# 分散システム 第8回
## ― 一貫性と複製 ―

大連理工大学・立命館大学 国際情報ソフトウェア学部

大森 隆行

# 講義内容

- 一貫性と複製
- 複製と一貫性モデル
  - 複製とは
  - データ中心一貫性モデル
  - クライアント中心一貫性モデル
- 複製の管理
  - 一貫性の実装

# 複製

- 分散システム上のデータを複製(replication)
- 複製の必要性
  - 信頼性向上
    - 同じデータを複製することで、一つのレプリカ(複製したもの)が使用不可能になっても、他のレプリカに切り替えることができる
  - 性能向上
    - データを近くに配置することで、アクセス時間を短縮
      → システムの規模拡大にもつながる
- 複製の代償
  - 一貫性(consistency)の保持
    - 大規模ネットワークでは一貫性保持のコストが大きい
    - 現実的には一貫性を諦めざるを得ないこともある

# データ中心一貫性モデル
(data-centric consistency model)

- プロセスが特定の条件を満たす場合、
  データストアが正常に動作することを保証
  - データストア(data store)：データの保管場所。
    分散している可能性あり

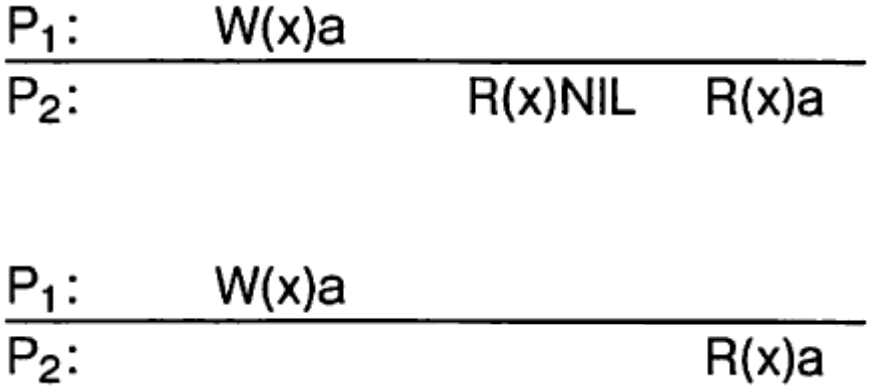
- 「正常」とは？
  - 完璧に一貫性を保証することは困難
    → どのように不一貫性を許容可能かはアプリケーションに依存
  - 連続的一貫性の3つの判断基準：
    - 数値の相違 (例：値の相違が0.5以内、値の相違が1%以内)
    - 陳腐化度の相違
      (例：最新版との相違が1時間以内、更新1回以内)
    - 更新操作の順序の相違

# データ中心一貫性モデル

- 操作の順序に関する一貫性
  - 厳密一貫性：
    すべての操作を絶対時間により順序づける
  - 順序一貫性：
    すべてのプロセスによるデータストアへの
    すべての読み取り、書き込みの操作の結果が、
    それらの操作が一定の順序で実行された結果と同じになる
  - 因果一貫性：
    前の操作の影響を受けている可能性のある書き込みは、
    すべてのプロセスによって、同じ順序で
    観測されなければならない

# 厳密一貫性

- すべての操作を絶対時間により順序づける
  - 書き込み後には、その書き込みが反映されていなければならない
    → 実システムでは遅延があるため実現不可能




厳密一貫性がない

プロセスP1によってxにaが書き込まれた後、
プロセスP2がxを読み込んだらNIL(値未設定)を得た。
その後プロセスP2がxを読み込んだらaを得た。



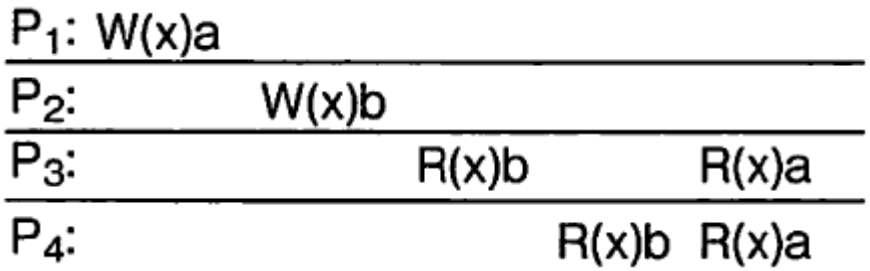

厳密一貫性がある

プロセスP1によってxにaが書き込まれた後、
プロセスP2がxを読み込んだらaを得た。

※記号の意味
W(x)a 「xにaを書き込む」
R(x)a 「xを読み込んだ結果aを得る」
Piはプロセス。横軸は時間。

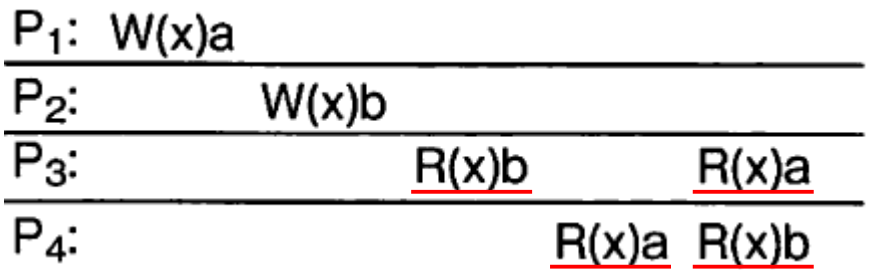
# 順序一貫性

- すべてのプロセスによるデータストアへのすべての読み取り、書き込みの操作の結果が、それらの操作が一定の順序で実行された結果と同じになる




順序一貫性がある

※実際の書き込み順と逆になっていてもOK




順序一貫性がない

←先にbが書き込まれたことになっている

←先にaが書き込まれたことになっている

※記号の意味
W(x)a 「xにaを書き込む」
R(x)a 「xを読み込んだ結果aを得る」
Piはプロセス。横軸は時間。

# 因果一貫性

- 前の操作の影響を受けている可能性のある書き込みは、すべてのプロセスによって、同じ順序で観測されなければならない
  - 前の操作の影響を受けている可能性がある
    ＝ 因果的に関連する

bはaの影響を受けているかもしれない

| | | | | | |
|---|---|---|---|---|---|
| $P_1$: | W(x)a | | | W(x)c | |
| $P_2$: | | R(x)a | W(x)b | | |
| $P_3$: | | R(x)a | | R(x)c | R(x)b |
| $P_4$: | | R(x)a | | R(x)b | R(x)c |

因果一貫性があるが、
順序一貫性はない

※bとcは因果的に関連がない

| | | | | |
|---|---|---|---|---|
| $P_1$: W(x)a | | | | |
| $P_2$: | R(x)a | W(x)b | | |
| $P_3$: | | | R(x)b | R(x)a |
| $P_4$: | | | R(x)a | R(x)b |

因果一貫性がない

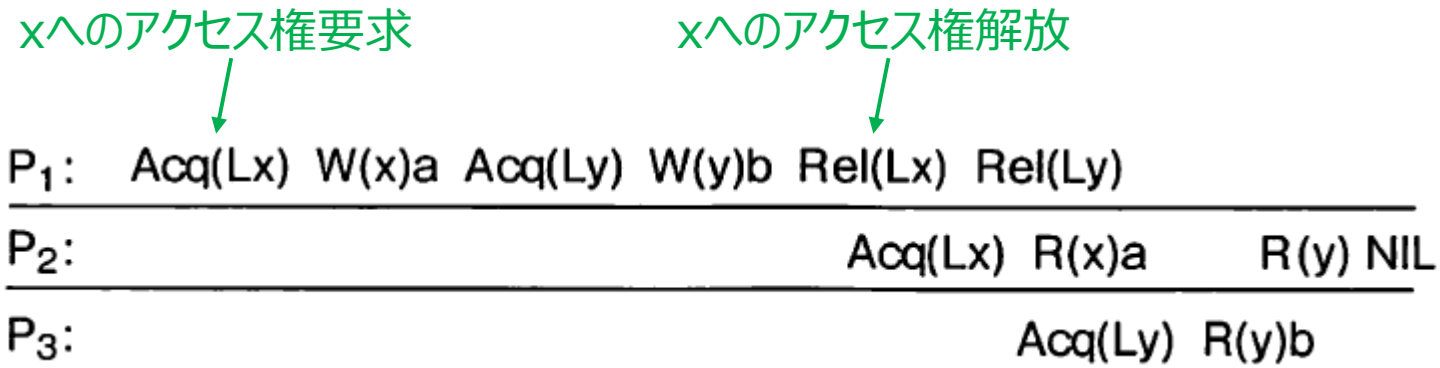
| | | | | |
|---|---|---|---|---|
| $P_1$: W(x)a | | | | |
| $P_2$: | | W(x)b | | |
| $P_3$: | | | R(x)b | R(x)a |
| $P_4$: | | | R(x)a | R(x)b |

因果一貫性がある

※記号の意味
W(x)a 「xにaを書き込む」
R(x)a 「xを読み込んだ結果aを得る」
Piはプロセス。横軸は時間。

# エントリー貫性

- 操作のグループ化に関する一貫性
- 操作のグループ化
  - 実際の分散システムではデータストア内のデータへのアクセスをロック・解放して、アクセス権をあるプロセスが占有することが多い
  - データの読み書きを1つのトランザクションとしてグループ化




P1: x,yへのアクセス権を取得し、値a,bを書き込んでいる
P2: xへのアクセス権を取得し、値aを取得しているが、
yへのアクセス権を取得していないので結果がNILとなる
P3: yへのアクセス権を取得し、値bを取得している

# 確認問題

※記号の意味
W(x)a 「xにaを書き込む」
R(x)a 「xを読み込んだ結果aを得る」
Piはプロセス。横軸は時間。

- 順序一貫性が保たれているデータストアは、以下のうちどれか。

A

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

B

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

C

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

D

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

# クライアント中心一貫性モデル

- あるクライアントから見てデータに一貫性があるように見えることを保証
  - 他のクライアントから見ると違った結果になっているかもしれない

# イベンチュアル一貫性

eventual [Adj] 最終的な、やがて起きる

- 時間が経過するとすべての複製が最新の状態に収束
  - アクセスするレプリカによっては新しい変更が反映されていない可能性がある
  - (例) Webキャッシュ
    - キャッシュは必ずしも最新ではない(ある程度の非一貫性を許容)
- 競合(conflict)への対処
  - 書き書き競合→更新プロセスを限定することで解消(実装が比較的容易なことが多い)
  - 読み書き競合→問題にしない(読み込んだものが最新でないことを許容)

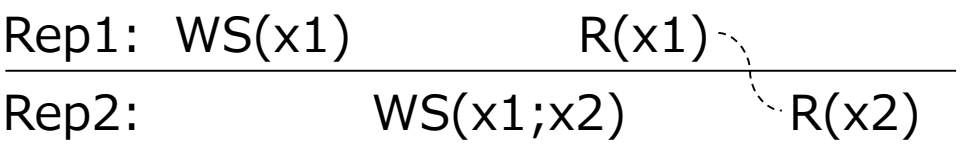
# クライアント中心一貫性モデル

- モノトニック読み取り一貫性

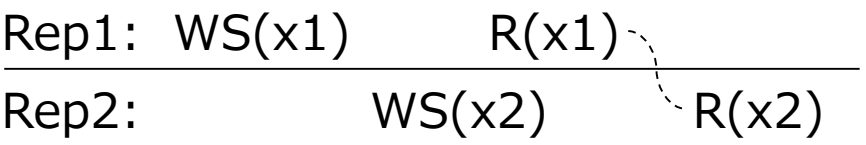
monotonic [Adj] 単調な

  - もし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についての後続のどの読み取りも
    同じ値を返答するか、またはより新しい値を返答する

モノトニック読み取り一貫性がある




モノトニック読み取り一貫性がない




同じプロセスが異なるレプリカにアクセスする可能性があるため、
レプリカへの書き込み時に、前の変更の反映が保証されていなければならない

※記号の意味
WS(xi) 「Repiにおけるデータxに対する書き込み処理の集合」
WS(xi;xj) 「Repjにおける書き込みの前にRepiにおけるそれまでの書き込みが反映されている」
R(x) 「データxを読み込む処理」
横軸は時間。破線は同一プロセスによる実行順序を示す。

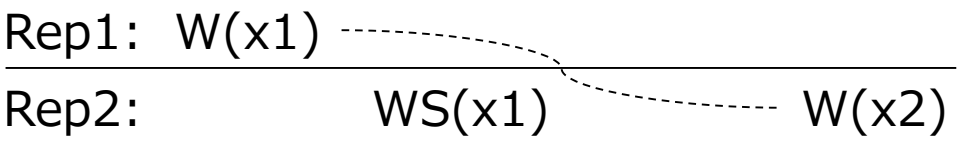
# クライアント中心一貫性モデル

- モノトニック書き込み一貫性

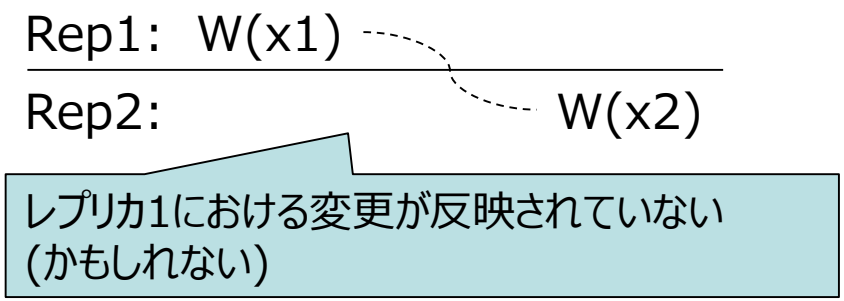
  monotonic [Adj] 単調な

  - あるデータ項目xへの1つのプロセスによる書き込みは、同じプロセスによるxへのどの後続の書き込みよりも前に完了されている

モノトニック書き込み一貫性がある




モノトニック書き込み一貫性がない

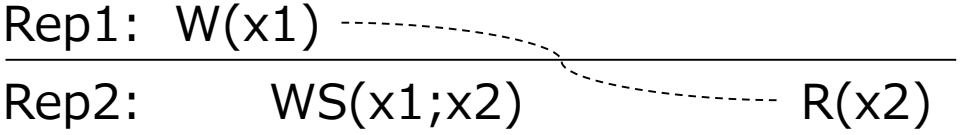



※記号の意味
WS(xi) 「Repiにおけるデータxに対する書き込み処理の集合」
W(xi) 「Repiにおけるデータxに対する書き込み処理」
横軸は時間。破線は同一プロセスによる実行順序を示す。

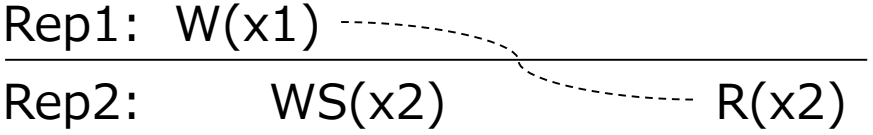
# クライアント中心一貫性モデル

- 書き込み後読み取り一貫性
  - データ項目xへのあるプロセスによる書き込みの結果は、同じプロセスによる後続する読み取りによって常に観測される
- 読み取り後書き込み一貫性
  - あるプロセスによるデータ項目への読み取りに後続する、同じプロセスによる同じデータ項目への書き込みが、読み取られたxの値と同じまたはより新しい値でなされる

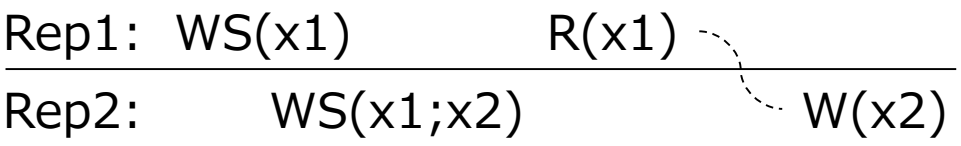
書き込み後読み取り一貫性がある



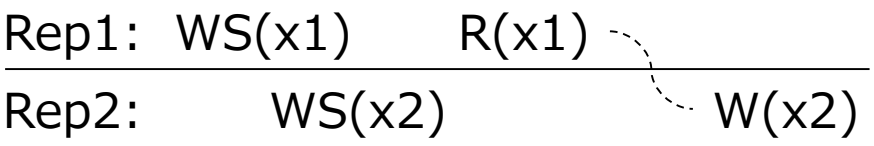

書き込み後読み取り一貫性がない




読み取り後書き込み一貫性がある




読み取り後書き込み一貫性がない



# 確認問題

※記号の意味
WS(xi)「Repiにおけるデータxに対する書き込み処理の集合」
WS(xi;xj)「Repjにおける書き込みの前にRepiにおけるそれまでの書き込みが反映されている」
R(x)「データxを読み込む処理」
横軸は時間。破線は同一プロセスによる実行順序を示す。

- モノトニック読み取り一貫性が保たれているデータストアは、以下のうちどちらか。

A

| | | |
|---|---|---|
| Rep1: WS(x1) | | R(x1) |
| Rep2: | WS(x1;x2) | |
| Rep3: | WS(x3) | R(x3) |

B

| | | |
|---|---|---|
| Rep1: WS(x1) | | R(x1) |
| Rep2: | WS(x1;x2) | |
| Rep3: | WS(x1;x3) | R(x3) |

- 書き込み後読み取り一貫性が保たれているデータストアは、以下のうちどちらか。

A

| | | |
|---|---|---|
| Rep1:W(x1) | | R(x1) |
| Rep2: | WS(x2) | |
| Rep3: | WS(x1;x3) | R(x3) |

B

| | | | |
|---|---|---|---|
| Rep1: W(x1) | | R(x1) | |
| Rep2: | WS(x2) | R(x2) | |
| Rep3: | WS(x1;x3) | | R(x3) |

# 講義内容

- 一貫性と複製
  - 複製と一貫性モデル
    - 複製とは
    - データ中心一貫性モデル
    - クライアント中心一貫性モデル
  - 複製の管理
    - 一貫性の実装

# レプリカの管理

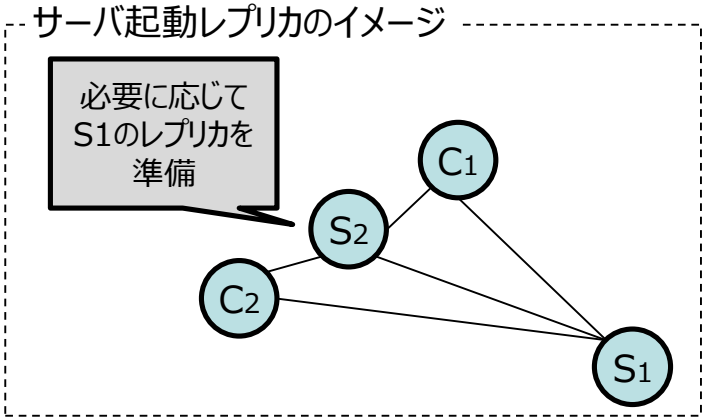
- どこにいつ誰によってレプリカが配置されるか
- 一貫性を維持するにはどのようなメカニズムを使用するべきか

# レプリカサーバ配置

- 最適化問題と言うより、管理や商売上の側面が大
- N個の位置の中から最善のK個の位置を選択
  - サーバとクライアントの距離が最小になるようにする
- ノード間の伝送遅れが小さい領域を選択し、その中にサーバを配置
  - 領域サイズが大きすぎると性能低下
  - 小さすぎるとサーバ過大

# コンテンツの複製と配置

- 3種類のレプリカ
  - 永久レプリカ
    - 分散データストアを構成するレプリカの初期セット
    - (例) ミラーリング：あるサーバのデータをミラーサーバにコピーする
  - サーバ起動レプリカ
    - サーバの要求によって作成されるレプリカ
    - (例) サーバ内のレプリカを動的に生成・削除
  - クライアント起動レプリカ
    - クライアントの要求によって作成されるレプリカ
    - (例) クライアントキャッシュ



# 一貫性プロトコル

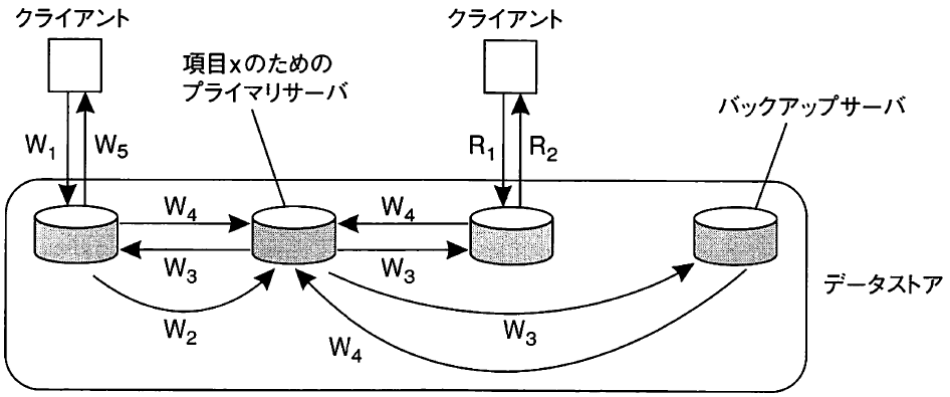
- 連続的一貫性
  - 数値的相違の制限
    - 書き込み前後の値の(絶対的or相対的な)差を記録し、差が閾値以上になると伝播
  - 陳腐化相違の制限
    - 書き込みのタイムスタンプを利用
    - 変更時に変更をプッシュするアプローチの場合、遅延の大きさによっては一貫性を保証できない
      → プル方式である程度は改善
  - 順番的相違の制限
    - 書き込みの待ち行列が最大長を越えたとき、他のサーバと交渉しつつ書き込みを実行

# 一貫性プロトコル

- プライマリベースプロトコル
  - データストア内の各データ項目xは関連するプライマリを持つ
  - プライマリ：xに対する書き込みを協調させる責任
  - 遠隔書き込みプロトコル
  - ローカル書き込みプロトコル

# 遠隔書き込みプロトコル

- すべての書き込み操作を
  一つのサーバ(プライマリサーバ)に送る
- サーバは複製を更新後、書き込みを要求した
  プロセスに通知
- 長所：順序一貫性が保証される
- 短所：書き込み後の処理再開が遅くなる

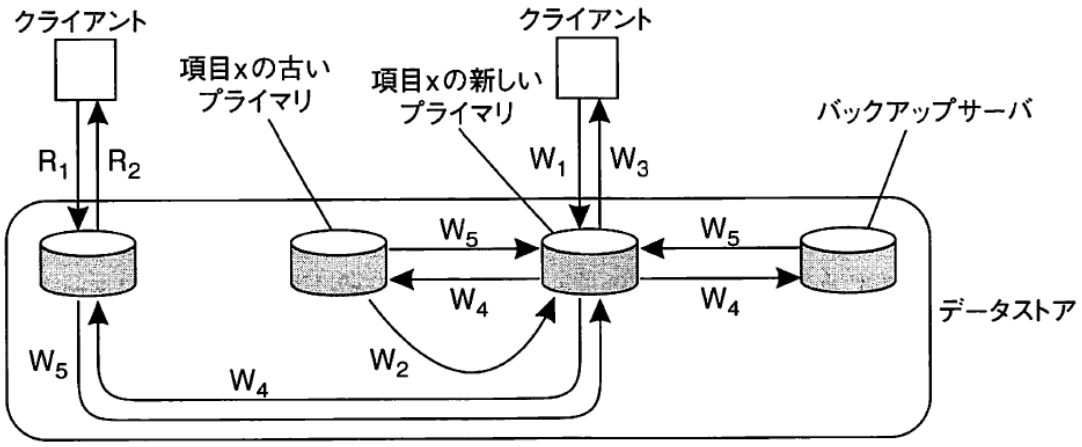



$W_1$. 書き込み要求
$W_2$. プライマリへ要求を転送する
$W_3$. バックアップに更新を告げる
$W_4$. 更新の確認通知
$W_5$. 書き込み完了の確認通知

$R_1$. 読み取り要求
$R_2$. 読み取りへの応答

# ローカル書き込みプロトコル

- 書き込み要求が発生すると、プライマリを
  ローカルサーバに移動して書き込みを行う




$W_1$. 書き込み要求
$W_2$. 項目xを新しいプライマリに移動する
$W_3$. 書き込み完了の確認通知を行う
$W_4$. バックアップに更新を告げる
$W_5$. 更新の確認通知を行う

$R_1$. 読み取り要求
$R_2$. 読み取りへの応答

- 複数の後続の読み書きがローカルで行える
- 非接続モードでローカルのみ更新しておく場合に利用可能

# クライアント中心一貫性の実装

- モノトニック読み取り一貫性の実装
  - すべての書き込み操作にIDを割り付け(以下共通)
  - 読み取りの前に、ID付き書き込みがローカルに反映されたか確認
  - 反映されていないなら、他のサーバから書き込み情報をもらう
    or 他のサーバに読み取り要求を送る
- モノトニック書き込み一貫性の実装
  - 新しい書き込み処理時、クライアントの
    書き込み集合が渡される
- 書き込み後読み取り一貫性の実装
  - 読み取り時、先に他のサーバから書き込み情報をもらう
- 読み取り後書き込み一貫性の実装
  - 初めの読み取り時に、クライアントにサーバ上の書き込み
    操作のIDを保存
  - 次にアクセスしたサーバに保存したIDを渡す

# 確認問題

- 下記のレプリカは語群に示した種類のうちどれに該当するか
  - クライアントが自らのキャッシュとしてデータを保持するもの
  - サーバがよりクライアントに近い他のサーバにデータをコピーするもの
  - あるサーバのデータを常に他のサーバにミラーリングするもの

    語群：サーバ起動レプリカ、クライアント起動レプリカ、永久レプリカ
- レプリカの管理における遠隔書き込みプロトコルの長所は以下のうちどれか
  - プライマリサーバに接続していなくても更新可能
  - 複数の更新がある場合、後続の更新をローカルで行える
  - 順序一貫性が保証される

# 参考文献


- 「分散システム」
  水野 忠則 監修、共立出版、2015
- 「分散システム 原理とパラダイム 第2版」
  アンドリュー・S・タネンバウム 他 著、
  ピアソン・エデュケーション、2009