Study of Hadoop Data Migration Based on Oozie

Kehe Wu[1,a], Yanwen An[2,b *], Tingting Wu[3,c], Wenjing Zeng[4,d]

[1]North China Electric Power University, Beijing, China

[2]North China Electric Power University, Beijing, China
[a]wukehe@ncepu.edu.cn, [b]an_yanwen@163.com

Keywords: data migration; Hadoop; Oozie; Flex

Abstract. As more companies use Hadoop, they need to migrate data to the platform. Traditional migration needs professional people operating on bash frequently. It requires a lot of manpower and is not easy to maintain the jobs. This paper puts forward a scheme that configures a migration task in webpage based on Flex then submits the mission to Oozie to execute timely. It requires less professional knowledge and easy to maintain because of friendly user interface. The paper also details the specific implementation of the scheme, and introduces the actual usage.

Introduction

With the evolution of information collection mechanism, storage technology and database technology, the information has exploded, leading to the traditional information and communication technology can't deal with them. According to IDC (Internet Data Center) forecasts, by 2020, the data will be increased from 1.8ZB in 2011 to 40ZB. Enterprises or institutions have great difficulty in the acquisition, operation, storage, search, retrieval, sharing, transfer, analysis of big data and the visualization of big data, especially the data volume is huge, and the existence of a large number of heterogeneous information, data operation contains numerous and complex business rules, integration and use of these data has become a huge challenge.

Migration is moving the electronic data from the original data into a new data environment system environment, which is usually a sub activity deployment of enterprise applications. Hosting the migration task to workflow and executing timely is one of the effective migration method. In this paper, the research content is the visualization of migration configuration, realizing the transparent effect of migration operation to users.

Overall architecture

This system adopts B/S mode, web page display part based on the flex implementation, User defined migration task will be submitted to the Oozie server to execute in the background, and the task configuration stored in the database by iBATIS. The main features of the system include create, modify, delete, query, start and stop migration tasks. The system architecture is shown in FIG.1.

The presentation layer is responsible for presenting information. Flex uses MXML to define UI layout and other non-visual static aspects, ActionScript to address dynamic aspects and as code-behind. The presentation layer contacts with the application layer through GET/POST method of HTTP requests. In the control layer of application layer, the request is accepted by Servlet and parameters included in the request will be transferred to java methods in business logic layer [1]. In Business logic layer, parameters will be integrated to configuration files and upload to Oozie server which locates in the third layer of the application layer-service layer.

The service layer runs on HDFS platform, it contacts with business logic layer by OozieClient, a java interface provided by Oozie. When the task is successfully submitted to the Oozie server, methods in application layer will save the data of configuration of the task and return value from Oozie server to database using iBATIS [2]. Users can check the information of the task in presentation layer through a browser.

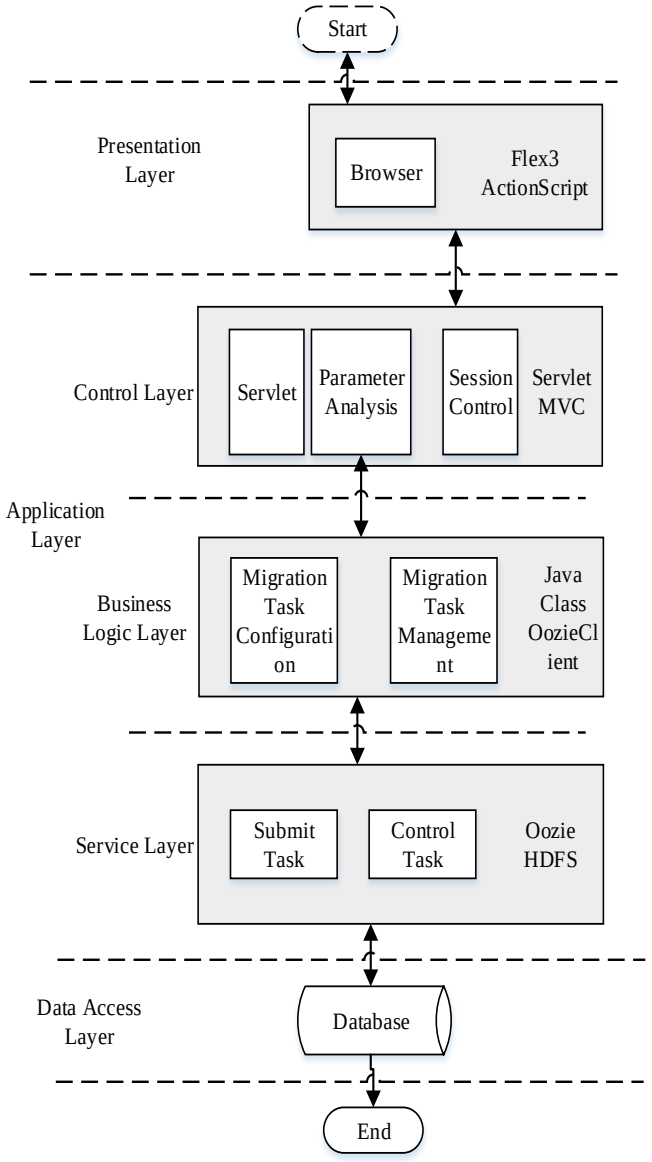

Figure 1. System Architecture

The details of the data migration task is shown in table I below.

Migration Type	Migration Configuration			Task Configuration
	FTP-HDFS	Oracle-HDFS	HDFS-HBase	
Source Type	Migration Name & Task Type			Task Name
Target Type	HDFS Path for Importing		HDFS path for exporting	Queue Name
Source Object	Parallel Number		HBase table name	Start Time
Target Object	Updating Mode	Ways of Migration	HBase Column Cluster	End Time
	FTP Path	Split field	HBase Row Key	Frequency
		Data Separator		Frequency Unit

TABLE I. DETAILS OF THE TASK

Design and implementation of data migration

Sometimes tasks running on Hadoop needs connect multiple Map/Reduce jobs together to finish the job. There is a relatively new component Oozie in Hadoop, which allows us combine multiple Map/Reduce jobs into a logical unit of work to finish huger tasks. Oozie is a kind of Java Web application which runs in Java servlet container tomcat, and uses database to store contents as follows:

- Workflow definition
- Currently running workflow instance, including status of instance and variables

Oozie workflow is a collection of actions (such as Mad/Reduce job in Hadoop, Pig job etc.) arranged in a control dependency DAG (Direct Acyclic Graph), in which assigns the order of executing actions. We use hPDL (a kind of XML process definition language) to describe the graph.

All computing and processing tasks triggered by an action node is not in Oozie – they are executed by Map/Reduce frame of Hadoop. It allows Oozie supporting existing Hadoop for load balancing, disaster recovery mechanism in this way. Most of the tasks are executed asynchronously (except actions of file system which are executed synchronously). It means that for most types of computing or processing tasks triggered by workflow actions, they will wait until workflow operations change to next node of workflow and continue after finishing computing or processing tasks. Oozie can detect computing or processing tasks completion through two different ways, which means callback and polling. When Oozie starts computing or processing tasks, it will provide a unique callback URL for the

task, and the task will send notification to specific URL when finished. In circumstance of tasks cannot trigger callback URL (caused by any reasons such as network wink), or type of tasks cannot triggers callback URL when finished, Oozie has a mechanism, which could polling computing or processing tasks for completion.

The Oozie Coordinator system allows the user to define workflow execution schedules based on regular time intervals and/or data availability and/or external events. Oozie coordinator allows to model workflow execution triggers in the form of the predicates, which can reference to data, time and/or external events. The workflow job is started after the predicate is satisfied.

Each user submits a migration task, the system will generate a unique UUID for this task, generate three configuration files (workflow.xml, coordinator.xml, job.properties) and upload to the folder named by the UUID under the JobContant directory. Then release task to Oozie server by calling interfaces of the OozieClient. The system will decide whether write to the database by judging if the task submitted successfully according to TaskId returned by Oozie Server. The flow chart is shown in Fig.2.

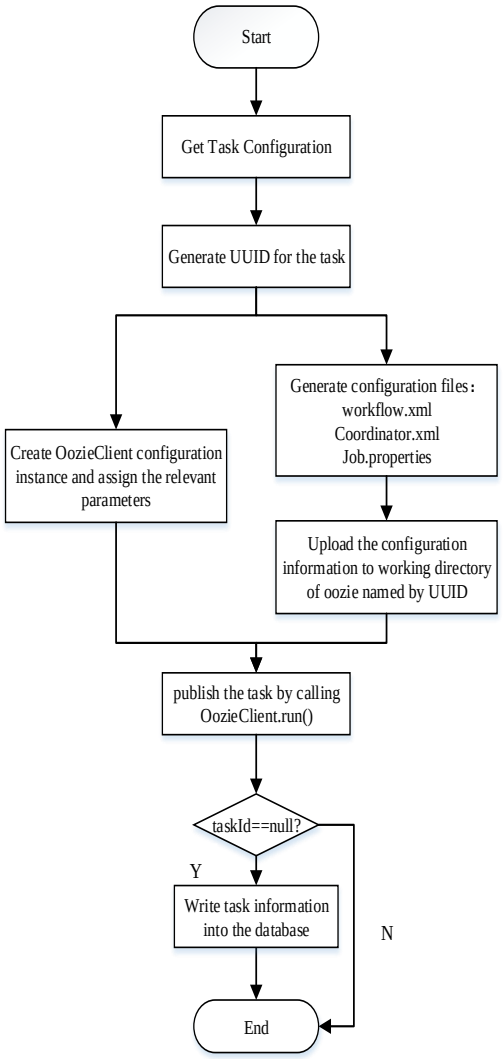

Figure 2. Flow chart of data migration

A. Job.properties

Defining the relevant environmental variables in job.properties, of which appsRoot defined as the generated UUID. The main configuration is as follows:

appsRoot=uuid

oozie.coord.application.path=${nameNode}/user/oozie/${appsRoot}

oozie.wf.application.path=${nameNode}/user/${user.name}/${appsRoot}/apps/shell

workflowAppUri=${nameNode}/user/oozie/${appsRoot}

start=starttime

end=endtime

B. coordinator.xml

This file mainly sets the frequency, starting time and ending time of execution of the timing task. The main configuration is as follows:

<coordinator-app name="migration" frequency= frequencyStr start=${start} end=${end} timezone="UTC" xmlns="uri:oozie:coordinator:0.2">

…

</coordinator-app>.

C. workflow.xml

Depending on the type of migration, workflow.xml wording also vary:

1) ftp to HDFS

Hadoop comes with a useful program called distcp for copying large amounts of data to and from Hadoop file system in parallel [3].

The canonical use case for distcp is for transferring data between two HDFS clusters. If the clusters are running identical versions of Hadoop, the hdfs scheme is appropriate:

% hadoop distcp hdfs://namenode1/foo hdfs://namenode2/bar

This will copy the /foo directory (and its contents) from the first cluster to the /bar directory on the second cluster, so the second cluster ends up with the directory struc- ture /bar/foo. If /bar doesn't exist, it will be created first. You can specify multiple source paths, and all will be copied to the destination. Source paths must be absolute [4].

Here we use another function of distcp: copy files from ftp to hdfs. Format is as follows：

% hadoop distcp ftp://fptuser:ftppassword@host/ftp/path/ hdfs://node:port1/hdfs/path/

distcp is implemented as a MapReduce job where the work of copying is done by the maps that run in parallel across the cluster. There are no reducers. Each file is copied by a single map, and distcp tries to give each map approximately the same amount of data by bucketing files into roughly equal allocations [4]. To keep an HDFS cluster balanced, it's best to start by running distcp with the default of 20 maps per node. In the Workflow configuration using the following format:

<distcp>
    <arg>ftppath</arg>
    <arg>hdfspath</arg>
</disctp>

2) Oracle to HDFS

Oracle migrated to HDFS has three different ways of migration, respectively all migration, custom SQL migration and migration of tables and columns specified [5]. Although we use the same tool called Sqoop to realize that.

With Sqoop, we can import data from a relational database system into HDFS. The input to the import process is a database table. Sqoop will read the table row-by-row into HDFS [6]. The output of this import process is a set of files containing a copy of the imported table. The import process is performed in parallel. For this reason, the output will be in multiple files. These files may be delimited text files (for example, with commas or tabs separating each field), or binary Avro or SequenceFiles containing serialized record data. Use command tag within sqoop tag to label specific migration configuration, such as the oracle database connection, the connection information (IP, port, username, password, etc.), whether the whole migration or custom SQL statement or specified table and column migration and the absolute path of migration target. The specific format is as shown below:

<sqoop>
<command>import --connect jdbc:oracle://localhost/db --username foo --table TEST</command>
</sqoop>

3) HDFS to HBase

There are three main ways of import text with formatted data into HBase: hive SQL, importtsv+ completebulkload and mapreduce+ completebulkload. Since the situation this paper deal with is mostly importing data to an empty table of HBase, not related to the split operation of HBase table, in which case, the choice of ways to generate Hfile file is most efficient[6]. So use importtsv+ completebulkload for data import task. The process is completed by two action, the first action execute importtsv commands, and converts formatted text data into HBase internal data storage format Hfile file. The second action executes completebulkload command, HFile data will be loaded into the HBase table. The essence of running the command is a HDFS mv operation, and MapReduce will not start, so the speed is very fast [7]. When executing this command, the Hadoop may not find the HBase dependencies, abnormal ClassNotFoundException. A simple solution is put HBase related jars under ${HADOOP_HOME}/share/hadoop/common/lib, so that Hadoop can load the related HBase jar files before running [8]. Using the execution of the shell script to perform the migration task, the format is as shown:

<exec>hadoop</exec>

<argument>…<argument>
<file>${HBASE_HOME}/hbase-server-VERSION.jar</file>
Migration configuration command line will be written in the argument label.

Since the first action creates new HFile file, possibly Oozie users do not have execute permission, it is needed to add an action between the two, to give permission to Oozie user executing HFile file generated. The flow chart shown in Fig.3.

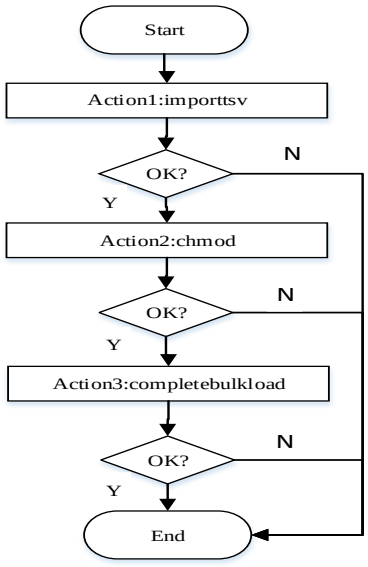

Figure 3. Flow chart of actions

The three configuration files will be uploaded to the folder named with UUID under Oozie working directory, and then create an instance of OozieClient Configuration to set the path of workflow application, parameters like nameNode and jobTracker. Note that, the property of OozieClient.USER_NAME is hdfs instead of oozie. After setting these parameters, the OozieClient instance calls interface to publish the work to Oozie server. The string returned is the TaskId of current task after successfully published. It's a unique identification for the task, used as parameter when start, suspend and stop the task.

Conclusion

The data migration system satisfies the requirement publishing data migration task to the Oozie Server and executing timely. In practice, since the relevant operation of this system is through user interface, and Oozie Server also provides us a user interface to manage job statement, thus the system greatly reduces the difficulty of the operation, enhances the user experience to maintain jobs. Also there are parts needs to be perfected, such as exception handling and logging, etc.

References

[1] Michael Labriola, Jeff Tapper, Matthew Boles. Adobe Flex 4 training from the source. Pearson Education Inc, USA.
[2] Clinton Begin, Barndon Goodin, Larry Meadors. iBATIS in action.Manning Publication Co, CAN.2008.
[3] Alex Holmes. Hadoop in practice. Manning Publications, USA. 2012.
[4] Danil Zburivsky, Sudheesh Narayanan. Hadoop cluster deployment, securing hadoop. Packt Publishing, USA. 2013.
[5] Tom Plunkett, Brian Macdonald, Bruce Nelson. Oracle big data handbook. McGraw-Hill Education. 2014.
[6] Lars George. HBase: the definitive guide. Reprint. Originally published: Sebastopol, CA: O'Reilly Media, c2011.
[7] Tom White. Hadoop: The Definitive Guide,Third Edition. O'Reilly Media, Inc, USA.2012-6-12.
[8] Boris Lublinsky, Kevin T. Smith, Alexey Yakubovich. Professional Hadoop solutions. Wiley Publishing, Inc, USA. 2013.