



Couchbase | No EQUAL

COUCHBASE 概要とアーキテクチャ

河野 泰幸 | ソリューション・エンジニア

2021年 5月 12日



アジェンダ

- 1 Couchbaseサービスの理由
- 2 データのパーティション分割
- 3 K-V 操作
- 4 フェイルオーバーとリバランスのデモ



1

COUCHBASE サービスの理由



既存の技術を使用して
優れたユーザーエクスペリエンスを
実現することはできません

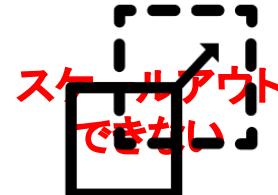


Couchbase NoSQL 技術はこれらの問題を解決する



静的スキーマに起因する
アジャイルと柔軟性の欠落

よりアジャイル・
市場投入までの時間を短縮



スケールアウト
できない

水平方向に弾力的に
スケールアウト



パフォーマンス
課題

スケールを問わない
パフォーマンスの向上



低いTCO

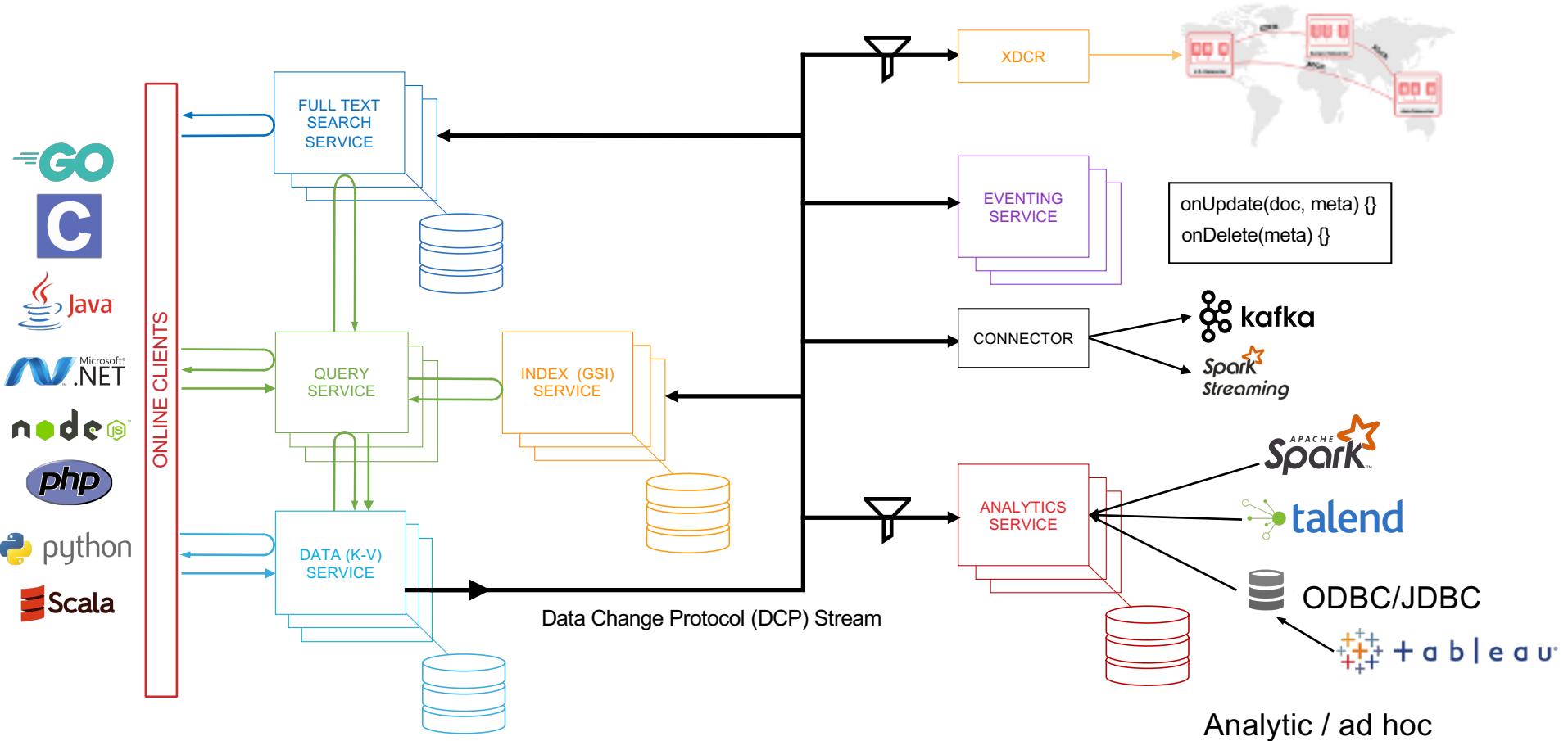


業界リーダーから選ばれたエンタープライズソリューション

Eコマース	旅行	ゲーム	デジタルヘルス	金融サービス	通信	デジタルメディア	産業用IoT
主要3社 (トップ10社中) Eコマース企業	トップ3社 GDS企業	主要6社 (トップ10社中) オンライン・ゲーム会社	Fortune 500 3社 ヘルスケア企業	トップ3社 クレジット企業	主要6社 (トップ10社中) 放送会社	トップ2社 IoT プラットフォーム	

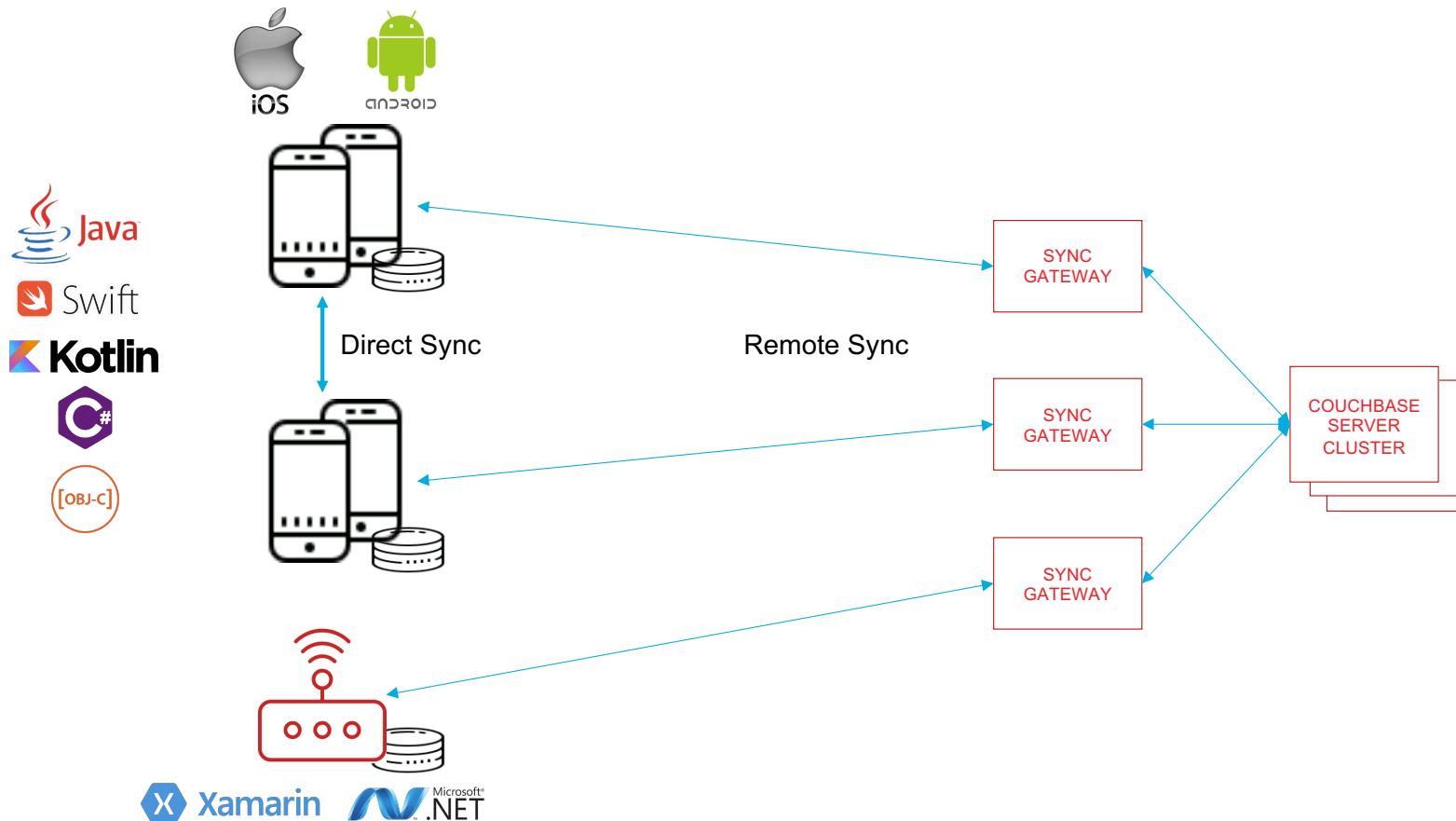


Couchbase サービスアーキテクチャ

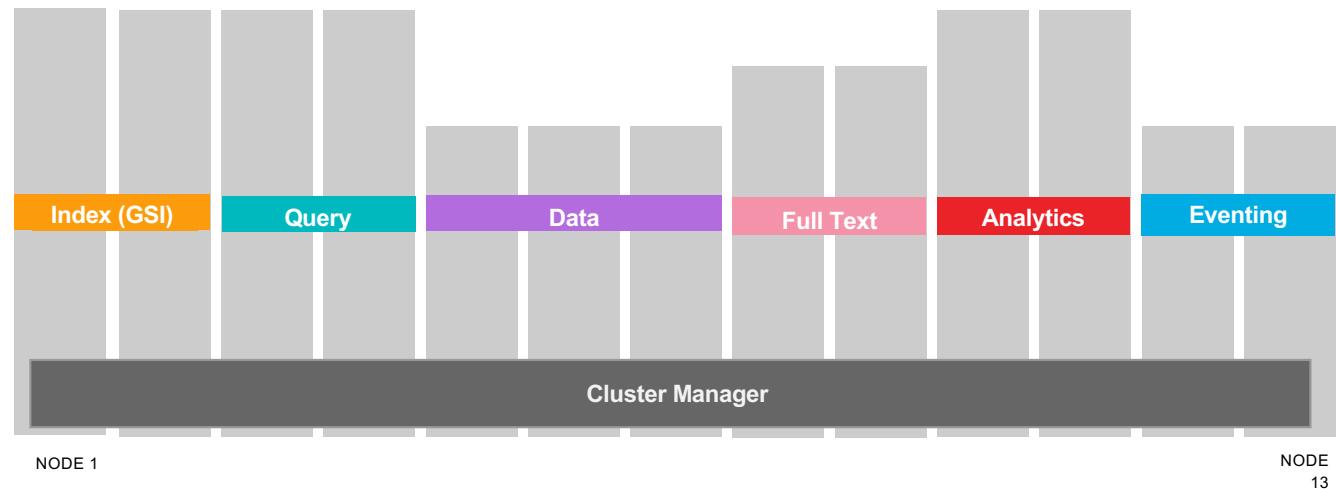




Couchbase Mobile

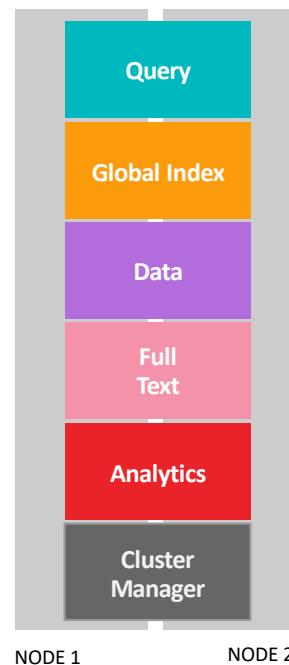


本番環境における 多次元スケーリング

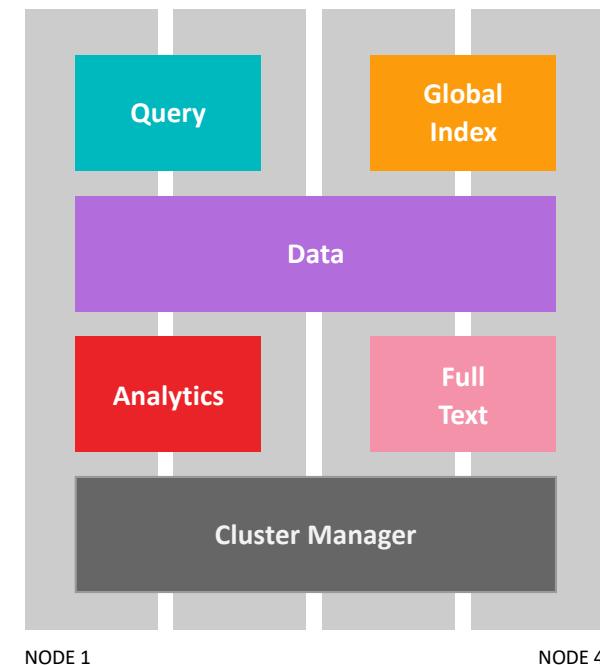


QAと開発における 多次元スケーリング

開発環境の例



QA環境の例

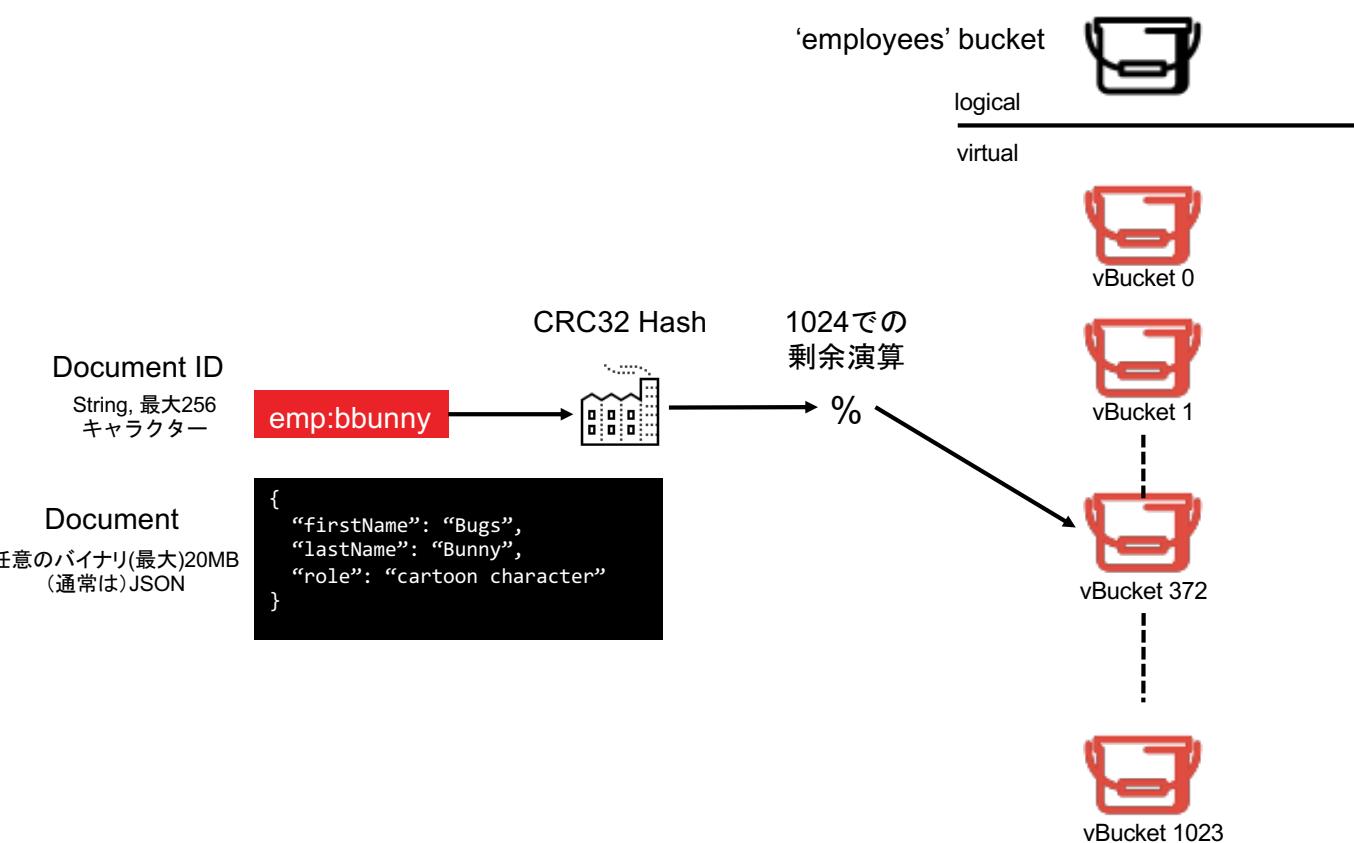


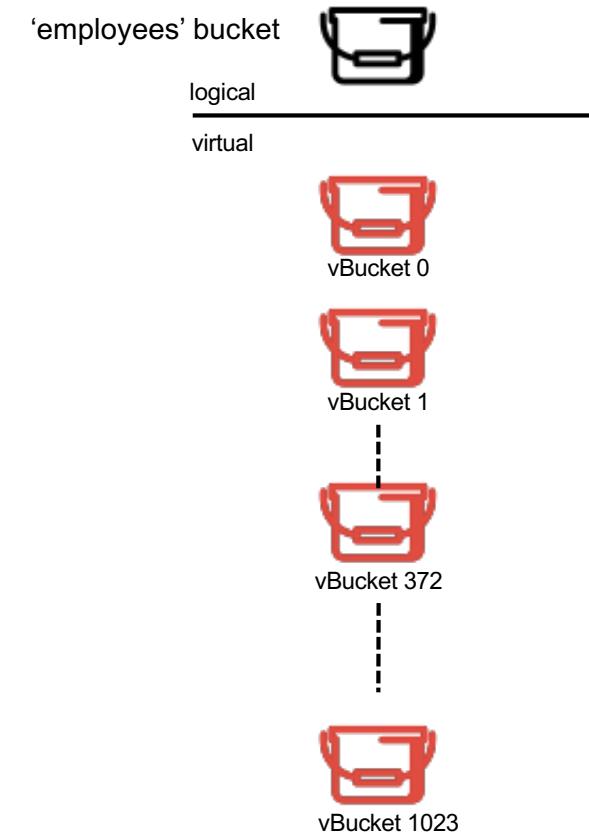


2

データのパーティション分割

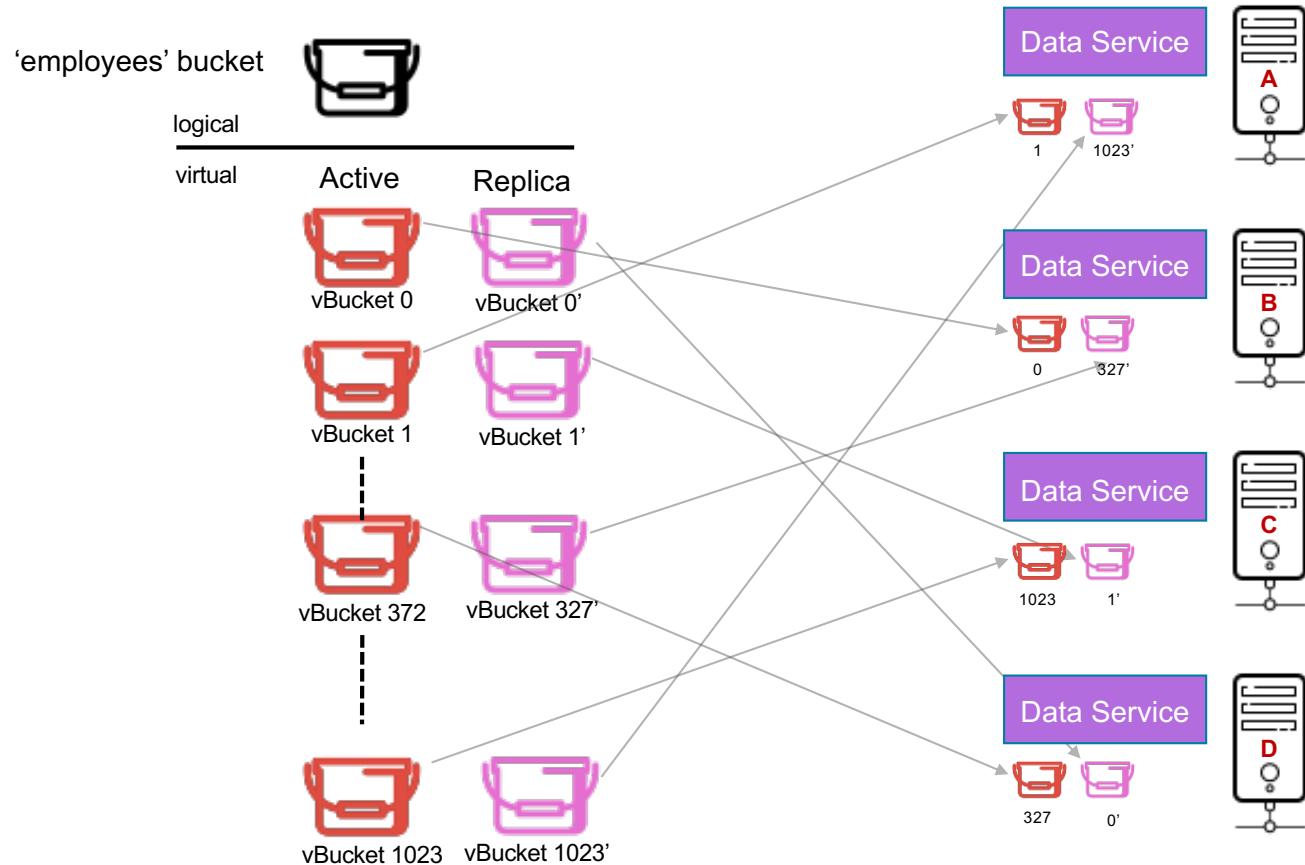
ハッシュを利用した均等なデータ配分





データ負荷分散 とレプリカ

レプリカのコピーは0、1、または2個の場合があります。ここでは簡潔にするために1つだけを示しています。



クライアント クラスターマップ



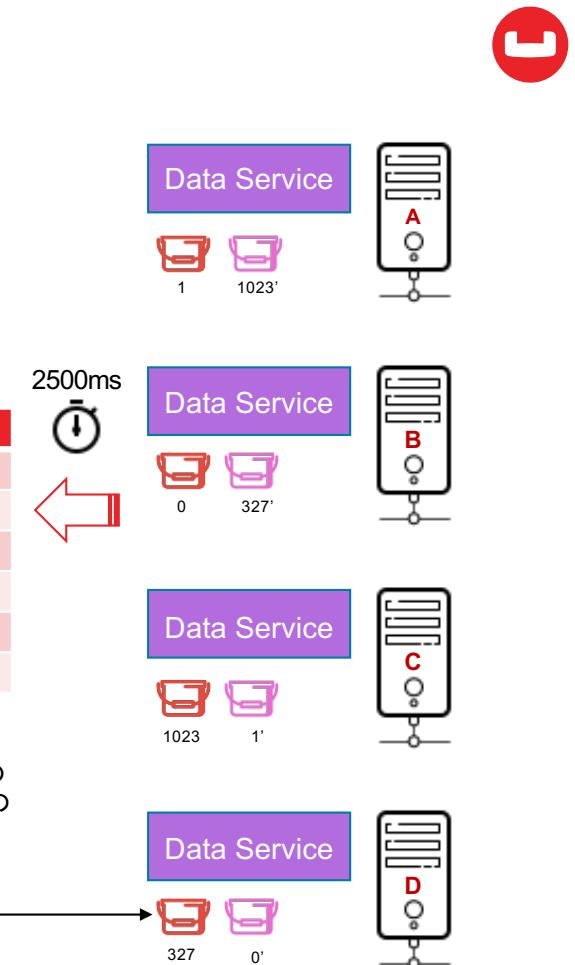
emp:bbunny

CRC32 %1024 = 327

クラスターマップ

vBucket	active	replica 1
0	B	D
1	A	C
...
327	D	B
...
1023	C	A

クラスタ マップには、Couchbase サービス (クエリ、FTS など) を実行しているマシンのホスト名や割り当てられたポートなどの他の情報も含まれています。



フェールオーバー vs リバランス



フェールオーバー (自動):

- すぐに起こってほしい～マシンが故障したか、または到達不能であることをシステムが検証するのにかかる時間に合わせて
- レプリカ vBucket をアクティブな vBuckets に昇格させる



リバランス (手動):

- 遅らせたい～可能であれば、障害が発生したノードを復元するまで。
- レプリカが2つある場合は、これを長く延期する余裕があります。
- リバランス前に複数のマシンが失われると、一部のデータが利用できなくなる可能性があります。
-



Kubernetes Autonomous Operator:

- Kubernetes環境でAOを使用して実行している場合、ポッドの交換、デルタリカバリ、リバランスを自動化できます。
- Couchbase Cloudは、水面下でAOを使用しています

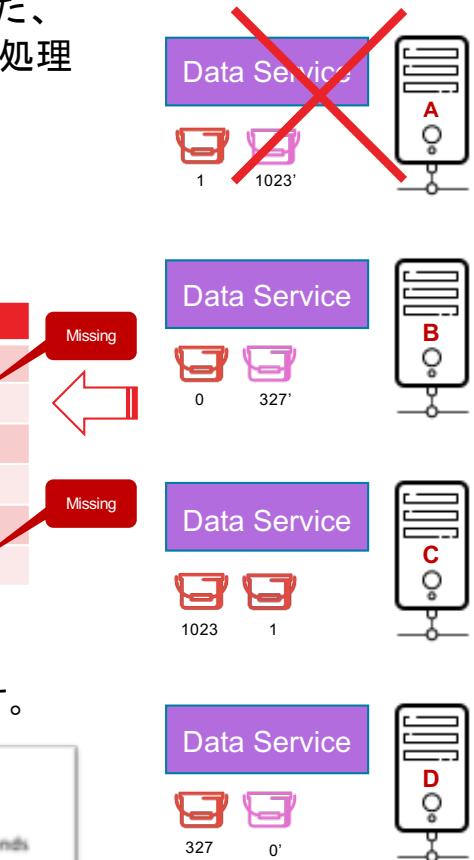
フェールオーバー

フェールオーバーはレプリカの vBucket を単にアクティブに昇格させます。マシン間で vBucket を移動することはありません。また、レプリカ・コピーを回復するための特別な処理は行いません。



vBucket	active	replica 1
0	B	D
1	C	
...
327	D	B
...
1023	C	

Promoted



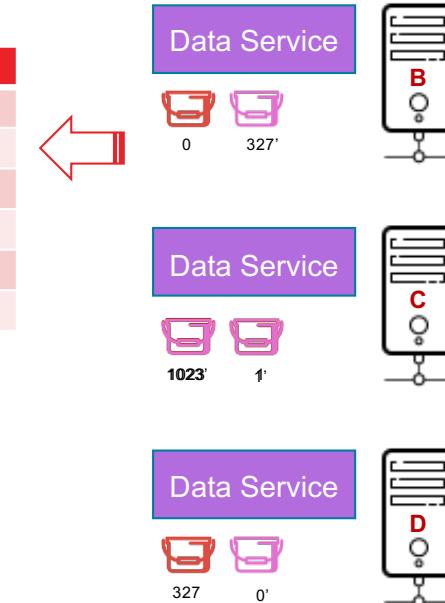
リバランス



リバランスは、データサービスノード全体でvBucket
完全なレプリカセットを再作成し、アクティブな
vBucketのセットを(可能な限り均等に)再配置します。



vBucket	active	replica 1
0	B	D
1	C	B
...
327	D	B
...
1023	C	D



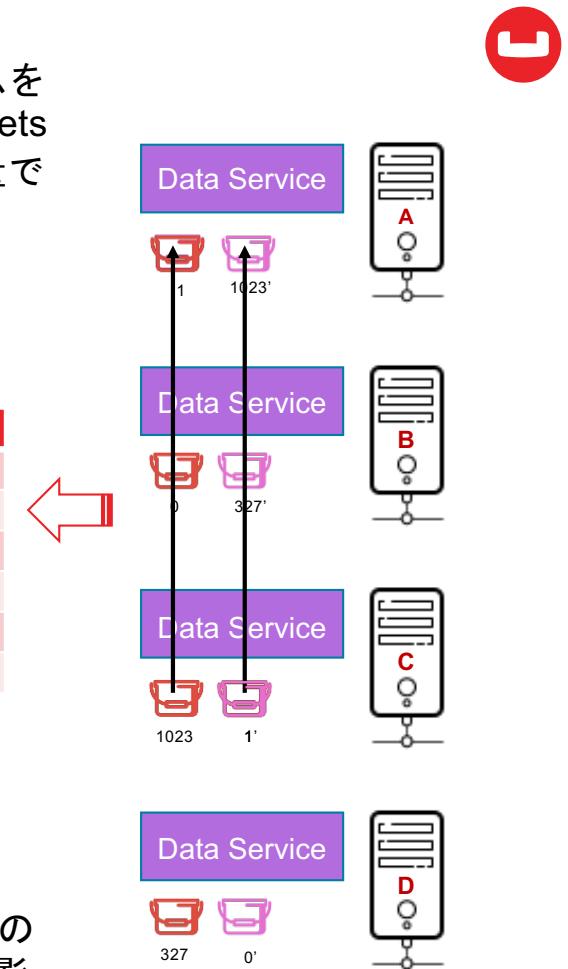
デルタリカバリ

(マシンを追加し直した後にリバランス)



障害が発生したノードからファイルシステムをリカバリして再アタッチできる場合、vBucketsは、より少ないネットワークトラフィック量で高速にリカバリできます。

vBucket	active	replica 1
0	B	D
1	A	C
...
327	D	B
...
1023	C	A

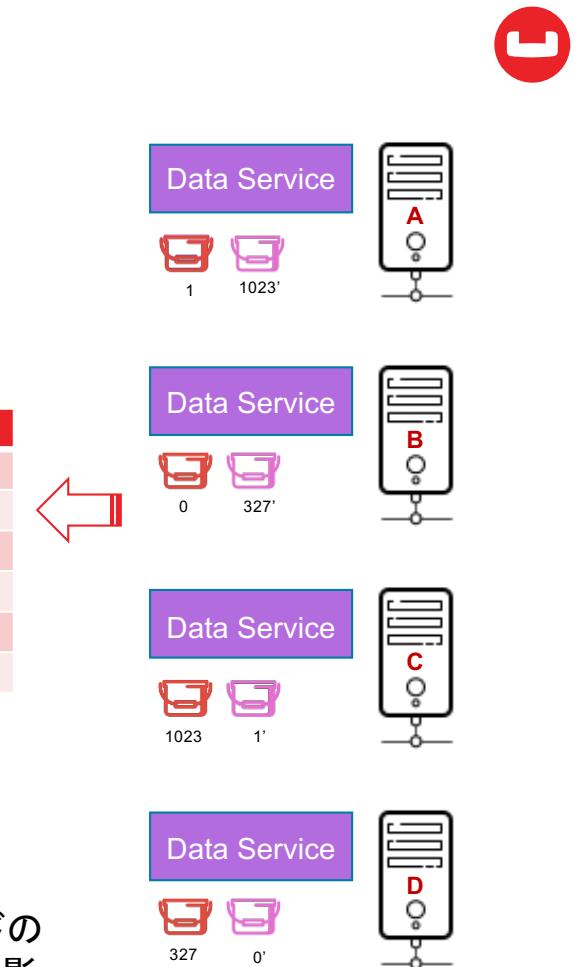


デルタリカバリでは、ローリングアップグレードの速度が大幅に向上升し、実行中のクラスタへの影響が少なくなります。



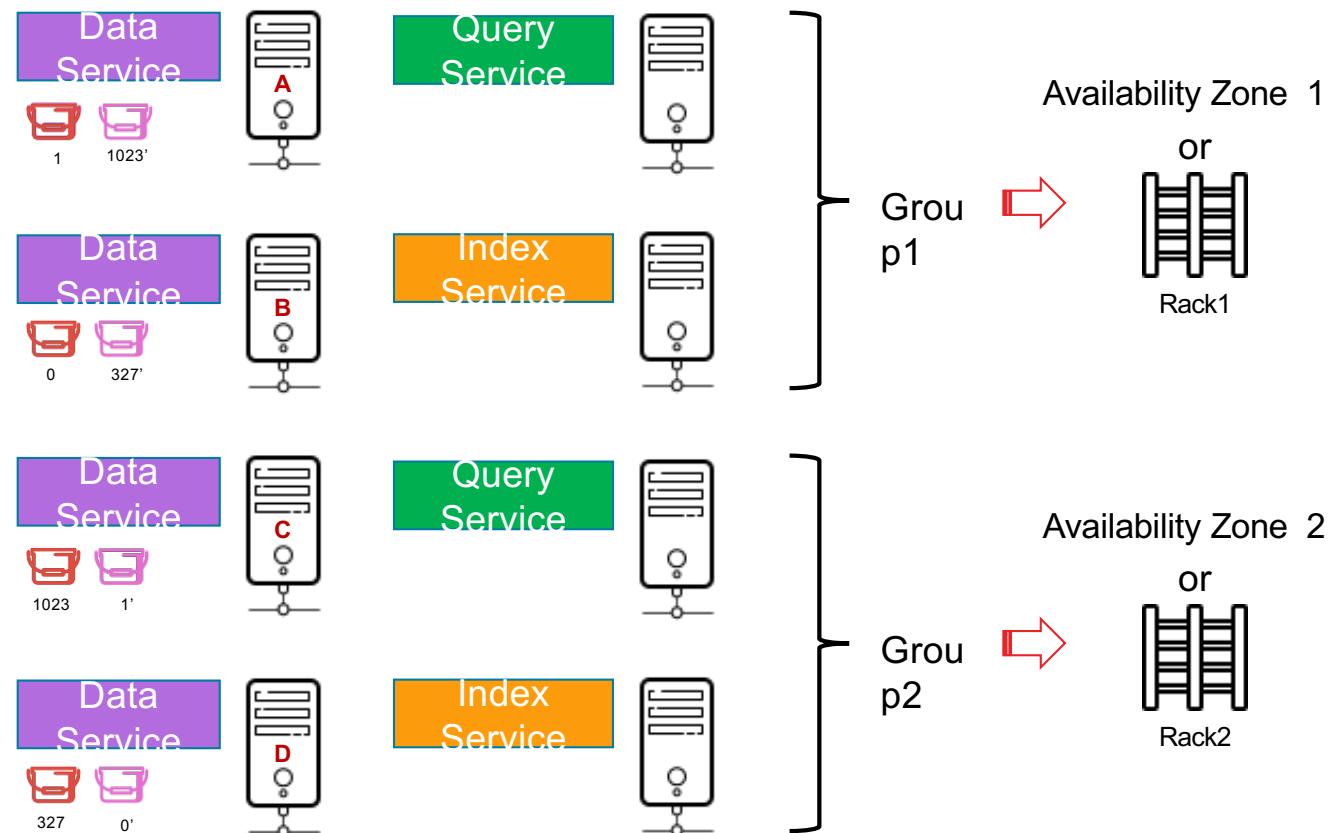
デルタリカバリでは、ローリングアップグレードの速度が大幅に向上し、実行中のクラスタへの影響が少なくなります。

vBucket	active	replica 1
0	B	D
1	A	C
...
327	D	B
...
1023	C	A



Groups

サーバー グループは、ラックまたはクラウドの可用性ゾーンにマッピングできます。



リバランスする前に、各グループ内に同数のアクティブ・データ・ノードがあることを確認してください。

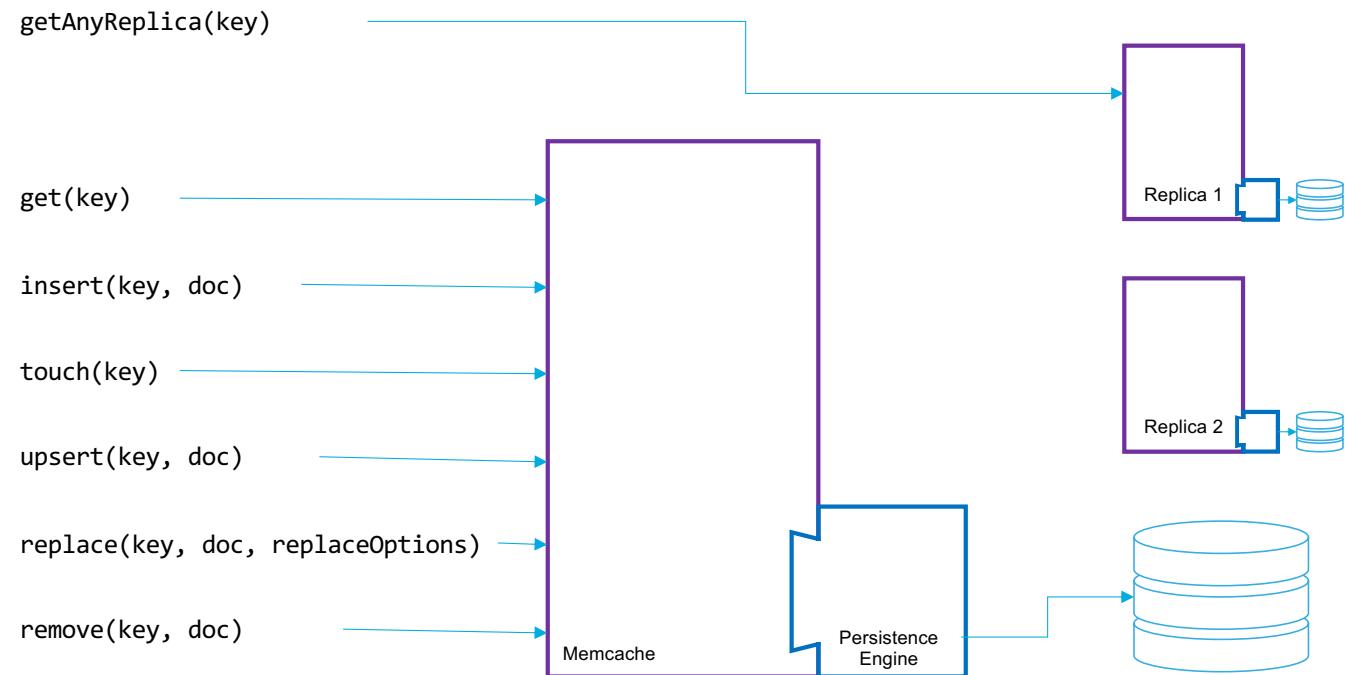


3 | K-V 操作

統合キャッシュ



キー・バリュー操作はすべて、ドキュメントキーのハッシュに対応するアクティブな vBucket に対してメモリ内で実行されます。これに対する1つの例外は、getAnyReplica(key)です。これは非常に特殊な状況でのみ使用する必要があります。NQ1LとFTSもデータを直接メモリに対して、読み取り・更新します。



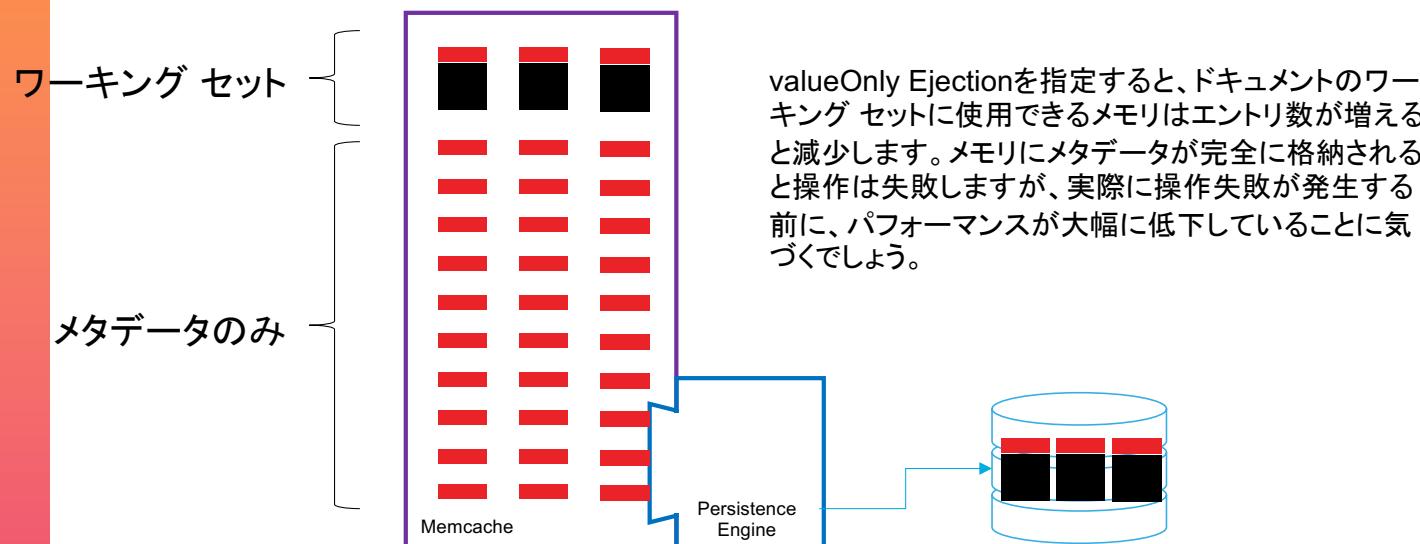


キャッシュ削除

メモリがいっぱいになると、次の 2 つのEjection(キャッシュ削除)オプションがあります。

- **valueOnly** キーを残して全てのバリューを削除する
 - **fullEviction** 最近アクセスされたキーとバリューのみをメモリに保持
- キーとバリューの両方がディスクに格納され、必要に応じて取得されます

レプリカvBucketsもメモリスペースが必要であることにご注意ください

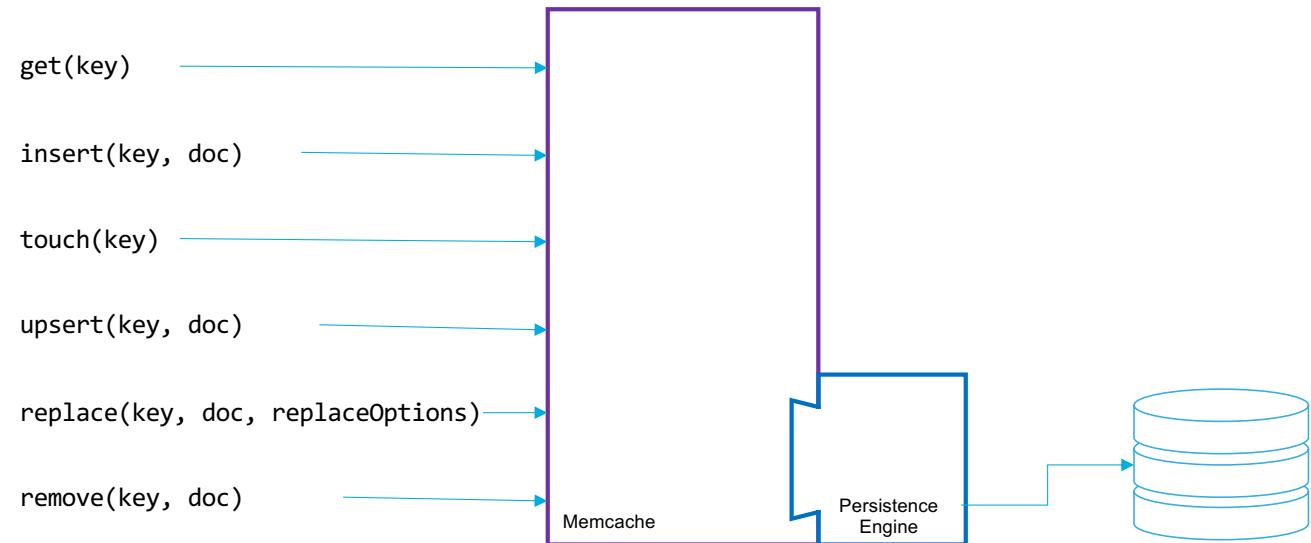




Full Cache Eviction (パフォーマンスへの影響)

すべての操作は「完全排出」と共に動作しますが、キーがメモリに保持されない場合、一部の操作は、「完全排出」前と同じ、パフォーマンスがないことに注意してください。

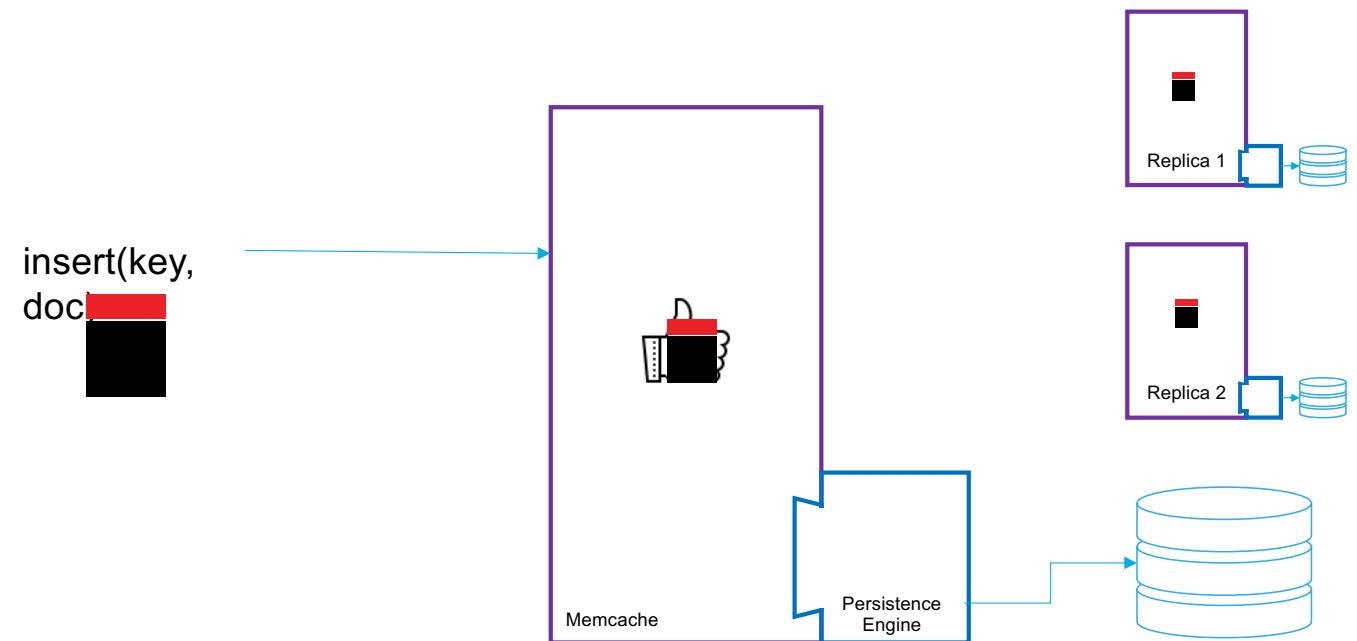
`upsert` は、「完全排出」の影響を受けない(エントリの存在を最初に確認する必要がないため)
`insert, touch, replace, remove` は、実行前にキーが存在するかどうかを知る必要がある
`get` は、値がメモリ内にある場合は高速になりますが、そうでない場合は、ファイル I/O が必要で



K-V 結果整合性 と耐久性 (デフォルト)



デフォルトでは、すべてのレプリケーションおよび永続化操作は非同期であり、結果整合性に基づいています。



K-V 同期レプリケーション

クライアントは、リクエストベースで、より厳密な一貫性および/または耐久性を要求することができます。他のクライアントには、整合性要件が満たされるまで変更が反映されません。

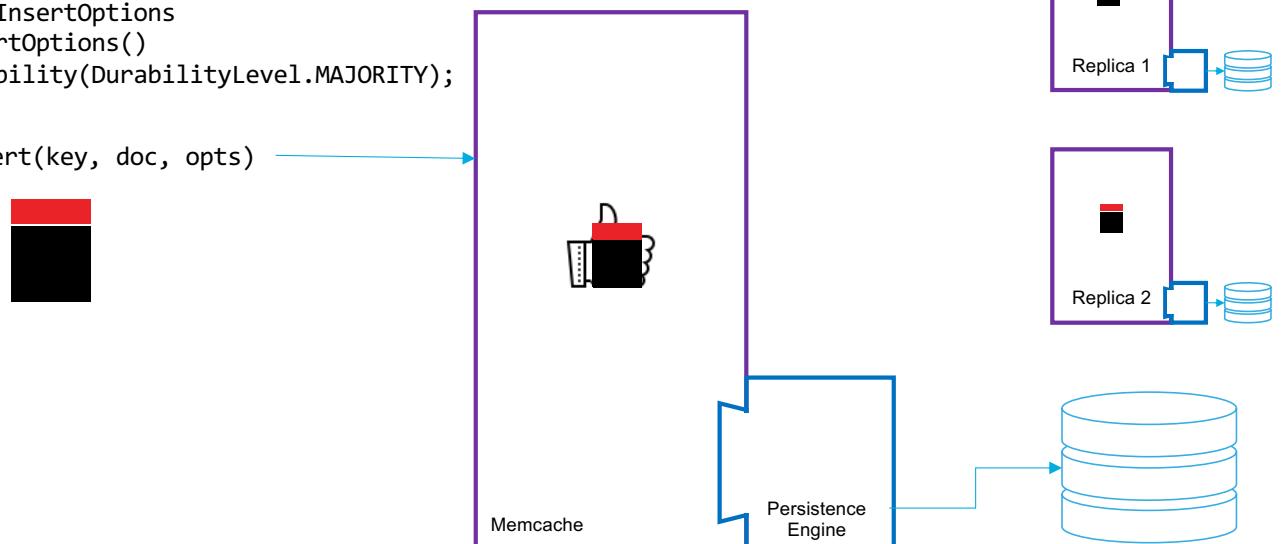


耐久性レベル:

- **None** – デフォルトの結果整合性と耐久性モデル
- **Majority** – 値を戻す前に、データコピーのマジョリティで、メモリを更新します
- **MajorityAndPersistToActive** – 値を戻す前に、アクティブなvBucket上のディスクとデータコピーのマジョリティで、メモリを更新します
- **PersistToMajority** – 値を戻す前に、データコピーのマジョリティで、メモリとディスクを更新します
-

```
opts = InsertOptions  
    .insertOptions()  
    .durability(DurabilityLevel.MAJORITY);
```

```
insert(key, doc, opts)
```



ACID トランザクション



K-V 同期レプリケーションは、Couchbase で分散された複数ドキュメント ACID トランザクションをサポートする基盤となるメカニズムです。

A	Atomicity	異なるノード上の複数のシャードで複数のドキュメントを更新するための、オール・オア・ナッシングの更新を保証します。
C	Consistency	レプリカは、トランザクションコミットと即座に一致します(Immediately Consistent)。 インデックスと XDCR は、トランザクション コミットと最終的に一致(結果整合)します。 (N1QL は、request_plus オプションを用いて読み取り時に一貫性を強制できます)
I	Isolation	同時実行トランザクションのためのリード・コミッテド アイソレーション
D	Durability	障害発生時のデータ保護: 3つの異なるレベル <i>replicate to majority of the nodes</i> <i>replicate to majority and persist to disk on primary</i> <i>persist to disk on majority of the nodes</i>

メモ: ドキュメントのコミットされていないバージョンは、k-v メタデータの xattrs セクションに格納されます。これらのコミットされていないバージョンのデータは、KV値の最大サイズ 20 MB の制限に含まれます。

ミューテーション トークン



クライアントで有効にすると、すべての CRUD 操作でミューテーショントークンを返すことができます。

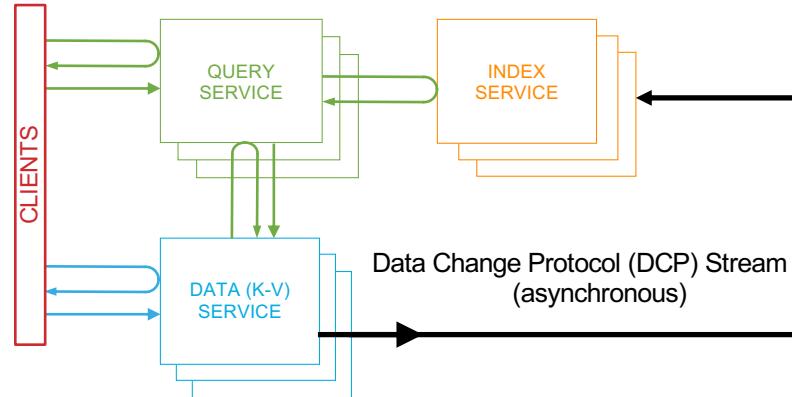
```
CouchbaseEnvironment env = DefaultCouchbaseEnvironment .buil
    .mutationTokensEnabled(true)
    .build();
```

k-v 操作から返されるドキュメントには、ミューテーショントークンを含む追加情報が含まれています。

```
result1 = collection.upsert(key1, doc1);
result2 = collection.upsert(key2, doc2);
mutationState = MutationState.from(result1, result2);
```

ここで得られたMutationStateは、クライアントのデータ整合性のニーズを満たすためにクエリを実行するときに使用することができます

自分の書き込み を読む (Read your own writes)



インデックスは非同期的に更新されるため、インデックスは、データから少し時代遅れの（遅れている）可能性があります。自分の書き込んだデータを直ちにクエリし、最新の変更がクエリ結果に含まれることを期待する必要がある場合。

```
queryOptions().scanConsistency(QueryScanConsistency.  
REQUEST_PLUS)
```

この操作により、クエリの実行は、インデックスの更新を待って行われます。

```
queryOptions().consistentWith(mutationState)
```

また、この操作により、インデックスにmutationStateに含まれる全ての変更が反映されるまでクエリが遅延します。

このオプションは、REQUEST_PLUSよりも高速になります。



4

フェイルオーバーと リバランスのデモ



質疑応答

利用可能なその他の
リソース

次のセクション：
モジュール 3
データ モデリング



Couchbase Download:

https://www.couchbase.com/downloads?utm_source=field_event&utm_medium=workshop&utm_campaign=developerworkshop

For More Information on:

- Couchbase Cloud: <https://www.couchbase.com/products/cloud>
- Couchbase Academy: <https://couchbase.com/academy>
 - Certification: <https://couchbase.com/academy/certification>
- Couchbase Professional Services: <https://www.couchbase.com/professional-services>

Our Website: <http://Couchbase.com>

