Vaidya Guide

Table of contents

1 Purpose

This document describes various user-facing facets of Hadoop Vaidya, a performance diagnostic tool for MapReduce jobs. It describes how to execute a default set of rules against your MapReduce job counters and how to write and execute new rules to detect specific performance problems.

A few sample test rules are provided with the tool with the objective of growing the rules database over the time. You are welcome to contribute new rules for everyone's benefit; to do so, follow the How to Contribute procedure specified on Apache Hadoop website.

2 Prerequisites

Ensure that Hadoop is installed and configured. More details:

- Make sure HADOOP_HOME environment variable is set.
- Make sure Java is installed and configured as a part of the Hadoop installation.

3 Overview

Hadoop Vaidya (Vaidya in Sanskrit language means "one who knows", or "a physician") is a rule based performance diagnostic tool for MapReduce jobs. It performs a post execution analysis of MapReduce job by parsing and collecting execution statistics through job history and job configuration files. It runs a set of predefined tests/rules against job execution statistics to diagnose various performance problems. Each test rule detects a specific performance problem with the MapReduce job and provides a targeted advice to the user. This tool generates an XML report based on the evaluation results of individual test rules.

4 Terminology

This section describes main concepts and terminology involved with Hadoop Vaidya,

- *PostExPerformanceDiagnoser*: This class extends the base Diagnoser class and acts as a driver for post execution performance analysis of MapReduce Jobs. It detects performance inefficiencies by executing a set of performance diagnosis rules against the job execution statistics.
- *Job Statistics*: This includes the job configuration information (job.xml) and various counters logged by MapReduce job as a part of the job history log file. The counters are parsed and collected into the Job Statistics data structures, which contains global job level aggregate counters and a set of counters for each Map and Reduce task.
- *Diagnostic Test/Rule*: This is a program logic that detects the inefficiency of M/R job based on the job statistics. The description of the Test is specified as an XML element (DiagnosticTest) in a test description file e.g. default tests description file,

$HADOOP_HOME/contrib/vaidya/conf/postex_diagnosis_tests.xml. The actual logic is coded as a java class and referenced in the DiagnosticTest XML element.

Following section describes the *DiagnosticTest* XML element in a diagnostic test description file

- *DiagnosticTest{Title}*: Specifies a short name/description of the test.
- *DiagnosticTest{ClassName}*: Specifies fully qualified class name that implements the test logic.
- *DiagnosticTest{Description}*: Specifies a full description of the test rule.
- *DiagnosticTest{Importance}*: Specifies a declarative value for overall importance of the test rule. (Values: High, Medium, Low)
- *DiagnosticTest{SuccessThreshod}*: This is a threshold value specified by test case writer such that if impact level of the test case is lesser, then test is declared as PASSED (or NEGATIVE). The impact level is calculated and returned by individual test's evaluate function, specifying the degree of problem job has with respect to the condition being evaluated.
- *DiagnosticTest{Prescription}*: This is a targeted advice written by the test case adviser for the user to follow when test is not PASSED.
- *DiagonsticTest{InputElement}*: This is a test specific input that test writer has to optionally provide. This will be supplied to individual test case class so that test writer can use it within test case. This is typically a test configuration information such that test writer need not change the Java code for test case but rather can configure the test case using these input values.

Following section describes the performance analysis report generated by the tool in XML format

- *PostExPerformanceDiagnosticReport*: This is a document (root) element from the XML report generated by the tool.
- *TestReportElement*: This is a XML report element from the test report document, one for each individual test specified in test description file
- *TestReportElement{TestTitle}*: Will be included from DiagnosticTest{Title}
- *TestReportElement{TestDescription}*: Will be included from DiagnosticTest{Description}
- *TestReportElement{TestImportance}*: Will be included from DiagnosticTest{Importance}
- *TestReportElement{TestSeverity}*: This is a product of Test Impact level and Test Importance. It indicates overall severity of the test.
- *TestReportElement{ReferenceDetails}*: This is a test specific runtime information provided by test case to support the test result and severity. Typically Test writer should print the test impact level in this section.

- *TestReportElement{TestResults}*: This is boolean outcome of the test based on the SuccessThreshold specified by test writer in the DiagnosticTest description. The test PASSED(NEGATIVE) indicates no problem vs. FAILED (POSITIVE) indicates a potential problem with the job for given test case.
- *TestReportElement{TestPrescription}*: This will be included from DiagnosticTest{Prescription}, unless test case writer overrides it in the test case class through getPrescription() method

5 How to Execute the Hadoop Vaidya Tool

Script to execute Hadoop Vaidya is in `$HADOOP_HOME/contrib/vaidya/bin/` directory. It comes with a default set of rules defined in file: `$HADOOP_HOME/contrib/vaidya/conf/postex_diagnosis_tests.xml`

- Make sure HADOOP_HOME environment variable is set and Java is installed and configured.
- Execute the Hadoop Vaidya script with -help (or without any arguments) to get the command line help. e.g. `=>sh $HADOOP_HOME/contrib/vaidya/bin/vaidya.sh -help`
- User needs to supply job's configuration file (`-jobconf job_conf.xml`), job history log file (`-joblog job_history_log_file`), and optionally the test description file (`-testconf postex_diagonostic_tests.xml`). If test description file is not specified then the default one is picked up from the Hadoop Vaidya Jar (`$HADOOP_HOME/contrib/vaidya/hadoop-{version}-vaidya.jar`). This default test description file is also available at following location for users to make a local copy, modify and add new test rules: `$HADOOP_HOME/contrib/vaidya/conf/postex_diagnostic_tests.xml`
- Use `-report report_file` option to store the xml report into specified report_file.

6 How to Write and Execute Your Own Tests

Writing and executing your own test rules is not very hard. You can take a look at Hadoop Vaidya source code for existing set of tests. The source code is at this hadoop svn repository location. The default set of tests are under `"postexdiagnosis/tests/"` folder.

- Writing a test class for your new test case should extend the `org.apache.hadoop.vaidya.DiagnosticTest` class and it should override following three methods from the base class,
  - evaluate()
  - getPrescription()
  - getReferenceDetails()

- Make a local copy of the `$HADOOP_HOME/contrib/vaidya/conf/postex_diagnostic_tests.xml` file or create a new test description XML file.
- Add the test description element for your new test case to this test description file.
- Compile your new test class (or multiple classes), archive them into a Jar file and add it to the CLASSPATH e.g. (`export CLASSPATH=$CLASSPATH:newtests.jar`)
- Execute the Hadoop Vaidya script with the job configuration, job history log and reference to newly created test description file using *--testconf* option.
  ```
  =>sh $HADOOP_HOME/contrib/vaidya/bin/vaidya.sh -joblog job_history_log_file -jobconf job.xml -testconf new_test_description_file -report report.xml
  ```

Java and JNI are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.