链滴

使用 hadoop api 获取任务日志

作者：arrayMi

正常情况下我们可以通过开启日志聚合在yarn webUi上查看任务日志，但是当我们需要定制日志呈方式时就需要使用到hadoop提供的api来获取。以下为demo。

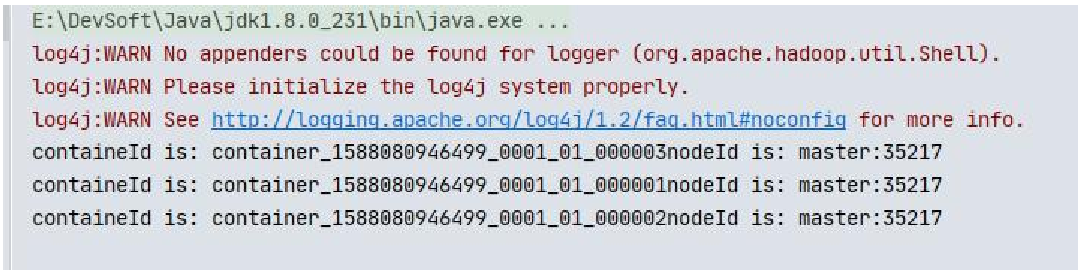
引入依赖

```
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-client</artifactId>
      <version>2.7.2</version>
    </dependency>
```

代码实现：

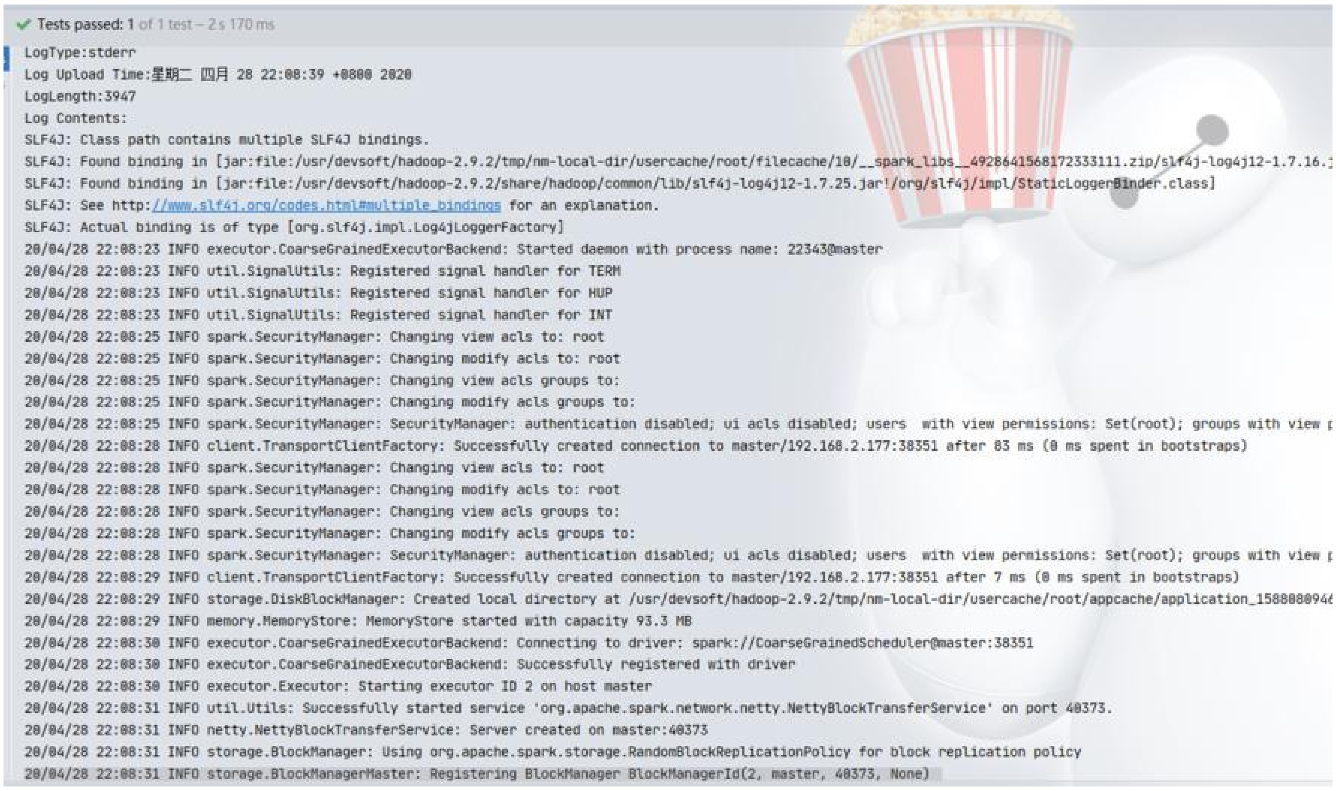
```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileContext;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.RemoteIterator;
import org.apache.hadoop.yarn.api.records.ApplicationId;
import org.apache.hadoop.yarn.conf.YarnConfiguration;
import org.apache.hadoop.yarn.logaggregation.AggregatedLogFormat;
import org.apache.hadoop.yarn.logaggregation.LogAggregationUtils;
import org.apache.hadoop.yarn.logaggregation.LogCLIHelpers;
import org.apache.hadoop.yarn.util.ConverterUtils;
import org.apache.hadoop.yarn.util.Times;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.DataInputStream;
import java.io.EOFException;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.PrintStream;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

public class HadoopLogUtils {
    private static Configuration yarnConfiguration;
    private static LogCLIHelpers logCLIHelpers;
    private static final Logger LOGGER = LoggerFactory.getLogger(HadoopLogUtils.class);

    /**
     * 初始化配置
     */
    static {
        yarnConfiguration = new YarnConfiguration();

        yarnConfiguration.addResource("core-site.xml");
        yarnConfiguration.addResource("hdfs-site.xml");
        yarnConfiguration.addResource("yarn-site.xml");
        logCLIHelpers = new LogCLIHelpers();
        logCLIHelpers.setConf(yarnConfiguration);
    }

    public static Configuration getYarnConfiguration() {
        return yarnConfiguration;
    }

    /**
     * 获取容器日志
     *
     * @param appId
     * @param containerId
     * @param nodeId
     * @param jobOwner
     * @param out
     * @return
     * @throws IOException
     */
    public static int dumpAContainersLogs(String appId, String containerId, String nodeId, Stri
g jobOwner, PrintStream out, List<String> logType) throws IOException {
        Path remoteRootLogDir = new Path(getYarnConfiguration().get("yarn.nodemanager.rem
te-app-log-dir", "/tmp/logs"));

        String suffix = LogAggregationUtils.getRemoteNodeLogDirSuffix(getYarnConfiguration())

        Path remoteAppLogDir = LogAggregationUtils.getRemoteAppLogDir(remoteRootLogDir,
ConverterUtils.toApplicationId(appId), jobOwner, suffix);

        RemoteIterator nodeFiles;
        try {
            Path qualifiedLogDir = FileContext.getFileContext(getYarnConfiguration()).makeQualif
ed(remoteAppLogDir);
            nodeFiles = FileContext.getFileContext(qualifiedLogDir.toUri(), getYarnConfiguration()).
istStatus(remoteAppLogDir);
        } catch (FileNotFoundException var16) {
            logDirNotExist(remoteAppLogDir.toString());
            return -1;
        }

        boolean foundContainerLogs = false;

        while (nodeFiles.hasNext()) {
            FileStatus thisNodeFile = (FileStatus) nodeFiles.next();
            String fileName = thisNodeFile.getPath().getName();
            if (fileName.contains(LogAggregationUtils.getNodeString(nodeId)) && !fileName.end
With(".tmp")) {
                AggregatedLogFormat.LogReader reader = null;

                try {
                    reader = new AggregatedLogFormat.LogReader(getYarnConfiguration(), thisNod
File.getPath());
                    if (dumpAContainerLogs(containerId, reader, out, thisNodeFile.getModificationTi
e(), logType) > -1) {
                        foundContainerLogs = true;
                    }
                } finally {
                    if (reader != null) {
                        reader.close();
                    }

                }
            }
        }

        if (!foundContainerLogs) {
            containerLogNotFound(containerId);
            return -1;
        } else {
            return 0;
        }
    }

    private static void logDirNotExist(String remoteAppLogDir) {
        System.out.println(remoteAppLogDir + " does not exist.");
        System.out.println("Log aggregation has not completed or is not enabled.");
    }

    private static void containerLogNotFound(String containerId) {
        System.out.println("Logs for container " + containerId + " are not present in this log-file."
;
    }

    public static int dumpAContainerLogs(String containerIdStr, AggregatedLogFormat.LogRea
er reader, PrintStream out, long logUploadedTime, List<String> logType) throws IOException

        AggregatedLogFormat.LogKey key = new AggregatedLogFormat.LogKey();
        DataInputStream valueStream;
        for (valueStream = reader.next(key); valueStream != null && !key.toString().equals(conta
nerIdStr); valueStream = reader.next(key)) {
            key = new AggregatedLogFormat.LogKey();
        }

        if (valueStream == null) {
            return -1;
        } else {
            boolean foundContainerLogs = false;

            while(true) {
                try {
                    readContainerLogs(valueStream, out, logUploadedTime, logType);
                    foundContainerLogs = true;
                } catch (EOFException var10) {
                    if (foundContainerLogs) {
                        return 0;
                    }

                    return -1;
                }
            }
        }
    }


    /**
     * 获取Containe nodeId列表
     *
     * @param appId
     * @param appOwner
     * @return
     * @throws IOException
     */
    public static Map<String, String> getContaines(String appId, String appOwner) throws IOE
ception {
        Path remoteRootLogDir = new Path(yarnConfiguration.get(
                YarnConfiguration.NM_REMOTE_APP_LOG_DIR,
                YarnConfiguration.DEFAULT_NM_REMOTE_APP_LOG_DIR));
        String user = appOwner;
        String logDirSuffix = LogAggregationUtils.getRemoteNodeLogDirSuffix(yarnConfigurati
n);
        // TODO Change this to get a list of files from the LAS.
        Path remoteAppLogDir = LogAggregationUtils.getRemoteAppLogDir(
                remoteRootLogDir, ConverterUtils.toApplicationId(appId), user, logDirSuffix);
        RemoteIterator<FileStatus> nodeFiles;
        Map<String, String> containerAndNodeId = new LinkedHashMap<>();
        try {
            Path qualifiedLogDir =
                    FileContext.getFileContext(yarnConfiguration).makeQualified(remoteAppLogDir);
            nodeFiles = FileContext.getFileContext(qualifiedLogDir.toUri(),
                    yarnConfiguration).listStatus(remoteAppLogDir);
        } catch (FileNotFoundException fnf) {
            logDirNotExist(remoteAppLogDir.toString());
            return Collections.emptyMap();
        }
        boolean foundAnyLogs = false;
        while (nodeFiles.hasNext()) {
            FileStatus thisNodeFile = nodeFiles.next();
            if (!thisNodeFile.getPath().getName()
                    .endsWith(LogAggregationUtils.TMP_FILE_SUFFIX)) {
                AggregatedLogFormat.LogReader reader =
                        new AggregatedLogFormat.LogReader(yarnConfiguration, thisNodeFile.getPat
h());
                try {

                    DataInputStream valueStream;
                    AggregatedLogFormat.LogKey key = new AggregatedLogFormat.LogKey();
                    valueStream = reader.next(key);

                    while (valueStream != null) {
                        // Container: container_1587284642166_0001_01_000003 on master_42757
                        containerAndNodeId.put(key.toString(), thisNodeFile.getPath().getName().repl
ace("_", ":"));

                        foundAnyLogs = true;
                        // Next container
                        key = new AggregatedLogFormat.LogKey();
                        valueStream = reader.next(key);
                    }
                } finally {
                    reader.close();
                }
            }
        }
        if (!foundAnyLogs) {
            emptyLogDir(remoteAppLogDir.toString());
            return Collections.emptyMap();
        }
        return containerAndNodeId;
    }


    private static void emptyLogDir(String remoteAppLogDir) {
        System.out.println(remoteAppLogDir + " does not have any log files.");
    }


    private static void readContainerLogs(DataInputStream valueStream,
                                          PrintStream out, long logUploadedTime, List<String> logType) thr
ows IOException {
        byte[] buf = new byte[65535];

        String fileType = valueStream.readUTF();
        //if (logType.contains(fileType)) {
            String fileLengthStr = valueStream.readUTF();
            long fileLength = Long.parseLong(fileLengthStr);
            out.print("LogType:");
            out.println(fileType);
            if (logUploadedTime != -1) {
                out.print("Log Upload Time:");
                out.println(Times.format(logUploadedTime));
            }
            out.print("LogLength:");
            out.println(fileLengthStr);
            out.println("Log Contents:");

            long curRead = 0;
            long pendingRead = fileLength - curRead;
            int toRead =
                    pendingRead > buf.length ? buf.length : (int) pendingRead;
            int len = valueStream.read(buf, 0, toRead);
            while (len != -1 && curRead < fileLength) {
                out.write(buf, 0, len);
                curRead += len;

                pendingRead = fileLength - curRead;
                toRead =
                        pendingRead > buf.length ? buf.length : (int) pendingRead;
                len = valueStream.read(buf, 0, toRead);
            }
            out.println("End of LogType:" + fileType);
            out.println("");
        // }
    }

    /**
     * covert appId
     * @param appId
     * @return
     */
    public static ApplicationId convert(String appId) {
        return ConverterUtils.toApplicationId(appId);
    }


}
```

实现效果：



```
E:\DevSoft\Java\jdk1.8.0_231\bin\java.exe ...
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
containeId is: container_1588080946499_0001_01_000003nodeId is: master:35217
containeId is: container_1588080946499_0001_01_000001nodeId is: master:35217
containeId is: container_1588080946499_0001_01_000002nodeId is: master:35217
```

原文链接：使用 hadoop api 获取任务日志



```
Tests passed: 1 of 1 test – 2 s 170 ms
LogType:stderr
Log Upload Time:星期二 四月 28 22:08:39 +0800 2020
LogLength:3947
Log Contents:
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/devsoft/hadoop-2.9.2/tmp/nm-local-dir/usercache/root/filecache/10/__spark_libs__4928641568172333111.zip/slf4j-log4j12-1.7.16.
SLF4J: Found binding in [jar:file:/usr/devsoft/hadoop-2.9.2/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
20/04/28 22:08:23 INFO executor.CoarseGrainedExecutorBackend: Started daemon with process name: 22343@master
20/04/28 22:08:23 INFO util.SignalUtils: Registered signal handler for TERM
20/04/28 22:08:23 INFO util.SignalUtils: Registered signal handler for HUP
20/04/28 22:08:23 INFO util.SignalUtils: Registered signal handler for INT
20/04/28 22:08:25 INFO spark.SecurityManager: Changing view acls to: root
20/04/28 22:08:25 INFO spark.SecurityManager: Changing modify acls to: root
20/04/28 22:08:25 INFO spark.SecurityManager: Changing view acls groups to:
20/04/28 22:08:25 INFO spark.SecurityManager: Changing modify acls groups to:
20/04/28 22:08:25 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(root); groups with view
20/04/28 22:08:28 INFO client.TransportClientFactory: Successfully created connection to master/192.168.2.177:38351 after 83 ms (0 ms spent in bootstraps)
20/04/28 22:08:28 INFO spark.SecurityManager: Changing view acls to: root
20/04/28 22:08:28 INFO spark.SecurityManager: Changing modify acls to: root
20/04/28 22:08:28 INFO spark.SecurityManager: Changing view acls groups to:
20/04/28 22:08:28 INFO spark.SecurityManager: Changing modify acls groups to:
20/04/28 22:08:28 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(root); groups with view
20/04/28 22:08:29 INFO client.TransportClientFactory: Successfully created connection to master/192.168.2.177:38351 after 7 ms (0 ms spent in bootstraps)
20/04/28 22:08:29 INFO storage.DiskBlockManager: Created local directory at /usr/devsoft/hadoop-2.9.2/tmp/nm-local-dir/usercache/root/appcache/application_15888094
20/04/28 22:08:29 INFO memory.MemoryStore: MemoryStore started with capacity 93.3 MB
20/04/28 22:08:30 INFO executor.CoarseGrainedExecutorBackend: Connecting to driver: spark://CoarseGrainedScheduler@master:38351
20/04/28 22:08:30 INFO executor.CoarseGrainedExecutorBackend: Successfully registered with driver
20/04/28 22:08:30 INFO executor.Executor: Starting executor ID 2 on host master
20/04/28 22:08:31 INFO util.Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 40373.
20/04/28 22:08:31 INFO netty.NettyBlockTransferService: Server created on master:40373
20/04/28 22:08:31 INFO storage.BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
20/04/28 22:08:31 INFO storage.BlockManagerMaster: Registering BlockManager BlockManagerId(2, master, 40373, None)
```

源码地址： https://github.com/arrayMi/hadoop_learning