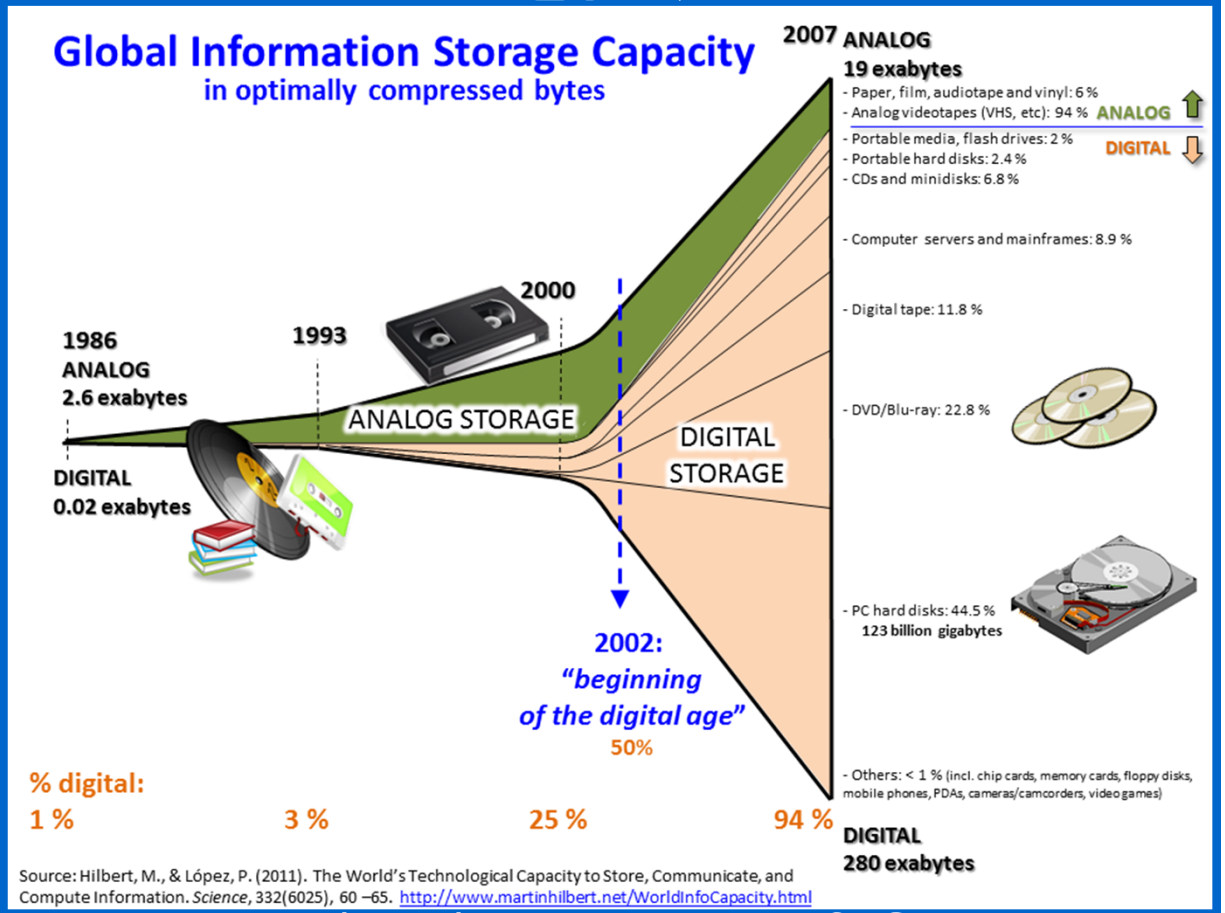
データベースの将来：ビッグデータへの対応

- ビッグデータとは？
  - ビッグデータとは、典型的なデータベースソフトウェアが管理できる能力を超えたサイズのデータを指す
  - 具体的なサイズとして利用の形態に依存するが、数十テラバイトから数ペタバイトの範囲に及ぶとされている（テラ＝10^{12}、ペタ＝10^{15}）



http://en.wikipedia.org/wiki/File:Hilbert_InfoGrowth.png

ビッグデータの4V

http://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data

- 容量（Volume）
  - 数テラバイトから数ペタバイトにおよぶ
- 頻度・スピード（Velocity）
  - 高頻度でデータが生成（ICタグやセンサーなど）
  - これらのデータに対してリアルタイムに対応することが必要
- 種類（Variety）
  - データベースシステムが通常扱うデータ（構造化データ）だけでなく、文書テキスト、音声、ビデオ、クリックストリーム、ログファイル等のさまざまな種類の非構造化データが存在
- 正確さ（Veracity）
  - ビジネスの意思決定に使えるように、信頼性を低下させる要因を排除して、信頼できるデータにすることが求められる
    - 信頼性を低下させる要因：データの矛盾、曖昧さによる不確実性、近似値を積み重ねた不正確さなど

ビッグデータの例(1)

- オンラインショッピングサイトやブログサイトにおいて蓄積される購入履歴やエントリー履歴
- ウェブ上の配信サイトで提供される音楽や動画等のマルチメディアデータ
- ソーシャルメディアにおいて参加者が書き込むプロフィールやコメント等のソーシャルメディアデータ

→上のようなデータについては、種類が多様なため**利用インタフェースで工夫が必要**であるが、データベースシステムとしては従来のシステムでもある程度は対応可能

ビッグデータの例(2)

- IoT (Internet of Things、物のインターネット)のデータ
  - 様々な「モノ(物)」がインターネットに接続され(単に繋がるだけではなく、モノがインターネットのように繋がる)、情報交換することにより相互に制御する仕組み
- GPS等において検知される位置等のセンサデータ
- 歩行者や自動車、鉄道、航空機、船舶などの位置・移動データ
- 人工衛星やモニタリング機器から得られる気象観測データ

→上のようなデータでは、時々刻々と場所ごとに動的に変化するなど時間・空間方向に広がりを持つため、単純に格納するだけでは利用が困難であり、データベースシステムのデータの収集方法や格納方法にかなりの工夫が必要

ビッグデータへの取組み

- 2012年3月29日に米国政府がビッグデータ関連の総額2億ドル以上を投じた研究開発イニシアティブの概要を発表
- このイニシアティブでは、米国の政府機関6機関が主導して、巨大なデジタルデータの組織化やそこからの知識抽出等を行うための技術やツールの開発を行うとされている

①巨大な量のデータの収集、保存、運用、分析、共有に必要な中核技術の進歩

②科学技術分野での発見速度の加速や、国家安全保障の強化、教育・学習の変化への当該技術の活用

③ビッグデータ技術の発展・活用に必要な労働人口の拡大

ビッグデータの活用

- ビッグデータの活用については、ウェブサービス分野において多量に生成・収集等されるデータを各種サービスの提供のために活用することを中心に進展
- 代表例：Amazon, Apple, Facebook, Googleなど
- 各社は、利用者の商品・デジタルコンテンツ等の購買履歴や決済情報、コミュニケーションの発信履歴など膨大なデータを蓄積しており、それらのデータを活用しつつサービス革新等を進めることが、各社の競争力につながっている

RDBMSの限界

- RDBMS（リレーショナルデータベース管理システム）は、高頻度に更新されるのは小規模のデータであり、大規模データについては更新は低頻度であるような、トランザクションに最適化されて設計されているため、ビッグデータに基づく応用事例では性能が劣化してしまう
- ビッグデータの格納場所は単一のサーバからネットワークに分散する複数のサーバへと移りつつあるが、RDBMSだとネットワークの通信速度がボトルネックとなる

NoSQLについて

- NoSQLとは、RDBMS以外のデータベース管理システムを指し、リレーショナルデータベースの長い歴史を打破するものとして、広い意味で**リレーショナルデータモデル以外に属するデータベースの発展を促進させようとする活動を指す**
- NoSQLという言葉から、当初は「SQLは不要」という主張という印象があったが、現在は”**Not Only SQL**”(SQLだけでない）と解釈されている

NoSQLデータベースとは？

- RDBMSの限界を克服するため登場したのがNoSQLデータベースである
- NoSQLデータベースでは、分散環境で高い処理性能を発揮することを念頭に置いて設計されている

RDBMSとNoSQLデータベースの更新処理の違い

- RDBMSは**整合性を保証**する
  - 同時実行制御で整合性・独立性を保つため、**直列化可能性（serializability**、逐次に実行したときと同じ結果となること）が必要となるため、多数の更新要求があると待ちが発生する
- NoSQLデータベースは**結果整合性(eventual consistency)を保証**するという考え方に基づく
  - 更新要求はデータベースに**到着した時点で即座に処理**
  - 更新時の整合性は即座に成立することを前提とせず、更新要求が全部処理された後、最終的には整合性がある状態になる（ある時点では更新が反映されていなくて整合性が保たれない可能性がある）という立場に立ち、多数の更新操作を効率的に処理する

（参考）RDBMSのトランザクションが満たすべき性質（**ACID特性**）

- 原子性（atomicity, 不可分性とも呼ばれる）
- 整合性（consistency, 一貫性とも呼ばれるがintegrityと区別するためここでは整合性と呼ぶ）
- 独立性（isolation）
- 永続性（durability）

（参考）整合性

- 整合性(consistency)は、次回で述べる一貫性(integrity)と基本的には同じ性質である
  - 「一貫性」という場合は、データの持つ性質（実世界で成り立っている性質）を基に、**データベースの設計時**に常に成り立つ性質として定義される
  - 「整合性」は、**トランザクションの開始時と終了時に満たされていることが要求される**性質である
  - 逆に言えば、トランザクションの処理中は処理の効率化のために満たさない場合が想定されている
  - **同時実行制御**（複数のトランザクションを並行に実行する制御）では整合性を保つ必要ある

（参考）独立性

- トランザクション中に行われる操作の過程がその**トランザクションの外から隠蔽される**ことを指す
- トランザクションの外部からは、トランザクションの開始時または終了時のデータベースの内容しか参照できない（つまり、トランザクション実行中の途中のデータベースの内容は参照できない）ことを保証する
- **同時実行制御**で整合性を保つための方針として、**直列化可能性（serializability**、逐次に実行したときと同じ結果となること）があるが、これには**独立性が必要**となる。

NoSQLデータベースのシステム例

- キー・バリュー型
  - キーに対して一つの値という単純な構造
- 列指向（カラム指向とも呼ばれる）
  - キーに対して、カラム（名前と値の組み合わせ）の集合
- ドキュメント指向
  - 文書データの格納を指向したフォーマット（現状では多くの場合JSONフォーマット）でデータを記述
- グラフ型
  - データ同士の関係をグラフで表現する
  - データの横断的な検索が可能

NoSQLのデータの例

- キー・バリュー型
  - キーに対して1つの値
  - 値は多くの場合、バイナリオブジェクト（文字列以外に画像や文書も可）
- 列指向
  - キーに対して複数の（列名・値）の組
  - **列名や値のデータ型は行ごとに可変**
  - 列名を指定した範囲検索、集計処理が可能
- RDBMSとの違い
  - キー・バリュー型や列指向ではキーでしか検索できない
  - キー以外で検索するときは、値からキーを検索するための転置インデックスを使用する

キー	値
01	a
02	b
03	a

キー	列1	列2	
01	a	p	
02	b		
03	a	q	r

列1	キー
a	01
a	03
b	02

列2	キー
p	01
q	02
q	03

NoSQLデータベースのシステム例

- キー・バリュー型

  Riak, Redis, okuyamaなど
- 列指向

  HBase, Cassandraなど
- ドキュメント指向

  MongoDB, CouchDBなど
- グラフ型

  Neo4jなど

Riak

- 代表的なキー・バリュー型NoSQLデータベースシステムの一つ（オープンソース版あり）
- データはバケット（SQLの表に対応）ごとに保存される
- キーとして時刻を指定して、センサーからのデータを格納することができる（**IoT向け応用**）
- データは自動的に複数のサーバに複製が取られる
  - 特定のサーバに障害が起きても別のサーバで代替できる
  - データの参照が複数のサーバに分散されるため効率がよいが、更新を反映するのに時間がかかる（結果整合性）
- 利用例：BestBuy社のシステム

HBase

- 列指向NoSQLデータベースの代表的なシステム例の一つである
- リレーショナルデータベースでは扱えないような大規模データの処理を、分散システムによる並列処理で解決することを目指しており、FacebookやTwitterなど大規模なデータを抱える企業で使われている
- 大規模なデータを大量のマシンで並列に処理するための分散計算技術である**MapReduce**をサポートしているのが特徴である

MapReduceとは?

- 大量のテキストデータを解析してデータマイニングを行う方式の一つ
- Map, Shuffle, Reduceの3段階の処理からなる
- データを分散させて大量のデータを効率的に処理できる

MapReduceの処理の流れ

入力文書

I have a pen I have an apple apple pen I have a pen I have a pineapple pineapple pen　©ピコ太郎

1. 単語抽出、Map処理

   <I,1> <have,1> <a,1> <pen,1> <I,1> <have,1>...

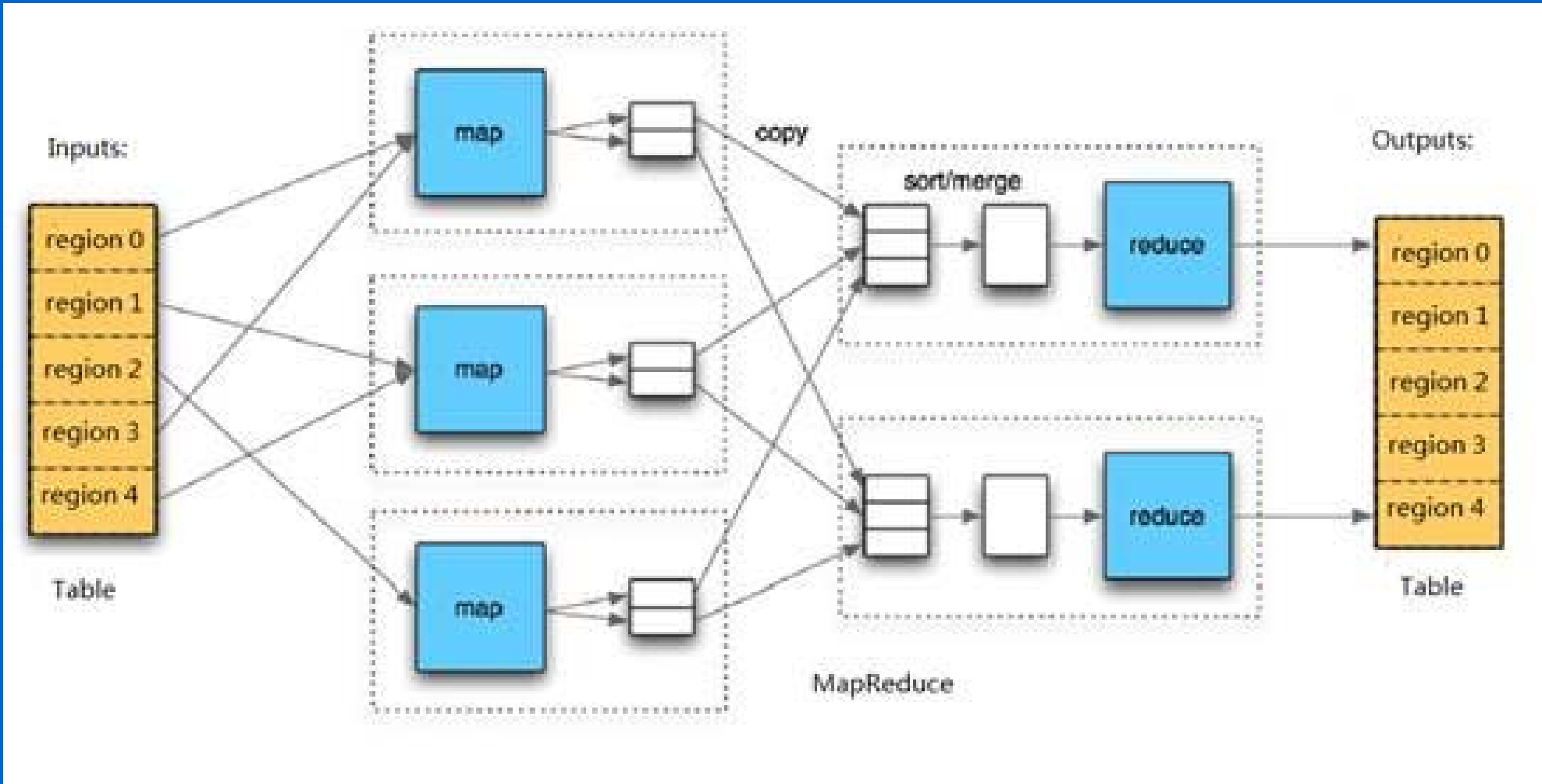
2. Shuffle処理（先頭文字で分割）

   {<a,1><a,1>..}, <have,1><have,1>..}, {<I,1>..}, {<pen,1>..}

3. Reduce処理（単語単位で集計）

   <a,3><an,1><apple,2><have,4><I,4><pen,4>...

MapReduceの実装



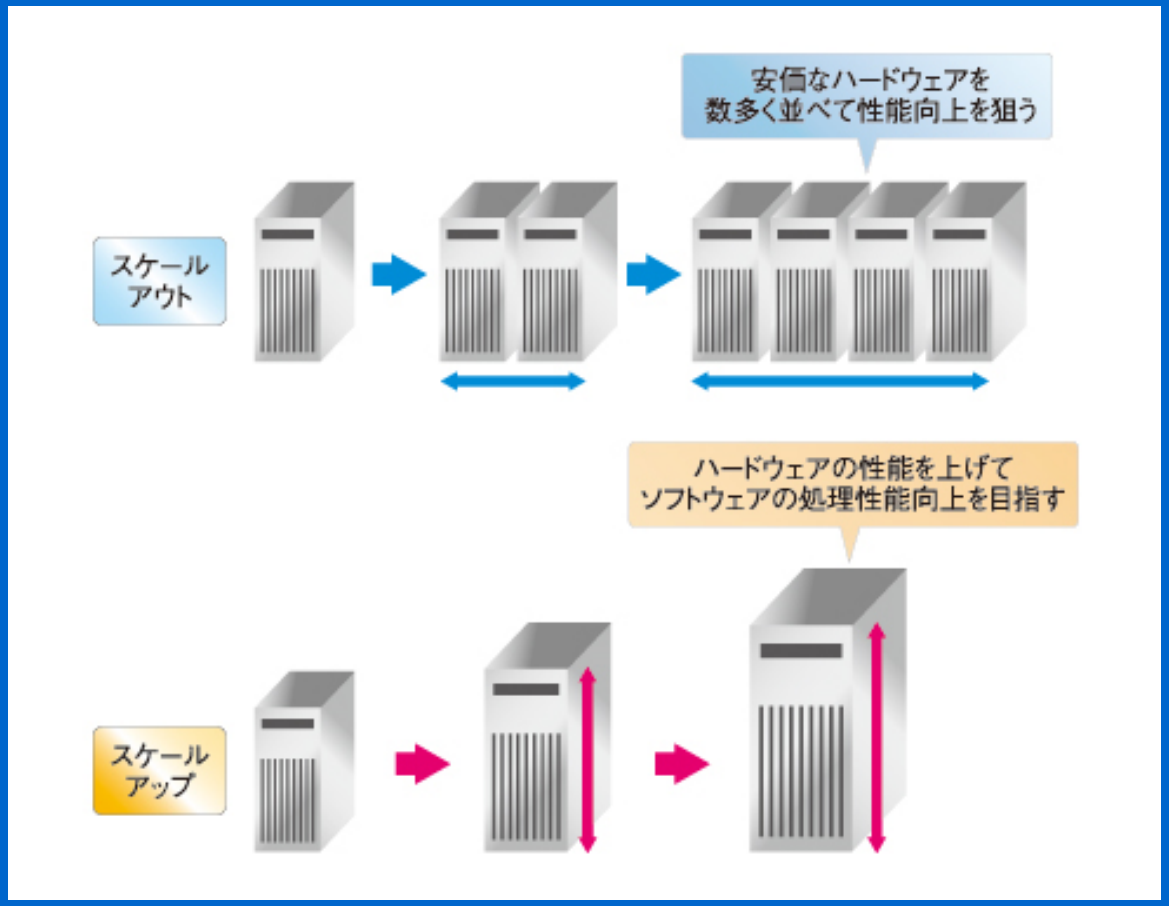
出典　"Turn" Introduction to HBase Technology,
URL: http://www.programering.com/a/MjNwQjNwATQ.html

NoSQLデータベースの特徴（リレーショナルデータベースと比較）

	NoSQL	リレーショナル
大量データへの対応	スケールアウトにより対応	スケールアップにより対応
データアクセス方式	• 限定されたアクセス方式 • アクセス効率を重視（結果整合性のみ保証）	• どの属性についても均一にアクセス • 整合性を重視
データベースのデータ構造	スキーマ定義は不要（スキーマレス）	スキーマ定義が必要

スケールアウトとスケールアップ

- スケールアウト(NoSQL)
  - サーバの数を増やしていくことで処理性能の向上
- スケールアップ(リレーショナル)
  - サーバ単体の性能を向上させて、ソフトウェアの処理性能を上げる



出典 NoSQLはRDBMSに取って代わるものなのか？
http://www.atmarkit.co.jp/ait/articles/1102/24/news098.html