HITACHI
Inspire the Next

Logging with Hadoop and Pentaho Data Integration (PDI)

Contents

This page intentionally left blank.

Overview

This document covers some best practices on logging with Hadoop and Pentaho Data Integration (PDI). In it, you will learn how to explore logs to find needed information, and how to customize and configure connections and logging.

Our intended audience is Pentaho and Hadoop administrators.

The intention of this document is to speak about topics generally; however, these are the specific versions covered here:

Software	Version(s)
Pentaho	6.x, 7.x, 8.x

The Components Reference in Pentaho Documentation has a complete list of supported software and hardware.

Before You Begin

Before beginning, use the following information to prepare for the procedures described in the main section of the document.

Prerequisites

This document assumes that you have knowledge of Pentaho, and administrator privileges in the Pentaho environment.

Use Case: Pentaho MapReduce Execution Troubleshooting

> *Janice is a Pentaho administrator who needs to troubleshoot issues that occur in the Hadoop side of the Pentaho orchestration process during Pentaho MapReduce execution.*
>
> *Since these logs are not part of the Pentaho side, Janice must be able to find the associated information in Hadoop that will let her track logs to detect underlying issues in the Pentaho processes.*

PDI Deployment and Execution on Hadoop Cluster

Pentaho allows you to interface with Hadoop in a way that gives you control and performance, maximizing the capabilities of your processes.

You can find details on these topics in the following sections:

- Installing PDI Libraries
- Hadoop Monitoring and Logging on Pentaho

Installing PDI Libraries

The first time you trigger a job against the Hadoop cluster to perform a Pentaho MapReduce (PMR) process, all PDI libraries are copied from the client or server running the job into the Hadoop Distributed File System (HDFS).

PDI libraries, which are Java classes in Java Archive (JAR) file format, default to the location `/opt/pentaho/mapreduce`.

However, you can change the PDI libraries location in this way:

1. Open `plugin.properties` in the `data-integration/plugins/pentaho-big-data-plugin` directory.
2. Change the configuration property `pmr.kettle.dfs.install.dir` to the alternate path you prefer.

Remember that the user performing the PDI process, whether an operating system (OS) user or a Kerberos principal, must have access to the PDI deployment path on HDFS.

This illustrates how the process works:

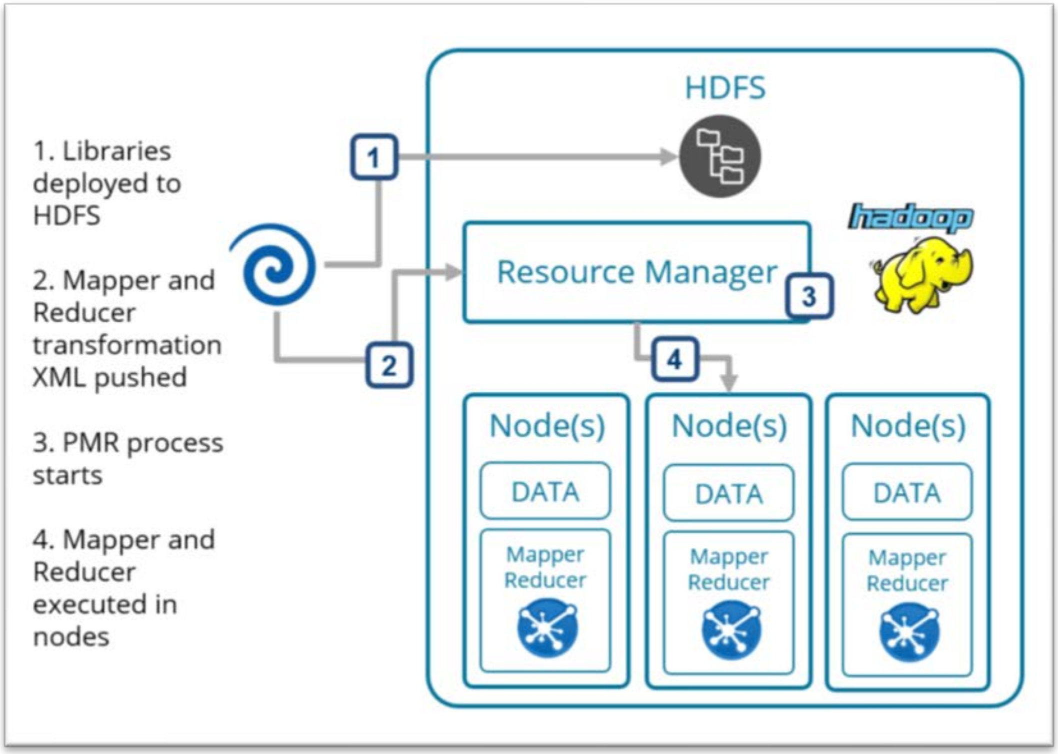


Figure 1: Executing PDI

Hadoop Monitoring and Logging on Pentaho

Interaction between PDI and the Hadoop cluster occurs in a few ways:

Table 1: PDI and Hadoop Integration

Interaction	Details
Hive	This interaction makes use of the JDBC driver and offers the same capabilities available for any other database vendor with PDI, such as. execute SQL, read/put data, etc.
HDFS	HDFS interacts with the cluster distributed file system (read/copy files to/from HDFS). HDFS files are inputs or outputs of the stream.
PMR	This is a native execution through YARN. You can distribute and scale processes to be executed in the cluster by using MapReduce as a YARN application.

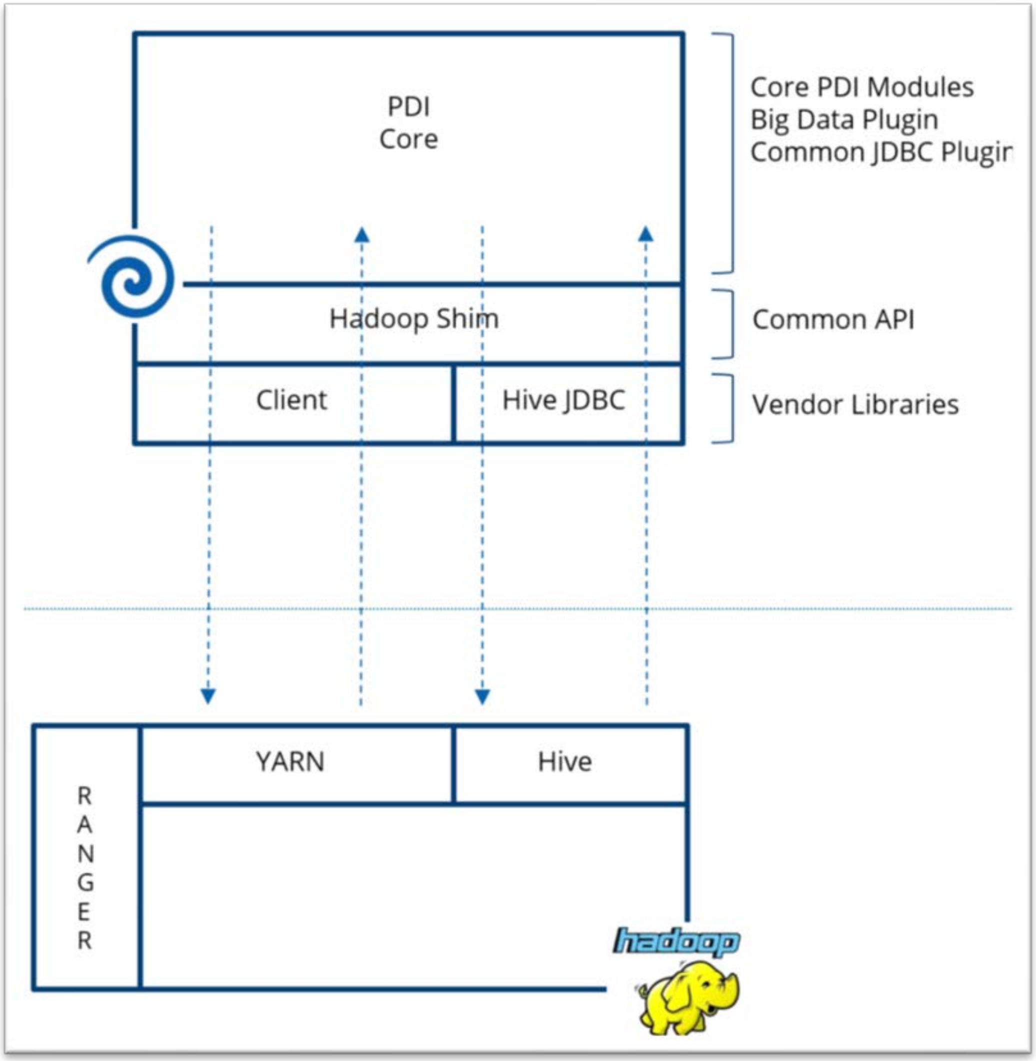


Figure 2: PDI and Hadoop Interaction

Using Resource Manager to Monitor PDI Execution in the Cluster

Resource Manager coordinates and manages all the resources for distributed applications running on YARN. You can monitor it using Resource Manager, just as you would for any other YARN application in the cluster, because PDI triggers a YARN application execution

Each PDI execution has an application ID associated with it in the Resource Manager.

More information on YARN use is available at Simplifying User-Logs Management and Access in YARN.

You can configure your cluster log retention time with `yarn.log-aggregation` parameters. Be aware of your configuration; it may be set to remove related content after a process ends.

Using AEL/Spark

Starting with Pentaho 8.0, execution over Spark using AEL is available. If you use this option, you will want to be able to monitor your events.

Capture running or completed Spark transformations an event log, which you can then view with the Spark History Server. The Spark History Server is a browser-based user interface to the event log. Before you can use the Spark History Server, you must configure AEL to log the events.

The steps for how to configure AEL for logging are available at Configure Event Logging in the Pentaho documentation.

Debugging and Problem Solving

This section contains various procedures and tips for solving problems with logging in Hadoop and PDI:

- Getting Pentaho Logs from PMR Execution
- Configuring PDI Memory Usage with JVM
- PMR Configuration

Getting Pentaho Logs from PMR Execution

Pentaho can show all the logs from the orchestration process, but you need to look in the YARN application execution logs for the part of the orchestration related to the PMR transformations executed as Mapper and Reducer.

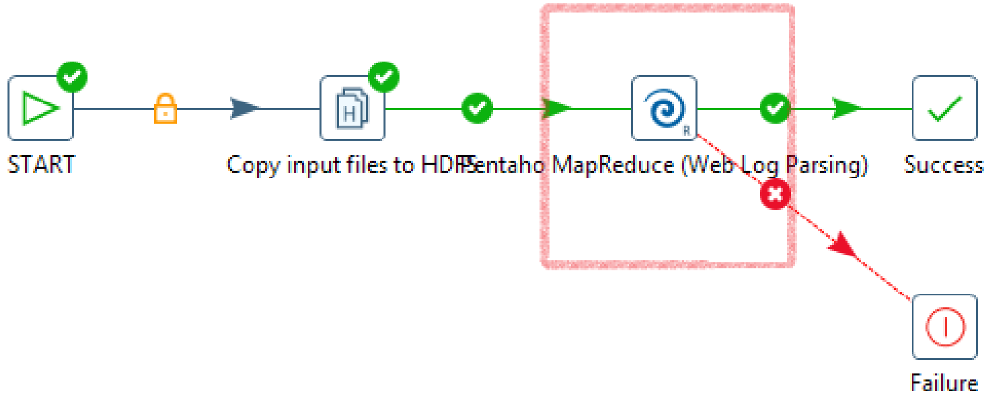
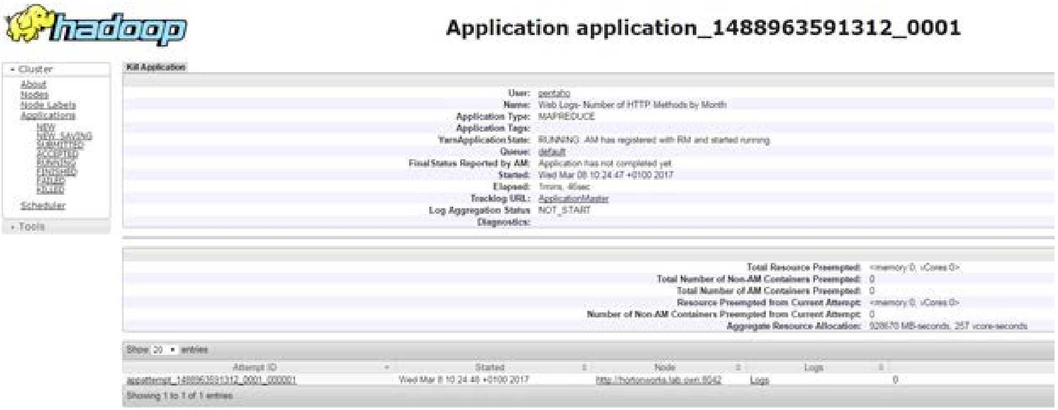


Figure 3: MapReduce Debugging

Troubleshooting Problems

Here are some steps to try when the PMR process is running, if you experience problems in this part of the process:

1. Go to Resource Manager.
2. Find the **application ID** of the PMR process and click on its URL.



Application application_1488963591312_0001

User: pentaho
Name: Web Logs- Number of HTTP Methods by Month
Application Type: MAPREDUCE
Application Tags:
YarnApplication State: RUNNING: AM has registered with RM and started running
Queue: default
FinalStatus Reported by AM: Application has not completed yet.
Started: Wed Mar 08 10:24:47 +0100 2017
Elapsed: 1mins, 46sec
Tracking URL: ApplicationMaster
Log Aggregation Status NOT_START
Diagnostics:

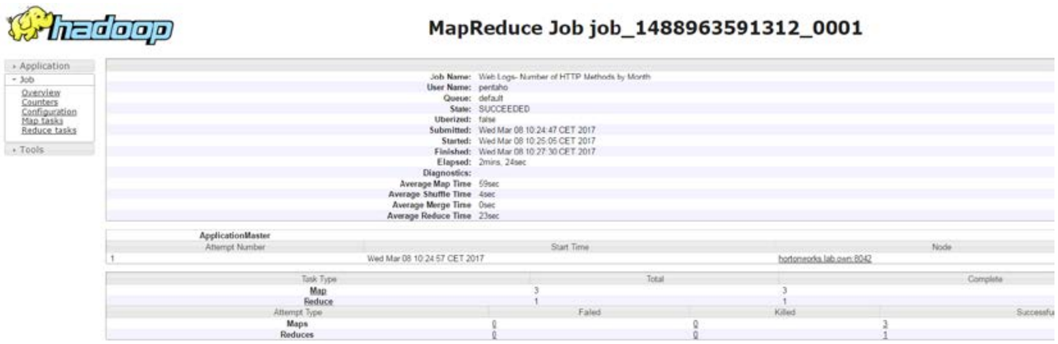
Total Resource Preempted: <memory:0, vCores:0>
Total Number of Non-AM Containers Preempted: 0
Total Number of AM Containers Preempted: 0
Resource Preempted from Current Attempt: <memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt: 0
Aggregate Resource Allocation: 928670 MB-seconds, 257 vcore-seconds

Attempt ID	Started	Node	Logs	
appattempt_1488963591312_0001_000001	Wed Mar 8 10:24:48 +0100 2017	http://hortonworks.lab.own:8042	Logs	0

Showing 1 to 1 of 1 entries

Figure 4: Application ID

3. Click on **ApplicationMaster** in the **Tracking URL** item.
4. Click on the **MapReduce Job ID**. The **Application Master** now shows a resume table with Mappers and Reducers performed and the status of each (**running**, **failed**, **killed**, **successful**).



MapReduce Job job_1488963591312_0001

Job Name: Web Logs- Number of HTTP Methods by Month
User Name: pentaho
Queue: default
State: SUCCEEDED
Uberized: false
Submitted: Wed Mar 08 10:24:47 CET 2017
Started: Wed Mar 08 10:25:05 CET 2017
Finished: Wed Mar 08 10:27:30 CET 2017
Elapsed: 2mins, 24sec
Diagnostics:
Average Map Time 59sec
Average Shuffle Time 4sec
Average Merge Time 0sec
Average Reduce Time 23sec

ApplicationMaster

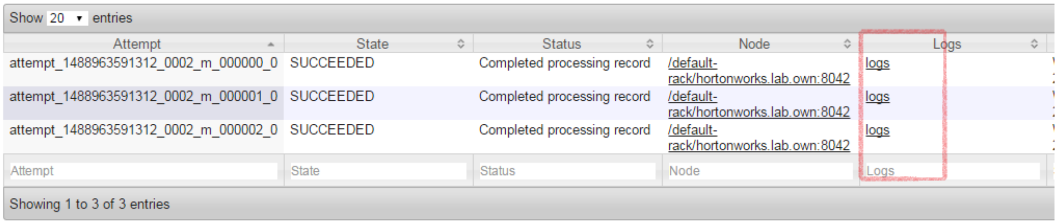
Attempt Number	Start Time	Node
1	Wed Mar 08 10:24:57 CET 2017	hortonworks.lab.own:8042

Task Type	Total	Complete
Map	3	3
Reduce	1	1

Attempt Type	Failed	Killed	Successfu
Maps	0	0	3
Reduces	0	0	1

Figure 5: MapReduce Job ID

5. You can now get into one of the Mappers or Reducers and see the logs, which are distributed in `stderr`, `stdout`, and `syslog`. The Pentaho execution content (transformation steps log) is visible under `stdout` content.

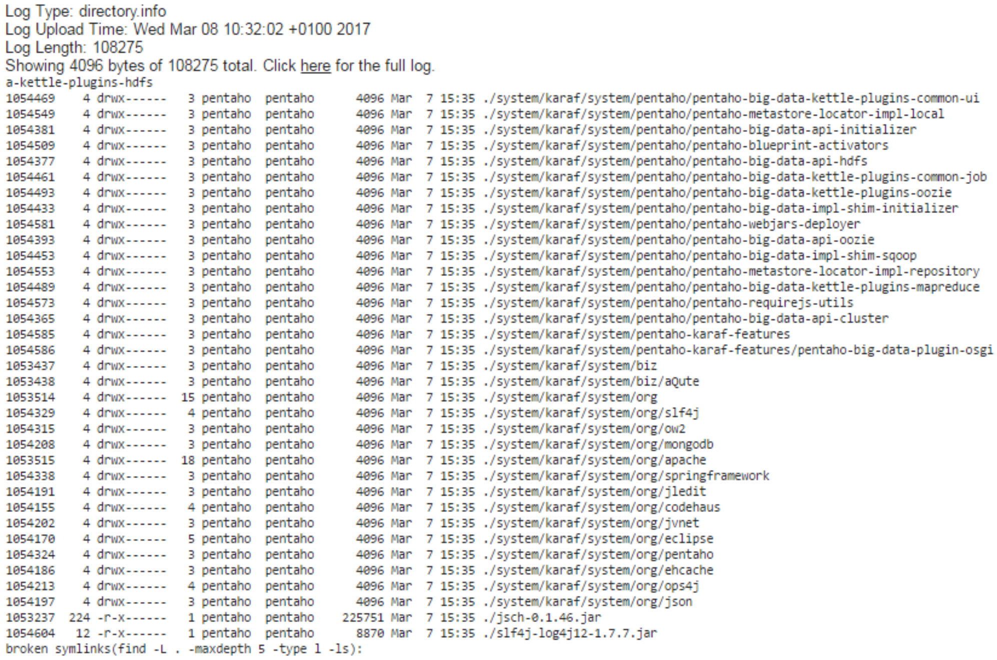


Show 20 ▾ entries

Attempt	State	Status	Node	Logs
attempt_1488963591312_0002_m_000000_0	SUCCEEDED	Completed processing record	/default-rack/hortonworks.lab.own:8042	logs
attempt_1488963591312_0002_m_000001_0	SUCCEEDED	Completed processing record	/default-rack/hortonworks.lab.own:8042	logs
attempt_1488963591312_0002_m_000002_0	SUCCEEDED	Completed processing record	/default-rack/hortonworks.lab.own:8042	logs
Attempt	State	Status	Node	Logs

Showing 1 to 3 of 3 entries

Figure 6: Logs URLs



```
Log Type: directory.info
Log Upload Time: Wed Mar 08 10:32:02 +0100 2017
Log Length: 108275
Showing 4096 bytes of 108275 total. Click here for the full log.
a-kettle-plugins-hdfs
1054469    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-kettle-plugins-common-ui
1054549    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-metastore-locator-impl-local
1054381    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-api-initializer
1054509    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-blueprint-activators
1054377    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-api-hdfs
1054461    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-kettle-plugins-common-job
1054493    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-kettle-plugins-oozie
1054433    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-impl-shim-initializer
1054581    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-webjars-deployer
1054393    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-api-oozie
1054453    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-impl-shim-sqoop
1054553    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-metastore-locator-impl-repository
1054489    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-kettle-plugins-mapreduce
1054573    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-requirejs-utils
1054365    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho/pentaho-big-data-api-cluster
1054585    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho-karaf-features
1054586    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/pentaho-karaf-features/pentaho-big-data-plugin-osgi
1053437    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/biz
1053438    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/biz/aQute
1053514    4 drwx------  15 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org
1054329    4 drwx------   4 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/slf4j
1054315    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/ow2
1054208    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/mongodb
1053515    4 drwx------  18 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/apache
1054338    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/springframework
1054191    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/jledit
1054155    4 drwx------   4 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/codehaus
1054202    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/jvnet
1054170    4 drwx------   5 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/eclipse
1054324    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/pentaho
1054186    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/ehcache
1054213    4 drwx------   4 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/ops4j
1054197    4 drwx------   3 pentaho  pentaho      4096 Mar  7 15:35 ./system/karaf/system/org/json
1053237  224 -r-x------   1 pentaho  pentaho    225751 Mar  7 15:35 ./jsch-0.1.46.jar
1054604   12 -r-x------   1 pentaho  pentaho      8870 Mar  7 15:35 ./slf4j-log4j12-1.7.7.jar
broken symlinks(find -L . -maxdepth 5 -type l -ls):
```

Figure 7: Pentaho Logs

If you have no web browser available, or cannot use the Resource Manager, use YARN commands to request logs per application or process ID or directly with the path of the logs' destination. For example, Cloudera Distribution Including Apache Hadoop (CDH)'s YARN default log files location is `/yarn/nm/usercache/%username%/appcache/%application_id%`.

Accessing Logs After PMR Process

If the PMR process has already finished when you need to access the logs, try these steps instead:

1. Go to Resource Manager.
2. Find the **application ID** of the PMR process and click on its URL.
3. Click on **History** in the **Tracking URL** item.
4. The Application Master now shows a resume table with Mappers and Reducers executed and the status of each (**running**, **failed**, **killed**, **successful**).
5. From here, you can explore Mappers and Reducers by clicking on the appropriate item and choosing the number under the classification you want (such as **Successful**). Logs links are accessible from there.

Configuring PDI Memory Usage with JVM

The amount of memory PDI uses is controlled by the parameter in the `spoon.bat` or `spoon.sh` file, specified with minimum and maximum values (default `-xms1024m -xmx2048m`).

We recommend that you have at least 2GB of memory dedicated to PDI in workstations and 8GB total with at least 4GB dedicated to Pentaho in a server-side configuration.

Many PDI transformation steps allow you to control how memory is allocated. Moving more memory over to PDI along with fine-tuning some step settings can have a big impact on performance.

You can change this configuration by setting the `PENTAHO_DI_JAVA_OPTIONS` environment variable. This does not affect the original configuration files.

PMR Configuration

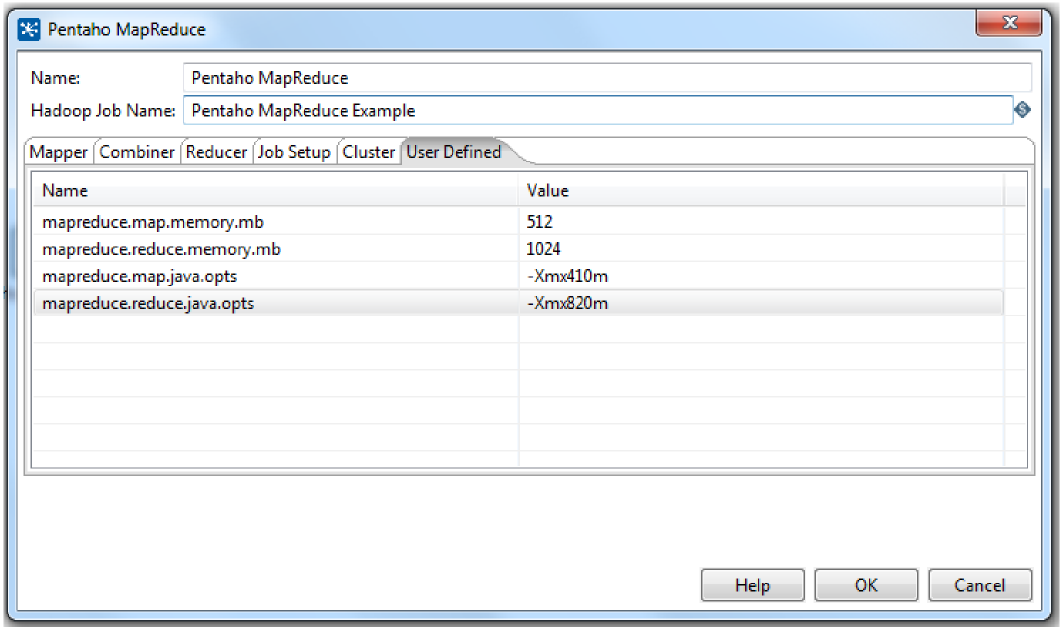
We recommend that you optimize and tune PMR executions to review and adapt your cluster configuration relative to YARN and MapReduce memory configuration settings.

Table 2: Cluster Memory Configuration Settings for YARN and MapReduce

Configuring File	Configuration Setting	Value Calculation (Recommendation)
`yarn-site.xml`	`yarn.nodemanager.resource.memory-mb`	= Containers * RAM-per-Container
`yarn-site.xml`	`yarn.scheduler.minimum-allocation-mb`	= RAM-per-Container
`yarn-site.xml`	`yarn.scheduler.maximum-allocation-mb`	= Containers * RAM-per-Container
`mapred-site.xml`	`mapreduce.map.memory.mb`	= RAM-per-Container
`mapred-site.xml`	`mapreduce.reduce.memory.mb`	= 2 * RAM-per-Container
`mapred-site.xml`	`mapreduce.map.java.opts`	= 0.8 * RAM-per-Container
`mapred-site.xml`	`mapreduce.reduce.java.opts`	= 0.8 * 2 * RAM-per-Container

Configuring File	Configuration Setting	Value Calculation (Recommendation)
`yarn-site.xml`	`yarn.app.mapreduce.am.resource.mb`	= 2 * RAM-per-Container
`yarn-site.xml`	`yarn.app.mapreduce.am.command-opts`	= 0.8 * 2 * RAM-per-Container

You can modify your parameters for a specific PMR execution using the **User Defined tab** in the Pentaho MapReduce entry. Doing this will overwrite the default configuration for the cluster, but only when this PMR is executed:



Name: Pentaho MapReduce

Hadoop Job Name: Pentaho MapReduce Example

Name	Value
mapreduce.map.memory.mb	512
mapreduce.reduce.memory.mb	1024
mapreduce.map.java.opts	-Xmx410m
mapreduce.reduce.java.opts	-Xmx820m

Figure 8: User Defined Tab

Hadoop Configuration

This section details common issues and solutions for Hadoop configuration that directly affects PDI execution. It includes standard Hadoop configuration procedures that any Hadoop administrator should be able to work with:

- Scheduling YARN Queues
- Security – Encryption Zones
- Security – Hive Scratch Directory
- Hive – JDBC URL

Scheduling YARN Queues

A queue is the fundamental unit of scheduling in YARN. By default, there is only one queue, `root`, which can access the full cluster resources. Queues affect the amount of resources that are available to applications running on the clusters.

The YARN scheduler runs the applications in first-in-first-out (FIFO) fashion. If the cluster or queue capacity is full, the applications are queued up.

Because PDI integrates natively with YARN and the MapReduce framework, it is subject to the same constraints from any YARN application that uses queue management policies. The YARN queue configuration will affect the Pentaho MapReduce applications' performance, because their execution will be throttled down.

Security – Encryption Zones

Hadoop provides transparent encryption for data at rest and in transit. The encryption is end-to-end:

- Only the client can encrypt and decrypt data.
- The client must have permission to access the correct decryption key to read the data.
- Keys are stored in the Key Management Server (KMS).

The correct access to the decryption and encryption keys must be provided beforehand, because PDI acts as a client for HDFS. If the user does not have the correct permissions to read or write a file, they will get an error saying they do not have access to the file.

It is possible to read and write to and from different encryption zones, but again, the user must have access to the proper encryption and decryption keys.

Security – Live Hatch

Hive uses temporary directories to store the intermediate and final outputs from its internal processes. This director defaults to `/tmp/hive` in HDFS.

When the final output from a Hive process is inside an encryption zone, the temporary directory must reside in that same encryption zone. The final step from the Hive processes moves a file into the final area, so the file encryption keys must match the encryption zone or the process will fail.

You can change the Hive scratch directory using a set command after the JDBC connection is established, or by changing the JDBC connection string:

```
set hive.exec.scratchdir=<scratchdir location>

jdbc:hive2://<server>:<port>/<db>;httpPath=cliservice;transportMode=http;pr
incipal=<principal>?hive.exec.scratchdir=<scratchdir location>
```

Hive – JDBC URL

You can customize the connection string in the PDI connection configuration. The connection string that you use to connect to Hive will depend on the security infrastructure deployed:

- No security

```
jdbc:hive2://<host>:<port>/<db>
```

- No security with HTTP transport

```
jdbc:hive2://<host>:<port>/<db>;transportMode=http;httpPath=<http_endpoint>
```

- With Kerberos authentication

```
jdbc:hive2://<host>:<port>/<db>;principal=<Server_Principal_of_HiveServer2>
```

- Kerberos authentication with HTTP transport

```
jdbc:hive2://<host>:<port>/<db>;transportMode=http;httpPath=<http_endpoint>
;principal=<Server_Principal_of_HiveServer2>
```

Related Information

Here are some links to information that you may find helpful while using this best practices document:

- JDBC in HiveServer2
- Pentaho Components Reference
- Securing JDBC and ODBC Clients' Access to HiveServer2 Using Apache Knox
- Simplifying User-Logs Management and Access in YARN
- YARN Commands

Finalization Checklist

This checklist is designed to be added to any implemented project that uses this collection of best practices, to verify that all items have been considered and reviews have been performed.

Name of the Project:____

Date of the Review:____

Name of the Reviewer:____

Item	Response	Comments
Did you place your cluster configuration in the proper SHIM folder?	YES_______ NO_______	
Did you configure your SHIM to place PDI libraries properly on a HDFS location?	YES_______ NO_______	
Did you have access to the Resource Manager to search for execution logs?	YES_______ NO_______	
Did you check cluster memory configuration to optimize PMR execution?	YES_______ NO_______	