Fast Cars, Big Data
How Streaming Can Help Formula1?
Agenda

• What’s the point of data in motorsports?
• Demo
• Architecture
• What’s next?
• Q&A
What’s the point of data in motorsports?
Some of the most heavily instrumented objects in the world

https://www.metasphere.co.uk/telemetry-data-journey-f1/
Got examples?

- Up to **300 sensors** per car
- Up to **2000 channels**
- Sensor data are sent to the paddock in **2ms**
- **1.5 billions** of data points for a race
- **5 billions** for a full race weekend
- **5/6Gb** of compressed data per car for 90mn

**US Grand Prix 2014 : 243 Tb** (race teams combined)
Data Communication
Data in Motorsports

- RPM
- Speed
- Lateral acceleration
- Gear
- Throttle
- Brake pressure

Race Strategy

Race Strategy Option Comparison

Time diff between Option-A versus Option-B

How does that work?
Production System Outline

FIA

Team Engineers

Factory Operations Room

MAPR EDGE

MAPR EDGE

Kafka API

Kafka API

Kafka API

Stream

Stream

Stream

Stream

Topic

Topic

Topic

Topic

Ad-hoc analysis

Real-time analysis

Reporting

Spark Streaming

Other Data Sources

MAPR STREAMS

MAPR-DB

APACHE

DRILL

Parquet

MAPR

MAPR

© 2017 MapR Technologies

MapR Confidential
Simplified Demo System Outline

- Torcs race simulator

MapR Sandbox

- D3
- Real-time analysis
- Ad-hoc analysis

MapR Streams

- Kafka API
- Apache Flink
TORCS for Cars, Physics & Drivers

• The Open Racing Car Simulator
  – http://torcs.sourceforge.net/
• a pseudo-physics based racing simulator
• AI Game & Research Platform
Architecture

Event Producers

Kafka API

sensor data

Real Time

Kafka API

Stream
Topic

MAPR-DB

Analytics with SQL

Event Consumers

https://github.com/mapr-demos/racing-time-series
Demo
How to Capture IOT events at scale?
Traditional Message queue

Legacy Message Queue: Message rate <100K/s

Publish Acks

Millions of Sources

Consume Acks

Hundreds of Destinations

insert delete
Data Streaming
Moving millions of events per h\text{mn}s
What is Kafka?

- http://kafka.apache.org/
- Created at LinkedIn, open sourced in 2011
- Distributed messaging system built to scale
- Implemented in Scala / Java
Topics Organize Events Into Categories
And Decouple Producers from Consumers
Topics are Partitioned for Concurrency and Scalability
Partition message order is like a Queue

Partitions:
• Messages are appended to the end
• delivered in order received

Offset:
• Sequential id of a message
Partition message order is like a Queue

- Producers Append New messages to end
- Consumers Read from front
- Read cursor: offset ID of most recent read message
Unlike a Queue, events are still persisted after delivered

- Messages remain on the partition, available to other consumers
- Minimizes Non-Sequential disk read-writes
Processing of the same message for different views
Processing of the same message for different views

Key-Val  Document  Graph

Inserts

Updates

Wide Column  Time Series  Relational
MapR Streams

- Distributed messaging system built to scale
- Use Apache Kafka API 0.9.0
- No code change
- Does not use the same “broker” architecture
  - Log stored in MapR Storage (Scalable, Secured, Fast, Multi DC)
  - No Zookeeper
Produce Messages

ProduceRecord<String, byte[]> rec = new ProducerRecord<>(
    "/apps/racing/stream:sensors_data",
    eventName,
    value.toString().getBytes());

producer.send(rec, (recordMetadata, e) -> {
    if (e != null) {
        ...}
    });

producer.flush();
Consume Messages

```java
long pollTimeOut = 800;
while (true) {
    ConsumerRecords<String, String> records = consumer.poll(pollTimeOut);
    if (!records.isEmpty()) {
        Iterable<ConsumerRecord<String, String>> iterable = records.iterator();

        while (iterable.hasNext()) {
            ConsumerRecord<String, String> record = iterable.next();
            System.out.println(record.value());
        }
    }
}
```
Consume Messages

```java
long pollTimeOut = 800;
while(true) {

    ConsumerRecords<String, String> records = consumer.poll(pollTimeOut);
    if (!records.isEmpty()) {
        Iterable<ConsumerRecord<String, String>> iterable = records::iterator;

        StreamSupport
        .stream(iterable.spliterator(), false)
        .forEach((record) -> {
            // work with record object
            record.value();
        });

        consumer.commitAsync();
    }
}
```
Where/How to store data?
Big Datastore

Distributed File System
HDFS/MapR-FS

NoSQL Database
HBase/MapR-DB

....
Relational Database vs. MapR-DB/HBase

RDBMS

Storage Model

MapR-DB/HBase

Distributed Joins, Transactions are Expensive

Normalized schema → joins for queries can cause bottleneck

Data that is accessed together is stored together
Designed for Partitioning and Scaling

Fast Reads and Writes by Key

Data is automatically partitioned by Key Range
What’s next?
Tables as Objects, Objects as Tables

Data collected is in JSON

Row-wise form

List of objects

Column-wise form

Object containing lists
Sensor Data V1

• For speed group sensor data
• Each message contains array of Sensor data:
  – Speed (m/s)
  – RPM
  – Distance (m)

```json
{  
  "_id":"1.458141858E9/0.324",
  "car" = "car1",
  "timestamp":1458141858,
  "racetime":0.324,
  "records":
    [  
      {  
        "sensors":{
          "Speed":3.588583,
          "Distance":2003.023071,
          "RPM":1896.575806
        },
        "racetime":0.324,
        "timestamp":1458141858
      },
      {  
        "sensors":{
          "Speed":6.755624,
          "Distance":2004.084717,
          "RPM":1673.264526
        },
        "racetime":0.556,
        "timestamp":1458141858
      }
    ]
}
Capture more data
Sensor Data V2

- Add 2 more data points:
  - Speed (m/s)
  - RPM
  - Distance (m)
  - Throttle
  - Gears
  - ...
Add new services
New Data Service

Event Producers

sensor data

Real Time

Stream

Topic

MAPR-DB

Analytics with SQL

Spark

MAPRSTREAMS

Alerts

Event Consumers

https://github.com/mapr-demos/racing-time-series
• Cluster Computing Platform
• Extends “MapReduce” with extensions
  – Streaming
  – Interactive Analytics
• Run in Memory
• http://spark.apache.org/
• Streaming Dataflow Engine
  – Datastream/Dataset APIs
  – CEP, Graph, ML
• Run in Memory
• https://flink.apache.org/
Apache Flink Stack

Libraries

DataStream API
Stream Processing

DataSet API
Batch Processing

Runtime
Distributed Streaming Data Flow

Streaming and batch as first class citizens.
A Stream Processing Pipeline

collect -> capture -> analyze -> Store and serve

Topic

kafka

MAPR-DB

Flink
Flink Consumer processing

```java
DataStream stream = env.addSource(
    new FlinkKafkaConsumer09<>
        ("/apps/racing/stream:sensors_data",
         new JSONDeserializationSchema(),
         properties)
);

stream.flatMap(new CarEventParser())
    .keyBy(0)
    .timeWindow(Time.seconds(10))
    .reduce(new AvgReducer())
    .flatMap(new AvgMapper())
    .map(new AvgPrinter())
    .print();
```
Flink Consumer processing

class AvgReducer implements ReduceFunction<Tuple3<String, Float, Integer>> {
    @Override
    public Tuple3<String, Float, Integer> reduce(Tuple3<String, Float, Integer> value1, 
        Tuple3<String, Float, Integer> value2) {
        return new Tuple3<>(value1.f0, value1.f1 + value2.f1, value1.f2+1);
    }
}

class AvgMapper implements FlatMapFunction<Tuple3<String, Float, Integer>, 
    Tuple2<String, Float>> {
    @Override
    public void flatMap(Tuple3<String, Float, Integer> carInfo, 
        Collector<Tuple2<String, Float>> out) throws Exception {
        out.collect( new Tuple2<>( carInfo.f0 , carInfo.f1/carInfo.f2 ) );
    }
}
What is Drill?

• SQL query engine for data/apps on Hadoop
• Schema optional - query schema-less data in situ
• Provides low latency interactive response times
• SQL engine on “everything”
  • Semi structured formats – Ex: JSON
  • Structured formats – Ex: parquet
• Ecosystem components – Hbase, MapRDB, Hive
Apache Drill: Drillbit components

- RPC Endpoint
- SQL Parser
- Optimizer
- Storage plug ins
  - Built in plugins for Hive, Hbase, MapRDB, DFS, traditional Linux filesystems
Apache Drill Architecture
Drill Query Example

\[
\text{SELECT } * \text{ FROM } \text{dfs.logs`/telemetry/all_cars`}
\]

LIMIT 10
Drill Query Example: average speed by car and race

```
SELECT race_id, car, 
    ROUND(AVG(`t`.`records`.`sensors`.`Speed`),2) as 'mps', 
    ROUND(AVG(`t`.`records`.`sensors`.`Speed` * 18 / 5), 2) as 'kph'
FROM ( 
    SELECT race_id, car, 
        flatten(records) as `records` 
    FROM dfs.`/telemetry/all_cars` 
) AS `t`
GROUP BY race_id, car
```
Streaming Architecture & Formula 1

• Stream & Process data in real time
  – Use distributed & scalable processing and storage
  – NoSQL Database
• Decouple the source from the consumer(s)
  – Dashboard, Analytics, Machine Learning
  – Add new use case....
Streaming Architecture & Formula 1

• Stream & Process data in real time
  – Use distributed & scalable processing
  – NoSQL Database, Distributed File System

• Decouple source from consumer(s)
  – Dashboard, Analytics, Machine Learning
  – Add new use case…

This is not only about Formula 1!
(Telco, Finance, Retail, Content, IT)
To Learn More:

• Download example code
  – Read explanation of example code
  – https://mapr.com/blog/fast-cars-fast-data-formula1
MONITORING UBER DATA USING SPARK MACHINE LEARNING, STREAMING, THE KAFKA API, AND VERT.X
MONITORING UBER DATA USING SPARK MACHINE LEARNING, STREAMING, THE KAFKA API, AND VERT.X

MONITORING UBER DATA USING SPARK MACHINE LEARNING, STREAMING, THE KAFKA API, AND VERT.X

One Last Thing…

Events Producers

BLE

MapR-Streams
Apache Kafka

Web Socket

Events Consumers

https://github.com/mapr-demos/racing-time-series
MapR Blog

- https://www.mapr.com/blog/

Featured Author

Carol McDonald

SOLUTIONS ARCHITECT, MAPR

Carol has extensive experience as a developer and architect building complex mission critical applications in the Banking, Health Insurance and Telecom Industries. As a Java Technology Evangelist at Sun Microsystems, Carol traveled all over the world speaking at Sun Tech Days, JUGs, Companies, and Conferences. She is a recognized speaker in Java communities.
Connect with fellow Apache Hadoop and Spark professionals

- Find answers
- Ask technical questions
- Join on-demand training course discussions
- Follow release announcements
- Share and vote on product ideas
- Find Meetup and event listings

community.mapr.com
MapR Converged Data Platform

Open Source Engines & Tools
- hadoop
- Spark
- Apache Drill
- Search and Others

Commercial Engines & Applications
- Cloud and Managed Services
- VERTICA
- Custom Apps

Utility-Grade Platform Services

Data
- Web-Scale Storage: MapR-FS
  - High Availability
  - Real Time
  - Unified Security
  - Multi-tenancy
- Database: MapR-DB
  - Disaster Recovery
  - Global Namespace
- Event Streaming: MapR Streams
  - Unified Management and Monitoring
Building a Complete Data Architecture

MapR File System (MapR-FS)

MapR Streams

MapR Database (MapR-DB)

MapR Converged Data Platform