第2回 広域センサネットワークとオーバレイネットワークに関するワークショップ
2008年 11月 1日, 慶應SFC

ときに

オーバレイによる分散キャッシュ

～ computing cloud を支える基盤ソフトウェア

首藤 一幸

Overlay Weaver

背景

- key-value store、特にオンメモリのものが、OSS、エンタープライズ問わず、広く使われ始めている。
  - オープンソース ソフトウェア
    - memcached
    - JBoss Cache
  - エンタープライズ向け
    - Oracle Coherence
    - IBM ObjectGrid
  - 他
    - Amazon's Dynamo
    - 楽天 ROMA
    - 分散ハッシュ表 (DHT) 実装
      - Overlay Weaver とか
- 使い方
  - RDB から得たデータのキャッシュ。
  - 有効期間が短い (volatile) データの保持。
    - e.g. HTTPやSIPセッション情報, 座席予約, 株式・債権取引, 最終ログイン時刻, …

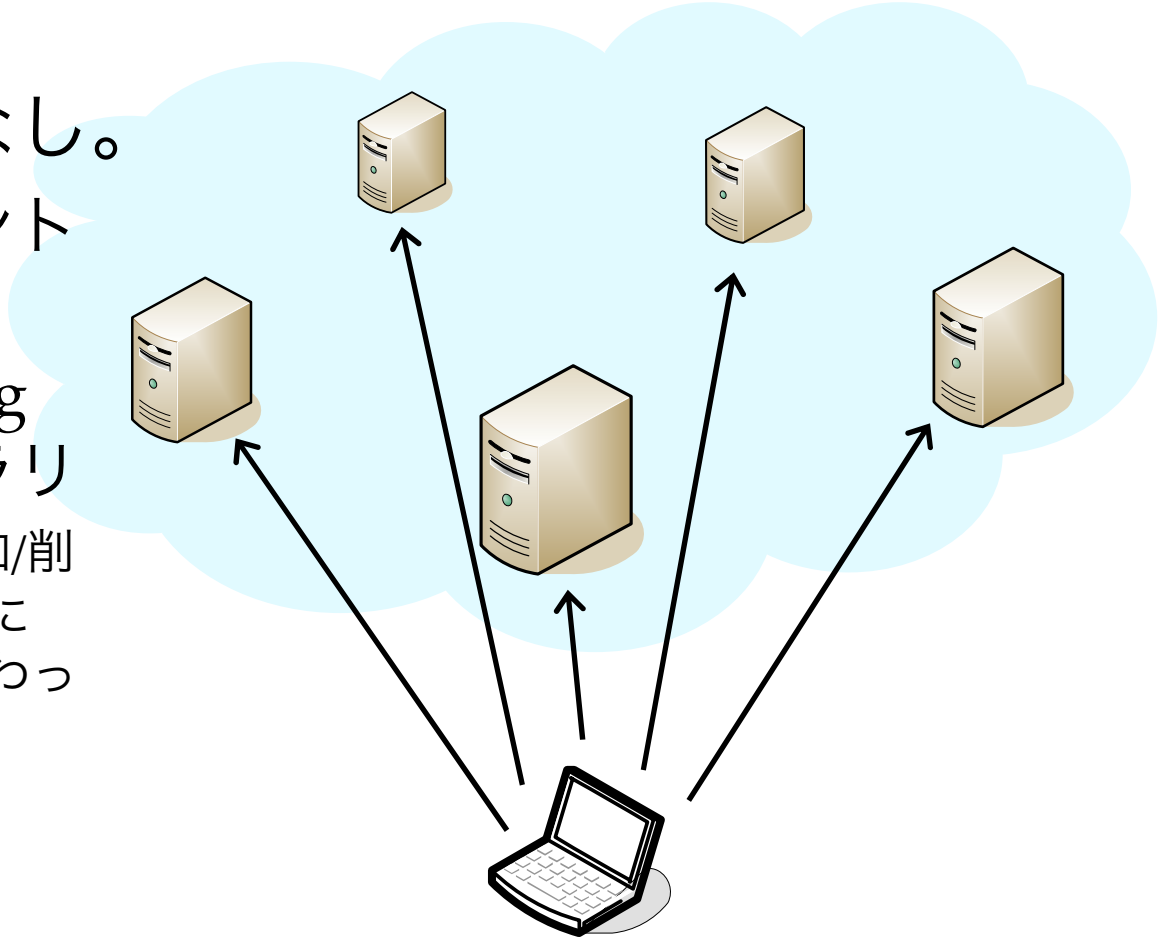
こうした現状を踏まえて、今後、**オーバレイ上のデータ処理をどうプログラミングするか**、を考えたい。

memcached

- Brad Fitzpatrick氏が、LiveJournal (blogのhosting) のために開発。
- 以後、ウェブ上サービスの裏側で広く使われている。
  - はてな, livedoor, mixi, Vox, Facebook, YouTube, Digg, Twitter, Wikipedia, …
- 規模の実績
  - mixi が 100台以上で運用。
- 特徴
  - テキストベースのシンプルなプロトコル。
    - set <key> <flags> <exptimes> <bytes> [noreply]¥r¥nデータ¥r¥n
    - get <key>*¥r¥n
    - 1.3 にはバイナリプロトコルも入る。少し速いらしい。
  - 性能重視の実装。数万 request/sec。
  - サーバ群は連携せず、担当ノードはクライアント (ライブラリ) が決める。

memcached

- 担当ノードはクライアントが決める。サーバ群は連携なし。
  - 方式はクライアントライブラリ次第。
  - consistent hashing ではないライブラリもある → ノード追加/削除によって、キー全体に渡って担当ノードが変わってしまう。



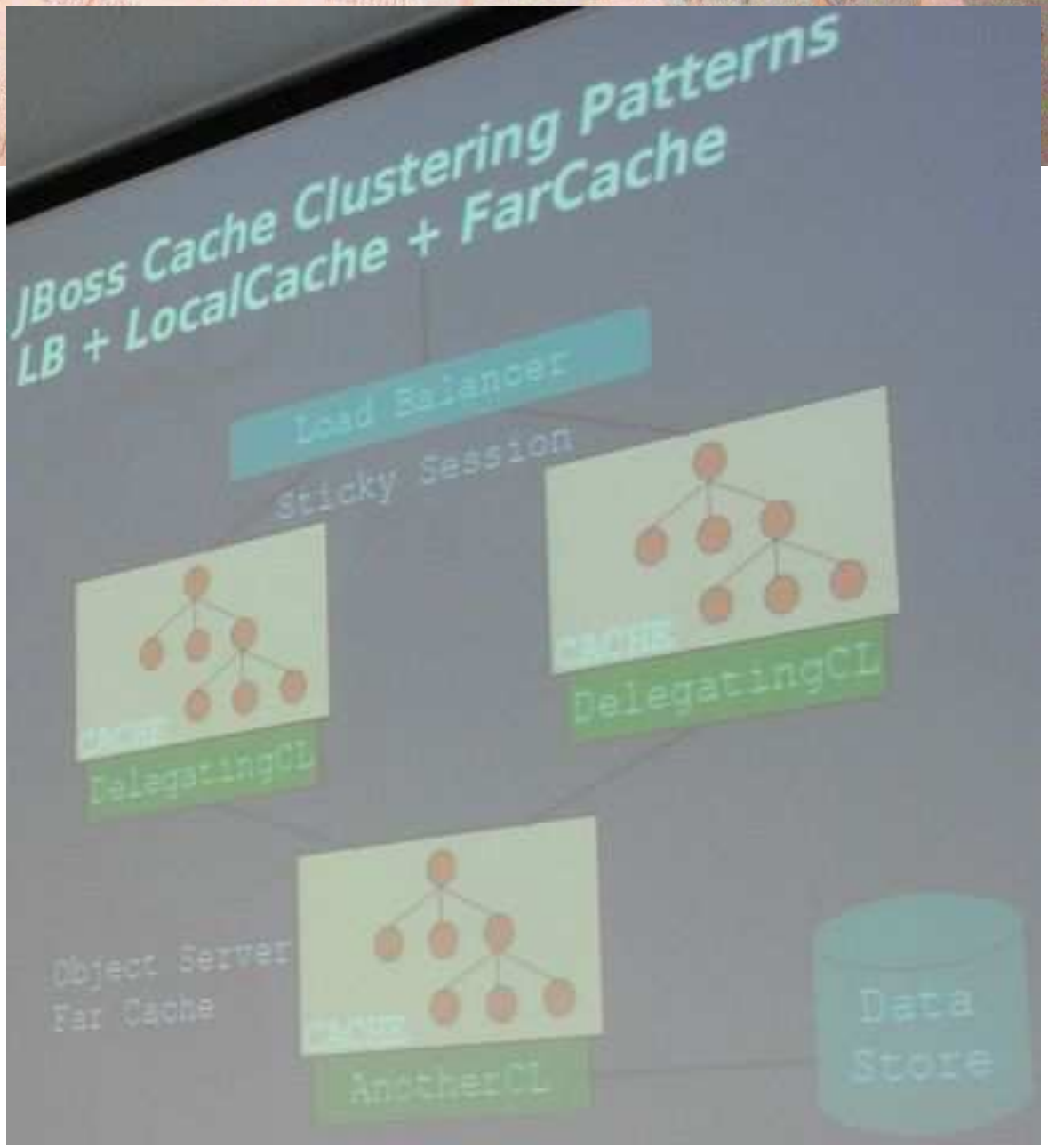
JBoss Cache

- JBoss: Java アプリケーションサーバ
- トランザクショナル キャッシュ エンジン
- 利用例
  - DB のデータキャッシュ
  - HTTP セッション レプリケーション
- 全ノードへの 複製
  - 同期 / 非同期 ← 性能と信頼性のトレードオフ
  - 1対全ではなく、同期を伝播させるトポロジを指定可。
    → スループット向上

JBoss Cache

2008年 2月、丸山先生レクチャーシリーズでの木村氏

JBoss Cache



3台での構成例

JBoss Cache

- トランザクショナル
  - 複数の要求が並行した場合、1つだけ成功する。
    - Pessimistic locking: ロックしてから更新
    - Optimistic locking
      - 作業コピーを更新して、書き戻す。
      - 書き戻し時にバージョン番号の変化を検地 → 失敗。
- API
  - ツリー状の名前空間、その各ノードに 1つずつハッシュ表がある。

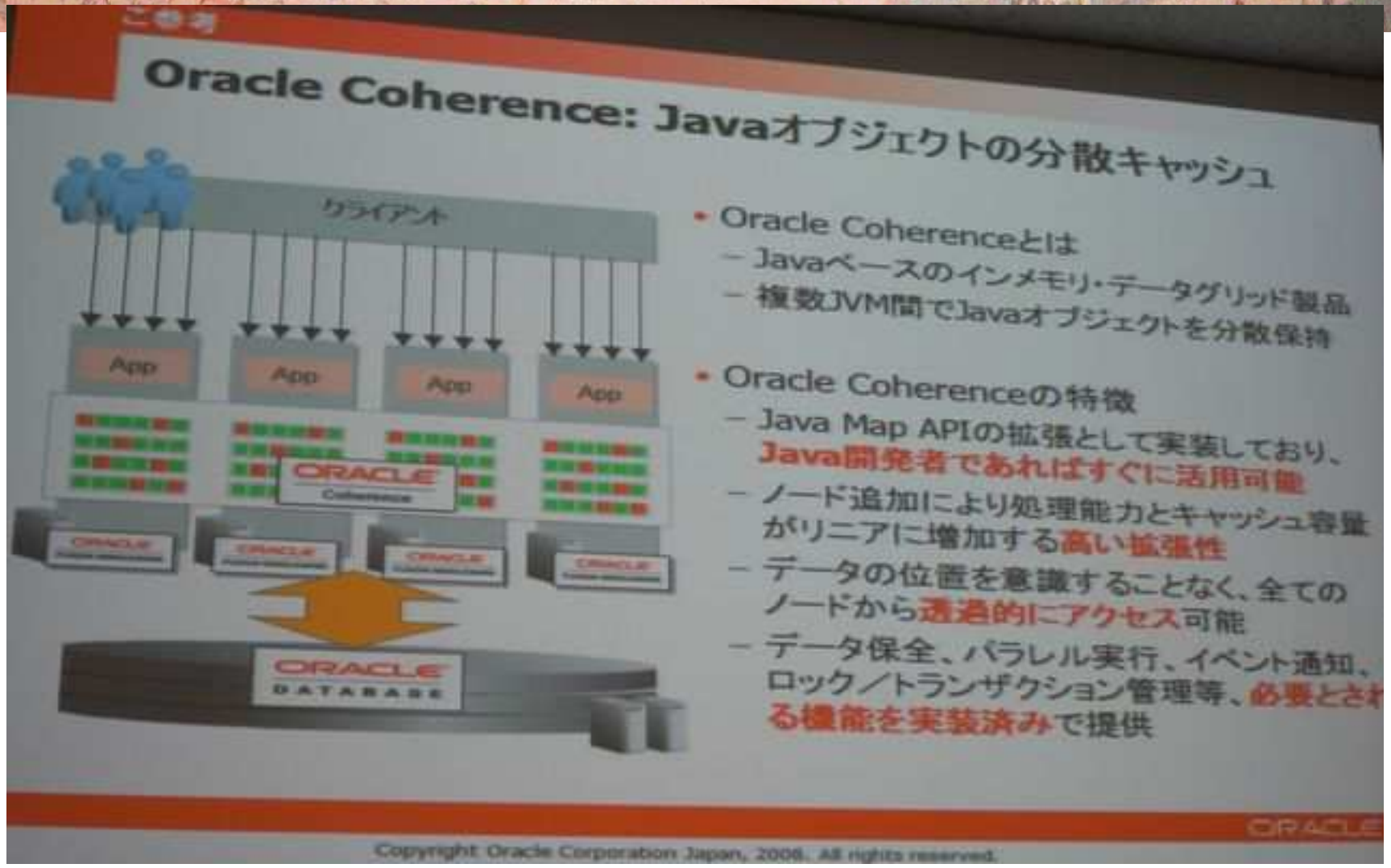
```
childFqn1 = Fqn.fromString(“/child1”);
child1 = root.addChild(childFqn1);
child1.put(“key1”, “value1”);
child1.get(“key1”);
child1.remove(“key1”);
```

Oracle社 Coherence

- 2007年 6月に Tangosol 社を買収して得た製品。
- RDB のフロントエンドとして配す。データをキャッシュする。
  - RDB が落ちても、後できちんと書き戻せる。

- 規模の実績
  - 1,000台程度。実地では6〜700台。by 杉達也氏 (日本オラクル)
- データの分散
  - Replicated cache
    - 全ノードが複製を持つ。
  - Partitioned (distributed) cache
    - クラスタ内で分散して持つ。
- API
  - NameCache nc = CacheFactory.getCache(“test”);
    nc.put(“key1”, “value1”);
    System.out.println(nc.get(“key1”));

Oracle社 Coherence



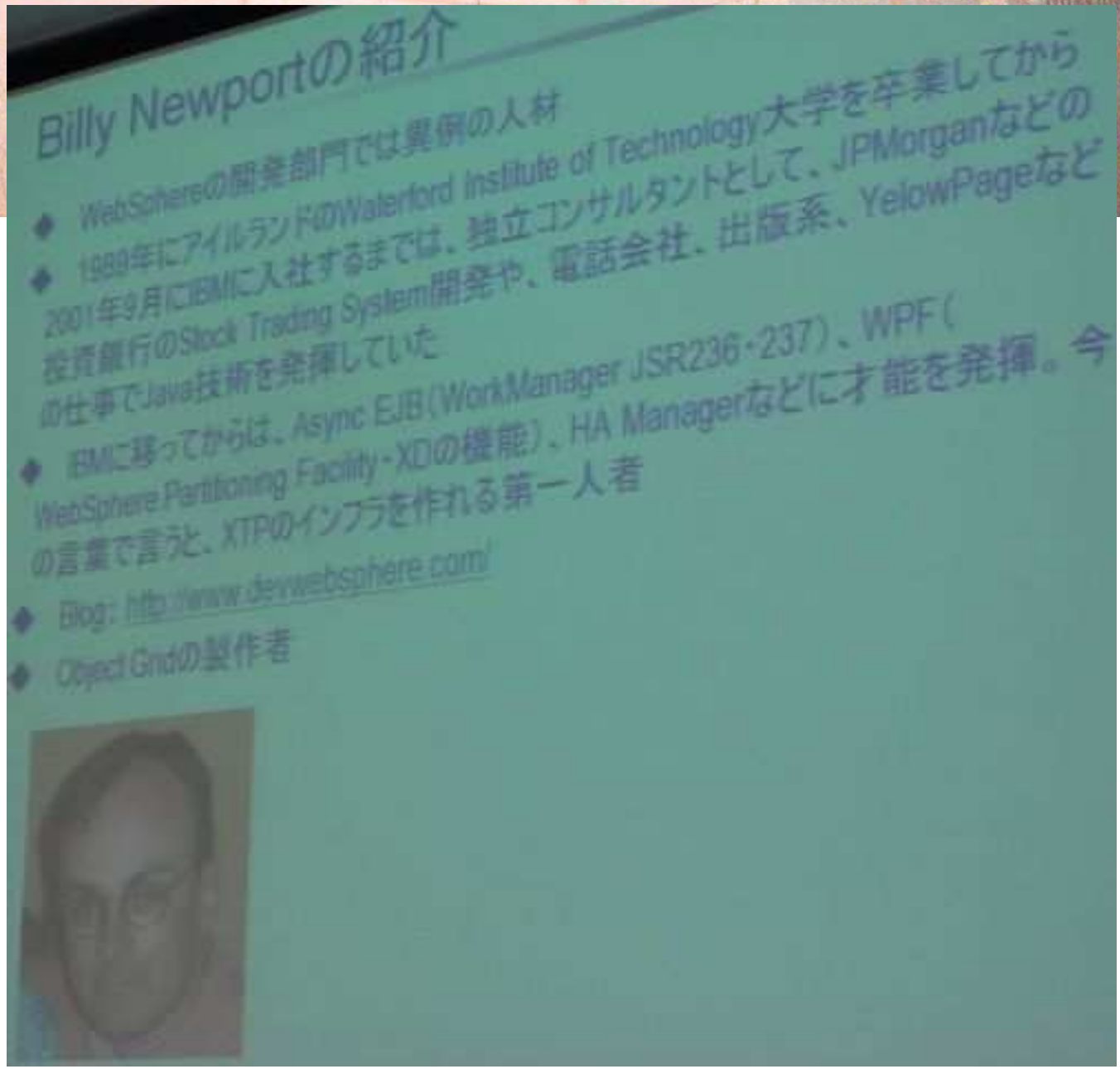
2008年 10月、クラウド研究会での杉氏のスライド

IBM社 ObjectGrid

- Java オブジェクトのキャッシュ。
- Billy Newport 氏が開発。

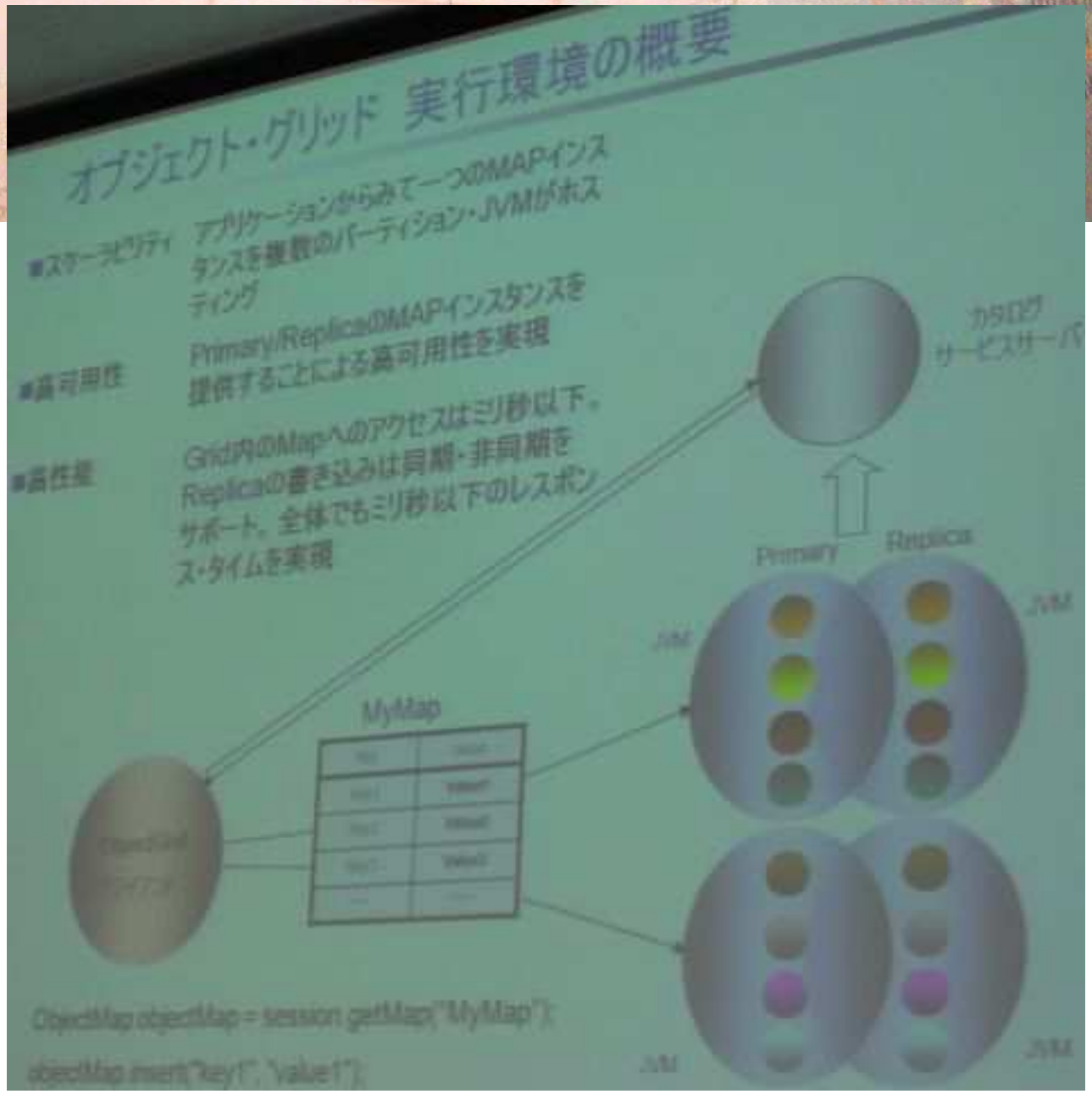
- 事例
  - New York の証券取引会社
    - ObjectGrid 4台: quad core AMD w/ 16 GB RAM, 64 bit RedHat Linux
    - 売買処理、FIX、送信完了までが 3.5 ミリ秒。
    - 1台あたり 2100 トランザクション/秒 (不確か)
- API – java.util.Map インタフェース
  - ObjectMap objMap = session.getMap(“MyMap”);
    objMap.insert(“key1”, “value1”);

IBM社 ObjectGrid



2008年 2月、丸山先生レクチャーシリーズでの清水氏のスライド

IBM社 ObjectGrid



2008年 2月、丸山先生レクチャーシリーズでの清水氏のスライド

Amazon's Dynamo

- 論文が出ている。SOSP'07。
  - 新奇なアイディアの提案ではなく、手堅い設計。

- Amazon の内部で使われている。
  - ショッピングカート。
  - Amazon Web Services (AWS) の SimpleDB は、おそらく Dynamo で運用されている。
    - SimpleDB
      - 2007年12月、β版公開。
      - データモデル: domain に item が入り、item は attribute-value ペアで表現される。スプレッドシートに例えると、domainはワークシート、attributeは列のヘッダ、valueはセルに入ったデータ。
      - API
        - » void PutAttribute(domain, item, 複数のattribute-valueペア)
        - » 複数の attribute-valueペア GetAttribute(domain, item)

amazon web services™

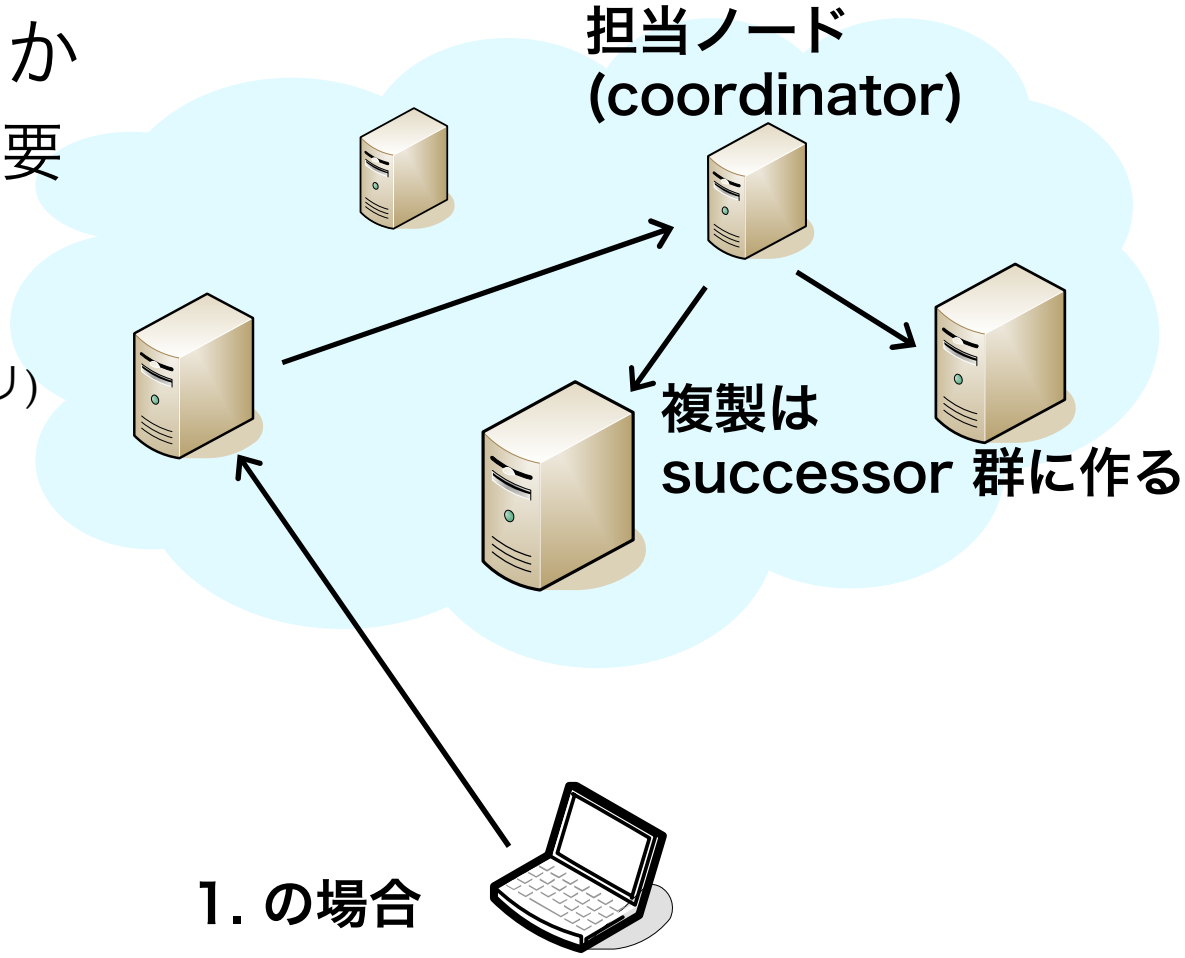
Amazon EC2
Amazon S3
Amazon SimpleDB
Amazon SQS

Amazon's Dynamo

- 狙う規模
  - 数百台
- 担当ノードの決め方
  - consistent hashing
- 一貫性
  - eventual consistency
  - 不整合が起きた場合、データに付いてるvector clocks [Lamport, 1978] を手がかりに、アプリ側で解決する。
- Service Level Agreements (SLA)
  - ウェブアプリのバックエンド → 遅延の低さ重視。
  - クライアントとの SLA に基づいて、パラメータ (e.g. 複製数) を調整する。
    - SLAの例: 500 req/s までなら要求の 99.9 % までは 300 msec 以内に返答を返す。

Amazon's Dynamo

- 担当ノードを決めるのは、次のどちらか
  1. クライアントから要求を受けたノード
  2. クライアント
     (要 クライアントライブラリ)



楽天 ROMA

- Rakuten On-Memory Architecture
  - 「複数マシンから構成されるオンメモリストレージ」
  - 「P2Pモデルを利用した分散ハッシュテーブルのruby実装」
    - これを「分散ハッシュ表」と呼ぶかどうかは、諸説あり？
- 2007年開発開始、2008年 9月社内向けα版。
- 狙い
  - 高い耐故障性 → 複製
  - それなりのパフォーマンス (耐故障性重視)
  - 高いスケーラビリティ
    - 狙いは数百台
    - 2008年 6月時点で「6台で 10ノードが動いてて…」by まつもとさん

楽天 ROMA

- 「ROMAの出発点をConsistent Hashingとする。」

ROMA への道（その２）

発想から基本設計へ

2007/09/01

楽天技術研究所　鳥居　順次

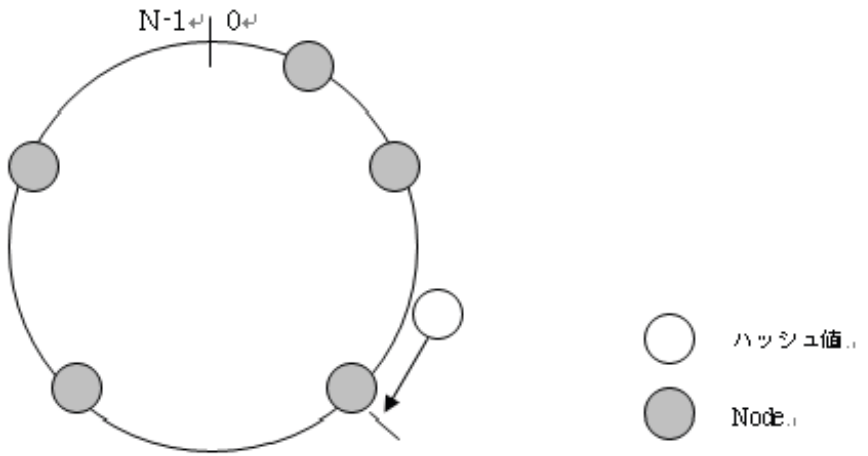
▪ これまでの道のり

ノードは環状に配置する方向で間違えはなさそう。

環状ハッシュは既に P2P の分野で大きな成功を収めている。広く知られている Chord 等の分散ハッシュのアルゴリズムを参考に ROMA を目指すゾ。

▪ ハッシュ関数とノード

ROMA の出発点を Consistent Hashing とする。ハッシュ関数は SHA-1 を想定し巨大な空間（最大 2^{160}）を定義する。ハッシュ値に対する担当範囲は時計回りに回ったときに到達する最初のノードとする。



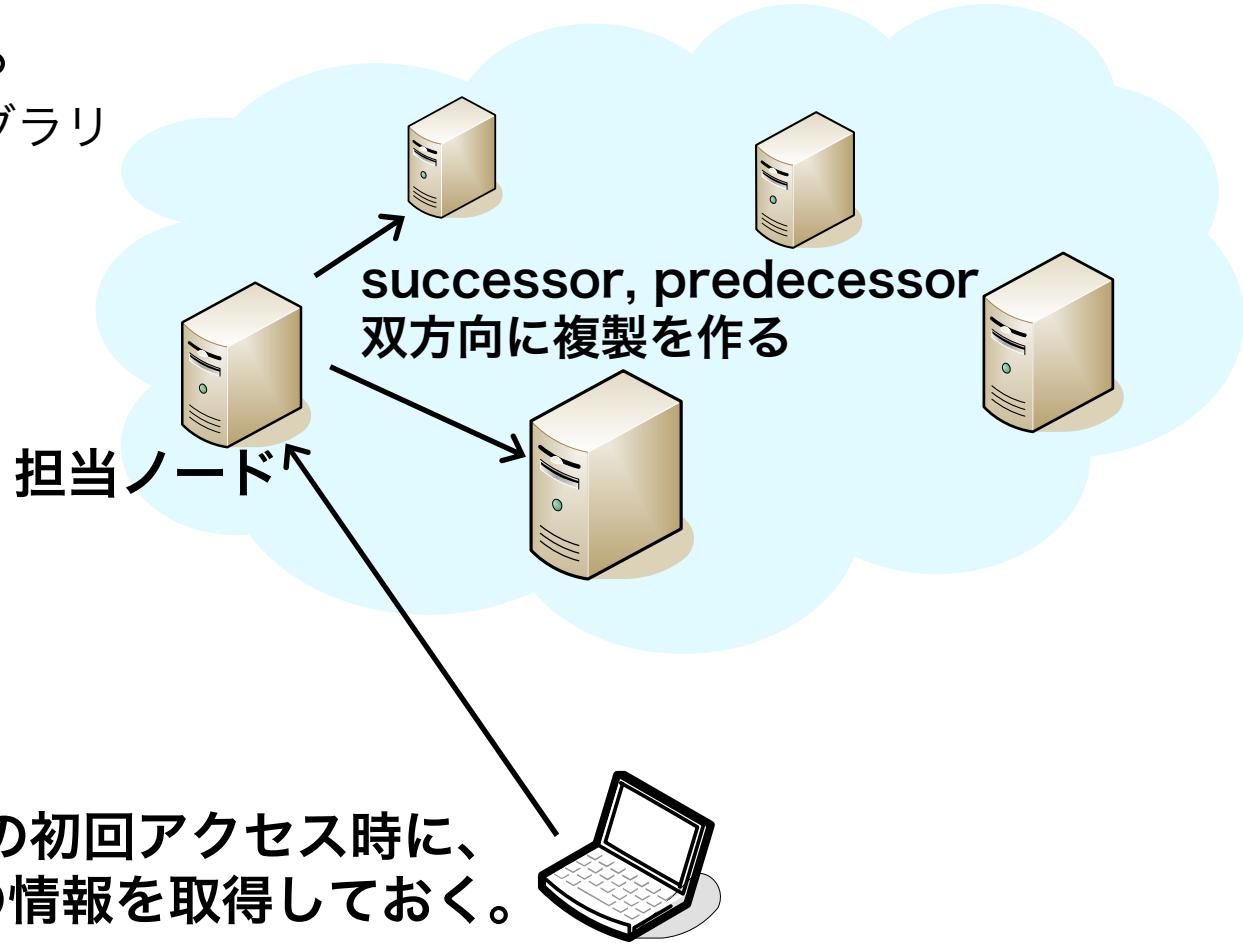
楽天技術研究所の紹介スライドより 2007年9月時点の設計メモ

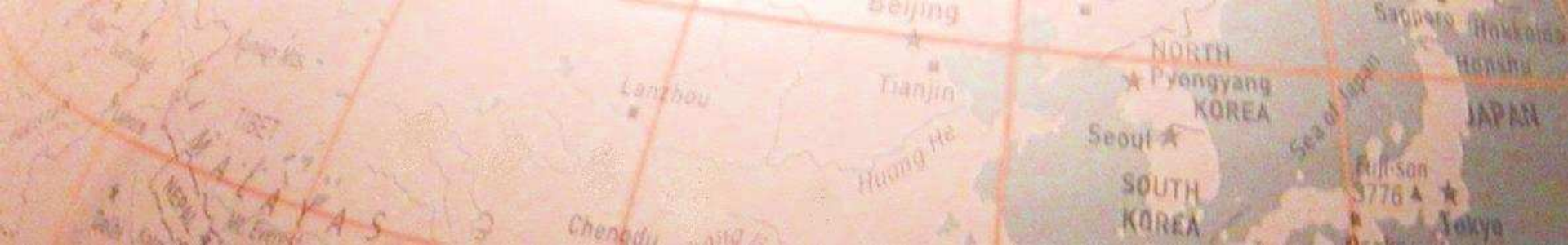
楽天 ROMA

2008年 10月、クラウド研究会での西澤氏

楽天 ROMA

- 担当ノードはクライアントが決める。
  - 要クライアントライブラリ





各システムの比較

スプレッドシート参照