Motivation
- Utilize Storm to automate allocation of nodes when distributing workflow applications across a computer cluster
- Minimize modification of the original workflow application to simplify parallelization
- Leverage Storm’s inherent scalability to allow the workflow application to scale automatically with the underlying cluster

Distributed Processing of Workflow Applications Using the Storm Framework

Background
Workflow Application: Lavatube
Lavatube is a visual programming framework used for computer vision research:
- Provides a graphical interface through which users piece together complex video and image processing workflows
- Provides a large library of editing functions which can be combined to perform complex operations, such as anomalous behavior detection systems
- Maximizes its use of multi-core systems with a design optimized for parallel processing

Storm
Storm is a distributed computation framework that is based on a streaming data model:
- Spreads complex topologies across a computer cluster, making it possible to run the topologies on torrents of data that would drown a single computer
- Uses efficient message passing to allow it to push as much as 1,000,000 messages per second per node
- Does this in a robust manner, including reordering requiring a non-blocking approach to achieve 100% identical processing

Motivation
- Users create workflows using HTML5 based GUI
- XML doc is generated which is passed to next layer
- Maximizes its use of multi-core systems with a design optimized for parallel processing

Storm
- Initializes the cluster, assigning sections of the workflow to various nodes
- Manages the messages passed between nodes
- Each runs its own instance of Lavatube locally, initialized using the sub-workflow XML doc received during Storm initialization
- Image frames arrive via Storm, are processed by the local sub-workflow, and then are sent out to consuming sub-workflows via Storm

Two Dimensions of Parallel Processing

1) Task Parallelism is inherent to Storm’s distribution of sub-workflows across the cluster
2) Data Parallelism involves duplicating sub-workflows and processing subsets of the video stream on the separate duplicated nodes

Frames are distributed to the duplicated nodes using a modular division based batching method
- Resolves issues with synchronizing the frames of merging video streams that arrive as input to parallelized components
- Maintains a partial ordering in the frames, reducing the burden on components which later need the frames to be in order

Research Results
Successfully validated this model using actual video data.
- Tested on grid of five computers using 16 Storm worker processes
- Input two video files, performed multiple transformations, including parallelized merge of video streams
- Output AVI file 100% identical to non-distributed processing

Tested performance of Storm infrastructure when supporting larger volumes of video data:
- Ran input in infinite loops to observe performance bottlenecks
- Primary bottleneck found to be computationally expensive Lavatube functions, such as the geometric resize function
- Indicates capability of the combined framework to increase the processing capacity of Lavatube for such functions by using data parallelism to spread computation across multiple nodes