syntax = "proto3";

package tensorflow.data;

import "tensorflow/core/framework/model.proto";

option go\_package = "github.com/tensorflow/tensorflow/tensorflow/go/core/framework/dataset\_options\_go\_proto";

// Represents the type of auto-sharding we enable.

enum AutoShardPolicy {

// AUTO: Attempts FILE-based sharding, falling back to DATA-based sharding.

AUTO = 0;

// FILE: Shards by input files (i.e. 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 = 1;

// DATA: 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 = 2;

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

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

HINT = 3;

// OFF: No sharding will be performed.

OFF = -1;

}

// next: 5

message AutotuneOptions {

// Whether to automatically tune performance knobs.

oneof optional\_enabled {

bool enabled = 1;

}

// When autotuning is enabled (through autotune), determines the CPU budget to

// use. Values greater than the number of schedulable CPU cores are allowed

// but may result in CPU contention.

oneof optional\_cpu\_budget {

int32 cpu\_budget = 2;

}

// When autotuning is enabled (through autotune), determines the RAM budget to

// use. Values greater than the available RAM in bytes may result in OOM. If

// 0, defaults to half of the available RAM in bytes.

oneof optional\_ram\_budget {

int64 ram\_budget = 3;

}

// When autotuning is enabled (through autotune), determines the algorithm to

// use. If not explicitly set by user, autotuning will follow HILL\_CLIMB

// algorithm but has more flexibility to tune parameters more aggressively,

// in which case the behavior is implementation specific and may change over

// time.

oneof optional\_autotune\_algorithm {

model.AutotuneAlgorithm autotune\_algorithm = 4;

}

}

// next: 2

message CardinalityOptions {

enum ComputeLevel {

CARDINALITY\_COMPUTE\_UNSPECIFIED = 0;

// Cardinality will only be computed if it can be determined in a cheap

// manner (ie. without reading from file sources). If the cardinality would

// be nontrivial to compute, Cardinality() will return UNKNOWN\_CARDINALITY.

CARDINALITY\_COMPUTE\_LOW = 1;

// Moderate effort will be made to determine cardinality, such as reading

// index data from source files. If significant work is needed to compute

// cardinality (e.g. reading entire source file contents or executing user

// defined functions), Cardinality() will return UNKNOWN\_CARDINALITY.

CARDINALITY\_COMPUTE\_MODERATE = 2;

}

ComputeLevel compute\_level = 1;

}

// next: 3

message DistributeOptions {

AutoShardPolicy auto\_shard\_policy = 1;

// The number of devices attached to this input pipeline.

oneof optional\_num\_devices {

int32 num\_devices = 2;

}

}

// next: 18

message OptimizationOptions {

// Whether to apply default graph optimizations. If False, only graph

// optimizations that have been explicitly enabled will be applied.

oneof optional\_apply\_default\_optimizations {

bool apply\_default\_optimizations = 1;

}

reserved 2;

reserved 3;

reserved 4;

reserved 5;

// Whether to fuse filter transformations.

oneof optional\_filter\_fusion {

bool filter\_fusion = 6;

}

// NOTE: field id 7 deleted in June 2021.

reserved 7;

// NOTE: field id 8 deleted in June 2021.

reserved 8;

// Whether to fuse map and batch transformations.

oneof optional\_map\_and\_batch\_fusion {

bool map\_and\_batch\_fusion = 9;

}

// Whether to fuse map and filter transformations.

oneof optional\_map\_and\_filter\_fusion {

bool map\_and\_filter\_fusion = 10;

}

// Whether to fuse map transformations.

oneof optional\_map\_fusion {

bool map\_fusion = 11;

}

// Whether to parallelize stateless map transformations.

oneof optional\_map\_parallelization {

bool map\_parallelization = 12;

}

// NOTE: field id 13 deleted in June 2021.

reserved 13;

// Whether to eliminate no-op transformations.

oneof optional\_noop\_elimination {

bool noop\_elimination = 14;

}

// Whether to parallelize copying of batch elements. This optimization is

// highly experimental and can cause performance degradation (e.g. when the

// parallelization overhead exceeds the benefits of performing the data copies

// in parallel). You should only enable this optimization if a) your input

// pipeline is bottlenecked on batching and b) you have validated that this

// optimization improves performance.

oneof optional\_parallel\_batch {

bool parallel\_batch = 15;

}

// Field id 16 was removed in 06/2021.

reserved 16;

// Whether to fuse shuffle and repeat transformations.

oneof optional\_shuffle\_and\_repeat\_fusion {

bool shuffle\_and\_repeat\_fusion = 17;

}

}

// next: 3

message ThreadingOptions {

// If set, it overrides the maximum degree of intra-op parallelism.

oneof optional\_max\_intra\_op\_parallelism {

int32 max\_intra\_op\_parallelism = 1;

}

// If set, the dataset will use a private threadpool of the given size.

oneof optional\_private\_threadpool\_size {

int32 private\_threadpool\_size = 2;

}

}

// Represents how to handle external state during serialization.

enum ExternalStatePolicy {

POLICY\_WARN = 0;

POLICY\_IGNORE = 1;

POLICY\_FAIL = 2;

}

// Message stored with Dataset objects to control how datasets are processed and

// optimized.

//

// next: 8

message Options {

// Whether the outputs need to be produced in deterministic order.

oneof optional\_deterministic {

bool deterministic = 1;

}

// The distribution strategy options associated with the dataset.

AutotuneOptions autotune\_options = 7;

// The distribution strategy options associated with the dataset.

DistributeOptions distribute\_options = 2;

// The optimization options associated with the dataset.

OptimizationOptions optimization\_options = 3;

// Whether to introduce 'slack' in the last `prefetch` of the input pipeline,

// if it exists. This may reduce CPU contention with accelerator host-side

// activity at the start of a step. The slack frequency is determined by the

// number of devices attached to this input pipeline.

oneof optional\_slack {

bool slack = 4;

}

// The threading options associated with the dataset.

ThreadingOptions threading\_options = 5;

// This option can be used to override the default policy for how to handle

// external state when serializing a dataset or checkpointing its iterator.

// There are three settings available - IGNORE: External state is ignored

// without a warning; WARN: External state is ignored and a warning is logged;

// FAIL: External state results in an error.

oneof optional\_external\_state\_policy {

ExternalStatePolicy external\_state\_policy = 6;

}

}