クラウド時代のWebストレージ戦略

(株)ライブドア
池邉 智洋

livedoor

クラウド時代？

- クラウドってなんだろ？
  - SaaS
  - PaaS
  - IaaS
  - パブリッククラウド
  - プライベートクラウド

livedoor

利用者から見たクラウド

- 物理的なサーバ、ネットワークを意識しない
- 高可用性
- 柔軟な課金体系

ストレージ: Amazon S3, CloudFiles ...

livedoor

提供者から見たクラウド

- 高価な機器に依存しない＝
  スケールアウト
- リソースを有効活用する事によるコスト効率
- 管理、運用の手間が少ない

livedoor

ライブドア的クラウド

- 物理サーバリソースは既に投資済
- 提供しているサービスはSaaS的なもの
- さらにコスト効率をあげたい
- (I|P)aaS的なクラウド環境を自分達で作って自分達で使う
- いわゆるプライベートクラウド

livedoor®

実例

ライブドアが提供しているサービス

livedoor ニュース

livedoor Blog

livedoor wiki

Blogger Alliance

Cosplay Community Site Cure

livedoor Reader

livedoor PICS

livedoor グルメ

※画像をアップロードさせるサービスが非常に多い

livedoor

ライブドアが求めるストレージ

- Webサービスのメディアストレージ
- 安価に容量を拡張可能
- データの冗長化/高可用性
- 実際の構成は「あまり」意識したくない

livedoor

作ってみた

STF

livedoor

名前の由来

Step over

Toe h

with Face Lock

livedoor

STFとは?

Pros

- シンプルな key-value
- クライアントからは巨大なストレージプールとして見える
- 普通のサーバを使用して拡張可能
- データのレプリケーション
- Apache モジュールベースなので拡張可能

livedoor

Cons

- ファイルシステムとしてマウント出来ない
  →コードの書き換えが必要
- 既にあるオブジェクトに追記出来ない
- オブジェクトの一部書き換えが出来ない
- パーミッション/アクセス制御出来ない
  →自分達で使うだけなのであまり細かくても、、

Design

- ファイル群を束ねる Bucket の概念
- ディレクトリの概念は無く、Key-value
- REST的な API で操作
- 認証は他の Apache モジュール

livedoor

REST API

- リソースに対してリクエストを発行
- GET/PUT/DELETE をサポート
- バケットの作成/削除
- オブジェクトの取得/作成/更新/削除
- レプリケーションレベル等はヘッダで指定

livedoor

Example

http://stf.example.com/<bucket>/<key>

- <key> **は / を含む事が可能**
- URI **は** OS **のファイルシステムとは独立した論理**URI
- /a/b/abc....fe.jpg **のような分割は不要**

livedoor

プロトコル

Request

```
PUT /ikebe/picture.jpg HTTP/1.1
X-Replication-Count: 3
Content-Length: 10240
Content-MD5: f05a…

<content-body>
```

Response

```
HTTP/1.1 201 CREATED
```


livedoor

STF Internals

livedoor

基本戦略

極力コードを書きたくない！！

- 作りながらサービスに投入
- 大事な部分は枯れたコードを使いたい

livedoor

基本戦略

- Apache モジュールを C で実装
- URI → 物理ロケーションのマッピングは全て MySQL に保存
- MessageQueue を利用した非同期処理の活用
- 非同期処理のワーカーは Perl で記述

使用しているソフトウェア

- Apache 2.2
- Perl
- MySQL
- memcached
- ActiveMQ or Q4M

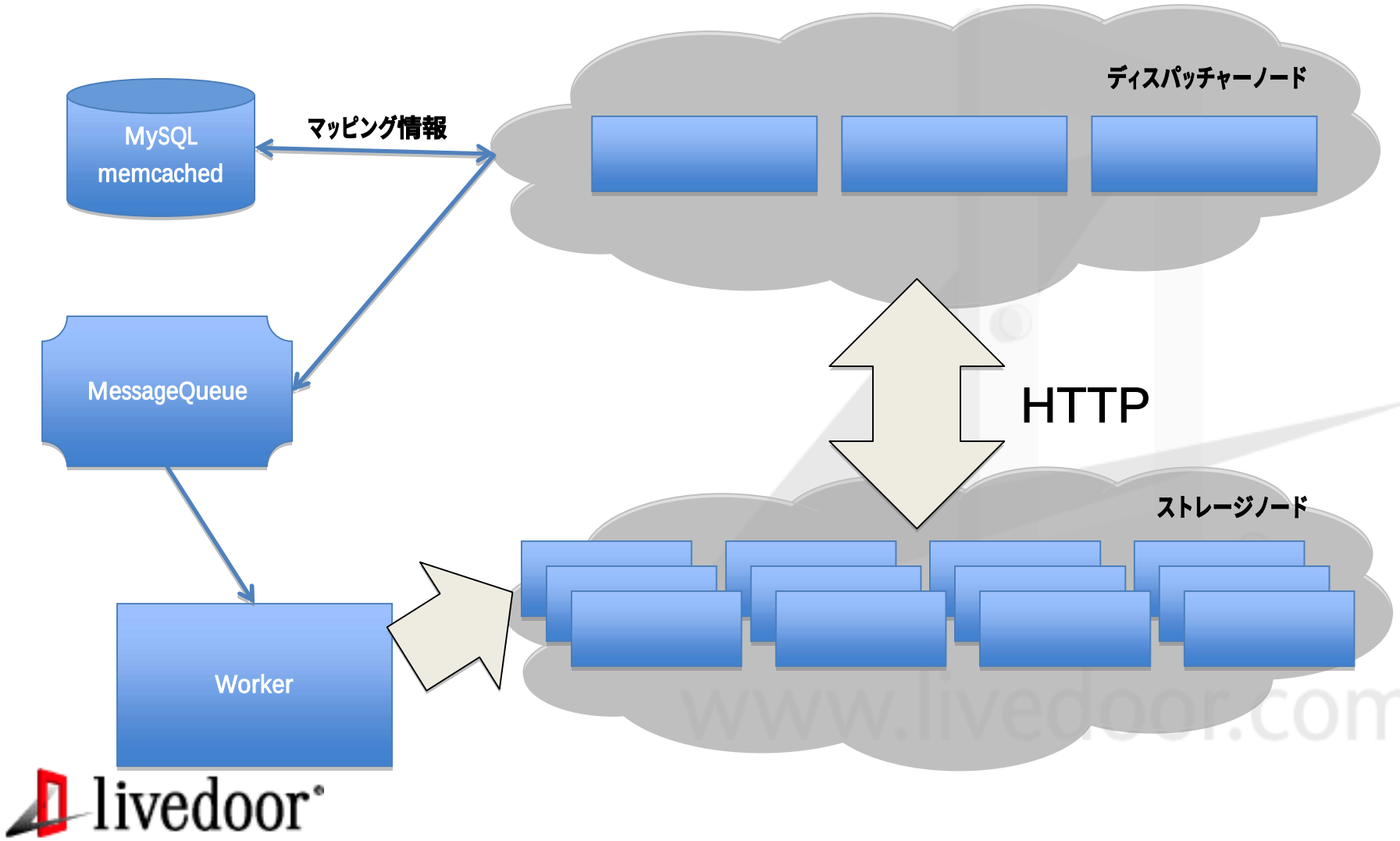
livedoor

CAP定理

- Consistency
- Availability
- Partition Tolerance

livedoor

構成



MySQL

- URI → **実体のマッピング/メタ情報**
- **テーブル一覧**
  - storage **ストレージノードマスタ**
  - bucket **バケット**
  - object URI ⇔ **オブジェクトID**
  - entity **オブジェクトID ⇔ ファイル実体**
- memcached **にキャッシュ**

livedoor

mod_stf.c

- **バックエンドストレージへのリクエストディスパッチャー**
- MySQL**と通信**
- MessageQueue **と通信**
  - ActiveMQ → Stomp
  - Q4M → libmysqlclient
- **認証は別の** mod_auth_* **で行う**

livedoor

mod_stf_storage.c

- GET/PUT/DELETE メソッドを受けつけてファイルの読み書き
- GET は default-handler 任せ
- PUT は Recursive にディレクトリを作成
- 同一ファイル名での上書きは出来ない
  - 更新は一回消して内部的には別ファイルを作成

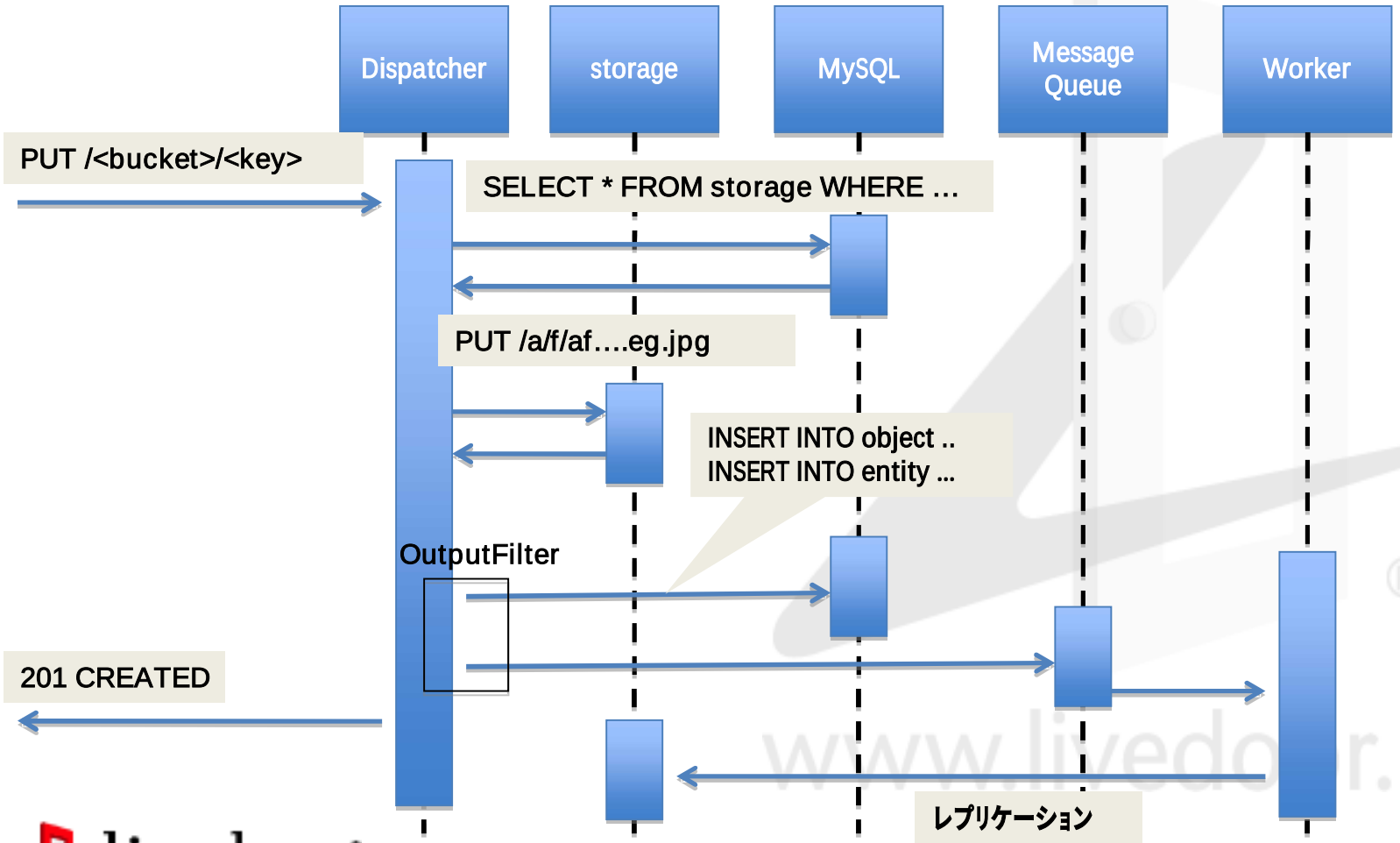
livedoor

非同期処理

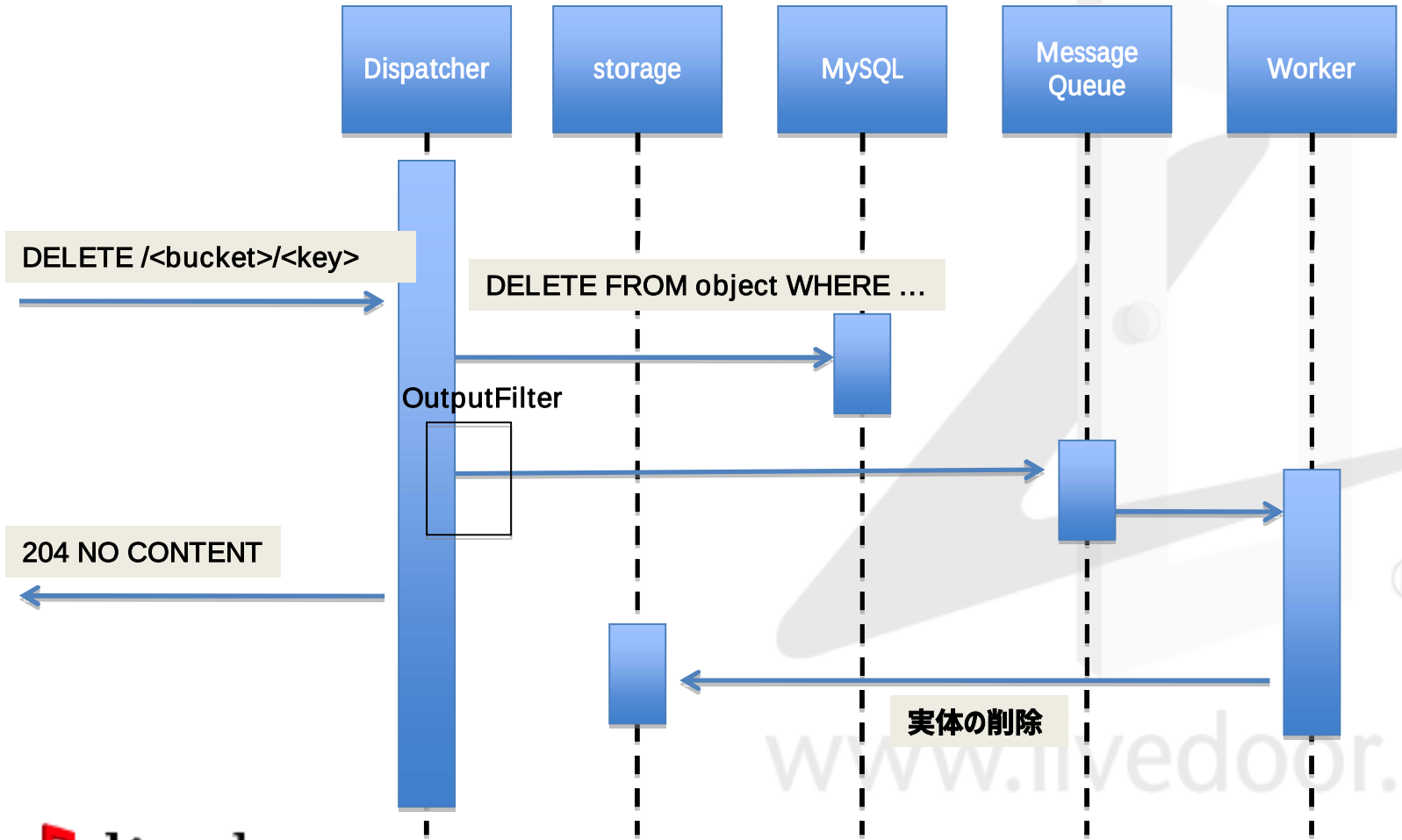
- Eventual Consistency(結果整合性)
- MessageQueue 経由
  - レプリケーション
  - オブジェクトの実体削除
  - バケット削除時の再帰的削除
- 定期実行
  - 使用容量の計算

livedoor

オブジェクトの作成



オブジェクトの削除



Web IF

- **ストレージ管理用の** Web IF
  - Catalyst
  - **ストレージの追加**
  - **ストレージ利用状況の確認**
  - **ストレージのモード切り替え**
    - rw, ro, down, crash

livedoor

Web IF

STF Web Interface

Storage Nodes
Buckets

ID	URI	使用領域	DISK容量	Mode
1	http://10.0.207.49:8080	101.5G	400.0G	rw
2	http://10.0.207.48:8080	102.0G	400.0G	rw
3	http://10.0.207.47:8080	102.1G	400.0G	rw
4	http://10.0.207.46:8080	102.0G	400.0G	rw
5	http://10.0.207.45:8080	102.0G	400.0G	rw
6	http://10.0.207.44:8080	101.9G	400.0G	rw
7	http://10.0.214.196:8080	101.8G	400.0G	rw
8	http://10.0.214.192:8080	101.6G	400.0G	rw

ストレージの追加

livedoor

耐障害性

- crash ノードの排除
  crashマークをつけられたノードがある場合レプリカ数の調整を行う
- なんらかの理由でレプリケーションに失敗 → Repair プロセスで再試行

livedoor

拡張機能

- 用途を限定して単純なストレージより便利に
- 画像の縮小の取り扱い
- 複数のサイズのファイルをあらかじめ作成しておく事が一般的
  - アップロード処理が複雑に
  - 大量のパターンがある場合(携帯)
  - サービスの仕様変更

livedoor

mod_small_light

- **ダイナミックに画像を変換するApacheモジュール(アウトプットフィルター)**
- Imlib2 or ImageMagick **を使用**
- **変換結果を** Squid **にキャッシュする事によりパフォーマンスを稼ぐ**

livedoor

TODO

- MySQL 依存からの脱却
  - マッピング保持に RDBMS はオーバースペック + SPOF気味(Dual Masterですが、、)
- 特定のノードの使用(SSD対応)
- オープンソースでの公開
  - mod_small_light
  - STF
  - ソースの整理が必要w

livedoor

ありがとうございました

livedoor