



Reactive Microservices - New Addition to SOA

April 26, 2017



AGENDA

1. **SOA Recalled**
2. **Hi !!! Microservices**
3. **Welcome !!! Reactive Microservices**
4. **Conclude**



SOA Principles

Reusability

Contract

Loose
Coupling

Abstraction

Autonomy

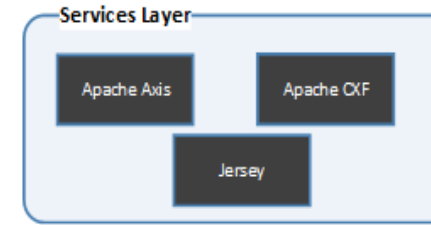
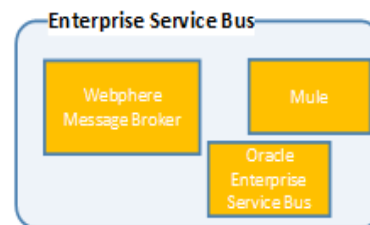
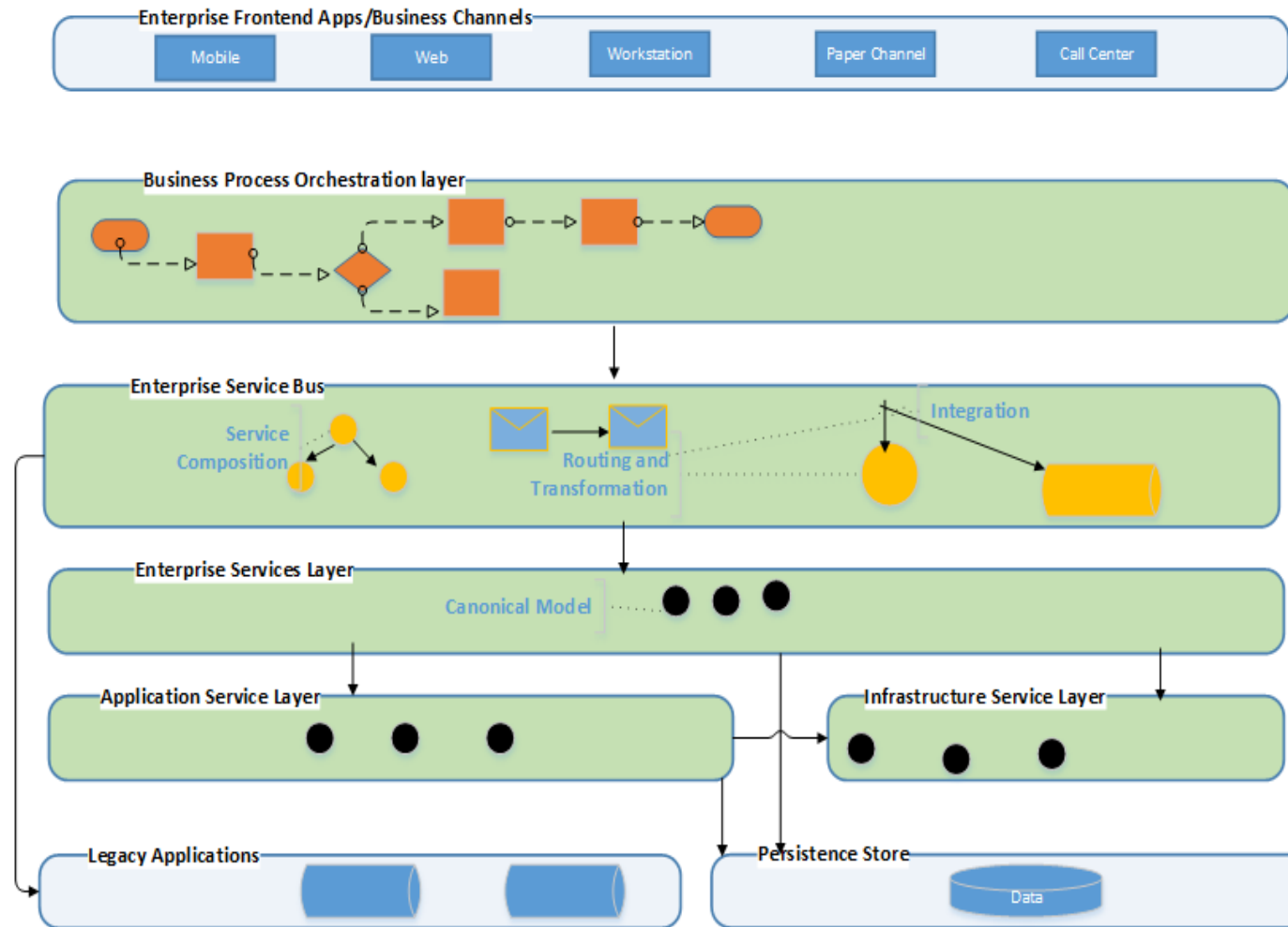
Composability

Statelessness

Discoverability



Typical SOA Architecture



Micro Services Design Considerations



Microservices comparison with SOA

Comparison Basis	Microservices	SOA
Service Taxonomy	Functional and Infrastructure Services	Business, Enterprise Application and Infrastructure Services
Service Ownership and co-ordination	Cross functional teams organized around capabilities, Requires less co-ordination	Concept of Service Owners, Teams organized around Service Taxonomy and requires more coordination
Service Granularity	Fine Grained	Coarse Grained
Component Sharing	Share-as-little-as-possible	Share-as-much-as-possible
Middleware vs API Layer	API layer	Messaging Middleware
Orchestration and Choreography	Choreography	Orchestration and Choreography
Technology	REST	SOAP, REST,AQMP, WS*, JMS...



Benefits of Using Microservices

Ease of
Deployment

Resilience

Scaling

Technology
Heterogeneity

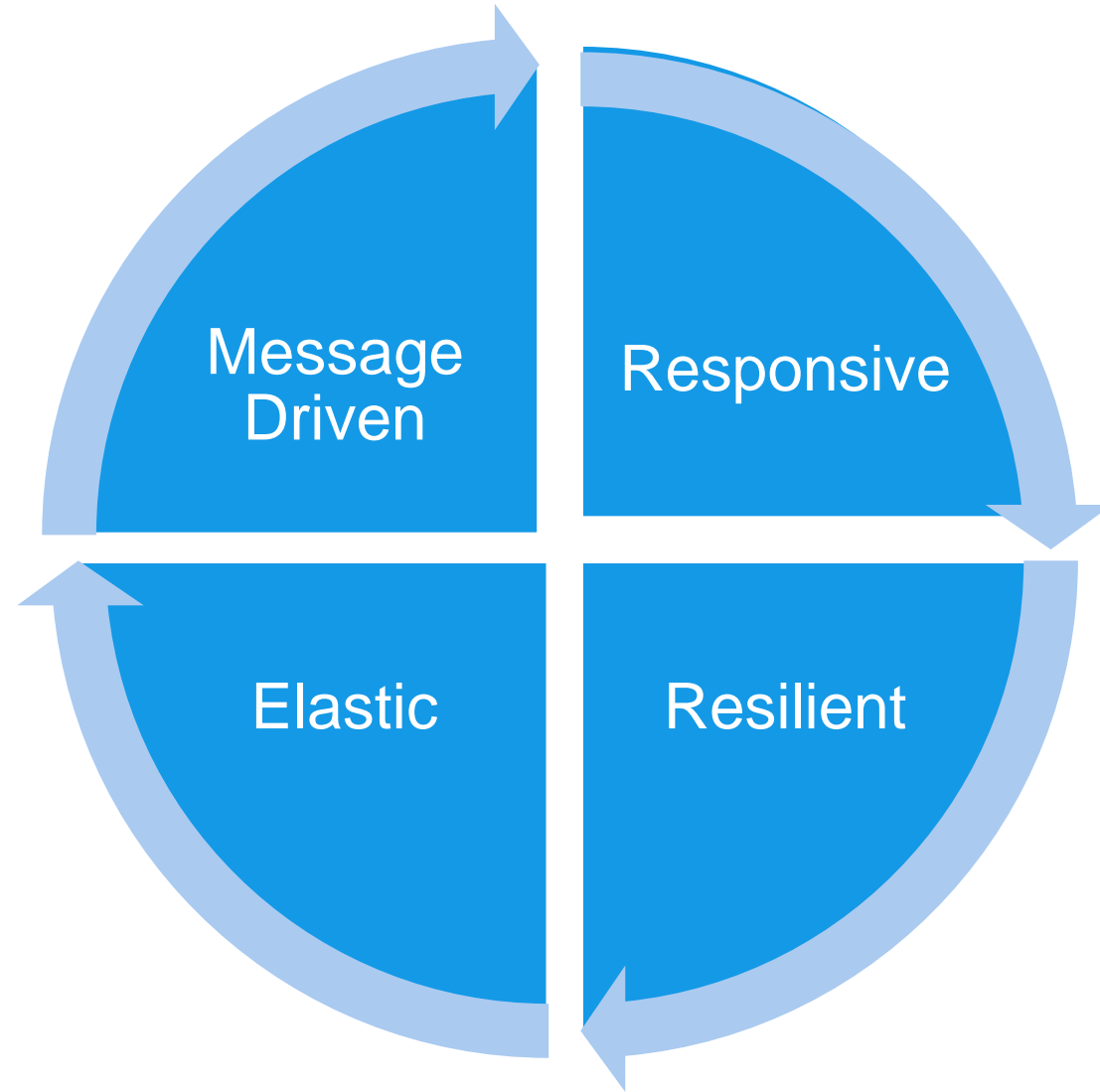
Compos-
ability

Optimized for
Replace-
ability

Organizational
Alignment



Reactive Manifesto



Reactive Programming Tools

Functional Programming

- First class citizen
- Immutable
- Referential Transparency
- Pass value as parameter
- Allows Parallel execution

Communicating Sequential Processes

- Mathematical theory of concurrency
- Free from common pitfalls
- JCSP
- Support Asyn, Non-Blocking, Message passing
- No scale outwards or failure

Event Loops

- Event Driven Programming
- Callbacks
- Asyn and Non-Blocking
- No message passing, scale outwards
- Javascript in NodeJs
- Vert.x supports distributed event bus

Future and Promises

- Read-only handle - Future
- Write-Handle – Promise
- Completion Callback
- Supports Async, Non-Blocking

Reactive Extensions

- Originated .Net
- Observer and Iterable Pattern
- Standard operations for transformation
- Supports Asyn and Non-Blocking
- No Outward scaling

Actor Model

- Computational Entity
- Single Responsibility
- Share nothing
- Supports Async, Non-Blocking
- Supports Message Passing
- Scale Up and Outwards
- Built in failure mechanism



Reactive Programming (Application Services) APIs and Frameworks

JAVA 8

- Lambda Expressions
- Futures and CompletableFuture
- Stream APIs
- ExecutorService

RxJava

- Observer pattern
- Asynchronous
- Operators and Transformation
- Backpressure

Akka

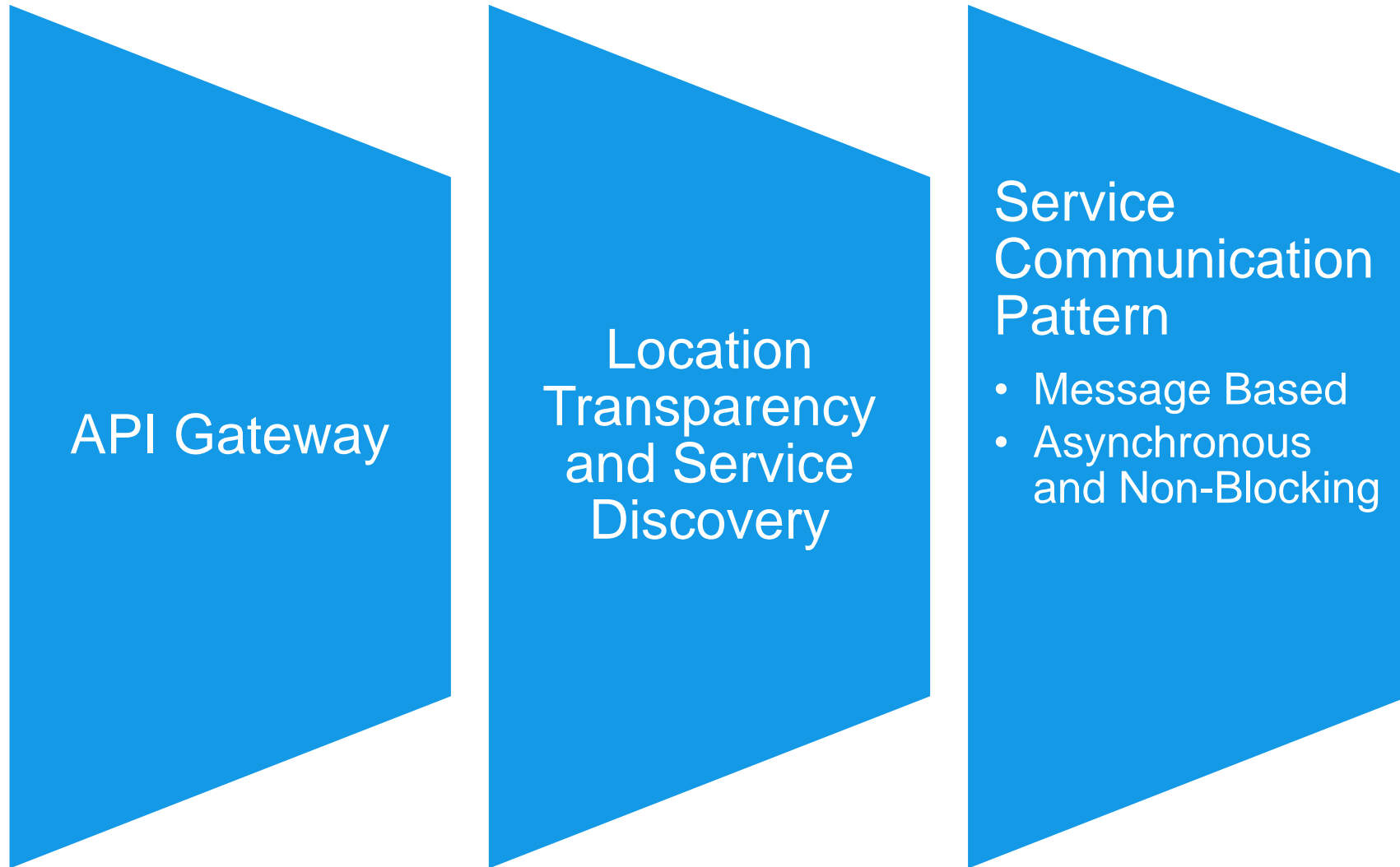
- Actor Model
- Location Transparency
- Asynchronous
- Supervision
- Remoting
- BackPressure

Others

- Vert.x
- Spring 5
- Play
- Lagom



Reactive System Design



Reactive Patterns

Fault Tolerance Pattern

- Circuit Breaker pattern

Message Flow Patterns

- Saga pattern

Flow control patterns

- Pull-Push pattern (Back Pressure)

State management and persistence patterns

- Event Sourcing and CQRS pattern

Replication

- Active-Active



Reactive System (Application Infrastructure and Persistence) APIs and Frameworks

Distributed Streaming

- STORM
- Others
 - SPARK
 - KAFKA
 - FLINK

Distributed Messaging

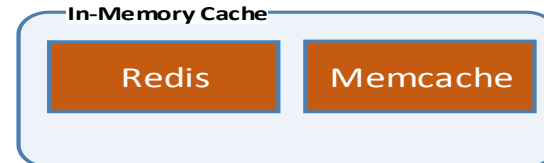
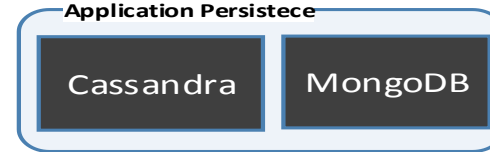
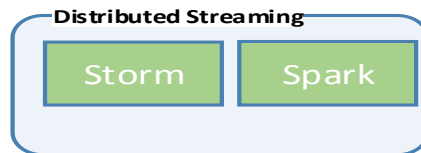
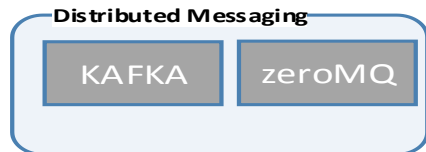
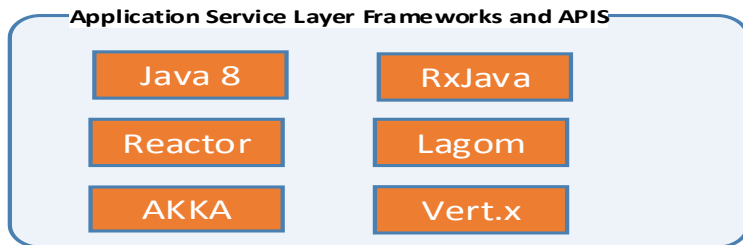
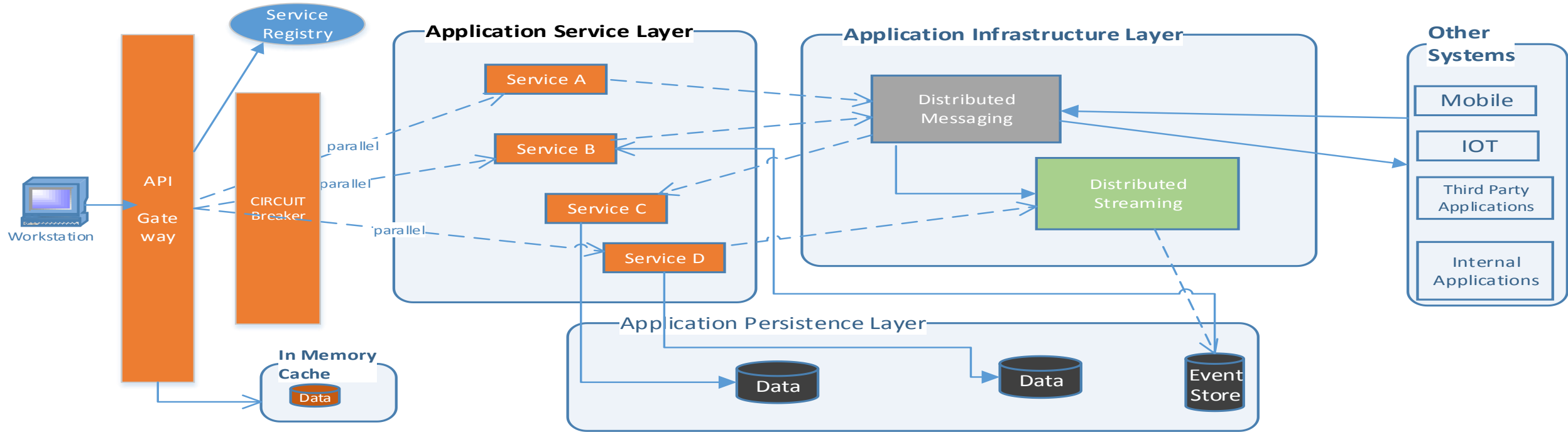
- KAFKA
- Others
 - zeroMQ

Distributed Database Management

- Cassandra
- Others
 - MongoDB
 - HBase



Typical Reactive Architecture



Thank You



MODS

**MOBILE & DISRUPTIVE
TECHNOLOGY SUMMIT**

October 5-6, 2017
IISc, Bangalore

www.modsummit.com

Register early and get the best discounts

April 23-28, 2018
IISc, Bangalore

www.developersummit.com

GREAT INDIAN
DEVELOPER
SUMMIT



PERTM
2018