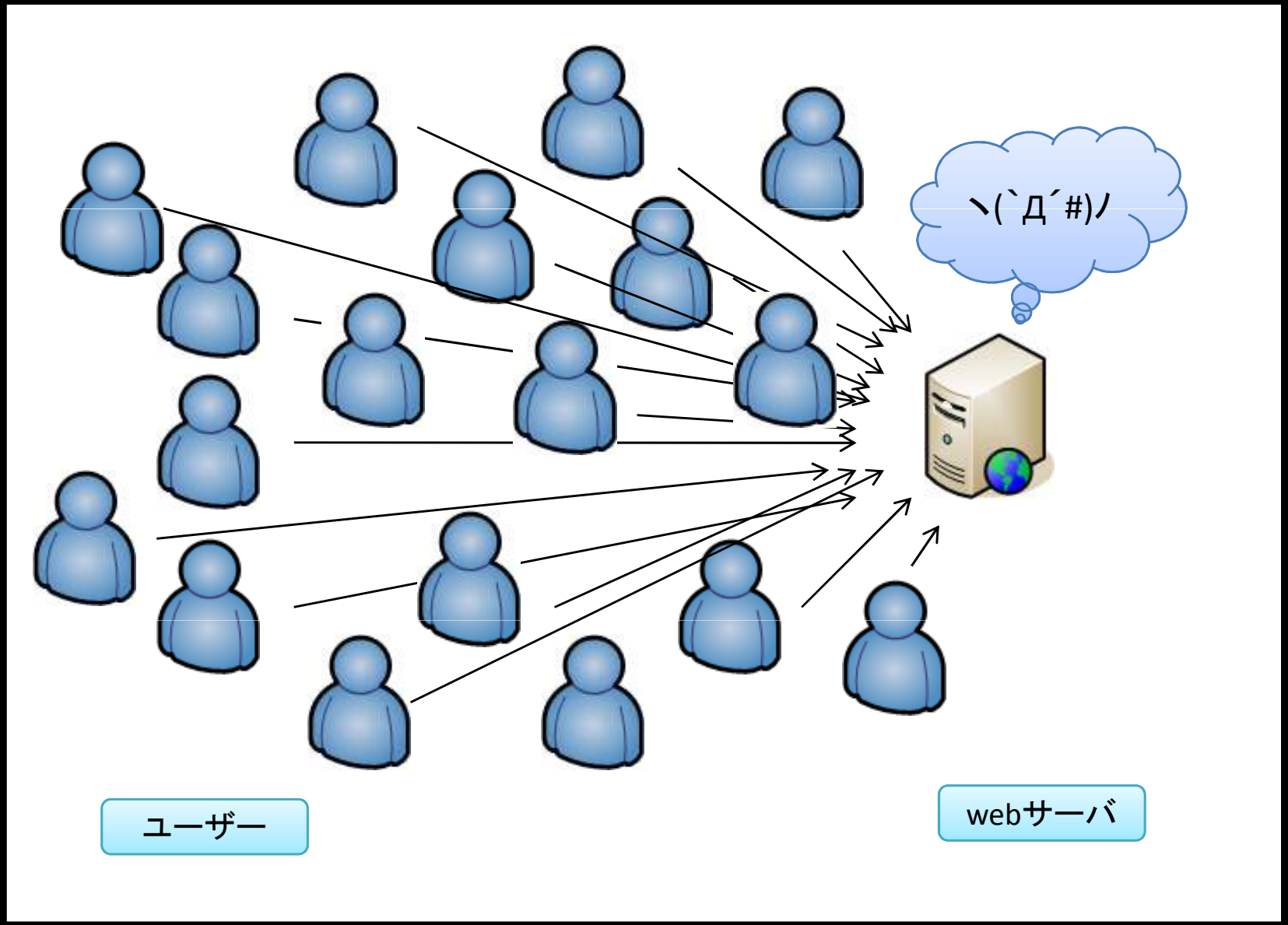
スケールアップ

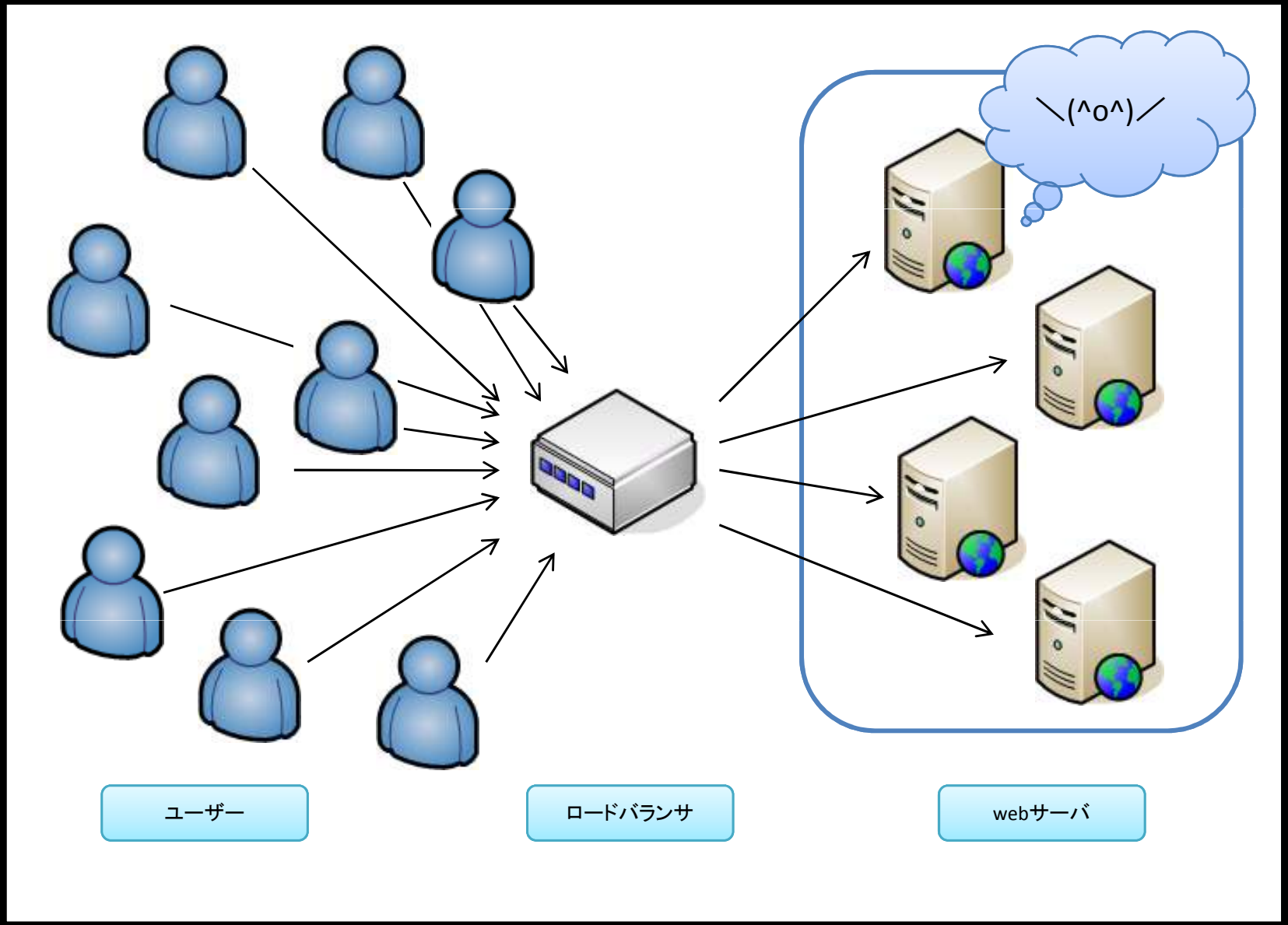

((((；゜д゜))))
ユーザー
webサーバ


＼(^o^)／
ユーザー
webサーバ

スケールアウト

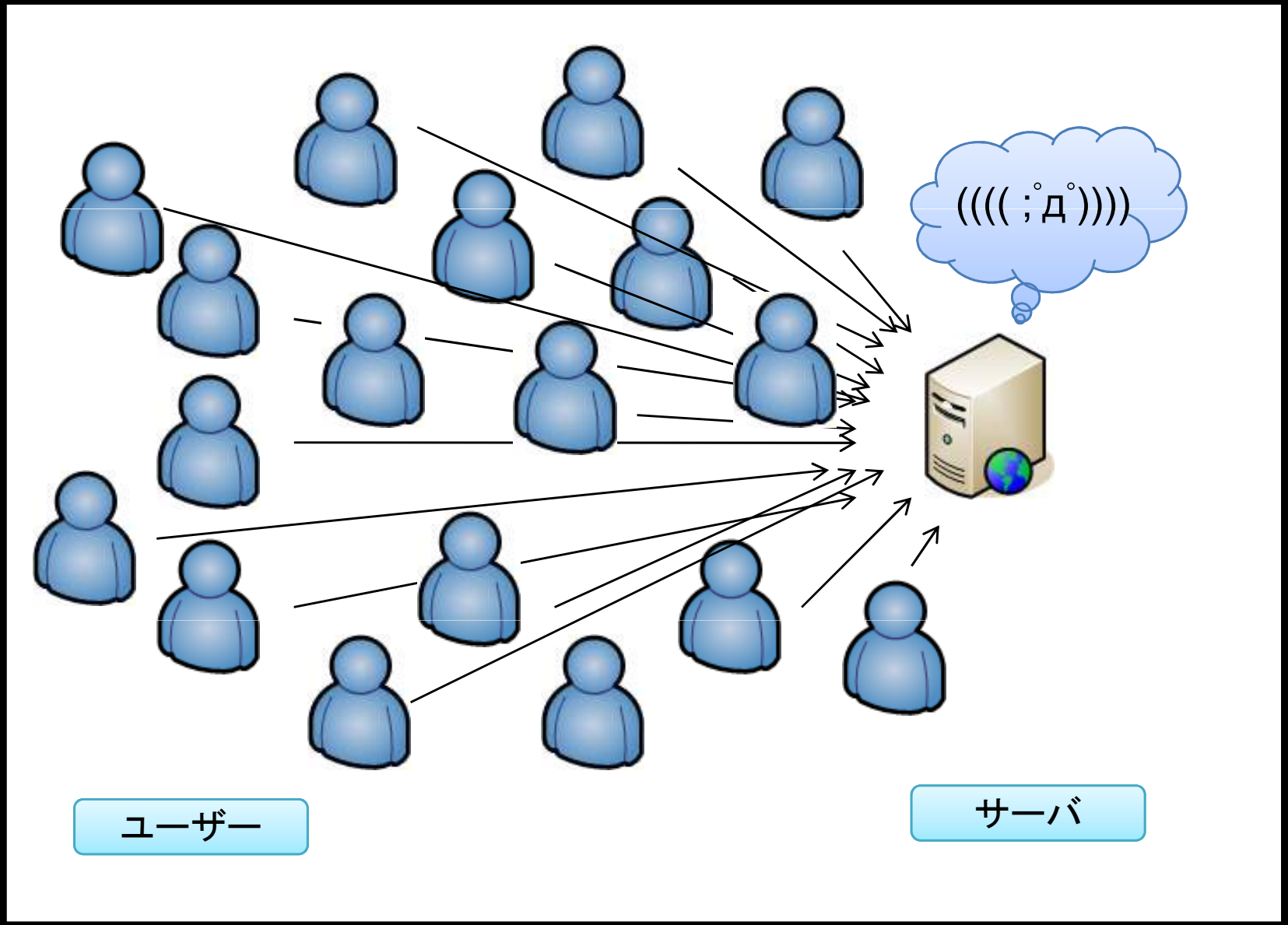
ロードバランサ


ヽ(`Д´#)ノ
ユーザー
webサーバ

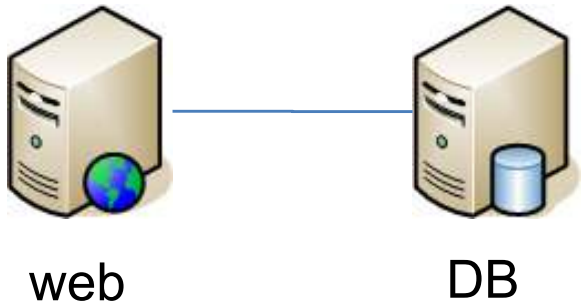

＼(^o^)／
ユーザー
ロードバランサ
webサーバ

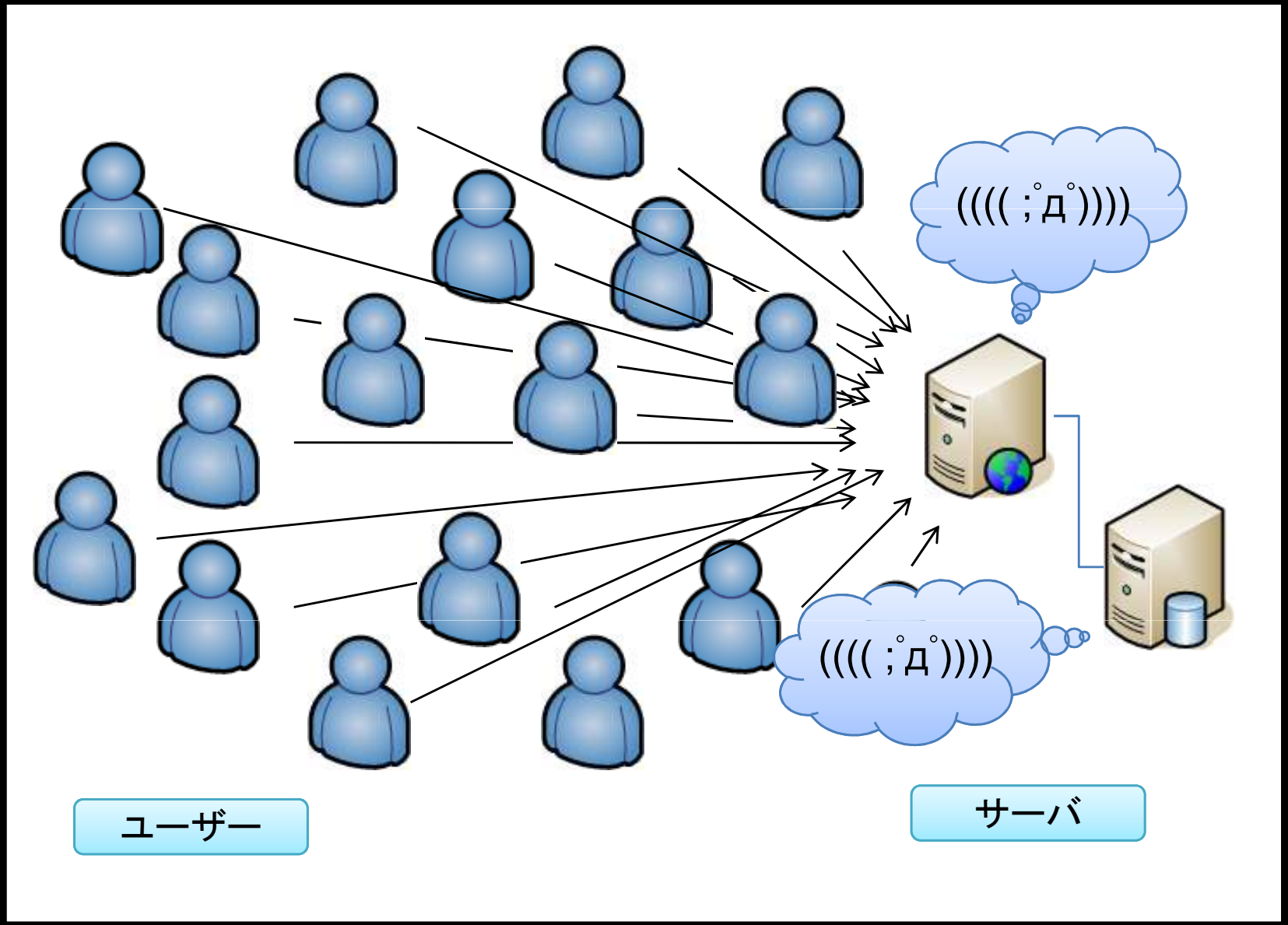
一般的にwebアプリケーションは
時間と共に機能とデータが
増えていきます

どうする？


((((；゜д゜))))
ユーザー
サーバ

1台追加したサーバ構成




((((；゜д゜))))
((((；゜д゜))))
ユーザー
サーバ

次、どうする？

今回の目標

システムを開発・拡張するときに、
思考停止しないようになる為の
入り口に立つ

何個か定番を知っていれば
問題にぶつかったときに
解決の糸口を見つけられる
はず…

インフラはどう拡張すべきか

アプリケーションはどう設計・実装するべきか

インフラ・アプリの設計が悪いと

- データ増加でパフォーマンス激しく低下
- 運用での負担大
- 改修が大変
- サーバが複数台になったときのみ発生するバグ

知っていれば回避できる
こともあるよね

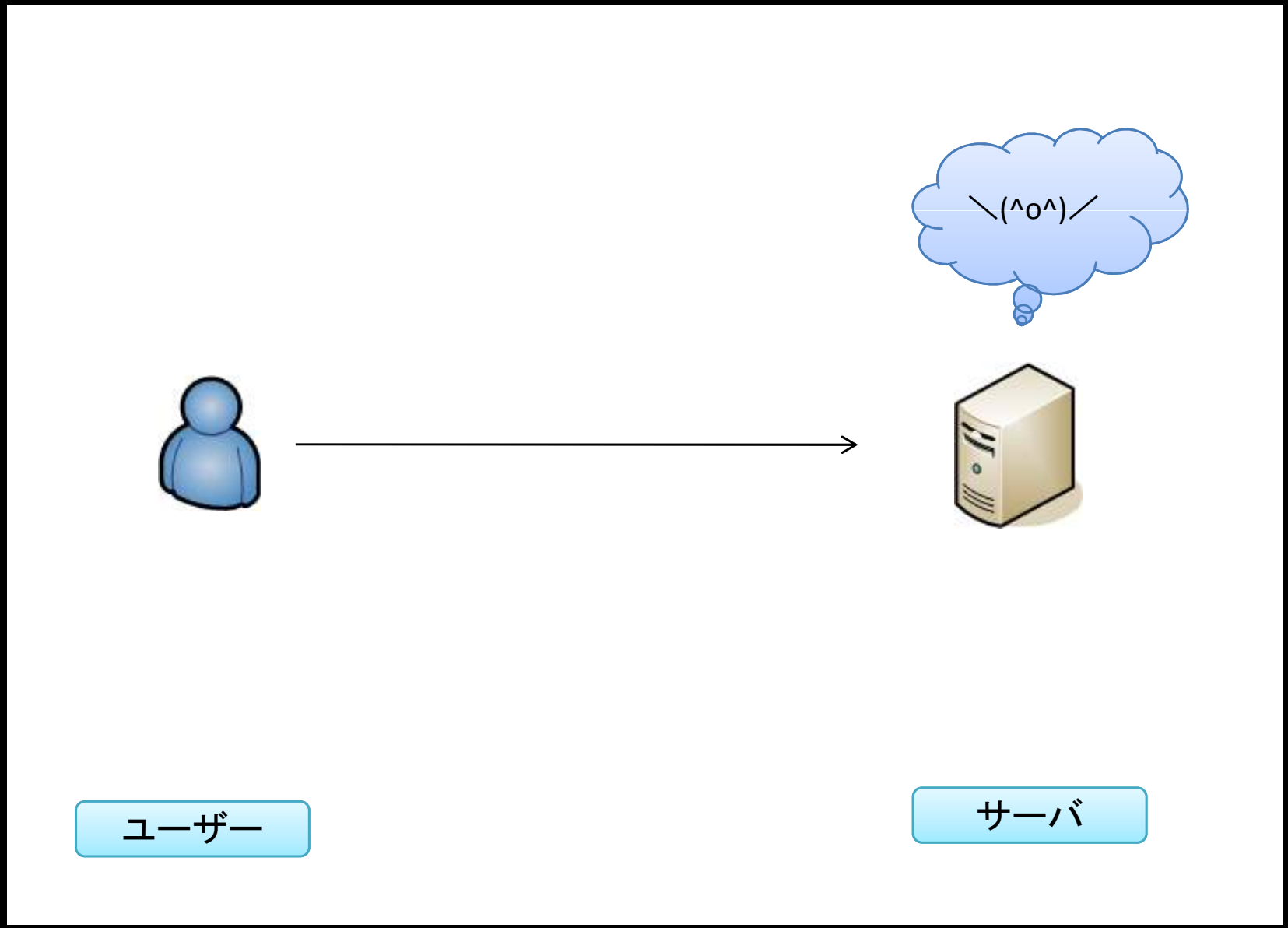
DBサーバの追加、webサーバの追加という実例を時系列で見てみましょう

(symfonyも交えて)

1日目

スモールスタート
サーバ1台

オールインワン
(web + DB)

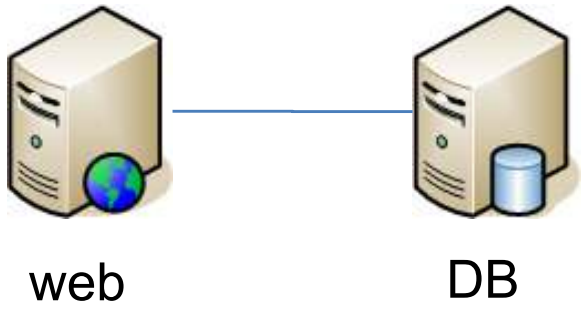

＼(^o^)／
ユーザー
サーバ

2日目


(((;ﾟдﾟ)))

アプリケーションの
チューニングって手も
あるけど

スケールアウトで


web
DB

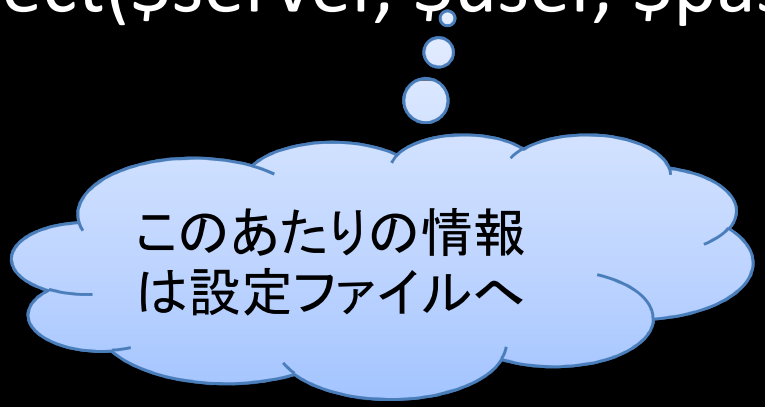
サーバの接続先を変更

DBの接続先情報は
ハードコードしない
設定ファイルとしてコードから
追い出しておく

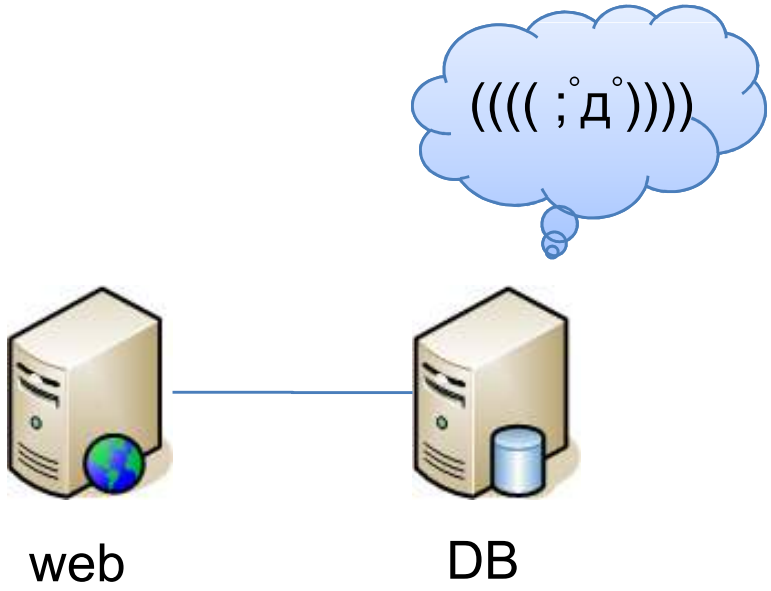
実装例

```
$link = mysql_connect('localhost', 'user', 'password');
```

```
$link = mysql_connect($server, $user, $password);
```



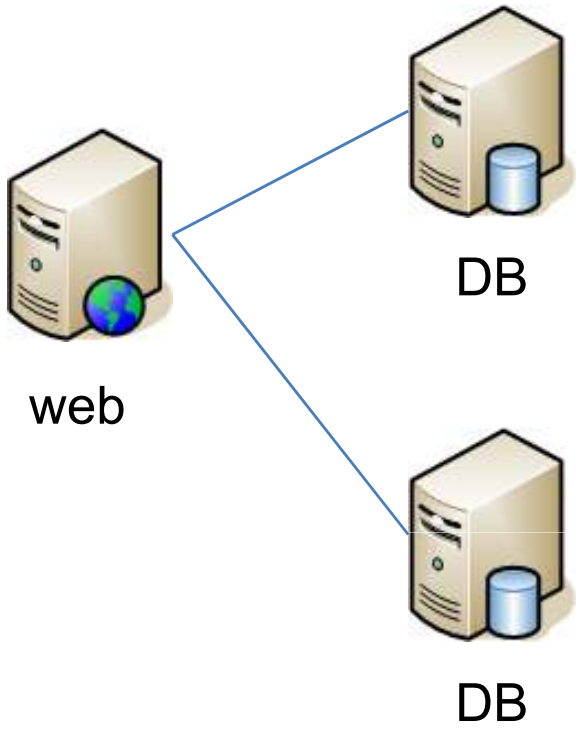
3日目


((((;゚д゚))))
web
DB

アプリケーションの
チューニングって手も
あるけど

ミドルウェアの
チューニングって手も
あるけど

スケールアウトで

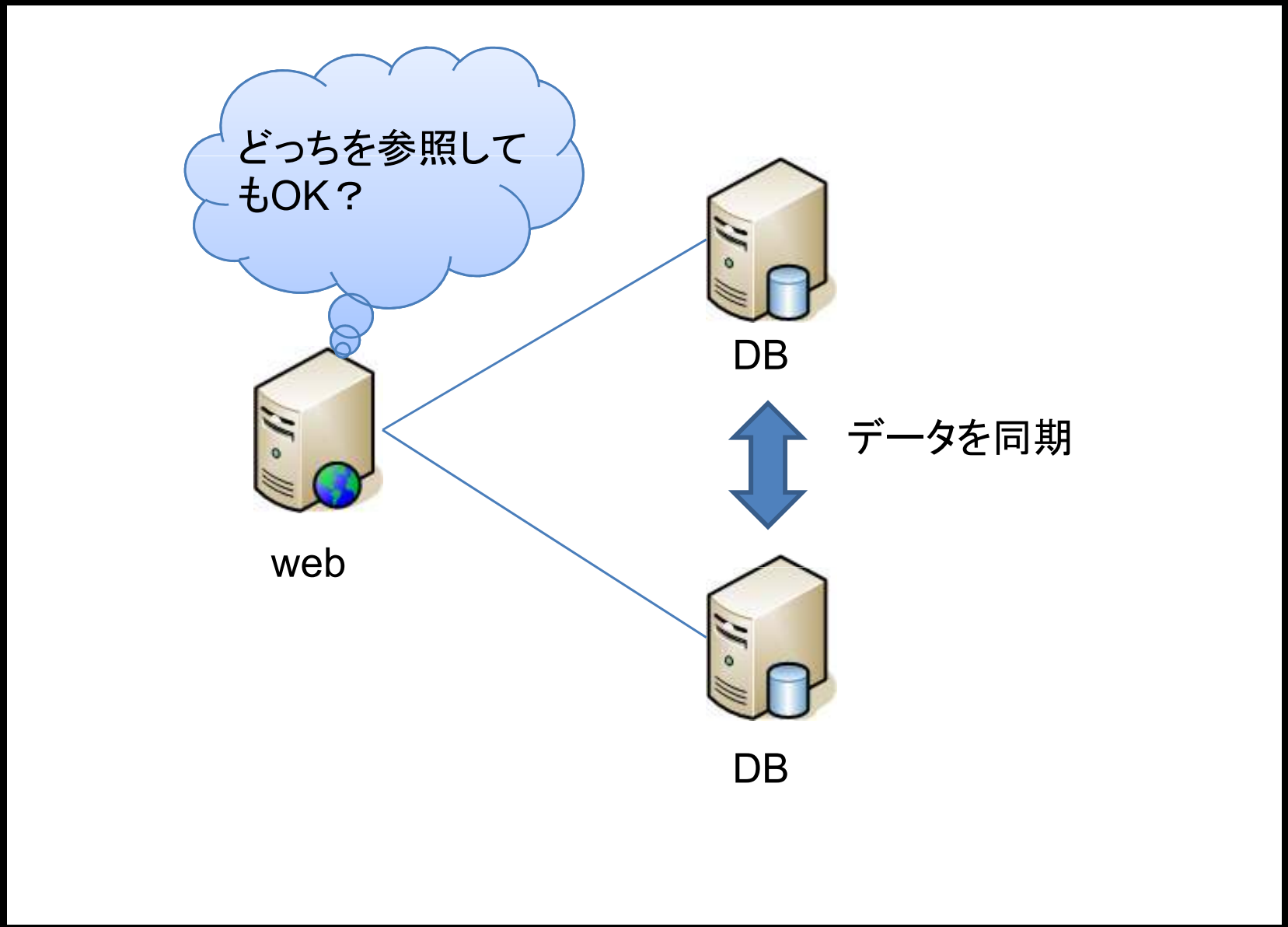

DB
web
DB

レプリケーション

データベース管理システムが持つ負荷分
散機能の一つ。
あるデータベースとまったく同じ内容の
複製(レプリカ)をネットワーク上に
複数配置し、通信回線や1台1台の
コンピュータにかかる負荷を
軽減する仕組みのこと。

出典：e-words.jp

みんなで同じデータを持って
みんなでやろうよ


どっちを参照して
もOK？
DB
データを同期
web
DB

マスターとスレーブ

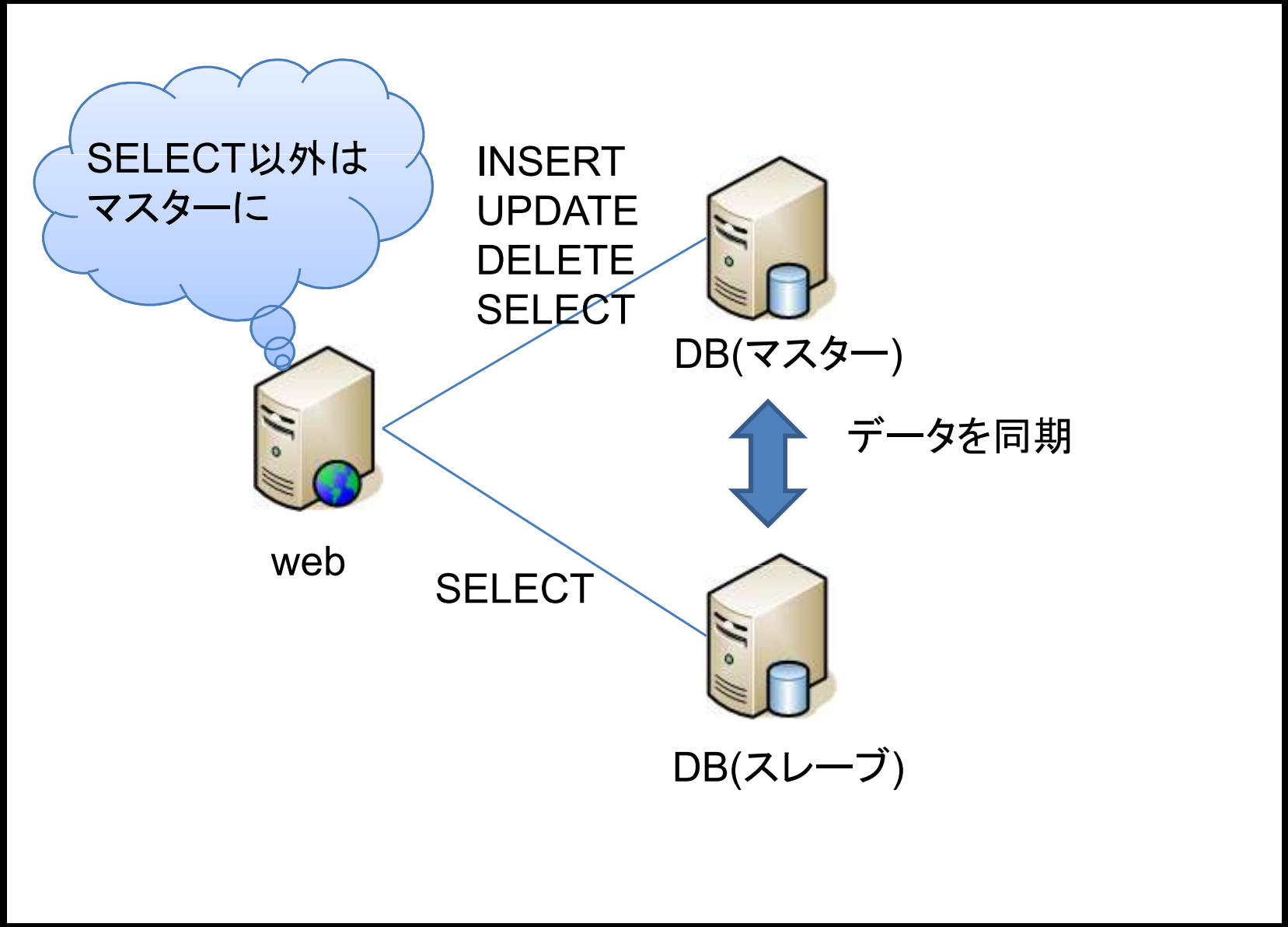
主人と奴隷？

主人 … 好き放題
奴隷 … ご主人様次第

マスターはデータの更新が
出来る
スレーブはマスターのデータ
を複製するだけ

例外
マルチマスタ

アプリケーションでは
INSERT, SELECT, UPDATE,
DELETEのなかで
SELECT以外は
マスターに対して行わうよう
実装しなければならない

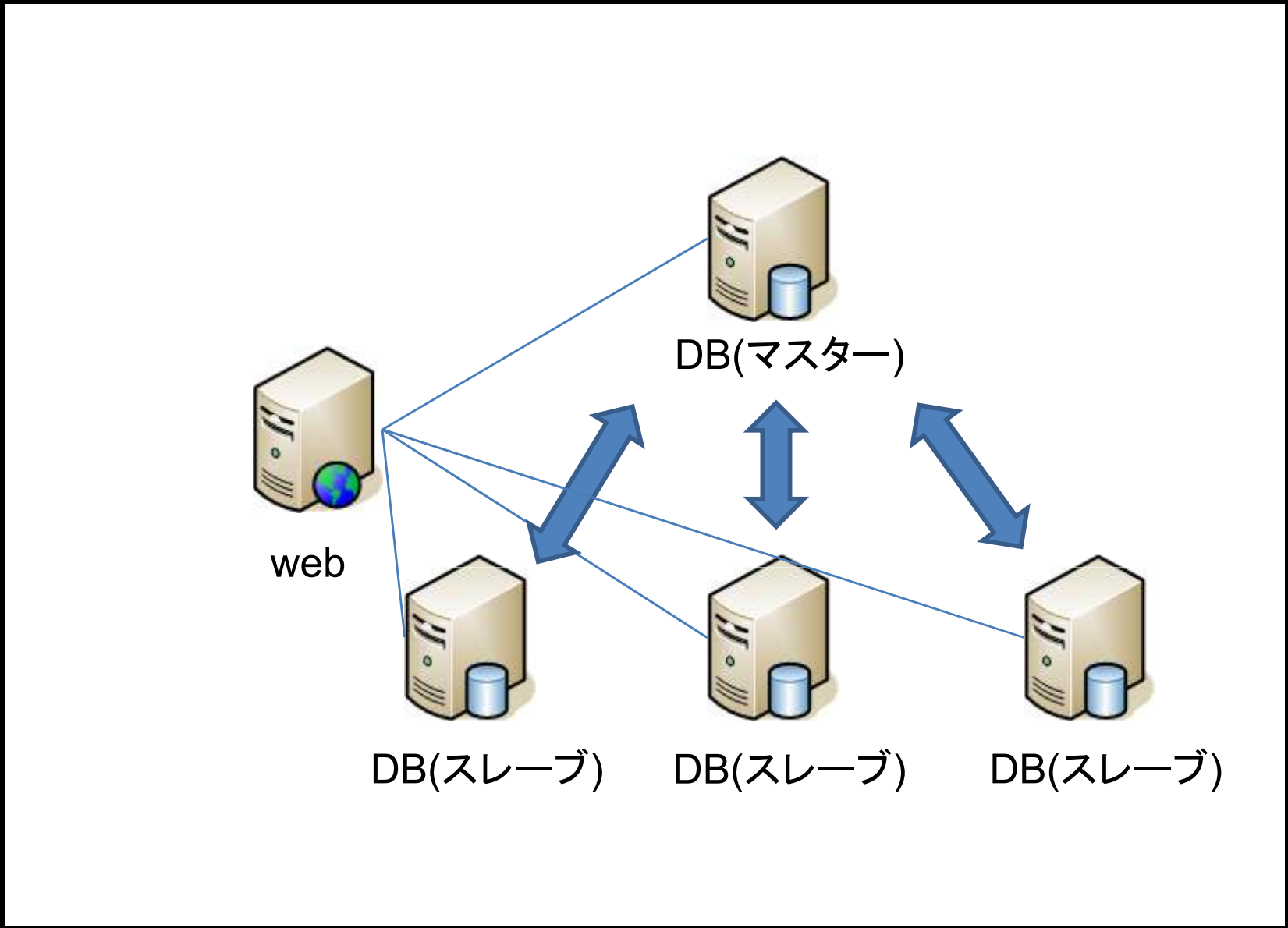

SELECT以外は
マスターに
INSERT
UPDATE
DELETE
SELECT
DB(マスター)
データを同期
web
SELECT
DB(スレーブ)

透過的にできるといいよね

symfonyではsfPropelLoadbalancerPluginというものがあるよ(α版だけど)

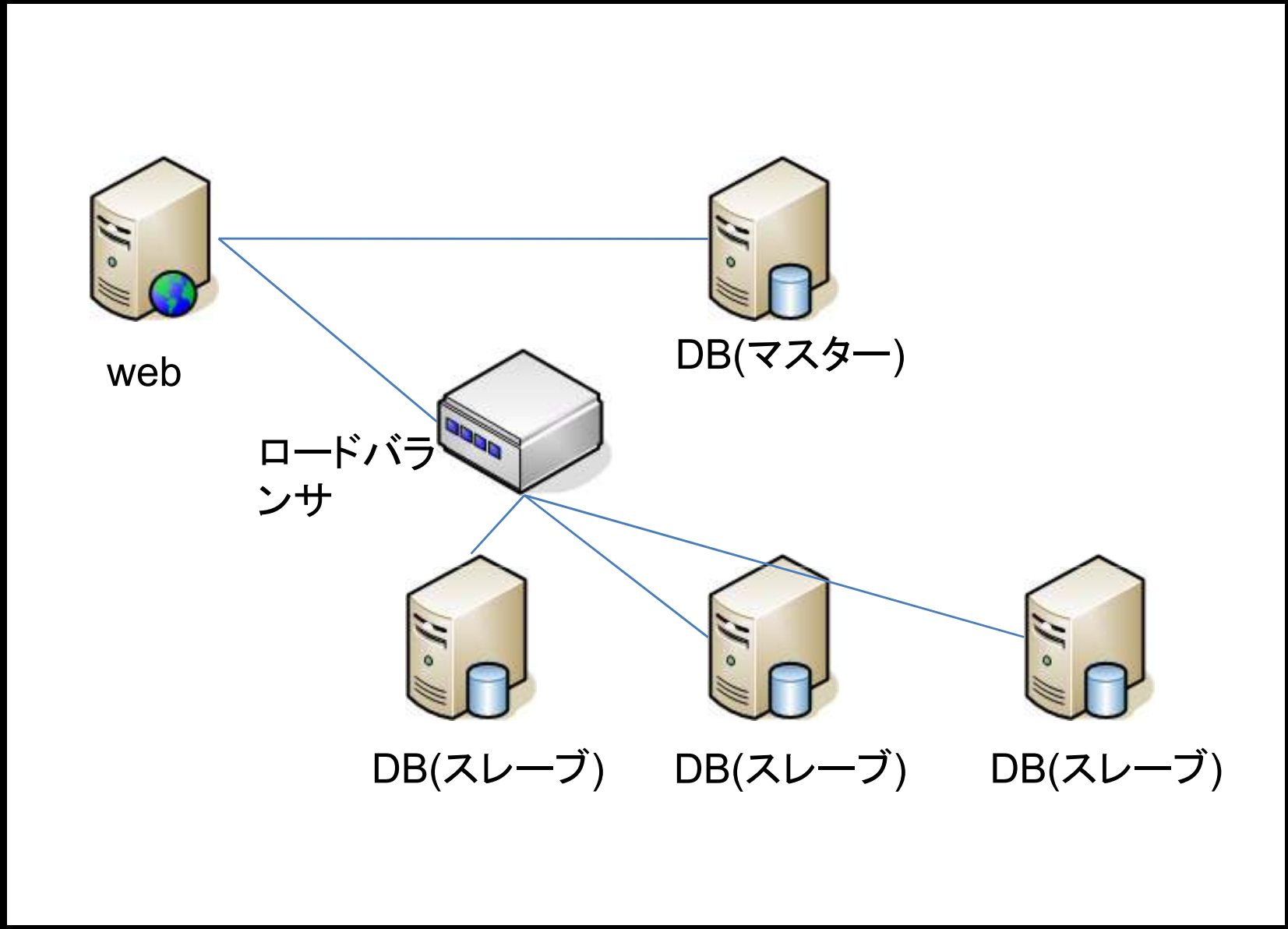
```
297  /**
298   * Returns the matching connection for a given query.
299   *
300   * This method checks the type of the query (SELECT or not) and
301   * returns master or a random slave connection.
302   *
303   * @param string Database query
304   *
305   * @return Connection instance.
306   */
307  public function balanceConnectionForQuery($query)
308  {
309    if (strtoupper(substr(ltrim($query), 0, 6)) == 'SELECT')
310    {
311      return $this->getSlave();
312    }
313    else
314    {
315      return $this->getMaster();
316    }
317  }
318
```

この実装が出来ていれば...


DB(マスター)
web
DB(スレーブ)
DB(スレーブ)
DB(スレーブ)

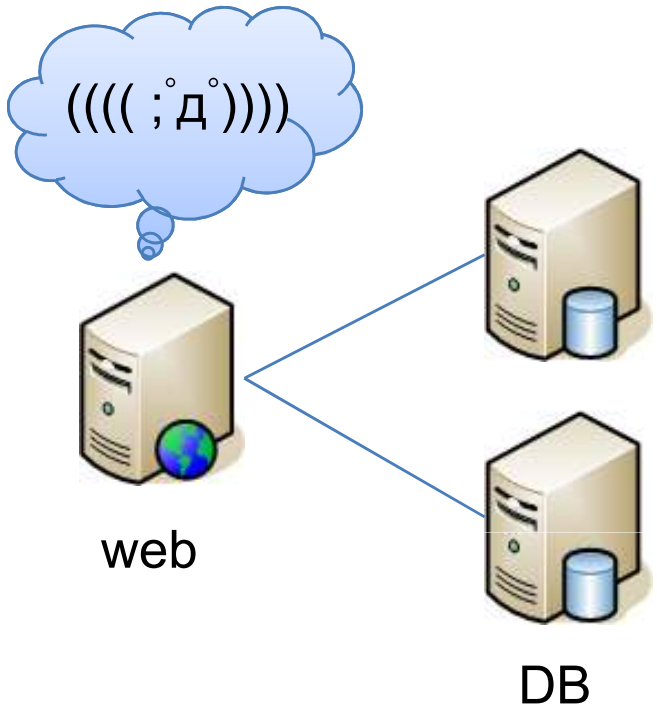
設定ファイルのDB情報
追加でおk

発展系

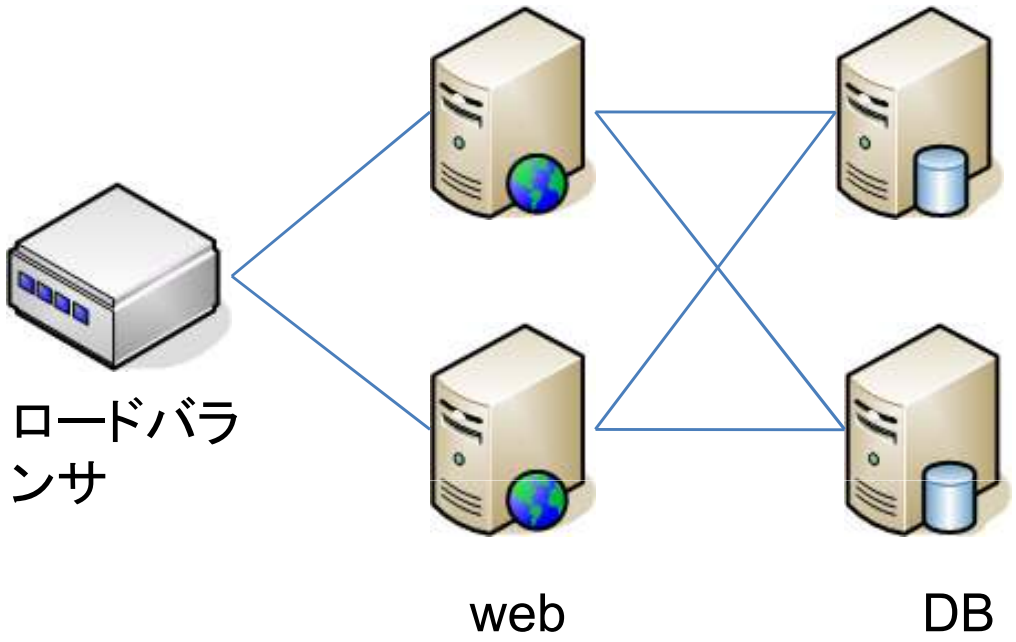

web
DB(マスター)
ロードバランサ
DB(スレーブ)
DB(スレーブ)
DB(スレーブ)

アプリケーションは
何も変更しなくてもおk

4日目


((((;゚д゚))))
web
DB

アプリケーションの
チューニングって(ry


ロードバラ
ンサ
web
DB

ロードバランサの事は
おいといて(前回やったし)

セッションの共有どうする？

セッションの共有？

セッションってどうやって
サーバ側では保存されて
いるの？

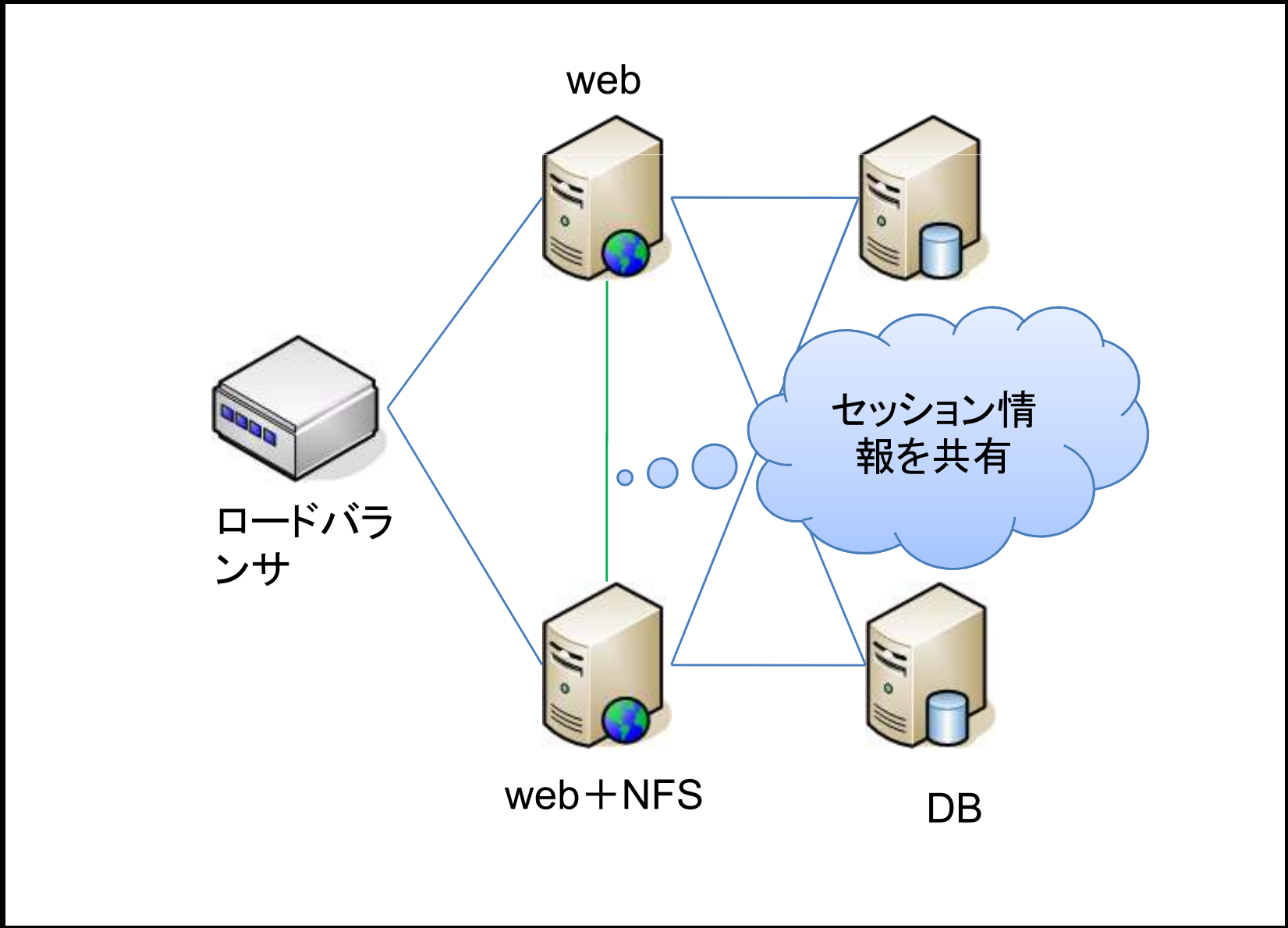
PHPではデフォルトはファイル

```
$ sudo ls /var/lib/php/session/
sess_0krrlo5oj2nau5hunjui0raia3
sess_0o13m5rvpeelhojrknkf0rmfb5
sess_0tclfvme1hoqivqlhpvt6bjcl2
sess_108bh2gqha2nhbmkofqvetell7
sess_11b8tloj8ea2sp06u0g0os8m01
sess_17o2hov23c2334f0gv5jlpbn00
sess_1bs481tn06q7ul59emrqletrd7
sess_1uvuln8fu3jonvadtmcc8k6rk4
sess_250ca4pgbnjm61jgqg0rcfhj33
sess_267h9rkqggi34o6t738l89qrf3
sess_2kp00t4davf7m8fdik4ee25ql7
```

何種類か解法を

ファイルならファイルを共有
してしまえばいいんじゃね？

解1 NFS


web
セッション情報を共有
ロードバランサ
web＋NFS
DB

php.iniのsession.save_path を
変えてNFSで共有

Good & Bad

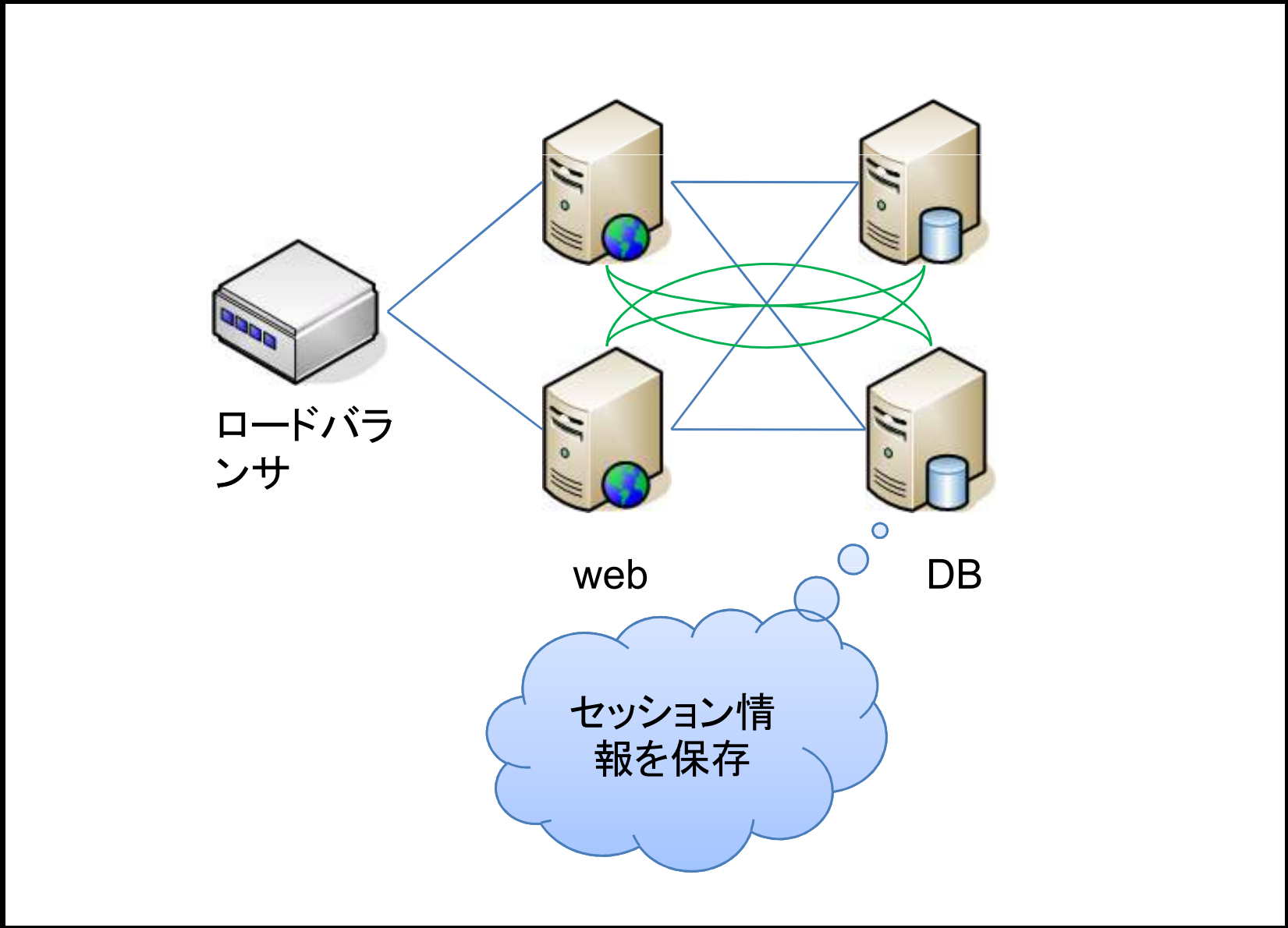
- アプリケーションの変更不要
  - NFSサーバ立てて、マウントして、設定変更してwebサーバ再起動でおk
- NFSサーバ落ちたらセッション使えないよ(SPOF)
- そもそもNFS(個人的に)？
  - 高負荷時にOSごとサヨナラとか素敵な思い出ができました

簡単だけど安定性と
冗長性に不安

他に解はない？

セッション情報が共有できればいいので、ファイルである必要はないんじゃね？

解2 DB


ロードバランサ
web
DB
セッション情報を保存

Good & Bad

- SQLが分かれば実装出来る
  - NFSよくわからんし
  - 他によさげなもの知らないし
- DBサーバのI/O負荷増える
  - 1リクエスト毎にDBの読み書き発生
  - 特にINSERT, UPDATEのマスター側
  - レプリケーションしてたらなおさら

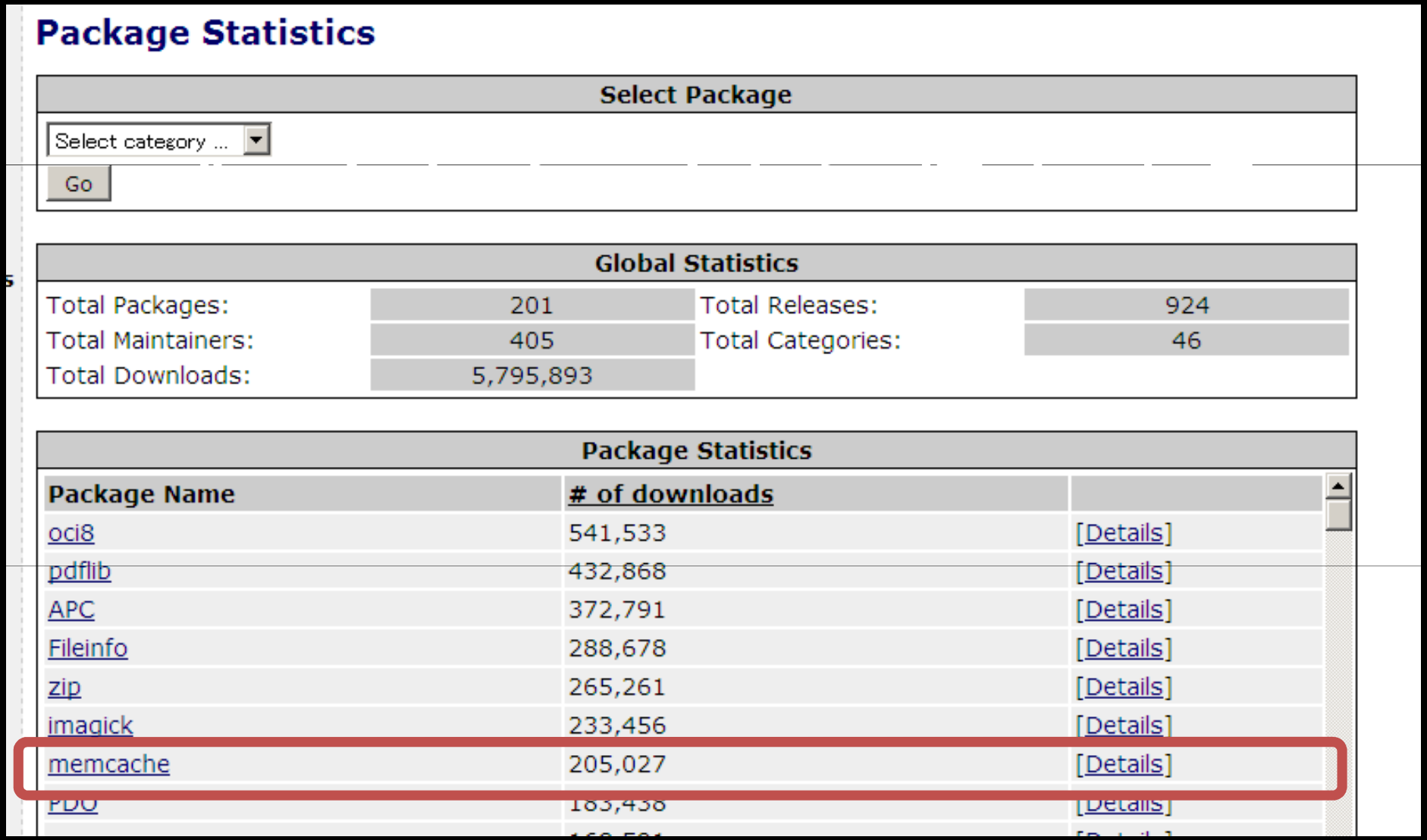
ぼちぼち簡単だけど
パフォーマンスに不安

セッション情報ってDBに
保存しておく程永続化する
必要あるの？

解3 memcached

データをメモリ上に保存する
ハッシュ型データベースサーバ

大人気



Package Statistics

Select Package

Select category ...

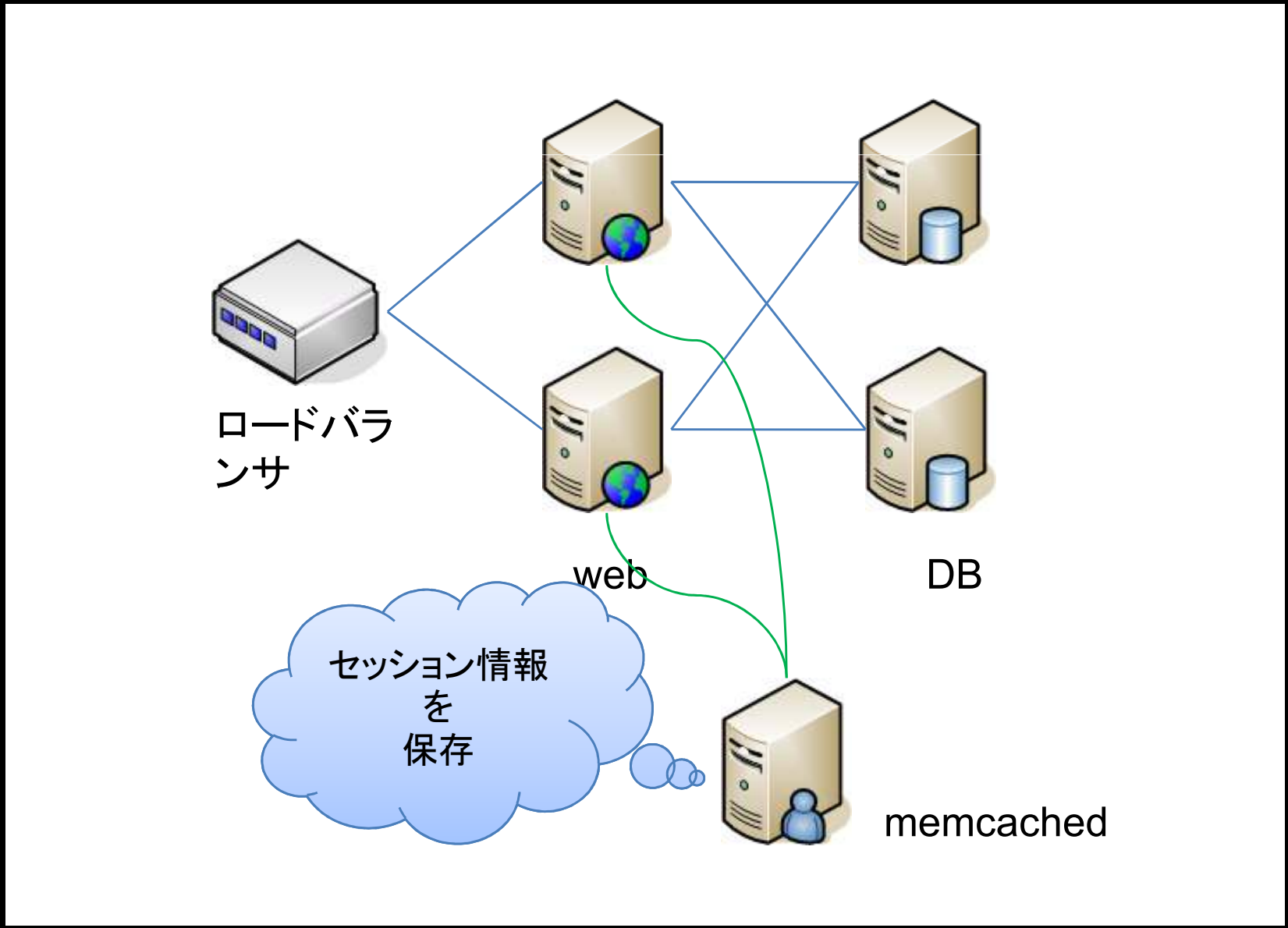
Go

Global Statistics

Total Packages:	201	Total Releases:	924
Total Maintainers:	405	Total Categories:	46
Total Downloads:	5,795,893		

Package Statistics

Package Name	# of downloads	
oci8	541,533	[Details]
pdflib	432,868	[Details]
APC	372,791	[Details]
Fileinfo	288,678	[Details]
zip	265,261	[Details]
imagick	233,456	[Details]
memcache	205,027	[Details]
PDO	183,438	[Details]


ロードバラ
ンサ
web
DB
セッション情報
を
保存
memcached

Good & Bad

- 速度
  - ハッシュのキーから検索してデータを返すだけなので速い
- 安定性
  - 2年ほど再起動なしで動いてます
- 可用性
  - 割り切って、なし。再起動でデータ消える
  - 複数台用意すれば負荷分散可能
  - 複数台用意すれば1台落ちても大丈夫
- それだけ
  - ハッシュのキーを保存・検索以外の事はできない

って事で、アプリケーション
の設計どうする？

PHPでセッションの保存の仕方(サーバ側)を変更するには

bool session_set_save_handler (callback $open , callback $close , callback $read , callback $write , callback $destroy , callback $gc)

やることは一緒だけど、
使うもの(ソフトウェア)が違う

インターフェースを決めて
同じメソッドをコールバックに
登録すれば差し替えが簡単

例えば

```
abstract class Session {
  abstract function open();
  abstract function read() ;
  abstract function write() ;
  abstract function close() ;
}
```

```
class FileSession extends Session {
  public function open(){ … }
  public function read() { … }
  public function write() { … }
  public function close() { … }
}
```

```
class MySQLSession extends Session {
  public function open(){ … }
  public function read() { … }
  public function write() { … }
  public function close() { … }
}
```

てな感じであれば、使用する
クラスを設定ファイル
に持っておく事で、
セッションの保存の仕方が
切り替えられる

単体テストもやりやすいよね

```
$session = SessionFactory('File');
$session->write(…);

$session = SessionFactory('MySQL');
$session->write(…);
```

今日はここまで

今日の所までのまとめ

- 同じ問題を解決するにもいくつものやり方があり、それぞれメリットデメリットがある
- スケールアウトさせて行くにはサーバ側とアプリケーション側両方での協力が必要。
- 言い換えるとサーバ側の構成をアプリケーションで抽象化しておくような設計がgood。
  - フレームワークはなぜ冗長なのか
- 先に知っておけば、後から修正しなくてもよいよね

アプリケーションと
インフラ(サーバ)の
バランスを見て最適な解を
出せるようになれれば
いいね(+予算も)

次回も同じような感じで
アプリケーションの機能追加
に伴うインフラ構成の変更
など見ていきましょう

ご静聴ありがとう
ございました

質疑応答

ネットワークチームはスタッフ
を募集しています。
興味を持った人は是非
一緒にスケールアウト
しましょう！