

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА (НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

ФАКУЛЬТЕТ ИНФОРМАТИКИ

КАФЕДРА ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ЛАБОРАТОРНОЙ РАБОТЕ

ТЕМА: «**Метод Оцу**»

Выполнили студенты

Проценко В. И.
Булдыгин Е.Ю.

Группа

6128

26 марта 2013 г.

1. Пороговая бинаризация методом Оцу

Бинарные изображения можно получать из полутоновых изображений посредством пороговой бинаризации. При выполнении этой операции часть пикселей выбирается в качестве пикселей переднего плана, представляющих объекты интереса, а остальные — в качестве фоновых пикселей. Зная распределение значений яркости на данном изображении, некоторые значения можно выбрать в качестве порогов, разделяющих пиксели на группы. В простейшем случае выбирается одно пороговое значение t . Все пиксели с яркостью больше или равной t становятся белыми, а остальные — чёрными.

Для автоматического выбора порога бинаризации было разработано много различных методов. В данной работе реализован метод Оцу. Выбор порога в этом методе основан на минимизации внутригрупповой дисперсии двух групп пикселей, разделяемых оператором пороговой бинаризации. Оцу показал, что минимум внутригрупповой дисперсии обеспечивает максимум межгрупповой дисперсии (квадрат разности между средними значениями групп). Интерпретируя гистограмму как частоту появления конкретного значения яркости, можно записать две величины:

$$p_1(t) = \frac{1}{|H|} \sum_{i=0}^t H_i, \quad p_2(t) = 1 - p_1(t) = \frac{1}{|H|} \sum_{i=t+1}^{255} H_i, \quad |H| = \sum_{i=0}^{255} H_i,$$

где $p_i(t)$ — вероятность класса, получаемая при бинаризации с порогом t . Значение порога в методе Оцу определяется как:

$$t = \arg \max p_1(t)(1 - p_1(t))(\mu_1(t) - \mu_2(t))^2,$$
$$\mu_1(t) = \frac{1}{\sum_{i=0}^t H_i} \sum_{i=0}^t iH_i, \quad \mu_2(t) = \frac{1}{\sum_{i=t+1}^{255} H_i} \sum_{i=t+1}^{255} iH_i.$$

Соответственно алгоритм заключается в простом переборе возможных пороговых значений t с выбором наилучшего, при этом все необходимые величины могут вычисляться рекуррентно.

Таким образом, для реализации пороговой обработки методом Оцу необходимо провести вычисления в три этапа:

- 1) сбор гистограммы яркости;
- 2) расчёт порогового значения t ;
- 3) пороговая бинаризация с полученным порогом.

2. Настройка оборудования

2.1. Настройка одного узла

Для настройки одиночного узла не требуется прилагать огромных усилий. Подробное описание действий по устройству кластера на базе Hadoop можно найти по ссылкам:

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/>

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/>

Дальнейшее описание представляет собой краткий обзор этих статей. Для установки Hadoop необходимо:

- 1) Компьютер с Ubuntu 12.04 LTS;
- 2) Дистрибутив Hadoop 1.0.4;
- 3) Свежая версия Java SDK (в данной работе был использован Java SE 1.7.0_17);
- 4) Настроенная SSH.

В случае установки Hadoop на чистую систему, необходимо установить Java SDK с сайта Oracle, так как предустановленный OpenJDK может не поддерживать некоторые элементы Hadoop. Для начала удаляется OpenJDK и всё что с ним связано:

```
$ sudo apt-get purge openjdk*
```

Необходимый Java SDK можно скачать с сайта (для Ubuntu выбираем .tar.gz):

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

Далее распаковываем скачанный архив и перемещаем содержимое в каталог /usr/lib/jvm/:

```
$ tar -xvf $HOME/Downloads/jdk-7u4-linux-x64.tar.gz
$ mv $HOME/jdk1.7.0_04/ /usr/lib/jvm/
```

Hadoop использует протокол SSH для управления узлами, поэтому необходимо сначала установить SSH-сервер:

```
$ sudo apt-get install ssh
```

Для пользователя создаётся ключ для SSH-соединения (в данном случае пользователь hduser), который соответствует пустому паролю:

```
$ su - hduser
$ ssh-keygen -t rsa -P ""
```

Затем открываем доступ к локальной машине с созданным ключом:

```
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Для проверки запускаем соединение с localhost, при этом он будет помещён в список известных хостов hduser:

```
$ ssh localhost
```

Следующим шагом является отключение IPv6. Для этого открываем файл /etc/sysctl.conf и добавляем в конец файла строки:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

После сохранения, можно проверить результат командой:

```
$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

Если всё было правильно сделано, то увидим 1.

Теперь можно приступить к установке Hadoop. Для этого скачиваем дистрибутив с официального сайта (<http://hadoop.apache.org/>) и извлекаем содержимое в симпатичный нам каталог (например, в /usr/local/hadoop), при этом не стоит забывать о правах доступа (владельцем должен быть hduser и группа hadoop). Процесс можно выполнить командами:

```
$ sudo tar xzf $HOME/Downloads/hadoop-1.0.4.tar.gz
$ sudo mv $HOME/hadoop-1.0.4 /usr/local/hadoop
$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

После этого необходимо обновить \$HOME/.bashrc в соответствии со следующими строками:

```
# Установка переменной, связанной с Hadoop
export HADOOP_HOME=/usr/local/hadoop

# Установка местоположения Java SDK
export JAVA_HOME=/usr/lib/jvm/java-6-sun

# Некоторые удобные псевдонимы и функции
# для работы Hadoop, связанной с командами
unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"

# If you have LZO compression enabled in your
# Hadoop cluster and compress job outputs with
# LZOP (not covered in this tutorial):
# C      onveniently inspect an LZOP compressed
# file from the command
# line; run via:
#
# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
#
# Requires installed 'lzop' command.
#
lzohead () {
    hadoop fs -cat $1 | lzop -dc | head -1000 | less
}

# Добавляем Hadoop bin/ в PATH
export PATH=$PATH:$HADOOP_HOME/bin
```

Теперь можно заняться конфигурированием одиночного узла. Для начала открываем файл /usr/local/hadoop/conf/hadoop-env.sh и заменяем строчку вида

```
# export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

на строчку с соответствующим указанием директории с Java SDK:

```
export JAVA_HOME=/usr/lib/jvm/java1.7.0_17
```

Можно создать отдельный каталог для служебного использования распределённой файловой системой (в нашем случае это будет /app/hadoop/tmp, не забываем при этом про права доступа):

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp
```

Открываем conf/core-site.xml в директории с Hadoop и между тегами <configuration> ... </configuration> добавляем следующие строки:

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>
    A base for other temporary directories.
  </description>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.
    A URI whose scheme and authority determine the
    FileSystem implementation. The uri's scheme
    determines the config property (fs.SCHEME.impl)
    naming the FileSystem implementation class.
    The uri's authority is used to determine the host,
    port, etc. for a filesystem.
  </description>
</property>
```

В файле conf/mapred-site.xml между тегами <configuration> ... </configuration> добавляем:

```
<property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce
    job tracker runs at. If "local", then jobs are
    run in-process as a single map and reduce task.
  </description>
</property>
```

В файле conf/hdfs-site.xml между тегами <configuration> ... </configuration> добавляем (пока что у нас только один узел — фактор репликации равен единице):

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
    The actual number of replications can
    be specified when the file is created.
    The default is used if replication is
    not specified in create time.
  </description>
</property>
```

Теперь всё готово. Перед запуском форматируем HDFS командой из терминала:

```
$ /usr/local/hadoop/bin/hadoop namenode -format
```

Если всё сконфигурировано правильно, то в терминале появятся примерно такие сообщения:

```
hduser@eugeneold-desktop:~$ /usr/local/hadoop/bin/hadoop namenode -format
Warning: $HADOOP_HOME is deprecated.

13/03/26 07:18:07 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = eugeneold-desktop/127.0.1.1
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.0.4
STARTUP_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/b
ranch-1.0 -r 1393290; compiled by 'hortonfo' on Wed Oct 3 05:13:58 UTC 2012
*****/
Re-format filesystem in /app/hadoop/tmp/dfs/name ? (Y or N) Y
13/03/26 07:18:14 INFO util.GSet: VM type = 32-bit
13/03/26 07:18:14 INFO util.GSet: 2% max memory = 19.33375 MB
13/03/26 07:18:14 INFO util.GSet: capacity = 2^22 = 4194304 entries
13/03/26 07:18:14 INFO util.GSet: recommended=4194304, actual=4194304
13/03/26 07:18:15 INFO namenode.FSNamesystem: fsOwner=hduser
13/03/26 07:18:15 INFO namenode.FSNamesystem: supergroup=supergroup
13/03/26 07:18:15 INFO namenode.FSNamesystem: isPermissionEnabled=true
13/03/26 07:18:15 INFO namenode.FSNamesystem: dfs.block.invalidate.limit=100
13/03/26 07:18:15 INFO namenode.FSNamesystem: isAccessTokenEnabled=false accessK
eyUpdateInterval=0 min(s), accessTokenLifetime=0 min(s)
13/03/26 07:18:15 INFO namenode.NameNode: Caching file names occurring more than
10 times
13/03/26 07:18:15 INFO common.Storage: Image file of size 112 saved in 0 seconds
.
13/03/26 07:18:16 INFO common.Storage: Storage directory /app/hadoop/tmp/dfs/nam
e has been successfully formatted.
13/03/26 07:18:16 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at eugeneold-desktop/127.0.1.1
*****/
hduser@eugeneold-desktop:~$
```

Для запуска одноузлового кластера используется скрипт start-all.sh:

```
hduser@eugeneold-desktop:~$ /usr/local/hadoop/bin/start-all.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/libexec/./logs/hadoop-hduser-na
menode-eugeneold-desktop.out
localhost: starting datanode, logging to /usr/local/hadoop/libexec/./logs/hadoo
p-hduser-datanode-eugeneold-desktop.out
localhost: starting secondarynamenode, logging to /usr/local/hadoop/libexec/./l
ogs/hadoop-hduser-secondarynamenode-eugeneold-desktop.out
starting jobtracker, logging to /usr/local/hadoop/libexec/./logs/hadoop-hduser-
jobtracker-eugeneold-desktop.out
localhost: starting tasktracker, logging to /usr/local/hadoop/libexec/./logs/ha
doop-hduser-tasktracker-eugeneold-desktop.out
```

Остановка кластера осуществляется скриптом stop-all.sh:

```
hduser@eugeneold-desktop:~$ /usr/local/hadoop/bin/stop-all.sh
Warning: $HADOOP_HOME is deprecated.

stopping jobtracker
localhost: stopping tasktracker
stopping namenode
localhost: no datanode to stop
localhost: no secondarynamenode to stop
hduser@eugeneold-desktop:~$
```

2.2. Настройка двухузлового кластера

Если в локальной сети есть два одноузловых кластера, настроенных по предыдущему пункту, то для объединения их в один двухузловой. Если компьютеры не являются членами локальной сети, но есть доступ в Интернет, то можно воспользоваться средствами построения VPN, например Hamachi.

Для удобства работы вместо IP-адресов можно использовать символичные значения, для этого добавляем в файл `/etc/hosts` следующие строки (на обеих машинах):

```
192.168.0.1    master
192.168.0.2    slave
```

Для корректного функционирования Hadoop необходимо обеспечить SSH-соединение без ввода пароля. Для этого выполняем команду копирования ключа (создан в предыдущем разделе):

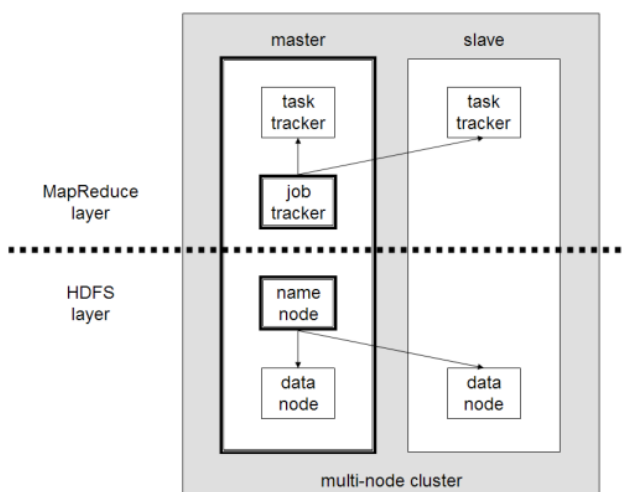
```
hduser@master:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave
```

После этого проверяем соединения master-to-master и master-to-slave, при запросе подтверждаем продолжение соединения:

```
hduser@master:~$ ssh master
The authenticity of host 'master (192.168.0.1)' can't be established.
RSA key fingerprint is 3b:21:b3:c0:21:5c:7c:54:2f:1e:2d:96:79:eb:7f:95.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master' (RSA) to the list of known hosts.
Linux master 2.6.20-16-386 #2 Thu Jun 7 20:16:13 UTC 2007 i686

hduser@master:~$ ssh slave
The authenticity of host 'slave (192.168.0.2)' can't be established.
RSA key fingerprint is 74:d7:61:86:db:86:8f:31:90:9c:68:b0:13:88:52:72.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave' (RSA) to the list of known hosts.
```

На рисунке представлено различие между master и slave. Помимо этого, на master предполагается запуск скриптов `start-dfs.sh` и `start-mapred.sh`, которые запускают необходимых демонов во вторичных узлах.



Для настройки узла-мастера необходимо добавить master в /conf/masters и пару строчек в /conf/slaves:

```
master
slave
```

На всех узлах производим следующие модификации /conf/*-site. Как и прежде, между тегами <configuration> ... </configuration> нужно обновить содержимое в соответствии со следующим текстом:

1) Для conf/core-site.xml на всех машинах:

```
<property>
  <name>fs.default.name</name>
  <value>hdfs://master:54310</value>
  <description>The name of the default file system.
  A URI whose scheme and authority determine
  the FileSystem implementation. The uri's scheme
  determines the config property (fs.SCHEME.impl)
  naming the FileSystem implementation class.
  The uri's authority is used to determine the host,
  port, etc. for a filesystem.
</description>
</property>
```

2) Для conf/mapred-site.xml на всех машинах:

```
<property>
  <name>mapred.job.tracker</name>
  <value>master:54311</value>
  <description>The host and port that the MapReduce
  job tracker runs at. If "local", then jobs are
  run in-process as a single map and reduce task.
</description>
</property>
```

3) Для conf/hdfs-site.xml на всех машинах:

```
<property>
  <name>dfs.replication</name>
  <value>2</value>
  <description>Default block replication.
  The actual number of replications can be specified
  when the file is created. The default is used if
  replication is not specified in create time.
</description>
</property>
```

Теперь всё настроено, можно форматировать HDFS и запускать кластер. На master для форматирования файловой системы вводим команду:

```
hduser@master$ /usr/local/hadoop/bin/hadoop namenode -format
```


Запуск кластера проводится в два этапа:

- 1) Запуск файловой системы:

```
hduser@master$ /usr/local/hadoop/bin/start-dfs.sh
```

- 2) Запуск фоновых программ MapReduce (JobTracker на мастере и TaskTracker на обоих узлах):

```
hduser@master$ /usr/local/hadoop/bin/start-mapred.sh
```

Успешность запуска можно проверить по соответствующим логам в logs/ в директории с Hadoop или же запустить jps:

```
hduser@master$ $JAVA_HOME/bin/jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode
```

На компьютере slave:

```
hduser@slave$ $JAVA_HOME/bin/jps
15183 DataNode
15897 TaskTracker
16284 Jps
```

Остановка кластера также происходит в два этапа:

- 1) Остановка фоновых программ MapReduce:

```
hduser@master$ /usr/local/hadoop/bin/stop-mapred.sh
```

- 2) Остановка файловой системы:

```
hduser@master$ /usr/local/hadoop/bin/stop-dfs.sh
```

Если сделать всё правильно, то получим примерный вывод (как легко заметить, вторичный узел в данном запуске называется `old_slave`, если всё было сделано по инструкции, то вторичный узел должен иметь название `slave`):

```
hduser@euegene-laptop:~$ /usr/local/hadoop/bin/start-dfs.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-na
menode-euegene-laptop.out
master: starting datanode, logging to /usr/local/hadoop/libexec/../logs/hadoop-h
duser-datanode-euegene-laptop.out
old_slave: starting datanode, logging to /usr/local/hadoop/libexec/../logs/hadoo
p-hduser-datanode-eugeneold-desktop.out
master: starting secondarynamenode, logging to /usr/local/hadoop/libexec/../logs
/hadoop-hduser-secondarynamenode-euegene-laptop.out
hduser@euegene-laptop:~$ /usr/local/hadoop/bin/start-mapred.sh
Warning: $HADOOP_HOME is deprecated.

starting jobtracker, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-
jobtracker-euegene-laptop.out
master: starting tasktracker, logging to /usr/local/hadoop/libexec/../logs/hadoo
p-hduser-tasktracker-euegene-laptop.out
old_slave: starting tasktracker, logging to /usr/local/hadoop/libexec/../logs/ha
doo-hduser-tasktracker-eugeneold-desktop.out
```

```
hduser@euegene-laptop:~$ /usr/local/hadoop/bin/stop-mapred.sh
Warning: $HADOOP_HOME is deprecated.

stopping jobtracker
master: stopping tasktracker
old_slave: stopping tasktracker
hduser@euegene-laptop:~$ /usr/local/hadoop/bin/stop-dfs.sh
Warning: $HADOOP_HOME is deprecated.

stopping namenode
master: stopping datanode
old_slave: stopping datanode
master: stopping secondarynamenode
hduser@euegene-laptop:~$
```

3. Описание проекта

Структура проекта

```
/src
/lib
/input
build.xml
buildrun.sh
run.sh
```

Проект поддерживает компиляцию под следующие архитектуры: linux i386, linux amd64, windows x86, windows x64. Скомпилированные версии тестировались в Windows 7 x64 с библиотекой Karmasphere: Hadoop MapReduce 0.20.203 и в Ubuntu (см. Версии использованного программного обеспечения, раздел 4.2) с версией Hadoop 1.0.4.

Для компиляции необходимо установить ant и находясь в папке с проектом запустить его. Например, проект находится в linux в директории /develop/java/project, запускаем консоль, в ней вводим две команды: "cd /develop/java/project" "ant".

3.1. Описание программы

Листинги программы приведены в конце отчёта. За основу был взят код открытого проекта <http://code.google.com/p/hadoop-computer-vision/>, а именно код примера Histogram.java.

Программа состоит из двух фаз. Нахождение порогового значения и применения порога к изображению.

Первая фаза пробегает по изображениям. Метод getSplits() задаёт отображение: одному изображению соответствует один split, файлы являются не делимыми для InputReader. ImageRecordReader делит полученное изображение прямоугольными окнами на более мелкие изображения, теперь представляющие из себя единичные записи.

Map класс подсчитывает частичные гистограммы для каждой записи.

Входные параметры map():

Ключ — имя файла,

Значение — объект типа Image, часть исходного изображения, соответствующая ключу.

Выходные параметры map():

Ключ — имя файла,

Значение — объект типа LongArrayWritable, частичная гистограмма.

Combiner класс подсчитывает итоговую гистограмму для всех значений ключа.

Входные параметры reduce():

Ключ — имя файла,

Значение — список значений типа LongArrayWritable, частичные гистограммы.

Выходные параметры reduce():

Ключ — имя файла,

Значение — объект типа LongArrayWritable, итоговая гистограмма.

Reducer класс подсчитывает итоговую гистограмму для всех значений ключа.

Входные параметры `reduce()`:

Ключ — имя файла,

Значение — объект типа `LongArrayWritable`, итоговая гистограмма.

Выходные параметры `reduce()`:

Ключ — имя файла,

Значение — текстовое значение подсчитанного порога.

Вторая фаза на вход получает те же изображения, что и первая фаза. Путь к каталогу с подсчитанными пороговыми значениями передаётся через конфигурацию, класс `Configuration`, вызовом

```
conf.setStrings("mapreduce.imagerecordreader.threshpath", args[1])
```

где `args[1]` — временная папка. Декомпозиция входного потока изображений такая же, как и в первой фазе.

Map класс находит пороговое значение для данного ключа—имени файла и применяет его к текущей записи.

Входные параметры `map()`:

Ключ — имя файла,

Значение — объект типа `Image`, часть исходного изображения, соответствующая ключу.

Выходные параметры `map()`:

Ключ — имя файла,

Значение — объект типа `Image`, обработанная часть исходного изображения.

Reducer класс объединяет части данного ключа в итоговое крупное изображение.

Входные параметры `reduce()`:

Ключ — имя файла,

Значение — объект типа `Image`, обработанная часть исходного изображения.

Выходные параметры `reduce()`:

Ключ — имя файла,

Значение — объект типа `Image`, обработанное исходное изображение.

3.2. Описание build.xml

Рассмотрим build скрипт.

```
<project default="endpoint"
        name="Create Runnable Jar for Project NewSandCastle"
        basedir=". ">
```

Описываем создание проекта "Create Runnable Jar for Project NewSandCastle текущий каталог - каталог расположения build.xml. Построение будет ждать завершения элемента с названием "endpoint" и всех зависимых элементов.

```
<property name="src" location="src"/>
<property name="build" location="build"/>
<property name="lib" location="lib"/>
<property name="jarname" location="MethodOzu.jar"/>
```

Устанавливаем необходимые переменные.

```
<condition property="isWindows86">
    <and>
        <os family="windows"/>
        <or>
            <equals arg1="{os.arch}" arg2="i386"/>
            <equals arg1="{os.arch}" arg2="x86"/>
        </or>
    </and>
</condition>
<condition property="isWindows64">
    <and>
        <os family="windows"/>
        <or>
            <equals arg1="{os.arch}" arg2="amd64"/>
            <equals arg1="{os.arch}" arg2="x86_64"/>
        </or>
    </and>
</condition>
<condition property="isLinux86">
    <and>
        <os family="unix"/>
        <or>
            <equals arg1="{os.arch}" arg2="i386"/>
            <equals arg1="{os.arch}" arg2="x86"/>
        </or>
    </and>
</condition>
<condition property="isLinux64">
    <and>
        <os family="unix"/>
        <or>
            <equals arg1="{os.arch}" arg2="amd64"/>
            <equals arg1="{os.arch}" arg2="x86_64"/>
        </or>
    </and>
</condition>
```

Определяем целевую платформу. `<os family="unix"/>` возвратит true если текущая система unix. `<condition property="isLinux64">` установит переменную isLinux64 в true, соответственно возвращаемому значению вложенного выражения.

```
<target name="init">
  <tstamp/>
  <mkdir dir="${build}"/>
</target>
```

Элемент построения "init". `<tstamp/>` - текущее время. `<mkdir dir="${build}"/>` создание папки для сохранения скопированных файлов.

```
<target name="compile" depends="init"
  description="compile the source">
  <javac srcdir="${src}"
    destdir="${build}">
    <classpath>
      <fileset dir="${lib}">
        <include name="**/*.jar" />
      </fileset>
    </classpath>
  </javac>
</target>
```

Элемент построения "compile". `depends="init"` выполняется после завершения "init". В `javac` устанавливаются каталоги с исходным кодом, каталог для скомпилированных классов соответственно ранее установленным переменным `src` и `build`. Также указывается переменная `classpath`, указывающая на каталог с библиотеками.

```
<target if="isLinux86" name="create_jar_linux86"
  depends="compile">
  <delete file="${jarname}"/>
  <jar destfile="${jarname}"
    filesetmanifest="mergewithoutmain">
    <manifest>
      <attribute name="Main-Class"
        value="mainpackage.MainClass"/>
    </manifest>
    <zipfileset excludes="META-INF/maven/**"
      src="${lib}/javacv-bin/javacpp.jar"/>
    <zipfileset excludes="META-INF/maven/**"
      src="${lib}/javacv-bin/javacv.jar"/>
    <zipfileset excludes="META-INF/maven/**"
      src="${lib}/jaivacv-bin/javacv-linux-x86.jar"/>
    <zipfileset excludes="META-INF/maven/**"
      src="${lib}/javacv-cppjars/opencv-2.4.4-linux-x86.jar"/>
    <fileset dir="${build}"/>
  </jar>
</target>
```

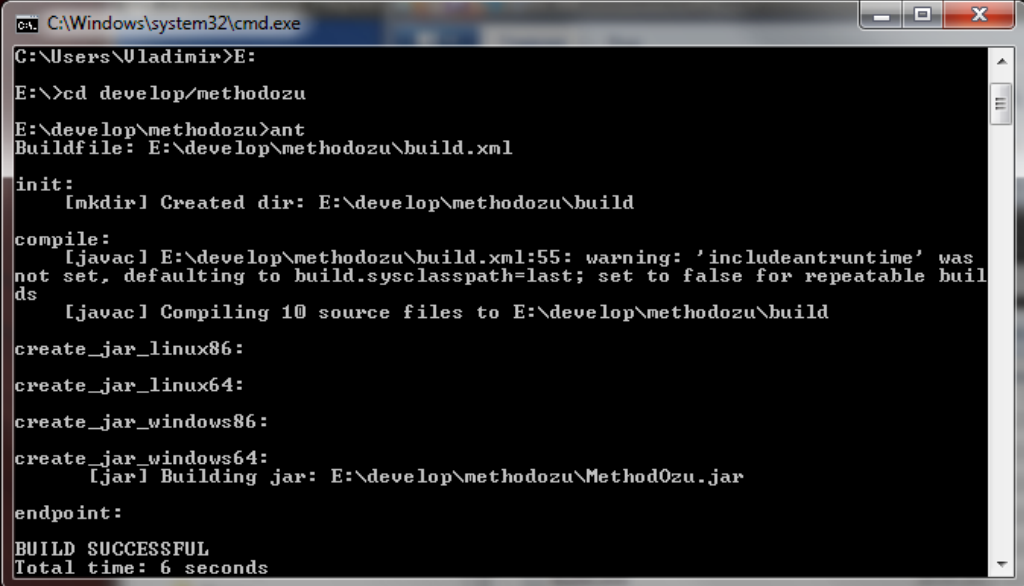
Один из четырёх элементов построения, собирающий скомпилированные файлы в jar. Он также объединяет полученный jar с необходимыми библиотеками.

```
<target name="endpoint" depends="create_jar_linux86 ,
  create_jar_linux64 ,
  create_jar_windows86 ,
  create_jar_windows64"/>
```

Конечная цель проекта построения. Построение на нём завершается.

3.3. Компиляция и запуск под Windows

Чтобы скомпилировать и запустить проект необходимо установить Eclipse, Karmasphere, Apache ant.



```
C:\Windows\system32\cmd.exe
C:\Users\Uladimir>E:
E:\>cd develop\methodozu
E:\develop\methodozu>ant
Buildfile: E:\develop\methodozu\build.xml

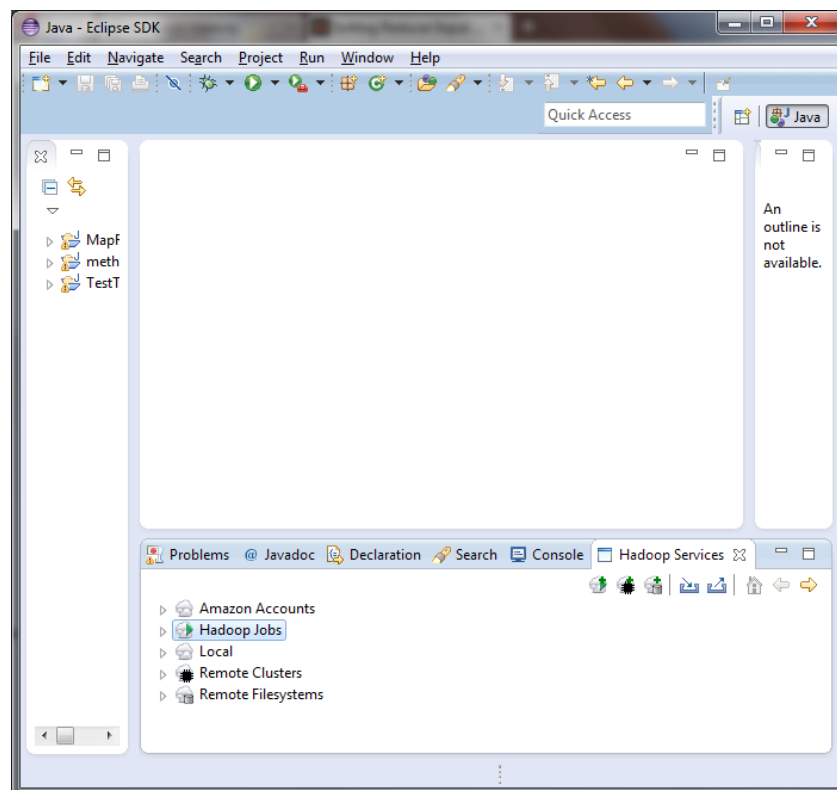
init:
[mkdir] Created dir: E:\develop\methodozu\build

compile:
[javac] E:\develop\methodozu\build.xml:55: warning: 'includeantruntime' was
not set, defaulting to build.sysclasspath=last; set to false for repeatable buil
ds
[javac] Compiling 10 source files to E:\develop\methodozu\build

create_jar_linux86:
create_jar_linux64:
create_jar_windows86:
create_jar_windows64:
[jar] Building jar: E:\develop\methodozu\MethodOzu.jar

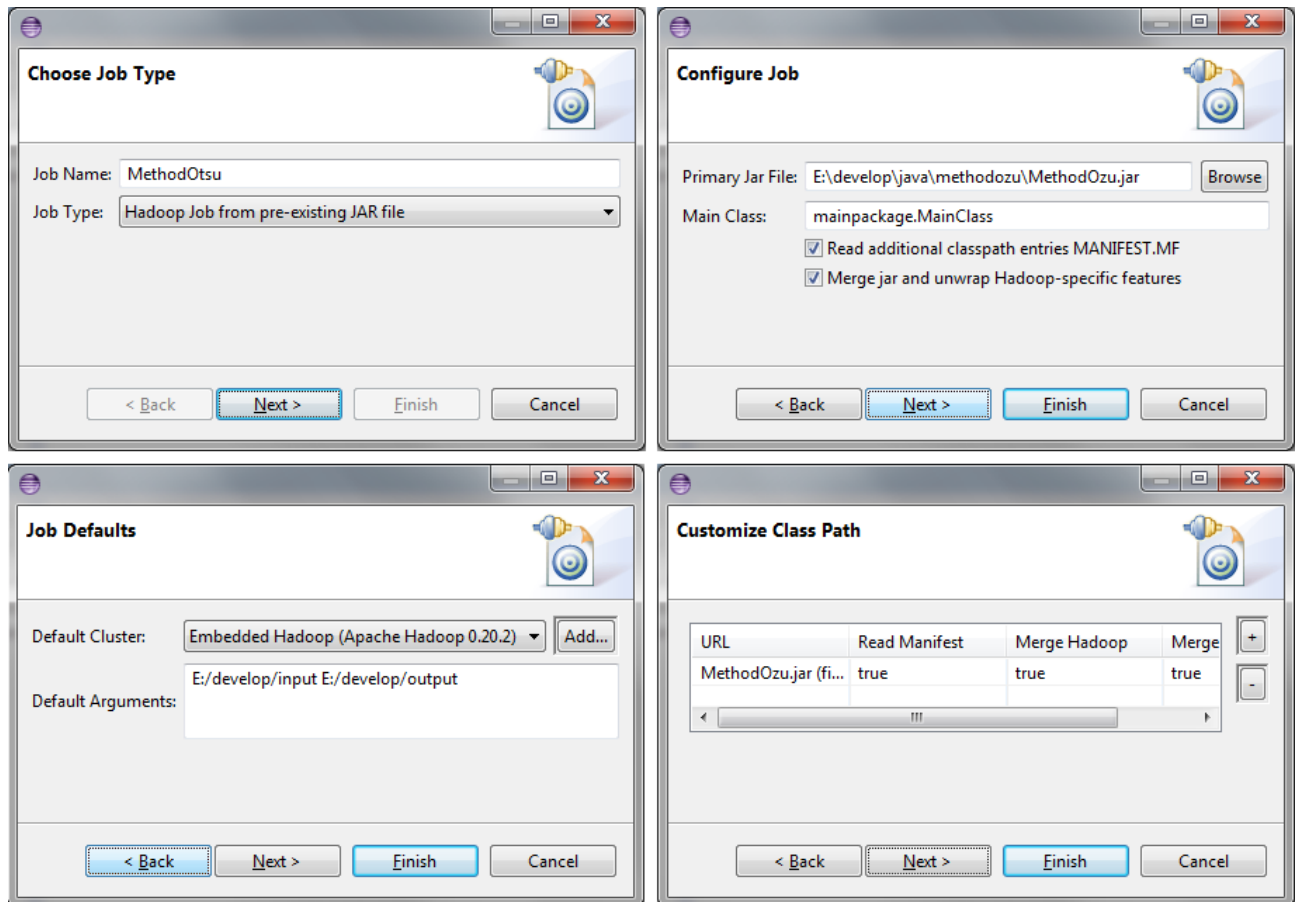
endpoint:
BUILD SUCCESSFUL
Total time: 6 seconds
```

После завершения подготовительных операций в Eclipse должна быть доступна вкладка Hadoop Services.



Вызвав окно консоли и перейдя в папку с проектом запускаем компиляцию командой `ant`. Результатом будет каталог `/build` и `jar` файл "MethodOzu.jar".

Далее работаем с `eclipse`. В `Hadoop Services` создаём `New Job`, выбрав правой клавишей мыши `Hadoop Jobs`. Следуем инструкциям.



Перед запуском необходимо предварительно убедиться, что существует указанный каталог `/input` и не существует `/output` и `/tmp` (может остаться в случае частичного завершения работы программы).

После запуска обработанные изображения будут сохраняться в указанный `/output`.



3.4. Компиляция и запуск под Linux

Чтобы скомпилировать и запустить проект необходимо установить Apache ant, настроить Apache Hadoop (см. Настройка оборудования, раздел 2).

Копируем изображения в папку /input. Убедившись что сервер запущен, а \$HADOOP_HOME указывает на каталог установленного сервера (echo \$HADOOP_HOME), запускаем один из скриптов buildrun.sh (компиляция и запуск), либо run.sh (запуск).

Результат будет находиться в hdfs в каталоге /user/hduser/method-otsu-out.

3.5. Установка ant под Linux

```
sudo apt-get update
sudo apt-get install ant
```

4. Описание среды

4.1. Классификация версий Apache Hadoop

1.0.X — текущая стабильная версия, 1.0 release

1.1.X — текущая бета версия, 1.1 release

2.X.X — текущая альфа версия

0.23.X — то же, что и 2.X.X без NN HA.

0.22.X — модули безопасности

0.20.203.X — старая legacy стабильная версия

0.20.X — старая legacy версия

4.2. Версии использованного программного обеспечения

Apache Ant(TM) version 1.9.0 compiled on March 5 2013,

Hadoop 1.0.4,

Karmasphere Hadoop MapReduce 0.20.203,

Ubuntu 11.10, Ubuntu 12.04 i386,

Windows 7 x64,

Eclipse 4.2.2,

javacv-0.4 (javacv-0.4-bin.zip, javacv-0.4-cppjars.zip).

4.3. Известные проблемы

- 1) Запуск в Karmasphere. Если в пути к /input (и /output) том диска указан заглавной буквой, например E:/develop/input, а не e:/develop/input, то временный каталог /tmp не удалится.

ПРИЛОЖЕНИЕ А

Исходный код функций

MainClass.java

```
1 package mainpackage;
2
3
4
5 import org.apache.hadoop.conf.Configuration;
6 import org.apache.hadoop.util.ToolRunner;
7 import org.codehaus.jackson.map.util.ArrayBuilders;
8
9 // Program consists of two phases. Calculating histogram and applying
   threshold.
10 //
11 // First phase
12 // is run over images. Method getSplits works that split=image, files are
   not splittable.
13 // ImageRecordReader divides source image by rectangular subwindows, so
   this image part is a record.
14 //
15 // Map calculates partial histograms.
16 // Map output:
17 //   key format – <filename> <width of original image> <height of original
   image>
18 //   value – list of histograms of image parts.
19 //
20 // Reduce collects combined pairs of key defining source image file and
   list of histograms of all it's parts.
21 // Each reduce method calculates threshold according to result histogram.
   This threshold would be written to tempdir path
22 // in following format:
23 //   <filename> <width> <height> <threshold> \newline
24 //
25 //
26 // Second phase
27 // similarly is run over the same images. Directory with files containing
   calculated thresholds is passed to algorithm
28 // via Configuration: conf.setStrings("mapreduce.imagerecordreader.
   threshpath", args[1]); , where args[1] is a tempdir,
29 // output path for first map reduce job.
30 // Division on parts remains unchanged.
31 //
32 // Map finds a threshold for given image filename in all files were with
   thresholds, converts image to grayscale and applies threshold.
33 //
34 // Reduce collects parts and for each key stitch all parts together.
35 public class MainClass {
36   // main args: <input path> <output path>
37   public static void main(String[] args) throws Exception {
38     String tempdir = args[0].substring(0, args[0].length()-1);
```

```

39     tempdir = tempdir.substring(0, tempdir.lastIndexOf('/') + 1) + "/tmp";
40     String[] newargs = new String[3];
41     newargs[0] = args[0];
42     newargs[1] = tempdir;
43     newargs[2] = args[1];
44     args[1] = tempdir;
45     int res = ToolRunner.run(new Configuration(), new Histogram(), args);
46     int res2 = ToolRunner.run(new Configuration(), new ThreshApply(),
        newargs);
47     System.exit(res);
48 }
49 }

```

Histogram.java

```

1 package mainpackage;
2
3 import java.io.IOException;
4 import java.nio.ByteBuffer;
5 import java.util.Iterator;
6
7 import org.apache.hadoop.conf.Configuration;
8 import org.apache.hadoop.conf.Configured;
9 import org.apache.hadoop.fs.Path;
10 import org.apache.hadoop.io.LongWritable;
11 import org.apache.hadoop.io.Text;
12 import org.apache.hadoop.io.Writable;
13 import org.apache.hadoop.mapreduce.Job;
14 import org.apache.hadoop.mapreduce.Mapper;
15 import org.apache.hadoop.mapreduce.Reducer;
16 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
17 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
18 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
19 import org.apache.hadoop.util.Tool;
20 import org.apache.hadoop.util.ToolRunner;
21
22 import static com.googlecode.javacv.cpp.opencv_core.*;
23 import static com.googlecode.javacv.cpp.opencv_imgproc.*;
24
25 import edu.vt.io.Image;
26 import edu.vt.io.*;
27 import edu.vt.input.ImageInputFormat;
28
29 public class Histogram extends Configured implements Tool {
30     public static class Map extends
31         Mapper<Text, Image, Text, LongArrayWritable> {
32         private final static LongWritable one = new LongWritable(1);
33
34         @Override
35         public void map(Text key, Image value, Context context)
36             throws IOException, InterruptedException {
37
38             // Convert to gray scale image

```

```

39     IplImage im1 = value.getImage();
40     IplImage im2 = cvCreateImage(cvSize(im1.width(), im1.height()),
41         IPL_DEPTH_8U, 1);
42     cvCvtColor(im1, im2, CV_BGR2GRAY);
43
44     // Initialize histogram array
45     LongWritable[] histogram = new LongWritable[256];
46     for (int i = 0; i < histogram.length; i++) {
47         histogram[i] = new LongWritable();
48     }
49
50     // Compute histogram
51     ByteBuffer buffer = im2.getByteBuffer();
52     while (buffer.hasRemaining()) {
53         int val = buffer.get() + 128;
54         histogram[val].set(histogram[val].get() + 1);
55     }
56
57     context.write(key, new LongArrayWritable(histogram));
58 }
59 }
60
61 public static class Combine extends
62     Reducer<Text, LongArrayWritable, Text, LongArrayWritable> {
63
64     @Override
65     public void reduce(Text key, Iterable<LongArrayWritable> values,
66         Context context) throws IOException, InterruptedException {
67
68         // Initialize histogram array
69         LongWritable[] histogram = new LongWritable[256];
70         for (int i = 0; i < histogram.length; i++) {
71             histogram[i] = new LongWritable();
72         }
73
74         // Sum the parts
75         Iterator<LongArrayWritable> it = values.iterator();
76         while (it.hasNext()) {
77             LongWritable[] part = (LongWritable[]) it.next().toArray();
78             for (int i = 0; i < histogram.length; i++) {
79                 histogram[i].set(histogram[i].get() + part[i].get());
80             }
81         }
82
83         context.write(key, new LongArrayWritable(histogram));
84     }
85 }
86
87 public static class Reduce extends
88     Reducer<Text, LongArrayWritable, Text, Text> {
89
90     @Override
91     public void reduce(Text key, Iterable<LongArrayWritable> values,

```

```

92         Context context) throws IOException, InterruptedException {
93
94         // Initialize histogram array
95         LongWritable[] histogram = new LongWritable[256];
96         for (int i = 0; i < histogram.length; i++) {
97             histogram[i] = new LongWritable();
98         }
99
100        // Sum the parts
101        Iterator<LongArrayWritable> it = values.iterator();
102        while (it.hasNext()) {
103            LongWritable[] part = (LongWritable[]) it.next().toArray();
104            for (int i = 0; i < histogram.length; i++) {
105                histogram[i].set(histogram[i].get() + part[i].get());
106            }
107        }
108
109        int m = 0;
110        int n = 0;
111        for (int t = 0; t < 256; t++) {
112            m += t * histogram[t].get();
113            n += histogram[t].get();
114        }
115
116        float maxSigma = -1;
117        int threshold = 0;
118
119        int alpha1 = 0;
120        int beta1 = 0;
121
122        for (int t = 0; t < 256; t++) {
123            alpha1 += t * histogram[t].get();
124            beta1 += histogram[t].get();
125
126
127            float w1 = (float) beta1 / n;
128
129            float a = (float) alpha1 / beta1 - (float) (m - alpha1)
130                / (n - beta1);
131
132
133            float sigma = w1 * (1 - w1) * a * a;
134
135            if (sigma > maxSigma) {
136                maxSigma = sigma;
137                threshold = t;
138            }
139        }
140
141        context.write(key, new Text(String.valueOf(threshold)));
142    }
143 }
144

```

```

145 public int run(String[] args) throws Exception {
146     // Set various configuration settings
147     Configuration conf = getConf();
148     conf.setInt("mapreduce.imagerecordreader.windowsizepercent", 23);
149     conf.setInt("mapreduce.imagerecordreader.windowoverlappercent", 0);
150
151     // Create job
152     Job job = new Job(conf);
153
154     // Specify various job-specific parameters
155     job.setJarByClass(Histogram.class);
156     job.setJobName("Histogram");
157
158     job.setOutputKeyClass(Text.class);
159     job.setOutputValueClass(Text.class);
160     job.setMapOutputKeyClass(Text.class);
161     job.setMapOutputValueClass(LongArrayWritable.class);
162
163     job.setMapperClass(Map.class);
164     job.setCombinerClass(Combine.class);
165     job.setReducerClass(Reduce.class);
166
167     job.setInputFormatClass(ImageInputFormat.class);
168     job.setOutputFormatClass(TextOutputFormat.class);
169
170     // Set input and output paths
171     FileInputFormat.addInputPath(job, new Path(args[0]));
172     FileOutputFormat.setOutputPath(job, new Path(args[1]));
173
174     return job.waitForCompletion(true) ? 0 : 1;
175 }
176 }

```

ThreshApply.java

```

1 package mainpackage;
2
3 import java.io.BufferedReader;
4 import java.io.File;
5 import java.io.FileReader;
6 import java.io.IOException;
7 import java.io.InputStreamReader;
8 import java.nio.ByteBuffer;
9 import java.util.ArrayList;
10 import java.util.Iterator;
11 import java.util.List;
12 import java.util.StringTokenizer;
13
14 import org.apache.commons.lang.StringUtils;
15 import org.apache.hadoop.conf.Configuration;
16 import org.apache.hadoop.conf.Configured;
17 import org.apache.hadoop.fs.Path;
18 import org.apache.hadoop.io.LongWritable;

```



```

19 import org.apache.hadoop.io.Text;
20 import org.apache.hadoop.io.Writable;
21 import org.apache.hadoop.mapred.InvalidInputException;
22 import org.apache.hadoop.mapreduce.Job;
23 import org.apache.hadoop.mapreduce.Mapper;
24 import org.apache.hadoop.mapreduce.Reducer;
25 import org.apache.hadoop.mapreduce.Reducer.Context;
26 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
27 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
28 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
29 import org.apache.hadoop.fs.FileSystem;
30 import org.apache.hadoop.fs.FileStatus;
31 import org.apache.hadoop.util.Tool;
32 import org.apache.hadoop.util.ToolRunner;
33 import java.util.logging.Logger;
34 import java.util.logging.Level;
35
36 import com.googlecode.javacv.cpp.opencv_core.CvRect;
37 import com.googlecode.javacv.cpp.opencv_core.IplImage;
38
39 import static com.googlecode.javacv.cpp.opencv_core.*;
40 import static com.googlecode.javacv.cpp.opencv_imgproc.*;
41
42 import edu.vt.io.Image;
43 import edu.vt.io.LongArrayWritable;
44 import edu.vt.io.WindowInfo;
45 import edu.vt.input.ImageInputFormat;
46 import edu.vt.output.ImageOutputFormat;
47
48 public class ThreshApply extends Configured implements Tool {
49     public static class Map extends Mapper<Text, Image, Text, Image> {
50
51         private final static LongWritable one = new LongWritable(1);
52         private static Logger logger = Logger.getLogger(ThreshApply.Map.class
53             .getName());
54         private boolean debug = false;
55
56         @Override
57         public void map(Text key, Image value, Context context)
58             throws IOException, InterruptedException {
59             Configuration conf = context.getConfiguration();
60             debug = conf.getBoolean("mapreduce.debug", true);
61
62             if (debug)
63                 logger.log(java.util.logging.Level.INFO,
64                     "VLPR ***** MAP " + key.toString());
65
66             String width = "", height = "", record = "";
67             int threshold = -1;
68
69             // image parameters record format: <filename> threshold
70             record = getImageParameters(key, conf);
71             StringTokenizer tok = new StringTokenizer(record);

```

```

72     tok.nextToken();
73     threshold = Integer.valueOf(tok.nextToken());
74
75     if (debug)
76         logger.log(java.util.logging.Level.INFO, "width " + width
77             + " height " + height);
78
79     // Getting image piece.
80     IplImage imgray = value.getImage();
81
82     cvThreshold(imgray, imgray, threshold, 255, CV_THRESH_BINARY);
83
84     context.write(key, new Image(imgray, value.getWindow()));
85 }
86
87 private String getImageParameters(Text key, Configuration conf)
88     throws IOException {
89     List<IOException> errors = new ArrayList<IOException>();
90     // Threshold preliminaries
91     Path threshdir = new Path(
92         conf.get("mapreduce.imagerecorder.threshpath"));
93
94     if (debug)
95         logger.log(java.util.logging.Level.INFO, "VLPR getImageParameters
96             threshdir " + threshdir.toString());
97
98     // Scan directory with calculated thresholds. Look through files
99     // and
100    // try to find threshold
101    // corresponding to source image file name.
102    FileSystem fs = threshdir.getFileSystem(conf);
103    FileStatus[] matches = fs.globStatus(threshdir);
104    if (matches == null) {
105        errors.add(new IOException("Input path does not exist: " +
106            threshdir.toString()));
107    } else if (matches.length == 0) {
108        errors.add(new IOException("Input Pattern " + threshdir.toString()
109            + " matches 0 files"));
110    } else {
111        if (matches.length != 1) {
112            errors.add(new IOException("More than 1 directory for " +
113                threshdir.toString()));
114        }
115
116        FileStatus globStat = matches[0];
117        for (FileStatus stat : fs.listStatus(globStat.getPath())) {
118            if (!stat.isDirectory()) {
119                BufferedReader br = new BufferedReader(new InputStreamReader(
120                    fs.open(stat.getPath())));
121                String line;
122
123                if (debug)
124                    logger.log(java.util.logging.Level.INFO, "VLPR

```

```

        getImageParameters stat " + stat.getPath().toString());
118
119        while ((line = br.readLine()) != null) {
120            if (debug) {
121                logger.log(java.util.logging.Level.INFO, "VLPR
                    getImageParameters key " + key.toString());
122                logger.log(java.util.logging.Level.INFO, "VLPR
                    getImageParameters key " + line);
123            }
124
125            if (StringUtils.contains(line, key.toString())) {
126                return line;
127            }
128        }
129    }
130 }
131 }
132
133 if (!errors.isEmpty()) {
134     throw new InvalidInputException(errors);
135 } else {
136     errors.add(new IOException("No record for image key " + key.
        toString()));
137     throw new InvalidInputException(errors);
138 }
139 }
140 }
141
142 public static class Reduce extends Reducer<Text, Image, Text, Image> {
143
144     private static Logger logger = Logger
145         .getLogger(ThreshApply.Reduce.class.getName());
146     private boolean debug = false;
147     private int currentSplit;
148
149     @Override
150     public void reduce(Text key, Iterable<Image> values, Context context)
151         throws IOException, InterruptedException {
152         debug = context.getConfiguration().getBoolean("mapreduce.debug",
153             true);
154
155         if (debug)
156             logger.log(java.util.logging.Level.INFO,
157                 "VLPR ***** REDUCE key: "
158                 + key.toString());
159         Image image;
160         IplImage bigimage = null;
161         IplImage imagepart = null;
162
163         boolean first = true;
164         Iterator it = values.iterator();
165         while (it.hasNext()) {
166             image = (Image)it.next();

```

```

167     imagepart = image.getImage();
168     WindowInfo window = image.getWindow();
169     if (first) {
170         logger.log(java.util.logging.Level.INFO, "VLPR before cvCreate
            parentWidth:" + window.getParentWidth() + " parenHeight:" +
            window.getParentHeight());
171         bigimage = cvCreateImage(new CvSize(window.getParentWidth(),
            window.getParentHeight()), IPL_DEPTH_8U, 1);
172         first = false;
173     }
174
175     logger.log(java.util.logging.Level.INFO, "VLPR imagechannels " +
        imagepart.nChannels() + " imagedepth " + imagepart.depth());
176     CvRect roi = window.computeROI();
177
178     if (debug) {
179         logger.log(java.util.logging.Level.INFO, "VLPR imagechannels "
            + imagepart.nChannels() + " imagedepth " + imagepart.depth()
            );
180         logger.log(java.util.logging.Level.INFO, "VLPR width " + image
            .getWidth());
181         logger.log(java.util.logging.Level.INFO, "VLPR height " + image
            .getHeight());
182         logger.log(java.util.logging.Level.INFO, "VLPR roi w: " + roi.
            width() + " h: " + roi.height() + " x: " + roi.x() + " y:" +
            roi.y());
183     }
184
185     cvSetImageROI(bigimage, roi);
186
187     // copy sub-image
188     cvCopy(imagepart, bigimage, null);
189     cvResetImageROI(bigimage);
190 }
191 context.write(key, new Image(bigimage));
192 }
193 }
194
195 public int run(String[] args) throws Exception {
196     // Set various configuration settings
197     Configuration conf = getConf();
198     conf.setInt("mapreduce.imagerecordreader.windowsizepercent", 23);
199     conf.setInt("mapreduce.imagerecordreader.windowoverlapperpercent", 0);
200     conf.setStrings("mapreduce.imagerecordreader.threshpath", args[1]);
201     conf.setBoolean("mapreduce.debug", true);
202     conf.setInt("mapreduce.imagerecordreader.iscolor", 0);
203
204     // Create job
205     Job job = new Job(conf);
206
207     // Specify various job-specific parameters
208     job.setJarByClass(ThreshApply.class);
209     job.setJobName("ThreshApply");

```

```

210
211     job.setOutputKeyClass( Text.class );
212     job.setOutputValueClass( Image.class );
213
214     job.setMapperClass( Map.class );
215     job.setReducerClass( Reduce.class );
216     // job.setNumReduceTasks(0);
217
218     job.setInputFormatClass( ImageInputFormat.class );
219     job.setOutputFormatClass( ImageOutputFormat.class );
220
221     // Set input and output paths
222     FileInputFormat.addInputPath( job , new Path( args[0] ) );
223     FileOutputFormat.setOutputPath( job , new Path( args[2] ) );
224
225     int ok = job.waitForCompletion( true ) ? 0 : 1;
226     // Optional
227     Path tmppath = new Path( args[1] );
228     tmppath.getFileSystem( conf ).delete( tmppath , true );
229     return ok;
230 }
231 }

```

ImageInputFormat.java

```

1 package edu.vt.input;
2
3 import java.io.IOException;
4
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.InputSplit;
8 import org.apache.hadoop.mapreduce.JobContext;
9 import org.apache.hadoop.mapreduce.RecordReader;
10 import org.apache.hadoop.mapreduce.TaskAttemptContext;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12
13 import edu.vt.io.Image;
14
15 public class ImageInputFormat extends FileInputFormat<Text, Image> {
16
17     @Override
18     public RecordReader<Text, Image> createRecordReader( InputSplit split ,
19         TaskAttemptContext context ) throws IOException ,
20         InterruptedException {
21         return new ImageRecordReader();
22     }
23
24     @Override
25     protected boolean isSplittable( JobContext context , Path file ) {
26         return false;
27     }
28 }

```

ImageRecordReader.java

```
1 package edu.vt.input;
2
3 import java.io.FileReader;
4 import java.io.IOException;
5 import java.io.File;
6 import java.io.BufferedReader;
7 import java.util.logging.Logger;
8 import java.util.logging.Level;
9
10 import org.apache.commons.lang.StringUtils;
11 import org.apache.commons.logging.Log;
12 import org.apache.commons.logging.LogFactory;
13
14 import org.apache.hadoop.conf.Configuration;
15 import org.apache.hadoop.fs.FSDataInputStream;
16 import org.apache.hadoop.fs.FileSystem;
17 import org.apache.hadoop.fs.Path;
18 import org.apache.hadoop.io.Text;
19 import org.apache.hadoop.mapreduce.InputSplit;
20 import org.apache.hadoop.mapreduce.RecordReader;
21 import org.apache.hadoop.mapreduce.TaskAttemptContext;
22 import org.apache.hadoop.mapreduce.lib.input.FileSplit;
23
24 import com.googlecode.javacpp.BytePointer;
25
26 import edu.vt.io.Image;
27 import edu.vt.io.WindowInfo;
28
29 import static com.googlecode.javacv.cpp.opencv_core.*;
30 import static com.googlecode.javacv.cpp.opencv_highgui.*;
31
32 public class ImageRecordReader extends RecordReader<Text, Image> {
33
34     private static final Log LOG = LogFactory.getLog(ImageRecordReader.
35         class);
36     private static final Logger logger = Logger.getLogger(ImageRecordReader
37         .class.getName());
38     // Image information
39     private String fileName = null;
40     private Image image = null;
41
42     // Key/Value pair
43     private Text key = null;
44     private Image value = null;
45
46     // Configuration parameters
47     // By default use percentage for splitting
48     boolean byPixel = false;
49     int sizePercent = 0;
50     int sizePixel = 0;
51     int borderPixel = 0;
```

```

50  int iscolor = -1;
51
52  // splits based on configuration parameters
53  int totalXSplits = 0;
54  int totalYSplits = 0;
55  int xSplitPixels = 0;
56  int ySplitPixels = 0;
57
58  // Current split
59  int currentSplit = 0;
60
61  @Override
62  public void close() throws IOException {
63
64  }
65
66  @Override
67  public Text getCurrentKey() throws IOException, InterruptedException {
68
69      return key;
70  }
71
72  @Override
73  public Image getCurrentValue() throws IOException, InterruptedException
74      {
75      return value;
76  }
77
78  @Override
79  public float getProgress() throws IOException, InterruptedException {
80
81      return (float) (totalXSplits * totalYSplits) / (float) currentSplit;
82  }
83
84  @Override
85  public void initialize(InputSplit genericSplit, TaskAttemptContext
86      context)
87      throws IOException, InterruptedException {
88      // Get file split
89      FileSplit split = (FileSplit) genericSplit;
90      Configuration conf = context.getConfiguration();
91
92      // Read configuration parameters
93      getConfig(conf);
94
95      // Open the file
96      Path file = split.getPath();
97      FileSystem fs = file.getFileSystem(conf);
98      FSDataInputStream fileIn = fs.open(split.getPath());
99
100     // Read file and decode image
101     byte[] b = new byte[fileIn.available()];

```

```

101     fileIn.readFully(b);
102     image = new Image(cvDecodeImage(
103         cvMat(1, b.length, CV_8UC1, new BytePointer(b)), iscolor));
104
105     // Get filename to use as key
106     fileName = split.getPath().getName().toString();
107
108     // Calculate the number of splits
109     calculateSplitInfo();
110     currentSplit = 0;
111
112
113 }
114
115 // Was modified
116 @Override
117 public boolean nextKeyValue() throws IOException, InterruptedException
118 {
119     if (currentSplit < (totalXSplits * totalYSplits) && fileName != null)
120     {
121         key = new Text(fileName);
122
123         if (totalXSplits * totalYSplits == 1) {
124             value = image;
125         } else {
126             value = getSubWindow();
127         }
128
129         currentSplit += 1;
130         return true;
131     }
132     return false;
133 }
134
135 private Image getSubWindow() {
136     WindowInfo window = createWindow();
137     CvRect roi = window.computeROI();
138
139     // sets the ROI
140     IplImage img1 = image.getImage();
141     cvSetImageROI(img1, roi);
142
143     // create destination image
144     IplImage img2 = cvCreateImage(cvSize(roi.width(), roi.height()),
145         img1.depth(), img1.nChannels());
146
147     // copy sub-image
148     cvCopy(img1, img2, null);
149
150     // reset the ROI
151     cvResetImageROI(img1);

```



```

152
153     return new Image(img2, window);
154 }
155
156 private void getConfig(Configuration conf) {
157     // Ensure that value is not negative
158     borderPixel = conf.getInt("mapreduce.imagerecordreader.borderPixel",
159         0);
160     if (borderPixel < 0) {
161         borderPixel = 0;
162     }
163     // Ensure that percentage is between 0 and 100
164     sizePercent = conf.getInt(
165         "mapreduce.imagerecordreader.windowsizepercent", 100);
166     if (sizePercent < 0 || sizePercent > 100) {
167         sizePercent = 100;
168     }
169
170     // Ensure that value is not negative
171     sizePixel = conf.getInt("mapreduce.imagerecordreader.windowsizepixel"
172         , Integer.MAX_VALUE);
173     if (sizePixel < 0) {
174         sizePixel = 0;
175     }
176
177     iscolor = conf.getInt("mapreduce.imagerecordreader.iscolor", -1);
178
179     byPixel = conf.getBoolean("mapreduce.imagerecordreader.windowbypixel"
180         , false);
181 }
182
183 private WindowInfo createWindow() {
184     WindowInfo window = new WindowInfo();
185
186     // Get current window
187     int x = currentSplit % totalXSplits;
188     int y = currentSplit / totalYSplits;
189
190     int width = xSplitPixels;
191     int height = ySplitPixels;
192
193     // Deal with partial windows
194     if (x * xSplitPixels + width > image.getWidth()) {
195         width = image.getWidth() - x * xSplitPixels;
196     }
197     if (y * ySplitPixels + height > image.getHeight()) {
198         height = image.getHeight() - y * ySplitPixels;
199     }
200
201     window.setParentInfo(x * xSplitPixels, y * ySplitPixels,

```

```

202         image.getHeight(), image.getWidth());
203     window.setWindowSize(height, width);
204
205     // Calculate borders
206     int top = 0;
207     int bottom = 0;
208     int left = 0;
209     int right = 0;
210
211     if (window.getParentXOffset() > borderPixel) {
212         left = borderPixel;
213     }
214     if (window.getParentYOffset() > borderPixel) {
215         top = borderPixel;
216     }
217     if (window.getParentXOffset() + borderPixel + window.getWidth() <
218         window
219         .getParentWidth()) {
220         right = borderPixel;
221     }
222     if (window.getParentYOffset() + borderPixel + window.getHeight() <
223         window
224         .getParentHeight()) {
225         bottom = borderPixel;
226     }
227
228     window.setBorder(top, bottom, left, right);
229     return window;
230 }
231
232 private void calculateSplitInfo() {
233     if (byPixel) {
234         xSplitPixels = sizePixel;
235         ySplitPixels = sizePixel;
236         totalXSplits = (int) Math.ceil(image.getWidth()
237             / Math.min(xSplitPixels, image.getWidth()));
238         totalYSplits = (int) Math.ceil(image.getHeight()
239             / Math.min(ySplitPixels, image.getHeight()));
240     } else {
241         xSplitPixels = (int) (image.getWidth() * (sizePercent / 100.0));
242         ySplitPixels = (int) (image.getHeight() * (sizePercent / 100.0));
243         totalXSplits = (int) Math.ceil(image.getWidth()
244             / (double) Math.min(xSplitPixels, image.getWidth()));
245         totalYSplits = (int) Math.ceil(image.getHeight()
246             / (double) Math.min(ySplitPixels, image.getHeight()));
247     }
248 }

```

LongArrayWritable.java

```

1 package edu.vt.io;
2

```

```

3 import org.apache.hadoop.io.ArrayWritable;
4 import org.apache.hadoop.io.LongWritable;
5
6 public class LongArrayWritable extends ArrayWritable {
7     public LongArrayWritable() {
8         super(LongWritable.class);
9     }
10
11     public LongArrayWritable(LongWritable[] values) {
12         super(LongWritable.class, values);
13     }
14
15     public String toString(){
16         String [] strings = toStrings();
17         String str = "(" + strings.length + ")[";
18         for (int i = 0; i < strings.length; i++) {
19             str += strings[i] + " ";
20         }
21         str += "];";
22         return str;
23     }
24 }

```

Image.java

```

1 package edu.vt.io;
2
3 import java.io.DataInput;
4 import java.io.DataOutput;
5 import java.io.IOException;
6 import java.nio.ByteBuffer;
7
8 import org.apache.commons.logging.Log;
9 import org.apache.commons.logging.LogFactory;
10
11 import org.apache.hadoop.io.Writable;
12 import org.apache.hadoop.io.WritableUtils;
13
14 import com.googlecode.javacpp.BytePointer;
15 import com.googlecode.javacv.cpp.opencv_core.IplImage;
16
17 import static com.googlecode.javacv.cpp.opencv_core.*;
18 import static com.googlecode.javacv.cpp.opencv_highgui.*;
19
20 public class Image implements Writable {
21
22     private static final Log LOG = LogFactory.getLog(Image.class);
23
24     // IPL image
25     private IplImage image = null;
26     private WindowInfo window = null;
27
28     public Image() {

```

```

29  }
30
31  // Create Image from IplImage
32  public Image(IplImage image) {
33      this.image = image;
34      this.window = new WindowInfo();
35  }
36
37  // Create empty Image
38  public Image(int height, int width, int depth, int nChannels){
39      this.image = cvCreateImage(cvSize(width, height), depth, nChannels);
40      this.window = new WindowInfo();
41  }
42
43  // Create Image from IplImage and IplROI
44  public Image(IplImage image, WindowInfo window) {
45      this.image = image;
46      this.window = window;
47  }
48
49  public IplImage getImage() {
50      return image;
51  }
52
53  // get window where image came from
54  public WindowInfo getWindow() {
55      return window;
56  }
57
58  // Pixel depth in bits
59  // PL_DEPTH_8U – Unsigned 8-bit integer
60  // IPL_DEPTH_8S – Signed 8-bit integer
61  // IPL_DEPTH_16U – Unsigned 16-bit integer
62  // IPL_DEPTH_16S – Signed 16-bit integer
63  // IPL_DEPTH_32S – Signed 32-bit integer
64  // IPL_DEPTH_32F – Single-precision floating point
65  // IPL_DEPTH_64F – Double-precision floating point
66  public int getDepth() {
67      return image.depth();
68  }
69
70  // Number of channels.
71  public int getNumChannel() {
72      return image.nChannels();
73  }
74
75  // Image height in pixels
76  public int getHeight() {
77      return image.height();
78  }
79
80  // Image width in pixels
81  public int getWidth() {

```

```

82     return image.width();
83 }
84
85 // The size of an aligned image row, in bytes
86 public int getWidthStep() {
87     return image.widthStep();
88 }
89
90 // Image data size in bytes.
91 public int getImageSize() {
92     return image.imageSize();
93 }
94
95 // Copies one image into current image using
96 // information contained in WindowInfo struct
97 public void insertImage(Image sourceImage){
98     IplImage img1 = this.image;
99     IplImage img2 = sourceImage.getImage();
100    WindowInfo win = sourceImage.getWindow();
101
102    // set the ROI on destination image
103    if(win.isParentInfoValid()){
104        cvSetImageROI(img1, cvRect(win.getParentXOffset(), win.
105            getParentYOffset(), win.getWidth(), win.getHeight()));
106    }
107    // set the ROI on source image
108    if(win.isBorderValid()){
109        cvSetImageROI(img2, cvRect(win.getBorderLeft(), win.getBorderTop(),
110            win.getWidth(), win.getHeight()));
111    }
112
113    // copy sub-image
114    cvCopy(img2, img1, null);
115
116    // reset the ROI
117    cvResetImageROI(img1);
118 }
119
120 @Override
121 public void readFields(DataInput in) throws IOException {
122    // Read image information
123    int height = WritableUtils.readVInt(in);
124    int width = WritableUtils.readVInt(in);
125    int depth = WritableUtils.readVInt(in);
126    int nChannels = WritableUtils.readVInt(in);
127    int imageSize = WritableUtils.readVInt(in);
128
129    // Read window information
130    int windowXOffest = WritableUtils.readVInt(in);
131    int windowYOffest = WritableUtils.readVInt(in);
132    int windowHeight = WritableUtils.readVInt(in);
133    int windowWidth = WritableUtils.readVInt(in);

```

```

133     int top = WritableUtils.readVInt(in);
134     int bottom = WritableUtils.readVInt(in);
135     int left = WritableUtils.readVInt(in);
136     int right = WritableUtils.readVInt(in);
137
138     int h = WritableUtils.readVInt(in);
139     int w = WritableUtils.readVInt(in);
140
141     window = new WindowInfo();
142     window.setParentInfo(windowXOffset, windowYOffset, windowHeight,
143         windowWidth);
144     window.setBorder(top, bottom, left, right);
145     window.setWindowSize(h, w);
146
147     // Read image bytes
148     byte[] bytes = new byte[imageSize];
149     in.readFully(bytes, 0, imageSize);
150
151     image = cvCreateImage(cvSize(width, height), depth, nChannels);
152     image.imageData(new BytePointer(bytes));
153 }
154
155 @Override
156 public void write(DataOutput out) throws IOException {
157     // Write image information
158     WritableUtils.writeVInt(out, image.height());
159     WritableUtils.writeVInt(out, image.width());
160     WritableUtils.writeVInt(out, image.depth());
161     WritableUtils.writeVInt(out, image.nChannels());
162     WritableUtils.writeVInt(out, image.imageSize());
163
164     // Write window information
165     WritableUtils.writeVInt(out, window.getParentXOffset());
166     WritableUtils.writeVInt(out, window.getParentYOffset());
167     WritableUtils.writeVInt(out, window.getParentHeight());
168     WritableUtils.writeVInt(out, window.getParentWidth());
169
170     WritableUtils.writeVInt(out, window.getBorderTop());
171     WritableUtils.writeVInt(out, window.getBorderBottom());
172     WritableUtils.writeVInt(out, window.getBorderLeft());
173     WritableUtils.writeVInt(out, window.getBorderRight());
174
175     WritableUtils.writeVInt(out, window.getHeight());
176     WritableUtils.writeVInt(out, window.getWidth());
177
178     // Write image bytes
179     ByteBuffer buffer = image.getByteBuffer();
180     while (buffer.hasRemaining()) {
181         out.writeByte(buffer.get());
182     }
183 }
184 }

```