syntax = "proto3";

package tensorflow.data;

option go\_package = "github.com/tensorflow/tensorflow/tensorflow/go/core/protobuf/for\_core\_protos\_go\_proto";

// Next tag: 2

message ProcessingModeDef {

// Specifies how data is sharded among tf.data service workers.

enum ShardingPolicy {

// No sharding will be performed. Each worker produces the entire dataset

// without any sharding. With this mode, the best practice is to shuffle the

// dataset nondeterministically so that workers process the dataset in

// different orders.

OFF = 0;

// The input dataset is dynamically split among workers at runtime. Each

// worker gets the next split when it reads data from the dispatcher. There

// is no fixed sharding with this mode.

DYNAMIC = 1;

// The following are static sharding policies. The semantics are similar to

// `tf.data.experimental.AutoShardPolicy`. These policies require:

// \* The tf.data service cluster has a fixed size, and you need to specify

// the workers in DispatcherConfig.

// \* Each client only reads from the local tf.data service worker.

//

// Shards by input files (each worker will get a set of files to process).

// When this option is selected, make sure that there is at least as many

// files as workers. If there are fewer input files than workers, a runtime

// error will be raised.

FILE = 2;

// Shards by elements produced by the dataset. Each worker will process the

// whole dataset and discard the portion that is not for itself. Note that

// for this mode to correctly partitions the dataset elements, the dataset

// needs to produce elements in a deterministic order.

DATA = 3;

// Attempts FILE-based sharding, falling back to DATA-based sharding on

// failures.

FILE\_OR\_DATA = 4;

// Looks for the presence of `shard(SHARD\_HINT, ...)` which is treated as a

// placeholder to replace with `shard(num\_workers, worker\_index)`.

HINT = 5;

}

ShardingPolicy sharding\_policy = 1;

}

// tf.data service deployment mode.

enum DeploymentMode {

DEPLOYMENT\_MODE\_UNSPECIFIED = 0;

// tf.data service workers colocate with TF workers.

DEPLOYMENT\_MODE\_COLOCATED = 1;

// tf.data service workers run in dedicated tf.data hosts.

DEPLOYMENT\_MODE\_REMOTE = 2;

// tf.data service workers run in colocated TF hosts and dedicated tf.data

// hosts.

DEPLOYMENT\_MODE\_HYBRID = 3;

}

// Metadata related to tf.data service datasets.

// Next tag: 4

message DataServiceMetadata {

oneof optional\_element\_spec {

// Serialized element spec.

bytes element\_spec = 1;

}

enum Compression {

COMPRESSION\_UNSPECIFIED = 0;

// No compression.

COMPRESSION\_OFF = 1;

// Snappy compression as defined in tensorflow/core/platform/snappy.h.

COMPRESSION\_SNAPPY = 2;

}

Compression compression = 2;

// Cardinality of the dataset.

int64 cardinality = 3;

}

// Data service config available to the client through GetDataServiceConfig RPC.

// Next tag: 2

message DataServiceConfig {

DeploymentMode deployment\_mode = 1;

}