



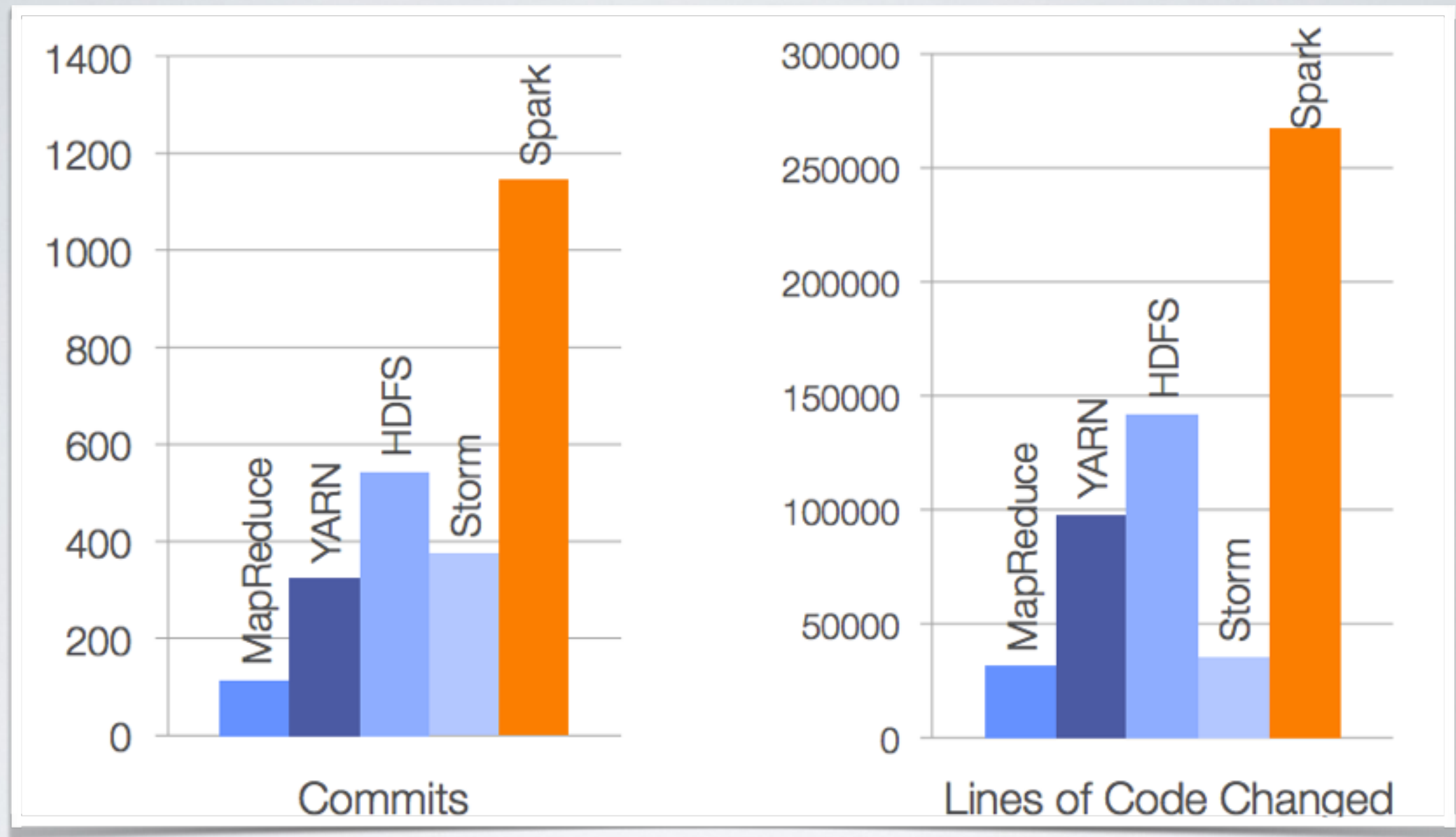
Apache Spark 勉強会

@teppei_tosa

Overview

1. Spark概要
2. Hadoop概要
3. Hadoop vs Spark
4. GraphX概要
5. WEBとJOBの融合
6. Getting Started

Sparkとは



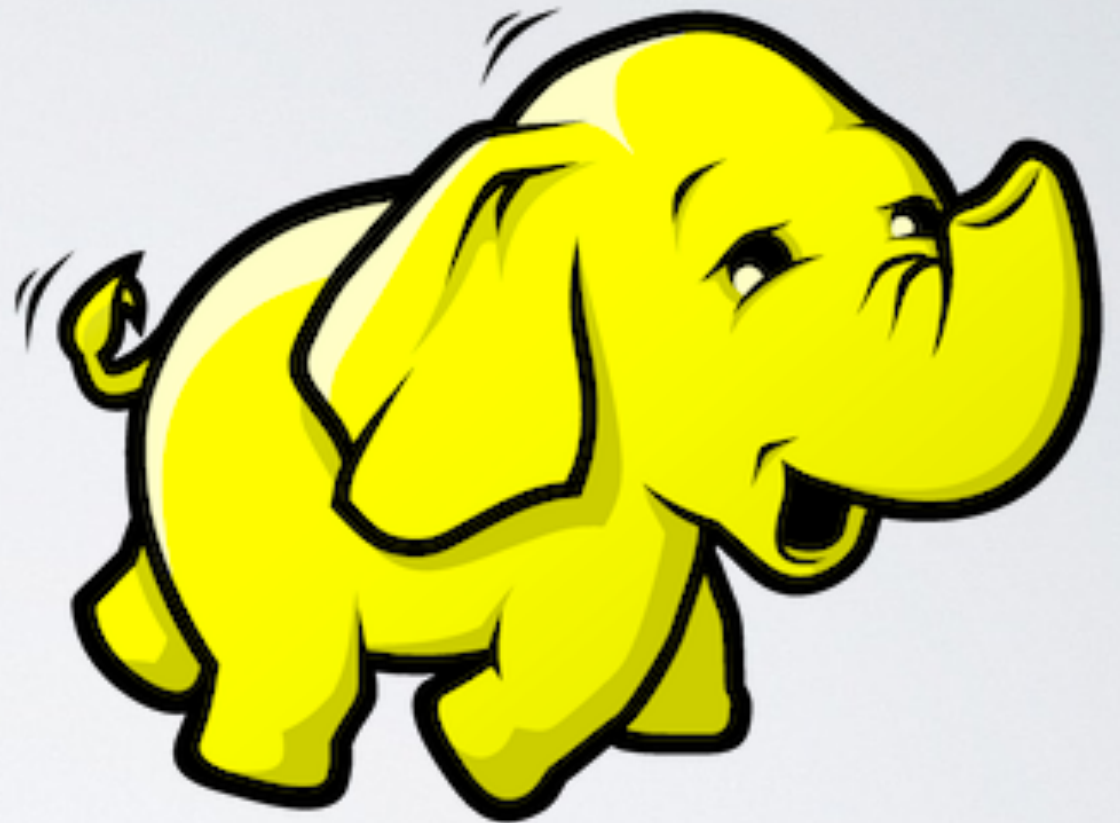
The most active project in the Hadoop ecosystem

- ・ 米カリフォルニア大学バークレー校 AMPLab で開発された分散コンピューティングフレームワーク
- ・ Databricks社がメインに開発
- ・ In-Memoryでの繰り返し処理が得意
- ・ MapReduceを置き換えるつつある
- ・ Scalaで実装されている。Python、Javaでも開発可能
- ・ コンポーネントとしてMLlib、GraphX、SQL

Hadoopを次のステージへ

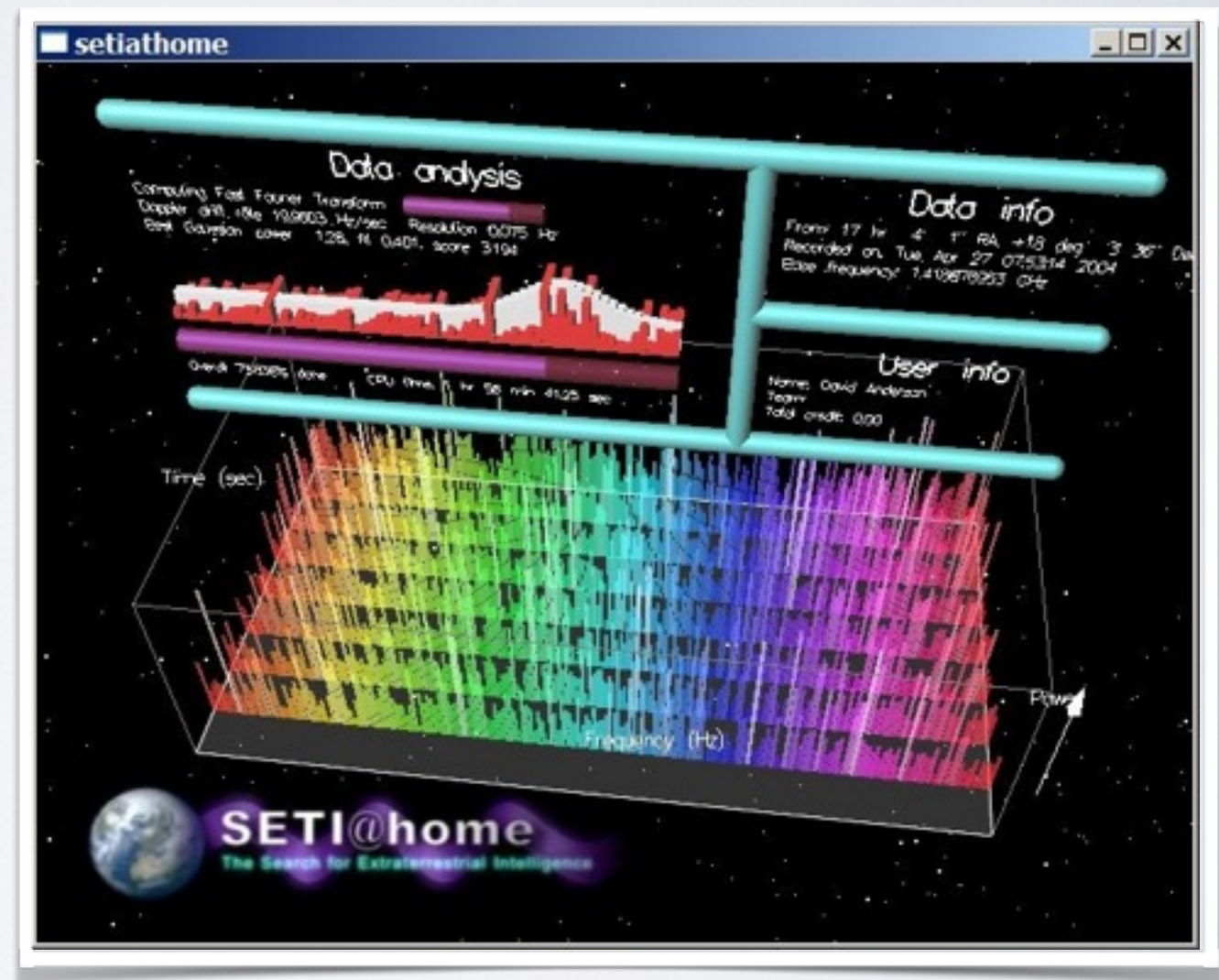
1. In-Memory処理による高速化
2. 俯瞰目線での設計
3. 分散処理間の通信

Hadoopとは

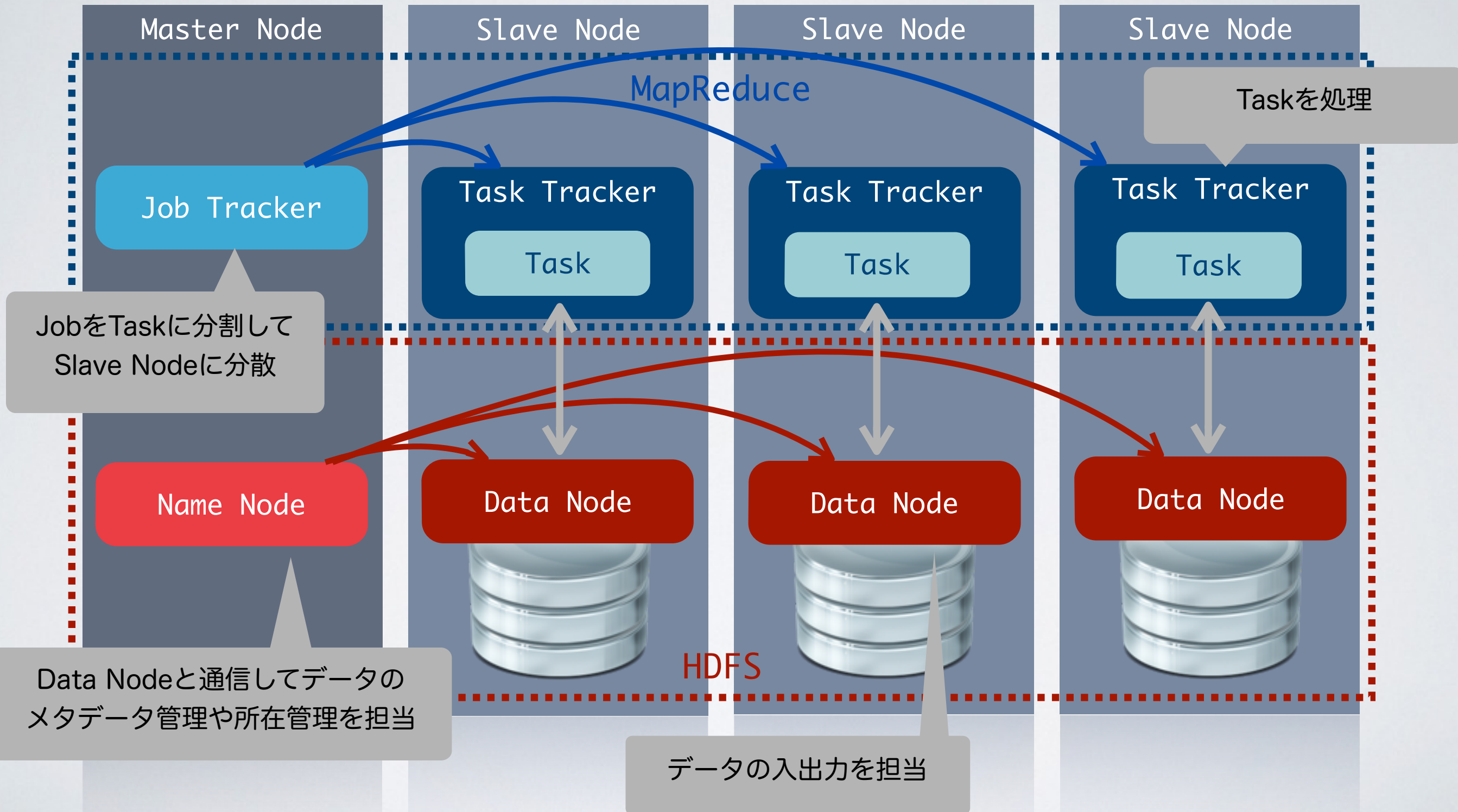


かつてのグリッドシステム

- ・ 分散コンピューティングのアイディアは昔から
- ・ 例：SETI@HOME
- ・ ネットI/O、ディスクI/Oがネックになってしまう
- ・ 処理の分散だけでは実現できない



HDFSによってネットI/OとディスクI/Oも分散



MapReduceによる効率的な分散処理開発

Apache Spark is an open-source data analytics cluster computing framework originally developed in the AMPLab at UC Berkeley.
Spark fits into the Hadoop ...

Data Block

Map Task

"Apache", 1
"Spark", 1
...

Data Block

Map Task

"data", 1
"analytics", 1
...

Data Block

Map Task

"Spark", 1
"fits", 1
...

Shuffle and Sort

"Spark", 1
"Spark", 1
...

Reduce Task

"Spark", 200

"data", 1
"data", 1
...

Reduce Task

"data", 100

"framework", 1
"framework", 1
...

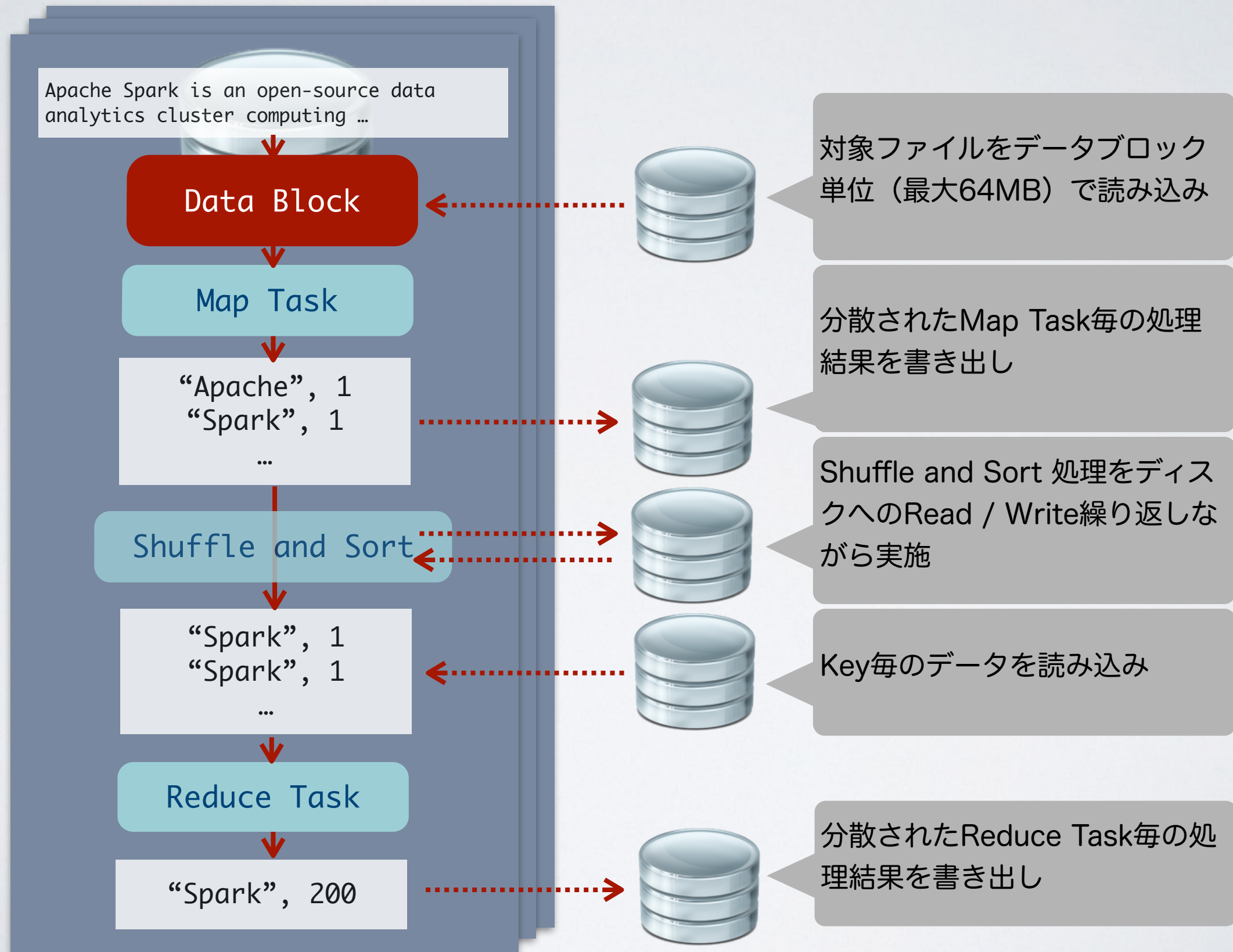
Reduce Task

"framework", 50

Hadoop vs. Spark

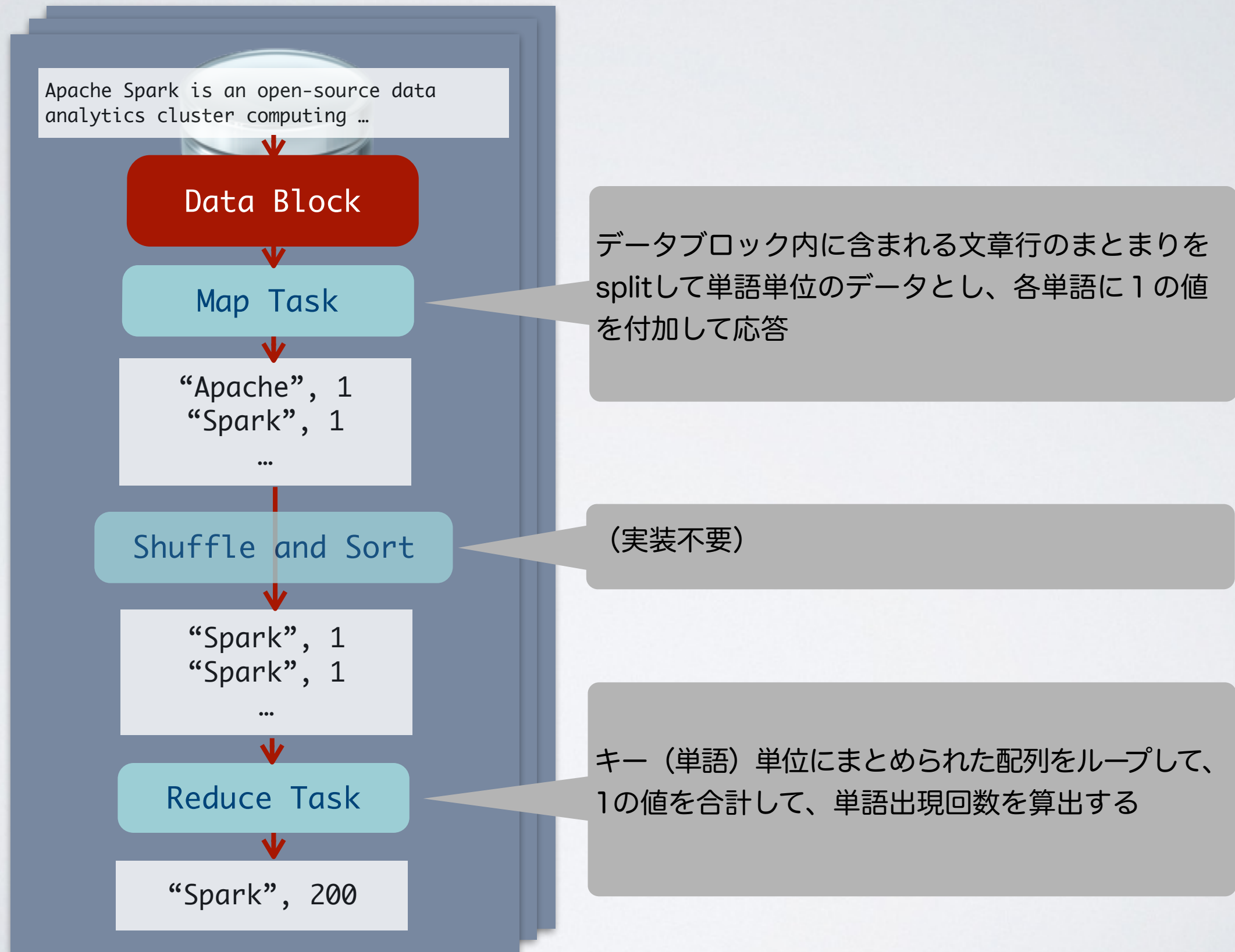
#	Hadoop	Spark
1	Disk I/O 過多	In-Memory処理による高速化
2	分散後処理設計の煩雑さ	俯瞰目線での設計
3	独立した分散処理	分散処理間の通信

Hadoop : Disk I/O過多



Hadoop：分散後処理設計の煩雑さ

(分散された後の処理を設計する難しさ)



Hadoop：独立した分散処理

(分散した処理間で情報共有できない)

分散処理間で情報共有したいケース

- ・ Shuffle & Sort処理

専用の処理としてストレージ Read / Write を
繰り返して実現

- ・ グラフ構造データ処理

辺でリンクされている頂点間の情報のやりとりをする処理

Apache Giraphでは分散処理機構管理用のエコシステムであるzookeeperをアプリケーション用に活用して解決

Sparkのプロセス構成

クラスタ管理機能

Slaveサーバ管理

Master Node

Slave Node

Slave Node

Slave Node

Spark

Cluster Manager

Worker Node

Worker Node

Worker Node

Driver

Executor

Executor

Executor

Spark Context

Task

Cache

Task

Cache

Task

Cache

Sparkアプリ本体

Name Node

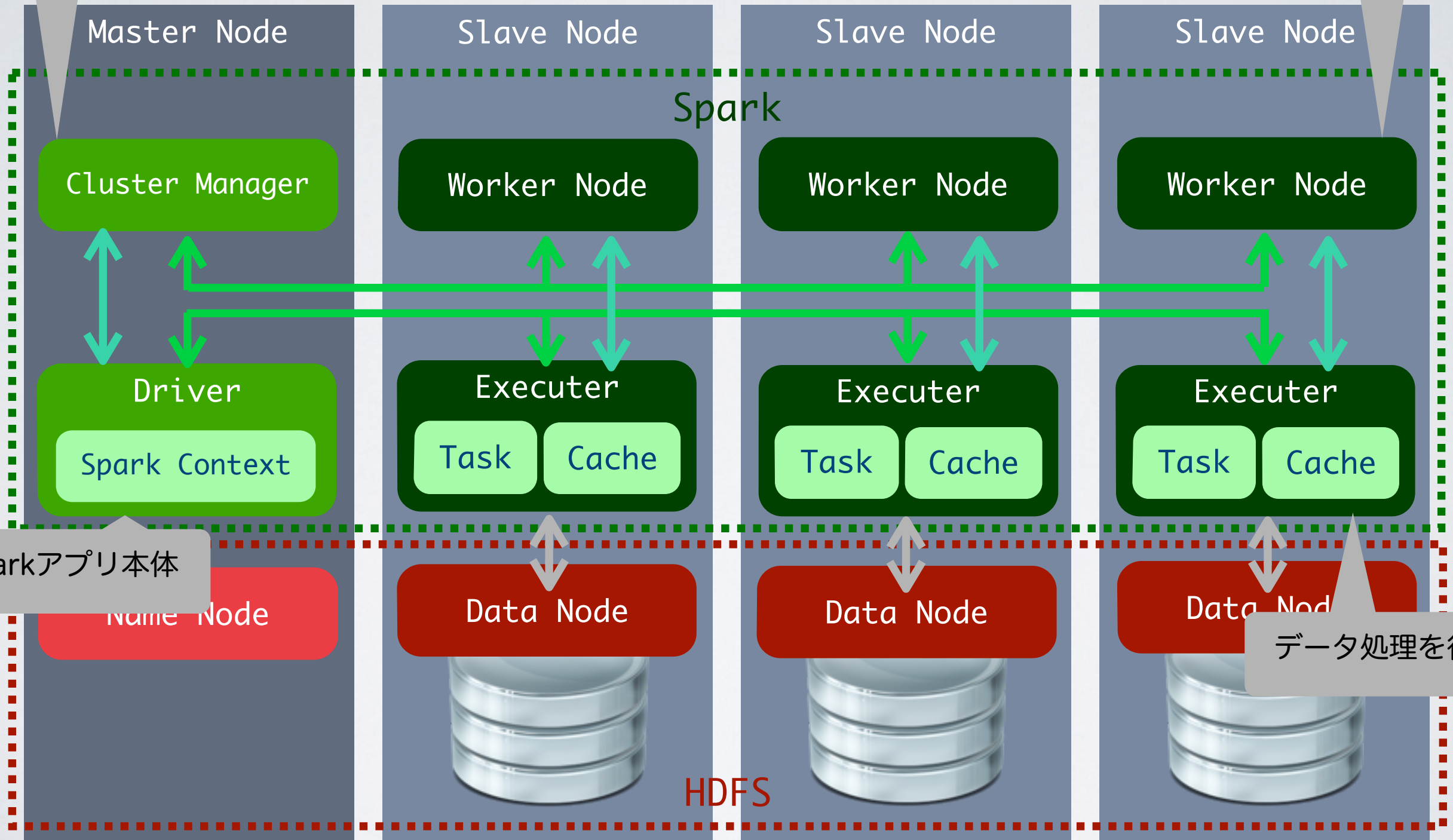
Data Node

Data Node

Data Node

データ処理を行う

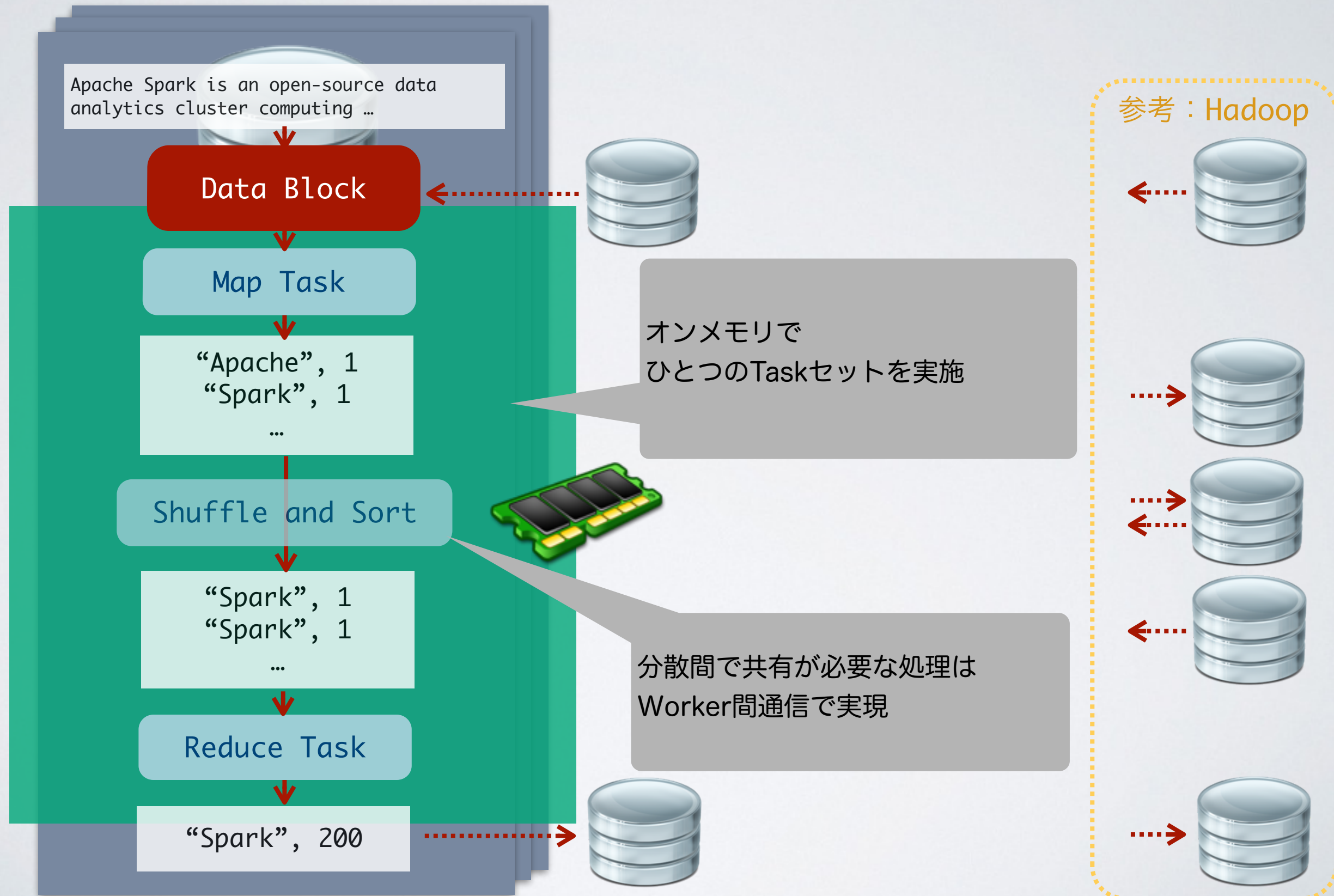
HDFS



Hadoop vs. Spark

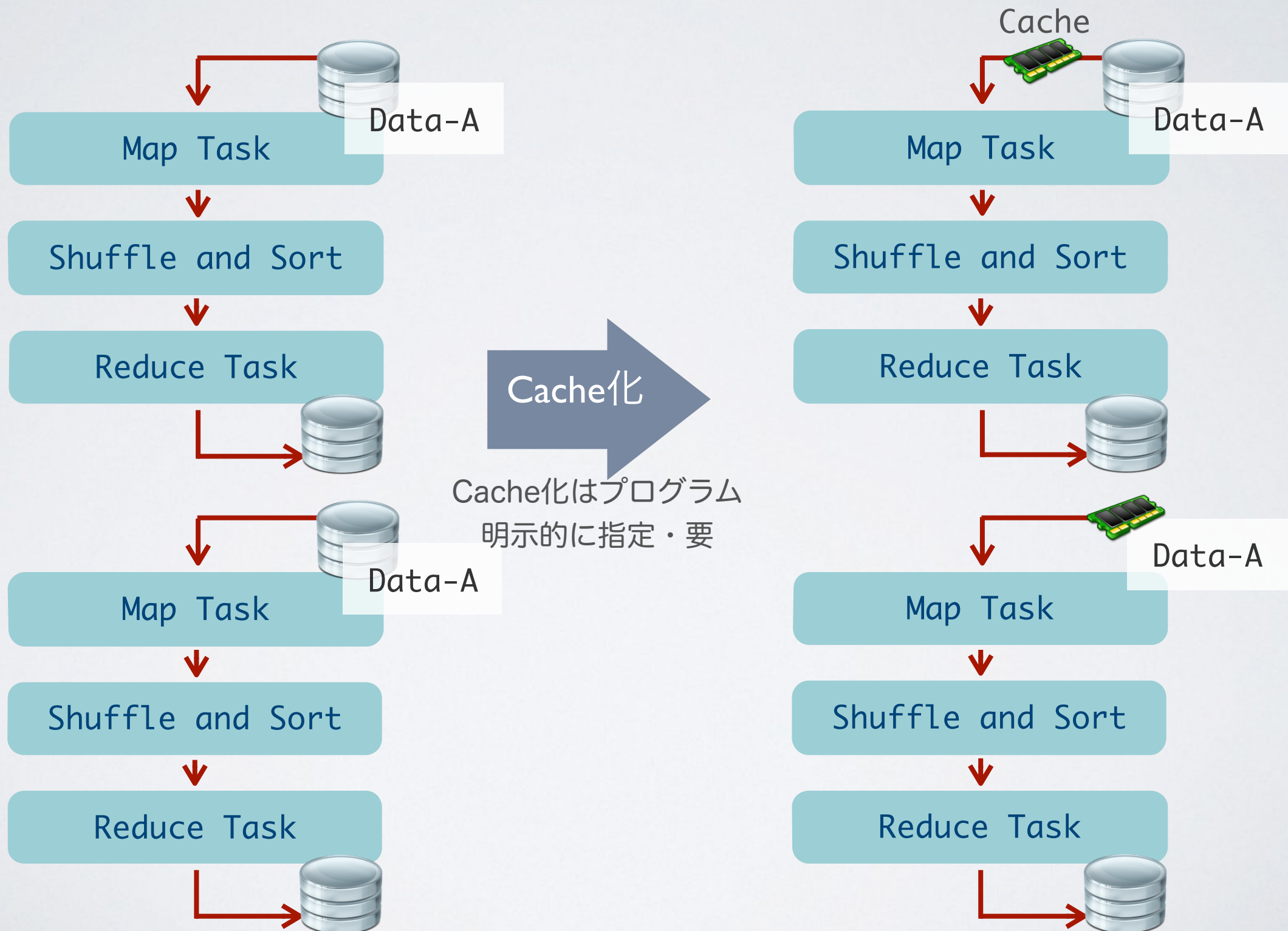
#	Hadoop	Spark
1	Disk I/O 過多	In-Memory処理による高速化
2	分散後処理設計の煩雑さ	俯瞰目線での設計
3	独立した分散処理	分散処理間の通信

Spark : In-Memory処理による高速化



Spark : In-Memory処理による高速化

Cache機能による複数処理セット間のデータ流用の効率化



Spark : 俯瞰目線での設計

WorkCount by MapReduce of Hadoop

```
....
public class WordCount {
    public static class Map extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, IntWritable> {
        ...

        public void map(    LongWritable key,
                           Text value,
                           OutputCollector<Text, IntWritable> output,
                           Reporter reporter)
            throws IOException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                output.collect(word, one);
            }
        }

        public static class Reduce extends MapReduceBase
            implements Reducer<Text, IntWritable, Text, IntWritable> {
            public void reduce(Text key,
                               Iterator<IntWritable> values,
                               OutputCollector<Text, IntWritable> output,
                               Reporter reporter
                               ) throws IOException {
                int sum = 0;
                while (values.hasNext()) {
                    sum += values.next().get();
                }
                output.collect(key, new IntWritable(sum));
            }
        }

        public static void main(String[] args) throws Exception {
            ...
            JobClient.runJob(conf);
        }
    }
}
```


Spark : 俯瞰目線での設計

WorkCount by MapReduce of Spark with Scala

```
val file = spark.textFile("hdfs://...")

val counts = file.flatMap(line => line.split(" "))
                  .map(word => (word, 1))
                  .reduceByKey(_ + _)

counts.saveAsTextFile("hdfs://...")
```


Spark : 俯瞰目線での設計

WorkCount by MapReduce of Spark with Java

```
JavaRDD<String> file = spark.textFile("hdfs://...");
```

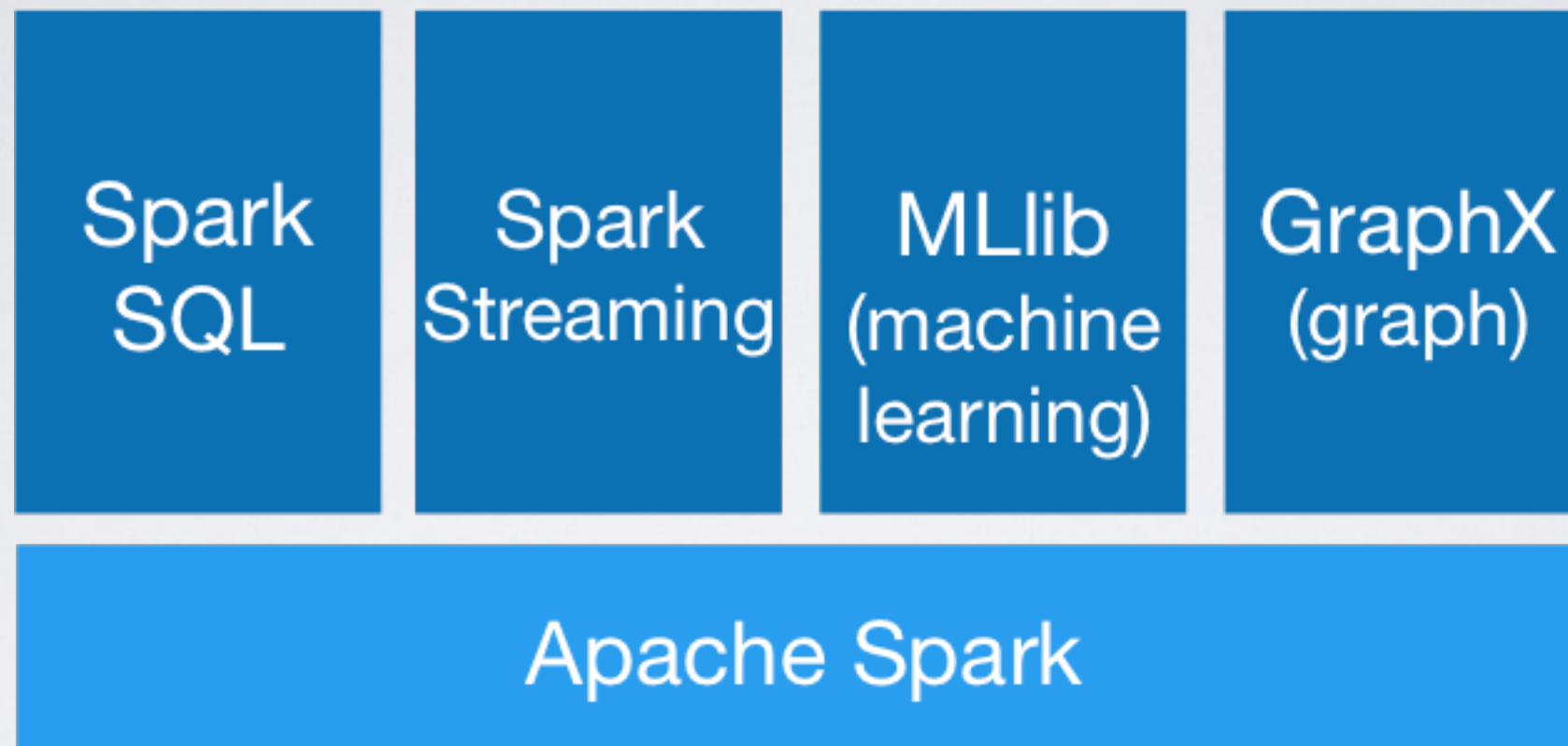
```
JavaRDD<String> words = file.flatMap(new FlatMapFunction<String, String>() {  
    public Iterable<String> call(String s) { return Arrays.asList(s.split(" ")); }  
});
```

```
JavaPairRDD<String, Integer> pairs = words.map(new PairFunction<String, String, Integer>() {  
    public Tuple2<String, Integer> call(String s) { return new Tuple2<String, Integer>(s, 1); }  
});
```

```
JavaPairRDD<String, Integer> counts = pairs.reduceByKey(new Function2<Integer, Integer>() {  
    public Integer call(Integer a, Integer b) { return a + b; }  
});
```

```
counts.saveAsTextFile("hdfs://...");
```

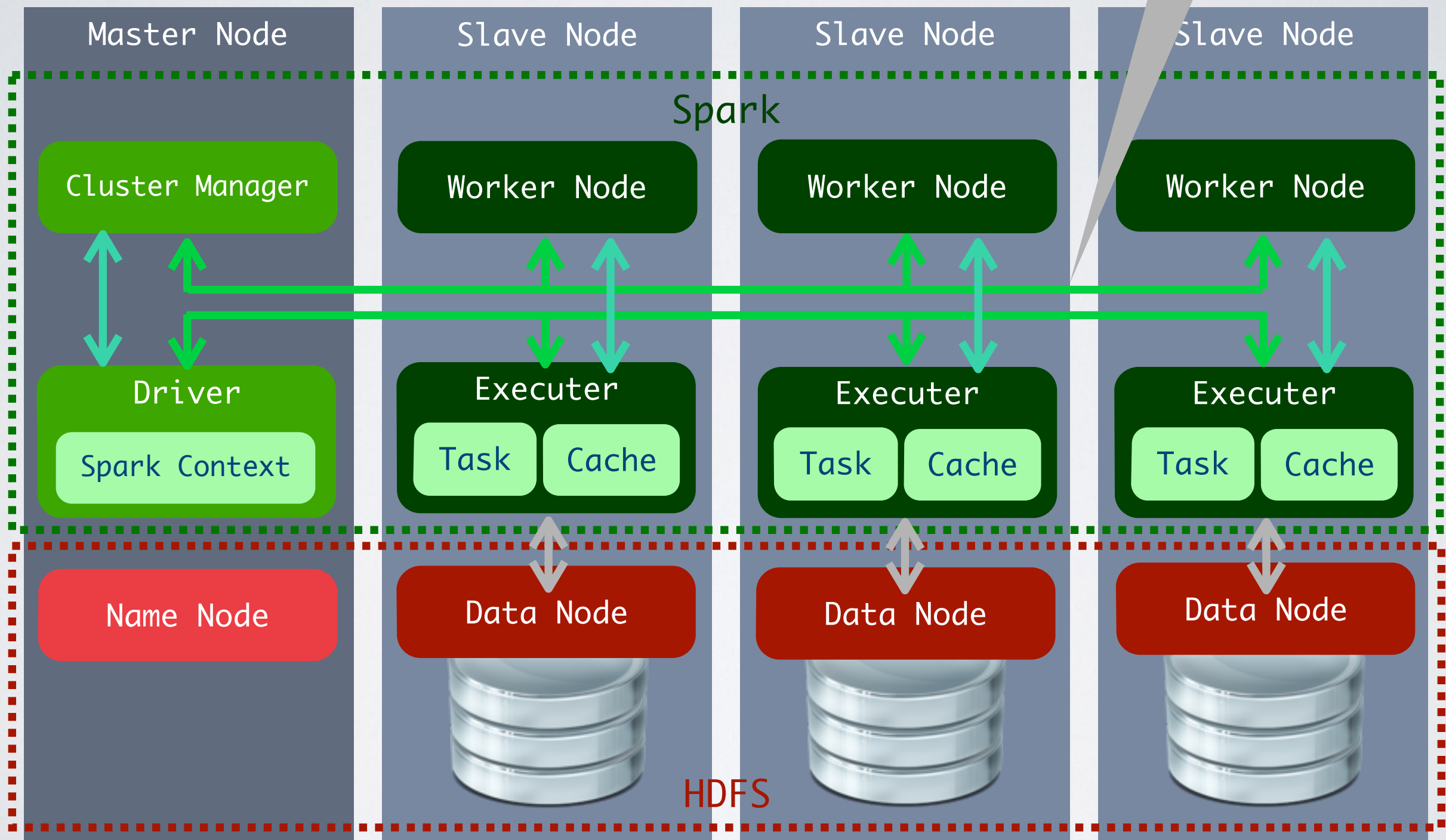
Spark : 俯瞰目線での設計



ひとつのSparkプログラム / JOB で、通常のETL処理、SQL、ストリーミング処理、機械学習による分析処理、グラフデータ分析処理の全てを実装可能
(これまでのHadoopでは、それぞれが別々のJOBとして実装する必要がある)

Spark : 分散処理間の通信

Worker間で通信可能



Spark：分散処理間の通信

分散処理間で情報共有したいケース

- ・ Shuffle & Sort処理

Worker間通信機能でオンメモリ処理を実現

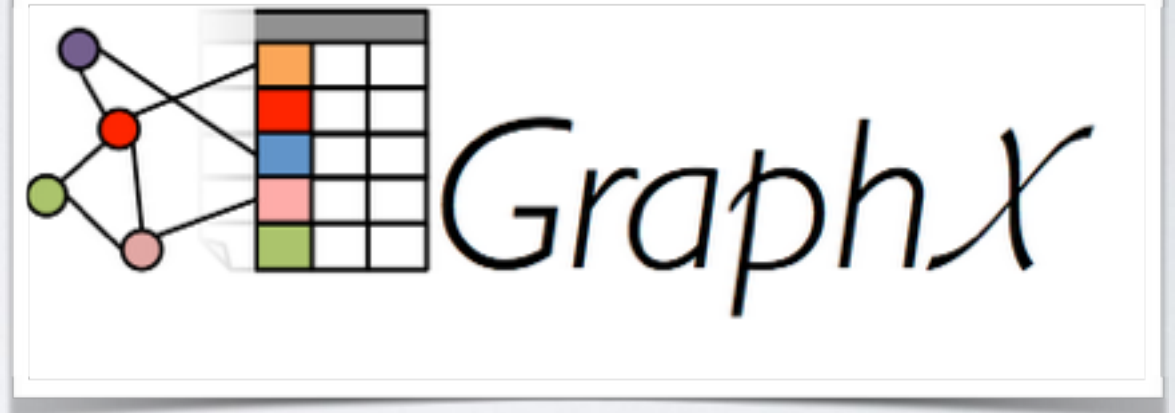
- ・ グラフ構造データ処理

Worker間通信機能でオンメモリ処理を実現

(その他、再帰的处理でメリットを発揮)

GraphX

- Sparkのコンポーネントのひとつ
- グラフ構造データを処理する
フレームワーク
- Sparkのメリットを最大限
活かしたフレームワーク

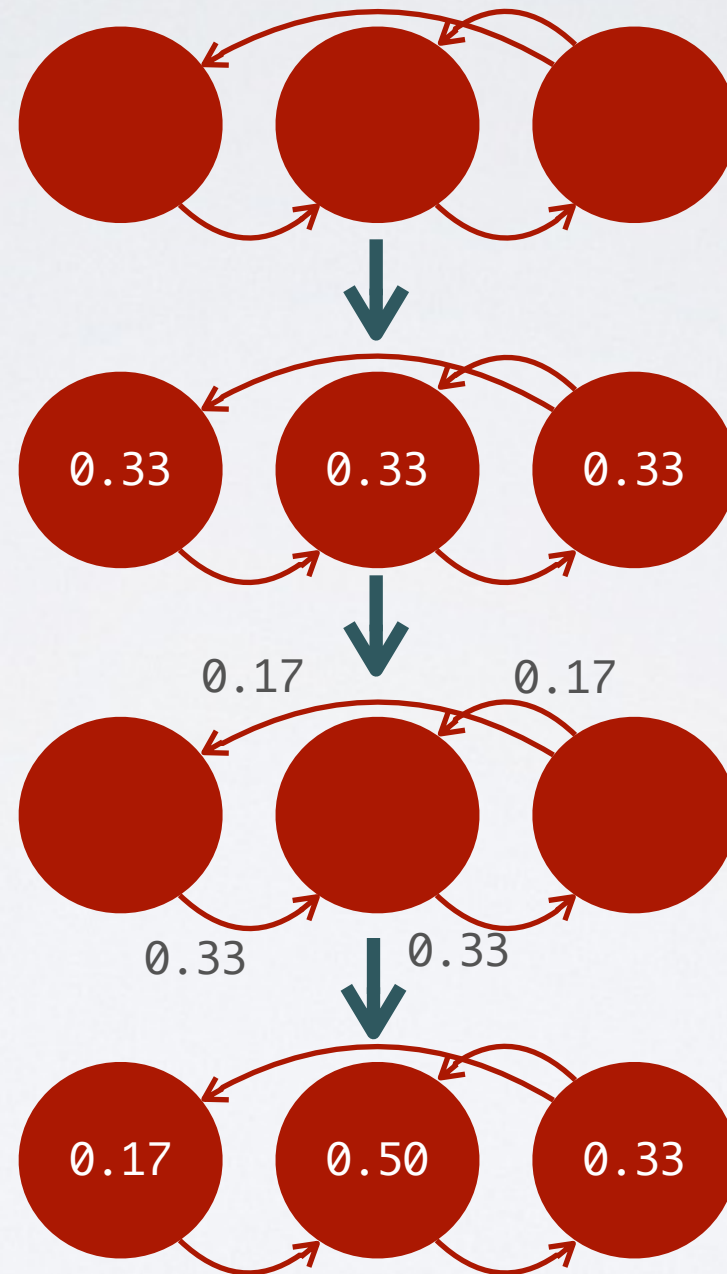


例：Page Rank Algorithm

1を頂点の数で割った値を
各頂点に配布

リンクを張っている数で
各頂点の値を割って
隣接頂点に配布

配布されてきた値を
合算して自頂点の値に設定



各頂点の値が
収束するまで繰り返す

Giraph vs. GraphX

#	Giraph on Hadoop	GraphX on Spark
1	Disk I/O 過多	In-Memory処理による高速化 → 再帰処理に好都合
2	分散後処理設計の煩雑さ → 頂点に成りきった処理設計	俯瞰目線での設計 → グラフ全体を俯瞰した設計 → 元データからのグラフ構造 データへの変換処理も含む
3	独立した分散処理 → ZooKeeperで実現	分散処理間の通信

WEBアプリとの融合へ

Databricks Cloud

<https://databricks.com/cloud>

WEBアプリとの融合へ

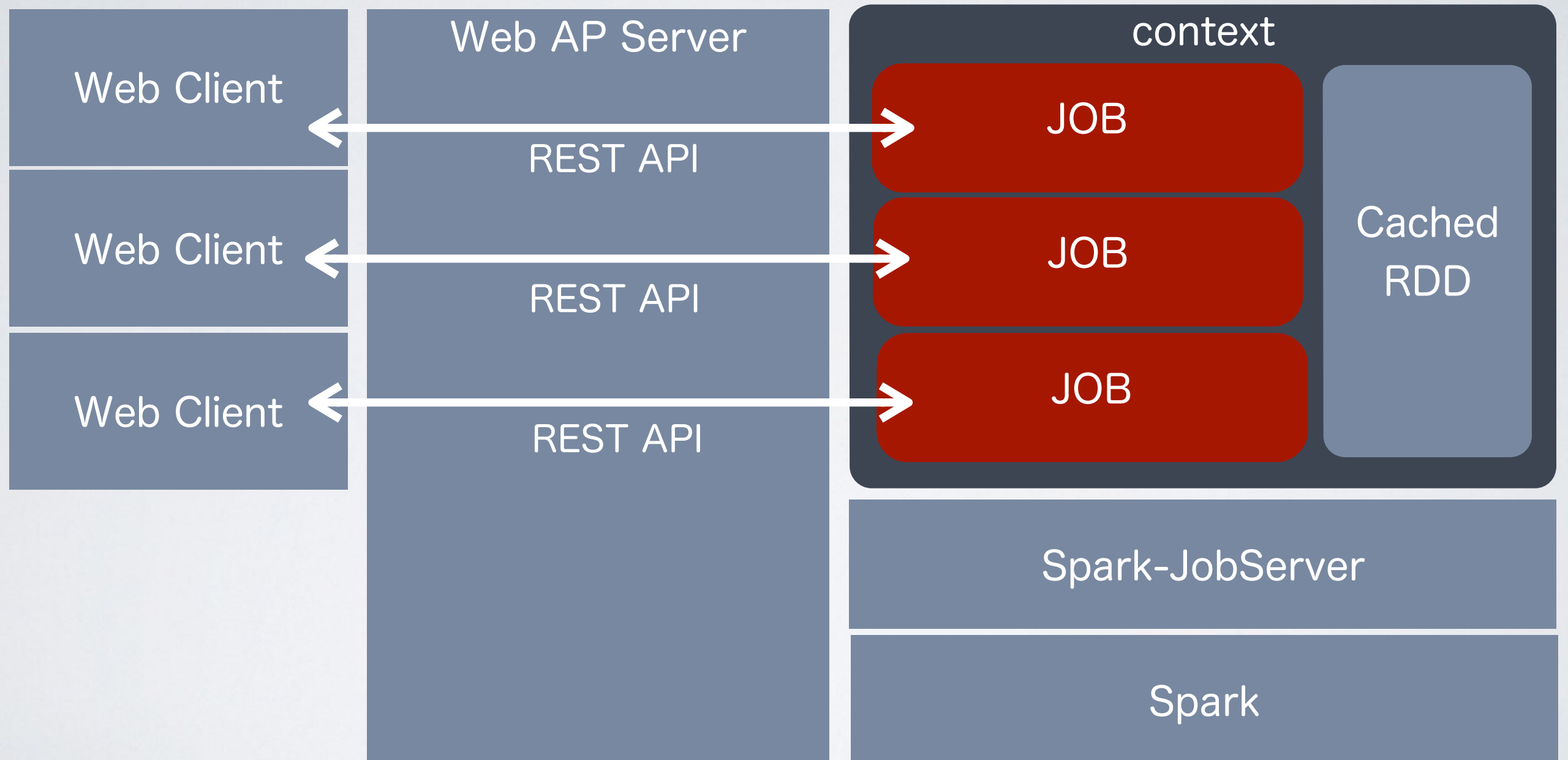
- ・ JOBの超高速化によりWEBシステムとの融合できる可能性が出て来た
- ・ WEB利用を想定したSparkのマルチスレッド機能
- ・ Spark-JobServerの登場

Spark-JobServer

- ・ “Spark as a Service”
- ・ SparkのjobとcontextのREST APIを提供
- ・ いかなる環境・言語からもSparkを扱うことが可能
- ・ Job間でひとつのcontextを共有
- ・ Job間でキャッシュされたRDDを共有
- ・ 同期/非同期API。JOB結果をJSON応答。

Demo

Graph-Web



Getting Started

動作モード

- Local Mode

クラスタを組まず、シングルノードで稼働するモード。開発端末で稼働させる際のモード。

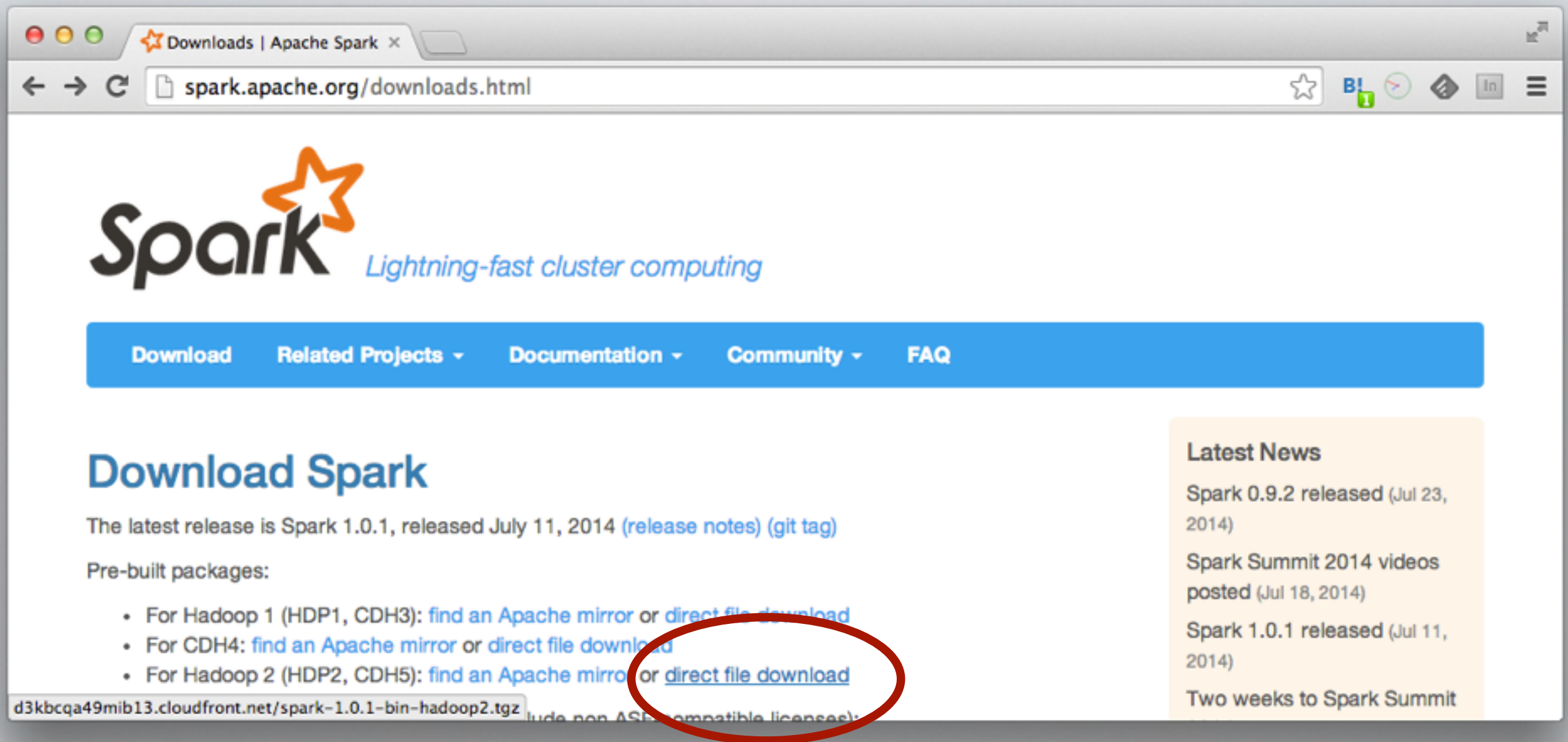
- Standalone Mode

Sparkだけでクラスタを構成するモード。Hadoop不要。

- YARN

Hadoop上で稼働させるモード。

ダウンロード



The screenshot shows a web browser window with the Apache Spark download page. The browser's address bar shows the URL `spark.apache.org/downloads.html`. The page features the Spark logo with the tagline "Lightning-fast cluster computing". A blue navigation bar contains links for "Download", "Related Projects", "Documentation", "Community", and "FAQ". The main heading is "Download Spark", followed by the text "The latest release is Spark 1.0.1, released July 11, 2014 (release notes) (git tag)". Under the heading "Pre-built packages:", there is a list of links for downloading Spark for different Hadoop distributions. The link "direct file download" for Hadoop 2 is circled in red. On the right side, a "Latest News" section lists recent releases and events.

Downloads | Apache Spark ×

spark.apache.org/downloads.html

Spark Lightning-fast cluster computing

Download Related Projects ▾ Documentation ▾ Community ▾ FAQ

Download Spark

The latest release is Spark 1.0.1, released July 11, 2014 ([release notes](#)) ([git tag](#))

Pre-built packages:

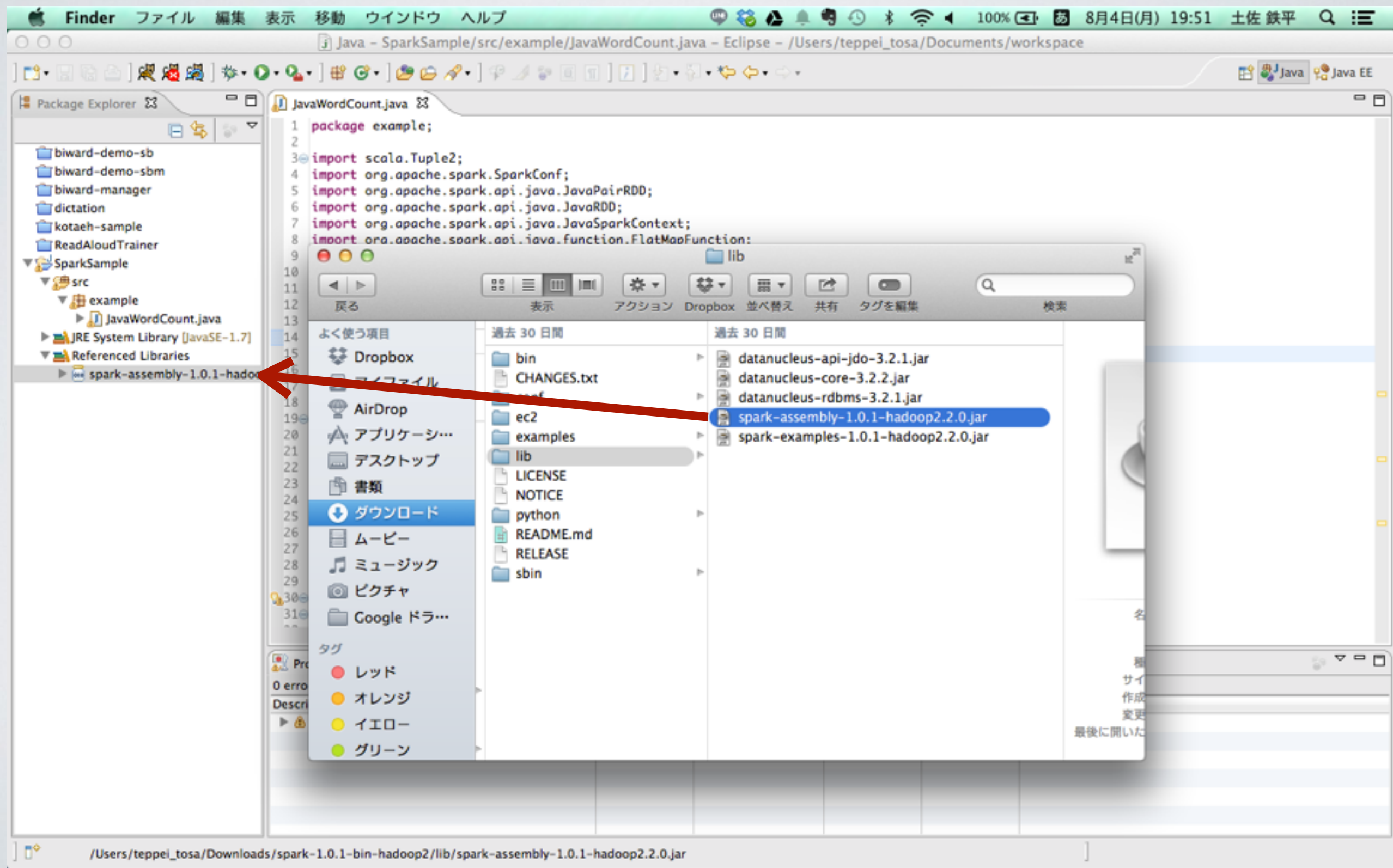
- For Hadoop 1 (HDP1, CDH3): [find an Apache mirror](#) or [direct file download](#)
- For CDH4: [find an Apache mirror](#) or [direct file download](#)
- For Hadoop 2 (HDP2, CDH5): [find an Apache mirror](#) or [direct file download](#)

d3kbcqa49mib13.cloudfront.net/spark-1.0.1-bin-hadoop2.tgz

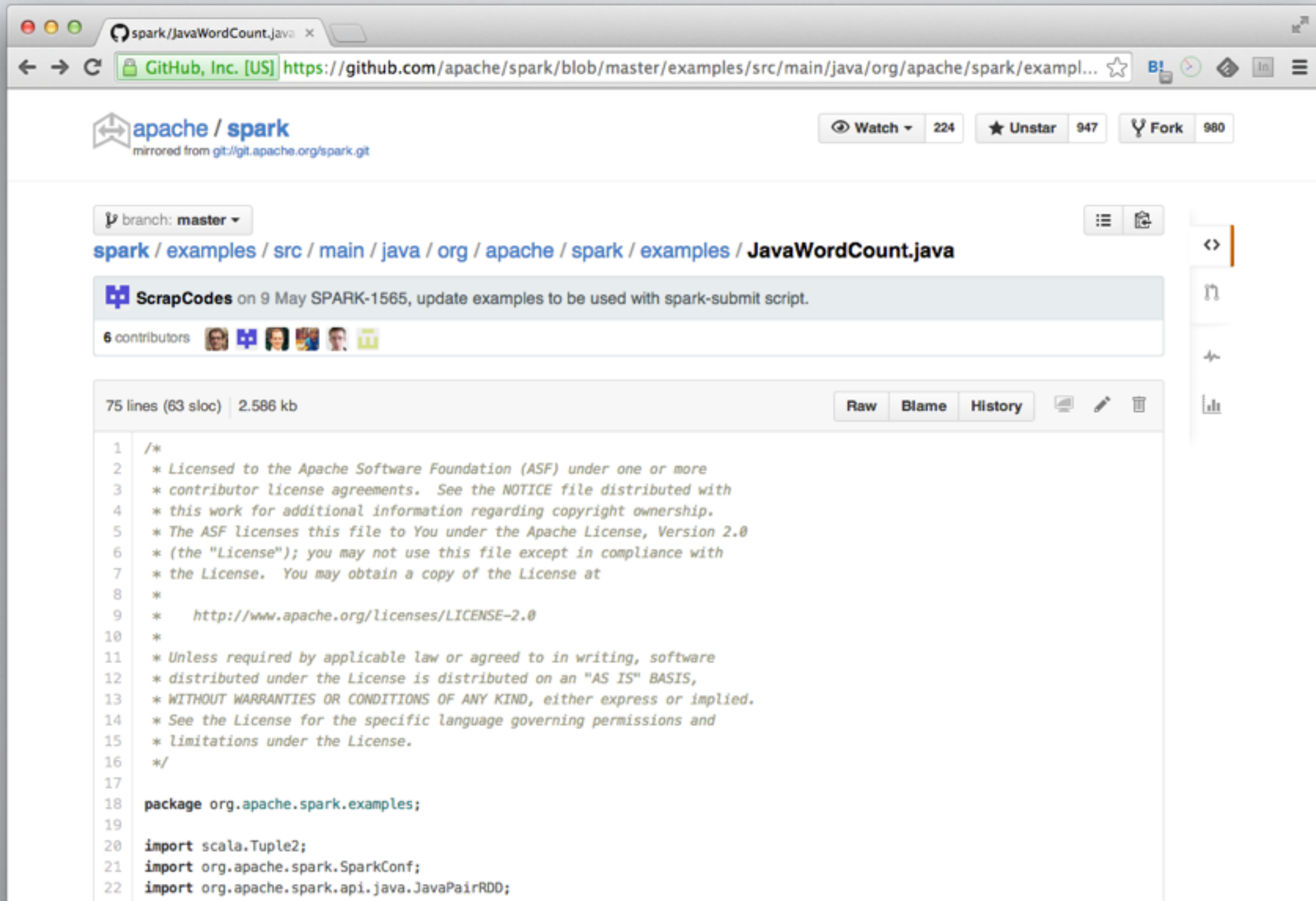
Latest News

- Spark 0.9.2 released (Jul 23, 2014)
- Spark Summit 2014 videos posted (Jul 18, 2014)
- Spark 1.0.1 released (Jul 11, 2014)
- Two weeks to Spark Summit

library 取り込み



Sample Source



The screenshot shows a web browser displaying the GitHub page for the file `spark/JavaWordCount.java`. The browser's address bar shows the URL `https://github.com/apache/spark/blob/master/examples/src/main/java/org/apache/spark/exampl...`. The page header includes the Apache Spark logo, a mirror link to `git://git.apache.org/spark.git`, and interaction buttons: Watch (224), Unstar (947), and Fork (980). Below the header, the file path `spark / examples / src / main / java / org / apache / spark / examples / JavaWordCount.java` is shown. A commit message by ScrapCodes on May 9, 2015, is visible. The file statistics show 75 lines (63 sloc) and 2.586 kb. The code is displayed in a light gray editor with line numbers. The first 16 lines are a multi-line comment block containing the Apache License, Version 2.0. The code starts with package and import statements.

```
1  /*
2   * Licensed to the Apache Software Foundation (ASF) under one or more
3   * contributor license agreements. See the NOTICE file distributed with
4   * this work for additional information regarding copyright ownership.
5   * The ASF licenses this file to You under the Apache License, Version 2.0
6   * (the "License"); you may not use this file except in compliance with
7   * the License. You may obtain a copy of the License at
8   *
9   * http://www.apache.org/licenses/LICENSE-2.0
10  *
11  * Unless required by applicable law or agreed to in writing, software
12  * distributed under the License is distributed on an "AS IS" BASIS,
13  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14  * See the License for the specific language governing permissions and
15  * limitations under the License.
16  */
17
18  package org.apache.spark.examples;
19
20  import scala.Tuple2;
21  import org.apache.spark.SparkConf;
22  import org.apache.spark.api.java.JavaPairRDD;
```

まとめ

- ・ SparkはHadoopを次のステージへ
- ・ WEBアプリとの融合して革新的な変革を
- ・ 簡単にスタート可能！

参考情報

- Apache Spark 本家ページ
<http://spark.apache.org/>
- Spark Summit 2014 Agenda (資料DL可)
<http://spark-summit.org/2014/agenda>

参考情報

- ・ NTTデータ Spark紹介資料

基本編

<http://www.slideshare.net/>

[hadoopxnttdata/apache-spark-spark](http://www.slideshare.net/hadoopxnttdata/apache-spark-spark)

技術トピック

<http://www.slideshare.net/>

[hadoopxnttdata/apache-spark](http://www.slideshare.net/hadoopxnttdata/apache-spark)

Thank you !

@teppei_tosa