



オープンソースの MapReduce/分散ストレージ実装

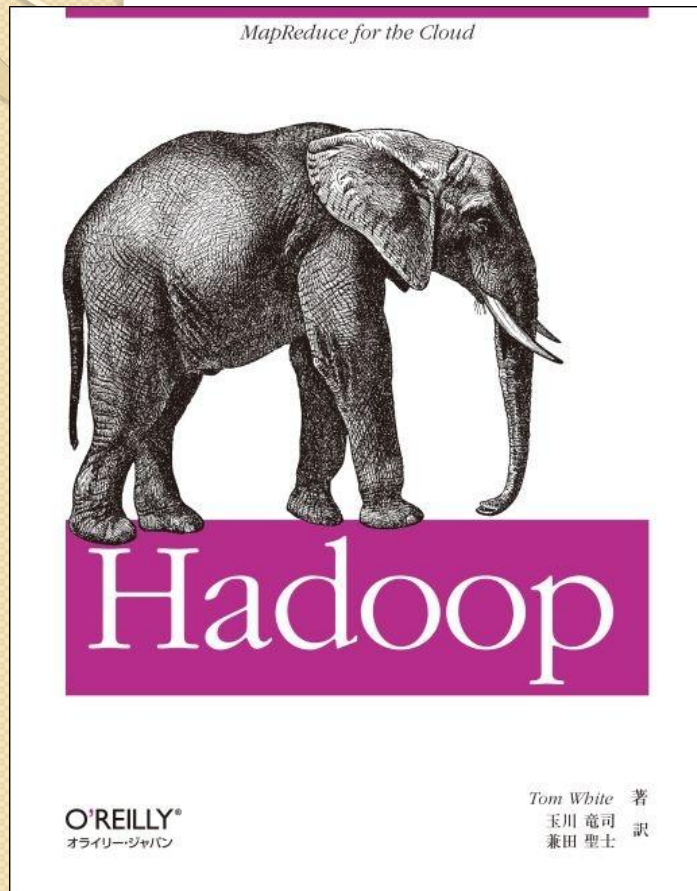
Hadoopの概要と利用動向

玉川竜司 (dragan10@gmail.com)
tamagawa_ryuji@twitter

本日の内容

- 自己紹介
- 悩める成長企業のお話
- クラウドによる解決
- Hadoopの紹介
- サンプルプログラム
- 技術動向
- ケーススタディの紹介
- コミュニティの紹介

自己紹介を少々



- 本職：ソフトウェア開発者
 - かつて勤め先がXMLコンソーシアム会員でした。
- 兼業翻訳者
 - Hadoop（象本）
 - Silverlightで開発するデータ駆動アプリケーション
 - セマンティックWebプログラミング
 - Programming Google App Engine（予定）
 - Programming Windows Azure（予定）
 - Windows Azureプラットフォーム 現場からの報告（<http://slidesha.re/9vo7fV>で公開）

Hadoop本の翻訳をすることになったのは、たまたまです

悩める成長企業のお話

- とあるWeb関連の企業があります
- 小さく始まり、大きく成長していきます
- システム管理者の苦闘が始まります・・・

(I) サービス公開

- ローカルワークステーションから、共有のリモートホスト上で動作するMySQLのインスタンスを使う。
- スキーマは美しく定義した。

(2) サービスの利用が広まる

- 読出負荷がデータベースを圧迫する。
- memcachedを使って、頻繁に行われるクエリをキャッシュする。
- 読み出しは、厳密にはACIDでなくなる。

(3) サービスの利用がさらに広まる

- 書込負荷がデータベースを圧迫する。
- 16コア、128GBのRAM、大量の15,000rpmのハードドライブ群を搭載した強力なサーバーを購入し、MySQLを垂直方向にスケールさせる。
- コストがかかる。

(4) 新しい機能を導入

- クエリが複雑化。
- 今度は結合の多さが問題に。
- 結合を減らすためにデータを非正規化。
- データベース設計者としては断腸の思い・・・

(5) さらにサービス利用の拡大

- サーバが行き詰まる。
- 何もかも遅すぎる。
- サーバーサイドの演算は一切止める。

(6) それでも遅いクエリがある

- 定期的にマテリアライズをするようになる。
- 事実上、結合処理は一切しなくなる。

(7) 読み込みはともかく、
書込はどんどん遅くなっていく...

- インデックスを次々削除。
- トリガも削除。
- インデックスが無くなる？

じゃあ、
RDBMSを使う意味って
何？

スケールアウトによる解決

- 必要なのは、スケールアップ=マシンのパワーアップではない
- スケールアウト=分散処理こそが必要
- 多くの場合、スケーラビリティのネックになっているのはI/O
(特にディスク)



In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox.

We shouldn't be trying for bigger computers, but for more systems of computers.

—Grace Hopper

分散処理の性格分類

- ボトルネックを大きく分けると
 - CPU依存の処理
 - ディスク（ストレージ）依存の処理
- CPU依存型の典型は
 - SETI@HOME
 - Folding@HOME
 - 比較的小さいデータに対して、莫大な計算を行う
- ディスク（ストレージ）依存の処理の典型は
 - （広義の）ログ解析
 - 計算は比較的単純だが、データの量がとにかく莫大
 - Hadoopはどちらかというところら向き

Hadoopの紹介

- MapReduce/分散ファイルシステムのオープンソース実装
- Apache Foundationのプロジェクト
- コアはMapReduce/HDFS
- 安価なハードを大量に並べて、水平分散でデータ処理を行うためのツールキット
- ローカルの安価なマシンを並べたクラスターや、Amazon EC2/S3上で利用可能
- Hadoop上で動作する各種ツールが増えてきている。分散処理のためのデファクトのフレームワークのようになってきている

Hadoopの構成

青はMapReduceの要素
赤は分散ファイルシステム（HDFS）の要素

ジョブトラッカーは、タスクトラッカーに対してタスクを依頼します。対象となるデータのありかは、ネームノードとやりとりをして知ります。障害発生時のリカバリも管理。

プログラマが書くのはこれだけ！

タスクトラッカーは、割り当てられたタスクを実行し、結果をジョブトラッカーに報告します。

マスターノード

ジョブ
トラッカー

ネームノード

分散処理をするのに必要な諸々は、Hadoopが面倒見てくれます。

- ・ハード障害対応
- ・効率の良いネットワーク運用
- ・ジョブの管理
- ・タスクの管理

スレーブノード

タスク
トラッカー

データノード

タスク

タスク

タスク

ネームノードは、HDFSのディレクトリと、ファイルを構成するブロックの場所を管理します。

タスク
トラッカー

データノード

タスク

タスク

タスク

スレーブノード

タスク
トラッカー

データノード

タスク

タスク

タスク

スレーブノード

タスク
トラッカー

データノード

タスク

タスク

タスク

MapReduce

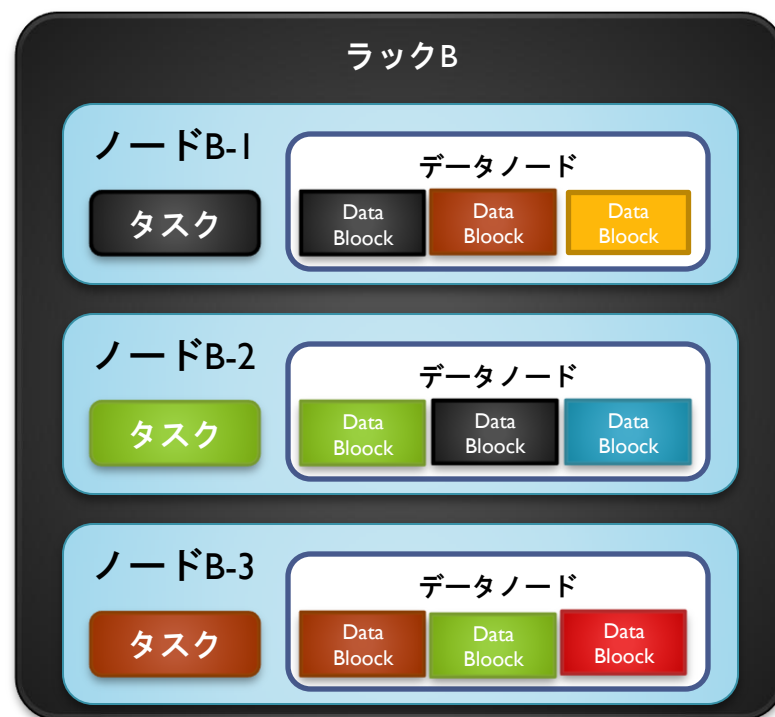
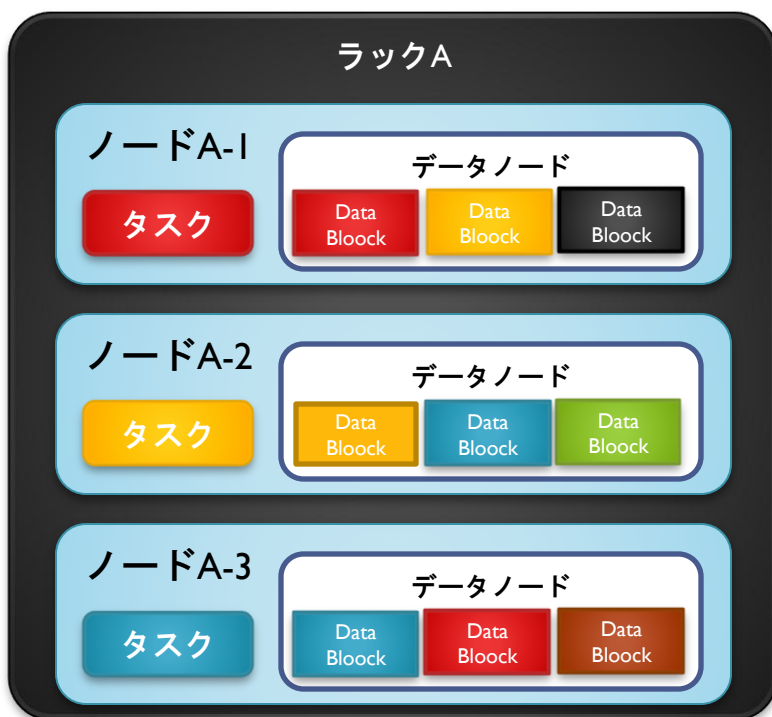
- すべての処理を、MapとReduceで行う
- MapもReduceも、入出力共に、キーと値のペアの集合を扱う
- この「型」に落とし込むと、分散処理にはめ込みやすい
- アルゴリズム的にベストとは限らない。しかし、水平分散のクラスタに簡単に落とし込める
- 簡単に力業に持ち込むためのフレームワーク

分散ファイルシステムHDFS

- Hadoop Distributed File System
- 耐障害性（データブロックの複製）
- ディスクのシークを減らすためのデータブロック（デフォルト64MB）
- ネットワークトポロジを意識し、効率的に分散処理を行うためのブロック配置を行う

Hadoopのいいところ

- データセンター内のもっとも貴重なリソースは？
 - ネットワーク帯域、次いでディスクの処理能力
- 「データローカリティ」を意識したデータブロックの複製とタスク配置
 - 複製度3なら、ローカルノード、別ラック内のノード、同一ラック内の別ノードへ
 - タスクはできる限りローカルのデータノード内のデータで処理を行えるように管理される



サンプル：最高気温を求める

```
00670119909999919 for line in sys.stdin:  
00430119909999919
```

•コードを書く方法：

- Hadoop Streaming（標準入出力。簡単だけど遅い）
- Java（これが標準）

この他にも状況に応じていろいろなやり方があるので、
Hadoop Confefence Japan 2009の資料も参考にしてください

```
line.strip()  
temp, q) = (val[15:19], val[87:92], val[92:93])
```

```
key, max_val) = (None, 0)
```

```
ne in sys.stdin:
```

```
, val) = line.strip().split("¥t")
```

```
ast_key and last_key != key:
```

```
print "%s¥t%s" % (last_key, max_val)
```

```
(last_key, max_val) = (key, int(val))
```

```
else:
```

```
(last_key, max_val) = (key, max(max_val, int(val)))
```

```
if last_key:
```

```
print "%s¥t%s" % (last_key, max_val)
```

mapタスクによる処理

Hadoopによるシャッフル

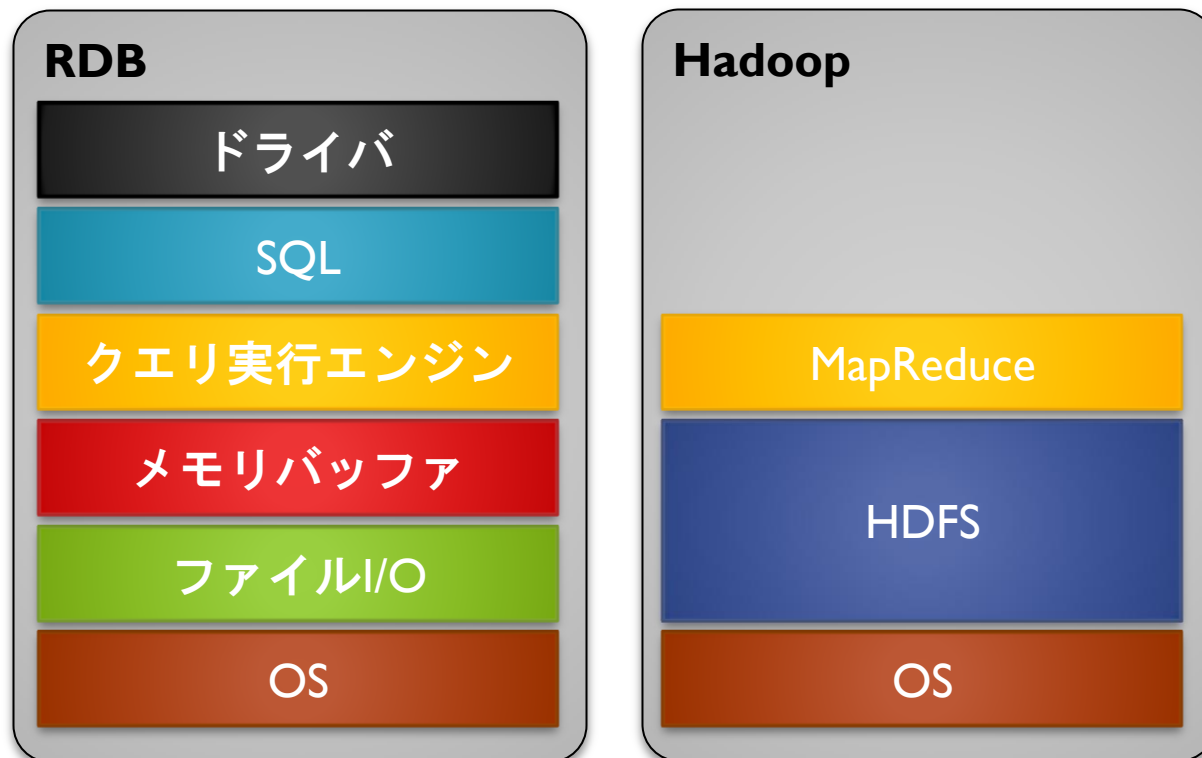
```
(1949, [111, 78, 119, 76])  
(1950, [0, 22, -11, 3, 21, -5])
```

reduceタスクによる処理

```
(1949, 119)  
(1950, 22)
```

HadoopとRDB

- HadoopはRDBをそのまま置き換えるものではない
- 少なくともレイヤーが異なっている
- 総合的に見れば、RDBは非常に優れている。極論を言えば、スケールしないだけ
- Hadoopの上で動作する、抽象度の高いプロジェクトが多く出てきた



NoSQL – Not Only SQL

- 正確にはNot Only RDBの方が正しい
- SQL / RDBがダメ（No SQL）ということではない。RDBはこれからも必須の技術
- BigDataの到来と共に、RDBだけで何でも片付けられる時代は終わりにつつある
- 技術者には、取り扱う問題に合わせて、ストレージやデータベースを選択する力が求められる
 - 例えばGoogle App EngineでもWindows Azureプラットフォームでも、データの保存にはテーブル・キュー・RDBが用意され（てい）る
 - ‘Free lunch’の時代は終わり。勉強しないと...
- Cassandra、MongoDBあたりが注目株？

事例紹介：Cookpad

- 日本最大のレシピサイト
 - 2010年3月現在の月間ユーザー数は884万人
 - 月間ページビュー数は4億6000万回
- 実は世界最大級のRuby on Railsサイトでもある
- Hadoopに関するプレゼンテーション
 - http://techlife.cookpad.com/2010/04/28/urapad_kyoto_presentation/
- 7000時間かかっていた処理が30時間に
 - 夏場のカレーはナスらしいです
- HadoopをAmazon EC2/S3で運用
 - 自社でデータセンターは持っていない

コミュニティの紹介

Hadoop Conference Japan 2009 講演



sbr



- Hadoop / NoSC

Hadoop Conference Japan 2009(2009/11/13)の公開講演スライドです。(英語の解説記事: [Overview of Hadoop Conference 2009 in Japan - English Edition](#))

gihyo.jp イベント Hadoop Hack Night Vol. 2

Qihyo.jp x YAHOO! JAPAN Presents "Open TechTalk"

HADOOP HACK NIGHT VOL. 2

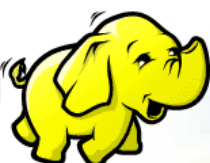


Photo by TheLizardQueen
http://www.flickr.com/photos/lizard_queen/ / CC BY 2.0

今回は「ケーススタディで知るHadoopの可能性」

2010年に入り企業内でのクラウドテクノロジーの導入が進んでいます。とは言うものの、まだまだ黎明期でもあるため、どの技術をどのように、どのタイミングで利用すればよいか悩んでいる方も多いのではないのでしょうか。

今回のHadoop Hack Nightは、「ケーススタディで知るHadoopの可能性」と題して、導入事例から見たHadoopに迫ります。Hadoop技術を牽引するYahoo! JAPAN、さらに十分な導入実績を持つリッテルのセッションに加えて、識者による熱いパネルディスカッションを予定しています。

これからHadoop導入を検討しているSierの方、これからHadoopについて学びたい方、必見です。

※好評のうちに終了しました、前回のHadoop Hack Nightのレポートは、こちらをご覧ください。

- Open TechTalk「Hadoop Hack Night」レポート (Techblog)

イベント概要

名称 Hadoop Hack Night Vol. 2

日程 8月4日 (水)

時間 19:00~21:00頃
※変更の場合があります。

場所 ヤフー株式会社
【アクセス: 東京ミッドタウン (六本木駅直結, 乃木坂駅徒歩10分)】

定員 100名
※応募多数の場合はお断りさせていただきます。あらかじめご了承ください。

URL <http://gihyo.jp/event/2010/hadoophn2>

株式会社技術評論社 ヤフー株式会社

西川徹さん(PFI)

技術研 河村さん/西岡さん(楽天市場)

大倉務さん

田中慎司さん(はてな)

政谷好伸さん(NTT Data)

小宮山康子さん(Telanium, Inc.)

中田さん(日経コンピュータ)

iesun Han(NexR/Korea HUG)

川さん(シリウステクノロジーズ)



はじめましょう

コミュニケーションが

cloudera

What is
Hadoop?

Why
Hadoop?

Downloads
Learn Hadoop
Get Support

Events
Blog
Twitter

Developer
Center

Products & Services Customers Partners Resources Downloads Company Contact

Search

Developer Center

Overview

Downloads

Learn Hadoop

Get Support

Blog

Newsletter

Subscribe to the
Cloudera Newsletter

Your email...

Download Cloudera Hadoop (CDH2) Quick Start Guide

Cloudera's Distribution for Hadoop (CDH) is more than all the other distributions combined. Added by Eli Collins, last edited by Paul Battaglia on 07/13/2010 (view change)

Installing Cloudera's Distribution for Hadoop (Version 2) on a Single Linux Node

Use the information in this section to quickly install CDH2 on a single Linux node in pseudo-distributed mode.

To install CDH2 on a single Linux node in pseudo-distributed mode:

1. Use a supported Linux version. The steps in this section assume you're installing CDH2 on a [supported Linux version](#).

Note for Ubuntu users: Cloudera does not provide Karmic packages, so you should use Jaunty instead of Karmic.

2. Install the Sun JDK.

Important

You must install Sun's JDK before installing Cloudera's RPM. Hadoop requires Sun JDK 1.6, update 8 at a minimum, which you can download from the [Java SE Homepage](#). You may be able to install the JDK with your package manager, depending on your choice of operating system. For JDK installation instructions, see [Java Development Kit Installation \(CDH2\)](#).

3. Install Cloudera's yum or apt repository.

A repository (yum for Red Hat systems or apt for Debian systems) enables your package manager to install the Cloudera packages very easily.

On a Debian system, add the following lines to /etc/apt/sources.list.d/cloudera.list (a new file). Replace DISTRO in the example below with the name of your distribution, which you can find by running `lsb_release -c` (using jaunty, for example):

```
deb http://archive.cloudera.com/debian DISTRO-cdh2 contrib
deb-src http://archive.cloudera.com/debian DISTRO-cdh2 contrib
```

On a Debian system, run the following commands. Note you may have to install curl by using the `sudo apt-get -y install curl` command:



ありがとうございました！