

インターネット計測とデータ解析 第12回

長 健二郎

2011年1月12日

前回のおさらい

データマイニング

- ▶ パターン抽出
- ▶ クラス分類
- ▶ クラスタリング
 - ▶ 距離と類似度
 - ▶ クラスタリング手法
- ▶ 最終レポートについて

今日のテーマ

スケールする計測と解析

- ▶ 計算量
- ▶ 分散並列処理
- ▶ クラウド技術
- ▶ MapReduce

計測、データ解析とスケーラビリティ

計測手法

- ▶ 測定マシン側の回線容量、データ量、処理能力

データ収集

- ▶ 複数箇所からデータを集める
- ▶ 収集マシン側の回線容量、データ量、処理能力

データ解析

- ▶ 膨大なデータの解析
- ▶ 比較的単純な処理の繰り返し
- ▶ データマイニング手法による複雑な処理
- ▶ データ解析マシン側のデータ量、処理能力、分散処理の場合
は通信能力

計算量 (computational complexity)

アルゴリズムの効率性の評価尺度

- ▶ 時間計算量 (time complexity)
- ▶ 空間計算量 (space complexity)

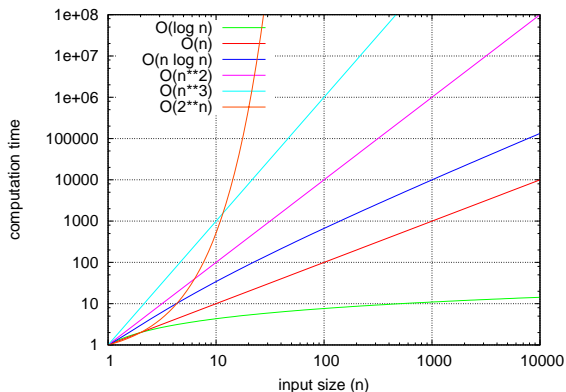
- ▶ 平均計算量
- ▶ 最悪計算量

オーダー表記

- ▶ 入力数 n の増大に対して計算量の増加する割合をその次数のみで表現
 - ▶ 例: $O(n)$, $O(n^2)$, $O(n \log n)$
- ▶ より正確には、「 $f(n)$ はオーダー $g(n)$ 」とは、ある関数 $f(n)$ と関数 $g(n)$ に対して $f(n) = O(g(n)) \Leftrightarrow$ ある定数 C, n_0 が存在して、 $|f(n)| \leq C|g(n)| (\forall n \geq n_0)$

時間計算量

- ▶ 対数時間 (logarithmic time)
- ▶ 多項式時間 (polynomial time)
- ▶ 指数時間 (exponential time)



計算量の例

サーチ

- ▶ リニアサーチ: $O(n)$
- ▶ バイナリサーチ: $O(\log_2 n)$

ソート

- ▶ 選択整列法: $O(n^2)$
- ▶ クイックソート: 平均で $O(n \log_2 n)$ 、最悪は $O(n^2)$

一般に、

- ▶ 全変数を調べる (ループ): $O(n)$
- ▶ バイナリツリー構造など: $O(\log n)$
- ▶ 変数に対する 2 重ループ: $O(n^2)$
- ▶ 変数に対する 3 重ループ: $O(n^3)$
- ▶ 全変数の組合せ (最短経路検索など): $O(c^n)$

分散アルゴリズム

並列アルゴリズム

- ▶ 問題を分割して並列実行
- ▶ 通信コスト、同期問題

分散アルゴリズム

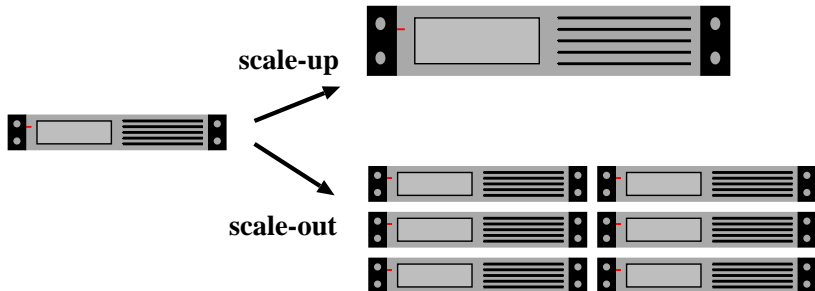
- ▶ 並列アルゴリズムのなかでも、独立したコンピュータ間のメッセージ交換のみによる通信を前提にしたもの
- ▶ コンピュータの故障やメッセージの損失を考慮

メリット

- ▶ スケーラビリティ
 - ▶ しかし、最善でも並列度に対してリニアな向上
- ▶ 耐故障性

スケールアップとスケールアウト

- ▶ スケールアップ
 - ▶ 単一ノードの拡張、強化
 - ▶ 並列処理の問題がない
- ▶ スケールアウト
 - ▶ ノード数を増やすことによる拡張
 - ▶ コスト効率、耐故障性 (安価な量産品を大量に使う)



クラウド技術

クラウド: さまざまな定義がある

▶ 広くは、ネットワークの向うにあるコンピュータ資源
背景

▶ 顧客ニーズ:

▶ 計算資源、管理、サービスのアウトソース

▶ 初期費用が不要、需要予測をしなくていい

▶ それによるコスト削減

▶ プロバイダ: スケールメリット、囲い込み

さまざまなクラウド

- ▶ public/private/hybrid
 - ▶ public cloud: インターネット経由の一般向けサービス
 - ▶ private cloud: 企業内利用向けのサービス
 - ▶ personal cloud, cloud federation
- ▶ サービス形態: SaaS/PaaS/IaaS
 - ▶ SaaS (Software as a Service)
 - ▶ ソフトウェアサービスの提供 (例: Google Apps, Microsoft Online Services)
 - ▶ PaaS (Platform as a Service)
 - ▶ アプリケーション用のプラットフォーム提供 (例: Google App Engine, Microsoft Windows Azure)
 - ▶ IaaS (Infrastructure as a Service)
 - ▶ 仮想化サーバや共有ディスクなどハードウェアやインフラの提供 (例: Amazon EC2, Amazon S3)
- ▶ cloud provider - cloud user (utility computing)
- ▶ SaaS provider - SaaS user (web applications)
- ▶ PaaS は、SaaS に third party を巻き込む仕組み
- ▶ スケールアウトクラウド / サーバークラウド

キーテクノロジー

- ▶ 仮想化: OS レベル、I/O レベル、ネットワークレベル
- ▶ ユーティリティコンピューティング
- ▶ 省エネ、省電力、低発熱
- ▶ データセンターネットワーク
- ▶ 管理、監視技術
- ▶ 自動スケーリング、ロードバランシング
- ▶ 大規模分散データ処理技術

- ▶ 関連研究分野: ネットワーク、OS、分散システム、データベース、グリッド
 - ▶ 結構商用サービスの方が先を行っている

MapReduce

MapReduce: Google が開発した並列プログラミングモデル

Dean, Jeff and Ghemawat, Sanjay.

MapReduce: Simplified Data Processing on Large Clusters.

OSDI'04. San Francisco, CA. December 2004.

<http://labs.google.com/papers/mapreduce.html>

本スライドの MapReduce 部分はこの資料から作成している

動機: 大規模データ処理

- ▶ 何百、何千台規模の CPU を利用してデータ処理したい
- ▶ ハードウェア構成等を意識せず簡単に利用したい

MapReduce のメリット

- ▶ 並列分散処理の自動化
- ▶ 耐故障性
- ▶ I/O スケジューリング
- ▶ 状態監視

MapReduce プログラミングモデル

Map/Reduce

- ▶ Lisp や他の関数型言語からのアイデア
- ▶ 汎用性: 幅広い応用が可能
- ▶ 分散処理に適している
- ▶ 故障時には再実行可能

Map/Reduce in Lisp

`(map square '(1 2 3 4))` → (1 4 9 16)

`(reduce + '(1 4 9 16))` → 30

Map/Reduce in MapReduce

`map(in_key, in_value) → list(out_key, intermediate_value)`

- ▶ key/value ペアのセットを入力に、別の key/value ペアを生成

`reduce(out_key, list(intermediate_value)) → list(out_value)`

- ▶ `map()` で生成された結果を使い、特定の key に対応する value をマージした結果を返す

例: 文書内の単語の出現頻度のカウント

```
map(String input_key, String input_value):
```

```
  // input_key: document name
  // input_value: document contents
  for each word w in input_value:
    EmitIntermediate(w, "1");
```

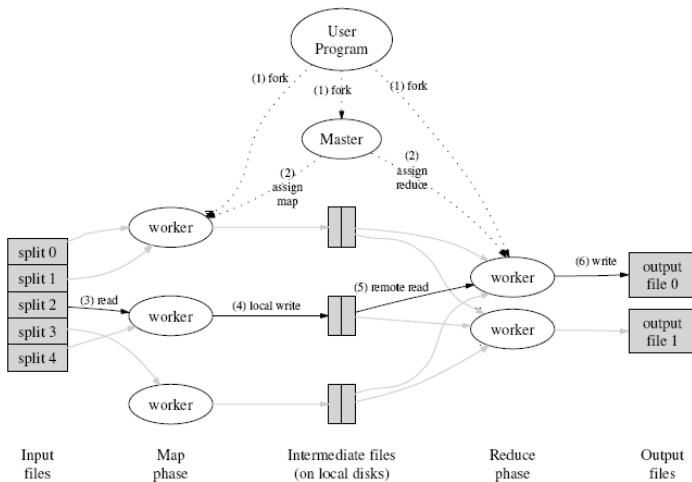
```
reduce(String output_key, Iterator intermediate_values):
```

```
  // output_key: a word
  // output_values: a list of counts
  int result = 0;
  for each v in intermediate_values:
    result += ParseInt(v);
  Emit(AsString(result));
```

その他の応用例

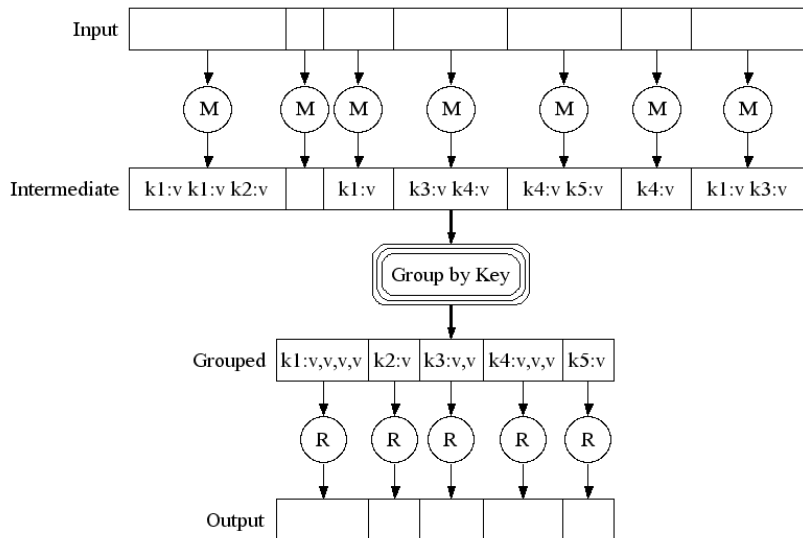
- ▶ 分散 grep
 - ▶ map: 特定パターンにマッチする行を出力
 - ▶ reduce: 何もしない
- ▶ URL アクセス頻度カウント
 - ▶ map: web access log から $\langle URL, 1 \rangle$ を出力
 - ▶ reduce: 同一 URL の回数を加算し $\langle URL, count \rangle$ を生成
- ▶ reverse web-link graph
 - ▶ map: source に含まれる link から、 $\langle target, source \rangle$ を出力
 - ▶ reduce: target を link する source list $\langle target, list(source) \rangle$ を生成
- ▶ 逆インデックス
 - ▶ map: ドキュメントに含まれる単語から $\langle word, docID \rangle$ を出力
 - ▶ reduce: 特定の単語を含むドキュメントリスト $\langle word, list(docID) \rangle$ を生成

MapReduce Execution Overview



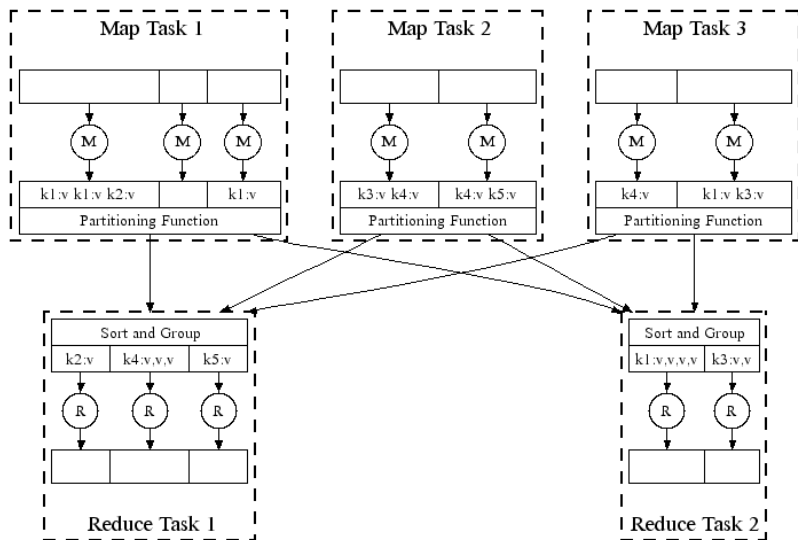
source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce Execution



source: MapReduce: Simplified Data Processing on Large Clusters

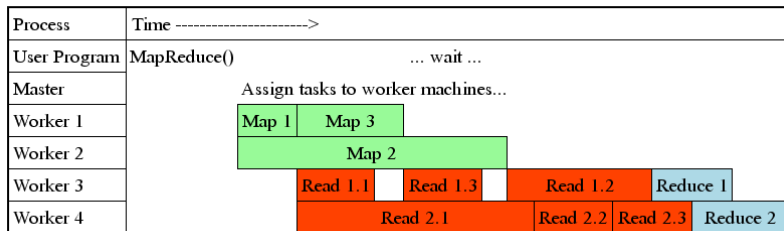
MapReduce Parallel Execution



source: MapReduce: Simplified Data Processing on Large Clusters

Task Granularity and Pipelining

- ▶ タスクは細粒度: Map タスク数 \gg マシン数
 - ▶ 故障復帰時間の低減
 - ▶ map 実行をシャフルしてパイプライン実行
 - ▶ 実行時に動的ロードバランシング可能
- ▶ 典型例: 2,000 台のマシンで、200,000 map/5,000 reduce tasks



source: MapReduce: Simplified Data Processing on Large Clusters

耐故障性

worker の故障

- ▶ 定期的なハートビートで故障を検出
- ▶ 故障したマシンの map task を再割当てして実行
 - ▶ 結果はローカルディスクにあるので終了した task も再実行する
- ▶ 実行中の reduce task を再実行
- ▶ master が task 終了を監視確認

1800 台中 1600 台のマシンが故障しても正常終了した実績

MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

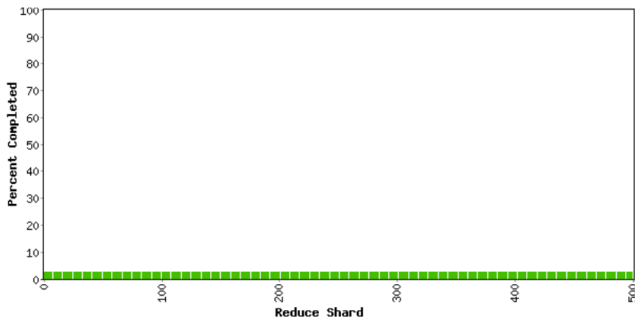
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 00 min 18 sec

323 workers; 0 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	0	323	878934.6	1314.4	717.0
Shuffle	500	0	323	717.0	0.0	0.0
Reduce	500	0	0	0.0	0.0	0.0

Counters

Variable	Minute
Mapped (MB/s)	72.5
Shuffle (MB/s)	0.0
Output (MB/s)	0.0
doc-index-hits	145825686
docs-indexed	506631
dups-in-index-merge	0
mr-operator-calls	508192
mr-operator- ...	506631



source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

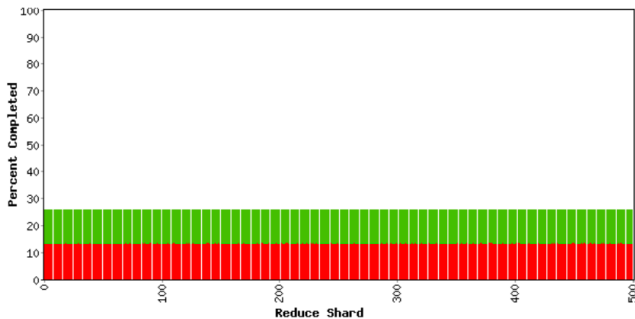
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 05 min 07 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	1857	1707	878934.6	191995.8	113936.6
Shuffle	500	0	500	113936.6	57113.7	57113.7
Reduce	500	0	0	57113.7	0.0	0.0

Counters

Variable	Minute
Mapped (MB/s)	699.1
Shuffle (MB/s)	349.5
Output (MB/s)	0.0
doc-index-hits	5004411944
docs-indexed	17290135
dups-in-index-merge	0
mr-operator-calls	17331371
mr-operator-outputs	17290135



source: MapReduce: Simplified Data Processing on Large Clusters

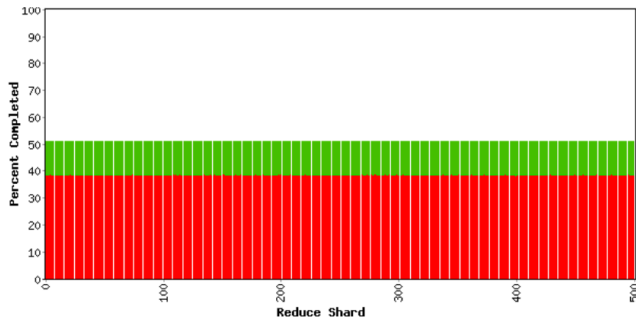
MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 10 min 18 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	5354	1707	878934.6	406020.1	241058.2
Shuffle	500	0	500	241058.2	196362.5	196362.5
Reduce	500	0	0	196362.5	0.0	0.0



Counters

Variable	Minute
Mapped (MB/s)	704.4
Shuffle (MB/s)	371.9
Output (MB/s)	0.0
doc-index-hits	5000364228
docs-indexed	17300709
dups-in-index-merge	0
mr-operator-calls	17342493
mr-operator-outputs	17300709

source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

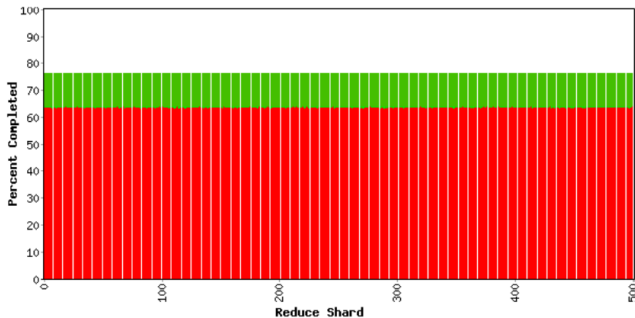
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 15 min 31 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	8841	1707	878934.6	621608.5	369459.8
Shuffle	500	0	500	369459.8	326986.8	326986.8
Reduce	500	0	0	326986.8	0.0	0.0

Counters

Variable	Minute
Mapped (MB/s)	706.5
Shuffle (MB/s)	419.2
Output (MB/s)	0.0
doc-index-hits	4982870667
docs-indexed	17229926
dups-in-index-merge	0
mr-operator-calls	17272056
mr-operator-outouts	17229926



source: MapReduce: Simplified Data Processing on Large Clusters

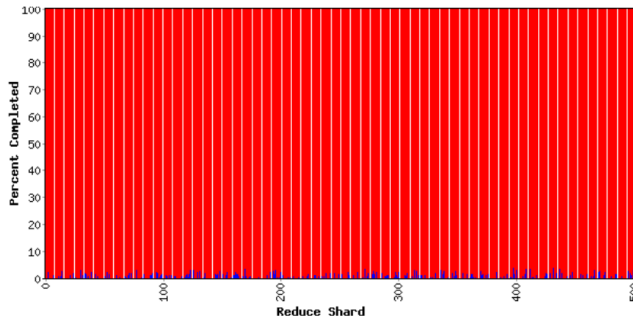
MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 29 min 45 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	195	305	523499.2	523389.6	523389.6
Reduce	500	0	195	523389.6	2685.2	2742.6



Counters

Variable	Minute	
Mapped (MB/s)	0.3	
Shuffle (MB/s)	0.5	
Output (MB/s)	45.7	
doc-index-hits	2313178	105
docs-indexed	7936	
dups-in-index-merge	0	
mr-merge-calls	1954105	
mr-merge-outputs	1954105	

source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

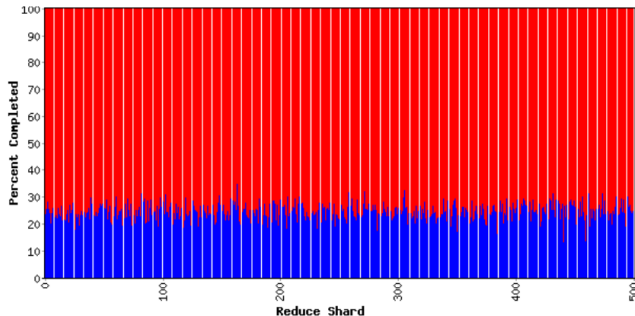
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 31 min 34 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	133837.8	136929.6

Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.1
Output (MB/s)	1238.8
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51738599
mr-merge-outputs	51738599



source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

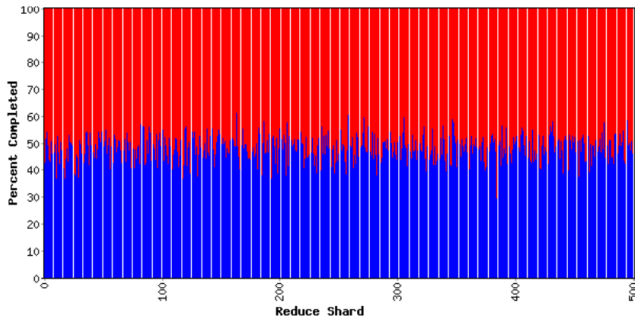
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 33 min 22 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	263283.3	269351.2

Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	1225.1
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51842100
mr-merge-outputs	51842100



source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

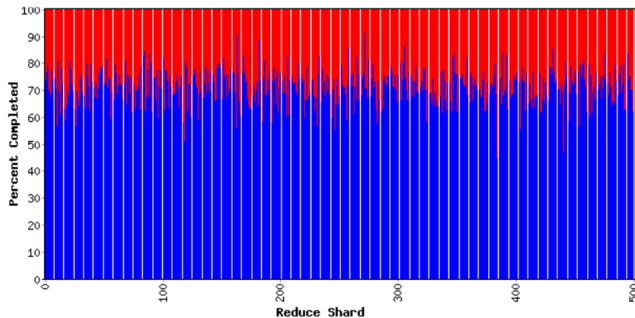
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 35 min 08 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	390447.6	399457.2

Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	1222.0
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51640600
mr-merge-outputs	51640600



source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

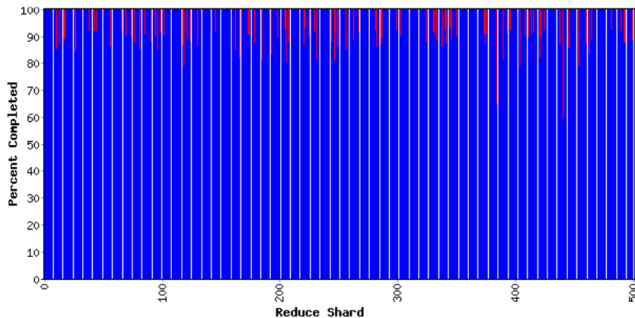
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 37 min 01 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	520468.6	520468.6
Reduce	500	406	94	520468.6	512265.2	514373.3

Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	849.5
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	35083350
mr-merge-outputs	35083350



source: MapReduce: Simplified Data Processing on Large Clusters

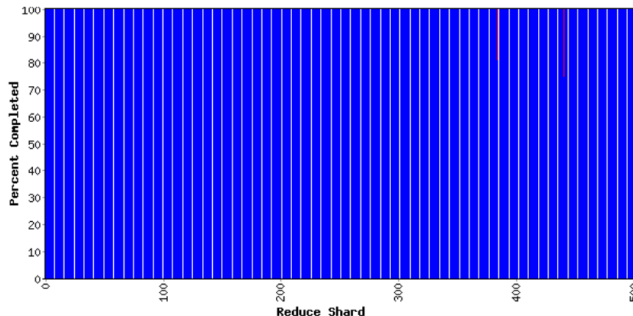
MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 38 min 56 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	519781.8	519781.8
Reduce	500	498	2	519781.8	519394.7	519440.7



Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	9.4
doc-index-hits	0 1056
docs-indexed	0 :
dups-in-index-merge	0
mr-merge-calls	394792 :
mr-merge-outs	394792 :

source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce status

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

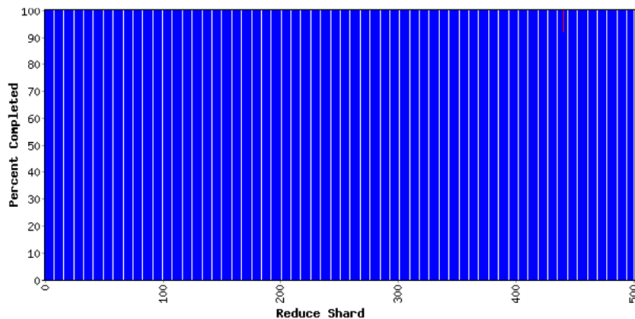
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 40 min 43 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	519774.3	519774.3
Reduce	500	499	1	519774.3	519735.2	519764.0

Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	1.9
doc-index-hits	0 1050
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	73442
mr-merge-outputs	73442



source: MapReduce: Simplified Data Processing on Large Clusters

冗長実行

遅い worker があると終了時間に大きな影響

- ▶ 別ジョブの影響
- ▶ ディスクのソフトエラー
- ▶ その他の要因: CPU cache が disable されていたケースも!

解決策: 全体処理終了近くで、backup tasks を起動

- ▶ 早く終了した task の結果を採用

ジョブ終了時間の大幅短縮に成功

ローカリティの最適化

Master のスケジューリングポリシー

- ▶ GFS に入力ファイルブロックの複製の位置を問い合わせ
- ▶ 入力を 64MB 単位 (GFS block size) に分割
- ▶ 入力データの複製があるマシンまたはラックに map task を割当ててる

効果: 何千台のマシンがローカルなディスクから入力を読み込み

- ▶ さもなければ、ラックのスイッチがボトルネックになる

不良レコードのスキップ

Map/Reduce 機能が時に特定のレコードの処理でクラッシュすることがある

- ▶ 原因はバグ: デバッグして問題解決するのが最善だが、そう出来ない場合も多い
- ▶ Segmentation Fault の発生時
 - ▶ シグナルハンドラからマスターに UDP パケットを送信
 - ▶ 処理中のレコード番号を通知
- ▶ Master は同じレコードの不良が 2 回起こると
 - ▶ 次の worker にそのレコードをスキップするよう指示

効果: サードパーティ製ライブラリのバグ回避

その他の最適化

- ▶ 各 reduce パーティション内でソートされた順序を保証
- ▶ 中間データの圧縮
- ▶ Combiner: 冗長な結果を集約してネットワーク使用量削減
- ▶ デバッグやテスト用のローカル実行環境
- ▶ ユーザ定義のカウンタが利用可能

性能評価

1800 台のマシクラスタを使った性能評価を実施

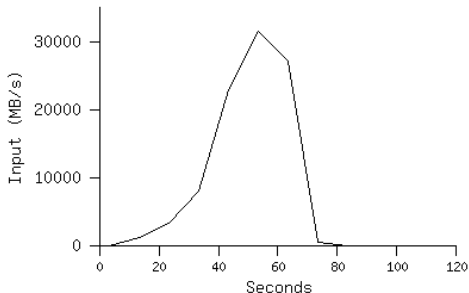
- ▶ 4GB of memory
- ▶ Dual-processor 2GHz Xeons with Hyperthreading
- ▶ Dual 160GB IDE disks
- ▶ Gigabit Ethernet per machine
- ▶ Bisection bandwidth approximately 100Gbps

2 種類のベンチマーク

- ▶ MR_Grep: 10^{10} 100B レコードをスキャンして特定のパターンを抜き出す
- ▶ MR_Sort: 10^{10} 100B レコードをソート (TeraSort ベンチマークのモデルを利用)

MR_Grep

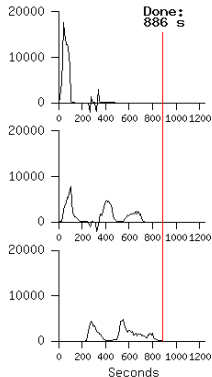
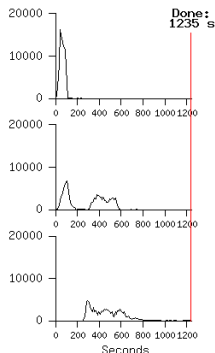
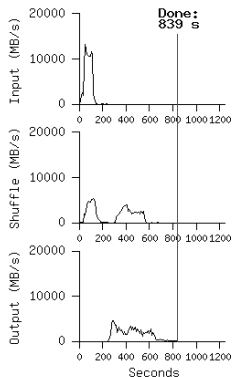
- ▶ ローカルティ最適化効果
 - ▶ 1800 台のマシンが 1TB のデータを最大 31GB/s で読み込み
 - ▶ さもなければ、ラックスイッチがボトルネックで 10GB/s 程度
- ▶ 短いジョブでは始動時のオーバーヘッドが大きい



source: MapReduce: Simplified Data Processing on Large Clusters

MR_Sort

- ▶ backup task による完了時間の大幅短縮
- ▶ 耐故障性の実現



source: MapReduce: Simplified Data Processing on Large Clusters

MapReduce まとめ

- ▶ MapReduce: 並列分散処理の抽象化モデル
- ▶ 大規模データ処理を大幅に簡略化
- ▶ 使い易い、楽しい
 - ▶ 並列処理部分の詳細をシステムにまかせ問題解決に専念できる
- ▶ Google 内部で検索インデックス作成をはじめさまざまな応用

補足

- ▶ Google の MapReduce 実装は非公開
- ▶ Hadoop: Apache プロジェクトのオープンソース MapReduce 実装

まとめ

スケールする計測と解析

- ▶ 計算量
- ▶ 分散並列処理
- ▶ クラウド技術
- ▶ MapReduce

次回予定

第 13 回 まとめ (1/19)

- ▶ これまでのまとめと今後の展望
- ▶ WIDE プロジェクトの計測研究