Parallel Video Analysis System for DIVA

Wenhe Liu
Carnegie Mellon University
wenhel@andrew.cmu.edu

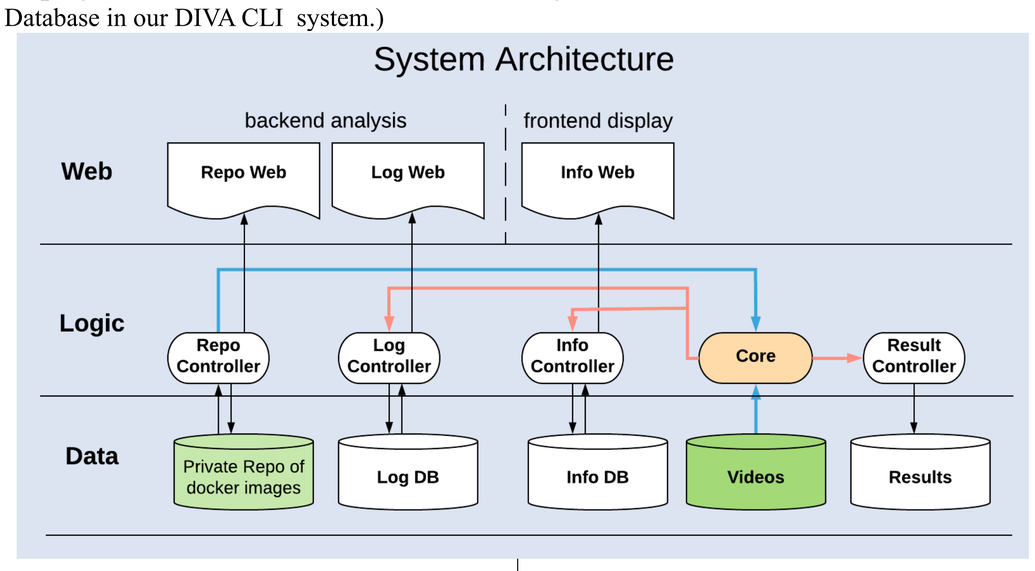
Abstract

Event detection is a widely known and popular task in video analysis. Multiple individual modules are developed for different purposes and experimental stages in event detection. However, different users may implement and build different research codes in different software and hardware conditions. Therefore, it is necessary to develop a common unified system that is capable of delivering results while removing the burden of differences modules. In this project, we develop a docker-based unified system to abstract away unnecessary overheads in terms of a scalable pipeline setup in order to allow the user to quickly and easily apply video analysis pipelines.

Our system supports parallel video analysis with multiple different modules. Moreover, in order to meet the requirements of NIST CLI system, we merged our system with the NIST CLI wrapper.

Architecture

We propose a three-tier architecture for our system:
Data tier: We manage the input videos and results, log and information of pipelines, repository that saves all the docker images used in the pipeline.
Logic tier: There is one core which manages the parallel video analysis in the system. It controls multiple controllers which manage the data and web UI.
Web tier: Status of docker images, logs and information of pipeline are displayed for backend and frontend analysis. (We omit the web level and Log/info Database in our DIVA CLI system.)



Support for DIVA CLI

The DIVA project provides an evaluation CLI with command API designed for NIST to test the submitted system. In the last week, we merged the pipeline system to the evaluation CLI framework and passed system validation test on the NIST smoking test machine.

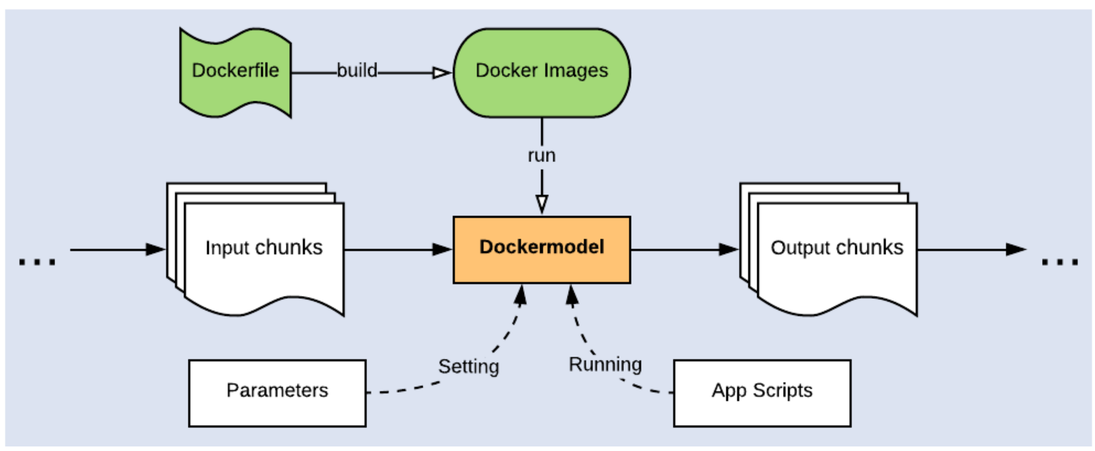
Commands	Description
system-setup	Download and load docker images
design-chunks	Given a file json and activity json, produce a chunk json that is suitable for the system.
experiment-init	Specifies the system config file.
pre-process-chunk	Specifies the ChunkID.
process-chunk	Parallel run our pipeline for the ChunkID.
post-process-chunk	Fusion within ChunkID.
reset-chunk	Delete all cached information for ChunkID so that the chunk can be re-run
merge-chunks	Return NIST format josn output for the listed chunk.
experiment-cleanup	Clean docker models and delete cache files.
exec	A default wrapper script that calls our system pipeline to run the whole dataset

The Scalable Pipeline

Our system supports scalable pipelines. All the modules in the pipeline are packed in docker images and used as dockermodel in our system. For each dockermodel, the model owner only needs to provide the docker images, parameters and scripts to run the model, and our system will manage them in the experiments.

For each Dockermodel, we provide two docs for setting and running the module:

- The **parameters** file (*.yaml*) is used for setting the attributes of Dockermodel.
- The **scripts** file (*.py*) is used for setting the procedure to run the Dockermodel. It calls the operation functions and runs the Dockermodels.



In the pipeline, every module is made as an instance of Dockermodel (DM). The Dockermodel contains attributes and operations for running a docker image. The attributes and operators of dockermodels are as follows:

Attributes of the Dockermodel

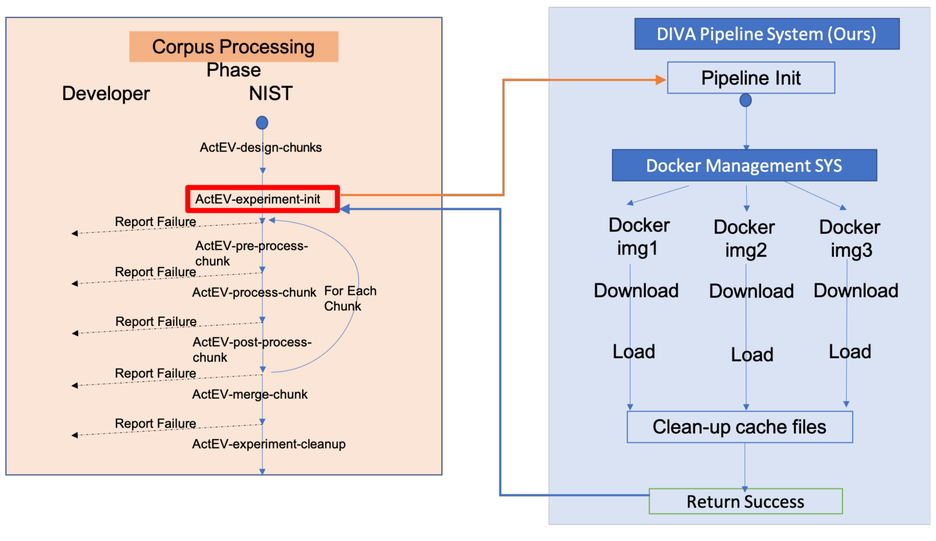
Name	Description
name	Name of the Dockermodel
option	Pre-options of Dockermodel
parameters	Parameters of the Dockermodel
volume	Host volume mapping to the Dockermodel
type	GPU/CPU model
logger	Logger to record the status of the Dockermodel

Operators of the Dockermodel

Name	Description
exist()	Checking existence of a Dockermodel
pull()	Pulling an Dockermodel from registry
run()	Running the Dockermodel
status()	Checking status of the Dockermodel
stop()	Stopping the Dockermodel

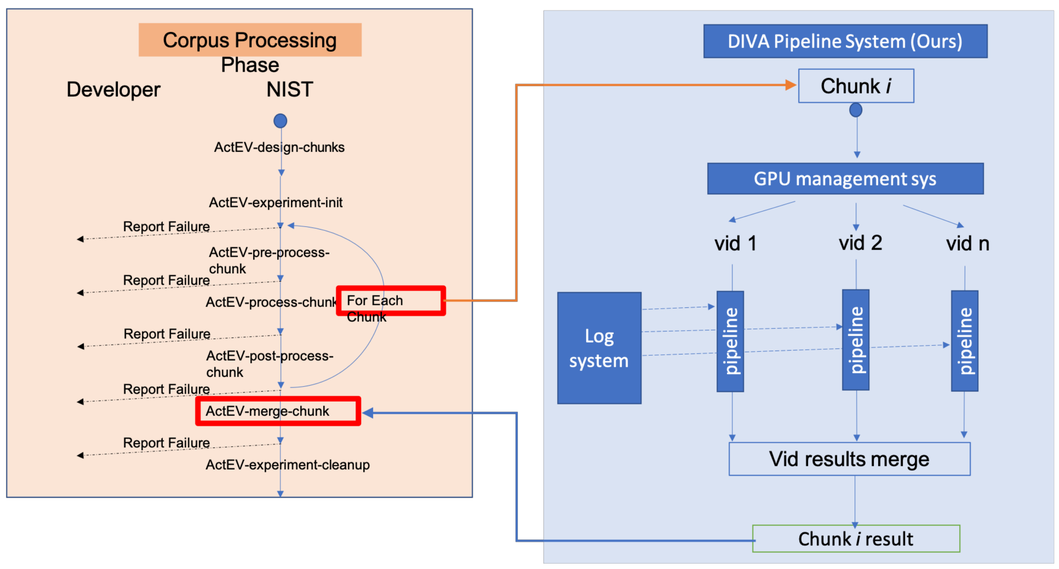
Efficient System Initialization

The NIST CLI requires automatically setup of system. In initialization, our system will download and set up the docker images and setup the system automatically. Such system is easy to re-setup on multiple host machines. We show the initialization of the system as follows.



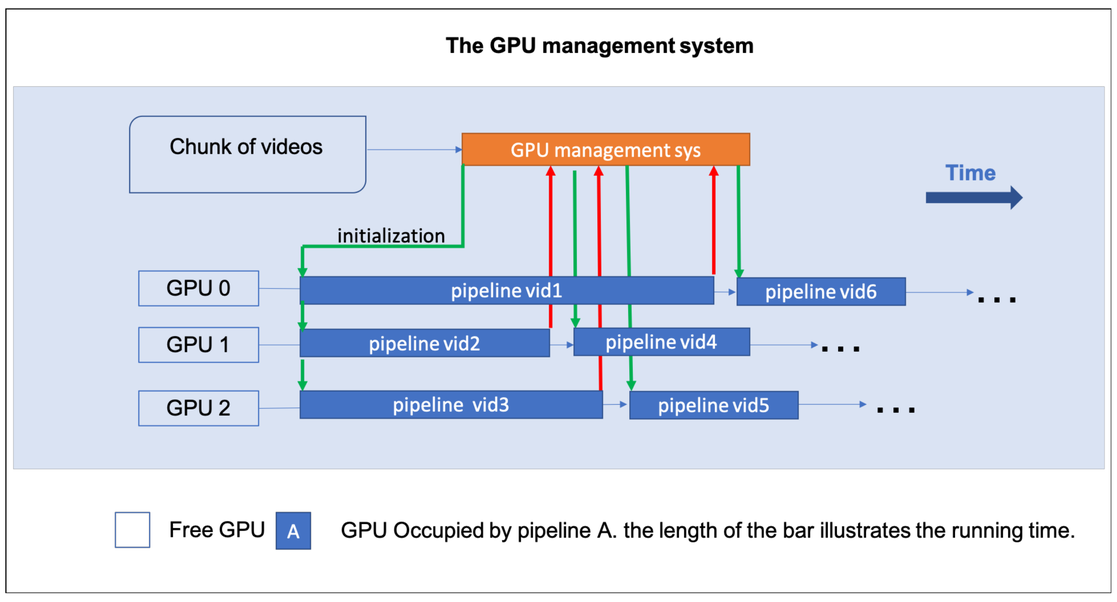
Parallel Video Analysis

The dataset is chunks of videos. For each chunk, the NIST CLI calls our system to run the experiments. Our system supports parallel video analysis in the chunk. We show the parallel video analysis in the system as follows.



Dynamic GPU Management

Our system is a GPU-wise parallel computation system. In the experiments, we find that it is hard to predict and allocate the resource before we analysis the videos. For example, a short but dense video (i.e., a video with many proposals of events in a short time) may cost more than a long but sparse video. Therefore, we develop a GPU management subsystem to dyadically allocate GPU for pipelines. In this system, the GPU management system will monitor the GPU usage and dyadically create a new pipeline when an old one is finished. We show the dynamic GPU management as follows.



Acknowledgement

This work is supported in part through the financial assistance award 60NANB17D156 from U.S. Department of Commerce, National Institute of Standards and Technology and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00340. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation herein. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government.