Hadoop導入のご提案

Littel A Total Search Solution for Information Literacy

株式会社リッテル

Hadoop活用のメリット

この様なことでお困りではありませんか？

大量データの高速処理化により処理時間の削減

■大量のデータ処理のスピードアップをはかりたい。

■手持ちのPOSデータ等の分析をしたい。

■ウェブクロールによる大量データを分析したい。

■毎日増え続けるlog解析を行いたい。

サーバー運用のコストダウン

■大量データ処理にかかるコストを低減したい。

■大量のデータ資産があるが処理に時間がかかるのでストレージ内に眠ったままになっている。

■ Web上の情報を集めて何かソリューションを作りたいが処理量が膨大で従来の技術では対応できない。

■現状のシステムではコストがかかりすぎている。

■クラウドコンピューティングに興味があるが、具体的にどうすればいいのかわからない。

こんな形で解決できたら？

- 大量データの高速処理
- 劇的なコストダウン
- 増え続けるストレージに対応
- 即座にサーバーを増やす

増え続けるHadoopを活用する米国企業

ヤフー(Yahoo!)社は、Hadoopを採用し、
　検索エンジンの処理の高速化・安全化・効率化。

　>>>>>>>

また、米・フェイスブック(Faccbook)社は、
アクセス解析・ログ解析にHadoopプラットフォームを活用して、
ユーザーの行動分析を行い、サイト上の広告効果を分析。

　>>>>>>>

さらに、New York Times紙は、Hadoopを活かして、
約1100万本の新聞記事をたった1日でデジタル化することに成功。

ウェブクロールの解析

インターネット上をクロールしたあとのデータの加工として、
大規模データの高速処理にhadoopを利用した事例

■キーワードを抽出する（インデックス化）

どんなキーワードがどんなページに現れるのか

■時刻とキーワードの関連性

キーワードごとにある時刻にブログが何件書かれているか

リレーショナルデータベースと比較して、データベースが大きくなればなるほど、レスポンスに差が出てくる （Hadoopが圧倒的に早い）

ログ解析

単語Aが含まれるページには、単語Aと関係する文章があり、
その単語Aと関連性があるURLを抽出解析する。

例えば、1年分の膨大な量のアクセスログデータからある5分間にアクセスされた
URLのリストを抽出し、キーワードを解析する。

AmazonEC2を活用すれば、
一挙に何十台、何百台のサーバーを一時的に立ち上げ処理することもでき、圧倒的な
パフォーマンスで処理を実行することが可能となる。

リレーショナルデータベースと比較して、大量データの時系列解析でのパフォーマンスが高い （Hadoopが圧倒的に早い）

Hadoopとは

Hadoopプラットフォーム構築の目的

■ グリッドコンピューティングの特性を活かし、低コストでスケーラビリティのあるマイニングプラットフォームを構築する

■ データ構造などの変更に柔軟に対応できるデータベースを稼働させる

■ 多様な分析ニーズに短時間で応じられるように、大量データの並列処理をMapReduceアーキテクチャで行えるようにする

■ 大規模データ高速処理を運用で実現し、新たなプラットフォームとする

Hadoopのプラットフォーム

- Googleが内部で使用インフラの概念を論文発表したことを発端に
  オープンソースファウンデーション「Apache」により、そのクローン化が実現。

- 分散ファイルシステムであるGFS(Google FileSystem)
  のクローンである**HDFS(Hadoop Distributed FileSystem)**

- 分散コンピューティングにおける処理方式であるMapReduce
  のクローンである**MapReduce**

- 以上の二つをあわせた環境である**Hadoop**と大規模分散環境で利用可能なデータベースであるBigTable
  のクローンである**hBase**を合わせて利用することで
  大規模分散環境下における、ファイル・データベース・プログラミング等の効率的な高速処理を行うことができる。

HDFSの機能

ファイルを分散し、耐障害性を高めた形で管理するシステム
基本的な機能はGFSの通りであれば以下の通り

※システムを管理するサーバであるマスタと、それに管理されるサーバ群を考える

1. ファイルを細かな断片に分割する
2. 断片を各サーバにばらまく
3. その際各断片は複数台のサーバにコピーされ、障害時に備える
4. マスタは各断片をどのサーバが所有するかの情報を管理する
5. アクセス時にはマスタから各断片の位置情報を得て、アクセスする

複数台のサーバが断片を保持することにより、負荷分散と障害時の復旧を満たしている

※ファイルの上書きはできない

◆ Map,Reduceの機能

MapとReduceは個々の独立した手続きであり、また連続した手続きである

・Mapとは複数個のデーターつーつに関数を適用する処理
・Reduceとは複数個のデータを取りまとめて関数を適用する処理

この二つをMapReduceの順に常に連続的に行う
分散処理のためにマスタがデータを任意の数に分割し、それを各サーバが処理することで
全体の処理速度を上げるための構造である
Mapによって効率的に小さな断片ごとに処理されたデータがReduceによって統合される

◆ hBaseの機能

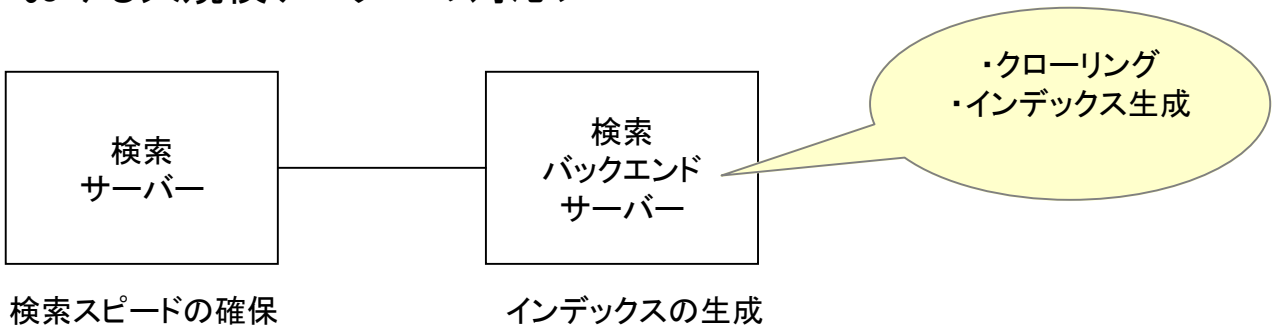
SQLなどに代表されるデータベースの処理を分散システム上で実現する機構
Bigtableのクローン
Bigtable同様にSQLのような高度な機能は持ち合わせていないが
分散されているという意味で高効率

HDFS上で実装されるので、HDFSの機能が必須

並列化処理について

～大規模データ処理に備えて～

＜Googleにおける大規模データへの対応＞



検索スピードが最重要
そのために事前にインデックスを生成している

億単位のキーを格納するための工夫
必要なディスクアクセスを最小限にする
マシン台数が増えたときの並列化のオーバーヘッドを最小限にする
安価なハードでその性能を十分に発揮できるソフトウェア

GFS、MapReduce といった独自の手法で、多数のマシンを使って最適化された並列化が行われている
また、大量データを効率よく処理するために**BigTable**という分散ストレージシステムが使われている

RDBMSとの違い

RDBMS (SQL)の問題点

★スケールアップが困難

- 単一プロセスで動作するため、規模が大きくなるほど飛躍的に高価なハードウェアが必要とされる

- 書き込み時に全体に対してロックがかかるため、書き込みの性能が著しく低くなる

- RDBMSサーバと他のマシンとの通信回路がボトルネックになってしまう

★「Impedance mismatch」問題

オブジェクト指向言語(OO)とのミスマッチ
=> 複雑なデータ構造が扱いにくい

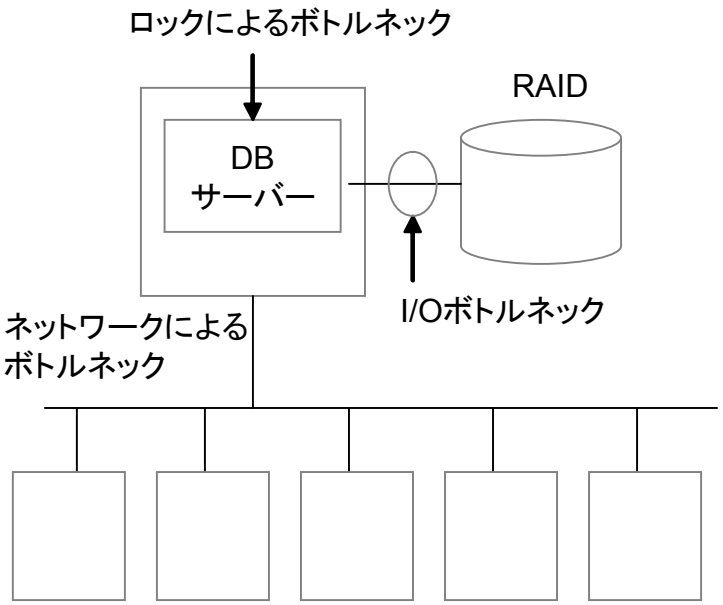
- データのカプセル化
OOではオブジェクト内のデータが隠蔽されるのに対して、RDBMSではそのようなことはできない

- データタイプの違い
OOではポインタ(レファレンス)でのデータ格納が可能だが、RDBMSではできない

- フィールド定義の違い
OOでは比較的簡単にフィールドの数を増やせるのに対して、RDBMSでは大がかりな変更が必要

RDBMS (SQL)とMapReduceの比較

RDBMS

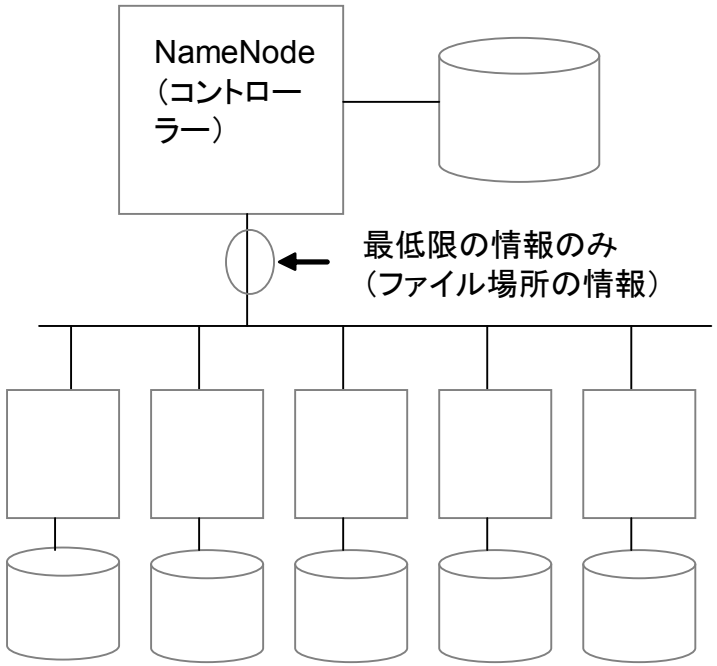


台数が増えるほどボトルネックが大きくなる

解消するには、
・回線を太くする
・CPUクロックを上げる
・I/O性能をアップする

コストが
高くつく

MapReduce



ボトルネックが生じにくい構造

高価なハードウェアを必要としない