MapReduceの
ネットワーク負荷分析

CQ研究会 2010-11
NTTサービスインテグレーション基盤研究所
森・木村・池田・上山・川原

背景

- 処理するべきデータの超大規模化
  - Web : (analytics, ads)
    - Facebook: 10 TB / day
    - Twitter: 12 TB / day
  - IP Packets : (security, forensics)
    - a 10GE link: 100TB / day
- 並列分散処理による効率化が必要
  1つの有効な solution → MapReduce / GFS

MapReduce / GFS

- MapReduce
  - 大規模データの並列分散処理を実現するためのプログラミングモデル
  - 並列分散：多数の安価なノードを使う
  - バッチ処理に適している
  - Google paper at OSDI 2004

- GFS (Global File System)
  - 分散ストレージシステム
  - 大容量ファイルの一括読み書きに適している
  - Google paper at SOSP 2003

世の中での普及状況

MapReduce /GFS のオープンソース実装である Hadoop のリリースにより、急激に普及

Hadoop World 2010: Keynote, Mike Olson - Hadoop: What's Next?より抜粋

本研究のゴール

- 多数のノードで構成されるMapReduceシステムの性能解析
  - 運用・設計の最適化・省エネ
  - 障害検知

- 一般にData-intensiveな処理を実行する大規模分散システムの性能評価

アプローチ

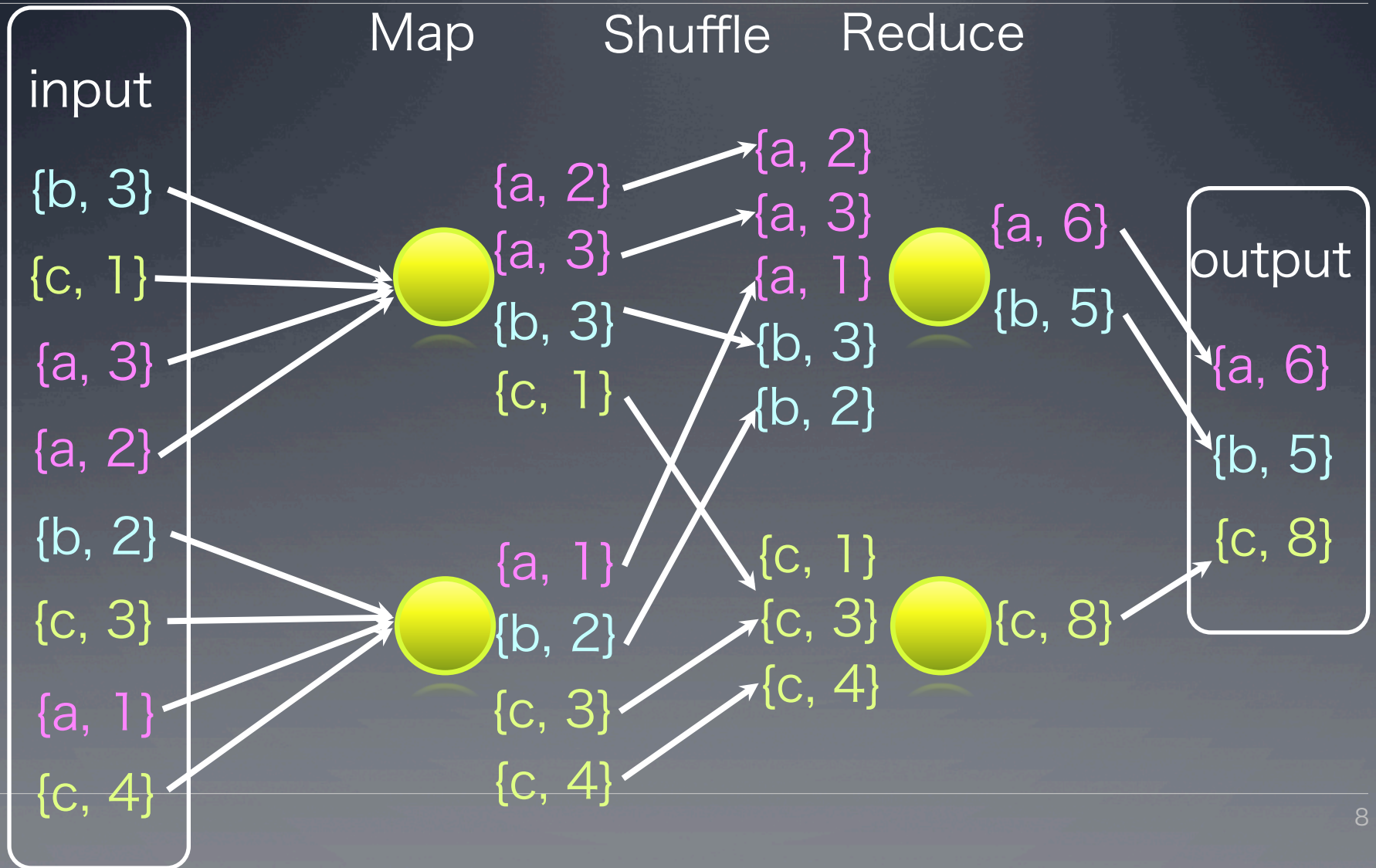
- 広く利用されている MapReduce の実装である Hadoop を用い、実験的な分析をする
  - MapReduce のパラメタを変更

- システムの負荷を計測・分析し、基礎的な統計的特徴を得る（ケーススタディ）
  - 特にタスクとネットワーク負荷に着目
  - ネットワーク負荷を詳細に計測・分析した点が既存研究との差違
  - “TCP Incast” 現象の有無について

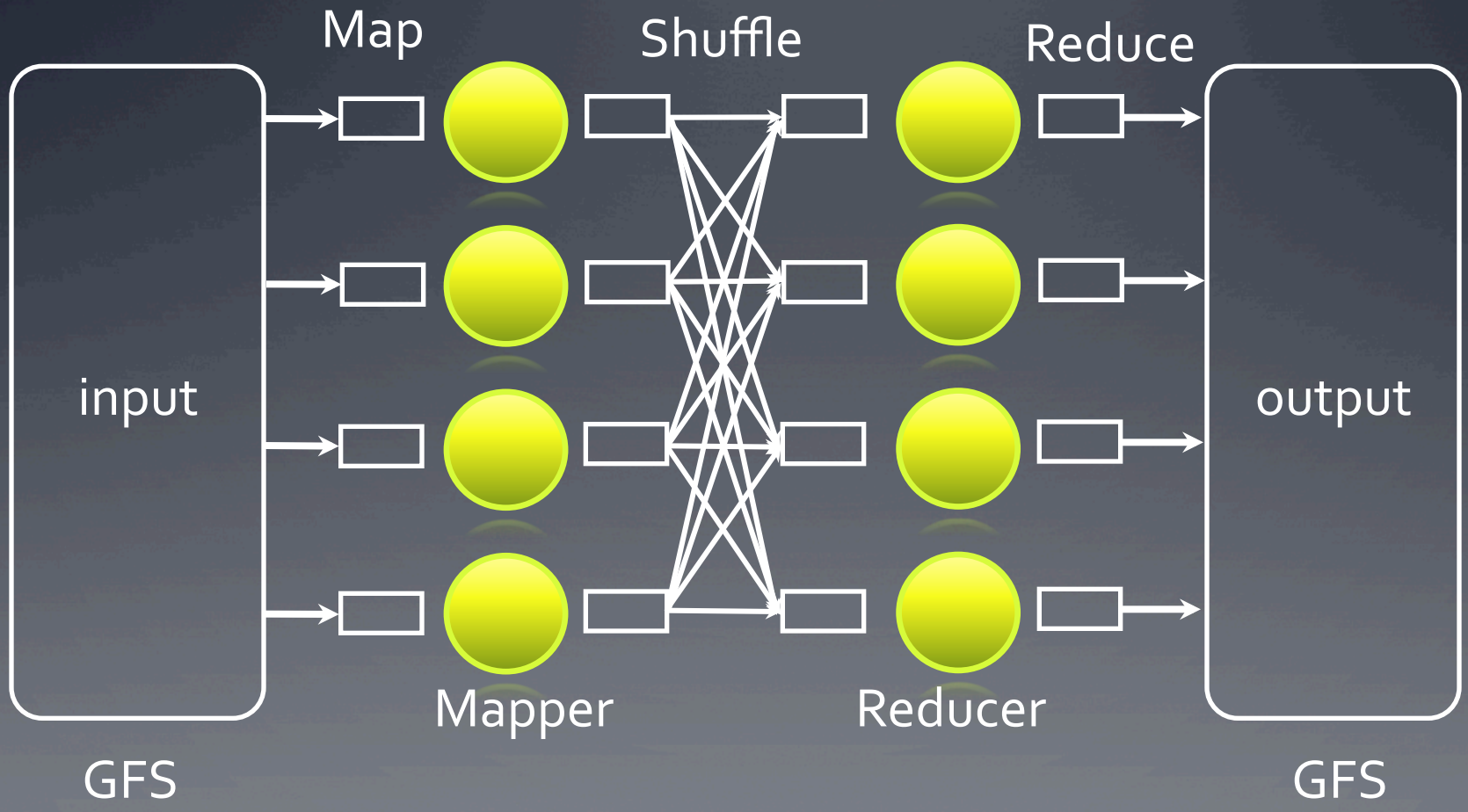
MapReduce

- <key, value> の変換によってデータを処理するシンプルなモデル
- Map:
  - <key, value> → <key', value'>
- Shuffle:
  - 同じ key' を持つレコードを収集・ソートする
- Reduce:
  - 同じ key' を持つレコードの集合 <key'; <value1', value2', ...>>を受け取り、新しい <key'', value''>を出力する

MapReduce


Map
Shuffle
Reduce
input
{b, 3}
{c, 1}
{a, 3}
{a, 2}
{b, 2}
{c, 3}
{a, 1}
{c, 4}
{a, 2}
{a, 3}
{b, 3}
{c, 1}
{a, 1}
{b, 2}
{c, 3}
{c, 4}
{a, 2}
{a, 3}
{a, 1}
{b, 3}
{b, 2}
{c, 1}
{c, 3}
{c, 4}
{a, 6}
{b, 5}
{c, 8}
output
{a, 6}
{b, 5}
{c, 8}

MapReduce System



実験環境

- Hardware
  - 1 master node
  - 10 slave nodes
  - CPU: Intel Xeon X3220 @ 2.40 GHz
  - Memory: DDR2-667, 2GB x 4
  - Storage: SATA 3 Gb/s, 7200 rpm, 32MB cache, 1TB x 3
  - GbE links with GbE Switch
- Software
  - Hadoop (0.20.2)
  - Tcpdump (network measurement)

MapReduce パラメタ

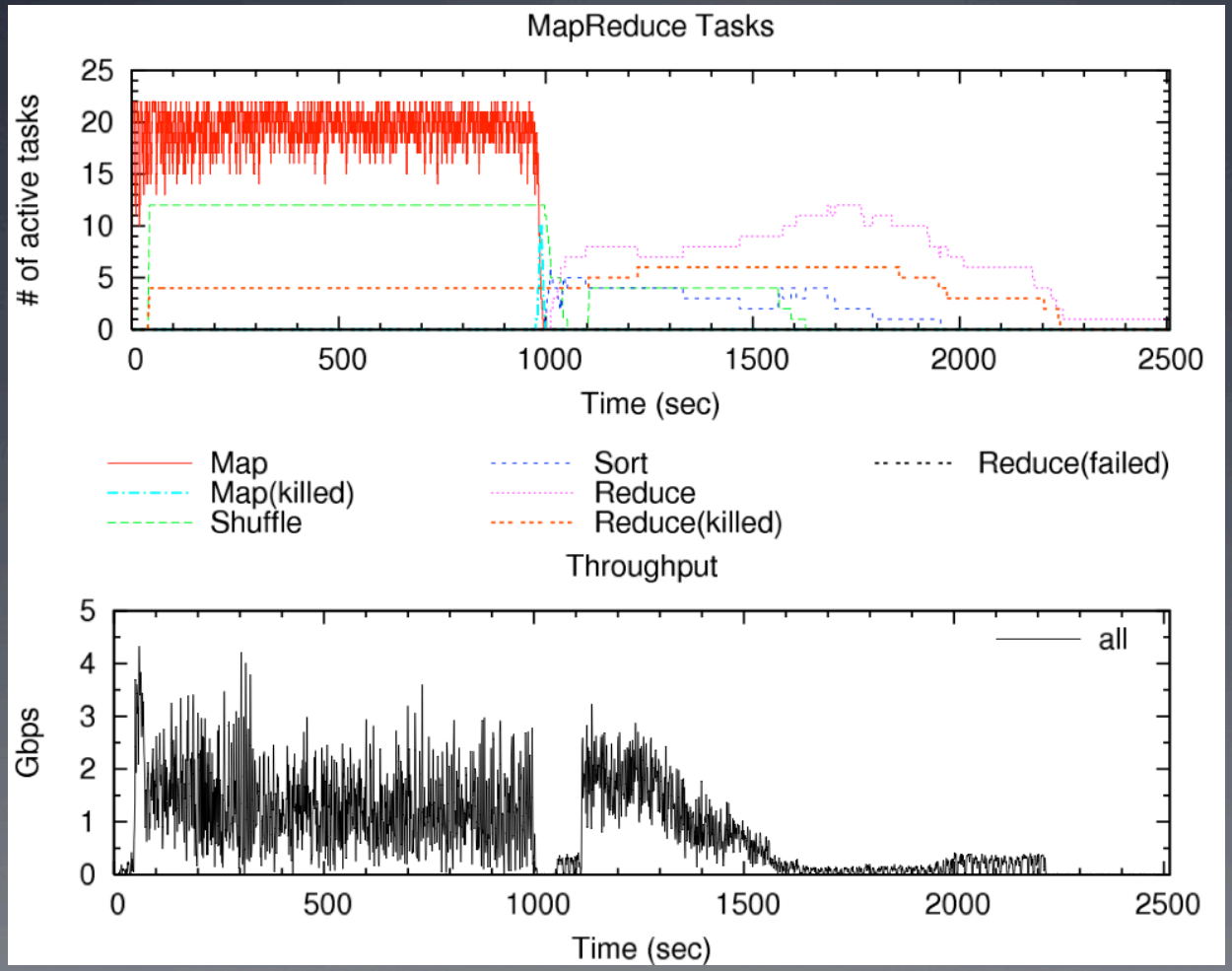
- ノード毎同時 Map タスク数の最大値（スロット）
- ノード毎同時 Reduce タスク数の最大値（スロット）
- あるジョブに対して割り当てる Mapタスクの総数
- あるジョブに対して割り当てる Reduce タスクの総数 (=r)
  - r が小さいと 1台に shuffle traffic が集中
- シャッフル時に reduce task が同時にデータをコピーする数 (=p)
  - p が大きいと Shuffle 時のデータ転送セッション数が増える

MapReduce タスクと
ネットワーク負荷の関係

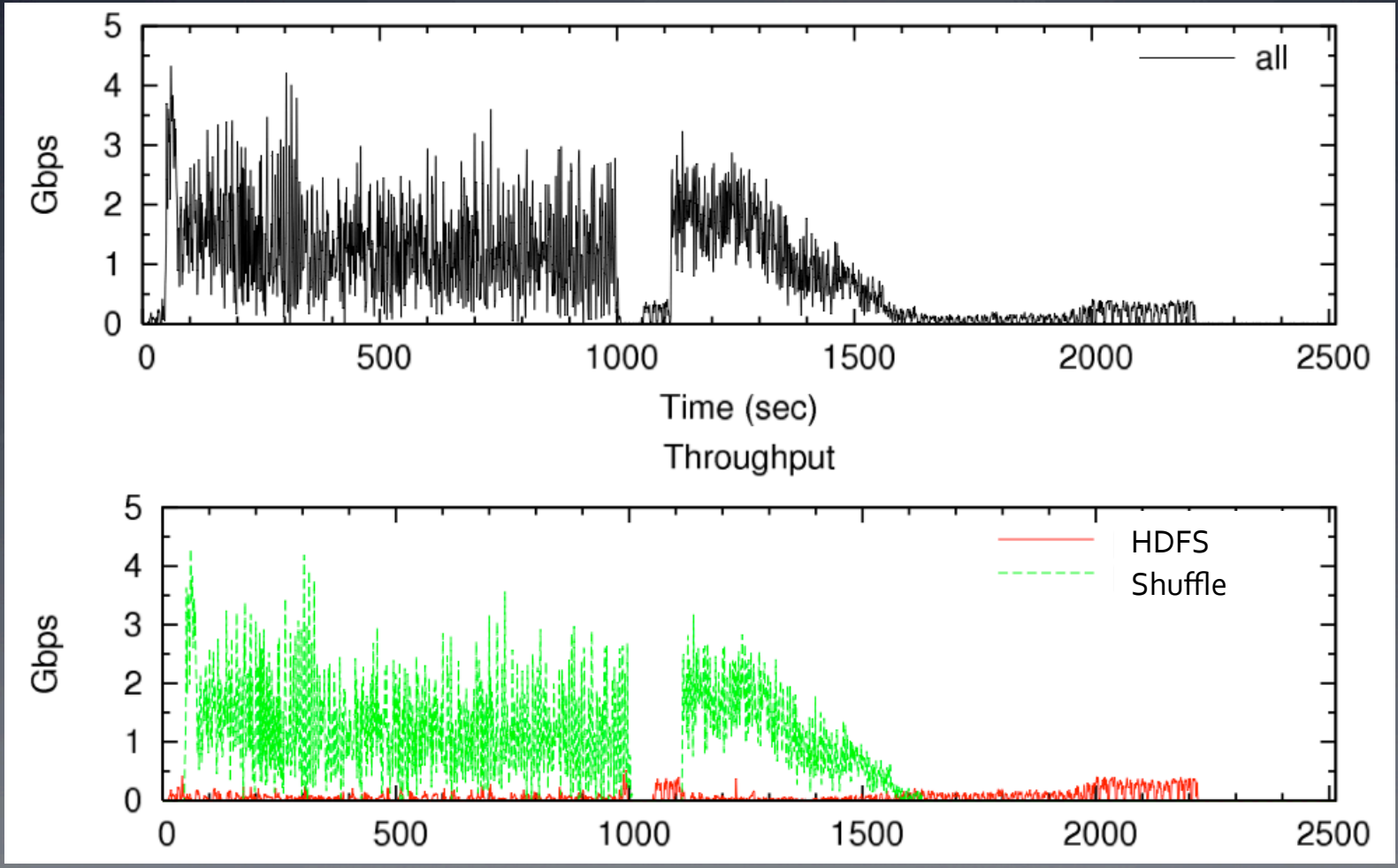
同時タスク数とネットワーク負荷

Task

Network

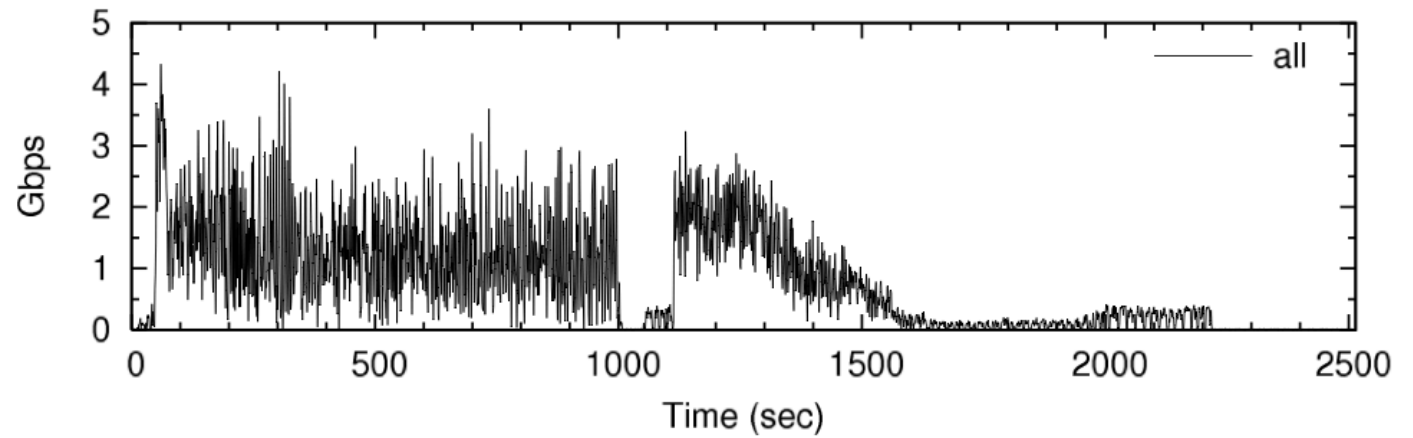


ネットワーク負荷の内訳

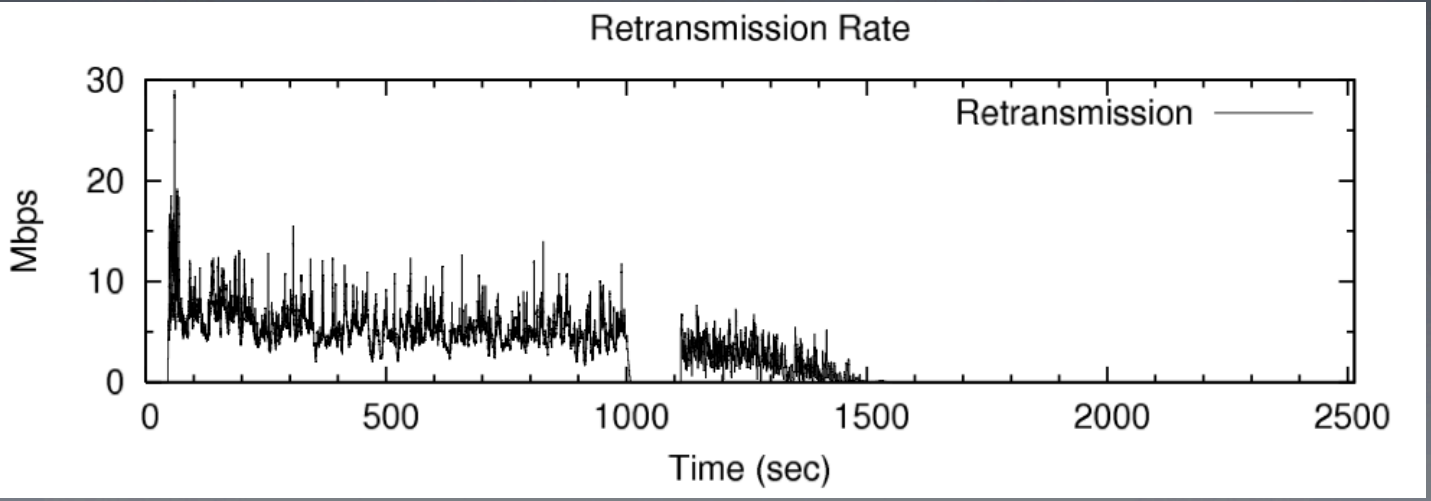


バイト再送率

total



Byte loss



observations

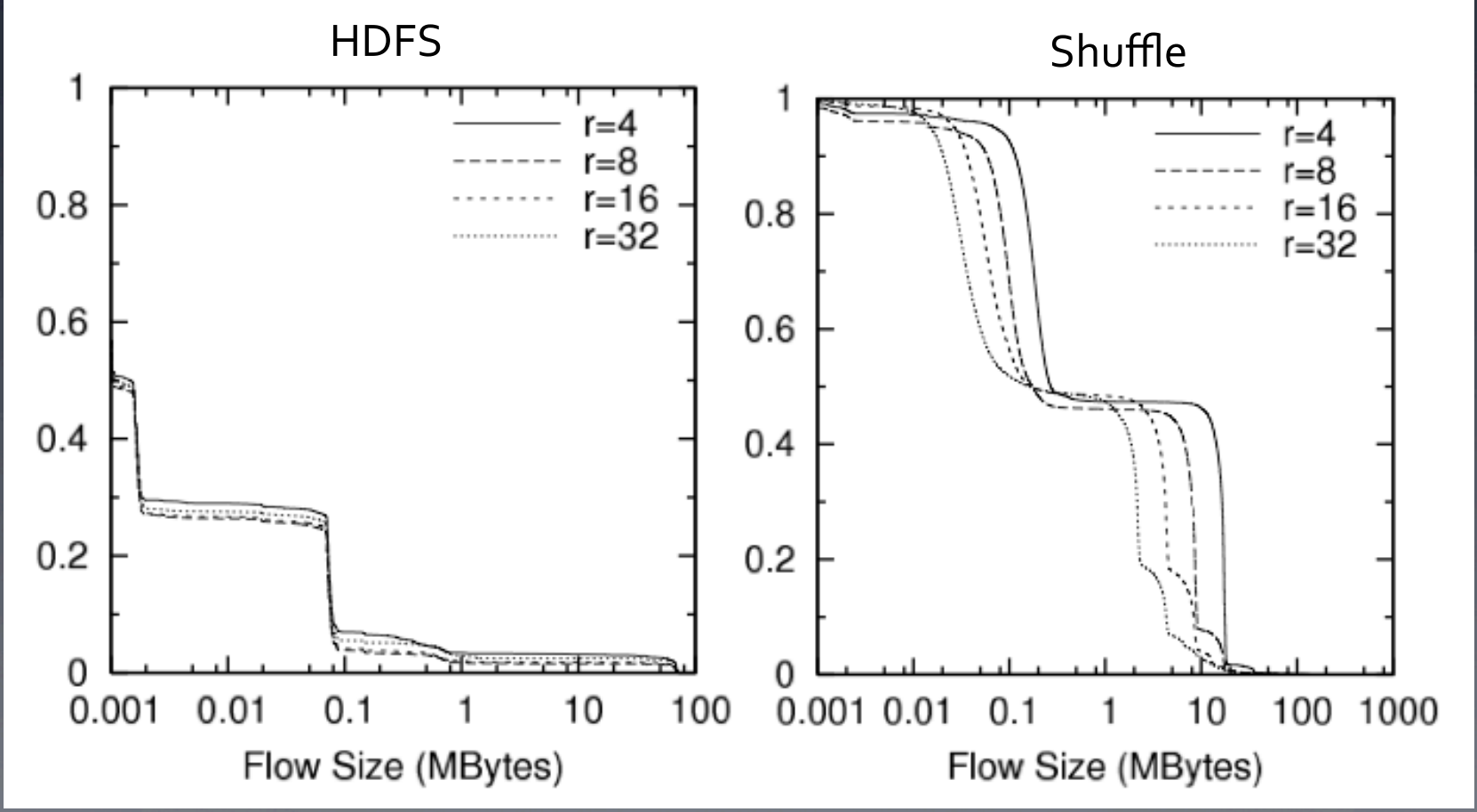
- タスク・フェーズによってネットワーク負荷の特徴が大きく異なる
- 特に Shuffle が　network I/O に大きく影響を与える
  - プログラミング、データレイアウトの工夫で Shuffle 時のリモートコピーはある程度減らせる。
- 今回のパラメタ空間では “TCP Incast”のような明白な輻輳崩壊現象は観測されなかった

フローの分析

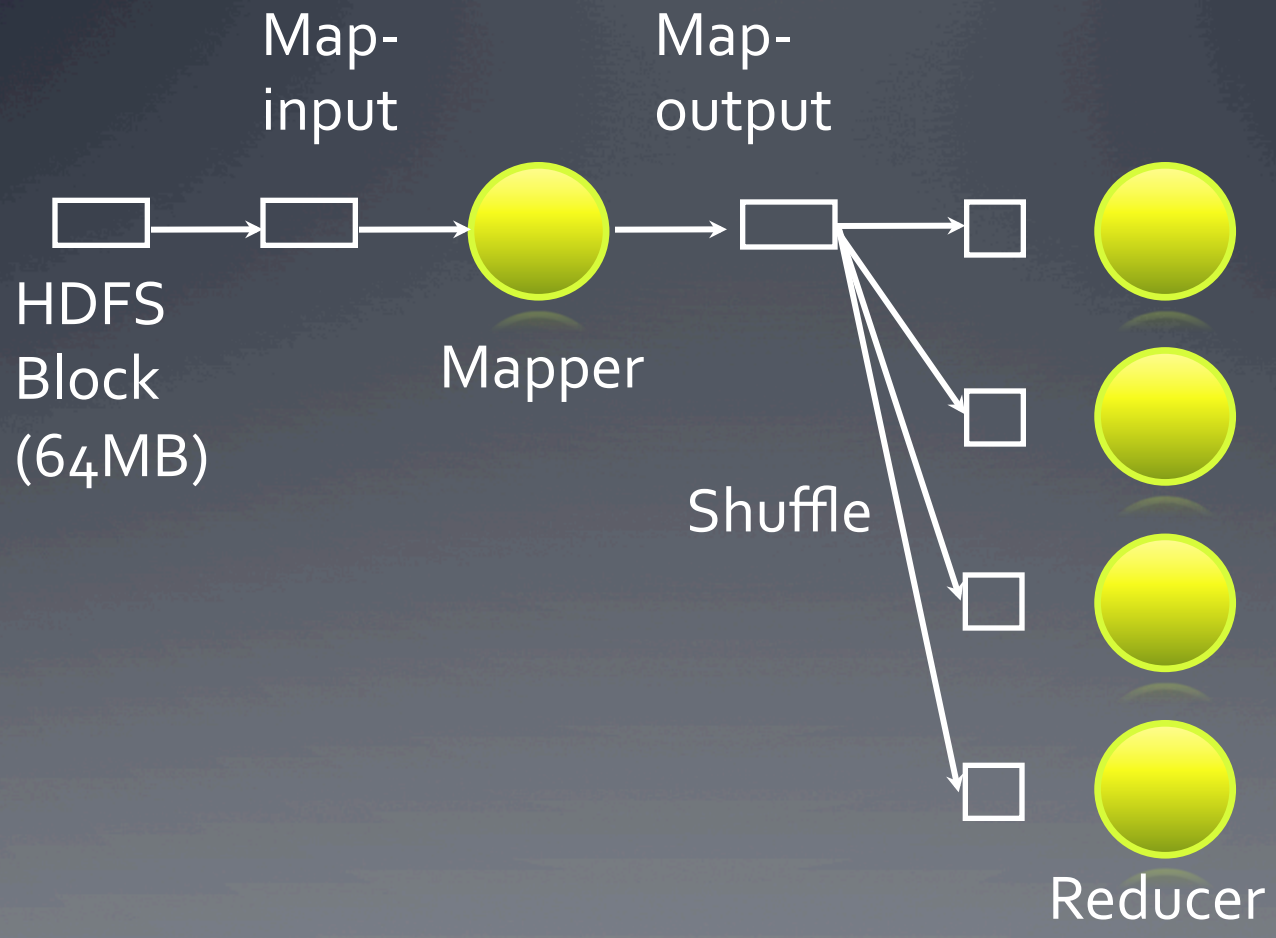
- フロー: Network負荷モデルにおける基本的な単位
  - 1フロー ~ 1 TCP session
    - Flow = {Src IP, Dst IP, Src Port, Dst Port, Protocol}
  - モデル: Aggregating On/Off traffic sources (flows)
  - 計測: NetFlow, sflow
  - 制御: OpenFlow Switch

- 各々のタスクが発生するフローの特徴を分析
  - Heart Beat, Request / response → short-lived flows
  - Data copy, HDFS read/write → long-lived flows

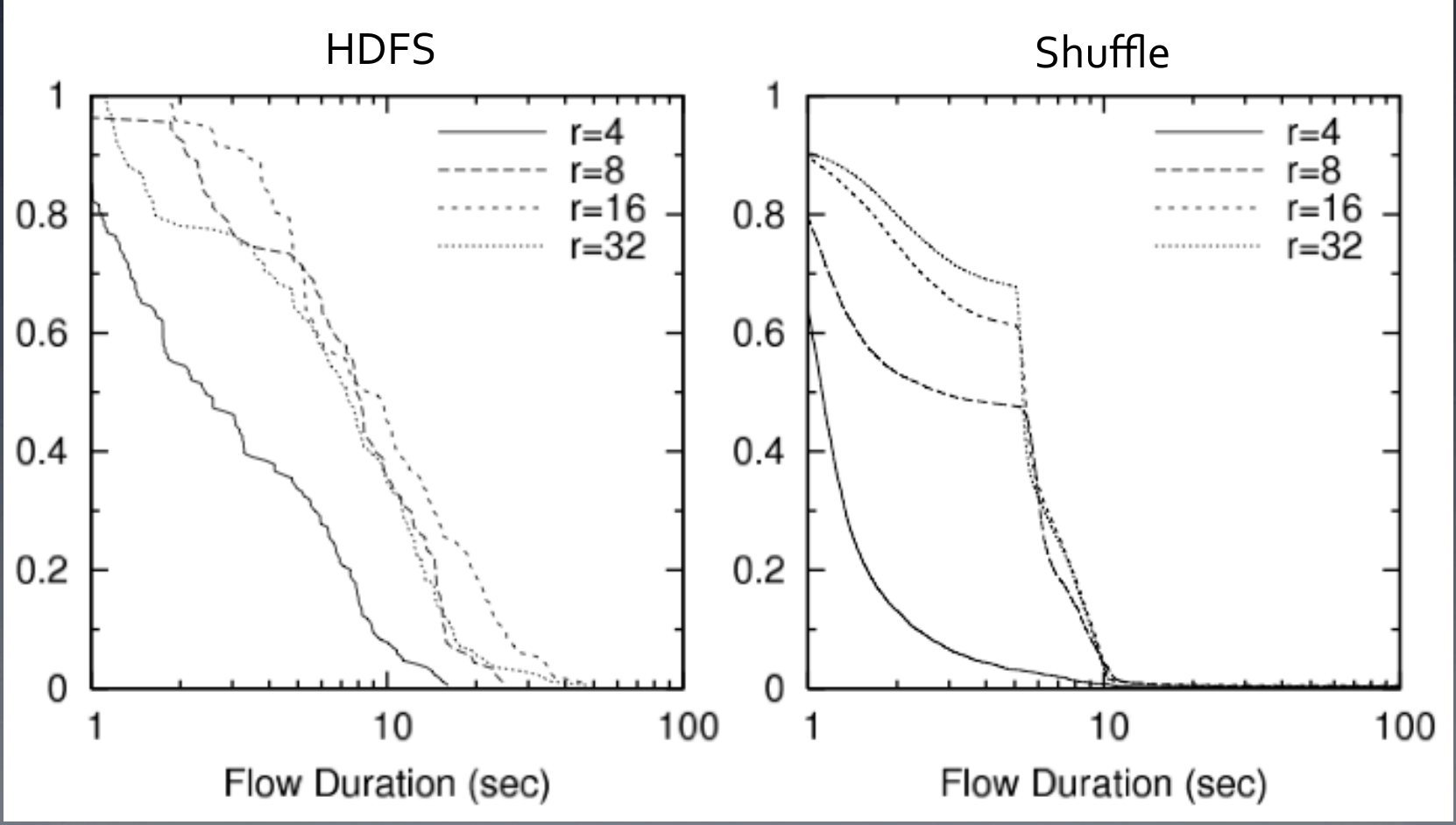
フローサイズ分布


HDFS
Shuffle
r=4
r=8
r=16
r=32
1
0.8
0.6
0.4
0.2
0
0.001
0.01
0.1
1
10
100
1000
Flow Size (MBytes)

Shuffle 時のフローサイズとreducer数



フロー持続時間分布


HDFS
Shuffle
1
0.8
0.6
0.4
0.2
0
1
10
100
r=4
r=8
r=16
r=32
Flow Duration (sec)
Flow Duration (sec)

observations

- MapReduce パラメタ(参加ノード数)によって、フローの特徴が大きく変わる
  - とくに　Shuffle フェーズにおけるデータ転送
  - 分割数 (=r)が大きいと short flow が頻発するようになる

- フロー特性の変化が全体の network I/O およびシステム全体に対してどのように影響を与えるかは今後の課題

まとめ

- Hadoop を利用したMapReduceシステム分析のケーススタディを示した
- ネットワーク負荷に着目した分析アプローチが特徴的
- 今回の実験では “TCP Incast” は観測されなかった
- MapReduceパラメタとフロー特徴の関係を示した

- Future work: 計測で明らかになった特徴を取り入れた性能評価モデルの構築と分析