Arenadata™

Версия - v1.0.1

Описание коннектора

2021 Arenadata™

Оглавление

1 Архитектура коннектора 3

2 Чтение данных 4

3 Запись данных 6

4 Поддерживаемые движки 8

5 Поддерживаемые типы данных 9

Arenadata QuickMarts (ADQM) — столбцовая СУБД для онлайн-обработки запросов, основанная на СУБД с открытым исходным кодом — **ClickHouse**.

Apache Spark — фреймворк с открытым исходным кодом для реализации распределенной обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов **Hadoop**.

ADQM-Spark коннектор предоставляет возможность высокоскоростного параллельного обмена данными между **Apache Spark** и **Arenadata QuickMarts**.

Глава 1

Архитектура коннектора

Каждое Spark-приложение состоит из управляющего процесса — драйвера (Driver) — и набора распределенных рабочих процессов — исполнителей (Executors).

Далее представлена *общая компонентная диаграмма взаимодействия*.

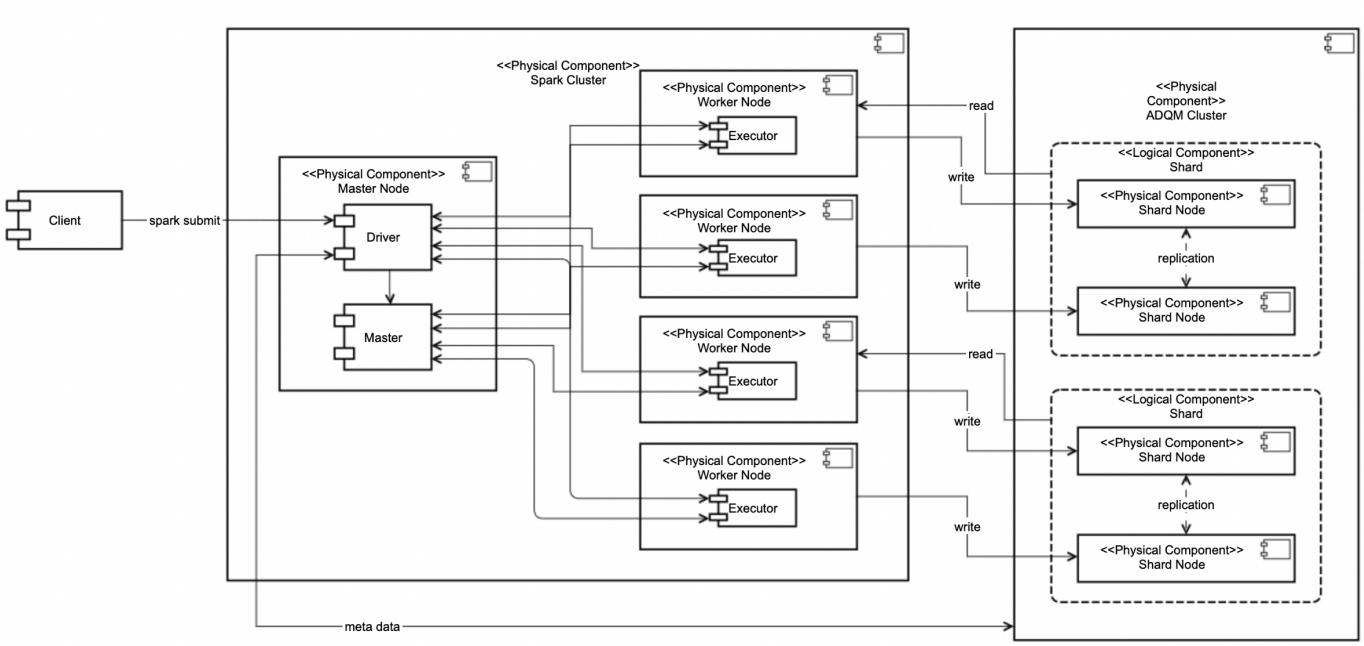


Рис.1.1.: Общая компонентная диаграмма взаимодействия

Глава 2

Чтение данных

Для загрузки таблицы **ADQM** в **Spark** на первичном этапе коннектор инициализирует драйвер через строку подключения, которая задана опцией `spark.adqm.url`. При этом если указана опция `spark.adqm.cluster.name`, драйвер, подключаясь к `spark.adqm.url` с помощью **JDBC**, также считывает метаинформацию о всех шардах и репликах указанного кластера **ADQM**. Эти метаданные содержат иформацию о шардах и репликах, что в дальнейшем позволяет эффективно в параллель считывать данные с таблиц шардов по имеющимся Spark-исполнителям.

Данные в **ADQM** хранятся на шардах, и Spark-приложение, используя коннектор, может получать данные с каждого шарда в одну или несколько Spark-партиций. В случае чтения из Distributed-таблиц, чтение происходит в параллель на каждом из шардов из соответствующей MergeTree-таблицы.

На текущий момент существует три режима партиционирования:

Important: Информация справедлива для случаев: когда опция `spark.adqm.cluster.name` указана и не указана

- Согласно *shard id*, при котором данные из таблиц ADQM считываются и распределяются по партициям Spark в соответсвии с номером шарда. В таком случае количество Spark-партиций соответсвует числу активных шардов ADQM. Данный режим партиционирования используется по умолчанию и не требует указания дополнительных параметров;
- Согласно указанной колонке в `spark.adqm.partition.column` и указанному количеству партиций `spark.adqm.partition.count`, при котором предполагается разбиение данных по Spark-партициям согласно диапазонам значений указанной колонки на указанное число партиций. В таком случае количество Spark-партиций соответсвует количеству диапазов. Необходимо указать соответсвующие параметры (options). Данный режим имеет ограничения: в качестве колонки могут быть использованы только целочисленные и дата/время поля таблицы;
- Согласно указанной колонке в `spark.adqm.partition.column`, при которой предполагается разбиение данных по Spark-партициям в соответствии с уникальными значениями указанной колонки. В таком случае количество Spark-партиций соответствует количеству уникальных значений заданной колонки. Необходимо указать соответствующий параметр (options). Рекомендуется использовать данный режим при небольшом, ограниченном наборе значений.

При такой архитектуре на каждый Spark-исполнитель назначается задача по обработке данных и соответствующая ему партиция из ранее созданных. Обмен данных производится в параллель с каждого шарда для каждой партиции. Если указана опция `spark.adqm.cluster.name`, обмен производится с каждым из шардов данного кластера, в ином случае обмен происходит с шардами, указанными в опции `spark.adqm.url`.

Диаграмма чтения данных в ADQM-Spark коннекторе:

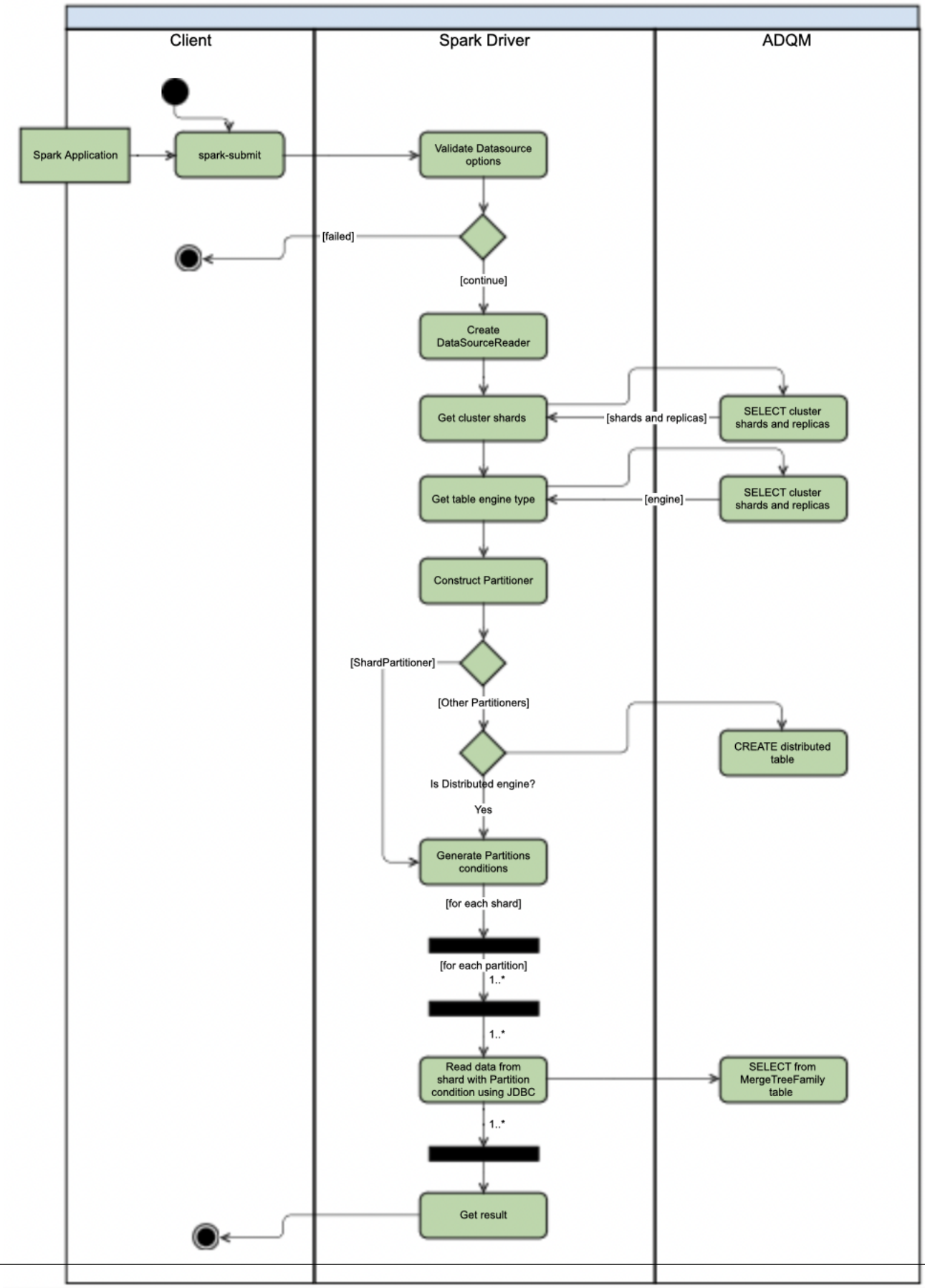


Рис.2.1.: Диаграмма чтения данных

Глава 3

Запись данных

Для загрузки таблицы из **Spark** в **ADQM** на первичном этапе коннектор инициализирует драйвер через строку подключения, которая задана опцией `spark.adqm.url`. При этом если указана опция `spark.adqm.cluster.name`, драйвер, подключаясь к `spark.adqm.url` с помощью **JDBC**, также считывает метаинформацию о всех шардах и репликах указанного кластера **ADQM**. Для каждого из шардов в зависимости от процесса записи происходит его подготовка к загрузке.

На текущий момент поддерживаются следующие режимы записи:

- Добавление (append), при котором происходит дозапись в целевую таблицу. При указаннии опции `spark.adqm.cluster.name` осуществляется попытка записи в каждый из шардов кластера, в ином случае – только в те шарды, которые указаны в опции `spark.adqm.url`. Если на каком-либо из шардов отсутствует таблица, в которую предполагается дозапись, процесс загрузки завершается ошибкой. В данном режиме на всех шардах должна быть предварительно создана целевая таблица;
- Ошибка, если целевая таблица существует (errorIfExists). В данном режиме предполагается, что целевой таблицы нет, она создается в процессе загрузки. Необходимо указать информацию о движке создаваемой целевой таблицы с помощью опции `spark.adqm.create.table.engine`. Если указана опция `spark.adqm.cluster.name`, то целевая таблица с указанным движком создается на каждом из шардов кластера, в ином случае целевая таблица создается на каждом из шардов, заданных в опции `spark.adqm.url`;
- Перезапись (overwrite), при которой целевая таблица в зависимости от `spark.adqm.table.truncate` либо удаляется полностью, либо в ней вычищаются все данные перед процессом записи. В случае полного удаления таблицы процесс записи аналогичен режиму errorIfExists, в случае вычищения - алаогичен режиму append.

Также доступен ряд опций `spark.adqm.create.table`, позволяющих указать дополнительные настройки при создании целевой таблицы (options).

Далее для каждой Spark-партиции создается задача по загрузке данных. Обмен данных производится в параллель для каждого шарда. Если указана опция `spark.adqm.cluster.name`, обмен производится с каждым из шардов данного кластера, в ином случае обмен происходит с шардами, указанными в опции `spark.adqm.url`.

Диаграмма записи данных в ADQM-Spark коннекторе:

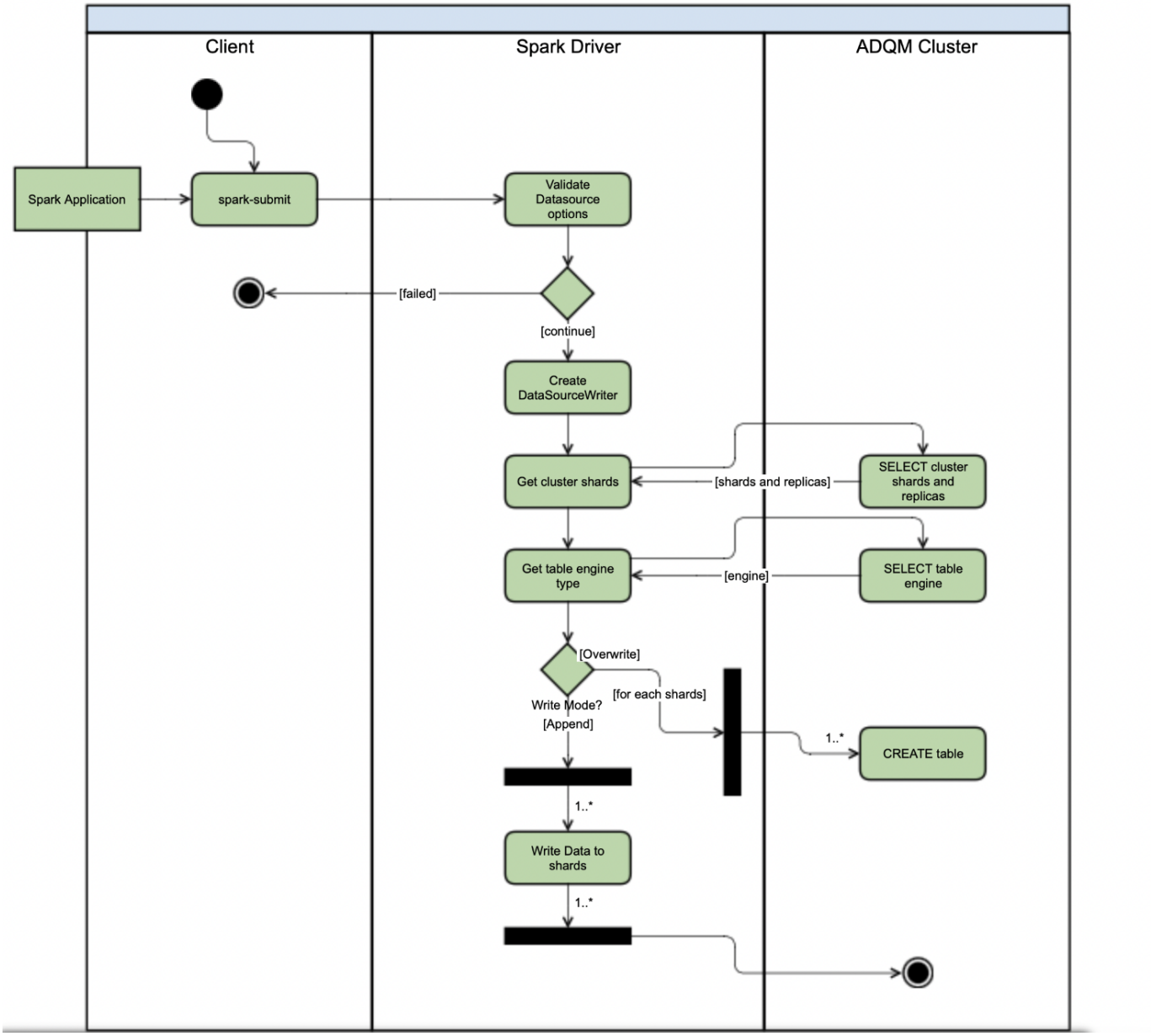


Рис.3.1.: Диаграмма записи данных

Глава 4

Поддерживаемые движки

Движок (Engine) — сущность **ADQM**, которая определяет, как данные хранятся в **Arenadata QuickMarts** и каким образом их записывать, откуда их читать и какой к ним имеется доступ, а также задает параметры запросов, индексов и репликации.

Движки семейства *MergeTree* предназначены для быстрой записи огромного объема данных по частям относительно заданных правил.

Движок *Distributed* не хранит данные, а позволяет обрабатывать запросы распределенно на нескольких серверах, выполняя параллельное чтение автоматически.

Поддерживаемые движки:

- Distributed;
- MergeTreeFamily:
  - AggregatingMergeTree;
  - CollapsingMergeTree;
  - ReplicatedAggregatingMergeTree;
  - ReplicatedCollapsingMergeTree;
  - ReplicatedMergeTree;
  - ReplicatedReplacingMergeTree;
  - ReplicatedSummingMergeTree;
  - ReplicatedVersionedCollapsingMergeTree;
  - ReplacingMergeTree;
  - SummingMergeTree;
  - VersionedCollapsingMergeTree.

Глава 5

Поддерживаемые типы данных

Поддерживаемые типы данных из **ADQM** в **Spark**:

Таблица5.1.: Поддерживаемые типы данных из ADQM в Spark

Тип данных ADQM	Тип данных Spark
Date	DateType
DateTime	TimestampType
DateTime64	TimestampType
Decimal	DecimalType
Decimal128	DecimalType
Decimal256	DecimalType
Decimal32	DecimalType
Decimal64	DecimalType
Float32	FloatType
Float64	DoubleType
IPv4	LongType
Int16	IntegerType
Int32	IntegerType
Int64	LongType
Int8	ShortType
String	StringType
UInt16	IntegerType
UInt32	LongType
UInt64	LongType
UInt8	ShortType
UUID	StringType
float8	DoubleType

Поддерживаемые типы данных из **Spark** в **ADQM**:

Таблица5.2.: Поддерживаемые типы данных из Spark в ADQM

Тип данных Spark	Тип данных ADQM
BinaryType	String
BooleanType	UInt8
ByteType	Int8
DateType	Date
DecimalType	Decimal
DoubleType	Float64
FloatType	Float32
IntegerType	Int32
LongType	Int64
ShortType	Int16
StringType	String
TimestampType	DateTime64