Project Tsurugi
SQL処理の実装

ノーチラス・テクノロジーズ

2020-10

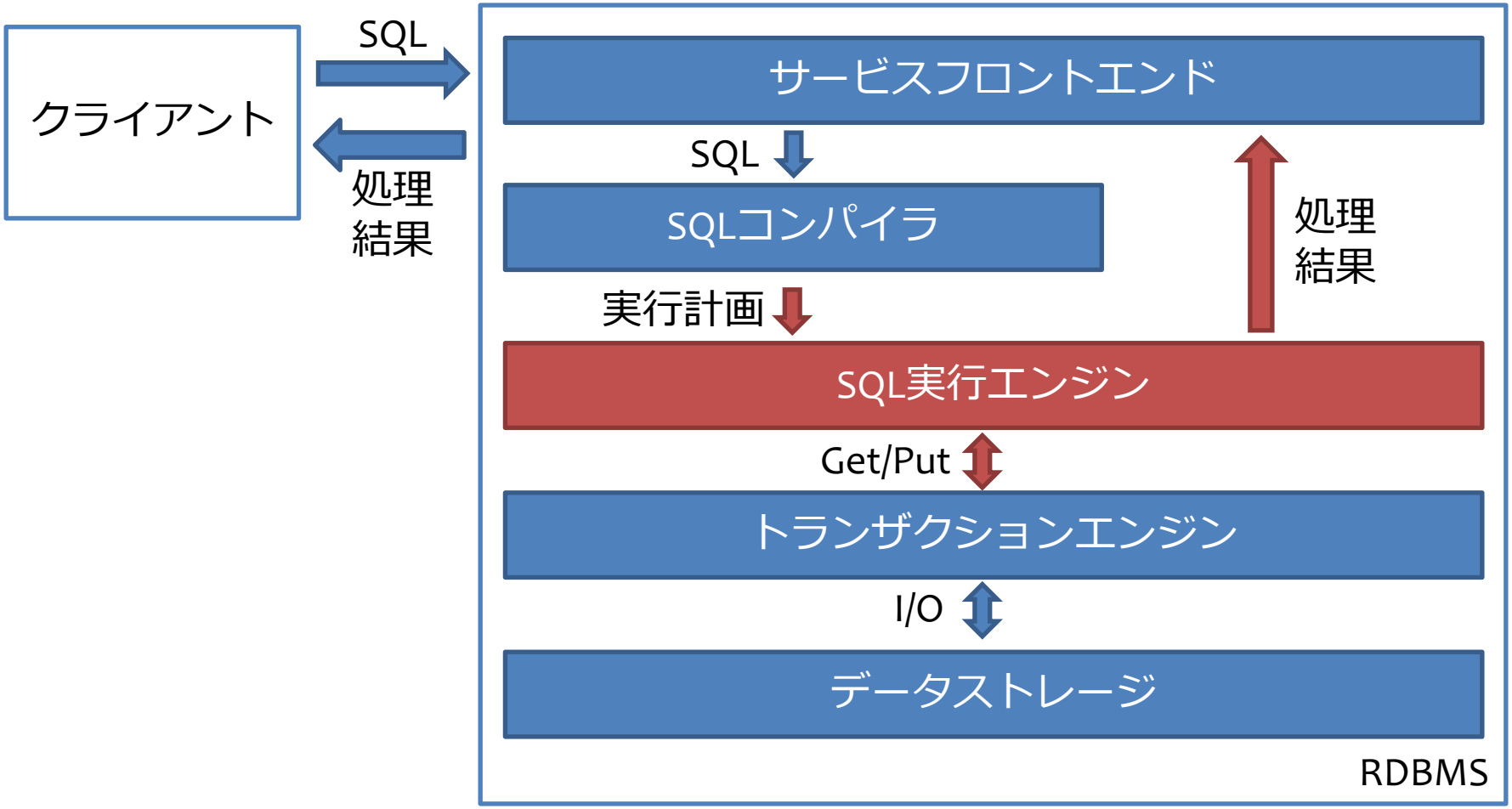
NAUTILUS

Agenda

- SQL実行エンジンの役割
- SQL実行エンジンの目指すもの
- 現代的HWにおけるデータ処理の課題
- SQL処理におけるデータ並列
- shuffle処理によるデータ分割
- shuffleにおけるキャッシュの効率的な利用
- shuffleはなぜsortベースか
- その他

SQL実行エンジンの役割

- SQLで記述されたデータ処理を実行



Tsurugi SQL実行エンジンの目指すもの

- SQL処理をインメモリで高速に実行
  - 特性の異なるワークロード両方に対応
    - 比較的単純な大量のSQL処理
    - 大規模なデータ処理を行う複雑なSQL
  - SQL内の関係演算子を高速に並列処理
    - inter-/intra-operator parallel
    - inter-query parallel

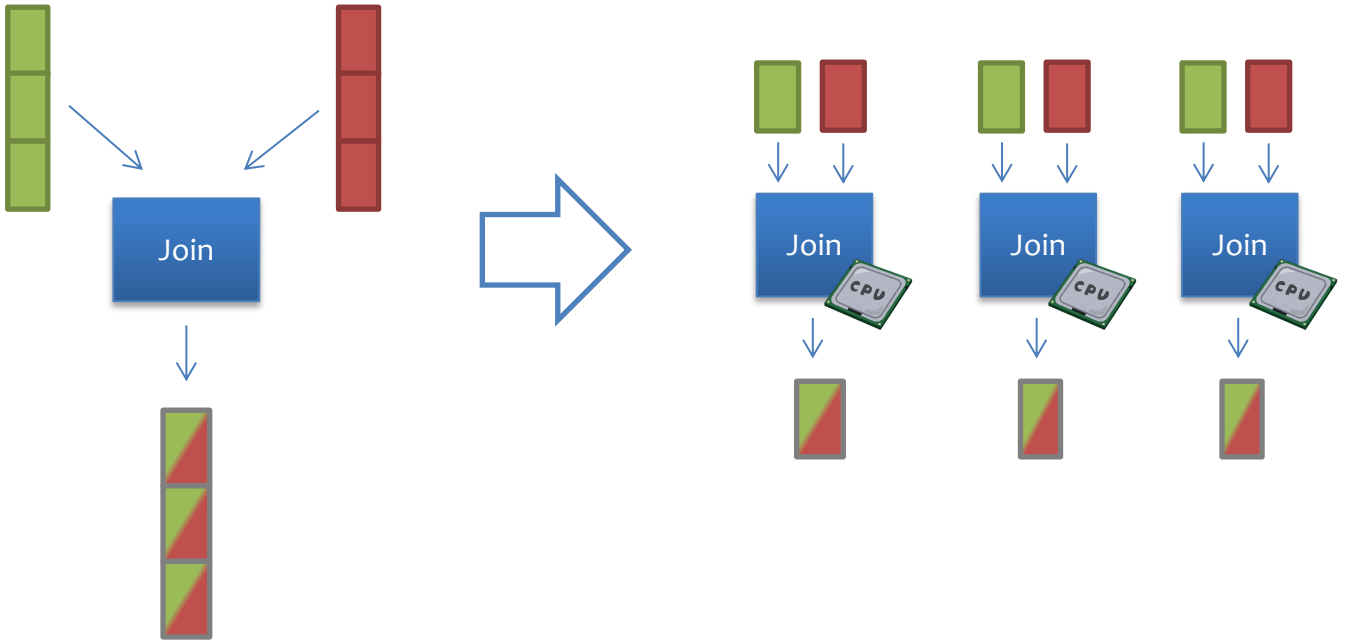
- 現代的なアーキテクチャのHW性能を活用
  - メニーコア - 数十~数百コアを利用可能
  - 大容量メモリ - 数TBの記憶領域

現代的HWにおけるデータ処理の課題

- 課題
  - 多数のCPUコアを十分に活用するためにプログラムとデータを効率よくコアに供給し続ける必要がある
  - メモリのアクセス速度の進歩は容量やCPUコア数の増加よりも緩やかであり、相対的にボトルネックとなりやすい
- 解決案
  - データを適切に分割しデータ並列を実現することによってコアを十分に活用する
  - メモリキャッシュを適切に利用して、メモリ本体へのアクセスを減らす

SQL処理におけるデータ並列

- SQL処理は関係演算子グラフに沿ったデータフロー処理
- 入力データを分割することで、関係演算子を並列実行可能にしたい



SQL処理におけるデータ並列の難しさ

- SQLの関係演算は素直にデータ並列化できない
  - 入力を単純に分割しても正しい結果を得られない
  - 重要な演算子の多くはグループ単位で処理を行う

- グループを壊さないようにデータを分割する必要がある
  - 等結合であれば結合に使用しているキーで分割
  - 集約の場合はグループ化キーで分割

例：Joinに対する適切な分割

- 下記のJoinを分割することを考える

X

C1	C2	
A	1	
B	2	

Y

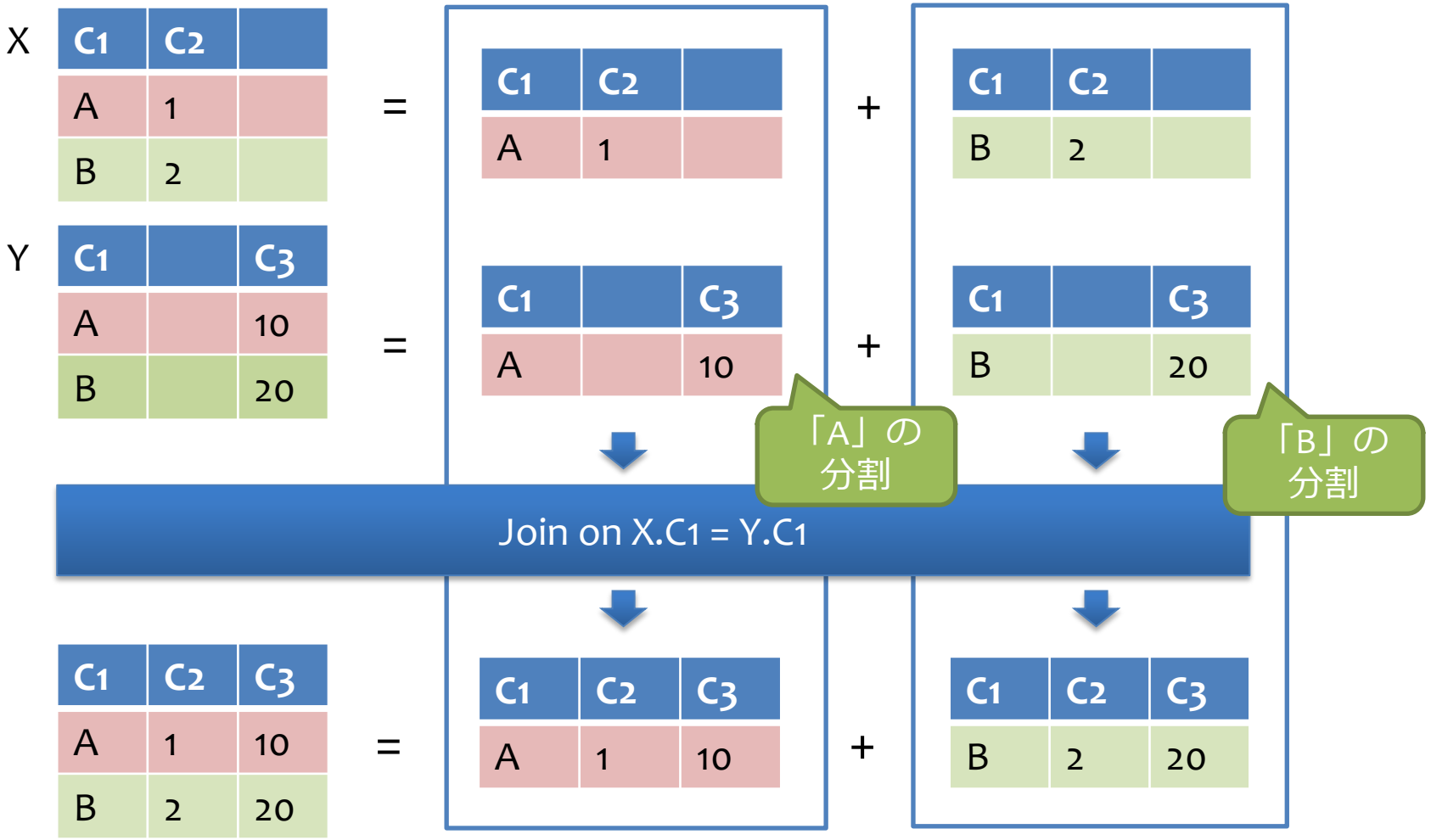
C1		C3
A		10
B		20

Join on X.C1 = Y.C1

C1	C2	C3
A	1	10
B	2	20

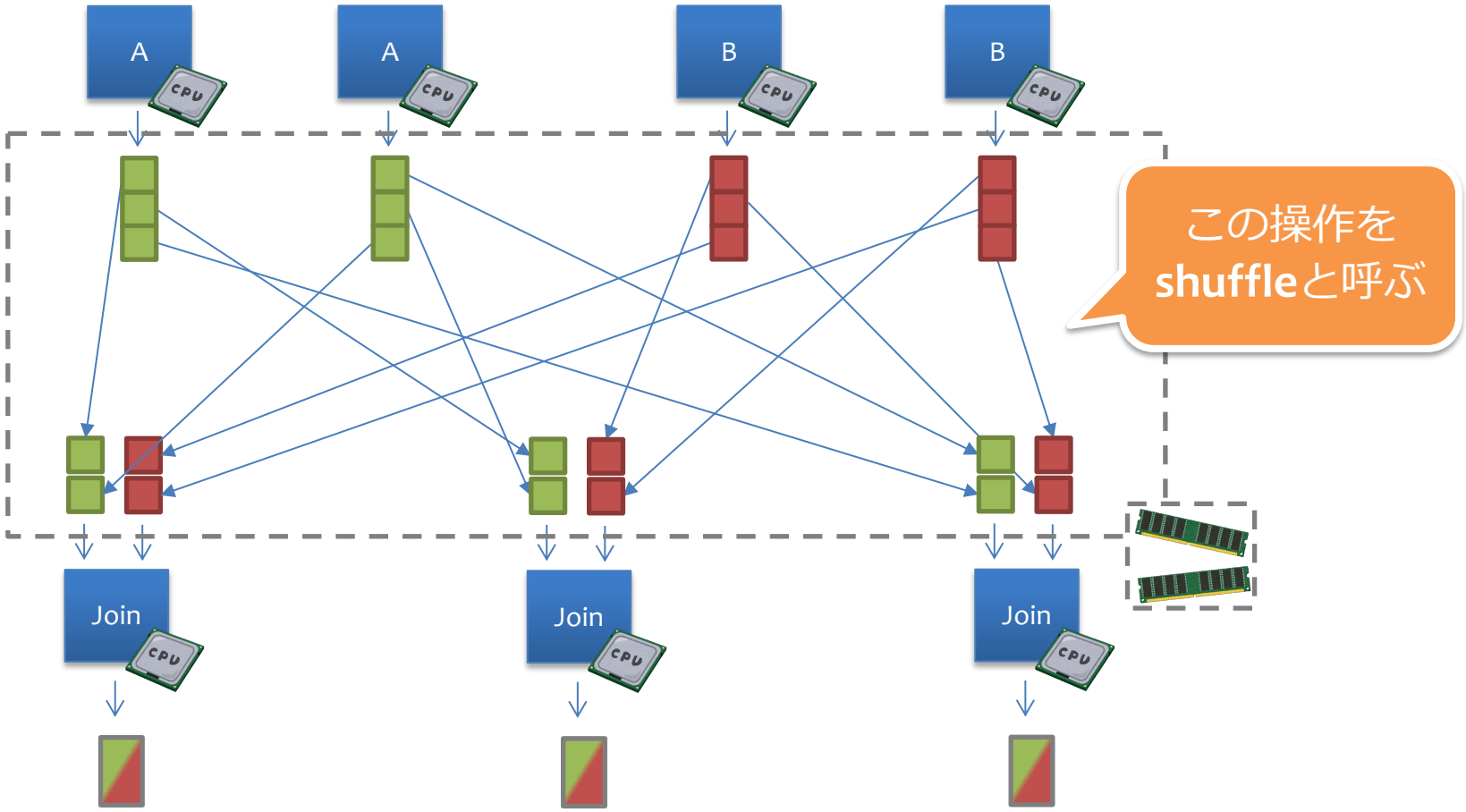
例：Joinに対する適切な分割

- 等結合キーC1が等しい行は同じ分割単位に所属



Tsurugi 実行エンジンにおけるデータ分割(1/2)

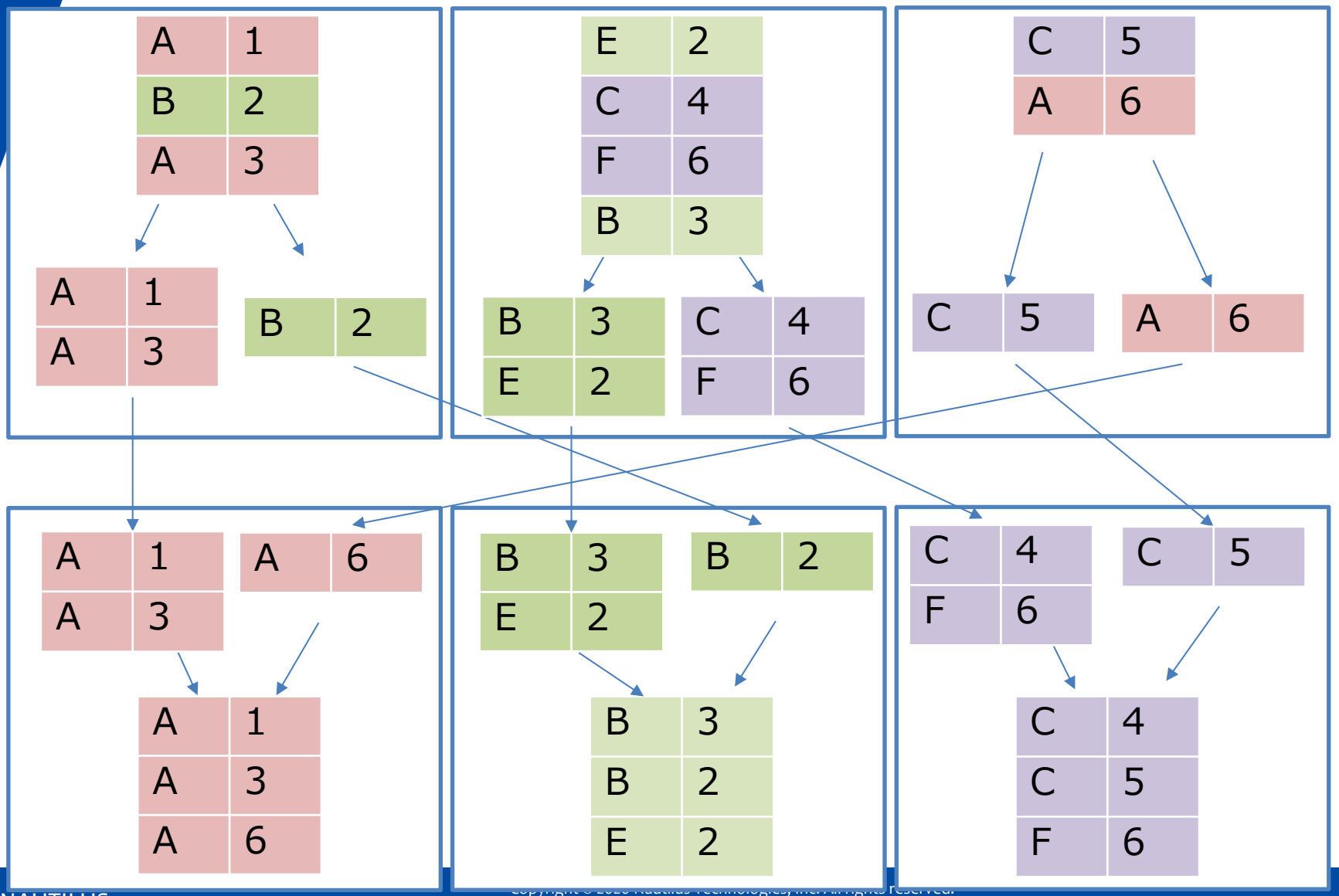
大規模メモリを活かし、インメモリでデータを分割



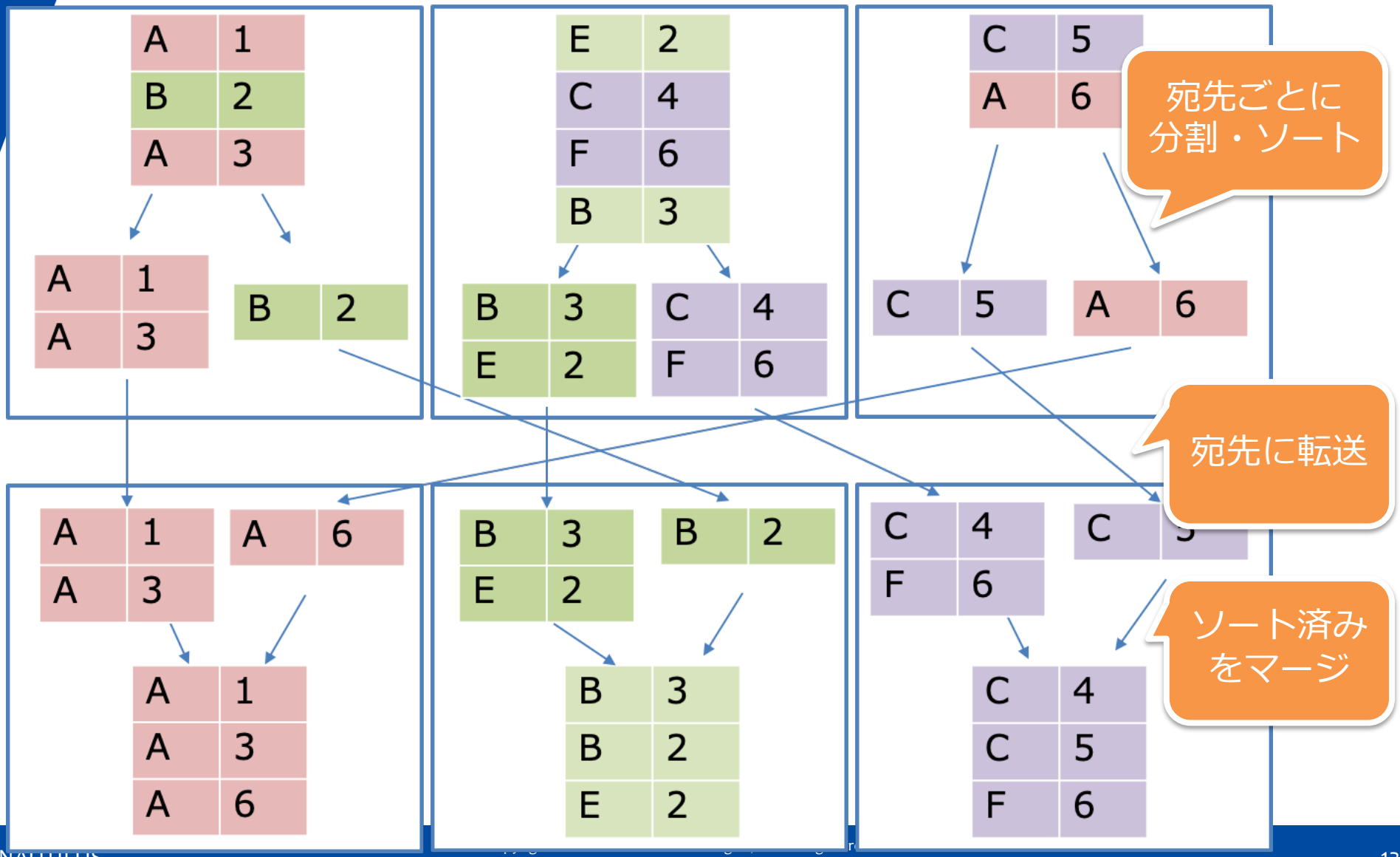
Tsurugi 実行エンジンにおけるデータ分割(2/2)

- メモリ上のshuffle処理は、多くのコアが同時に大量の読み書きを行い、メモリアクセスが逼迫しやすい
- そこでメモリキャッシュの活用に効果的なアルゴリズムを用い、メモリへの直接アクセスを削減する
  - 複数のコアで同じ領域を同時に扱わない
  - シーケンシャルアクセスを多用しプリフェッチ効率を高める(空間局所性)
  - あるメモリアドレス上のデータを複数回アクセスする場合、その間の処理を最小にする (時間局所性)

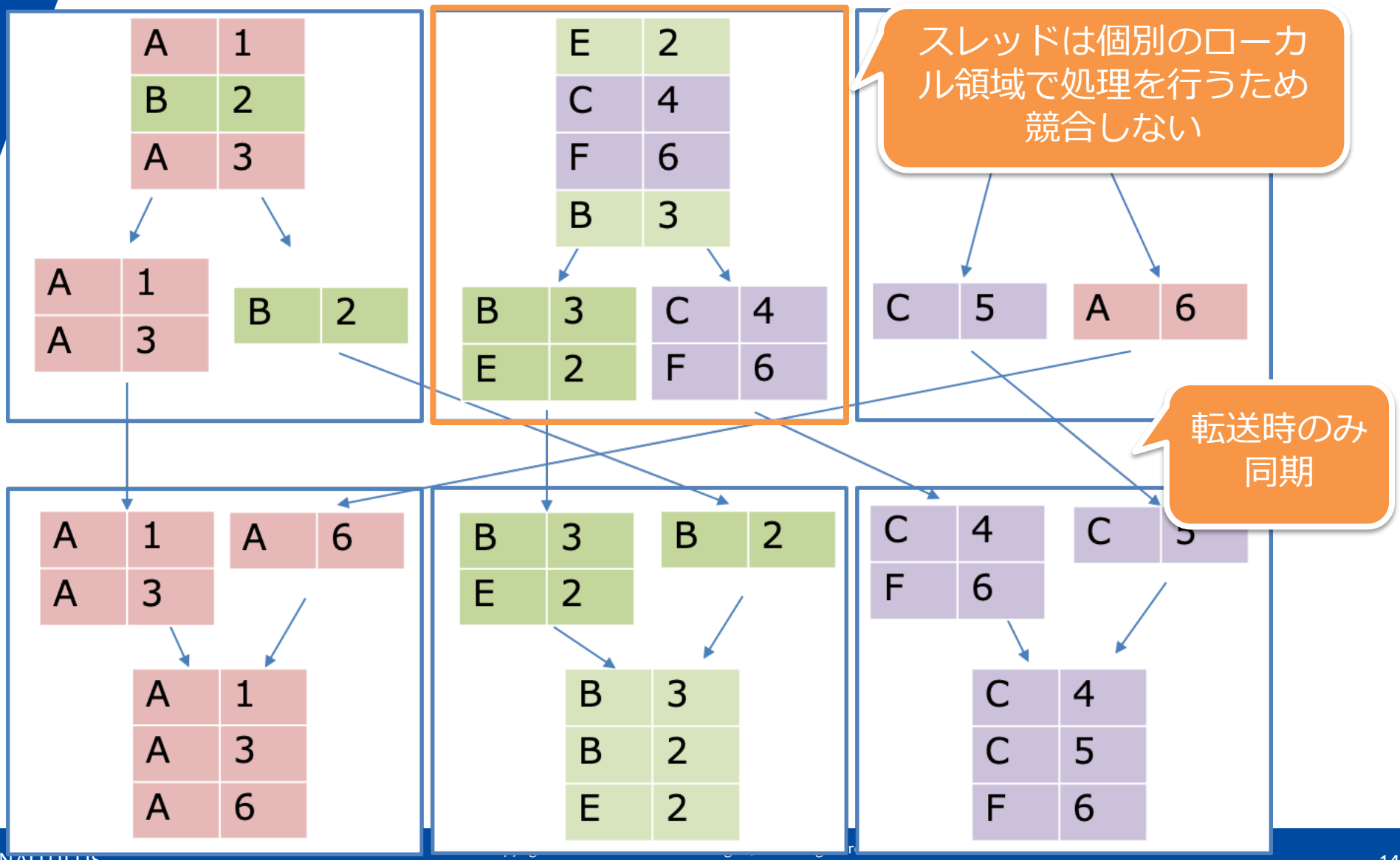
shuffle処理の例(1/3)



shuffle処理の例(2/3)

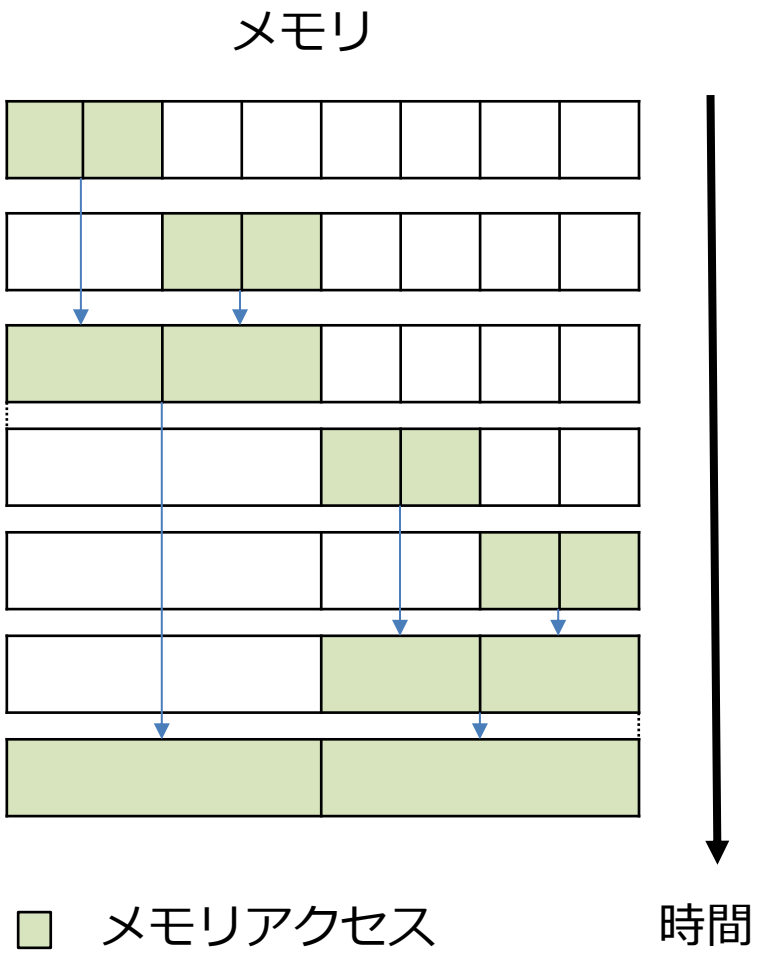


shuffle処理の例(3/3)



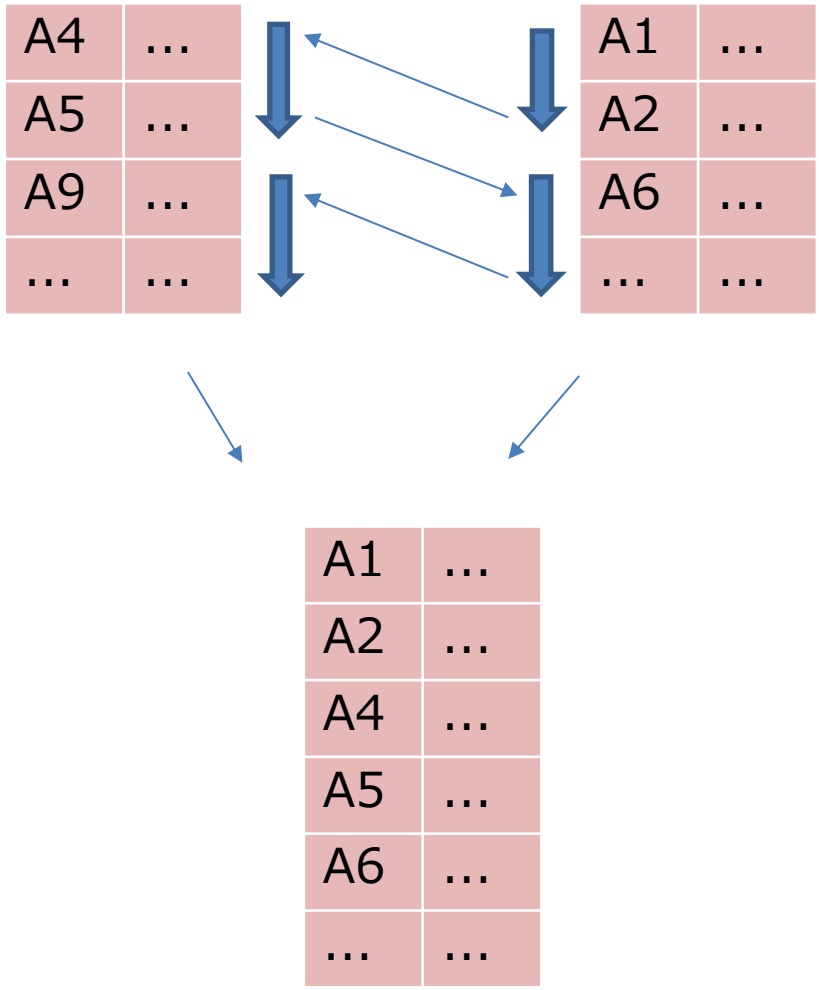
shuffleにおけるキャッシュの効率的な利用(1/2)

- ソートによる処理方式を採用
  - 大規模ソートは分割統治方式が多い
  - 時間/空間局所性が高く、キャッシュを活かしやすい
- C++標準ライブラリのintro sortを利用
  - quick sortの改良版で分割統治方式



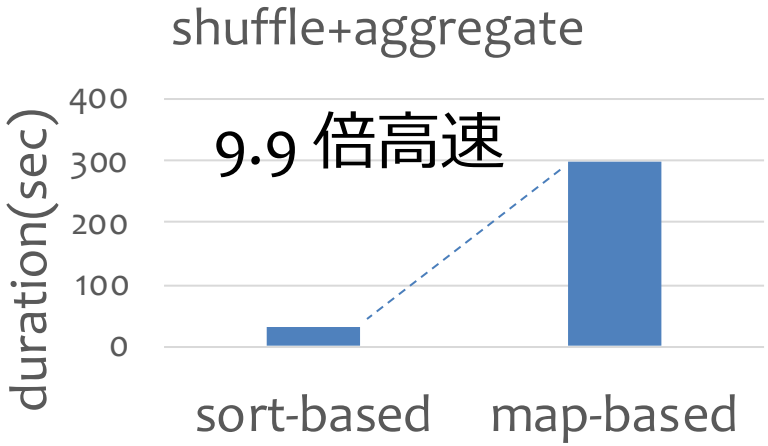
shuffleにおけるキャッシュの効率的な利用(2/2)

- マージ処理は、個々のソート済み領域にシーケンシャルアクセス
  - キャッシュプリフェッチが活きて効率が良い
  - 個々のソート済み領域には、別ソケット上のメモリもあるため、キャッシュの活用は重要



shuffleはなぜsortベースか

- hash mapによる分割・マージも可能
- しかしhash mapを現代的HW上で大規模データに使用すると性能が芳しくない実験結果を複数観察している
- 例：集約処理を含むshuffleの比較 (sortベース vs hash mapベース)
  - 10億レコード
  - 100パーティション
  - hash mapは hopscotchを使用



その他

他にも高性能処理のために様々な機能を検討中

- shuffleだけに頼らないためのbroadcastによるjoinの仕組み
- shuffle処理の並列化やマルチノードで分散処理
- 一連の関係演算子群をLLVM等の現代的なコンパイラ機構を用いてコード生成
- 複数SQLの処理へ適切にリソース配分を行う並列スケジューリング
- ストアドプロシージャの実行基盤としても利用予定

まとめ

- Tsurugiではメニーコア・大規模メモリ等の現代的なハードウェアを効率的に活用し、SQL処理を高速並列化するための実行エンジンを開発中です