Big Data Trends

• Bigger data volumes
• More data sources
  – DBs, logs, behavioral & business event streams, sensors …
• Faster analysis
  – Next day to hours to minutes to seconds
• Newer processing models
  – MR, in-memory, stream processing, Lambda …
What is Pulsar

Open-source real-time analytics platform and stream processing framework
Business Needs for Real-time Analytics

• Near real-time insights
• React to user activities or events within seconds
• Examples:
  – Real-time reporting and dashboards
  – Business activity monitoring
  – Personalization
  – Marketing and advertising
  – Fraud and bot detection
Systemic Quality Requirements

• **Scalability**
  – Scale to millions of events / sec

• **Latency**
  – <1 sec delivery of events

• **Availability**
  – No downtime during upgrades
  – Disaster recovery support across data centers

• **Flexibility**
  – User driven complex processing rules
  – Declarative definition of pipeline topology and event routing

• **Data Accuracy**
  – Should deal with missing data
  – 99.9% delivery guarantee
Pulsar Real-time Analytics

- **Complex Event Processing (CEP):** SQL on stream data
- **Custom sub-stream creation:** Filtering and Mutation
- **In Memory Aggregation:** Multi Dimensional counting
Pulsar Framework Building Block (CEP Cell)

- Event = Tuples (K,V) – Mutable
- Channels: Message, File, REST, Kafka, Custom
- Event Processor: Esper, RateLimiter, RoundRobinLB, PartitionedLB, Custom
Pulsar Framework Flexibility

• Stream Processing Pipeline
  – Consist of loosely coupled stages (cluster of CEP cells)
  – CEP cells (channels and processors) configured as Spring beans
  – Declarative wiring of CEP cells to define pipeline
  – Each stage can adopt its own release and deployment cycles
  – Support topology changes without pipeline restart

• Stream Processing Logic
  – Two approaches: Java or SQL-like syntax through Esper integration
  – SQL statements can be hot deployed without restarting applications
Pulsar Real-time Analytics Pipeline

1. **Collector**
   - Producing Applications
   - BOT Detector
   - Enriched Events

2. **Sessionizer**
   - Enriched Sessionized Events

3. **Event Distributor**
   - Metrics Calculator
   - Other Real Time DataClients
   - Real Time Dashboard

4. **Real Time Metrics & Alert Consumer**
   - Metrics Store
Complex Event Processing in Real-time Analytics Pipeline

• Enrichment
• Filtering and mutation
• Analysis over windows of time (rolling vs. tumbling)
  – Aggregation
  – Grouping and ordering
• Stateful processing
• Integration with other systems
Event Filtering and Routing Example

```
insert into SUBSTREAM select D1, D2, D3, D4
from RAWSTREAM where D1 = 2045573 or D2 = 2047936 or D3 = 2051457 or D4 = 2053742; // filtering
@PublishOn(topics="TOPIC1")  // publish sub stream at TOPIC1
@OutputTo("OutboundMessageChannel")
@ClusterAffinityTag(column = D1);  // partition key based on column D1
select * FROM SUBSTREAM;
```
Aggregate Computation Example

// create 10-second time window context
create context MCContext start @now end pattern [timer:interval(10)];
// aggregate event count along dimension D1 and D2 within specified time window
context MCContext insert into AGGREGATE select count(*) as METRIC1, D1, D2 FROM RAWSTRE
AM group by D1,D2 output snapshot when terminated;
select * from AGGREGATE;
TopN Computation Example

```java
// create 60-second time window context
create context MCContext start @now end pattern [timer:interval(60)];
// sort to find top 10 event counts along dimensions D1, D2, and D3
// within specified time window
context MCContext insert into TOPITEMS select count(*) as totalCount, D1, D2, D3 from RawEventStream group by D1, D2, D3 order by count(*) limit 10;
select * from TOPITEMS;
```

- TopN computation can be expensive with high cardinality dimensions
- Consider approximate algorithms
- Implemented as aggregate functions e.g. select ApproxTopN(10, D1, D2, D3)
Pulsar Deployment Architecture
Availability And Scalability

- Self Healing
- Datacenter failovers
- State management
- Shutdown Orchestration
- Dynamic Partitioning
- Elastic Clusters
- Dynamic Flow Routing
- Dynamic Topology Changes
Pulsar Integration with Kafka

• Kafka
  – Persistent messaging queue
  – High availability, scalability and throughput

• Pulsar leveraging Kafka
  – Supports pull and hybrid messaging model
  – Loading of data from real-time pipeline into Hadoop and other metric stores
Messaging Models

Push Model
(At most once delivery semantics)

Pull Model
(At least once delivery semantics)
Pulsar Integration with Kylin

• Apache Kylin
  – Distributed analytics engine
  – Provide SQL interface and multi-dimensional analysis (OLAP) on Hadoop
  – Support extremely large datasets

• Pulsar leveraging Kylin
  – Build multi-dimensional OLAP cube over long time period
  – Aggregate/drill-down on dimensions such as browser, OS, device, geo location
  – Capture metrics such as session length, page views, event counts
Pulsar Integration with Druid

• Druid
  – Real-time ROLAP engine for aggregation, drill-down and slice-n-dice
• Pulsar leveraging Druid
  – Real-time analytics dashboard
  – Near real-time metrics like number of visitors in the last 5 minutes, refreshing every 10 seconds
  – Aggregate/drill-down on dimensions such as browser, OS, device, geo location
Key Takeaways

• Creating pipelines declaratively
• SQL driven processing logic with hot deployment of SQL
• Framework for custom SQL extensions
• Dynamic partitioning and flow control
• < 100 millisecond pipeline latency
• 99.99% Availability
• < 0.01% steady state data loss
• Cloud deployable
Future Development and Open Source

• Real-time reporting API and dashboard
• Integration with Druid and other metrics stores
• Session store scaling to 1 million insert/update per sec
• Rolling window aggregation over long time windows (hours or days)
• Dynamic Joins with graphs and RDBMS tables
• Hot deployment of Java source code
More Information

• **GitHub:** http://github.com/pulsarIO
  – repos: pipeline, framework, docker files

• **Website:** http://gopulsar.io
  – Technical whitepaper
  – Getting started
  – Documentation

• **Google group:** http://groups.google.com/d/forum/pulsar
Appendix
Twitter Storm/Spark Streaming vs Pulsar – Key Differences

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Pulsar</th>
<th>Storm/Trident</th>
<th>Spark Streaming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declarative Pipeline Wiring</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Pipeline stitching</td>
<td>Run time</td>
<td>Build time</td>
<td>Build time</td>
</tr>
<tr>
<td>Topology change requires reboot</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL support</td>
<td>Yes</td>
<td>No</td>
<td>Yes*</td>
</tr>
<tr>
<td>Hot deployment of processing rules</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Guaranteed Message Processing</td>
<td>Yes (batching)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pipeline Flow Control</td>
<td>Yes</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Stateful Processing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>