Serverless Platforms

QCon New York

Diptanu Choudhury
@diptanu

The GOES Mission

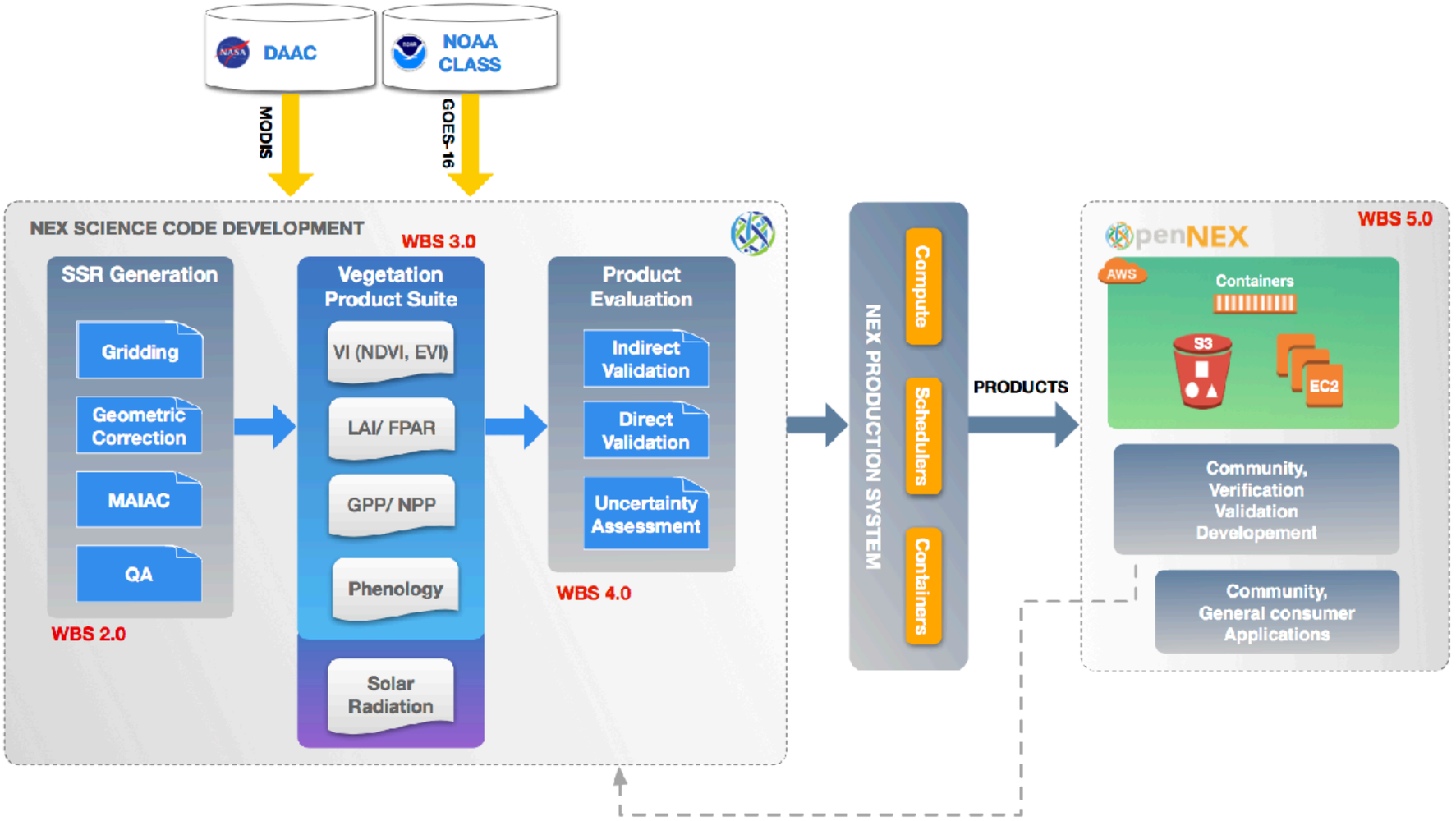
A lineage of geostationary satellites

Current mission is called GOES-R

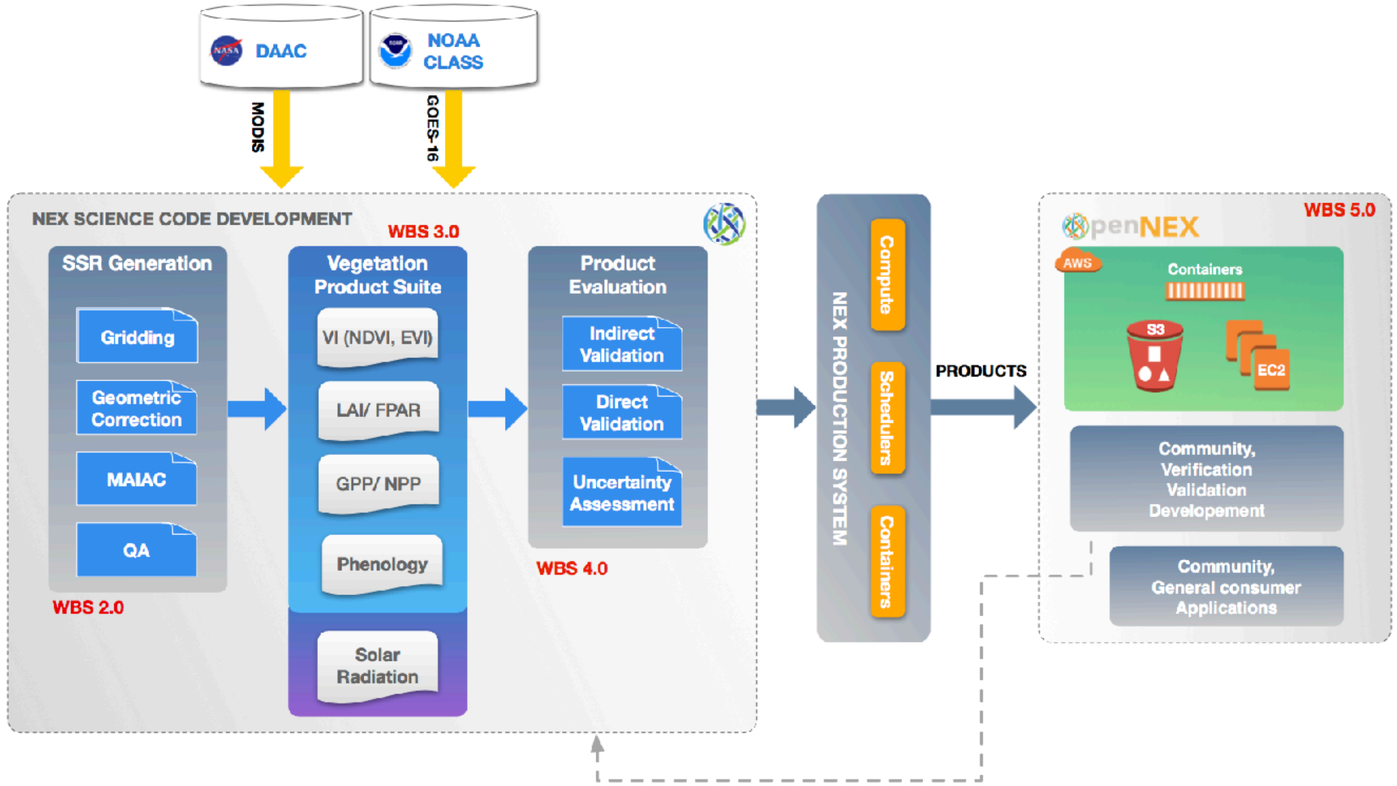
ABI views earth with 16 different spectral bands

GOES-R Products

- Atmospheric events
- Fire control
- Vegetation Monitoring
- NASA makes most of the data available to research community



Product Development Workflow



Iterative Experimentation based Model Development

Deployment of Models in Production and Image Processing

Data serving and Collaboration APIs

Product Development Workflow

Science

- Plans mission objectives
- Develop models and products
- Conducts research and drives collaboration

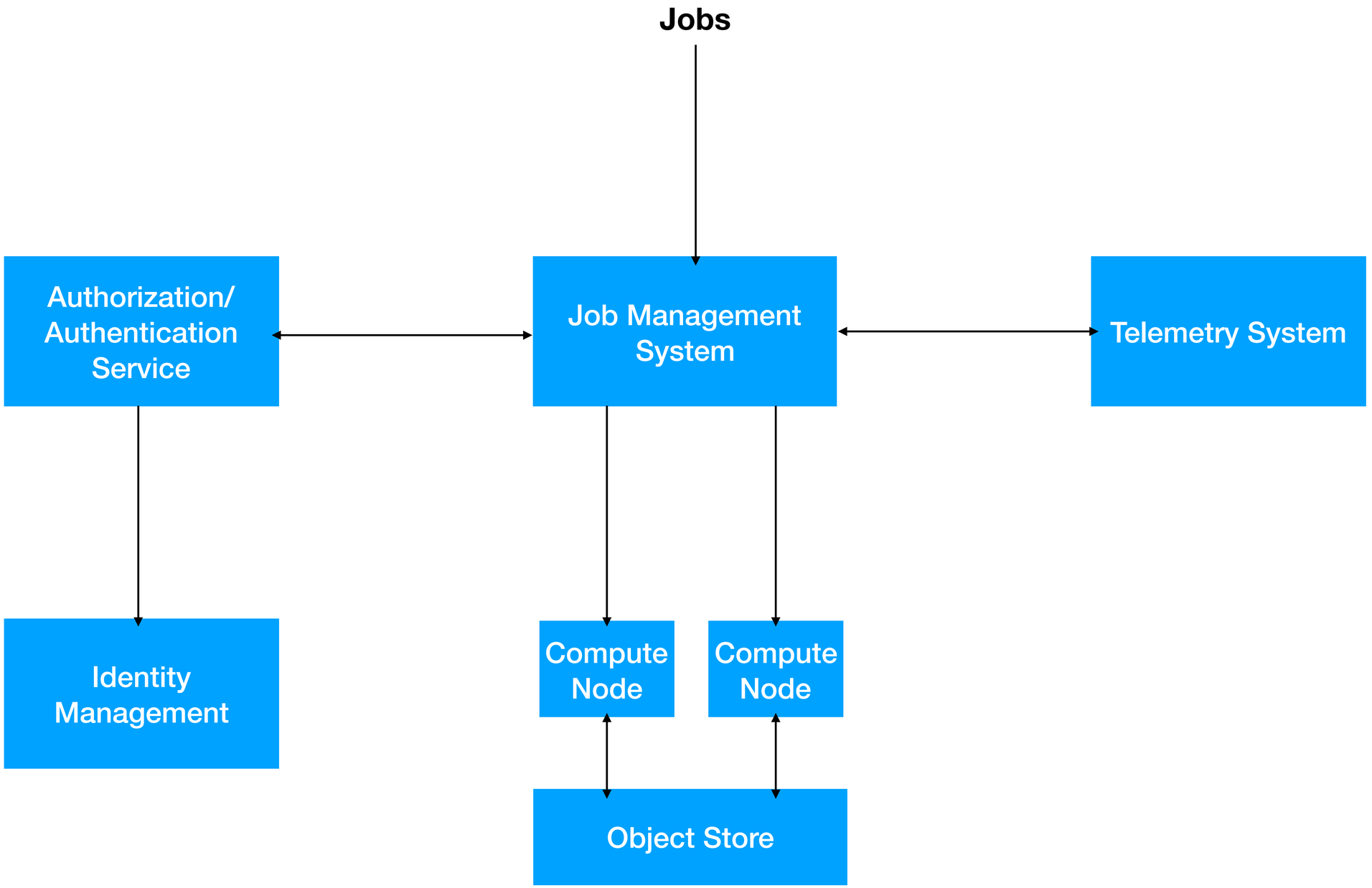
Engineering

- Operates infrastructure to run models
- Design systems to scale to petabytes of data and 10s of 1000s of cores.
- Design real time services for services to consume data
- Dev-Infra to improve productivity of researchers

The organization

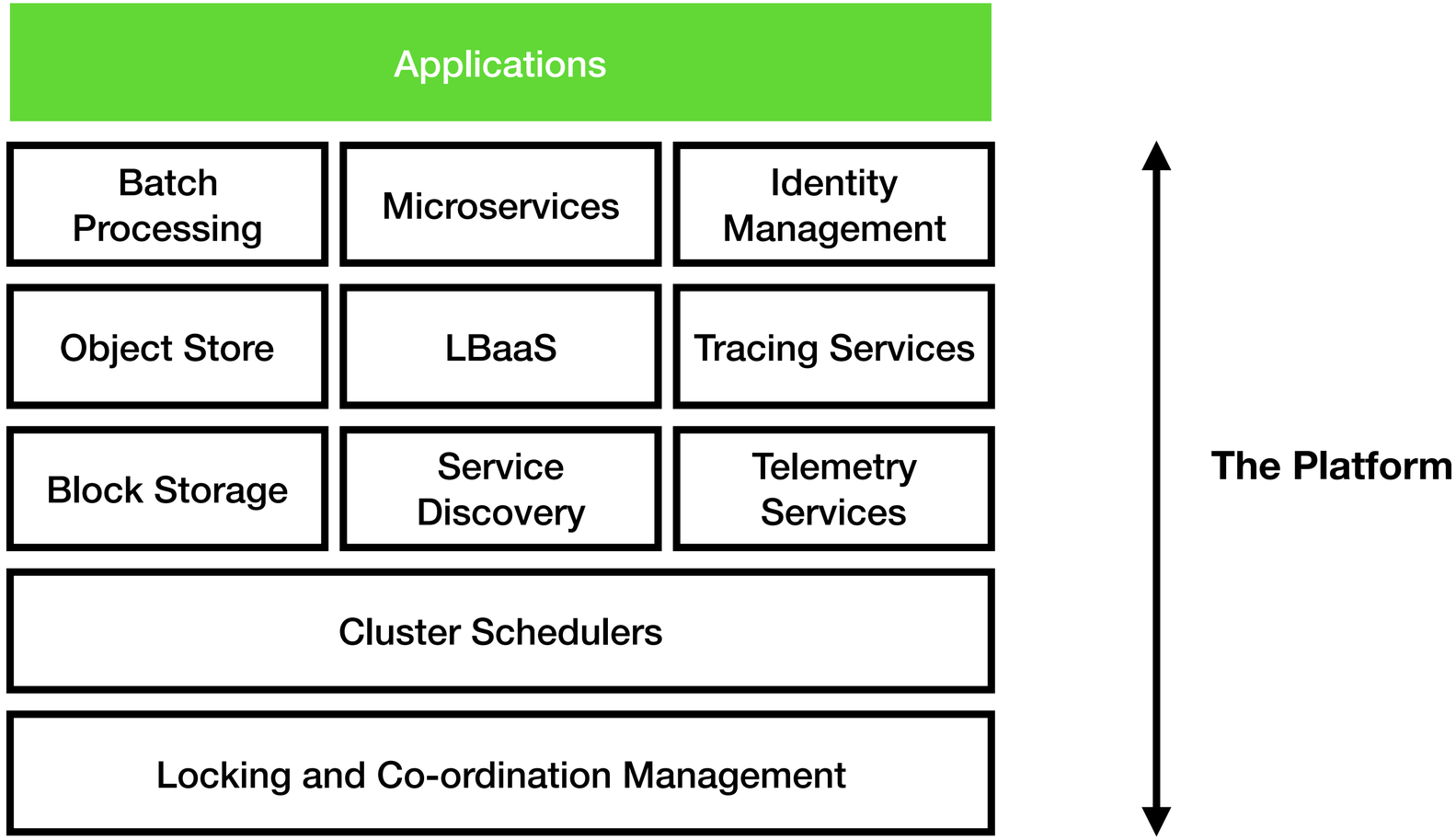
Infrastructure

- Cluster Schedulers that can scale to 10s of 1000s of cores
- Storage services that can scale to petabytes of data
- Interactive batch processing systems
- Services to stream real time information
- APIs to access data from various transformation jobs

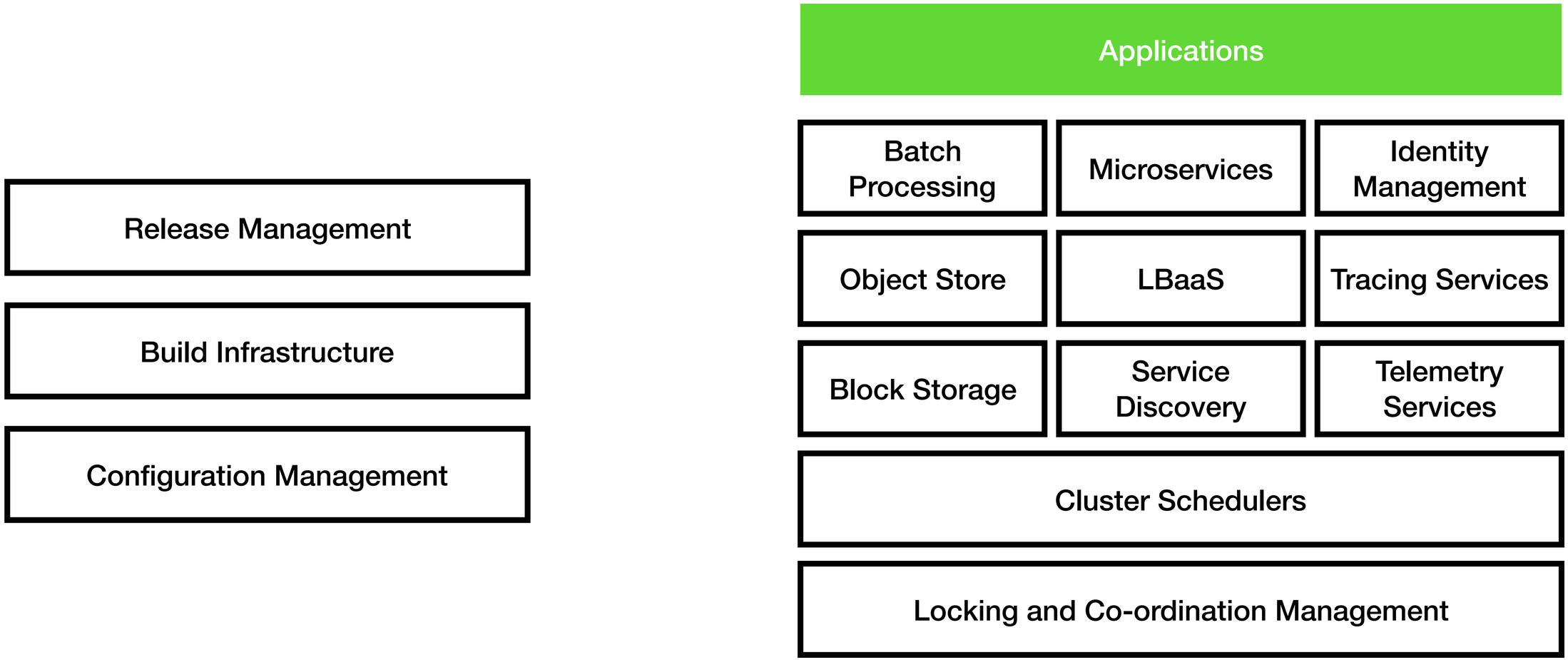

Jobs
Authorization/
Authentication
Service
Job Management
System
Telemetry System
Identity
Management
Compute
Node
Compute
Node
Object Store

There's Just No Getting around It:
You're Building a Distributed System

Infrastructure Stack of Circa 2017

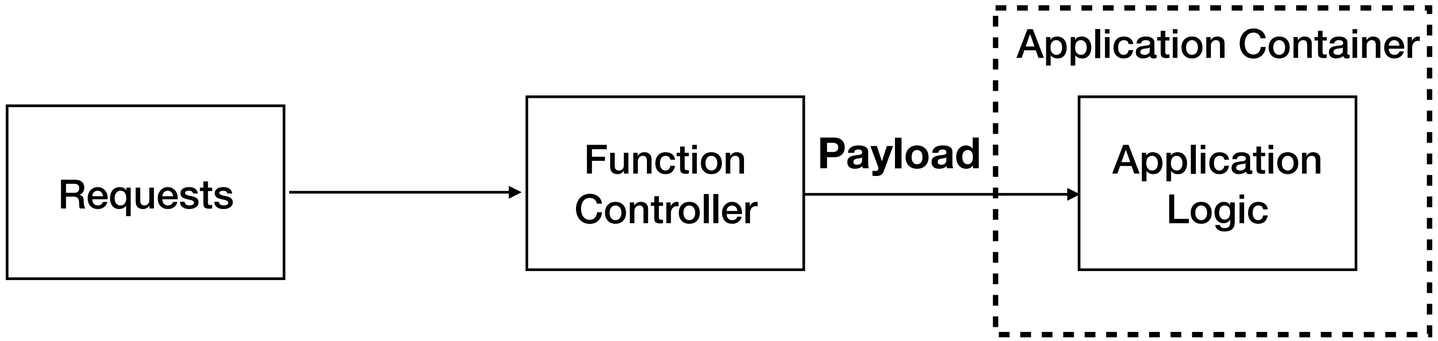


Platforms as a Service



Serverless Architecture

Serverless Architecture



Serverless Architecture

- Incremental evolution over PAAS
- *Serverless* from the perspective of users
- Clear separation of concerns between infrastructure engineering and application developers

Serverless Platform

- Optimistically concurrent Cluster Scheduler
- Application Container
- Function Controller
- RPC Middleware
- Function Invocation shims in data sources
- Packaging and distribution service

Cluster Scheduler

- Low latency and high throughput scheduling
- No head of line blocking
- Optimistically concurrent
- Scale up to a large number of containers

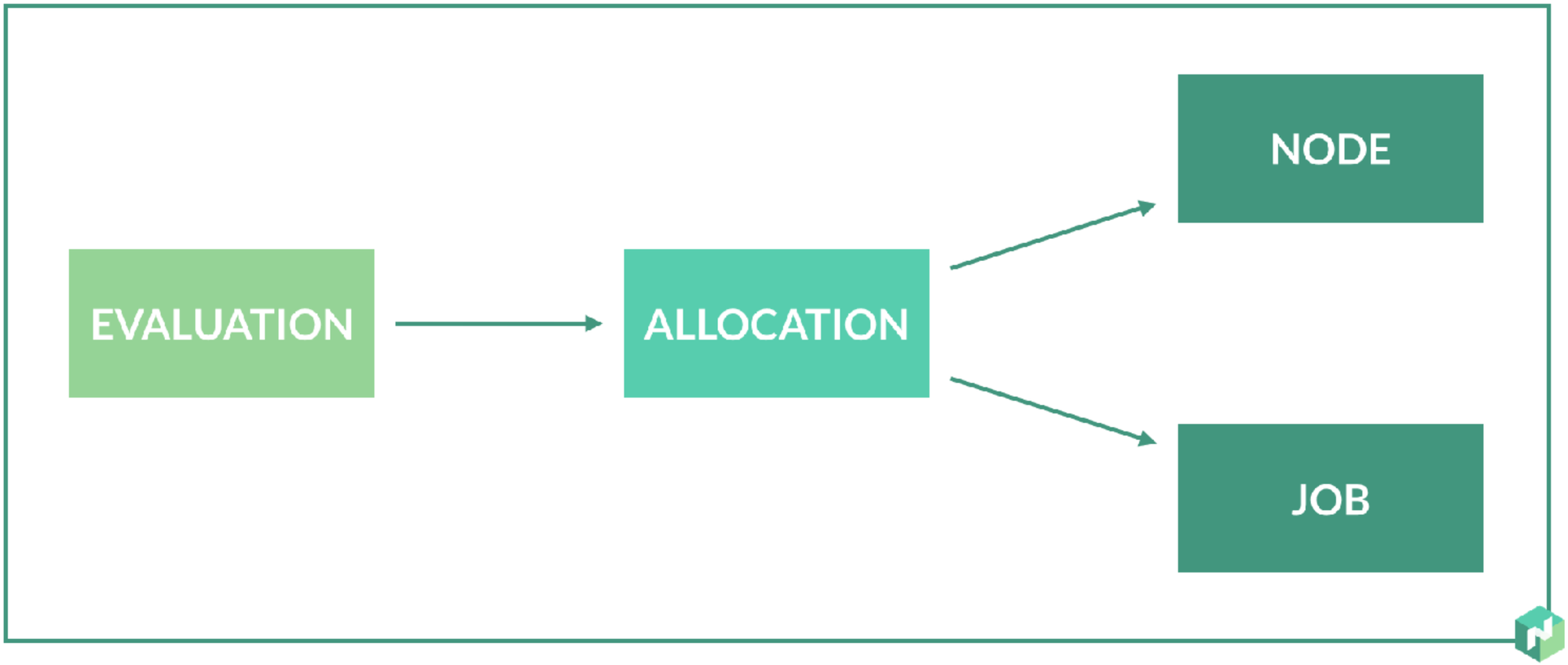
THE PATH TO PRODUCTION FOR A SCHEDULER IS FILLED WITH PAIN

CHOOSE AN EXISTING SCHEDULER IF YOU CAN

Event Driven Scheduler

Evaluations ~= State Change Event

Data Model



External Event

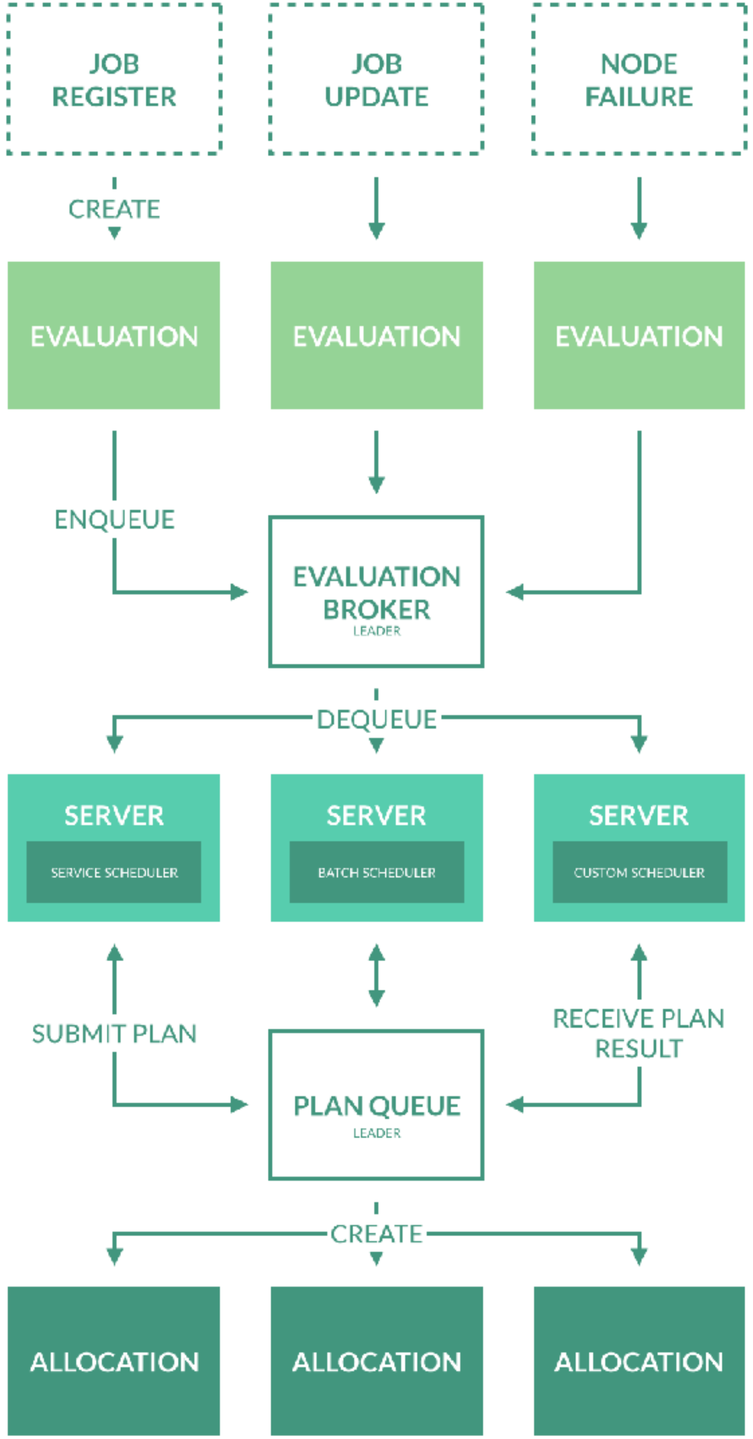
Evaluation Creation

Evaluation Queuing

Evaluation Processing

Optimistic Coordination

State Updates



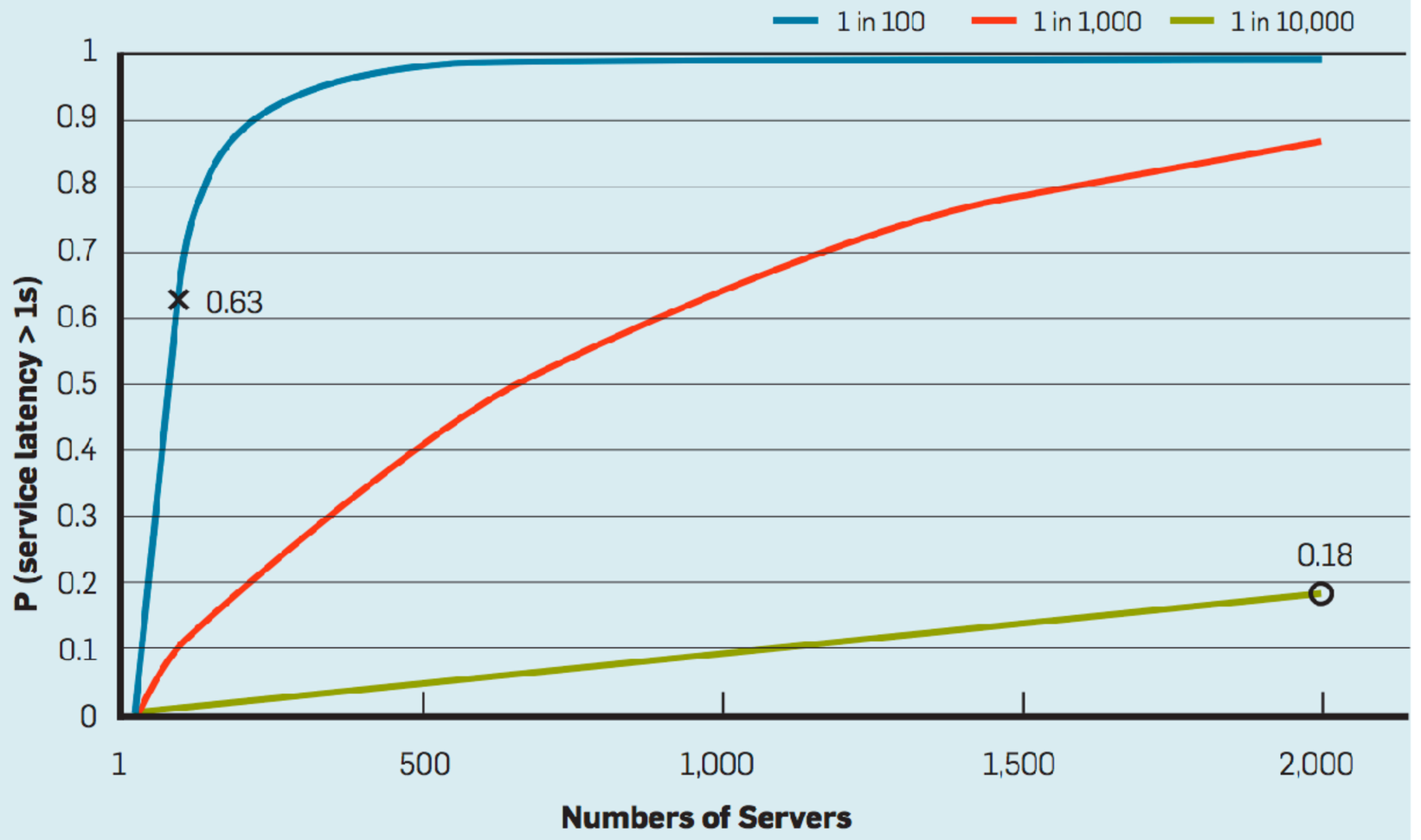
Templated Jobs

- Can be used for low throughput use cases
- Expensive to start new processes
- Bounded by network operations of scheduler

Variability

- Sharing resources
- Daemons
- Global resources
- Maintainance activities
- Queuing
- Power Limits

Probability of one-second service-level response time as the system scales and frequency of server-level high-latency outliers varies.



Source: Tale at Scale

Reduce Variability

- Differentiate service classes
- Break down long running functions into small ones.
- Minimize background activity

Application Containers

- Application containers range from language sandboxes to full blown containers
- Reduced surface area helps in making the UX better
- Application containers have function invocation middleware which are packaged during build

Function Controller

- Maintains the network topology of functions
- Optimizes for low latency and not cost
- Works with the scheduler to scale containers
- Works with telemetry system to scale underlying clusters

Software Load Balancer

- Service discovery aware LB
- Reactive Socket protocol for delivery of request payloads to function containers
- Doesn't effect container life cycles
- Drops requests which doesn't have any containers associated

Reactive Sockets

- RSocket is a means for asynchronous communication between functions and data sources
- Messages are multiplexed over a single connection
- Flexible interaction models
- Provides mechanisms for back pressure and request cancellation

RPC Middleware

- RPC implementation moved into the platform
- Hedged requests to reduce tail latency
- Tied requests to reduce mean and tail latency
- Heavy use of back pressure

Cluster Topology

- Scheduler chooses the cluster topology for containers
- Function Controller can change the topology based on real time latency
- Functions which are chained needs to be placed as close as possible

Caching

- Various forms of caching
- Depends on life cycle management of functions
- External caches like Redis, Memcached works best
- Hard to use in-memory clustered caches like Groupcache

External Data Sources

- Functions can read from external data sources just like traditional applications
- Avoid polling external data sources
- Data Sources like Kafka could co-operate with functions by a shim which calls functions with data mutations

Build and Distribution

- Packaging is part of the server less experience
- Packaging pipeline should transform a function to an application container.
- Unit Testing should be much easier with Functions than they were for traditional applications
- Performance of the platform can be improved by caching containers on compute nodes

Operations in the world of Serverless

- Reduced amount of operations for developers
- Metrics related to various concerns of the platform should be very targetted.

Application metrics

- Latency at the API Gateway level
- Throughput of events processed
- Latency distribution and planning.

Platform Metrics

- Latency of function invocation
- Throughput of events dispatched to the platform
- Number of active functions