OpenSource/Linux技術文書

# Cloudera Distribution for Hadoop 4.3.0 インストール手順及び利用例

日本ヒューレット・パッカード株式会社
プリセールス統括本部　ソリューションセンター　Linux・OSSソリューション部
古賀 政純

2013 年 8 月 5 日

目次

図表目次

[本ドキュメントについて]

- 本ドキュメントでは、NameNode サーバーのみでの作業を(NN)、DataNode サーバーのみでの作業を(DN)、NameNode サーバーと DataNode サーバー両方での作業を(NN,DN)と記すことにします。
  - 例 1)
    「ファイルをコピーします。（NN）」と記載してあるものは、NameNode だけでファイルをコピーするという意味になります。
  - 例 2)
    「rpm コマンドでパッケージをインストールします（NN,DN）」と記載してあるものは、NameNode と DataNode の両方で rpm コマンドを使ってインストールを行うという意味になります。

- コマンドラインでの入力が長く紙面の都合で折り返して記載する場合は、下記のように「¥」記号を挿入して複数行にわたって記載しています。複数行にわたって記載されていても実際には 1 行で入力するものは、その記述の最後に「（実際には 1 行で入力）」を挿入しています。
  - 例 3)

```
# alternatives --install /etc/hadoop-conf hadoop-conf ¥
/etc/hadoop-conf.cluster 20   （実際には 1 行で入力）
```

- 本ドキュメントの内容については、その正確性を保証するものではありません。また、将来、予告なしに変更することがあります。
- 本ドキュメントの使用で生じるいかなる結果も利用者の責任となります。日本ヒューレット・パッカード株式会社は、本ドキュメントの内容に一切の責任を負いません。
- ハードウェア構成、OS、アプリケーション等の使用環境により大幅に性能が変化する場合がありますので、十分なテストを個別に実施されることを強くお薦め致します。
- 本ドキュメント内で表示・記載されている会社名・サービス名・商品名等は各社の商標又は登録商標です。
- 本ドキュメントで提供する資料は、日本の著作権法、条約及び他国の著作権法にいう著作権により保護されています。

本ドキュメントはCloudera distribution for Hadoop 4.3.0(通称CDH4.3.0)をHP ProLiantサーバーで構築するためのガイドです。

## システム構成

以下にCDH4.3.0のシステム構成例を示します。

| | |
|---|---|
| ハードウェア | : HP ProLiant SL4540 Gen8 |
| OS | : Red Hat Enterprise Linux 6.4 x86-64 (NameNodeおよびDataNode) |
| JDK | : 1.7.0u25 x64 (Oracle社が提供するパッケージを利用) |
| Hadoop | : CDH4.3.0 (Red Hat系OSに対応したRPMパッケージを利用) |

本番環境においては、NameNodeのデータ保全性とサービスの可用性を考慮して下さい。以下は、CDHをSL4540 Gen8で構成する場合の一例です。

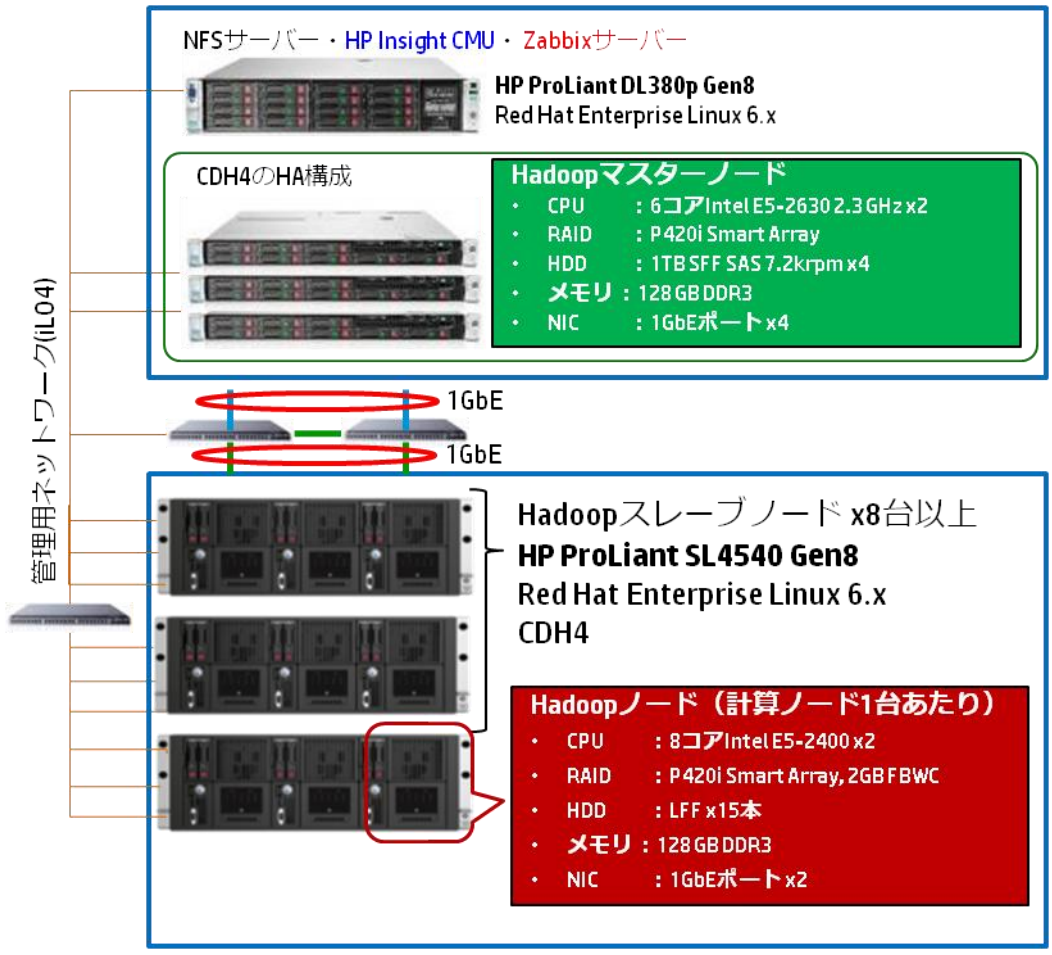


図 1. Cloudera Distribution for Hadoop on HP ProLiant SL4540 のシステム構成例

## RHEL6.x のインストールと各種 OS の設定（NN,DN）

OSのディスクパーティションは以下を想定します。今回は、テスト環境での利用のため、/boot、swap、/パーティションのみで分割していますが、本番環境では適宜データ用のディスクパーティションを分割する等の設計を行って下さい。OSはRHEL6.xのx86-64版を使います。

NameNodeのディスクパーティション例：

```
/boot    100MB
swap     1024MB
/        残りすべて
```

DataNodeのディスクパーティション例：

```
/boot    100MB
swap     1024MB
/        残りすべて
```

## iptables と SELinux の設定の無効化

パケットフィルタリングをOFFに、SELinuxをDisableにして、OSを再起動します。

```
# chkconfig iptables off
# service iptables stop
# vi /etc/sysconfig/selinux
...
SELINUX=disabled
...
SELINUXTYPE=targeted

# reboot
```

## iptables と SELinux の設定無効化の確認

iptablesとSELinuxが無効になっているかを確認します。

```
# getenforce
Disabled

# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

今回はiptablesを無効にしましたが、商用利用ではセキュリティの観点から、CDH4.3.0が利用するポートを考慮し、iptablesを適切に設定することもありえますので、システム要件によって適切なセキュリティ設定を行ってください。

## hosts ファイルの編集

NameNodeとDataNodeの/etc/hostsファイルを編集します。（NN,DN）

```
# vi /etc/hosts
127.0.0.1               localhost.localdomain localhost
172.16.1.1              hd01.jpn.linux.hp.com hd01
172.16.1.2              hd02.jpn.linux.hp.com hd02
172.16.1.3              hd03.jpn.linux.hp.com hd03
172.16.1.4              hd04.jpn.linux.hp.com hd04
172.16.1.5              hd05.jpn.linux.hp.com hd05
172.16.1.6              hd06.jpn.linux.hp.com hd06
172.16.1.7              hd07.jpn.linux.hp.com hd07
172.16.1.8              hd08.jpn.linux.hp.com hd08
```

## JRE のインストール(NN,DN)

NameNodeとDataNodeにJREをインストールします。JREのキットは以下のURLから入手可能です。

http://www.oracle.com/technetwork/java/javase/downloads/index.html

JREのキットは、Oracle社提供のjre-7u25-linux-x64.rpmをダウンロードします。ダウンロードしたキットをrpmコマンドでインストールします。

```
# rpm -vhi jre-7u25-linux-x64.rpm
```

## JRE のバージョン確認（NN,DN）

NameNodeとDataNodeにおいて、JREのバージョンを確認します。JREのバージョンの確認はjavaコマンドで行います。

```
# java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b15)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
```

## Cloudera Distribution for Hadoop (CDH) のインストール

CDHを、以下URLに示すClouderaのダウンロードサイトから入手します。
http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/4.3.0/RPMS/noarch/

CDHの特徴：

- オープンソース100% Apacheライセンス
- 多くのバグフィックスをバックポート
- RPMパッケージ等により導入が容易
- Hadoop本体以外の周辺のHiveやHbaseなどもRPMパッケージ化されている
- コンポーネント間のバージョン依存関係の複雑さを解決

rpmコマンドでCDHをNameNodeにインストールします。今回は、NameNodeとなるサーバーが1台ですので、NameNodeにインストールするものは、Hadoop本体とNameNode用のパッケージ、jobtracker、secondarynamenodeになります。(NN)

```
# rpm -vhi bigtop-utils-0.6.0+73-1.cdh4.3.0.p0.17.el6.noarch.rpm
# rpm -vhi bigtop-jsvc-1.0.10-1.cdh4.3.0.p0.14.el6.x86_64.rpm
# rpm -vhi zookeeper-3.4.5+19-1.cdh4.3.0.p0.14.el6.noarch.rpm
# rpm -vhi hadoop-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
# rpm -vhi hadoop-hdfs-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
# rpm -vhi hadoop-0.20-mapreduce-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
# rpm -vhi hadoop-0.20-mapreduce-jobtracker-2.0.0+1357-1.cdh4.3.0.p0.21.el6.noarch.rpm
# rpm -vhi hadoop-hdfs-namenode-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
# rpm -vhi hadoop-hdfs-secondarynamenode-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
```

DataNode群にパッケージをインストールします。DataNodeにインストールするものは、Hadoop本体とDataNode用のパッケージ、tasktrackerとなります。(DN)

```
# rpm -vhi bigtop-utils-0.6.0+73-1.cdh4.3.0.p0.17.el6.noarch.rpm
# rpm -vhi bigtop-jsvc-1.0.10-1.cdh4.3.0.p0.14.el6.x86_64.rpm
# rpm -vhi zookeeper-3.4.5+19-1.cdh4.3.0.p0.14.el6.noarch.rpm
# rpm -vhi hadoop-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
# rpm -vhi hadoop-hdfs-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
# rpm -vhi hadoop-hdfs-datanode-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
# rpm -vhi hadoop-0.20-mapreduce-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
# rpm -vhi hadoop-0.20-mapreduce-tasktracker-2.0.0+1357-1.cdh4.3.0.p0.21.el6.noarch.rpm
```

Hadoop の設定ファイルの雛型があるディレクトリ conf.empty を/etc/hadoop-conf.cluster にコピーします。今回はディレクトリ名を conf.cluster とします。(NN,DN)

```
# cp -r /etc/hadoop-conf.empty /etc/hadoop-conf.cluster
# ls -l /etc/hadoop-conf
conf/ conf.empty/ conf.cluster/
# ls -l /etc/hadoop-conf.cluster
合計 76
-rw-r--r-- 1 root root 5041     2月 24 15:00 capacity-scheduler.xml
-rw-r--r-- 1 root root  535     2月 24 15:00 configuration.xsl
-rw-r--r-- 1 root root  178     2月 24 15:00 core-site.xml
-rw-r--r-- 1 root root 3032     2月 24 15:00 fair-scheduler.xml
-rw-r--r-- 1 root root 2340     2月 24 15:00 hadoop-env.sh
-rw-r--r-- 1 root root 2109     2月 24 15:00 hadoop-metrics.properties
-rw-r--r-- 1 root root 4644     2月 24 15:00 hadoop-policy.xml
```

```
-rw-r--r-- 1 root root  178     2月 24 15:00 hdfs-site.xml
-rw-r--r-- 1 root root 4198     2月 24 15:00 log4j.properties
-rw-r--r-- 1 root root 2033     2月 24 15:00 mapred-queue-acls.xml
-rw-r--r-- 1 root root  178     2月 24 15:00 mapred-site.xml
-rw-r--r-- 1 root root   10     2月 24 15:00 masters
-rw-r--r-- 1 root root   10     2月 24 15:00 slaves
-rw-r--r-- 1 root root 1243     2月 24 15:00 ssl-client.xml.example
-rw-r--r-- 1 root root 1195     2月 24 15:00 ssl-server.xml.example
-rw-r--r-- 1 root root  382     2月 24 15:00 taskcontroller.cfg
```

現在の Hadoop の設定情報を alternatives コマンドによって確認します。(NN,DN)

```
# alternatives --display hadoop-conf
hadoop-conf -ステータスは自動です。
リンクは現在 /etc/hadoop/conf.empty を指しています。
/etc/hadoop/conf.empty - 優先項目 10
現在の「最適」バージョンは /etc/hadoop/conf.empty です
```

利用する設定ファイルを update-alternatives コマンドで選択します。(NN, DN)

```
# update-alternatives --install /etc/hadoop/conf hadoop-conf ¥
 /etc/hadoop/conf.cluster 20   (実際には1行で入力)
```

現在の設定情報を alternatives コマンドによって確認します。(NN,DN)

```
# alternatives --display hadoop-conf
hadoop-conf -ステータスは自動です。
リンクは現在 /etc/hadoop/conf.cluster を指しています。
/etc/hadoop/conf.empty - 優先項目 10
/etc/hadoop/conf.cluster - 優先項目 20
現在の「最適」バージョンは /etc/hadoop/conf.cluster です。
```

hadoop-env.sh ファイルを編集します。設定ファイル hadoop-env.sh において、JAVA_HOME を以下のように設定します。(NN,DN)

```
# pwd
/etc/hadoop-conf.cluster
# vi hadoop-env.sh
...
export JAVA_HOME=/usr/java/default
...
```

また上記 JAVA_HOME=/usr/java/default と設定した場合に、以下のようにシンボリックリンクが張られているかどうかを確認します。

```
# ls -l /usr/java/default
lrwxrwxrwx 1 root root 16  7月 26 16:49 2013 /usr/java/default -> /usr/java/latest
# ls -l /usr/java/latest
lrwxrwxrwx 1 root root 21  7月 30 21:33 2013 /usr/java/latest -> /usr/java/jre1.7.0_25
```

Hadoop 用のディレクトリを作成します。 キャッシュ用ディレクトリは/var 以下に作成します。(NN, DN)

```
# mkdir -p /var/data1/tmp/
```

```
# chmod -R 1777 /var/data1/tmp/
# mkdir -p /var/data1/dfs/nn /var/data2/dfs/nn /var/data3/dfs/nn
# chown -R hdfs:hdfs /var/data1/dfs/nn /var/data2/dfs/nn /var/data3/dfs/nn
# chmod 700 /var/data1/dfs/nn /var/data2/dfs/nn /var/data3/dfs/nn
# mkdir -p /var/data1/dfs/dn /var/data2/dfs/dn /var/data3/dfs/dn
# chown -R hdfs:hdfs /var/data1/dfs/dn /var/data2/dfs/dn /var/data3/dfs/dn
# chmod 700 /var/data1/dfs/dn /var/data2/dfs/dn /var/data3/dfs/dn
# mkdir -p /var/data1/dfs/nn/ns /var/data2/dfs/nn/ns /var/data3/dfs/nn/ns
# chown -R hdfs:hdfs /var/data1/dfs/nn/ns /var/data2/dfs/nn/ns /var/data3/dfs/nn/ns
# chmod 700 /var/data1/dfs/nn/ns /var/data2/dfs/nn/ns /var/data3/dfs/nn/ns
# mkdir -p /var/data1/mapred/local /var/data2/mapred/local /var/data3/mapred/local
# chmod -R 1777 /var/data1/mapred/local /var/data2/mapred/local /var/data3/mapred/local
# mkdir -p /var/data1/mapred/tmp
# chmod -R 1777 /var/data1/mapred/tmp
# mkdir -p /var/data1/mapred/history
# chmod -R 755 /var/data1/mapred/history
# chown -R mapred:hadoop /var/data1/mapred /var/data2/mapred /var/data3/mapred
```

設定ファイル core-site.xml を以下のように作成します。(NN, DN)
下記において、hdfs://hd01:54310 は NameNode のホスト名 hd01 を指定しています。

```
# cd /etc/hadoop/conf.cluster
# cp core-site.xml core-site.xml.org
# vi core-site.xml
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hd01:54310</value>
  </property>

  <property>
    <name>hadoop.tmp.dir</name>
    <value>/var/data1/tmp/${user.name}</value>
  </property>

  <property>
    <name>io.file.buffer.size</name>
    <value>65536</value>
  </property>

</configuration>
```

設定ファイル hdfs-site.xml を作成します。 (NN, DN)
下記の場合、パラメーターdfs.replication は 3 を指定していますので、DataNode に生成されるデータのレプリカ数が 3 で構成されます。

```
# pwd
```

```
/etc/hadoop/conf.cluster
# cp hdfs-site.xml hdfs-site.xml.org
# vi hdfs-site.xml
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
  <property>
     <name>dfs.namenode.name.dir</name>
     <value>/var/data1/dfs/nn,/var/data2/dfs/nn,/var/data3/dfs/nn</value>
  </property>

  <property>
     <name>dfs.datanode.data.dir</name>
     <value>/var/data1/dfs/dn,/var/data2/dfs/dn,/var/data3/dfs/dn</value>
  </property>

  <property>
     <name>dfs.permissions.superusergroup</name>
     <value>hadoop</value>
  </property>

  <property>
    <name>dfs.blocksize</name>
    <value>134217728</value>
  </property>

  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>

  <property>
    <name>dfs.namenode.http-address</name>
    <value>hd01:50070</value>
  </property>

  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>hd01:50090</value>
  </property>

  <property>
     <name>dfs.namenode.checkpoint.dir</name>
     <value>/var/data1/dfs/nn/ns,/var/data2/dfs/nn/ns,/var/data3/dfs/nn/ns</value>
  </property>

  <property>
     <name>dfs.namenode.checkpoint.edits.dir</name>
```

```
    <value>/var/data1/dfs/nn/ns,/var/data2/dfs/nn/ns,/var/data3/dfs/nn/ns</value>
  </property>

  <property>
    <name>dfs.hosts</name>
    <value>/etc/hadoop/conf/slave-hosts</value>
  </property>

  <property>
    <name>dfs.hosts.exclude</name>
    <value>/etc/hadoop/conf/slave-exhosts</value>
  </property>

  <property>
    <name>dfs.balance.bandwidthPerSec</name>
    <value>1073741824</value>
  </property>

</configuration>
```

設定ファイル mapred-site.xml を作成します。

```
# pwd
/etc/hadoop/conf.cluster
# cp mapred-site.xml mapred-site.xml.org
# vi mapred-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>hd01:9001</value>
  </property>

  <property>
    <name>mapred.job.tracker.http.address</name>
    <value>hd01:50030</value>
  </property>

  <property>
    <name>mapred.local.dir</name>

<value>/var/data1/mapred/local,/var/data2/mapred/local,/var/data3/mapred/local</value>
  </property>

  <property>
    <name>mapred.temp.dir</name>
    <value>/var/data1/mapred/tmp</value>
  </property>
```

```
<property>
  <name>mapred.child.tmp</name>
  <value>/tmp/${user.name}/tasks</value>
</property>

<property>
  <name>mapred.system.dir</name>
  <value>/mapred/system</value>
</property>

<property>
  <name>mapreduce.jobtracker.staging.root.dir</name>
  <value>/user</value>
</property>

<property>
  <name>mapred.reduce.slowstart.completed.maps</name>
  <value>0.5</value>
</property>

<property>
  <name>mapred.compress.map.output</name>
  <value>false</value>
</property>

<property>
  <name>mapred.tasktracker.map.tasks.maximum</name>
  <value>24</value>
</property>

<property>
  <name>mapred.tasktracker.reduce.tasks.maximum</name>
  <value>24</value>
</property>

<property>
  <name>mapred.map.tasks</name>
  <value>168</value>
</property>

<property>
  <name>mapred.reduce.tasks</name>
  <value>126</value>
</property>

<property>
  <name>mapred.job.reuse.jvm.num.tasks</name>
```

```
  <value>1</value>
</property>

<property>
  <name>mapred.child.java.opts</name>
  <value>-Xmx2048M</value>
</property>

<property>
  <name>mapred.map.child.java.opts</name>
  <value>-Xmx2048M</value>
</property>

<property>
  <name>mapred.reduce.child.java.opts</name>
  <value>-Xmx2048M</value>
</property>

<property>
  <name>tasktracker.http.threads</name>
  <value>40</value>
</property>

<property>
  <name>io.sort.mb</name>
  <value>512</value>
</property>

<property>
  <name>io.sort.factor</name>
  <value>100</value>
</property>

<property>
  <name>mapred.hosts</name>
  <value>/etc/hadoop/conf/slave-hosts</value>
</property>

<property>
  <name>mapred.hosts.exclude</name>
  <value>/etc/hadoop/conf/slave-exhosts</value>
</property>

<property>
  <name>hadoop.job.history.location</name>
  <value>file:///var/data1/mapred/history</value>
</property>
<property>
```

```
    <name>mapred.job.tracker.history.completed.location</name>
    <value>/user/history/done</value>
  </property>

  <property>
    <name>dfs.balance.bandwidthPerSec</name>
    <value>1073741824</value>
  </property>

  <property>
    <name>mapred.compress.map.output</name>
    <value>true</value>
  </property>
  <property>
    <name>mapred.map.output.compression.codec</name>
    <value>org.apache.hadoop.io.compress.SnappyCodec</value>
  </property>

</configuration>
```

HDFS（Hadoop 分散ファイルシステム）をフォーマットします。（NN）
HDFS のフォーマットは NameNode から hdfs コマンドに-format オプションを付与して行います。フォーマット作業はユーザーhdfs で行います。/var/data1/dfs/nn、/var/data2/dfs/nn、/var/data3/dfs/nn をフォーマットするかどうか聞かれるので、y を入力し、Enter キーを押します。

```
# su - hdfs
$ which hdfs
/usr/bin/hdfs
$ hdfs namenode -format
13/07/30 22:43:56 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = hd01/172.16.10.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.0.0-cdh4.3.0
...
...
STARTUP_MSG:                                                    build                 =
file:///data/1/jenkins/workspace/generic-package-rhel64-6-0/topdir/BUILD/hadoop-2.0.0-cdh4
.3.0/src/hadoop-common-project/hadoop-common  -r  48a9315b342ca16de92fcc5be95ae3650629155a;
compiled by 'jenkins' on Mon May 27 19:45:25 PDT 2013
STARTUP_MSG:   java = 1.7.0_25
************************************************************/
13/07/30 22:43:56 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT]
13/07/30 22:43:56 WARN common.Util: Path /var/data1/dfs/nn should be specified as a URI in
configuration files. Please update hdfs configuration.
13/07/30 22:43:56 WARN common.Util: Path /var/data2/dfs/nn should be specified as a URI in
configuration files. Please update hdfs configuration.
```

```
13/07/30 22:43:56 WARN common.Util: Path /var/data3/dfs/nn should be specified as a URI in
configuration files. Please update hdfs configuration.
13/07/30 22:43:56 WARN common.Util: Path /var/data1/dfs/nn should be specified as a URI in
configuration files. Please update hdfs configuration.
13/07/30 22:43:56 WARN common.Util: Path /var/data2/dfs/nn should be specified as a URI in
configuration files. Please update hdfs configuration.
13/07/30 22:43:56 WARN common.Util: Path /var/data3/dfs/nn should be specified as a URI in
configuration files. Please update hdfs configuration.
Formatting using clusterid: CID-de815062-c10e-4dea-a44d-e1c7a8ff5050
13/07/30 22:43:56 INFO util.HostsFileReader: Refreshing hosts (include/exclude) list
13/07/30 22:43:56 INFO util.HostsFileReader: Adding hd02 to the list of hosts from
/etc/hadoop/conf/slave-hosts
13/07/30 22:43:56 INFO util.HostsFileReader: Adding hd03 to the list of hosts from
/etc/hadoop/conf/slave-hosts
13/07/30 22:43:56 INFO util.HostsFileReader: Adding hd04 to the list of hosts from
/etc/hadoop/conf/slave-hosts
13/07/30 22:43:56 INFO util.HostsFileReader: Adding hd05 to the list of hosts from
/etc/hadoop/conf/slave-hosts
13/07/30 22:43:56 INFO util.HostsFileReader: Adding hd06 to the list of hosts from
/etc/hadoop/conf/slave-hosts
13/07/30 22:43:56 INFO util.HostsFileReader: Adding hd07 to the list of hosts from
/etc/hadoop/conf/slave-hosts
13/07/30 22:43:56 INFO util.HostsFileReader: Adding hd08 to the list of hosts from
/etc/hadoop/conf/slave-hosts
13/07/30 22:43:56 INFO blockmanagement.DatanodeManager: dfs.block.invalidate.limit=1000
13/07/30 22:43:56 INFO blockmanagement.BlockManager: dfs.block.access.token.enable=false
13/07/30 22:43:56 INFO blockmanagement.BlockManager: defaultReplication         = 3
13/07/30 22:43:56 INFO blockmanagement.BlockManager: maxReplication             = 512
13/07/30 22:43:56 INFO blockmanagement.BlockManager: minReplication             = 1
13/07/30 22:43:56 INFO blockmanagement.BlockManager: maxReplicationStreams      = 2
13/07/30 22:43:56 INFO blockmanagement.BlockManager: shouldCheckForEnoughRacks  = false
13/07/30 22:43:56 INFO blockmanagement.BlockManager: replicationRecheckInterval = 3000
13/07/30 22:43:56 INFO blockmanagement.BlockManager: encryptDataTransfer        = false
13/07/30 22:43:56 INFO namenode.FSNamesystem: fsOwner             = hdfs (auth:SIMPLE)
13/07/30 22:43:56 INFO namenode.FSNamesystem: supergroup          = hadoop
13/07/30 22:43:56 INFO namenode.FSNamesystem: isPermissionEnabled = true
13/07/30 22:43:56 INFO namenode.FSNamesystem: HA Enabled: false
13/07/30 22:43:56 INFO namenode.FSNamesystem: Append Enabled: true
13/07/30 22:43:57 INFO namenode.NameNode: Caching file names occuring more than 10 times
13/07/30 22:43:57 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct =
0.9990000128746033
13/07/30 22:43:57 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
13/07/30 22:43:57 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension     = 30000
Re-format filesystem in Storage Directory /var/data1/dfs/nn ? (Y or N) y
Re-format filesystem in Storage Directory /var/data2/dfs/nn ? (Y or N) y
Re-format filesystem in Storage Directory /var/data3/dfs/nn ? (Y or N) y
13/07/30 22:44:55 INFO namenode.NNStorage: Storage directory /var/data1/dfs/nn has been
successfully formatted.
13/07/30 22:44:55 INFO namenode.NNStorage: Storage directory /var/data2/dfs/nn has been
successfully formatted.
```

```
13/07/30 22:44:55 INFO namenode.NNStorage: Storage directory /var/data3/dfs/nn has been
successfully formatted.
13/07/30 22:44:55 INFO namenode.FSImage: Saving image file
/var/data1/dfs/nn/current/fsimage.ckpt_0000000000000000000 using no compression
13/07/30 22:44:55 INFO namenode.FSImage: Saving image file
/var/data2/dfs/nn/current/fsimage.ckpt_0000000000000000000 using no compression
13/07/30 22:44:55 INFO namenode.FSImage: Saving image file
/var/data3/dfs/nn/current/fsimage.ckpt_0000000000000000000 using no compression
13/07/30 22:44:55 INFO namenode.FSImage: Image file of size 115 saved in 0 seconds.
13/07/30 22:44:55 INFO namenode.FSImage: Image file of size 115 saved in 0 seconds.
13/07/30 22:44:55 INFO namenode.FSImage: Image file of size 115 saved in 0 seconds.
13/07/30 22:44:55 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid
>= 0
13/07/30 22:44:55 INFO util.ExitUtil: Exiting with status 0
13/07/30 22:44:55 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at hd01/172.16.10.1
************************************************************/
$ whoami
hdfs
$ exit
logout
#
```

## NameNode のサービスを起動(NN)

サービスを起動する前に、ユーザーが適切に作成されているかを確認します。CDH4.3.0 の RPM パッケージを新規インストールした場合は、標準で以下のようになっているはずです。 (NN,DN)

```
# cat /etc/passwd |grep hadoop
hdfs:x:493:487:Hadoop HDFS:/var/lib/hadoop-hdfs:/bin/bash
mapred:x:494:486:Hadoop MapReduce:/usr/lib/hadoop:/bin/bash
```

ユーザーのホームディレクトリが適切に設定されていない場合は、NameNode や JobTracker のサービスの起動に失敗しますので、正しく設定されているかを必ず確認してください。

NameNode 上で、hadoop-hdfs-namenode サービスと/hadoop-0.20-mapreduce-jobtracker サービスを起動します。

```
# chkconfig --list |grep hadoop
hadoop-0.20-mapreduce-jobtracker  0:off   1:off   2:off   3:on    4:on    5:on    6:off
hadoop-hdfs-namenode              0:off   1:off   2:off   3:on    4:on    5:on    6:off
hadoop-hdfs-secondarynamenode     0:off   1:off   2:off   3:on    4:on    5:on    6:off

# service hadoop-hdfs-namenode start
Starting Hadoop namenode:                                  [  OK  ]
starting namenode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-namenode-hd01.out

# service hadoop-hdfs-namenode status
Hadoop namenode is running                                 [  OK  ]
```

```
# /etc/init.d/hadoop-0.20-mapreduce-jobtracker start
Starting Hadoop jobtracker:                                [  OK  ]
starting jobtracker, logging to ¥
 /var/log/hadoop-0.20-mapreduce/hadoop-hadoop-jobtracker-hd01.out

# /etc/init.d/hadoop-0.20-mapreduce-jobtracker status
Hadoop jobtracker is running                               [  OK  ]
```

### DataNode のサービスを起動(DN)

DataNode 上で、hadoop-hdfs-datanode サービスと hadoop-0.20-mapreduce-tasktracker サービスを起動します。

```
# /etc/init.d/hadoop-hdfs-datanode start
Starting Hadoop datanode:                                  [  OK  ]
starting datanode, logging to /var/log/hadoop-hdfs/hadoop-hdfs-datanode-hd02.out

# /etc/init.d/hadoop-hdfs-datanode status
Hadoop datanode is running                                 [  OK  ]

# /etc/init.d/hadoop-0.20-mapreduce-tasktracker start
Starting Hadoop tasktracker:                               [  OK  ]
starting tasktracker, logging to ¥
 /var/log/hadoop-0.20-mapreduce/hadoop-hadoop-tasktracker-hd02.out

# /etc/init.d/hadoop-0.20-mapreduce-tasktracker status
Hadoop tasktracker is running                              [  OK  ]
```

HDFS が利用可能かどうかを確認(NN)
ext3 上に作成したファイルを HDFS 上にコピーできるかどうかをテストします。テストは Hadoop のユーザーkoga で行います。

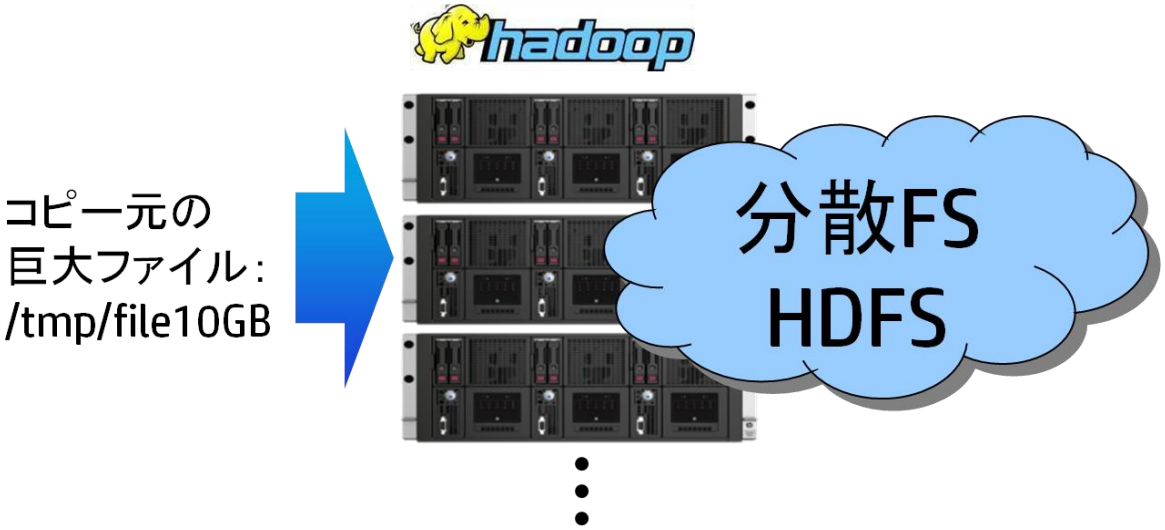


図 2. Hadoop 分散ファイルシステムへのファイルのコピー

Hadoop クラスターにアクセスするクライアントマシンで、hadoop コマンドが利用できるようにするため、クライアントマシン上で必要なパッケージをインストールします。(Client)

```
[root@client ~]# rpm -vhi bigtop-utils-0.6.0+73-1.cdh4.3.0.p0.17.el6.noarch.rpm
[root@client ~]# rpm -vhi zookeeper-3.4.5+19-1.cdh4.3.0.p0.14.el6.noarch.rpm
[root@client ~]# rpm -vhi hadoop-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
[root@client ~]# rpm -vhi bigtop-jsvc-1.0.10-1.cdh4.3.0.p0.14.el6.x86_64.rpm
[root@client ~]# rpm -vhi hadoop-hdfs-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
[root@client ~]# cat /etc/passwd |grep hadoop
hdfs:x:490:489:Hadoop HDFS:/var/lib/hadoop-hdfs:/bin/bash

[root@client ~]# rpm -vhi hadoop-0.20-mapreduce-2.0.0+1357-1.cdh4.3.0.p0.21.el6.x86_64.rpm
[root@client ~]# cat /etc/passwd |grep hadoop
hdfs:x:493:487:Hadoop HDFS:/var/lib/hadoop-hdfs:/bin/bash
mapred:x:494:486:Hadoop MapReduce:/usr/lib/hadoop:/bin/bash
```

NameNode の Hadoop の設定ファイルのディレクトリ/etc/hadoop/conf.cluster をクライアントにコピーします。(Client)

```
[root@client ~]# scp -r hd01:/etc/hadoop/conf.cluster /etc/hadoop/
[root@client ~]# ls -l /etc/hadoop/conf
conf/ conf.empty/ conf.cluster/
[root@client ~]# ls -l /etc/hadoop/conf.cluster
合計 76
-rw-r--r-- 1 root root 5041     2 月  24 15:00 capacity-scheduler.xml
-rw-r--r-- 1 root root  535     2 月  24 15:00 configuration.xsl
-rw-r--r-- 1 root root  178     2 月  24 15:00 core-site.xml
-rw-r--r-- 1 root root 3032     2 月  24 15:00 fair-scheduler.xml
-rw-r--r-- 1 root root 2340     2 月  24 15:00 hadoop-env.sh
```

```
-rw-r--r-- 1 root root 2109     2月 24 15:00 hadoop-metrics.properties
-rw-r--r-- 1 root root 4644     2月 24 15:00 hadoop-policy.xml
-rw-r--r-- 1 root root  178     2月 24 15:00 hdfs-site.xml
-rw-r--r-- 1 root root 4198     2月 24 15:00 log4j.properties
-rw-r--r-- 1 root root 2033     2月 24 15:00 mapred-queue-acls.xml
-rw-r--r-- 1 root root  178     2月 24 15:00 mapred-site.xml
-rw-r--r-- 1 root root   10     2月 24 15:00 masters
-rw-r--r-- 1 root root   10     2月 24 15:00 slaves
-rw-r--r-- 1 root root 1243     2月 24 15:00 ssl-client.xml.example
-rw-r--r-- 1 root root 1195     2月 24 15:00 ssl-server.xml.example
-rw-r--r-- 1 root root  382     2月 24 15:00 taskcontroller.cfg
```

現在の Hadoop の設定情報を alternatives コマンドによって確認します。(Client)

```
[root@client ~]# alternatives --display hadoop-conf
hadoop-conf -ステータスは自動です。
リンクは現在 /etc/hadoop/conf.empty を指しています。
/etc/hadoop/conf.empty - 優先項目 10
現在の「最適」バージョンは /etc/hadoop/conf.empty です
```

利用する設定ファイルを update-alternatives コマンドで選択します。(Client)

```
[root@client ~]# update-alternatives --install /etc/hadoop/conf hadoop-conf ¥
 /etc/hadoop/conf.cluster 20   (実際には 1 行で入力)
```

現在の設定情報を alternatives コマンドによって確認します。(Client)

```
[root@client ~]# alternatives --display hadoop-conf
hadoop-conf -ステータスは自動です。
リンクは現在 /etc/hadoop/conf.cluster を指しています。
/etc/hadoop/conf.empty - 優先項目 10
/etc/hadoop/conf.cluster - 優先項目 20
現在の「最適」バージョンは /etc/hadoop/conf.cluster です。
```

Hadoop クラスターを利用するユーザーをクライアント上に作成します。今回はユーザー名「koga」を追加します。また、HDFS 上に Hadoop ユーザーkoga のホームディレクトリを作成します。(Client)

```
[root@client ~]# useradd koga
[root@client ~]# sudo -u koga hadoop fs -mkdir /user/koga
```

ユーザーの HDFS 上のホームディレクトリ/user/koga の所有者を koga に設定します。(Client)

```
[root@client ~]# sudo -u koga hadoop fs -chown koga /user/koga
```

ユーザーkoga が HDFS 上に持つホームディレクトリを確認します。(Client)

```
[root@client ~]# sudo -u koga hadoop fs -ls -d
Found 1 items
drwxr-xr-x   - koga hadoop          0 2013-07-31 23:42 .
```

HDFS 上に持つホームディレクトリ配下にディレクトリ dir1 を作成してみます。(Client)

```
[root@client ~]# sudo -u koga hadoop fs -ls
Found 1 items
drwxr-xr-x   - koga hadoop          0 2013-07-31 23:46 dir1
```

dd コマンドによりファイルを生成し、クライアントマシン上の/tmp 等にコピーします。テストは/tmp に空き容量が十分確保できている状態で行ってください。以下では、/tmp に生成したファイルサイズが 10GB のファイル file10GB を HDFS 上の/user/koga/dir1 ディレクトリにコピーできるかどうかをテストします。(Client)

```
[root@client ~]# dd if=/dev/zero of=/tmp/file10GB bs=1024M count=10
# ls -lh /tmp/file10GB
-rw-r--r-- 1 root root 10G  7月 30 12:41 /tmp/file10GB

[root@client ~]# sudo -u koga hadoop fs -put /tmp/file10GB /user/koga/dir1/

[root@client ~]# sudo -u koga hadoop fs -ls dir1/
Found 1 items
-rw-r--r--   3 koga hadoop 10737418240 2013-08-01 00:03 dir1/file10GB
```

上記のように、hadoop コマンドに fs -ls オプションを付けて HDFS 上の/user/koga/dir1 ディレクトリの中身を確認し、file10GB が正常にコピーされていることを確認します。(Client)

## HDFS の使用量を確認

Web ブラウザで HDFS の使用量を確認します。Web ブラウザで、Name Node の IP アドレスまたはホスト名に 50070 ポートでアクセスします。

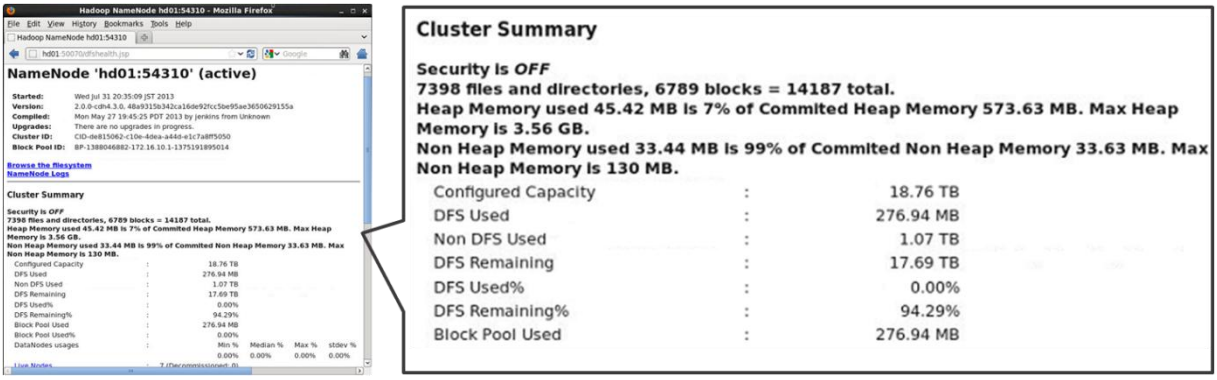


図 3. Hadoop 分散ファイルシステムの使用量を Web ブラウザで確認

## MapReduce のテスト

テキストファイルの単語数をカウントする処理を Hadoop の MapReduce を使ってテストします。テストに利用する対象のテキストファイルは、NameNode の/usr/share/doc/glibc-2.x ディレクトリ以下にある NEWS ファイルです。このテキストファイルに含まれる単語で一番出現頻度の高いものを得るという処理を MapReduce で行います。ファイルの容量が小さい場合はそのメリットが享受できたかどうか分かりにくいですが、巨大あるいは大量のテキストファイルに対して、大量の DataNode を用意し、MapReduce を行うと、従来のシステムに比べ驚異的な処理時間の短縮が図れます。

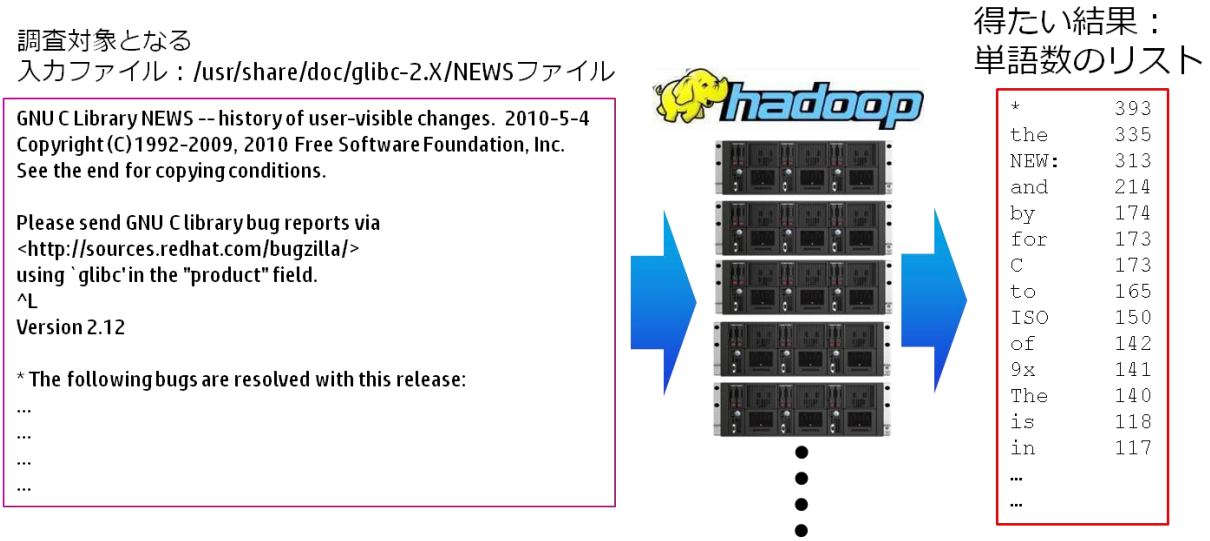


図 4. テキストファイルの単語数をカウントする Map Reduce 処理

まず、Map を行うスクリプト map.py を作成し、非 Hadoop 環境でテストします。スクリプト map.py は、標準入力のテキストファイルの文字列を一行ずつ読み取り、出現する単語を出力します。
今回は Python で記述しましたが、Python 以外の言語で記述しても構いません。

```
#!/usr/bin/env python
import sys
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for w in words:
        print '%s¥t%s' % (w, 1)
```

図 5. Map を行うスクリプト map.py の例

Map 処理を行うスクリプト map.py を非 Hadoop クラスターのテスト環境で実行してみます。入力ファイルは/usr/share/doc/glibc-2.x/NEWS ファイルです。

```
[root@client ~]# pwd
/root
[root@client ~]# chmod +x ./map.py
[root@client ~]# cp /usr/share/doc/glibc-2.12/NEWS /root/
[root@client ~]# ./map.py < ./NEWS > temp.txt
```

Map 処理を行うスクリプト map.py で得られる結果を temp.txt に保存しておきます。Map 処理の結果が記録されている temp.txt を less コマンド等で確認してみます。

```
[root@client ~]# less temp.txt
GNU     1
C       1
Library 1
NEWS    1
--      1
history 1
of      1
user-visible    1
changes.        1
2010-5-4        1
Copyright       1
(C)     1
...
...
...
```

次に reduce を行うスクリプトを作成し、同様に非 Hadoop 環境でテストします。スクリプト red.py は、標準入力のテキスト内の文字列を一行ずつ読み取り同じ文字列をカウントします。Reduce 処理を行うスクリプトも map.py と同様に、Python 以外の言語で記述しても問題ありません。以下は reduce 処理を行うスクリプト red.py の例です。

```
#!/usr/bin/env python
from operator import itemgetter
import sys
wc = {}
for line in sys.stdin:
    line = line.strip()
    w, c = line.split('¥t', 1)
    try:
        c = int(c)
        wc[w] = wc.get(w, 0) + c
    except ValueError:
        pass
s = sorted(wc.items(), key=itemgetter(0))
for w, c in s:
    print '%s¥t%s'% (w, c)
```

図 6. Reduce を行うスクリプト red.py の例

スクリプト red.py は、先ほどの map.py で得られた結果のファイル temp.txt を入力とします。map.py で得られた単語リスト temp.txt を red.py に読み込ませた結果を出現回数（2 列目）でソートします。ソートした結果は out.txt に記録します。すると out.txt に目的の単語の出現回数のリストが得られます。

```
[root@client ~]# chmod +x ./red.py
[root@client ~]# ./red.py < temp.txt |sort -k 2 -nr > out.txt
[root@client ~]# less out.txt
*       393
the     335
NEW:    313
and     214
by      174
for     173
C       173
to      165
ISO     150
of      142
9x      141
The     140
is      118
in      117
a       111
Ulrich  102
now     94
GNU     80
Drepper.        80
Implemented     75
new     74
...
...
```

上記の処理は、スクリプト map.py と red.py をパイプで繋いで処理ができます。一時ファイルを生成せずにパイプで繋いで実行できるか確認します。

```
[root@client ~]# ./map.py < ./NEWS | ./red.py | sort -k 2 -nr > out2.txt

[root@client ~]# less out2.txt
*       393
the     335
NEW:    313
...
...
```

## HDFS を使った MapReduce のテスト

先述の MapReduce 処理を Hadoop 環境で実行してみます。Hadoop クラスターが提供する HDFS 上にファイル NEWS をコピーします。Hadoop ユーザーkoga でクライアントから HDFS へファイルのコピー操作を行います。

```
[root@client ~]# sudo -u koga hadoop fs -put /usr/share/doc/glibc-2.12/NEWS /user/koga/
[root@client ~]# sudo -u koga hadoop fs -ls
Found 2 items
drwx------   - koga hadoop          0 2013-08-05 13:17 .staging
-rw-r--r--   3 koga hadoop      72289 2013-08-05 15:19 NEWS
```

MapReduce を行うスクリプト map.py と red.py を NameNode にコピーします。

```
# scp map.py red.py hd01:/usr/lib/hadoop-0.20-mapreduce/bin
# cd /usr/lib/hadoop-0.20-mapreduce/bin
# chmod 755 map.py
# chmod 755 red.py
# ls -l map.py red.py
-rwxr-xr-x 1 root root 158  8月  5 13:09 2013 map.py
-rwxr-xr-x 1 root root 333  8月  5 13:09 2013 red.py
```

HDFS に配置した NEWS ファイルを Hadoop の MapReduce で分散処理を行います。Hadoop Streaming を利用することで、Java で記述しなくても Python や Perl などのスクリプトで MapReduce を実現できます。

```
# cd /usr/lib/hadoop-0.20-mapreduce
# sudo -u koga hadoop jar contrib/streaming/hadoop-streaming-2.0.0-mr1-cdh4.3.0.jar ¥
-file bin/map.py -mapper bin/map.py -file bin/red.py -reducer bin/red.py ¥
-input /user/koga/NEWS -output OUTDIR001/ （実際には１行で入力）


...
13/08/05 15:25:16 INFO streaming.StreamJob:  map 0%  reduce 0%
13/08/05 15:25:37 INFO streaming.StreamJob:  map 1%  reduce 0%
13/08/05 15:25:38 INFO streaming.StreamJob:  map 2%  reduce 0%
13/08/05 15:25:39 INFO streaming.StreamJob:  map 28%  reduce 0%
13/08/05 15:25:40 INFO streaming.StreamJob:  map 95%  reduce 0%
13/08/05 15:25:41 INFO streaming.StreamJob:  map 99%  reduce 0%
13/08/05 15:25:44 INFO streaming.StreamJob:  map 100%  reduce 0%
13/08/05 15:25:50 INFO streaming.StreamJob:  map 100%  reduce 2%
13/08/05 15:25:51 INFO streaming.StreamJob:  map 100%  reduce 3%
```

```
13/08/05 15:25:52 INFO streaming.StreamJob:  map 100%  reduce 4%
13/08/05 15:25:53 INFO streaming.StreamJob:  map 100%  reduce 5%
13/08/05 15:25:54 INFO streaming.StreamJob:  map 100%  reduce 7%
13/08/05 15:25:55 INFO streaming.StreamJob:  map 100%  reduce 9%
13/08/05 15:25:56 INFO streaming.StreamJob:  map 100%  reduce 10%
13/08/05 15:25:57 INFO streaming.StreamJob:  map 100%  reduce 12%
13/08/05 15:25:58 INFO streaming.StreamJob:  map 100%  reduce 14%
13/08/05 15:26:00 INFO streaming.StreamJob:  map 100%  reduce 18%
13/08/05 15:26:01 INFO streaming.StreamJob:  map 100%  reduce 20%
13/08/05 15:26:02 INFO streaming.StreamJob:  map 100%  reduce 26%
13/08/05 15:26:03 INFO streaming.StreamJob:  map 100%  reduce 31%
13/08/05 15:26:04 INFO streaming.StreamJob:  map 100%  reduce 36%
13/08/05 15:26:05 INFO streaming.StreamJob:  map 100%  reduce 40%
13/08/05 15:26:06 INFO streaming.StreamJob:  map 100%  reduce 43%
13/08/05 15:26:07 INFO streaming.StreamJob:  map 100%  reduce 51%
13/08/05 15:26:08 INFO streaming.StreamJob:  map 100%  reduce 64%
13/08/05 15:26:09 INFO streaming.StreamJob:  map 100%  reduce 85%
13/08/05 15:26:10 INFO streaming.StreamJob:  map 100%  reduce 86%
13/08/05 15:26:24 INFO streaming.StreamJob:  map 100%  reduce 88%
13/08/05 15:26:25 INFO streaming.StreamJob:  map 100%  reduce 98%
13/08/05 15:26:26 INFO streaming.StreamJob:  map 100%  reduce 100%
13/08/05 15:26:30 INFO streaming.StreamJob: Job complete: job_201307312035_0006
13/08/05 15:26:30 INFO streaming.StreamJob: Output: OUTDIR001/
```

出力するディレクトリに OUTDIR001 を指定したので、HDFS 上に OUTDIR001 が生成されていることを確認します。

```
# sudo -u koga hadoop fs -ls
Found 3 items
drwx------   - koga hadoop          0 2013-08-05 15:26 .staging
-rw-r--r--   3 koga hadoop      72289 2013-08-05 15:19 NEWS
drwxr-xr-x   - koga hadoop          0 2013-08-05 15:26 OUTDIR001
```

HDFS に生成された OUTDIR1 ディレクトリ配下に、目的の結果のファイルが格納されているかどうかを確認します。得られた結果は、ファイル名「part-XXXXX」として保存されます。目的のファイルは HDFS 上に保管されていますが、確認作業はクライアントまたは NameNode から行います。

```
# sudo -u koga hadoop fs -ls OUTDIR001 |less
Found 128 items
-rw-r--r--   3 koga hadoop          0 2013-08-05 15:30 OUTDIR001/_SUCCESS
drwxr-xr-x   - koga hadoop          0 2013-08-05 15:29 OUTDIR001/_logs
-rw-r--r--   3 koga hadoop        265 2013-08-05 15:29 OUTDIR001/part-00000
-rw-r--r--   3 koga hadoop        280 2013-08-05 15:29 OUTDIR001/part-00001
-rw-r--r--   3 koga hadoop        347 2013-08-05 15:29 OUTDIR001/part-00002
-rw-r--r--   3 koga hadoop        288 2013-08-05 15:29 OUTDIR001/part-00003
...
...
-rw-r--r--   3 koga hadoop        227 2013-08-05 15:29 OUTDIR001/part-00125
```

結果のファイル part-XXXXX が HDFS 上の OUTDIR001 ディレクトリに生成されていますので、内容を確認するため、ソートします。HDFS 上のテキストファイルを閲覧するには、hadoop fs -cat に HDFS のパスとファイルを指定することで可能です。

```
# sudo -u koga hadoop fs -cat OUTDIR001/part-????? |sort -k 2 -nr |less
*       393
the     335
NEW:    313
and     214
by      174
for     173
C       173
to      165
ISO     150
of      142
...
```

以上