SPARK SUMMIT

SMACK STACK AND BEYOND

BUILDING FAST DATA PIPELINES

Jörg Schad, Mesosphere

@dcos @joerg_schad

Jörg Schad

Software Engineer @Mesosphere

@joerg_schad

@joerg.mesosphere


Ancient Times...
MapReduce is crunching Data

Today...

We need to turn faster!

Data Processing

Days *Hours* *Minutes* *Seconds* *Microseconds*

Batch | **Micro-Batch** | **Event Processing**

Reports what has happened using descriptive analytics

Solves problems using predictive and prescriptive analytics

Billing, Chargeback

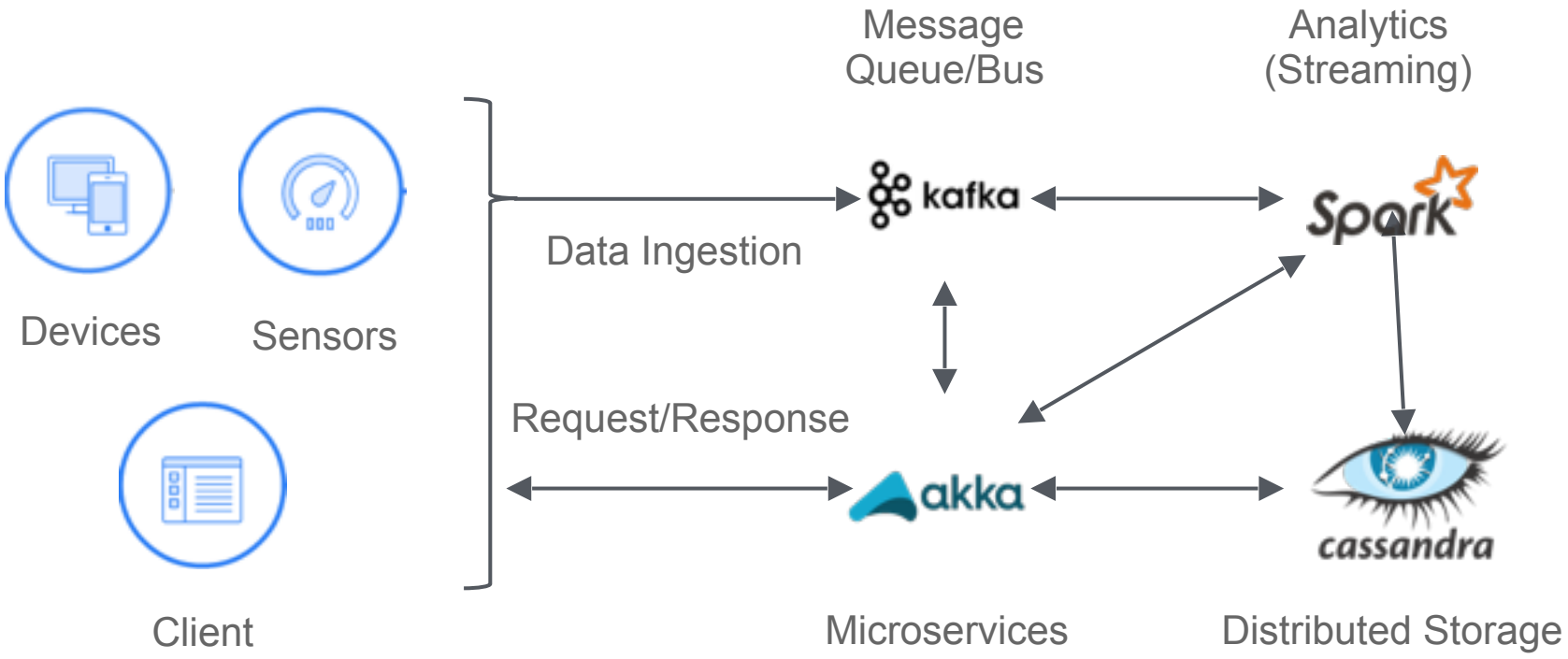
Product recommendations

Real-time Pricing and Routing

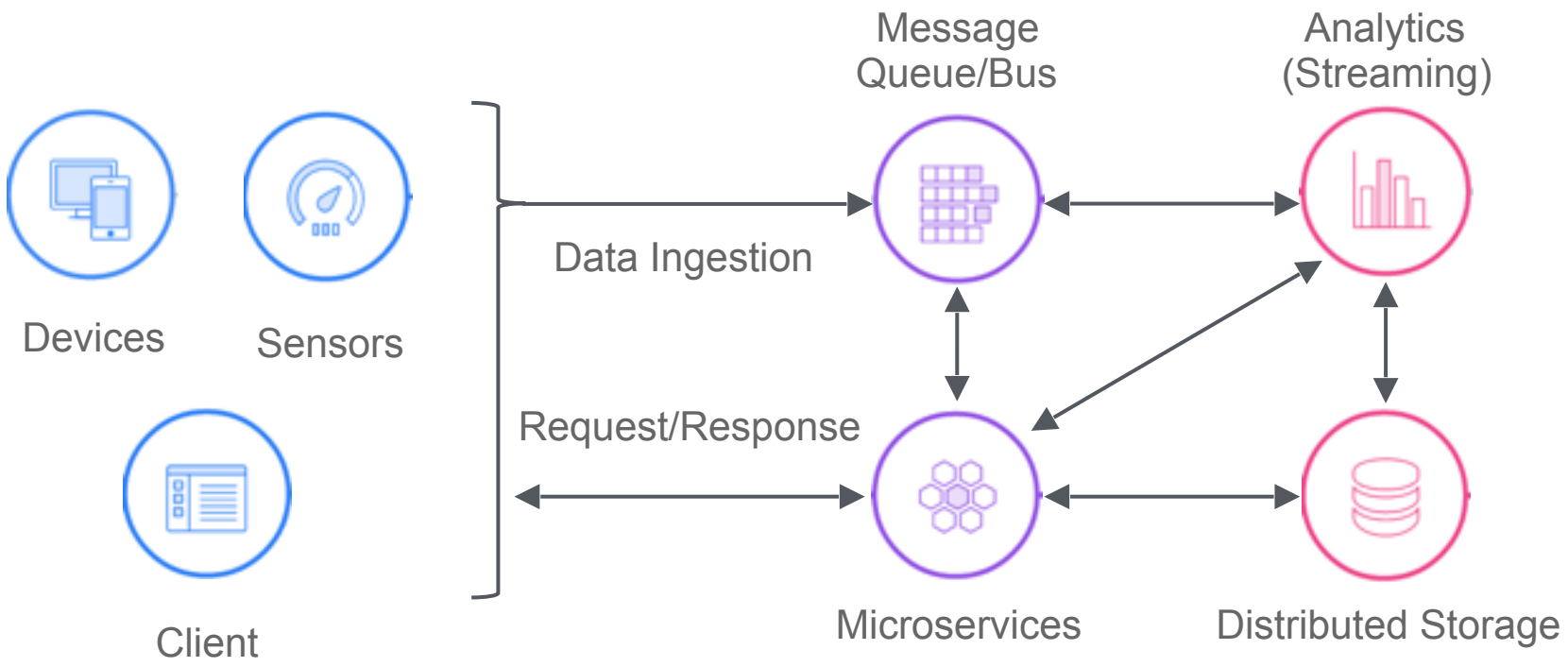
Real-time Advertising

Predictive User Interface

Fast Data Pipelines



Fast Data Pipelines



Fast Data Pipelines

EVENTS

Ubiquitous data streams from connected devices

INGEST

Ingest millions of events per second

ANALYZE

Real-time and batch process data

STORE

Distributed & highly scalable database

ACT

Visualize data and build data driven applications

SMACK Stack

EVENTS

Ubiquitous data streams from connected devices

INGEST

Ingest millions of events per second

ANALYZE

Real-time and batch process data

STORE

Distributed & highly scalable database

ACT

Visualize data and build data driven applications


Sensors


Devices


Clients


Apache Kafka

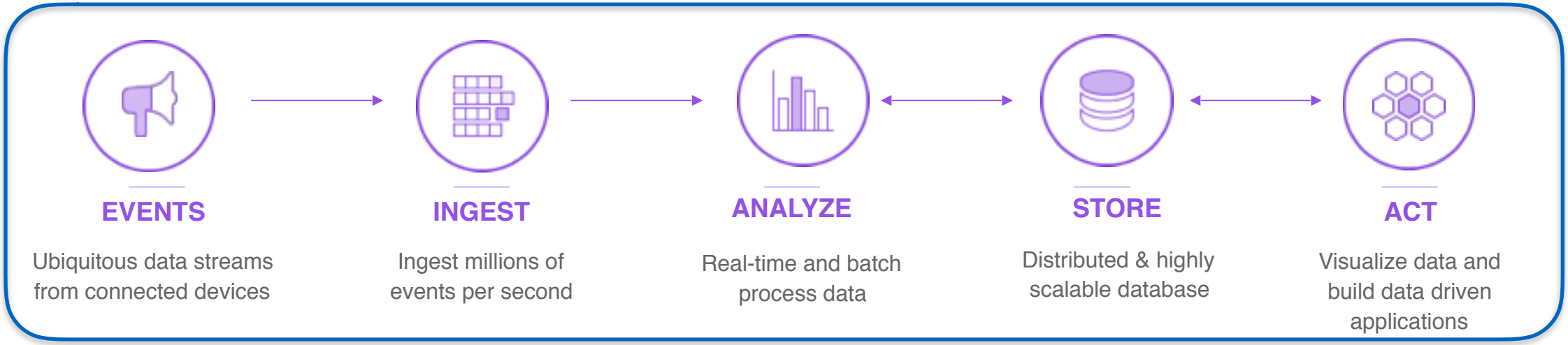

Apache Spark


Apache Cassandra

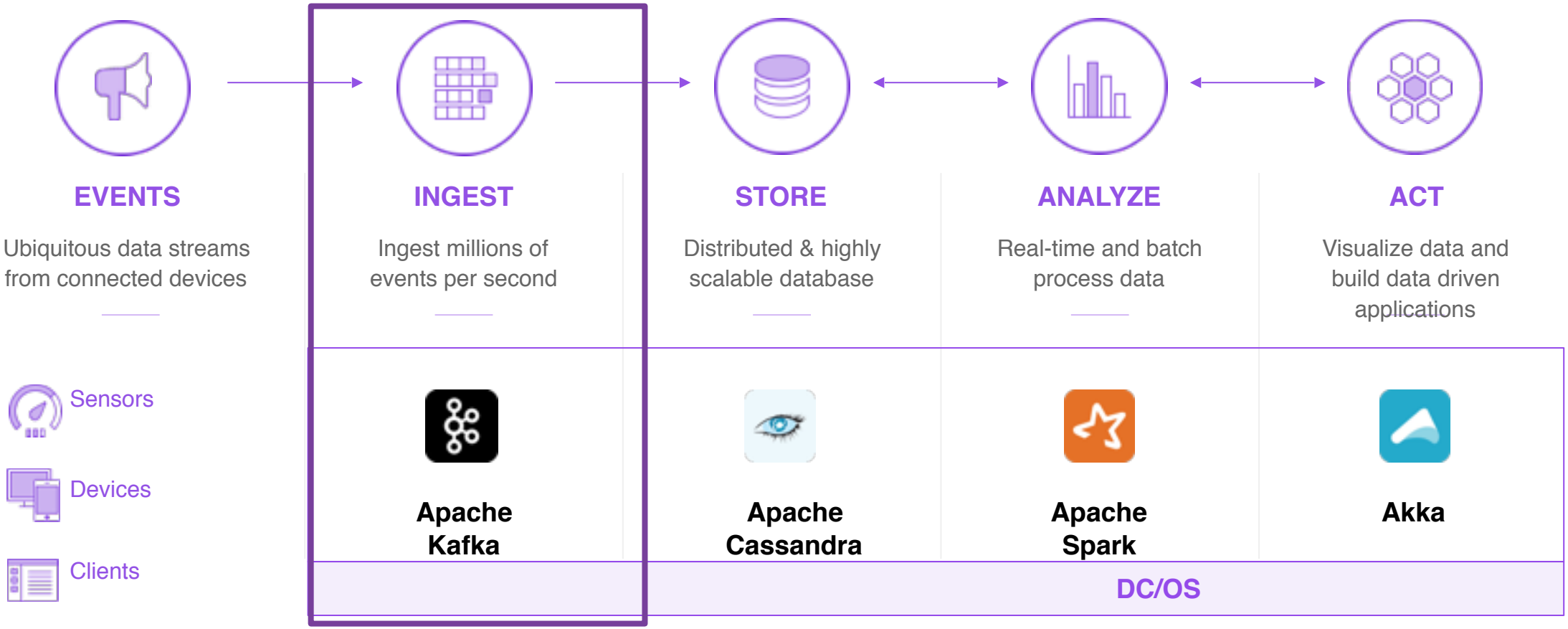

Akka

Apache Mesos

SMACK Stack

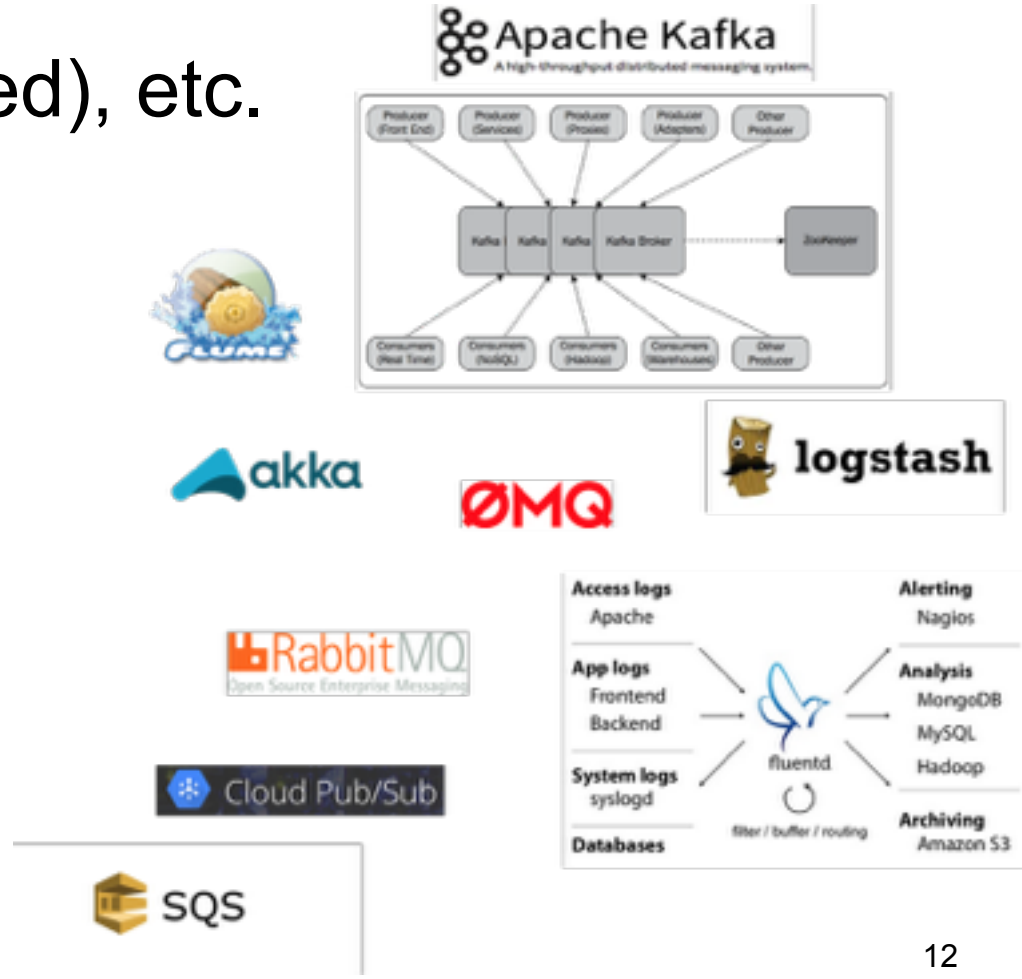

EVENTS
Ubiquitous data streams from connected devices
INGEST
Ingest millions of events per second
ANALYZE
Real-time and batch process data
STORE
Distributed & highly scalable database
ACT
Visualize data and build data driven applications

Message Queues



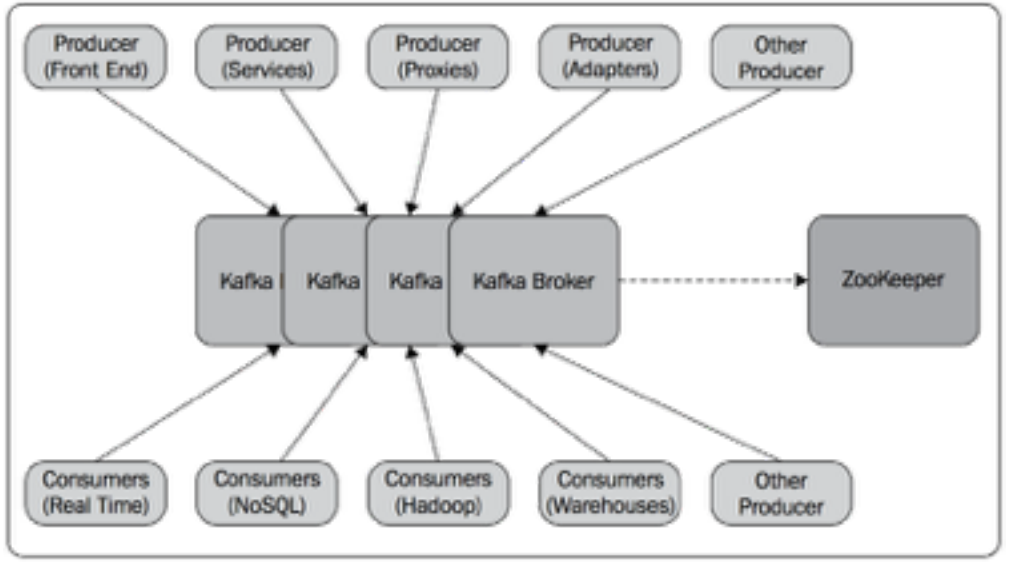
MESSAGE QUEUES

- Apache Kafka
- ØMQ, RabbitMQ, Disque (Redis-based), etc.
- fluentd, Logstash, Flume
- Akka streams
- cloud-only:
  - AWS SQS
  - Google Cloud Pub/Sub

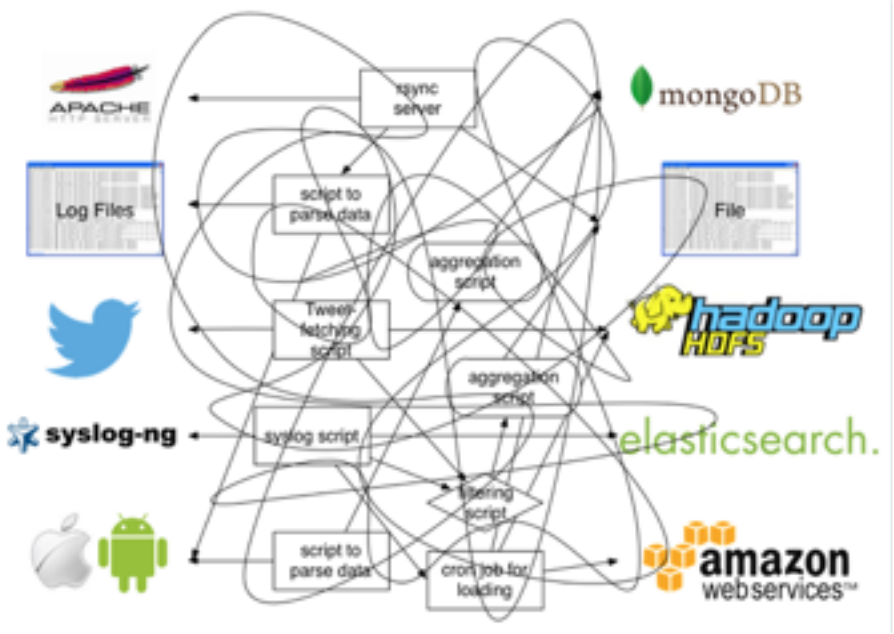


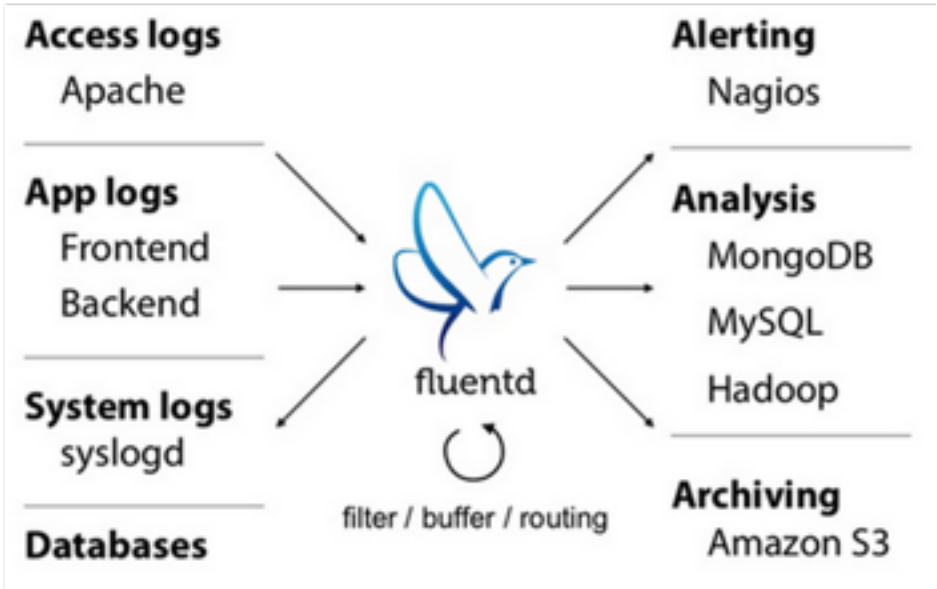
APACHE KAFKA

- High-throughput, distributed, persistent publish-subscribe messaging system
- Originates from LinkedIn
- Typically used as buffer/de-coupling layer in online stream processing



fluentd


APACHE
rsync server
mongoDB
Log Files
script to parse data
File
aggregation script
Tweet fetching script
hadoop HDFS
aggregation script
syslog-ng
syslog script
elasticsearch.
filtering script
script to parse data
cron job for loading
amazon web services


Access logs
Apache
App logs
Frontend
Backend
System logs
syslogd
Databases
fluentd
filter / buffer / routing
Alerting
Nagios
Analysis
MongoDB
MySQL
Hadoop
Archiving
Amazon S3

HOW TO CHOOSE?

- Scalability
- Message Type
  - Log vs …
- Delivery Guarantees/Message durability
- Routing Capabilities
- Failover
- Community
- Mesos Support ;-)

DELIVERY GUARANTEES

At most once—Messages may be lost but are never redelivered.
At least once—Messages are never lost but may be redelivered.
Exactly once—this is what people actually want, each message is delivered once and only once.

Murphy's Law of Distributed Systems:

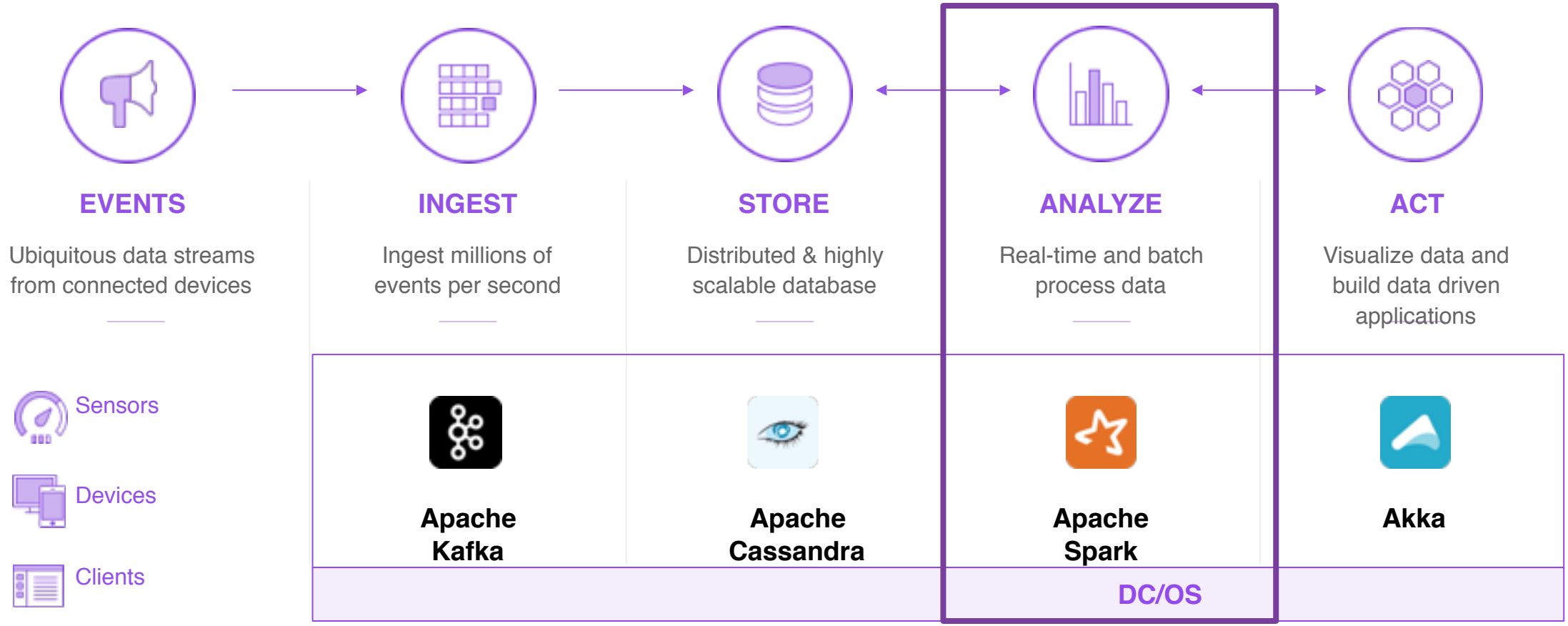
Anything that can go wrong, will go wrong … partially!

Routing

Simple Pipes

Routing

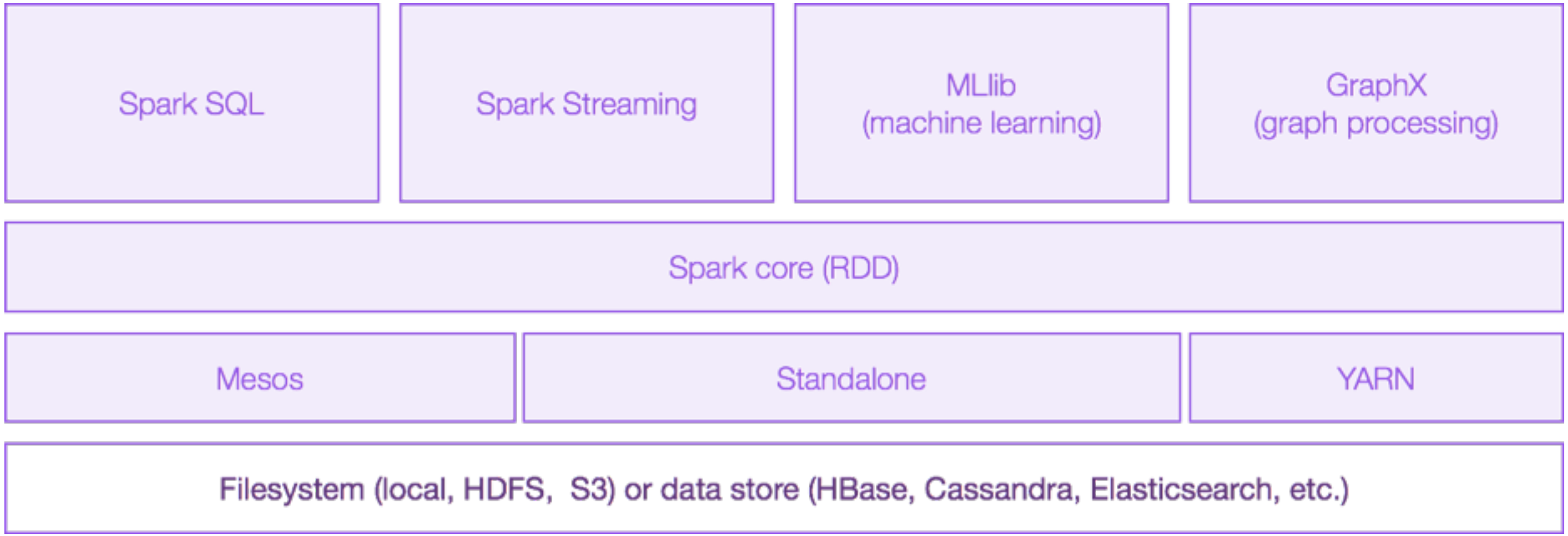
Stream Processing



STREAM PROCESSING

- Apache Storm
- Apache Spark
- Apache Samza
- Apache Flink
- Apache Apex
- Concord
- cloud-only: AWS Kinesis, Google Cloud Dataflow

APACHE SPARK

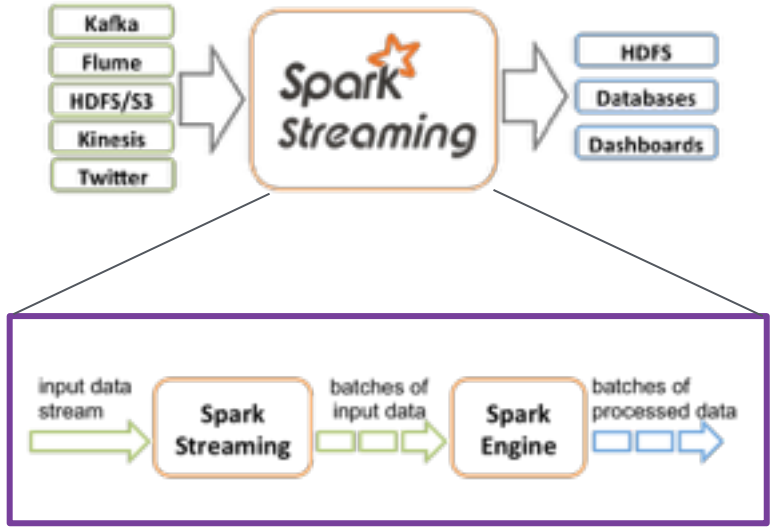


APACHE SPARK (STREAMING 2.0)

Typical Use: distributed, large-scale data processing; micro-batching

Why Spark Streaming?

- Micro-batching creates very low latency, which can be faster
- Well defined role means it fits in well with other pieces of the pipeline

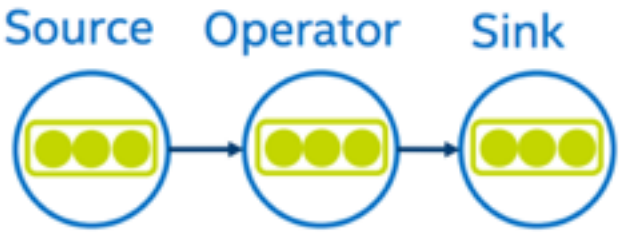


HOW TO CHOOSE?

- Execution Model
  - Native Streaming vs Microbatch
- Fault Tolerance Granularity
  - Per record, per batch
- Delivery Guarantees
- API
  - SQL
  - Spark
- Performance….
  - Realtime ≠ Realtime
- Community
- Mesos Support ;-)

EXECUTION MODEL

Micro-Batching



Spark

Native Streaming

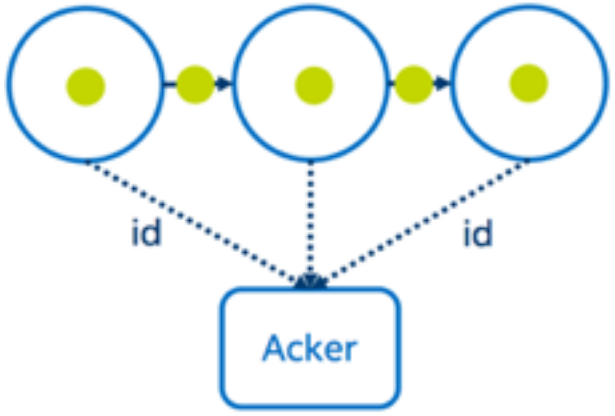
APEX

Flink

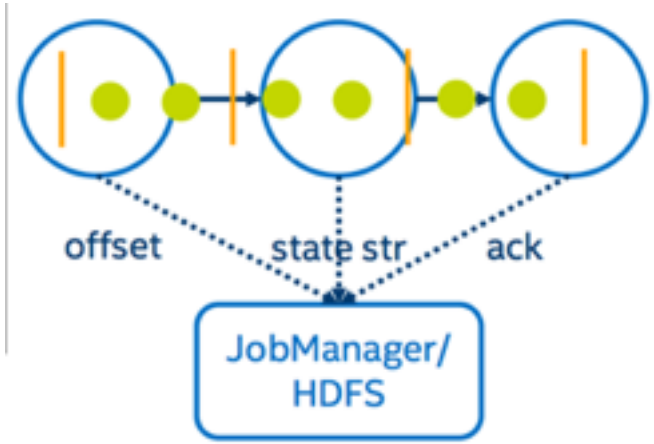
STORM

FAULT TOLERANCE

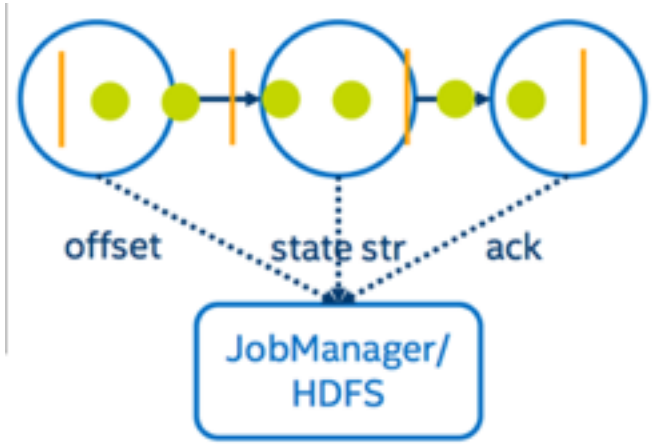
Ack-Per-Record



Checkpoint per "Batch"



Checkpoint per Batch



STORM

Flink

APEX

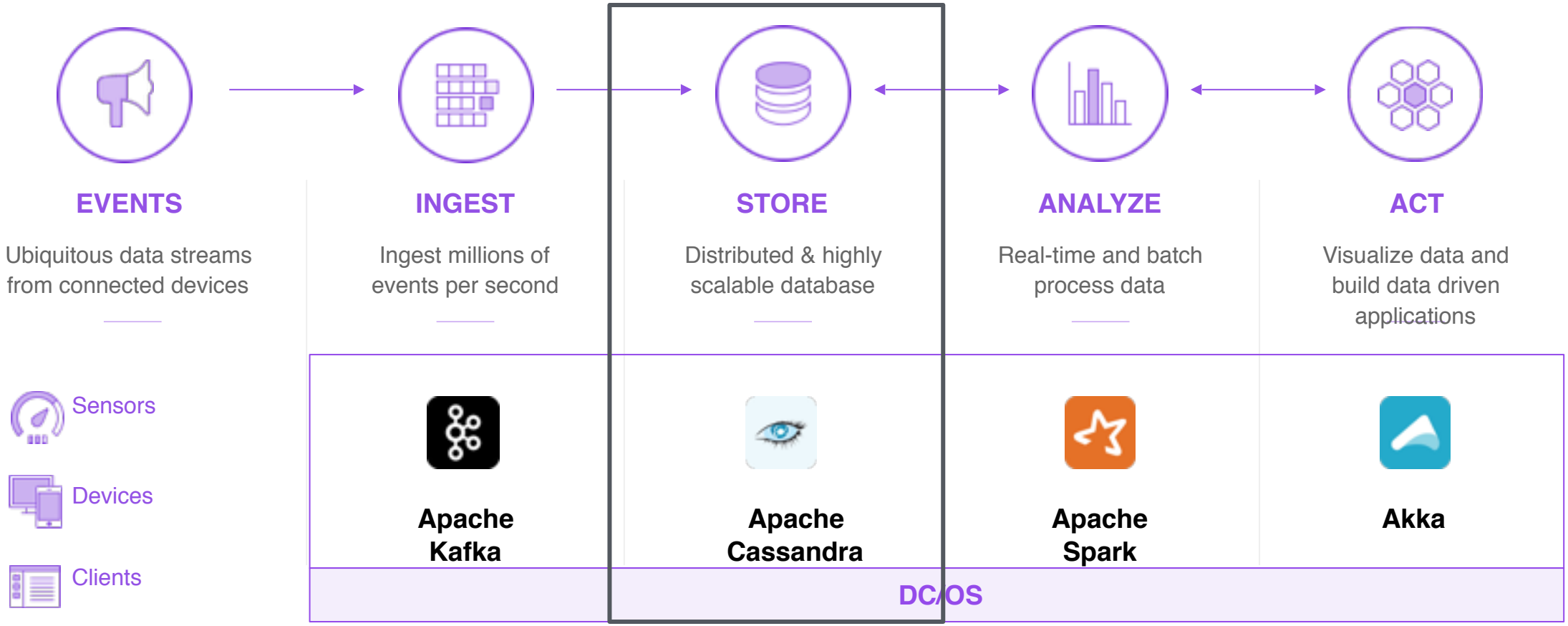
Spark

DELIVERY GUARANTEES

"Exactly once"

At least Once

Storage


EVENTS
Ubiquitous data streams from connected devices
INGEST
Ingest millions of events per second
STORE
Distributed & highly scalable database
ANALYZE
Real-time and batch process data
ACT
Visualize data and build data driven applications
Sensors
Devices
Clients
Apache Kafka
Apache Cassandra
Apache Spark
Akka
DC/OS

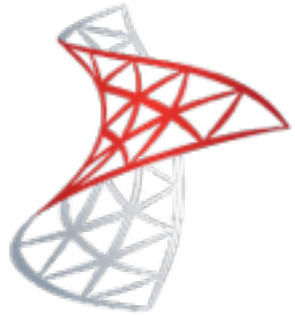
Datastores


neo4j


cassandra


HDFS


ArangoDB




influxdb


MEMSQL


mongoDB


ceph


redis
APACHE
HBASE

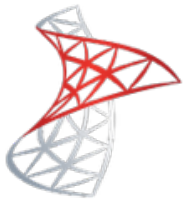

Couchbase


Quobyte

Data Model

Relational

- Schema
- SQL
- Foreign Keys/Joins
- OLTP/ OLAP



MEMSQL

Key-Value

- Simple
- Scalable
- Cache

APACHE HBASE

cassandra

redis

Graph

- Complex relations
- Social Graph
- Recommendation
- Fraud detections

neo4j

ArangoDB

Document

- Schema-Less
- Semi-structured queries
- Product catalogue
- Session data

Couchbase

mongoDB

Time-Series

influxdb

Files

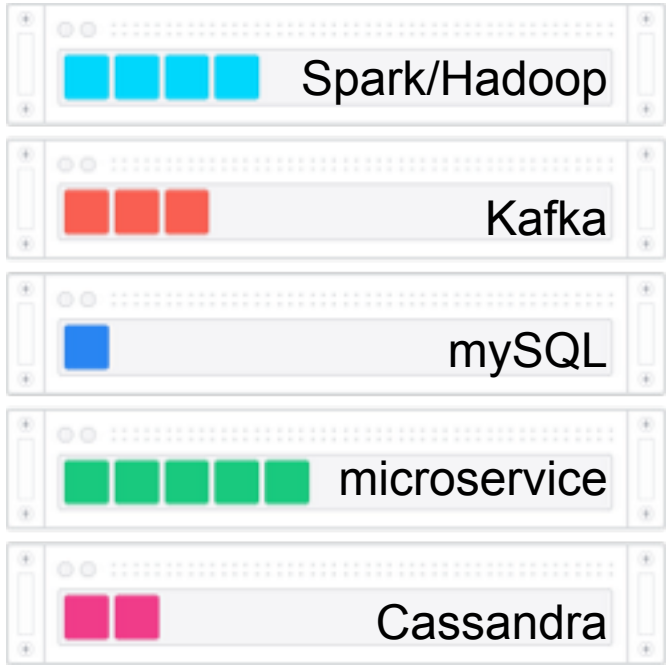
HDFS

Quobyte

ceph


Datacenter
Google
google.com/datacenters

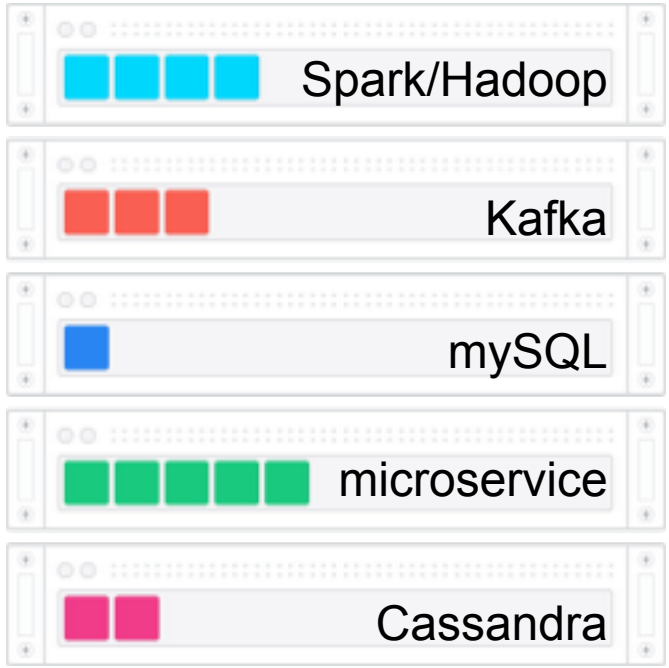
NAIVE APPROACH



Industry Average
12-15% utilization

Typical Datacenter
siloed, over-provisioned servers,
low utilization

MULTIPLEXING OF DATA, SERVICES, USERS, ENVIRONMENTS



Typical Datacenter
siloed, over-provisioned servers, low utilization

Mesos/ DC/OS
automated schedulers, workload multiplexing onto the same machines

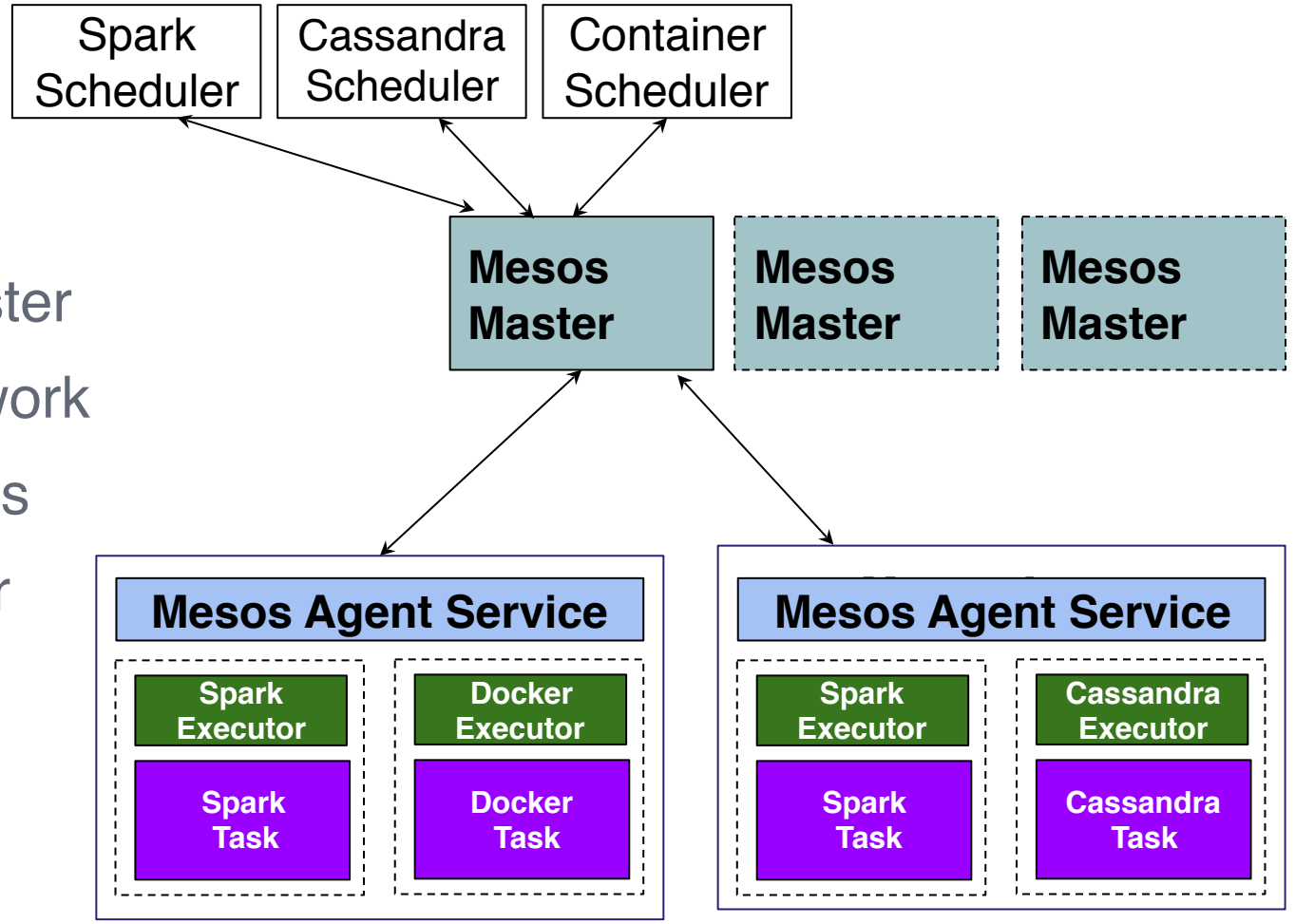
Apache Mesos

- A top-level Apache project
- A cluster resource negotiator
- Scalable to 10,000s of nodes
- Fault-tolerant, battle-tested
- An SDK for distributed apps
- Native Docker support

MESOS: FUNDAMENTAL ARCHITECTURE

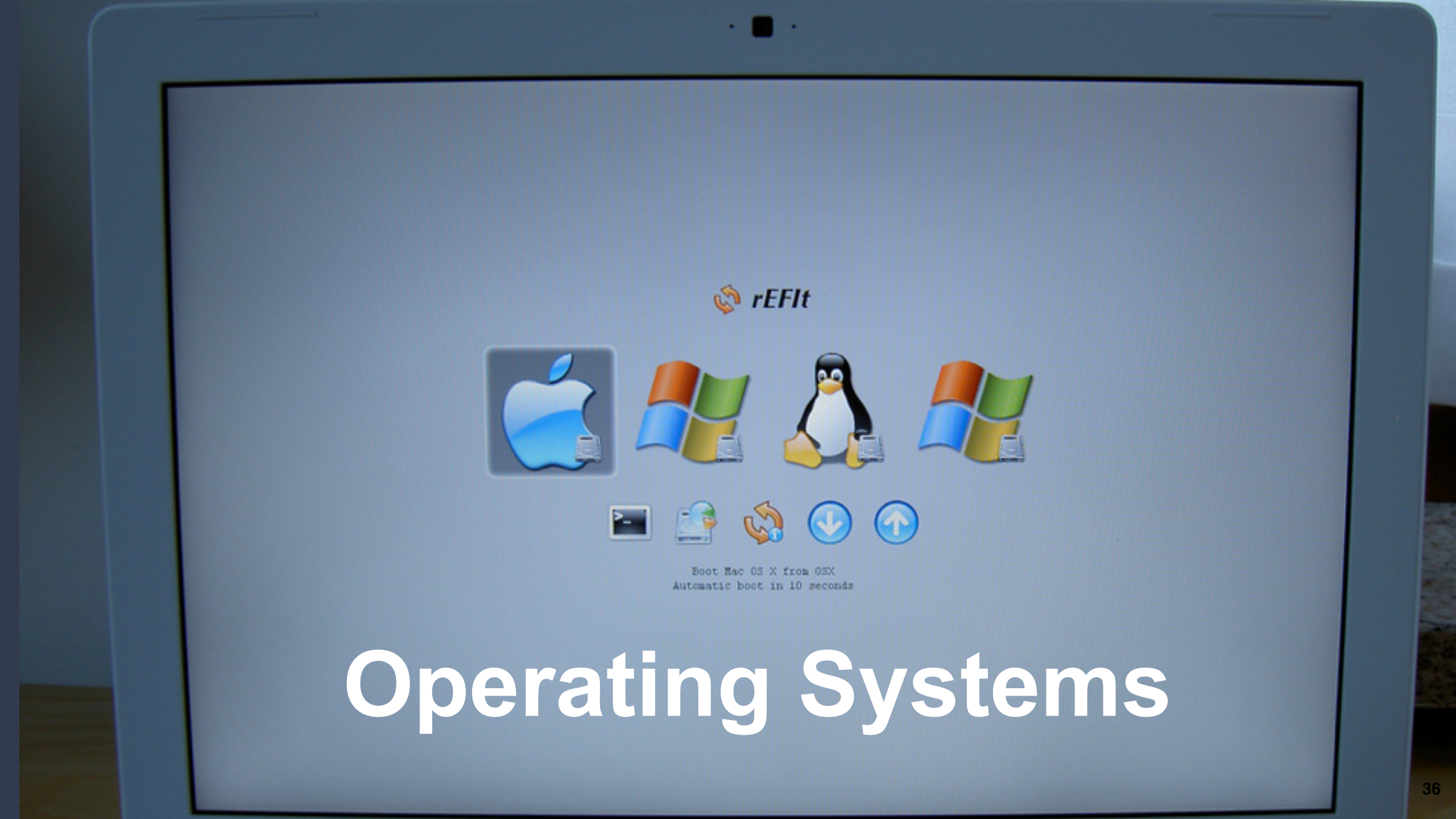
Two-level Scheduling

1. Agents advertise resources to Master
2. Master offers resources to Framework
3. Framework rejects / uses resources
4. Agent reports task status to Master



Challenges

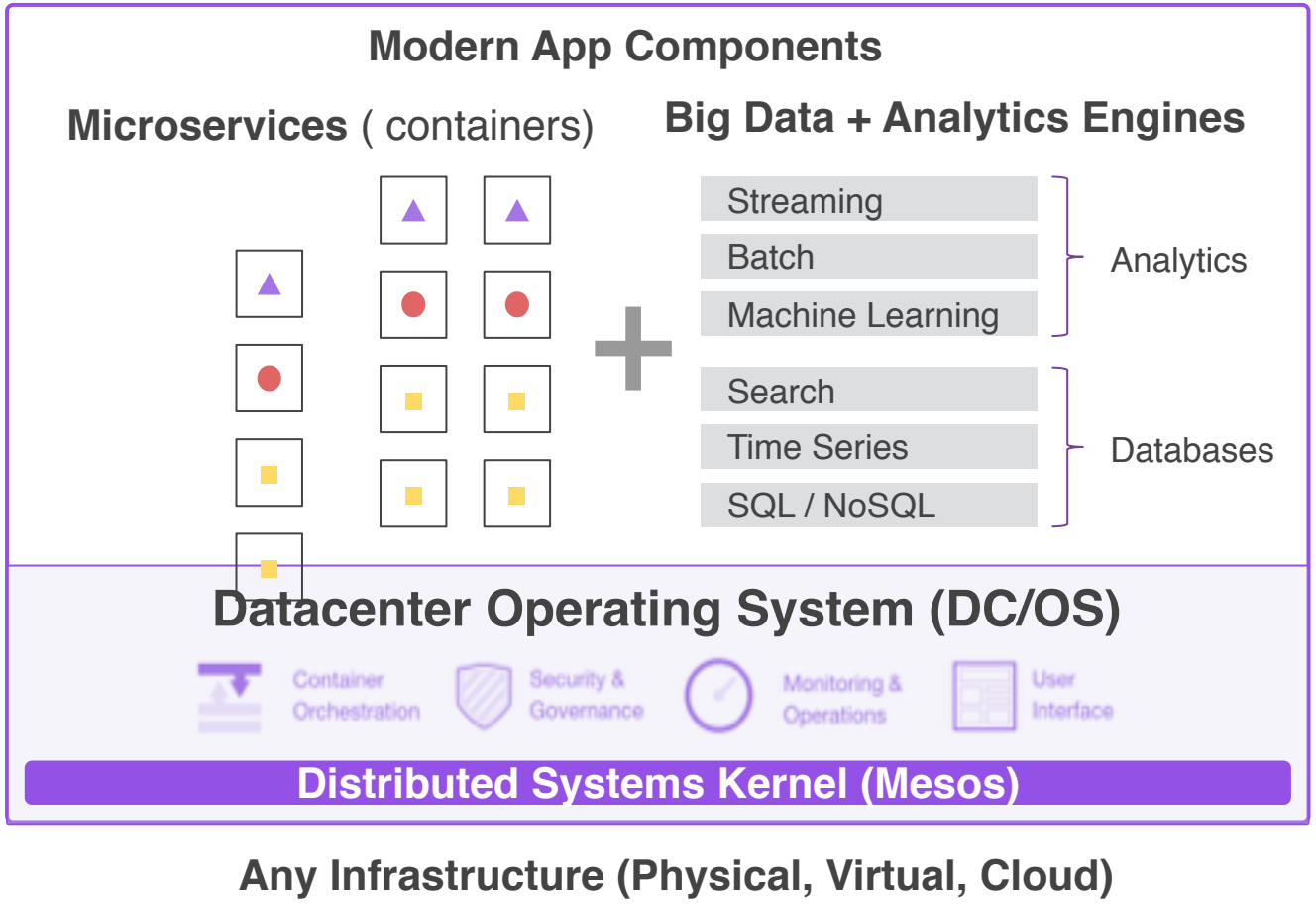
- Mesos is just the kernel
- Need for OS:
    - Scheduler
    - Monitoring
    - Security
    - CLI
    - Package Repository
    - …



Operating Systems

DC/OS

DC/OS



DC/OS

- Container operations & big data operations
- Security, fault tolerance & high availability
- Open Source (ASL2.0)
- Based on Apache Mesos
- Production proven at scale

DC/OS Universe

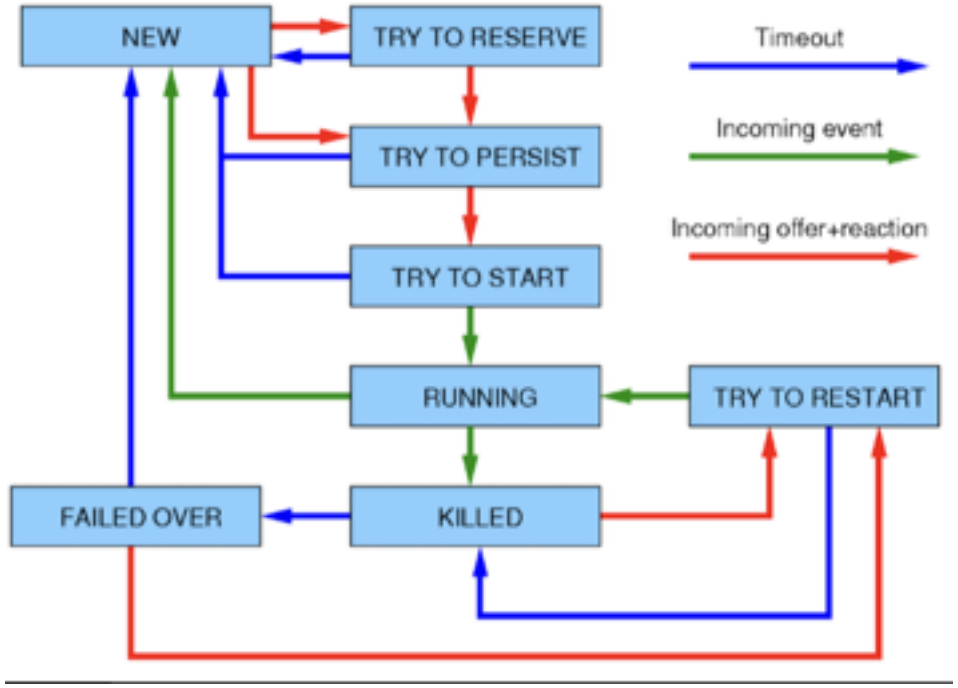
- Datacenter-wide services to power your apps
- Turnkey installation and lifecycle management

Any Infrastructure

- Requires only a modern linux distro (windows coming soon)
- Hybrid Datacenter

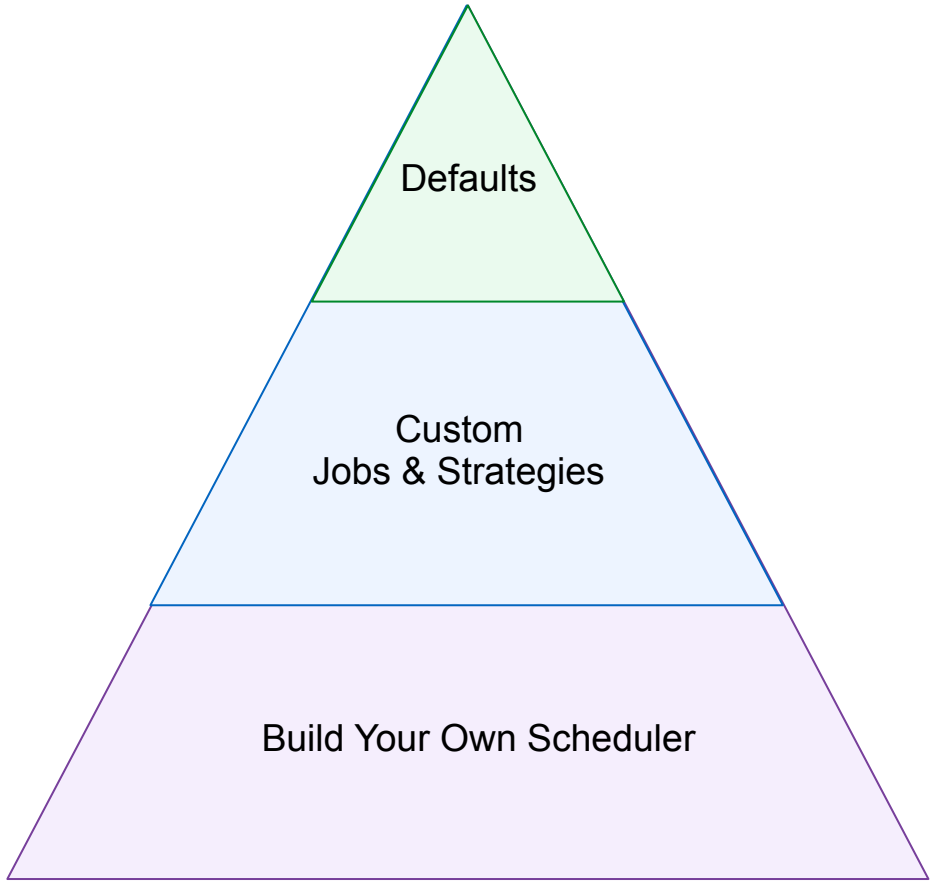
Developing Distributed Services

- Failures (Task, Node, Network,…)
- Zero Downtime Upgrades
- Persistence
- Multiple Frameworks
- Service Discovery
- Metrics
- ….



Operating Distributed Services

Distributed Services: Challenges

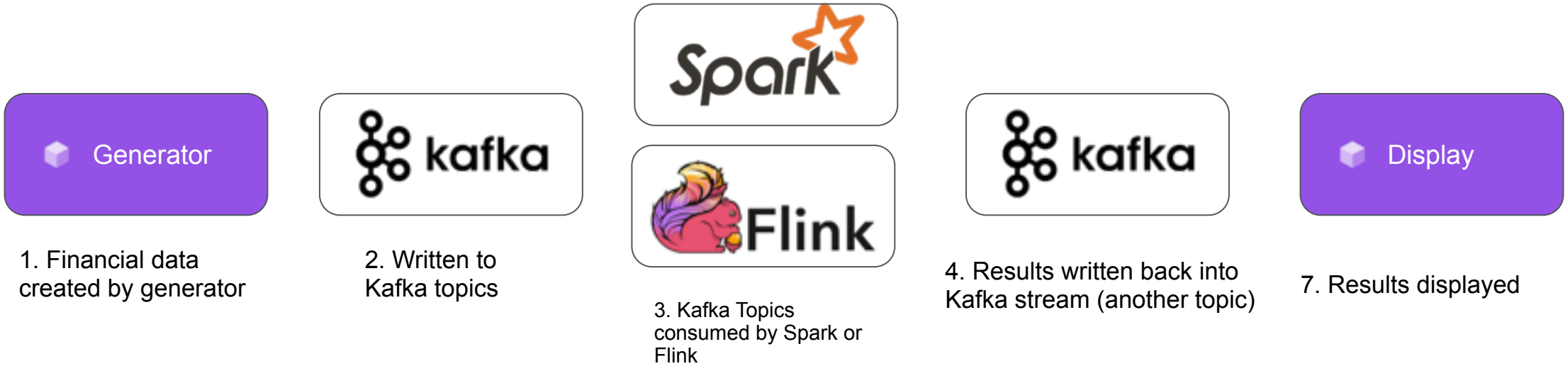


- As simple as Docker Compose
- Don’t need to write any Java code
- Don’t need to be an app expert

- Need to be an app expert
- Need to write a little Java code
- Don’t want to understand DC/OS

- Can’t use the default scheduler
- Need to write a lot of Java code
- Willing to understand DC/OS

Demo Time


Generator
kafka
Spark
Flink
kafka
Display
1. Financial data created by generator
2. Written to Kafka topics
3. Kafka Topics consumed by Spark or Flink
4. Results written back into Kafka stream (another topic)
7. Results displayed

Keep it running!

SERVICE OPERATIONS

- Configuration **Updates** (ex: Scaling, re-configuration)
- Binary **Upgrades**
- Cluster **Maintenance** (ex: Backup, Restore, Restart)
- **Monitor** progress of operations
- **Debug** any runtime blockages

SPARK SUMMIT

Questions?

dcos.io
Demo

@dcos @joerg_schad