VisFlow: A Declarative Platform for Parallelizing Large-Scale Vision Programs

Yao Lu[1,2], Aakanksha Chowdhery[1,3], Srikanth Kandula[1]

Microsoft Research[1], University of Washington[2], Princeton University[3]

luyao@cs.washington.edu[1]

Motivation

Previous big-data platforms for vision programs, such as MapReduce or Spark, are suboptimal, due to the following issues:

- **Performance**
  - Manual system configuration (parallelism, scheduling, data storage, etc.) is tedious and time-consuming.
  - Optimizing vision programs is ad-hoc and cannot be automated.
- **Ease-of-use**
  - Special programming expertise is required (languages, APIs, etc.).
  - Popular libraries such as OpenCV and Caffe are not naturally deployed on the cluster.

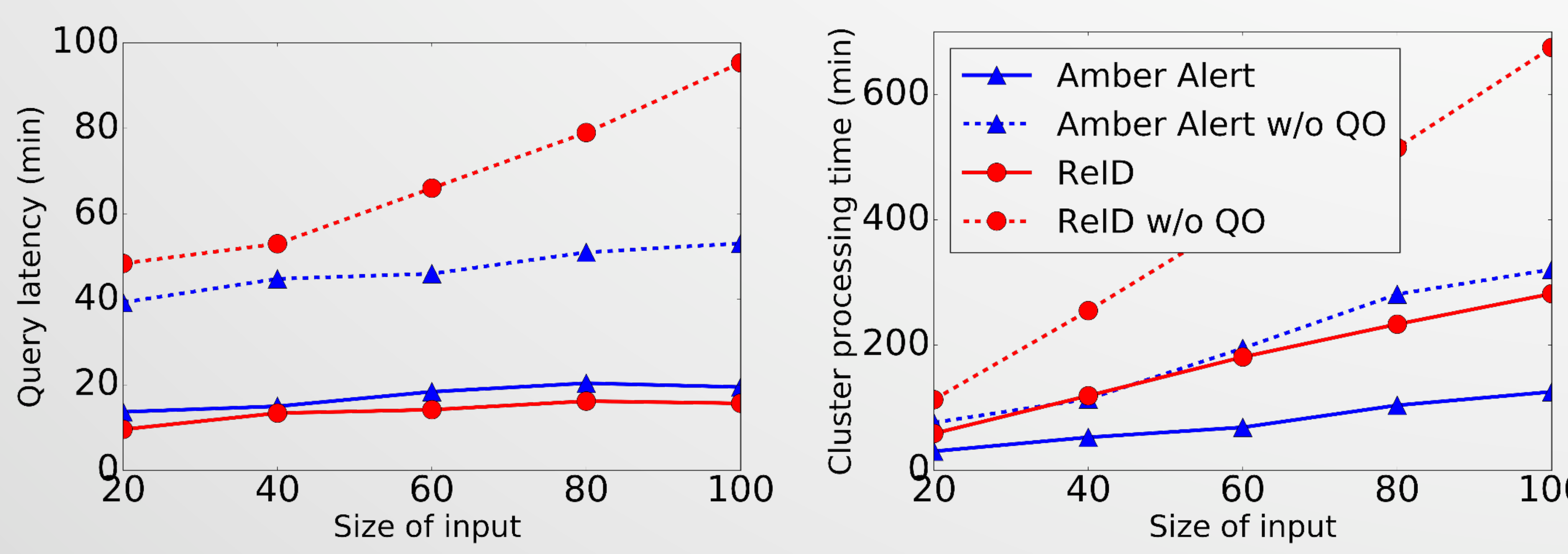
System Design

The key attributes of VisFlow are listed below.

- **A declarative dataflow engine.**
  The vision programs are declared as independent algorithmic modules. Data goes through the modules like a flow.
- **SQL-like programming interface.**
  An extended SQL interface is leveraged that is familiar to both industrial and research communities.
- **Employment of a powerful query optimizer.**
  We apply a cost-based query optimizer that (1) chooses an execute plan with the least cost based on pre-defined transformation rules, (2) parallelizes the vision modules according to input size, and (3) de-duplicate the vision modules shared by different user queries.
- **Data storage.**
  VisFlow utilizes a distributed file system to store large-scale off-line datasets. Online data can be imported using a streaming extractor.
- **Fault tolerance**
  is automatically handled in VisFlow. Usually the usability of the system is above 99.9%.
- **Code base.**
  We implemented 5K lines in C++ and OpenCV for various vision modules, 700 lines in C# for SCOPE wrappers. Each user query has a few tens of lines.
- **Public access.**
  Our system is built upon the SCOPE [2] dataflow engine on Microsoft's Cosmos clusters. A public version of Cosmos is the Azure Data Lake (ADL) [3].

For more details please refer to our technical report [1].

Benchmarks



Data: 100Gb of traffic surveillance videos.

Tasks:
(1) **Amber alert** query retrieves a vehicle with certain color (red, etc.), type (SUV, etc.), and license plate number,
(2) **Re-ID** query matches and tracks one category of vehicles across two cameras.

Results are shown above. We report two different metrics. VisFlow achieves roughly 3x speedup for different input sizes, and near constant user time.

An Extended SQL Interface

Example query to retrieve a vehicle:

```
USING VisFlow;

$rawdata ← EXTRACT camId :int, frameId :int, frame :binary
FROM @"/videos/*.avi" USING VideoExtractor();

$bs ← REDUCE $rawdata USING BackgroundSubtractionReducer(0.05)
PRODUCE camId, frameId, frameBlob;

$lp ← PROCESS $bs USING LicensePlateRecognitionProcessor()
PRODUCE camId, frameId, LP;

$hogfeat ← PROCESS $bs USING HOGFeatureProcessor()
PRODUCE camId, frameId, vFeat;

$VehType ← PROCESS $hogfeat USING LinearSVMProcessor("veh-type.model")
PRODUCE camID, frameID, label, conf;

SELECT camId, frameId FROM $lp, $VehType
ON $lp.{camId,frameId}=$VehType.{camId,frameId}
WHERE $lp.LP = "ABC1234" AND $VehType.label = "SUV";
```

Remark:

- **Extractors** ingest data (e.g., image, video, text) from outside of the system.
- **Processors** are row manipulators, e.g., feature extraction and classification.
- **Reducers** are operations over groups of rows, e.g., background subtraction.
- **Combiners** join two or more row sets, e.g., object/keypoint matching.

The above language interface generates the following user roles:

- **Vision engineers** are responsible for the individual vision modules written in popular libraries, e.g, OpenCV and Caffe.
- **System engineers** focus on robust and efficient infrastructures, which save the other roles from tedious performance adjustment.
- **End users** such as application engineers and data scientists, only need to take care of the logic described in the end query.

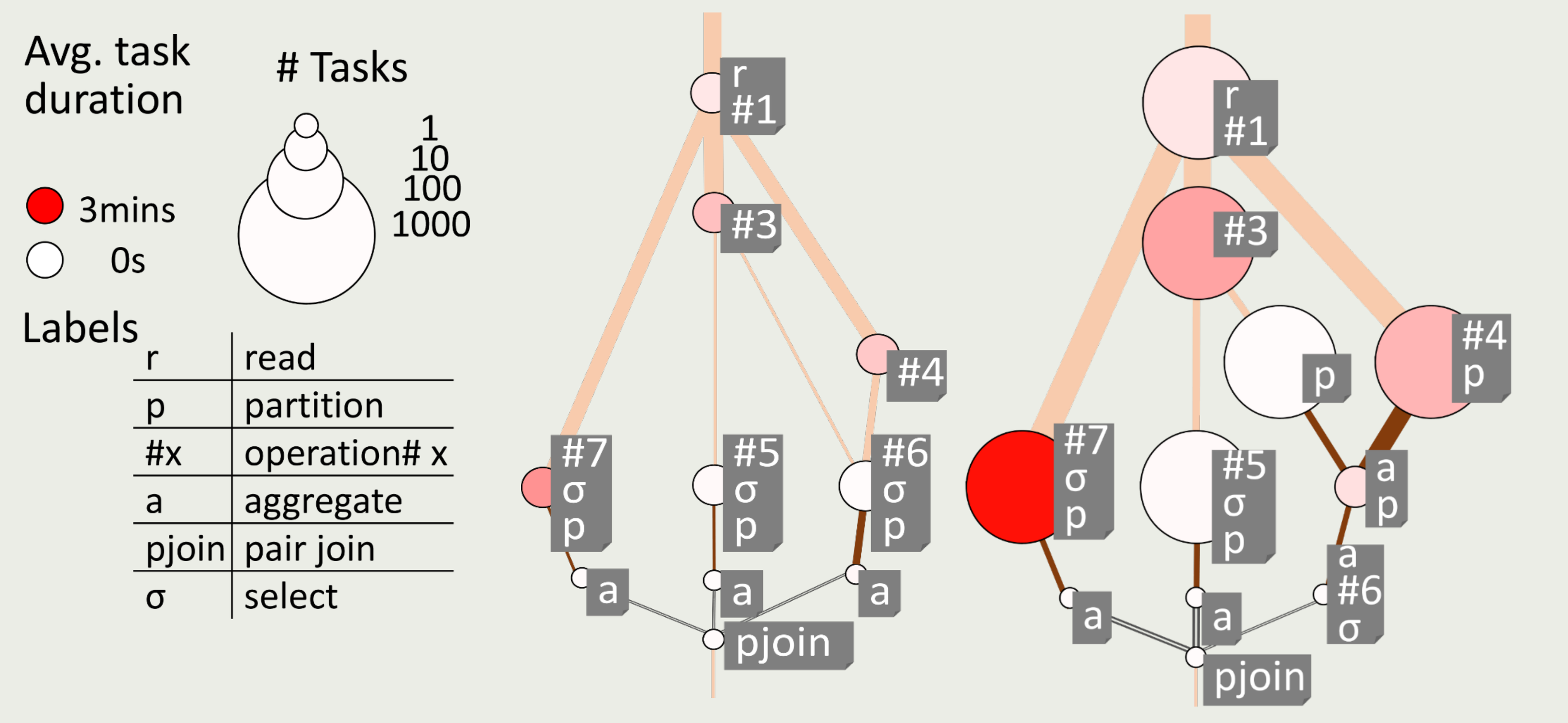
Query Optimizer for Vision Programs

We apply a cost-based query optimizer to accelerate the query execution.

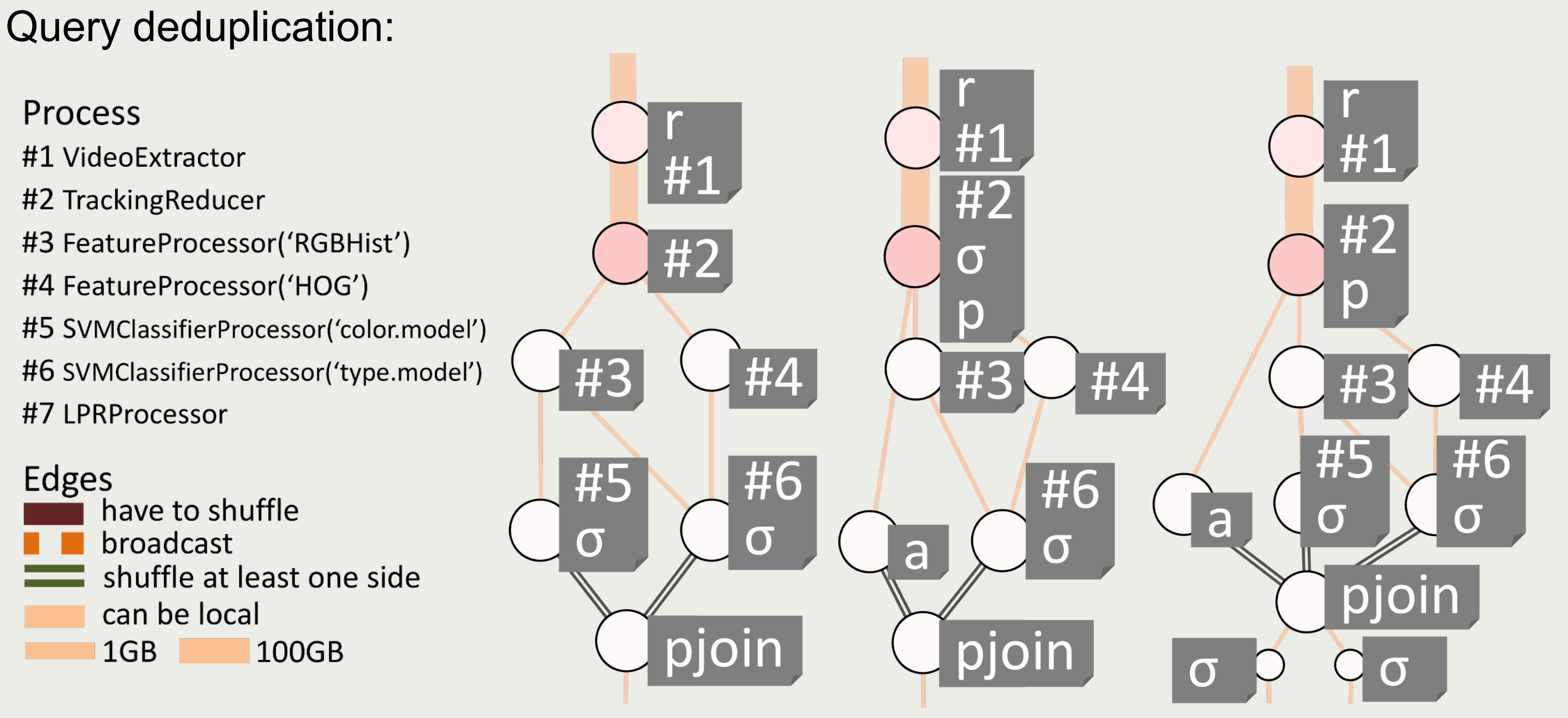
- A variety of transformation rules are applied to replace the query sub-expression. E.g., Predicate push-down:

$$\epsilon_1 \rightarrow S \rightarrow Filter \rightarrow \epsilon_2 \Rightarrow \epsilon_1 \rightarrow Filter \rightarrow S \rightarrow \epsilon_2$$

- Auto parallelization, 1 Gb vs. 100 Gb:



- Query deduplication:



References

[1] Y. Lu, A. Chowdhery, S. Kandula. VisFlow: A Relational Platform for Efficient Large-Scale Video Analytics. MSR Report 2016-28.
[2] R. Chaiken et al. SCOPE: Easy and Efficient Parallel Processing of Massive Datasets. In VLDB, 08'
[3] Azure Data Lake. http://bit.ly/1Miq8RP