Spark源码编译遇到的问题解决

1、内存不够

[ERROR] PermGen space -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors,re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions,
please read the following articles:
[ERROR] [Help 1]http://cwiki.apache.org/confluence/display/MAVEN/OutOfMemoryError

上面这个错误是因为编译的时候内存不够导致的，可以在编译的时候加大内存。

export MAVEN_OPTS="-Xmx2g -XX:MaxPermSize=512M -XX:ReservedCodeCacheSize=512m"

2、缺少scala依赖

[ERROR] Failed to execute goal org.apache.maven.plugins:maven-antrun-plugin:1.7:
run (default) on project spark-core_2.10: An Ant BuildException has occured:
Please set the SCALA_HOME (or SCALA_LIBRARY_PATH if scala is on the path)
environment variables and retry.
[ERROR] around Ant part ...<fail message="Please set the SCALA_HOME
(or SCALA_LIBRARY_PATH if scala is on the path) environment variables and retry.">
@ 6:126 in /export1/spark/spark-0.9.0-incubating/core/target/antrun/build-main.xml

上面这个错误已经写的很清楚了，因为编译Spark的时候需要用到scala，所以得去下载scala，此处编译我用的是2.10.3版本，安装如下：

[wyp@master spark]$ wget http://www.scala-lang.org/files/archive/scala-2.10.3.tgz
[wyp@master spark]$ tar -zxf scala-2.10.3.tgz
[wyp@master spark]$ export SCALA_HOME=/export1/spark/scala-2.10.3

3、mqtt-client-0.4.0无法下载

```
[error] Server access Error: sun.security.validator.ValidatorException:
PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
url=https://repo.eclipse.org/content/repositories/paho-releases/
      org/eclipse/paho/mqtt-client/0.4.0/mqtt-client-0.4.0.pom
[warn]  module not found: org.eclipse.paho#mqtt-client;0.4.0
```

这个错误是Maven无法下载mqtt-client-0.4.0.pom，因为他用到的是HTTPS协议。解决这个错误可以通过添加对repo.eclipse.org的信任从而解决这个问题。那如何解决呢？下载InstallCert.java类：

```
/**
 * http://blogs.sun.com/andreas/resource/InstallCert.java
 * Use:
 * java InstallCert hostname
 * Example:
 *% java InstallCert ecc.fedora.redhat.com
 */

import javax.net.ssl.*;
import java.io.*;
import java.security.KeyStore;
import java.security.MessageDigest;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

/**
 * Class used to add the server's certificate to the KeyStore
 * with your trusted certificates.
 */
public class InstallCert {

    public static void main(String[] args) throws Exception {
        String host;
        int port;
        char[] passphrase;
        if ((args.length == 1) || (args.length == 2)) {
            String[] c = args[0].split(":");
            host = c[0];
            port = (c.length == 1) ? 443 : Integer.parseInt(c[1]);
```

```
    String p = (args.length == 1) ? "changeit" : args[1];
    passphrase = p.toCharArray();
} else {
    System.out.println("Usage: java InstallCert <host>[:port"+
                       "[passphrase]");
    return;
}

File file = new File("jssecacerts");
if (file.isFile() == false) {
    char SEP = File.separatorChar;
    File dir = new File(System.getProperty("java.home") + SEP
            + "lib" + SEP + "security");
    file = new File(dir, "jssecacerts");
    if (file.isFile() == false) {
        file = new File(dir, "cacerts");
    }
}
System.out.println("Loading KeyStore " + file + "...");
InputStream in = new FileInputStream(file);
KeyStore ks = KeyStore.getInstance(KeyStore.getDefaultType());
ks.load(in, passphrase);
in.close();

SSLContext context = SSLContext.getInstance("TLS");
TrustManagerFactory tmf =
        TrustManagerFactory.getInstance(
                        TrustManagerFactory.getDefaultAlgorithm());
tmf.init(ks);
X509TrustManager defaultTrustManager =
                    (X509TrustManager) tmf.getTrustManagers()[0];
SavingTrustManager tm = new SavingTrustManager(defaultTrustManager);
context.init(null, new TrustManager[]{tm}, null);
SSLSocketFactory factory = context.getSocketFactory();

System.out.println("Opening connection to " + host + ":" + port + "...");
SSLSocket socket = (SSLSocket) factory.createSocket(host, port);
socket.setSoTimeout(10000);
try {
    System.out.println("Starting SSL handshake...");
    socket.startHandshake();
    socket.close();
    System.out.println();
    System.out.println("No errors, certificate is already trusted");
} catch (SSLException e) {
    System.out.println();
```

```
    e.printStackTrace(System.out);
}

X509Certificate[] chain = tm.chain;
if (chain == null) {
    System.out.println("Could not obtain server certificate chain");
    return;
}

BufferedReader reader =
        new BufferedReader(new InputStreamReader(System.in));

System.out.println();
System.out.println("Server sent " + chain.length + " certificate(s):");
System.out.println();
MessageDigest sha1 = MessageDigest.getInstance("SHA1");
MessageDigest md5 = MessageDigest.getInstance("MD5");
for (int i = 0; i < chain.length; i++) {
    X509Certificate cert = chain[i];
    System.out.println
            (" " + (i + 1) + " Subject " + cert.getSubjectDN());
    System.out.println("   Issuer  " + cert.getIssuerDN());
    sha1.update(cert.getEncoded());
    System.out.println("   sha1    " + toHexString(sha1.digest()));
    md5.update(cert.getEncoded());
    System.out.println("   md5     " + toHexString(md5.digest()));
    System.out.println();
}

System.out.println("Enter certificate to add to trusted keystore or"+
                   " 'q' to quit: [1]");
String line = reader.readLine().trim();
int k;
try {
    k = (line.length() == 0) ? 0 : Integer.parseInt(line) - 1;
} catch (NumberFormatException e) {
    System.out.println("KeyStore not changed");
    return;
}

X509Certificate cert = chain[k];
String alias = host + "-" + (k + 1);
ks.setCertificateEntry(alias, cert);

OutputStream out = new FileOutputStream("jssecacerts");
ks.store(out, passphrase);
```

```
    out.close();

    System.out.println();
    System.out.println(cert);
    System.out.println();
    System.out.println
            ("Added certificate to keystore 'jssecacerts' using alias '"
                    + alias + "'");
}

private static final char[] HEXDIGITS = "0123456789abcdef".toCharArray();

private static String toHexString(byte[] bytes) {
    StringBuilder sb = new StringBuilder(bytes.length * 3);
    for (int b : bytes) {
        b &= 0xff;
        sb.append(HEXDIGITS[b >> 4]);
        sb.append(HEXDIGITS[b & 15]);
        sb.append(' ');
    }
    return sb.toString();
}

private static class SavingTrustManager implements X509TrustManager {

    private final X509TrustManager tm;
    private X509Certificate[] chain;

    SavingTrustManager(X509TrustManager tm) {
        this.tm = tm;
    }

    public X509Certificate[] getAcceptedIssuers() {
        throw new UnsupportedOperationException();
    }

    public void checkClientTrusted(X509Certificate[] chain, String authType)
            throws CertificateException {
        throw new UnsupportedOperationException();
    }

    public void checkServerTrusted(X509Certificate[] chain, String authType)
            throws CertificateException {
        this.chain = chain;
        tm.checkServerTrusted(chain, authType);
    }
```

```
    }

}
```

编译上面的类：

```
$ javac InstallCert.java
$ java InstallCert repo.eclipse.org
```

这样会在当前目录下面生成名为jssecacerts
的文件，将这个文件拷贝到你java安装目录的$JAVA_HOME/lib/security/下面，然后上述问题即可解决。