Simple | Smart | Speed

Sqoop Performance Tuning Guidelines

Abstract

When you use Sqoop with Informatica Developer to transfer data between relational databases and Hadoop File System (HDFS), multiple factors impact the performance. You can optimize the performance by tuning Sqoop command line arguments, hardware parameters, database parameters, and Informatica mapping parameters. This article provides guidelines to help you to tune the performance of Sqoop when you transfer data between relational databases and HDFS.

Supported Versions

- Big Data Management 10.1

Table of Contents

Overview

Sqoop is a Hadoop command line program to process data between relational databases and HDFS through MapReduce programs. You can use Sqoop to import and export data.

This document describes the key Sqoop command line arguments, hardware, database, and Informatica mapping parameters that you can tune to optimize the performance of Sqoop. It also includes case studies that illustrate the impact that the tuning has on the Sqoop performance.

Note: The performance testing results listed in this article are based on observations in an internal Informatica environment using data from real-world scenarios. The Sqoop performance might vary based on individual environments and other parameters even when you use the same data.

Performance Tuning Areas

You can optimize the performance of Sqoop mappings by tuning the following areas:

- Sqoop command line arguments
- Hardware
- Database
- Informatica mapping

Tune the Sqoop Command Line Arguments

You can tune the following Sqoop arguments in the JDBC connection or Sqoop mapping to optimize performance:

- batch
- boundary-query
- compress or z
- direct
- Dsqoop.export.records.per.statement
- Enable primary key
- fetch-size
- num-mapper
- split-by

batch

Specifies that you can group the related SQL statements into a batch when you export data.

Use the following syntax:

```
--batch
```

You can configure the batch argument in the JDBC interface using the available API.

boundary-query

Specifies the range of values that you can import. You can use boundary-query if you do not get the desired results by using the split-by argument alone.

Use the following syntax:

```
--boundary-query select min(id), max(id) from <table name>
```

When you configure the boundary-query argument, you must specify the min(id) and max(id) along with the table name. If you do not configure the argument, Sqoop runs the following query:

```
select min (<split-by>), max(<split-by>) from <table name>
```

compress or z

When you configure the compress or z argument, you can compress the data approximately by 60% and reduce the amount of disk space required in the target. You can configure compression when the target storage is limited.

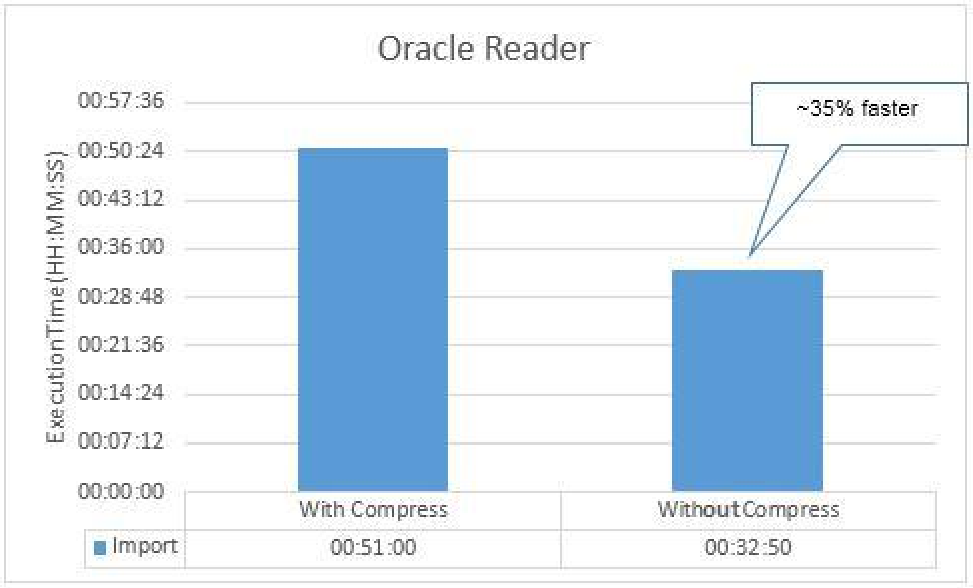
Use the following syntax to configure compression:

```
-z or --compress
```

However, when you use the compress or z argument, the overall execution time increases by 35%. To reduce the execution time, do not configure the compress or z argument.

Case Study

The following image shows the performance impact of disabling the compress argument:



	With Compress	Without Compress
Import	00:51:00	00:32:50

direct

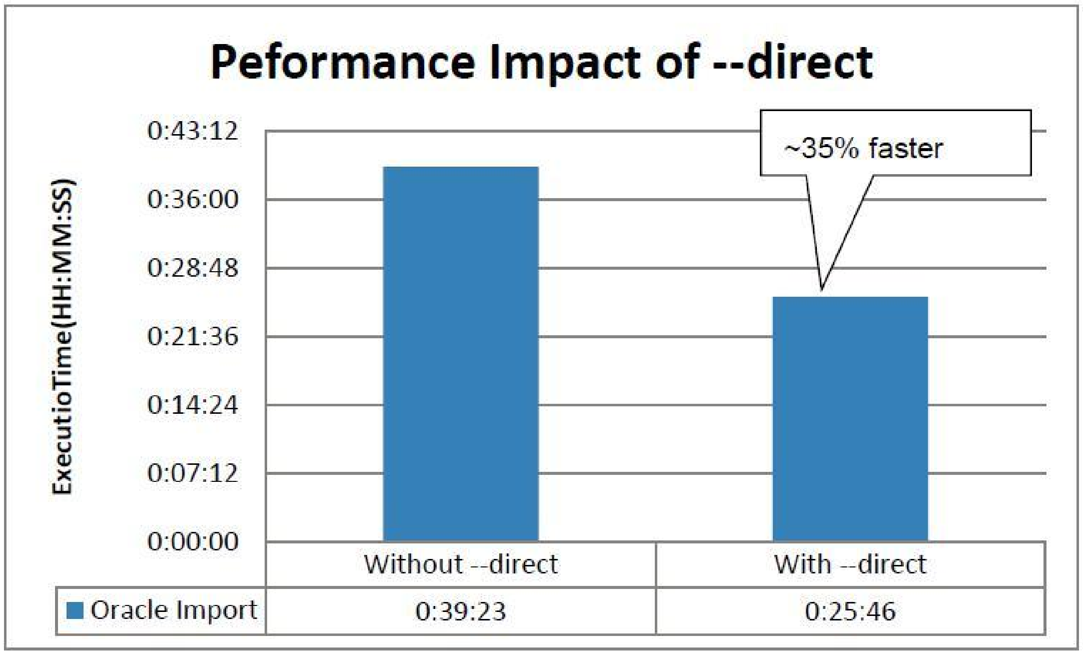
Specifies the direct import fast path when you import data from Oracle.

Use the following syntax:

```
    --direct
```

Case Study

The following image shows the performance impact of tuning the direct argument:



Dsqoop.export.records.per.statement

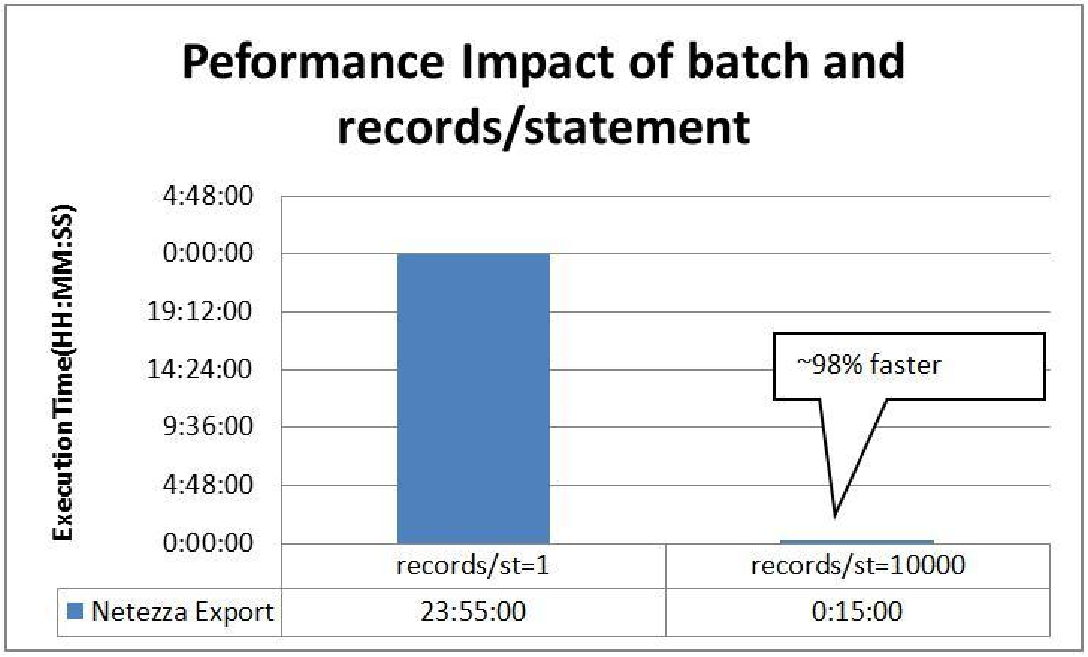
When you export data, you can configure the batch argument along with the Dsqoop.export.records.per.statement argument to insert multiple rows with a single statement.

Sqoop runs the following query when you configure the Dsqoop.export.records.per.statement argument.

```
INSERT INTO table VALUES (...), (...), (...),...;
```

Case Study

The following image shows the performance impact of tuning the -Dsqoop.export.records.per.statement argument.



	records/st=1	records/st=10000
■ Netezza Export	23:55:00	0:15:00

Enable Primary Key

Before you import data, you can enable the primary key constraint on the source table to optimize the performance while reading data from a source.

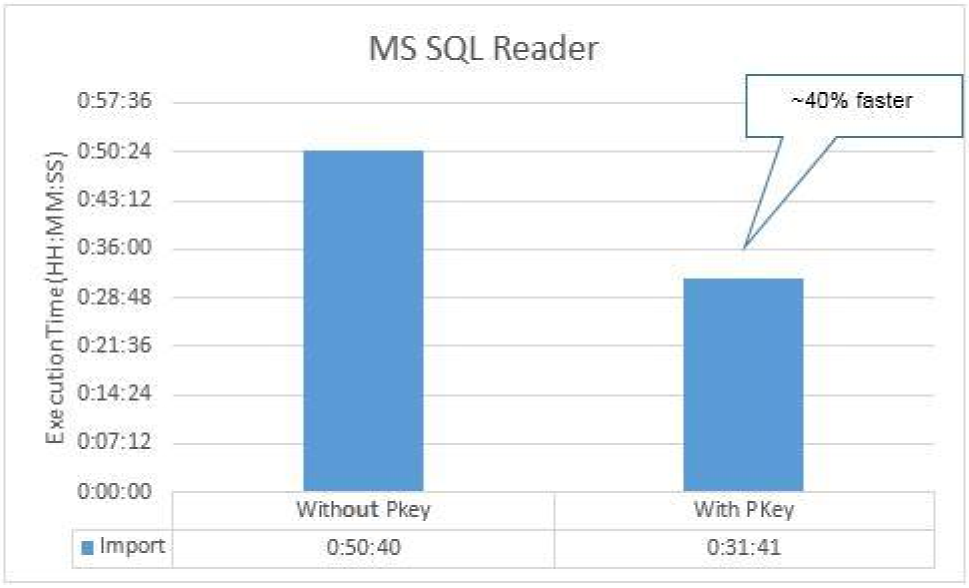
Use the following query to enable the primary key constraint:

```
alter table <table_name> add constraint <table_name_pk primary key {ID}>;
```

Note: You must remove the primary key-foreign key relationship when you insert data into a table.

Case Study

The following image shows the performance impact of enabling the primary key constraint:



fetch-size

Specifies the number of entries that Sqoop can import at a time.

Use the following syntax:

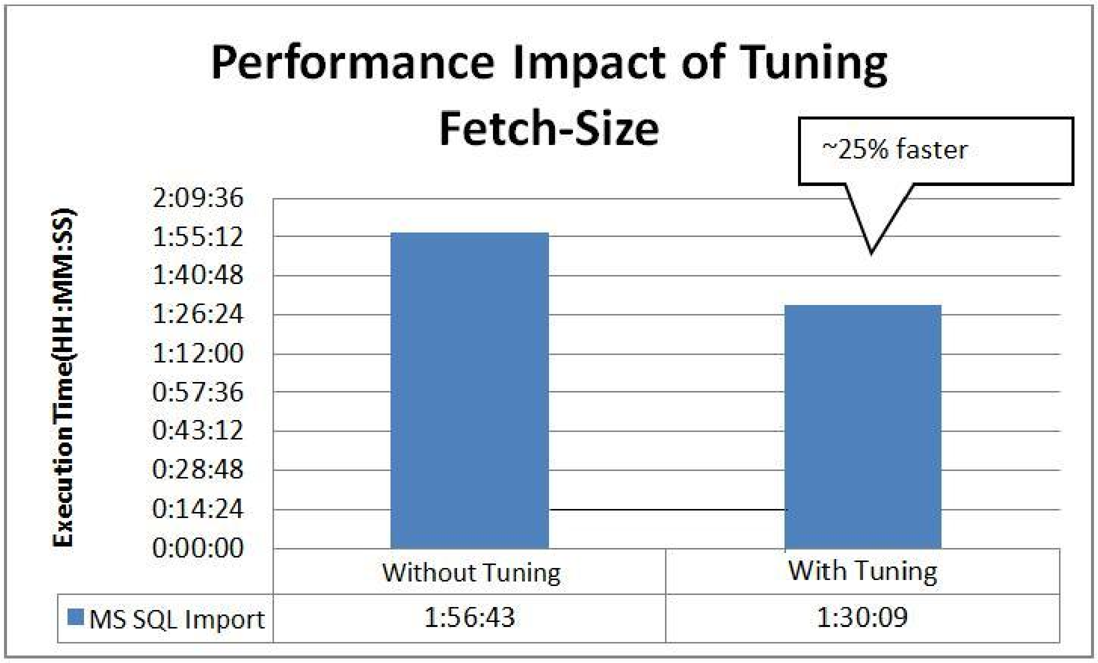
```
--fetch-size=<n>
```

Where <n> represents the number of entries that Sqoop must fetch at a time. Default is 1000.

You can increase the value of the fetch-size argument based on the volume of data that you want to read. Set the value based on the available memory and bandwidth.

Case Study

The following image summarizes the impact of tuning the Sqoop fetch-size argument to 10000:



num-mappers

Specifies the number of map tasks that can run in parallel.

Use the following syntax:

```
--num-mappers <number of map tasks>
```

Default is 4.

To optimize performance, set the number of map tasks to a value lower than the maximum number of connections that the database supports.

split-by

Specifies the column name based on which Sqoop must split the work units.

Use the following syntax:

```
--split-by <column name>
```

Note: If you do not specify a column name, Sqoop splits the work units based on the primary key.

Tune the Hardware

You can tune the following hardware parameters to optimize the performance:

- CPU frequency
- NIC card ring buffer size

Tune the Database

To optimize the performance of relational databases, perform the following tasks:

- Analyze database statistics to fine tune queries.
- Maintain different physical disks for different tablespaces.
- Determine the expected database growth.
- Use the EXPLAIN PLAN statement to fine tune queries.
- Avoid foreign key constraints.
- Drop indexes before loading data.
- Optimize the JDBC connection URL parameters.
- Determine the best connectivity interface.

Tune the Mapping

You can tune the following parameters at the Informatica mapping level to optimize the performance of relational databases:

- Data movement mode
- Data type mapping
- Optimizer hints
- Partitions
- Ports precision
- Pushdown optimization

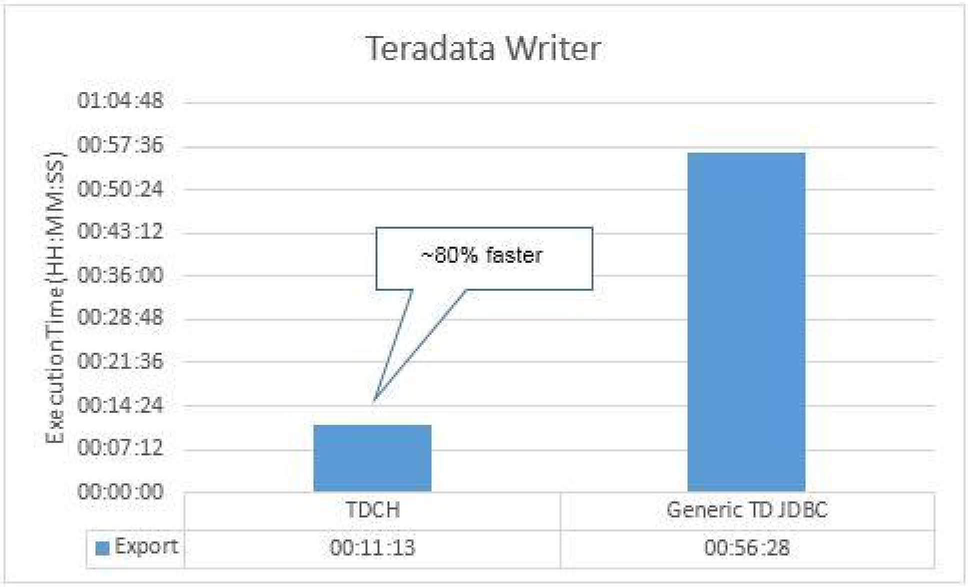
For more information about tuning the mapping parameters, see the *Informatica Performance Tuning Guide*.

Performance Improvement with the Third-Party Teradata TDCH Driver

The Teradata TDCH driver uses native drivers internally, which results in faster read and write operations when compared to the generic Teradata JBDC driver.

Use the Teradata TDCH driver for improved performance.

The following image shows the performance improvement when you use the Teradata TDCH driver when compared to the generic Teradata JDBC driver:



Authors

Krishna Prabhakar Devarakonda

Siddiq Hussain

Anu Chandrasekharan

Pranav Sharma