Apache Apex (incubating)

Stream processing for Big Data

David Yan
DataTorrent, Inc.
HPTS 2015, 9/30/2015

Project History & Status

- Project development started in 2012 at DataTorrent
- Open-sourced in July 2015
- Apache Apex started incubation in August 2015
- 50+ committers from Apple, GE, Capital One, DirecTV, Silver Spring Networks, Barclays, DataTorrent

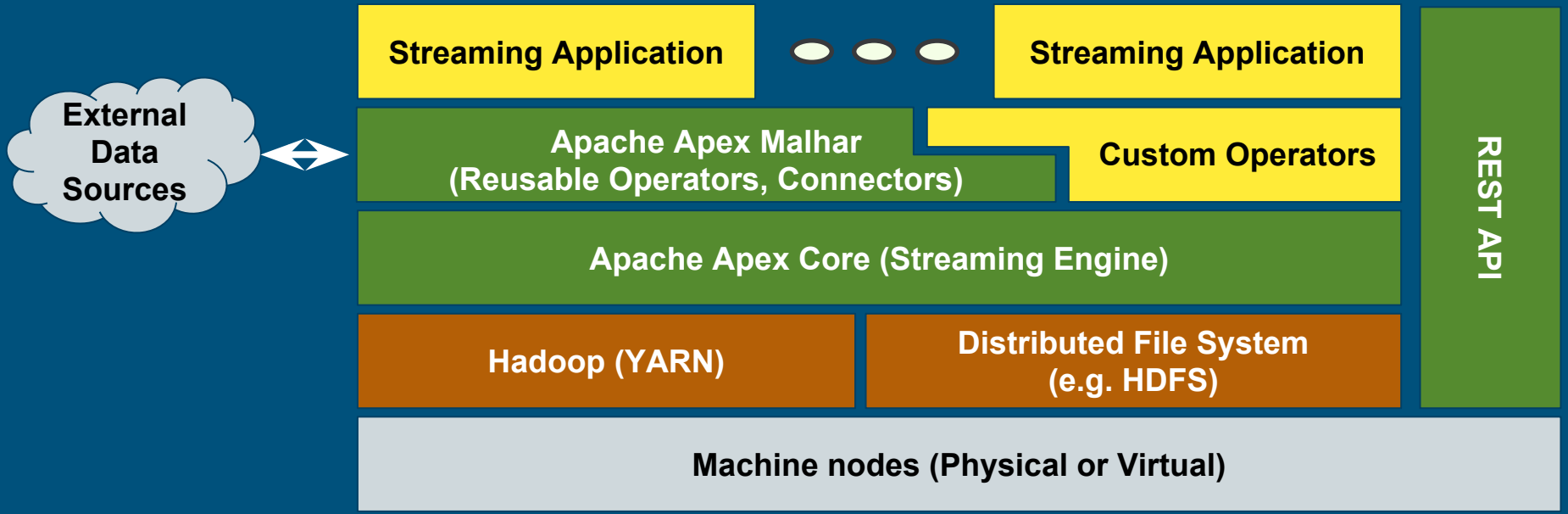
Use Cases

- Advertising
- Finance
- Telecoms and Networks
- Security
- Ingestion
- Many others

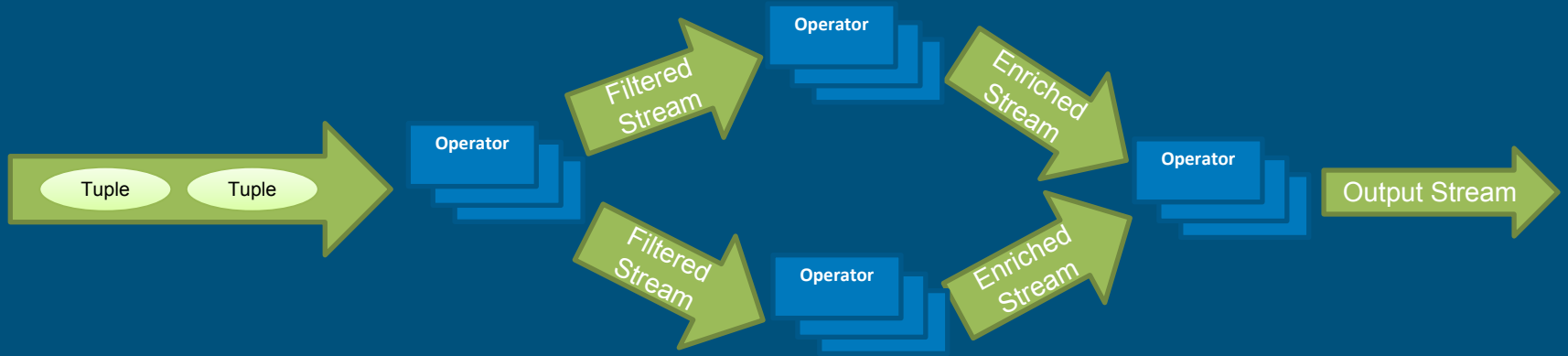
Guiding Principles

- Highly scalable and performant
- Fault tolerant
- Hadoop native
- Easily integrated
- Easily operable
- Easily developed

Architecture



Application Programming Model



- Data-in-motion architecture
- Directed Acyclic Graph (DAG) is made up of *Operators* and *Streams*
- A *Stream* is a sequence of data tuples and control tuples
- An *Operator* takes one or more input *Streams*, performs computation and emits one or more output *Streams*.

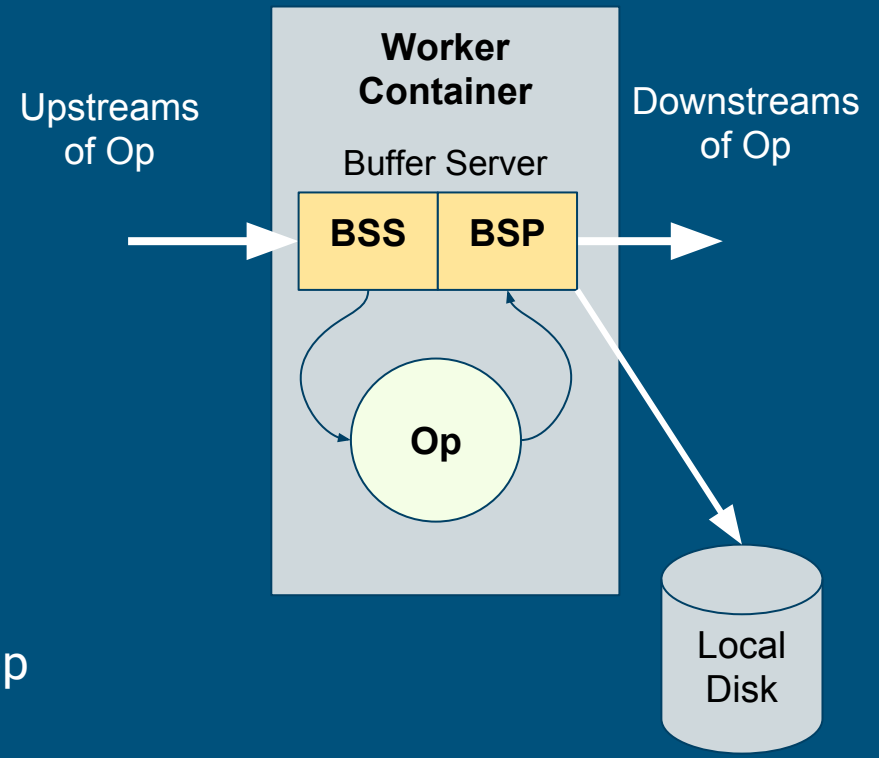
Demo Time!

Apex App Master

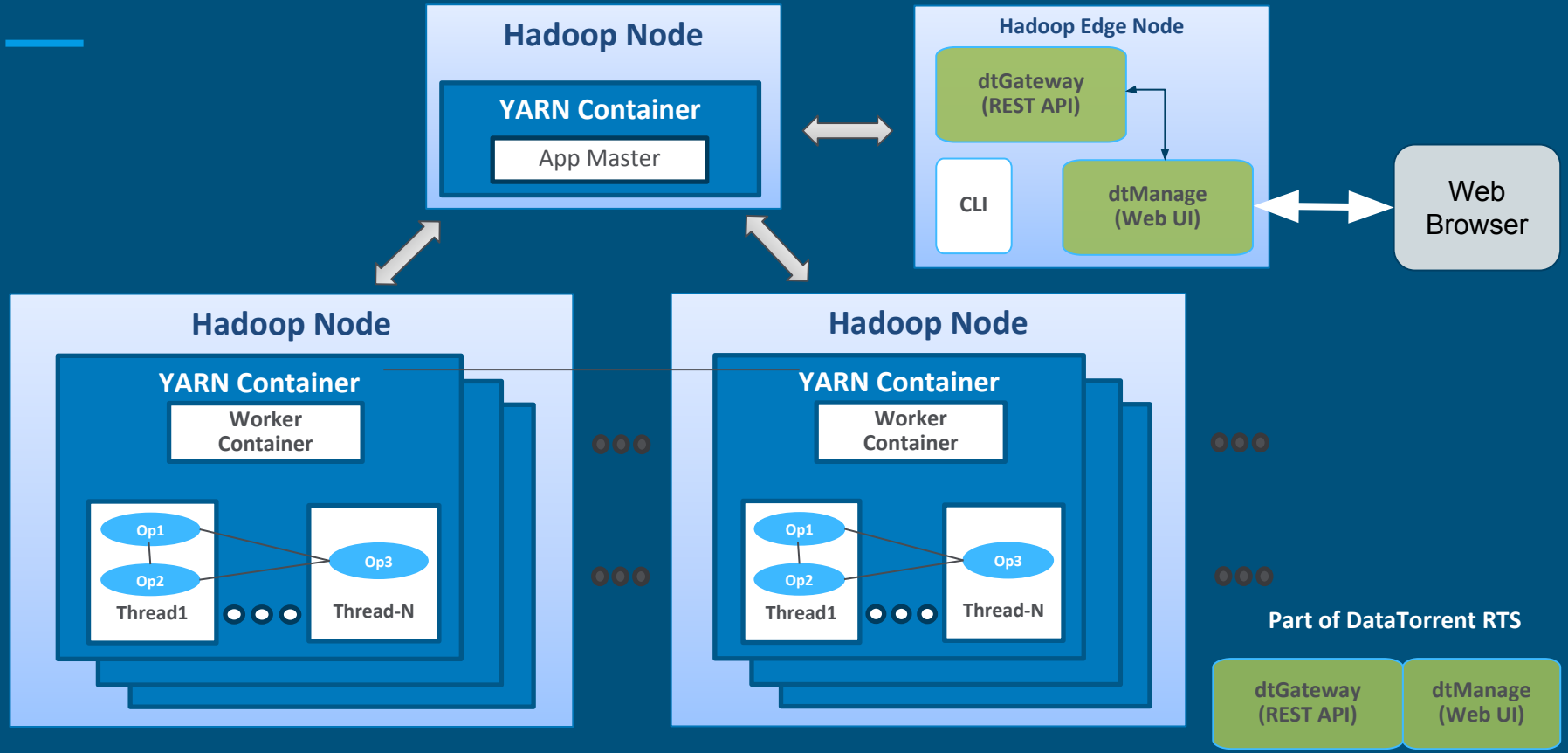
- Native YARN application
- Key functions
  - Provisions and monitors Apex Worker Containers for operators
  - Partitions and merges operators for auto-scaling
  - Detects failure and restarts failed operators
  - Updates DAG topology
- HA
  - Checkpoints its state in HDFS
  - Leverages YARN’s monitoring and restarting feature

Apex Worker Container

- Resides in a YARN container
- Runs operator instances
- Includes Buffer Server
- Manages bookkeeping and checkpointing
- Executes commands from Apex App Master
  - Starts new operator instances
  - Purges old data
- Sends heartbeats with stats to Apex App Master



Apex Component Overview



Windowing

- Tuples are divided into time slices called streaming windows
- Input operators insert control tuples to mark the window boundary
- Checkpointing and management are done at window boundary
- Users can define application window size that is a multiple of the streaming window size.
- Sliding and tumbling application window are supported natively

Checkpointing

- Saving operators' state for recovery
- Decentralized
- Input operators send *checkpoint* control tuples to all downstream operators
- During a checkpoint, operator serialized state is asynchronously written to HDFS
- If all operators have checkpointed a particular window, that window is "committed" and all previous checkpoints will be purged
- The state of Apex App Master is also checkpointed

Recovery

- Apex App Master detects the failure of an operator
- All downstream operators from the failed operator will be restarted by the Apex App Master.
- Upon restart, operators read from the last committed checkpoint to recover their states.
- Data is replayed upstream from the recovery checkpoint by Buffer Server
- Recovery is automatic and typically takes only a few seconds.

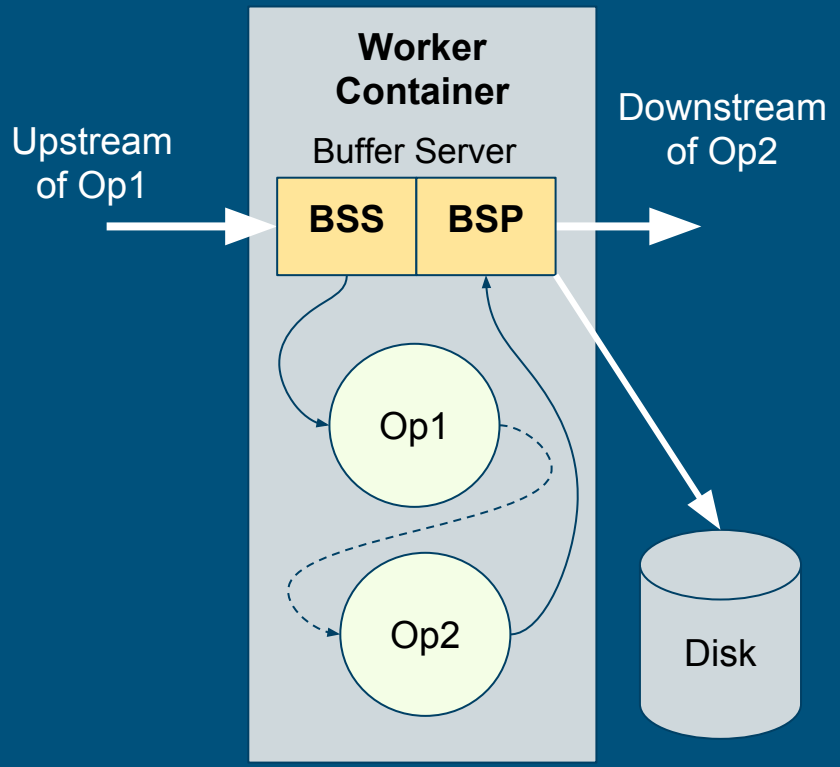
Process Modes

- **AT_LEAST_ONCE** (default): Windows are processed at least once
- **AT_MOST_ONCE**: Windows are processed at most once
  - During recovery, all downstream operators are fast-forwarded to the window of latest checkpoint
- **EXACTLY_ONCE**: Windows are processed exactly once
  - Checkpoint every window
  - Checkpointing becomes blocking

Compute Locality

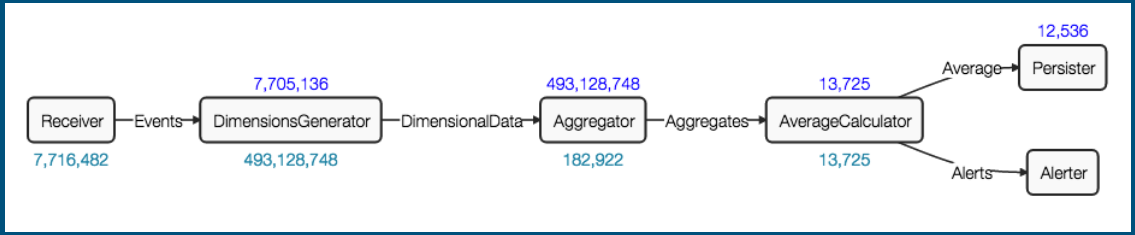
User can specify *compute locality* of each stream

- **NODE_LOCAL**: Same node, separate container
- **CONTAINER_LOCAL**: Same container, separate thread (in-memory queue)
- **THREAD_LOCAL**: Same thread (function call)
- Default is no locality

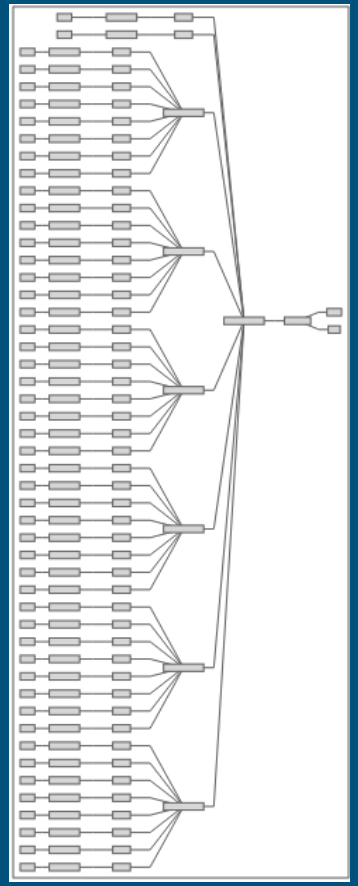


Partitioning & Auto-scaling

- A logical operator can be split into multiple physical operators
- Apex is able to scale up or down dynamically
- Default and custom Unifiers to unify the results from partitions

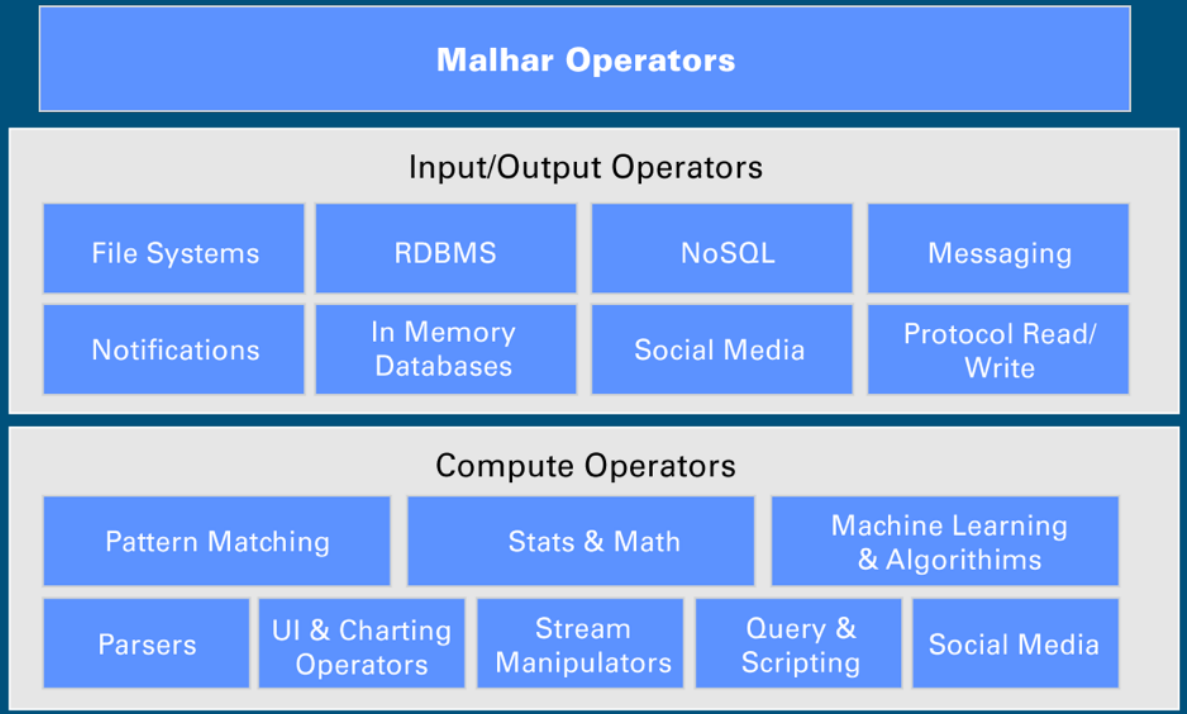


Logical Plan



Physical Plan

Apex Malhar Libraries



For more info

- Mailing List: dev@apex.incubator.apache.org
- Apache Apex: http://apex.apache.org/
- Github
  - Apex Core: http://github.com/apache/incubator-apex-core
  - Apex Malhar: http://github.com/apache/incubator-apex-malhar
- DataTorrent: http://www.datatorrent.com

Thank you!