Next Gen Big Data Analytics with Apache Apex

Apache Big Data, Vancouver
May 9th 2016
Stream Processing

- Data from a variety of sources (IoT, Kafka, files, social media etc.)
- Unbounded stream data
  - Batch can be processed as stream (but a stream is not a batch)
- (In-memory) Processing with temporal boundaries (windows)
- Transformation, Aggregation, Rules, Actions, … -> Analytics
- Results stored to a variety of sinks or destinations
  - Streaming application can also serve data with very low latency
Apache Apex Features

• In-memory stream processing platform
  ◦ Developed since 2012, ASF TLP since 04/2016
• Unobtrusive Java API to express (custom) logic
• Scale out, distributed, parallel
• High throughput & low latency processing
• Windowing (temporal boundary)
• Reliability, fault tolerance, operability
• Hadoop native
• Compute locality, affinity
• Dynamic updates, elasticity
Applications on Apex

• Distributed processing
  • Application logic broken into components called operators that run in a distributed fashion across your cluster

• Natural programming model
  • Code as if you were writing normal Java logic
  • Maintain state in your application variables

• Scalable
  • Operators can be scaled up or down at runtime according to the load and SLA

• Fault tolerant
  • Automatically recover from node outages without having to reprocess from beginning
  • State is preserved
  • Long running applications

• Operational insight
  • See how each operator is performing and even record data
Apex Stack Overview

Graphical Application Design and Launch

Data Visualization Dashboard

Management Console

Malhar: Open Source Operators

Streaming Application

Streaming Runtime
High Performance, fault tolerant, Complex & In-Memory Processing of Data In Motion

Hadoop – YARN + HDFS
Certified With All Major Hadoop Distros (Cloudera, Hortonworks, Pivotal, MapR)

Physical | Virtual | Cloud
## Apache Apex Malhar Library

### MALHAR OPERATORS

#### INPUT/OUTPUT OPERATORS
- File Systems
- RDBMS
- NoSQL
- Messaging
- Notification
- In Memory Databases
- Social Media
- Protocol read/write

#### COMPUTE OPERATORS
- Pattern Matching
- Stats and Math
- Machine Learning and Algorithms
- Parsers
- UI and Charting Operators
- Stream Manipulators
- Query and Scripting
- Social Media
Native Hadoop Integration

- YARN is the resource manager
- HDFS used for storing any persistent state
A Stream is a sequence of data tuples

A typical Operator takes one or more input streams, performs computations & emits one or more output streams
- Each Operator is YOUR custom business logic in java, or built-in operator from our open source library
- Operator has many instances that run in parallel and each instance is single-threaded

Directed Acyclic Graph (DAG) is made up of operators and streams
Streaming Windows

- Application window
- Sliding window and tumbling window
- Checkpoint window
- No artificial latency
Event time & Dimensions Computation

- (All) : 5
  - t=4:00 : 2
  - t=5:00 : 3
  - k=A, t=4:00 : 2
  - k=A, t=5:00 : 1
  - k=B, t=5:00 : 2

- (All) : 4
  - t=4:00 : 2
  - t=5:00 : 2
  - k=A, t=4:00 : 2
  - K=B, t=5:00 : 2

- (All) : 1
  - t=4:00 : 1
  - k=A, t=4:00 : 1
The source for the logical plan can be in different formats. Using the Apex Java API, the WordCount example could look like this:

```java
@ApplicationAnnotation(name="MyFirstApplication")
public class Application implements StreamingApplication {
    @Override
    public void populateDAG(DAG dag, Configuration conf) {
        LineReader lineReader = dag.addOperator("input", new LineReader());
        Parser parser = dag.addOperator("parser", new Parser());
        UniqueCounter counter = dag.addOperator("counter", new UniqueCounter());
        ConsoleOutputOperator cons = dag.addOperator("console", new ConsoleOutputOperator());
        dag.addStream("lines", lineReader.output, parser.input);
        dag.addStream("words", parser.output, counter.data);
        dag.addStream("counts", counter.count, cons.input);
    }
}
```
public class Parser extends BaseOperator {

    public transient final DefaultInputPort<String> input = new DefaultInputPort<String>() {
        @Override
        public void process(String s) {
            String[] words = s.split("[\p{Punct}\s\\"\\‘"]+");
            for (String word : words) {
                output.emit(word);
            }
        }
    }

    public transient final DefaultOutputPort<String> output = new DefaultOutputPort<>();
}
Operators (contd)

```java
public class UniqueCounter<K> extends BaseOperator {
    private Map<K, MutableInt> counts = new HashMap<>();

    public transient final DefaultInputPort<K> input = (tuple) -> {
        MutableInt count = counts.get(tuple);
        if (count == null) {
            count = new MutableInt();
            counts.put(tuple, count);
        }
        count.increment();
    };

    @Override
    public void endWindow() {
        for (Map.Entry<K, MutableInt> entry : counts.entrySet()) {
            output.emit(new KeyValPair<K, Integer>(entry.getKey(), entry.getValue().toInteger()));
        }
    }

    public transient final DefaultOutputPort<KeyValPair<K, Integer>> output = new DefaultOutputPort<>();
}
```
Partitioning

Unifier

Logical Diagram

0 1 2

Physical Diagram with operator 1 with 3 partitions

0 1 1

NxM Partitions

Logical DAG

0 1 2 3

Physical DAG with (1a, 1b, 1c) and (2a, 2b): Bottleneck on intermediate Unifier

0 1b 1c 2b

Physical DAG with (1a, 1b, 1c) and (2a, 2b): No bottleneck

0 1b 1c 2a

Unifier
Advanced Partitioning

Parallel Partition

Logical DAG

Physical DAG

Physical DAG with Parallel Partition

Cascading Unifiers

Logical Plan

Execution Plan, for $N = 4; M = 1$

Execution Plan, for $N = 4; M = 1, K = 2$ with cascading unifiers
Dynamic Partitioning

- Partitioning change while application is running
  - Change number of partitions at runtime based on stats
  - Determine initial number of partitions dynamically
    - Kafka operators scale according to number of kafka partitions
  - Supports re-distribution of state when number of partitions change
  - API for custom scaler or partitioner

"Unifiers not shown"
How tuples are partitioned

• Tuple hashcode and mask used to determine destination partition
  ◦ Mask picks the last n bits of the hashcode of the tuple
  ◦ hashcode method can be overridden

• StreamCodec can be used to specify custom hashcode for tuples
  ◦ Can also be used for specifying custom serialization

```plaintext
tuple: {
    Name, 
    24204842, 
    San Jose 
}
```

Hashcode: 00101010001 0101

<table>
<thead>
<tr>
<th>Mask (0x11)</th>
<th>Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
</tr>
</tbody>
</table>
Fault Tolerance

• Operator state is checkpointed to persistent store
  ◦ Automatically performed by engine, no additional coding needed
  ◦ Asynchronous and distributed
  ◦ In case of failure operators are restarted from checkpoint state

• Automatic detection and recovery of failed containers
  ◦ Heartbeat mechanism
  ◦ YARN process status notification

• Buffering to enable replay of data from recovered point
  ◦ Fast, incremental recovery, spike handling

• Application master state checkpointed
  ◦ Snapshot of physical (and logical) plan
  ◦ Execution layer change log
Buffer Server

- In-memory PubSub
- Stores results emitted by operator until committed
- Handles backpressure / spillover to local disk
- Ordering, idempotency
Recovery Scenario

... EW₂, 1, 3, BW₂, EW₁, 4, 2, 1, BW₁

... EW₂, 1, 3, BW₂, EW₁, 4, 2, 1, BW₁

... EW₂, 1, 3, BW₂, EW₁, 4, 2, 1, BW₁

... EW₂, 1, 3, BW₂, EW₁, 4, 2, 1, BW₁

... EW₂, 1, 3, BW₂, EW₁, 4, 2, 1, BW₁

sum 0

sum 7

sum 10

sum 7
Processing Guarantees

At-least-once

- On recovery data will be replayed from a previous checkpoint
  - No messages lost
  - Default, suitable for most applications
- Can be used to ensure data is written once to store
  - Transactions with meta information, Rewinding output, Feedback from external entity, Idempotent operations

At-most-once

- On recovery the latest data is made available to operator
  - Useful in use cases where some data loss is acceptable and latest data is sufficient

Exactly-once

- At-least-once + idempotency + transactional mechanisms (operator logic) to achieve end-to-end exactly once behavior
End-to-End Exactly Once

• Becomes important when writing to external systems
• Data should not be duplicated or lost in the external system even in case of application failures
• Common external systems
  ◦ Databases
  ◦ Files
  ◦ Message queues
• Platform support for at least once is a must so that no data is lost
• Data duplication must still be avoided when data is replayed from checkpoint
  ◦ Operators implement the logic dependent on the external system
    • Aid of platform features such as stateful checkpointing and windowing
• Three different mechanisms with implementations explained in next slides
Compute Locality

• By default operators are deployed in containers (processes) on different nodes across the Hadoop cluster

• Locality options for streams
  ◦ RACK_LOCAL: Data does not traverse network switches
  ◦ NODE_LOCAL: Data transfer via loopback interface, frees up network bandwidth
  ◦ CONTAINER_LOCAL: Data transfer via in memory queues between operators, does not require serialization
  ◦ THREAD_LOCAL: Data passed through call stack, operators share thread

• Host Locality
  ◦ Operators can be deployed on specific hosts

• New in 3.4.0: (Anti-)Affinity (APEXCORE-10)
  ◦ Ability to express relative deployment without specifying a host
Data Processing Pipeline Example

App Builder
Monitoring Console

Logical View
Monitoring Console

Physical View
Real-Time Dashboards

Real Time Visualization
Maximize Revenue w/ real-time insights

PubMatic is the leading marketing automation software company for publishers. Through real-time analytics, yield management, and workflow automation, PubMatic enables publishers to make smarter inventory decisions and improve revenue performance.

**Business Need**

- Ingest and analyze high volume clicks & views in real-time to help customers improve revenue
  - 200K events/second data flow
- Report critical metrics for campaign monetization from auction and client logs
  - 22 TB/day data generated
- Handle ever increasing traffic with efficient resource utilization
- Always-on ad network

**Apex based Solution**

- DataTorrent Enterprise platform, powered by Apache Apex
- In-memory stream processing
- Comprehensive library of pre-built operators including connectors
- Built-in fault tolerance
- Dynamically scalable
- Management UI & Data Visualization console

**Client Outcome**

- Helps PubMatic deliver ad performance insights to publishers and advertisers in real-time instead of 5+ hours
- Helps Publishers visualize campaign performance and adjust ad inventory in real-time to maximize their revenue
- Enables PubMatic reduce OPEX with efficient compute resource utilization
- Built-in fault tolerance ensures customers can always access ad network
GE is dedicated to providing advanced IoT analytics solutions to thousands of customers who are using their devices and sensors across different verticals. GE has built a sophisticated analytics platform, Predix, to help its customers develop and execute Industrial IoT applications and gain real-time insights as well as actions.

Business Need

- Ingest and analyze high-volume, high speed data from thousands of devices, sensors per customer in real-time without data loss
- Predictive analytics to reduce costly maintenance and improve customer service
- Unified monitoring of all connected sensors and devices to minimize disruptions
- Fast application development cycle
- High scalability to meet changing business and application workloads

Apex based Solution

- Ingestion application using DataTorrent Enterprise platform
- Powered by Apache Apex
- In-memory stream processing
- Built-in fault tolerance
- Dynamic scalability
- Comprehensive library of pre-built operators
- Management UI console

Client Outcome

- Helps GE improve performance and lower cost by enabling real-time Big Data analytics
- Helps GE detect possible failures and minimize unplanned downtimes with centralized management & monitoring of devices
- Enables faster innovation with short application development cycle
- No data loss and 24x7 availability of applications
- Helps GE adjust to scalability needs with auto-scaling
Silver Spring Networks helps global utilities and cities connect, optimize, and manage smart energy and smart city infrastructure. Silver Spring Networks receives data from over 22 million connected devices, conducts 2 million remote operations per year.

**Business Need**
- Ingest high-volume, high speed data from millions of devices & sensors in real-time without data loss
- Make data accessible to applications without delay to improve customer service
- Capture & analyze historical data to understand & improve grid operations
- Reduce the cost, time, and pain of integrating with 3rd party apps
- Centralized management of software & operations

**Apex based Solution**
- DataTorrent Enterprise platform, powered by Apache Apex
- In-memory stream processing
- Pre-built operator
- Built-in fault tolerance
- Dynamically scalable
- Management UI console

**Client Outcome**
- Helps Silver Spring Networks ingest & analyze data in real-time for effective load management & customer service
- Helps Silver Spring Networks detect possible failures and reduce outages with centralized management & monitoring of devices
- Enables fast application development for faster time to market
- Helps Silver Spring Networks scale with easy to partition operators
- Automatic recovery from failures
Resources

- Learn more: [http://apex.apache.org/docs.html](http://apex.apache.org/docs.html)
- Subscribe - [http://apex.apache.org/community.html](http://apex.apache.org/community.html)
- Follow @ApacheApex - [https://twitter.com/apacheapex](https://twitter.com/apacheapex)
- More examples: [https://github.com/DataTorrent/examples](https://github.com/DataTorrent/examples)
- Slideshare: [http://www.slideshare.net/ApacheApex/presentations](http://www.slideshare.net/ApacheApex/presentations)
- [https://www.youtube.com/results?search_query=apache+apex](https://www.youtube.com/results?search_query=apache+apex)
Q&A