分散システム

第7回 一貫性と複製

双見 京介（FUTAMI Kyosuke）
高田秀志（TAKADA Hideyuki）

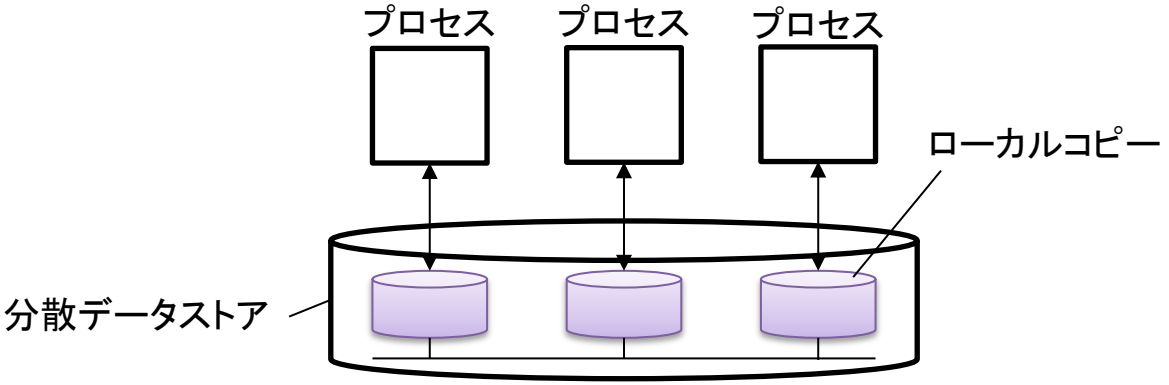
2025年11月

はじめに

- 「データの複製」を行う理由
  - **信頼性**: ファイルシステムが複製されていると，1つのレプリカがクラッシュした後も，他のレプリカに切り替えるだけで動作を継続可能
  - **性能**: データを使用するプロセスの近くにデータのコピーを置くことにより，データへアクセスする時間を削減可能
- データが複製されるとき，払うべき代償が存在
  - あるコピーが変更されると，そのコピーは他のコピーとは異なる状態になる
  - 変更はすべてのコピーで行われなければならない（「一貫性」の保持）

データ中心一貫性モデル

- 1つのデータストアが複数のマシンにまたがって物理的に分散され得る
- データストアのデータにアクセスするプロセスは，それぞれローカルなコピーを保有していることを想定
- 書き込み操作は他のコピーに伝搬される



- 一貫性モデルはプロセスとデータストアの間での契約であり，プロセスがあるルールに従うことに同意するなら，データストアは正しく動作することを約束する

操作への一貫性のある順序付け

- それぞれのマシンが自身のクロックを持ち，グローバルなクロックの存在を仮定できない場合，どの書き込み操作が一番最後に行われたのかを正確に定義することは困難
- 一貫性を緩和した他のモデルを与える
  - 順序一貫性（Sequential consistency）
  - 因果一貫性（Causal consistency）
- 以降，プロセスの操作を以下に示す記法で示す

P1: W(x)a

P2: R(x)NIL R(x)a

- P1 がデータ項目 x に書き込みを行い，値を a に変更
- この操作は，まず P1 のローカルコピーに対して行われ，他のコピーに伝播される
- その後，P2 は値 NIL（空値）を読み，しばらく後に値 a を読む

順序一貫性

どの実行結果も，すべてのプロセスによるデータストアへの操作（読み込み・書き込み）が，ある逐次的な順序によって実行された結果と同じであり，また，個々のプロセスの操作は，そのプログラムによって指定された順序の並びで現れる

P1: W(x)a				
P2:	W(x)b			
P3:		R(x)b		R(x)a
P4:			R(x)b	R(x)a

(a) 順序一貫性であるデータストア

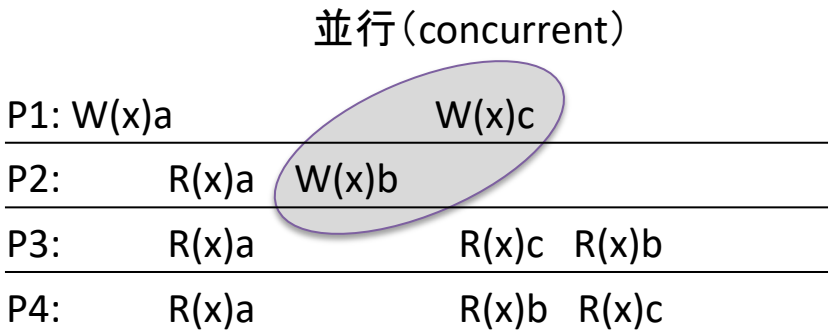
P1: W(x)a				
P2:	W(x)b			
P3:		R(x)b		R(x)a
P4:			R(x)a	R(x)b

(b) 順序一貫性でないデータストア

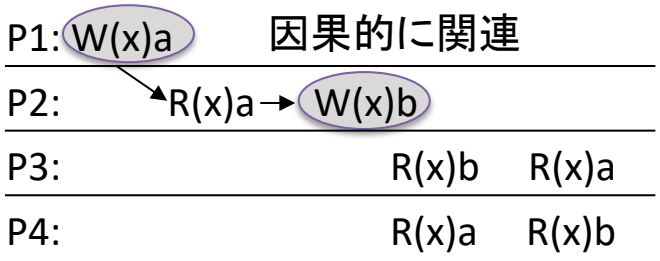
読み込み操作と書き込み操作はどのように綴じ込まれても良いが，すべてのプロセスが同じ綴じ込みを観測する
（P1とP2のどちらが先に書き込んだのかは重要ではなく，P3とP4が同じ順序で値を読むことが重要）

因果一貫性

因果的に関連している可能性のある書き込みは，すべてのプロセスにおいて，同じ順序で観測されなければならない。並行書き込みは，異なるプロセス上で異なる順序で観測されてもよい。



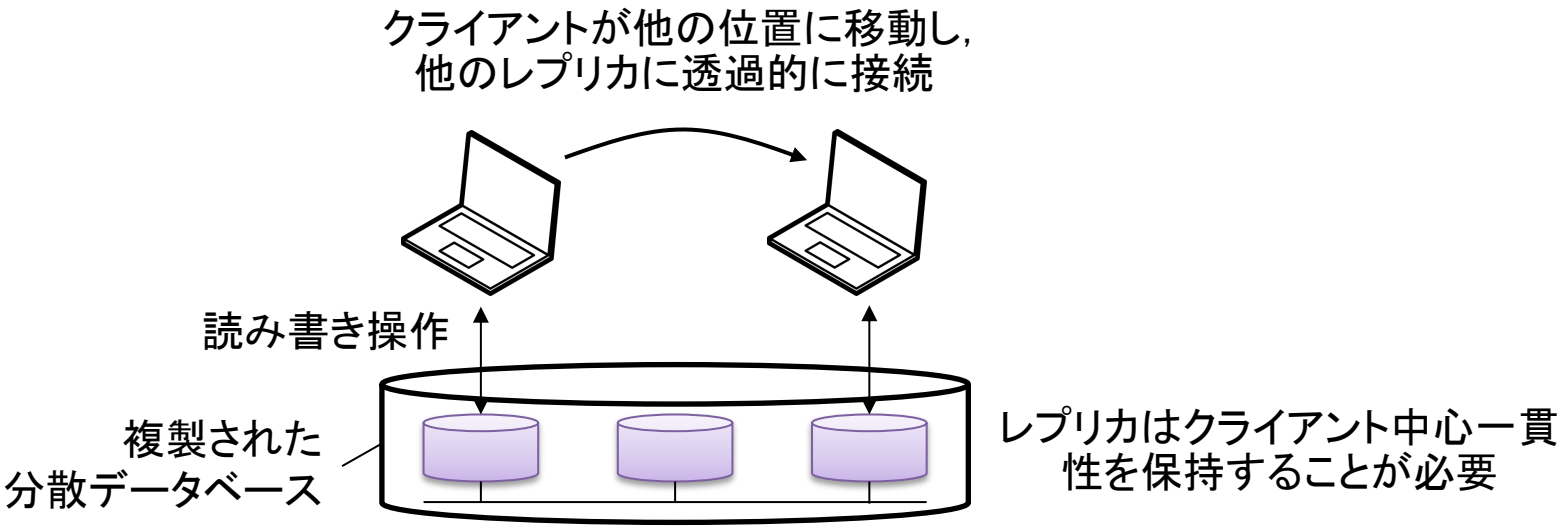
このイベントの並びは因果一貫性のデータストアでは許容されるが，順序一貫性のデータストアでは許容されない



(a) 因果一貫性の違反例

(b) 因果一貫性における適正なイベントの並び

クライアント中心一貫性モデル



- 単一のクライアントに対して，そのクライアントによるデータストアへのアクセスに関する一貫性を保証
- 異なるクライアントによる並行アクセスに関する保証は与えられない（書き書き競合が容易に起こる）

クライアント中心一貫性モデルのクラス

- モノトニック読み取り（Monotonic reads）

  あるプロセスがデータ項目 x を読み取るならば，そのプロセスによる x に対する後続の読み取りでは，同じ値か，より新しい値を返答する

- モノトニック書き込み（Monotonic writes）

  あるプロセスによるデータ項目 x への書き込みは，同じプロセスによる x への後続の書き込みよりも前に完了する

- 書き込み後読み取り（Read your writes）

  あるプロセスによるデータ項目 x への書き込みは，同じプロセスによる x への後続の読み込み操作によって常に観測される

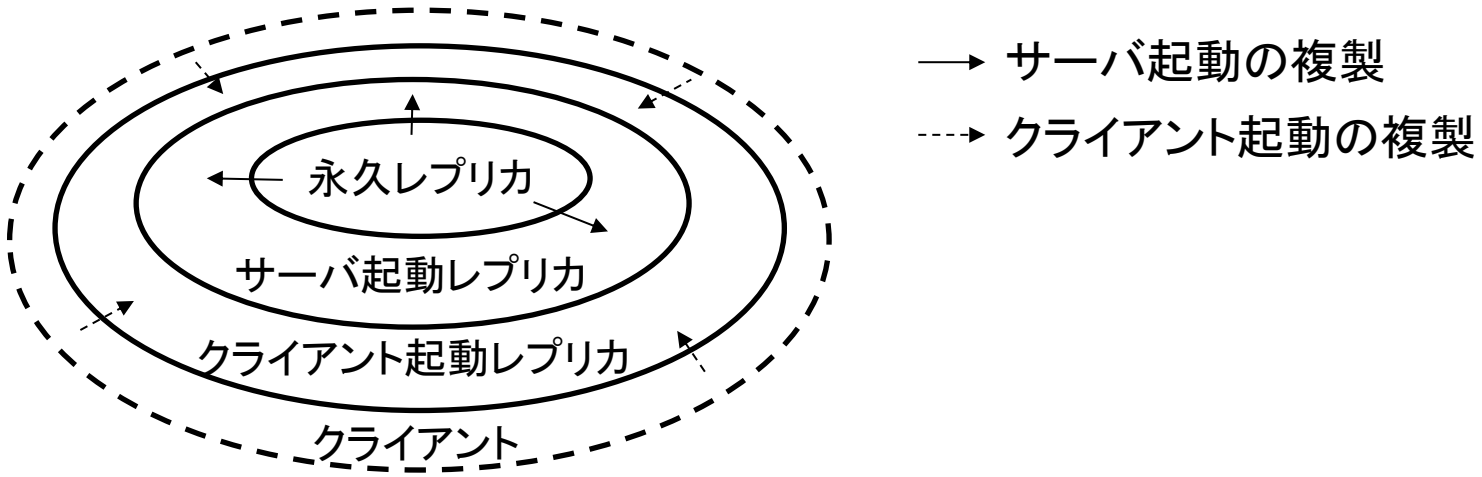
- 読み取り後書き込み（Writes follow reads）

  あるプロセスによるデータ項目 x への読み取り操作に後続する同じプロセスによる書き込み操作は，読み取られた x と同じか，より新しい値でなされることが保証されている

レプリカ管理

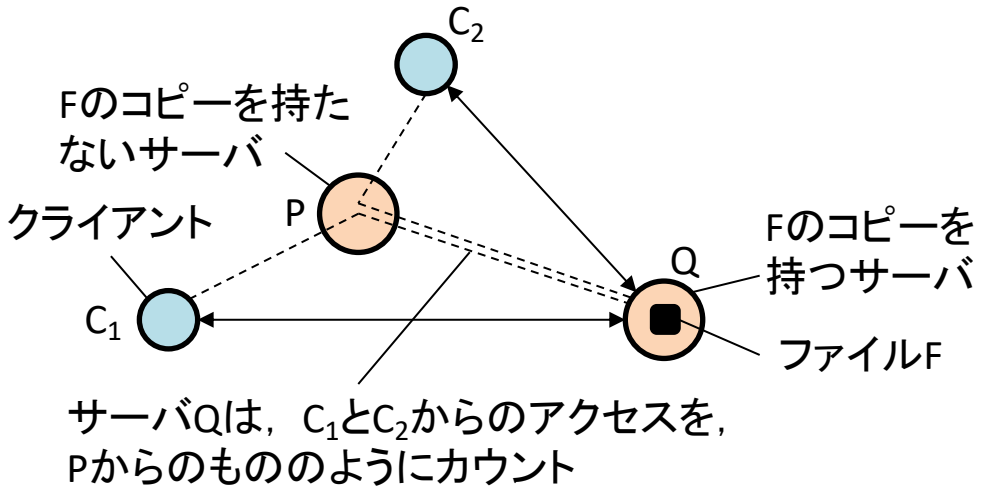
- レプリカが，どこで，いつ，誰によって配置されるか
  - レプリカサーバの配置
  - コンテンツの配置
- レプリカサーバの配置は，N 個の位置の中から最良のK 個の位置を選択するという（計算量の大きい）最適化問題として計算することが可能
- コンテンツの配置は，コンテンツを配置するのに最適なサーバを見つけるという問題を扱う

コンテンツの複製と配置



- 永久レプリカ（Permanent replicas）
  - 分散データストアを構成するレプリカの初期セット
  - **ミラーリング（*Mirroring*）** は分散したWebサイトの一つの形態
- サーバ起動レプリカ（Server-initiated replicas）
  - 性能向上のために，データストア側の主導によって作られるデータストアのコピー
  - 要求がやってきている地域に多くの一時的なレプリカが配置される
- クライアント起動レプリカ（Client-initiated replicas）
  - **クライアントキャッシュ（*client caches*）**として一般的に知られている
  - 取得されたデータがしばらくの間変更されない場合に効果的

サーバ起動レプリカ



- 各サーバは，ファイルごとのアクセス数と，アクセス要求がどこから来たのかを追跡管理
- 各サーバは，あるクライアント C にどのサーバが最も近いのかを決定できる。もし，クライアント C_1 と C_2 が最も近いサーバ P を共有しているならば，C_1 と C_2 からのサーバ Q におけるファイル F へのすべてのアクセス要求は，単一のアクセス数 $\mathrm{cnt}_Q(P, F)$ として Q 上で結合して登録される
- もし，あるサーバ P において，$\mathrm{cnt}_Q(P, F)$ が Q における F へのアクセス要求の半分を超える場合は，Q は F を P へ移動させる
- もし，Q における F へのアクセス要求の総数が削除閾値 $\mathrm{del}(Q, F)$ を下回ったときには，サーバ Q は，それが最後のコピーでない限り，F を削除する

コンテンツ配信

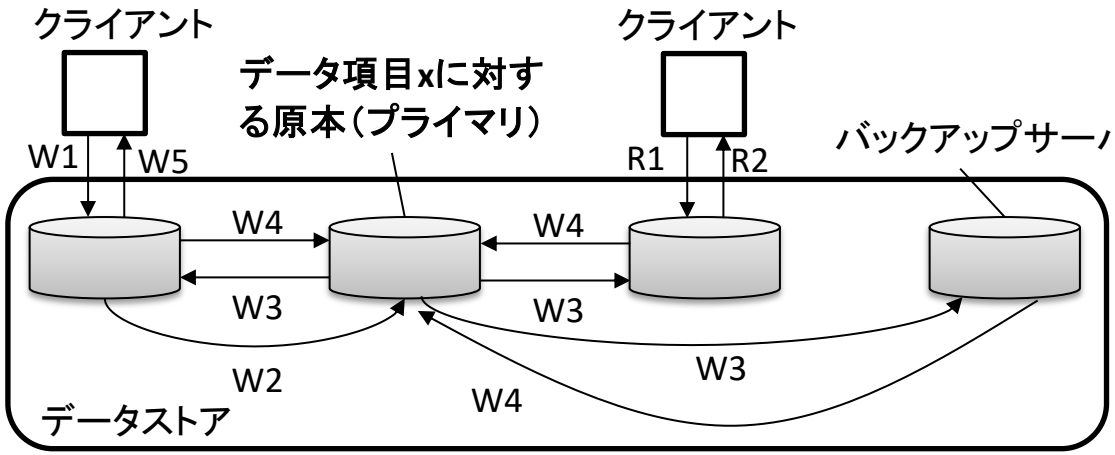
- 更新通知のみを伝播（無効通知プロトコル，***Invalidation protocols***）
  - 他のコピーは，更新が発生し，それらのコピーが保持しているデータは無効であることを知らされる
  - ネットワーク帯域を少ししか使用しないことが利点
  - 読み込み操作に対して書き込み操作が多く発生する（読み書き比率が低い）場合に有効
- あるコピーから他のコピーへデータを転送
  - 読み込み操作に対して書き込み操作があまり発生しない（読み書き比率が高い）
- 更新操作を他のコピーへ伝播（アクティブ複製，***active replication***）
  - どの更新操作を行うべきかをパラメータ値とともに各レプリカに通知
  - 操作に関連するパラメータのサイズが比較的小さい場合に，更新は小さい帯域幅コストで伝搬され得る

コンテンツ配信に影響する要因

- プル対プッシュプロトコル
  - プッシュベースアプローチは，クライアント側に更新を伝播
    - 読み書き比率が高い場合に有効
    - サーバはすべてのクライアントキャッシュの状態を追跡することが必要
  - プルベースアプローチでは，クライアントがサーバに対してその時点で持っている更新を送信することを要求
    - 読み書き比率が低い場合に有効
    - キャッシュミスが発生した場合は応答時間が増加する
- ユニキャスト対マルチキャスト
  - マルチキャストが使える場合，プッシュベースプロトコルが効率よく実装可能
  - ユニキャストは，プルベースプロトコルに対して最良

プライマリベース一貫性プロトコル

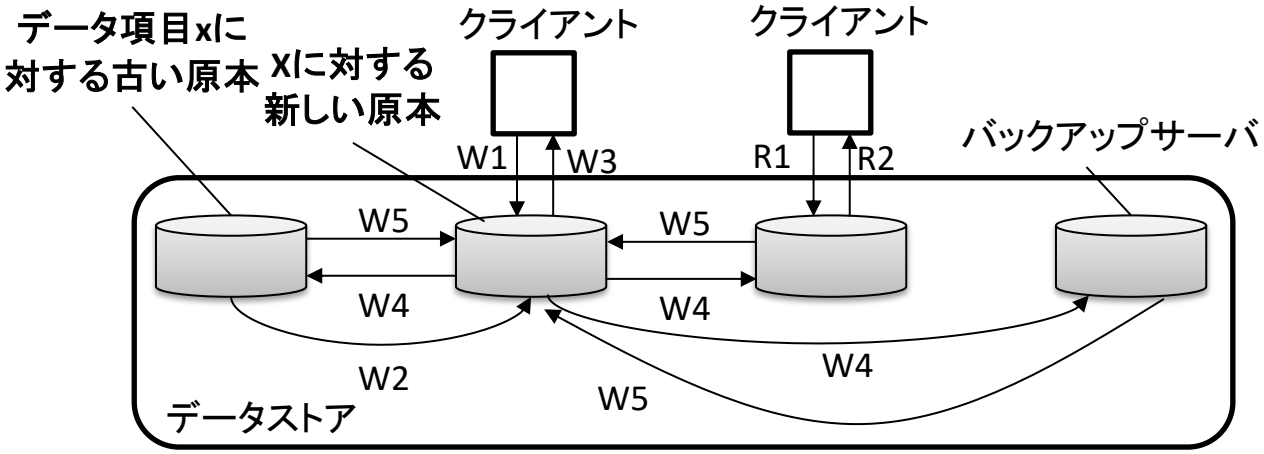
- 遠隔書き込みプロトコル すべての書き込み要求を単一の原本サーバへ転送



W1: 書き込み要求
W2: 要求を原本へ転送
W3: バックアップへ更新を転送
W4: 更新を承認
W5: 書き込み完了を承認

R1: 読み込み要求
R2: 読み込みに対する応答

- ローカル書き込みプロトコル 原本を書き込み操作が行われるサーバへ移動

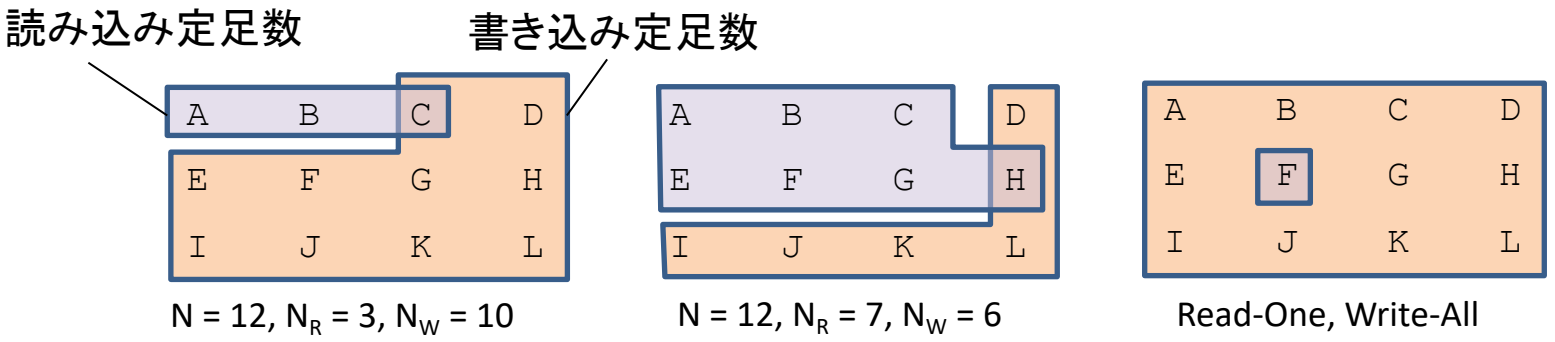


W1: 書き込み要求
W2: データ項目xを新しい原本へ移動
W3: 書き込み完了を承認
W4:バックアップへ更新を転送
W5:更新を承認

R1: 読み込み要求
R2: 読み込みに対する応答

定足数ベース一貫性プロトコル

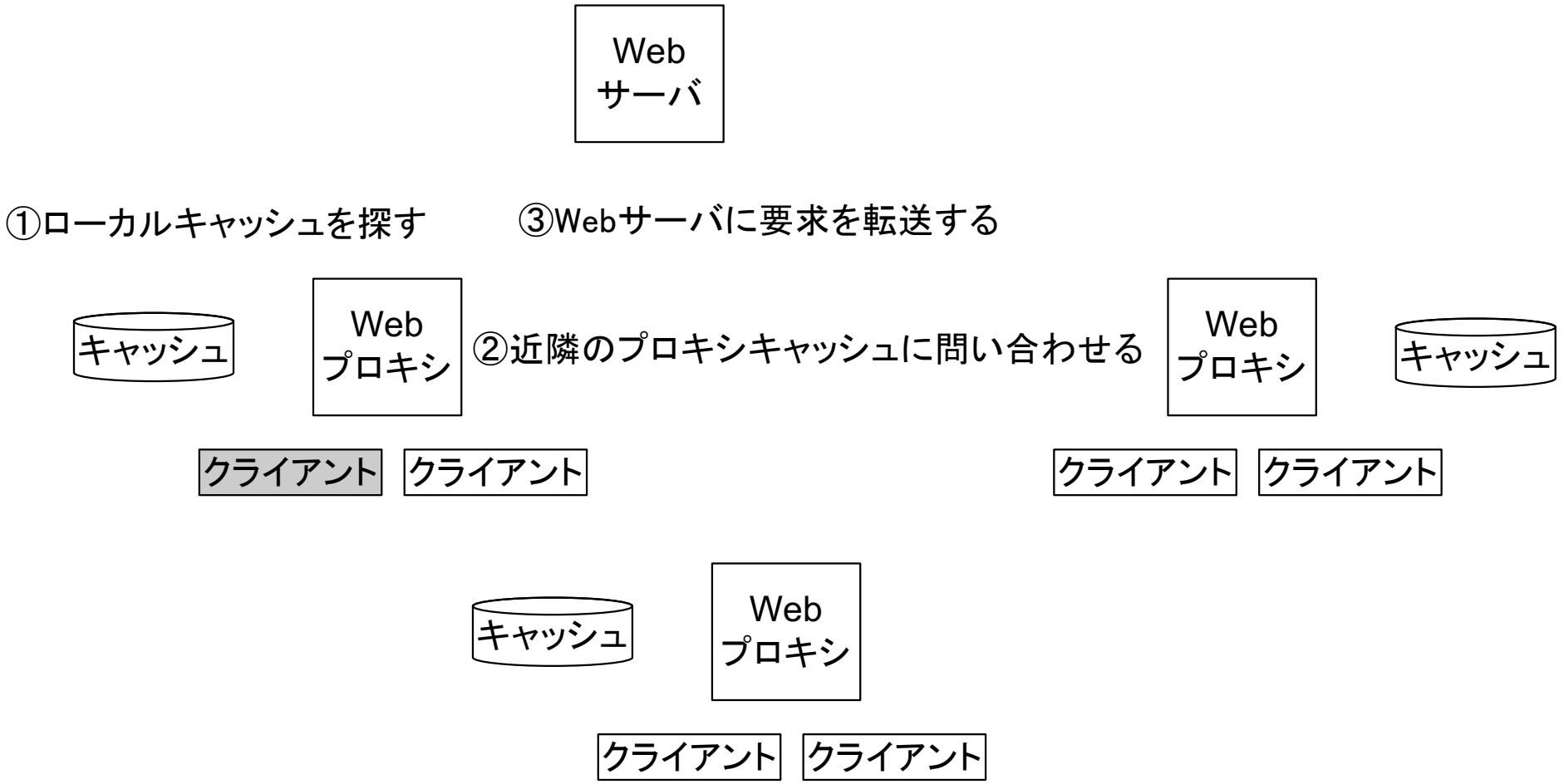
- 書き込み操作を複数の複製上で実行
- クライアントは，複製の読み出し･書き込みの前に，複数のサーバから許可を取得
- ファイルが N 個のサーバに複製され，読み出しには読み出し定足数 N_{R}，書き込みには書き込み定足数 N_{W} が必要
- N_{R} と N_{W} は以下の条件を満たす必要がある
  - $N_{\mathrm{R}} + N_{\mathrm{W}} > N$（読み出し時，少なくとも一つの複製が最新版を保持）
  - $N_{\mathrm{W}} > N/2$（書き込み時，少なくとも一つの最新版が更新される）



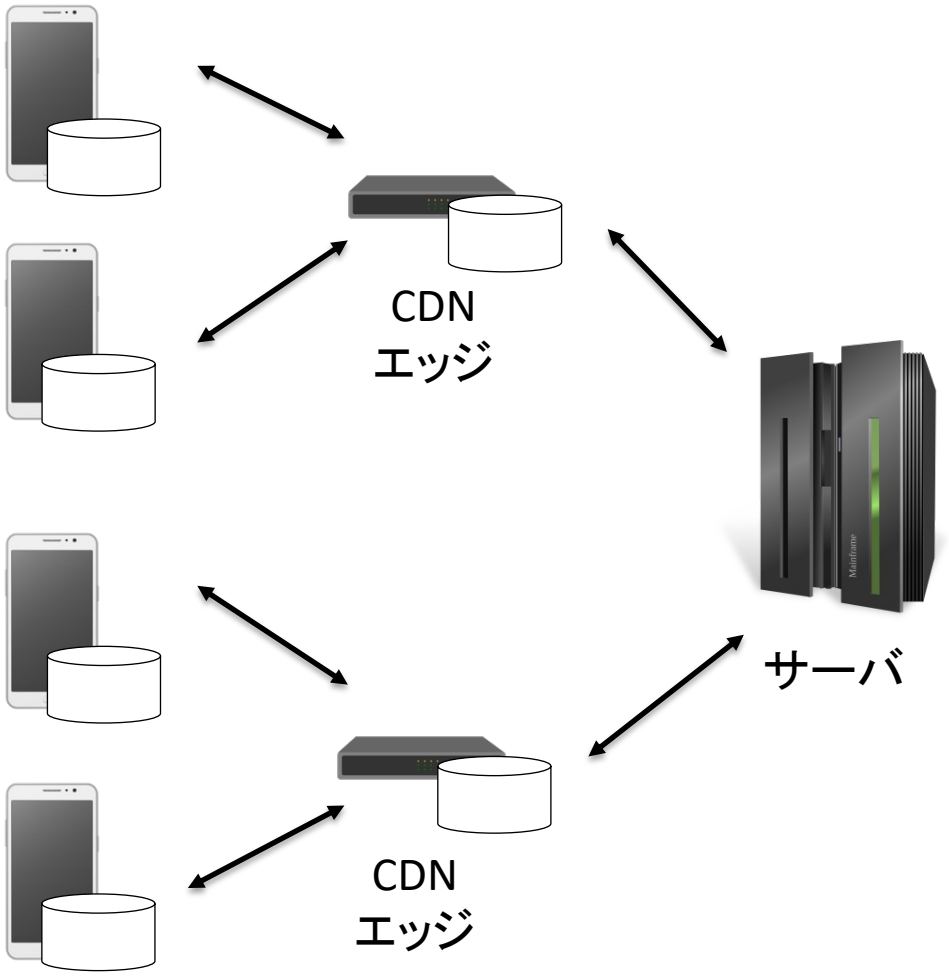
キャッシュ首尾一貫性プロトコル

- 首尾一貫性検出戦略（Coherence detection strategy）
  （不整合が生じた場合の対応）
  A) 静的な解決策: コンパイラが実行前に解析を行い，不整合を防ぐ命令を挿入
  B) 動的検出に基づくプロトコル
     1. クライアントはデータが一貫しているかを検証
     2. 検証を行っている間もトランザクションを進行させ，不整合が検出されたときには，トランザクションをアボート
     3. トランザクションがコミットするときにキャッシュされたデータが最新かどうかを検証し，もし古いデータが使われていれば，トランザクションをアボート
- 首尾一貫性強制戦略（Coherence enforcement strategy）
  （キャッシュの一貫性を保持）
  A) サーバがすべてのキャッシュに無効通知を送信
  B) 単に更新を伝播

Webにおけるキャッシュと複製



CDN (Content Delivery Network)



- プロバイダなどの経路上にキャッシュを設置
- ユーザごとに変化しないデータや，頻繁に更新されないデータをキャッシュ
- DNSにより，クライアントに近いキャッシュへ誘導

DNSによる負荷分散

www.youtube.comのIPアドレスを異なるネットワークから調べてみる

① 立命館大学のネットワークから

```
% nslookup www.youtube.com
...
Non-authoritative answer:
www.youtube.com  canonical name = youtube-ui.l.google.com.
Name:    youtube-ui.l.google.com
Address: 142.250.76.142
Name:    youtube-ui.l.google.com
Address: 142.250.207.110
```

② eo光から

```
www.youtube.com  canonical name = youtube-ui.l.google.com.
Name:    youtube-ui.l.google.com
Address: 172.217.25.174
Name:    youtube-ui.l.google.com
Address: 172.217.161.206
```

③ さくらサーバから

```
www.youtube.com  canonical name = youtube-ui.l.google.com.
Name:    youtube-ui.l.google.com
Address: 142.250.196.110
```