【SIGMOD/VLDB/ICDE20xx勉強会】

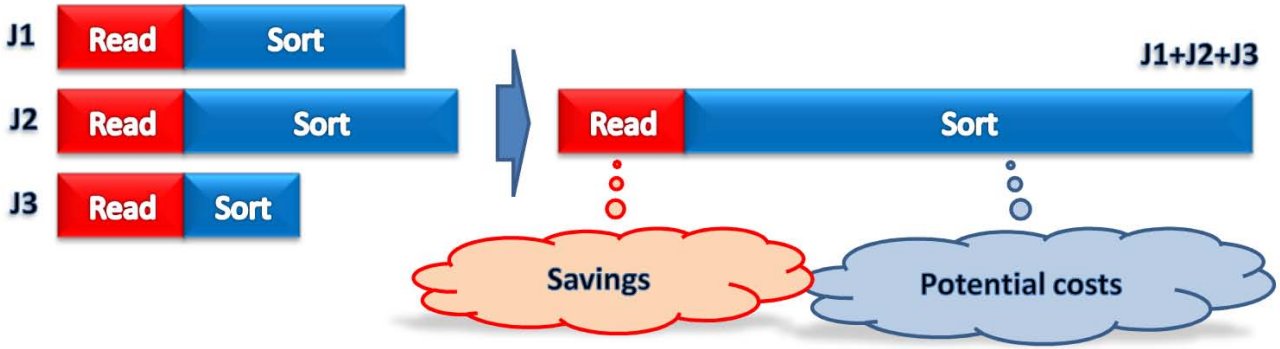
Session 15: **Cloud Computing**

担当: 鬼塚　真（NTT サイバースペース研）

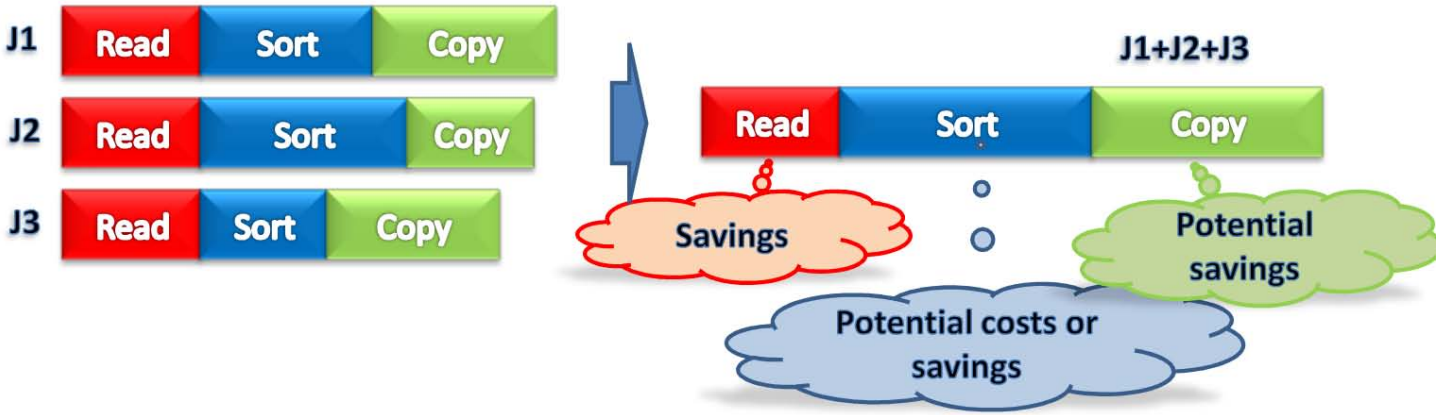
MRShare: Sharing Across Multiple Queries in MapReduce

- Tomasz Nykiel (University of Toronto, Canada), Michalis Potamias (Boston University, United States of America), Chaitanya Mishra (Facebook, United States of America), George Kollios (Boston University, United States of America), Nick Koudas (University of Toronto, Canada)
- http://www.slideshare.net/potamias/mr-share-11-sep-2010
- [概要] 複数の MapReduce jobs の最適化．Mapタスクの入出力データを共有化する．出力についてはサイズを見積もって最適に複数クエリをグループ化して統合して，グループ単位で複数クエリを同時処理する．
- [要素技術]グループ化・Mapタスクの入力/出力の共有化を考慮したMapReduce のコストをモデル化，動的計画法により最適化．
- [効果] blogデータを対象に grep (mapper)+ word count (reducer) 処理で評価 (EC2 40仮想マシン，combiner は不使用)．Mapタスクの入力データ共有による性能改善効果 10%前後，出力データ共有の効果 30%～70%(grep の選択率に改善効果は依存する)

MRShare: Sharing Across Multiple Queries in MapReduce



Mapタスク入力の共有化イメージと効果・コスト



Mapタスク出力の共有化イメージと効果・コスト

MRShare: Sharing Across Multiple Queries in MapReduce

[所感]

- RDBMSにおける典型的な問題の1つを MapReduce で解いた技術. ストリーム処理におけるマルチクエリの最適化とも共通性が高い.
- combine を使用しないことで mapタスクの出力結果をパラメータとしているが(技術検証のため), word count のようなケースでは combine を利用しないのは不自然な設定に見えてしまう.

- [今後の課題] 処理レベルでの共有化, 後続 job の動的な共有化 (現状では複数 job を入力して静的に共有化を行っていて, 処理途中からの job 統合はできない)

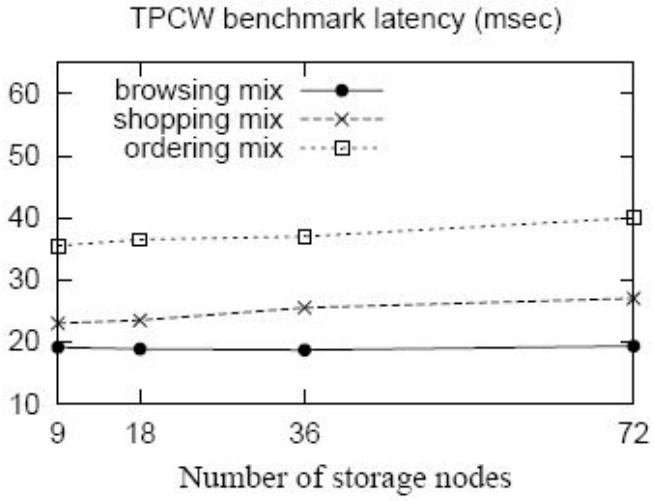
Towards Elastic Transactional Cloud Storage with Range Query Support

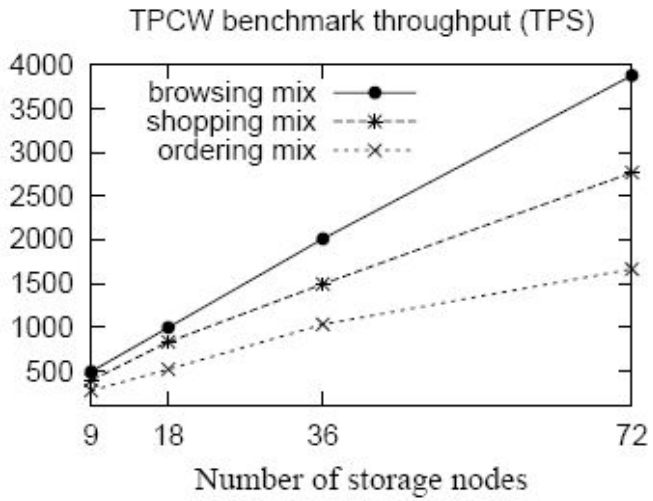
- Hoang Tam Vo (National University of Singapore, Republic of Singapore), Chun Chen (Zhejiang University, People's Republic of China), Beng Chin Ooi (National University of Singapore, Republic of Singapore)
- http://www.comp.nus.edu.sg/~voht/ecstore-vldb.ppt
- [概要] ecStore: 範囲検索可能な P2Pシステム BATON* を拡張し，欠如していたレプリカ管理と，複数レコード・複数ノードにまたがるトランザクション管理機能を導入．

  BATON*: VLDB2005, google scholar で引用数 209件
- [要素技術]
- レプリカ管理は2層(最小replica数保障層と負荷分散層)で，動的な範囲を単位をした統計情報を用いて負荷を制御．
  - 更新: primary replica を更新後に secondary を非同期更新．version を用いて primary の更新操作順を secondary で保障することで 2PCを回避．
  - 参照: quorum を利用して一貫性のレベルをコントロール
- トランザクション管理は entity group などを利用することを前提(1トランザクションは大抵1マシン内で実行)として，複数版を利用した楽観的トランザクションにより分散 snapshot isolation を実現．

Towards Elastic Transactional Cloud Storage with Range Query Support

- 評価実験
  - SNS ベンチマークにより latency/throughput 性能と，負荷の偏りに対する負荷分散の効果を確認.
  - TPC-W による性能評価 (更新におけるデータ局所性が高い)





  - Cassandra (ordered partitioner)との比較 (18ノード)
    - 範囲検索: ecStore は Cassandra より2倍高速(latency)
    - 一致検索: Cassandra の方が 10%程度高速(bloom filter の効果)

Towards Elastic Transactional Cloud Storage with Range Query Support

[所感]

- 多様な技術が集大成されたシステムになっていて，技術レベルが高い．
- トランザクション管理機能があるにも関わらず，Cassandra と同等あるいはより高速な結果もすごい．但し，Cassandra は更新系の方が性能が良くて，ecStore はトランザクション管理のオーバーヘッドがあるため，更新系の詳細な評価がないのは恣意的な感じがする．
- グローバルな transaction 番号(commit number)の割り当て処理はスケーラビリティの阻害要因になるかも

Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing)

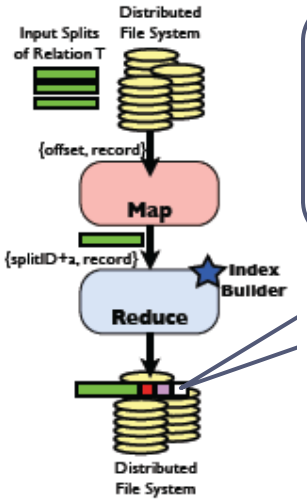
- Jens Dittrich, Jorge-Arnulfo Quiane-Ruiz, Alekh Jindal, Yagiz Kargin, Vinay Setty. Jorg Schad (Saarland University, International Max Planck Research School)

- [概要] MapReduce に 1) index を導入，2) joinが高速になるよう2テーブルを join key で co-partitioning することで高速化を実現. Hadoop 本体には手を加えていないことを主張 (それで Trojan index/join と命名している様子).
- [効果] selection task, join task それぞれにおいて
  - オリジナルの hadoop の 5-8倍, 8-10倍高速
  - さらに HadoopDBよりも 20-30%程度高速

- 最大の主張点:「ワークロードが事前に決まれば，DBMSの技術をMapReduce に適用可能」

Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing)

構築処理



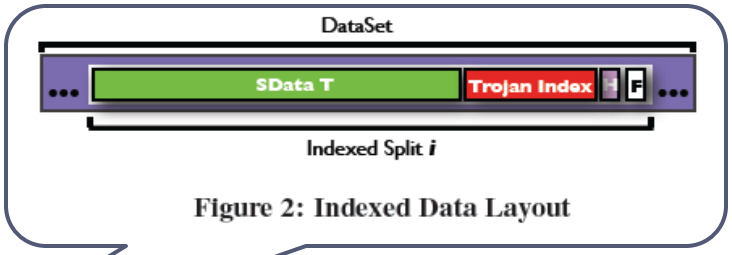
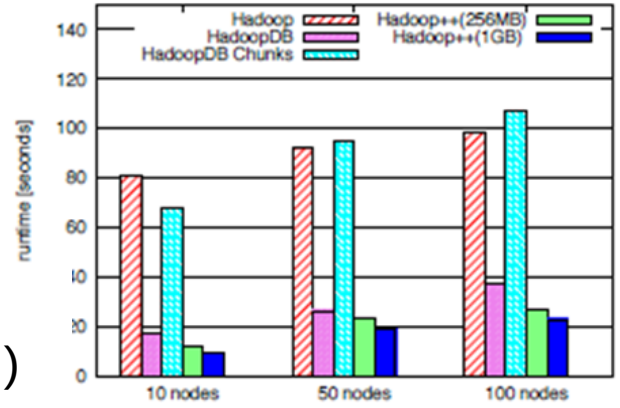
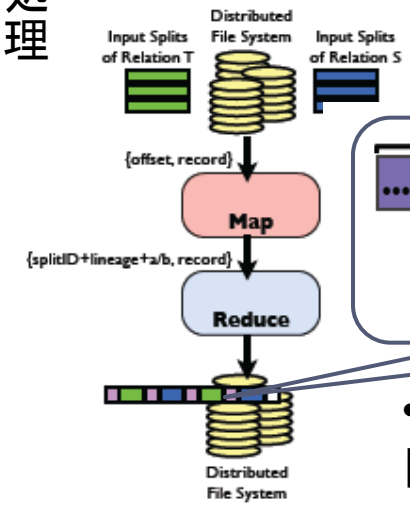


Figure 2: Indexed Data Layout

- データは idx 属性でソートされ格納
- headerは属性範囲を保持(skip判定)
- indexでは(属性値,offset)を保持



(b) Selection Task



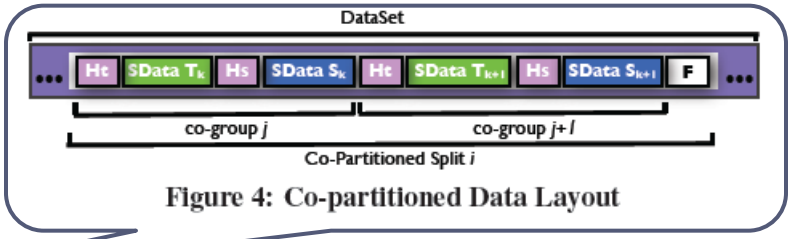
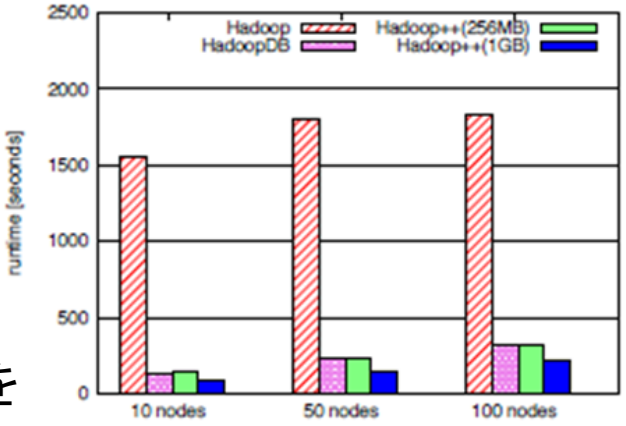


Figure 4: Co-partitioned Data Layout

- 2テーブルの同一 join key レコードを同一 split に格納して，mapper で join



(c) Join Task

Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing)

[所感]

- MapReduce にも高い拡張性がある.
- Join task において, HadoopDB が Trojan Join より 20-30% 程度しか遅くないのは優秀
- RDBMSの技術が徐々にMapReduceに適用される研究動向にある. 当分, RDBMS から MapReduce への拡張と逆方向への拡張が続きそう.