视频分析平台（Goku）后端设计

总体流程

1、利用Init组件事先将model group与model以及model group与topic的关系写入数据库。
2、用户手工启动worker，根据指定的model group，预先加载所有model group里的所有model。
3、通过 web 界面发送annotate video request，获取该video指定的所有models，并计算出所有models对应的所有model group，利用消息组件，将任务调度至指定的 workers，最终保存结果至数据库。

术语介绍

1、video request

1个video request里至少需要包括该video的path以及解析该video所需的所有models。

2、model

model指深度学习模型，可以对视频文件的每帧进行inference，最终，1个model产生1个annotation结果。

3、model group

model group是“批量model”的概念。1个model group由1个或多个model组成。model group包含哪些model可以事先初始化好（本期目标），或者由用户来指定（远期目标，通过topic来实现，因为1个model group对应1个topic，因此，通过指定model所属的topic确定model所属的model group）。

一般来讲，推荐的方式是1个model group用于解析1类视频，由解析该类视频所需的1个或多个models组成。如用来解析暴力恐怖的1个或多个models可以组成1个model group。用来解析淫秽色情的1个或多个models可以组成另1个model group。

不仅如此，同1类的models可以根据需要，分成多个model group。例如，若分析暴力恐怖的model有20个，可以每10个组成1个model group，最终形成2个model group。

若1个视频既暴力又淫秽，则根据用户选择，可能需要1个或者2个model group里的model来分析。

每个model group监听1种kafka topic。

4、worker

worker是用于加载视频和模型，并进行inference的组件，worker以model group为最小单位加载模型。当worker组建开始running的时候，就把model group里的所有model加载起来，并监听model group对应的topic。

1个worker一次性可加载多个model group，相应的，会监听多个topic。

最终，worker将解析的结果按照每个model1行数据的形式写入数据库。

5、message

message是kafka转发的消息的具体内容。通常被封装成1种定义好的proto类型，以[]bytes的形式被发送或者接收。

6、topic

topic是用来指明让kafka（消息组件）将message路由到哪个worker的依据。1个worker可能监听1个或多个topic。当kafka收到topic+message时，根据topic内容，将message路由到该topic下，监听该topic的worker会收到该消息（PS：暂时忽略kafka中partition和consumer group的影响）。

术语关系

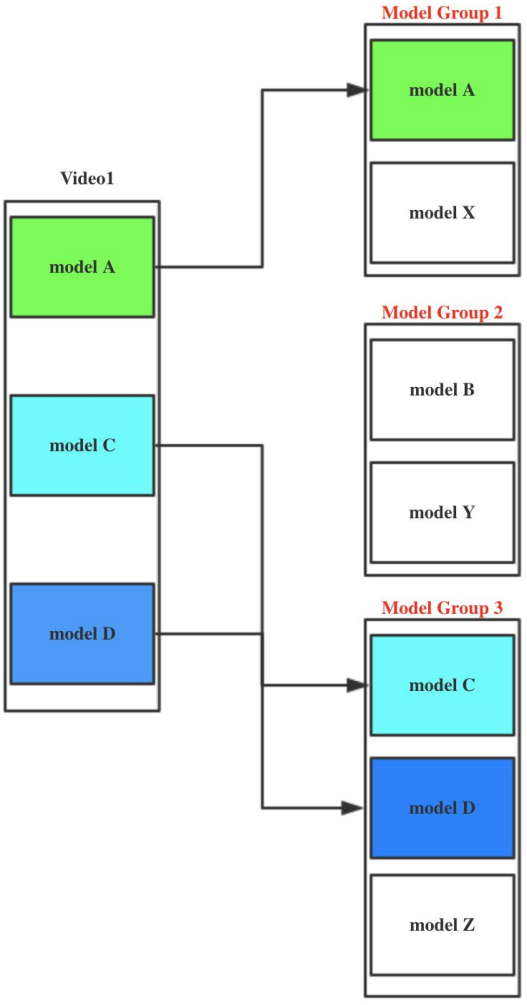
1、video request : models = 1:N

1个video request包括1个video，该video可以由多个model来inference，最终1个model出1个结果写入数据库。

该视频对应的所有models，需要计算得出对应的model group。最终按照最佳匹配原则计算出的model group可能是1个或者多个。

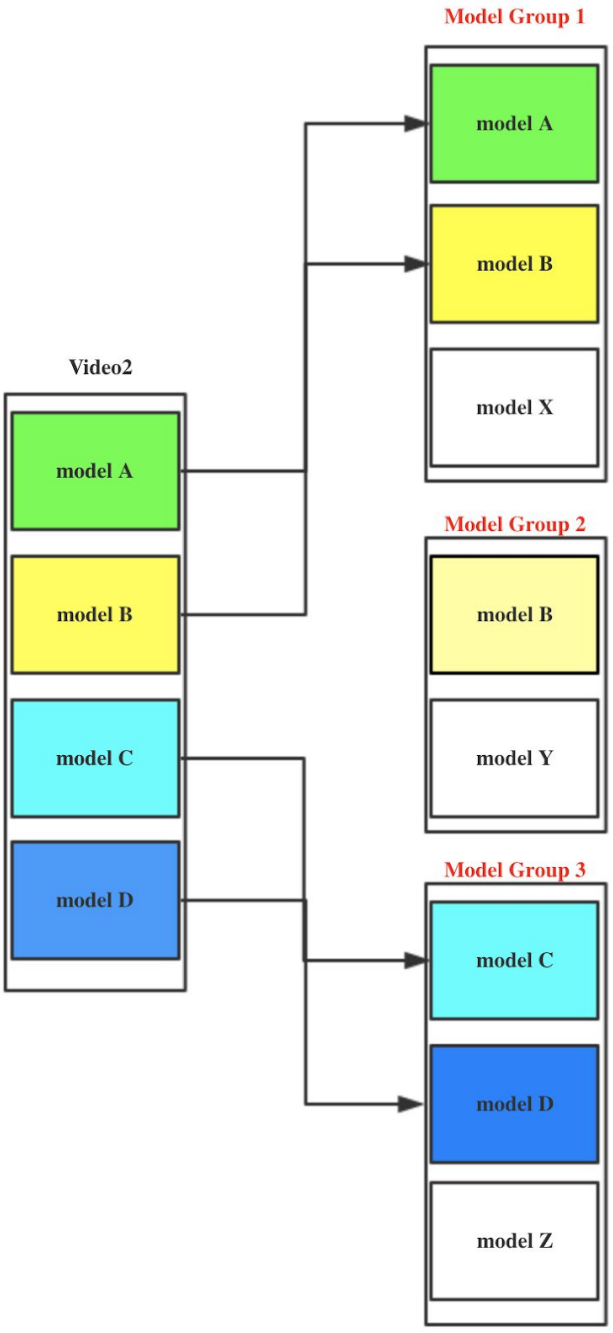
情况1：

如下图所示：video1需要model A、C和 D来分析，在model group 1中，包括model A、model B和model X，model group 2中包括model B和model Y，model group 3中包括model C、model D和model Z。根据计算，最后的model group有2个，model group 1和model group 3。



情况2：

model group的计算根据**最佳匹配原则**进行计算，如下图所示，video2需要用到model A、B、C、和 D 来解析。model group 1和model group 2里都包含model B，但最终计算的model group只有model group 1和model group 3。



2、model group : topic = 1:1

1个model group对应kafka的一个topic。知道topic就一定知道model group的信息。topic需要预先初始化好。

3、model group : model = 1:N

1个model group中包括N个model。不同的model group里，可以有相同的model。

4、worker : model group = 1:N

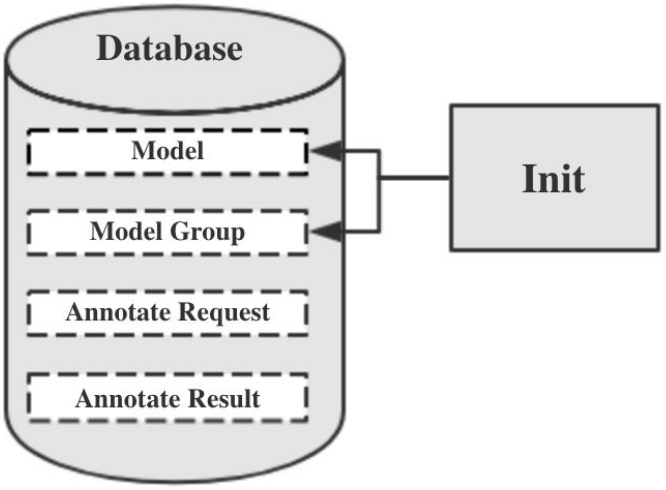
1个worker可以**同时**加载N个model group。worker根据每个model group里包含的 model，加载所有的model。若不同的model group里包含了相同的model，则该model只加载1次。

5、worker : topic = 1:N

1个worker可以**同时**监听多个topic，即一个worker可以收到多个topic发送过来的消息。

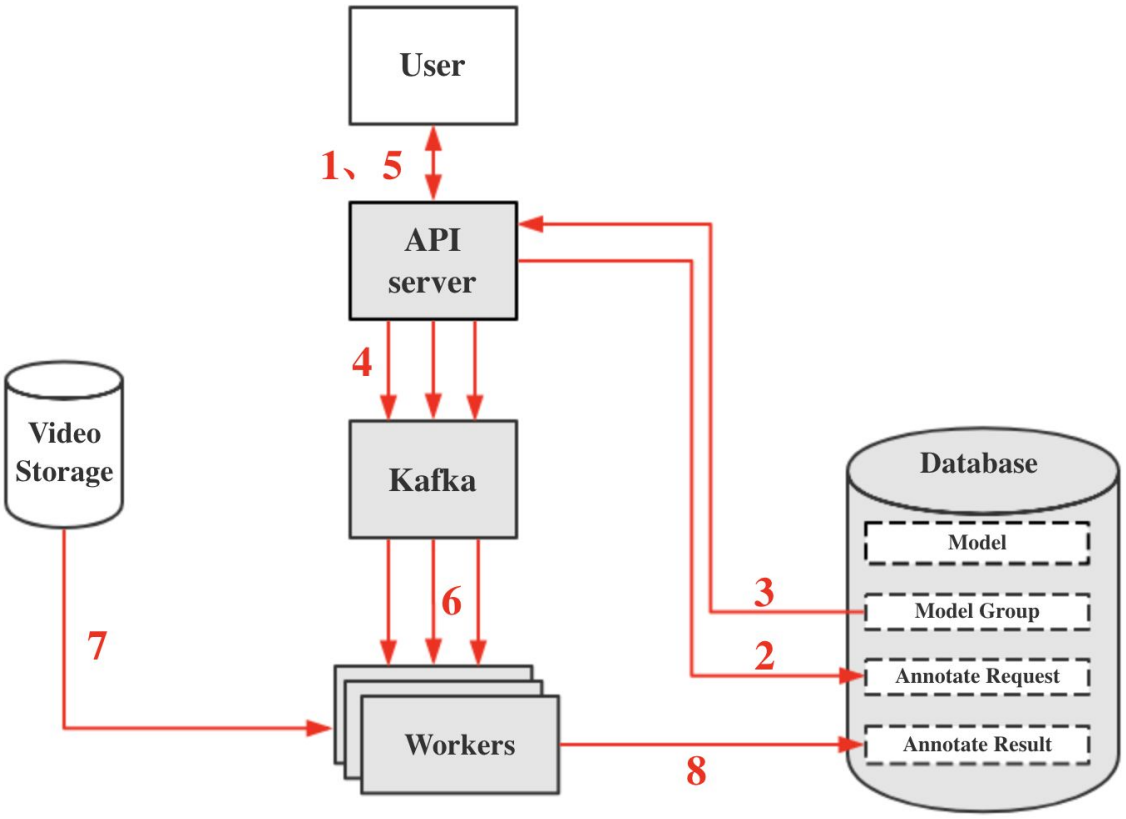
具体业务流程

1、Init流程



启动Init组件。Init组件将model与model group的对应关系及model group与topic的对应关系插入到数据库的Model和Model group表中。

2、Annotate Video流程



- 1：用户将视频解析请求发送给API server
- 2：API server收到请求后，将request信息写入数据库。
- 3：API server根据request信息，读取model group表，计算出分析该视频需要的model group。
- 4：API server根据计算的结果，向指定的 topic 发送内容为request id 和所有model的 id 信息的消息。
- 5：若所有topic的消息发送成功，API Sever向用户返回内容为request id的response。
- 6：Worker收到消息后，将request id、video path和所有model id加入task queue
- 7：Worker根据request info，从指定位置读取视频文件。
- 8：每个model分析完，Worker组件将结果写入数据库。

详细设计

DB设计（MySQL）

表设计

Model 表		
model_id	varchar	无
name	varchar	便于分页search
tags	*varchar*	*属于哪些model group，即哪些topic，便于分页search*
model_info	medium blob	以proto形式存储model name及model path等信息。

Model Group 表		
model_group_id	varchar	无
model_group_info	medium blob	以proto形式存储该model group 对应的 topic name 及包含哪些model

Task 表		
task_id	varchar	主键
task_info	medium blob	以proto形式存储task信息，包括视频路径、models等信息
task_start_time	int64	unit: ms

task_end_time	int64	unit: ms
task_status	int	存储task状态信息，为了select的时候过滤。 0代表PENDING 1代表RUNNING 2代表SUCCESS 3代表FAILED

Task Result 表		
result_id	varchar	无
task_id	varchar	request id
model_id	varchar	该request中的一个model的id
start_time	int64	unit: ms
end_time	int64	unit: ms
result_info	medium blob	以proto形式存储该model的inference结果（proto设计by黄亮）

DB函数设计(需要参照gitlab上具体的proto设计)

1、InsertModel(model) error

2、GetModels( )[]model, error

3、InsertModelGroup(modelGroup) error

4、GetModelGroups()[]modelGroup, error

5、InsertTask(task) error

6、GetTask(task_id) task, error　　Go/Python

7、InsertTaskResult(task, model_id,taskResult) error　　Python

//taskResult存储的是storage.proto里的ModelOutput

8、GetTaskResult(taskId) []ModelOutput, error

9、UpdateTask(task)error　　Python

10、GetTasks(offset, limit int, statusCode int)[]task,**total**，error

11、UpdateTaskTimeAndStatus(task_id)　　Go

Topic分类

暴力、恐怖、涉政、淫秽、传教、表情、姿态、年龄。。。。。。。

Proto设计

```
message Model {
   string id = 1;
   string name = 2;
   string model_dir = 3;
   Loader  loader = 4;
}

message Loader {
  CaffeLoader caffe_loader = 1;
  Tensorloader tf_loader = 2;
}

// 模型在load时需要指定其默认处理的图像尺寸，Inference时图片会resize成该指定的尺寸
message ImageSize {
  int32 width = 1;
  int32 height = 2;
  int32 channels = 3;
}

message CaffeLoader {
  string model_def = 1;
  string pretrained_model = 2;
  ImageSize image_size = 3;
  string mean_file = 4;
  double input_scale = 5;
  double raw_scale = 6;
  string label_text = 7;
```

```
  repeated string other_params = 8;
}

Message TensorLoader {
  string ckpt_path = 1;
  ImageSize image_size = 2;
  repeated string other_params = 3;
}

message ModelGroup {
  string topic_name = 1;
  repeated  string model_ids = 2;
}
  // 用于API sever向指定topic发送message
message  RequiredModelsMsg {
  string request_id = 1;
  repeated string model_ids = 2;
}
```

API设计

AnnotateVideo

```
message AnnotateVideoRequest {
  string video_path = 1;
  repeated Model models = 2;    //该视频需要哪些model来inference
}
message AnnotateVideoResponse {
  string request_id = 1;
}
rpc AnnotateVideo(AnnotateVideoRequest) returns (AnnotateVideoResponse) {
  option (google.api.http) = {
    post: "/annotate_video"
    body: "*"
```

```
  };
}
```

GetResult

```
message GetResultRequest {
  string request_id = 1;
}
message GetResultResponse {
  repeated ModelOutput outputs = 1; // ModelOutput 定义见视频结构化系统Worker设计
}
rpc GetResult(GetResultRequest) returns (GetResultResponse) {
  option (google.api.http) = {
    post: "/get_result"
    body: "*"
  };
}
```