PHPで
大規模ブラウザゲームを
開発してわかったこと

LOCAL PHP部
ke-tai.org
松井　健太郎

自己紹介

松井　健太郎

- LOCAL PHP部　部長
- 株式会社インフィニットループ　代表
- ke-tai.org　管理人
- コーラが好き

自己紹介(2)

http://ke-tai.org/

本日の内容

1. ソーシャルゲーム大流行
2. 困った！ゲームなんて作ったことないぞ！？
   - 通常の案件とゲーム製作の違い
3. なぜPHPを使ってるの？
4. ゲーム向けの基本設計とサーバ構成
5. PHP実装のポイント
6. データベース設計で注意するところ
7. これが一番の問題！負荷対策について
8. その他こんなところに苦労した

ソーシャルゲーム大流行

- それまでも兆しはあった
  → ブラウザゲームというジャンルは以前から存在していた
  → 海外での利用者数増加

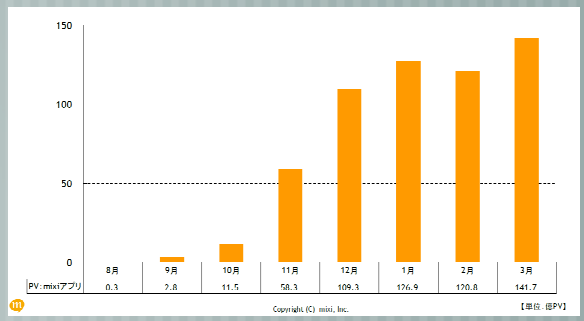
- はじまりはmixiアプリから
  → 2009年8月mixiアプリオープン
  → 2009年10月mixiアプリモバイルオープン

サンシャイン牧場

- 国内ウェブコンテンツ制作の業者がこぞってゲーム開発に参入

- モバゲー、GREE、大手ゲーム会社などを巻き込んでの大戦争に

- 2010年には1000億円市場になると言われている



2010 mixi IR資料より

ブラウザ三国志について

運営(株)AQインタラクティブ

開発ONE-UP(株)

プログラム開発を(株)インフィニットループが担当

2009年6月サービス開始

2010年8月mixi版サービス開始

会員数約80万人以上（2010年5月現在）

月間PV数　約2億6千万PV（2010年5月の実績）
(http://donnamedia.shoeisha.jp/site/detail/4392より)

ブラウザ三国志について(2)

特徴

- インストール不要でブラウザ上だけで遊べるゲーム
- 他のプレイヤーと同盟を組み、都市を育てたり戦争を行う
- いわゆる村ゲーと言われるジャンル
- HTML(PHP)とJavaScriptのみで製作されている

困った！ゲームなんて作ったことないぞ！？

- 弊社にとっても初のゲーム開発
  - それまでに作ったことがあるもの
    - ▼ウェブサイト向けの動的コンテンツ
    - ▼業務用のウェブアプリケーション
- 調べてみると基本動作は同じ
  - ▼HTMLとCSSで作られたページの遷移
  - ▼DBへの登録・修正・削除
  - ▼JavaScriptによる装飾処理
- サーバ構成もウェブアプリケーションと大差は無い
  - ▼通常のLAMP構成
  - ▼ただし負荷対策にはしっかりとした考慮が必要

→ *通常のウェブアプリケーションの基礎がしっかり出来ていれば大丈夫*

なぜPHPを使ってるの？

- もちろん弊社がPHPが得意というのはあるが…
- 数多くの高負荷運用の実績がある
- 開発スタッフ集めが容易（開発スピードが求められる）
- 処理速度がわりと高速、APCなどのアクセラレータで更にスピードアップ
- プログラム更新の容易さ。メンテナンスなしでゲームを動かしながら柔軟にアップ作業ができる
- PHPだけが特段ゲーム向きの言語というわけではない

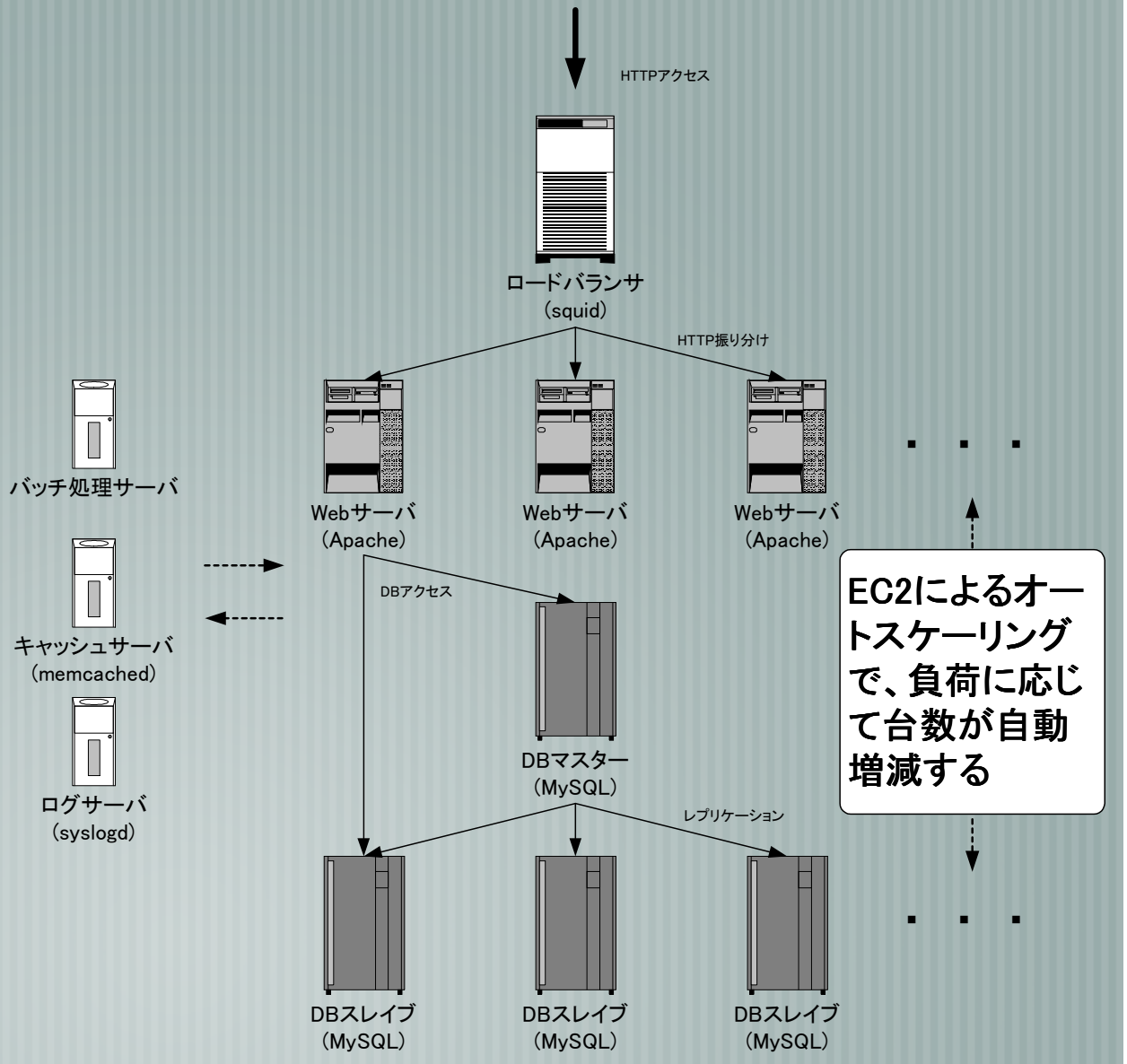
基本設計で気をつけたいところ

- 特に通常案件と大きく変わるわけではない
  - いわゆるLAMP構成（Linux, Apache, MySQL, PHP）
- 運用後どれだけ利用者が増えるか予測がしにくいので、いつ人が増えても破綻をきたさない設計
  - 「利用者数が増えてきてから対処」では間に合わない
  - サーバを追加することで対処可能なように
- 利用数の増加スピードには大きな波がある
  - ソーシャルアプリならではのクチコミやランキング入りなど
  - サーバ追加の容易さが求められる
- 落とさないサーバ作りは大変
  - 重要サーバのみを守り、他はどんどん落とすという方針

サーバ構成例

Amazon EC2上に構築

- squid（ロードバランサー）
- Apache
- PHP+APC
- MySQL
- memcached
- syslogd（ログサーバ）



PHP実装のポイント(1)

- 既成フレームワークは使っていない
  → 負荷が高すぎた際に手が出せない恐れがあるため
  → ただし工数削減のために導入もあり
- テンプレートエンジンにはSmartyを利用
  → 遅いと言われているがベンチを取ったところ別に遅くない
  → デザイン連携がしやすい
- クラス化が重要
  → 通常の案件と比べると、遥かにクラス化の効果が高い
  → ゲームは再利用が多いので、クラス化と相性が良いと思われる
  → 例えばポーションクラスを継承してハイポーションのアイテムを作成など

PHP実装のポイント(2)

- ログはしっかり取る
  → PEAR::Logを使用している
  → ログレベルを可変にできるように
  → 規模が大きくなるほどバグの把握と再現が難しくなる
- 実行されるSQLはすべて管理できるようにする
  → ごく一部の遅いSQLが、DBサーバの負荷の大半を占める
  → O/Rマッパーなど自動でSQLが生成されるようなものは利用しない
- DBのロックはしっかり行う
  → 当たり前だがテーブルロックは使えない
  → 行ロック(FOR UPDATE)を使用
  → しっかり行わないと同時操作や高負荷時にのみ起こるバグが残ってしまう
- その他大きく変わったことはしていない、とにかく基本が大事

データベース設計の注意(1)

- 基本的には通常の案件と同じでOK
  → ゲームならではの特殊な設計があるわけではない
  → 負荷を意識して設計することが必要
- テーブルは必ずしもキレイに正規化されるわけではない
  → 高負荷の部分は、あえて冗長な構造にしたり、キャッシュテーブルを設ける
- DB容量の増加はなるべく抑えるようにする
  → データ量を少なくするのは、地味なようで効果がある
  → なるべくオンメモリで動作するように
  → DBスレイブをRAMディスク上で動作させることができる
  → ログテーブルなどはバッチで定期的に書き出して削除する

データベース設計の注意(2)

- ロックを考慮して設計
  → ロックの使用方法によってはCPUパワーを使いきれない
  → ロック待ちをなるべく減らすように
  → 使用するデータベースソフトや分離レベルによって挙動が微妙に異なるのを理解する

例：プレイヤーのステータス（HP, MP）を格納するテーブルがあったとして、左のように設計してしまうと、player_id単位で行ロックがかかるため、HPとMPを同時に更新できない。
（下記の例だと魔法を使いながら他者から攻撃を受けることができない）

```
SELCT * FROM player_tbl WHER player_id = x FOR UPDATE;
```

これが一番の問題！
負荷対策について

- 負荷対策にも色々ある
- ウェブサーバの負荷対策
  → CPU負荷の軽減
  → 転送量軽減
- DBサーバの負荷対策
  → DBスレイブの負荷対策
  → DBマスターの負荷対策
- キーバリューストアの導入
  → memcached導入の悩ましい問題

負荷対策について
ウェブサーバの負荷対策(1)

- Apacheのチューニング
  → httpd.confなどの設定値を見直しチューニングする
- PHPアクセラレータを導入
  → これ無しの運用は考えられない（入れない理由がない）
  → 比較・評価の結果、APCを利用している
- スケールアウトで対応
  → ウェブサーバの台数を増やし、それをLBで振り分けることで対応
  → セッションの共有はmemcachedを利用
  （他にもDBで共有、LB側で対応などの方法がある）
- PHPプログラムの改善でウェブサーバの負荷を下げるのは結構難しく、地道な作業が必要

負荷対策について
ウェブサーバの負荷対策(2)

- アプリサーバとは別に専用の画像サーバを用意する
  → 画像、CSS、JavaScriptはアプリサーバとは別に設置する
  → アプリ側で画像参照先を切り替えられるようにしておく
  例：

- 転送量を減らす
  → ob_gzhandlerで出力を圧縮
  → 画像のサイズを小さく
  → CSSやJSはツールを利用して圧縮する
  → HTMLのコメントやインデントは削除

  ※Smartyのコメント化してしまうと良い
  例： <!– comment ->　→　{*<!– comment ->*}

負荷対策について
ウェブサーバの負荷対策(3)

- 画像などのキャッシュは強めに設定する

  例:
  ExpiresActive On
  ExpiresDefault "access plus 3 days"

- 任意でキャッシュクリアできる仕掛けを用意する

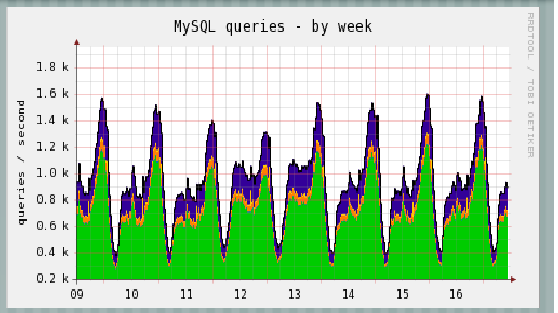
  例:

  のように画像パスに変更可能な文字列を入れておく
  画像更新のタイミングでインクリメント

  シンボリックリンクをはるか、mod_rewriteで
  日付の部分を無視するようにしておく。

```
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteBase /
  RewriteRule ^[0-9]{8}-[0-9]+/(.*) $1
</IfModule>
```

負荷対策
DBスレイブの負荷対策



- MySQLのチューニング
  → my.cnfなどの設定値を見直しチューニングする
- スケールアウトで対応
  → MySQLのレプリケーション機能で台数を増やして対応
- マスターサーバからスレイブサーバの伝達遅延対策
  → これが起こるのは非同期レプリケーションの宿命
  → データベースの容量はなるべく小さく、オンメモリ内に収まるように
  → RAMディスクやSSD上にDB領域を作成して対応
- 重いクエリを修正する
  → スロークエリのログをチェックし、重い順に対策していく
  → これをひたすら繰り返す（最初のうちは負荷はみるみる下がる）

負荷対策
DBマスターの負荷対策

- DBマスターの負荷対策は難しい
  → スケールアウトがしずらい
  → マスターサーバだけ強化しても、スレイブがついていけない

- 当たり前のことを淡々と
  → 不要なクエリが実行されていないかを調査
  → スロークエリのログを取って、重い順に対策していく
  → トランザクションの時間をなるべく短くなるようにプログラムを調整

- それでもダメなら
  → ゲームの仕様で調整できる部分がないか検討する
  → マスター分割を検討、ただし開発工数の増加を招きやすい
  → MySQL Clusterってどうなんだろう

キーバリューストアの導入

- キーバリューストア（memcachedなど）の導入は意外と悩ましい
- ゲームではキャッシュが使いづらい
  → 参照がほとんどのWebコンテンツ（例えばニュース、掲示板など）とは違う
  → キャッシュするのは簡単だが、キャッシュクリア処理が難しい
- トランザクション中はキャッシュを使いづらい
  → トランザクション中はmemcached上のデータをあてにできない
  → 利用できるのは変更が入らないマスタ類だけ
  → DBマスターサーバの負荷軽減にはあまり繋がらない
  → DBのロールバックに対する対処が必要
- 実装やデバッグのコストが増えやすい
  → サーバ代と開発とどちらにコストをかけるかの問題
  → 実はDBもうまく使えば結構早い
  → DeNAもMySQLメインのアプローチでやってるらしい？

参考：@IT Special PR：600億PVもMySQLで！ モバゲーのインフラ底力(http://www.atmarkit.co.jp/ad/dena/dena1004/index.html)

その他こんなところに苦労した

- APCの不可解な挙動
  → APCのキャッシュが更新されず、プログラムがアップされない
  → touchコマンドでタイムスタンプを更新すると直る
  → rsyncに「--no-times」オプションを付けることでひとまず解決
- プログラム規模が大きくなることへの対処
  → コーディング規約とルールをしっかりと策定することが大事
- AmazonEC2のトラブル
  → ゲームとクラウドの相性は良いが、まだまだ安定していない
  → 国内含め他のクラウドも似たりよったり、まだまだこれから
- 技術者としてのこだわりと、実装のコストと妥協
  → プログラムが汚くなることもあるよね
  → 基本動いているものがえらい
  → 時には仕様を曲げてもらうことも必要かも

FAQ(1)

Q. ゲーム開発は難しくないですか？

A. 簡単か難しいかと言われたら、結構難しいです。
テスト環境で動かすだけなら比較的簡単ですが、実際にユーザを入れてちゃんと動くものを作るのは大変です。
今もわからないことがいっぱいです。

Q. AmazonEC2を使用しているそうですが、使ってみてどうですか？

A. サーバがすぐに増設できる、負荷に合わせて自動でサーバを増減させることができる、など長所も多くありますが、安定度という点ではまだまだと思います。
適材適所で使い分ける必要があるのではないでしょうか。

FAQ(2)

Q. ゲーム開発は面白いですか？

A. はい、通常案件の倍増しくらいで面白いです。
技術的な面はもちろんですが、コミュニティなどで利用者の反応がすぐに見られる点が特に楽しいです。仕事中に堂々とゲームができるというメリットもあります（笑）

Q. バグ技を教えてください

A. バグの内容にもよりますが、悪用できるようなバグはすぐ対処されるため、知りません。
もしバグ技を見つけたらブログなどには書かないで、どうかこっそりと運営までご連絡をお願いします m(_ _)m

FAQ(3)

Q. ゲームを作る上で一番重要なことは何ですか？

A. クライアントから言われた一言をご紹介します。

「既存WEBエンジニアからのゲーム転身で大事なのは、
参考ゲームをやって面白さを理解してくれること。
ゲームは仕様を読んでもわからない。」

まとめ

- ソーシャルゲーム開発は難しいところもあるが面白い。ゲーム開発の経験がない人でも意外と何とかなる。
- 負荷対策が大変。基本は台数でカバーできる作りに、それ以外はできることから淡々と。
- 設計は開発コストとサーバコストのバランスの見極めが重要。
- 基礎が大事、負荷対策やロックなどの基本的なところをしっかりと身につけることが必要。

最後に
LOCAL PHP部のご紹介

LOCALの一部のメンバでPHPのイベントをゆるく、
ながくやっていこうか、という会です。

大体二ヶ月に一度くらいのペースで、
札幌市内で勉強会を開催しています。

LOCAL PHP部のページ:
http://php.local.gr.jp/

Googleグループ:
http://groups.google.co.jp/group/local-php