Game Optimization

Stream Me Up Before You Go-Go: A Tale of 2 Streams

Dionysios Kakolyris

PL Seminar 2018

http://autofire.io
contact@autofire.io

What We Do

Game Intelligence & Optimization cloud service

- Profile your player
- Predict her behavior
- Adapt your game to her
- Let **AI** optimize it
- **All this made dead easy for you**

http://autofire.io
contact@autofire.io

What We Do

SematicLens with **GameOptix** technology

Scalable Distributed Stateful Fault-Tolerant Real-Time Computation System

- Inter-operability
- Focus on Evolution processes
- AI Service oriented at games / gamification processes (i.e. challenge / reward cycle)
- In production

http://autofire.io
contact@autofire.io

About

STREAM PROCESSING

- Some data naturally comes as a never-ending stream of events
- One-pass nature
- Sometimes data is huge and it is not even possible to store it
- It has evolved to a Big data technology

Background

The story so far...

2011: Heterogeneous Concurrency Framework for Multimedia Processing *(Diploma Thesis)*

2013: LDB Stream DataBase *(BugSense - acquired by Splunk)*

2014: Fleet Management Dynamic Messaging *(DGA)*

2016: SemanticLens Stream Processing *(Autofire)*

Patterns

CQS: Bertrand Meyer (Eiffel Language - 1986)

Design by Contract

Command-Query Separation

Uniform Access Principle

Single Choice Principle

Open-Closed Principle

PATTERNS

Command-Query Responsibility Segregation **(CQRS)**

Different models for Writing (Command) **and Reading** (Query)

- Greg Young

Event Sourcing **(ES)**

Ensures that all changes to application state are stored as a sequence of events

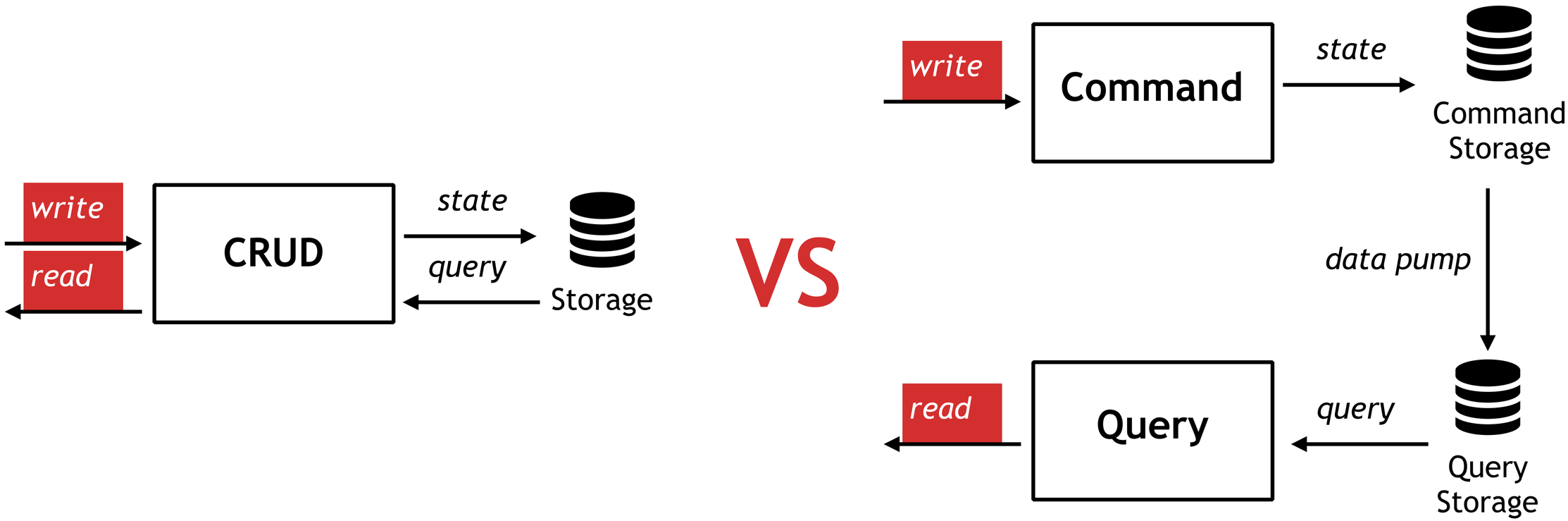
- Martin Fowler

Domain-Driven Design **(DDD)**

The domain **is the "sphere of knowledge and activity around which the application logic revolves"**

- Eric Evans

CRUD vs CQRS


write
read
CRUD
state
query
Storage
VS
write
Command
state
Command
Storage
data pump
read
Query
query
Query
Storage

SEMANTICLENS

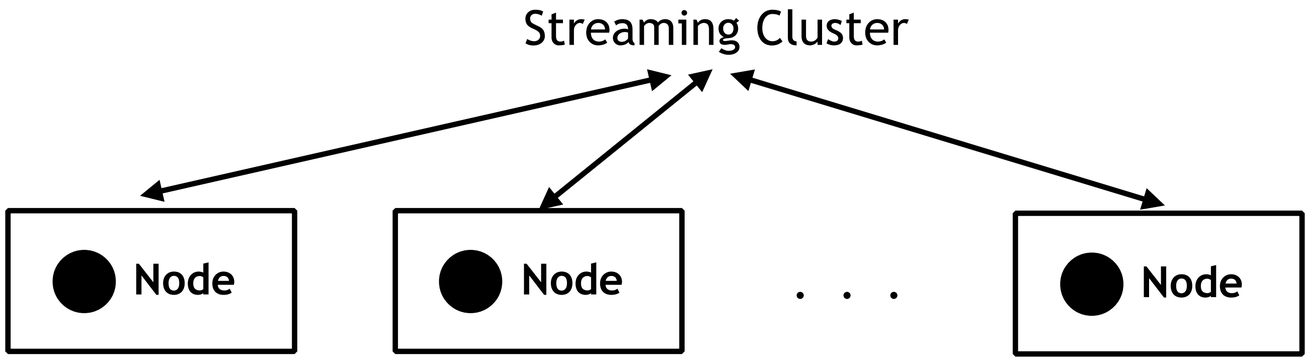
http://autofire.io
contact@autofire.io

Why

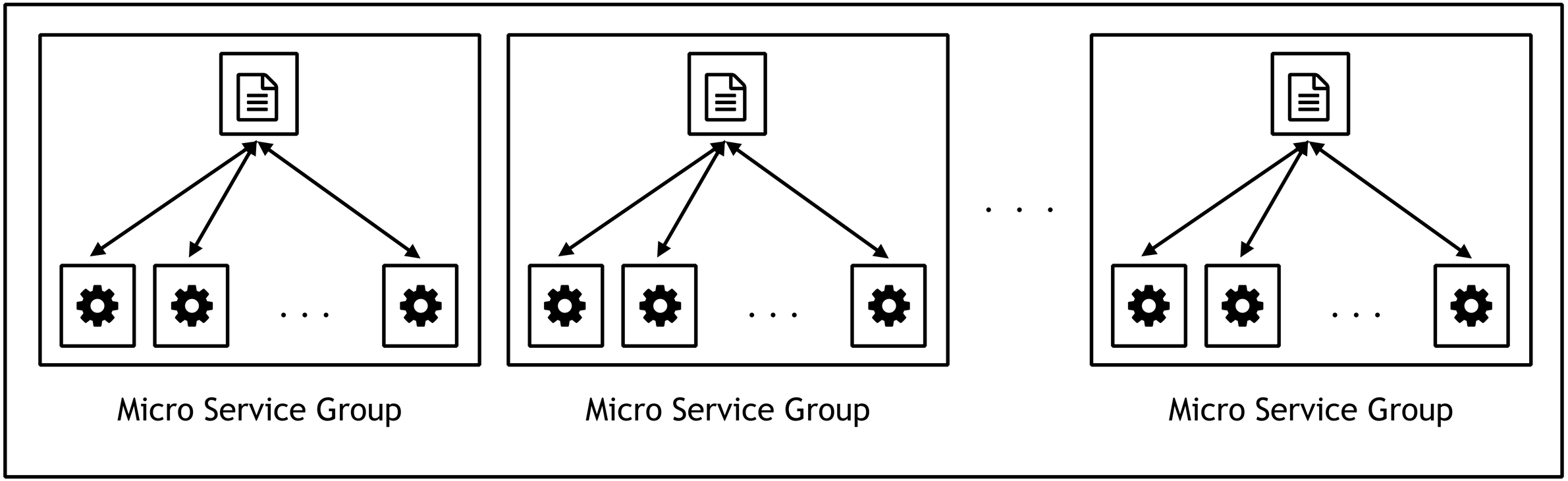
Focus on the Business Logic - not so simple / free

- State management
- I/O, Concurrency
- Lifecycle (init, tear down)
- Efficiency, Scalability
- Flexibility, Modularity
- Error handling, Supervision
- Robustness, Correctness
- *...et al*

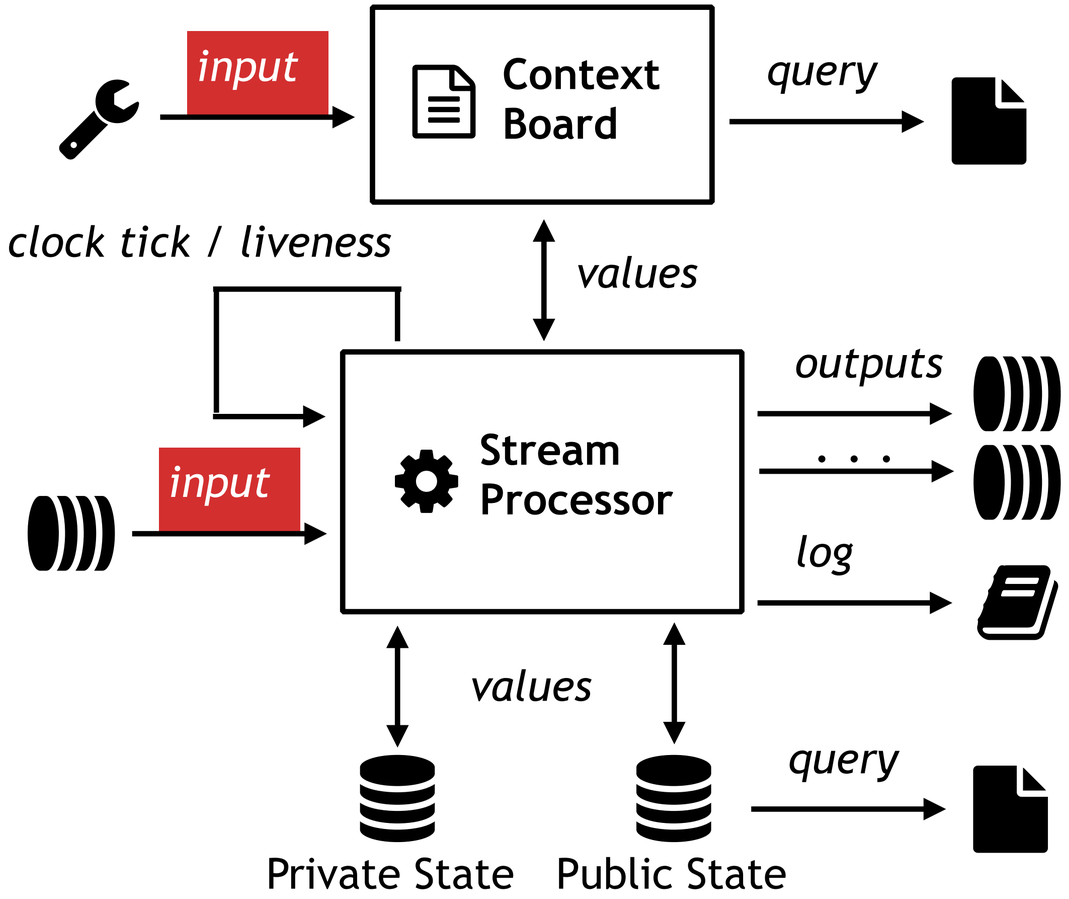
SemanticLens Architecture



SemanticLens Architecture



Micro Service: Processor & Board



⚙ Micro Service: Processor & Board

Down to the core

- Board
  - `getValue(), setValue()`
  - Micro Service Group
  - **Key, Value** Hash Map
  - Can be queried
  - Changes (deltas) are not recorded by default (ideal for **CRUD**)

⚙ Micro Service: Processor & Board

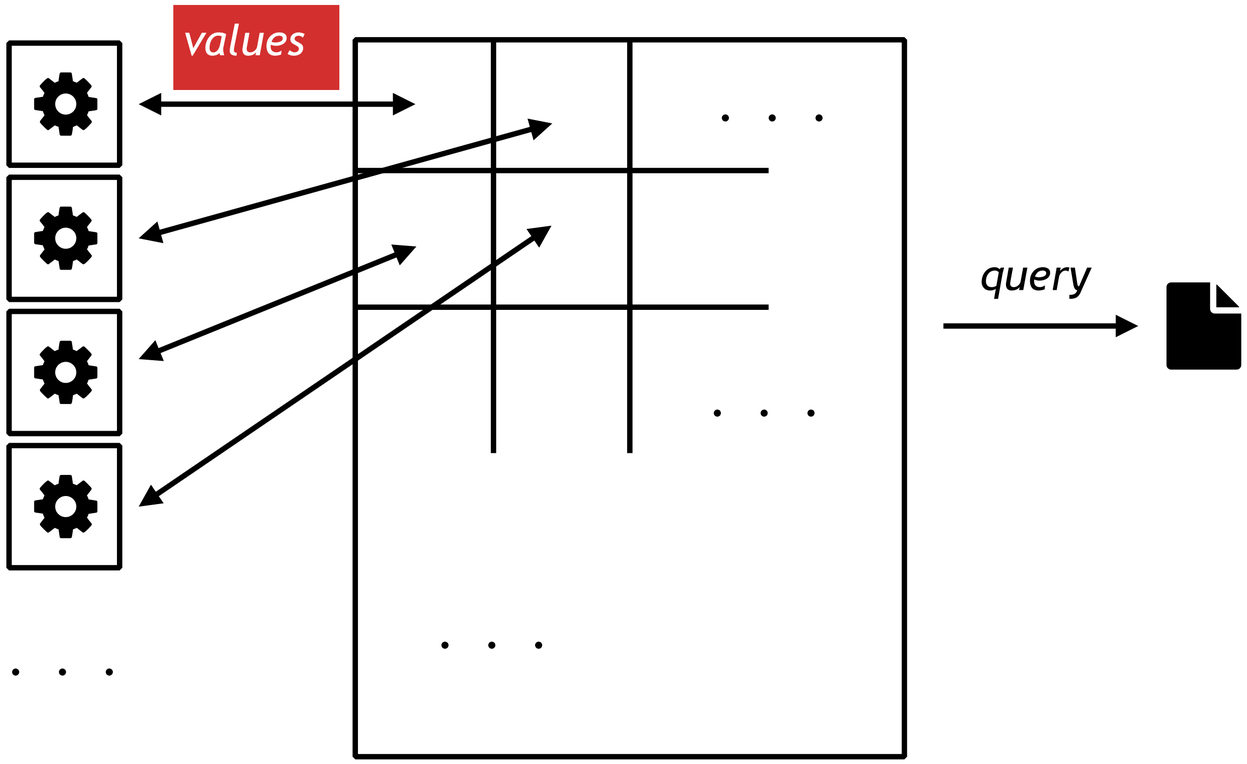
Down to the core

- Processor
  - `processInput()`
  - Micro Service
  - **Structure** with attributes, methods and **life cycle**
  - Cannot be directly queried (its **Public State** can be queried)
  - Changes (deltas) can be recorded (ideal for **CQRS Command**)

Computation / Processing Model

- Unit of Computation: Processor
- A Processor does **sequential** processing
- A Processor is **implementing** a Micro Service **Business Logic**
- **Subject**: Each Processor's Entity ID
- A Processor resembles (a little bit) a Java 9+ `j.u.c.Flow.Processor<byte[], byte[][]>`
- A Board contains Processors' **configuration**
- A Board serves requests **sequentially**

QUERY



Public States

Patterns

OLAP Hypercubes / Fact Tables

- Domain: DataBase
- Fact: Table
- Dimensions: Row
- Value: Column

Query

Query API Overview

- **Functional** style (as opposed to SQL-like)
- Abstract **Organization**: DataBase, Table, Row, Column
- **Sources**: Fetch filtered Organizations
- **Flow**:
    - Tables(filter, duplicate, drop, flatten, join)
    - Rows(reduce)
    - Columns(map, reduce, apply_function)
- **Sink**: View optionally written to a DataBase (Materialized View)
- View: Data Frame

REQUEST

```
{
  "sources": [{
    "dataBase": "reducers",
    "filterFunction": {
      "tables": ["STA-H"],
      "columns": ["Count", "Uniq"],
      "rows": ["row_filter", {"dimensionsPaths": [[".", [
        [["Y", ["from", 2018, "to", 2018]], ["M", ["from", 11, "to", 11]], ["D", ["from", 5, "to", 7]], ["H", ["from", 0, "to", 23]]],
        []
      ]]]}]
    }
  }],
  "flow": {
    "imperativeOption":[
      ["reduce_rows", {
        "tables": ["STA-H"],
        "reduceFunction": [["fn", ["append"]], ["D"]]
      }]
    ],
    "viewOption": ["ASC"]
  }
}
```

RESPONSE


```
{
  "response": {
    "sinkType": "SparseDF",
    "flowResult": {
      "result": {
        "code": 1,
        "analyzer": [
          ["rowsFilter(xxxx;STA-H)", 0],
          ["getDimensionsPathRows(xxxx;STA-H)::getRange", 278],
          ["addBinaryIR", 1],
          ["reduceRows", 0],
          ["toSparseDF", 1]
        ]
      },
      "body": {"dataFrame": [
        ["/Y=2018/M=11/D=05/", [["STA-H#Count", 1361]]],
        ["/Y=2018/M=11/D=06/", [["STA-H#Count", 1243]]],
        ["/Y=2018/M=11/D=07/", [["STA-H#Count", 439]]]
      ]}
    }
  }
}
```

Capabilities

- Data Frame adapter **(ready-to-render results)**
- **Modular** design that **scales (up and out)** and **inter-operates** with other systems
- **Cloud** and **On-prem** deployment capabilities
- Allows implementation of industry standard and proprietary **algorithms** from various science disciplines (Machine Learning, Data Mining, Evolution Theory, Control Theory, Linear Programming / Optimizations, Approximation Theory, Finite Element Analysis)

Patterns

CQRS / ES (Command Query Responsibility Segregation with Event Sourcing)

- Aggregate Root: Processor
- Aggregate: Subject
- Command / Event: Protocol, Processor input validation, Processor input to output
- Service Group
- Event Stream is persisted

Compare To

- Apache Apex
- Apache Samza
- Apache Storm
- Apache Flink
- Apache Spark Streaming
- Google Pregel & Dremel
- Lightbend Lagom
- Apache Drill

GameOptix

http://autofire.io
contact@autofire.io

Autofire GameOptix

- Games Domain
- SemanticLens Processors implementing **Game Event processing** *(Scala, Scalaz)*
- SemanticLens **Queries** *(Scala, Scalaz)*
- **Off-line post-processing** *(Hadoop)*
- Autofire **Client SDKs** (in Autofire enabled devices)
- Autofire **Dashboard** *(PHP Laravel, JavaScript Vue.js)*

http://autofire.io
contact@autofire.io

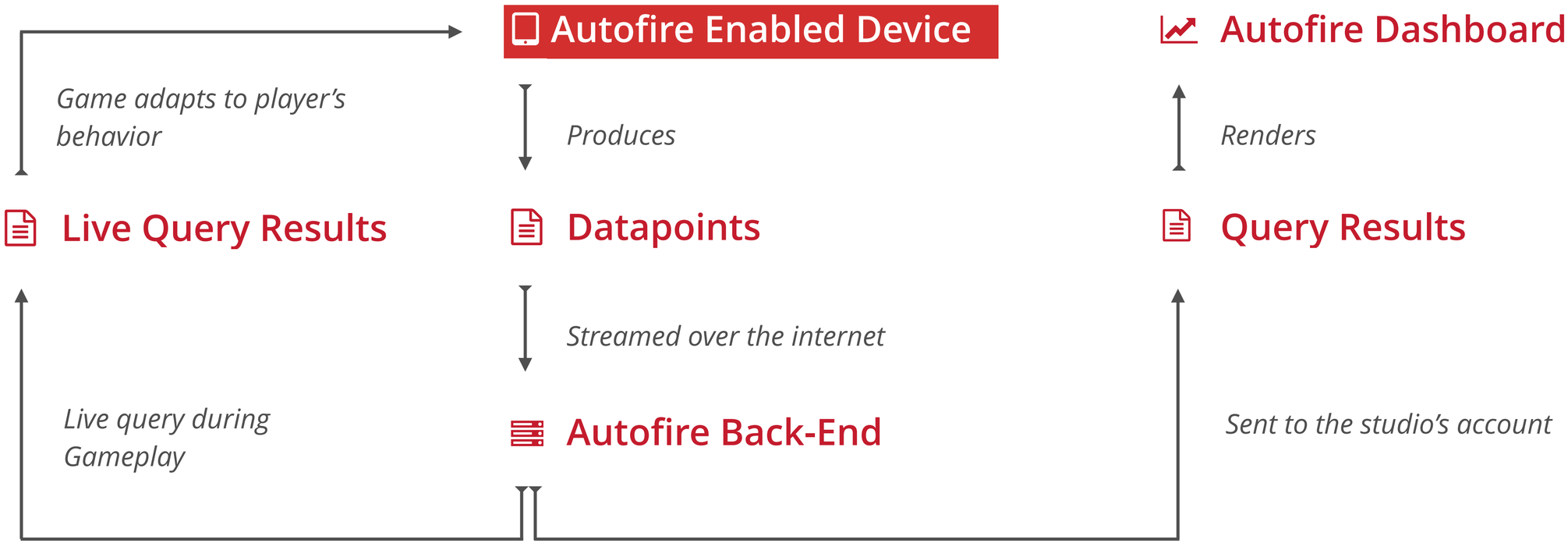
AUTOFIRE GAMEOPTIX

Gameplay Evolution **Analysis & Optimization**

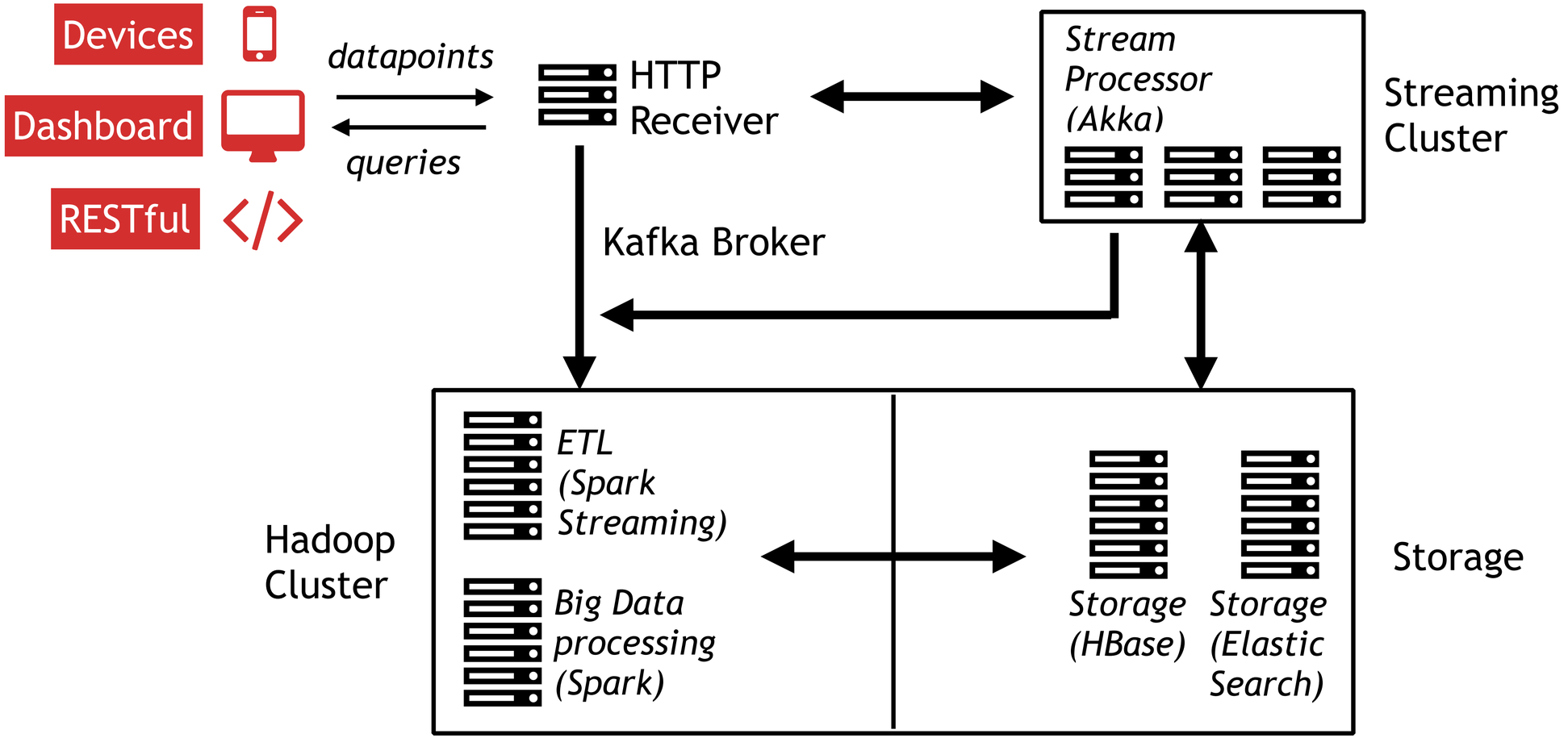
- Telemetry across important dimensions (e.g. platform, game version)
- GameFlow Analysis
- Adaptive Gameplay
- Economy, Monetization, Progression Optimizers
- Game Hive Mind

http://autofire.io
contact@autofire.io

How it Works


Autofire Enabled Device
Autofire Dashboard
Game adapts to player's behavior
Produces
Renders
Live Query Results
Datapoints
Query Results
Streamed over the internet
Live query during Gameplay
Autofire Back-End
Sent to the studio's account

Lambda Architecture



Lambda Architecture

- Streaming Cluster for on-line processing (**Speed** & **Serving** Layer)
- Hadoop Cluster for off-line processing (**Batch** Layer)
- Storage (DBMS) for results
- PubSub (message broker) for inter-operability

⚙ Autofire GameOptix Indexing Processors

Streaming (i.e. one-pass, space-bounded) data types include:

- Monoidal **Scalar** incrementable **primitive** types (Integer, Real)
- Monoidal **Range primitive** types (Integer, Real)
- Monoidal **Distinct counting set** type (HyperLogLog sketch)
- **Frequency** set type (CountMin sketch)
- **TopK / heavy hitters set** type (Stream Summary sketch)
- **Average** types (Decaying, Short-Term, Mean (Monoidal))
- **Quantile** type (t-digest (Monoidal), Frugal sketch)

Autofire GameOptix Clients

Autofire **SDKs**

- **Secure** and **efficient** communication
- Platform **agnostic**
- Optimised for reduced **network traffic** and **battery usage**
- Device **resources friendly**

AI

AI

AI traits

- State / Memory / History
- Reaction to Action
- Clock tick / Spontaneousness / Liveness
- Non-determinism
- Encoded behaviour

Thank You!

http://autofire.io
contact@autofire.io