OSDC OPEN SCIENCE DATA CLOUD

Name:	Joseph Korpela
University:	UCLA
Adviser:	Isao Kojima
Organization:	AIST

Motivation

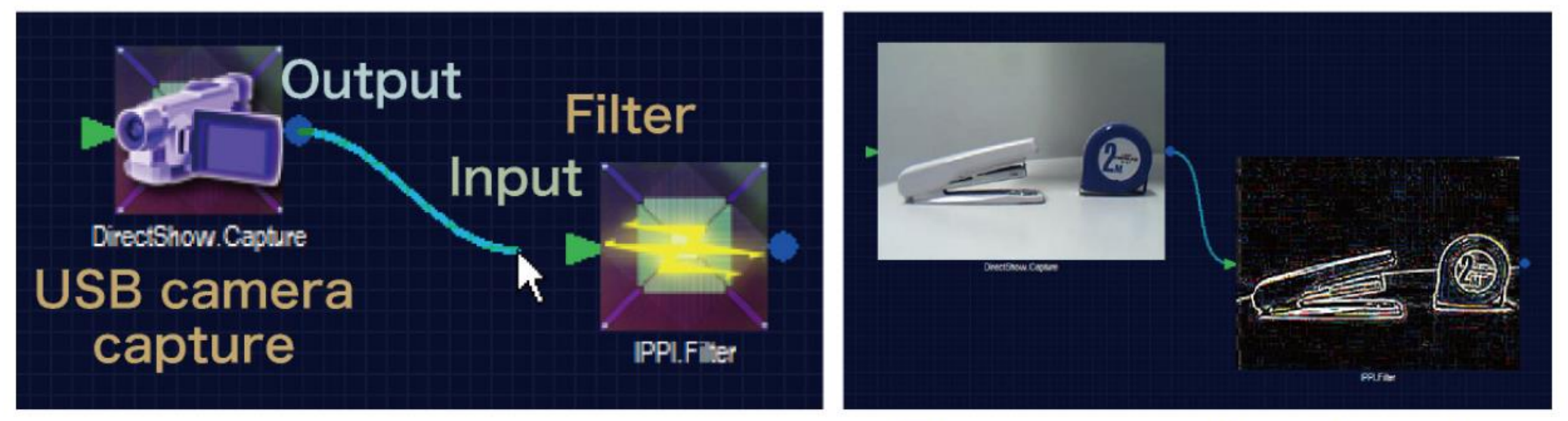
- Utilize Storm to automate allocation of nodes when distributing workflow applications across a computer cluster
- Minimize modification of the original workflow application to simplify parallelization
- Leverage Storm's inherent scalability to allow the workflow application to scale automatically with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

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

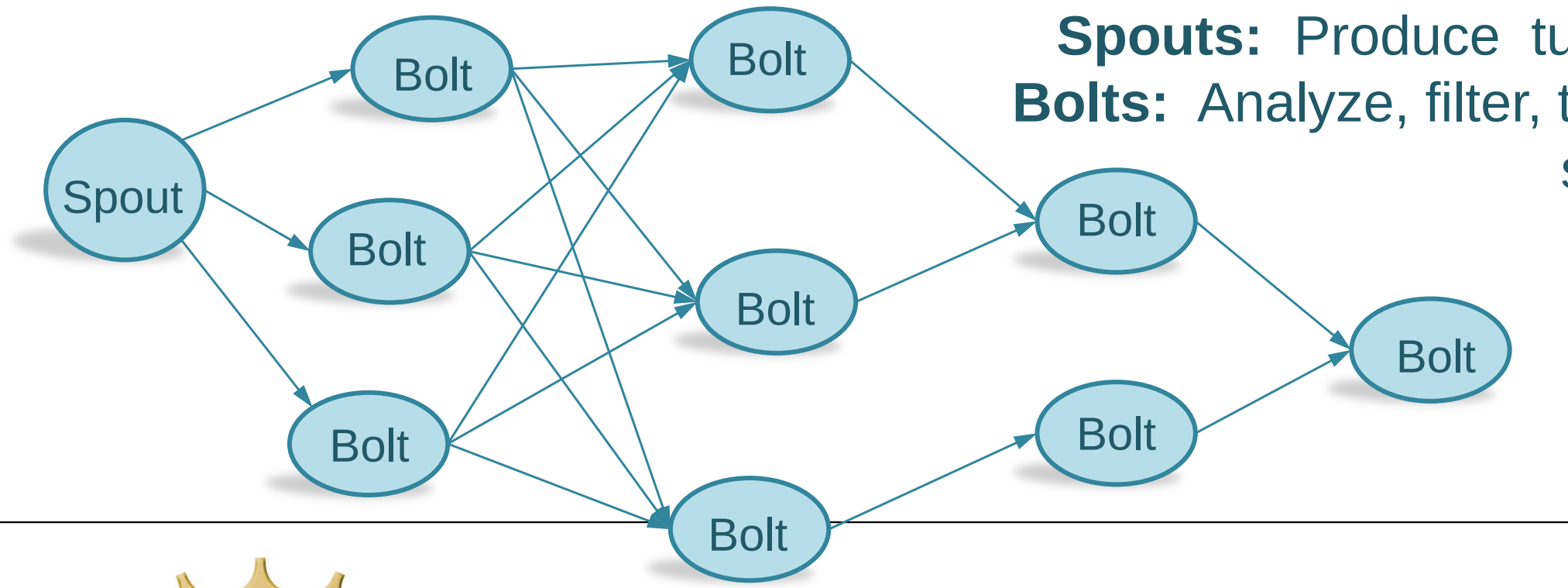


Connecting by drag & drop → The processing flow is easily constructed.

Storm

Storm is a distributed computation framework that is based on a streaming data model:

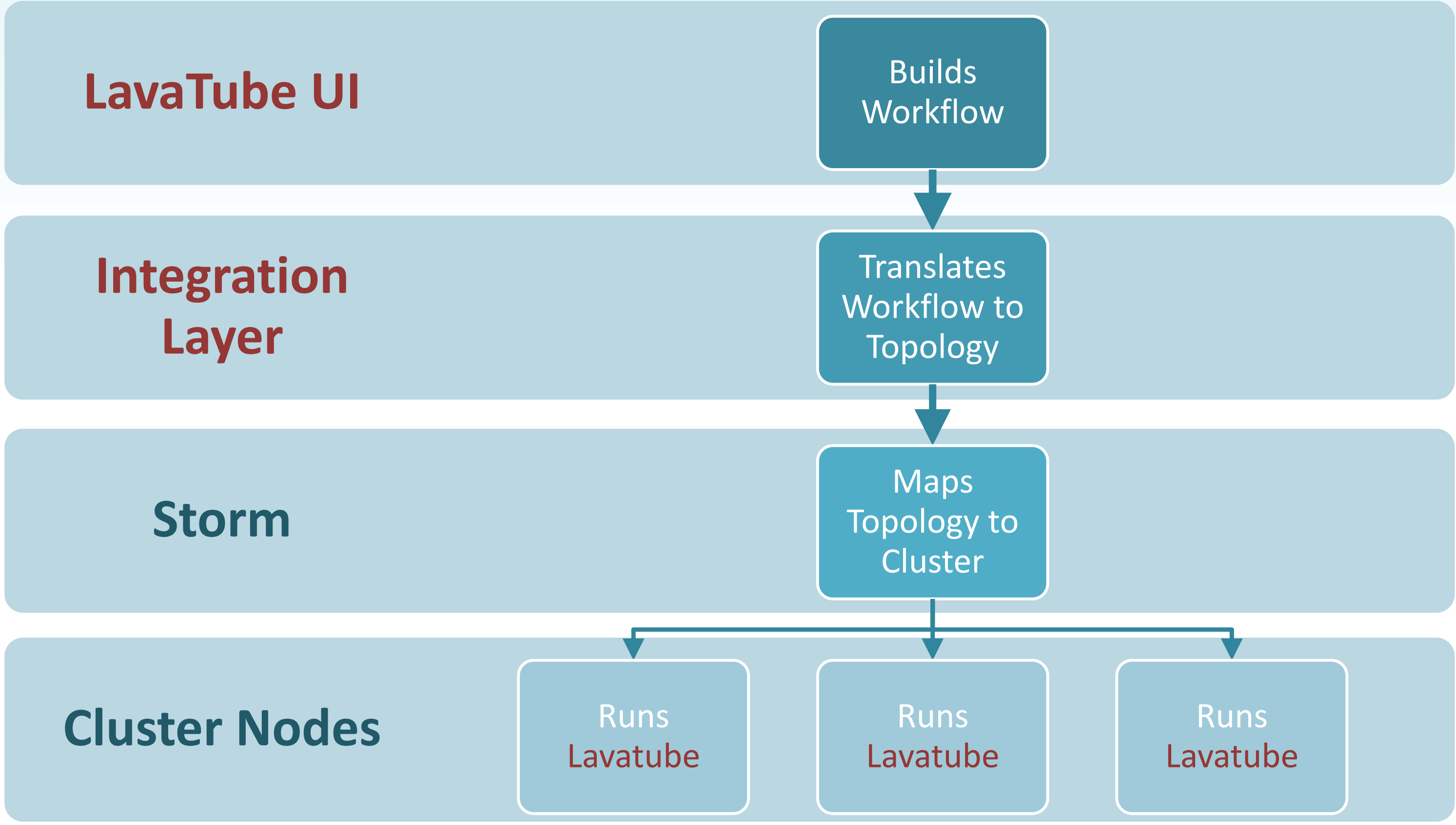
- Spreads complex topologies across a computer cluster, making it possible to run the topologies on torrents of data that would drown a single computer
- Uses efficient message passing to allow it to push as much as 1,000,000 messages per second per node
- Does this in a robust manner, with the ability to automatically detect and recover from failed nodes, assign new nodes, and redistribute tasks dynamically



Storm Topologies:
Spouts: Produce tuples of data
Bolts: Analyze, filter, transform, and store that data

Distributed Processing of Workflow Applications Using the Storm Framework

Mapping the Application's Workflow onto Storm



Lavatube UI:

- Users create workflows using HTML5 based GUI
- XML doc is generated which is passed to next layer

Integration layer:

- Parses the XML and separates it into multiple sub-workflows
- Analyzes the workflow's structure and uses it to build a Storm topology

Storm:

- Initializes the cluster, assigning sections of the workflow to various nodes
- Manages the messages passed between nodes

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

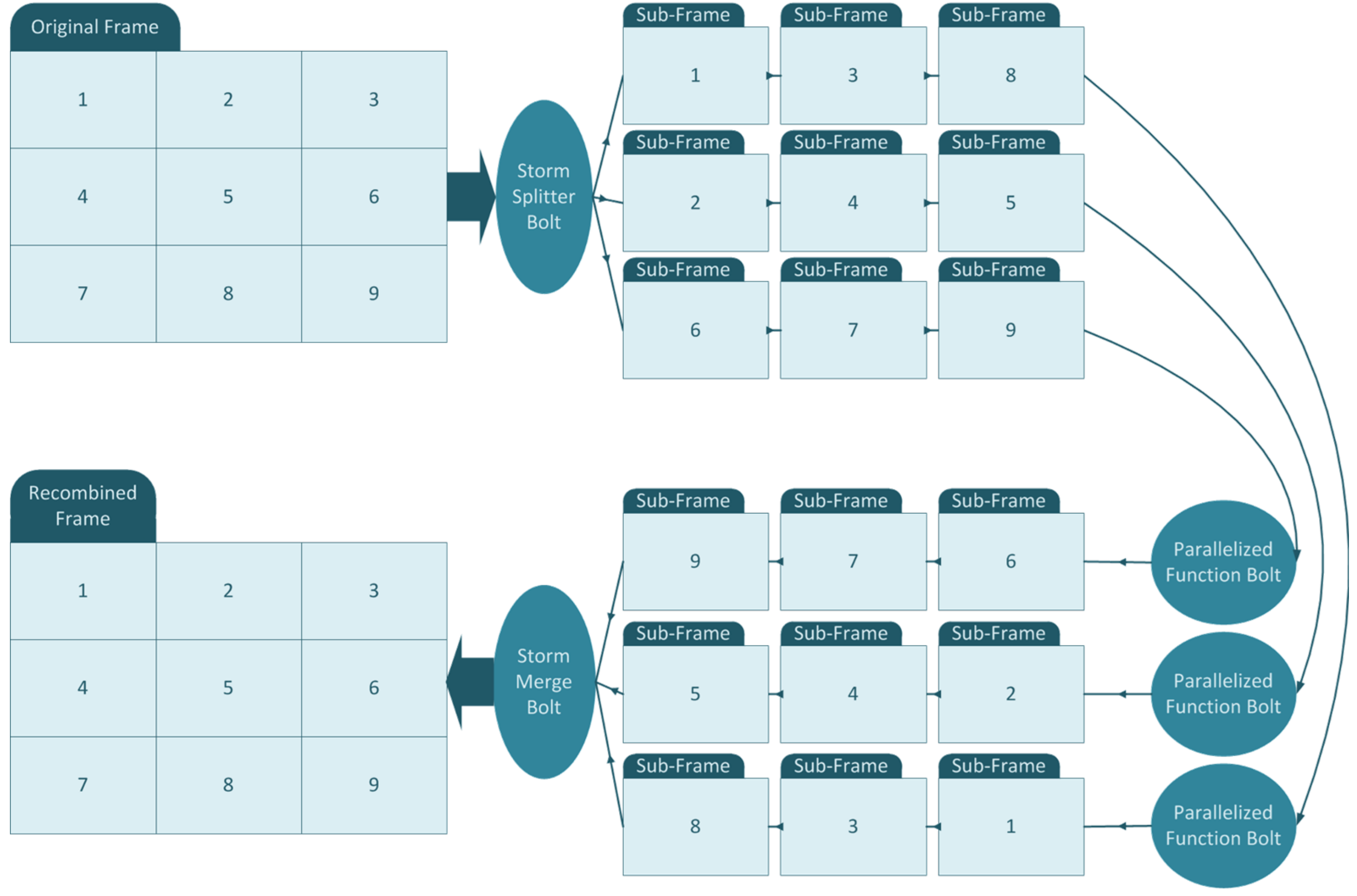
Two Dimensions of Parallel Processing

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

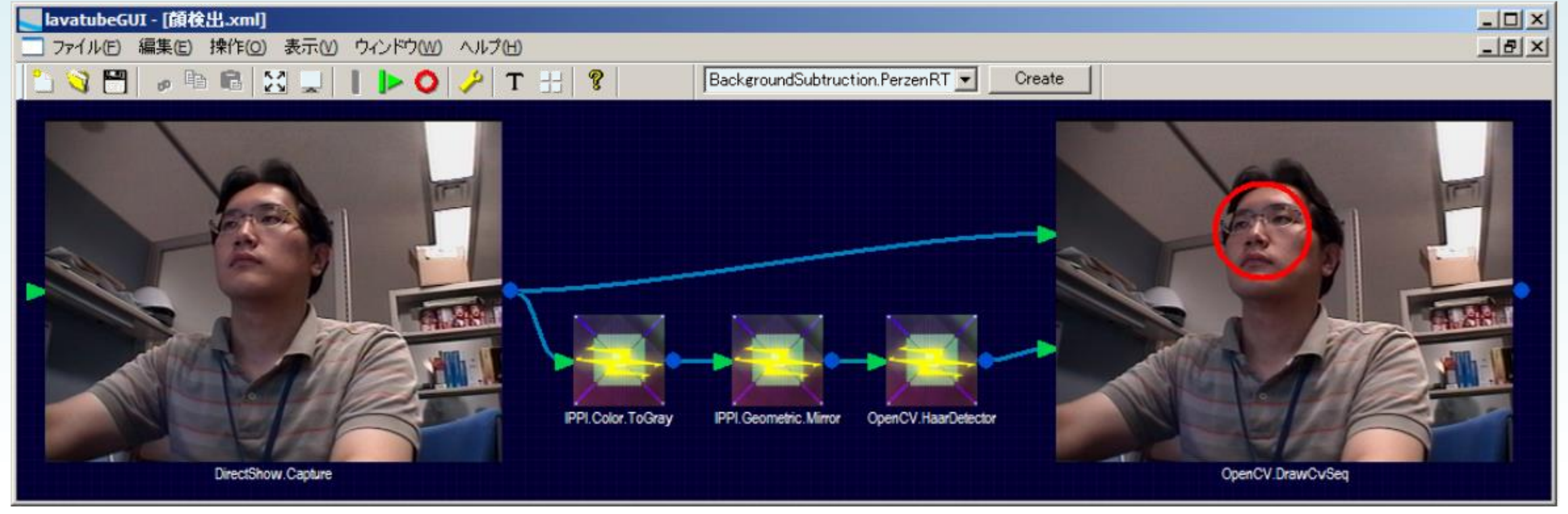
**Further data parallelism can also be achieved by splitting individual video frames into macro blocks, allowing large frames, such as with 4K Ultra HD, to be processed in parallel*



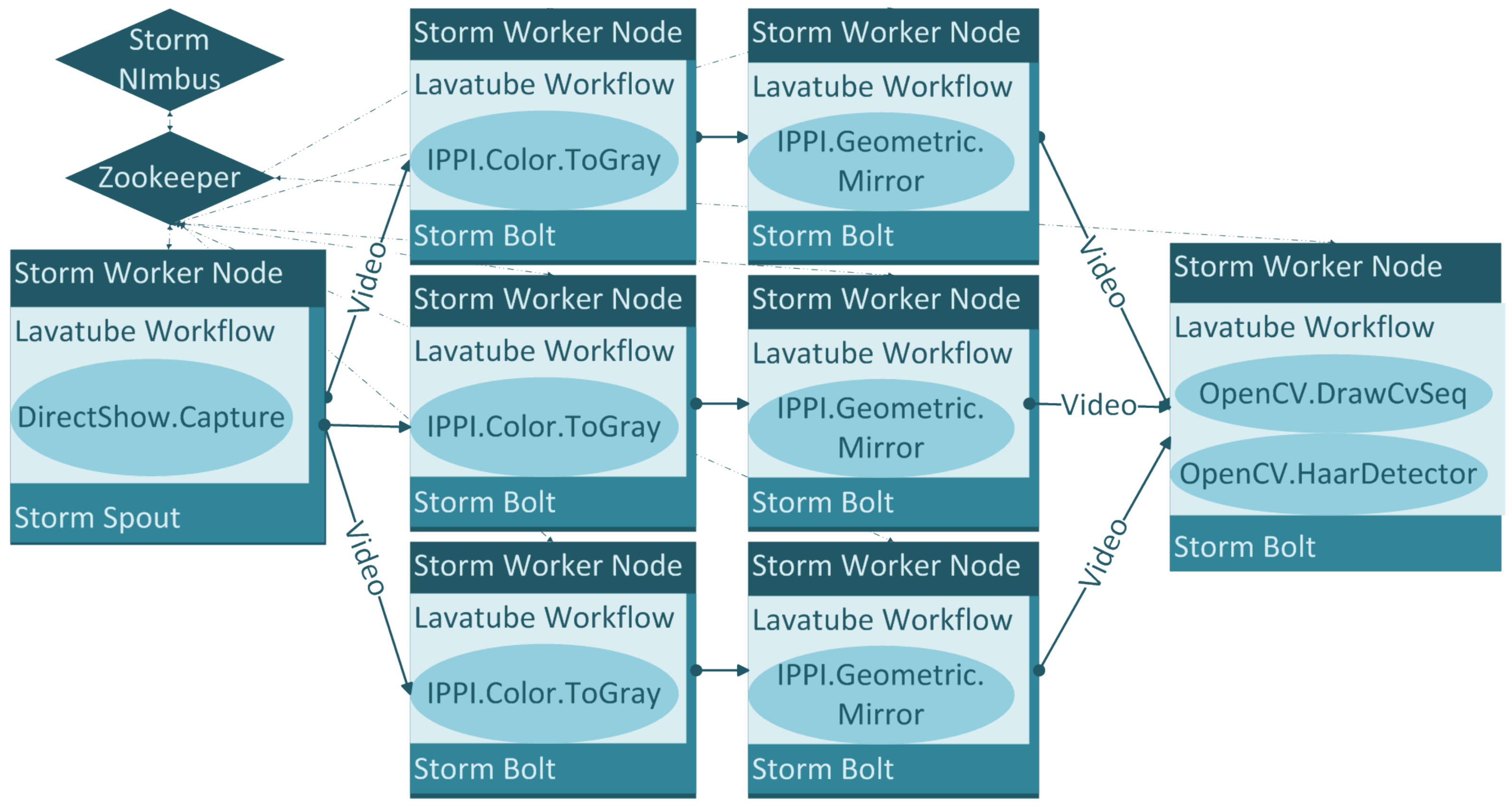
Macro Block based Data Parallelization *(even splitting and merging can be accomplished in parallel, with only frame reordering requiring a non-parallelized component)*

Example Mapping From Lavatube to Storm

Original Workflow (Lavatube Face detection algorithm):



Mapping of the above Lavatube face detection workflow to a Storm cluster:



The above example demonstrates both data and task parallelism:

- Data Parallelism: IPPI.Color.ToGray and IPPI.Geometric.Mirror functions are each duplicated on three nodes each, with each duplicate processing a subset of the original video stream
- Task Parallelism: Separate worker nodes are used to process sequential functions, such as with DirectShow.Capture being processed on a separate worker node from IPPI.Color.ToGray

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

NSF
OSDC OPEN SCIENCE DATA CLOUD
PIRE
THE UNIVERSITY OF CHICAGO
FIU FLORIDA INTERNATIONAL UNIVERSITY
CIARA Center for Internet Augmented Research & Assessment research • collaboration • scholarship
Ucla
cisa
COMMIT/
sicsa* The Scottish Informatics & Computer Science Alliance
AMPATH
OPEN CLOUD CONSORTIUM
UNIVERSITY OF AMSTERDAM
USP
edinburgh data-intensive research