ebay inc
GLOBAL PLATFORM
& INFRASTRUCTURE

Pulsar

Realtime Analytics At Scale

Wang Xinglang

ebay

Agenda

- **Pulsar** : Real Time Analytics At eBay
  - Business Use Cases
  - Product Requirements
- **Pulsar** : Technology Deep Dive

Pulsar

Business Use Case: Behavioral Analytics

- Behavioral Analytics : Analysis Of User Behavior on eBay
  - Inputs : User Clicks, Impression, Conversions etc
  - Problem
    - In Session Targeting
    - Campaign Optimization

Requirements

- **Low Latency**
  - <1 Sec E2E latency
- **Scalability**
  - Millions of events/sec
- **Data Accuracy**
  - Data Sources are lossy

Business Use Case: Monitoring

- Monitoring : Application, Systems and Infrastructure Monitoring
  - Inputs : Heartbeats, SNMP, Logs
  - Problem
    - Collection, Aggregation of Metrics
    - Co-relation and Alerting

Requirements

- **Scalability**
  - 10s of Millions of events/sec
- **Availability**
  - 99.99% Uptime
- **Flexibility**
  - User Driven Rules
  - On the fly joins with other streams, data sources

Business Use Case: Security

- Security : Traffic Limiting, DOS Detection, Bot Detection
  - Inputs : Clicks, Impressions, Logs
  - Problem
    - Enforce Quotas in Real-time
    - Detect and Prevent Bad Bots

Requirements

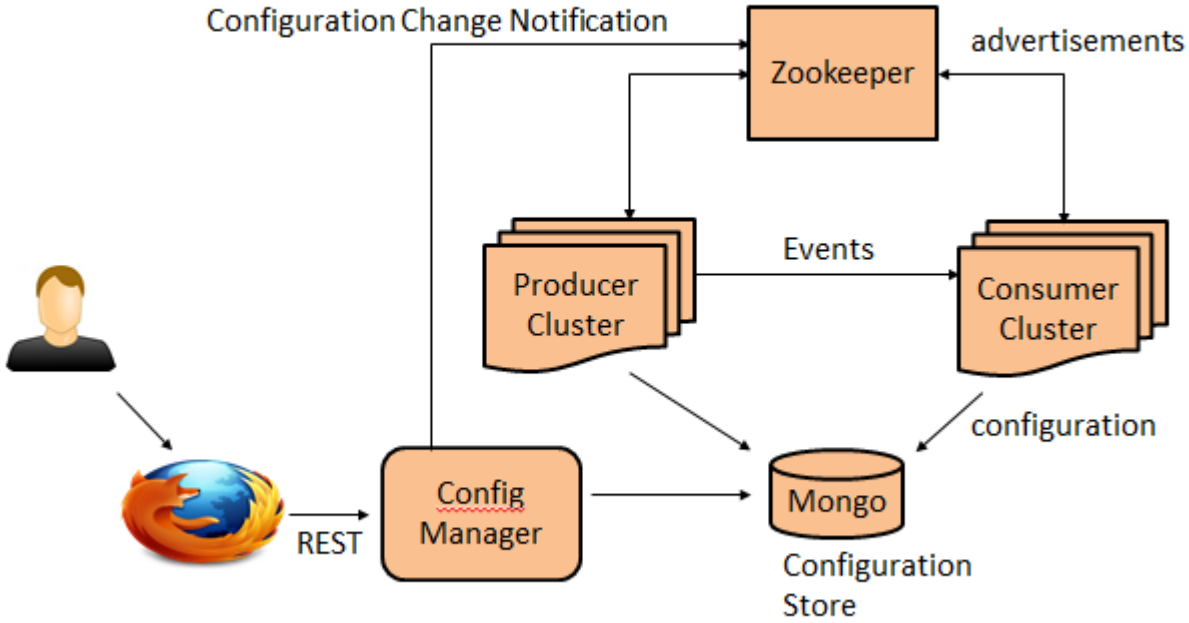
- **Scalability**
  - 10s of Millions of events/sec
- **Latency**
  - 1 sec SLA
- **Flexibility**
  - User Drive Custom Rules
- **Data Accuracy**
  - 99% is acceptable

Pulsar Product Requirements Summary

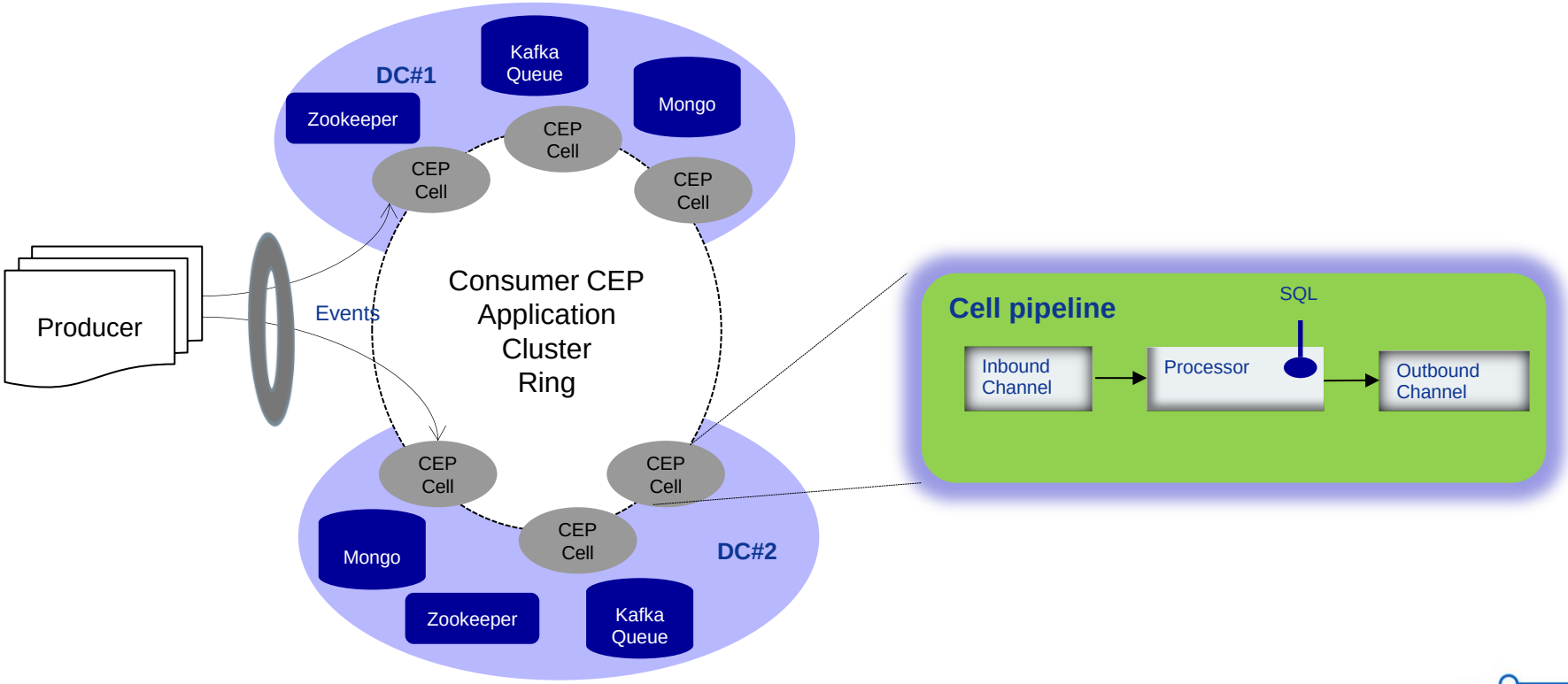
- **Scalability**
  - Scale to 10s of Millions Events/Sec
- **Latency**
  - <1 Sec Delivery of Events
- **Availability**
  - Highly Available System
  - No downtime during upgrades
  - Disaster Recovery Support across data centers
- **Flexibility**
  - User Driven Complex Rules. Eg (CPU/TPS > 3 AND ERRORS/TPS>0.2)
- **Data Accuracy**
  - Should deal with missing data
  - 99.9% Delivery Guarantee

Pulsar Technical Deep Dive

Pulsar CEP framework
JetStream

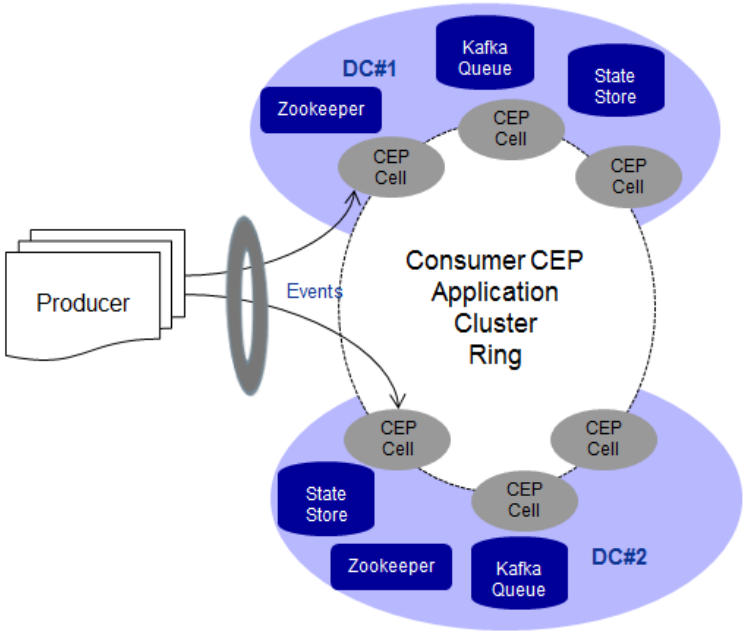


Pulsar Deployment Architecture

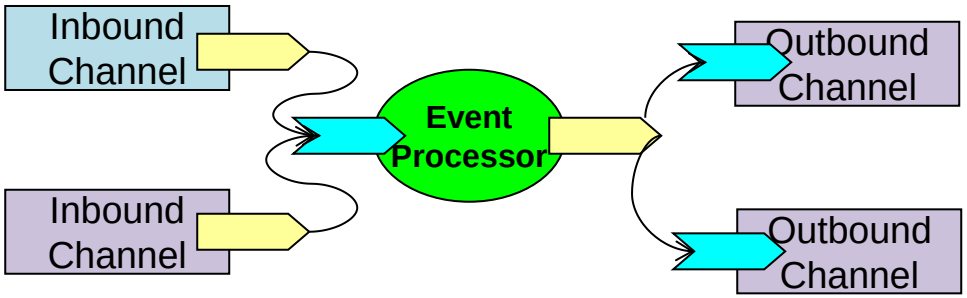


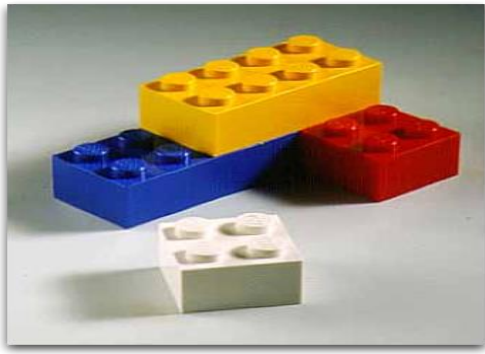
Availability And Scalability

- Multi datacenter failovers
- Dynamic Partitioning
- Elastic Clusters
- Self Healing
- Shutdown Orchestration
- Dynamic Flow Routing
- Dynamic Topology Changes



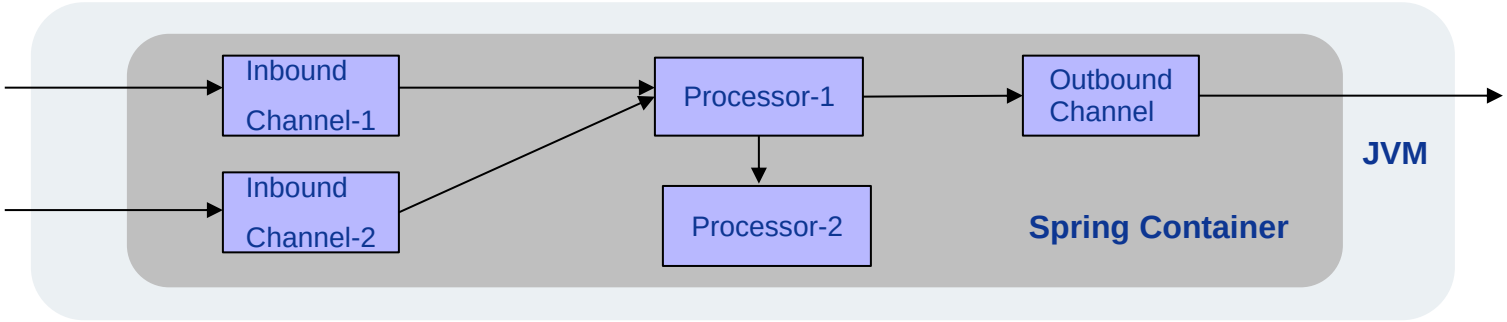
Pulsar Framework Building Blocks





- Event = Tuples (K,V) - Mutable
- Inbound Channel (Event Source) adapts to external world and sources events
- Outbound Channel (Event Sink) adapts to external world and consumes events
- Event Processor (Event Sink and Event Source)
- Pipelining, Batching & Flow control

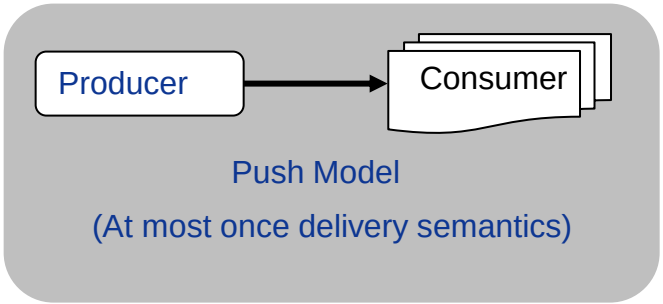
Pulsar Application (CEP Cell)

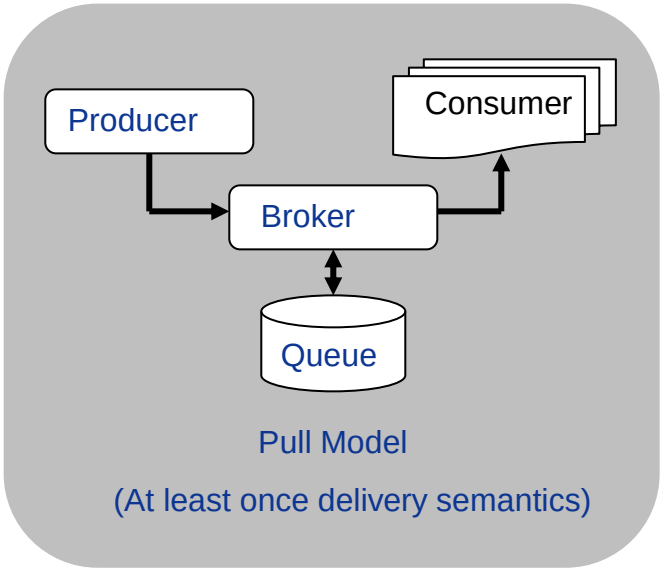


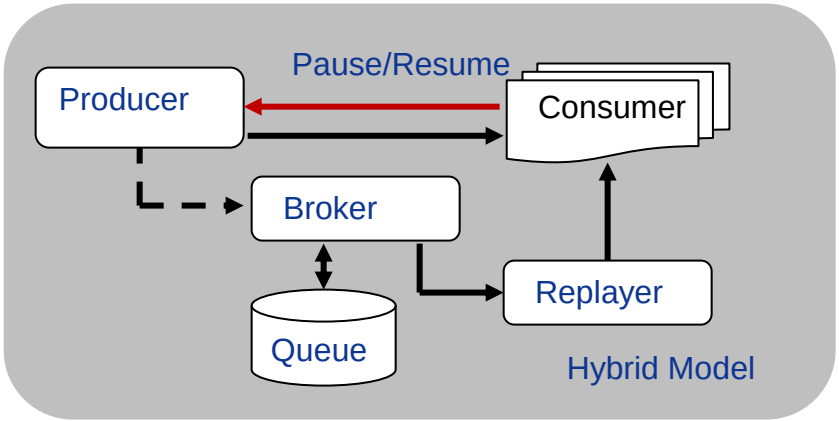

```
<bean id="InboundChannel-1"
class="com.ebay.jetstream.event.channel.messaging.InboundMessagingChannel">
        <property name="address" ref="InboundChannelAddress" />
        <property name="eventSinks">
                <list>
                        <ref bean="Processor-1" />
                </list>
        </property>
</bean>
```

```
<bean id="Processor-1"
class="com.ebay.jetstream.event.processor.esper.EsperProcessor">
        <property name="esperEventListener" ref="EsperEventListener" />
        <property name="configuration" ref="EsperConfiguration" />
        <property name="epl" ref="EPL" />
        <property name="eventSinks">
                <list>
                        <ref bean="outboundChannel" />
                </list>
        </property>
</bean>
```

Messaging And Clustering







Real Time Stream Processing

- The 8 Requirements of Real-Time Stream Processing (Ref. Stonebraker)
  - Rule 1: Keep the Data Moving
  - Rule 2: Query using SQL on Streams (StreamSQL)
  - Rule 3: Handle Stream Imperfections
  - Rule 4: Generate Predictable Outcomes
  - Rule 5: Integrate Stored and Streaming Data
  - Rule 6: Guarantee Data Safety and Availability
  - Rule 7: Partition and Scale Applications Automatically
  - Rule 8: Process and Respond Instantaneously
  - **Rule 9: Continuous Processing**
  - **Rule 10: Dynamic Topology Changes**
  - **Rule 11: Dynamic SQL changes & flow control**

Flexibility

Flexibility : CEP Engine (Esper)

- SQL like language for specifying processing rules
- Analysis over rolling and tumbling windows of time
- Filtering and Joining streams
- Grouping and Ordering output
- For routing events between stages and between clusters
- Event Mutation
- Correlation
- Patterns

Flexibility : DATA MODEL

```
<bean id="EventDefinition" class="com.ebay.jetstream.event.processor.esper.EsperDeclaredEvents">
        <property name="eventTypes">
                <list>
          <bean class="com.ebay.jetstream.event.processor.esper.MapEventType">
                  <property name="eventAlias" value="RawEvent"/>
                  <property name="eventFields">
                   <map>
                          <entry key="D1" value="java.lang.String"/>
                         <entry key="D2" value="java.lang.String"/>
                         <entry key="D3" value="java.lang.String"/>
                         <entry key="D4" value="java.lang.Integer"/>
                         <entry key="D5" value="java.lang.Integer"/>
                         <entry key="createtime" value="java.lang.Long"/>
                   </map>
                  </property>
          </bean>
                <list>
        </property?
</bean>
```

key	value
js_ev_type	RawEvent
D1	“2045573”
D2	“79843743994”
D3	“908098404060”
D4	73840754
D5	1
createtime	7076303760

Flexibility : Event Processing Language (EPL)

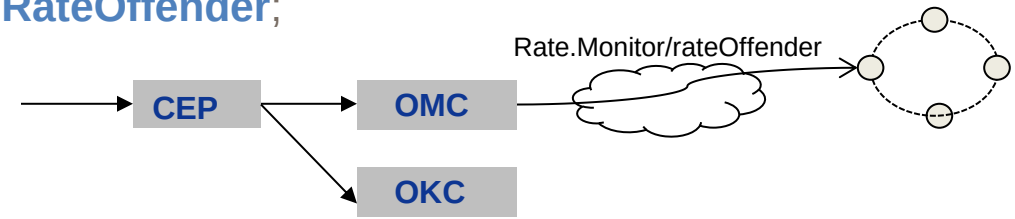
Metrics over Rolling window

INSERT INTO **RateOffender** SELECT D1, count(*) AS total_count FROM **RawEvent(D5 = 1).win:time(10 sec)** group by D1 having count(*) > 8 output last every 1 second;

@OutputTo(“OMC")

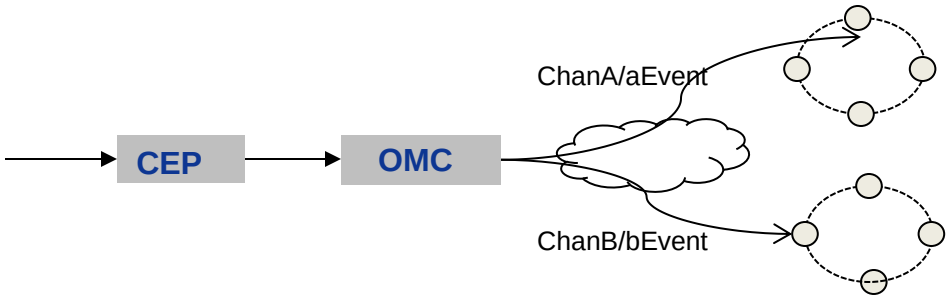
@PublishOn(topics=“Rate.Monitor/rateOffender”)

SELECT D1, ‘block’ as Policy FROM **RateOffender**;



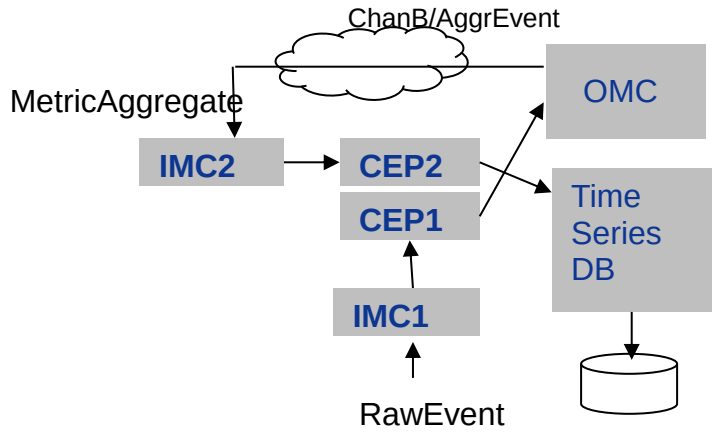
Flexibility : Event Processing Language (EPL) –
Filtering, Mutation & Routing

```
INSERT INTO STREAMROUTE SELECT D1, D2, D3, D4 FROM RawEvent(D1 in
('2045573','2053742') and Common.isNumeric(D2) and D2 != '0');


@OutputTo("OMC")
@PublishOn(topics="ChanA/aEvent,ChanB/bEvent")
@ClusterAffinityTag(colname = "D2")
SELECT * FROM STREAMROUTE;
```



Flexibility : Event Processing Language (EPL) –
Creating Multidimensional Metrics Over Tumbling Windows

```
create context MCContext start @now end after 10 seconds;
context MCContext
insert into MetricAggregate select count(*) as count, D1, D2, ‘M1' as metricName from
RawEvent(D1 is not null and D2 is not null) group by D1, D2 output snapshot when
terminated;


@OutputTo(“OMC")
@PublishOn(topics="ChanB/AggrEvent")
@ClusterAffinityTag(dimension=@CreateDimension(
name="grpdim", dimensionspan=“D1, D2, M1"))
select * from MetricAggregate;
```



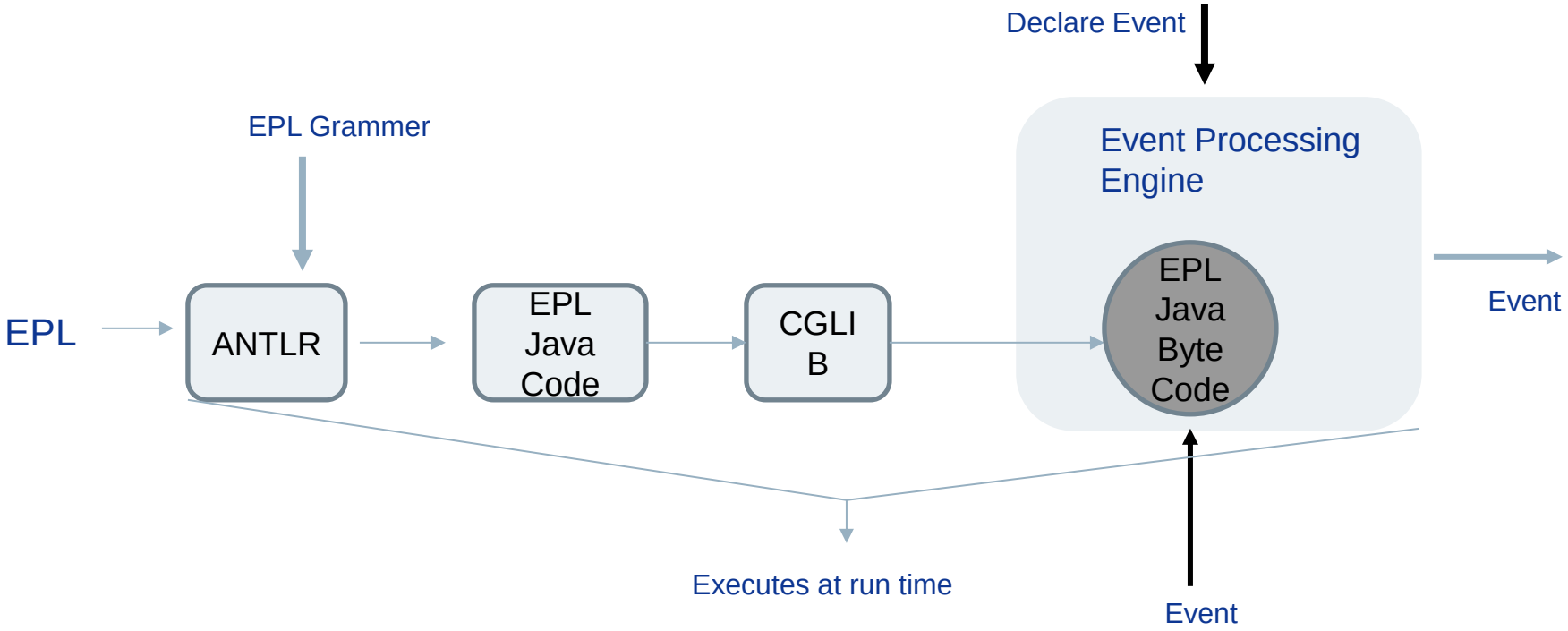
Flexibility : Top N, Distinct Count & Percentiles Computation

insert into TOPITEMSTREAM select count(*) as count, D1 from RawEvent() group by D1;
select * from TOPITEMSTREAM order by count;

- Computations are heavy both in time and space.
- High Cardinality dimensions makes it worse
- Consider approximate algorithms
  - Margin of error – around1%
  - Costs very little in space and time
  - Trade off accuracy for performance
- Implemented as aggregate functions

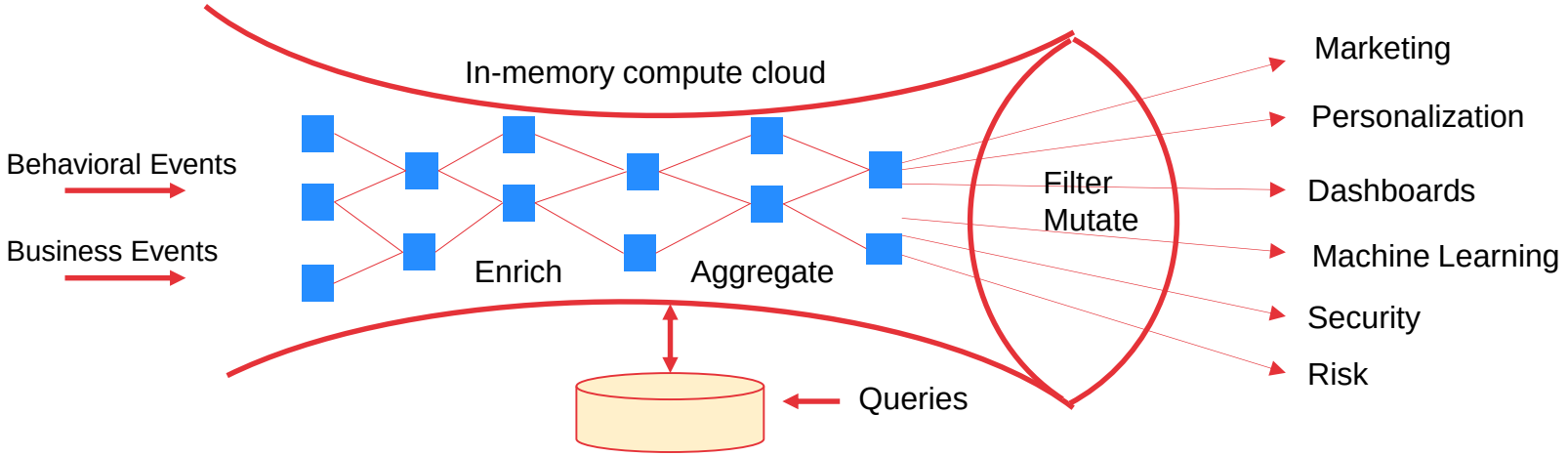
  insert into TOPITEMSTREAM select TopN(1000, 10, D1) as topItems from RawEvent();
  select * from TOPITEMSTREAM;

Flexibility : Hot Deployment of EPL



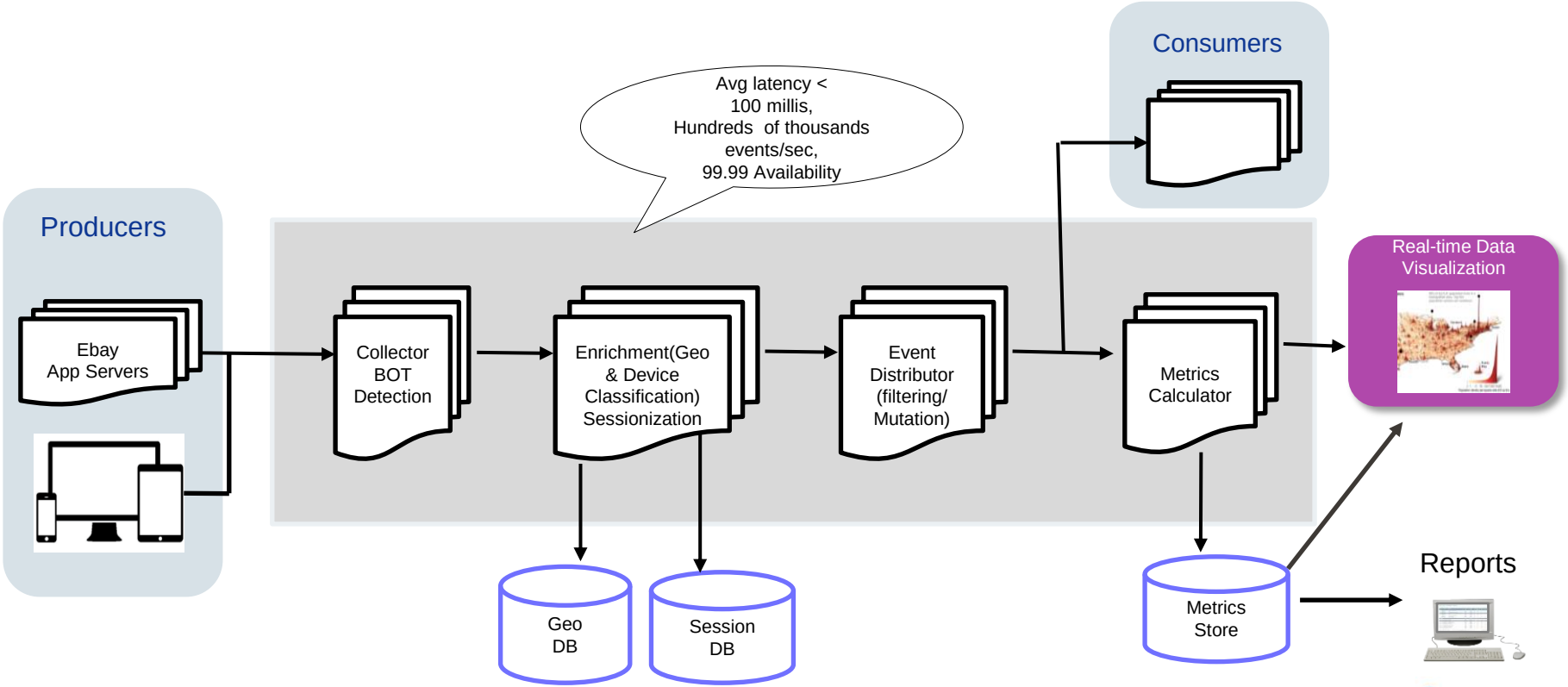
Pulsar Stream Pipeline

Pulsar Real-time Analytics



- **Complex Event Processing**: SQL on stream data
- **Custom sub-stream creation:** Filtering and Mutation
- **In Memory Aggregation:** Multi Dimensional counting

Pulsar Real Time Pipeline



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

http://gopulsar.io

Q&A

Thanks