King

Building & Operating Large-scale Streaming Applications

Gyula Fóra
gyula.fora@king.com

Background

About King

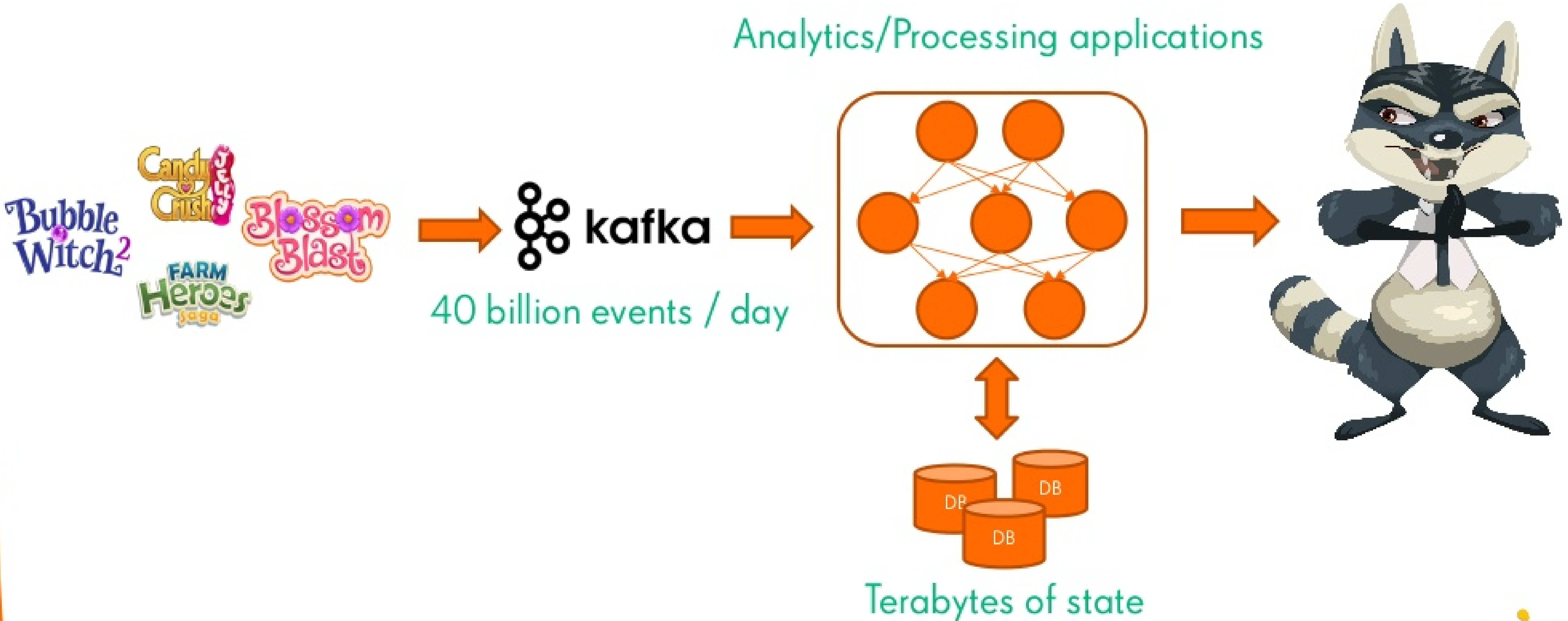
We make awesome mobile games

Hundreds of millions of players

40 billion events per day

And a lot of data…

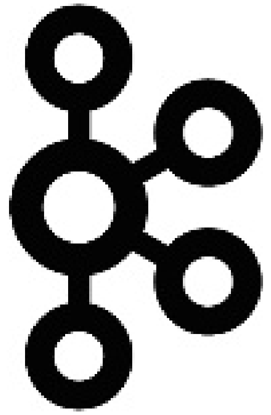
From a streaming perspective…



Stream processing at King

Real-time dashboards

Real-time analytics platform



Kafka Consumers

King Streaming SDK & Libraries

The RBea platform

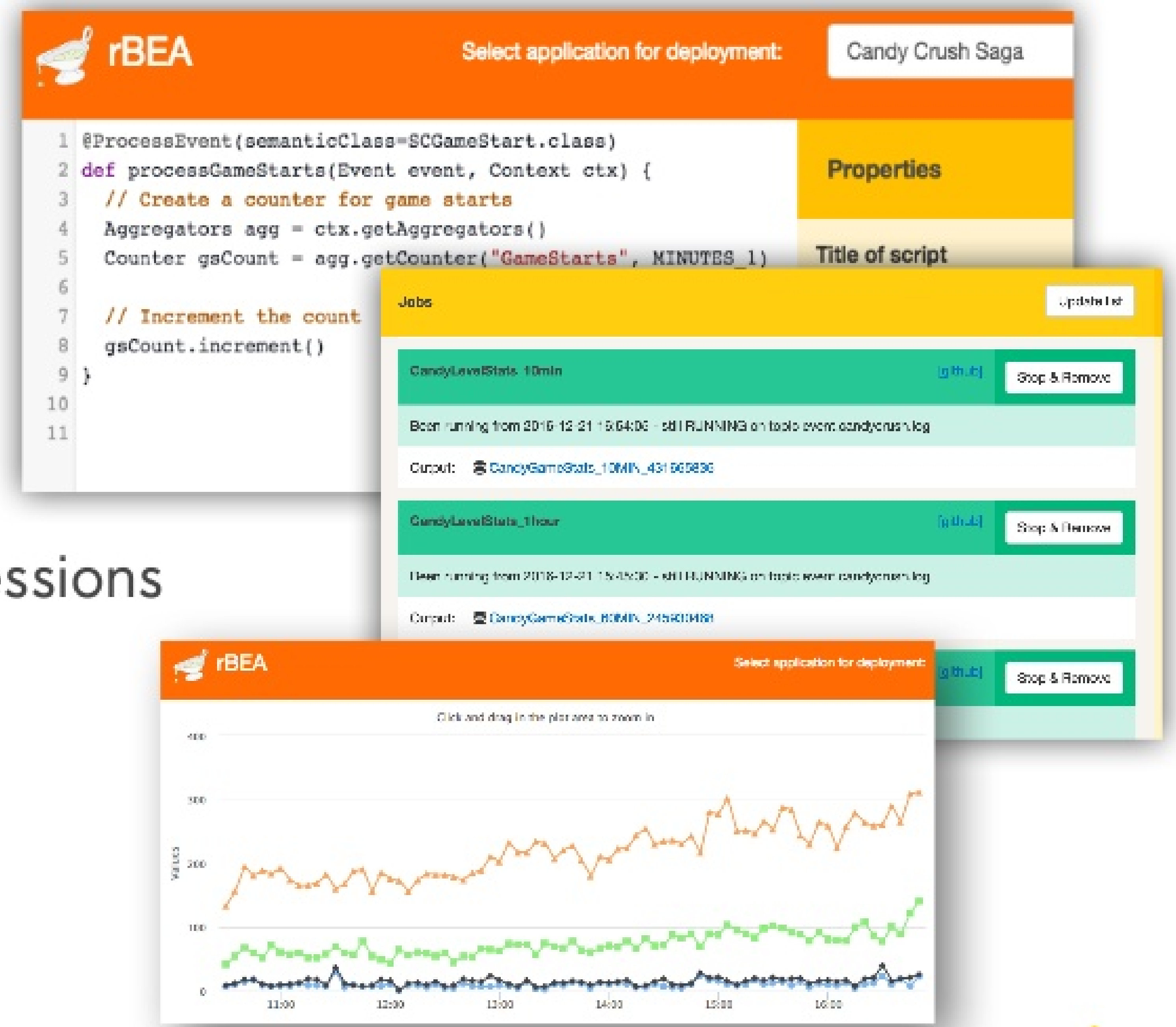
Powered by Apache Flink

Scripting on the live streams

Window aggregates, Timers, Sessions

Complex stateful computations

Scalable + fault tolerant



```
@ProcessEvent(semanticClass=SCGameStart.class)
def processGameStarts(Event event, Context ctx) {
  // Create a counter for game starts
  Aggregators agg = ctx.getAggregators()
  Counter gsCount = agg.getCounter("GameStarts", MINUTES_1)

  // Increment the count
  gsCount.increment()
}
```

CandyLevelStats_10min

Been running from 2016-12-21 15:04:06 - still RUNNING on topic event.candycrush.log

Output: CandyGameStats_10MIN_431965836

CandyLevelStats_1hour

Been running from 2016-12-21 15:45:00 - still RUNNING on topic event.candycrush.log

Output: CandyGameStats_60MIN_245900489

RBea in production

"Stable" since summer 2016

Running for 20+ games

Live/QA environments

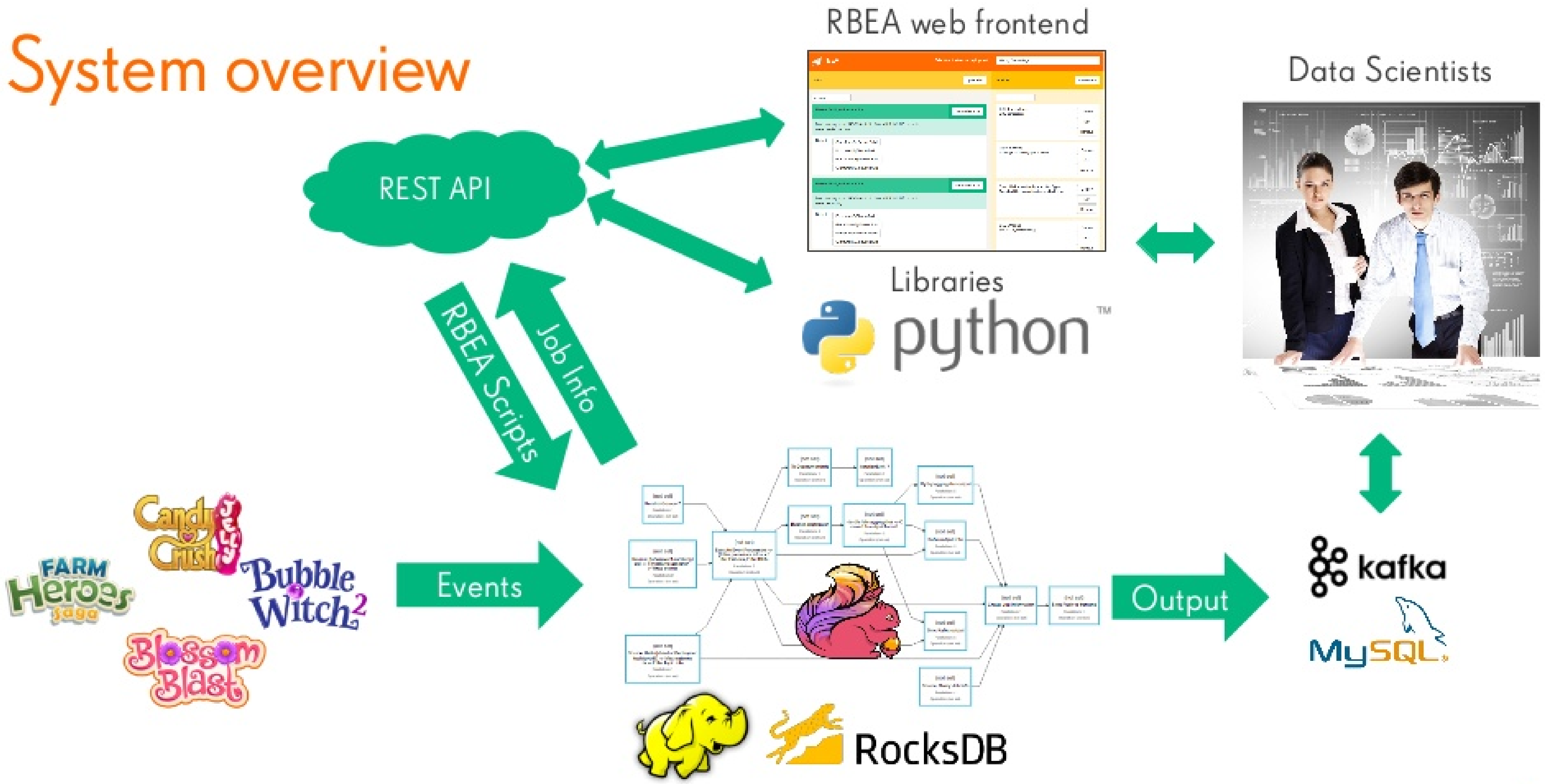
250+ live scripts

5+ TBs of user state (and growing)

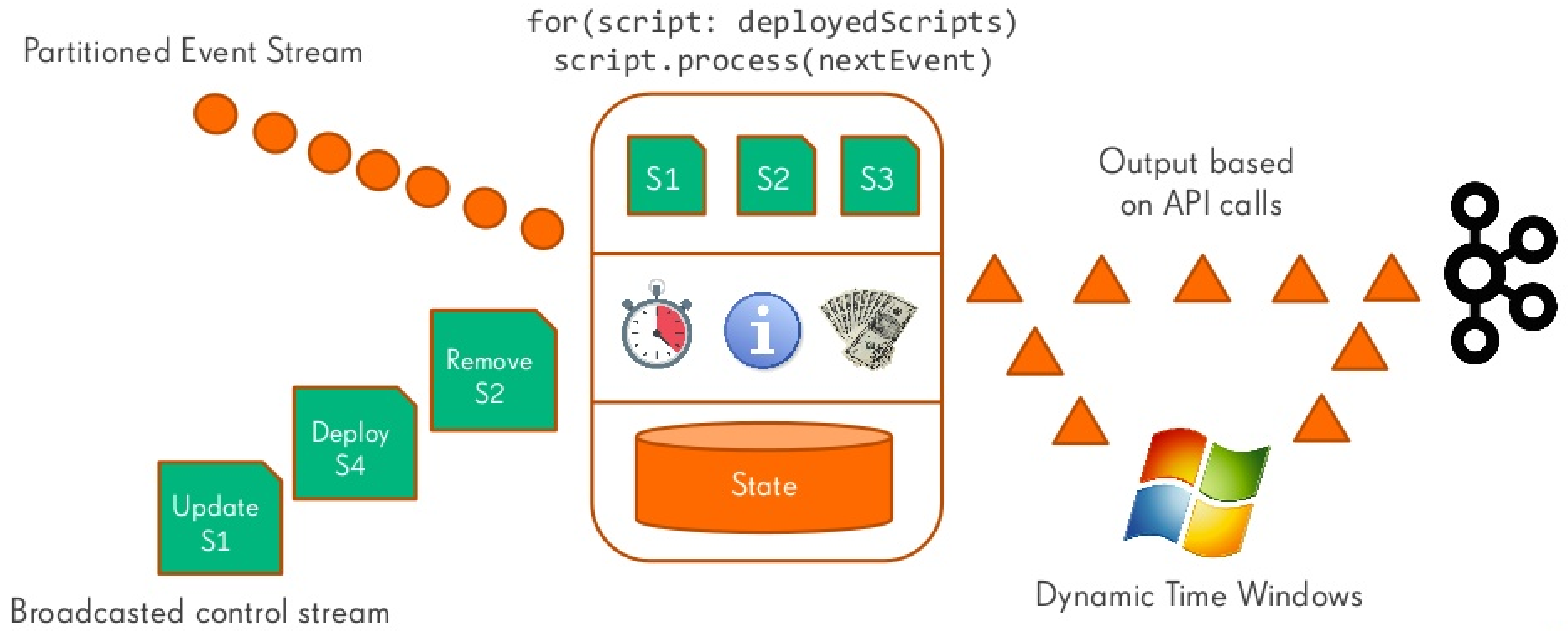
Building a streaming platform

A.K.A the coolest Flink program I have ever written...

System overview



RBea backend architecture

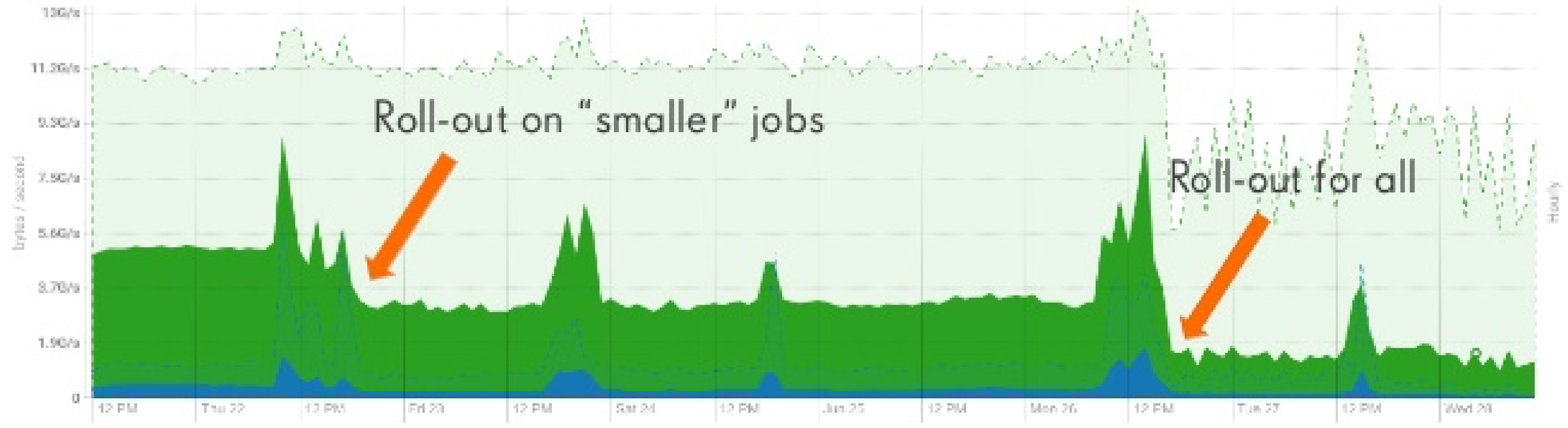

Partitioned Event Stream
for(script: deployedScripts)
script.process(nextEvent)
S1
S2
S3
State
Remove S2
Deploy S4
Update S1
Broadcasted control stream
Output based on API calls
Dynamic Time Windows

User states and state backends

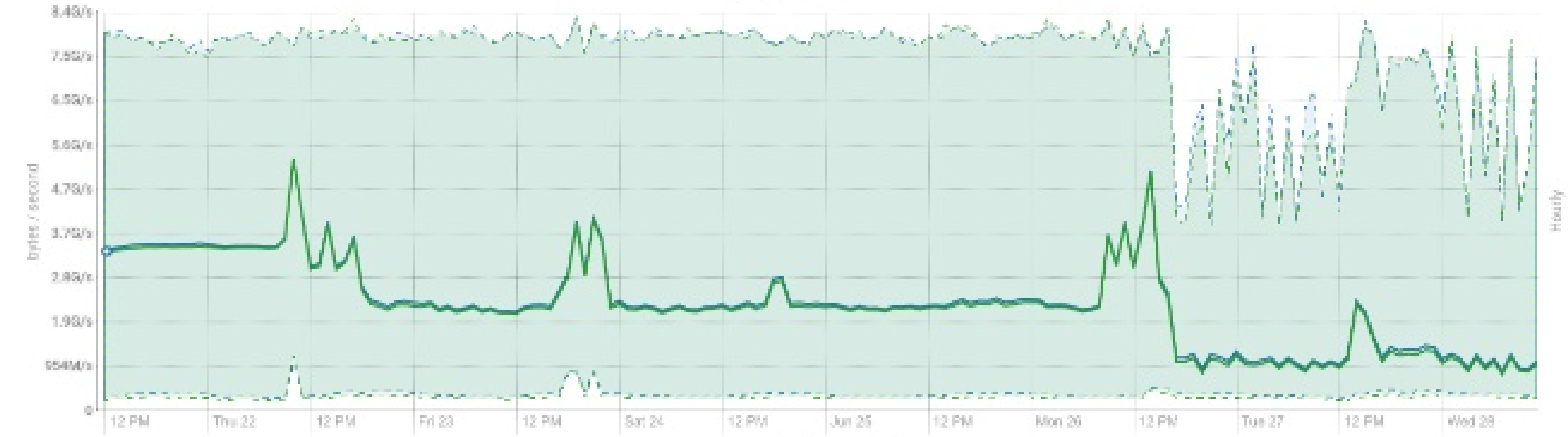
Roughly 1 billion keys

< 10 GB	< 100 GB	< 1-2 TB	< ?? TB
File StateBackend →	Remote DB + cache →	RocksDB (HDFS) + cache →	Incremental RocksDB + cache
- Too much memory	- Poor read performance - Operationally complex	- Poor recovery speed - Large checkpoints hurt the cluster	- Poor recovery speed

"Incremental" improvements



Disk IO



Network traffic

Window aggregates and dashboards

Problems

Connection issues

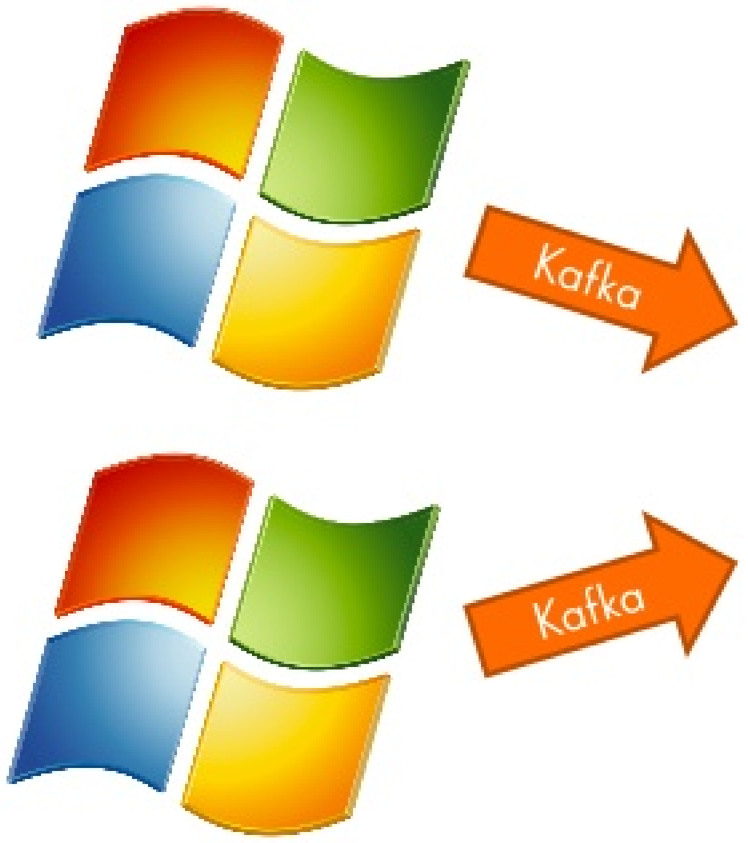
Throughput problems

Duplicated logic

Sneaky bugs

rBEA
Select application for deployment
Click and drag in the plot area to zoom in
400
300
200
100
0

Window aggregates and dashboards



Aggrigato

Read Aggregates
(ts, name, dimensions, value)

Dynamically create schemas

UPSERT new values

Expose restful query interface

REST API

Pipeline testing

```
testJob = RBEATestPipeline
        .startWithDeployment(1000, new TestProcessor1())
        .thenDeploy(6000, new TestProcessor2())
        .thenEvent(3, "1500")
        .thenRemoveProcessor(1000)
        .thenWatermark(800)
        .thenFailAndRestoreJob()
        .thenEvent(2, "1000")
        .thenWatermark(1600)
        .thenDeploy(2000, new ProcWithFailingTimer());
outputs = runTestPipeline(testJob)
// Do some output verification…
```

Running a streaming platform

Now that we are done with the easy part...

Apache Flink infrastructure

	Standalone	hadoop YARN	hadoop YARN
	16 cores 24G RAM HDD	32 cores 380G RAM 2TB SSD 10+TB HDD	32 cores 380G RAM 2TB SSD 10+TB HDD
	x6	x12	x36
Checkpoints	ceph	ceph	hadoop HDFS

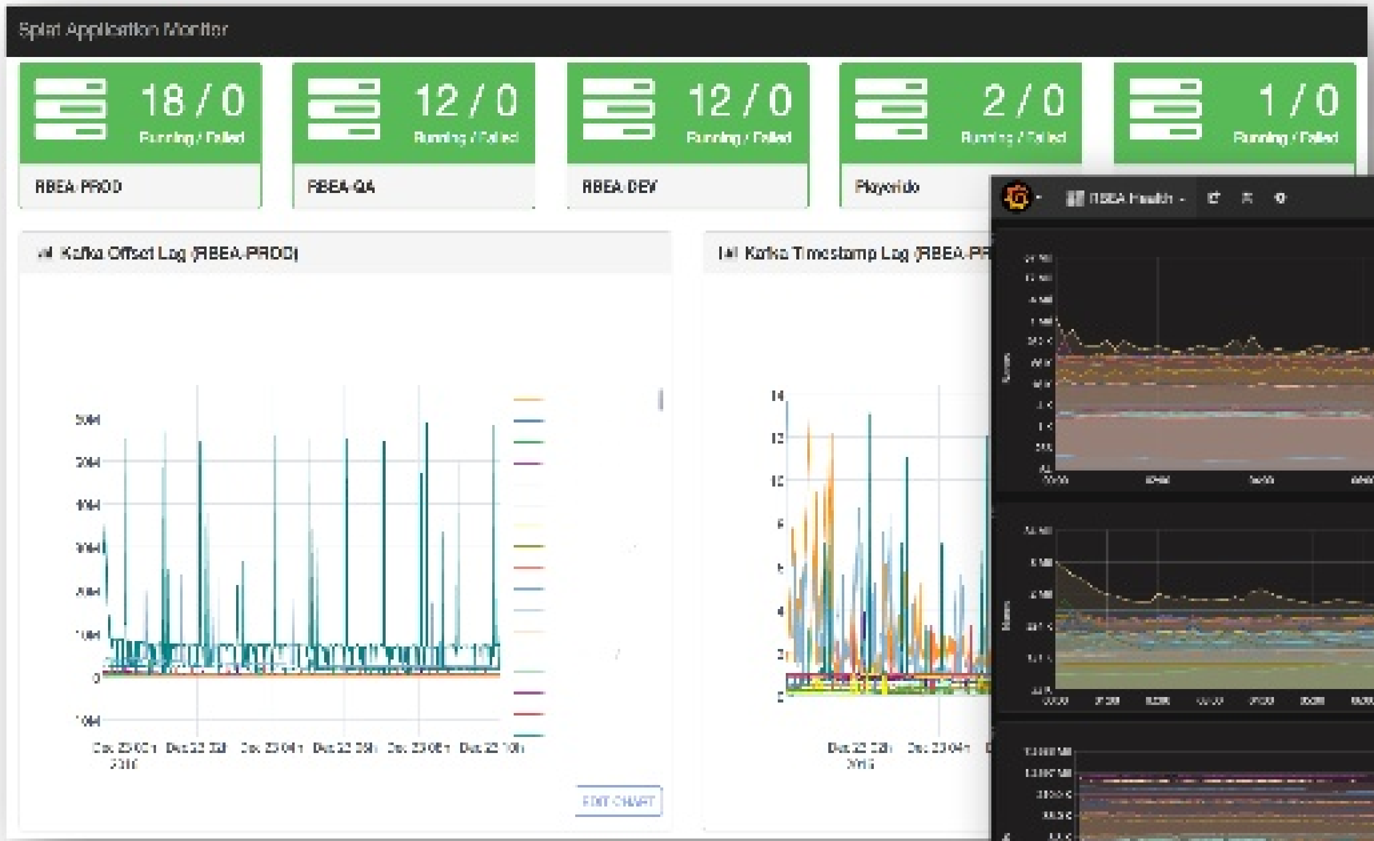
Flink application management

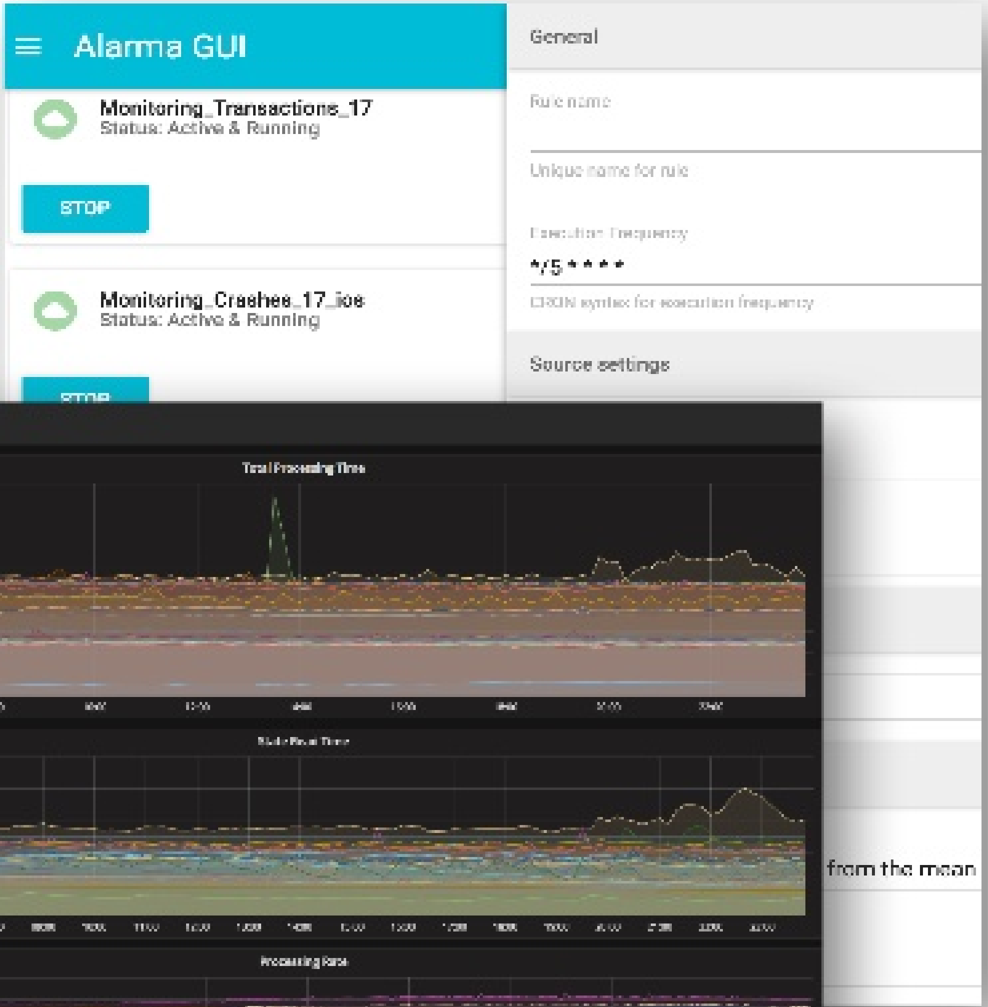
The heavy lifting is done by Flink: Checkpoints, Savepoints, Rescaling

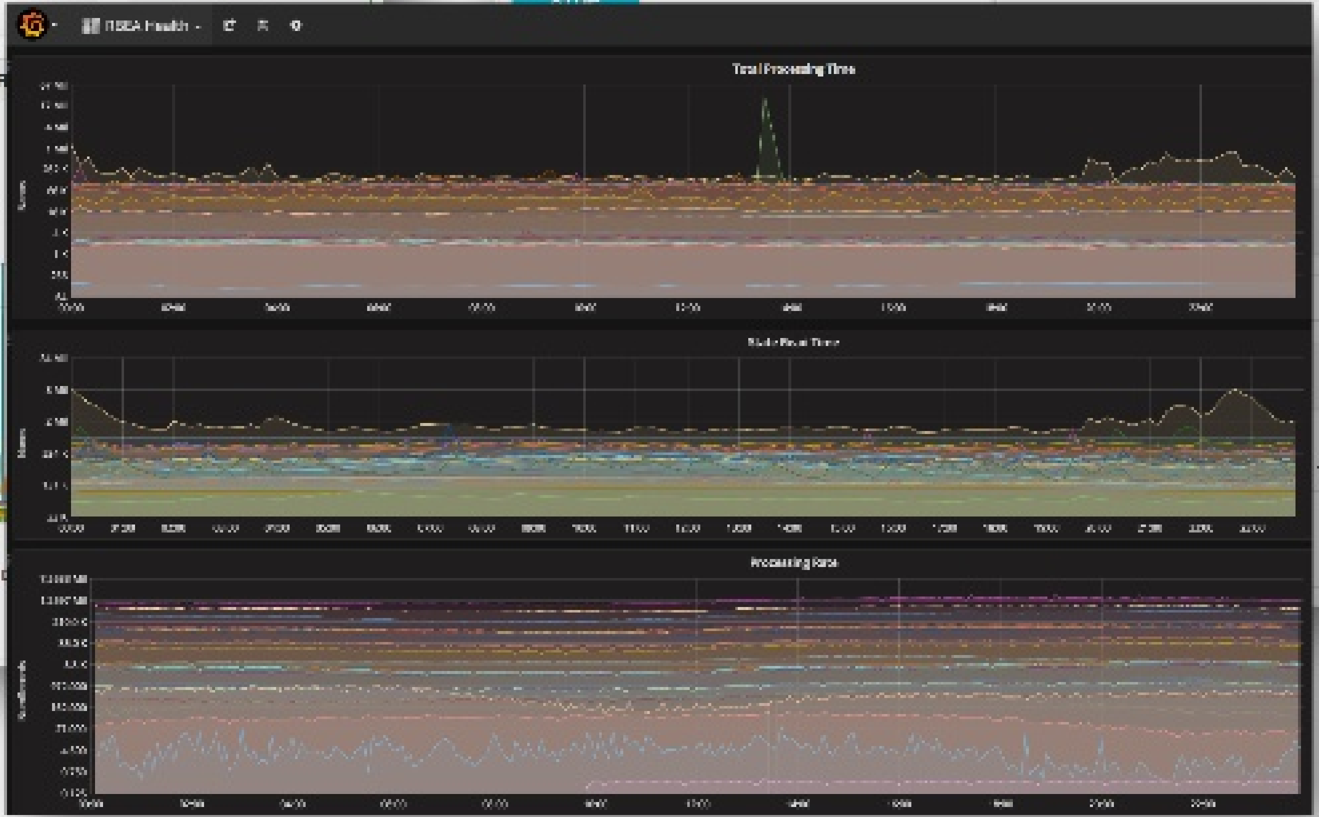
But we have a lot of Python scripts to make it nice

- Lookup and monitor applications (Application IDs)
- Savepoint management (restore from-latest/at, fork)
- Application versioning, automatic fallback to stable
- Continuous deployment (through Jenkins)
- Across several clusters/environments

Metrics, metrics everywhere...







Metrics, metrics everywhere...

Use custom Flink metrics to measure all API interactions

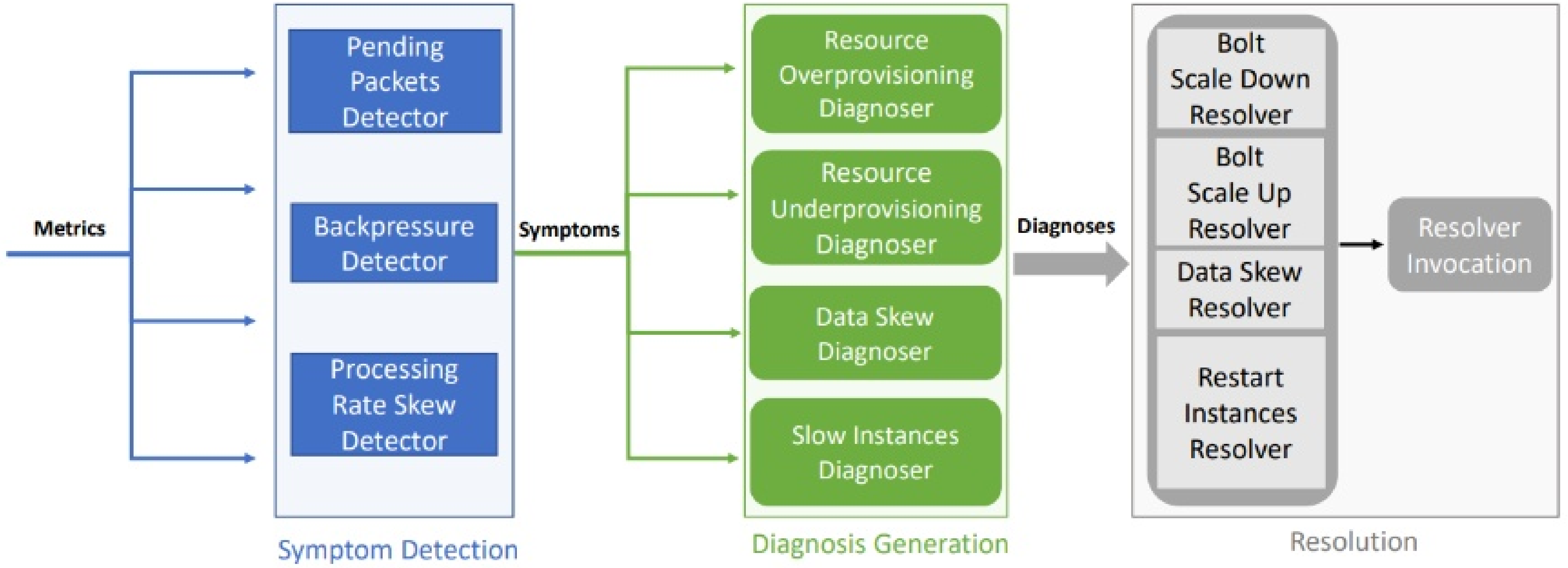
- State read/write times, sizes
- Cache statistics
- Script execution stats

Exponential Moving Average (Gauge) for most metrics

Leverage windowing mechanism for dynamic stats

We also do a lot of CPU profiling ☺

Microsoft Dhalion



https://www.microsoft.com/en-us/research/project/dhalion/
https://github.com/Microsoft/Dhalion

Join the Streaming Platform Team!

We are looking for passionate developers to help us transform the gaming industry!

Talk to us at the conference or search all available roles at
https://jobs.king.com/

Thank you!